Ngày 5 – Chủ đề: Dự báo bằng mô hình hồi qui tuyến tính

Đầu tiên là chúc mừng bạn đã rất cố gắng đọc tiếp cuốn eBook đến ngày thứ năm. Từ ngày này thì các vấn đề về phân tích dữ liệu sẽ được thảo luận chi tiết hơn mong rằng eBook mang đến cho các bạn trải nghiệm nhẹ nhàng, không nặng về lý thuyết. Đặc biệt giúp bạn có thể thực hành ngay để cảm nhận được vấn đề.

Ngày thứ năm này sẽ gồm 5 bài:

Bài 21: Giúp bạn ôn lại hoặc làm quen với mô hình hồi qui tuyến tinh. Thoạt nghe rất là cao siêu. Tuy nhiên nó cũng khá đơn giản và bạn có thể nắm bắt được nó nhanh chóng và trải nghiệm với code R và một chút Python.

Bài 22: Giúp bạn có thể hiểu một chút và diễn giải được các khái niệm liên quan. Đặc biệt khi sử dụng R thì lý giải được kết quả.

Bài 23: Mở rộng mô hình hồi qui tuyến tính từ một biến thành nhiều biến.

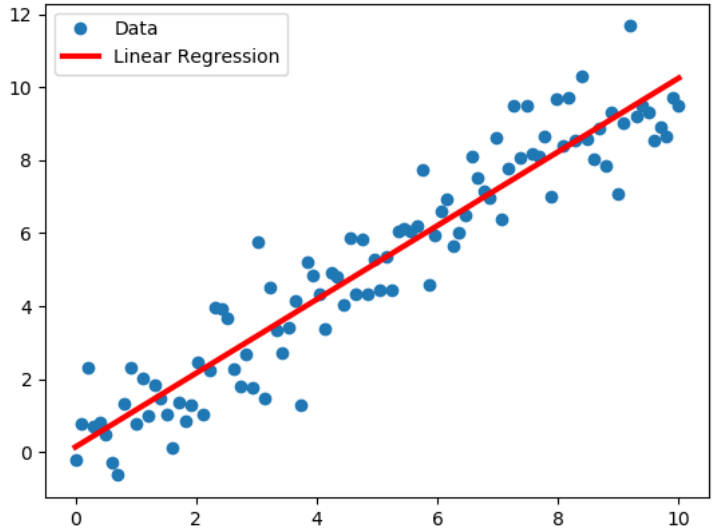
Bài 24: Giúp bạn cách để lựa chọn mô hình tối ưu để áp dụng trong thực tế.

Bài 25: Trải nghiệm trước một chút về việc xây dựng mô hình dự báo thông qua mô hình hồi qui tuyến tính.

Tương tự ngày thứ 3, ngày thứ 5 này và vài ngày sau nữa tôi tham khảo và sử dụng khá nhiều kiến thức, tài liệu từ các lớp học Phân tích dữ liệu của nhóm các Bác sĩ và anh Nguyễn Văn Tuấn. Các lớp học này tổ chức tại trường Đại Học Tôn Đức Thắng và vài nơi khác ở Sài Gòn.

Bài 22: Giới thiệu mô hình hồi qui tuyến tính

Hồi qui tuyến tính (linear regression) có phiên âm là /ˈlɪniər rɪˈgreʃn / nên âm đầu tiên đọc là **Lin** chứ không phải là **Lai** trong chữ line. Regression theo nghĩ đen là sự trở về trạng thái trước đó (a return to a former; a return to an earlier of life); linear có nghĩa là đường thẳng (line), hoặc gần, dọc theo đường thẳng (line near: gần, sát cạnh đường thẳng. Tức là sự sắp xếp dữ liệu dọc theo đường thẳng như hình minh họa bên dưới.

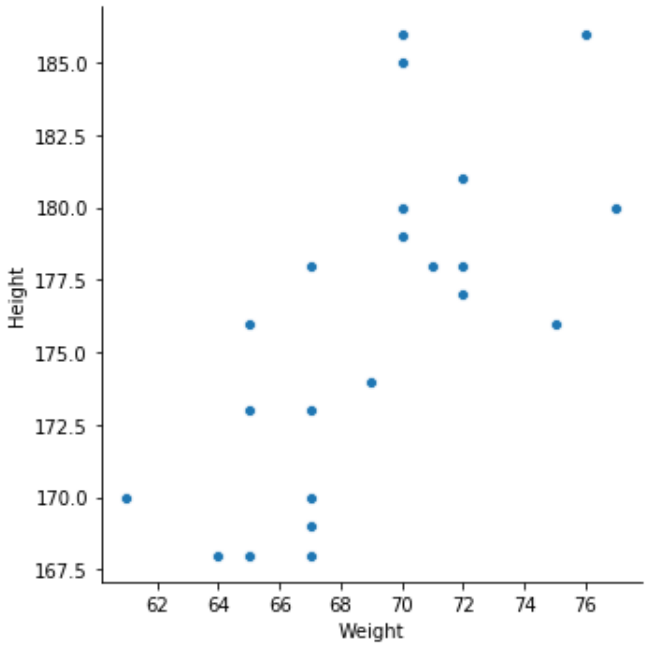


Tình huống đặt ra là nếu bạn có dữ liệu gồm hai thông tin dạng số thì liệu hai thông tin này có liên quan gì với nhau không? Nếu có liên quan thì có quy luật gì không? Nếu biết một thông tin này thì có thể suy đoán được thông tin kia không? Cụ thể là nếu dùng một đường thẳng dạng `y = ax + b` trong đó x và y đại diện cho hai yếu tố thì các giá trị thực tế của x và y (tức 2 giá trị cụ thể: x_i, y_i) có gần với đường thẳng đó không?

Ví dụ khi đo chiều cao và cân nặng của 24 vận động viên thì có thấy mối liên hệ gì giữa chiều cao và cân nặng không?

Thử plot dữ liệu với code Python:

```
import pandas as pd
import seaborn as sns


df =
pd.read_csv('https://thachln.github.io/datasets/TuyenVN_2019.
csv')
df.columns
sns.relplot(x="Weight", y="Height", data=df)
```



Câu hỏi đặt ra tiếp theo là: Nếu vì lý do nào đó cái cân bị hư, chỉ đo chiều cao của vận động viên thứ 11 thì có thể đoán được cân nặng không?

Hướng tiếp cận:

Mục tiêu của hồi qui tuyến tính là vẽ một đường thẳng xuyên qua dữ liệu sao cho khoảng cách giữa dữ liệu quan sát được gần với đường thẳng nhất. Tức là xây dựng phương trình đường thẳng dạng $y = ax + b$ để khái quát hóa mối quan hệ của cặp giá trị x_i và y_i. Trong đó x_i là một giá trị quan sát cụ thể (observation) và y_i là kết quả dự đoán của quan sát đó.

Nhìn kết quả plot bằng Python chắc bạn có cảm giác là có mối quan hệ giữa chiều cao và cân nặng phải không?

Vì a và b chỉ là ước tính từ dữ liệu đã có nên qui trước chung được kí hiệu là α và β, gọi là ước số. Ngoài ra còn một thông tin nữa là nhiễu, tức là các thông tin không giải thích được, kí hiệu là epsilon (ε).

Phương trình được viết theo ước số là $y = \alpha x + \beta + \varepsilon$

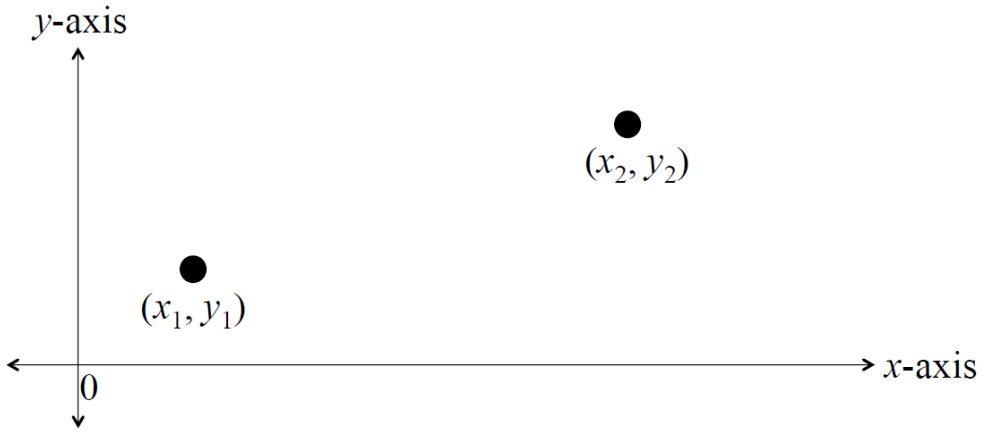
- β gọi là **Intercept**. Tức là khi $x = 0$ thì $y = \beta$, khi vẽ đồ thị lên hệ trục Decarter thì đường thẳng đi qua điểm có hoành độ = 0, và tung độ là β.

- α gọi là **Slope**, có nghĩa là độ dốc của đường thẳng. Có nghĩa là α càng lớn thì đồ thị có xu hướng vọt thẳng lên trời. Trong một số ngữ cảnh khác như Machine Learning chẳng hạn thì slope gọi là weight (trọng số)

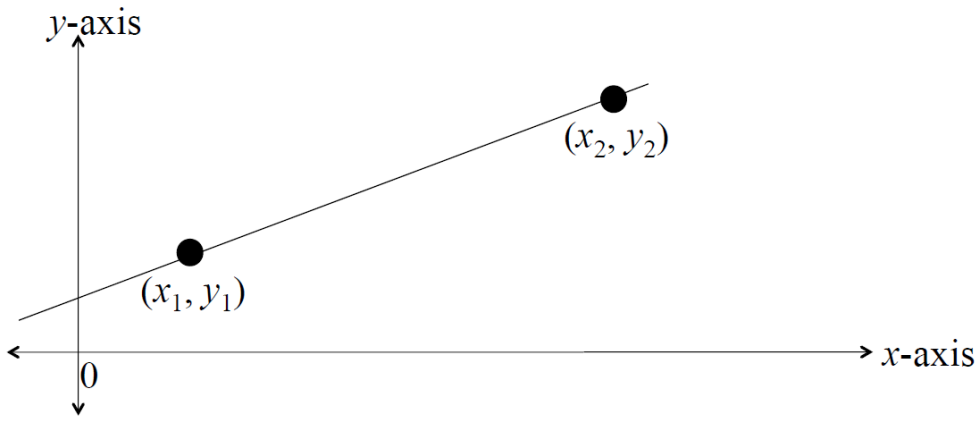
Làm sao để xác định được a và b với data frame cho trước nhỉ?

Một chút lý thuyết:

Nếu cho hai điểm (x1, y1) và (x2, y2) như hình bên dưới



thì với kiến thức toán phổ thông bạn không khó để viết một phương trình nối hai điểm này.



Tính **slope** (có sách viết là **gradient**, slope và gradient có nghĩa như nhau: độ dốc, độ nghiêng) bằng:

$$\text{slope} = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

Tính giá trị khởi đầu intercept của y khi x = 0.

Code Python

Xây dựng mô hình

```
import pandas as pd
import seaborn as sns
from sklearn.linear_model import LinearRegression
```

```
df =
pd.read_csv('https://thachln.github.io/datasets/TuyenVN_2019.
csv')
df.columns
sns.relplot(x='Weight', y='Height', data=df)


model = LinearRegression()
model.fit(df[['Weight']], df[['Height']])


LinearRegression(copy_X = True, fit_intercept = True, n_jobs
= None, normalize = False)


intercept = model.intercept_
print('intercept = {}'.format(intercept))


coefficent = model.coef_
print('coefficient = {}'.format(coefficent))
```

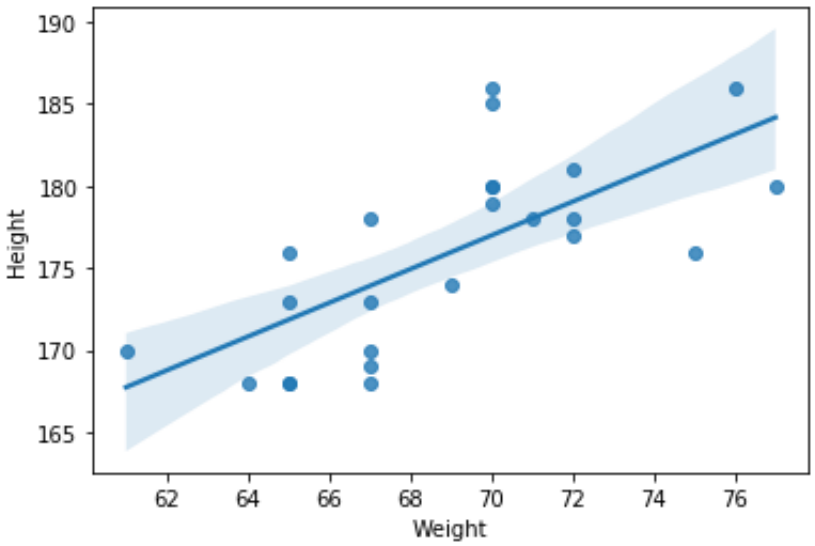
Kết quả:

```
intercept = [105.04843968]
coefficient = [[1.02771309]]
```

Thư viện `sklearn` sử dụng thuật ngữ coefficient (hệ số tương quan) có nghĩa là hệ số tương quan giữa biến `Height` và `Weight`, chính là slope.

Để vẽ thêm đường hồi qui thì thay bằng hàm `relplot(..)` ở dòng số 6 thành hàm `regplot`:

```
sns.regplot(x='Weight', y='Height', data=df)
```



Lưu mô hình

Sử dụng hàm `open('Đường dẫn file','wb')` để tạo file mới và thư viện `pickle` để lưu mô hình vào file.

```
import pickle
pickle.dump(model, open('TuyenVN_2019_LinearRegression.pkl',
'wb'))
```

Nạp và sử dụng lại mô hình đã lưu

Sử dụng hàm `open('Đường dẫn file','rb')` để đọc file và thư viện `pickle.load` để nạp mô hình.

```
model = pickle.load(open('TuyenVN_2019_LinearRegression.pkl',
'rb'))
```

Bài 23: Diễn giải mô hình hồi qui tuyến tính

Tiếp tục mô hình hồi qui tuyến tính của bài trước, phần này sử dụng thư viện `statsmodels` trong Python để phân tích mô hình:

```
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
df = pd.read_csv('https://thachln.github.io/datasets/TuyenVN_2019.csv')

# generate model for linear regression
my_model = smf.ols(formula='Height ~ Weight', data=df)

# fit model to data to obtain parameter estimates
my_model_fit = my_model.fit()

# print summary of linear regression
print(my_model_fit.summary())

# show anova table
anova_table = sm.stats.anova_lm(my_model_fit, typ=2)
print(anova_table)
```

```
                            OLS Regression Results
==========================================Dep.              Variable:
Height   R-squared:                       0.488
Model:                            OLS   Adj. R-squared:                  0
.465
Method:                 Least Squares   F-statistic:                     2
0.96
Date:                Fri, 16 Apr 2021   Prob (F-statistic):              0
.000147
Time:                        11:34:07   Log-Likelihood:                 -
67.716
No. Observations:                  24   AIC:                             1
39.4
Df Residuals:                      22   BIC:                             1
41.8
Df Model:                           1
Covariance Type:            nonrobust
========================================                 coef    s
td err          t      P>|t|      [0.025      0.975]
```

```
------------------------------------------------------------------------------
Intercept      105.0484      15.494       6.780       0.000       72.915
137.182
Weight           1.0277       0.224       4.578       0.000        0.562
1.493
==============================================Omnibus:
1.023   Durbin-Watson:                   1.752
Prob(Omnibus):                  0.600   Jarque-Bera (JB):                0
.990
Skew:                           0.404   Prob(JB):                        0
.610
Kurtosis:                       2.419   Cond. No.                        1
.23e+03
==========================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors
is correctly specified.
[2] The condition number is large, 1.23e+03. This might indicate tha
t there are
strong multicollinearity or other numerical problems.
               sum_sq    df          F    PR(>F)
Weight     377.941488   1.0  20.960571  0.000147
Residual   396.683512  22.0        NaN       NaN
```

Phần kết quả thứ hai (cột "coef") trình bày kết quả hệ số hồi qui. Estimate là ước số. Trong đó Intercept (hệ số β trong phương trình y = αx + β) là 105.0484. Weight bằng 1.0277 chính là trọng số của biến chiều cao, là α trong phương trình.

R-squared

Mô hình giải thích được 48.79% phương sai của Chiều cao. Tức là chỉ cần một biến Cân nặng có thể giải thích 48.79% dao động của Chiều cao.

Ngoài ra còn có các chỉ số sau:

- Adjusted R-squared
- F-statistic (F-test)
- Residuals – độ dao động dư

Để hiểu được các chỉ số trên thì cần Phân tích phương sai một chút.

Phân tích phương sai

Quay lại biểu đồ mối quan hệ giữa chiều cao và cân nặng của cầu thủ. Có thể nhìn mô hình bằng công thức sau:

```
Dao động quan sát = giá trị của mô hình + nhiễu
```

Dao động quan sát: observed variation

Giá trị của mô hình: model

Nhiễu: (random) thông tin ngẫu nhiên mà mô hình không giải thích được.

Variation ở đây chính là dao động. Dao động được tính là tổng các bình phương độ lệch của các giá trị so với giá trị trung bình. Gọi ngắn ngọn là tổng bình phương, tức là **sum of squares**. Kí hiệu là SS_{total}.

Model ở đây là tổng bình phương có thể giải thích bằng mô hình hồi qui. Kí hiệu là SS_{reg}.

Random là tổng bình phương mà không thể giải thích được. Kí hiệu là SS_{error}.

Viết lại công thức ngắn gọn như sau:

$$SS_{total} = SS_{reg} + SS_{error}$$

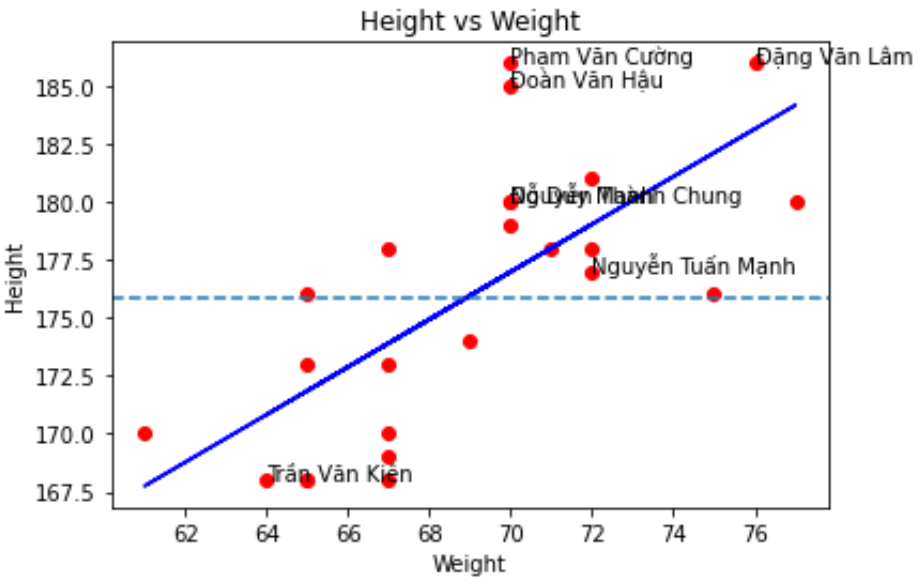
Vẽ lại biểu đồ tương quan giữa Chiều cao và Cân nặng với các chấm đỏ dùng hàm scatter; Plot các giá trị cân nặng và chiều cao được tiên lượng bằng đường màu xanh. Trong đó có vẽ thêm đường trung bình:

```
import pandas as pd
from sklearn.linear_model import LinearRegression


df =
pd.read_csv('https://thachln.github.io/datasets/TuyenVN_2019.
csv')


model = LinearRegression()
model.fit(df[['Weight']], df[['Height']])


LinearRegression(copy_X = True, fit_intercept = True, n_jobs
= None, normalize = False)


intercept = model.intercept_
print('intercept = {}'.format(intercept))


coefficent = model.coef_
print('coefficient = {}'.format(coefficent))


r_sq = model.score(df[['Weight']], df[['Height']])
print('coefficient of determination:', r_sq)
```

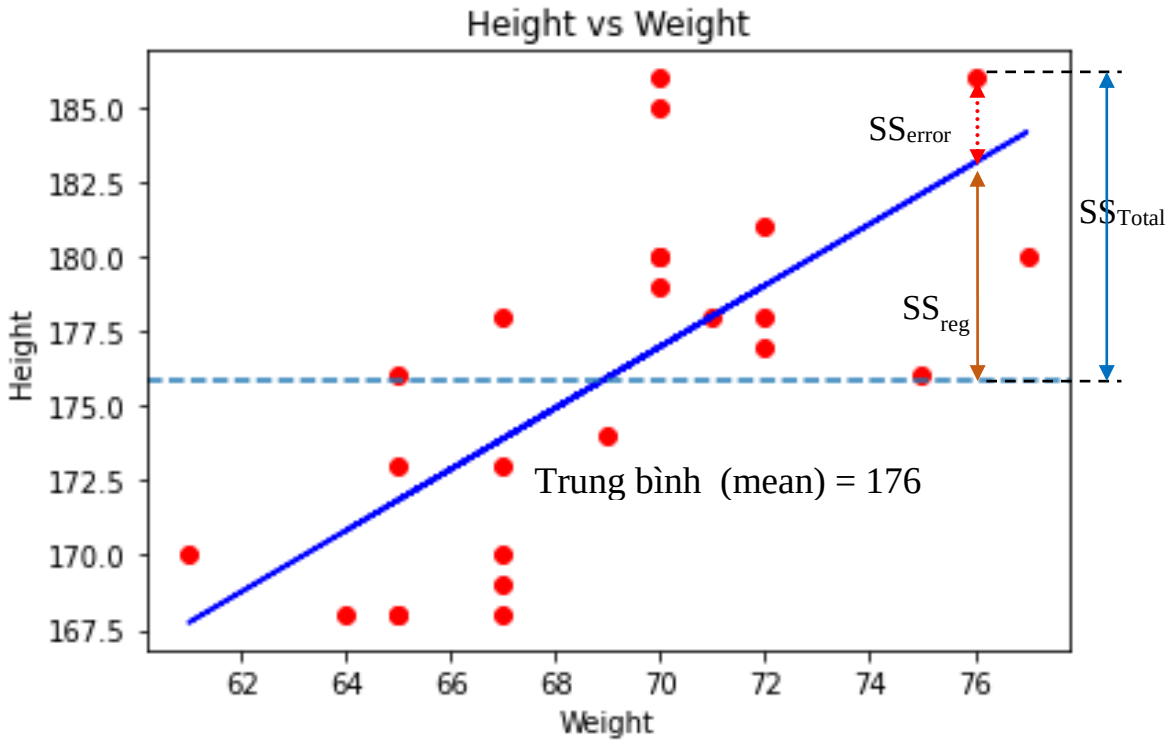
```
import matplotlib.pyplot as plt
import numpy as np
plt.scatter(df[['Weight']], df[['Height']], color = 'red')

plt.plot(df[['Weight']], model.predict(df[['Weight']]), color
= 'blue')

mean = np.mean(df['Height'])

plt.axhline(y=mean, linestyle='--')

plt.title('Height vs Weight')
plt.xlabel('Weight')
plt.ylabel('Height')
plt.show()
```

Thêm lệnh text để hiển thị thêm tên của cầu thủ:

```
for i, item in enumerate(df):
    name = df.loc[i]['Name']
    x = df.iloc[i]['Weight']
    y = df.iloc[i]['Height']
    plt.annotate(name, (x, y))
```



Nếu quan sát dữ liệu của Đặng Văn Lâm (Cao 186 cm, Nặng 76 Kg) thì điểm quan sát này không nằm trên đường hồi qui (không nằm trên mô hình).



Chiều cao thật sự của Đặng Văn Lâm là SS_{Total} = 186cm. Đường trung bình là 176 cm (đã làm tròn). Phần dao động so với đường trung bình là 186 – 176 = 10 cm.

Hình dung 10 cm này tượng trưng bằng đoạn SS_{Total}. SS_{Total} này gồm hai phần:

- Phần giải thích bằng mô hình là SS_{reg}. SSreg tượng trưng bằng = 183 - 176 = 7 cm. Trong đó 183 là giá trị y trên đường hồi qui.
- Phần mô hình không giải thích được (random error) là SS_{error}. SS_{error} tượng trưng bằng 186 - 183 = 3cm

Số liệu ở trên được tính là tượng trưng là vì nó không phải giá trị chính xác như vậy do SS_{Total}, SS_{reg}, SS_{error} tính bằng tổng bình phương (SS: Sum Squared). "Tượng trưng" để bạn hình dung công thức đang bàn ở trên thôi. Hy vọng bạn đã hình dung phần nào vấn đề và mối liên hệ của công thức.

Tính chiều cao của Đặng Văn Lâm theo mô hình hồi qui bằng cách lấy dòng dữ liệu của Lâm dựa và biến BirthPlace không có dấu (dữ liệu chỉ có Lâm là sinh ra ở Nga). Biến Name có dấu nên thư viện `dplyr` không hỗ trợ so sánh trực tiếp được.

```
df1 = df[df['BirthPlace'] == "Nga"]
model.predict(df1[['Weight']]))
```

```
array([[183.15463437]])
```

Phân tích phương sai bằng Python với thư viện `statsmodels`:

```
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
df =
pd.read_csv('https://thachln.github.io/datasets/TuyenVN_2019.
csv')

# generate model for linear regression
my_model = smf.ols(formula='Height ~ Weight', data=df)

# fit model to data to obtain parameter estimates
my_model_fit = my_model.fit()

# print summary of linear regression
print(my_model_fit.summary())

# show anova table
anova_table = sm.stats.anova_lm(my_model_fit, typ=2)
print(anova_table)
```

```
                            OLS Regression Results
==========================================Dep.              Variable:
Height   R-squared:                       0.488
Model:                            OLS   Adj. R-squared:                  0
.465
Method:                 Least Squares   F-statistic:                     2
0.96
Date:                Fri, 16 Apr 2021   Prob (F-statistic):              0
.000147
Time:                        11:34:07   Log-Likelihood:                 -
67.716
No. Observations:                  24   AIC:                             1
39.4
Df Residuals:                      22   BIC:                             1
41.8
Df Model:                           1
Covariance Type:            nonrobust
==========================================                   coef    s
td err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------
----------
```

```
Intercept     105.0484      15.494       6.780       0.000       72.915
137.182
Weight          1.0277       0.224       4.578       0.000        0.562
1.493
==============================================Omnibus:
1.023   Durbin-Watson:                   1.752
Prob(Omnibus):                  0.600   Jarque-Bera (JB):                0
.990
Skew:                           0.404   Prob(JB):                        0
.610
Kurtosis:                       2.419   Cond. No.                        1
.23e+03
==========================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors
is correctly specified.
[2] The condition number is large, 1.23e+03. This might indicate tha
t there are
strong multicollinearity or other numerical problems.
              sum_sq     df          F    PR(>F)
Weight    377.941488    1.0  20.960571  0.000147
Residual  396.683512   22.0        NaN       NaN
```

Diễn giải:

Nguồn dao động do Weight (cân nặng) và Residuals.

Residuals là phần dư chưa giải thích được: 396.68

Cột thứ hai là df, viết tắt của Degree of freedom (bậc tự do). Weight có bậc tự do là 1 do mô hình có 1 biến. 22 bậc tự do của Residuals chính là 22 bậc tự do của Sum Squared Error (SS_{error})

Cột sum_sq là Sum Squared. 377.94 chính là SS_{reg}. 396.68 chính là SS_{error}

$$SS_{Total} = 377.94 + 396.68 = 774.62$$

Tự tính Mean Squared bằng cách lấy Sum Squared chia cho Bậc tự do:

- Mean Squared của Weight = 377.94 / 1 = 377.94
- Mean Squared của Residuals = 396.68 / 22 = **18.03**

Cột F value được tính bằng Mean Squared của Cân Nặng chia cho Mean Squared của Error = 377.94 / 18.03 = **20.96**. Nói cách khác:

$$\text{F value} = \frac{\text{Mean Squared của biến cần phân tích}}{\text{Mean Squared của Residuals}}$$

Mean Squared của biến cần phân tích chính là Tín hiệu.

Mean Squared của Residuals chính là Nhiễu (Noise).

F value chính là F-statistic.

Nói cái khác nữa:

$$\text{F statistic} = \frac{\text{Tính hiệu}}{\text{Nhiễu}}$$

Cột cuối cùng là chỉ số P, rất nhỏ: ý nói mô hình có ý nghĩa.

Tính R Squared

Xác định R^2 (R-squared) = Sum Squared của Cân nặng/ Total Sum Square

= 377.94 / 774.62

≈ **0.49**

R^2 = 0.49 có nghĩa là sự khác biệt về **Cân nặng** giải thích **49%** sự khác biệt của Chiều cao của cầu thủ.

Tính Adjusted R Squared (Adj. R-squared)

Khái niệm Adjusted R^2 (Hệ số hiệu chỉnh) được dùng khi cần thêm biến số để giải thích mô hình. Tức là khi tăng biến số thì % dao động được giải thích sẽ tăng lên. Adjusted R^2 dùng để "phạt" mô hình nhiều biến số.

Cách tính đơn giản:

$$R^2_{adj} = 1 - \frac{MS_{error}}{MS_{total}}$$

MS_{error} (Mean squared error)

MS_{total} (Mean squared total)

MS_{error} = 18.03

MS_{total} = (377.94 + 396.68) / 23 = 33.68

R^2_{adj} = 1 – (18.03 / 33.68) = 0.465

Tính bằng Python

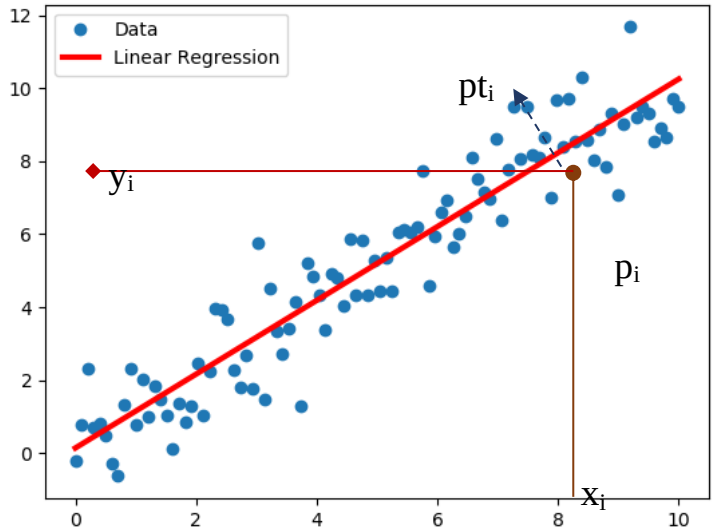
```
import pandas as pd
import statsmodels.formula.api as smf
df =
pd.read_csv('https://thachln.github.io/datasets/TuyenVN_2019.
csv')


# generate model for linear regression
my_model = smf.ols(formula='Height ~ Weight', data=df)


# fit model to data to obtain parameter estimates
m = my_model.fit()
```

```
print('R squared: %.2f' % m.rsquared)
print('Adj. R-squared: %.2f' % m.rsquared_adj)
```

```
R squared: 0.49
Adj. R-squared: 0.46
```

Dữ liệu quan sát = Mô hình tiên lượng + nhiễu

Quan sát lại phương trình hồi quy tuyến tính tổng quát $y = \alpha x + \beta$ và biểu đồ minh họa bên dưới:



Đường màu đỏ (Linear Regression) là biểu diễn của giá trị tiên lượng: nếu cho một giá trị x_i trên trục hoành, bạn kẻ một đường thẳng xuất phát từ x_i trên trục hoành, dóng thẳng lên đụng đường Linear Regression – gọi điểm tiếp xúc này là p_i. Tiếp theo kẻ đường thẳng từ p_i ngang qua trục tung, gặp nhau tại y_i. Tức là nếu cho trước x_i thì có thể suy ra y_i dễ dàng và dữ liệu tiên lượng là p_i.

Giá trị tiên lượng p_i có thể là một giá trị thật nếu bạn may mắn. Tuy nhiên hầu hết các trường hợp là p_i không phải là giá trị thật (vì thế mới gọi là giá trị tiên lượng).

Trong thường hợp x_i có trong bộ dữ liệu đã thu thập thì p_i thật (gọi là pt_i) có thể là điểm tròn gần bên, hoặc ngay tại p_i.

Trong tường hợp x_i không có trong bộ dữ liệu (tức là có trong quần thể mà chúng ta không thu thập để phân tích) thì ptt_i có thể là điểm tròn đâu đó gần với p_i và pt_i.

Dài dòng như vậy để bạn thấy rằng **Giá trị quan sát**, hoặc **Giá trị thực tế** hiếm khi bằng **Giá trị tiên lượng**. Chính vì điều này thì người ta mới gọi là **Mô hình** và đòi hỏi nhiều dữ liệu để huấn luyện mô hình, như thế Big Data mới có ý nghĩa. Vì nếu giá trị tiên tượng hoàn hảo, chính xác 100% thì lúc này **Mô hình** sẽ chuyển thành **Công thức toán**.

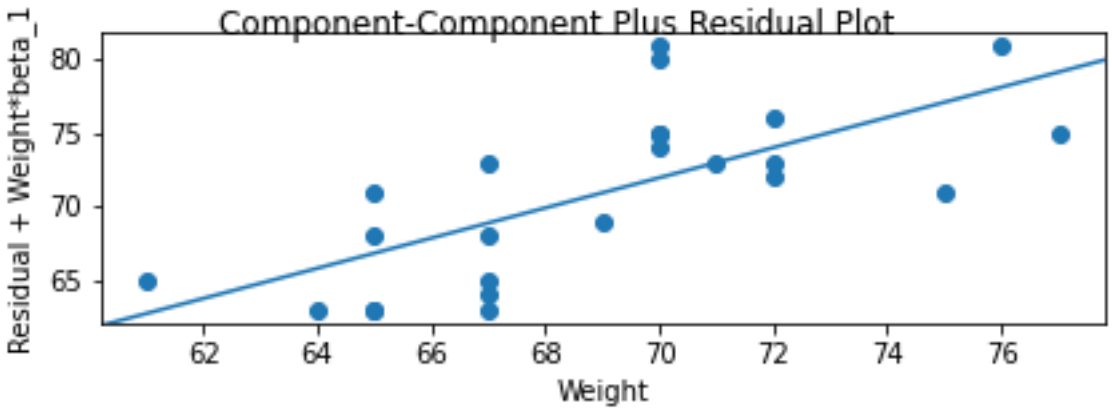
Độ lệch giữa **Giá trị quan sát được** và **Giá trị tiên lượng** gọi là **Phần dư**. Phần dư (residual) còn gọi là nhiễu (noise) hoặc **loss** (dùng trong ngữ cảnh Machine Learning).

Ta có công thức để nhớ:

```
Dữ liệu quan sát = Mô hình tiên lượng + Nhiễu
```

Quay lại tập dữ liệu 24 tuyển thủ trong đội tuyển bóng đá Nam.

```
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
df =
pd.read_csv('https://thachln.github.io/datasets/TuyenVN_2019.
csv')

# generate model for linear regression
my_model = smf.ols(formula='Height ~ Weight', data=df)

# fit model to data to obtain parameter estimates
m = my_model.fit()

sm.graphics.plot_ccpr_grid(m)
```



- 24 chấm tròn là 24 dữ liệu chiều cao và cân nặng của 24 cầu thủ mà ta thu thập: gọi là **dữ liệu quan sát**.
- Đường kẻ màu xanh là đường hồi quy tuyến tính.

Tham khảo:

```
https://www.statsmodels.org/stable/examples/notebooks/generated/regression_plots.html
```

Sử dụng lớp `OLSInfluence` để tính toán phần dư bằng cách tính độ lệch **giá trị quan sát** (chấm tròn) và giá trị tiên lượng (trên đường màu xanh).

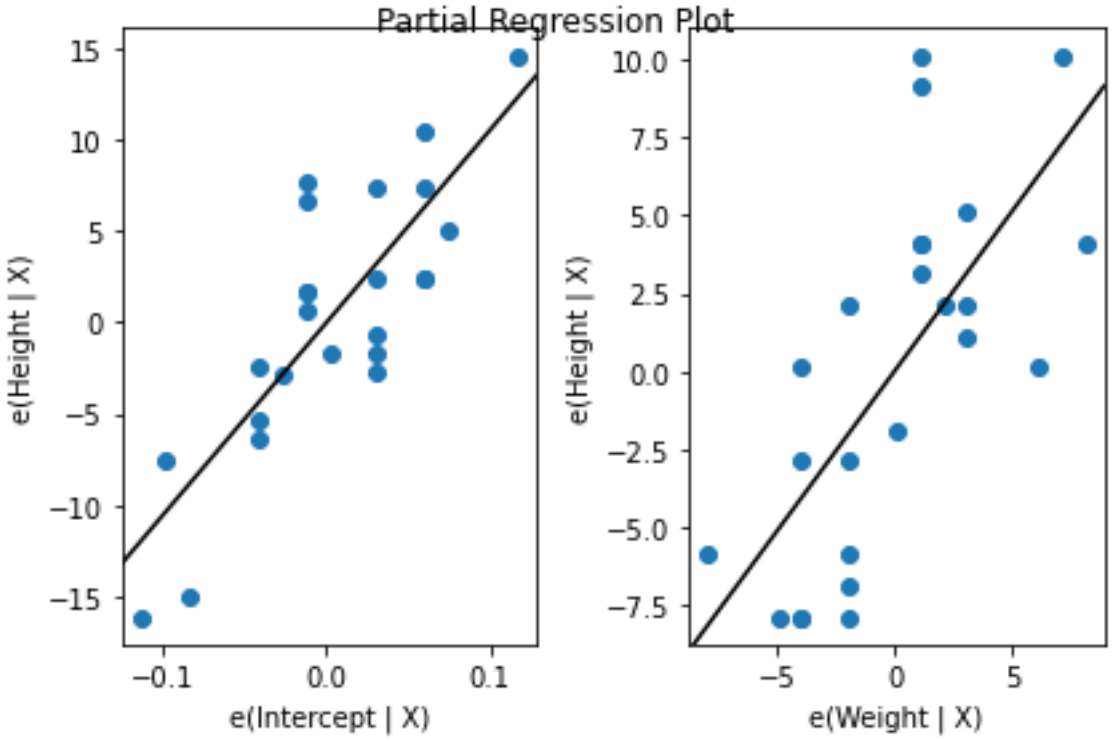
```
from statsmodels.stats.outliers_influence import OLSInfluence
# compute the residuals and other metrics
influence = OLSInfluence(m)
influence_summary_frame = influence.summary_frame()
print(influence_summary_frame)
```

Kết quả là danh sách 24 độ lệch:

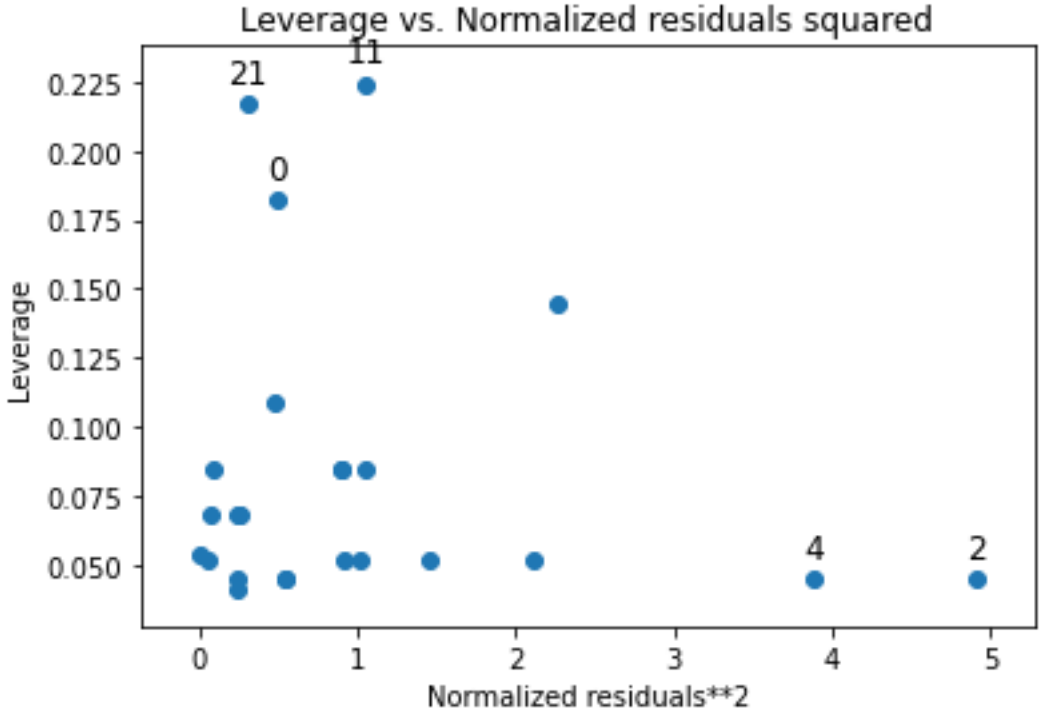
```
    dfb_Intercept  dfb_Weight  ...  student_resid    dffits
0       -0.293725    0.303454  ...       0.732999  0.345613
1        0.076805   -0.082730  ...      -0.489933 -0.132583
2       -0.112084    0.140273  ...       2.393674  0.519277
3       -0.033606    0.042058  ...       0.717694  0.155694
4       -0.096913    0.121287  ...       2.069690  0.448992
5       -0.199751    0.191632  ...      -0.695860 -0.243665
6       -0.033606    0.042058  ...       0.717694  0.155694
7       -0.024760    0.022283  ...      -0.214138 -0.050118
8       -0.022296    0.027904  ...       0.476164  0.103298
9       -0.138527    0.124672  ...      -1.198073 -0.280405
10       0.541472   -0.562319  ...      -1.616749 -0.666031
11       0.530385   -0.545825  ...      -1.125018 -0.604903
12       0.063177   -0.059964  ...       0.277100  0.084205
13      -0.108932    0.098037  ...      -0.942112 -0.220498
14      -0.215515    0.204554  ...      -0.945262 -0.287246
15       0.233138   -0.221281  ...       1.022560  0.310735
16       0.039060   -0.042074  ...      -0.249164 -0.067427
17      -0.169390    0.152448  ...      -1.464988 -0.342875
18      -0.003322   -0.002084  ...      -0.463169 -0.096601
19      -0.073479    0.079148  ...       0.468718  0.126841
20       0.114460   -0.103012  ...       0.989926  0.231689
21       0.287520   -0.280311  ...       0.592749  0.311876
22      -0.215515    0.204554  ...      -0.945262 -0.287246
23       0.000385   -0.000430  ...      -0.003801 -0.000906

[24 rows x 8 columns]
```

Partial Regression Plots

```
sm.graphics.plot_partregress_grid(m)
```




```
sm.graphics.plot_leverage_resid2(m)
```



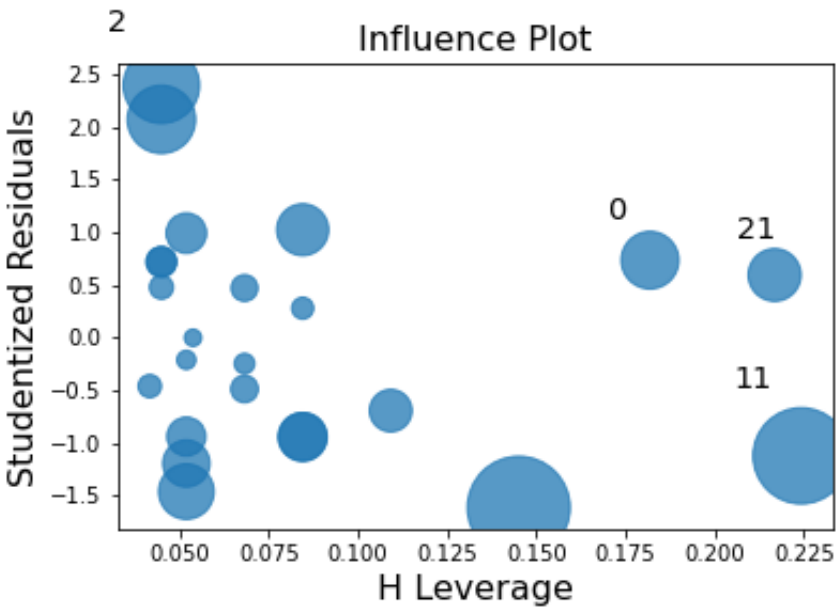
Influential observations

```
from statsmodels.stats.outliers_influence import OLSInfluence
# compute the residuals and other metrics
influence = OLSInfluence(m)
influence_summary_frame = influence.summary_frame()
print(influence_summary_frame)
```

```
    dfb_Intercept  dfb_Weight  ...  student_resid    dffits
0       -0.293725    0.303454  ...       0.732999  0.345613
1        0.076805   -0.082730  ...      -0.489933 -0.132583
2       -0.112084    0.140273  ...       2.393674  0.519277
3       -0.033606    0.042058  ...       0.717694  0.155694
4       -0.096913    0.121287  ...       2.069690  0.448992
5       -0.199751    0.191632  ...      -0.695860 -0.243665
6       -0.033606    0.042058  ...       0.717694  0.155694
7       -0.024760    0.022283  ...      -0.214138 -0.050118
8       -0.022296    0.027904  ...       0.476164  0.103298
9       -0.138527    0.124672  ...      -1.198073 -0.280405
10       0.541472   -0.562319  ...      -1.616749 -0.666031
11       0.530385   -0.545825  ...      -1.125018 -0.604903
12       0.063177   -0.059964  ...       0.277100  0.084205
13      -0.108932    0.098037  ...      -0.942112 -0.220498
14      -0.215515    0.204554  ...      -0.945262 -0.287246
15       0.233138   -0.221281  ...       1.022560  0.310735
16       0.039060   -0.042074  ...      -0.249164 -0.067427
17      -0.169390    0.152448  ...      -1.464988 -0.342875
18      -0.003322   -0.002084  ...      -0.463169 -0.096601
19      -0.073479    0.079148  ...       0.468718  0.126841
20       0.114460   -0.103012  ...       0.989926  0.231689
21       0.287520   -0.280311  ...       0.592749  0.311876
22      -0.215515    0.204554  ...      -0.945262 -0.287246
23       0.000385   -0.000430  ...      -0.003801 -0.000906

[24 rows x 8 columns]
```

Influence Plot

```
sm.graphics.influence_plot(m)
```


Influence Plot
Studentized Residuals
H Leverage
2
0
21
11
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.050
0.075
0.100
0.125
0.150
0.175
0.200
0.225

Bài 24: Mô hình hồi qui tuyến tính đa biến

Trong bài trước chúng ta đã bàn mô hình hồi qui tuyến tính đơn biến, tức là chỉ có một biến tiên lượng. Trong thực tế có nhiều yếu tố ảnh hưởng đến đến thông tin đầu ra như:

- Khi xây dựng đội ngũ nhân sự vì việc tuyển chọn người dựa vào rất nhiều yếu tố. Ví dụ cần đánh giá Thái độ, Kỹ năng, Kiến thức theo tam giác ASK. Mỗi mục như vậy cần nhiều câu hỏi và tình huống để ứng viên trả lời và xử lý.

- Chế độ ăn uống có liên quan đến bệnh ung thư? Món ăn nào, hoặc cách chế biến nào là yếu tố liên quan?

Bài này bàn đến các mục tiêu của mô hình hồi qui đa biến như:

- ✓ Tiên lượng (prediction)
- ✓ Đánh giá ảnh hưởng của biến đầu vào
- ✓ Kiểm soát

Khái niệm và mô hình

Ta có nhiều dữ liệu đầu vào kí hiệu là các biến X_1, X_2,…X_p. Mỗi biến sẽ có các hệ số tương tứng $\beta_1, \beta_2, \ldots, \beta_p$.

Mô hình được viết dưới dạng

$$Y = \beta_0 + X_1\beta_1 + X_2\beta_2 + \ldots + X_p\beta_p + \varepsilon$$

Trong đó Y là biến phụ thuộc, thông tin đầu ra hay còn gọi là outcome, hoặc target. Y là biến liên tục.

X_1, X_2,…X_p là biến tiên tượng, thông tin đầu vào hay còn gọi là đặc trưng (features).

Các giá trị beta $\beta_1, \beta_2, \ldots, \beta_p$ là trọng số của các biến tiên lượng, hay còn gọi là hệ số hồi qui (regression coefficients)

Giá trị epsilon ε là sai số ngẫu nhiên.

Giả định:

- Mối liên hệ giữa outcome và biến tiên lượng là quan hệ tuyến tính
- ε tuân theo luật phân bố chuẩn (normal)

Tham số

Làm cách nào để ước tính các giá trị beta này. Vì thực tế là không bao giờ chúng ta biết chính xác các giá trị beta này mà chỉ có thể ước lượng dựa vào dữ liệu chúng ta có.

Vì vậy thời đại này gọi là Big Data là như vậy. Khi có dữ liệu càng nhiều thì việc ước lượng càng chính xác. Giá trị được tính toán dựa trên số liệu này gọi là ước số: b_1, b_2,...b_p

Phương trình được viết lại theo dữ liệu như sau:

$$\hat{Y} = b_0 + X_1 b_1 + X_2 b_2 + \ldots + X_p b_p$$

e là phần dư (residual), tính bằng Y - $\hat{Y}$

$\hat{Y}$ đọc là Y hat (hat là cái mũ, dấu mũ trên đầu chữ y).

Nhắc lại mô hình căn bản:

$$Y = \beta_0 + X_1 \beta_1 + X_2 \beta_2 + \ldots + X_p \beta_p + \varepsilon$$

Phương pháp tính tham số

Phương pháp bình phương tối thiểu (least squares method)

Hàm least squares: tính tổng bình phương các phần dư:

$$L = \sum_{i=1}^{p} e_i^2 = \sum_{i=1}^{p} (y_i - \beta_0 - \sum_{j=1}^{k} \beta_j x_{ji})^2$$

Ý tưởng là đi tìm các hệ số β sao cho L nhỏ nhất.

Triển khai bằng Python

Đọc dữ liệu

Sử dụng dữ liệu mẫu trong nghiên cứu tưởng tượng về sức khỏe:

```
import pandas as pd
import statsmodels.formula.api as smf
df =
pd.read_csv('https://thachln.github.io/datasets/sample_health
_vn.csv')


df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3960 entries, 0 to 3959
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
```

```
 0   id      3960 non-null   int64
 1   age     3960 non-null   int64
 2   sex     3960 non-null   int64
 3   height  3960 non-null   int64
 4   waist   3960 non-null   int64
 5   risk    3960 non-null   float64
 6   weight  3960 non-null   int64
 7   hit     3960 non-null   int64
 8   life    3960 non-null   float64
dtypes: float64(2), int64(7)
memory usage: 278.6 KB
```

Tính tỉ số vòng eo và chiều cao

```
df['whtr'] = df['waist'] / df['height']
```

Chuyển kiểu dữ liệu giới tính từ số sang dạng danh mục

```
df['sex'] = df['sex'].astype('category')
```

Tạo mô hình hồi qui tuyến tính đa biến

```
my_model = smf.ols(formula='life ~ age + sex + height + waist
+ risk + weight + hit + whtr', data=df)
```

Fit mô hình để ước tính các tham số

```
m = my_model.fit()
```

Xem thông tin mô hình

```
print(m.summary())
```

```
                            OLS Regression Results
==========================================
Dep. Variable:                    life   R-squared:          0.994
Model:                             OLS   Adj. R-squared:     0.994
Method:                  Least Squares   F-statistic:    8.724e+04
Date:                 Mon, 19 Apr 2021   Prob (F-statistic):  0.00
Time:                         19:30:28   Log-Likelihood:   -1276.9
No. Observations:                3960   AIC:                 2572.
Df Residuals:                    3951   BIC:                 2628.
Df Model:                           8
Covariance Type:            nonrobust
==========================================
               coef    std err    t      P>|t|    [0.025    0.975]
```

```
------------------------------------------------------------------------------
Intercept    74.7145     1.748     42.739      0.000     71.287     78.142
sex[T.1]     -0.0495     0.022     -2.222      0.026     -0.093     -0.006
age           0.0012     0.000      2.637      0.008      0.000      0.002
height        0.1253     0.011     11.506      0.000      0.104      0.147
waist         0.3198     0.021     14.987      0.000      0.278      0.362
risk         -0.0089     0.014     -0.633      0.527     -0.037      0.019
weight       -0.3852     0.001   -716.264      0.000     -0.386     -0.384
hit          -0.0024     0.001     -2.335      0.020     -0.004     -0.000
whtr        -51.3537     3.430    -14.970      0.000    -58.079    -44.628
==============================================
Omnibus:                      391.130   Durbin-Watson:         1.996
Prob(Omnibus):                  0.000   Jarque-Bera (JB):   2664.850
Skew:                          -0.187   Prob(JB):              0.00
Kurtosis:                       7.001   Cond. No.           1.54e+05
==============================================
Notes:
[1] Standard Errors assume that the covariance matrix of the errors
is correctly specified.
[2] The condition number is large, 1.54e+05. This might indicate tha
t there are
strong multicollinearity or other numerical problems.
```

Mã nguồn đầy đủ và có thêm cách lấy các chỉ số AIC, BIC, R-squared, Adjusted R-squared:

```
import pandas as pd
import statsmodels.formula.api as smf
df =
pd.read_csv('https://thachln.github.io/datasets/sample_health
_vn.csv')


df.info()


df['whtr'] = df['waist'] / df['height']
df['sex'] = df['sex'].astype('category')
my_model = smf.ols(formula='life ~ age + sex + height + waist
+ risk + weight + hit + whtr', data=df)
m = my_model.fit()
print(m.summary())


print('R squared: %.2f' % m.rsquared)
print('Adj. R-squared: %.2f' % m.rsquared_adj)
print('AIC= %.2f' % m.aic)
```

```
print('BIC= %.2f' % m.bic)
```

Lưu model

Sử dụng thư viện joblib

Sử dụng thư viện `joblib` để lưu mô hình:

```
import joblib

joblib.dump(m,
'linear_regression_model_sample_health_vn.pkl')
```

Sử dụng thư viện pickle

Pickle là module dùng để lưu trữ và khởi tạo (serializing and de-serializing) đối tượng trong Python. Pickle được dùng cho nhiều mục đích khác nhau. Ở đây chỉ quan tâm đến mục đích lưu mô hình mà chúng ta đã xây dựng ở trên. Cách sử dụng như sau:

```
import pickle

pickle.dump(m,
open('linear_regression_model_sample_health_vn.pkl', 'wb'))
```

Load model

Sử dụng thư viện joblib

Sử dụng hàm open('*filepath*', 'rb') để đọc file vào model. Sau đó dùng hàm `joblib.load(model)` để nạp file model.

```
import joblib

model = open('linear_regression_model_sample_health_vn.pkl',
'rb')


m = joblib.load(model)

print(m.summary())

print('R squared: %.2f' % m.rsquared)

print('Adj. R-squared: %.2f' % m.rsquared_adj)

print('AIC= %.2f' % m.aic)

print('BIC= %.2f' % m.bic)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   life   R-squared:                       0
.994
Model:                            OLS   Adj. R-squared:                  0
.994
```

```
Method:                 Least Squares   F-statistic:                8
.724e+04
Date:             Thu, 10 Jun 2021   Prob (F-statistic):             0
.00
Time:                     14:20:48   Log-Likelihood:                -
1276.9
No. Observations:             3960   AIC:                           2
572.
Df Residuals:                 3951   BIC:                           2
628.
Df Model:                        8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
Intercept     74.7145      1.748     42.739      0.000      71.287
78.142
sex[T.1]      -0.0495      0.022     -2.222      0.026      -0.093
-0.006
age            0.0012      0.000      2.637      0.008       0.000
0.002
height         0.1253      0.011     11.506      0.000       0.104
0.147
waist          0.3198      0.021     14.987      0.000       0.278
0.362
risk          -0.0089      0.014     -0.633      0.527      -0.037
0.019
weight        -0.3852      0.001   -716.264      0.000      -0.386
-0.384
hit           -0.0024      0.001     -2.335      0.020      -0.004
-0.000
whtr         -51.3537      3.430    -14.970      0.000     -58.079
-44.628
==============================================================================
Omnibus:                      391.130   Durbin-Watson:                 1
.996
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              2
664.850
Skew:                          -0.187   Prob(JB):                      0
.00
Kurtosis:                       7.001   Cond. No.                      1
.54e+05
==============================================================================


Notes:
[1] Standard Errors assume that the covariance matrix of the errors
is correctly specified.
```

```
[2] The condition number is large, 1.54e+05. This might indicate tha
t there are
strong multicollinearity or other numerical problems.
R squared: 0.99
Adj. R-squared: 0.99
AIC= 2571.89
BIC= 2628.45
```

Sử dụng thư viện pickle

```
import pickle


m = pickle.load(
open('linear_regression_model_sample_health_vn.pkl', 'rb'))


print(m.summary())
print('R squared: %.2f' % m.rsquared)
print('Adj. R-squared: %.2f' % m.rsquared_adj)
print('AIC= %.2f' % m.aic)
print('BIC= %.2f' % m.bic)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   life   R-squared:                       0
.994
Model:                            OLS   Adj. R-squared:                  0
.994
Method:                 Least Squares   F-statistic:                     8
.724e+04
Date:                Thu, 10 Jun 2021   Prob (F-statistic):              0
.00
Time:                        19:40:42   Log-Likelihood:                -
1276.9
No. Observations:                3960   AIC:                             2
572.
Df Residuals:                    3951   BIC:                             2
628.
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
Intercept     74.7145      1.748     42.739      0.000      71.287
78.142
```

```
sex[T.1]       -0.0495       0.022      -2.222       0.026       -0.093
-0.006
age             0.0012       0.000       2.637       0.008        0.000
0.002
height          0.1253       0.011      11.506       0.000        0.104
0.147
waist           0.3198       0.021      14.987       0.000        0.278
0.362
risk           -0.0089       0.014      -0.633       0.527       -0.037
0.019
weight         -0.3852       0.001    -716.264       0.000       -0.386
-0.384
hit            -0.0024       0.001      -2.335       0.020       -0.004
-0.000
whtr          -51.3537       3.430     -14.970       0.000      -58.079
-44.628
==============================================================================
Omnibus:                      391.130   Durbin-Watson:                   1
.996
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                2
664.850
Skew:                          -0.187   Prob(JB):                        0
.00
Kurtosis:                       7.001   Cond. No.                        1
.54e+05
==============================================================================


Notes:
[1] Standard Errors assume that the covariance matrix of the errors
is correctly specified.
[2] The condition number is large, 1.54e+05. This might indicate tha
t there are
strong multicollinearity or other numerical problems.
R squared: 0.99
Adj. R-squared: 0.99
AIC= 2571.89
BIC= 2628.45
```

Bài 25: Khai thác các thư viện hỗ trợ hồi qui tuyến tính

Trong Bài 24, chúng ta đã dùng thư viện "statsmodels" để triển khai mô hình hồi qui tuyến tính. Trong bài này, chúng ta sẽ trải nghiệm thêm vài thư viện phổ biến nữa.

XGBoost

Với dữ liệu đã phân tích trong Bài 14, chương trình sử dụng thư viện xgboost để xây dựng mô hình. Bạn tự phân tích và khám phát thêm nhé!

```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from xgboost import XGBRegressor



df =
pd.read_csv('https://thachln.github.io/datasets/sample_health
_vn.csv')

df.info()

df['whtr'] = df['waist'] / df['height']
df['sex'] = df['sex'].astype('bool')

df.info()
df.head()

X = df[['age', 'sex', 'height', 'waist', 'risk', 'weight',
'hit', 'whtr']]
y = df['life']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3)

#%% predict
# define the model
```

```
m = XGBRegressor(n_estimators=100, eta=0.05)
# evaluate the model
m.fit(X_train, y_train)



predict = m.predict(X_test)
print('Train score: %.3f' % m.score(X_train, y_train))


print('Test score: %.3f' % m.score(X_test, y_test))


# MSE
print('Residual sum of squares: %.3f'  % np.mean((y_test -
predict) ** 2))
# or
print('Mean squared error: %.3f'  %
mean_squared_error(y_test, predict))



# MAE
print('Mean absolute error: %.3f'  %
mean_absolute_error(y_test, predict))



# R^2


print('R^2: %.3f'  % r2_score(y_test, predict))



#%% feature importance
feature_ipt = pd.DataFrame({'Feature_names': X_train.columns,
'Importances': m.feature_importances_})
feature_sort = feature_ipt.sort_values(by='Importances',
ascending=False)


#%% Save model
import pickle
pickle.dump(m, open('XGBRegressor.pkl', 'wb'))
```