🤖 What is Machine Learning (ML)?

📝 Definition:

Machine Learning is a branch of Artificial Intelligence (AI) that focuses on building systems that **learn from data** — without being explicitly programmed for every rule.

Instead of hardcoding rules, we give the machine **examples**, and it **learns patterns** from the data to make predictions or decisions.

Let's dive into **Machine Learning (ML) fundamentals**, focusing on the key concepts of **Supervised Learning**, **Unsupervised Learning**, and **Model Evaluation**.

🔍 1. Supervised Learning

🔧 Definition:

In supervised learning, the algorithm is trained on a **labeled dataset**, meaning each training example includes the **input data** and the **correct output** (label). The goal is to learn a mapping from inputs to outputs.

🗃️ Examples:

Use Case	Input	Output (Label)	Type
Spam detection	Email text	Spam or Not Spam	Classification
House price prediction	Features (size, location)	Price ($)	Regression
Image classification	Image pixels	Object category	Classification

---

📦 Common Algorithms:

- **Linear Regression** (for regression)
- **Logistic Regression** (for binary classification)
- **Decision Trees**
- **Random Forest**
- **Support Vector Machines (SVM)**

- **K-Nearest Neighbors (KNN)**
- **Neural Networks**

---

2. Unsupervised Learning

Definition:

Unsupervised learning works on **unlabeled data**. The goal is to find **hidden patterns** or **structures** in the data.

Examples:

Use Case	Input	Output	Type
Customer segmentation	Purchase history	Customer groups	Clustering
Dimensionality reduction	High-dimensional data	Lower-dim data	Compression
Anomaly detection	Network traffic	Outliers	Detection

Common Algorithms:

- **K-Means Clustering**
- **Hierarchical Clustering**
- **DBSCAN**
- **Principal Component Analysis (PCA)**
- **Autoencoders**

✅ 3. Model Evaluation

Once you've trained a model, you need to measure how well it's performing. Different techniques are used depending on whether you're doing **classification** or **regression**.

□ a. For Classification

🔍 Confusion Matrix:

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

🔢 Key Metrics:

Metric	Formula	Meaning
Accuracy	(TP + TN) / Total	Overall correctness
Precision	TP / (TP + FP)	How many predicted positives were correct
Recall	TP / (TP + FN)	How many actual positives were found
F1 Score	2 × (Precision × Recall) / (Precision + Recall)	Balance between precision and recall
ROC-AUC	Area under the ROC curve	Measures tradeoff between TPR and FPR

📉 b. For Regression

Metric	Formula	Meaning
Mean Absolute Error (MAE)	avg(	y_true - y_pred
Mean Squared Error (MSE)	avg((y_true - y_pred)²)	Penalizes larger errors more
Root Mean Squared Error (RMSE)	√MSE	Square root of MSE
R^2 Score (R-squared)	1 - (SS_res / SS_tot)	Proportion of variance explained by the model

---

□ c. Cross-Validation

Instead of training and testing on the same dataset, we use **cross-validation** to ensure the model generalizes well.

🌀 K-Fold Cross-Validation:

- Split data into **K equal parts**
- Train on **K-1** folds, test on the remaining one
- Repeat **K times** and average the results

🎯 Summary Table

Concept	Description	Example Algorithms
Supervised Learning	Learn from labeled data	Linear/Logistic Regression, SVM, Random Forest
Unsupervised Learning	Discover patterns in unlabeled data	K-Means, PCA, Autoencoders
Model Evaluation (Classification)	Accuracy, Precision, Recall, F1, ROC-AUC	Confusion matrix
Model Evaluation (Regression)	MAE, MSE, RMSE, R^2 Score	Residual plots, error metrics
Cross-Validation	Robust model testing method	K-Fold, Stratified CV

All above ML Basics concepts guide given step by step with code examples :-

https://geocities.ws/ommauryasir/ai/ai-model/What%20Is%20an%20AI%20Model%20and%20How%20to%20develop%20ai%20model.pdf

Check Simple Example On ML Concept With Code Example :-

House Price Prediction in India (Simulated)

📌 We'll Use:

• scikit-learn for model training (LinearRegression)

• pandas for a custom dataset

• Features relevant to Indian housing

https://geocities.ws/ommauryasir/ai/ai-model/ai-model-for-house-price-prediction.pdf

Here's your Full Pytorch Code for Indian housing price prediction, to accept user input after training — so users can predict prices for their own custom home features (like size, location, bedrooms, and bathrooms).

https://geocities.ws/ommauryasir/ai/ai-model/pytorch-housing-ai-model.pdf

House price prediction model with user inputs using Tensorflow :-

https://geocities.ws/ommauryasir/ai/deep-learning/House%20price%20prediction-model-with-tensorflow.pdf

Learn Machine Learning :-

☑ 1. Linear Regression

Goal: Predict a continuous value.

✅ Example: Predicting House Prices

You want to predict the **price of a house** based on its **size (in square feet)**.

How it works:

- The algorithm learns a line of best fit between "House Size" (input) and "Price" (output).
- It finds the line:
  `Price = m * Size + b`
  Where `m` is slope and `b` is the intercept.

Input data:

Size (sqft)	Price ($)
1000	150,000
1200	180,000
1500	220,000

Use: `LinearRegression()` from `scikit-learn` in Python

🧠 2. Classification

Goal: Predict categories (labels), not values.

✅ Example: Spam Email Detection

You want to classify whether an email is **Spam** or **Not Spam**.

How it works:

- Model looks at features like:
  - Does the email contain “Buy now”?
  - Is there an unknown link?

    - Are there too many capital letters?
- Then it assigns a label: `Spam` or `Not Spam`

Input example:

Email Text	Label
"Get rich quick now!"	Spam
"Meeting at 10 am"	Not Spam
"WIN A FREE iPHONE"	Spam

Use: `LogisticRegression()`, `RandomForestClassifier()`, or `NaiveBayes()` from `scikit-learn`

🌳 3. Decision Trees

Goal: Make decisions by splitting data based on questions.

✅ Example: Should a person get a loan?

You want to decide whether to **approve or reject** a loan application.

How it works:

The tree asks questions like:

1. Is credit score > 700?
2. Is income > $50k?
3. Has existing debt?

And based on answers (yes/no), it follows a branch to reach a decision.

Example:

```
Is credit score > 700?
    ├── Yes: Is income > 50k?
    │        ├── Yes → Approve
    │        └── No  → Reject
    └── No → Reject
```

Use: `DecisionTreeClassifier()` from `scikit-learn`

🔑 4. K-Nearest Neighbors (K-NN)

Goal: Classify a new data point based on its neighbors.

✅ Example: Classify Type of Fruit

You have fruits with features like weight and color, and you want to predict if a new fruit is an **apple or orange**.

How it works:

- Measure distance (like Euclidean) from the new fruit to all others.
- Pick the **k** nearest fruits.
- Majority vote determines the class.

Input example:

Weight	Color Score	Label
150g	0.8	Apple
170g	0.75	Apple
200g	0.4	Orange

A new fruit (160g, 0.77) → Closest to Apples → Classify as **Apple**

Use: `KNeighborsClassifier()` from `scikit-learn`

🔍 5. Clustering (K-Means)

Goal: Group similar items when there are no labels.

✅ Example: Customer Segmentation

You want to group customers based on **spending habits** to target marketing.

How it works:

- You have data like yearly income and spending score.
- K-Means groups customers into **k clusters** based on similarity.

Input data:

Customer	Income ($k)	Spending Score
A	15	39
B	16	81
C	100	77
D	120	20

→ K-Means might form 2 or 3 customer segments:

- Low income, high spending
- High income, low spending
- High income, high spending

Use: `KMeans()` from `scikit-learn`

🧠 Summary Table

Algorithm	Task Type	Real-World Example	Library Function
Linear Regression	Regression	Predict house prices	`LinearRegression()`
Classification	Classification	Spam email detection	`LogisticRegression()`, `NaiveBayes()`
Decision Trees	Classification	Loan approval decision	`DecisionTreeClassifier()`
K-Nearest Neighbors	Classification	Fruit type based on features	`KNeighborsClassifier()`
K-Means Clustering	Clustering	Customer segmentation	`KMeans()`

Below are **Python code examples** for each of the 5 core machine learning algorithms using **`scikit-learn`**, with explanations.

🛠️ Prerequisites (install if not already):

```
pip install numpy pandas matplotlib scikit-learn
```

☑ 1. Linear Regression – Predict House Prices

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

# Example data: house sizes (in sqft) and prices
X = np.array([[1000], [1200], [1500], [1800], [2000]])  # Features
y = np.array([150000, 180000, 220000, 260000, 280000])  # Target

# Train model
model = LinearRegression()
model.fit(X, y)

# Predict price for a 1700 sqft house
prediction = model.predict([[1700]])
print("Predicted price for 1700 sqft house: $", int(prediction[0]))
```

😂 2. Classification – Spam Detection (Simple Example)

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB

# Example email texts and labels
emails = ["Win a FREE iPhone", "Let's meet for lunch", "Buy now!",
"Important update", "FREE offer just for you"]
labels = [1, 0, 1, 0, 1]  # 1 = Spam, 0 = Not spam

# Convert text to feature vectors
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(emails)

# Train Naive Bayes model
model = MultinomialNB()
model.fit(X, labels)

# Test with a new email
test_email = ["Claim your FREE prize now"]
X_test = vectorizer.transform(test_email)
prediction = model.predict(X_test)
print("Prediction:", "Spam" if prediction[0] == 1 else "Not Spam")
```

🌳 3. Decision Tree – Loan Approval

```
from sklearn.tree import DecisionTreeClassifier

# Features: [credit_score, income_in_thousands]
X = [[750, 60], [680, 45], [710, 55], [600, 30], [720, 35]]
y = [1, 0, 1, 0, 0]  # 1 = Approve, 0 = Reject

# Train model
model = DecisionTreeClassifier()
model.fit(X, y)

# Predict for a new applicant
applicant = [[730, 50]]
prediction = model.predict(applicant)
print("Loan Decision:", "Approved" if prediction[0] == 1 else "Rejected")
```

🔑 4. K-Nearest Neighbors – Fruit Classification

```
from sklearn.neighbors import KNeighborsClassifier

# Features: [weight (g), color_score]
X = [[150, 0.8], [170, 0.75], [200, 0.4], [180, 0.5], [160, 0.82]]
y = ["Apple", "Apple", "Orange", "Orange", "Apple"]

# Train KNN model
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X, y)

# Predict new fruit
new_fruit = [[165, 0.78]]
prediction = model.predict(new_fruit)
print("Predicted fruit:", prediction[0])
```

🔍 5. K-Means Clustering – Customer Segmentation

```
from sklearn.cluster import KMeans
```

```
import numpy as np
import matplotlib.pyplot as plt

# Customer data: [Income, Spending Score]
X = np.array([
    [15, 39],
    [16, 81],
    [100, 77],
    [120, 20],
    [85, 60],
    [25, 40],
    [110, 25]
])

# Apply KMeans with 2 clusters
kmeans = KMeans(n_clusters=2, random_state=0)
kmeans.fit(X)

# Cluster labels
print("Cluster Labels:", kmeans.labels_)

# Visualize clusters
plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_, cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
s=200, c='red', label='Centers')
plt.xlabel("Income")
plt.ylabel("Spending Score")
plt.title("Customer Segments")
plt.legend()
plt.show()
```

✅ Summary of Libraries Used

Task	Model Used	Library
Linear Regression	`LinearRegression`	`sklearn.linear_model`
Classification	`MultinomialNB`	`sklearn.naive_bayes`
Decision Trees	`DecisionTreeClassifier`	`sklearn.tree`
K-NN	`KNeighborsClassifier`	`sklearn.neighbors`
Clustering	`KMeans`	`sklearn.cluster`