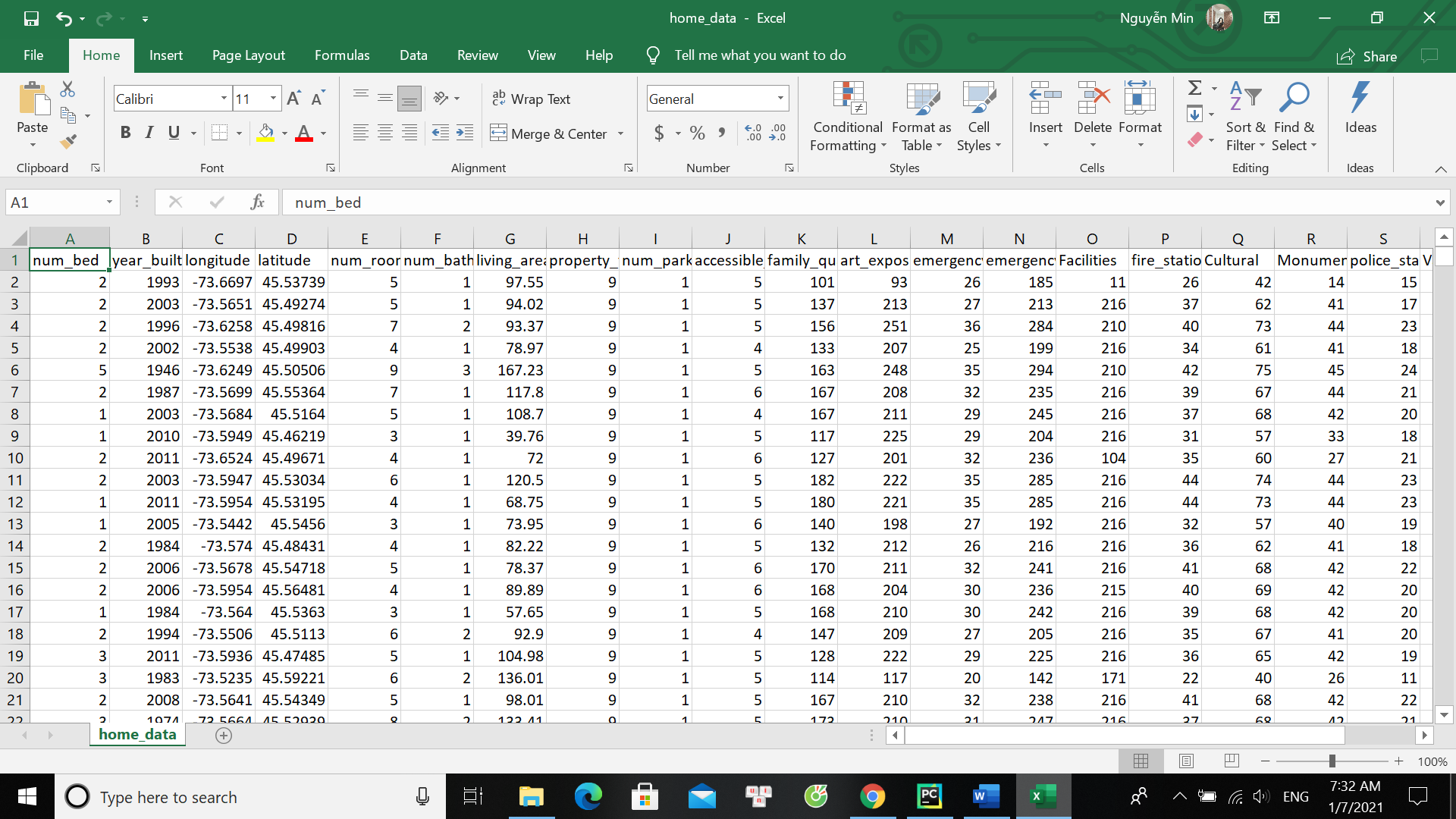
# **CHƯƠNG 2: Bài toán học máy hồi quy tuyến tính**

## **1.Mô tả dữ liệu**

-Bộ dữ liệu boston.csv gôm có 22 thuộc tính

Hình 1:Mô tả dữ liệu

-Mô tả thuộc tính:

|  |  |  |
| --- | --- | --- |
| STT | Thuộc tính | Mô tả |
| 1 | num\_bed | Số phòng ngủ |
| 2 | year\_built | Năm xây dựng |
| 3 | longitude | kinh độ |
| 4 | latitude | vĩ độ |
| 5 | num\_rum |  |
| 6 | num\_bath | số phòng tắm |
| 7 | living\_area | khu vực sống |
| 8 | property | bất động sản |
| 9 | num\_park | công viên |
| 10 | accessible | ôn hòa |
| 11 | family\_quality | chất lượng gia đình |
| 12 | art\_expos | triển lãm nghệ thuật |
| 13 | emergency\_shelters | nơi trú ẩn khẩn cấp |
| 14 | emergency\_water | nước khẩn cấp |
| 15 | facilities | cơ sở vật chất |
| 16 | fire\_stations | trạm cứu hỏa |
| 17 | cultural | văn hóa |
| 18 | monuments | tượngđài |
| 19 | police\_stations | đồn cảnh sát |
| 20 | vacant | chỗ trống |
| 21 | free\_parking | bãi đỗ xe |
| 22 | askprice | giá |

-Xác định bài toán học máy.

-Với bộ dữ liệu này, ta sử dụng bài toán hồi quy tuyến tính.

+Đầu vào X gồm 21 thuộc tính(cột) đầu.

+Đầu ra Y là thuộc tính (cột) cuối cùng.

-Sử dụng kĩ thuật LinearRegression.

## **2.Tiền xử lý dữ liệu**

-Đang nhập thư viện máy học **sklearn** , **numpy** và **gấu trúc** .

import pandas as pd

import numpy as np

from sklearn import linear\_model

from sklearn.model\_selection import train\_test\_split

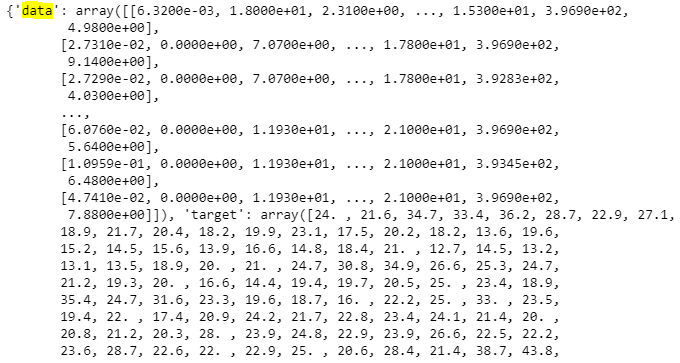
#Load the Boston Housing Data Set from sklearn.datasets and print it

from sklearn.datasets import load\_boston

boston = load\_boston()

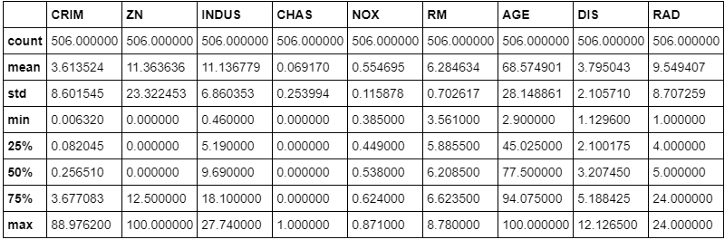
print(boston)

-kết quả:



hình 2:chuyển đổi tập dữ liệu thành khung dữ liệu

-chia dữ liệu thành 67% dữ liệu đào tạo và 33% dữ liệu thử nghiệm



-khởi tạo mô hình hồi quy tuyến tính

#Initialize the linear regression model

reg = linear\_model.LinearRegression()

#Split the data into 67% training and 33% testing data

#NOTE: We have to split the dependent variables (x) and the target or independent variable (y)

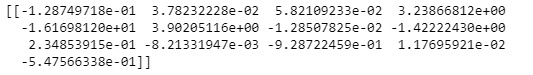
x\_train, x\_test, y\_train, y\_test = train\_test\_split(df\_x, df\_y, test\_size=0.33, random\_state=42)

#Train our model with the training data

reg.fit(x\_train, y\_train)

#Print the coefecients/weights for each feature/column of our model

print(reg.coef\_)

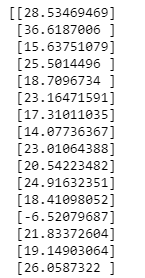


-in dự đoán giá trên dữ liệu thử nghiệm

#print our price predictions on our test data

y\_pred = reg.predict(x\_test)

print(y\_pred)



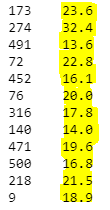
-in dự đoán cho hàng thứ ba của dữ liệu thực nghiệm giá thực tế= 13.6

#Print the the prediction for the third row of our test data actual price = 13.6

y\_pred[2]

#print the actual price of houses from the testing data set

y\_test[0]



-lỗi bình phương trung bình (MSE)

# Two different ways to check model performance/accuracy using,

# mean squared error which tells you how close a regression line is to a set of points.

# 1. Mean squared error by numpy

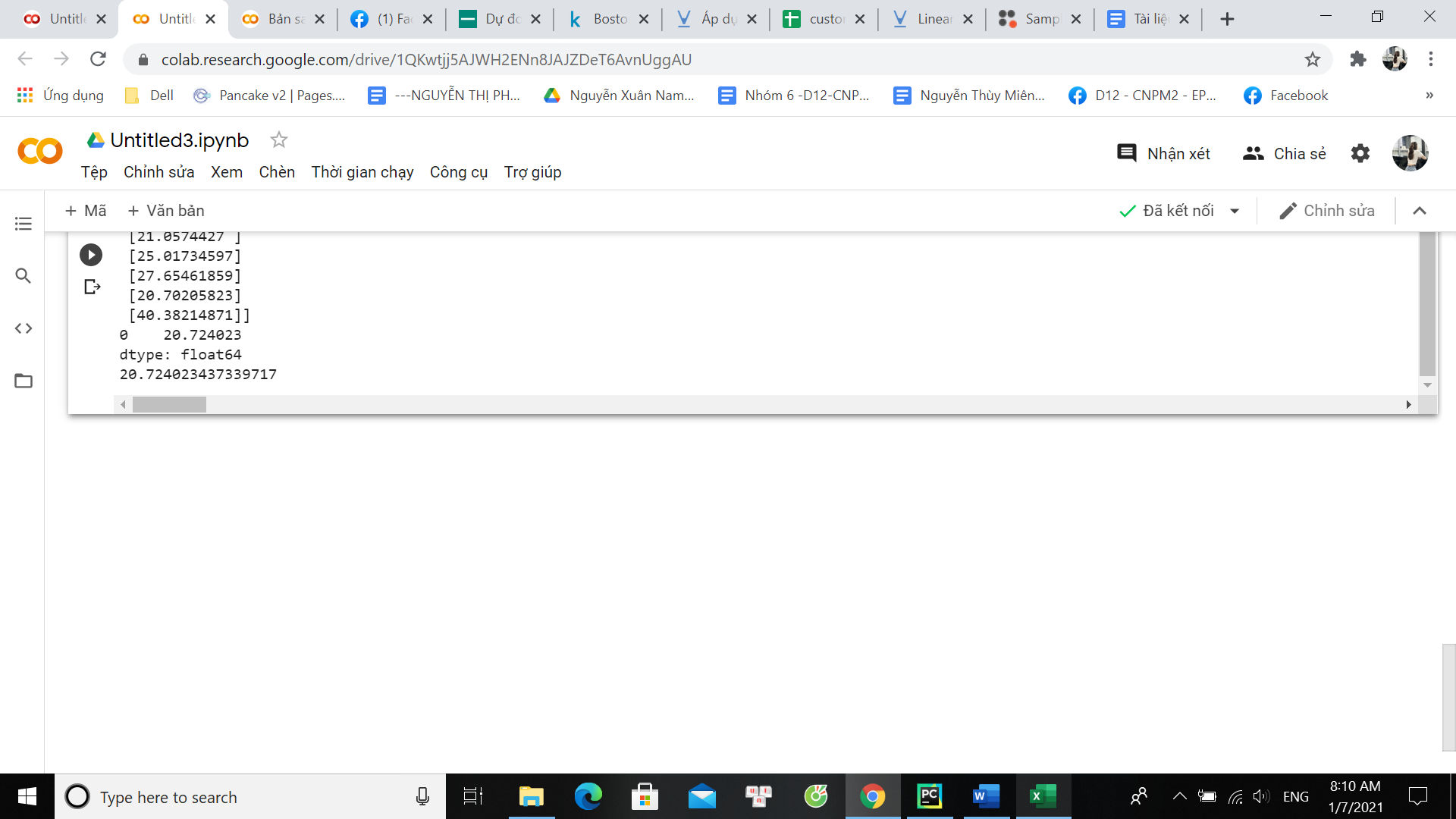
print(np.mean((y\_pred-y\_test)\*\*2))

# 2. Mean squared error by sklearn

# Resource: https://stackoverflow.com/questions/42453875/precision-score-and-accuracy-score-showing-value-error?rq=1

from sklearn.metrics import mean\_squared\_error

print(mean\_squared\_error(y\_test, y\_pred))



## -kích thước của tập dữ liệu với kích thước column.

## **print(boston.data.shape)**

## 

-kích thước của tập dữ liệu với kích thước column

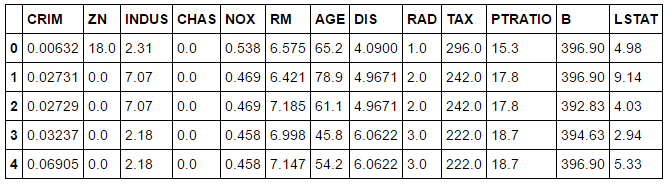
print(boston.data.shape)



-show được tên cột

bos.columns = boston.feature\_names

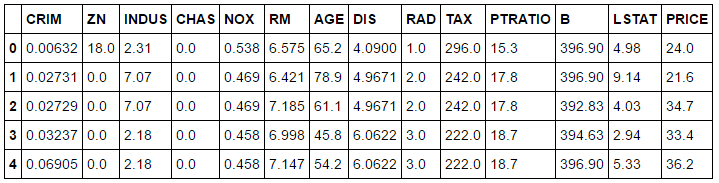
print(bos.head())



-column tên là PRICE

bos['PRICE'] = boston.target

print(bos.head())



-chạy hồi quy tuyến tính

from sklearn.linear\_model import LinearRegression

lm = LinearRegression()

lm.fit(X\_train, Y\_train)

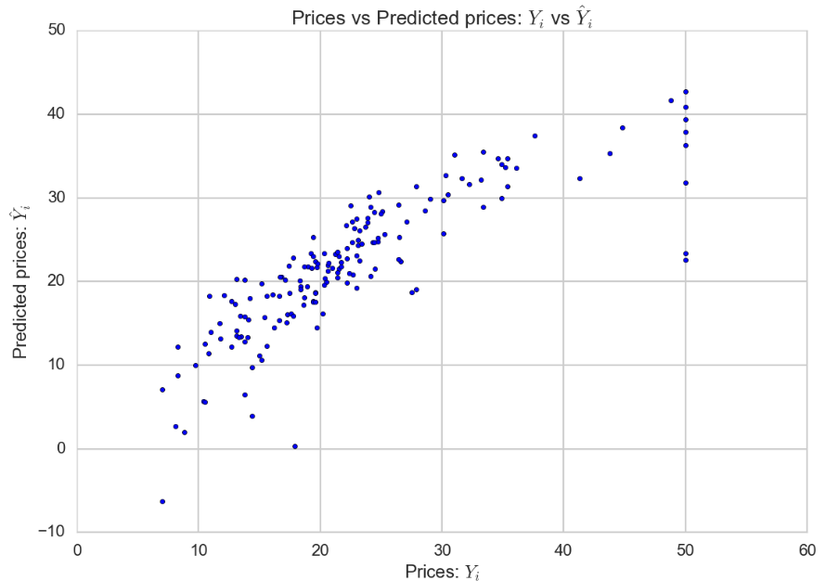
Y\_pred = lm.predict(X\_test)

plt.scatter(Y\_test, Y\_pred)

plt.xlabel("Prices: $Y\_i$")

plt.ylabel("Predicted prices: $\hat{Y}\_i$")

plt.title("Prices vs Predicted prices: $Y\_i$ vs $\hat{Y}\_i$")



## **3.Kết luận**

linear regression chạy khá nhanh và chuẩn xác, tuy nhiên nhược điểm của nó cũng như các thuật toán hồi quy khác Đơn giản để thực hiện,Được sử dụng để dự đoán các giá trị số,Dễ bị overfitting,Không thể sử dụng khi mối quan hệ giữa các biến độc lập và phụ thuộc là phi tuyến.Chính vì thế, để đảm bảo dữ liệu được huấn luyện tốt và cho ra dự đoán chính xác, cần tiến hành xử lý mất cân bằng ngay từ khâu ban đầu trước khi chạy mô hình dự đoán kết quả