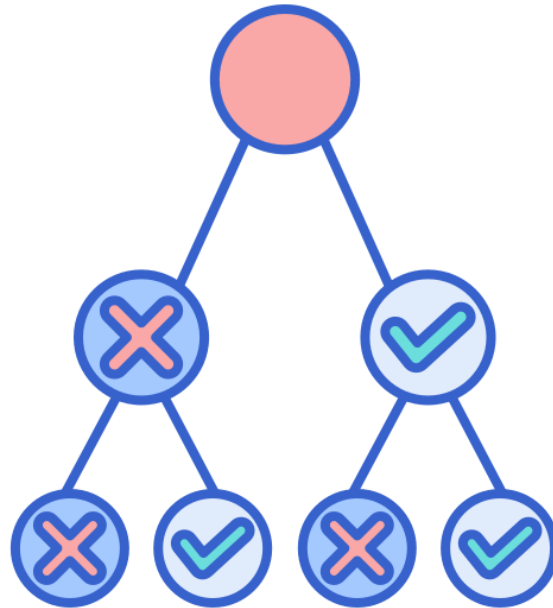


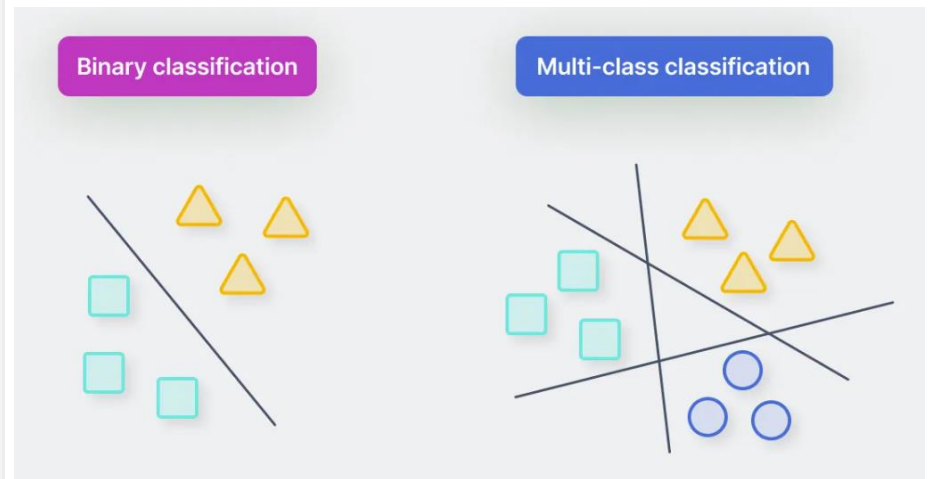
Classification using Decision Trees



Outline

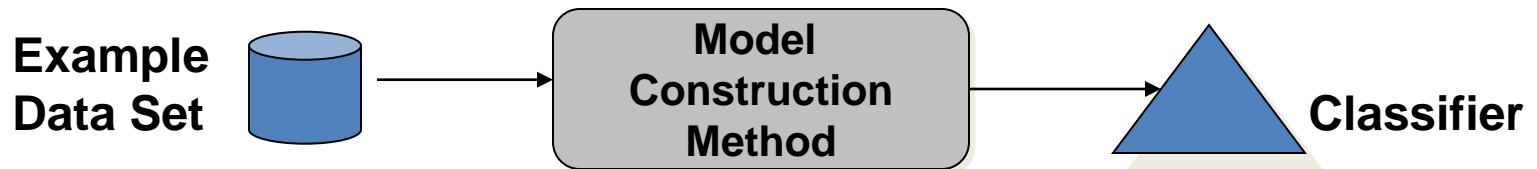
- Classification
- Decision Tree (DT)
- **DT Attribute Selection Measure (ASM)**
 - Classification Error Rate
 - Gini Impurity Index
 - Entropy
- Decision Boundaries

Classification

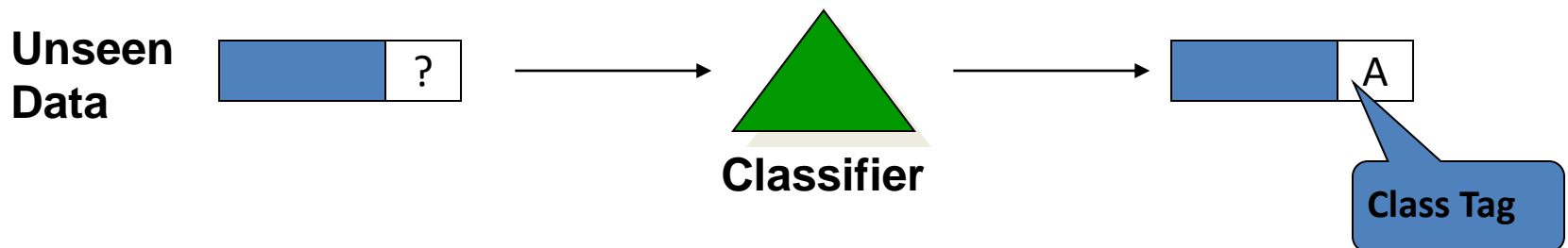


Classification Models

- Assign labels to objects (predicting classes)
- Two-Stage Process
 - Given a data set of **labeled** examples, use a classification method to train a classification model, known as **classifier**



- Given a trained classifier, classify a data record with unknown class to one of the pre-defined classes



Classification Examples

- **Spam Email Filter (binary classifier):** classify emails labeled as spam or not spam by learning patterns in the content, sender information, and other features
- **Sentiment Analysis (multi-class classifier):** classify media posts or product reviews as positive, negative, or neutral sentiments expressed by the author
- **Medical Diagnosis (binary):** a model trained on patient symptoms and medical history can classify whether a patient is likely to have a certain disease
- **Credit Risk Assessment (multi-class):** classify loan applicants as low, medium, or high risk based on factors such as credit score, income, and debt-to-income ratio
- **Image Recognition (multi-class):** a model can classify images of animals into different categories such as cats, dogs, or birds

Classification Algorithms

Include:

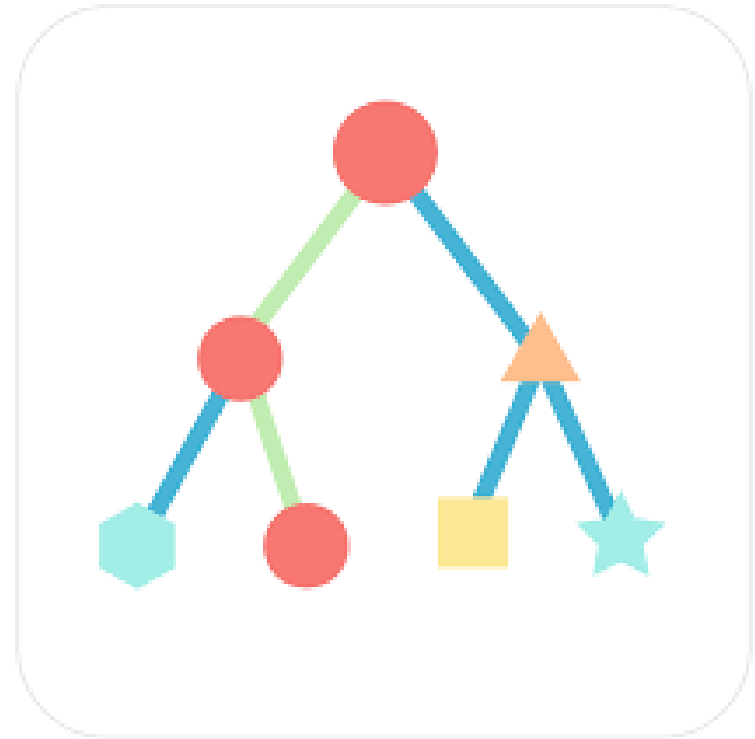
- Decision Trees & Random Forest
- Logistic Regression
- Naive Bayes
- K-Nearest Neighbors (KNN)
- Support Vector Machine (SVM)
- Artificial Neural Network (ANN)

ML Metrics: Influential Factors for a Good Model

- Accuracy
 - Estimated accuracy during development stage vs. actual accuracy during practical use
- Performance
 - Time taken for model construction (training time)
 - Time taken for the model to infer
- Interpretability
 - Ease of interpreting decisions by the model
 - Understanding and insight provided by the model
- Robustness:
 - Handling noise and missing values
- Scalability:
 - Ability to handle large datasets
- Other measures, e.g., decision tree size or compactness of rules

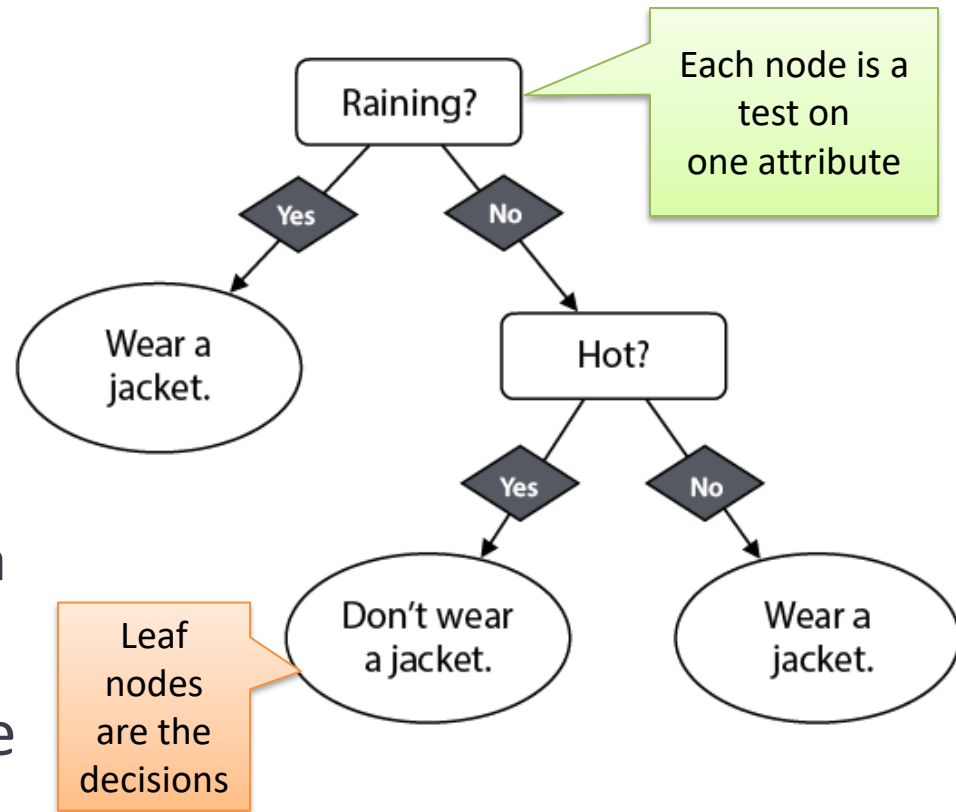


Decision Trees



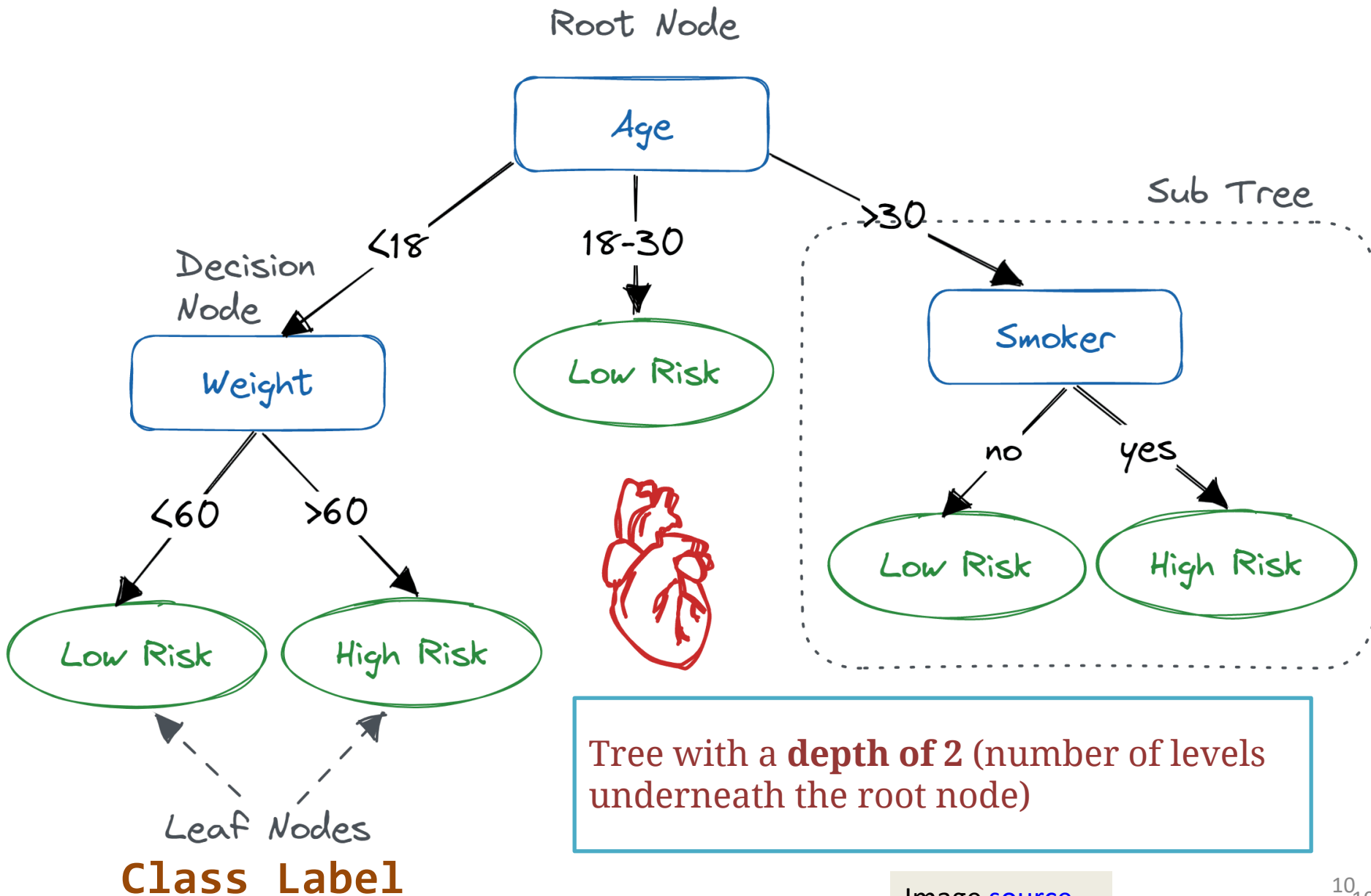
Should I wear jacket today?

- If it's raining, then wear a jacket
- If it's not, then we check the temperature:
 - If it is hot, then don't wear a jacket
 - But if it is cold, then wear a jacket
- The decision tree depicts the decision process, where the **decisions are made by traversing the tree from top to bottom**



We arrive to a decision (i.e., a class) by asking a series of questions

Decision Tree – Risk of heart attack



Decision Tree

- **Decision Tree (DT):** ML model based on yes-or-no questions and represented by a binary tree that describes the decision flow
 - The tree has a **root node**, **decision nodes**, **leaf nodes**, and **branches**
- Can be used:
 - When a series of questions (yes/no) are answered to arrive at a **classification** decision
 - E.g., Checklist of symptoms during a doctor's evaluation of a patient
 - When **interpretable** “if-then” conditions are preferred to mathematical models
 - E.g.: Financial decisions such as loan approval or fraud detection

Decision Tree for the Iris dataset: using all the four attributes

```
► X_iris = iris.data.values
y_iris = iris.target
tree_clf = DecisionTreeClassifier(max_depth=4, random_state=42)
tree_clf = tree_clf.fit(X_iris, y_iris)
```

```
► tree_clf.fit(X_iris, y_iris)
plt.figure(figsize=(10,8))
plot_tree(tree_clf, filled=True)
plt.title("Decision tree trained on all attributes")
plt.show()
```

Decision tree trained on all attributes



```
In [31]: ► print(tree_clf.predict([[.5, 1.5,5,7.5]]))
print(tree_clf.predict([[.5, 1.5,2.5,1.2]]))
print(tree_clf.predict([[5, 1.5,2,3]]))
```

```
In [31]: ► print(tree_clf.predict([[.5, 1.5,5,7.5]]))
print(tree_clf.predict([[.5, 1.5,2.5,1.2]]))
print(tree_clf.predict([[5, 1.5,2,3]]))
```

```
[2]
[1]
[0]
```

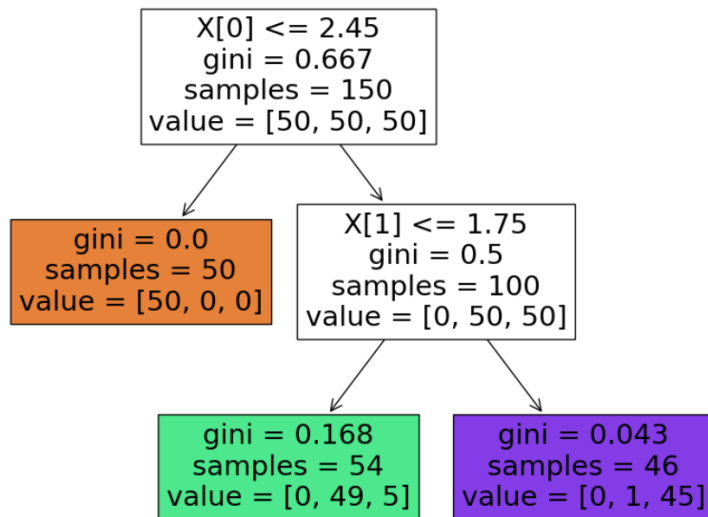


Decision Tree for the Iris dataset: using two attributes

```
In [15]: > iris = load_iris(as_frame=True)
columns_to_use = ["petal length (cm)", "petal width (cm)"]
X_iris = iris.data[columns_to_use].values
y_iris = iris.target
tree_clf = DecisionTreeClassifier(max_depth=2, random_state=42)
tree_clf = tree_clf.fit(X_iris, y_iris)
```

```
In [18]: > plt.figure(figsize=(10,8))
plot_tree(tree_clf, filled=True)
plt.title("Decision tree trained on two attributes")
plt.show()
```

Decision tree trained on two attributes



```
In [19]: > tree_clf.predict([[3, 2.5]])
```

```
Out[19]: array([2])
```

```
In [16]: > tree_clf.predict([[5, 1.5]])
```

```
Out[16]: array([1])
```

```
In [17]: > tree_clf.predict([[.5, 1.5]])
```

```
Out[17]: array([0])
```







```
In [20]: > r = export_text(tree_clf, feature_names=columns_to_use)
print(r)
```

```
|--- petal length (cm) <= 2.45
|   |--- class: 0
|--- petal length (cm) > 2.45
|   |--- petal width (cm) <= 1.75
|       |--- class: 1
|   |--- petal width (cm) > 1.75
|       |--- class: 2
```



App Recommendation System using a DT

What to recommend for?

Gender	Age	App
Female	15	
Female	25	
Male	32	
Female	35	
Male	12	
Male	14	



Female, 16 years old



Female, 30 years old









Male, 35 years old

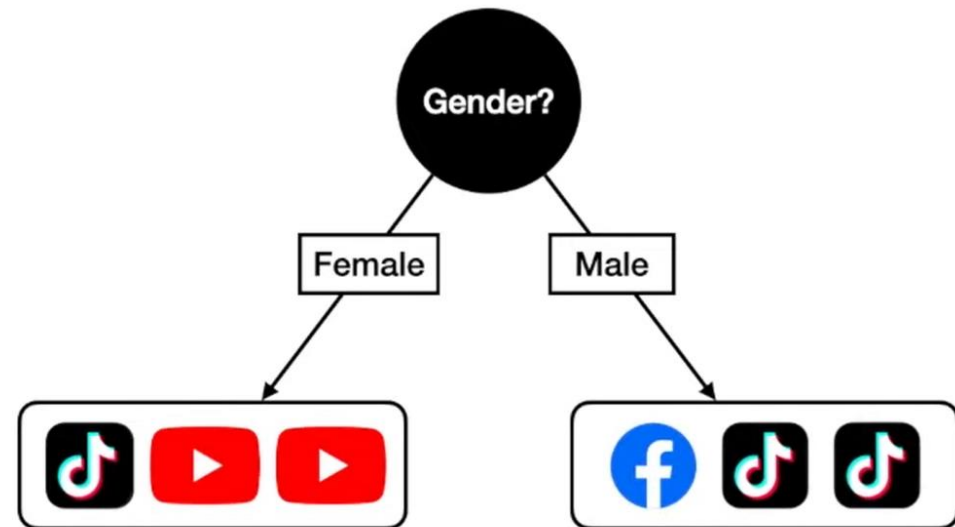


Which feature is more important?







- Which one of the two features (gender or age) **is more important** in determining the app to recommend?
 - This is the most important step in building a decision tree!
- Let's build Gender decision stump by Gender and another one by Age then compare them
 - In other words, **split** the data by gender then by age to compare them

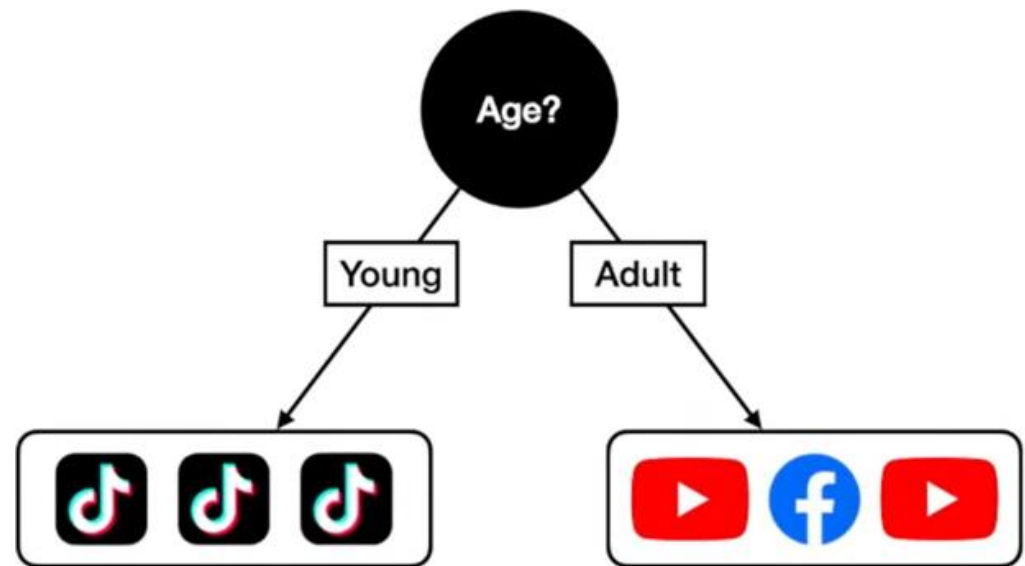
Splitting by Gender => Gender decision stump

Gender	App
Female	
Female	
Female	
Male	
Male	
Male	



Splitting by Age => Age decision stump

Age	App
Young	
Young	
Young	
Adult	
Adult	
Adult	

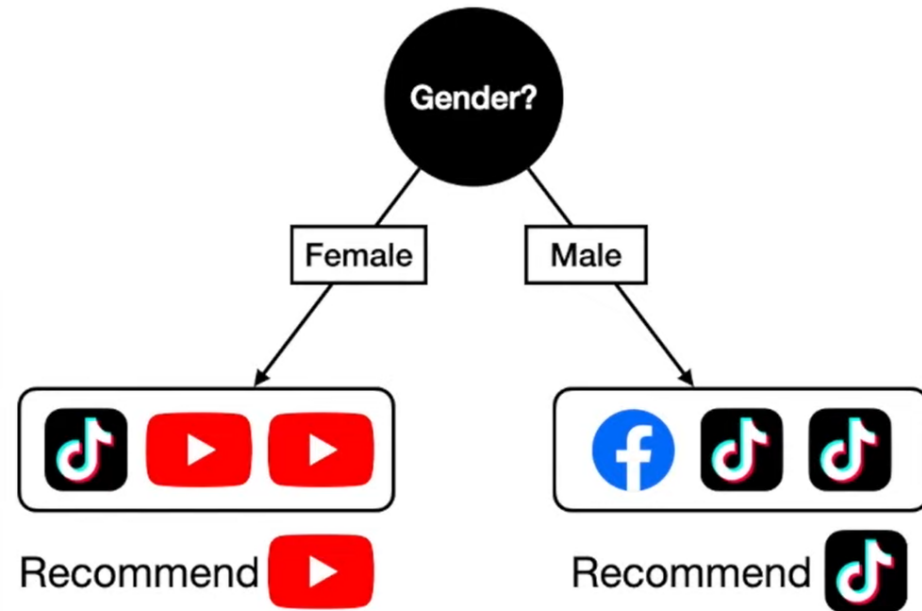


Young = Age \leq 18

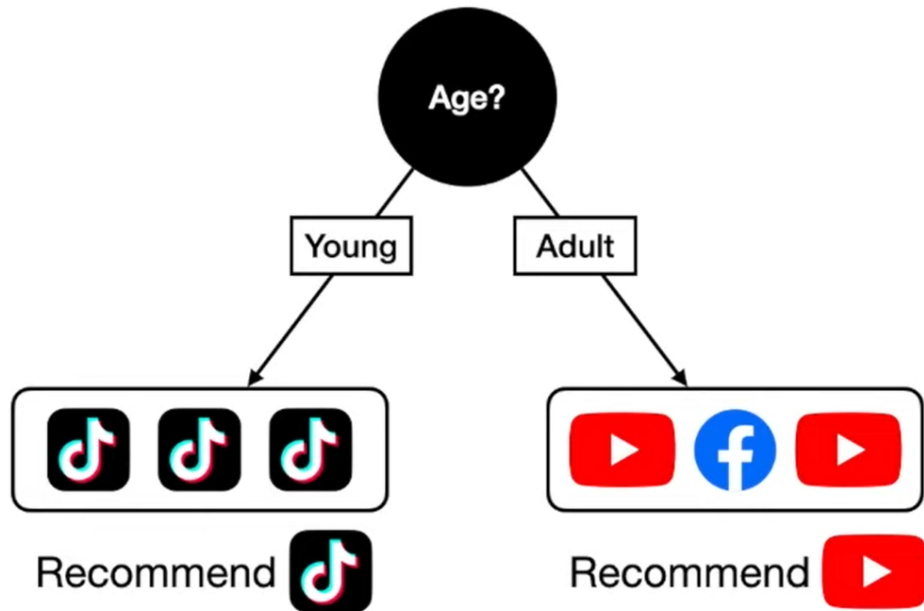
Adult : Age $>$ 18

Which one is better?

Gender decision stump



Age decision stump

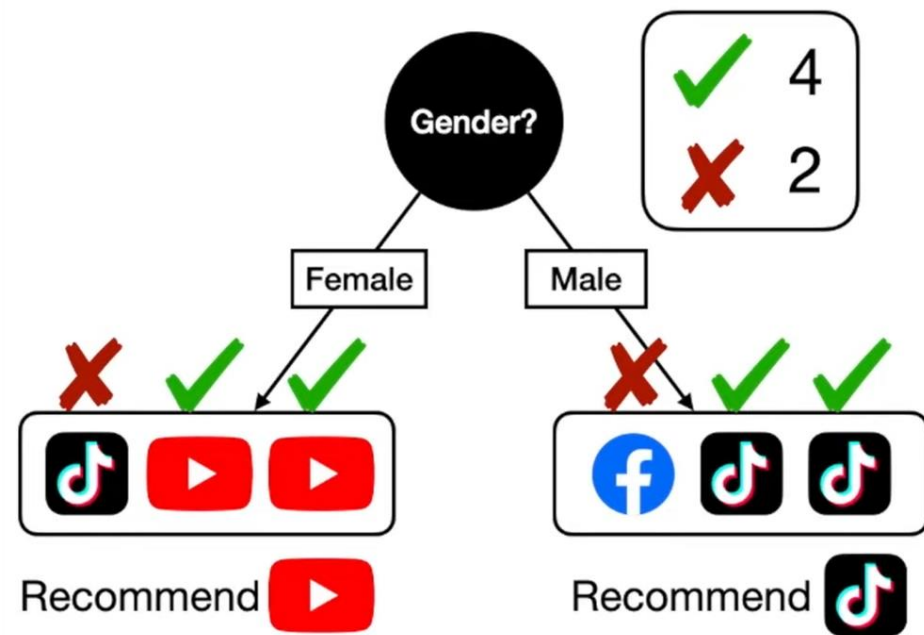


Each feature splits the data into two smaller datasets

Which one is better?

Accuracy: 66.7% & Error Rate: 33.3%

Gender decision stump

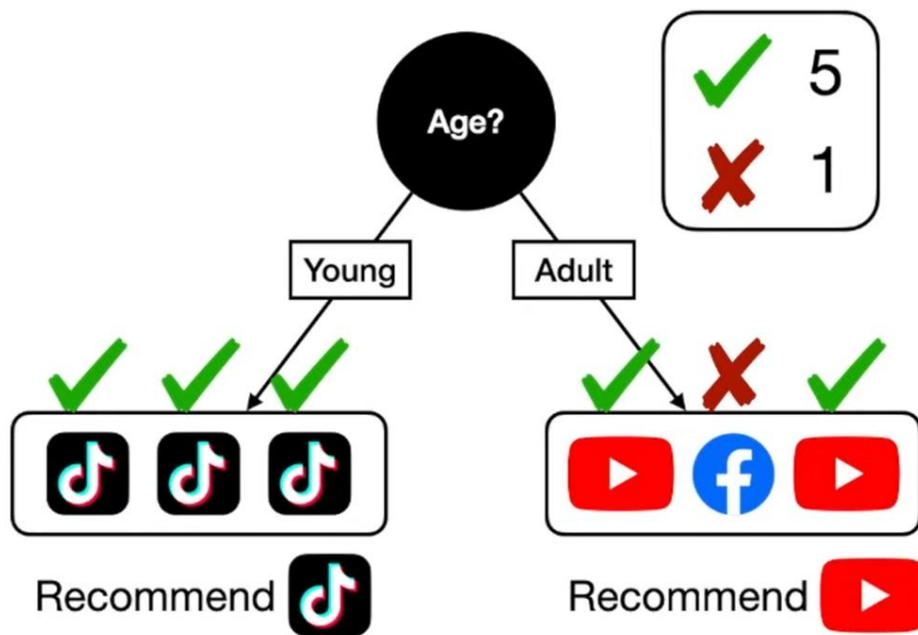


Accuracy: 83.3%









& Error Rate: 16.7%

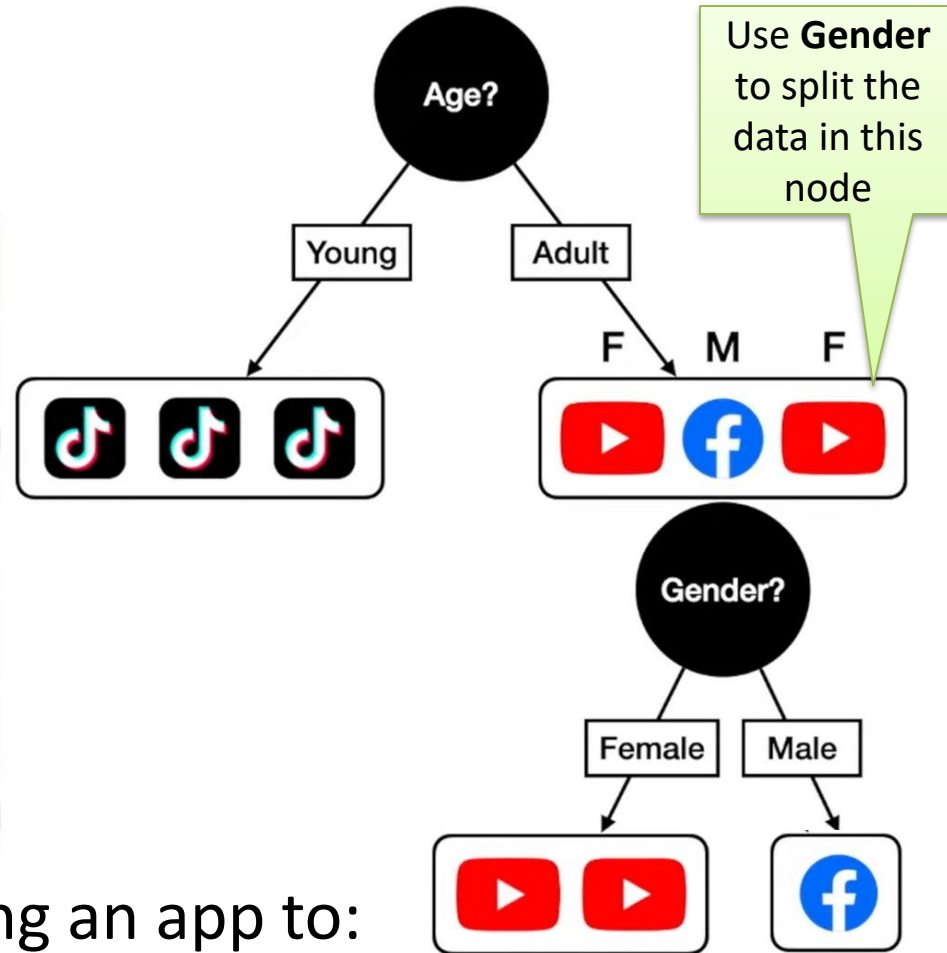
Age decision stump






- Based on **accuracy**, the **Age** feature is the winner => It is **determinant** in the prediction (differentiator) and deserve to be the **root of the tree** 🏆 (it has the highest accuracy and the **lowest classification error rate**)
 - It is more successful at determining which app to recommend

Building the Tree

Gender	Age	App
Female	Young	
Female	Adult	
Male	Adult	
Female	Adult	
Male	Young	
Male	Young	



- Let's test it for recommending an app to:
 - Female, 16 years old 
 - Female, 30 years old 
 - Male, 35 years old 

Building the Tree – Key Decisions

1. **Choose the Best Feature to Split On (i.e., asking the best question):**
 - At each node, **select the feature that best separates the data** into distinct classes or reduces **impurity** (uncertainty) the most
 - Use **Attribute Selection Measure (ASM)** like **Classification error**, **Gini impurity**, **Entropy**
2. **Determine the Split Point:**
 - Make the selected feature a decision node then find the **optimal Split Point (i.e., splitting condition/threshold)** that minimizes the impurity (using the same ASM used in step 1)
3. **Recursive dataset partitioning until a stop condition:**
 - Recursively split the dataset into subsets based on the selected best **feature** and its **split point**. Continues **until a stopping condition is met**
 - Common stopping criteria include limiting the maximum depth of the tree

Decision Tree Construction Algorithms

Many algorithms exist for Tree Construction (aka **Tree induction**), they mainly differ in adopted **Attribute Selection Measure (ASM)**:

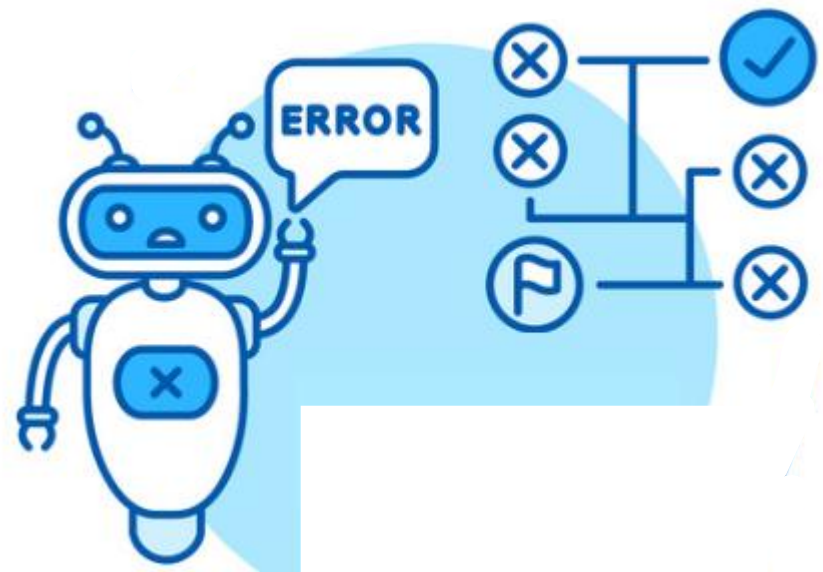
- **CART Algorithm** (Classification and Regression Trees)
 - Produce binary decision tree
 - Use Gini Impurity Index of as ASM
- **ID3** (Iterative Dichotomiser 3)
 - Uses **Entropy** and **Information Gain** as ASM
 - C4.5: An extension of ID3 that handles both continuous and discrete attributes and can handle missing values
- **CHAID Algorithm** (Chi-squared Automatic Interaction Detection)
 - Use Chi-square test (χ^2) as ASM
- Studies show that there are only marginal differences among the attribute selection measures w.r.t. model accuracy

Attribute Selection Measure (ASM)

Classification Error Rate

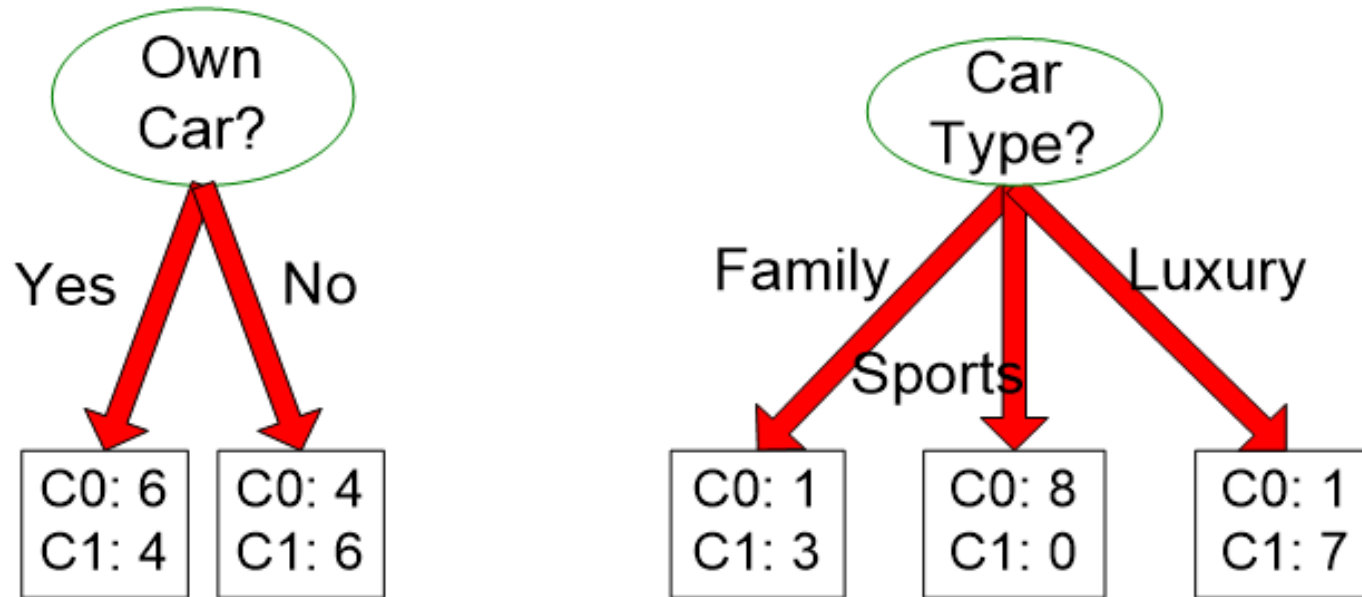
$$Error(t) = 1 - \max_i P(i|t)$$

Where $P(i|t)$ is the probability of class i at node t



How to choose the best Feature to Split On?

Before Splitting: 10 records of Class 0 (Not Defaulted Borrower)
10 records of Class 1 (Defaulted Borrower)



Which best feature to split on?

=> We need an **Attribute Selection Measure (ASM)**
for choosing the best feature to split that **maximizes the separation of classes**
(i.e., **minimizes impurity within each subset**)

How to choose the best Feature to Split On?

- Nodes with **homogeneous** class distribution (having **low impurity**) are preferred

C0: 5 C1: 5

**Non-homogeneous,
High impurity**

C0: 9 C1: 1

**Homogeneous,
Low impurity**

- Need an **Attribute Selection Measure (ASM)** to measure **the node impurity** and compare features to choose the best Feature to Split On
 - Classification Error Rate
 - **Gini Impurity Index** (or Gini index), and **Entropy** are measures of node impurity
- Then choose the feature that **minimizes impurity** within each subset (i.e., maximizes the **separation** of classes)

Classification Error Rate

- The Classification Error Rate, aka **Misclassification Rate**, is the ratio of the number of incorrectly classified instances to the total number of instances in the node

$$\text{Classification Error Rate} = \frac{\text{Number of Misclassified Instances}}{\text{Total Number of Instances}}$$

$$\text{Error}(t) = 1 - \max_i P(i|t)$$

Where $P(i|t)$ is the probability of class i at node t

- $\text{Error}(t)$ measures the classification error made by a node
 - Fraction of the instances in the node that do not belong to the most common class
 - Minimum **0** for pure node (containing one class label)
 - Maximum $(1 - \frac{1}{n_c})$ when the node has equally distributed n_c classes
- The DT algorithms selects the feature that minimizes the Classification Error Rate

Examples for Computing Classification Error Rate

$$Error(t) = 1 - \max_i P(i | t)$$

Female (Left)	T	1
	Y	2

$$P(T) = \frac{1}{3} \quad P(Y) = \frac{2}{3}$$

$$Error(L) = 1 - \max\left(\frac{1}{3}, \frac{2}{3}\right) = 1 - \frac{2}{3} = \frac{1}{3}$$

Male (Right)	F	1
	T	2

$$P(F) = \frac{1}{3} \quad P(T) = \frac{2}{3}$$

$$Error(R) = 1 - \max\left(\frac{1}{3}, \frac{2}{3}\right) = 1 - \frac{2}{3} = \frac{1}{3}$$

Weighted Average

$$\bar{x}_{weighted} = \frac{\sum_{i=1}^n w_i x_i}{\sum_{i=1}^n w_i}$$

$$Error(Gender) = \frac{1}{3} = 33.3\%$$

Young (Left)	T	3

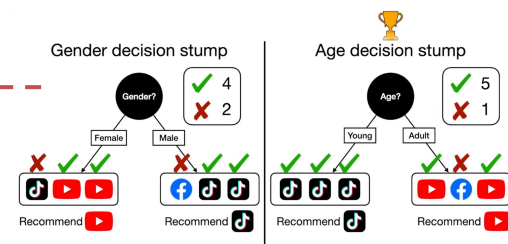
$$P(T) = \frac{3}{3}$$

$$Error(L) = 1 - \max(1) = 1 - 1 = 0$$

Adult (Right)	Y	2
	F	1

$$P(Y) = \frac{2}{3} \quad P(F) = \frac{1}{3}$$

$$Error(R) = 1 - \max\left(\frac{2}{3}, \frac{1}{3}\right) = 1 - \frac{2}{3} = \frac{1}{3}$$

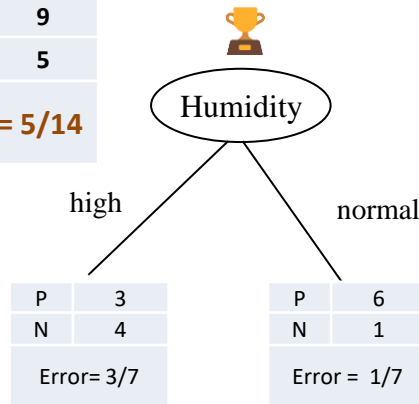


$$Error(Age) = \frac{1}{6} = 16.7\%$$



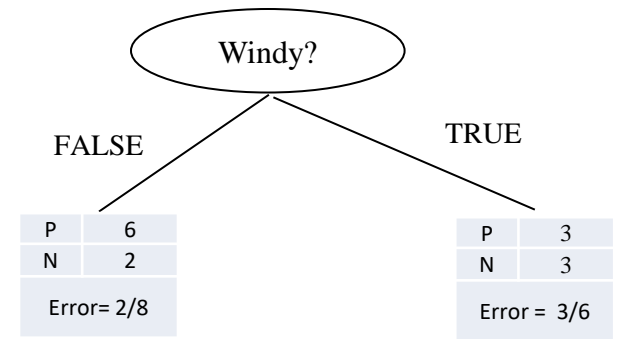
Outlook	Temperature	Humidity	Windy	Class
sunny	Hot	high	FALSE	N
sunny	Hot	High	TRUE	N
Overcast	Hot	high	FALSE	P
Rain	Mild	high	FALSE	P
Rain	Cool	Normal	FALSE	P
Rain	Cool	Normal	TRUE	N
Overcast	Cool	Normal	TRUE	P
Sunny	Mild	High	FALSE	N
Sunny	Cool	normal	FALSE	P
Rain	Mild	Normal	FALSE	P
Sunny	Mild	Normal	TRUE	P
Overcast	Mild	High	TRUE	P
Overcast	Hot	Normal	FALSE	P
Rain	mild	high	TRUE	N

	Parent
P	9
N	5
Error= 5/14	



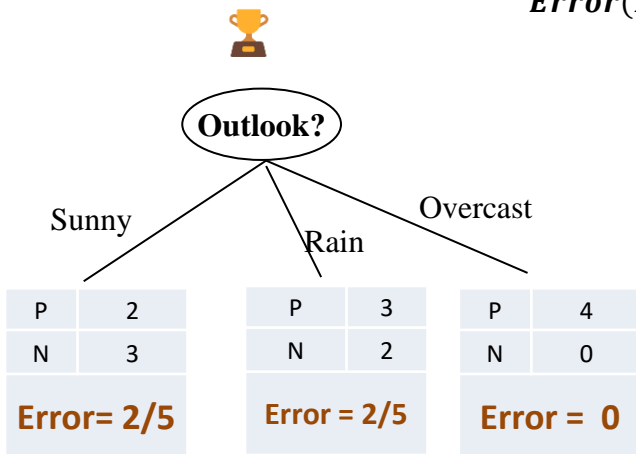
Weighted Average Error for the Humidity Split

$$Error(Humidity) = \frac{7}{14} \times \frac{3}{7} + \frac{7}{14} \times \frac{1}{7} = \frac{3}{14} + \frac{1}{14} = \frac{4}{14}$$



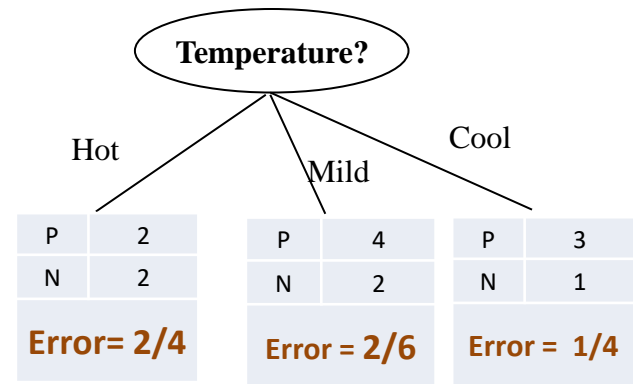
Weighted Average Error for the Windy Split

$$Error(Windy) = \frac{8}{14} \times \frac{2}{8} + \frac{6}{14} \times \frac{3}{6} = \frac{5}{14}$$



Weighted Average Error for the Outlook Split

$$Error(Outlook) = \frac{5}{14} \times \frac{2}{5} + \frac{5}{14} \times \frac{2}{5} + \frac{4}{14} \times 0 = \frac{2}{14} + \frac{2}{14} = \frac{4}{14}$$

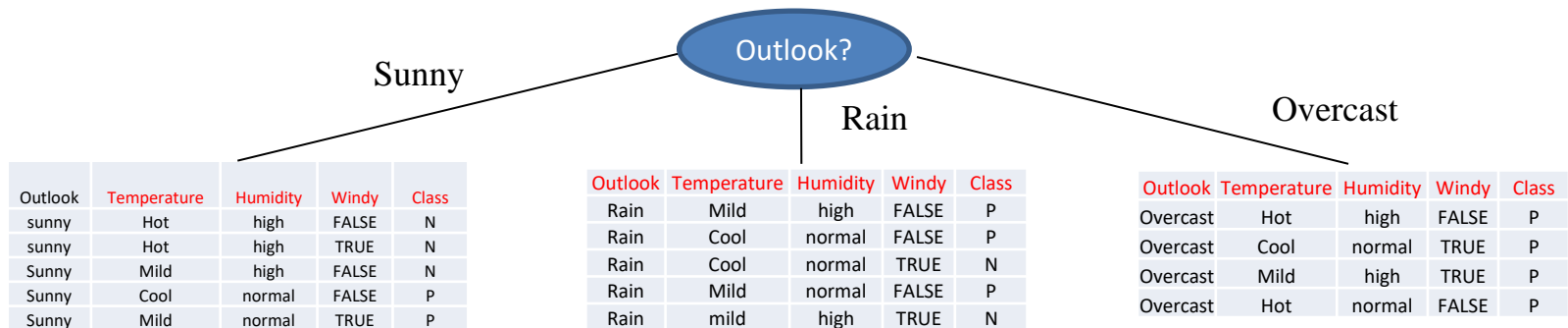


Weighted Average Error for the Temperature Split

$$Error(Temperature) = \frac{4}{14} \times \frac{2}{4} + \frac{6}{14} \times \frac{2}{6} + \frac{4}{14} \times \frac{1}{4} = \frac{2}{14} + \frac{2}{14} + \frac{1}{14} = \frac{5}{14}$$

The best two splits are Outlook and Humidity
We can use Outlook or Humidity since both have the same error

Outlook	Temperature	Humidity	Windy	Class
sunny	Hot	high	FALSE	N
sunny	Hot	high	TRUE	N
Overcast	Hot	high	FALSE	P
Rain	Mild	high	FALSE	P
Rain	Cool	normal	FALSE	P
Rain	Cool	normal	TRUE	N
Overcast	Cool	normal	TRUE	P
Sunny	Mild	high	FALSE	N
Sunny	Cool	normal	FALSE	P
Rain	Mild	normal	FALSE	P
Sunny	Mild	normal	TRUE	P
Overcast	Mild	high	TRUE	P
Overcast	Hot	normal	FALSE	P
Rain	mild	high	TRUE	N



Next: Repeat selecting the best feature to split on

Attribute Selection Measure (ASM)

Gini Impurity Index

$$Gini(t) = 1 - \sum_{i=1}^k p_i(t)^2$$

Where $p_i(t)$ is the frequency of class i at node t , and k is the number of classes at node t



Low Gini
impurity index



High Gini
impurity index

Gini Index

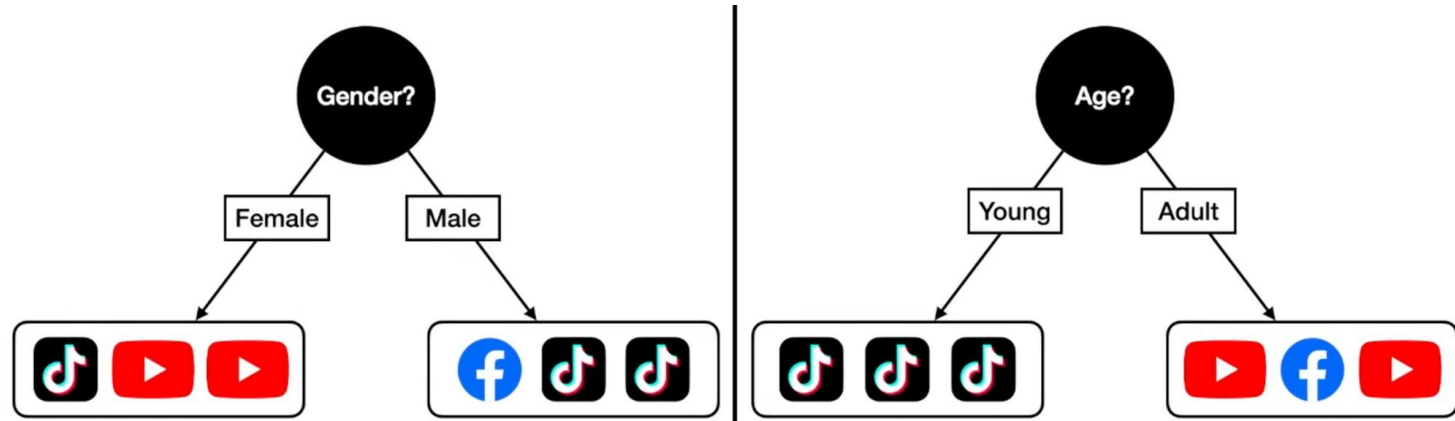
- The Gini Index is a **measure of impurity** or randomness in a dataset
 - The Gini Index ranges from 0 to **$1-1/k$** , where:
 - **0**: indicates perfect purity, meaning all node elements belong to a single class
 - Maximum of **$1-1/k$** indicates maximally impure node, having instances evenly distributed across all **k** classes
- The formula to calculate the Gini Index for a node **t** with **k** classes is:

$$Gini(t) = 1 - \sum_{i=1}^k p_i(t)^2$$

Where **$p_i(t)$** is the frequency of class **i** at node **t** , and **k** is the number of classes at node **t**

- DT algorithm selects the **feature** and **split point** that **minimizes** the weighted sum of the Gini indices for the resulting child nodes

Which one is better? => Compute Gini Index



Classifier 1 (by Gender): Avg Gini = $((3 \times 0.44) + (3 \times 0.44)) / 6 = 0.44$

- Left leaf (Female): {T, Y, Y}

$$\text{Gini} = 1 - (\mathbf{P(T)^2} + \mathbf{P(Y)^2}) = 1 - \left(\left(\frac{1}{3}\right)^2 + \left(\frac{2}{3}\right)^2 \right) = \mathbf{0.44}$$

- Right leaf (Male): {F, T, T}

$$\text{Gini} = 1 - (\mathbf{P(F)^2} + \mathbf{P(T)^2}) = 1 - \left(\left(\frac{1}{3}\right)^2 + \left(\frac{2}{3}\right)^2 \right) = \mathbf{0.44}$$

Measures the
impurity of
the split

Classifier 2 (by age): Avg Gini = $((3 \times 0) + (3 \times 0.44)) / 6 = 0.22$



- Left leaf (young): {T, T, T}. **Gini** = $1 - \mathbf{P(T)^2} = 1 - \left(\frac{3}{3}\right)^2 = \mathbf{0}$

- Right leaf (adult): {Y, F, Y}. **Gini** = $1 - (\mathbf{P(Y)^2} + \mathbf{P(F)^2}) = 1 - \left(\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2 \right) = \mathbf{0.44}$

Compute Gini Index – Example 2

$$Gini(t) = 1 - \sum_{i=1}^K p(i|t)^2$$

Where $p(i|t)$ is the probability of class i at node t

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$Gini = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$Gini = 1 - (1/6)^2 - (5/6)^2 = 0.278$$

C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$Gini = 1 - (2/6)^2 - (4/6)^2 = 0.444$$

Splitting Based on Gini

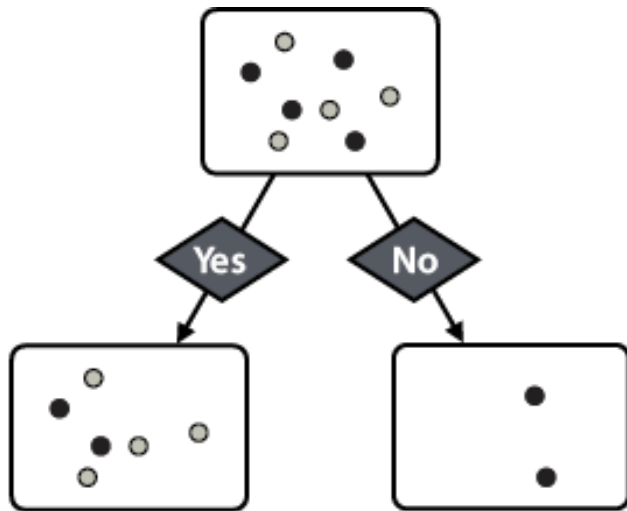
- When a node p is split into k partitions (children), the **quality of split** is computed as

$$GINI_{split} = \sum_{i=1}^k \frac{n_i}{n} GINI(i)$$

Where: n_i = number of instances at child partition i
 n = number of instances at node p

Gini Index of a Node => take weighted average

- A split of a Node of size 8 into 2 partitions of sizes 6 and 2
- We calculate the Gini index of the Node as the **weighted average** of the Gini of partitions:
 - We weight the index of the left partition by 6/8 and that of the right partition by 2/8



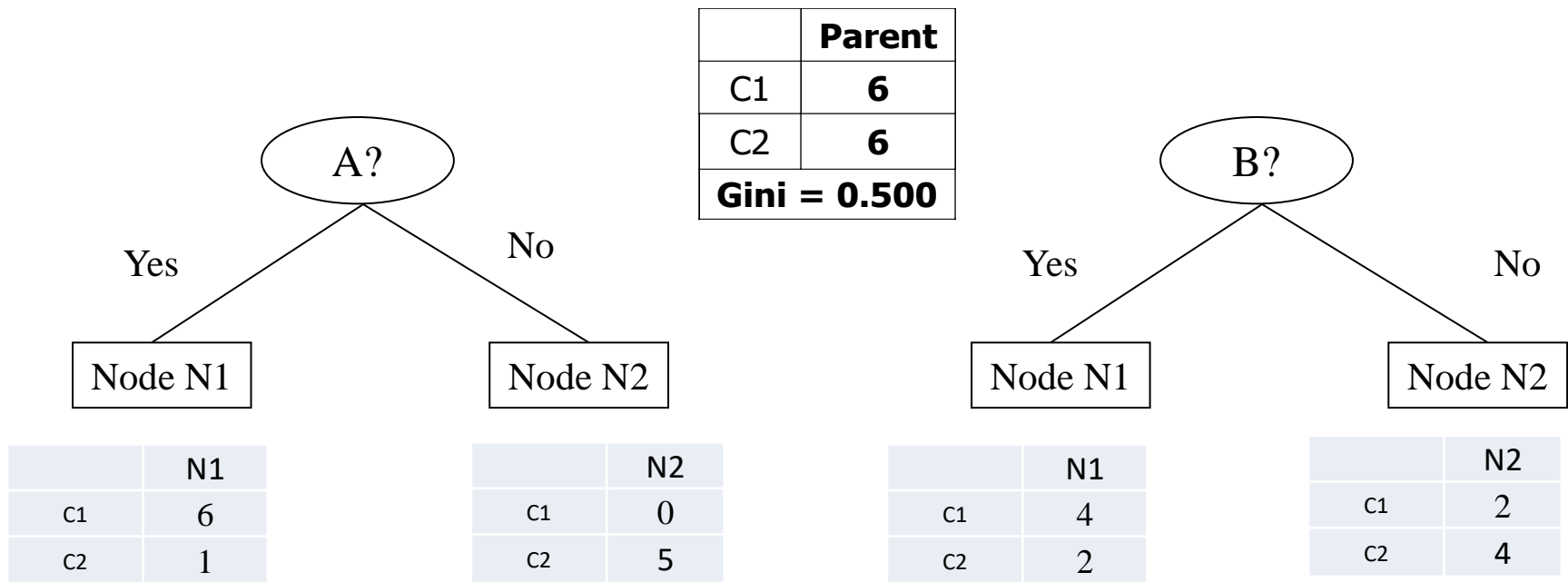
Gini = 0.444

Gini = 0

$$\text{Weighted average Gini} = 0.444 \cdot \frac{6}{8} + 0 \cdot \frac{2}{8} = 0.333$$

Computing Gini Index of a Split

- Split into two partitions
- Compute the Gini Index per split
- Choose the split with the lowest Gini Index



$$\text{Gini}(N1) = 1 - (6/7)^2 - (1/7)^2 = 0.245$$

$$\text{Gini}(N2) = 1 - (0/5)^2 - (5/5)^2 = 0$$

$$\text{Gini}(A) = 7/12 * 0.245 + 5/12 * 0 = \mathbf{0.1429}$$

$$\text{Gini}(N1) = 1 - (4/6)^2 - (2/6)^2 = 0.2222$$

$$\text{Gini}(N2) = 1 - (2/6)^2 - (4/6)^2 = 0.2222$$

$$\text{Gini}(B) = 6/12 * 0.2222 + 6/12 * 0.2222 = \mathbf{0.2222}$$

Choose split A as it has a Gini Index < split B

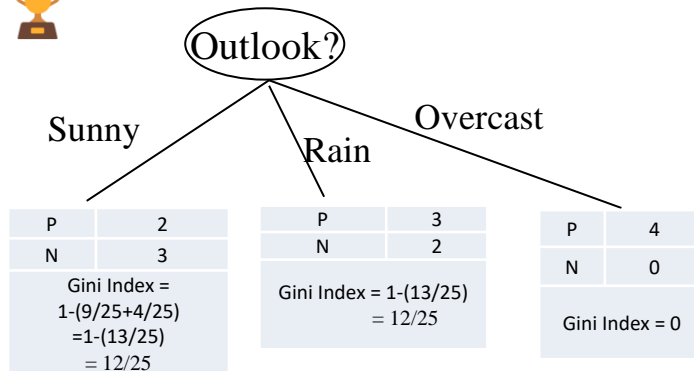
Computing Gini Index – Example (1/2)

$$Gini(t) = 1 - \sum_{i=1}^K p(i|t)^2$$

Where $p(i|t)$ is the probability of class i at node t

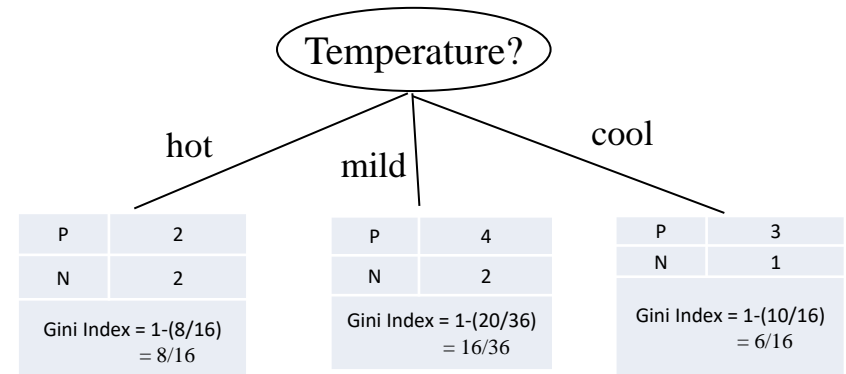
Outlook	Temperature	Humidity	Windy	Class
sunny	Hot	high	FALSE	N
sunny	Hot	high	TRUE	N
Overcast	Hot	high	FALSE	P
Rain	Mild	high	FALSE	P
Rain	Cool	normal	FALSE	P
Rain	Cool	normal	TRUE	N
Overcast	Cool	normal	TRUE	P
Sunny	Mild	high	FALSE	N
Sunny	Cool	normal	FALSE	P
Rain	Mild	normal	FALSE	P
Sunny	Mild	normal	TRUE	P
Overcast	Mild	high	TRUE	P
Overcast	Hot	normal	FALSE	P
Rain	mild	high	TRUE	N

	Parent
P	9
N	5
Gini Index = $1 - (25/196 + 81/196)$ = $90/196 = 0.4592$	



Weighted Average Gini Index for the Outlook Split

$$Gini(Outlook) = \frac{5}{14} \times \frac{12}{25} + \frac{5}{14} \times \frac{12}{25} + \frac{4}{14} \times 0 = \frac{6}{35} + \frac{6}{35} = \frac{12}{35} = 0.3429$$



Weighted Average Gini Index for the Temperature Split

$$Gini(Temperature) = \frac{4}{14} \times \frac{8}{16} + \frac{6}{14} \times \frac{16}{36} + \frac{4}{14} \times \frac{6}{16}$$

$$Gini(Temperature) = \frac{2}{14} + \frac{4}{21} + \frac{3}{28} = \frac{4}{21} + \frac{7}{28} = 0.4405$$

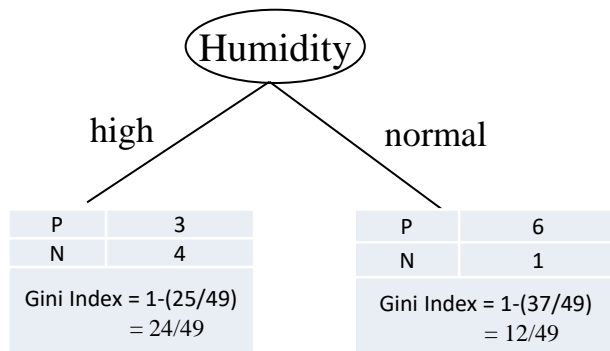
Computing Gini Index – Example (2/2)

$$Gini(t) = 1 - \sum_{i=1}^k p_i(t)^2$$

Where $p_i(t)$ is the frequency of class i at node t , and k is the number of classes at node t

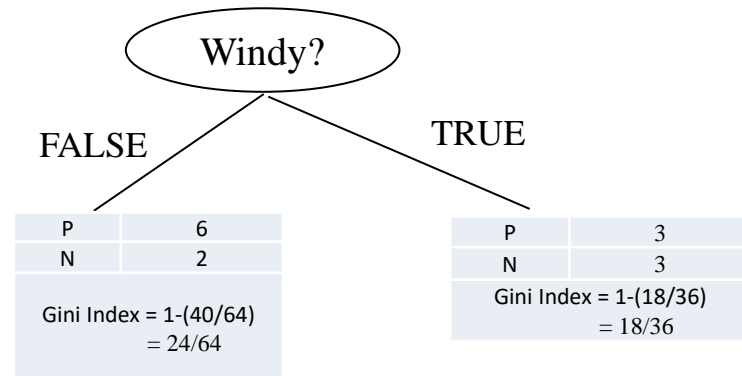
Outlook	Temperature	Humidity	Windy	Class
sunny	Hot	high	FALSE	N
sunny	Hot	high	TRUE	N
Overcast	Hot	high	FALSE	P
Rain	Mild	high	FALSE	P
Rain	Cool	normal	FALSE	P
Rain	Cool	normal	TRUE	N
Overcast	Cool	normal	TRUE	P
Sunny	Mild	high	FALSE	N
Sunny	Cool	normal	FALSE	P
Rain	Mild	normal	FALSE	P
Sunny	Mild	normal	TRUE	P
Overcast	Mild	high	TRUE	P
Overcast	Hot	normal	FALSE	P
Rain	mild	high	TRUE	N

	Parent
P	9
N	5
Gini Index = $1 - (25/196 + 81/196)$ = $90/196 = 0.4592$	



Weighted Average Gini Index for the Humidity Split

$$Gini(Outlook) = \frac{7}{14} \times \frac{24}{49} + \frac{7}{14} \times \frac{12}{49} = \frac{12}{49} + \frac{6}{49} = \frac{18}{49} = 0.3673$$



Weighted Average Gini Index for the Windy Split

$$Gini(Temperature) = \frac{8}{14} \times \frac{24}{64} + \frac{6}{14} \times \frac{18}{36} = \frac{3}{14} + \frac{3}{14} = \frac{6}{14} = 0.4286$$

The best split is Outlook (0.3429)

Decide optimal **Split Point** using Gini Index for Categorical Attributes

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make **Split Point** decisions

Multi-way split



	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

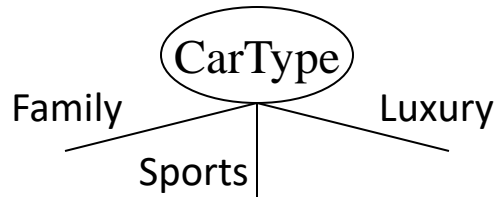
Two-way split
(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

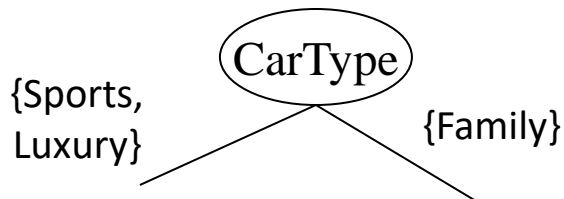
	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

Splitting Based on Nominal Attributes

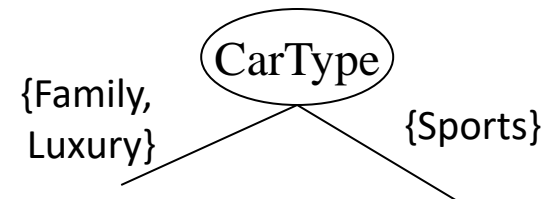
- **Multi-way split:** Use as many partitions as distinct values



- **Binary split:** Divides values into two subsets
Need to find optimal partitioning

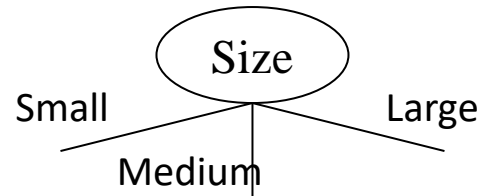


OR

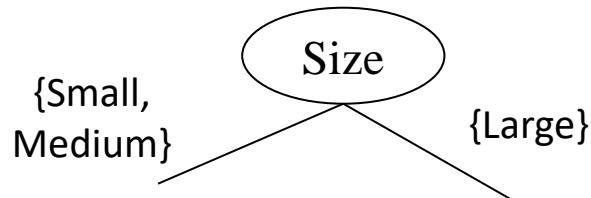


Splitting Based on Ordinal Attributes

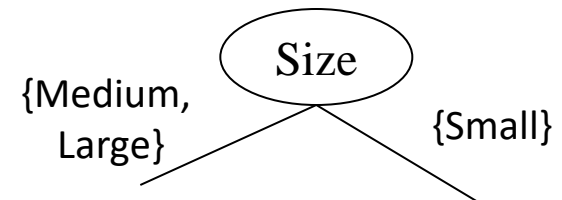
- **Multi-way split:** Use as many partitions as distinct values



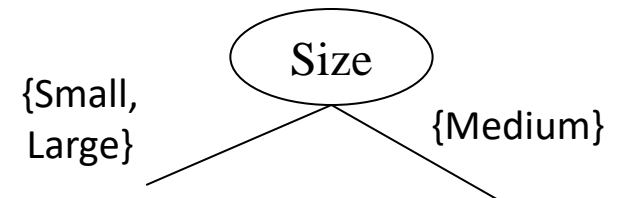
- **Binary split:** Divides values into two subsets
Need to find optimal partitioning



OR



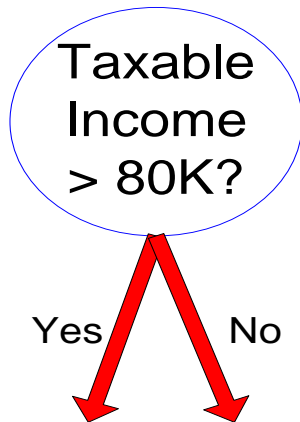
- What about this split?
No! the grouping should not violate the order property of the attribute values



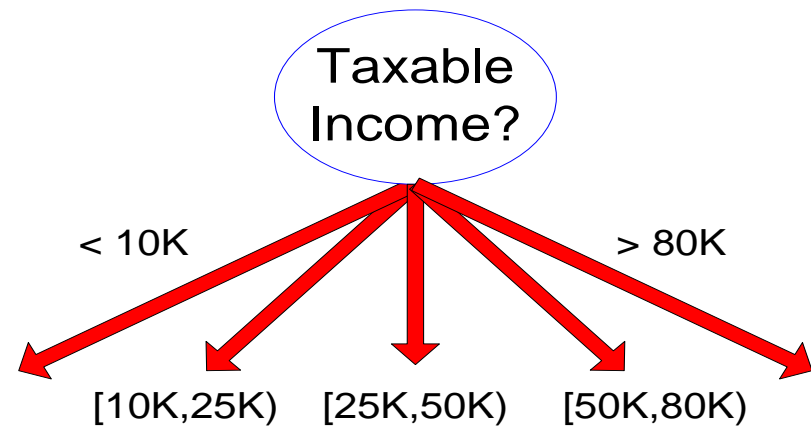
Splitting Based on Continuous Attributes

- Different ways of handling continuous attributes
 - Let attribute A be a continuous attribute
 - **Discretization** to form an ordinal categorical attribute
 - Ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering
 - Sort the value A in increasing order
 - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
 - $(a_i + a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
 - **Binary split**: $(A < v)$ or $(A \geq v)$, OR A in a range
 - Consider all possible splits and finds the best cut
 - **Multi-way split**: A in one of the ranges
- Can be compute intensive

Splitting Based on Continuous Attributes









(i) Binary split

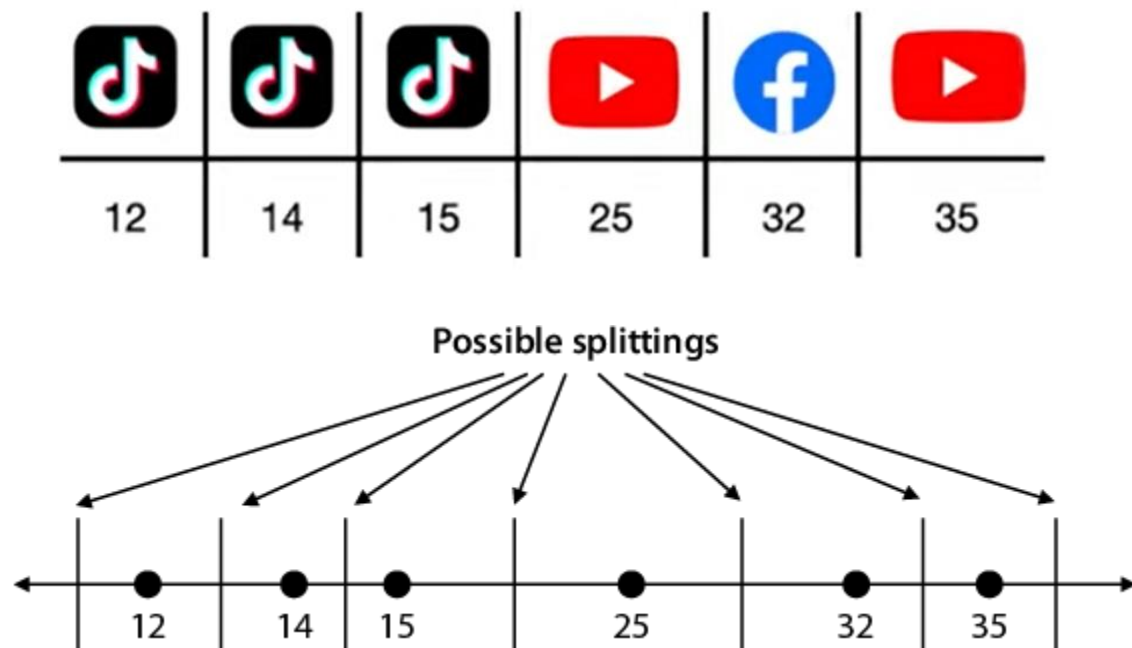


(ii) Multi-way split

Decide optimal **Split Point** using Gini Index

Continuous Attributes

Gender	Age	App
Female	15	
Female	25	
Male	32	
Female	35	
Male	12	
Male	14	



- Sort the entries by **age**
- We pick the **midpoints** between consecutive ages to be the age for **splitting**
 - For the endpoints, we can pick any random value that is out of the interval
- Then calculate the Gini impurity index of each of the splits
- Choose the **Split Point** that has the lowest Gini index

Decide optimal Split Point using Gini Index - Example



	Age													
	12		14		15		25		32		35			
App	7		13		14.5		20		28.5		33.5		100	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
T	0	3	1	2	2	1	3	0	3	0	3	0	3	0
Y	0	2	0	2	0	2	0	2	1	1	2	1	2	0
F	0	1	0	1	0	1	0	1	2	0	1	0	1	0
Gini	0.611		0.533		0.417		0.22		0.416		0.467		0.611	



The best Split Point is age ≤ 20

Attribute Selection Measure (ASM)

Entropy

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5



Entropy(PlayGolf) = Entropy(5,9)
= Entropy(0.36, 0.64)
= - (0.36 log₂ 0.36) - (0.64 log₂ 0.64)
= 0.94

Entropy

- Entropy is a **measure of impurity** or randomness in a dataset. It quantifies the **amount of uncertainty** associated with the distribution of class labels in the dataset
- The formula to calculate entropy for a set S with K classes is:

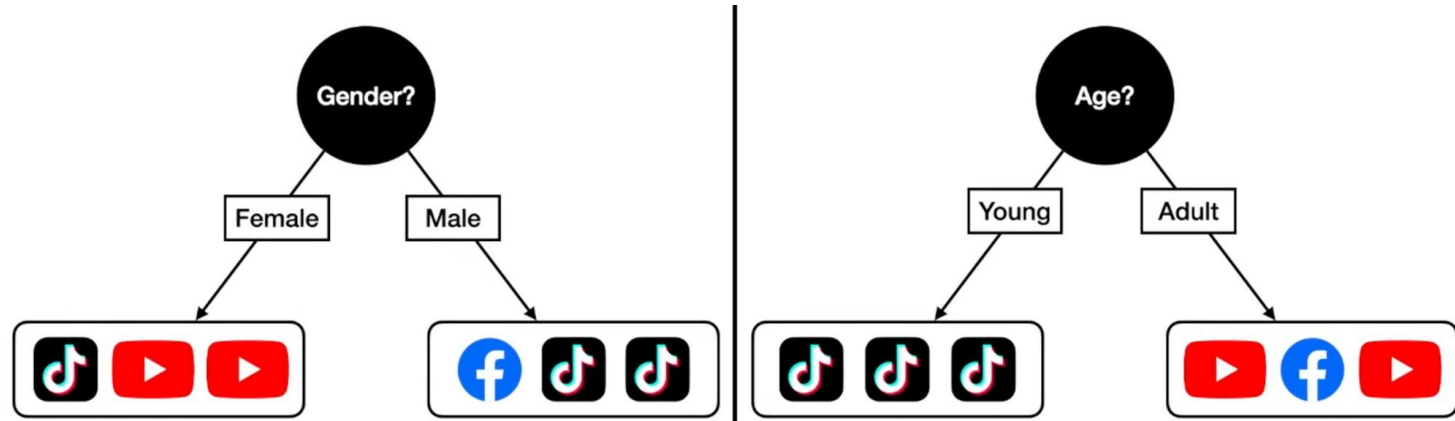
$$\text{Entropy}(S) = - \sum_{i=1}^K p_i \log_2(p_i)$$

Where p_i is the probability of class i in set S .

Fraction of instances
of a given class....

- Entropy ranges from 0 to $\log_2(K)$, where:
 - 0 indicates that the set S is pure (all instances belong to the same class)
 - **$\log_2(K)$** indicates maximum entropy (the instances are evenly distributed across all classes)
- DT algorithm selects the feature and split point that **result in subsets with lower entropy**

Which one is better? => Compute Entropy



Classifier 1 (by Gender): Avg Entropy = $((3 \times 0.918) + (3 \times 0.918))/6 = 0.918$

- Left leaf (Female): {T, Y, Y}

$$\text{Entropy} = -\mathbf{P(T)} \log_2 \mathbf{P(T)} - \mathbf{P(Y)} \log_2 \mathbf{P(Y)} = -\frac{1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} = \mathbf{0.918}$$

- Right leaf (Male): {F, T, T}

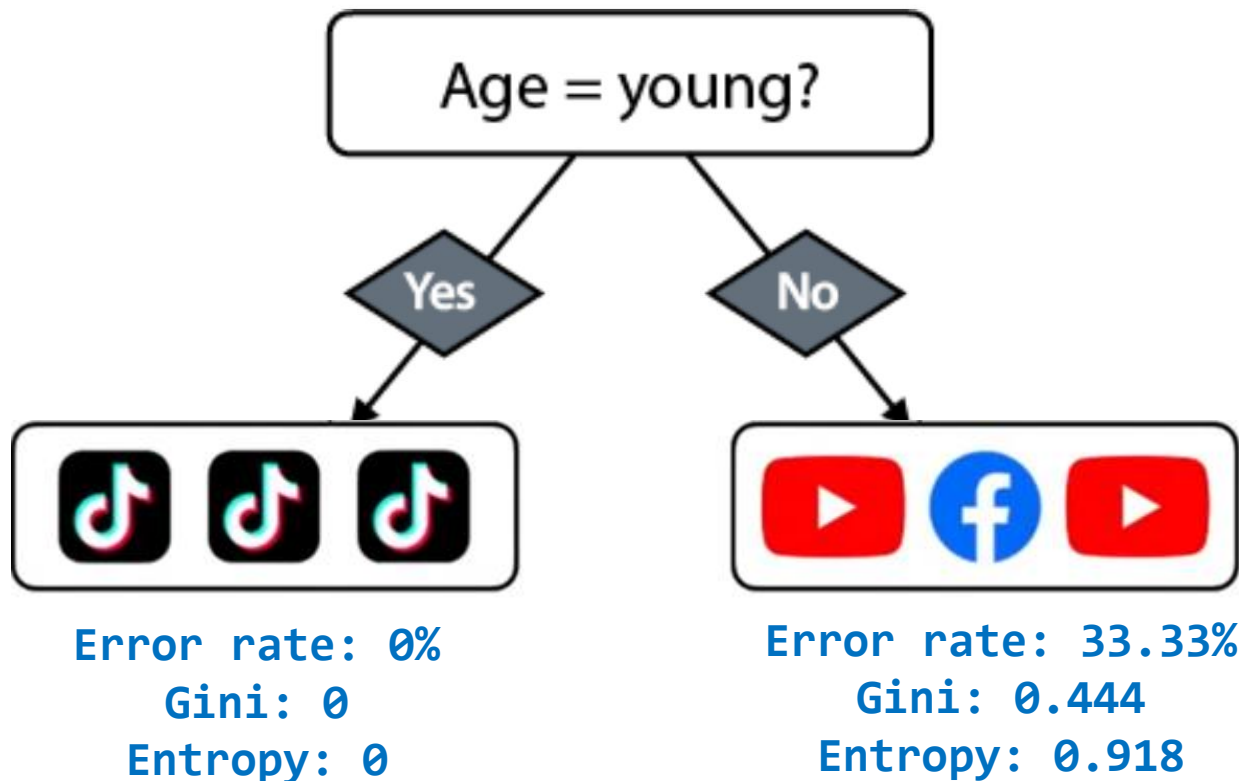
$$\text{Entropy} = -\mathbf{P(F)} \log_2 \mathbf{P(F)} - \mathbf{P(T)} \log_2 \mathbf{P(T)} = -\frac{1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} = 0.918$$

Classifier 2 (by age): Avg Entropy = $((3 \times 0) + (3 \times 0.918))/6 = 0.459$

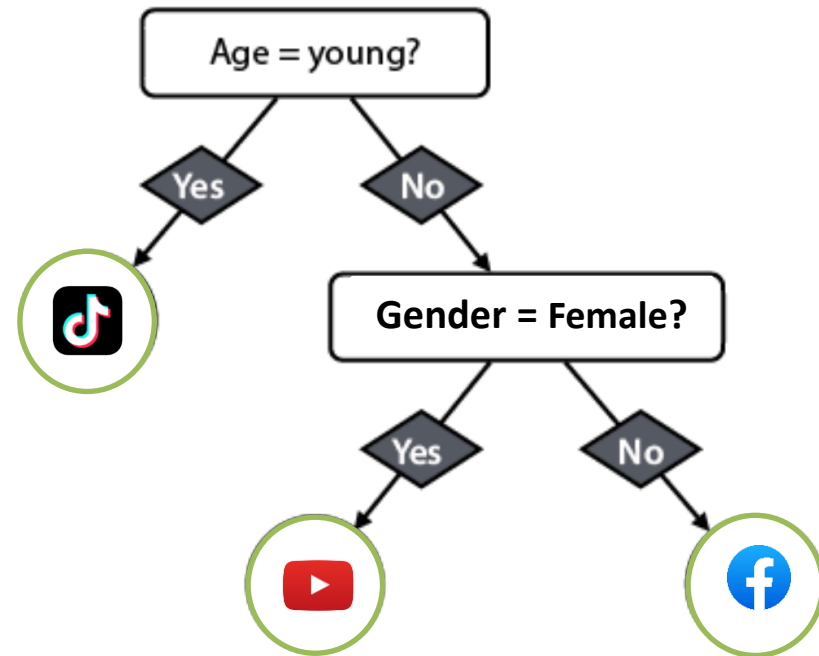
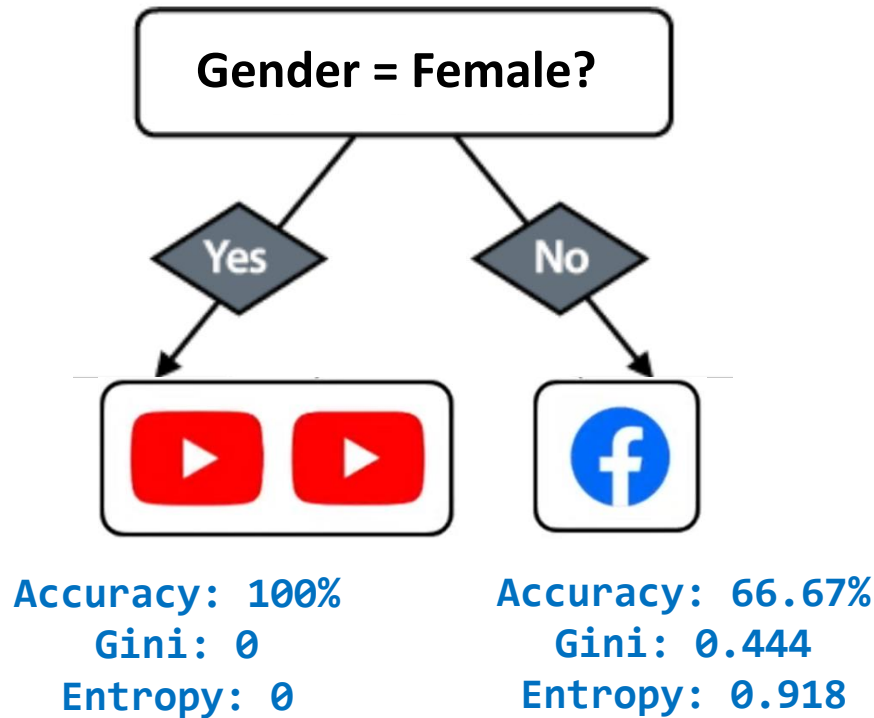


- Left leaf (young): {T, T, T}. **Entropy** = $-\mathbf{P(T)} \log_2 \mathbf{P(T)} = -\frac{3}{3} \log_2 \frac{3}{3} = \mathbf{0}$

- Right leaf (adult): {Y, F, Y}. **Entropy** = $-\mathbf{P(Y)} \log_2 \mathbf{P(Y)} - \mathbf{P(F)} \log_2 \mathbf{P(F)}$
 $= -\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3} = 0.918$



- When we split our dataset by age, we get two partitioned datasets
- The one on the left **is pure** (all the labels are the same), its error rate is 0%, and its Gini index and entropy are both 0
 - Thus, this node becomes **a leaf node**, and when we get to that leaf, we return the prediction **TikTok**
- The split on the right is **impure** and can still be divided using the Gender feature



- We can split the right leaf of the tree in the previous slide using Gender and we obtain two pure datasets. Each having an accuracy of 100% and a Gini index and entropy of 0
- After this split, we are done, because we can't improve our splits any further
- The resulting decision tree (shown on the right) has two nodes and three leaves. This tree predicts every point in the original dataset correctly

Entropy Calculation – Example 2 (1/5)

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	FALSE	N
sunny	hot	high	TRUE	N
overcast	hot	high	FALSE	P
rain	mild	high	FALSE	P
rain	cool	normal	FALSE	P
rain	cool	normal	TRUE	N
overcast	cool	normal	TRUE	P
sunny	mild	high	FALSE	N
sunny	cool	normal	FALSE	P
rain	mild	normal	FALSE	P
sunny	mild	normal	TRUE	P
overcast	mild	high	TRUE	P
overcast	hot	normal	FALSE	P
rain	mild	high	TRUE	N

$$\text{Entropy}(S) = - \sum_{i=1}^K p_i \log_2(p_i)$$

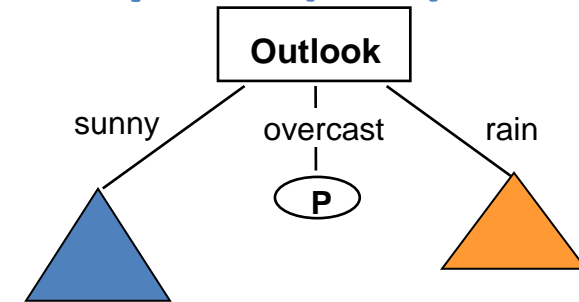
Where p_i is the probability of class i in set S .

$$E(S) = -\frac{9}{9+5} \cdot \log_2 \frac{9}{9+5} - \frac{5}{9+5} \cdot \log_2 \frac{5}{9+5} \approx 0.94$$

Entropy Calculation ID3 Algorithm – Example 2 (2/5)

$$E(S) = -\frac{9}{9+5} \cdot \log_2 \frac{9}{9+5} - \frac{5}{9+5} \cdot \log_2 \frac{5}{9+5} \approx 0.94$$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	FALSE	N
sunny	hot	high	TRUE	N
overcast	hot	high	FALSE	P
rain	mild	high	FALSE	P
rain	cool	normal	FALSE	P
rain	cool	normal	TRUE	N
overcast	cool	normal	TRUE	P
sunny	mild	high	FALSE	N
sunny	cool	normal	FALSE	P
rain	mild	normal	FALSE	P
sunny	mild	normal	TRUE	P
overcast	mild	high	TRUE	P
overcast	hot	normal	FALSE	P
rain	mild	high	TRUE	N



sunny

overcast

rain

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	FALSE	N
sunny	hot	high	TRUE	N
sunny	mild	high	FALSE	N
sunny	cool	normal	FALSE	P
sunny	mild	normal	TRUE	P

Outlook	Temperature	Humidity	Windy	Class
overcast	hot	high	FALSE	P
overcast	cool	normal	TRUE	P
overcast	mild	high	TRUE	P
overcast	hot	normal	FALSE	P

Outlook	Temperature	Humidity	Windy	Class
rain	mild	high	FALSE	P
rain	cool	normal	FALSE	P
rain	cool	normal	TRUE	N
rain	mild	normal	FALSE	P
rain	mild	high	TRUE	N

$$E(\text{Sunny}) = -\frac{2}{5} \cdot \log_2 \frac{2}{5} - \frac{3}{5} \cdot \log_2 \frac{3}{5} \approx 0.971$$

$$E(\text{Overcast}) = -\frac{4}{4} \cdot \log_2 \frac{4}{4} = 0$$

$$E(\text{Rain}) = -\frac{3}{5} \cdot \log_2 \frac{3}{5} - \frac{2}{5} \cdot \log_2 \frac{2}{5} \approx 0.971$$

$$E(\text{Outlook}) = \frac{5}{14} \cdot E(\text{Sunny}) + \frac{4}{14} \cdot E(\text{Overcast}) + \frac{5}{14} \cdot E(\text{Rain}) = 0.694$$

$$\text{Gain}(\text{Outlook}) = E(S) - E(\text{Outlook}) = 0.94 - 0.694 = 0.246$$

Information gain **Gain(A)** is difference between the entropy before splitting the dataset **S** and the entropy after splitting based on the attribute **A**:

$$\text{Gain}(A) = \text{Entropy}(S) - \text{Entropy}(S, A)$$

Entropy Calculation ID3 Algorithm – Example 2 (3/5)

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	FALSE	N
sunny	hot	high	TRUE	N
overcast	hot	high	FALSE	P
rain	mild	high	FALSE	P
rain	cool	normal	FALSE	P
rain	cool	normal	TRUE	N
overcast	cool	normal	TRUE	P
sunny	mild	high	FALSE	N
sunny	cool	normal	FALSE	P
rain	mild	normal	FALSE	P
sunny	mild	normal	TRUE	P
overcast	mild	high	TRUE	P
overcast	hot	normal	FALSE	P
rain	mild	high	TRUE	N



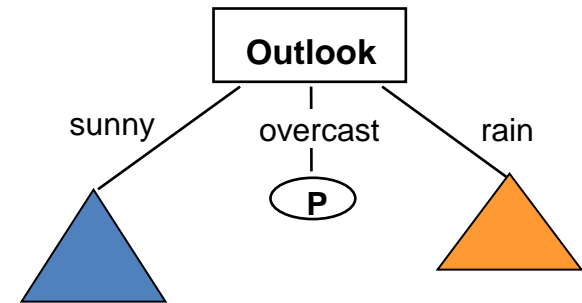
Gain(Outlook) = 0.246

Gain(Temperature) = 0.029

Gain(Humidity) = 0.151

Gain(Windy) = 0.048

∴ Outlook is chosen as the root



Choose the attribute with **the highest information gain** as the splitting attribute at the node i.e., the attribute that **provides the most reduction in uncertainty or impurity** in the dataset when used for splitting

Entropy Calculation ID3 Algorithm – Example 2 (4/5)

Training Set(outlook=Sunny)

Temperature	Humidity	Windy	Class
hot	high	FALSE	N
hot	high	TRUE	N
mild	high	FALSE	N
cool	normal	FALSE	P
mild	normal	TRUE	P

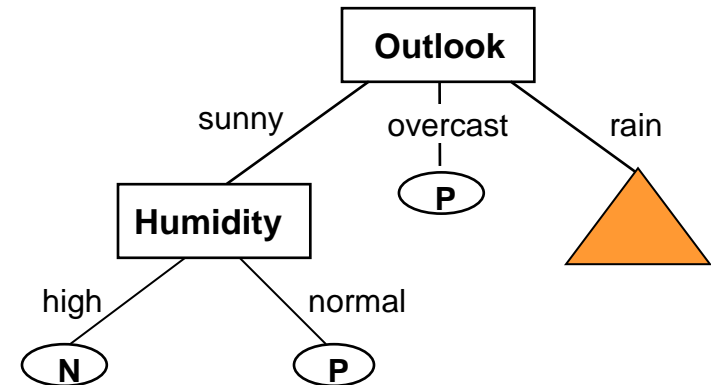


 **Gain(Temperature) = 0.571**

Gain(Humidity) = 0.971

Gain(Windy) = 0.020

∴ Humidity is chosen as the root



Entropy Calculation ID3 Algorithm – Example 2 (5/5)

Training Set(outlook=Rain)

Temperature	Humidity	Windy	Class
mild	high	FALSE	P
cool	normal	FALSE	P
cool	normal	TRUE	N
mild	normal	FALSE	P
mild	high	TRUE	N

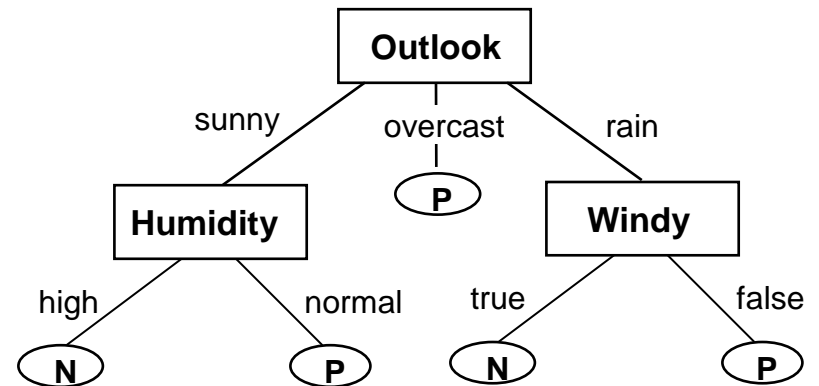


Gain(Temperature) = 0.02

Gain(Humidity) = 0.020

Gain(Windy) = 0.971

∴ Windy is chosen as the root



When to stop building the tree

- We built a decision tree by **recursively splitting our dataset**
 - Each split was performed by choosing the best feature to split (using Error rate, Gini index, or Entropy)
 - We stop when the leaf nodes is pure (i.e., all its instances have the same label)
- To avoid overfitting the stop condition can be any of the following:
 - Stop building the tree after you reach a certain depth
 - Don't split a node if the change in Error rate, Gini index, or Entropy is below some threshold
 - Don't split a node if it has less than a certain number of instances
 - Split a node only if both of the resulting leaves contain at least a certain number of instances

Decision Tree Algorithm

1. Choose the Best Feature to Split On:

- Based on an **Attribute Selection Measure (ASM)** such as **Classification error, Gini impurity, Entropy**

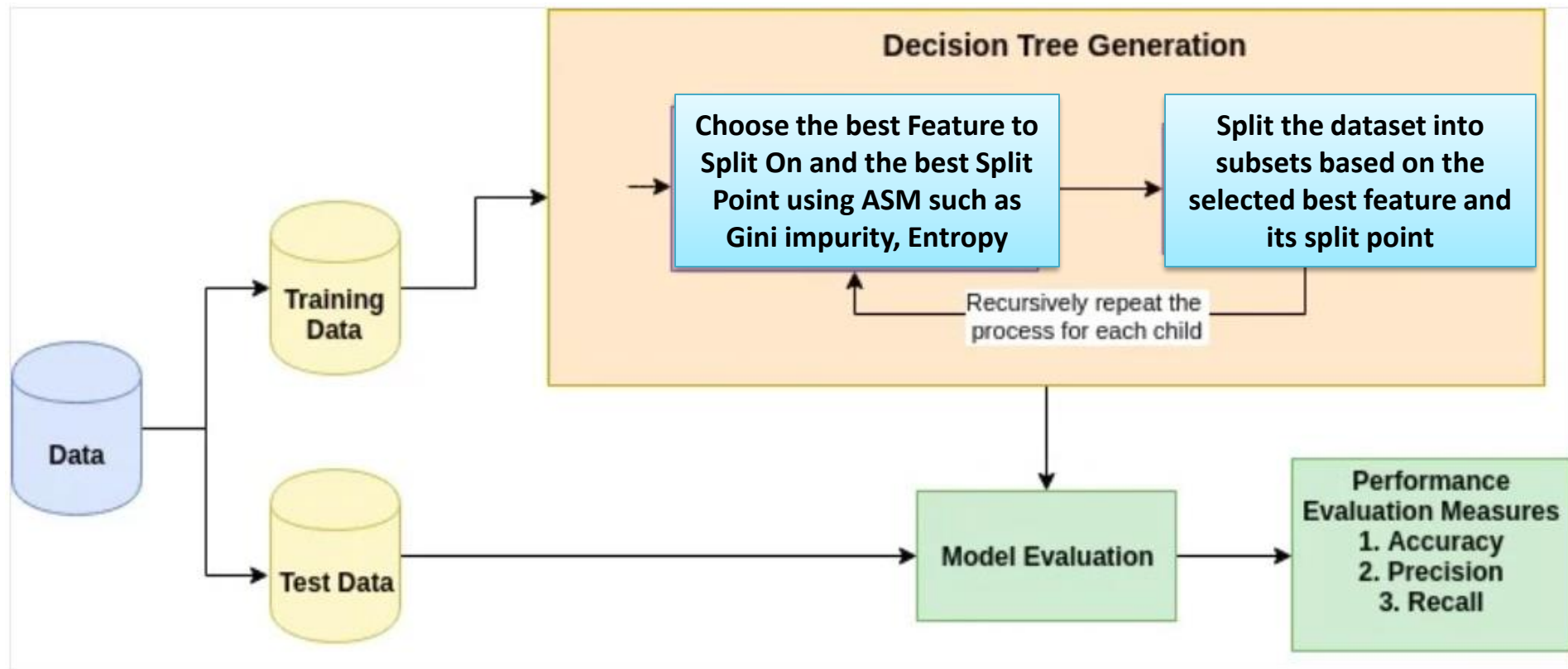
2. Determine the Split Point:

- Make the selected feature a **decision node** then find the **optimal Split Point (i.e., splitting condition/threshold)** that minimizes the impurity (using the same ASM used in step 1)

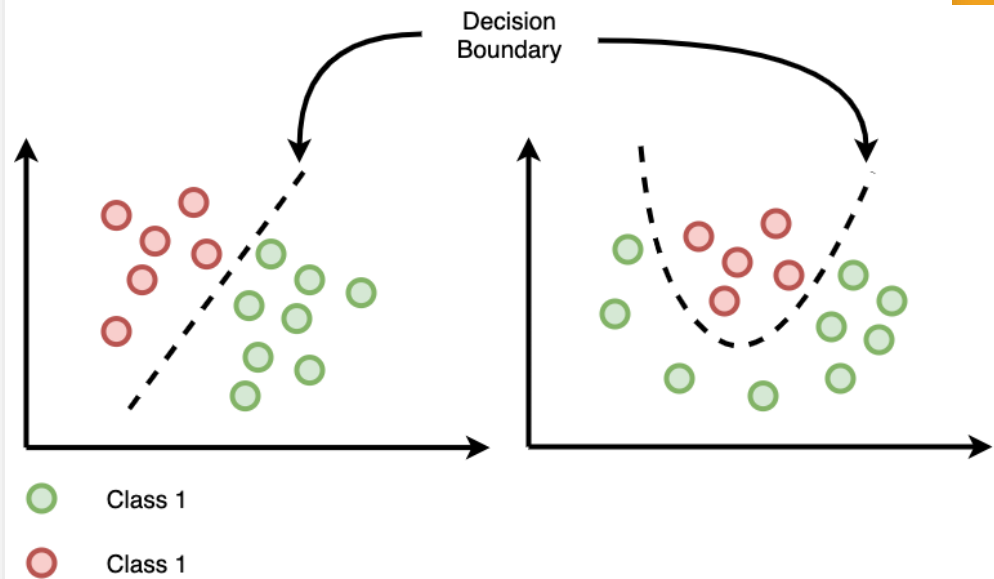
3. Recursively partition the dataset until a stop condition:

- Recursively split the dataset into subsets based on the selected best feature and its split point. Continues **until a stopping condition is met**
 - Common stopping criteria include limiting the maximum depth of the tree, minimum number of instances per leaf, or no further improvement in impurity reduction

Decision tree algorithm

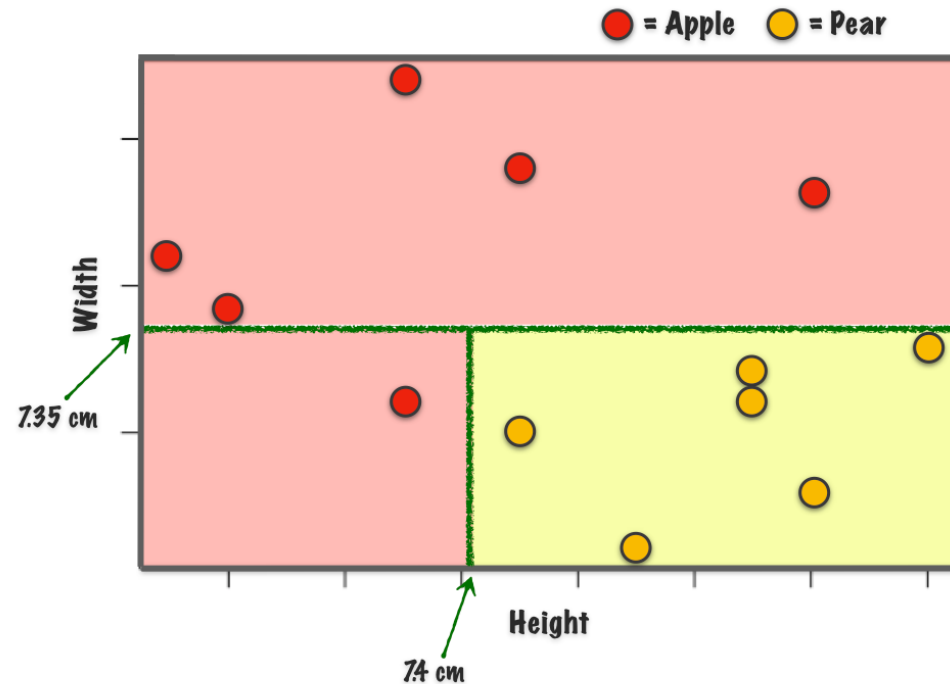
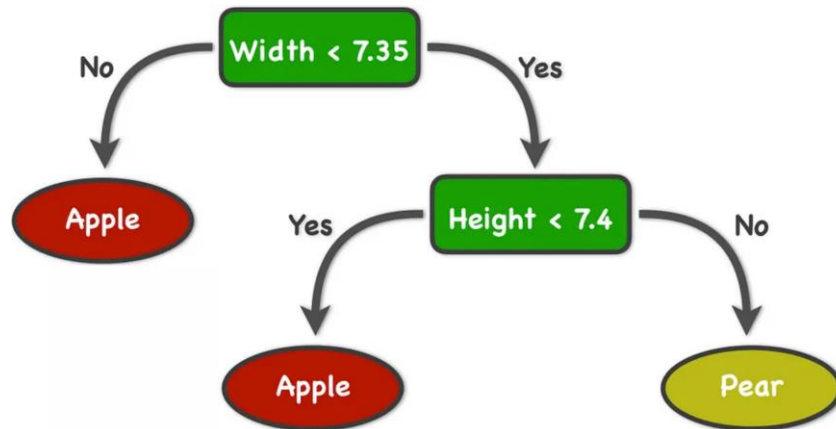


Decision Boundaries

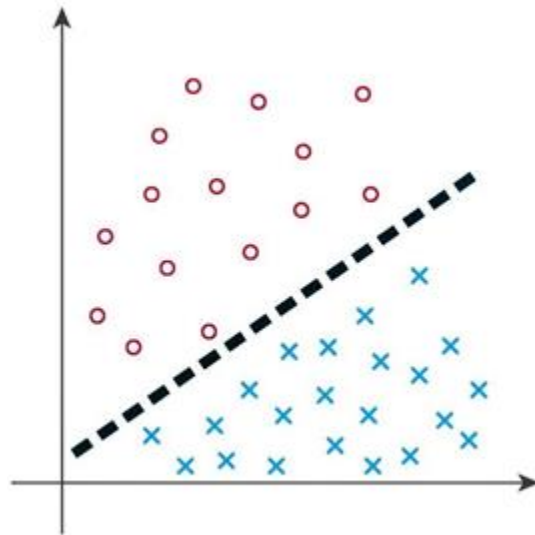


Decision tree constructs decision boundaries

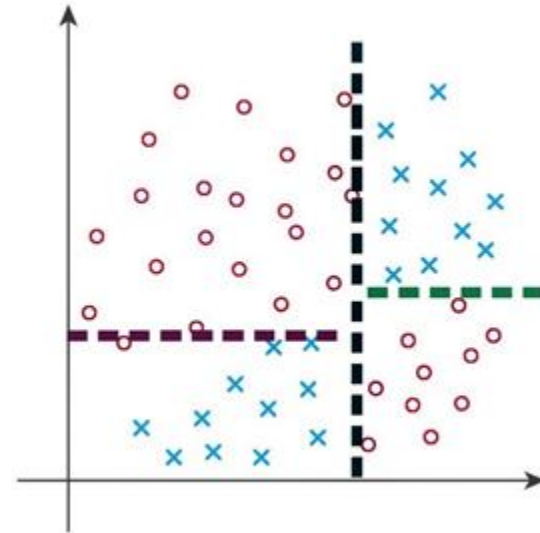
- A visual representation of a decision tree and how this translates into 2D feature space with **decision boundaries**
- The splits divide the plane into a number of boxes with each one corresponding to a classification (apple/pear) for an observation



Decision tree constructs its decision boundaries to fit the linearly inseparable dataset



Linearly separable dataset

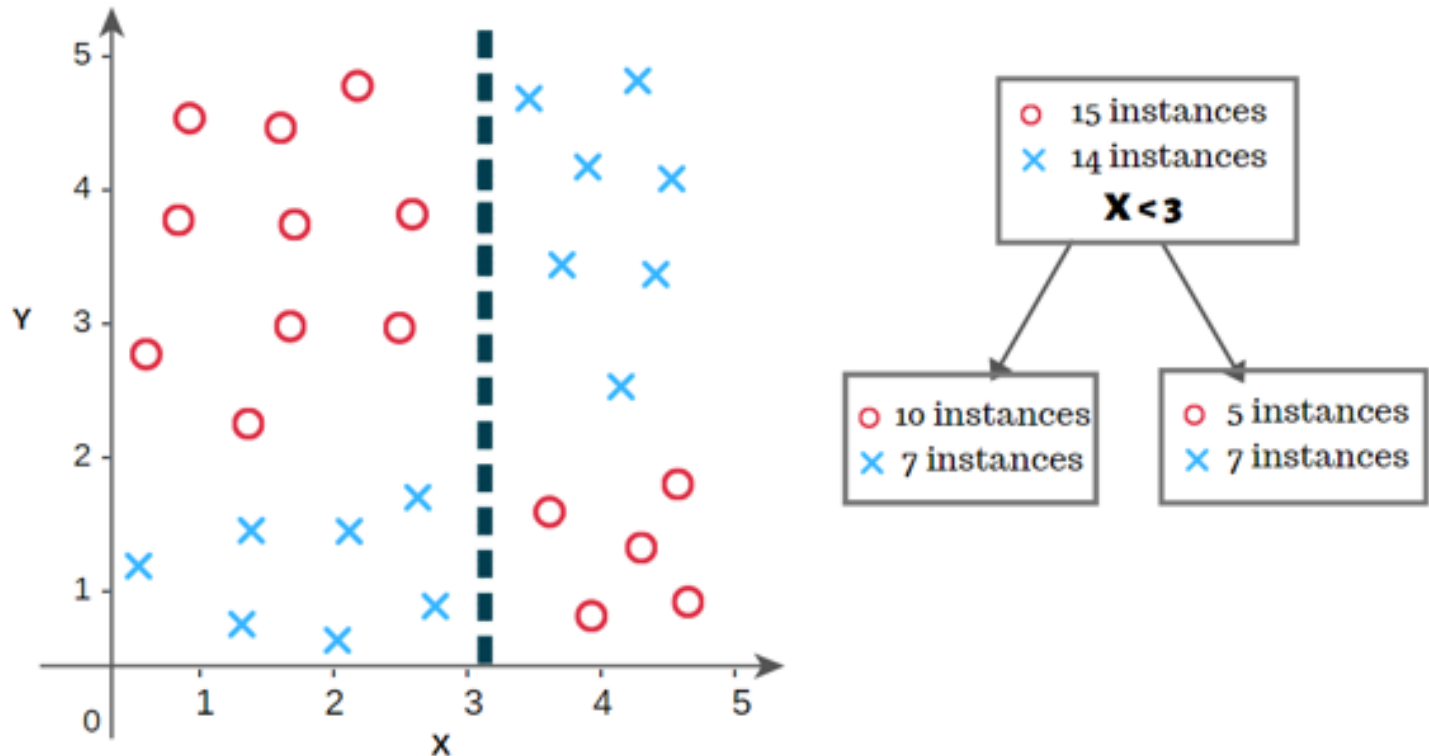


Linearly inseparable dataset

Source [link](#)

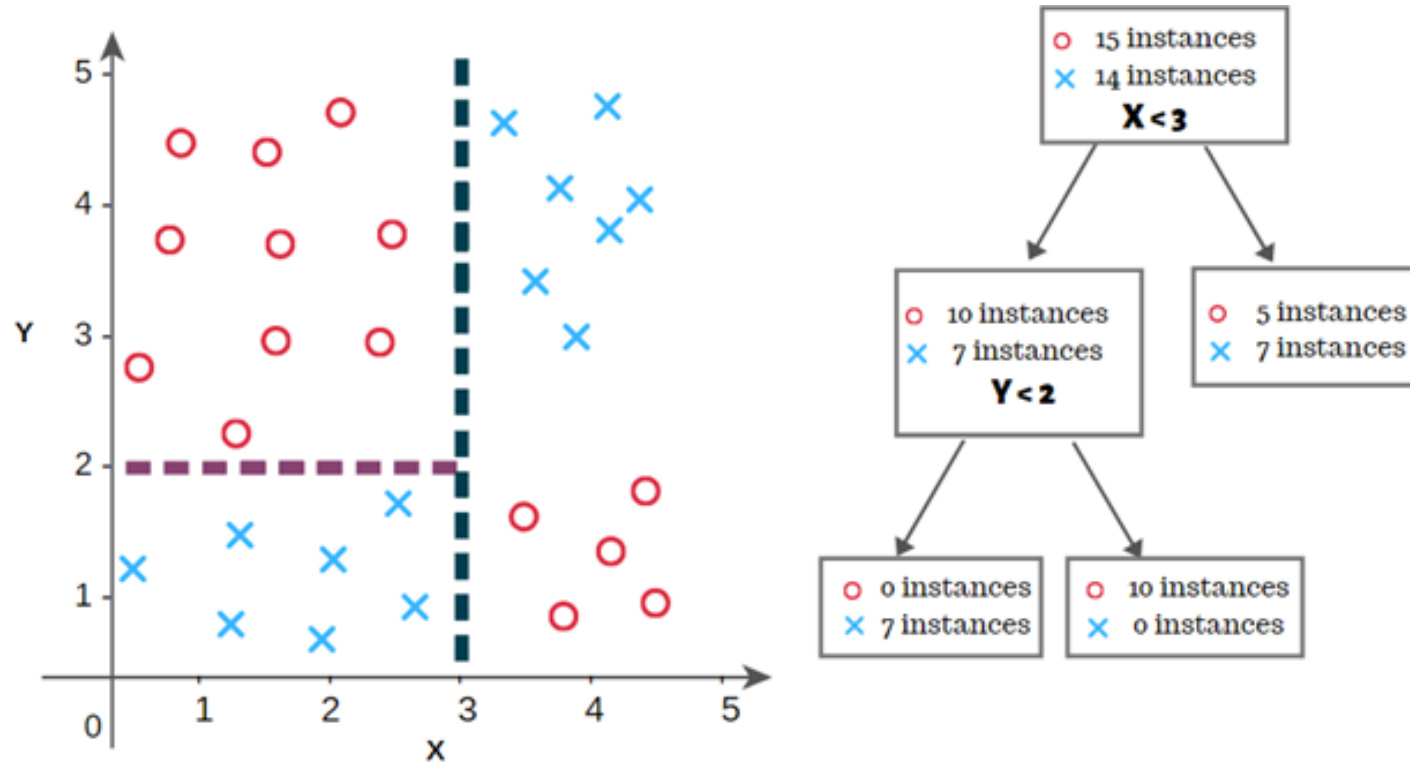
- Decision trees can, unlike linear models, fit **linearly inseparable datasets**
 - A linearly inseparable dataset is one where data points of different classes cannot be separated by a single line, as opposed to linearly separable where a single line is enough
 - DT relies on **recursive partitioning** of the datasets into “homogeneous” (pure) regions

$X < 3$ as our Split Point



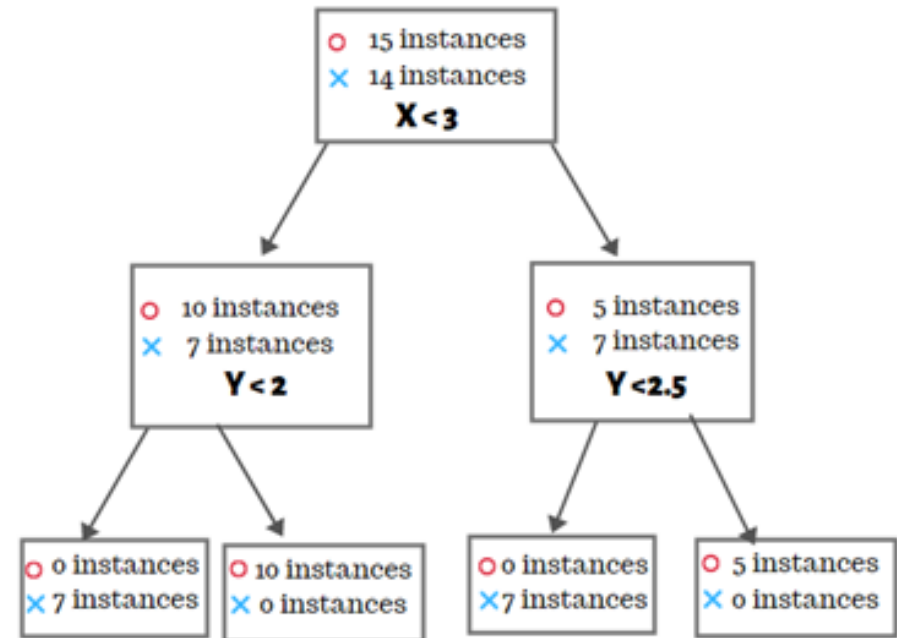
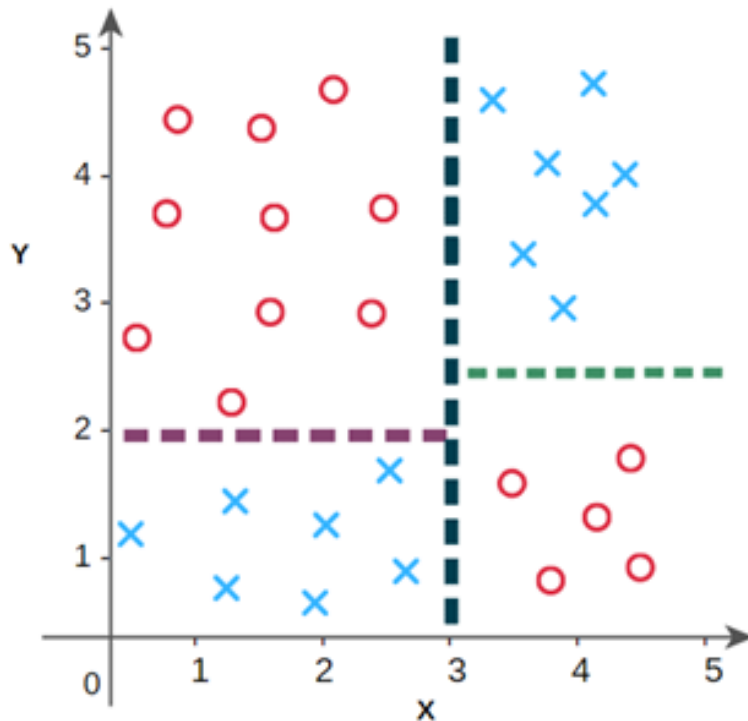
- Taking $X=3$ as our split point, yields 2 decision regions represented as child-nodes of the tree where each split contains the number of respective instances

$Y < 2$ as our Split Point for the Left region



- We'll split the left region of the resultant graph at $Y=2$
- This yields 2 decision regions represented as child-nodes of the tree. The split resulted into 2 pure leaf nodes => no further split needed for the left-side of the tree

$Y < 2.5$ as our Split Point for the Right region



- We'll split the right region of the resultant graph at $Y=2.5$
- This yields 2 decision regions represented as child-nodes of the tree. The split resulted into 2 pure leaf nodes => no further split needed for the right-side of the tree

Decision Boundaries

```
In [27]: import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.inspection import DecisionBoundaryDisplay

# Parameters
n_classes = 3
plot_colors = "ryb"
plot_step = 0.02

pair_of_columns = [2, 3]

# We only take the two corresponding features
X = iris.data.values[:, pair]
y = iris.target
# Train
tree_clf = DecisionTreeClassifier().fit(X, y)

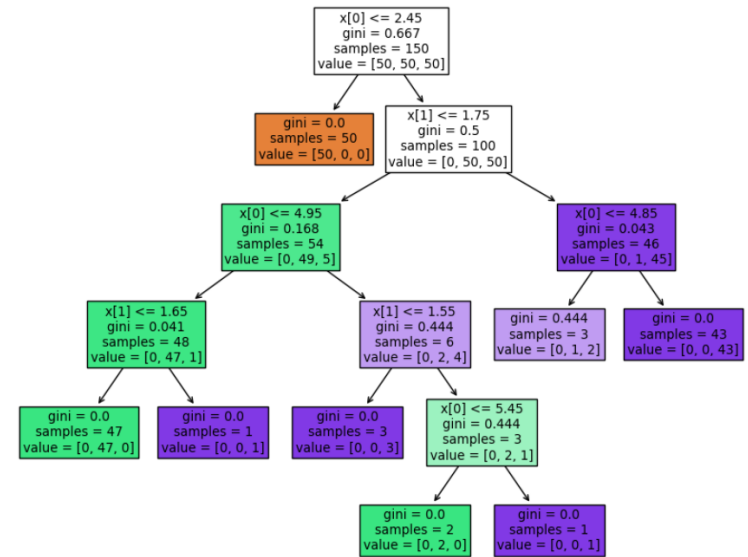
plt.figure(figsize=(10,8))
plot_tree(tree_clf, filled=True)
plt.title("Decision tree trained on two attributes")
plt.show()
```

```
ax = plt.subplot(1, 1, 1)
# Plot the decision boundary
DecisionBoundaryDisplay.from_estimator(tree_clf, X, cmap=plt.cm.RdYlBu, response_method="predict",
ax=ax,
xlabel=iris.feature_names[pair[0]],
ylabel=iris.feature_names[pair[1]],
)

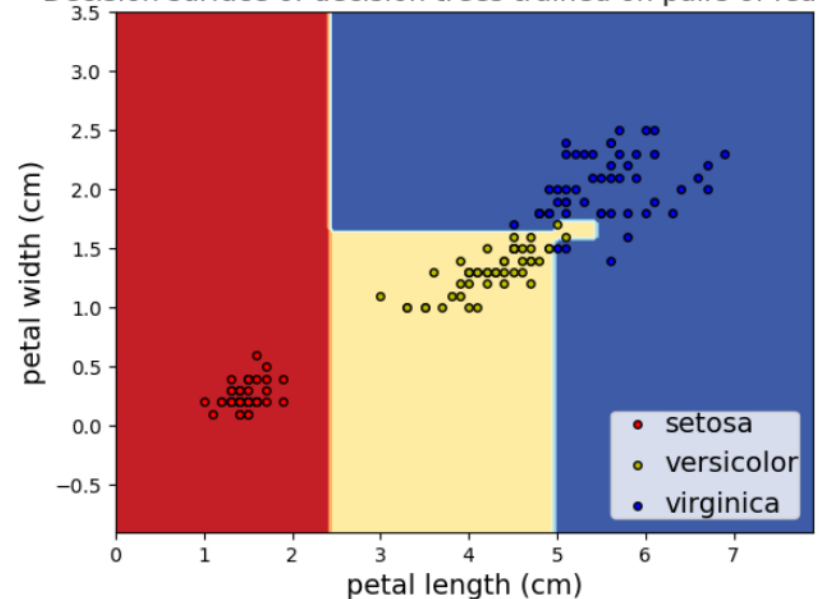
# Plot the training points
for i, color in zip(range(n_classes), plot_colors):
    idx = np.where(y == i)
    plt.scatter(X[idx, 0], X[idx, 1], c=color, label=iris.target_names[i],
cmap=plt.cm.RdYlBu, edgecolor="black", s=15,)

plt.title("Decision surface of decision trees trained on pairs of features")
plt.legend(loc="lower right", borderpad=0, handletextpad=0)
plt.show()
```

Decision tree trained on two attributes



Decision surface of decision trees trained on pairs of features



Summary

- A DT is a graphical representation of a set of rules for predicting or classifying new instances
- A decision tree is constructed from a set of training examples by using a tree construction algorithm such as CHAD, ID3, ...
- These algorithms differ mainly on the attribute selection measure used; GINI Index, Information gain, ...

Summary

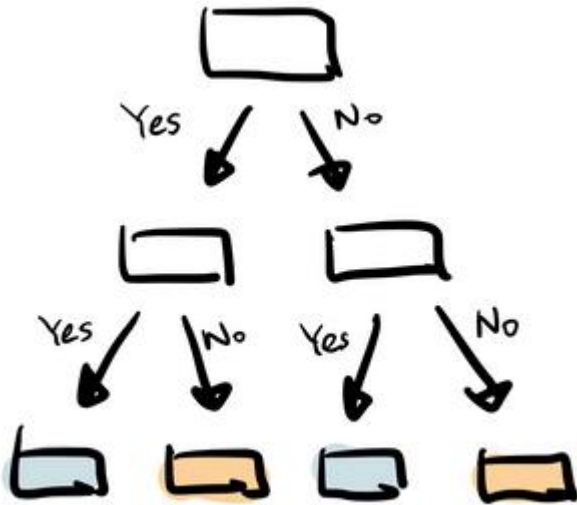
- **Pros:**

- Resulting tree is often simple and easy to interpret
- Relatively inexpensive to construct
- More closely mirror human decision-making
- Variable selection & reduction is automatic

- **Cons:**

- The greedy approach (makes the locally optimal choice at each step) does not guarantee best solution: Since the process deals with one variable at a time, no way to capture interactions between variables
- Limited expressiveness: May not perform well where there is structure in the data that is not well captured by horizontal or vertical splits
- Non robust: sensitive to small changes in the data

Decision Trees & Random Forest

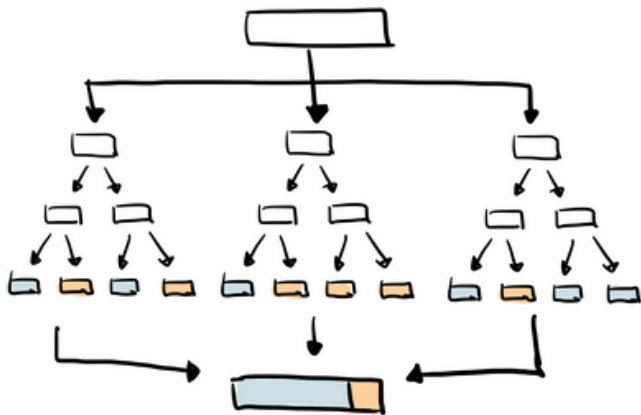


- **Decision Trees:**

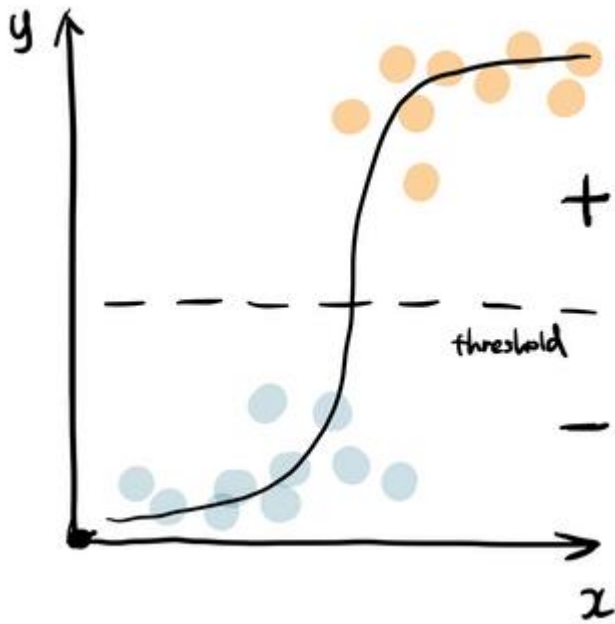
- A tree-like model where each internal node represents a "test" on an attribute
- It splits the data into different branches based on the attribute values
- Decision trees are interpretable and can handle both numerical and categorical data

- **Random Forest:**

- A collection of decision trees where each tree is built using a random subset of features and a random subset of the training data
- It reduces overfitting and improves generalization compared to individual decision trees
- Random forests are robust and perform well on a variety of datasets

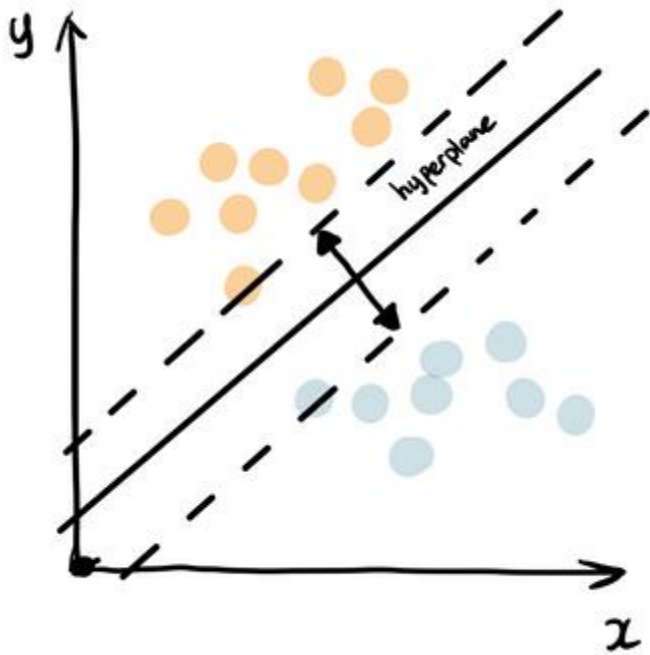


Logistic Regression



- A linear model used for **binary** classification problems
- It models the probability that a given input belongs to a certain class using the logistic function
- It's simple, interpretable, and efficient for linearly separable data

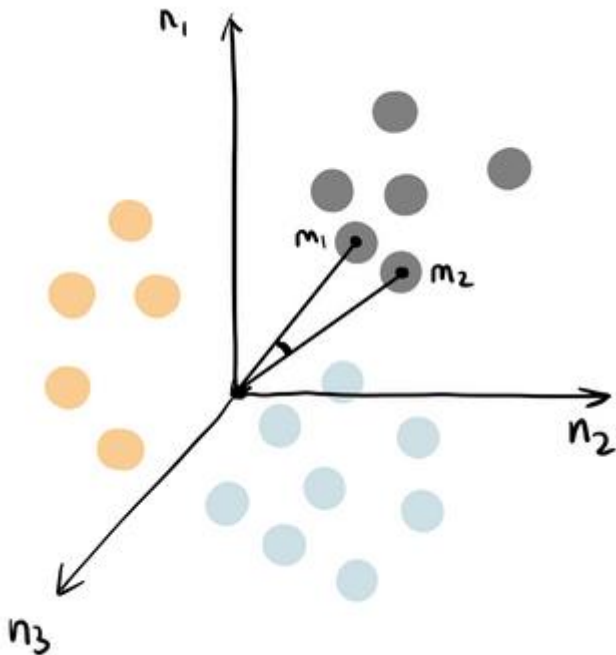
Support Vector Machine (SVM)



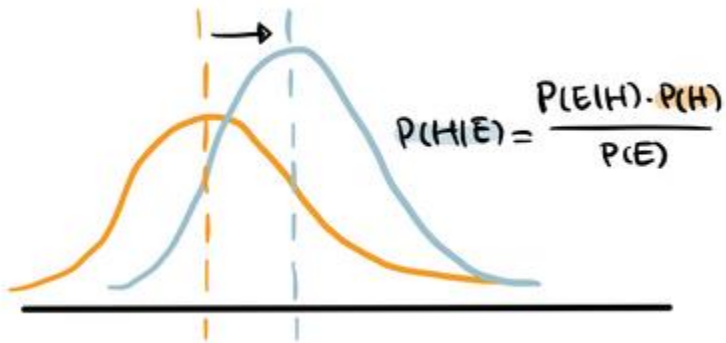
- SVM is a model that finds the optimal hyperplane separating different classes in the feature space
 - Classify the data based on the position in relation to the hyperplane between positive class and negative class
- SVM aims to maximize the margin between classes, thus enhancing generalization
- It can handle both linear and non-linear classification tasks using different kernel functions

K-Nearest Neighbors (KNN)

- Each data point is represented in a n dimensional space, which is defined by n features
 - And it calculates the distance between one point to another, then assign the label of unobserved data based on the labels of nearest observed data points
 - The classification of a data point is determined by the majority class among its k nearest neighbors in the feature space
- KNN is simple to understand and implement, especially for small datasets
- It does not learn explicit models and can be sensitive to the choice of k



Naive Bayes



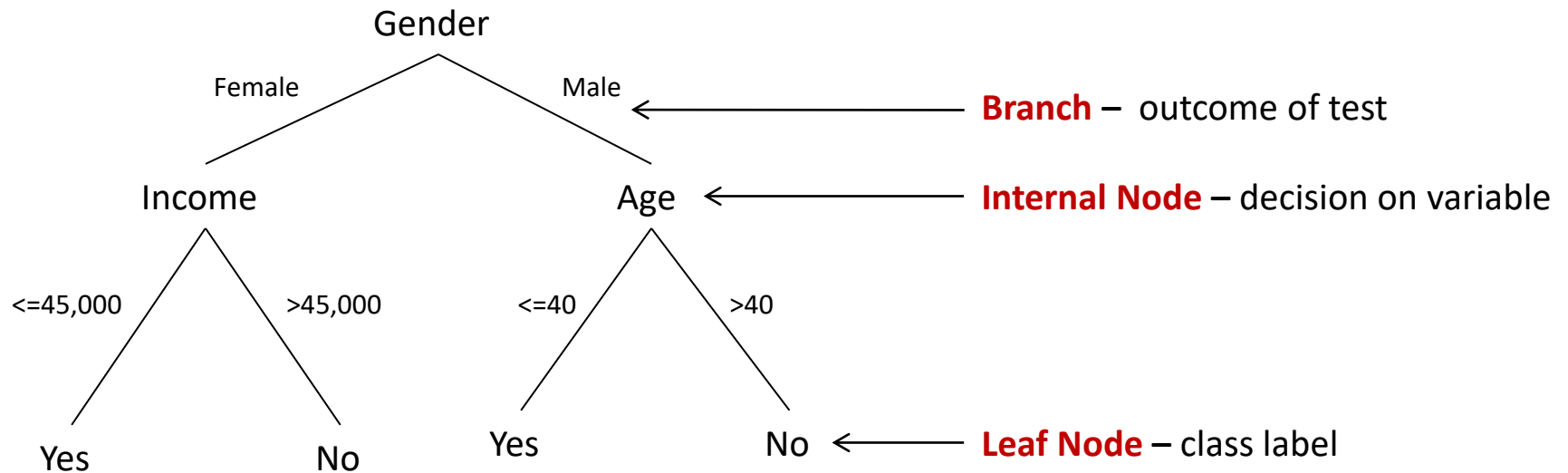
- A probabilistic classifier based on Bayes' theorem with an assumption of independence between features
- It calculates the probability of each class given a set of features and selects the class with the highest probability
- Naive Bayes is efficient, especially for text classification and other high-dimensional datasets

Decision Trees Induction

- **Input** variables can be continuous or discrete
- **Output:**
 - A tree that describes the decision flow.
 - Leaf nodes return **class labels** and, in some implementations, they also return the **probability scores**.
 - Trees can be converted to a set of "**decision rules**"
 - "IF Refund=Yes AND Marital_Status=No THEN Cheat=No with 75% probability"

Decision Tree

Example of Visual Structure



Example of a Decision Tree

Training Data

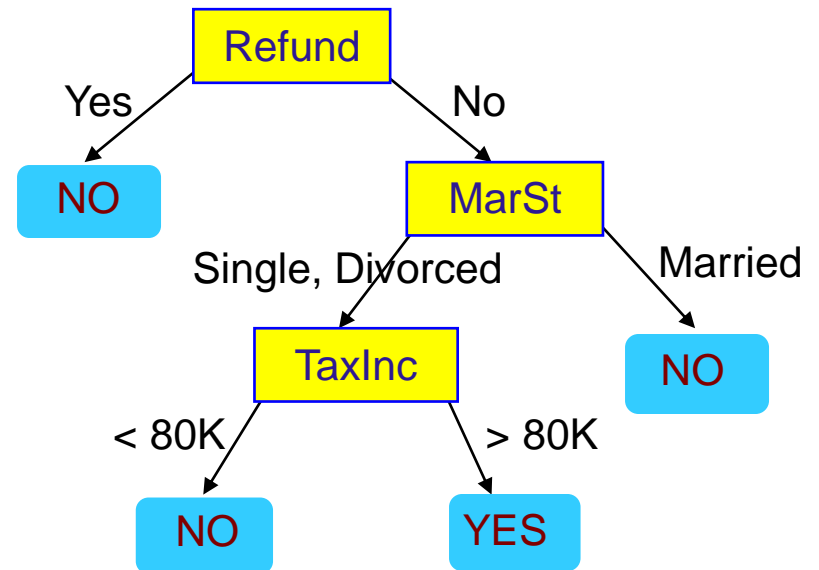
categorical categorical continuous class

<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Descriptive Attributes

Class attribute

Model: Decision Tree



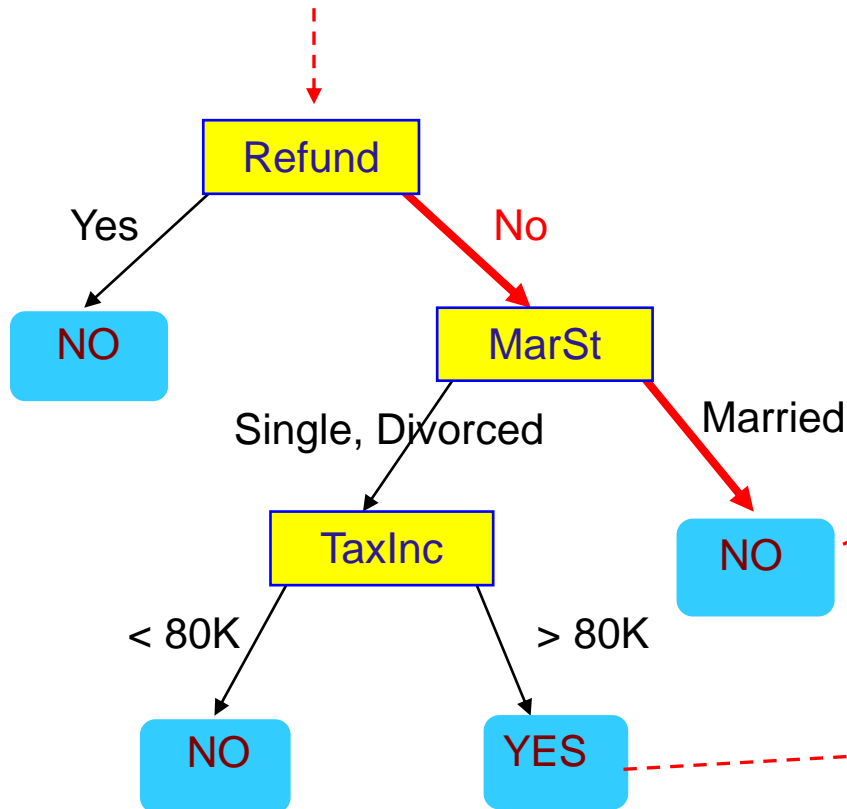
Internal Nodes: Splitting Attributes

Links: Attribute values

Leaf Nodes: Class labels

Apply Model to Test Data

Start from the root of tree.



Test Data

Assign Cheat to "No"

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to "Yes"

Refund	Marital Status	Taxable Income	Cheat
No	Single	90K	?

Another Example of a Decision Tree

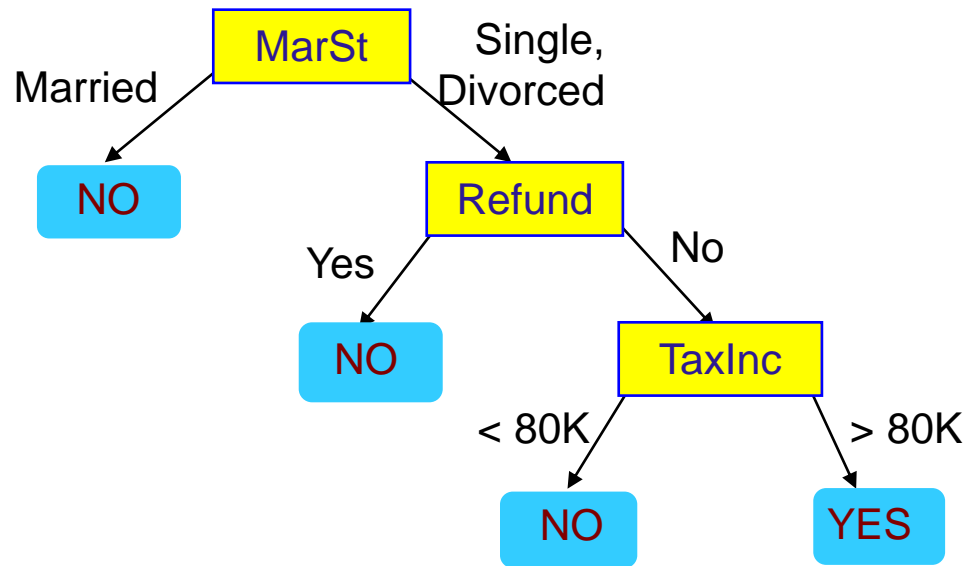
<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

categorical

categorical

continuous

class



There could be more than one tree that fits the same data!

Decision Tree Induction

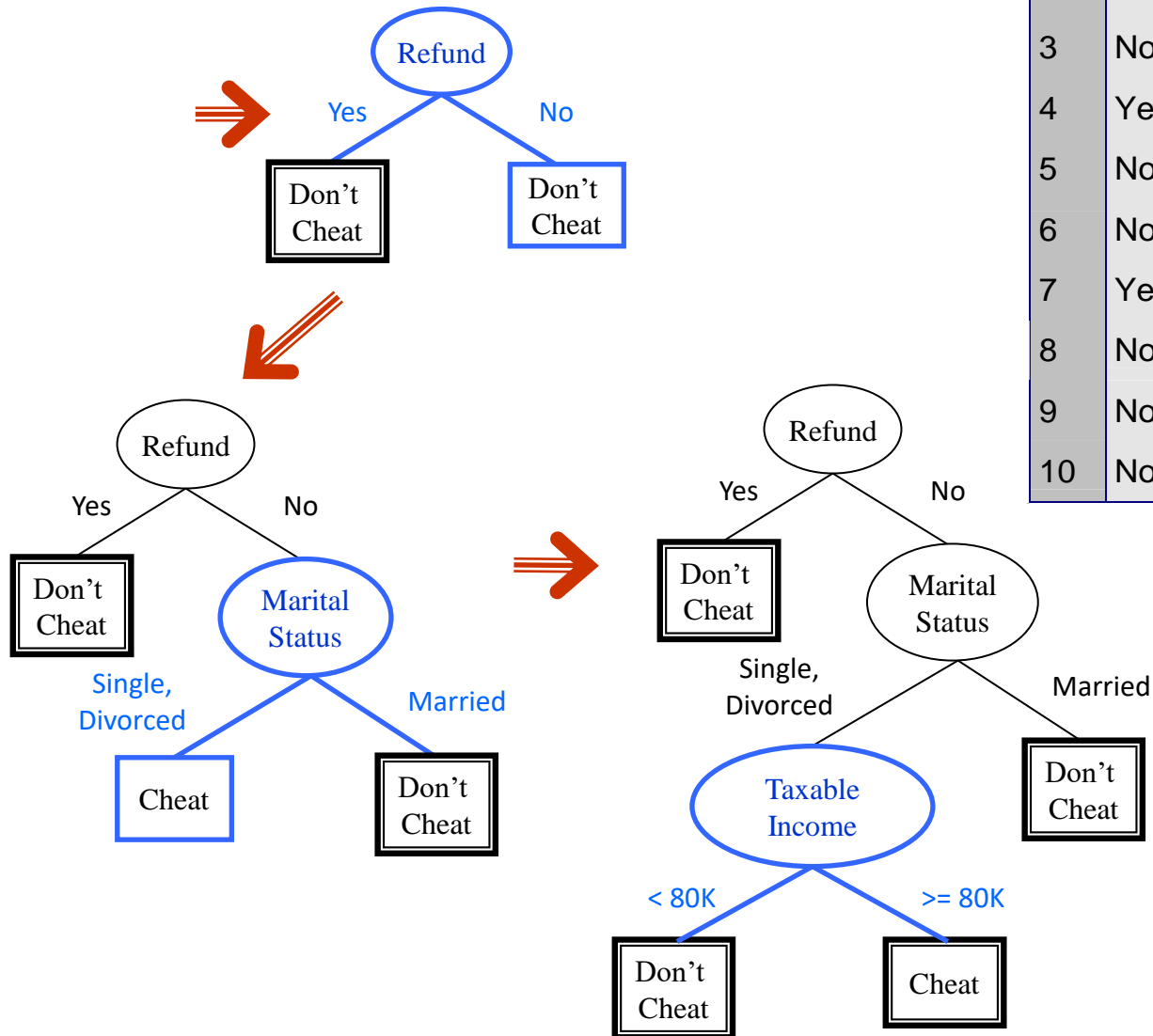
- Algorithms:
 - CART
 - ID3, C4.5, C5
 - CHAID
 - C-SEP
 - G-statistics
 - SLIQ
 - SPRINT
 - ...others

Decision Tree Induction

- **Principle of Tree Construction**

1. If all examples are of the **same class**, create a **leaf node** labelled by the class
2. If examples in the training set are of different classes, determine which attribute should be **selected** as the **root** of the current tree
3. Partition the input examples into **subsets** according to the values of the selected root attribute
4. Construct a decision tree **recursively** for each subset
5. Connect the roots for the subtrees to the root of the whole tree via **labelled links**

Example



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

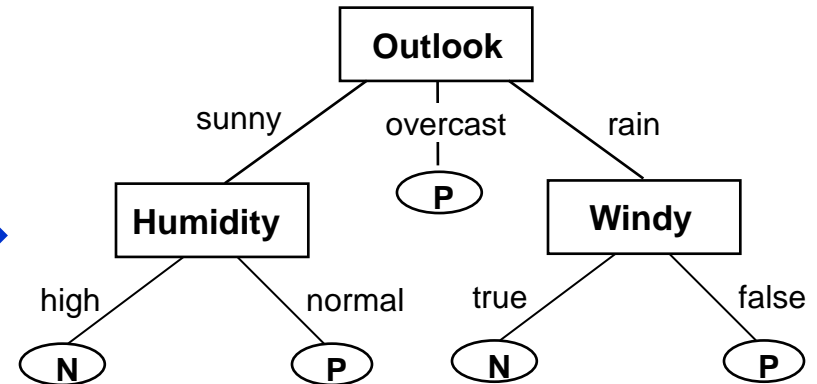
Example

Input Training Examples

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	FALSE	N
sunny	hot	high	TRUE	N
overcast	hot	high	FALSE	P
rain	mild	high	FALSE	P
rain	cool	normal	FALSE	P
rain	cool	normal	TRUE	N
overcast	cool	normal	TRUE	P
sunny	mild	high	FALSE	N
sunny	cool	normal	FALSE	P
rain	mild	normal	FALSE	P
sunny	mild	normal	TRUE	P
overcast	mild	high	TRUE	P
overcast	hot	normal	FALSE	P
rain	mild	high	TRUE	N

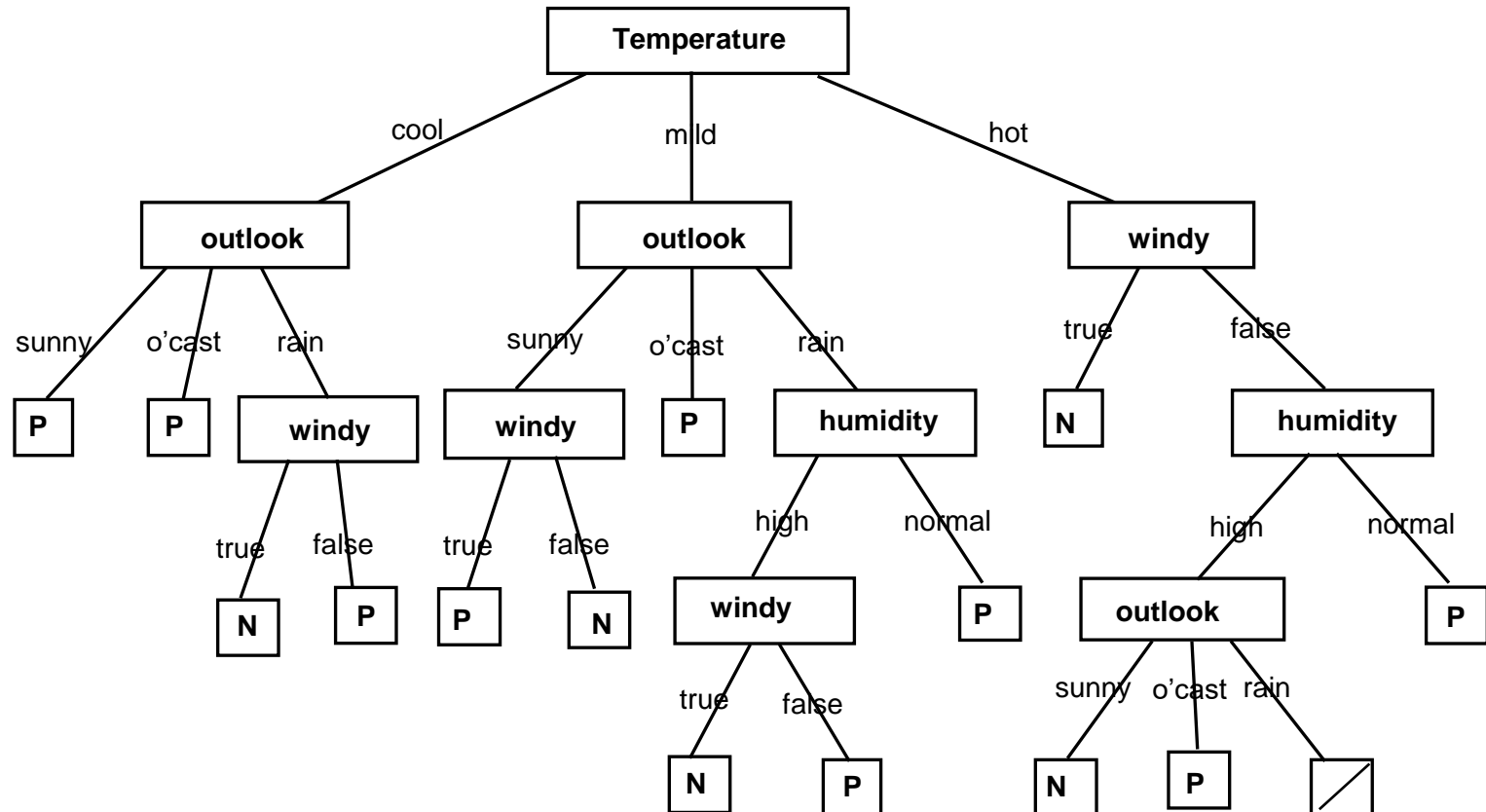
Descriptive Attributes

Class attribute



Decision Tree Induction

- What could random selection of attributes produce?



Tree Construction Algorithm

Algorithm TreeConstruct (C: Training Set) : Decision Tree

begin

Tree := \emptyset ;

if C is not empty **then**

if all examples in C are of one class **then**

 Tree := a leaf node labelled by the class tag

else begin

select attribute T according to an attribute selection measure as the root;

 partition C into C_1, C_2, \dots, C_w by values of T ;

for each C_i ($1 \leq i \leq w$) **do**

$t_i \leftarrow \text{TreeConstruct}(C_i)$;

 Tree := a tree T as root and t_1, t_2, \dots, t_w as subtrees

 label the links from T to the subtrees with values of T

end;

 return(Tree);

end;

Continuous Attributes: Computing Gini Index...

- For efficient computation: for each attribute,
 - Sort the attribute on values
 - Linearly scan these values, each time updating the count matrix and computing gini index
 - Choose the split position that has the least gini index

		Cheat																							
		No		No		No		Yes		Yes		Yes		No		No		No		No					
		Taxable Income																							
Sorted Values	→	60		70		75		85		90		95		100		120		125		220					
Split Positions	→	55		65		72		80		87		92		97		110		122		172		230			
		<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>				
		Yes		0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0		
		No		0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
		Gini		0.420		0.400		0.375		0.343		0.417		0.400		<u>0.300</u>		0.343		0.375		0.400		0.420	

Tree Induction

- Greedy strategy.
 - Split the records based on an attribute test that optimizes certain criterion.
- Issues
 - Determine how to split the records
 - How to specify the attribute test condition?
 - How to determine the best split?
 - Determine when to stop splitting

How to Specify Test Condition?

- Depends on **attribute types**
 - Nominal
 - Ordinal
 - Continuous
- Depends on **number of ways to split**
 - 2-way split
 - Multi-way split

Attribute Selection Measures

- **Information Gain (ID3 Algorithm)**

- An information system:
 - A set of n possible events E_1, E_2, \dots, E_n ;
 - Each event may occur with a probability $p(E_k)$ ($1 \leq k \leq n$)
 - $p(E_1) + p(E_2) + \dots + p(E_n) = 1$.
- Given a training set, each attribute can be considered as such an information system (e.g. Outlook).
- Classification involves two information systems:
 - **Attribute A (e.g. Outlook)**: events: sunny, overcast and rain;
event probabilities: $p(\text{sunny}) = 5/14$, $p(\text{overcast}) = 4/14$, $p(\text{rain}) = 5/14$
 - **Class attribute**: events: P and N;
event probabilities: $p(P) = 9/14$, $p(N) = 5/14$

Attribute Selection Measures

- **Information Gain**

- **Self-information** of event E: the amount of information being conveyed when E occurs, defined as:

$$I(E) = \log \frac{1}{p(E)} = -\log p(E)$$

If someone says “I have something” This does not provide any information. $I(E) = 0$

If we know that someone has a coin, and they say “we have a coin” again, no information

“I was born in a day” No information

“I was born in January” Now this provided us with some information. $I(E) = \log(1/(1/12))$, 1 out of twelve months

“I have a number of two digits, and it is 15 .. $I(E) = \log(1/(1/100))$

Chi-square (χ^2) statistic

- χ^2 : measure for the degree of association/dependence between two **categorical** variables; a given attribute and the class variable.
- Given N examples of w classes, C_1, C_2, \dots, C_w and an attribute A of v values, a_1, a_2, \dots, a_v , the (χ^2) function is defined as follows:

$$\chi^2 = \sum_{j=1}^v \sum_{i=1}^w \frac{(x_{ij} - E_{ij})^2}{E_{ij}}$$

where x_{ij} is the actual frequency that examples have attribute value a_j and class C_i , and E_{ij} is the expected frequency.

with the assumption that the class and the attribute are not dependant on each other, $E_{ij} = \frac{n_i \times n_j}{N}$, where

n_i is the number of records with class C_i

n_j is the number of records with attribute A with value a_j

Chi-square (χ^2) statistic

$$n_P = 9$$

of records with class P

$$\chi^2 = \sum_{j=1}^v \sum_{i=1}^w \frac{(x_{ij} - E_{ij})^2}{E_{ij}}$$

$$n_{\text{sunny}} = 5$$

of records with Attribute=sunny

$$x_{P,\text{sunny}} = 2$$

$$E_{P,\text{sunny}} = \frac{n_P \times n_{\text{sunny}}}{n} = \frac{9 \times 5}{14}$$

- $C = \{P, N\}$.. $A = \{\text{sunny, overcast, rain}\}$

$$\chi^2(\text{Outlook}) = \left(\frac{\left(\left(2 - \frac{9 \times 5}{14} \right)^2 \right)}{\left(\frac{9 \times 5}{14} \right)} + \frac{\left(\left(3 - \frac{5 \times 5}{14} \right)^2 \right)}{\left(\frac{5 \times 5}{14} \right)} \right) + \left(\frac{\left(\left(4 - \frac{9 \times 4}{14} \right)^2 \right)}{\left(\frac{9 \times 4}{14} \right)} + \frac{\left(\left(3 - \frac{9 \times 5}{14} \right)^2 \right)}{\left(\frac{9 \times 5}{14} \right)} + \frac{\left(\left(2 - \frac{5 \times 5}{14} \right)^2 \right)}{\left(\frac{5 \times 5}{14} \right)} \right) \approx 29.653$$

- Similarly,

- $\chi^2(\text{Temperature}) \approx 7.985$,
- $\chi^2(\text{Humidity}) \approx 39.2$
- $\chi^2(\text{Windy}) \approx 13.067$

- Thus, Humidity is chosen as the root of the tree.

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	FALSE	N
sunny	hot	high	TRUE	N
overcast	hot	high	FALSE	P
rain	mild	high	FALSE	P
rain	cool	normal	FALSE	P
rain	cool	normal	TRUE	N
overcast	cool	normal	TRUE	P
sunny	mild	high	FALSE	N
sunny	cool	normal	FALSE	P
rain	mild	normal	FALSE	P
sunny	mild	normal	TRUE	P
overcast	mild	high	TRUE	P
overcast	hot	normal	FALSE	P
rain	mild	high	TRUE	N

Attribute Selection Measures

- **Other Attribute Selection Measures**

- **Information Gain Ratio**

- $\text{GainRatio}(A) = \text{Gain}(A)/H(A)$, e.g. $\text{GainRatio}(\text{Outlook}) \approx 0.156$
 - Used to avoid (by normalization) bias towards attributes with many values

- **Gini Index of Impurity**

- Use Gini impurity function: Based on the concept of reducing impurity (mixture of different classes) of data set by selecting a right attribute

- **Chi-square (χ^2) Statistic**

- Based on the concept of measuring the degree of dependence of class to a chosen attribute

- **and others..**

Attribute Selection Measures

- **Entropy**

- The average of the self-information of all events in an information system S, known as *entropy*, is defined as: **The expected amount of information (entropy)**

For any random variable S that follows a probability distribution P

$$H(S) = \sum_{k=1}^N p(E_k) \cdot I(E_k) = - \sum_{k=1}^N p(E_k) \cdot \log p(E_k)$$

H(S) represents degree of uncertainty. When one event always occurs, i.e. the least uncertain, $H(S) = 0$. When all events have equal chance to occur, i.e. the most uncertain, $H(S) = \log N$.

C1	0
C2	16
Entropy $H(S)=0.000$	

C1	4
C2	12
Entropy $H(S) = 0.811$	

C1	6
C2	10
Entropy $H(S) = 0.9644$	

C1	8
C2	8
Entropy $H(S) = 1$	

- Entropy will be low if a message is highly predictable.
- On the other hand, if the message is highly unpredictable, then the entropy will be high.

Attribute Selection Measures

- **Information Gain (cont'd)**

- **Conditional self-information** of event E of system S_1 given that event F of system S_2 has occurred is defined as:

$$I(E | F) = \log \frac{1}{p(E | F)} = -\log p(E | F) = -\log \frac{p(E \cap F)}{p(F)}$$

- The **expected information** of system S_1 in the presence of system S_2 , is defined as the average of conditional self-information of all events in S_1 :

$$H(S_1 | S_2) = \sum_{i=1}^n \sum_{j=1}^m p(E_i \wedge F_j) \cdot I(E_i | F_j) = - \sum_{i=1}^n \sum_{j=1}^m p(E_i \wedge F_j) \cdot \log \frac{p(E_i \wedge F_j)}{p(F_j)}$$

Attribute Selection Measures

- **Information Gain (cont'd)**

- The **information gain**, also known as **mutual information** between S_1 and S_2 , is defined as

$$Gain(S_1) = H(S_1) - H(S_1 | S_2)$$

- Rationale behind information gain:

- $H(S)$ represents degree of uncertainty. When one event always occurs, i.e. the least uncertain, $H(S) = 0$. When all events have equal chance to occur, i.e. the most uncertain, $H(S) = \log N$.
- $Gain(S_1)$ signifies a reduction of uncertainty in system S_1 before and after system S_2 is present.
- *In classification, $Gain(A)$ represents a reduction of uncertainty over class attribute before and after attribute A is chosen as the root*

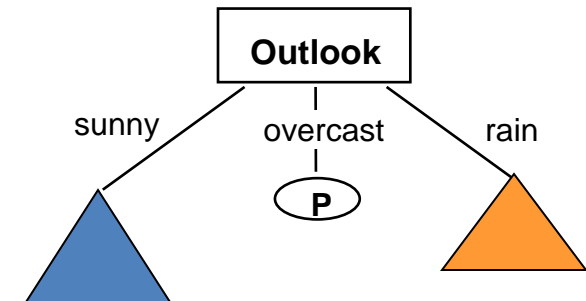
ID3 Algorithm

• Algorithm Illustration

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	FALSE	N
sunny	hot	high	TRUE	N
overcast	hot	high	FALSE	P
rain	mild	high	FALSE	P
rain	cool	normal	FALSE	P
rain	cool	normal	TRUE	N
overcast	cool	normal	TRUE	P
sunny	mild	high	FALSE	N
sunny	cool	normal	FALSE	P
rain	mild	normal	FALSE	P
sunny	mild	normal	TRUE	P
overcast	mild	high	TRUE	P
overcast	hot	normal	FALSE	P
rain	mild	high	TRUE	N

$$\text{Entropy}(S) = - \sum_{i=1}^K p_i \log_2(p_i)$$

Where p_i is the probability of class i in set S .



$$H(S) = - \sum_{k=1}^n p(E_k) \cdot \log p(E_k)$$

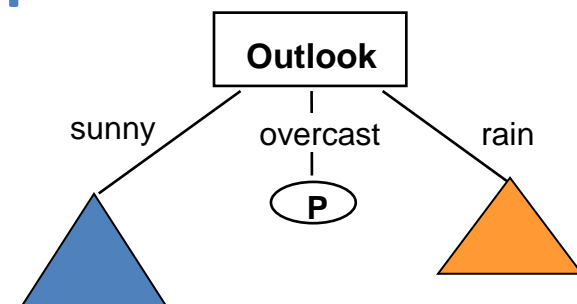
$$E(S) = - \frac{9}{9+5} \cdot \log_2 \frac{9}{9+5} - \frac{5}{9+5} \cdot \log_2 \frac{5}{9+5} \approx 0.94$$

Entropy Calculation - Example

Algorithm Illustration

$$E(S) = -\frac{9}{9+5} \cdot \log_2 \frac{9}{9+5} - \frac{5}{9+5} \cdot \log_2 \frac{5}{9+5} \approx 0.94$$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	FALSE	N
sunny	hot	high	TRUE	N
overcast	hot	high	FALSE	P
rain	mild	high	FALSE	P
rain	cool	normal	FALSE	P
rain	cool	normal	TRUE	N
overcast	cool	normal	TRUE	P
sunny	mild	high	FALSE	N
sunny	cool	normal	FALSE	P
rain	mild	normal	FALSE	P
sunny	mild	normal	TRUE	P
overcast	mild	high	TRUE	P
overcast	hot	normal	FALSE	P
rain	mild	high	TRUE	N



Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	FALSE	N
sunny	hot	high	TRUE	N
sunny	mild	high	FALSE	N
sunny	cool	normal	FALSE	P
sunny	mild	normal	TRUE	P

$$H(S) = \sum_{k=1}^n p(E_k) \cdot I(E_k) = -\sum_{k=1}^n p(E_k) \cdot \log p(E_k)$$

$$H(Class | Outlook = sunny) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} \approx 0.971$$

Outlook	Temperature	Humidity	Windy	Class
overcast	hot	high	FALSE	P
overcast	cool	normal	TRUE	P
overcast	mild	high	TRUE	P
overcast	hot	normal	FALSE	P

$$H(S) = \sum_{k=1}^n p(E_k) \cdot I(E_k) = -\sum_{k=1}^n p(E_k) \cdot \log p(E_k)$$

$$H(Class | Outlook = overcast) = -\frac{4}{4} \log_2 \frac{4}{4} = 0$$

Outlook	Temperature	Humidity	Windy	Class
rain	mild	high	FALSE	P
rain	cool	normal	FALSE	P
rain	cool	normal	TRUE	N
rain	mild	normal	FALSE	P
rain	mild	high	TRUE	N

$$H(S) = \sum_{k=1}^n p(E_k) \cdot I(E_k) = -\sum_{k=1}^n p(E_k) \cdot \log p(E_k)$$

$$H(Class | Outlook = rain) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} \approx 0.971$$

$$H(Class | Outlook) = \frac{5}{14} H(Class | Outlook = sunny) + \frac{4}{14} H(Class | Outlook = overcast) + \frac{5}{14} H(Class | Outlook = rain)$$

$$\searrow H(Class | Outlook) = \frac{5}{14} \times 0.971 + \frac{5}{14} \times 0 + \frac{5}{14} \times 0.971 = 0.694$$

$$Gain(Outlook) = E(S) - E(Outlook) = 0.94 - 0.694 = 0.246$$

ID3 Algorithm

- Algorithm Illustration

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	FALSE	N
sunny	hot	high	TRUE	N
overcast	hot	high	FALSE	P
rain	mild	high	FALSE	P
rain	cool	normal	FALSE	P
rain	cool	normal	TRUE	N
overcast	cool	normal	TRUE	P
sunny	mild	high	FALSE	N
sunny	cool	normal	FALSE	P
rain	mild	normal	FALSE	P
sunny	mild	normal	TRUE	P
overcast	mild	high	TRUE	P
overcast	hot	normal	FALSE	P
rain	mild	high	TRUE	N



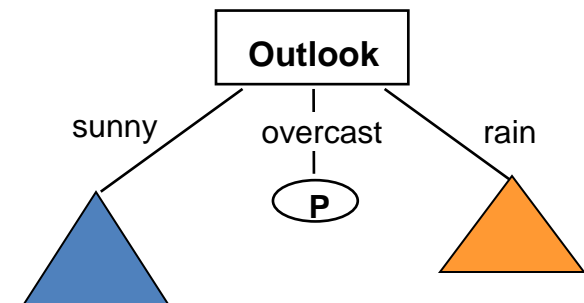
Gain(Outlook) = 0.246 bits

Gain(Temperature) = 0.029 bits

Gain(Humidity) = 0.151 bits

Gain(Windy) = 0.048 bits

=> Outlook is chosen as the root.



ID3 Algorithm

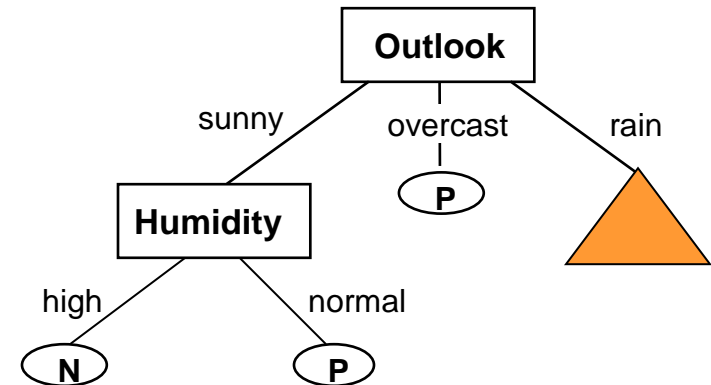
- Algorithm Illustration

Training Set(outlook=sunny)

Temperature	Humidity	Windy	Class
hot	high	FALSE	N
hot	high	TRUE	N
mild	high	FALSE	N
cool	normal	FALSE	P
mild	normal	TRUE	P



$\text{Gain}(\text{Temperature}) = 0.571 \text{ bits}$
 $\text{Gain}(\text{Humidity}) = 0.971 \text{ bits}$
 $\text{Gain}(\text{Windy}) = 0.020 \text{ bits}$
 \therefore Humidity is chosen as the root.



ID3 Algorithm

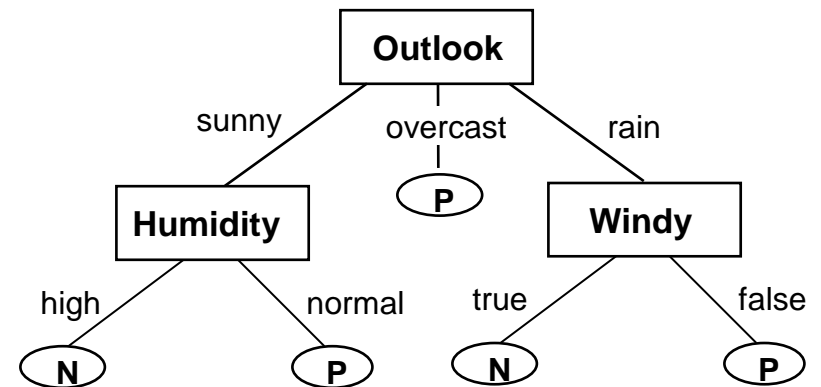
- Algorithm Illustration

Training Set(outlook=rain)

Temperature	Humidity	Windy	Class
mild	high	FALSE	P
cool	normal	FALSE	P
cool	normal	TRUE	N
mild	normal	FALSE	P
mild	high	TRUE	N



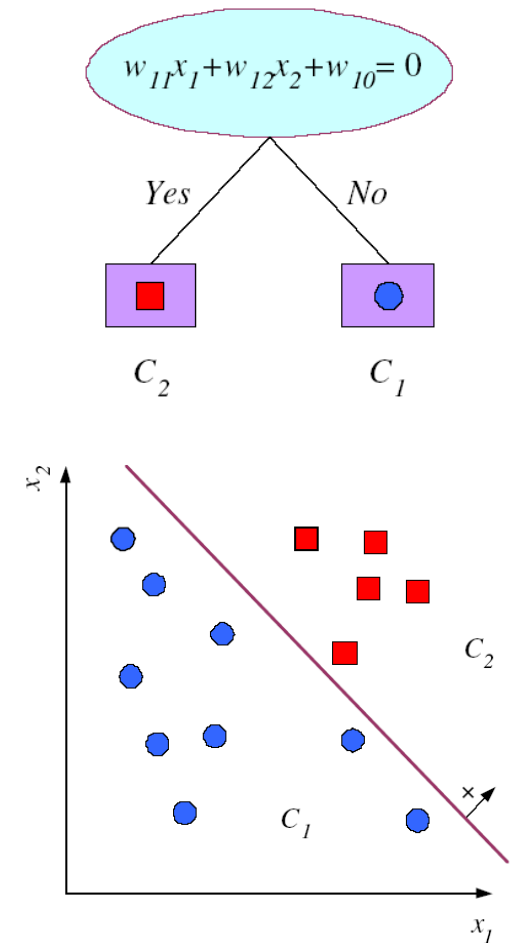
Gain(Temperature) = 0.020 bits
Gain(Humidity) = 0.020 bits
Gain(Windy) = 0.971 bits
∴ Windy is chosen as the root.



Decision Boundaries

(Multivariate/Multi-variable)

- A multi-variable split, in the form of a linear equation* is more general
 - Takes the form:
$$\mathbf{w}_m^T \mathbf{x} + w_0 > 0$$
where \mathbf{w}_m^T is the vector of weights
 - Requires all attributes to be **numeric**
 - CART was the original algorithm for generating multi-variable trees
 - Generating the linear equation is **not cheap**
- (*) Doesn't have to be linear!

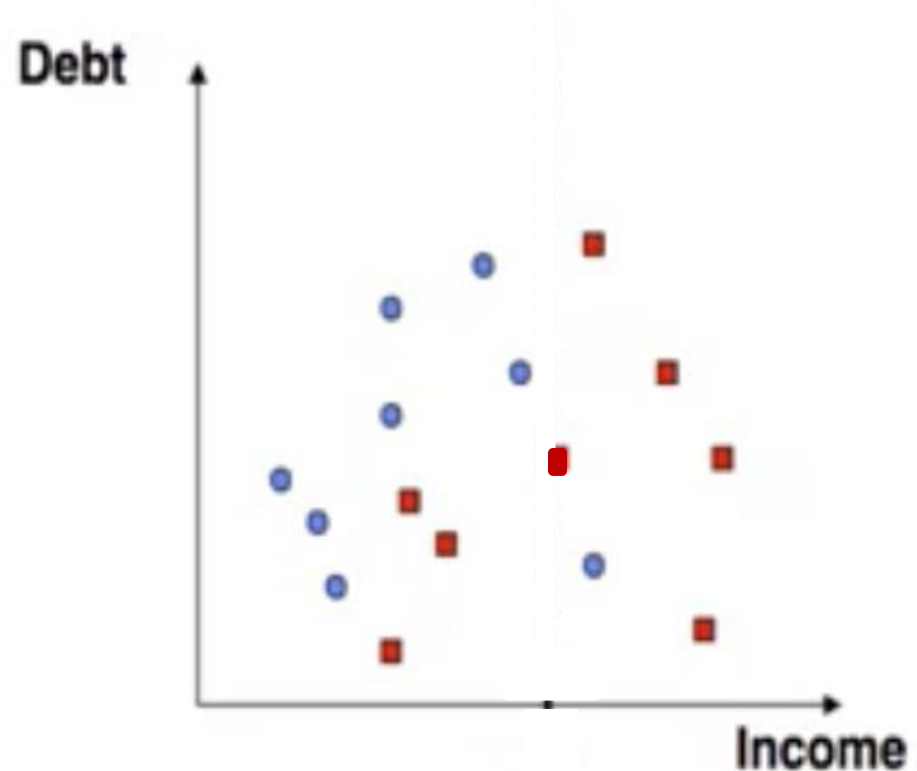


Decision Boundries (Univeriate)

- In the Decision Tree, each internal node represents a decision based on a feature (e.g., income > 100k), and each leaf node represents a class (e.g., Likely to repay)
 - The decision boundaries determined by the Decision Tree **divide the feature space into regions** corresponding to different classes

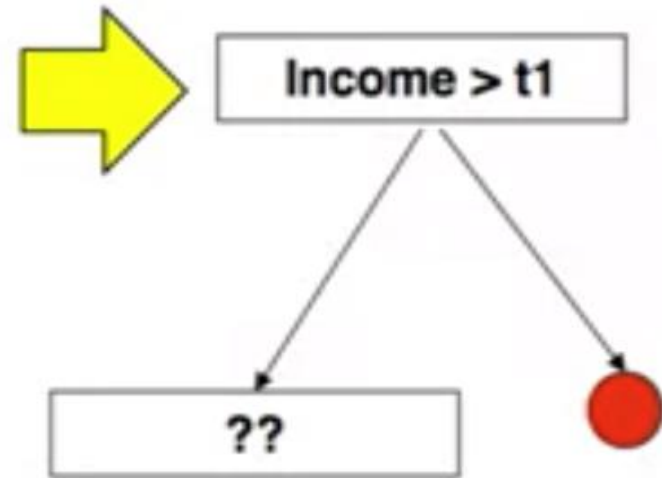
E.g., classify loan applicants to:

→ Not-/likely to repay,
based on income and
existing debt



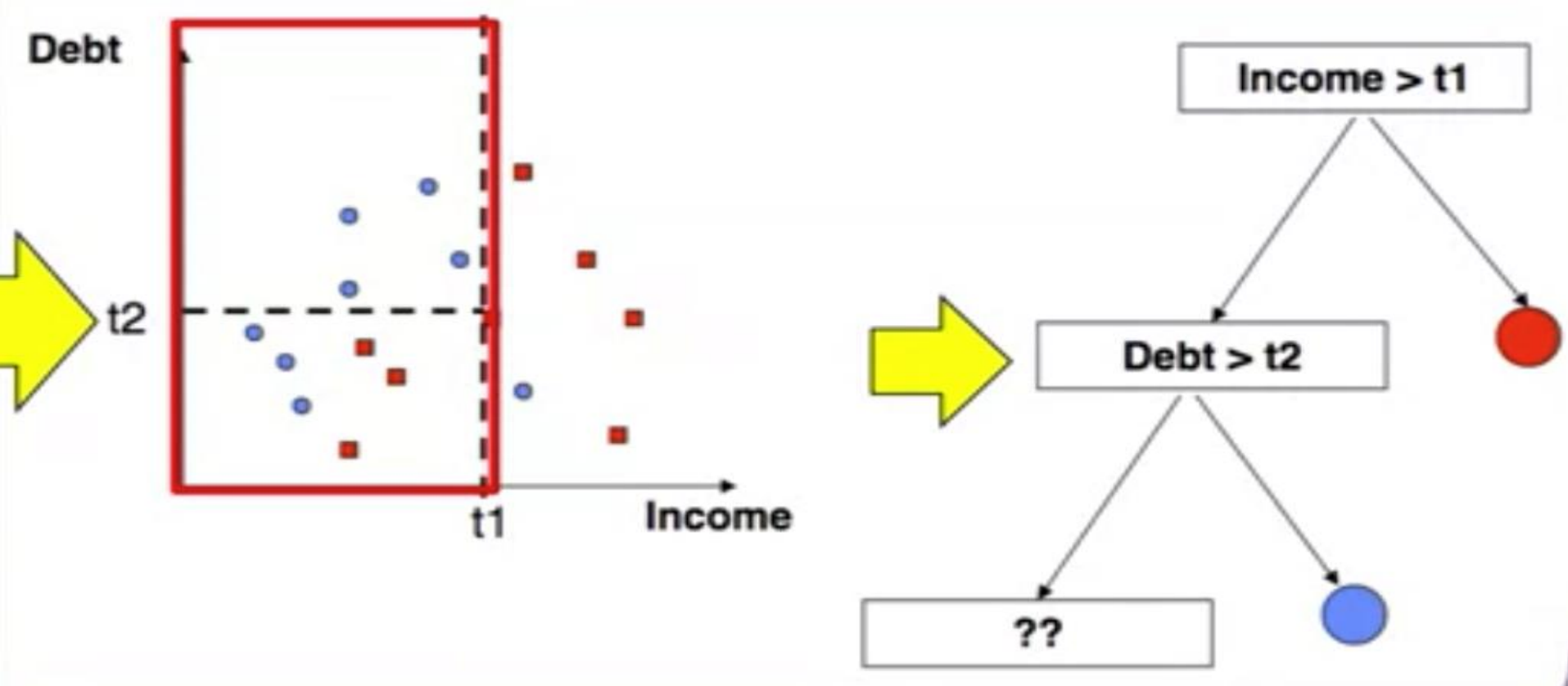
Decision Boundaries

- Split 1



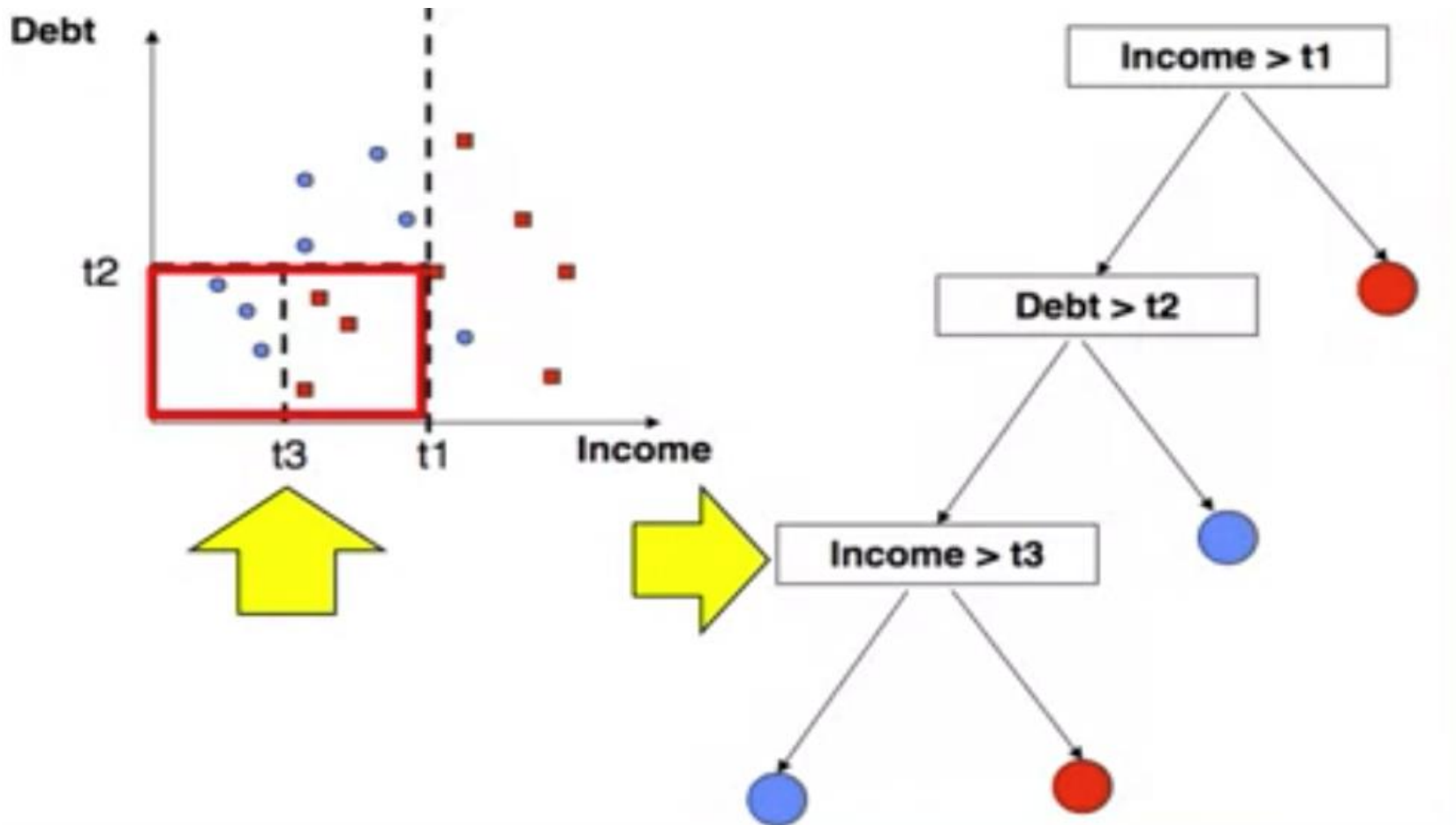
Decision Boundries

- Split 2



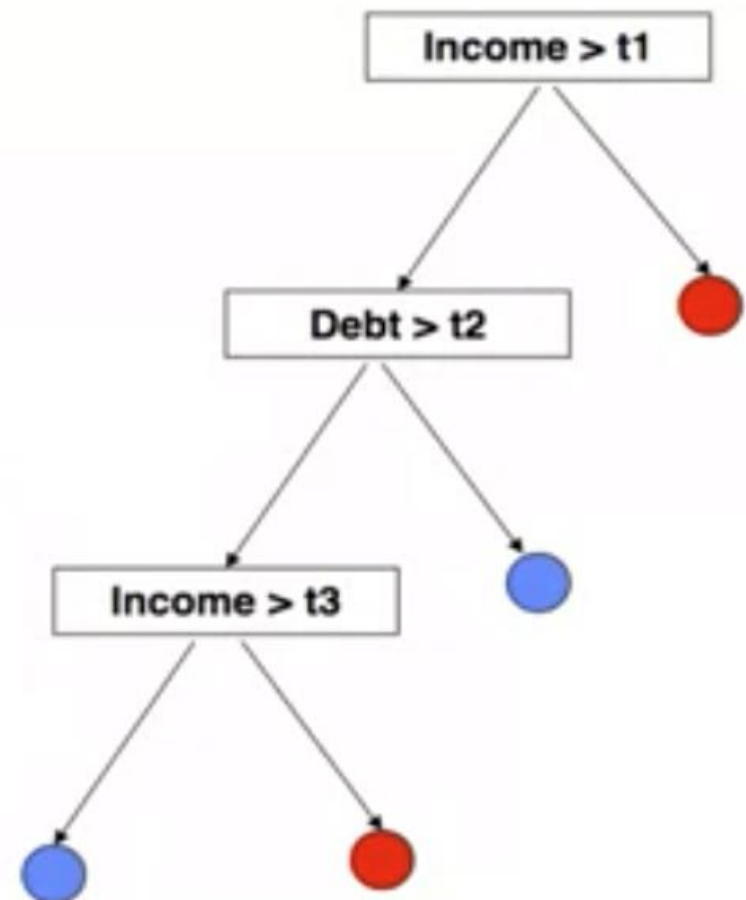
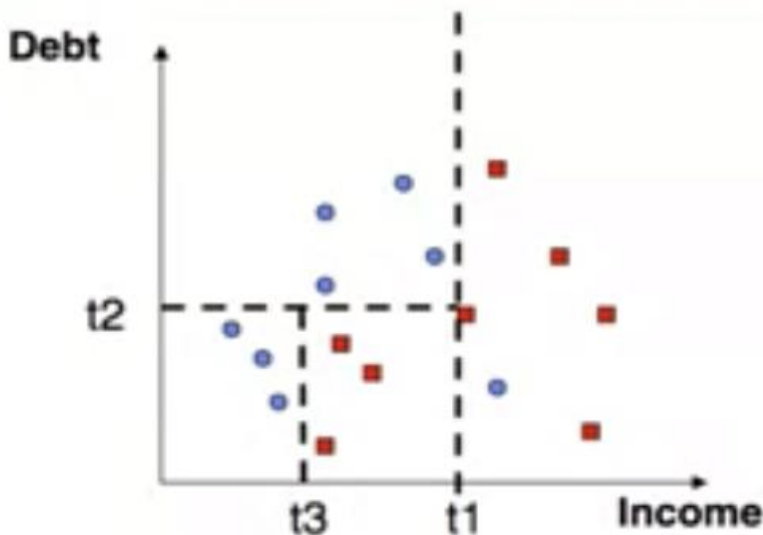
Decision Boundaries

- Split 3



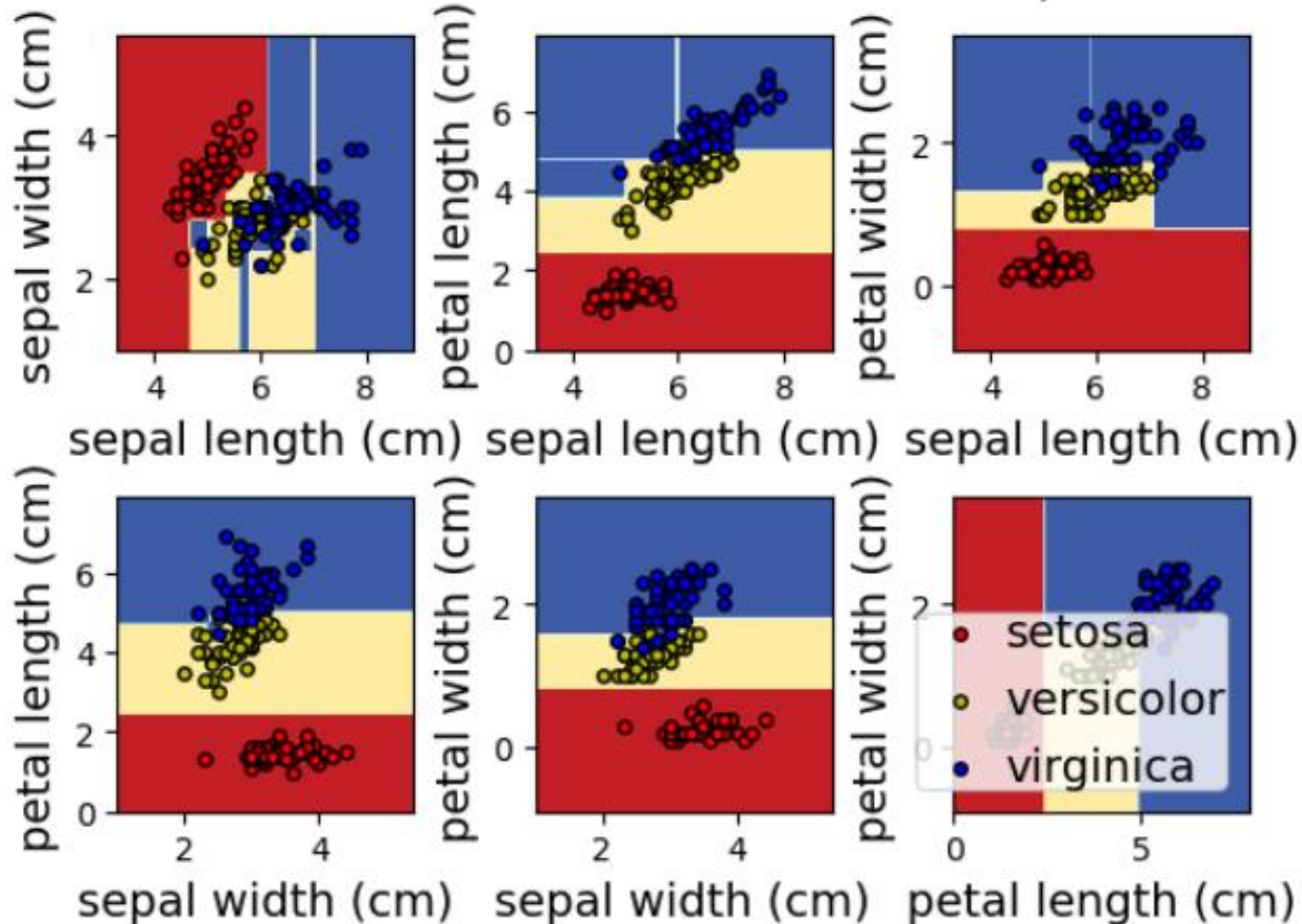
Decision Boundaries

- Resulting model
 - Boundaries are “**Rectilinear**” = Parallel to the axes



Decision Boundaries

Decision surface of decision trees trained on pairs of features



Decision Tree Induction Algorithms (Summary)

- Cons:
 - Must use discrete (or discretized) attributes
 - The greedy approach does not guarantee best solution
 - Since the process deals with one variable at a time, no way to capture interactions between variables
 - Rectilinear decision boundaries → limited expressiveness of the resulting model
 - May not perform well where there is structure in the data that is not well captured by horizontal or vertical splits
 - Non robust. Sensitive to small changes in the data
 - Trees generally do not have the same level of predictive accuracy as some of the other classification approaches
 - Decision trees suffer from a problem of errors propagating throughout a tree
 - Since decision trees work by a series of local decisions, what happens when one of these local decisions is wrong?
 - Every decision from that point on may be wrong
 - We may never return to the correct path of the tree