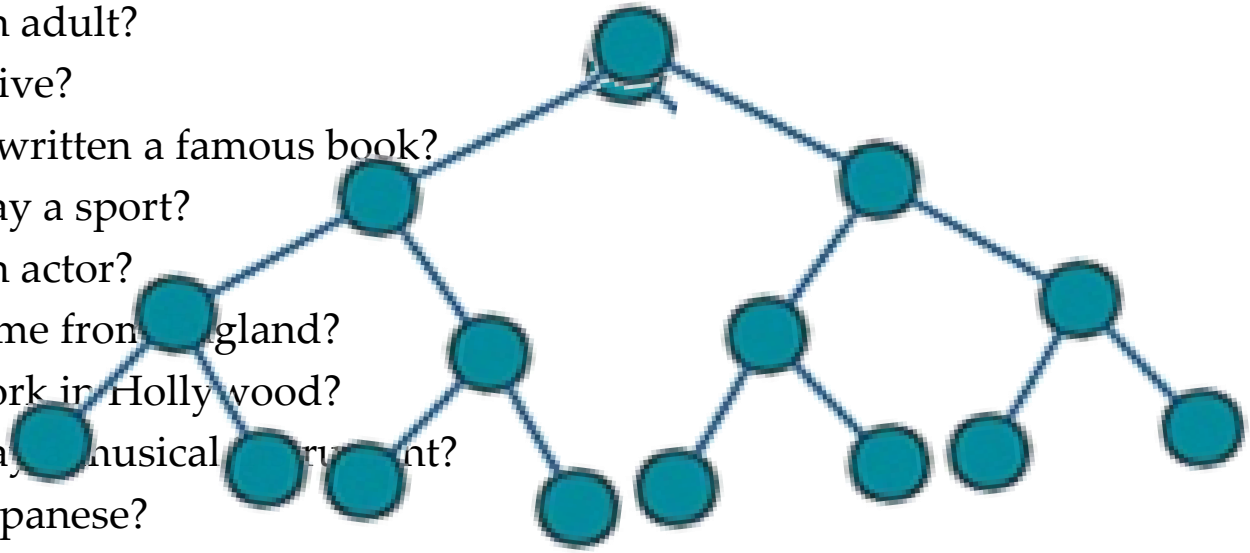
Decision Tree

Zhou Zhao (赵洲)

College of Computer Science
Zhejiang University

Decision Tree

- Yes/No Question Game
  - Aim: To guess the name of a famous person or character by asking yes/no questions
  - Example questions:
    - Are you male?
    - Are you a real person?
    - Are you an adult?
    - Are you alive?
    - Have you written a famous book?
    - Do you play a sport?
    - Are you an actor?
    - Do you come from England?
    - Do you work in Hollywood?
    - Do you play a musical instrument?
    - Are you Japanese?
    - Are you a scientist?
    - Are you a cartoon character?

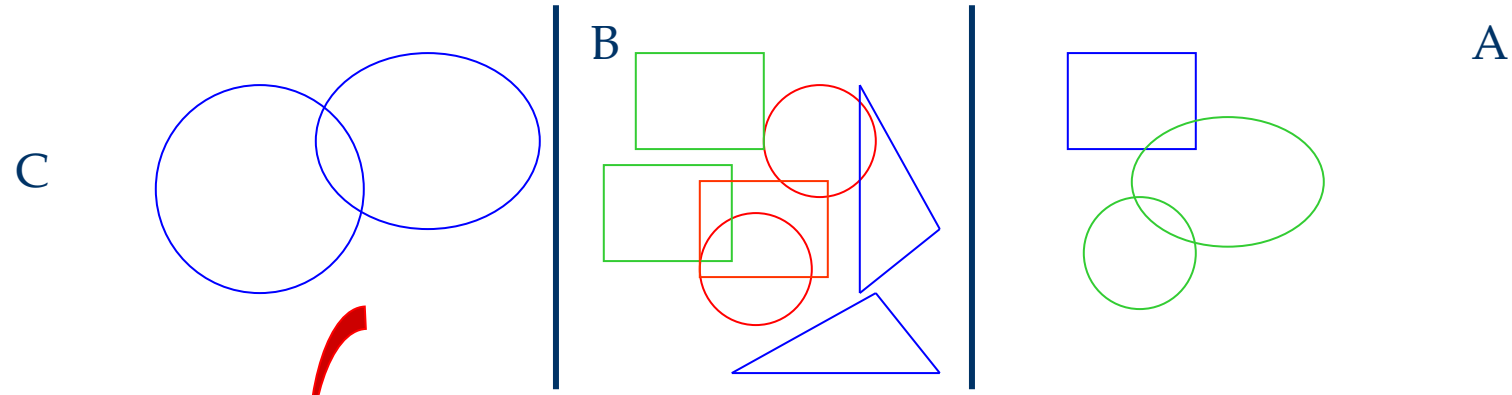


Decision Tree

- A hierarchical data structure that represents data by implementing a divide and conquer strategy
  - Given a collection of examples, learn a decision tree that represents it.
  - Use this representation to classify new examples

- Can be used as a non-parametric classification and regression method.

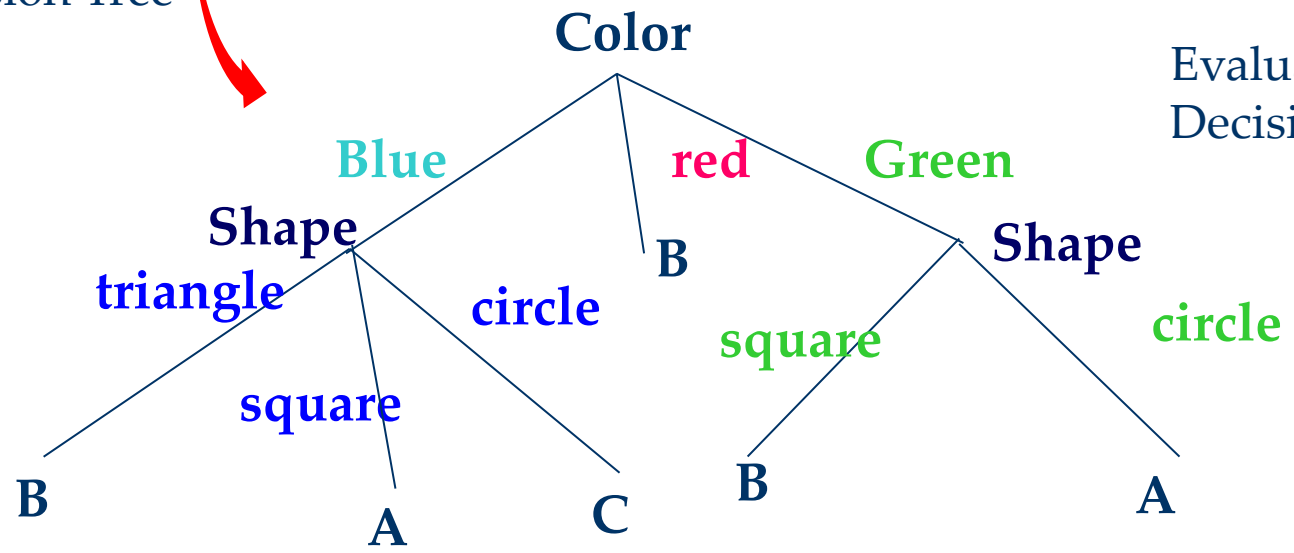
Decision Tree



Learning a Decision Tree

(color= RED ;shape=triangle)

Evaluation of a Decision Tree



Decision Tree

- Decision Trees are classifiers for instances represented as features vectors (color= ;shape= ;label=)
- Nodes are tests for feature values;
- There is one branch for each value of the feature
- Leaves specify the categories (labels)
- Can categorize instances into multiple disjoint categories
- Output is a discrete category. Real valued outputs are possible (regression trees)
- There are efficient algorithms for processing large amounts of data. (But not too many features)
- There are methods for handling noisy data (classification noise and attribute noise) and for handling missing attribute values.

Basic Decision Tree Learning Algorithm

- Data is processed in Batch (I.e., all the data is available).
- Recursively build a decision tree top-down.

- DT(Examples, Attributes)

```
If all Examples have same label:
        return a leaf node with Label
Else
        If  Attributes is empty:
           return a leaf with majority Label
        Else
           Pick an attribute A as root
           For each value v of A
           Let Examples(v) be all the examples for which  A=v
           Add a branch out of the root for the test  A=v
              If  Examples(v)  is empty
                    create a leaf node labeled with the majority label in Examples
              Else
                    recursively create subtree by calling DT(Examples(v), Attribute-{A})
```

Decision Tree

- Yes/No Question Game
  - Aim: To guess the name of a famous person or character by asking yes/no questions
  - Example questions:
    - Are you male?
    - Are you a real person?
    - Are you an adult?
    - Are you alive?
    - Have you written a famous book?
    - Do you play a sport?
    - Are you an actor?
    - Do you come from England?
    - Do you work in Hollywood?
    - Do you play a musical instrument?
    - Are you Japanese?
    - Are you a scientist?
    - Are you a cartoon character?

How many trees we can build?

All the trees are the same? Or Some trees are better than others?

Decision Tree

- A better tree (model)
  - Small trees

$$\mathrm{EPE}(f) = \textbf{(bias)}^2 \textbf{ + variance + noise}$$

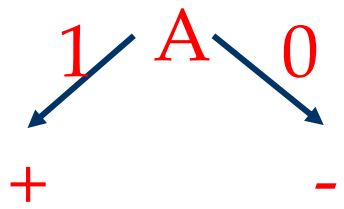
  - Same bias, less variance.
- Finding the minimal decision tree consistent with the data is NP-hard

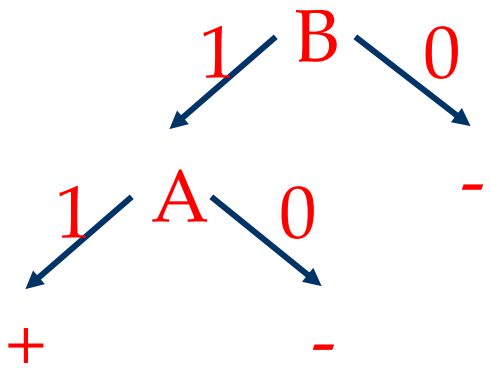
- The recursive algorithm is a greedy heuristic search for a simple tree, but cannot guarantee optimality.
- The main decision in the algorithm is the selection of the next attribute to condition on.

Picking the Best Feature (Attribute)

- Consider data with two Boolean attributes (A,B).
  - < (A=0,B=0), - >: 50 examples
  - < (A=0,B=1), - >: 50 examples
  - < (A=1,B=0), - >: 0 examples
  - < (A=1,B=1), + >: 100 examples

- What should be the first attribute we select? A or B?
- Splitting on A: we get purely labeled nodes
- Splitting on B: we don't get purely labeled nodes.

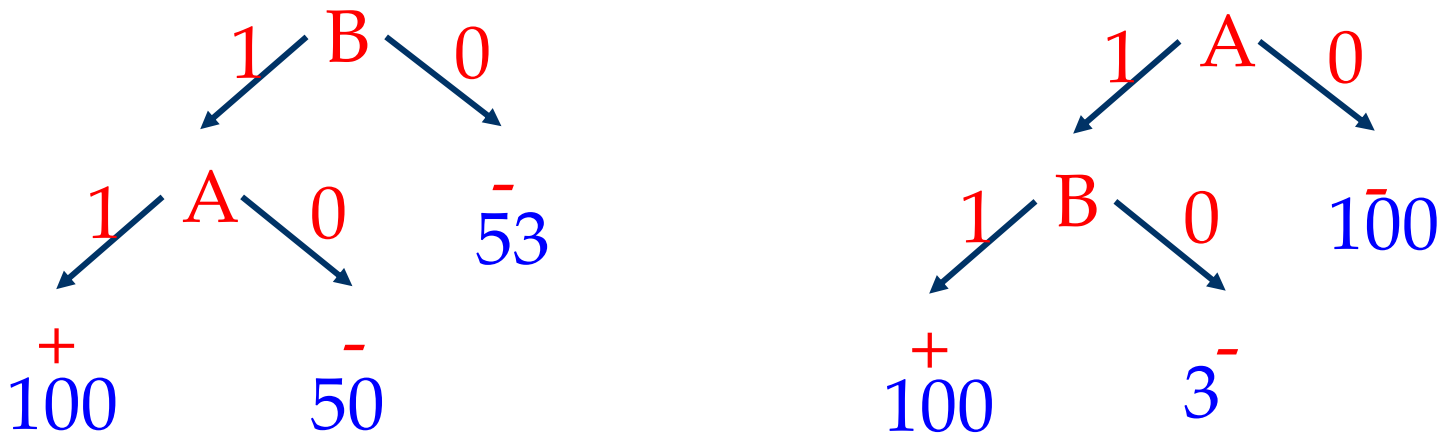




Picking the Best Feature (Attribute)

- Consider data with two Boolean attributes (A,B).
  - < (A=0,B=0), - >: 50 examples
  - < (A=0,B=1), - >: 50 examples
  - < (A=1,B=0), - >: 3 examples
  - < (A=1,B=1), + >: 100 examples

- Trees looks structurally similar; which attribute should we choose?



Advantage A. But…
Need a way to quantify things

Picking the Best Feature (Attribute)

- We want attributes that split the examples to sets that are relatively pure in one label; this way we are closer to a leaf node.

- The most popular heuristics is based on information gain, originated with the ID3 system of Quinlan.

Entropy

- Entropy (impurity, disorder) of a set of examples, S, relative to a binary classification is:

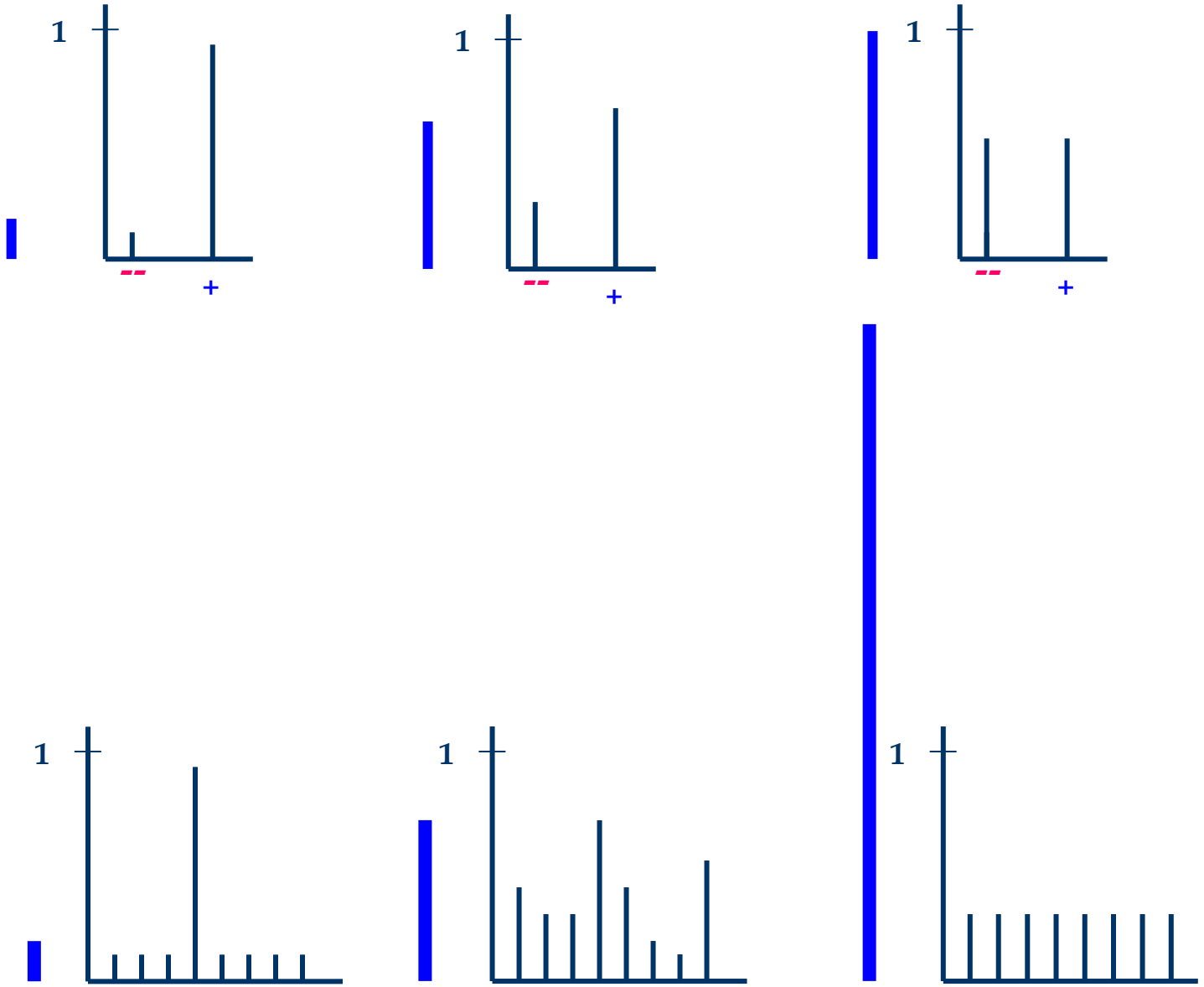
$$Entropy(S) = -P_+ \log P_+ - P_- \log P_-$$

  - where P_+ is the proportion of positive examples in S
  - P_- is the proportion of negative examples
  - If all the examples belong to the same category $Entropy = 0$
  - If the examples are equally mixed (0.5,0.5) $Entropy = 1$
- **In general, when p_i is the fraction of examples labeled i:**

$$Entropy(S) = -\sum_{i=1}^{c} P_i \log P_i$$

- Entropy can be viewed as the number of bits required, on average, to encode the class of labels. If the probability for + is 0.5, a single bit is required for each example; if it is 0.8 -- can use less then 1 bit.

Entropy



Picking the Best Feature (Attribute)

- We want attributes that split the examples to sets **that are relatively pure in one label**; this way we are closer to a leaf node.

- We can pick the feature that the resulting data partitions have low entropy

Information Gain

- **The information gain** of an attribute a is the expected reduction in entropy caused by partitioning on this attribute.

$$Gain(S, a) = Entropy(S) - \sum_{v \in values(a)} \frac{|S_v|}{|S|} Entropy(S_v)$$

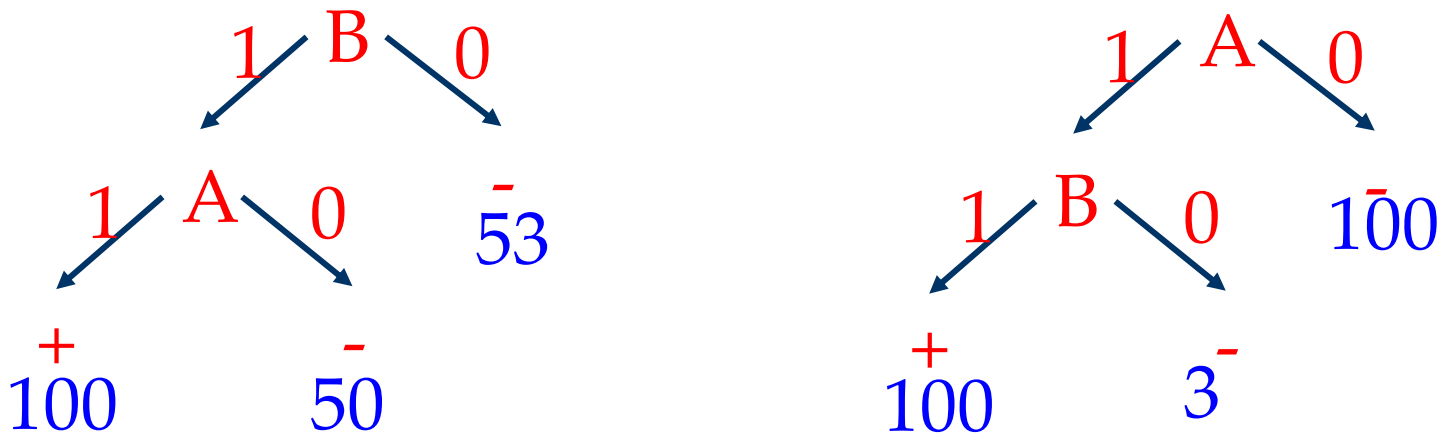
  - where S_v is the subset of S for which attribute a has value v

- Partitions of low entropy lead to high gain.

Picking the Best Feature (Attribute)

- Consider data with two Boolean attributes (A,B).
  - < (A=0,B=0), - >: 50 examples
  - < (A=0,B=1), - >: 50 examples
  - < (A=1,B=0), - >: 3 examples
  - < (A=1,B=1), + >: 100 examples

- Trees looks structurally similar; which attribute should we choose?



The information gain of A and B

An Illustrative Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

An Illustrative Example (2)

$$Entropy(S) = -\frac{9}{14}\log\left(\frac{9}{14}\right) - \frac{5}{14}\log\left(\frac{5}{14}\right) = 0.94$$

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

9+,5-

An Illustrative Example (2)

Humidity	Wind	PlayTennis
High	Weak	No
High	Strong	No
High	Weak	Yes
High	Weak	Yes
Normal	Weak	Yes
Normal	Strong	No
Normal	Strong	Yes
High	Weak	No
Normal	Weak	Yes
Normal	Weak	Yes
Normal	Strong	Yes
High	Strong	Yes
Normal	Weak	Yes
High	Strong	No

9+,5-

E=.94

Humidity

High: 3+,4- E=.985

Normal: 6+,1- E=.592

$Gain(S,\text{Humidity})$= .94 - 7/14 0.985 - 7/14 0.592= 0.151

Wind

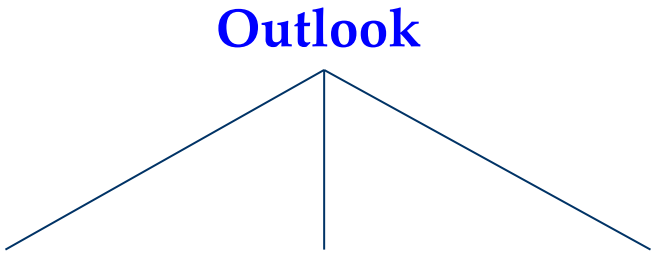
Weak: 6+2- E=.811

Strong: 3+,3- E=1.0

$Gain(S,\text{Wind})$= .94 - 8/14 0.811 - 6/14 1.0 = 0.048

$$Gain(S,a) = Entropy(S) - \sum_{v \in values(a)} \frac{|S_v|}{|S|} Entropy(S_v)$$

An Illustrative Example (3)



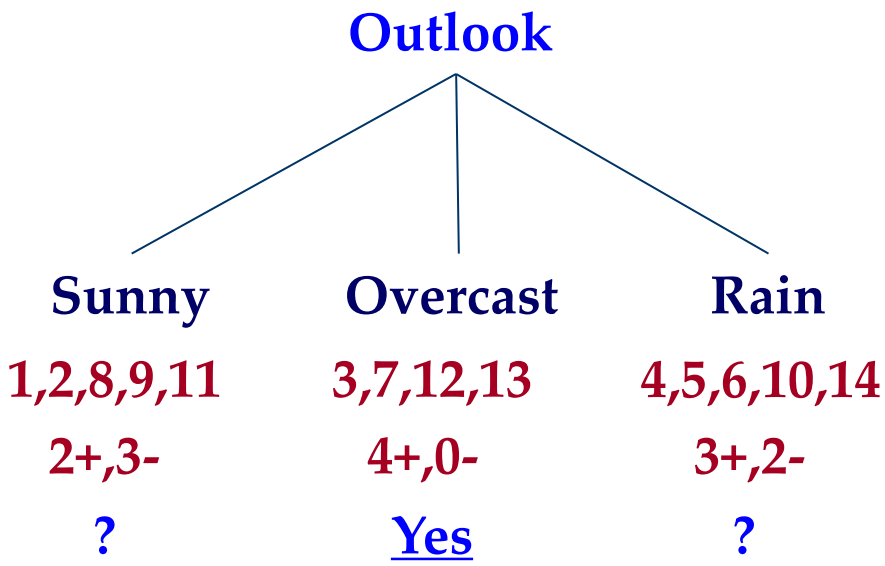
$Gain(S,\text{Humidity})=0.151$

$Gain(S,\text{Wind})=0.048$

$Gain(S,\text{Temperature})=0.029$

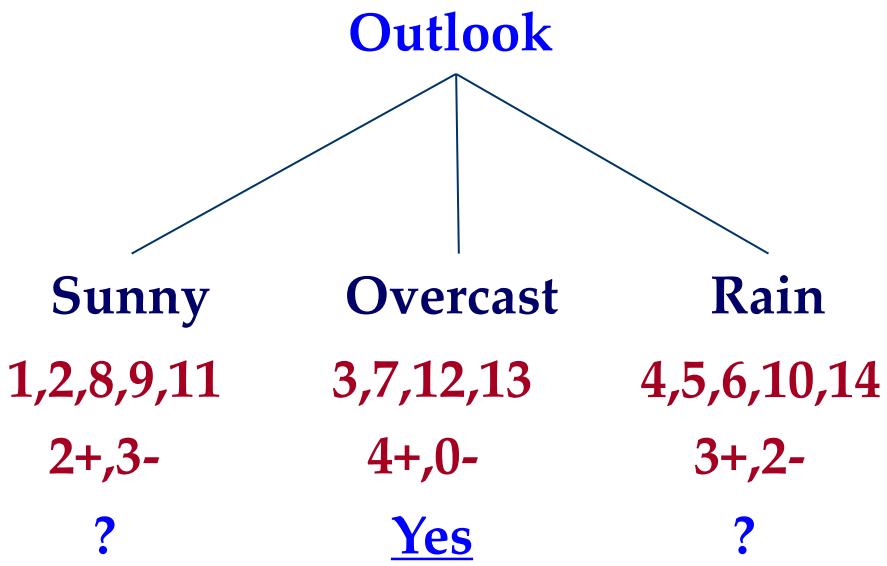
$Gain(S,\text{Outlook})=0.246$

An Illustrative Example (3)



Day	Outlook	PlayTennis
1	Sunny	No
2	Sunny	No
3	Overcast	Yes
4	Rain	Yes
5	Rain	Yes
6	Rain	No
7	Overcast	Yes
8	Sunny	No
9	Sunny	Yes
10	Rain	Yes
11	Sunny	Yes
12	Overcast	Yes
13	Overcast	Yes
14	Rain	No

An Illustrative Example (3)



Day	Outlook	PlayTennis
1	Sunny	No
2	Sunny	No
3	Overcast	Yes
4	Rain	Yes
5	Rain	Yes
6	Rain	No
7	Overcast	Yes
8	Sunny	No
9	Sunny	Yes
10	Rain	Yes
11	Sunny	Yes
12	Overcast	Yes
13	Overcast	Yes
14	Rain	No

Continue until:

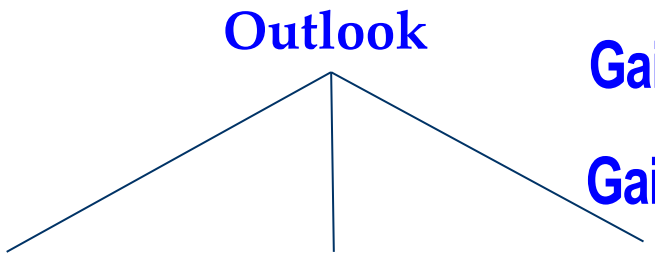
- **Every attribute is included in path, or,**
- **All examples in the leaf have same label**

An Illustrative Example (4)

$Gain(S_{sunny}, Humidity) = .97-(3/5)\ 0-(2/5)\ 0 = .97$

$Gain(S_{sunny}, Temp) = .97-\ 0-(2/5)\ 1 = .57$

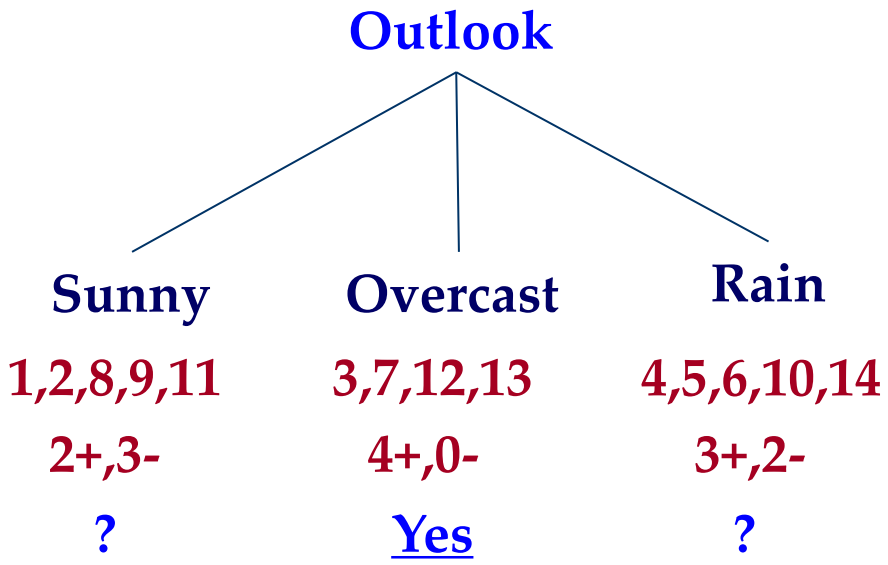
$Gain(S_{sunny}, Wind) = .97-(2/5)\ 1\ -\ (3/5)\ .92 = .02$



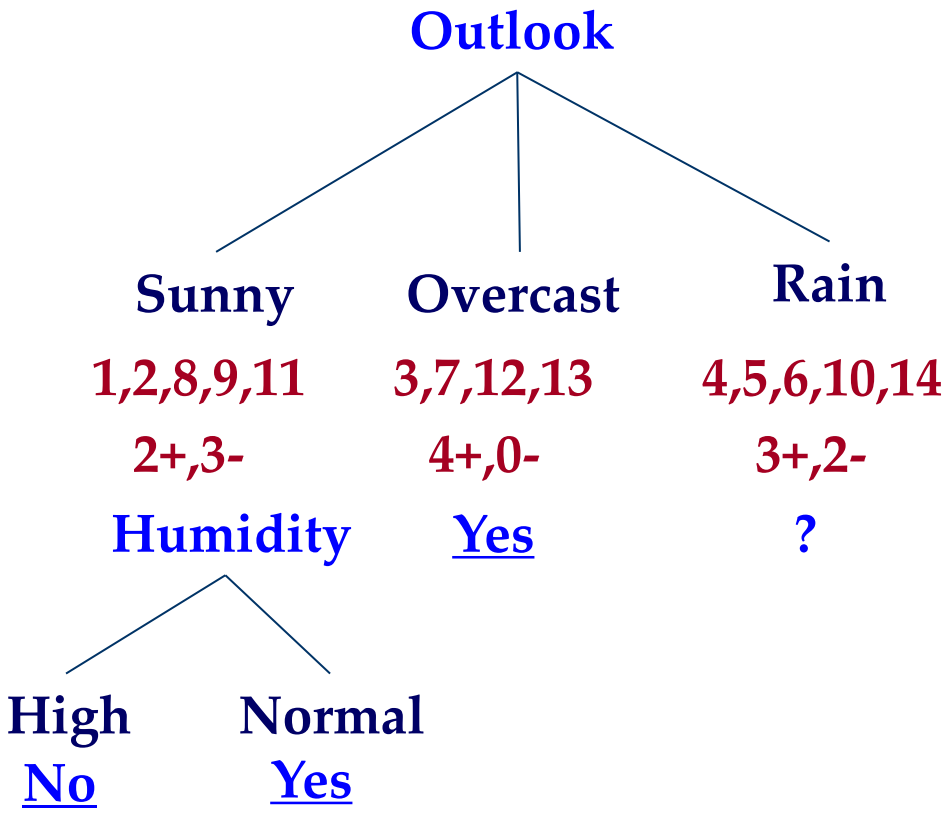
Sunny	Overcast	Rain
1,2,8,9,11	3,7,12,13	4,5,6,10,14
2+,3-	4+,0-	3+,2-
?	Yes	?

Day	Outlook	Temperature	Humidity	Wind	PlayTenn
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

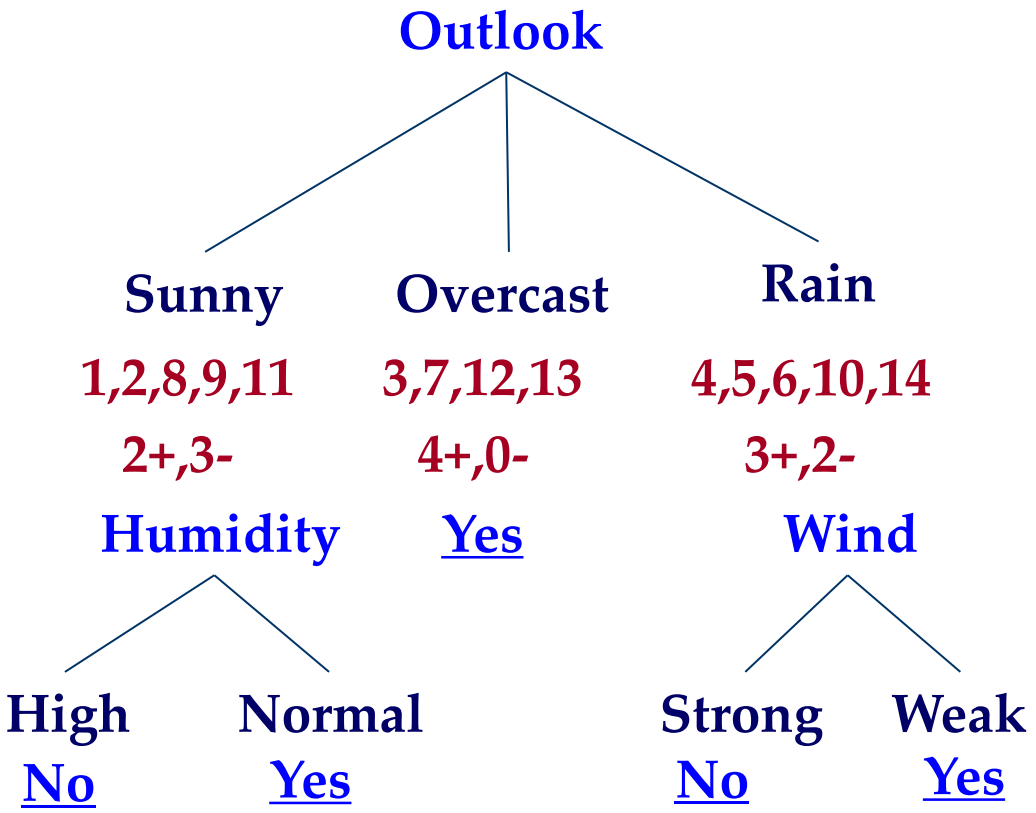
An Illustrative Example (5)



An Illustrative Example (5)



An Illustrative Example (6)



Summary: ID3(Examples, Attributes, Label)

- Let S be the set of Examples
  Label is the target attribute (the prediction)
  Attributes is the set of measured attributes
-
- Create a Root node for tree
- If all examples are labeled the same return a single node tree with Label
- Otherwise Begin
- A = attribute in Attributes that *best* classifies S
- for each possible value v of A
- Add a new tree branch corresponding to A=v
- Let *Sv* be the subset of examples in S with A=v
- if *Sv* is empty: add leaf node with the common value of Label in S
- Else: below this branch add the subtree
- ID3(*Sv*, Attributes - {a}, Label)

End

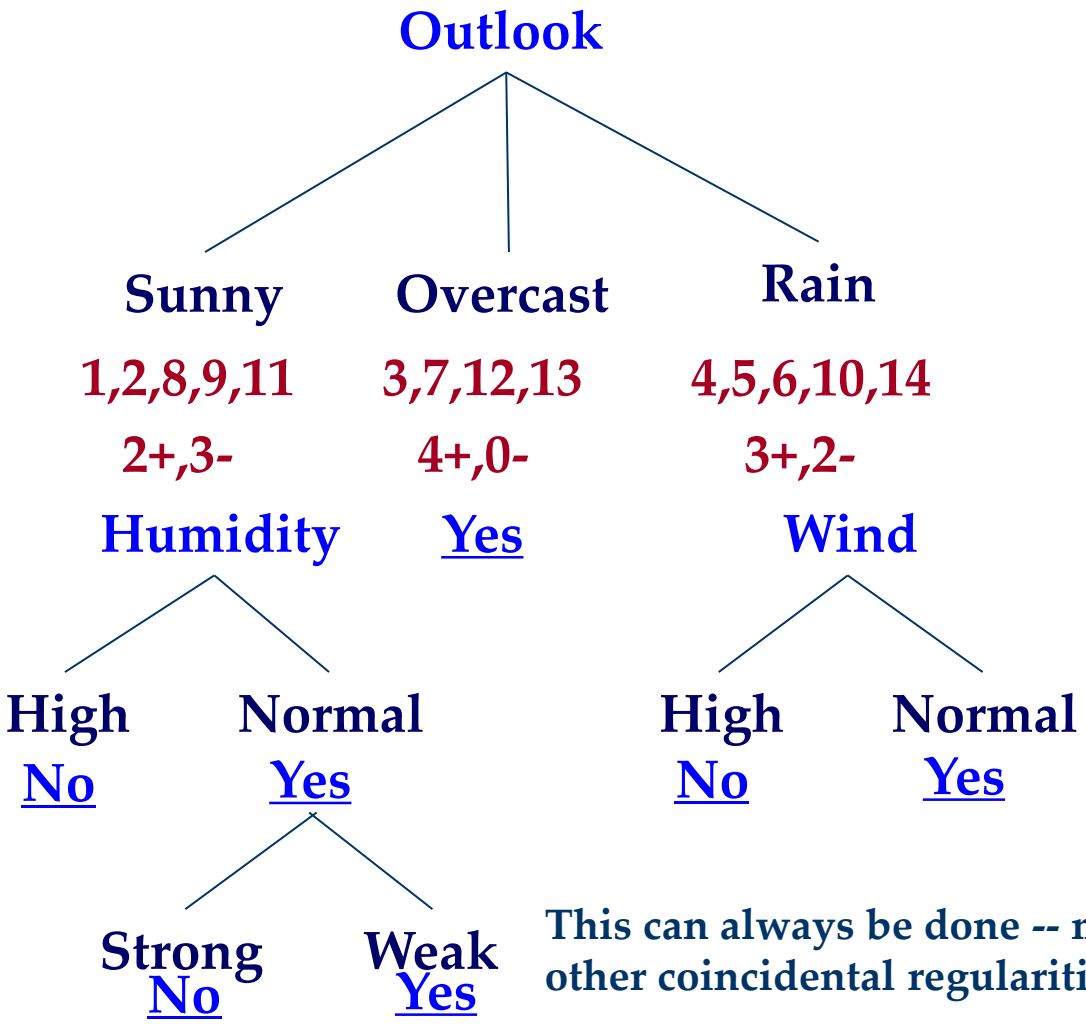
Return Root

History of Decision Tree Research

- Hunt and colleagues in Psychology used full search decision trees methods to model human concept learning in the 60's
- Quinlan developed ID3, with the information gain heuristics in the late 70's to learn expert systems from examples
- Breiman, Friedmans and colleagues in statistics developed CART (classification and regression trees) simultaneously
- A variety of improvements in the 80's: coping with noise, continuous attributes, missing data, non-axis parallel etc.
- Quinlan's updated algorithm, C4.5 (1993) is commonly used (New:C5)
- Boosting (or Bagging) over DTs is a very good general purpose algorithm

Decision Trees: Low Bias Model

Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, NO



This can always be done -- may fit noise or other coincidental regularities

Avoid Overfitting

- How the overfitting can this be avoided with linear classifiers?

- For Decision Trees: Two basic approaches
  - Prepruning:
    - Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.
  - Postpruning:
    - Grow the full tree and then remove nodes that seem not to have sufficient evidence.

- Validation Set

Classification and Regression Trees (CART)

Problems with ID3

- It is only for classification
- The features must be discrete
- CART is designed to solve these two problems

Example

- We have a regression problem.
- Training data: $\{\boldsymbol{x}_1, y_1\}, \{\boldsymbol{x}_2, y_2\}, \cdots \{\boldsymbol{x}_n, y_n\}$
- Two real value features x_1 and x_2

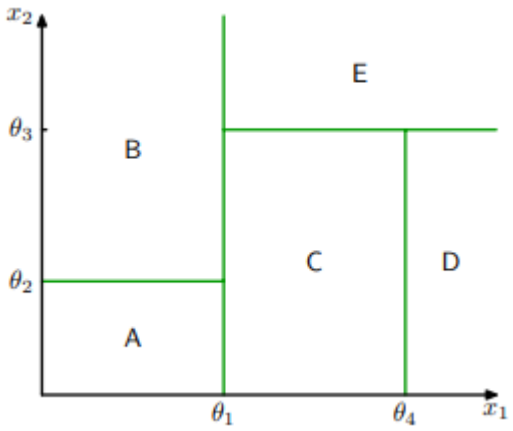
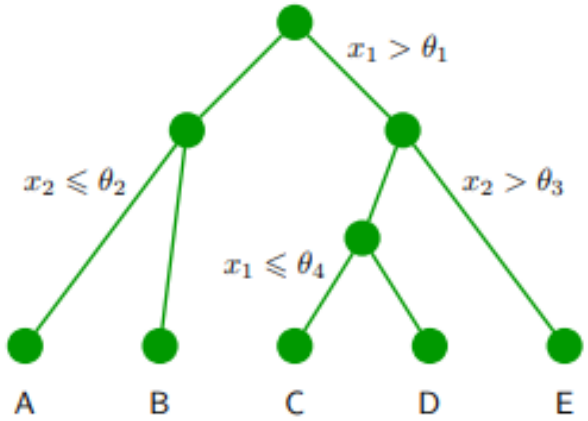


Illustration of a two-dimensional input space that has been partitioned into five regions using axis-aligned boundaries



Binary tree corresponding to the partitioning of input space

Example

- Within each region, there is a separate model to predict the target variable. For instance, in regression we might simply predict a constant over each region, or in classification we might assign each region to a specific class.

$$f(\boldsymbol{x}) = \sum_{\tau=1}^{M} c_\tau I(\boldsymbol{x} \in \mathcal{R}_\tau)$$

- Question 1:
- What value (response) should we assign for each region?
- Aiming at achieving the lowest sum-of-squares error.
- The optimal prediction for region $\mathcal{R}_\tau$ is then given by

$$c_\tau = \frac{1}{N_\tau} \sum_{\boldsymbol{x}_i \in \mathcal{R}_\tau} y_i \qquad Q_\tau(T) = \sum_{\boldsymbol{x}_i \in \mathcal{R}_\tau} (y_i - c_\tau)^2$$

Example

- Question 2:
- How to build a tree ?
- Even for a fixed number of nodes in the tree, the problem of determining the optimal structure (including choice of input variable for each split as well as the corresponding thresholds) to minimize the sum-of-squares error is usually computationally infeasible due to the · large number of possible solutions.
- We have to use a greedy solution:
- Question 3: How to pick the best feature and the best thresholds so far?
- Assume we pick the feature $x^{(j)}$ as the splitting variable and s as the splitting point, we have two regions

$$\mathcal{R}_1(j,s) = \{\boldsymbol{x}|x^{(j)} \leq s\} \qquad \mathcal{R}_2(j,s) = \{\boldsymbol{x}|x^{(j)} > s\}$$

Example

- Assume we pick the feature $x^{(j)}$ as the splitting variable and s as the splitting point, we have two regions

$$\mathcal{R}_1(j,s) = \{\boldsymbol{x}|x^{(j)} \leq s\} \qquad \mathcal{R}_2(j,s) = \{\boldsymbol{x}|x^{(j)} > s\}$$

$$\min_{j,s}\left[\min_{c_1}\sum_{\boldsymbol{x}_i\in\mathcal{R}_1(j,s)}(y_i - c_1)^2 + \min_{c_2}\sum_{\boldsymbol{x}_i\in\mathcal{R}_2(j,s)}(y_i - c_2)^2\right]$$

- Pick a feature, sort all $\boldsymbol{x}$ according to this feature.
- Divide $\boldsymbol{x}$ into two regions, compute c and calculate the above equation, find the best s
- Enumerate all the features, find the best feature, the best (j, s) pair and we now have two regions.
- For each region, repeat until meet the stop conditions.

When to Stop?

- We have M regions finally, and the tree can be represented as

$$f(\boldsymbol{x}) = \sum_{\tau=1}^{M} c_\tau I(\boldsymbol{x} \in \mathcal{R}_\tau)$$

- Possible solution: the reduction in residual error falls below some threshold.
- However, it is found empirically that often
  - None of the available splits produces a significant reduction in error,
  - After several more splits a substantial error reduction is found.
- It is common practice to grow a large tree, using a stopping criterion based on the number of data points associated with the leaf nodes, and then prune back the resulting tree.

How to prune?

- The pruning is based on a criterion that balances residual error against a measure of model complexity.

$$f(\boldsymbol{x}) = \sum_{\tau=1}^{M} c_\tau I(\boldsymbol{x} \in \mathcal{R}_\tau)$$

$$c_\tau = \frac{1}{N_\tau} \sum_{\boldsymbol{x}_i \in \mathcal{R}_\tau} y_i \qquad Q_\tau(T) = \sum_{\boldsymbol{x}_i \in \mathcal{R}_\tau} (y_i - c_\tau)^2$$

- Pruning criterion:

$$C(T) = \sum_{T=1}^{|T|} Q_\tau(T) + \lambda |T|$$

- The regularization parameter λ determines the trade-off between the overall residual sum-of-squares error and the complexity of the model as measured by the number $|T|$ of leaf nodes, and its value is chosen by cross-validation.

CART For Classification

- For classification problems , we just use Entropy or Gini index instead of residual sum-of-squares:
- If we define $p_{\tau k}$ to be the proportion of data points in region $\mathcal{R}_\tau$ assigned to class k, where $k = 1, \ldots, K$, then the entropy:

$$Q_\tau(T) = -\sum_{k=1}^{K} p_{\tau k} \log p_{\tau k}$$

- and the Gini index

$$Q_\tau(T) = \sum_{k=1}^{K} p_{\tau k}(1 - p_{\tau k})$$

- These both vanish for $p_{\tau k} = 0$ and $p_{\tau k} = 1$ and have a maximum at $p_{\tau k} = 0.5$. They encourage the formation of regions in which a high proportion of the data points are assigned to one class.