

## BÀI THỰC HÀNH 1: MÁY HỌC CƠ BẢN (PHẦN 1)

**Mục tiêu:** Thao tác cơ bản với Python. Hiểu được cách sử dụng Python để hiện thực một số thuật toán máy học cơ bản.

### 1. Các kiểu dữ liệu có cấu trúc trong Python

**Array** là một cấu trúc đặc biệt dùng để biểu diễn cho 1 vector. Python cung cấp thư viện `numpy` để xử lý đối với dữ liệu dạng vector. Để sử dụng `numpy`, ta phải import vào bằng lệnh:

```
import numpy as np
```

Chiều của các vector có thể khác nhau:

+ 1-D: `arr = np.array([1, 2, 3, 4, 5])`

+ 2-D: `arr = np.array([[1, 2, 3], [4, 5, 6]])`

+ 3-D: `arr = np.array([[[1, 2], [4, 5]], [[1, 3], [5, 6]]])`

Ngoài ra, kiểu dữ liệu **set** và **tuple** cũng khá thường được sử dụng để biểu diễn các dữ liệu có cấu trúc phức tạp.

Ví dụ về **set**: `s = {1, 1, -2, 3, 1, 4, 6}`

Ví dụ về **tuple**: `t = (1, 2)`

Điểm khác biệt chính giữa `array` và `set`:

+ Các phần tử trong **array** có thứ tự, và có thể trùng nhau.

+ Các phần tử trong **set** không có thứ tự, và không trùng.

Ta có thể tận dụng đặc điểm của `list` và `set` để **loại bỏ phần tử trùng nhau**. Ví dụ:

```
a = np.array([1, 2, 3, 1, 1, 2, 4, 5, 5])
```

```
b = list(set(a))
```

Kết quả sẽ là: 1,2,3,4,5.

Mặt khác, `tuple` được dùng để hoán vị giá trị của 2 biến với nhau. Cụ thể:

```
a = 2
```

```
b = 3
```

```
(a, b) = (b, a)
```

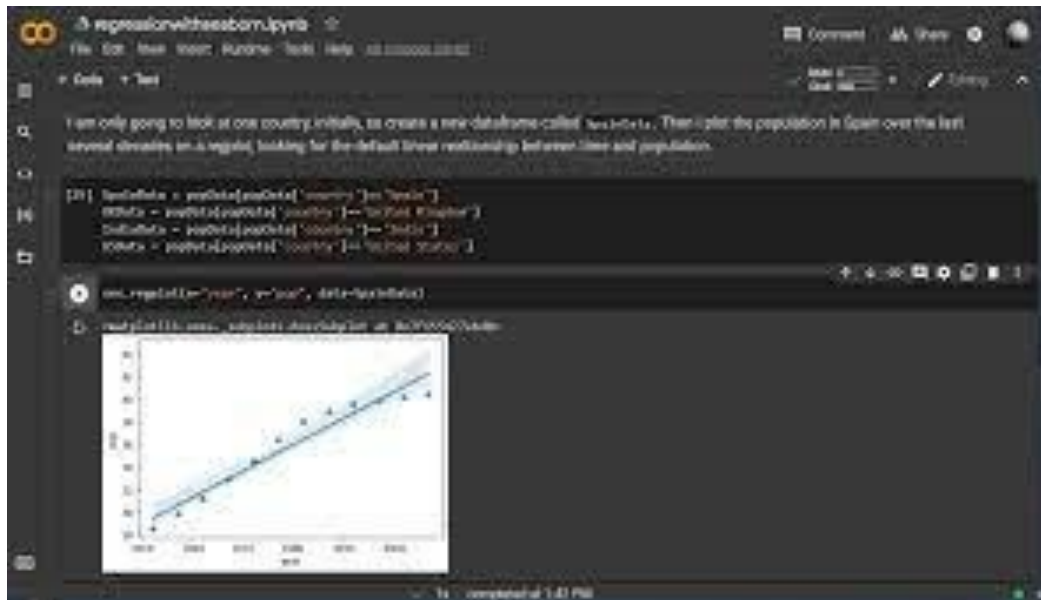
Kết quả sẽ là: `a = 3` và `b = 2`.

## 2. Môi trường Python và các thư viện

Có 2 dạng môi trường chính để làm việc với Python:

- Dạng 1: File code thuần (có đuôi .py).
- Dạng 2: File code kết hợp văn bản trình bày (có tài liệu gọi là nhúng code vào văn bản).

Các IDE thường dùng để code Python gồm: PyCharm, VS Code, VS Studio, Google Colab, và Jupyter Notebook (2 IDE này thuộc dạng 2).



Giao diện chính của Google Colab

Các thư viện thường dùng trong máy học gồm:

- **Numpy**: thư viện xử lý ma trận và vector.
- **sklearn**: thư viện thực hiện các mô hình máy học cơ bản, và các độ đo khác nhau.
- **Matplotlib**: thư viện vẽ đồ thị.
- **Pandas**: thư viện thao tác với dataframe. Hỗ trợ thao tác đọc và xử lý dữ liệu.
- **tensorflow**: thư viện hỗ trợ lập trình xử lý ma trận và vector phục vụ cho xây dựng mạng neural.

Để cài đặt thư viện trong Python, ta dùng lệnh: `pip install <tên thư viện>`.

## 3. Xây dựng mô hình hồi quy tuyến tính cơ bản.

Khởi tạo dữ liệu: khởi tạo 100 điểm dữ liệu gồm X và y.  $y = 4 + 3 \cdot X + e$ .

Gợi ý: sử dụng hàm `np.random.rand(100, 1)`. Hàm này sẽ tạo ra 1 vector với 100 phần tử ngẫu nhiên.

Code mẫu:

```
X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)
```

Thêm giá trị bias\_term vào dữ liệu X ban đầu. Mặc định bias\_term là 1.

Sử dụng thư viện `np.c_[v1, v2]` để nối 2 vector v1 và v2 lại với nhau. X ban đầu có 100 điểm dữ liệu, do đó vector bias\_term sẽ là 1 vector có 100 phần tử có giá trị 1.

Để tạo ra 1 vector 100 phần tử có giá trị là 1, ta dùng lệnh `np.ones((100, 1))`.

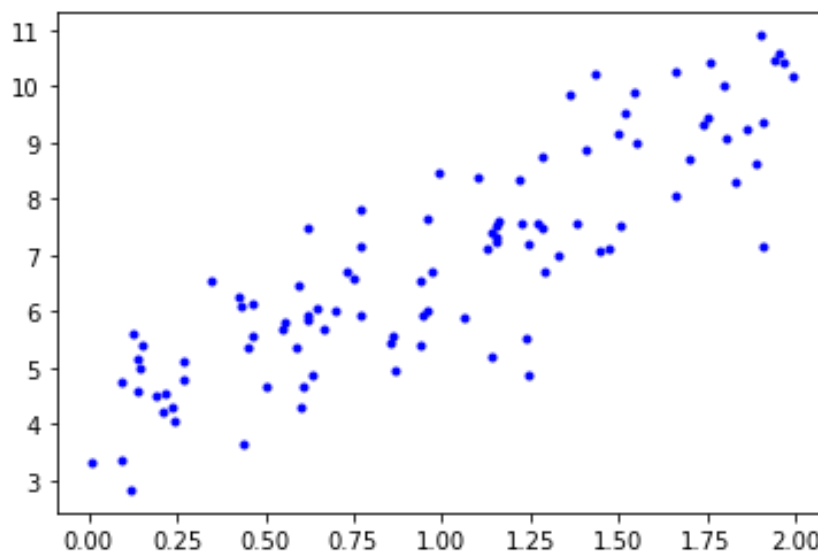
Code mẫu:

```
X_b = np.c_[np.ones((100, 1)), X]
```

**Câu 1:** Hãy vẽ biểu đồ cho dữ liệu (X,y) vừa tạo ở trên.

Gợi ý: Sử dụng hàm `plot` với 3 tham số: X, y và "b.". Nhớ khai báo thư viện `matplotlib` để vẽ đồ thị.

*Kết quả dự kiến:*



**Câu 2:** Hãy cho biết chiều (shape) của X\_b và y.

Gợi ý: Dùng lệnh `X_b.shape`

**Câu 3:** Hãy viết code Python tìm ra tham số tối ưu cho mô hình hồi quy tuyến tính. Tìm tham số tối ưu cho mô hình bằng phương trình sau:  $\hat{\theta} = (X^T X)^{-1} X^T y$  (Xem lại slide bài giảng về hồi quy để hiểu cách tại sao tìm ra được phương trình này). Kết quả lưu vào biến `theta_best`.

Gợi ý: **Sử dụng  $X\_b$  thay cho  $X$**  ( $X\_b$  là dữ liệu  $X$  ban đầu đã thêm bias\_term vào).

- $X^T$  là ma trận chuyển vị. Sử dụng lệnh `X.T` trong Python (điều kiện là  $X$  đã ép kiểu về `numpy_array`).
- $X^T X$  là phép tích vô hướng (dot product) giữa 2 vector. Sử dụng lệnh `np.dot()` trong thư viện `numpy`.
- $(X^T X)^{-1}$  là phép tính nghịch đảo của ma trận. Sử dụng lệnh `np.linalg.inv(A)` để tính nghịch đảo của ma trận  $A$ . Thay  $A = X^T X$ .

Tạo ra dữ liệu mới  $X\_new$  bằng đoạn lệnh sau (dùng để kiểm tra):

```
X_new = np.array([[0], [2]])  
X_new_b = np.c_[np.ones((2, 1)), X_new]
```

**Câu 4:** Hãy dùng tham số tối ưu của hồi quy tuyến tính vừa tìm được ở Câu 3, dự đoán kết quả cho  $X\_new\_b$ . Giá trị dự đoán lưu vào biến  **$y\_pred$** .

Gợi ý:  $\hat{y} = X\hat{\theta}$ . Sử dụng tích vô hướng (dot product) để tìm ra  $\hat{y}$  (chính là kết quả dự đoán).  
Tích tích vô hướng bằng lệnh `np.dot()`.

**Câu 5:** Hãy vẽ biểu đồ thể hiện kết quả dự đoán trên dữ liệu.

Gợi ý: Vẽ 2 biểu đồ bằng lệnh `plot` trong thư viện `matplotlib`:

- *Biểu đồ 1:* vẽ biểu đồ cho dữ liệu huấn luyện ban đầu  $X, y$ . Tham số gồm:  $X, y, "b."$  (chấm màu xanh dương - blue)
- *Biểu đồ 2:* vẽ biểu đồ cho mô hình hồi quy tuyến tính dưới dạng đường thẳng. Tham số gồm:  $X\_new\_b, y\_pred, "r."$  (dấu gạch màu đỏ - red).

#### 4. Bài tập

*Các bạn sử dụng Jupyter notebook hoặc Google Colab để thực hiện các câu hỏi.*

**Bài 1:** Thực hiện lại các câu hỏi (Câu 1 đến Câu 5) trong mục 2 để xây dựng lại mô hình hồi quy tuyến tính.

**Bài 2:** Thực hiện xây dựng mô hình hồi quy đa thức (Polynomial Regression) cho dữ liệu đa thức như sau:

Dữ liệu huấn luyện:

```
m=100
```

```
X = 6 * np.random.rand(m, 1) - 3  
y = 0.5 * X**2 + X + 2 + np.random.randn(m, 1)
```

Dữ liệu dự đoán:

```
X_new = np.linspace(-3, 3, 100).reshape(100, 1)
```

**Gợi ý:** sử dụng lớp *PolynomialFeatures* trong sklearn để chuyển giá trị trong X và X\_new về thành dạng bậc 2 (dữ liệu phi tuyến) theo đoạn code mẫu như sau:

```
from sklearn.preprocessing import PolynomialFeatures  
  
poly_features = PolynomialFeatures(degree=2,  
include_bias=False)  
X_poly = poly_features.fit_transform(X)
```

Dữ liệu huấn luyện sẽ thành: (**X\_poly, y**).

Các bước còn lại các bạn thực hiện giống như Bài 1

**Bài 3:** Hãy xây dựng mô hình hồi quy Ridge. Thực hiện tương tự như Bài 1.

Gợi ý: Tham số tối ưu cho mô hình Ridge Regression được tính như sau:

$$\theta = (X^T X + \alpha I)^{-1} X^T y$$

Chọn tham số  $\alpha$  là 0.02. Để tạo ma trận đơn vị, ta sử dụng lệnh: `np.identity(M)`, với M là kích thước của ma trận đơn vị I.

**Bài 4\*:** Các bạn hãy so sánh giữa tham số tối ưu tìm được bằng tay và bằng thư viện sklearn trong Bài 1 và Bài 3. So sánh kết quả thu được giữa thực hiện bằng tay và bằng thư viện.

Gợi ý: Các bạn tham khảo thư viện như sau:

*Hồi quy tuyến tính:* [https://scikit-](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html)

[learn.org/stable/modules/generated/sklearn.linear\\_model.LinearRegression.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html)

*Hồi quy Ridge:* [https://scikit-](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html)

[learn.org/stable/modules/generated/sklearn.linear\\_model.Ridge.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html)

Các bạn làm trực tiếp trên file jupyter notebook, đặt tên là:

**MSSV\_BaiThucHanh1.ipynb** (hoặc .jpynb)

Các bạn nộp trên course theo thời gian quy định nhé.