目录

1 实验目标与基本要求

1.1 实验目标

重点结合航空公司客户价值分析的案例介绍 K-Means 聚类算法在客户价值分析中的应用。此外，介绍基于 RFM 客户价值分析模型的不足，使用 K-Means 算法构建航空客户价值分析 LRFMC 模型，详细的描述数据分析的整个过程。

1.2 基本要求

1. 了解 RFM 模型的基本原理。
2. 掌握 K-Means 算法的基本原理与使用方法。
3. 比较不同类别客户的客户价值，制定相应的营销策略。

2 主要知识点、重点与难点

2.1 主要知识点

1. RFM 模型的基本原理
2. K-Means 算法的基本原理与使用方法
3. 比较不同类别客户的客户价值，制定相应的营销策略

2.2 重点

（1）航空客户价值分析的步骤和流程。
（2）RFM 模型的基本原理。
（3）K-Means 算法的基本原理与使用方法。
（4）比较不同类别客户的客户价值。

2.3 难点

（1）RFM 模型的基本原理。

（2）KMeans 算法的基本原理与使用方法。

2.4 主要知识点讲解

2.4.1 RFM 模型

RFM 模型由三个基础指标组成：

- 最近一次消费（Recency）：客户最近一次消费的时间间隔。间隔越短，表示客户价值越高。
- 消费频率（Frequency）：客户在最近一段时间内消费的次数。次数越多，表示客户价值越高。
- 消费金额（Monetary）：客户在最近一段时间内消费的金额。金额越多，表示客户价值越高。

在 RFM 模型里，三个变量的含义都是很具体的。

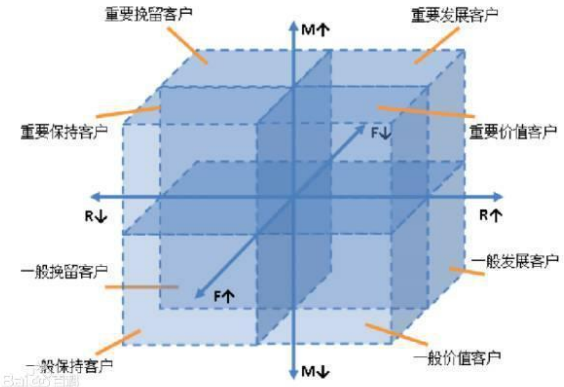


图 1: RFM 模型示意图

但是，RFM 模型存在着一个明显的缺陷，那就是没有关注客户的消费周期和消费行为，比如说，作为航空公司，不知道客户是坐头等舱和经济舱，这就催生了 LRFMC 模型的出现。

2.4.2 LRFMC 模型

传统的 RFM 模型它是依据各个属性的平均值进行划分，但是，细分的客户群太多，精准营销的成本太高。因此 LRFMC 模型将客户聚类为重要保持客户，重要发展客户，重要挽留客户，一般客户，低价值客户，从而针对每种类别的客户制定对应的价格和服务。

一下是 LRFMC 模型的具体指标：

简称	含义	意义	关系
L	入会至当前时间的间隔	反映可能的活跃时长	观测窗口结束的时间-入会时间
R	最近消费时间据当前的间隔	反映当前的活跃状态	客户最近一次乘坐公司飞机距观测窗口结束的月数
F	乘机次数	反映客户的忠诚度	观测窗口的飞行次数
M	飞行里程数	反映客户对乘机的依赖性	观测窗口的总飞行公里数
C	舱位等级对应的折扣系数	侧面反映客户价值高低	折扣率

图 2: LRFMC 模型指标

2.4.3 K-Means 算法

作为数据挖掘的一个重要研究课题，聚类分析技术越来越受到人们的关注。聚类就是将物理或抽象对象的集合分成多个相似的数据子集，同一个子集内的对象之间具有较高的相似度，而不同子集内的对象差别较大。经过专家学者们的研究，目前聚类算法可以归纳为如下几类：基于划分的方法、基于密度的方法、基于层次的方法、基于模型的方法和高维数据的方法。K-means 算法作为一种基于划分的动态聚类算法, 它以误差平方和 SSE 作为聚类准则函数，具有简单有效、收敛速度较快、便于处理大型数据集等优点，从而获得了广泛的应用。

简单总结起来就是：初始化聚类中心、样本点划分、更新聚类中心、样本点划分、更新聚类中心.... 直至收敛即可。

(1) 数据预处理。主要是标准化、异常点过滤。

(2) 随机选取K个中心，记为 $\mu_1^{(0)}, \mu_2^{(0)}, \ldots, \mu_k^{(0)}$

(3) 定义损失函数： $J(c, \mu) = min \sum_{i=1}^{M} ||x_i - \mu_{c_i}||^2$

(4) 令t=0,1,2,... 为迭代步数，重复如下过程知道 J 收敛:

(4.1) 对于每一个样本 x_i ，将其分配到距离最近的中心

$$c_i^t <— argmin_k ||x_i - \mu_k^t||^2$$

(4.2) 对于每一个类中心k，重新计算该类的中心

$$\mu_k^{(t+1)} <— argmin_\mu \sum_{i:c_i^t=k}^{b} ||x_i - \mu||^2$$

图 3: K-Means 算法步骤

3 实验过程设计

3.1 实验理论

（1）分析航空公司现状。
（2）认识客户价值分析。
（3）熟悉航空客户价值分析的步骤与流程。
（4）处理缺失值与异常值。
（5）构建爱你航空客户价值分析关键特征。
（6）标准化 LRFMC 5 个特征。
（7）了解 K-Means 聚类算法。
（8）分析聚类结果。
（9）模型应用。

3.2 实验教学过程

（1）处理数据缺失值与异常值。
（2）构建航空客户价值分析的关键特征。
（3）标准化 LRFMC 5 个特征。
（4）构建 K-Means 聚类模型。
（5）评价 K-Means 聚类模型。

4 主要实现过程

4.1 数据预处理

4.1.1 数据清洗

数据清洗是数据挖掘的第一步，也是最重要的一步。数据清洗的目的是检查数据是否有错误，以及是否有对于分析过程无用的数据。数据清洗的主要任务是填补缺失值和处理异常值。

首先调用 describe () 函数将 air_data.csv 的文件导入，并分别查看数据的缺失值和异常值。函数的输出是两个文件，一个是数据描述.csv，记录各个属性的数据描述；另一个是空值项数量.csv，记录各个属性的空值项数量。

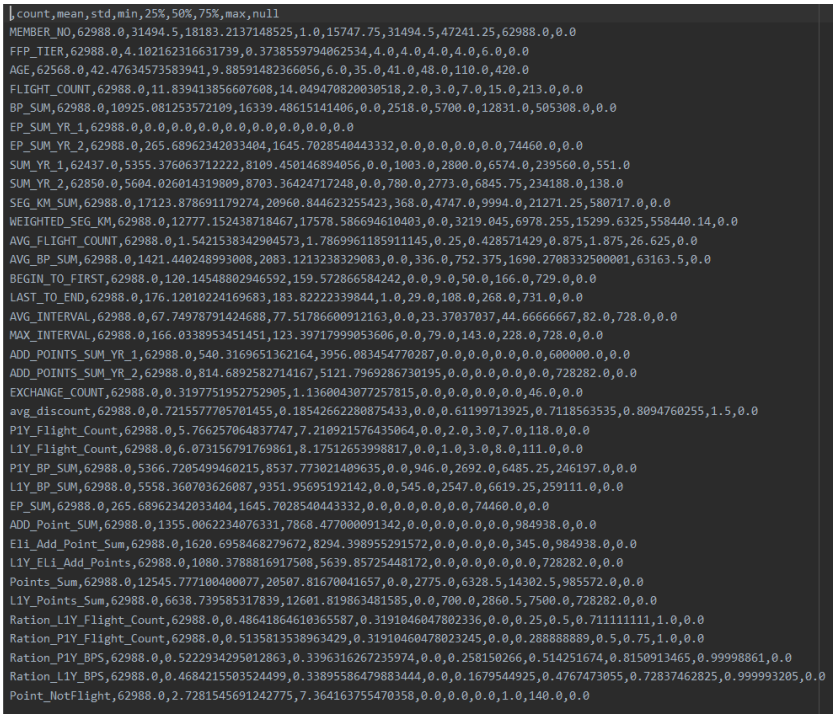
describe () 函数的代码如下：

```
# 查看数据的整体情况和缺失值
def describe():
    airline_data = pd.read_csv("../data/air_data.csv",
                               encoding="gb18030")  # 导入航
                                  空数据
    explore = airline_data.describe().T
    explore['null'] = len(airline_data) - explore['count']
    explore.to_csv('数据描述.csv')
    df = explore[['max', 'min', 'null']]
    nullData = df[df['null'] > 0]
```

```
        nullData = nullData['null']
11      nullData.to_csv('空值项数量.csv')
        return airline_data
```

describe () 函数

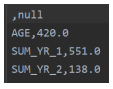
运行程序之后，得到两个 csv 文件，分别是数据描述.csv 和空值项数量.csv。数据描述.csv 的内容如下：



```
,count,mean,std,min,25%,50%,75%,max,null
MEMBER_NO,62988.0,31494.5,18183.2137148525,1.0,15747.75,31494.5,47241.25,62988.0,0.0
FFP_TIER,62988.0,4.102162316631739,0.3738559794062534,4.0,4.0,4.0,4.0,6.0,0.0
AGE,62568.0,42.47634573583941,9.88591482366056,6.0,35.0,41.0,48.0,110.0,420.0
FLIGHT_COUNT,62988.0,11.839413856607608,14.049470820030518,2.0,3.0,7.0,15.0,213.0,0.0
BP_SUM,62988.0,10925.081253572109,16339.48615141406,0.0,2518.0,5700.0,12831.0,505308.0,0.0
EP_SUM_YR_1,62988.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
EP_SUM_YR_2,62988.0,265.68962342033404,1645.7028540443332,0.0,0.0,0.0,0.0,74460.0,0.0
SUM_YR_1,62437.0,5355.376063712222,8109.450146894056,0.0,1003.0,2800.0,6574.0,239560.0,551.0
SUM_YR_2,62850.0,5604.026014319809,8703.36424717248,0.0,780.0,2773.0,6845.75,234188.0,138.0
SEG_KM_SUM,62988.0,17123.878691179274,20960.844623255423,368.0,4747.0,9994.0,21271.25,580717.0,0.0
WEIGHTED_SEG_KM,62988.0,12777.152438718467,17578.586694610403,0.0,3219.045,6978.255,15299.6325,558440.14,0.0
AVG_FLIGHT_COUNT,62988.0,1.5421538342904573,1.7869961185911145,0.25,0.428571429,0.875,1.875,26.625,0.0
AVG_BP_SUM,62988.0,1421.440248993008,2083.1213238329083,0.0,336.0,752.375,1690.2708332500001,63163.5,0.0
BEGIN_TO_FIRST,62988.0,120.14548802946592,159.572866584242,0.0,9.0,50.0,166.0,729.0,0.0
LAST_TO_END,62988.0,176.12010224169683,183.82222339844,1.0,29.0,108.0,268.0,731.0,0.0
AVG_INTERVAL,62988.0,67.74978791424688,77.51786600912163,0.0,23.37037037,44.66666667,82.0,728.0,0.0
MAX_INTERVAL,62988.0,166.0338953451451,123.39717999053606,0.0,79.0,143.0,228.0,728.0,0.0
ADD_POINTS_SUM_YR_1,62988.0,540.3169651362164,3956.083454770287,0.0,0.0,0.0,0.0,600000.0,0.0
ADD_POINTS_SUM_YR_2,62988.0,814.6892582714167,5121.7969286730195,0.0,0.0,0.0,0.0,728282.0,0.0
EXCHANGE_COUNT,62988.0,0.3197751952752905,1.1360043077257815,0.0,0.0,0.0,0.0,46.0,0.0
avg_discount,62988.0,0.7215577705701455,0.18542662280875433,0.0,0.61199713925,0.7118563535,0.8094760255,1.5,0.0
P1Y_Flight_Count,62988.0,5.766257064837747,7.210921576435064,0.0,2.0,3.0,7.0,118.0,0.0
L1Y_Flight_Count,62988.0,6.073156791769861,8.175126539988l7,0.0,1.0,3.0,8.0,111.0,0.0
P1Y_BP_SUM,62988.0,5366.7205499460215,8537.773021409635,0.0,946.0,2692.0,6485.25,246197.0,0.0
L1Y_BP_SUM,62988.0,5558.360703626087,9351.95695192142,0.0,545.0,2547.0,6619.25,259111.0,0.0
EP_SUM,62988.0,265.68962342033404,1645.7028540443332,0.0,0.0,0.0,0.0,74460.0,0.0
ADD_Point_SUM,62988.0,1355.0062234076331,7868.477000091342,0.0,0.0,0.0,0.0,984938.0,0.0
Eli_Add_Point_Sum,62988.0,1620.69584682796 72,8294.398955291572,0.0,0.0,0.0,345.0,984938.0,0.0
L1Y_ELi_Add_Points,62988.0,1080.3788816917508,5639.85725448172,0.0,0.0,0.0,0.0,728282.0,0.0
Points_Sum,62988.0,12545.777100400077,20507.81670041657,0.0,2775.0,6328.5,14302.5,985572.0,0.0
L1Y_Points_Sum,62988.0,6638.739585317839,12601.819863481585,0.0,700.0,2860.5,7500.0,728282.0,0.0
Ration_L1Y_Flight_Count,62988.0,0.48641864610365587,0.3191046047802336,0.0,0.25,0.5,0.711111111,1.0,0.0
Ration_P1Y_Flight_Count,62988.0,0.5135813538963429,0.3191046047802324 5,0.0,0.288888889,0.5,0.75,1.0,0.0
Ration_P1Y_BPS,62988.0,0.5222934295012863,0.3396316267235974,0.0,0.258150266,0.514251674,0.8150913465,0.99998861,0.0
Ration_L1Y_BPS,62988.0,0.4684215503524499,0.33895586479883444,0.0,0.1679544925,0.4767473055,0.72837462825,0.999993205,0.0
Point_NotFlight,62988.0,2.7281545691242775,7.364163755470358,0.0,0.0,0.0,1.0,140.0,0.0
```

图 4: 数据描述.csv

空值项数量.csv 的内容如下：



```
,null
AGE,420.0
SUM_YR_1,551.0
SUM_YR_2,138.0
```

图 5: 空值项数量.csv

可以看到其中 AGE 缺失 420 条数据，SUM_YR_1 缺失 551 条数据，SUM_YR_2 缺失 138 条数据。这些缺失值的存在会影响到后续的数据分析，因此需要对缺失值进行处理，本实验中由于样本数量比较庞大，因此删除少量数据不会影响到整体的分析结果，因此采用删除缺失值的方法进行处理。

利用 notnull()函数筛选出 air_data.csv 中不为空值的行,将 SUM_YR_1 和 SUM_YR_2 中的非空值做与运算,再通过 loc()函数筛选出 airline_notnull，实现上述操作的代码如下：

```
# 删除缺失值和异常值
def data_cleaning(airline_data):
    exp1 = airline_data["SUM_YR_1"].notnull()
    exp2 = airline_data["SUM_YR_2"].notnull()
    exp = exp1 & exp2
    print(exp)
    print('exp的形状是：', exp.shape)
    airline_notnull = airline_data.loc[exp, :]
    print('删除缺失记录后数据的形状为：', airline_notnull.
        shape)
```

删除缺失值

得到删除缺失值之后数据的结果：

```
0         True
1         True
2         True
3         True
4         True
         ...
62983     True
62984     True
62985     True
62986     True
62987    False
Length: 62988, dtype: bool
exp的形状是： (62988,)
删除缺失记录后数据的形状为： (62299, 44)
```

图 6: 删除缺失值后的数据

得到的删除缺失值后的数据的形状为（62299，44），即删除了缺失值后

的数据集中还剩下 62299 条数据。但是这些数据中仍然存在异常值，因此需要对异常值进行处理。

需要丢弃第一年或者第二年票价为 0，平均折扣不为 0 同时总飞行公里数大于 0 的记录。在 airline_notnull 的基础上进行操作，并将预处理完的数据返回，操作代码如下：

```
# 删除缺失值和异常值
def data_cleaning(airline_data):
    exp1 = airline_data["SUM_YR_1"].notnull()
    exp2 = airline_data["SUM_YR_2"].notnull()
    exp = exp1 & exp2
    # print(exp)
    # print('exp的形状是：', exp.shape)
    airline_notnull = airline_data.loc[exp, :]
    print('删除缺失记录后数据的形状为：', airline_notnull.
        shape)

    # 只保留票价非零的，或者平均折扣率不为0且总飞行公里数大
        于0的记录。
    index1 = airline_notnull['SUM_YR_1'] != 0
    index2 = airline_notnull['SUM_YR_2'] != 0
    index3 = (airline_notnull['SEG_KM_SUM'] > 0) & \
             (airline_notnull['avg_discount'] != 0)
    airline = airline_notnull[(index1 | index2) & index3]
    print('删除异常记录后数据的形状为：', airline.shape)
    return airline
```

删除缺失值 + 处理异常值

删除缺失值和异常值之后数据的形状为：

```
删除缺失记录后数据的形状为：  (62299, 44)
删除异常记录后数据的形状为：  (62044, 44)
```

图 7: 删除缺失值 + 处理异常值后的数据

4.1.2 数据筛选

根据上面的 LRFMC 模型，只有第一个特征量 L 是需要进行计算的，通过 to_datetime () 函数进行时间间隔的计算，然后强制转换成字符串类型，选取其中的天数，再通过天数获得月份，最后连接上原始数据集，得到最终筛选好的的数据集，代码如下：

```
    # 数据筛选
def data_selection(airline):
    # 选取需求特征
    """
    L: 入会时间
    FLIGHT_COUNT: F 乘机次数
    LAST_TO_END: R 最近消费次数
    avg_discount: C 折扣率
    SEG_KM_SUM: M 飞行里程数
    """
    airline_selection = airline[["FFP_DATE", "LOAD_TIME",
                                 "FLIGHT_COUNT", "LAST_TO_END",
                                 "avg_discount", "SEG_KM_SUM"]]
    L = pd.to_datetime(airline_selection["LOAD_TIME"]) - \
        pd.to_datetime(airline_selection["FFP_DATE"])
    L = L.astype("str").str.split(' ').str[0]
    # 获得会员入会的月数
    L = L.astype("int") / 30
    # 合并特征
    airline_features = pd.concat([L, airline_selection.iloc[:,
        2:]], axis=1)
    return airline_features
```

数据筛选

构建的 LRFMC 特征前五的行为如下：

```
构建LRFMC特征前五行为：
           0  FLIGHT_COUNT  LAST_TO_END  avg_discount  SEG_KM_SUM
0  90.200000           210            1      0.961639      580717
1  86.566667           140            7      1.252314      293678
2  87.166667           135           11      1.254676      283712
3  68.233333            23           97      1.090870      281336
4  60.533333           152            5      0.970658      309928
```

图 8: LRFMC 特征前五的行

4.1.3 数据标准化处理

对完成上述数据处理完之后的数据集进行标准化处理。函数中使用 sklearn 的 StandardScaler 对"airline_features" 数据进行标准化，并将结果保存到全局变量"data" 中。代码如下：

```
    # 数据标准化
2   def data_normalization(airline_features):
        global data
4       data = StandardScaler().fit_transform(airline_features)
        np.savez('../tmp/airline_scale.npz', data)
6       print('标准化后LRFMC五个特征为：\n', data[:5, :])
```

数据标准化处理

标准化后的 LRFMC 五个特征为：

```
标准化后LRFMC五个特征为：
 [[ 1.43571897 14.03412875 -0.94495516  1.29555058 26.76136996]
 [ 1.30716214  9.07328567 -0.9119018   2.86819902 13.1269701 ]
 [ 1.32839171  8.71893974 -0.88986623  2.88097321 12.65358345]
 [ 0.65848092  0.78159082 -0.41610151  1.99472974 12.54072306]
 [ 0.38603481  9.92371591 -0.92291959  1.3443455  13.89884778]]
```

图 9: 标准化后的 LRFMC 五个特征

4.2 数据分析

4.2.1 读取数据

read_data () 函数从保存的文件中加载经过预处理和标准化的航空数据。

```
def read_data():
    airline_scale = np.load('../tmp/airline_scale.npz')['
        arr_0']
    return airline_scale
```

读取数据

4.2.2 K-Means 聚类分析

cluster () 函数使用 scikit-learn 中的 K-Means 算法对航空数据进行 K 均值聚类。它将聚类中心数（k）设置为 5，拟合模型并打印聚类中心和标签。

```
# 聚类数据分析
def cluster(airline_scale):
    k = 5   # 聚类中心数
    kmeans_model = KMeans(n_clusters=k, n_init=4,
        random_state=666)
    kmeans_model.fit(airline_scale)  # 模型训练
    print('聚类中心:\n', kmeans_model.cluster_centers_)  #
        查看聚类中心
    print('类别标签:\n', kmeans_model.labels_)  # 查看样本的
        类别标签
    # 统计不同类别样本的数目
    r1 = pd.Series(kmeans_model.labels_).value_counts()
    print('最终每个类别的数目为: \n', r1)
    return kmeans_model
```

K-Means 聚类分析

模型训练后，得到的聚类中心、样本类别标签以及每个类别的数量：

```
聚类中心:
 [[-0.31565422 -0.57399354  1.68514055 -0.16503129 -0.53693112]
 [ 0.48263385  2.47791197 -0.79880852  0.30050789  2.41983256]
 [-0.70068394 -0.16389374 -0.41195274 -0.23954974 -0.16468602]
 [ 0.09999032 -0.19885942 -0.01303134  2.3086448  -0.19708344]
 [ 1.15602342 -0.09166004 -0.37406447 -0.15300522 -0.09967427]]
类别标签:
 [1 1 1 ... 2 0 0]
最终每个类别的数目为:
 2    24888
4    15828
0    12178
1     5339
3     3811
```

图 10: K-Means 聚类分析结果

4.3 聚类模型评价

4.3.1 FMI 评价法

Fowlkes-Mallows 指数是一种外部评估方法，用于确定两个聚类（通过聚类算法获得的聚类）之间的相似性，也是衡量混淆矩阵的指标。这种相似性度量可以是两个层次聚类之间的相似性，也可以是一个聚类和一个基准分类之间的相似性。FMI 的取值范围为 [0,1]，值越大，聚类结果越好。FMI 的计算公式如下：

$$FM = \sqrt{PPV \cdot TPR} = \sqrt{\frac{TP}{TP+FP} \cdot \frac{TP}{TP+FN}}$$

函数使用 scikit-learn 的 fowlkes_mallows_score 函数，计算不同聚类数（2 到 6）下的 Fowlkes-Mallows 指数（FMI）。代码如下：

```
# FMI评价法
def FMI(airline_scale, kmeans_model):
```

```
    for i in range(2, 7):
        kmeans = KMeans(n_clusters=i, random_state=123).fit(
            airline_scale)
        print(kmeans.labels_)
        print(type(kmeans.labels_))
        score = fowlkes_mallows_score(kmeans_model.labels_,
            kmeans.labels_)
        print('数据聚%d类FMI评价分值为：%f' % (i, score))
```

FMI 评价法

展示结果如下：

```
[1 1 1 ... 0 0 0]
<class 'numpy.ndarray'>
数据聚2类FMI评价分值为：0.532749
[0 0 0 ... 2 1 1]
<class 'numpy.ndarray'>
数据聚3类FMI评价分值为：0.651823
[3 3 3 ... 2 1 2]
<class 'numpy.ndarray'>
数据聚4类FMI评价分值为：0.920015
[1 1 1 ... 4 3 3]
<class 'numpy.ndarray'>
数据聚5类FMI评价分值为：0.986579
[2 2 2 ... 5 3 5]
<class 'numpy.ndarray'>
数据聚6类FMI评价分值为：0.850608
```

图 11: FMI 评价法结果

由运算结果得到，数据聚 5 类的 FMI 评价分值最高，为 0.986579，说明聚类效果最好。但是由于 FMI 评价方法需要有明确的标签值，但是本题没有标签（无监督学习），所以使用 FMI 评价方法计算出来的分数并不具有参考价值。

4.3.2 轮廓系数（Silhouette Coefficient）评价

轮廓系数，是用于评价聚类效果好坏的一种指标。可以理解为描述聚类后各个类别的轮廓清晰度的指标。其包含有两种因素——内聚度和分离度。

内聚度可以理解为反映一个样本点与类内元素的紧密程度。

分离度可以理解为反映一个样本点与类外元素的紧密程度。

轮廓系数的公式如下：

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

其中 $a(i)$ 是样本 i 与同簇其他样本的平均距离，$b(i)$ 是样本 i 与最近其他簇的平均距离。

$a(i)$ 的计算公式是：

$$a(i) = \frac{1}{n-1} \sum_{j \neq i}^{n} distance(i, j)$$

$b(i)$ 的计算公式与 $a(i)$ 类似，是样本 i 与其他簇的平均距离。只不过需要遍历所有的簇，找到最近的那个簇。

所有样本 S_i 的轮廓系数的均值就是聚类结果的轮廓系数，定义为 S，是该聚类是否合理、有效的度量。聚类结果的轮廓系数的取值在 $[-1, 1]$ 之间，值越大，说明同类样本相距越近、不同样本相距越远，则聚类效果越好；当轮廓系数为负值时，说明聚类结果不合理。

计算轮廓系数的代码如下：

```
    # silhouetteScore相似度评价法
def SS(airline_scale):
    silhouetteScore = []
```

```
    for i in range(2, 7):
        kmeans = KMeans(n_clusters=i, random_state=123).fit(
            airline_scale)
        score = silhouette_score(airline_scale, kmeans.labels_)
        print('航空公司数据聚%d类silhouette评价分值为：%f' % (i,
            score))
        silhouetteScore.append(score)
    plt.figure(figsize=(10, 6))
    plt.plot(range(2, 7), silhouetteScore, linewidth=1.5,
        linestyle="-")
    plt.show()
```

轮廓系数评价

运行程序之后，得到 2 到 6 类的轮廓系数如下：

```
航空公司数据聚2类silhouette评价分值为：0.369902
航空公司数据聚3类silhouette评价分值为：0.267208
航空公司数据聚4类silhouette评价分值为：0.269582
航空公司数据聚5类silhouette评价分值为：0.277145
航空公司数据聚6类silhouette评价分值为：0.281438
```

图 12: 轮廓系数评价结果

在本题中，轮廓系数最大的是 2 类，此外根据评价结果绘制了轮廓系数与聚类数目的折线图，如下：

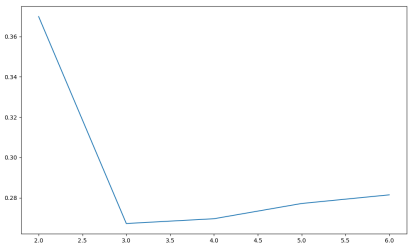


图 13: 轮廓系数评价结果

4.3.3 Calinski-Harabasz Index（CH）评价

Calinski-Harabasz 指数的本质是簇间距离与簇内距离的比值，且整体计算过程与方差计算方式类似，所以又将其称之为方差比准则。

将容量为 N 的数据集合 X 聚成 K 类，通过计算类内各点与类中心的距离平方和来度量类内的紧密度（类内距离），各个类中心点与数据集中心点距离平方和来度量数据集的分离度（类间距离）。

CH 指标的计算公式为：

$$s = \frac{tr(B_k)}{tr(W_k)} \cdot \frac{N-K}{K-1}$$

其中 $tr(B_k)$ 是类间距离，$tr(W_k)$ 是类内距离，N 是样本总数，K 是类别数。

B_k 和 W_k 的计算公式如下：

$$B_k = \sum_{i=1}^{k} n_i \cdot \|\mu_i - \mu\|^2$$

$$W_k = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

Calinski-Harabasz 指数的分数越大说明越好（类别内部协方差越小越好，类别之间协方差越大越好）。指数的取值范围为 $[0, +\infty)$，但是由于类别数目 K 的限制，所以指数的取值范围为 $[0, +\infty)$。

计算 CH 指数的代码如下：

```
# calinski_harabaz指数评价法
def CH(airline_scale):
    ch = []
    for i in range(2, 7):
        # 构建并训练模型
        kmeans = KMeans(n_clusters=i, random_state=123).fit(
            airline_scale)
        score = calinski_harabasz_score(airline_scale,
            kmeans.labels_)
```

```
            print('航空公司数据聚%d类calinski_harabaz指数为：%f'
                % (i, score))
            ch.append(score)
        plt.figure(figsize=(10, 6))
        plt.plot(range(2, 7), ch, linewidth=1.5, linestyle="-")
        plt.show()
```

CH 评价

运行程序之后，得到 2 到 6 类的 CH 指数如下：

```
航空公司数据聚2类calinski_harabaz指数为：21876.774212
航空公司数据聚3类calinski_harabaz指数为：21180.980723
航空公司数据聚4类calinski_harabaz指数为：21802.063722
航空公司数据聚5类calinski_harabaz指数为：20567.779167
航空公司数据聚6类calinski_harabaz指数为：20192.643014
```

图 14: CH 评价结果

在本题中，CH 指数最大的依然是 2 类。综合上述的三种评价方法，在聚类数的选择上选择聚类数为 3，但是由于 LRFMC 模型的特殊性，需要将客户划分成五类，所以最终选择聚类数为 5。

4.4 客户价值分析

4.4.1 降维图像分析

TSNE 的定位是高维数据可视化。对于聚类来说，输入的特征维数是高维的（大于三维），一般难以直接以原特征对聚类结果进行展示。而 TSNE 提供了一种有效的数据降维模式，是一种非线性降维算法，可以在 2 维的空间里展示聚类结果。

在本题中，可以将 LRFMC 模型的五个维度作为输入，然后将聚类结果进行可视化，代码如下：

```
    # 降维图像分析
    def dimensionality_reduction(airline_scale, kmeans_model):
```

```
        tsne = TSNE(n_components=2, init='random',
                    random_state=177).fit(airline_scale)
        # init：初始化，可以是PCA或random；随机数种子
        df = pd.DataFrame(tsne.embedding_)  # 将原始数据转换为DataFrame
        print(df)
        df['labels'] = kmeans_model.labels_  # 将聚类结果存储进df数据表
        print(df['labels'])

        # 提取不同标签的数据
        df1 = df[df['labels'] == 0]
        df2 = df[df['labels'] == 1]
        df3 = df[df['labels'] == 2]
        df4 = df[df['labels'] == 3]
        df5 = df[df['labels'] == 4]

        #  绘制图形
        fig = plt.figure(figsize=(9, 6))  # 设定空白画布，并制定大小
        # 用不同的颜色表示不同数据
        plt.plot(df2[0], df2[1], 'r*')
        plt.plot(df3[0], df3[1], 'gD')
        plt.plot(df4[0], df4[1], 'kD')
        plt.plot(df5[0], df5[1], 'lD')
        plt.savefig('../tmp/聚类结果.png')
        plt.show()  # 显示图片
```

TSNE 可视化

运行程序之后，得到的降维图像如下：

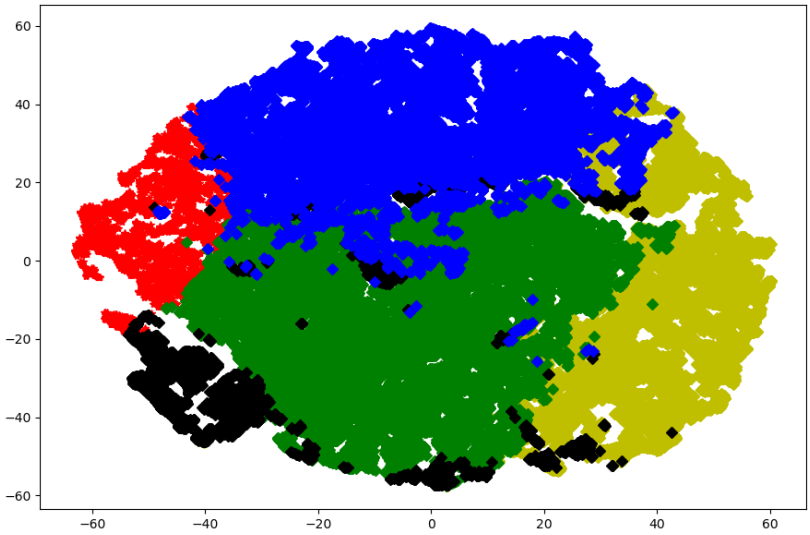


图 15: 降维图像

从图中可以看出，不同类别的客户被分布在不同的区域，说明聚类结果是比较合理的。

4.4.2 雷达图分析

雷达图是一种用于显示多变量数据的图表，它通常以两维的形式绘制，其中每个变量以轴从图表的中心点开始延伸。在本题中，可以将 LRFMC 模型的五个维度作为输入，然后将聚类结果进行可视化，通过雷达图对客户的价值进行分析代码如下：

```
# 画雷达图
def draw(kmeans_model=None):
    datafile = '标准数据.csv'
    data = pd.read_csv(datafile)
    r2 = pd.DataFrame(kmeans_model.cluster_centers_)  # 聚类
        的中心数学数值
    print(r2)
```

```
        r3 = pd.Series(['客户群1', '客户群2', '客户群3', '客户群
            4', '客户群5', ])
8       labels = np.array(['L', 'R', 'F', 'M', 'C'])  # 标签
        dataLength = 5  # 数据个数
10      r4 = r2.T
        print(data.columns)
12      r4.columns = list(data.columns)
        fig = plt.figure()
14      y = []
        for x in list(data.columns):
16          dt = r4[x]
            dt = np.concatenate((dt, [dt[0]]))
18          y.append(dt)
        ax = fig.add_subplot(111, polar=True)
20      angles = np.linspace(0, 2 * np.pi, dataLength, endpoint=
            False)
        angles = np.concatenate((angles, [angles[0]]))
22      labels = np.concatenate((labels, [labels[0]]))
        ax.plot(angles, y[0], 'b-', linewidth=2)
24      ax.plot(angles, y[1], 'r-', linewidth=2)
        ax.plot(angles, y[2], 'g-', linewidth=2)
26      ax.plot(angles, y[3], 'y-', linewidth=2)
        ax.plot(angles, y[4], 'm-', linewidth=2)
28      plt.rcParams['font.sans-serif'] = ['SimHei']
        plt.rcParams['axes.unicode_minus'] = False  # 用来正常显
            示负号
30      ax.legend(r3, loc=1)
        ax.set_thetagrids(angles * 180 / np.pi, labels,
            fontproperties="SimHei")
32      ax.set_title("用户价值分析雷达图", va='bottom',
            fontproperties="SimHei")
        ax.grid(True)
34      plt.show()
```

雷达图分析

得到的雷达图如下：

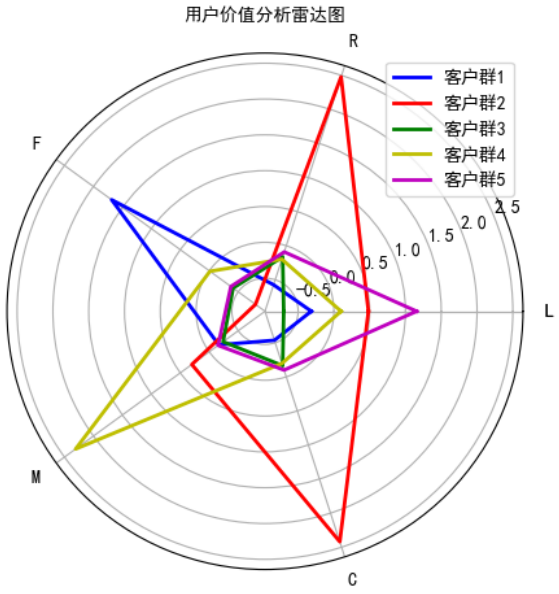


图 16: 雷达图

通过对雷达图的分析可以得知：

- 客户群 1：L 值较低，R 值较低，F 值较高，M 值较低，C 值较低，这类客户经常买票，且比较活跃（上次购票时间较近），但是消费金额较低，这类客户是重要发展客户，应该尽量提高这类客户的消费金额。
- 客户群 2：L 值较高，R 值较高，F 值较低，M 值较低，C 值较高，这类客户是老用户，最近消费较少，但是消费金额较高，这类客户是重要挽留客户，应该尽量提高这类客户的忠诚度。
- 客户群 3：L 值较低，R 值较低，F 值较低，M 值较低，C 值较低，这类客户是低价值客户，应该尽量避免这类客户流失。
- 客户群 4：L 值较低，R 值较低，F 值较低，M 值较高，C 值较低，这类客户是老用户，最近消费多，总飞行里程数高，折扣高，是重要保

持客户，应该优先将资源投入到这类客户中。

- 客户群 5：L 值较低，R 值较低，F 值较低，M 值较低，C 值较高，这类客户元老级用户，但是享受折扣、频率和里程数都比较低，是一般用户，不在重要维持范围内。

重要发展客户、重要保持客户、重要挽留客户分别可以归入航空公司客户生命周期管理的发展期、成熟期和衰退期，而低价值客户和一般用户则可以归入航空公司客户生命周期管理的成长期。

关于五个类别的客户数量如下图

图 17: 客户数量

可以看出，目前类别 2、4、0 的客户数量较多，对应的是处于发展期的用户，说明公司正在处于上升期。

5 总结

本次实验是对航空公司的客户进行分析，在分析过程中，学习到了 LRFMC 模型的构建方法，以及如何使用 K-Means 算法对客户进行聚类，最后通过雷达图对客户进行分析，得到了客户的价值分析结果。

本次实验加强了我对数据挖掘的理解，对于数据挖掘的流程有了更深的认识，同时也加强了我对 Python 的使用，对于 Python 的数据分析库有了更深的认识。

当然，在今后的学习中，我还需要继续加强对数据挖掘的理解，同时也需要加强对 Python 的学习，提高自己的编程能力。

6 附录

6.1 DataPretreatment.py

```
    import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.manifold import TSNE
from sklearn.preprocessing import StandardScaler


data = []


# 查看数据的整体情况和缺失值
def describe():
    airline_data = pd.read_csv("../data/air_data.csv",
                               encoding="gb18030")  # 导入航空数
                                   据
    explore = airline_data.describe().T
    explore['null'] = len(airline_data) - explore['count']
```

```
    explore.to_csv('数据描述.csv')
    df = explore[['max', 'min', 'null']]
    nullData = df[df['null'] > 0]
    nullData = nullData['null']
    nullData.to_csv('空值项数量.csv')
    return airline_data


# 删除缺失值和异常值
def data_cleaning(airline_data):
    exp1 = airline_data["SUM_YR_1"].notnull()
    exp2 = airline_data["SUM_YR_2"].notnull()
    exp = exp1 & exp2
    # print(exp)
    # print('exp的形状是：', exp.shape)
    airline_notnull = airline_data.loc[exp, :]
    print('删除缺失记录后数据的形状为：', airline_notnull.shape)

    # 只保留票价非零的，或者平均折扣率不为0且总飞行公里数大于0的
        记录。
    index1 = airline_notnull['SUM_YR_1'] != 0
    index2 = airline_notnull['SUM_YR_2'] != 0
    index3 = (airline_notnull['SEG_KM_SUM'] > 0) & \
             (airline_notnull['avg_discount'] != 0)
    airline = airline_notnull[(index1 | index2) & index3]
    print('删除异常记录后数据的形状为：', airline.shape)
    return airline


# 数据筛选
def data_selection(airline):
    # 选取需求特征
    """
    L：入会时间
    FLIGHT_COUNT：F 乘机次数
```

```
    LAST_TO_END: R 最近消费次数
    avg_discount: C 折扣率
    SEG_KM_SUM: M 飞行里程数
    """
    airline_selection = airline[["FFP_DATE", "LOAD_TIME",
                                 "FLIGHT_COUNT", "LAST_TO_END",
                                 "avg_discount", "SEG_KM_SUM"]]
    L = pd.to_datetime(airline_selection["LOAD_TIME"]) - \
        pd.to_datetime(airline_selection["FFP_DATE"])
    L = L.astype("str").str.split(' ').str[0]
    # 获得会员入会的月数
    L = L.astype("int") / 30
    # 合并特征
    airline_features = pd.concat([L, airline_selection.iloc[:,
        2:]], axis=1)
    print('构建LRFMC特征前五行为: \n',airline_features.head())
    return airline_features


# 数据标准化
def data_normalization(airline_features):
    global data
    data = StandardScaler().fit_transform(airline_features)
    np.savez('../tmp/airline_scale.npz', data)
    print('标准化后LRFMC五个特征为: \n', data[:5, :])


def main():
    airline_data = describe()
    airline = data_cleaning(airline_data)
    airline_features = data_selection(airline)
    data_normalization(airline_features)


if __name__ == '__main__':
```

```
    main()
```

DataPretreatment.py

6.2 DataAnalysis.py

```
    import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
from sklearn.manifold import TSNE
from sklearn.metrics import fowlkes_mallows_score,
    silhouette_score, calinski_harabasz_score


# 读入数据
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC


def read_data():
    airline_scale = np.load('../tmp/airline_scale.npz')['arr_0']
    return airline_scale


# 聚类数据分析
def cluster(airline_scale):
    k = 5   # 聚类中心数
    kmeans_model = KMeans(n_clusters=k, n_init=4, random_state
        =666)
    kmeans_model.fit(airline_scale)  # 模型训练
    print('聚类中心:\n', kmeans_model.cluster_centers_)  # 查看
        聚类中心
```

```
26      print('类别标签:\n', kmeans_model.labels_)  # 查看样本的类别
            标签
        # 统计不同类别样本的数目
28      r1 = pd.Series(kmeans_model.labels_).value_counts()
        print('最终每个类别的数目为: \n', r1)
30      return kmeans_model


32
# FMI评价法
34  def FMI(airline_scale, kmeans_model):
        for i in range(2, 7):
36          kmeans = KMeans(n_clusters=i, random_state=123).fit(
                airline_scale)
            print(kmeans.labels_)
38          print(type(kmeans.labels_))
            score = fowlkes_mallows_score(kmeans_model.labels_,
                kmeans.labels_)
40          print('数据聚%d类FMI评价分值为: %f' % (i, score))


42
# silhouetteScore相似度评价法
44  def SS(airline_scale):
        silhouetteScore = []
46      for i in range(2, 7):
            kmeans = KMeans(n_clusters=i, random_state=123).fit(
                airline_scale)
48          score = silhouette_score(airline_scale, kmeans.labels_)
            print('航空公司数据聚%d类silhouette评价分值为: %f' % (i,
                score))
50          silhouetteScore.append(score)
        plt.figure(figsize=(10, 6))
52      plt.plot(range(2, 7), silhouetteScore, linewidth=1.5,
            linestyle="-")
        plt.show()
54
```

```
# calinski_harabaz指数评价法
def CH(airline_scale):
    ch = []
    for i in range(2, 7):
        # 构建并训练模型
        kmeans = KMeans(n_clusters=i, random_state=123).fit(
            airline_scale)
        score = calinski_harabasz_score(airline_scale, kmeans.
            labels_)
        print('航空公司数据聚%d类calinski_harabaz指数为：%f' % (
            i, score))
        ch.append(score)
    plt.figure(figsize=(10, 6))
    plt.plot(range(2, 7), ch, linewidth=1.5, linestyle="-")
    plt.show()


# 用支持向量机预测数据
def SVC_prediction(airline_scale, kmeans_model):
    # 划分测试集和训练集
    airline_data_train, airline_data_test, airline_target_train,
        airline_target_test = \
        train_test_split(airline_scale, kmeans_model.labels_,
            test_size=0.2, random_state=666)

    # 数据标准化
    stdScaler = StandardScaler().fit(airline_data_train)
    airline_trainStd = stdScaler.transform(airline_data_train)
    airline_testStd = stdScaler.transform(airline_data_test)

    svm = SVC().fit(airline_trainStd, airline_target_train)
    print('建立的SVM模型为：\n', svm)

    #  预测训练集结果
```

```
    airline_target_pred = svm.predict(airline_testStd)
86  print('预测前20个结果为: \n', airline_target_pred[:20])


88
# 画雷达图
90 def draw(kmeans_model=None):
    datafile = '标准数据.csv'
92  data = pd.read_csv(datafile)
    r2 = pd.DataFrame(kmeans_model.cluster_centers_) # 聚类的中
        心数学数值
94  print(r2)
    r3 = pd.Series(['客户群1', '客户群2', '客户群3', '客户群4',
        '客户群5', ])
96  labels = np.array(['L', 'R', 'F', 'M', 'C']) # 标签
    dataLength = 5 # 数据个数
98  r4 = r2.T
    print(data.columns)
100 r4.columns = list(data.columns)
    fig = plt.figure()
102 y = []
    for x in list(data.columns):
104     dt = r4[x]
        dt = np.concatenate((dt, [dt[0]]))
106     y.append(dt)
    ax = fig.add_subplot(111, polar=True)
108 angles = np.linspace(0, 2 * np.pi, dataLength, endpoint=
        False)
    angles = np.concatenate((angles, [angles[0]]))
110 labels = np.concatenate((labels, [labels[0]]))
    ax.plot(angles, y[0], 'b-', linewidth=2)
112 ax.plot(angles, y[1], 'r-', linewidth=2)
    ax.plot(angles, y[2], 'g-', linewidth=2)
114 ax.plot(angles, y[3], 'y-', linewidth=2)
    ax.plot(angles, y[4], 'm-', linewidth=2)
116 plt.rcParams['font.sans-serif'] = ['SimHei']
```

```
    plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
    ax.legend(r3, loc=1)
    ax.set_thetagrids(angles * 180 / np.pi, labels, fontproperties="SimHei")
    ax.set_title("用户价值分析雷达图", va='bottom', fontproperties="SimHei")
    ax.grid(True)
    plt.show()


# 降维图像分析
def dimensionality_reduction(airline_scale, kmeans_model):
    tsne = TSNE(n_components=2, init='random',
                random_state=177).fit(airline_scale)
    # init: 初始化，可以是PCA或random；随机数种子
    df = pd.DataFrame(tsne.embedding_)  # 将原始数据转换为DataFrame
    print(df)
    df['labels'] = kmeans_model.labels_  # 将聚类结果存储进df数据表
    print(df['labels'])

    # 提取不同标签的数据
    df1 = df[df['labels'] == 0]
    df2 = df[df['labels'] == 1]
    df3 = df[df['labels'] == 2]
    df4 = df[df['labels'] == 3]
    df5 = df[df['labels'] == 4]

    #  绘制图形
    fig = plt.figure(figsize=(9, 6))  # 设定空白画布，并制定大小
    # 用不同的颜色表示不同数据
    plt.plot(df2[0], df2[1], 'r*')
    plt.plot(df3[0], df3[1], 'gD')
```

```
    plt.plot(df4[0], df4[1], 'kD')
    plt.plot(df5[0], df5[1], 'bD')
    plt.savefig('../tmp/聚类结果.png')
    plt.show()  # 显示图片


def main():
    airline_scale = read_data()
    kmeans_model = cluster(airline_scale)
    # FMI(airline_scale, kmeans_model)
    # SS(airline_scale)
    # CH(airline_scale)
    # SVC_prediction(airline_scale, kmeans_model)
    # dimensionality_reduction(airline_scale, kmeans_model)
    draw(kmeans_model)


if __name__ == '__main__':
    main()
```

DataAnalysis.py