Classification is a **data mining function that assigns items in a collection to target categories or classes**. The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks.

Following are the examples of cases where the data analysis task is Classification −

- A bank loan officer wants to analyze the data in order to know which customer (loan applicant) are risky or which are safe.
- A marketing manager at a company needs to analyze a customer with a given profile, who will buy a new computer.

In both of the above examples, a model or classifier is constructed to predict the categorical labels. These labels are risky or safe for loan application data and yes or no for marketing data.

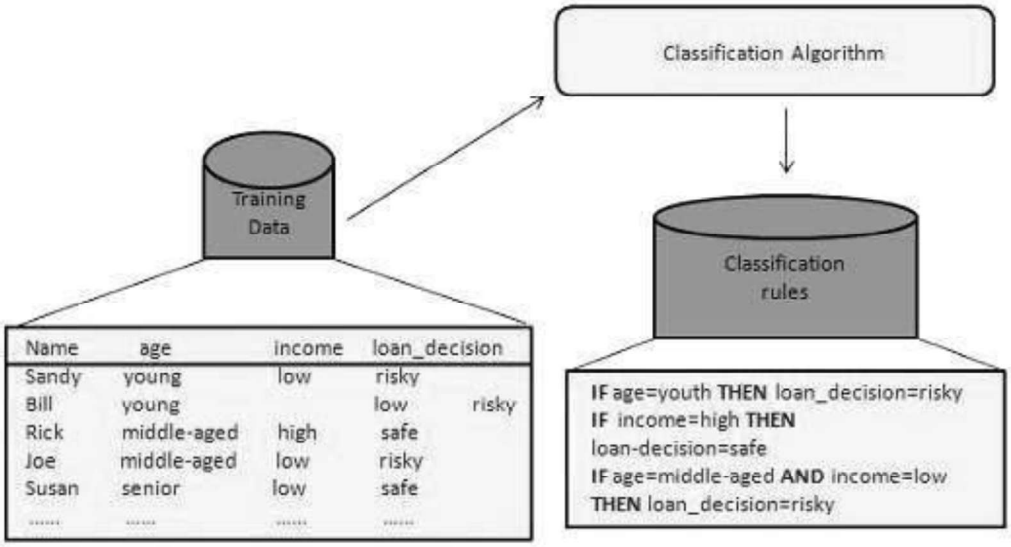
How Does Classification Works?

With the help of the bank loan application that we have discussed above, let us understand the working of classification. The Data Classification process includes two steps −

- Building the Classifier or Model
- Using Classifier for Classification

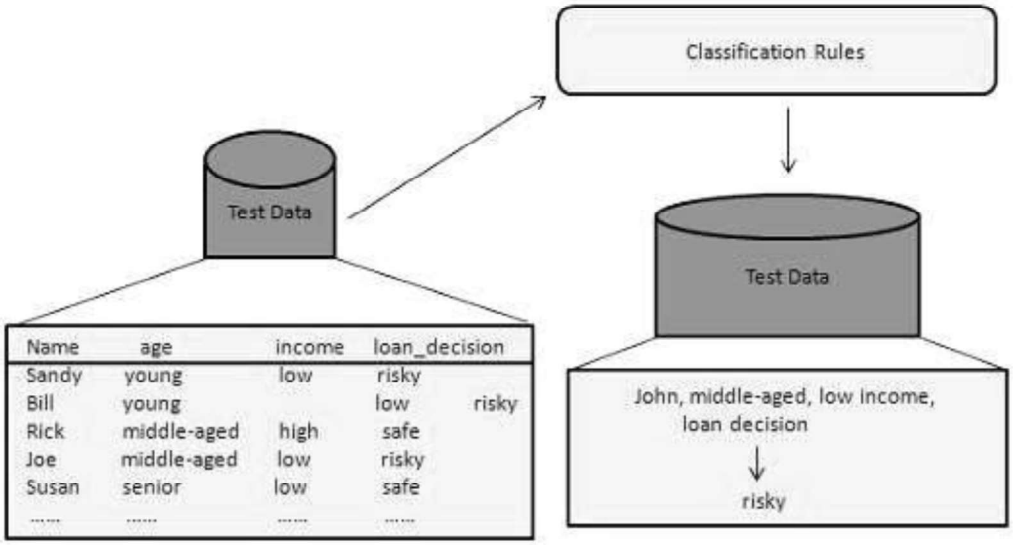
Building the Classifier or Model

- This step is the learning step or the learning phase.
- In this step the classification algorithms build the classifier.
- The classifier is built from the training set made up of database tuples and their associated class labels.
- Each tuple that constitutes the training set is referred to as a category or class. These tuples can also be referred to as sample, object or data points.



Using Classifier for Classification

In this step, the classifier is used for classification. Here the test data is used to estimate the accuracy of classification rules. The classification rules can be applied to the new data tuples if the accuracy is considered acceptable.



Classification and Prediction Issues

The major issue is preparing the data for Classification and Prediction. Preparing the data involves the following activities −

- **Data Cleaning −** Data cleaning involves removing the noise and treatment of missing values. The noise is removed by applying smoothing techniques and

the problem of missing values is solved by replacing a missing value with most commonly occurring value for that attribute.

- **Relevance Analysis −** Database may also have the irrelevant attributes. Correlation analysis is used to know whether any two given attributes are related.
- **Data Transformation and reduction −** The data can be transformed by any of the following methods.
    - **Normalization −** The data is transformed using normalization. Normalization involves scaling all values for given attribute in order to make them fall within a small specified range. Normalization is used when in the learning step, the neural networks or the methods involving measurements are used.
    - **Generalization −** The data can also be transformed by generalizing it to the higher concept. For this purpose we can use the concept hierarchies.

Note − Data can also be reduced by some other methods such as wavelet transformation, binning, histogram analysis, and clustering.

Comparison of Classification and Prediction Methods

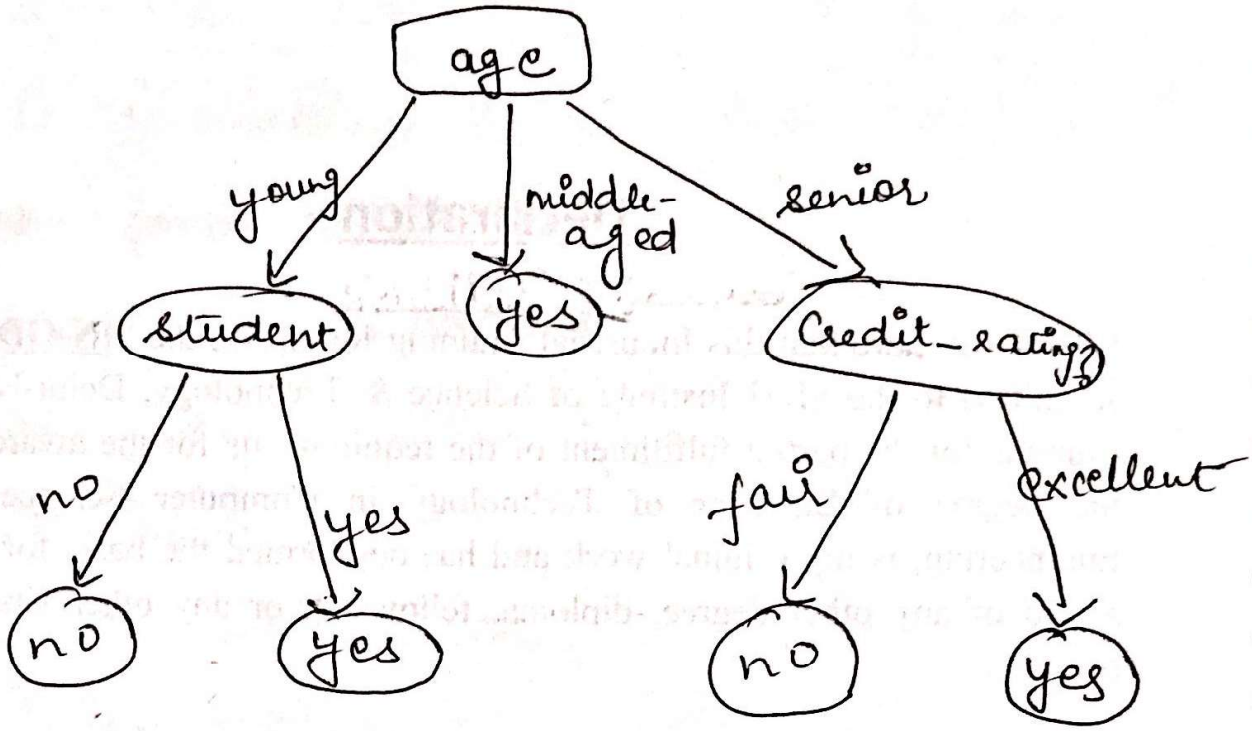
Here is the criteria for comparing the methods of Classification and Prediction **−**

- **Accuracy −** Accuracy of classifier refers to the ability of classifier. It predict the class label correctly and the accuracy of the predictor refers to how well a given predictor can guess the value of predicted attribute for a new data.
- **Speed −** This refers to the computational cost in generating and using the classifier or predictor.
- **Robustness** − It refers to the ability of classifier or predictor to make correct predictions from given noisy data.
- **Scalability −** Scalability refers to the ability to construct the classifier or predictor efficiently; given large amount of data.
- **Interpretability** − It refers to what extent the classifier or predictor understands.

Decision Tree induction

It is a method of learning the decision trees from the training sets. The dataset is broken down into smaller subsets and is present in the form of nodes of a trees. The tree structure has a root node, internal nodes or decision nodes, leaf nodes and branches. The root node is the topmost node.

→ Each internal node denotes a test of an attribute, each branch ~~nodes~~ denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node.



Decision tree induction algorithm

developed a decision tree algorithm known as ID3 (Iterative Dichotomiser) later, he presented C4.5, which was the successor of ID3. ID3 and C4.5 adopt a greedy approach. In this algo, there is no backtracking; the trees are constructed in a top-down recursive divide and conquer manner.

Tree Pruning

Tree Pruning is performed in order to remove anomalies in the training data due to noise or outliers. The pruned trees are smaller and less complex.

Tree-Pruning Approaches

There are two approaches to prune a tree -

Pre-pruning → The tree is pruned by halting its construction early.

Post-pruning → This approach removes a sub-tree from a fully grown tree.

Key factors

Entropy :-

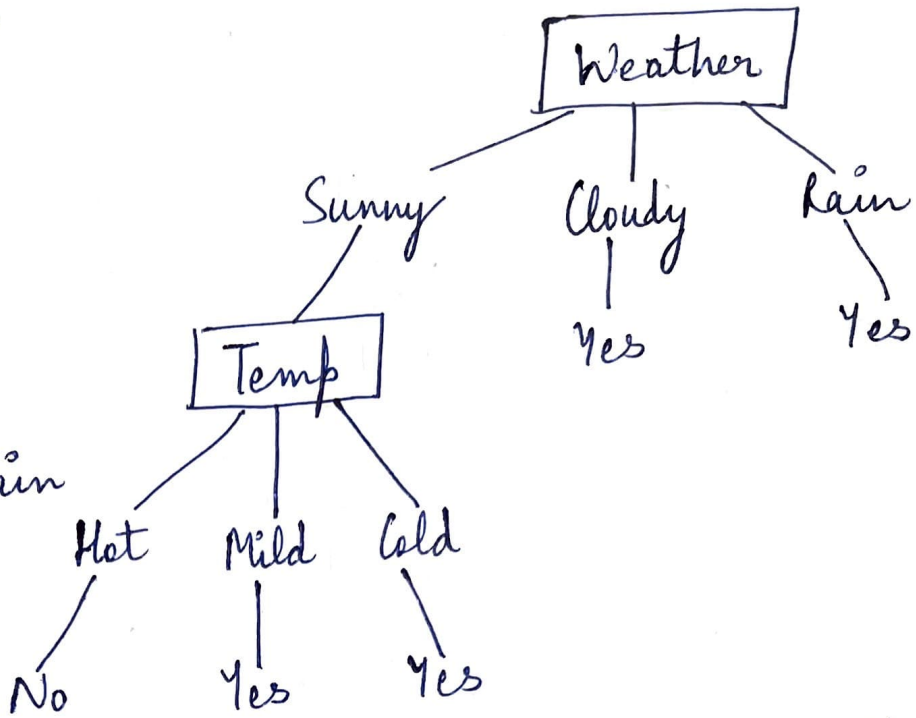
It refers to a common way to measure impurity. In the decision trees, it measures the randomness or impurity in data sets.

Information Gain :-

Information gain refers to the decline in entropy after the dataset is split. It is also called entropy reduction. Building a decision tree is all about discovering attributes that return the highest data gain.

Decision Tree Induction - Machine Learning algorithm used for classification.

→ Tree Structure
→ Decision Nodes
→ Leaf Nodes
→ Splitting
→ Entropy
→ Information Gain
→ Pruning



Entropy is a measure of uncertainity or impurity in a dataset. It tells us how mixed the class labels are in a given dataset.

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

$S \rightarrow$ Dataset
$c \rightarrow$ no. of classes
$p_i \rightarrow$ proportion of class i in dataset
Entropy = 0 if all instances belong to same class

Information Gain measures how much entropy is reduced after a dataset is split on an attribute. It helps select the best attribute for splitting at each step.

Day	Weather	Temperature	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Cloudy	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Cloudy	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Cloudy	Mild	High	Strong	Yes
13	Cloudy	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Calculate IG of Weather

(1) Entropy of entire dataset

$$S\{+9, -5\} = \frac{-9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$$

(2) Entropy of all attributes

Entropy of Sunny $(+2, -3) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}$

$= 0.97$

Entropy of cloudy $(+4, -0) = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0$

Entropy of Rain $\{+3, -2\} = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.97$

$IG = \text{Entropy (Whole Data)} - \frac{5}{14}Ent(S) - \frac{4}{14}Ent(C) - \frac{5}{4}Ent(R)$

$= 0.246$

Calculate IG of Humidity

Entropy of all attributes

" " High $\{+3, -4\} = \frac{-3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} = 0.98$

" " Normal $\{+6, -1\} = \frac{-6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7} = 0.59$

$IG = \text{Entropy (whole data)} - \frac{7}{14}Ent(H) - \frac{7}{14}Ent(N)$

$= 0.15$

Calculate IG of Wind

Entropy of Strong $\{+3, -3\} = \frac{-3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 1.0$

Entropy of Normal $\{+6, -2\} = \frac{-6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8} = 0.81$

$IG = \text{Entropy (whole data)} - \frac{6}{14}Ent(S) - \frac{8}{14}Ent(W)$

$= 0.0478$

Entropy of Temperature

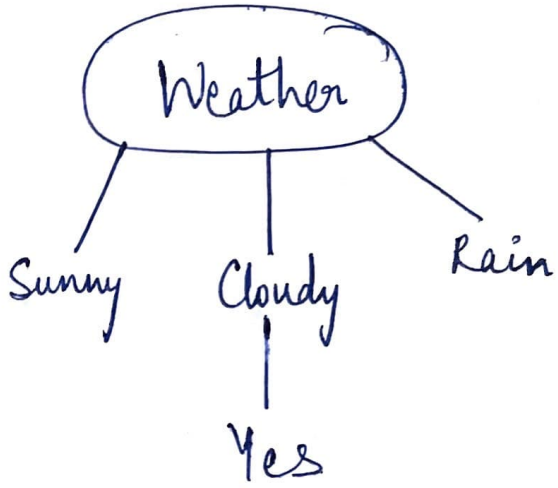
Entropy of Hot $\{+2, -2\} = \frac{-2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 1.0$

Entropy of Wind $\{+4, -2\} = \frac{-4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} = 0.91$

Entropy of Cold $\{+3, -1\} = \frac{-3}{4}\log_2\frac{3}{4} - \frac{1}{3}\log_2\frac{1}{3} = 0.81$

$IG = \text{Entropy (whole data)} - \frac{4}{14}\,Ent(H) - \frac{6}{14}\,Ent(M) - \frac{4}{14}\,Ent(C)$

$= 0.029$



Day	Weather	Temperature	Humidity	Wind	Play
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

IG of Temperature

① Entropy of Sunny $= \{+2, -3\} = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}$

$= 0.97$

② Entropy of Hot $= \{+0, -2\} = 0$

Entropy of Mild $= \{+1, -1\} = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1.0$

Entropy of Cool $= \{+1, -0\} = -\frac{1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1} = 0$

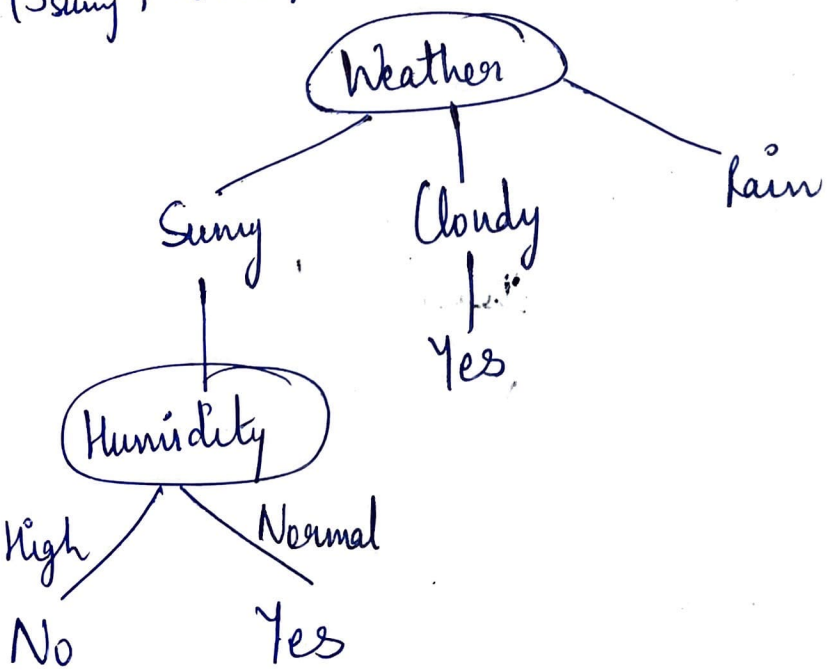
$\therefore$

$IG = \text{Entropy}(\text{Sunny}) - \frac{2}{5}\text{Ent}(H) - \frac{2}{5}\text{Ent}(M) - \frac{1}{5}\text{Ent}(C)$

$IG(S_{sun}, \text{Temp}) = 0.57$

$IG(S_{sunny}, \text{Humidity}) = 0.97$

$IG(S_{sunny}, \text{Wind}) = 0.019$

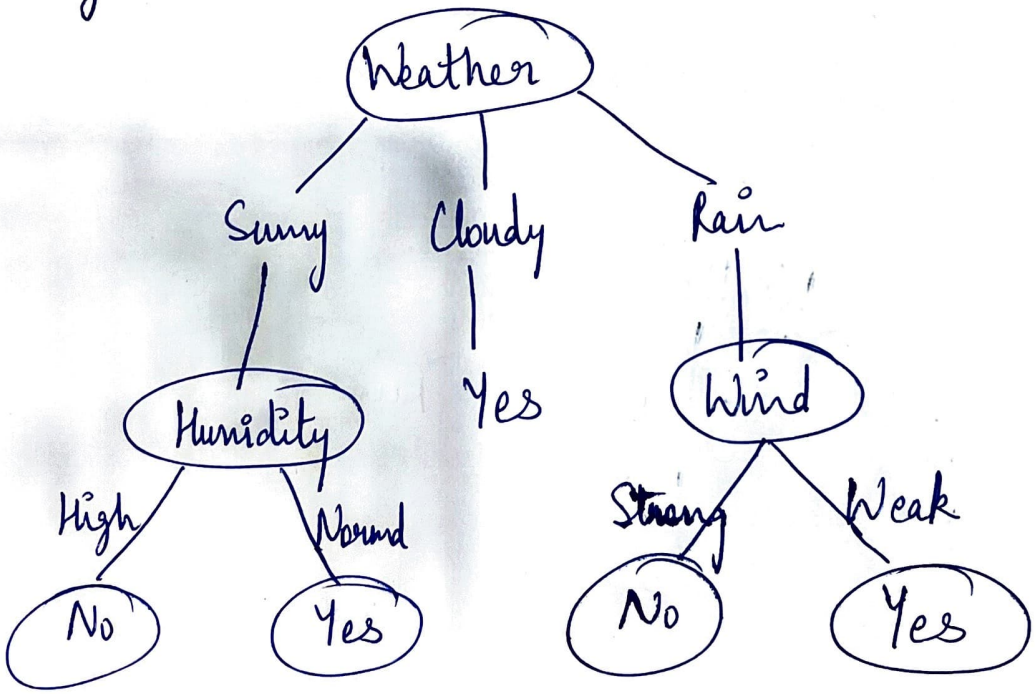


Day	Weather	Temperature	Humidity	Wind	Play
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

IG of Temp = 0.019

IG of Humidity = 0.019

IG of Wind = 0.97



Attribute selection Measure

In a **decision tree**, selecting the best attribute (feature) to split at each node is crucial for building an effective model. **Attribute Selection Measures (ASMs)** help in choosing the attribute that best separates the data into distinct classes. The most commonly used measures are:

1. Information Gain (IG)

- Based on **Entropy**, which measures the impurity (randomness) in data.
- Information Gain calculates the reduction in entropy after a dataset is split on an attribute.
    - Formula:

$$IG(S, A) = Entropy(S) - \sum \left(\frac{|S_v|}{|S|} \times Entropy(S_v) \right)$$

      where:

      - S is the dataset,
      - A is the attribute,
      - S_v is the subset after splitting on A.
- **Higher IG** means the attribute provides a better split.
- Used in **ID3 algorithm**.

---

2. Gain Ratio

- Addresses the bias of **Information Gain** towards attributes with many values.
- Normalizes **Information Gain** using **Split Information** (which measures the potential splits of an attribute).
    - Formula:

$$Gain\ Ratio(A) = \frac{Information\ Gain(A)}{Split\ Information(A)}$$

      Higher Gain Ratio means a better attribute selection.
- Used in **C4.5 algorithm**.

---

3. Gini Index

- Measures the impurity of a dataset.
  - Formula:

$$Gini(S) = 1 - \sum p_i^2$$

    where p_i is the probability of class i in dataset S.
- A **lower Gini Index** indicates a better attribute.
- Used in **CART (Classification and Regression Trees)**.

---

4. Chi-Square Test

- Measures the **statistical significance** of an attribute.
  - Formula:

$$\chi^2 = \sum \frac{(O - E)^2}{E}$$

    where:
    - O is the observed frequency,
    - E is the expected frequency.

  A **higher chi-square value** means a stronger correlation with the target variable.

---

5. Reduction in Variance (for Regression Trees)

- Used when the target variable is continuous.
- Splits the dataset by minimizing variance in child nodes.
- Formula:

$$Reduction\,in\,Variance = Variance_{parent} - \sum \left(\frac{|S_v|}{|S|} \times Variance_{S_v} \right)$$

---

6. Mean Squared Error (MSE) (for Regression Trees)

- Used in regression decision trees.
- Splits the dataset by minimizing the **Mean Squared Error**.

Summary Table

Measure	Used in	Works on	Goal
Information Gain	ID3	Categorical	Reduce entropy
Gain Ratio	C4.5	Categorical	Normalize IG
Gini Index	CART	Categorical	Reduce impurity
Chi-Square Test	CHAID	Categorical	Statistical correlation
Reduction in Variance	CART	Continuous	Reduce variance
Mean Squared Error	CART	Continuous	Minimize MSE

Importance of Attribute Selection Measures in Decision Trees

Attribute Selection Measures (ASMs) play a crucial role in **decision tree learning** as they determine how effectively the tree splits data at each step. Their importance can be understood in the following ways:

1. Enhancing Predictive Accuracy

- Selecting the most relevant attributes ensures that the decision tree accurately classifies unseen data.
- Helps prevent incorrect splits that can lead to poor classification.

2. Reducing Overfitting

- If a tree is built using irrelevant attributes, it can become **overly complex** and fit noise rather than actual patterns.
- ASMs help in choosing **only the most informative features**, making the tree more **generalizable** to new data.

3. Improving Computational Efficiency

- Decision tree training involves multiple calculations for each attribute.
- Efficient ASMs (like **Gini Index** or **Information Gain**) help speed up training by **minimizing unnecessary computations**.

4. Reducing Tree Size (Pruning Effect)

- Selecting the best attributes early in the tree leads to a **smaller tree**.
- A smaller tree is **easier to interpret** and has **lower storage and computational requirements**.

5. Handling Data with Multiple Attributes

- Real-world datasets contain **both categorical and continuous** features.
- ASMs like **Information Gain (for categorical)** and **Reduction in Variance (for continuous)** ensure the right approach for different data types.

6. Ensuring Better Data Splits

- The **right attribute selection** maximizes the **purity** of the resulting subsets.
- This ensures that the decision boundary between classes is clear, improving classification quality.

CART Algorithm

The CART (Classification and Regression Trees) algorithm is a widely used decision tree algorithm that builds models for both classification and regression tasks. It was introduced by Breiman et al. (1984) and is a foundational algorithm in machine learning.

Key Features of CART

✅ Can be used for classification (discrete output) and regression (continuous output)
✅ Uses binary splits (each node splits into exactly two branches)
✅ Uses Gini Index for classification and Variance Reduction for regression
✅ Supports pruning to prevent overfitting

---

2. Working of CART Algorithm

The CART algorithm builds a decision tree using the following steps:

Step 1: Selecting the Best Attribute (Feature Selection)

- For Classification: Uses Gini Index to measure impurity and selects the attribute that minimizes impurity.
- For Regression: Uses Variance Reduction or Mean Squared Error (MSE) to minimize prediction error.

Step 2: Splitting the Data

- The dataset is recursively split into two homogeneous groups (binary split).
- Each split reduces impurity or variance, making the decision boundaries clearer.

Step 3: Stopping Criteria

- The tree stops growing when:

- A predefined maximum depth is reached.
- Further splitting does not significantly reduce impurity/variance.
- The number of samples in a node is below a threshold.

Step 4: Pruning the Tree (Optional)

- Post-pruning is applied to remove unnecessary branches, reducing overfitting.
- A validation set is used to decide the optimal pruning level.

Attribute Selection Measures in CART

(A) For Classification: Gini Index

- Measures how impure a dataset is.
  - The Gini Index for a dataset S is:

$$Gini(S) = 1 - \sum p_i^2$$

  - where p_i is the probability of class i.

(B) For Regression: Variance Reduction

- Measures the spread of numerical values.

The variance of a dataset is:

$$Variance = \frac{1}{n}\sum (x_i - \bar{x})^2$$

Lower variance = Better split.

Example of CART Algorithm

Example 1: Classification (Using Gini Index)

Dataset

Person	Age	Income	Buys Car (Class)
A	25	Low	No
B	35	High	Yes
C	40	Medium	No
D	50	High	Yes
E	30	High	Yes

Step 1: Calculate Gini for the Parent Node

- Class Distribution: 3 Yes, 2 No

$$Gini(S) = 1 - \left(\left(\frac{3}{5}\right)^2 + \left(\frac{2}{5}\right)^2 \right) = 0.48$$

Step 2: Compute Gini for Splitting on "Income"

- Split into Low, Medium, High groups and compute Gini values for each.
- Compute Weighted Gini Index for the split.
- Choose the split with the lowest Gini Index.

Example 2: Regression (Using Variance Reduction)

Dataset

	House Size (sq.ft)	Price ($1000s)
A	1400	245
B	1600	312
C	1700	279
D	1875	308

Step 1: Compute Variance for Parent Node

- Compute the mean and variance of house prices.

Step 2: Compute Variance Reduction for a Split

- If splitting at Size = 1600, compute variance for left and right nodes.
- Compute Variance Reduction:

$$Reduction = Variance_{parent} - Variance_{split}$$

Choose the split that maximizes variance reduction.

Tree Pruning in Decision Trees

Tree pruning is a technique used in decision trees to **reduce overfitting** and improve **generalization** on unseen data. It removes unnecessary nodes from the tree, making the model simpler and more interpretable.

Why is Pruning Important?

✅ Prevents **overfitting**
✅ Reduces **tree complexity**
✅ Improves **accuracy on test data**
✅ Speeds up **prediction time**

2. Types of Tree Pruning

(A) Pre-Pruning (Early Stopping)

- Stops tree growth **before** it becomes too complex.
- Uses criteria like:
  - Minimum number of samples in a node
  - Maximum depth of the tree
  - Minimum improvement in split quality
- **Advantage**: Faster training
- **Disadvantage**: Might **stop too early**, leading to underfitting.

(B) Post-Pruning (Prune After Building the Tree)

- First, grow the full tree.
- Then, remove **less significant branches** using validation data.
- Methods:
  - **Cost Complexity Pruning (CCP)**
  - **Reduced Error Pruning**
- **Advantage**: More reliable than pre-pruning.
- **Disadvantage**: Requires extra validation data.

---

3. Post-Pruning Methods

(A) Cost Complexity Pruning (CCP)

- Introduced in the **CART algorithm**.
- Adds a **penalty** for tree complexity.
- Uses the **cost complexity function**:

$$R(T)=Remp(T)+\alpha\times|T|$$

where:

- $R_{emp}(T)$ = Error on training data
- α = Complexity parameter
- $|T|$ = Number of leaf nodes

- Smaller subtrees are preferred if they improve generalization.

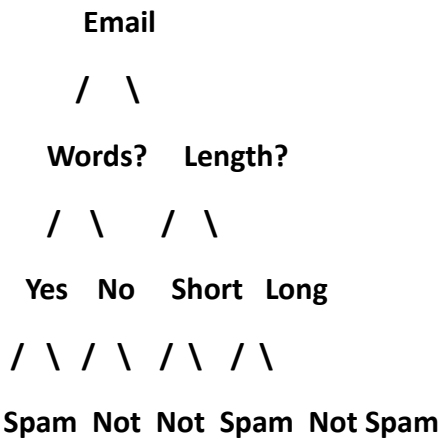
(B) Reduced Error Pruning

- Uses **validation data** to remove nodes.
- Removes a node if it **does not decrease accuracy**.
- **Simpler but less optimal** than CCP.

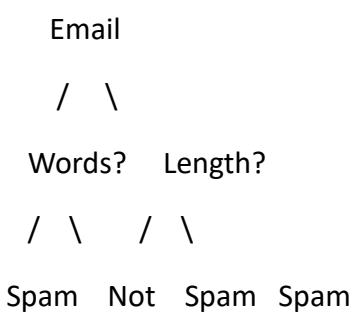
4. Example of Pruning

Let's say we have a **decision tree for classifying emails as spam or not**.

Before Pruning



After Pruning



Difference Between Pre-Pruning and Post-Pruning

Feature	Pre-Pruning (Early Stopping)	Post-Pruning (Prune After Training)
Definition	Stops tree growth early, before it becomes too complex.	Grows the full tree first, then removes unnecessary branches.
When Applied	During the tree-building phase.	After the tree is fully built.
Stopping Criteria	Based on conditions like: ✓ Maximum depth ✓ Minimum samples per node ✓ Minimum information gain	Uses **validation data** or **cost complexity pruning (CCP)** to remove less effective branches.
Risk	**May stop too early** → Can lead to **underfitting**.	**Overfits first, then prunes** → Usually gives better generalization.
Accuracy Impact	Can sometimes produce a tree that is **too simple**.	Generally **better accuracy** on test data.
Computational Cost	Faster, as it stops early.	Slightly slower, as it grows the full tree first.
Example Methods	✓ Depth limit ✓ Minimum leaf size ✓ Minimum gain threshold	✓ Cost Complexity Pruning (CCP) ✓ Reduced Error Pruning
Used In	Decision Trees, Random Forests.	CART, Decision Trees, Ensemble Models.

Naive Bayes Classifiers

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

To start with, let us consider a dataset.

Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit("Yes") or unfit("No") for plaing golf.

Here is a tabular representation of our dataset.

	Outlook	Temperature	Humidity	Windy	Play Golf
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes

	Outlook	Temperature	Humidity	Windy	Play Golf
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

The dataset is divided into two parts, namely, **feature matrix** and the **response vector**.

- Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of **dependent features**. In above dataset, features are 'Outlook', 'Temperature', 'Humidity' and 'Windy'.
- Response vector contains the value of **class variable**(prediction or output) for each row of feature matrix. In above dataset, the class variable name is 'Play golf'.

Assumption:

The fundamental Naive Bayes assumption is that each feature makes an:

- independent
- equal

contribution to the outcome.

With relation to our dataset, this concept can be understood as:

- We assume that no pair of features are dependent. For example, the temperature being 'Hot' has nothing to do with the humidity or the outlook being 'Rainy' has no effect on the winds. Hence, the features are assumed to be **independent**.
- Secondly, each feature is given the same weight(or importance). For example, knowing only temperature and humidity alone can't predict the outcome accuratey. None of the attributes is irrelevant and assumed to be contributing **equally** to the outcome.

Note: The assumptions made by Naive Bayes are not generally correct in real-world situations. In-fact, the independence assumption is never correct but often works well in practice.

Naive - Bayes Theorem

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

Day	Outlook	Temp	Humidity	Wind	Play Cricket?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	mild	High	Weak	yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	mild	Normal	Weak	yes
11	Sunny	mild	Normal	Strong	Yes
12	Overcast	mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	mild	High	Strong	No

Q→ Find the probability to play cricket on 15th day where conditions are —

Temp = cool Humidity = High

Wind = strong Outlook = Sunny

Step-1

Total(14)

Yes(9) NO(5)

$P(Yes) = \frac{9}{14}$ $P(NO) = \frac{5}{14}$ expected % out come / Total no. of outcome

Step-2

Wind(14)

Weak(8) Strong(6)

Yes(6) NO(2) Yes(3) NO(3)

$P(Weak/Yes) = \frac{6}{9}$ $P(Strong/Yes) = \frac{3}{9}$

$P(Weak/NO) = \frac{2}{5}$ $P(Strong/NO) = \frac{3}{5}$

Step-3

Humidity(14)

High(7) Normal(7)

Yes(3) NO(4) Yes(6) No(1)

$P(High/Yes) = \frac{3}{9}$ $P(Normal/Yes) = \frac{6}{9}$

$P(High/NO) = 4/5$ $P(Normal/No) = 1/5$

Step 4

Temp(14)

Hot(4) Mild(6) Cold(4)

Yes(2) No(2) Yes(4) No(2) Yes(3) No(1)

$P(Hot/Yes) = \frac{2}{9}$ $P(Mild/Yes) = \frac{4}{9}$ $P(Cool/Yes) = \frac{3}{9}$

$P(Hot/No) = \frac{2}{5}$ $P(Mild/No) = \frac{2}{5}$ $P(Cool/No) = \frac{1}{5}$

Step-5

Outlook(14)

Sunny(5) Overcast(4) Rain(5)

Yes(2) No(3) Yes(4) No(0) Yes(3) No(2)

$P(Sunny/Yes) = \frac{2}{9}$ $P(Overcast/Yes) = \frac{4}{9}$ $P(rain/Yes) = \frac{3}{9}$

$P(Sunny/No) = \frac{3}{5}$ $P(overcast/No) = \frac{0}{5}$ $P(rain/No) = \frac{2}{5}$

Let X = { Sunny, Cool, High, Strong }

$P(X/Yes) = P(Yes) * P(Sunny/Yes) * P(Cool/Yes) * P(High/Yes) * P(Strong/Yes)$

$= \frac{9}{14} * \frac{2}{9} * \frac{3}{9} * \frac{3}{9} * \frac{3}{9}$

$= 0.00529$

$P(X/No) = P(No) * P(Sunny/No) * P(Cool/No) * P(High/No) * P(Strong/No)$

$= \frac{5}{14} * \frac{3}{5} * \frac{1}{5} * \frac{4}{5} * \frac{3}{5}$

$= \frac{18}{7 \times 125} = 0.02057$

Probability of No is High

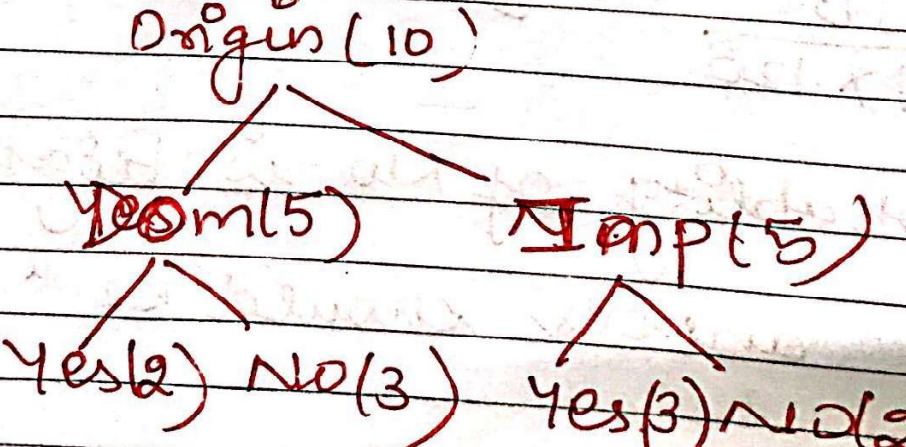
So answer should be "No".

Q →

Car No.	Color	Type	Origin	Stolen
1	Red	Sports	Dom	Yes
2	Red	Sports	Dom	No
3	Red	Sports	Dom	Yes
4	Yellow	Sports	Dom	No
5	Yellow	Sports	Imp	Yes
6	Yellow	SUV	Imp	No
7	Yellow	SUV	Imp	Yes
8	Yellow	SUV	Dom	No
9	Red	SUV	Imp	No
10	Red	Sports	Imp	Yes

$P(Yes) = 5/10$

$P(No) = 5/10$

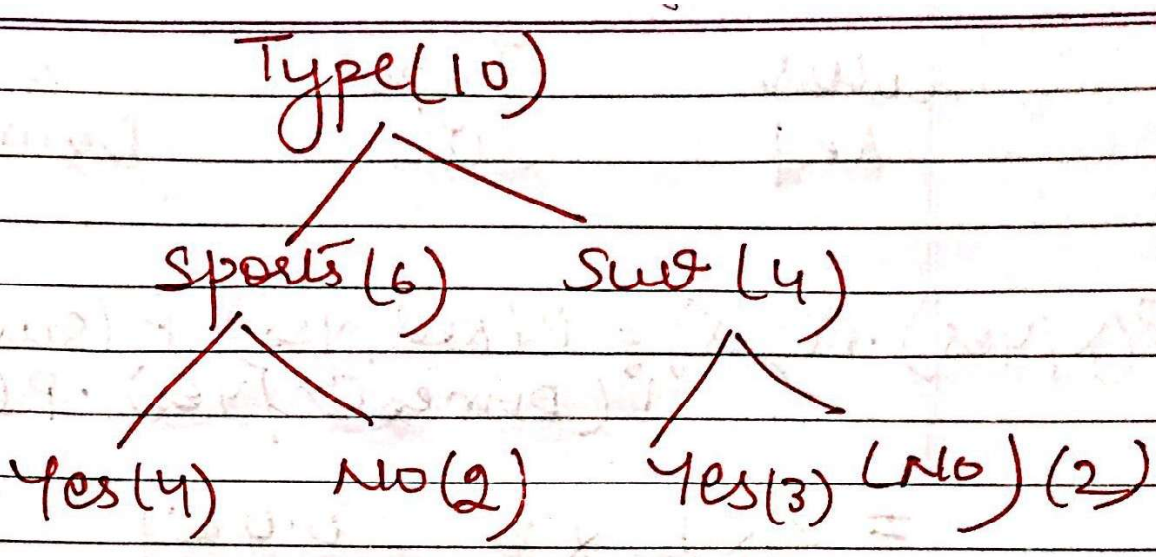


$P(Dom/Yes) = \frac{2}{5}$

$P(Dom/No) = \frac{3}{5}$

$P(Imp/Yes) = \frac{3}{5}$

$P(Imp/No) = \frac{2}{5}$

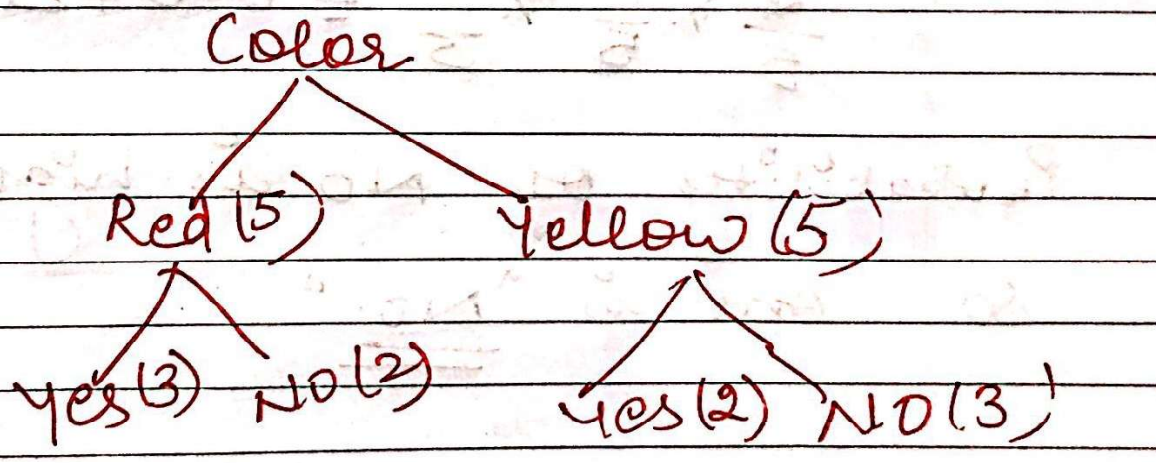


$P(\text{Sports}|\text{Yes}) = \frac{4}{5}$

$P(\text{Sports}|\text{NO}) = \frac{2}{5}$

$P(\text{SUV}|\text{Yes}) = \frac{3}{5}$

$P(\text{SUV}|\text{NO}) = \frac{2}{5}$



$P(\text{Red}|\text{Yes}) = \frac{3}{5}$

$P(\text{Red}|\text{NO}) = \frac{2}{5}$

$P(\text{Yellow}|\text{Yes}) = \frac{2}{5}$

$P(\text{Yellow}|\text{NO}) = \frac{3}{5}$

Color	Type	Origin
Red	SUV	Domestic

$P(X/Yes) \cdot P(Yes) = P(Red/Yes)\ P(SUV/Yes)\ P(Domestic/Yes) \cdot P(Yes)$

$= \frac{3}{5} \times \frac{1}{5} \times \frac{2}{5} = 0.024$

$P(X/No) \cdot P(No) = P(Red/No)\ P(SUV/No)\ P(Dom/No)\ P(No)$

$= \frac{2}{5} \times \frac{3}{5} \times \frac{3}{5} = 0.072$

Probability of No is high.

So Ans is "No."

Classification by Backpropagation

Classification by backpropagation refers to using the backpropagation algorithm to train a neural network to perform classification tasks.

Backpropagation is a supervised learning algorithm used for training artificial neural networks. It works by:

1. Forward pass: Input data is passed through the network to get predictions.
2. Loss computation: The difference between the prediction and the true label is calculated using a loss function.
3. Backward pass: The error is propagated backward through the network using calculus (the chain rule) to compute the gradients of the loss with respect to each weight.
4. Weight update: The weights are updated (usually with gradient descent) to reduce the error.

Using Backpropagation for Classification

In classification problems, the goal is to categorize input data into predefined classes. A typical setup:

- Input layer: Features (e.g., pixels of an image)
- Hidden layers: Learn abstract representations
- Output layer: One neuron per class (often with softmax activation)
- Loss function: Usually cross-entropy loss

Steps in Classification by Backpropagation

1. Prepare the dataset (e.g., labeled images)
2. Build a neural network
    - Input layer size = number of features
    - Hidden layers with activation functions like ReLU
    - Output layer with softmax (for multiclass classification)
3. Train the network using backpropagation and gradient descent
4. Evaluate the model on validation/test data.

Support Vector Machine

A Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks, though it's most commonly used for classification.

SVM aims to find the best separating hyperplane that divides the data into different classes with the maximum margin.

- Hyperplane: A decision boundary (line in 2D, plane in 3D, etc.)
- Margin: The distance between the hyperplane and the closest data points from each class
- Support Vectors: The data points that lie closest to the hyperplane and "support" it—removing them would change the decision boundary

Architecture of SVM

SVM doesn't have an architecture like neural networks (with layers), but we can conceptually break it down into three main components:

1. Input Space

- Data points represented as feature vectors $x \in R^n$
- Each data point is labeled with a class $y \in \{-1,+1\}$ for binary classification

2. Decision Function

- SVM tries to learn a decision function:

$$f(x)=\text{sign}(w^T x+b)$$

  - w: weight vector (normal to the hyperplane)
  - b: bias (offset)
  - The sign function determines the predicted class

3. Optimization Engine

- Uses Quadratic Programming to find the optimal hyperplane that maximizes the margin between classes while minimizing classification errors.

If the data is non-linearly separable, it uses:

- Kernel Trick: To map input into higher-dimensional feature space
- Slack Variables: To allow some misclassifications (soft margin)

Summary Table

Component	**Description**
Input	Feature vectors xi with labels $y_i \in \{-1,1\}$
Objective	Maximize margin between classes

Component	Description
Decision Boundary	$wTx+b=0$
Key Data Points	Support vectors
Optimization	Quadratic Programming
Kernel Trick	Implicitly map to higher-dimensional space
Output	Class label predicted by sign of decision function

SVM-Lazy Learner

In machine learning, algorithms are often categorized as:

1. Eager Learners

- Build a model during training.
- Generalize from the training data before seeing test data.
- Examples: SVM, Decision Trees, Neural Networks.

2. Lazy Learners

- Do not build a model during training.
- Store the training data and generalize only when queried (i.e., during prediction).
- Examples: k-Nearest Neighbors (k-NN), Case-Based Reasoning.

SVM is an *eager learner*.

- During training, SVM goes through an optimization process (solving a convex problem) to find the optimal hyperplane that separates the classes.
- It does not simply store the training data and wait for a query to make a prediction.
- The result is a learned model (support vectors, weights w, and bias b) that is used to classify new instances efficiently.

Comparison: SVM vs k-NN

Aspect	SVM (Eager Learner)	k-NN (Lazy Learner)
Model building	During training	During prediction
Training speed	Slower (optimization involved)	Very fast (just stores data)
Prediction speed	Fast (simple calculation)	Slow (distance computation with all training data)

Aspect	SVM (Eager Learner)	k-NN (Lazy Learner)
Memory usage	Stores only support vectors	Stores entire training set
Generalization	Learns a decision boundary	No generalization until test time

KNN-Metrics for evaluating classifier performance

1. Confusion Matrix

A confusion matrix summarizes the performance of a classifier by showing correct and incorrect predictions across all classes.

For binary classification:

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

2. Common Metrics Derived from the Confusion Matrix

Accuracy

Proportion of total correct predictions:

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

Easy to understand, but can be misleading with imbalanced classes.

Precision

- How many of the predicted positives are actually positive?

$$\text{Precision}=TP/TP+FP$$

- High precision = low false positive rate

Recall (Sensitivity, True Positive Rate)

- How many actual positives were correctly predicted?

$$\text{Recall}=TP/TP+FN$$

- High recall = low false negative rate

F1 Score

- Harmonic mean of precision and recall:

$$\text{F1 Score}=2\cdot\text{Precision}\cdot\text{Recall}/\text{Precision}+\text{Recall}\backslash$$

- Useful when you need a balance between precision and recall.

3. Receiver Operating Characteristic (ROC) Curve

- Plots True Positive Rate vs False Positive Rate at different thresholds.
- AUC (Area Under the Curve) gives a single score:
    - AUC = 1: perfect classifier
    - AUC = 0.5: random guessing

Note: KNN doesn't output probability scores by default, but you can get class probabilities by taking the proportion of neighbors belonging to each class.

Techniques to improve classification accuracy

Improving classification accuracy involves enhancing the model, tuning parameters, and improving data quality. Here are some effective techniques to boost classification performance, especially for models like KNN, SVM, decision trees, or neural networks.

1. Data Preprocessing and Feature Engineering

Normalize/Scale Data

- Especially important for **KNN** and **SVM**, which are distance-based.
- Use **Min-Max scaling** or **Standardization (Z-score)**.

Feature Selection

- Remove irrelevant/redundant features to reduce noise and overfitting.
- Techniques: **Recursive Feature Elimination (RFE)**, **Chi-Squared test**, **Lasso**.

Feature Extraction

- Use domain knowledge or algorithms (e.g., **PCA**, **t-SNE**) to create more meaningful features.

2. Model Selection and Optimization

Hyperparameter Tuning

Use **Grid Search** or **Random Search** with cross-validation:

- **KNN**: Choose optimal kkk, distance metric (Euclidean, Manhattan)
- **SVM**: Tune **C**, **gamma**, and **kernel** type
- **Decision Trees**: Adjust depth, split criteria, pruning

3. Use Ensemble Methods

Bagging

- Combine predictions from multiple models trained on different data subsets
- Example: **Random Forest**

Boosting

- Sequentially correct errors of previous models
- Examples: **AdaBoost**, **XGBoost**, **Gradient Boosting**

Stacking

- Combine multiple base learners and feed their outputs into a meta-model

4. Data Augmentation / Enrichment

Add More Data

- More data can reduce overfitting and improve generalization.

Data Augmentation

- Especially useful in **image** and **text** tasks.
    - Image: Rotation, flipping, scaling
    - Text: Synonym replacement, back translation

5. Cross-Validation

- Use **k-fold cross-validation** to get a better estimate of model performance and prevent overfitting.

6. Handle Class Imbalance

If your classes are imbalanced (e.g., 95% of class A, 5% class B):

Techniques:

- **Resampling**: Oversample minority or undersample majority class
- **Synthetic Data**: Use SMOTE (Synthetic Minority Over-sampling Technique)
- **Class Weighting**: Adjust cost of misclassification

7. Use a Different Model

Sometimes, the model itself might not be powerful enough:

- Try more complex models (e.g., move from KNN to SVM or neural nets)
- Use **deep learning** for high-dimensional data like images or text

8. Noise Reduction / Error Correction

- **Clean the data**: Remove or correct mislabelled or noisy data points.
- Use **outlier detection** to clean extreme values before training.

Prediction: Regression Analysis

Regression analysis is a supervised learning technique used to predict a continuous (numeric) value based on the relationship between independent variables (features) and a dependent variable (target).

In classification, we predict categories (like "spam" or "not spam").

In regression, we predict real-valued outputs, such as:

- House price
- Stock price
- Temperature
- Age

Common Types of Regression

Type	Use Case Example
Linear Regression	Predicting house prices
Polynomial Regression	Modeling curved relationships
Ridge/Lasso Regression	Regularized regression to prevent overfitting
Logistic Regression	(Though named regression, it’s for classification)
SVR (Support Vector Regression)	Non-linear regression using SVM concepts
Decision Tree/Random Forest Regression	Complex, non-linear data patterns