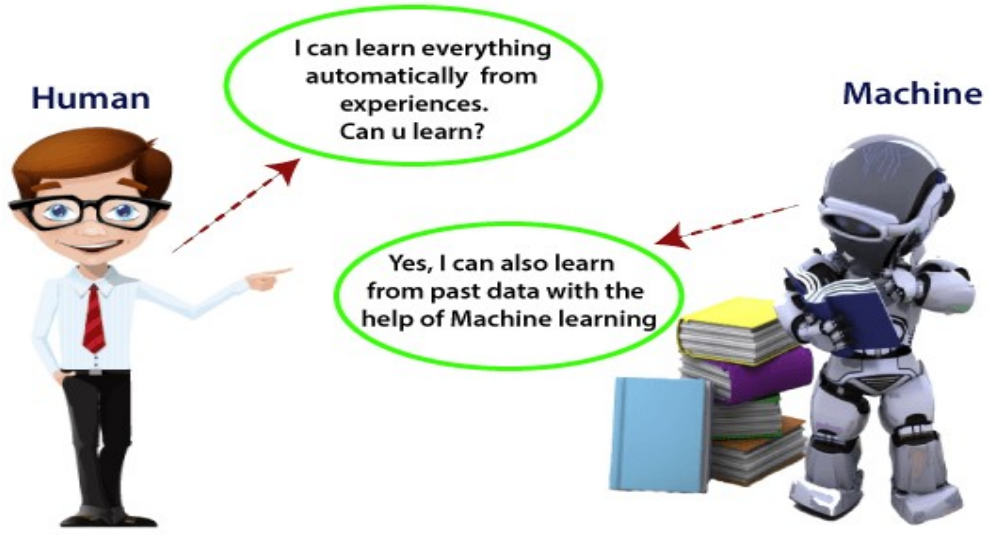
MODULE 2.1 – MACHINE LEARNING



1. What is Machine Learning?

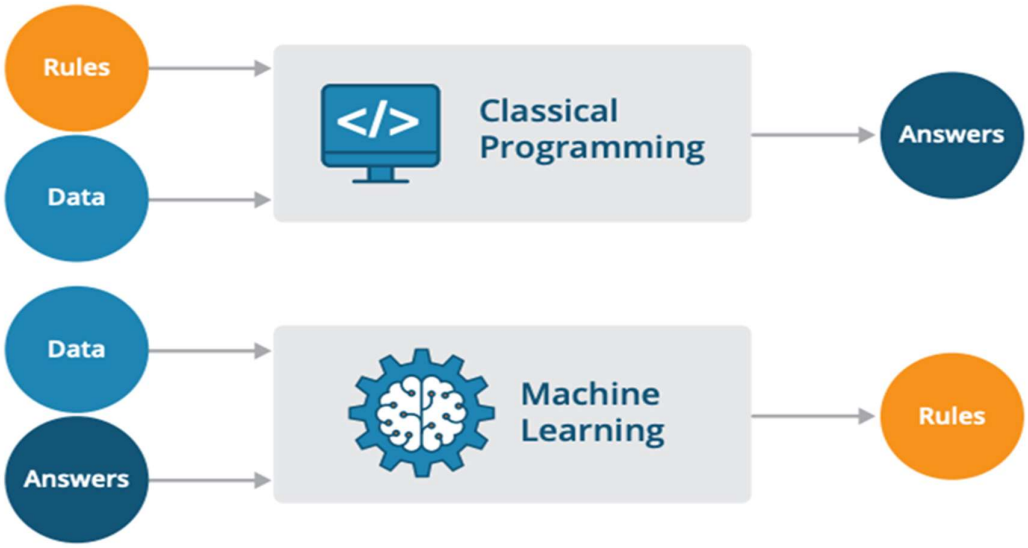
1.1 Intuition

Suppose you are a teacher.
A student comes and asks: 'Sir, new flat ka price kitna hoga?'
You don't have a fixed formula, but you've seen many flats:

Area 500 sqft → 30L
Area 800 sqft → 50L
Area 1000 sqft → 65L

After seeing enough examples, your brain automatically learns a pattern like:
'💭 Bigger area ⇒ generally more price.'

TRADITIONAL PROGRAMMING VS ML:



MACHINE LEARNING IS EXACTLY THIS.
INSTEAD OF US WRITING EXPLICIT RULES, WE SHOW THE COMPUTER LOTS OF EXAMPLES, AND IT LEARNS THE PATTERN BY ITSELF.

"ML IS ABOUT LEARNING PATTERNS FROM DATA AND USING THOSE PATTERNS TO MAKE PREDICTIONS OR DECISIONS."

1.2 A WORKING DEFINITION

A COMPUTER PROGRAM IS SAID TO LEARN FROM EXPERIENCE E WITH RESPECT TO SOME TASK T AND PERFORMANCE MEASURE P, IF ITS PERFORMANCE ON T, MEASURED BY P, IMPROVES WITH EXPERIENCE E.

EXAMPLE (SPAM DETECTION):

- TASK T: CLASSIFY EMAIL AS SPAM OR NOT SPAM.
- EXPERIENCE E: MANY LABELED EMAILS (SPAM/NOT SPAM).
- PERFORMANCE P: ACCURACY ON A TEST SET.

AS IT SEES MORE LABELED EMAILS, ACCURACY IMPROVES → IT IS "LEARNING".

1.3 Math View

Let's define things properly.

input/feature : x

output/Label : y

Training data $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$

We choose a model $f_\theta(x)$ with parameters θ (theta might be weights of a linear model or a deep neural network).

Our goal:

Find parameters such that the model's predictions are close to true labels on training data and generalise well to new unseen data.

We define a loss function $L(\hat{y}, y)$ for example:

- Regression: Mean Squared Error (MSE)

$$L(\hat{y}, y) = (\hat{y} - y)^2$$

- Classification: Cross-entropy loss.

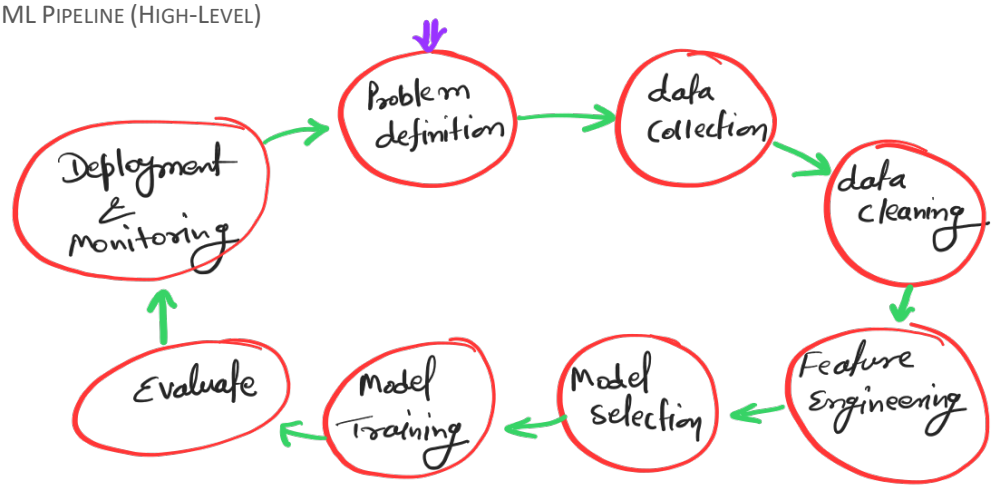
Then the ML training problem is:

$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} L(f_\theta(x_i), y_i)$$

This is the core optimization view of ML:

Find parameters that minimise average loss on training data.

1.4 Basic ML Pipeline (High-Level)



1. Problem definition
    - What do we want to predict? (price, click, disease, etc.)
2. Collect data
    - Get examples from the real world.
3. Preprocess / Clean / Feature Engineering
    - Handle missing values, scaling, encoding categories, etc.
4. Choose model
    - Linear model, tree, SVM, neural network, etc.
5. Train model
    - Use optimisation (like gradient descent) to find parameters.
6. Evaluate
    - Check performance on unseen test/validation data.
7. Deploy & Monitor
    - Use in production, monitor drift, retrain when needed.

We'll go deep into each step later; here just give the big picture.

1.5 Tiny Numerical Example (Regression Intuition)

Simple one-feature linear regression:

We assume relationship: $\hat{y} = f_\theta(x) = wx + b$

Training data :

Area (x)	Price (y)
500	30
800	50
1000	65

find w, b so that $y = wx + b$ roughly fits these points

Loss on 3 points :

$$L(w,b) = \frac{1}{3} \sum_{i=1}^{3} (\hat{y}_i - y_i)^2 = \frac{1}{3} \sum_{i=1}^{3} (wx_i + b - y_i)^2$$

Gradient descent will adjust w, b to reduce this loss.

"We start with random w, b compute error, and keep adjusting them to reduce error – that's training a model."

1.6 Simple Python Example (Regression)

```python
import numpy as np
from sklearn.linear_model import LinearRegression

# Toy data: area (sq ft)
X = np.array([[500],
              [800],
              [1000],
              [1200],
              [1500]], dtype=float)

# Prices in lakh
y = np.array([30, 50, 65, 80, 100], dtype=float)

# Create and train model
model = LinearRegression()
model.fit(X, y)

print("Weight (w):", model.coef_[0])
print("Bias (b):", model.intercept_)

# Predict for a new flat
new_area = np.array([[1100]])
pred_price = model.predict(new_area)
print("Predicted price for 1100 sq ft:", pred_price[0], "lakh")
```

"This is a simple ML model in action: it learned patterns from data and now can predict for new inputs."

---

2. Types of Learning: Supervised, Unsupervised & Reinforcement

Now we classify ML problems based on what kind of feedback we get from data.

- Supervised Learning → We get correct answers (labels).
- Unsupervised Learning → No labels, only raw data.
- Reinforcement Learning → Feedback in form of reward over time.

2.1 Supervised Learning

2.1.1 Intuition

Supervised learning is like learning from a teacher who gives you questions and the correct answers. You practise, see where you are wrong, and slowly improve.

Here we have:

- Input x
- Output/label y
- A dataset of (x_i, y_i)

Goal: Learn a function $f_\theta(x)$ that maps inputs to outputs.

We have training data: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$

We choose model $f_\theta(x)$.
We define an appropriate loss $L(f_\theta(x_i), y_i)$.
We minimise empirical risk:

$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} L(f_\theta(x_i), y_i)$$

Two Main Types:

1. Regression – output is continuous

   Example:

   - Predict house price, temperature, sales quantity.

Loss (typical): MSE

$$L = (f_\theta(x_i) - y_i)^2$$

2. Classification – output is discrete category

   Example:

   - Spam vs Not spam
   - Disease A vs B vs C

Output: probabilities $\hat{p}_k = P(y=k|x)$, then take argmax.

Loss : cross-entropy.

2.1.3 Real-Life Examples & Use Cases

- Email spam detection (binary classification).
- Credit card fraud detection (binary classification).
- Handwritten digit recognition (multi-class classification).
- House price prediction (regression).
- Predicting number of likes on a post (regression).
- Predicting if a customer will churn (binary classification).

2.1.4 Python Example – Supervised Classification

Let's do a small classification with scikit-learn.

```python
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Create synthetic 2D data with 2 classes
X, y = make_blobs(n_samples=500, centers=2, random_state=42, cluster_std=2.0)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model
clf = LogisticRegression()
clf.fit(X_train, y_train)

# Evaluate
y_pred = clf.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print("Test Accuracy:", acc)

# Predict a new point
new_point = [[0.5, 3.0]]
print("Predicted class for", new_point, ":", clf.predict(new_point)[0])
```

Explanation:

- make_blobs creates 2 clusters in 2D.
- Logistic regression learns a decision boundary.
- Accuracy shows how well it learned patterns.

2.2 Unsupervised Learning

2.2.1 Intuition

Unsupervised learning is like being dropped in a new city without a guide.
No one tells you which area is 'student area', 'posh area', 'market area', but from observations you start seeing patterns yourself.

Similarly, in unsupervised learning, we only have inputs x, No labels.

The goal is to discover hidden structure in the data.

Here we have data:

$$D = \{x_1, x_2, \dots, x_n\}$$

; No y_i

Typical goals:

- Group similar data points (clustering).
- Reduce dimensionality while preserving structure (PCA).
- Find anomalies (points that don't belong to any cluster).

2.2.2 Clustering – Example & Math (K-Means)

Clustering = grouping similar points together.

Example: Customer segmentation

- Data: each customer's attributes (age, income, spending score).
- Goal: automatically discover groups like "high income high spending", "low income low spending", etc.

K-Means is a popular clustering algorithm.

We choose number of clusters K.
We want to find cluster centers (centroids) $\mu_1, \mu_2, \dots, \mu_K$.

Objective:

$$\min_{\mu_1, \dots, \mu_K} \sum_{i=1}^{n} \min_{k \in \{1, \dots, K\}} \|x_i - \mu_k\|^2$$

Intuition:

"Assign each point to nearest centroid, update centroid as mean of its points, repeat until assignments don't change much."

2.2.3 Real-Life Use Cases

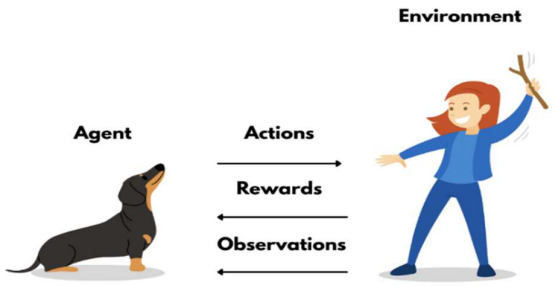
- Market/customer segmentation (clustering).
- Image Organization (e.g., Google Photos)
- Anomaly detection in transactions (outlier detection).
- Grouping similar users/items in recommendation systems.

2.2.5 Python Example – K-Means

```python
import numpy as np
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans

# Create synthetic data with 3 clusters
X, _ = make_blobs(n_samples=500, centers=3, random_state=42, cluster_std=1.5)

# K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
cluster_labels = kmeans.fit_predict(X)

print("Cluster centers:\n", kmeans.cluster_centers_[:3])
```

"See, we never gave labels. The algorithm itself discovered 3 groups from the data – that's unsupervised learning."

2.3 Reinforcement Learning (RL)



This is a bit different from supervised/unsupervised.

2.3.1 Intuition

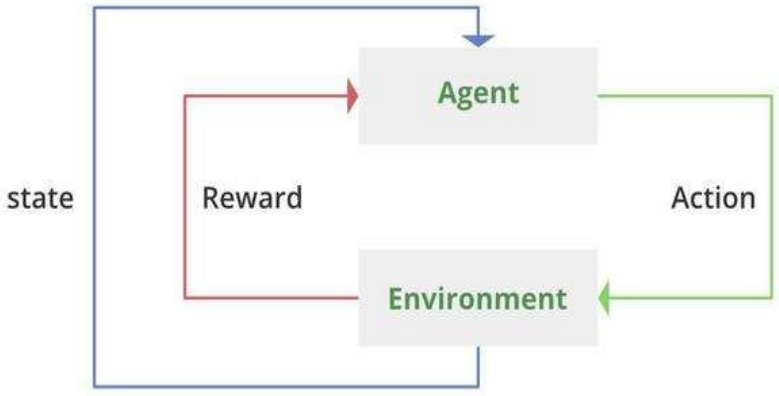
"Reinforcement learning is like training a pet or playing a video game.

- When the agent does something good → give reward.
- When it does something bad → negative reward or no reward.
  Over time, it learns which actions give higher long-term reward."

We don't give correct action for each situation explicitly.
We give feedback in form of rewards.

2.3.2 RL Setup (Formal Elements)

We have:



- Environment: the world in which agent acts (game, robot world, etc.)
- Agent: the learner/decision maker.
- State $s \in S$: current situation.

- Action $a \in A$: what the agent can do in that state.
- Reward $r \in R$: scalar feedback after action (positive or negative).

2.3.3 Example: Game Playing

- State: position of player, enemies, score.
- Actions: move left/right/up/down, shoot, jump.
- Reward:
    - +1 for collecting coin,
    - -10 for dying,
    - +100 for completing level.

Over many iterations, RL algorithm tries different actions, sees rewards, and improves policy.

2.3.4 Real-Life Use Cases of RL

- Game playing:
    - AlphaGo, AlphaZero for Go, Chess.
- Robotics:
    - Navigation, walking, manipulation.
- Recommendation systems:
    - Choosing which item to show, getting feedback (click/no click) as reward.
- Operations research:
    - Dynamic pricing, inventory management.

2.4 Summary of the Three Types

End this portion with a clear comparison slide:

- Supervised Learning
  - Data: (x,y) with labels.
  - Goal: Learn mapping from inputs to outputs.
  - Examples: Spam detection, house price, disease classification.

- Unsupervised Learning
  - Data: x only (no labels).
  - Goal: Discover structure (clusters, lower dimensions, anomalies).
  - Examples: Customer segmentation, anomaly detection, topic discovery.

- Reinforcement Learning
  - Data: Experiences (state, action, reward, next state).
  - Goal: Learn policy to maximise long-term reward.
  - Examples: Game playing, robotics, recommendation with feedback.

I will also provide you Practice assignment for module's.

Join whatsApp Group && channel, I will keep posting everything there.