MODULE -04

Learning– Forms of Learning, Supervised Learning, Machine Learning - Decision Trees, Regression and Classification with Linear Models, Artificial Neural Networks, Support Vector Machines

4. LEARNING IN AI

- In Artificial Intelligence (AI), "learning" refers to the process by which a system (often a machine or computer program) improves its performance over time by gaining experience or data.
- Learning in AI is the ability of a machine to improve its performance on a task through experience (data), without being explicitly programmed for every scenario.
- Learning refers to the process by which a system acquires knowledge or improves its performance based on experience.
- The idea behind learning is that precepts should be used not only for acting, but also for improving the agent's ability to act in the future. Learning takes place as the agent observes its interactions with the world and its own decision-making processes.
- Learning can range from trivial memorization of experience.
- Learning refers to the process by which a system acquires knowledge or improves its performance based on experience. Machine learning researchers have come up with a large variety of learning elements.

Learning in AI means teaching a system (or agent) to improve its performance by gaining knowledge through experience. Instead of programming every action manually, the system learns from past actions and feedback to do better in the future.

Why is Learning Important?

- It helps the agent act better over time.
- The agent learns from what it sees and does in the world.
- Learning can be as simple as remembering things or as complex as making better decisions.

Types of Learning Elements

Machine learning researchers have created many different parts (elements) that help an agent learn. To design a good learning system, we need to think about:

1. What should be learned?
   → Which parts of the agent's brain or system do we want to improve?
2. How does the agent get feedback?
   → What information does the agent receive to know if it's doing well or not?
3. How is the information shown or stored?
   → How is learning material (data, knowledge) represented?

Components of a Learning Agent

These are the parts an agent might learn:

1. **Rules to take actions** based on the current situation.
   → Like: "If it's raining, take an umbrella."
2. **Ways to understand the world** from what it senses (perceives).
   → Understanding the situation from inputs.
3. **Info about changes in the world** and what might happen after actions.
   → Predicting future results.
4. **How good or bad a situation is** (utility).
   → Like scoring different situations.
5. **How good or bad an action is.**
   → Choosing the best move.
6. **Goals to achieve.**
   → What the agent is trying to do (like winning a game or solving a task).

Characteristics of Learning:

1. **Learning is based on experience**
    1. The system improves its performance by learning from **past actions or data**.
    2. Example: A robot gets better at walking by practicing over time.
2. **Learning improves performance**
    1. The goal of learning is to make the system **perform better** at a specific task.
    2. Example: A self-driving car learns to drive more safely.
3. **Learning is adaptive**
    1. A good learning system can **adjust to changes** in the environment.
    2. Example: A chatbot can adapt its replies based on the user's tone.
4. **Learning involves generalization**
    1. The system should not only remember past data but also **apply it to new situations**.
    2. Example: After seeing a few types of chairs, the system should recognize a new chair.
5. **Learning can be supervised or unsupervised**
    1. **Supervised**: Learns from labeled data (with answers).
    2. **Unsupervised**: Finds patterns in data without labels.
6. **Learning requires feedback**
    1. Feedback (like success, failure, rewards, or corrections) helps the system **improve over time**.
7. **Learning reduces the need for programming**
    1. Instead of writing code for every situation, the agent **learns on its own**.

Advantages of Learning

1. **Improves Performance Over Time**
    1. The system gets better with more experience.

    2. Example: A speech recognition app improves with more usage.

2. **Reduces Manual Programming**
    1. No need to hard-code every rule or behavior.
    2. The system learns behaviors automatically.
3. **Adapts to New Situations**
    1. Learning enables the system to handle changes in the environment.
    2. Example: A spam filter updates itself as new spam types appear.
4. **Discovers Hidden Patterns**
    1. It can find patterns in large data that humans may miss.
    2. Useful in data analysis and predictions.
5. **Enables Personalization**
    1. Systems can learn user preferences and personalize experiences.
    2. Example: Netflix recommendations.

Disadvantages of Learning

1. **Requires a Lot of Data**
    1. Learning systems often need **huge amounts of data** to perform well.
    2. Without enough data, learning is inaccurate.
2. **High Computational Cost**
    1. Training models can be **time-consuming and resource-heavy**.
3. **Possibility of Overfitting**
    1. The system may **memorize** instead of learning to generalize, leading to poor performance on new data.
4. **Quality of Learning Depends on Data Quality**
    1. **Bad or biased data** can lead to wrong learning outcomes.
5. **Unpredictable Behavior**
    1. In some cases, a learning system might make **unexpected or incorrect decisions**.

4.1Forms of Learning

Definition: Forms of Learning refer to the different ways individuals acquire knowledge, skills, behaviors, or attitudes through experience, practice, or observation

- An agent is learning if it improves its performance on future tasks after making observations about the world.
- Any component of an agent can be improved by learning from data. It depends upon 4 factors:
- Which component is to be improved

- direct mapping from conditions on the current state to actions
- infer relevant properties of the world
- results of possible actions
- Action-value information
- Goals that describe classes of states

➢ What prior knowledge the agent already has.

➢ What representation is used for the data and the component.

- representations: propositional and first-order logical sentences
- Bayesian networks for the inferential components
- factored representation—a vector of attribute values—and outputs that can be either a
- continuous numerical value or a discrete value

➢ What feedback is available to learn from : types of feedback that determine the three main types of learning

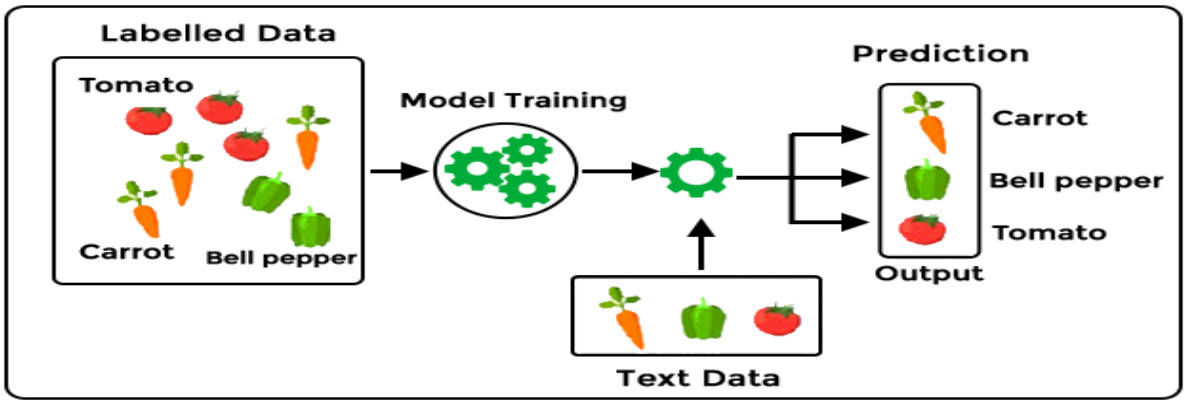
- In unsupervised learning the agent learns patterns in the input even though no explicit
- feedback is supplied
- reinforcement learning the agent learns from a series of reinforcements—rewards or
- punishments.
- supervised learning the agent observes some example input–output pairs and learns a function that maps from input to output

Form of Learning	Example	Goal	Type of Label Used
Supervised Learning	Email spam detection	Predict a label (spam or not spam)	Labeled data (input-output pairs)
Unsupervised Learning	Customer segmentation	Group similar items without predefined labels	No labels
Semi-Supervised Learning	Document classification with few labeled documents	Improve accuracy using few labeled + many unlabeled examples	Few labeled + many unlabeled

Reinforcement Learning	Game-playing AI (e.g., AlphaGo)	Learn actions to maximize reward over time	Reward signal (not direct labels)
Self-Supervised Learning	Predict next word in a sentence (e.g., GPT pre-training)	Learn useful representations from raw data	Labels generated from data itself
Multi-Instance Learning	Medical diagnosis based on multiple scans	Classify a bag of instances (e.g., all scans per patient)	Label on group, not individual instances
Online Learning	Stock market prediction	Continuously learn from streaming data	Labeled data in sequence

1.Supervised Learning:

Supervised learning, also known as **supervised machine learning**, is a type of machine learning that trains the model using **labeled datasets** to predict outcomes. A Labeled dataset is one that consists of input data (features) along with corresponding output data (targets).



✅ 1. Labeled Data

- **Definition:** Training data includes both inputs and correct outputs (labels).
- **Example:** An image labeled as “cat” or “dog.”

🎯 2. Goal-Oriented Learning

- **Purpose:** Learn a mapping function from inputs to outputs to make accurate predictions.
- **Use Cases:** Spam detection, image classification, sales forecasting.

🔄 3. Training Process

- The model is **trained using known data** so it can generalize to new, unseen data.
- The algorithm iteratively adjusts to minimize error between predicted and actual outputs.

📊 **4. Evaluation is Straightforward**

- Since you have labels, you can directly compare predictions to actual outcomes using metrics like accuracy, precision, recall, or MSE.

5. **Types of Problems**

- **Classification:** Output is categorical (e.g., yes/no, red/blue/green)
- **Regression:** Output is continuous (e.g., house prices, temperature)

🧪 **6. Requires Data Splitting**

- Typically split into:
  - **Training set** – to train the model
  - **Test/Validation set** – to evaluate generalization

Types of Supervised Learning Algorithm

1**. Classification:**The task is to predict a discrete label or category.

Categorical (discrete values).

Ex: Disease diagnosis (e.g., cancer vs. no cancer)

2. Regression: The task is to predict a continuous output.

Continuous (numerical values).

Ex: Predicting temperature.

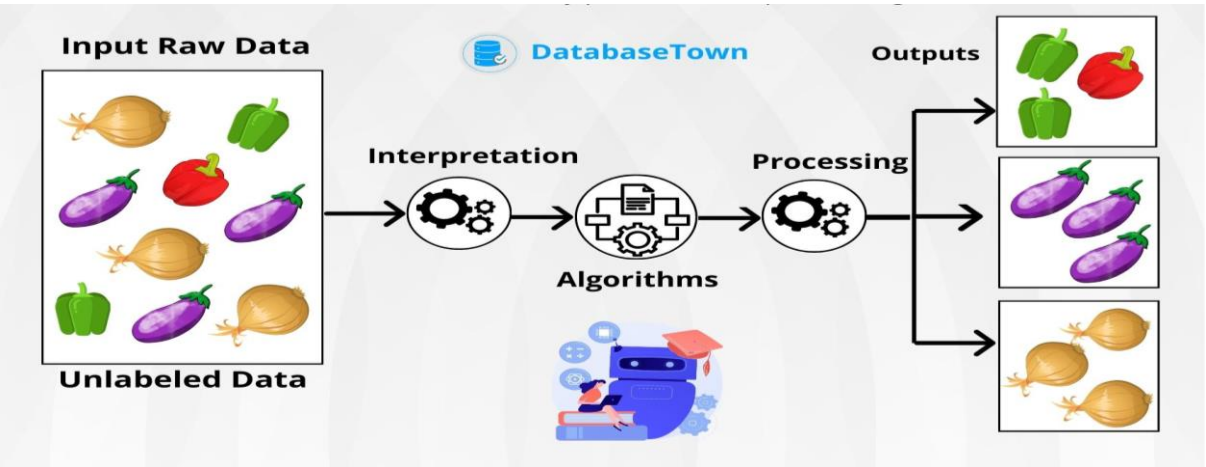
Type	Output	Examples	Algorithms
Classification	Categorical (discrete)	Email spam detection, Handwritten digit recognition	Logistic Regression, Decision Trees, SVM, Random Forest
Regression	Continuous (numerical)	House price prediction, Stock price forecasting	Linear Regression, Ridge Regression, SVR, Neural Networks

Classification	Regression

• **Logistic Regression** • **Decision Trees** • **Random Forest** • **Support Vector Machines (SVM)** • **K-Nearest Neighbors (KNN)** • **Naive Bayes** • **Neural Networks (for complex tasks)**	• Linear Regression • Polynomial Regression • Ridge/Lasso Regression • Support Vector Regression (SVR) • Decision Trees (Regression Trees) • Neural Networks (for more complex tasks)

2.Unsupervised learning: **I**s a type of machine learning where the algorithm is given data without explicit labels (i.e., no "correct answers"). The goal is to find hidden patterns or structures in the data.

Unlike supervised learning, where we have labeled data to guide the learning process, unsupervised learning algorithms must infer the structure of the data on their own.



3.Reinforcement Learning:

Reinforcement Learning (RL) is a key area of Artificial Intelligence (AI) where an *agent* learns how to make decisions by interacting with an *environment*.

Instead of being told exactly what to do (like in supervised learning), the agent **tries different actions** and learns based on **rewards or punishments** it gets.

Example: **Robots** learning to walk or pick up objects (rewarded when they succeed).

 Self-driving cars learning to navigate without crashing

4. Semi-Supervised Learning:

Semi-Supervised Learning (SSL) is a mix between Supervised Learning and Unsupervised Learning.

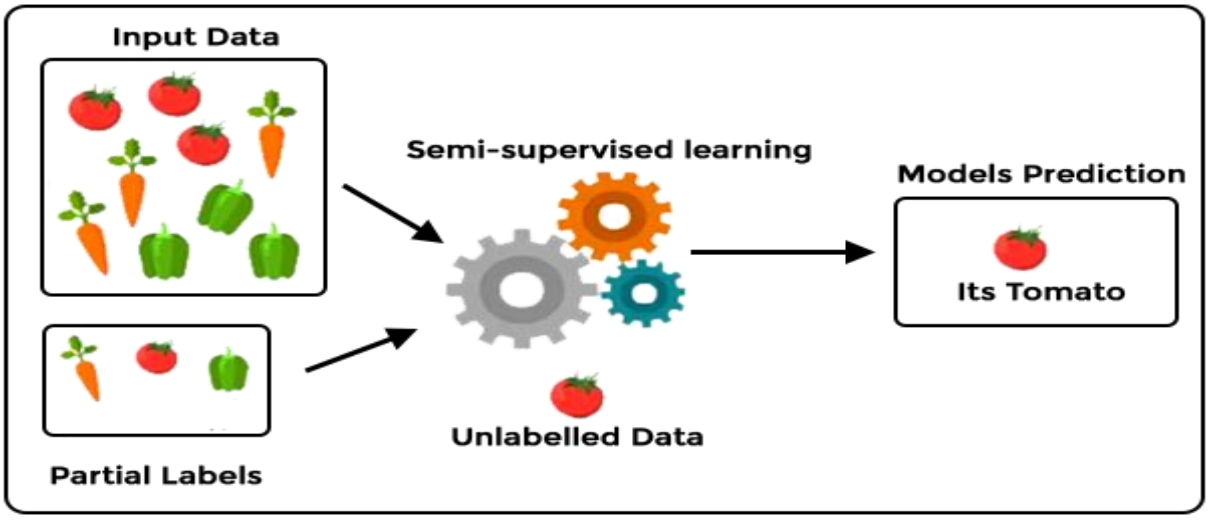
- In Supervised Learning, *every data point* has a label (like a photo labeled "dog" or "cat").
- In Unsupervised Learning, *no data* has labels — the system just tries to find patterns.

👉 Semi-Supervised Learning happens when only a small part of the data is labeled, and the rest is unlabeled.

Example: You have 1000 photos.

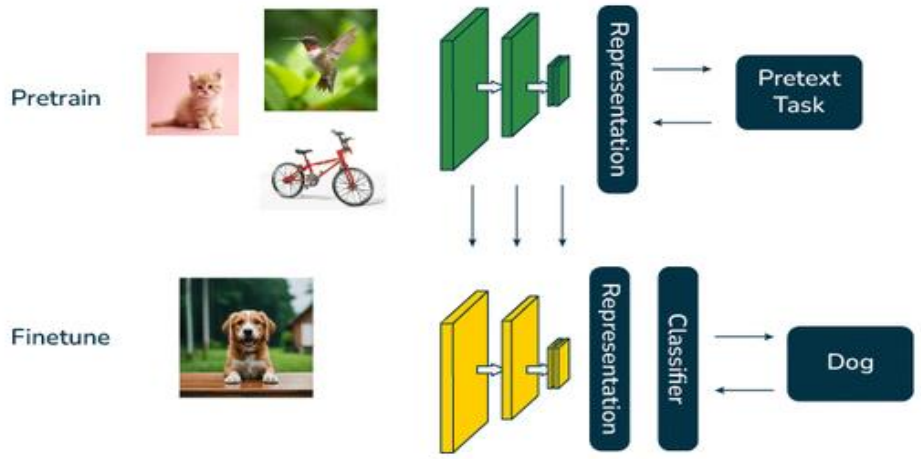
only 50 are labeled ("dog", "cat", "car"), and the other 950 are unlabeled.

Instead of labeling all 1000 manually (which is expensive), SSL lets you train a good model using just those 50 labels + the structure in the other 950.



5. **Self-supervised learning** :Self-supervised learning is a deep learning methodology where a **model is pre-trained using unlabelled data** and the data labels are generated automatically, which are further used in subsequent iterations as ground truths.

Example: Predicting Missing Words in Sentences



Supervised Learning:

As its name suggests, Supervised machine learning is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. More preciously, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset.

Let's understand supervised learning with an example. Suppose we have an input dataset of cats and dog images. So, first, we will provide the training to the machine to understand the images, such as the **shape & size of the tail of cat and dog, Shape of eyes, colour, height (dogs are taller, cats are smaller), etc.** After completion of training, we input the picture of a cat and ask the machine to identify the object and predict the output. Now, the machine is well trained, so it will check all the features of the object, such as height, shape, colour, eyes, ears, tail, etc., and find that it's a cat. So, it will put it in the Cat category. This is the process of how the machine identifies the objects in Supervised Learning.

The main goal of the supervised learning technique is to map the input variable(x) with the output variable(y). Some real-world applications of supervised learning are **Risk Assessment, Fraud Detection, Spam filtering,** etc.

Categories of Supervised Machine Learning

Supervised machine learning can be classified into two types of problems, which are given below:

1. Classification
2. **Regression**

1. **Classification:** Classification algorithms are used to solve the classification problems in which the output variable is categorical, such as **"Yes" or No, Male or Female, Red or Blue, etc**. The classification algorithms predict the categories present in the dataset. Some real-world examples of classification algorithms are **Spam Detection, Email filtering, etc.**

Some popular classification algorithms are given below:

- **Random Forest Algorithm**
- **Decision Tree Algorithm**
- **Logistic Regression Algorithm**
- **Support Vector Machine Algorithm**

2.Regression

Regression algorithms are used to solve regression problems in which there is a linear relationship between input and output variables.

These are used to predict continuous output variables, such as market trends, weather prediction, etc.

Some popular Regression algorithms are given below:

- **Simple Linear Regression Algorithm**
- **Multivariate Regression Algorithm**
- **Decision Tree Algorithm**
- **Lasso Regression**

Advantages and Disadvantages of Supervised Learning

Advantages:

Since supervised learning work with the labelled dataset so we can have an exact idea about the classes of objects.

- These algorithms are helpful in predicting the output on the basis of prior experience.

Disadvantages:

- These algorithms are not able to solve complex tasks.
- It may predict the wrong output if the test data is different from the training data.

It requires lots of computational time to train the algorithm

Applications of Supervised Learning

Some common applications of Supervised Learning are given below:

1. **Image Segmentation:** Supervised Learning algorithms are used in image segmentation. In this process, image classification is performed on different image data with pre-defined labels.

2. **Medical Diagnosis:** Supervised algorithms are also used in the medical field for diagnosis purposes. It is done by using medical images and past labelled data with labels for disease conditions. With such a process, the machine can identify a disease for the new patients.
3. **Fraud Detection** - Supervised Learning classification algorithms are used for identifying fraud transactions, fraud customers, etc. It is done by using historic data to identify the patterns that can lead to possible fraud.
4. **Spam detection** - In spam detection & filtering, classification algorithms are used. These algorithms classify an email as spam or not spam. The spam emails are sent to the spam folder.
5. **Speech Recognition** - Supervised learning algorithms are also used in speech recognition. The algorithm is trained with voice data, and various identifications can be done using the same, such as voice-activated passwords, voice commands, etc.

Steps Involved in Supervised Learning:

- First Determine the type of training dataset
- Collect/Gather the labelled training data.
- Split the training dataset into training **dataset, test dataset, and validation dataset**.
- Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

Example Program: In AI, a linear regression model is a statistical method used to model the relationship between a dependent variable (often called the "target" or "output") and one or more independent variables (also called "predictors" or "features").

simple linear regression (one predictor), the model can be written as:

$Y=\beta 0+\beta 1X+\epsilon$

Algorithm:

Step 1: **Gather the Data**: Collect data with features (independent variables) and a target

(dependent variable).

Step 2: Define the Model: Represent the relationship as $y=\beta0+\beta1x1+\cdots+\beta nXn+\epsilon$

Step 3:Calculate Coefficients: Use the least squares method to find the coefficients (β) that minimize prediction error.

Step 4:Make Predictions: Compute predicted values using the formula $y^=\beta0+\beta1x1+\ldots$

Step 5:Evaluate the Model: Assess model performance using metrics like R-squared,

Step 6:Adjust the Model: If necessary, apply regularization to improve accuracy.

Step 7: Make Predictions: Use the trained model to predict outcomes on new data.

Source code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
# Data
data = {
    'Hours_Studies': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Exam_Score': [10, 20, 30, 40, 50, 60, 65, 80, 85, 95]
}
# Create DataFrame
df = pd.DataFrame(data)
# Features and Labels
x = df[['Hours_Studies']]
y = df[['Exam_Score']]
# Split the dataset
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
# Train the model
Model = LinearRegression()
Model.fit(x_train, y_train)
# Predict
Predicted_Score = Model.predict([[7.5]])
print(f"Predicted Exam Score for 7.5 hours of Study: {Predicted_Score[0][0]:.2f}")
# Plot
plt.scatter(x, y, color='blue', label="Actual Data")
plt.plot(x, Model.predict(x), color='red', label='Regression Line')
plt.xlabel("Hours Studied")
plt.ylabel("Exam Score")
plt.title("Linear Regression: Hours of Study vs. Exam Score")
plt.legend()
plt.show()
```

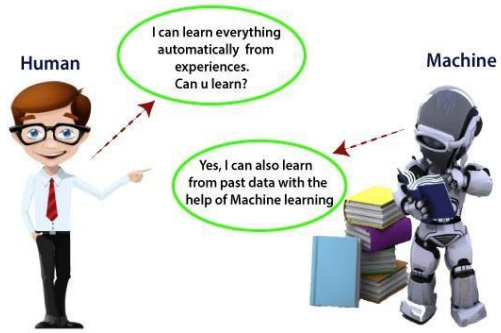
Output: Predicted Exam Score for 7.5 hours of Study: 71.32

4.2 Machine Learning: A subset of artificial intelligence known as machine learning focuses primarily on the creation of algorithms that enable a computer to independently learn

from data and previous experiences.

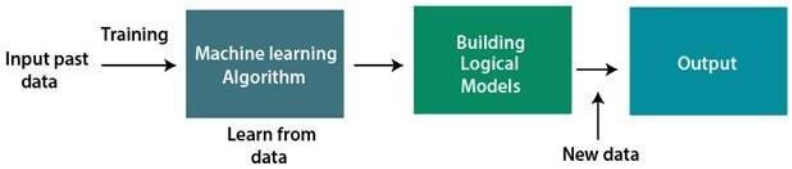
Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed.

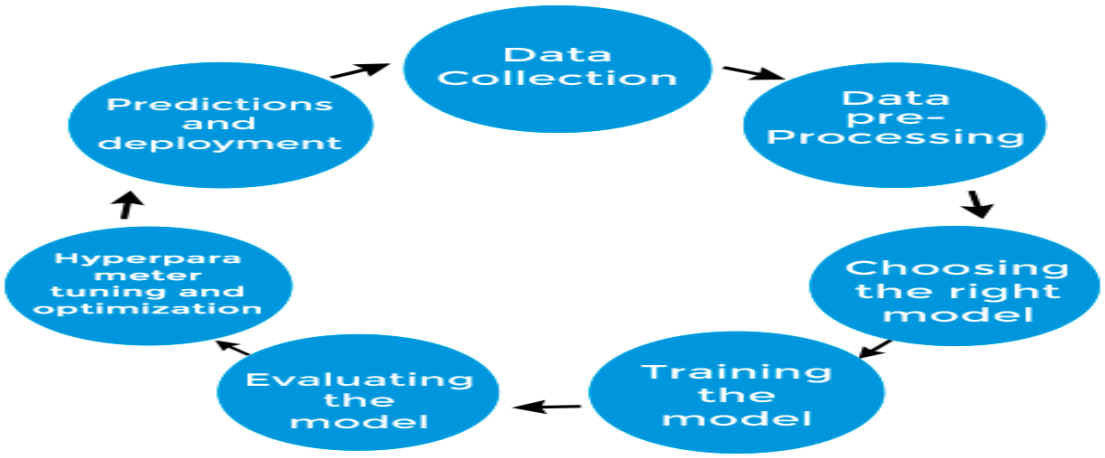
Machine learning algorithms create a mathematical model that, without being explicitly programmed, aids in making predictions or decisions with the assistance of sample historical data, or training data.



How does Machine Learning work

- A machine learning system builds prediction models, learns from previous data, and predicts the output of new data whenever it receives it. The amount of data helps to build a better model that accurately predicts the output, which in turn affects the accuracy of the predicted output.
- Let's say we have a complex problem in which we need to make predictions. Instead of writing code, we just need to feed the data to generic algorithms, which build the logic based on the data and predict the output. Our perspective on the issue has changed as a result of machine learning. The Machine Learning algorithm's operation is depicted in the following block diagram:





Features of Machine Learning:

- Machine learning uses data to detect various patterns in a given dataset.
- It can learn from past data and improve automatically.
- It is a data-driven technology.
- Machine learning is much similar to data mining as it also deals with the huge amount of the data.

Applications of Machine learning:

1. Healthcare

- Disease Prediction and Diagnosis: ML models help predict diseases like cancer, diabetes, and heart problems early by analyzing patient data.
- Medical Imaging: ML is used to read X-rays, MRIs, and CT scans to detect abnormalities.
- Drug Discovery: Speeds up finding new drugs by predicting how different chemicals will react.

---

2. Finance

- Fraud Detection: ML algorithms spot unusual patterns in transactions to catch fraud.
- Algorithmic Trading: ML helps in predicting stock prices and automating buying/selling decisions.
- Credit Scoring: Banks use ML to decide if a person is eligible for a loan.

---

3. Retail and E-commerce

- Recommendation Systems: Platforms like Amazon and Netflix suggest products/movies based on your past behaviour.
- Customer Segmentation: ML helps businesses group customers based on buying habits to target ads better.
- Inventory Management: Predicts which products will sell more and when.

---

4. Transportation

- Self-Driving Cars: ML powers autonomous vehicles to detect obstacles, read signs, and make decisions.
- Route Optimization: Apps like Google Maps and Uber use ML to suggest faster routes.

---

5. Entertainment

- Personalized Content: Spotify, YouTube, and Netflix use ML to recommend music and videos you might like.
- Game AI: Video games use ML to create smarter, more challenging opponents.

---

6. Manufacturing

- Predictive Maintenance: ML predicts when machines will break down so repairs can happen before failures.
- Quality Control: Automatically detects defects in products using computer vision.

---

7. Education

- Personalized Learning: Platforms adjust the difficulty of lessons based on student performance.
- Automated Grading: ML helps grade essays and exams faster and more objectively.

---

8. Agriculture

- Crop Monitoring: ML analyzes satellite images to monitor crop health.
- Yield Prediction: Helps farmers predict how much crop they will produce.

---

9. Security

- Facial Recognition: Used in phones, airports, and buildings for authentication.
- Cybersecurity: ML detects unusual network activity and protects against hacking.

---

10. Natural Language Processing (NLP)

- Chatbots and Virtual Assistants: Siri, Alexa, and customer service bots use ML to understand and respond to human language.
- Language Translation: Tools like Google Translate are powered by ML

Advantages:

Automation of Tasks

- ML can automate repetitive, time-consuming tasks without human intervention once trained (like sorting emails, tagging photos, etc.).

---

2. Improved Decision Making

- ML analyzes huge amounts of data quickly and finds hidden patterns, helping businesses and professionals make better, data-driven decisions.

---

3. Continuous Improvement

- ML systems keep learning from new data over time, making their predictions or actions more accurate as they gain experience.

---

4. Handling Complex Data

- ML can manage and learn from massive, complex, and high-dimensional data that traditional methods would struggle to handle.

---

5. Personalization

- ML helps create personalized experiences — for example, recommending products, songs, or movies based on your behavior.

---

6. Wide Range of Applications

- ML is flexible and can be applied across industries: healthcare, finance, transportation, agriculture, marketing, education, and more.

---

7. Cost Efficiency

- Although training ML models can be expensive initially, once deployed, they can significantly reduce operational costs by automating and optimizing processes.

---

8. Early Detection and Prevention

- In healthcare and cybersecurity, ML can detect issues early (like diseases or security threats), helping prevent bigger problems**.**

Disadvantages:

High Cost of Training

- Building and training ML models can require a lot of time, computational power, and financial investment.

---

2. Data Dependency

- ML models need large amounts of quality data to work properly. If the data is inaccurate, incomplete, or biased, the results will be bad too.

---

3. Lack of Explainability

- Many ML models (especially deep learning models) are like "black boxes" — they make decisions without easily explainable reasons, which can be risky in critical areas like healthcare or law.

---

4. Risk of Bias

- If the training data has biases (like gender or racial bias), the ML model can learn and even amplify these biases in its predictions.

---

5. Overfitting and Underfitting

- Sometimes models are too tuned to the training data (overfitting) or not tuned enough (underfitting), leading to poor performance on real-world data.

Inductive learning in Artificial Intelligence (AI) refers to the process where an AI system learns general patterns or rules from a set of observed examples or experiences (data). The goal of inductive learning is to infer a general rule (hypothesis) that can be applied to new, unseen data. This is in

contrast to **deductive reasoning**, where conclusions are drawn from already known facts or rules.

In **inductive learning**, the system generalizes from specific instances to create a general model or function. The learned model can then be used to make predictions or decisions about new, unseen data.

Real-World Example: Email Spam Filter

Let's consider a **real-world example** of inductive learning: an **email spam filter**. **Problem:**

The goal of the spam filter is to determine whether an incoming email is **spam** or **not spam** (ham). The system needs to learn from past emails to make this decision for future, unseen emails.

Training Data:

The training data consists of a set of labeled examples (emails) that have been classified as spam or not spam by human users. Each email in the training set has a set of **features** (input), and the corresponding label (output) indicating whether the email is spam or not.

Email Content	**Features (Keywords)**	**Label (Spam or Ham)**
"Buy now! Huge discount on laptops"	Buy, Discount	Spam
"Your bank account statement"	Bank, Account	Ham
"Limited time offer for insurance"	Offer, Insurance	Spam
"Meeting schedule for today"	Meeting, Schedule	Ham

Learning Process:

1. **Input Features**: The email's content is analyzed to extract relevant features, such as keywords, email sender, frequency of certain words (e.g., "free," "discount," "buy"), and other patterns that are common in spam emails.
2. **Inductive Inference**: The AI system (e.g., a machine learning algorithm like Naive Bayes, Decision Trees, or a Neural Network) uses these labeled examples (features and labels) to identify patterns in the data. For example, it might learn that emails containing the words "buy" or "discount" are more likely to be spam.
3. **Hypothesis Generation**: The learning algorithm generates a **hypothesis** based on the data. For instance:
    - If an email contains keywords like "buy," "offer," or "discount," it is likely to be spam.
    - If the email contains words like "meeting," "schedule," or mentions a bank account, it is likely to be ham (not spam).
4. **Generalization**: After learning from the training examples, the spam filter generalizes these patterns into a model that can be applied to new, unseen emails.

Prediction:

Once trained, the spam filter can classify new emails based on the learned rules. For example, if a new email arrives with the content "50% off on all products," the filter will recognize the words "off" and "products" as features associated with spam and classify the email as **spam**.

Evaluation:

The model's performance is evaluated by testing it on a set of new emails that were not part of the training data. If the model correctly classifies most of the new emails as spam or not spam, it is considered to have generalized well.

Key Characteristics of Inductive Learning in AI:

1. **Learning from Examples**: Inductive learning involves deriving a general rule from specific examples (labeled data).
2. **Generalization**: The goal is to build a model that works well on unseen data, not just the examples in the training set.
3. **Hypothesis Space**: The model searches for the best hypothesis within a predefined hypothesis space, which defines the types of patterns or functions it can learn (e.g., decision boundaries, rules).

Inductive learning is essential for AI systems that need to generalize from specific examples to make decisions or predictions on new data. In the **email spam filter** example, the AI system learns from historical emails (the training data) and develops a rule that can classify future emails as spam or not spam, even if the exact email content is new or unseen.

an **inductive learning algorithm** implemented in AI using the **Naive Bayes classifier**, which is a probabilistic machine learning algorithm often used for classification tasks.

Problem:

We want to classify whether an email is **spam** or **not spam (ham)** based on certain keywords in the email content. We'll use **Naive Bayes** to implement the inductive learning process

Example Dataset:

Email Content	Features (Keywords)	Label (Spam or Ham)
"Buy now! Huge discount on laptops"	Buy, Discount	Spam
"Your bank account statement"	Bank, Account	Ham
"Limited time offer for insurance"	Offer, Insurance	Spam
"Meeting schedule for today"	Meeting, Schedule	Ham

Steps of Naive Bayes:

1. **Training Phase**: Learn the probability distribution for each feature (keywords) given the class label (Spam or Ham).
2. **Prediction Phase**: For a new email, calculate the probability of it being Spam or Ham based on its features, and classify the email.

```python
import math
# Sample dataset
emails = [
    ("Buy now! Huge discount on laptops", "Spam"),
    ("Your bank account statement", "Ham"),
    ("Limited time offer for insurance", "Spam"),
    ("Meeting schedule for today", "Ham")
]
# Preprocessing: Extract features (keywords) from emails
keywords = ["Buy", "Discount", "Bank", "Account", "Offer", "Insurance", "Meeting", "Schedule"]
# Dictionary to store count of keywords for each class
feature_counts = {"Spam": {key: 0 for key in keywords}, "Ham": {key: 0 for key in keywords}}
class_counts = {"Spam": 0, "Ham": 0}
# Count occurrences of each keyword in the emails
for email, label in emails:
    class_counts[label] += 1
    for word in keywords:
        if word in email:
            feature_counts[label][word] += 1
# Ca (function) def calculate_probabilities() -> tuple[float, float, dict[str, dict]]
def calculate_probabilities():
    total_emails = len(emails)
    spam_prob = class_counts["Spam"] / total_emails
    ham_prob = class_counts["Ham"] / total_emails
    # Probabilities for features given the class (P(feature|class))
    feature_probabilities = {"Spam": {}, "Ham": {}}
    for label in ["Spam", "Ham"]:
        total_features = sum(feature_counts[label].values())  # total count of keywords for the class
        for word in keywords:
            feature_probabilities[label][word] = (feature_counts[label][word] + 1) / (total_features + len(keywords))

    return spam_prob, ham_prob, feature_probabilities
```

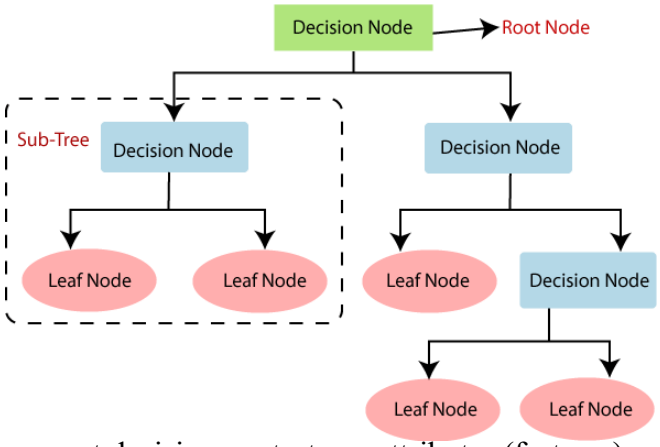
```python
    return spam_prob, ham_prob, feature_probabilities
# Function to classify new email
def classify_email(email):
    spam_prob, ham_prob, feature_probabilities = calculate_probabilities()
    # Calculate the probability of Spam and Ham for the new email
    spam_score = math.log(spam_prob)
    ham_score = math.log(ham_prob)
    for word in keywords:
        if word in email:
            spam_score += math.log(feature_probabilities["Spam"][word])
            ham_score += math.log(feature_probabilities["Ham"][word])
    # Classify based on which score is higher
    if spam_score > ham_score:
        return "Spam"
    else:
        return "Ham"
# Test the classifier with a new email
new_email = "Get a huge discount on products"
prediction = classify_email(new_email)
print(f"The email is classified as: {prediction}")
```

Output:
The email is classified as: Ham

DECISION TREE:

A **decision tree** is a supervised machine learning algorithm that is used for both classification and regression tasks.

- Decision trees are a fundamental component in AI, particularly in machine learning, where they are used for classification and regression tasks.
- They are a model that divides data into branches to make decisions or predictions based on the features of the data.
- It models decisions and their possible consequences, including chance event outcomes, resource costs, and utility.
- The decision tree is a flowchart-like structure where:



- **Nodes** represent decisions or tests on attributes (features).
- **Branches** represent the outcome of the test.
- **Leaf nodes** represent the final decision or class label.

A decision tree is a tree-like structure

- Each internal node represents a **test** or **decision** based on an attribute (feature).
- Each branch represents the outcome of the test.
- Each leaf node represents a **class label** (in classification) or **value** (in regression).

Real-World Example

A common real-world example of a decision tree is in loan approval. Given a person's financial details (such as credit score, income, and loan amount), a decision tree could predict whether they will be approved for a loan or not.

- **Root node**: "Does the applicant have a credit score greater than 650?"
- **Branch**: If yes, check income, if no, reject the loan.
- **Leaf node**: Final decision, like "Loan Approved" or "Loan Rejected."

Characteristics of Decision Trees

1. **Hierarchical structure**: Decision trees consist of a root, decision nodes, and leaf nodes, forming a tree-like structure.
2. **Non-linear model**: Unlike linear models, decision trees can model non-linear relationships between features and the target variable.

3. **Easy to interpret**: The decision-making process is transparent and easy to understand.
4. **Works with both categorical and continuous data**: Decision trees can handle both types of data.
5. **Greedy algorithm**: Decision trees are typically built using a greedy approach, selecting the best split at each step without considering future splits.

Applications of Decision Trees

- **Healthcare**: Predicting patient outcomes or classifying diseases based on medical records (e.g., diagnosing cancer from medical scans).
- **Finance**: Credit scoring and loan approval.
- **E-commerce**: Customer segmentation and predicting buying behavior.
- **Marketing**: Targeted advertising and identifying profitable customer segments.
- **Manufacturing**: Quality control and predictive maintenance.

Advantages of Decision Trees

1. **Easy to Understand**: Decision trees are simple to visualize and interpret, making them useful for explaining the model to stakeholders.
2. **No Data Scaling Required**: Decision trees do not require normalization or scaling of data, unlike algorithms such as KNN or SVM.
3. **Versatile**: Can be used for both classification and regression problems.
4. **Handles Missing Values**: Decision trees can handle missing values by splitting on available features.
5. **Non-Linear Relationships**: They can model complex, non-linear relationships between features and the target variable.

Disadvantages of Decision Trees

1. **Overfitting**: Decision trees are prone to overfitting, especially with deep trees, which means they may model noise in the training data.
2. **Instability**: Small changes in the data can result in a completely different tree.
3. **Bias towards certain features**: Decision trees tend to favor features with more levels or categories, leading to bias.

4. **Poor generalization**: Without proper pruning, decision trees may have poor generalization to new, unseen data.
5. **Complexity**: If the tree becomes very deep, it becomes harder to interpret and less efficient.

Uses of Decision Trees

1. **Predictive modeling**: Used in both classification (e.g., spam detection) and regression (e.g., predicting house prices).
2. **Feature selection**: Decision trees can be used to identify the most important features in a dataset.
3. **Risk management**: In finance or healthcare, decision trees help model and assess risk factors.
4. **Automation**: Decision trees are employed in decision-making systems, automating processes like loan approval, credit risk assessment, and more.

Real-World Example of Decision Tree: Loan Approval System

In a **bank's loan approval system**, a decision tree can be used to decide whether a person should be approved for a loan or not based on various factors.

Scenario:

Imagine a bank wants to automate the loan approval process. They collect the following data from applicants:

- **Credit score** (e.g., 600, 650, 700, etc.)
- **Annual income** (e.g., $40,000, $60,000)
- **Loan amount** requested (e.g., $10,000, $50,000)
- **Employment status** (e.g., employed, self-employed, unemployed)
- **Debt-to-income ratio** (e.g., 30%, 40%)

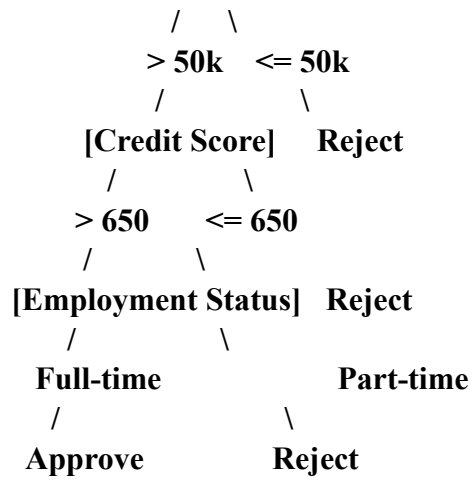
The bank can use a decision tree model to automatically approve or reject loan applications based on this data.

Decision Tree Construction:

1. **Root Node**: The first test (split) might be based on the **credit score**. For example, if the credit score is greater than 650, the applicant moves to the next node; if it's below 650, the loan application might be rejected right away.

2. **Branch 1**: If the credit score is above 650, the next test could be whether the **annual income** is greater than a certain threshold (e.g., $50,000).
    - If yes, move to the next node.
    - If no, consider whether the **debt-to-income ratio** is less than 30%.
3. **Branch 2**: If the debt-to-income ratio is below 30%, the loan might be approved. If the ratio is higher, the loan could be rejected.
4. **Leaf Nodes**: The final outcomes at the leaf nodes could be:
    - **Loan Approved**: When all conditions are met.
    - **Loan Denied**: When the conditions (credit score, income, debt-to-income ratio) do not meet the threshold.

Example Decision Tree:



- **Root Node**: Credit Score > 650?
    - **Yes**: Income > $50,000?
        - **Yes**: Loan Approved
        - **No**: Debt-to-Income Ratio < 30%?
            - **Yes**: Loan Approved
            - **No**: Loan Denied
    - **No**: Loan Denied

How Decision Trees Work

The decision tree algorithm splits data recursively, choosing the feature that best separates the data at each step. The process continues unt All data points are classified correctly, or

- The tree reaches a predefined depth, or

- The data cannot be further split in a meaningful way.

This splitting is based on a criterion, such as **Gini impurity**, **information gain**, or **variance reduction**, depending on the type of tree (classification or regression).

1.4 Decision trees as performance elements

- **Interpretability**: One of the biggest advantages of decision trees is their simplicity and interpretability. The decisions can be easily visualized, making it straightforward to understand how predictions are made. This is crucial in applications where model transparency is needed (e.g., healthcare or finance).
- **Handling Non-linearity**: Decision trees are non-parametric models, meaning they don't assume any specific form for the data distribution. This makes them powerful for modeling non-linear relationships between features.
- **Feature Selection**: During tree construction, decision trees perform automatic feature selection by choosing the most relevant features for splitting data at each step. This makes them less prone to overfitting compared to other models that may require manual feature selection.
- **Handling Missing Data**: Decision trees can handle missing data more naturally by making decisions based on available features and ignoring missing ones during splits. Some algorithms, like C4.5, implement strategies for dealing with missing values.
- **Scalability**: Decision trees can handle large datasets efficiently, though performance might degrade when data becomes too large or complex. Advanced versions, such as **Random Forests** and **Gradient Boosting Machines**, improve scalability and accuracy.
- **Overfitting**: One of the common issues with decision trees is overfitting, especially when the tree grows too deep, capturing noise in the data. Pruning methods (removing branches) are used to avoid this.
- **Adaptability**: Decision trees can be used for both classification (e.g., predicting the class of an object) and regression (e.g., predicting continuous values), making them versatile.

4. Limitations Decision trees in AI

- **Instability**: Small changes in the data can result in a very different tree structure.
- **Bias Toward Dominant Classes**: In imbalanced datasets, decision trees may favor the majority class.
- **Complexity with Deep Trees**: Deep trees are prone to overfitting, making them less generalizable.

5. Enhanced Models Using Decision Trees

- **Random Forest**: A collection of decision trees working together (ensemble method) to improve performance, reduce overfitting, and increase robustness.
- **Gradient Boosting**: A sequential ensemble method that builds trees one after another, each correcting the errors of the previous tree.

Example2 **The Problem: Should we wait for a table?**

Imagine you're standing at a restaurant entrance. You see it's busy. Now you must decide — **wait or leave?**

This is a typical **decision-making problem** — and in **AI**, we solve it using a **Decision Tree**.

To **predict** whether you (or anyone) will wait for a table at a restaurant.

How do we make this prediction?

We look at **10 important facts** (called **attributes** or **features**) about the situation.

Here's the list of those facts:

No.	Attribute	Description
1.	Alternate	Is there another good restaurant nearby? (Yes/No)
2.	Bar	Is there a bar area to wait comfortably? (Yes/No)
3.	Fri/Sat	Is today Friday or Saturday? (True/False)
4.	Hungry	Are we hungry? (Yes/No)
5.	Patrons	How many people are in the restaurant? (None / Some / Full)
6.	Price	What's the cost? ($ / $$ / $$$)
7.	Raining	Is it raining outside? (Yes/No)

8.	Reservation	Did we make a reservation? (Yes/No)
9.	Type	What kind of food is served? (French / Italian / Thai / Burger)
10.	WaitEstimate	How long is the wait? (0–10 / 10–30 / 30–60 / >60 minutes)

How Decision Tree Works Here

A **decision tree** is like a flowchart that helps make decisions based on answers to **Yes/No or multiple-choice questions**.

Here's how you use the tree:

1. **Start at the top (root).**
2. Ask the question written at that node.
3. Follow the arrow based on the answer.
4. Continue asking until you reach a **leaf** (Yes/No decision).

🟩 Example Case:

Let's say:

- **Patrons = Full**
- **WaitEstimate = 0–10 mins**

You follow the tree:

- Start at the top.
- Check the **Patrons** value → it's **Full** → go down that branch.
- Then, check **WaitEstimate** → it's **0–10** → tree says: 🟩 **Yes, we will wait!**

This is called **classification** – based on conditions, we classified the situation as "Wait."

Not All Attributes Are Useful

Interestingly, the decision tree **ignores** some attributes like:

- **Price**
- **Type**
- This means: in this particular learned model, **price** and **type of food** don't really affect our decision to wait.This is how the model **automatically finds what's important** and ignores the rest.

Concept	**Explanation**
Goal	Predict "Will Wait" (Yes/No)
Attributes	Facts about the situation
Decision Tree	A flowchart-like structure that uses attributes to make decisions
Example	Patrons = Full, Wait = 0–10 → We will wait
Irrelevant Info	Some attributes (like Price or Type) may not affect the final decision
AI Relevance	This is a supervised learning model used to make predictions based on past data

In Artificial Intelligence (AI), a **decision tree** is a popular and easy-to-understand model used for decision making. It works like a **flowchart** to make predictions based on **conditions**.

In this example, we're trying to decide:
"Should we wait for a table at a restaurant?"

To help make that decision, we collect information about the situation, such as:

- Is there an alternative restaurant nearby?
- Is there a bar to wait in?
- Is it a busy day (Friday/Saturday)?
- Are we hungry?
- How many people are in the restaurant (None, Some, Full)?
- How long is the wait time (0–10 mins, 10–30, etc.)?

These are called **attributes**.

The **decision tree** uses these attributes to **split the data** into smaller parts and find patterns. Based on those patterns, it predicts whether the answer is **Yes (Wait)** or **No (Don't Wait)**.

For example:

- If **Patrons = Full**, and **WaitEstimate = 0–10 mins**, the decision tree says: 🟩

  Yes, we will wait.

- If **Patrons = None**, the decision might be + **No, we won't wait**, assuming no one being there is a bad sign.

The tree **ignores unhelpful information**, like **Price** and **Type**, because those factors didn't help much in past decisions (according to the learning data).

- A **decision tree** is a powerful tool in AI for making decisions based on several conditions.
- It works like a set of "if-then" rules, which makes it **easy to follow and understand**.
- It helps **classify** new situations by learning from past examples.
- In the restaurant example, it learns when we are likely to wait based on past situations.
- Not all information is equally important — the tree **focuses only on the most useful attributes**.
- Decision trees are widely used in AI for tasks like **classification, prediction, and decision support systems**.

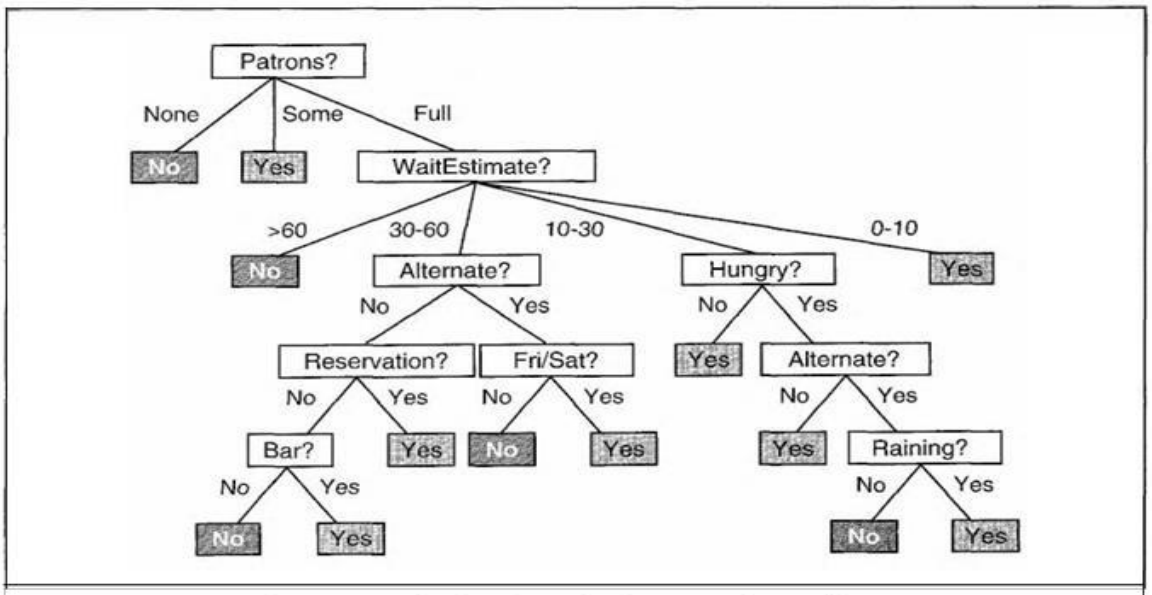


Figure 18.2 A decision tree for deciding whether to wait for a table.

EXPRESSIVENESS OF DECISION TRESS

🟩 What is a Decision Tree?

A **decision tree** is like a **flowchart** that helps us make decisions based on answering yes/no questions at each step.

Think of it like ordering food at a restaurant:

- Is it raining? → Yes
- Is there indoor seating? → No
- Is the wait time less than 30 minutes? → Yes
  🟩 **I'll Wait**
  ➕ **I'll Leave**

This kind of structure is **great for making decisions based on specific combinations of attributes**.

🧠 Logical Form of a Decision Tree

A decision tree says:

"I will wait **if** any of these conditions are true."

Logically:

WillWait(s) ← P1(s) ∨ P2(s) ∨ ... ∨ Pn(s)

Where each Pi(s) is a combination of tests like:

- (IsWeekend(s) ∧ Hungry(s) ∧ Price(s) = $)
- (Raining(s) ∧ HasUmbrella(s) ∧ NoWait(s))

So even though it *looks* like logic, it's basically just a fancy **propositional statement** — one input, many yes/no tests.

What Can't Decision Trees Do Well?

They struggle with cases where you compare **two or more different objects**.

➕ Example:

"Is there a restaurant nearby that is **cheaper** than this one?"
That needs to compare **two restaurants** — not just attributes of a single one. Decision trees can't naturally represent this.
We *could* add a special Boolean feature like CheaperRestaurantNearby, but we'd need to add **tons** of such features for every possible combination — which becomes **impossible (intractable)**.

Good News: Decision Trees Are Powerful!

- They can represent **any Boolean function**.
- Just like a truth table: each row becomes a **path** in the tree.

But... this can result in **huge** trees.

1 Where Decision Trees Struggle

Some functions are very **complex** to represent with decision trees.

Examples:

1. **Parity Function**:
   - Output = 1 if **even number of inputs are 1**
   - Example: [1, 0, 1, 0] → Even → 1
   - Decision tree becomes **huge**.
2. **Majority Function**:
   - Output = 1 if **more than half** inputs are 1.
   - Requires checking combinations — not easy for a tree.

Decision Tree

Day	Weather	Temperature	Humidity	Wind	Play football ?
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.

Formula : $IG_1 = \frac{-P}{P+N}\log_2\left(\frac{P}{P+N}\right) * \frac{-N}{P+N}\log_2\left(\frac{N}{P+N}\right)$

$E(A) = \sum_{i=2}^{v} \frac{P_i + N_i}{P+N} (I(P_i, N_i))$

Gain = $IG_1 - E(A)$

1) Entropy for entire dataset

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

2) Entropy for the weather

* Entropy for Sunny $\{+2, -3\} = \frac{-2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.97$

* Entropy for Cloudy $\{+4, -0\} = \frac{-4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0.$

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

3) Entropy for Temperature

* Entropy for {+2, -2} = $-\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 1.0$
HOT

* Entropy for {+3, -1} = $-\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.81$
Cool

* Entropy for {+4, -2} = $-\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} = 0.91$
Mild

4) Entropy for Humidity

* Entropy for {+3, -4} = $-\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} = 0.98$
high

* Entropy for {+6, -1} = $-\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7} = 0.59$
Normal

5) Entropy for wind

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

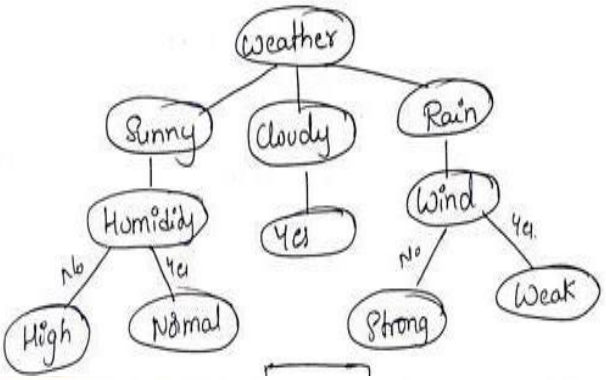
* Entropy for {+6, -2} = $-\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8} = 2.81$
weak

(a) Weather

IG = Entropy (whole data) $-\frac{5}{14}$ Entr(S) $-\frac{4}{14}$ Entr(C) $-\frac{5}{14}$ Entr(R)

IG = $0.94 - \frac{5}{14} \times 0.97 - \frac{4}{14} \times 0 - \frac{5}{14} \times 0.97$

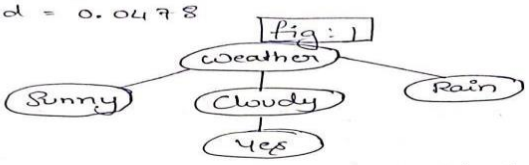
IG = 0.246.



Similary the Information Gain for :-

1) Temperature = 0.029
2) Humidity = 0.15
3) Wind = 0.0478

Fig: 1



Take the Sunny table and the calculated value is as follows :-

Day	Weather	Temperature	Humidity	Wind	Play football ?
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.

Information Gain (Temperature) = 0.57
Information Gain (Humidity) = 0.97 [maximum value]
Information Gain (Wind) = 0.019

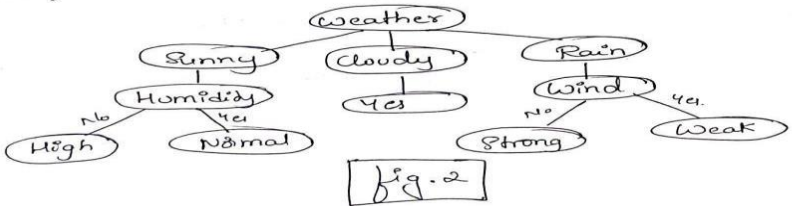


fig. 2

Noise and Overfitting. Explain Broadening the Applicability of Decision Trees.

Noise is any random error or irrelevant information in your data that doesn't represent the true relationship between the features (inputs) and the target (output).

- Example: Imagine you're trying to predict house prices. Some houses may be priced oddly high because the owner is emotionally attached and won't accept less. That odd pricing is *noise*.

Overfitting happens when a machine learning model learns *too much* from the training data — it captures the *noise* instead of the actual patterns. As a result, the model performs well on training data but poorly on new (unseen) data.

- Real-world example: A tutor who teaches students only the exact questions that came in last year's exam. If the exam changes slightly, the students are confused because they "overfitted" their learning to last year's paper instead of understanding the subject properly.

Noise → messy, random data

Overfitting → model memorizes noise instead of learning the pattern

Broadening the Applicability of Decision Trees

Decision Trees are powerful but have some issues:

- They can overfit easily.
- They might not work well on different types of problems (like very noisy data).

Broadening the Applicability means making Decision Trees more flexible, stable, and useful in more situations. Some ways to do this include:

Technique	How it helps	Real-World Example
Pruning	Cut back the tree after it grows too complex to avoid overfitting.	Cutting off unnecessary branches like "color of the house" if it's not really affecting house price.
Ensemble Methods (e.g., Random Forests)	Use lots of decision trees together (each trained slightly differently) to make a more accurate and robust model.	Amazon recommending products by averaging thousands of small decision tree predictions.
Boosting (e.g., XGBoost)	Combine many weak trees, each correcting errors from the previous one, to build a strong model.	Fraud detection where catching rare fraud cases requires careful layering of many small decisions.

Handling Continuous Features Better	Decision trees can split on numeric values cleverly rather than only categorical ones.	Predicting patient age-related risk factors where age is a continuous number, not a category.
Use of Soft Splits	Instead of a hard yes/no at each node, allow probabilities.	Diagnosing diseases where symptoms are not binary (fever might be mild or severe).

Concept	Explanation	Real World
Decision Tree	Flowchart for yes/no decisions	Should I wait at a restaurant?
Strength	Good at handling specific attribute combinations	Weather, wait time, price, etc.
Limitation	Can't compare multiple entities easily	Cheaper alternative nearby
Complexity	Can represent any Boolean function, but size can explode	Parity/Majority functions are hard
General Truth	No single representation is perfect for all logic	Smart algorithms are needed

Real-World Example: Restaurant Decision

Decision to be made:

"Should I wait for a table at this restaurant?"

Input Attributes (Features):

- Is it **raining**? (Yes/No)
- Is the restaurant **cheap**? (Yes/No)
- Is there a **long wait time**? (Yes/No)
- Am I **hungry**? (Yes/No)

Hungry?

├── No → Don't Wait
└── Yes → Price Cheap?

    ├── No → Don't Wait
    └── Yes → Wait Time Short?

├── Yes → Wait

└── No → + Don't Wait

This is a **simple decision tree** — it's **expressive enough** to model the logic of this decision using **combinations of input attributes**.

When It's Not Expressive Enough:

Now consider a new question:

"Is there a cheaper restaurant nearby than this one?"

To answer that, you need to compare **this restaurant** with **another restaurant**. That means you are dealing with **multiple objects** — not just one.

🔴 **Decision trees cannot naturally do that**.

They are limited to:

- One object at a time
- Simple yes/no based logic

To make this work, you'd have to create a special feature like:

CheaperRestaurantNearby = Yes/No

But doing this for **all possible comparisons** becomes **impossible** as the number of objects increases.

Feature	Can Decision Trees Handle It?	Example
Based on one object's attributes	🟩 Yes	Hungry + Cheap → Wait
Simple Boolean combinations	🟩 Yes	Yes/No type questions
Comparing multiple objects	+ No	Is another restaurant cheaper?

Complex patterns (like parity/majority)	➡ Difficult	Even number of "Yes" answers

Inducing decision trees from examples

Inducing Decision Trees from Examples means building a decision tree model based on a set of labeled data (i.e., examples with known outcomes). The goal is to find patterns in the data and use them to make decisions or predictions for new data points.

🔍Definition:

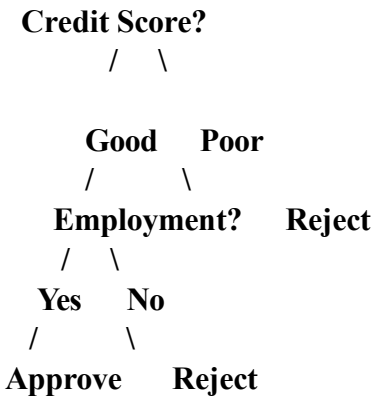
Inducing a decision tree is the process of automatically constructing a decision tree from a dataset using algorithms like ID3, C4.5, CART, etc. The tree is built by recursively splitting the data based on features that best separate the classes, typically using measures like Information Gain or Gini Index.

🤔**Real-World Example: Loan Approval System**

Imagine a bank that wants to build a system to approve or reject loan applications. It has historical data of past applicants, including:

Age	Income	Employment	Credit Score	Loan Approved
25	**Low**	**No**	**Poor**	**No**
45	**High**	**Yes**	**Good**	**Yes**
35	**Medium**	**Yes**	**Good**	**Yes**
50	**Medium**	**No**	**Good**	**No**
28	**Low**	**Yes**	**Poor**	**No**

Using this dataset (examples), the algorithm induces a decision tree like this:



🟩 **How It Works:**

- Step 1: Start with all data and choose the best attribute to split (e.g., *Credit Score*).
- Step 2: For each branch, split the data again using the next best attribute (e.g., *Employment*).

- Step 3: Repeat until all data is classified or no further splits are beneficial.

Outcome:

The decision tree acts as a model to predict future loan approvals based on user input like credit score and employment status.

Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*3040*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	*Yes*

Figure 18.3 Examples for the restaurant domain.

Problem Statement

Goal: Predict whether a person will play tennis based on weather conditions. Medium

Attributes:

- **Outlook: Sunny, Overcast, Rainy**
- **Temperature: Hot, Mild, Cool**
- **Humidity: High, Normal**
- **Wind: Weak, Strong**Spiceworks Inc

Target Variable: PlayTennis (Yes or No)

Sample Dataset

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rainy	Mild	High	Weak	Yes

Rainy	**Cool**	**Normal**	**Weak**	**Yes**
Rainy	**Cool**	**Normal**	**Strong**	**No**
Overcast	**Cool**	**Normal**	**Strong**	**Yes**
Sunny	**Mild**	**High**	**Weak**	**No**
Sunny	**Cool**	**Normal**	**Weak**	**Yes**
Rainy	**Mild**	**Normal**	**Weak**	**Yes**
Sunny	**Mild**	**Normal**	**Strong**	**Yes**
Overcast	**Mild**	**High**	**Strong**	**Yes**
Overcast	**Hot**	**Normal**	**Weak**	**Yes**
Rainy	**Mild**	**High**	**Strong**	**No**

Step-by-Step Algorithm Execution

1. Check for Uniform Classification:
   - The dataset contains both 'Yes' and 'No' in the PlayTennis column.
   - Proceed to the next step.
2. Check for Empty Attributes:
   - Attributes are available: Outlook, Temperature, Humidity, Wind.
   - Proceed to the next step.
3. Select the Best Attribute:
   - Calculate the Information Gain for each attribute.
   - Suppose 'Outlook' has the highest Information Gain.
   - Select 'Outlook' as the decision node.Medium+1Kinds on the Genius+1
4. Create Decision Node for 'Outlook':
   - Split the dataset into subsets based on 'Outlook' values: Sunny, Overcast, Rainy.
5. Recursive Steps for Each Subset:
   - Subset: Outlook = Sunny
     - PlayTennis: 2 'No', 1 'Yes'
     - Attributes left: Temperature, Humidity, Wind
     - Select the best attribute (e.g., 'Humidity')
     - Create a decision node for 'Humidity'
     - Split further based on 'Humidity' values
   - Subset: Outlook = Overcast
     - PlayTennis: All 'Yes'
     - Create a leaf node with classification 'Yes'
   - Subset: Outlook = Rainy
     - PlayTennis: 3 'Yes', 2 'No'
     - Attributes left: Temperature, Humidity, Wind
     - Select the best attribute (e.g., 'Wind')

- **Create a decision node for 'Wind'**
- **Split further based on 'Wind' values**

6. **Continue Recursion:**
   - **Repeat the process for each subset until:**
     - **All instances have the same classification.**
     - **No more attributes are left.**
     - **A stopping criterion is met (e.g., maximum tree depth).**

Final Decision Tree Structure

Outlook?

├── Sunny:
│ └── Humidity?
│ ├── High: No
│ └── Normal: Yes
├── Overcast: Yes
└── Rainy:
└── Wind?
├── Weak: Yes
└── Strong: No

Decision Trees split the dataset based on attributes that provide the most significant information gain.

- **The process is recursive and continues until a stopping condition is met.**
- **The final tree can be used to make predictions on new data by traversing the tree based on attribute values.**

Algorithm:

```
Function DECISION-TREE-LEARNING(examples, attributes, default):
   If examples is empty:
      Return default
   If all examples have the same classification:
      Return the classification
   If attributes is empty:
      Return MAJORITY-VALUE(examples)
   best = CHOOSE-ATTRIBUTE(attributes, examples)
   tree = New decision node with test: best
   For each value v of best:
      examples_v = subset of examples where best = v
      subtree = DECISION-TREE-LEARNING(examples_v, attributes - {best}, MAJORITY-VALUE(examples))
      Add branch to tree with label v and subtree
   Return tree
```

Function: DECISION_TREE_LEARNING(examples, attributes, default)
Inputs:

- examples: A set of training instances.
- attributes: A list of attributes/features that can be used to split the data.
- default: The default classification to return if no examples are available.

Output:

- A decision tree that classifies instances based on the provided attributes.

Step-by-Step Procedure

1. **Check if examples is empty**:
   - **Yes**: Return default.
   - **No**: Proceed to the next step. ResearchGate
2. **Check if all examples have the same classification**:
   - **Yes**: Return that classification (create a leaf node).
   - **No**: Proceed to the next step.
3. **Check if attributes is empty**:
   - **Yes**: Return the majority classification among examples.
   - **No**: Proceed to the next step.IBM - United States+7Wikipedia+7Medium+7
4. **Select the best attribute to split on**:
   - Use a selection criterion (e.g., Information Gain) to choose the attribute that best separates the examples.
   - Let's denote this attribute as best.
5. **Create a decision node that splits on best**:
   - Initialize a new decision tree node with best as the test condition.
6. **For each possible value v of attribute best**:
   - Create a subset examples_v of examples where best == v.
   - Recursively call DECISION_TREE_LEARNING(examples_v, attributes - {best}, majority_classification) to create a subtree.
   - Attach the subtree to the current decision node with a branch labeled v.
   - **Return the constructed decision tree.**

Choosing attribute tests

Attribute selection (or feature selection) is a key step in machine learning and AI models, where the goal is to select the most relevant attributes (or features) from the dataset to improve the model's performance, reduce complexity, and prevent overfitting.

Types of Attribute Selection Methods:

1. **Filter Methods**: These methods evaluate the relevance of features based on statistical techniques before the learning algorithm is applied.

   Example: **Chi-Square Test** for selecting features based on their statistical significance

2. **Wrapper Methods**: These methods evaluate feature subsets by using a predictive model, and the subset that leads to the best performance is selected.
    - Example: **Recursive Feature Elimination (RFE)** which uses the performance of the model to decide which features to eliminate.
3. **Embedded Methods**: These methods perform feature selection during the model training process, often using regularization techniques.
    - Example: **Lasso Regression**, where feature selection is done by shrinking coefficients of less important features to zero

Regression: Regression is a statistical method used to model and analyze the relationships between a dependent variable and one or more independent variables. Its main goal is to predict the value of the dependent variable based on the values of the independent variables.

Regression is a process of finding the correlations between dependent and independent variables. It helps in predicting the continuous variables such as prediction of **Market Trends**, prediction of House prices, etc.

The task of the Regression algorithm is to find the mapping function to map the input variable(x) to the continuous output variable(y).

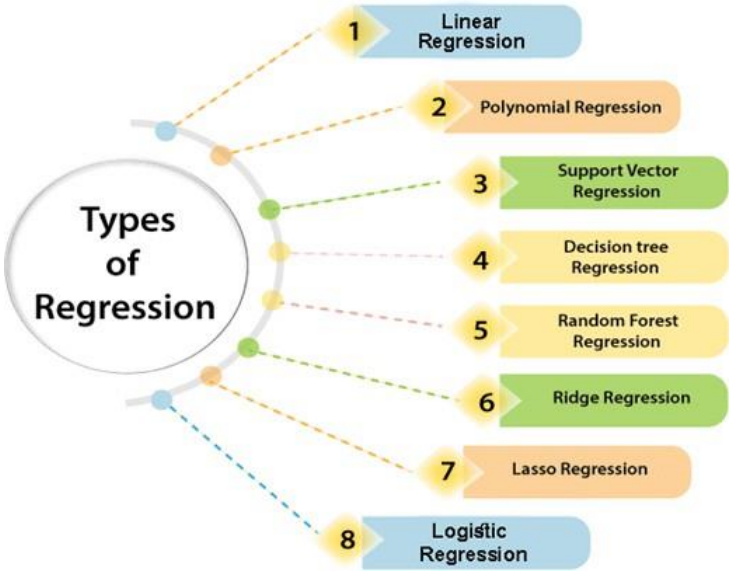
Example: Suppose we want to do weather forecasting, so for this, we will use the Regression algorithm. In weather prediction, the model is trained on the past data, and once the training is completed, it can easily predict the weather for future days.

Types of Regression

There are various types of regressions which are used in data science and machine learning. Each type has its own importance on different scenarios, but at the core, all the regression methods analyze the effect of the independent variable on dependent variables. Here we are discussing some important types of regression which are given below:

- **Linear Regression**
- **Logistic Regression**
- **Polynomial Regression**
- **Support Vector Regression**
- **Decision Tree Regression**
- **Random Forest Regression**
- **Ridge Regression**
- **Lasso Regression:**

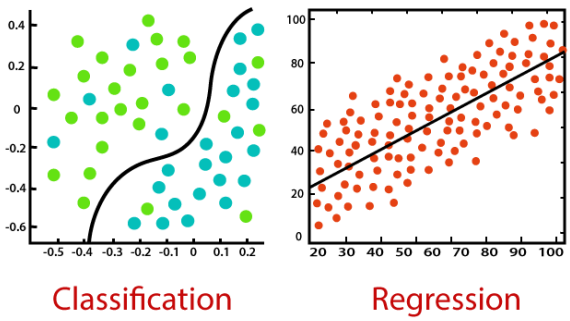
.



Regression and Classification algorithms are Supervised Learning algorithms. Both the algorithms are used for prediction in Machine learning and work with the labeled datasets. But the difference between both is how they are used for different machine learning problems.

The main difference between Regression and Classification algorithms that Regression algorithms are used to **predict the continuous** values such as price, salary, age, etc. and Classification algorithms are used to **predict/Classify the discrete values** such as Male or Female, True or False, Spam or Not Spam, etc.

Consider the below diagram:



Classification: Classification is a process of finding a function which helps in dividing the dataset into classes based on different parameters. In Classification, a computer program is trained on the training dataset and based on that training, it categorizes the data into different classes.
The task of the classification algorithm is to find the mapping function to map the input(x) to the discrete output(y).
Example: The best example to understand the Classification problem is Email Spam Detection. The model is trained on the basis of millions of emails on different parameters, and whenever it receives a new email, it identifies whether the email is spam or not. If the email is spam, then it is moved to the Spam folder.
Types of ML Classification Algorithms:
Classification Algorithms can be further divided into the following types:

- Logistic Regression
- K-Nearest Neighbours
- Support Vector Machines
- Kernel SVM
- Naïve Bayes

- Decision Tree Classification
- Random Forest Classification

Logistic Regression Algorithm: Logistic regression may be a **supervised learning** classification algorithm wont to **predict the probability** of a target variable. It's one among the only ML algorithms which will be used for various classification problems like **spam detection**, **Diabetes prediction**, **cancer detection** etc. Logistic regression is simpler to **implement**, **interpret**, and really **efficient** to coach.

If the amount of observations is lesser than the **amount of features**, **Logistic Regression** shouldn't be used, otherwise, it's going to cause **overfitting**.

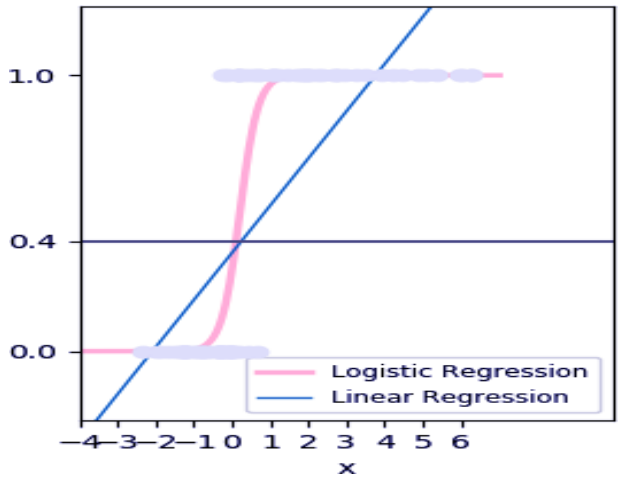
We use logistic regression for the binary classification of **data-points**. We perform categorical classification such that an output belongs to either of the **two classes (1 or 0)**.

For example – we can predict whether it will rain today or not, based on the **current weather conditions**.

Two of the important parts of logistic regression are **Hypothesis and Sigmoid Curve**. With the help of this hypothesis, we can derive the **likelihood** of the event.

The data generated from this hypothesis can fit into the **log function** that creates an **S-shaped curve** known as **"sigmoid"**. Using this **log function**, we can further predict the **category of class**.

We can represent the sigmoid as follows:



The produced graph is through this logistic function:

1 / (1 + e^-x)

The **'e'** in the above equation represents the **S-shaped curve** that has values between **0 and 1**.

We write the equation for **logistic regression** as follows:

y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x))

In the above equation, **b0** and **b1** are the **two coefficients** of the **input x**.

We estimate these two coefficients using **"maximum likelihood estimation"**.

Linear Regression:

- Linear regression is a statistical regression method which is used for predictive analysis.
- It is one of the very simple and easy algorithms which works on regression and shows the relationship between the continuous variables.
- It is used for solving the regression problem in machine learning.
- Linear regression shows the linear relationship between the independent variable (X- axis) and the dependent variable (Y-axis), hence called linear regression.
- If there is only one input variable (x), then such linear regression is called **simple linear regression**. And if there is more than one input variable, then such linear regression is called **multiple linear regression**.

- The relationship between variables in the linear regression model can be explained using the below image. Here we are predicting the salary of an employee on the basis of **the year of experience**.

Define linear Model. In AI, a linear model is a type of mathematical model that makes predictions based on a linear relationship between input features and the output.

In simple terms:

- It assumes the output is a straight-line combination of the inputs.
- It looks like this:

$$y = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

where:

- y = predicted output
- $x_1, x_2, \ldots, x_n$ = input features
- $w_1, w_2, \ldots, w_n$ = weights (parameters the model learns)
- b = bias (an extra constant term)

Examples of linear models in AI:

- Linear Regression (for predicting continuous values)
- Logistic Regression (for classification tasks)

Linear Regression (for predicting continuous values)

Linear Regression is a supervised machine learning algorithm used to predict continuous numeric values by finding the best-fitting straight line (or hyperplane) that describes the relationship between one or more independent variables (features) and a dependent variable (target).

Univariate Linear Regression

It's when we try to model the relationship between one feature (independent variable, say X) and a target (dependent variable, say y) using a straight line:

y=mX+c

where:

- m = slope (how much y changes for a unit change in x)
- c = intercept (where the line crosses the y-axis)

Predict a person's weight based on their height.

Suppose you have the following data:

Height (in cm)	Weight (in kg)
150	50
160	55
170	60
180	65
190	70

We want to **build a model** that can predict someone's weight given their height.

🛠️ Step 1: Calculate the values

First, we calculate the **slope (m)** and **intercept (c)** manually using formulas:

Slope (m):

$$m = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$$

Intercept (c):

$$c = \frac{\sum y - m(\sum x)}{n}$$

Where:

- n is number of data points
- $\sum x$ is sum of all x (heights)
- $\sum y$ is sum of all y (weights)
- $\sum xy$ is sum of product of x and y
- $\sum x^2$ is sum of squares of x

Step 2: Compute necessary sums

Let's compute:

x (height)	y (weight)	xy	x^2
150	50	7500	22500
160	55	8800	25600
170	60	10200	28900
180	65	11700	32400
190	70	13300	36100

Now, add up:

- $\sum x = 150 + 160 + 170 + 180 + 190 = 850$
- $\sum y = 50 + 55 + 60 + 65 + 70 = 300$
- $\sum xy = 7500 + 8800 + 10200 + 11700 + 13300 = 51500$
- $\sum x^2 = 22500 + 25600 + 28900 + 32400 + 36100 = 145500$

and $n = 5$

Step 3: Plug into formulas

Slope (m):

$$m = \frac{5(51500) - (850)(300)}{5(145500) - (850)^2}$$

$$m = \frac{257500 - 255000}{727500 - 722500}$$

$$m = \frac{2500}{5000}$$

$$m = 0.5$$

Intercept (c):

$$c = \frac{300 - 0.5(850)}{5}$$

$$c = \frac{300 - 425}{5}$$

$$c = \frac{-125}{5}$$

$$c = -25$$

🎯 **Final Model:**

$$\boxed{y = 0.5x - 25}$$

🔮 **Example Prediction:**

Suppose someone is **175 cm** tall. Predict their weight:

$$y = 0.5(175) - 25$$

$$y = 87.5 - 25$$

$$y = 62.5\,\text{kg}$$

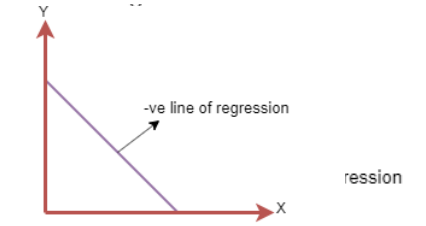
✅ According to our model, a person with **175 cm height** should weigh approximately **62.5 kg**.

1. Linear Regression Line

A linear line showing the relationship between the dependent and independent variables is called a regression line. A regression line can show two types of relationship:

- Positive Linear Relationship:

If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.



The line equation will be: $Y= a_0+a_1x$

- **Negative Linear Relationship:** If the dependent variable decreases on the Y-axis and independent variable increases on the X- axis, then such a relationship is called a negative linear relationship.

Multivariate Linear Regression:It is a machine learning technique where **we predict an output based on two or more input features**.

- It extends *simple linear regression* (which uses one input variable) by handling **multiple independent variables** (features).
- The main **goal** is to find the best-fit linear relationship between multiple inputs and a single output.

AI, multivariate regression is useful for:

- **Predictive modeling** (e.g., predicting housing prices, stock market trends, health risks)
- **Feature analysis** (e.g., finding important factors influencing an output)
- **Training AI systems** that require numeric predictions.

Suppose we have features $x_1, x_2, ..., x_n$.

The prediction model is:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n$$

Where:

- y = predicted value (dependent variable)
- $x_1, x_2, ..., x_n$ = input features
- θ_0 = bias (intercept)
- $\theta_1, ..., \theta_n$ = coefficients (weights)

We find θ values by minimizing the **error** between predictions and actual values — often using **Mean Squared Error (MSE)**.

Problem: Predict the price of a house based on:

- Size of the house (in square feet)
- Number of bedrooms
- Age of the house

Size (sq ft)	Bedrooms	Age (years)	Price ($)
2100	3	20	500,000
1600	2	15	330,000
2400	4	5	600,000
1400	3	30	275,000

Model:

$$\text{Price} = \theta_0 + \theta_1 \times \text{Size} + \theta_2 \times \text{Bedrooms} + \theta_3 \times \text{Age}$$

Suppose after training, we find:

- $\theta_0 = 50000$ (bias)
- $\theta_1 = 100$ (per square foot)
- $\theta_2 = 15000$ (per bedroom)
- $\theta_3 = -1200$ (per year of age)

Prediction Example: For a new house:

- Size = 1800 sq ft
- Bedrooms = 3
- Age = 10 years

Prediction:

$$\begin{aligned}\text{Price} &= 50000 + 100(1800) + 15000(3) - 1200(10) \\ &= 50000 + 180000 + 45000 - 12000 \\ &= 263,000\end{aligned}$$

Thus, the predicted price is **$263,000**.

Linear Classifiers with hard Threshold: A linear classifier is a model that separates data into classes based on a linear decision boundary. It tries to find a straight line (in 2D), a plane (in 3D), or a hyperplane (in higher dimensions) that divides the data into different categories.

Mathematically, a linear classifier makes predictions using a function like:

$$f(x) = w^T x + b$$

Where:

- x = input feature vector
- w = weight vector (learned parameters)
- b = bias (shifts the decision boundary)

What does "Hard Threshold" mean?

A **hard threshold** applies a strict decision rule:

- If the output $f(x) \geq 0$, classify as **Class 1**
- If $f(x) < 0$, classify as **Class 0**

This means there's no gradual probability or soft decision — it's a sharp, binary (0/1) decision at the threshold (usually 0).

Mathematically, it's like:

$$\text{prediction} = \begin{cases} 1, & \text{if } w^T x + b \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

This rule is **non-continuous** — even a tiny change in x around the decision boundary can flip the classification.

How does it work?

1. **Input**: The model gets input features (e.g., height, weight).
2. **Linear Combination**: It computes $w^T x + b$.
3. **Thresholding**: It applies the hard threshold at 0.
4. **Output**: Based on the result, it outputs a class label (0 or 1).

Suppose you want to classify whether a student passes (1) or fails (0) based on study hours.

Assume:

- $w = 2, b = -5$

Then:

$$f(x) = 2x - 5$$

Now, if a student studied:

- 4 hours: $f(4) = 2(4) - 5 = 3 \rightarrow 3 > 0 \rightarrow$ **Pass (1)**
- 2 hours: $f(2) = 2(2) - 5 = -1 \rightarrow -1 < 0 \rightarrow$ **Fail (0)**

Simple and sharp, right?

Visualization:

- In 2D, it's just a **line** splitting the plane.
- On one side: all points are Class 1.
- On the other side: all points are Class 0.

Feature	Univariate Linear Regression	Multivariate Linear Regression
Definition	Regression with **one independent variable** and one dependent variable.	Regression with **two or more independent variables** and one dependent variable.
Number of Features	1 feature (predictor).	2 or more features (predictors).
Mathematical Equation	y=mx+by = mx + by=mx+b	y=b0+b1x1+b2x2+⋯+bnxny = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_ny=b0+b1x1+b2x2+⋯+bnxn
Visualization	Straight **line** in 2D space.	**Plane** or **hyperplane** in multidimensional space.
Complexity	Simple and easy to implement.	More complex, needs handling of multiple variables.
Use Case	Prediction depends on a **single factor**.	Prediction depends on **multiple factors**.
Model Fitting Method	Ordinary Least Squares (OLS).	OLS, Ridge, Lasso (for better handling of many features).
Overfitting Risk	Low.	Higher if many features without sufficient data.

Interpretation	Easy: Effect of a single feature on output.	Complex: Effect of each feature, keeping others constant.
Example	Predicting **salary based on experience** only.	Predicting **house price based on area, location, and age**.

Regression Algorithm	Classification Algorithm
In Regression, the output variable must be of continuous nature or real value.	In Classification, the output variable must be a discrete value.
The task of the regression algorithm is to map the input value (x) with the continuous output variable(y).	The task of the classification algorithm is to map the input value(x) with the discrete output variable(y).
Regression Algorithms are used with continuous data.	Classification Algorithms are used with discrete data.
In Regression, we try to find the best fit line, which can predict the output more accurately.	In Classification, we try to find the decision boundary, which can divide the dataset into different classes.
Regression algorithms can be used to solve the regression problems such as Weather Prediction, House price prediction, etc.	Classification Algorithms can be used to solve classification problems such as Identification of spam emails, Speech Recognition, Identification of cancer cells, etc.
The regression Algorithm can be further divided into Linear and Non-linear Regression.	The Classification algorithms can be divided into Binary Classifier and Multi-class Classifier.

Linear Classification with logistic Regression

Linear classification is a method used in Artificial Intelligence (AI) and Machine Learning (ML) to separate data into different classes using a straight line (in 2D), a plane (in 3D), or a hyperplane (in higher dimensions).

Logistic Regression is a popular technique for linear classification, especially for binary classification problems (where there are only two classes, like "yes" or "no", "spam" or "not spam").

How Logistic Regression Works:

1. **Input and Linear Combination:**
   - Given input features $x = (x_1, x_2, \ldots, x_n)$,
   - Logistic regression computes a linear function:

$$z = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

   where $w_0, w_1, \ldots, w_n$ are weights.
2. **Sigmoid Activation:**
   - Instead of directly outputting z, logistic regression passes it through a **sigmoid function**:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

   - This **squashes** the output to a value between **0 and 1**, which can be interpreted as a **probability**.
3. **Prediction:**
   - If $\sigma(z) > 0.5$ → predict class 1,
   - Otherwise → predict class 0.

Why It's Called Linear Classification:

- Even though logistic regression outputs probabilities, the decision boundary (where it switches between classes) is linear.
- This means it separates the classes with a straight line (or hyperplane).

🛠️ Step-by-Step Logistic Regression Example

We'll continue with the **bank loan** case:

We are given **one input feature**:

- x = Applicant's Income (in $1000 units)

And a **binary output**:

- $y = 1$ if defaulted, $y = 0$ otherwise.

Let's take a tiny dataset:

Applicant	Income x	Defaulted y
1	2	0
2	4	0
3	6	1
4	8	1

Step 1 : Set up the Model

The logistic regression model is:

$$z = w_0 + w_1 x$$

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}$$

We need to find w_0 and w_1.

Initially, let's assume:

- $w_0 = 0$
- $w_1 = 0$

These will be updated during **training**.

Step 2 : Compute the Predictions

For Applicant 1:

- $x = 2$
- $z = 0 + 0 \times 2 = 0$
- $\hat{y} = \frac{1}{1+e^{-0}} = 0.5$

For Applicant 2:

- $x = 4$
- $z = 0 + 0 \times 4 = 0$
- $\hat{y} = 0.5$

Step 3 : Compute the Error (Loss)

Use the **Cross-Entropy Loss** formula:

$$\text{Loss} = -\left(y \log(\hat{y}) + (1-y)\log(1-\hat{y})\right)$$

For Applicant 1:

- $y = 0$
- $\hat{y} = 0.5$
- Loss = $-(0 \times \log(0.5) + (1-0)\log(1-0.5)) = -\log(0.5) = 0.693$

Same for others.

Step 4 : Update the Weights (Gradient Descent)

We use **gradients** to update w_0 and w_1.

Gradient for w_1 is:

$$\frac{\partial \text{Loss}}{\partial w_1} = (\hat{y} - y)x$$

Step 4 : Update the Weights (Gradient Descent)

We use **gradients** to update w_0 and w_1.

Gradient for w_1 is:

$$\frac{\partial \text{Loss}}{\partial w_1} = (\hat{y} - y)x$$

Similarly for w_0 (with $x = 1$).

👉 For Applicant 1:

- $\hat{y} = 0.5, y = 0, x = 2$
- Gradient = $(0.5 - 0) \times 2 = 1.0$

👉 For Applicant 2:

- Gradient = $(0.5 - 0) \times 4 = 2.0$

👉 For Applicant 3:

- Gradient = $(0.5 - 1) \times 6 = -3.0$

👉 For Applicant 4:

- Gradient = $(0.5 - 1) \times 8 = -4.0$

Total gradient for w_1 = $1.0 + 2.0 - 3.0 - 4.0 = -4.0$

Average gradient = $-4.0/4 = -1.0$

Step 5 : Update Rule

Suppose learning rate $\alpha = 0.1$.

Update weight:

$$w_1 = w_1 - \alpha \times \text{gradient}$$

$$w_1 = 0 - 0.1 \times (-1.0) = 0.1$$

Similarly update w_0.

Step 6 : Repeat

- Repeat Steps 2 to 5 for several **epochs** (passes through the data).
- Gradually, w_0 and w_1 will learn values that correctly separate defaulters and non-defaulters.

In the end, you get a **decision boundary**:
For example:

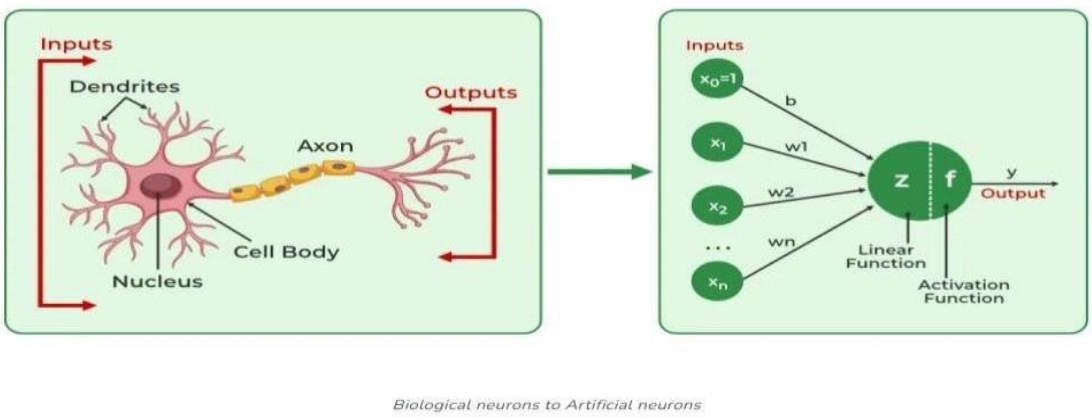
$$\text{Predict Default if } 0.1 \times x + w_0 > 0$$

🎯 Final Prediction:

For a new applicant with Income = 5:

- $z = w_0 + w_1 \times 5$
- Compute probability $\hat{y}$
- If $\hat{y} > 0.5$, predict Default, else No Default.

Artificial Neural Network :The term "Artificial Neural Network" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.



Biological neurons to Artificial neurons

Figure depicting the biological neuron and Artificial neuron

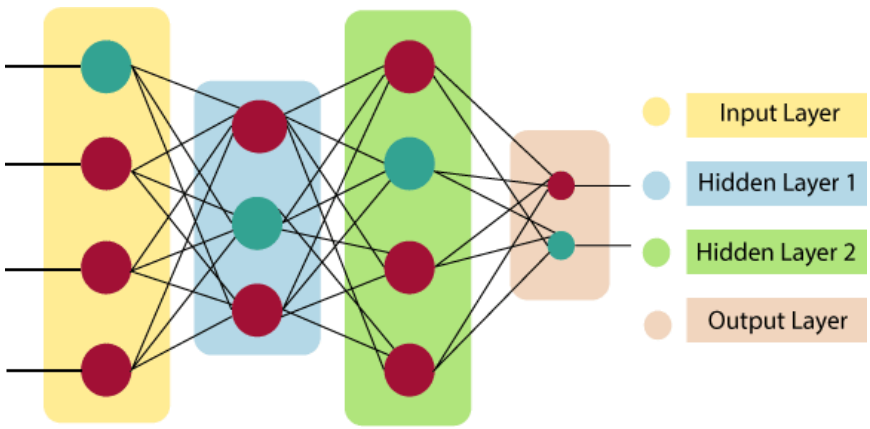
Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output.

Relationship between Biological neural network and artificial neural network:

Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Cell nucleus	Nodes
Synapse	Weights
Axon	Output

The architecture of an artificial neural network:

- Let us look at various types of layers available in an artificial neural network.
- Artificial Neural Network primarily consists of three layers:
- **Input Layer:** As the name suggests, it accepts inputs in several different formats provided by the programmer.
- **Hidden Layer:** The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns.
- **Output Layer:** The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.



It determines weighted total is passed as an input to an activation function to produce the output.

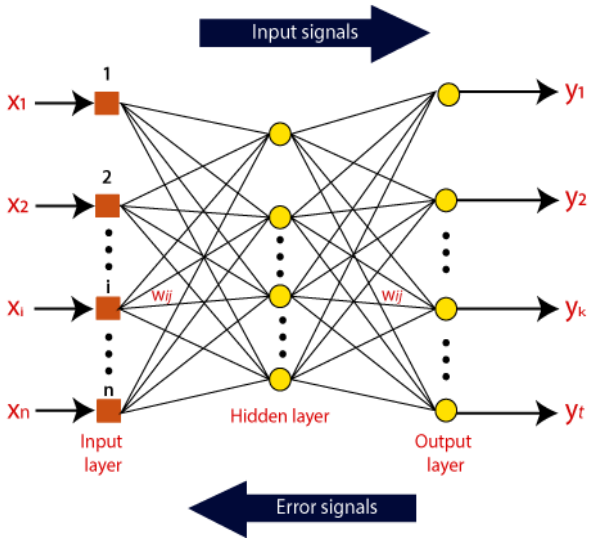
Activation: A mathematical function known as an activation function maps the input to theoutput, and executes activations.

$$\sum_{i=1}^{n} Wi * Xi + b$$

How do artificial neural networks work?

Artificial Neural Network can be best represented as a weighted directed graph, where the artificial neurons form the nodes. The association between the neurons outputs and neuron inputs can be viewed as the directed edges with weights. The Artificial Neural Network receives the input signal from the external source in the form of a pattern and image in the form of a vector. These inputs are then

mathematically assigned by the notations x(n) for every n number of inputs.



Afterward, each of the input is multiplied by its corresponding weights (these weights are the details utilized by the artificial neural networks to solve a specific problem). In general terms, these weights normally represent the strength of the interconnection between neurons inside the artificial neural network. All the weighted inputs are summarized inside the computing unit.

If the weighted sum is equal to zero, then bias is added to make the output non-zero or something else to scale up to the system's response. Bias has the same input, and weight equals to 1. Here the total of weighted inputs can be in the range of 0 to positive infinity.
Here, to keep the response in the limits of the desired value, a certain maximum value is benchmarked, and the total of weighted inputs is passed through the activation function.

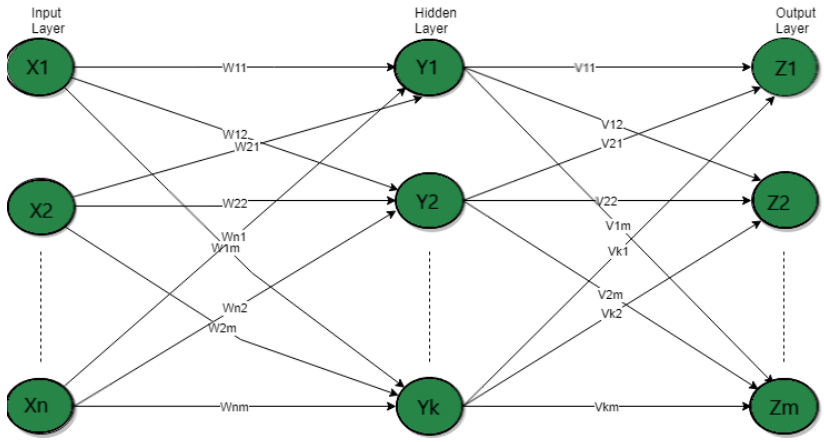
The activation function refers to the set of transfer functions used to achieve the desired output. There is a different kind of the activation function, but primarily either linear or non- linear sets of functions. Some of the commonly used sets of activation functions are the Binary, linear, and Tan hyperbolic sigmoidal activation functions. Let us take a look at each of them in details:
Binary:In binary activation function, the output is either a one or a 0. Here, to accomplish this, there is a threshold value set up. If the net weighted input of neurons is more than 1, then the final output of the activation function is returned as one or else the output is returned as 0.
Sigmoidal Hyperbolic:
The Sigmoidal Hyperbola function is generally seen as an "**S**" shaped curve. Here the tan hyperbolic function is used to approximate output from the actual net input. The function is defined as:
F(x) = (1/1 + exp(-????x))
Where ???? is considered the Steepness parameter.

Types of Artificial Neural Networks

1. Feedforward Neural Networks

Feedforward neural networks are a form of artificial neural network where without forming any cycles between layers or nodes means inputs can pass data through those nodes within the hidden level to the output nodes.

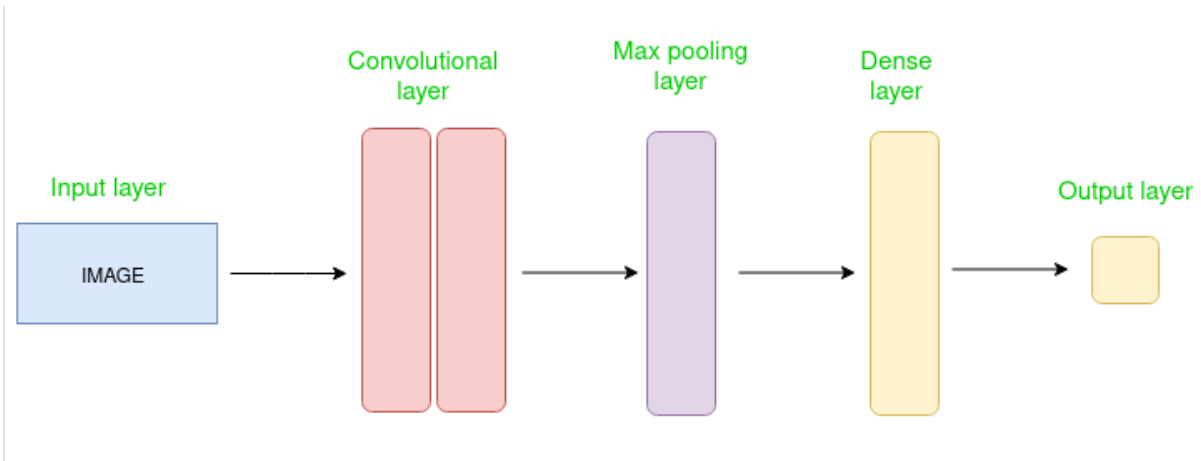
- Architecture: Made up of layers with unidirectional flow of data (from input through hidden and the output layer).
- Training: Backpropagation is often used during training for the main aim of reducing the prediction errors.
- Applications: In visual and voice recognition, NLP, financial forecasting, and recommending system



2. Convolutional Neural Networks (CNNs)

Convolutional neural networks structure is focused on processing the grid type data like images and videos by using convolutional layers filtering driving the patterns and spatial hierarchies.

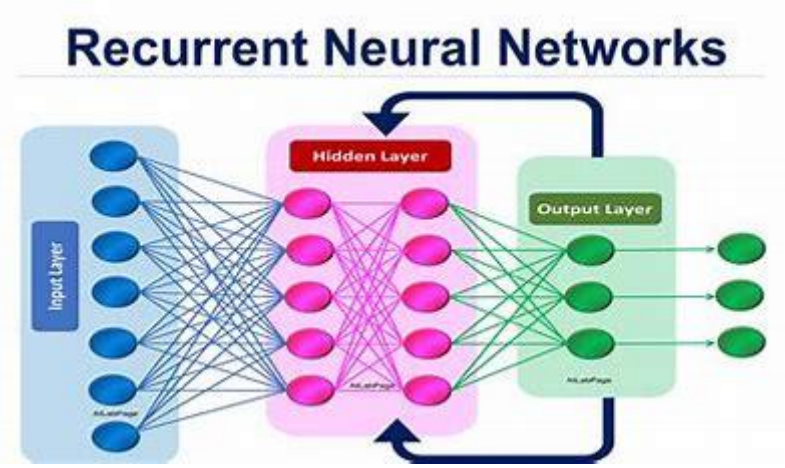
- Key Components: Utilizing convolutional layers, pooling layers and fully connected layers.
- Applications: Used for classification of images, object detection, medical imaging analyzes, autonomous driving and visualization in augmented reality.



3. Recurrent Neural Networks (RNNs)

Recurrent neural network handles sequential data in which the current output is a result of previous inputs by looping over themselves to hold internal state (memory).

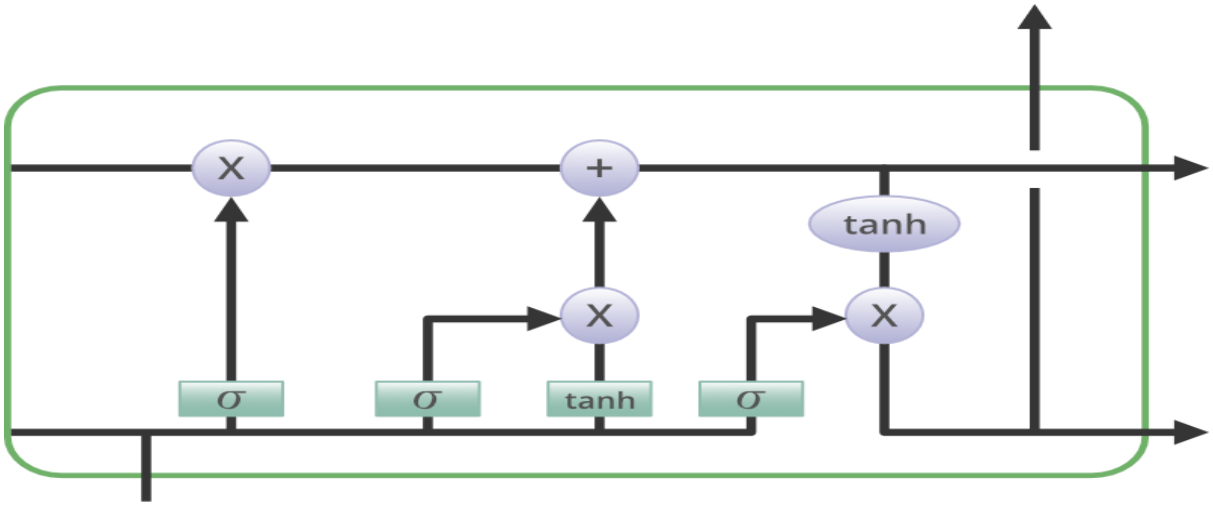
- Architecture: Contains recurrent connections that enable feedback loops for processing sequences.
- Challenges: Problems such as vanishing gradients become apparent since they limit the mode detectors' ability to comprehensively capture interdependence on a long scale.
- Applications: Language translation, open-ended text classification, ones to ones interaction, and time series prediction are its applications.



4. Long Short-Term Memory Networks (LSTMs)

Long Short-Term Memory Networks (LSTMs) are a variant of RNNs. They exhibit memory cells to solve the disappearing gradient issue and keep large ranges of information in their memory.

- Key Features: Capture memory cells in pass information flowing and graduate greediness issue.
- Applications: Value of RNNs is in terms of importing long-term memory into the model e.g., language translation, and time-series forecasting.

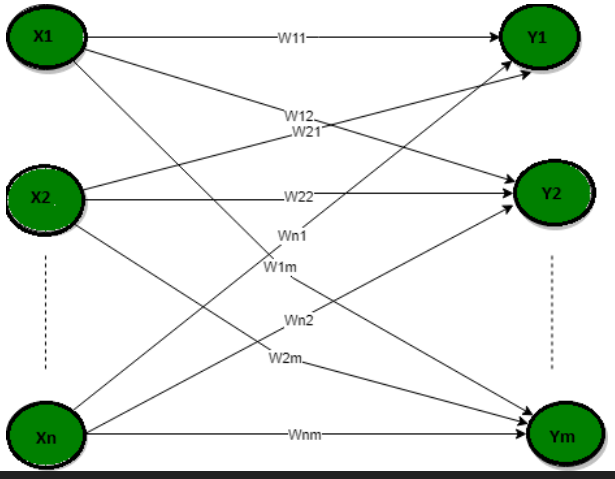


There are four basic types of ANN feedforward network. They are

1) **Single-layer feed-forward network:** In this type of network, we have only two layers input layer and the output layer. The output layer is formed when different weights are applied to input nodes. After this, the neurons collectively give the output layer to compute the output signals.
   A **Single Layer Feed Forward Neural Network** is the simplest type of neural network. It has three main parts:

- **Input Layer**: Receives the input features (data).
- **Output Layer**: Produces the output (predicted value or class).
- **Weights**: Connect each input to the output neuron, with each connection having a specific weight.

  There is **no hidden layer** in a single-layer network. The information moves only in one direction — *forward* — from input to output.



Structure of a Perceptron

The **Perceptron** is the most basic unit of a neural network, specifically a single neuron model for binary classification. Here's how it works:

1. **Inputs** $x_1, x_2, \ldots, x_n$ are fed into the neuron.
2. Each input is multiplied by a **weight** $w_1, w_2, \ldots, w_n$.
3. The weighted inputs are summed together, and a **bias** b is added:

$$z = (w_1x_1 + w_2x_2 + \ldots + w_nx_n) + b$$

4. The sum z is passed through an **activation function** (usually a step function or a sign function) to produce the final output:

$$\text{output} = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

We want to teach a perceptron to learn the **AND logic gate**:

Input x_1	Input x_2	Output
0	0	0
0	1	0
1	0	0
1	1	1

The output is `1` only when both inputs are `1`.

🛠️ Step 1: Initialize

- Start with **random weights**:
  $w_1 = 0.5, w_2 = 0.5$
- Bias $b = -0.7$
- **Activation function**:

$$\text{Output} = \begin{cases} 1 & \text{if } (w_1x_1 + w_2x_2 + b) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

🧮 Step 2: Calculate outputs

Case 1: Inputs (0, 0)

$$z = (0.5)(0) + (0.5)(0) + (-0.7) = -0.7$$

Since $-0.7 < 0$, output = **0** ✅

Case 2: Inputs (0, 1)

$$z = (0.5)(0) + (0.5)(1) + (-0.7) = -0.2$$

Since $-0.2 < 0$, output = **0** ✅

Case 3: Inputs (1, 0)

$$z = (0.5)(1) + (0.5)(0) + (-0.7) = -0.2$$

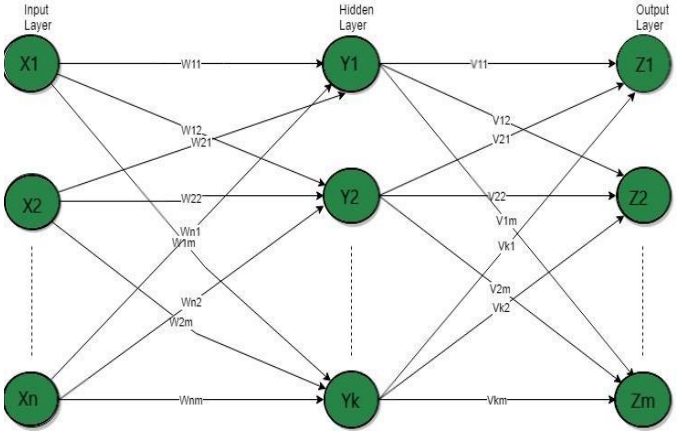
Since $-0.2 < 0$, output = **0** ✅

Case 4: Inputs (1, 1)

$$z = (0.5)(1) + (0.5)(1) + (-0.7) = 0.3$$

Since $0.3 > 0$, output = **1** ✅

2) **Multilayer feed-forward network:** This layer has a hidden layer that is internal to the network and has no direct contact with the external layer. The existence of one or more hidden layers enables the network to be computationally stronger, a feed-forward network because of information flow through the input function, and the intermediate computations used to determine the output.



A **Multilayer Feedforward Neural Network** (also called a Multilayer Perceptron or MLP) is one of the most basic and widely used types of **artificial neural networks**.
In this network:

- Data flows **only in one direction** — from input to output.
- It has **multiple layers**: one input layer, one or more hidden layers, and one output layer.
- **No loops or cycles** are present (unlike recurrent neural networks).

Each layer consists of **neurons** (also called **nodes**) that process the input data and pass it to the next layer.

Structure:

1. **Input Layer**:
    - Takes input features (like pixel values of an image, or characteristics of data).
    - Does not perform any computation, just passes the data to the next layer.
2. **Hidden Layers** (one or more):
    - Each neuron in a hidden layer performs two things:
        - **Weighted Sum**: multiplies inputs by weights and adds bias.
        - **Activation Function**: applies a nonlinear function (like ReLU, sigmoid, tanh) to introduce nonlinearity.
    - These layers allow the network to **learn complex patterns**.
3. **Output Layer**:
    - Produces the final result (like class labels, regression output, etc.).
    - The activation function here depends on the problem (e.g., softmax for classification).

How it works (Step-by-Step):

Step1:ForwardPassData passes from input to output through the hidden layers. Each neuron processes the data and sends it forward.

Step2:LossCalculation

Compare the output of the network with the actual correct output using a loss function (like Mean Squared Error, Cross Entropy).

Step3:BackwardPass(Backpropagation)

The network calculates errors and adjusts the weights using gradient descent to minimize the loss.

Step4:UpdateWeights

Weights are updated slightly in the direction that reduces the loss.

Step5:RepeatRepeat steps 1–4 for many epochs (iterations) until the network learns well.

Example:

Problem: Recognize handwritten digits (0–9) from images.

- **Input Layer**:
  Each image is 28×28 pixels = 784 inputs.
- **Hidden Layers**:
  - First hidden layer: 128 neurons with ReLU activation.
  - Second hidden layer: 64 neurons with ReLU activation.
- **Output Layer**:
  - 10 neurons (one for each digit 0–9) with **softmax** activation (for classification).

Working:

An image of a '5' is fed into the input layer.

It is processed through hidden layers where important features (like edges, curves) are detected.

Finally, the output layer gives probabilities — the highest probability is assigned as the predicted digit.

Back propagation Algorithm for learning in multilayer networks.

Backpropagation is a **learning algorithm** used to train **artificial neural networks**.

It is mainly used to **adjust the weights** of neurons in the network to **minimize the error** between the **predicted output** and the **actual (desired) output**.

- First, **propagate** the input forward through the network to calculate the output.
- Then, **propagate** the error **backward** from the output towards the input layer.
- Use the error to **update** the weights and biases to improve predictions.

Algorithm:

1. Forward Propagation

- Inputs are passed into the network.
- Each neuron processes the inputs, calculates a weighted sum, applies an activation function, and sends the output to the next layer.
- Finally, the network gives an output.

2. Error Calculation

- Compare the network output with the actual target value.
- Calculate the error (difference) using a **loss function** (example: Mean Squared Error).

3. Backward Propagation

- Start from the output layer and move backward.
- Calculate how much each neuron contributed to the final error.
- Use the **chain rule of calculus** to find the gradients (partial derivatives).

4. Update Weights and Biases

- Adjust weights and biases slightly in the direction that reduces the error.
- Use **gradient descent**:

$$w = w - \eta \times \text{gradient}$$

where η is the **learning rate** (a small positive number).

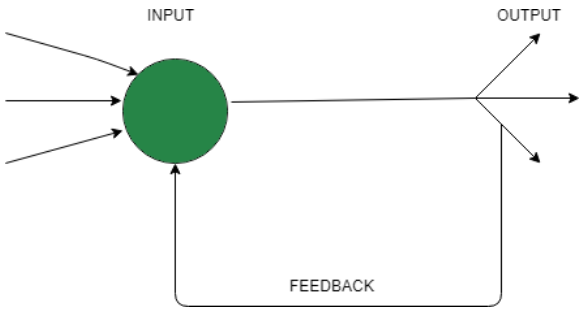
5. Repeat

- Repeat this process for many **iterations (epochs)** until the network performs well (low error).

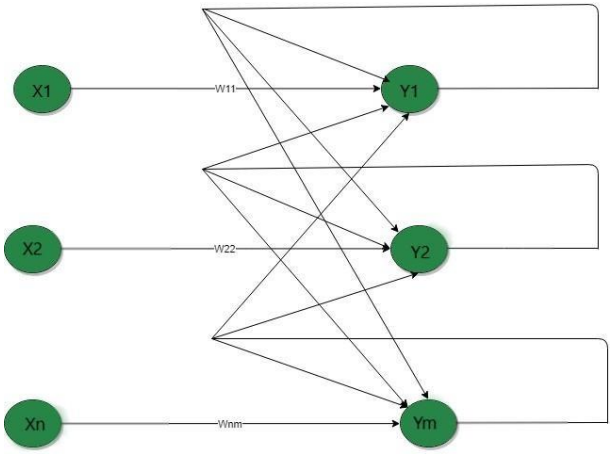
Forward Pass → Calculate Error → Backward Pass → Update Weights → Repeat

3) **Single node with its own feedback:** When outputs can be directed back as inputs to the same layer or preceding layer nodes, then it results in feedback networks. Recurrent networks are

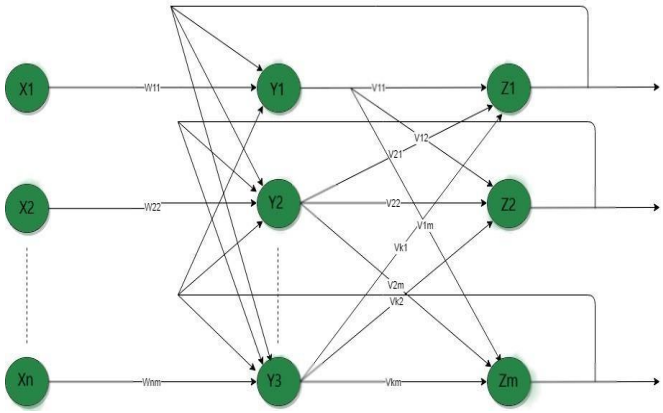
feedback networks with closed loops.



4) **Single-layer recurrent network:** A single-layer network with a feedback connection in which the processing element's output can be directed back to itself or to another processing element or both.



5) **Multilayer recurrent network:** In this type of network, processing element output can be directed to the processing element in the same layer and in the preceding layer forming a multilayer recurrent network.



Advantages of Artificial Neural Network (ANN)

- Parallel processing capability: Artificial neural networks have a numerical value thatcan perform more than one task simultaneously.
- Storing data on the entire network: Data that is used in traditional programming isstored on the whole network, not on a database.

- Capability to work with incomplete knowledge: After ANN training, the information may produce output even with inadequate data.

Disadvantages of Artificial Neural Network:

- Assurance of proper network structure: There is no particular guideline for determining the structure of artificial neural networks. The appropriate network structure is accomplished through experience, trial, and error.
    - Unrecognized behaviour of the network
    - Hardware dependence
    - Difficulty of showing the issue to the network
    - The duration of the network is unknown

Applications of Artificial Neural Networks

They can perform tasks that are easy for a human but difficult for a machine −

1. **Aerospace** − Autopilot aircrafts, aircraft fault detection.
2. **Automotive** − Automobile guidance systems.
3. **Military** − Weapon orientation and steering, target tracking, object discrimination, facial recognition, signal/image identification.
4. **Electronics** − Code sequence prediction, IC chip layout, chip failure analysis, machine vision, voice synthesis.
5. **Financial** − Real estate appraisal, loan advisor, mortgage screening, corporate bond rating, portfolio trading program, corporate financial analysis, currency value prediction, document readers, credit application evaluators.
6. **Industrial** − Manufacturing process control, product design and analysis, quality inspection systems, welding quality analysis, paper quality prediction, chemical product design analysis, dynamic modeling of chemical process systems, machine maintenance analysis, project bidding, planning, and management.
7. **Medical** − Cancer cell analysis, EEG and ECG analysis, prosthetic design, transplant time optimizer.
8. **Speech** − Speech recognition, speech classification, text to speech conversion.
9. **Telecommunications** − Image and data compression, automated information services, real-time spoken language translation.
10. **Transportation** − Truck Brake system diagnosis, vehicle scheduling, routing systems.
11. **Software** − Pattern Recognition in facial recognition, optical character recognition,
12. **Time Series Prediction** − ANNs are used to make predictions on stocks and natural calamities.
13. **Signal Processing** − Neural networks can be trained to process an audio signal and filter it appropriately in the hearing aids.
14. **Control** − ANNs are often used to make steering decisions of physical vehicles.
15. **Anomaly Detection** − As ANNs are expert at recognizing patterns, they can also be trained to generate an output when something unusual occurs that misfits the pattern.

4.8 Support Vector Machines

An SVM works by finding the optimal hyperplane that best separates data points of different classes in a high-dimensional space. The goal is to maximize the margin — the distance between the hyperplane and the nearest data points from each class, called *support vectors*.

Email Spam Detection

- **Problem**: Classify emails as *spam* or *not spam*.
- **HowSVMhelps**:

  An SVM can be trained using features extracted from emails — like the presence of certain keywords ("free", "buy now", "winner"), the number of links, the frequency of capital letters, etc.

The SVM finds the best hyperplane that separates spam emails from legitimate ones based on these features. New incoming emails are then classified by which side of the hyperplane they fall on.

- **Why SVM?**
  - SVMs handle high-dimensional data really well (emails can have hundreds or thousands of features).
  - They perform great even when the number of features is much larger than the number of examples.

APPLICATION

- **Image Classification**
  - Recognizing handwritten digits (like in postal services).
  - Face detection and facial expression recognition.
  - Object recognition in photos.
- **Text and Hypertext Categorization**
  - Spam detection in emails.
  - Sentiment analysis (like determining if a review is positive or negative).
  - Categorizing articles into topics (sports, politics, entertainment, etc.).
- **Bioinformatics**
  - Classifying proteins or genes.
  - Disease diagnosis from genetic data or medical imaging.
- **Handwriting Recognition**
  - Reading and converting handwritten documents into digital text.
- **Speech Recognition**
  - Helping in voice-to-text systems and virtual assistants.
- **Financial Applications**
  - Stock market trend prediction.
  - Credit risk analysis (deciding if a borrower is risky).
- **Anomaly Detection**
  - Detecting fraud in transactions.
  - Finding defects in manufacturing.
- **Medical Diagnosis**
  - Classifying tumors as benign or malignant.
  - Diagnosing diseases from various health parameters.
- **Face Verification**
  - Verifying a person's identity using their face (used in security systems).
- **Intrusion Detection**
  - Identifying malicious activities in a network.

Advantages of SVM:

- **High Accuracy**

  SVMs often deliver great performance, especially on small to medium-sized datasets.
- **Effective in High-Dimensional Spaces**

  Works very well even when the number of features (dimensions) is larger than the number of samples.
- **Memory Efficient**

  Only a subset of training points (the support vectors) are used to define the decision boundary.
- **Versatile**

  Can be used for classification, regression, and even outlier detection.
- **Flexible with Kernels**

  Different kernel functions (linear, polynomial, RBF, sigmoid) allow SVM to adapt to different data complexities.

Disadvantages of SVM:

- **Not Great for Large Datasets**

  Training time can be very slow when the dataset is big (because SVM optimization is computationally intensive).
- **Not Good with Noisy Data**

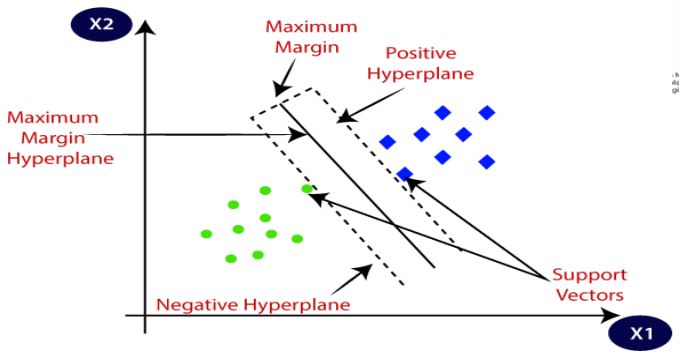
  Especially when classes overlap significantly; SVM tries to find a hard margin which might not exist.
- **Choosing the Right Kernel is Hard**

  Selecting the correct kernel function and tuning its parameters (like C and gamma) can be tricky.
- **Less Interpretable**

  Unlike decision trees, it's hard to interpret how exactly the SVM is making its decisions.
- **Sensitive to Parameter Settings**

  SVMs are sensitive to the choice of regularization parameter (C), kernel type, and kernel parameters.

Hyperplane and Support Vectors in the SVM algorithm:



Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

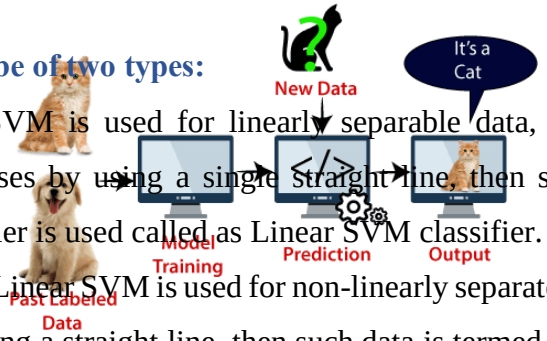
The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then

hyperplane will be a 2-dimension plane.
We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

Example: Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created byusing the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On thebasis of the support vectors, it will classify it as a cat.



Types of SVM:SVM can be of two types:

1.Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can beclassified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

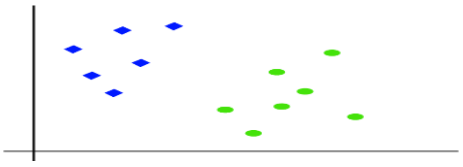
2.**Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if adataset cannot be classified by using a straight line, then such data is termed as non-linear dataand classifier used is called as Non-linear SVM classifier.

How does SVM works?

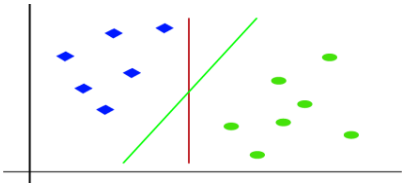
Linear SVM:

The working of the SVM algorithm can be understood by using an example. Suppose we havea dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair (x1, x2) of coordinates in either green or blue. Consider the below image:

How does SVM works

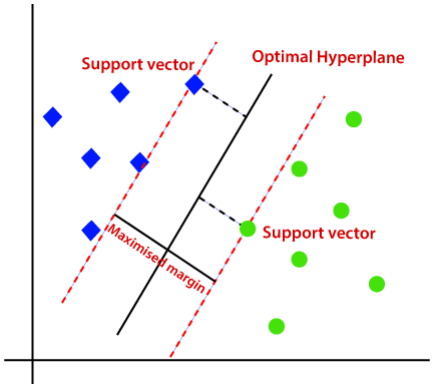


So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



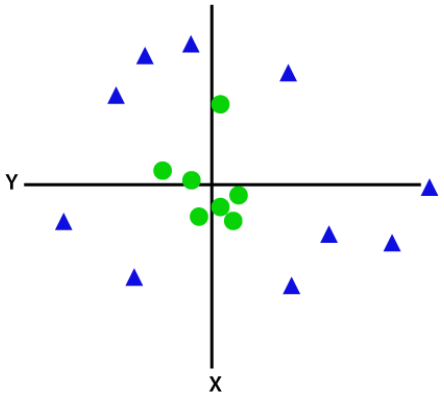
Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These

points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



Non-Linear SVM:

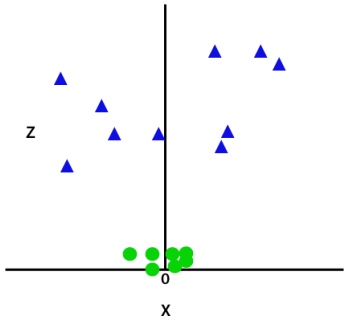
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



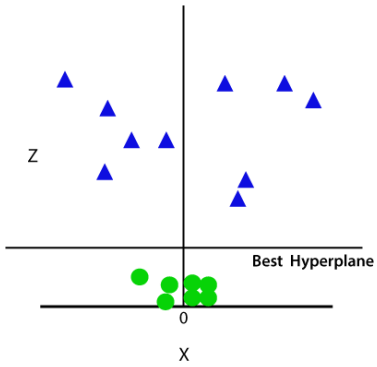
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$$z=x^2+y^2$$

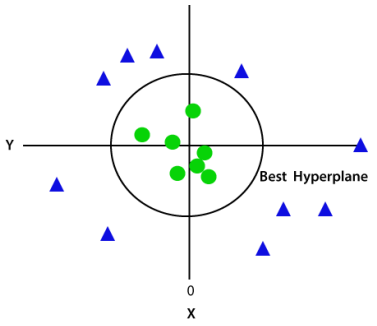
By adding the third dimension, the sample space will become as below image:



So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:



Hence we get a circumference of radius 1 in case of non-linear data

Simple program:

```
# Load the important packages
from sklearn.datasets import load_breast_cancer
import matplotlib.pyplot as plt
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.svm import SVC

# Load the datasets
cancer = load_breast_cancer()
X = cancer.data[:, :2]
y = cancer.target

#Build the model
svm = SVC(kernel="rbf", gamma=0.5, C=1.0)
```

```
# Trained the model
svm.fit(X, y)
# Plot Decision Boundary
DecisionBoundaryDisplay.from_estimator(
        svm,
        X,
        response_method="predict",
        cmap=plt.cm.Spectral,
        alpha=0.8,
        xlabel=cancer.feature_names[0],
        ylabel=cancer.feature_names[1],
    )
# Scatter plot
plt.scatter(X[:, 0], X[:, 1],
            c=y,
            s=20, edgecolors="k")
plt.show()
```

Output: