TP1: Introduction to Machine Learning
Linear Regression Application

Objectives

- Install and configure a Python environment for Machine Learning.
- Manipulate libraries such as NumPy**,** Matplotlib**,** and Scikit-learn.
- Apply a Linear Regression algorithm to real or synthetic data.
- Visualize and interpret prediction results.

Part 1: Environment Setup

Option A : Local Installation (VS Code)

Step 1 – Install Visual Studio Code

- Download and install from: https://code.visualstudio.com/
- Install the following extensions:
  - Python (Microsoft)
  - Jupyter

Step 2 – Install Python

- Download and install Python 3.10+ from: https://www.python.org/downloads/
- Verify installation in terminal:
- python --version
- pip --version

Step 3 – Install Required Libraries

In VS Code terminal, run:

pip install numpy scikit-learn matplotlib

Option B : Google Colab (Alternative Setup)

Google Colab is an online environment for writing and executing Python code directly in your browser, no installation needed.

Steps:

1. Go to https://colab.research.google.com/
2. Sign in with your Google account.
3. Click “New Notebook”.
4. Rename your notebook to:
5. TP1_Linear_Regression.ipynb

6. In the first cell, import the necessary libraries (NumPy, Matplotlib, Scikit-learn).
7. In the following cells, write the steps to:
    - Generate data
    - Train the linear regression model
    - Display coefficients
    - Make predictions
    - Plot the results

Google Colab already has most libraries installed. If a library is missing, you can install it inside a cell using:

!pip install library_name

Task: Try to implement the same example from Part 3 by yourself in Colab, step by step.

Part 2: Linear Regression – Theory Reminder

Linear regression is a *supervised learning algorithm* used to model the relationship between one (or more) independent variables (X) and a dependent variable (Y) by fitting a straight line to the data.

Model: $Y=aX+b$ where:

- **Y** → dependent variable (output to predict)
- **X** → independent variable(s) (input features)
- **a** → slope (how much Y changes when X changes)
- **b** → intercept (value of Y when X = 0)

The goal is to find the **best-fitting line** that minimizes the prediction error (difference between predicted and actual values).

Part 3: Implementation with Scikit-learn

Step 1 – Create a Python file

Name it:

tp1.linear.regression.py

Step 2 – Add the following code

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

# 1. Generate sample data
X = np.array([1, 2, 3, 4, 5, 6]).reshape(-1, 1)
y = np.array([2, 4, 5, 4, 5, 7])

# 2. Create and train the model
model = LinearRegression()
```

```
model.fit(X, y)

# 3. Display model parameters
print("Coefficient (a):", model.coef_[0])
print("Intercept (b):", model.intercept_)

# 4. Make predictions
X_new = np.array([[7], [8], [9]])
y_pred = model.predict(X_new)
print("\nPredictions for X = 7, 8, 9 :", y_pred)

# 5. Visualization
plt.scatter(X, y, color='blue', label="Training data")
plt.plot(X, model.predict(X), color='red', label="Regression line")
plt.scatter(X_new, y_pred, color='green', label="Predictions")
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Linear Regression Example")
plt.legend()
plt.show()
```

Expected Output

Console:

Coefficient (a):
Intercept (b):
Predictions for X = 7, 8, 9 : [ ]

Plot:

- *Blue points*: training data
- *Red line*: regression line
- *Green points*: predictions

Part 4: Questions & Analysis

1. What does the coefficient (a) represent?
2. How does changing the training data affect the regression line?
3. What happens if you add noise to your dataset?
4. What are possible applications of linear regression in cybersecurity?