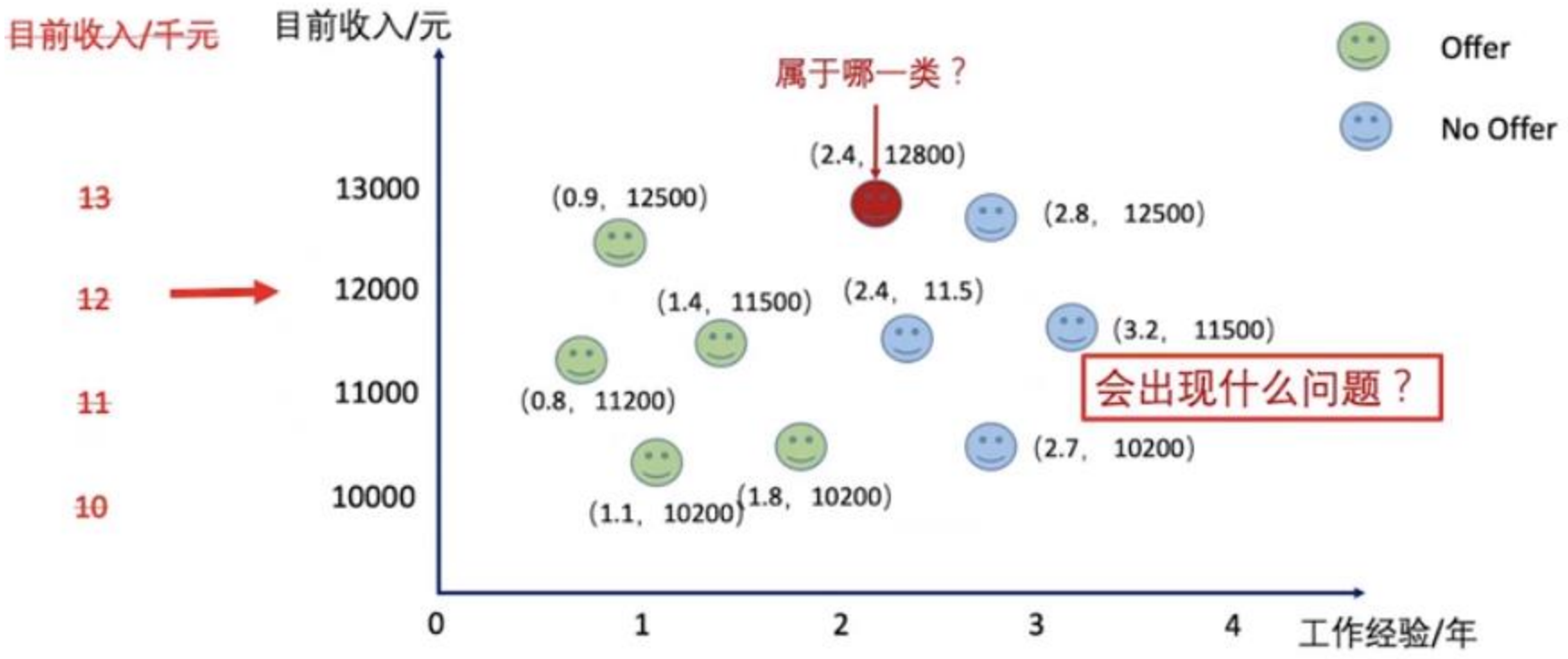
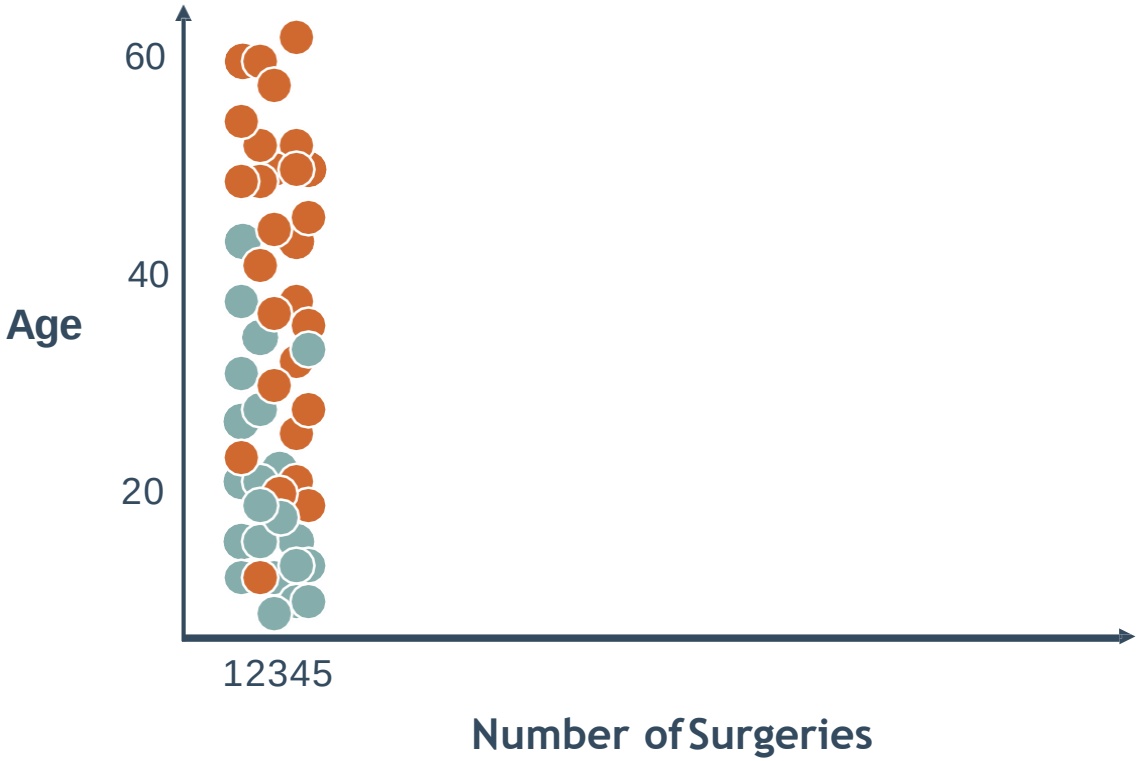
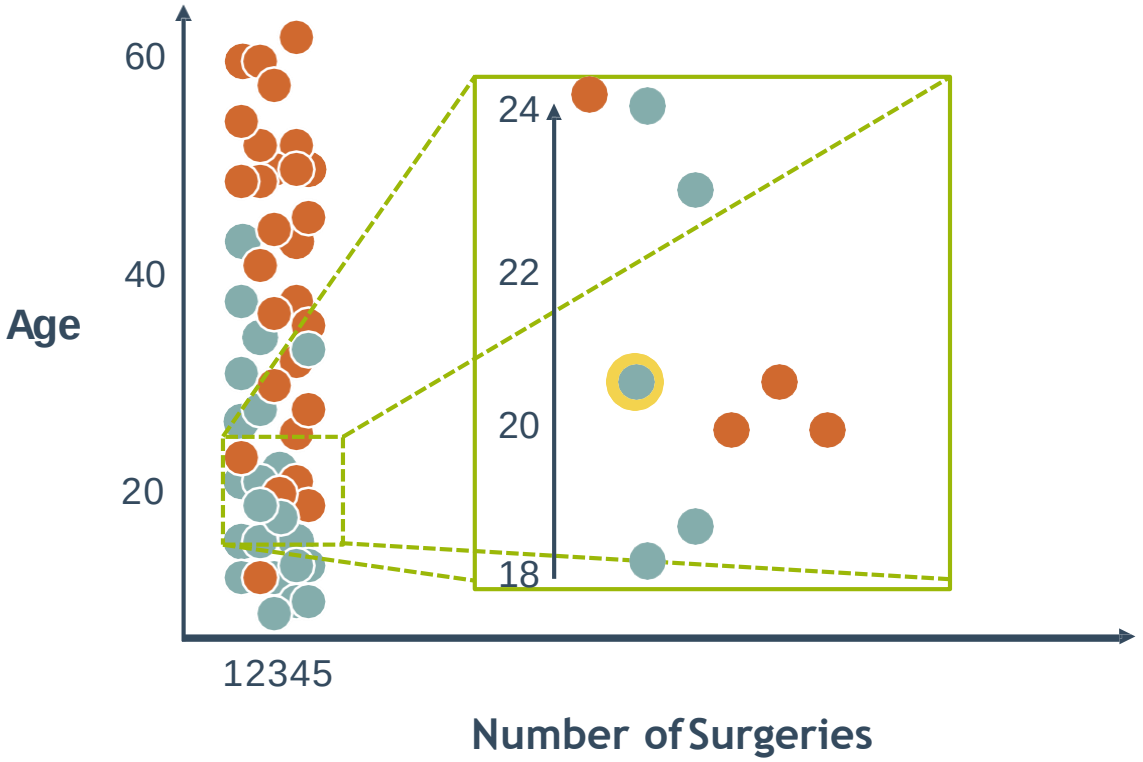
数据缩放


目前收入/千元
目前收入/元
13
12
11
10
13000
12000
11000
10000
属于哪一类？
(2.4，12800)
(0.9，12500)
(2.8，12500)
(1.4，11500)
(2.4，11.5)
(3.2，11500)
(0.8，11200)
会出现什么问题？
(2.7，10200)
(1.1，10200)
(1.8，10200)
Offer
No Offer
0
1
2
3
4
工作经验/年

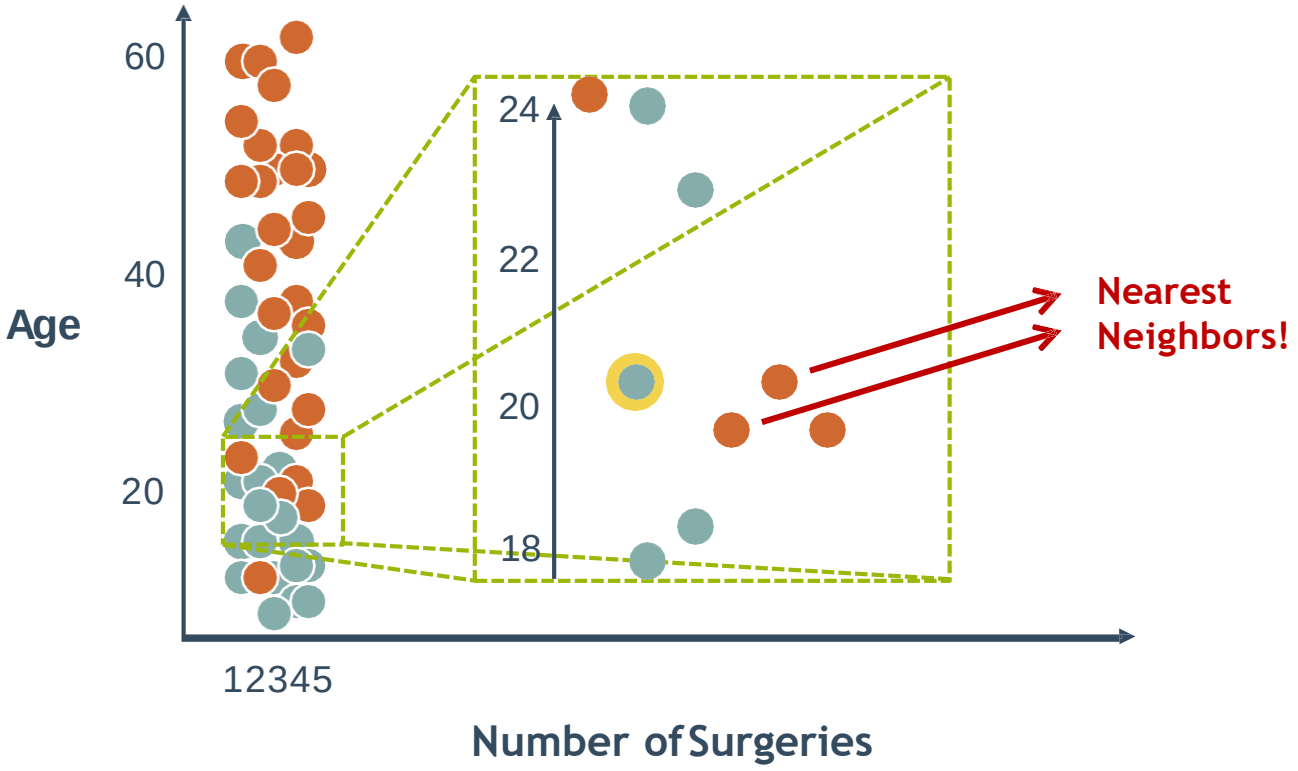
缩放比例对计算距离非常重要



缩放比例对计算距离非常重要

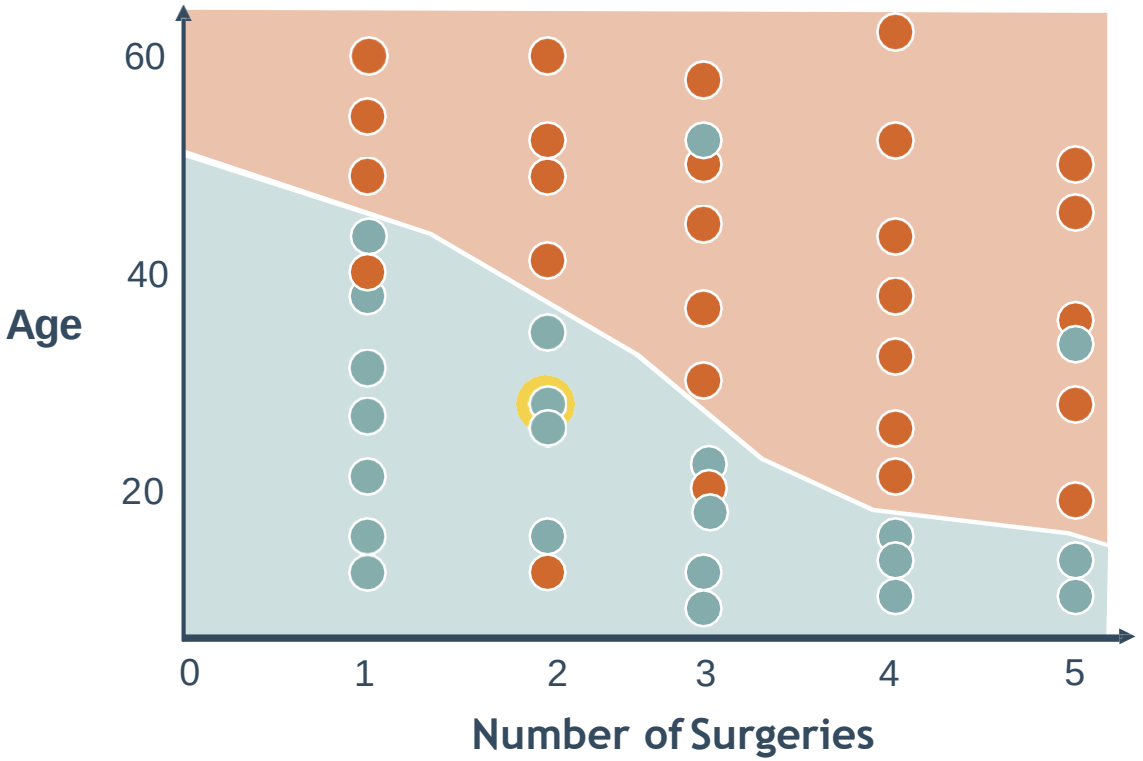


缩放比例对计算距离非常重要



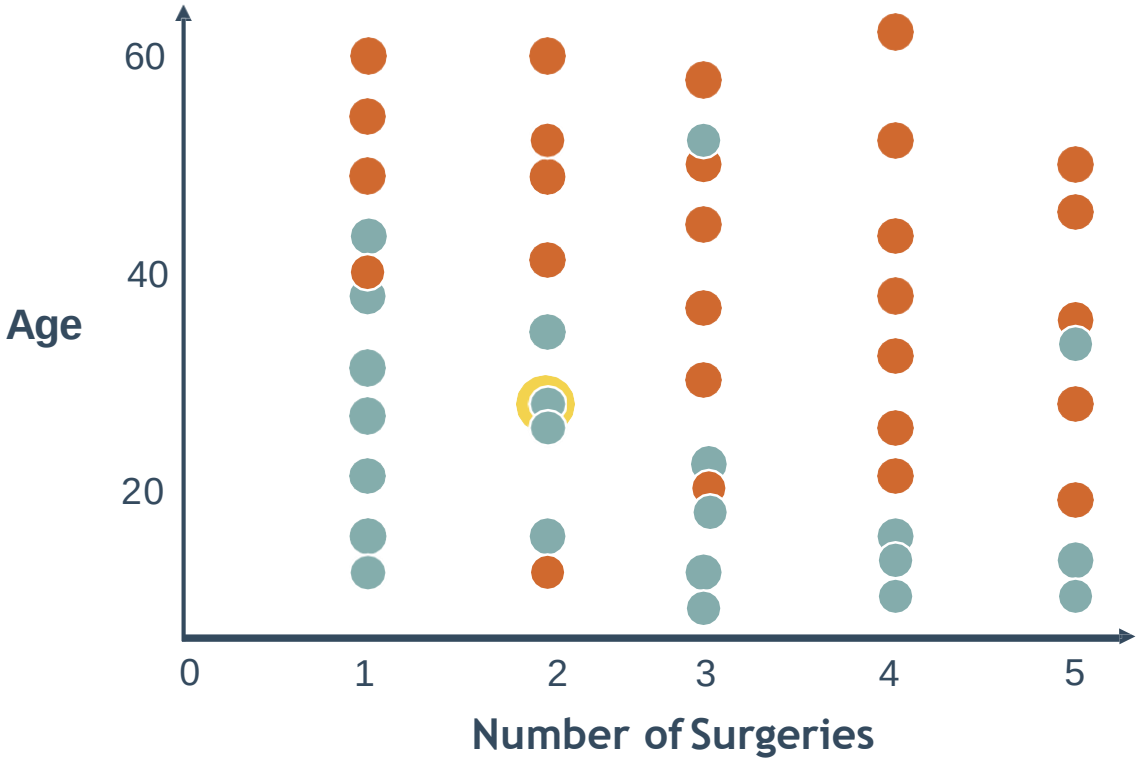
缩放比例对计算距离非常重要

"特征缩放"



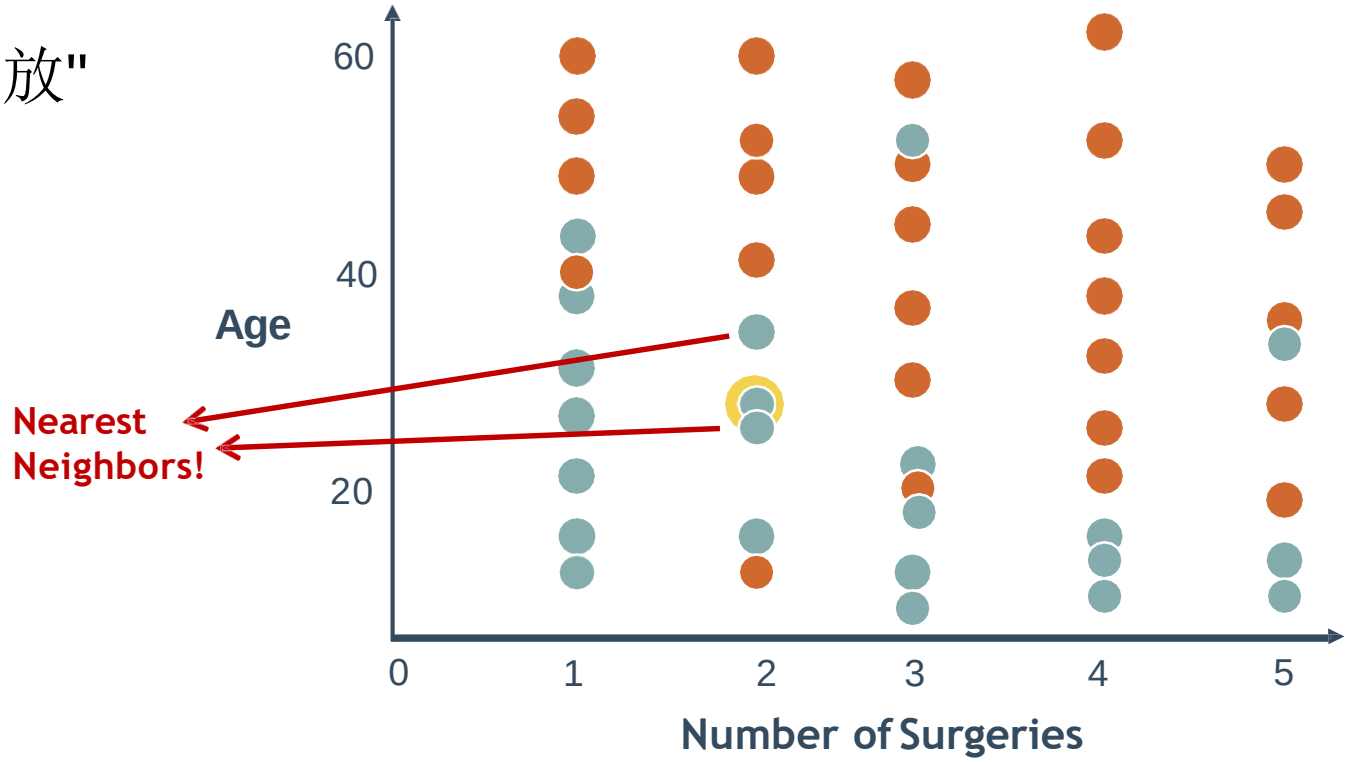
缩放比例对计算距离非常重要

"特征缩放"



缩放比例对计算距离非常重要

"特征缩放"



特征缩放的不同方法

- **Minimum-Maximum Scaler**：将数据缩放到某一给定范围（通常是[0, 1]）。
- **Standard Scaler**：即标准化，尽量将数据转化为均值为0，方差为1的数据，形如标准正态分布（高斯分布）。
- **Maximum Absolute Value Scaler**：通过除以最大绝对值，将数据缩放到[-1, 1]。

- 使各特征的数值都处于同一数量级上。

特征缩放—

线性归一化(**Min-Max Normalization)**

特征
1.5
2
3
4
1.5
2.5
2
2.3
3
1.5

数值范围 = [1.5，4]

特征
0
0.2
0.6
1.0
0
0.4
0.2
0.33
0.6
0

数值范围 = [0，1]

$$X_{new} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

X的最小值

X的最大值

特征缩放—

标准差标准化(**Z-score Normalization**)

特征
1.5
2
3
4
1.5
2.5
2
2.3
3
1.5

→

特征
-1.0165
-0.4041
0.8205
2.0452
-1.0165
0.2082
-0.4041
-0.0367
0.8205
-1.0165

$$X_{new} = \frac{X - mean(X)}{std(X)}$$

$mean(X)$：X的平均值

$std(X)$：X的标准差

均值=2.33　标准差=0.8166

特征缩放的语法

导入包含缩放方法的类：

```
from sklearn.preprocessing import StandardScaler
```

创建该类的一个对象：

```
StdSc = StandardScaler()
```

拟合缩放的参数，然后对数据做转换：

```
StdSc = StdSc.fit(X_data)
X_scaled = StdSc.transform(X_data)
```

或者 `X_scaled = StdSc.fit_transform(X_data)`

特征缩放的语法

导入包含缩放方法的类：

```
from sklearn.preprocessing import StandardScaler
```

创建该类的一个对象：

```
StdSc = StandardScaler()
```

拟合缩放的参数，然后对数据做转换：

```
StdSc = StdSc.fit(X_data)

X_scaled = StdSc.transform(X_data)
```

其他缩放方法: MinMaxScaler, MaxAbsScaler.

K近邻算法—多分类

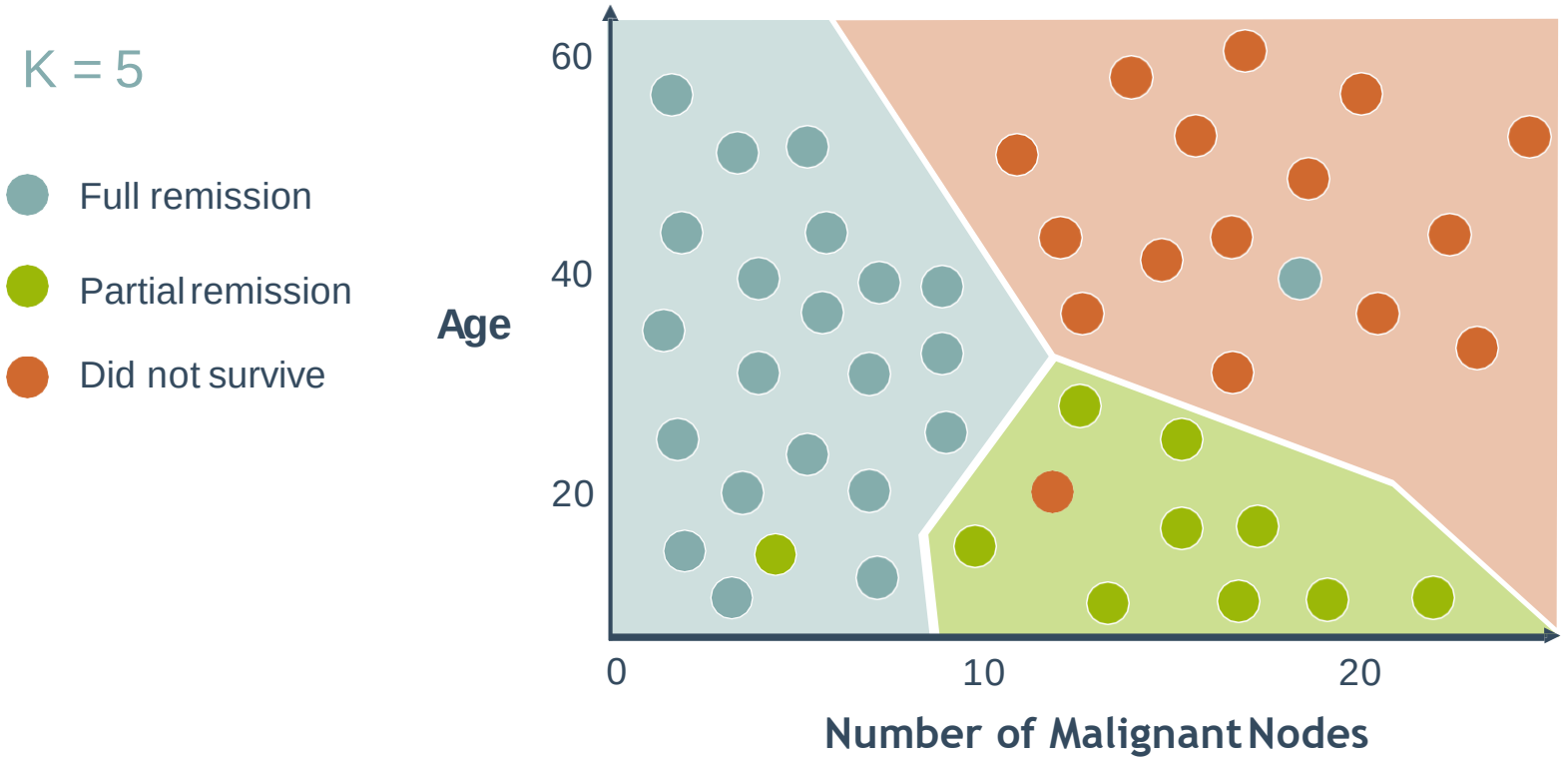
序号	名称	搞笑镜头	拥抱镜头	打斗镜头	电影类型
1	电影1	39	0	31	喜剧片
2	电影2	3	2	65	动作片
3	电影3	2	3	55	爱情片
4	电影4	9	38	2	爱情片
5	电影5	8	34	17	爱情片
6	电影6	5	2	57	动作片
7	电影7	21	17	5	喜剧片
8	电影8	45	2	9	喜剧片
9	电影9	23	3	17	?

例：

K近邻算法多分类

序号	名称	搞笑镜头	拥抱镜头	打斗镜头	电影类型	距离	K=
1	电影1	39	0	31	喜剧片	21.5	
2	电影2	3	2	65	动作片	52.0	
3	电影3	2	3	55	爱情片	43.4	
4	电影4	9	38	2	爱情片	40.6	
5	电影5	8	34	17	爱情片	34.4	
6	电影6	5	2	57	动作片	43.9	
7	电影7	21	17	5	喜剧片	18.6	
8	电影8	45	2	9	喜剧片	23.4	
9	电影9	23	3	17	？	——	

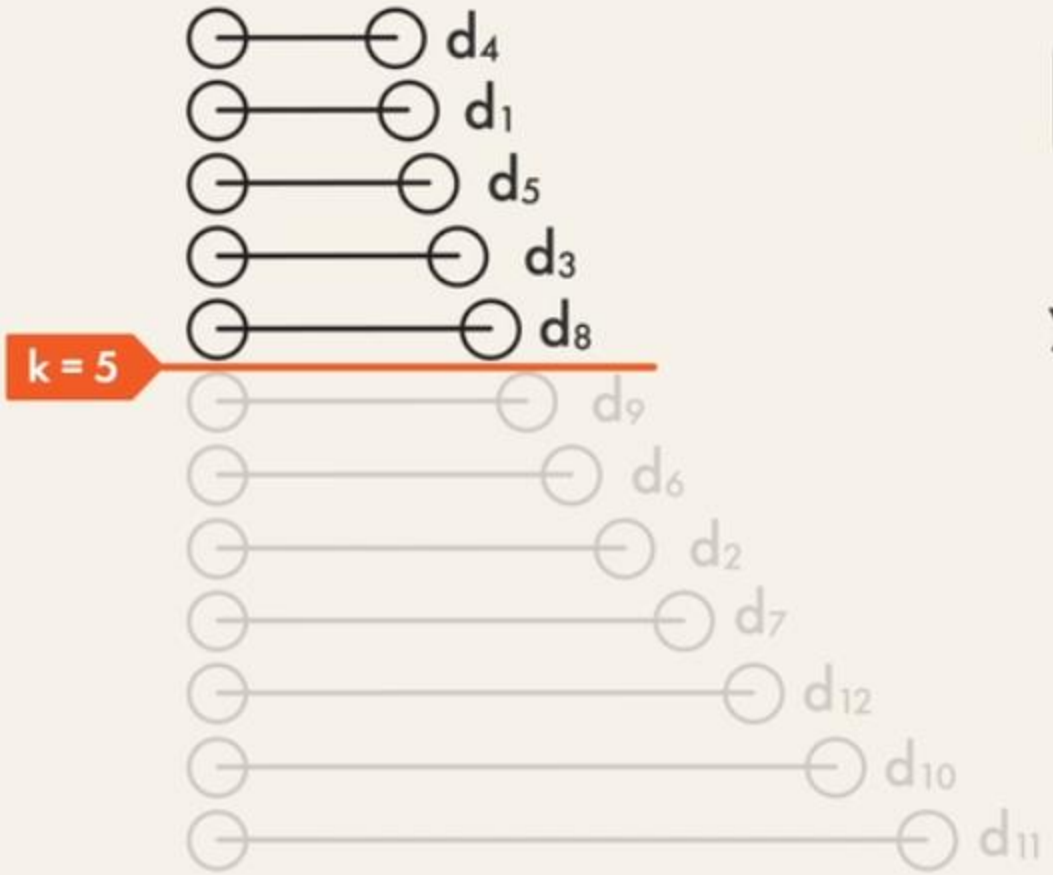
K近邻多分类的判定边界



投票决定：少数服从多数，近邻中哪个类别的点最多就分为该类。

加权投票法：根据距离的远近，对近邻的投票进行加权，距离越近则权重越大（权重为距离平方的倒数）

K近邻回归

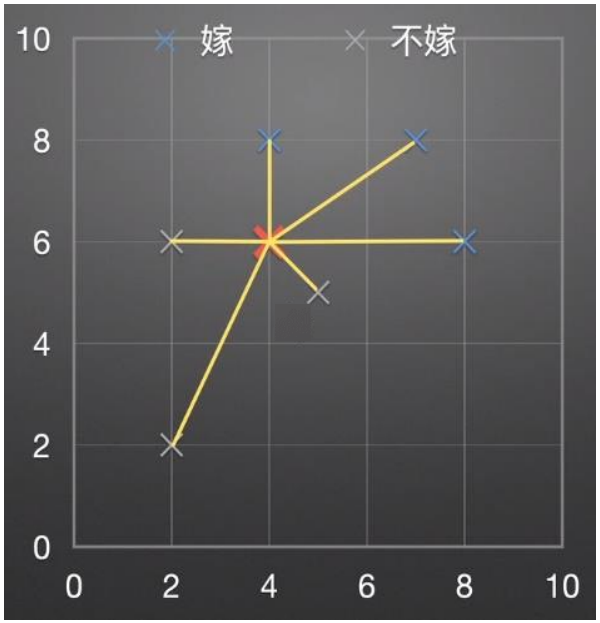


Regression

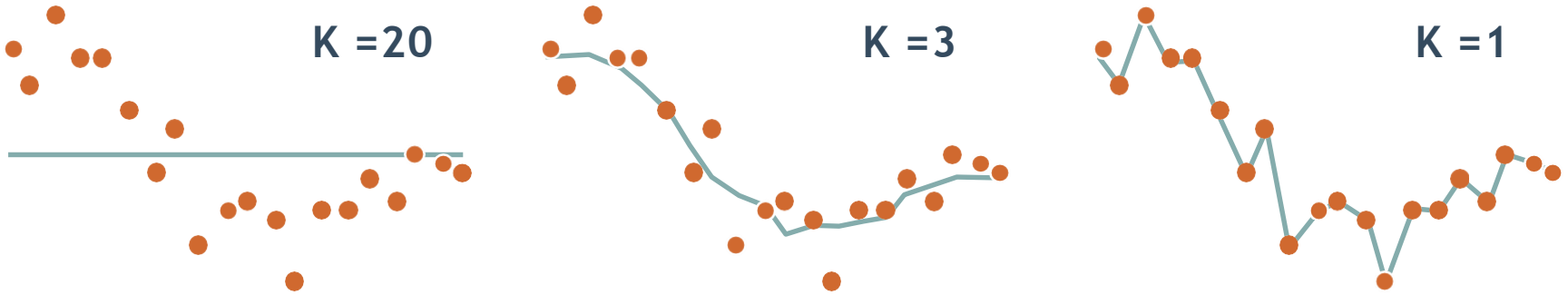
$$y = \frac{y_4 + y_1 + y_5 + y_3 + y_8}{k}$$

K近邻回归

序号	财富	颜值	嫁吗	综合
1	7	8	嫁	8
2	8	6	嫁	7
3	4	8	嫁	6
4	5	5	不嫁	5
5	2	2	不嫁	2
6	2	6	不嫁	4
7	4	6		?



K近邻回归



K近邻模型的特点

优点：

1. 建模快，因为它只是简单地存储数据；
2. 思想简单，理论成熟，既可以用来做分类也可以用来做回归；
3. 可用于非线性分类；
4. 准确度高，对数据没有假设，对离群值（异常值）不敏感；

缺点：

1. 运行速度慢，因为需要计算很多的距离，计算量大；
2. 样本不平衡问题（即有些类别的样本数量很多，而其它样本的数量很少）；
3. 需要大量的内存。

K近邻模型的语法

导入包含分类方法的类：

```
from sklearn.neighbors import KNeighborsClassifier
```

创建该类的一个对象：

```
KNN = KNeighborsClassifier(n_neighbors=3)
```

拟合数据集，即训练KNN模型，并用训练好的模型预测数据的标签：

```
KNN = KNN.fit(X_data, y_data)

y_predict = KNN.predict(X_data)
```

http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

K近邻模型的语法

导入包含分类方法的类：

```
from sklearn.neighbors import KNeighborsClassifier
```

创建该类的一个对象：

```
KNN = KNeighborsClassifier(n_neighbors=3)
```

拟合数据集，即训练KNN模型，并用训练好的模型预测数据的标签：

```
KNN = KNN.fit(X_data, y_data)

y_predict = KNN.predict(X_data)
```

这种 fit 和 predict/transform 语法会贯穿整个课程

K近邻模型的语法

导入包含分类方法的类：

```
from sklearn.neighbors import KNeighborsClassifier
```

创建该类的一个对象：

```
KNN = KNeighborsClassifier(n_neighbors=3)
```

拟合数据集，即训练KNN模型，并用训练好的模型预测数据的标签：

```
KNN = KNN.fit(X_data, y_data)
y_predict = KNN.predict(X_data)
```

回归使用KNeighborsRegressor

K-D树

从一组数据（2，5，4，9，6，8，3）中找到“8”？

K-D树

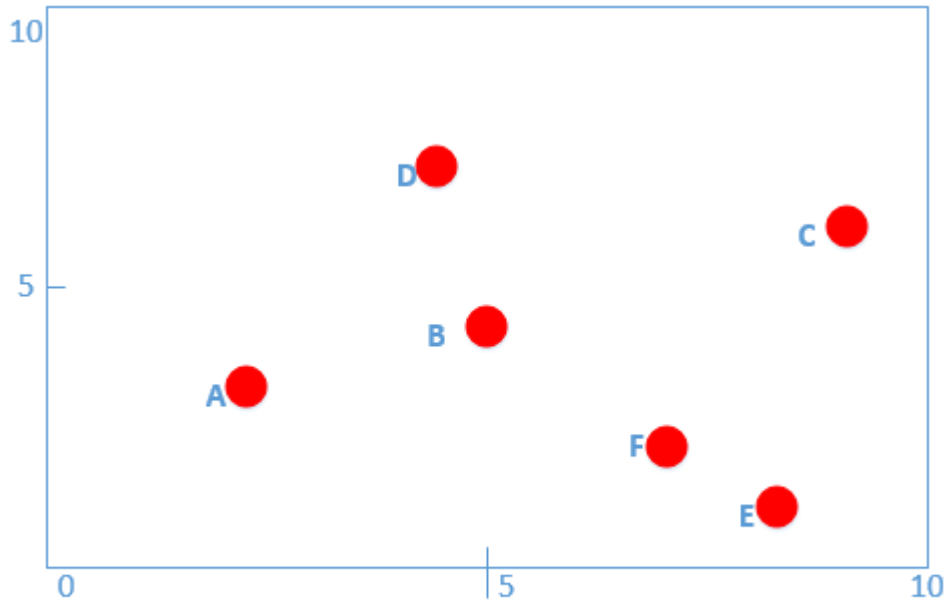
给定一个二维空间的数据集：

T = {（2,3），（5,4），（9,6），(4,7），（8,1），（7,2）},

构造一个平衡K-D树。

为了方便，我这里进行编号

A(2,3)、B（5,4）、C（9,6）、D（4,7）、E（8,1）、F（7,2）

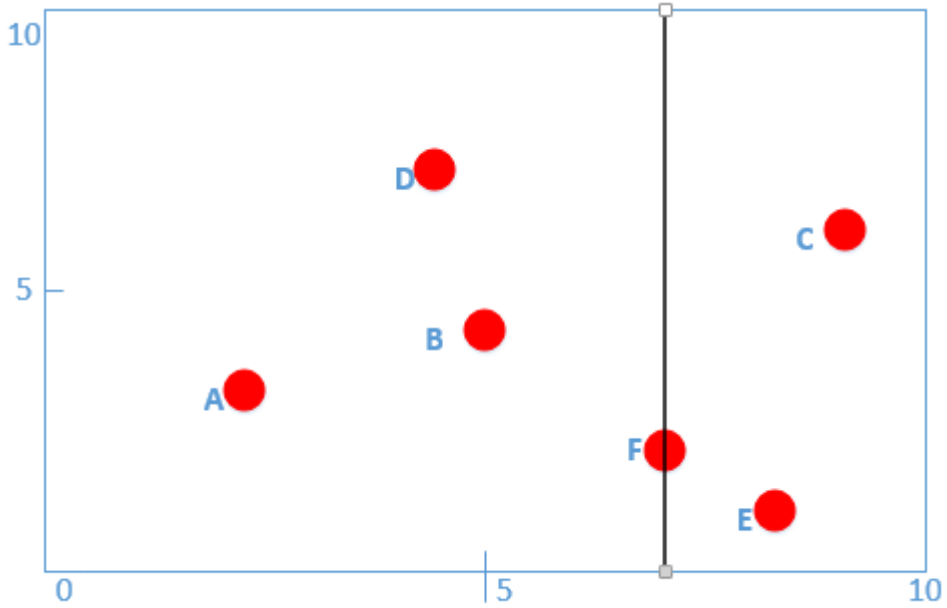


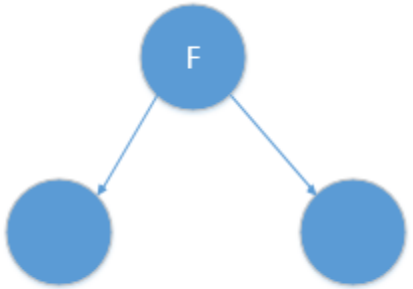
K-D树

首先先沿 x 坐标进行切分，我们选出 x 坐标的中位点，获取最根部节点的坐标，对数据点x坐标进行排序得：

A(2,3)、D（4,7）、B（5,4）、F（7,2）、E（8,1）、C（9,6）

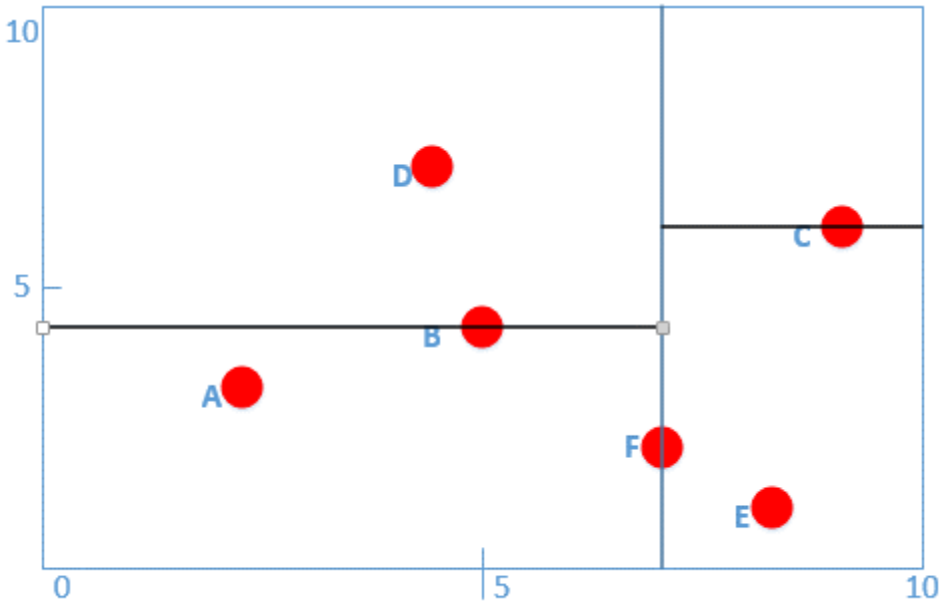
则我们得到中位点为B或者F，我这里选择F作为我们的根结点，并作出切分（并得到左右子树）

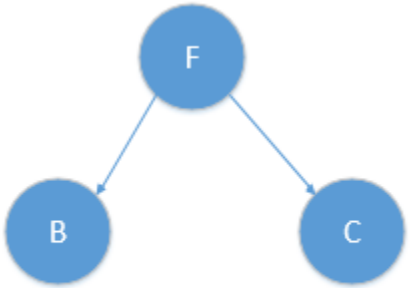




K-D树

再沿 y 坐标进行切分，分别递归的在F对应的左子树与右子树按y轴进行分类，得到中位节点分别为B，C点





K-D树

再沿 x 坐标进行切分，B的左子树为A，右子树为D，C的左子树为E