Machine Learning

Week 3 • Class 2

Simple Linear Regression Using Python

Class Objective

By the end of this class, students will be able to:

- Load CSV data using NumPy and Pandas
- Inspect and understand tabular datasets
- Plot data points using Matplotlib
- Apply Simple Linear Regression in Python
- Visualize the regression line
- Interpret model output (slope & intercept)

Recap: What We Did Last Class

Previously, we:

- Defined linear regression
- Derived the regression equation manually
- Calculated slope and intercept
- Predicted values mathematically

Today: We let Python do the same work programmatically.

Dataset Used

We will use a CSV file containing:

Budget (X)	Sales (Y)
2	3
3	5
4	6
5	8
6	9

- X → Advertising Budget (in thousands)
- Y → Sales (in thousands)

Step 1: Required Libraries

We will use the following Python libraries:

- **NumPy** → numerical computation
- **Pandas** → data loading & manipulation
- **Matplotlib** → visualization
- **Scikit-learn** → regression model

Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

Step 2: Loading CSV Data Using Pandas

Assume the file is named `sales_data.csv`.

```
data = pd.read_csv("sales_data.csv")

print(data)
```

Inspecting the Dataset

```
data.head()

data.info()

data.describe()
```

Purpose:

- Understand structure
- Check data types
- Look for missing values

Separating Independent and Dependent Variables

```
X = data[['Budget']]   # Independent variable
Y = data['Sales']      # Dependent variable
```

Step 3: Visualizing the Data (Scatter Plot)

```
plt.scatter(X, Y)
plt.xlabel("Advertising Budget (in thousands)")
plt.ylabel("Sales (in thousands)")
plt.title("Budget vs Sales")
plt.show()
```

This plot shows the linear relationship between X and Y.

Step 4: Creating the Linear Regression Model

```
model = LinearRegression()

model.fit(X, Y)
```

Train (fit) the model on the data.

Extracting Model Parameters

```
m = model.coef_[0]
c = model.intercept_

print("Slope (m):", m)
print("Intercept (c):", c)
```

These values correspond to the formula:

$$Y = mX + c$$

Comparing With Manual Calculation

From previous class:

- Manual slope ≈ **1.5**
- Manual intercept ≈ **0.2**

Python output should closely match these values.

Step 5: Plotting Regression Line

```
Y_pred = model.predict(X)

plt.scatter(X, Y, label="Actual Data")
plt.plot(X, Y_pred, color="red", label="Regression Line")
plt.xlabel("Advertising Budget")
plt.ylabel("Sales")
plt.legend()
plt.show()
```

Making Predictions Using the Model

Predict sales when budget = 7:

```
new_budget = np.array([[7]])
prediction = model.predict(new_budget)

print("Predicted Sales:", prediction[0])
```

Interpretation of the Output

- **Slope (m)**
  → Increase in sales for every unit increase in budget
- **Intercept (c)**
  → Expected sales when budget = 0

This matches our theoretical understanding from Class 1.

Error Concept (Brief)

```
errors = Y - Y_pred
```

Later we will study:

- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)

Why Use Libraries Instead of Manual Calculation?

- Handles large datasets
- Less error-prone
- Faster computation
- Industry standard practice

Manual calculation is important for understanding, not for production.

In-Class Exercise

1. Load the CSV file provided
2. Plot the dataset
3. Train a linear regression model
4. Print slope and intercept
5. Predict output for a new input

Homework

- Complete the in-class exercise
- Write a full Python script from data loading to prediction
- Test with the practice dataset from Class 1
- Prepare for Multiple Linear Regression