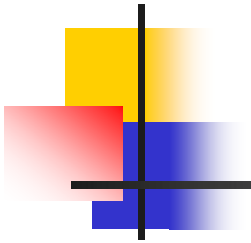
- **Các phương pháp học có giám sát**
  - **Hồi quy tuyến tính (Linear regression)**



Học có giám sát

- **Học có giám sát (Supervised learning)**
  - Tập dữ liệu học (*training data*) bao gồm các quan sát (*examples, observations*), mà mỗi quan sát được *gắn kèm với một giá trị đầu ra mong muốn.*
  - Mục đích là học một hàm (vd: một phân lớp, một hàm hồi quy,...) phù hợp với tập dữ liệu hiện có và khả năng tổng quát hoá cao.
  - Hàm học được sau đó sẽ được dùng để dự đoán cho các quan sát mới.
  - *Phân loại (classification):* nếu đầu ra (output – y) thuộc tập rời rạc và hữu hạn.
  - *Hồi quy (regression):* nếu đầu ra (output – y) là các số thực.

Hồi quy tuyến tính: Giới thiệu

- Bài toán hồi quy: cần học một hàm $y = f(\mathbf{x})$ từ một tập học cho trước $\mathbf{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_M, y_M)\}$ trong đó $y_i \cong f(\mathbf{x}_i)$ với mọi i.
  - Mỗi quan sát được biểu diễn bằng một véctơ *n* chiều, chẳng hạn $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})^T$.
  - Mỗi chiều biểu diễn một thuộc tính (attribute/feature)
- Mô hình tuyến tính: nếu giả thuyết hàm $y = f(\mathbf{x})$ là hàm tuyến tính.

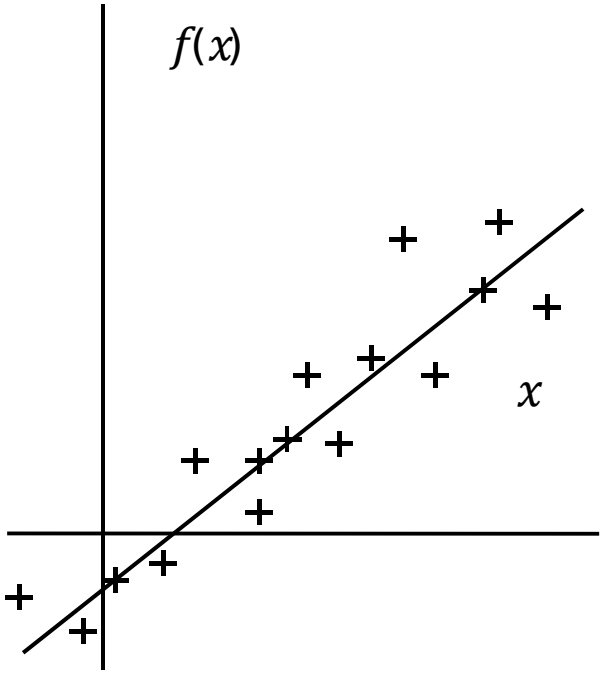
$$f(\mathbf{x}) = w_0 + w_1 x_1 + \ldots + w_n x_n$$

- Học một hàm hồi quy tuyến tính thì tương đương với việc học véctơ trọng số $\mathbf{w} = (w_0, w_1, \ldots, w_n)^T$

Hồi quy tuyến tính: Ví dụ

Hàm tuyến tính f(**x**) nào phù hợp?

0.13	-0.91
1.02	-0.17
3.17	1.61
-2.76	-3.31
1.44	0.18
5.28	3.36
-1.74	-2.46
7.93	5.56
...	...



Ví dụ: $f(x) = -1.02 + 0.83x$

Phán đoán tương lai

- Đối với mỗi quan sát $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$:
  - Giá trị đầu ra mong muốn c_x
    (Không biết trước đối với các quan sát trong tương lai)
  - Giá trị phán đoán (bởi hệ thống)

$$y_x = w_0 + w_1 x_1 + \ldots + w_n x_n$$

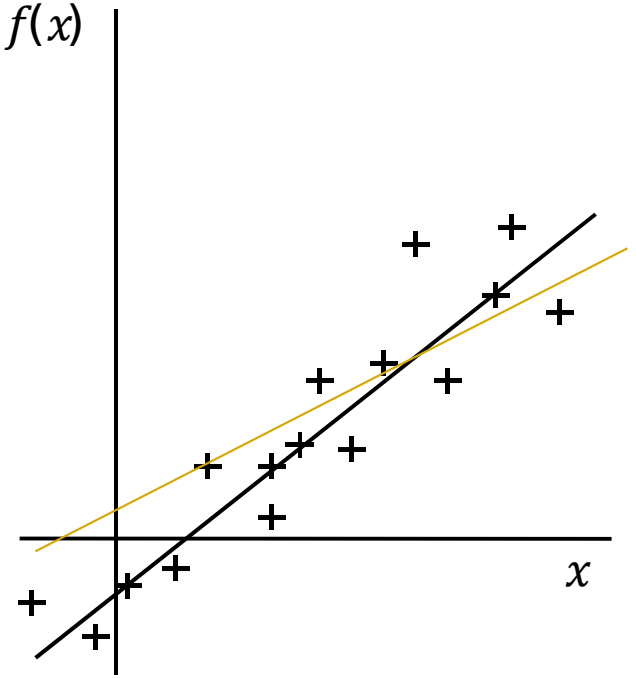
  - Ta thường mong muốn y_x xấp xỉ tốt c_x

- Phán đoán cho quan sát tương lai $\mathbf{z} = (z_1, z_2, \ldots, z_n)^T$
  - Cần dự đoán giá trị đầu ra, bằng cách áp dụng hàm mục tiêu đã học được `f`:

$$f(\mathbf{z}) = w_0 + w_1 z_1 + \ldots + w_n z_n$$

Học hàm hồi quy

- Mục tiêu học: *học một hàm f* sao cho khả năng phán đoán trong tương lai là tốt nhất.*
  - Tức là sai số $|c_z - f(\mathbf{z})|$ là nhỏ nhất cho các quan sát tương lai **z**.
  - Khả năng tổng quát hóa (generalization) là tốt nhất.
- Vấn đề: Có vô hạn hàm tuyến tính!!
  - Làm sao để học? Quy tắc nào?
- Dùng một tiêu chuẩn để đánh giá.
  - Tiêu chuẩn thường dùng là hàm lỗi (generalization error, loss function, …)



Hàm đánh giá lỗi (loss function)

- Định nghĩa hàm lỗi E
  - Lỗi (error/loss) phán đoán cho quan sát $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$

$$r(\mathbf{x}) = [cx\_f*(\mathbf{x})]^2 = (cx\_w_0 - w_1x_1 - \ldots - w_nx_n)^2$$

  - Lỗi của hệ thống trên toàn bộ không gian của **x**:

$$E = \mathbf{E}_x[r(\mathbf{x})] = \mathbf{E}_x[cx\_f*(\mathbf{x})]^2$$

- Mục tiêu học là tìm hàm f* mà E là nhỏ nhất:

$$f^* = \arg\min_{f \in H} E_x[r(x)]$$

Trong đó ***H*** là không gian của hàm *f*.

- Nhưng: trong quá trình học ta không thể làm việc được với bài toán này.

Hàm lỗi thực nghiệm

- Ta chỉ quan sát được một tập **D** = {(**x**$_1$, y$_1$), (**x**$_2$, y$_2$), …, (**x**$_M$, y$_M$)}. Cần học hàm f từ **D**.
- Lỗi thực nghiệm (empirical loss; residual sum of squares)

$$RSS(f) = \sum_i (y_i - f(x_i))^2$$

$$= \sum_i (y_i - w_0 - w_1 x_{i1} - \cdots - w_n x_{in})^2$$

  - *RSS/M* là một xấp xỉ của **E**$_x$[r(**x**)] trên tập học ***D***

- Nhiều phương pháp học thường gắn với RSS.

Bình phương tối thiểu

- Cho trước **D**, ta đi tìm hàm *f* mà có *RSS* nhỏ nhất.

$$f^* = \arg\min_{f \in H} RSS(f)$$

- Đây được gọi là bình phương tối thiểu (least squares).
- Tìm nghiệm **w*** bằng cách lấy đạo hàm của RSS và giải phương trình RSS' = 0. Thu được:

$$\boldsymbol{w}^* = (\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{y}$$

Trong đó **A** là ma trận dữ liệu cỡ Mx(n+1) mà hàng thứ i là $\mathbf{A}_i = (1, x_{i1}, x_{i2}, \ldots, x_{in})$; $\mathbf{B}^{-1}$ là ma trận nghịch đảo;

$\mathbf{y} = (y_1, y_2, \ldots, y_M)^T$.

Chú ý: giả thuyết $\mathbf{A}^T\mathbf{A}$ tồn tại nghịch đảo.

Bình phương tối thiểu: thuật toán

- Input: **D** = {($\mathbf{x}_1$, y_1), ($\mathbf{x}_2$, y_2), …, ($\mathbf{x}_M$, y_M)}
- Output: **w***
- Học **w*** bằng cách tính:

$$\boldsymbol{w^* = (A^T A)^{-1} A^T y}$$

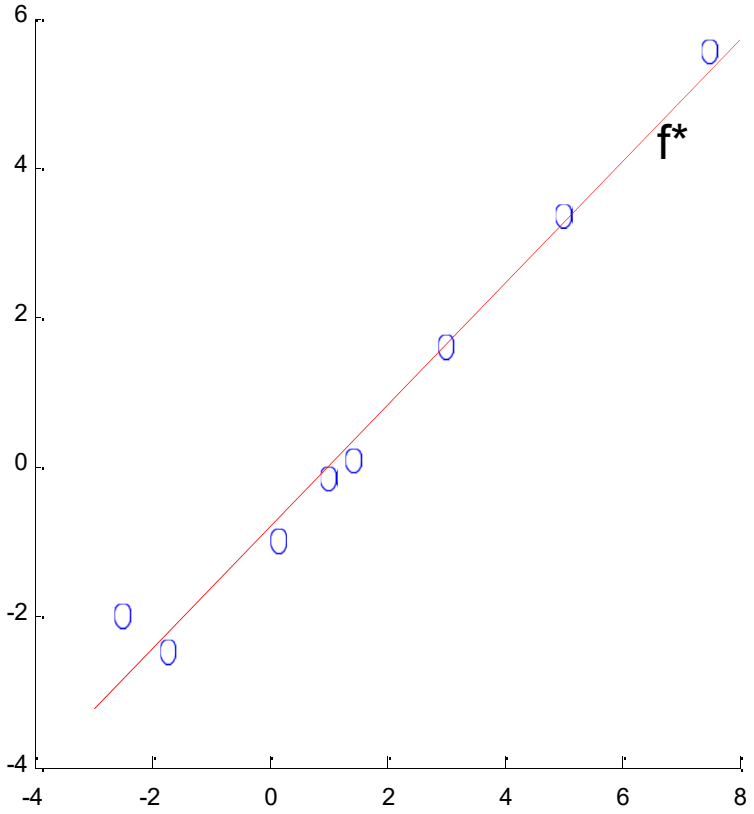
  - Trong đó **A** là ma trận dữ liệu cỡ Mx(n+1) mà hàng thứ i là $\mathbf{A}_i$ = (1, x_{i1}, x_{i2}, …, x_{in}); $\mathbf{B}^{-1}$ là ma trận nghịch đảo; **y** = (y_1, y_2, …, y_M)^T.
  - Chú ý: giả thuyết $\mathbf{A}^T\mathbf{A}$ tồn tại nghịch đảo.
- Phán đoán cho quan sát mới **x**:

$$y_s = w_0^* + w_1^* x_1 + \cdots + w_n^* x_n$$

Bình phương tối thiểu: ví dụ

Kết quả học bằng bình phương tối thiểu

x	y
0.13	-1
1.02	-0.17
3	1.61
-2.5	-2
1.44	0.1
5	3.36
-1.74	-2.46
7.5	5.56



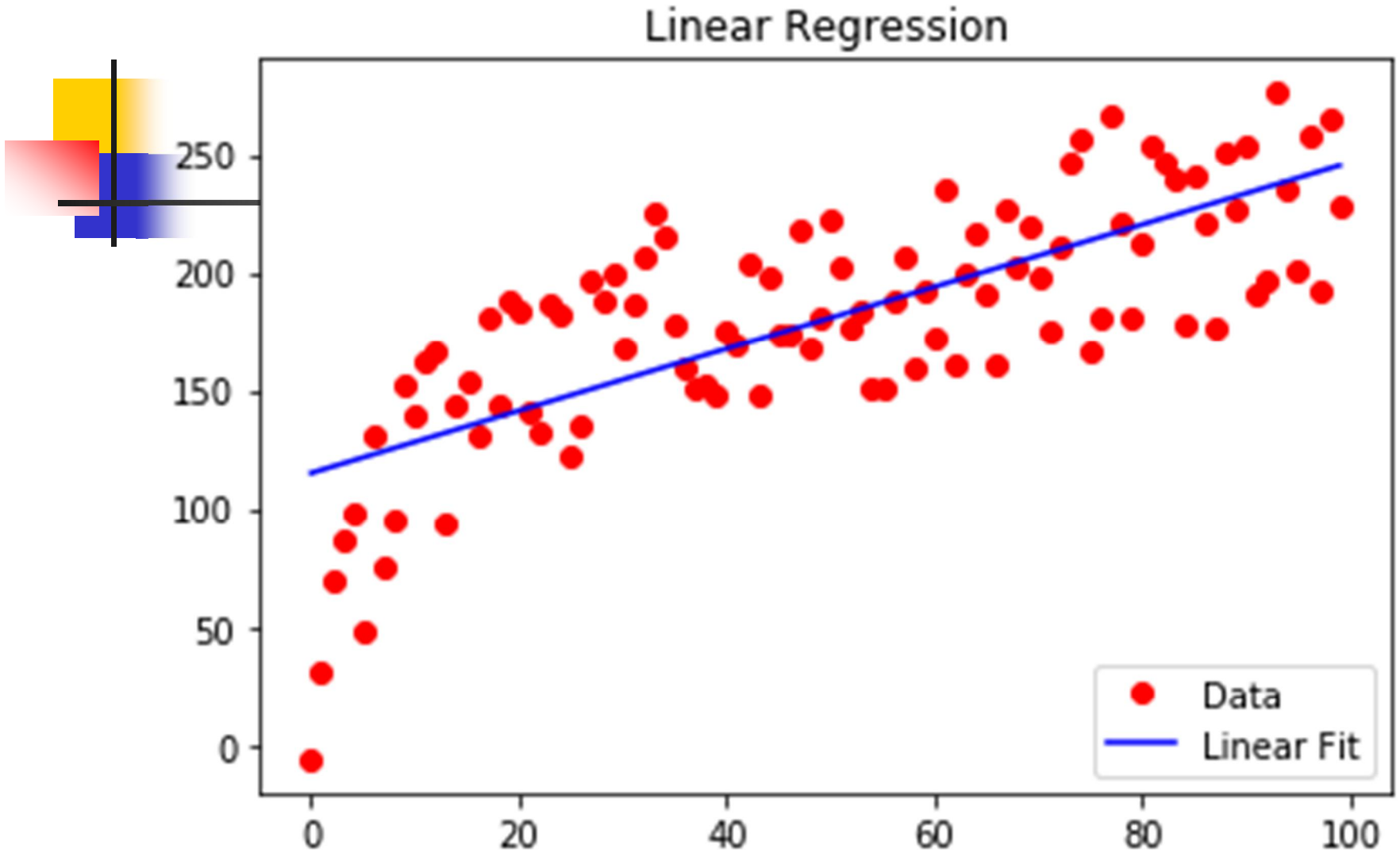
$f^*(x) = 0.81x - 0.78$

Bình phương tối thiểu: nhược điểm

- Nếu $\mathbf{A}^T\mathbf{A}$ không tồn tại nghịch đảo thì không học được.
  - Nếu các thuộc tính (cột của A) có phụ thuộc lẫn nhau.
- Độ phức tạp tính toán lớn do phải tính ma trận nghịch đảo.
  $\rightarrow$Không làm việc được nếu số chiều n lớn.
- Khả năng overfitting cao vì việc học hàm f chỉ quan tâm tối thiểu lỗi đối với tập học đang có.

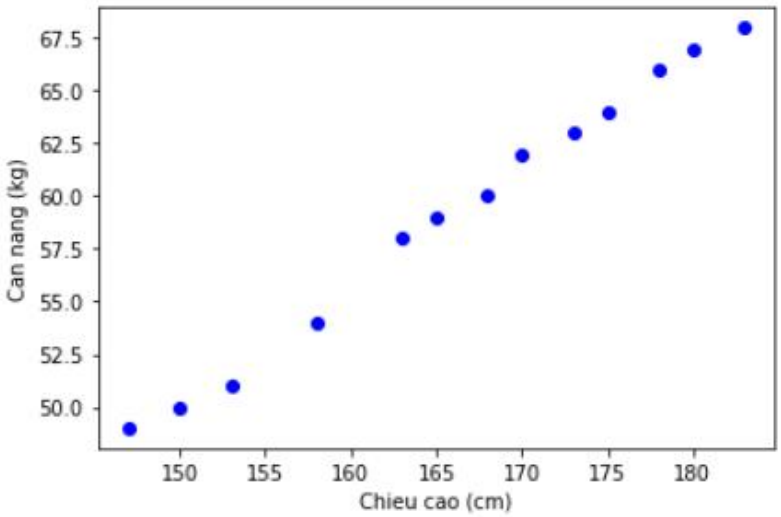
Một số ví dụ trên Python

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.utils import check_random_state
n = 100
x = np.arange(n)
rs = check_random_state(0)
y = rs.randint(-50, 50, size=(n,)) + 50. * np.log1p(np.arange(n))
# Fit LinearRegression models
lr = LinearRegression()
lr.fit(x[:, np.newaxis], y)  # x needs to be 2d for LinearRegression
fig = plt.figure()
plt.plot(x, y, 'r.', markersize=12)
plt.plot(x, lr.predict(x[:, np.newaxis]), 'b-')
plt.legend(('Data', 'Linear Fit'), loc='lower right')
plt.title('Regression')
plt.show()
```


Linear Regression
250
200
150
100
50
0
0
20
40
60
80
100
Data
Linear Fit

Ví dụ hiển thị dữ liệu

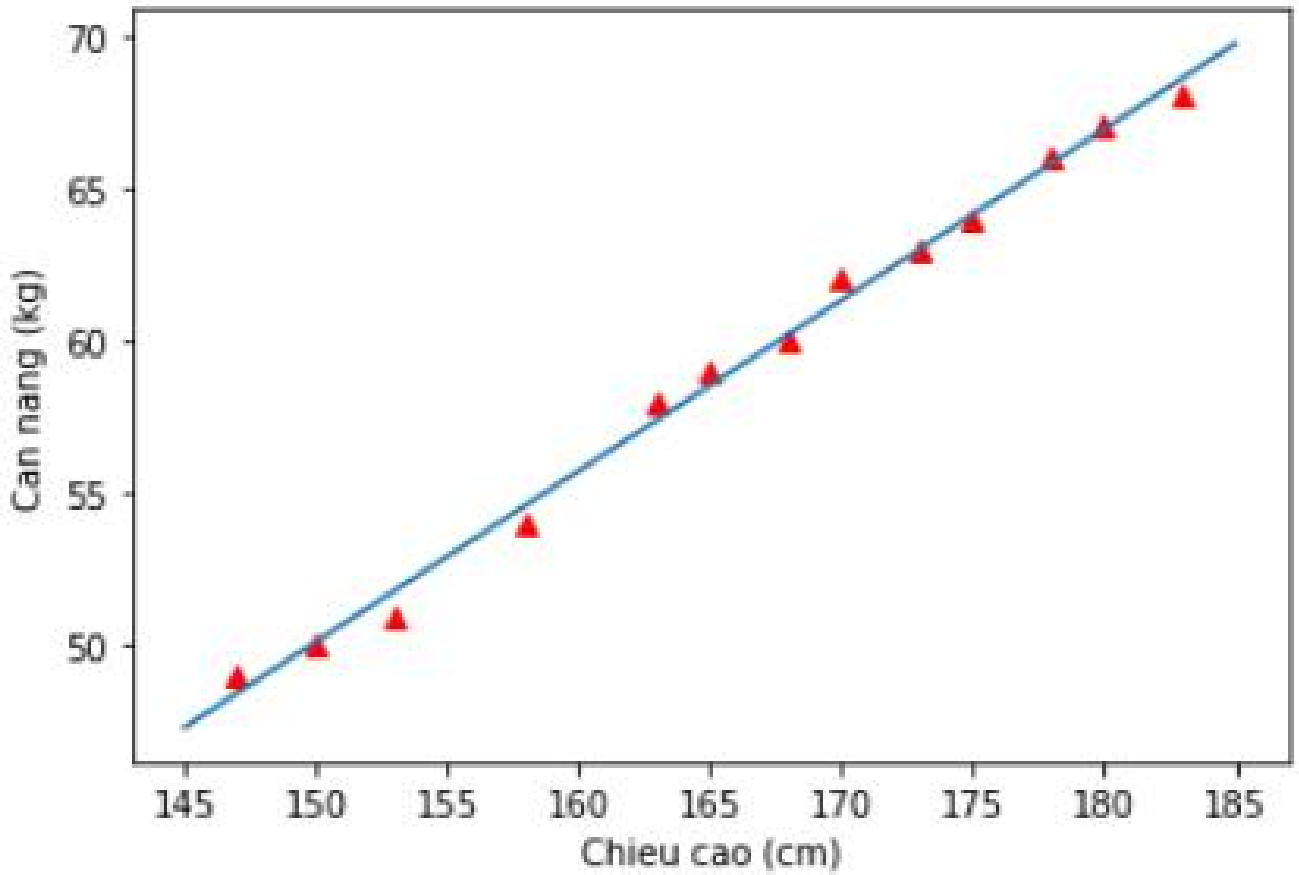
```
import numpy as np
import matplotlib.pyplot as plt
# chieu cao (cm)
X = np.array([[147, 150, 153, 158, 163, 165, 168, 170, 173, 175, 178, 180, 183]]).T
# can nang (kg)
y = np.array([49, 50, 51,  54, 58, 59, 60, 62, 63, 64, 66, 67, 68])
# hien thi du lieu
plt.plot(X, y, 'bo')
plt.xlabel('Chieu cao (cm)')
plt.ylabel('Can nang (kg)')
plt.show()
```



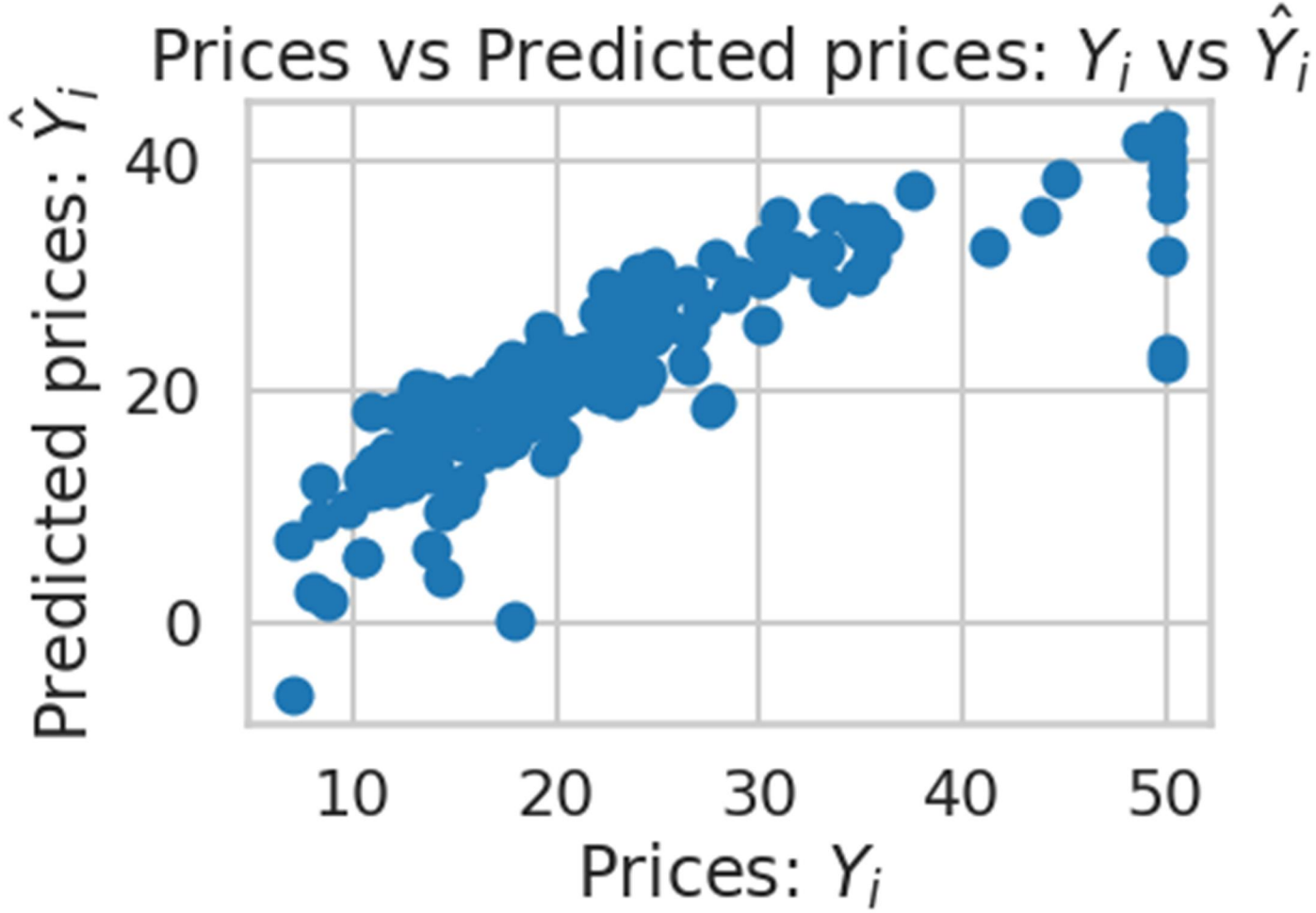
Hồi quy tuyến tính sử dụng thư viện Scikit-learn

```
from sklearn import linear_model
regr = linear_model.LinearRegression()
# height (cm)
X = np.array([[147, 150, 153, 158, 163, 165, 168, 170, 173, 175, 178, 180, 183
]]).T
# weight (kg)
y = np.array([49, 50, 51,  54, 58, 59, 60, 62, 63, 64, 66, 67, 68])
regr.fit(X, y)
print( 'He so w_1=', regr.coef_[0],",w_0 =",regr.intercept_ )
# Preparing the fitting line
w_0 = regr.intercept_
w_1 = regr.coef_[0]
x0 = np.linspace(145, 185, 2)
y0 = w_0 + w_1*x0
# Drawing the fitting line
plt.plot(X, y, 'r^')  # data
plt.plot(x0, y0)      # the fitting line
plt.xlabel('Chieu cao (cm)')
plt.ylabel('Can nang (kg)')
plt.show()
```

```
He so w_1= 0.5592049619396674 ,w_0 = -33.73541020580774
```

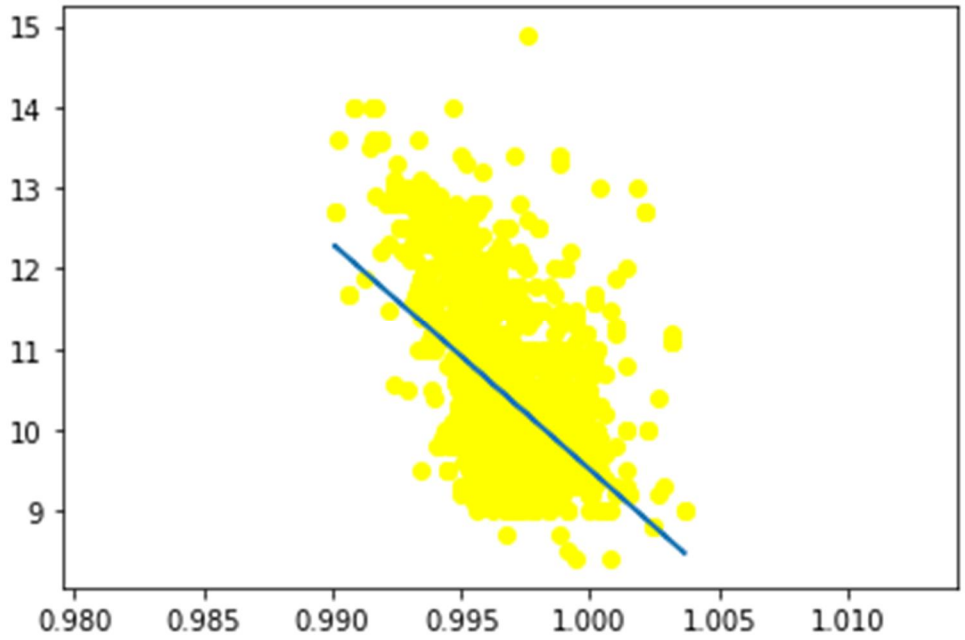


Ví dụ bộ dữ liệu Boston



Ví dụ: Bộ dữ liệu winequality-red.csv

```
from sklearn import linear_model
import pandas as pd
import numpy as np
wine = pd.read_csv("winequality-red.csv", sep=";")
clf = linear_model.LinearRegression()
# Sử dụng "density (mật độ)" làm biến giải thích
X = wine.loc[:, ['density']].as_matrix()
# Sử dụng "alcohol (Số độ cồn)" làm biến mục đích
Y = wine['alcohol'].as_matrix()
# Tạo model suy đoán
clf.fit(X, Y)
# Hệ số hồi quy
print("Hệ số Hồi quy:",clf.coef_)
# Sai số
print(clf.intercept_)
# Score
print(clf.score(X, Y))
```



Hoi quy da bien

```
from sklearn import linear_model
clf = linear_model.LinearRegression()
# Tạo dataframe chỉ chứa data làm biến giải thích
wine_except_quality = wine.drop("quality", axis=1)
X = wine_except_quality
# Sử dụng quality làm biến mục tiêu
Y = wine['quality']
# Tạo model
clf.fit(X, Y)
# Hệ số hồi quy
print(pd.DataFrame({"Name":wine_except_quality.columns
, "Coefficients":clf.coef_}).sort_values(by='Coefficients'))
```

	Name	Coefficients
7	density	-17.881164
4	chlorides	-1.874225
1	volatile acidity	-1.083590
8	pH	-0.413653
2	citric acid	-0.182564
6	total sulfur dioxide	-0.003265
5	free sulfur dioxide	0.004361
3	residual sugar	0.016331
0	fixed acidity	0.024991
10	alcohol	0.276198
9	sulphates	0.916334

Chuẩn hóa dữ liệu

```
from sklearn import linear_model
clf = linear_model.LinearRegression()
# chuẩn hoá dữ liệu các cột
wine2 = wine.apply(lambda x: (x - np.mean(x)) / (np.max(x) -
 np.min(x)))
# Tạo dataframe không chứa quality làm biến giải thích
X = wine2.drop("quality", axis=1)
# Sử dụng quality làm biến mục tiêu
Y = wine2['quality']
clf.fit(X, Y)
print(pd.DataFrame({"Name":X.columns, "Coefficients":np.abs(clf.coef_)
}).sort_values(by='Coefficients') )
print(clf.intercept_)
```

Chuẩn hóa dữ liệu:

	Name	Coefficients
2	citric acid	0.036513
3	residual sugar	0.047687
7	density	0.048708
0	fixed acidity	0.056479
5	free sulfur dioxide	0.061931
8	pH	0.105068
6	total sulfur dioxide	0.184775
4	chlorides	0.224532
9	sulphates	0.306056
1	volatile acidity	0.316408
10	alcohol	0.359057