Nikita Kazeev

Decision Trees

2021

MLHEP 2021

HIGHER SCHOOL OF ECONOMICS
NATIONAL RESEARCH UNIVERSITY

LAMBDA • HSE

Yandex

SCHOOL OF DATA ANALYSIS

EPFL

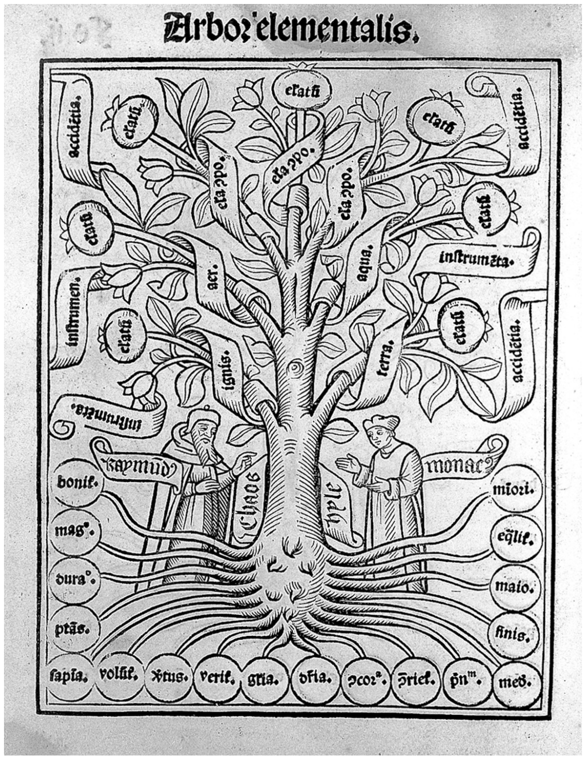
SIT
Schaffhausen Institute of Technology

Lecture overview

After the lecture, you will be able to:

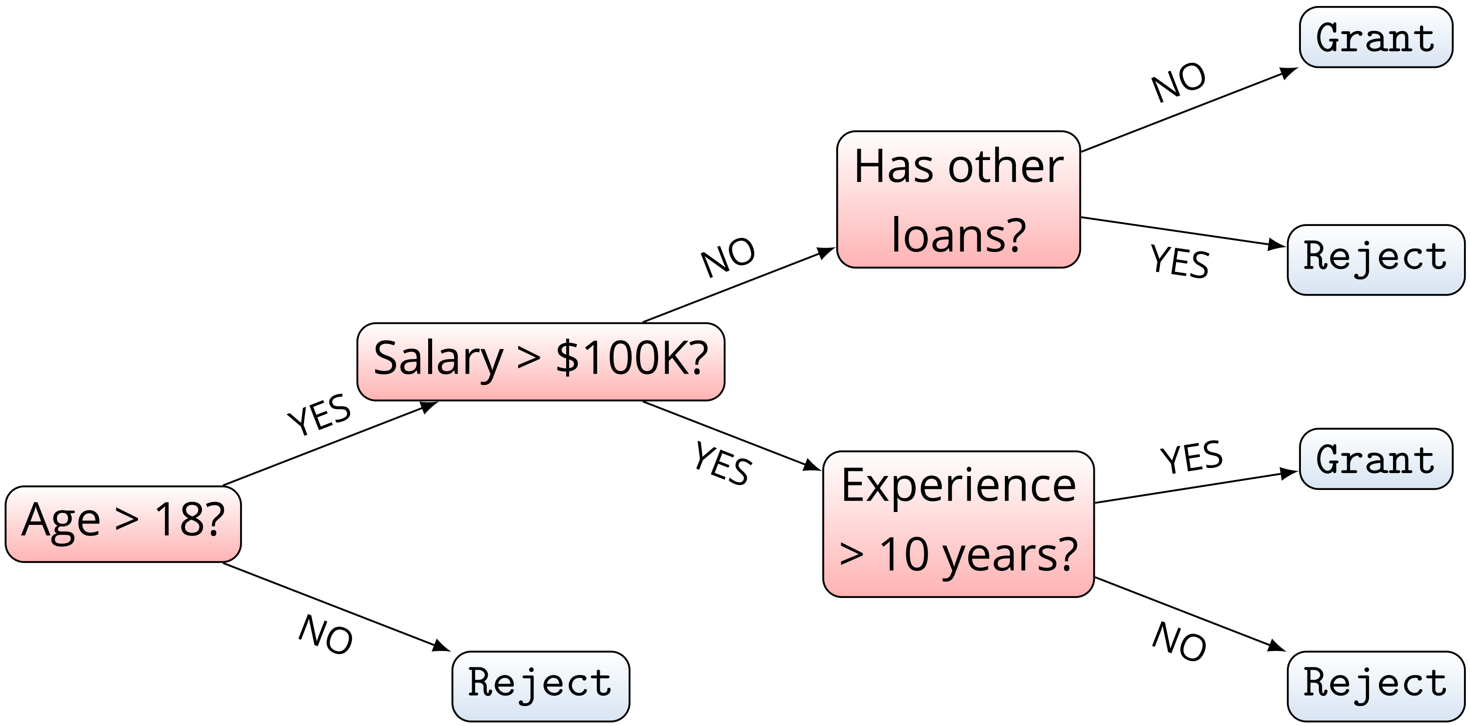
- Use the decision tree algorithm for classification and regression;
- Tune the algorithm parameters improve its performance.

Arbor Porphyrii



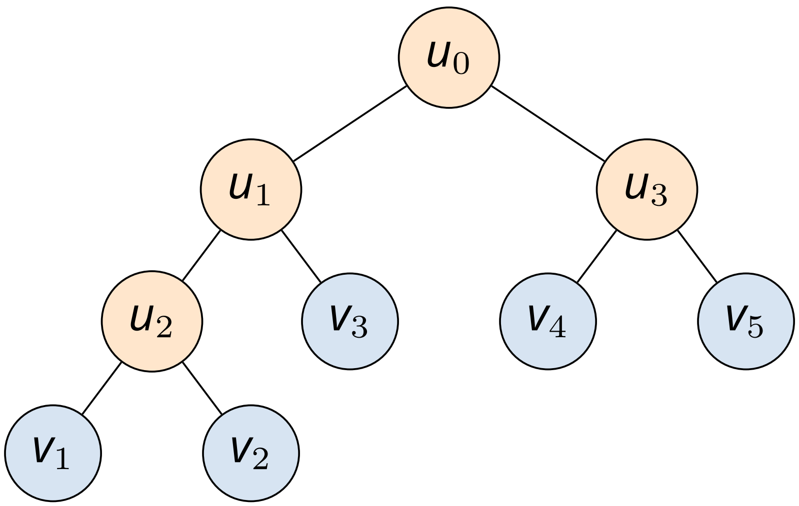
Picture credit: R. Lull, showing the Arbor Elementalis to monk

Decision making at a bank



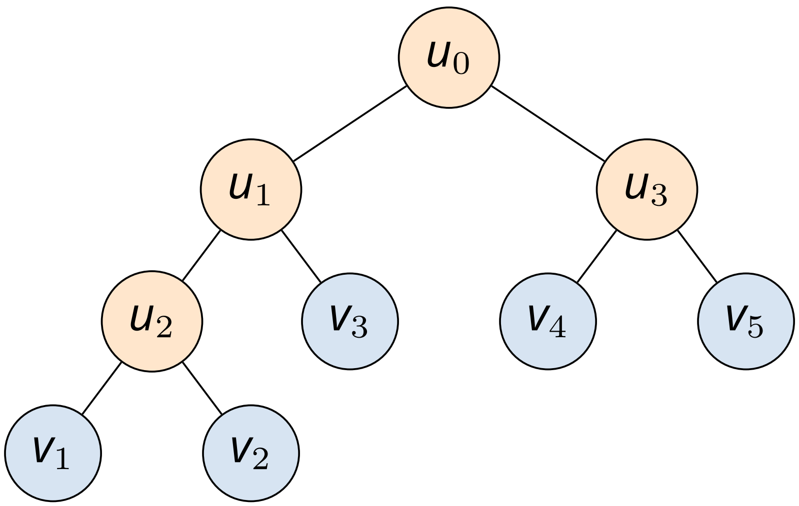
Decision tree formalism

- Decision tree is a binary tree V
- Internal nodes $u \in V$: *predicates* $\beta_u : \mathcal{X} \to \{0, 1\}$
- Leaves $v \in V$: *predictions* $\beta_v : \mathcal{X} \to \mathcal{Y}$



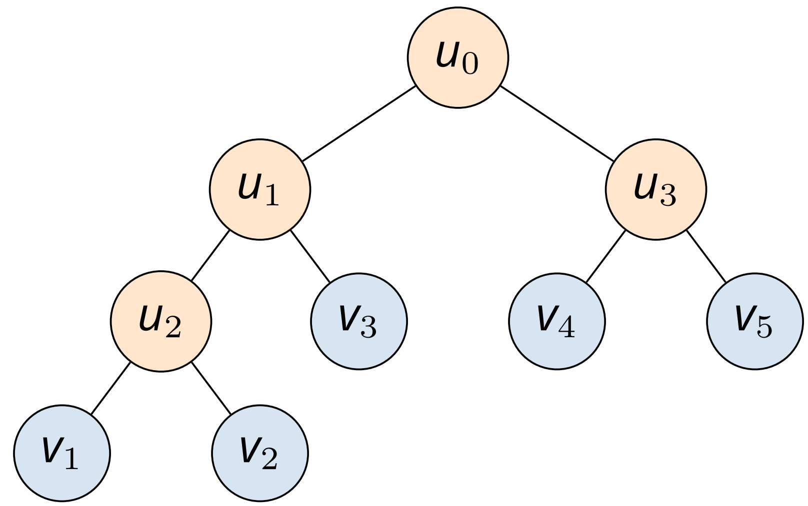
Decision tree formalism

- Decision tree is a binary tree V
- Internal nodes $u \in V$: *predicates* $\beta_u : \mathcal{X} \to \{0, 1\}$
- Leaves $v \in V$: *predictions* $\beta_v : \mathcal{X} \to \mathcal{Y}$
- Algorithm $h(\mathbf{x})$ starts at $u = u_0$
  - Compute $b = \beta_u(\mathbf{x})$
  - If u is a leaf, return b
  - If $b = 0$, $u \leftarrow \mathrm{LeftChild}(u)$
  - If $b = 1$, $u \leftarrow \mathrm{RightChild}(u)$

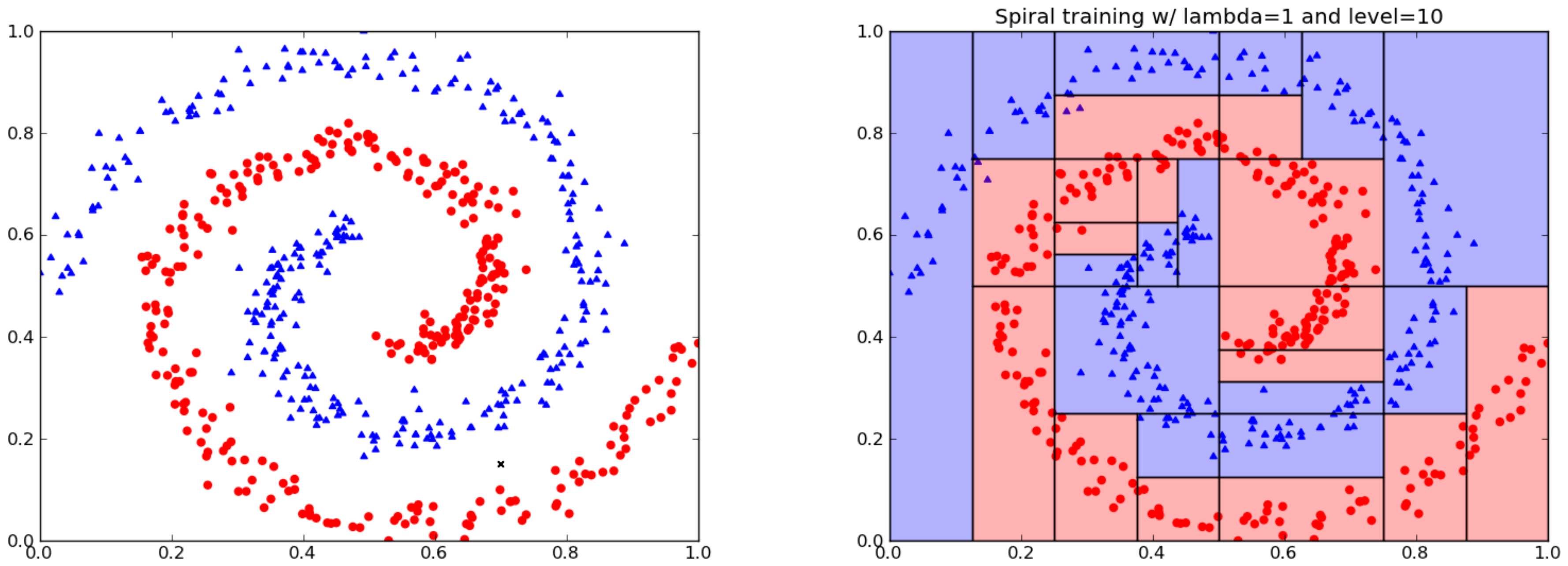


Decision tree formalism

- Decision tree is a binary tree V
- Internal nodes $u \in V$: *predicates* $\beta_u : \mathcal{X} \to \{0, 1\}$
- Leaves $v \in V$: *predictions* $\beta_v : \mathcal{X} \to \mathcal{Y}$
- Algorithm $h(\mathbf{x})$ starts at $u = u_0$
  - Compute $b = \beta_u(\mathbf{x})$
  - If u is a leaf, return b
  - If $b = 0$, $u \leftarrow \mathrm{LeftChild}(u)$
  - If $b = 1$, $u \leftarrow \mathrm{RightChild}(u)$
- In ML practice: $\beta_u(\mathbf{x}; j, t) = [\mathbf{x}^j < t]$



Decision surface



Picture credit: `https://www.classes.cs.uchicago.edu/archive/2015/winter/12200-1/assignments/pa5/index.html`

Decision tree training

- Greedy algorithm – on each iteration partition the data as if the split was final

Decision tree training

- Greedy algorithm – on each iteration partition the data as if the split was final
- There are two key components:
  - Split criterion to compare different ways to partition the data
  - Stopping condition to decide when to stop building the tree

Greedy tree learning

- Input: training set $D = \{(\mathbf{x}_i, y_i)\}$
- Let R_u be the data subset corresponding to node u

Greedy tree learning

- Input: training set $D = \left\{(\mathbf{x}_i, y_i)\right\}$
- Let R_u be the data subset corresponding to node u
  1. Greedily split R_u into R_l and R_r:

$$R_l(j,t) = \{\mathbf{x} \in R_u | \mathbf{x}^j \le t\}, \qquad R_r(j,t) = \{\mathbf{x} \in R_u | \mathbf{x}^j > t\}$$

     optimising a given loss: $Q(R_u, j, t) \to \min\limits_{(j,t)}$

Greedy tree learning

- Input: training set $D = \left\{(\mathbf{x}_i, y_i)\right\}$
- Let R_u be the data subset corresponding to node u

1. Greedily split R_u into R_l and R_r:

$$R_l(j,t) = \{\mathbf{x} \in R_u | \mathbf{x}^j \leq t\}, \qquad R_r(j,t) = \{\mathbf{x} \in R_u | \mathbf{x}^j > t\}$$

   optimising a given loss: $Q(R_u, j, t) \to \min_{(j,t)}$

2. If a stopping criterion is satisfied for u, declare it a leaf

Greedy tree learning

- Input: training set $D = \{(\mathbf{x}_i, y_i)\}$
- Let R_u be the data subset corresponding to node u

1. Greedily split R_u into R_l and R_r:

$$R_l(j,t) = \{\mathbf{x} \in R_u | \mathbf{x}^j \leq t\}, \qquad R_r(j,t) = \{\mathbf{x} \in R_u | \mathbf{x}^j > t\}$$

   optimising a given loss: $Q(R_u, j, t) \to \min\limits_{(j,t)}$
2. If a stopping criterion is satisfied for u, declare it a leaf
3. If not, create internal node u corresponding to the predicate $[\mathbf{x}^j \leq t]$ and repeat 1–3 for $R_u \equiv R_l(j,t)$ and $R_u \equiv R_r(j,t)$

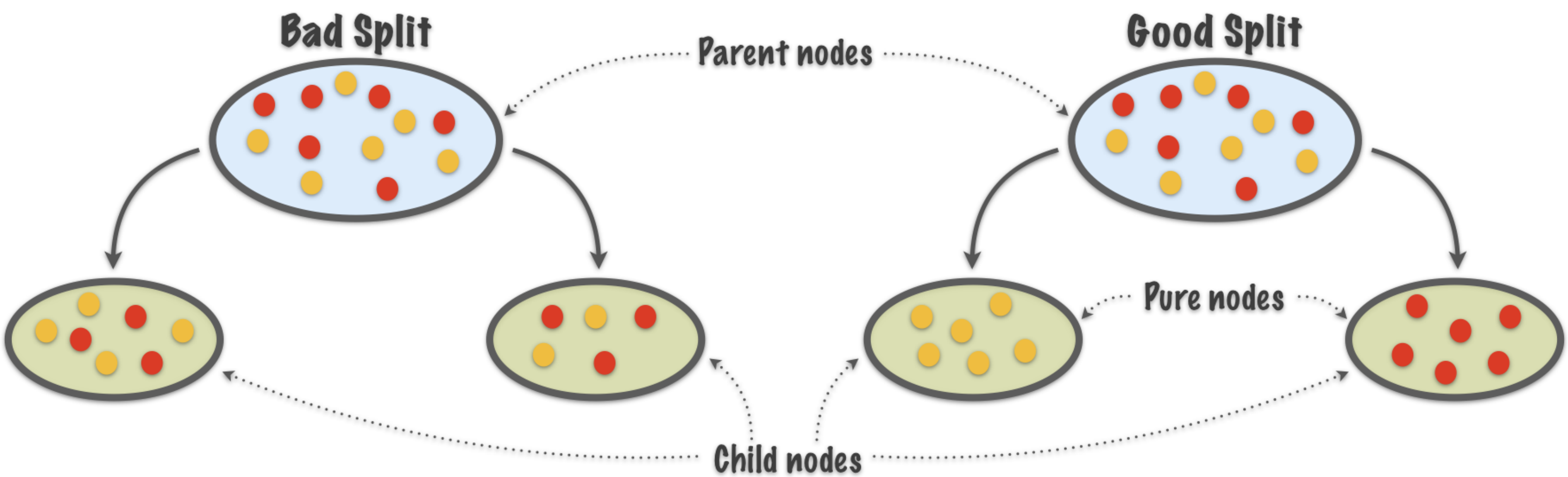
Greedy tree learning

- Input: training set $D = \{(\mathbf{x}_i, y_i)\}$
- Let R_u be the data subset corresponding to node u
  1. Greedily split R_u into R_l and R_r:

$$R_l(j,t) = \{\mathbf{x} \in R_u | \mathbf{x}^j \leq t\}, \qquad R_r(j,t) = \{\mathbf{x} \in R_u | \mathbf{x}^j > t\}$$

     optimising a given loss: $Q(R_u, j, t) \to \min_{(j,t)}$
  2. If a stopping criterion is satisfied for u, declare it a leaf
  3. If not, create internal node u corresponding to the predicate $[\mathbf{x}^j \leq t]$ and repeat 1–3 for $R_u \equiv R_l(j,t)$ and $R_u \equiv R_r(j,t)$
- Output: a decision tree V

The idea: maximize purity



Picture credit: `https://alanjeffares.wordpress.com/tutorials/decision-tree/`

The loss function $Q(D, j, t)$

- R_u: the subset of D corresponding to node u

The loss function $Q(D, j, t)$

- R_u: the subset of D corresponding to node u
- With the current split, let $R_l \subseteq R_u$ go left and $R_r \subseteq R_u$ go right

The loss function $Q(D, j, t)$

- R_u: the subset of D corresponding to node u
- With the current split, let $R_l \subseteq R_u$ go left and $R_r \subseteq R_u$ go right
- Choose a predicate to optimise

$$Q(R_u, j, t) = H(R_u) - \frac{|R_l|}{|R_u|}H(R_l) - \frac{|R_r|}{|R_u|}H(R_r) \to \max$$

The loss function $Q(D, j, t)$

- R_u: the subset of D corresponding to node u
- With the current split, let $R_l \subseteq R_u$ go left and $R_r \subseteq R_u$ go right
- Choose a predicate to optimise

$$Q(R_u, j, t) = H(R_u) - \frac{|R_l|}{|R_u|} H(R_l) - \frac{|R_r|}{|R_u|} H(R_r) \to \max$$

- $H(R)$: impurity criterion

The loss function $Q(D, j, t)$

- R_u: the subset of D corresponding to node u
- With the current split, let $R_l \subseteq R_u$ go left and $R_r \subseteq R_u$ go right
- Choose a predicate to optimise

$$Q(R_u, j, t) = H(R_u) - \frac{|R_l|}{|R_u|}H(R_l) - \frac{|R_r|}{|R_u|}H(R_r) \rightarrow \max$$

- $H(R)$: impurity criterion
- Generally, $H(R) = \min_{c \in \mathcal{Y}} |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} L(y_i, c)$

Examples of impurity criteria

- **Regression:**

Examples of impurity criteria

- **Regression:**
  - $H(R) = \min_{c \in \mathcal{Y}} |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} (y_i - c)^2$

Examples of impurity criteria

- **Regression:**
  - $H(R) = \min\limits_{c \in \mathcal{Y}} |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} (y_i - c)^2$
  - Sum of squared residuals is minimised by $c = |R|^{-1} \sum_{(\mathbf{x}_j, y_j) \in R} y_j$

Examples of impurity criteria

- **Regression:**
  - $H(R) = \min\limits_{c \in \mathcal{Y}} |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} (y_i - c)^2$
  - Sum of squared residuals is minimised by $c = |R|^{-1} \sum_{(\mathbf{x}_j, y_j) \in R} y_j$
  - Impurity $\equiv$ variance of the target

Examples of impurity criteria

- **Regression:**
  - $H(R) = \min\limits_{c \in \mathcal{Y}} |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} (y_i - c)^2$
  - Sum of squared residuals is minimised by $c = |R|^{-1} \sum_{(\mathbf{x}_j, y_j) \in R} y_j$
  - Impurity $\equiv$ variance of the target
- **Classification:**

Examples of impurity criteria

- **Regression:**
  - $H(R) = \min_{c \in \mathcal{Y}} |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} (y_i - c)^2$
  - Sum of squared residuals is minimised by $c = |R|^{-1} \sum_{(\mathbf{x}_j, y_j) \in R} y_j$
  - Impurity $\equiv$ variance of the target
- **Classification:**
  - Let's enumerate $\mathcal{Y} = \{\dots, y_k, \dots, y_N\}$

Examples of impurity criteria

- **Regression:**
  - $H(R) = \min_{c \in \mathcal{Y}} |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} (y_i - c)^2$
  - Sum of squared residuals is minimised by $c = |R|^{-1} \sum_{(\mathbf{x}_j, y_j) \in R} y_j$
  - Impurity $\equiv$ variance of the target
- **Classification:**
  - Let's enumerate $\mathcal{Y} = \{\ldots, y_k, \ldots, y_N\}$
  - Let $p_k = |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} [y_i = y_k]$ (share of examples belonging to the k'th class)

Examples of impurity criteria

- **Regression:**
  - $H(R) = \min_{c \in \mathcal{Y}} |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} (y_i - c)^2$
  - Sum of squared residuals is minimised by $c = |R|^{-1} \sum_{(\mathbf{x}_j, y_j) \in R} y_j$
  - Impurity $\equiv$ variance of the target
- **Classification:**
  - Let's enumerate $\mathcal{Y} = \{\ldots, y_k, \ldots, y_N\}$
  - Let $p_k = |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} [y_i = y_k]$ (share of examples belonging to the k'th class)
  - Miss rate (inaccuracy): $H(R) = \min_{c \in \mathcal{Y}} |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} [y_i \neq c]$

Examples of impurity criteria

- **Regression:**
  - $H(R) = \min\limits_{c \in \mathcal{Y}} |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} (y_i - c)^2$
  - Sum of squared residuals is minimised by $c = |R|^{-1} \sum_{(\mathbf{x}_j, y_j) \in R} y_j$
  - Impurity $\equiv$ variance of the target
- **Classification:**
  - Let's enumerate $\mathcal{Y} = \{\ldots, y_k, \ldots, y_N\}$
  - Let $p_k = |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} [y_i = y_k]$ (share of examples belonging to the k'th class)
  - Miss rate (inaccuracy): $H(R) = \min\limits_{c \in \mathcal{Y}} |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} [y_i \neq c]$
  - Gini index: $H(R) = \sum_{k=1}^{N} p_k (1 - p_k)$

Examples of impurity criteria

- **Regression:**
  - $H(R) = \min_{c \in \mathcal{Y}} |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} (y_i - c)^2$
  - Sum of squared residuals is minimised by $c = |R|^{-1} \sum_{(\mathbf{x}_j, y_j) \in R} y_j$
  - Impurity $\equiv$ variance of the target
- **Classification:**
  - Let's enumerate $\mathcal{Y} = \{\dots, y_k, \dots, y_N\}$
  - Let $p_k = |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} [y_i = y_k]$ (share of examples belonging to the k'th class)
  - Miss rate (inaccuracy): $H(R) = \min_{c \in \mathcal{Y}} |R|^{-1} \sum_{(\mathbf{x}_i, y_i) \in R} [y_i \neq c]$
  - Gini index: $H(R) = \sum_{k=1}^{N} p_k (1 - p_k)$
  - Cross-entropy: $H(R) = -\sum_{k=1}^{N} p_k \log p_k$

Stopping rules for decision tree learning

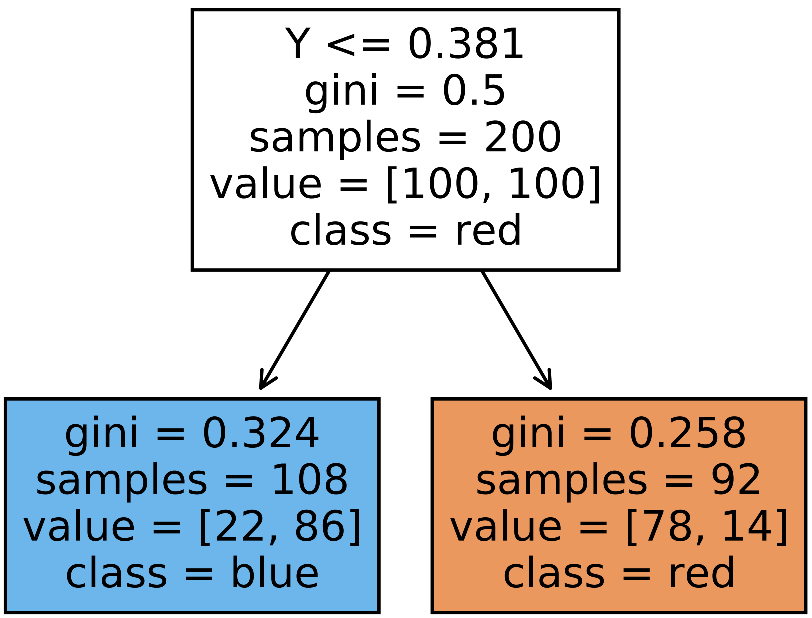
> What will happen if we build a decision tree to minimise the impurity to the greatest possible extent?

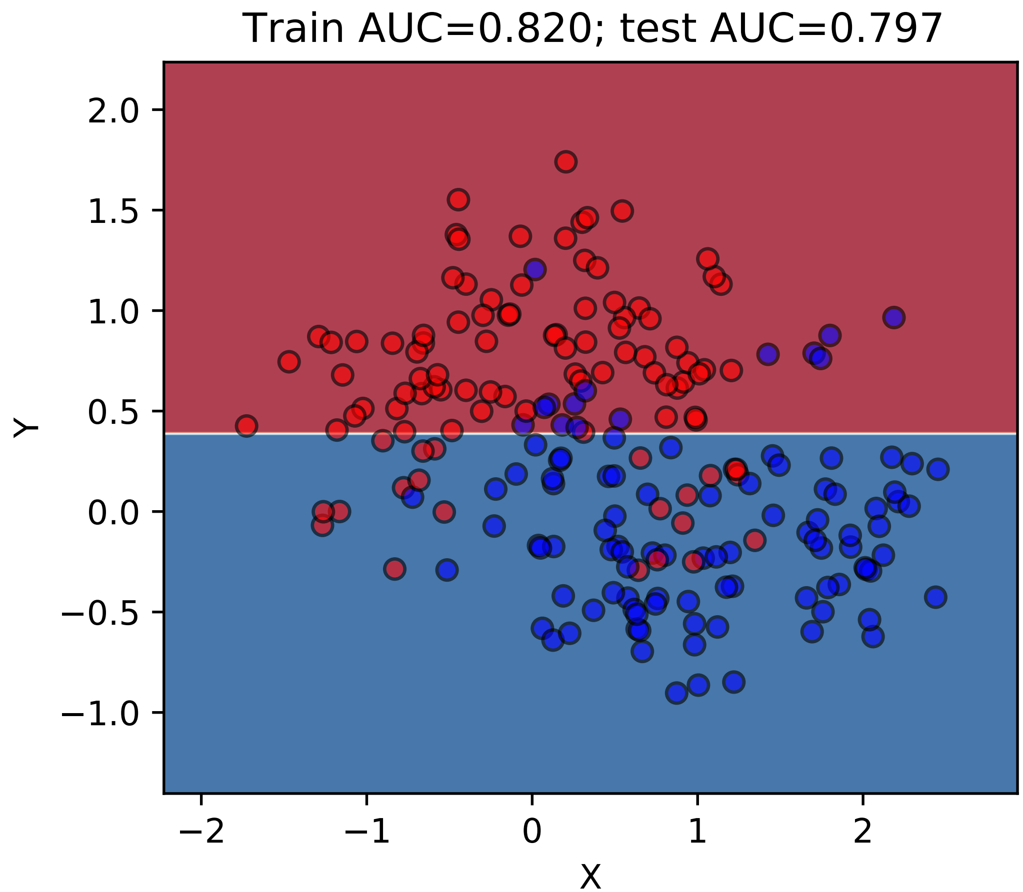
Stopping rules for decision tree learning

What will happen if we build a decision tree to minimise the impurity to the greatest possible extent?

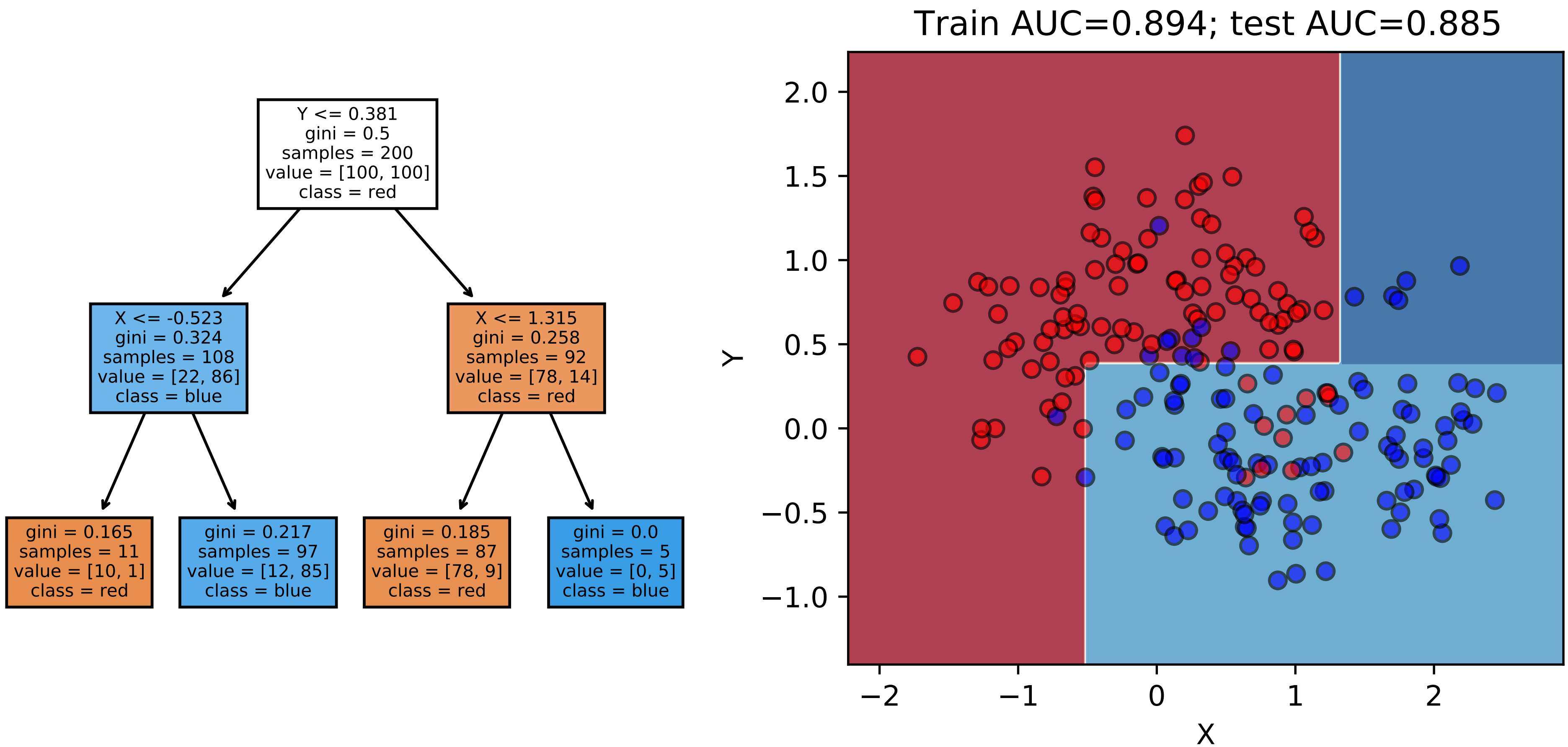
The tree will be built until all leaves are 100% pure, up to single-example leaves.

With decision trees, overfitting is easy!

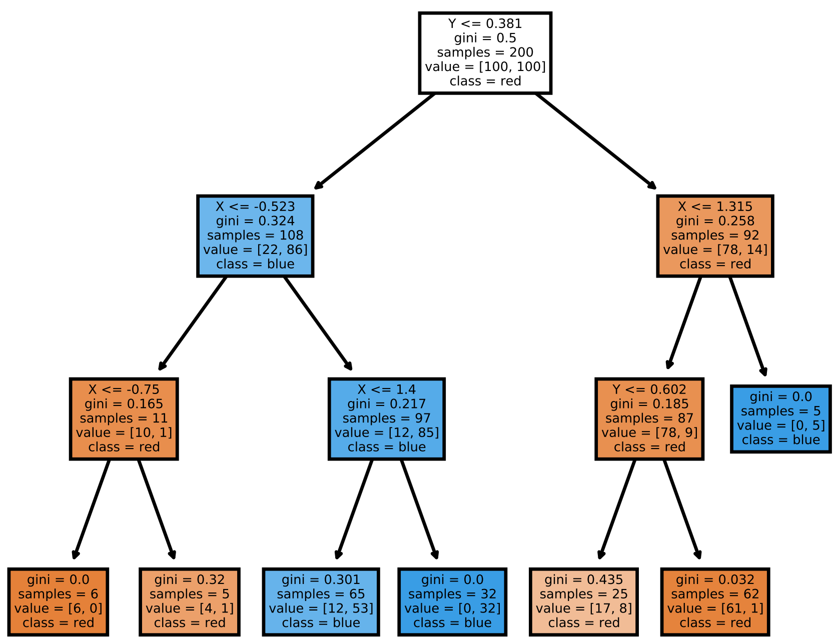


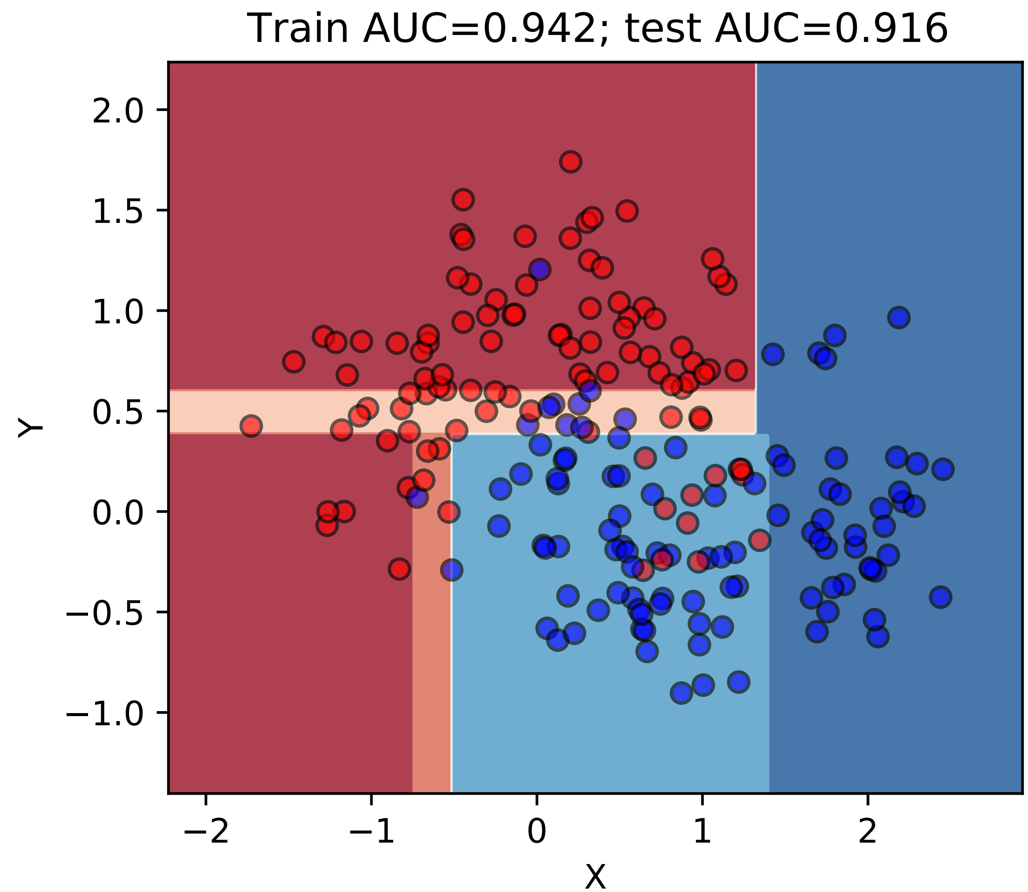


With decision trees, overfitting is easy!

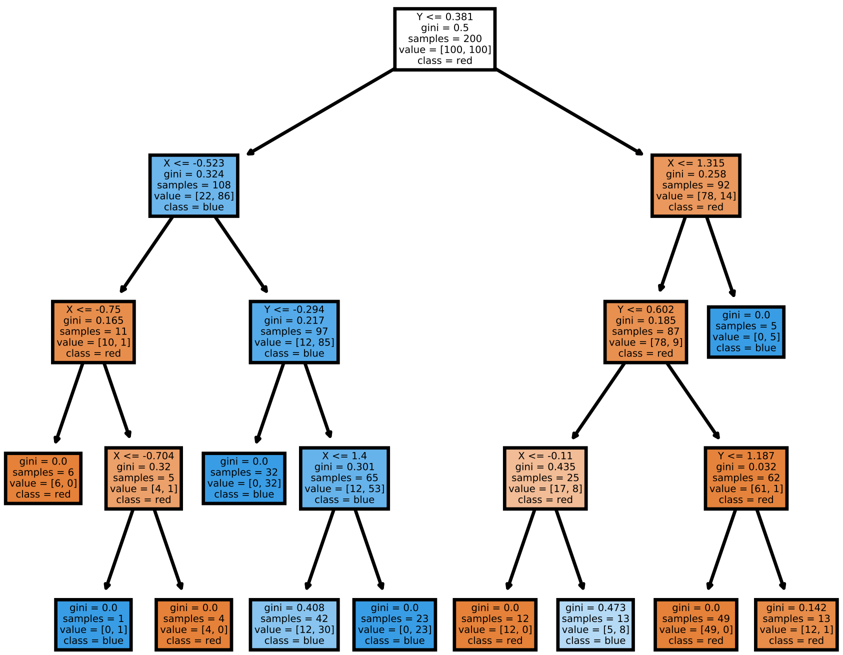


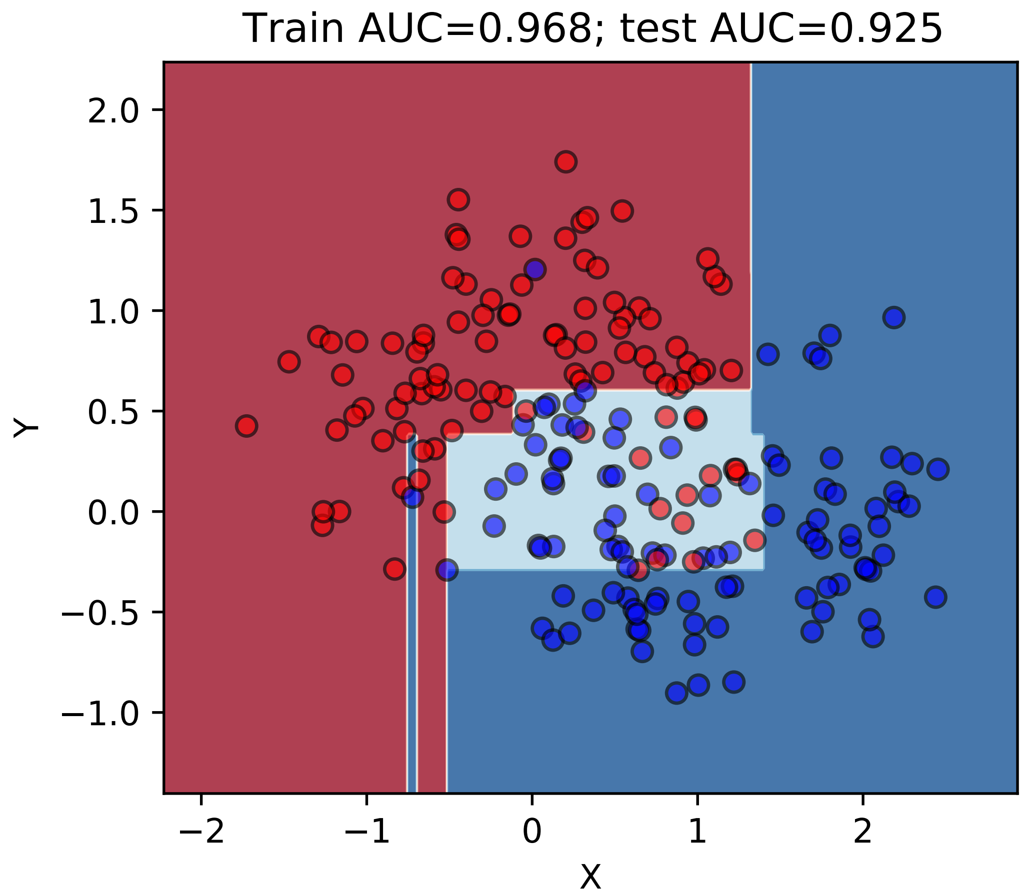
With decision trees, overfitting is easy!



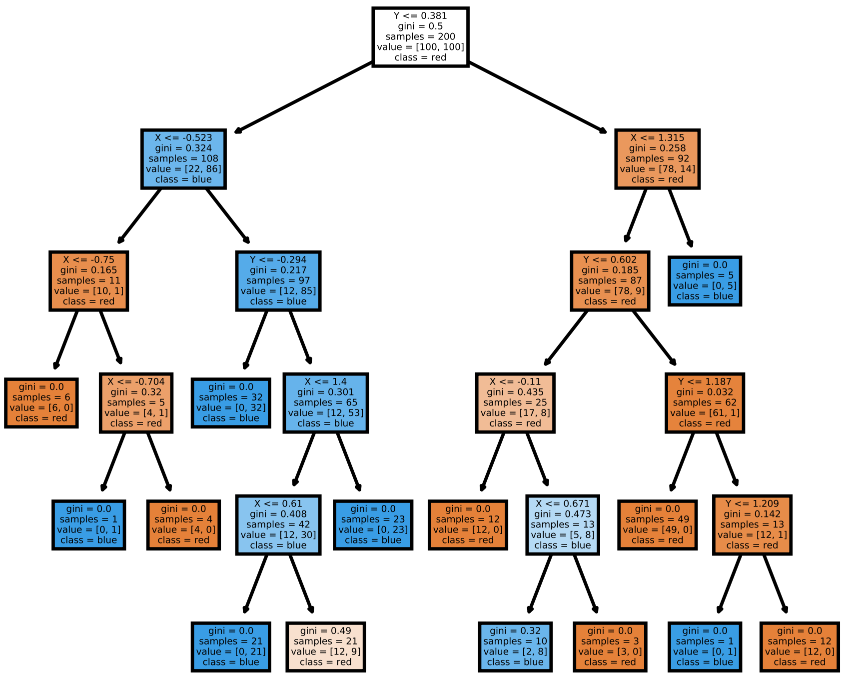


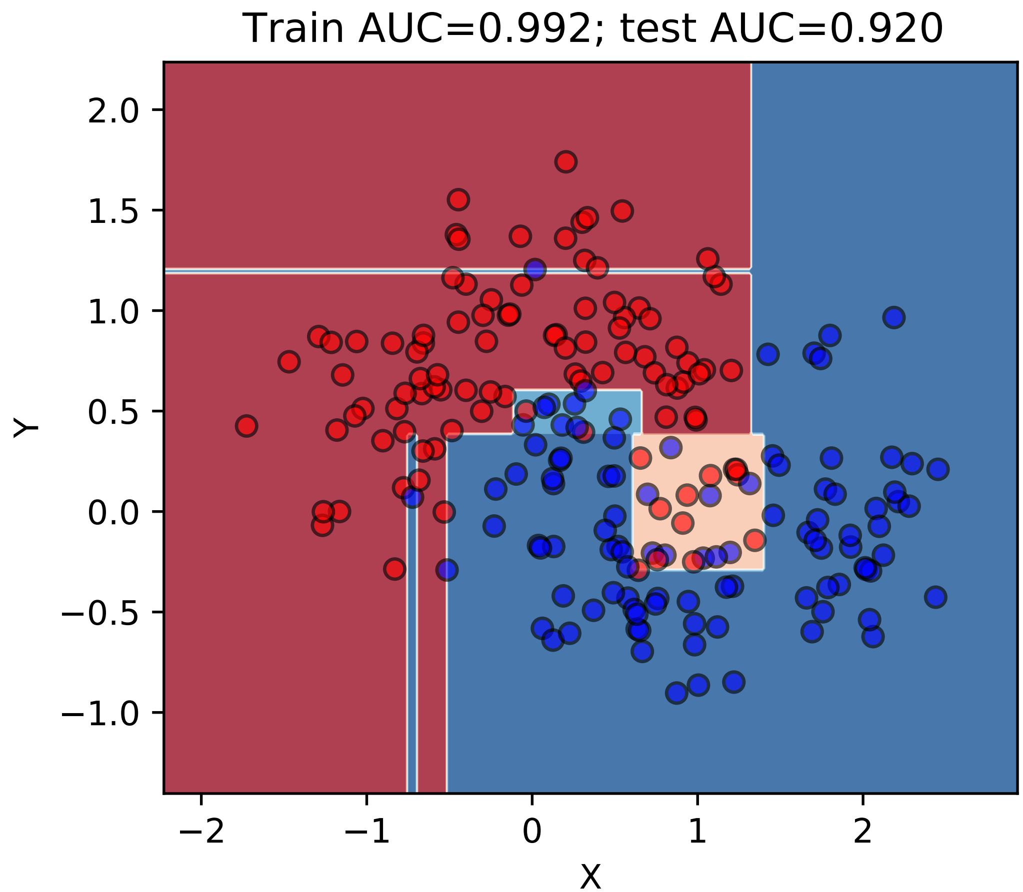
With decision trees, overfitting is easy!



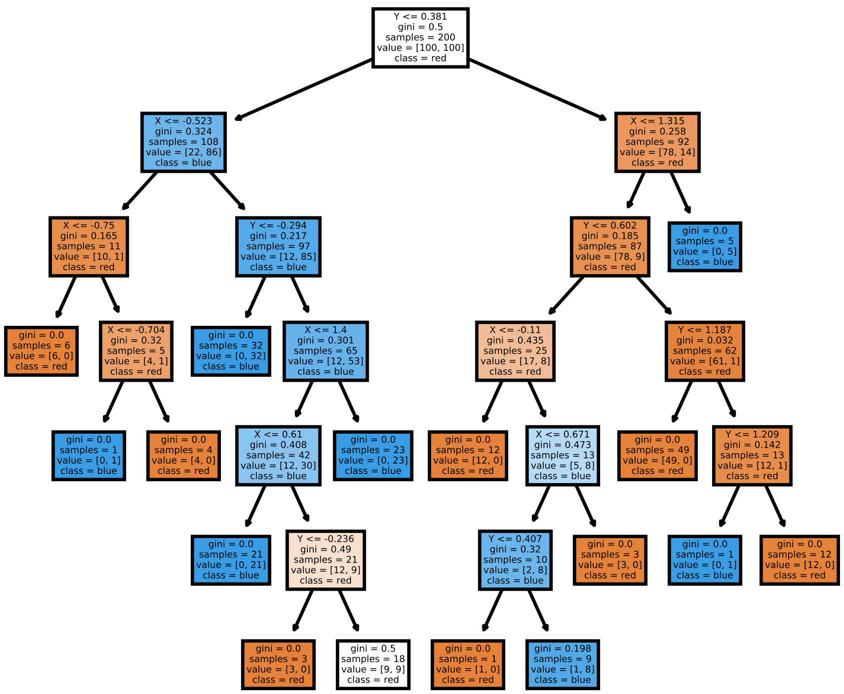


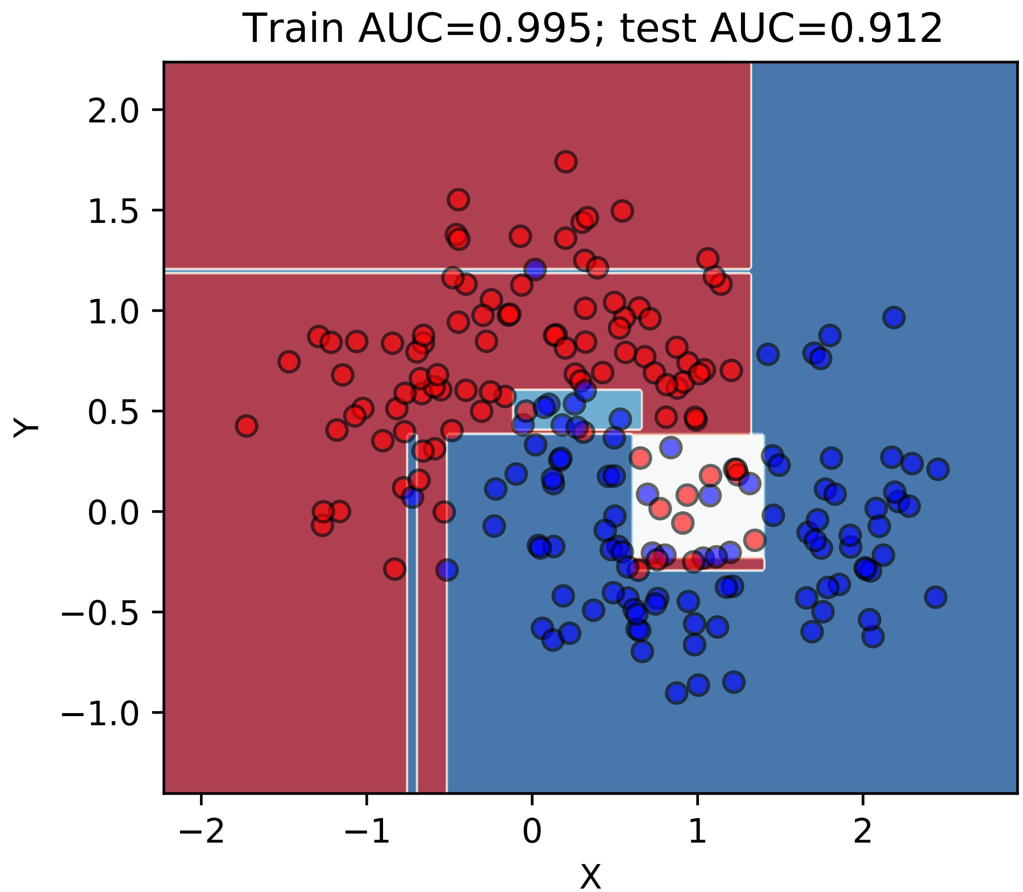
With decision trees, overfitting is easy!



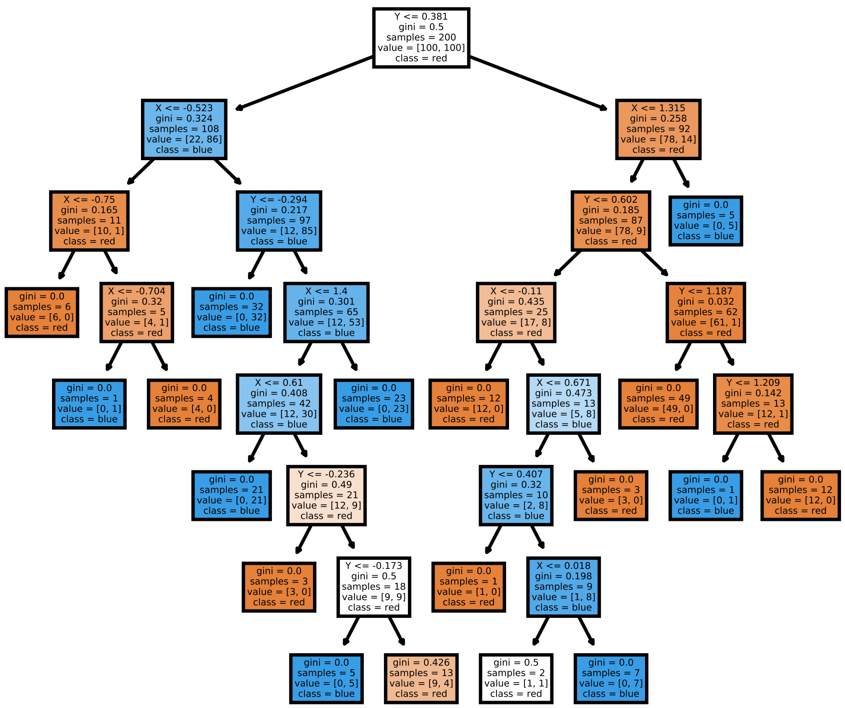


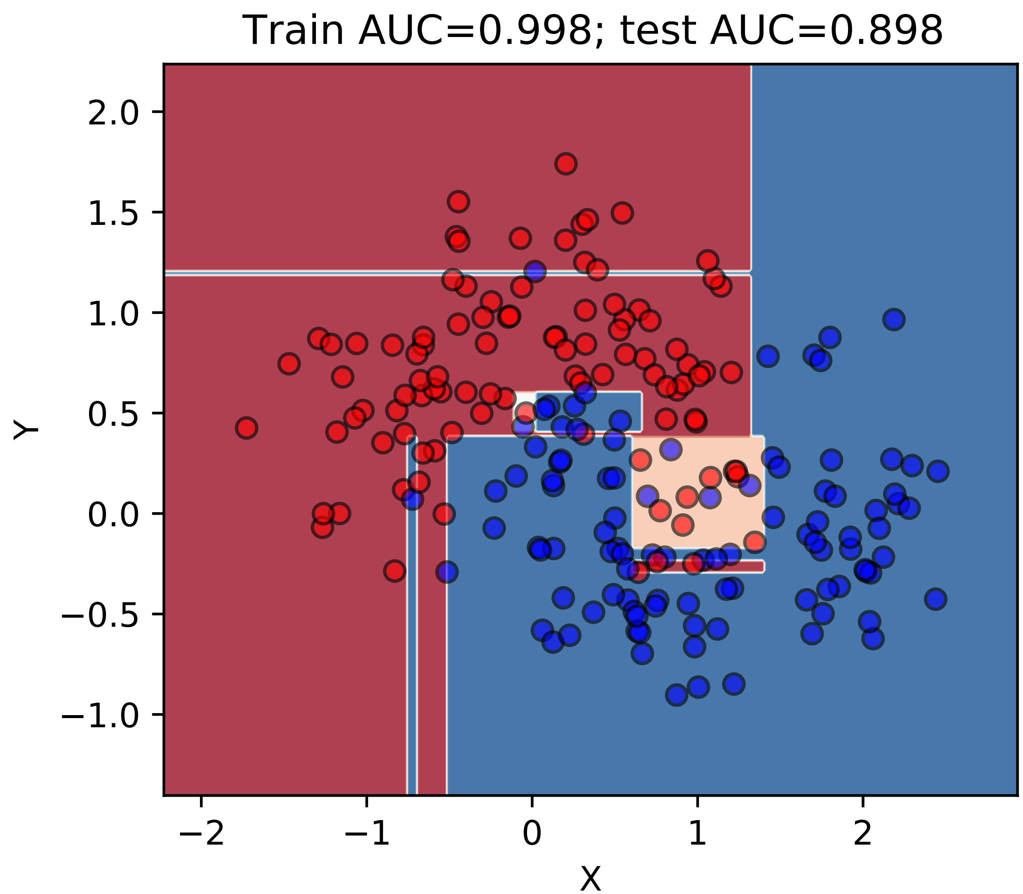
With decision trees, overfitting is easy!



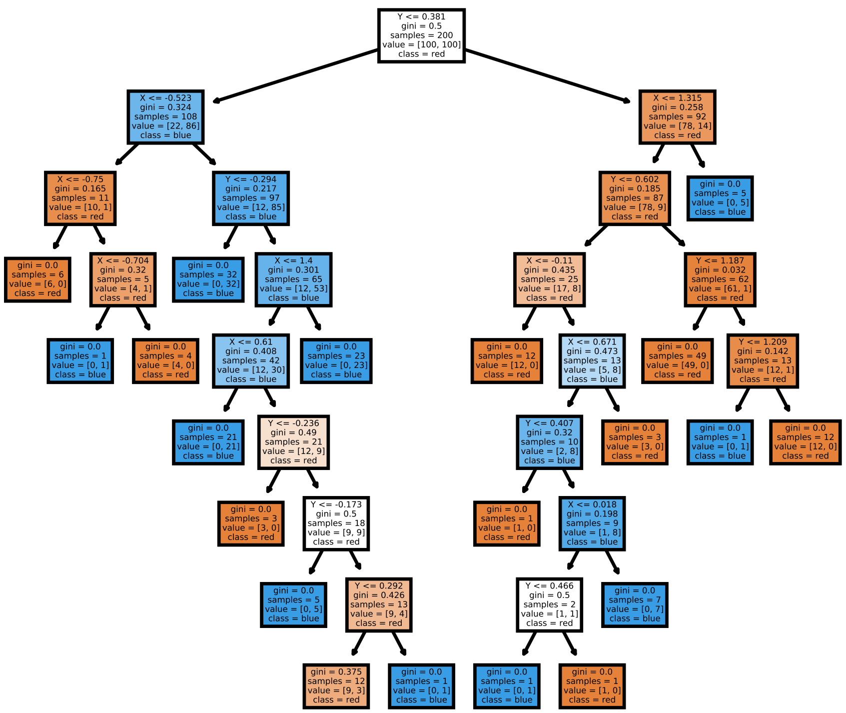


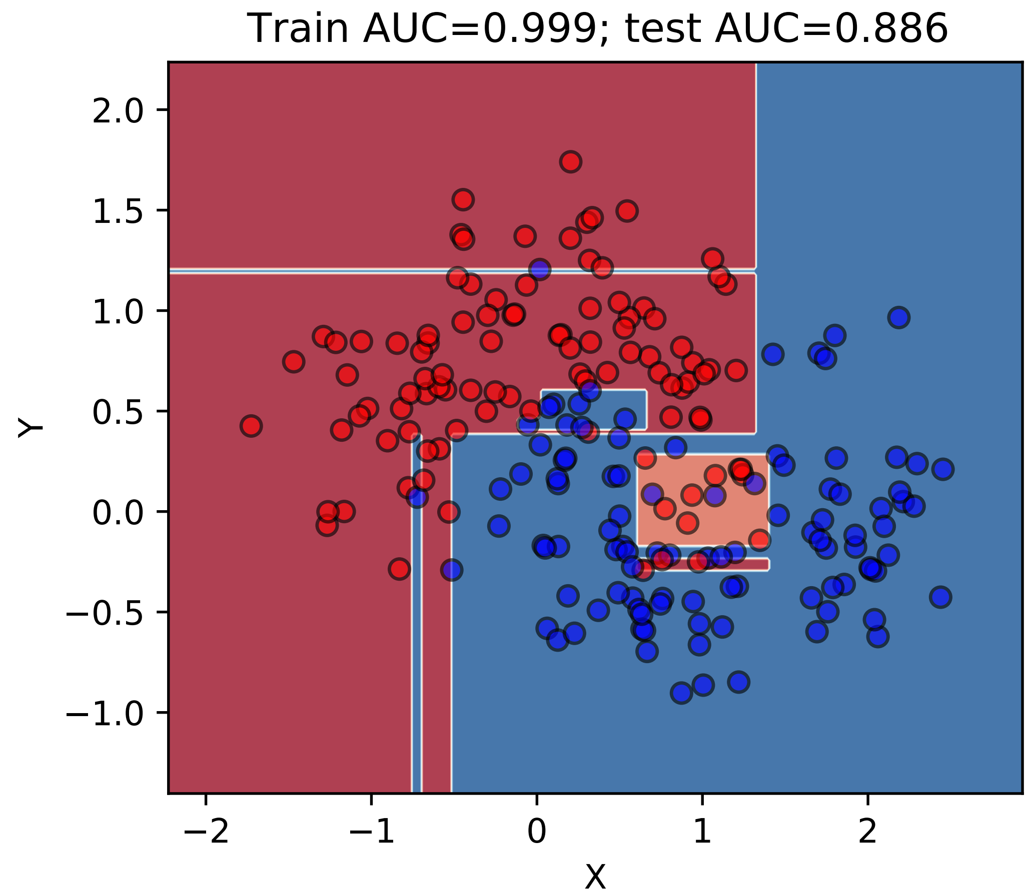
With decision trees, overfitting is easy!



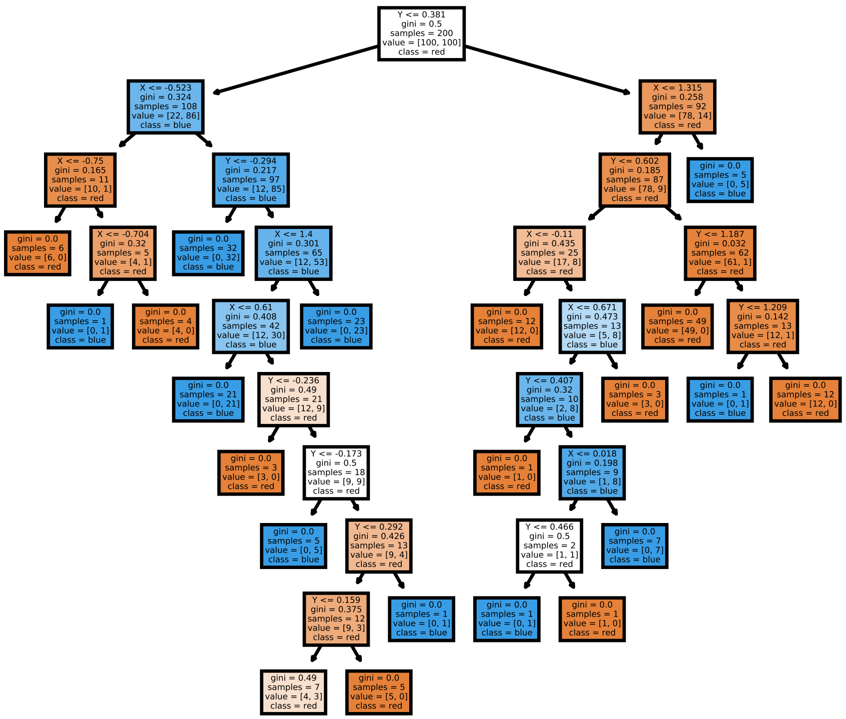


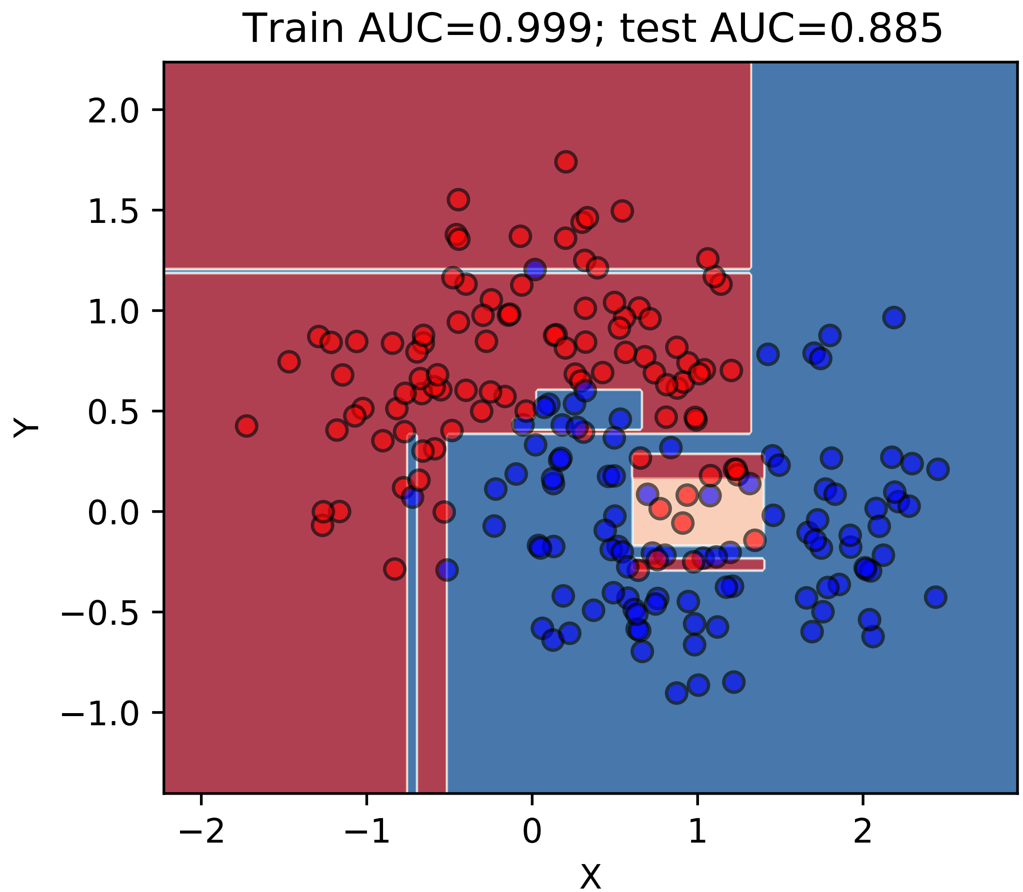
With decision trees, overfitting is easy!





With decision trees, overfitting is easy!





Stopping rules for decision tree learning

- Significantly impacts learning performance

Stopping rules for decision tree learning

- Significantly impacts learning performance
- Multiple choices available:

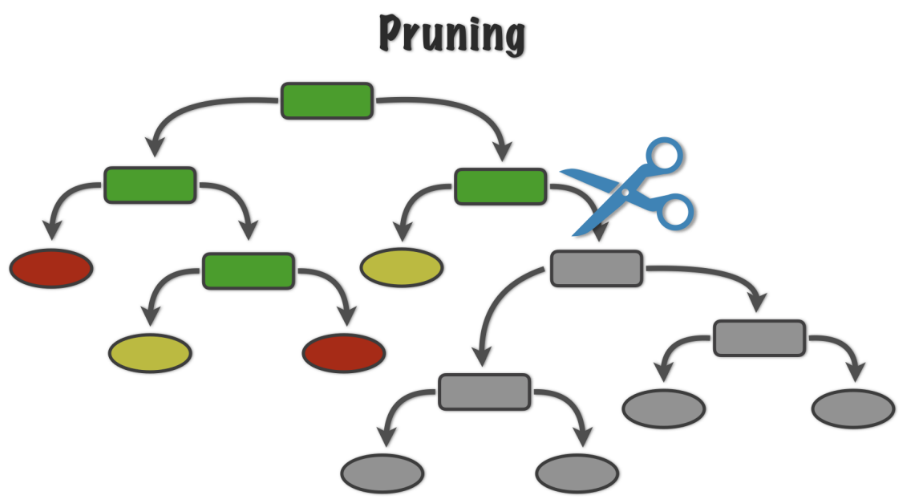
Stopping rules for decision tree learning

- Significantly impacts learning performance
- Multiple choices available:
  - Maximum tree depth
  - Minimum number of objects in leaf
  - Maximum number of leaves in tree
  - Stop if all objects fall into same leaf
  - Constrain quality improvement
    (stop when improvement gains drop below a defined threshold)

Stopping rules for decision tree learning

- Significantly impacts learning performance
- Multiple choices available:
  - Maximum tree depth
  - Minimum number of objects in leaf
  - Maximum number of leaves in tree
  - Stop if all objects fall into same leaf
  - Constrain quality improvement
    (stop when improvement gains drop below a defined threshold)
- Typically selected via exhaustive search and cross-validation

Decision tree pruning



Picture credit: `https://alanjeffares.wordpress.com/tutorials/decision-tree/`

- Learn a large tree (effectively overfit the training set)
- Remove the least important nodes
- Often leads to better results than simply learning a smaller tree. Why?

Summary

- The good things. Decision trees:

Summary

- The good things. Decision trees:
  - ... are interpretable, you can show why an algorithm made the prediction it did and what you need to do to make the decision different

Summary

- The good things. Decision trees:
  - ... are interpretable, you can show why an algorithm made the prediction it did and what you need to do to make the decision different
  - ... require little preprocessing, don't depend on the features' scale

Summary

- The good things. Decision trees:
  - ... are interpretable, you can show why an algorithm made the prediction it did and what you need to do to make the decision different
  - ... require little preprocessing, don't depend on the features' scale
- The bad thing:
  - sub par performance. But that can be improved with ensembles, stay tuned for the next lecture!

Acknowledgements

These slides are based on the slides for for the previous edition of the MLHEP school by Alexey Artemov.