***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?***

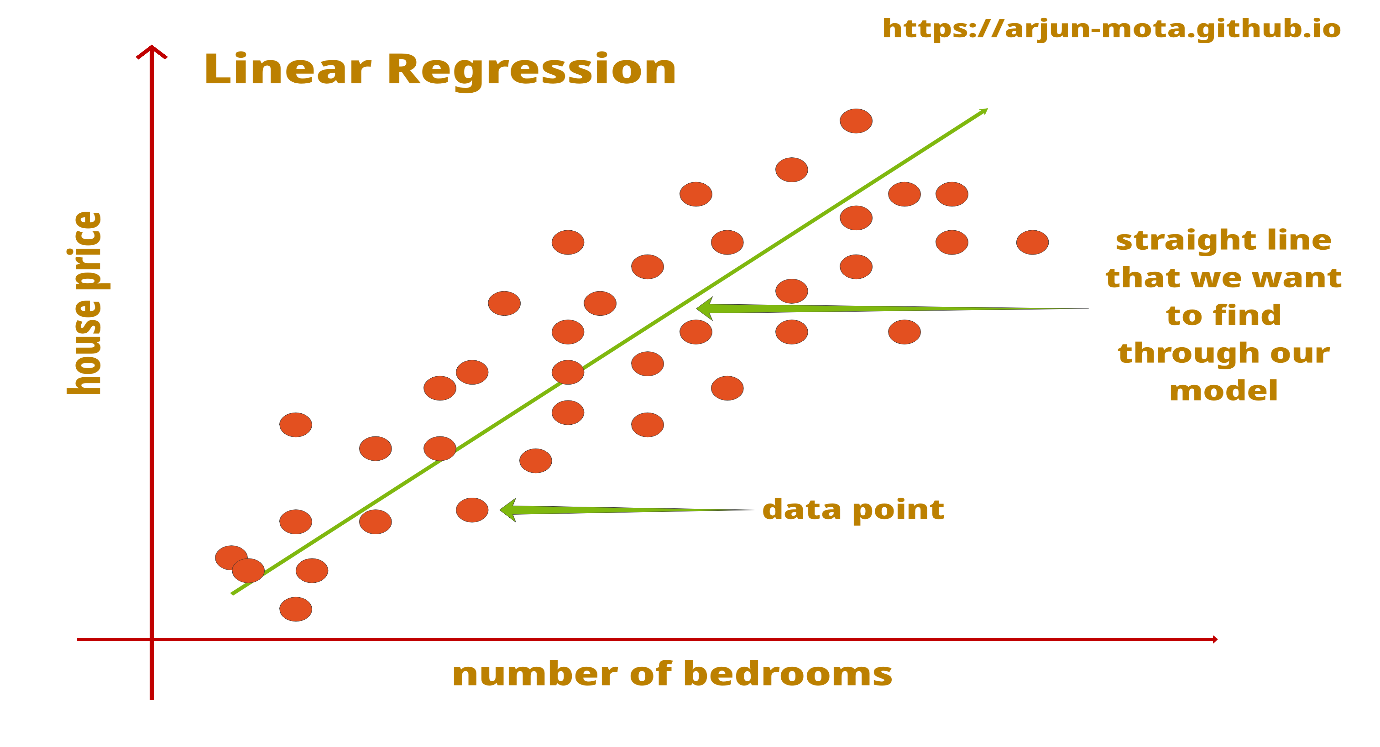
It is a **supervised machine learning algorithm** used for **predicting numerical values** based on a linear relationship between input and output.

**Mathematical Representation:**

y=mX+c

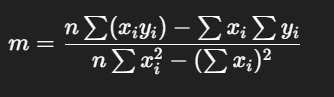
Where:

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

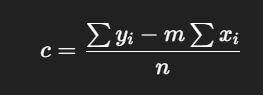


**Mathematical Formula**

**Step 1: Calculate slope (m)**

​​

**Step 2: Calculate intercept (c)**



**🔢 Example:**

Suppose you have this data:

|  |  |
| --- | --- |
| **x** | **y** |
| 1 | 2 |
| 2 | 4 |
| 3 | 5 |
| 4 | 4 |
| 5 | 5 |

**Step 1: Compute needed sums**

∑x=1+2+3+4+5=15∑y=2+4+5+4+5=20

∑xy=(1×2)+(2×4)+(3×5)+(4×4)+(5×5)=66

**Step 2: Plug into slope formula**

m=5×66−15×205×55−152=330−300275−225=3050=0.6m = \frac{5×66 - 15×20}{5×55 - 15^2} = \frac{330 - 300}{275 - 225} = \frac{30}{50} = 0.6m=5×55−1525×66−15×20​=275−225330−300​=5030​=0.6

**Step 3: Plug into intercept formula**

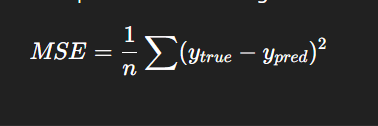
c=20−0.6×155=20−95=115=2.2c = \frac{20 - 0.6×15}{5} = \frac{20 - 9}{5} = \frac{11}{5} = 2.2c=520−0.6×15​=520−9​=511​=2.2

**✅ Final Regression Line:**

y=0.6x+2.2y = 0.6x + 2.2y=0.6x+2.2

**Goal of Linear Regression:**

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



***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()