航空公司客户价值分析

郭锦红

目录

一、目的与要求

1.1. 目的:

1、借助航空公司客户数据，对客户进行分类。
2、对不同的客户类别进行特征分析，比较不同类客户的价值。
3、对不同价值的客户类别提供个性化服务，制定相应的营销策略。

1.2. 要求:

1、从航空公司的数据源中进行数据抽取。
2、进行数据探索分析和预处理，包括数据缺失值与异常值的探索分析，数据的属性规约、清洗和变换。
3、使用预处理之后的建模数据，基于旅客价值 LRFMC 模型进行客户分群，对各个客户群进行特征分析，识别出有价值的客户。
4、针对模型结果得到不同价值的客户，采用不同的营销手段，提供定制化的服务。

二、数据说明

2.1. 数据量

62988 条数据，44 列属性

2.2. 示例数据

	A	B	C	D	E	F	G	H	I	J	K
1	MEMBER_NO	FFP_DATE	FIRST_FLIGHT_DATE	GENDER	FFP_TIER	WORK_CITY	WORK_PROVINCE	WORK_COUNTRY	AGE	LOAD_TIME	FLIGHT_COUNT
2	54993	2006/11/2	2008/12/24	男	6	.	北京	CN	31	2014/3/31	210
3	28065	2007/2/19	2007/8/3	男	6		北京	CN	42	2014/3/31	140
4	55106	2007/2/1	2007/8/30	男	6	.	北京	CN	40	2014/3/31	135
5	21189	2008/8/22	2008/8/23	男	5	Los Angele	CA	US	64	2014/3/31	23

	L	M	N	O	P	Q	R	S	T	U	V
1	BP_SUM	EP_SUM_YR_1	EP_SUM_YR_2	SUM_YR_1	SUM_YR_2	SEG_KM_SUM	WEIGHTED_SEG_KM	LAST_FLIGHT_DATE	AVG_FLIGHT_COUNT	AVG_BP_SUM	BEGIN_TO_FIRST
2	505308	0	74460	239560	234188	580717	558440.14	2014/3/31	26.25	63163.5	2
3	362480	0	41288	171483	167434	293678	367777.2	2014/3/25	17.5	45310	2
4	351159	0	39711	163618	164982	283712	355966.5	2014/3/21	16.875	43894.875	10
5	337314	0	34890	116350	125500	281336	306900.88	2013/12/26	2.875	42164.25	21

	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG
1	LAST_TO_END	AVG_INTERVAL	MAX_INTERVAL	ADD_POINTS_SUM_YR_1	ADD_POINTS_SUM_YR_2	EXCHANGE_COUNT	avg_discount	P1Y_Flight_Count	L1Y_Flight_Count	P1Y_BP_SUM	L1Y_BP_SUM
2	1	3.483253589	18	3352	36640	34	0.961639043	103	107	246197	259111
3	7	5.194244604	17	0	12000	29	1.25231444	68	72	177358	185122
4	11	5.298507463	18	3491	12000	20	1.254675516	65	70	169072	182087
5	97	27.86363636	73	0	0	11	1.090869565	13	10	186104	151210

	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR
1	EP_SUM	ADD_Point_SUM	Eli_Add_Point_Sum	L1Y_ELi_Add_Points	Points_Sum	L1Y_Points_Sum	Ration_L1Y_Flight_Count	Ration_P1Y_Flight_Count	Ration_P1Y_BPS	Ration_L1Y_BPS	Point_NotFlight
2	74460	39992	114452	111100	619760	370211	0.50952381	0.49047619	0.487220691	0.51277733	50
3	41288	12000	53288	53288	415768	238410	0.514285714	0.485714286	0.489289094	0.510708147	33
4	39711	15491	55202	51711	406361	233798	0.518518519	0.481481481	0.481467137	0.518530015	26
5	34890	0	34890	34890	372204	186100	0.434782609	0.565217391	0.551721684	0.448275351	12

三、从数据源中读取数据

3.1. 启动 spark

```
# start-dfs.sh
# start-yarn.sh
# start-spark.sh
# spark-shell
```

3.2. 读取数据

代码：

```
import org.apache.spark.sql.SparkSession

val spark=SparkSession.builder().getOrCreate()

import spark.implicits._

// 事先要把 csv 专成 csv 的 utf8 形式
val df=spark.read.format("csv").option("header","true").load("file:///simple/air_data_base2.csv")

df.show()
```

运行结果如下图：

```
scala> import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.SparkSession

scala> val spark=SparkSession.builder().getOrCreate()
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@4
afe8464

scala> import spark.implicits._
import spark.implicits._

scala> val df=spark.read.format("csv").option("header","true").load("file:///s
imple/air_data_base2.csv")
```

```
---------+-------------+-------------+--------------+
|MEMBER_NO|  FFP_DATE|FIRST_FLIGHT_DATE|GENDER|FFP_TIER|      WORK_CITY|WORK_P
ROVINCE|WORK_COUNTRY|AGE|LOAD_TIME|FLIGHT_COUNT|BP_SUM|EP_SUM_YR_1|EP_SUM_YR_2
|SUM_YR_1|SUM_YR_2|SEG_KM_SUM|WEIGHTED_SEG_KM|LAST_FLIGHT_DATE|AVG_FLIGHT_COUN
T|AVG_BP_SUM|BEGIN_TO_FIRST|LAST_TO_END|AVG_INTERVAL|MAX_INTERVAL|ADD_POINTS_S
UM_YR_1|ADD_POINTS_SUM_YR_2|EXCHANGE_COUNT|avg_discount|P1Y_Flight_Count|L1Y_F
light_Count|P1Y_BP_SUM|L1Y_BP_SUM|EP_SUM|ADD_Point_SUM|Eli_Add_Point_Sum|L1Y_E
Li_Add_Points|Points_Sum|L1Y_Points_Sum|Ration_L1Y_Flight_Count|Ration_P1Y_Fli
ght_Count|Ration_P1Y_BPS|Ration_L1Y_BPS|Point_NotFlight|
+---------+----------+-----------------+------+--------+--------------+------
-------+------------+---+---------+------------+------+-----------+----------
+--------+--------+----------+--------------+----------------+--------------
-+----------+--------------+-----------+------------+------------+------------
-------+-------------------+--------------+------------+----------------+-----
-----------+----------+----------+------+-------------+-----------------+-----
-------------+----------+--------------+-----------------------+-------------
---------+-------------+-------------+--------------+
|    54993| 2006/11/2|       2008/12/24|     男|       6|              .|
      北京|          CN| 31|2014/3/31|         210|505308|          0|      74
460|  239560|  234188|    580717|      558440.14|       2014/3/31|           2
6.25|   63163.5|             2|           1| 3.483253589|          18|
      3352|              36640|            34| 0.961639043|             103|
           107|    246197|    259111| 74460|        39992|           114452|
          111100|    619760|        370211|             0.50952381|
  0.49047619|   0.487220691|    0.51277733|             50|
|    28065| 2007/2/19|         2007/8/3|     男|       6|           null|
```

四、数据探索与预处理

4.1. 数据探索性分析

运行代码以及结果如下：

```
scala> val df_desc = df.describe("MEMBER_NO","FFP_DATE","FIRST_FLIGHT_DATE","F
FP_TIER","AGE","LOAD_TIME","FLIGHT_COUNT","BP_SUM","EP_SUM_YR_1","EP_SUM_YR_2"
,"SUM_YR_1","SUM_YR_2","SEG_KM_SUM","WEIGHTED_SEG_KM","LAST_FLIGHT_DATE","AVG_
FLIGHT_COUNT","AVG_BP_SUM","BEGIN_TO_FIRST","LAST_TO_END","AVG_INTERVAL","MAX_
INTERVAL","ADD_POINTS_SUM_YR_1","ADD_POINTS_SUM_YR_2","EXCHANGE_COUNT","avg_di
scount","P1Y_Flight_Count","L1Y_Flight_Count","P1Y_BP_SUM","L1Y_BP_SUM","EP_SU
M","ADD_Point_SUM","Eli_Add_Point_Sum","L1Y_ELi_Add_Points","Points_Sum","L1Y_
Points_Sum","Ration_L1Y_Flight_Count","Ration_P1Y_Flight_Count","Ration_P1Y_BP
S","Ration_L1Y_BPS","Point_NotFlight")
[Stage 28:>                                                         (0 + 4) /
[Stage 28:==============>                                           (1 + 3) /
[Stage 28:============================>                             (2 + 2) /
[Stage 28:==========================================>               (3 + 1) /

df_desc: org.apache.spark.sql.DataFrame = [summary: string, MEMBER_NO: string
... 39 more fields]
```

```
scala> df_desc.write.format("csv").option("header","true").save("file:///simpl
e/air_data_desc.csv")
```

结果展示如下：

summary	Null	mean	stddev	min	max
MEMBER_NO	0	31494.5	18183.21	1	9999
FFP_DATE	0			38292	41342
FIRST_FLIGHT_DATE	0			2192	42154
FFP_TIER	0	4.102162	0.373856	4	6
WORK_CITY	2268				
WORK_PROVINCE	3244				
WORK_COUNTRY	25				
AGE	420	42.47635	9.885915	110	92
LOAD_TIME	0			41729	41729
FLIGHT_COUNT	0	11.83941	14.04947	10	99
BP_SUM	0	10925.08	16339.49	0	9999
EP_SUM_YR_1	0	0	0	0	0
EP_SUM_YR_2	0	265.6896	1645.703	0	998
SUM_YR_1	591	6040.723	9315.788	0	9998
SUM_YR_2	138	5604.026	8703.364	0	9999
SEG_KM_SUM	0	17123.88	20960.84	1000	9999
WEIGHTED_SEG_KM	0	12777.15	17578.59	0	9999.98
LAST_FLIGHT_DATE	0			41183	41707
AVG_FLIGHT_COUNT	0	1.542154	1.786996	0.25	9.875
AVG_BP_SUM	0	1421.44	2083.121	0	999.875
BEGIN_TO_FIRST	0	120.1455	159.5729	0	99
LAST_TO_END	0	176.1201	183.8222	1	99
AVG_INTERVAL	0	67.74979	77.51787	0	99.85714
MAX_INTERVAL	0	166.0339	123.3972	0	99
ADD_POINTS_SUM_YR_1	0	540.317	3956.083	0	9990
ADD_POINTS_SUM_YR_2	0	814.6893	5121.797	0	9991
EXCHANGE_COUNT	0	0.319775	1.136004	0	9
avg_discount	0	0.721558	0.185427	0	1.5
P1Y_Flight_Count	0	5.766257	7.210922	0	90
L1Y_Flight_Count	0	6.073157	8.175127	0	99
P1Y_BP_SUM	0	5366.721	8537.773	0	9998
L1Y_BP_SUM	0	5558.361	9351.957	0	9998
EP_SUM	0	265.6896	1645.703	0	998
ADD_Point_SUM	0	1355.006	7868.477	0	999
Eli_Add_Point_Sum	0	1620.696	8294.399	0	9995
L1Y_ELi_Add_Points	0	1080.379	5639.857	0	999
Points_Sum	0	12545.78	20507.82	0	9999
L1Y_Points_Sum	0	6638.74	12601.82	0	9998
Ration_L1Y_Flight_Count	0	0.486419	0.319105	0	1
Ration_P1Y_Flight_Count	0	0.513581	0.319105	0	1
Ration_P1Y_BPS	0	0.522293	0.339632	0	0.999989
Ration_L1Y_BPS	0	0.468422	0.338956	0	0.999993
Point_NotFlight	0	2.728155	7.364164	0	96

4.2. 数据预处理

根据上面的数据统计，丢弃所有不符合的数据:

1、票价为空的

2、票价为 0，但是折扣不是 0，而且飞行里程大于 0，

这样的数据是错误数据，可能是客户不存在乘机记录造成，直接删除。

代码如下:

```
val df2 = spark.sql("SELECT * FROM air WHERE sum_yr_1 is NOT NULL AND sum_yr_2 is NOT NULL AND sum_yr_1 != 0 AND sum_yr_2 != 0 AND avg_discount != 0 AND seg_km_sum > 0")
// 将处理后的数据保存起来
df2.write.format("csv").option("header","true").save("file:///simple/air_data_handle.csv")
```

运行结果如下:

```
scala> val df2 = spark.sql("SELECT * FROM air WHERE sum_yr_1 is NOT NULL AND sum_y
r_2 is NOT NULL AND sum_yr_1 != 0 AND avg_discount != 0 AND seg_km_sum > 0")
df2: org.apache.spark.sql.DataFrame = [MEMBER_NO: string, FFP_DATE: string ... 42
more fields]
```

```
scala> df2.write.format("csv").option("header","true").save("file:///simple/ai
r_data_handle.csv")
[Stage 16:>                                                         (0 + 4) /
[Stage 16:==============>                                           (1 + 3) /
```

处理后的数据有 61480 条，占原数据的 97.60%，因此不会对分析结果产生较大的影响

4.3. 属性规约

传统的识别客户价值应用最广泛的模型主要通过 3 个指标（最近消费时间间隔（Recency）、消费频率（Frequency）和消费金额（Monetary））来进行客户细分，识别出价值高的客户，简称 RFC 模型。

在 RFC 模型中，消费金额表示在一段时间内，客户购买产品的总金额。但是不适用于航空公司的数据处理。因此我们用客户在一段时间内的累计飞行里程 M 和客户在一定时间内乘坐舱位的折扣系数 C 代表消费金额。再在模型中增加客户关系长度 L，所以我们用 LRFMC 模型。

去掉不相管的属性，只留下与 LRFMC 模型相关的属性。

- MEMBER_NO 会员卡号
- FFP_DATE 入会时间
- LOAD_TIME 观测窗口结束时间
- FLIGHT_COUNT 飞行频率
- avg_discount 平均折扣
- SEG_KM_SUM 总飞行千米数
- LAST_TO_END 最后一次乘机时间至观察窗口末端时长

代码如下:

```
// 属性规约,选取所需字段
val df3 =
df2.select("MEMBER_NO","FFP_DATE","LOAD_TIME","FLIGHT_COUNT","avg_discount","SEG_K
```

```
M_SUM","LAST_TO_END")
// 查看规约后的数据
df3.show()
// 保存规约后的数据
df3.write.format("csv").option("header","true").save("file:///simple/air_data_guiyue.csv")
```

运行结果如下:

```
scala> val df3 = df2.select("MEMBER_NO","FFP_DATE","LOAD_TIME","FLIGHT_COUNT",
"avg_discount","SEG_KM_SUM","LAST_TO_END")
df3: org.apache.spark.sql.DataFrame = [MEMBER_NO: string, FFP_DATE: string ...
 5 more fields]

scala> df3.show()
20/01/02 09:30:02 WARN util.Utils: Truncated the string representation of a pl
an since it was too large. This behavior can be adjusted by setting 'spark.deb
ug.maxToStringFields' in SparkEnv.conf.
+---------+----------+---------+------------+------------+----------+---------
--+
|MEMBER_NO|  FFP_DATE|LOAD_TIME|FLIGHT_COUNT|avg_discount|SEG_KM_SUM|LAST_TO_E
ND|
+---------+----------+---------+------------+------------+----------+---------
--+
|    54993| 2006/11/2|2014/3/31|         210| 0.961639043|    580717|
 1|
|    28065| 2007/2/19|2014/3/31|         140|  1.25231444|    293678|
 7|
...

only showing top 20 rows


scala> df3.write.format("csv").option("header","true").save("file:///simple/ai
r_data_guiyue.csv")

scala>
```

五、LRFMC 模型

这里选择用 python 对已处理的数据进行建模

5.1. LRFMC 模型指标含义

（1） L：Long，会员入会时间距观测窗口结束的时间（月份），入会时间。

（2） R：Recency 客户最近一次乘坐公司飞机距离观测窗口结束的时间（月份）。

（3） F：Frequency 客户在观测窗口内乘坐公司飞机的次数。

（4） M：Milepost，客户在观测窗口内累计的飞行里程碑。

（5） C：Count，客户在观测窗口内乘坐仓位所对应的折扣系数的平均值。

5.2. 变换为 LRFMC 数据

5.2.1.数据的再次读取

```
# 读取数据
data1 = pd.read_csv('./air_data_guiyue/part-r-00000-b22b9f30-4b74-4d4d-af80-e5d6cb796c62.csv',encoding='utf-8')
data2 = pd.read_csv('./air_data_guiyue/part-r-00001-b22b9f30-4b74-4d4d-af80-e5d6cb796c62.csv',encoding='utf-8')
data3 = pd.read_csv('./air_data_guiyue/part-r-00002-b22b9f30-4b74-4d4d-af80-e5d6cb796c62.csv',encoding='utf-8')
data4 = pd.read_csv('./air_data_guiyue/part-r-00003-b22b9f30-4b74-4d4d-af80-e5d6cb796c62.csv',encoding='utf-8')
print(len(data1),len(data2),len(data3),len(data4))
data1.tail()
```

```
data = pd.concat([data1,data2])
data = pd.concat([data,data3])
data = pd.concat([data,data4])
data.shape
```

```
data.tail()
```

```
data.reset_index(drop=True,inplace=True)
data.tail()
```

	MEMBER_NO	FFP_DATE	LOAD_TIME	FLIGHT_COUNT	avg_discount	SEG_KM_SUM	LAST_TO_END
61475	11163	2005/5/8	2014/3/31	2	0.710	368	89
61476	30765	2008/11/16	2014/3/31	2	0.670	368	121
61477	10380	2010/7/8	2014/3/31	2	0.225	1062	39
61478	16372	2012/12/20	2014/3/31	2	0.250	904	464
61479	22761	2011/4/14	2014/3/31	2	0.280	760	282

5.2.2.计算 LRFMC 数据

```
# 转化为时间格式
data['FFP_DATE'] = pd.to_datetime(data['FFP_DATE'])
data['LOAD_TIME'] = pd.to_datetime(data['LOAD_TIME'])
print(data.dtypes)
data.head()
```

```
# 创建一个新的数据框
data_LRFMC = pd.DataFrame()
# data_LRFMC['MEMBER_NO'] = data['MEMBER_NO']
data_LRFMC['L'] = [x.days for  x in (data['LOAD_TIME']-data['FFP_DATE'])/30]
data_LRFMC['R'] = data['LAST_TO_END']
data_LRFMC['F'] = data['FLIGHT_COUNT']
data_LRFMC['M'] = data['SEG_KM_SUM']
data_LRFMC['C'] = data['avg_discount']
data_LRFMC.head()
# data_LRFMC.shape
```

	L	R	F	M	C
0	90	1	210	580717	0.961639
1	86	7	140	293678	1.252314
2	87	11	135	283712	1.254676
3	68	97	23	281336	1.090870
4	60	5	152	309928	0.970658

5.3. 数据探索性分析及数据标准化

最大值和最小值间隔较大，需要对数据进行标准化。

```
data_LRFMC.describe().T
```

	count	mean	std	min	25%	50%	75%	max
L	61480.0	49.129652	28.260054	12.000000	24.000000	42.000000	72.000000	114.0
R	61480.0	173.230368	182.051208	1.000000	29.000000	105.000000	262.000000	731.0
F	61480.0	12.041851	14.076794	2.000000	3.000000	7.000000	15.000000	213.0
M	61480.0	17428.848650	20988.237322	368.000000	5023.000000	10309.000000	21649.250000	580717.0
C	61480.0	0.723944	0.183587	0.112043	0.615228	0.713096	0.809702	1.5

```
# 最大最小标准化数据
from sklearn.preprocessing import MinMaxScaler
min_max_scaler = MinMaxScaler()  # 实例化一个最小最大化方法
data_mms = min_max_scaler.fit_transform(data_LRFMC)  # 对数据进行标准化
data_mms
```

```
array([[7.64705882e-01, 0.00000000e+00, 9.85781991e-01, 1.00000000e+00,
        6.12119775e-01],
       [7.25490196e-01, 8.21917808e-03, 6.54028436e-01, 5.05402783e-01,
        8.21546623e-01],
       [7.35294118e-01, 1.36986301e-02, 6.30331754e-01, 4.88230358e-01,
```

5.4. K-Means 算法

5.4.1.尝试分 4 簇

```
from sklearn.cluster import KMeans
```

```
n_cluster = 4
cluster = KMeans(n_clusters=n_cluster,random_state=0).fit(data_mms)
```

```
# 重要属性，cluster_centers_,查看质心(中间点)
centerid = cluster.cluster_centers_
centerid
```

```
array([[0.15582143, 0.62731649, 0.01047321, 0.01098468, 0.43208885],
       [0.18232938, 0.11984225, 0.05053539, 0.03096135, 0.43407097],
       [0.63709182, 0.58525105, 0.01371961, 0.01218058, 0.4426757 ],
       [0.68908362, 0.09695206, 0.07375957, 0.04233565, 0.456036  ]])
```

```
#  各客户群体的中心点
clf = pd.DataFrame(data=centerid,index=['客户群1','客户群2','客户群3','客户群4'],
                   columns=data_LRFMC.columns).round(2)
clf
```

	L	R	F	M	C
客户群1	0.16	0.63	0.01	0.01	0.43
客户群2	0.18	0.12	0.05	0.03	0.43
客户群3	0.64	0.59	0.01	0.01	0.44
客户群4	0.69	0.10	0.07	0.04	0.46

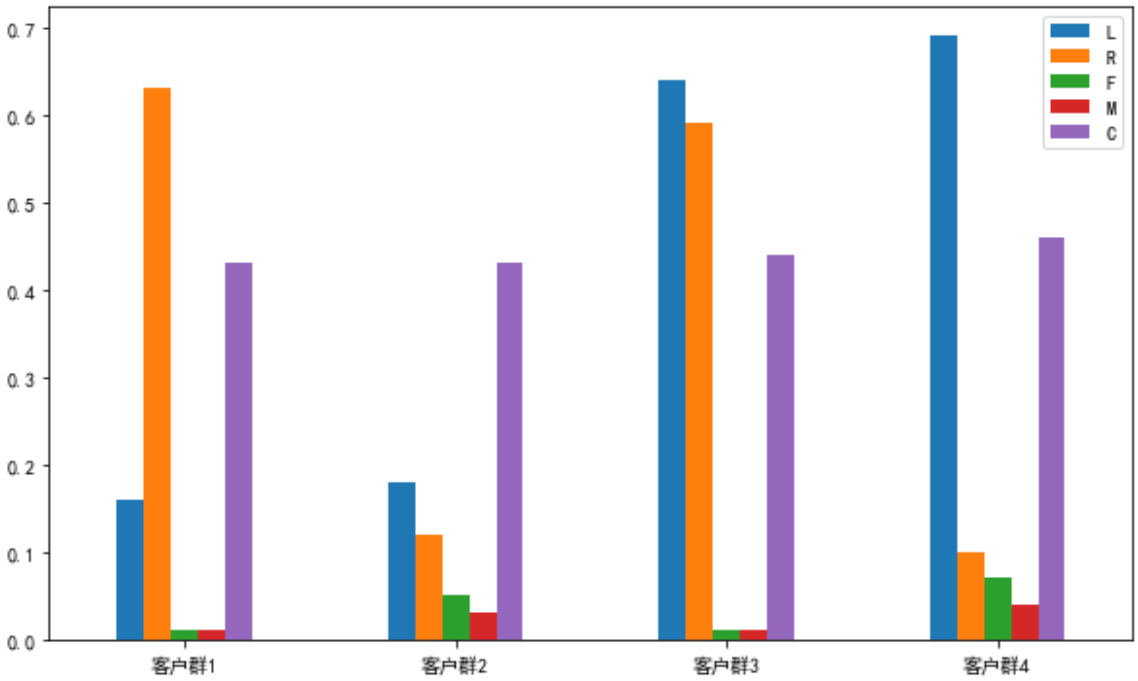
```
data_LRFMC['MEMBER_NO'] = data['MEMBER_NO']
data_LRFMC['labels'] = cluster.labels_
data_LRFMC[['MEMBER_NO','L','R','F','M','C','labels']].tail()
```

	MEMBER_NO	L	R	F	M	C	labels
61475	11163	108	89	2	368	0.710	3
61476	30765	65	121	2	368	0.670	3
61477	10380	45	39	2	1062	0.225	1
61478	16372	15	464	2	904	0.250	0
61479	22761	36	282	2	760	0.280	0

5.4.2.数据可视化

```
import matplotlib.pyplot as plt
plt.rcParams['font.family'] = 'SimHei'
clf.plot.bar(figsize=(10,6))
plt.xticks(rotation=0)
```

```
(array([0, 1, 2, 3]), <a list of 4 Text xticklabel objects>)
```



5.4.3.客户价值分析

- **客户群 1：一般客户**。原因：L 偏低，但 R 偏高，入会时间短。
- **客户群 2：重要挽留客户**。原因：L 虽低，但 FM 不低，入会时间长，乘坐飞机次数却不少。
- **客户群 3：低价值客户**。原因：L 高但 F、M 低，入会时间长，却很少乘坐飞机，总飞机里程数也少。
- **客户群 4：重要保持客户**。原因：L、F、M 高，入会时间长，乘坐飞机次数多，总飞机里程数也多。