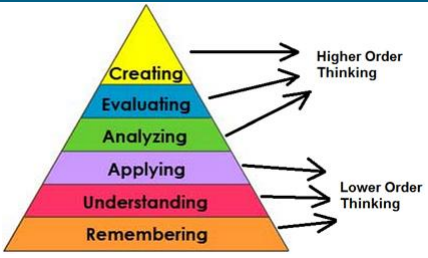
PHÁT TRIỂN MÔ HÌNH

1

Mục tiêu

- Vận dụng hồi quy tuyến tính đơn và đa biến
- Biết đánh giá mô hình dùng trực quan
- Vận dụng hồi quy đa thức và kỹ thuật Pipeline
- Áp dụng thang đo R-squared và MSE dùng để đánh giá tập mẫu



Thang đo BLOOM

2

Nội dung

1. Phát triển mô hình
2. Hồi quy tuyến tính đơn biến
3. Hồi quy tuyến tính đa biến
4. Đánh giá mô hình dùng trực quan
5. Hồi quy đa thức
6. Kỹ thuật Pipeline /ˈpīpˌlīn/
7. Thang đo MSE và R-squared (R^2 – R^2)
8. Đánh giá

Tóm tắt

- **Đặt vấn đề** (Bài 1):

  Chúng ta nên bán xe đã qua sử dụng với giá bao nhiêu?

- **Hướng giải quyết** (Bài 2):

  Tiền xử lý dữ liệu liên quan để phục vụ phân tích.

- **Phân tích thăm dò** (Bài 5):

  Những đặc điểm nào có ảnh hưởng nhất đến giá xe?

- **Phát triển mô hình để dự đoán** (Bài 6):

  Chúng ta xác định giá hợp lý cho xe đã qua sử dụng như thế nào?

Tóm tắt

- **Phân tích thăm dò - EDA** (Bài 5):

  Những đặc điểm nào có ảnh hưởng nhất đến giá xe?

- **Kết quả (tham khảo):** Các biến quan trọng đã thăm dò được tìm thấy là:

Numerical feature	Categorical feature
• curb-weight • engine-size • length • width • horsepower • city-mpg • highway-mpg • wheel-base • bore	• drive-wheels

Bộ dữ liệu

Bộ dữ liệu dùng để phát triển mô hình

Lý thuyết	Thực hành
https://raw.githubusercontent.com/datasethub/ds105/master/Model_Dataset.csv	https://raw.githubusercontent.com/datasethub/ds105/master/Model_Dataset_Lab.csv

1. Phát triển mô hình

Giới thiệu

- Mô hình có thể được xem là một phương trình toán học được dùng để dự đoán (predict) một giá trị hoặc nhiều giá trị.
- Biểu diễn mối liên quan hệ một hoặc nhiều biến độc lập (independent) với các biến phụ thuộc (dependent).

Biến độc lập
'city-mpg'
39

Mô hình

Biến phụ thuộc
'price'
7000

1. Phát triển mô hình

Giới thiệu

- Thông thường, dữ liệu càng phù hợp, liên quan thì mô hình càng chính xác.

'horsepower'
'highway-mpg'
'engine-size'

Mô hình

'price'
5900

- Khi nào dữ liệu là phù hợp?
  => Thu thập => Đạt chuẩn. Vậy, dùng phương pháp gì để kiểm tra?

Padlet

1. Phát triển mô hình

Giới thiệu

- Để hiểu tại sao nhiều dữ liệu (thuộc tính) là quan trọng hơn, hãy xem xét tình huống sau:
  1. Có hai chiếc xe gần như giống hệt nhau.
  2. Thực tế, đôi khi xe màu xanh bán với giá thấp hơn.

'body-style'
'horsepower'
'highway-mpg'
'engine-size'

Mô hình

'price'
5900

'body-style'
'horsepower'
'highway-mpg'
'engine-size'

Mô hình

'price'
7900

1. Phát triển mô hình

Giới thiệu

- Ngoài việc có được nhiều dữ liệu (tốt, phù hợp, có liên quan, đạt chuẩn), chúng ta có thể thử các loại mô hình khác để có kết quả tốt hơn.

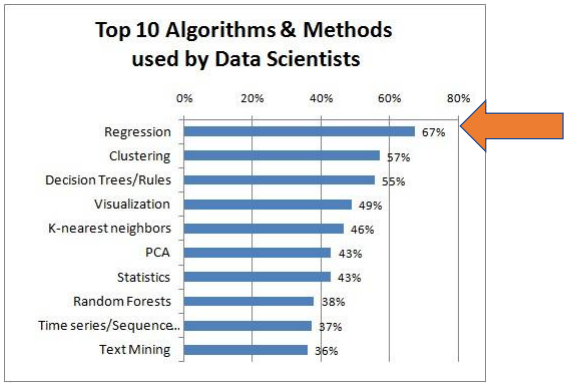
- Trong phần này, chúng ta sẽ tập trung xây dựng các loại mô hình sau:
  1. Hồi quy tuyến tính đơn biến.
  2. Hồi quy tuyến tính đa biến.
  3. Hồi quy đa thức
  4. Kỹ thuật Pipeline

Tại sao chọn hồi quy?

1. Phát triển mô hình

Tại sao chọn hồi quy?

Năm 2016

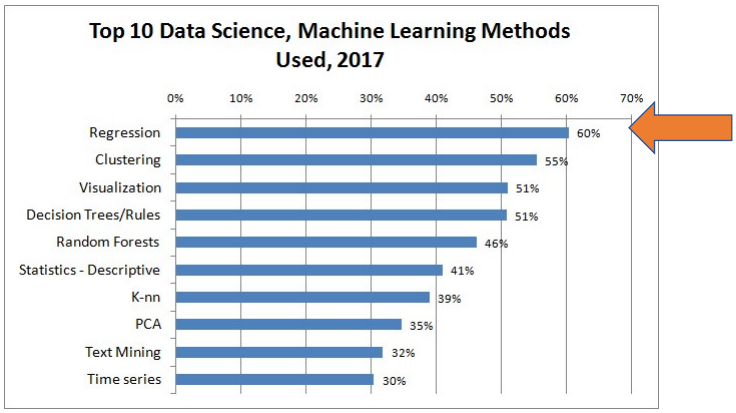


11

1. Phát triển mô hình

Tại sao chọn hồi quy?

Năm 2017

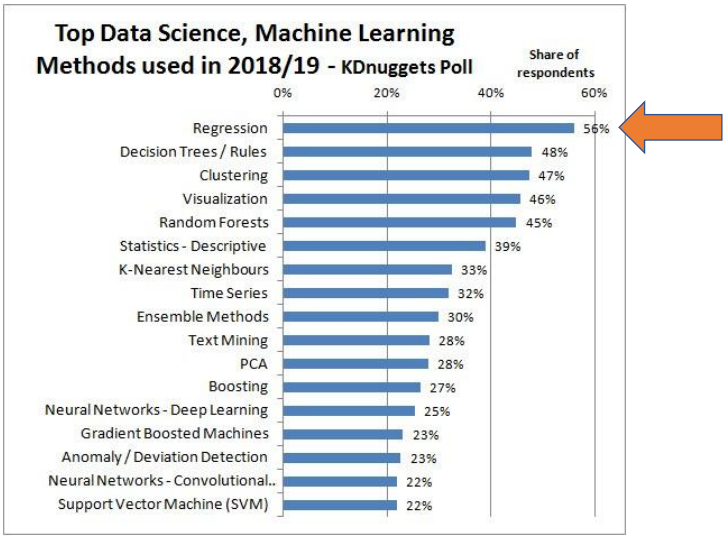


12

1. Phát triển mô hình

Tại sao chọn hồi quy?

Năm 2018-2019

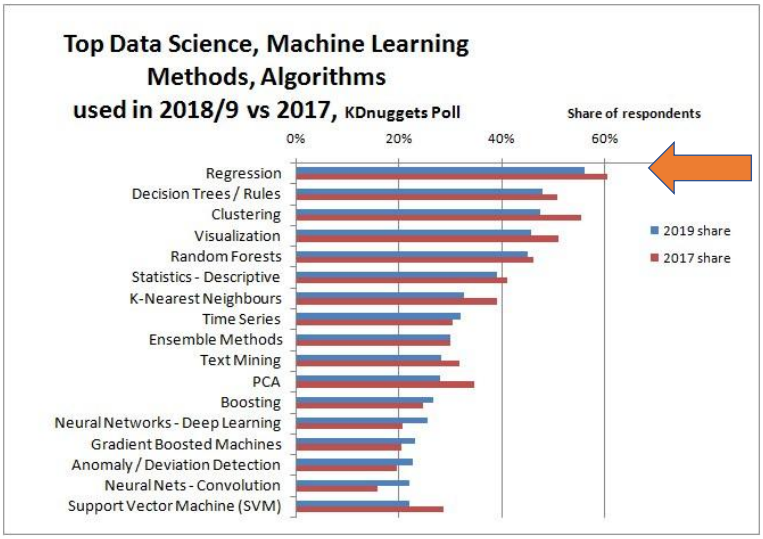


13

1. Phát triển mô hình

Tại sao chọn hồi quy?

So sánh năm 2018-19 và 2017

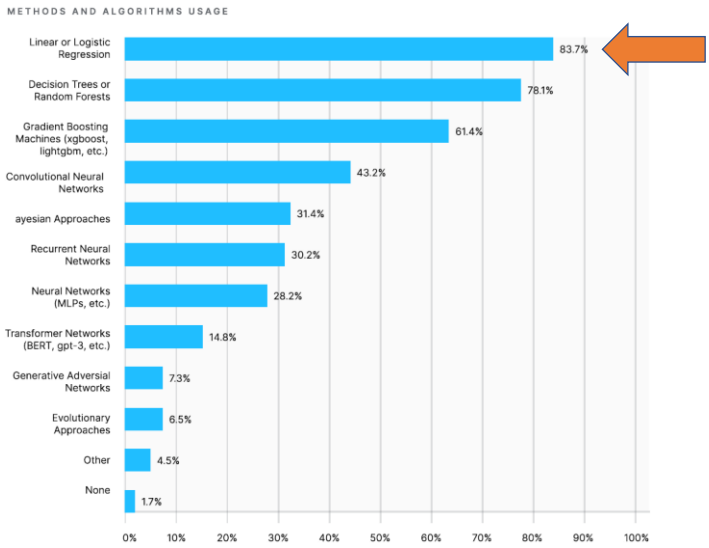


14

1. Phát triển mô hình

Tại sao chọn hồi quy?

Năm 2020

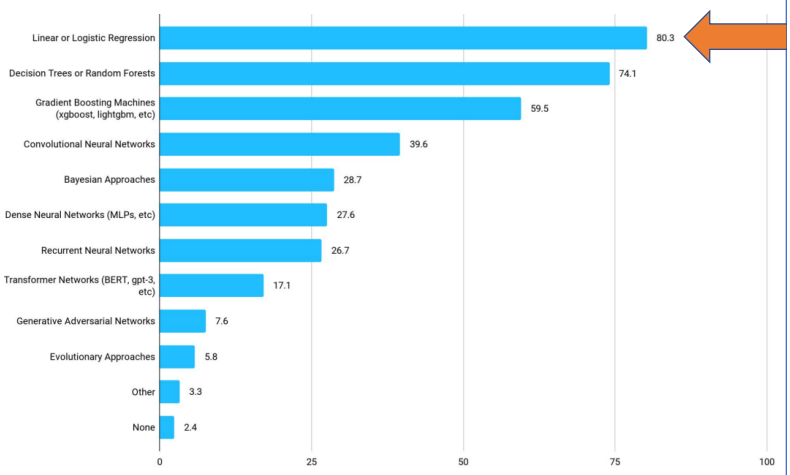


15

1. Phát triển mô hình

Tại sao chọn hồi quy?

Năm 2021



16

1. Phát triển mô hình

Chú ý

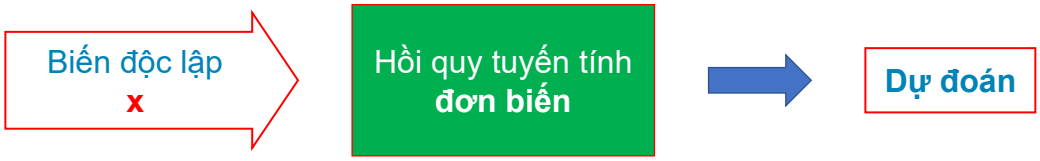
Biến độc lập | Biến phụ thuộc => Target

	feature 1	feature 2	feature 3	feature 4	price	type
0	5.1	3.5	1.4	0.2	4394	2
1	4.9	3.0	1.4	0.2	4336	1
2	4.7	3.2	1.3	0.2	6561	3
3	4.6	3.1	1.5	0.2	4137	2
4	5.0	3.6	1.4	0.2	1762	1
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	8530	3
146	6.3	2.5	5.0	1.9	464	3
147	6.5	3.0	5.2	2.0	2680	4
148	6.2	3.4	5.4	2.3	3591	2
149	5.9	3.0	5.1	1.8	6003	3

2. Hồi quy tuyến tính đơn biến

Giới thiệu

- Hồi quy tuyến tính đơn biến sẽ dùng một biến độc lập để đưa ra dự đoán.



2. Hồi quy tuyến tính đơn biến

Mô hình - Model

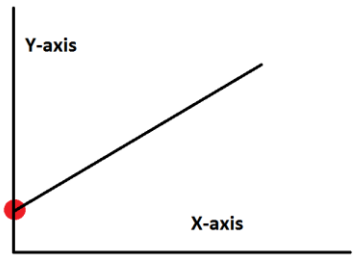
- **x**: Biến độc lập (hay biến **predictor**)
- **y**: Biến phụ thuộc (hay biến **target**)

$$y = b_0 + b_1 x$$

- b_0: là **intercept**
- b_1: là **slope**

Thể hiện độ dốc

Thể hiện điểm chặn đầu của mô hình



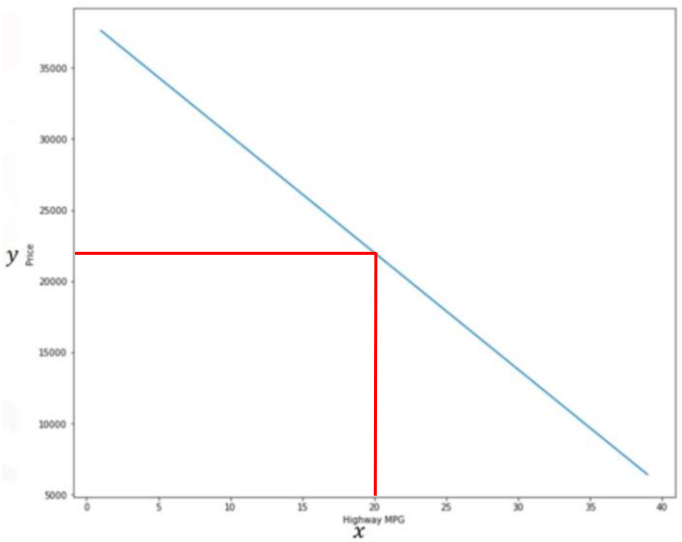
2. Hồi quy tuyến tính đơn biến

Dự đoán - Prediction

- Dùng mô hình hiện tại để dự đoán kết quả.

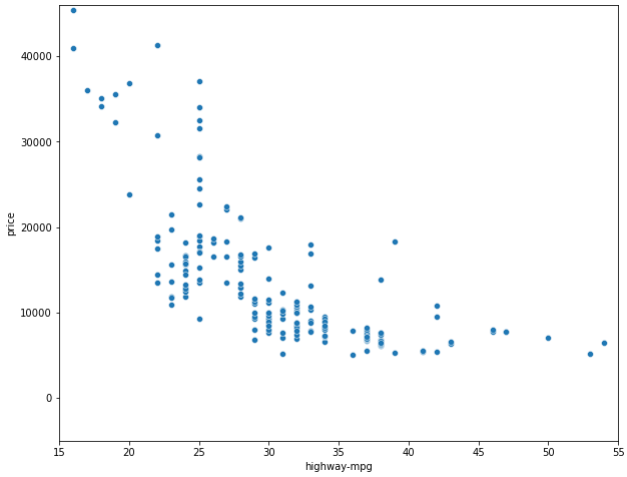
Ví dụ ta có mô hình:

- **y** = 38423 – 821**x**
- y = 38423 – 821(**20**)
- y = 22003



2. Hồi quy tuyến tính đơn biến

Xây dựng mô hình (gọi tắt là fit)



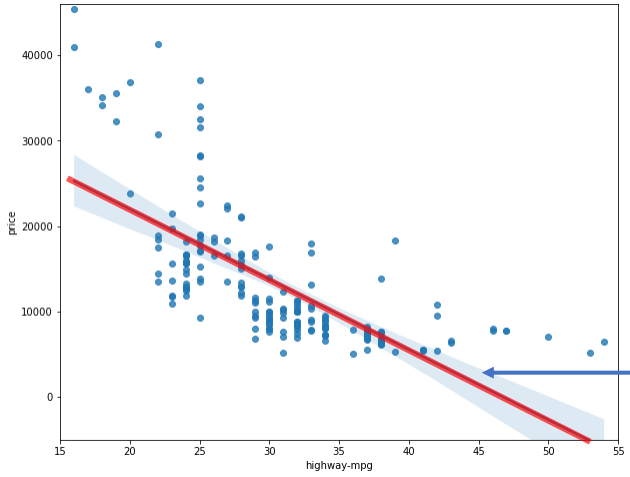
Fit

$y = b_0 + b_1 x$

21

2. Hồi quy tuyến tính đơn biến

Xây dựng mô hình (gọi tắt là fit)



Fit

model

22

2. Hồi quy tuyến tính đơn biến

Xây dựng mô hình hồi quy tuyến tính đơn biến

Chuẩn bị:

- **X**: Biến độc lập - Biến dùng để dự đoán (predictor variable)
- **Y**: Biến phụ thuộc - Biến mục tiêu (target variable)

Cài đặt:

1. import linear_model từ gói scikit-learn

```
from sklearn.linear_model import LinearRegression
```

2. Khởi tạo đối tượng LinearRegression dùng constructor:

```
lm=LinearRegression()
```


2. Hồi quy tuyến tính đơn biến

Xây dựng mô hình hồi quy tuyến tính đơn biến

3. Chọn biến độc lập X và biến phụ thuộc Y

```
X = df[['highway-mpg']]
Y = df['price']
```

4. Gọi phương thức lm.fit(X, Y) để xây dựng mô hình, tìm b_0 và b_1

```
lm.fit(X, Y)
```

5. Dự đoán

```
Yhat=lm.predict(X)
```

X	Yhat
10	13400
…	…
15	10300

2. Hồi quy tuyến tính đơn biến

Xây dựng mô hình hồi quy tuyến tính đơn biến

- Xuất kết quả intercept (b_0):

```
lm.intercept_
```

Kết quả: **38423.305858**

- Xuất kết quả slope (b_1):

```
lm.coef_
```

Kết quả: **-821.73337832**

- Mô hình thể hiện mối quan hệ giữa **Price** và **Highway_MPG**:

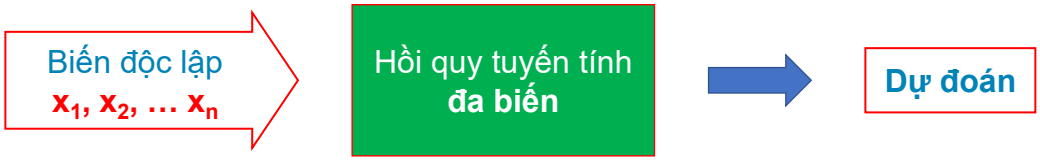
Price = 38423.31 – 821.73 * highway-mpg

$$Y = b_0 + b_1 x$$

3. Hồi quy tuyến tính đa biến

Giới thiệu

- Hồi quy tuyến tính đa biến sẽ dùng nhiều biến độc lập để đưa ra dự đoán.



3. Hồi quy tuyến tính đa biến

Model

Phương pháp này được sử dụng để biểu diễn mối quan hệ giữa:

- Hai hoặc nhiều biến độc lập **X** (hay biến predictor)
- Một biến phụ thuộc **Y** (hay biến target)

$$Y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 + \ldots$$

- b_0 intercept (X = 0)
- b_1 coefficient hoặc parameter của x_1
- b_2 coefficient hoặc parameter of x_2 và tương tự…

3. Hồi quy tuyến tính đa biến

Xây dựng mô hình hồi quy tuyến tính đa biến

1. Chọn 4 biến độc lập

```
Z = df[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']]
```

2. Xây dựng mô hình bằng phương thức fit():

```
lm.fit(Z, df['price'])
```

3. Dự đoán

```
Yhat=lm.predict(X)
```

x_1	x_2	x_3	x_4
3	5	-4	3
:	:	:	:
2	4	2	-4

Yhat
2
:
3

3. Hồi quy tuyến tính đa biến

Xây dựng mô hình hồi quy tuyến tính đa biến

1. Lấy giá trị intercept (b_0)

```
lm.intercept_
-15678.742628061467
```

2. Lấy coefficients (b_1, b_2, b_3, b_4)

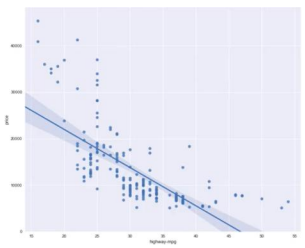
```
lm.coef_
array([52.65851272 ,4.69878948,81.95906216 , 33.58258185])
```

3. Mô hình:

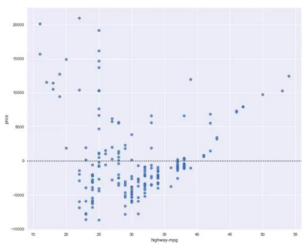
Price = -15678.74 + (52.66) * **horsepower** + (4.70) * **curb-weight** + (81.96) * **engine-size** + (33.58) * **highway-mpg**

4. Đánh giá mô hình dùng trực quan

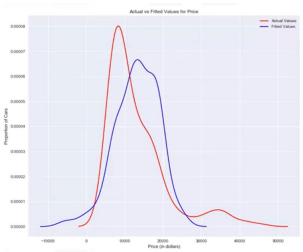
4.1. Regression plot



4.2. Residual plot



4.3. Distribution plot



4.1 Regression plot

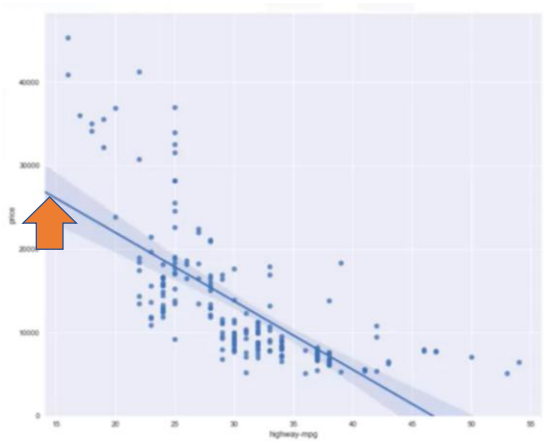
- Tại sao phải dùng đến regression plot?

- Biểu đồ hồi quy là một tiêu chí đánh giá tốt về mô hình. Vì thể hiện:
  1. Mối quan hệ giữa các biến.
  2. Độ mạnh của mối tương quan (correlation).
  3. Chiều hướng của các mối quan hệ đồng thuận và không đồng thuận (positive and negative)

4.1 Regression plot

Regression plot cho chúng ta thấy sự kết hợp của:

- Sự tán xạ của các điểm dữ liệu sẽ đại diện cho một ý nghĩa (target) khác nhau.
- Đường hồi quy tuyến tính được xây dựng (fit) dựa trên các điểm dữ liệu.

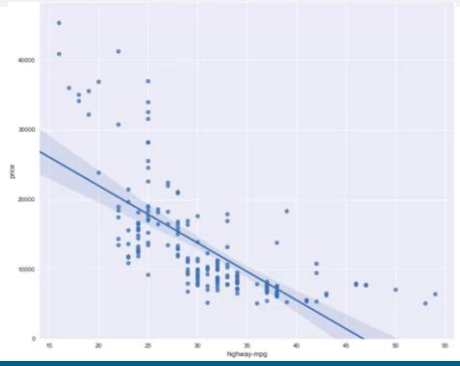


4.1. Regression plot

Cài đặt Regression plot

```
import seaborn as sns
```

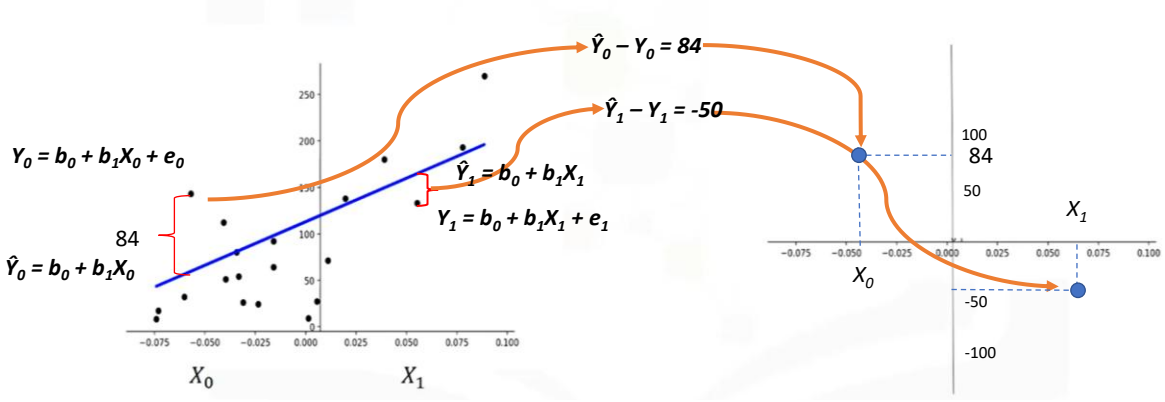
```
sns.regplot(x="highway-mpg", y="price", data=df)
```



4.2. Residual Plot

Residual Plot (Phần dư, hoặc errors)

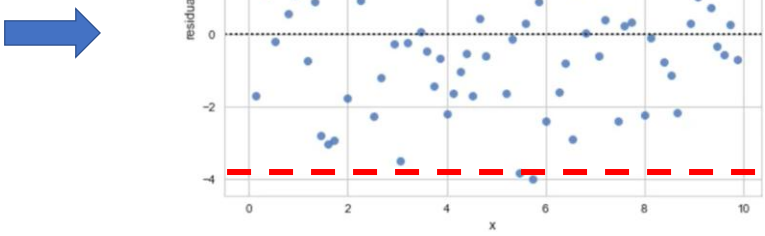
Y-axis: residual



X-axis: the predictor variable or fitted values

4.2. Residual Plot

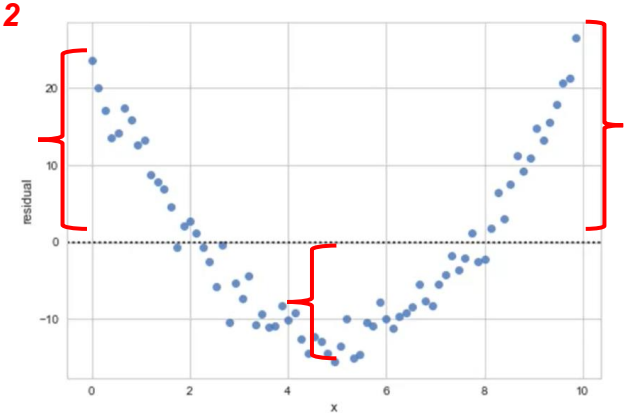
Trường hợp 1



- Quan sát các điểm trải đều trong **residual**:
  - (Ngẫu nhiên) trải ra xung quanh trục thì mô hình tuyến tính là phù hợp

4.2. Residual Plot

Trường hợp 2

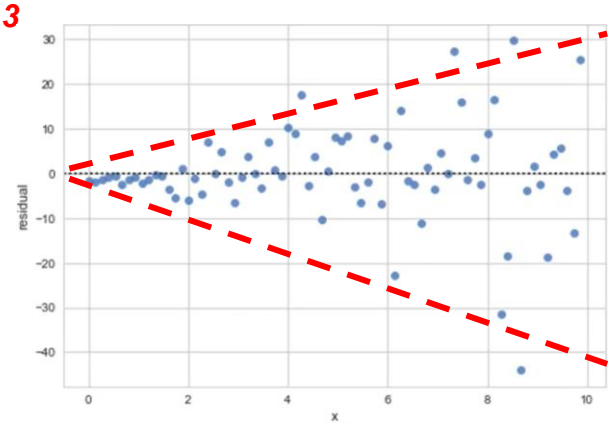


- **Không ngẫu nhiên trải ra xung quanh trục x**
- **Tìm mô hình khác phù hợp hơn**

Phi tuyến tính

4.2. Residual plot

Trường hợp 3

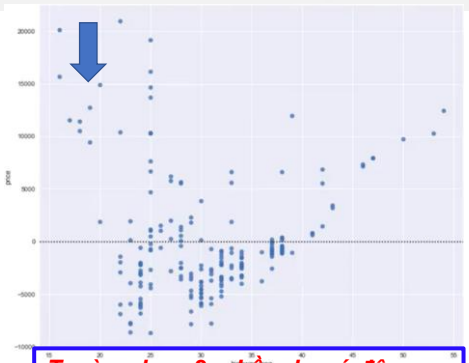


- **Không tập trung quanh trục, phần dư tăng theo x**
- **Mô hình không chính xác**

4.2. Residual plot

Cài đặt Residual plot

```
import seaborn as sns
sns.residplot(df['highway-mpg'], df['price'])
```



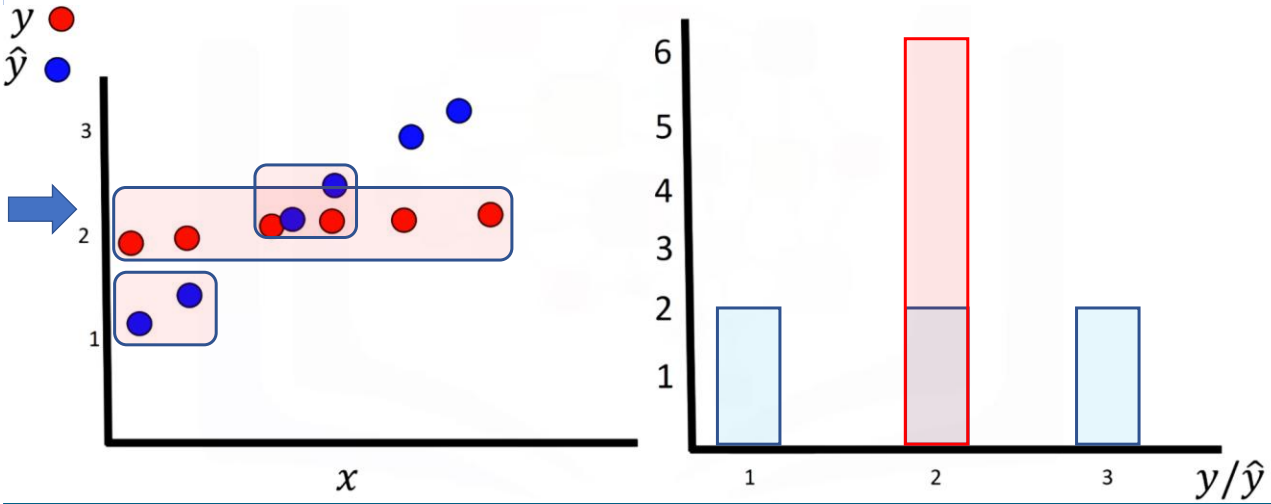
Trường hợp 2: phần dư có độ cong

Chủ đề chọn báo cáo 15 phút

- Thiết kế Residual plot dành cho đa biến.

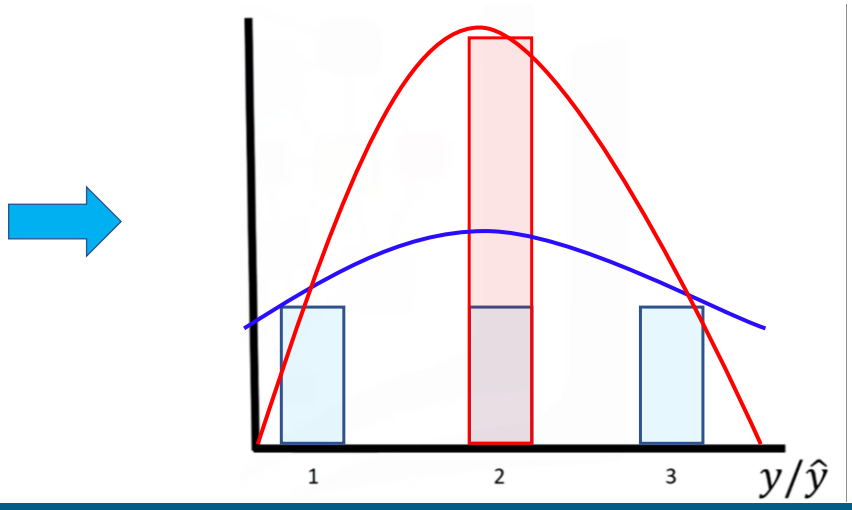
4.3. Distribution plot

Distribution Plot



4.3. Distribution plot

Distribution Plots

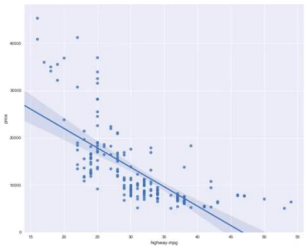


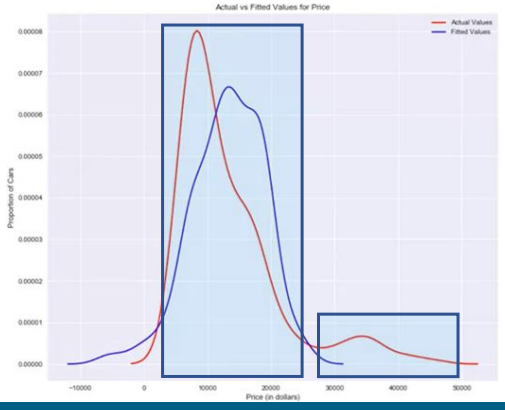
46

4.3. Distribution plot

Ví dụ: So sánh đường phân phối chuẩn giữa

- Giá trị đã dự đoán từ kết quả mô hình (Đường màu xanh)
- Giá trị thực tế (Đường màu đỏ)

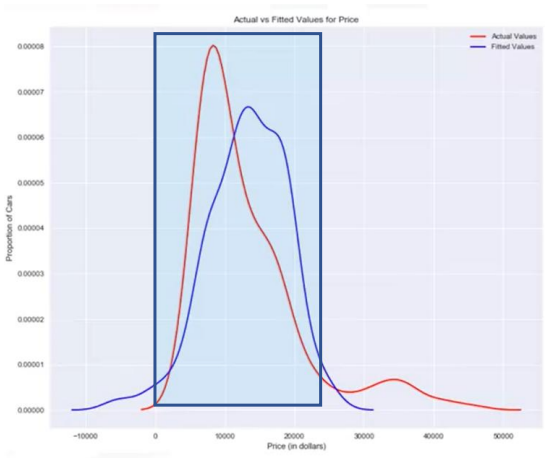




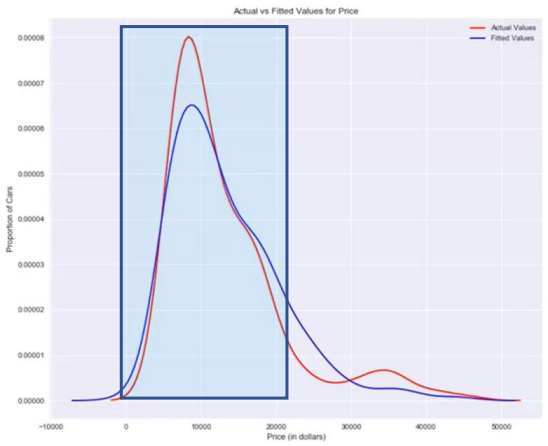
47

4.3. Distribution plot

Ví dụ



A



B

48

4.3. Distribution plot

Cài đặt Distribution plot

```
import seaborn as sns
```

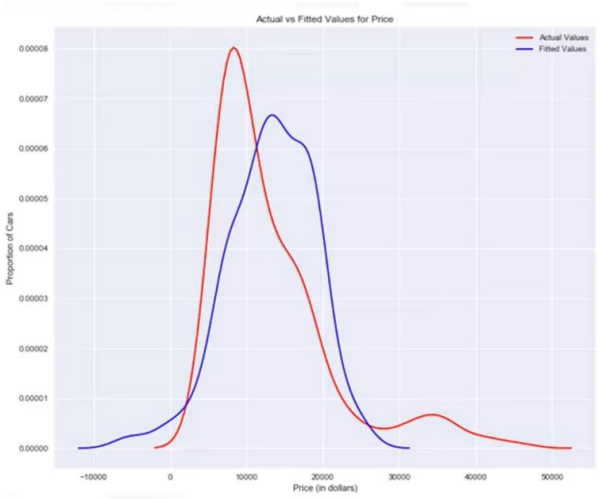
```
ax1 = sns.distplot(df['price'], hist=False, color="r", label="Actual Value")

sns.distplot(Yhat, hist=False, color="b", label="Fitted Values" , ax=ax1)
```


49

4.3. Distribution plot

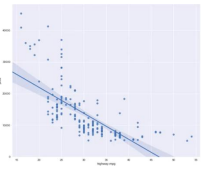
Kết quả



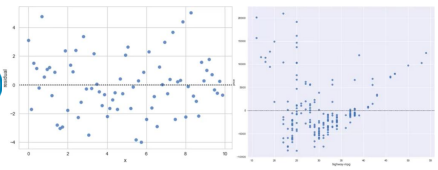
4. Đánh giá mô hình dùng trực quan

Tóm lại

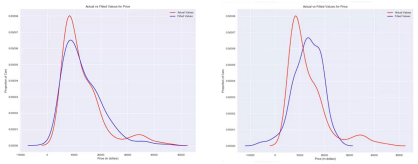
- Regression Plot => regplot(x="…", y="...", data=…)



- Residual Plot => residplot(df.a, df.b)



- Distribution Plot => distplot()



Bài tập

Vận dụng hồi quy tuyến tính đa biến cho các biến độc lập sau:

- curb-weight, engine-size, length, width, horsepower, city-mpg, highway-mpg, wheel-base, bore, drive-wheels

Sau đó, đánh giá bằng trực quan.

5. Hồi quy đa thức

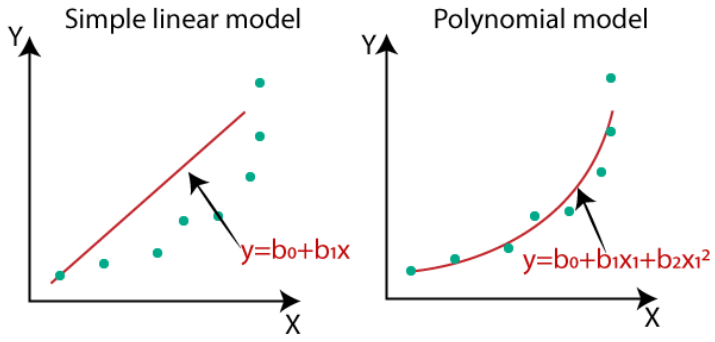
Hồi quy đa thức - Polynomial regression

- Là một trường hợp đặc biệt của mô hình hồi quy tuyến tính tổng quát.
- Biểu diễn các mối quan hệ **cong phi tuyến (curvilinear)** rất tốt.

ˌkərvəˈlinēər

Quan hệ cong phi tuyến

- Bình phương
- Hoặc thiết lập các bậc cao hơn cho biến độc lập



5. Hồi quy đa thức

Cong phi tuyến

- Bậc 2 (Quadratic – 2nd order)

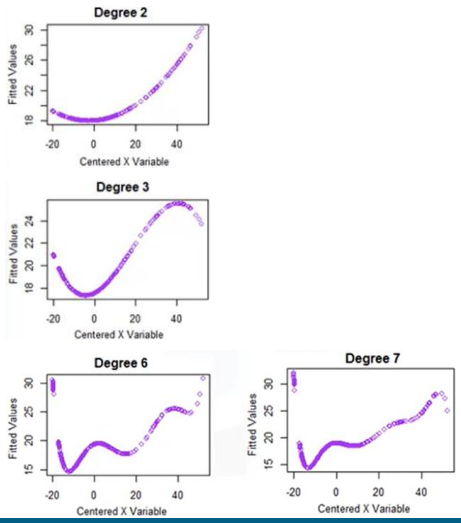
$$\hat{Y} = b_0 + b_1 x_1 + b_2 (x_1)^2$$

- Bậc 3 (Cubic – 3rd order)

$$\hat{Y} = b_0 + b_1 x_1 + b_2 (x_1)^2 + b_3 (x_1)^3$$

- Bậc cao hơn

$$\hat{Y} = b_0 + b_1 x_1 + b_2 (x_1)^2 + b_3 (x_1)^3 + ..$$



54

5. Hồi quy đa thức

Xây dựng mô hình (đơn biến)

x = df['highway-mpg']
y = df['price']

1. Xây dựng mô hình bậc 3

```
f=np.polyfit(x,y,3)
p=np.polydl(f)
```

2. Xuất mô hình

```
print (p)
```

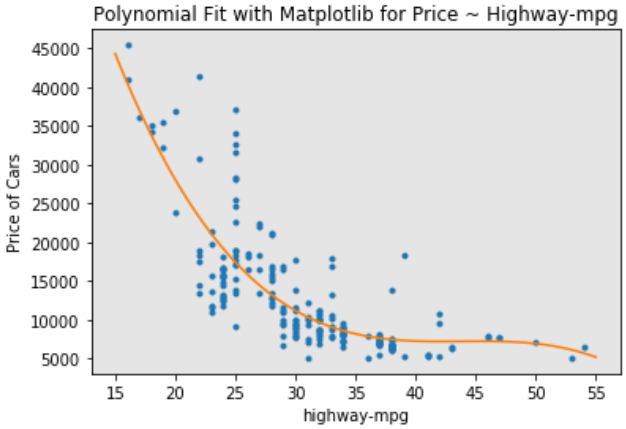
$$-1.557(x_1)^3 + 204.8(x_1)^2 + 8965\, x_1 + 1.37 \times 10^5$$

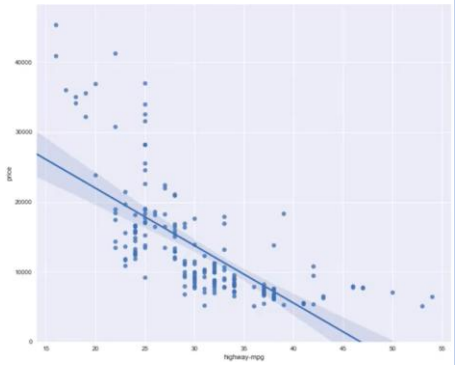
55

5. Hồi quy đa thức

Mô hình

- Kết quả: $-1.557(x_1)^3 + 204.8(x_1)^2 + 8965\,x_1$+1.37x$10^5$





5. Hồi quy đa thức

Đa biến và bậc cao hơn

- Mô hình

$$\mathbf{yHat} = b_0 + b_1\mathbf{X_1} + b_2\mathbf{X_2} + b_3\mathbf{X_1X_2} + b_4(\mathbf{X_1})^2 + b_5(\mathbf{X_2})^2 + \ldots..$$

$$yHat = b_0 + b_1\mathbf{a} + b_2\mathbf{b} + b_3\mathbf{c} + b_4\mathbf{d} + b_5\mathbf{e} + \ldots..$$

- Các bước:
  - Bước #1: Biến đổi các biến (Transform=>feature)
  - Bước #2: Dùng LinearRegression để Fit - xây dựng

5. Hồi quy đa thức

Bước #1: Biến đổi biến

- Sử dụng mô-đun **PolynomialFeatures** trong package **preprocessing** của **sklearn**

```
from sklearn.preprocessing import PolynomialFeatures
```

```
pr=PolynomialFeatures(degree=2)
```

```
pr.fit_transform([[1, 2]])
```

```
array([[1., 1., 2., 1., 2., 4.]])
```

Chuyển [a, b] thành bậc 2
Kết quả: [1, a, b, a^2, ab, b^2]

5. Hồi quy đa thức

Bước #1: Biến đổi biến

- Sử dụng mô-đun **PolynomialFeatures** trong package **preprocessing** của **sklearn**

```
from sklearn.preprocessing import PolynomialFeatures
```

```
pr=PolynomialFeatures(degree=2, include_bias=False)
```

```
pr.fit_transform([[1, 2]])
```

```
array([[1., 2., 1., 2., 4.]])
```

Chuyển [a, b] thành bậc 2
Kết quả: [a, b, a^2, ab, b^2]

5. Hồi quy đa thức

Bước #1: Biến đổi biến

- Áp dụng vào bộ dữ liệu hiện tại

```
from sklearn.preprocessing import PolynomialFeatures
```

```
pr = PolynomialFeatures(degree=2, include_bias=False)
```

```
X_poly = pr.fit_transform(df[['horsepower', 'curb-weight']])
```

```
X_poly.shape
```

```
(201, 5)
```


5. Hồi quy đa thức

Bước #2: Dùng LinearRegression để Fit

- Áp dụng vào bộ dữ liệu

```
from sklearn.linear_model import LinearRegression
```

```
lm = LinearRegression()
```

```
lm.fit(X_poly, df['price'])
```

```
LinearRegression()
```

```
lm.intercept_
```

```
17062.609692007238
```

```
lm.coef_
```

```
array([ 1.07696940e+02, -1.80653670e+01,  5.48648597e-01, -5.38543914e-02,
        5.82043770e-03])
```

5. Hồi quy đa thức

Khuyến nghị

- Mô hình:

$$yHat = b_0 + b_1X_1 + b_2X_2 + b_3X_1X_2 + b_4(X_1)^2 + b_5(X_2)^2 + \ldots..$$

$$yHat = b_0 + b_1a + b_2b + b_3c + b_4d + b_5e + \ldots..$$

- Khuyến nghị xây dựng theo bước sau:
  - Bước #1: Chuẩn hóa (Normalize)
  - Bước #2: Biến đổi các biến (Transform=>feature)
  - Bước #3: Dùng LinearRegression để Fit - xây dựng

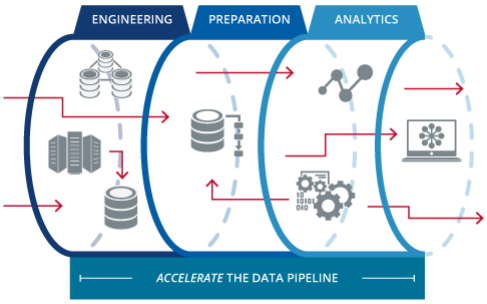
5. Hồi quy đa thức

Chuẩn hóa

- Nên chuẩn hóa các biến khi áp dụng nhiều chiều.
- Ví dụ, thực hiện **chuẩn hóa** cho **biến** horsepower và highway-mpg

```
from sklearn.preprocessing import StandardScaler
SCALE=StandardScaler()
SCALE.fit(x_data[['horsepower', 'highway-mpg']])
```

Pipeline



65

Pipeline

- Là một thư viện trong scikit-learn.
- Tập hợp các phép biến đổi của các bước để tìm ra mô hình.
- Mục tiêu của kỹ thuật pipeline là thể hiện các quá trình phân tích thành một qui trình tự động.

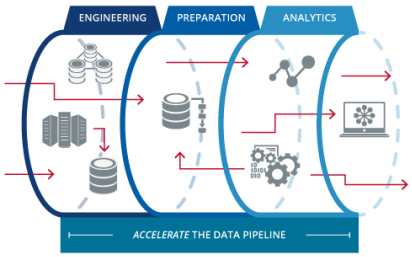
sklearn.pipeline.Pipeline

class `sklearn.pipeline.Pipeline`(*steps*, *memory=None*, *verbose=False*) [source]

Pipeline of transforms with a final estimator.

Sequentially apply a list of transforms and a final estimator. Intermediate steps of the pipeline must be 'transforms', that is, they must implement fit and transform methods. The final estimator only needs to implement fit. The transformers in the pipeline can be cached using `memory` argument.

The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters. For this, it enables setting parameters of the various steps using their names and the parameter name separated by a '__', as in the example below. A step's estimator may be replaced entirely by setting the parameter with its name to another estimator, or a transformer removed by setting it to 'passthrough' or `None`.

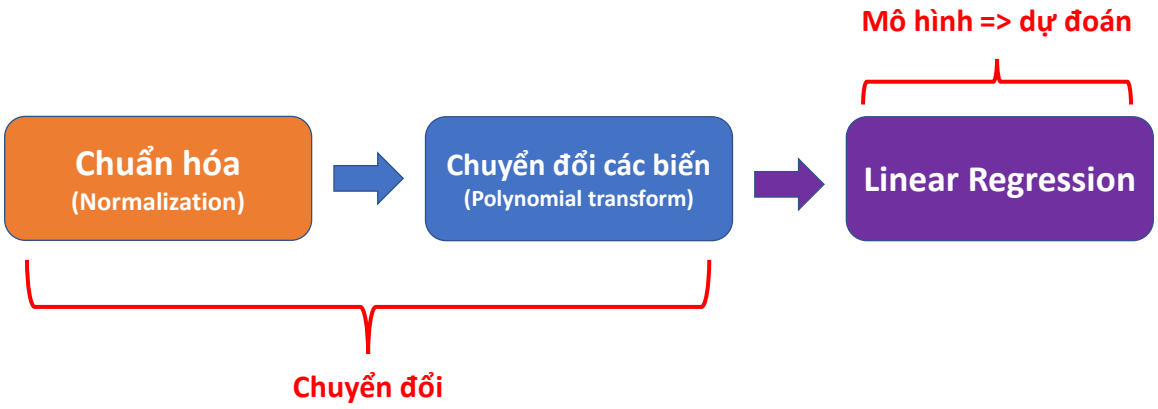


66

6. Kỹ thuật Pipeline

Pipeline

- Để có được một mô hình dùng để dự đoán thì phải trải qua nhiều bước:



6. Kỹ thuật Pipeline

Cài đặt kỹ thuật Pipeline

- Nhập thư viện hỗ trợ có liên quan

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
```

6. Kỹ thuật Pipeline

Pipeline Constructor

```
Input=[('scale',StandardScaler()),
       ('polynomial', PolynomialFeatures(include_bias=False)),
       ('model',LinearRegression())]
```

- **Dùng Pipeline constructor để khởi tạo đối tượng**

```
pipe=Pipeline(Input)
```

pipeline object

6. Kỹ thuật Pipeline

Pipeline Constructor

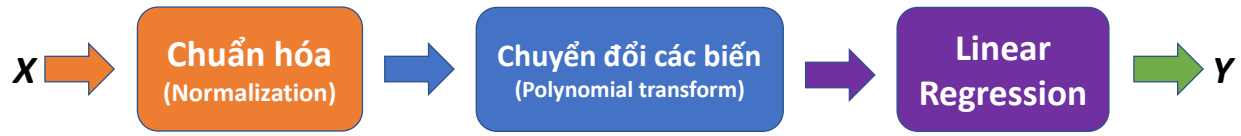
- Dùng phương thức fit() để xây dựng mô hình từ đối tượng pipeline

```
pipe.fit(Z,y)
```

```
Pipeline(steps=[('scale', StandardScaler()),
                ('polynomial', PolynomialFeatures(include_bias=False)),
                ('model', LinearRegression())])
```

```
ypipe=pipe.predict(Z)
ypipe[0:4]
```

```
array([13102.74784201, 13102.74784201, 18225.54572197, 10390.29636555])
```



```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
```

```
Input=[('scale',StandardScaler()),
       ('polynomial', PolynomialFeatures(include_bias=False)),
       ('model',LinearRegression())]
```

```
pipe=Pipeline(Input)
pipe
```

```
Pipeline(steps=[('scale', StandardScaler()),
                ('polynomial', PolynomialFeatures(include_bias=False)),
                ('model', LinearRegression())])
```

```
pipe.fit(Z,y)
```

```
Pipeline(steps=[('scale', StandardScaler()),
                ('polynomial', PolynomialFeatures(include_bias=False)),
                ('model', LinearRegression())])
```

```
ypipe=pipe.predict(Z)
ypipe[0:4]
```

```
array([13102.74784201, 13102.74784201, 18225.54572197, 10390.29636555])
```


71

Áp dụng

- ['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']
- Áp dụng pipeline tìm mô hình hồi quy đa thức bậc 3 cho 4 biến trên.

Code it!

72

7. Thang đo đánh giá

Giới thiệu các thang đo

- Thang đo là 1 giá trị dùng để định lượng mức độ phù hợp của mô hình với bộ dữ liệu.

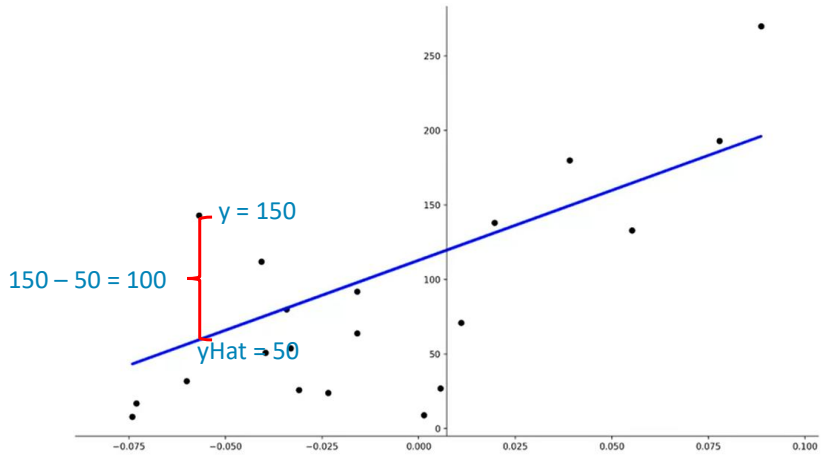
- Hai thang đo quan trọng để xác định mức độ phù hợp của mô hình:
  - **7.1. Mean Squared Error (MSE)**
  - **7.2. R-squared (R^2)**

Có thể gọi là:
Sai số bình phương trung bình
Sai số toàn phương trung bình

Hệ số xác định

7.1. Mean Squared Error (MSE)

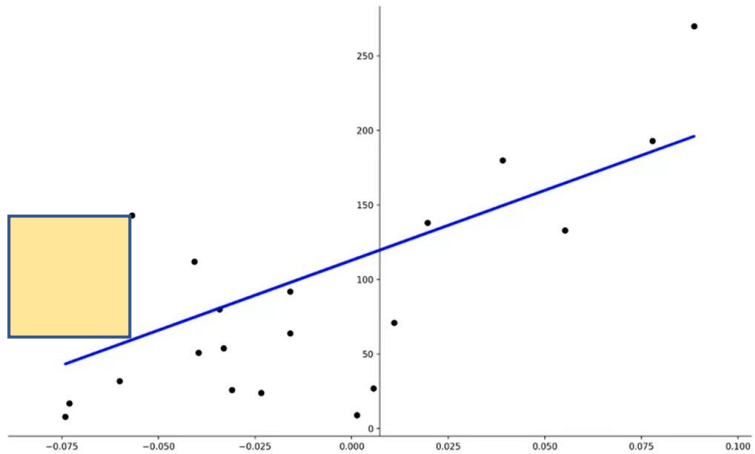
Mean Squared Error (MSE)



7.1. Mean Squared Error (MSE)

Mean Squared Error (MSE)

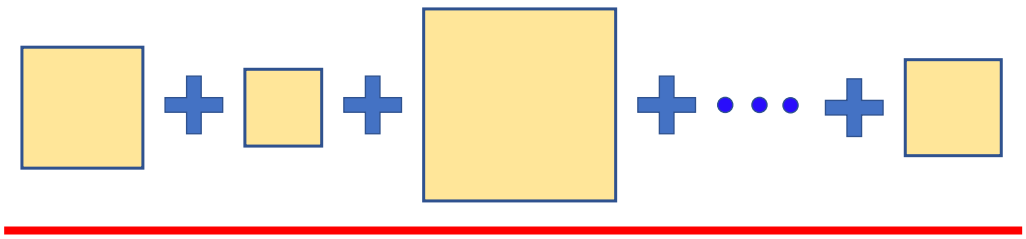
$(100)^2$



75

7.1. Mean Squared Error (MSE)

Mean Squared Error (MSE)



Số lượng mẫu

76

7.1. Mean Squared Error (MSE)

Mean Squared Error (MSE)

- Sử dụng mô-đun metrics trong sklearn để tính thang đo MSE như sau:

```
from  sklearn.metrics import mean_squared_error

mean_squared_error(df['price'],Y_predict_simple_fit)


3163502.944639888
```

- Thử tính bằng cách thủ công?

77

7.2 R-squared (R^2)

R-squared (R^2)

- Hệ số xác định (coeffient of determination) <=> R-squared (R^2)
- Là một thang đo để xác định độ phù hợp của dữ liệu so với đường hồi quy đã được fit.
- R^2: thể hiện tỉ lệ phần trăm thay đổi (sự biến thiên) của biến target (Y), được giải thích bởi mô hình tuyến tính.

78

7.2 R-squared (R^2)

Công thức tính R-squared (R^2)

$$R^2 = 1 - \frac{\text{MSE của đường hồi quy}}{\text{MSE của đường trung bình}}$$

7.1. R-squared (R^2)

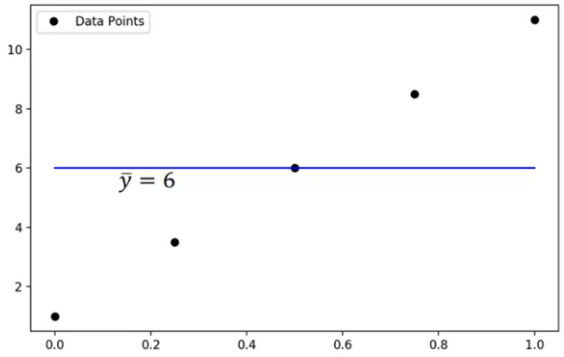
MSE của đường trung bình

X là biến độc lập

Y là biến phụ thuộc

- $\bar{y}$ là trung bình cộng của biến **Y**, tức giá trị trung bình của mỗi điểm dữ liệu
- Ví dụ hình bên cạnh $\bar{y} = 6$

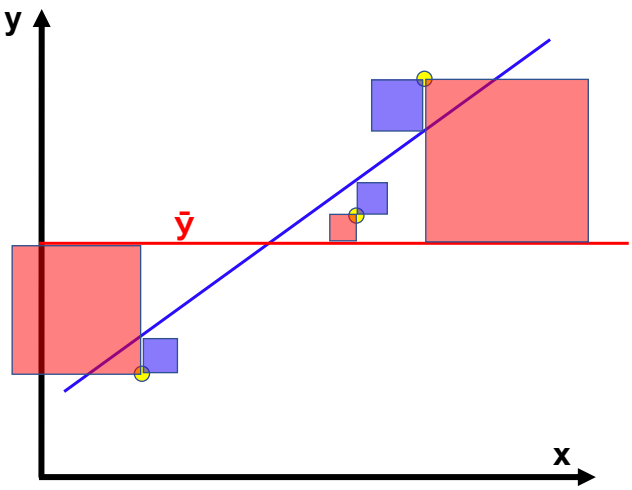
Tính MSE của $\bar{y}$



7.1. R-squared (R^2)

$\hat{Y}$

Phương pháp



- Đường màu xanh thể hiện đường hồi quy.
- Các hình vuông màu xanh thể hiện MSE của đường hồi quy.
- Đường màu đỏ thể hiện giá trị trung bình của các điểm dữ liệu (còn gọi là đường trung bình $\bar{y}$)
- Các hình vuông màu đỏ thể hiện MSE của đường màu đỏ

=> Diện tích của hình vuông màu xanh nhỏ hơn nhiều so với diện tích của hình vuông màu đỏ.

7.1. R-squared (R^2)

Phương pháp

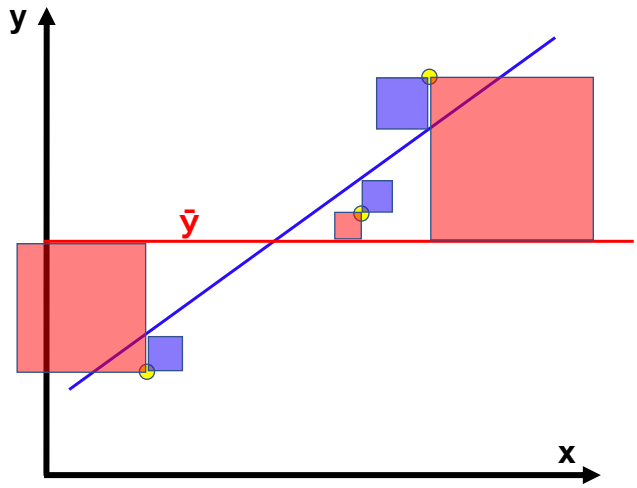
- Nếu đường hồi quy là 1 đường tốt thì tỉ lệ các diện tích của MSE tiến về 0.

$$\frac{\text{MSE của đường hồi quy}}{\text{MSE của đường trung bình } (\bar{y})} = \frac{\square + \square + \square}{\square + \square + \square}$$

= tiến về 0

7.1. R-squared (R^2)

Phương pháp



$$R^2 = 1 - \frac{\text{MSE của đường hồi quy}}{\text{MSE của đường trung bình } (\bar{y})}$$

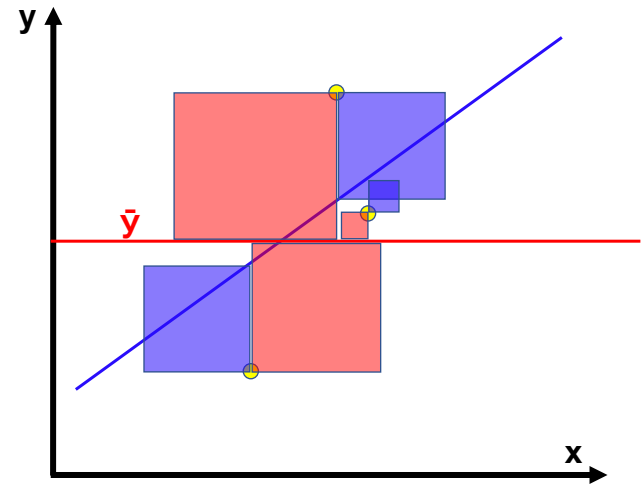
= 1 - tiến về 0

= tiến về 1

Tóm lại, Nếu R^2 tiến về 1 thì regression line là tốt

7.1. R-squared (R^2)

Phương pháp



$$R^2 = 1 - \frac{\text{MSE của đường hồi quy}}{\text{MSE của đường trung bình } (\bar{y})}$$

= 1 – (tiến về 1)

= tiến về 0

Tóm lại, Nếu R^2 tiến về 0 thì regression line là **không** TỐT

7.1. R-squared (R^2)

Tính R^2

- Giá trị MSE thuộc đoạn [0; 1]
- Gọi phương thức score() trong đối tượng mô hình **lm** để tính R^2

```
X = df[['highway-mpg']]
Y = df['price']

lm.fit(X, Y)
```

```
lm.score(X,y)
0.496591188
```

=> Tính **R^2** bằng phương thức **score**

8. Đánh giá

Để xác định một kết quả fit tốt nhất (final best fit), chúng ta nên xem xét các tiêu chí sau:

1. Các giá trị dự đoán có hợp lý không?
2. Sử dụng trực quan.
3. Thang đo định lượng R^2 *(tiến về 1)* và MSE
4. So sánh các mô hình lẫn nhau.

8. Đánh giá

Đánh giá kết quả dự đoán

- Xây dựng mô hình

```
lm.fit(df['highway-mpg'],df['prices']))
```

- Kết quả dự đoán cho biến highway-mpg có giá trị là 30

```
lm.predict(30)
```

- Kết quả: $13771.30

```
lm.coef_    -821.73337832
```

Price = 38423.31 - 821.73 * highway-mpg

8. Đánh giá

Đánh giá kết quả dự đoán

- Sử dụng import numpy

```
import numpy as np
```

- Dùng phương thức arrange() để tạo một mảng giá trị từ 1 đến 100

```
new_input=np.arange(1,101,1).reshape(-1,1)
```

1	2	...	99	100

8. Đánh giá

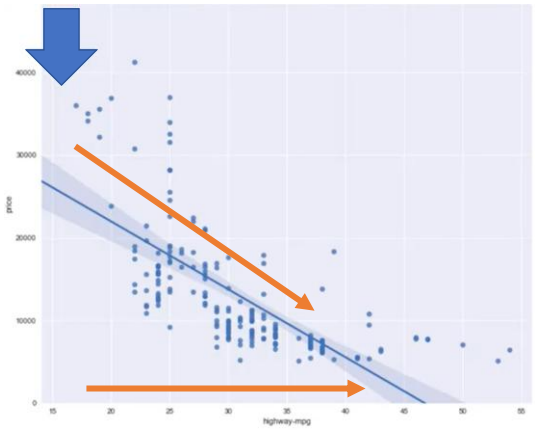
Đánh giá kết quả dự đoán

```
yhat=lm.predict(new_input)
```

```
array([ 37601.57247984,  36779.83910151,  35958.10572319,  35136.37234487,
        34314.63896655,  33492.90558823,  32671.1722099 ,  31849.43883158,
        31027.70545326,  30205.97207494,  29384.23869662,  28562.50531829,
        27740.77193997,  26919.03856165,  26097.30518333,  25275.57180501,
        24453.83842668,  23632.10504836,  22810.37167004,  21988.63829172,
        21166.9049134 ,  20345.17153508,  19523.43815675,  18701.70477843,
        17879.97140011,  17058.23802179,  16236.50464347,  15414.77126514,
        14593.03788682,  13771.3045085 ,  12949.57113018,  12127.83775186,
        11306.10437353,  10484.37099521,   9662.63761689,   8840.90423857,
         8019.17086025,   7197.43748192,   6375.7041036 ,   5553.97072528,
         4732.23734696,   3910.50396864,   3088.77059031,   2267.03721199,
         1445.30383367,    623.57045535,   -198.16292297,  -1019.8963013 ,
        -1841.62967962,  -2663.36305794,  -3485.09643626,  -4306.82981458,
        -5128.5631929 ,  -5950.29657123,  -6772.02994955,  -7593.76332787,
        -8415.49670619,  -9237.23008451, -10058.96346284, -10880.69684116,
       -11702.43021948, -12524.1635978 , -13345.89697612, -14167.63035445,
       -14989.36373277, -15811.09711109, -16632.83048941, -17454.56386773,
       -18276.29724606, -19098.03062438, -19919.7640027 , -20741.49738102,
       -21563.23075934, -22384.96413767, -23206.69751599, -24028.43089431,
       -24850.16427263, -25671.89765095, -26493.63102927, -27315.3644076 ,
       -28137.09778592, -28958.83116424, -29780.56454256, -30602.29792088,
       -31424.03129921, -32245.76467753, -33067.49805585, -33889.23143417,
       -34710.96481249, -35532.69819082, -36354.43156914, -37176.16494746,
       -37997.89832578, -38819.6317041 , -39641.36508243, -40463.09846075,
       -41284.83183907, -42106.56521739, -42928.29859571])
```

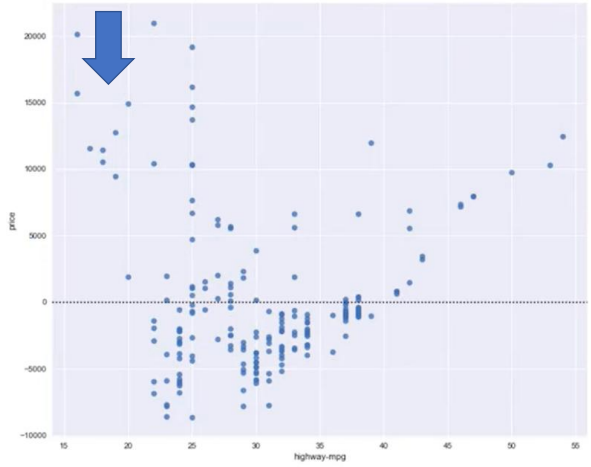

8. Đánh giá

Dùng trực quan để đánh giá – Regression plot



8. Đánh giá

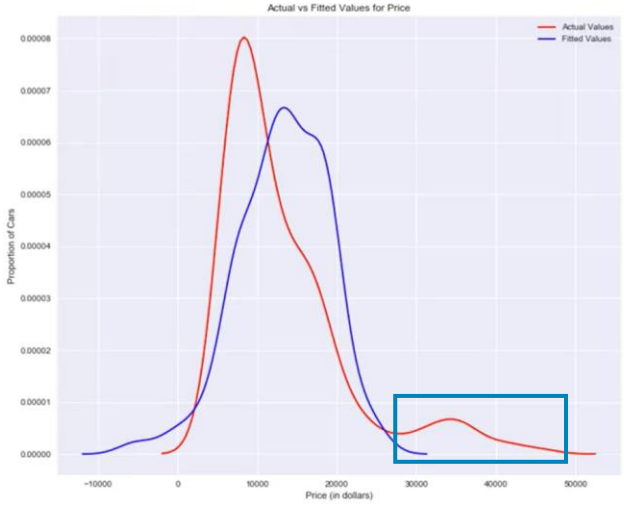
Dùng trực quan để đánh giá – Residual plot



94

8. Đánh giá

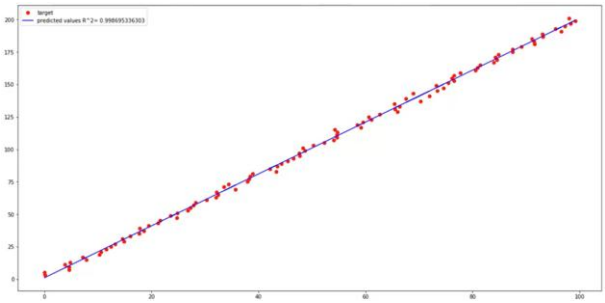
Dùng trực quan để đánh giá – Distribution plot



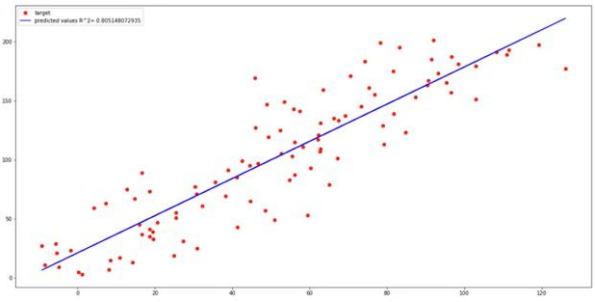
95

8. Đánh giá

R^2



A



B

8. Đánh giá

Bài tập: So sánh giữa hồi quy tuyến tính đa biến và đơn biến

1. Có phải MSE thấp hơn thì luôn luôn có một đường fit tốt hơn?
   - Không nhất thiết. Tại sao?
2. MSE của mô hình MLR sẽ luôn thấp hơn MSE của mô hình SLR, vì các sai số của dữ liệu sẽ giảm khi nhiều biến có trong mô hình. Giải thích?
3. Hồi quy đa thức cũng sẽ có MSE thấp hơn MLR và SLR. Giải thích?
4. R^2

Chương sau sẽ giúp chúng ta tìm ra hướng tốt hơn để đánh giá mô hình.

Áp dụng

98

100

Hỏi - Đáp

101