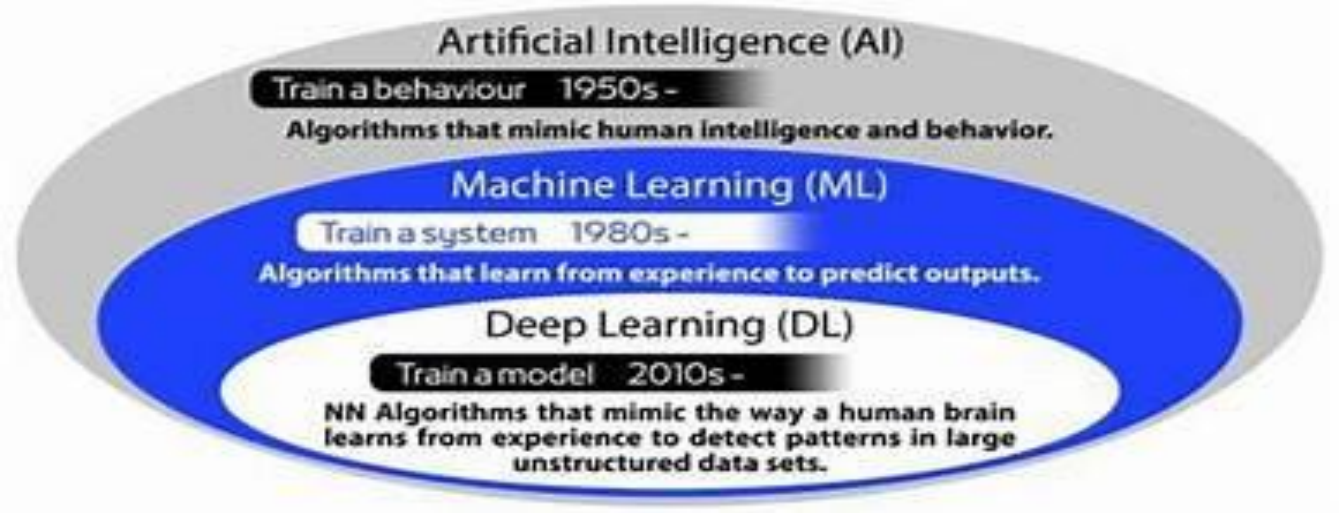
Intro to Machine Learning (ML)

What is Machine Learning?

Machine Learning is a branch of Artificial Intelligence that gives computers the ability to learn from data without being explicitly programmed.

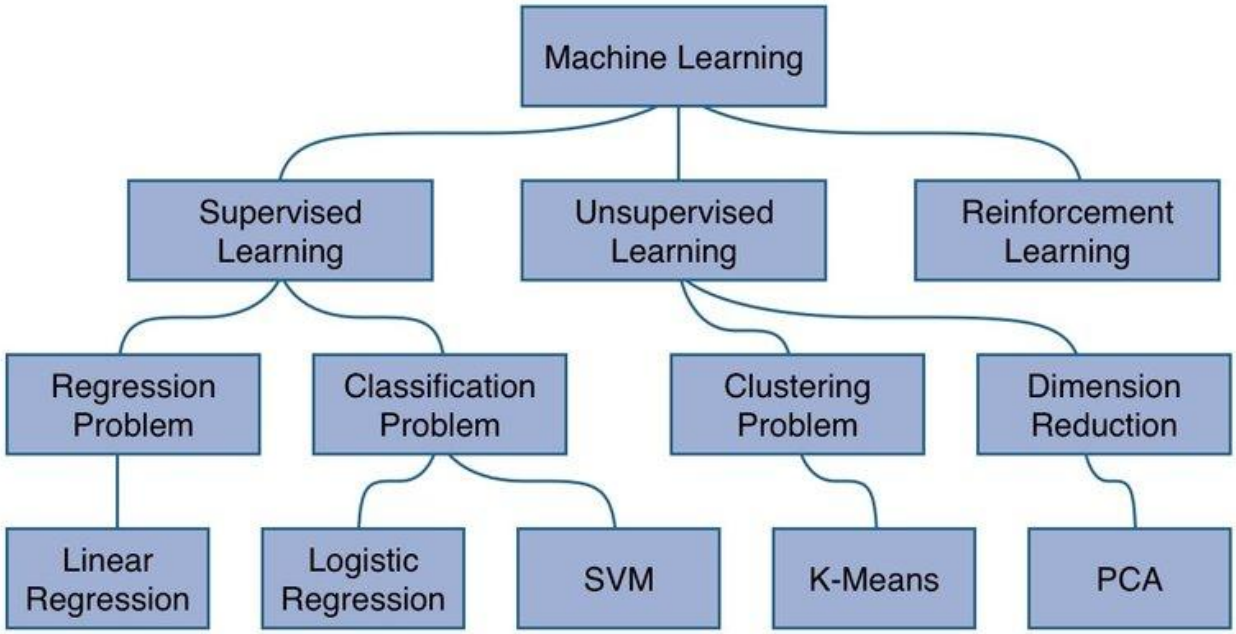


Key Idea

Instead of writing fixed rules, you provide data — the algorithm learns patterns and makes decisions.

Types of Machine Learning

Type	**Description**	**Example**
Supervised	Learn from labeled data	Predict salary based on experience
Unsupervised	Find structure in unlabeled data	Group customers by behavior
Reinforcement	Learn via reward feedback	Game AI, robots in environments



Why Scikit-learn?

Scikit-learn is a powerful Python library used for:

- Preprocessing data
- Training machine learning models
- Evaluating performance

Common features:

- Regression, Classification, Clustering
- Easy API
- Built on NumPy, SciPy, and matplotlib

Simple Linear Regression

What is Linear Regression?

It is a **supervised machine learning algorithm** used for **predicting numerical values** based on a linear relationship between input and output.

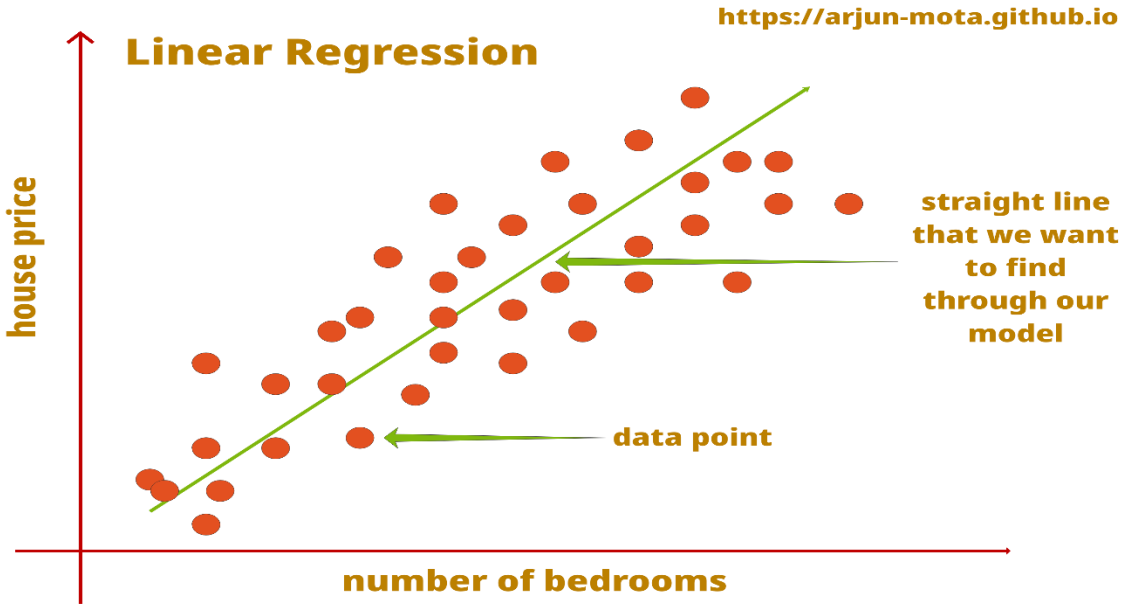
Mathematical Representation:

$y=mX+c$

Where:

- **X** = independent variable (e.g., CGPA)

- **y** = dependent variable (e.g., Placement Package)
- **m** = slope (weight)
- **c** = intercept



Mathematical Formula

Step 1: Calculate slope (m)

$$m = \frac{n\sum(x_i y_i) - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2}$$

Step 2: Calculate intercept (c)

$$c = \frac{\sum y_i - m\sum x_i}{n}$$

🔢 **Example:**

Suppose you have this data:

x	y
1	2
2	4

3	5
4	4
5	5

Goal of Linear Regression:

To find the **best-fitting line** that minimizes prediction error using a metric like **Mean Squared Error (MSE)**:

$$MSE = \frac{1}{n}\sum(y_{true} - y_{pred})^2$$

Practical Implementation using Scikit-learn

pip install numpy joblib scikit-learn

Step 1: Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import joblib
```

Step 2: Load Data

```
data = pd.read_csv("placements_new.csv")
print(data.head())
```

Assume the dataset contains:

- cgpa: student CGPA
- package: offered salary in LPA

Step 3: Prepare Features and Labels

```
X = data[['cgpa']]  # Feature (2D)
y = data['package']  # Target (1D)
```

- **X** must be 2D (each row = one student, each column = one feature)
- **y** is the output we're predicting

Step 4: Split Data

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- 80% for training, 20% for testing
- random_state ensures reproducibility

Step 5: Train the Model

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

Step 6: Predict and Evaluate

```
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

Step 7: Save Model

```
joblib.dump(model, 'package_predictor.joblib')
```

Step 8: Visualize

```
plt.scatter(X, y, color='blue')
plt.plot(X, model.predict(X), color='red')
plt.xlabel('CGPA')
plt.ylabel('Package (LPA)')
plt.title('CGPA vs Package')
plt.show()
```

Y=MX+C

```
model.coef_[0]  # gives m (slope)
model.intercept_  # gives c (intercept)
```

Linear Regression Model With GUI

```
import tkinter as tk
from tkinter import messagebox
import joblib
import numpy as np

# Load the trained model (ensure this file exists in the same directory)
model = joblib.load("package_predictor.joblib")

# Create the main window
app = tk.Tk()
app.title("Placement Package Predictor")
app.geometry("350x200")
app.resizable(False, False)

# Label and Entry for CGPA input
tk.Label(app, text="Enter your CGPA:", font=("Arial", 12)).pack(pady=10)
cgpa_entry = tk.Entry(app, font=("Arial", 12), width=10, justify='center')
cgpa_entry.pack(pady=5)

# Function to make prediction
def predict_package():
    try:
        cgpa = float(cgpa_entry.get())
```

```
        input_data = np.array([[cgpa]])  # Convert input to 2D array for model

        # Predict the package
        predicted_package = model.predict(input_data)[0]

        # Show result
        messagebox.showinfo("Predicted Package", f"Estimated Package: {predicted_package:.2f} LPA")
    except ValueError:
        messagebox.showerror("Invalid Input", "Please enter a valid CGPA (e.g., 8.2).")
    except Exception as e:
        messagebox.showerror("Error", f"Something went wrong:\n{e}")

# Add Predict Button
tk.Button(app, text="Predict Package", font=("Arial", 12), command=predict_package).pack(pady=20)

# Start the GUI event loop
app.mainloop()
```