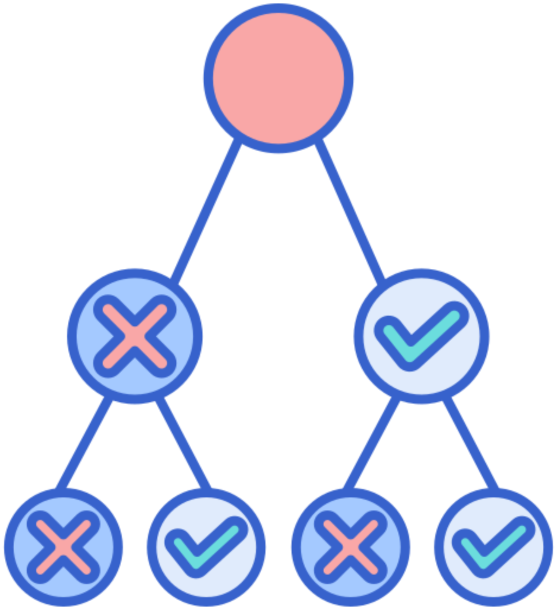
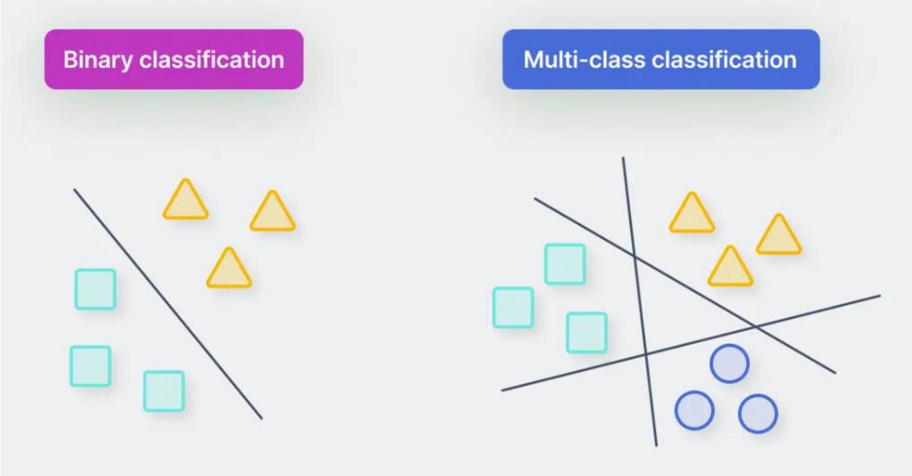
Classification using Decision Trees



Outline

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

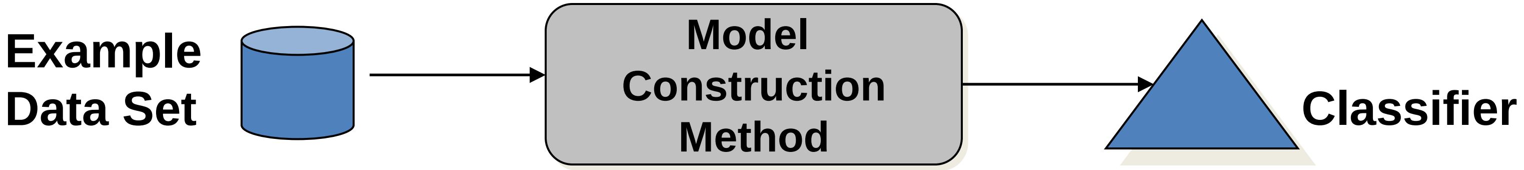
Classification



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**

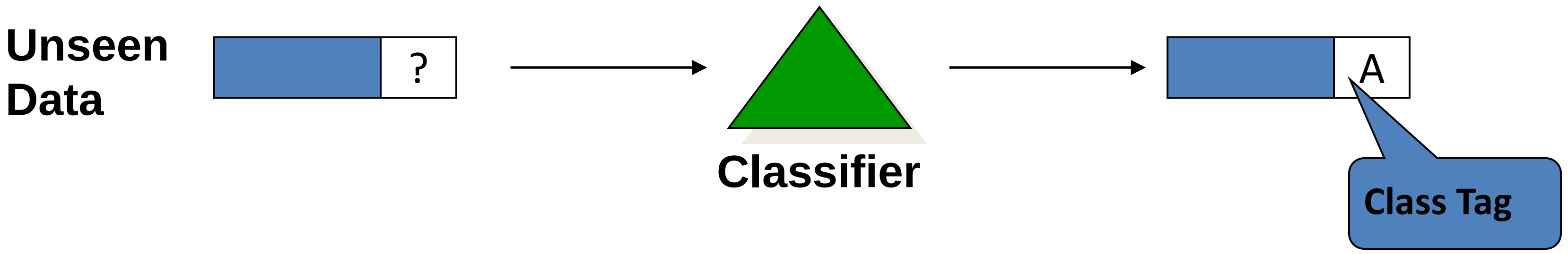


Example Data Set

Model Construction Method

Classifier

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



Unseen Data

?

Classifier

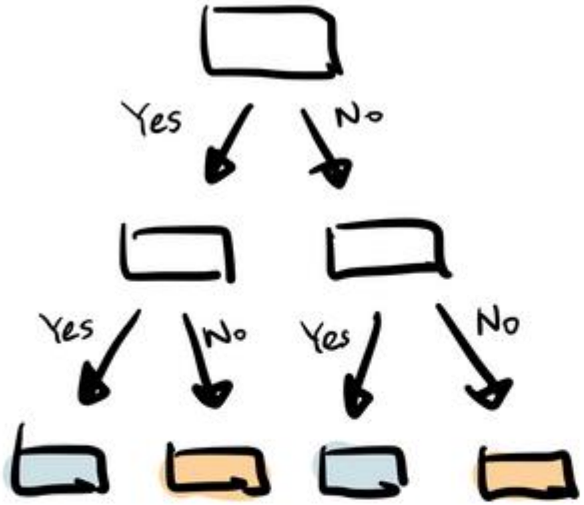
A

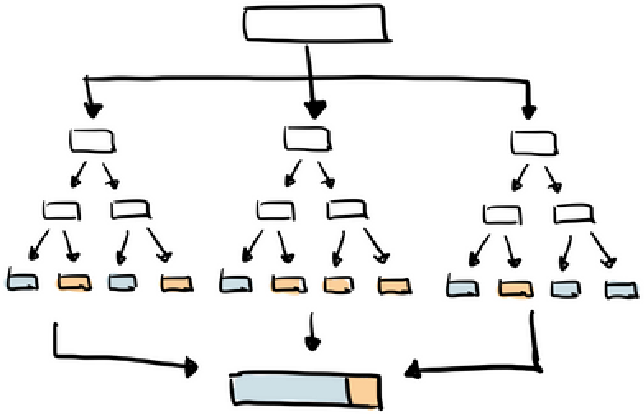
Class Tag

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

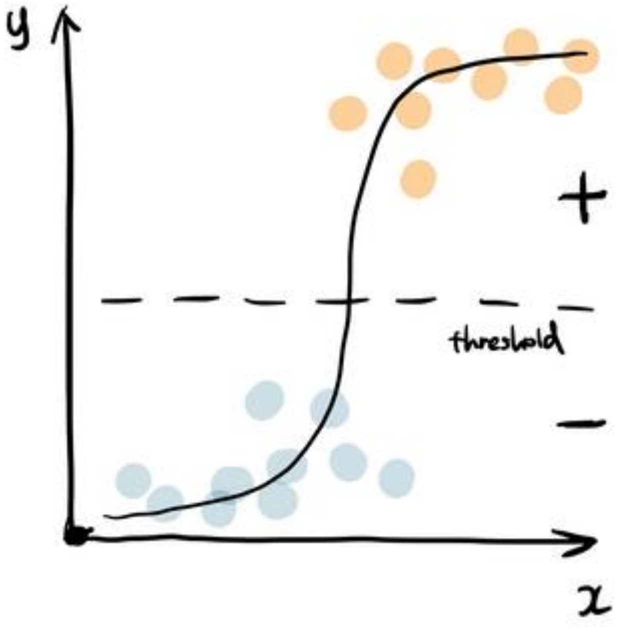
Decision Trees & Random Forest





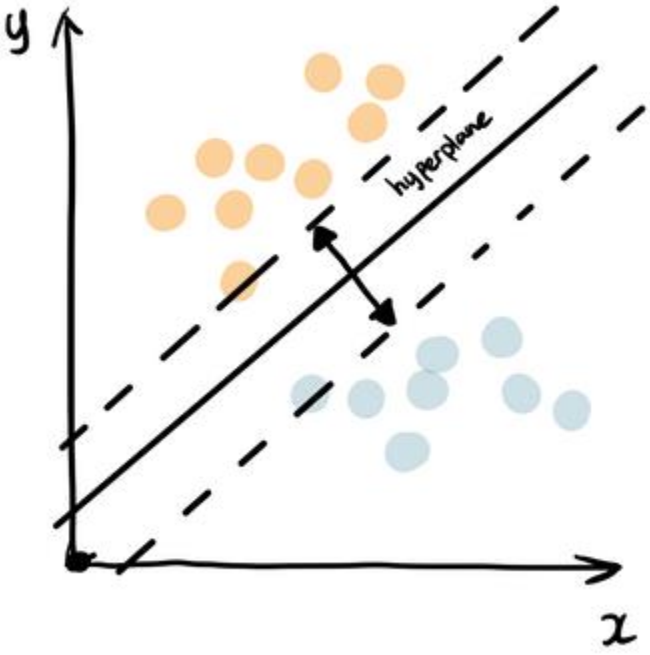
- **Decision Trees**:
  - A tree-like model where each internal node represents a "test" on an attribute
  - It splits the data into different branches based on the attribute values
  - Decision trees are interpretable and can handle both numerical and categorical data
- **Random Forest**:
  - A collection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



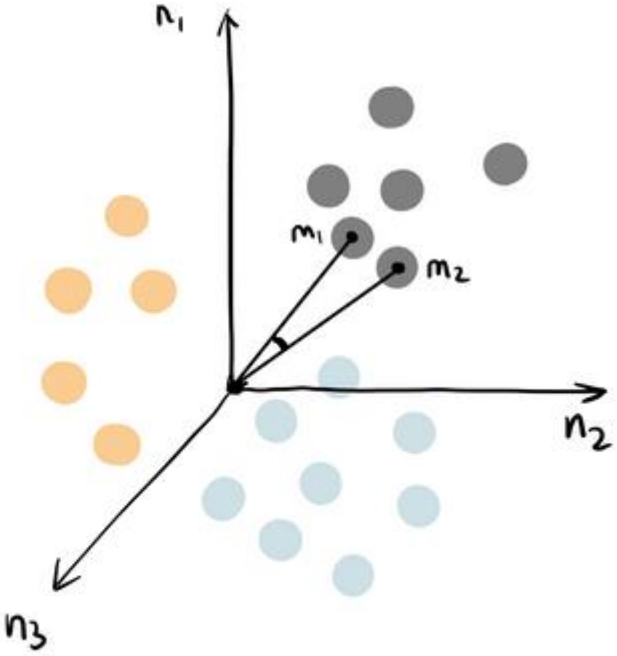
- A linear model used for **binary** classification problems
- It models the probability that a given input belongs to a certain class using the logistic function
- It's simple, interpretable, and efficient for linearly separable data

Support Vector Machine (SVM)



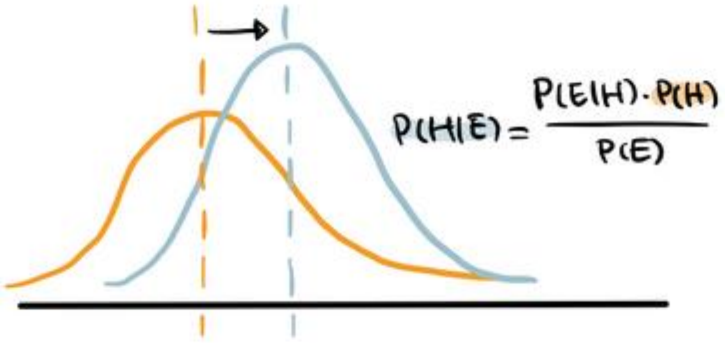
- A model that finds the optimal hyperplane separating different classes in the feature space
  - Classify the data based on the position in relation to the hyperplane between positive class and negative class
- SVM aims to maximize the margin between classes, thus enhancing generalization
- It can handle 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

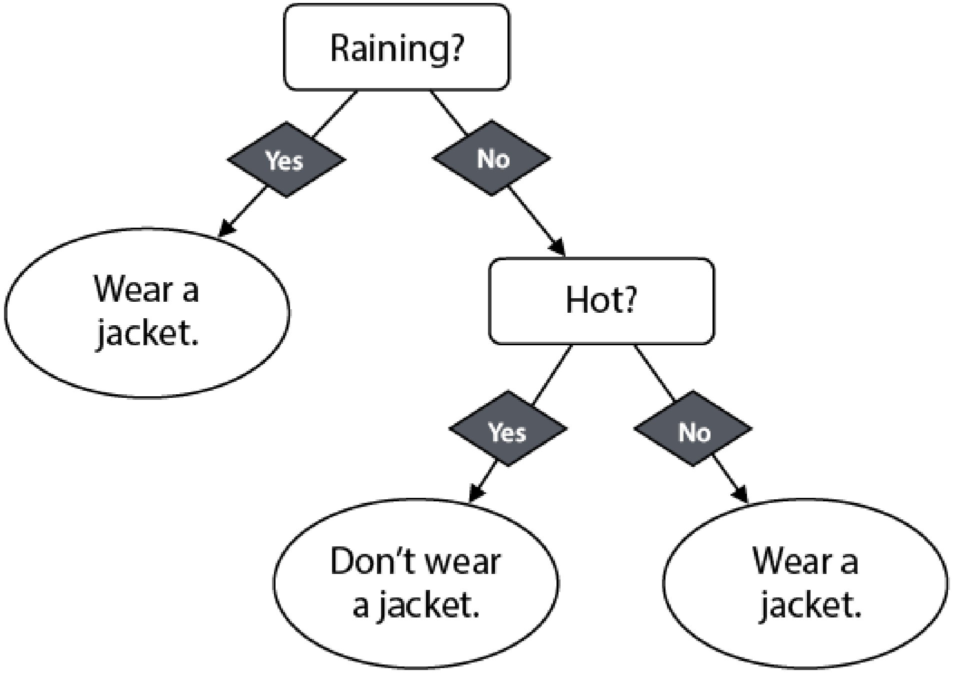
ML Metrics: Influential Factors for a Good Model

- 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 ware jacket today?

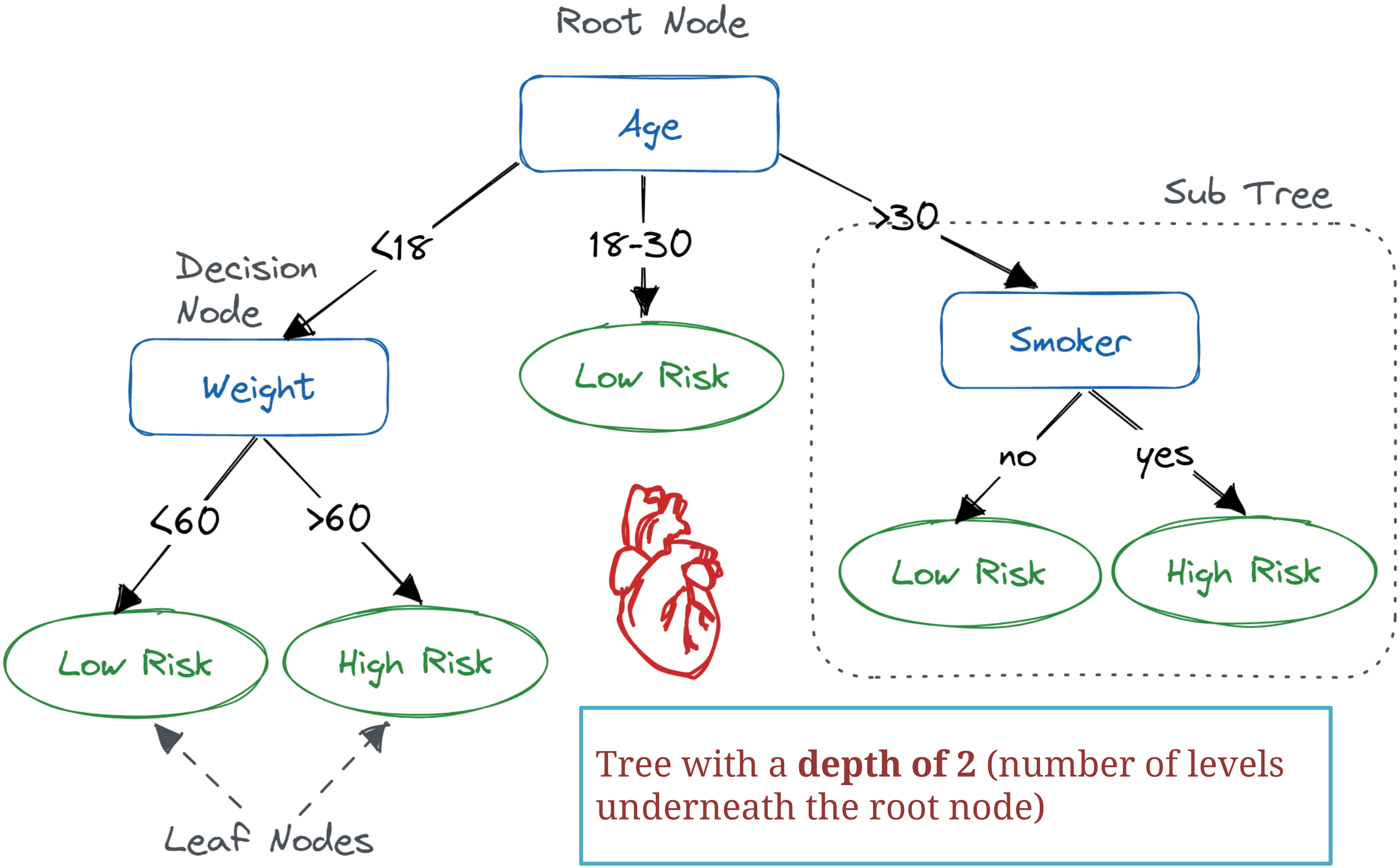
- If it’s raining, then wear a jacket
- If it’s not, then we check the temperature:
  - If it is hot, then don’t wear a jacket
  - But if it is cold, then wear a jacket
- The decision tree depicts the decision process, where the **decisions are made by traversing the tree from top to bottom**



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

Image source

Decision Tree – Risk of heart attack



Tree with a **depth of 2** (number of levels underneath the root node)

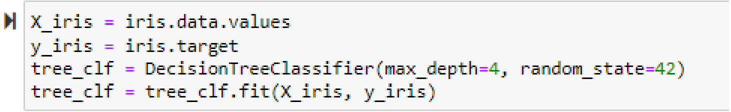
Class Label

Image source

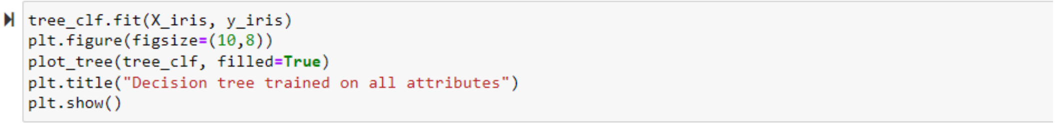
Decision Tree

- **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

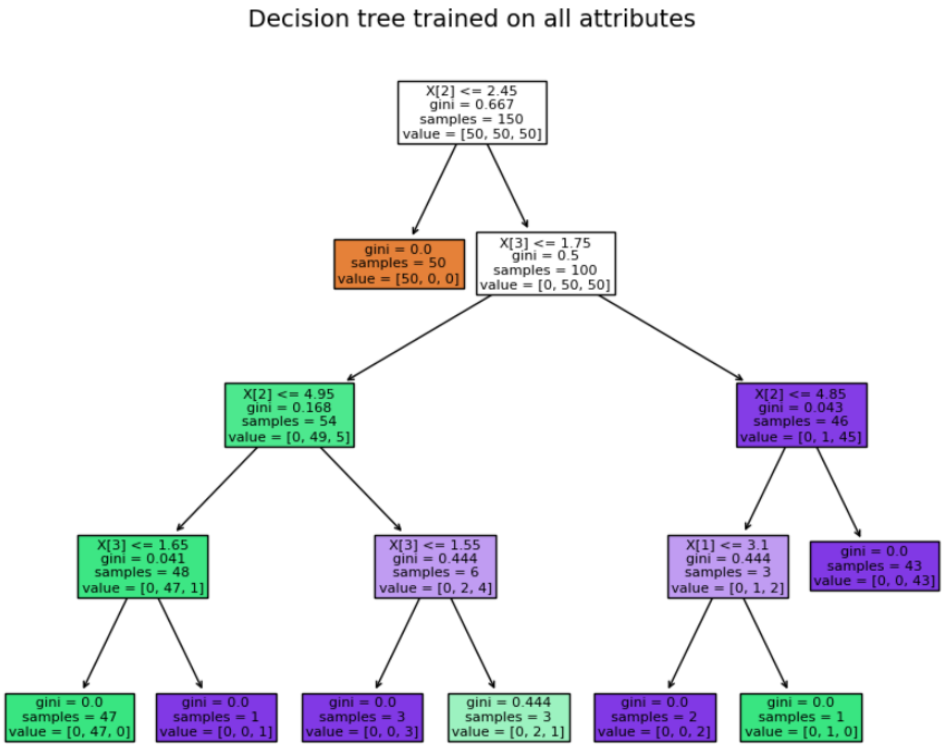
Decision Tree for the Iris dataset: using all the four attributes

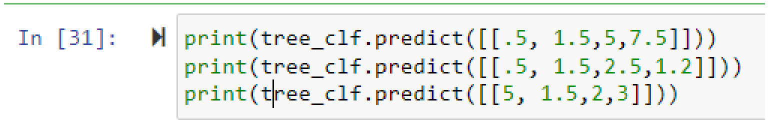


```
X_iris = iris.data.values
y_iris = iris.target
tree_clf = DecisionTreeClassifier(max_depth=4, random_state=42)
tree_clf = tree_clf.fit(X_iris, y_iris)
```

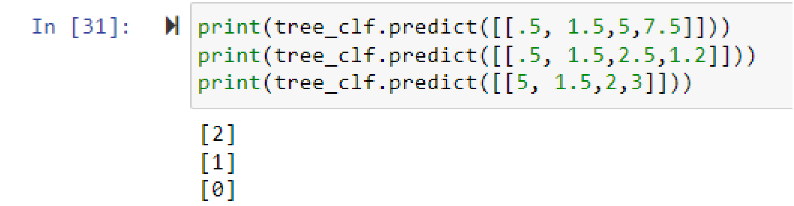


```
tree_clf.fit(X_iris, y_iris)
plt.figure(figsize=(10,8))
plot_tree(tree_clf, filled=True)
plt.title("Decision tree trained on all attributes")
plt.show()
```





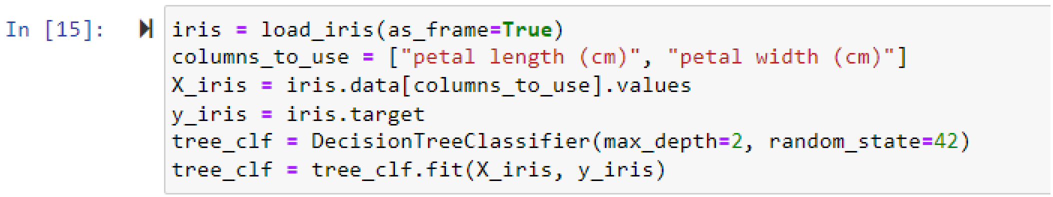
```
In [31]:  print(tree_clf.predict([[.5, 1.5,5,7.5]]))
           print(tree_clf.predict([[.5, 1.5,2.5,1.2]]))
           print(tree_clf.predict([[5, 1.5,2,3]]))
```



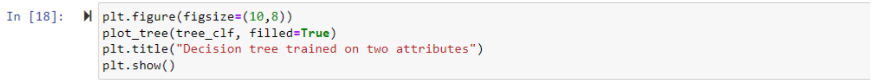
```
In [31]:  print(tree_clf.predict([[.5, 1.5,5,7.5]]))
           print(tree_clf.predict([[.5, 1.5,2.5,1.2]]))
           print(tree_clf.predict([[5, 1.5,2,3]]))

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

05.dt**1.dt_example.ipynb**

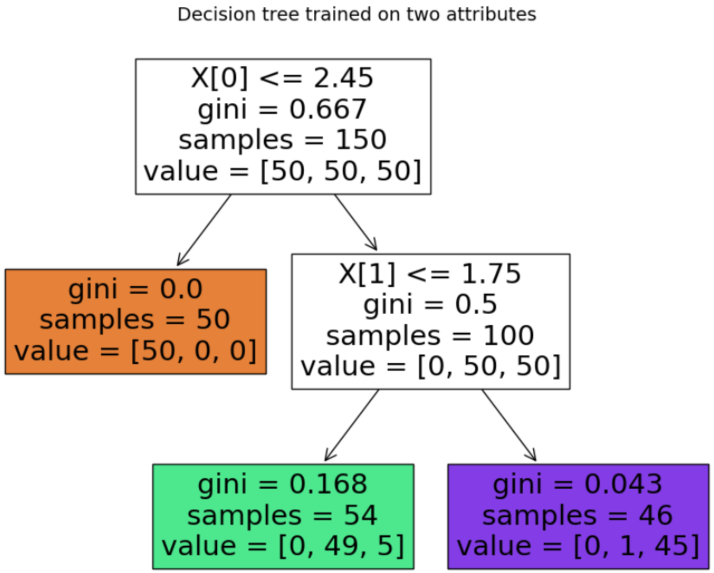
Decision Tree for the Iris dataset: using two attributes

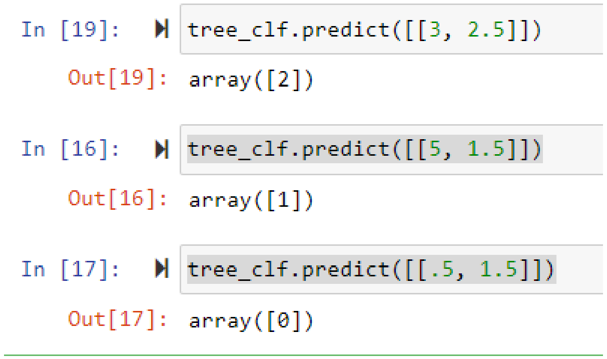


```
In [15]:  iris = load_iris(as_frame=True)
          columns_to_use = ["petal length (cm)", "petal width (cm)"]
          X_iris = iris.data[columns_to_use].values
          y_iris = iris.target
          tree_clf = DecisionTreeClassifier(max_depth=2, random_state=42)
          tree_clf = tree_clf.fit(X_iris,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

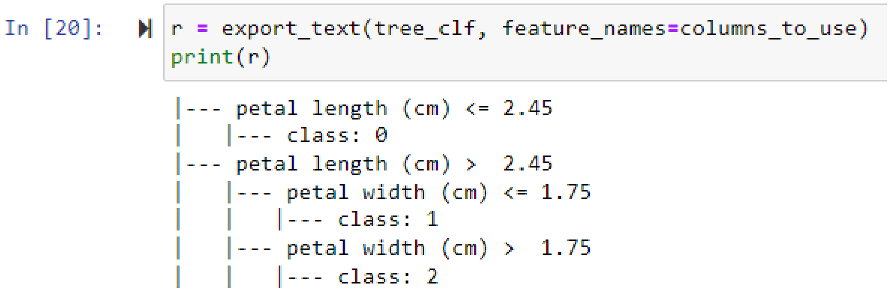


```
In [18]:  plt.figure(figsize=(10,8))
          plot_tree(tree_clf, filled=True)
          plt.title("Decision tree trained on two attributes")
          plt.show()
```





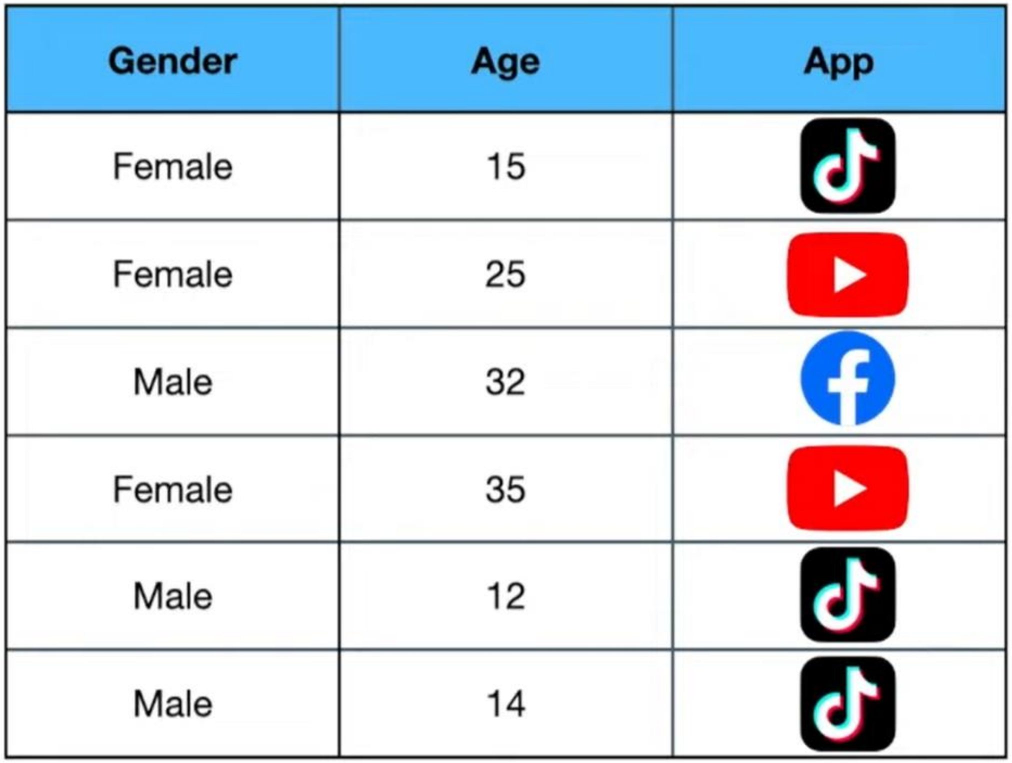
```
In [19]:  tree_clf.predict([[3, 2.5]])

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
```

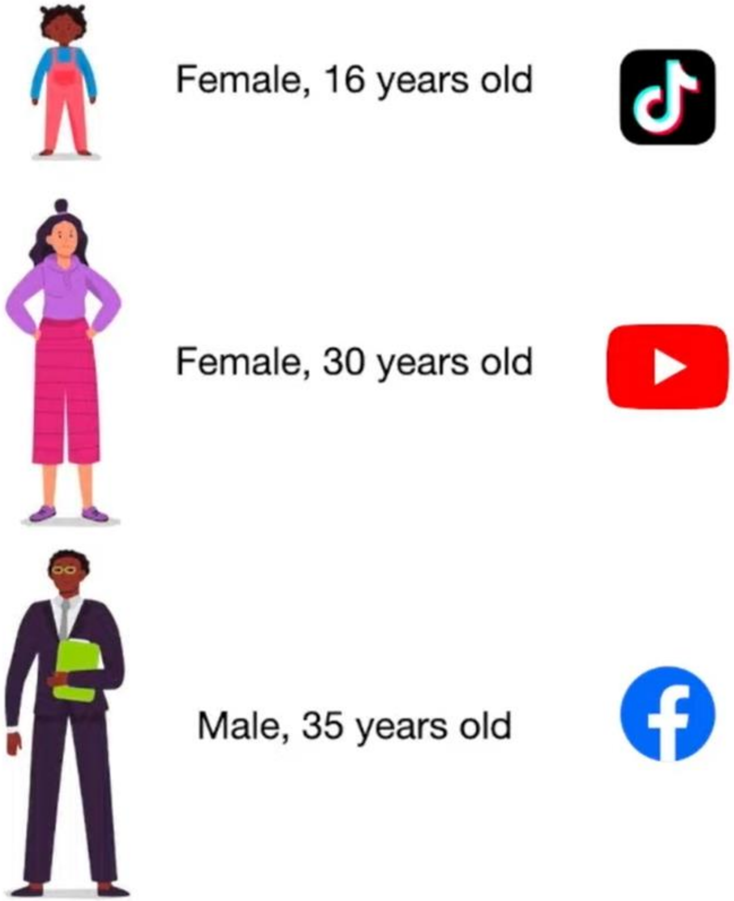
05.dt\5.decision_boundry_example.ipynb

App Recommendation System using a DT



Gender	Age	App
Female	15	♪
Female	25	▶
Male	32	f
Female	35	▶
Male	12	♪
Male	14	♪

What to recommend for?



Female, 16 years old ♪

Female, 30 years old ▶

Male, 35 years old f

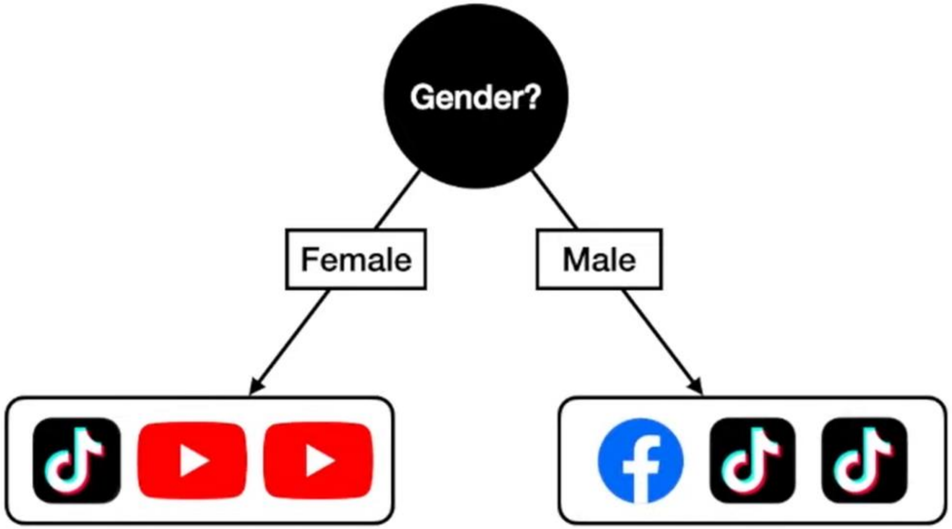
Adopted from source

Which feature is more important?

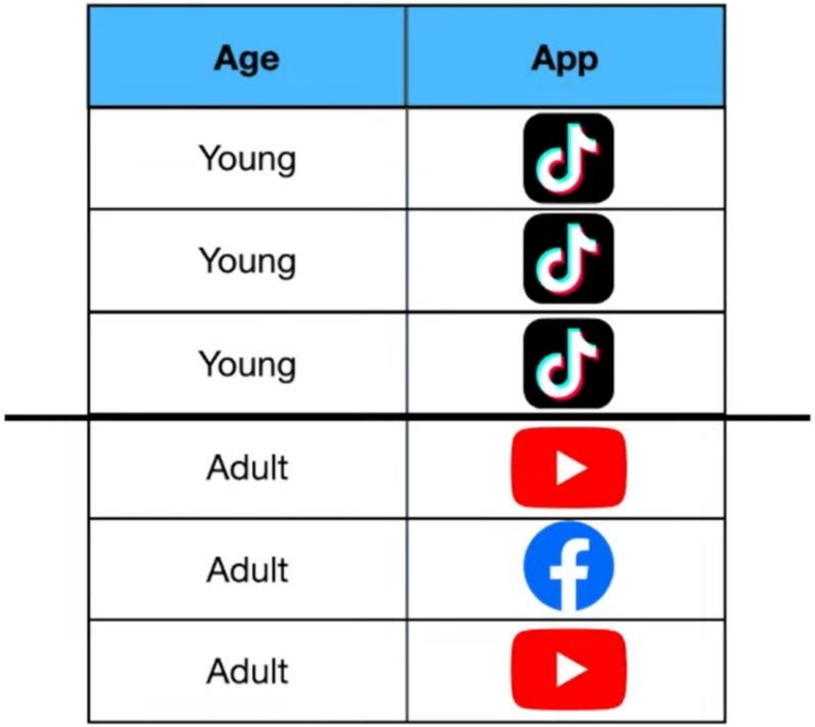
- Which one of the two features (gender or age) **is more important** in determining the app to recommend?
  - This is the most important step in building a decision tree!
- Let’s **split** the data by gender then by age to compare them

Splitting by Gender => Gender decision stump

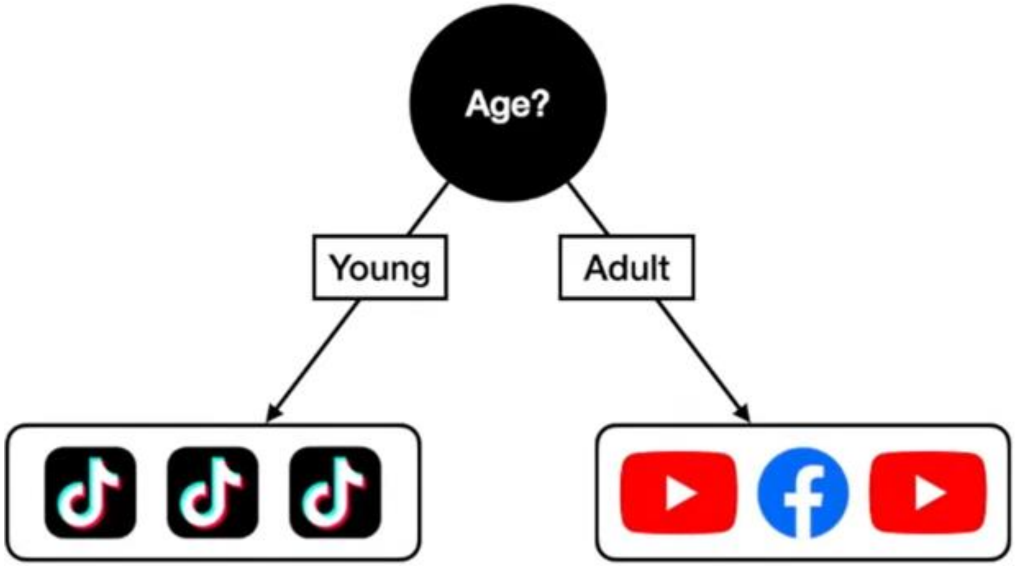
Gender	App
Female	♪
Female	▶
Female	▶
Male	f
Male	♪
Male	♪



Splitting by Age => Age decision stump



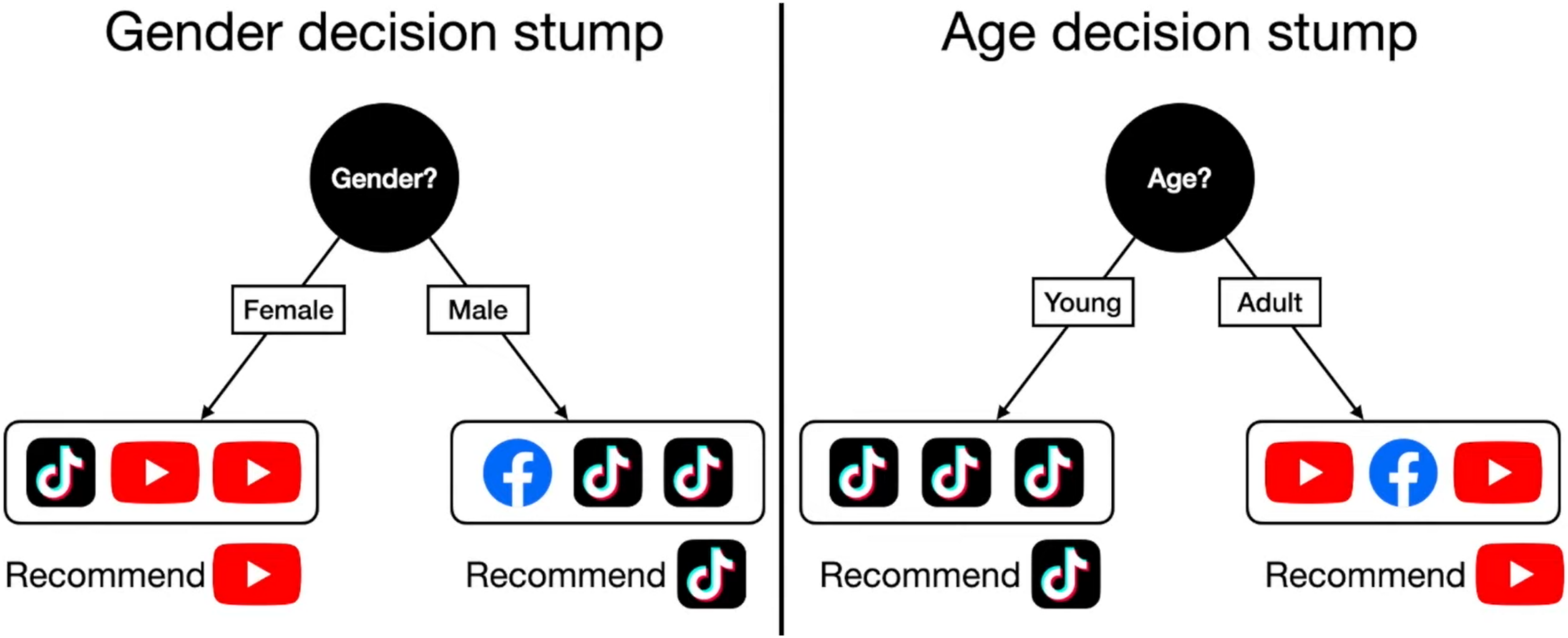
Age	App
Young	♪
Young	♪
Young	♪
Adult	▶
Adult	f
Adult	▶



Young = Age <= 18

Adult : Age > 18

Which one is better?

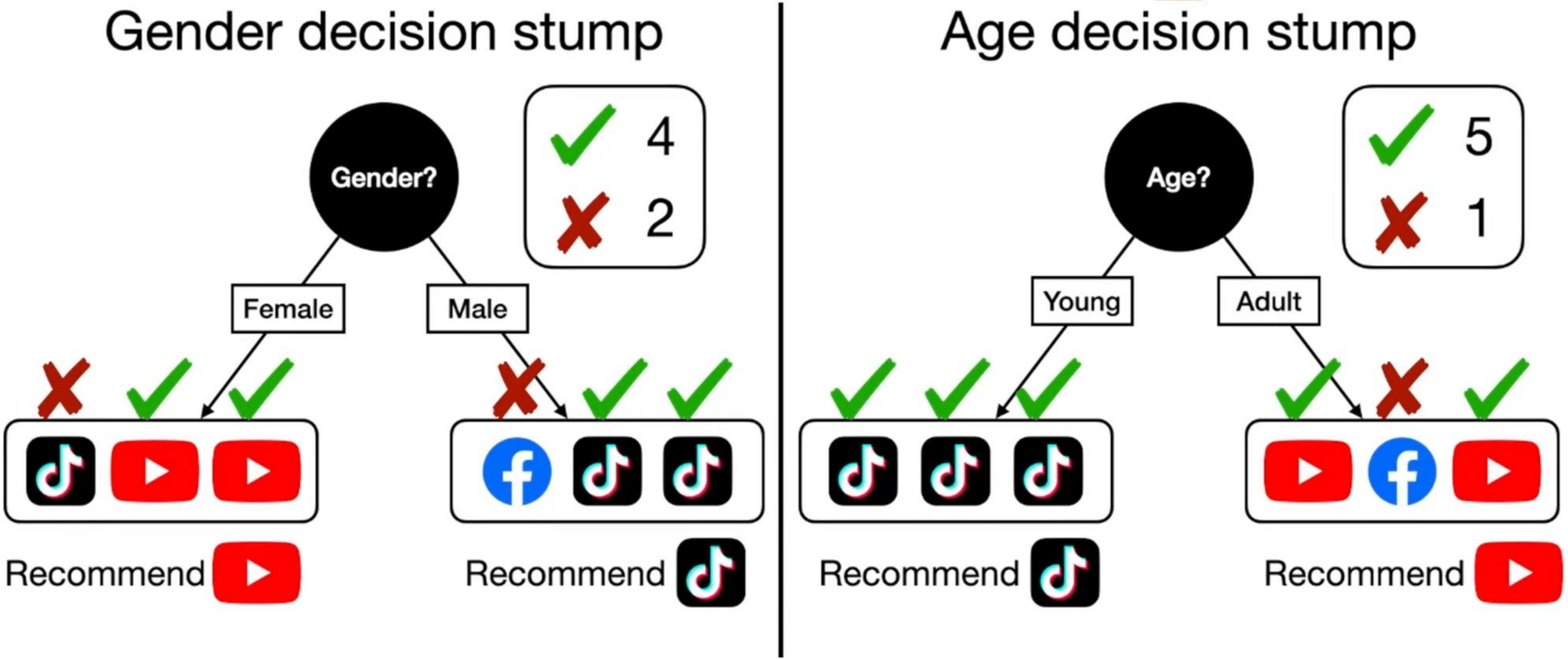


Each feature splits the data into two smaller datasets

Which one is better?

Accuracy: 66.7% & Error Rate: 33.3%

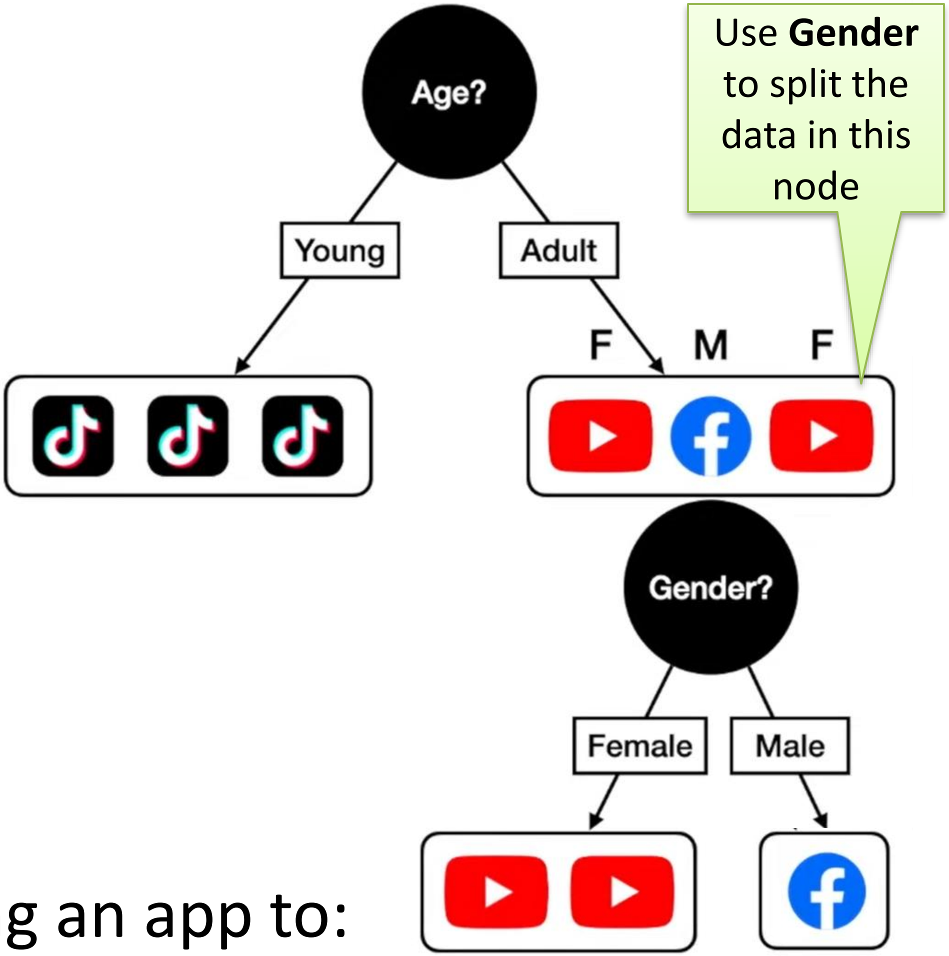
Accuracy: 83.3% 🏆 & Error Rate: 16.7%



- Based on **accuracy**, the **Age** feature is the winner => It is **determinant** in the prediction 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	f
Female	Adult	▶
Male	Young	♪
Male	Young	♪



Use **Gender** to split the data in this node

- Let’s test it for recommending an app to:
  - Female, 16 years old ♪
  - Female, 30 years old ▶
  - Male, 35 years old f

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, 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

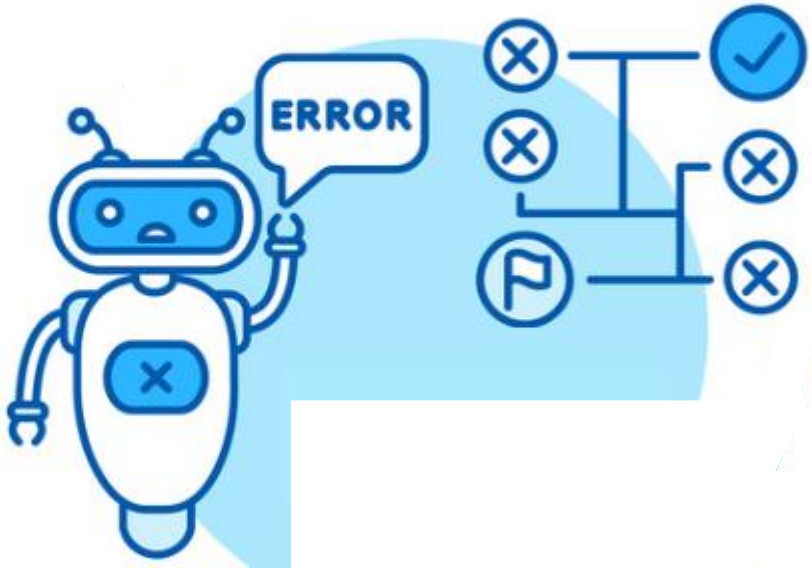
- Studies show that there are only marginal differences among the attribute selection measures w.r.t. model accuracy

Attribute Selection Measure (ASM)

Classification Error Rate

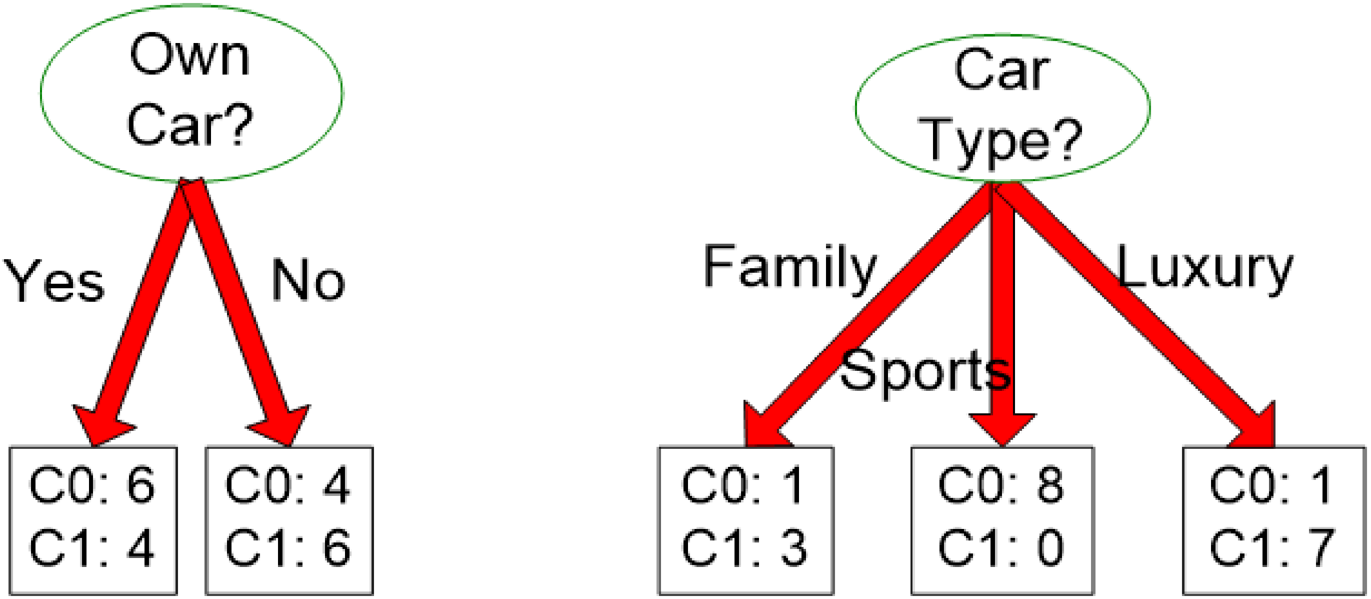
$$Error(t) = 1 - \max_{i} P(i|t)$$

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

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

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

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

- $\mathit{Error(t)}$ 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 ($\mathbf{1 - \frac{1}{n_c}}$) when the node has equally distributed $\mathbf{n_c}$ classes
- The DT algorithms selects the feature that minimizes the Classification Error Rate

Examples for Computing Classification Error Rate

$$\boxed{Error(t) = 1 - \max_i P(i \mid t)}$$

Weighted Average

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

Female (Left)	T	1
	Y	2

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

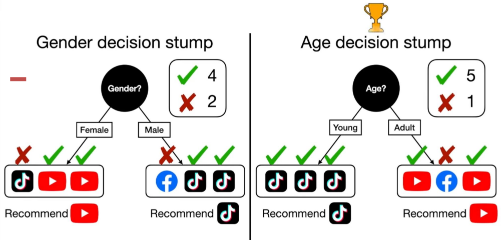
$\text{Error(L)} = 1 - \max\left(\frac{1}{3}, \frac{2}{3}\right) = 1 - \frac{2}{3} = \frac{1}{3}$

Male (Right)	F	1
	T	2

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

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

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



Young (Left)	T	3

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

$\text{Error(L)} = 1 - \max(1) = 1 - 1 = 0$

Adult (Right)	Y	2
	F	1

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

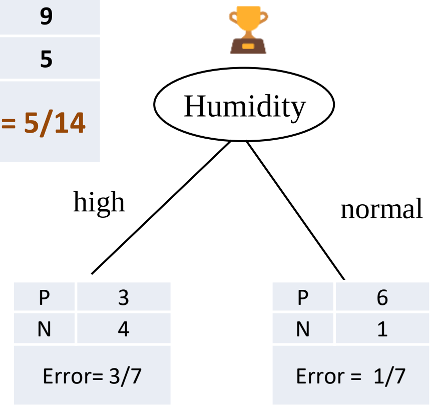
$\text{Error(R)} = 1 - \max\left(\frac{2}{3}, \frac{1}{3}\right) = 1 - \frac{2}{3} = \frac{1}{3}$

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

🏆

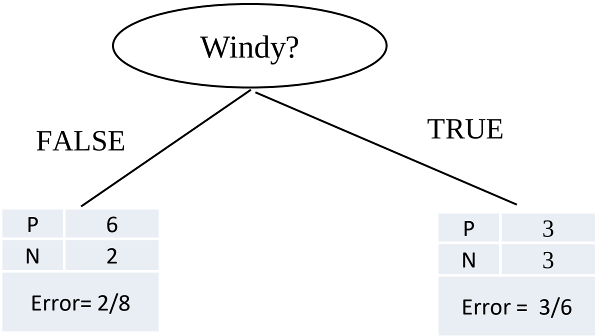
Outlook	Temperature	Humidity	Windy	Class
sunny	Hot	high	FALSE	N
sunny	Hot	High	TRUE	N
Overcast	Hot	high	FALSE	P
Rain	Mild	high	FALSE	P
Rain	Cool	Normal	FALSE	P
Rain	Cool	Normal	TRUE	N
Overcast	Cool	Normal	TRUE	P
Sunny	Mild	High	FALSE	N
Sunny	Cool	normal	FALSE	P
Rain	Mild	Normal	FALSE	P
Sunny	Mild	Normal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



P	3
N	4
Error= 3/7	

P	6
N	1
Error = 1/7	



P	6
N	2
Error= 2/8	

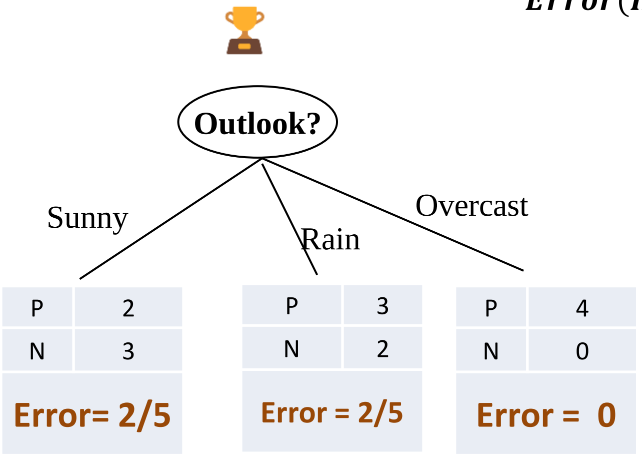
P	3
N	3
Error = 3/6	

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 Humidity Split

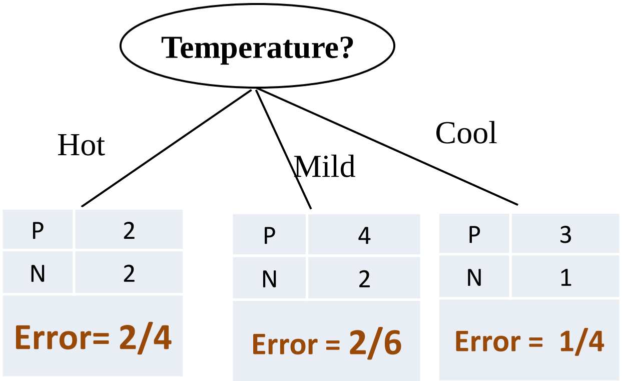
$$Error(Humidity) = \frac{7}{14} \times \frac{3}{7} + \frac{7}{14} \times \frac{1}{7} = \frac{3}{14} + \frac{1}{14} = \frac{4}{14}$$



P	2
N	3
Error= 2/5	

P	3
N	2
Error = 2/5	

P	4
N	0
Error = 0	



P	2
N	2
Error= 2/4	

P	4
N	2
Error = 2/6	

P	3
N	1
Error = 1/4	

Weighted Average Error for the Outlook Split

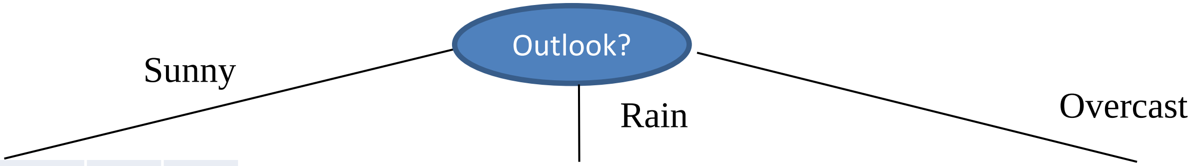
$$Error(Outlook) = \frac{5}{14} \times \frac{2}{5} + \frac{5}{14} \times \frac{2}{5} + \frac{4}{14} \times 0 = \frac{2}{14} + \frac{2}{14} = \frac{4}{14}$$

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



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
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

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

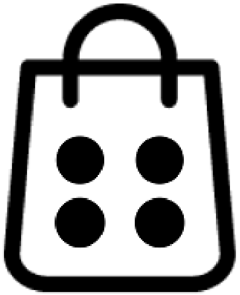
Next: Repeat selecting the best feature to split on

Attribute Selection Measure (ASM)

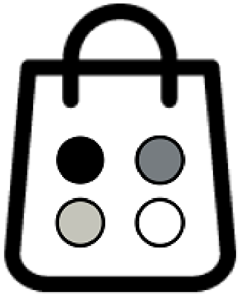
Gini Impurity Index

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

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



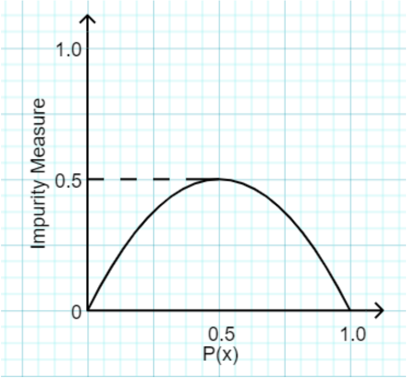
Low Gini impurity index



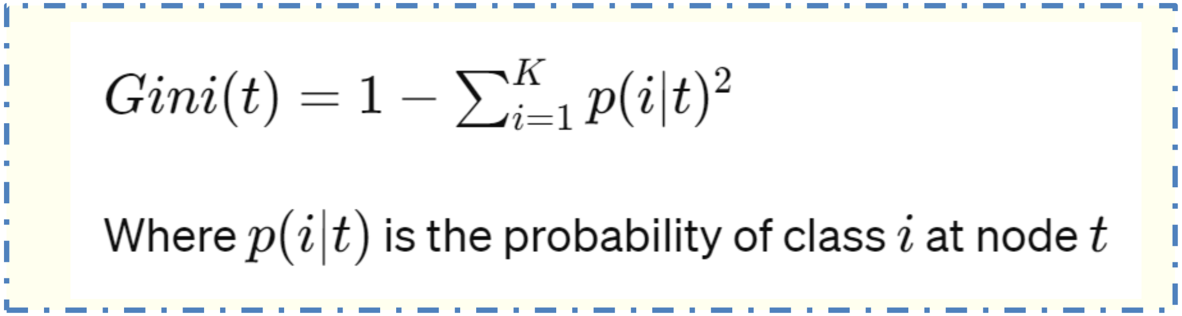
High Gini impurity index

Gini Index

- The Gini Index is a **measure of impurity** or randomness in a dataset
  - The Gini Index ranges from 0 to 0.5, where:
    - **0**: indicates perfect purity, meaning all node elements belong to a single class
    - **0.5**: indicates maximally impure node, having elements evenly distributed across all classes



- The formula to calculate the Gini Index for a node **t** with **K** classes is:

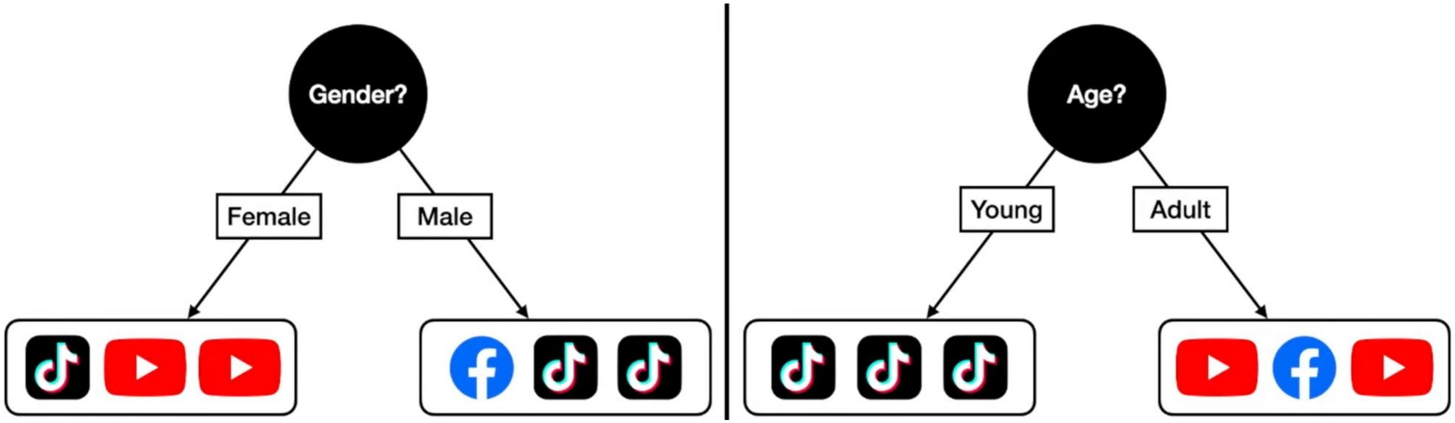


$$Gini(t) = 1 - \sum_{i=1}^{K} p(i|t)^2$$

Where $p(i|t)$ is the probability of class i 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 x 0.44) + (3 x 0.44)) / 6 = 0.44**

Measures the **impurity** of the split

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

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

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

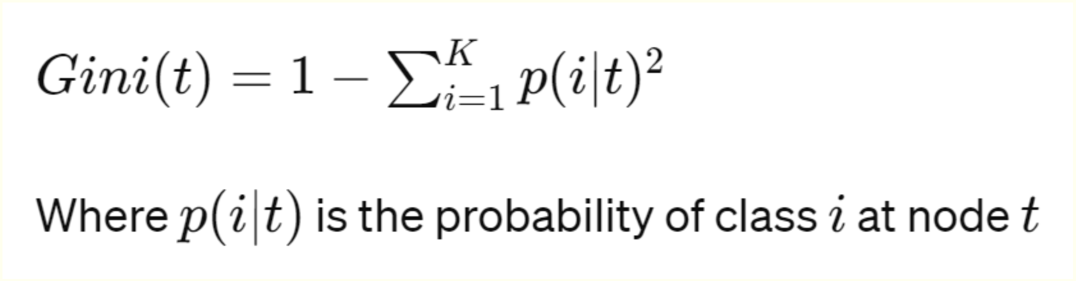
$$\textbf{Gini} = 1 - (\mathbf{P}(\mathrm{F})^2 - \mathbf{P}(\mathrm{T})^2) = 1 - \left(\left(\frac{1}{3}\right)^2 + \left(\frac{2}{3}\right)^2\right) = \mathbf{0.44}$$

Classifier 2 (by age): **Avg Gini = ((3 x 0) + (3 x 0.44)) / 6 = 0.22** 🏆

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

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

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   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   P(C2) = 5/6

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

C1	**2**
C2	**4**

P(C1) = 2/6   P(C2) = 4/6

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

Splitting Based on Gini

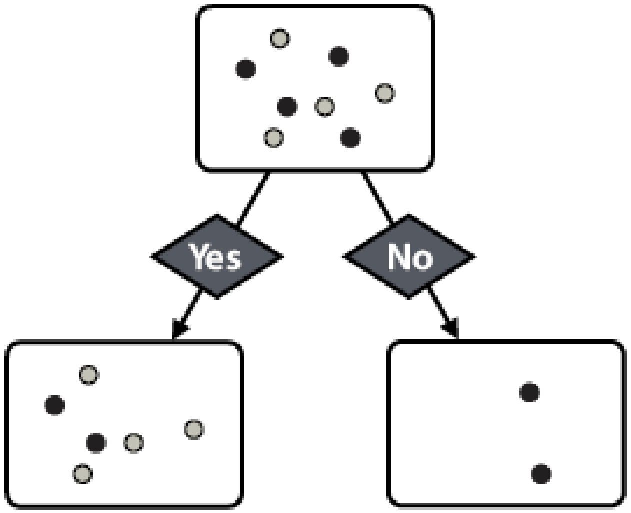
- When a node p is split into k partitions (children), the **quality of split** is computed as

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

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



Gini = 0.444

Gini = 0

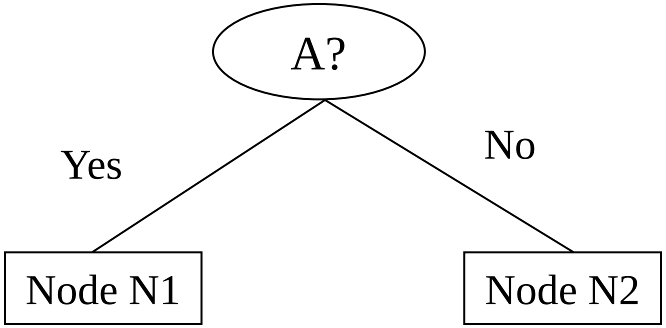
$$\text{Weighted average Gini} = 0.444 \cdot \frac{6}{8} + 0 \cdot \frac{2}{8} = 0.333$$

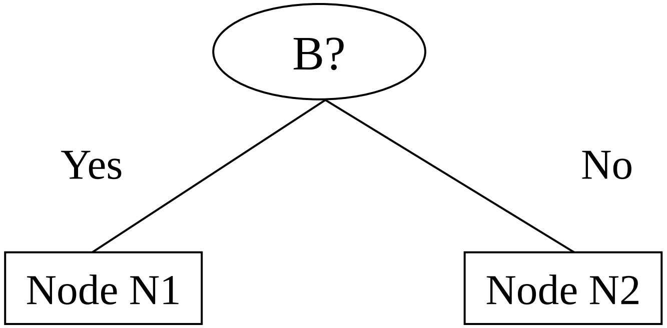
- A split of a Node of size 8 into 2 partitions of sizes 6 and 2
- We calculate the Gini index of the Node as the **weighted average** of the Gini of partitions:
  - We weight the index of the left partition by 6/8 and that of the right partition by 2/8

Computing Gini Index of a Split

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

	Parent
C1	**6**
C2	**6**
Gini = 0.500	





	N1
C1	6
C2	1

	N2
C1	0
C2	5

	N1
C1	4
C2	2

	N2
C1	2
C2	4

Gini(N1) = $1 – (6/7)^2 – (1/7)^2$ = 0.245

Gini(N2) = $1 – (0/5)^2 – (5/5)^2$ = 0

Gini(A) = 7/12 * 0.245 + 5/12 * 0 = **0.1429**

Gini(N1) = $1 – (4/6)^2 – (2/6)^2$ = 0.2222

Gini(N2) = $1 – (2/6)^2 – (4/6)^2$ = 0.2222

Gini(B) = 6/12 * 0.2222 + 6/12 * 0.2222 = **0.2222**

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

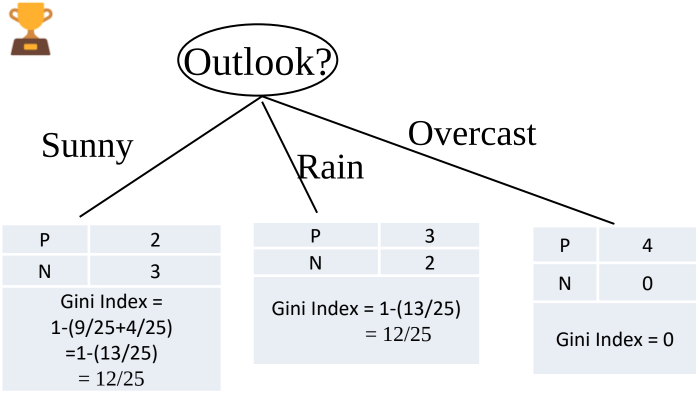
Computing Gini Index – Example (1/2)

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

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

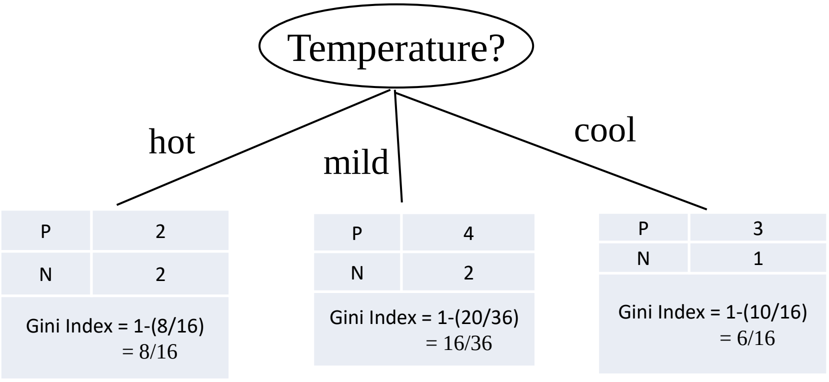
	Parent
P	9
N	5
Gini Index = 1 –(25/196 + 81/196) = 90/196 = 0.4592	



P	2
N	3
Gini Index = 1-(9/25+4/25) =1-(13/25) = 12/25	

P	3
N	2
Gini Index = 1-(13/25) = 12/25	

P	4
N	0
Gini Index = 0	



P	2
N	2
Gini Index = 1-(8/16) = 8/16	

P	4
N	2
Gini Index = 1-(20/36) = 16/36	

P	3
N	1
Gini Index = 1-(10/16) = 6/16	

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

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

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

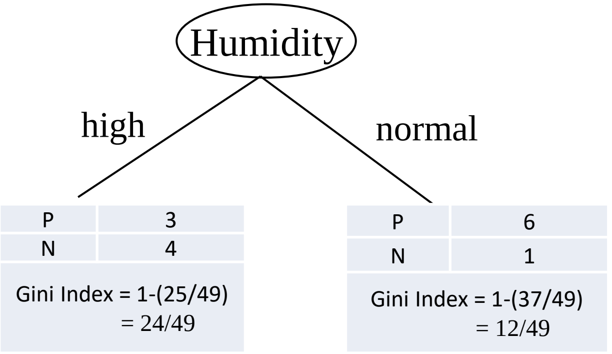
Computing Gini Index – Example (2/2)

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

Where $p(i|t)$ is the 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	

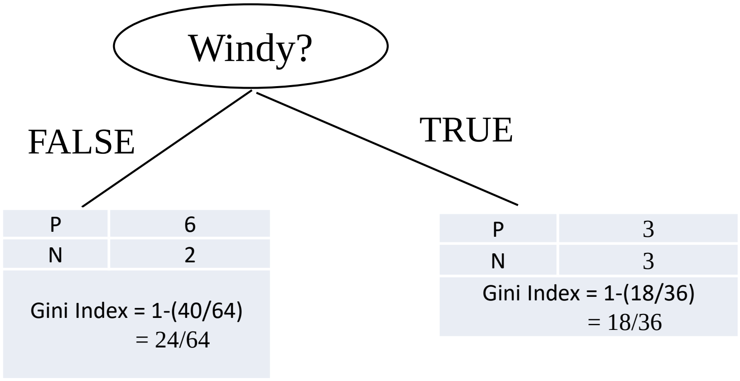


P	3
N	4
Gini Index = 1-(25/49) = 24/49	

P	6
N	1
Gini Index = 1-(37/49) = 12/49	

Weighted Average Gini Index for the Humidity Split

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



P	6
N	2
Gini Index = 1-(40/64) = 24/64	

P	3
N	3
Gini Index = 1-(18/36) = 18/36	

Weighted Average Gini Index for the Windy Split

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

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**		

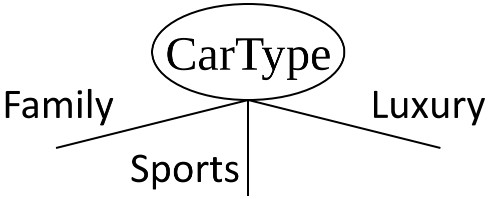
Two-way split
(find best partition of values)

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

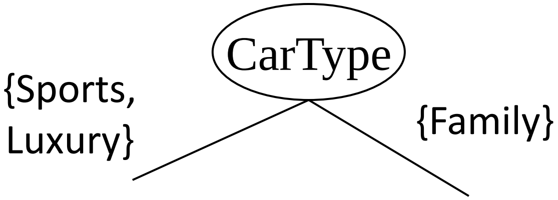
	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	**0.419**	

Splitting Based on Nominal Attributes

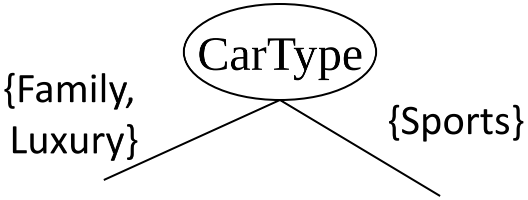
- Multi-way split: Use as many partitions as distinct values


CarType
Family
Sports
Luxury

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

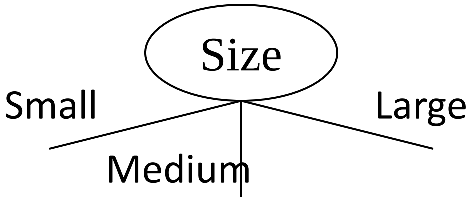

CarType
{Sports, Luxury}
{Family}

OR

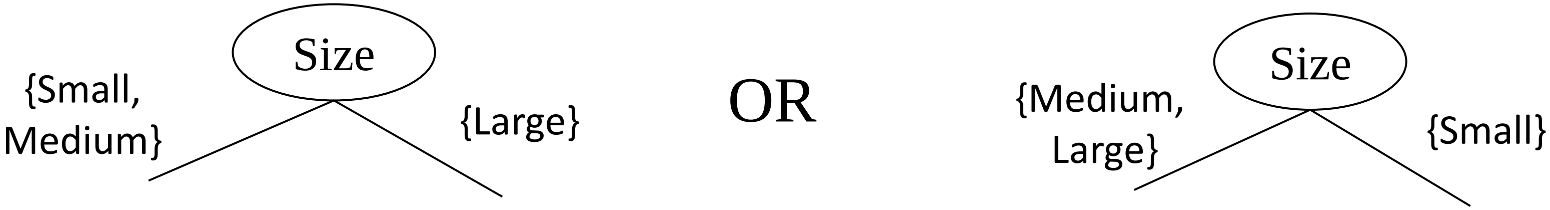

CarType
{Family, Luxury}
{Sports}

Splitting Based on Ordinal Attributes

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

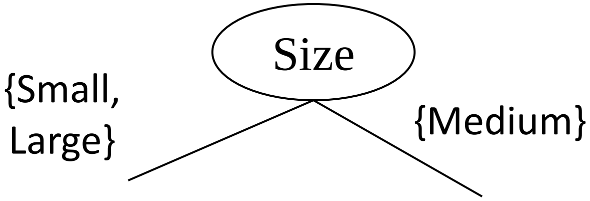


- Binary split: Divides values into two subsets
  Need to find optimal partitioning



- What about this split?

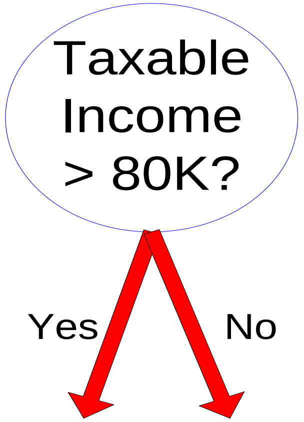
  No! the grouping should not violate the order property of the attribute values



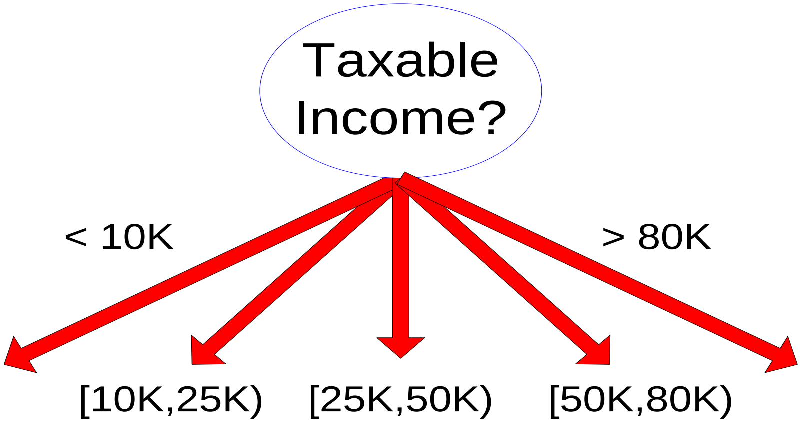
Splitting Based on Continuous Attributes

- 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

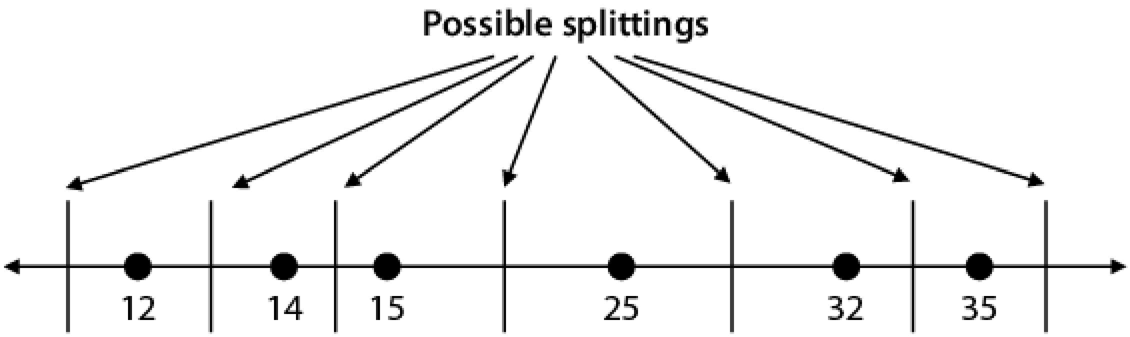


(ii) Multi-way split

Decide optimal Split Point using Gini Index
Continuous Attributes

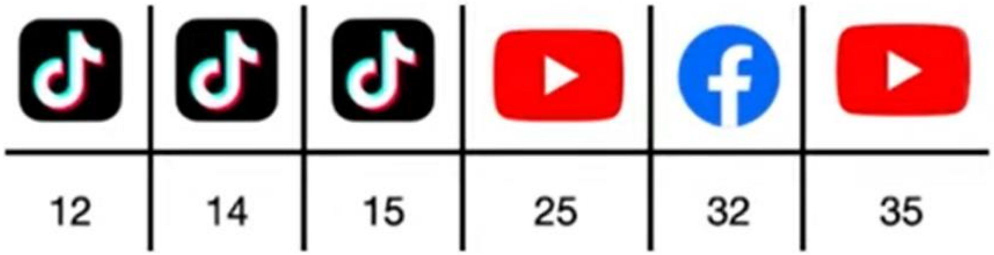
Gender	Age	App
Female	15	♪
Female	25	▶
Male	32	f
Female	35	▶
Male	12	♪
Male	14	♪

♪	♪	♪	▶	f	▶
12	14	15	25	32	35



- 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

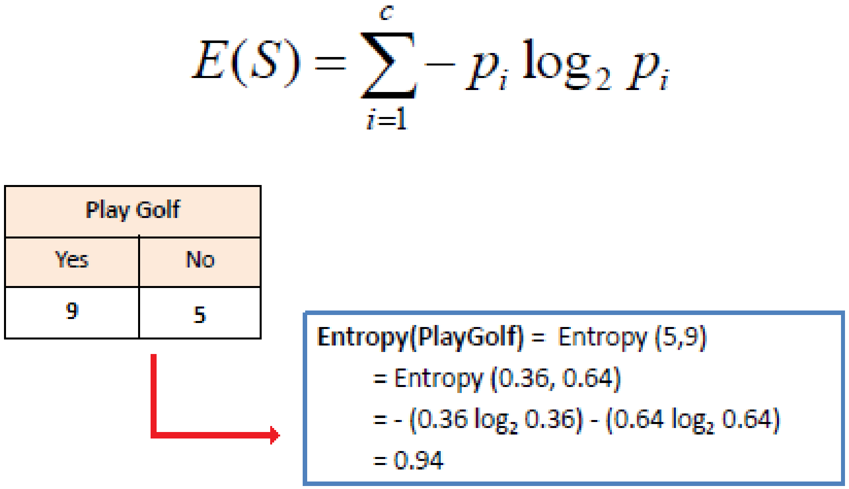


	Age													
		12		14		15		25		32		35		
	7		**13**		**14.5**		**20**		**28.5**		**33.5**		**100**	
App	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
T	0	3	1	2	2	1	3	0	3	0	3	0	3	0
Y	0	2	0	2	0	2	0	2	1	1	2	1	2	0
F	0	1	0	1	0	1	0	1	2	0	1	0	1	0
Gini	**0.611**		**0.533**		**0.417**		**0.22**		**0.416**		**0.467**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

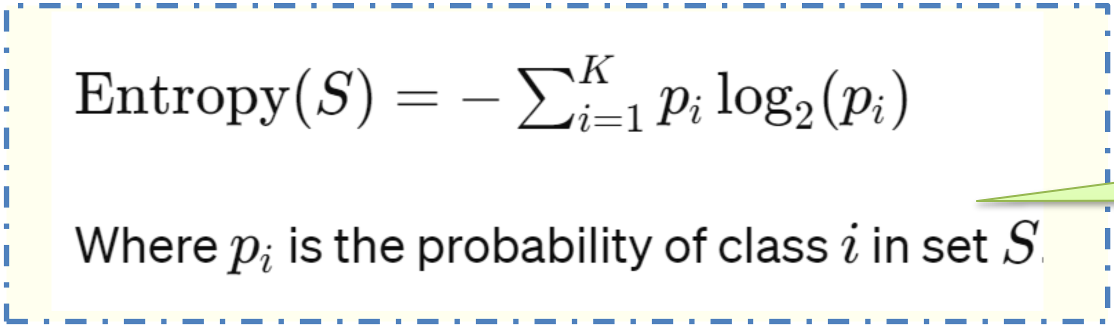
= Entropy (0.36, 0.64)

= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 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:



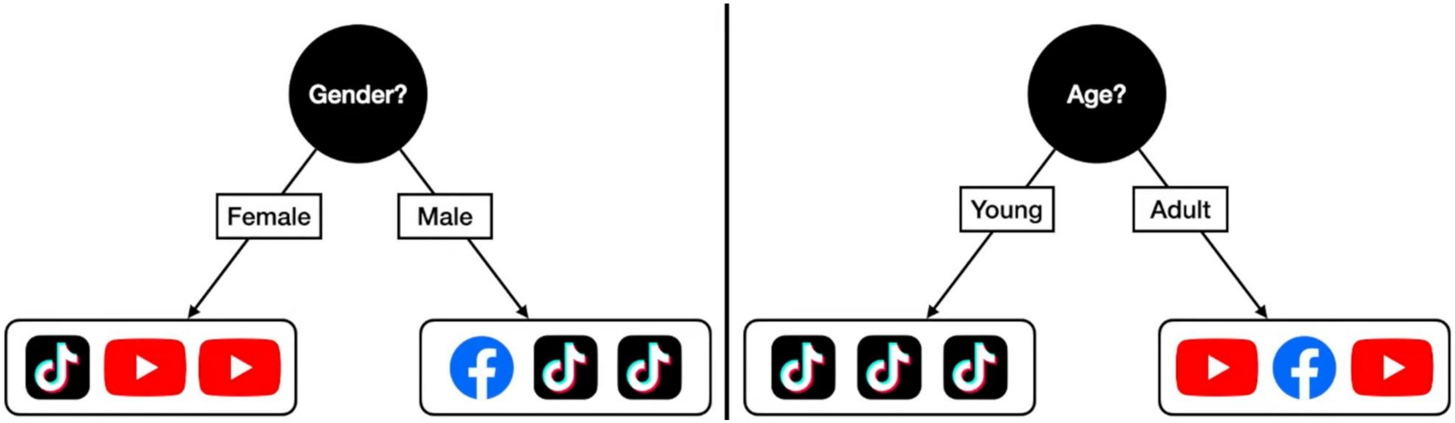
$$\text{Entropy}(S) = -\sum_{i=1}^{K} p_i \log_2(p_i)$$

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 x 0.918) + (3 x 0.918))/6 = 0.918**

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

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

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

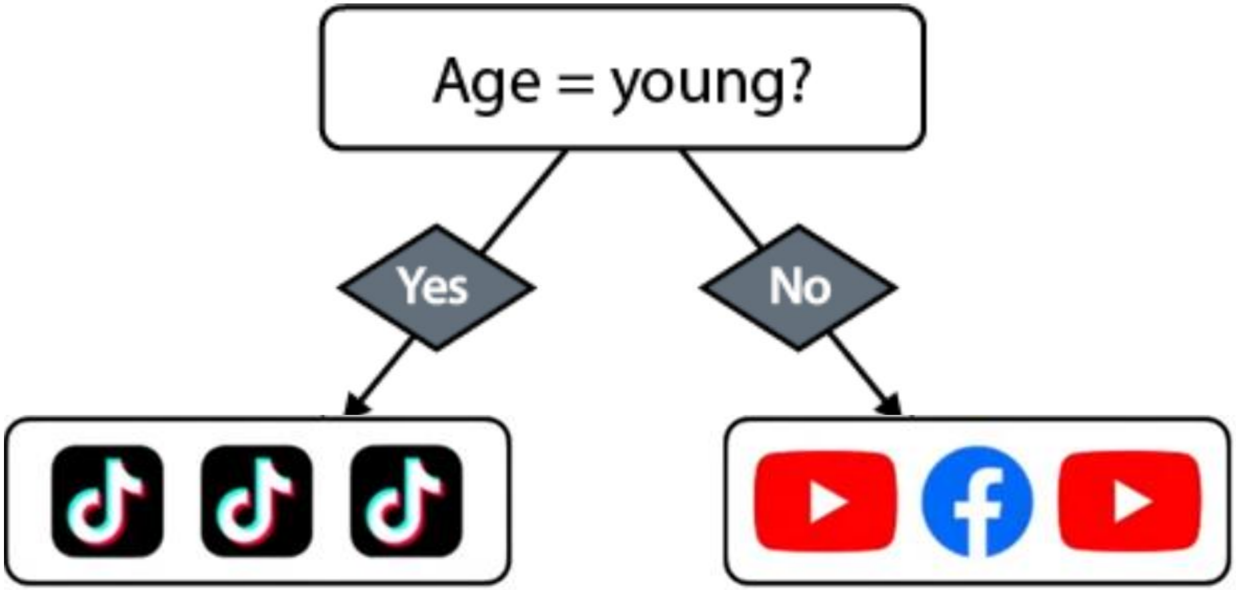
Gini $= -\mathbf{P}(\mathrm{F})\log_2 \mathbf{P}(\mathrm{F}) - \mathbf{P}(\mathrm{T})\log_2 \mathbf{P}(\mathrm{T}) = -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.918$

Classifier 2 (by age): **Avg Gini = ((3 x 0)) + (3 x 0.918))/6 = 0.459** 🏆

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

- Right leaf (adult): {Y, F, Y}. **Gini** $= -\mathbf{P}(\mathrm{Y})\log_2 \mathbf{P}(\mathrm{Y}) - \mathbf{P}(\mathrm{F})\log_2 \mathbf{P}(\mathrm{F})$

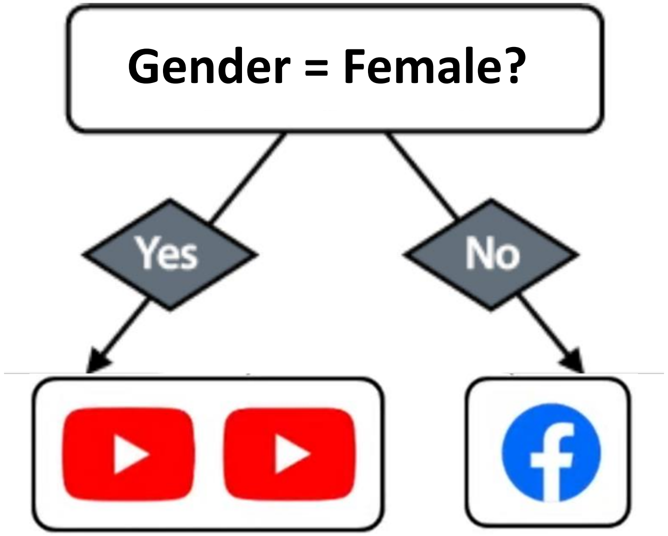
$= -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.918$



Error rate: 0%
Gini: 0
Entropy: 0

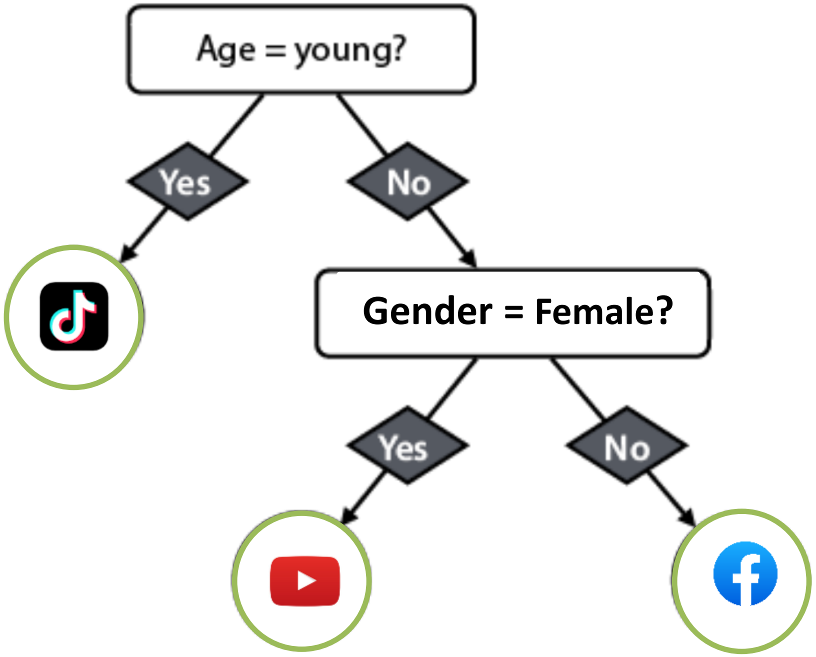
Error rate: 33.33%
Gini: 0.444
Entropy: 0.918

- 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



Accuracy: 100%
Gini: 0
Entropy: 0

Accuracy: 66.67%
Gini: 0.444
Entropy: 0.918



- 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

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

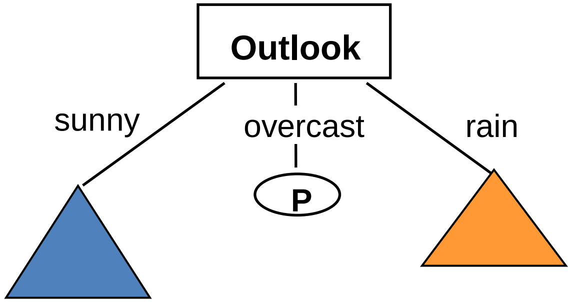
Where p_i is the probability of class i in set S

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

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

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

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

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

$$\boldsymbol{E(Outlook) = \frac{5}{15}\cdot E(Sunny) + \frac{5}{15}\cdot E(Overcast) + \frac{5}{15}\cdot E(Rain) = 0.694}$$

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

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

$$Gain(A) = Entropy(S) - Entropy(S, A)$$

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

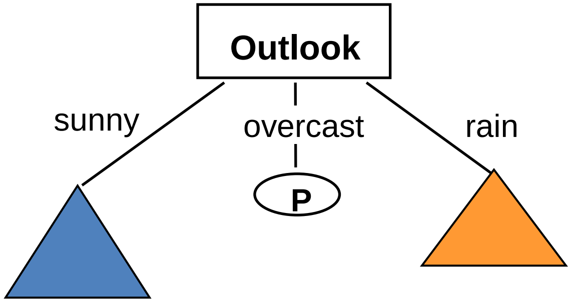
🏆 **Gain(Outlook) = 0.246**

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

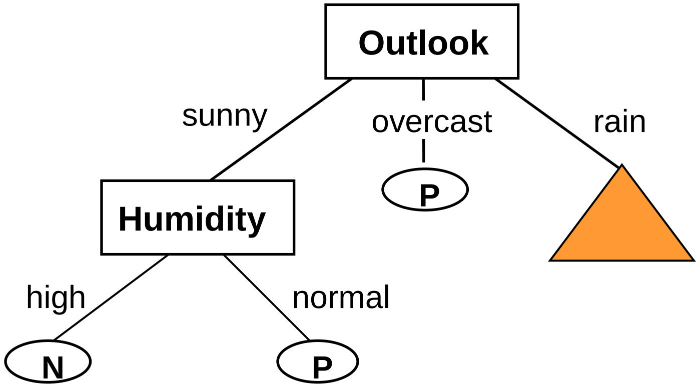
⬇

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)

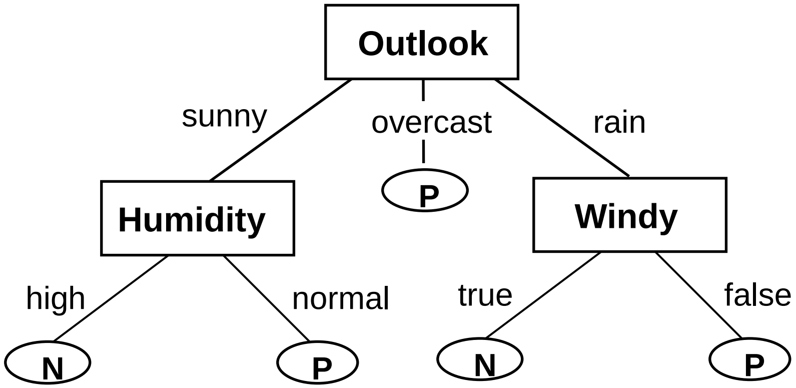
Temperature	Humidity	Windy	Class
mild	high	FALSE	P
cool	normal	FALSE	P
cool	normal	TRUE	N
mild	normal	FALSE	P
mild	high	TRUE	N

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:
  - 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
  - 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

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. 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, minimum number of instances per leaf, or no further improvement in impurity reduction

Decision tree algorithm

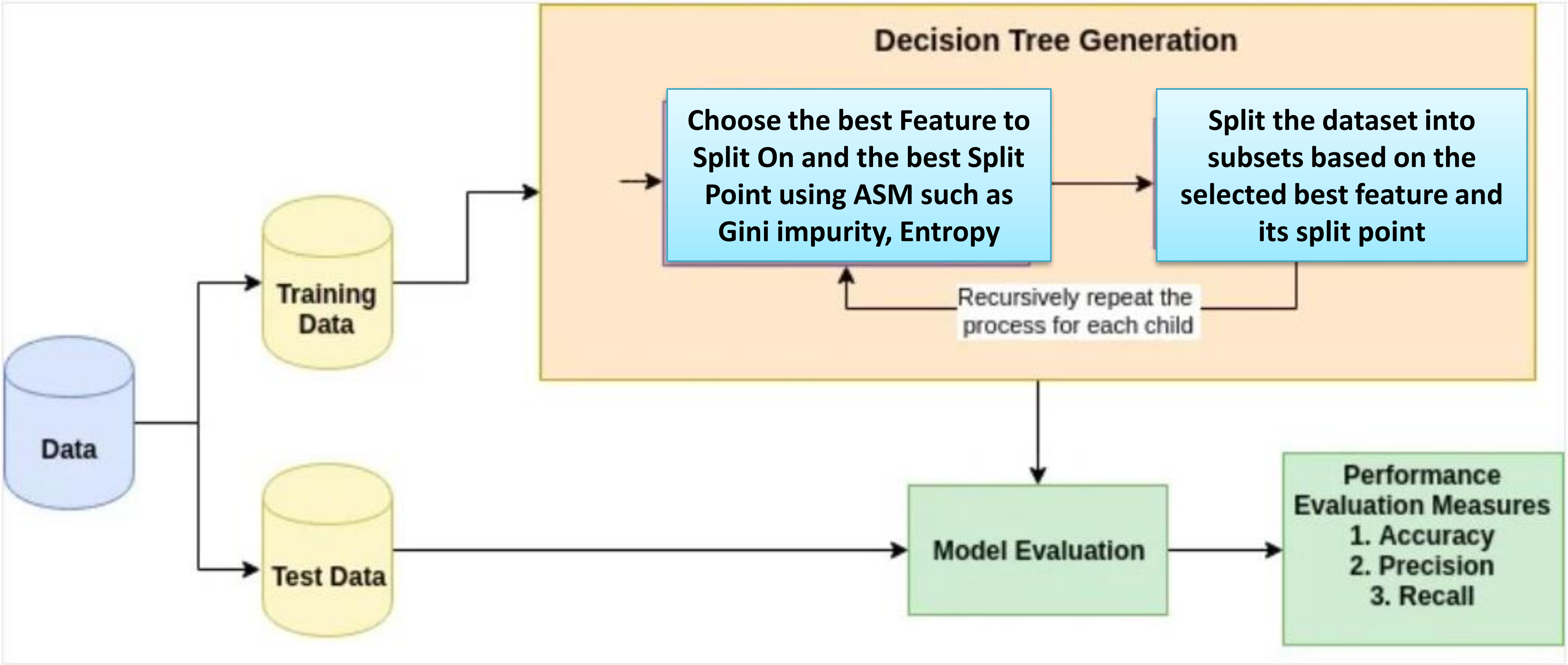
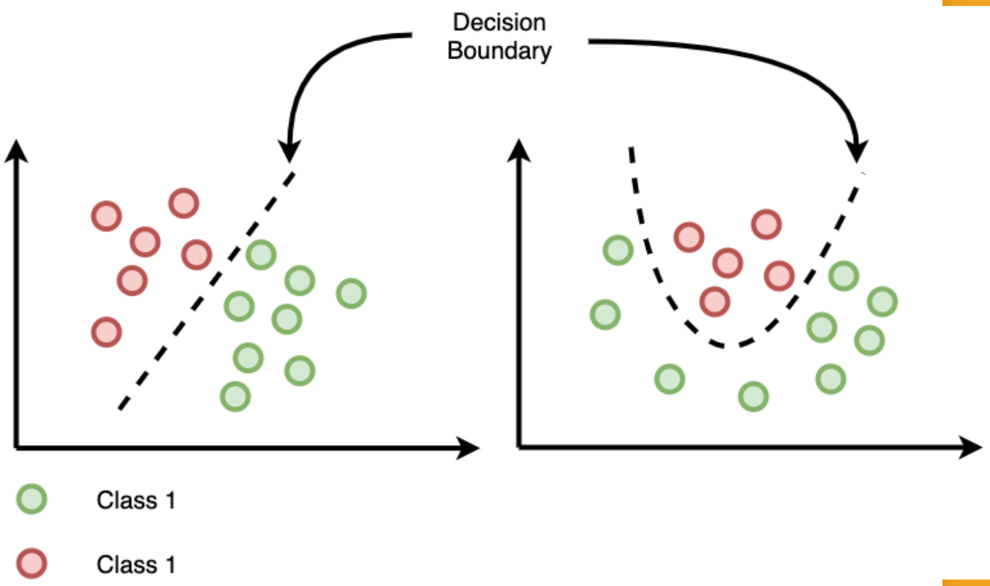
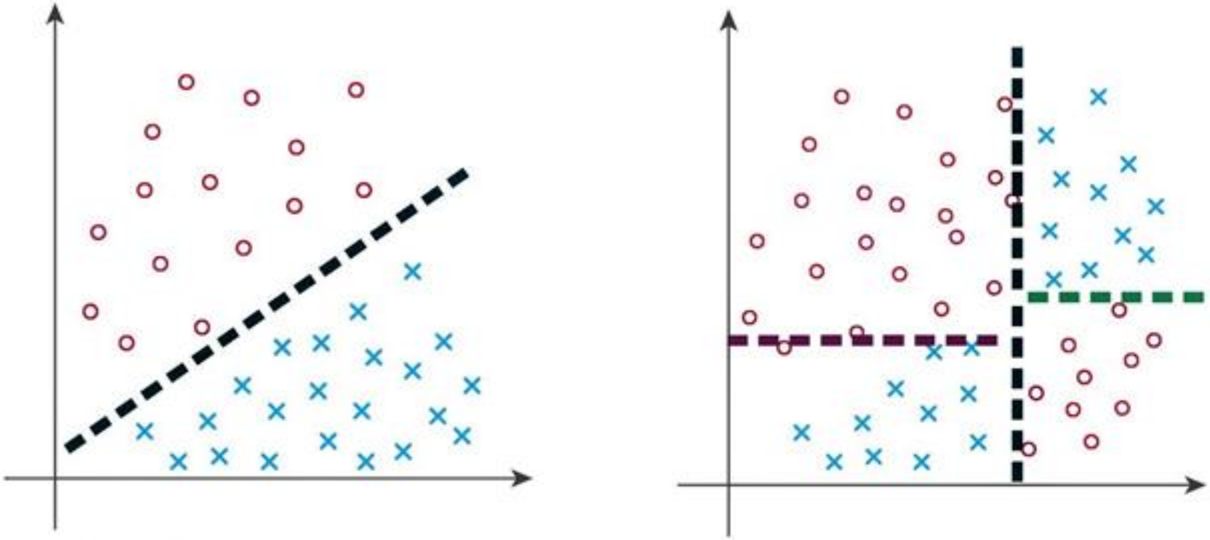

Decision Tree Generation
Choose the best Feature to Split On and the best Split Point using ASM such as Gini impurity, Entropy
Split the dataset into subsets based on the selected best feature and its split point
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3. Recall

Image source

Decision Boundaries



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



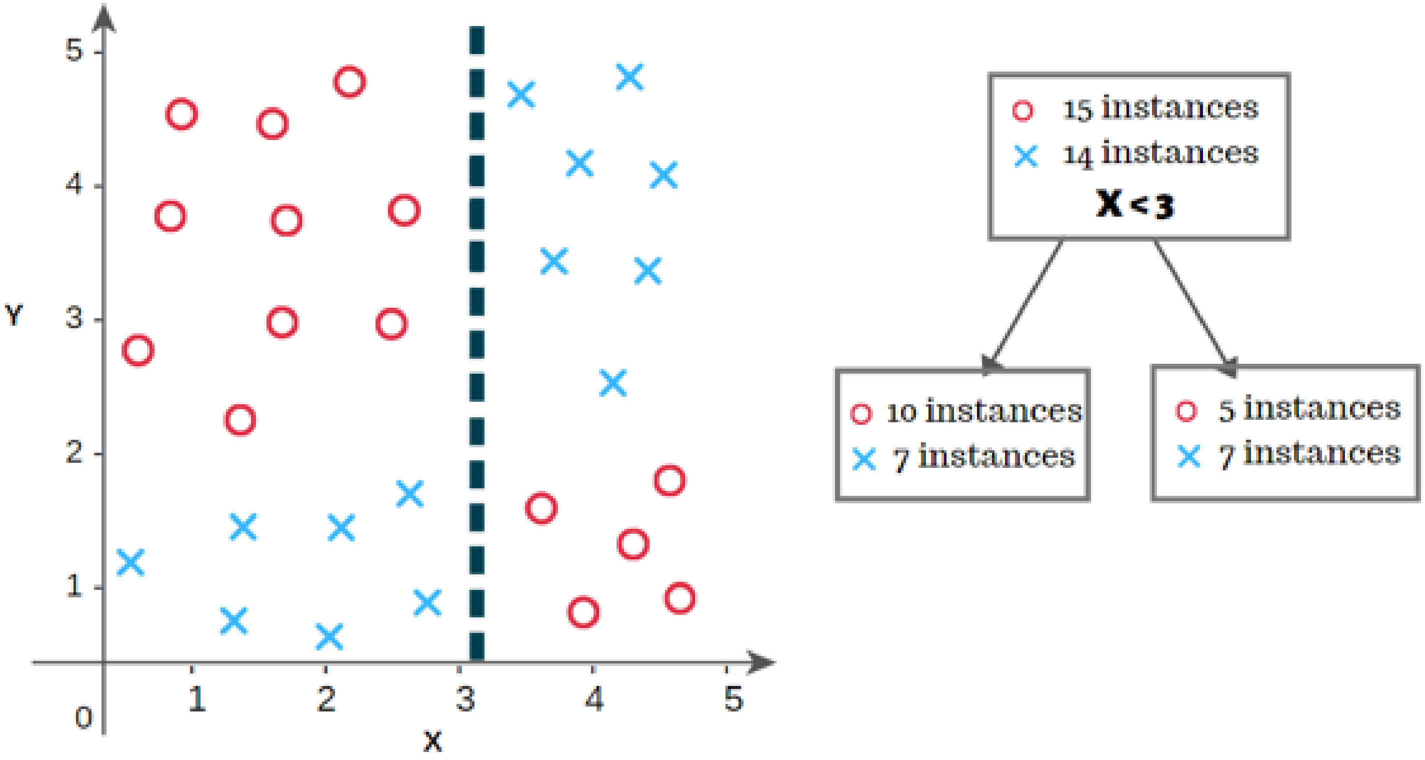
Linearly separable dataset

Linearly inseparable dataset

- 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

Source link

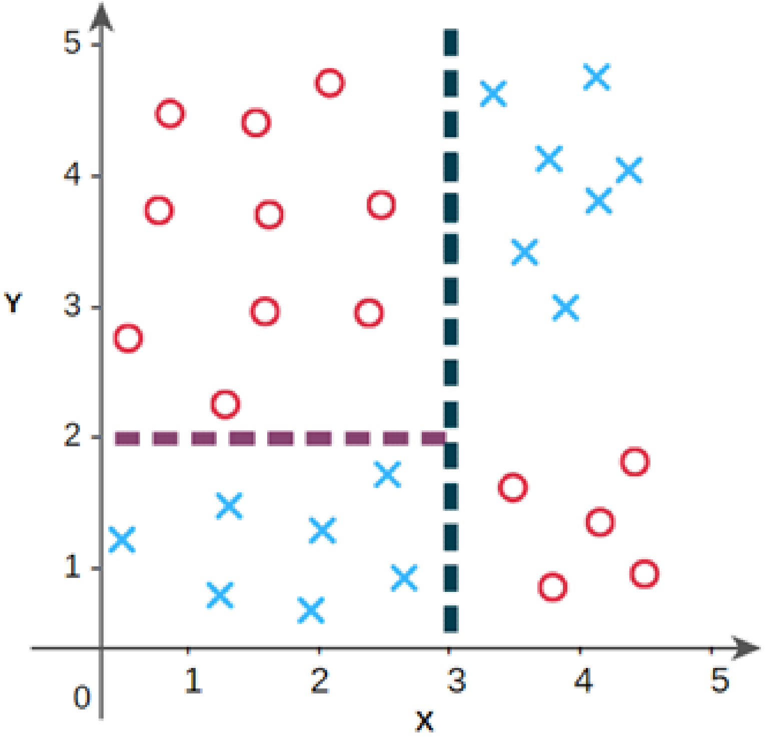
X < 3 as our Split Point

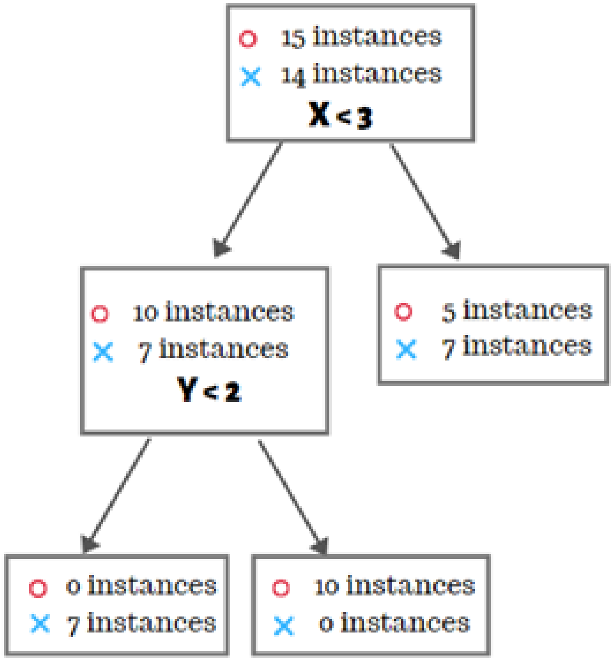


- Taking X=3 as our split point, yields 2 decision regions represented as child-nodes of the tree where each split contains the number of respective instances

Source link

Y < 2 as our Split Point for the Left region




○ 15 instances
× 14 instances
X<3

○ 10 instances
× 7 instances
Y<2

○ 5 instances
× 7 instances

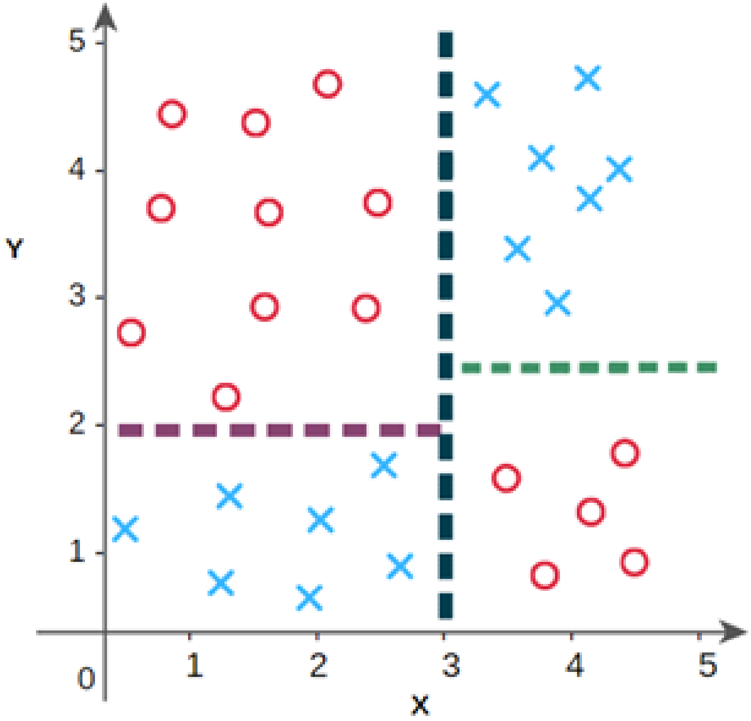
○ 0 instances
× 7 instances

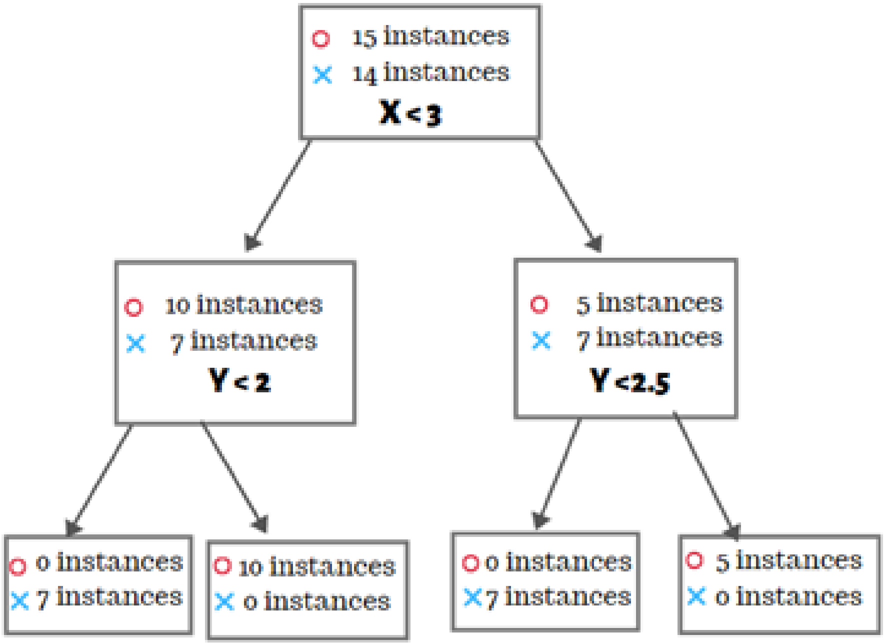
○ 10 instances
× 0 instances

- 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

Source link

Y < 2.5 as our Split Point for the Right region





○ 15 instances
× 14 instances
X<3

○ 10 instances
× 7 instances
Y<2

○ 5 instances
× 7 instances
Y<2.5

○ 0 instances
× 7 instances

○ 10 instances
× 0 instances

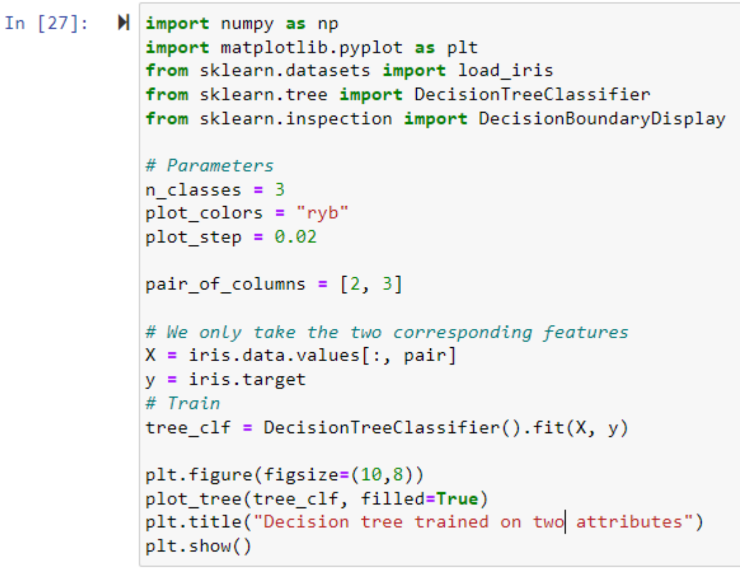
○ 0 instances
× 7 instances

○ 5 instances
× 0 instances

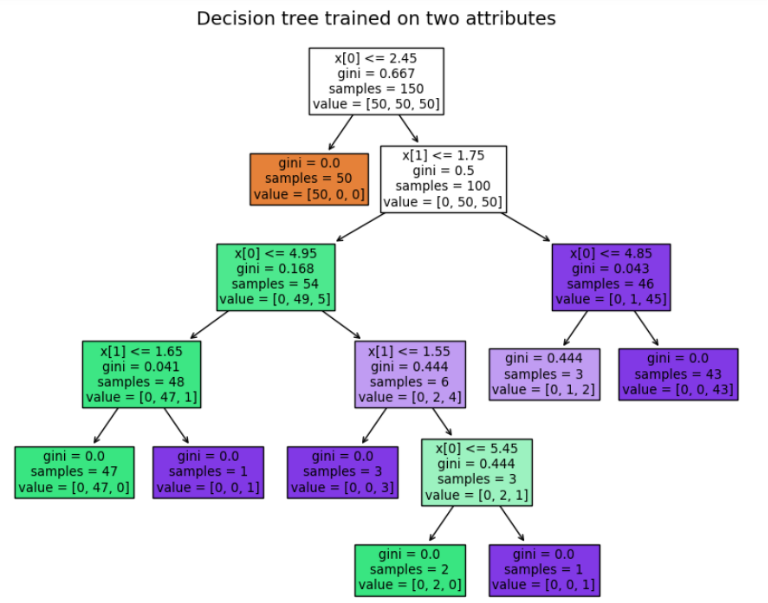
- We’ll split the right region of the resultant graph at Y=2.5
- This yields 2 decision regions represented as child-nodes of the tree. The split resulted into 2 pure leaf nodes => no further split needed for the right-side of the tree

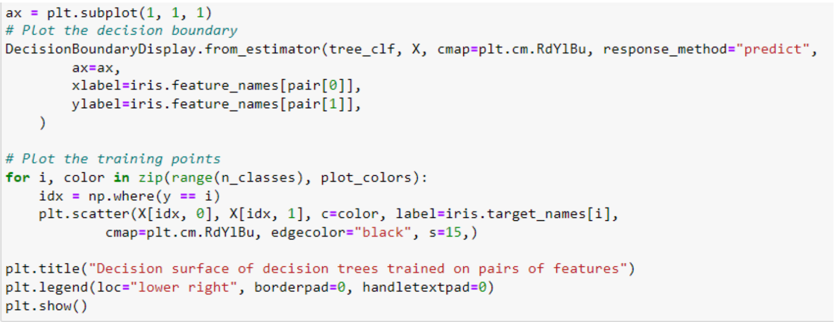
Source link

Decision Boundries

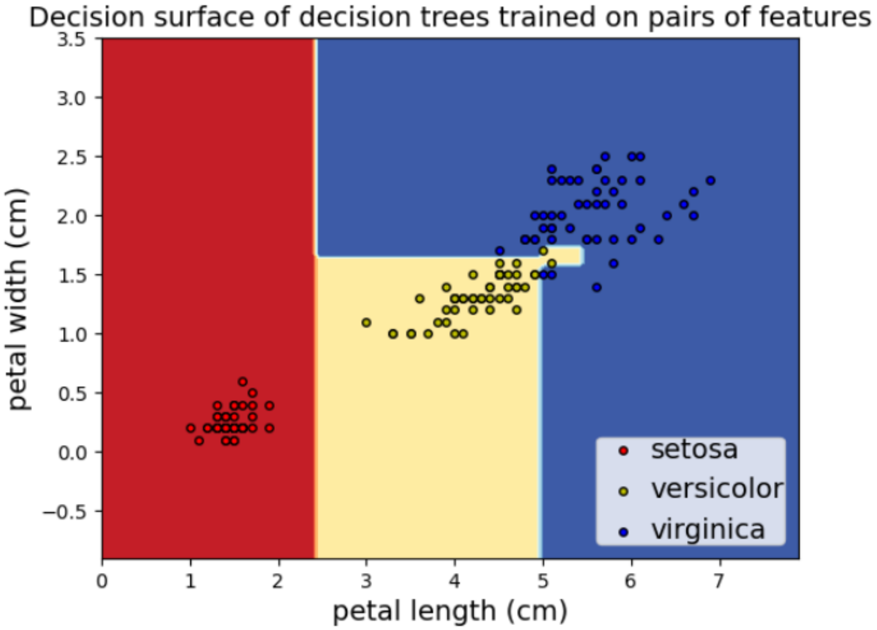


```
In [27]:  import numpy as np
          import matplotlib.pyplot as plt
          from sklearn.datasets import load_iris
          from sklearn.tree import DecisionTreeClassifier
          from sklearn.inspection import DecisionBoundaryDisplay

          # 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()
```



05.dt\5.decision_boundry_example.ipynb