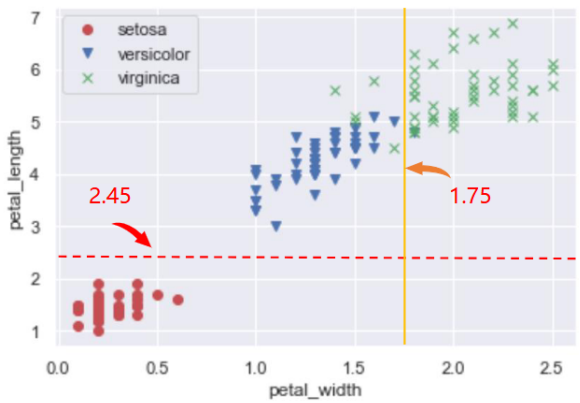
决策树叶

理解决策树本质

取iris中petal width 、petal length两特征，画出散点图。有什么发现？

iris-两个属性



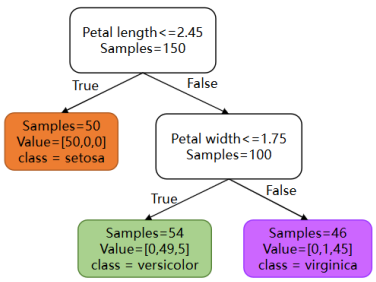
（1）观测所有样本在**petal length**上的取值，
setosa的petal length： 1.0~1.9；
非setosa的petal length： 3.0~6.9。
在1.9和3.0之间设个阈值，如(1.9+3.0)/2=2.45，就能将setosa和非setosa样本正确分开。

（2）再观察非setosa样本在**petal width**上取值，
versicolor的petal width： 1.0~1.8，有1个>1.7
virginica的petal width： 1.4~2.5，有5个<1.8。
在1.7和1.8之间设个阈值，如 1.75，也能将versicolor和virginica基本分开。（错分到versicolor中5个样本，错分到virginica中1个样本

从以上信息可以看出，而二维空间中，决策树的目的是将平面多次分割，直到分割完所有类别。

拓展到高维空间，就是对超平面进行分割。由于分割的过程是逐步的，大致可以表现为一个树状结构。

将上述决策过程用一棵树表达，就得到如下所示的结果。



预测一朵新鸢尾花的类别：

从根节点开始，先看其“petal_length≤2.45?”。如果是，则进入根的左子节点。该节点是叶节点，它不再提问，查看该节点给出的预测类别，为setosa。

如果花瓣长大于2.45cm，则进入根的右子节点，该节点不是叶节点，它提问“petal_width ≤1.75?”，如果“是”则预测为versicolor，否则预测为virginica。

一棵决策树(Decision Tree)由一系列节点和有向边组成。

· 根节点包含训练样本全集。

· 内部节点（包括根节点）：有一个关于已知特征的提问。每个内部节点所包含的样本集合根据特征检测结果被划分到子节点中。

· 叶节点：对应于决策结果。

· 从根节点到每个叶节点的路径，对应一个判定规则。

样本出发---从一个节点到下一个节点（对应一次决策的结果）

决策树训练算法

— CART算法（Classification and Regression Tree）（重点）

— ID3算法、C4.5算法、C5.0算法

训练方式

将训练集划分成两个子集，使得

划分得到的两个子集尽可能纯。然后使用上述逻辑

对子集进行划分，递归地进行，直至到达终止条件

目标：最小化加权平均不纯度（这里的m为样本数量）

$$\text{最小化}\quad J(k,t_k)=\frac{m_{\text{left}}}{m}G_{\text{left}}+\frac{m_{\text{right}}}{m}G_{\text{right}}$$

评估指标：

$$\textbf{Gini不纯度}\ (\text{Gini impurity}):\ G_i=1-\sum_{k=1}^{K}P_{i,k}{}^2$$

当一个集合所有实例均属同一类别时，它是纯的，它的基尼值为零。

熵（entropy）：$H_i = -\sum_{k=1}^{K} P_{i,k} \log_2(P_{i,k})$，其中$P_{i,k} \neq 0$

当一个集合只包含一个类的实例时，它的熵为零。当一个事件的概率为1（即100%确定）时，其对数为0，这意味着没有不确定性，因此信息量为0。相反，当概率接近0时，对数会变得非常大，表示不确定性很高。（用负对数可以去做惩罚）

递归算法找到最优子集

CART分类算法采用贪心策略，递归地选出特征⊠和阈值⊠⊠ 将当前节点的样本集划

分成两个子集，直到达到预设的最大深度，或者找不到一个划分能使不纯度降低。

决策树避免过拟合的常用策略：预剪枝（正则化）、后剪枝（常简称剪枝）

— 预剪枝：设置正则化超参数，在训练过程中降低模型的自由度。

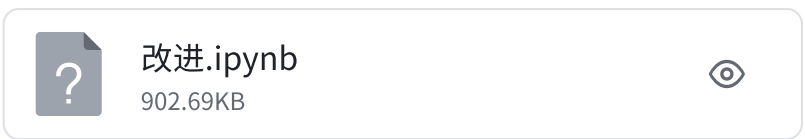
— 后剪枝：先不加约束地训练模型，然后再对不必要的节点进行剪枝(删除)。

可视化

1.调用sklearn.tree下的plot_tree()函数，绘制决策树

2.需要安装graphviz包，pip install graphviz

2 个案例



导入库--导入数据集--切分（split）--训练fit---可视化---剪枝之后fit--可视化