Machine learning (ML) is a field of artificial intelligence (AI) that enables systems to learn from data and improve their performance over time without being explicitly programmed.
It involves using algorithms to analyse large datasets, identify patterns, and then make predictions or decisions based on those patterns.

#How Machine Learning Works:

- Data Input: The process starts with feeding large amounts of data into the machine learning algorithm.
- Algorithm Training: Algorithms analyse this data to find patterns and relationships.
- Model Creation: The output of this training process is a "machine learning model" – a set of learned instructions that can perform a specific task.
- Inference and Prediction: This trained model is then used on new, unseen data to make predictions or classifications without human intervention.

#Types of ML:

1. What Is Supervised Learning?

- The model learns from labelled data (input-output pairs).
- Goal: Predict target variable (classification or regression).
- Features (X): Input variables used by the model.
  Example: Age, Salary, Years of Experience.
- Labels/Targets (Y): Output variable the model predicts.
  Example (Classification): “Spam” or “Not Spam”
  Example (Regression): House price in dollars.
- Ex: linear regression: Predict continuous values, logistic regression: Classification task (binary) etc, Decision Tree: tree structure where decisions split data based on feature thresholds.

2. What Is Unsupervised Learning?

- Learns patterns from unlabelled data.
- Goal: Discover hidden structure.
- Ex: **K-Means Clustering:** Groups data into k clusters based on feature similarity, **Principal Component Analysis (PCA)**: Dimensionality reduction → Find directions (principal components) that capture most variance.

#Different ML algorithms:

1) Linear Regression: Supervised → Regression (Predict continuous output)

- **Working Principle:** Learns relationship between independent variables (features) and a continuous target variable using a linear equation.
- $\mathbf{Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta n Xn + \varepsilon}$
- $\mathbf{Loss\ Function: Mean\ Squared\ Error\ (MSE) = (1/n)\ \Sigma\ (Y\_actual - Y\_predicted)^2}$
- **Example:** Predict house price based on size and number of rooms.

2) Logistic Regression: Supervised → Classification (Binary)

- **Working Principle:** Predicts probability of an event (e.g., spam vs. not spam) using the logistic (sigmoid) function.
- **$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta n Xn$**
- Apply Sigmoid Function: P(Y=1 | X) = 1 / (1 + e^(-z))
- Output between 0 and 1 → Threshold (e.g., 0.5) for class label.
- Example: Predict if an email is spam based on features like the number of links and words.

3) Decision Tree: Supervised → Can do Classification or Regression

- **Working Principle:** Tree-like structure splits data based on feature thresholds → Leaf nodes represent final prediction.

- **Example:** Classify whether a customer will buy a product:
    - Age < 30 → No
    - Age ≥ 30 and Income > $50k → Yes

4) Random Forest: Supervised → Classification or Regression

- **Working Principle:** Ensemble of decision trees → Aggregate predictions (majority voting for classification or averaging for regression).
- **Example:**
  Predict customer churn:
  Train multiple trees on random subsets of data + features → Final prediction is majority vote.

5) Support Vector Machine (SVM): Supervised → Classification

- **Working Principle:** Finds optimal hyperplane that separates classes with the largest margin.
- **Example:** Classify tumours as malignant or benign.
- **Kernel Trick:** Projects data into higher dimensions to make them linearly separable.

6) K-Means Clustering: Unsupervised → Clustering

- **Working Principle:**

    1. Initialize k centroids randomly.
    2. Assign points to nearest centroid.
    3. Recompute centroids as cluster means.
    4. Repeat until convergence.

- **Example:** Customer segmentation based on purchase history.

7) Principal Component Analysis (PCA): Unsupervised → Dimensionality Reduction

- **Working Principle:** Projects data onto principal components → Maximizes variance capture in fewer dimensions.
- **Example:** Reduce 50 features to top 3 components for visualization.

Evaluation Metrics:

#Classification Metrics:

1) **Confusion Matrix:**

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

2) **Accuracy:** (TP + TN) / (TP + TN + FP + FN)
3) **Precision:** TP / (TP + FP)
4) **Recall (Sensitivity):** TP / (TP + FN)
5) **F1-Score**: 2 × (Precision × Recall) / (Precision + Recall)
6) **ROC-AUC Curve**: Area under the curve → How well model distinguishes classes.

#Regression Metrics:

1. **Mean Absolute Error (MAE)**: (1/n) Σ |Y_actual - Y_predicted|
2. **Mean Squared Error (MSE)**: (1/n) Σ (Y_actual - Y_predicted)²
3. **R² Score**: $R^2 = 1 - (\Sigma (Y_{actual} - Y_{pred})^2) / (\Sigma (Y_{actual} - Y_{mean})^2)$

Code sample:

#Import Libraries:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, mean_squared_error, f1_score
```

#Load Dataset:
```
data = pd.read_csv('data.csv')  # Example dataset
X = data.drop('target', axis=1)  # Features
y = data['target']              # Target variable
```

#Train-Test Split:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

#Preprocessing:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

#Initialize Model (Example: Logistic Regression):
```
model = LogisticRegression()
```

#Train Model:
model.fit(X_train, y_train)

 # Predict:
y_pred = model.predict(X_test)

#Evaluate:
acc = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

print(f'Accuracy: {acc}')
print(f'Confusion Matrix:\n{cm}')