MSSV: 19127102

Họ tên: Võ Hoàng Gia Bảo

1. Hàm chọn cột theo thứ tự 1-11

```
def switch_col(argument):
    switcher = {
        0: "fixed acidity",
        1: "volatile acidity",
        2: "citric acid",
        3: "residual sugar",
        4: "chlorides",
        5: "free sulfur dioxide",
        6: "total sulfur dioxide",
        7: "density",
        8: "pH",
        9: "sulphates",
        10: "alcohol",
    }
    return switcher.get(argument, "Invalid")
```

Hàm giúp chọn ra cột trong dữ liệu csv thuận tiện hơn bằng cách sử dụng switch

2. Mô hình hồi quy tuyến tính Y = theta + theta * X

```
# y = theta_1 + theta_2 * x
def getAb(x, y):
    col1 = np.ones(len(x))
    colx = np.transpose(np.array(x))
    A = np.transpose(np.vstack((col1, colx)))
    b = np.array(y).reshape(len(y), 1)
    return A, b

def model(x, theta):
    return theta[0] + theta[1]*x
```

Mô hình để áp dụng với câu a, do dữ liệu csv cho cả 11 cột là 1 matrix nên ở đây sẽ dùng hàm numpy.vstack để kết hợp ma trận cột 1 và ma trận đã lấy được. Trước khi dùng vstack phải chuyển vị ma trận lấy được từ data trong csv, sau đó mới add cộ để từ size (1199,11) thành (1199,12), nếu chỉ làm với 1 lần chuyển vị sẽ thành

(1199,11) thành (1200,11) lúc đó sẽ không nhân được với ma trận cột 1 có size là (1199,1)

3. Hàm tính theta, tính norm và hiện biểu đồ

```
def theta(A, b):
    theta = np.linalg.inv(np.matmul(np.transpose(A), A)) @ np.matmul(np.transpose(A), b)
    return theta

def norm(A, b, theta):
    return np.linalg.norm(A @ theta - b)

def show(xs, ys, theta):
    plt.plot(xs, ys, "o", color="blue")
    ts = np.linspace(min(xs), max(xs), 50)
    yts = [model(t, theta) for t in ts]
    plt.plot(ts, yts, color="red")
    plt.show()
```

4. Mô hình tuyến tính ln(Y) = theta + theta * x^2 + theta * x^4

```
#Cau c
#Sigmoid : ln(y) = theta + theta * x ^ 2 + theta * x^4
def self_getAb(x, y):
    col1 = np.ones(len(x))
    colx = np.transpose(np.array(x))
    A = np.transpose(np.vstack((col1, colx**2, colx**4)))
    b = np.array(y).reshape(len(y), 1)
    return A, b

def self_model(x, theta):
    return np.exp(theta[0] + theta[1] * x ** 2 + theta[1] * x ** 4)
```

2 hàm để dùng cho câu c, tương tự với hàm mô hình hồi quy tuyến tính trên, chỉ khác công thức

Sau khi xây dựng các hàm trước tiên ta phải lấy data từ file wine.csv đã cho, ta phải import thư viện panda để lấy dữ liệu, sau đó chuyển về array hoặc matrix bằng cách gọi .to_numpy()

```
data = pd.read_csv('wine.csv', sep=';')
dt = data.to_numpy()
```

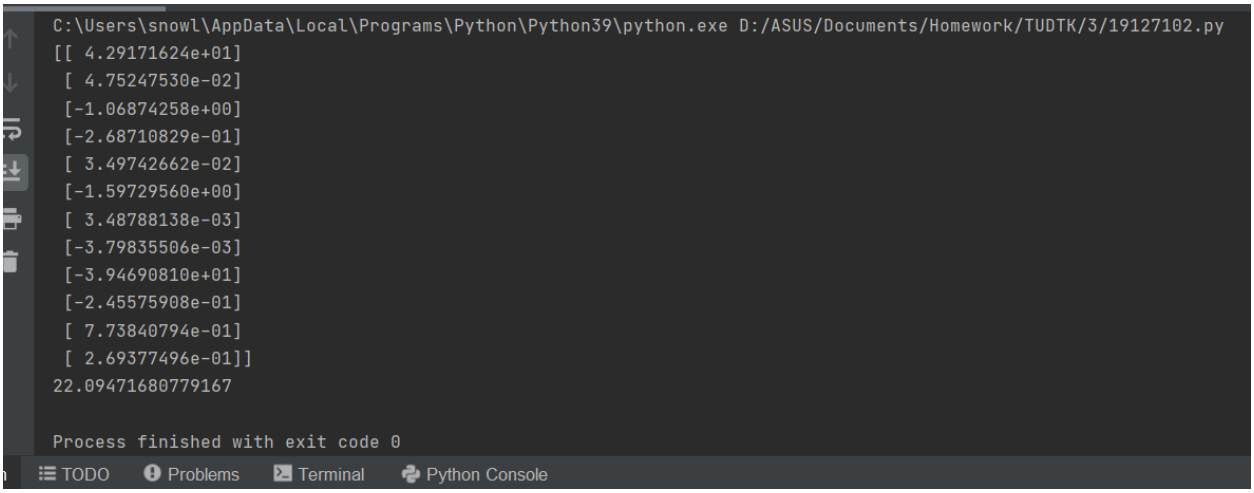
* Câu a

```
# Cau a
def Cau_a():
    x = (data.loc[:, "fixed acidity":"alcohol"]).to_numpy()
    y = (data.loc[:, ["quality"]]).to_numpy()
    A, b = getAb(x, y)
    print(theta(A, b))
    print(norm(A, b, theta(A, b)))
    #print(model_a(theta(A, b)))
```

Đối với điều kiện lấy hết cả 11 cột dữ liệu ta sẽ dùng .loc để lấy dữ liệu từ cột fixed acidity đến cột alcohol và chuyển về numpy để gán cho x. Tương tự với y với cột quality

Gọi hàm getAb và in ra theta và norm

Kết quả của câu a:

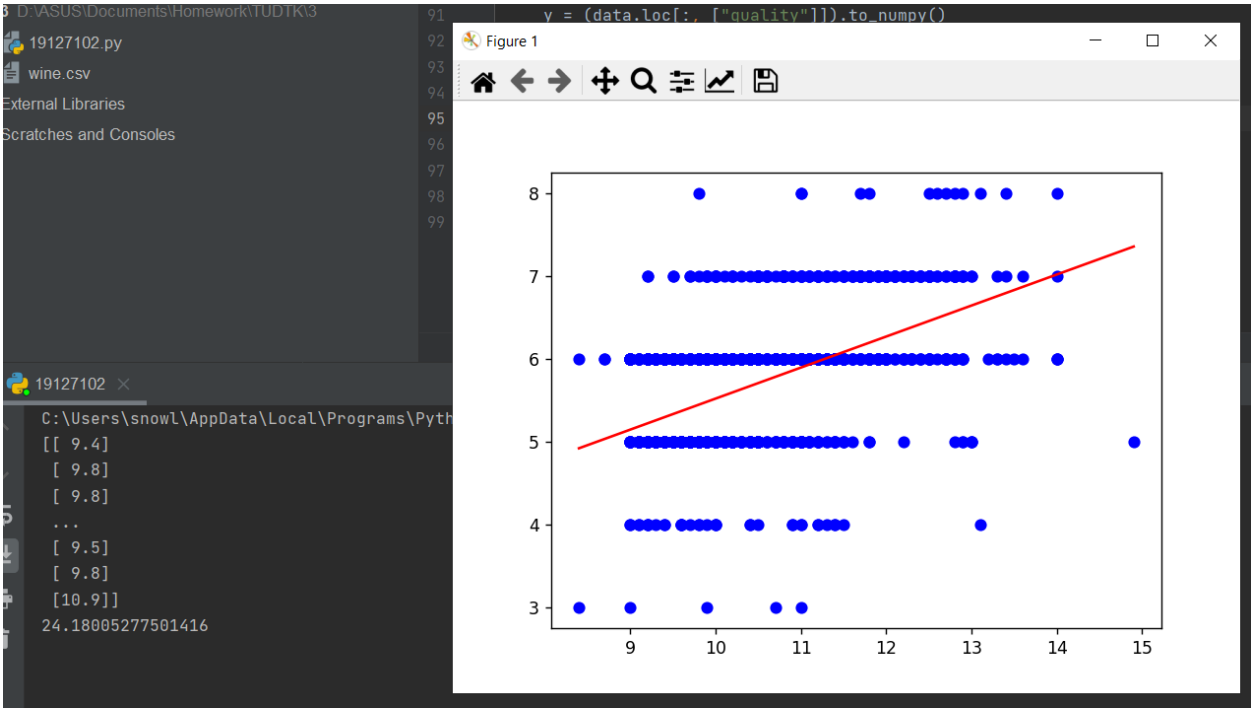


** Câu b

```python
#Cau_b
def Cau_b():
    col = None
    temp = None
    y = (data.loc[:, ["quality"]]).to_numpy()
    for i in range(11):
        x = (data.loc[:, [switch_col(i)]]).to_numpy()
        A, b = getAb(x, y)
        if i == 0 or norm(A, b, theta(A, b)) < temp:
            temp = norm(A, b, theta(A, b))
            col = x

    print(col)
    print(temp)
    show(col, y, theta(A, b))
```

Đối với điều kiện dùng lần lượt từng cột để đánh giá quality ta sẽ dùng vòng lặp để chọn ra từng cột và gán biến tạm với số norm nhỏ nhất, sau đó in x, norm của cột đó, in ra biểu đồ



*** Câu c

```python
def Cau_c():
    x = (data.loc[:, "fixed acidity":"alcohol"]).to_numpy()
    y = (data.loc[:, ["quality"]]).to_numpy()
    A, b = self_getAb(x, y)
    print(theta(A, b))
    print(norm(A, b, theta(A, b)))
```

Tương tự câu a nhưng dùng mô hình 4 ở trên

Đáp án câu c

```
C:\Users\snowl\AppData\Local\Programs\Python\Python39\python.exe D:/ASUS/Documents/Homework/TUDTK/3/19127102.py
[[ 1.41396564e+03]
 [ 9.53088197e-03]
 [-1.01654241e+00]
 [-1.25970132e+00]
 [ 5.92287871e-03]
 [-2.70837779e+00]
 [ 2.46239009e-05]
 [-3.33033975e-05]
 [-2.81240831e+03]
 [ 4.19897163e-01]
 [ 1.07152087e+00]
 [ 2.77540077e-02]
 [-3.37499169e-05]
 [ 1.92859024e-01]
 [ 2.56461806e+00]
 [-2.81448660e-05]
 [-5.71705577e+00]
 [ 7.94902750e-10]
 [ 4.43030680e-10]
 [ 1.39912521e+03]
 [-2.01845510e-02]
 [-2.59598419e-01]
 [-6.76303297e-05]]
21.765791093167202

Process finished with exit code 0
```