K均值聚类模型

讲师：刘顺祥

课程目标

1. 理解Kmeans聚类的思想和原理
2. 最佳K值的选择
3. 掌握Kmeans聚类的实操

Kmeans聚类

模型介绍

对于有监督的数据挖掘算法而言，数据集中需要包含标签变量（即因变量y的值）。但在有些场景下，并没有给定的y值，对于这类数据的建模，一般称为无监督的数据挖掘算法，最为典型的当属聚类算法。

Kmeans聚类算法利用距离远近的思想将目标数据聚为指定的k个簇，进而使样本呈现簇内差异小，簇间差异大的特征。

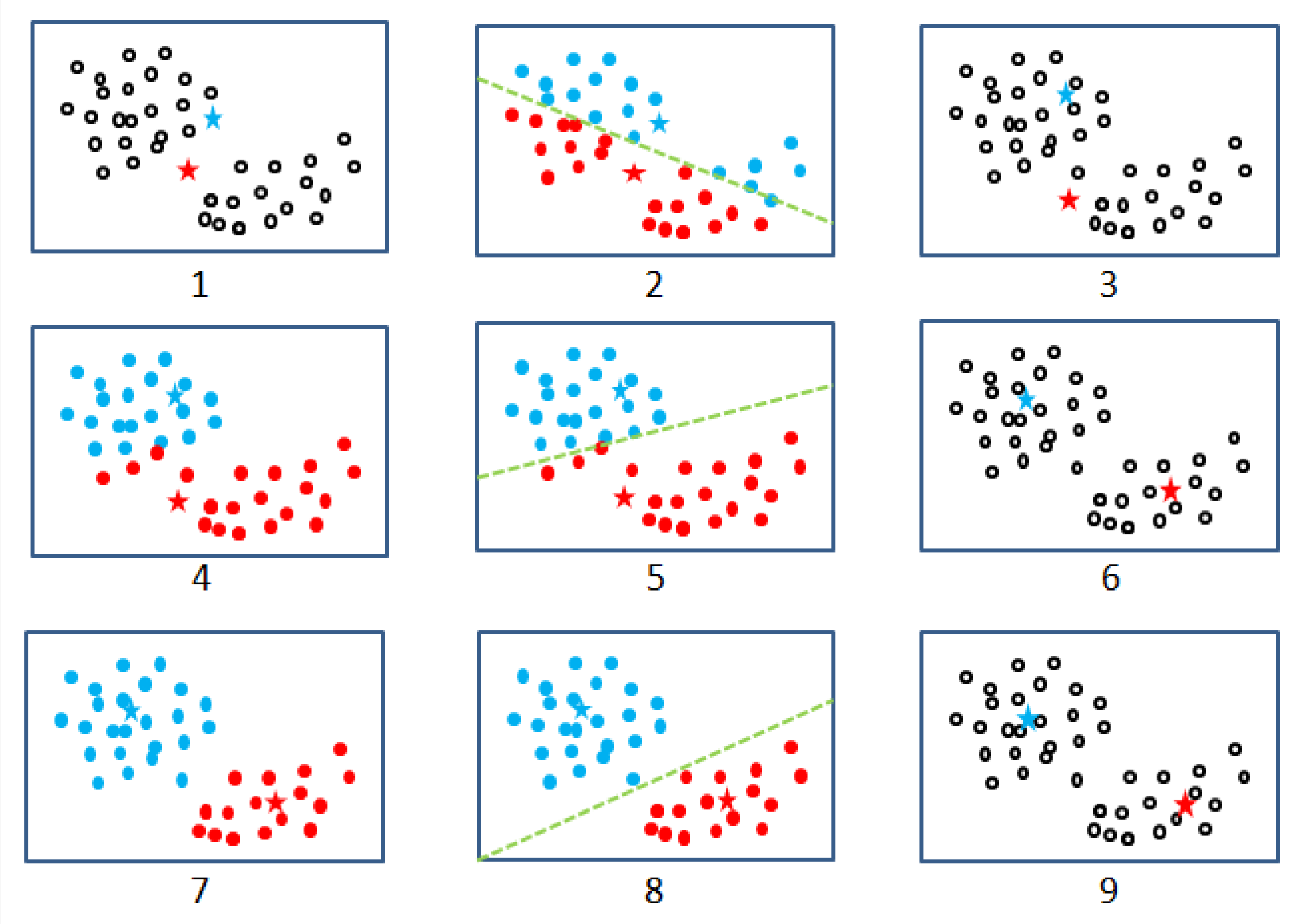
Kmeans聚类

聚类步骤

- 从数据中随机挑选k个样本点作为原始的簇中心
- 计算剩余样本与簇中心的距离，并把各样本标记为离k个簇中心最近的类别
- 重新计算各簇中样本点的均值，并以均值作为新的k个簇中心
- 不断重复第二步和第三步，直到簇中心的变化趋于稳定，形成最终的k个簇

Kmeans聚类

聚类步骤



原理介绍

在Kmeans聚类模型中，对于指定的k个簇，只有簇内样本越相似，聚类效果才越好。基于这个思想，可以理解为簇内样本的离差平方和之和达到最小即可。进而可以衍生出Kmeans聚类的目标函数：

$$J(c_1, c_2, \dots c_k) = \sum_{j=1}^{k}\sum_{i}^{n_j}(x_i - c_j)^2$$

其中，c_j表示第j个簇的簇中心，x_i属于第j个簇的样本i，n_j表示第j个簇的样本总量。对于该目标函数而言，c_j是未知的参数，要想求得目标函数的最小值，得先知道参数c_j的值。

原理介绍

- 对目标函数求偏导

$$\frac{\partial J}{\partial c_j} = \sum_{j=1}^{k}\sum_{i=1}^{n_j}\frac{\left(x_i - c_j\right)^2}{\partial c_j} = \sum_{i=1}^{n_j}\frac{\left(x_i - c_j\right)^2}{\partial c_j} = \sum_{i=1}^{n_j} -2\left(x_i - c_j\right)$$

- 令导函数为0

$$\sum_{i=1}^{n_j} -2\left(x_i - c_j\right) = 0$$

$$n_j c_j - \sum_{i=1}^{n_j} x_i = 0$$

$$\therefore c_j = \frac{\sum_{i=1}^{n_j} x_i}{n_j} = \mu_j$$

拐点法

簇内离差平方和拐点法的思想很简单，就是在不同的k值下计算簇内离差平方和，然后通过可视化的方法找到“拐点”所对应的k值。当折线图中的斜率由大突然变小时，并且之后的斜率变化缓慢，则认为突然变化的点就是寻找的目标点，因为继续随着簇数k的增加，聚类效果不再有大的变化。

K值的选择

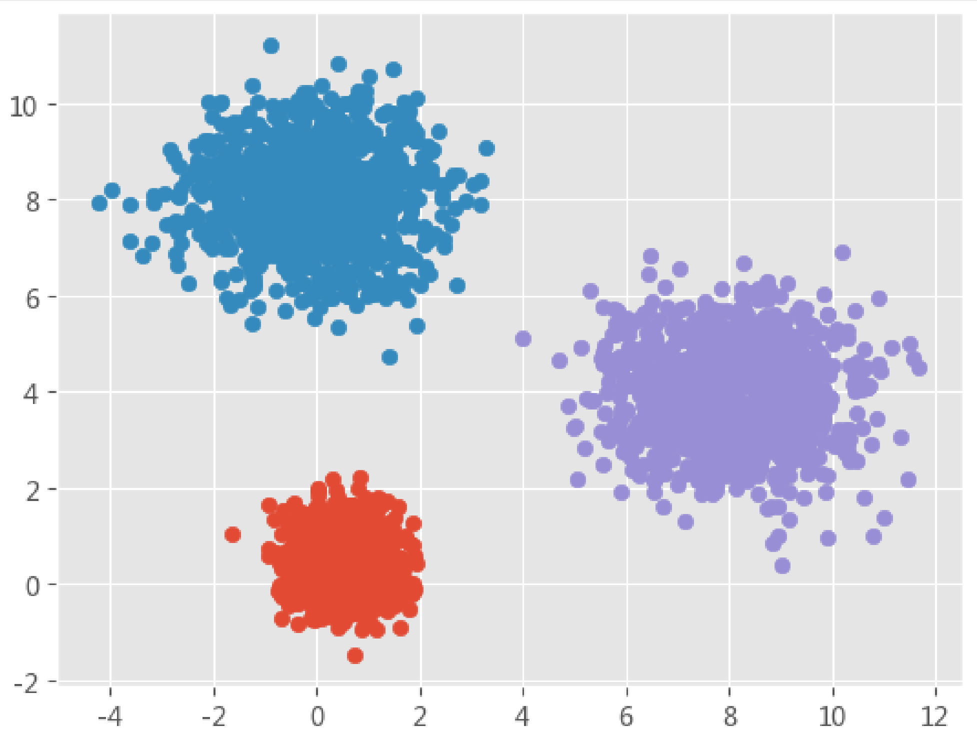
CSDN学院 IT实战派

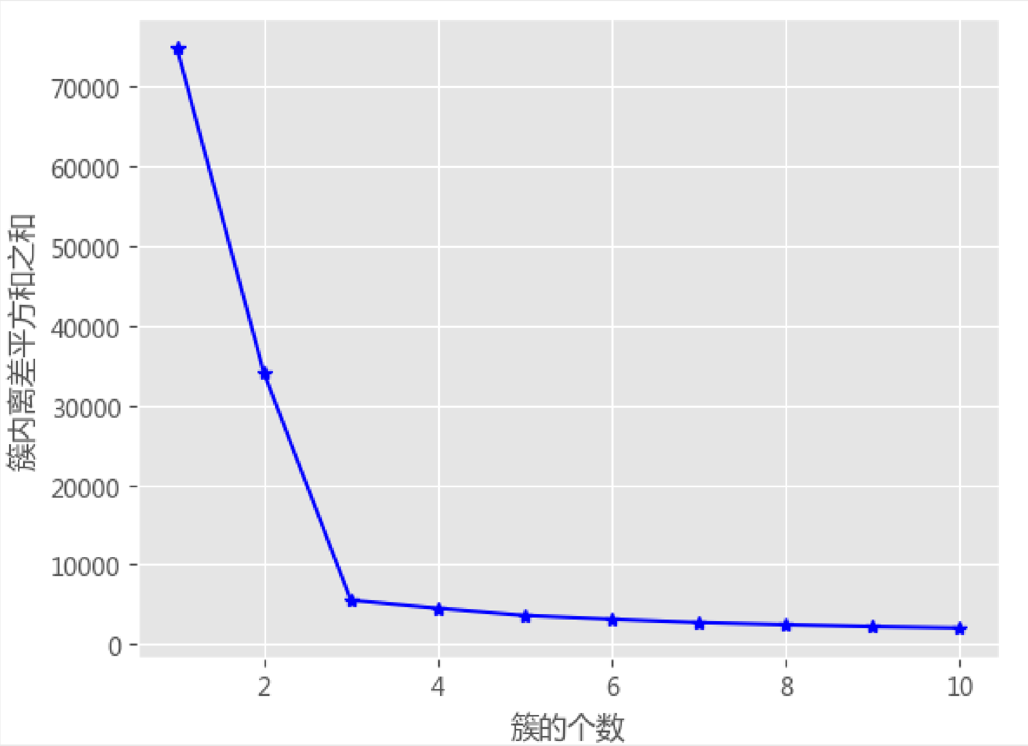
拐点法

```
def k_SSE(X, clusters):
    # 选择连续的K种不同的值
    K = range(1,clusters+1)
    # 构建空列表用于存储总的簇内离差平方和
    TSSE = []
    for k in K:
        # 用于存储各个簇内离差平方和
        SSE = []
        kmeans = KMeans(n_clusters=k)
        kmeans.fit(X)
        # 返回簇标签
        labels = kmeans.labels_
        # 返回簇中心
        centers = kmeans.cluster_centers_
        # 计算各簇样本的离差平方和，并保存到列表中
        for label in set(labels):
            SSE.append(np.sum((X.loc[labels == label,]-centers[label,:])**2))
        # 计算总的簇内离差平方和
        TSSE.append(np.sum(SSE))
```

K值的选择

拐点法


10
8
6
4
2
0
-2
-4
-2
0
2
4
6
8
10
12


70000
60000
50000
40000
30000
20000
10000
0
簇内离差平方和之和
2
4
6
8
10
簇的个数

轮廓系数法

该方法综合考虑了簇的密集性与分散性两个信息，如果数据集被分割为理想的k个簇，那么对应的簇内样本会很密集，而簇间样本会很分散。轮廓系数的计算公式可以表示为：

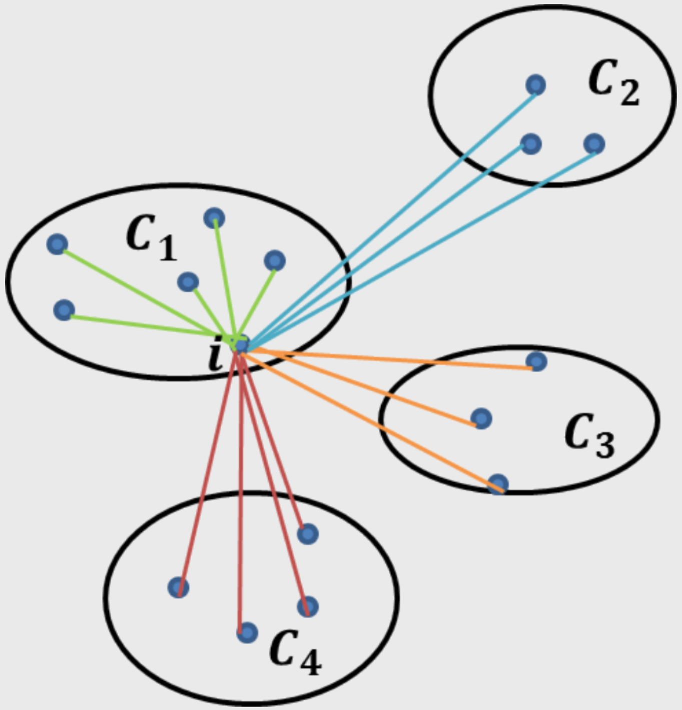
$$S(i)=\frac{b(i)-a(i)}{max\left(a(i),b(i)\right)}$$

其中，$a(i)$体现了簇内的密集性，代表样本i与同簇内其他样本点距离的平均值；$b(i)$反映了簇间的分散性，它的计算过程是，样本i与其他非同簇样本点距离的平均值，然后从平均值中挑选出最小值。

当$S(i)$接近于-1时，说明样本i分配的不合理，需要将其分配到其他簇中；当$S(i)$近似为0时，说明样本i落在了模糊地带，即簇的边界处；当$S(i)$近似为1时，说明样本i的分配是合理的。

K值的选择

轮廓系数法



假设数据集被拆分为4个簇，样本i对应的$a(i)$值就是所有C_1中其他样本点与样本i的距离平均值；样本i对应的$b(i)$值分两步计算，首先计算该点分别到C_2、C_3和C_4中样本点的平均距离，然后将三个平均值中的最小值作为$b(i)$的度量。

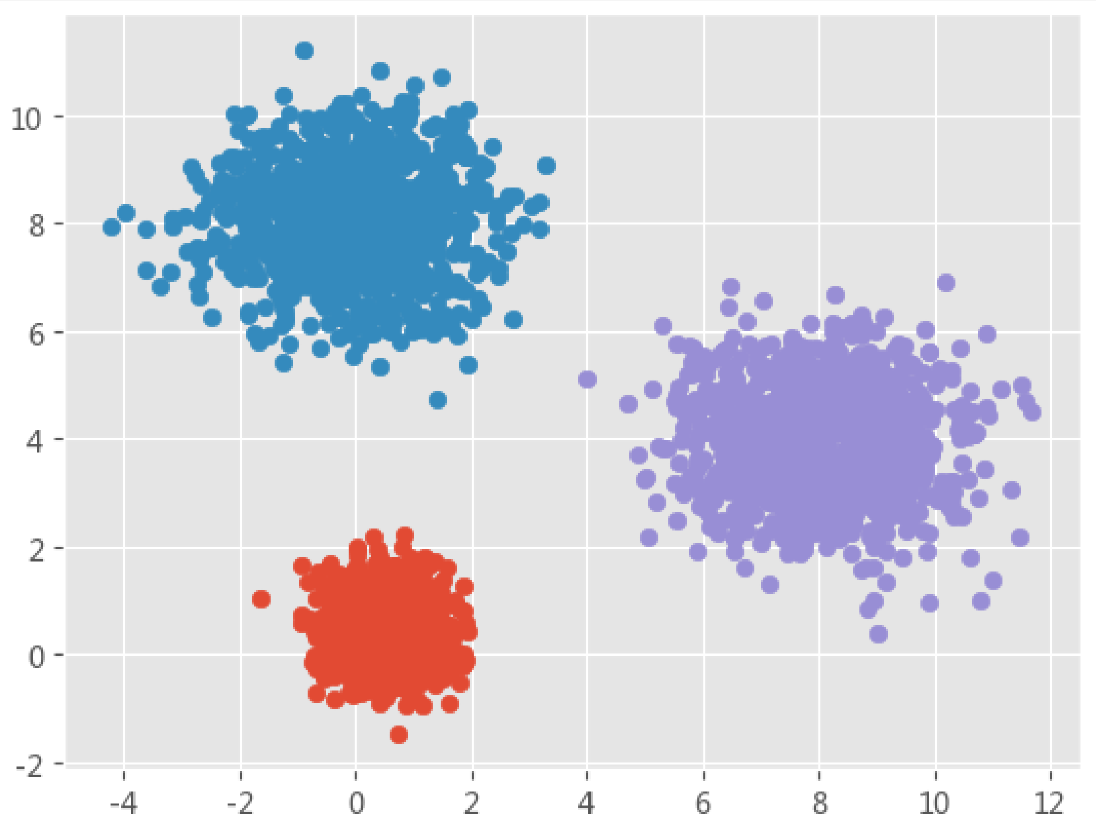
K值的选择

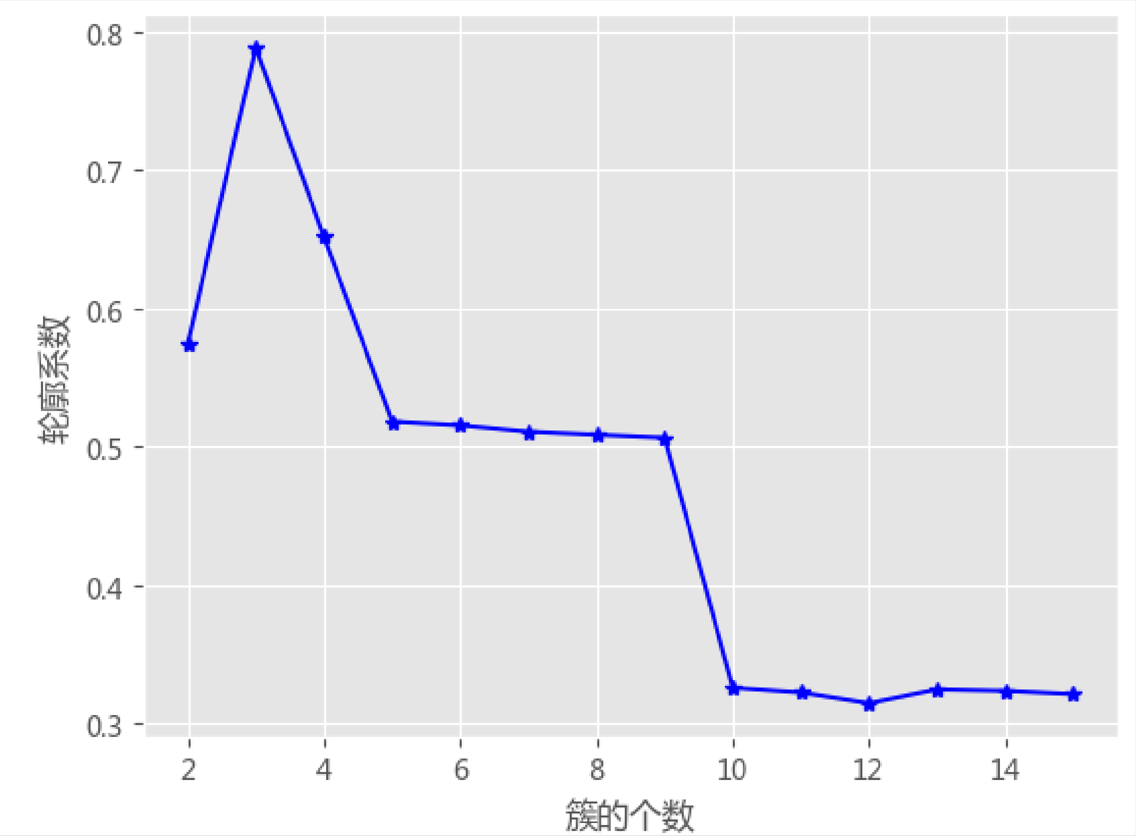
轮廓系数法

```
# 构造自定义函数
def k_silhouette(X, clusters):
    K = range(2,clusters+1)
    # 构建空列表，用于存储不同簇数下的轮廓系数
    S = []
    for k in K:
        kmeans = KMeans(n_clusters=k)
        kmeans.fit(X)
        labels = kmeans.labels_
        # 调用子模块metrics中的silhouette_score函数，计算轮廓系数
        S.append(metrics.silhouette_score(X, labels, metric='euclidean'))
```

K值的选择

轮廓系数法





函数介绍

KMeans(n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001)

n_clusters：用于指定聚类的簇数
init：用于指定初始的簇中心设置方法，如果为'k-means++'，则表示设置的初始簇中心之间相距较远；如果为'random'，则表示从数据集中随机挑选k个样本作为初始簇中心；如果为数组，则表示用户指定具体的簇中心
n_init：用于指定Kmeans算法运行的次数，每次运行时都会选择不同的初始簇中心，目的是防止算法收敛于局部最优，默认为10
max_iter：用于指定单次运行的迭代次数，默认为300
tol：用于指定算法收敛的阈值，默认为0.0001

模型实战

iris聚类--已知k值的情况

```
# 读取iris数据集
iris = pd.read_csv(r'C:\Users\Administrator\Desktop\iris.csv'  )
# 提取出用于建模的数据集X
X = iris.drop(labels = 'Species', axis = 1)
# 构建Kmeans模型
kmeans = KMeans(n_clusters = 3)
kmeans.fit(X)
# 聚类结果标签
X['cluster'] = kmeans.labels_
# 三个簇的簇中心
centers = kmeans.cluster_centers_
# 三个簇的簇中心
centers = kmeans.cluster_centers_
```

模型实战

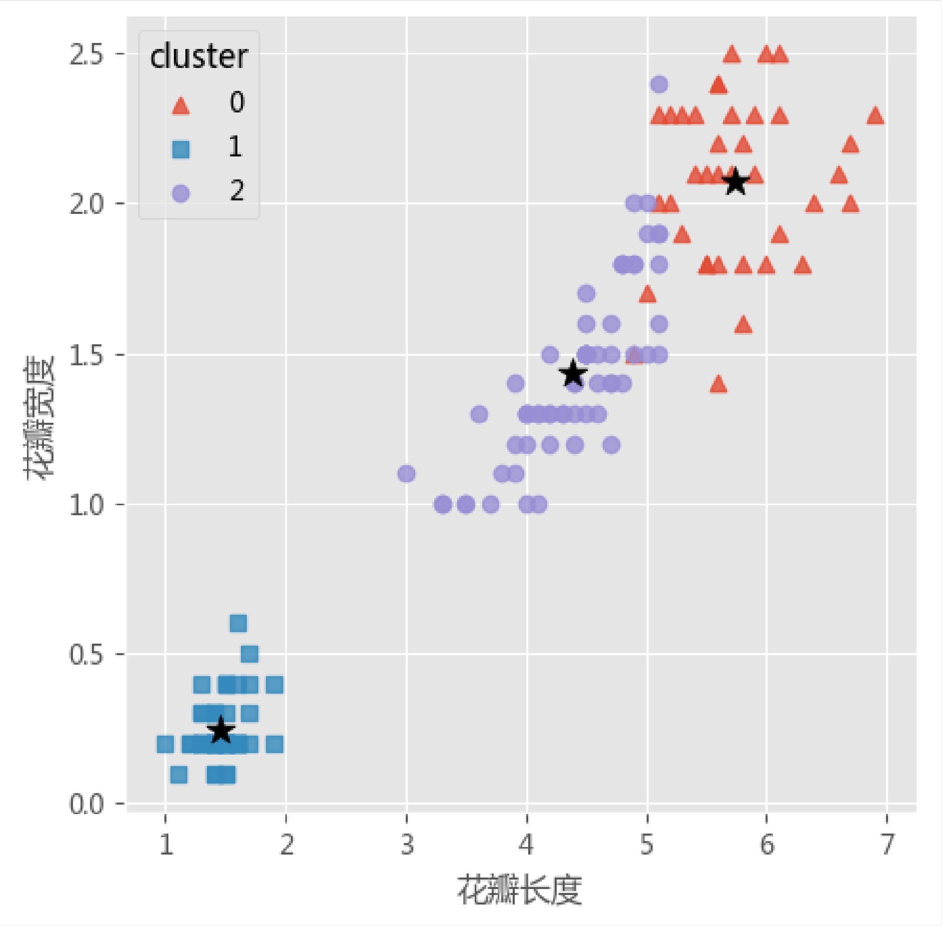
iris聚类--已知k值的情况

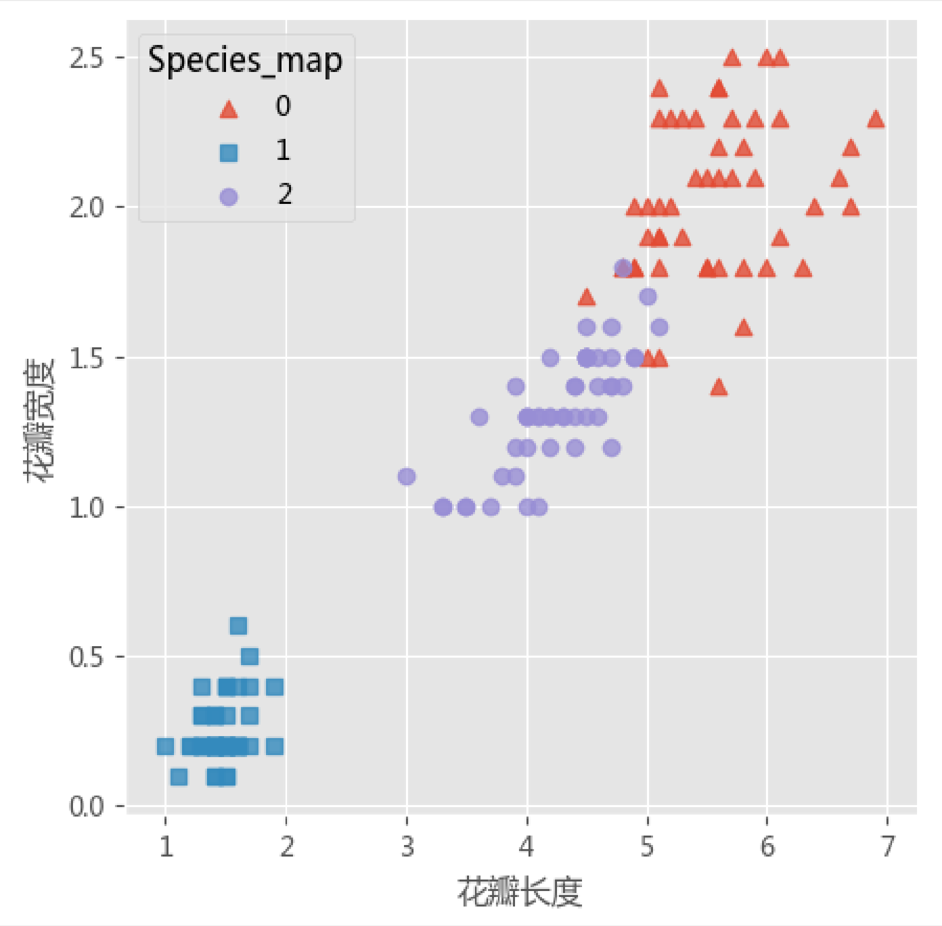
```
# 绘制聚类效果的散点图
sns.lmplot(x = 'Petal_Length', y = 'Petal_Width', hue = 'cluster', markers = ['^','s','o'],
           data = X, fit_reg = False, scatter_kws = {'alpha':0.8}, legend_out = False)
plt.scatter(centers[:,2], centers[:,3], marker = '*', color = 'black', s = 130)
plt.xlabel('花瓣长度')
plt.ylabel('花瓣宽度')
# 图形显示
plt.show()


# 增加一个辅助列，将不同的花种映射到0,1,2三种值，目的是方便后面图形的对比
iris['Species_map'] = iris.Species.map({'virginica':0,'setosa':1,'versicolor':2})
# 绘制原始数据三个类别的散点图
sns.lmplot(x = 'Petal_Length', y = 'Petal_Width', hue = 'Species_map', data = iris,
           markers = ['^','s','o'],fit_reg = False, scatter_kws = {'alpha':0.8},
           legend_out = False)
plt.xlabel('花瓣长度')
plt.ylabel('花瓣宽度')
# 图形显示
plt.show()
```

模型实战

iris聚类--已知k值的情况

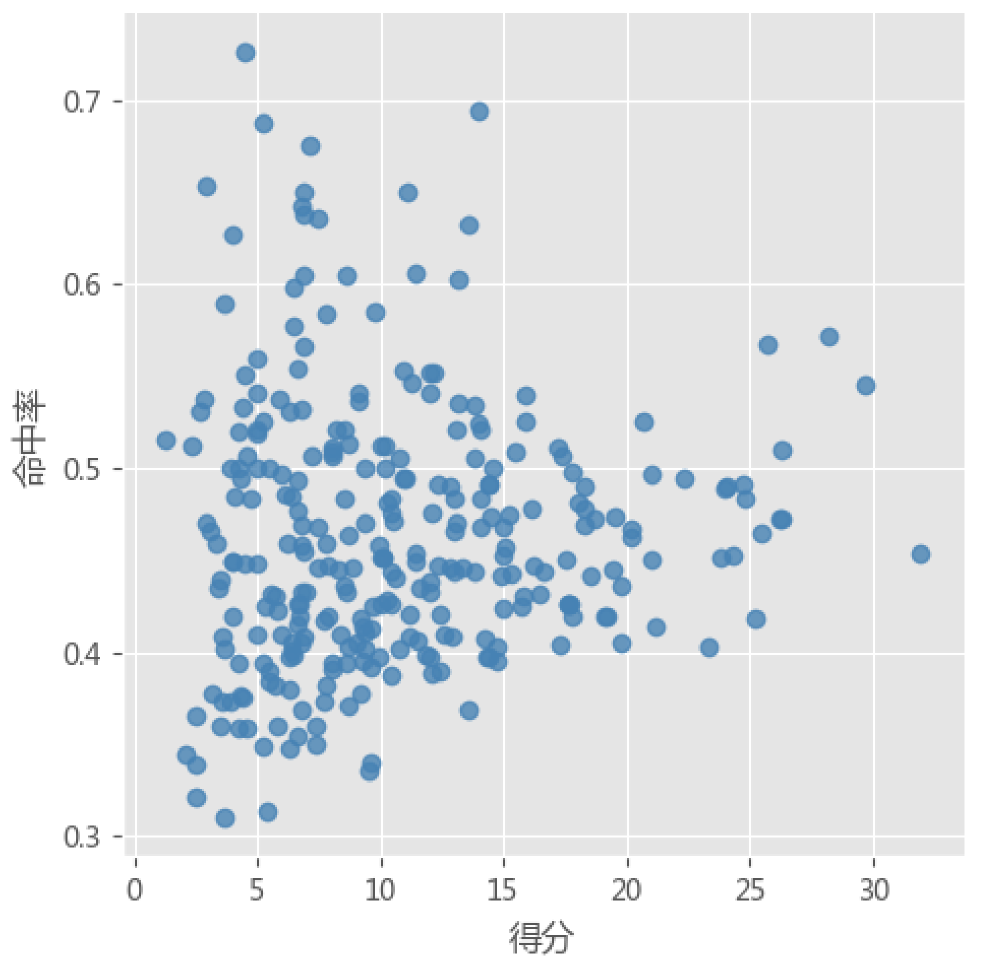




模型实战

EDU
CSDN学院 IT实战派

NBA球员聚类--未知k值的情况

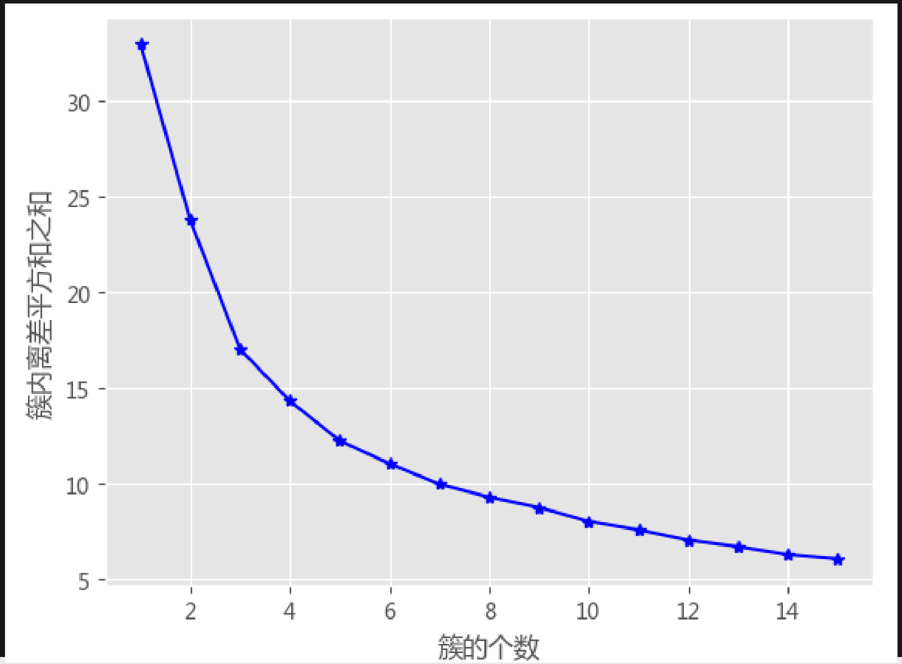


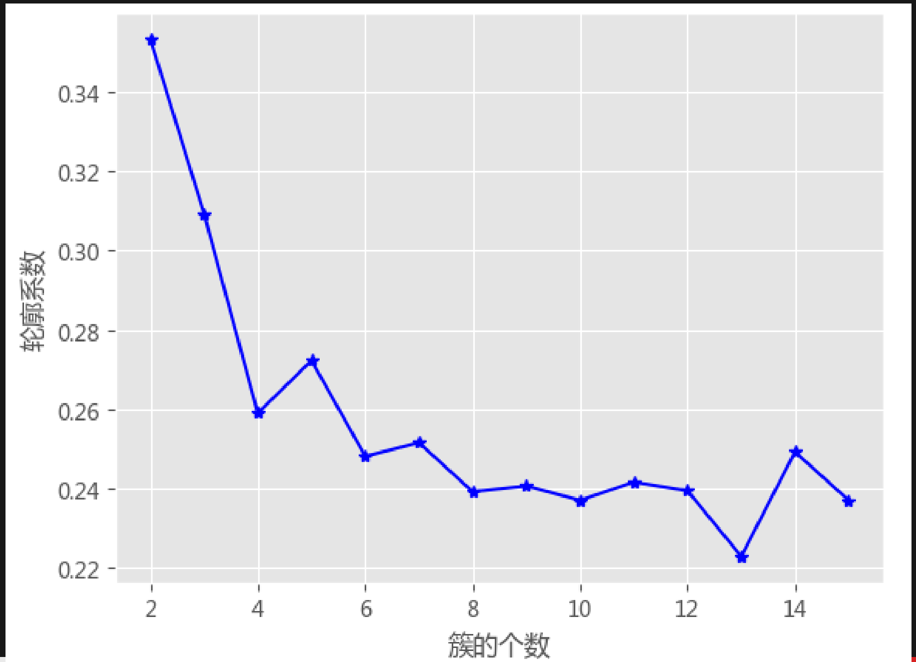
模型实战

NBA球员聚类--未知k值的情况

```
# 数据标准化处理
X = preprocessing.minmax_scale(players[['得分','罚球命中率','命中率','三分命中率']])
# 将数组转换为数据框
X = pd.DataFrame(X, columns=['得分','罚球命中率','命中率','三分命中率'])
# 使用拐点法选择最佳的K值
k_SSE(X, 15)

# 调用自定义函数，使用轮廓系数选择最佳的K值
k_silhouette(X, 15)
```





模型实战

NBA球员聚类--未知k值的情况

```
# 将球员数据集聚为3类
kmeans = KMeans(n_clusters = 3)
kmeans.fit(X)
# 将聚类结果标签插入到数据集players中
players['cluster'] = kmeans.labels_
# 构建空列表，用于存储三个簇的簇中心
centers = []
for i in players.cluster.unique():
    centers.append(players.ix[players.cluster == i,['得分','罚球命中率','命中率','三分命中率']].mean())
# 将列表转换为数组，便于后面的索引取数
centers = np.array(centers)

# 绘制散点图
sns.lmplot(x = '得分', y = '命中率', hue = 'cluster', data = players, markers = ['^','s','o
           fit_reg = False, scatter_kws = {'alpha':0.8}, legend = False)
# 添加簇中心
plt.scatter(centers[:,0], centers[:,2], c='k', marker = '*', s = 180)
plt.xlabel('得分')
plt.ylabel('命中率')
# 图形显示
plt.show()
```