西安电子科技大学

安全前沿讨论班（Ⅰ） 课程实验报告

实验名称 使用 K-Means 聚类进行青少年市场细分

学　院 网络与信息安全学院　　班 级 1618019

姓　名 曹寅峰　　学 号 16020610025

实验日期 _____ 年 _____ 月_____日

指导教师评语：   指导教师： _____年____月____日

实验报告内容基本要求及参考格式

一、实验目的

通过本次实验，掌握 K-Means 聚类方法，了解如何使用 Python 中的 K-Means 模块进行数据分析，加深对 pandas 库、numpy 库、sklearn 库的了解，掌握这些库中常见的数据分析方法，理解 K-Means 进行数据分析的基本步骤。

1. 任务 1：利用 pandas 库加载数据并查看数据；
2. 任务 2：处理数据中的缺失值并对数据进行标准化；
3. 任务 3：应用 K-Means 模块训练模型并对结果进行分析。

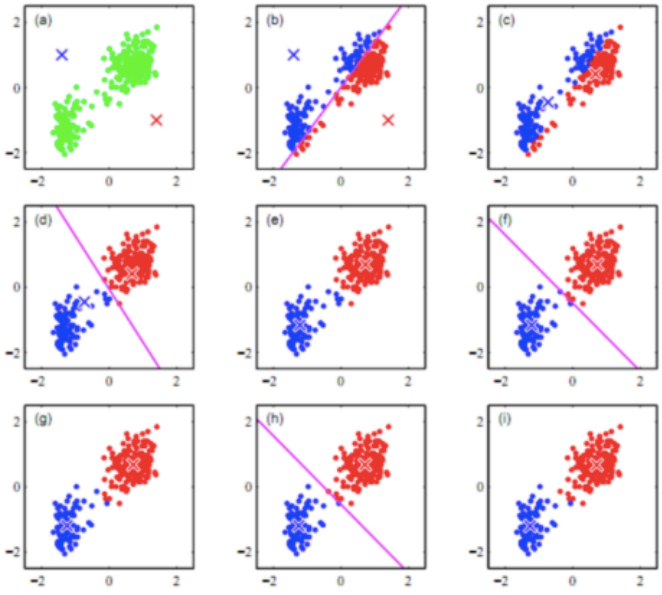
二、实验环境

Anaconda

三、实验基本原理及步骤（或方案设计及理论计算）

1. K-Means 算法的原理和流程；

1. 在特征空间中选择k个点作为初始质心（通常是随机选择）
2. 计算每个样本到各个簇质心的距离（欧式距离等距离度量）
3. 将样本分配到距离自己最近的质心所代表的簇
4. 以簇内样本的均值更新各簇的质心
5. 重复 2-4 优化过程，直至样本的分配和质心不再改变，（或迭代次数超过某个值）

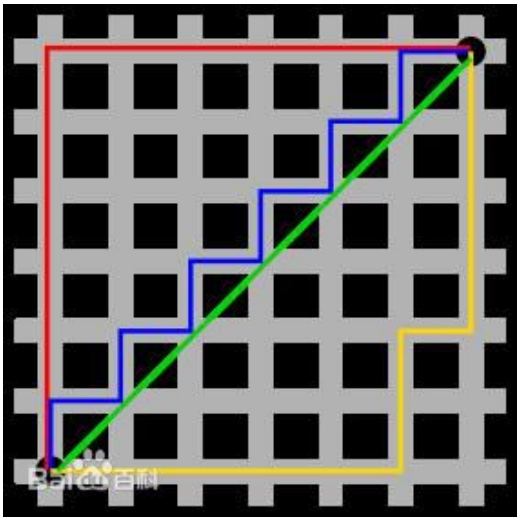


2. 质心距离度量方式，如欧氏距离、曼哈顿距离、马氏距离、海明距离、余弦相似度和杰卡德距离等；

欧式距离：欧几里得距离或欧几里得度量是欧几里得空间中两点间“普通”（即直线）距离。使用这个距离，欧氏空间成为度量空间，这也是我们最早接触的距离公式，比较好理解

$$dis(X_i, C_j) = \sqrt{\sum_{t=1}^{m}(X_{it} - C_{jt})^2}$$

曼哈顿距离：表示两个点在标准坐标系上的绝对轴距之和



图中红线代表曼哈顿距离，绿色代表欧氏距离，也就是直线距离，而蓝色和黄色代表等价的曼哈顿距离。曼哈顿距离——两点在南北方向上的距离加上在东西方向上的距离，即 d（i，j）=|xi-xj|+|yi-yj|

马氏距离：马氏距离是由印度统计学家马哈拉诺比斯提出的，表示数据的协方差距离。它是一种有效的计算两个未知样本集的相似度的方法。与欧氏距离不同的是它考虑到各种特性之间的联系，即独立于测量尺度。 对于一个均值为

$$\mu = (\mu_1, \mu_2, \mu_3, \ldots, \mu_p)^T$$

，协方差矩阵为 Σ 的多变量矢量

$$x = (x_1, x_2, x_3, \ldots, x_p)^T$$

，其马氏距离为

$$D_M(x) = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

海明距离：在信息编码中，两个合法代码对应位上编码不同的位数称为码距，又称海明距离。举例如下：10101 和 00110 从第一位开始依次有第一位、第四、第五位不同，则海明距离为 3。

余弦相似度：余弦距离，也称为余弦相似度，是用向量空间中两个向量夹角的余弦值作为衡量两个个体间差异的大小的度量。
余弦值越接近 1，就表明夹角越接近 0 度，也就是两个向量越相似，这就叫"余弦相似性"

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}.$$

杰卡德相似度系数：Jaccard（杰卡德）相似性系数主要用于计算符号度量或布尔值度量的样本间的相似度。若样本间的特征属性由符号和布尔值标识，无法衡量差异具体值的大小，只能获得“是否相同”这样一种结果，而 Jaccard 系数关心的是样本间共同具有的特征。

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

杰卡德距离：与 Jaccard 系数相反的概念是 Jaccard 距离，用两个集合中不同元素所占元素的比例来衡量两个集合（样本）的区分度，可用如下公式表示：

$$J_\delta(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

皮尔逊相关系数：Pearson 相关系数是用协方差除以两个变量的标准差得到的，虽然协方差能反映两个随机变量的相关程度（协方差大于 0 的时候表示两者正相关，小于 0 的时候表示两者负相关），但其数值上受量纲的影响很大，不能简单地从协方差的数值大小给出变量相关程度的判断。为了消除这种量纲的影响，于是就有了相关系数的概念。

当两个变量的方差都不为零时，相关系数才有意义，相关系数的取值范围为[-1,1]

3. 聚类结果评价指标。

对于一个聚类的结果，有两种评价的指标，一种是外部指标，一种是内部指标。

聚类评估：

簇的内部衡量指标

紧凑度：衡量一个簇样本点之间的是否足够紧凑。

分离度：衡量该样本是否到其他簇的距离是否足够远

簇的外部衡量指标

将聚类结果与某个“参考模型”进行比较

介绍

聚类衡量指标
- 内部
  - 轮廓系数引用
  - Calinski – Harabaz-Index
- 外部
  - Homogeneity、completeness　V-measure
  - Mutual Information

1.轮廓系数引用（内部）

簇内不相似度：计算样本 i 到同簇其他样本的平均距离 a_i，a_i　越小，表示样本点越应该聚类到该簇，簇 C 的样本均值被称为簇 C 的簇不相似度

簇间不相似度：计算该簇所有样本 i 到其他簇样本中心点 C_j 的的平均距离 $\bar{b}_{ij}$，

$b_{ij} = \min(\bar{b}_{i1}, \bar{b}_{i2}, ..., \bar{b}_{ik})$，$b_{ij}$ 越大，表示样本 i 越不属于其它簇

轮廓系数： S_i越接近 1，表示样本 i 聚类越合理；越接近-1，表示样本 i 应该分类到另外的簇中；近似为 0，表示样本 i 应该在边界上；所有样本S_i的均值被成为聚类结果的轮廓系数：

$S(i)=\frac{b(i)-a(i)}{\max\{a(i),b(i)\}}$， $S(i)$的取值： $S(i)=\begin{cases}1-\frac{a(i)}{b(i)}, a(i)<b(i)\\ 0, a(i)=b(i)\\ \frac{b(i)}{a(i)}-1, a(i)>b(i)\end{cases}$

2.Homogeneity,completeness and V-measure（外部）

同质性（homogeneity）：一个簇中只包含一个类别的样本，则满足同质性。可以认为是分类算法衡量的精确度（每个聚类簇中正确分类的样本数占该聚类簇中的样本数）

$p=\frac{1}{k}\sum_{i=1}^{k}\frac{N(C_i==K_i)}{N(K_i)}$，即每个簇包含一个类成员，$N(K_i)$表示第 i 簇的样本簇，$N(C_i==K_i)$表示第 i 簇中正确分类的样本数

完整性（completeness）：同类别样本被归类到同一聚类簇中，则满足完整性，可以认为是分类算法衡量的召回率（每个聚类中正确分类样本数占该类别样本的数量）

$r=\frac{1}{k}\sum_{i=1}^{k}\frac{N(C_i==K_i)}{N(C_i)}$，即给定的所有成员都分配给同一簇，$N(C_i)$代表第 i 类别的样本数，$N(C_i==K_i)$表示第 i 簇中正确分类的样本数

V-measure:均一性和完整性的加权平均（为均一性和完整性的调和平均）

$V_\beta=\frac{(1+\beta^2).pr}{\beta^2.p+r}$，$\beta$为加权值

3.互信息（K-L 离散）—引用 KNN 中分支条件（熵）熵越小越好—基尼指数（外部）

该衡量**两个数据分布**的吻合程度。假设 U 和 V 是对 N 个样本标签的分布情况，则两种熵的表示：

$$H(U)=-\sum_{i=1}^{|U|}p(i)\log p(i), H(V)=-\sum_{j=1}^{|V|}p'(j)\log p'(j)$$

其中 $p(i)=|U_i|/N, p'(j)=|V_j|/N$。U 和 V 之间的互信息（MI）定义为：

$$MI(U,V)=\sum_{i=1}^{|U|}\sum_{j=1}^{|V|}P(i,j)\log\left(\frac{P(i,j)}{p(i)p'(j)}\right)$$，越小越相似

4.Calinski-Harabaz Index（内部）

得到 Calinski-Harabaz Index 分值 s 越大则，聚类效果越好。

对于第 k 个簇，$s(k)=\frac{Tr(B_k)}{Tr(W_k)}\times\frac{N-k}{K-1}$，其中 N 为训练样本数，Tr 为矩阵的迹：

$W_k=\sum_{q=1}^{k}\sum_{x\in C_q}(x-c_q)(x-c_q)^T$，$W_k$ 为类别内部数据的协方差矩阵。

$B_k=\sum_q n_q(c_q-c)(c_q-c)^T$，$B_k$ 为类别之间的协方差。

C_q 为 q 簇的集合点，c_q 为 q 簇的中心点，c 为 E 的中心点，n_q 为 q 簇的中心。也就是说，类别内部数据的协方差越小越好，类别之间的协方差越大越好。

实际上，对于不同的问题，选择不同的方法，要使用不同的聚类指标进行评价

四、实验数据记录（或仿真及软件设计）

```
9]: # 首先，使用pandas中的 read_csv() 函数将数据加载到数据框中,并赋值给变量teenager_sns
    import pandas as pd
    teenager_sns=pd.read_csv("./teenager_sns.csv")

0]: # 其次，查看 teenager_sns 前5行记录
    teenager_sns.head(5)
```

0]:

	gradyear	gender	age	friends	basketball	football	soccer	softball	volleyball	swimming	...	l
0	2006	M	18.980	7	0	0	0	0	0	0	...	
1	2006	F	18.801	0	0	1	0	0	0	0	...	
2	2006	M	18.335	69	0	1	0	0	0	0	...	
3	2006	F	18.875	0	0	0	0	0	0	0	...	
4	2006	NaN	18.995	10	0	0	0	0	0	0	...	

5 rows × 40 columns

首先导入 CSV 文件

```
]: # 返回 teenager_sns 的维度大小
   teenager_sns.shape

]: (30000, 40)
```

查看维度

```
# 使用value_counts()对gender变量进行分析
pd.value_counts(teenager_sns["gender"],dropna=False)
```

```
F      22054
M       5222
NaN     2724
Name: gender, dtype: int64
```

统计毕业年份的信息，注意设置不要忽略 NA

```
# 打印变量age的缺失值数量
print('age变量缺失值数目：', len(teenager_sns["age"]) -  teenager_sns["age"].count())  # 补充
```

```
age变量缺失值数目：  5086
```

```
# 使用 describe() 函数则可以对 age 变量进行描述性统计。
teenager_sns["age"].describe()
```

```
count    24914.000000
mean        17.993949
std          7.858054
min          3.086000
25%         16.312000
50%         17.287000
75%         18.259000
max        106.927000
Name: age, dtype: float64
```

利用 describe 函数可以查看统计信息，包括最值，均值，标准差，数据类型等

```
import numpy as np

# your function is here
def tag_nan(value):
    if 13<= value <=20:
        return value
    else:
        return np.NaN
```

为了筛选无效成绩，只保留 13-20 的年龄段

```
teenager_sns["gender"] = teenager_sns["gender"].replace(np.NaN, "unkown")
teenager_sns["gender"].value_counts()
```

```
F          22054
M           5222
unkown      2724
Name: gender, dtype: int64
```

```
# 查看 teenager_sns 的前10行
teenager_sns.head(10)
```

	gradyear	gender	age	friends	basketball	football	soccer	softball	volleyball	swimming	...
0	2006	M	18.980	7	0	0	0	0	0	0	...
1	2006	F	18.801	0	0	1	0	0	0	0	...
2	2006	M	18.335	69	0	1	0	0	0	0	...
3	2006	F	18.875	0	0	0	0	0	0	0	...
4	2006	unkown	18.995	10	0	0	0	0	0	0	...
5	2006	F	NaN	142	0	0	0	0	0	0	...
6	2006	F	18.930	72	0	0	0	0	0	0	...
7	2006	M	18.322	17	0	0	0	1	0	0	...
8	2006	F	19.055	52	0	0	0	0	0	0	...
9	2006	F	18.708	39	0	0	0	0	0	0	...

```
gender_dummies = pd.get_dummies(teenager_sns["gender"], prefix="gender")
# 查看 gender_dummies 的前10行
gender_dummies.head(10)
```

再通过虚拟编码，把 gender 转换成三个二元变量，这样就可以计算欧式距离了

```
teenager_sns = pd.concat([teenager_sns, gender_dummies], axis = 1)
# 查看 teenager_sns 的前15行
teenager_sns.head(15)
```

	gradyear	gender	age	friends	basketball	football	soccer	softball	volleyball	swimming	...
0	2006	M	18.980	7	0	0	0	0	0	0	...
1	2006	F	18.801	0	0	1	0	0	0	0	...
2	2006	M	18.335	69	0	1	0	0	0	0	...
3	2006	F	18.875	0	0	0	0	0	0	0	...
4	2006	unkown	18.995	10	0	0	0	0	0	0	

将处理后的数据进行拼接

```
age_mean = teenager_sns["age"].mean()
age_mean
```

```
17.25242885157497
```

查看均值

```
teenager_sns["age_avg_imputated"] = teenager_sns["age"].fillna(value = age_mean)
# 查看 teenager_sns 的前15行
teenager_sns.head(15)
```

ister	abercrombie	die	death	drunk	drugs	gender_F	gender_M	gender_unkown	age_avg_imputated
0	0	0	0	0	0	0	1	0	18.980000
0	0	0	0	0	0	1	0	0	18.801000
0	0	0	1	0	0	0	1	0	18.335000
0	0	0	0	0	0	1	0	0	18.875000
0	0	0	0	1	1	0	0	1	18.995000
0	0	0	0	1	0	1	0	0	17.252429

再把之前的无效年龄用平均年龄代替，例如最右下角的 17.252

```
# 查看 teenager_sns 所有列名列表
teenager_sns.columns
```

```
Index(['gradyear', 'gender', 'age', 'friends', 'basketball', 'football',
       'soccer', 'softball', 'volleyball', 'swimming', 'cheerleading',
       'baseball', 'tennis', 'sports', 'cute', 'sex', 'sexy', 'hot', 'kissed',
       'dance', 'band', 'marching', 'music', 'rock', 'god', 'church', 'jesus',
       'bible', 'hair', 'dress', 'blonde', 'mall', 'shopping', 'clothes',
       'hollister', 'abercrombie', 'die', 'death', 'drunk', 'drugs',
       'gender_F', 'gender_M', 'gender_unkown', 'age_avg_imputated'],
      dtype='object')
```

查看当前的列名

```
# 去除了gender,age两个属性
from sklearn import preprocessing
filtered_columns = ["gradyear","friends","basketball","football",
                    "soccer","softball","volleyball","swimming","cheerleading",
                    "baseball","tennis","sports","cute","sex","sexy","hot","kissed",
                    "dance","band","marching","music","rock","god","church","jesus",
                    "bible","hair","dress","blonde","mall","shopping","clothes",
                     "hollister","abercrombie","die","death","drunk","drugs",
                     "gender_M","gender_F","gender_unkown","age_avg_imputated"]

teenager_sns_zscore = pd.DataFrame(preprocessing.scale(teenager_sns[filtered_columns]),\
                                   columns = teenager_sns[filtered_columns].columns)
teenager_sns_zscore.head(5)
```

```
/Users/cyf/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:11: DataConversio
nWarning: Data with input dtype uint8, int64, float64 were all converted to float64 by t
he scale function.
  # This is added back by InteractiveShellApp.init_path()
```

e	die	death	drunk	drugs	gender_M	gender_F	gender_unkown	age_avg_imputated
2	-0.294793	-0.261530	-0.220403	-0.174908	2.178285	-1.665979	-0.316019	1.652413
2	-0.294793	-0.261530	-0.220403	-0.174908	-0.459077	0.600248	-0.316019	1.481200
2	-0.294793	2.027908	-0.220403	-0.174908	2.178285	-1.665979	-0.316019	1.035474
2	-0.294793	-0.261530	-0.220403	-0.174908	-0.459077	0.600248	-0.316019	1.551981
2	-0.294793	-0.261530	2.285122	2.719316	-0.459077	-1.665979	3.164367	1.666760

进行数据标准化，方便计算

```
# 打印 teenager_sns_zscore 中.age_avg_imputated 的均值
teenager_sns_zscore["age_avg_imputated"]
print(teenager_sns_zscore["age_avg_imputated"].mean())
```

```
-3.4896382080281305e-15
```

```
# 打印 teenager_sns_zscore 中.age_avg_imputated 的标准方差
print(teenager_sns_zscore["age_avg_imputated"].std())
```

```
1.0000166670833448
```

查看标准化的结果，这里和同学的同一份代码均值结果不同，可能是 mac 下 python3.7 版本中忽略了某些过小值导致，但方差是一致的。

```
from sklearn.cluster import KMeans # import class
teenager_cluster_model = KMeans(n_clusters = 5)  # initialization
teenager_cluster_model.fit(teenager_sns_zscore) # fit the data
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=5, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

最后进行训练，设置分为 5 类（k=5）

五、实验结果分析及回答问题（或测试环境及测试结果）

```
teenager_clusters = pd.Series(teenager_cluster_model.labels_)
teenager_clusters.value_counts().sort_index()
```

```
0     1740
1     2660
2     5097
3     9880
4    10623
dtype: int64
```

我们成功将青少年分为 5 类

```
]:
```

	gradyear	friends	basketball	football	soccer	softball	volleyball	swimming	cheerleadin
0	0.172203	0.140046	0.554819	0.532061	0.239598	0.239701	0.269379	0.413701	0.37149
1	-0.057163	-0.110006	-0.144410	-0.167954	-0.053513	-0.100017	-0.070928	-0.068722	-0.05090
2	-0.114326	-0.150637	0.048373	0.274545	-0.017635	-0.206788	-0.171550	-0.108171	-0.19206
3	-0.881844	-0.045620	-0.126944	-0.122076	-0.076024	-0.032519	-0.070883	-0.043242	-0.02844
4	0.861127	0.119312	0.040139	-0.063284	0.053323	0.115245	0.121873	0.041565	0.07050

其中每一个类的中心是如下特征，代表青少年的不同偏好种类

```
: # 改变K的值，观察聚类结果

print("k=2")
from sklearn.cluster import KMeans # import class
teenager_cluster_model = KMeans(n_clusters = 2)  # initialization
teenager_cluster_model.fit(teenager_sns_zscore) # fit the data

teenager_clusters = pd.Series(teenager_cluster_model.labels_)
teenager_clusters.value_counts().sort_index()

centers = pd.DataFrame(teenager_cluster_model.cluster_centers_, \
                       columns = teenager_sns_zscore.columns)
print("centroid: ",centers)

centers_t = centers.T
centers_t.columns = ["cluster_0","cluster_1"]

print("cluster_0: ")
print(centers_t["cluster_0"].sort_values(ascending = False, inplace = False).head(10))

print("cluster_1: ")
print(centers_t["cluster_1"].sort_values(ascending = False, inplace = False).head(10))
```

```
k=2
centroid:     gradyear   friends  basketball  football    soccer  softball  volleyball
\
0 -0.092587 -0.135256   -0.004213   0.130544 -0.027635 -0.169371   -0.135724
1   0.033137  0.048408    0.001508 -0.046721  0.009890  0.060617    0.048575

   swimming  cheerleading  baseball        ...          hollister  \
0 -0.094844     -0.145361   0.17994        ...          -0.084041
1   0.033944      0.052024  -0.06440        ...           0.030078

   abercrombie       die     death     drunk     drugs  gender_M  gender_F  \
0    -0.090741 -0.024034 -0.051899 -0.045805 -0.042038  1.280708 -1.665979
1     0.032476  0.008602  0.018574  0.016393  0.015045 -0.458361  0.596247
```

也可以将 k 设置为 2，这样就变成了二分类问题，可以从 gender 看出，这次性别拥有相当大的区别。

```
print(centers_t["cluster_1"].sort_values(ascending = False, inplace = False).head(10))
```

```
k=3
centroid:     gradyear   friends  basketball  football    soccer  softball  volleybal
\
0 -0.002969  0.013485   -0.077101 -0.123416 -0.030194  0.025107    0.003274
1   0.273386  0.271146    0.541757  0.481800  0.274442  0.272837    0.334462
2 -0.102573 -0.143179   -0.023613  0.117358 -0.034322 -0.173177   -0.142909

   swimming  cheerleading  baseball        ...          hollister  \
0 -0.017280     -0.006141 -0.109171        ...          -0.096178
1   0.360491      0.421395  0.257967        ...           0.882849
2 -0.101541     -0.154160  0.171535        ...          -0.112810

   abercrombie       die     death     drunk     drugs  gender_M  gender_F  \
0    -0.098055 -0.084087 -0.051586 -0.079532 -0.109307 -0.459077  0.600248
1     0.903953  0.699689  0.536834  0.703800  0.926534 -0.327080  0.377087
2    -0.116571 -0.069573 -0.085996 -0.082718 -0.097279  1.289732 -1.665979

   gender_unkown  age_avg_imputated
0      -0.316019          -0.004453
1      -0.147486          -0.262503
2       0.856556           0.116913
```

同理设置 k=3

最后我们并没有进行定量分析，而是进行定性分析，查看各个种类间区别，这是由于我们的目的——定向营销决定的，我们更关心青少的的种类

总结，k 聚类有如下优缺点：

优点：

1、解决聚类问题的经典算法，简单、快速

2、当处理大数据集时，该算法保持可伸缩性和高效率

3、当簇近似为高斯分布时，它的效果较好

缺点：

1、在簇的平均值可被定义的情况下才能使用，可能不适用于某些应用

2、必须实现给出 k（要生成簇的数目），而且对初值敏感，即对于不同的初值，可能会导致不同结果

3、不适合非凸形状的簇或者大小差别很大的簇

4、对噪声和孤立点敏感

期间遇到了同一份代码在两种平台运行结果不同的问题，初步判定是由于数据过小，mac 平台忽略的问题。

通过这次实验，了解到了简单聚类算法与评价方法，在很多日常生活中都可以试用。

同时这应该也是本次课程最后一次报告了，个人感觉安全前沿讨论班这门课程内容翔实，收获较大。了解到了 jupyter 这个强大的工具，也锻炼了我的 python 编程能力，这在今后的各个领域都能用到，如果能早一点学就更好了。

同时感谢老师的辛苦备课，批改作业。祝王利娟老师日后工作顺利，身体健康！

曹寅峰