

Advanced Analytical Theory and Methods: Classification

Prof. Dr. Shamim Akhter

Professor, Dept. of CSE

Ahsanullah University of Science and Technology

Classification

- In classification learning,
 - a classifier is presented with a set of already classified examples; from these examples, the classifier learns to assign unseen examples.
 - assign class labels to new observations.
 - Logistic regression is one of the popular classification methods.

Tree-Based Learning

- Segmenting the predictor space into several simple regions.
 - Apply some splitting rules
 - Rather training and observation and classification
 - Formulate a tree-based learning

Decision Tree Methods

A decision tree (also called a prediction tree) **uses a tree structure** to specify sequences of decisions and consequences.

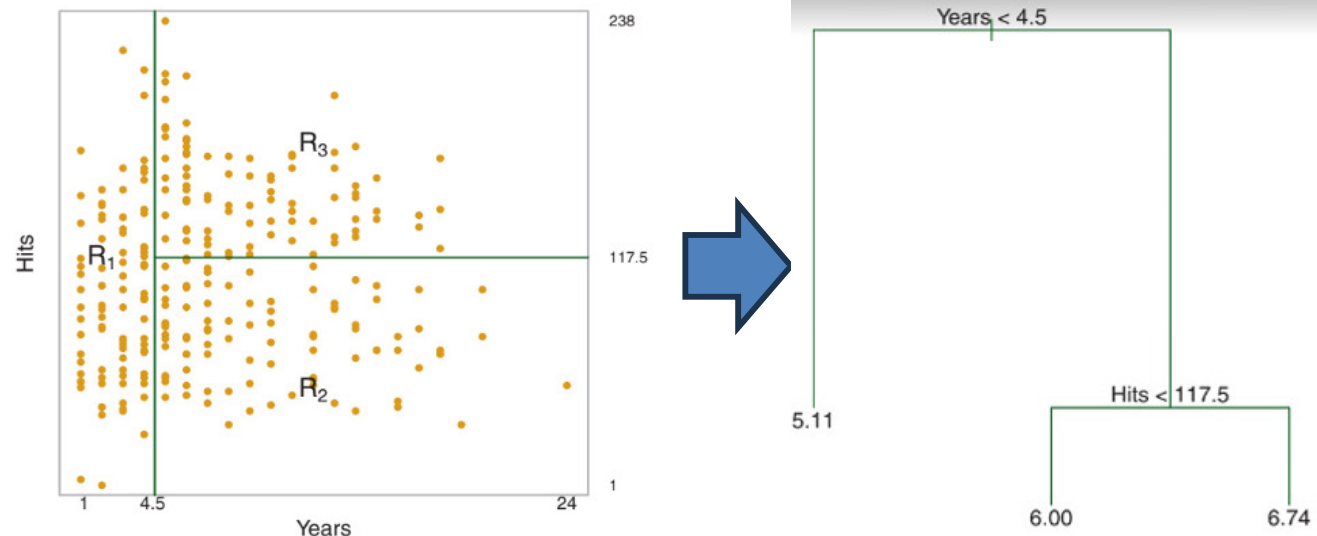
Given input $X = \{x_1, x_2, \dots, x_n\}$, the goal is **to predict a response or output variable Y** . Each set member $\{x_1, x_2, \dots, x_n\}$ is called an input variable.

Decision Tree

- Decision trees have two varieties: classification trees and regression trees.
 - Classification trees usually apply to output variables that are **categorical-often binary in nature**, such as yes or no, purchase or not purchase, and so on.
 - Regression trees, on the other hand, can apply to **output variables that are numeric or continuous**, such as the predicted price of a consumer good or the likelihood that a subscription will be purchased.

Process of building a regression tree.

- We divide the predictor space—that is, the set of possible values for X_1, X_2, \dots, X_p —into J distinct and non-overlapping regions, R_1, R_2, \dots, R_J .
- For every observation that falls into the region R_j , we make the same prediction, which is simply the mean of the response values for the training observations in R_j .
- R_j can be any shape



How do we construct the regions R_1, \dots, R_J ?

Goal is to find regions R_1, \dots, R_J that minimize the Residual (RSS), given by

$$\sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2, \quad (8.1)$$

where \hat{y}_{R_j} is the mean response for the training observations within the j^{th} box.

Computationally Infeasible

To consider every possible partition of the feature space into J boxes.

Recursive Binary Splitting

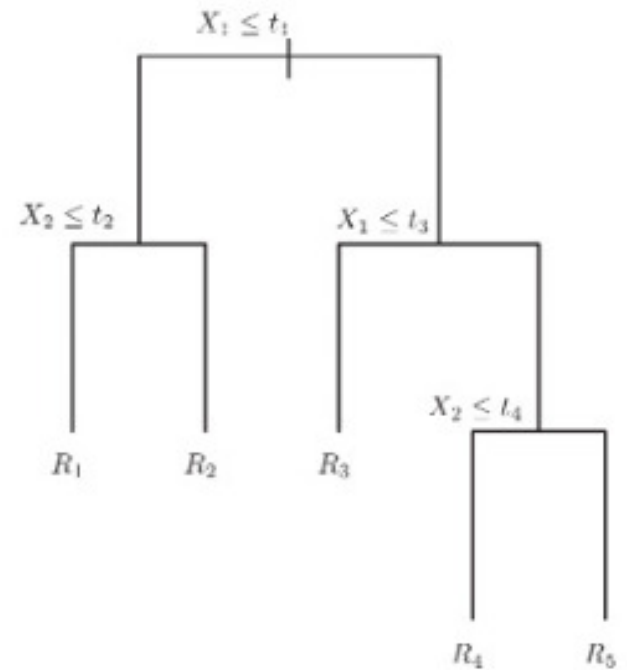
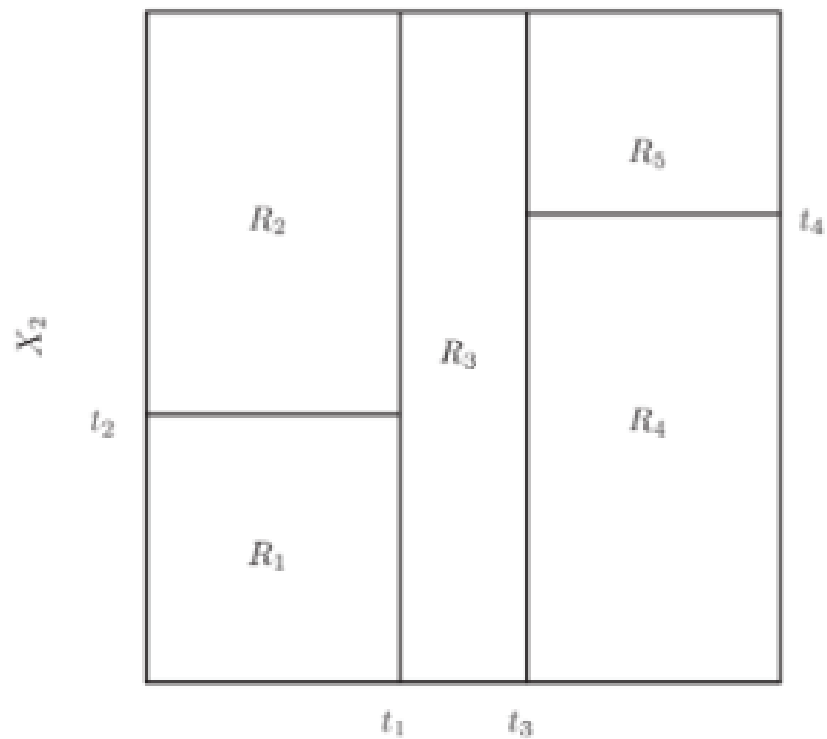
- A top-down greedy approach
- We consider all predictors X_1, \dots, X_p , and all possible values of the cutpoints for each of the predictors, and then choose the predictor and cutpoint such that the resulting tree has the lowest RSS.

$$R_1(j, s) = \{X | X_j < s\} \quad \text{and} \quad R_2(j, s) = \{X | X_j \geq s\}, \quad (8.2)$$

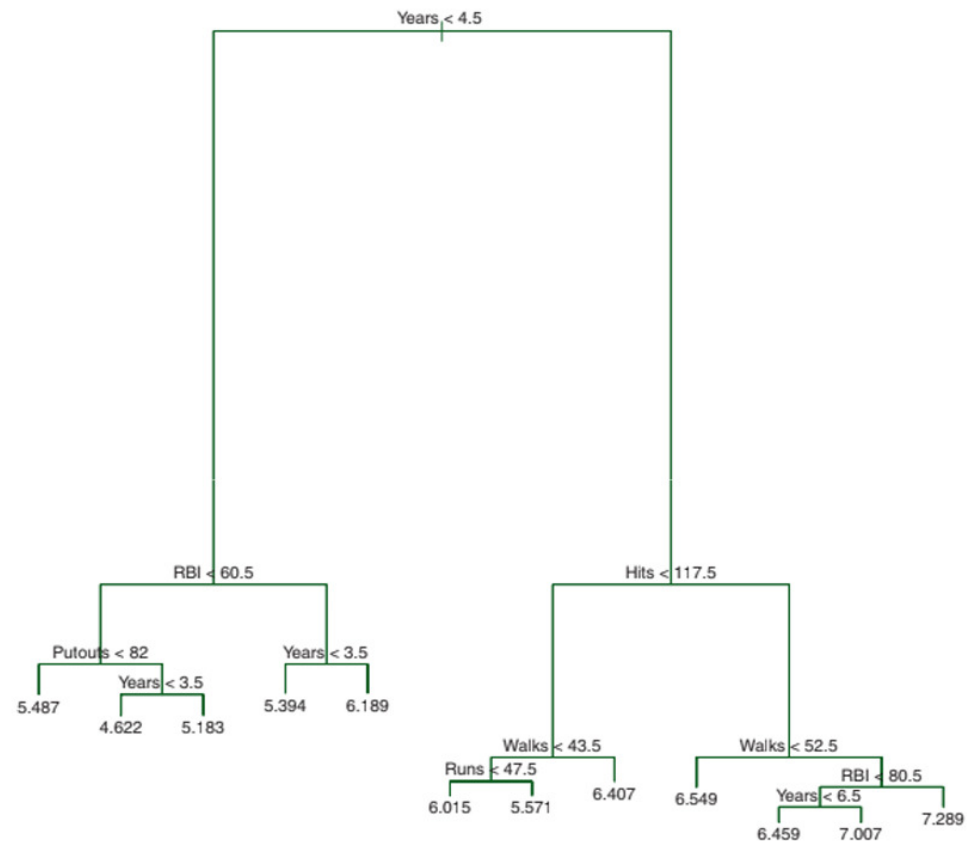
and we seek the value of j and s that minimize the equation

$$\sum_{i: x_i \in R_1(j, s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j, s)} (y_i - \hat{y}_{R_2})^2, \quad (8.3)$$

Next, we repeat the process, looking for **the best predictor and best cutpoint** in order to split the data further so as to minimize the RSS within each of the resulting regions. However, we just split one of the two previously identified regions.



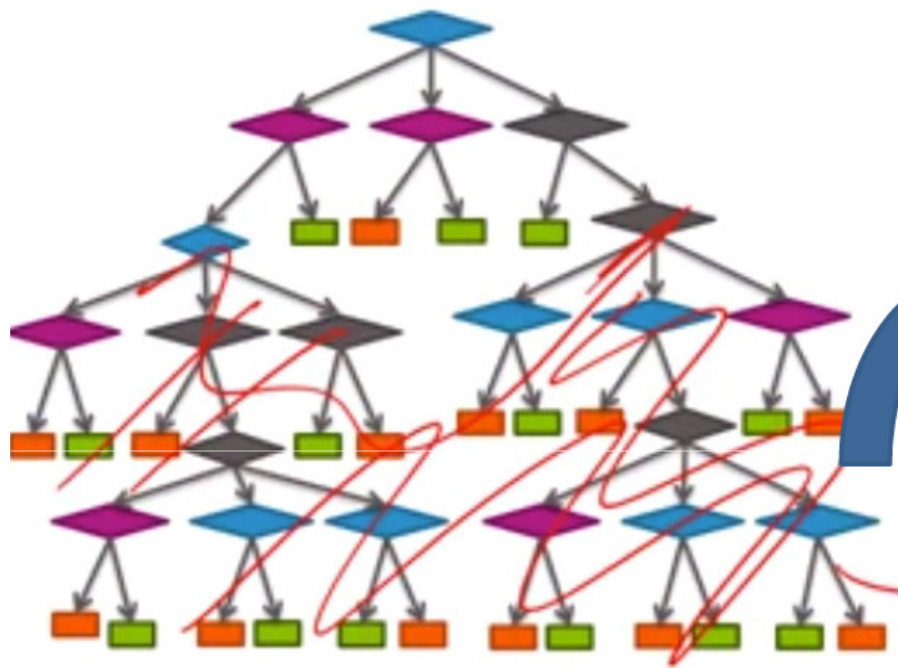
- Recursive Binary Splitting
 - Creates over fitting problem



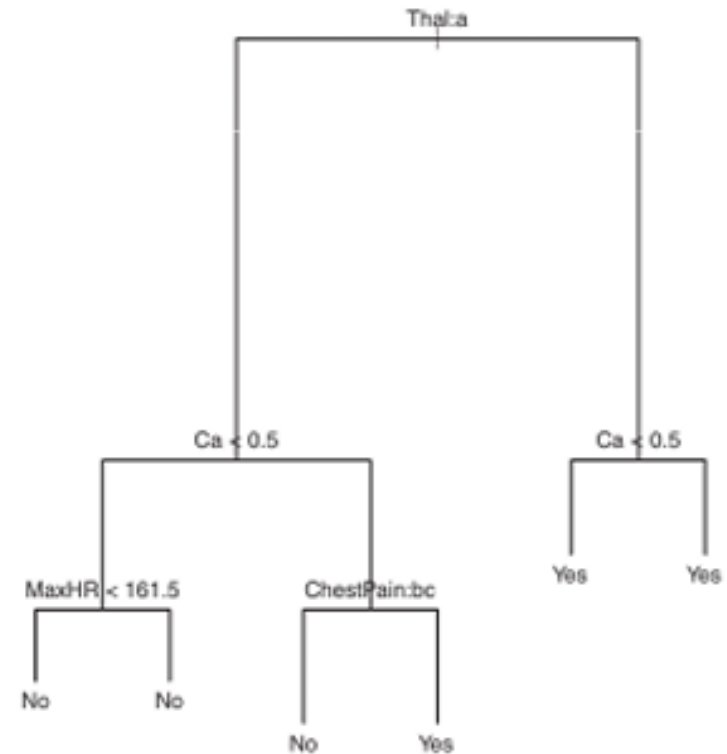
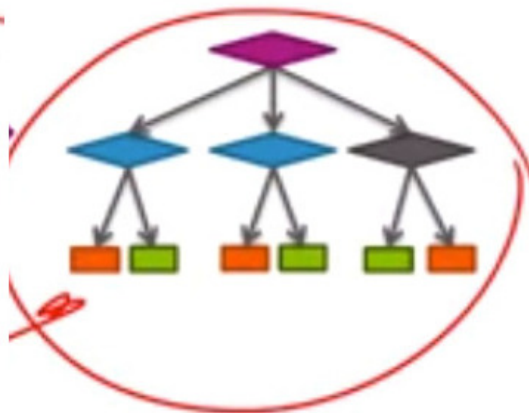
Solution: • Build the tree only so long as the decrease in the RSS due to each split exceeds some (high) threshold.

– result smaller trees

Complex Tree

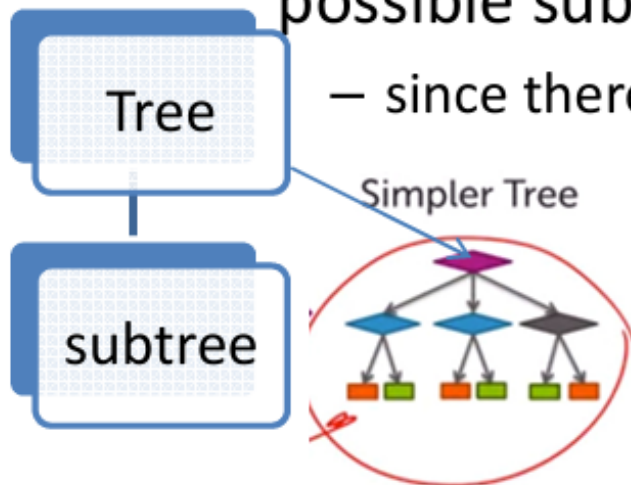


Simpler Tre



How do we determine **the best prune way** to prune the tree?

- The goal is to select **a subtree that leads to the lowest test error rate**:
 - Given a subtree, we can estimate its test error using cross-validation approach.
 - However, estimating the cross-validation error for every possible subtree would be too cumbersome
 - since there is an extremely large number of possible subtrees.



Cost Complexity Pruning

- Simpler tree -> balanced tree
- **Two observations**
 - How well the tree fits data -> Error
 - Complexity of the tree
 - Lower depth & lower leaf (standard #)
- Total Cost = measure of fit + measure of complexity

Regression error
Larger = bad fits to
training data

No of leaf nodes

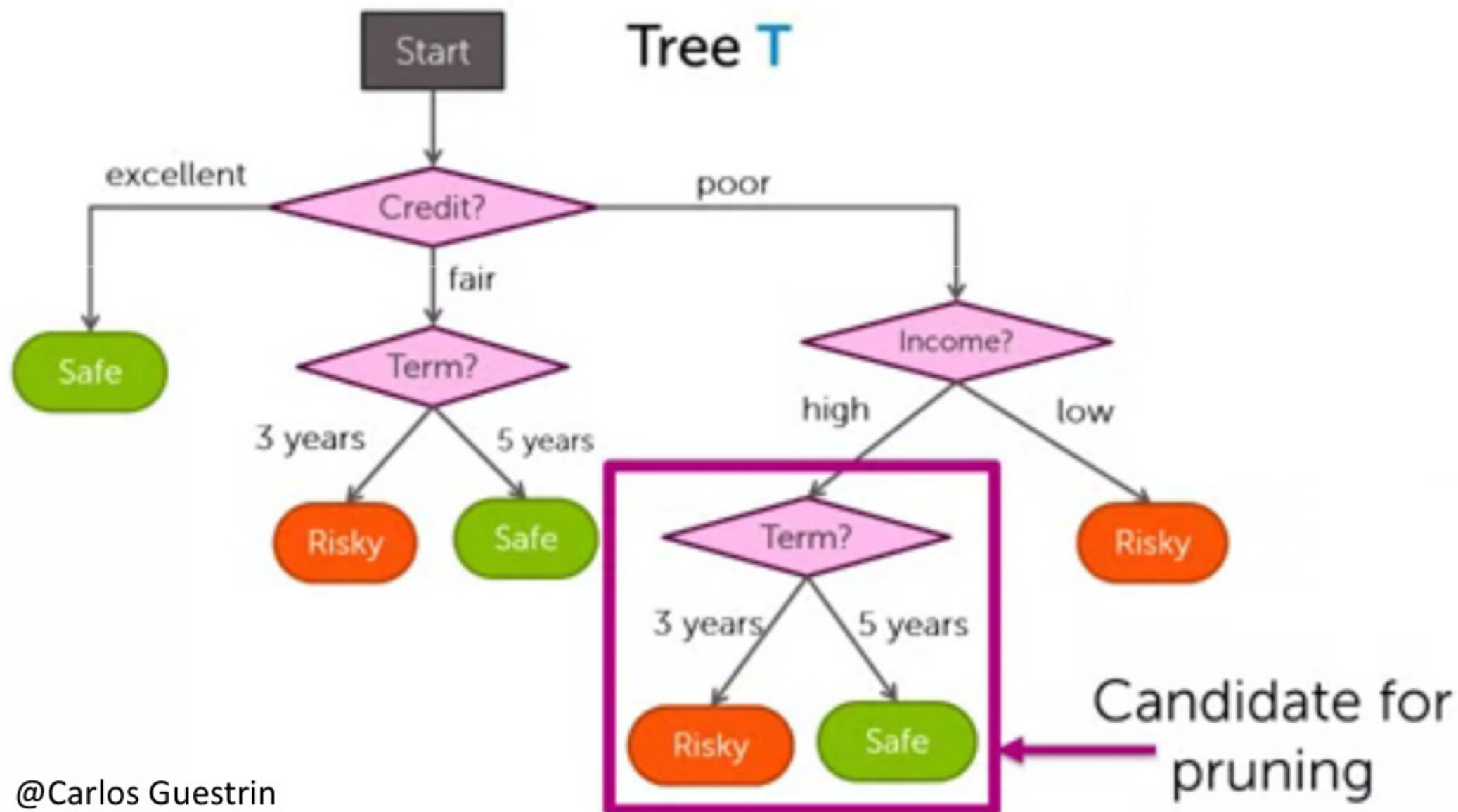
Total Cost = Regression Error + Number of leaf Nodes

Total Cost = Regression Error + Number of leaf Nodes

- A tuning parameter α can be added to balance the trade-offs between above error and leaf
- $\alpha = 0$ standard decision tree learning
- $\alpha = \infty$ big penalty \rightarrow many leaf node
- α in between balance fit and complexity

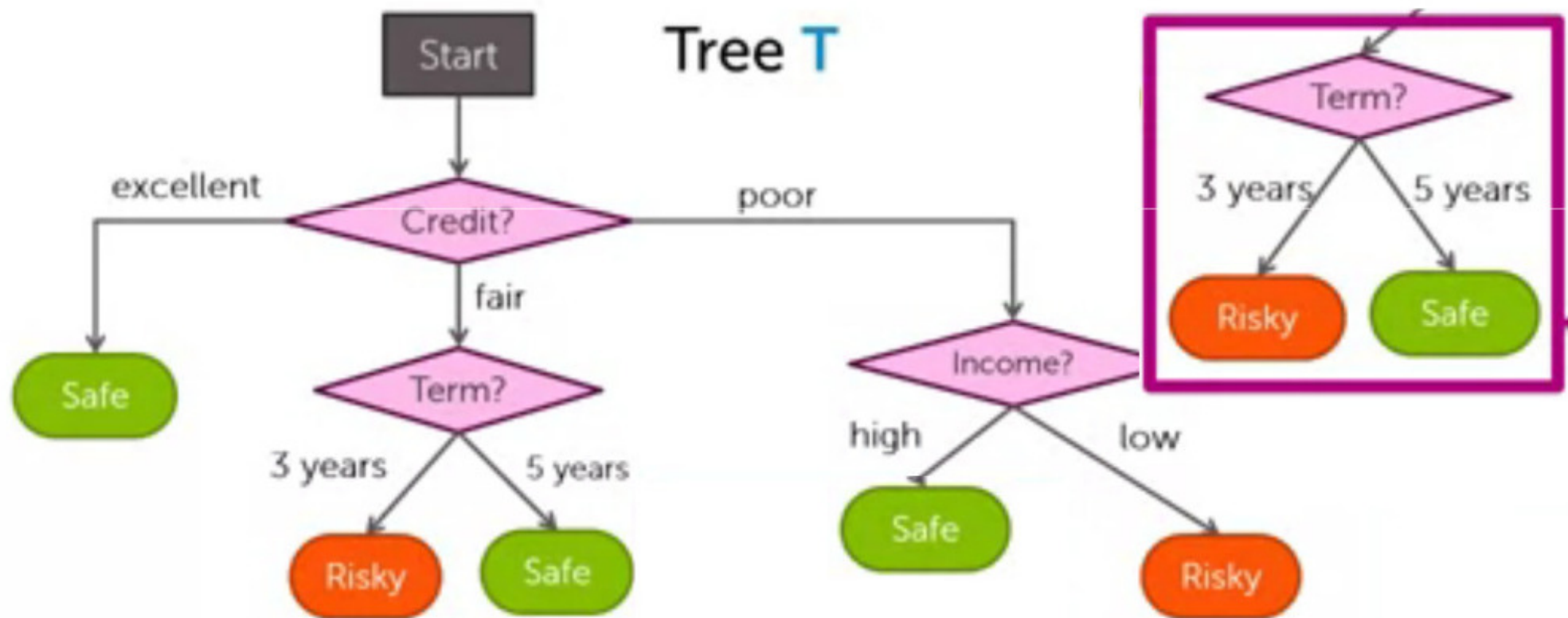
Pruning Tree Simulation-Before pruning subtree

Tree	Error	#Leave	Total	α
T	0.25	6	$E(T)=0.25+0.3 \times 6=0.43$	0.3



Pruning Tree Simulation-After pruning subtree

Tree	Error	#Leave	Total	α
T	0.26	5	$E(T)=0.26+5*0.3=0.41$	0.3



Classification Tree

- Task of growing a classification tree is quite similar to the task of a regression tree.
- Regression uses
 - Recursive binary splitting and RSS
- Classification uses
 - Recursive binary splitting and classification error rate
 - Assign an observation to most commonly occurring class of training observations in that region
 - Classification error rate
 - The fraction of the training observations in that region that do not belong to the most common class

Classification Error Rate

$$E = 1 - \max_k (\hat{p}_{mk}).$$

$E=1-P(m=k)$

- \hat{p}_{mk} represents the proportion of training observations in the m th region that are from the k th class.

However, classification error is not sufficiently sensitive for tree-growing, and in practice two other measures are preferable.

General Algorithm

Example

There are $2^6 \times 3^2 \times 4^2 = 9,216$ possible combinations of values for the input attributes. But we are given the correct output for only 12 of them; each of the other 9,204 could be either true or false; we don't know.

Example	Input Attributes										Output
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>WillWait</i>
x₁	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>0–10</i>	<i>y₁ = Yes</i>
x₂	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>30–60</i>	<i>y₂ = No</i>
x₃	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Some</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	<i>y₃ = Yes</i>
x₄	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Thai</i>	<i>10–30</i>	<i>y₄ = Yes</i>
x₅	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>>60</i>	<i>y₅ = No</i>
x₆	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Italian</i>	<i>0–10</i>	<i>y₆ = Yes</i>
x₇	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	<i>y₇ = No</i>
x₈	<i>No</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Thai</i>	<i>0–10</i>	<i>y₈ = Yes</i>
x₉	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>>60</i>	<i>y₉ = No</i>
x₁₀	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>Italian</i>	<i>10–30</i>	<i>y₁₀ = No</i>
x₁₁	<i>No</i>	<i>No</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>0–10</i>	<i>y₁₁ = No</i>
x₁₂	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>30–60</i>	<i>y₁₂ = Yes</i>

Objective: find the smallest tree that is consistent with the inputted data set

```
function LEARN-DECISION-TREE(examples, attributes, parent_examples) returns a tree
    if examples is empty then return PLURALITY-VALUE(parent_examples)
    else if all examples have the same classification then return the classification
    else if attributes is empty then return PLURALITY-VALUE(examples)
    else
         $A \leftarrow \operatorname{argmax}_{a \in \text{attributes}} \text{IMPORTANCE}(a, \text{examples})$ 
        tree  $\leftarrow$  a new decision tree with root test A
        for each value v of A do
            exs  $\leftarrow \{e : e \in \text{examples} \text{ and } e.A = v\}$ 
            subtree  $\leftarrow$  LEARN-DECISION-TREE(exs, attributes – A, examples)
            add a branch to tree with label (A = v) and subtree subtree
    return tree
```

- Unfortunately, it is intractable to find a guaranteed smallest consistent tree.
- But applying **some simple heuristics**, find one that is close to the smallest.
- LEARN-DECISION-TREE algorithm adopts **a greedy divide-and-conquer strategy**: always test **the most important attribute first**, then recursively solve the smaller subproblems that are defined by the possible results of the test.
- By “most important attribute,” means **the one that makes the most difference to the classification of an example**.

- The decision tree learning algorithm chooses the attribute with the **highest IMPORTANCE**.
- How to measure importance?
 - using the notion of **information gain**, which is **defined in terms of entropy**, which is the fundamental quantity in information theory.

Entropy

Entropy is a **measure of the uncertainty** of a random variable; the more information, the less entropy.

- $Entropy(T)$ = expected number of bits needed to encode class (\oplus or \ominus) of randomly drawn member of T (under the optimal, shortest-length code)
- Why?
In Information theory: optimal length code assigns $-\log_2 p$ bits to message having probability p .
- So, expected number of bits to encode \oplus or \ominus of random member of T :

Bernoulli distribution

$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

$$Entropy(T) = -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$



To know the outcome of the event, need to decrease uncertainty to 0 (i.e. certainty to 1). If the probability of some event A is p , knowing its outcome means **decreasing the uncertainty by the factor of $1/p$** .

So, we need $\lg(1/p)$ number of bits to know about the event, which is equal to $-\lg(p)$.

Cross-Entropy

- The average number of bits needed to know about the event is different from the average number of bits used to transfer the information.
- Cross-entropy is the average number of bits used to transfer the information. The cross-entropy is always greater than or equal to the entropy.

Let's take a probability distribution with four possible outcomes — with probabilities of 0.5, 0.25, 0.125, and 0.125. If we use two bits to transfer this information, the cross entropy becomes 2. Wait, what is the entropy in this case?

$$\begin{aligned}\text{Entropy} &= 0.5 * \lg(2) + 0.25 * \lg(4) + 0.125 * \lg(8) + 0.125 * \lg(8) \\ &= 0.5 + 0.5 + 0.375 + 0.375 = 1.75\end{aligned}$$

The entropy in this case is 1.75 but we used 2 bits to transfer this information, so cross entropy = 2.

This difference between cross-entropy and entropy is called **KL Divergence**.

Attribute Selection Measure 1: Information Gain

- Test attributes are selected based on a heuristics or statistical measure (e.g., **information gain**) Used in ID3
- Suppose we take the training examples in T and split T into subsets based on one of the attributes A .
- Let $Gain(T, A)$ = expected reduction in entropy due to splitting on A . It is the difference between entropies before splitting and after splitting on attribute A .

$$Gain(T, A) \equiv Entropy(T) - \sum_{v \in Values(A)} \frac{|T_v|}{|T|} Entropy(T_v)$$

Example: Play Tennis Yes/Not

Suppose we are given a table of training examples. Here *PlayTennis* is the concept to be learned.

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

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

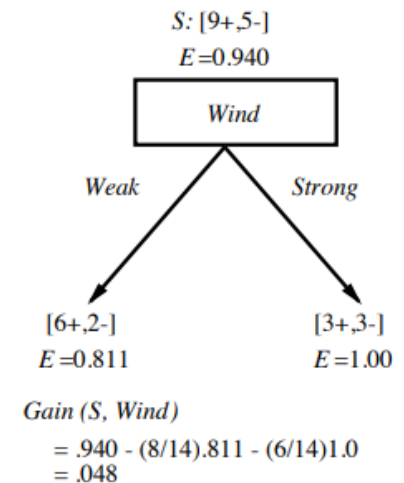
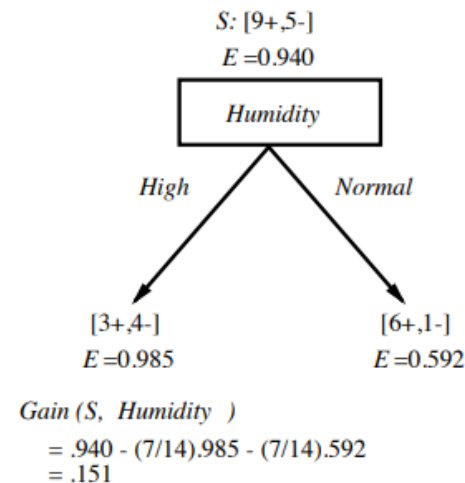
$$\text{Gain}(T, \text{Humidity}) = 0.151$$

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

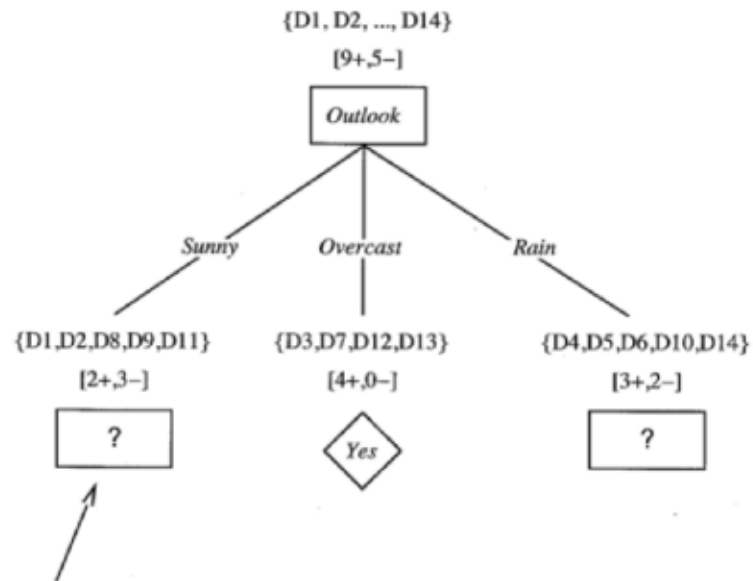
$$\text{Gain}(T, \text{Temperature}) = 0.029$$

Choose *Outlook* as the root attribute since it gives the most information gain.

Which attribute is the best classifier?



Selecting Next Attribute



Which attribute should be tested here?

$$T_{\text{Sunny}} = \{D1, D2, D8, D9, D11\}$$

$$\text{Gain}(T_{\text{sunny}}, \text{Humidity}) = .970 - (3/5)0.0 - (2/5)0.0 = .970$$

$$\text{Gain}(T_{\text{sunny}}, \text{Temperature}) = .970 - (2/5)0.0 - (1/5)0.0 = .570 \quad -(2/5(-0.5\log(0.5)-0.5\log(0.5)))=-0.4) = .570$$

$$\text{Gain}(T_{\text{sunny}}, \text{Wind}) = .970 - (2/5)1.0 - (3/5)0.918 = .019$$

	Temp	Hum	Wind	Play
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

$$H(\text{Sunny}) = -2/5 \cdot \log 2/5 - 3/5 \log 3/5 = -0.4 \log 0.4 - 0.6 \log 0.6 = 0.53 + 0.44 = .97$$

$$\log_2 x = \log_{10} x / \log_{10} 2$$

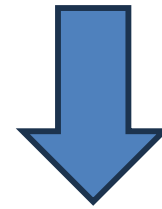
Choose *Humidity* as next attribute under *Sunny* because it gives most information gain.

Attribute Selection Measure 2: Gain Ratio

- Information gain measure is biased towards attributes with a large number of values **The highest number of distinct values, student ID**
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain):

$$SplitInfo_A(D) = -\sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

- $GainRatio(A) = Gain(A)/SplitInfo(A)$



Adjusts Information Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized!

Super attributes: Such an attribute will split into as many partitions as the number of values and each partition would be impure i.e. information gain would be highest and entropy would be zero which is not good for training a machine learning model. It would lead to overfitting the model.

Example: Gain Ratio

$$\text{GainRatio}(T,X) = \frac{\text{Gain}(T,X)}{\text{SplitInformation}(T,X)}$$

$$\text{Split}(T,X) = -\sum_{c \in A} P(c) \log_2 P(c)$$

		Play Golf		
		Yes	No	total
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
		Gain = 0.247		

		Play Golf		
		Yes	No	total
ID	id1	1	0	1
	id2	0	1	1
	id3	1	0	1
	id4	1	0	1
	id5	0	1	1
	id6	0	1	1
	id7	1	0	1
	id8	1	0	1
	id9	0	1	1
	id10	1	0	1
	id11	1	0	1
	id12	0	1	1
	id13	1	0	1
	id14	1	0	1

Entropy (Play Golf, ID) = 0



Gain (Play Golf, ID) = 0.94



Split (Play Golf, ID) = 3.81



Gain Ratio (Play Golf, ID) = 0.94 / 3.81 = 0.25

$$\text{Split}(\text{Play}, \text{Outlook}) = -(5/14 * \log_2(5/14) + 4/14 * \log_2(4/14) + 5/14 * \log_2(5/14))$$


$$= 1.577$$

$$\text{Gain Ratio}(\text{Play}, \text{Outlook}) = 0.247 / 1.577 = 0.156$$

Attribute Selection Measure 3: Gini Index

- If a data set T contains examples from n classes, gini index, $\text{gini}(T)$ is defined as

$$\text{gini}(T) = 1 - \sum_{j=1}^n p_j^2$$

where p_j is the relative frequency of class j in T . $\text{gini}(T)$ is minimized if the classes in T are skewed.  **the proportion of a class.**

- After splitting T into two subsets T_1 and T_2 with sizes N_1 and N_2 , the gini index of the split data is defined as

$$\text{gini}_{\text{split}}(T) = \frac{N_1}{N} \text{gini}(T_1) + \frac{N_2}{N} \text{gini}(T_2)$$

- The attribute providing smallest $\text{gini}_{\text{split}}(T)$ is chosen to split the node.

Confusion Matrix-Binary Classification

Sensitivity = How many positive records are correctly predicted?

Specificity = How many negative records were correctly predicted?

Precision = How many correct predictions out of all predictions?

Recall- How many actual records were correctly predicted?

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.

		Actual class (ground truth)		
Total (n)=100		Dog (Positive)	Not a Dog (Negative)	
Predicted class	Dog (Positive)	15 (TP)	20 (FP, Type I Error)	Precision =TP/(TP+FP) =0.42
	Not a Dog (Negative)	5 (FN, Type II Error)	60 (TN)	
	Accuracy =(TP+TN)/Total =0.75	Sensitivity, Recall, TPR =TP/(TP+FN) = 0.75	FPR = FP/(FP+TN) =0.25	F1 Score =2*(Precision*Recall) / (Precision+Recall) =0.53
	Error Rate =(FP+FN)/Total =0.25	Miss Rate, FNR =FN/(TP+FN) =0.25	Specificity, TNR = TN/(FP+TN) =0.75	

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- *true positive (TP)*: sample and classification are positive
- *false positive (FP)*: sample is negative, classification is positive
- *false negative (FN)*: sample is positive, classification is negative
- *true negative (TN)*: sample and classification are negative

Confusion Matrix-Multi-Class Classification

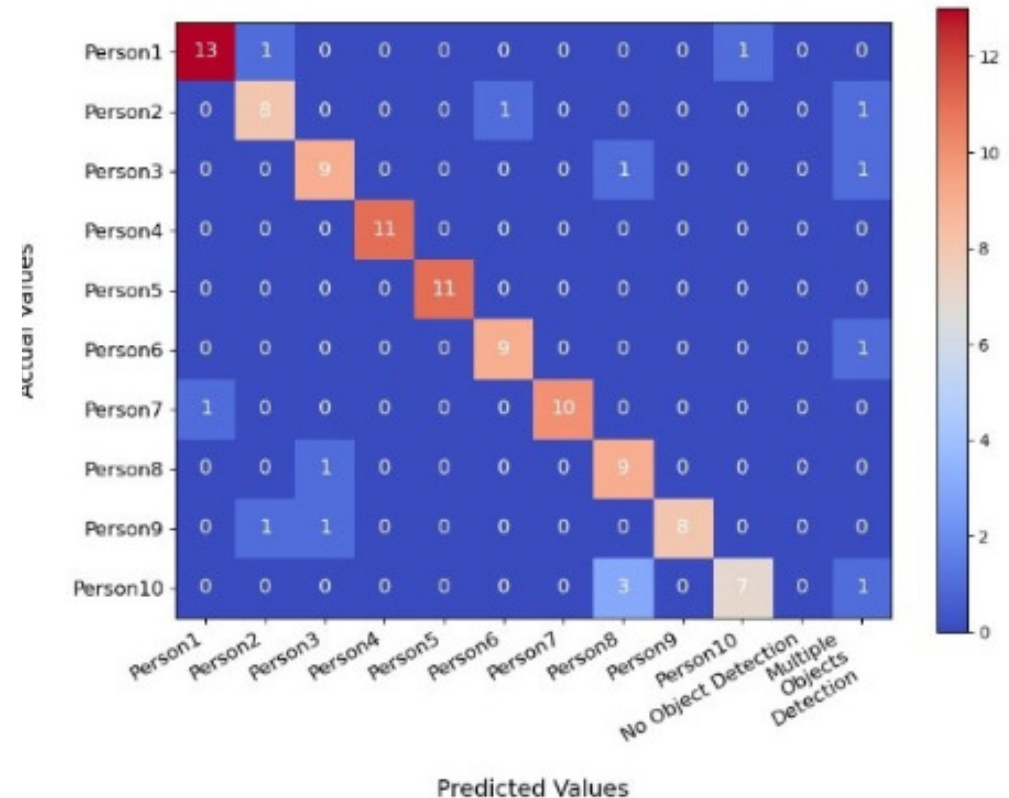
The 3×3 confusion matrix multiclass for the IRIS dataset is as below:

		Predicted Values				Setosa	Versicolor	Virginica
Actual Values		Setosa	Versicolor	Virginica	TP	C1=16	C5=17	C9=11
					FP	C4+C7 0	C2+C8 0	C3+C6 1
	Setosa	16 (cell 1)	0 (cell 2)	0 (cell 3)	FN	C2+C3 0	C4+C6 1	C7+C8 0
	Versicolor	0 (cell 4)	17 (cell 5)	1 (cell 6)	TN	C5+C6+C8 +C9 29	C1+C3+C7+ C9 27	C1+C2+C4 +C5 33
	Virginica	0 (cell 7)	0 (cell 8)	11 (cell 9)				

- FN: The False-negative value for a class will be the sum of values of corresponding rows except for the TP value.
- FP: The False-positive value for a class will be the sum of values of the corresponding column except for the TP value.
- TN: The True-negative value for a class will be the sum of the values of all columns and rows except the values of that class that we are calculating the values for.
- TP: The True-positive value is where the actual value and predicted value are the same.

Confusion Matrix-Multi-Class Classification

Class	Precision	Recall	F1-Score
Person1	0.93	0.87	0.90
Person2	0.80	0.80	0.80
Person3	0.82	0.82	0.82
Person4	1.00	1.00	1.00
Person5	1.00	1.00	1.00
Person6	0.90	0.90	0.90
Person7	1.00	0.91	0.95
Person8	0.69	0.90	0.78
Person9	1.00	0.80	0.89
Person10	0.88	0.64	0.74



The x-axis defines the model's expected values, and the y-axis defines the actual values. The 10x12(MxN) matrix structure was used to build this matrix. According to our model, some images have numerous objects detected while others have no objects detected at all. As a result, the matrix has two additional columns labeled Multiple obj detection and No Object Detection, respectively.

In the figure, the number of nonobject detection images is zero, but four images are identified as multiples of person for Person2, Person3, Person6, and Person10.

Confusion Matrix-Multi-Class Classification

- **Precision:** measures the model's ability to identify instances of a particular class correctly.

$$\text{Precision}_{\text{Class A}} = \frac{TP_{\text{Class A}}}{TP_{\text{Class A}} + FP_{\text{Class A}}}$$

- **Recall:** is the fraction of instances in a class that the model correctly classified out of all instances in that class.

$$\text{Recall}_{\text{Class A}} = \frac{TP_{\text{Class A}}}{TP_{\text{Class A}} + FN_{\text{Class A}}}$$

- **Accuracy** measures the proportion of correctly classified cases from the total number of objects in the dataset.

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

F β score

The F β score is a generalized version of the F1 score. It computes the harmonic mean, just like an F1 score, but with a priority given to either precision or recall. “ β ” represents the weighting coefficient (a hyperparameter set by the user, which is always greater than 0). Mathematically, it is represented as follows:

$$\begin{aligned} \text{F}\beta \text{ Score} &= \frac{1 + \beta^2}{\frac{1}{\text{Precision}} + \frac{\beta^2}{\text{Recall}}} \\ &= \frac{(1 + \beta^2) \times \text{Precision} \times \text{Recall}}{(\beta^2 \times \text{Precision}) + \text{Recall}} \end{aligned}$$

We talk about the F1 score in cases where β is 1. A β value greater than 1 favors the recall metric, while values lower than 1 favor the precision metric. F0.5 and F2 are the most commonly used measures other than F1 scores.

The F β score is useful when we want to prioritize one measure while preserving results from the other measure.

For example, in the case of COVID-19 detection, False Negative results are detrimental—since a COVID positive patient is diagnosed as COVID negative, leading to the spread of the disease. In this case, the F2 measure is more useful to minimize the False Negatives while also trying to keep the precision score as high as possible. In other cases, it might be necessary to reduce the False Positives, where a lower β value (like an F0.5 score) is desired.

Macro-averaging VS Micro-averaging

- **Macro-Averaging:** Average the precision and recall across all classes to get the final macro-averaged precision and recall scores.
- A macro-average will compute the metric independently for each class and then take the average (hence treating all classes equally).
- **Micro-averaging:** a micro-average will aggregate the contributions of all classes to compute the average metric.
- In a multi-class classification setup, micro-average is preferable if you suspect a class imbalance.

$$\text{Precision}_{\text{Macro-average}} = \frac{\text{Precision}_{\text{Class A}} + \text{Precision}_{\text{Class B}} + \dots \text{Precision}_{\text{Class N}}}{N}$$

$$\text{Recall}_{\text{Macro-average}} = \frac{\text{Recall}_{\text{Class A}} + \text{Recall}_{\text{Class B}} + \dots \text{Recall}_{\text{Class N}}}{N}$$

$$\text{Precision}_{\text{Micro-average}} = \frac{TP_A + TP_B + \dots TP_N}{TP_A + FP_A + TP_B + FP_B + \dots TP_N + FP_N}$$

$$\text{Recall}_{\text{Micro-average}} = \frac{TP_A + TP_B + \dots TP_N}{TP_A + FN_A + TP_B + FN_B + \dots TP_N + FN_N}$$

It gives equal importance to every individual prediction, regardless of the class distribution in the dataset. This makes it a useful metric for understanding the model's performance on a global scale, particularly when classes are well-balanced.

Receiver Operator Characteristics (ROC)

- Evaluating different machine learning configurations
 - May have dozens, hundreds, or thousands of confusion matrices (diff classification threshold)
 - Tedious to review-summarize with a receiver operator characteristic (ROC) curve.

True Positive Rate (TPR) is a synonym for recall and is

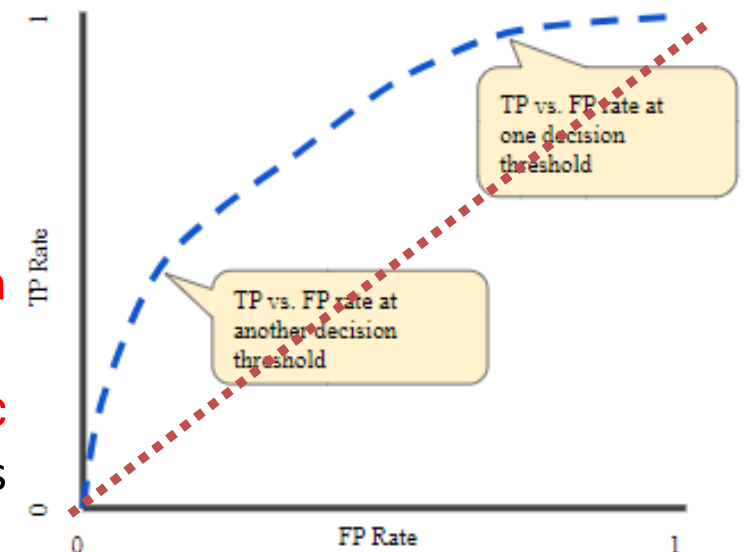
$$TPR = \frac{TP}{TP + FN}$$

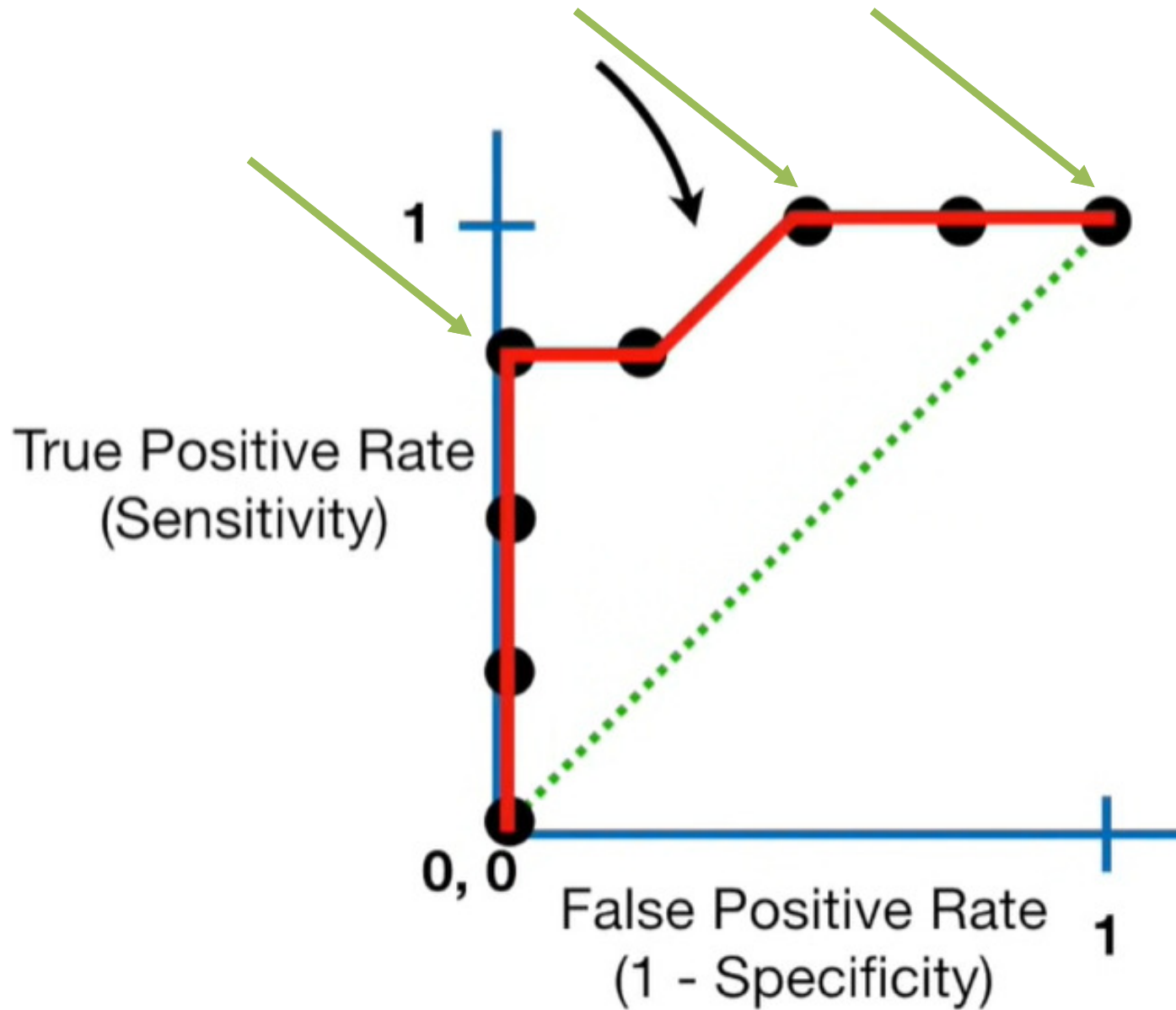
False Positive Rate (FPR) is defined as :

$$FPR = \frac{FP}{FP + TN}$$

A ROC curve plots TPR vs. FPR at different **classification thresholds**.

Lowering the classification threshold a **logistic regression models** more items as positive, thus increasing both False Positives and True Positives.

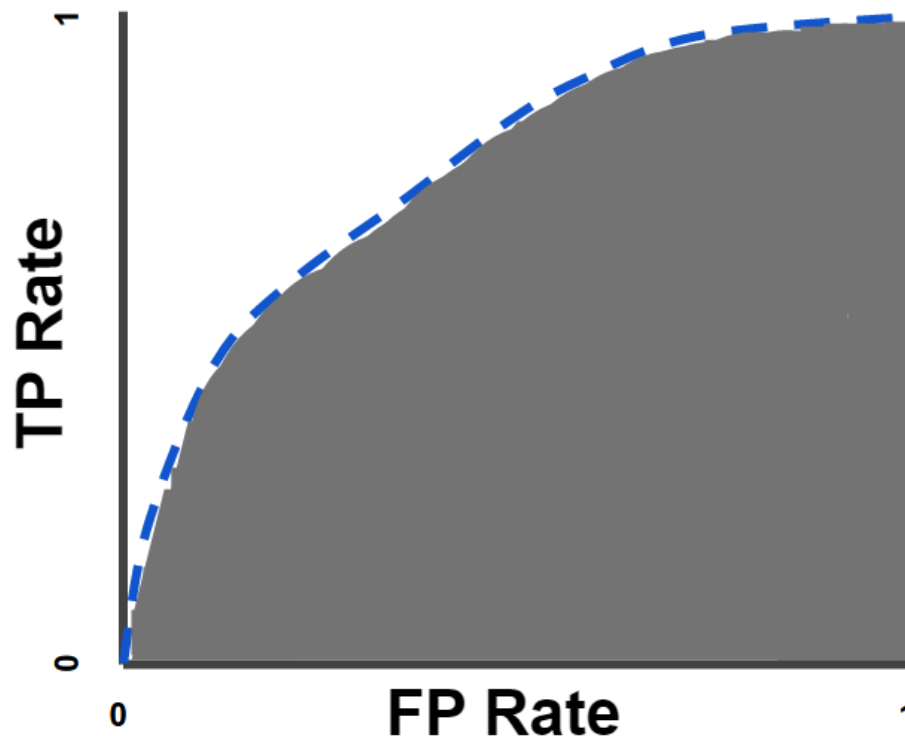




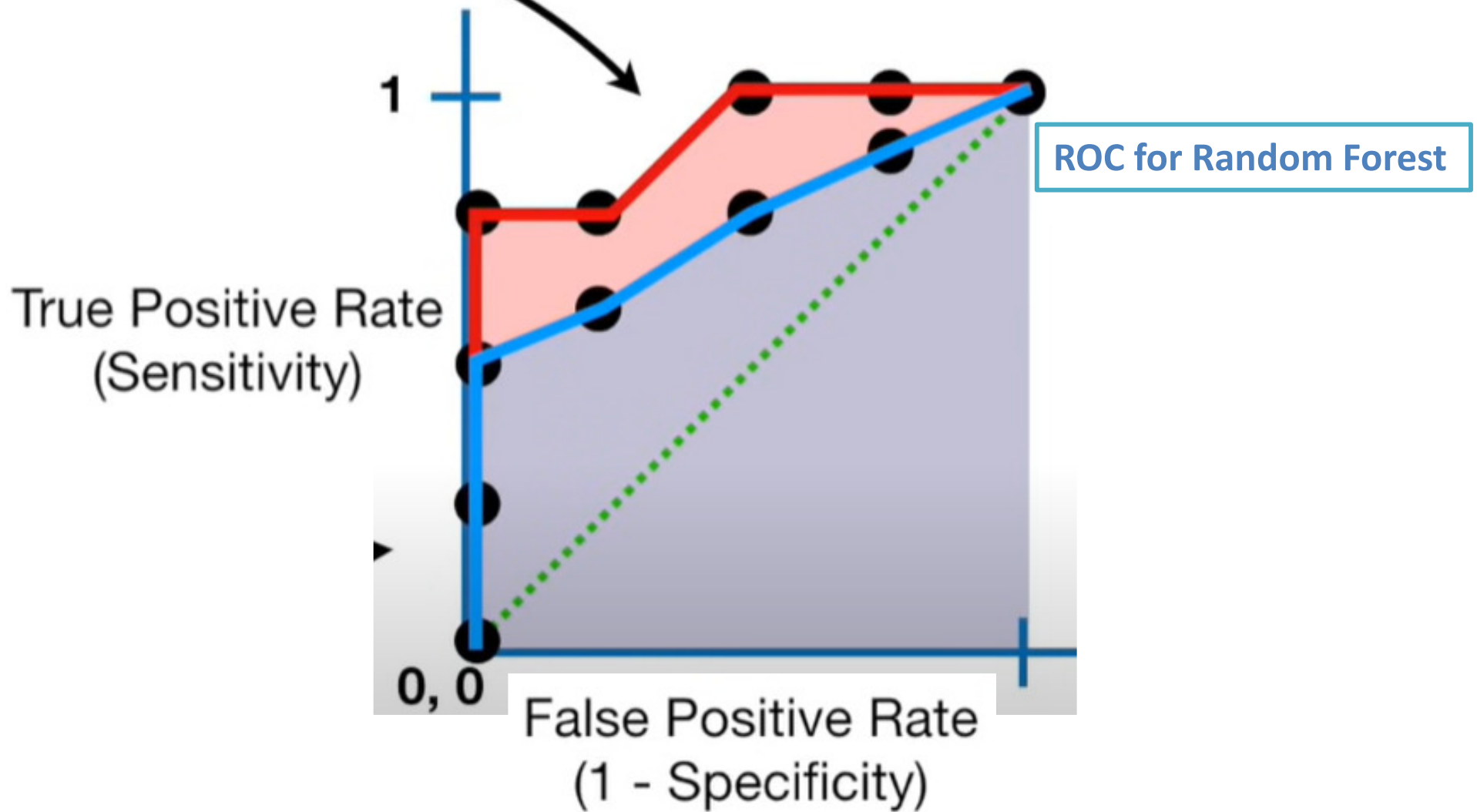
- **Which Threshold is Better?**
 - From ROC we can determine without creating confusion matrix

Area Under the ROC Curve (AUC)

- To compute the points in a ROC curve, we could evaluate a logistic regression model many times **with different classification thresholds, which would be inefficient.**
- Fortunately, an efficient, sorting-based algorithm can provide this information to us, called AUC.
- AUC stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the ROC as a whole curve (think integral calculus) from (0,0) to (1,1).



ROC for Logistic Regression



- **Which ROC is Better?**

- From ROC we can determine the best classification model

- People often replace the **False Positive Rate** with **Precision**.

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

This is because **Precision** does not include the number of **True Negatives** in its calculation, and is not effected by the imbalance.

Example

ID	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Sunny	>25	High	Weak	No
2	Sunny	>25	High	Strong	No
3	Overcast	>25	High	Weak	Yes
4	Rain	15-25	High	Weak	Yes
5	Rain	<15	Normal	Weak	Yes
6	Rain	<15	Normal	Strong	No
7	Overcast	<15	Normal	Strong	Yes
8	Sunny	15-25	High	Weak	No
9	Sunny	<15	Normal	Weak	Yes
10	Rain	15-25	Normal	Weak	Yes
11	Sunny	15-25	Normal	Strong	Yes
12	Overcast	15-25	High	Strong	Yes
13	Overcast	>25	Normal	Weak	Yes
14	Rain	15-25	High	Strong	No

Tree induction example

■ Entropy of data S

$$\text{Info}(S) = -9/14(\log_2(9/14)) - 5/14(\log_2(5/14)) = 0.94$$

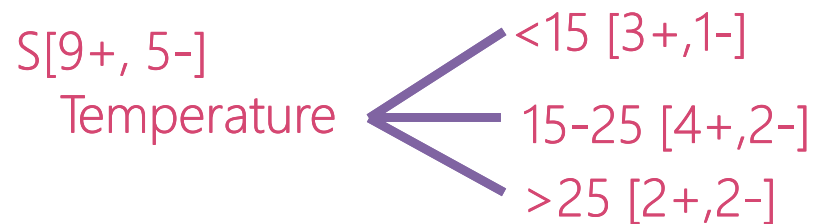
■ Split data by attribute Outlook



$$\begin{aligned}\text{Gain}(\text{Outlook}) &= 0.94 - 5/14[-2/5(\log_2(2/5)) - 3/5(\log_2(3/5))] \\ &\quad - 4/14[-4/4(\log_2(4/4)) - 0/4(\log_2(0/4))] \\ &\quad - 5/14[-3/5(\log_2(3/5)) - 2/5(\log_2(2/5))] \\ &= 0.94 - 0.69 = 0.25\end{aligned}$$

Tree induction example

- Split data by attribute **Temperature**



$$\begin{aligned}\text{Gain}(\text{Temperature}) &= 0.94 - 4/14[-3/4(\log_2(3/4)) - 1/4(\log_2(1/4))] \\ &\quad - 6/14[-4/6(\log_2(4/6)) - 2/6(\log_2(2/6))] \\ &\quad - 4/14[-2/4(\log_2(2/4)) - 2/4(\log_2(2/4))] \\ &= 0.94 - 0.91 = 0.03\end{aligned}$$

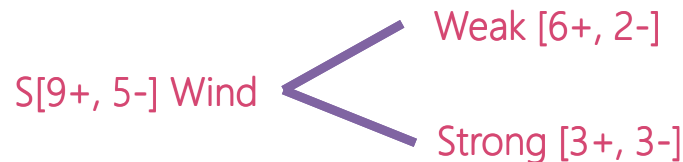
Tree induction example

- Split data by attribute Humidity



$$\begin{aligned}\text{Gain}(\text{Humidity}) &= 0.94 - 7/14[-3/7(\log_2(3/7)) - 4/7(\log_2(4/7))] \\ &\quad - 7/14[-6/7(\log_2(6/7)) - 1/7(\log_2(1/7))] \\ &= 0.94 - 0.79 = 0.15\end{aligned}$$

- Split data by attribute Wind

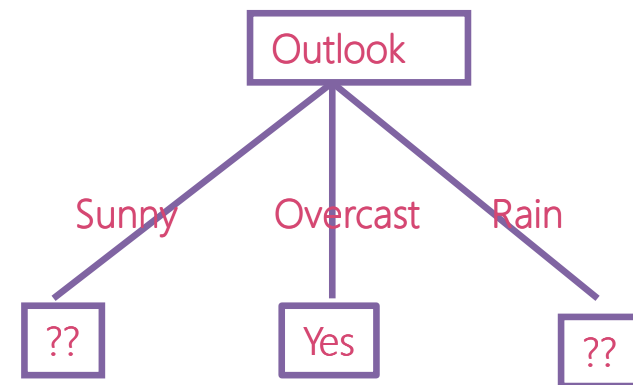


$$\begin{aligned}\text{Gain}(\text{Wind}) &= 0.94 - 8/14[-6/8(\log_2(6/8)) - 2/8(\log_2(2/8))] \\ &\quad - 6/14[-3/6(\log_2(3/6)) - 3/6(\log_2(3/6))] \\ &= 0.94 - 0.89 = 0.05\end{aligned}$$

Tree induction example

Outlook	Temperature	Humidity	Wind	Play Tennis
Sunny	>25	High	Weak	No
Sunny	>25	High	Strong	No
Overcast	>25	High	Weak	Yes
Rain	15-25	High	Weak	Yes
Rain	<15	Normal	Weak	Yes
Rain	<15	Normal	Strong	No
Overcast	<15	Normal	Strong	Yes
Sunny	15-25	High	Weak	No
Sunny	<15	Normal	Weak	Yes
Rain	15-25	Normal	Weak	Yes
Sunny	15-25	Normal	Strong	Yes
Overcast	15-25	High	Strong	Yes
Overcast	>25	Normal	Weak	Yes
Rain	15-25	High	Strong	No

Gain(Outlook) = 0.25
Gain(Temperature) = 0.03
Gain(Humidity) = 0.15
Gain(Wind) = 0.05



■ Entropy of branch Sunny

$$\text{Info}(\text{Sunny}) = -2/5(\log_2(2/5)) - 3/5(\log_2(3/5)) = 0.97$$

■ Split Sunny branch by attribute Temperature

Sunny[2+,3-]
Temperature

$<15 [1+,0-]$
 $15-25 [1+,1-]$
 $>25 [0+,2-]$

Gain(Temperature)
 $= 0.97$
 $- 1/5[-1/1(\log_2(1/1)) - 0/1(\log_2(0/1))]$
 $- 2/5[-1/2(\log_2(1/2)) - 1/2(\log_2(1/2))]$
 $- 2/5[-0/2(\log_2(0/2)) - 2/2(\log_2(2/2))]$
 $= 0.97 - 0.4 = 0.57$

■ Split Sunny branch by attribute Humidity

Sunny[2+,3-]
Humidity

$\text{High } [0+,3-]$
 $\text{Normal } [2+, 0-]$

Gain(Humidity)
 $= 0.97$
 $- 3/5[-0/3(\log_2(0/3)) - 3/3(\log_2(3/3))]$
 $- 2/5[-2/2(\log_2(2/2)) - 0/2(\log_2(0/2))]$
 $= 0.97 - 0 = \underline{0.97}$

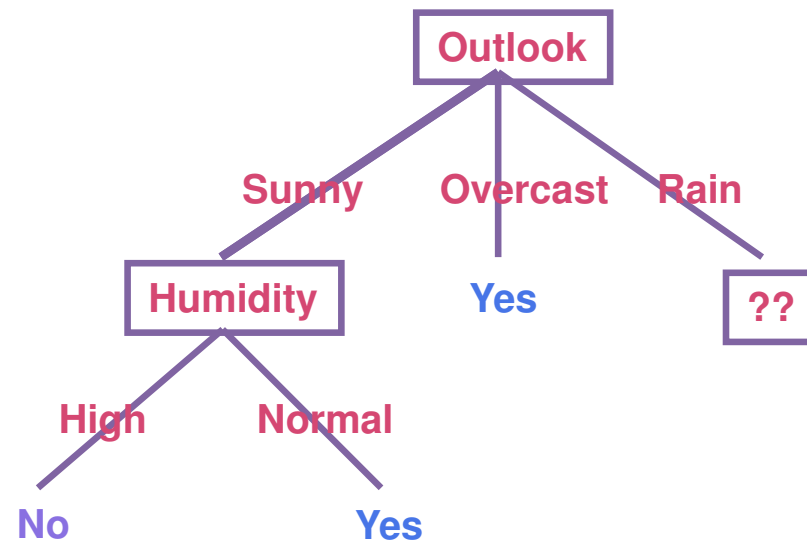
■ Split Sunny branch by attribute Wind

Sunny[2+, 3-]
Wind

$\text{Weak } [1+, 2-]$
 $\text{Strong } [1+, 1-]$

Gain(Wind)
 $= 0.97$
 $- 3/5[-1/3(\log_2(1/3)) - 2/3(\log_2(2/3))]$
 $- 2/5[-1/2(\log_2(1/2)) - 1/2(\log_2(1/2))]$
 $= 0.97 - 0.95 = 0.02$

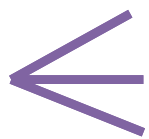
Tree induction example



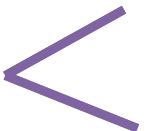
■ Entropy of branch Rain

$$\text{Info}(\text{Rain}) = -3/5(\log_2(3/5)) - 2/5(\log_2(2/5)) = 0.97$$

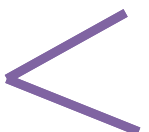
■ Split Rain branch by attribute Temperature

Rain[3+,2-] Temperature		<15 [1+,1-]	Gain(Outlook) = 0.97 - 2/5[-1/2(log ₂ (1/2))-1/2(log ₂ (1/2))] - 3/5[-2/3(log ₂ (2/3))-1/3(log ₂ (1/3))] - 0/5[-0/0(log ₂ (0/0))-0/0(log ₂ (0/0))] = 0.97 - 0.95 = 0.02
		15-25 [2+,1-]	
		>25 [0+,0-]	

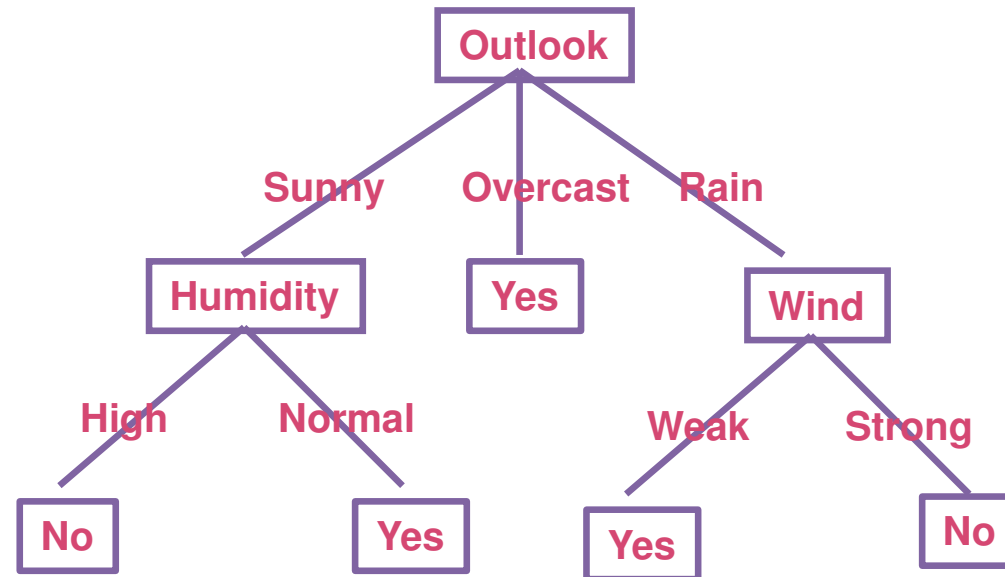
■ Split Rain branch by attribute Humidity

Rain[3+,2-] Humidity		High [1+,1-]	Gain(Humidity) = 0.97 - 2/5[-1/2(log ₂ (1/2))-1/2(log ₂ (1/2))] - 3/5[-2/3(log ₂ (2/3))-1/3(log ₂ (1/3))] = 0.97 - 0.95 = 0.02
		Normal [2+, 1-]	

■ Split Rain branch by attribute Wind

Rain[3+,2-] Wind		Weak [3+, 0-]	Gain(Wind) = 0.97 - 3/5[-3/3(log ₂ (3/3))-0/3(log ₂ (0/3))] - 2/5[-0/2(log ₂ (0/2))-2/2(log ₂ (2/2))] = 0.97 - 0 = <u>0.97</u>
		Strong [0+, 2-]	

Tree induction example



Example: Gini Index

Resp srl no	Target variable	Predictor variable	Predictor variable	Predictor variable
	Exam Result	Other online courses	Student background	Working Status
1	Pass	Y	Maths	NW
2	Fail	N	Maths	W
3	Fail	y	Maths	W
4	Pass	Y	CS	NW
5	Fail	N	Other	W
6	Fail	Y	Other	W
7	Pass	Y	Maths	NW
8	Pass	Y	CS	NW
9	Pass	n	Maths	W
10	Pass	n	CS	W
11	Pass	y	CS	W
12	Pass	n	Maths	NW
13	Fail	y	Other	W
14	Fail	n	Other	NW
15	Fail	n	Maths	W

Gini index for the root node for Student Background attribute.

$$Gini_{maths} = 1 - \left(\frac{4}{7}\right)^2 - \left(\frac{3}{7}\right)^2 = .4897$$

$$Gini_{CS} = 1 - \left(\frac{4}{4}\right)^2 - \left(\frac{0}{5}\right)^2 = 0$$

$$Gini_{CS} = 1 - \left(\frac{4}{4}\right)^2 - \left(\frac{0}{5}\right)^2 = 0$$

The overall Gini Index for this split

$$Gini_{bkgrd} = \frac{7}{15} * .4897 + \frac{4}{15} * 0 + \frac{4}{15} * 0 = .2286$$

The overall Gini Index for this split with the work status variable

$$Gini_{working} = 1 - \left(\frac{6}{9}\right)^2 - \left(\frac{3}{9}\right)^2 = .44$$

$$Gini_{notworking} = 1 - \left(\frac{5}{6}\right)^2 - (6)^2 = .278$$

$$Gini_{workstatus} = \frac{9}{15} * .44 + \frac{6}{15} * .278 = .378$$

The overall Gini Index for this split with the online variable

$$Gini_{online} = 1 - \left(\frac{5}{8}\right)^2 - \left(\frac{3}{8}\right)^2 = .4688$$

$$Gini_{notonline} = 1 - \left(\frac{3}{7}\right)^2 - \left(\frac{4}{7}\right)^2 = .4898$$

$$Gini_{online} = \frac{8}{15} * .4688 + \frac{7}{15} * .4898 = .479$$

The **Gini Index is lowest** for the Student Background variable.
Thus, we will pick this variable for the root node.