Decision Trees and Ensemble Methods

Prof. Hariprasad Kodamana

November 12, 2025

Introduction

1. Decision Tree is one of the commonly used approaches for supervised learning. It can be used to solve both Regression and Classification tasks. However, the latter is more popular in practical application.
2. The regression tree is used when the predicted outcome is a real number, and the classification tree is used to predict the class to which the data belongs.
3. The attributes can be continuous or categorical in both regression and classification trees. However, the output is continuous in the former whereas categorical in the later.
4. A decision tree is a tree in which each branch node represents a choice between number of alternatives and each leaf node represents a decision.
5. A node with outgoing edges is called an internal or test node. All other nodes are called leaves (also known as terminal or decision nodes).

Construction of a Decision Tree

1. In the first step, analyze all attributes and select the one that will function as the best root.
2. Break-up the training set into subsets based on the branches of the root node.
3. Test the remaining attributes to see which ones fit best underneath the branches of the root node.
4. Continue this process for all other branches until:
    1. all the examples of a subset are of one type,
    2. there are no examples left,
    3. there are no more attributes left.

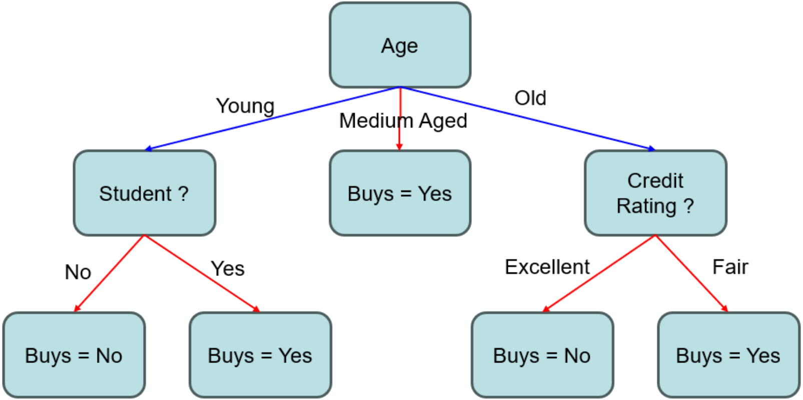
Popular Variants:

1. Decision Trees
2. ID3 Algorithm (Iterative Dichotomizer 3, developed by Quinlan 1975)
3. C4.5 Algorithm (Extended from ID3 by Quinlan)

Example 1 of Decision Tree: Dataset

SR No.	Age	Income	Student?	Credit Rating?	Buys Computer?
1	<= 30 (Young)	High	No	Fair	No
2	<= 30 (Young)	High	No	Excellent	No
3	31 – 40 (Middle Aged)	High	No	Fair	Yes
4	> 40 (Old)	Medium	No	Fair	Yes
5	> 40 (Old)	Low	Yes	Fair	Yes
6	> 40 (Old)	Low	Yes	Excellent	No
7	31 – 40 (Middle Aged)	Low	Yes	Excellent	Yes
8	<= 30 (Young)	Medium	No	Fair	No
9	<= 30 (Young)	Low	Yes	Fair	Yes
10	> 40 (Old)	Medium	Yes	Fair	Yes
11	<= 30 (Young)	Medium	Yes	Excellent	Yes
12	31 – 40 (Middle Aged)	Medium	No	Excellent	Yes
13	31 – 40 (Middle Aged)	High	Yes	Fair	Yes
14	> 40 (Old)	Medium	No	Excellent	No

Example 1 of Decision Tree: Result



Example 2 of Decision Tree: Transportation Dataset

SR. No.	Gender	Car Ownership	Travel Cost	Income Level	Transportation
1	Male	0	Cheap	Low	Bus
2	Male	1	Cheap	Medium	Bus
3	Female	1	Cheap	Medium	Train
4	Female	0	Cheap	Low	Bus
5	Male	1	Cheap	Medium	Bus
6	Male	0	Standard	Medium	Train
7	Female	1	Standard	Medium	Train
8	Female	1	Expensive	High	Car
9	Male	2	Expensive	Medium	Car
10	Female	2	Expensive	High	Car

Aim is to train a model which can predict the suitable transportation.

Example 2: Prediction Task

Predict the mode of transportation using the trained model for the following data:

SR. No.	Gender	Car Ownership	Travel Cost	Income Level	Transportation
1	Male	1	Standard	High	?
2	Male	0	Cheap	Medium	?
3	Female	1	Cheap	High	?

Example 2: The ID3 Algorithm

Algorithm Steps

1. Calculate the Information Entropy of every attribute using the training dataset.
2. Split the set into subsets using the attribute for which information entropy is minimum (or, equivalently, information gain is maximum).
3. Select the above attribute as a decision tree node.
4. Perform the same procedure as above using remaining attributes on the subsets obtained.

Splitting Criterion 1: Information Entropy

1. A measure of homogeneity of the sample data is the Information Entropy. The concept is given by Shannon. Lower is the Entropy, higher is the homogeneity of the samples.
2. A completely homogeneous sample has entropy of 0 (leaf node). An equally divided sample has entropy of 1.
3. The formula for entropy is as follows. For an output with three classes (Bus, Train, Car), $m = 3$:

$$\text{Entropy}(S) = -\sum_{k=1}^{m} p_k \log_2 p_k$$

Where, p_k is the proportion of S belonging to class k.

Splitting Criterion 2: Information Gain

4. The information gain is based on the decrease in entropy after a dataset is split on an attribute.
5. The information gain is calculated as follows:

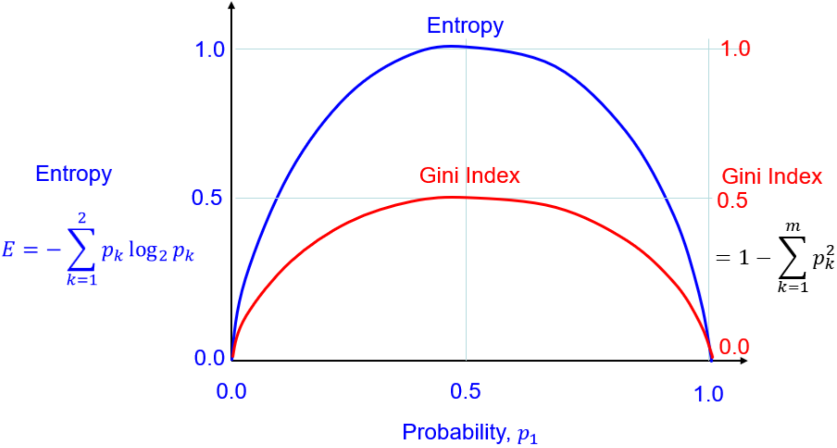
$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v)$$

Where S_v is the subset of S for which attribute A has value v. For the attribute 'Gender', v would be Male and Female.

Splitting Criterion 3: Gini Index

6. Gini Index is popularly used in Random Forest and is calculated as follows:

$$\text{Gini}(S) = 1 - \sum_{k=1}^{m} p_k^2$$



The Calculation Steps

1. First the entropy of the total dataset is calculated (i.e., $E(S)$).
2. The dataset is then split on the different attributes (i.e., A).
3. The entropy for each branch is calculated (i.e., $E(S_v)$).
4. Then it is added proportionally, to get total entropy for the split.

$$I(S, A) = \sum_{v=1}^{m_v} \frac{|S_v|}{|S|} \times E(S_v)$$

5. The resulting entropy is subtracted from the entropy before the split.
6. The result is the Information Gain (IG) or decrease in entropy (i.e., Gain(S, A)).
7. The attribute that yields the largest IG is chosen for the decision node.

Step 1: Calculate Entropy of the Full Dataset

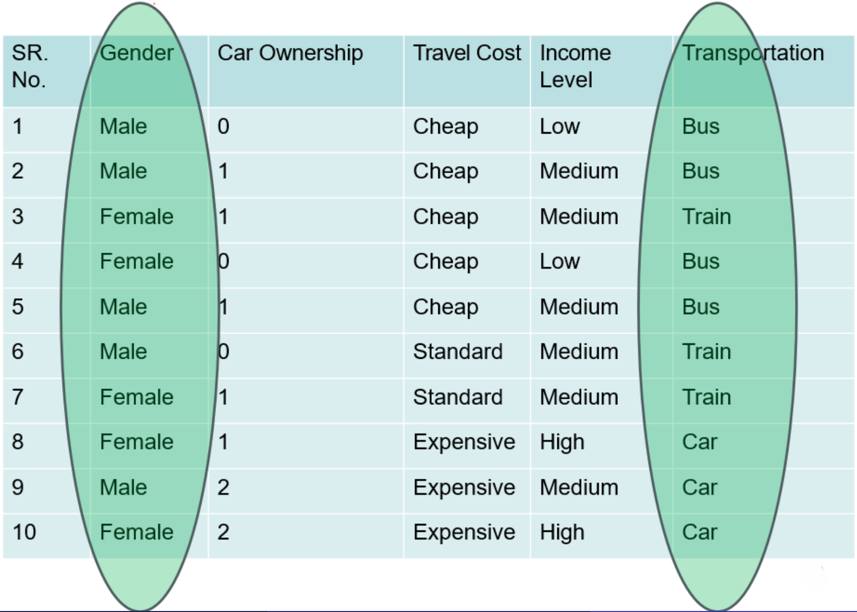
The entropy of the training set is:

$$\text{Entropy}(S) = E(S) = -\sum_{k=1}^{m} p_k \log_2 p_k$$

Probability of Bus $= \frac{4}{10} = 0.4$
Probability of Train $= \frac{3}{10} = 0.3$
Probability of Car $= \frac{3}{10} = 0.3$

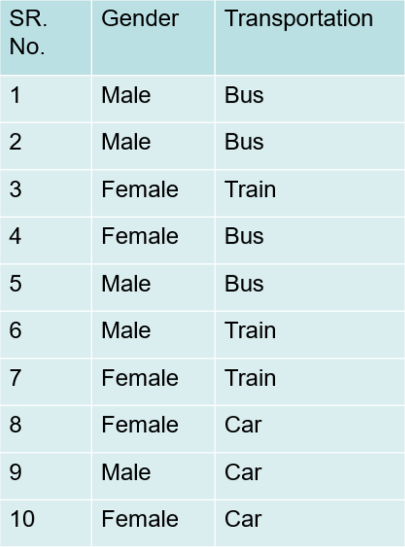
$$\begin{aligned} E(S) &= -0.4\log_2 0.4 - 0.3\log_2 0.3 - 0.3\log_2 0.3 \\ &\approx -0.4(-1.32) - 0.3(-1.74) - 0.3(-1.74) \\ &\approx 0.528 + 0.522 + 0.522 \\ &= 1.572 \end{aligned}$$

Split the Data Based on Features

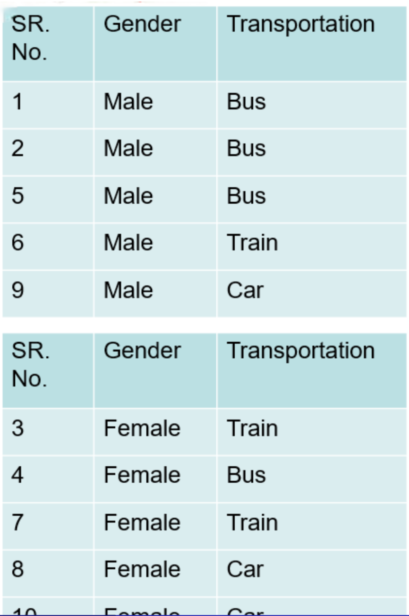


SR. No.	Gender	Car Ownership	Travel Cost	Income Level	Transportation
1	Male	0	Cheap	Low	Bus
2	Male	1	Cheap	Medium	Bus
3	Female	1	Cheap	Medium	Train
4	Female	0	Cheap	Low	Bus
5	Male	1	Cheap	Medium	Bus
6	Male	0	Standard	Medium	Train
7	Female	1	Standard	Medium	Train
8	Female	1	Expensive	High	Car
9	Male	2	Expensive	Medium	Car
10	Female	2	Expensive	High	Car

Split the Data Based on Features



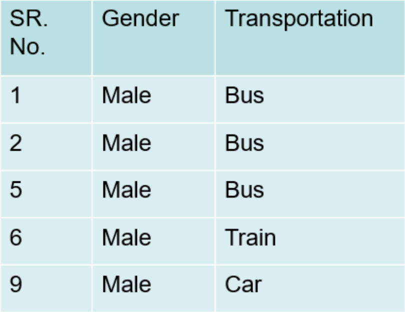
SR. No.	Gender	Transportation
1	Male	Bus
2	Male	Bus
3	Female	Train
4	Female	Bus
5	Male	Bus
6	Male	Train
7	Female	Train
8	Female	Car
9	Male	Car
10	Female	Car



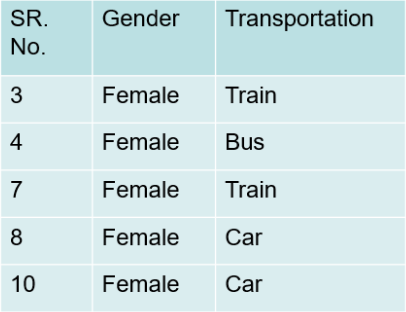
SR. No.	Gender	Transportation
1	Male	Bus
2	Male	Bus
5	Male	Bus
6	Male	Train
9	Male	Car

SR. No.	Gender	Transportation
3	Female	Train
4	Female	Bus
7	Female	Train
8	Female	Car
10	Female	Car

Step 2: Split the Data Based on Gender



SR. No.	Gender	Transportation
1	Male	Bus
2	Male	Bus
5	Male	Bus
6	Male	Train
9	Male	Car



SR. No.	Gender	Transportation
3	Female	Train
4	Female	Bus
7	Female	Train
8	Female	Car
10	Female	Car

Calculating Information Gain for 'Gender'

For Males:

- Bus: $3/5 = 0.6$
- Train: $1/5 = 0.2$
- Car: $1/5 = 0.2$

$E(S_{\text{Male}}) = -0.6\log_2 0.6 - 0.2\log_2 0.2 - 0.2\log_2 0.2 \approx 1.371$

For Females:

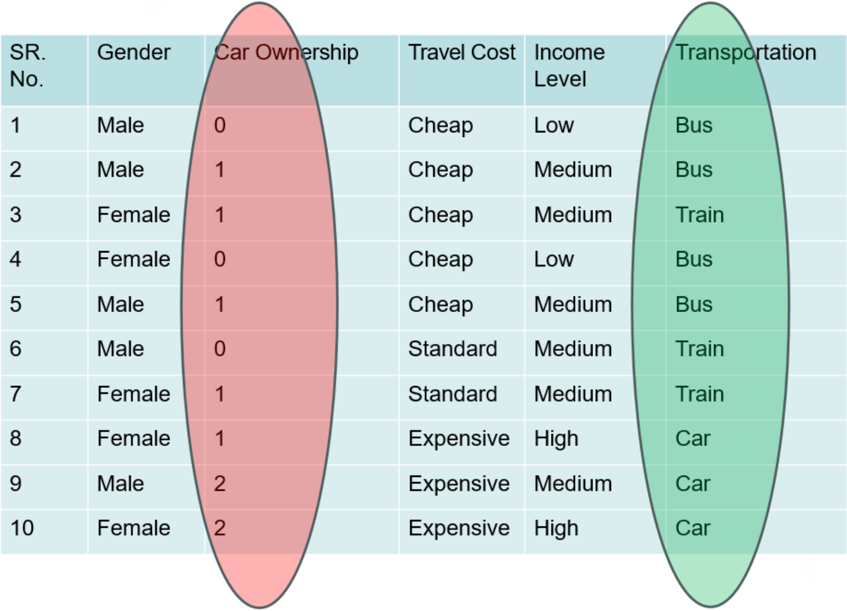
- Bus: $1/5 = 0.2$
- Train: $2/5 = 0.4$
- Car: $2/5 = 0.4$

$E(S_{\text{Female}}) = -0.2\log_2 0.2 - 0.4\log_2 0.4 - 0.4\log_2 0.4 \approx 1.522$

Information Gain for Gender

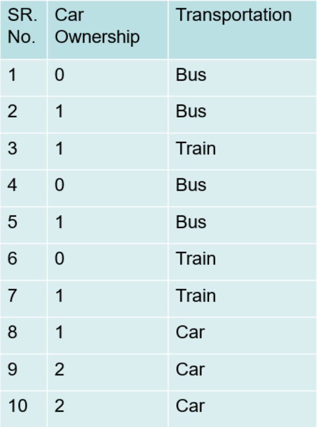
$$\begin{aligned} I(S,\text{Gender}) &= (\frac{5}{10})E(S_{\text{Male}}) + (\frac{5}{10})E(S_{\text{Female}}) \\ &= 0.5 \times 1.371 + 0.5 \times 1.522 = 1.4465 \\ \text{Gain}(S,\text{Gender}) &= E(S) - I(S,\text{Gender}) \\ &= 1.572 - 1.4465 = 0.1255 \end{aligned}$$

Step 2: Split Data by Attribute 'Car Ownership'

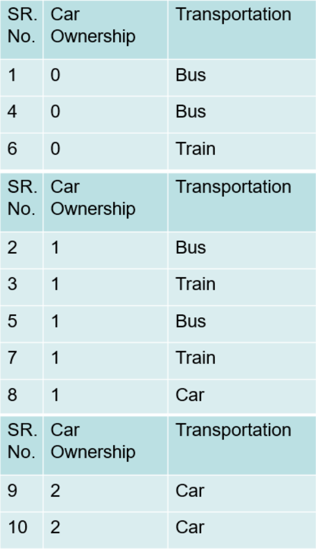


SR. No.	Gender	Car Ownership	Travel Cost	Income Level	Transportation
1	Male	0	Cheap	Low	Bus
2	Male	1	Cheap	Medium	Bus
3	Female	1	Cheap	Medium	Train
4	Female	0	Cheap	Low	Bus
5	Male	1	Cheap	Medium	Bus
6	Male	0	Standard	Medium	Train
7	Female	1	Standard	Medium	Train
8	Female	1	Expensive	High	Car
9	Male	2	Expensive	Medium	Car
10	Female	2	Expensive	High	Car

Step 2: Split Data by Attribute 'Car Ownership'



SR. No.	Car Ownership	Transportation
1	0	Bus
2	1	Bus
3	1	Train
4	0	Bus
5	1	Bus
6	0	Train
7	1	Train
8	1	Car
9	2	Car
10	2	Car



SR. No.	Car Ownership	Transportation
1	0	Bus
4	0	Bus
6	0	Train

SR. No.	Car Ownership	Transportation
2	1	Bus
3	1	Train
5	1	Bus
7	1	Train
8	1	Car

SR. No.	Car Ownership	Transportation
9	2	Car
10	2	Car

Calculating Information Gain for 'Car Ownership'

Ownership = 0:

- Bus: $2/3$
- Train: $1/3$

$E(S_0) \approx 0.918$

Ownership = 1:

- Bus: $2/5$
- Train: $2/5$
- Car: $1/5$

$E(S_1) \approx 1.522$

Ownership = 2:

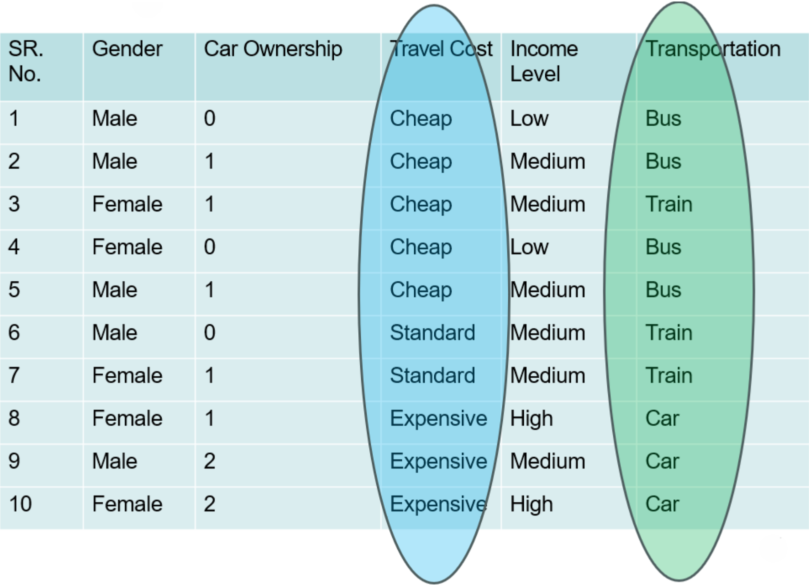
- Car: $2/2 = 1$

$E(S_2) = 0$

Information Gain for Car Ownership

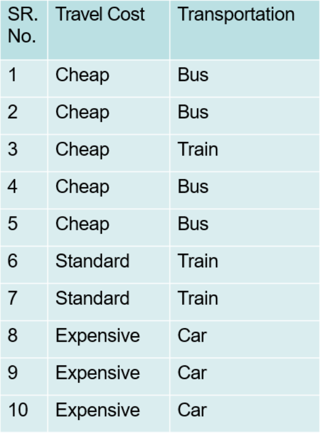
$$
\begin{aligned}
I(S, \text{Own}) &= (\frac{3}{10})E(S_0) + (\frac{5}{10})E(S_1) + (\frac{2}{10})E(S_2) \\
&= 0.3(0.918) + 0.5(1.522) + 0.2(0) = 1.0364 \\
\text{Gain}(S, \text{Own}) &= 1.572 - 1.0364 = 0.5356
\end{aligned}
$$

Step 2: Split Data by Attribute 'Travel Cost'

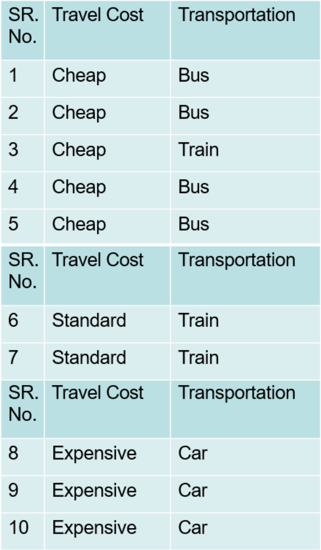


SR. No.	Gender	Car Ownership	Travel Cost	Income Level	Transportation
1	Male	0	Cheap	Low	Bus
2	Male	1	Cheap	Medium	Bus
3	Female	1	Cheap	Medium	Train
4	Female	0	Cheap	Low	Bus
5	Male	1	Cheap	Medium	Bus
6	Male	0	Standard	Medium	Train
7	Female	1	Standard	Medium	Train
8	Female	1	Expensive	High	Car
9	Male	2	Expensive	Medium	Car
10	Female	2	Expensive	High	Car

Step 2: Split Data by Attribute 'Travel Cost'



SR. No.	Travel Cost	Transportation
1	Cheap	Bus
2	Cheap	Bus
3	Cheap	Train
4	Cheap	Bus
5	Cheap	Bus
6	Standard	Train
7	Standard	Train
8	Expensive	Car
9	Expensive	Car
10	Expensive	Car



SR. No.	Travel Cost	Transportation
1	Cheap	Bus
2	Cheap	Bus
3	Cheap	Train
4	Cheap	Bus
5	Cheap	Bus

SR. No.	Travel Cost	Transportation
6	Standard	Train
7	Standard	Train

SR. No.	Travel Cost	Transportation
8	Expensive	Car
9	Expensive	Car
10	Expensive	Car

Calculating Information Gain for 'Travel Cost'

Cost = Cheap:

- Bus: $4/5 = 0.8$
- Train: $1/5 = 0.2$

$E(S_{\text{Cheap}}) \approx 0.7219$

Cost = Standard:

- Train: $2/2 = 1$

$E(S_{\text{Std}}) = 0$

Cost = Expensive:

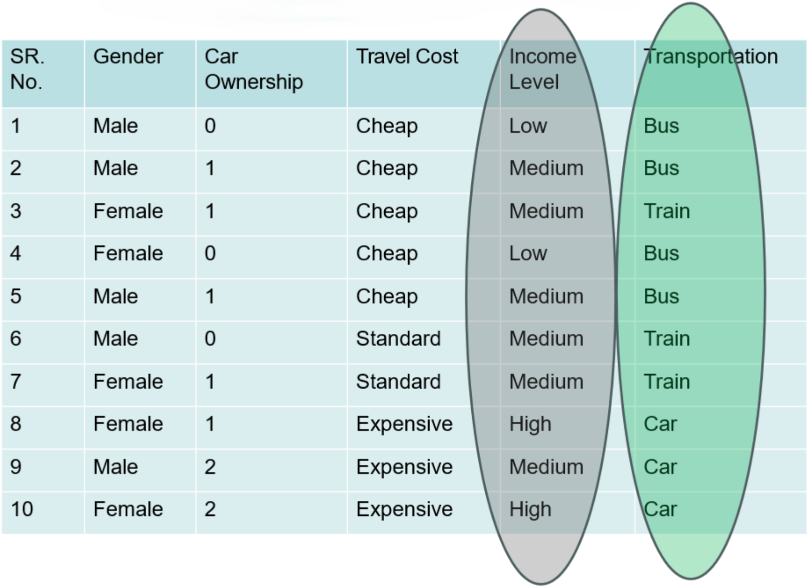
- Car: $3/3 = 1$

$E(S_{\text{Exp}}) = 0$

Information Gain for Travel Cost

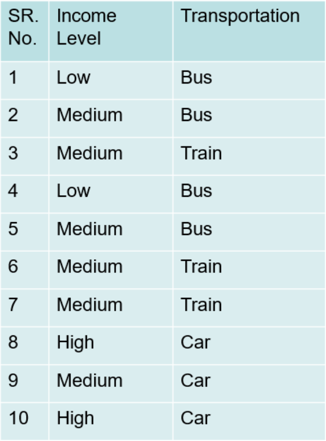
$$\begin{aligned} I(S, \text{Cost}) &= (\frac{5}{10})E(S_{\text{Cheap}}) + (\frac{2}{10})E(S_{\text{Std}}) + (\frac{3}{10})E(S_{\text{Exp}}) \\ &= 0.5(0.7219) + 0.2(0) + 0.3(0) = 0.36095 \\ \text{Gain}(S, \text{Cost}) &= 1.572 - 0.36095 = 1.211 \end{aligned}$$

Step 2: Split Data by Attribute 'Income Level'

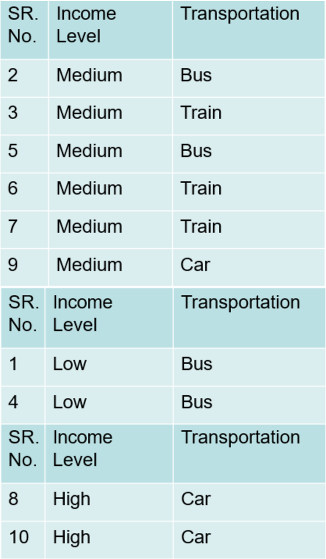


SR. No.	Gender	Car Ownership	Travel Cost	Income Level	Transportation
1	Male	0	Cheap	Low	Bus
2	Male	1	Cheap	Medium	Bus
3	Female	1	Cheap	Medium	Train
4	Female	0	Cheap	Low	Bus
5	Male	1	Cheap	Medium	Bus
6	Male	0	Standard	Medium	Train
7	Female	1	Standard	Medium	Train
8	Female	1	Expensive	High	Car
9	Male	2	Expensive	Medium	Car
10	Female	2	Expensive	High	Car

Step 2: Split Data by Attribute 'Income Level'



SR. No.	Income Level	Transportation
1	Low	Bus
2	Medium	Bus
3	Medium	Train
4	Low	Bus
5	Medium	Bus
6	Medium	Train
7	Medium	Train
8	High	Car
9	Medium	Car
10	High	Car



SR. No.	Income Level	Transportation
2	Medium	Bus
3	Medium	Train
5	Medium	Bus
6	Medium	Train
7	Medium	Train
9	Medium	Car

SR. No.	Income Level	Transportation
1	Low	Bus
4	Low	Bus

SR. No.	Income Level	Transportation
8	High	Car
10	High	Car

Calculating Information Gain for 'Income Level'

Low Income:

- Bus: $2/2 = 1$

$E(S_{\text{Low}}) = 0$

Medium Income:

- Bus: $2/6 = 0.333$
- Train: $3/6 = 0.5$
- Car: $1/6 = 0.167$

$E(S_{\text{Med}}) \approx 1.459$

High Income:

- Car: $2/2 = 1$

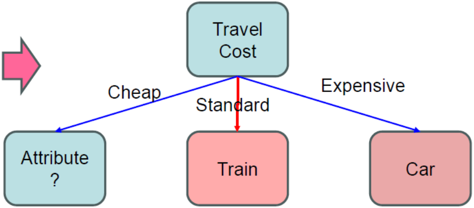
$E(S_{\text{High}}) = 0$

Information Gain for Income Level

$$\begin{aligned} I(S, \text{Income}) &= (\frac{2}{10})E(S_{\text{Low}}) + (\frac{6}{10})E(S_{\text{Med}}) + (\frac{2}{10})E(S_{\text{High}}) \\ &= 0.2(0) + 0.6(1.459) + 0.2(0) = 0.8754 \\ \text{Gain}(S, \text{Income}) &= 1.572 - 0.8754 = 0.6955 \end{aligned}$$

Selecting the Best Attribute

Attribute	Information Gain
Gender	0.125
Car Ownership	0.534
Travel Cost	1.21
Income Level	0.695



SR. No.	Gender	Car Ownership	Travel Cost	Income Level	Transportation
1	Male	0	Cheap	Low	Bus
2	Male	1	Cheap	Medium	Bus
3	Female	1	Cheap	Medium	Train
4	Female	0	Cheap	Low	Bus
5	Male	1	Cheap	Medium	Bus

Dataset for "Travel Cost = Cheap"

SR. No.	Gender	Car Ownership	Income Level	Transportation
1	Male	0	Low	Bus
2	Male	1	Medium	Bus
3	Female	1	Medium	Train
4	Female	0	Low	Bus
5	Male	1	Medium	Bus

Target Class Distribution: Bus: 4, Train: 1, Car: 0

Step 1: Calculate Initial Entropy

$$
\begin{aligned}
E(S_{\text{cheap}}) &= -\left[p_{\text{Bus}} \log_2(p_{\text{Bus}}) + p_{\text{Train}} \log_2(p_{\text{Train}})\right] \\
E(S_{\text{cheap}}) &= -\left[\left(\frac{4}{5} \log_2\left(\frac{4}{5}\right)\right) + \left(\frac{1}{5} \log_2\left(\frac{1}{5}\right)\right)\right] \\
E(S_{\text{cheap}}) &= -\left[(0.8 \times -0.3219) + (0.2 \times -2.3219)\right] \\
E(S_{\text{cheap}}) &= -\left[-0.2575 - 0.4644\right] \\
E(S_{\text{cheap}}) &= 0.7219
\end{aligned}
$$

Initial Entropy: $E(S_{\text{cheap}}) = 0.7219$

Step 2: Feature Analysis - Gender

Feature: Gender (Male: 3, Female: 2)

Male (3 instances):

- Bus: 3, Train: 0
- Entropy(Male) $= 0$

Female (2 instances):

- Bus: 1, Train: 1
- Entropy(Female) $= 1$

$$E(S_{\text{cheap}}, \text{Gender}) = \frac{3}{5} \times 0 + \frac{2}{5} \times 1 = 0.4$$
$$IG(S_{\text{cheap}}, \text{Gender}) = 0.7219 - 0.4 = 0.3219$$

Step 2: Feature Analysis - Car Ownership

Feature: Car Ownership (0: 2, 1: 3)

0 (2 instances):

- Bus: 2, Train: 0
- Entropy(0) $= 0$

1 (3 instances):

- Bus: 2, Train: 1
- Entropy(1) ≈ 0.9183

$$E(S_{\text{cheap}}, \text{Car Ownership}) = \frac{2}{5} \times 0 + \frac{3}{5} \times 0.9183 = 0.551$$

$$IG(S_{\text{cheap}}, \text{Car Ownership}) = 0.7219 - 0.551 = 0.1709$$

Step 2: Feature Analysis - Income Level

Feature: Income Level (Low: 2, Medium: 3)

Low (2 instances):

- Bus: 2, Train: 0
- Entropy(Low) $= 0$

Medium (3 instances):

- Bus: 2, Train: 1
- Entropy(Medium) ≈ 0.9183

$$E(S_{\text{cheap}}, \text{Income Level}) = \frac{2}{5} \times 0 + \frac{3}{5} \times 0.9183 = 0.551$$

$$IG(S_{\text{cheap}}, \text{Income Level}) = 0.7219 - 0.551 = 0.1709$$

Step 3: Compare Information Gains

Feature	Information Gain
Gender	**0.3219**
Car Ownership	0.1709
Income Level	0.1709

Decision

Gender has the highest Information Gain for the "Cheap" branch.
It becomes the decision node for this branch.

Step 4: Build Sub-tree for "Travel Cost = Cheap"

Node: Gender

Branch: Male

- 3 instances
- Bus: 3, Train: 0
- **Pure Node** → Transportation = Bus

Branch: Female

- 2 instances
- Bus: 1, Train: 1
- **Not pure** - needs further splitting

Step 5: "Travel Cost = Cheap" and "Gender = Female"

Filtered Dataset:

SR. No.	Gender	Car Ownership	Income Level	Transportation
3	Female	1	Medium	Train
4	Female	0	Low	Bus

Target Class Distribution: Bus: 1, Train: 1

$$E(S_{\text{cheap, female}}) = -\left[\left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right) + \left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right)\right] = 1$$

Step 5.2: Information Gain for Remaining Features

Car Ownership

- 0: Bus: 1 → Entropy $= 0$
- 1: Train: 1 → Entropy $= 0$

$$E = \frac{1}{2} \times 0 + \frac{1}{2} \times 0 = 0$$
$$IG = 1 - 0 = 1$$

Income Level

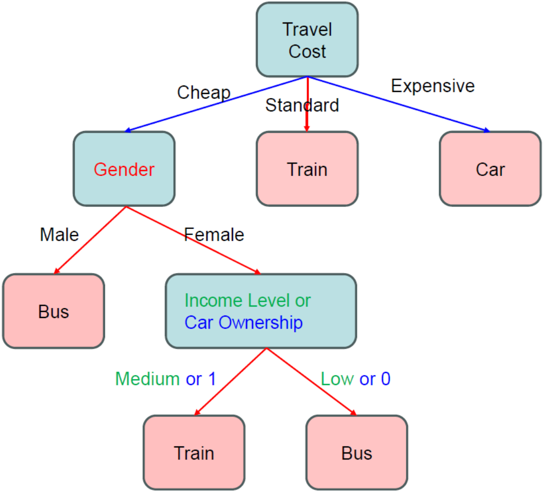
- Low: Bus: 1 → Entropy $= 0$
- Medium: Train: 1 → Entropy $= 0$

$$E = \frac{1}{2} \times 0 + \frac{1}{2} \times 0 = 0$$
$$IG = 1 - 0 = 1$$

Decision

Both features have maximum gain (IG $= 1$).
Arbitrarily choose **Car Ownership**.

Resulting Decision Tree



Prediction on New Data

SR. No.	Gender	Car Ownership	Travel Cost	Income Level	Predicted Transport
1	Male	1	Standard	High	Train
2	Male	0	Cheap	Medium	Bus
3	Female	1	Cheap	High	Train

Limitations of ID3 Algorithm:

- Tends to overfit small datasets.
- Tests only one attribute at a time for splitting.
- Classifying continuous data can be computationally expensive.
- Cannot handle missing values effectively.
- Does not support numeric attributes directly.

Summary

- Initial entropy for "Cheap" subset: 0.7219
- Gender provided highest information gain: 0.3219
- Male branch is pure: Transportation = Bus
- Female branch required further splitting
- Car Ownership chosen for female branch (IG = 1)
- Final tree provides complete classification

Key Insight

The ID3 algorithm successfully partitions the data,
creating a decision tree that minimizes entropy at each step.

Classification Metrics: The Confusion Matrix

	Predicted Positive	**Predicted Negative**
Actual Positive (P)	True Positive (TP)	False Negative (FN)
Actual Negative (N)	False Positive (FP)	True Negative (TN)

True Positive Rate (TPR) (Sensitivity, Recall, Hit Rate)

$$\text{TPR} = \frac{\text{TP}}{\text{Actual Positive (P)}} = 1 - \text{FNR}$$

False Positive Rate (FPR) (Fall-out, False Alarm)

$$\text{FPR} = \frac{\text{FP}}{\text{Actual Negative (N)}} = 1 - \text{TNR}$$

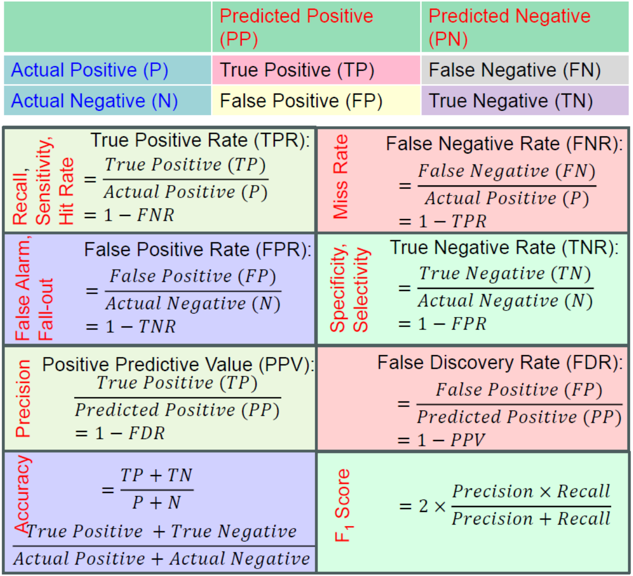
False Negative Rate (FNR) (Miss Rate)

$$\text{FNR} = \frac{\text{FN}}{\text{Actual Positive (P)}} = 1 - \text{TPR}$$

True Negative Rate (TNR) (Specificity, Selectivity)

$$\text{TNR} = \frac{\text{TN}}{\text{Actual Negative (N)}} = 1 - \text{FPR}$$

Classification Metrics



	Predicted Positive (PP)	Predicted Negative (PN)
Actual Positive (P)	True Positive (TP)	False Negative (FN)
Actual Negative (N)	False Positive (FP)	True Negative (TN)

Recall, Sensitivity, Hit Rate	True Positive Rate (TPR): $= \frac{True\ Positive\ (TP)}{Actual\ Positive\ (P)}$ $= 1 - FNR$	Miss Rate	False Negative Rate (FNR): $= \frac{False\ Negative\ (FN)}{Actual\ Positive\ (P)}$ $= 1 - TPR$
False Alarm, Fall-out	False Positive Rate (FPR): $= \frac{False\ Positive\ (FP)}{Actual\ Negative\ (N)}$ $= 1 - TNR$	Specificity, Selectivity	True Negative Rate (TNR): $= \frac{True\ Negative\ (TN)}{Actual\ Negative\ (N)}$ $= 1 - FPR$
Precision	Positive Predictive Value (PPV): $\frac{True\ Positive\ (TP)}{Predicted\ Positive\ (PP)}$ $= 1 - FDR$		False Discovery Rate (FDR): $= \frac{False\ Positive\ (FP)}{Predicted\ Positive\ (PP)}$ $= 1 - PPV$
Accuracy	$= \frac{TP + TN}{P + N}$ $\frac{True\ Positive\ + True\ Negative}{Actual\ Positive + Actual\ Negative}$	F_1 Score	$= 2 \times \frac{Precision \times Recall}{Precision + Recall}$

Classification Metrics

Accuracy

Accuracy is the regularly used metric. Accuracy answers the question "Out of all the predictions we made, how many were true?"

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{True Negatives} + \text{False Positives} + \text{False Negatives}}$$

Precision

Precision is a metric that indicates the proportion of true positives among the total positives predicted by the model. It answers the question "Out of all the positive predictions we made, how many were true?"

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Minimizes False Positives

Recall

Recall represents how well the model performs in identifying all the positives. Recall is also called the true positive rate. It answers the question "Out of all the data points that should be predicted as true, how many did we correctly predict as true?"

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

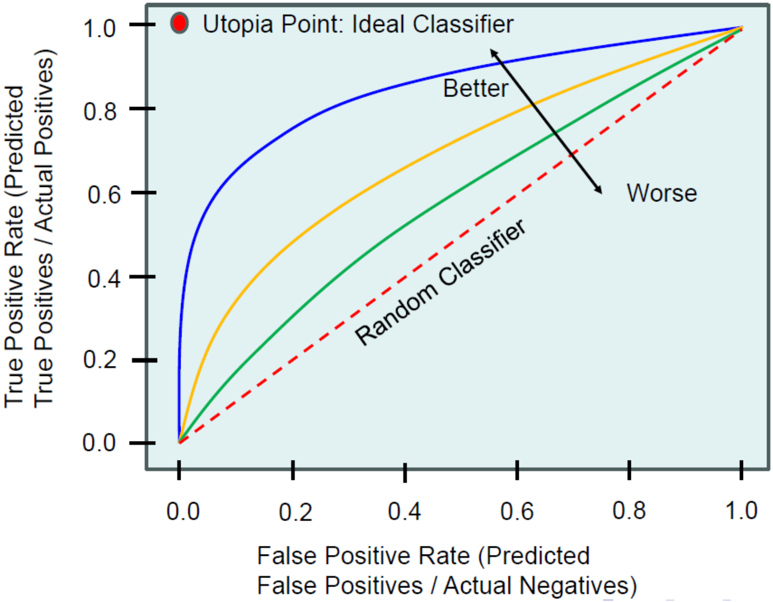
Minimizes False Negatives

F1 Score

F1 Score is a measure that combines recall and precision. As we have seen, there is a trade-off between precision and recall; F1 can therefore be used to measure how effectively our models make that trade-off. F1 score → zero if any of the components (precision or recall) → zero. Thereby, it penalizes extreme values of either component.

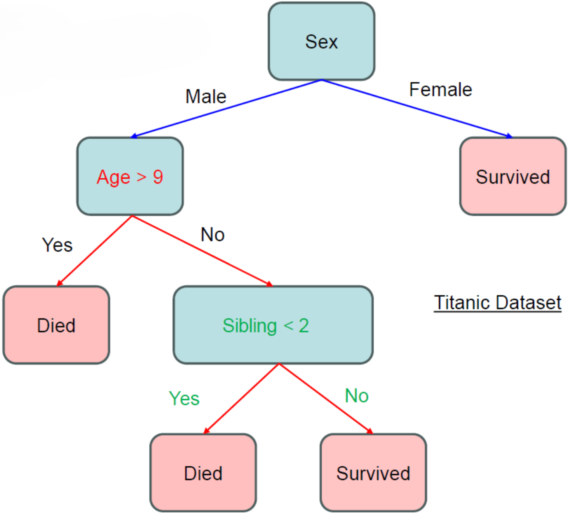
$$F_1 \text{ Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Receiver Operating Characteristic (ROC) Curve



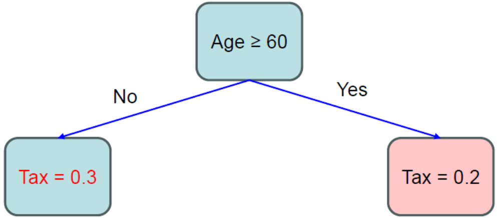
Decision Trees for Regression

Regression can be used for noncategorical data.



Decision Trees for Regression

- It is also possible to use decision trees to produce value

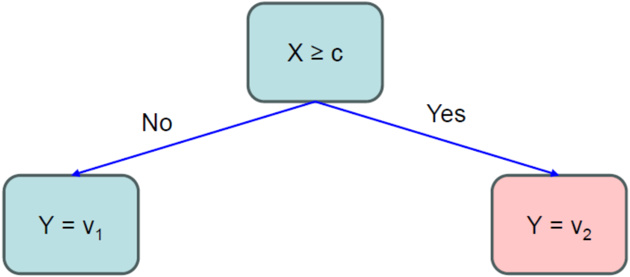


The question is, how do we learn the decision boundaries, such as 60 in this example?

1. The problem is more complex than learning decision trees for classification using categorical data.
2. A heuristic algorithm is required here.
3. As with ID3, the algorithm is recursive. Therefore, if we know how to split one node, we simply continue recursively.

Decision Trees for Regression

1. Since we are trying to do regression, we should use mean squared error.
2. Suppose that our tree is trying to predict the value Y from the variable X.

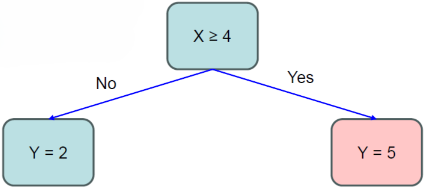


What are the correct values of v_1, v_2, and c?

Simple Example

Suppose that we have the following data set:

X	y
1	2
2	5
3	4
7	3
8	4

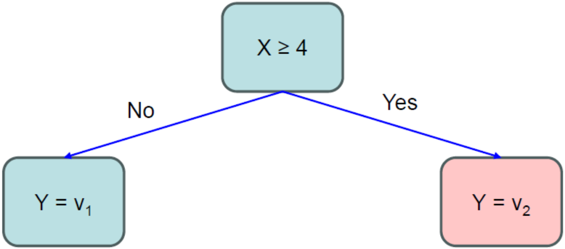


Running through our example we need to separate our data into two sets $x < 4$ with predicted value $y = 2$ and $x \geq 4$ with predicted value $y = 5$. So, we can calculate the squared error.

1. $x < 4$ is the set $\{(1, 2), (2, 5), (3, 4)\}$ gives the error: $(2-2)^2 + (5-2)^2 + (4-2)^2$ equals $0 + 9 + 4 = 13$.
2. $x \geq 4$ is the set $\{(7, 3), (8, 4)\}$ gives the error $(3-5)^2 + (4-5)^2$ equals $4 + 1 = 5$.

So, the **total mean squared error** is: $1/5 \times (13 + 5) = 3.6$.

Simple Example



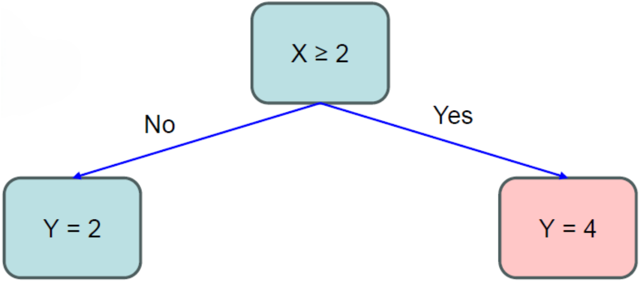
Finding v_1 and v_2

Suppose we are given c, then what are the sensible choices for v_1 and v_2?

1. We have the two sets: $\{(1, 2), (2, 5), (3, 4)\}$ and $\{(7, 3), (8, 4)\}$. We want to minimize the resulting mean squared error. Picking the average value in each set minimizes the mean squared error.
   - $v_1 = (2 + 5 + 4)/3 = 3.67$
   - $v_2 = (3 + 4)/2 = 3.5$
2. If you run through the calculations, you'll get an error of about 1.033

Finding c

- Given c, we now know how to calculate the best values of v_1 and v_2.
- To find the best value of c, we simply sort our data set by error, go through all the possible values of c for our data set, and pick the one that gives the minimum error.
- If you run through the calculations, you'll get this tree:



Dataset Overview

Table: Car Pricing Dataset

Car	Age (years)	Mileage (k miles)	Price ($)
1	2	25	22,000
2	5	60	14,500
3	1	15	26,000
4	7	85	11,000
5	3	35	19,500
6	4	50	16,000

Goal: Predict ‘Price‘ using ‘Age‘ and ‘Mileage‘.

Method: Find splits that maximize the reduction in Sum of Squared Errors (SSE).

- $SSE = \sum_{i=1}^{n}(y_i - \bar{y})^2$
- $SSE_{reduction} = SSE_{parent} - (SSE_{left} + SSE_{right})$

Step 1: The Root Node

Calculate Initial State (All Data)

- **Prices:** $\{22\text{k}, 14.5\text{k}, 26\text{k}, 11\text{k}, 19.5\text{k}, 16\text{k}\}$
- **Mean Price ($\bar{y}_{root}$):** $\frac{109{,}000}{6} =$ **\$18,166.67**
- This is the prediction if we don't make any splits.

Step 1: The Root Node

Calculate Initial State (All Data)

- **Prices:** {22k, 14.5k, 26k, 11k, 19.5k, 16k}
- **Mean Price ($\bar{y}_{root}$):** $\frac{109{,}000}{6} =$ **\$18,166.67**
- This is the prediction if we don't make any splits.

Calculate Initial SSE (SSE_{parent})

- $SSE_{root} = (22000 - 18167)^2 + ... + (16000 - 18167)^2$
- $SSE_{root} =$ **147,333,333.33**

Our goal is to find a split that reduces this large error value as much as possible.

Step 2.1: Finding the Best Split (Evaluating 'Age')

Sorted Unique 'Age' Values: $\{1, 2, 3, 4, 5, 7\}$
Possible Split Points: $\{1.5, 2.5, 3.5, 4.5, 6.0\}$

Split Rule	Left SSE	Right SSE	SSE Reduction
Age $<$ 1.5	Car 3- Mean=26000, SSE=0	Cars 1,2,4,5,6- Mean=16600, SSE=73,600,000	73.73 M
Age $<$ 2.5	Cars 3,1-Mean=24000, SSE= 8,000,000	Cars 2,4,5,6 - Mean=15250, SSE=37,250,000	102.08 M
Age $<$ 3.5	Cars 3,1,5-Mean=13833.33, SSE=**21,500,000**	Cars 2,4,6-Mean=13833.33, SSE=**13,166,667**	**112.66 M**
Age $<$ 4.5	Cars 3,1,5,6- Mean=20875,SSE=53,750,000	Cars 2,4-6,Mean=12750, SSE=125,000	87.46 M
Age $<$ 6.0	Cars 3,1,5,6,2-Mean=19600,SSE=73,600,000	Car 4-Mean=11000, SSE=0	73.73 M

The best split for the 'Age' feature is **Age $<$ 3.5**.

Step 2.2: Finding the Best Split (Evaluating 'Mileage')

Sorted Unique 'Mileage' Values: {15, 25, 35, 50, 60, 85}
Possible Split Points: {20, 30, 42.5, 55, 72.5}

Table: Detailed evaluation of splits on the 'Mileage' feature

Split Rule	Left Node (SSE)	Right Node (SSE)	SSE Reduction
Mileage $<$ 20	Car 3 — Mean=26000, SSE=0	Cars 1,2,4,5,6 — Mean=16600, SSE=73,600,000	73.73 M
Mileage $<$ 30	Cars 3,1 — Mean=24000, SSE=8,000,000	Cars 2,4,5,6 — Mean=15250, SSE=37,250,000	102.08 M
Mileage $<$ 42.5	**Cars 3,1,5 — Mean=22500, SSE=21,500,000**	**Cars 2,4,6 — Mean=13833, SSE=13,166,667**	**112.66 M**
Mileage $<$ 55	Cars 3,1,5,6 — Mean=20875, SSE=53,750,000	Cars 2,4 — Mean=12750, SSE=6,125,000	87.46 M
Mileage $<$ 72.5	Cars 3,1,5,6,2 — Mean=19600, SSE=73,600,000	Car 4 — Mean=11000, SSE=0	73.73 M

Just like the 'Age' feature, the best split for 'Mileage' is the one that groups Cars {1, 3, 5} and {2, 4, 6}, which is **Mileage $<$ 42.5**. This yields the maximum SSE reduction of **112.66 M**.

Step 2.3: Decision for the First Split

- The max SSE Reduction from 'Age' is **112.66 M** (for split ‘Age $<$ 3.5‘).
- The max SSE Reduction from 'Mileage' is **112.66 M** (for split ‘Mileage $<$ 42.5‘).

Both splits result in the same grouping of data and the same maximum reduction. We can choose either. Let's pick **Age** $<$ **3.5**.

This creates two new nodes:

Left Node (Node 1)

- Condition: Age $<$ 3.5
- Data: $\{$Car 1, 3, 5$\}$
- Prediction: $22,500

Right Node (Node 2)

- Condition: Age $\geq$ 3.5
- Data: $\{$Car 2, 4, 6$\}$
- Prediction: $13,833

Step 3: Recursively Splitting the Children

We repeat the process for each new node.

For Left Node (Prediction $22,500):

- Data: {Car 1 (Age 2), Car 3 (Age 1), Car 5 (Age 3)}
- Best Split: **Age** $<$ **1.5** (SSE Reduction $=$ 18.375 M)

For Right Node (Prediction $13,833):

- Data: {Car 2 (Age 5), Car 4 (Age 7), Car 6 (Age 4)}
- Best Split: **Age** $<$ **6.0** (SSE Reduction $=$ 12.04 M)

This process continues until a stopping condition is met (e.g., all data points in a leaf are identical, which is what will happen here).

Step 3.1: Splitting the Left Child Node (Age < 3.5)

Node Data: {Car 1, Car 3, Car 5}
Parent SSE for this node: 21,500,000

Table: Evaluation of Splits for 'Age' in Node 1

Split Rule	Left Data	Right Data	SSE Reduction
Age $<$ 1.5	{Car 3}	{Car 1, Car 5}	**18,375,000**
Age $<$ 2.5	{Car 3, Car 1}	{Car 5}	13,500,000

Table: Evaluation of Splits for 'Mileage' in Node 1

Split Rule	Left Data	Right Data	SSE Reduction
Mileage $<$ 20	{Car 3}	{Car 1, Car 5}	**18,375,000**
Mileage $<$ 30	{Car 3, Car 1}	{Car 5}	13,500,000

Decision for Node 1: The best split is **Age $<$ 1.5**.

Step 3.2: Splitting the Right Child Node (Age >= 3.5)

Node Data: {Car 2, Car 4, Car 6}
Parent SSE for this node: 13,166,667

Table: Evaluation of Splits for 'Age' in Node 2

Split Rule	Left Data	Right Data	SSE Reduction
Age $<$ 4.5	{Car 6}	{Car 2, Car 4}	7,041,667
Age $<$ 6.0	{Car 6, Car 2}	{Car 4}	**12,041,667**

Table: Evaluation of Splits for 'Mileage' in Node 2

Split Rule	Left Data	Right Data	SSE Reduction
Mileage $<$ 55	{Car 6}	{Car 2, Car 4}	7,041,667
Mileage $<$ 72.5	{Car 6, Car 2}	{Car 4}	**12,041,667**

Decision for Node 2: The best split is **Age $<$ 6.0**.

Ensemble Learning

1. Decision trees are interesting. However, they lead to overfitting as they can learn anything.
2. Small changes in the data set can give you different trees because they can learn anything and therefore tend to overfit your data set.
3. Basic concept of **ensemble learning** is to combine multiple weak learners and take majority votes.
4. Various techniques, such as boosting and bagging, can extract more statistical mileage from your data.
5. At a 1906 country fair in Plymouth (UK), 800 people participated in a contest to estimate the weight of a slaughtered and dressed ox. Statistician Francis Galton observed that the median guess, 1207 pounds, was accurate within 1% of the true weight of 1198 pounds.
6. General idea is to take lots of incorrect observations. On average, the errors cancel out.
7. In Hardware design, run three devices in parallel and take a majority vote on a decision or an action (e.g., take three temperature sensors for monitoring the temperature).

Ensemble Learning

1. If a single device has probability p of failing, then the whole system fails if 2 or 3 out of the 3 devices fail. Therefore, the following failure scenarios (strike-through represents failure) are obtained:

$$\cancel{1\;2\;3} = p^3$$
$$1\;\cancel{2}\;\cancel{3} = (1-p)p^2$$
$$\cancel{1}\;\cancel{2}\;3 = p(1-p)p$$
$$\cancel{1}\;2\;\cancel{3} = p^2(1-p)$$

2. Thus, the total probability of failure is $p^3 + 3p^2(1-p)$.
3. Even if you have a system with a probability of failure of say 0.2, you could reduce this to about 0.104.
4. This concept can be used intelligently. Suppose you have independent classifiers that are slightly better than random guessing (weak learners). If you have enough of them, then the probability of making a wrong decision is when a majority of them get the wrong answer.

Weak Learners

1. Suppose there are 25 independent classifiers, each with an error rate $\epsilon = 0.35$.
2. Then the probability that the whole ensemble makes a wrong prediction is given as follows.

$$P(X \geq 13) = \sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1-\epsilon)^{25-i} = 0.06$$

   The ϵ^i term is the probability of i incorrect predictions. The $(1-\epsilon)^{25-i}$ is the probability of 25-i correct predictions. The $\binom{25}{i}$ term gives the number of ways of choosing a subset of size i from 25 items. You need at least 13 of the classifiers to be wrong. You sum of all the terms.
3. Thus, we have transformed a probability of $1 - 0.35 = 0.65$ of being correct into a probability of $1 - 0.06 = 0.94$ of being correct.
4. The analysis only works if the weak learners are independent of each other. They can make mistakes, but the mistakes that they make must be independently correlated.
5. The techniques, such as Random Forest, Gradient Boosting, and XGBoost, follow the above concept.

What is Ensemble Learning?

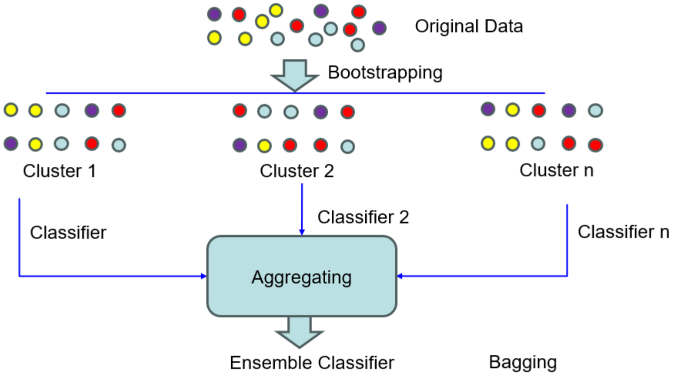
- **Ensemble Learning**: Combining multiple models to improve performance
- **Key Principle**: "Wisdom of the crowd" - multiple weak learners can create a strong learner
- **Main Types**:
  - Bagging (Bootstrap Aggregating)
  - Boosting
  - Stacking

Why Ensembles Work?

$$\text{Error} = \text{Bias}^2 + \text{Variance} + \text{Noise}$$

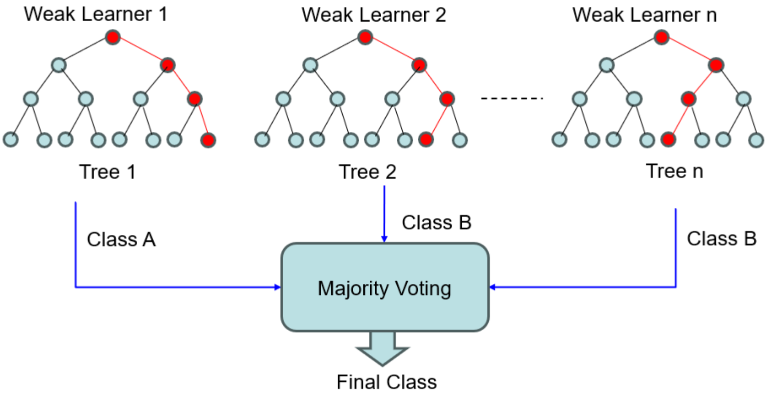
Ensembles primarily reduce variance without increasing bias

Bootstrap Aggregating: Bagging



1. A way of sampling the original data set to make several statistically independent data sets.
2. You sample with replacement. You can use the same item twice in each subset.
3. This provides statistically similar but independent samples of the same size, which are generated from your original dataset.

Random Forests



1. Build weak learners by picking random subsets of the features. The learners will be weak, because you won't use all the features. The learners will be independent because they work on different features.
2. Often works very well in practice. Lots of hyperparameters to tune, the number of trees, and the number of features in each tree.

Bagging Numerical Problem Setup

Training Data

$D = \{(1,0),(2,1),(3,0),(4,1)\}$
Binary classification: $y \in \{0,1\}$

Bootstrap Samples (B=3)

- $D_1 = \{(1,0),(2,1),(3,0),(2,1)\}$
- $D_2 = \{(4,1),(1,0),(4,1),(3,0)\}$
- $D_3 = \{(3,0),(1,0),(4,1),(1,0)\}$

Bagging: Individual Classifiers

Classifier h_1:

- Data: (1,0), (2,1), (3,0), (2,1)
- $h_1(x) = \begin{cases} 0 & \text{if } x \leq 1.5 \\ 1 & \text{otherwise} \end{cases}$

Classifier h_2:

- Data: (4,1), (1,0), (4,1), (3,0)
- $h_2(x) = \begin{cases} 0 & \text{if } x \leq 3.5 \\ 1 & \text{otherwise} \end{cases}$

Classifier h_3:

- Data: (3,0), (1,0), (4,1), (1,0)
- $h_3(x) = \begin{cases} 0 & \text{if } x \leq 3.5 \\ 1 & \text{otherwise} \end{cases}$

Bagging: Predictions and Results

Test Point: x = 2.5

- $h_1(2.5)$: $2.5 > 1.5 \rightarrow \mathbf{1}$
- $h_2(2.5)$: $2.5 \leq 3.5 \rightarrow \mathbf{0}$
- $h_3(2.5)$: $2.5 \leq 3.5 \rightarrow \mathbf{0}$

Majority voting: $\{1, 0, 0\} \rightarrow \mathbf{0}$

Random Forest: Enhanced Bagging

Key Innovation

Random Forest = Bagging + Random Feature Selection

- Each tree uses random bootstrap sample **AND**
- Each split considers random subset of features

Our Dataset with Multiple Features

Each instance has 3 features:

- Instance 1: (1,5,2), y=0
- Instance 2: (2,3,4), y=1
- Instance 3: (3,1,6), y=0
- Instance 4: (4,4,1), y=1

Random Forest: Bootstrap with Feature Randomness

Tree h_1 on D_1:

- Instances: (1,5,2,0), (3,1,6,0), (3,1,6,0), (4,4,4,1)
- Features: {1,2}

Tree h_2 on D_2:

- Instances: (4,4,1,1), (1,5,2,0), (4,4,1,1), (3,1,6,0)
- Features: {2,3}

Tree h_3 on D_3:

- Instances: (3,1,6,0), (1,5,2,0), (4,4,1,1), (1,5,2,0)
- Features: {1,3}

Double Randomness

Each tree sees different data **AND** different features → Maximum diversity!

Random Forest: Tree Construction

Tree h_1 (Features 1,2):

- Best split: Feature 1 ≤ 3.5
- $h_1(x) = \begin{cases} 0 & \text{if } x_1 \leq 3.5 \\ 1 & \text{otherwise} \end{cases}$

Tree h_2 (Features 2,3):

- Best split: Feature 3$\leq$1.5
- $h_2(x) = \begin{cases} 0 & \text{if } x_3 \leq 1.5 \\ 1 & \text{otherwise} \end{cases}$

Tree h_3 (Features 1,3):

- Best split: Feature 1 ≤ 3.5
- $h_3(x) = \begin{cases} 0 & \text{if } x_1 \leq 3.5 \\ 1 & \text{otherwise} \end{cases}$

Key Insight

Different trees use different features for splitting $\rightarrow$ More diverse ensemble

Random Forest: Predictions and Feature Importance

Test Point: x = (2.5, 3.5, 3.0)

- $h_1(2.5, 3.5, 3.0)$: Uses Features 1,2 → $x_1 = 2.5 \leq 3.5 \rightarrow \mathbf{0}$
- $h_2(2.5, 3.5, 3.0)$: Uses Features 2,3 → $x_3 = 3 > 1.5 \rightarrow \mathbf{1}$
- $h_3(2.5, 3.5, 3.0)$: Uses Features 1,3 → $x_1 = 2.5 \leq 3.5 \rightarrow \mathbf{0}$

Majority voting: $\{0, 1, 0\} \rightarrow \mathbf{0}$

Key Advantages of Random Forest

- **Higher Diversity**: Double randomness creates more varied trees
- **Better Variance Reduction**: Less correlated predictions
- **Feature Importance**: Built-in feature selection
- **Robustness**: Handles noisy features better

Boosting

1. In Bagging, we train everything in parallel - in Boosting, we train a sequence of weak learners. In this, the weak learners learn from the mistakes that previous weak learners made using some weighting scheme.
2. Initially, we give each data sample in our training set an equal weight. Then repeat the following steps:
    1. Train a new weak learner on a weighted dataset in adaptive boosting. Note that weight represents the relative importance of the given data sample in the entire data set. Thus, data samples with higher weights are preferentially selected when sampling the entire dataset to construct the new dataset for training subsequent weak learners.
    2. Work out where the weak learner misclassifies and update the weights of those training samples.
    3. Combine all the weak learners that were trained.
3. In gradient boosting, weak learners are successively trained on pseudo-residual errors from previous weak learners using the concept of gradient descent.
4. Again, we train things sequentially but try to pick the next weak classifier that improves the overall training error. Extreme Gradient Boosting, or XGBoost, is an efficient parallel version of gradient boosting that works well on large datasets and with missing values. In this, extreme pruning of the decision tree is performed by using L_1 or L_2 regularization.

AdaBoost Algorithm Overview

Key Steps

1. Initialize sample weights equally
2. For each round $t = 1$ to T:
    - Find best weak classifier h_t with minimum error ϵ_t
    - Compute classifier weight $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
    - Update sample weights: $w_i^{(t+1)} = w_i^{(t)} \exp(-\alpha_t y_i h_t(x_i))$
    - Normalize weights
3. Final classifier: $H(x) = \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right)$

Dataset Overview

Sample	Gender	Age	Income	Illness	Initial Weight
1	Male	41	40000	Yes	0.2
2	Male	54	30000	No	0.2
3	Female	42	25000	No	0.2
4	Female	40	60000	Yes	0.2
5	Male	46	50000	Yes	0.2

Encoding

- Target: $+1$ for Illness "Yes", -1 for Illness "No"
- Initial weights: $w_i^{(1)} = \frac{1}{5} = 0.2$ for all samples

Detailed Weight Calculations: Iteration 1

Weak Learner 1: Gender-based Rule

Rule: If Gender = Male → +1, else → -1

Sample	Gender	True y	$h_1(x)$	Correct?	$w_i^{(1)}$	Error	Update Factor	$w_i^{(2)}$
1	M	+1	+1	✓	0.2000	0.0000	$e^{-0.2027} = 0.8167$	0.1667
2	M	-1	+1	✗	0.2000	0.2000	$e^{+0.2027} = 1.2247$	0.2500
3	F	-1	-1	✓	0.2000	0.0000	$e^{-0.2027} = 0.8167$	0.1667
4	F	+1	-1	✗	0.2000	0.2000	$e^{+0.2027} = 1.2247$	0.2500
5	M	+1	+1	✓	0.2000	0.0000	$e^{-0.2027} = 0.8167$	0.1667

Calculations

- Total Error: $\epsilon_1 = 0.2 + 0.2 = 0.4$
- Classifier Weight: $\alpha_1 = \frac{1}{2}\ln(\frac{0.6}{0.4}) \approx 0.2027$
- Normalization Factor: 4.8998
- Misclassified samples (2,4) get higher weights

Weight Calculations: Iteration 2

Weak Learner 2: Age-based Rule

Rule: If Age $< 45 \rightarrow +1$, else $\rightarrow$ -1

Sample	Age	True y	$h_2(x)$	Correct?	$w_i^{(2)}$	Error	Update Factor	$w_i^{(3)}$
1	41	$+1$	$+1$	✓	0.1667	0.0000	$e^{-0.3436}$	0.1250
2	54	-1	-1	✓	0.2500	0.0000	$e^{-0.3436}$	0.1875
3	42	-1	$+1$	✗	0.1667	0.1667	$e^{+0.3436}$	0.2500
4	40	$+1$	$+1$	✓	0.2500	0.0000	$e^{-0.3436}$	0.1875
5	46	$+1$	-1	✗	0.1667	0.1667	$e^{+0.3436}$	0.2500

Calculations

- Total Error: $\epsilon_2 = 0.1667 + 0.1667 = 0.3333$
- Classifier Weight: $\alpha_2 = \frac{1}{2} \ln(\frac{0.6667}{0.3333}) \approx 0.3466$
- Samples 3 and 5 (misclassified) get weight increases

Weight Calculations: Iteration 3

Weak Learner 3: Income-based Rule

Rule: If Income $>$ 45000 $\rightarrow$ $+1$, else $\rightarrow$ -1

Sample	Income	True y	$h_3(x)$	Correct?	$w_i^{(3)}$	Error	Update Factor	$w_i^{(4)}$
1	40000	$+1$	-1	✗	0.1250	0.1250	$e^{0.9726}$	0.2667
2	30000	-1	-1	✓	0.1875	0.0000	$e^{-0.9726}$	0.1333
3	25000	-1	-1	✓	0.2500	0.0000	$e^{-0.9726}$	0.2000
4	60000	$+1$	$+1$	✓	0.1875	0.0000	$e^{-0.9726}$	0.1333
5	50000	$+1$	$+1$	✓	0.2500	0.0000	$e^{-0.9726}$	0.2667

Calculations

- Total Error: $\epsilon_3 = 0.1250$
- Classifier Weight: $\alpha_3 = \frac{1}{2}\ln(\frac{0.875}{0.125}) \approx 0.9726$
- Sample 1 gets highest final weight (consistently hard to classify)

Complete Weight Evolution

Sample	Initial	After Iter 1	After Iter 2	Final	Change
1	0.2000	0.1667	0.1250	0.2667	+33.4%
2	0.2000	0.2500	0.1875	0.1333	-33.4%
3	0.2000	0.1667	0.2500	0.2000	0.0%
4	0.2000	0.2500	0.1875	0.1333	-33.4%
5	0.2000	0.1667	0.2500	0.2667	+33.4%

Key Observations

- **Samples 1 & 5**: Hardest to classify → Highest final weights (0.2667)
- **Samples 2 & 4**: Easier to classify → Lowest final weights (0.1333)
- **Sample 3**: Medium difficulty → Medium weight (0.2000)
- AdaBoost successfully identifies and focuses on difficult samples

Complete Classification Process

Final Combined Classifier

$$H(x) = \text{sign}(0.2027 \cdot h_1(x) + 0.3466 \cdot h_2(x) + 0.9726 \cdot h_3(x))$$

Weak Learner	**Rule**	**Error** ϵ_m	**Weight** α_m
$h_1(x)$	If Gender $=$ Male $\rightarrow +1$, else -1	0.4000	0.2027
$h_2(x)$	If Age $< 45 \rightarrow +1$, else -1	0.3333	0.3466
$h_3(x)$	If Income $> 45000 \rightarrow +1$, else -1	0.1250	0.9726

Detailed Prediction Calculations

Sample	Features	True y	$h_1(x)$	$h_2(x)$	$h_3(x)$	Weighted Sum	Final Pred	Correct?
1	Male, 41, 40k	$+1$	$+1$	$+1$	-1	$0.2027 + 0.3466 - 0.9726 = -0.4233$	-1	✗
2	Male, 54, 30k	-1	$+1$	-1	-1	$0.2027 - 0.3466 - 0.9726 = -1.1165$	-1	✓
3	Female, 42, 25k	-1	-1	$+1$	-1	$-0.2027 + 0.3466 - 0.9726 = -0.8287$	-1	✓
4	Female, 40, 60k	$+1$	-1	$+1$	$+1$	$-0.2027 + 0.3466 + 0.9726 = +1.1165$	$+1$	✓
5	Male, 46, 50k	$+1$	$+1$	-1	$+1$	$0.2027 - 0.3466 + 0.9726 = +0.8287$	$+1$	✓

Sample 1 Detailed Calculation

- h_1: Male $\rightarrow +1 \rightarrow +1 \times 0.2027 = +0.2027$
- h_2: Age $41 < 45 \rightarrow +1 \rightarrow +1 \times 0.3466 = +0.3466$
- h_3: Income $40k < 45k \rightarrow$ -1 $\rightarrow -1 \times 0.9726 = -0.9726$
- **Sum**: $+0.2027 + 0.3466 - 0.9726 = -0.4233 \rightarrow$ Prediction: -1 ✗

Complete Dataset After Boosting

Sample	Gender	Age	Income	True Illness	Initial Weight	Final Weight	Prediction	Corre
1	Male	41	40000	Yes	0.2000	0.2667	No	✗
2	Male	54	30000	No	0.2000	0.1333	No	✓
3	Female	42	25000	No	0.2000	0.2000	No	✓
4	Female	40	60000	Yes	0.2000	0.1333	Yes	✓
5	Male	46	50000	Yes	0.2000	0.2667	Yes	✓

Key Insights

- **AdaBoost Success**: Combined weak classifiers (60-75% accuracy) into strong classifier (80% accuracy)
- **Adaptive Learning**: Correctly identified hardest samples (1 & 5) and gave them highest weights
- **Feature Importance**: Income was most discriminative feature
- **Confidence**: All correct predictions have high confidence margins (¿0.8)

Why Sample 1 Was Misclassified

- **Features**: Male, 41 years, \$40,000 income
- **True Label**: Has illness (+1)
- **Conflicting Signals**:
    - Gender (Male) suggests illness
    - Age (41) suggests illness
    - Income (\$40K) suggests no illness
- **Problem**: Income classifier had highest weight (0.9726)
- **Result**: Income feature dominated the decision

Classifier Weights

Feature	Weight
Income	0.9726
Age	0.3466
Gender	0.2027