🧠 Machine Learning Workflow Overview

Machine Learning (ML) is the process of teaching computers to learn patterns from data and make predictions or decisions.
A complete ML workflow usually follows these steps:

🔹 1. Data Collection

You first gather data relevant to the problem — for example:

- Sales records for predicting revenue
- Exam scores for predicting performance
- Customer data for predicting churn

The quality of your data determines the quality of your model.

🔹 2. Data Preprocessing

Raw data is rarely ready for modeling.
We need to clean and prepare it by:

- Handling **missing values**
- Encoding **categorical variables**
- Normalizing or scaling numeric data
- Splitting the data into **features (X)** and **target (y)**

🔹 3. Train/Test Split

Once the data is ready, we split it into two main parts:

- **Training set (usually 70–80%)** → used to train the model
- **Test set (20–30%)** → used to evaluate how well the model performs on *unseen* data

Example:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

✅ **Purpose:**
This helps check if your model generalizes well — not just memorizes the training data.

🔹 4. Model Training

We choose a suitable algorithm (e.g., Linear Regression, Decision Tree, KNN, etc.) and train it using the training data.

👉 The model learns relationships between features (inputs) and the target (output).

Example:

```
model = LinearRegression()
model.fit(X_train, y_train)
```

🔹 5. Model Evaluation (Metrics)

After training, we use the test data to measure performance.
These **evaluation metrics** tell us how accurate or reliable the model's predictions are.

For Regression problems (continuous output):

- **MSE (Mean Squared Error):** average of squared differences between predicted and actual values.
  Smaller is better.

  $$MSE = n1 \sum (ypred - ytrue)2$$
- **RMSE (Root Mean Squared Error):** square root of MSE.
  Interpretable in original units.
- **R^2 (R-squared):** how much variance in the target is explained by the model.
  Closer to 1 → better fit.

For Classification problems (categorical output):

- **Accuracy:** percentage of correct predictions.
- **Precision & Recall:** measure how well the model handles class imbalance.
- **F1-score:** balance between precision and recall.

🔹 6. Overfitting & Underfitting

Term	Meaning	Visual Description
Underfitting	Model is too simple, fails to learn patterns (e.g., straight line through curved data)	High training error, high test error
Overfitting	Model memorizes the training data too well, performs poorly on new data	Low training error, high test error
Good fit	Model captures underlying pattern and generalizes well	Low training & test errors

✅ **Goal:** Find a balance — a model complex enough to learn patterns but simple enough to generalize.

🔹 7. Model Optimization

We can tune model parameters (called **hyperparameters**) to improve performance using methods like:

- Grid Search
- Cross-validation

🔹 8. Deployment

Once a model performs well, it can be deployed into production to make real-world predictions (e.g., recommending products, detecting fraud, forecasting sales).