Decision Trees & Time Complexity

Nipun Batra and teaching staff

IIT Gandhinagar

August 21, 2025

Table of Contents

ID3 Algorithm for Decision Tree Building

ID3 (Examples, Target Attribute, Attributes)

- Create a root node for tree
- If all examples are +/-, return root with label = +/-
- If attributes = empty, return root with most common value of Target Attribute in Examples
- Begin
  - A $\leftarrow$ attribute from Attributes which best classifies Examples
  - Root $\leftarrow$ A
  - For each value (v) of A
    - Add new tree branch : A = v
    - Examples_v: subset of examples that A = v
    - If Examples_v is empty: add leaf with label = most common value of Target Attribute
    - Else: ID3 (Examples_v, Target attribute, Attributes - A)

Training Data

Day	Outlook	Temp	Humidity	Windy	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Entropy calculated

We have 14 examples in S: 5 No, 9 Yes

$$\text{Entropy}(S) = -p_{\text{No}} \log_2 p_{\text{No}} - p_{\text{Yes}} \log_2 p_{\text{Yes}}$$
$$= -\frac{5}{14} \log_2 \left(\frac{5}{14}\right) - \frac{9}{14} \log_2 \left(\frac{9}{14}\right) = 0.940$$

Information Gain for Outlook

Outlook	Play
Sunny	No
Sunny	No
Overcast	Yes
Rain	Yes
Rain	Yes
Rain	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rain	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rain	No

Information Gain for Outlook

Outlook	Play
Sunny	No
Sunny	No
Sunny	No
Sunny	Yes
Sunny	Yes

We have 2 Yes, 3 No Entropy $= -\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right) = 0.971$

Outlook	Play
Overcast	Yes
Overcast	Yes
Overcast	Yes
Overcast	Yes

We have 4 Yes, 0 No Entropy $= 0$ (pure subset)

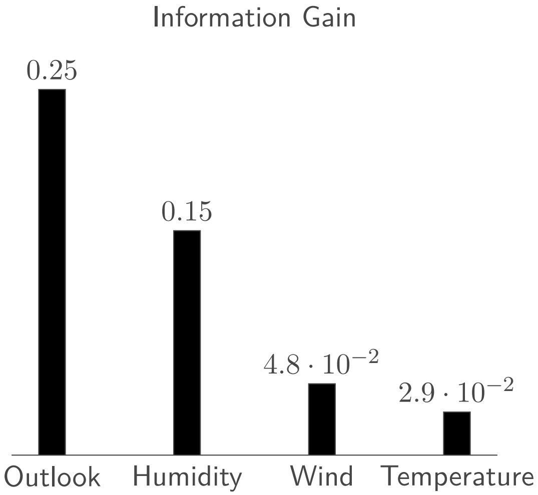
Outlook	Play
Rain	Yes
Rain	Yes
Rain	No
Rain	Yes
Rain	No

We have 3 Yes, 2 No Entropy $= -\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right) = 0.971$

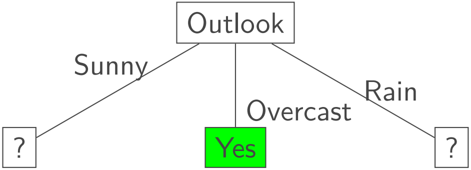
Information Gain

$$
\begin{aligned}
\mathrm{Gain}(S, \mathsf{Outlook}) &= \mathsf{Entropy}(S) - \\
&\sum_{v \in \{\mathsf{Rain,\ Sunny,\ Overcast}\}} \frac{|S_v|}{|S|} \mathsf{Entropy}(S_v) \\
&= 0.940 - \frac{5}{14} \times 0.971 - \frac{4}{14} \times 0 - \frac{5}{14} \times 0.971 \\
&= 0.940 - 0.347 - 0 - 0.347 \\
&= 0.246
\end{aligned}
$$

Information Gain



Learnt Decision Tree

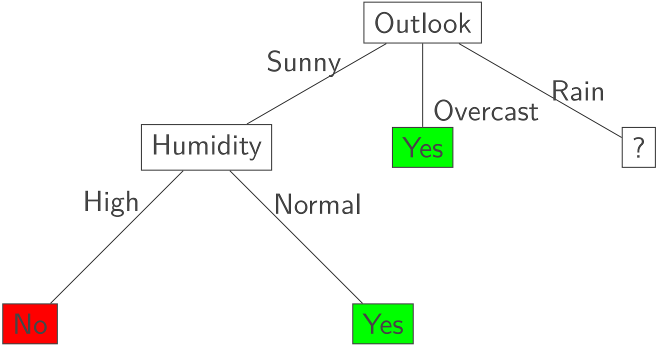


Calling ID3 on Outlook=Sunny

Day	**Temp**	**Humidity**	**Windy**	**Play**
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

- Gain($S_{\text{Outlook=Sunny}}$, Temp) = Entropy(2 Yes, 3 No) - (2/5)*Entropy(0 Yes, 2 No) -(2/5)*Entropy(1 Yes, 1 No) - (1/5)*Entropy(1 Yes, 0 No)
- Gain($S_{\text{Outlook=Sunny}}$, Humidity) = Entropy(2 Yes, 3 No) - (2/5)*Entropy(2 Yes, 0 No) -(3/5)*Entropy(0 Yes, 3 No) $\Longrightarrow$ **maximum possible for the set**
- Gain($S_{\text{Outlook=Sunny}}$, Windy) = Entropy(2 Yes, 3 No) - (3/5)*Entropy(1 Yes, 2 No) -(2/5)*Entropy(1 Yes, 1 No)

Learnt Decision Tree

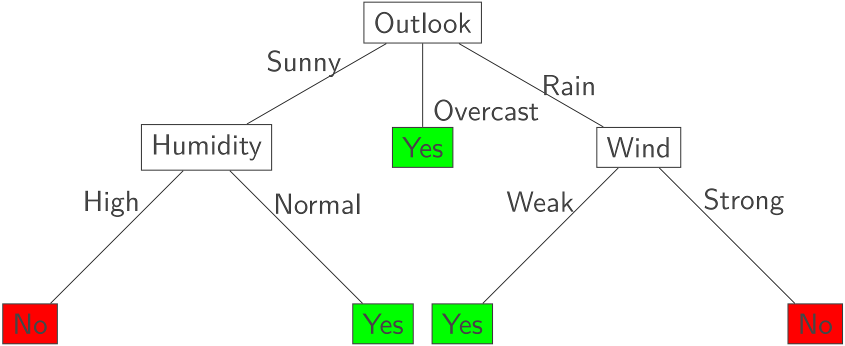


Calling ID3 on (Outlook=Rain)

Day	Temp	Humidity	Windy	Play
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
D10	Mild	Normal	Weak	Yes
D14	Mild	High	Strong	No

- The attribute Windy gives the highest information gain

Learnt Decision Tree



DT Complexity Analysis

Goal & Scope

- **Goal:** Build intuition for how Decision Tree runtime scales.

Goal & Scope

- **Goal:** Build intuition for how Decision Tree runtime scales.
- **Scope:** Real inputs $\rightarrow$

Goal & Scope

- **Goal:** Build intuition for how Decision Tree runtime scales.
- **Scope:** Real inputs $\rightarrow$
    1. Discrete output (classification: Gini/Entropy)

Goal & Scope

- **Goal:** Build intuition for how Decision Tree runtime scales.
- **Scope:** Real inputs $\rightarrow$
  1. Discrete output (classification: Gini/Entropy)
  2. Real output (regression: MSE)

Goal & Scope

- **Goal:** Build intuition for how Decision Tree runtime scales.
- **Scope:** Real inputs $\rightarrow$
  1. Discrete output (classification: Gini/Entropy)
  2. Real output (regression: MSE)
- **Symbols:**

Goal & Scope

- **Goal:** Build intuition for how Decision Tree runtime scales.
- **Scope:** Real inputs $\rightarrow$
  1. Discrete output (classification: Gini/Entropy)
  2. Real output (regression: MSE)
- **Symbols:**
  - $N =$ number of samples

Goal & Scope

- **Goal:** Build intuition for how Decision Tree runtime scales.
- **Scope:** Real inputs $\rightarrow$
  1. Discrete output (classification: Gini/Entropy)
  2. Real output (regression: MSE)
- **Symbols:**
  - $N =$ number of samples
  - $M =$ number of features

Goal & Scope

- **Goal:** Build intuition for how Decision Tree runtime scales.
- **Scope:** Real inputs $\rightarrow$
  1. Discrete output (classification: Gini/Entropy)
  2. Real output (regression: MSE)
- **Symbols:**
  - $N =$ number of samples
  - $M =$ number of features
  - $h =$ tree depth

Goal & Scope

- **Goal:** Build intuition for how Decision Tree runtime scales.
- **Scope:** Real inputs $\rightarrow$
  1. Discrete output (classification: Gini/Entropy)
  2. Real output (regression: MSE)
- **Symbols:**
  - $N =$ number of samples
  - $M =$ number of features
  - $h =$ tree depth
  - $S =$ samples at a given node

Work at One Node

- We test axis-aligned splits: thresholds $x_j \leq t$ for each feature j.

Work at One Node

- We test axis-aligned splits: thresholds $x_j \leq t$ for each feature j.
- For a node with S samples:

Work at One Node

- We test axis-aligned splits: thresholds $x_j \leq t$ for each feature j.
- For a node with S samples:
  - Sort values of feature j (or reuse sorted order).

Work at One Node

- We test axis-aligned splits: thresholds $x_j \leq t$ for each feature j.
- For a node with S samples:
  - Sort values of feature j (or reuse sorted order).
  - Consider up to $S - 1$ midpoints as candidate thresholds.

Work at One Node

- We test axis-aligned splits: thresholds $x_j \leq t$ for each feature j.
- For a node with S samples:
  - Sort values of feature j (or reuse sorted order).
  - Consider up to $S - 1$ midpoints as candidate thresholds.
  - Sweep left $\rightarrow$ right, updating split quality.

Work at One Node

- We test axis-aligned splits: thresholds $x_j \leq t$ for each feature j.
- For a node with S samples:
  - Sort values of feature j (or reuse sorted order).
  - Consider up to $S - 1$ midpoints as candidate thresholds.
  - Sweep left $\rightarrow$ right, updating split quality.
- Costs per feature at a node:

Work at One Node

- We test axis-aligned splits: thresholds $x_j \leq t$ for each feature j.
- For a node with S samples:
  - Sort values of feature j (or reuse sorted order).
  - Consider up to $S - 1$ midpoints as candidate thresholds.
  - Sweep left $\rightarrow$ right, updating split quality.
- Costs per feature at a node:
  - With sorted order: $O(S)$ (one linear sweep).

Work at One Node

- We test axis-aligned splits: thresholds $x_j \leq t$ for each feature j.
- For a node with S samples:
  - Sort values of feature j (or reuse sorted order).
  - Consider up to $S - 1$ midpoints as candidate thresholds.
  - Sweep left $\rightarrow$ right, updating split quality.
- Costs per feature at a node:
  - With sorted order: $O(S)$ (one linear sweep).
  - If sort here: $O(S \log S)$ sort $+$ $O(S)$ scan $\Rightarrow$ $O(S \log S)$.

Work at One Node

- We test axis-aligned splits: thresholds $x_j \leq t$ for each feature j.
- For a node with S samples:
  - Sort values of feature j (or reuse sorted order).
  - Consider up to $S - 1$ midpoints as candidate thresholds.
  - Sweep left $\rightarrow$ right, updating split quality.
- Costs per feature at a node:
  - With sorted order: $O(S)$ (one linear sweep).
  - If sort here: $O(S \log S)$ sort $+$ $O(S)$ scan $\Rightarrow O(S \log S)$.
- Over all M features: $O(MS)$ (presorted) vs. $O(MS \log S)$ (naïve).

Training Data (Real Features)

Day	Temperature	Humidity	Wind Speed	Play
D1	34.2	85	6.2	No
D2	33.5	88	14.3	No
D3	31.8	90	5.7	Yes
D4	27.6	80	7.5	Yes
D5	23.9	65	9.1	Yes
D6	22.5	70	15.0	No
D7	24.1	60	13.5	Yes
D8	29.8	82	6.8	No
D9	25.3	72	7.0	Yes
D10	26.7	68	8.2	Yes
D11	28.9	75	12.0	Yes
D12	30.5	89	11.5	Yes
D13	32.0	70	5.9	Yes
D14	27.2	85	13.2	No

Training Complexity: Efficient Implementation

- Presort once per feature at the root: $O(MN \log N)$.

Training Complexity: Efficient Implementation

- Presort once per feature at the root: $O(MN \log N)$.
- Children inherit sorted order by partitioning.

Training Complexity: Efficient Implementation

- Presort once per feature at the root: $O(MN \log N)$.
- Children inherit sorted order by partitioning.
- At any level, total samples processed $= N$.

Training Complexity: Efficient Implementation

- Presort once per feature at the root: $O(MN \log N)$.
- Children inherit sorted order by partitioning.
- At any level, total samples processed $= N$.
- Over depth h (balanced $\approx \log_2 N$): total mass $= N \cdot h$.

Training Complexity: Efficient Implementation

- Presort once per feature at the root: $O(MN \log N)$.
- Children inherit sorted order by partitioning.
- At any level, total samples processed $= N$.
- Over depth h (balanced $\approx \log_2 N$): total mass $= N \cdot h$.
- Per level: $O(MN)$, across h levels: $O(MNh)$.

Training Complexity: Efficient Implementation

- Presort once per feature at the root: $O(MN \log N)$.
- Children inherit sorted order by partitioning.
- At any level, total samples processed $= N$.
- Over depth h (balanced $\approx \log_2 N$): total mass $= N \cdot h$.
- Per level: $O(MN)$, across h levels: $O(MNh)$.
- Add presort $O(MN \log N) \Rightarrow$ overall $O(MN \log N)$.

Training Complexity: Efficient Implementation

- Presort once per feature at the root: $O(MN \log N)$.
- Children inherit sorted order by partitioning.
- At any level, total samples processed $= N$.
- Over depth h (balanced $\approx \log_2 N$): total mass $= N \cdot h$.
- Per level: $O(MN)$, across h levels: $O(MNh)$.
- Add presort $O(MN \log N) \Rightarrow$ overall $O(MN \log N)$.

Result (both tasks): Train $= O(MN \log N)$.

Training Complexity: Naïve (Re-sorting)

- If re-sort at each node:

Training Complexity: Naïve (Re-sorting)

- If re-sort at each node:
  - Node cost: $O(MS \log S)$.

Training Complexity: Naïve (Re-sorting)

- If re-sort at each node:
  - Node cost: $O(MS \log S)$.
  - Level ℓ: 2^ℓ nodes of size $\approx N/2^\ell$.

Training Complexity: Naïve (Re-sorting)

- If re-sort at each node:
  - Node cost: $O(MS \log S)$.
  - Level ℓ: 2^ℓ nodes of size $\approx N/2^\ell$.
  - Level cost: $MN(\log N - \ell)$.

Training Complexity: Naïve (Re-sorting)

- If re-sort at each node:
  - Node cost: $O(MS \log S)$.
  - Level ℓ: 2^ℓ nodes of size $\approx N/2^\ell$.
  - Level cost: $MN(\log N - \ell)$.
- Sum over $\ell = 0$ to $h - 1$ ($h \approx \log N$):

$$\sum MN(\log N - \ell) = MN \cdot \frac{(\log N)(\log N + 1)}{2}.$$

Training Complexity: Naïve (Re-sorting)

- If re-sort at each node:
  - Node cost: $O(MS \log S)$.
  - Level ℓ: 2^ℓ nodes of size $\approx N/2^\ell$.
  - Level cost: $MN(\log N - \ell)$.
- Sum over $\ell = 0$ to $h - 1$ ($h \approx \log N$):

$$\sum MN(\log N - \ell) = MN \cdot \frac{(\log N)(\log N + 1)}{2}.$$

- Complexity: $O(MN(\log N)^2)$.

Training Complexity: Naïve (Re-sorting)

- If re-sort at each node:
  - Node cost: $O(MS \log S)$.
  - Level ℓ: 2^ℓ nodes of size $\approx N/2^\ell$.
  - Level cost: $MN(\log N - \ell)$.
- Sum over $\ell = 0$ to $h - 1$ ($h \approx \log N$):

$$\sum MN(\log N - \ell) = MN \cdot \frac{(\log N)(\log N + 1)}{2}.$$

- Complexity: $O(MN(\log N)^2)$.

Result: Train $= O(MN(\log N)^2)$ if you re-sort.

Prediction Complexity

- Each test point follows one path root $\rightarrow$ leaf.

Prediction Complexity

- Each test point follows one path root $\rightarrow$ leaf.
- Per sample: $O(h)$ comparisons.

Prediction Complexity

- Each test point follows one path root $\rightarrow$ leaf.
- Per sample: $O(h)$ comparisons.
- For N_{test} points: $O(N_{\text{test}} h)$.

Prediction Complexity

- Each test point follows one path root $\rightarrow$ leaf.
- Per sample: $O(h)$ comparisons.
- For N_{test} points: $O(N_{\text{test}}h)$.
- Balanced: $h \approx \log N \Rightarrow O(N_{\text{test}} \log N)$.

Prediction Complexity

- Each test point follows one path root $\to$ leaf.
- Per sample: $O(h)$ comparisons.
- For N_{test} points: $O(N_{\text{test}} h)$.
- Balanced: $h \approx \log N \Rightarrow O(N_{\text{test}} \log N)$.
- Worst case (unbalanced): $h = \Theta(N) \Rightarrow O(N_{\text{test}} N)$.

Classification vs Regression

- Same asymptotics: $O(MN \log N)$ train, $O(N_{\text{test}} h)$ predict.

Classification vs Regression

- Same asymptotics: $O(MN \log N)$ train, $O(N_{\text{test}} h)$ predict.
- Difference is only in constants:

Classification vs Regression

- Same asymptotics: $O(MN \log N)$ train, $O(N_{\text{test}} h)$ predict.
- Difference is only in constants:
  - Entropy slower than Gini (logs).

Classification vs Regression

- Same asymptotics: $O(MN \log N)$ train, $O(N_{\text{test}} h)$ predict.
- Difference is only in constants:
  - Entropy slower than Gini (logs).
  - MSE similar to Gini.

Classification vs Regression

- Same asymptotics: $O(MN \log N)$ train, $O(N_{\text{test}} h)$ predict.
- Difference is only in constants:
  - Entropy slower than Gini (logs).
  - MSE similar to Gini.
- Binary features: no sorting needed, but per-level scan still linear. What about fixed discrete features?

APPENDIX: Why the Sweep is Linear

Classification (real $\rightarrow$ discrete):

- Maintain class counts (c_L, c_R).

APPENDIX: Why the Sweep is Linear

Classification (real $\rightarrow$ discrete):

- Maintain class counts (c_L, c_R).
- Update counts in $O(1)$ as samples shift.

APPENDIX: Why the Sweep is Linear

Classification (real $\rightarrow$ discrete):

- Maintain class counts (c_L, c_R).
- Update counts in $O(1)$ as samples shift.
- Compute Gini $= 1 - \sum p_k^2$ or Entropy $= -\sum p_k \log p_k$ in $O(1)$.

APPENDIX: Why the Sweep is Linear

Classification (real $\rightarrow$ discrete):

- Maintain class counts (c_L, c_R).
- Update counts in $O(1)$ as samples shift.
- Compute Gini $= 1 - \sum p_k^2$ or Entropy $= -\sum p_k \log p_k$ in $O(1)$.
- $\Rightarrow$ Whole sweep per feature: $O(S)$.

APPENDIX: Why the Sweep is Linear

Classification (real $\to$ discrete):

- Maintain class counts (c_L, c_R).
- Update counts in $O(1)$ as samples shift.
- Compute Gini $= 1 - \sum p_k^2$ or Entropy $= -\sum p_k \log p_k$ in $O(1)$.
- $\Rightarrow$ Whole sweep per feature: $O(S)$.

Regression (real $\to$ real):

- Maintain $(\sum y, \sum y^2, n)$ for left; right $=$ totals $-$ left.

APPENDIX: Why the Sweep is Linear

Classification (real $\to$ discrete):

- Maintain class counts (c_L, c_R).
- Update counts in $O(1)$ as samples shift.
- Compute Gini $= 1 - \sum p_k^2$ or Entropy $= -\sum p_k \log p_k$ in $O(1)$.
- $\Rightarrow$ Whole sweep per feature: $O(S)$.

Regression (real $\to$ real):

- Maintain $(\sum y, \sum y^2, n)$ for left; right $=$ totals $-$ left.
- SSE $= \sum y^2 - (\sum y)^2/n$, updated in $O(1)$.

APPENDIX: Why the Sweep is Linear

Classification (real $\rightarrow$ discrete):

- Maintain class counts (c_L, c_R).
- Update counts in $O(1)$ as samples shift.
- Compute Gini $= 1 - \sum p_k^2$ or Entropy $= -\sum p_k \log p_k$ in $O(1)$.
- $\Rightarrow$ Whole sweep per feature: $O(S)$.

Regression (real $\rightarrow$ real):

- Maintain $(\sum y, \sum y^2, n)$ for left; right $=$ totals $-$ left.
- SSE $= \sum y^2 - (\sum y)^2/n$, updated in $O(1)$.
- $\Rightarrow$ Whole sweep per feature: $O(S)$.

APPENDIX: Why the Sweep is Linear

Classification (real $\rightarrow$ discrete):

- Maintain class counts (c_L, c_R).
- Update counts in $O(1)$ as samples shift.
- Compute Gini $= 1 - \sum p_k^2$ or Entropy $= -\sum p_k \log p_k$ in $O(1)$.
- $\Rightarrow$ Whole sweep per feature: $O(S)$.

Regression (real $\rightarrow$ real):

- Maintain $(\sum y, \sum y^2, n)$ for left; right $=$ totals $-$ left.
- SSE $= \sum y^2 - (\sum y)^2/n$, updated in $O(1)$.
- $\Rightarrow$ Whole sweep per feature: $O(S)$.

Takeaway: Both Gini/Entropy and MSE need linear sweeps.