kmeans算法实战

简单聚类

数据集为20个酒啤酒，酒的属性包括：calories（卡路里）、sodium（钠）、alcohol（酒精）、cost

我们不需要标签y，所以我们只需要设置X就可以：

```
X = beer[["calories","sodium","alcohol","cost"]]
```

我们创建两个kmeans模型，**聚类簇数**分别设置为3和2。

```
from sklearn.cluster import KMeans

km = KMeans(n_clusters=3).fit(X)
km2 = KMeans(n_clusters=2).fit(X)
```

.labels_属性可以表示所有数据所属的簇：

array([0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 2, 0, 0, 2, 1])

第一个数据属于第0个类别...

我们可以聚类后的数据是不是符合实际情况：

	name	calories	sodium	alcohol	cost	cluster	cluster2
0	Budweiser	144	15	4.7	0.43	0	1
1	Schlitz	151	19	4.9	0.43	0	1
2	Lowenbrau	157	15	0.9	0.48	0	1
3	Kronenbourg	170	7	5.2	0.73	0	1
4	Heineken	152	11	5.0	0.77	0	1
5	Old_Milwaukee	145	23	4.6	0.28	0	1
6	Augsberger	175	24	5.5	0.40	0	1
7	Srohs_Bohemian_Style	149	27	4.7	0.42	0	1
17	Heilemans_Old_Style	144	24	4.9	0.43	0	1
16	Hamms	139	19	4.4	0.43	0	1
10	Coors	140	18	4.6	0.44	0	1
14	Kirin	149	6	5.0	0.79	0	1
12	Michelob_Light	135	11	4.2	0.50	0	1
13	Becks	150	19	4.7	0.76	0	1
9	Budweiser_Light	113	8	3.7	0.40	1	0
8	Miller_Lite	99	10	4.3	0.43	1	0
11	Coors_Light	102	15	4.1	0.46	1	0
19	Schlitz_Light	97	7	4.2	0.47	1	0
15	Pabst_Extra_Light	68	15	2.3	0.38	2	0
18	Olympia_Goled_Light	72	6	2.9	0.46	2	0

我们还可以观察对应聚类的数据差异：

```
beer.groupby("cluster").mean()
```

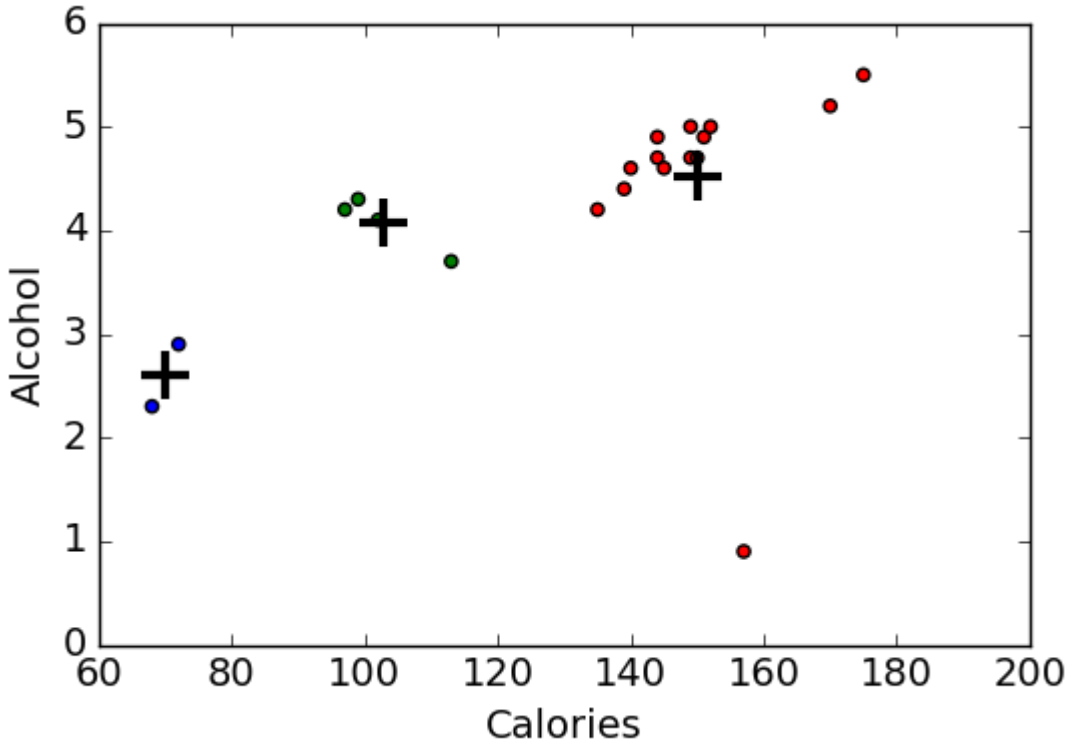
	calories	sodium	alcohol	cost	cluster2
cluster					
0	150.00	17.0	4.521429	0.520714	1
1	102.75	10.0	4.075000	0.440000	0
2	70.00	10.5	2.600000	0.420000	0

这样可以看到每个类对应的数据差异。

画图

接下来我们把聚类后的数据画出来，同时画出每个类的特征均值（下面只画两个特征Calories和Alcohol）：

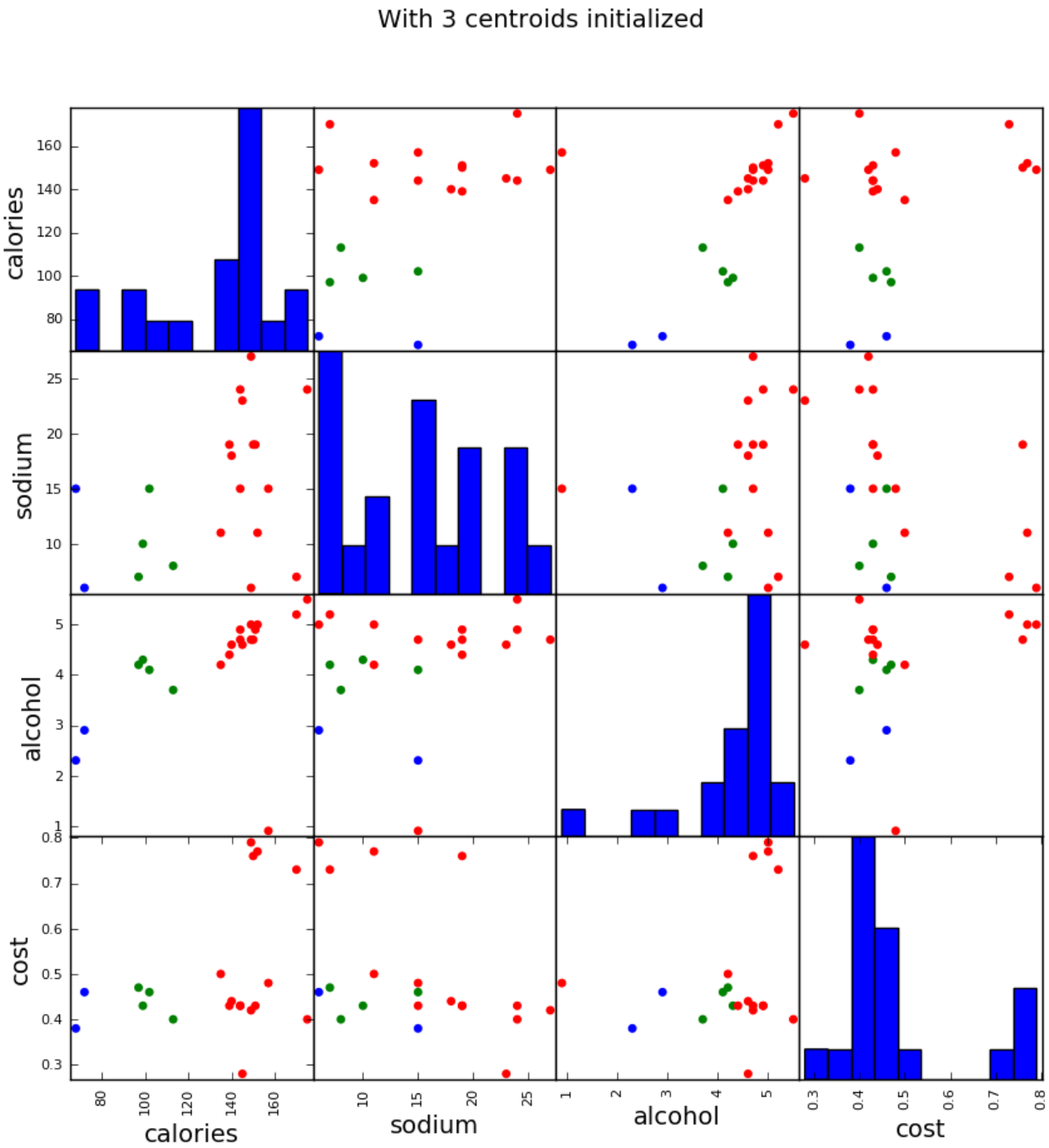
```
centers = beer.groupby("cluster").mean().reset_index()
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams['font.size'] = 14
import numpy as np
colors = np.array(['red', 'green', 'blue', 'yellow'])
plt.scatter(beer["calories"], beer["alcohol"],c=colors[beer["cluster"]])

plt.scatter(centers.calories, centers.alcohol, linewidths=3, marker='+', s=300, c='black')

plt.xlabel("Calories")
plt.ylabel("Alcohol")
```



从图中可以发现有三类数据，最左边只有两个样本，中间有四个样本，最右边的类样本最多。

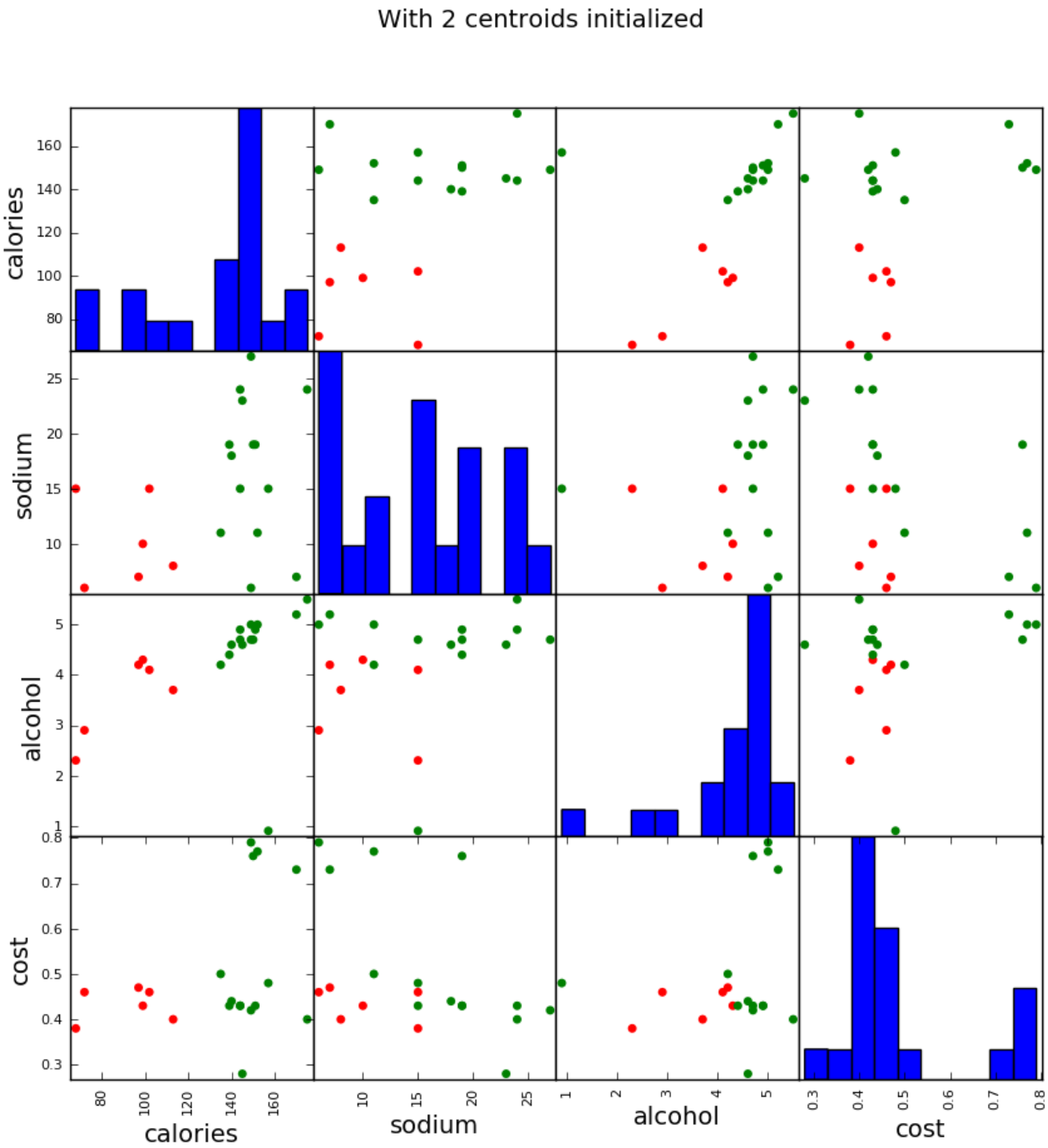
这里我们只看到两类特征，我们想看所有的两两特征，就可以画个简单的散点图：

```
scatter_matrix(beer[["calories","sodium","alcohol","cost"]],s=100, alpha=1,
c=colors[beer["cluster"]], figsize=(10,10))
plt.suptitle("With 3 centroids initialized")
```



X轴有四个属性，Y轴有四个属性，对角线是数据的分布。

接下来我们看两个簇的效果：



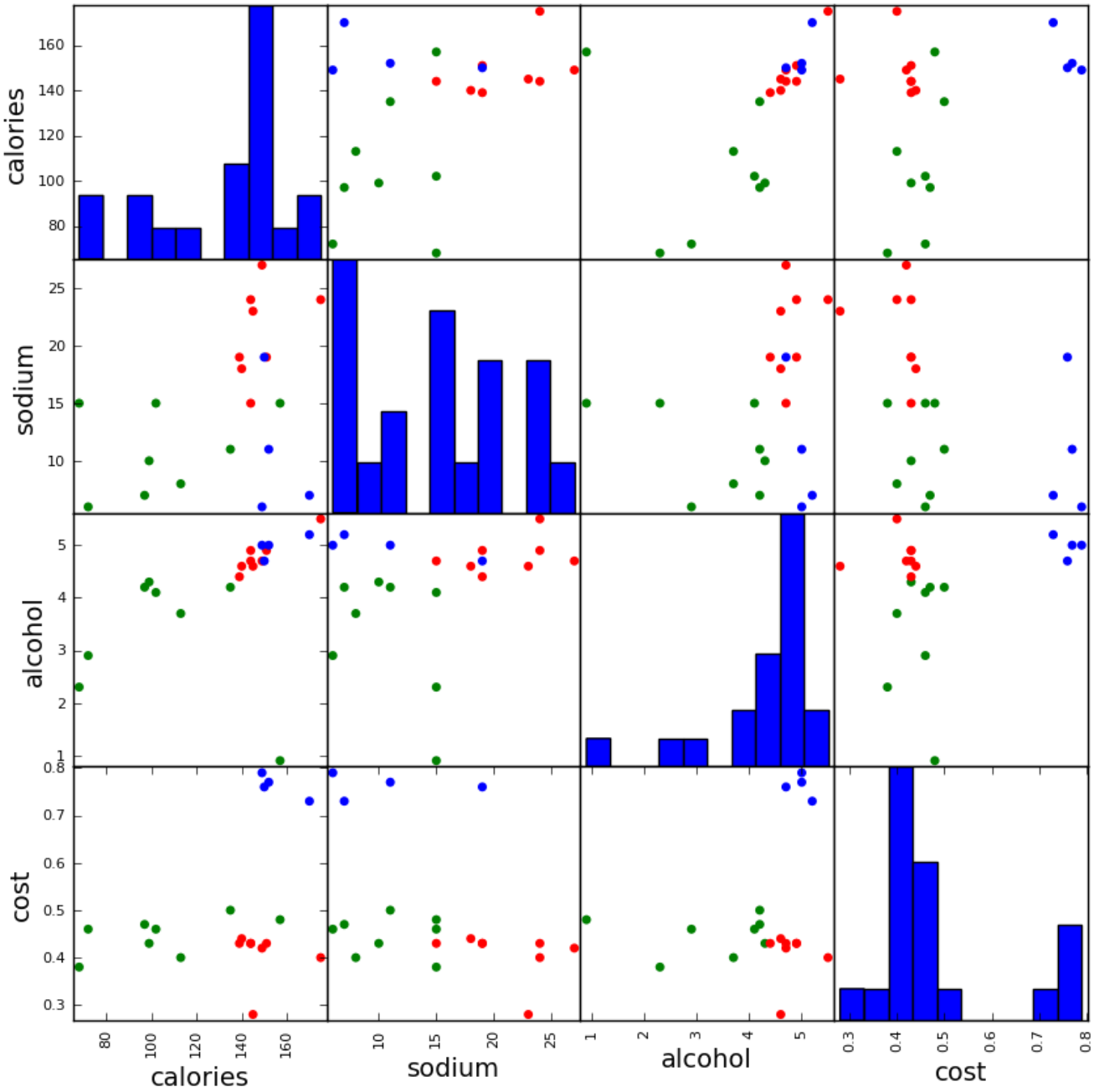
我们可以通过观察图来判断两个簇好还是三个簇好。

标准化/归一化

接下来我们对数据进行**标准化或者归一化**(消除数据之间的差异性，聚类之前基本上都要这么做)：

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

再对处理过的数据进行聚类，得到新的散点图：



在处理标准化之前，我们发现calories比较重要（值比较大），cost不太重要。处理之后，就是同样重要。

聚类评估(轮廓系数Silhouette Coefficient)

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

$$s(i)=\begin{cases}1-\dfrac{a(i)}{b(i)}, & a(i)<b(i)\\ 0, & a(i)=b(i)\\ \dfrac{b(i)}{a(i)}-1, & a(i)>b(i)\end{cases}$$

这个指标计算的是样本i到同簇其他样本的平均距离a_i ,a_i 越小，说明样本i越应该被聚类到该簇。将a_i 称为样本i的簇内不相似度。

计算样本i到其他某簇C_j的所有样本的平均距离b_{ij}，称为样本i与簇C_j 的不相似度。

- s_i接近1，则说明样本i聚类合理
- s_i接近-1，则说明样本i更应该分类到另外的簇
- 若s_i 近似为0，则说明样本i在两个簇的边界上。

我们对标准化前后的数据进行轮廓系数计算：

```
from sklearn import metrics
score_scaled = metrics.silhouette_score(X,beer.scaled_cluster)
score = metrics.silhouette_score(X,beer.cluster)
print(score_scaled, score)
```

计算结果为：0.179780680894 0.673177504646

我们发现，做标准化的结果比较低，不做标准化的结果比较高。这是因为特征的重要性我们是不知道的，我们将calories的重要度通过标准化降低之后可能会造成不好的影响。

我们要确定簇数量k，可以进行遍历：

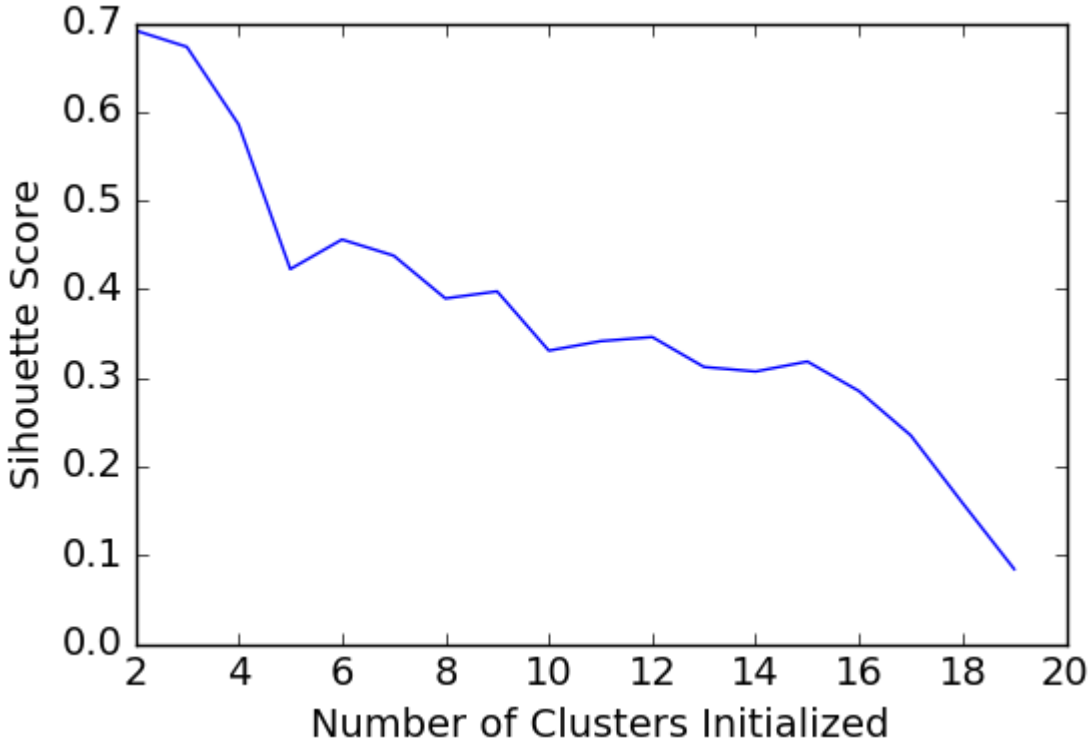
```
scores = []
for k in range(2,20):
    labels = KMeans(n_clusters=k).fit(X).labels_
    score = metrics.silhouette_score(X, labels)
    scores.append(score)

scores
```

结果为：

```
[0.69176560340794857,
 0.67317750464557957,
 0.58570407211277953,
 0.42254873351720201,
 0.4559182167013377,
 0.43776116697963124,
 0.38946337473125997,
 0.39746405172426014,
 0.33061511213823314,
 0.34131096180393328,
 0.34597752371272478,
 0.31221439248428434,
 0.30707782144770296,
 0.31834561839139497,
 0.28495140011748982,
 0.23498077333071996,
 0.15880910174962809,
 0.084230513801511767]
```

我们选择轮廓系数比较大的，所以选择了k=2。

肉眼看可能不太直观，我们可以画个图：



到这里呢，我们就有了kmeans的一个标准流程：我们先进行聚类，然后可视化展示，之后再评估，想一想什么参数比较合适，再重新聚类。