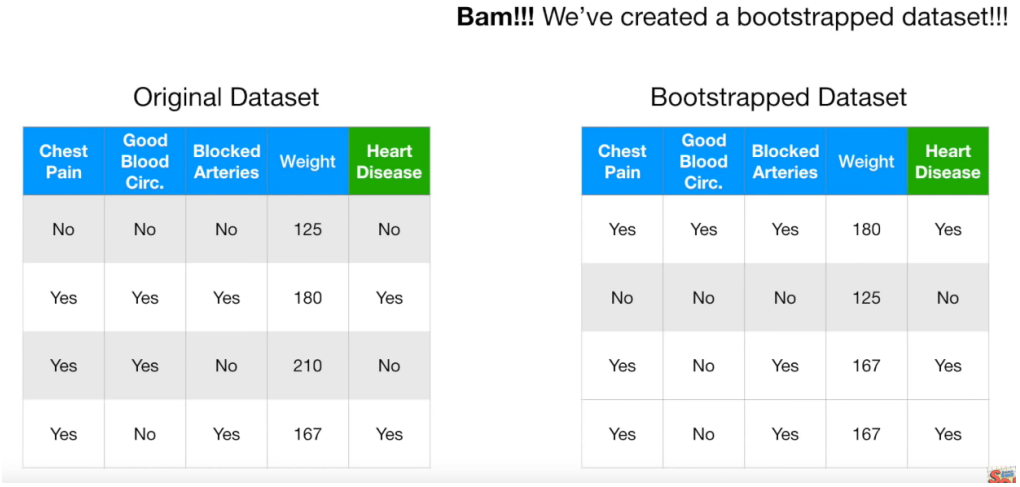
Decision Tree对目前样本有很好的效果，但对新样本则不够灵活。

## 如何建立Random Forest：

### Step1: Create a "bootstrapped" dataset

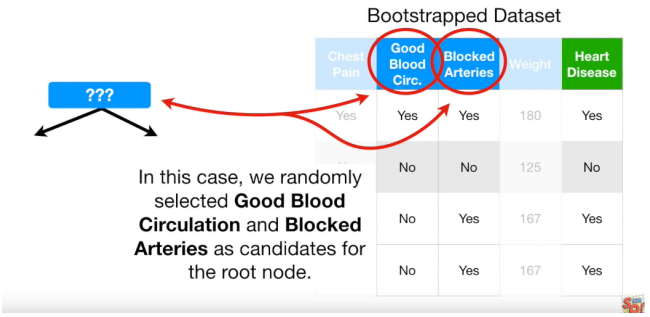
从原dataset随机选取样本，构建一个总样本数与原dataset相同的bootstrapped dataset.

重点：可以重复选取某一样本

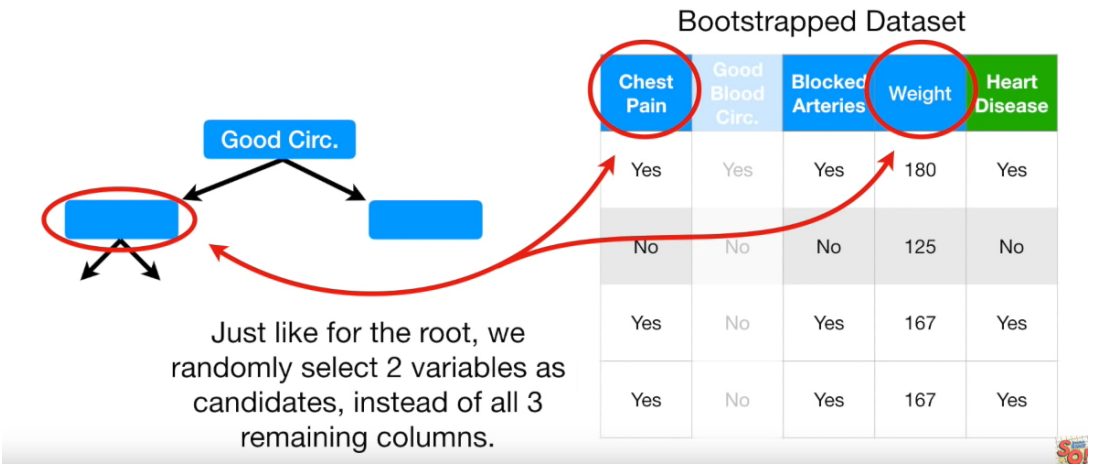



### Step2：使用bootstrapped数据集构建decision tree，但每一步只用其中的某几个特征

E.g 随机选取两个feature作为root node的候选feature




选取Good Circ作为root node之后，在剩余三个feature中选取两个作为candidate feature，继续构建子树。

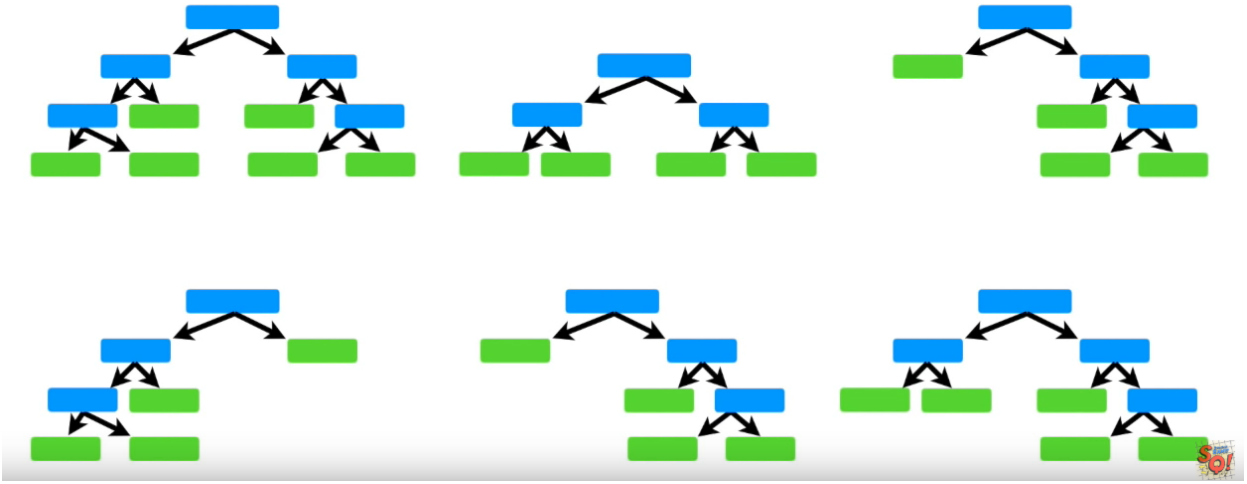


We built a tree…
1) Using a bootstrapped dataset

2) Only considering a random a subset of variables at each step.

**Step3：重复1.2步，使用不同的bootstrapped dataset构建多个decision tree，建立random forest**

Using a bootstrapped sample and considering only a subset of the variables at each step results in a wide variety of trees.



## 如何使用random forest进行分类?

将新样本输入到每个decision tree中，用投票决策的方式，得出最后的分类。

**Terminology Alert!!!**

**B**ootstrapping the data plus using the **agg**regate to make a decision is called "**Bagging**"

| Chest Pain | Good Blood Circ. | Blocked Arteries | Weight | Heart Disease |
|---|---|---|---|---|
| Yes | No | No | 168 | YES |

| Heart Disease | |
|---|---|
| Yes | No |
| 5 | 1 |

5个decision tree中将样本分类到YES，1个decision tree将样本分类到NO。因此该样本的类别为YES。

## 如何评价一个random forest?

**Step1：将没有选入到bootstrapped dataset中的所有样本组成一个新的out-of-bag dataset**

一般来说，out-of-bag中的样本数为原dataset中样本数的1/3

**Step2：将out-of-bag中的每一样本输入到random forest中的每一个decision tree，统计正确/误分类数**

| Classification of the Out-Of-Bag sample | |
|---|---|
| Yes | No |
| 1 | 3 |

| Classification of the Out-Of-Bag sample | |
|---|---|
| Yes | No |
| 4 | 0 |

| Classification of the Out-Of-Bag sample | |
|---|---|
| Yes | No |
| 3 | 1 |

同样通过vote的方式确定每个out-of-bag sample是否被正确分类。

**Step3：计算Out-Of-Bag Error**

Ultimately, we can measure how accurate our random forest is by the proportion of Out-Of-Bag samples that were correctly classified by the Random Forest.

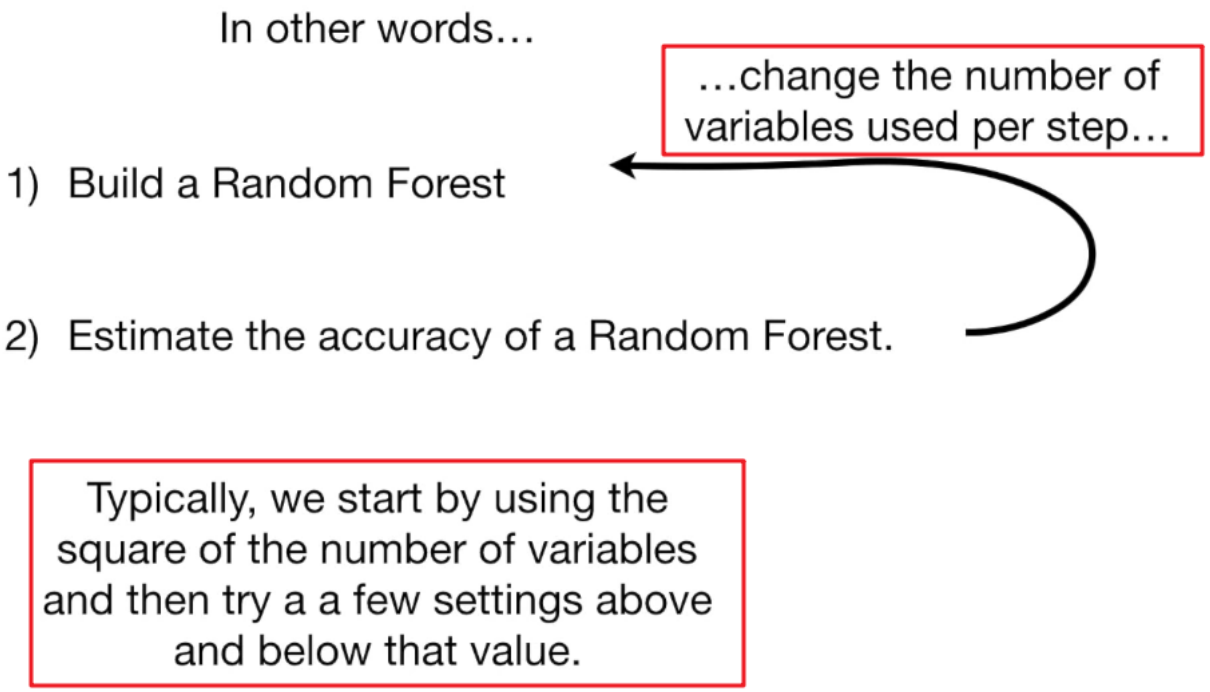
The proportion of Out-Of-Bag samples that were *incorrectly* classified is the "**Out-Of-Bag Error**"

Out-Of-Bag Error = 正确分类的out-of-bag sample数/总out-of-bag sample数

---

## 如何改进random forest?

在构建decision tree时，可以选择不同数目的候选feature（上面一开始选择2个feuature作为candidate，可以改为选择3个等）

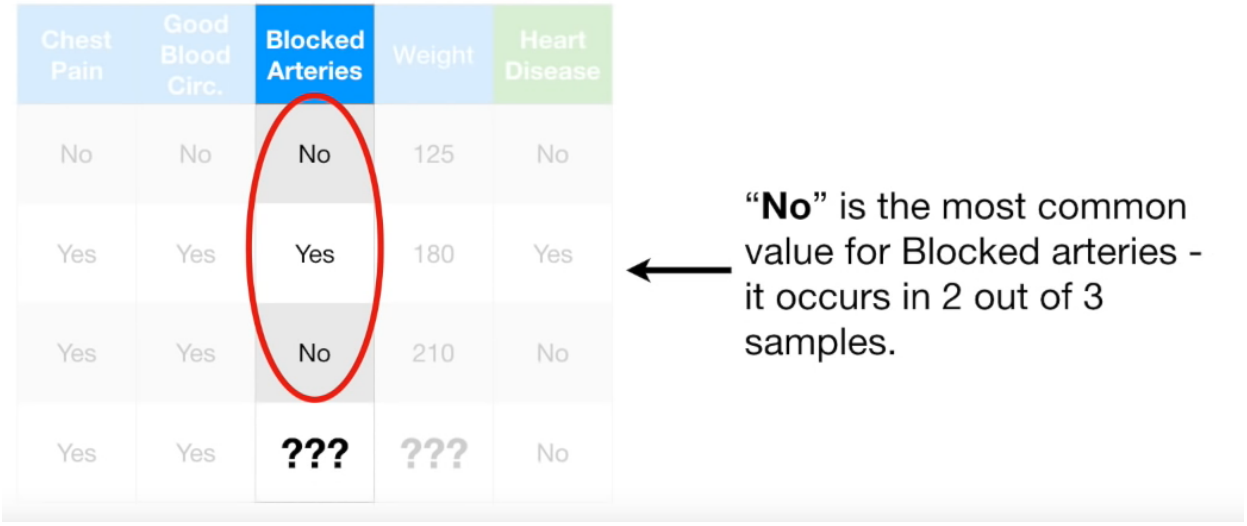



一般来说，随机选择的feature数为总feature数的平方根，在此基数上做微调。

## Missing Data and Clustering

**处理训练集中的Missing Data:**

**Step1：对Missing Data进行初步猜测**




对YES/NO的，根据most common的原则进行猜测。

| Chest Pain | Good Blood Circ. | Blocked Arteries | Weight | Heart Disease |
|---|---|---|---|---|
| No | No | No | 125 | No |
| Yes | Yes | Yes | 180 | Yes |
| Yes | Yes | No | 210 | No |
| Yes | Yes | No | 180 | No |

In this case, the median value is 180

对数值型的，取中值。

**Step2：对猜测进行refine，找出最相似的样本进行refine**

使用proximity matrix进行相似度分析

Proximity matrix:

行列均为样本编号。对N个样本，则matrix的维度为N*N。

每个样本输入到每个decision tree中，记录哪些样本和当前猜测的样本到达相同的叶节点。

Filled-in Missing Values

| Chest Pain | Good Blood Circ. | Blocked Arteries | Weight | Heart Disease |
|---|---|---|---|---|
| No | No | No | 125 | No |
| Yes | Yes | Yes | 180 | Yes |
| Yes | Yes | No | 210 | No |
| Yes | Yes | **No** | **180** | No |

Because no other pair of samples ended in the same leaf node, our proximity matrix looks like this after running the samples down the first tree.

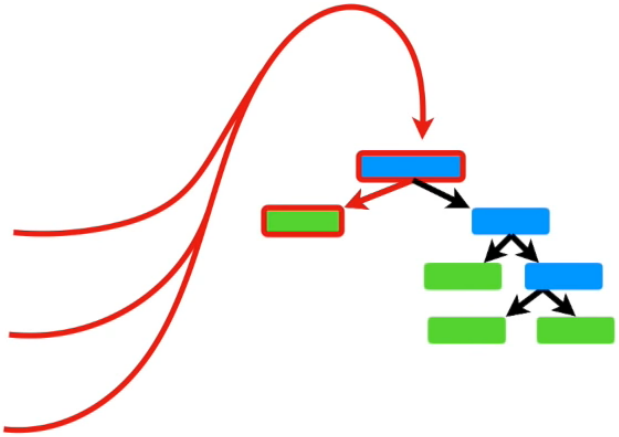
| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | 1 |
| 4 | | | 1 | |

在第一个decision tree中，样本3和4到达相同的叶节点，则在proximity matrix中相应位置+1.

Filled-in Missing Values

| Chest Pain | Good Blood Circ. | Blocked Arteries | Weight | Heart Disease |
|---|---|---|---|---|
| No | No | No | 125 | No |
| Yes | Yes | Yes | 180 | Yes |
| Yes | Yes | No | 210 | No |
| Yes | Yes | **No** | **180** | No |

**NOTE:** Samples 2, 3 and 4 all ended up in the same leaf node.



在第二个decision tree中，样本2,3,4到达相同的叶节点，则matrix更新为：

…after the second tree, we add 1 to any pair of samples that ended up in the same leaf node.

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | 1 | 1 |
| 3 | | 1 | | 2 |
| 4 | | 1 | 2 | |

Sample 2 also ended up in that same node.

如此遍历所有decision tree，最终得到matrix为：

Ultimately, we run the data down all the trees and the proximity matrix fills in.

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | | 2 | 1 | 1 |
| 2 | 2 | | 1 | 1 |
| 3 | 1 | 1 | | 8 |
| 4 | 1 | 1 | 8 | |

将每个元素除以decision tree的总数，更新为：

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | | 0.2 | 0.1 | 0.1 |
| 2 | 0.2 | | 0.1 | 0.1 |
| 3 | 0.1 | 0.1 | | 0.8 |
| 4 | 0.1 | 0.1 | 0.8 | |

Then we divide each proximity value by the total number of trees. In this example, assume we had 10 trees.

**Refine YES/NO:**

**计算YES：**

Filled-in Missing Values

| Chest Pain | Good Blood Circ. | Blocked Arteries | Weight | Heart Disease |
|---|---|---|---|---|
| No | No | No | 125 | No |
| Yes | Yes | Yes | 180 | Yes |
| Yes | Yes | No | 210 | No |
| Yes | Yes | ??? | ??? | No |

For Blocked Arteries, we calculate the weighted frequency of "Yes" and "No, using proximity values as the weights.

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | | 0.2 | 0.1 | 0.1 |
| 2 | 0.2 | | 0.1 | 0.1 |
| 3 | 0.1 | 0.1 | | 0.8 |
| 4 | 0.1 | 0.1 | 0.8 | |

Filled-in Missing Values

| Chest Pain | Good Blood Circ. | Blocked Arteries | Weight | Heart Disease |
|---|---|---|---|---|
| No | No | No | 125 | No |
| Yes | Yes | Yes | 180 | Yes |
| Yes | Yes | No | 210 | No |
| Yes | Yes | ??? | ??? | No |

The weighted frequency for **"Yes"** is…

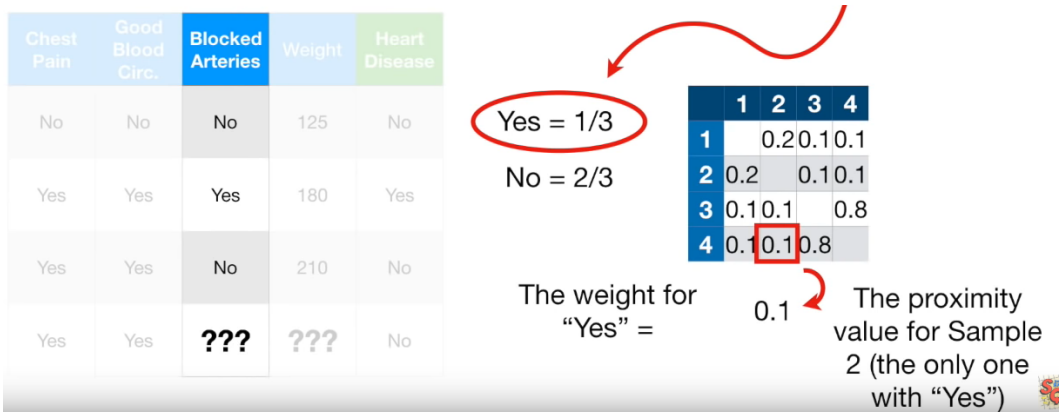
Yes = $\frac{1}{3}$ x The weight for **"Yes"**

Yes = 1/3

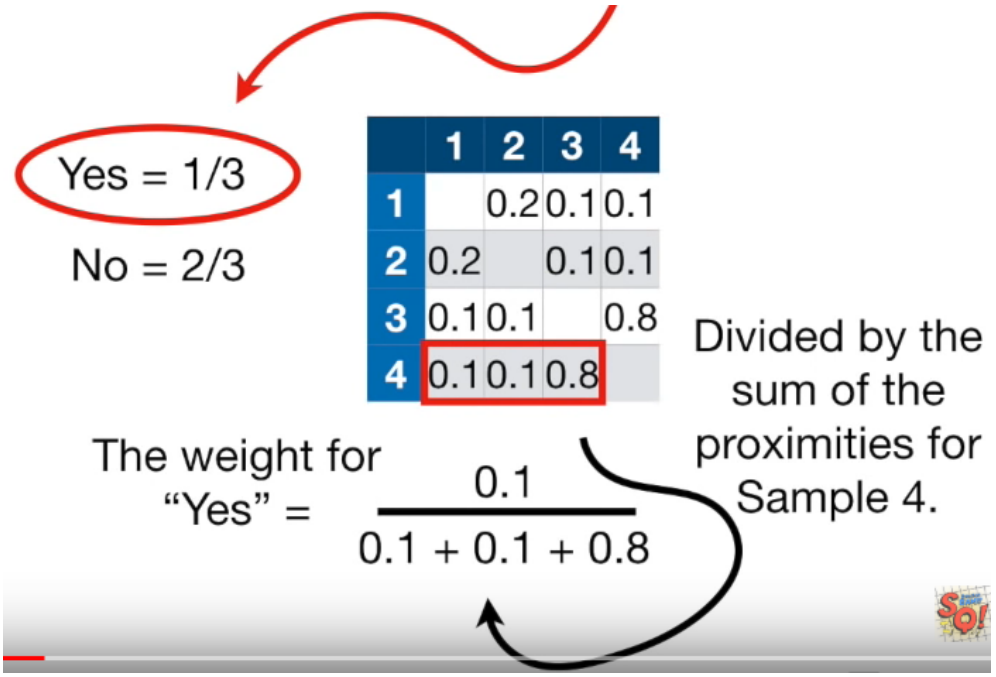
No = 2/3

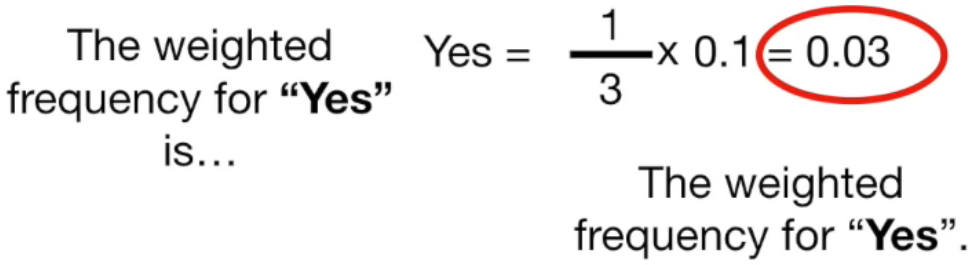
| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | | 0.2 | 0.1 | 0.1 |
| 2 | 0.2 | | 0.1 | 0.1 |
| 3 | 0.1 | 0.1 | | 0.8 |
| 4 | 0.1 | 0.1 | 0.8 | |

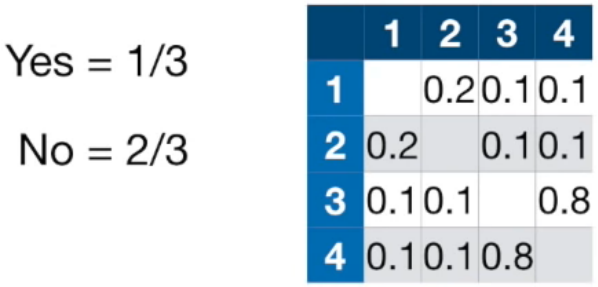
The weight for "Yes" = $\frac{\text{Proximity of "Yes"}}{\text{All Proximities}}$



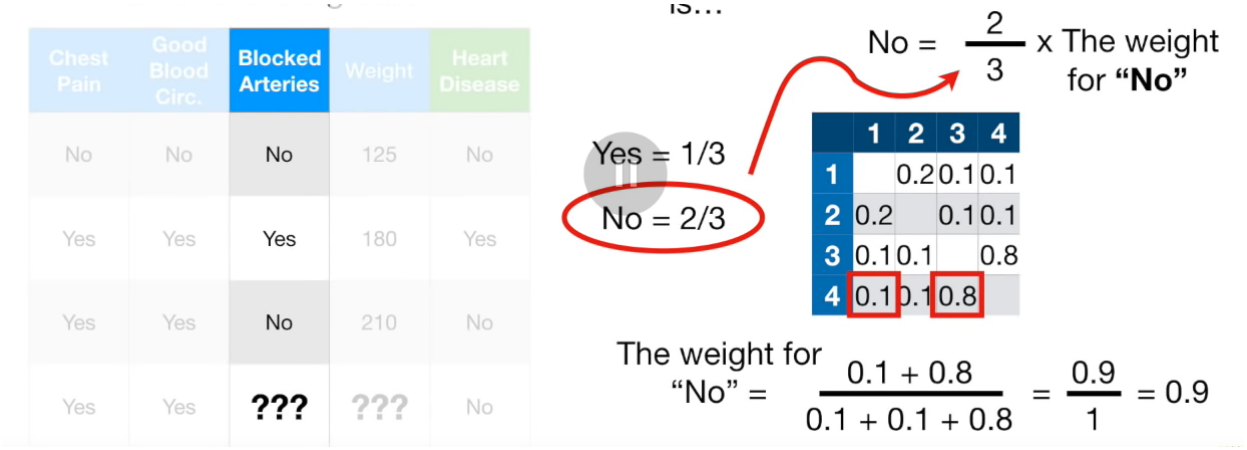

在4对应的行中，只有样本2的Blocked Arteries为YES。




**计算NO：**




The weighted frequency for **"No"** is…

$$\text{Yes} = \frac{1}{3} \times 0.1 = 0.03$$

$$\text{No} = \frac{2}{3} \times 0.9 = 0.6$$

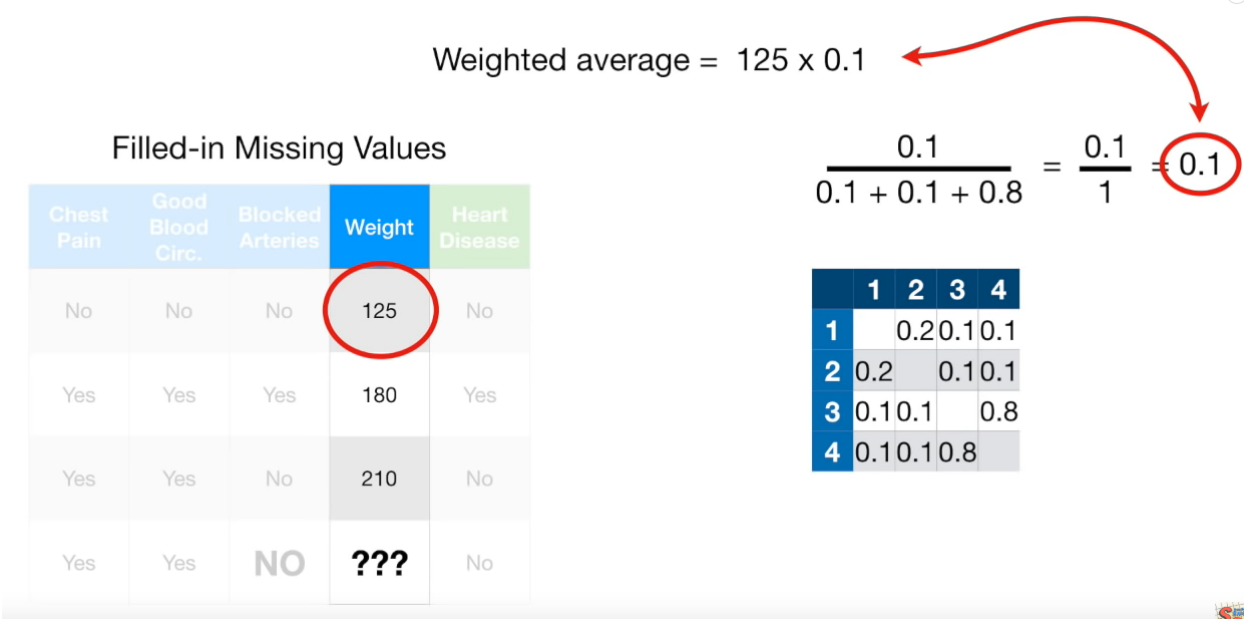
Yes = 1/3

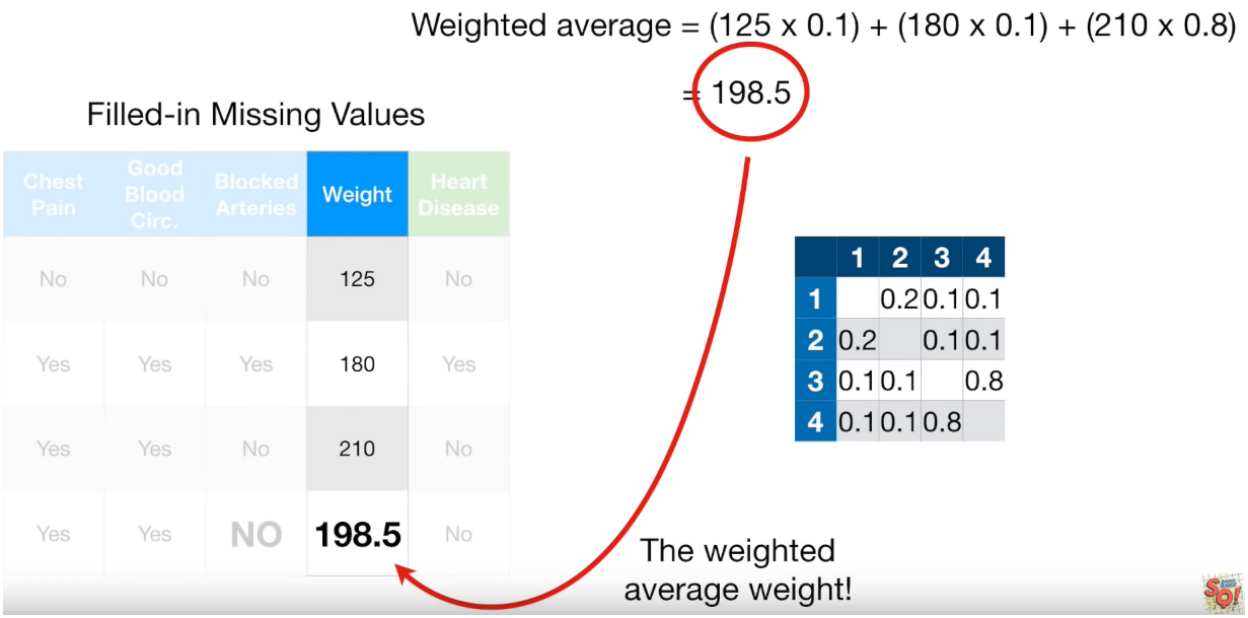
No = 2/3

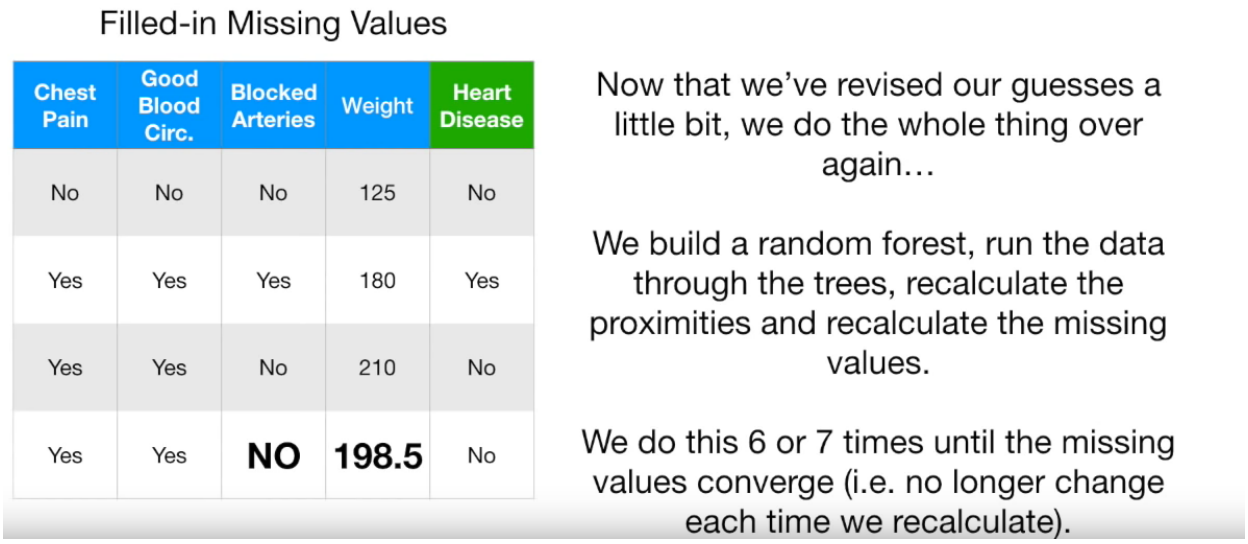
| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | | 0.2 | 0.1 | 0.1 |
| 2 | 0.2 | | 0.1 | 0.1 |
| 3 | 0.1 | 0.1 | | 0.8 |
| 4 | 0.1 | 0.1 | 0.8 | |

No的weighted frequency=0.6，比YES的0.03高，因此预测为NO。

**Refine Weighted(数值型特征值):**

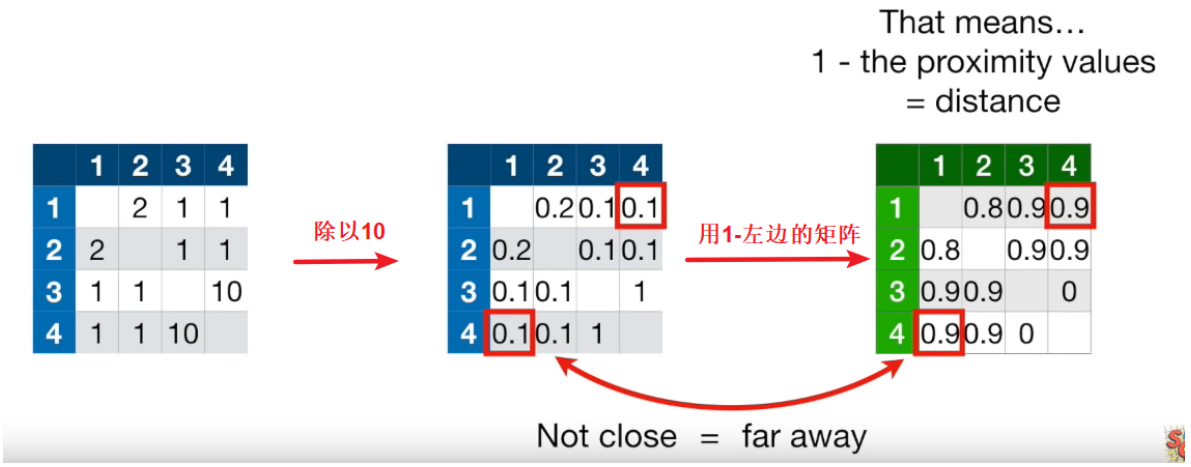




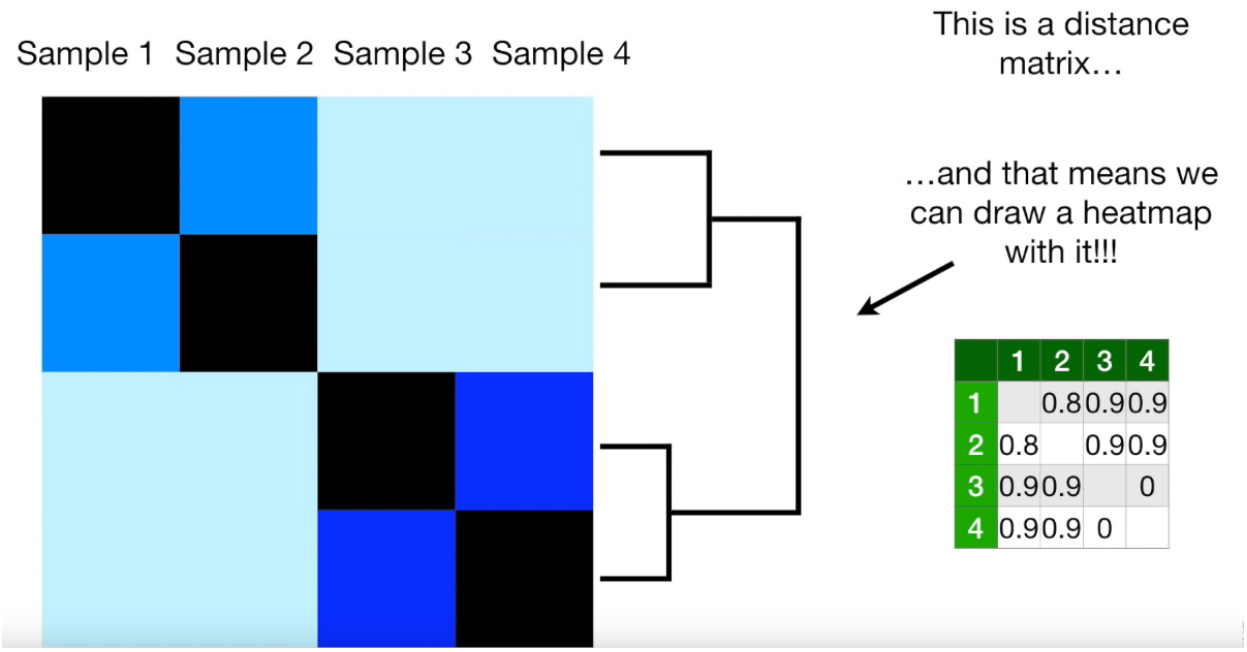




用新填充的数据重新计算，直到填充数据不在改变。

PS：proximity matrix和distance的关系



Proximity越大，距离越近，越相似。




可以讲Distance矩阵转化为heat map进行聚类。

**处理测试集中的Missing Data:**

| Chest Pain | Good Blood Circ. | Blocked Arteries | Weight | Heart Disease |
|---|---|---|---|---|
| Yes | No | ??? | 168 | |

So we want to know if they have heart disease or not…

Blocked Arteries特征值的缺失导致无法预测（无法在decision tree上运行）。

生成样本的两个copy，一个预测为YES，一个为NO。

In this case, you plug in the most common value/median value for all observations in the training dataset that have that same category as the new copy that you created. For example, we created two new copies of the observation: one with with heart disease and one without heart disease. Now, for the new copy with heart disease, we plug in the most common value from the observations in the training dataset that have heart disease. For the new copy without heart disease, we plug in the most common value from the observations in the training dataset that do not have heart disease. We can then use the iterative method to refine the guess if we

want, or we can just run those two copies down the tree and use the classification from the copy that got the most correct votes.

确定填充值的两种方式：
1. initial的填充值根据label中的common value或者median value进行确定，再用上面proximity的方法进行修正，得到可靠的填充值后再进行分类。
2. 使用initial填充值，输入到每个decision tree中，根据投票方式决定哪个是正确的。
E.g：Blocked Arteries为YES时，label为YES的decision tree的个数为3，Blocked Arteries为NO时，label为NO的decision tree个数为1，则确定Blocked Arteries为YES。