

第13章 聚类

机器学习类型

有监督

数据点有已知的结果

无监督

数据点没有已知的结果



目标是通过在无标记训练样本的学习来揭示数据的内在性质及规律，为进一步的数据分析提供基础。

无监督学习的类型

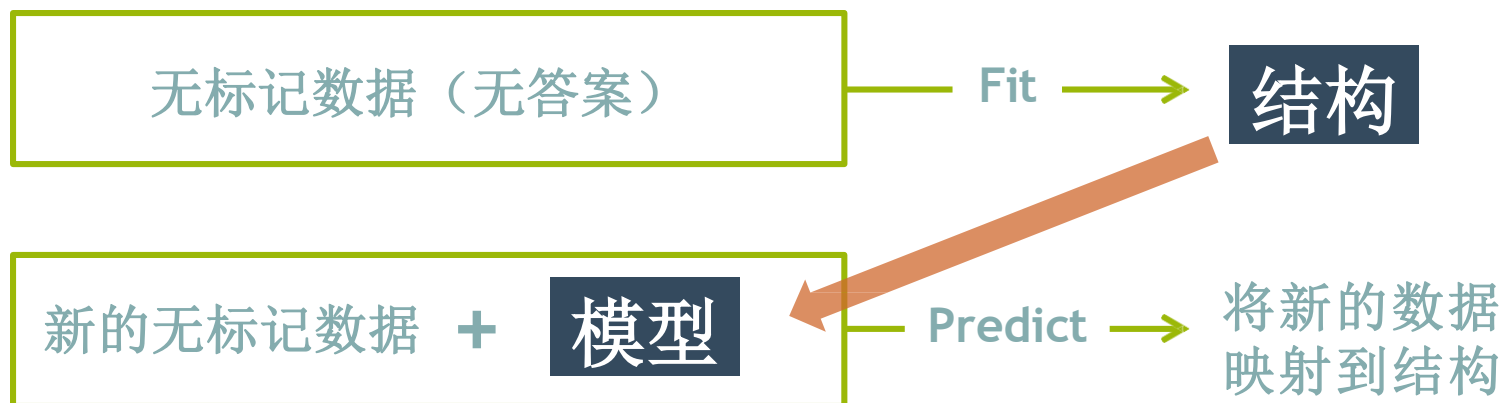
聚类

识别数据中未知的结构

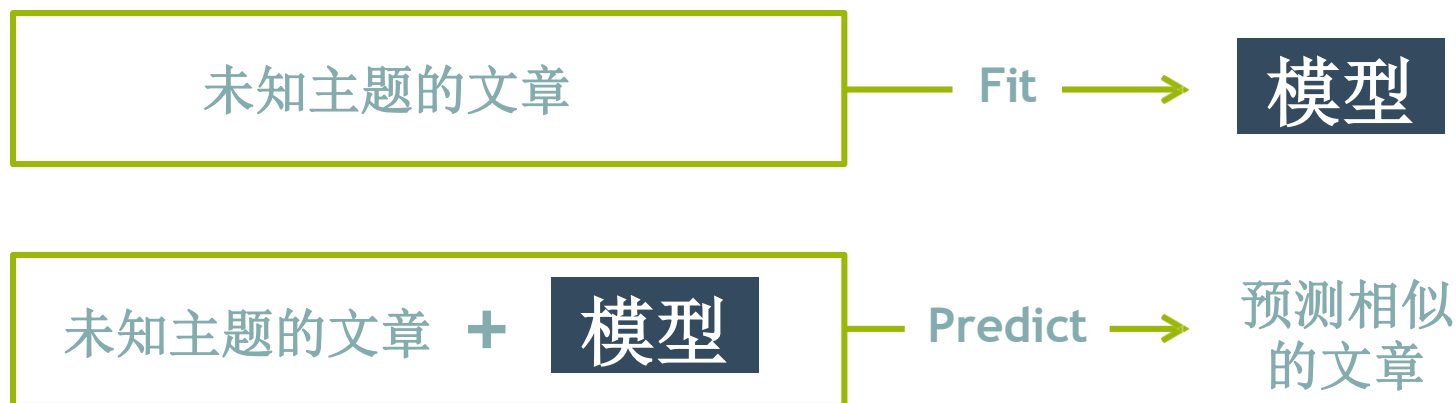
降维

使用数据中的结构特征来简化数据

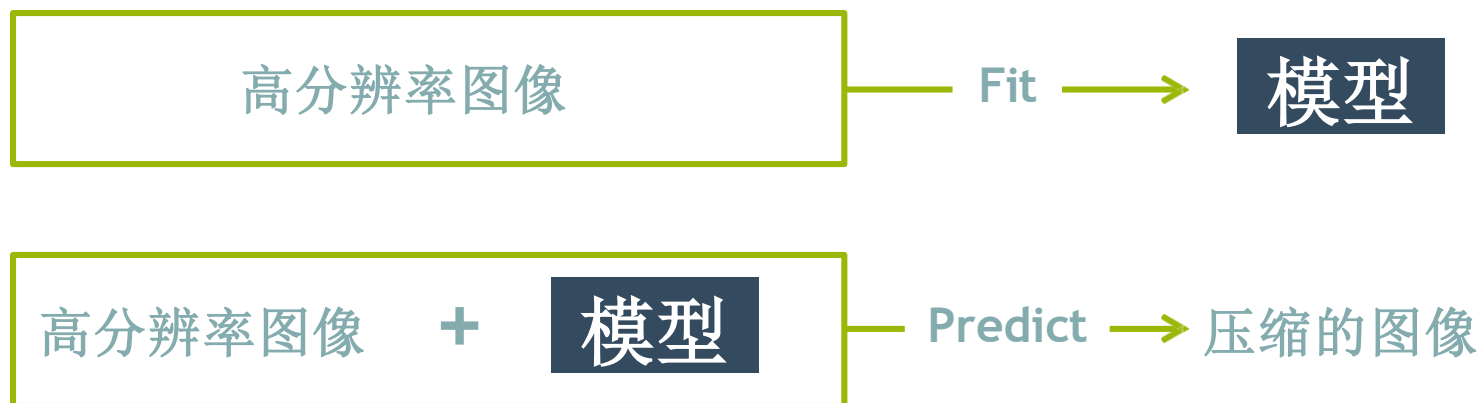
无监督学习概述



聚类：发现不同的数据组



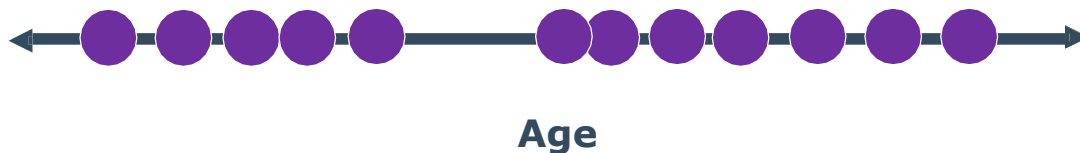
降维：简化结构



聚类算法

一个web应用的所有用户：

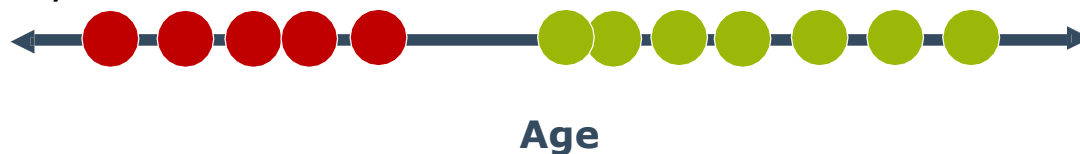
- 一个特征 (age)



聚类算法

一个web应用的所有用户：

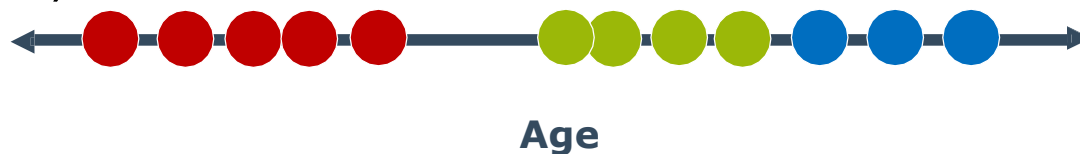
- 一个特征 (age)
- 两个聚簇 (cluster)



聚类算法

一个web应用的所有用户：

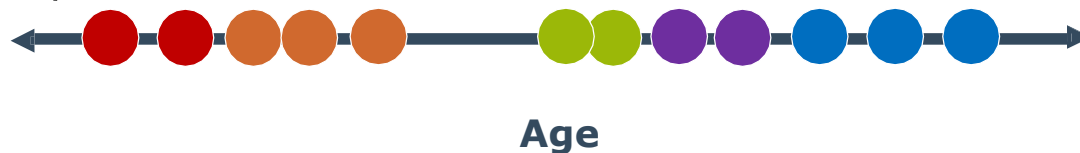
- 一个特征 (age)
- 三个聚簇 (cluster)



聚类算法

一个web应用的所有用户：

- 一个特征 (age)
- 五个聚簇 (cluster)



聚类算法

目标是寻找数据内在的分布，推导出数据的隐式标签，利用隐式标签对数据分组，使得每个组内数据具有相似的分布，而组间的数据分布不同。

假设样本集 $D = \{x_1, x_2, \dots, x_N\}$ 包含 N 个无标记样本，则聚类算法的目标是将样本集 D 划分为 K 个不相交的簇 $\{C_k | k = 1, 2, \dots, K\}$ ，其中任意两个簇的样本没有交集，并且所有簇的样本组合成全部样本集 D 。

用 $\lambda_n \in \{1, 2, \dots, K\}$ 表示样本 x_n 的“簇标签”，即 $x_n \in C_{\lambda_n}$ ，于是聚类的结果可用包含 N 个元素的簇标签向量 $\lambda_n = (\lambda_1, \lambda_2, \dots, \lambda_N)$ 表示。

聚类算法

常用的聚类算法：

- 原型聚类：假设聚类结构能通过一组原型刻画，先对原型进行初始化，然后对原型进行迭代更新求解
 - K-means (K均值) 算法：初始随机选取样本作为质心
 - 学习向量量化：初始随机选取样本作为原型向量
 - 高斯混合聚类：概率模型表达聚类原型
- 密度聚类：假设聚类结构能通过样本分布的紧密程度确定，从密度的角度考察样本之间的可连接性，并基于可连接样本不断扩展聚类簇
- 层次聚类：试图在不同层次对数据集进行划分，形成树形的聚类结构

K-MEANS聚类算法

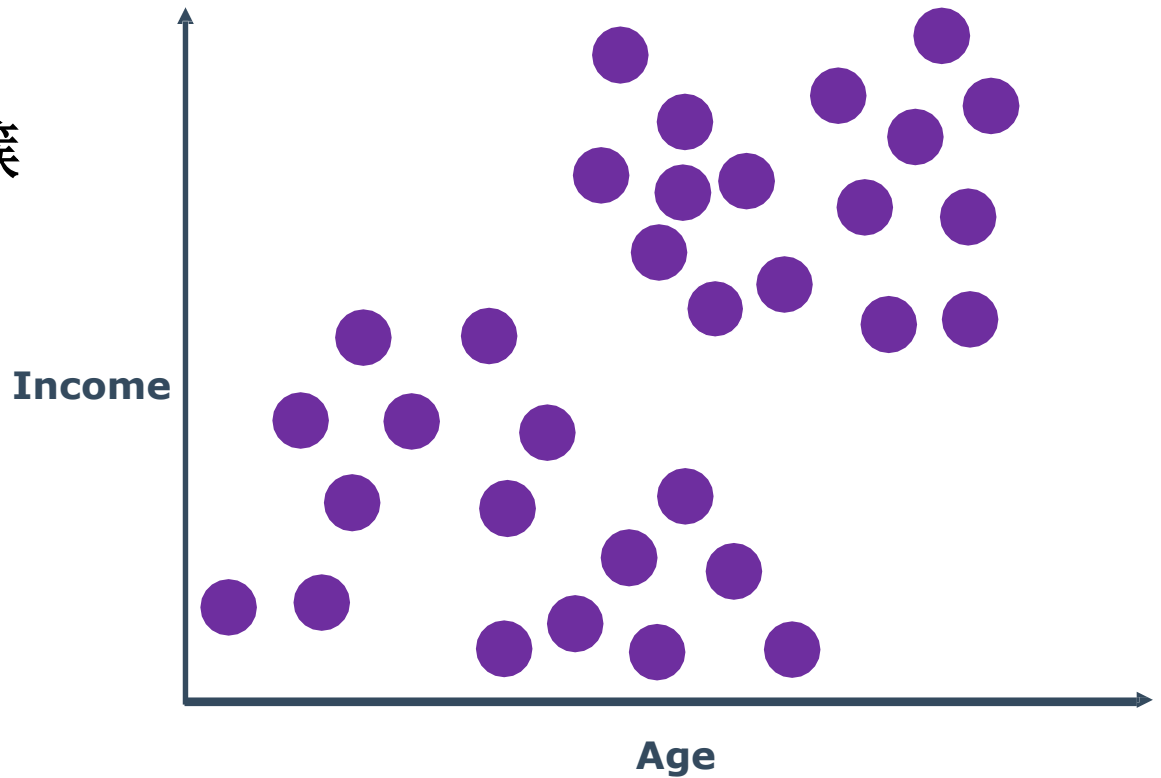
K-Means 算法

大致过程：

- (1) 随机选取K个样本点作为K个初始簇的质心(centroid)
- (2) 计算其他每个样本点到这K个质心的距离。对于每一个样本 x_n ，找到离它最近的质心 λ_n ，将 x_n 划分到第 λ_n 个簇中，并将 x_n 的簇标签置为 λ_n
- (3) 当所有样本所属的簇都更新完毕以后，对于更新后的每个簇计算其包含的所有样本的平均向量（中心点），作为新的质心
- (4) 重复步骤(2)(3)，迭代至所有质心都不变化为止

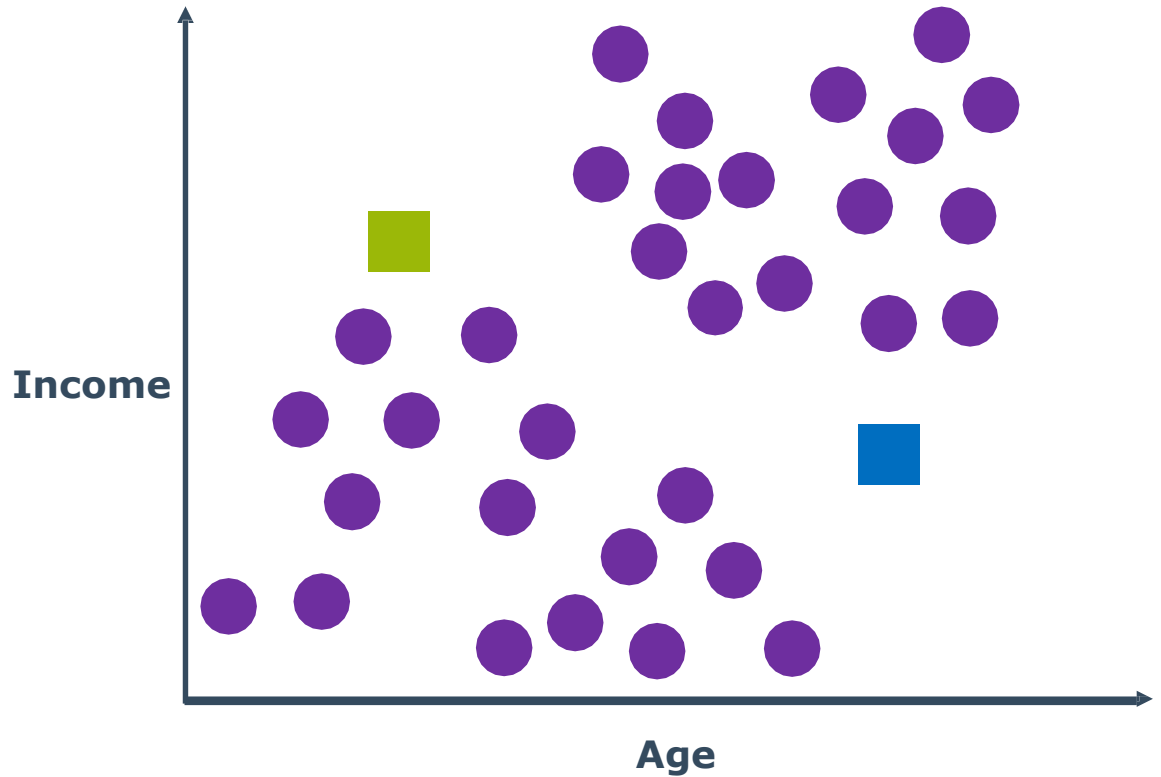
K-Means 算法

K = 2: 发现两个聚簇



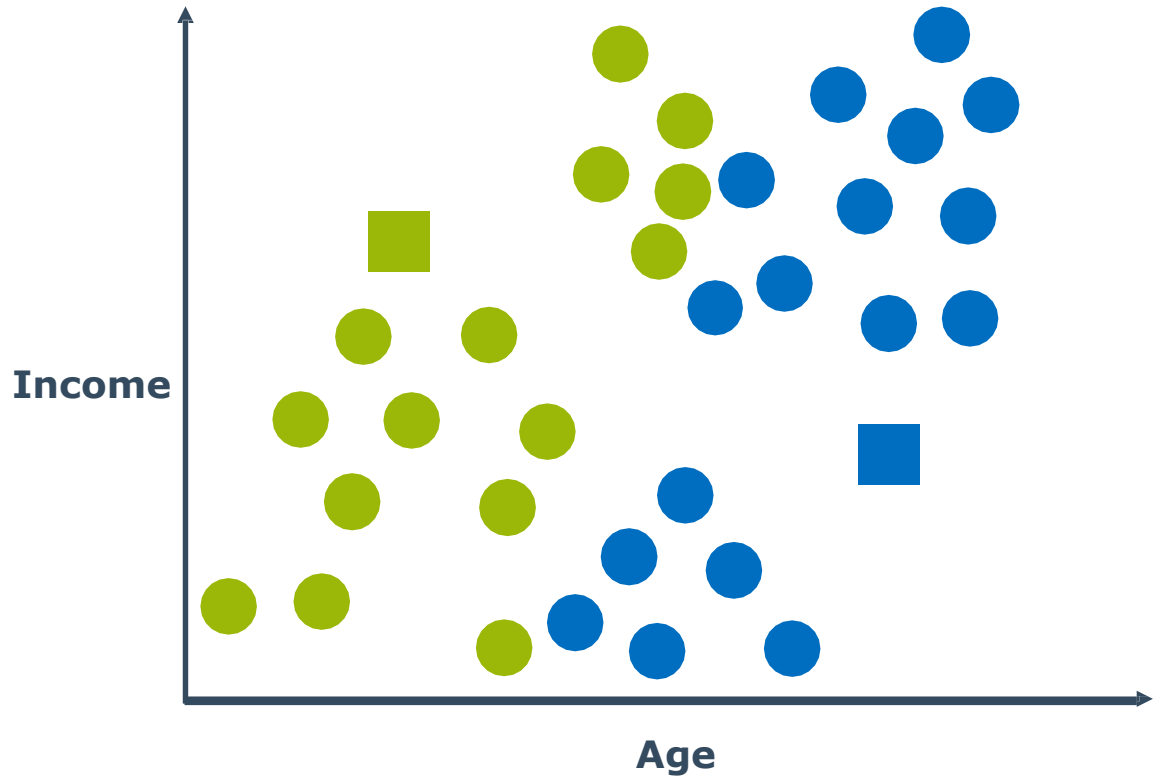
K-Means 算法

$K = 2$, 随机地选择
两个聚簇的中心点



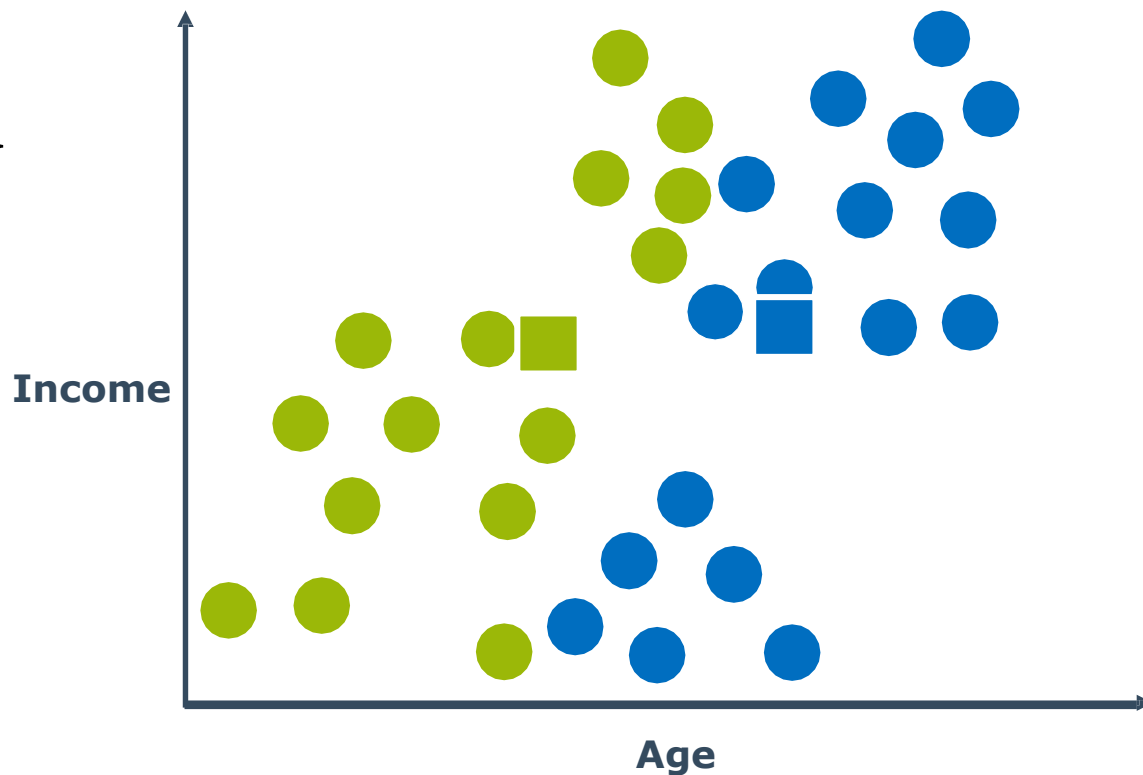
K-Means 算法

$K = 2$, 每个点被聚到离它最近的中心点代表的聚簇中



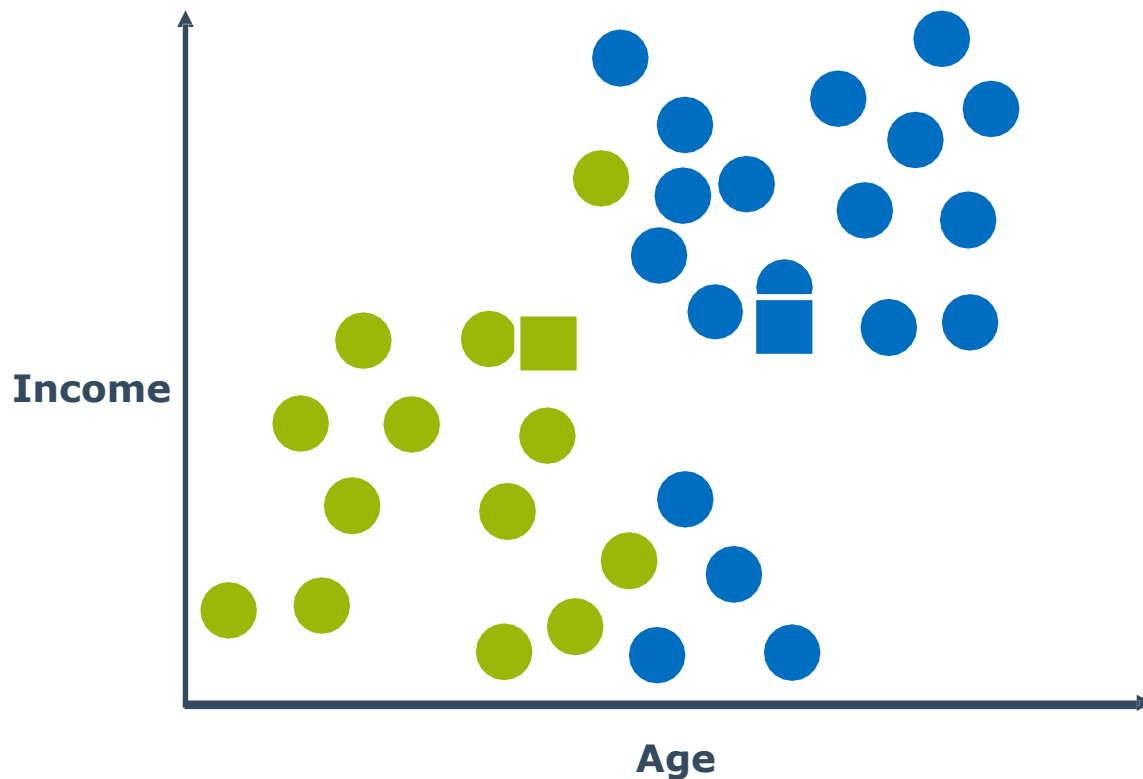
K-Means 算法

$K = 2$, 重新计算每个聚簇的中心点



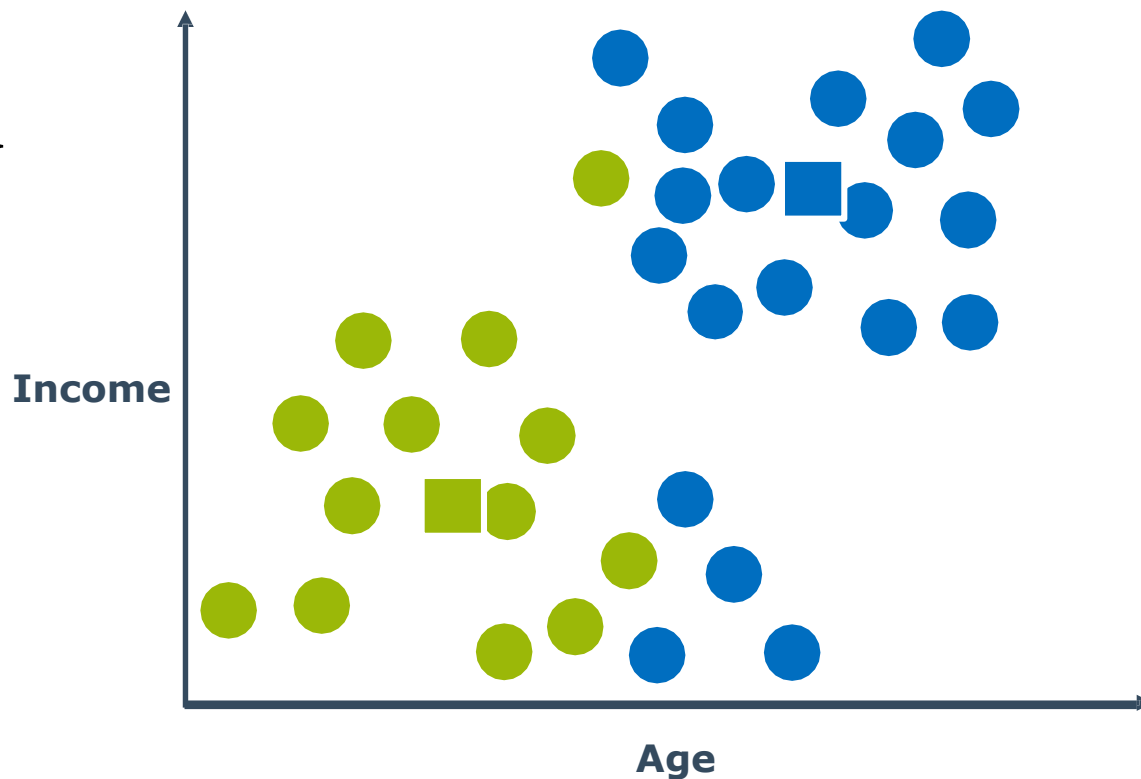
K-Means 算法

$K = 2$, 每个点属于
离它最近的中心点



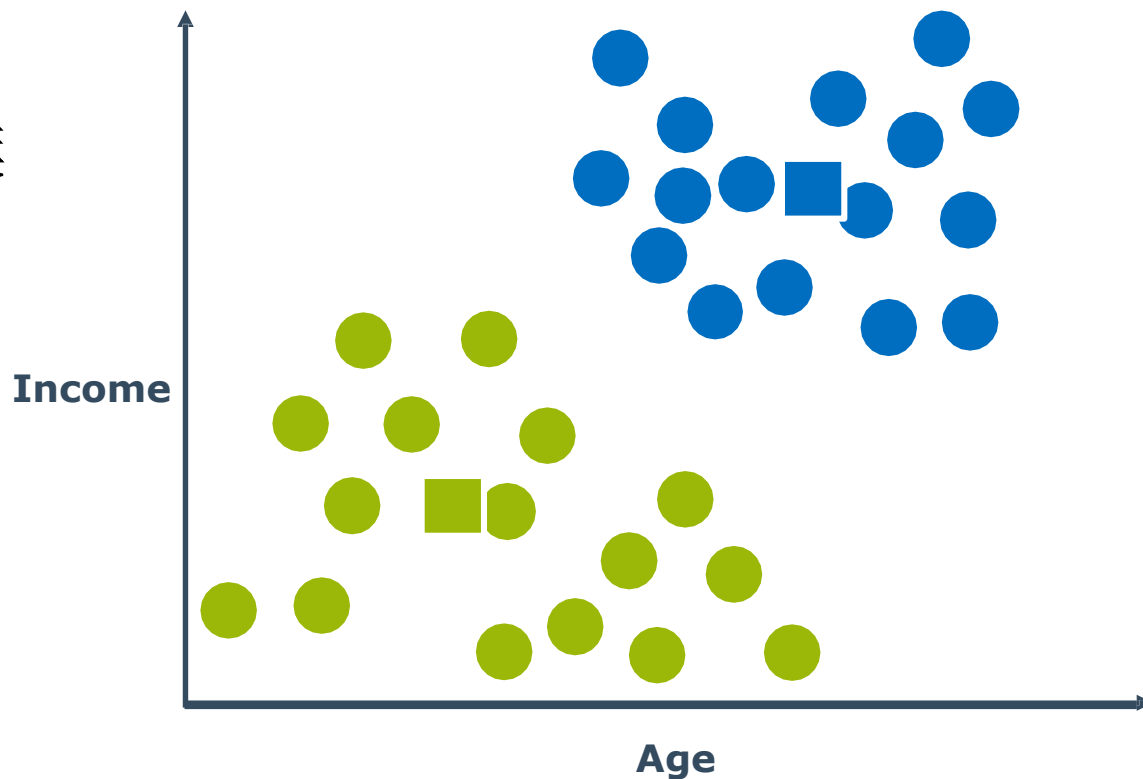
K-Means 算法

$K = 2$, 重新计算每个聚簇的中心点



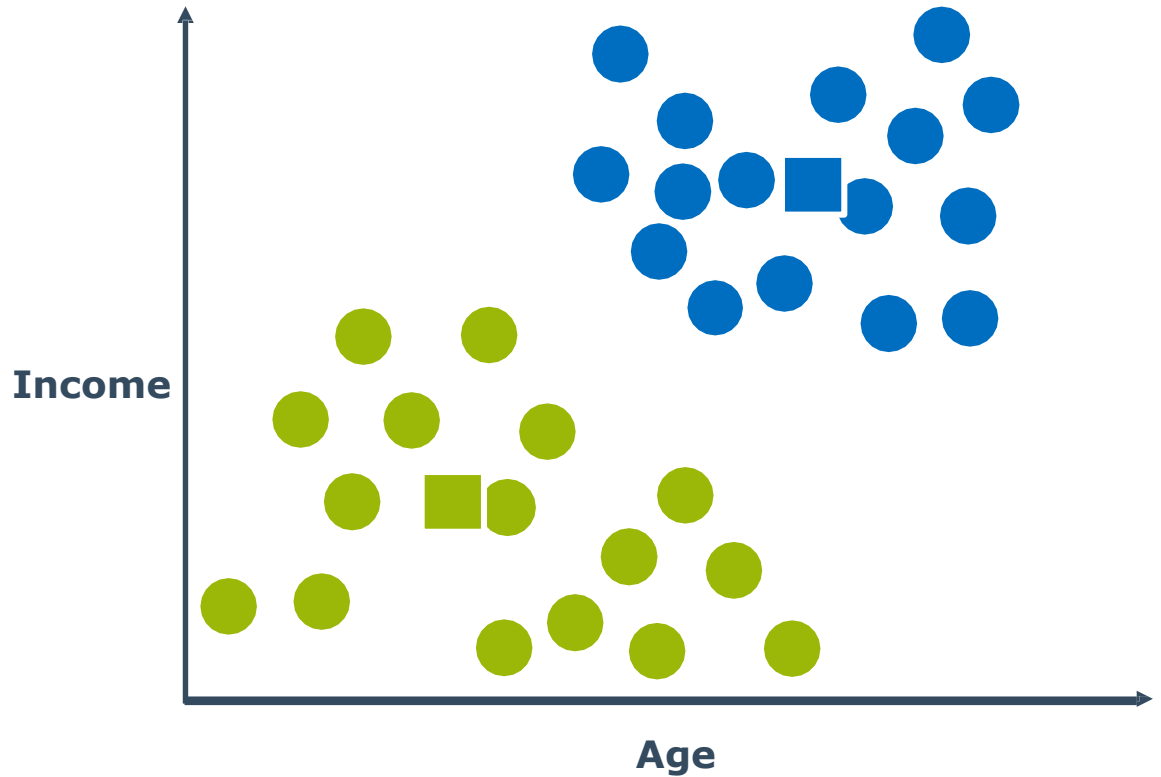
K-Means 算法

$K = 2$, 数据点的聚簇
不再改变时 → 算法
收敛



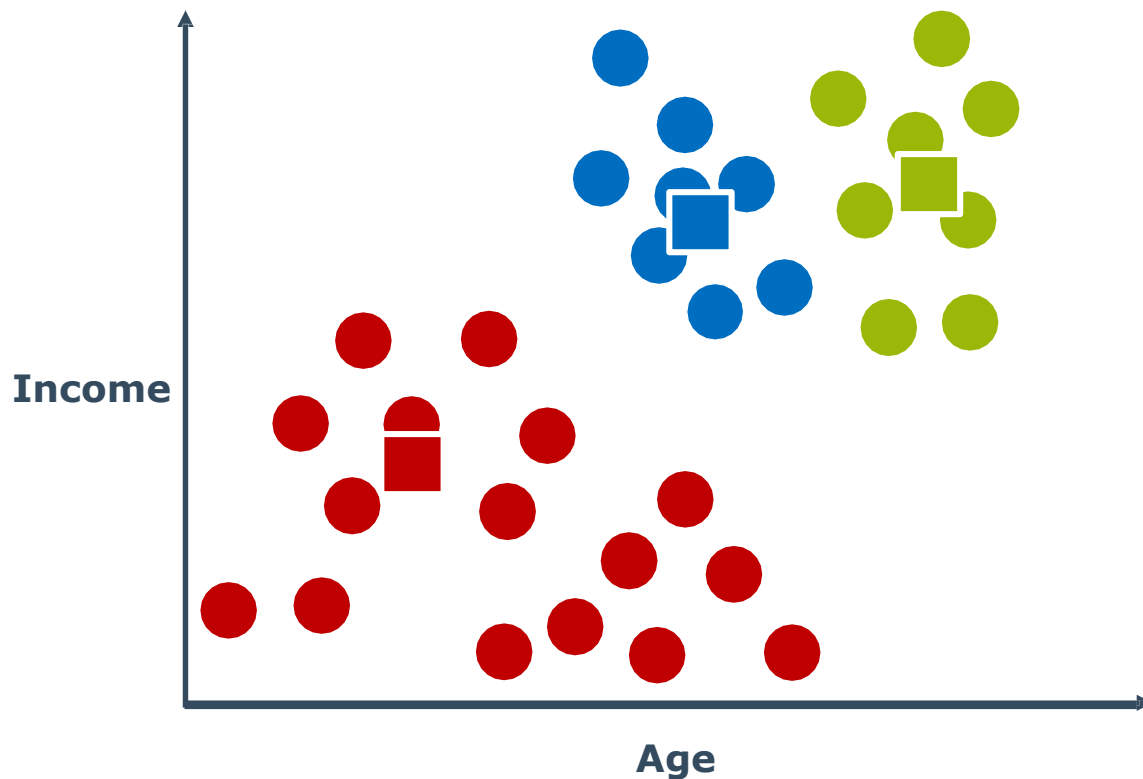
K-Means 算法

$K = 2$, 最后每个点
属于离它最近的那
个聚簇



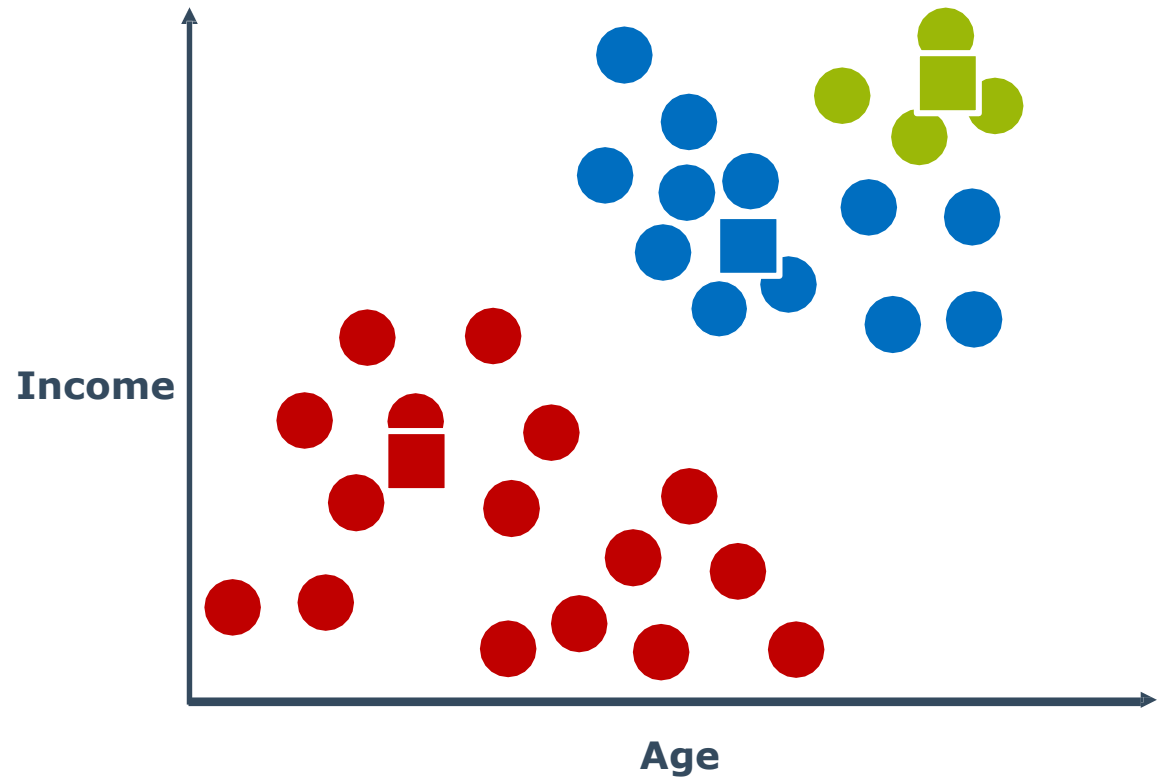
K-Means 算法

$K = 3$, 最终结果依赖于初始的聚簇分配



聚类的评价指标

哪个模型正确？



聚类评价指标

- **Inertia（簇内平方和）：**

- 每个数据点(x_i)距其聚簇中心(C_k)的距离平方和

$$\sum_{i=1}^n (x_i - C_k)^2$$

- 值越小表示聚簇越紧密

- **轮廓系数（Silhouette Coefficient）：**

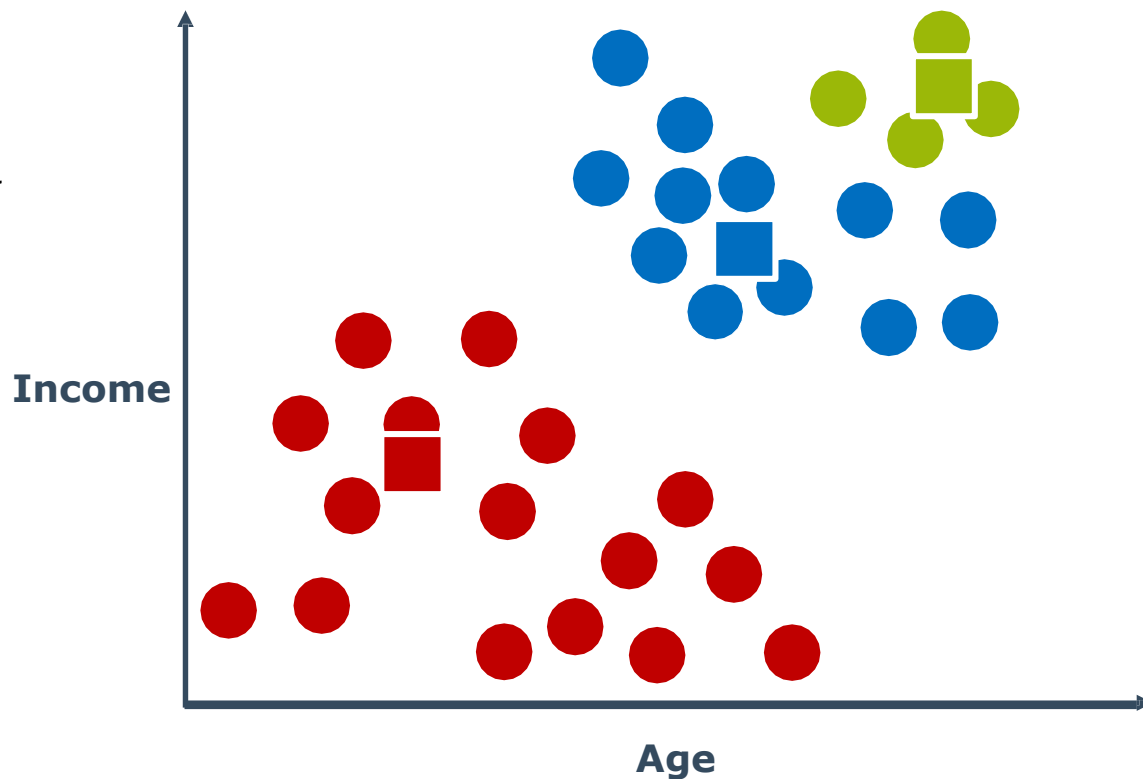
- 对每个数据点计算一个轮廓系数：

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

- a = 此数据点到同簇中所有其他点的平均距离，凝聚度
- b = 此数据点到最近簇中所有点的平均距离，分离度
- 将所有数据点的轮廓系数取平均值就得到一个总的评分
- 取值在 $[-1, 1]$ 之间，值越大，聚类效果越好

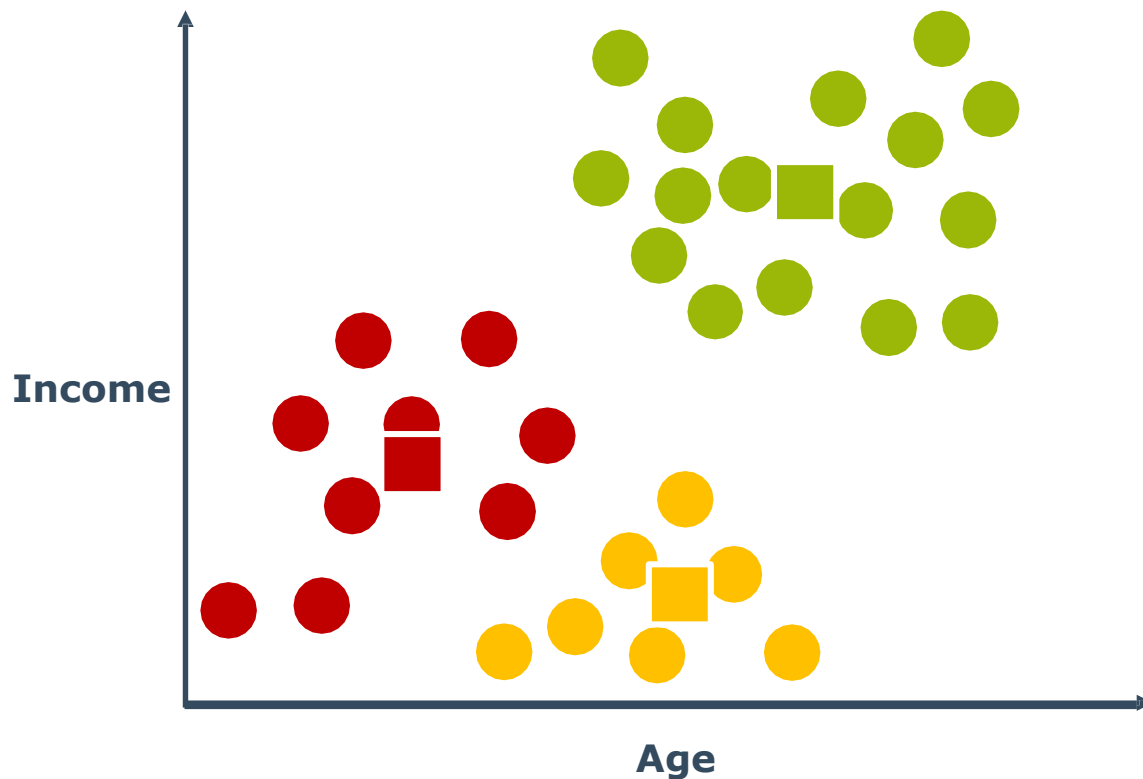
哪个模型正确？

初始化多次，然后选择得分最高的模型



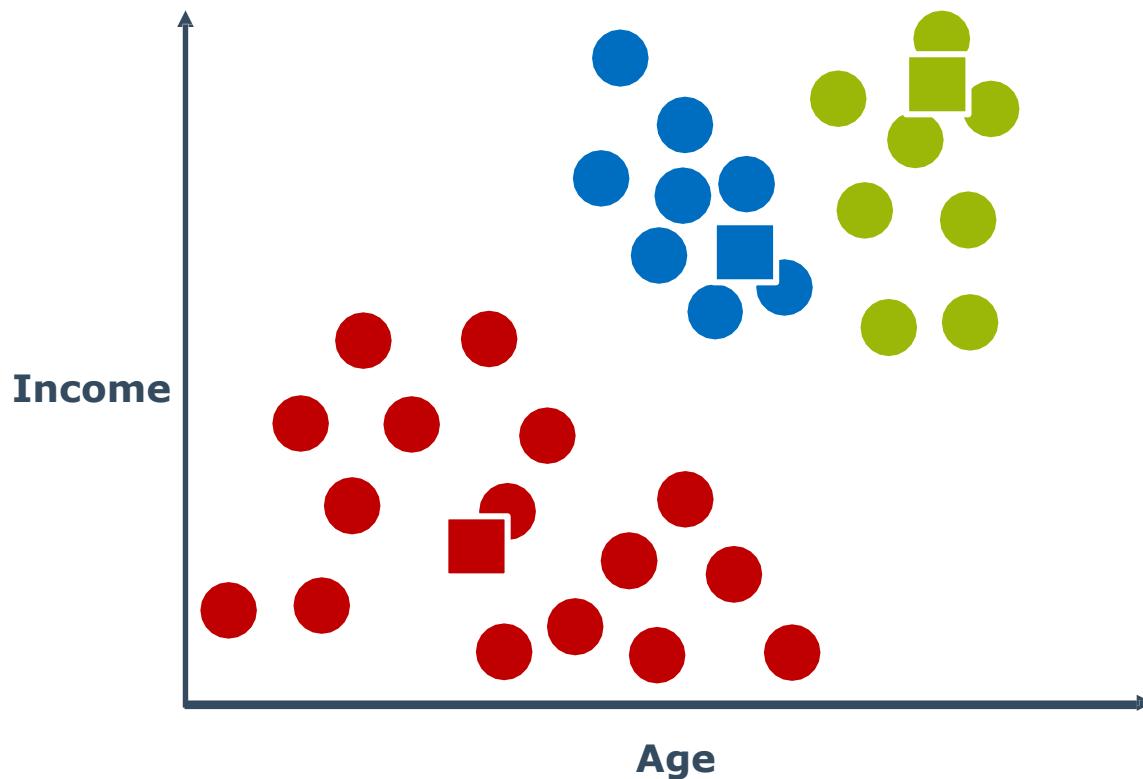
哪个模型正确？

Inertia = 12.645



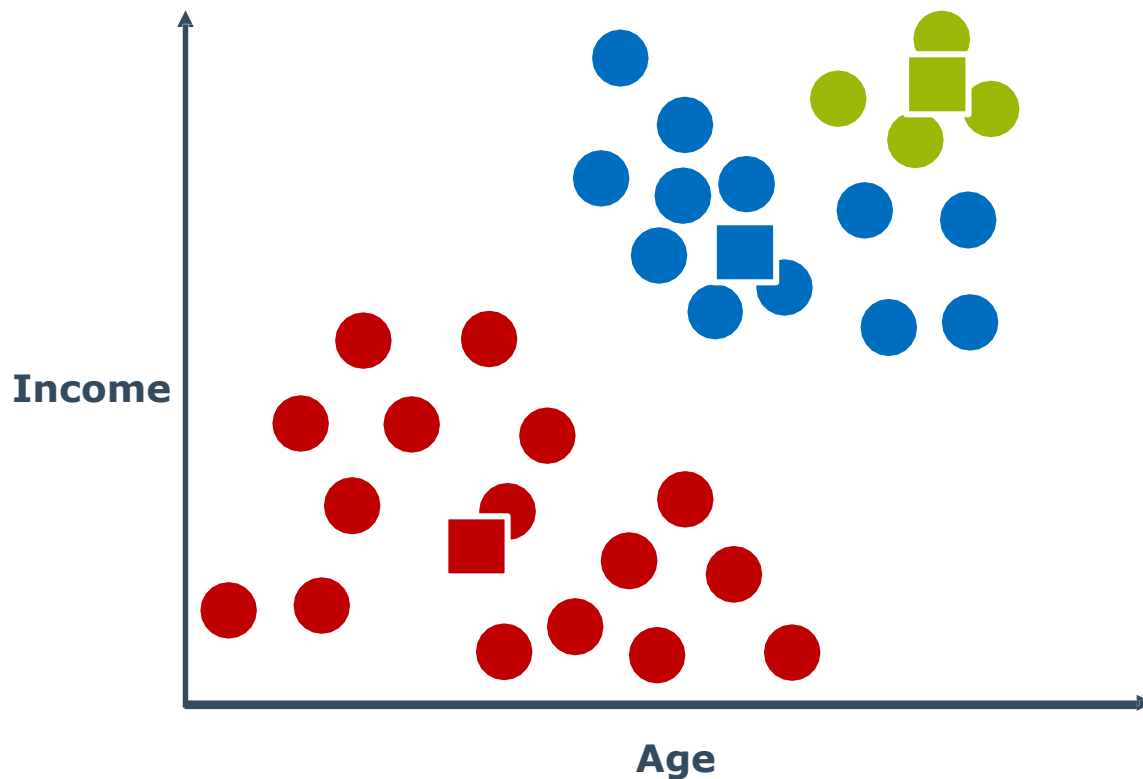
哪个模型正确？

Inertia = 12.943



哪个模型正确？

Inertia = 13.112



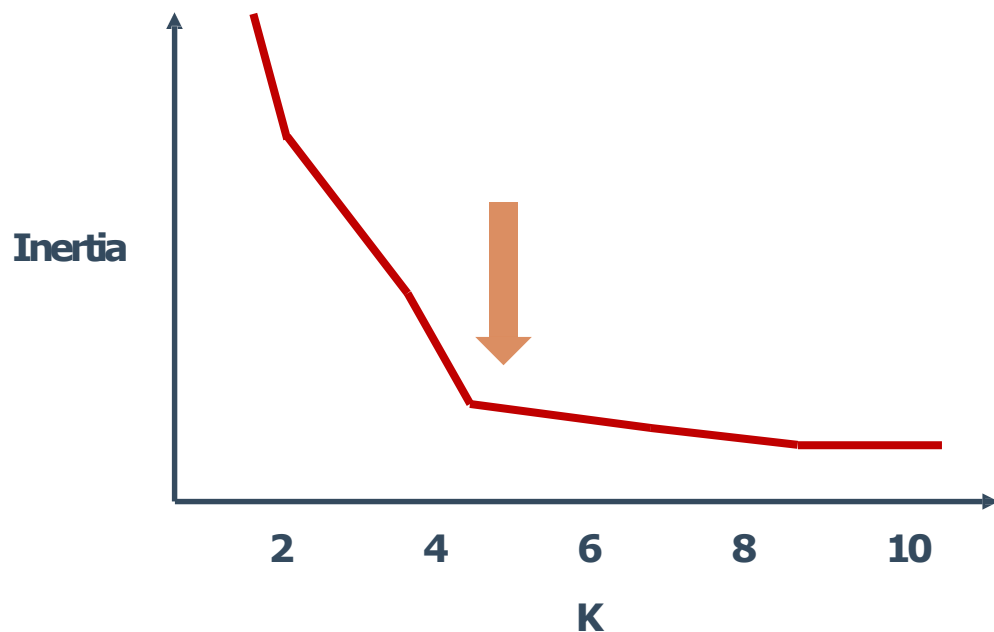
K值的选择方法

选择正确的聚簇数

- 有时应用中有一个**K**:
 - 将相似的任务聚集在四个**CPU**核上 (**K=4**)
 - 一件服装设计成**10**种不同大小以适合不同的人 (**K=10**)
 - 一个导览界面可以浏览**20**种不同科学领域的论文 (**K=20**)

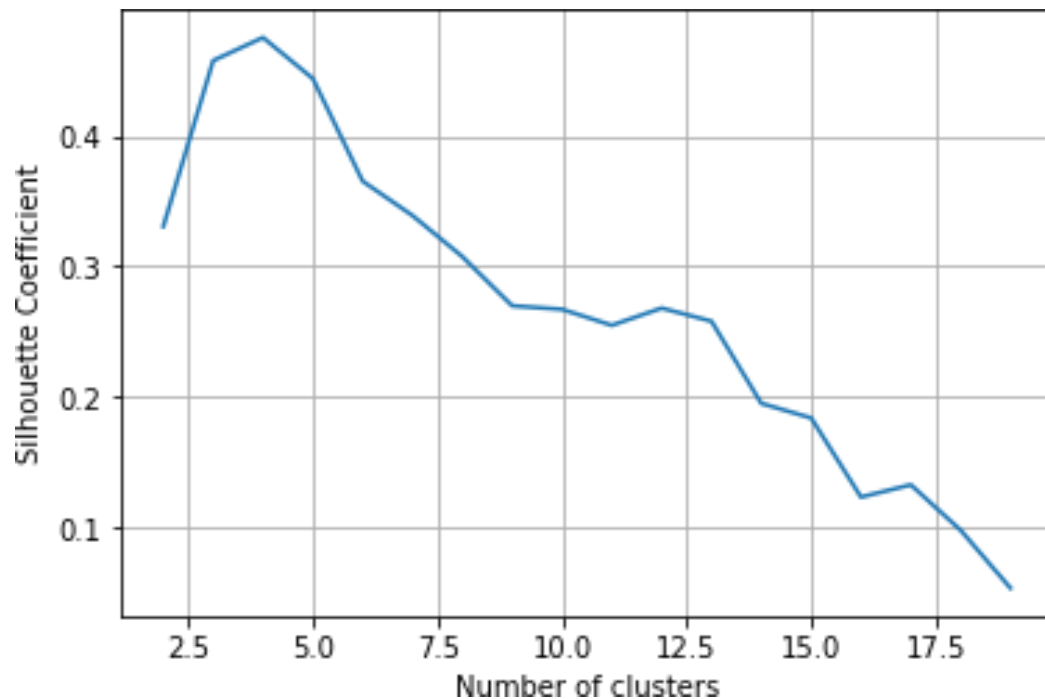
选择正确的聚簇数

- **Inertia**衡量点到聚簇中心的距离
- 值会随着**K**的增大而不断降低，只要聚簇密度在不断增大
- 小于真实**K**值时，下降幅度很大；超过真实**K**值时，下降趋于平缓



选择正确的聚簇数

- 选择轮廓系数最大的K值



K-Means的语法

导入包含聚类方法的类:

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

创建该类的一个对象:

```
kmeans = KMeans(n_clusters=3,  
                 init='k-means++')
```

拟合数据, 并在新数据上预测聚簇:

```
kmeans = kmeans.fit(X1)  
y_predict = kmeans.predict(X2)
```

<http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

K-Means的语法

导入包含聚类方法的类:

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

创建该类的一个对象:

```
kmeans = KMeans(n_clusters=3,  
                 init='k-means++')
```



最终的聚簇数

拟合数据，并在新数据上预测聚簇:

```
kmeans = kmeans.fit(X1)  
y_predict = kmeans.predict(X2)
```

K-Means的语法

导入包含聚类方法的类:

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

创建该类的一个对象:

```
kmeans = KMeans(n_clusters=3,  
                 init='k-means++')
```



kmeans++
初始化方法

拟合数据, 并在新数据上预测聚簇:

```
kmeans = kmeans.fit(X1)  
y_predict = kmeans.predict(X2)
```

K-Means的语法

导入包含聚类方法的类:

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

创建该类的一个对象:

```
kmeans = KMeans(n_clusters=3,  
                 init='k-means++')
```

拟合数据, 并在新数据上预测聚簇:

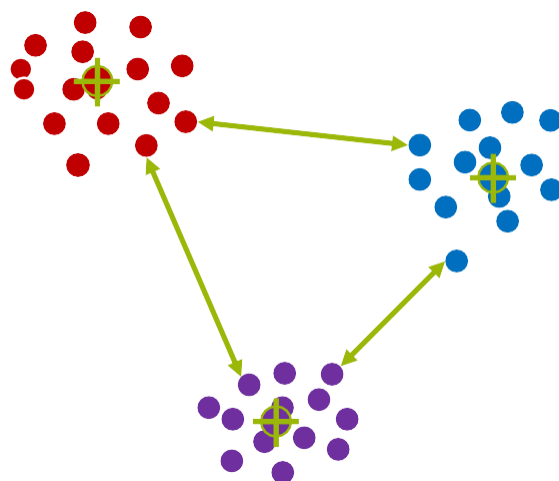
```
kmeans = kmeans.fit(X1)  
y_predict = kmeans.predict(X2)
```

也可以用**MiniBatchKMeans**使用批处理方式

距离指标

距离指标的选择

- 距离指标的选择对聚类的成功至关重要
- 每个指标有各自的优点和适用情况
- 但有时距离指标的选择也是基于经验性的评价



距离指标的选择

属性分为连续属性和离散属性，连续属性也称有序属性，在定义域范围内有无穷多值；离散属性也称无序属性，在定义域范围内只有有限个取值。

根据属性的不同，距离指标分为有序距离指标、无序距离指标性和混合距离指标。

有序属性可以采用闵可夫斯基距离作为距离评价指标。

无序属性可以采用价值差异指标 VDM (Value Difference Metric) 作为距离评价指标。

混合属性可以采用闵可夫斯基距离和VDM结合起来的方式作为距离评价指标。

闵可夫斯基距离（Minkowski Distance）：

$$d_{12} = \sqrt[p]{\sum_{k=1}^n |x_{1k} - x_{2k}|^p}$$

其中 p 是一个变参数：

当 $p=1$ 时，曼哈顿距离；

当 $p=2$ 时，欧氏距离；

当 $p \rightarrow \infty$ 时，切比雪夫距离；

根据的 p 不同，**闵氏距离**可以表示某一类/种距离。

欧氏距离（Euclidean Distance）：

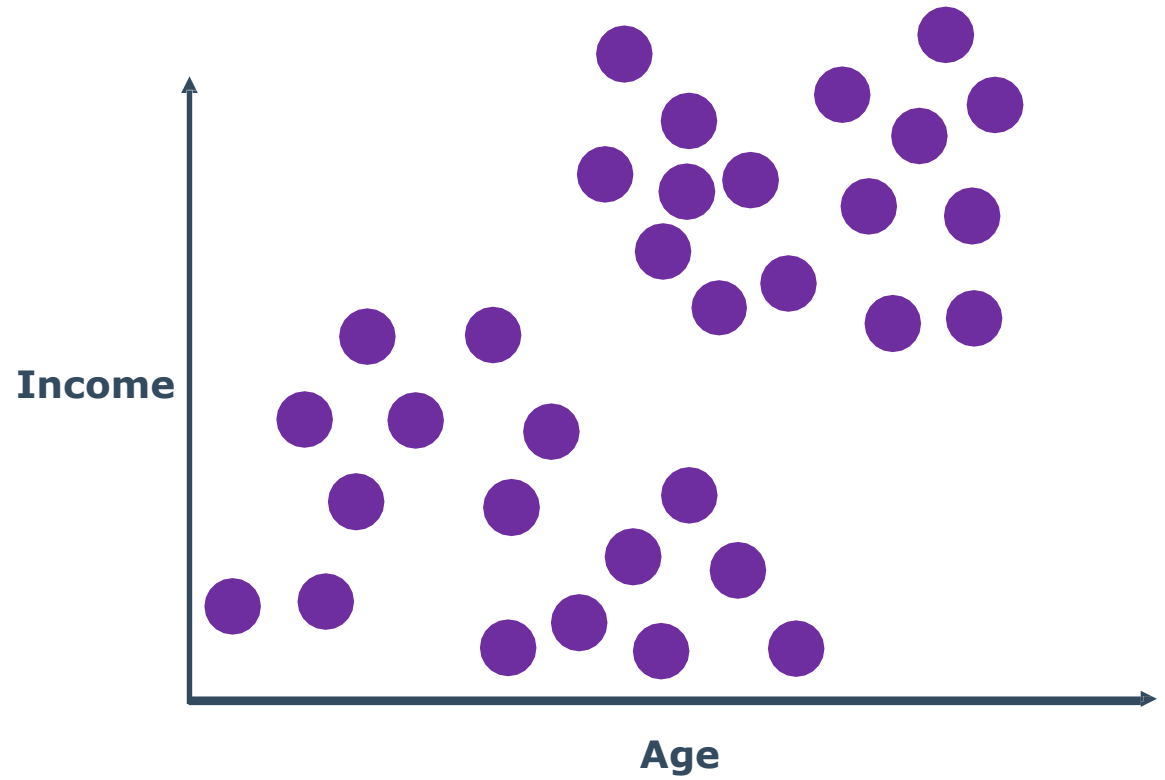
也叫欧几里得距离， L2距离

二维：
$$d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

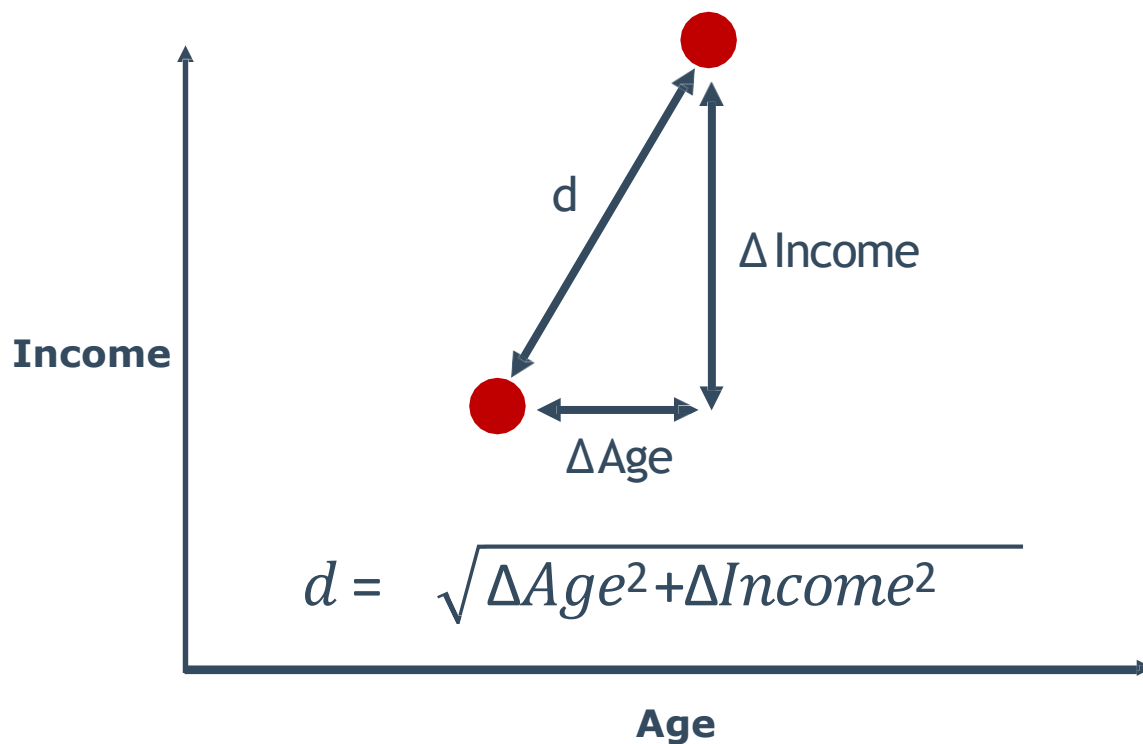
三维：
$$d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

n维：
$$d_{12} = \sqrt{\sum_{k=1}^n (x_{1k} - x_{2k})^2}$$

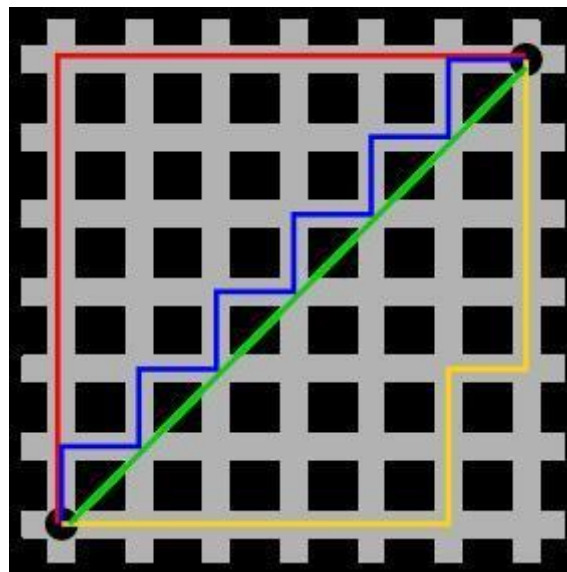
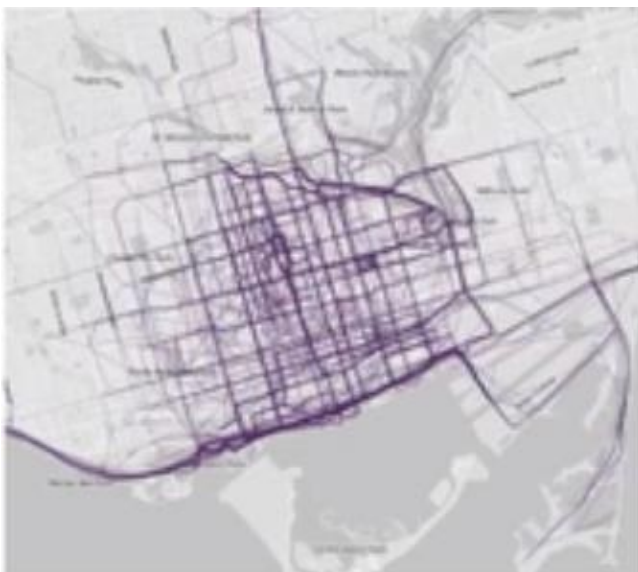
欧几里得距离



欧几里得距离（L2距离）



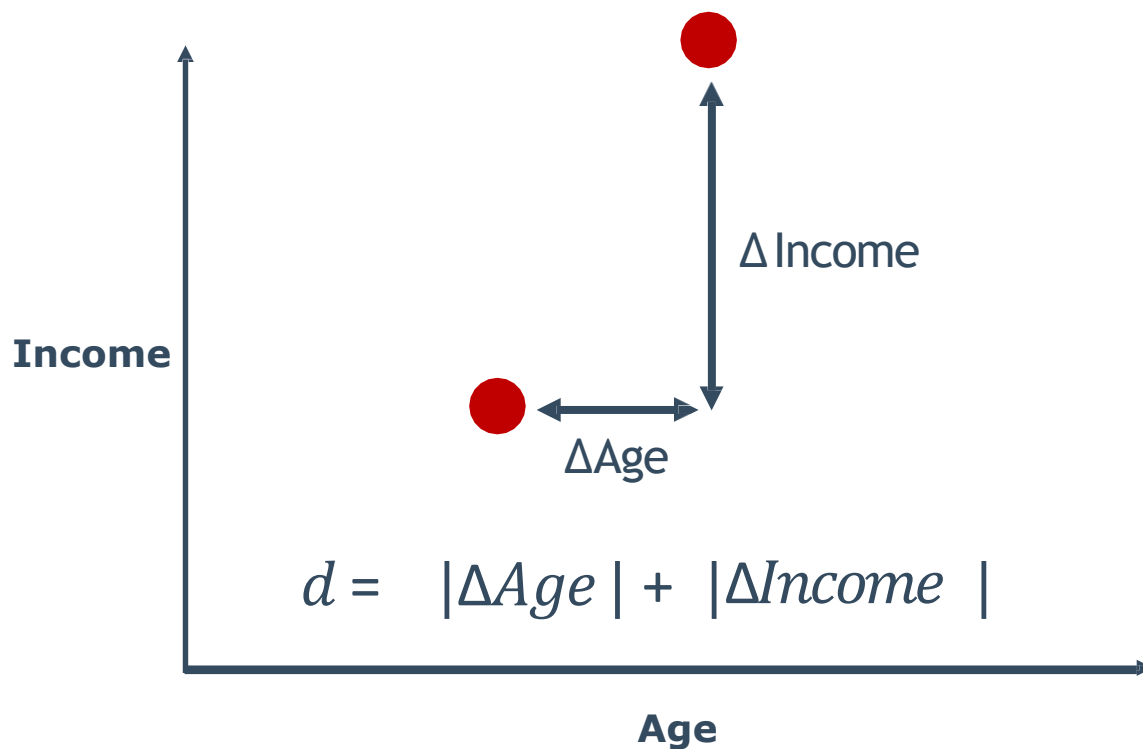
曼哈顿距离（Manhattan Distance）：
也叫出租车距离，街区距离，L1距离



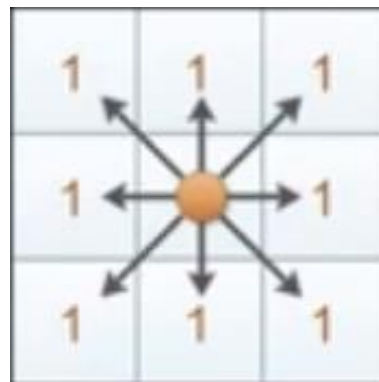
二维： $d_{12} = |x_1 - x_2| + |y_1 - y_2|$

n维： $d_{12} = \sum_{k=1}^n |x_{1k} - x_{2k}|$

曼哈顿距离（L1或街区距离）



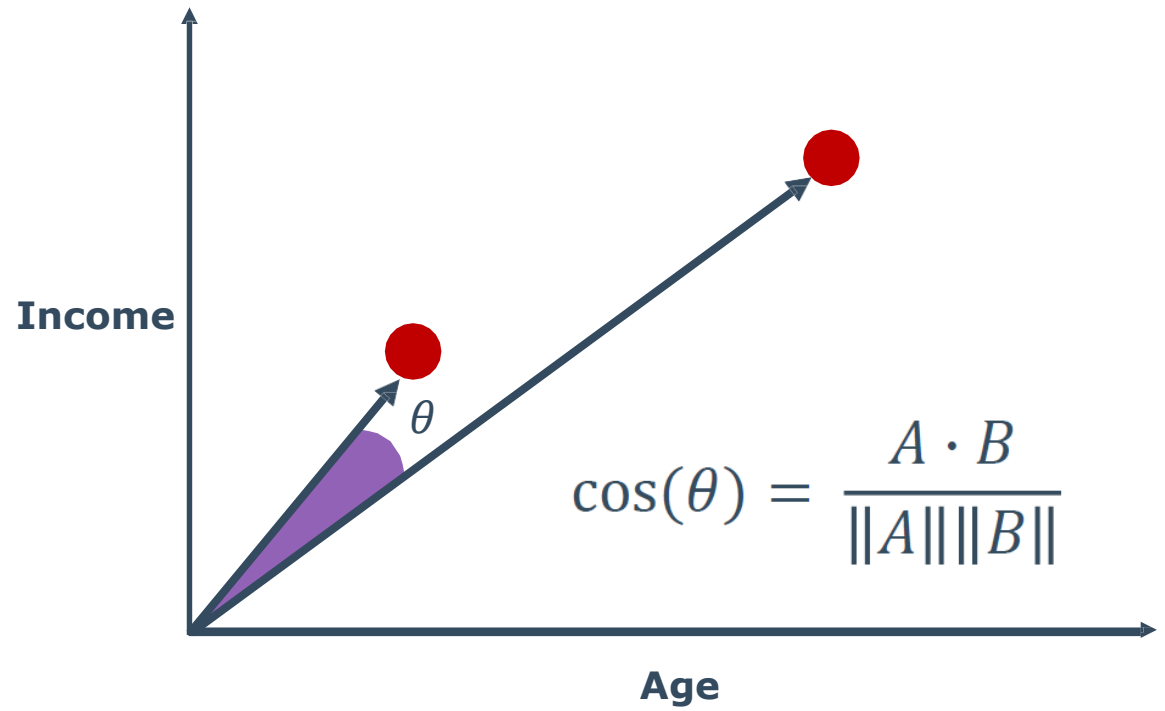
切比雪夫距离 (Chebyshev Distance) :



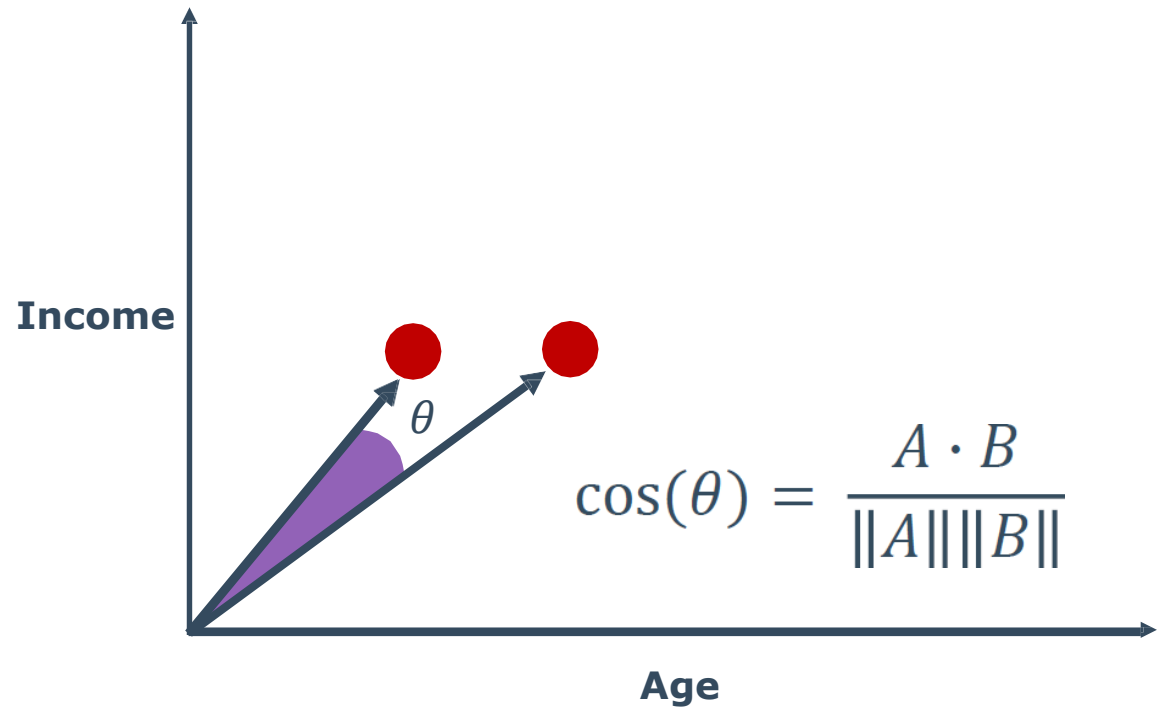
二维: $d_{12} = \max(|x_1 - x_2|, |y_1 - y_2|)$

n维: $d_{12} = \max(|x_{1k} - x_{2k}|)$

余弦距离



余弦距离



欧几里得vs.余弦距离

- 欧几里得距离适用于基于坐标的度量
- 余弦距离更适合那些出现位置不重要的数据，例如文本数据
- 欧几里得距离对维度灾难更敏感

VDM (Value Difference Metric) :

$$VDM_p = \sum_{k=1}^K \left| \frac{m_{u,a,k}}{m_{u,a}} - \frac{m_{u,b,k}}{m_{u,b}} \right|^p$$

其中 $m_{u,a}$ 表示属性u上取值为a的样本数； $m_{u,a,k}$ 为第k个样本簇中在属性u上取值为a的样本数；K为样本簇数。

Jaccard距离

杰卡德距离(Jaccard Distance) 是用来衡量两个集合差异性的一种指标，它是杰卡德相似系数的补集，被定义为1减去Jaccard相似系数。而杰卡德相似系数(Jaccard similarity coefficient)，也称杰卡德指数(Jaccard Index)，是用来衡量两个集合相似度的一种指标。

Jaccard相似系数:
$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Jaccard距离:
$$d_J(A, B) = 1 - J(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

Jaccard距离

应用于集合（例如单词的出现）

- 句子 **A**: “I like chocolate ice cream.”
- set A = {I, like, chocolate, ice, cream}
- 句子 **B**: “Do I want chocolate cream or vanilla cream?”
- set B = {Do, I, want, chocolate, cream, or, vanilla}

$$1 - \frac{A \cap B}{A \cup B} = 1 - \frac{\text{len}(\text{shared})}{\text{len}(\text{unique})}$$

Jaccard距离

应用于集合（例如单词的出现）

- 句子A: “I like chocolate ice cream.”
- set A = {I, like, chocolate, ice, cream}
- 句子 B: “Do I want chocolate cream or vanilla cream?”
- set B = {Do, I, want, chocolate, cream, or, vanilla}

$$1 - \frac{A \cap B}{A \cup B} = 1 - \frac{3}{9}$$

距离指标的语法

导入一般的两两距离计算函数：

```
from sklearn.metrics import pairwise_distances
```

计算距离：

```
dist = pairwise_distances(X,Y,  
                           metric='euclidean')
```



选择的距离指标

距离指标的语法

导入一般的两两距离计算函数：

```
from sklearn.metrics import pairwise_distances
```

计算距离：

```
dist = pairwise_distances(X,Y,  
                           metric='euclidean')
```

其他的距离指标选择有：cosine, manhattan, jaccard, 等等

距离指标的语法

导入一般的两两距离计算函数：

```
from sklearn.metrics import pairwise_distances
```

计算距离：

```
dist = pairwise_distances(X,Y,  
                           metric='euclidean')
```

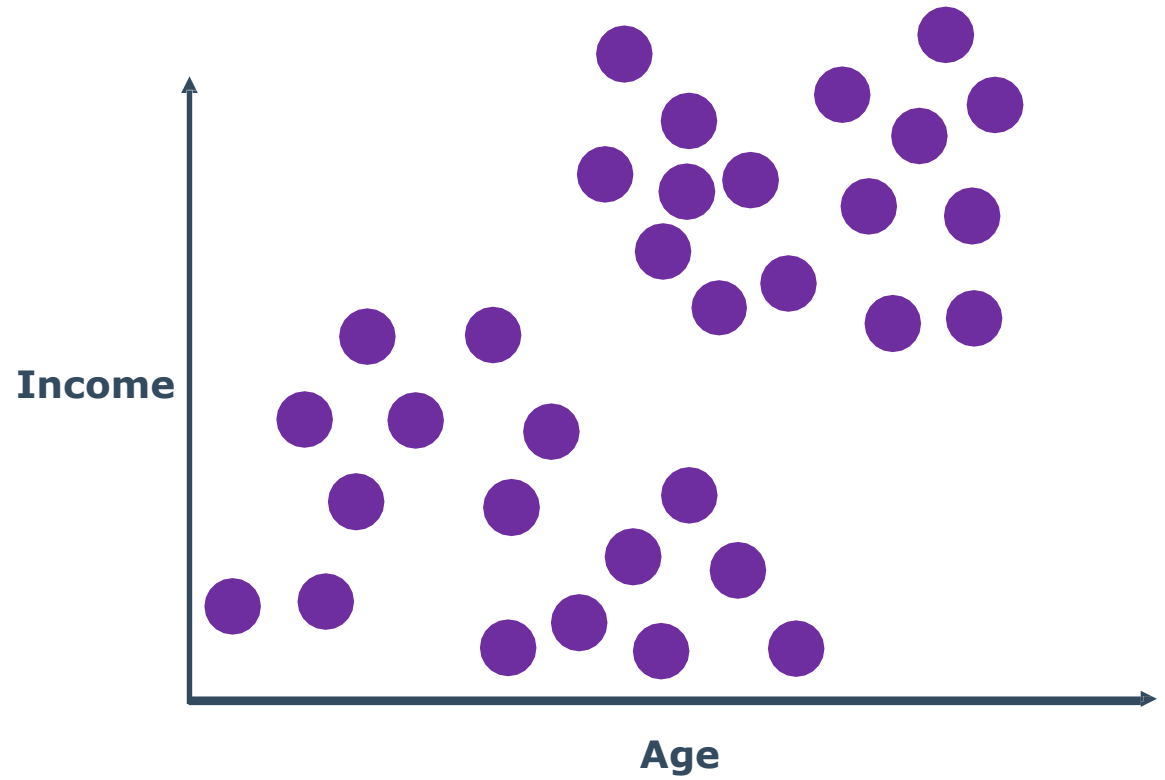
其他的距离指标选择有：cosine, manhattan, jaccard, 等等

距离指标函数也可以被专门导入，如：

```
from sklearn.metrics import euclidean_distances
```

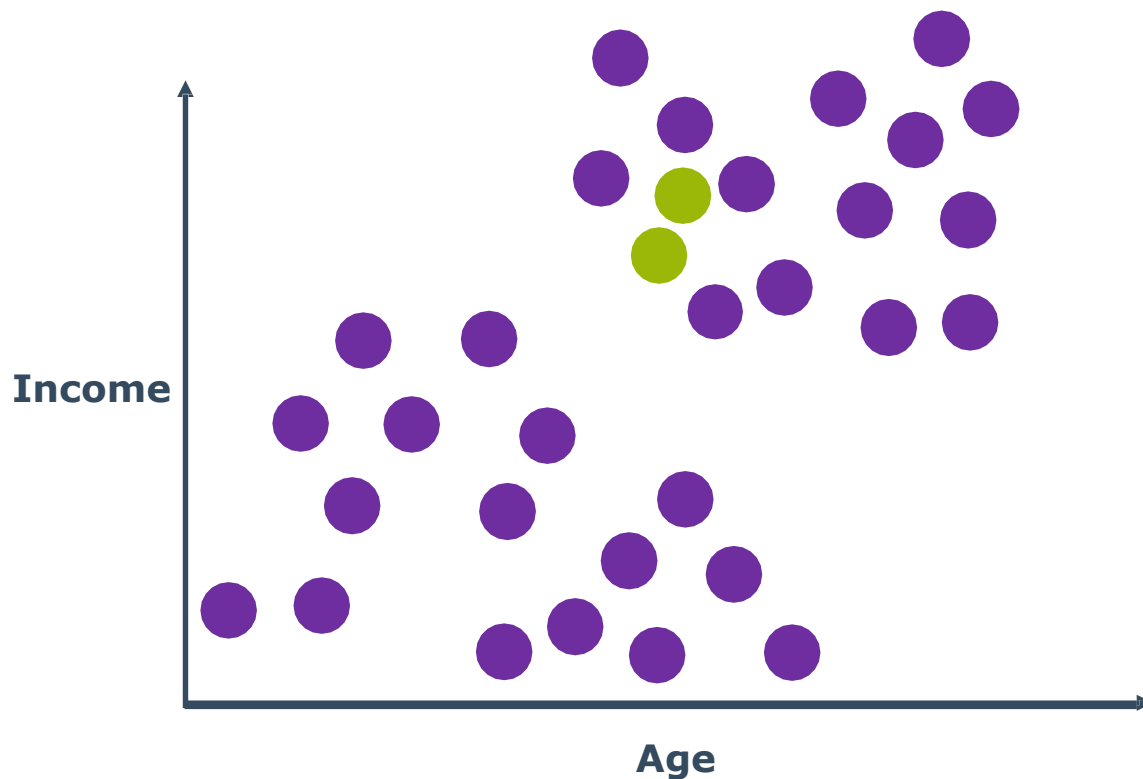
聚合式层次聚类

聚合式层次聚类



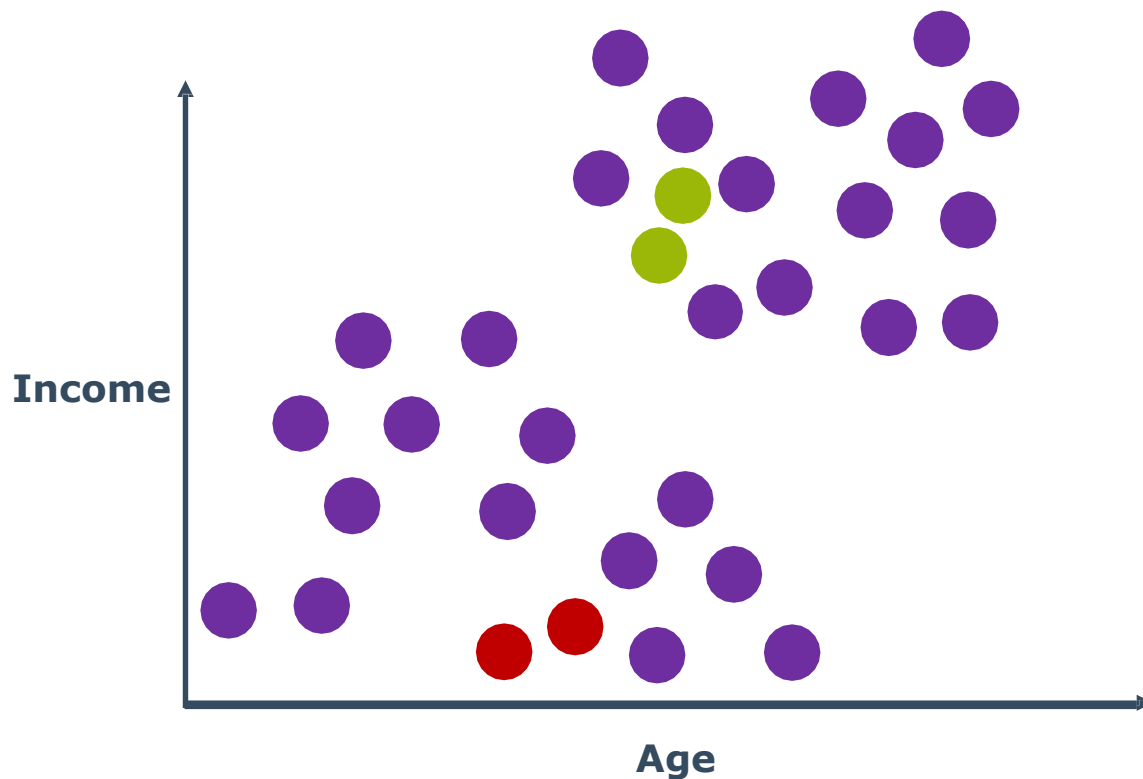
聚合式层次聚类

寻找最近的一对点，
聚成一个聚簇



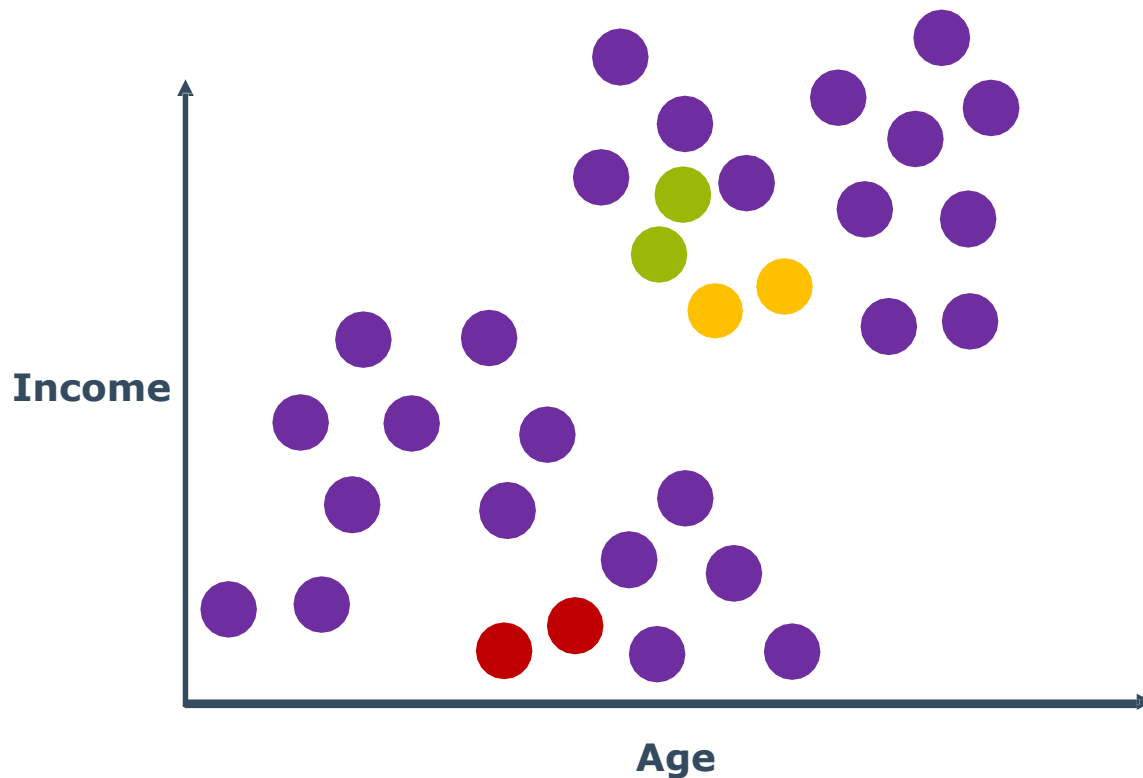
聚合式层次聚类

寻找下一对最近的点，并合并



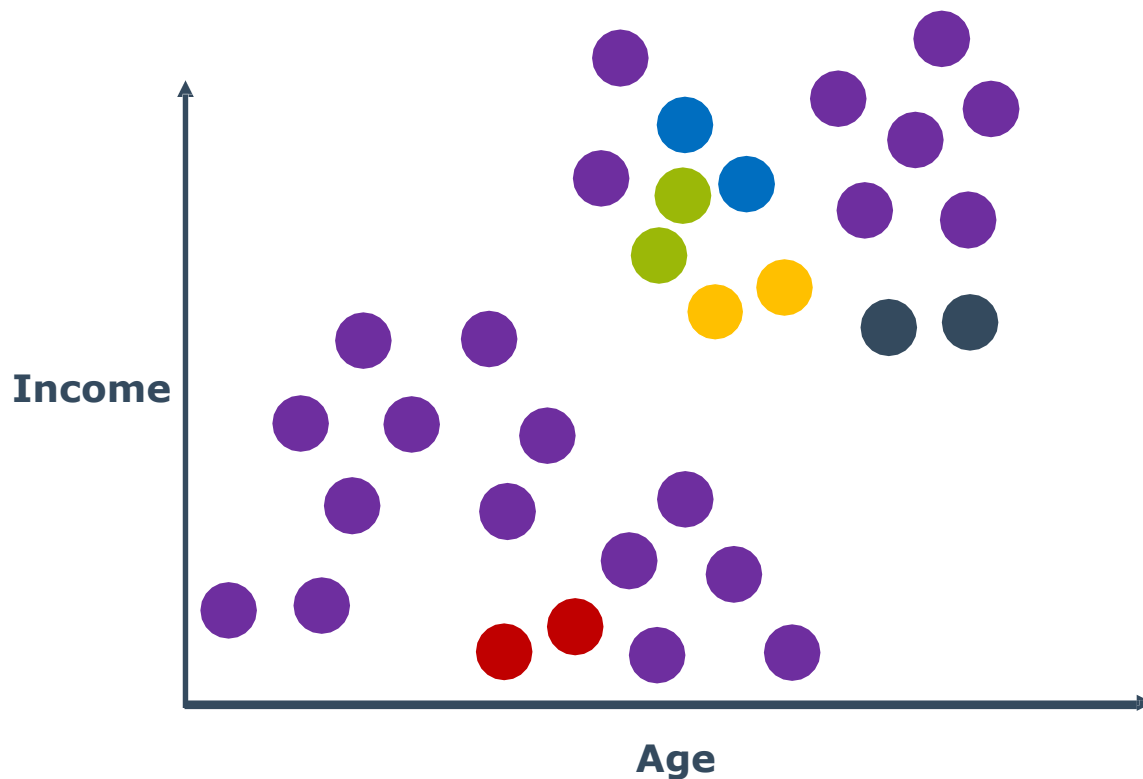
聚合式层次聚类

寻找下一对最近的点，并合并



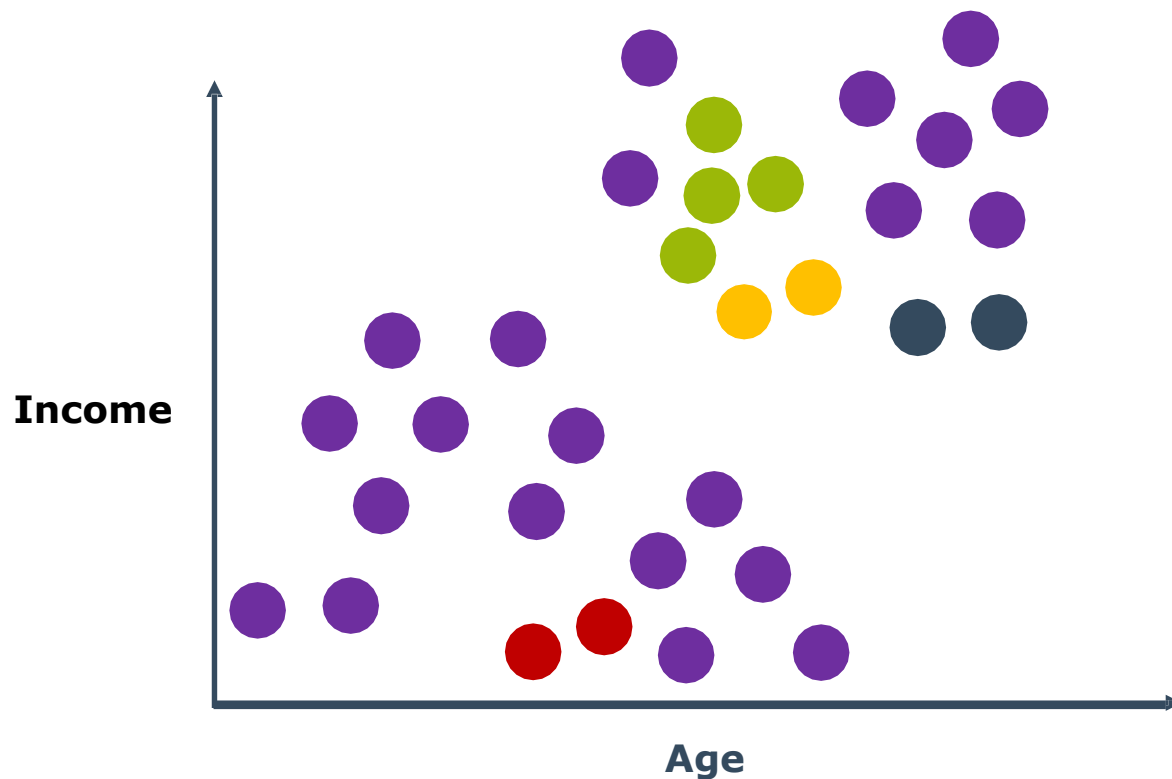
聚合式层次聚类

继续合并距离最近的点



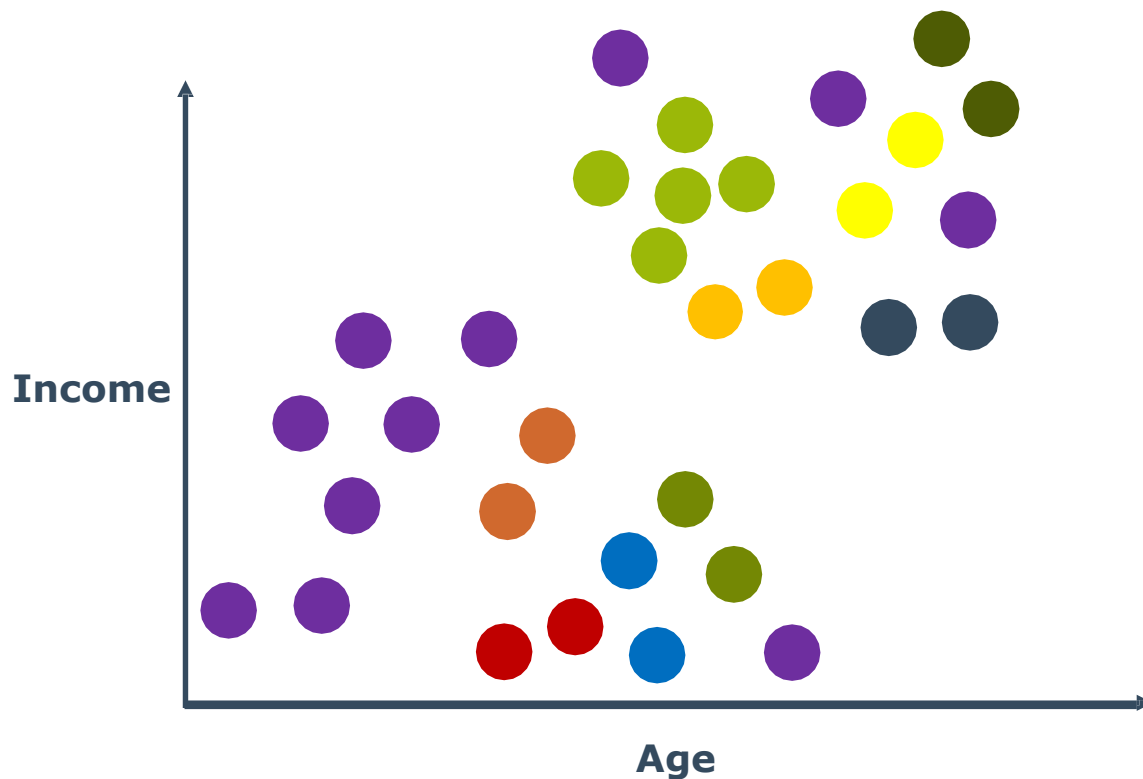
聚合式层次聚类

如果距离最近
的是两个聚簇，
则合并它们



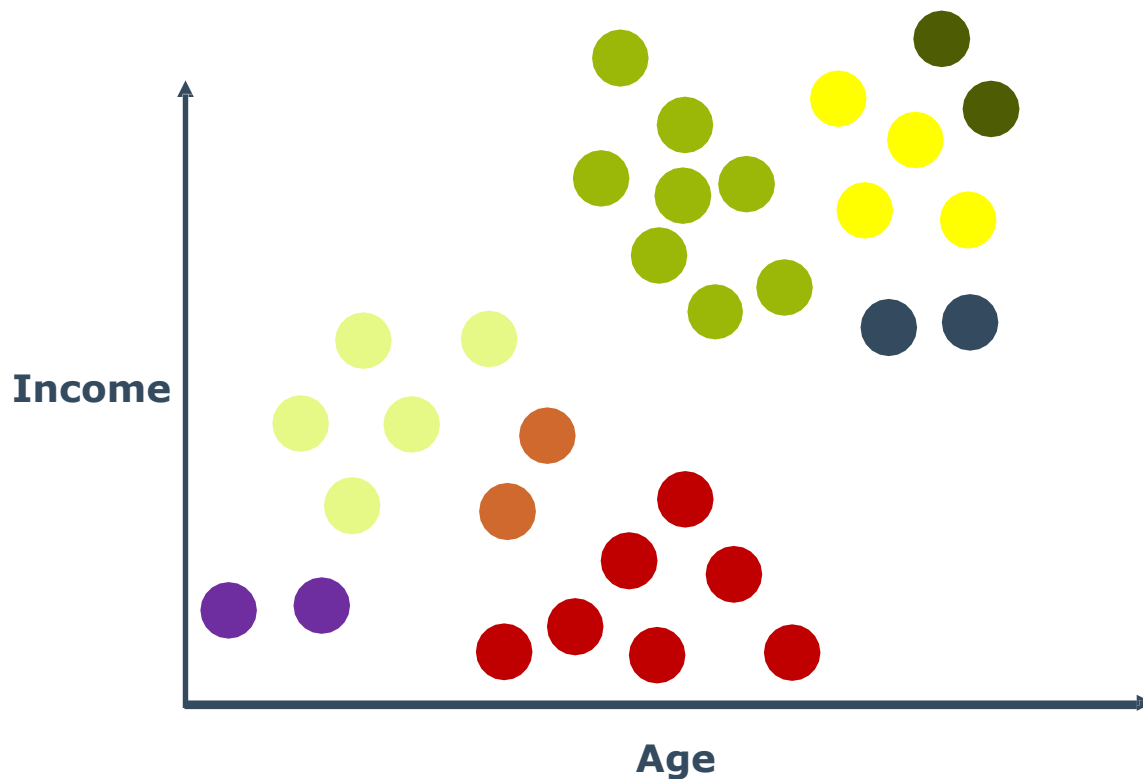
聚合式层次聚类

继续合并距离最近的
点和聚簇



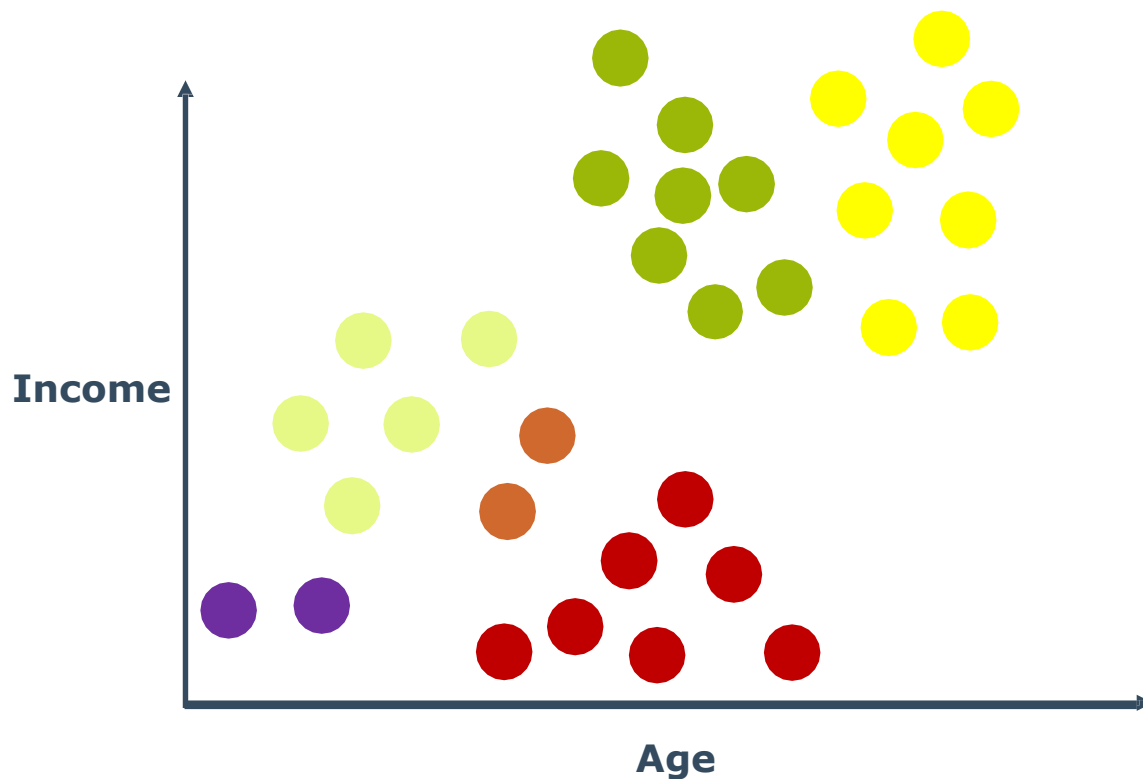
聚合式层次聚类

继续合并距离最近的
点和聚簇



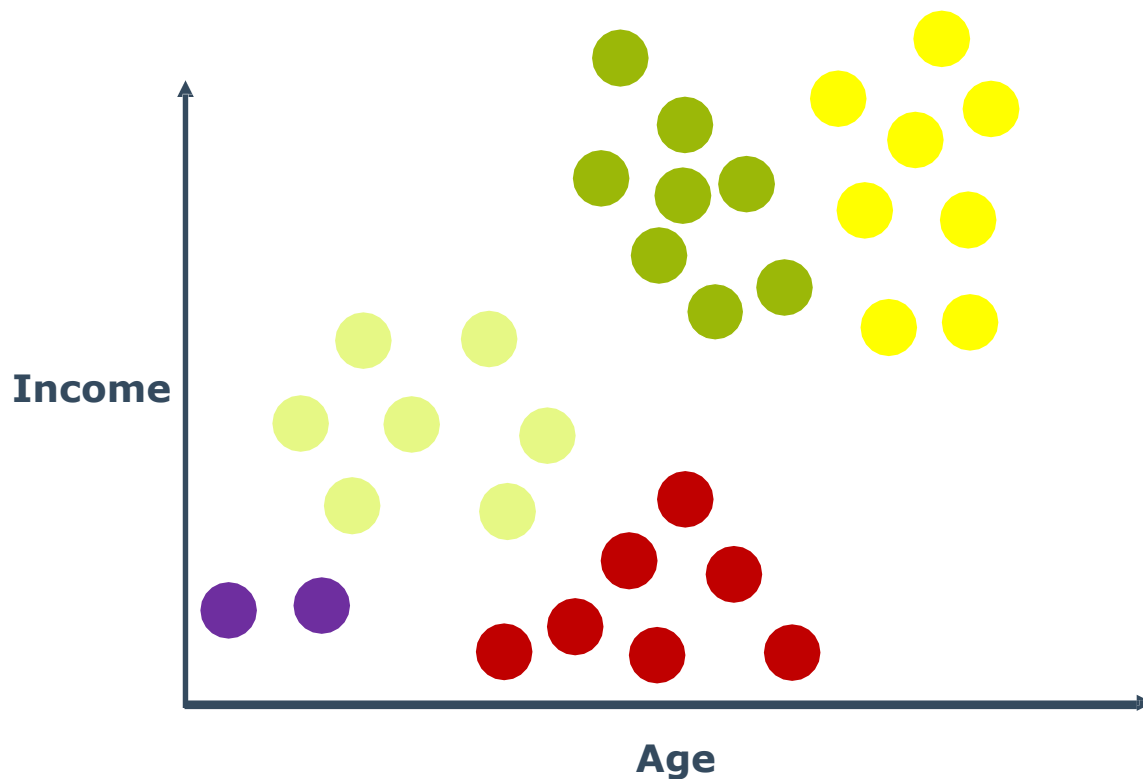
聚合式层次聚类

当前聚簇数 = 6



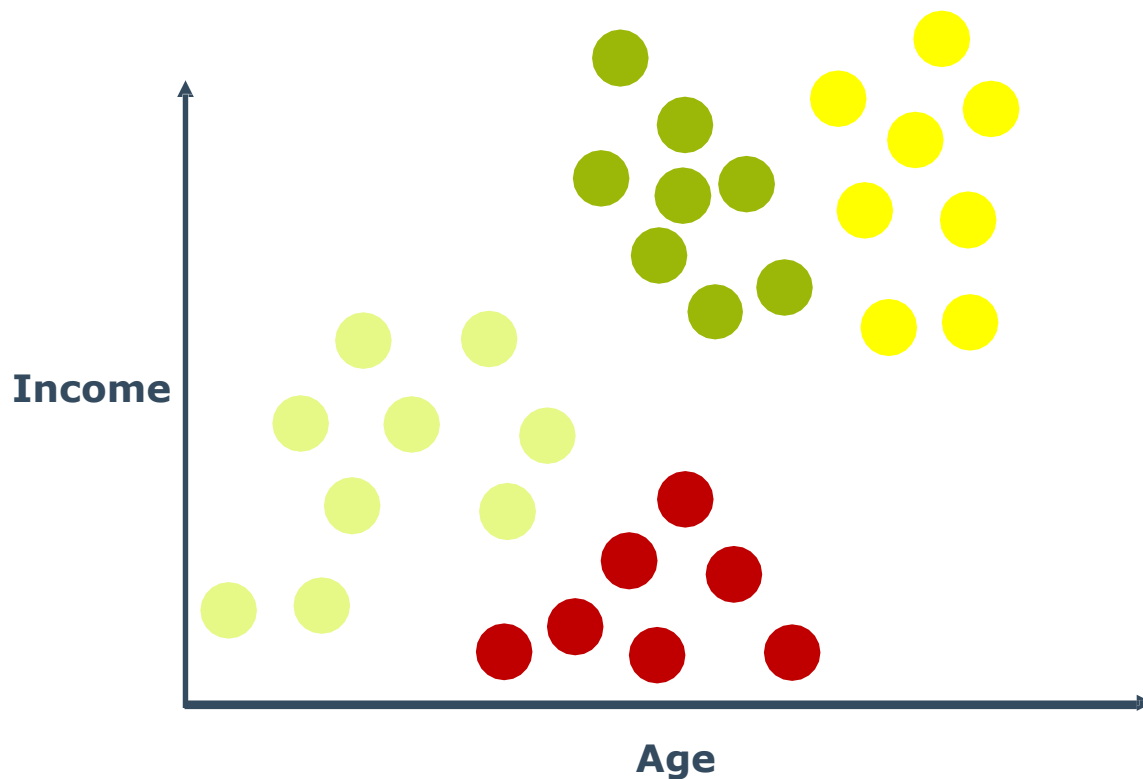
聚合式层次聚类

当前聚簇数 = 5



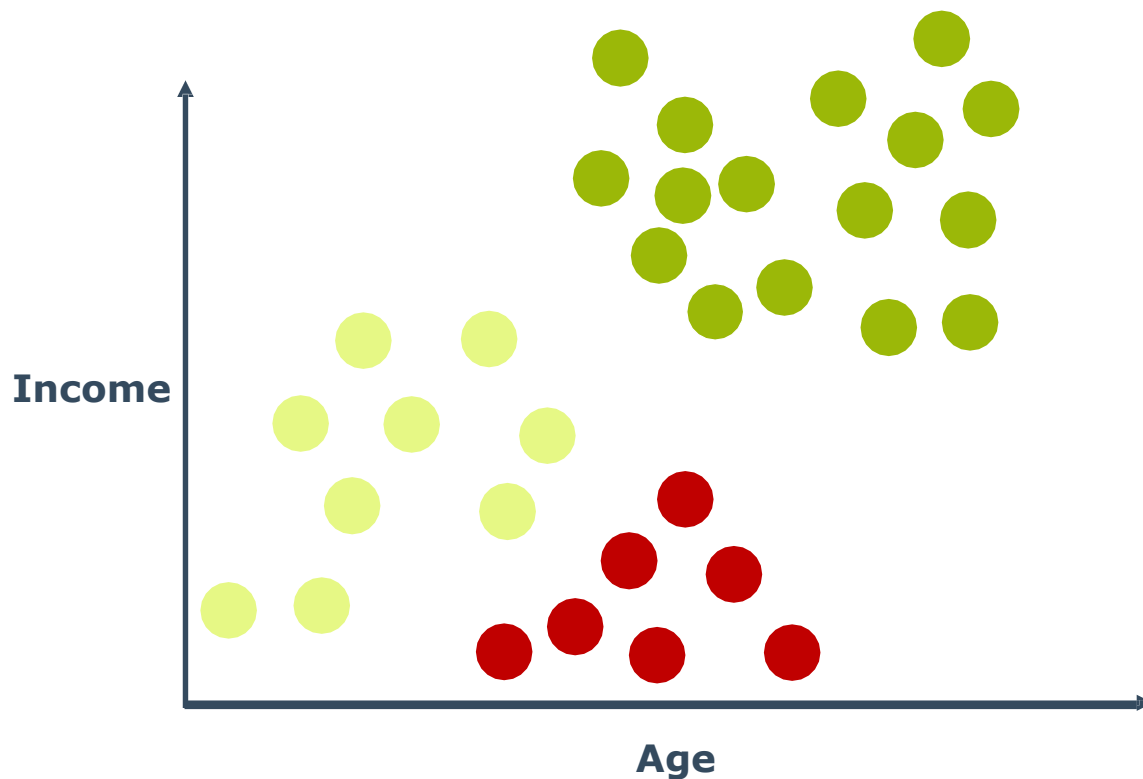
聚合式层次聚类

当前聚簇数 = 4



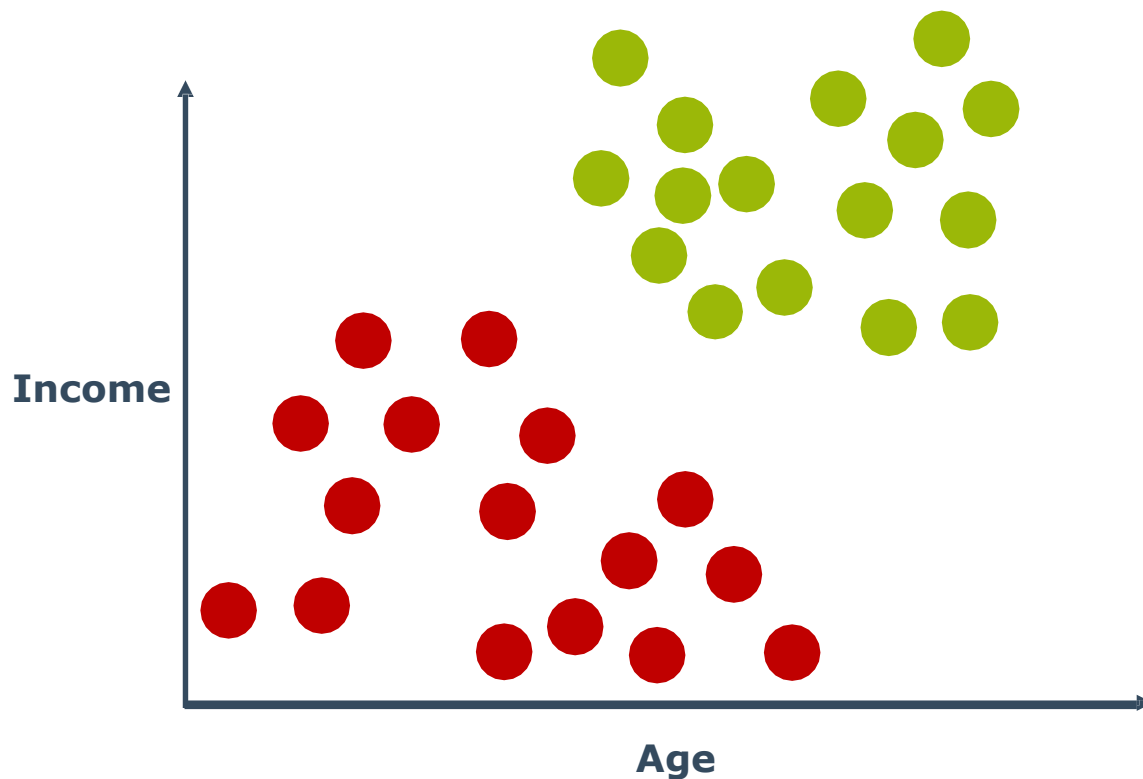
聚合式层次聚类

当前聚簇数 = 3



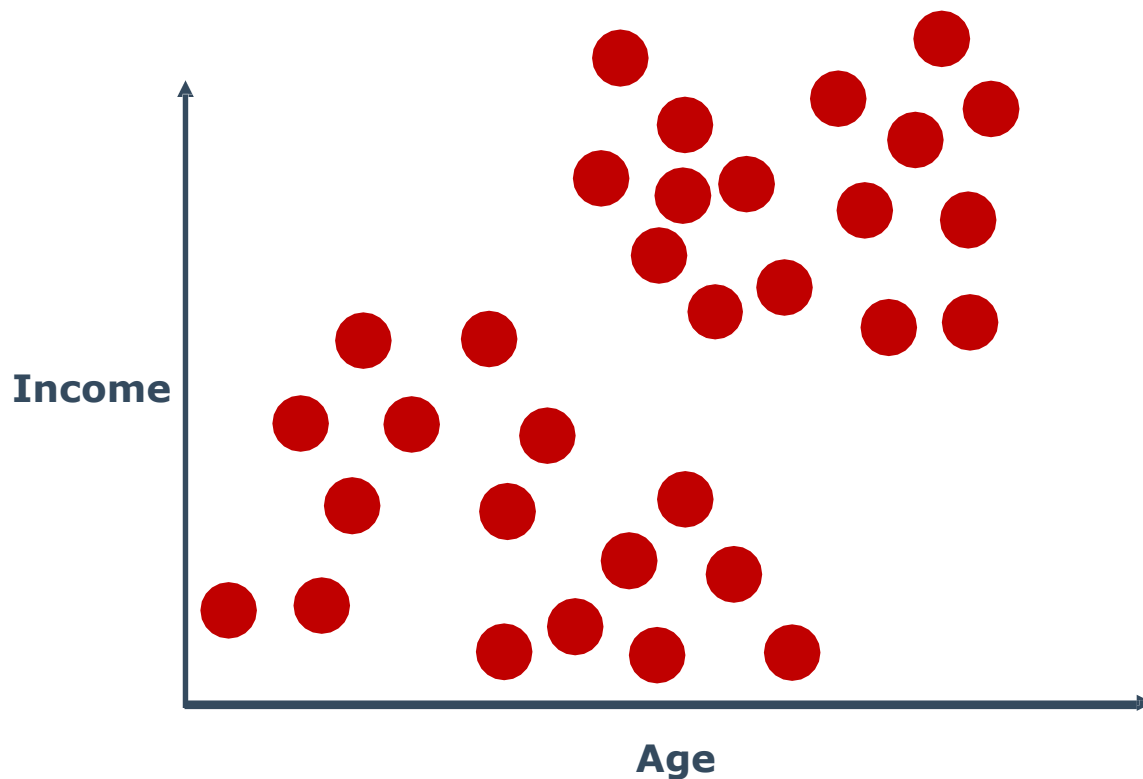
聚合式层次聚类

当前聚簇数 = 2



聚合式层次聚类

当前聚簇数 = 1



聚合式层次聚类的停止条件

条件 1

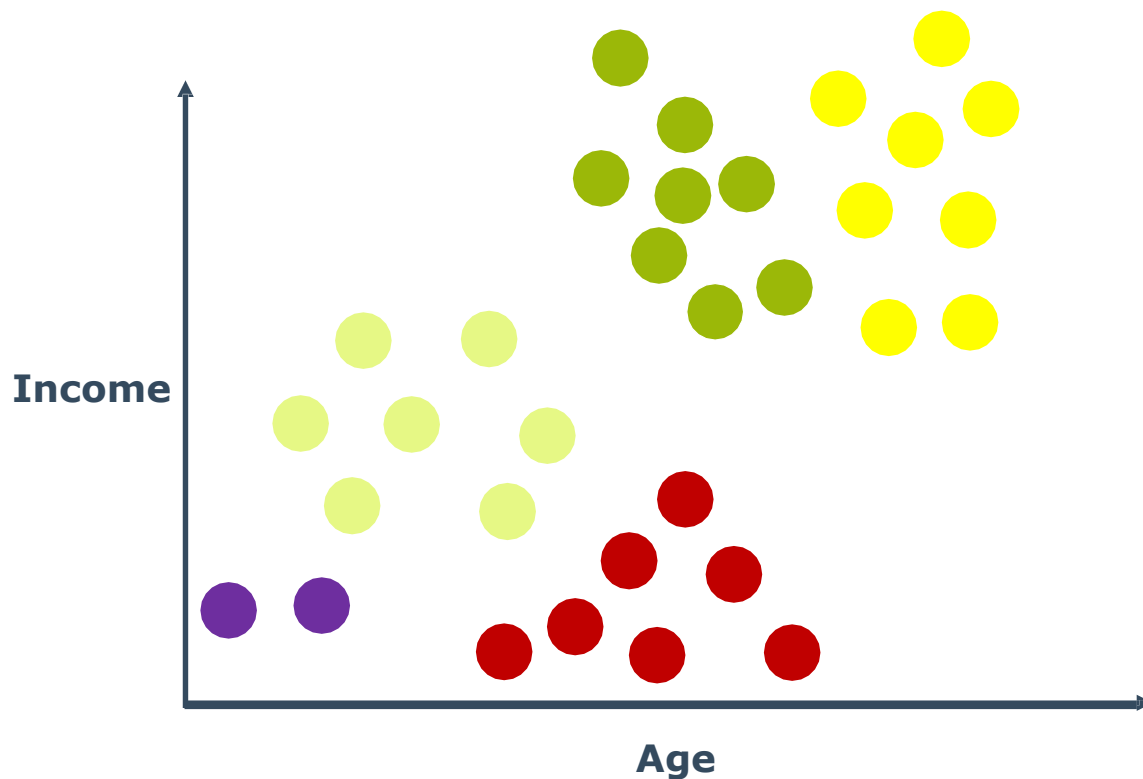
达到正确的聚类数

条件 2

最小平均聚簇距离达到预设的值

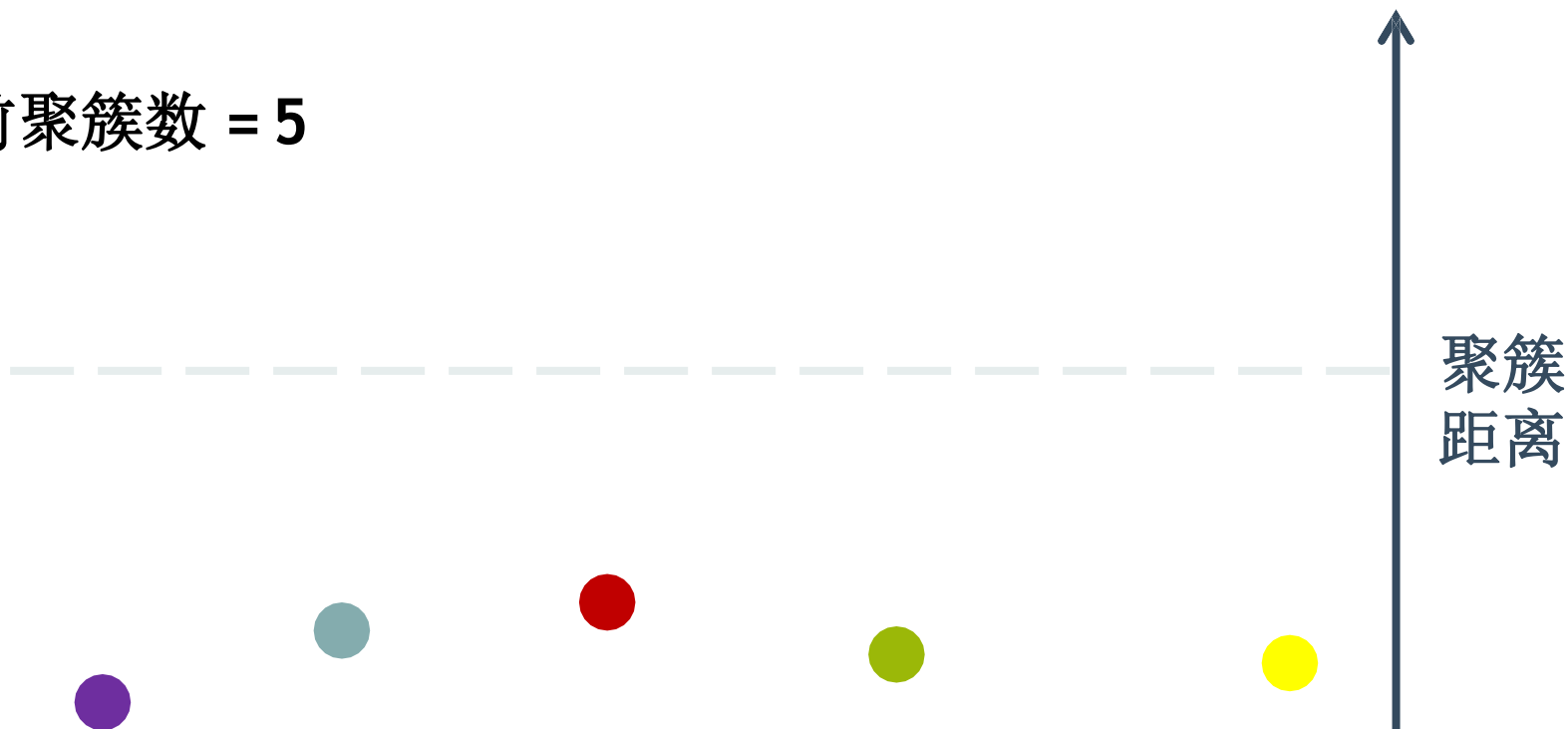
聚合式层次聚类

当前聚簇数 = 5



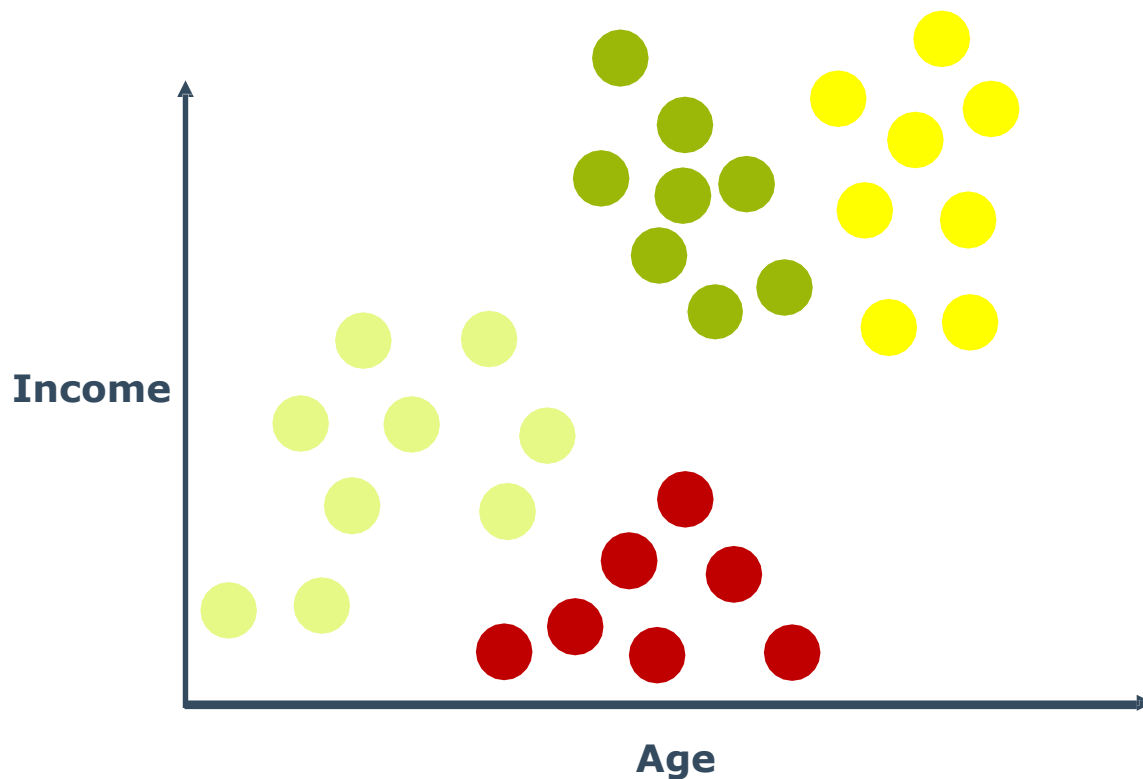
聚合式层次聚类

当前聚簇数 = 5



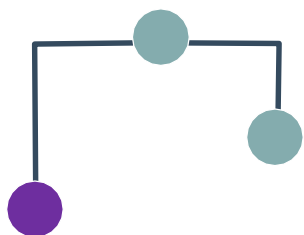
聚合式层次聚类

当前聚簇数 = 4



聚合式层次聚类

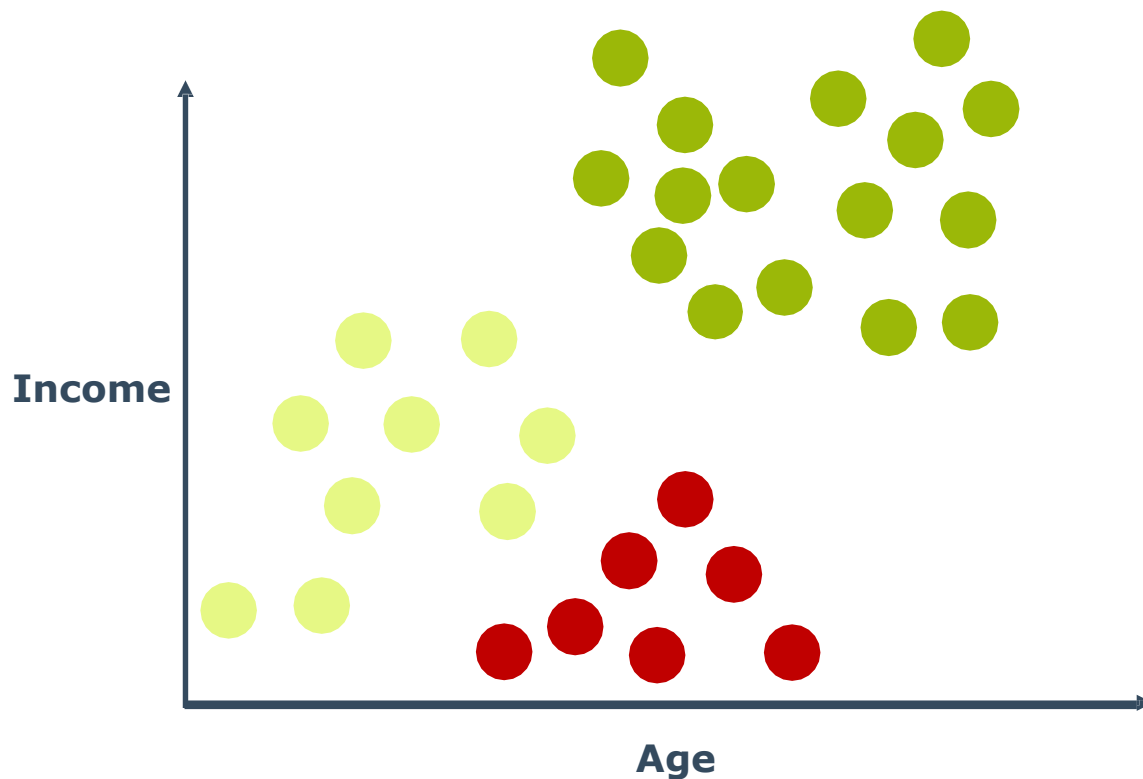
当前聚簇数 = 4



聚簇
距离

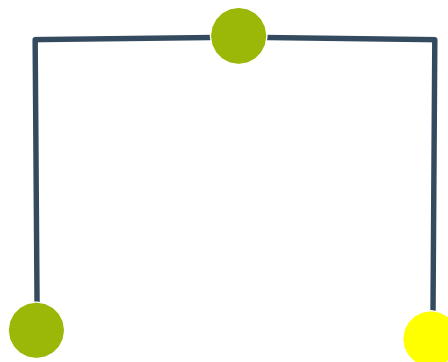
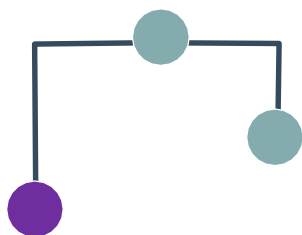
聚合式层次聚类

当前聚簇数 = 3



聚合式层次聚类

当前聚簇数 = 3

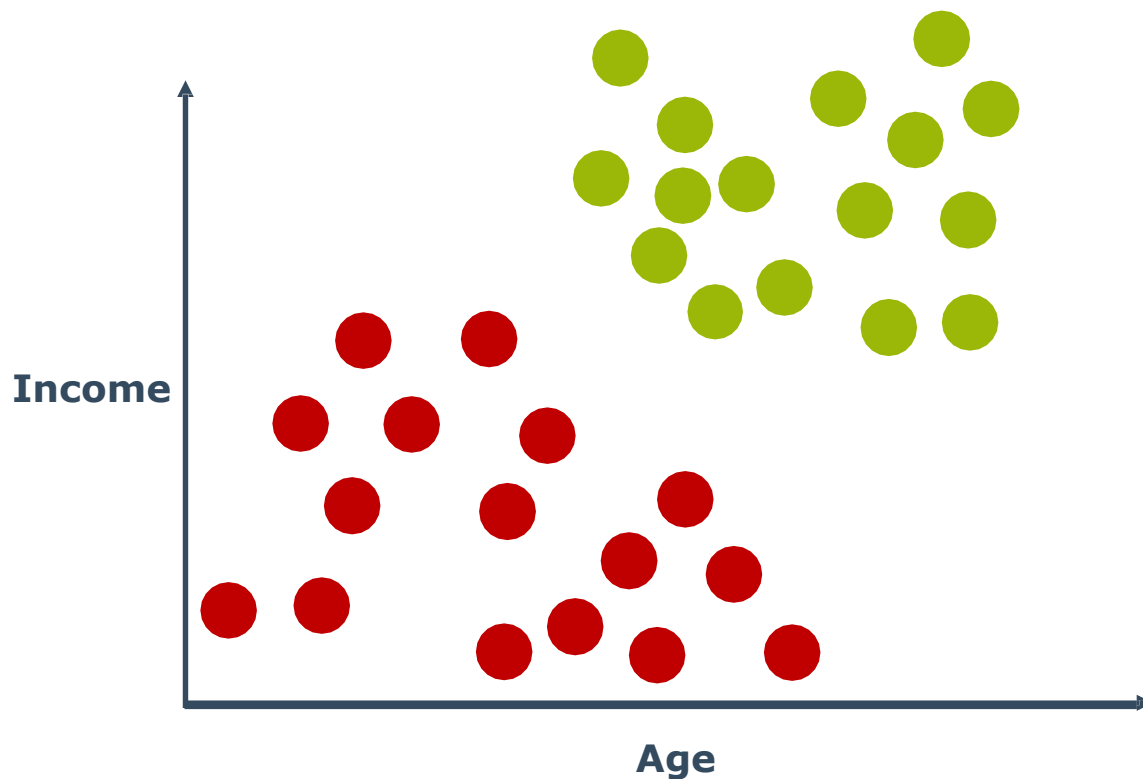


聚簇
距离



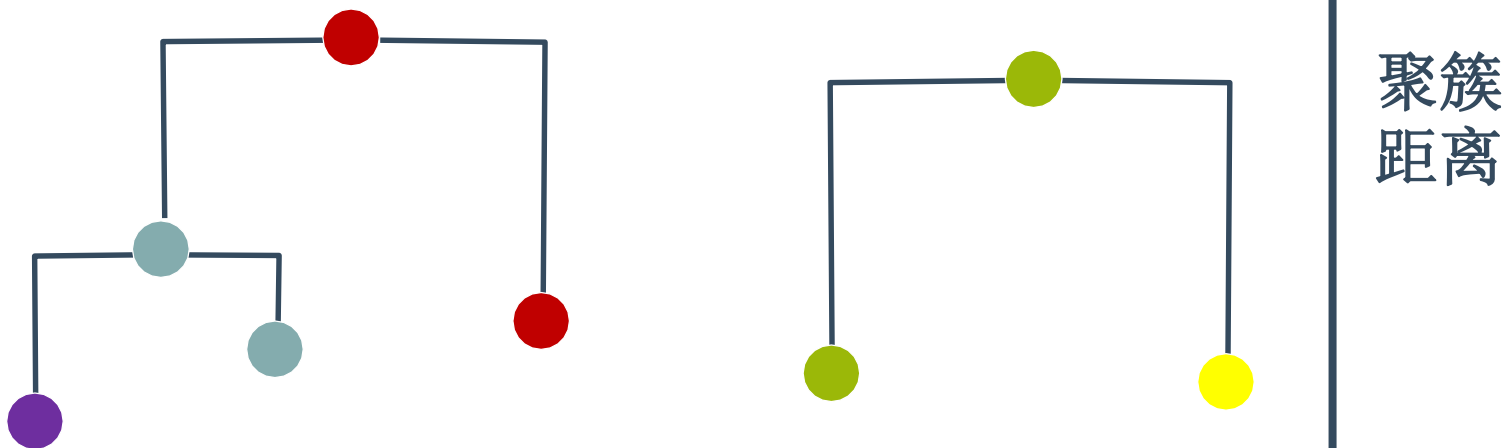
聚合式层次聚类

当前聚簇数 = 2

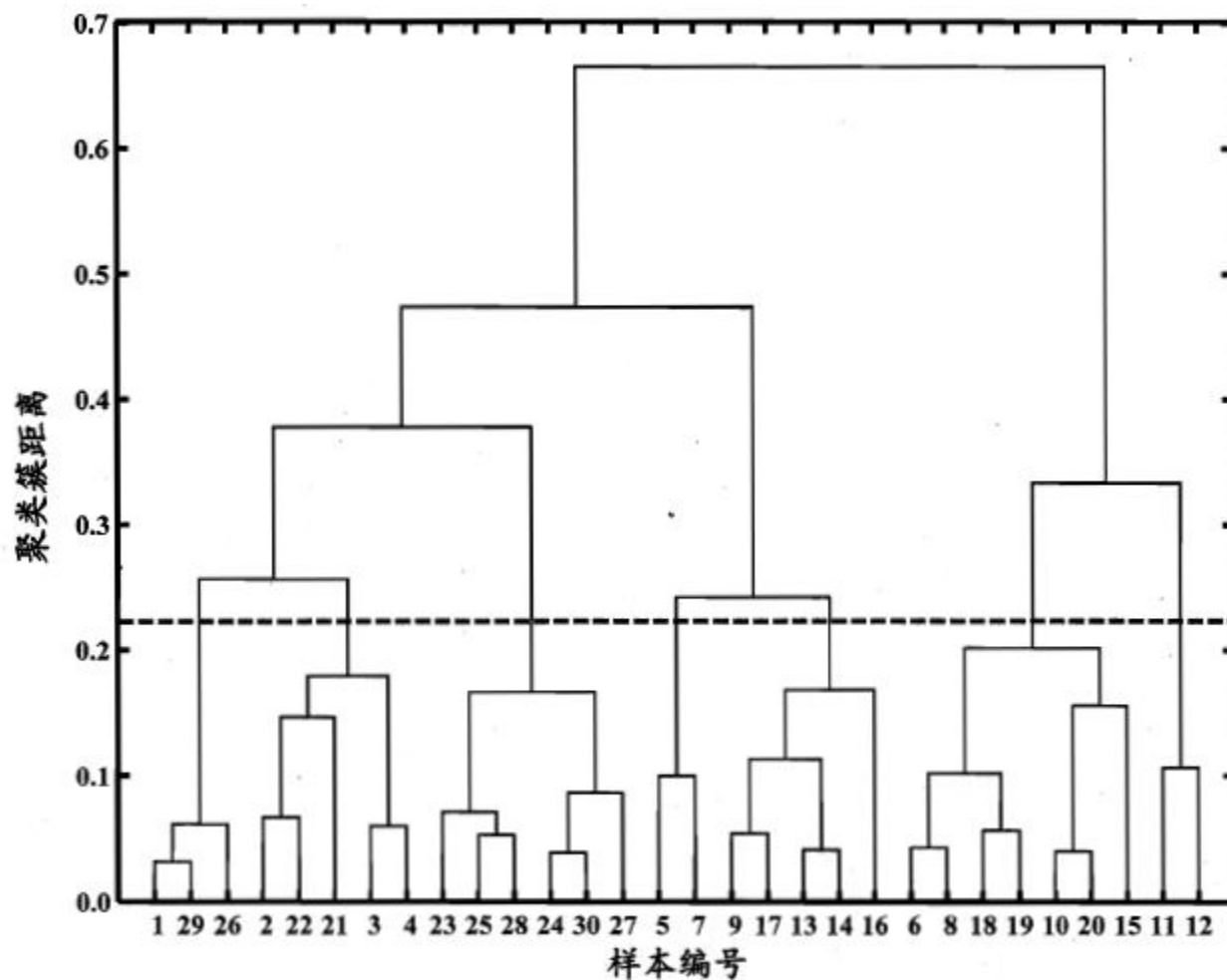


聚合式层次聚类

当前聚簇数 = 2



聚合式层次聚类



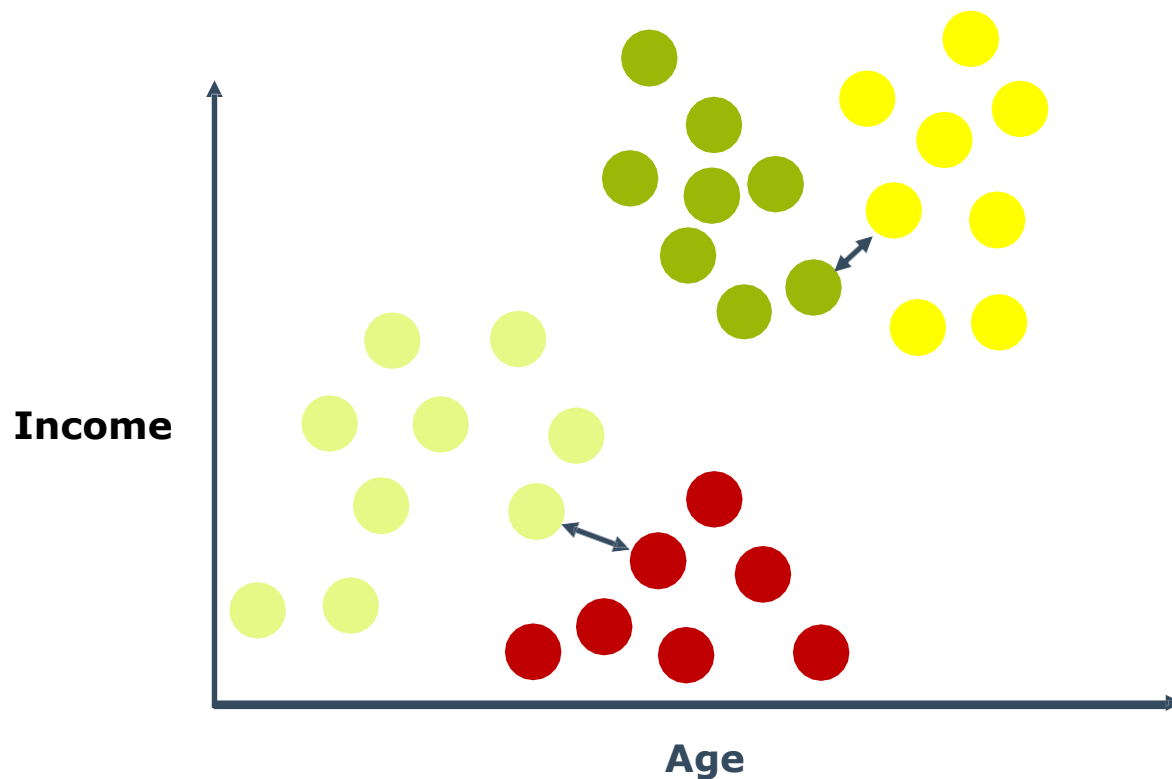
簇间距离类型

簇间距离的类型

- **单链接（Single Linkage）**：簇间最小的两点距离
- **完全链接（Complete Linkage）**：簇间最大的两点距离
- **平均链接（Average Linkage）**：簇间所有两点距离的平均值
- **Ward链接（Ward Linkage）**：每次选择导致最佳inertia的合并

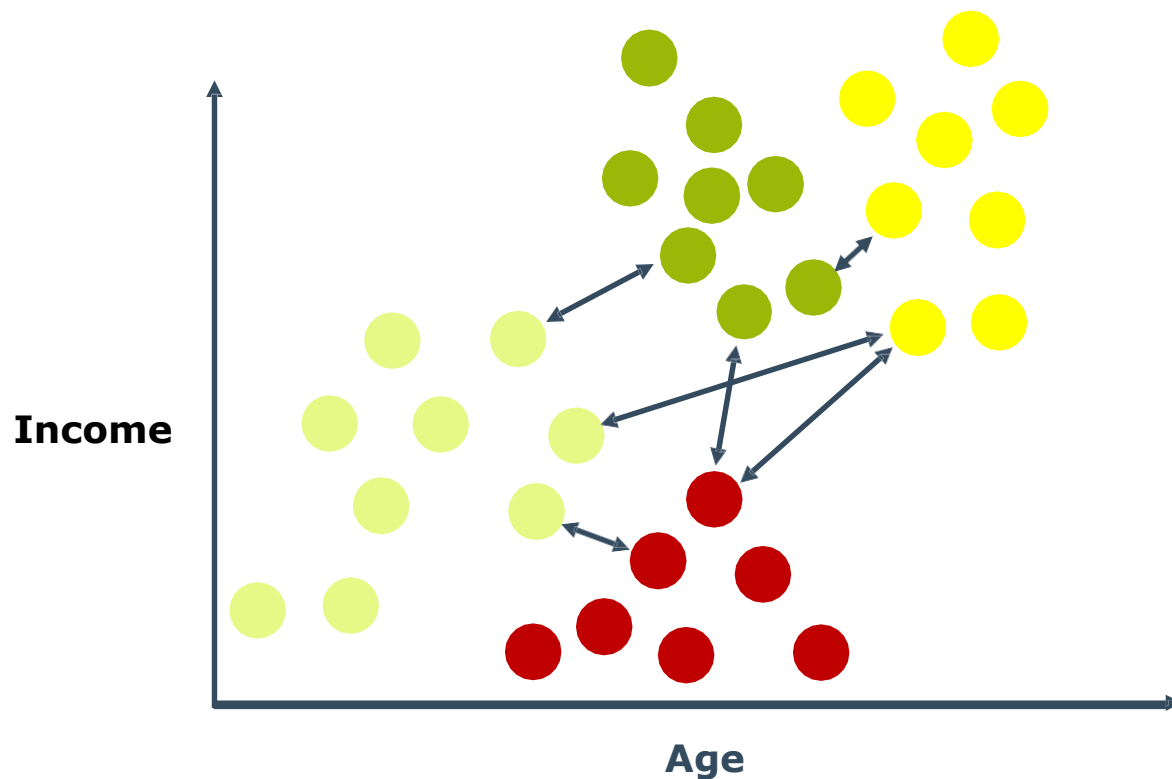
簇间距离的类型

单链接：簇间最小的
两点距离



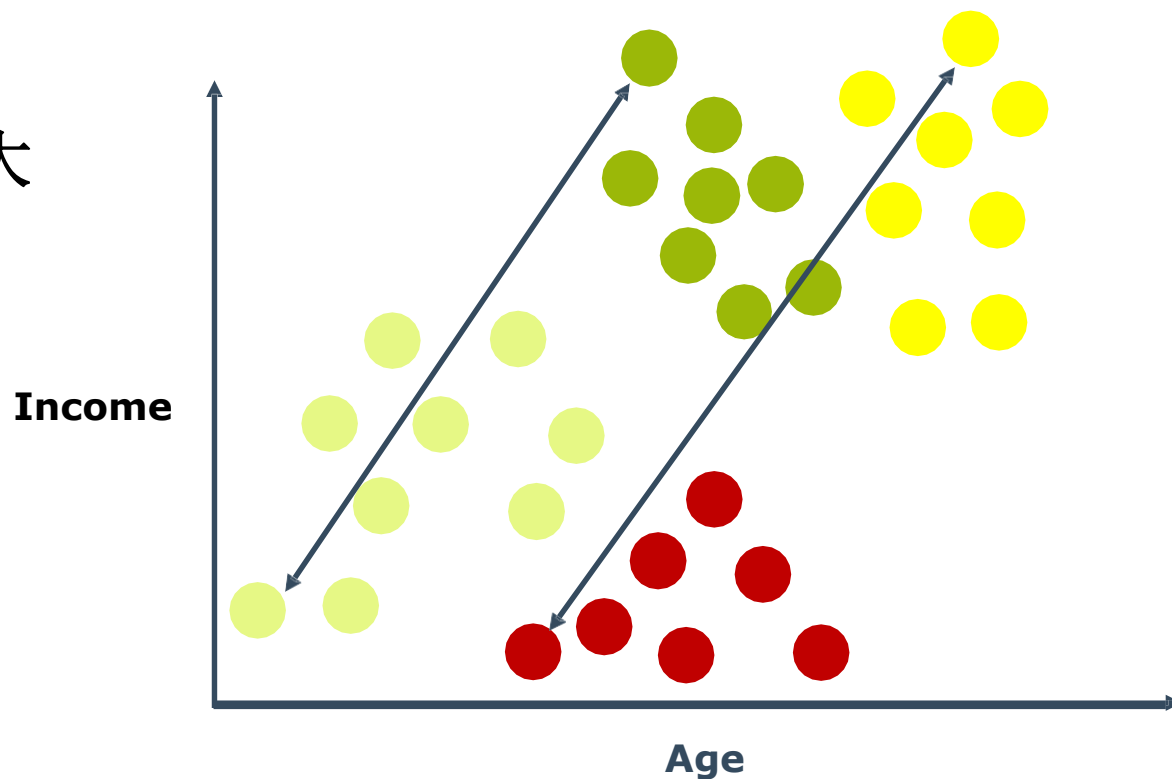
簇间距离的类型

单链接：簇间最小的
两点距离



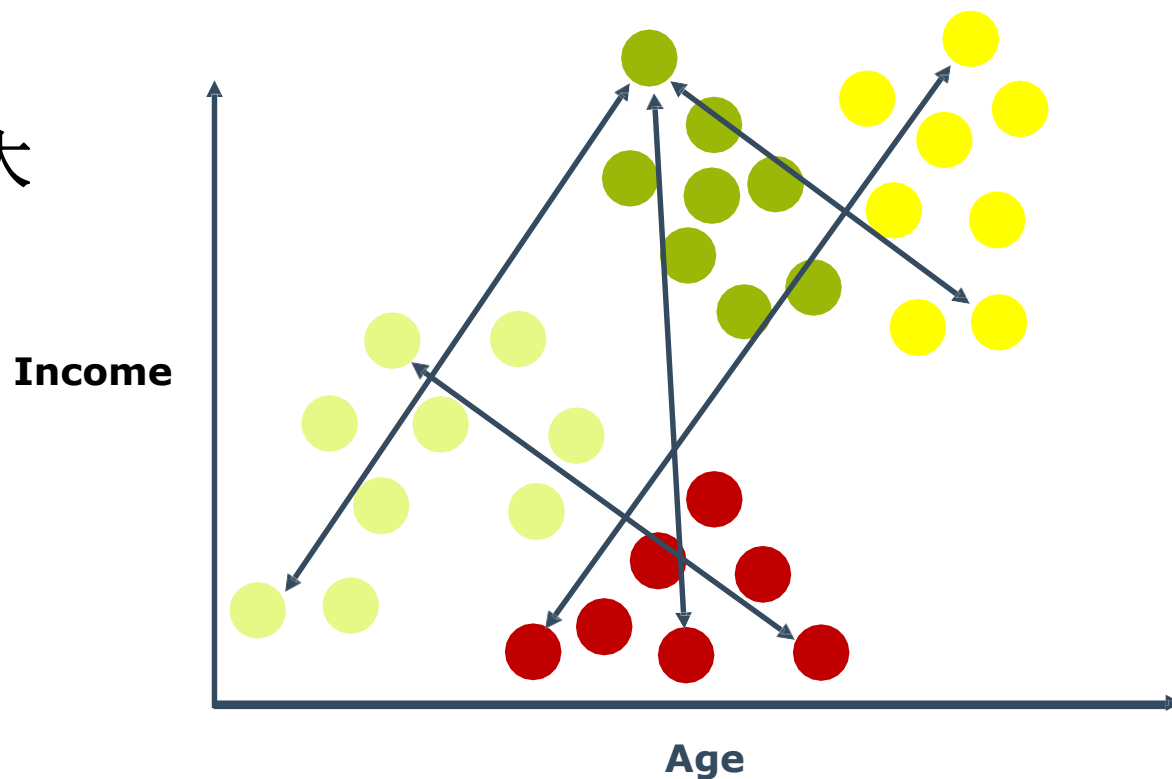
簇间距离的类型

完全链接：簇间最大的
的两点距离



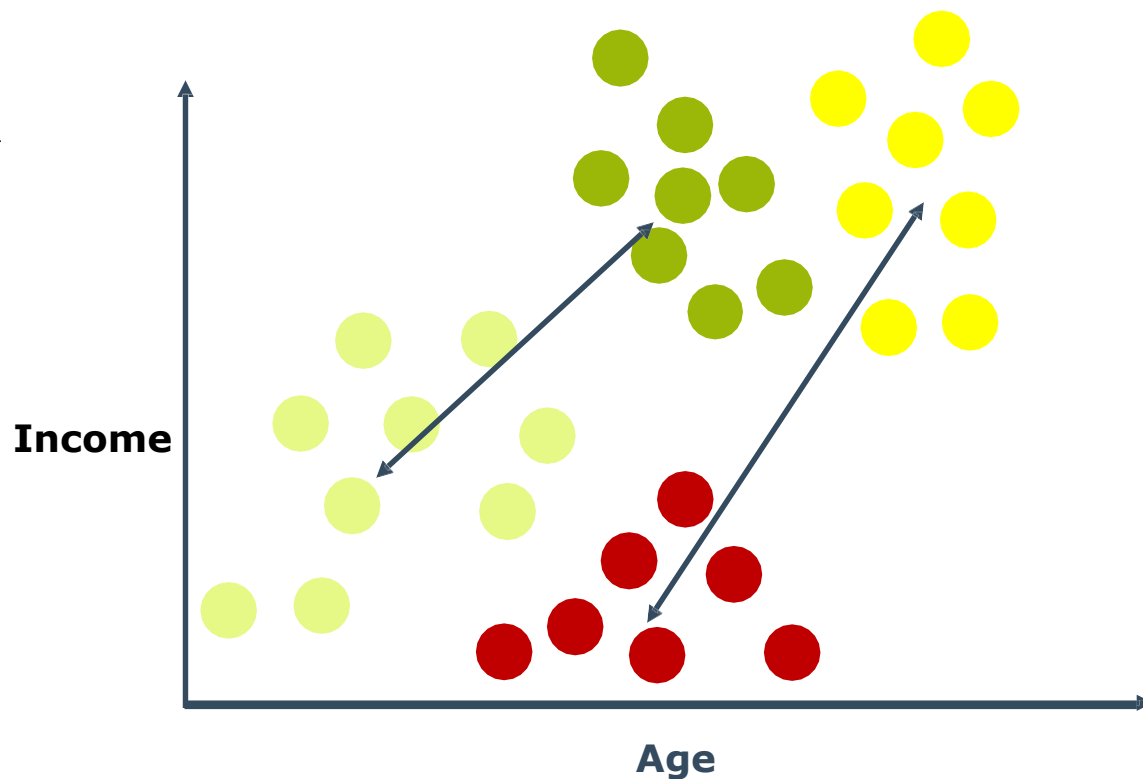
簇间距离的类型

完全链接：簇间最大的
的两点距离



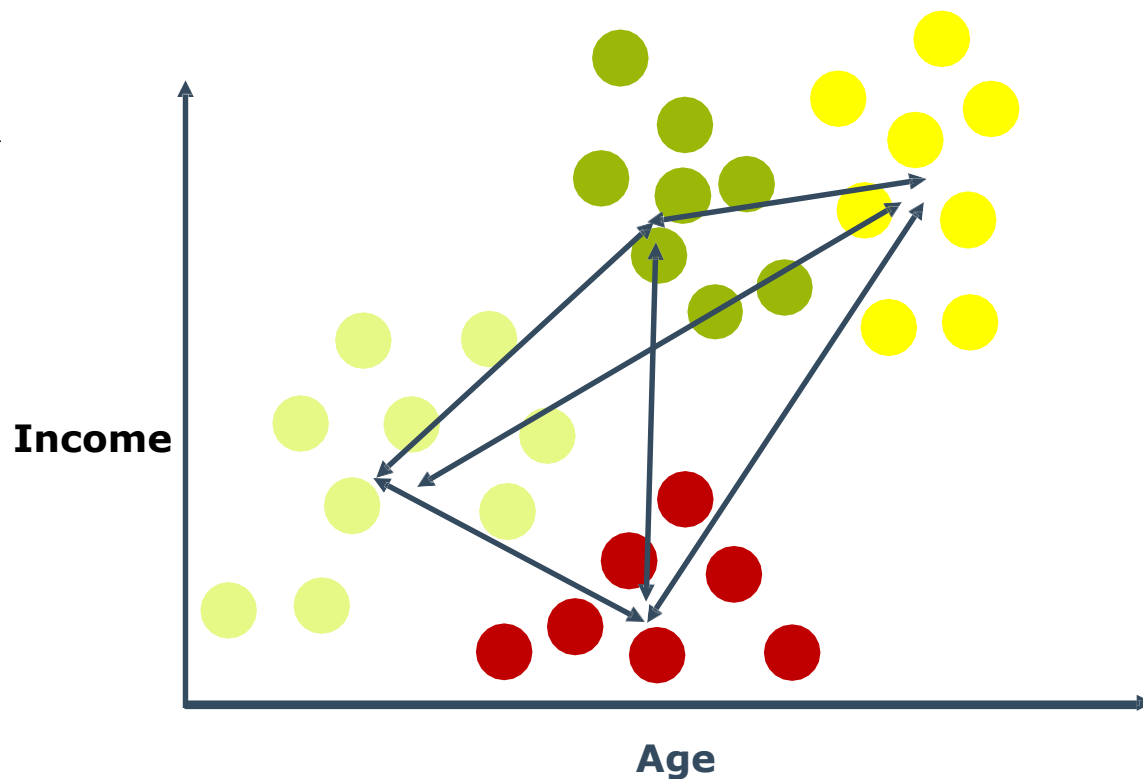
簇间距离的类型

平均链接：簇间所有
两点距离的平均值



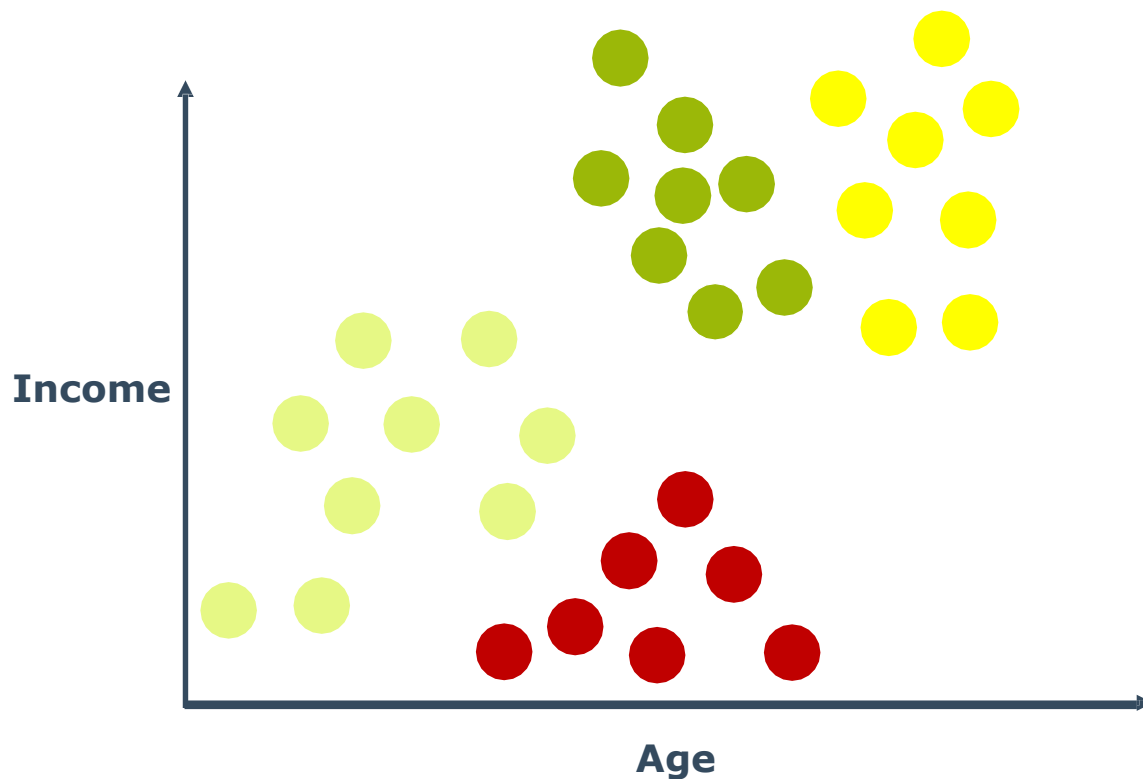
簇间距离的类型

平均链接：簇间所有
两点距离的平均值



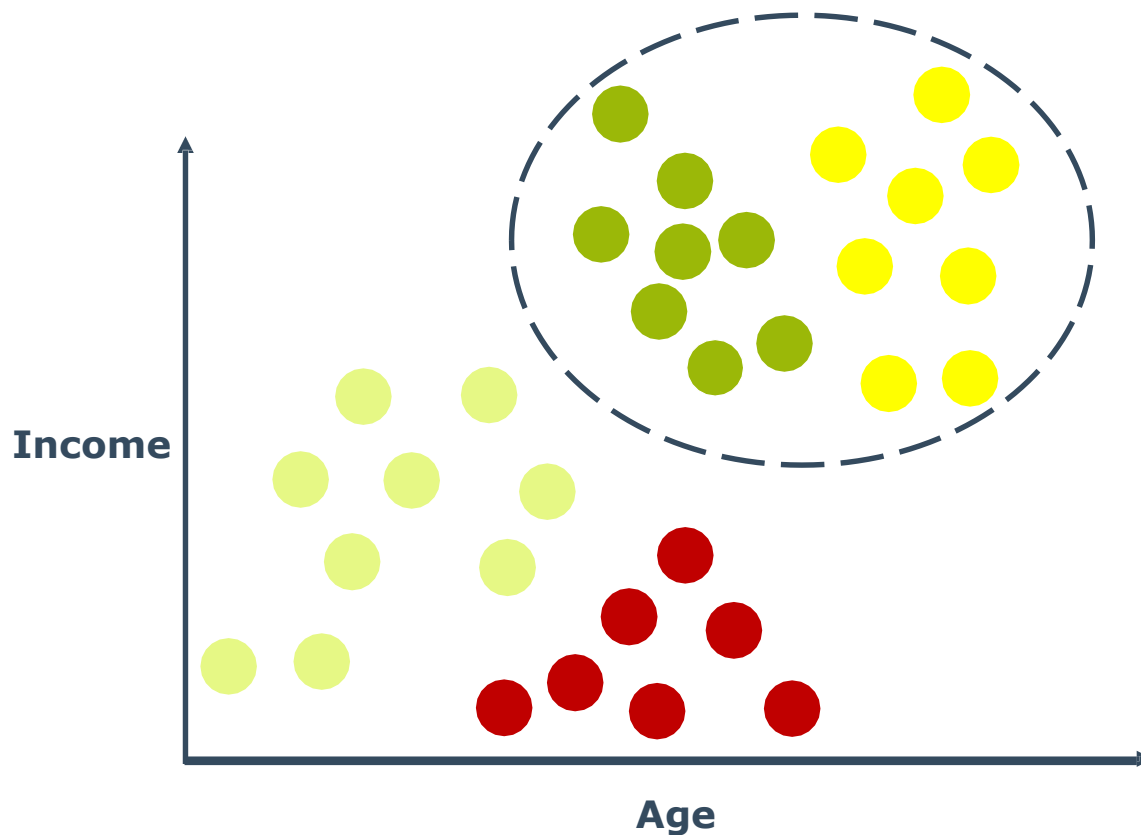
簇间距离的类型

Ward链接：每次选择导致最佳inertia的合并



簇间距离的类型

Ward链接：每次选择导致最佳inertia的合并



聚合式层次聚类的语法

导入包含聚类方法的类:

```
from sklearn.cluster import AgglomerativeClustering
```

创建该类的一个对象:

```
agg = AgglomerativeClustering(n_clusters=3,  
                               affinity='euclidean',  
                               linkage='ward')
```

拟合数据，并预测新数据的聚簇:

```
agg = agg.fit(X1)  
y_predict = agg.predict(X2)
```


<http://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html>

聚合式层次聚类的语法

导入包含聚类方法的类：

```
from sklearn.cluster import AgglomerativeClustering
```

创建该类的一个对象：

```
agg = AgglomerativeClustering(n_clusters=3,  最终的聚簇数  
                               affinity='euclidean',  
                               linkage='ward')
```

拟合数据，并预测新数据的聚簇：

```
agg = agg.fit(X1)  
y_predict = agg.predict(X2)
```


聚合式层次聚类的语法

导入包含聚类方法的类:

```
from sklearn.cluster import AgglomerativeClustering
```

创建该类的一个对象:

```
agg = AgglomerativeClustering(n_clusters=3,  
                               affinity='euclidean',  
                               linkage='ward')
```

聚簇间距离
的计算和聚
合方法



拟合数据，并预测新数据的聚簇:

```
agg = agg.fit(X1)  
y_predict = agg.predict(X2)
```