Intro and Terminology

Decision trees

- Are a structured sequence of questions
- That recursively partitions the universe of possible examples
- May be used for both Classification and Regression tasks

We will illustrate the terminology by referring to a [Decision Tree for the Titanic Survival problem (Decision_Trees.ipynb#Example:-Decision-Tree-for-Titanic-Survival)](Decision_Trees.ipynb#Example:-Decision-Tree-for-Titanic-Survival)

Training

As you've seen, a Decision Tree is really nothing more than a structured series of questions.

Thus, the parameters Θ, which will be discovered by training (as usual), consists of

- A specification of the structure (parent/child relationships)
- An encoding of the "best" question to ask at each node
- A class prediction on the leaf nodes

High level view

We start with a high level introduction to the algorithm that constructs a decision tree.

Let's visit the notebook section on training (Decision_Trees.ipynb#Training:-a-first-look-at-the-algorithm)

Deeper view: creating the test

At this point we have a rough idea of the algorithm.

There are still unanswered questions

- How to choose the test/question at each node
- Is there a better point at which to stop ?

Let's visit the notebook section [Training: a deeper look (Decision_Trees.ipynb#Training:-a-deeper-look-at-the-algorithm)](Decision_Trees.ipynb#Training:-a-deeper-look-at-the-algorithm)

Decision Trees for Regression

It might seem surprising to use a Classification model (discrete targets) for a Regression task (continuous targets).

The notebook section [Decision Tree Regression (Decision_Trees.ipynb#Decision-Tree-Regression)](Decision_Trees.ipynb#Decision-Tree-Regression) shows just how to do that.

Overfitting

If we continue the training algorithm to it conclusion

- We wind up with leaf nodes that are pure (all examples in same category)
- Potentially have leaf nodes with small number of examples

Thus, the danger of overfitting (poor out of sample generalization) is present.

Let's go to the notebook for an example (Decision_Trees.ipynb#Overfitting-example)

Hyper parameters

We can control for the possibility of overfitting

- With hyper-parameters to control various aspects of training

There are other hyper-parameters than affect performance.

These hyper-parameters may be adjusted as part of Fine Tuning, or earlier.

Let's go to the notebook for a quick introduction to hyper-parameters (Hyper-parameters-for-Decision-Trees)

The Good and the Bad of Decision Trees

The good

- Very fast prediction
    - Just a small number of comparisons
- Relatively interpretable
    - Can *explain* why prediction was made
    - Sequence of tests lead to prediction

The bad

- Prone to overfitting
- Interpretable ?
    - Does the answer to a *long* sequence of tests really clarify the choice ?
    - Explainable rather than understandable
- More procedural/less mathematical
    - Greedy tests: locally optimal but perhaps not globally

Categorical variables: one more time

In the Titanic Survival example used in this module

- We treated Pclass (Passenger class) as a categorical rather than a numeric feature

The test applied by a node in a Decision Tree highlights the difference

- Consider Titanic Passenger Class `Pclass` $\in \{1, 2, 3\}$
- As a numeric, `Pclass` implies an ordering (magnitude doesn't matter for Decision Tree split)
  - The test: $\neg(\text{PClass} \le 2)$
  - Is True for any example where $\text{PClass} \in \{3\}$
- As a categorical
  - The test: $\neg\text{Is}_{Pclass=2}$
  - Is True for any example where $\text{PClass} \in \{1, 3\}$

Not the same !

The latter is probably what we had in mind, but `Pclass` as numeric does not give us this result.

- Arbitrary encodings are dangerous ! Treat variables with discrete values as categorical.

In [4]:

```
print("Done")
```

```
Done
```