III. Decision Trees

- Decision Trees Basics
- Impurity Functions
- Decision Tree Algorithms
- Decision Tree Pruning
Decision Trees Basics
Classification Problems with Nominal Features

Setting:

- $X$ is a set of feature vectors.
- $C$ is a set of classes.
- $c : X \to C$ is the (unknown) ideal classifier for $X$.
- $D = \{(x_1, c(x_1)), \ldots, (x_n, c(x_n))\} \subseteq X \times C$ is a set of examples.

Todo:

- Approximate $c(x)$, which is implicitly given via $D$, with a decision tree.
Decision Trees Basics

Decision Tree for the Concept “EnjoySport”

<table>
<thead>
<tr>
<th>Example</th>
<th>Sky</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Water</th>
<th>Forecast</th>
<th>EnjoySport</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>sunny</td>
<td>warm</td>
<td>normal</td>
<td>strong</td>
<td>warm</td>
<td>same</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>sunny</td>
<td>warm</td>
<td>high</td>
<td>strong</td>
<td>warm</td>
<td>same</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>rainy</td>
<td>cold</td>
<td>high</td>
<td>strong</td>
<td>warm</td>
<td>change</td>
<td>no</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Sky:** sunny, rainy, cloudy
- **Temperature:** cold, warm
- **Humidity:** normal, high
- **Wind:** strong
- **Water:** warm
- **Forecast:** same, change
- **EnjoySport:** yes, no

Diagram: Decision Tree for EnjoySport with features Sky, Temperature, and Wind.
Decision Trees Basics

Decision Tree for the Concept “EnjoySport”

<table>
<thead>
<tr>
<th>Example</th>
<th>Sky</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Water</th>
<th>Forecast</th>
<th>EnjoySport</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>sunny</td>
<td>warm</td>
<td>normal</td>
<td>strong</td>
<td>warm</td>
<td>same</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>sunny</td>
<td>warm</td>
<td>high</td>
<td>strong</td>
<td>warm</td>
<td>same</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>rainy</td>
<td>cold</td>
<td>high</td>
<td>strong</td>
<td>warm</td>
<td>change</td>
<td>no</td>
</tr>
</tbody>
</table>

Splitting of $X$ at the root node:

$$X = \{x \in X : x_{\text{Sky}} = \text{sunny}\} \cup \{x \in X : x_{\text{Sky}} = \text{cloudy}\} \cup \{x \in X : x_{\text{Sky}} = \text{rainy}\}$$
Definition 1 (Splitting)

Let $X$ be a set of feature vectors and $D$ a set of examples. A splitting of $X$ is a decomposition of $X$ into mutually exclusive subsets $X_1, \ldots, X_s$. I.e.,

$$X = X_1 \cup \ldots \cup X_s \text{ with } X_j \neq \emptyset \text{ and } X_j \cap X_{j'} = \emptyset, \text{ where } j, j' \in \{1, \ldots, s\}, j \neq j'.$$

A splitting $X_1, \ldots, X_s$ of $X$ induces a splitting $D_1, \ldots, D_s$ of $D$, where $D_j$, $j = 1, \ldots, s$, is defined as $\{(x, c(x)) \in D \mid x \in X_j\}$. 
Definition 1 (Splitting)

Let \( X \) be a set of feature vectors and \( D \) a set of examples. A splitting of \( X \) is a decomposition of \( X \) into mutually exclusive subsets \( X_1, \ldots, X_s \). I.e.,

\[
X = X_1 \cup \ldots \cup X_s \text{ with } X_j \neq \emptyset \text{ and } X_j \cap X_{j'} = \emptyset, \text{ where } j, j' \in \{1, \ldots, s\}, j \neq j'.
\]

A splitting \( X_1, \ldots, X_s \) of \( X \) induces a splitting \( D_1, \ldots, D_s \) of \( D \), where \( D_j, j = 1, \ldots, s \), is defined as \( \{(x, c(x)) \in D \mid x \in X_j\} \).

A splitting depends on the measurement scale of a feature:

\[
x = \begin{pmatrix}
x_1 \\
x_2 \\
x_3 \\
\vdots \\
x_p
\end{pmatrix}
\]

\( x \bigg|_A = x_3 \)
Decision Trees Basics

**Definition 1 (Splitting)**

Let $X$ be a set of feature vectors and $D$ a set of examples. A splitting of $X$ is a decomposition of $X$ into mutually exclusive subsets $X_1, \ldots, X_s$. I.e.,

$$X = X_1 \cup \ldots \cup X_s$$

with $X_j \neq \emptyset$ and $X_j \cap X_{j'} = \emptyset$, where $j, j' \in \{1, \ldots, s\}$, $j \neq j'$.

A splitting $X_1, \ldots, X_s$ of $X$ induces a splitting $D_1, \ldots, D_s$ of $D$, where $D_j$, $j = 1, \ldots, s$, is defined as

$$\{(x, c(x)) \in D \mid x \in X_j\}.$$ 

A splitting depends on the **measurement scale** of a feature:

1. An $m$-ary splitting induced by a (nominal) feature $A$ with finite domain:

   $$A = \{a_1, \ldots, a_m\} : X = \{x \in X : x|_A = a_1\} \cup \ldots \cup \{x \in X : x|_A = a_m\}$$
Decision Trees Basics

**Definition 1 (Splitting)**

Let $X$ be a set of feature vectors and $D$ a set of examples. A splitting of $X$ is a decomposition of $X$ into mutually exclusive subsets $X_1, \ldots, X_s$. I.e.,

$$X = X_1 \cup \ldots \cup X_s$$

with $X_j \neq \emptyset$ and $X_j \cap X_{j'} = \emptyset$, where $j, j' \in \{1, \ldots, s\}, j \neq j'$.

A splitting $X_1, \ldots, X_s$ of $X$ induces a splitting $D_1, \ldots, D_s$ of $D$, where $D_j$, $j = 1, \ldots, s$, is defined as

$$\{(x, c(x)) \in D \mid x \in X_j\}.$$

A splitting depends on the **measurement scale** of a feature:

1. **$m$-ary splitting** induced by a (nominal) feature $A$ with finite domain:
   $$A = \{a_1, \ldots, a_m\} : \quad X = \{x \in X : x|_A = a_1\} \cup \ldots \cup \{x \in X : x|_A = a_m\}$$

2. **Binary splitting** induced by a (nominal) feature $A$:
   $$A' \subset A : \quad X = \{x \in X : x|_A \in A'\} \cup \{x \in X : x|_A \notin A'\}$$

3. **Binary splitting** induced by an ordinal feature $A$:
   $$v \in \text{dom}(A) : \quad X = \{x \in X : x|_A \geq v\} \cup \{x \in X : x|_A < v\}$$
Remarks:

- The syntax $x|_A$ denotes the projection operator, which returns that vector component (dimension) of $x = (x_1, \ldots, x_p)$ that is associated with the feature $A$. Without loss of generality this projection can be presumed being unique.

- A splitting of $X$ into two disjoint, non-empty subsets is called a binary splitting.

- We consider only splittings of $X$ that are induced by a splitting of a single feature $A$ of $X$. Keyword: monothetic splitting.
  By contrast, a polythetic splitting considers several features at the same time.
Definition 2 (Decision Tree)

Let $X$ be a set of features and $C$ a set of classes. A decision tree $T$ for $X$ and $C$ is a finite tree with a distinguished root node. A non-leaf node $t$ of $T$ has assigned (1) a set $X(t) \subseteq X$, (2) a splitting of $X(t)$, and (3) a one-to-one mapping of the subsets of the splitting to its successors.

$X(t) = X \text{ iff } t \text{ is root node.}$

A leaf node of $T$ has assigned a class from $C$. 

Definition 2 (Decision Tree)

Let $X$ be a set of features and $C$ a set of classes. A decision tree $T$ for $X$ and $C$ is a finite tree with a distinguished root node. A non-leaf node $t$ of $T$ has assigned (1) a set $X(t) \subseteq X$, (2) a splitting of $X(t)$, and (3) a one-to-one mapping of the subsets of the splitting to its successors.

$X(t) = X$ iff $t$ is root node. A leaf node of $T$ has assigned a class from $C$.

Classification of some $x \in X$ given a decision tree $T$:

1. Find the root node $t$ of $T$.

2. If $t$ is a non-leaf node, find among its successors that node $t'$ whose subset of the splitting of $X(t)$ contains $x$. Repeat this step with $t = t'$.

3. If $t$ is a leaf node, label $x$ with the respective class.

→ The set of possible decision trees forms the hypothesis space $H$. 
Remarks:

- The classification of an \( x \in X \) determines a unique path from the root node of \( T \) to some leaf node of \( T \).
- At each non-leaf node a particular feature of \( x \) is evaluated in order to find the next node along with a possible next feature to be analyzed.
- Each path from the root node to some leaf node corresponds to a conjunction of feature values, which are successively tested. This test can be formulated as a decision rule.

**Example:**

\[
\text{IF Sky}=\text{rainy AND Wind}=\text{light THEN EnjoySport}=\text{yes}
\]

If all tests in \( T \) are of the kind shown in the example, namely, an equality test regarding a feature value, all feature domains must be finite.

- If in all non-leaf nodes of \( T \) only one feature is evaluated at a time, \( T \) is called a *monothetic* decision tree. Examples for *polythetic* decision trees are the so-called oblique decision trees.
- Decision trees became popular in 1986, with the introduction of the ID3 Algorithm by J. R. Quinlan.
Decision Trees Basics

Notation

Let $T$ be a decision tree for $X$ and $C$, let $D$ be a set of examples, and let $t$ be a node of $T$. Then we agree on the following notation:

- $X(t)$ denotes the subset of $X$ that is represented by $t$.
- $D(t)$ denotes the subset of the example set $D$ that is represented by $t$, where $D(t) = \{(x, c(x)) \in D \mid x \in X(t)\}$. (see the splitting definition)

Illustration:
Remarks:

- The set $X(t)$ is comprised of those members $x$ of $X$ that are filtered by a path from the root node of $T$ to the node $t$.

- $leaves(T)$ denotes the set of all leaf nodes of $T$.

- A single node $t$ of a decision tree $T$, and hence $T$ itself, encode a piecewise constant function. This way, $t$ as well as $T$ can form complex, non-linear classifiers. The functions encoded by $t$ and $T$ differ in the number of evaluated features of $x$, which is one for $t$ and the tree height for $T$.

- In the following we will use the symbols “$t$” and “$T$” to denote also the classifiers that are encoded by a node $t$ and a tree $T$ respectively:

$$
t, T : X \rightarrow C \quad \text{(instead of } y_t, y_T : X \rightarrow C)\)$$
Decision Trees Basics

Algorithm Template: Construction

Algorithm: \textit{DT-construct} Decision Tree Construction

Input: \( D \) (Sub)set of examples.

Output: \( t \) Root node of a decision (sub)tree.

\( \text{DT-construct}(D) \)

1. \( t = \text{newNode}() \)
   \( \quad \text{label}(t) = \text{representativeClass}(D) \)

2. \textbf{IF} \( \text{impure}(D) \)
   \textbf{THEN} \( \text{criterion} = \text{splitCriterion}(D) \)
   \textbf{ELSE} \( \text{return}(t) \)

3. \( \{D_1, \ldots, D_s\} = \text{decompose}(D, \text{criterion}) \)

4. \textbf{FOREACH} \( D' \ \text{IN} \ \{D_1, \ldots, D_s\} \ \text{DO} \)
   \( \quad \text{addSuccessor}(t, \text{DT-construct}(D')) \)
   \textbf{ENDDO}

5. \( \text{return}(t) \)
Decision Trees Basics
Algorithm Template: Classification

Algorithm: \( DT\text{-classify} \) Decision Tree Classification

Input:
- \( x \) Feature vector.
- \( t \) Root node of a decision (sub)tree.

Output:
- \( y(x) \) Class of feature vector \( x \) in the decision (sub)tree below \( t \).

\( DT\text{-classify}(x, t) \)

1. \textbf{IF} \ isLeafNode\( (t) \)
   \textbf{THEN} \ return\( (\text{label}(t)) \)
   \textbf{ELSE} \ return\( (DT\text{-classify}(x, \text{splitSuccessor}(t, x))) \)
Remarks:

- Since \textit{DT-construct} assigns to each node of a decision tree \( T \) a class, each subtree of \( T \) (as well as each pruned version of a subtree of \( T \)) represents a valid decision tree on its own.

- Functions of \textit{DT-construct}:
  - \texttt{representativeClass}(D)
    Returns a representative class for the example set \( D \). Note that, due to pruning, each node may become a leaf node.
  - \texttt{impure}(D)
    Evaluates the (im)purity of a set \( D \) of examples.
  - \texttt{splitCriterion}(D)
    Returns a split criterion for \( X(t) \) based on the examples in \( D(t) \).
  - \texttt{decompose}(D, criterion)
    Returns a \texttt{splitting} of \( D \) according to \texttt{criterion}.
  - \texttt{addSuccessor}(t, t')
    Inserts the successor \( t' \) for node \( t \).

- Functions of \textit{DT-classify}:
  - \texttt{isLeafNode}(t)
    Tests whether \( t \) is a leaf node.
  - \texttt{splitSuccessor}(t, x)
    Returns the (unique) successor \( t' \) of \( t \) for which \( x \in X(t') \) holds.
Problem characteristics that may suggest a decision tree classifier:

- the objects can be described by feature-value combinations
- the domain and range of the target function are discrete
- hypotheses can be represented in disjunctive normal form
- the training set contains noise

Selected application areas:

- medical diagnosis
- fault detection in technical systems
- risk analysis for credit approval
- basic scheduling tasks such as calendar management
- classification of design flaws in software engineering
Decision Trees Basics
On the Construction of Decision Trees

- How to exploit an example set both efficiently and effectively?
- According to what rationale should a node become a leaf node?
- How to assign a class for nodes of impure example sets?
- How to evaluate decision tree performance?
Decision Trees Basics

Evaluation of Decision Trees

1. Size

2. Classification error
Decision Trees Basics
Evaluation of Decision Trees

1. Size

Among those theories that can explain an observation, the most simple one is to be preferred (*Ockham’s Razor*):

\[
\textit{Entia non sunt multiplicanda sine necessitate.}
\]

[Johannes Clauberg 1622-1665]

Here: among all decision trees of minimum classification error we choose the one of smallest size.

2. Classification error

Quantifies the rigor according to which a class label is assigned to \( x \) in a leaf node of \( T \), based on the examples in \( D \).  

If all leaf nodes of a decision tree \( T \) represent a single example of \( D \), the classification error of \( T \) with respect to \( D \) is zero.
Decision Trees Basics

Evaluation of Decision Trees: Size

- Leaf node number
- Tree height
- External path length
- Weighted external path length
Decision Trees Basics

Evaluation of Decision Trees: Size

- **Leaf node number**
  The leaf node number corresponds to number of rules that are encoded in a decision tree.

- **Tree height**
  The tree height corresponds to the maximum rule length and bounds the number of premises to be evaluated to reach a class decision.

- **External path length**
  The external path length totals the lengths of all paths from the root of a tree to its leaf nodes. It corresponds to the space to store all rules that are encoded in a decision tree.

- **Weighted external path length**
  The weighted external path length is defined as the external path length with each length value weighted by the number of examples in $D$ that are classified by this path.
Example set $D$ for mushrooms, implicitly defining a feature space $X$ over the three dimensions color, size, and points:

<table>
<thead>
<tr>
<th></th>
<th>Color</th>
<th>Size</th>
<th>Points</th>
<th>Edibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>red</td>
<td>small</td>
<td>yes</td>
<td>toxic</td>
</tr>
<tr>
<td>2</td>
<td>brown</td>
<td>small</td>
<td>no</td>
<td>edible</td>
</tr>
<tr>
<td>3</td>
<td>brown</td>
<td>large</td>
<td>yes</td>
<td>edible</td>
</tr>
<tr>
<td>4</td>
<td>green</td>
<td>small</td>
<td>no</td>
<td>edible</td>
</tr>
<tr>
<td>5</td>
<td>red</td>
<td>large</td>
<td>no</td>
<td>edible</td>
</tr>
</tbody>
</table>
The following trees correctly classify all examples in $D$:

(a) 
```
feature: Color
  red
    feature: Size
      small
        label: toxic
      large
        label: edible
  green
    feature: Size
      small
        label: toxic
      large
        label: edible
brown
```

(b) 
```
feature: Size
  small
    feature: Points
      yes
        label: toxic
      no
        label: edible
  large
    label: edible
```

<table>
<thead>
<tr>
<th>Criterion</th>
<th>(a)</th>
<th>(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leaf node number</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Tree height</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>External path length</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>
The following trees correctly classify all examples in $D$:

(a) feature: Color
   - red
     - small
       label: toxic 1x
       label: edible 1x
     - large
       label: edible 1x
   - green
     - small
       label: toxic 1x
       label: edible 1x
     - large
       label: edible 1x
   - brown

(b) feature: Size
   - small
     label: edible 1x
   - large
     label: edible 2x

Criterion (a) (b)

<table>
<thead>
<tr>
<th>Criterion</th>
<th>(a)</th>
<th>(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leaf node number</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Tree height</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>External path length</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Weighted external path length</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
Decision Trees Basics
Evaluation of Decision Trees: Size (continued)

Theorem 3 (External Path Length Bound)
The problem to decide for a set of examples $D$ whether or not a decision tree exists whose external path length is bounded by $b$, is NP-complete.
Decision Trees Basics
Evaluation of Decision Trees: Classification Error

Given a decision tree $T$, a set of examples $D$, and a node $t$ of $T$ that represents the example subset $D(t) \subseteq D$. Then, the class that is assigned to $t$, $label(t)$, is defined as follows [Illustration]:

$$label(t) = \arg\max_{c \in C} \frac{|\{(x, c(x)) \in D(t) : c(x) = c\}|}{|D(t)|}$$
### Decision Trees Basics

#### Evaluation of Decision Trees: Classification Error

Given a decision tree \( T \), a set of examples \( D \), and a node \( t \) of \( T \) that represents the example subset \( D(t) \subseteq D \). Then, the class that is assigned to \( t \), \( \text{label}(t) \), is defined as follows [Illustration]:

\[
\text{label}(t) = \arg\max_{c \in C} \frac{|\{(x, c(x)) \in D(t) : c(x) = c\}|}{|D(t)|}
\]

---

**Misclassification rate** of node classifier \( t \) wrt. \( D(t) \):

\[
\text{Err}(t, D(t)) = \frac{|\{(x, c(x)) \in D(t) : c(x) \neq \text{label}(t)\}|}{|D(t)|} = 1 - \max_{c \in C} \frac{|\{(x, c(x)) \in D(t) : c(x) = c\}|}{|D(t)|}
\]
Decision Trees Basics

Evaluation of Decision Trees: Classification Error

Given a decision tree $T$, a set of examples $D$, and a node $t$ of $T$ that represents the example subset $D(t) \subseteq D$. Then, the class that is assigned to $t$, $\text{label}(t)$, is defined as follows [Illustration]:

$$\text{label}(t) = \arg\max_{c \in C} \frac{|\{(x, c(x)) \in D(t) : c(x) = c\}|}{|D(t)|}$$

Misclassification rate of node classifier $t$ wrt. $D(t)$:

$$\text{Err}(t, D(t)) = \frac{|\{(x, c(x)) \in D(t) : c(x) \neq \text{label}(t)\}|}{|D(t)|} = 1 - \max_{c \in C} \frac{|\{(x, c(x)) \in D(t) : c(x) = c\}|}{|D(t)|}$$

Misclassification rate of decision tree classifier $T$ wrt. $D$:

$$\text{Err}(T, D) = \sum_{t \in \text{leaves}(T)} \frac{|D(t)|}{|D|} \cdot \text{Err}(t, D(t))$$
Remarks:

- Observe the difference between \(\max(f)\) and \(\arg\max(f)\). Both expressions maximize \(f\), but the former returns the maximum \(f\)-value (the image) while the latter returns the argument (the preimage) for which \(f\) becomes maximum:

\[
\begin{align*}
\max_{c \in C}(f(c)) &= \max\{f(c) \mid c \in C\} \\
\arg\max_{c \in C}(f(c)) &= c^* \quad \Rightarrow \quad f(c^*) = \max_{c \in C}(f(c))
\end{align*}
\]

- The classifiers \(t\) and \(T\) may not have been constructed using \(D(t)\) as training data. I.e., the example set \(D(t)\) is in the role of a holdout test set.

- The true misclassification rate \(Err^*(T)\) is based on a probability measure \(P\) on \(X \times C\) (and not on relative frequencies). For a node \(t\) of \(T\) this probability becomes minimum iff:

\[
\text{label}(t) = \arg\max_{c \in C} P(c \mid X(t))
\]

- If \(D\) has been used as training set, a reliable interpretation of the (training) error \(Err(T, D)\) in terms of \(Err^*(T)\) requires the Inductive Learning Hypothesis to hold. This implies that the distribution of \(C\) over the training set \(D\) corresponds to the distribution of \(C\) over \(X\).
Decision Trees Basics
Evaluation of Decision Trees: Misclassification Costs

Given a decision tree $T$, a set of examples $D$, and a node $t$ of $T$ that represents the example subset $D(t) \subseteq D$. In addition, there is a cost measure for misclassification. Then, the class that is assigned to $t$, $label(t)$, is defined as follows:

$$label(t) = \arg\min_{c' \in C} \sum_{c \in C} \frac{|\{(x, c(x)) \in D(t) : c(x) = c\}|}{|D(t)|} \cdot cost(c' | c)$$
Decision Trees Basics

Evaluation of Decision Trees: Misclassification Costs

Given a decision tree $T$, a set of examples $D$, and a node $t$ of $T$ that represents the example subset $D(t) \subseteq D$. In addition, there is a cost measure for misclassification. Then, the class that is assigned to $t$, $\text{label}(t)$, is defined as follows:

$$
\text{label}(t) = \arg\min_{c' \in C} \sum_{c \in C} \frac{|\{(x, c(x)) \in D(t) : c(x) = c\}|}{|D(t)|} \cdot \text{cost}(c' | c)
$$

**Misclassification costs** of node classifier $t$ wrt. $D(t)$:

$$
\text{Err}_{\text{cost}}(t, D(t)) = \frac{1}{|D(t)|} \cdot \sum_{(x, c(x)) \in D(t)} \text{cost}(\text{label}(t) | c(x)) = \min_{c' \in C} \sum_{c \in C} \frac{|\{(x, c(x)) \in D(t) : c(x) = c\}|}{|D(t)|} \cdot \text{cost}(c' | c)
$$
Decision Trees Basics

Evaluation of Decision Trees: Misclassification Costs

Given a decision tree $T$, a set of examples $D$, and a node $t$ of $T$ that represents the example subset $D(t) \subseteq D$. In addition, there is a cost measure for misclassification. Then, the class that is assigned to $t$, $\text{label}(t)$, is defined as follows:

$$
\text{label}(t) = \arg\min_{c' \in C} \sum_{c \in C} \frac{|\{(x, c(x)) \in D(t) : c(x) = c\}|}{|D(t)|} \cdot \text{cost}(c' \mid c)
$$

Misclassification costs of node classifier $t$ wrt. $D(t)$:

$$
\text{Err}_{\text{cost}}(t, D(t)) = \frac{1}{|D(t)|} \cdot \sum_{(x, c(x)) \in D(t)} \text{cost}(\text{label}(t) \mid c(x)) = \min_{c' \in C} \sum_{c \in C} \frac{|\{(x, c(x)) \in D(t) : c(x) = c\}|}{|D(t)|} \cdot \text{cost}(c' \mid c)
$$

Misclassification costs of decision tree classifier $T$ wrt. $D$:

$$
\text{Err}_{\text{cost}}(T, D) = \sum_{t \in \text{leaves}(T)} \frac{|D(t)|}{|D|} \cdot \text{Err}_{\text{cost}}(t, D(t))
$$
Remarks:

- Again, observe the difference between \( \min(f) \) and \( \text{argmin}(f) \). Both expressions minimize \( f \), but the former returns the minimum \( f \)-value (the image) while the latter returns the argument (the preimage) for which \( f \) becomes minimum.