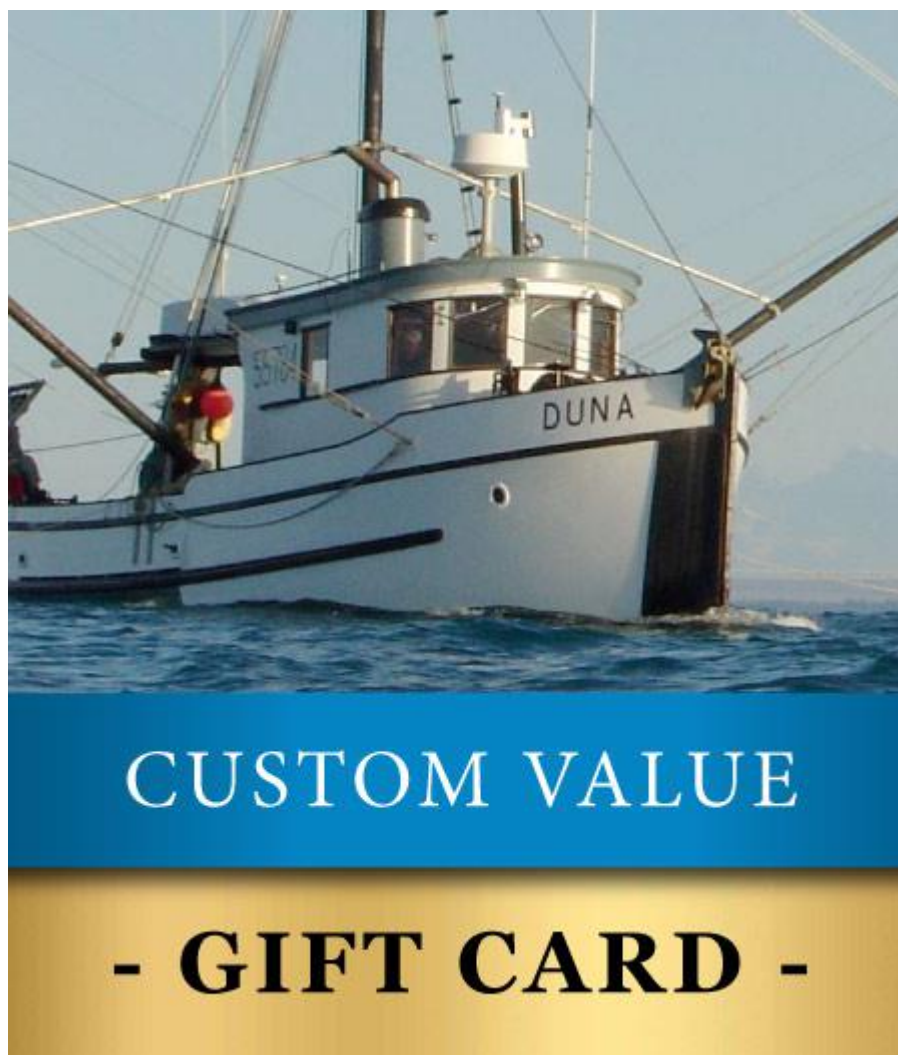


# 基于聚类(Kmeans)算法实现的客户价值分析系统



## 一、项目背景

### 【项目简介】

航空公司业务竞争激烈，借助航空公司客户数据，对客户进行分类，从产品中心转化为客户中心，通过对不同的客户类别进行特征分析，比较不同类客户的客户价值，针对不同类型客户，进行精准营销，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实现利润最大化，建立客户价值评估模型，进行客户分类，是解决问题的办法，由此应运而生基于聚类(Kmeans)算法实现的客户价值分析系统。

### 【项目涉及知识点】

- ✧ Sklearn 基本操作
- ✧ K-means 算法的理解
- ✧ K-means 算法实现

---

## 二、项目基本需求及目的

### 【项目需求】

本课程需要你具有一定的 Sklearn 基础，以下为推荐在本课程之前需要学习的课程（已按先后顺序进行排列）：

- √ Python 基础
- √ Sklearn 入门
- √ K-means 原理理解

### 【项目目的】

基于《python 数据分析与挖掘实战》一书中 air\_data.csv 数据集，通过 Sklearn 和机器学习相关技巧，通过数据分析手段比较不同类客户的客户价值。

## 三、项目准备工作

### 【项目平台】

- 1, PC 机，如果你的电脑内存低于 512M，希望你不要安装虚拟机及项目所需的环境。
- 2, Python 安装。本项目使用 python 3.6
- 3, pandas, matplotlib, numpy 安装

## 四、项目实施步骤

### 【项目实施步骤】

## 1、数据集简介及准备

### 1.1 数据集简介

---

数据集给出了关于 62988 个客户的基本信息和在观测窗口内的消费积分等相关信息，其中包含了会员卡号、入会时间、性别、年龄、会员卡级别、在观测窗口内的飞行公里数、飞行时间等 44 个特征属性。

原始数据集的特征属性太多，而且各属性不具有降维的特征，故这里选取几个对航空公司来说比较有价值的几个特征进行分析，这里并没有完全按照书中的做法选取特征，最终选取的特征是第一年总票价、第二年总票价、观测窗口总飞行公里数、飞行次数、平均乘机时间间隔、观察窗口内最大乘机间隔、入会时间、观测窗口的结束时间、平均折扣率这八个特征。

下面说明这么选的理由：

1，选取的特征是第一年总票价、第二年总票价、观测窗口总飞行公里数是要计算平均飞行每公里的票价，因为对于航空公司来说并不是票价越高，飞行公里数越长越能创造利润，相反而是那些近距离的高等舱的客户创造更大的利益。

2，总飞行公里数、飞行次数也都是评价一个客户价值的重要指标

- 3, 入会时间可以看出客户是不是老用户及忠诚度
- 4, 通过平均乘机时间间隔、观察窗口内最大乘机间隔可以判断客户的乘机频率是不是固定
- 5, 平均折扣率可以反映出客户给公里带来的利益, 毕竟来说越是高价值的客户享用的折扣率越高

数据集局部图如下图所示:

MEMBER_NO	FFP_DATE	FIRST_FLIGHT	GENDER	FFP_TIER	WORK_CITY	WORK_PROV	WORK_COUNAGE	LOAD_TIME	FLIGHT_CCBP_SUM
54993	2006/11/2	2008/12/24	男	6	北京	CN	31	2014/3/31	210
28065	2007/2/19	2007/8/3	男	6	北京	CN	42	2014/3/31	140
55106	2007/2/1	2007/8/30	男	6	北京	CN	40	2014/3/31	135
21189	2008/8/22	2008/8/23	男	5	Los Angeles	CA	US	64	2014/3/31
39546	2009/4/10	2009/4/15	男	6	贵阳	贵州	CN	48	2014/3/31
56972	2008/2/10	2009/9/29	男	6	广州	广东	CN	64	2014/3/31
44924	2006/3/22	2006/3/29	男	6	乌鲁木齐市	新疆	CN	46	2014/3/31
22631	2010/4/9	2010/4/9	女	6	温州市	浙江	CN	50	2014/3/31
32197	2011/6/7	2011/7/1	男	5	DRANCY		FR	50	2014/3/31
31645	2010/7/5	2010/7/5	女	6	温州	浙江	CN	43	2014/3/31
58877	2010/11/18	2010/11/20	女	6	PARIS	PARIS	FR	34	2014/3/31
37994	2004/11/13	2004/12/2	男	6	北京		CN	47	2014/3/31
28012	2006/11/23	2007/11/18	男	5	SAN MARIN	CA	US	58	2014/3/31
54943	2006/10/25	2007/10/27	男	6	深圳	广东	CN	47	2014/3/31
57881	2010/2/1	2010/2/1	女	6	广州	广东	CN	45	2014/3/31
1254	2008/3/28	2008/4/5	男	4	BOWLAND	CA	US	63	2014/3/31
8253	2010/7/15	2010/8/20	男	6	乌鲁木齐	新疆	CN	48	2014/3/31
58899	2010/11/10	2011/2/23	女	6	PARIS		FR	50	2014/3/31
26955	2006/4/6	2007/2/22	男	6	乌鲁木齐市	新疆	CN	54	2014/3/31
41616	2011/8/29	2011/10/22	男	6	东莞	广东	CN	41	2014/3/31
21501	2008/7/30	2008/11/21	男	6	北京		CN	49	2014/3/31
41281	2011/6/7	2011/6/9	男	6	VECHEL	NORD BRAE	AN		2014/3/31
47229	2005/4/10	2005/4/10	男	6	广州	广东	CN	69	2014/3/31
28474	2010/4/13	2010/4/13	男	6		CA	US	41	2014/3/31
58472	2010/2/14	2010/3/1	女	5			FR	48	2014/3/31





注 1:

已清洗的数据仅供本课程学习使用, 有一定的模拟性质。如需要更多的信息, 则需要从原始数据按照相应的目的进行清洗。

注 2:

CSV 格式是数据分析工作中常见的一种数据格式。CSV 意为逗号分隔值 (Comma-Separated Values), 其文件以纯文本形式存储表格数据 (数字和文本)。每行只有一条记录, 每条记录被逗号分隔符分隔为字段, 并且每条记录都有同样的字段序列。

CSV 格式能被大多数应用程序所支持, 广泛用于在不同的系统之间转移数据, 是一种容易被兼容的格式。实验楼中大量的数据分析类课程都使用了 CSV 格式的数据集, 不仅如此, 我们也推荐你在今后的数据分析工作中应用此格式来存储数据。

名称	修改日期	类型	大小
 air_data	2015/12/5 21:05	Microsoft Excel ...	14,139 KB
 zscoredata	2015/12/5 21:04	Microsoft Excel ...	4,105 KB
 zscoreddata	2015/12/5 21:04	Microsoft Excel ...	6,722 KB
 客户信息属性说明	2016/11/20 15:51	Microsoft Excel ...	29 KB

## 2、聚类案例实战

### 2.1 启动 Python 运行 Sklearn

为了更好地处理 CSV 格式的数据集，我们启动 shell 为后续观察数据作准备

请在刚刚下载的文件夹 ml-latest-small 空白处按 Shift 键同时鼠标右键打开命令行窗口，并键入 Python 进入 Python 环境。

```
PS C:\Users\jiang\Desktop\ml-latest-small> python
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
```

### 2.2 导入数据

#### 2.2.1 加载实验所需的包

首先我们需要加载本节实验所需要的包。这些包主要有：

```
请在 Python Shell 中输入以下代码：import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

上述包的使用方法可以查阅 Sklearn 的 API 手册，地址为：<http://scikit-learn.org/stable/modules/classes.html>

#### 2.2.2 观察数据集

我们用 pandas 观察数据：

接着将数据读取到程序中，并查看每个特征属性的相关信息，以便对数据进行处理。

请在 Python Shell 中输入以下代码：

```
datafile = "air_data.csv"
data = pd.read_csv(datafile, encoding="utf-8")
print(data.shape)
print(data.info())
print(data[0:5])
```

```
chapter7\demo\code> python 1.py
  MEMBER_NO  FFP_DATE  FIRST_FLIGHT_DATE  ...  Ration_P1Y_BPS  Ration_L1Y_BPS  Point_NotFlight
0         54993  2006/11/02      2008/12/24  ...           0.487221           0.512777           50
1         28065  2007/02/19      2007/08/03  ...           0.489289           0.510708           33
2         55106  2007/02/01      2007/08/30  ...           0.481467           0.518530           26
3         21189  2008/08/22      2008/08/23  ...           0.551722           0.448275           12
4         39546  2009/04/10      2009/04/15  ...           0.469054           0.530943           39
```

通过观测可知，数据集中存在票价为零但是飞行公里大于零的不合理值，但是所占比例较小，这里直接删去：

```
data = data[data["SUM_YR_1"].notnull() & data["SUM_YR_2"].notnull()]
```



```

index1 = data["SUM_YR_1"] != 0
index2 = data["SUM_YR_2"] != 0
index3 = (data["SEG_KM_SUM"] == 0) & (data["avg_discount"] == 0)
data = data[index1 | index2 | index3]
print(data.shape)

```

```

PS C:\Users\jiang\Desktop\chapter7\chapter7\demo\code> python 1.py
(62044, 44)

```

删除后剩余的样本值是 62044 个，可见异常样本的比例不足 1.5%，因此不会对分析结果产生较大的影响。

```

(62988, 44)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62988 entries, 0 to 62987
Data columns (total 44 columns):
MEMBER_NO          62988 non-null int64
FFP_DATE           62988 non-null object
FIRST_FLIGHT_DATE  62988 non-null object
GENDER             62985 non-null object
FFP_TIER           62988 non-null int64
WORK_CITY          60719 non-null object
WORK_PROVINCE      59740 non-null object
WORK_COUNTRY       62962 non-null object
AGE               62568 non-null float64
LOAD_TIME          62988 non-null object
FLIGHT_COUNT       62988 non-null int64
BP_SUM             62988 non-null int64
EP_SUM_YR_1        62988 non-null int64
EP_SUM_YR_2        62988 non-null int64
SUM_YR_1           62437 non-null float64
SUM_YR_2           62850 non-null float64
SEG_KM_SUM         62988 non-null int64
WEIGHTED_SEG_KM    62988 non-null float64
LAST_FLIGHT_DATE   62988 non-null object
AVG_FLIGHT_COUNT   62988 non-null float64
AVG_BP_SUM         62988 non-null float64
BEGIN_TO_FIRST     62988 non-null int64
LAST_TO_END        62988 non-null int64
AVG_INTERVAL       62988 non-null float64

```

```

AVG_INTERVAL      62988 non-null float64
MAX_INTERVAL      62988 non-null int64
ADD_POINTS_SUM_YR_1 62988 non-null int64
ADD_POINTS_SUM_YR_2 62988 non-null int64
EXCHANGE_COUNT     62988 non-null int64
avg_discount       62988 non-null float64
P1Y_Flight_Count   62988 non-null int64
L1Y_Flight_Count   62988 non-null int64
P1Y_BP_SUM         62988 non-null int64
L1Y_BP_SUM         62988 non-null int64
EP_SUM            62988 non-null int64
ADD_Point_SUM      62988 non-null int64
Eli_Add_Point_Sum  62988 non-null int64
L1Y_ELi_Add_Points 62988 non-null int64
Points_Sum         62988 non-null int64
L1Y_Points_Sum     62988 non-null int64
Ration_L1Y_Flight_Count 62988 non-null float64
Ration_P1Y_Flight_Count 62988 non-null float64
Ration_P1Y_BPS     62988 non-null float64
Ration_L1Y_BPS     62988 non-null float64
Point_NotFlight    62988 non-null int64
dtypes: float64(12), int64(24), object(8)
memory usage: 21.1+ MB
None

```

选择数据集中较重要的数据输出：

请在 Python Shell 中输入以下代码：

```

filter_data = data[["FFP_DATE", "LOAD_TIME", "FLIGHT_COUNT", "SUM_YR_1", "SUM_YR_2", "SEG_KM_SUM", "AVG_INTERVAL",
"MAX_INTERVAL", "avg_discount"]]
filter_data[0:5]

```

```

PS C:\Users\jiang\Desktop\chapter7\chapter7\demo\code> python 1.py
   FFP_DATE  LOAD_TIME  FLIGHT_COUNT  ...  AVG_INTERVAL  MAX_INTERVAL  avg_discount
0  2006/11/02  2014/03/31      210      ...      3.483254      18      0.961639
1  2007/02/19  2014/03/31      140      ...      5.194245      17      1.252314
2  2007/02/01  2014/03/31      135      ...      5.298507      18      1.254676
3  2008/08/22  2014/03/31       23      ...     27.863636      73      1.090870
4  2009/04/10  2014/03/31     152      ...      4.788079      47      0.970658

[5 rows x 9 columns]

```



	FFP_DATE	LOAD_TIME	FLIGHT_COUNT	SUM_YR_1	SUM_YR_2	SEG_KM_SUM	AVG_INTERVAL	MAX_INTERVAL	avg_discount
0	2006/11/02	2014/03/31	210	239560.0	234188.0	580717	3.483254	18	0.961639
1	2007/02/19	2014/03/31	140	171483.0	167434.0	293678	5.194245	17	1.252314
2	2007/02/01	2014/03/31	135	163618.0	164982.0	283712	5.298507	18	1.254676
3	2008/08/22	2014/03/31	23	116350.0	125500.0	281336	27.863636	73	1.090870
4	2009/04/10	2014/03/31	152	124560.0	130702.0	309928	4.788079	47	0.970658

对特征进行变换：

请在 Python Shell 中输入以下代码：

```
filter_data = data[["FFP_DATE", "LOAD_TIME", "FLIGHT_COUNT", "SUM_YR_1", "SUM_YR_2", "SEG_KM_SUM", "AVG_INTERVAL",
"MAX_INTERVAL", "avg_discount"]]
filter_data[0:5]
data["LOAD_TIME"] = pd.to_datetime(data["LOAD_TIME"])
data["FFP_DATE"] = pd.to_datetime(data["FFP_DATE"])
data["入会时间"] = data["LOAD_TIME"] - data["FFP_DATE"]
data["平均每公里票价"] = (data["SUM_YR_1"] + data["SUM_YR_2"]) / data["SEG_KM_SUM"]
data["时间间隔差值"] = data["MAX_INTERVAL"] - data["AVG_INTERVAL"]
deal_data = data.rename(
    columns = {"FLIGHT_COUNT": "飞行次数", "SEG_KM_SUM": "总里程", "avg_discount": "平均折扣率"},
    inplace = False
)
filter_data = deal_data[["入会时间", "飞行次数", "平均每公里票价", "总里程", "时间间隔差值", "平均折扣率"]]
print(filter_data[0:5])
filter_data['入会时间'] = filter_data['入会时间'].astype(np.int64)/(60*60*24*10**9)
print(filter_data[0:5])
print(filter_data.info())
```

```

    入会时间 飞行次数 平均每公里票价 总里程 时间间隔差值 平均折扣率
0 2706 days 210 0.815798 580717 14.516746 0.961639
1 2597 days 140 1.154043 293678 11.805755 1.252314
2 2615 days 135 1.158217 283712 12.701493 1.254676
3 2047 days 23 0.859648 281336 45.136364 1.090870
4 1816 days 152 0.823617 309928 42.211921 0.970658
1.py:31: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
filter_data['入会时间'] = filter_data['入会时间'].astype(np.int64)/(60*60*24*10**9)
    入会时间 飞行次数 平均每公里票价 总里程 时间间隔差值 平均折扣率
0 2706.0 210 0.815798 580717 14.516746 0.961639
1 2597.0 140 1.154043 293678 11.805755 1.252314
2 2615.0 135 1.158217 283712 12.701493 1.254676
3 2047.0 23 0.859648 281336 45.136364 1.090870
4 1816.0 152 0.823617 309928 42.211921 0.970658
<class 'pandas.core.frame.DataFrame'>
Int64Index: 62044 entries, 0 to 62978
Data columns (total 6 columns):
入会时间      62044 non-null float64
飞行次数      62044 non-null int64
平均每公里票价 62044 non-null float64
总里程        62044 non-null int64
时间间隔差值  62044 non-null float64
平均折扣率    62044 non-null float64
dtypes: float64(4), int64(2)
memory usage: 3.3 MB
None

```

由于不同的属性相差范围较大，这里进行标准化处理：

请在 Python Shell 中输入以下代码：

```

filter_zscore_data = (filter_data - filter_data.mean(axis=0))/(filter_data.std(axis=0))
filter_zscore_data[0:5]

kmodel = KMeans(n_clusters=4, n_jobs=4)

```

```

    入会时间 飞行次数 平均每公里票价 总里程 时间间隔差值 平均折扣率
0 2706.0 210 0.815798 580717 14.516746 0.961639
1 2597.0 140 1.154043 293678 11.805755 1.252314
2 2615.0 135 1.158217 283712 12.701493 1.254676
3 2047.0 23 0.859648 281336 45.136364 1.090870
4 1816.0 152 0.823617 309928 42.211921 0.970658

```

## 2.2.3 寻找 K 值和肘点

对于 K-Means 方法，k 的取值是一个难点，因为是无监督的聚类分析问题，所以不寻在绝对正确的值，需要进行研究试探。这里采用计算 SSE 的方法，尝试找到最好的 K 数值。编写函数如下：

请在 Python IDE 中输入以下代码

```

def distEclud(vecA, vecB):
    # 计算两个向量的欧式距离的平方，并返回

    return np.sum(np.power(vecA - vecB, 2))

def test_Kmeans_nclusters(data_train):
    #计算不同的 k 值时，SSE 的大小变化
    data_train = data_train.values
    nums=range(2,10)
    SSE = []
    for num in nums:

```



```

sse = 0
kmodel = KMeans(n_clusters=num, n_jobs=4)
kmodel.fit(data_train)
# 簇中心
cluster_ceter_list = kmodel.cluster_centers_
# 个样本属于的簇序号列表
cluster_list = kmodel.labels_.tolist()
for index in range(len(data)):
    cluster_num = cluster_list[index]
    sse += distEclud(data_train[index, :], cluster_ceter_list[cluster_num])
print("簇数是", num, "时; SSE 是", sse)
SSE.append(sse)
return nums, SSE
if __name__ == '__main__':
    nums, SSE = test_Kmeans_nclusters(filter_zscore_data)

```

输出为:

```

簇数是 2 时; SSE 是 296587.688611
簇数是 3 时; SSE 是 245317.292202
簇数是 4 时; SSE 是 209299.798194
簇数是 5 时; SSE 是 183885.938906
簇数是 6 时; SSE 是 167465.10385
簇数是 7 时; SSE 是 151869.163041
簇数是 8 时; SSE 是 142922.824005
簇数是 9 时; SSE 是 135003.92238

```

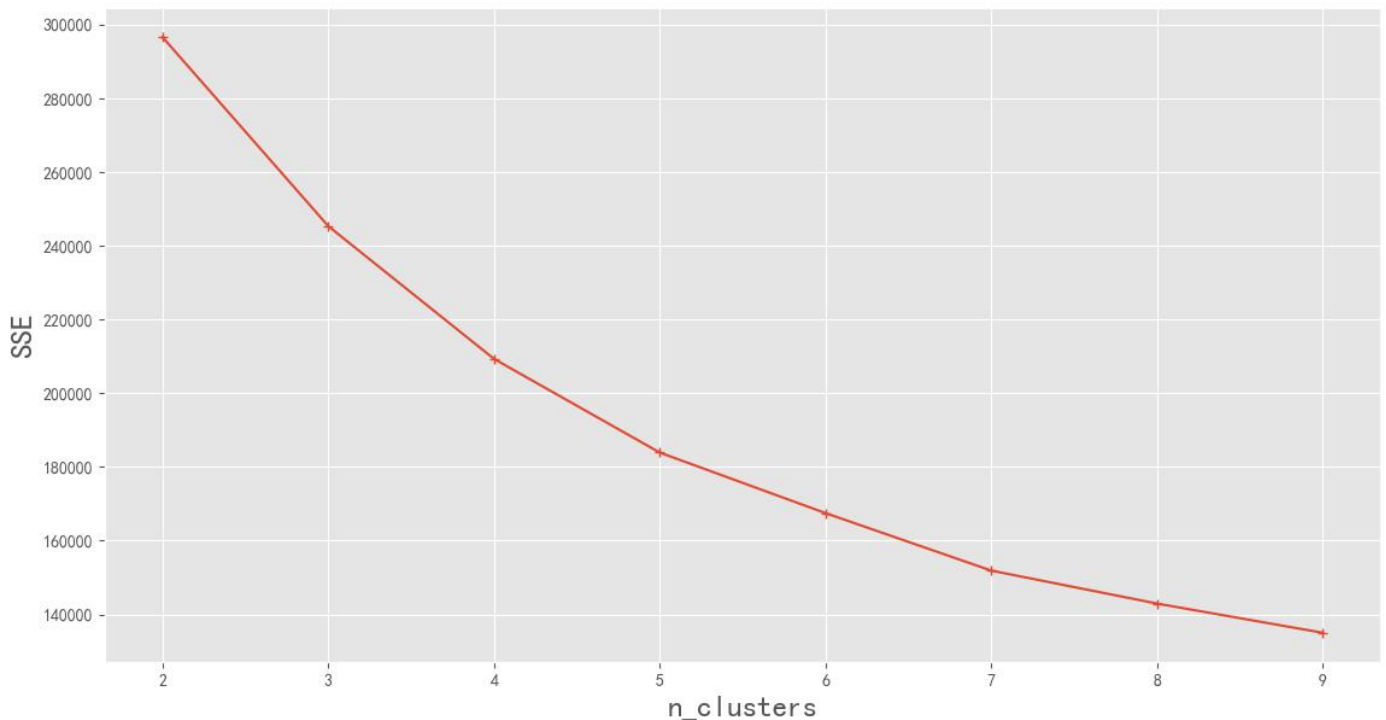
## 2.2.4 选择合适的 K 值并做图

通过观察 SSE 与 k 的取值尝试找出合适的 k 值

```

请在 Python IDE 中输入以下代码。
#画图，通过观察 SSE 与 k 的取值尝试找出合适的 k 值
# 中文和负号的正常显示
plt.rcParams['font.sans-serif'] = 'SimHei'
plt.rcParams['font.size'] = 12.0
plt.rcParams['axes.unicode_minus'] = False
# 使用 ggplot 的绘图风格
plt.style.use('ggplot')
## 绘图观测 SSE 与簇个数的关系
fig=plt.figure(figsize=(10, 8))
ax=fig.add_subplot(1,1,1)
ax.plot(nums,SSE,marker="+")
ax.set_xlabel("n_clusters", fontsize=18)
ax.set_ylabel("SSE", fontsize=18)
fig.suptitle("KMeans", fontsize=20)
plt.show()

```



观察图像，并没有所谓的“肘”点出现，是随 k 值的增大逐渐减小的，这里选取当 k 分别取 4，5，6 时进行，看能不能通过分析结果来反向选取更合适的值，k 取值 4 时的代码如下：

请在 Python IDE 中输入以下代码：

```
kmodel = KMeans(n_clusters=4, n_jobs=4)

kmodel.fit(filter_zscore_data)

# 简单打印结果
r1 = pd.Series(kmodel.labels_).value_counts() #统计各个类别的数目
r2 = pd.DataFrame(kmodel.cluster_centers_) #找出聚类中心
# 所有簇中心坐标值中最大值和最小值
max = r2.values.max()
min = r2.values.min()

r = pd.concat([r2, r1], axis = 1) #横向连接（0 是纵向），得到聚类中心对应的类别下的数目
r.columns = list(filter_zscore_data.columns) + ['类别数目'] #重命名表头

# 绘图
fig=plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, polar=True)
center_num = r.values
feature = ["入会时间", "飞行次数", "平均每公里票价", "总里程", "时间间隔差值", "平均折扣率"]
N =len(feature)
for i, v in enumerate(center_num):
    # 设置雷达图的角度，用于平分切开一个圆面
    angles=np.linspace(0, 2*np.pi, N, endpoint=False)
    # 为了使雷达图一圈封闭起来，需要下面的步骤
    center = np.concatenate((v[:-1],[v[0]]))
    angles=np.concatenate((angles,[angles[0]]))
    # 绘制折线图
    ax.plot(angles, center, 'o-', linewidth=2, label = "第%d 簇人群,%d 人"%(i+1,v[-1]))
```

```

# 填充颜色
ax.fill(angles, center, alpha=0.25)

# 添加每个特征的标签
ax.set_thetagrids(angles * 180/np.pi, feature, fontsize=15)

# 设置雷达图的范围
ax.set_ylim(min-0.1, max+0.1)

# 添加标题
plt.title('客户群特征分析图', fontsize=20)

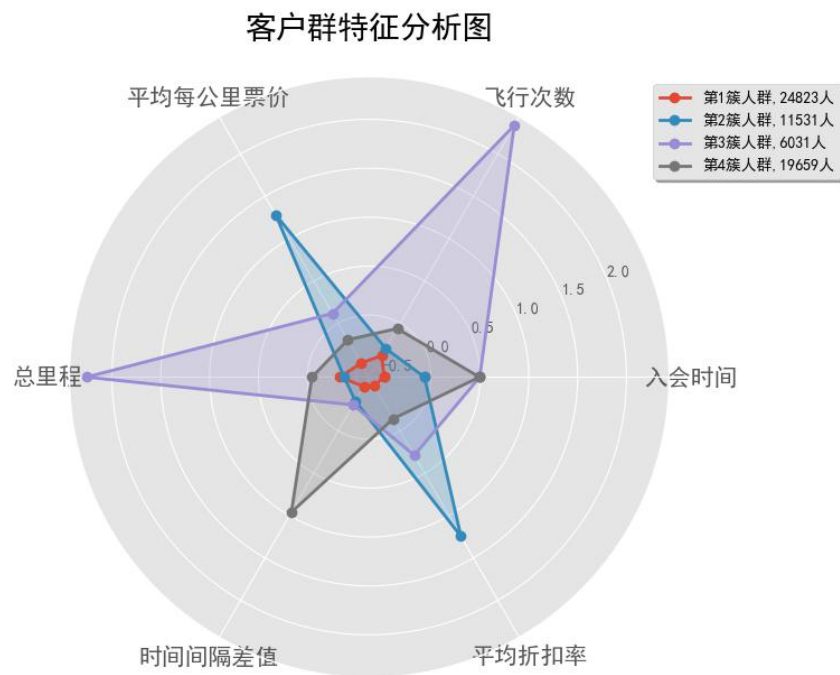
# 添加网格线
ax.grid(True)

# 设置图例
plt.legend(loc='upper right', bbox_to_anchor=(1.3,1.0),ncol=1,fancybox=True,shadow=True)

# 显示图形
plt.show()

```

Figure 1

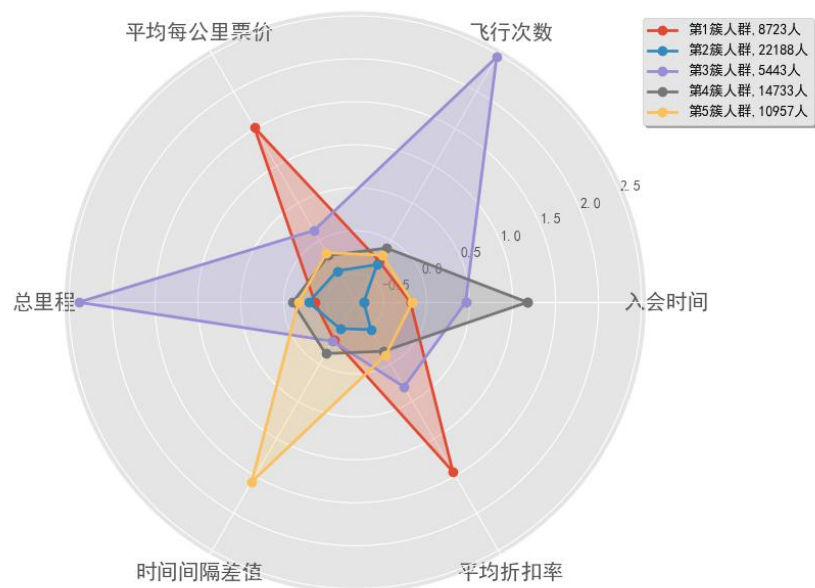


k 取值 5, 6 时的代码与上述类似, 不再给出, 直接给出结果图:

k=5 如下图所示:

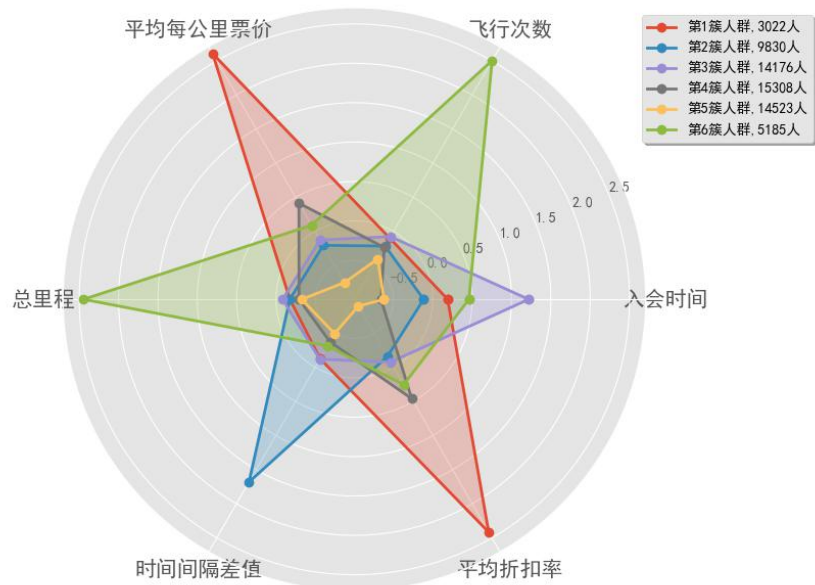


客户群特征分析图



K=6 如下图所示:

客户群特征分析图



---

## 五、K-means 聚类算法原理

### 5.1 K-means 算法简介

---

选择 K 个点作为初始质心

repeat

    将每个点指派到最近的质心，形成 K 个簇

    重新计算每个簇的质心

until 簇不发生变化或达到最大迭代次数

时间复杂度： $O(tKmn)$ ，其中，t 为迭代次数，K 为簇的数目，m 为记录数，n 为维数

空间复杂度： $O((m+K)n)$ ，其中，K 为簇的数目，m 为记录数，n 为维数

### 5.2 K 值如何确定

---

kmeans 算法首先选择 K 个初始质心，其中 K 是用户指定的参数，即所期望的簇的个数。这样做的前提是我们已经知道数据集中包含多少个簇，但很多情况下，我们并不知道数据的分布情况，实际上聚类就是我们发现数据分布的一种手段：

#### 1.与层次聚类结合[2]

经常会产生较好的聚类结果的一个有趣策略是，首先采用层次凝聚算法决定结果粗的数目，并找到一个初始聚类，然后用迭代重定位来改进该聚类。

#### 2.稳定性方法[3]

稳定性方法对一个数据集进行 2 次重采样产生 2 个数据子集，再用相同的聚类算法对 2 个数据子集进行聚类，产生 2 个具有 k 个聚类的聚类结果，计算 2 个聚类结果的相似度的分布情况。2 个聚类结果具有高的相似度说明 k 个聚类反映了稳定的聚类结构，其相似度可以用来估计聚类个数。采用该方法试探多个 k，找到合适的 k 值。

#### 3.系统演化方法[3]

系统演化方法将一个数据集视为伪热力学系统，当数据集被划分为 K 个聚类时称系统处于状态 K。系统由初始状态  $K=1$  出发，经过分裂过程和合并过程，系统将演化到它的稳定平衡状态  $K_i$ ，其所对应的聚类结构决定了最优类数  $K_i$ 。系统演化方法能提供关于所有聚类之间的相对边界距离或可分程度，它适用于明显分离的聚类结构和轻微重叠的聚类结构。

#### 4.使用 canopy 算法进行初始划分[4]

基于 Canopy Method 的聚类算法将聚类过程分为两个阶段

Stage1、聚类最耗费计算的地方是计算对象相似性的时候，Canopy Method 在第一阶段选择简单、计算代价较低的方法计算对象相似性，将相似的对象放在一个子集中，这个子集被叫做 Canopy，通过一系列计算得到若干 Canopy，Canopy 之间可以是重叠的，但不会存在某个对象不属于任何 Canopy 的情况，可以把这一阶段看做数据预处理；

Stage2、在各个 Canopy 内使用传统的聚类方法(如 K-means)，不属于同一 Canopy 的对象之间不进行相似性计算。

从这个方法起码可以看出两点好处：首先，Canopy 不要太大且 Canopy 之间重叠的不要太多的话会大大减少后续需要计算相似性的对象的个数；其次，类似于 K-means 这样的聚类方法是需要人为指出 K 的值的，通过 Stage1 得到的 Canopy 个数完全可以作为这个 K 值，一定程度上减少了选择 K 的盲目性。

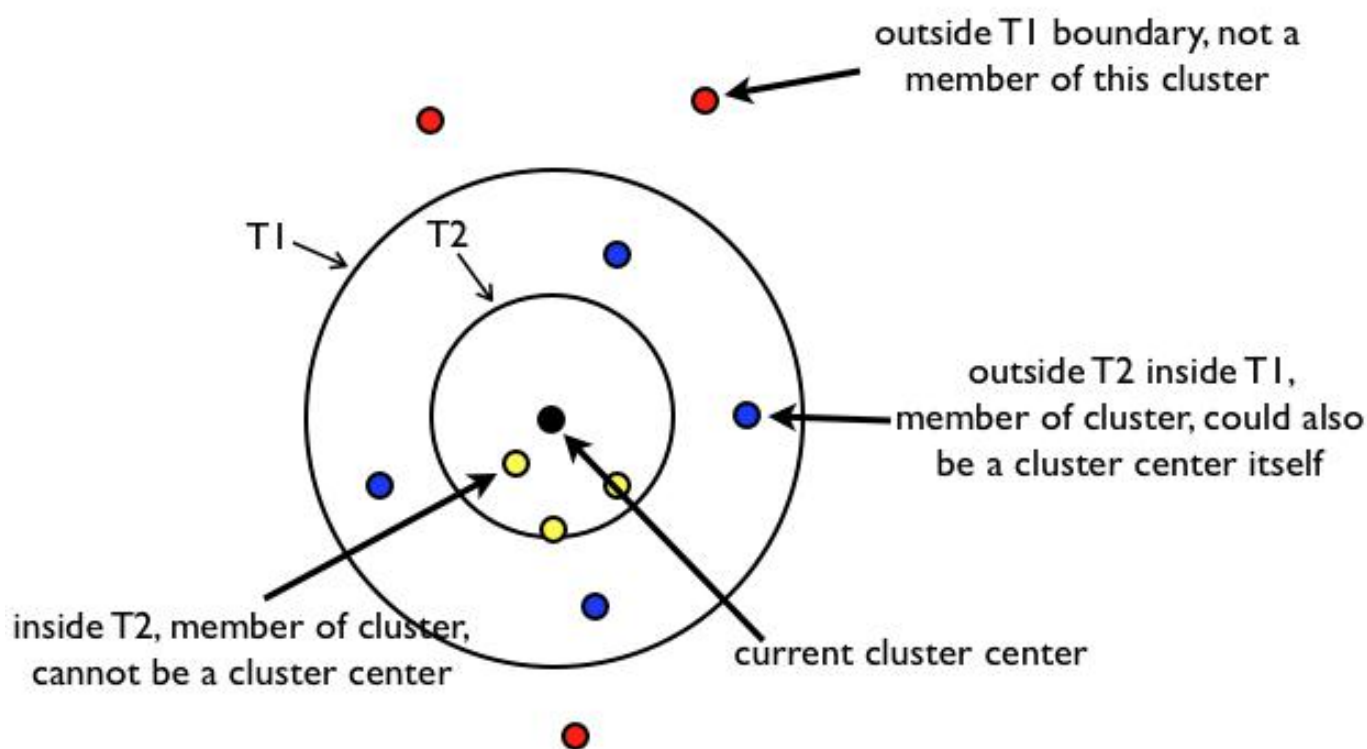
## 5.3 初始质心的选取

选择适当的初始质心是基本 kmeans 算法的关键步骤。常见的方法是随机的选取初始质心，但是这样簇的质量常常很差。处理选取初始质心问题的一种常用技术是：多次运行，每次使用一组不同的随机初始质心，然后选取具有最小 SSE（误差的平方和）的簇集。这种策略简单，但是效果可能不好，这取决于数据集和寻找的簇的个数。

第二种有效的方法是，取一个样本，并使用层次聚类技术对它聚类。从层次聚类中提取 K 个簇，并用这些簇的质心作为初始质心。该方法通常很有效，但仅对下列情况有效：（1）样本相对较小，例如数百到数千（层次聚类开销较大）；（2）K 相对于样本大小较小。

第三种选择初始质心的方法，随机地选择第一个点，或取所有点的质心作为第一个点。然后，对于每个后继初始质心，选择离已经选取过的初始质心最远的点。使用这种方法，确保了选择的初始质心不仅是随机的，而且是散开的。但是，这种方法可能选中离群点。此外，求离当前初始质心集最远的点开销也非常大。为了克服这个问题，通常该方法用于点样本。由于离群点很少（多了就不是离群点了），它们多半不会在随机样本中出现。计算量也大幅减少。

第四种方法就是上面提到的 canopy 算法。



### (4) 质心的计算

对于距离度量不管是采用欧式距离还是采用余弦相似度，簇的质心都是其均值，即向量各维取平均即可。对于距离度量采用其它方式时，这个还没研究过。

注:

在机器学习算法应用过程中，经常面临的工作便是为机器学习算法提供的 API 的各项参数“制备”数据。使用 `from sklearn.model_selection import train_test_split` 将 `train_test_split` 导入：

`sklearn.model_selection.train_test_split` 随机划分训练集和测试集：

`train_test_split` 是交叉验证中常用的函数，功能是从样本中随机的按比例选取 train data 和 testdata，形式为：



```
X_train,X_test, y_train, y_test =cross_validation.train_test_split(train_data,train_target,test_size=0.4, random_state=0)
```

train\_data: 所要划分的样本特征集

train\_target: 所要划分的样本结果

test\_size: 样本占比，如果是整数的话就是样本的数量

random\_state: 随机数的种子。

随机数种子: 其实就是该组随机数的编号，在需要重复试验的时候，保证得到一组一样的随机数。比如你每次都填 1，其他参数一样的情况下你得到的随机数组是一样的。但填 0 或不填，每次都会不一样。

随机数的产生取决于种子，随机数和种子之间的关系遵从以下两个规则：

- 1, 种子不同，产生不同的随机数；
- 2, 种子相同，即使实例不同也产生相同的随机数。

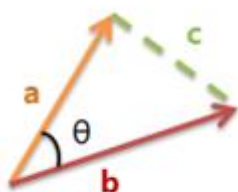
## 5.4 距离的度量

常用的距离度量方法包括：欧几里得距离和余弦相似度。两者都是评定个体间差异的大小的。欧几里得距离度量会受指标不同单位刻度的影响，所以一般需要先进行标准化，同时距离越大，个体间差异越大；空间向量余弦夹角的相似度量不会受指标刻度的影响，余弦值落于区间 $[-1,1]$ ，值越大，差异越小。但是针对具体应用，什么情况下使用欧氏距离，什么情况下使用余弦相似度？

从几何意义上来说， $n$  维向量空间的一条线段作为底边和原点组成的三角形，其顶角大小是不确定的。也就是说对于两条空间向量，即使两点距离一定，他们的夹角余弦值也可以随意变化。感性的认识，当两用户评分趋势一致时，但是评分值差距很大，余弦相似度倾向给出更优解。举个极端的例子，两用户只对两件商品评分，向量分别为 $(3,3)$ 和 $(5,5)$ ，这两位用户的认知其实是一样的，但是欧式距离给出的解显然没有余弦值合理。[6]

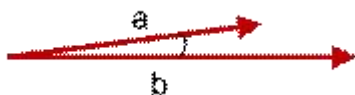
以余弦相似度为例来计算用户之间的相似度关系：

余弦相似度，又称为余弦相似性，是通过计算两个向量的夹角余弦值来评估他们的相似度。求得他们的夹角，并得出夹角对应的余弦值，此余弦值就可以用来表征，这两个向量的相似性。

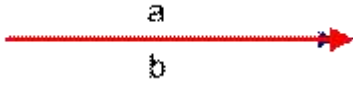


夹角越小，余弦值越接近于 1，它们的方向更加吻合，则越相似。余弦值的范围在 $[-1,1]$ 之间，值越趋近于 1，代表两个向量的方向越接近；越趋近于-1，他们的方向越相反；接近于 0，表示两个向量近乎于正交。

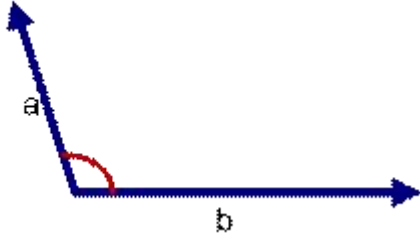
[Source ]<https://blog.csdn.net/u012160689/article/details/15341303>



上图两个向量  $a, b$  的夹角很小可以说  $a$  向量和  $b$  向量有很高的相似性，极端情况下， $a$  和  $b$  向量完全重合。如下图：



如上图：可以认为 a 和 b 向量是相等的，也即 a, b 向量代表的文本是完全相似的，或者说是相等的。如果 a 和 b 向量夹角较大，或者反方向。如下图：



如上图：两个向量 a,b 的夹角很大可以说 a 向量和 b 向量有很低的相似性，或者说 a 和 b 向量代表的事物基本不相似。

向量 a 和向量 b 的夹角的余弦计算如下：

$$\cos(\theta) = \frac{a \bullet b}{\|a\| \times \|b\|} = \frac{(x_1, y_1) \bullet (x_2, y_2)}{\sqrt{x_1^2 + y_1^2} \times \sqrt{x_2^2 + y_2^2}} = \frac{x_1 y_1 + x_2 y_2}{\sqrt{x_1^2 + y_1^2} \times \sqrt{x_2^2 + y_2^2}}$$

扩展，如果向量 a 和 b 不是二维而是 n 维，上述余弦的计算法仍然正确。假定 a 和 b 是两个 n 维向量，a 是  $(x_1, x_2, \dots, x_n)$ ，b 是  $(y_1, y_2, \dots, y_n)$ ，则 a 与 b 的夹角的余弦等于：

$$\cos(\theta) = \frac{a \bullet b}{\|a\| \times \|b\|} = \frac{\sum_{i=1}^n (x_i \times y_i)}{\sqrt{\sum_{i=1}^n (x_i)^2} \times \sqrt{\sum_{i=1}^n (y_i)^2}}$$

## 5.5 算法停止条件

一般是目标函数达到最优或者达到最大的迭代次数即可终止。对于不同的距离度量，目标函数往往不同。当采用欧式距离时，目标函数一般为最小化对象到其簇质心的距离的平方和，如下：

$$\min \sum_{i=1}^K \sum_{x \in C_i} dist(c_i, x)^2$$

当采用余弦相似度时，目标函数一般为最大化对象到其簇质心的余弦相似度和，如下：

$$\max \sum_{i=1}^K \sum_{x \in C_i} cosine(c_i, x)$$

---

## 5.6 空聚类的处理

---

如果所有的点在指派步骤都未分配到某个簇，就会得到空簇。如果这种情况发生，则需要某种策略来选择一个替补质心，否则的话，平方误差将会偏大。一种方法是选择一个距离当前任何质心最远的点。这将消除当前对总平方误差影响最大的点。另一种方法是从具有最大 SSE 的簇中选择一个替补的质心。这将分裂簇并降低聚类的总 SSE。如果有多个空簇，则该过程重复多次。另外，编程实现时，要注意空簇可能导致的程序 bug。

## 5.6 适用范围及缺陷

---

Kmeans 算法试图找到使平方误差准则函数最小的簇。当潜在的簇形状是凸面的，簇与簇之间区别较明显，且簇大小相近时，其聚类结果较理想。前面提到，该算法时间复杂度为  $O(tkmn)$ ，与样本数量线性相关，所以，对于处理大数据集合，该算法非常高效，且伸缩性较好。但该算法除了要事先确定簇数  $k$  和对初始聚类中心敏感外，经常以局部最优结束，同时对“噪声”和孤立点敏感，并且该方法不适用于发现非凸面形状的簇或大小差别很大的簇。

## 六、项目总结

通过观察可知：

当  $k$  取值 4 时，每个人群包含的信息比较复杂，且特征不明显

当  $k$  取值 5 时，分析的结果比较合理，分出的五种类型人群都有自己的特点又不相互重复

当  $k$  取值 6 时，各种人群也都有自己的特点，但是第 4 簇人群完全在第 5 簇人群特征中包含了，有点冗余

综上，当  $k$  取值为 5 时，得到最好的聚类效果，将所有的客户分成 5 个人群，再进一步分析可以得到以下结论：

1. 第一簇人群，10957 人，最大的特点是时间间隔差值最大，分析可能是“季节型客户”，一年中在某个时间段需要多次乘坐飞机进行旅行，其他的时间则出行的不多，这类客户我们需要在保持的前提下，进行一定的发展；
2. 第二簇人群，14732 人，最大的特点就是入会的时间较长，属于老客户按理说平均折扣率应该较高才对，但是观察窗口的平均折扣率较低，而且总里程和总次数都不高，分析可能是流失的客户，需要在争取一下，尽量让他们“回心转意”；
3. 第三簇人群，22188 人，各方面的数据都是比较低的，属于一般或低价值用户
4. 第四簇人群，8724 人，最大的特点就是平均每公里票价和平均折扣率都是最高的，应该是属于乘坐高等舱的商务人员，应该重点保持的对象，也是需要重点发展的对象，另外应该积极采取相关的优惠政策是他们的乘坐次数增加
5. 第五簇人群，5443 人，总里程和飞行次数都是最多的，而且平均每公里票价也较高，是重点保持对象

分析完毕，结果暗合市场的二八法则的，价值不大的第二三簇的客户数最多，而价值较大的第四五簇的人数较少。

## 七、常见问题

Q：本案例中 `if __name__ == '__main__':` 这段代码如何理解？有什么作用。



---

A:

1) 首先: 对于很多编程语言来说, 程序都必须要有个入口, 比如 C, C++, 以及完全面向对象的编程语言 Java, C#等。如果你接触过这些语言, 对于程序入口这个概念应该很好理解, C, C++都需要有一个 main 函数作为程序的入口, 也就是程序的运行会从 main 函数开始。同样, Java, C#必须要有一个包含 Main 方法的主类, 作为程序入口。而 Python 则不同, 它属于脚本语言, 不像编译型语言那样先将程序编译成二进制再运行, 而是动态的逐行解释运行。也就是从脚本第一行开始运行, 没有统一的入口。

2) 一个 Python 源码文件 (.py) 除了可以被直接运行外, 还可以作为模块(也就是库), 被其他.py 文件导入。不管是直接运行还是被导入, .py 文件的最顶层代码都会被运行(Python 用缩进来区分代码层次), 而当一个.py 文件作为模块被导入时, 我们可能不希望一部分代码被运行。

3) 举例: 通俗地讲, `__name__ == '__main__':` 假如你叫小明.py, 在朋友眼中, 你是小明(`__name__ == '小明'`); 在你自己眼中, 你是你自己(`__name__ == '__main__'`)。

4) `if __name__ == '__main__':` 的意思是: 当.py 文件被直接运行时, `if __name__ == '__main__'` 之下的代码块将被运行; 当.py 文件以模块形式被导入时, `if __name__ == '__main__'` 之下的代码块不被运行。

## 八、参考文献及 blog

[1]Pang-Ning Tan 等著, 《数据挖掘导论》, 2011

[2]Jiawei Han 等著, 《数据挖掘概念与技术》, 2008

[3]聚类分析中类数估计方法的实验比较

[4]<http://www.cnblogs.com/vivounicorn/archive/2011/09/23/2186483.html>

[5]一种基于贝叶斯信息准则的 k 均值聚类方法

[6]<https://blog.csdn.net/qli125596718/article/details/8243404/>

[7]张良均、王路等著《python 数据分析与挖掘实战》

[8]<https://blog.csdn.net/a857553315/article/details/79177524>