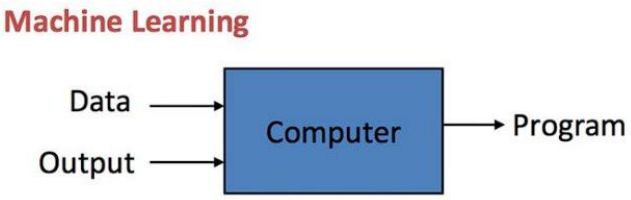
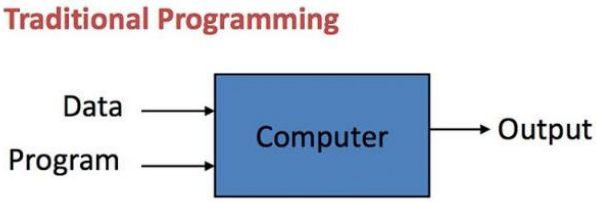
Machine Learning	Traditional Programming	Artificial Intelligence
A subset of AI focusing on creating algorithms that learn from data and make predictions.	Writing rule-based, deterministic code based on specific problem statements.	Technology that enables machines to perform tasks that typically require human intelligence.
Data-driven, learns from historical data to predict future outcomes.	Rule-based and deterministic, relies on explicit instructions from developers.	Uses a mix of data-driven techniques and predefined rules, incorporating ML, deep learning, and traditional programming.
Capable of finding patterns and insights in large datasets, learning and improving over time.	Lacks self-learning capabilities; output is directly tied to input and predefined rules.	Adapts and evolves to perform complex tasks with high accuracy, often exceeding human capabilities in specific domains.
Used in predictive analytics, autonomous vehicles, chatbots, and other AI-based applications.	Used to build applications with specific functionalities like software tools and systems.	Broad applications including natural language processing, computer vision, robotics, and more.
Dependent on the quality and diversity of data. Can perform poorly if data is not representative.	Dependent on the intelligence and foresight of developers. Limited to known scenarios.	Combines the strengths of both ML and traditional programming to tackle complex, multi-faceted problems.





Machine learning.

Machine learning is the branch of Artificial Intelligence that focuses on developing models and algorithms that let computers learn from data and improve from previous experience without being explicitly programmed for every task. In simple words, ML teaches the systems to think and understand like humans by learning from the data.

Types of Machine Learning

1. Supervised Machine Learning
2. Unsupervised Machine Learning
3. Reinforcement Learning

1. Supervised Machine Learning

Supervised learning is defined as when a model gets trained on a "Labelled Dataset". Labelled datasets have both input and output parameters. In Supervised Learning algorithms learn to map points between inputs and correct outputs. It has both training and validation datasets labelled.

- **Examples:**
  - **Email Spam Detection** (Spam or Not Spam)
  - **House Price Prediction** based on area, location, etc.
- **Algorithms:** Linear Regression, Logistic Regression, Decision Trees, Support Vector Machines (SVM)

2. Unsupervised Machine Learning

Unsupervised Learning Unsupervised learning is a type of machine learning technique in which an algorithm discovers patterns and relationships using unlabelled data. Unlike supervised learning, unsupervised learning doesn't involve providing the algorithm with labelled target outputs. The primary goal of Unsupervised learning is often to discover hidden patterns, similarities, or clusters within the data, which can then be used for various purposes, such as data exploration, visualization, dimensionality reduction, and more.

- **Examples:**
  - **Customer Segmentation** in marketing
  - **Anomaly Detection** in network traffic
- **Algorithms:**
  - K-Means Clustering, Hierarchical Clustering, Principal Component Analysis (PCA)

3. Reinforcement Machine Learning

Reinforcement machine learning algorithm is a learning method that interacts with the environment by producing actions and discovering errors. **Trial, error, and delay** are the most relevant characteristics of reinforcement learning. In this technique, the model keeps on increasing its performance using Reward Feedback to learn the behaviour or pattern. These algorithms are specific to a particular problem e.g. Google Self Driving car, AlphaGo where a bot competes with humans and even itself to get better and better performers in Go Game. Each time we feed in data, they learn and add the data to their knowledge which is training data. So, the more it learns the better it gets trained and hence experienced.

- **Examples:**
  - **Game-playing AI** (e.g., AlphaGo)
  - **Robotics** for walking or path planning
- **Key Concepts:**
  - Agent, Environment, Action, Reward, Policy

Aspect	Supervised Learning	Unsupervised Learning
Input Data	Uses labeled data (input features + corresponding outputs).	Uses unlabeled data (only input features, no outputs).
Goal	Predicts outcomes or classifies data based on known labels.	Discovers hidden patterns, structures, or groupings in data.
Computational Complexity	Less complex, as the model learns from labeled data with clear guidance.	More complex, as the model must find patterns without any guidance.
Types	Two types : **Classification** (for discrete outputs) or **regression** (for continuous outputs).	C**lustering** and **association**
Testing the Model	Model can be tested and evaluated using labeled test data.	Cannot be tested in the traditional sense, as there are no labels.

Reinforcement Machine Learning

Reinforcement machine learning algorithm is a learning method that interacts with the environment by producing actions and discovering errors. **Trial, error, and delay** are the most relevant characteristics of reinforcement learning. In this technique, the model keeps on increasing its performance using Reward Feedback to learn the behaviour or pattern. These algorithms are specific to a particular problem e.g. Google Self Driving car, AlphaGo where a bot competes with humans and even itself to get better and better performers in Go Game. Each time we feed in data, they learn and add the data to their knowledge which is training data. So, the more it learns the better it gets trained and hence experienced.

Example of Reinforcement Learning:

🕹️ *Game Playing – Example: Chess AI or AlphaGo*

- **Agent**: The AI playing the game.
- **Environment**: The game board and rules.
- **Actions**: Moving pieces on the board.
- **Reward**: +1 for winning, 0 for a draw, -1 for losing.
- **Learning Goal**: The AI learns which moves lead to victory by playing many games.

Other Real-Life Examples:

1. **Self-Driving Cars**
   - Agent learns to drive by interacting with the simulated or real environment, getting rewards for safe driving and penalties for collisions.
2. **Robotics**
   - A robot learns to walk by trying different movements and being rewarded for stable walking.
3. **Dynamic Pricing Systems**
   - Online platforms adjust prices based on user interaction and sales data to maximize profit.

Feature	Machine Learning	Deep Learning
Data Dependency	Can work effectively with smaller datasets. Performance may plateau after a certain point, even with more data.	Requires large amounts of data for effective learning and performance improvement. Performance generally increases with more data.
Feature Extraction	Requires manual feature extraction, where domain expertise is needed to identify and select relevant features.	Automatically learns features from the data, eliminating the need for manual feature extraction.
Complexity	Algorithms are generally simpler and designed for specific tasks.	Models are more complex, mimicking the human brain's neural network, and can handle larger and more diverse datasets.
Computational Power	Requires less computational power and resources.	Requires high computational power and resources, often utilizing GPUs for training.
Task Suitability	Suitable for structured and simpler tasks, such as fraud detection and recommender systems.	Ideal for complex tasks involving unstructured data, such as image and speech recognition, natural language processing, and autonomous driving.
Intervention	Often requires human intervention to correct errors and improve outcomes.	Can improve outcomes through repetition without human intervention.
Applications	Image recognition, speech recognition, recommender systems, fraud detection, medical diagnosis, stock market trading.	Healthcare, personalized marketing, financial fraud detection, natural language processing, autonomous vehicles, facial recognition.
Accuracy	Accuracy depends on the algorithm and the quality of the data.	Can achieve high levels of accuracy in tasks like image and speech recognition due to automatic feature extraction.
Model Training	Involves feeding the algorithm with labeled data and adjusting parameters to minimize the difference between predictions and actual values.	Similar to machine learning, but often involves more complex architectures and optimization techniques.

What is Data Normalization?

Data normalization is a way to **prepare and adjust data** before using it in machine learning. It involves **changing the values of features (columns)** in a dataset so they all follow a **similar scale** or range — usually **between 0 and 1**, or having a **mean of 0 and standard deviation of 1**.

This is especially useful when features have **different units or sizes** (like height in cm and income in lakhs). Normalization helps to **make all features comparable**, so the model doesn't get biased toward features with larger values.

Why is Normalization Important?

- It ensures that **all features contribute equally** during training.
- It helps **speed up** training for algorithms that use optimization (like gradient descent).
- It improves performance in **distance-based models** like **k-Nearest Neighbours (k-NN)**.
- It reduces sensitivity in models like **Support Vector Machines (SVMs)** and **Neural Networks**.
- It allows regularization methods (like **L1** and **L2**) to work properly.

Without normalization, features with large values could **dominate the learning process**, making the model less accurate.

Common Normalization Methods:

1. **Min-Max Scaling:**
   - Scales values to a range between 0 and 1.
2. **Z-Score Normalization (Standardization):**
   - Adjusts values to have a **mean of 0** and **standard deviation of 1**.

In Short:

Normalization is key to building **accurate, fair, and efficient** machine learning models, especially when your data includes features with **different ranges or units**.

Aspect	Precision	Recall
Definition	Measures how many of the predicted positives are actually correct	Measures how many of the actual positives were correctly predicted
Formula	`TP / (TP + FP)`	`TP / (TP + FN)`
Focus	Correctness of positive predictions	Coverage of actual positive cases
Also Called	Positive Predictive Value	Sensitivity or True Positive Rate
When to Prioritize	When **false positives** are more harmful	When **false negatives** are more harmful
Example Use Case	Spam detection: Don't mark important emails as spam	Disease detection: Don't miss identifying sick patients
High Value Indicates	Very few incorrect positive predictions	Very few missed actual positives

What is a Confusion Matrix?

A **confusion matrix** is a table that shows how well a classification model is working.
It compares what the model **predicted** with what the **actual result** was.

📊 Confusion Matrix Table

	Predicted: Yes	Predicted: No
Actual: Yes	True Positive (TP)	False Negative (FN)
Actual: No	False Positive (FP)	True Negative (TN)

🔍 Meanings:

- **TP (True Positive):** Model said "Yes" and it was really "Yes"
- **FP (False Positive):** Model said "Yes" but it was really "No"
- **FN (False Negative):** Model said "No" but it was really "Yes"
- **TN (True Negative):** Model said "No" and it was really "No"

📐 Simple Formulas:

Metric	Formula	What It Tells
Accuracy	(TP + TN) / (TP + TN + FP + FN)	Overall, how many predictions were correct
Precision	TP / (TP + FP)	Out of predicted "Yes", how many were right
Recall	TP / (TP + FN)	Out of actual "Yes", how many were found
F1 Score	2 × (Precision × Recall) / (Precision + Recall)	Balance between Precision and Recall
Specificity	TN / (TN + FP)	Out of actual "No", how many were predicted correctly
False Positive Rate	FP / (FP + TN)	Wrongly predicted "Yes" from "No" cases

What is the difference between L1 and L2 regularization? What is their significance?

L1 and L2 regularization are techniques used to prevent overfitting in machine learning models by adding a penalty term to the loss function. The key difference lies in how they calculate this penalty:

- **L1 Regularization (LASSO):**

Adds the sum of the absolute values of the model's coefficients to the loss function. This encourages sparsity, meaning it tends to shrink less important feature coefficients to zero, effectively performing feature selection.

- **L2 Regularization (Ridge):**

Adds the sum of the squares of the model's coefficients to the loss function. This encourages smaller, more evenly distributed weights, preventing any single feature from dominating the model.
Significance

- **L1 Regularization:**
    - **Feature Selection:** Useful when dealing with high-dimensional datasets with many irrelevant features.
    - **Model Interpretability:** By setting some coefficients to zero, it simplifies the model and makes it easier to understand which features are most important.
- **L2 Regularization:**
    - **Handles Multicollinearity:** Effective in situations where features are highly correlated, as it shrinks the coefficients of correlated features together.
    - **Model Stability:** Leads to more stable and generalizable models by preventing extreme coefficient values.
    - **Improved Accuracy:** By reducing overfitting, it often results in better predictive performance on unseen data.

❓ Is Accuracy Always a Good Metric to Measure Classification Model Performance?

No, accuracy is not always a good metric, especially when the data is **imbalanced**.

☑ When Accuracy is Good:

- When the dataset has **balanced classes** (i.e., equal number of positive and negative cases).
- When **all types of errors** (false positives and false negatives) are **equally important**.

❌ When Accuracy is Misleading:

- In **imbalanced datasets**, where one class is much more frequent than the other.

🔍 Example of Misleading Accuracy:

Suppose you're predicting if a customer has a rare disease:

- 95 people are healthy (negative)
- 5 people are sick (positive)

If a model predicts **everyone is healthy**:

- ☑ Accuracy = 95/100 = **95%**
- ❌ But it missed all 5 sick people! (Recall = 0%)

➡ This means the model is **useless** for finding the disease, even though the accuracy looks high.

☑ Better Metrics in Such Cases:

Metric	Use When
Precision	You care about **how many positive predictions are correct**
Recall	You care about **finding all actual positives**
F1 Score	You want a balance between precision and recall
AUC-ROC	You want to measure model performance across thresholds

🧠 Conclusion:

Accuracy is useful only when the data is balanced and all errors are equally important. In most real-world problems, use **Precision, Recall, F1-Score**, or **AUC-ROC** for better evaluation.

🎯 What is the Purpose of Splitting a Dataset into Training and Validation Data?

Splitting a dataset into **training** and **validation** parts helps in **building** and **evaluating** a machine learning model effectively.

☑ Main Purpose:

Part	Purpose
Training Data	Used to **teach the model** – it learns patterns, relationships, and rules from this data.
Validation Data	Used to **evaluate the model's performance** during training. Helps check if the model is generalizing well (not memorizing).

🔍 Why is this important?

- **Prevents Overfitting** – If we only train and test on the same data, the model may **memorize** the answers and **fail on new data**.
- **Helps with Model Tuning** – Validation data is used to **tune hyperparameters** (like learning rate, depth, etc.).
- **Gives a Realistic Performance Estimate** – It shows how well the model may perform on **unseen data**.

🧠 In Simple Terms:

Splitting data helps make sure the model **learns properly** and performs well on **new, real-**

🤖 What Are the Assumptions Behind the K-Means Algorithm?

The **K-Means** clustering algorithm has several built-in assumptions that affect how well it works. Understanding these assumptions is important to know **when** and **how** to use it properly.

📌 Assumptions of K-Means:

Assumption	Explanation	Effect on Results
1. **Clusters are spherical (circular/round)**	K-means assumes clusters are shaped like circles (or spheres in higher dimensions).	If clusters are irregular shapes (like crescent or elongated), K-means may group badly.
2. **Clusters have similar size (variance)**	All clusters are expected to be about the same size.	If some clusters are big and others are small, K-means may split or merge them incorrectly.
3. **Equal density of points in each cluster**	The points in each cluster are assumed to be evenly spread out.	Clusters with varying density may confuse the algorithm.
4. **Features are continuous and numeric**	Works best with numerical data.	Categorical or non-numeric features need to be converted or avoided.
5. **Distance-based similarity** (Euclidean distance)	It assumes that closer points belong to the same cluster.	If the true similarity isn't based on distance, K-means may give wrong results.
6. **The number of clusters (k) is known beforehand**	You must choose how many clusters you want.	If the chosen value of k is wrong, the results may be misleading.

🔍 How These Assumptions Affect Results:

- If the **data doesn't match these assumptions**, K-means might:
  - Group unrelated points together,
  - Split a single cluster into multiple parts,
  - Merge distinct clusters into one,
  - Or place centroids in the wrong spots.

🧠 In Simple Terms:

K-Means works best when your data has **round, equally-sized, and equally-dense clusters**. If not, it may give poor or misleading results.

The **difference between K-Means and (KNN)**

Feature	K-Means Clustering	K-Nearest Neighbors (KNN)
Type	Unsupervised learning	Supervised learning
Purpose	Clustering: Grouping data into k clusters	Classification or regression based on neighbors
Labeled Data Needed	✗ No (works on unlabeled data)	☑ Yes (needs labeled training data)
How it Works	Iteratively assigns data to nearest cluster center and updates centers	Finds k nearest labeled points and predicts based on majority vote (classification) or average (regression)
Output	Cluster labels (e.g., Cluster 1, Cluster 2...)	Class label (e.g., "Cat", "Dog", etc.) or value
Distance Metric	Typically Euclidean distance	Typically Euclidean distance
Use Case Example	Customer segmentation, document clustering	Spam detection, handwriting recognition
Training Phase	Involves optimization (iterative centroid update)	No training; lazy learner (computation at query time)