

Tree, Forest, Bagging, Boosting

Tuấn Ngọc, Khoa Nguyên, Thiên Phúc, Phúc Nguyên

Faculty of Computer Science
University of Information Technology (UIT)
Vietnam National University - Ho Chi Minh City (VNU-HCM)

December, 2025

Contents

- 1 Decision Tree
- 2 Các thuật toán xây dựng Decision Tree
- 3 Regularization
- 4 Bagging
- 5 Random Forest
- 6 Boosting

Cây Quyết Định (Decision Tree)

Tổng quan

Cây quyết định là thuật toán thuộc nhóm **Supervised Learning** (Học có giám sát). Điểm mạnh là tính linh hoạt cao:

- **Phân loại (Classification):** Xác định nhãn của dữ liệu.
Ví dụ: *Email là Spam hay Không Spam.*
- **Hồi quy (Regression):** Dự đoán giá trị liên tục.
Ví dụ: *Dự đoán giá nhà, nhiệt độ.*

Cơ chế hoạt động

- Mô phỏng quá trình ra quyết định của con người dưới dạng **sơ đồ hình cây**.
- Chia nhỏ dữ liệu thông qua các chuỗi **câu hỏi (quy tắc)** liên tiếp để đi đến kết luận cuối cùng.

Cấu tạo của Cây Quyết Định

1. Root Node (Nút Gốc)

- Vị trí cao nhất, điểm bắt đầu.
- Đại diện cho toàn bộ tập dữ liệu (dataset).
- Nơi chọn đặc trưng quan trọng nhất để phân chia đầu tiên.

2. Internal Nodes (Nút Quyết Định)

- Các nút nằm giữa Gốc và Lá.
- Thực hiện các "bài kiểm tra"(Test/Rule).
- Ví dụ: $Tuổi > 30?$ hoặc $Thu nhập > 10tr?$

Cấu tạo của Cây Quyết Định

3. Branches (Nhánh)

- Đường nối giữa các nút.
- Thể hiện luồng đi của quyết định (Kết quả của bài kiểm tra).
- Ví dụ: Nhánh "Có"(Yes) hoặc "Không"(No).

4. Leaf Nodes (Nút Lá)

- Điểm cuối cùng, không phân chia thêm.
- Chứa kết quả dự đoán cuối cùng:
 - **Classification:** Nhãn phân loại (VD: Spam/Not Spam).
 - **Regression:** Giá trị số thực (VD: Giá nhà).

Cơ chế hoạt động

① Bắt đầu từ Nút Gốc:

- Đánh giá toàn bộ dữ liệu.
- Chọn đặc trưng (feature) tốt nhất để đặt câu hỏi.

② Đặt câu hỏi Đúng/Sai:

- Tạo quy tắc Yes/No hoặc True/False.
- Mục tiêu: Chia dữ liệu thành các nhóm "tinh khiết" hơn.

③ Phân nhánh:

- **Yes:** Đi theo một nhánh.
- **No:** Đi theo nhánh còn lại.
- Giúp lọc dần đối tượng.

④ Tiếp tục phân chia (Recursive Partitioning):

- Lặp lại quy trình tại các nút con.
- Đặt câu hỏi cụ thể hơn cho từng nhóm nhỏ.
- Thu hẹp phạm vi dữ liệu.

⑤ Đến Nút Lá & Ra quyết định:

- Dừng lại khi: Hết câu hỏi, Đạt độ sâu tối đa, hoặc Dữ liệu thuần nhất.
- Đưa ra dự đoán cuối cùng tại lá.

Tại sao nên dùng Decision Tree?

- **Dễ hiểu và Trực quan:** Cấu trúc giống tư duy con người, người không chuyên cũng hiểu được.
- **Đa năng:** Xử lý tốt cả Phân loại (Classification) và Hồi quy (Regression).
- **Không cần chuẩn hóa:** Không yêu cầu Normalization/Scaling dữ liệu.
- **Xử lý phi tuyến tính:** Mô hình hóa tốt các mối quan hệ phức tạp, không cần giả định đường thẳng.
- **Xử lý dữ liệu thiếu:** Có cơ chế nội tại để làm việc với Missing Values.

Nhược điểm cần lưu ý

Hạn chế và Thách thức

- **Overfitting:** Dễ học vẹt nếu không giới hạn độ sâu (Pruning).
- **Không ổn định (High Variance):** Thay đổi nhỏ ở dữ liệu đầu vào có thể làm thay đổi hoàn toàn cây.
- **Thiên vị đặc trưng nhiều giá trị:** Có xu hướng chọn thuộc tính có nhiều nhóm con (như ID, Số seri).
- **Khó khăn với bài toán XOR:** Khó nắm bắt mối quan hệ chéo phức tạp giữa các biến.
- **Chi phí tính toán cao:** Tốn tài nguyên khi tìm điểm cắt tối ưu trên dữ liệu lớn.

Định Nghĩa:

- Hàm loss tính toán tỷ lệ các mẫu bị phân loại sai tại một nút.
- Nó được dùng để đánh giá độ "vẩn đục"(impurity), nhưng độ nhạy kém hơn Entropy hay Gini.

Công Thức:

$$\text{Error} = 1 - \max(p_i)$$

Trong đó:

- p_i : tỉ lệ mẫu thuộc class i .
- $\max(p_i)$: tỷ lệ của class chiếm đa số.

Vai trò trong Cắt tỉa cây (Pruning)

Mặc dù ít dùng để "trồng cây" (chia nút), nhưng nó lại rất hữu ích trong việc **Pruning**: Sau khi cây đã xây dựng hoàn chỉnh (thường bị *overfitting*), người ta dùng Misclassification Loss để đánh giá xem việc cắt bỏ một nhánh con có làm giảm **độ chính xác tổng thể** đáng kể hay không.

→ Lý do: Lúc này ta quan tâm đến kết quả dự đoán cuối cùng (đúng/sai) hơn là độ tinh khiết thống kê.

Cross Entropy Loss

Định nghĩa và Công thức Toán học

Khái niệm cơ bản

Cross Entropy đo lường sự khác biệt (khoảng cách) giữa hai phân phối xác suất:

- **Phân phối thực tế (y):** Là nhãn đúng (Ground truth).
- **Phân phối dự đoán (\hat{y}):** Là xác suất mô hình đưa ra.

Công thức cho bài toán phân loại nhị phân (Binary):

$$\mathcal{L} = - \left[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y}) \right]$$

Trong đó:

- $y \in \{0, 1\}$: Nhãn thực tế.
- $\hat{y} \in (0, 1)$: Xác suất dự đoán.
- \log : Logarit tự nhiên (thường ký hiệu là log trong ML).

Cross Entropy Loss

Cơ chế hoạt động và Sự trừng phạt

- **Đo lường sự tự tin:** Cross Entropy không chỉ quan tâm đến việc dự đoán đúng sai, mà còn quan tâm đến việc mô hình *tự tin đến mức nào* về dự đoán đó.

Cơ chế trừng phạt (Penalty)

Cross Entropy **trừng phạt rất nặng** những dự đoán sai mà lại có độ tin cậy cao.

Ví dụ minh họa (Nhãn thực tế $y = 1$):

Dự đoán (\hat{y})	Độ tự tin	Loss	Đánh giá
0.95	Cao (Đúng)	≈ 0.05	Tốt
0.50	Lưỡng lự	≈ 0.69	Trung bình
0.01	Cao (SAI)	≈ 4.60	Bị phạt cực nặng

*Khi \hat{y} tiến về 0 (đoán sai hoàn toàn), giá trị $-\ln(\hat{y})$ tiến về dương vô cùng.

- **ID3** (Iterative Dichotomizer 3) sử dụng cách tiếp cận **tham lam (greedy)** để chọn ra thuộc tính tốt nhất cho mỗi bước phân chia.
- Thuật toán chia dữ liệu vào các nút con (child nodes) dựa trên giá trị của thuộc tính, sau đó đệ quy áp dụng cho từng nút con.
- ID3 sử dụng hàm **Entropy** và **Information Gain** để đo độ tinh khiết, nhằm quyết định phép chia nào là tối ưu.

Hàm Số Entropy

Định Nghĩa:

- Entropy đo lường mức độ hỗn loạn hoặc không chắc chắn của dữ liệu.
- Giá trị thấp nhất (0) đạt được khi dữ liệu hoàn toàn tinh khiết (cùng 1 class).

Công Thức:

$$H(S) = - \sum_{c=1}^C \frac{N_c}{N} \log \left(\frac{N_c}{N} \right)$$

Ký hiệu:

- S : Tập dữ liệu tại nút hiện tại.
- N : Tổng số phần tử trong S .
- N_c : Số phần tử thuộc class c .
- Quy ước: $0 \log 0 = 0$.

Entropy Có Trọng Số (Conditional Entropy)

Công thức tổng quát

Tổng Entropy của các nút con sau khi chia theo thuộc tính x :

$$H(x, S) = \sum_{k=1}^K \frac{m_k}{N} H(S_k)$$

Giải thích các thành phần:

- $H(x, S)$: Độ hỗn loạn trung bình của hệ thống sau khi chia.
- N : Tổng số mẫu dữ liệu trong tập cha (S) trước khi chia.
- m_k : Số lượng mẫu dữ liệu rơi vào nhánh con thứ k .
- S_k : Tập hợp con các mẫu tại nhánh thứ k .
- $\frac{m_k}{N}$: Trọng số của nhánh k (nhánh càng lớn càng ảnh hưởng nhiều).

Information Gain & Quy Tắc Chọn Thuộc Tính

Information Gain (Lượng Tin Tức Thu Được)

Đo lường mức độ giảm độ nhiễu (Entropy) sau khi biết giá trị của thuộc tính x .

$$G(x, S) = H(S) - H(x, S)$$

(Lấy Entropy ban đầu trừ đi Entropy sau khi chia)

Quy Tắc Chọn Thuộc Tính Tối Ưu

ID3 hoạt động theo nguyên tắc tham lam (Greedy). Nó sẽ chọn thuộc tính x^* sao cho Gain là lớn nhất:

$$x^* = \operatorname{argmax}_x G(x, S)$$

⇒ Giúp cây quyết định "tinh khiết" nhanh nhất.

Dữ Liệu Ban Đầu & Entropy Gốc

Bảng dữ liệu:

ID	Outlook	Temp	Humidity	Wind	Play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rainy	mild	high	strong	no

Tính Entropy tại Root Node ($H(S)$)

Công Thức Entropy tại Root Node $H(S)$:

Dữ Liệu Thống Kê

- Tổng số mẫu (N): 14
- Số mẫu lớp 'No' (N_{No}): 5
- Số mẫu lớp 'Yes' (N_{Yes}): 9

Ghi chú:

- $p_{No} = 5/14 \approx 0.357$
- $p_{Yes} = 9/14 \approx 0.643$

$$H(S) = -p_{No} \log(p_{No}) - p_{Yes} \log(p_{Yes})$$

Thay số:

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

$$H(S) \approx 0.65$$

Phân Tích Thuộc Tính "Outlook"

Tách theo giá trị của Outlook:

Sunny

- Tổng: 5 (2 Yes, 3 No)
- $H(S_{Sunny}) \approx 0.673$

Overcast

- Tổng: 4 (4 Yes, 0 No)
- $H(S_{Overcast}) = 0$
- (Tinh khiết)

Rainy

- Tổng: 5 (3 Yes, 2 No)
- $H(S_{Rainy}) \approx 0.673$

Entropy Có Trọng Số Của Outlook:

$$H(\text{Outlook}, S) = \frac{5}{14}(0.673) + \frac{4}{14}(0) + \frac{5}{14}(0.673) \approx 0.48$$

So Sánh Entropy Sau Khi Chia

Information Gain lớn nhất \iff Conditional Entropy nhỏ nhất

- **Outlook:** $H(x, S) \approx 0.48$ ($\implies G \approx 0.17$)
- **Humidity:** $H(x, S) \approx 0.547$ ($\implies G \approx 0.103$)
- **Wind:** $H(x, S) \approx 0.618$ ($\implies G \approx 0.032$)
- **Temperature:** $H(x, S) \approx 0.631$ ($\implies G \approx 0.019$)

\Rightarrow **Outlook** được chọn làm Root Node.

Phân tích:

- Quan sát dữ liệu khi Outlook = **Overcast**.
- Entropy hiện tại: 0 (tinh khiết).
- Tất cả các mẫu đều là **Yes**.

Kết Luận

⇒ Tạo nút lá (Leaf Node): **YES**.

Không tiếp tục phân chia nữa để tránh overfitting.

Trạng thái:

- Entropy: 0.673.
- Xét tiếp: Temp, Humidity, Wind.

Kết quả:

- **Humidity** cho Gain lớn nhất.
- Humidity = High → No.
- Humidity = Normal → Yes.

Dữ liệu con (Sunny)

ID	Outlook	Temp	Humidity	Wind	Play
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

Trạng thái:

- Entropy: 0.673.
- Xét tiếp: Temp, Humidity, Wind.

Kết quả:

- **Wind** cho Gain lớn nhất.
- Wind = Weak → Yes.
- Wind = Strong → No.

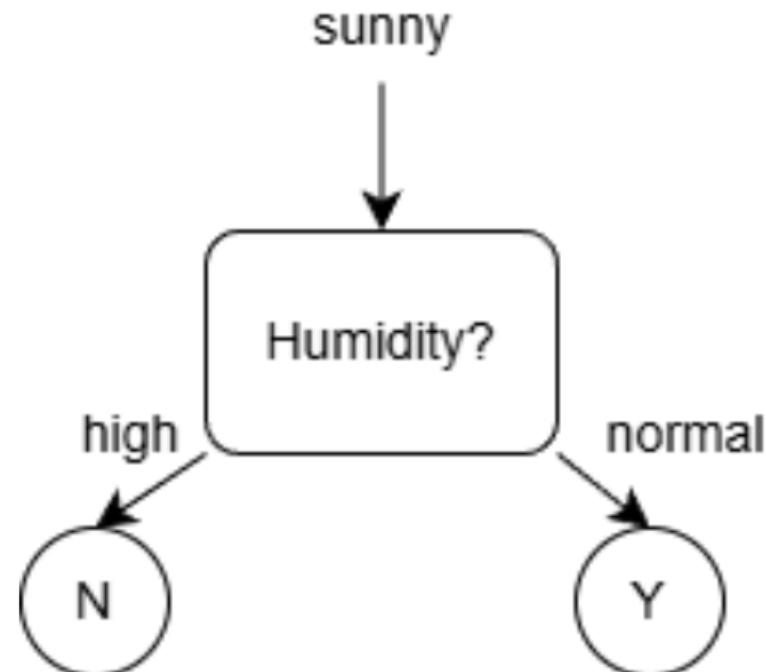
Dữ liệu con (Rainy)

ID	Outlook	Temp	Humidity	Wind	Play
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
10	rainy	mild	normal	weak	yes
14	rainy	mild	high	strong	no

Outlook = Sunny

- If Outlook = **Sunny**: Kiểm tra Humidity.

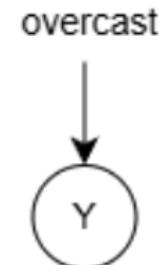
- Humidity ≤ 77.5 (Normal) \rightarrow Yes
- Humidity > 77.5 (High) \rightarrow No



Xây dựng các nhánh con: Overcast và Rainy

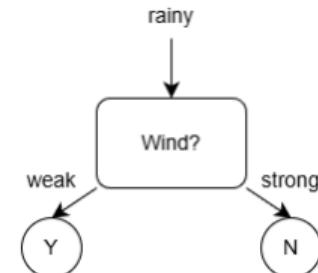
1. Nhánh Outlook = Overcast

- Dữ liệu tại nhánh này hoàn toàn thuần nhất (Pure).
- Tất cả mẫu đều là Yes.
- → Tạo ngay nút lá YES.

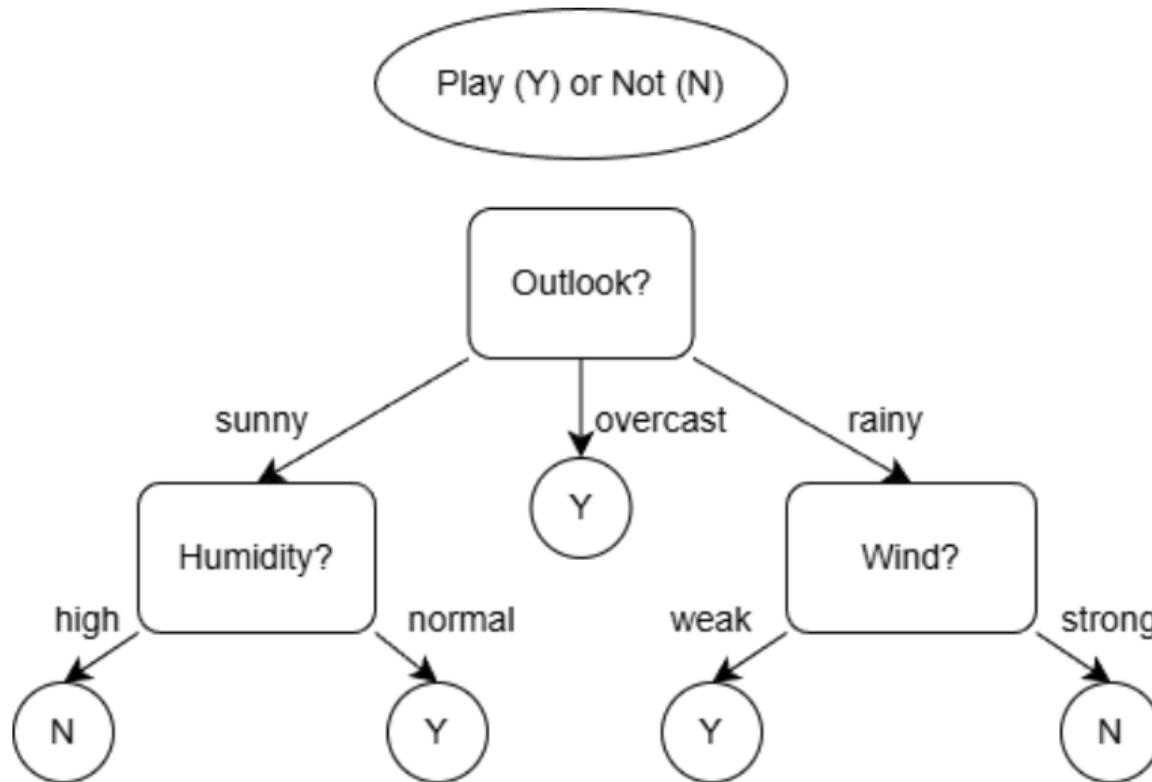


2. Nhánh Outlook = Rainy

- Cần xét tiếp thuộc tính Windy.
 - Windy = No (Weak) → Yes
 - Windy = Yes (Strong) → No



Cây quyết định



Hình 1: Mô hình cây sau khi hoàn tất phân chia

Ưu điểm của ID3

① Dễ hiểu và trực quan:

- Kết quả là các luật If-Then rõ ràng.
- Con người có thể dễ dàng lý giải tại sao mô hình đưa ra quyết định đó (White-box model).

② Tốc độ xây dựng:

- Khá nhanh với các tập dữ liệu kích thước nhỏ và trung bình.
- Chỉ cần tính toán thống kê đơn giản, không cần các phép toán ma trận phức tạp.

③ Xử lý tốt dữ liệu phân loại (Categorical):

- Hoạt động hiệu quả khi các đặc trưng (features) là dạng rời rạc (ví dụ: Nắng, Mưa, U ám).

Vấn đề lớn nhất của Information Gain

ID3 có xu hướng thiên vị các thuộc tính có **nhiều giá trị** (High Cardinality).

Ví dụ: Thuộc tính "Mã số sinh viên"(ID).

- Mỗi sinh viên có 1 ID duy nhất → Chia nhánh theo ID sẽ tạo ra các nhóm con chỉ có 1 phần tử (tinh khiết tuyệt đối).
- Entropy sau khi chia = 0 → Information Gain đạt cực đại.
- **Hậu quả:** Cây quyết định chọn ID làm nút gốc. Cây này vô dụng với dữ liệu mới vì nó chỉ "nhớ" ID cũ.

① Không xử lý được dữ liệu liên tục (Continuous Data):

- ID3 gốc không thể làm việc với các số thực (ví dụ: Lương = 15.5 triệu).
- Cần phải rời rạc hóa dữ liệu trước khi đưa vào (Pre-processing).

② Không xử lý được dữ liệu thiếu (Missing Values):

- Nếu một dòng dữ liệu bị khuyết thuộc tính, ID3 sẽ bỏ qua hoặc gấp lối tính toán Entropy.

③ Nhạy cảm với nhiễu (Outliers):

- Dễ bị thay đổi cấu trúc lớn chỉ vì một vài điểm dữ liệu nhiễu.

Nhược điểm về cấu trúc mô hình

- **Overfitting (Quá khớp):**

- ID3 cố gắng xây dựng cây cho đến khi mọi lá đều tinh khiết ($\text{Entropy} = 0$).
- Đến đến cây quá sâu, phức tạp, học thuộc lòng dữ liệu huấn luyện nhưng dự đoán kém trên dữ liệu kiểm thử.
- *Giải pháp:* Cần cắt tỉa cây (Pruning) - thứ mà ID3 gốc không có.

- **Chỉ tìm kiếm cục bộ (Local Optima):**

- Vì dùng giải thuật tham lam (Greedy), ID3 chọn cái tốt nhất NGAY LÚC ĐÓ.
- Nó không đảm bảo tạo ra cây quyết định tối ưu nhất về tổng thể (Global Optimum).

Cải tiến cốt lõi: Gain Ratio

Tại sao là C4.5?

C4.5 là phiên bản nâng cấp giải quyết các hạn chế nghiêm trọng của ID3, giúp áp dụng hiệu quả trong các bài toán thực tế.

Thay đổi tiêu chí phân chia: Thay vì dùng *Information Gain* (dễ bị thiên vị các thuộc tính nhiều giá trị), C4.5 dùng **Gain Ratio**:

$$GainRatio(S, A) = \frac{Gain(S, A)}{SplitInfo(S, A)}$$

Trong đó **Split Information** là bộ chuẩn hóa:

$$SplitInfo(S, A) = - \sum_{i=1}^k \frac{|S_i|}{|S|} \times \log_2 \left(\frac{|S_i|}{|S|} \right)$$

→ Trừng phạt các thuộc tính có quá nhiều nhánh con (như ID, Mã số), giúp chọn đặc trưng có ý nghĩa thực sự.

So sánh khả năng xử lý dữ liệu: ID3 vs C4.5

Hạn chế của ID3

1. Với dữ liệu liên tục:

- Chỉ làm việc với dữ liệu phân loại (Categorical/Discrete).
- Không xử lý được các con số cụ thể.
- Ví dụ: Chỉ hiểu Nắng/Mưa, không hiểu 35°C .

2. Với dữ liệu thiếu (Missing):

- Gặp lỗi khi tính toán Entropy nếu dữ liệu bị khuyết.
- Yêu cầu dữ liệu đầu vào phải được làm sạch hoàn hảo.

Cải tiến của C4.5

1. Xử lý số thực (Numeric):

- Tự động tìm **ngưỡng (threshold)** tối ưu để chia nhị phân.
- Ví dụ: Chuyển "Nhiệt độ" thành $\text{Temperature} > 28.5$.

2. Xử lý dữ liệu thiếu:

- Có cơ chế nội tại để xử lý:
 - Bỏ qua giá trị thiếu khi tính Gain.
 - Hoặc phân bổ xác suất (weights) vào các nhánh con dựa trên tỷ lệ đã biết.

Xử lý dữ liệu bị thiếu

Vấn đề cốt lõi

Làm thế nào để xử lý các giá trị bị thiếu (NULL/?) mà không làm mất thông tin quan trọng hoặc gây nhiễu mô hình?

Giải pháp "Nội tại" của C4.5

Thay vì đoán mò (imputation) hoặc xóa bỏ dữ liệu, C4.5 coi giá trị bị thiếu là "**có thể rơi vào bất kỳ trường hợp nào**" với xác suất tương ứng dựa trên dữ liệu đã biết.

Quy trình này diễn ra ở 2 giai đoạn chính:

- ① **Khi chọn thuộc tính (Attribute Selection):** Trừng phạt Gain.
- ② **Khi chia dữ liệu (Splitting):** Phân rã mẫu (Fractional Instances).

Giai đoạn Tính Gain (Attribute Selection)

Vấn đề

Làm sao tính Entropy cho cột "Outlook" nếu có dòng bị trống (NULL)?

Giải pháp của C4.5: Trừng phạt bằng tỷ lệ F

- ① **Bước 1:** Tính Information Gain dựa trên các dòng **ĐÃ BIẾT**.

$$Gain_{tam_thoi} = 0.246 \quad (\text{Ví dụ})$$

- ② **Bước 2:** Tính tỷ lệ mẫu đã biết (F). Giả sử tập S có 14 dòng, 1 dòng thiếu Outlook:

$$F = \frac{\text{Số mẫu đã biết}}{\text{Tổng số mẫu}} = \frac{13}{14}$$

- ③ **Bước 3:** Tính Gain thực tế (có trừng phạt).

$$Gain_{real} = Gain_{tam_thoi} \times F = 0.246 \times \frac{13}{14} \approx 0.228$$

→ Ý nghĩa: Thuộc tính càng nhiều dữ liệu thiếu, Gain càng giảm, ít khả năng được chọn làm nút phân chia.

Giai đoạn Phân bổ mẫu (Splitting Data)

Cơ chế: Phân rã mẫu (Fractional Instances)

- Dòng thiếu dữ liệu không đi về một nhánh duy nhất.
- Nó bị "xé nhỏ" và gửi đến tất cả các nhánh với **Trọng số (Weight)**.

Ví dụ minh họa: Dòng 10 bị thiếu Outlook. Thống kê 13 mẫu đã biết:

Sunny (5 mẫu)

Tỷ lệ: $5/13 \approx 38\%$

Overcast (4 mẫu)

Tỷ lệ: $4/13 \approx 31\%$

Rainy (4 mẫu)

Tỷ lệ: $4/13 \approx 31\%$

Nhánh Sunny	Nhánh Overcast	Nhánh Rainy
Nhận thêm: 0.38 dòng	Nhận thêm: 0.31 dòng	Nhận thêm: 0.31 dòng

Hệ quả: Tại các nút con, dòng số 10 tham gia tính Entropy với trọng số lẻ (ví dụ 0.38) thay vì 1.

Ví dụ Minh họa: Phân rã dòng số 10

1. Dữ liệu đầu vào: Giả sử tập dữ liệu có 14 dòng, trong đó dòng số 10 bị thiếu Outlook.

ID	Outlook	Temperature	Play
...
10	? (Missing)	Hot	Yes
...

2. Bảng phân phối trọng số (Weight Distribution): Dựa trên 13 mẫu đã biết: Sunny (5), Overcast (4), Rainy (4).

Nhánh (Branch)	Số mẫu cũ	Trọng số (w)	Tổng mẫu mới tại nút con
Sunny	5	$\frac{5}{13} \approx 0.38$	$5 + 0.38 = 5.38$
Overcast	4	$\frac{4}{13} \approx 0.31$	$4 + 0.31 = 4.31$
Rainy	4	$\frac{4}{13} \approx 0.31$	$4 + 0.31 = 4.31$
Tổng cộng	13	1.0	Tham gia cả 3 nhánh

→ Kết quả: Dòng số 10 không biến mất, mà hiện diện ở cả 3 nhánh với "thân phận" nhỏ hơn.

Tính toán và Chọn Ngưỡng Tối Ưu (1)

1. Sắp xếp dữ liệu (Sorting)

Trước khi tính toán, thuật toán phải sắp xếp các giá trị của thuộc tính liên tục theo thứ tự **tăng dần**. Các ngưỡng cắt (T) thường là trung điểm của hai giá trị liền kề.

2. Tính Gain cho từng ứng viên

Máy tính thực hiện vòng lặp (loop) qua tất cả các ngưỡng T vừa tìm được. Với mỗi ngưỡng, dữ liệu chia làm 2 nhánh:

- **Thử cắt tại $T = 64.5$:** (≤ 64.5 vs > 64.5) $\rightarrow Gain_1 \approx 0.012$
- **Thử cắt tại $T = 71.5$:** (≤ 71.5 vs > 71.5) $\rightarrow Gain_2 \approx 0.005$
- **Thử cắt tại $T = 75.5$:**
 - Nhóm ≤ 75.5 : Tập $\{64, \dots, 75\}$
 - Nhóm > 75.5 : Tập $\{80, \dots, 85\}$
 - $\rightarrow Gain_3 \approx 0.025$ (Cao nhất \rightarrow Chọn)

2. So sánh và Chọn (Selection)

So sánh các giá trị Gain: $0.025 > 0.018 > 0.012 \dots$

→ **Kết luận:** Chọn **75.5** làm ngưỡng cắt cho thuộc tính *Temperature*.

Lưu ý Kỹ thuật: Gain vs Gain Ratio

- ① **Tìm điểm cắt (Nội bộ thuộc tính):** C4.5 dùng **Information Gain**.

Lý do: Tránh chọn các điểm cắt sát mép (tạo ra nhóm quá nhỏ) mà Gain Ratio thường mắc phải.

- ② **Chọn thuộc tính (So sánh Temp vs Outlook):** Sau khi có ngưỡng tối ưu, mới dùng **Gain Ratio** để so sánh các thuộc tính với nhau.

Tập dữ liệu và Entropy Ban đầu

Out	Temp	Hum	Wind	Play
Sunny	85	85	No	No
Sunny	80	90	Yes	No
Overcast	83	78	No	Yes
Rainy	70	96	No	Yes
Rainy	68	80	No	Yes
Rainy	65	70	Yes	No
Overcast	64	65	Yes	Yes

Out	Temp	Hum	Wind	Play
Sunny	72	95	No	No
Sunny	69	70	No	Yes
Rainy	75	80	No	Yes
Sunny	75	70	Yes	Yes
Overcast	72	90	Yes	Yes
Overcast	81	75	No	Yes
Rainy	71	80	Yes	No

Tổng kết dữ liệu: 14 mẫu (9 Yes, 5 No).

Entropy toàn cục (S)

$$\text{Entropy}(S) = - \left(\frac{9}{14} \log \frac{9}{14} + \frac{5}{14} \log \frac{5}{14} \right) \approx \mathbf{0.651}$$

(Lưu ý: Kết quả tính theo log tự nhiên - \ln , hoặc logarit cơ số e)

Phân tích Thuộc tính: Outlook (Categorical)

Thông kê:

- Sunny (5): 2 Yes, 3 No $\rightarrow E = 0.673$
- Overcast (4): 4 Yes, 0 No $\rightarrow E = 0$
- Rainy (5): 3 Yes, 2 No $\rightarrow E = 0.673$

Tính toán:

$$\text{Info Gain} = 0.651 - \left(\frac{5}{14} \times 0.673 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.673 \right) = \mathbf{0.17}$$

$$\text{Split Info} = - \left(\frac{5}{14} \log \frac{5}{14} + \frac{4}{14} \log \frac{4}{14} + \frac{5}{14} \log \frac{5}{14} \right) = \mathbf{1.093}$$

Kết quả Outlook

$$\text{Gain Ratio} = \frac{0.17}{1.093} \approx \mathbf{0.155}$$

Phân tích các Thuộc tính còn lại

Temperature (Numerical)	Humidity (Numerical)	Windy (Categorical)
Best Split: 75.5	Best Split: 82.5	True/False
Gain: 0.018	Gain: 0.072	Gain: 0.034
Split Info: 0.598	Split Info: 0.652	Split Info: 0.683
Ratio: 0.03	Ratio: 0.11	Ratio: 0.049

Lưu ý: Với thuộc tính số (Temp, Humidity), C4.5 đã thực hiện tìm ngưỡng cắt tối ưu trước khi tính Gain Ratio.

So sánh và Chọn Nút Gốc

Bảng xếp hạng Gain Ratio:

Hạng	Thuộc tính	Gain Ratio	Loại
1	Outlook	0.155	Categorical
2	Humidity	0.110	Numerical (≤ 82.5)
3	Windy	0.049	Categorical
4	Temperature	0.030	Numerical (≤ 75.5)

Kết luận

Thuộc tính **Outlook** có Gain Ratio cao nhất (0.155).

→ **Outlook** được chọn làm **Nút Gốc (Root Node)** của cây quyết định.

Xây dựng nhánh con: Outlook = Overcast

Bước 1: Lọc dữ liệu (Filter)

Chỉ giữ lại các dòng có Outlook = Overcast.

Outlook	Temp	Hum	Wind	Play
Overcast	83	78	No	Yes
Overcast	64	65	Yes	Yes
Overcast	72	90	Yes	Yes
Overcast	81	75	No	Yes

Kết luận

Tất cả mẫu đều có nhãn là Yes.

→ Tạo nút lá (Leaf Node): YES.

(Không cần chia tiếp, Entropy = 0)

Xây dựng nhánh con: Outlook = Sunny

Dữ liệu con (Subset): 5 dòng Sunny.

Temp	Humidity	Play
85	85	No
80	90	No
72	95	No
69	70	Yes
75	70	Yes

Phân tích:

- Nhìn vào cột *Humidity*:
- Nhóm > 80 : Toàn bộ **No**.
- Nhóm ≤ 70 : Toàn bộ **Yes**.

Chọn thuộc tính: Humidity.

Ngưỡng cắt (C4.5): 77.5

Quy tắc rẽ nhánh (Humidity)

- Humidity $\leq 77.5 \rightarrow \text{Yes}$ (Lá)
- Humidity $> 77.5 \rightarrow \text{No}$ (Lá)

Xây dựng nhánh con: Outlook = Rainy

Dữ liệu con (Subset): 5 dòng Rainy.

Temp	Windy	Play
70	No	Yes
68	No	Yes
65	Yes	No
75	No	Yes
71	Yes	No

Phân tích:

- Khi $Windy = No$: 3 Yes (Thuần nhất).
- Khi $Windy = Yes$: 2 No (Thuần nhất).

Chọn thuộc tính: Windy.

Quy tắc rẽ nhánh (Windy)

- $Windy = No \rightarrow \text{Yes}$ (Lá)
- $Windy = Yes \rightarrow \text{No}$ (Lá)

Kết quả: Cây Quyết Định C4.5 Hoàn Chỉnh

Tóm tắt các luật (Rules):

① Outlook = Overcast

→ Yes

② Outlook = Sunny

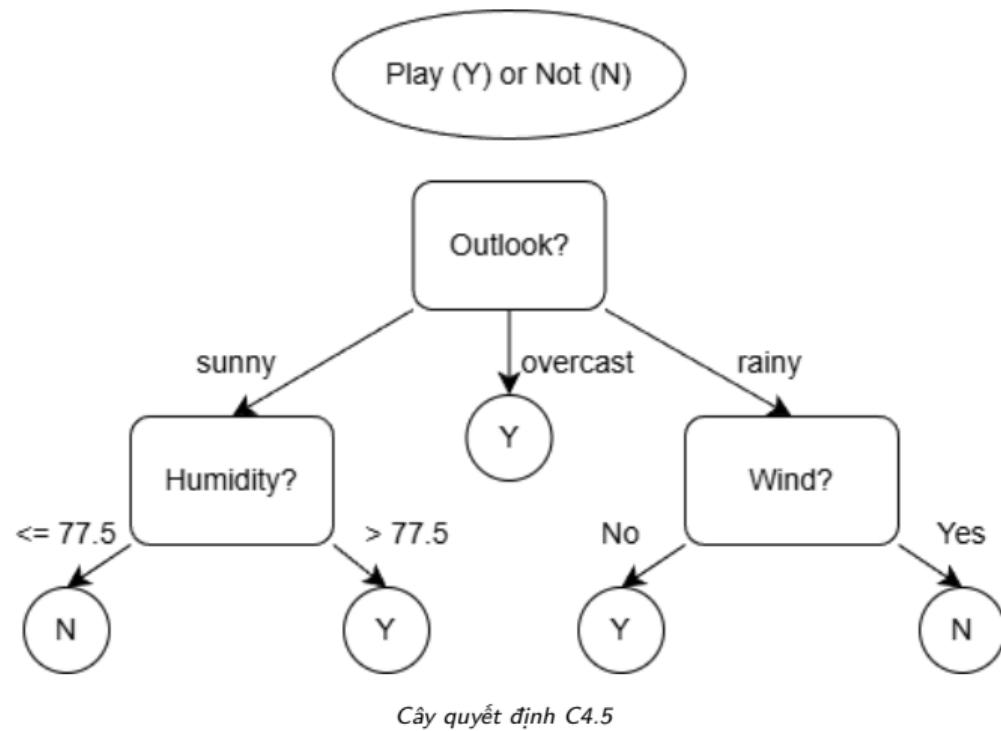
Kiểm tra Humidity:

- $\leq 77.5 \rightarrow$ Yes
- $> 77.5 \rightarrow$ No

③ Outlook = Rainy

Kiểm tra Windy:

- No → Yes
- Yes → No



- **CART** (Classification And Regression Tree) là một biến thể của thuật toán cây quyết định
- **Thuật toán:**
 - **Cấu trúc cây:** CART xây dựng một cấu trúc hình cây bao gồm các nút và nhánh
 - **Tiêu chí chia (splitting criteria):** CART sử dụng phương pháp tham lam (greedy) để chia dữ liệu tại mỗi nút
 - Cây phân loại: sử dụng chỉ số Gini impurity
 - Cây hồi quy: sử dụng mức giảm sai số
- **Điều kiện dừng:**
 - Nút trở nên **thuần khiết** (tất cả mẫu thuộc cùng một lớp)
 - Số lượng mẫu trong nút **nhỏ hơn ngưỡng tối thiểu**
 - **Mức giảm impurity hoặc error** của việc chia tiếp theo **nhỏ hơn một ngưỡng quy định**
 - Đã đạt đến **độ sâu tối đa của cây**
 - Không còn **thuộc tính nào để tiếp tục chia**

CART: Classification Trees

- **Gini Impurity**

- Gini impurity kiểm tra mức độ một mẫu được chọn ngẫu nhiên sẽ bị gán nhãn sai nếu được gán nhãn theo xác suất của các lớp

$$G(t) = 1 - \sum_{i=1}^C p_i^2 \quad \text{với } p_i \text{ là xác xuất của lớp } i$$

$$(\quad G(t) = 1 - \text{"probability of yes}^2 - \text{"probability of no}^2 \quad)$$

- Giá trị Gini càng thấp → nút càng sạch và đồng nhất

- **Total Gini:**

$$G_{\text{total}} = \sum_{j=1}^k \frac{N_j}{N} G_j$$

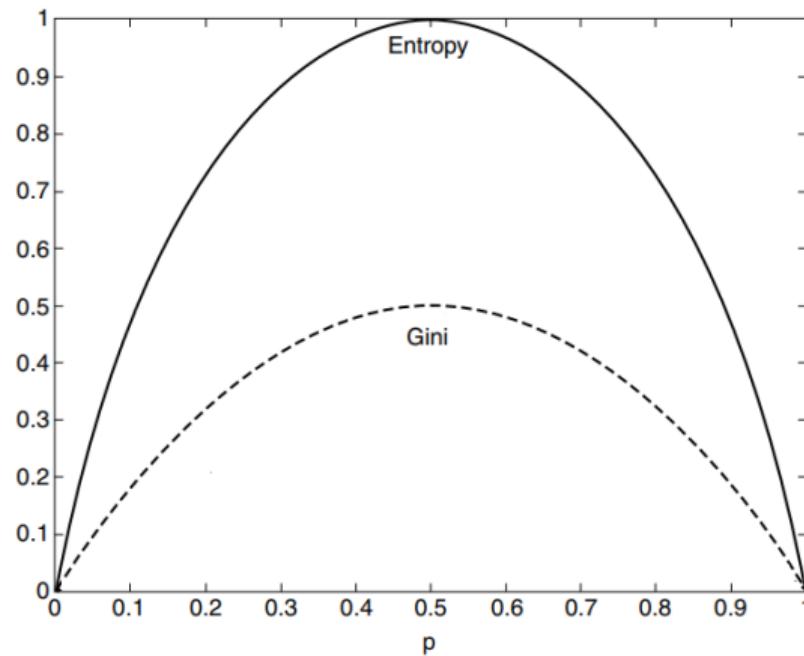
trong đó p_i là xác suất của lớp i , G_j là Gini của nút con thứ j , N_j là số mẫu trong nút đó, và $N = \sum_{j=1}^k N_j$.

CART: Classification Trees

• Gini Impurity and Entropy

Tiêu chí	Gini Impurity	Entropy
Tốc độ huấn luyện	Tính toán nhanh hơn vì không dùng phép log	Chậm hơn một chút do phải dùng phép log
Chia nút	Tạo điểm chia nhanh, thường ưu tiên lớp chiếm đa số	Tạo ra các nút con cân bằng hơn
Kích thước tập dữ liệu	Hiệu quả cho bộ dữ liệu lớn, nhiều chiều	Phù hợp với dữ liệu có cấu trúc và phân bố lớp cân bằng
Độ nhạy với phân phối	Ít nhạy với thay đổi nhỏ trong xác suất	Nhạy hơn với những thay đổi nhỏ trong phân phối lớp

CART: Classification Trees



CART: Building Classification Trees

Outlook	Temp	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

CART: Building Classification Trees

- **Tính Gini Impurity cho từng thuộc tính:**

- $G(\text{Outlook} = \text{Sunny}) = 1 - \left[\left(\frac{2}{5} \right)^2 + \left(\frac{3}{5} \right)^2 \right] = 0.48$
 - $G(\text{Outlook} = \text{Overcast}) = 1 - \left[\left(\frac{4}{4} \right)^2 + \left(\frac{0}{4} \right)^2 \right] = 0$
 - $G(\text{Outlook} = \text{Rain}) = 1 - \left[\left(\frac{3}{5} \right)^2 + \left(\frac{2}{5} \right)^2 \right] = 0.48$
- $$\Rightarrow G_{\text{total}}(\text{Outlook}) = \frac{5}{14} \cdot 0.48 + \frac{4}{14} \cdot 0 + \frac{5}{14} \cdot 0.48 = 0.343$$

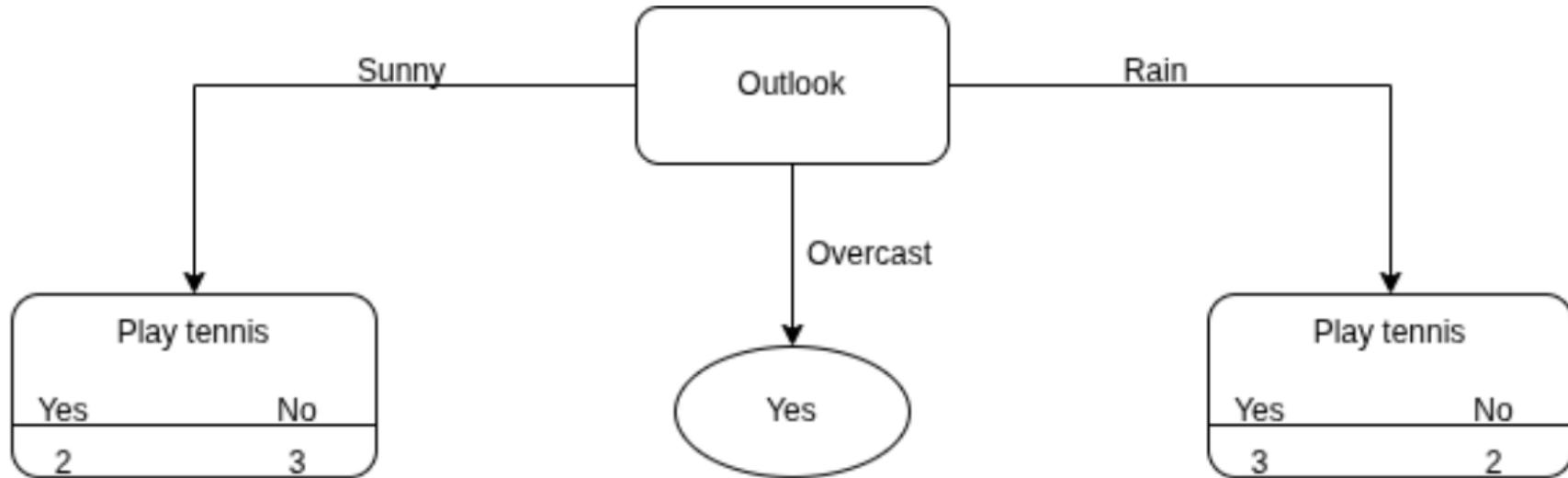
- Tương tự, ta tính cho các thuộc tính khác và có kết quả như sau:

Thuộc tính	Total Gini
Outlook	0.343
Temp	0.367
Humidity	0.394
Wind	0.371

- **Kết quả:** Thuộc tính có Total Gini thấp nhất (Outlook) được chọn làm nút chia

CART: Building Classification Trees

- Chúng ta có cây được chia bước đầu như sau:

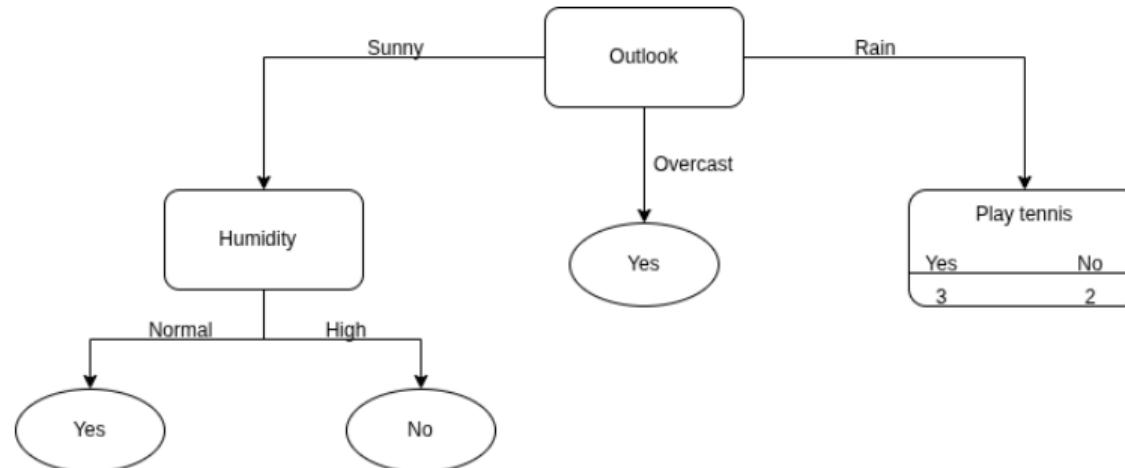


CART: Building Classification Trees

- Khi Outlook = Sunny:

Thuộc tính	Total Gini
Temp	0.2
Humidity	0
Wind	0.584

- Kết quả: Thuộc tính có Total Gini thấp nhất (Humidity) được chọn làm nút chia

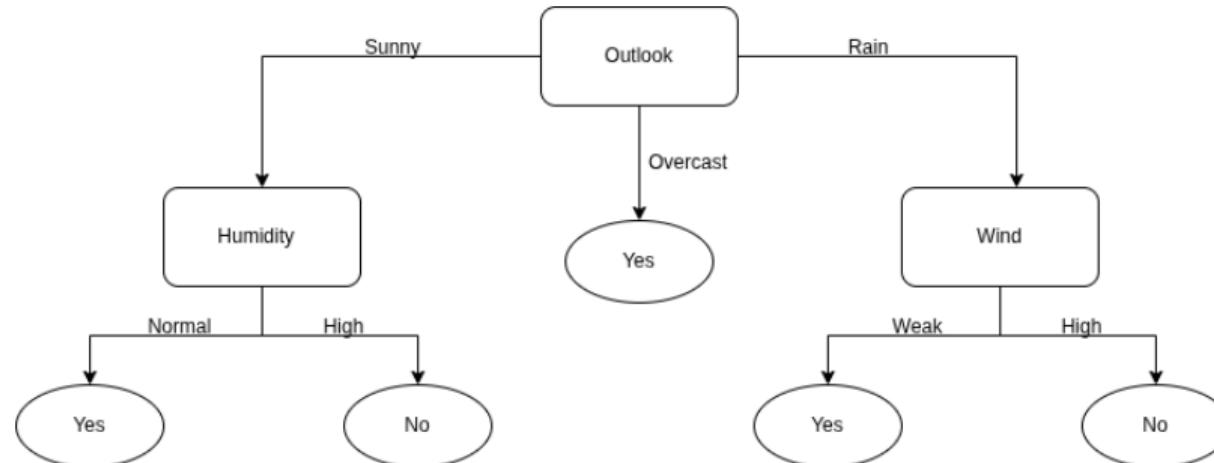


CART: Building Classification Trees

- Khi Outlook = Rain:

Attribute	Total Gini
Temp	0.464
Humidity	0.464
Wind	0

- Kết quả: Thuộc tính có Total Gini thấp nhất (Wind) được chọn làm nút chia



CART: Regression Trees

- Ở mỗi nút, ta chọn thuộc tính A sao cho nó **giảm phương sai (độ lệch chuẩn)** của biến mục tiêu y nhiều nhất.
- Công thức tính Độ giảm độ lệch chuẩn (Standard Deviation Reduction – SDR):

$$SDR(A) = SD_{\text{parent}} - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|} SD(S_v)$$

- **Giải thích ký hiệu:**

- $SDR(A)$: Độ giảm độ lệch chuẩn (mục tiêu là **tối đa hóa** giá trị này).
- SD_{parent} : Độ lệch chuẩn của biến mục tiêu y tại nút cha (trước khi chia).
- A : Thuộc tính (feature) được xét để thực hiện phép chia.
- $|S_v| / |S|$: Số lượng mẫu của tập con tương ứng với giá trị v của thuộc tính A (tức một nút con).
- $SD(S_v)$: Độ lệch chuẩn tại nút con S_v sau khi chia.
- Biểu thức tổng là **trung bình có trọng số** của độ lệch chuẩn của các nút con sau khi chia.

CART: Building Regression Trees

Outlook	Temp	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	25
Sunny	Hot	High	Strong	30
Overcast	Hot	High	Weak	46
Rain	Mild	High	Weak	45
Rain	Cool	Normal	Weak	52
Rain	Cool	Normal	Strong	23
Overcast	Cool	Normal	Strong	43
Sunny	Mild	High	Weak	35
Sunny	Cool	Normal	Weak	38
Rain	Mild	Normal	Weak	46
Sunny	Mild	Normal	Strong	48
Overcast	Mild	High	Strong	52
Overcast	Hot	Normal	Weak	44
Rain	Mild	High	Strong	30

CART: Building Regression Trees

- **Bước 1:** Tính độ lệch chuẩn của biến mục tiêu trong nút cha trước khi chia

$$\text{Average of tennis player} = \frac{25 + 30 + 46 + \dots + 30}{14} = 39.78$$

$$SD_{\text{parent}} = \sqrt{\frac{(25 - 39.78)^2 \cdot (30 - 39.78)^2 \dots (30 - 39.78)^2}{14}} = 9.32$$

- **Bước 2:** Tính độ lệch chuẩn cho từng thuộc tính

CART: Building Regression Trees

- Outlook
 - Sunny

Outlook	Temp	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	25
Sunny	Hot	High	Strong	30
Sunny	Mild	High	Weak	35
Sunny	Cool	Normal	Weak	38
Sunny	Mild	Normal	Strong	48

$$\text{Average of tennis players for sunny outlook} = \frac{25 + 30 + 35 + 38 + 48}{5} = 35.2$$

$$\text{SD of tennis players for sunny outlook} = \sqrt{\frac{(25 - 35.2)^2 + (30 - 35.2)^2 + (35 - 35.2)^2 + (38 - 35.2)^2 + (48 - 35.2)^2}{5}} = 7.78$$

CART: Building Regression Trees

- Outlook

- Overcast

Outlook	Temp	Humidity	Wind	Play Tennis
Overcast	Hot	High	Weak	46
Overcast	Cool	Normal	Strong	43
Overcast	Mild	High	Strong	52
Overcast	Hot	Normal	Weak	44

$$\text{Average of tennis players for overcast outlook} = \frac{46 + 43 + 52 + 44}{4} = 46.25$$

$$\text{SD of tennis players for overcast outlook} = \sqrt{\frac{(46 - 46.25)^2 + (43 - 46.25)^2 + (52 - 46.25)^2 + (44 - 46.25)^2}{4}} = 3.49$$

CART: Building Regression Trees

- Outlook
 - Rain

Outlook	Temp	Humidity	Wind	Play Tennis
Rain	Mild	High	Weak	45
Rain	Cool	Normal	Weak	52
Rain	Cool	Normal	Strong	23
Rain	Mild	Normal	Weak	46
Rain	Mild	High	Strong	30

$$\text{Average of tennis players for rain outlook} = \frac{45 + 52 + 23 + 46 + 30}{5} = 39.2$$

$$\text{SD of tennis players for rain outlook} = \sqrt{\frac{(45 - 39.2)^2 + (52 - 39.2)^2 + (23 - 39.2)^2 + (46 - 39.2)^2 + (30 - 39.2)^2}{5}} = 10.87$$

CART: Building Regression Trees

Outlook	SD	Instances
Sunny	7.78	5
Overcast	3.49	4
Rain	10.87	5

$$\sum_{v \in \text{values}(Outlook)} \frac{|S_v|}{|S|} SD(S_v) = \frac{5}{14} * 7.78 + \frac{4}{14} * 3.49 + \frac{5}{14} * 10.87 = 7.66$$

$$SDR(Outlook) = SD_{\text{parent}} - \sum_{v \in \text{values}(Outlook)} \frac{|S_v|}{|S|} SD(S_v) = 9.32 - 7.66 = 1.66$$

CART: Building Regression Trees

- Tương tự, ta tính cho các thuộc tính còn lại
 - Temp

Temp	SD	Instances
Hot	8.95	4
Mild	10.51	4
Cool	7.65	6

$$\sum_{v \in \text{values}(Temp)} \frac{|S_v|}{|S|} SD(S_v) = \frac{4}{14} * 8.95 + \frac{4}{14} * 10.51 + \frac{6}{14} * 7.65 = 8.84$$

$$SDR(Temp) = SD_{\text{parent}} - \sum_{v \in \text{values}(Temp)} \frac{|S_v|}{|S|} SD(S_v) = 9.32 - 8.84 = 0.48$$

CART: Building Regression Trees

- Humidity

Humidity	SD	Instances
High	9.36	7
Normal	8.73	7

$$\sum_{v \in \text{values}(\text{Humidity})} \frac{|S_v|}{|S|} \text{SD}(S_v) = \frac{7}{14} * 9.36 + \frac{7}{14} * 8.73 = 9.04$$

$$\text{SDR}(\text{Humidity}) = \text{SD}_{\text{parent}} - \sum_{v \in \text{values}(\text{Humidity})} \frac{|S_v|}{|S|} \text{SD}(S_v) = 9.32 - 9.04 = 0.28$$

CART: Building Regression Trees

- Wind

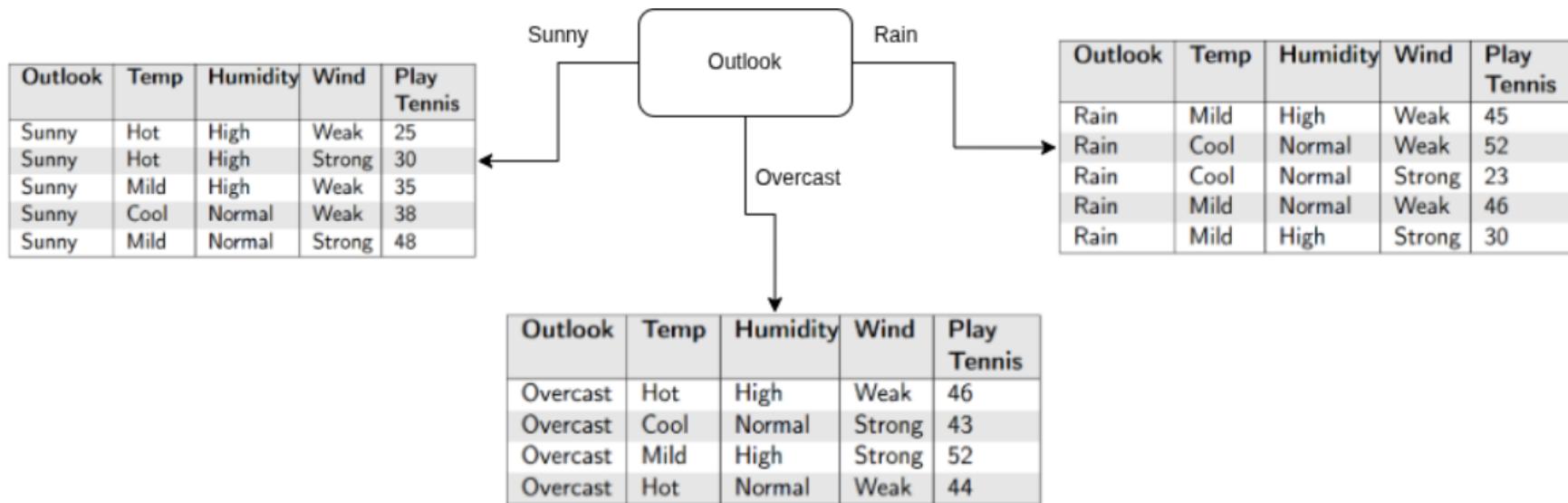
Wind	SD	Instances
Strong	10.59	6
Weak	7.87	8

$$\sum_{v \in \text{values}(Wind)} \frac{|S_v|}{|S|} SD(S_v) = \frac{6}{14} * 10.59 + \frac{8}{14} * 7.87 = 9.03$$

$$SDR(Wind) = SD_{\text{parent}} - \sum_{v \in \text{values}(Wind)} \frac{|S_v|}{|S|} SD(S_v) = 9.32 - 9.03 = 0.29$$

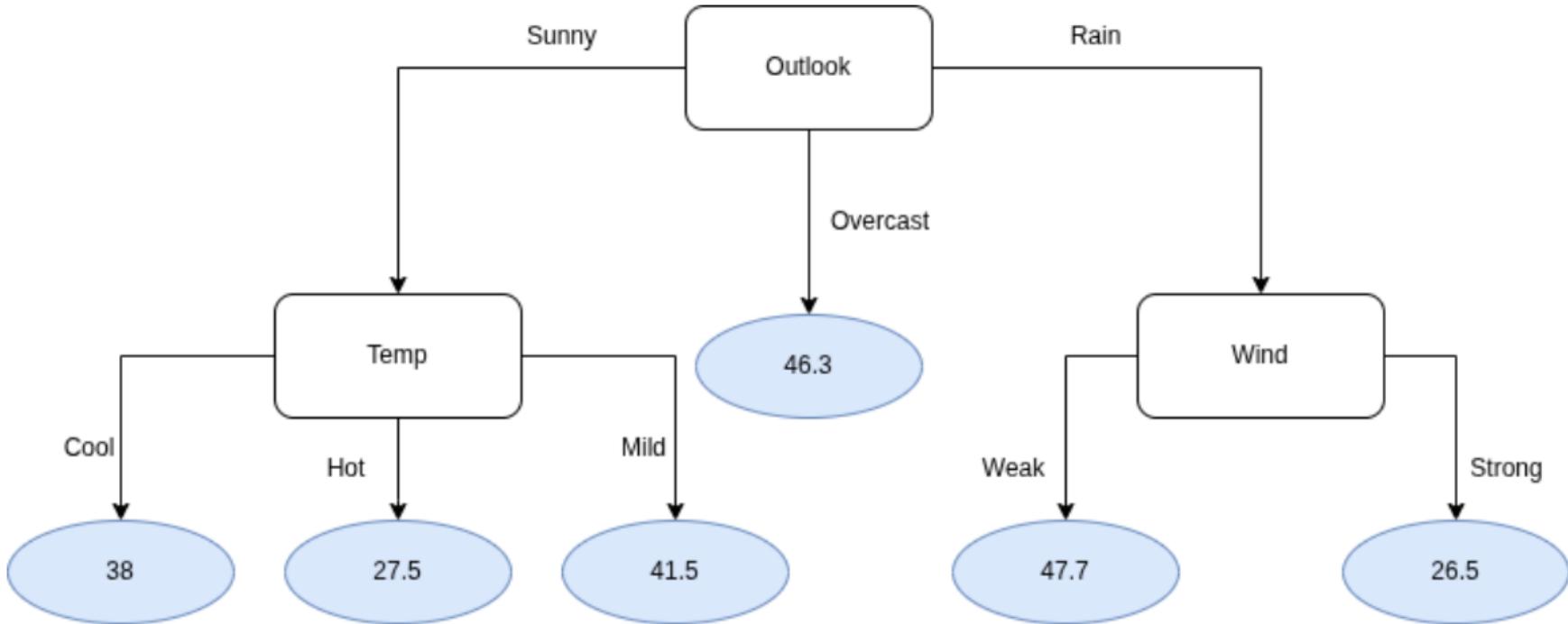
CART: Building Regression Trees

- Bước 3: Ta chọn thuộc tính có SDR cao nhất (Outlook)



CART: Building Regression Trees

- **Bước 4:** Ta làm tương tự cho các nút con và cuối cùng ta được cây hồi quy như sau:



CHAID: Giới thiệu

- Được đề xuất vào năm 1980 bởi Gordon V. Kass
- **CHAID (Chi-square Automatic Interaction Detection)**: thuật toán tự động phát hiện tương tác giữa các biến dựa trên kiểm định Chi-square.
- Xây dựng cây quyết định cho các bài toán phân loại
- **Chi-Square**
 - Được đề xuất bởi Karl Pearson (người phát minh ra hệ số tương quan Pearson)
 - CHAID dùng Chi-square để đo độ khác biệt giữa phân phối quan sát và phân phối kỳ vọng

$$\text{Chi-Square Test} = \sqrt{\frac{(y - y')^2}{y'}} \quad \text{trong đó } y \text{ là giá trị thực và } y' \text{ là giá trị kỳ vọng}$$

CHAID: Building Classification Trees

Outlook	Temp	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

CHAID: Building Classification Trees

- **Outlook Feature**
- Outlook \Rightarrow *sunny, rain, overcast*

	Yes (O)	No (O)	Expected	Chi-Square (Yes)	Chi-Square (No)
Sunny	2	3	2.5	0.316	0.316
Overcast	4	0	2	1.414	1.414
Rain	3	2	2.5	0.316	0.316

- $\text{Chi-Square(Outlook)} = 0.316 + 1.414 + 0.316 + 0.316 + 1.414 + 0.316 = 4.092$

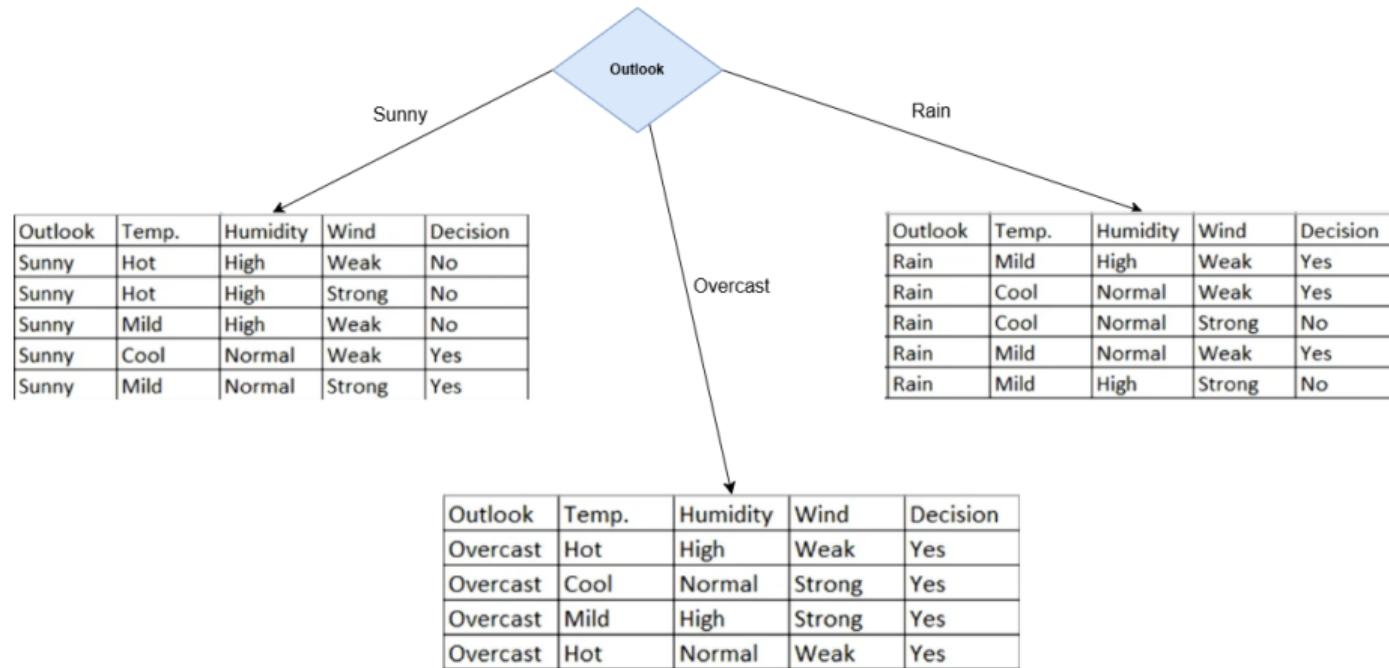
CHAID: Building Classification Trees

- Tương tự, ta tính giá trị Chi-Square cho các thuộc tính còn lại

Thuộc tính	Chi-Square
Outlook	4.092
Temp	2.569
Humidity	3.207
Wind	1.604

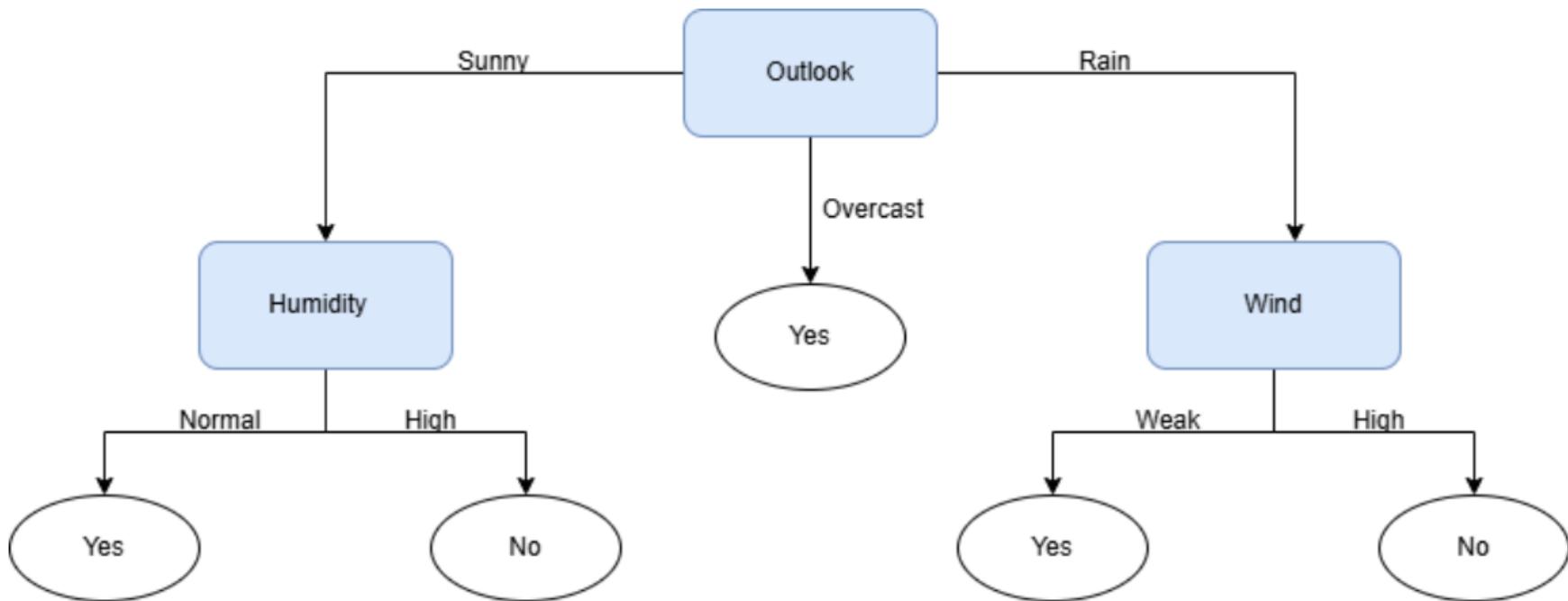
CHAID: Building Classification Trees

- Ta chọn thuộc tính có giá trị Chi-Square cao nhất \Rightarrow Outlook



CHAID: Building Classification Trees

- Áp dụng các bước này cho các nút con và ta có cây phân loại như sau:



Vấn đề

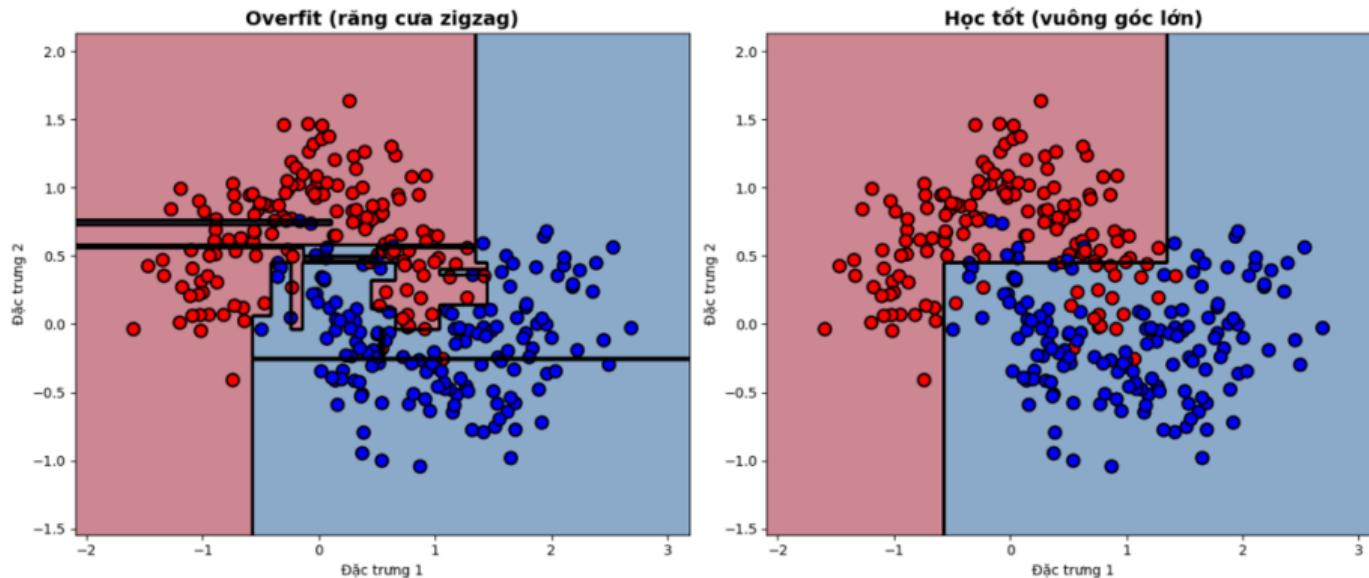
- Decision Tree rất linh hoạt và có thể dễ dàng ghi nhớ toàn bộ dữ liệu huấn luyện. Nếu không có các ràng buộc, chúng sẽ bị quá khớp (overfitting), dẫn đến hiệu quả kém khi hoạt động trên dữ liệu mới.

Tại sao chúng ta cần hiệu chỉnh (Regularization)

- **Ngăn chặn việc ghi nhớ dữ liệu huấn luyện:** Ngăn không cho các nút lá cô lập từng mẫu dữ liệu đơn lẻ
- **Kiểm soát độ phức tạp của cây:** Tránh việc phân chia quá nhiều hoặc cây phát triển quá sâu so với kích thước của tập dữ liệu.
- **Bỏ qua các đặc trưng yếu hoặc không liên quan:** Ngăn chặn các phép chia chỉ khớp với nhiễu (noise).

Regularization

- **Làm trơn biên quyết định:** Giảm thiểu độ nhạy của mô hình đối với các biến động nhỏ ở đầu vào.



- **Giảm thiểu phương sai và cải thiện tính ổn định:** Giúp cây ít nhạy cảm hơn trước những thay đổi nhỏ trong dữ liệu huấn luyện.

Phương pháp: Cắt tỉa (Pruning)

Mục đích: Giảm độ phức tạp, tránh việc quá khớp (overfitting) và tăng tính tổng quát hóa trên toàn bộ cây

Các loại cắt tỉa:

- **Pre-Pruning:** Dừng chia cây sớm hơn, nhưng cây có thể bị chưa khớp (underfitting)
- **Post-Pruning:** Hoàn thành việc xây dựng cây, sau đó loại bỏ các nhánh không hữu ích

Pre-Pruning:

- **Kích thước lá tối thiểu (Min Leaf Size):** Số lượng mẫu tối thiểu yêu cầu tại mỗi nút lá. Ngăn chặn hiện tượng quá khớp (overfitting) xuất phát từ các nút lá chứa quá ít dữ liệu.
- **Độ sâu tối đa (Max Depth):** Độ sâu lớn nhất cho phép của cây. Tham số này kiểm soát độ phức tạp của mô hình; cây quá sâu dễ dẫn đến quá khớp, trong khi cây quá nông dễ dẫn đến chưa khớp (underfitting).
- **Số nút tối đa (Max Nodes):** Số lượng nút lá tối đa cho phép. Giới hạn mức độ chi tiết của cây và giảm thiểu nguy cơ quá khớp.
- **Độ giảm hàm mất mát tối thiểu (Min Loss Decrease):** Mức giảm độ vẫn đục (impurity) tối thiểu cần thiết để thực hiện một phép chia. Giá trị càng lớn sẽ dẫn đến cấu trúc cây càng đơn giản.

Post-Pruning:

- Reduced Error Pruning (REP)
- Cost-Complexity Pruning (CCP)
- Critical Value Pruning (CVP)
- Minimum Error Pruning (MEP)

Reduced Error Pruning (REP)

Định nghĩa: Reduced Error Pruning (REP) là một kỹ thuật post-pruning (được đề xuất bởi Quinlan, 1987), sử dụng một tập dữ liệu xác thực (validation set) độc lập để quyết định xem một nhánh cây con có nên được loại bỏ hay không.

Ý tưởng chính:

- Chỉ thực hiện cắt tỉa một nhánh cây con nếu thao tác này không làm suy giảm độ chính xác trên tập xác thực.
- Mục tiêu: đơn giản hóa cấu trúc cây đồng thời bảo toàn khả năng tổng quát hóa (generalization).

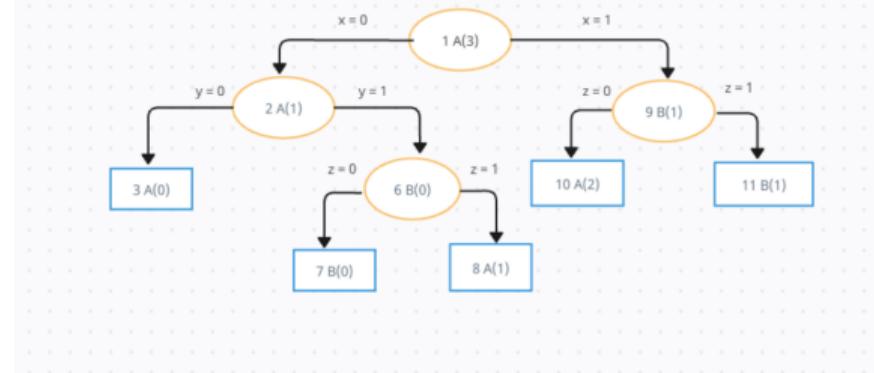
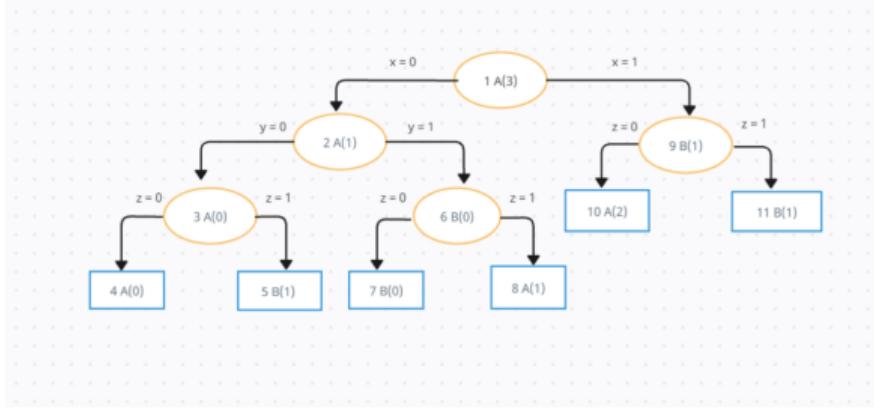
Reduced Error Pruning (REP)

Quy trình thực hiện:

- ① **Huấn luyện cây hoàn chỉnh:** Xây dựng cây quyết định với kích thước lớn nhất có thể.
- ② **Đánh giá sai số xác thực** tại từng nút nội bộ (internal node):
 - Sai số nếu giữ nguyên nhánh cây con.
 - Sai số nếu thay thế nhánh cây con bằng một nút lá (gán nhãn theo lớp đa số - majority class).
- ③ **Cắt tỉa có điều kiện:** Thực hiện cắt tỉa nếu độ chính xác không bị sụt giảm.
- ④ **Lặp lại từ dưới lên (Bottom-up):** Tiếp tục quy trình cho đến khi không còn khả năng cắt tỉa an toàn nào nữa.
- ⑤ **Lựa chọn cây tối ưu:** Chọn mô hình có độ chính xác xác thực cao nhất.

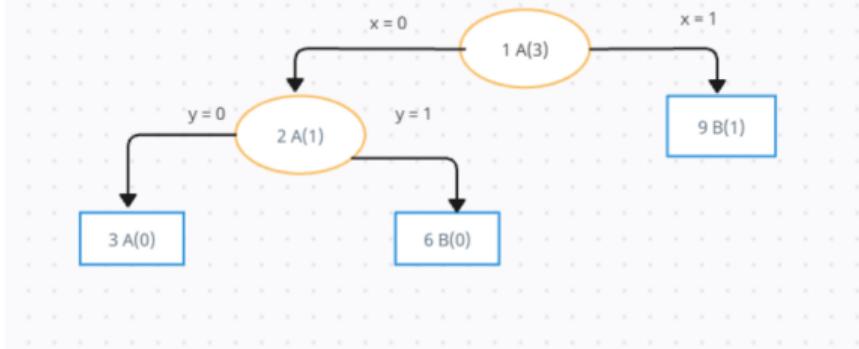
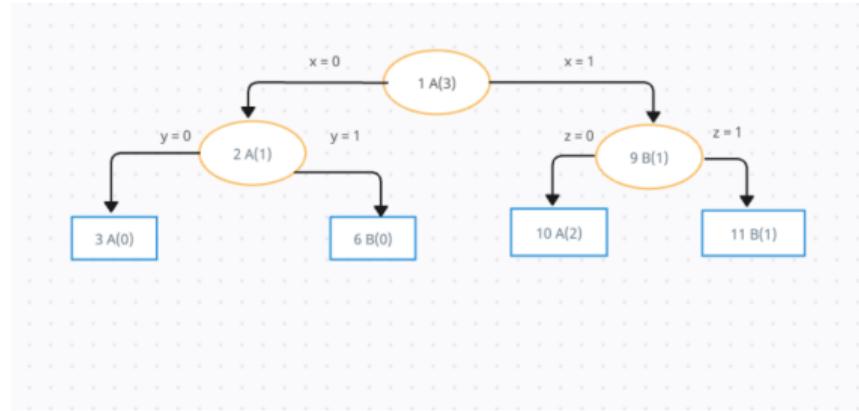
Reduced Error Pruning (REP)

X	Y	Z	Class
0	0	1	A
0	1	1	B
0	0	0	A
0	1	0	B
1	0	0	B
1	1	0	B
1	1	1	A



Reduced Error Pruning (REP)

X	Y	Z	Class
0	0	1	A
0	1	1	B
0	0	0	A
0	1	0	B
1	0	0	B
1	1	0	B
1	1	1	A



Reduced Error Pruning (REP)

Điểm mạnh:

- Đơn giản và dễ thực hiện.
- Giúp tránh việc quá khớp (overfitting).
- Sinh ra cây đỗ phức tạp và dễ giải thích hơn.

Hạn chế:

- Cần một lượng lớn tập dữ liệu xác thực (validation set).
- Tập dữ liệu xác thực nhỏ có thể dẫn đến việc cắt tỉa quá mức (over-pruning).
- Tham lam và cục bộ: cắt tỉa sao cho giảm được lỗi tối đa tại mỗi nút.

Kết quả nhận được sau khi cắt tỉa:

- Một cây đơn giản, sạch hơn mà vẫn giữ (hoặc cải thiện) được độ chính xác so với cây trước khi cắt tỉa.

Minimize Error Pruning (MEP)

Định nghĩa:

- MEP là kỹ thuật **post-pruning** (Niblett & Bratko, 1986), sử dụng **ước lượng lỗi dự kiến** tại từng nút dựa trên phân phối lớp và Dirichlet smoothing, thay vì dựa vào validation set.
- Quyết định xem một nhánh cây con có nên được cắt bỏ hay không dựa trên E_{prune} và $E_{\text{no_prune}}$.

Ý tưởng chính:

- Duyệt **bottom-up từ lá lên**, so sánh lỗi dự kiến nếu prune nút và nếu **giữ nguyên cây con**.
- Chỉ cắt tía nếu prune **không làm tăng lỗi dự kiến**.

Mục tiêu:

- Đơn giản hóa cây, giảm overfitting.
- Bảo toàn khả năng tổng quát hóa (generalization) trên dữ liệu mới.

Minimize Error Pruning (MEP) – Công thức & Quy trình

Công thức:

- Lỗi nếu prune (biến nút thành lá đa số):

$$E_{\text{prune}} = \frac{n - n_c + (k - 1)}{n + k}$$

- n = tổng số mẫu, n_c = số mẫu lớp chiếm đa số, k = số lớp

- Lỗi nếu giữ nguyên cây con:

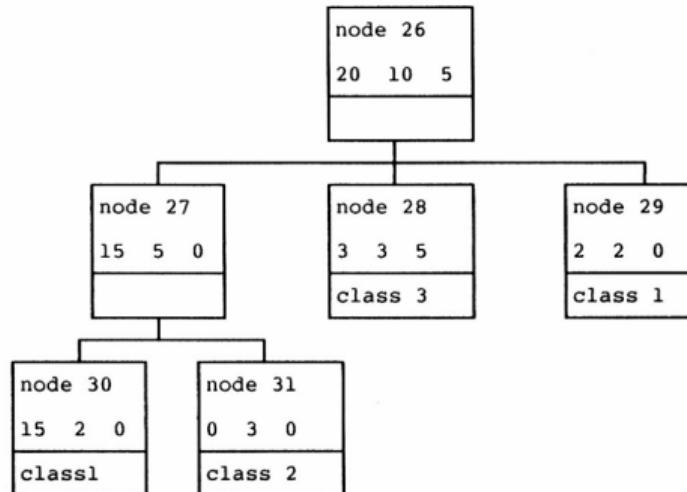
$$E_{\text{no_prune}} = \sum_{\text{child}} \frac{n_{\text{child}}}{n} E(\text{child})$$

- n_{child} = số mẫu nhánh con - $E(\text{child})$ = lỗi Dirichlet của nhánh con

Quy trình tóm tắt:

- ❶ Huấn luyện cây đầy đủ (full tree).
- ❷ Duyệt bottom-up từng nút nội bộ t :
 - Tính E_{prune} và $E_{\text{no_prune}}$
 - Nếu $E_{\text{prune}} \leq E_{\text{no_prune}} \Rightarrow$ prune nút t
- ❸ Lặp lại đến khi không còn nút thỏa điều kiện.
- ❹ Kết quả: cây prune với lỗi dự kiến tối thiểu.

Minimize Error Pruning (MEP)



Cây con bị tẩy một phần.

Ví dụ: Tại nút 27 của cây, tính lỗi dự kiến:

- Nếu prune (biến thành lá đa số):

$$E_k = \frac{20 - 15 + 3 - 1}{20 + 3} = 0.304$$

- Nếu không prune (giữ nguyên cây con):

$$E_k = \frac{17}{20} \left(\frac{17 - 15 + 3 - 1}{17 + 3} \right) + \frac{3}{20} \left(\frac{3 - 3 + 3 - 1}{3 + 3} \right) = 0.220$$

Nhận xét: $E_{\text{prune}} = 0.304 > E_{\text{no prune}} = 0.220 \Rightarrow \text{không prune nút này.}$

Minimize Error Pruning (MEP)

Ưu điểm:

- Không cần validation set
- Có cơ sở thống kê, phù hợp dữ liệu ít
- Dirichlet smoothing giúp ổn định ước lượng lỗi

Nhược điểm:

- Quyết định cục bộ, không đảm bảo global-optimal
- Nhạy với số lớp k và phân bố mẫu

So sánh REP vs MEP

Giống nhau:

- Duyệt cây từ lá lên gốc (bottom-up)
- Tại mỗi nút, so sánh giữ nhánh vs cắt nhánh
- Mục tiêu: giảm overfitting, tạo cây gọn hơn

Khác nhau:

Tiêu chí	REP	MEP
Cắt nhánh khi	Accuracy trên validation không giảm	$E_{\text{prune}} \leq E_{\text{no_prune}}$
Dữ liệu cần	Validation đủ lớn	Training set, dựa ước lượng
Độ ổn định	Trực tiếp, gần thực tế; validation nhỏ → prune nhầm	Phụ thuộc ước lượng; số mẫu nhỏ → cây quá lớn/nhỏ
Chi phí	Chạy validation nhiều lần	Tính toán dựa trên số mẫu, nhanh hơn

Cost-Complexity Pruning (CCP)

Mục đích: Giảm overfitting, cân bằng **sai số** và **độ phức tạp**.

Hệ số α cho nút t :

$$\alpha(t) = \frac{R(t) - R(T)}{|leaves(T)| - 1}$$

- t : Nút nội bộ đang xét cắt tỉa.
- $R(T)$: Sai số của cây con hiện tại trước khi cắt tỉa.
- $R(t)$: sai số của cây con nếu prune tại nút t .
- $leaves(T)$: Số lá trong cây con T trước khi cắt tỉa.
- $\alpha(t)$: Mức tăng sai số trung bình trên mỗi lá bị loại khi cắt tỉa nút t . - α nhỏ → nút t là ứng viên tốt để cắt tỉa.

Quy trình thực hiện:

- ❶ Tính $\alpha(t)$ cho mọi nút.
- ❷ Chọn nút có α nhỏ nhất → Cắt tỉa.
- ❸ Lặp lại → Tạo chuỗi cây con.
- ❹ Chọn cây tối ưu bằng **cross-validation**.

Cost-Complexity Pruning (CCP)

Hàm Score cho Pruning cây quyết định

$$\text{Score} = R(T) + \lambda \cdot \text{Complexity}$$

- $R(T)$: sai số dự đoán của cây
- Complexity: độ phức tạp (số lá, số node, chiều cao...)
- λ : hệ số phạt, cân bằng giữa **chính xác** và **đơn giản**

Ý nghĩa:

- λ nhỏ \rightarrow cây sâu, nhiều lá \rightarrow accuracy cao nhưng dễ overfit
- λ lớn \rightarrow cây nông, ít lá \rightarrow tổng quát tốt, đơn giản hơn

Sum of Squares Residuals (SSR)

$$SSR = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2$$

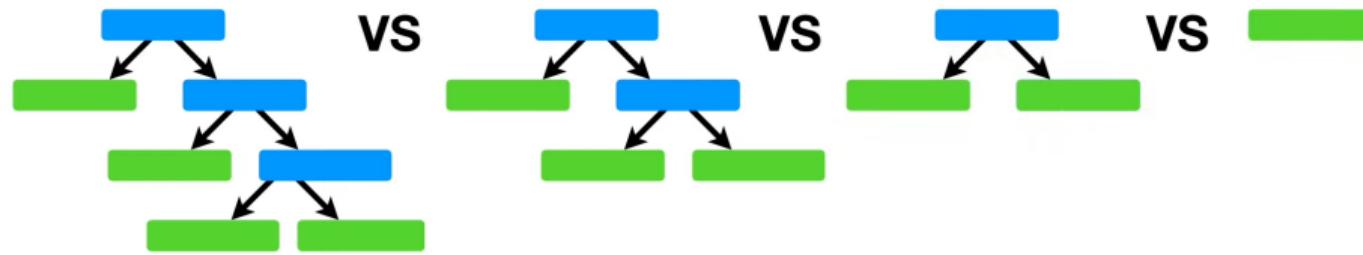
- \hat{y}_i : giá trị dự đoán từ mô hình
- \bar{y} : giá trị trung bình của y (mean of observed values)
- n : số quan sát

Ý nghĩa: SSR đo lường phần biến thiên của y được giải thích bởi mô hình.

Cost-Complexity Pruning (CCP)

So the question is:

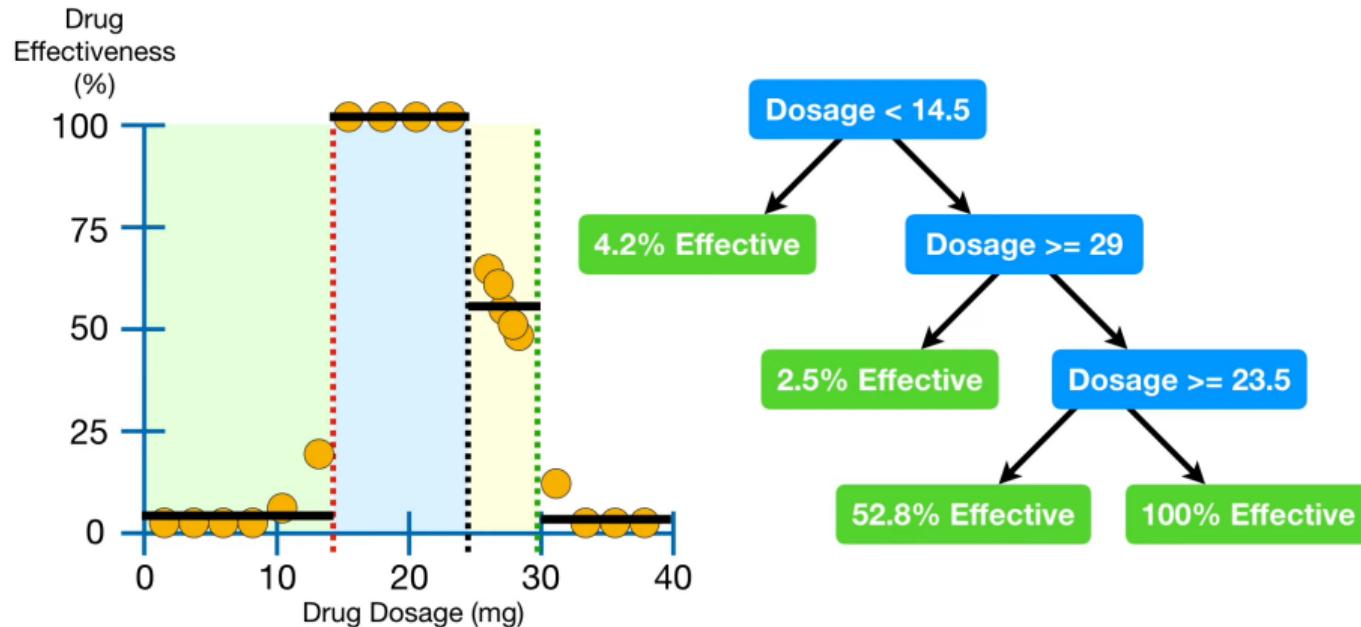
“How do we decide which tree to use?”



Here is the original, full sized tree.

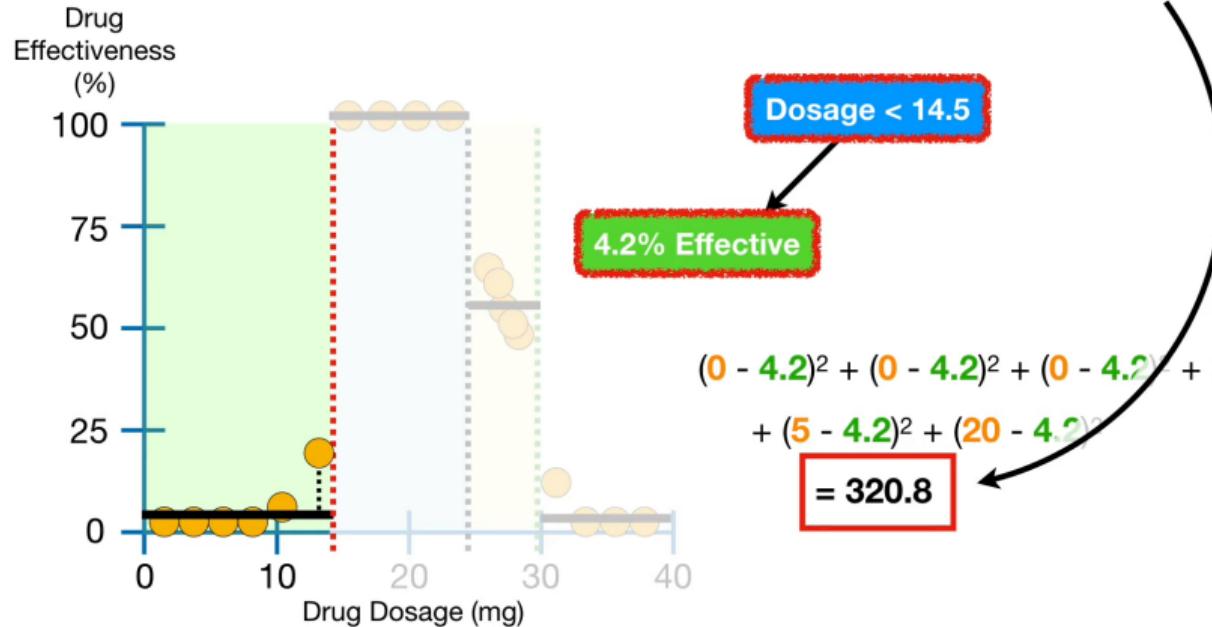
Cost-Complexity Pruning (CCP)

Here is the original, full sized tree.

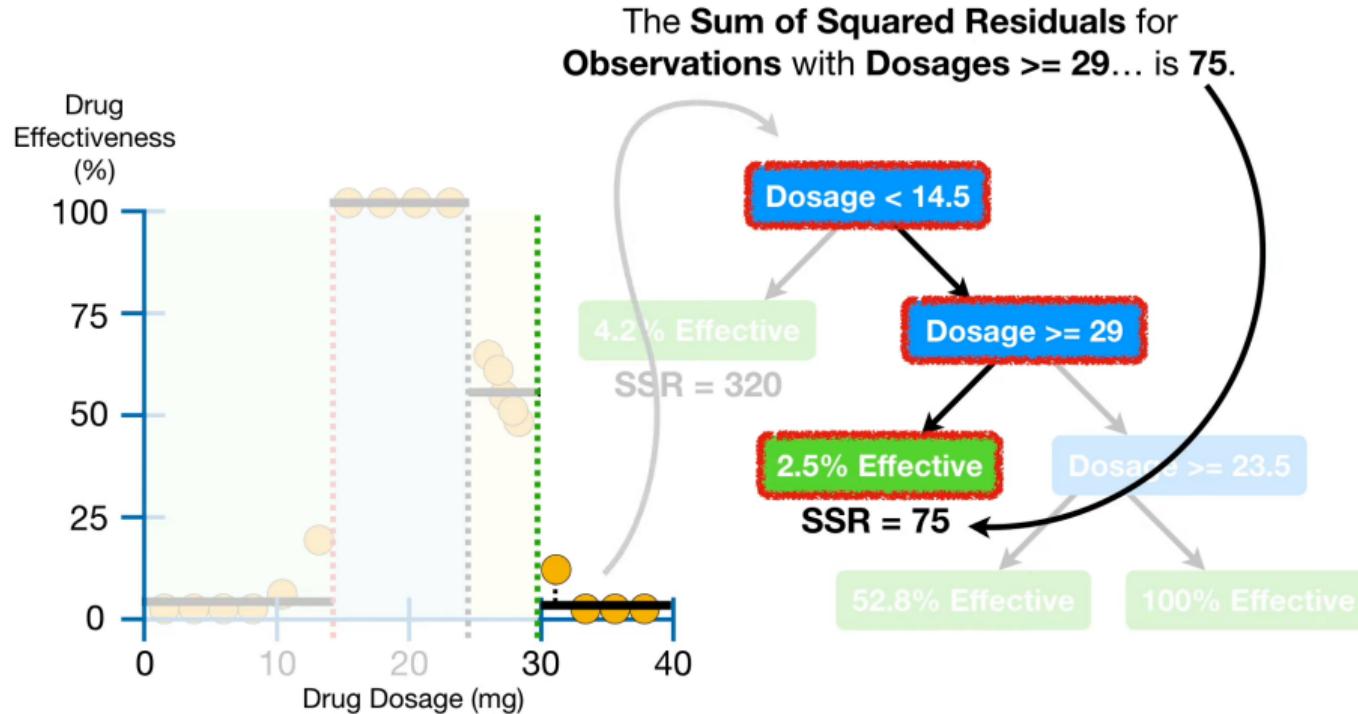


Cost-Complexity Pruning (CCP)

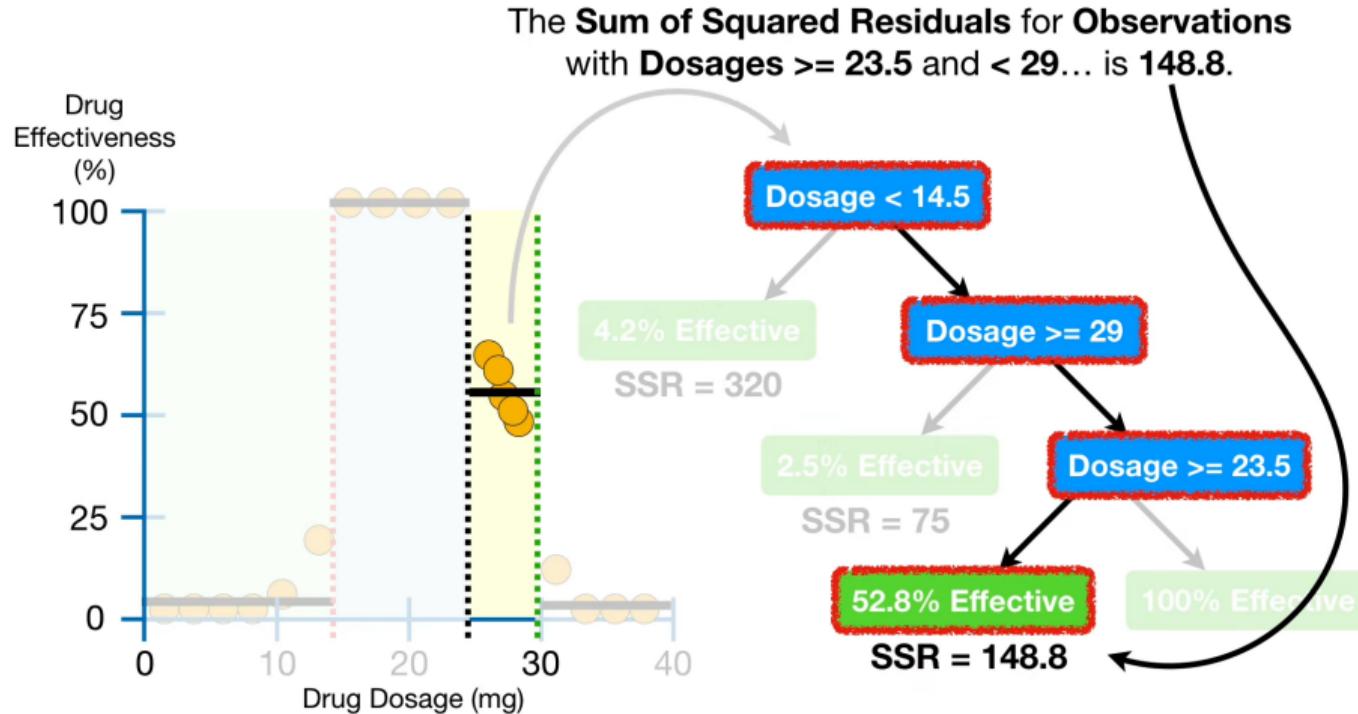
The **Sum of Squared Residuals** for the Observations with Dosages < 14.5 is...



Cost-Complexity Pruning (CCP)

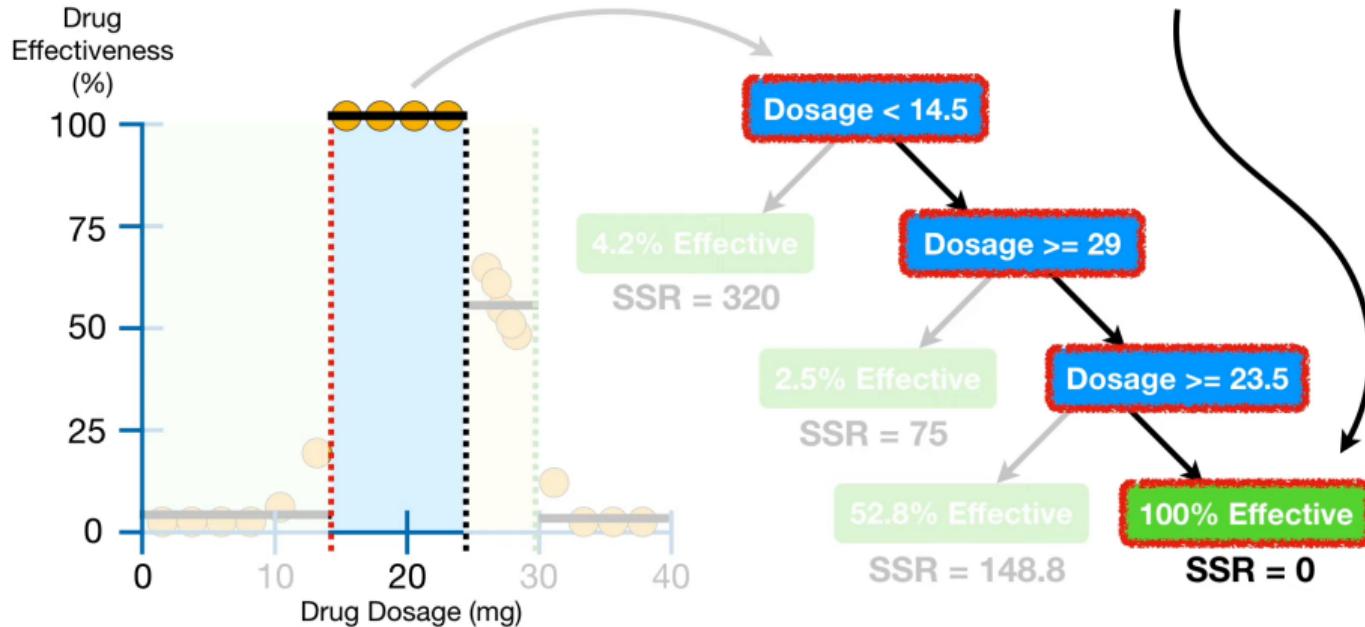


Cost-Complexity Pruning (CCP)



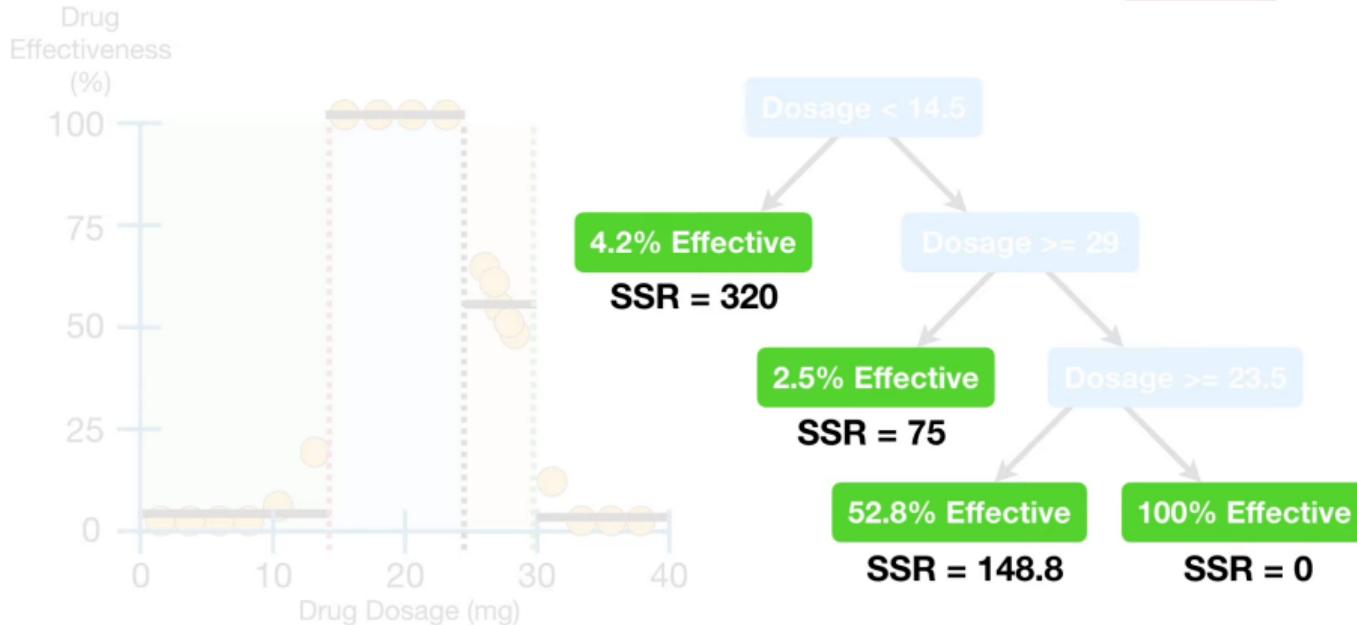
Cost-Complexity Pruning (CCP)

And the **Sum of Squared Residuals** for **Observations** with **Dosages ≥ 14.5 and $< 23.5\dots$** is 0.



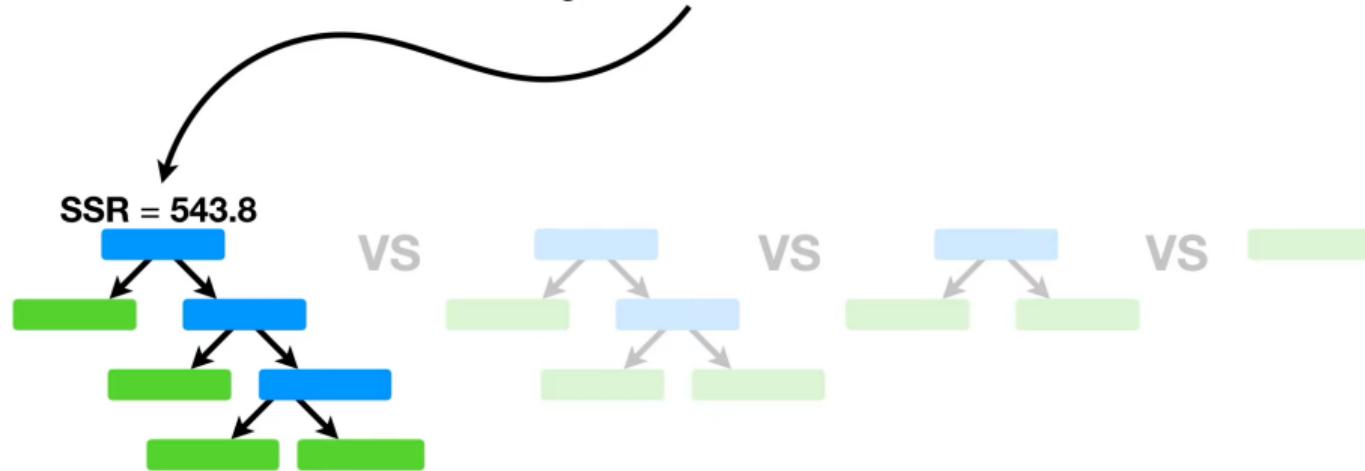
Cost-Complexity Pruning (CCP)

Thus, the total **Sum of Squared Residuals (SSR)** for the whole tree is $320 + 75 + 148.8 + 0 = \boxed{543.8}$



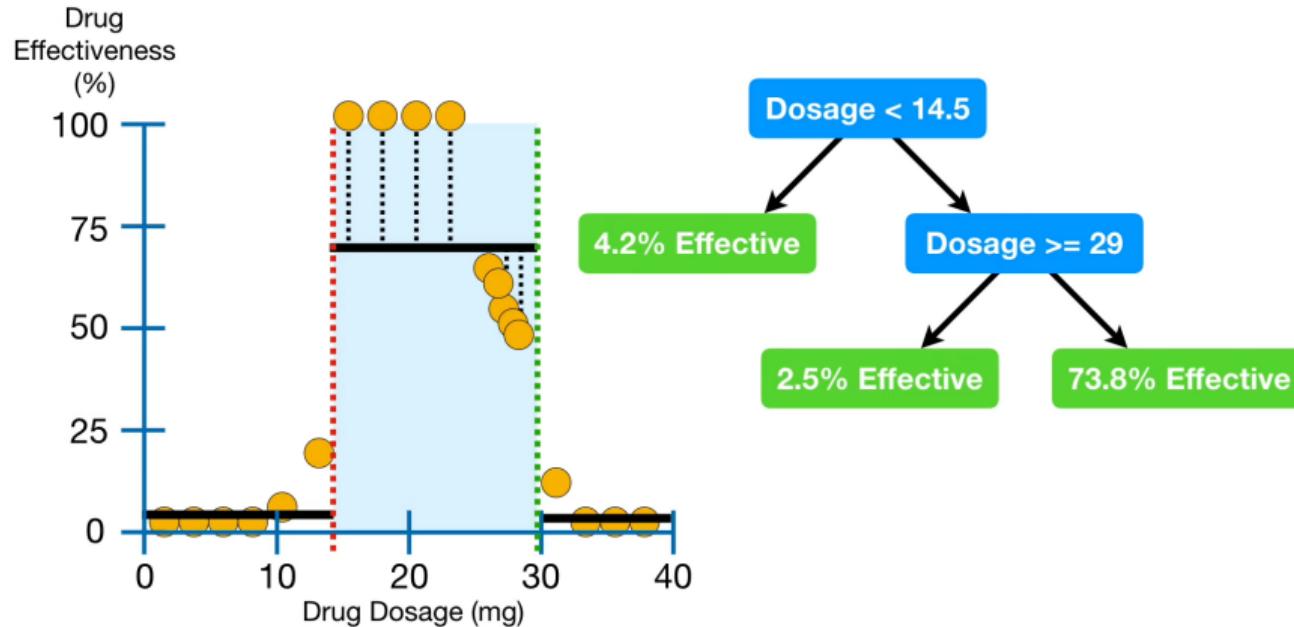
Cost-Complexity Pruning (CCP)

So let's put $\text{SSR} = 543.8$ on top of the original, full sized tree.

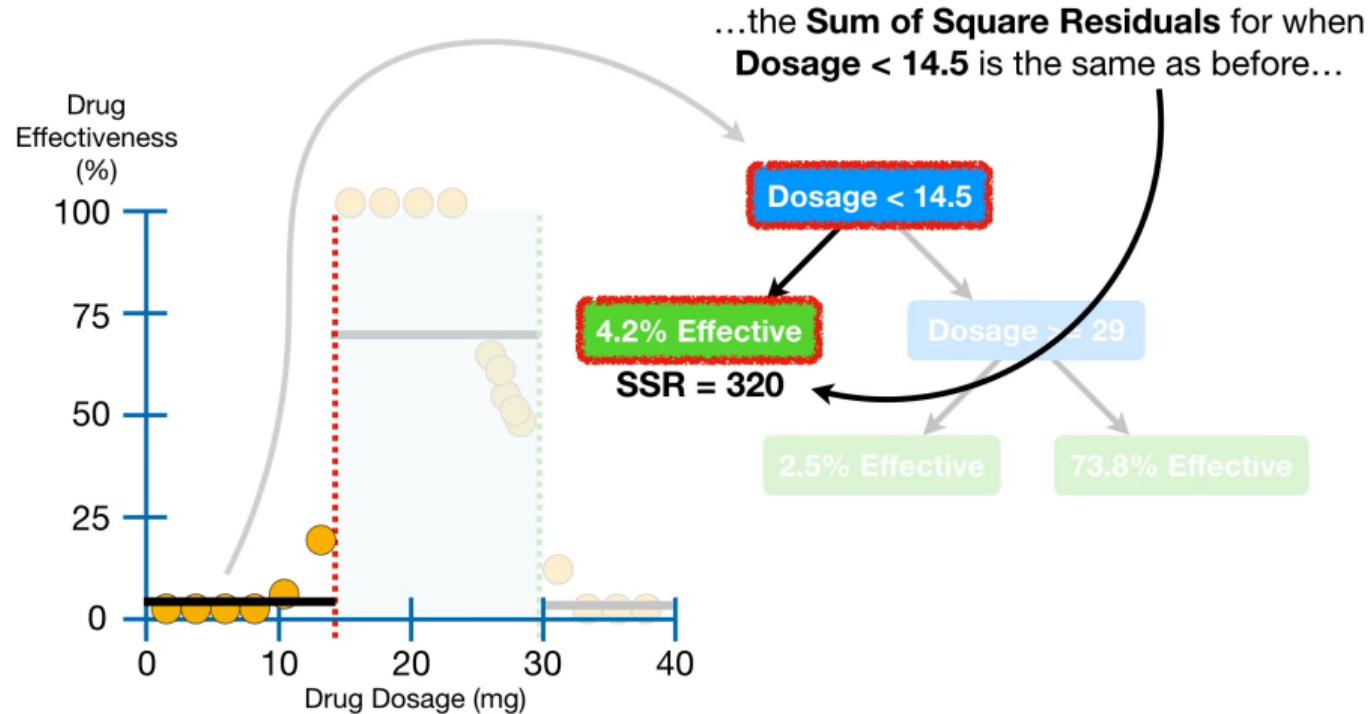


Cost-Complexity Pruning (CCP)

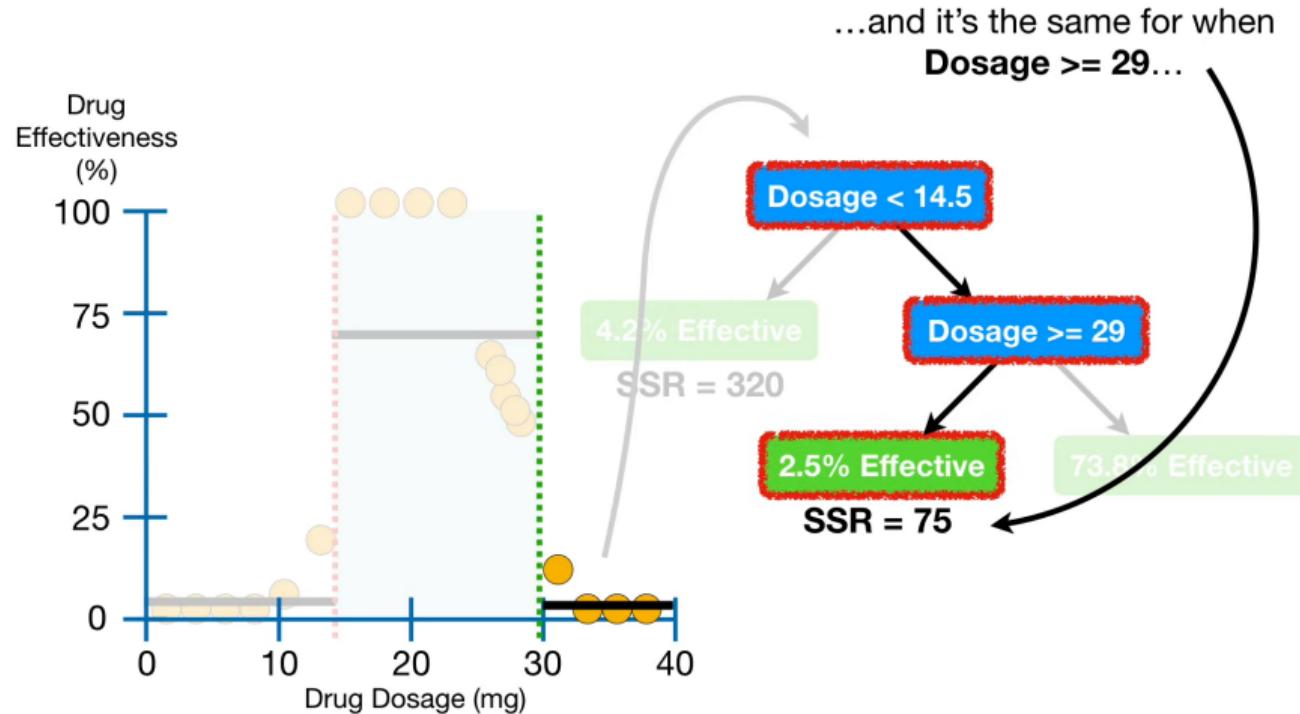
Going back to the data...



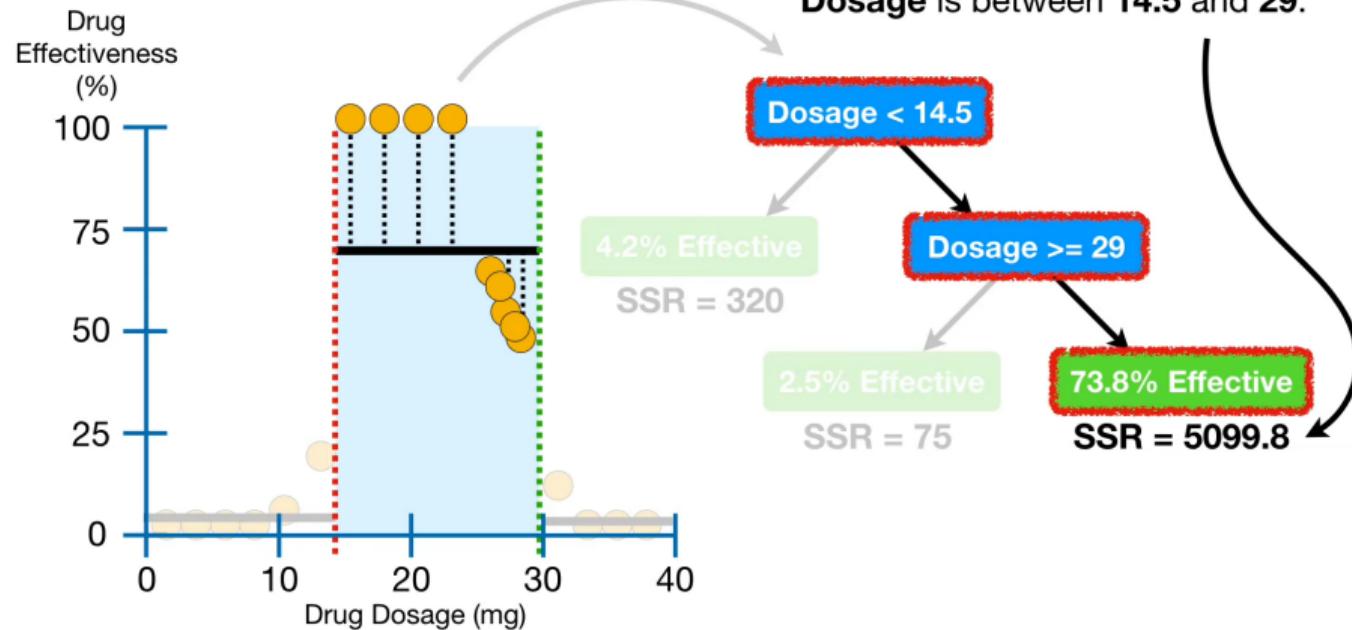
Cost-Complexity Pruning (CCP)



Cost-Complexity Pruning (CCP)

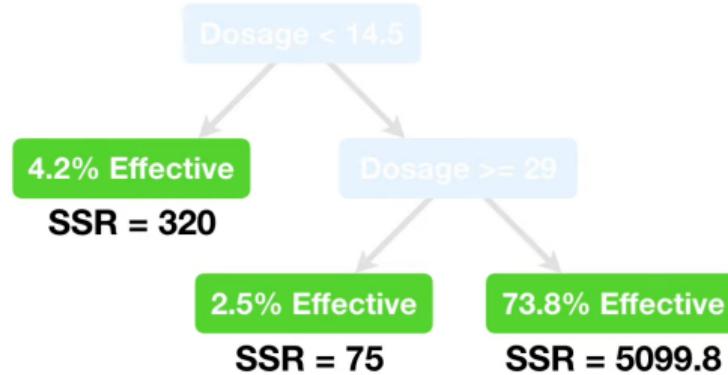
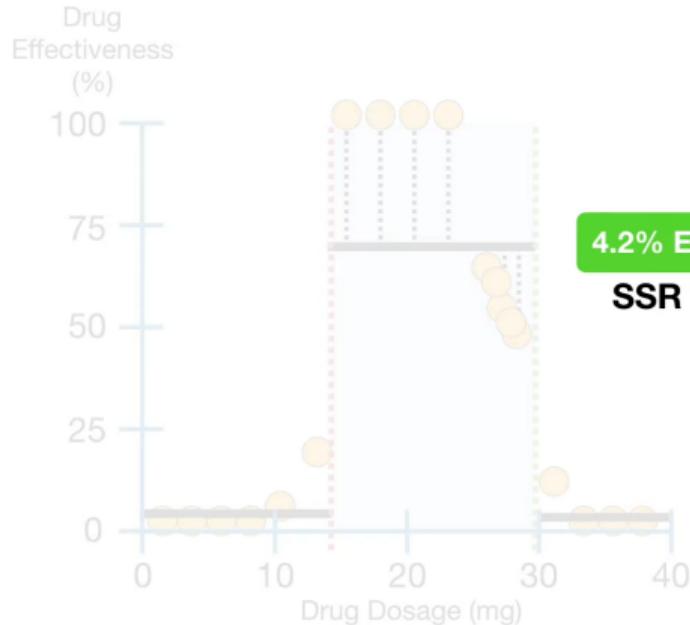


Cost-Complexity Pruning (CCP)



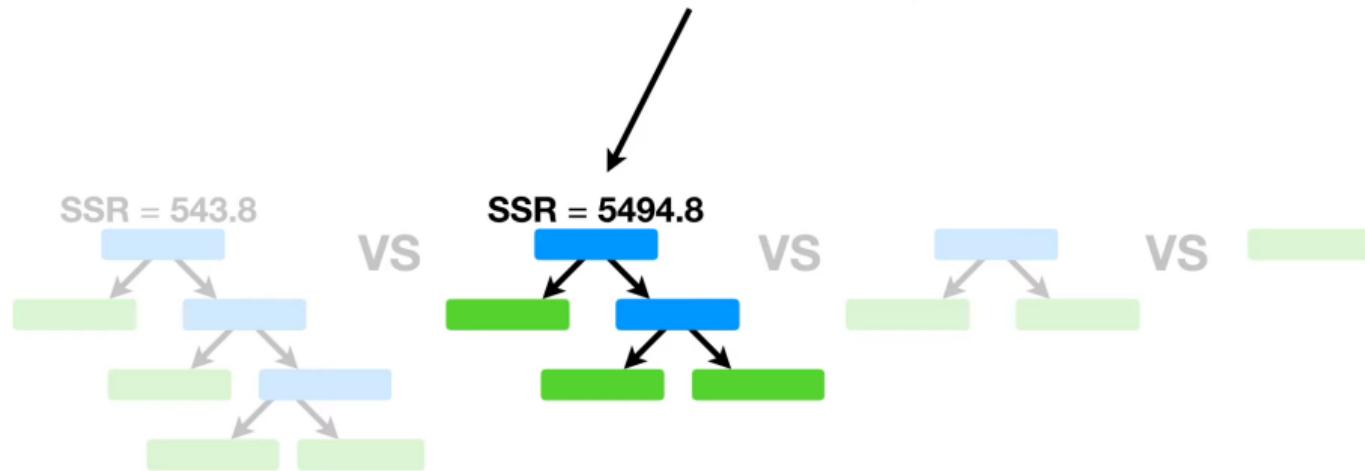
Cost-Complexity Pruning (CCP)

Thus, the total **Sum of Squared Residuals** for
this tree is $320 + 75 + 5099.8 = \boxed{5494.8}$



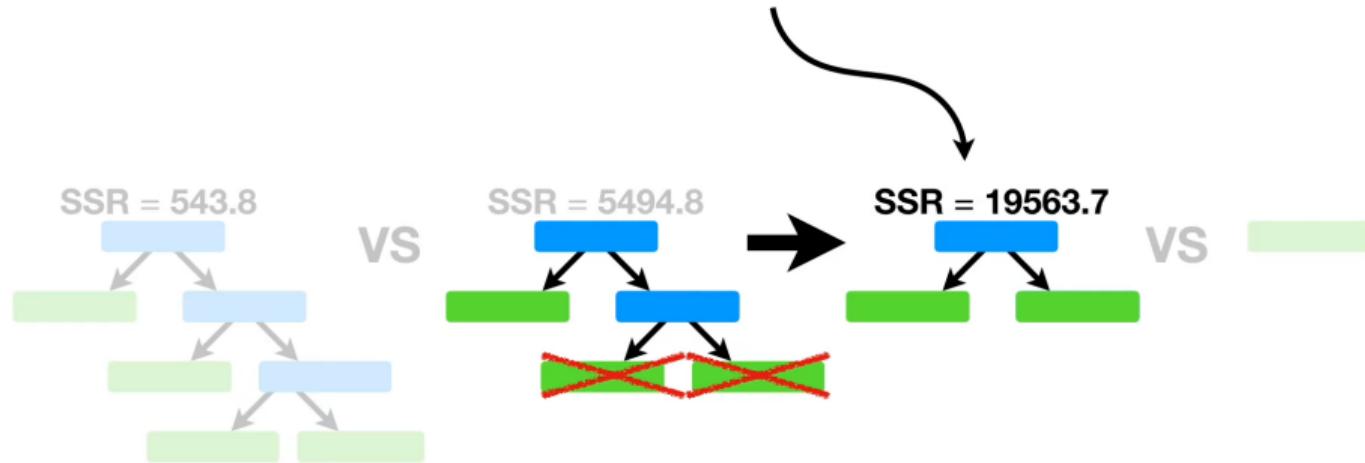
Cost-Complexity Pruning (CCP)

So let's put **SSR = 5494.8** on top
of the sub-tree with **3** leaves.

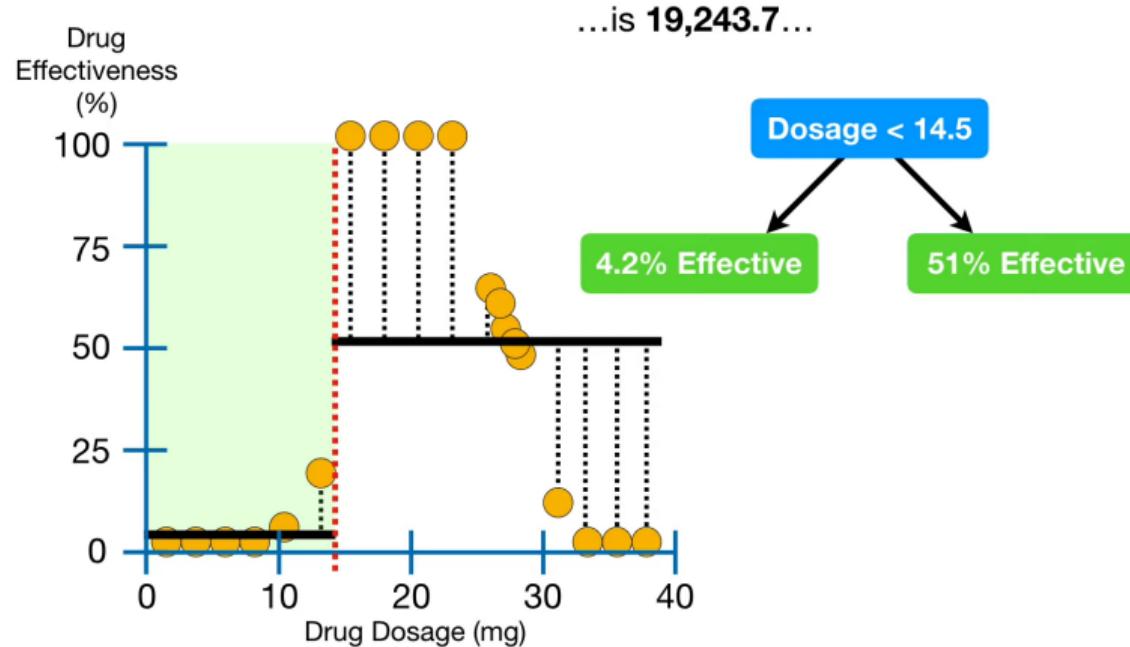


Cost-Complexity Pruning (CCP)

Similarly, the **Sum of Squared Residuals** for
the sub-tree with **2 leaves**...

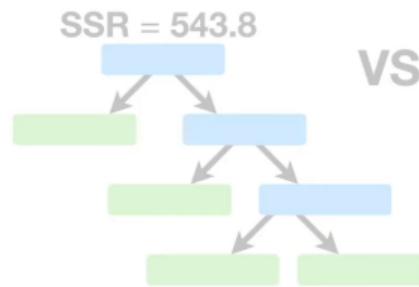


Cost-Complexity Pruning (CCP)

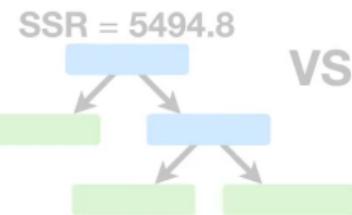


Cost-Complexity Pruning (CCP)

...so we put **SSR = 19243.7** on top of the sub-tree with **2 leaves**.



VS



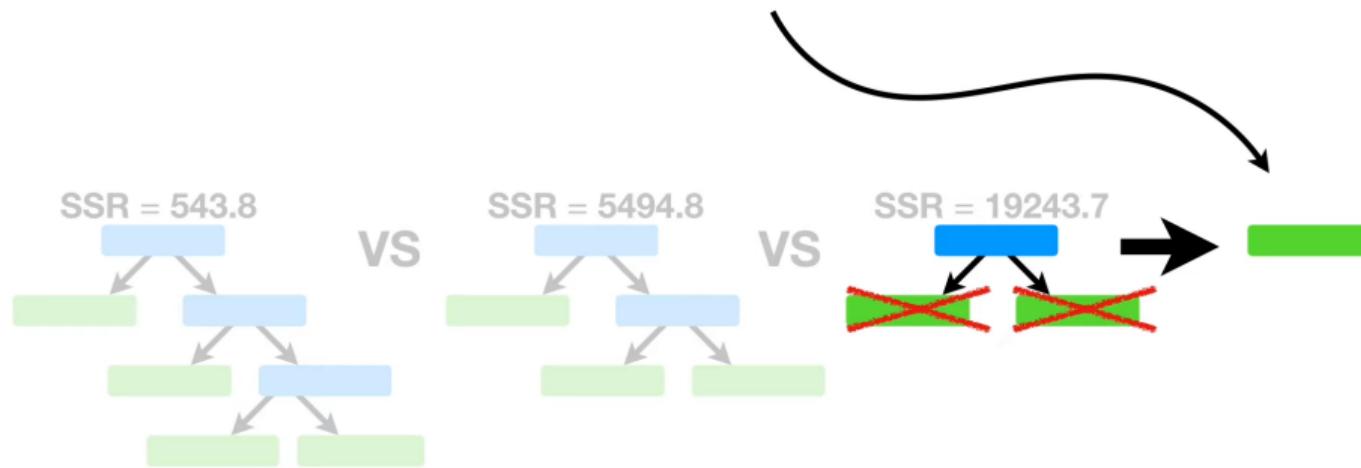
VS



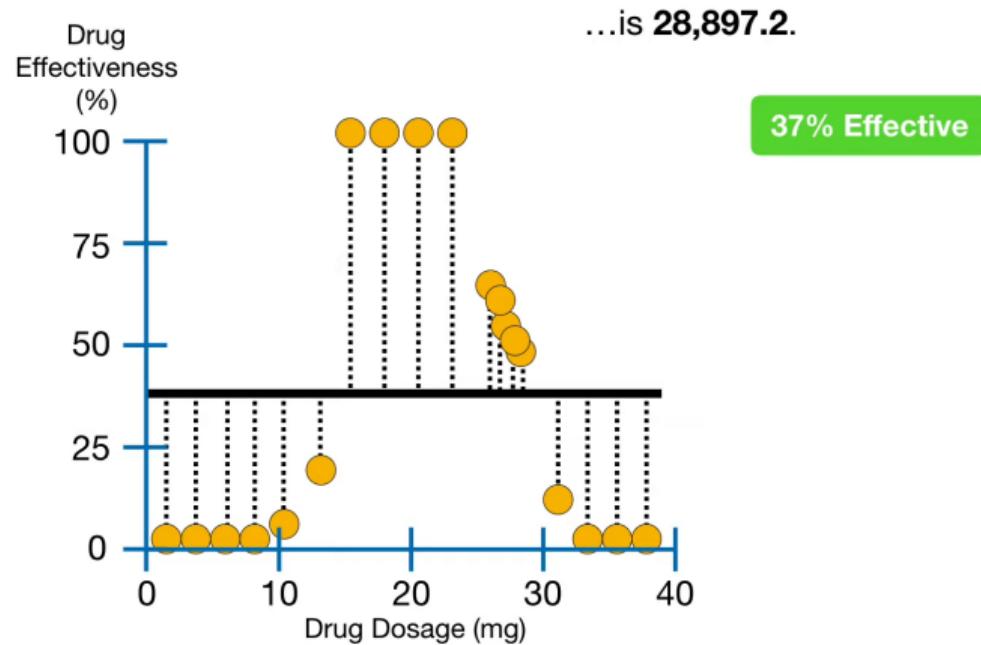
VS

Cost-Complexity Pruning (CCP)

Lastly, the **Sum of Squared Residuals** for the sub-tree with only one leaf...

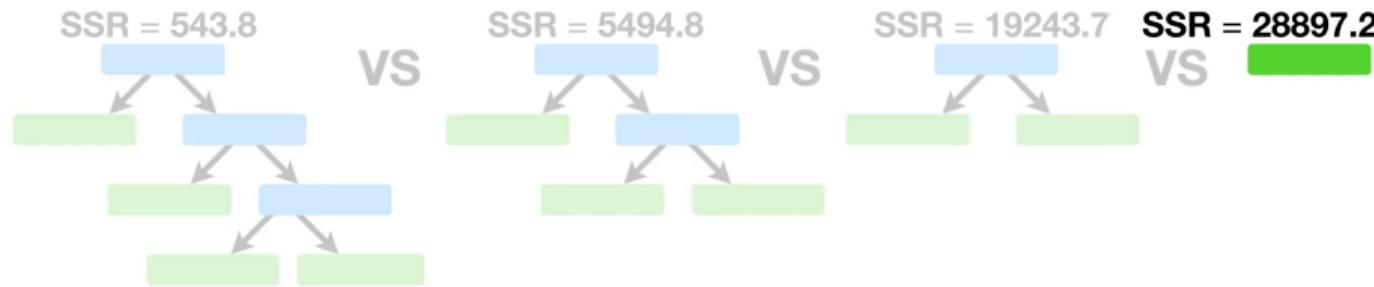


Cost-Complexity Pruning (CCP)



Cost-Complexity Pruning (CCP)

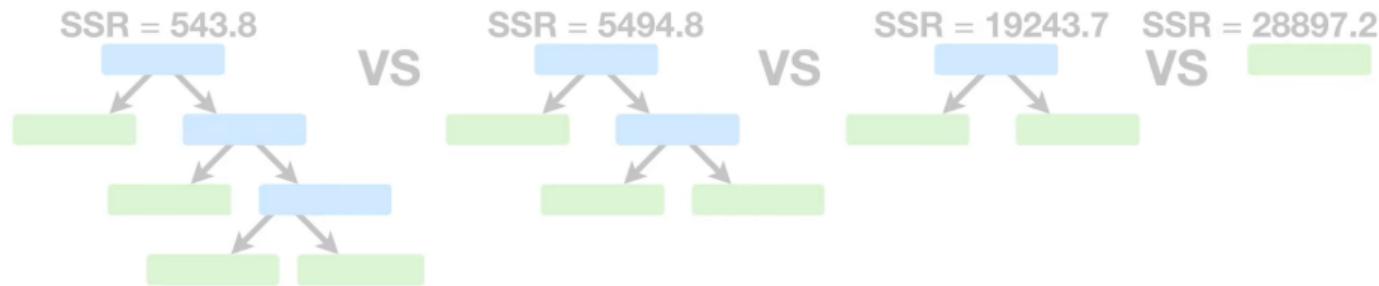
...so let's put **SSR = 28897.2** on top of the sub-tree with **1 leaf**.



Cost-Complexity Pruning (CCP)

NOTE: α (alpha) is a tuning parameter that we finding using **Cross Validation** and we'll talk more about it in a bit.

$$\text{Tree Score} = \text{SSR} + \alpha T$$

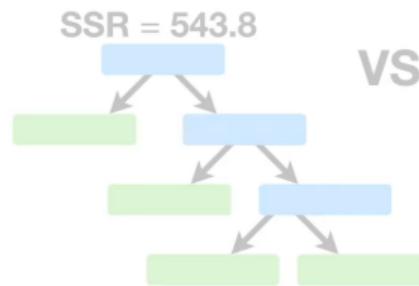


Cost-Complexity Pruning (CCP)

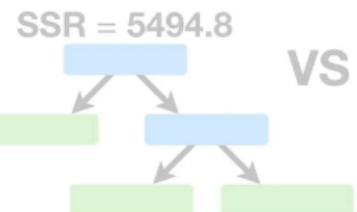
For now, let's let $\alpha = 10,000$.



$$\text{Tree Score} = \text{SSR} + \alpha T$$



VS



VS



$\text{SSR} = 28897.2$



```
graph TD; Root[SSR = 28897.2] --> Node1[ ]; Node1 --> Leaf1[ ]; Node1 --> Node2[ ]; Node2 --> Leaf2[ ]; Node2 --> Node3[ ]; Node3 --> Leaf3[ ]; Node3 --> Leaf4[ ]
```

Cost-Complexity Pruning (CCP)

The **Tree Score** for the original, full sized tree is...

$$\text{Tree Score} = \text{SSR} + 10,000 \times T$$



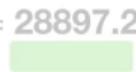
VS



VS



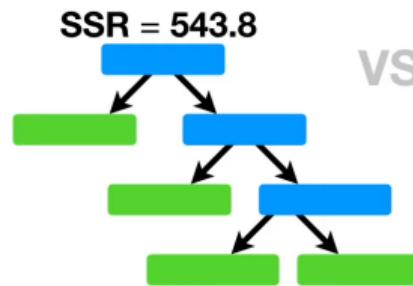
VS



Cost-Complexity Pruning (CCP)

So the **Tree Score** for the original,
full sized tree is **40,543.8**.

$$\text{Tree Score} = 543.8 + 10,000 \times 4 = 40,543.8$$



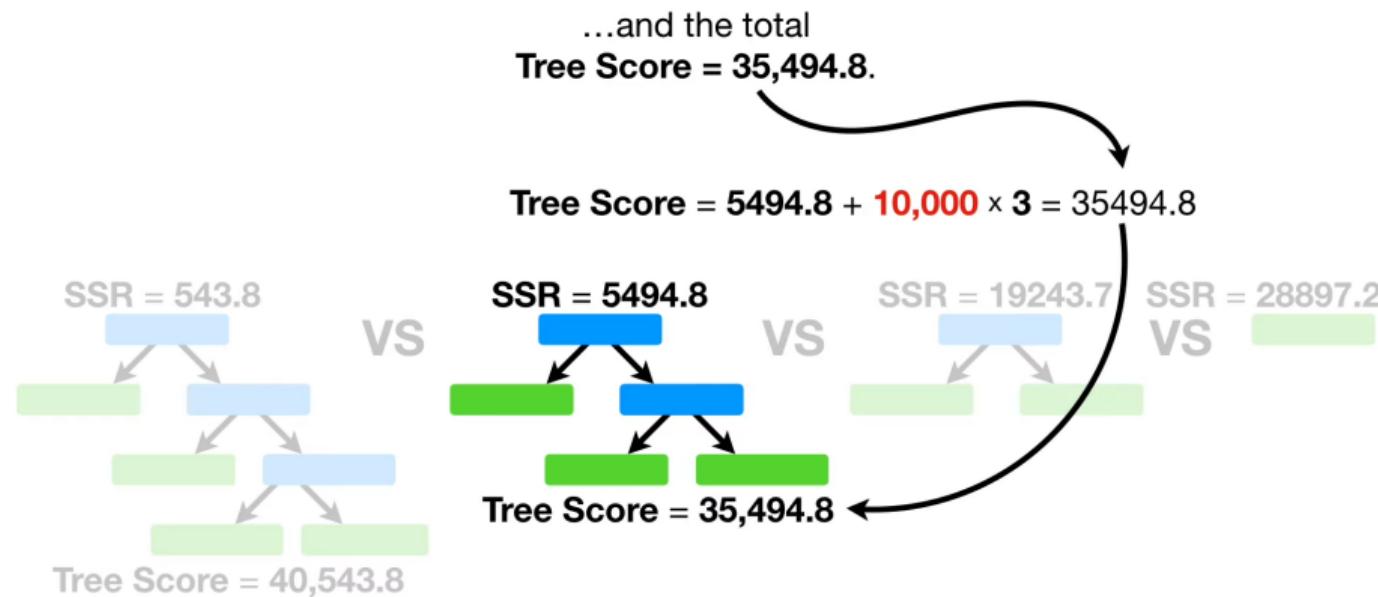
VS



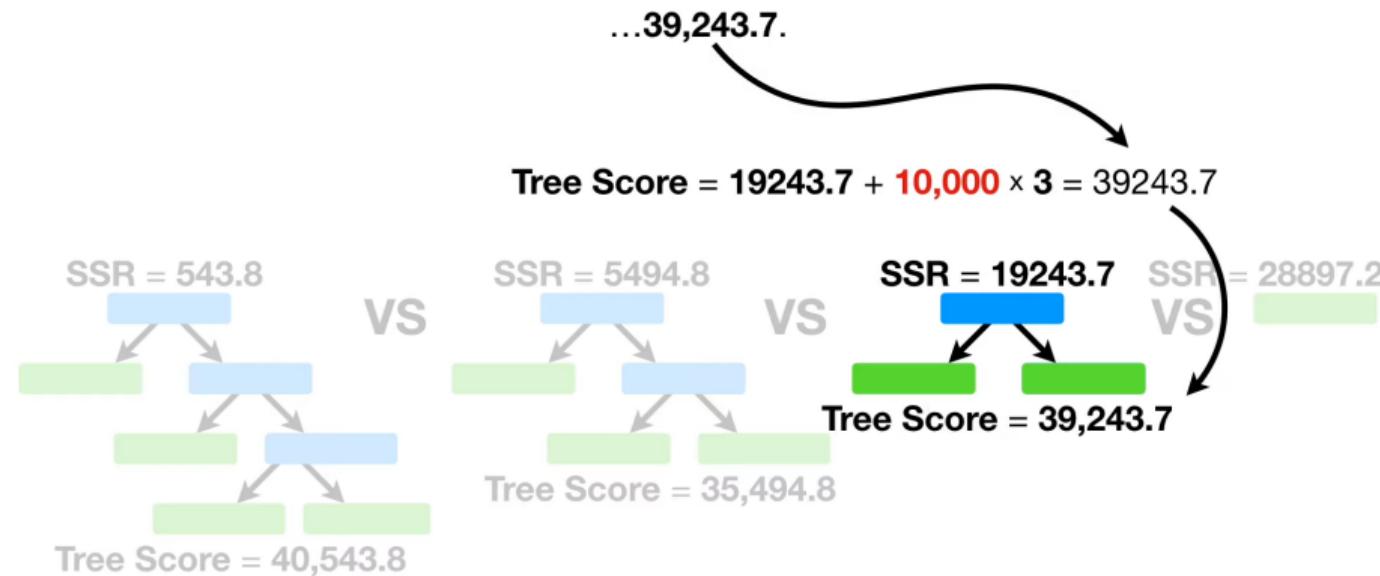
VS



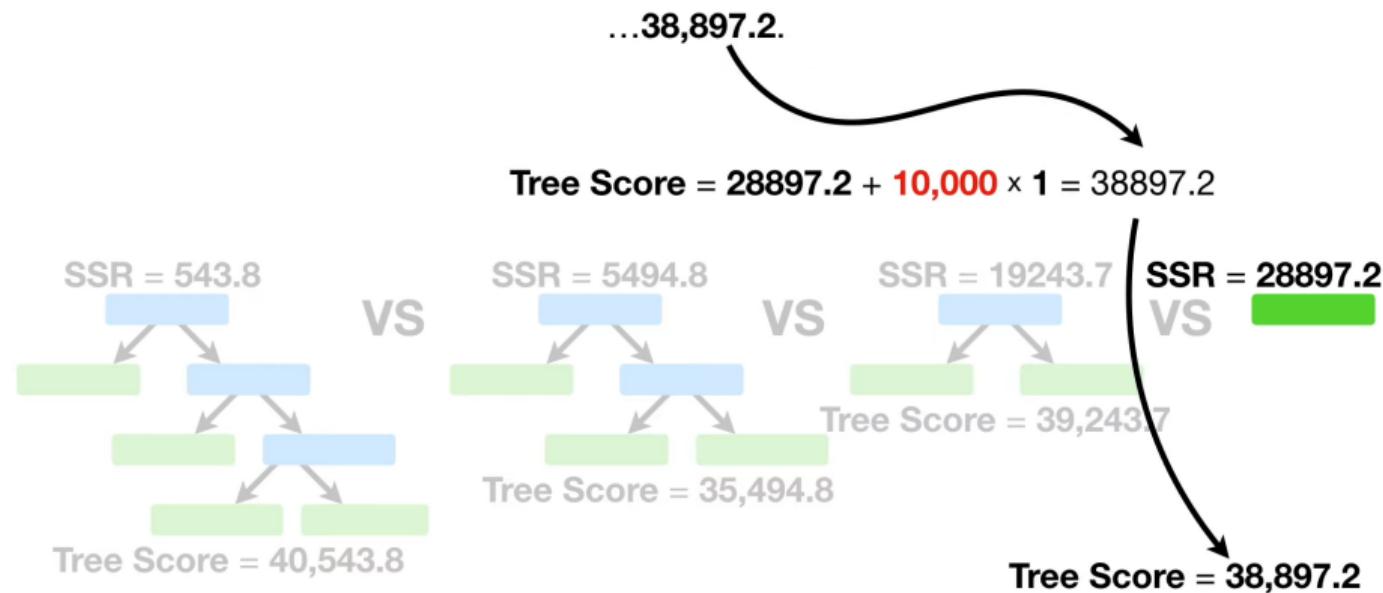
Cost-Complexity Pruning (CCP)



Cost-Complexity Pruning (CCP)

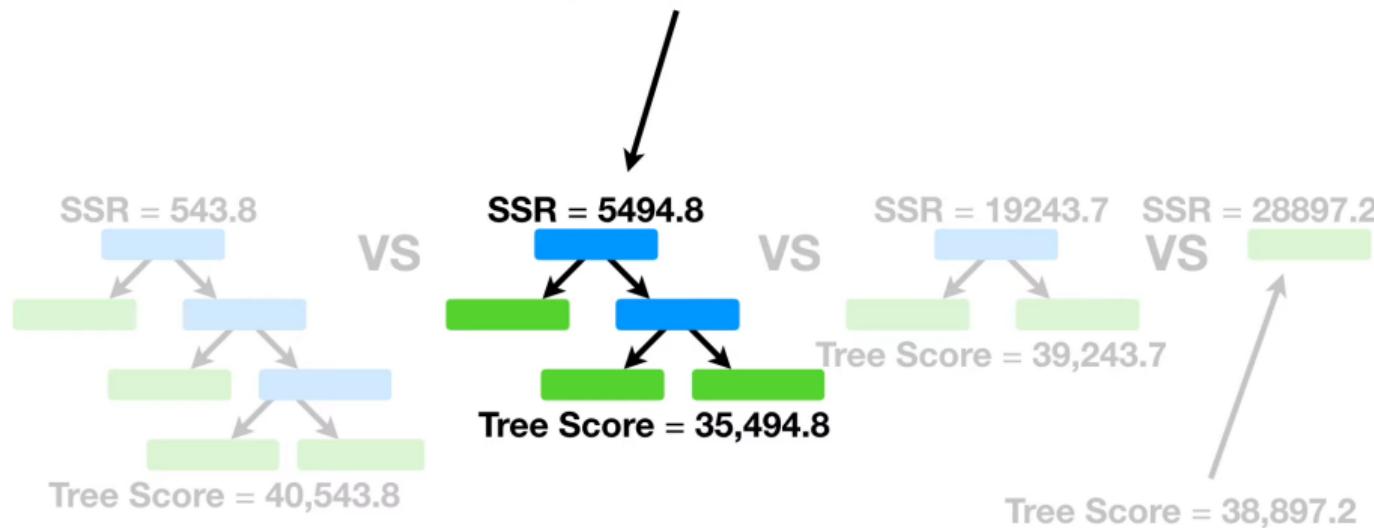


Cost-Complexity Pruning (CCP)



Cost-Complexity Pruning (CCP)

...we pick this sub-tree because it has
the lowest **Tree Score**.

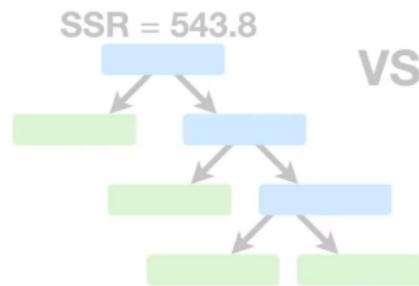


Cost-Complexity Pruning (CCP)

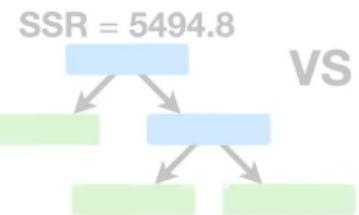
NOTE: If we set $a = 22,000 \dots$



$$\text{Tree Score} = \text{SSR} + aT$$



VS



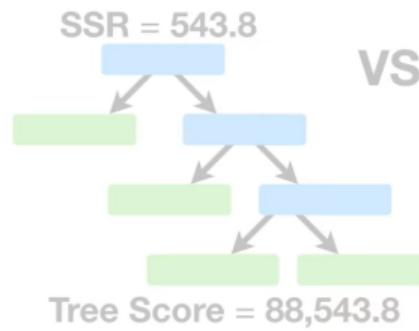
VS



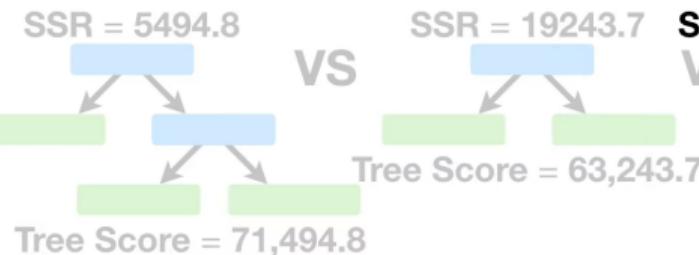
Cost-Complexity Pruning (CCP)

...then we would use the sub-tree with
only one leaf because it has lowest
Tree Score.

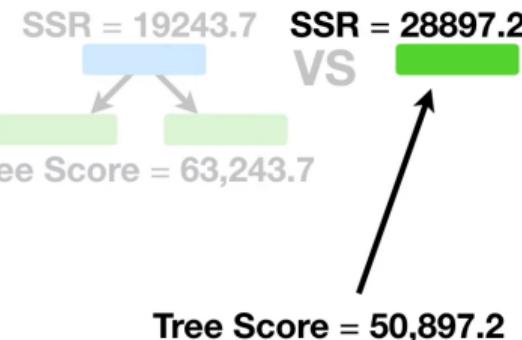
$$\text{Tree Score} = \text{SSR} + 22,000 \times T$$



VS

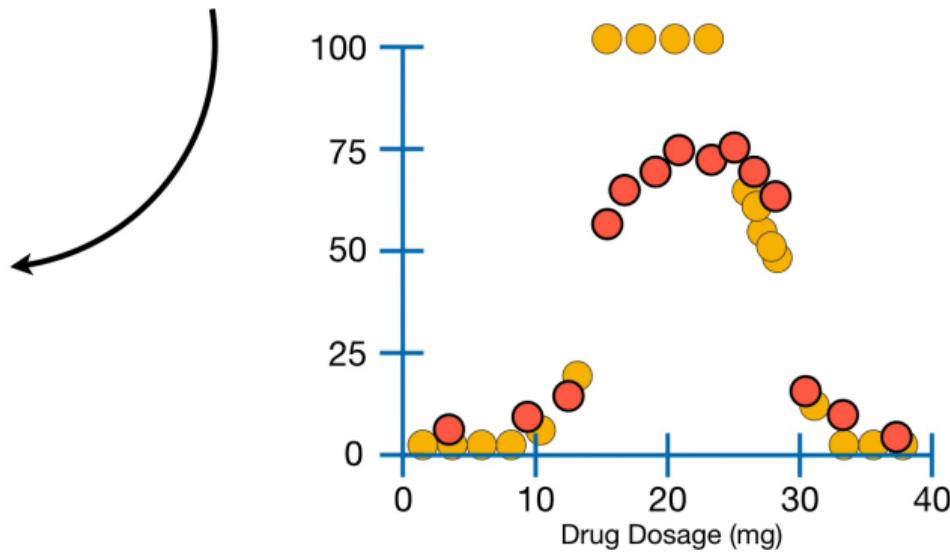
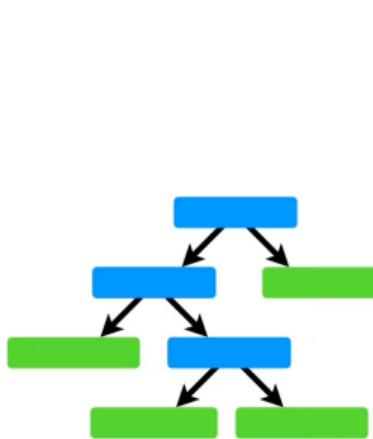


VS



Cost-Complexity Pruning (CCP)

NOTE: This full sized tree is different than before because it was fit to *all* of the data, not just the **Training Data**.

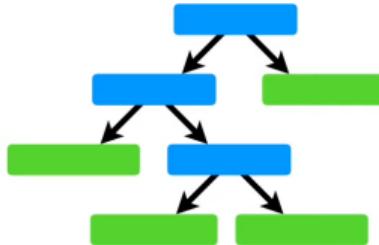


Cost-Complexity Pruning (CCP)

This is because when $\alpha = 0$, the **Tree Complexity Penalty** becomes 0...



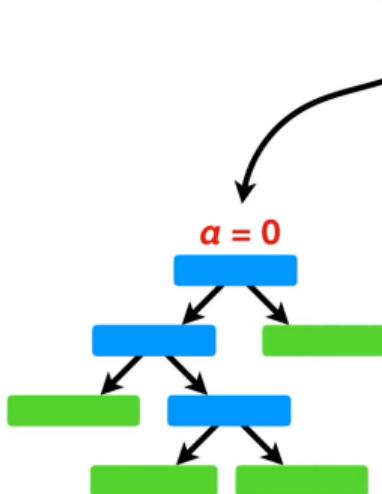
$$\text{Tree Score} = \text{SSR} + 0T$$



Cost-Complexity Pruning (CCP)

So let's put $\alpha = 0$ here, to remind us
that this tree has the lowest **Tree Score**
when $\alpha = 0$.

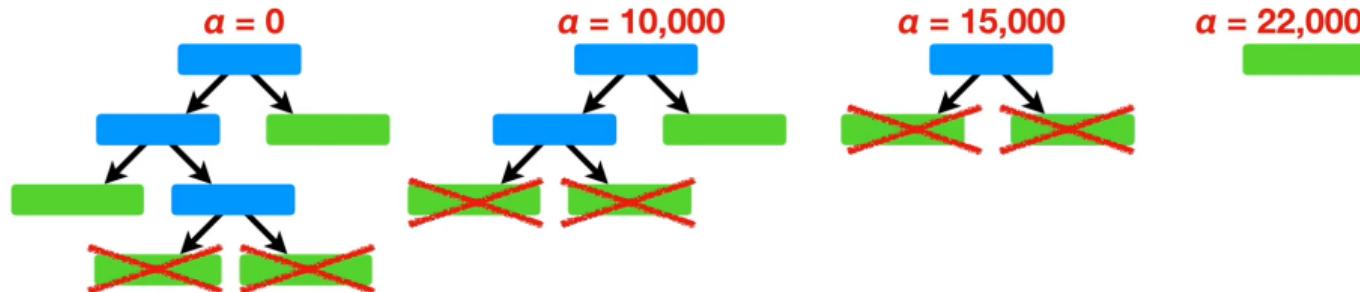
Tree Score = SSR



Cost-Complexity Pruning (CCP)

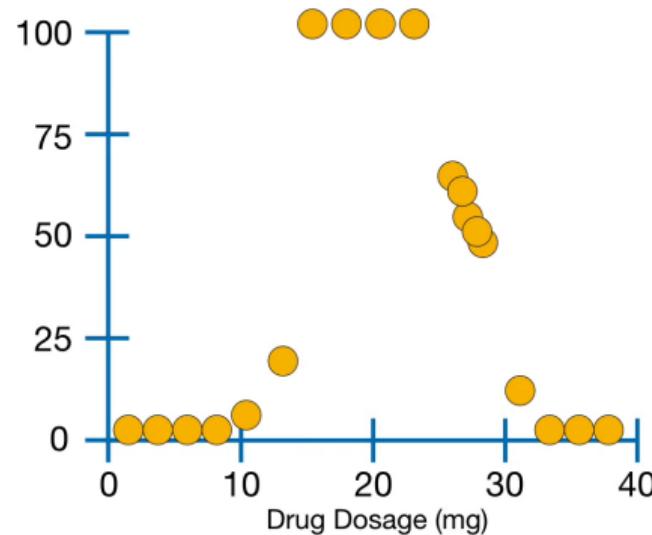
...and use this sub-tree instead.

$$\text{Tree Score} = \text{SSR} + aT$$

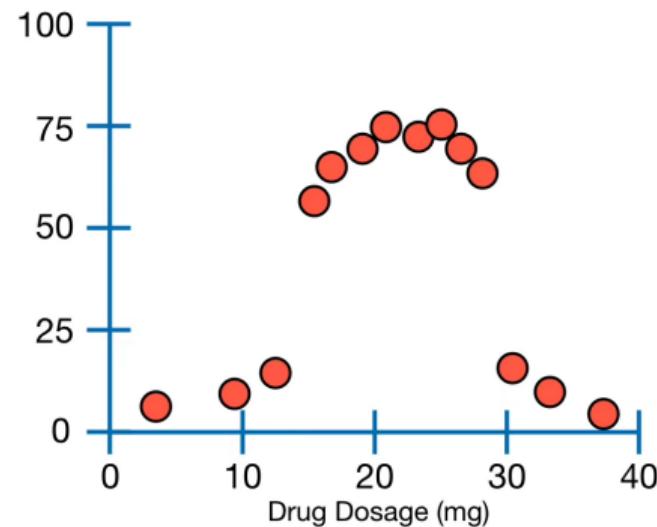


Cost-Complexity Pruning (CCP)

...and divide it into **Training**...

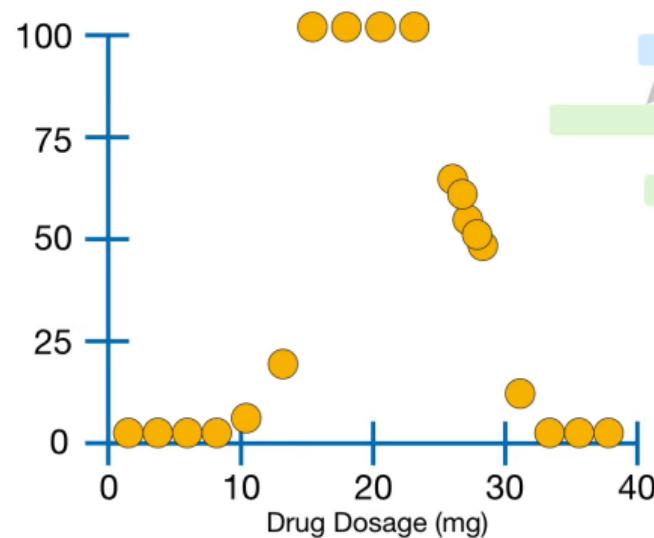


...and **Testing Datasets**.

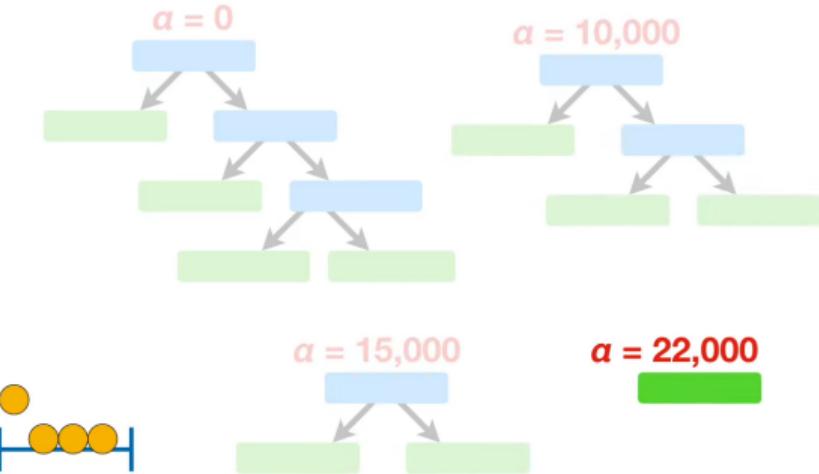


Cost-Complexity Pruning (CCP)

And just using the
Training Data...

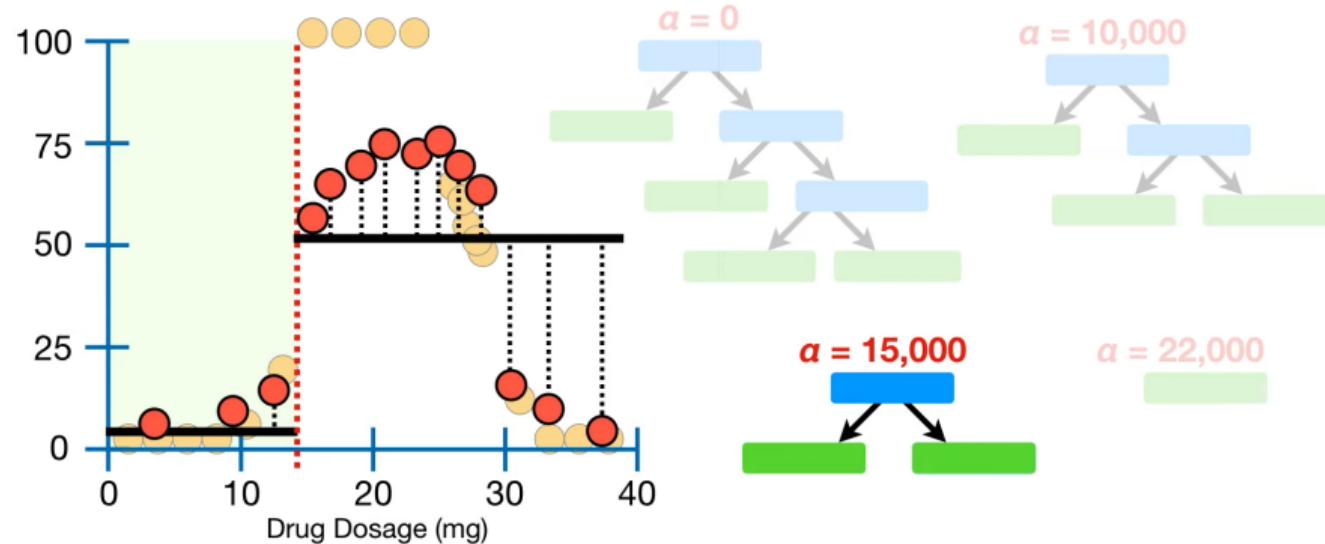


...use the α values we found before to build a full tree and a sequence of sub-trees that minimize the **Tree Score**.



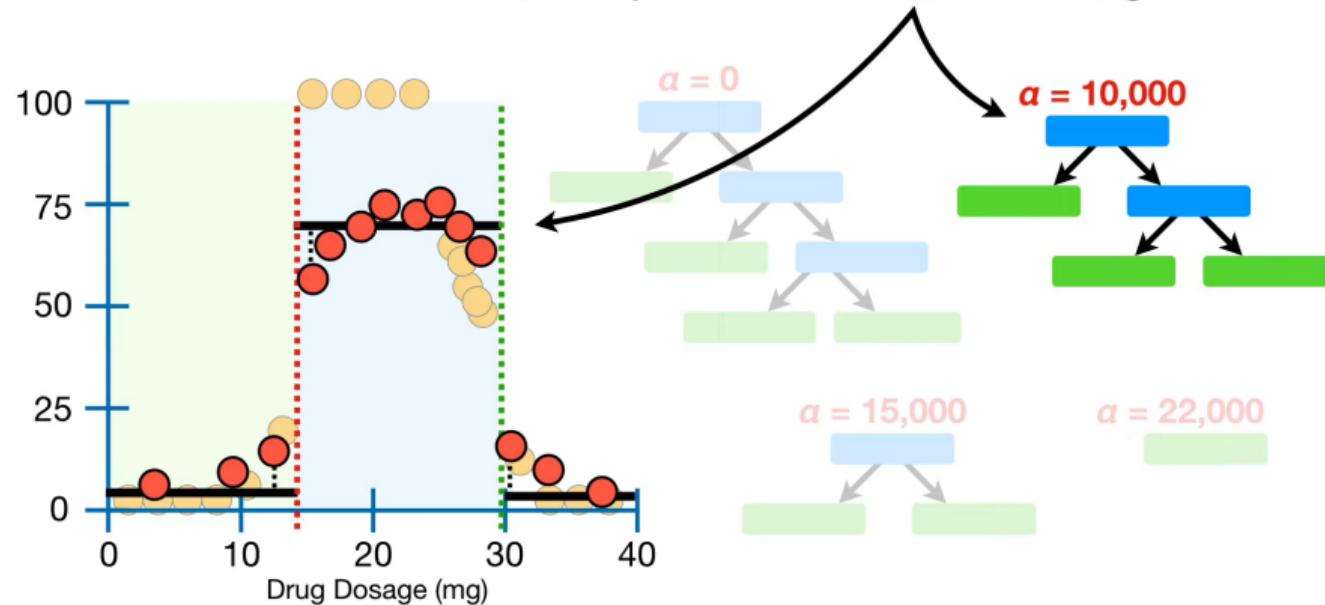
Cost-Complexity Pruning (CCP)

Now calculate the **Sum of Squared Residuals** for each new tree using only the **Testing Data**.



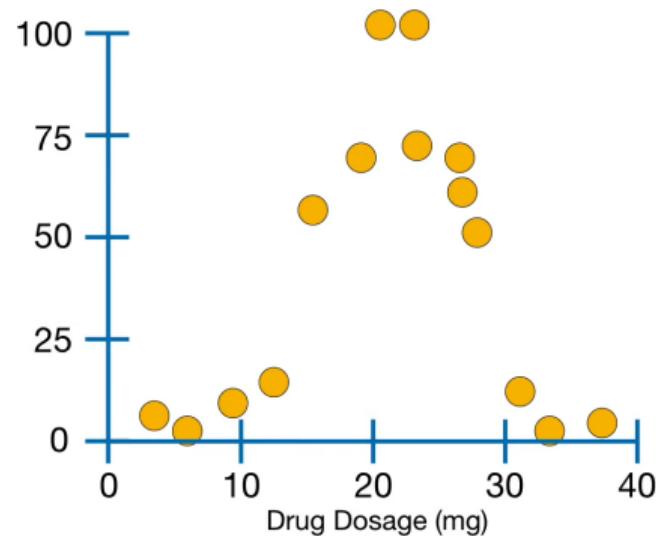
Cost-Complexity Pruning (CCP)

In this case, the tree with $\alpha = 10,000$ had the smallest **Sum of Squared Residuals** for the **Testing Data**.

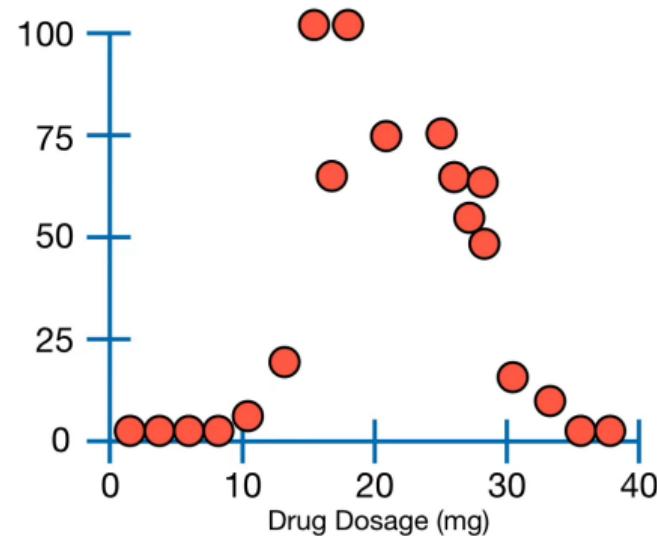


Cost-Complexity Pruning (CCP)

Now we go back and create
new **Training Data**...

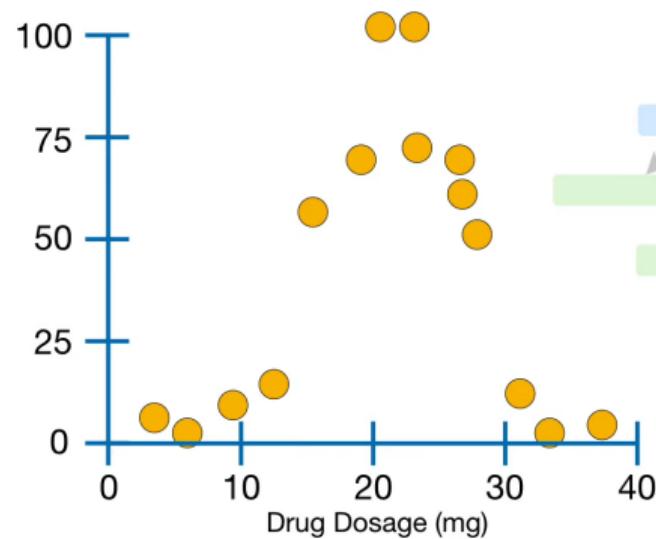


...and new **Testing Data**.

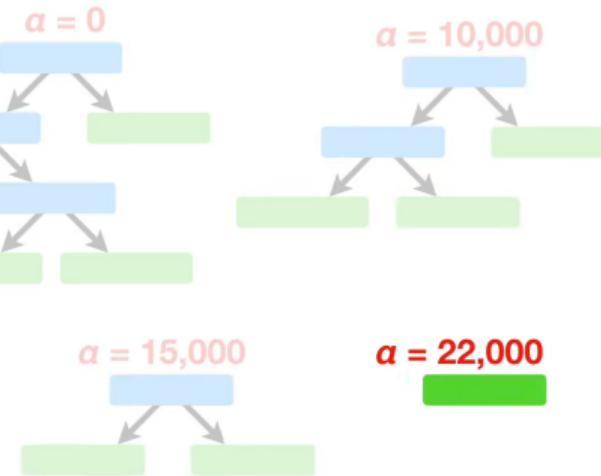


Cost-Complexity Pruning (CCP)

And just using the new
Training Data...

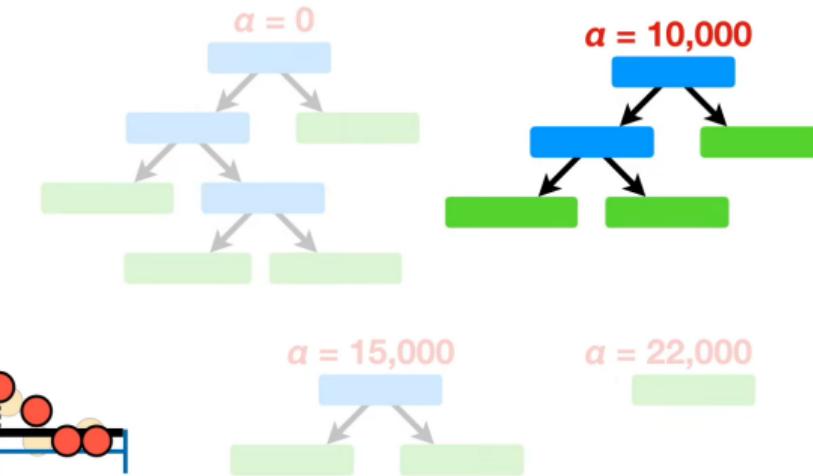
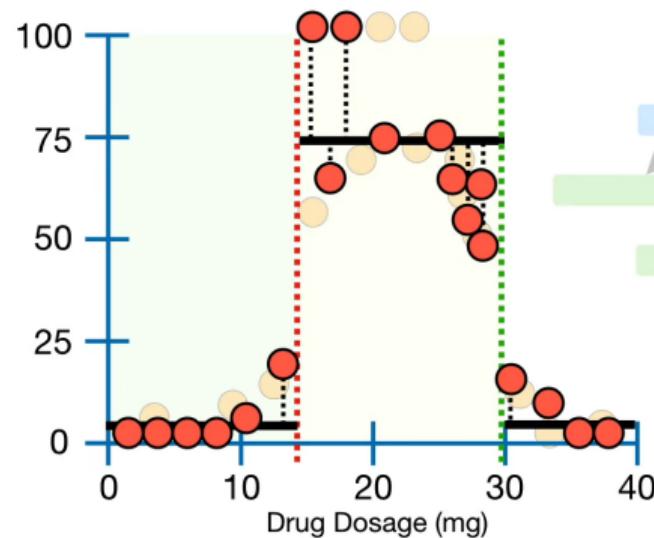


...build a new sequence of trees, from full sized to a leaf, using the α values we found before.

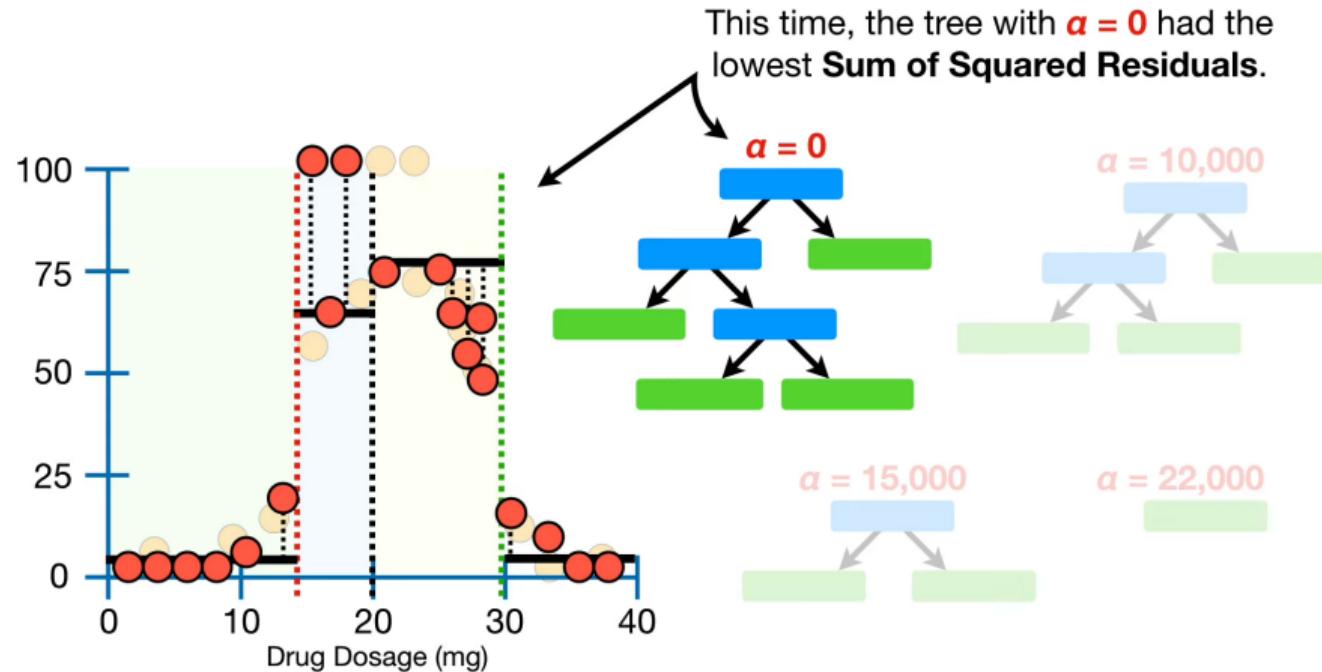


Cost-Complexity Pruning (CCP)

Then we calculate the **Sum of Squared Residuals** using the new **Testing Data**.

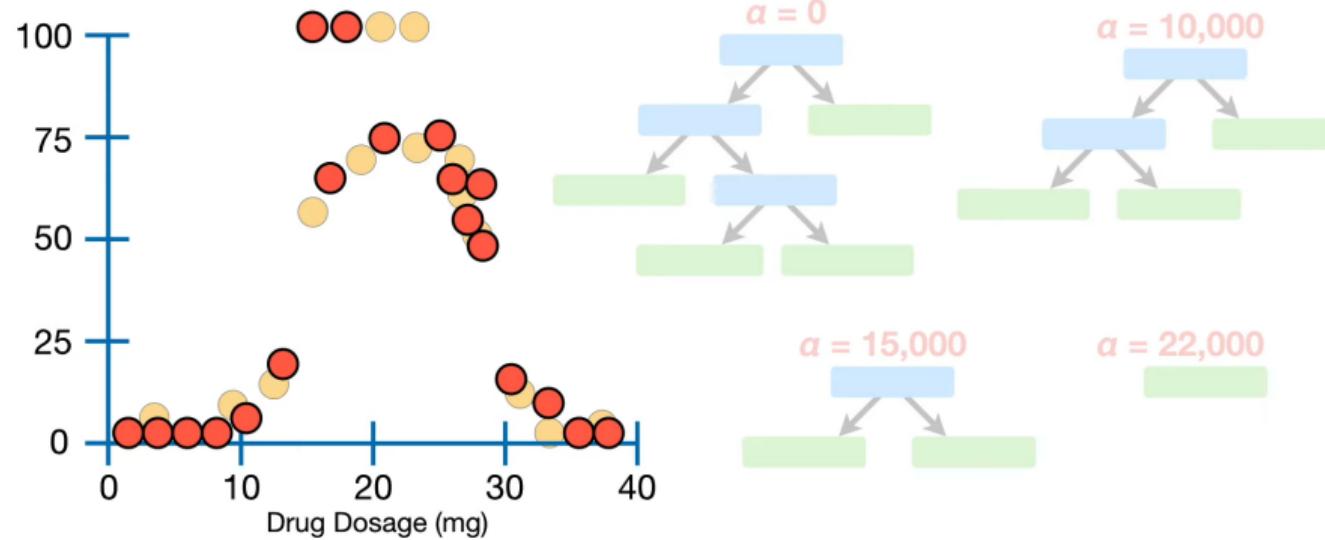


Cost-Complexity Pruning (CCP)

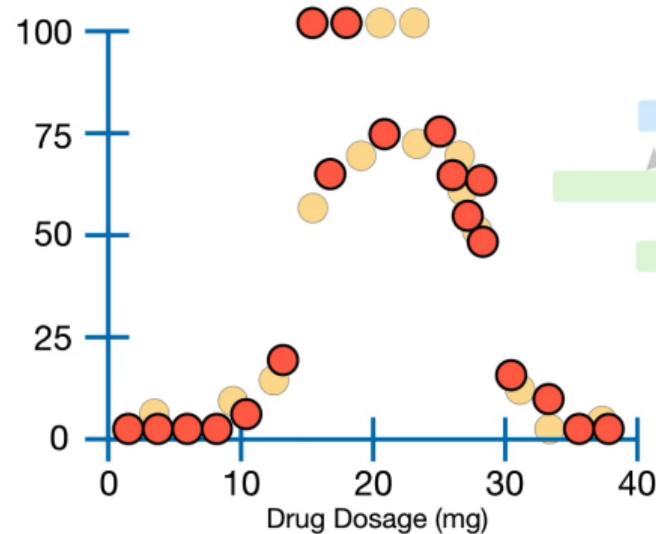


Cost-Complexity Pruning (CCP)

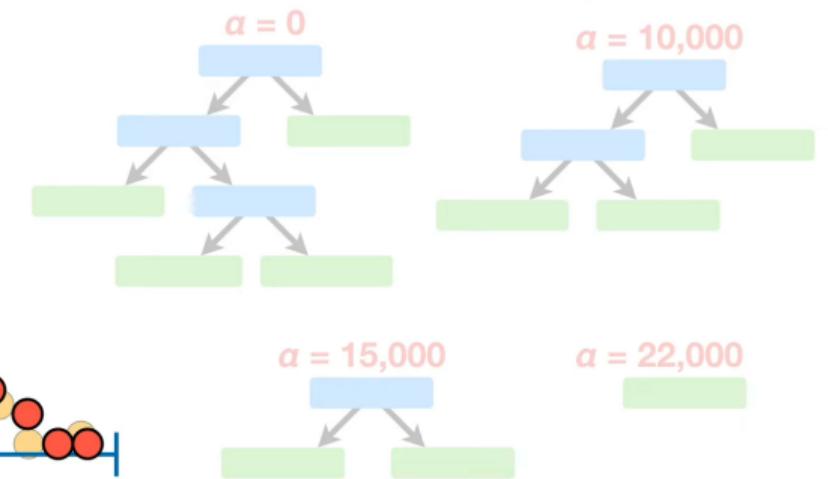
Now we just keep repeating until we have done **10-Fold Cross Validation...**



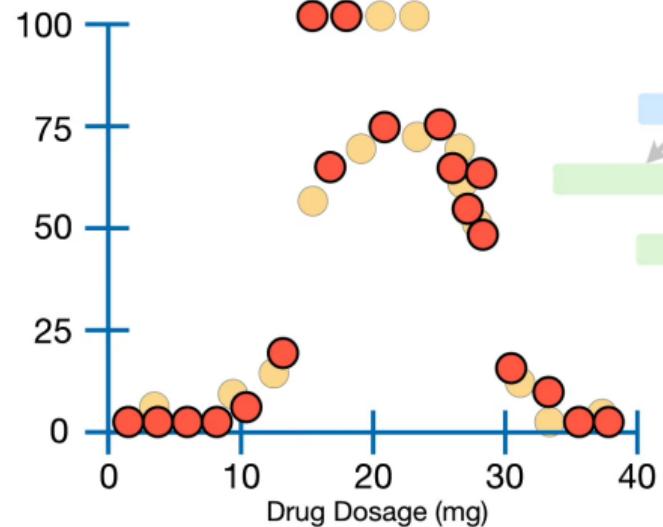
Cost-Complexity Pruning (CCP)



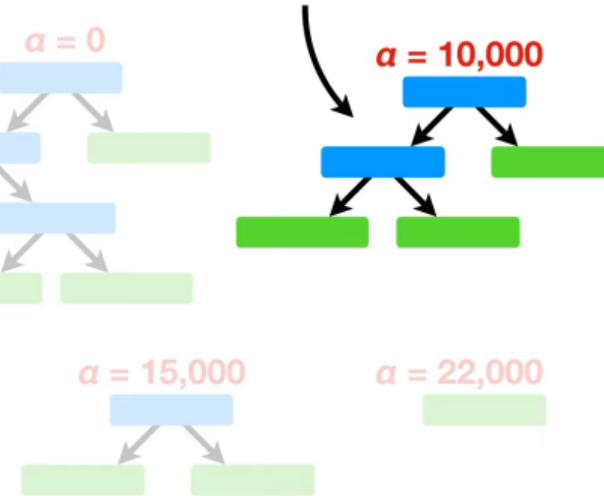
...and the value for α that, on average, gave us the lowest **Sum of Squared Residuals** with the **Testing Data**, is the final value for α .



Cost-Complexity Pruning (CCP)

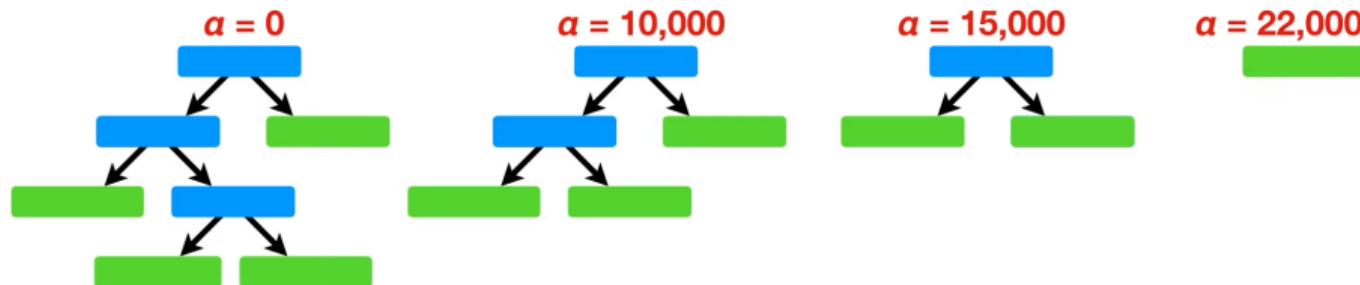


In this case, the optimal trees built with $\alpha = 10,000$ had, on average, the lowest Sum of Squared Residuals...



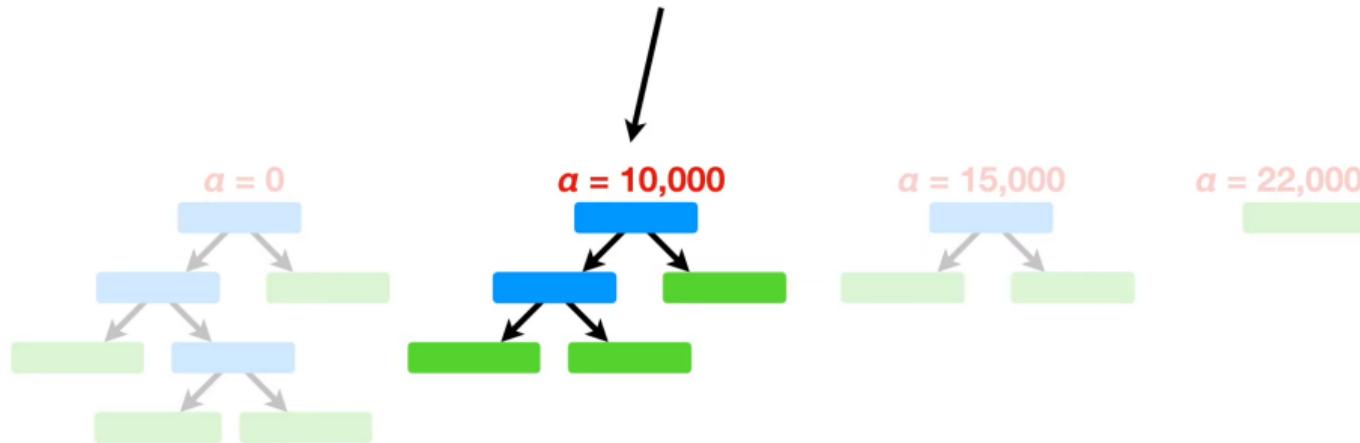
Cost-Complexity Pruning (CCP)

Lastly, we go back to the original trees and sub-trees made from the full data.



Cost-Complexity Pruning (CCP)

...and pick the tree that corresponds to the value for a that we selected ($a = 10,000$).



Cost-Complexity Pruning (CCP)

Ưu điểm:

- Ổn định, giảm overfitting hiệu quả.
- Dễ triển khai, sinh chuỗi cây từ ít đến nhiều lá.
- Áp dụng rộng rãi trong CART, C4.5.

Nhược điểm:

- Cần tính toán nhiều $\alpha \rightarrow$ tốn thời gian với cây lớn.
- Chọn λ tối ưu cần cross-validation.
- Không đảm bảo global optimum (cắt tỉa từng bước).

Critical Value Pruning (CVP)

Ý tưởng:

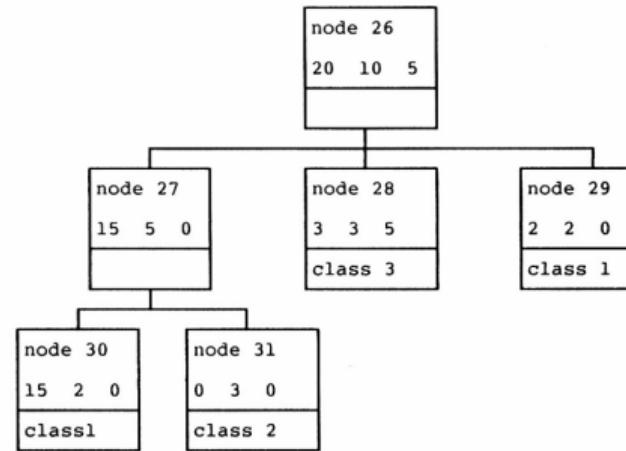
- Mỗi node có measure (Information gain, Gain Ratio, ...)
- Chọn **critical value** (ngưỡng)
- Node measure < critical → prune, trừ khi nhánh có node con vượt ngưỡng

Ảnh hưởng của ngưỡng:

- Critical value càng lớn → prune càng nhiều → cây càng nhỏ
- Thường tạo ra dãy cây pruned, sau đó chọn cây tối ưu

Critical Value Pruning (CVP)

Ví dụ:



Node 27 có $IG = 0.44$

- $\text{Critical} \geq 0.44 \rightarrow$ Node 27 và nhánh con bị prune
- $\text{Critical} < 0.44 \rightarrow$ Node 27 được giữ

Critical Value Pruning (CVP)

Ưu điểm:

- Dễ hiểu, dựa trên measure đã tính
- Giảm overfitting
- Kiểm soát mức độ prune bằng critical value

Nhược điểm:

- Phụ thuộc vào measure đã chọn
- Cần thử nhiều critical để chọn cây tối ưu
- Measure không tốt → prune nhầm node quan trọng

Tổng kết các thuật toán Post-Pruning

REP	MEP
Prune dựa trên tập kiểm tra. Đơn giản, trực quan. Ngăn overfitting, cần data riêng.	Prune dựa trên lỗi dự đoán. Không cần tập kiểm tra. Prune mạnh hơn REP.
CCP	CVP
Tối ưu lỗi + độ phức tạp ($R_\alpha(T)$). Chọn α qua cross-validation. Cân bằng chính xác và phức tạp.	Dựa trên measure (IG, Gain Ratio...). Prune nếu measure < critical value. Giữ nhánh quan trọng.

Nhìn chung tất cả các phương pháp nhằm giảm overfitting và cải thiện khả năng tổng quát của cây quyết định.

Ensembling

- **Định nghĩa:** Xét X là các biến ngẫu nhiên (Random Variable) độc lập và phân phối giống hệt nhau (Independent Identically Distributed).
 - Phương sai của một biến ngẫu nhiên:

$$\mathbf{Var}(X_i) = \sigma^2$$

Trong đó:

- X_i : Biến ngẫu nhiên.
- σ : Độ lệch chuẩn.
- Phương sai của giá trị trung bình mẫu \bar{X} :

$$\mathbf{Var}(\bar{X}) = \mathbf{Var}\left(\frac{1}{n} \sum_i X_i\right) = \frac{\sigma^2}{n}$$

Trong đó:

- \bar{X} : Trung bình mẫu.
- σ : Độ lệch chuẩn.

- Trong thực tế, các biến đều có một mức độ tương quan nhất định với nhau nên giả định biến độc lập là không đúng. \Rightarrow Ta bỏ giả định độc lập nên các biến ngẫu nhiên chỉ còn lại phân phối giống nhau và có tương quan với hệ số ρ (rho).
- Phương sai của giá trị trung bình mẫu:

$$\text{Var}(\bar{X}) = \rho\sigma^2 + \frac{1 - \rho}{n}\sigma^2$$

Trong đó:

- \bar{X} : Trung bình mẫu.
- σ : Độ lệch chuẩn.
- n : Kích thước mẫu.
- ρ : Mức độ tương quan.

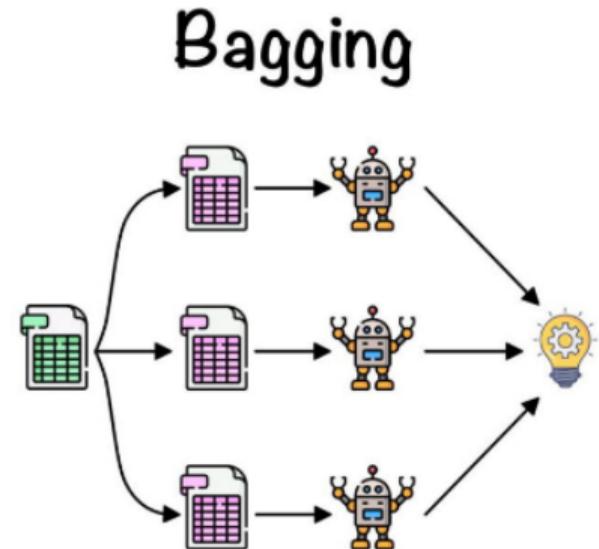
- **Một số cách làm giảm mức độ tương quan (ρ) giữa các mô hình thành phần:**
 - **Sử dụng các thuật toán khác nhau:** Đảm bảo sự đa dạng của các ước lượng cơ sở.
 - **Sử dụng nhiều tập huấn luyện khác nhau:** Thay đổi dữ liệu đầu vào cho mỗi mô hình cơ sở.
 - **Bagging (Bootstrap Aggregating):** Tạo các tập huấn luyện khác nhau bằng cách lấy mẫu ngẫu nhiên có hoàn lại (bootstrap) để giảm ρ và phương sai.
 - **Boosting:** Huấn luyện mô hình tuần tự, mô hình sau sửa lỗi của mô hình trước, tập trung giảm độ chêch (bias).

- **Giả định (Assumption):**

- Thực tế: P là phân phối tổng thể của dữ liệu.
- Giả định: Coi S là tập dữ liệu huấn luyện lấy từ P ($S \approx P$).

- **Bootstrap Samples ($Z_k \sim S$):**

- Tạo ra m tập dữ liệu con bằng cách **lấy mẫu có hoàn lại** từ S .
- Ví dụ: $S = [1, 2, 3, 4, 5] \implies Z_1 = [1, 2, 2, 4, 5]$.



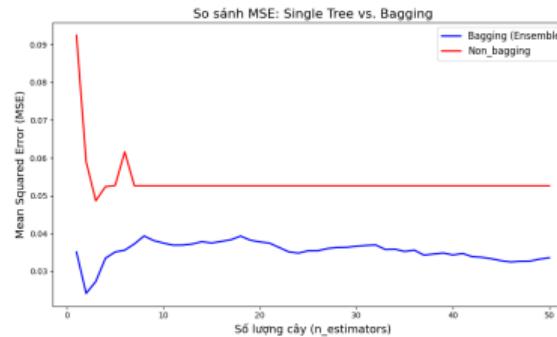
Bagging - Bootstrap Aggregation

- **Aggregation (Tổng hợp kết quả):**

- Huấn luyện mô hình độc lập trên từng Z_k .
- **Regression:** Lấy trung bình cộng các dự đoán (Mean).
- **Classification:** Bỏ phiếu đa số (Majority Voting).

- **Mục tiêu chính:**

- Giảm **Variance** (Phương sai) của mô hình tổng thể nhờ cơ chế trung bình hóa các mô hình con ít tương quan.



Comparison of MSE

Bias - Variance Analysis

- Phương sai của giá trị trung bình mẫu với kích thước mẫu là M :

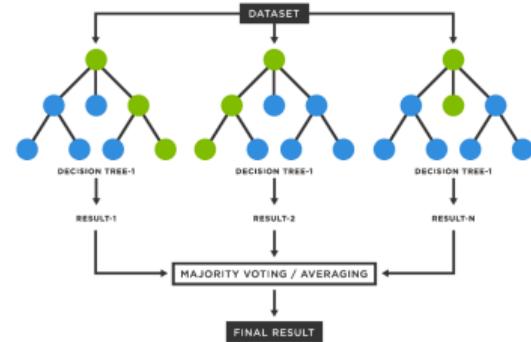
$$\text{Var}(\bar{X}) = \rho\sigma^2 + \frac{1 - \rho}{M}\sigma^2$$

- Bootstrapping làm giảm mối tương quan (ρ) giữa các mô hình.
- Tăng mẫu M làm giảm Variance (Var).
- Bias tăng nhẹ.

- "Trees have one aspect that prevents them from being the ideal tool for predictive learning, namely inaccuracy. The Elements of Statistical Learning
- **Random Forests** là sự kết hợp giữa đơn giản, trực quan của Decision Trees và tận dụng Bagging để làm giảm Variance.

Random Forests: Định nghĩa

- **Định nghĩa:** Random Forest là một cải tiến đáng kể của kỹ thuật Bagging, giúp xây dựng một tập hợp lớn các cây quyết định không tương quan (de-correlated), sau đó tổng hợp kết quả bằng cách lấy trung bình.



- Phương sai của trung bình mẫu:

$$\text{Var}(\bar{X}) = \rho\sigma^2 + \frac{1 - \rho}{M}\sigma^2$$

- Khi số lượng cây M tăng ($M \rightarrow \infty$):
 - Về sau $\frac{1 - \rho}{M}\sigma^2$ sẽ giảm dần về 0.
 - Về đầu $\rho\sigma^2$ giữ nguyên (hằng số).
- **Kết luận:** Mỗi tương quan ρ giữa các cây (Bagged Trees) tạo ra một **giới hạn dưới** cho sai số. Phương sai không thể giảm thấp hơn mức $\rho\sigma^2$.

Random Forest: Variance Reduction

- **Cơ chế giảm phương sai:** Random Forests tăng cường giảm phương sai bằng cách **giảm tương quan (ρ)** giữa các cây con.
- **Cách thực hiện:** Quá trình xây dựng cây sử dụng kỹ thuật **chọn ngẫu nhiên đặc trưng**:

Tại mỗi nút split, chọn ngẫu nhiên $m \leq p$ biến làm ứng viên.

- p : Tổng số biến đầu vào.
- m : Số lượng biến được chọn (thường là \sqrt{p} hoặc nhỏ nhất là 1).

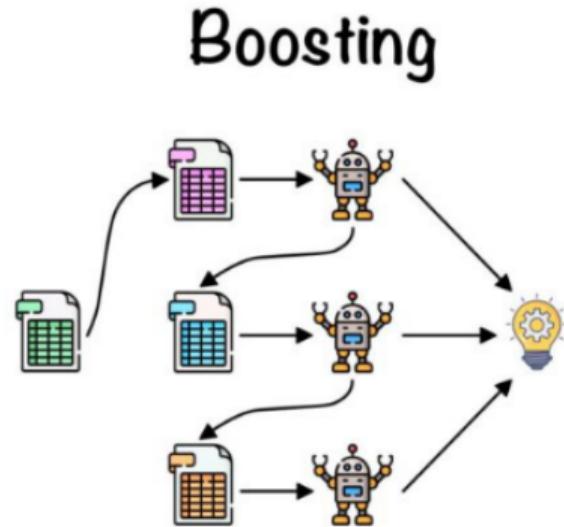
- **Inventors' Recommendation:**

Bài toán	Giá trị mặc định của m	Min Node Size
Phân loại (Classification)	$\lfloor \sqrt{p} \rfloor$	1
Hồi quy (Regression)	$\lfloor p/3 \rfloor$	5

- Độ chính xác của Random Forest (RF) có thể được đánh giá trực tiếp bằng dữ liệu **Out-Of-Bag (OOB)**.
- Cụ thể:**
 - Tỷ lệ các mẫu OOB được **phân loại đúng** (correctly classified) cung cấp một ước lượng tin cậy về **độ chính xác (accuracy)** của mô hình.
 - Ngược lại, tỷ lệ các mẫu OOB bị **phân loại sai** (incorrectly classified) chính là **error of the outside gap**.

Boosting: Định nghĩa

- **Định nghĩa:** Boosting là một kỹ thuật **mô hình hóa hợp thể** (*ensemble modeling*) nhằm mục đích xây dựng một **bộ phân loại mạnh** (*strong classifier*) từ sự kết hợp của nhiều **bộ phân loại yếu** (*weak classifiers*).
- **Mục tiêu chính:**
 - Cải thiện độ chính xác (Accuracy) của mô hình.
 - Tập trung giải quyết vấn đề bằng cách: **Giảm độ chêch** (*Reducing Bias*).



Sequential

Boosting: Cách hoạt động



Boosting: Công thức và Cơ chế Hoạt động

- **Mô hình tổng hợp (Ensemble Model):** Mô hình cuối cùng được xây dựng bằng cách kết hợp tuyến tính các mô hình con có trọng số:

$$G(x) = \sum_{m=1}^M \alpha_m G_m(x)$$

- **Trong đó:**

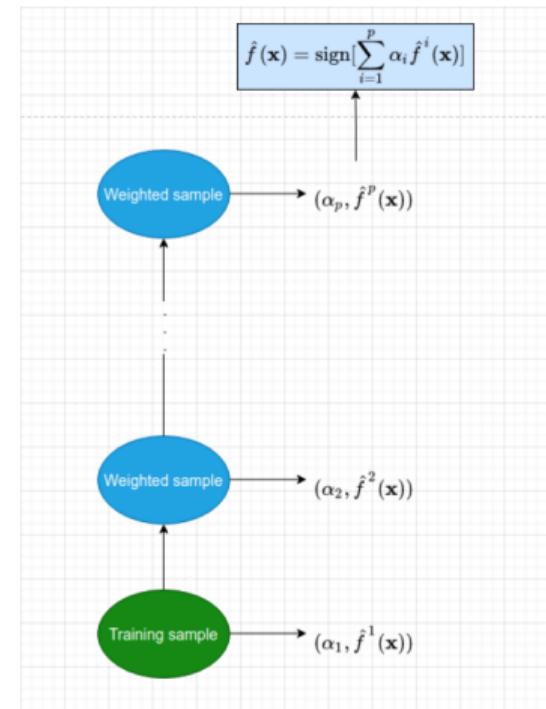
- $G_m(x)$: **Mô hình yếu** (Weak Learner) thứ m .
- α_m : **Trọng số quyết định** của mô hình G_m .
 - Nguyên tắc: Mô hình càng chính xác (sai số thấp) $\rightarrow \alpha_m$ càng lớn.
 - $\alpha_m \propto \log\left(\frac{1 - err_m}{err_m}\right)$.
- $G(x)$: **Mô hình mạnh** cuối cùng (Strong Classifier).

- **Cơ chế trọng số (Reweighting):**

- Dữ liệu được đánh giá lại sau mỗi vòng lặp.
- Các ví dụ bị phân loại **sai** sẽ được **tăng trọng số** để mô hình tiếp theo tập trung giải quyết chúng.

Adaboost: Định nghĩa

- Một mô hình phân loại yếu (weak classifier) có tỷ lệ dự báo sai lớn và giả định nó chỉ tốt hơn so với phân loại ngẫu nhiên một chút.
- Mỗi một mô hình con được huấn luyện từ bộ dữ liệu được đánh trọng số theo tính toán từ mô hình tiền nhiệm. Dữ liệu có trọng số sau đó được đưa vào huấn luyện mô hình tiếp theo.
- Đồng thời ta cũng tính ra một trọng số quyết định α_p thể hiện vai trò của mỗi mô hình ở từng bước huấn luyện.



Adaboost: Định nghĩa

- Kết quả dự báo từ mô hình cuối cùng là một kết hợp từ những mô hình với trọng số a_i :

$$\hat{f}(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^p \alpha_i \hat{f}^i(\mathbf{x}) \right)$$

Trong đó:

- α_i : trọng số quyết định
- $\hat{f}(\mathbf{x})$: kết quả dự báo từ mô hình

Bước 1: Khởi tạo trọng số ban đầu

Ban đầu, gán trọng số bằng nhau cho mỗi điểm dữ liệu:

$$w_i = \frac{1}{N}, \quad \forall i \in \{1, 2, \dots, N\}.$$

trong đó w_i là trọng số của điểm dữ liệu thứ i tại bộ phân loại yếu đầu tiên.

Bước 2: Huấn luyện chuỗi mô hình

Lặp lại quá trình huấn luyện chuỗi mô hình ở mỗi bước b , với $b \in \{1, 2, \dots, B\}$:

Tính sai số huấn luyện:

$$r_b = \frac{\sum_{i=1}^N w_i \mathbf{1}(y_i \neq \hat{f}^b(\mathbf{x}_i))}{\sum_{i=1}^N w_i}$$

- Ở đây $\mathbf{1}(y_i \neq \hat{f}^b(\mathbf{x}_i))$ là những quan sát bị dự báo sai ở mô hình thứ b.
- Giá trị sai số $r_b \in [0, 1]$.

Bước 3: Tính trọng số quyết định cho từng mô hình

$$\alpha_b = \log\left(\frac{1 - r_b}{r_b}\right)$$

AdaBoosting: Step by step

Cập nhật trọng số cho từng quan sát

$$w_i := w_i \exp \left[\alpha_b \mathbf{1}(y_i \neq \hat{f}^b(\mathbf{x}_i)) \right]$$

với $\forall i = \overline{1, N}$. Như vậy ta có thể nhận thấy rằng:

$$w_i := \begin{cases} w_i & \text{nếu } y_i = \hat{f}^b(\mathbf{x}_i) \\ w_i \exp(\alpha_b) & \text{nếu } y_i \neq \hat{f}^b(\mathbf{x}_i) \end{cases}$$

- **Ý tưởng chính:**

- Học từ những sai lầm
- Dựa trên sức mạnh của tập thể
- Sử dụng gradient để biết hướng nào giúp giảm sai số nhanh nhất

→ Gradient Boosting là xây dựng một **mô hình cộng gộp** một cách tuần tự, trong đó mỗi mô hình con mới sinh ra có nhiệm vụ **triệt tiêu sai số (Residuals)** của các mô hình trước đó, giống như việc **leo xuồng núi (Gradient Descent)** từng bước nhỏ để đến nơi sai số thấp nhất

Gradient Boosting: Sử dụng cho Regression

- **Input:** Tập dữ liệu $\{(x_i, y_i)\}_{i=1}^n$ và một **hàm mất mát** khả vi $L(y, F(x))$

$$L(y, F(x)) = \frac{1}{2}(\text{Observed} - \text{Predicted})^2$$

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56

Bước 1: Khởi tạo mô hình bằng một hằng số

$$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^n L(y_i, \gamma)$$

- $F_0(x)$: Mô hình dự đoán ban đầu (bước 0).
- y_i : Giá trị quan sát (thực tế) của mẫu dữ liệu thứ i .
- γ : Giá trị dự đoán
- Kết quả giải toán cho thấy giá trị γ tối ưu hóa hàm mất mát chính là **giá trị trung bình** của tất cả quan sát i

$$F_0(x) = \frac{1}{N} \sum_{i=1}^n y_i = \frac{88 + 76 + 56}{3} = 73.3$$

Bước 2: Vòng lặp từ $m = 1$ đến M

- (A) Tính phần dư giả (pseudo-residuals) r_{im} :

$$r_{im} = - \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)}$$

- (B) Huấn luyện cây trên r_{im} , tạo vùng lá R_{jm} ($j = 1 \dots J_m$).
- (C) Tính giá trị đầu ra tối ưu γ_{jm} cho mỗi lá:

$$\gamma_{jm} = \operatorname{argmin}_{\gamma} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$$

- (D) Cập nhật mô hình (với tốc độ học η):

$$F_m(x) = F_{m-1}(x) + \eta \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$$

Gradient Boosting: Sử dụng cho Regression

- (A) Tính phần dư giả (pseudo-residuals) r_{im} :

$$r_{im} = - \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)} \quad \text{với } i \text{ là chỉ số của tập dữ liệu và } m \text{ là chỉ số của cây}$$

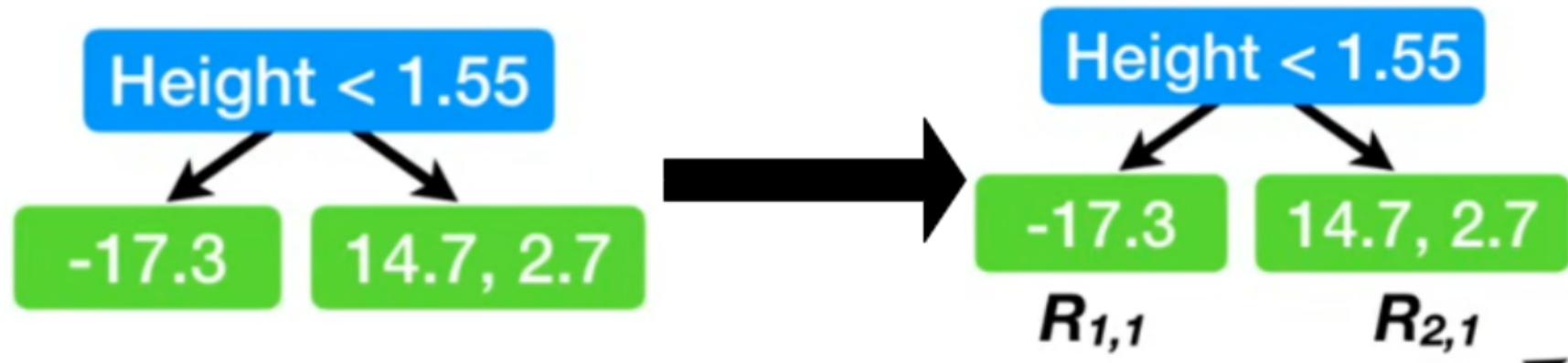
- Đạo hàm ta được:

$$r_{im} = - \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)} = \text{Residual} = \text{Observed} - \text{Predicted}$$

Height (m)	Favorite Color	Gender	Weight (kg)	$r_{i,1}$
1.6	Blue	Male	88	14.7
1.6	Green	Female	76	2.7
1.5	Blue	Female	56	-17.3

Gradient Boosting: Sử dụng cho Regression

- (B) Huấn luyện cây trên r_{im} , tạo vùng lá R_{jm} ($j = 1 \dots J_m$) với j là chỉ số của nút lá, m là chỉ số của cây
- Dùng Height, Color, Gender để xây dựng một cây quyết định mới để dự đoán các Residuals chứ không phải dự đoán cân nặng



Gradient Boosting: Sử dụng cho Regression

- (C) Tính giá trị đầu ra tối ưu γ_{jm} cho mỗi lá:

$$\gamma_{jm} = \operatorname{argmin}_{\gamma} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$$

- Do cây bị giới hạn độ sâu, nhiều mẫu dữ liệu sẽ rơi vào cùng một lá. Ta cần tìm một giá trị đại diện (γ_{jm}) cho lá đó. Hay ta có:

$$\gamma_{jm} = \operatorname{argmin}_{\gamma} \sum_{x_i \in R_{ij}} \frac{1}{2} (y_i - (F_{m-1}(x_i) + \gamma))^2$$

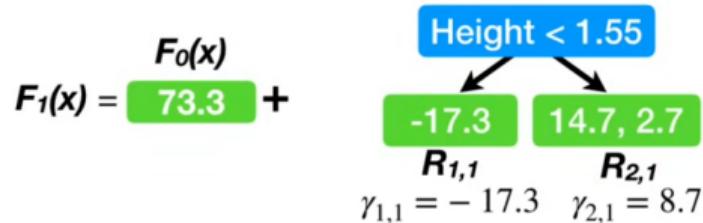
$$\gamma_{1,1} = \operatorname{argmin}_{\gamma} \frac{1}{2} (56 - (73.3 + \gamma))^2 \rightarrow \gamma = -17.3$$

$$\gamma_{2,1} = \operatorname{argmin}_{\gamma} [\frac{1}{2} (y_1 - (F_{m-1}(x_1) + \gamma))^2 + \frac{1}{2} (y_2 - (F_{m-1}(x_2) + \gamma))^2] \rightarrow \gamma = \frac{14.7 + 2.7}{2}$$

Gradient Boosting: Sử dụng cho Regression

- (D) Cập nhật mô hình (với tốc độ học η):

$$F_m(x) = F_{m-1}(x) + \eta \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$$



- Cập nhật dự đoán mới:

- $F_1(x_1) = 73.3 + 0.1 \times 8.7 = 74.2$
- $F_1(x_2) = 73.3 + 0.1 \times 8.7 = 74.2$
- $F_1(x_3) = 73.3 + 0.1 \times (-17.3) = 71.6$

Gradient Boosting: Sử dụng cho Classification

Xây dựng hàm mất mát

① Xuất phát điểm (Likelihood Function):

$$\mathcal{L} = \prod_{i=1}^n p_i^{y_i} (1 - p_i)^{1-y_i}$$

② Chuyển sang Log-Likelihood (Biên Tích thành Tổng):

$$\log(\mathcal{L}) = \sum_{i=1}^n [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

③ Negative Log-Likelihood (Để cực tiểu hóa):

$$\text{Loss} = -\log(\mathcal{L})$$

④ Tham số hóa bằng Log-odds ($F(x)$):

- Ta có: $p = \frac{e^{F(x)}}{1+e^{F(x)}} \Rightarrow \log(p) = F(x) - \log(1 + e^{F(x)})$
- Và: $1 - p = \frac{1}{1+e^{F(x)}} \Rightarrow \log(1 - p) = -\log(1 + e^{F(x)})$

Gradient Boosting: Sử dụng cho Classification

⑤ Thể và Rút gọn:

$$\begin{aligned} L &= - \left[y_i(F(x) - \log(1 + e^{F(x)})) + (1 - y_i)(-\log(1 + e^{F(x)})) \right] \\ &= - \left[y_i F(x) \underbrace{- y_i \log(1 + e^{F(x)}) + y_i \log(1 + e^{F(x)})}_{\text{Triệt tiêu nhau}} - \log(1 + e^{F(x)}) \right] \end{aligned}$$

⑥ Kết quả cuối cùng:

$$L(y, F(x)) = -y_i F(x) + \log(1 + e^{F(x)})$$

Gradient Boosting: Sử dụng cho Classification

- **Input:** Tập dữ liệu $\{(x_i, y_i)\}_{i=1}^n$ và một **hàm mất mát** khả vi $L(y, F(x))$

$$L(y_i, F(x_i)) = -y_i F(x_i) + \log(1 + e^{F(x_i)})$$

Likes Popcorn	Age	Favorite color	Loves Cinema
Yes	12	Blue	Yes
No	87	Green	Yes
No	44	Blue	No

Bước 1: Khởi tạo mô hình bằng một hằng số

$$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^n L(y_i, \gamma)$$

- $F_0(x)$: Mô hình dự đoán ban đầu (bước 0).
 - y_i : Giá trị quan sát (thực tế) của mẫu dữ liệu thứ i .
 - γ : Giá trị dự đoán (Ở đây chính là $\log(odds)$)
-
- $F_0(x)$ chính là $\log(odds)$ của toàn bộ dữ liệu ban đầu để các cây sau dựa vào đó sửa sai

$$F_0(x) = \log\left(\frac{2}{1}\right) = 0.69$$

Gradient Boosting: Sử dụng cho Classification

- (A) Tính phần dư giả (pseudo-residuals) r_{im} :

$$r_{im} = - \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)} \quad \text{với } i \text{ là chỉ số của tập dữ liệu và } m \text{ là chỉ số của cây}$$

- Đạo hàm ta được:

$$r_{im} = - \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)} = - \left[-y_i + \frac{e^{F(x_i)}}{1 + e^{F(x_i)}} \right] = y_i - p_i$$

Likes Popcorn	Age	Favorite color	Loves Cinema	$r_{i,1}$
Yes	12	Blue	Yes	0.33
No	87	Green	Yes	0.33
No	44	Blue	No	-0.67

Gradient Boosting: Sử dụng cho Classification

- (B) Huấn luyện cây trên r_{im} , tạo vùng lá R_{jm} ($j = 1 \dots J_m$) với j là chỉ số của nút lá, m là chỉ số của cây
- Dùng Likes Popcorn, Age, Favorite color để xây dựng một cây quyết định mới để dự đoán các Pseudo-residuals



Gradient Boosting: Sử dụng cho Classification

- (C) Tính giá trị đầu ra tối ưu γ_{jm} cho mỗi lá:

$$\gamma_{jm} = \operatorname{argmin}_{\gamma} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$$

- Hay ta có:

$$\gamma_{jm} = \operatorname{argmin}_{\gamma} \sum_{x_i \in R_{ij}} -y_i[F_{m-1}(x_i) + \gamma] + \log(1 + e^{F_{m-1}(x_i) + \gamma})$$

- Vì phương trình trên chứa γ trong hàm logarit nên không có nghiệm chính xác. Ta sử dụng **Khai triển Taylor bậc 2** để xấp xỉ:

$$L(y_i, F_{m-1}(x_i) + \gamma) \approx L(y_i, F_{m-1}(x_i)) + g_i\gamma + \frac{1}{2}h_i\gamma^2$$

Trong đó:

- $g_i = \frac{\partial L}{\partial F} = -(y_i - p_i)$ (Gradient bậc 1)
- $h_i = \frac{\partial^2 L}{\partial F^2} = p_i(1 - p_i)$ (Hessian bậc 2)

Gradient Boosting: Sử dụng cho Classification

- Để tìm γ tối ưu, ta lấy đạo hàm theo γ và cho bằng 0:

$$\frac{\partial}{\partial \gamma} \sum \left(g_i \gamma + \frac{1}{2} h_i \gamma^2 \right) = \sum (g_i + h_i \gamma) = 0$$

- Kết quả cuối cùng:

$$\boxed{\gamma_{jm} = -\frac{\sum g_i}{\sum h_i} = \frac{\sum_{x_i \in R_{jm}} (y_i - p_i)}{\sum_{x_i \in R_{jm}} p_i (1 - p_i)}}$$

- $\gamma_{1,1} = \frac{Residual_1}{p_1(1-p_1)} = \frac{0.33}{0.67(1-0.67)} = 1.5$

- $\gamma_{2,1} = \frac{Residual_2 + Residual_3}{[p_2(1-p_2)] + [p_3(1-p_3)]} = \frac{0.33 - 0.67}{[0.67(1-0.67)] + [0.67(1-0.67)]} = -0.77$

Gradient Boosting: Sử dụng cho Classification

- (D) Cập nhật mô hình (với tốc độ học η):

$$F_m(x) = F_{m-1}(x) + \eta \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$$



- Cập nhật dự đoán mới:

- $F_1(x_1) = 0.69 + 0.8 \times 1.5 = 1.89$
- $F_1(x_2) = 0.69 + 0.8 \times (-0.77) = 0.07 \rightarrow$ Dự đoán tệ hơn trước nhưng những cây sau sẽ tập trung sửa lại điểm này
- $F_1(x_3) = 0.69 + 0.8 \times (-0.77) = 0.07$

So sánh Bagging và Boosting

Đặc điểm	Bagging (VD: Random Forest)	Boosting (VD: AdaBoost, GBM)
Cơ chế	Song song: Xây dựng độc lập cùng lúc.	Tuần tự: Cây sau sửa lỗi cho cây trước.
Mục tiêu	Giảm Phương sai: Tăng độ ổn định, chống Overfit.	Giảm Độ lệch: Tăng độ chính xác từ mô hình yếu.
Dữ liệu	Bootstrap: Lấy mẫu ngẫu nhiên có hoàn lại.	Trọng số: Tập trung vào các điểm bị dự đoán sai.
Mô hình cơ sở	Phức tạp: Cây sâu (Low Bias, High Variance).	Đơn giản: Cây nông (High Bias, Low Variance).
Nhiều	Mạnh mẽ: Ít bị ảnh hưởng.	Nhạy cảm: Dễ bị Overfit nếu nhiều.