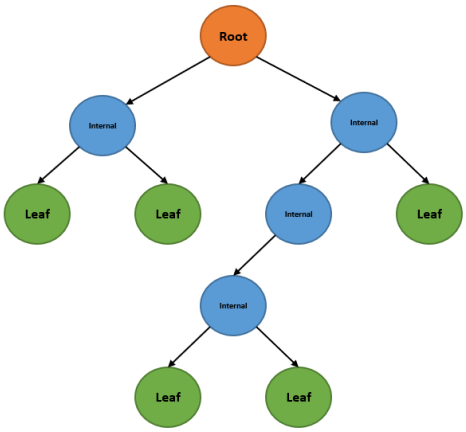
Decision Tree 决策树

Decision Tree 是一种基本的分类 (Classification) 方法和回归 (Regression) 方法，可以被认为是 $if-then$ 规则的集合，也可以是定义在特征空间与类空间上的条件概率分布。Decision Tree 具有可读性，分类的速度快。要使用 Decision Tree，有 3 个步骤：

1. 特征选择 (Features Selection)
2. 生成决策树 (Generate Decision Tree)
3. 修剪决策树 (Pruning)

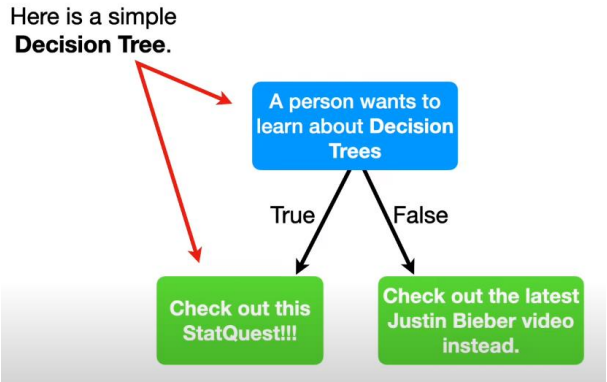
决策树的架构 (Decision Tree Structure)



上图是决策树的基本架构，决策树里有结点 (Node) 和向边 (Directed Edge)。Node 分为 Root Node，Internal Node 和 Leaf Node。通常 Root Node 和 Internal Node 是对某一个特征进行验证，而 Leaf Node 则是答案，在 Classification 的问题里，Leaf Node 给出的就是分类的 Class。

决策树 If-Then

决策树可以看作是一个 If-Then 的集合，决策树的 Root 和 Internal Node 是一个 Condition，当 Input 走完所有的 Root 和 Internal Node，就会到达 Leaf Node 也就是得到最终的类别。



上图是一个决策树的 Example。根据给入的 Feature，会去到不同的 Branches。

Classification Tree

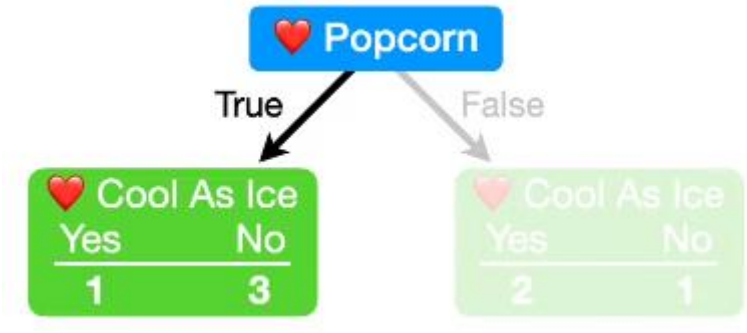
特征选择

首先需要对每一个 Feature 做特征选择计算，特征选择计算有几种方法，比较出名的是 Gini Impurity，别的特征选择方法还有 Entropy，Information Gain 等等。

$$Gini\ Impurity\ for\ the\ leaves = 1 - (Probability\ of\ \text{Yes})^2 - (Probability\ of\ \text{No})^2$$

$$Total\ Gini\ Impurity = Weighted\ Average\ of\ Gini\ Impurities\ for\ the\ leaves$$

Example



上图是整个 Dataset 里其中一个 Feature，对 Feature 和 Label 进行划分，当 Feature 是 True/False 的时候 Output Label 有多少个是 Yes 和 No。

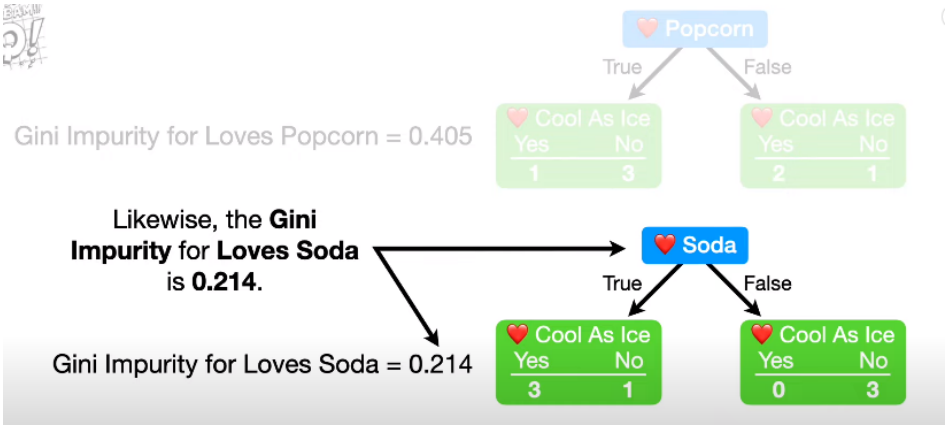
$$Gini\ Impurity\ (True) = 1 - \left(\frac{1}{1+3}\right)^2 - \left(\frac{3}{1+3}\right)^2 = 0.375$$

$$Gini\ Impurity\ (False) = 1 - \left(\frac{2}{2+1}\right)^2 - \left(\frac{1}{2+1}\right)^2 = 0.444$$

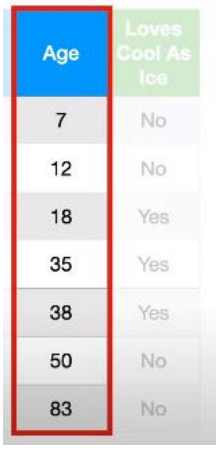
分别对两个不同的情况计算出 Gini Impurity 之后，需要对其做出总和。

$$Total\ Gini\ Impurity = \left(\frac{4}{4+3}\right)0.375 + \left(\frac{3}{4+3}\right)0.444 = 0.405$$

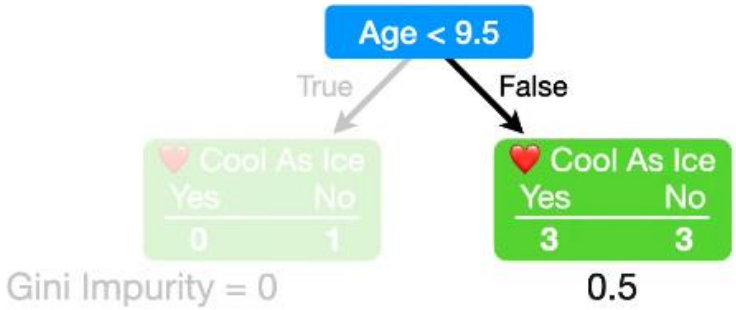
以上计算的只是对 Dataset 里其中一个 Feature 进行 Gini Impurity 的计算，需要对每个 Feature 都进行计算。



当 Feature 不是 Yes 或者 No，而是数目的时候，计算的方法有些许不同。首先先把该 Feature 与 Output Label 按照顺序排列



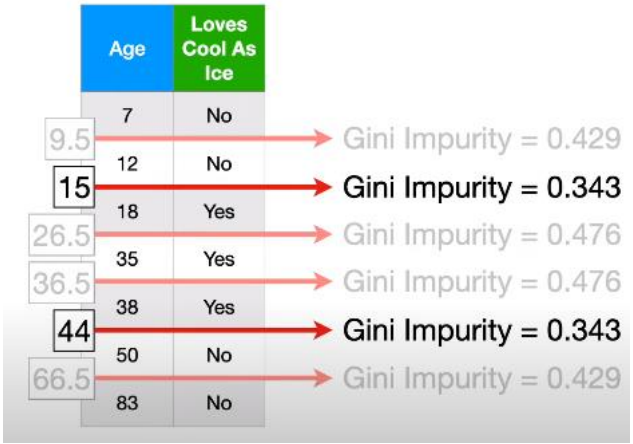
之后去每 2 个数值的平均，然后进行 Gini Impurity 的计算



$$Gini\ Impurity\ (Age < 9.5) = 1 - \left(\frac{0}{0+1}\right)^2 - \left(\frac{1}{0+1}\right)^2 = 0$$

$$Gini\ Impurity\ (Age \geq 9.5) = 1 - \left(\frac{3}{3+3}\right)^2 - \left(\frac{3}{3+3}\right)^2 = 0.5$$

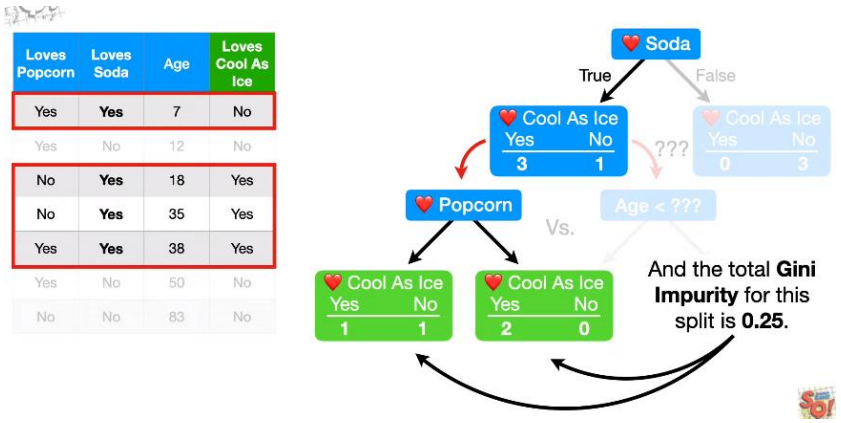
$$Total\ Gini\ Impurity = \left(\frac{1}{1+6}\right)0 + \left(\frac{6}{1+6}\right)0.5 = 0.429$$

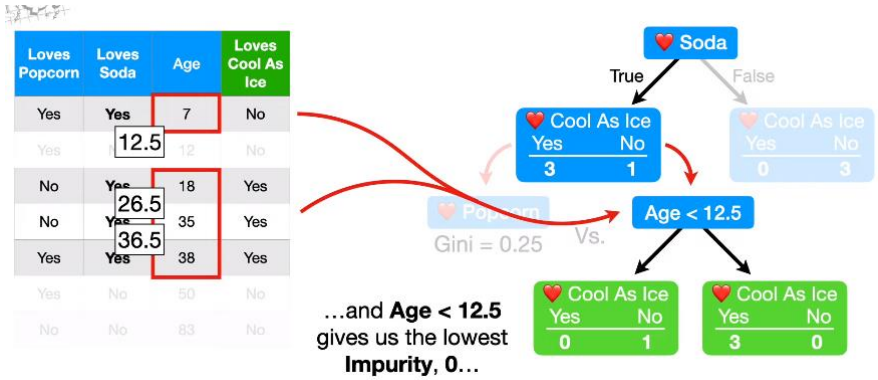


当对每个值都计算完 Gini Impurity 之后，选出最低值的 Gini Impurity 当作 Threshold，如果出现多个一样的最低值，随机选取任意一个只都行，上图可以使用 15 或者 44。

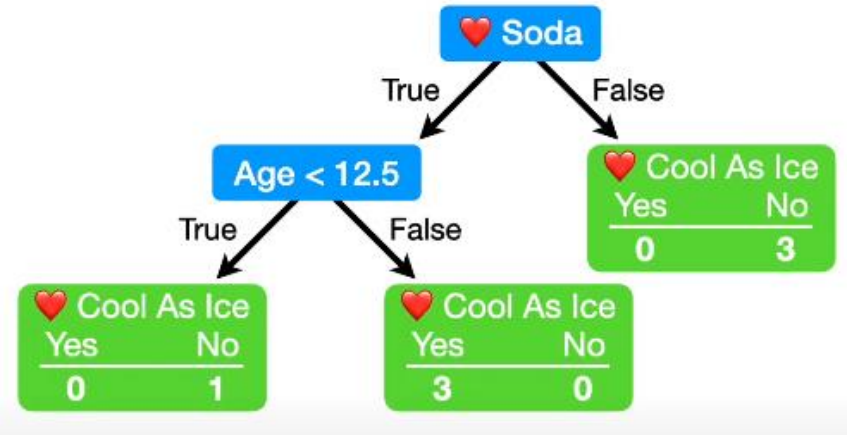
当对每个 Feature 进行 Gini Impurity 计算之后，需要选择哪一个 Feature 当作 Root。

一般的情况会选择最低 Gini Impurity 的 Feature 当作 Decision Tree 的 Root。选择了 Root 之后，需要选择其他的 Feature 当作下一层 Decision Tree 的 Condition (Internal Node)。这时候需要重新计算 Gini Impurity。





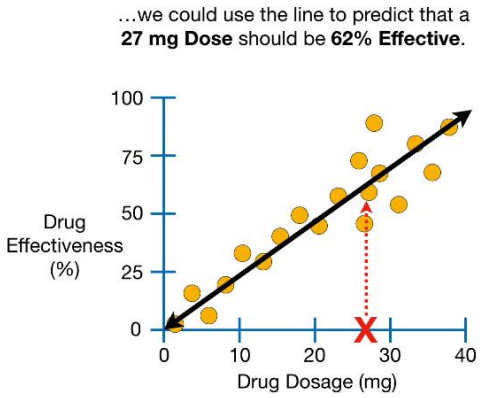
分别计算了剩余的 Feature 的 Gini Impurity 之后，选出有着最低的 Gini Impurity 的 Feature 当作该层的 Condition。如果是树的左侧，只需要看该 Condition 的数据，上图左侧是 True，所以只需要看所有 Love Soda 是 True 的数据。



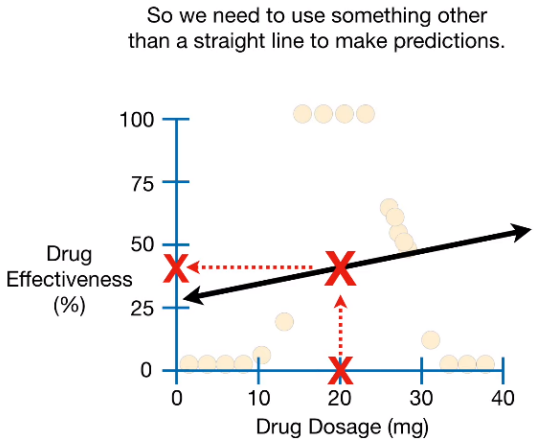
反复计算之后，就能得到最终的 Decision Tree，在上图没有放入 Love Popcorn 的 Feature 是因为没有必要了，因为当 Loves Soda 是 Yes 的时候，年龄小于 12.5 的，所有这样的数据表示 Output 都是 No，反之 Loves Soda 是 Yes 的时候，年龄大于 12.5 的，所有 Output 都是 Yes。在这种情况没必要再参考其他 Feature 了。

Regression Tree

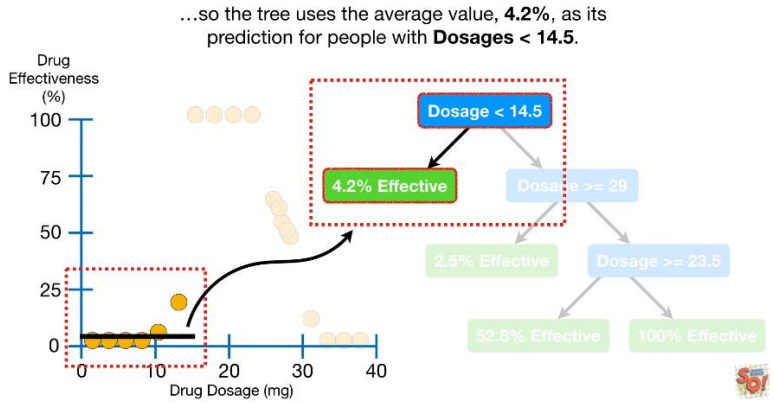
Regression Tree 和 Classification Tree 有些许不同。Regression Tree 的 Output 是一个值。



当 Dataset 是能被直线划分的时候，可以轻易的得出 Output。



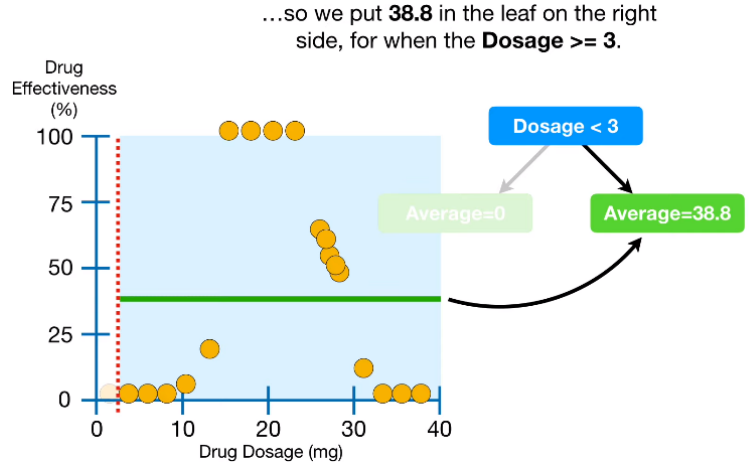
但是当 Data 不能被直线划分的时候，其中一个解决的办法就是使用 Regression Tree。



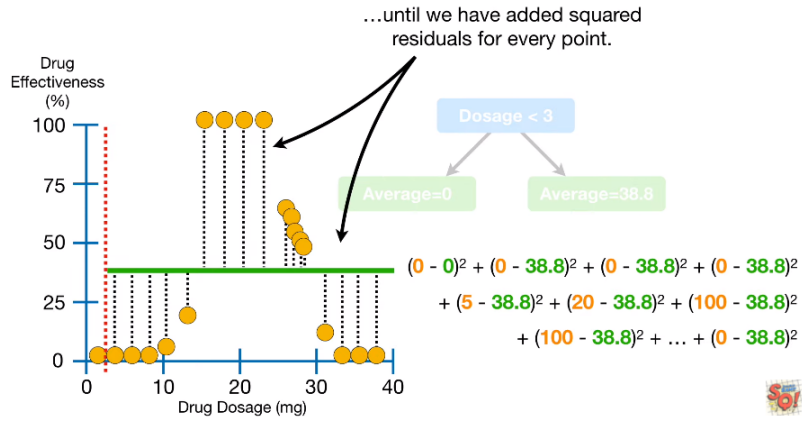
上图是一个 Regression Tree，在 Regression Tree 里，每个 Leaf Node 都代表着一个 Numeric Value。而这个 Numeric Value 是计算了一个 Region 里面所有 Data Points 的平均值。

当 Dataset 不是只有一个 Feature 的时候，其实是很难把这些 Data Plot 出来，但是使用 Regression Tree 的计算方法，能够实现，并给出预测值。

特征选择

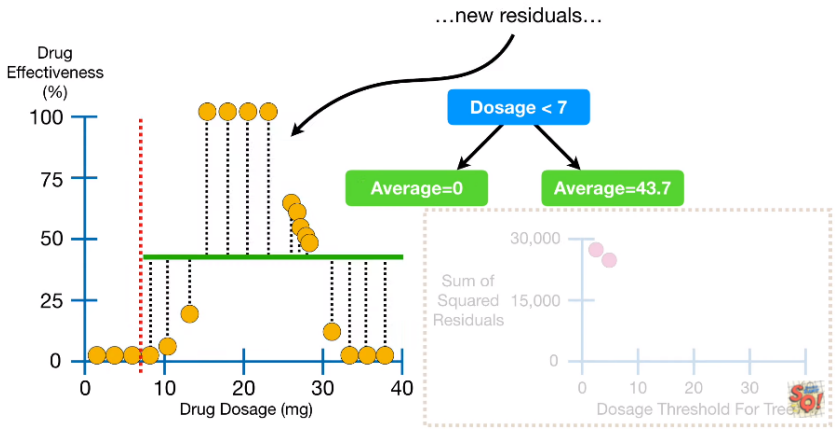


对于 Numeric Value 的 Feature，需要按照顺序排列，然后对 2 个点取平均值，然后再算出两个可能性的平均值 (小于 Threshold 的全部 Data Points 的平均值，和大于 Threshold 的全部 Data Points 的平均值)。

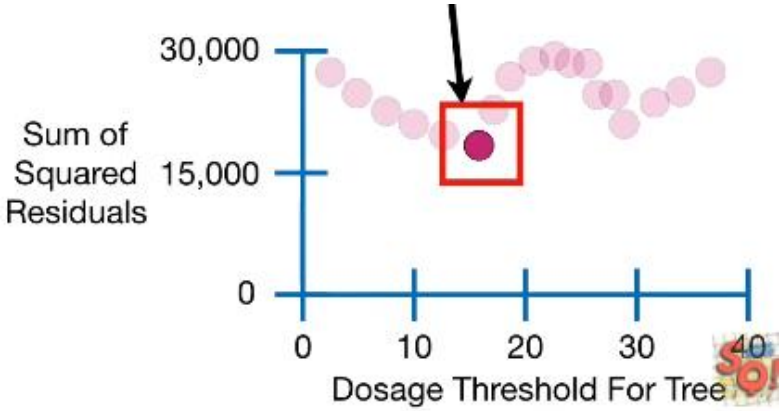


之后对每个 Data Points 计算 Squared Residual。小于 Threshold 的所有 Data Points 减去它们的 Mean 再取平方，大于 Threshold 的所有 Data Points 减去它们的 Mean 再取平方。

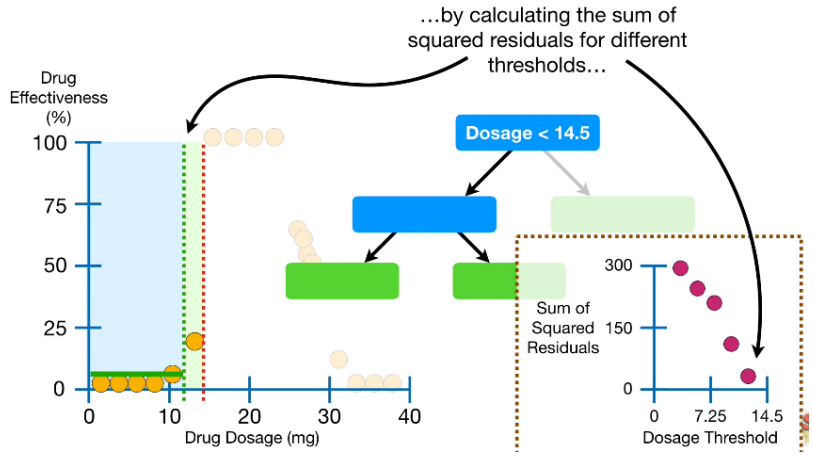
$$Squared\ Residual = \sum_{i=1}^{n}(x_i - \mu_1) + \sum_{i=n}(x_i - \mu_2)^2$$



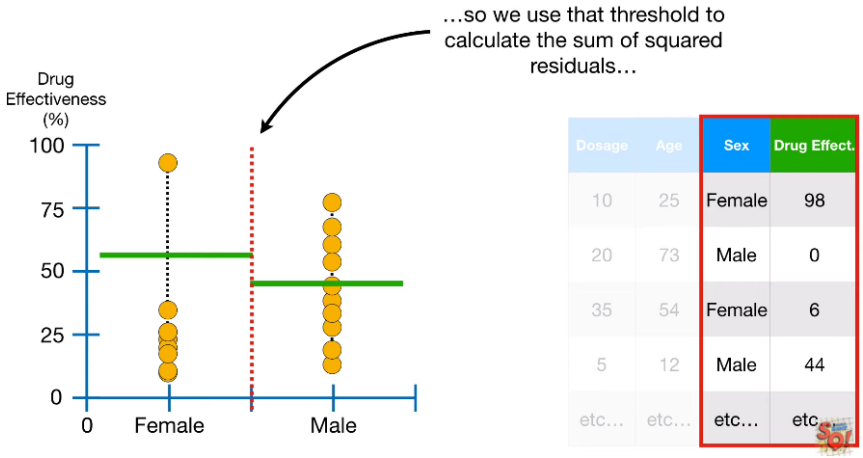
当前两个 Data Points 的 Squared Residual 计算好后，就对前一个与当前 Data Points 取平均，后计算 Average 值，再计算 Squared Residual，直到算完所有 Data Points。



算完所有 Data Points 之后，会得到一个如上图这样的 Sum of Squared Residuals Graph。有着最低 Squared Residual 的 Threshold 就是最适合的。



如果要对有更仔细的 Threshold，可以对各 True/False Threshold 的所有 Data Points 进行 Squared Residual 的计算 (Dosage < 14.5，就只看所有 < 14.5 的 Data Points)，然后选出最低 Squared Residual 的 Threshold，然后设置在 Regression Tree 里。分的越详细，Threshold 越多，这个 Tree 有可能会 Overfit，通常只对 ≥ 20 的 Data Points 再进行划分。

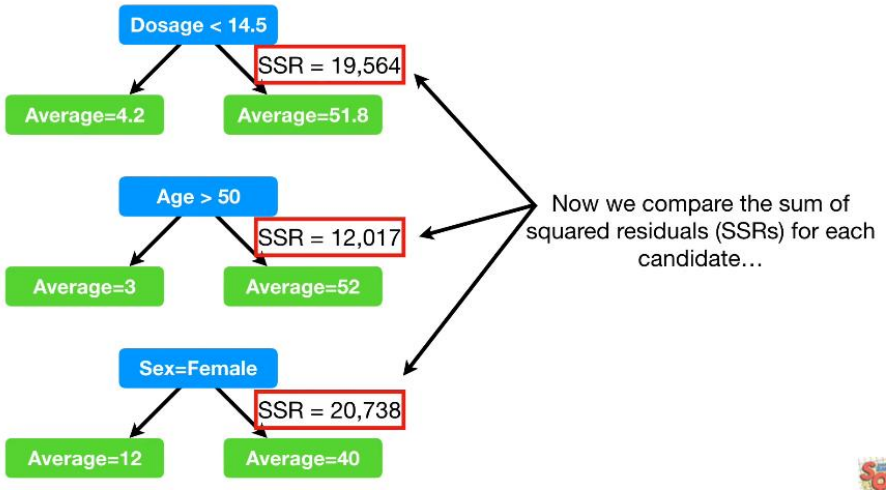


$$Squared\ Residual = \sum (Female_i - \mu_{Female})^2 + \sum (Male_i - \mu_{Male})^2$$

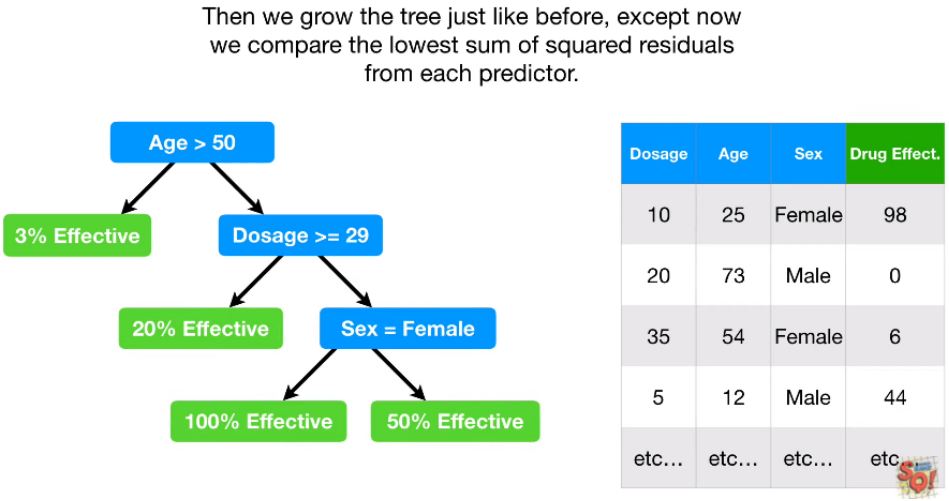
当数据不是 Numeric Value，而是两种类别，计算 Squared Residual 的方法则是对不同类别里所有 Data Points 计算平均值，然后就能计算 Squared Residual。

Multi-Dimensional Features

当 Features 是多维的时候，选择哪一个 Feature 当作树的 Root 就变成比较困难。需要对每一个 Feature 进行 Squared Residual 的计算，然后选出最低值的 Squared Residual。



当计算好所有 Feature 里最低的 SSR (Sum of Squared Residual) 值，然后进行比较，选出最低的 SSR 值。这个 Threshold 就会被当成树的 Root。

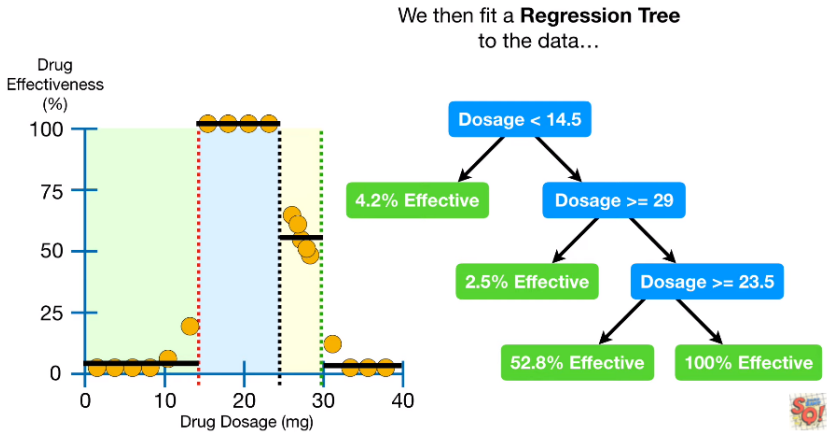


Dosage	Age	Sex	Drug Effect.
10	25	Female	98
20	73	Male	0
35	54	Female	6
5	12	Male	44
etc...	etc...	etc...	etc.

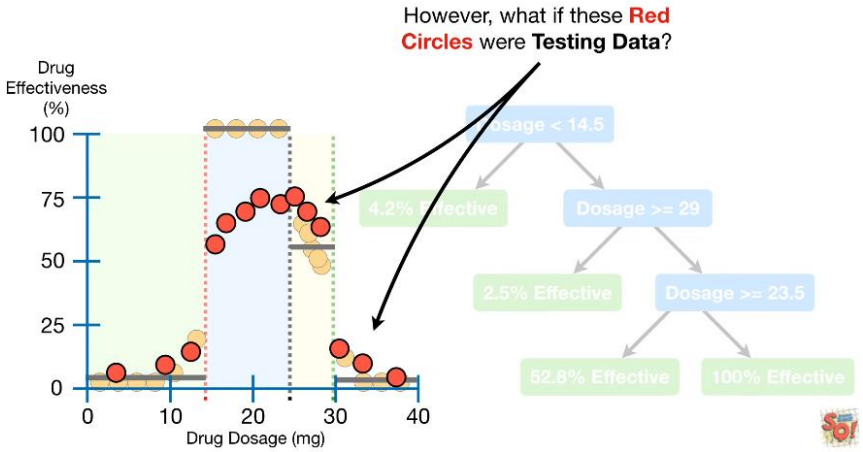
上图是一个 Multi-Features Regression Tree 的 Example。

修剪决策树 (Pruning)

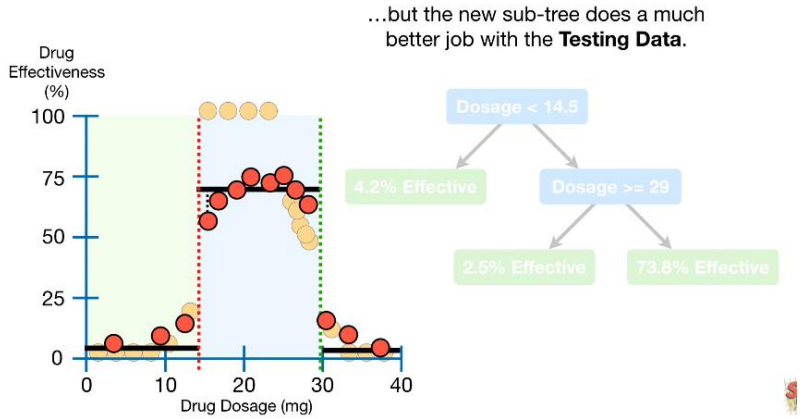
有许多 Pruning 的方法，在这里解释的是 Cost Complexity Pruning (Weakest Link Pruning)。



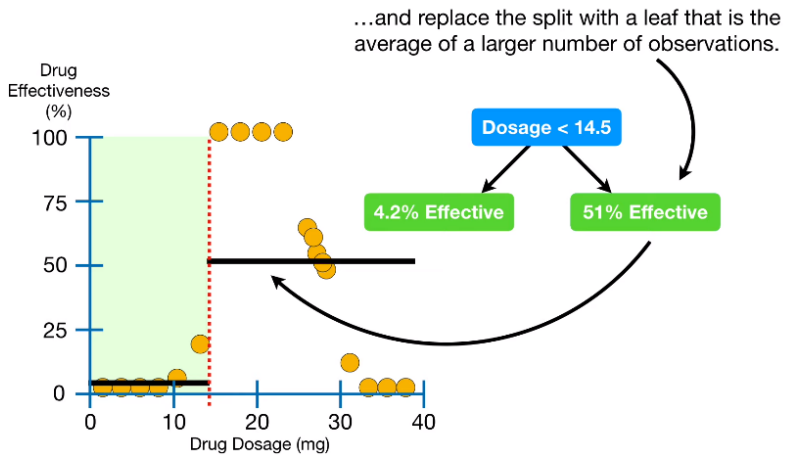
上图是一个 Regression Tree 的 Example，它很好的分类了 Data。



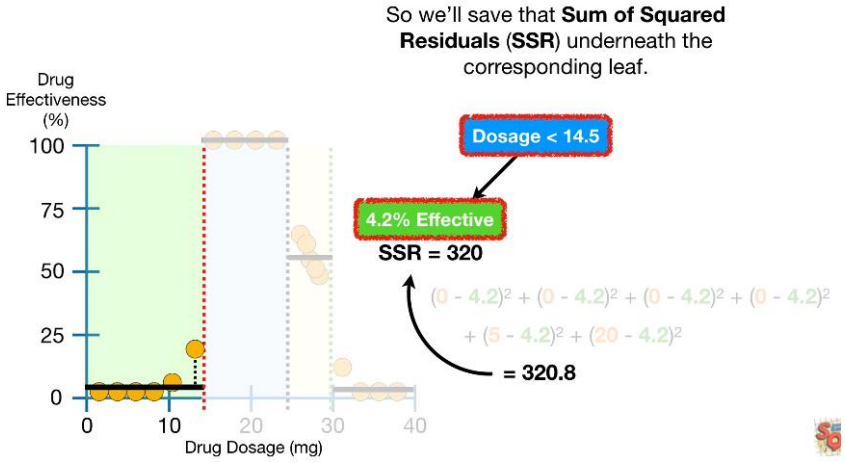
但是当这个 Regression Tree 使用在 Testing Data 的时候，可以看出，这个 Regression Tree 不能有很好的表现，也代表这个树已经 Overfit 了。要解决这个方法，需要拆除一些 Leaf Node，然后再重新计算 Output 的平均值。



这时候，可以发现，当 Regression Tree 没有分类的这么仔细的时候，数据变得比较平滑，Overfit 的问题也解决。

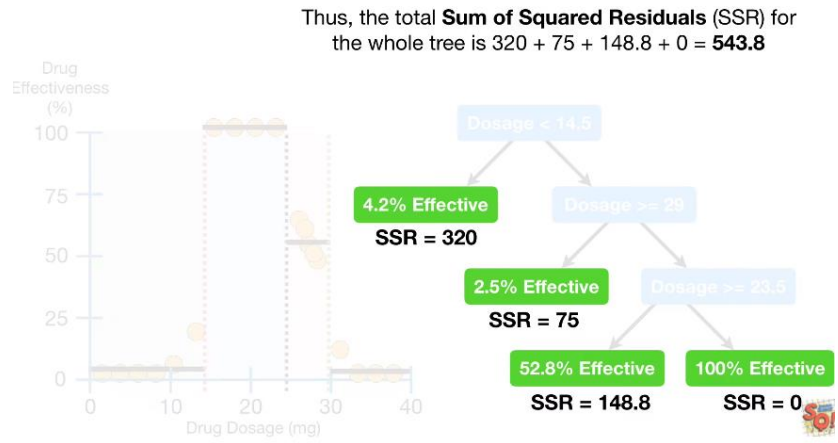


可以继续拆除跟多的 Leaf，然后对其进行测试，找出表现最好的树。要计算每个树的 Sum of Squared Residual 来衡量树的好坏。

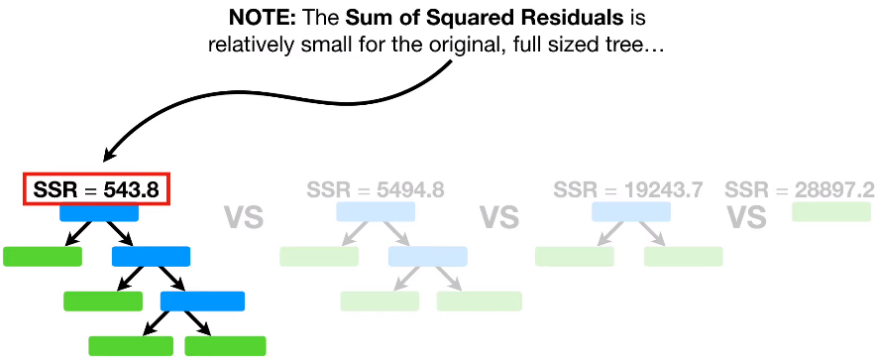


对每一个 Threshold 里的所有 Data Points 进行 Sum of Squared Residual 的计算。

$$Sum\ of\ Squared\ Residual = \sum (x_i - \mu)^2$$



当计算完每个 Threshold 的 Sum of Squared Residual 之后，将每个 Threshold 的 SSR 值加起来，就是这棵树的 SSR 值。



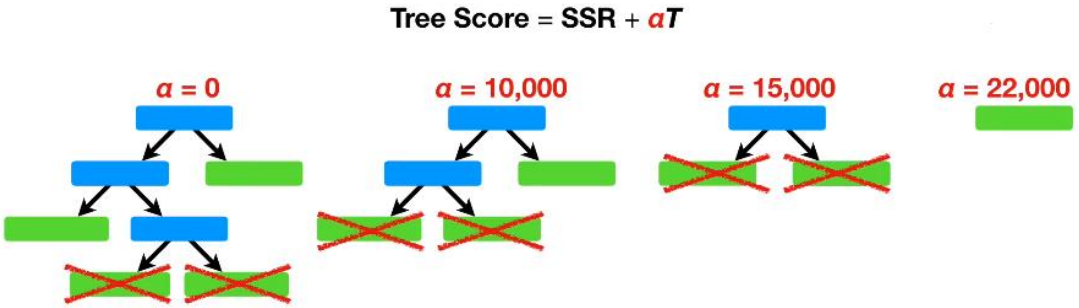
对每颗树计算完 SSR 之后，通过观察发现，越多 Threshold 的树，它的 SSR 值都是最低的，这是因为它与训练数据更加的贴近。这时候，除了计算每棵树的 SSR 之外，还需要增加一个 Tree Complexity Penalty。

$$Tree\ Score = SSR + \alpha T$$

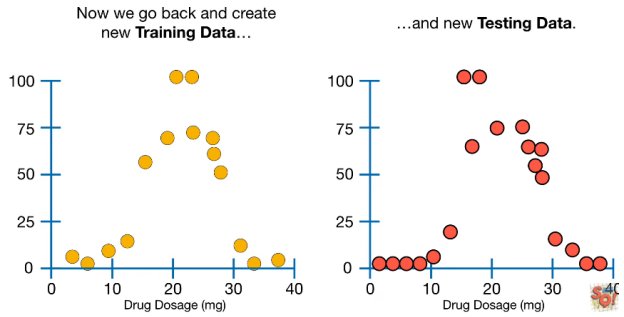
$$\alpha \rightarrow Tuning\ Parameter$$

$$T \rightarrow Number\ of\ Leaves\ in\ the\ Tree$$

首先先将所有 Dataset 里面的 Data 都拿来训练一个 Full Size 的 Regression Tree。然后 Prune 这棵树直到剩下 Leaf。设置不同的 α 值，让每一颗树都有最低 Tree Score 的时候。



当 α 的值越低，最完整的树的 Tree Score 就会是最低的，随着 α 的增加，越少 Leaf Node 的树越有 Advantage。



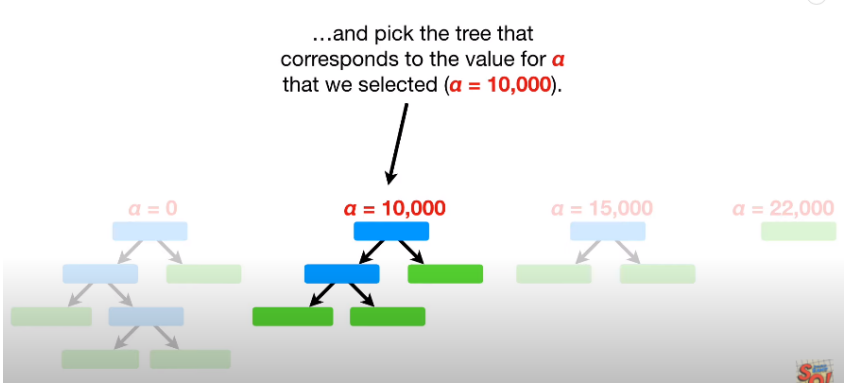
有了不同的 α 值之后，这时候需要使用 Cross Validation 的方法来选出最适合的 α。在这里可以使用 K-Fold Cross Validation 的方法。首先将 Dataset 里面的数据划分成 K 份，然后每次选不同的一份当作 Testing Data，其他的所有数据当作 Training Data，直到 K 份数据都当过一次 Testing Data。

每一次训练出同样数目的树 (从 Full Size Tree 到剩下叶子的树)，然后使用 Testing Data 来计算 Tree Score (使用先前已经选择的所有 α 值)。计算之后，保存在不同 α 值情况下，最低 SSR 值 。

当 Cross Validation 的步骤跑完之后，把所有已经记录相同 α 值的 SSR 全部加起来，之后选出有着最低 SSR 值的 α 值。这个 α 值就是最适合的。

	$\alpha = 0$	$\alpha = 500$	$\alpha = 1000$
1st Fold	Tree Score = 10	Tree Score = 5	Tree Score = 12
2nd Fold	Tree Score = 15	Tree Score = 8	Tree Score = 25
3rd Fold	Tree Score = 25	Tree Score = 10	Tree Score = 25
4th Fold	Tree Score = 15	Tree Score = 5	Tree Score = 30
5th Fold	Tree Score = 10	Tree Score = 6	Tree Score = 28
	Total = 75	**Total = 34**	Total = 120

上图是一个 Example，在使用 5-Fold Cross Validation，使用 Training Data Generate 了 Full Size Tree 然后 Prune 到剩下叶子，使用 3 种不同的 α 值对每棵树进行 Tree Score 的计算，然后选出所有树里面最低Tree Score 的值保存起来。当所有 Fold 已经计算完毕，把全部 Tree Score 总和，选出最低 Sum of Tree Score 的 α 值，这个 α 值就是最适合这个 Dataset 的 α 值。



选好 α 值之后，看回那组使用了所有 Dataset 来训练的所有 Trees，选出在使用了选好的 α 值的时候，有着最低 Tree Score 的那棵树，这棵树就是最适合这组 Data 的树。