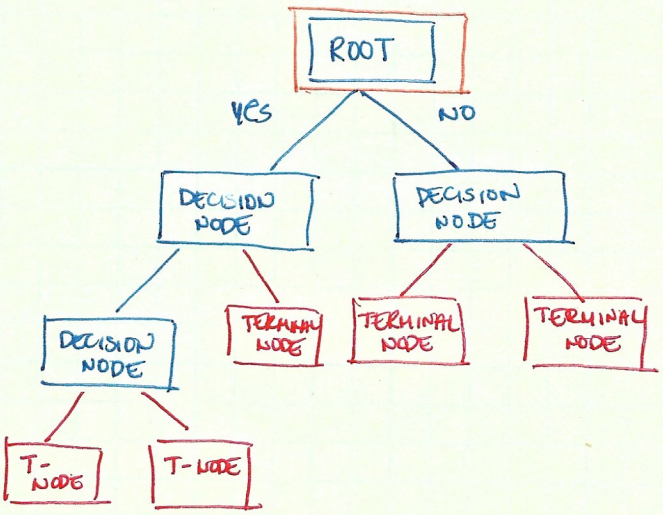
Decision trees:

- this model breaks down data by asking a series of questions



MAIN IDEA:

The root node of the tree represents the entire data set.

Leaf nodes are split until all leaves are pure (i.e all data points are same label) or cannot be split any further (in the rare case with two identical points with same label)

ALGORITHM:

- at each decision, the tree splits data along every single dimension (i.e. each feature) and computes the information gain at each split. The feature that becomes the deciding feature for the split has the largest information gain (IG) and the lowest impurity at the leaf nodes

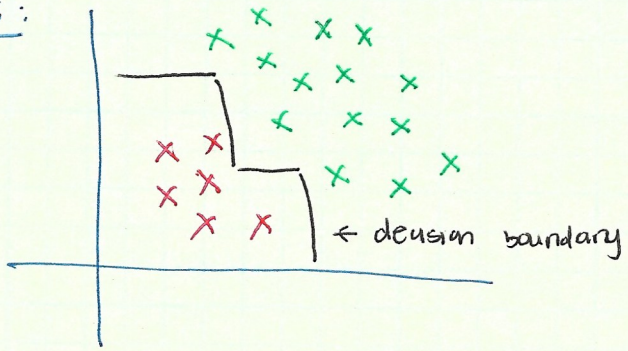
$$IG = \text{Impurity of the parent node} - \sum_{i=1}^{n} \text{impurity of child}_n$$

There are three impurity measures or splitting criteria.

1) Gini Impurity
2) Entropy
3) Classification Error

NOTES: splits are generally axis-aligned

EXAMPLE:



SUMMARY:

- Classification and regression trees are light weight classifiers
- very fast during testing
- usually not competitive in accuracy but can become very strong through bagging (Random forest) and boosting (gradient boosted trees)

Random Forest (variant of a decision tree)

- motivation stems from a bias-variance trade off

GOAL of ML MODELS

1) CAPTURE REGULARITIES IN TRAINING DATA
2) GENERALIZE TO UNSEEN DATA

SADLY, NOT POSSIBLE TO DO BOTH SIMULTANEOUSLY

- HIGH VARIANCE learning methods can represent training data well but may be overfitting and can't generalize.
- HIGH BIAS learning models produce simpler models that don't overfit but may underfit, thereby failing to catch important regularities

CASE

WE WANT TO FIND A FUNCTION $\hat{f}(x)$ THAT APPROXIMATES THE TRUE FUNCTION $f(x)$

TO ACHIEVE THIS: WE WANT TO MINIMIZE THE MEAN SQUARED ERROR

DECOMPOSE the expected error on an unseen sample x as follows:

$$E\left[(y-\hat{f}(x))^2\right] = \left(\text{Bias}[\hat{f}(x)]\right)^2 + \text{Var}[\hat{f}(x)] + \sigma^2$$

σ^2 ← irreducible error

$\left(\text{Bias}[\hat{f}(x)]\right)^2$: error caused by the simplifying assumptions built into the method

$\text{Var}[\hat{f}(x)]$: how much the $\hat{f}(x)$ will move around its mean

* *

- Random forest = ensemble of decision trees

MAIN IDEA: Average multiple (deep) decision trees that individually suffer from high variance to build a more robust model that has better generalization performance and is less susceptible to overfitting

* *

ALGORITHM:

1) Draw a random bootstrap of size n (randomly choose n ~~exam~~ samples from the training set with replacement
2) Grow a decision tree from the bootstrap sample. At each node:
   a. randomly select d features w/o replacement *
   b. split node based on feature w/ largest information gain
3) Repeat steps 1 and 2 k times
4) Aggregate the prediction by each tree to assign the class label by majority vote

d = random subset of features available

ALGORITHM

* NOTES: In most implementations, including the RandomForestClassifier implementation in scikit learn, the size of the bootstrap sample is chosen to be equal to the number of samples in the training set

(1)

(2) for the number of d features at each split, we want to choose a value that is smaller than the total number of features in the training set. A reasonable default that is used in sklearn and other implementations is $d = \sqrt{m}$, where m is the # of features in the training set

Random forests are one of the best classifiers and most popular BC

1) the only parameter we need to care about in practice is the number of trees that we choose for the random forest. Typically, more trees = better performance

2) not much data preprocessing. features can be of different scale, magnitude, or slope. highly advantageous in scenarios with heterogeneous data (i.e. medical settings where features could be things like blood pressure, age, gender, ... each of which is recorded in completely different units