5

Classification

Classification: Definition

Classification is a supervised learning task where the goal is to assign a label or category to an input based on its characteristics.

The model is trained using labeled data, meaning the input data is paired with the correct output (label).

Objective: learn to mapp a function that can predict the correct label for new, unseen data.

Classification: Examples

1. Spam Detection: Classifying emails into spam or not spam.

2. Disease Prediction: Classifying whether a person has a disease based on medical data (e.g., predicting whether a patient has diabetes based on features like age, BMI, and blood pressure).

3. Customer Churn Prediction: Predicting whether a customer will leave a service based on their usage patterns and other features.

Classification: Examples

4. Credit Scoring: Predicting whether a customer will default on a loan based on financial features such as income, credit history, and loan amount.

- In this case, the classifier would label customers into two categories: Default or No Default.
- Historical data on previous loan applicants (with features like income, age, and credit score) is used to train the model to predict the likelihood of default for new applicants.

Decision tree : definition

A decision tree is a popular classification model used in machine learning. It is a supervised learning algorithm that makes decisions by learning simple decision rules inferred from the data features. It mimics human decision-making by splitting the dataset into smaller subsets based on feature values, ultimately leading to a final decision (classification)

Decision tree

It’s called a tree because its structure resembles a tree, where each decision leads to a further branch (split) until a final classification is reached at the leaf nodes.

Decision tree: structure

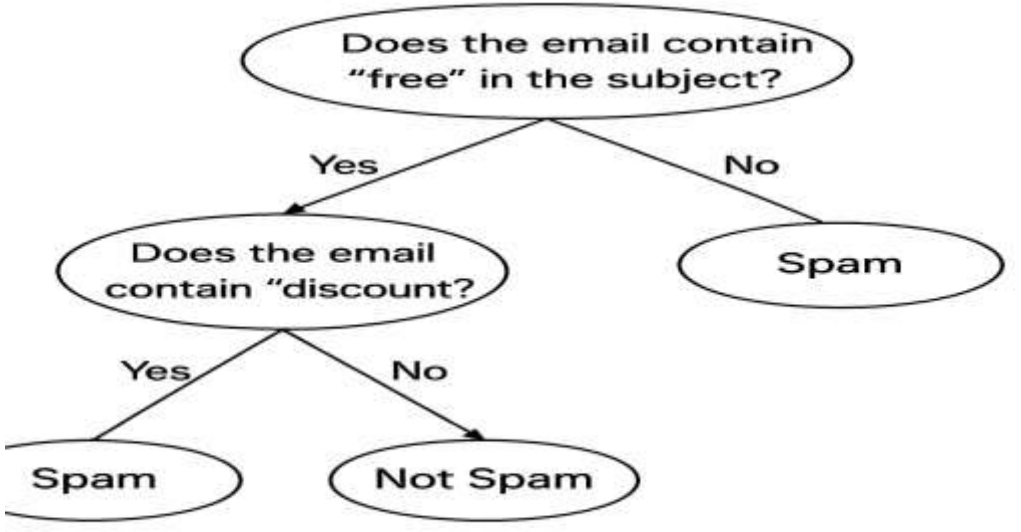
- Root Node: The topmost node of the tree that represents the entire dataset. It is the starting point of the decision-making process.
- Example: If you're classifying emails, the root node might split emails based on whether the word "free" appears in the email subject or body.
- Decision Nodes: Nodes where the dataset is split based on a specific feature (or attribute). Each decision node represents a test or decision that determines how the data should be split.

Decision tree: structure

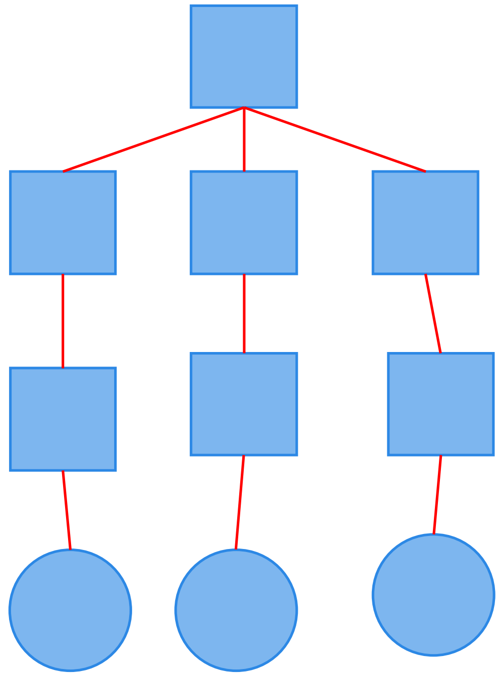
- Example: A decision node might test whether the email subject contains certain keywords like "discount" or "limited offer".
- Leaf Nodes: The terminal nodes of the tree that provide the final classification or decision.
- Example: In the email classification example, a leaf node might represent the final classification of the email as either spam or not spam.

Decision tree: structure

- Branches: The connections between nodes that represent the outcome of the decision or test at a node.


Does the email contain "free" in the subject?
Yes
No
Does the email contain "discount?
Spam
Yes
No
Spam
Not Spam

Decision tree: structure



■	Attribute
—	values of the attribute
●	Class= leaf= pure node

Decision tree : Algorithms

1. ID3 (Iterative Dichotomiser 3):

ID3 builds decision trees by selecting the feature with the highest Information Gain (based on Entropy) to split the data. It is used for classification tasks with categorical data.

2. CART :

CART constructs binary trees for both classification (using Gini Impurity) and regression (using Mean Squared Error), splitting the data recursively at each node based on the feature that minimizes the impurity.

Decision tree : Algorithms

3. C4.5 : is an extension of ID3 that uses Gain Ratio (instead of Information Gain) to split the data and supports pruning to avoid overfitting. It can handle both continuous and categorical features for classification tasks.

4. C5.0 : is an advanced version of the C4.5 decision tree algorithm and is used for classification tasks. It improves on C4.5 by providing better performance, faster training, and more efficient tree construction.

Building the decision tree : Entropy

1. Definition of Entropy

- Entropy is a measure of uncertainty or disorder in a dataset. In Decision Trees, it quantifies how "mixed" the classes are:

Low Entropy (∼0): The data is pure (all samples belong to one class).

- High Entropy (∼1): The data is evenly split between classes (maximal uncertainty).
- Formula:

Building the decision tree : Entropy

- Entropy(S)=$-\sum_{i=1}^{K} p_i log_2(p_i)$
- S = Dataset
- p_i= $\frac{|i|}{|S|}$
- k = Number of classes
- remarks : $log_2(\mathrm{x})$=$\frac{log(_x)}{log(_2)}$
- $0 \leq E(S) \leq 1$

Building the decision tree : Entropy

- Example:
- If 90% of samples are "Yes" and 10% are "No"
  Entropy=$-(0.9 log_2 0.9+0.1 log_2 0.1)$=0.47(Low)
- If 50% are "Yes" and 50% are "No":
- Entropy=$-(0.5\, log_2\, 0.5+0.5\, log_2 0.5)$=1(High)
- the value of the etropy =0 is considered as the leaf of the tree.

Building the decision tree : Entropy

"Imagine a bag of colored marbles. If all marbles are red (pure), entropy = 0. If colors are mixed randomly, entropy is high. Decision Trees aim to split the bag into smaller, pure groups."

Scenario	**Entropy Value**	**Interpretation**
All "Yes"	**0**	**Perfect order**
50% "Yes", 50% "No"	**1**	**Maximal disorder**
90% "Yes", 10% "No"	**0.47**	**Mostly ordered**

Building the decision tree : Information gain (IG)

- Decision trees rely on splitting criteria to determine the best feature to split the data at each step. Information Gain (IG) is one of the most widely used metrics because it directly measures how much a feature reduces uncertainty (entropy) in the dataset.

☐ When building a decision tree, we need to decide:

☐ Which feature should we split on first?

☐ How do we ensure that splits make the data more predictable?

Building the decision tree : Information gain (IG)

- Information Gain (IG) is a measure used in decision tree algorithms to determine the most informative feature for splitting a dataset. It quantifies how much a given feature reduces uncertainty (entropy) in the data when used for partitioning.

$$\text{Gain(S, attribute)=E(S)-}\sum_{i=1}^{K} p_i E(S_i)$$

Algorithm Steps

1. Start at the root node with the complete dataset
2. Calculate entropy of the target attribute for current node

   For each candidate attribute:

3. Calculate information gain
4. Select attribute with highest information gain
5. Create branches for each value of the selected attribute
6. Recurse on each branch using remaining attributes

of the majority class)

Algorithm Steps

7. Stop when:

☐ All instances belong to one class (pure node)

☐ No attributes remain for splitting

☐No instances in a branch (assign majority class)

Example of application :

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

Example of application

- In this table, we have 14 individuals, 4 attributes and 2 classes Yes and No.
- we calculate the entropy and the information gain for each attribute.

Example of application

1. Step 1: Calculate Root Entropy

- Total instances: 14 (9 Yes, 5 No)
- Entropy(S) = $-(9/14)*\log_2(9/14) - (5/14)*\log_2(5/14) = 0.940$

2. Step 2: Calculate Information Gains

- For each attribute (Outlook, Temp, Humidity, Wind):

Example of application

Outlook:

Sunny: [2 Yes, 3 No] → Entropy = 0.971

Overcast: [4 Yes, 0 No] → Entropy = 0

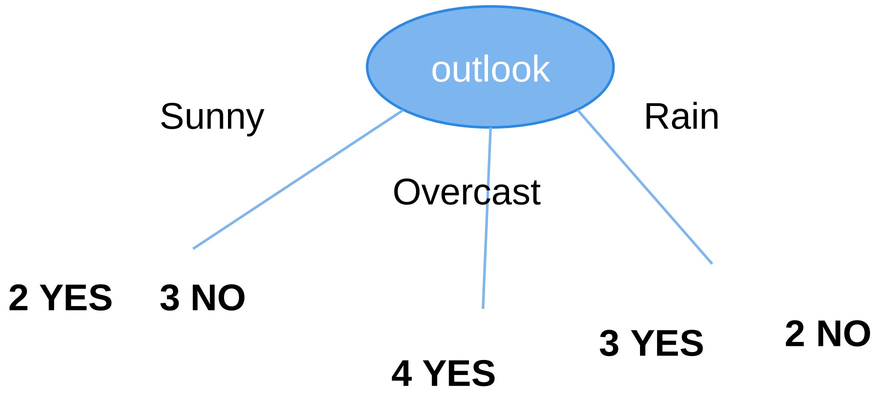
Rain: [3 Yes, 2 No] → Entropy = 0,971

Example of application

$$E(sunny) = -[\frac{2}{5} log_2 \frac{2}{5}] - [\frac{3}{5} log_2 \frac{3}{5}] = 0,971$$

$$E(rain) = -[\frac{3}{5} log_2 \frac{3}{5}] - [\frac{2}{5} log_2 \frac{2}{5}] = 0,971$$

$$E(overcast) = 0$$



Example of application

- Weighted average: (5/14)*0.971 + (4/14)*0 + (5/14)*0.971 = 0.693
- Gain(Outlook) = 0.940 - 0.693 = 0.247
- Humidity:
- High: [3 Yes, 4 No] → Entropy = 0.985
- Normal: [6 Yes, 1 No] → Entropy = 0.592
- Weighted average: (7/14)*0.985 + (7/14)*0.592 =0.789
- Gain(Humidity) = 0.940 - 0.789 = 0.151

Example of application

- Wind:
- Weak: [6 Yes, 2 No] → Entropy = 0.811
- Strong: [3 Yes, 3 No] → Entropy = 1.0
- Weighted average: (8/14)*0.811 + (6/14)*1.0 = 0.892
- Gain(Wind) = 0.940 - 0.892 = 0.048
- Temp:
- Hot: [2 Yes, 2 No] → Entropy = 1.0
- Mild: [4 Yes, 2 No] → Entropy = 0.918
-

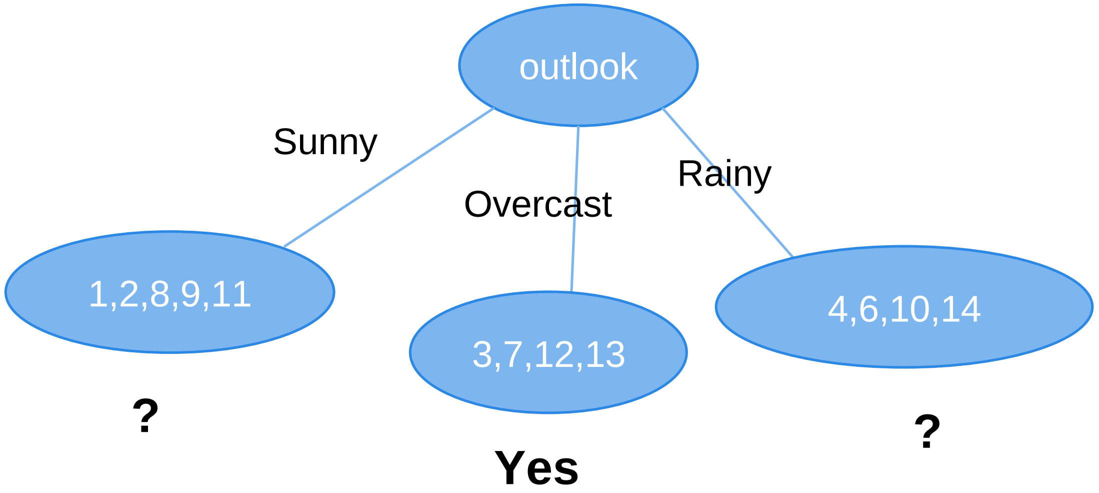
Example of application

- Cool: [3 Yes, 1 No] → Entropy = 0.811
- Weighted average: (4/14)*1.0 + (6/14)*0.918 + (4/14)*0.811 = 0.911
- Gain(Temp) = 0.940 - 0.911 = 0.029

3. Step 3: Select Best Attribute

- Outlook has the highest information gain (0.247), so we split on Outlook first.

Example of application



the overcast represents the leaf of the tree, we don't need to subdivide it. for the nodes rainy and sunny we calculate the information gain.

Example of application

4. Step 4: Recurse on Subsets

- For each Outlook value:
  - Overcast branch: Pure "Yes" → Leaf node
  - Rain branch: Recurse with remaining attributes
  - Sunny branch: Recurse with remaining attributes
- This process continues until all branches reach stopping conditions.

Example of application

- 1. Overcast Branch (Pure Node)
- Contains 4 instances (all "Yes")
- Condition: All examples have the same target value ("Yes")
- Action: Create a leaf node labeled "Yes"
- No further splitting needed

Example of application

- 2. Sunny Branch (Needs Further Splitting)
- Contains 5 instances: [2 Yes, 3 No]
- Remaining attributes: Temp, Humidity, Wind
- Current entropy: $-(2/5)\log_2(2/5) - (3/5)\log_2(3/5) = 0.971$

Example of application

- Calculate Information Gains:
- a) Temperature:
- Hot: [0 Yes, 2 No] → Entropy = 0
- Mild: [1 Yes, 1 No] → Entropy = 1
- Cool: [1 Yes, 0 No] → Entropy = 0
- Weighted average: (2/5)*0 + (2/5)*1 + (1/5)*0 = 0.4
- Gain(Temp) = 0.971 - 0.4 = 0.571

Example of application

- b) Humidity:
- High: [0 Yes, 3 No] → Entropy = 0
- Normal: [2 Yes, 0 No] → Entropy = 0
- Weighted average: (3/5)*0 + (2/5)*0 = 0
- Gain(Humidity) = 0.971 - 0 = 0.971
- c) Wind:
- Weak: [1 Yes, 2 No] → Entropy = 0.918
- Strong: [1 Yes, 1 No] → Entropy = 1
-

Example of application

- Weighted average: (3/5)*0.918 + (2/5)*1 = 0.951
- Gain(Wind) = 0.971 - 0.951 = 0.020
- Select Best Attribute:
- Humidity has the highest gain (0.971)
- Split on Humidity:
- High: 3 instances (all "No") → Leaf node "No"
- Normal: 2 instances (both "Yes") → Leaf node "Yes"

Example of application

- 3. Rain Branch (Needs Further Splitting)
- Contains 5 instances: [3 Yes, 2 No]
- Remaining attributes: Temp, Humidity, Wind
- Current entropy: $-(3/5)\log_2(3/5) - (2/5)\log_2(2/5) = 0.971$
- Calculate Information Gains:
- a) Temperature:
- Mild: [2 Yes, 1 No] $\rightarrow$ Entropy = 0.918

Example of application

- Cool: [1 Yes, 1 No] → Entropy = 1
- Weighted average: (3/5)*0.918 + (2/5)*1 = 0.951
- Gain(Temp) = 0.971 - 0.951 = 0.020
- b) Humidity:
- High: [1 Yes, 1 No] → Entropy = 1
- Normal: [2 Yes, 1 No] → Entropy = 0.918
- Weighted average: (2/5)*1 + (3/5)*0.918 = 0.951
- Gain(Humidity) = 0.971 - 0.951 = 0.020

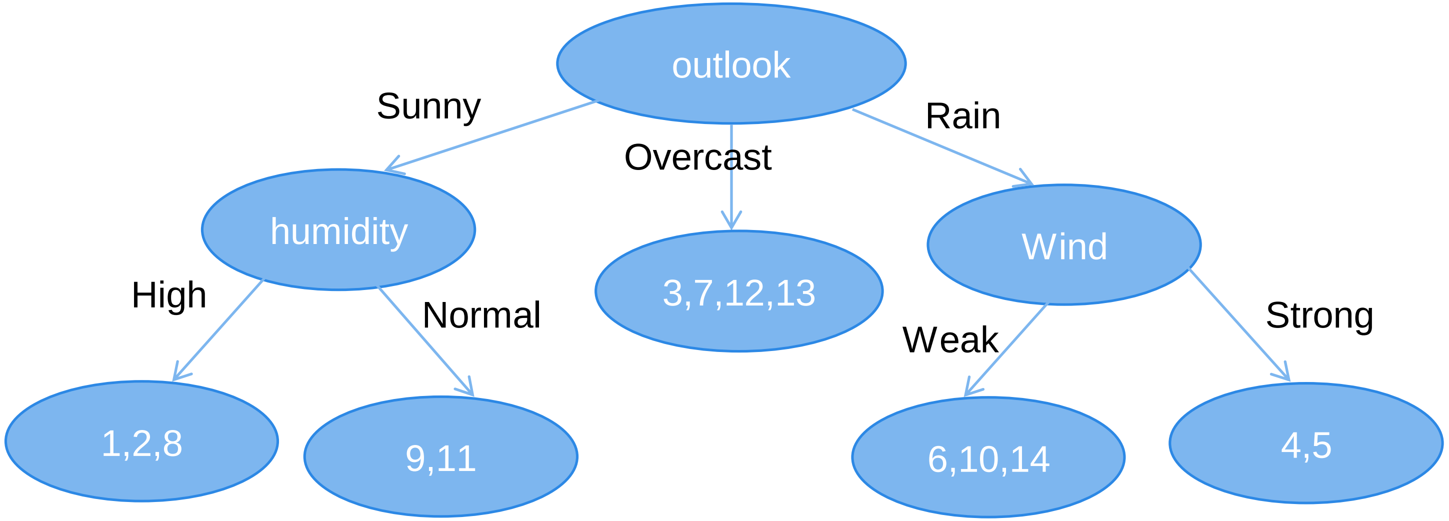
Example of application

- c) Wind:
- weak: [3 Yes, 0 No] → Entropy = 0
- Strong: [0 Yes, 2 No] → Entropy = 0
- Weighted average: (3/5)*0 + (2/5)*0 = 0
- Gain(Wind) = 0.971 - 0 = 0.971
- Select Best Attribute:
- Wind has the highest gain (0.971)

Example of application

- Split on Wind:
- Weak: 3 instances (all "Yes") → Leaf node "Yes"
- Strong: 2 instances (both "No") → Leaf node "No"

Final decision tree



- to express the class we identify all the paths from the root node into the leaf containing positive examples

Final decision tree

- the class of the positive examples is expressed as a disjunction (OU-∨) of conjunctions (ET ∧) for each path :

  (Outlook = Sunny ∧ Humidity = Normal) ∨ (Outlook = Overcast) ∨ (Outlook = Rain ∧ Wind = Weak)

- if we want to classify a new individual, we go through the tree from the root node until the leaf. for every internal node, we take the branche corresponding to the result of the test. once we reach the leaf node, we assign the example to the most frequent class of the training examples containing in this leaf.

Exercice

- Below is two simplified datasets for loan approval and disease diagnosis. Your task is to construct a decision tree step-by-step using entropy-based splits (no Python).
- **Dataset 1**: Loan ApprovalFull-Time Job
- **Dataset 2**: disease diagnosis

Exercice 1:

Applicant	Credit Score ≥ 700	Income ≥ \$50k	Full-Time Job	Loan Amount ≤ \$20k	Loan Approved
1	YES	YES	YES	YES	YES
2	YES	YES	NO	NO	NO
3	YES	NO	NO	YES	YES
4	NO	NO	NO	NO	NO
5	NO	YES	YES	YES	YES
6	NO	NO	YES	NO	NO

Exercice:2

patient	Age	BMI	Blood pressure	Family history	Exercice frequency	diagnosis
1	45	29.5	140	YES	2	YES
2	60	25.3	130	NO	3	NO
3	35	22.4	125	YES	5	YES
4	50	28.0	150	YES	1	YES
5	40	24.8	135	NO	4	NO