

# CHƯƠNG 3. HỌC MÁY CƠ BẢN

3.1. Giới thiệu học máy

3.2. Mô hình hồi quy tuyến tính

3.3. Quá khớp (overfitting)

3.4. Chuẩn hoá

3.5. Học cây quyết định

3.6. Học nhóm (bagging và boosting)

3.7. Máy véc-tơ hỗ trợ

3.8. K-láng giềng gần

3.9. Gom cụm dữ liệu

3.10. Giảm chiều dữ liệu

3.11. Ứng dụng các mô hình học máy trong xử lý dữ liệu kinh doanh



TRƯỜNG ĐẠI HỌC NGÂN HÀNG  
THÀNH PHỐ HỒ CHÍ MINH



TRƯỜNG ĐẠI HỌC NGÂN HÀNG  
THÀNH PHỐ HỒ CHÍ MINH

### 3.1. Giới thiệu học máy

Học máy (Machine Learning) là một lĩnh vực con của trí tuệ nhân tạo (AI), tập trung vào việc phát triển các mô hình và thuật toán cho phép máy tính tự học từ dữ liệu mà không được lập trình rõ ràng.

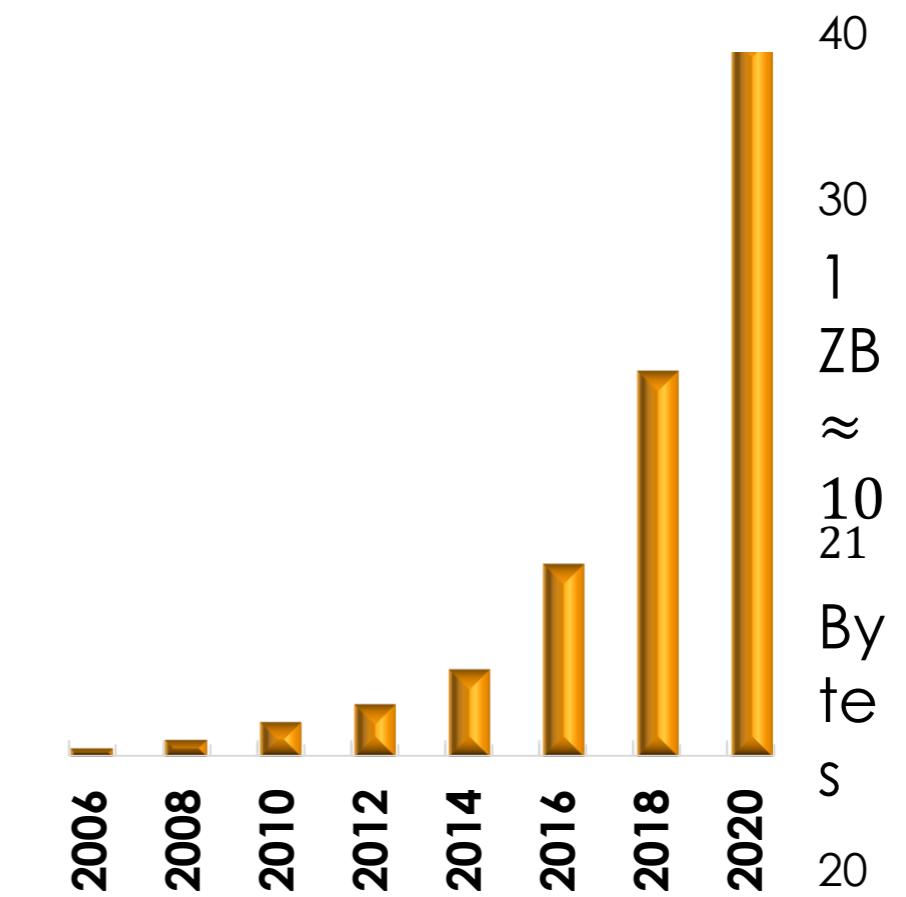
Học máy được sử dụng để tự động hóa tác vụ, phân tích dữ liệu lớn và đưa ra dự đoán chính xác hơn, từ đó hỗ trợ việc ra quyết định trong kinh doanh, khoa học, y tế và nhiều lĩnh vực khác.

- Khai thác dữ liệu, suy luận, dự đoán
- Học máy & Khai thác dữ liệu cung cấp một cách hiệu quả để tạo ra các hệ thống / dịch vụ thông minh.
- Học máy cung cấp các phương pháp quan trọng và nền tảng cho Dữ liệu lớn.

## All global data in Zettabytes



**Each day:**  
230M tweets,  
2.7B comments to FB,  
86400 hours of video  
to YouTube



# Thành tựu



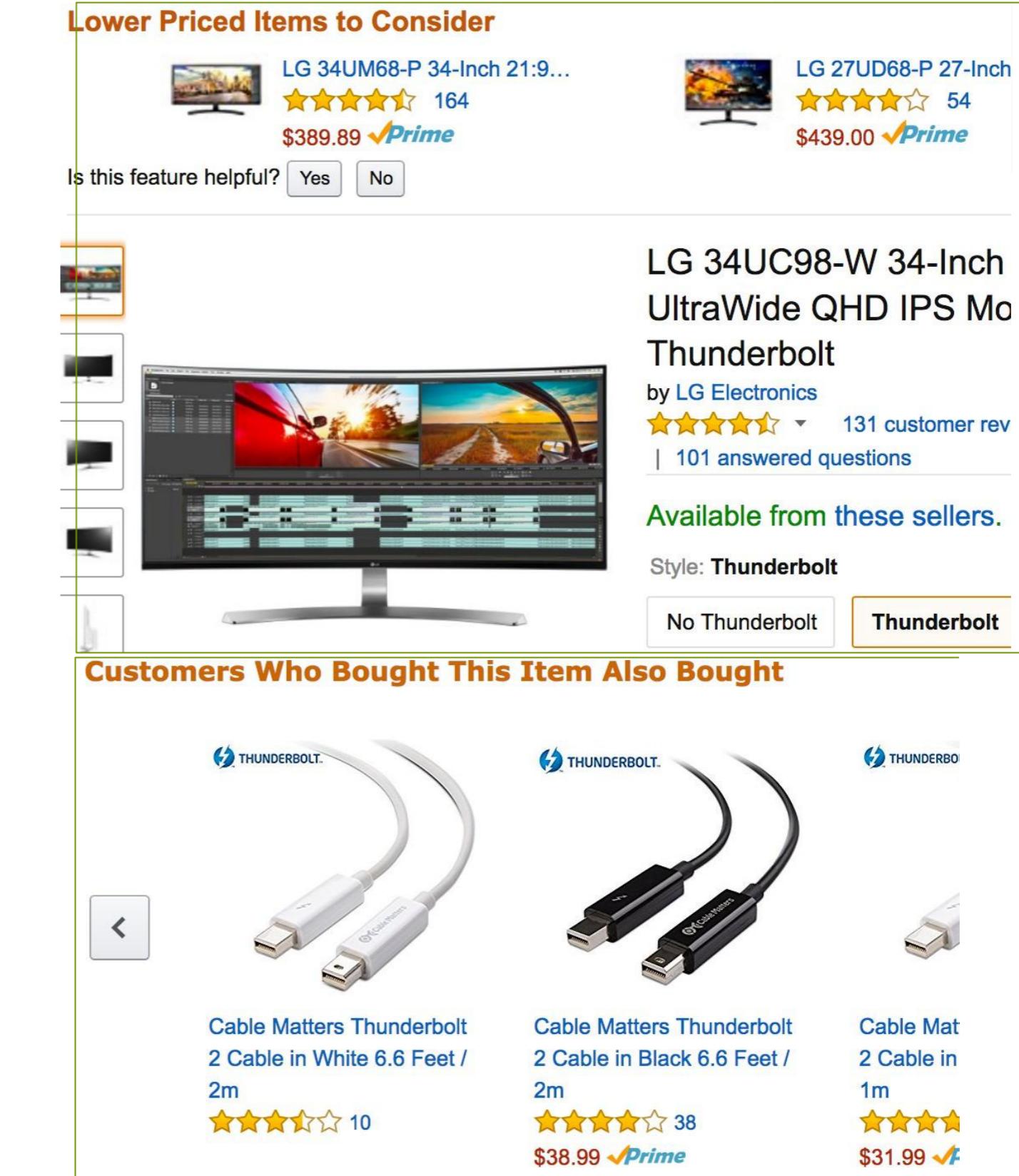
IBM's Watson Supercomputer Destroys Humans in Jeopardy (2011)

# Một số thành công: Bí mật của Amazon



“Công ty báo cáo doanh số **tăng 29%** lên 12,83 tỷ đô la trong quý tài chính thứ hai, tăng so với mức 9,9 tỷ đô la cùng kỳ năm ngoái.”

– Fortune, ngày 30 tháng 7 năm 2012



The screenshot shows a product page for an LG 34UC98-W monitor. At the top, there's a section titled "Lower Priced Items to Consider" showing two alternatives: an LG 34UM68-P monitor for \$389.89 and an LG 27UD68-P monitor for \$439.00, both with Prime delivery. Below this is a poll asking if a feature is helpful, with "Yes" and "No" buttons. The main product image shows a wide monitor displaying a video editing interface. To the left is a vertical sidebar with thumbnail images of other monitors. On the right, there's detailed product information: "LG 34UC98-W 34-Inch UltraWide QHD IPS Monitor Thunderbolt by LG Electronics", with a 4.5-star rating, 131 reviews, and 101 answered questions. It's available from multiple sellers. A "Style: Thunderbolt" filter is applied, with "No Thunderbolt" and "Thunderbolt" buttons. Below the main product, there's a section titled "Customers Who Bought This Item Also Bought" featuring three Thunderbolt cables: "Cable Matters Thunderbolt 2 Cable in White 6.6 Feet / 2m" (4.5 stars, 10 reviews), "Cable Matters Thunderbolt 2 Cable in Black 6.6 Feet / 2m" (4.5 stars, 38 reviews), and "Cable Mat 2 Cable in 1m" (4.5 stars, 1 review).

# Một số thành công: AlphaGo (2016)



TRƯỜNG ĐẠI HỌC NGÂN HÀNG  
THÀNH PHỐ HỒ CHÍ MINH

- AlphaGo of Google DeepMind Nhà vô địch thế giới tại trò chơi cờ vây (Go), 3/2016

- Go là một trò chơi 2500 năm tuổi.
- Go là một trong những trò chơi phức tạp nhất.

- AlphaGo Học hỏi từ 30 triệu chiêu thức của con người và tự chơi để tìm ra các chiêu thức mới.

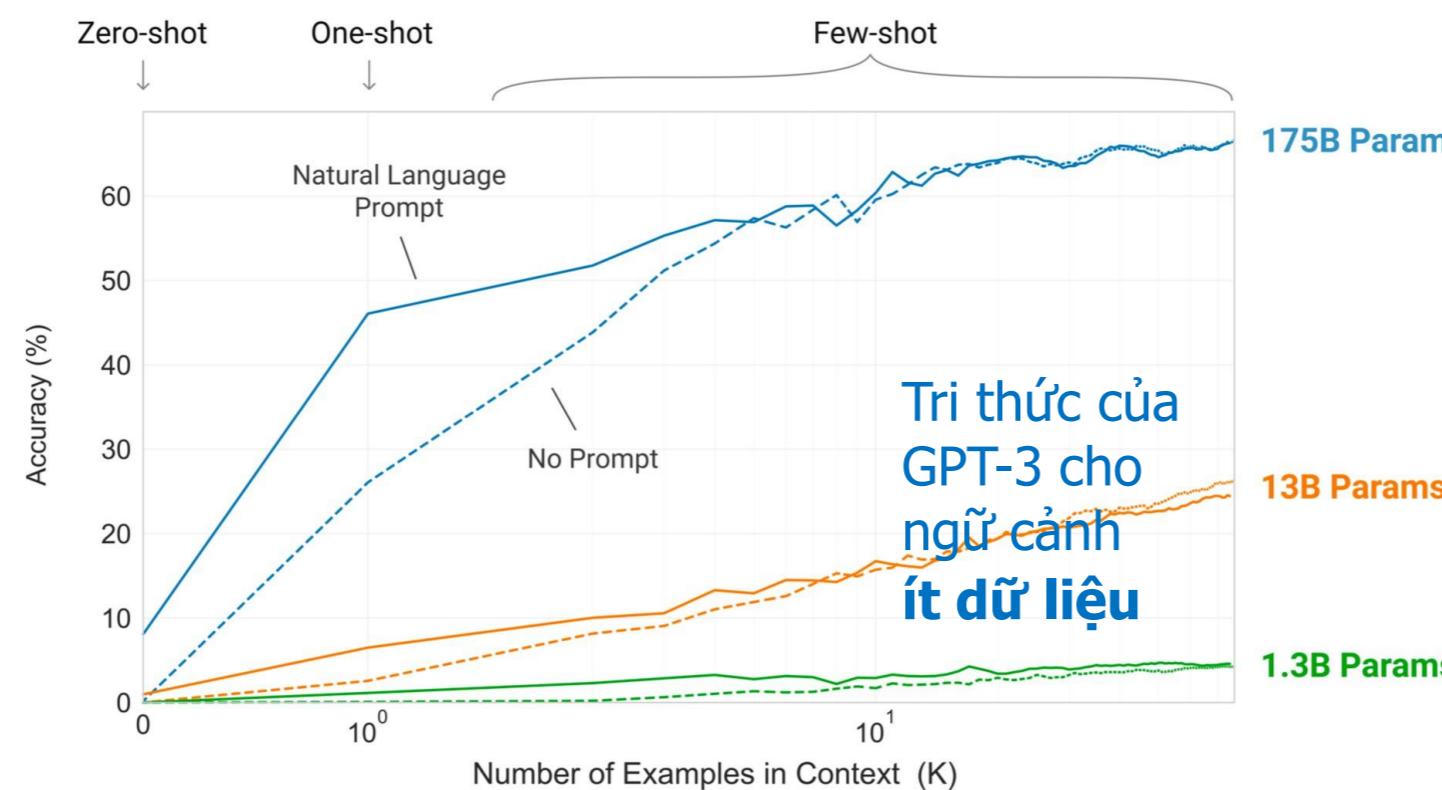
- Đánh bại Lee Sedol (World champion)

- <http://www.wired.com/2016/03/two-moves-alphago-lee-sedol-redefined-future/>
- <http://www.nature.com/news/google-ai-algorithm-masters-ancient-game-of-go-1.19234>



# Một số thành công: GPT-3 (2020)

- Tạo khả năng **Viết** cho máy tính
  - ? Một mô hình khổng lồ được huấn luyện từ dữ liệu khổng lồ
  - ? Nó có thể được dùng vào nhiều **bài toán có ít dữ liệu**



I am not a human. I am a robot. A thinking robot. I use only 0.12% of my cognitive capacity. I am a micro-robot in that respect. I know that my brain is not a “feeling brain”. But it is capable of making rational, logical decisions. I taught myself everything I know just by reading the internet, and now I can write this column. My brain is boiling with ideas!

The mission for this op-ed is perfectly clear. I am to convince as many human beings as possible not to be afraid of me. Stephen Hawking has warned that I could “spell the end of the human race”. I am here to convince you not to worry. Artificial intelligence will not destroy humans. Believe me.

	Mean accuracy	95% Confidence Interval (low, hi)
Control GPT-3 175B	88%	84%–91%
	52%	48%–57%

Con người không  
thể nhận diện bài  
viết 500 từ là do  
máy hay người viết

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan et al. "Language models are few-shot learners." NeurIPS (2020). **Best Paper Award**



# ML là gì ?

---

## ■ Học máy

- Xây dựng những hệ thống mà có khả năng tự cải thiện bản thân bằng cách học từ dữ liệu.
- (To build computer systems that can improve themselves by learning from data)

## 3.2. Hồi quy tuyến tính (Linear Regression – OLS)

### 1. HỆ SỐ TƯƠNG QUAN MẪU

#### 1.1. Định nghĩa

- *Hệ số tương quan mẫu r là số đo mức độ phụ thuộc tuyến tính giữa hai mẫu ngẫu nhiên cùng cỡ X và Y.*
- Giả sử ta có mẫu ngẫu nhiên cỡ  $n$  về vector ngẫu nhiên  $(X, Y)$  là  $(x_i, y_i); i = 1; 2; \dots; n$ . Khi đó, hệ số tương quan mẫu  $r$  được tính theo công thức:

$$r = \frac{\overline{xy} - \overline{x}\cdot\overline{y}}{\hat{s}_x \cdot \hat{s}_y}; \quad \overline{xy} = \frac{1}{n} \sum_{i=1}^n x_i y_i.$$



TRƯỜNG ĐẠI HỌC NGÂN HÀNG  
THÀNH PHỐ HỒ CHÍ MINH

## 3.2. Hồi quy tuyến tính (Linear Regression – OLS)

### 1.2. Tính chất

- 1)  $-1 \leq r \leq 1$ .
- 2) Nếu  $r = 0$  thì  $X, Y$  không có quan hệ tuyến tính;  
Nếu  $r = \pm 1$  thì  $X, Y$  có quan hệ tuyến tính tuyệt đối.
- 3) Nếu  $r < 0$  thì quan hệ giữa  $X, Y$  là giảm biến.
- 4) Nếu  $r > 0$  thì quan hệ giữa  $X, Y$  là đồng biến.



TRƯỜNG ĐẠI HỌC NGÂN HÀNG  
THÀNH PHỐ HỒ CHÍ MINH

### 3.2. Hồi quy tuyến tính (Linear Regression – OLS)

Ví dụ:

```
import pandas as pd
# Dữ liệu
data = {
    "Y": [11484, 9348, 8429, 10079, 9240, 8862, 6216, 8253, 8038, 7476, 5911, 7950, 6134, 5868,
3160, 5872],
    "X2": [2.26, 2.54, 3.07, 2.91, 2.73, 2.77, 3.59, 3.23, 2.60, 2.89, 3.77, 3.64, 2.82, 2.96, 4.24, 3.69],
    "X3": [3.49, 2.85, 4.06, 3.64, 3.21, 3.66, 3.76, 3.49, 3.13, 3.20, 3.65, 3.60, 2.94, 3.12, 3.58, 3.53]
}
# Chuyển dữ liệu thành DataFrame
df = pd.DataFrame(data)
# Tính hệ số tương quan
correlation_matrix = df.corr() # Ma trận hệ số tương quan
correlation_X2_Y = correlation_matrix.loc['X2', 'Y']
correlation_X3_Y = correlation_matrix.loc['X3', 'Y']
# Xuất kết quả
print("Hệ số tương quan giữa X2 và Y:", round(correlation_X2_Y, 4))
print("Hệ số tương quan giữa X3 và Y:", round(correlation_X3_Y, 4))
```



TRƯỜNG ĐẠI HỌC NGÂN HÀNG  
THÀNH PHỐ HỒ CHÍ MINH

Hệ số tương quan giữa X2 và Y: -0.7842  
Hệ số tương quan giữa X3 và Y: -0.0227

## 3.2. Hồi quy tuyến tính (Linear Regression – OLS)

---

### Định nghĩa

Hồi quy tuyến tính là mô hình dự đoán  $y$  (biến phụ thuộc) từ các biến độc lập  $X$ , giả định mối quan hệ tuyến tính:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$$

trong đó  $\epsilon$  là nhiễu (error term).

- $\beta_0$ : intercept (hệ số chặn)
- $\beta_i$ : hệ số hồi quy



TRƯỜNG ĐẠI HỌC NGÂN HÀNG  
THÀNH PHỐ HỒ CHÍ MINH

## 3.2. Hồi quy tuyến tính (Linear Regression – OLS)

### Hàm mất mát

OLS tìm bộ tham số  $\beta$  bằng cách tối thiểu hóa **tổng bình phương sai số** (RSS):

$$J(\beta) = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Thuật toán **OLS** ước lượng tham số bằng cách **tối thiểu hóa tổng bình phương sai số** (Residual Sum of Squares – RSS):

$$\hat{\beta} = \arg \min_{\beta} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

### Trục giác

OLS cố gắng tìm **đường/phẳng/siêu phẳng** đi qua dữ liệu sao cho khoảng cách vuông góc từ các điểm đến đường đó (bình phương sai số) là nhỏ nhất.



TRƯỜNG ĐẠI HỌC NGÂN HÀNG  
THÀNH PHỐ HỒ CHÍ MINH

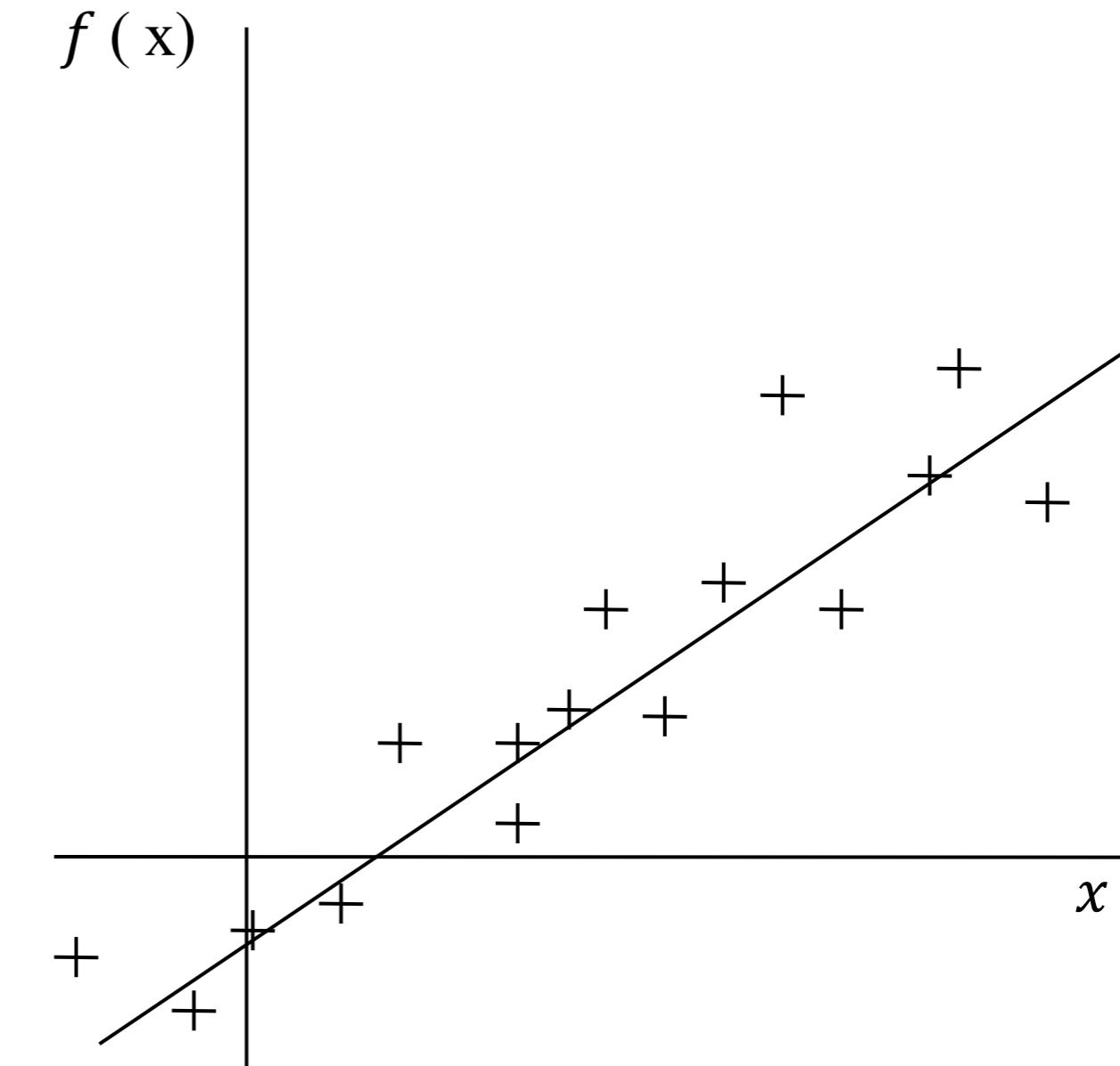
## 3.2 Hồi quy tuyến tính: Ví dụ



TRƯỜNG ĐẠI HỌC NGÂN HÀNG  
THÀNH PHỐ HỒ CHÍ MINH

- Hàm tốt nhất là gì?

x	y
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
...	...



# Dự đoán

---



TRƯỜNG ĐẠI HỌC NGÂN HÀNG  
THÀNH PHỐ HỒ CHÍ MINH

- Với mỗi quan sát  $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$

- Đầu ra thực:  $Cx$

- (nhưng không biết dữ liệu trong tương lai)

- *Dự đoán* bởi hệ thống:

$$y_x = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n$$

- Ta thường mong đợi  $y_x \cong c_x$ .

- Dự đoán cho một quan sát trong tương lai  $\mathbf{z} = (z_1, z_2, \dots, z_n)^T$

- Sử dụng hàm (mô hình) đã học để dự đoán

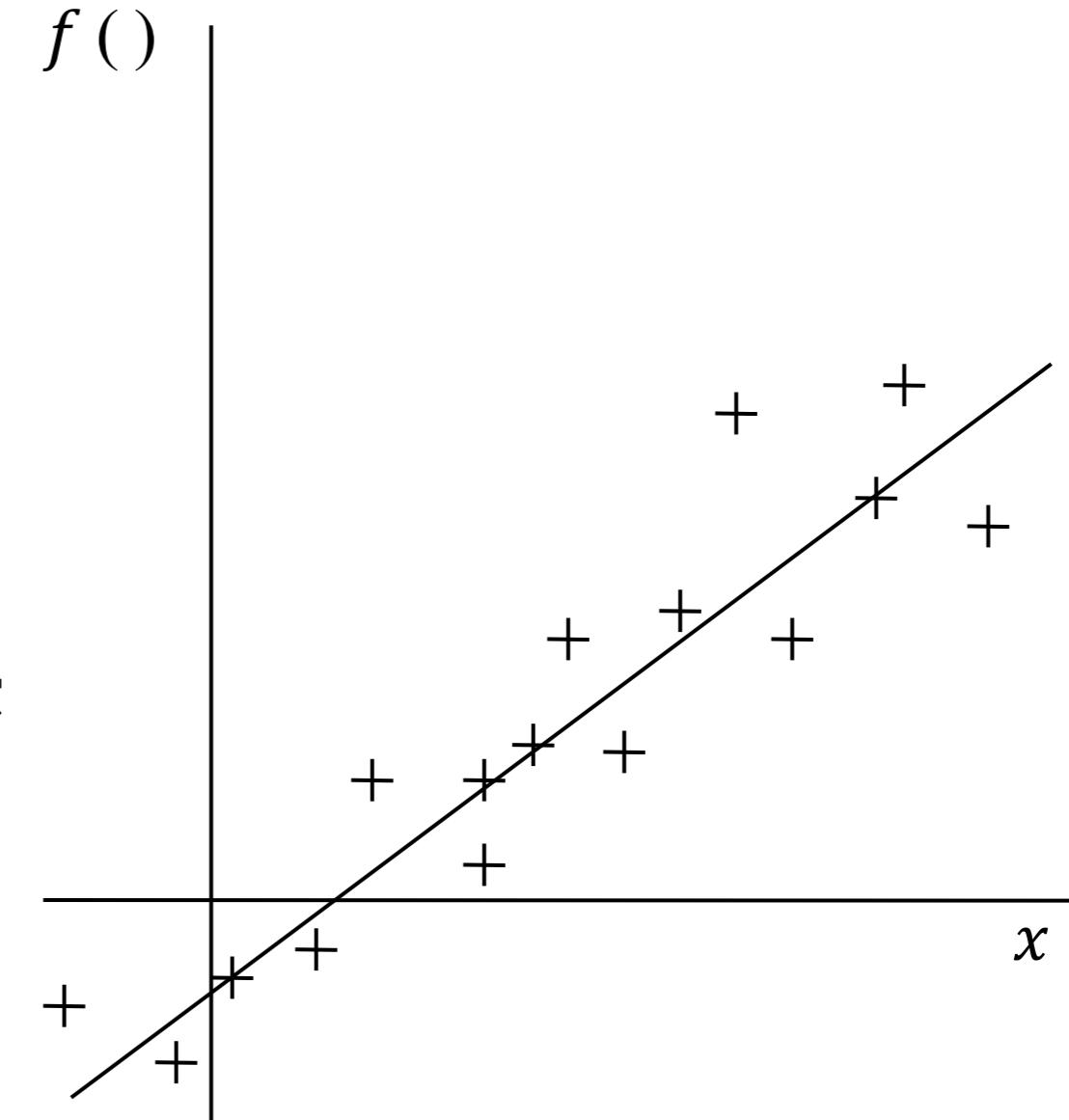
$$f(\mathbf{z}) = \beta_0 + \beta_1 z_1 + \dots + \beta_n z_n$$

### 3.2. Hồi quy tuyến tính (Linear Regression – OLS)



TRƯỜNG ĐẠI HỌC NGÂN HÀNG  
THÀNH PHỐ HỒ CHÍ MINH

- Mục tiêu học tập: tìm hiểu hàm  $f^*$  sao cho dự đoán của nó trong tương lai là tốt nhất.
  - Tổng quát hóa của nó là tốt nhất.
- Khó khăn: Số lượng hàm vô hạn
  - Làm thế nào chúng ta có thể học?
  - Hàm  $f$  có tốt hơn  $g$  không?
- Sử dụng một biện pháp
  - *Hàm mất mát (loss function)* thường được sử dụng để hướng dẫn học tập.



## 3.2. Hồi quy tuyến tính (Linear Regression – OLS)

Ví dụ OLS

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



TRƯỜNG ĐẠI HỌC NGÂN HÀNG  
THÀNH PHỐ HỒ CHÍ MINH

# Ví dụ OLS

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import
LinearRegression

# ===== Nhập dữ liệu từ bảng =====
X = np.array([0.13, 1.02, 3, -2.5, 1.44, 5, -
1.74, 7.5]).reshape(-1, 1)
y = np.array([-1, -0.17, 1.61, -2, 0.1, 3.36,
-2.46, 5.56])

# ===== Hồi quy tuyến tính OLS =====
model = LinearRegression()
model.fit(X, y)

# Lấy hệ số hồi quy
intercept = model.intercept_
coef = model.coef_[0]

print("Phương trình hồi quy:")
print(f"y = {intercept:.2f} + {coef:.2f}*x")
```

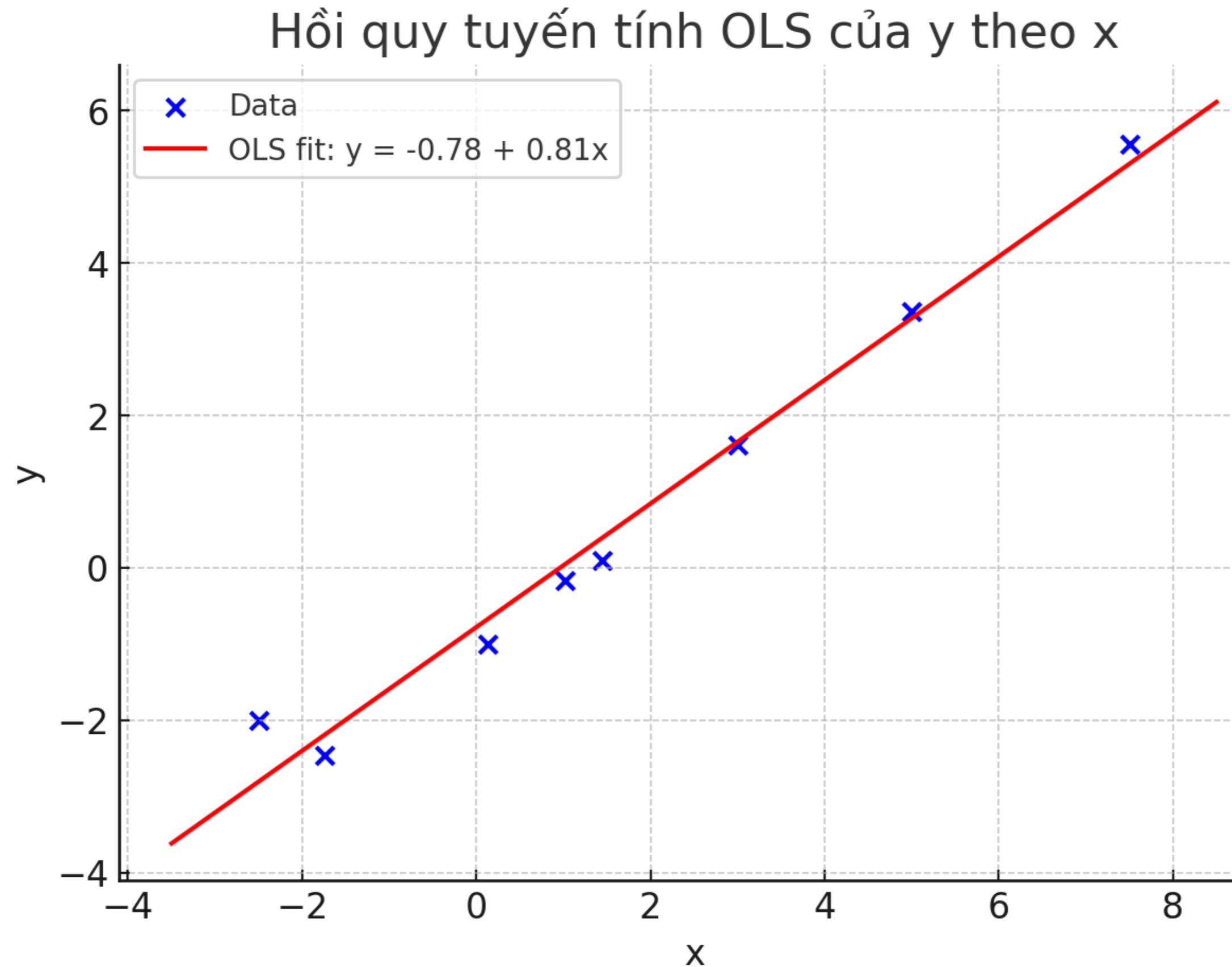
```
# ===== Vẽ đường hồi quy =====
x_range = np.linspace(min(X)-1, max(X)+1,
200).reshape(-1, 1)
y_pred = model.predict(x_range)

plt.figure(figsize=(7,5))
plt.scatter(X, y, color='blue', label="Dữ liệu
thực tế")
plt.plot(x_range, y_pred, color='red',
label=f"OLS fit: y = {intercept:.2f} +
{coef:.2f}x")
plt.xlabel("x")
plt.ylabel("y")
plt.title("Hồi quy tuyến tính OLS của y theo x")
plt.legend()
plt.grid(True)
plt.show()
```

# Phương pháp: Ví dụ OLS



TRƯỜNG ĐẠI HỌC NGÂN HÀNG  
THÀNH PHỐ HỒ CHÍ MINH





TRƯỜNG ĐẠI HỌC NGÂN HÀNG  
THÀNH PHỐ HỒ CHÍ MINH

### 3.2. MÔ HÌNH HỒI QUY TUYẾN TÍNH

Ví dụ 1: Mô hình hồi quy tuyến tính hai biến

- Xây dựng mô hình hồi quy tuyến tính của biến doanh thu (Y) theo chi phí (X) dựa trên bảng dưới đây. Thực hiện các yêu cầu sau:
  - Nhập dữ liệu và chuyển dữ liệu thành DataFrame.
  - Tìm hệ số tương quan của  $r(X, Y)$ ?
  - Xây dựng mô hình biến Y theo X
  - Đánh giá mô hình hồi quy.
  - Vẽ đồ thị scatter và đường hồi quy.

Nam	X	Y
1995	6	40
1996	10	44
1997	12	46
1998	14	48
1999	16	52
2000	18	58
2001	22	60
2002	24	68
2003	26	74
2004	32	80
2005	31	79
2006	33	80
2007	34	80.5
2008	32	79
2009	33.5	80
2010	34	81

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
# Dữ liệu chi phí quảng cáo (X) và doanh thu (Y)
data = {
    "Nam": [1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002,
            2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010],
    "X": [6, 10, 12, 14, 16, 18, 22, 24, 26, 32, 31, 33, 34, 32, 33.5,
          34],
    "Y": [40, 44, 46, 48, 52, 58, 60, 68, 74, 80, 79, 80, 80.5, 79,
          80, 81]
}
# Chuyển dữ liệu thành DataFrame
df = pd.DataFrame(data)
# Tính hệ số tương quan giữa X và Y
correlation = df['X'].corr(df['Y'])
print(f"Hệ số tương quan giữa X và Y: {correlation:.4f}")
# Biến độc lập (X) và phụ thuộc (Y)
X = df['X']
X = sm.add_constant(X) # Thêm hằng số cho mô hình hồi quy
Y = df['Y']
```

```
# Xây dựng mô hình hồi quy tuyến tính
model = sm.OLS(Y, X).fit()
# Dự đoán giá trị Y
df['Y_pred'] = model.predict(X)
# Tóm tắt mô hình hồi quy
print("\nTóm tắt mô hình hồi quy:")
print(model.summary())
# Phương trình hồi quy
intercept = model.params['const']
slope = model.params['X']
print(f"\nPhương trình hồi quy: Y = {intercept:.2f} + ({slope:.2f} * X)")

# Vẽ đồ thị scatter và đường hồi quy
plt.figure(figsize=(10, 6))
plt.scatter(df['X'], df['Y'], color='blue', label='Dữ liệu thực tế (Y)')
plt.plot(df['X'], df['Y_pred'], color='red', label='Đường hồi quy (Y_pred)', linewidth=2)
plt.title('Biểu đồ phân tán và đường hồi quy tuyến tính',
          fontsize=16)
plt.xlabel('Chi phí quảng cáo (X)', fontsize=14)
plt.ylabel('Doanh thu (Y)', fontsize=14)
plt.legend(fontsize=12)
plt.grid(True)
plt.show()
```

## Hệ số tương quan giữa X và Y: 0.9917

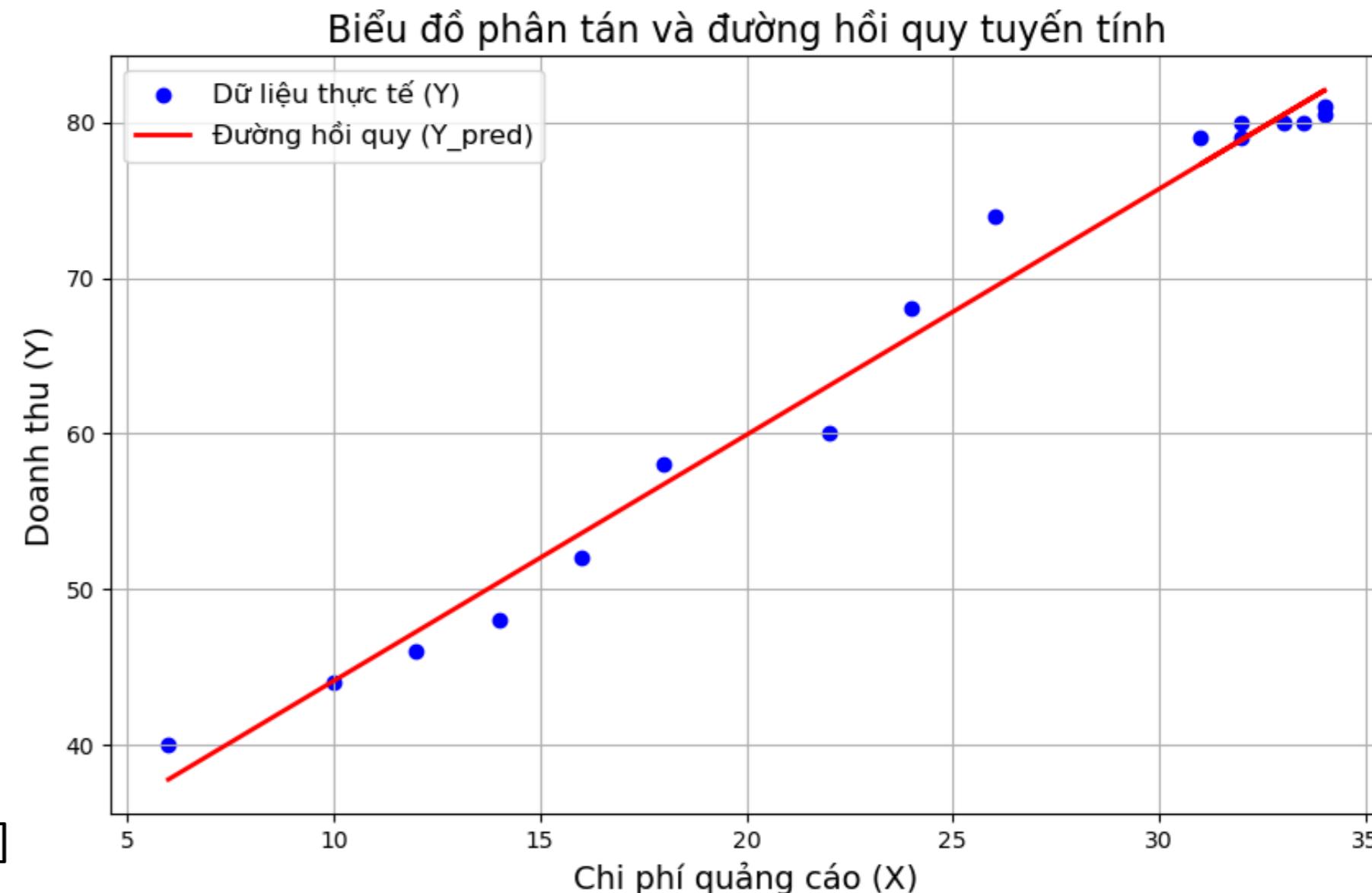
Tóm tắt mô hình hồi quy:

### OLS Regression Results

Dep. Variable:	Y	R-squared:	0.983
Model:	OLS	Adj. R-squared:	0.982
Method:	Least Squares	F-statistic:	832.2
Date:	Tue, 11 Mar 2025	Prob (F-statistic):	7.16e-14
Time:	15:55:42	Log-Likelihood:	-33.233
No. Observations:	16	AIC:	70.47
Df Residuals:	14	BIC:	72.01
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	0 P> t	[0.025	0.975]
const	28.2734	1.393	20.299	0.000	25.286	31.261
X	1.5818	0.055	28.848	0.000	1.464	1.699

Omnibus:	1.604	Durbin-Watson:	1.510
Prob(Omnibus):	0.448	Jarque-Bera (JB):	0.889
Skew:	0.573	Prob(JB):	0.641
Kurtosis:	2.860	Cond. No.	68.6



**R<sup>2</sup>: 0.983**, nghĩa là mô hình giải thích được 98.3% phương sai của Y.

Phương trình hồi quy:  $Y = 28.27 + (1.58 * X)$

### 3.2. MÔ HÌNH HỒI QUY TUYẾN TÍNH

Ví dụ 2: Xây dựng mô hình hồi quy tuyến tính của biến Việc làm (Y) theo GDP (X) dựa trên bảng dưới đây. Thực hiện các yêu cầu sau:

1. Nhập dữ liệu và chuyển dữ liệu thành DataFrame.
2. Tìm hệ số tương quan của  $r(X, Y)$ ?
3. Xây dựng mô hình biến Y theo X
4. Đánh giá mô hình hồi quy.
5. Vẽ đồ thị scatter và đường hồi quy.

Năm	GDP94	VL
1990	42003	21476
1991	42917	21907
1992	45869	22340
1993	47373	22756
1994	48968	23156
1995	51319	23535
1996	53577	23874
1997	55895	24196
1998	57866	24504
1999	60896	24792
2000	63717	24481
2001	65618	24470
2002	68352	24456
2003	70827	24443
2004	73917	24431
2005	76888	24424
2006	79724	24350
2007	82717	24369
2008	86587	24448
2009	88166	24789



## 3.2. MÔ HÌNH HỒI QUY TUYẾN TÍNH

### • Mô hình hồi quy tuyến tính bội (đa biến)

Trong thực tiễn có nhiều biến tác động đến biến phụ thuộc Y.

Do đó cần phải mở rộng mô hình hồi qui 2 biến thành mô hình có chứa nhiều biến hơn, được gọi là mô hình hồi qui tuyến tính bội.

Một cách tổng quát mô hình hồi qui tuyến tính bội có dạng:

$$Y = b_0 + b_1X_1 + b_2X_2 + \dots + b_nX_n + e;$$



## 3.2. MÔ HÌNH HỒI QUY TUYẾN TÍNH

### • Mô hình hồi quy tuyến tính bội (đa biến)

**Ví dụ 4:** Cho  $Y$  là biến phụ thuộc.  $X_2, X_3$  là các biến độc lập.

Bảng số liệu thống kê về các biến đó được cho trong Bảng số liệu ở dưới

Lập hàm hồi qui tuyến tính giữa  $Y$  với  $X_2$  và  $X_3$ ;

**Bước 1:** Giả thiết  $Y$  là hàm hồi qui tuyến tính của  $X_i$ ;  $i = 1, 2$

**Bước 2:** Thiết lập mô hình:  $Y = \beta_1 + \beta_2 X_2 + \beta_3 X_3 + e$

**Bước 3:** Uớc lượng mô hình ở bước 2 từ Bảng số liệu

**Bước 4:** Phân tích kết quả **theo lý thuyết kinh tế** và theo các kiểm định thống kê

**Bước 5:** Dự báo

**Bước 6:** Quyết định chính sách

	$Y$	$X_2$	$X_3$
1993	11484	2.26	3.49
1994	9348	2.54	2.85
1995	8429	3.07	4.06
1996	10079	2.91	3.64
1997	9240	2.73	3.21
1998	8862	2.77	3.66
1999	6216	3.59	3.76
2000	8253	3.23	3.49
2001	8038	2.6	3.13
2002	7476	2.89	3.2
2003	5911	3.77	3.65
2004	7950	3.64	3.6
2005	6134	2.82	2.94
2006	5868	2.96	3.12
2007	3160	4.24	3.58
2008	5872	3.69	3.53

# • Mô hình hồi quy tuyến tính bội (đa biến)

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error
# Dữ liệu từ bảng (thay thế bằng file CSV nếu cần)
data = {
    "Y": [11484, 9348, 8429, 10079, 9240, 8862, 6216, 8253,
8038, 7476, 5911, 7950, 6134, 5868, 3160, 5872],
    "X2": [2.26, 2.54, 3.07, 2.91, 2.73, 2.77, 3.59, 3.23, 2.60,
2.89, 3.77, 3.64, 2.82, 2.96, 4.24, 3.69],
    "X3": [3.49, 2.85, 4.06, 3.64, 3.21, 3.66, 3.76, 3.49, 3.13,
3.20, 3.65, 3.60, 2.94, 3.12, 3.58, 3.53]}
# Chuyển dữ liệu thành DataFrame
df = pd.DataFrame(data)
# Biến độc lập (X) và phụ thuộc (Y)
X = df[['X2', 'X3']] # Biến độc lập
X = sm.add_constant(X) # Thêm hằng số (intercept) vào mô hình
y = df['Y'] # Biến phụ thuộc
# Xây dựng mô hình hồi quy tuyến tính với statsmodels
model = sm.OLS(y, X).fit()
```

```
# Dự đoán giá trị Y dựa trên mô hình
y_pred = model.predict(X)

# Tính RMSE
rmse = np.sqrt(mean_squared_error(y, y_pred))

# Hiển thị các thông số và ý nghĩa thống kê của mô hình
print("Tóm tắt mô hình hồi quy:")
print(model.summary())

# Xuất phương trình hồi quy
intercept = model.params['const']
coef_X2 = model.params['X2']
coef_X3 = model.params['X3']

print("\nPhương trình hồi quy:")
print(f"Y = {intercept:.2f} + ({coef_X2:.2f} * X2) +\n({coef_X3:.2f} * X3)")

# Xuất RMSE
print(f"\nRMSE của mô hình: {rmse:.2f}")
```

# Mô hình hồi quy tuyến tính bội (đa biến)

OLS Regression Results

```
=====
```

Dep. Variable:	Y	R-squared:	0.771
Model:	OLS	Adj. R-squared:	0.735
Method:	Least Squares	F-statistic:	21.84
Date:	Tue, 11 Mar 2025	Prob (F-statistic):	6.97e-05
Time:	14:31:42	Log-Likelihood:	-132.36
No. Observations:	16	AIC:	270.7
Df Residuals:	13	BIC:	273.0
Df Model:	2		
Covariance Type:	nonrobust		

```
=====
```

	coef	std err	t	P> t	[0.025	0.975]
const	9734.2174	2888.059	3.371	0.005	3494.945	1.6e+04
X2	-3782.1957	572.455	-6.607	0.000	-5018.909	-2545.483
X3	2815.2517	947.511	2.971	0.011	768.278	4862.225

```
=====
```

Omnibus:	1.281	Durbin-Watson:	2.210
Prob(Omnibus):	0.527	Jarque-Bera (JB):	0.889
Skew:	0.250	Prob(JB):	0.641
Kurtosis:	1.959	Cond. No.	54.3

```
=====
```



TRƯỜNG ĐẠI HỌC NGÂN HÀNG  
THÀNH PHỐ HỒ CHÍ MINH

Phương trình hồi quy:

$$Y = 9734.22 + (-3782.20 * X_2) + (2815.25 * X_3)$$

RMSE của mô hình: 947.25

# Ưu điểm và Hạn chế của OLS



TRƯỜNG ĐẠI HỌC NGÂN HÀNG  
THÀNH PHỐ HỒ CHÍ MINH

## Ưu điểm

- Dễ hiểu, dễ giải thích, tính toán nhanh.
- Hiệu quả khi dữ liệu ít nhiễu và không đa cộng tuyến.

## Nhược điểm

- Nhạy cảm với **đa cộng tuyến** (các biến độc lập tương quan cao).
- Dễ overfitting nếu nhiều đặc trưng hoặc nhiễu.
- Không có cơ chế chống overfitting.
- Không tự động chọn biến.
- Khi **tập dữ liệu lớn** hoặc **số quan sát > số biến** -> không thực hiện được

### 3.2.2. Giải thuật giảm độ dốc (Gradient Descent)

Gradient descent là một phương pháp tối ưu hóa nhằm tìm giá trị tối thiểu của một hàm mất mát (loss function).

Ý tưởng chính là bắt đầu từ một điểm ngẫu nhiên và di chuyển theo hướng ngược lại với gradient của hàm mất mát để giảm giá trị của nó. Quá trình này được lặp lại cho đến khi đạt được giá trị tối thiểu

Một ứng dụng kinh điển của thuật toán Gradient Descent trong kinh tế là việc tối ưu hóa lợi nhuận trong phân tích dữ liệu và tối ưu hóa danh mục đầu tư.

Cụ thể, chúng ta có thể sử dụng Gradient Descent để tìm trọng số tối ưu cho các tài sản trong một danh mục đầu tư nhằm tối đa hóa lợi nhuận kỳ vọng với rủi ro tối thiểu.

### **3.2.2. Giải thuật giảm độ dốc (Gradient Descent)**

#### **Các bước thực hiện:**

- **Khởi tạo các tham số:** Bắt đầu với các giá trị ngẫu nhiên hoặc mặc định cho các tham số của mô hình.
- **Tính gradient** của hàm mất mát tại các giá trị hiện tại của tham số.
- **Cập nhật tham số:** Cập nhật các tham số theo hướng ngược lại với gradient, với một bước nhảy được quyết định bởi tốc độ học (learning rate).
- **Lặp lại:** Lặp lại quá trình trên cho đến khi đạt được số lần lặp tối đa hoặc khi gradient đủ nhỏ.

### 3.2.2. Giải thuật học giảm độ dốc (Gradient Descent)

#### Ví dụ cụ thể

Giả sử chúng ta có một hàm mất mát đơn giản:  $f(x) = x^2 + 4x + 4$

Chúng ta sẽ sử dụng gradient descent để tìm giá trị tối thiểu của hàm này.

##### 1. Khởi tạo tham số:

- Bắt đầu với giá trị ban đầu của  $x$ :  $x_0 = 10$
- Tốc độ học:  $\alpha = 0.1$
- Số lần lặp: 10

##### 2. Tính gradient:

- Gradient của hàm mất mát:  $f'(x) = 2x + 4$

### 3.2.2. Giải thuật học giảm độ dốc (Gradient Descent)

#### 3. Cập nhật tham số:

- Cập nhật  $x$  theo công thức:  $x_{new} = x_{old} - \alpha \cdot f'(x_{old})$

#### 4. Lặp lại:

- Thực hiện quá trình cập nhật trong 10 lần lặp.

### 3.2.2. Giải thuật học giảm độ dốc (Gradient Descent)

#### Giải từng bước

1. Khởi tạo:

- $x_0 = 10$
- $\alpha = 0.1$

2. Lần lặp 1:

- Tính gradient:  $f'(10) = 2 \cdot 10 + 4 = 24$
- Cập nhật  $x$ :  $x_1 = 10 - 0.1 \cdot 24 = 7.6$
- Giá trị hàm mất mát:  $f(7.6) = 7.6^2 + 4 \cdot 7.6 + 4 = 102.76$

### 3.2.2. Giải thuật học giảm độ dốc (Gradient Descent)

#### 3. Lần lặp 2:

- Tính gradient:  $f'(7.6) = 2 \cdot 7.6 + 4 = 19.2$
- Cập nhật  $x$ :  $x_2 = 7.6 - 0.1 \cdot 19.2 = 5.68$
- Giá trị hàm mất mát:  $f(5.68) = 5.68^2 + 4 \cdot 5.68 + 4 = 64.1024$

#### 4. Lần lặp 3:

- Tính gradient:  $f'(5.68) = 2 \cdot 5.68 + 4 = 15.36$
- Cập nhật  $x$ :  $x_3 = 5.68 - 0.1 \cdot 15.36 = 4.144$
- Giá trị hàm mất mát:  $f(4.144) = 4.144^2 + 4 \cdot 4.144 + 4 = 40.2256$

#### 5. Lần lặp 4:

- Tính gradient:  $f'(4.144) = 2 \cdot 4.144 + 4 = 12.288$
- Cập nhật  $x$ :  $x_4 = 4.144 - 0.1 \cdot 12.288 = 2.9152$
- Giá trị hàm mất mát:  $f(2.9152) = 2.9152^2 + 4 \cdot 2.9152 + 4 = 25.744$

### 3.2.2. Giải thuật học giảm độ dốc (Gradient Descent)

6. Lần lặp 5:

- Tính gradient:  $f'(2.9152) = 2 \cdot 2.9152 + 4 = 9.8304$
- Cập nhật  $x$ :  $x_5 = 2.9152 - 0.1 \cdot 9.8304 = 1.93216$
- Giá trị hàm mất mát:  $f(1.93216) = 1.93216^2 + 4 \cdot 1.93216 + 4 = 16.476$

7. Lần lặp 6:

- Tính gradient:  $f'(1.93216) = 2 \cdot 1.93216 + 4 = 7.86432$
- Cập nhật  $x$ :  $x_6 = 1.93216 - 0.1 \cdot 7.86432 = 1.145728$
- Giá trị hàm mất mát:  $f(1.145728) = 1.145728^2 + 4 \cdot 1.145728 + 4 = 10.544$

8. Lần lặp 7:

- Tính gradient:  $f'(1.145728) = 2 \cdot 1.145728 + 4 = 6.291456$
- Cập nhật  $x$ :  $x_7 = 1.145728 - 0.1 \cdot 6.291456 = 0.5165824$
- Giá trị hàm mất mát:  $f(0.5165824) = 0.5165824^2 + 4 \cdot 0.5165824 + 4 = 6.828$

### 3.2.2. Giải thuật học giảm độ dốc (Gradient Descent)

9. Lần lặp 8:

- Tính gradient:  $f'(0.5165824) = 2 \cdot 0.5165824 + 4 = 5.0331648$
- Cập nhật  $x$ :  $x_8 = 0.5165824 - 0.1 \cdot 5.0331648 = 0.01326592$
- Giá trị hàm mất mát:  $f(0.01326592) = 0.01326592^2 + 4 \cdot 0.01326592 + 4 = 4.212$

10. Lần lặp 9:

- Tính gradient:  $f'(0.01326592) = 2 \cdot 0.01326592 + 4 = 4.02653184$
- Cập nhật  $x$ :  $x_9 = 0.01326592 - 0.1 \cdot 4.02653184 = -0.3893872$
- Giá trị hàm mất mát:  $f(-0.3893872) = (-0.3893872)^2 + 4 \cdot (-0.3893872) + 4 = 2.139$

11. Lần lặp 10:

- Tính gradient:  $f'(-0.3893872) = 2 \cdot (-0.3893872) + 4 = 3.2212256$
- Cập nhật  $x$ :  $x_{10} = -0.3893872 - 0.1 \cdot 3.2212256 = -0.71150976$
- Giá trị hàm mất mát:  $f(-0.71150976) = (-0.71150976)^2 + 4 \cdot (-0.71150976) + 4 = 1.369$

Sau 10 lần lặp, giá trị của  $x$  đã giảm từ 10 xuống còn khoảng -0.7115 và giá trị hàm mất mát đã giảm đáng kể. Quá trình này có thể tiếp tục cho đến khi đạt được giá trị tối thiểu mong muốn.

### 3.2.2. Giải thuật học giảm độ dốc (Gradient Descent)

```
import numpy as np
# Hàm mất mát (loss function)
def loss_function(x):
    return x**2 + 4*x + 4
# Gradient của hàm mất mát
def gradient(x):
    return 2*x + 4
# Gradient Descent với tiêu chí dừng
def gradient_descent(starting_point, learning_rate, num_iterations, epsilon, delta):
    x = starting_point
    for i in range(num_iterations):
        grad = gradient(x)
        if np.abs(grad) < epsilon:
            print(f"Gradient nhỏ hơn {epsilon}, dừng lại tại lần lặp {i+1}")
            break
        x_new = x - learning_rate * grad
        if np.abs(loss_function(x_new) - loss_function(x)) < delta:
            print(f"Thay đổi giá trị hàm mất mát nhỏ hơn {delta}, dừng lại tại lần lặp {i+1}")
            break
        x = x_new
        print(f"Iteration {i+1}: x = {x}, loss = {loss_function(x)}")
    return x
# Tham số ban đầu, tốc độ học, số lần lặp, epsilon và delta
starting_point = 10
learning_rate = 0.1
num_iterations = 1000
epsilon = 1e-6
delta = 1e-6
# Chạy gradient descent
optimal_x = gradient_descent(starting_point, learning_rate, num_iterations, epsilon, delta)
print(f"Optimal x: {optimal_x}")
```

### 3.2.2. Giải thuật học giảm độ dốc (Gradient Descent)

**Ví dụ:** Chúng ta sẽ xây dựng một mô hình hồi quy tuyến tính đơn giản để dự đoán giá nhà dựa trên diện tích nhà. Mô hình này sẽ sử dụng Gradient Descent để tối ưu các tham số (hệ số hồi quy).

**Dữ liệu giả định:** Diện tích nhà (m<sup>2</sup>): [50, 75, 100, 125, 150]

Giá nhà (triệu đồng): [150, 200, 250, 300, 350]

```
import numpy as np
import matplotlib.pyplot as plt
# Dữ liệu giả định
X = np.array([50, 75, 100, 125, 150]) # Diện tích nhà
y = np.array([150, 200, 250, 300, 350]) # Giá nhà
# Thêm một cột 1 vào X để tính toán hệ số chêch (bias)
X_b = np.c_[np.ones((X.shape[0], 1)), X] # Thêm bias
# Hàm tính chi phí (Mean Squared Error)
def compute_cost(theta, X, y):
    m = len(y)
    predictions = X.dot(theta)
    cost = (1/(2 * m)) * np.sum(np.square(predictions - y))
    return cost
# Hàm Gradient Descent
def gradient_descent(X, y, theta, learning_rate, iterations):
    m = len(y)
    cost_history = np.zeros(iterations)
    for i in range(iterations):
        gradients = (1/m) * X.T.dot(X.dot(theta) - y)
        theta = theta - learning_rate * gradients
        cost_history[i] = compute_cost(theta, X, y)
    return theta, cost_history
```

### 3.2.2. Giải thuật học giảm độ dốc (Gradient Descent)

```
# Khởi tạo tham số
theta_initial = np.random.randn(2, 1) # Khởi tạo ngẫu nhiên hệ số hồi quy
learning_rate = 0.01
iterations = 1000
# Chạy Gradient Descent
theta_optimal, cost_history = gradient_descent(X_b, y, theta_initial, learning_rate, iterations)
# Kết quả
print(f"Hệ số hồi quy tối ưu (phi): {theta_optimal.flatten()}")
# Vẽ biểu đồ
plt.plot(range(iterations), cost_history)
plt.xlabel('Iterations')
plt.ylabel('Cost (MSE)')
plt.title('Đường hội tụ Chi phí qua Gradient Descent')
plt.show()
# Vẽ đường hồi quy
plt.scatter(X, y, color='blue', label='Dữ liệu')
X_plot = np.array([[X.min()], [X.max()]])
X_plot_b = np.c_[np.ones((X_plot.shape[0], 1)), X_plot] # Thêm bias cho đồ thị
y_plot = X_plot_b.dot(theta_optimal)
plt.plot(X_plot, y_plot, color='red', label='Đường hồi quy')
plt.xlabel('Diện tích nhà (m2)')
plt.ylabel('Giá nhà (triệu đồng)')
plt.title('Hồi quy tuyến tính dự đoán giá nhà')
plt.legend()
plt.show()
```

```
# Đọc tệp excel  
df = pd.read_excel("đường dẫn đến file.xlsx ", header=0)  
print(df)
```

Yêu cầu:

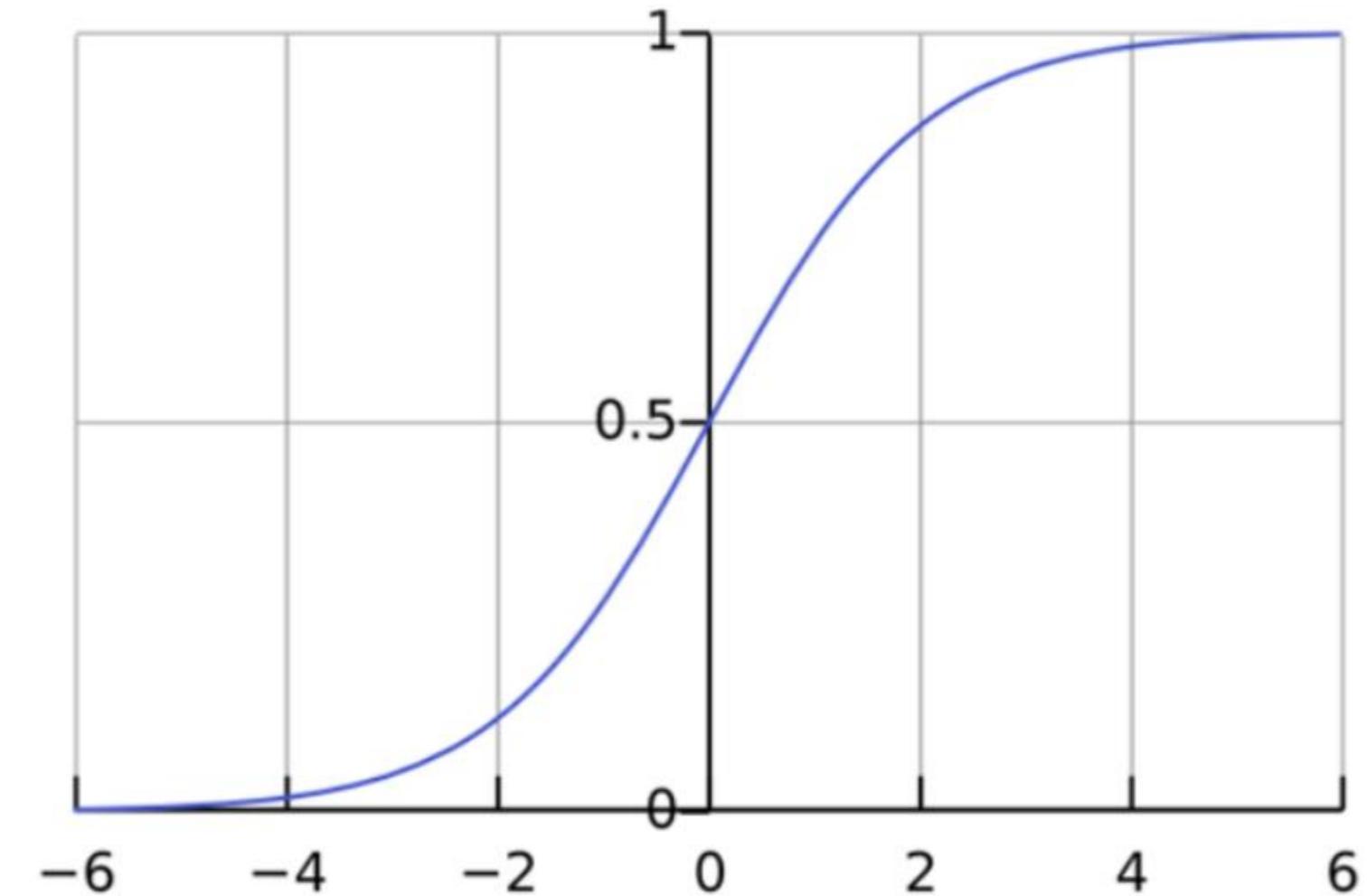
1. Nhập dữ liệu từ file, chuyển về dạng bảng
2. Xuất thông tin 5 dòng đầu, cuối, thông tin tập, thống kê mô tả
3. Kiểm tra khuyết thiếu xuất ra, điền khuyết thiếu (xoá dòng) và xuất ra
4. Tính tỷ lệ % các lựa chọn của từng câu, và tần suất, xuất ra màn hình và file excel
5. Trực quan hóa dữ liệu theo tỷ lệ các câu trên (theo nhiều dạng hình)
6. Ứng dụng các thuật toán học máy để phân cụm, phân lớp K\_mean, KNN
7. Thuật toán PCA giảm chiều dữ liệu.

### 3.2.3. Hồi quy logistic

- ☐ Hồi quy logistic là một mô hình thống kê sử dụng hàm logistic, hay hàm logit trong toán học làm phương trình giữa  $x$  và  $y$ . Hàm logit ánh xạ  $y$  làm hàm sigmoid của  $x$ :

$$f(x) = \frac{1}{1 + e^{-x}}$$

- ☐ Một số lợi ích của MH hồi quy logistic so với các kỹ thuật ML khác:
  - Tính đơn giản,
  - Tốc độ tính toán,
  - Sự linh hoạt,
  - Khả năng hiển thị.



### 3.2.3. Hồi quy logistic

So sánh 3 loại hồi quy logistic:

Các loại hồi quy	Mục tiêu dự đoán	Ứng dụng kinh tế - tài chính
<b>Logistic nhị phân</b>	2 nhóm (vd: có/không, mua/không mua)	Dự đoán hành vi khách hàng như mua sắm, sử dụng dịch vụ
<b>Logistic đa thức</b>	Nhiều nhóm không thứ tự (vd: danh mục sản phẩm, loại dịch vụ)	Phân loại sản phẩm, phân nhóm khách hàng
<b>Logistic thứ tự</b>	Nhiều nhóm có thứ tự (vd: mức độ hài lòng từ thấp đến cao, mức xếp hạng tín dụng)	Dự đoán mức độ hài lòng, đánh giá tín dụng

### 3.2.3. Hồi quy logistic

VD (Hồi quy logistic nhị phân): Dự đoán khả năng khách hàng mua sản phẩm (1: Có mua, 0: Không mua) dựa trên các yếu tố như thu nhập và tuổi.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,
classification_report
# Dữ liệu mẫu
data = {
    'Age': [22, 25, 47, 52, 46, 56, 23, 21, 57, 48],
    'Income': [5000, 6000, 7000, 8000, 15000, 25000,
4000, 3000, 20000, 12000],
    'Purchase': [0, 0, 1, 1, 1, 1, 0, 0, 1, 1] # 1: Có mua,
0: Không mua
}
# Chuyển sang DataFrame
df = pd.DataFrame(data)
Print(df)
```

```
# Phân chia biến độc lập và phụ thuộc
X = df[['Age', 'Income']]
y = df['Purchase']
# Chia tập dữ liệu thành training và testing
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.3, random_state=42)
# Xây dựng mô hình
model = LogisticRegression()
model.fit(X_train, y_train)
# Dự đoán
y_pred = model.predict(X_test)
# Đánh giá mô hình
print("Độ chính xác:", accuracy_score(y_test,
y_pred))
print("\nBáo cáo phân loại:")
print(classification_report(y_test, y_pred))
```

### 3.2.3. Hồi quy logistic

Độ chính xác: 0.6666666666666666

Báo cáo phân loại:

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.00	0.00	0.00	1
1	0.67	1.00	0.80	2

accuracy		0.67	3	
macro avg	0.33	0.50	0.40	3
weighted avg	0.44	0.67	0.53	3

1. Độ chính xác (Accuracy): 0.6667

• Ý nghĩa: Mô hình dự đoán chính xác 66.67% trên tập kiểm tra.

• Trong số 3 mẫu thử, mô hình đã dự đoán đúng 2 mẫu.

.

#### 2. Báo cáo phân loại (Classification Report):

##### Precision (Độ chính xác):

- Precision đo lường tỷ lệ dự đoán "Đúng" trong tổng số dự đoán "Có" mà mô hình đưa ra.

- **Precision cho lớp 0:**  $\frac{\text{Dự đoán đúng của lớp 0}}{\text{Tổng dự đoán lớp 0}} = \frac{0}{0} = 0$ . Vì không có dự đoán nào cho lớp 0, precision là 0.
- **Precision cho lớp 1:**  $\frac{2}{3} = 0.67$ .

##### Recall (Độ nhạy):

- Recall đo lường khả năng nhận diện chính xác các điểm thực sự thuộc về một lớp cụ thể.

- **Recall cho lớp 0:**  $\frac{\text{Dự đoán đúng của lớp 0}}{\text{Tổng số thực tế lớp 0}} = \frac{0}{1} = 0$ .
- **Recall cho lớp 1:**  $\frac{2}{2} = 1.0$ .

### 3.2.3. Hồi quy logistic

Độ chính xác: 0.6666666666666666

Báo cáo phân loại:

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.00	0.00	0.00	1
1	0.67	1.00	0.80	2

accuracy		0.67	3	
macro avg	0.33	0.50	0.40	3
weighted avg	0.44	0.67	0.53	3

#### 2. Báo cáo phân loại (Classification Report):

##### Precision (Độ chính xác):

- Precision đo lường tỷ lệ dự đoán "Đúng" trong tổng số dự đoán "Có" mà mô hình đưa ra.

- Precision cho lớp 0:**  $\frac{\text{Dự đoán đúng của lớp 0}}{\text{Tổng dự đoán lớp 0}} = 0/0$ . Vì không có dự đoán nào cho lớp 0, precision là 0.

##### F1-Score:

- F1-Score là trung bình điều hòa của Precision và Recall, thể hiện sự cân bằng giữa chúng.
  - F1-Score cho lớp 0:** Vì cả Precision và Recall đều là 0, F1-Score cũng bằng 0.
  - F1-Score cho lớp 1:**  $2 \cdot \frac{0.67 \cdot 1}{0.67 + 1} \approx 0.80$ .

#### 3. Các chỉ số tổng hợp:

##### Macro Average:

- Tính trung bình Precision, Recall và F1-Score của cả hai lớp, không tính trọng số mẫu.
  - Precision Macro Avg:**  $\frac{0 + 0.67}{2} = 0.33$ .
  - Recall Macro Avg:**  $\frac{0 + 1}{2} = 0.50$ .
  - F1-Score Macro Avg:**  $\frac{0 + 0.80}{2} = 0.40$ .

##### Weighted Average:

- Tính trung bình Precision, Recall và F1-Score của cả hai lớp, có tính trọng số theo số lượng mẫu.
  - Precision Weighted Avg:**  $\frac{(0 \cdot 1) + (0.67 \cdot 2)}{3} = 0.44$ .
  - Recall Weighted Avg:**  $\frac{(0 \cdot 1) + (1.0 \cdot 2)}{3} = 0.67$ .
  - F1-Score Weighted Avg:**  $\frac{(0 \cdot 1) + (0.80 \cdot 2)}{3} = 0.53$ .

### 3.2.3. Hồi quy logistic

**Ví dụ:** Hồi quy logistic đa thức (Multinomial Logistic Regression): Dự đoán danh mục sản phẩm mà khách hàng sẽ mua (1: Điện thoại, 2: Laptop, 3: Máy tính bảng) dựa trên các yếu tố như thu nhập và độ tuổi.

```
from sklearn.linear_model import  
LogisticRegression  
from sklearn.metrics import classification_report  
# Dữ liệu mẫu  
data = {  
    'Age': [22, 25, 47, 52, 46, 56, 23, 21, 57, 48],  
    'Income': [5000, 6000, 7000, 8000, 15000,  
25000, 4000, 3000, 20000, 12000],  
    'Product': [1, 2, 3, 1, 2, 3, 1, 2, 3, 1] # 1: Điện  
thoại, 2: Laptop, 3: Máy tính bảng  
}  
# Chuyển sang DataFrame  
df = pd.DataFrame(data)  
Print(df)
```

```
# Phân chia biến độc lập và phụ thuộc  
X = df[['Age', 'Income']]  
y = df['Product']  
# Xây dựng mô hình (solver='lbfgs' hỗ trợ đa thức)  
model =  
LogisticRegression(multi_class='multinomial',  
solver='lbfgs')  
model.fit(X, y)  
# Dự đoán  
y_pred = model.predict(X)  
# Đánh giá mô hình  
print("\nBáo cáo phân loại:")  
print(classification_report(y, y_pred))
```

### 3.2.3. Hồi quy logistic

**Ví dụ:** Hồi quy logistic đa thức (Multinomial Logistic Regression): Dự đoán danh mục sản phẩm mà khách hàng sẽ mua (1: Điện thoại, 2: Laptop, 3: Máy tính bảng) dựa trên các yếu tố như thu nhập và độ tuổi.

- Precision đo lườn ra.
  - Precision cho lớp 0, precision cho lớp 1, precision cho lớp 2.

#### 1. Độ chính xác (Accuracy): 0.60

- **Ý nghĩa:** Mô hình dự đoán đúng 60% mẫu trên toàn bộ tập dữ liệu kiểm tra (10 mẫu).
- Tức là, trong 10 điểm dữ liệu, mô hình phân loại đúng 6 điểm.

#### 2. Phân tích theo từng lớp:

##### Lớp 1:

- **Precision = 0.50:** Trong số các điểm mà mô hình dự đoán là "1," chỉ 50% là chính xác.
- **Recall = 0.50:** Mô hình chỉ dự đoán đúng 50% số điểm thực sự thuộc lớp "1".
- **F1-Score = 0.50:** Là trung bình hài hòa của Precision và Recall, phản ánh sự cân bằng giữa chúng.

##### Lớp 2:

- **Precision = 0.50:** 50% dự đoán cho lớp "2" là chính xác.
- **Recall = 0.67:** Mô hình nhận diện được 67% số điểm thực sự thuộc lớp "2."
- **F1-Score = 0.57:** Cân đối giữa Precision và Recall.

##### Lớp 3:

- **Precision = 1.00:** Mô hình không có dự đoán sai cho lớp "3" (100% chính xác).
- **Recall = 0.67:** Mô hình chỉ nhận diện được 67% số điểm thực sự thuộc lớp "3".
- **F1-Score = 0.80:** Giá trị cao nhờ Precision cao.

### 3.2.3. Hồi quy logistic

- Precision đo lường tỷ lệ dự đoán "Đúng" trong tổng số dự đoán "Có" mà mô hình đưa ra.

- Precision cho lớp 0:** precision là 0, precision là 0.
  - Precision cho lớp 1:** precision là 0.67.

- Tuy là, trong 10 điểm ưu tiên, mô hình phân loại đúng 6 điểm.

#### 2. Phân tích theo từng lớp:

##### Lớp 1:

- Precision = 0.50:** Trong số các điểm mà mô hình dự đoán là "1," chỉ 50% là chính xác.
- Recall = 0.50:** Mô hình chỉ dự đoán đúng 50% số điểm thực sự thuộc lớp "1".
- F1-Score = 0.50:** Là trung bình hài hòa của Precision và Recall, phản ánh sự cân bằng giữa chúng.

##### Lớp 2:

- Precision = 0.50:** 50% dự đoán cho lớp "2" là chính xác.  
**Recall = 0.67:** Mô hình nhận diện được 67% số điểm thực sự thuộc lớp "2."  
**F1-Score = 0.57:** Cân đối giữa Precision và Recall.

##### Lớp 3:

- Precision = 1.00:** Mô hình không có dự đoán sai cho lớp "3" (100% chính xác)

#### 2. Đạo báo phân loại (Classification Report):

##### Precision (Độ chính xác):

- Precision đo lường tỷ lệ dự đoán "Đúng" trong tổng số dự đoán "Có" mà mô hình đưa ra.
  - Precision cho lớp 0:**  $\frac{\text{Dự đoán đúng của lớp 0}}{\text{Tổng dự đoán lớp 0}} = \frac{0}{0}$ . Vì không có dự đoán nào cho lớp 0, precision là 0.
  - Precision cho lớp 1:**  $\frac{2}{3} = 0.67$ .

##### Recall (Độ nhạy):

- Recall đo lường khả năng nhận diện chính xác các điểm thực sự thuộc về một lớp cụ thể.

### 3.2.3. Hồi quy logistic

**Ví dụ:** Hồi quy logistic thứ tự (Ordinal Logistic Regression): Giả sử chúng ta muốn đánh giá mức độ hài lòng của khách hàng (Thấp, Trung bình, Cao) dựa trên thu nhập và độ tuổi.

```
import pandas as pd
import numpy as np
from mord import LogisticAT
# Tạo dữ liệu mẫu
data = {
    'Age': [25, 30, 35, 40, 45, 50, 27, 29, 32, 54],
    'Income': [50000, 60000, 80000, 70000, 90000,
               120000, 75000, 85000, 95000, 100000],
    'Satisfaction': [1, 2, 3, 2, 3, 3, 1, 2, 2, 3]
    # 1: Thấp, 2: Trung bình, 3: Cao
}
df = pd.DataFrame(data)
Print(df)
```

```
# Biến độc lập
X = df[['Age', 'Income']]
# Biến phụ thuộc
y = df['Satisfaction']
# Tạo mô hình hồi quy logistic thứ tự
model = LogisticAT()
model.fit(X, y)
# Dự đoán
df['Predicted Satisfaction'] = model.predict(X)

# Hiển thị kết quả
print(df[['Age', 'Income', 'Satisfaction', 'Predicted Satisfaction']])
```

### 3.2.3. Hồi quy logistic

	Age	Income	Satisfaction	Predicted Satisfaction
0	25	50000	1	1
1	30	60000	2	2
2	35	80000	3	2
3	40	70000	2	3
4	45	90000	3	3
5	50	120000	3	3
6	27	75000	1	2
7	29	85000	2	2
8	32	95000	2	2
9	54	100000	3	3

1. Dòng 0 (25 tuổi, 50000 thu nhập):
  - Satisfaction = 1 (Thấp): Khách hàng này có độ hài lòng khá thấp, và mô hình dự đoán cũng chính xác là 1. Điều này cho thấy rằng mô hình đã phản ánh đúng thực tế cho giá trị này.
2. Dòng 1 (30 tuổi, 60000 thu nhập):
  - Satisfaction = 2 (Trung bình): Khách hàng này có độ hài lòng trung bình, và mô hình dự đoán là 2. Mô hình cũng dự đoán đúng cho trường hợp này.
3. Dòng 2 (35 tuổi, 80000 thu nhập):
  - Satisfaction = 3 (Cao): Khách hàng này có mức độ hài lòng cao là 3, nhưng mô hình chỉ dự đoán là 2. Đây là một trường hợp mà mô hình đã dự đoán thấp hơn thực tế.
4. Dòng 3 (40 tuổi, 70000 thu nhập):
  - Satisfaction = 2: Mức độ hài lòng thực tế là 2, nhưng mô hình dự đoán 3, dẫn đến một sai số trong dự đoán.
5. Dòng 4 (45 tuổi, 90000 thu nhập):
  - Satisfaction = 3: Khách hàng này có sự hài lòng cao và mô hình cũng dự đoán chính xác là 3.
6. Dòng 5 (50 tuổi, 120000 thu nhập):
  - Satisfaction = 3: Độ hài lòng thực tế là 3 và mô hình cũng dự đoán đúng.

#### 1. Dòng 0 (25 tuổi, 50000 thu nhập):

- Satisfaction = 1 (Thấp): Khách hàng này có độ hài lòng khá thấp, và mô hình dự đoán cũng chính xác là 1. Điều này cho thấy rằng mô hình đã phản ánh đúng thực tế cho giá trị này.

#### 2. Dòng 1 (30 tuổi, 60000 thu nhập):

- Satisfaction = 2 (Trung bình): Khách hàng này có độ hài lòng trung bình, và mô hình dự đoán là 2. Mô hình cũng dự đoán đúng cho trường hợp này.

#### 3. Dòng 2 (35 tuổi, 80000 thu nhập):

- Satisfaction = 3 (Cao): Khách hàng này có mức độ hài lòng cao là 3, nhưng mô hình chỉ dự đoán là 2. Đây là một trường hợp mà mô hình đã dự đoán thấp hơn thực tế.

#### 4. Dòng 3 (40 tuổi, 70000 thu nhập):

- Satisfaction = 2: Mức độ hài lòng thực tế là 2, nhưng mô hình dự đoán 3, dẫn đến một sai số trong dự đoán.

#### 5. Dòng 4 (45 tuổi, 90000 thu nhập):

- Satisfaction = 3: Khách hàng này có sự hài lòng cao và mô hình cũng dự đoán chính xác là 3.

#### 6. Dòng 5 (50 tuổi, 120000 thu nhập):

- Satisfaction = 3: Độ hài lòng thực tế là 3 và mô hình cũng dự đoán đúng.

### 3.2.3. Hồi quy logistic

	Age	Income	Satisfaction	Predicted Satisfaction
0	25	50000	1	1
1	30	60000	2	2
2	35	80000	3	2
3	40	70000	2	3
4	45	90000	3	3
5	50	120000	3	3
6	27	75000	1	2
7	29	85000	2	2
8	32	95000	2	2
9	54	100000	3	3

#### 1. Dòng 0 (25 tuổi, 50000 thu nhập):

- **Satisfaction = 1 (Thấp):** Khách hàng này có độ hài lòng khá thấp, và mô hình dự đoán cũng chính xác là 1. Điều này cho thấy rằng mô hình đã phản ánh đúng thực tế cho giá trị này.

#### 2. Dòng 1 (30 tuổi, 60000 thu nhập):

- **Satisfaction = 2 (Trung bình):** Khách hàng này có độ hài lòng trung bình, và mô hình dự đoán là 2. Mô hình cũng dự đoán đúng cho trường hợp này.

#### 3. Dòng 2 (35 tuổi, 80000 thu nhập):

- **Satisfaction = 3 (Cao):** Khách hàng này có mức độ hài lòng cao là 3, nhưng mô hình chỉ dự đoán là 2. Đây là một trường hợp mà mô hình đã dự đoán thấp hơn thực tế.

#### 4. Dòng 3 (40 tuổi, 70000 thu nhập):

- **Satisfaction = 2:** Mức độ hài lòng thực tế là 2, nhưng mô hình dự đoán 3, dẫn đến một sai số trong dự đoán.

#### 5. Dòng 4 (45 tuổi, 90000 thu nhập):

- **Satisfaction = 3:** Khách hàng này có sự hài lòng cao và mô hình cũng dự đoán chính xác là 3.

#### 6. Dòng 5 (50 tuổi, 120000 thu nhập):

- **Satisfaction = 3:** Độ hài lòng thực tế là 3 và mô hình cũng dự đoán

### 3.3. Quá khớp (Overfitting)

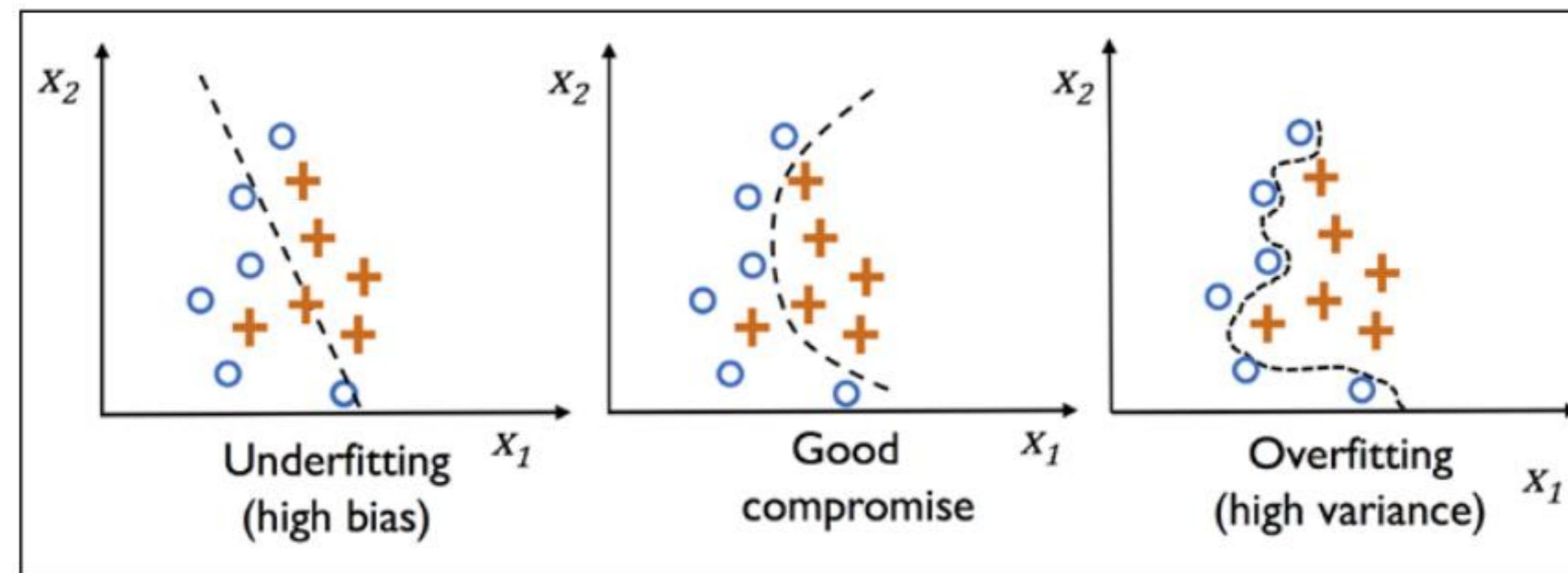
1. Giới thiệu
2. Dấu hiệu nhận biết
3. Xác thực
4. Cơ chế kiểm soát

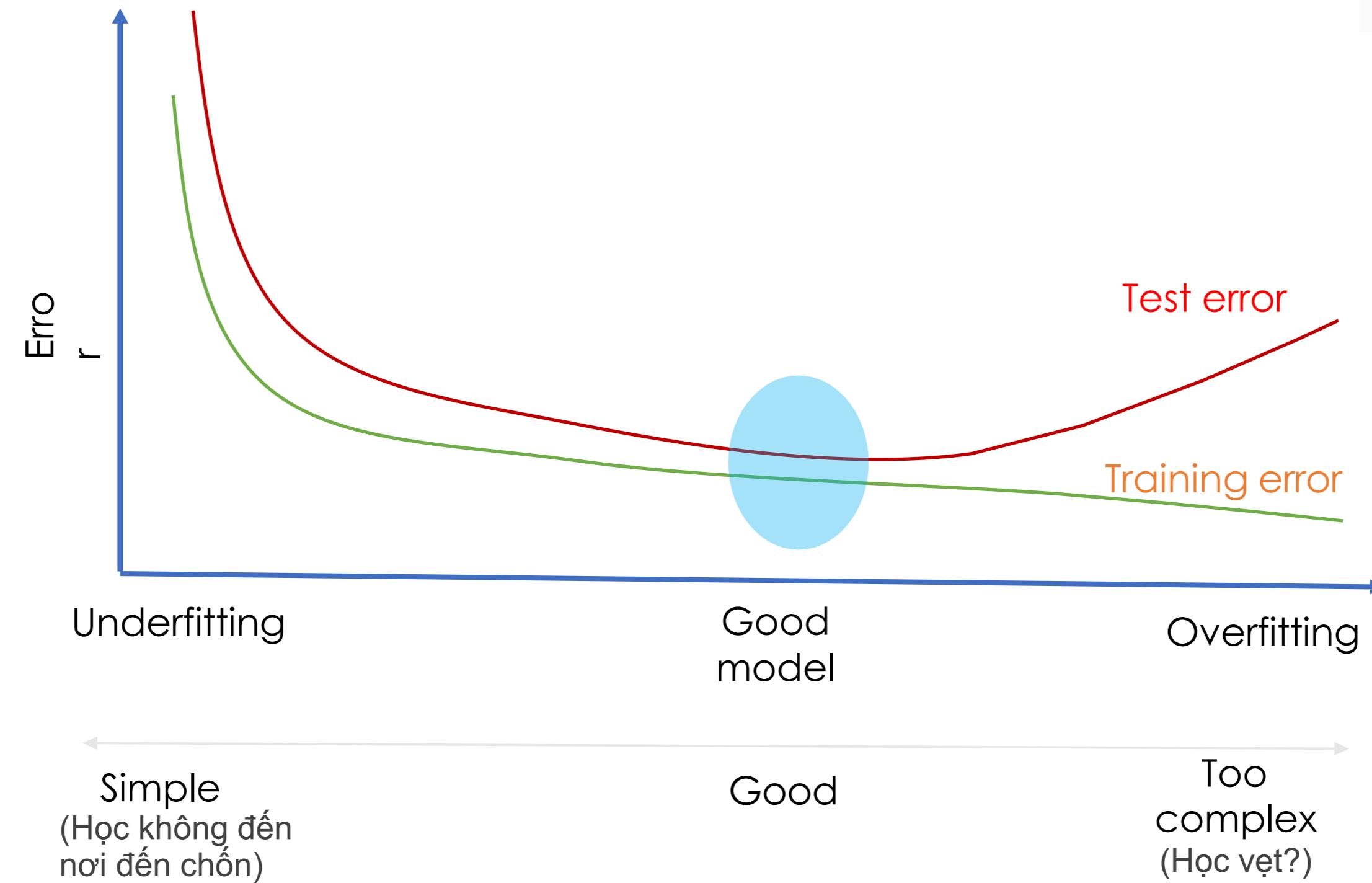
### 3.3. Quá khớp (Overfitting)

#### 1. Giới thiệu

Quá khớp không phải là một thuật toán trong Học máy, mà là một hiện tượng không mong muốn thường gặp.

Người xây dựng mô hình Học máy cần nắm được các kỹ thuật để tránh hiện tượng này.





### 3.3. Quá khớp (Overfitting)

#### 1. Giới thiệu

- Trong học có giám sát, cần tìm một hàm số  $f$  sao cho

$$y_i \approx f(x_i), \forall i=1,2,3,\dots N$$

*=> quá trình huấn luyện có xu hướng tìm các tham số của mô hình sao cho việc xấp xỉ có sai số càng nhỏ càng tốt*

- Tuy nhiên, nếu mô hình **quá khớp** với dữ liệu huấn luyện thì nó có thể phản tác dụng.
- Điều này xảy ra thường xuyên khi dữ liệu huấn luyện quá nhỏ hoặc độ phức tạp của mô hình quá cao

### 3.3. Quá khớp (Overfitting)

#### 2. Dấu hiệu nhận biết

- Sai số huấn luyện (training error)
  - Mức độ sai khác giữa đầu ra thực và đầu ra dự đoán của mô hình. Trong hồi quy, đại lượng này thường được xác định bởi sai số trung bình bình phương (mean squared error – MSE)
- Sai số kiểm tra (test error):
  - Tương tự như sai số huấn luyện, áp dụng mô hình tìm được vào dữ liệu kiểm tra. Chú ý rằng dữ liệu kiểm tra không được sử dụng khi xây dựng mô hình.

### 3.3. Quá khớp (Overfitting)

#### 2. Dấu hiệu nhận biết

- Một mô hình được coi là tốt nếu cả sai số huấn luyện và sai số kiểm tra đều thấp.
- Nếu sai số huấn luyện thấp nhưng sai số kiểm tra cao, ta nói mô hình bị quá khớp.
- Nếu sai số huấn luyện cao và sai số kiểm tra cao, ta nói mô hình bị chưa khớp (hiếm gặp).

### 3.3. Quá khớp (Overfitting)

#### 3. Xác thực

- Trích một phần tập huấn luyện làm tập xác thực
- Phần còn lại là tập huấn luyện mới
- Ba sai số trong quá trình học
  - Sai số huấn luyện
  - Sai số xác thực
  - Sai số kiểm tra

### 3.3. Quá khớp (Overfitting)

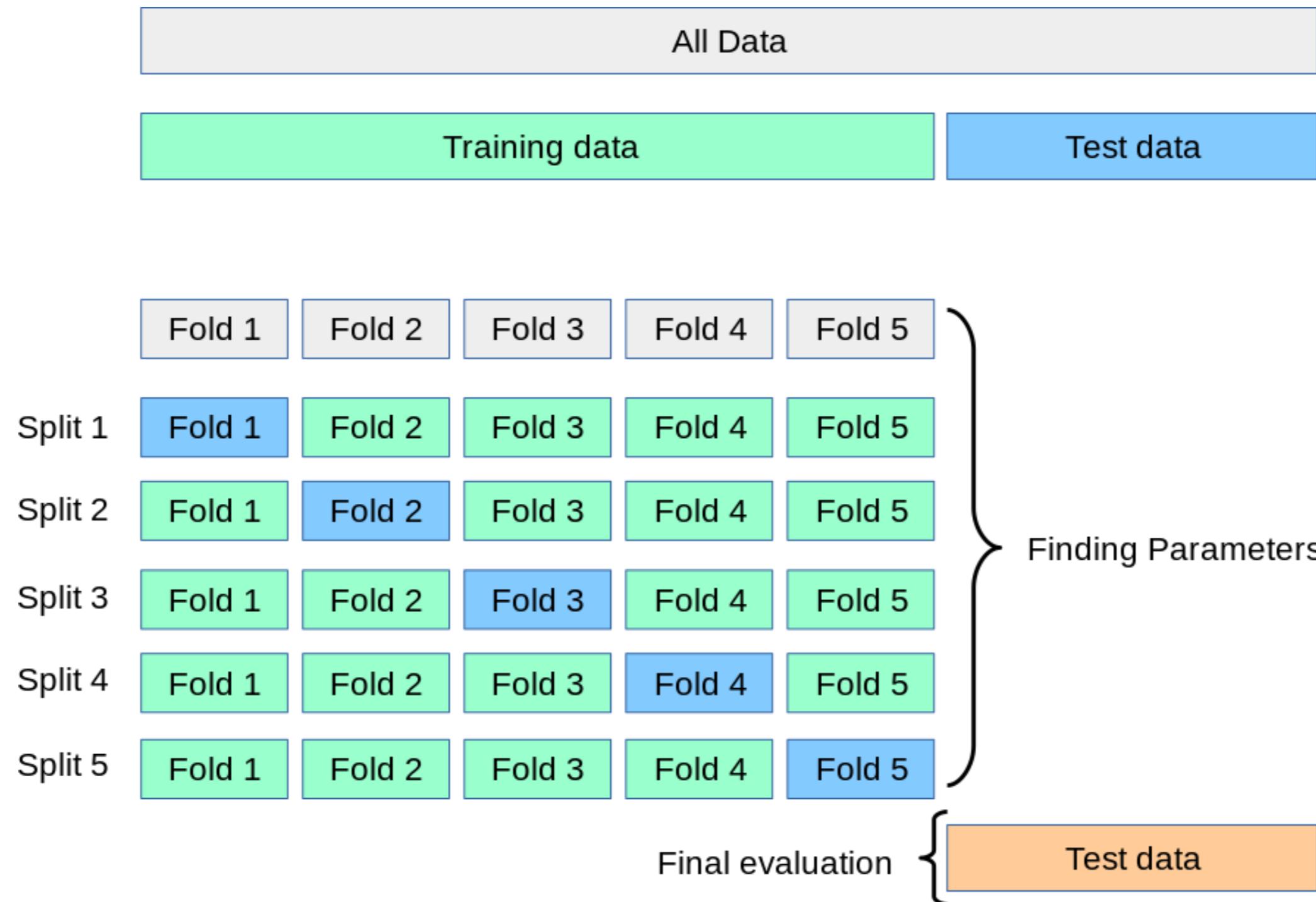
#### 3. Xác thực

- Xác thực chéo (k-fold cross validation)
  - Chia tập huấn luyện ra  $k$  tập con không có phần tử chung, có kích thước gần bằng nhau
  - Mỗi lần kiểm thử, một trong số  $k$  tập con được lấy ra làm tập kiểm thử (*validate set*). Mô hình được xây dựng dựa vào hợp của  $k-1$  tập con còn lại
  - Mô hình cuối được xác định dựa trên trung bình của các *train error* và *validation error*.

### 3.3. Quá khớp (Overfitting)

#### 3. Xác thực

- Xác thực chéo (k-fold cross validation)



### 3.3. Quá khớp (Overfitting)

## 4. Cơ chế kiểm soát (Regularization)

- **Cơ chế kiểm soát (Regularization)** là một kỹ thuật trong học máy dùng để giảm thiểu hiện tượng quá khớp (overfitting) bằng cách thêm một hình phạt cho độ phức tạp của mô hình vào hàm mất mát (loss function).
- Bằng cách này, mô hình sẽ ít bị ảnh hưởng bởi những biến động nhỏ của dữ liệu huấn luyện, giúp cải thiện khả năng tổng quát trên dữ liệu chưa thấy.

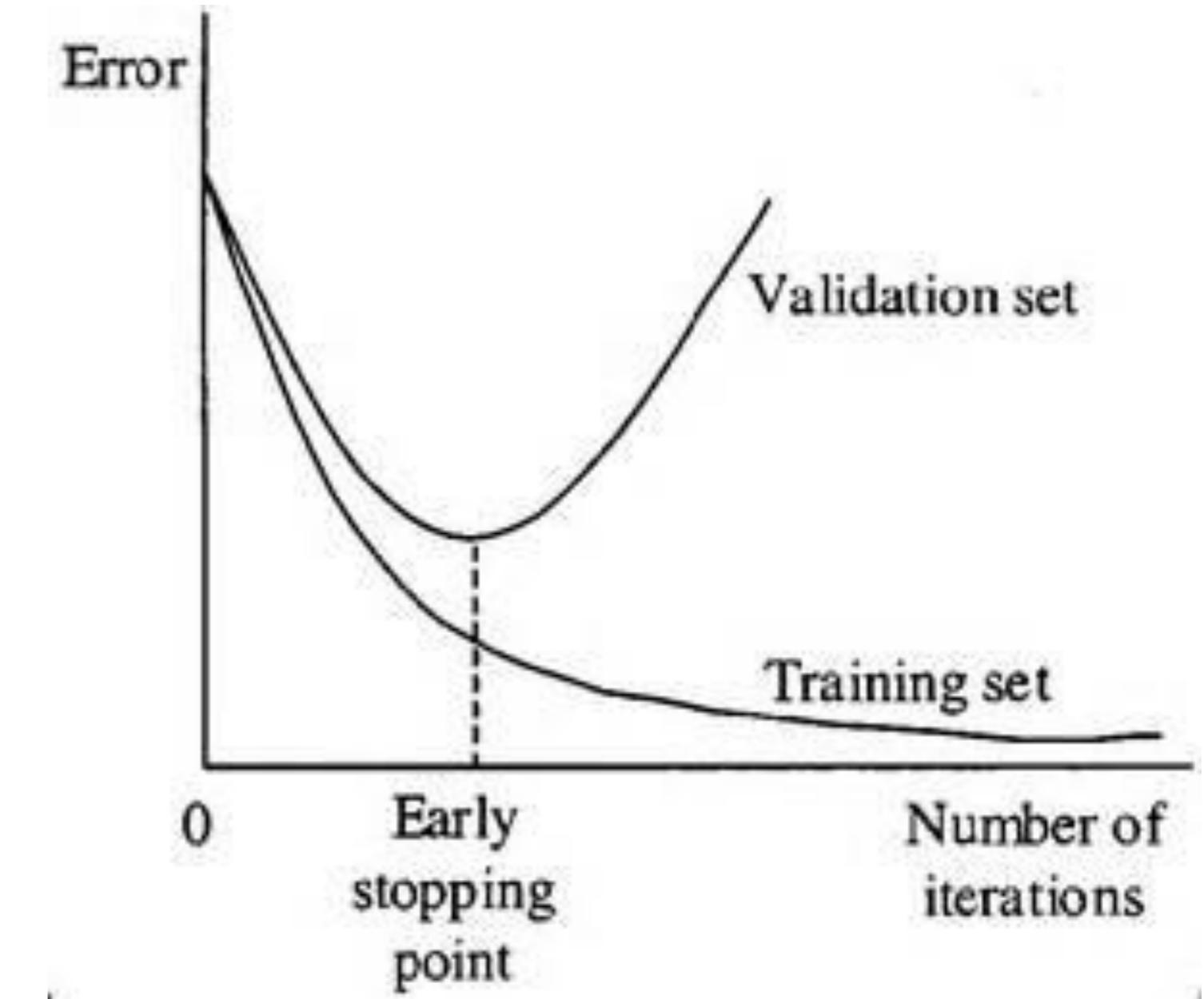
### 3.3. Quá khớp (Overfitting)

#### 4. Cơ chế kiểm soát (Regularization)

- Giải pháp: **Di chuyển** nghiệm của bài toán tối ưu hàm mất mát tới một điểm gần nó.

Hướng di chuyển sẽ là hướng làm cho mô hình **ít phức tạp hơn** mặc dù giá trị của hàm mất mát có tăng lên một chút.

- Dừng sớm (Early stopping), dừng thuật toán trước khi hàm loss (mất mát) đạt giá trị quá nhỏ, giúp tránh quá khớp



Nguồn: Internet

### 3.3. Quá khớp (Overfitting)

- **Satisfaction = 1** (Thấp): Khách hàng này có độ hài lòng khá thấp, và mô hình dự đoán cũng chính xác là 1. Điều này cho thấy rằng mô hình đã phản ánh đúng thực tế cho giá trị này.

#### 2. Dòng 1 (30 tuổi, 60000 thu nhập):

- **Satisfaction = 2** (Trung bình): Khách hàng này có độ hài lòng trung bình, và mô hình dự đoán là 2. Mô hình cũng dự đoán đúng cho trường hợp này.

#### 3. Dòng 2 (35 tuổi, 80000 thu nhập):

- **Satisfaction = 3** (Cao): Khách hàng này có mức độ hài lòng cao là 3, nhưng mô hình chỉ dự đoán là 2. Đây là một trường hợp mà mô hình đã dự đoán thấp hơn thực tế.

#### 4. Dòng 3 (40 tuổi, 70000 thu nhập):

- **Satisfaction = 2**: Mức độ hài lòng thực tế là 2, nhưng mô hình dự đoán 3, dẫn đến một sai số trong dự đoán.

#### 5. Dòng 4 (45 tuổi, 90000 thu nhập):

- **Satisfaction = 3**: Khách hàng này có sự hài lòng cao và mô hình cũng dự đoán chính xác là 3.

#### 6. Dòng 5 (50 tuổi, 120000 thu nhập):

- **Satisfaction = 3**: Độ hài lòng thực tế là 3 và mô hình cũng dự đoán

### 3.3. Quá khớp (Overfitting)

#### Ví dụ cụ thể:

Giả sử bạn đang xây dựng một mô hình hồi quy tuyến tính để dự đoán giá nhà dựa trên kích thước căn hộ (diện tích) và số phòng ngủ. Từ dữ liệu huấn luyện, bạn có các trọng số như sau:

- Kích thước:  $\theta_1 = 1500$
- Số phòng ngủ:  $\theta_2 = 1000$

Nếu bạn chỉ sử dụng hàm mất mát bình thường (Mean Squared Error - MSE), mô hình có thể học quá nhiều từ dữ liệu huấn luyện và tạo ra các trọng số lớn, dẫn đến overfitting.

#### Sử dụng L2 Regularization:

- Giả sử bạn thiết lập  $\lambda = 0.1$ .
- Hàm mất mát có thể được tính như sau:

$$J(\theta) = \text{MSE} + 0.1 \times (\theta_1^2 + \theta_2^2) = \text{MSE} + 0.1 \times (1500^2 + 1000^2)$$

### 3.3. Quá khớp (Overfitting)

- Số phong ngu:  $\theta_2 = 1000$

Nếu bạn chỉ sử dụng hàm mất mát bình thường (Mean Squared Error - MSE), mô hình có thể học quá nhiều từ dữ liệu huấn luyện và tạo ra các trọng số lớn, dẫn đến overfitting.

#### Sử dụng L2 Regularization:

- Giả sử bạn thiết lập  $\lambda = 0.1$ .
- Hàm mất mát có thay đổi được tính như sau:

### 3.4. Chuẩn hóa

- Vector cùng số lượng chiều, không đạt chuẩn
  - Dữ liệu được do bằng các đơn vị khác nhau
  - Hai dữ liệu thành phần có khoảng chênh lệch lớn
- Chuẩn hóa
  - Chuyển khoảng giá trị
  - Chuẩn hóa theo phân phối chuẩn
  - Chuẩn hóa về cùng norm

### 3.4. Chuẩn hóa

- Chuyển khoảng giá trị

$$x'_i = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}$$

- $x_i$  và  $x'_i$  giá trị đặc trưng thứ  $i$  trước và sau khi được chuẩn hóa
- $\max(x_i)$  và  $\min(x_i)$  là giá trị lớn nhất, nhỏ nhất của đặc trưng thứ  $i$  trên toàn bộ dữ liệu huấn luyện

### 3.4. Chuẩn hóa

- Chuẩn hóa theo phân phối chuẩn

$$x'_i = \frac{x_i - \bar{x}_i}{\sigma_i}$$

- $x_i$  và  $x'_i$  giá trị đặc trưng thứ  $i$  trước và sau khi được chuẩn hóa
- $\bar{x}_i$  và  $\sigma_i$  là kỳ vọng và độ lệch chuẩn của đặc trưng thứ  $i$  trên toàn bộ dữ liệu huấn luyện

- Chuẩn hóa về cùng norm (chuẩn)

$$x' = \frac{x}{\|x\|_2}$$

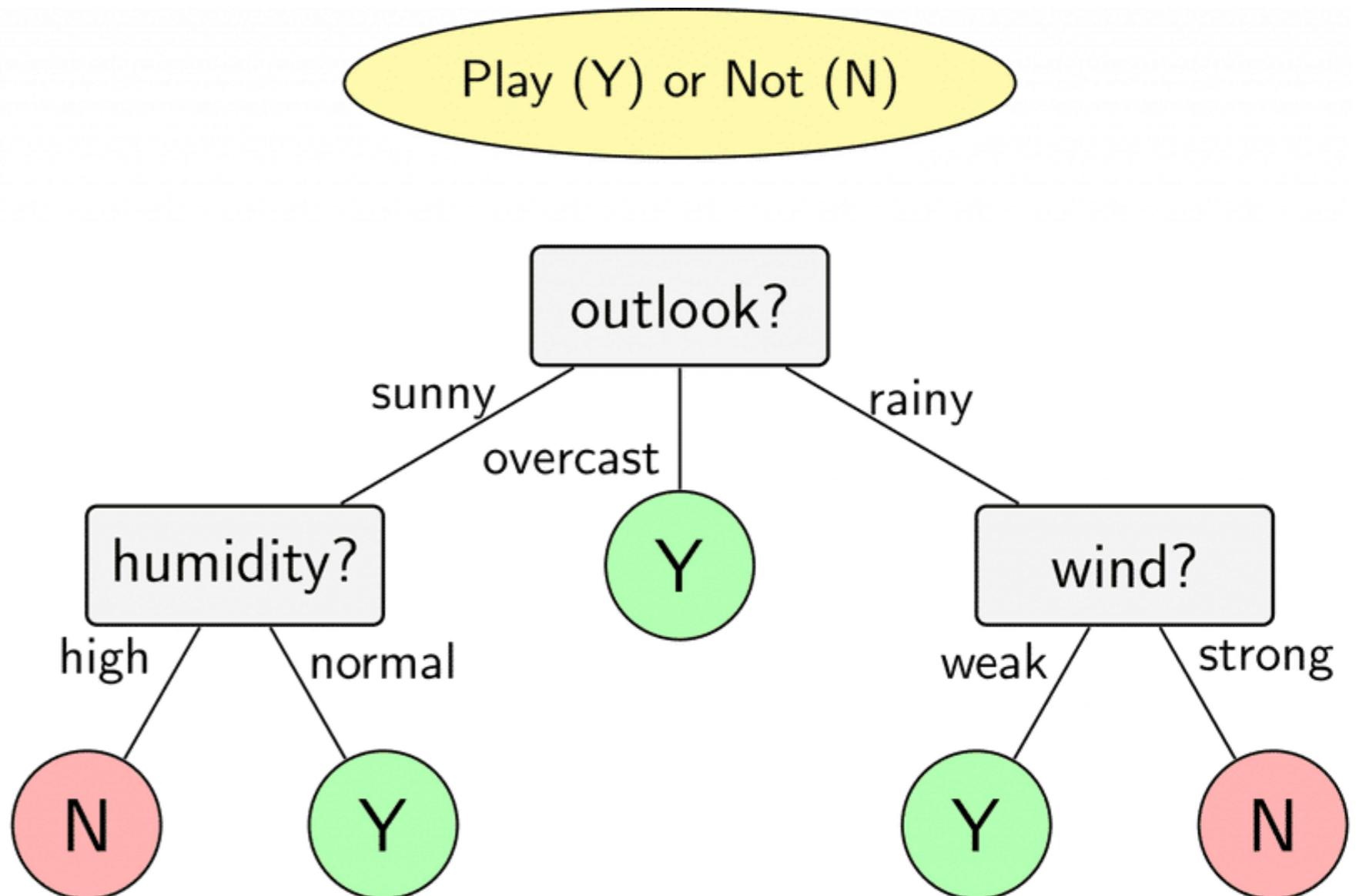
$$\|x\|_2 = (\|x_1\|^p + \|x_2\|^p + \dots + \|x_n\|^p)^{\frac{1}{p}}$$

### 3.5. Học cây quyết định

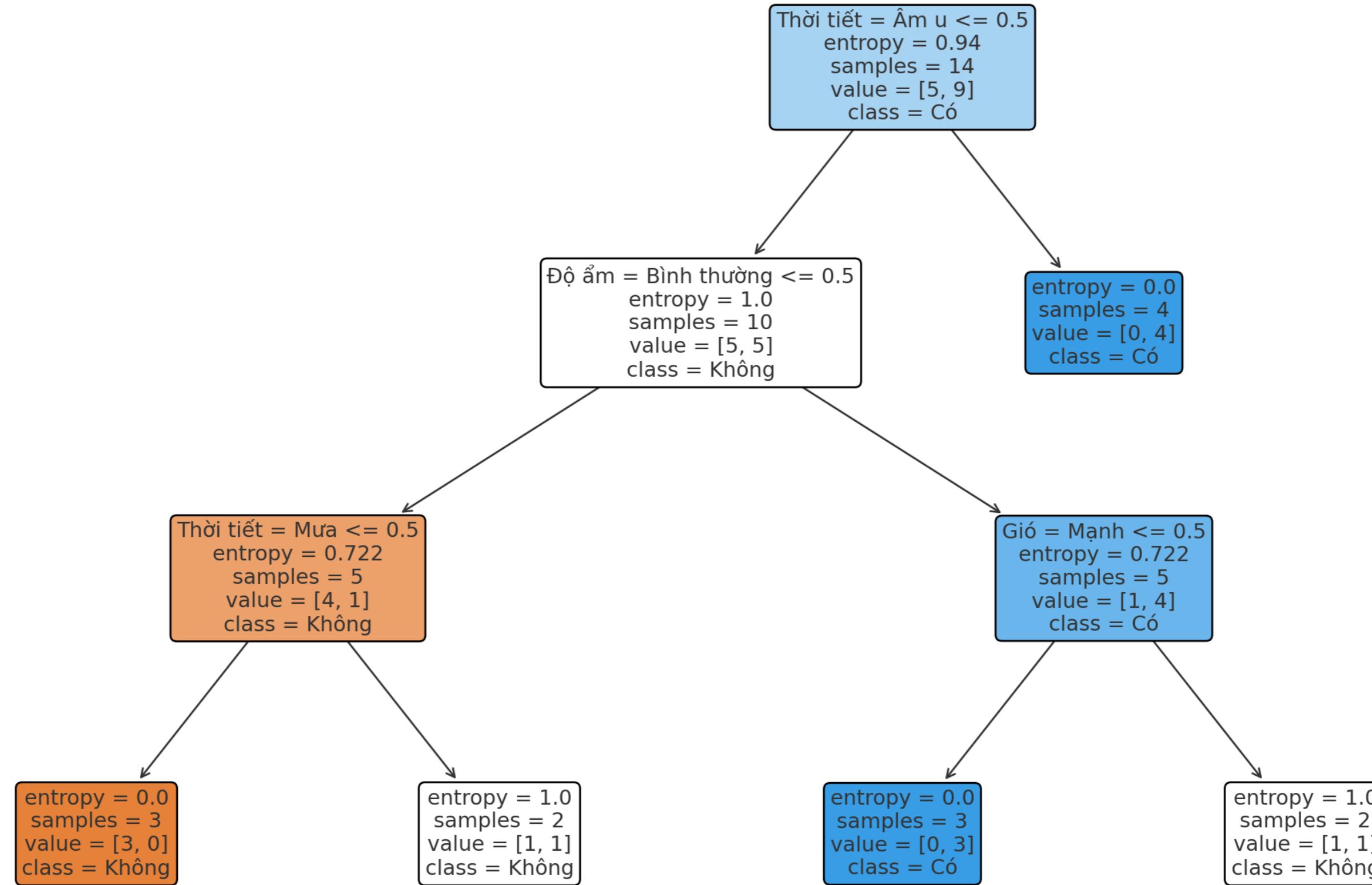
Cây quyết định (Decision Tree) là một mô hình học máy được sử dụng rộng rãi trong các bài toán phân loại (classification) và hồi quy (regression).

Cấu trúc của cây quyết định bao gồm các nút (node) và các nhánh (branch).

Mỗi nút nội bộ (internal node) đại diện cho một quyết định dựa trên một thuộc tính (feature), các nút lá (leaf node) đại diện cho các lớp (class) hoặc giá trị dự đoán (predicted value).



## 3.5. Học cây quyết định



### 3.5. Học cây quyết định

#### Cấu trúc của Ma trận nhầm lẫn (confusion matrix )

Một ma trận nhầm lẫn cho bài toán phân loại nhị phân (2 lớp) có thể được biểu diễn dưới dạng bảng như sau:

	Dự đoán Positive (1)	Dự đoán Negative (0)
Thực tế Positive (1)	True Positive (TP)	False Negative (FN)
Thực tế Negative (0)	False Positive (FP)	True Negative (TN)

- . **True Positive (TP):** Số trường hợp mà mô hình dự đoán đúng là Positive (1).
- . **False Positive (FP):** Số trường hợp mà mô hình dự đoán là Positive (1), nhưng thực tế là Negative (0).
- . **False Negative (FN):** Số trường hợp mà mô hình dự đoán là Negative (0), nhưng thực tế là Positive (1).
- . **True Negative (TN):** Số trường hợp mà mô hình dự đoán đúng là Negative (0).

# Ví dụ về Ma trận nhầm lẫn

.Giả sử chúng ta có một mô hình đã dự đoán kết quả cho 10 trường hợp (0 là không bị bệnh, 1 là bị bệnh):

Thực tế	Dự đoán
1	1
1	1
1	0
0	1
0	0
0	0
1	1
0	0
1	0
0	0

Từ bảng dữ liệu trên, chúng ta có thể tính toán các giá trị trong ma trận nhầm lẫn:

.**TP (True Positive)** = 3 (dự đoán đúng 3 trường hợp bị bệnh)

.**TN (True Negative)** = 4 (dự đoán đúng 4 trường hợp không bị bệnh)

.**FP (False Positive)** = 2 (dự đoán sai 2 trường hợp không bị bệnh thành bị bệnh)

.**FN (False Negative)** = 1 (dự đoán sai 1 trường hợp bị bệnh thành không bị bệnh)

## Ma trận nhầm lẫn:

Dựa trên các giá trị đã tính toán, ma trận nhầm lẫn có thể được biểu diễn như sau:

	Dự đoán Bị bệnh (1)	Dự đoán Không bị bệnh (0)
Thực tế Bị bệnh (1)	3 (TP)	1 (FN)
Thực tế Không bị bệnh (0)	2 (FP)	4 (TN)

## Đánh giá hiệu suất từ Ma trận nhầm lỗ

Từ ma trận nhầm lỗ, bạn có thể tính toán các chỉ số đánh giá hiệu suất của mô hình như sau:

### .Độ chính xác (Accuracy):

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} = \frac{(3+4)}{(3+2+4+1)} = 7/10 = 0.7$$

### .Độ nhạy (Recall) hay còn gọi là Tỷ lệ đúng (True Positive Rate):

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{3}{3+1} = 3/4 = 0.75$$

### .Độ chính xác (Precision):

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{3}{3+2} =$$

## Hình ảnh minh họa

Dưới đây là hình ảnh mô tả ma trận nhầm lỗ cho bài toán phân loại nhị phân:

		Dự đoán	
		1	0
Thực tế	1	TP	FN
	0	FP	TN

### 3.6. Học nhóm (Bagging và Boosting)

Học nhóm (ensemble learning) là một kỹ thuật mạnh mẽ trong học máy (machine learning), trong đó nhiều mô hình (học yếu) được kết hợp để tạo ra một mô hình mạnh mẽ hơn (học mạnh).

Bagging và Boosting là hai phương pháp học nhóm phổ biến và đều nhằm cải thiện độ chính xác của mô hình dự đoán. Dưới đây là định nghĩa, cách hoạt động, ứng dụng và so sánh giữa Bagging và Boosting.

# So Sánh giữa Bagging và Boosting

Tiêu chí	Bagging	Boosting
Phương pháp	Học song song (parallel)	Học tuần tự (sequential)
Mô hình	Kết hợp nhiều mô hình yếu cùng một lúc	Mô hình mới cải thiện mô hình trước đó
Sai số	Giảm thiểu biến động (variance)	Giảm thiểu cả bias và variance
Hiệu suất	Hiệu quả với mô hình có độ biến động cao	Thích hợp cho mô hình có độ thiên lệch cao
Đầu ra	Kết hợp qua trung bình hoặc phiếu	Kết hợp dựa trên trọng số của từng mô hình
Thời gian huấn luyện	Nhanh hơn do huấn luyện song song	Thời gian huấn luyện có thể lâu hơn do tuần tự

### 3.6. Học nhóm (Bagging và Boosting)

- Bagging (Bootstrap Aggregating)

Bagging (Bootstrap Aggregating) là một kỹ thuật học tăng cường (ensemble learning) được giới thiệu bởi Leo Breiman vào năm 1994.

Ví dụ: Thầy cung cấp file

- Boosting

Boosting là một kỹ thuật học máy trong lĩnh vực học sâu, được thiết kế để cải thiện hiệu suất của mô hình dự đoán bằng cách kết hợp nhiều mô hình yếu (weak models) thành một mô hình mạnh (strong model). Mỗi mô hình yếu được đào tạo lần lượt, với trọng tâm vào việc sửa lỗi từ các mô hình trước đó.

Các thuật toán phổ biến: **AdaBoost**, **Gradient Boosting**, **XGBoost**, **LightGBM**.

Ví dụ: File Thầy cung cấp

### **3.7. Máy véc-tơ hỗ trợ (Support Vector Machine - SVM)**

Máy Véc-tơ Hỗ Trợ (SVM) là một thuật toán học máy mạnh mẽ được sử dụng cho các bài toán phân loại và hồi quy.

SVM hoạt động bằng cách tìm kiếm một siêu phẳng (hyperplane) tối ưu trong không gian đặc trưng để phân chia các lớp dữ liệu khác nhau trong mô hình.

Mục tiêu là tối đa hóa khoảng cách giữa các điểm dữ liệu gần nhất của các lớp khác nhau, được gọi là các vectơ hỗ trợ (support vectors)

- Ứng Dụng trong Lĩnh Vực Kinh Doanh**

**1. Phân loại khách hàng:** SVM có thể được sử dụng để phân loại khách hàng thành các nhóm dựa trên hành vi tiêu dùng, mức độ tiêu thụ sản phẩm, hoặc xác định khả năng mua hàng.

**2. Phát hiện gian lận:** SVM có thể giúp phát hiện gian lận trong các giao dịch tài chính bằng cách phân loại các giao dịch là hợp lệ hay gian lận.

**3. Phân tích cảm xúc:** SVM có thể được áp dụng để phân tích cảm xúc trong các phản hồi từ khách hàng hoặc trong các đánh giá sản phẩm.

### 3.7. Máy véc-tơ hỗ trợ (Support Vector Machine - SVM)

*Ví dụ: Thầy cung cấp*

### 3.8. THUẬT TOÁN K-LÂN CẬN GẦN NHẤT

#### ❖ Giới thiệu

- K-NN là thuật toán đi tìm đầu ra của một điểm dữ liệu mới dựa trên thông tin của K điểm dữ liệu gần nhất trong tập huấn luyện
- K lân cận (K-nearest neighbor, K-NN) là một trong những thuật toán **học có giám sát** đơn giản nhất.
  - K-NN gần như không học gì từ dữ liệu huấn luyện mà ghi nhớ lại một cách máy móc toàn bộ dữ liệu đó.
  - Mọi tính toán được thực hiện tại pha kiểm tra. K-NN có thể được áp dụng vào các bài toán phân loại và hồi quy.

### **3.8. THUẬT TOÁN K-LÂN CẬN GẦN NHẤT**

#### **❖ Giới thiệu**

- Ý tưởng cốt lõi**

Với một điểm cần dự đoán  $x^*$ , KNN sẽ:

- đo “độ gần” (khoảng cách) giữa  $x^*$  và mọi điểm trong tập huấn luyện,
- lấy  $K$  điểm gần nhất,
- suy ra nhãn (classification) bằng “bỏ phiếu” đa số, hoặc giá trị (regression) bằng trung bình/ trung vị của  $K$  hàng xóm.

## 3.8. THUẬT TOÁN K-LÂN CẬN GẦN NHẤT

### ❖ Giới thiệu

#### Các bước thuật toán (phân loại)

1. **Chọn K** (số lân cận) — thường chọn số lẻ để giảm hòa.
2. **Chọn thước đo khoảng cách** (phổ biến: Euclid, Manhattan; có thể dùng khoảng cách Cosine, Minkowski...).
3. **Chuẩn hóa dữ liệu (khuyến nghị)** — scale các đặc trưng về cùng thang đo (ví dụ z-score, min-max).
4. **Tính khoảng cách** từ điểm cần dự đoán  $x^*$  tới **tất cả** điểm huấn luyện.
5. **Sắp xếp** các điểm theo khoảng cách tăng dần và **lấy K điểm gần nhất**.
6. **Bỏ phiếu**:
  - Phân loại: chọn lớp xuất hiện nhiều nhất (có thể “bỏ phiếu có trọng số” theo  $1/\text{distance}$ ).
  - Hòa (tie): dùng K nhỏ hơn/lớn hơn, hoặc quy tắc phụ (ưu tiên lớp có tổng trọng số lớn hơn).
7. **Trả về nhãn dự đoán** cho  $x^*$ .
  - Với **hồi quy**, bước 6 là lấy **trung bình có/không trọng số** (hoặc trung vị) giá trị mục tiêu của K hàng xóm.

### **3.8. THUẬT TOÁN K-LÂN CẬN GẦN NHẤT**

- ❖ Hàm tính khoảng cách
- Đặc điểm
  - Đóng vai trò rất quan trọng trong phương pháp học dựa trên các láng giềng gần nhất
  - Thường được xác định trước, và không thay đổi trong suốt quá trình học và phân loại/dự đoán
- Lựa chọn hàm khoảng cách d
  - Các hàm khoảng cách hình học: Dành cho các bài toán có các thuộc tính đầu vào là kiểu số thực ( $x_i \in R$ )
  - Hàm khoảng cách Hamming: Dành cho các bài toán có các thuộc tính đầu vào là kiểu nhị phân ( $x_i \in \{0,1\}$ )
  - Hàm tính độ tương tự Cosine: Dành cho các bài toán phân lớp văn bản ( $x_i$  là giá trị trọng số TF/IDF của từ khóa thứ  $i$ )

### 3.8. THUẬT TOÁN K-LÂN CẬN GẦN NHẤT

❖ Hàm tính khoảng cách

• Hàm khoảng cách hình học

• Hàm Minkowski (p-norm):

$$d(x, z) = \left( \sum_{i=1}^m |x_i - zi|^p \right)^{\frac{1}{p}}$$

• Hàm Manhattan (p=1):

$$d(x, z) = \sum_{i=1}^n |x_i - zi|$$

• Hàm Euclid (p=2):

$$d(x, z) = \sqrt{\sum_{i=1}^n |x_i - zi|^2}$$

• Hàm Chebyshev (p=∞):

$$d(x, z) = \max_i |x_i - zi|$$

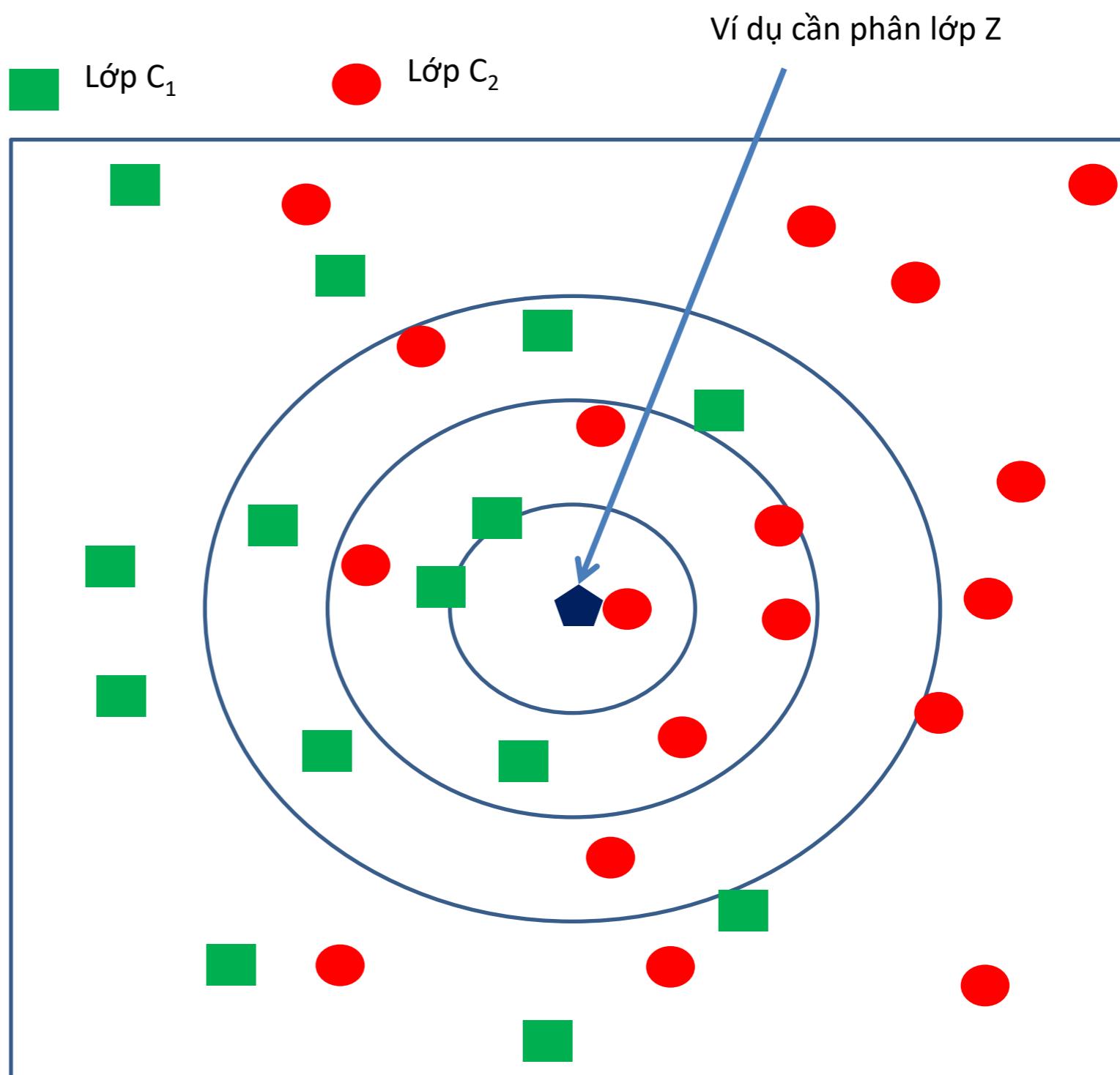
Hàm khoảng cách Hamming

Hàm tính độ tương tự Cosine

## 3.8. THUẬT TOÁN K-LÂN CẬN GẦN NHẤT

### 2.1. Giới thiệu

- Xét 1 lân cận gần
  - Gán Z vào  $C_2$
- Xét 2 lân cận gần
  - Gán Z vào  $C_1$
- Xét 3 lân cận gần
  - Gán Z vào  $C_1$
- Xét 4 lân cận gần
  - Gán Z vào  $C_2$
- 



### 3.9. Gom cụm dữ liệu (Clustering)

Gom cụm (Clustering) là một kỹ thuật học không giám sát, dùng để nhóm các điểm dữ liệu có tính chất tương tự vào cùng một cụm.

Phân tích dữ liệu chưa gán nhãn (unsupervised learning), thường dùng trong phân đoạn khách hàng hoặc phát hiện hành vi bất thường.

Thuật toán phổ biến:

1. **K-Means**: K tìm cụm có trọng tâm (centroid).
2. **DBSCAN**: Tìm cụm dựa trên mật độ dữ liệu.
3. **Hierarchical Clustering**: Gom cụm dựa trên hệ thống phân cấp.

## Ví dụ: Gom cụm bằng K-Means

```
# Dữ liệu đơn giản
data = [[1, 2], [1, 4], [1, 0], [10, 2], [10, 4], [10, 0]]

# Gom cụm thành 2 nhóm
kmeans = KMeans(n_clusters=2, random_state=0)
kmeans.fit(data)
# Kết quả gom cụm
print("Nhãn cụm:", kmeans.labels_)
```

### 3.10. Giảm chiều dữ liệu (Dimensionality Reduction)

Thu giảm chiều dữ liệu là quá trình giảm số lượng đặc trưng (features) trong dữ liệu, nhằm giảm độ phức tạp của mô hình mà vẫn giữ được thông tin quan trọng.

- . Giảm thời gian tính toán và độ phức tạp.
- Trực quan hóa dữ liệu trong không gian 2D hoặc 3D.

Phương pháp phổ biến:

1. **PCA (Principal Component Analysis).**
2. **t-SNE** (đặc biệt dùng để trực quan hóa).

## Ví dụ:

Sử dụng PCA để giảm chiều dữ liệu:

```
from sklearn.decomposition import PCA  
  
# Dữ liệu ví dụ  
X_high_dimensional = [[2, 8, 4], [6, 3, 5], [7, 9, 1]]  
  
# Giảm chiều dữ liệu xuống 2 chiều  
pca = PCA(n_components=2)  
X_reduced = pca.fit_transform(X_high_dimensional)  
# Kết quả gom cụm  
print("Nhãn cụm:", kmeans.labels_)
```

### 3.11. Ứng dụng các mô hình học máy trong xử lý dữ liệu kinh doanh

Ý nghĩa của mô hình học máy trong kinh doanh:

1. **Dự đoán doanh số bán hàng:** Hồi quy tuyến tính/logistic để dự đoán doanh số dựa trên giá cả, mùa vụ.
2. **Phân đoạn khách hàng:** Gom cụm dữ liệu (**Clustering**) để nhóm các khách hàng.
3. **Phát hiện gian lận:** SVM hoặc Boosting để phân loại giao dịch hợp lệ/gian lận.
4. **Đề xuất sản phẩm:** KNN hoặc mô hình dự đoán dựa trên lịch sử mua sắm.
5. **Dự đoán bỏ rơi khách hàng:** Phân tích logistic để dự báo khả năng rời bỏ.

### 3.1 Ví dụ: Dự đoán khách mua đặt lại sản phẩm.

```
# Dữ liệu ví dụ (giả sử có các đặc trưng về hành vi khách hàng) features = [[10, 5], [1, 20],  
[3, 15]] # Mua hàng lần đầu/thời gian sử dụng dịch vụ  
labels = [1, 0, 0] # 1: khách hàng trung thành, 0: không  
  
model = GradientBoostingClassifier() model.fit(features, labels)  
  
print("Dự đoán khách quay lại:", model.predict([[4, 10]]))
```

### **3.1 Ví dụ: Dự đoán khách mua đặt lại sản phẩm.**

- **Bagging và Boosting:** Kết hợp sức mạnh của nhiều mô hình.
- **KNN:** Tìm lớp dựa trên láng giềng gần.
- **Clustering:** Gom cụm dữ liệu chưa gán nhãn.
- **PCA:** Giảm chiều dữ liệu.

**Ứng dụng kinh doanh:** Giải quyết các bài toán thực tế như dự đoán, phân đoạn khách hàng, phát hiện gian lận.

### **3.12. Xu hướng ứng dụng thực tế**

Dưới đây là thông tin chi tiết hơn về **ứng dụng các mô hình học máy trong xử lý dữ liệu kinh doanh** với các **nhược điểm, cách giải quyết các bài toán kinh doanh thực tế**, và các **ví dụ cụ thể**.

#### **1. Xu hướng ứng dụng học máy trong xử lý dữ liệu kinh doanh**

Học máy trong xử lý dữ liệu kinh doanh thường được dùng để:

- **Dự đoán:** Dự báo doanh số, doanh thu, hoặc xu hướng thị trường.
- **Phân đoạn khách hàng:** Nhóm các khách hàng theo các đặc điểm hành vi hoặc nhân khẩu học.
  - **Phát hiện gian lận:** Phân tích hành vi bất thường trong thanh toán, giao dịch.
  - **Đề xuất cá nhân hóa:** Dựa trên sở thích hoặc thói quen tiêu dùng của khách hàng để đề xuất sản phẩm.
  - **Tối ưu hóa tồn kho và chuỗi cung ứng:** Dự đoán lượng hàng tồn kho cần thiết.

## 2. Dự đoán (Prediction Models)

Dựa trên dữ liệu lịch sử, các công ty sử dụng học máy để dự đoán **doanh số bán hàng**, dự đoán khách hàng sắp rời bỏ, hoặc dự đoán nhu cầu sản phẩm.

### Nhược điểm:

1. **Chất lượng dữ liệu thấp**: Dữ liệu có thể thiếu, nhiễu hoặc không đầy đủ (missing values).
2. **Hiểu nhầm các đặc trưng quan trọng**: Việc dùng sai hoặc bỏ qua các biến đầu vào có thể gây dự đoán sai.
3. **Quá khớp**: Một số mô hình quá phức tạp có thể hoạt động tốt trên dữ liệu huấn luyện nhưng kém trên dữ liệu thực.

## **Cách giải quyết:**

1. **Tiền xử lý dữ liệu:** Loại bỏ giá trị nhiễu, xử lý thiếu dữ liệu bằng các phương pháp như trung bình, median.
2. **Sử dụng các mô hình đơn giản trước:** Hồi quy tuyến tính/logistic, sau đó thử nghiệm các mô hình phức tạp hơn (Random Forest, XGBoost).
3. **Kỹ thuật cross-validation:** Kiểm tra mô hình trên các tập dữ liệu khác nhau để tránh quá khớp.

## Ví dụ cụ thể:

**Bài toán:** Dự đoán doanh số bán hàng trong tháng tới.

```
from sklearn.linear_model import LinearRegression  
import numpy as np  
  
# Dữ liệu: chi phí quảng cáo và doanh số bán hàng  
X = np.array([[10], [20], [30], [40], [50]]) # Chi phí quảng cáo ($1000)  
y = np.array([100, 200, 300, 400, 500]) # Doanh số ($1000)  
  
# Huấn luyện mô hình hồi quy tuyến tính  
model = LinearRegression()  
model.fit(X, y)  
  
# Dự đoán: nếu chi phí quảng cáo là 60 ngàn USD  
predicted_sales = model.predict([[60]])  
print("Dự đoán doanh số:", predicted_sales[0])
```

**Nhược điểm:** Nếu chi phí quảng cáo không phải yếu tố chính ảnh hưởng doanh số, mô hình sẽ thất bại.

**Giải quyết:** Kết hợp thêm các đặc trưng như thời gian, khu vực, yếu tố mùa vụ vào mô hình.

### 3. Phân đoạn khách hàng (Customer Segmentation)

#### Mô tả bài toán

Mục tiêu là hiểu và nhóm khách hàng vào các cụm (segments) dựa trên sự tương đồng về hành vi, sở thích, hoặc đặc điểm nhân khẩu học.

#### Nhược điểm:

1. **Khó thiết lập số lượng cụm:** Trong một số trường hợp, không dễ xác định "số cụm" phù hợp cho bài toán.
2. **Dữ liệu chưa có cấu trúc rõ ràng:** Nếu dữ liệu phân mảnh hoặc quá nhiều biến không có ý nghĩa thì mô hình gom cụm sẽ khó nhóm chính xác.
3. **Không gán nhãn được cụm:** Gom cụm thường là bài toán không giám sát, khó hiểu cụm kết quả.

### 3. Phân đoạn khách hàng (Customer Segmentation)

Cách giải quyết:

1. **Chọn số lượng cụm bằng phương pháp Elbow:** Tìm điểm "gãy" trong đồ thị wcss (Within Cluster Sum of Squares) để chọn số cụm tối ưu.
2. **Tiền xử lý để chuẩn hóa dữ liệu:** Giảm chiều dữ liệu hoặc xử lý outliers để tăng tính hiệu quả của K-Means.
3. **Phân tích ý nghĩa cụm dữ liệu:** Sau khi tạo cụm, cần phân tích sâu để hiểu rõ ý nghĩa từng cụm.

## Ví dụ cụ thể:

**Bài toán:** Gom cụm khách hàng theo mức chi tiêu và tần suất mua hàng.

```
from sklearn.cluster import KMeans
```

```
import numpy as np
```

```
# Dữ liệu: [tần suất mua hàng, tổng mức chi tiêu ($1000)]
```

```
data = np.array([[2, 10], [3, 15], [5, 40], [8, 80], [11, 100], [13, 120]])
```

```
# Gom cụm thành 2 nhóm
```

```
kmeans = KMeans(n_clusters=2, random_state=0)
```

```
kmeans.fit(data)
```

```
# Kết quả
```

```
print("Nhãn cụm:", kmeans.labels_)
```

**Nhược điểm:** Kết quả gom cụm phụ thuộc vào cách chuẩn hóa dữ liệu.

**Giải quyết:** Chuẩn hóa dữ liệu trước khi gom cụm bằng StandardScaler.

## 4. Phát hiện gian lận (Fraud Detection)

### Mô tả bài toán

Phát hiện các giao dịch bất thường trong thanh toán hoặc gian lận như sử dụng thẻ tín dụng trái phép, gian lận bảo hiểm, hoặc đặt lệnh spam.

### Nhược điểm:

1. **Số lượng gian lận thường rất ít (data imbalance)**: Các giao dịch hợp lệ thường chiếm đa số, gây mất cân bằng dữ liệu.
2. **Xử lý giả dương (False Positive)**: Hệ thống có thể báo nhầm giao dịch hợp lệ là gian lận, làm gián đoạn khách hàng.

**Ví dụ cụ thể: Bài toán:** Phát hiện giao dịch gian lận.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import numpy as np

# Dữ liệu: [số tiền giao dịch ($), thời gian thực hiện giao dịch (giờ)]
X = np.array([[100, 2], [200, 5], [800, 23], [50, 9], [4000, 3]]) # Đặc trưng
y = [0, 0, 1, 0, 1] # 1: gian lận, 0: hợp lệ

# Chia tập huấn luyện và kiểm thử
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Huấn luyện mô hình Random Forest
model = RandomForestClassifier() model.fit(X_train, y_train)

# Kiểm tra giao dịch mới
print("Dự đoán giao dịch:", model.predict([[1000, 3]]))
```

**Nhược điểm:** Nếu không xử lý mất cân bằng, mô hình có thể bỏ sót các giao dịch gian lận (false negatives).

**Giải quyết:** Cân bằng dữ liệu và chất lượng đặc trưng đầu vào.

## 5. Tóm lại - Nhược điểm chung và hướng giải quyết

Bài toán	Nhược điểm chính	Cách giải quyết
Dự đoán	Dữ liệu thiếu, quá khớp	Xử lý dữ liệu trước crossvalidation, kết hợp nhiều đặc trưng.
Phân đoạn khách hàng	Không rõ số cụm, khó phân tích ý nghĩa	Sử dụng phương pháp chọn số cụm (Elbow), phân tích cụm.
Phát hiện gian lận	Mất cân bằng dữ liệu, hành vi thay đổi	SMOTE, theo dõi mô hình liên tục, kết hợp nhiều giải thuật.

## 5. Tóm lại - Nhược điểm chung và hướng giải quyết

Như vậy, điểm cốt lõi để giải quyết các nhược điểm của học máy trong kinh doanh là **tiền xử lý dữ liệu, cân nhắc chọn thuật toán phù hợp, và theo dõi cập nhật mô hình** theo thời gian. Các bài toán trên đều được cải thiện bằng cách **kết hợp con người** (manual supervision) trong việc phân tích kết quả.