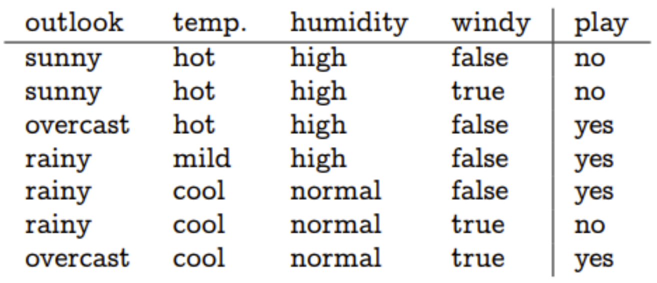
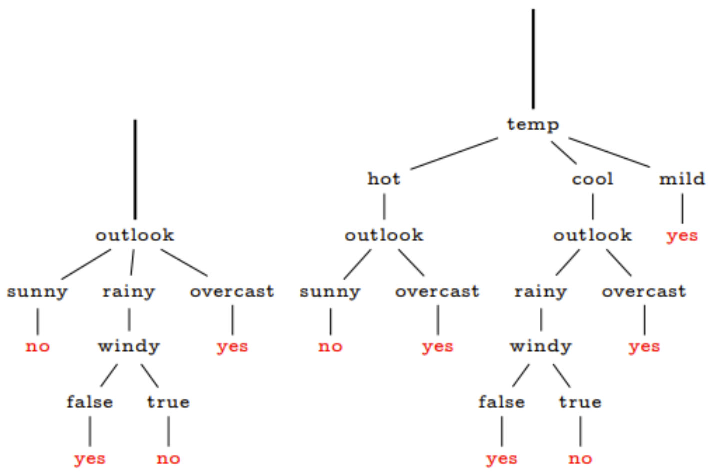
Decision trees and ensembles

Decision tree

Trees in which nodes represent features and edges represent different values for the future.



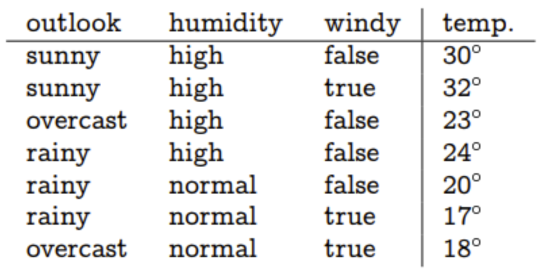
outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes



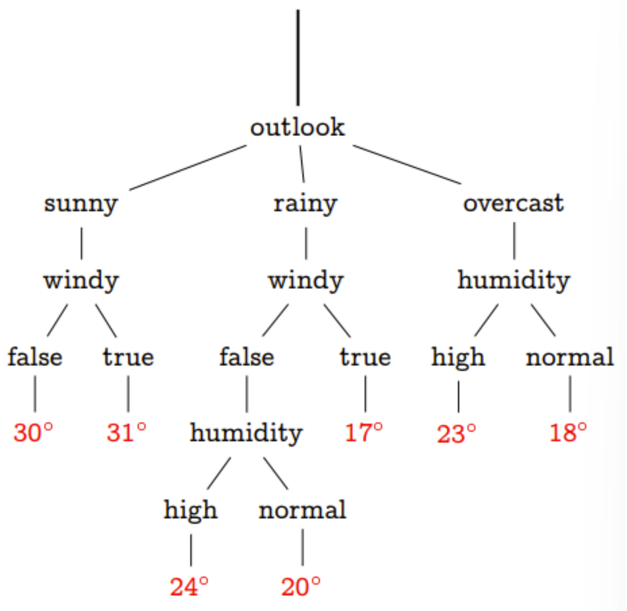
Both trees solve the problem of describing the data, which is preferred? The smaller one, but finding the smallest possible tree is shown to be NP-hard. So, in order to construct our decision tree we will use greedy methods.

Greedy methods: algorithms that make locally optimal choices at each step in the hope of finding a globally optimum solution.

The same idea applies to regression trees:



outlook	humidity	windy	temp.
sunny	high	false	30°
sunny	high	true	32°
overcast	high	false	23°
rainy	high	false	24°
rainy	normal	false	20°
rainy	normal	true	17°
overcast	normal	true	18°



OBS: Clearly, in the case of non-contradicting data we can obtain zero training error.

OBS2: Trees are highly prone to overfitting.

In order to avoid overfitting some tools can be applied:

- ▼ Control for the depth of the tree

  Grow a tree as deep as some predefined hyperparameter allows it to do.

  The depth of a leaf is defined as the number of edges to the root node. By forcing the depth of any left to be at most k for some fixed k, will force the tree from growing further.

- ▼ Control for the maximum number of splits

  Similar to the control of depth, we grow the splits until a predefined hyperparameter allows it to do.

  We force the tree to have at most k number of splits for some integer k.

- ▼ Pruning

  Grow a full tree and then prune it in order to reduce the number of unnecessary branches.

  We will use *cost complexity pruning.* We will find a subtree T such that:

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T|$$

  where alpha > 0 is a tuning parameter and |T| is the number of leaves of the tree.

    Small values for alpha led to less impact on the penalty, i.e. deeper tree.

    Optimal alpha can be found using a validation method.

Main idea of building a decision tree

Divide the feature space into J distinct non-overlapping regions, R1, …, Rj. One region corresponds to one leaf in the tree.

For every observation that falls into the region Rj, we make the same prediction, which is the mean of the responses of the observation in that region.

Classification setting

When we come to a leaf in the tree instead of using the mean of the instance as output we use mode.

In the classification setting, we can no longer use the MSE or RSS as a penalty to branch the tree, since we don't have a continuous numerical value. Therefore, the evaluation metrics for the decision tree split in classification focus on *measuring the purity or impurity of the classes within each node.*

For this, we use:

Classification error

Gini index

Cross-entropy

Classification Error

Metric calculated as the ratio of misclassified instances to the total number of instances in a node. It represents the probability of misclassification. The goal is to minimize this metric by selecting splits that minimize misclassifications.

Assume that we have some classification problem with K = 4 classes (A, B, C, D) and we want to use a decision tree to model it.

Let T be the classification tree with J regions

For each region, Em denotes the error of the mth region:

$$E_m = 1 - \max_k(\hat{p}_{mk})$$

where p*mk* is the proportion of instances in region m of the kth class.

We define the cost of the tree as the cost of each individual region

$$\sum_{m=1}^{J} E_m$$

Example: In the mth leaf there are 10 instances distributed as follows: 6A, 2B, 1C, 1D. Then Em = 1 - 0.6 = 0.4

Gini index

Assume that we have some classification problem with K = 4 classes (A, B, C, D) and we want to use the decision trees to model it.

Let T be a classification tree with J regions

For each region let Gm denote the error of the mth region:

$$G_m = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) = 1 - \sum_{k=1}^{K} \hat{p}_{mk}^2.$$

We define the cost of the tree as the cost of each individual region

$$\sum_{m=1}^{J} G_m$$

Example: In the mth leaf there are 10 instances distributed as follows: 6A; 2B; 1C; 1D. Then:

Gm = 0.6*0.4 + 0.2*0.8 + 0.1*0.9 + 0.1*0.9 = 1 - (0.6^2 + 0.2^2 + 0.1^2 + 0.1^2) = 0.58

Cross-entropy

Assume that we have some classification problem with K = 4 classes (A, B, C, D) and we want to use the decision trees to model it.

Let T be a classification tree with J regions

For each region let Dm denote the error of the mth region:

$$D_m = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk}).$$

We define the cost of the tree as the cost of each individual region

$$\sum_{m=1}^{J} D_m$$

Example: In the mth leaf there are 10 instances distributed as follows: 6A; 2B; 1C; 1D. Then:

Dm = (0.6*log(0:6) + 0.2*log(0.2) + 0.1*log(0.1) + 0.1*log(0.1)) = 1.5710

Em is not as sensitive for node purity ⇒ use it to measure performance not to grow the tree

Purity gain

Let vj denote the elements in the jth leaf of a split at some part in the tree.

We define the purity gain of that split as:

$$\Delta = I(\text{parent}) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

where I is our impurity measure either E (classification error), G (gini index) D (cross-entropy), and N(vj) is the number of elements in the node vj, and N is the number of elements in the parent node.

Advantages of decision trees

Simple to understand and to interpret. Can be visualized

Can handle both categorical and numerical data, and this often requires little data preparation

The cost of prediction is logarithmic in the number of data points used to train the tree ⇒ cheaper than KNN, but more expensive than logistic regression

Can handle multiclass problems without having to resort to other techniques

Disadvantages of decision trees

Non-robust, a small change in data might lead to a large change in the model

Can easily overfit the data

As it is NP-hard to find the optimal tree, using greedy methods only provides us with locally optimal improvements.

Ensemble methods

The ensemble method consists in combine models to increase the accuracy of the results significantly, one famous example is Random Forest, which runs multiple decision trees using different samples of the training set for each run and then gets the majority of votes for the determined class to be the final classification.

Bagging and pasting

Bagging creates multiple random samples with replacements from the original training set. Each sample is used to train a separate model. In the end, predictions from all models are combined, but taking the majority vote (for classification) or the average (for regression) to make the final prediction.

Pasting is similar to bagging, but it doesn’t use replacement from the original training set.