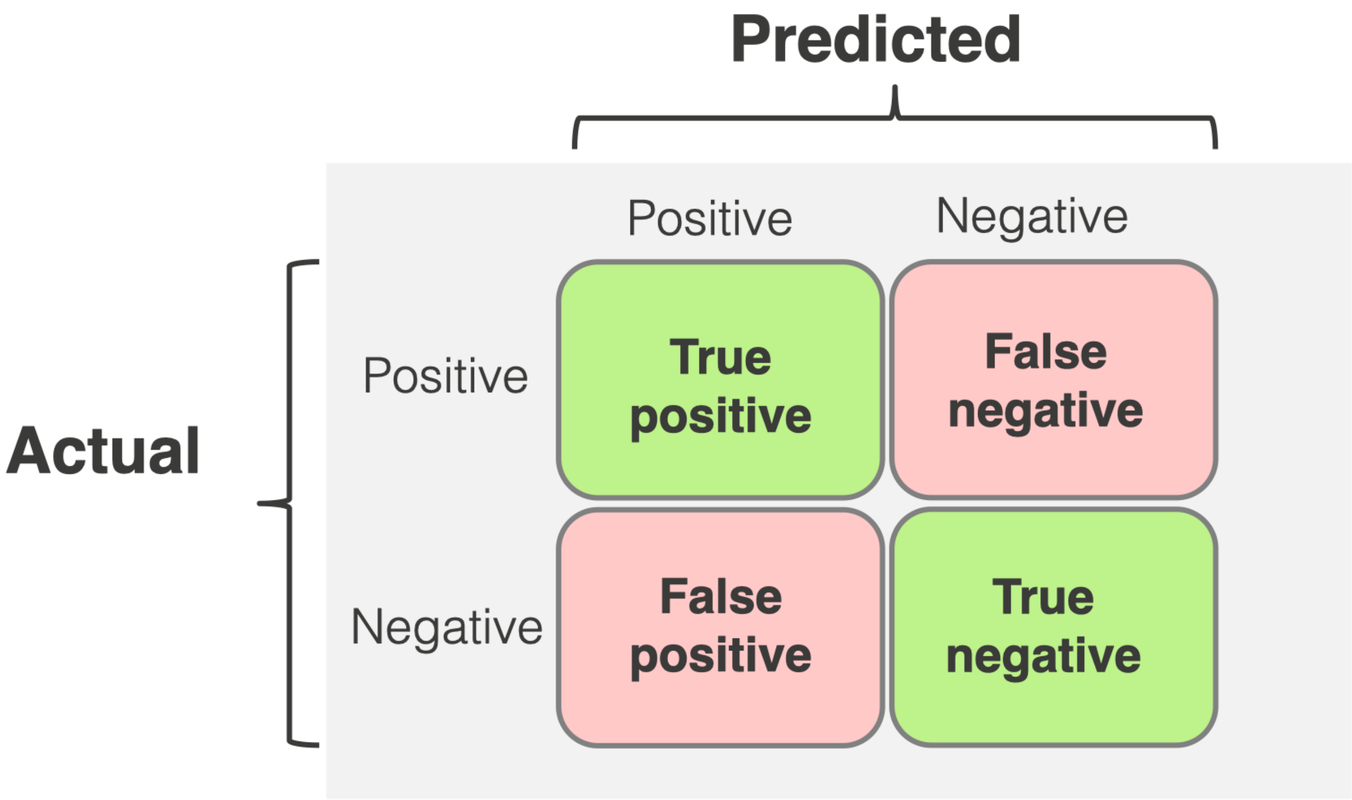
COMP30027 MACHINE LEARNING TUTORIAL

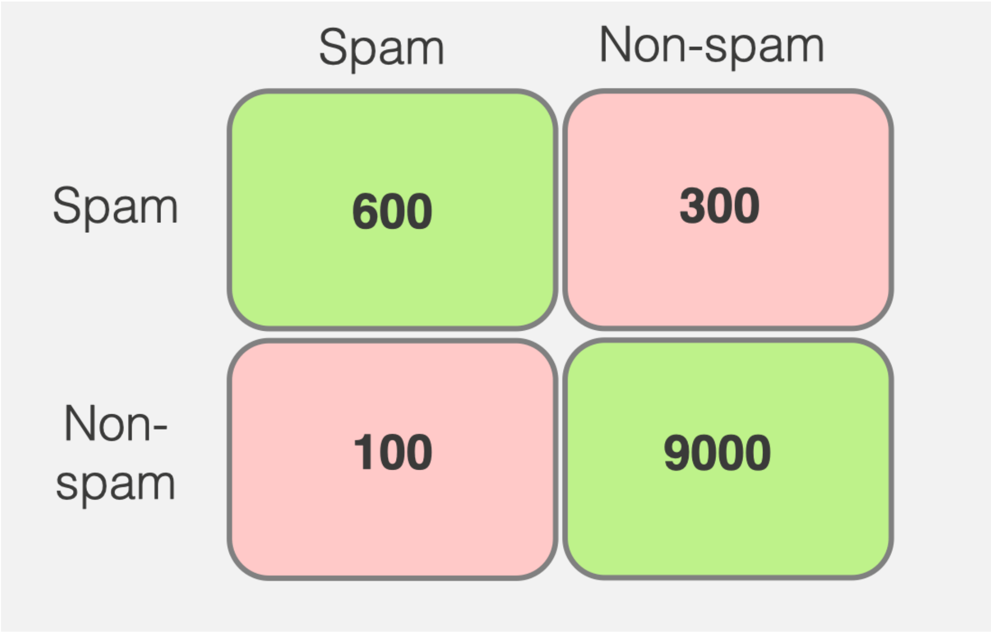
Workshop - 4

Model Evaluation and Decision Trees

We'll cover evaluation metrics, baselines, decision trees, and data-splitting strategies.

Understanding Classification Results





- True Positive (TP): The model **correctly predicts** the **positive class**. For example, a spam email is correctly identified as spam.
- False Positive (FP): The model **incorrectly predicts positive class** when it's actually negative. For Example, normal email is wrongly labelled as spam. (Also known as a **Type I Error**.)
- False Negative (FN): The model **misses** the positive class—it predicts negative when it's actually positive. For example, a spam email is misclassified as not spam. (Also known as a **Type II Error**.)
- True Negative (TN): The model **correctly predicts** the **negative class**. For example, a normal email is correctly identified as not spam.

https://www.evidentlyai.com/classification-metrics/confusion-matrix

Understanding Model Evaluation Metrics

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{All predictions}}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

	Fraud	Not Fraud
Fraud	0	100
Not Fraud	0	99,900

Scenario: Fraud Detection in Credit Card Transactions

You're building a model to detect **fraudulent transactions**.

•You have **100,000 transactions**

•Only **100 are actually fraudulent**

•That means:

- **99,900 are legitimate**
- **100 are fraud**

So, the class distribution is **heavily imbalanced**

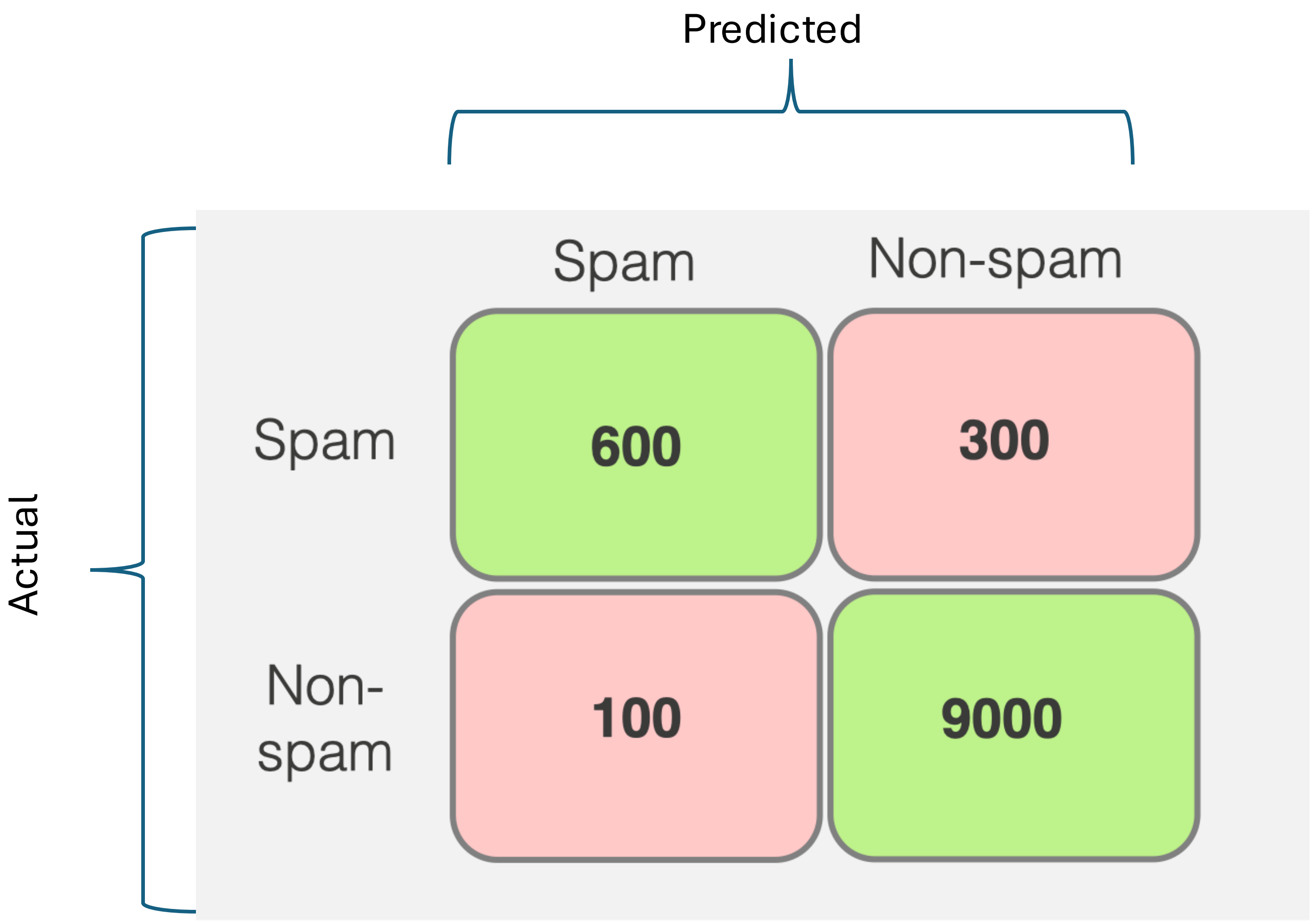
$$\frac{TP + TN}{Total} = \frac{0 + 99,900}{100,000} = 99.9\%$$

Even though the model achieves 99.9% accuracy, it fails to detect any fraudulent transactions. So, when the class distribution is highly imbalanced, accuracy is not a reliable evaluation metric

Understanding Model Evaluation Metrics

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

$$\text{Precision} = \frac{600}{600+100}$$



https://www.evidentlyai.com/classification-metrics/confusion-matrix

Understanding Model Evaluation Metrics

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

$$\text{Recall} = \frac{5}{5+10}$$

	Detected With Cancer	Not Detected With Cancer
Have Cancer	5 (TP)	10 (FN)
Do Not Have Cancer	45 (FP)	940 (TN)

https://medium.com/x8-the-ai-community/understanding-ml-evaluation-metrics-precision-recall-2b3fb915b666

Understanding Model Evaluation Metrics

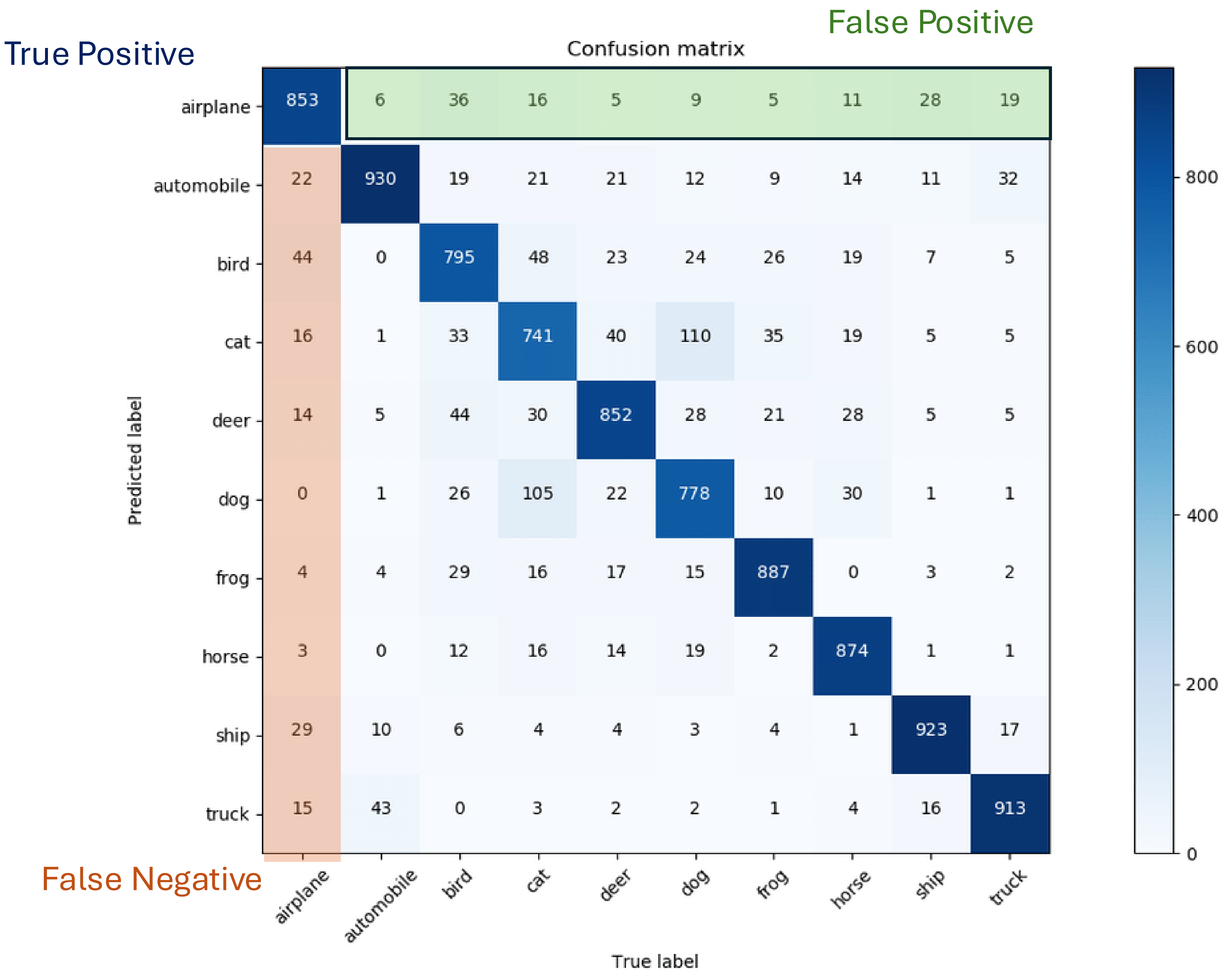
The general formula for non-negative real β is:

$$F_\beta = \frac{(1+\beta^2)\cdot(\mathrm{precision}\cdot\mathrm{recall})}{(\beta^2\cdot\mathrm{precision}+\mathrm{recall})}$$

Beta (β) controls the **balance** between **precision** and **recall**:

- **β > 1**: More weight on **recall** (sensitive to false negatives)
- **β < 1**: More weight on **precision** (sensitive to false positives)
- **β = 1**: Equal weight → This gives the **F1-score**

Understanding Model Evaluation Metrics



A **confusion matrix** is a table used to evaluate the performance of a classification model by comparing the predicted labels with the actual labels.

The **highlighted green** areas represent **false positives** for the *aeroplane* class, while the **brown** areas indicate **false negatives**. The **dark blue** section corresponds to the **true positives** for the *aeroplane* class in the confusion matrix.

https://doi.org/10.48550/arXiv.2109.00573

Q4
A confusion matrix is a summary of the performance of a (supervised) classifier over a set of development ("test") data, by counting the various instances:

	Predicted +	**Predicted -**
Actual +	10	2
Actual -	5	7

Calculate the precision, recall, and F-score (where β = 1) for this classifier.

- **Precision** : 10/15 = 0.667
- **Recall** : 10/12 = 0.833
- **F-score:** $PR/\ P+R$ =2∗0.667∗0.8330.667+0.833=0.741

Metric	Formula
True positive rate, recall	$\frac{\text{TP}}{\text{TP+FN}}$
False positive rate	$\frac{\text{FP}}{\text{FP+TN}}$
Precision	$\frac{\text{TP}}{\text{TP+FP}}$
Accuracy	$\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$
F-measure	$\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

Baseline vs Benchmark

Baseline: starting point for comparison

- **Zero-R Rule** (0-R)
- **One Rule** (1-R)

Benchmark: known standard or state-of-the-art that you compare your model against

- CNN
- Linear Regression
- Random Forest

Zero-R (Majority Class)

Always predicts the most frequent class. Simplest possible classifier. Implementation requires just a counter.

One-R (Single Rule)

Generates one rule based on one feature. Finds the attribute with minimum error rate.

Training set:

ID	Outlook	Temp	Humid	Wind	Play
A	S	H	H	F	N
B	S	H	H	T	N
C	O	H	H	F	Y
D	R	M	H	F	Y
E	R	C	N	F	Y
F	R	C	N	T	N

Test set:

ID	Outlook	Temp	Humid	Wind	Play
G	O	M	N	T	?
H	S	H	H	F	?

Zero-R Rule (O-R)

Training set:

ID	Outlook	Temp	Humid	Wind	Play
A	S	H	H	F	N
B	S	H	H	T	N
C	O	H	H	F	Y
D	R	M	H	F	Y
E	R	C	N	F	Y
F	R	C	N	T	N

3

It's a tie, so you can choose either N or Y

Test set:

ID	Outlook	Temp	Humid	Wind	Play
G	O	M	N	T	N
H	S	H	H	F	N

One-R Rule (1-R)

Step 1: Find the majority label and compute errors for each feature

ID	Outlook	Temp	Humid	Wind	Play
A	S	H	H	F	N
B	S	H	H	T	N
C	O	H	H	F	Y
D	R	M	H	F	Y
E	R	C	N	F	Y
F	R	C	N	T	N

Outlook	**Labels**	**Majority Label**	**Errors**
S	N, N	N	0
O	Y	Y	0
R	Y, Y, N	**Y (2)**	1 (1 N misclassified)

Temp	**Labels**	**Majority Label**	**Errors**
H	N, N, Y	N (2)	1 (Y is misclassified)
M	Y	Y	0
C	Y, N	Y	1 (1 N misclassified)

Humid	**Labels**	**Majority Label**	**Errors**
H	N, N, Y, Y	**N (2)**	2 (2 Y misclassified)
N	Y, N	Y	1 (1 N misclassified)

Wind	**Labels**	**Majority Label**	**Errors**
F	N, Y, Y, Y	**Y (3)**	1 (1 N misclassified)
T	N, N	N	0

Outlook: **1 error**

Temp: 2 errors

Humid: 3 errors

Wind: **1 error**

Here, both outlook and wind have the same number of errors, so we are randomly choosing outlook

Step 2: Create Final Rule Based on Best Feature (Outlook)

Outlook	Predict
S	N
O	Y
R	Y

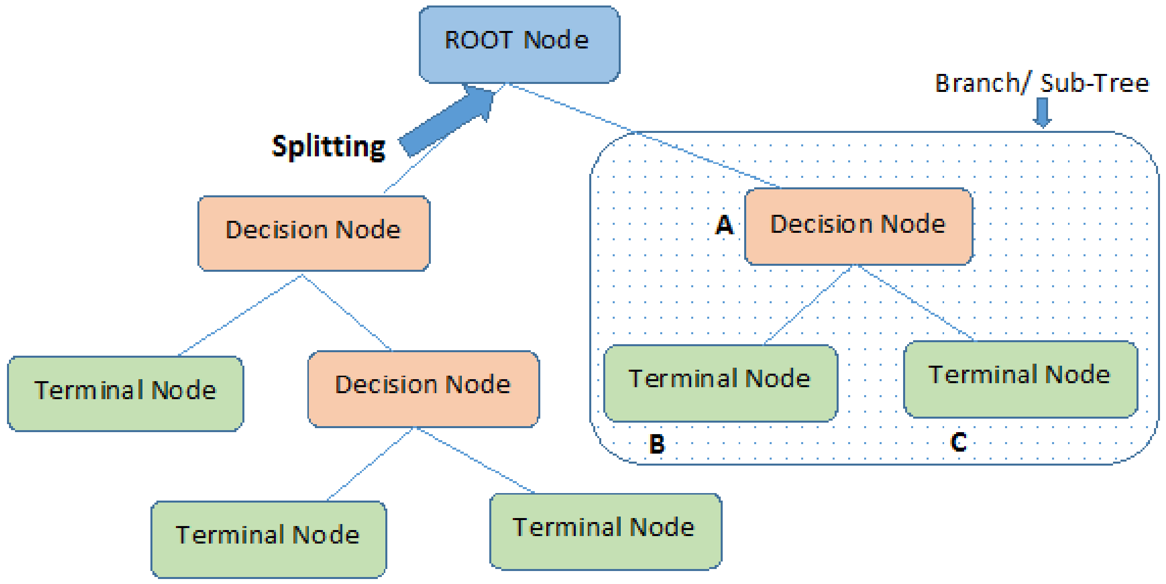
Test set:

ID	Outlook	Temp	Humid	Wind	Play
G	O	M	N	T	?
H	S	H	H	F	?

Step 3: Make Final Prediction based on Best Feature (Outlook) from step 2

ID	Outlook	Prediction
G	O	Y
H	S	N

Decision Trees: Fundamentals



Tree Structure

Hierarchical model with nodes and branches. Each node represents a decision point.

Splitting Criteria

Uses information gain (used in ID3), Gain ratio, etc

Leaf Nodes

Terminal nodes that provide final predictions.

https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html

ID3 Decision Tree Implementation Algorithm

- Basic method: construct decision trees in **recursive** divide-and-conquer fashion

FUNCTION ID3 (*Root*)
IF all instances at *Root* have the same class label*
THEN stop
ELSE 1. Select an attribute to use in partitioning *Root* node instances
2. Create a branch for each attribute value and partition up *Root* node instances according to each value
3. Call **ID3**($LEAF_i$) for each leaf node $LEAF_i$

* Note: we may not end up with pure leaves (all instances with the same class label). Therefore, our stopping criterion may actually be a threshold $purity(Root) > \theta$

At each step, it:

1. **Computes entropy of parent** (how mixed the class labels are).
2. **Computes information gain** for each feature.
3. **Chooses the feature with highest gain** to split the data.
4. Repeats this process **recursively** on child branches.

Entropy

The entropy of a discrete random event x with possible states $1, \ldots, n$ is:

$$H(x) = -\sum_{i=1}^{n} P(i)\ \log_2 P(i)$$

where $0\ \log_2 0 \stackrel{\text{def}}{=} 0$

In the context of Decision Trees, we are looking at the class distribution at a node:

- **50 Y instances, 5 N instances**:
$$H = -\left[\frac{50}{55}\log_2\frac{50}{55} + \frac{5}{55}\log_2\frac{5}{55}\right] \approx 0.44\ bits$$
- **30 Y instances, 25 N instances**:
$$H = -\left[\frac{30}{55}\log_2\frac{30}{55} + \frac{25}{55}\log_2\frac{25}{55}\right] \approx 0.99\ bits$$
- We want leaves with **low entropy**!

How do we choose the attribute to partition the root node instances?

Attribution Selection: Information Gain

- Select attribute R_A (with values $x_1, \dots, x_m$) best splits the instances at a given root node R according to information gain:

$$IG(R_A|R) = H(R) - \sum_{j=1}^{m} P(x_j)\, H(x_j)$$

$$\text{Information Gain} = \text{Entropy of parent} - \text{Mean Information}$$

Q3

Classify the test instances using the ID3 Decision Tree method:

1.Using **information gain** as the splitting criterion

Training set:

ID	Outlook	Temp	Humid	Wind	Play
A	S	H	H	F	N
B	S	H	H	T	N
C	O	H	H	F	Y
D	R	M	H	F	Y
E	R	C	N	F	Y
F	R	C	N	T	N

Test set:

ID	Outlook	Temp	Humid	Wind	Play
G	O	M	N	T	?
H	S	H	H	F	?

Step 1: Calculate Entropy of Root (Target Class - Play)

From the training set, we have:

- 3 instances where **Play = Y**
- 3 instances where **Play = N**

Entropy at root is:

$$H(R) = -\left(\frac{3}{6}\log_2\frac{3}{6} + \frac{3}{6}\log_2\frac{3}{6}\right) = 1$$

This means we're **completely uncertain** at the start (equal split).

At each step, it:

1. **Computes entropy of parent** (how mixed the class labels are).
2. **Computes information gain** for each feature.
3. **Chooses the feature with highest gain** to split the data.
4. Repeats this process **recursively** on child branches.

Step 2: Try Splitting on All Features

Now we calculate **Information Gain** for each feature.

Information Gain =

$$IG(\text{Feature}) = \text{Entropy at parent} - \text{Weighted average entropy of children}$$

Feature: Outlook

Value	Counts	Play Yes	Play No
S	2	0	2
O	1	1	0
R	3	2	1

- $Ent(S) = -\left[0 \log_2 0 + \frac{2}{2} \log_2 \frac{2}{2}\right] = 0$
- $Ent(O) = 0$
- $Ent(R) = -\left(\frac{2}{3} \log_2 \frac{2}{3} + \frac{1}{3} \log_2 \frac{1}{3}\right) \approx 0.918$

$$Gain(Outlook) = 1 - \left(\frac{2}{6} \cdot 0 + \frac{1}{6} \cdot 0 + \frac{3}{6} \cdot 0.918\right) = 1 - 0.459 = 0.541$$

At each step, it:

1. **Computes entropy of parent** (how mixed the class labels are).
2. **Computes information gain** for each feature.
3. **Chooses the feature with highest gain** to split the data.
4. Repeats this process **recursively** on child branches.

Feature: Temp

Value	Counts	Play Yes	Play No
H	3	1	2
M	1	1	0
C	2	1	1

$$Ent(H) \approx 0.918,\ Ent(M) = 0,\ Ent(C) = 1$$

$$Gain(Temp) = 1 - \left(\frac{3}{6} \cdot 0.918 + \frac{1}{6} \cdot 0 + \frac{2}{6} \cdot 1\right) = 1 - (0.459 + 0 + 0.333) = 0.208$$

Feature: Humid

Value	Counts	Play Yes	Play No
H	4	2	2
N	2	1	1

- $Ent(H) = 1,\ Ent(N) = 1$

$$Gain(Humid) = 1 - (4/6 * 1 + 2/6 * 1) = 1 - 1 = 0$$

Feature: Wind

Value	Counts	Play Yes	Play No
F	4	3	1
T	2	0	2

- $Ent(F) = -\left(\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}\right) \approx 0.811$
- $Ent(T) = 0$

$$Gain(Wind) = 1 - \left(\frac{4}{6}\cdot 0.811 + \frac{2}{6}\cdot 0\right) = 1 - 0.541 = 0.459$$

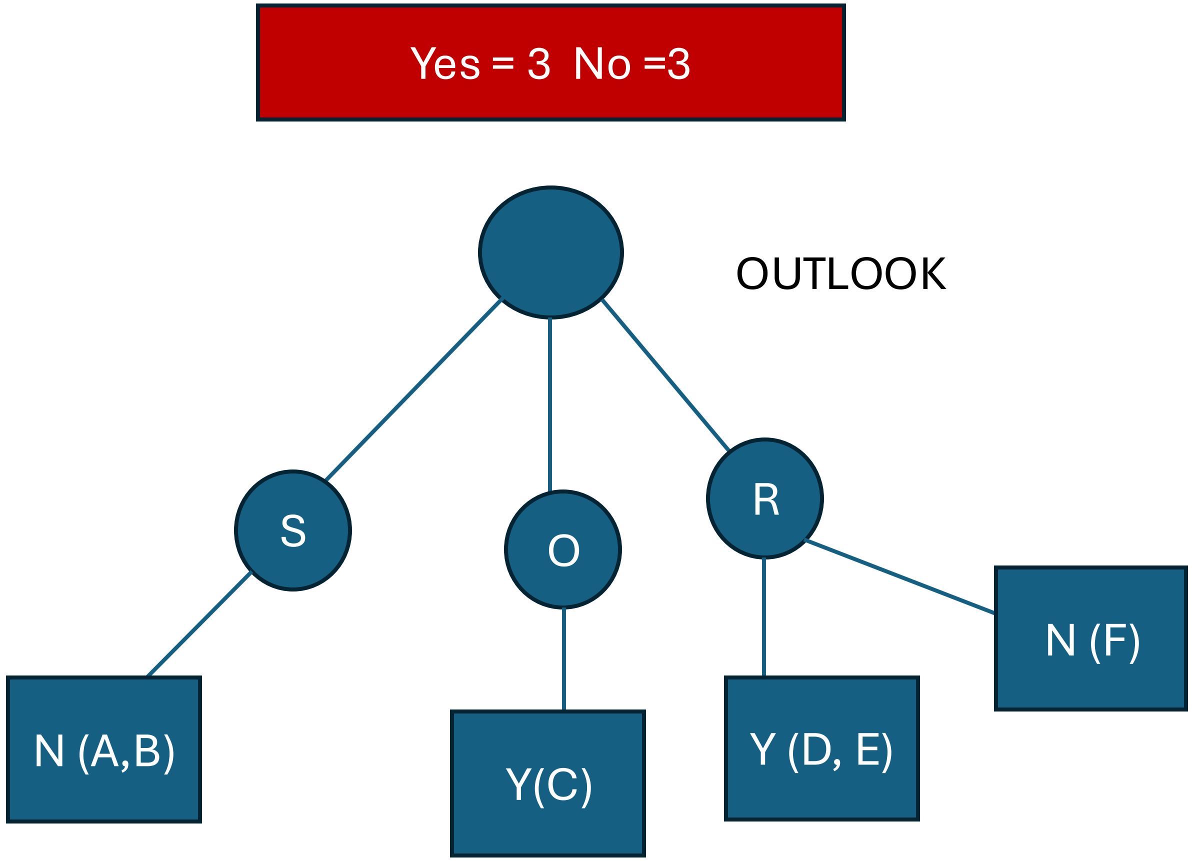
Best Split: Highest IG

Feature	Gain
Outlook	**0.541** ← Best
Wind	0.459
Temp	0.208
Humid	0

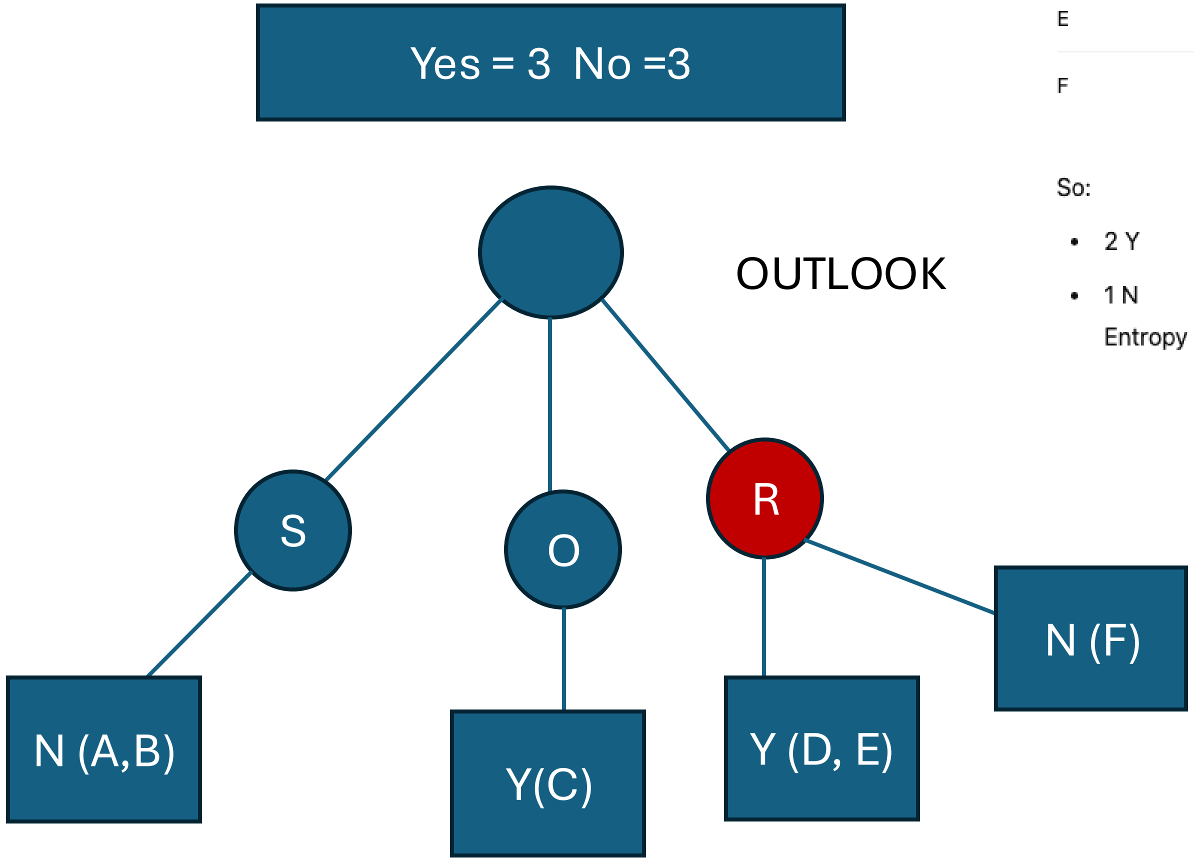
At each step, it:

1. **Computes entropy of parent** (how mixed the class labels are).
2. **Computes information gain** for each feature.
3. **Chooses the feature with highest gain** to split the data.
4. Repeats this process **recursively** on child branches.

Build Tree (First Split: Outlook)



Step 1: Calculate Entropy of Root (Outlook = R)



ID	Temp	Humid	Wind	Play
D	M	H	F	Y
E	C	N	F	Y
F	C	N	T	N

So:

- 2 Y
- 1 N

Entropy $H(R) = -\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right) \approx 0.9183$

At each step, it:

1. **Computes entropy of parent** (how mixed the class labels are).
2. **Computes information gain** for each feature.
3. **Chooses the feature with highest gain** to split the data.
4. Repeats this process **recursively** on child branches.

Step 2: Try Splitting on All Features

Feature: Temp

Temp	Play
M	Y → Entropy = 0
C	Y, N → Entropy = 1

Mean Info:

$$MI(Temp) = \frac{1}{3}(0) + \frac{2}{3}(1) = 0.6667$$

$$IG(Temp) = 0.9183 - 0.6667 = 0.2516$$

Feature: Humid

Humid	Play
H	Y → Entropy = 0
N	Y, N → Entropy = 1

Same calculation as above:

$$MI(Humid) = \frac{1}{3}(0) + \frac{2}{3}(1) = 0.6667$$

$$IG(Humid) = 0.9183 - 0.6667 = 0.2516$$

ID	Temp	Humid	Wind	Play
D	M	H	F	Y
E	C	N	F	Y
F	C	N	T	N

Feature: Wind

Wind	Play
F	Y, Y → Entropy = 0
T	N → Entropy = 0

$$MI(Wind) = \frac{2}{3}(0) + \frac{1}{3}(0) = 0$$

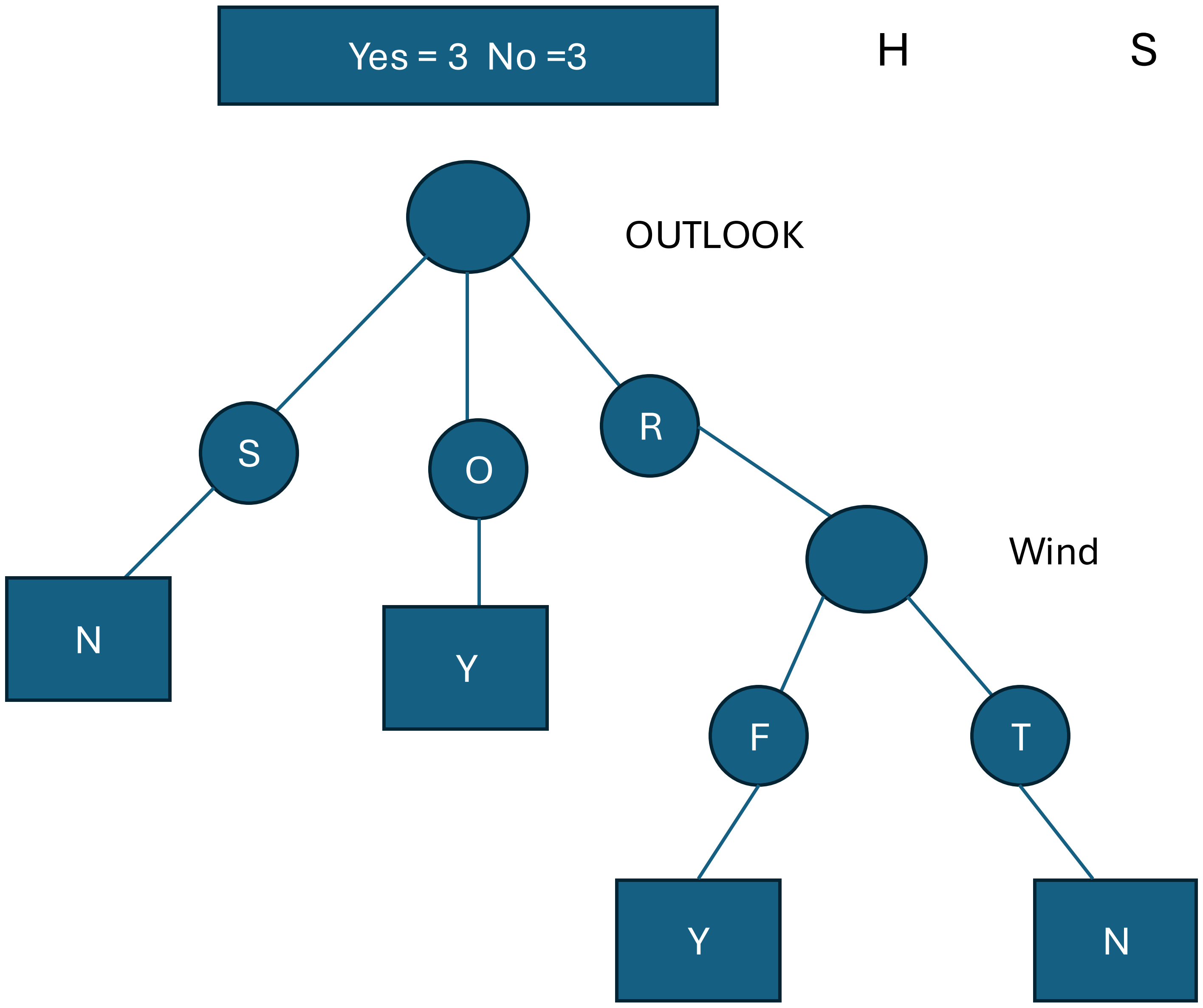
$$IG(Wind) = 0.9183 - 0 = 0.9183$$

Best Split: Highest IG

Wind

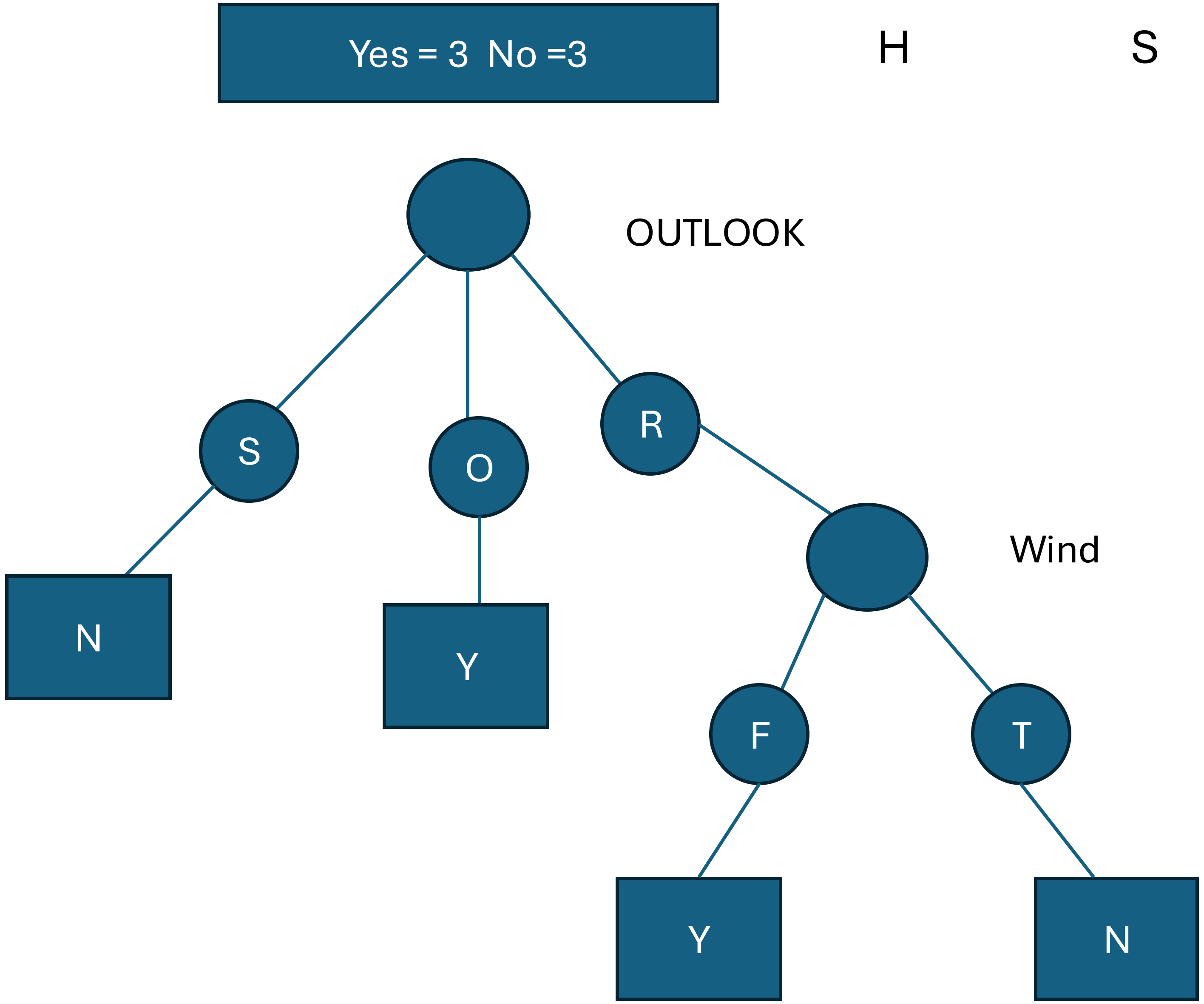
Final Decision Tree (Second split = Wind)

ID	Outlook	Temp	Humid	Wind	Play
G	O	M	N	T	?
H	S	H	H	F	?



Final Decision Tree

ID	Outlook	Temp	Humid	Wind	Play
G	O	M	N	T	Y
H	S	H	H	F	N



Shortcomings of Information Gain

- Information gain tends to prefer highly-branching attributes

- What if we split instances using ID label?

When using **information gain**, ID3 tends to prefer attributes with **many distinct values**, which can be misleading (e.g., ID has a unique value for each example — perfect split, but useless for generalization).

Gain Ratio

Gain Ratio (GR) reduces the bias for information gain towards highly-branching attributes by normalising relative to the split info

$$\text{Gain Ratio} = \frac{\text{Information Gain}}{\text{Split Information}}$$

$$SplitInfo = -\sum \frac{N_i}{N} log_2 \frac{N_i}{N}$$

where Ni is the number of data points containing each value of the variable

and N is the total number of data points

	R	Outlook			Temp			Humid		Wind		ID					
		s	o	r	h	m	c	h	n	T	F	A	B	C	D	E	F
Y	3	0	1	2	1	1	1	2	1	0	3	0	0	1	1	1	0
N	3	2	0	1	2	0	1	2	1	2	1	1	1	0	0	0	1
Total	6	2	1	3	3	1	2	4	2	2	4	1	1	1	1	1	1
P(Y)	1/2	0	1	2/3	1/3	1	1/2	1/2	1/2	0	3/4	0	0	1	1	1	0
P(N)	1/2	1	0	1/3	2/3	0	1/2	1/2	1/2	1	1/4	1	1	0	0	0	1
H	1	0	0	0.9183	0.9183	0	1	1	1	0	0.8112	0	0	0	0	0	0
MI		0.4592			0.7924			1		0.5408		0					
IG		0.5408			0.2076			0		0.4592		1					

Step 2: Try Splitting on All Features

Feature: Outlook

Values: S, O, R

- S → A, B → both N → Entropy = 0
- O → C → Y → Entropy = 0
- R → D, E, F → Y, Y, N → Entropy ≈ 0.9183

Mean Information:

$$MI = \frac{2}{6}(0) + \frac{1}{6}(0) + \frac{3}{6}(0.9183) \approx 0.4592$$

$$IG = 1 - 0.4592 = 0.5408$$

$$SI = -\left(\frac{2}{6}\log_2\frac{2}{6} + \frac{1}{6}\log_2\frac{1}{6} + \frac{3}{6}\log_2\frac{3}{6}\right) \approx 1.459$$

$$GR = \frac{0.5408}{1.459} \approx 0.3707$$

Step 2: Try Splitting on All Features

Step 2: Choose Root = **Wind**

Feature: Temp

Values: H, M, C

- H: A, B, C → N, N, Y → Entropy = 0.9183
- M: D → Y → Entropy = 0
- C: E, F → Y, N → Entropy = 1

$$MI = \frac{3}{6}(0.9183) + \frac{1}{6}(0) + \frac{2}{6}(1) \approx 0.7924$$

$$IG = 1 - 0.7924 = 0.2076$$

$$SI = -[0.5 \log_2 0.5 + 0.1667 \log_2 0.1667 + 0.3333 \log_2 0.3333] \approx 1.459$$

$$GR \approx \frac{0.2076}{1.459} \approx 0.1423$$

Feature: Humid

H → A, B, C, D → N, N, Y, Y → Entropy = 1

N → E, F → Y, N → Entropy = 1

→ Mean Information = 1

→ IG = 0

→ GR = 0

Feature: Wind

T → B, F → N, N → Entropy = 0

F → A, C, D, E → N, Y, Y, Y → Entropy ≈ 0.8112

$$MI = \frac{2}{6}(0) + \frac{4}{6}(0.8112) \approx 0.5408$$

$$IG = 1 - 0.5408 = 0.4592$$

$$SI = -\left(\frac{2}{6}\log_2\frac{2}{6} + \frac{4}{6}\log_2\frac{4}{6}\right) \approx 0.9183$$

$$GR = \frac{0.4592}{0.9183} \approx 0.5001$$

Feature: ID

$$MI(ID) = \sum_{i=1}^{6} \frac{1}{6} \cdot 0 = 0$$

$$IG(ID) = H(R) - MI = 1 - 0 = 1$$

There are 6 branches (A to F), each with probability $\frac{1}{6}$:

$$SI(ID) = -\sum_{i=1}^{6} \frac{1}{6}\log_2\frac{1}{6} = 6 \cdot \left(-\frac{1}{6} \cdot \log_2\frac{1}{6}\right) = -\log_2\frac{1}{6}$$

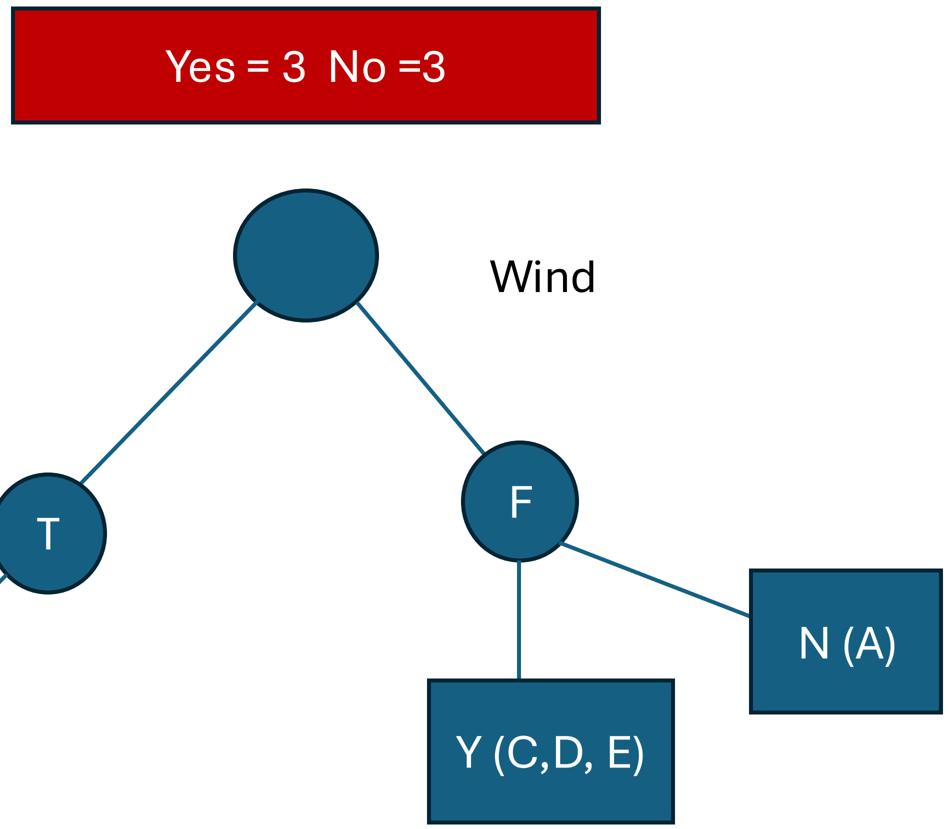
$$\log_2\frac{1}{6} = \log_2 1 - \log_2 6 = 0 - \log_2 6 \approx -2.585$$

So,

$$SI(ID) = 2.585$$

$$GR(ID) = \frac{IG}{SI} = \frac{1}{2.585} \approx 0.387$$

Build Tree (First Split: Wind)



Step 3: Evaluate Attributes in Wind = F subset

ID	Outlook	Temp	Humid	Wind	Play
A	S	H	H	F	N
C	O	H	H	F	Y
D	R	M	H	F	Y
E	R	C	N	F	Y

Feature: Outlook

Value	Play	Entropy
s	N	0
o	Y	0
r	Y, Y	0

MI = 0

IG = 0.8112

SI = −[¼log¼ + ¼log¼ + ½log½] = 1.5

GR = 0.8112 / 1.5 = **0.5408**

Feature: Temp

2 h instances (1 Y, 1 N, (H = 1))

1 m instance (Y, (H = 0))

1 c instance (Y, (H = 0))

MI=2/4(1)+1/4(0)+1/4(0)=0.5

⇒IG=0.8112−0.5=0.3112

Same instance distribution as **Outlook**, so the split information is also 1.5, and the Gain ratio is GR(Temp|Wind=F)=0.31121.5≈0.2075

Step 2: Try Splitting on All Features

Feature: Humid

3 h instances (2 Y, 1 N, (H = 0.9183))

1 n instance (Y, (H = 0))

MI=3/4(0.9183)+1/4(0)=0.6887

⇒IG=0.8112−0.6887=0.1225

Split information: $SI(\mathrm{H} \mid \mathrm{Wind} = F) = -\left[\frac{3}{4}\log_2 \frac{3}{4} + \frac{1}{4}\log_2 \frac{1}{4}\right] \approx 0.8112$

Gain ratio: $GR(\mathrm{H} \mid \mathrm{Wind} = F) = \frac{0.1225}{0.8112} \approx 0.1387$

Feature: Id

Mean information is obviously still 0, so IG = 0.8112

Split information: $SI(\mathrm{ID}) = -\left[4 \times \frac{1}{4}\log_2 \frac{1}{4}\right] = 2$

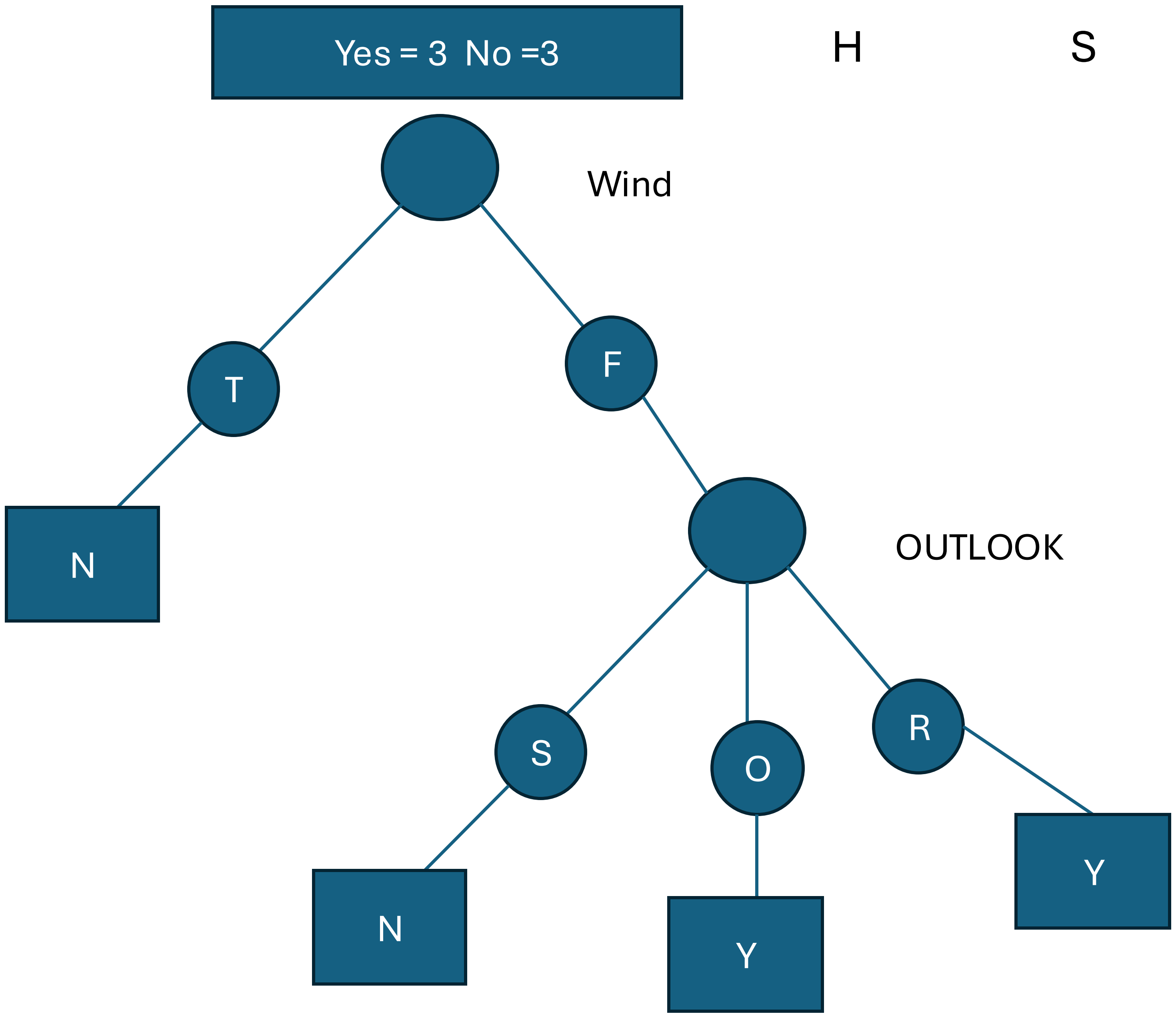
Gain ratio: $GR(\mathrm{ID}) = \frac{0.8112}{2} \approx 0.4056$

Best Split: Highest GR

Outlook = **0.5408**

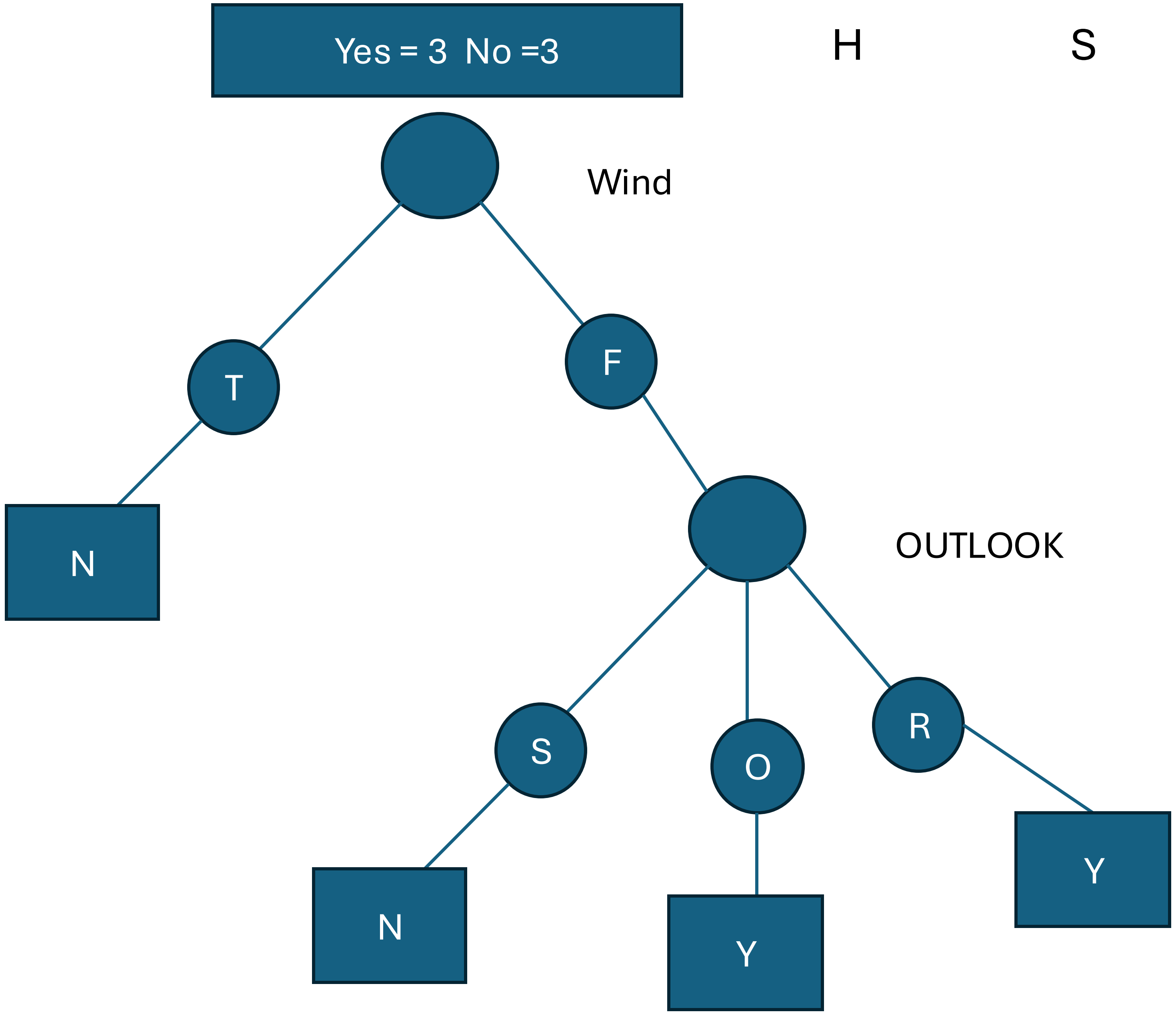
Final Decision Tree (Second split = outlook)

ID	Outlook	Temp	Humid	Wind	Play
G	O	M	N	T	?
H	S	H	H	F	?



Final Decision Tree

ID	Outlook	Temp	Humid	Wind	Play
G	O	M	N	T	N
H	S	H	H	F	N

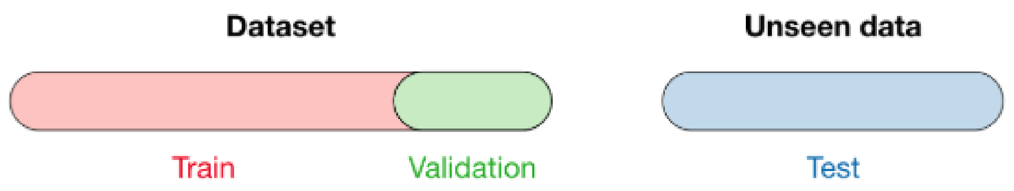


Data Splitting Techniques

Holdout

You split your dataset into two or three parts:

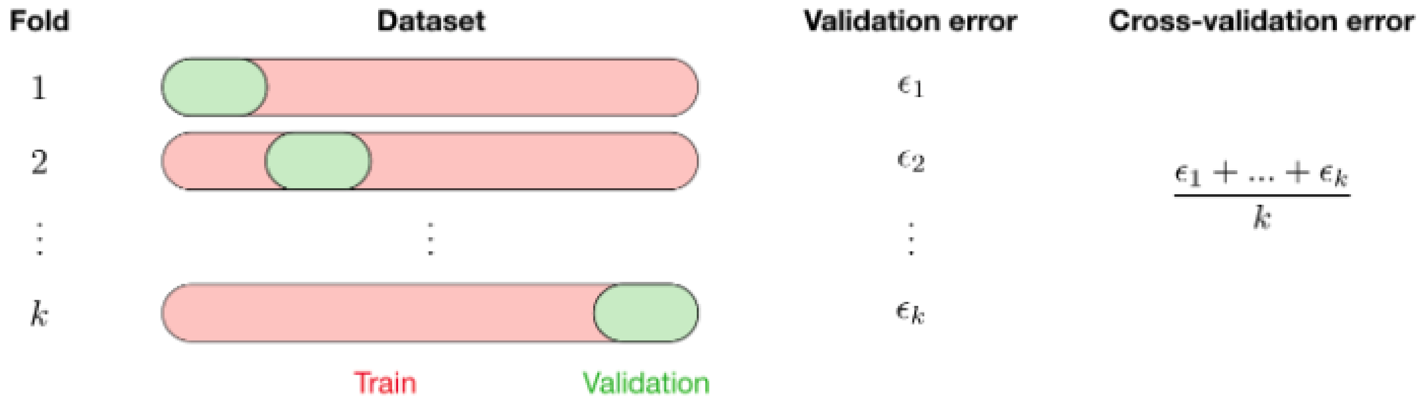
Training set: used to train the model
Validation set: used to optimize hyperparameters
Test set: used to evaluate the model's performance



K-Fold Cross-Validation

You split your dataset into k equal parts (folds) and perform k rounds of training/testing:

In each round:
Train on $k-1$ folds
Test on the remaining fold
Final score = average performance across all rounds



https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-machine-learning-tips-and-tricks

Key Takeaways and Best Practices

Choose Metrics Wisely

Select evaluation metrics that match your problem domain and business objectives.

Always Use Baselines

Establish minimum performance thresholds with simple models for comparison.

Implement Proper Data Splits

Ensure train/validation/test splits reflect real-world data distribution and use cases.

Iterate and Improve

Use evaluation insights to refine models continuously. Monitor for drift in production.