SYLLABUS

Introduction to Machine Learning and Pre-requisites: Introduction to Machine Learning – Learning Paradigms – PAC learning – Version Spaces – Role of Machine Learning in Artificial Intelligence

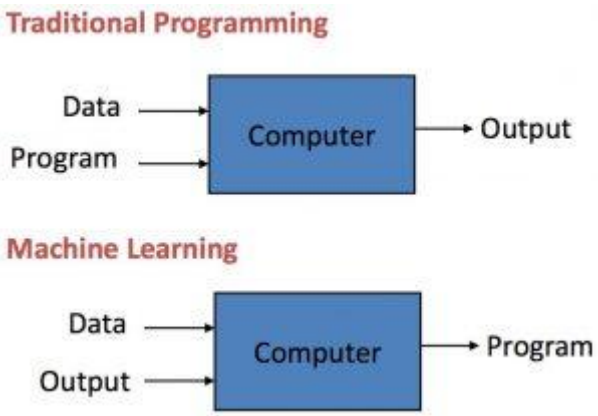
Introduction to Machine Learning and Pre-requisites

Introduction to Machine Learning

Definition 1:
According to **Tom M.Mitchell** - A computer program is said to learn from **experience E** with respect to some class of **tasks T** and **performance** measure P, if its performance at tasks in **T**, as measured by **P**, improves with **experience E**.

Definition 2:
According to **Arthur Samuel** - Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.

Definition 3:
Machine Learning (ML) is a branch of Artificial Intelligence (AI) that focuses on building systems that learn from data to make predictions or decisions without being explicitly programmed.



Why Machine Learning?

Traditional programming uses fixed rules. However, real-world problems like spam detection, image recognition, or stock prediction require models that adapt to data. ML solves this by allowing computers to ***learn patterns*** and ***improve performance*** over time.

Applications of ML

- **Healthcare:** Disease prediction, medical image analysis
- **Agriculture:** Crop yield prediction, disease detection
- **Finance:** Fraud detection, credit scoring
- **Transportation:** Autonomous vehicles, route optimization
- **Marketing:** Recommendation systems, customer targeting

Learning Paradigms

- Machine Learning (ML) is a subfield of Artificial Intelligence (AI) that enables systems to learn from data and improve their performance over time without being explicitly programmed.
- The methods used in ML can be broadly categorized into **learning paradigms**, based on how the model learns from data.
- The major learning paradigms in ML are:

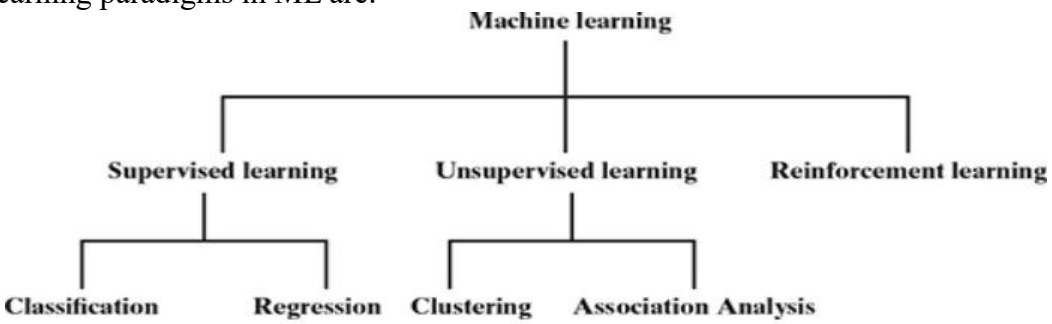


FIG. 1.3 Types of machine learning

1. **Supervised Learning:**

   - **Definition**: This is the most common scenario, where the learner receives a set of **labelled examples as training data** and makes predictions for all unseen points. The machine builds a **predictive model** based on this past information to assign labels or values to unknown data, referred to as test data.
   - **Objective**: To predict a specific outcome variable (e.g., Y) based on a set of input features (e.g., X). Supervised learning primarily focuses on solving **predictive problems**.
   - The 'experience' refers to the past information available to the learner, typically in the form of **digitized human-labelled training sets**. The **quality and size of this data are crucial** for the success of predictions.
   - This learning process is often compared to human **learning under direct expert guidance**, such as infants learning from parents or students from teachers.
   - Common problems addressed by supervised learning include **classification, and regression.**
   - **Classification**:
     - Involves assigning a label, category, or class to a test data point based on the class information imparted by the training data. The target feature is typically categorical.
     - Examples include **spam detection** or predicting whether a **tumour is malignant or benign**.
     - Some algorithms are K-Nearest Neighbors (KNN), Decision Tree, Random Forest, Support Vector Machine (SVM), Naive Bayes etc
   - **Regression**:
     - Aims to predict numerical features that are continuous in nature, such as **real estate or stock prices**, temperature, or sales revenue.
     - It is used to understand the relation between dependent and independent variables.
     - Some algorithms are Linear Regression, Ridge Regression, Lasso Regression, K-Nearest Neighbors (KNN), Decision Tree Regression etc.

2. **Unsupervised Learning**

   - **Definition**: The learner exclusively receives **unlabelled training data** and makes predictions or finds patterns in unseen points. There are no labelled examples available in this setting.
   - **Objective**: The primary goal is **knowledge discovery** rather than prediction. It helps in finding groups or patterns within unknown objects by grouping similar ones together, or by understanding the associations between features. It primarily solves **descriptive problems**.

- Since no labelled examples are provided, it can be difficult to quantitatively evaluate the performance of an unsupervised learner.
- This type of learning is likened to human beings trying to group objects of similar shapes without **explicit guidance** on what each shape is.
- Major classes include **clustering and dimensionality reduction**.
- **Clustering**:
  - This task involves identifying natural groupings or subgroups within a dataset based on the characteristics of the objects themselves.
  - The principle is that objects within a cluster should be very similar to each other but distinctly different from those outside the cluster.
  - Examples include **customer segmentation, Anomaly or Outlier Detection etc.**

- **Association Analysis:**
  - An Association Analysis or Association rule is an unsupervised learning method used to find relationships between different items in a dataset.
  - It determines the set of items that occurs together in the dataset.
  - Association rule makes marketing strategy more effective.
  - Such as people **who buy X item** (suppose a bread) are also tend to **purchase Y** (Butter/Jam) item.
  - A typical example of Association rule is **Market Basket Analysis, Recommendation System** etc.
- **Dimensionality Reduction**:
  - Involves transforming an initial high-dimensional representation of items into a lower-dimensional one while preserving certain properties of the original representation.
  - A common application is preprocessing digital images in computer vision

3. **Reinforcement Learning**

- **Definition**: Reinforcement learning is a computational approach where a learner (agent) actively interacts with an environment to achieve a certain goal, learning what actions to take to **maximize a numerical reward signal**. The agent is not explicitly told which actions to take but must discover them through trial and error.
- It is fundamentally a **closed-loop problem**, where the agent's actions influence its subsequent inputs and rewards over extended time periods. The learning process occurs through a **penalty/reward mechanism**.
- It is considered a **third machine learning paradigm**, distinct from supervised and unsupervised learning, as its goal is to maximize reward rather than finding hidden structure or learning from explicit examples of correct behaviour.
- RL has strong connections to **control theory, optimization, and cognitive sciences**, and is closely related to **optimal control problems**, especially **stochastic optimal control problems formulated as Markov Decision Processes (MDPs)**.
- Commonly applied to tasks such as **self-driving cars**, intelligent robots, and games like chess and backgammon.
- Most RL methods revolve around estimating **value functions**, which are crucial for efficient search in the space of policies. Specific algorithms include **Value Iteration, Policy Iteration, Q-learning, SARSA, and Temporal Difference (TD) algorithms** like TD(0) and TD(λ). **Policy gradient methods** are also used to adjust policy parameters for performance improvement

4. **Semi-Supervised Learning**

- **Definition**: This paradigm involves a learner receiving a training sample that consists of **both labelled and unlabelled data**, and then making predictions for all unseen points.
- It is distinct from active learning, where the algorithm interactively queries a user for labels of specific data points.
- It is particularly useful in settings where unlabelled data is abundant and easily accessible, but obtaining labels is expensive.
- The hope is that the distribution of the unlabelled data can help the learner achieve better performance than relying solely on the limited labelled data.
- Unlabelled data can provide valuable insights into the **manifolds** in which the data is embedded and the overall density structure, guiding the learning process

5. **Self-Supervised Learning**

- **Definition:** Self-supervised learning is a machine learning paradigm where the model learns from raw, unlabelled data by automatically generating pseudo-labels from the data itself, eliminating the need for manual annotation.
- **It is distinct from supervised learning**, where explicit labels are provided by humans, and also from unsupervised learning, where no labels are used at all. In self-supervised learning, the labels are derived from the data through pretext tasks.
- **It is particularly useful in domains like natural language processing, computer vision, and speech recognition**, where large amounts of unlabelled data are available but manually labelling them is costly and time-consuming.
- **The idea is that learning from these pseudo-labels helps the model extract rich, generalizable features**, which can later be fine-tuned on small amounts of labelled data for specific downstream tasks.
- **These pretext tasks might include predicting missing parts of an input, determining whether two views come from the same object, or solving jigsaw puzzles**, allowing the model to learn structure and relationships within the data without external supervision.

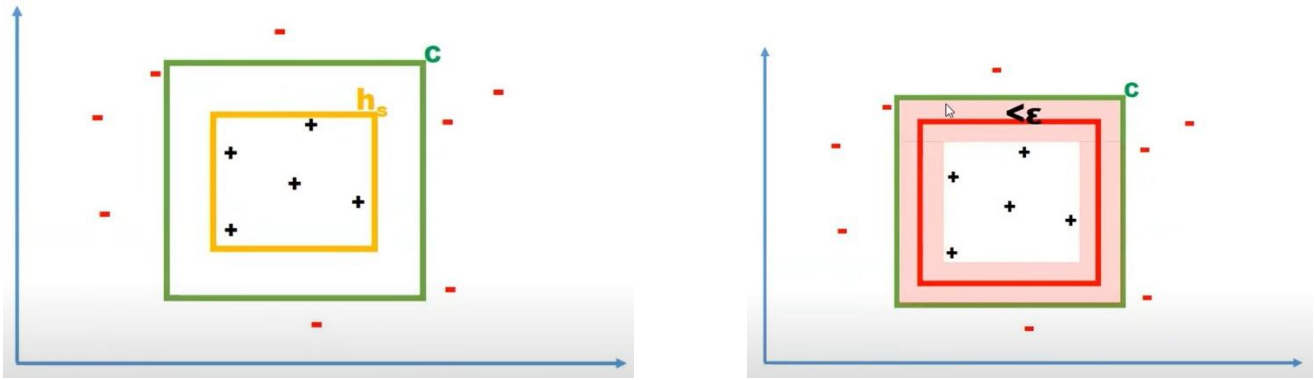
Probably Approximately Correct (PAC) Learning:

- **Definition:** PAC learning is a **theoretical framework** in machine learning that helps us understand whether an algorithm can **learn a concept correctly and efficiently** from data, with **high probability**.
- It addresses key questions about algorithms that learn from examples. It helps to define
  - ✓ **What can be learned efficiently,**
  - ✓ **What is inherently difficult to learn, and**
  - ✓ **How many examples are required for successful learning**.
- The PAC framework is a **computational framework** that considers the cost of computational representations and the time complexity of the learning algorithm.
- **Core Idea (PAC Terminology)**
  - A concept class is considered **PAC-learnable** if an algorithm can return a hypothesis that is **approximately correct** with **high probability**.
  - **"Approximately Correct"** refers to the accuracy, meaning the error of the hypothesis is at most a small value, denoted by **ε (epsilon) > 0**. The true generalization error is the probability that the learned hypothesis misclassifies a new, unseen instance drawn from the same underlying data distribution. So, we want the **error to be ≤ ϵ**.
  - **"Probably Correct"** refers to the confidence, implying that this approximation holds with a high probability, **at least 1 – δ(delta)**, where $\delta > 0$ is a small confidence parameter. The probability of failure is at most δ. So, we want **$P(error(h) \leq \epsilon) \geq 1-\delta$.**
- **The PAC Learning Model**
  - The model assumes a set of all possible examples or instances, X, and a set of all possible labels or target values, Y (often binary, $Y = \{0, 1\}$).
  - Examples are assumed to be independently and identically distributed (i.i.d.) according to some fixed but unknown distribution D.
  - The learning problem involves a hypothesis set (H), which is a fixed set of possible concepts the learner considers. This set may or may not coincide with the true concept class (C) from which the target concept c in C is drawn.
  - The learner receives a sample (S) of labelled examples, and its task is to select a hypothesis h_S in H that has a small generalization error with respect to the target concept c.
- **Key Error Definitions**
  - **Generalization Error** (True Error or Risk: This is the expected error of a hypothesis h with respect to the underlying distribution D and the target concept c. It measures the probability that h(x) will be different from c(x) for a randomly drawn example x sim D. The generalization error is generally not accessible to the learner as D and c are unknown.
  - **Empirical Error** (Empirical Risk): This is the average error of a hypothesis h over the given labelled sample S. It is the fraction of misclassified points in the training sample. For a fixed hypothesis h, the expectation of its empirical error over i.i.d. samples are equal to its generalization error.
- **Examples and Learnability**
  - **Learning Axis-Aligned Rectangles:** The concept class of axis-aligned rectangles is shown to be PAC-learnable. An algorithm that returns the **tightest axis-aligned rectangle** containing positive examples can achieve a generalization error bounded by ε (epsilon) with high probability if the sample size **$O((1/\varepsilon) \log(1/\delta))$**. This demonstrates that even infinite hypothesis sets can be PAC-learnable.
  - **Conjunctions of Boolean Literals:** This class is PAC-learnable with a sample complexity polynomial in **$1/\varepsilon$, $1/\delta$**, and n (number of literals).
  - **Universal Concept Class:** The class of all subsets of Boolean vectors has a sample complexity exponential in n, implying it is generally not PAC-learnable.

Axis-Aligned Rectangles

- The concept class of **Axis-Aligned Rectangles** is a fundamental example used in machine learning to illustrate **PAC-learnability** and other theoretical concepts.
- In this learning problem, the **instances (X) are points in a two-dimensional plane (R 2)**.
- The **concept class (C)** is defined as **the set of all axis-aligned rectangles** lying within this plane. This means each individual concept 'c' is the set of points contained inside a specific axis-aligned rectangle.
- The learning task involves **determining a target axis-aligned rectangle with a small error**, utilizing a labelled training sample. For instance, in the "family car" example, the class of family cars might be represented as a rectangle in a space defined by price and engine power.

- **PAC-Learnability Proof and Algorithm:**
  - The sources explicitly state that **the concept class of axis-aligned rectangles is PAC-learnable**.
  - A simple **PAC-learning algorithm (A)** for this class works as follows: given a labelled sample (S), the algorithm **returns the tightest axis-aligned rectangle (R') that contains all the points labelled with '1'**. This 'tightest rectangle' is denoted as R_S.
  - By its definition, R_S **does not produce any false positives** since all its points are inherently included within the target concept (R). Therefore, any errors made by this algorithm are contained within the target rectangle (R) but outside of R_S.
  - The proof demonstrates that the **sample complexity** required for PAC-learning axis-aligned rectangles is **in O(1/ε log 1/δ)**. This means that to achieve an error rate of at most ε with a confidence of at least 1-δ, a polynomial number of training examples, depending on ε and δ, is sufficient.
  - Equivalently, a **generalization bound** can be stated: with a probability of at least 1-δ, the generalisation error of R_S is bounded as **R(R_S) ≤ (4/m) log(4/δ)**.
  - The proof relies on a specific geometric argument involving four rectangular regions along the sides of R_S, each with a probability mass of at least ε/4.
  - In this specific example, the **hypothesis set (H)** that the algorithm considers **coincides with the concept class (C)**, and notably, its cardinality is infinite.
  - The hypothesis (R_S) returned by this algorithm is **consistent**, meaning it commits no error on the training sample (S).
  - The concept of PAC-learnability of axis-aligned rectangles can be **extended to axis-aligned hyper-rectangles in n-dimensional space** (R n) by a similar proof.
  - Furthermore, this algorithm can **still PAC-learn axis-aligned rectangles even in the presence of noise** where positive labels might be flipped to negative with a certain probability



Version Spaces:

- **Definition**: A **version space** is formed by all hypotheses that are **consistent with a given training set**, meaning they have no error on that set.
- **Consistency**: A hypothesis **h** is considered consistent with the training set **D** if it correctly separates all positive examples from negative ones i.e., **h(x) = c(x)**, meaning it has no misclassification error on the training data. The version space comprises all such consistent hypotheses.
- It is defined by two boundary sets:
  - **The S-set (Most Specific Hypothesis)**: This represents the **tightest (smallest) possible rectangle** that encompasses all positive training examples while excluding all negative ones. No consistent hypotheses can be more specific than those in the **S-set**.
  - **The G-set (Most General Hypothesis)**: This represents the **largest possible rectangle** that includes all positive training examples and none of the negative ones. No consistent hypotheses can be more general than those in the G-set.
- Any hypothesis h belonging to the hypothesis class H that lies **between the S-set and the G-set** (inclusive) is **considered a valid hypothesis, consistent** with the training set, and is thus part of the version space.

$$\mathbf{VS_{H,D} = \{ h \in H \mid consistent(h, D)\}}$$
(Or)
$$\mathbf{VS_{H,D} = \{ h \in H \mid h(x)=y \text{ for all } (x, y)\in D\}}$$
(Or)
$$\mathbf{VS_{H,D}=\{ h \in H \mid \forall(x_i,y_i) \in D , h(x_i)=y_i\}}$$

List then Eliminate Algorithm:

This algorithm builds the version space by enumerating all possible hypotheses and then eliminating the ones inconsistent with the training examples.

Assumptions:

- Hypothesis space **H** is finite.
- Training set **D** contains labeled examples (positive or negative).
- Each hypothesis is a conjunction of constraints over features.

1. **Initialization version space**
   VS ← H (i.e., all hypotheses in the hypothesis space)
2. **For each training example (x, y) in D:**
   Step 1: **For each hypothesis h in VS:**
   Compute prediction: **$h(x_i)$**
   Compare with true label **y_i**
   If **$h(x_i) \neq y_i$:**
   Remove h from VS

This eliminates all hypotheses that are **inconsistent** with the current training example.

Step 3: Repeat Step 2 for all examples in the dataset

Step 4: Return final Version Space (VS)

VS contains only those hypotheses that correctly classify all training examples

Example

Dataset: COVID-19 Symptoms

Sample	Fever	Cough	Label
x1	Yes	Yes	Positive
x2	No	yes	Negative
x3	Yes	No	Positive
x4	No	No	Negative

Initial hypothesis space H: all 9 possible combinations:

[Yes, Yes], [Yes, No], [No, Yes], [No, No], [Yes, ?], [?, Yes], [?, No], [No, ?], [?, ?]

Step 1: Start with all hypotheses in Version Space

VS = H = all 9 hypotheses listed above

Step 2: Process x_1 = (Yes, Yes) → Positive

Keep only hypotheses that predict Positive for (Yes, Yes):

Check each in VS:

[Yes, Yes] - **consistent**
[Yes, No] - inconsistent
[No, Yes] - inconsistent
[No, No] - inconsistent
[Yes, ?] - **consistent**
[?, Yes] - **consistent**
[?, No] - inconsistent
[No, ?] - inconsistent
[?, ?] - **consistent**

VS = {[Yes, Yes], [Yes, ?], [?, Yes], [?, ?]}

Step 3: Process x_2 = (No, Yes) → Negative

Remove any hypothesis that incorrectly predicts Positive:

Check each in VS:

[Yes, Yes] - **consistent**
[Yes, ?] - **consistent**
[?, Yes] - inconsistent
[?, ?] - inconsistent

VS = {[Yes, Yes], [Yes, ?]}

Step 4: Process x_3 = (Yes, No) → Positive

Keep only hypotheses that correctly predict Positive for (Yes, No):

Check:

- [Yes, Yes] - inconsistent
- [Yes, ?] - **consistent**

VS = {[Yes, ?]}

Step 5: Process x_4 = (No, No) → Negative

Eliminate hypotheses that wrongly classify it as Positive.

[Yes, ?] - **consistent**

Final Version Space = {[Yes, ?]}

Final Hypothesis

If a person has **Fever = Yes**, regardless of Cough, then predict **COVID Positive**.

Candidate Elimination Algorithm

- The **candidate elimination algorithm** is a method used to incrementally update the **S-set** and **G-set** as new training instances are presented one by one. This iterative process refines the boundaries of the version space.
- The algorithm's steps are generally as follows:

Input:

Hypothesis space H

Set of training examples D={(x1,y1),(x2,y2),...,(xn,yn)}

Output:

Final **version space** VSH,D, bounded by sets S and G

3. **Initialization**

Let S={ Ø , Ø, ... , Ø }

Let G={ ?, ?, ... ,? }

4. **For each training example (x, y) in D:**

Step 1: If (x, y) is a Positive Example:

a) Remove from G any hypothesis **inconsistent** with x

b) For each hypothesis s in S:

If s does **not cover** x:

Replace s by the **minimal generalizations** of s that **cover x**

Ensure each generalization is **more specific** than some member of G

c) Remove from S any hypothesis **more general** than another hypothesis in S

3. **For each training example (x, y) in D:**

Step 2: **If (x, y) is a Negative Example:**

a) Remove from S any hypothesis **that covers x**

b) For each hypothesis g in G:

If g **does cover** x:

Replace g by its **minimal specializations** that **exclude x**

Each specialization must **still be more general** than some member of S

c) Remove from G any hypothesis **less general** than another in G.

4. Repeat until the S-set and G-set converge (i.e., become identical or define a narrow range of hypotheses), or until no more consistent hypotheses can be found.

- The candidate elimination algorithm is effective when the training data is assumed to be **noise-free** and the **target concept is present within the hypothesis class**. However, adaptations exist to handle instances perturbed by small amounts of noise.

Example

Dataset: COVID-19 Symptoms

Sample	Fever	Cough	Label
x1	Yes	Yes	Positive
x2	Yes	No	Positive
x3	No	Yes	Negative
x4	No	No	Negative

Initial:

- **S =** ['Ø', 'Ø'] → Most specific hypothesis
- **G =** [['?', '?']] → Most general hypothesis

Step 1: Process x1 = ['Yes', 'Yes'] → Positive

- G remains [['?', '?']] (matches the positive example)
- S is updated from ['Ø', 'Ø'] → ['Yes', 'Yes']

After step 1:

- S = ['Yes', 'Yes']
- G = [['?', '?']]

Step 2: Process x2 = ['Yes', 'No'] → Positive

Now filter G to keep only hypotheses that cover ['Yes', 'No']:

- ['?', '?'] matches → keep

Compare with S = ['Yes', 'Yes']

- S[1] = 'Yes' ≠ 'No' → generalize that attribute to '?'
- S becomes ['Yes', '?']

After step 2:

- S = ['Yes', '?']
- G = [['?', '?']]

Step 3: Process x3 = ['No', 'Yes'] → Negative

Compare with S = ['Yes', ?]

- ['Yes', '?'] **match x3**
- S becomes ['Yes', '?']

Now filter G to keep only hypotheses that cover ['No', 'Yes']:

- ['?', '?'] specialize it to **not match x3**
- New G = [['Yes','?'], ['?', 'No']]

After step 2:

- S = ['Yes', '?']
- G = [['Yes', '?']]

Step 4: Process x4 = ['No', 'No'] → Negative

Compare with S = ['Yes', ?]

- ['Yes', '?'] **match x3**
- S becomes ['Yes', '?']

Now filter G to keep only hypotheses that cover ['No', 'No']:

- ['Yes', '?'] **match x3**

After step 2:

- S = ['Yes', '?']
- G = [['Yes', '?']]

Final version spaces

Specific Boundary (S): ['Yes', '?']

General Boundary (G): [['Yes', 'No']]

Both S and G are identical (i.e., G = S)

Final Hypothesis

If a person has **Fever = Yes**, regardless of Cough, then predict **COVID Positive**.

Role of Machine Learning in Artificial Intelligence

- Machine learning (ML) plays a pivotal and ever-expanding role within the field of Artificial Intelligence (AI). It is considered a cornerstone for creating intelligent systems that can learn and adapt to dynamic environments.
- Here's a breakdown of the role of Machine Learning in Artificial Intelligence:

a) **Enabling Adaptability and Intelligence**

- At its core, **machine learning involves computational methods that use "experience" (past data) to improve performance or make accurate predictions**. This capability is crucial for an intelligent system, especially when operating in changing environments, as it allows the system to learn and adapt without requiring explicit programming for every conceivable situation.
- Early pioneers like Alan Turing, Arthur Samuel, and Frank Rosenblatt laid the groundwork, proposing that machines could "learn" and become artificially intelligent through processes like trial-and-error.

b) **Solving Complex, Data-Driven Problems**

- ML algorithms combine fundamental concepts from computer science with ideas from statistics, probability, and optimisation. They are designed to efficiently detect patterns in large datasets and then utilise these patterns to make future predictions or inform decisions. This application of ML to large databases is often referred to as **data mining**.
- ML offers solutions to problems that are inherently difficult or impossible to program using traditional methods due to the vastness and variability of data, such as **facial recognition, speech recognition, and robotics**.

c) **Key Machine Learning Paradigms Contributing to AI**

- **Supervised Learning:** This paradigm involves machines learning from "human-labelled training sets" to make predictions on new, unseen data. It is widely used for tasks like **text or document classification (e.g., spam detection), medical diagnosis, fraud detection, and predicting real estate or stock prices**. The quality and size of the training data are critical to the success of predictions made by the learner.
- **Unsupervised Learning:** Unlike supervised learning, this approach focuses on finding hidden structures or patterns within unlabelled data without prior training. Examples include **customer segmentation and recommendation systems**. Unsupervised learning, often combined with supervised learning, is used in modern AI innovations such as **chatbots and self-driven cars**.
- **Reinforcement Learning (RL):** This is a distinct ML paradigm where an "agent" learns to take actions in an environment to maximise a numerical reward signal through "trial-and-error". RL is gaining significant attention in both industry and academia for applications such as **autonomous vehicle control (e.g., self-driving cars) and game playing (e.g., AlphaGo beating human Go players)**. The field of reinforcement learning has strong connections to artificial intelligence, machine learning, and neural networks.

d) **Advancing AI Capabilities**

- **Machine learning success has reshaped AI thinking**, showing that large datasets, computing power, and simple algorithms can drive intelligent behaviour—without needing entirely new computational paradigms.
- **ML bridges multiple disciplines** like statistics, optimization, information theory, and game theory, establishing it as a core area in computer science and enabling innovations like Waymo, AlphaGo, and Google Brain.