Bài tập Machine learning

TS. Trần Quang Quý - Bộ môn Khoa học Máy tính

ĐT: 0818981166; Email: tqquy@ictu.edu.vn

Mô hình hồi quy tuyến tính

Khái niệm cơ bản

Mô hình hồi quy tuyến tính là một phương pháp thống kê dùng để dự đoán giá trị của một biến phụ thuộc (biến kết quả) dựa trên một hoặc nhiều biến độc lập (biến dự đoán). Phương trình hồi quy tuyến tính đơn giản có dạng:

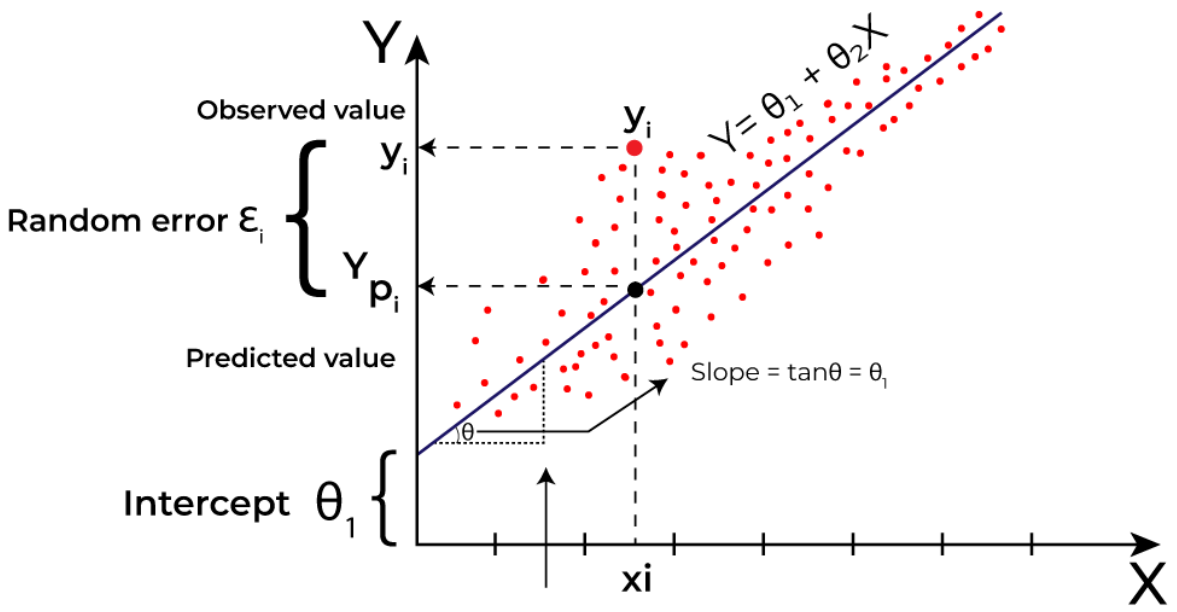
$$y = \beta_0 + \beta_1 x + \epsilon$$

- y: Biến phụ thuộc.
- x: Biến độc lập.
- β_0: Hệ số chặn (intercept).
- β_1: Hệ số góc (slope).
- ϵ: Sai số ngẫu nhiên.

Trong mô hình này, mục tiêu là tìm các hệ số β_0 và β_1 sao cho sai số giữa giá trị thực tế và giá trị dự đoán là nhỏ nhất, thường thông qua phương pháp bình phương tối thiểu (least squares).

Các bước thực hiện:

1. **Thu thập dữ liệu**: Tập hợp dữ liệu với các giá trị x và y.
2. **Ước lượng tham số**: Sử dụng phương pháp bình phương tối thiểu để tính toán các hệ số β_0 và β_1.
3. **Đánh giá mô hình**: Sử dụng các chỉ số như RMSE và R^2 để đánh giá độ chính xác của mô hình.
4. **Dự đoán**: Sử dụng phương trình hồi quy để dự đoán giá trị y cho các giá trị x mới.



Ước lượng tham số cho mô hình hồi quy tuyến tính

Để ước lượng các tham số của mô hình hồi quy tuyến tính, bạn có thể sử dụng phương pháp bình phương tối thiểu (Least Squares). Dưới đây là các bước chi tiết:

1. **Phương trình hồi quy**:

$$y = \beta_0 + \beta_1 x$$

2. **Mục tiêu**:
   Tìm các giá trị của β_0 và β_1 sao cho tổng bình phương sai số (SSE) là nhỏ nhất:

$$SSE = \sum (y_i - (\beta_0 + \beta_1 x_i))^2$$

3. **Công thức tính**:

   - Hệ số góc (β_1):

$$\beta_1 = \frac{n(\sum x_i y_i) - (\sum x_i)(\sum y_i)}{n(\sum x_i^2) - (\sum x_i)^2}$$

   - Hệ số chặn (β_0):

$$\beta_0 = \frac{\sum y_i - \beta_1(\sum x_i)}{n}$$

   - n là số lượng điểm dữ liệu.

4. **Tính toán**:

   - Tính tổng $\sum x_i$, $\sum y_i$, $\sum x_i y_i$, và $\sum x_i^2$.
   - Thay các giá trị vào công thức để tìm β_0 và β_1.

5. **Sử dụng các tham số**:
   Dùng β_0 và β_1 để xây dựng phương trình hồi quy tuyến tính và dự đoán giá trị y từ các giá trị x.

Ví dụ cụ thể:

Giả sử bạn có dữ liệu:

x	y
1	2
2	3
3	4
4	5

Tính các tổng:

- $\sum x_i = 1 + 2 + 3 + 4 = 10$
- $\sum y_i = 2 + 3 + 4 + 5 = 14$
- $\sum x_i y_i = 1 * 2 + 2 * 3 + 3 * 4 + 4 * 5 = 40$
- $\sum x_i^2 = 1^2 + 2^2 + 3^2 + 4^2 = 30$

Áp dụng vào công thức, bạn sẽ tìm được β_1 và β_0.

Trong trường hợp mô hình hồi quy có nhiều tham số (hồi quy tuyến tính bội), bạn có thể sử dụng phương pháp bình phương tối thiểu mở rộng. Dưới đây là các bước:

1. **Phương trình hồi quy**:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

2. **Ma trận và vector**:
   - Biến phụ thuộc y được biểu diễn dưới dạng vector.
   - Biến độc lập X được biểu diễn dưới dạng ma trận, với mỗi hàng là một mẫu dữ liệu và mỗi cột là một biến.

3. **Công thức ước lượng tham số**:

$$\beta = (X^T X)^{-1} X^T y$$

   - X^T là ma trận chuyển vị của X.
   - $(X^T X)^{-1}$ là ma trận nghịch đảo của $X^T X$.

4. **Cách tính toán**:
   - Xây dựng ma trận X với cột đầu tiên là các giá trị 1 (để ước lượng β_0).
   - Sử dụng công thức trên để tính toán vector hệ số β.

Cách tính chỉ số RMSE và R bình phương

Chỉ số RMSE

Để tính chỉ số RMSE (Root Mean Square Error), bạn làm theo các bước sau:

1. **Tính sai số**: Tính sai số giữa giá trị thực tế y_i và giá trị dự đoán $\hat{y}_i$ cho từng điểm dữ liệu.

$$e_i = y_i - \hat{y}_i$$

2. **Bình phương sai số**: Tính bình phương của từng sai số.

$$e_i^2 = (y_i - \hat{y}_i)^2$$

3. **Tính trung bình sai số bình phương**: Tính trung bình của tất cả các sai số bình phương.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} e_i^2$$

   (n là số lượng điểm dữ liệu)

4. **Lấy căn bậc hai của trung bình sai số bình phương**:

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Ví dụ:

Giả sử bạn có dữ liệu thực tế và dự đoán như sau:

y (Thực tế)	$\hat{y}$ (Dự đoán)
2	2.2
3	2.8
4	3.9
5	5.1

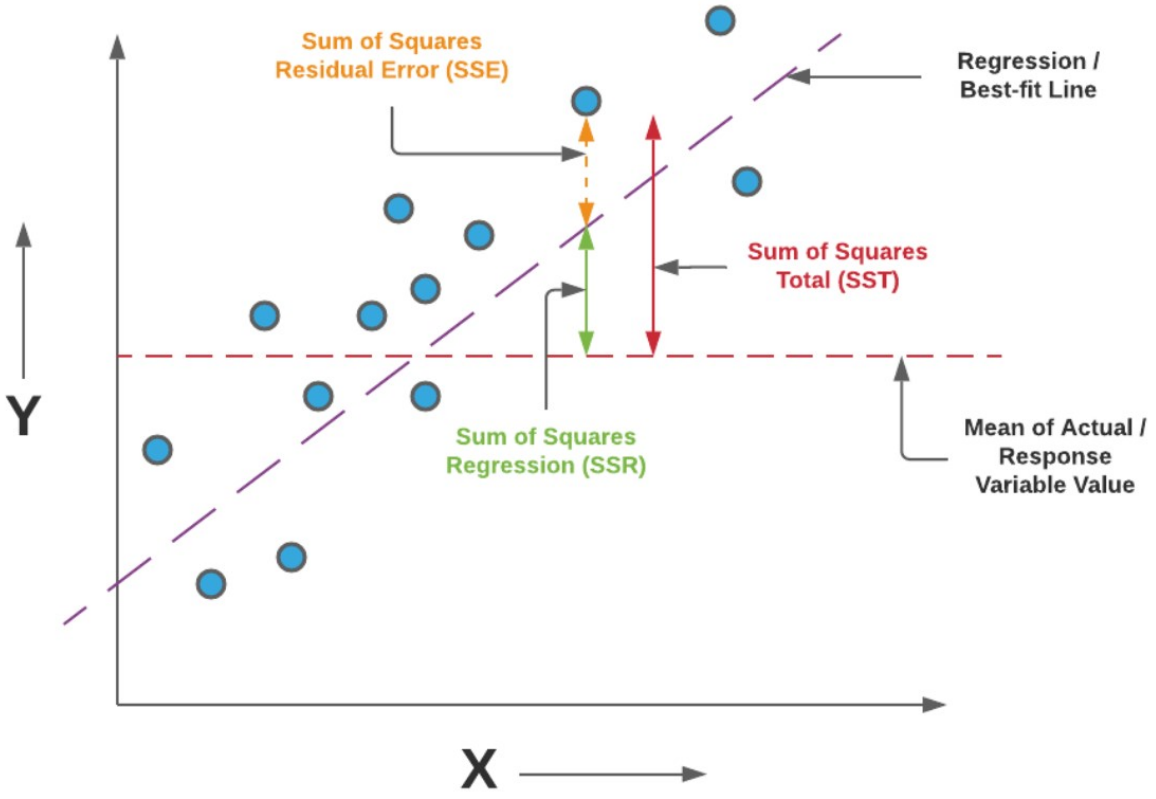
1. Tính sai số:
   - $e_1 = 2 - 2.2 = -0.2$
   - $e_2 = 3 - 2.8 = 0.2$
   - $e_3 = 4 - 3.9 = 0.1$
   - $e_4 = 5 - 5.1 = -0.1$
2. Tính bình phương sai số:
   - $e_1^2 = 0.04$
   - $e_2^2 = 0.04$
   - $e_3^2 = 0.01$
   - $e_4^2 = 0.01$
3. Tính MSE:

$$\text{MSE} = \frac{0.04 + 0.04 + 0.01 + 0.01}{4} = 0.025$$

4. Tính RMSE:

$$\text{RMSE} = \sqrt{0.025} \approx 0.158$$

Chỉ số R bình phương


Sum of Squares
Residual Error (SSE)
Regression /
Best-fit Line
Sum of Squares
Total (SST)
Y
Sum of Squares
Regression (SSR)
Mean of Actual /
Response
Variable Value
X

Để tính chỉ số R^2 (R bình phương), bạn thực hiện các bước sau:

1. **Tính giá trị trung bình của y:**

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

2. **Tính tổng sai số bình phương toàn phần (SST):**

$$SST = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

3. **Tính tổng sai số bình phương của phần còn lại (SSE):**

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

4. **Tính tổng sai số bình phương dự đoán (SSR):**

$$SSR = SST - SSE$$

5. **Tính chỉ số R^2:**

$$R^2 = 1 - \frac{SSE}{SST}$$

Ví dụ:

Giả sử bạn có dữ liệu thực tế và dự đoán như sau:

y (Thực tế)	$\hat{y}$ (Dự đoán)
2	2.2
3	2.8
4	3.9
5	5.1

1. **Tính $\bar{y}$:**

$$\bar{y} = \frac{2+3+4+5}{4} = 3.5$$

2. **Tính SST:**

$$SST = (2-3.5)^2 + (3-3.5)^2 + (4-3.5)^2 + (5-3.5)^2 = 2.25 + 0.25 + 0.25 + 2.2$$

3. **Tính SSE:**

$$SSE = (2-2.2)^2 + (3-2.8)^2 + (4-3.9)^2 + (5-5.1)^2 = 0.04 + 0.04 + 0.01 + 0.($$

4. **Tính R^2:**

$$R^2 = 1 - \frac{0.10}{5.00} = 0.98$$

Vậy, chỉ số R^2 là 0.98, cho thấy mô hình dự đoán khá tốt.

Bài tập ví dụ

Giả sử bạn có dữ liệu thực tế và dự đoán như sau:

x	y (Thực tế)	y' (Dự đoán)
1	2	2.2
2	3	2.8
3	4	3.9
4	5	5.1

Tính RMSE

1. **Tính sai số bình phương** cho mỗi cặp giá trị:
   - $(2 - 2.2)^2 = 0.04$
   - $(3 - 2.8)^2 = 0.04$
   - $(4 - 3.9)^2 = 0.01$
   - $(5 - 5.1)^2 = 0.01$
2. **Tính tổng sai số bình phương**:
   - Tổng = $0.04 + 0.04 + 0.01 + 0.01 = 0.10$

3. **Tính trung bình sai số bình phương:**
   - Trung bình = $\frac{0.10}{4} = 0.025$
4. **Lấy căn bậc hai của trung bình sai số bình phương:**
   - RMSE = $\sqrt{0.025} \approx 0.158$

Tính hệ số R^2

1. **Tính giá trị trung bình của y thực tế:**
   - $\bar{y} = \frac{2+3+4+5}{4} = 3.5$
2. **Tính tổng sai số bình phương dự đoán** (SSR):
   - SSR = $(2.2 - 3.5)^2 + (2.8 - 3.5)^2 + (3.9 - 3.5)^2 + (5.1 - 3.5)^2 = 1.69 + 0.49 + 0.16 + 2.56 = 4.90$
3. **Tính tổng sai số bình phương toàn phần** (SST):
   - SST = $(2 - 3.5)^2 + (3 - 3.5)^2 + (4 - 3.5)^2 + (5 - 3.5)^2 = 2.25 + 0.25 + 0.25 + 2.25 = 5.00$
4. **Tính tổng sai số bình phương còn lại** (SSE):
   - SSE = $(2 - 2.2)^2 + (3 - 2.8)^2 + (4 - 3.9)^2 + (5 - 5.1)^2 = 0.10$
5. **Tính hệ số R^2:**
   - $R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{0.10}{5.00} = 0.98$

Như vậy, trong ví dụ này, RMSE xấp xỉ 0.158 và hệ số R^2 là 0.98.

Bài tập luyện tập

Bài 1:

Giả sử bạn có dữ liệu thực tế và dự đoán như sau:

y (Thực tế)	$\hat{y}$ (Dự đoán)
3	2.9
4	4.1
5	5.2
6	5.8

Yêu cầu: Tính RMSE và chỉ số R^2.

Bài 2:

Giả sử bạn có dữ liệu thực tế và dự đoán với nhiều biến như sau:

y (Thực tế)	$\hat{y}$ (Dự đoán)
7	6.8
5	5.2
9	8.5
6	6.3
8	7.7
10	9.9

Yêu cầu: Tính RMSE và chỉ số R^2.

Thực hành code trên Python

Bài tập dự đoán giá nhà, giả sử chúng ta có tập dữ liệu như sau:

Số phòng ngủ	Diện tích đất (m^2)	Khoảng cách đến trung tâm (km)	Giá nhà (tỷ VNĐ)
3	150	5	2.5
4	200	3	3.2
2	120	7	1.8
3	180	4	2.9
5	250	2	4.5
4	210	3.5	3.7
3	160	6	2.3
2	130	8	1.6
4	190	4.5	3.1
3	170	5.5	2.7

Trong đó:

- **Số phòng ngủ**: Số lượng phòng ngủ trong từng ngôi nhà.
- **Diện tích đất**: Diện tích đất của ngôi nhà tính bằng mét vuông.
- **Khoảng cách đến trung tâm thành phố**: Khoảng cách từ ngôi nhà đến trung tâm thành phố tính bằng kilômét.
- **Giá nhà**: Giá của từng ngôi nhà trong đơn vị tỷ VNĐ.

Để giải quyết bài toán dự đoán giá nhà dựa trên số phòng ngủ, diện tích đất và khoảng cách đến trung tâm thành phố sử dụng mô hình hồi quy tuyến tính trong scikit-learn như sau:

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Dữ liệu mẫu
X = np.array([
    [3, 150, 5],    # Ngôi nhà 1: 3 phòng ngủ, diện tích đất 1
    [4, 200, 3],    # Ngôi nhà 2: 4 phòng ngủ, diện tích đất 2
    [2, 120, 7],    # Ngôi nhà 3: 2 phòng ngủ, diện tích đất 1
    [3, 180, 4],    # Ngôi nhà 4: 3 phòng ngủ, diện tích đất 1
    [5, 250, 2],    # Ngôi nhà 5: 5 phòng ngủ, diện tích đất 2
    [4, 210, 3.5],  # Ngôi nhà 6: 4 phòng ngủ, diện tích đất 2
```

```
    [3, 160, 6],    # Ngôi nhà 7: 3 phòng ngủ, diện tích đất :
    [2, 130, 8],    # Ngôi nhà 8: 2 phòng ngủ, diện tích đất :
    [4, 190, 4.5],  # Ngôi nhà 9: 4 phòng ngủ, diện tích đất :
    [3, 170, 5.5]   # Ngôi nhà 10: 3 phòng ngủ, diện tích đất
])

y = np.array([2.5, 3.2, 1.8, 2.9, 4.5, 3.7, 2.3, 1.6, 3.1, 2.

# Tạo mô hình hồi quy tuyến tính
model = LinearRegression()

# Huấn luyện mô hình
model.fit(X, y)

# Dự đoán giá nhà
y_pred = model.predict(X)

# Tính toán RMSE và R^2
rmse = np.sqrt(mean_squared_error(y, y_pred))
r2 = r2_score(y, y_pred)

print("Hệ số hồi quy:", model.coef_)
print("Hệ số chặn:", model.intercept_)
print("RMSE:", rmse)
print("R^2:", r2)
```

Phương pháp K-NN (K Lân cận)

Ví dụ mở đầu

K-nearest neighbors là thuật toán học máy có giám sát, đơn giản và dễ triển khai. Thường được dùng trong các bài toán phân loại và hồi quy.

Bài toán đặt ra: Bạn có điểm của một môn học nhưng bạn không biết thuộc loại nào (Giỏi, khá, trung bình, yếu). Giả sử bạn không biết bất kì quy tắc nào để phân loại cả.

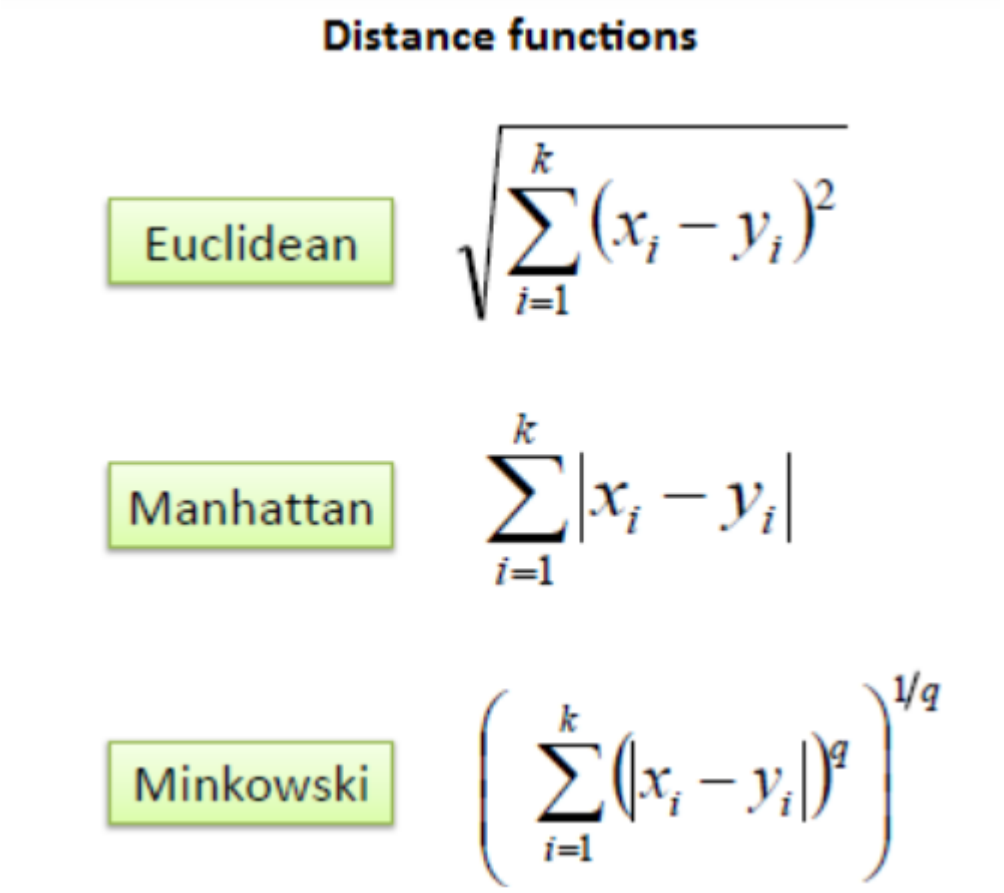
Điểm của tôi: 7

Điểm của bạn tôi:

- 7.1 ⇒ Khá
- 7.2 ⇒ Khá
- 6.7 ⇒ Khá
- 6.6 ⇒ Khá
- 6.4 ⇒ Trung bình

Ý tưởng về khoảng cách

Thuật toán KNN cho rằng những dữ liệu tương tự nhau sẽ tồn tại **gần nhau** trong một không gian, từ đó công việc của chúng ta là sẽ tìm k điểm gần với dữ liệu cần kiểm tra nhất. Việc tìm khoảng cách giữa 2 điểm củng có nhiều công thức có thể sử dụng, tùy trường hợp mà chúng ta lựa chọn cho phù hợp. Đây là 3 cách cơ bản để tính khoảng cách 2 điểm dữ liệu x, y có k thuộc tính:



Khái niệm cơ bản

1. **K-Near Neighbors (K-Láng giềng gần nhất)**: KNN hoạt động dựa trên nguyên tắc tìm kiếm k-láng giềng gần nhất (k điểm dữ liệu gần nhất) trong không gian đặc trưng của dữ liệu đã biết.
2. **Khoảng cách**: KNN thường sử dụng các loại khoảng cách như khoảng cách Euclidean, khoảng cách Manhattan, hoặc khoảng cách Minkowski để đo lường sự gần gũi giữa các điểm dữ liệu.

Cách hoạt động của KNN

1. **Huấn luyện**: Thuật toán KNN không thực sự có giai đoạn huấn luyện. Thay vào đó, nó chỉ lưu trữ toàn bộ tập dữ liệu huấn luyện.
2. **Dự đoán**:
   - **Phân loại**: Đối với một điểm dữ liệu mới, KNN tìm k điểm dữ liệu gần nhất trong tập huấn luyện. Lớp của điểm mới được quyết định bằng cách bỏ phiếu đa số từ các lớp của k láng giềng.
   - **Hồi quy**: Đối với một điểm dữ liệu mới, giá trị của điểm mới được dự đoán bằng cách lấy trung bình các giá trị của k láng giềng gần nhất.

Các bước chi tiết:

1. **Chọn số lượng k láng giềng (k)**: Chọn giá trị của k. Giá trị nhỏ của k có thể bị ảnh hưởng bởi nhiễu, trong khi giá trị lớn của k có thể làm mất chi tiết của dữ liệu.
2. **Tính toán khoảng cách**: Tính khoảng cách giữa điểm dữ liệu mới và tất cả các điểm dữ liệu trong tập huấn luyện.
3. **Tìm k láng giềng gần nhất**: Sắp xếp tất cả các khoảng cách và chọn k điểm có khoảng cách nhỏ nhất.
4. **Dự đoán kết quả**:
   - **Phân loại**: Dự đoán lớp bằng cách lấy lớp chiếm đa số trong k láng giềng.
   - **Hồi quy**: Dự đoán giá trị bằng cách lấy trung bình giá trị của k láng giềng.

Ưu điểm:

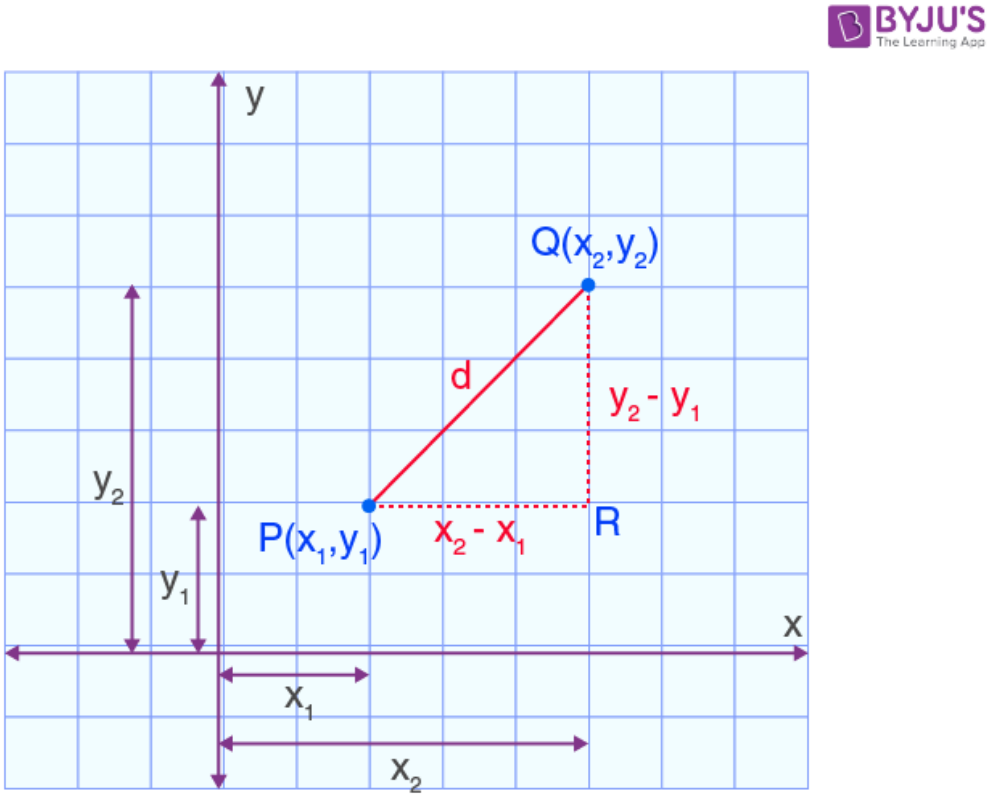
- **Đơn giản và dễ hiểu**.

- **Không yêu cầu huấn luyện**.
- **Hiệu quả với tập dữ liệu nhỏ**.

Nhược điểm:

- **Hiệu suất chậm** với tập dữ liệu lớn vì cần tính toán khoảng cách cho tất cả các điểm dữ liệu.
- **Nhạy cảm với tỷ lệ các đặc trưng**: Các đặc trưng cần được chuẩn hóa để tránh đặc trưng có thang đo lớn hơn chi phối kết quả.

Khoảng cách Euclid (Euclidean distance)



Khoảng cách Euclid là một cách đo lường khoảng cách giữa hai điểm trong không gian Euclid, và nó thường được sử dụng trong các thuật toán học máy, đặc biệt là trong K-Nearest Neighbors (KNN). Khoảng cách này được tính bằng công thức Pythagoras.

Công thức:

Nếu bạn có hai điểm $A(x_1, y_1)$ và $B(x_2, y_2)$ trong không gian 2D, khoảng cách Euclid giữa chúng được tính bằng:

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Trong không gian n-chiều, nếu bạn có hai điểm $A(x_1, x_2, \ldots, x_n)$ và $B(y_1, y_2, \ldots, y_n)$, công thức tổng quát cho khoảng cách Euclid là:

$$d(A, B) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

Ví dụ:

Giả sử bạn có hai điểm trong không gian 3D:

- Điểm A: (1, 2, 3)
- Điểm B: (4, 6, 8)

Khoảng cách Euclid giữa A và B sẽ là:

$$d(A, B) = \sqrt{(4 - 1)^2 + (6 - 2)^2 + (8 - 3)^2}$$
$$d(A, B) = \sqrt{3^2 + 4^2 + 5^2}$$
$$d(A, B) = \sqrt{9 + 16 + 25}$$
$$d(A, B) = \sqrt{50}$$
$$d(A, B) = 7.07$$

Code Python để tính khoảng cách Euclid

```
import math
```

```
# Hàm tính khoảng cách Euclid
def euclidean_distance(point1, point2):
    distance = 0
    for i in range(len(point1)):
        distance += (point1[i] - point2[i]) ** 2
    return math.sqrt(distance)

# Ví dụ với hai điểm 3D
pointA = [1, 2, 3]
pointB = [4, 6, 8]

# Tính khoảng cách
distance = euclidean_distance(pointA, pointB)
print(f"Khoảng cách Euclid giữa điểm A và điểm B là: {distanc
```

```
Khoảng cách Euclid giữa điểm A và điểm B là: 7.07106781186547
```

Khoảng cách Manhattan (Manhattan distance)

Khoảng cách Manhattan, còn được gọi là khoảng cách L1 hoặc khoảng cách City Block, là một cách đo lường khoảng cách giữa hai điểm trong không gian. Thay vì đo lường theo đường thẳng như khoảng cách Euclid, khoảng cách Manhattan đo lường tổng khoảng cách theo các trục tọa độ.

Công thức:

Nếu bạn có hai điểm $A(x_1, y_1)$ và $B(x_2, y_2)$ trong không gian 2D, khoảng cách Manhattan giữa chúng được tính bằng:

$$d(A, B) = |x_2 - x_1| + |y_2 - y_1|$$

Trong không gian n-chiều, nếu bạn có hai điểm $A(x_1, x_2, \ldots, x_n)$ và $B(y_1, y_2, \ldots, y_n)$, công thức tổng quát cho khoảng cách Manhattan là:

$$d(A, B) = \sum_{i=1}^{n} |y_i - x_i|$$

Ví dụ:

Giả sử bạn có hai điểm trong không gian 3D:

- Điểm A: (1, 2, 3)
- Điểm B: (4, 6, 8)

Khoảng cách Manhattan giữa A và B sẽ là:

$$d(A,B) = |4-1| + |6-2| + |8-3|$$
$$d(A,B) = 3 + 4 + 5$$
$$d(A,B) = 12$$

Code Python

```
# Hàm tính khoảng cách Manhattan
def manhattan_distance(point1, point2):
    distance = 0
    for i in range(len(point1)):
        distance += abs(point1[i] - point2[i])
    return distance

# Ví dụ với hai điểm 3D
pointA = [1, 2, 3]
pointB = [4, 6, 8]

# Tính khoảng cách
distance = manhattan_distance(pointA, pointB)
print(f"Khoảng cách Manhattan giữa điểm A và điểm B là: {dista
```

```
Khoảng cách Manhattan giữa điểm A và điểm B là: 12
```

Ứng dụng:

Khoảng cách Manhattan thường được sử dụng trong các bài toán mà đường đi không phải là đường thẳng mà phải đi qua các đoạn đường vuông góc, chẳng hạn như trong các bài toán đường phố trong thành phố (như tên gọi "Manhattan" ám chỉ cấu trúc lưới của các con đường ở Manhattan, New York). Nó cũng được sử dụng trong các ứng dụng học máy khi các đặc trưng của dữ liệu không có mối quan hệ tuyến tính.

Khám phá dữ liệu EDA trong Học máy

Phương pháp khám phá dữ liệu EDA (Exploratory Data Analysis) là một quá trình quan trọng trong phân tích dữ liệu, được thực hiện để hiểu rõ hơn về cấu trúc của dữ liệu, phát hiện các mẫu (patterns), xác định các đặc điểm nổi bật, và kiểm tra các giả thuyết. Các bước chính trong EDA bao gồm:

1. **Tóm tắt dữ liệu**:
    - Sử dụng các thống kê mô tả như trung bình, trung vị, độ lệch chuẩn, giá trị tối đa, tối thiểu, và các giá trị phần tư.
    - Xem xét phân phối của dữ liệu để hiểu rõ hơn về sự phân bố và tính chất của dữ liệu.
2. **Trực quan hóa dữ liệu**:
    - Sử dụng biểu đồ như biểu đồ cột, biểu đồ phân tán, biểu đồ hộp, và biểu đồ histograms để trực quan hóa mối quan hệ giữa các biến số.
    - Biểu đồ heatmap để khám phá tương quan giữa các biến.
3. **Phát hiện giá trị ngoại lệ và thiếu hụt**:
    - Xác định và xử lý các giá trị ngoại lệ (outliers) và các giá trị thiếu hụt (missing values).
4. **Kiểm tra giả thuyết**:
    - Thực hiện các kiểm định thống kê để kiểm tra các giả thuyết ban đầu về dữ liệu.
5. **Biến đổi dữ liệu**:

- Thực hiện các phép biến đổi dữ liệu như chuẩn hóa, chuẩn trực, hoặc mã hóa các biến không phải số (categorical variables) để chuẩn bị dữ liệu cho các bước phân tích sâu hơn.

Ví dụ minh họa đơn giản cho EDA với dữ liệu giá nhà:

1. **Tóm tắt dữ liệu**:
   - Tính toán giá trị trung bình, trung vị, độ lệch chuẩn của giá nhà.
2. **Trực quan hóa dữ liệu**:
   - Vẽ biểu đồ phân tán giữa giá nhà và diện tích đất để xem mối quan hệ giữa hai biến này.
3. **Phát hiện giá trị ngoại lệ**:
   - Sử dụng biểu đồ hộp để phát hiện các giá trị ngoại lệ trong giá nhà.
4. **Kiểm tra giả thuyết**:
   - Kiểm tra xem có sự khác biệt đáng kể về giá nhà giữa các khu vực khác nhau không.
5. **Biến đổi dữ liệu**:
   - Chuẩn hóa giá nhà và diện tích đất để dữ liệu có cùng đơn vị chuẩn trước khi thực hiện các phân tích tiếp theo.

Các công cụ phổ biến hỗ trợ cho EDA bao gồm Python (với các thư viện Pandas, Matplotlib, Seaborn), R (với các gói ggplot2, dplyr), và các phần mềm phân tích dữ liệu như Excel, Tableau.

Tập dữ liệu Iris Dataset

Tập dữ liệu Iris là một trong những tập dữ liệu nổi tiếng nhất trong học máy và thống kê. Nó thường được sử dụng để minh họa các kỹ thuật phân tích dữ liệu và học máy cơ bản. Dưới đây là một số thông tin chi tiết về tập dữ liệu Iris:

Giới thiệu về tập dữ liệu Iris

- **Nguồn gốc**: Tập dữ liệu này lần đầu tiên được giới thiệu bởi nhà thống kê người Anh, Sir Ronald A. Fisher, trong bài báo của ông năm 1936 "The use of multiple measurements in taxonomic problems".
- **Số mẫu**: 150
- **Số lượng biến (features)**: 4

- **Số lượng lớp (classes)**: 3

Các đặc trưng (features)

Tập dữ liệu bao gồm bốn đặc trưng đo lường về hình dạng của hoa Iris:

1. **Chiều dài cánh hoa (sepal length)**: Tính bằng cm.
2. **Chiều rộng cánh hoa (sepal width)**: Tính bằng cm.
3. **Chiều dài nhị hoa (petal length)**: Tính bằng cm.
4. **Chiều rộng nhị hoa (petal width)**: Tính bằng cm.

Các lớp (classes)

Tập dữ liệu Iris được chia thành ba lớp, mỗi lớp đại diện cho một loài hoa Iris khác nhau:

1. **Iris-setosa**
2. **Iris-versicolor**
3. **Iris-virginica**

Mỗi lớp có 50 mẫu, làm cho tổng số mẫu là 150.

Khái niệm về tứ phân vị

Tứ phân vị (quartiles) là các giá trị chia một tập dữ liệu thành bốn phần bằng nhau. Tứ phân vị giúp mô tả sự phân phối của dữ liệu, đặc biệt là để hiểu sự biến động và phát hiện các giá trị ngoại lệ. Dưới đây là các khái niệm cơ bản về tứ phân vị:

Các tứ phân vị chính

1. **Tứ phân vị thứ nhất (Q1)**: Còn được gọi là phần tư dưới, là giá trị mà dưới đó có 25% dữ liệu.
2. **Tứ phân vị thứ hai (Q2)**: Chính là giá trị trung vị, chia tập dữ liệu thành hai phần bằng nhau, với 50% dữ liệu dưới giá trị này.
3. **Tứ phân vị thứ ba (Q3)**: Còn được gọi là phần tư trên, là giá trị mà dưới đó có 75% dữ liệu.
4. **Khoảng cách tứ phân vị (IQR - Interquartile Range)**: Là khoảng cách giữa Q3 và Q1, tức là IQR = Q3 - Q1. Khoảng cách này giúp xác định mức độ phân tán của tập dữ liệu và phát hiện các giá trị ngoại lệ.

Ví dụ

Giả sử bạn có tập dữ liệu sau: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

1. **Sắp xếp dữ liệu theo thứ tự tăng dần** (dữ liệu đã sắp xếp):
   - [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2. **Xác định Q1**:
   - Q1 là giá trị ở vị trí 25% của tập dữ liệu.
   - Với 10 giá trị, Q1 nằm giữa giá trị thứ 2 và thứ 3.
   - Q1 = (2 + 3) / 2 = 2.5
3. **Xác định Q2 (trung vị)**:
   - Q2 là giá trị ở vị trí 50% của tập dữ liệu.
   - Với 10 giá trị, Q2 nằm giữa giá trị thứ 5 và thứ 6.
   - Q2 = (5 + 6) / 2 = 5.5
4. **Xác định Q3**:
   - Q3 là giá trị ở vị trí 75% của tập dữ liệu.
   - Với 10 giá trị, Q3 nằm giữa giá trị thứ 8 và thứ 9.
   - Q3 = (8 + 9) / 2 = 8.5
5. **Xác định IQR**:
   - IQR = Q3 - Q1 = 8.5 - 2.5 = 6

Sử dụng tứ phân vị để phát hiện giá trị ngoại lệ

Các giá trị ngoại lệ thường được xác định bằng cách sử dụng quy tắc 1.5 * IQR. Cụ thể:

- Giá trị ngoại lệ dưới là bất kỳ giá trị nào nhỏ hơn Q1 - 1.5 * IQR.
- Giá trị ngoại lệ trên là bất kỳ giá trị nào lớn hơn Q3 + 1.5 * IQR.

Ví dụ sử dụng Python để tính tứ phân vị

Dưới đây là một ví dụ sử dụng thư viện NumPy và Pandas để tính tứ phân vị và IQR cho một tập dữ liệu:

```
import numpy as np
import pandas as pd
```

```
# Tạo một tập dữ liệu ví dụ
data = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Sử dụng Pandas để tính tứ phân vị
df = pd.DataFrame(data, columns=['value'])
Q1 = df['value'].quantile(0.25)
Q2 = df['value'].quantile(0.5)
Q3 = df['value'].quantile(0.75)
IQR = Q3 - Q1

print("Q1:", Q1)
print("Q2:", Q2)
print("Q3:", Q3)
print("IQR:", IQR)

# Xác định giá trị ngoại lệ
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print("Lower Bound:", lower_bound)
print("Upper Bound:", upper_bound)
```

Giá trị trung vị

Giá trị trung vị (median) là một thước đo thống kê, đại diện cho giá trị nằm ở giữa một tập dữ liệu được sắp xếp theo thứ tự tăng dần hoặc giảm dần. Trung vị chia tập dữ liệu thành hai phần bằng nhau: 50% giá trị của tập dữ liệu nằm dưới trung vị và 50% giá trị nằm trên trung vị.

Cách tính giá trị trung vị

1. **Sắp xếp dữ liệu**: Đầu tiên, sắp xếp các giá trị của tập dữ liệu theo thứ tự tăng dần hoặc giảm dần.
2. **Xác định vị trí trung vị**:
    - Nếu số lượng giá trị (n) là lẻ, trung vị là giá trị ở vị trí giữa của tập dữ liệu.

- Nếu số lượng giá trị (n) là chẵn, trung vị là trung bình của hai giá trị ở giữa của tập dữ liệu.

Ví dụ

Trường hợp số lượng giá trị là lẻ:
Giả sử bạn có tập dữ liệu sau: [3, 1, 4, 1, 5]

1. Sắp xếp dữ liệu: [1, 1, 3, 4, 5]
2. Trung vị là giá trị ở vị trí thứ 3 (giá trị giữa): Trung vị = 3

Trường hợp số lượng giá trị là chẵn:
Giả sử bạn có tập dữ liệu sau: [3, 1, 4, 1]

1. Sắp xếp dữ liệu: [1, 1, 3, 4]
2. Trung vị là trung bình của hai giá trị ở giữa (vị trí thứ 2 và 3): Trung vị = (1 + 3) / 2 = 2

Lợi ích của giá trị trung vị

- **Không bị ảnh hưởng bởi các giá trị ngoại lệ**: Trung vị không bị ảnh hưởng bởi các giá trị cực kỳ lớn hoặc cực kỳ nhỏ, làm cho nó trở thành một thước đo tốt hơn trung bình (mean) trong các tập dữ liệu có nhiều giá trị ngoại lệ.
- **Dễ hiểu và tính toán**: Trung vị đơn giản và dễ tính toán, đặc biệt là với các tập dữ liệu nhỏ.

Ví dụ sử dụng Python để tính giá trị trung vị

Dưới đây là một ví dụ sử dụng thư viện NumPy để tính giá trị trung vị:

```
import numpy as np

# Tạo một tập dữ liệu ví dụ
data = [3, 1, 4, 1, 5]

# Tính trung vị
median = np.median(data)

print("Giá trị trung vị:", median)
```

Biểu đồ histogram

Biểu đồ histogram là một loại biểu đồ dùng để biểu thị phân phối tần suất (hoặc tần số) của một biến số số lượng (numeric variable). Đây là công cụ hữu ích để thể hiện sự phân bố của dữ liệu và giúp đánh giá tính đại diện của mẫu dữ liệu đối với toàn bộ quần thể.

Đặc điểm chính của biểu đồ histogram:

1. **Phân bố dữ liệu**: Biểu đồ histogram cho phép chúng ta quan sát cách mà dữ liệu được phân bố trên một khoảng giá trị liên tục. Mỗi cột của histogram biểu thị số lượng các quan sát rơi vào một khoảng giá trị nhất định, được gọi là "bin" (ngăn).
2. **Trục ngang và trục dọc**: Trục ngang (x-axis) của biểu đồ histogram biểu thị các giá trị của biến số số lượng, trong khi trục đứng (y-axis) biểu thị tần suất hoặc tỷ lệ phần trăm của số lượng các quan sát trong mỗi bin.
3. **Khối lượng dữ liệu**: Chiều rộng của mỗi bin thường được lựa chọn sao cho tổng số quan sát trong mỗi bin có thể dễ dàng so sánh và đưa ra những nhận định về phân bố của dữ liệu.

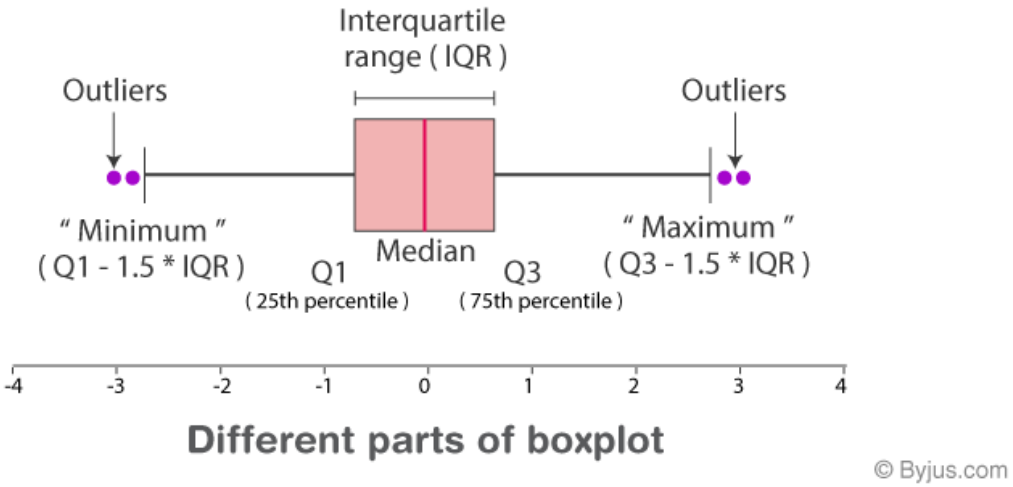
Ví dụ về mã nguồn Python để tạo biểu đồ histogram:

Dưới đây là một ví dụ đơn giản với Python sử dụng thư viện Matplotlib để tạo biểu đồ histogram từ một tập dữ liệu:

```
import matplotlib.pyplot as plt
import numpy as np

# Tạo một tập dữ liệu ngẫu nhiên
data = np.random.randn(1000)

# Vẽ biểu đồ histogram
plt.hist(data, bins=30, edgecolor='black')
plt.xlabel('Giá trị biến số số lượng')
plt.ylabel('Tần suất')
plt.title('Biểu đồ histogram')

# Hiển thị biểu đồ
plt.show()
```

Biểu đồ hộp Box-Plot

Biểu đồ hộp (Boxplot) là một loại biểu đồ thống kê sử dụng để mô tả phân phối của dữ liệu thông qua các đại lượng thống kê cơ bản như median, tứ phân vị và phạm vi giá trị. Nó cho phép nhìn nhận sự phân tán của dữ liệu và phát hiện các điểm ngoại lệ (outlier) một cách trực quan.



Biểu đồ hộp (Box Plot)

Định nghĩa

Biểu đồ hộp trong tiếng Anh là **Box Plot** hay **Box and Whisker plot**.

Biểu đồ hộp do John Tukey sáng tạo ra năm 1977.

Biểu đồ hộp (Box plot) hay còn gọi là *biểu đồ hộp và râu (Box and whisker plot)* là biểu đồ diễn tả 5 vị trí phân bố của dữ liệu, đó là: giá trị nhỏ nhất (min), tứ phân vị thứ nhất (Q1), trung vị (median), tứ phân vị thứ 3 (Q3) và giá trị lớn nhất (max).

Đặc trưng của biểu đồ hộp

- **Biểu đồ hộp** giúp biểu diễn các đại lượng quan trọng của dãy số như giá trị nhỏ nhất (min), giá trị lớn nhất (max), tứ phân vị (quartile), khoảng biến thiên tứ phân vị (Interquartile Range) một cách trực quan, dễ hiểu.
- Trên biểu đồ hộp, ngoài các đại lượng số trung bình, trung vị, còn thể hiện một số thông số sau:

(1) Số phân tử hay còn gọi là tứ phân vị (Quartiles): Tứ phân vị là đại lượng mô tả sự phân bố và sự phân tán của tập dữ liệu. Số phân tử có 3 giá trị, đó là số phân

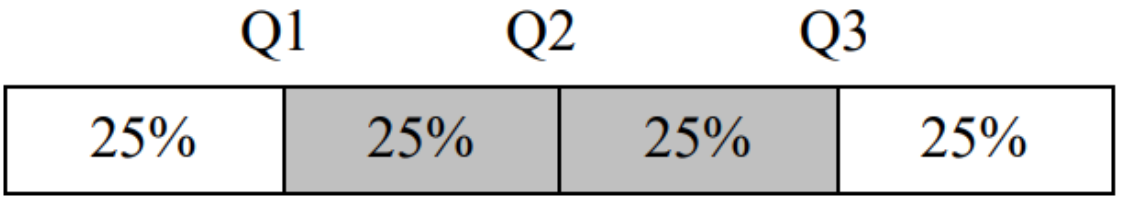
tử thứ nhất (Q1), thứ nhì (Q2) và thứ ba (Q3). Ba giá trị này chia một tập hợp dữ liệu (đã sắp xếp dữ liệu theo trật từ bé đến lớn) thành 4 phần có số lượng quan sát đều nhau.

Tứ phân vị được xác định như sau:

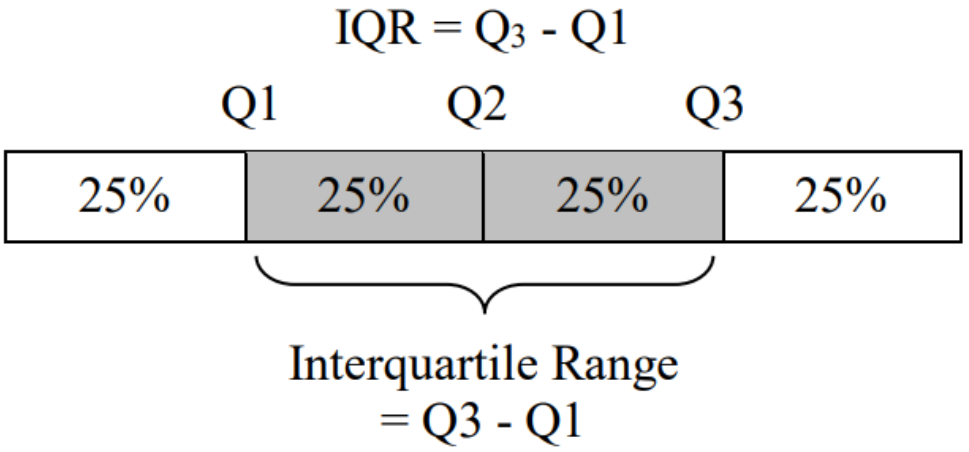
+ Sắp xếp các số theo thứ tự tăng dần

+ Cắt dãy số thành 4 phần bằng nhau

+ Tứ phân vị là các giá trị tại vị trí cắt



(2) Khoảng biến thiên số phân tử (Interquartile Range - IQR) IQR được xác định như sau:

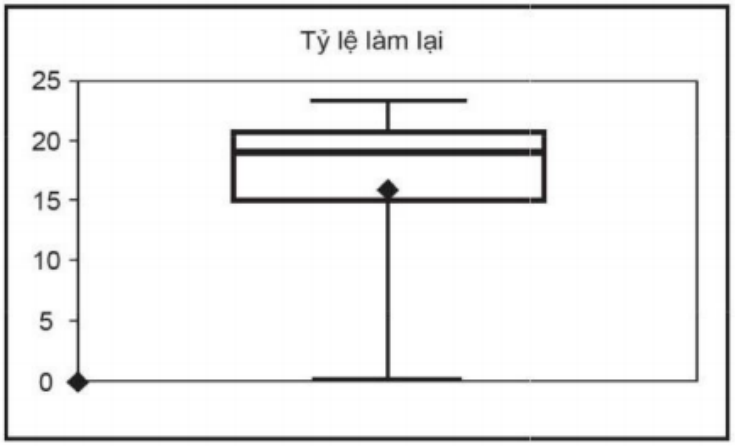


Ví dụ

Xét một ví dụ về việc sử dụng biểu đồ hộp:

Dưới đây mô tả sử dụng biểu đồ hộp để phân tích, nhận biết vấn đề.

Ví dụ, với số liệu thu thập được về tỉ lệ làm lại (Rework Ratio) trong quá trình sản xuất, (có xmin = 0,0; Q1 = 14,9; x = 19,0; x = 15,8; Q3 = 20,6; xmax =23,2) ta có biểu đồ hộp với hình dáng biểu đồ như sau:



Rework Ratio	
Min (Actual):	0.0
1st Quartile:	14.9
Median:	19.0
Mean:	15.8
3rd Quartile:	20.6
Max (Actual):	23.2

Hình 2.20.a: Hình dáng biểu đồ hộp

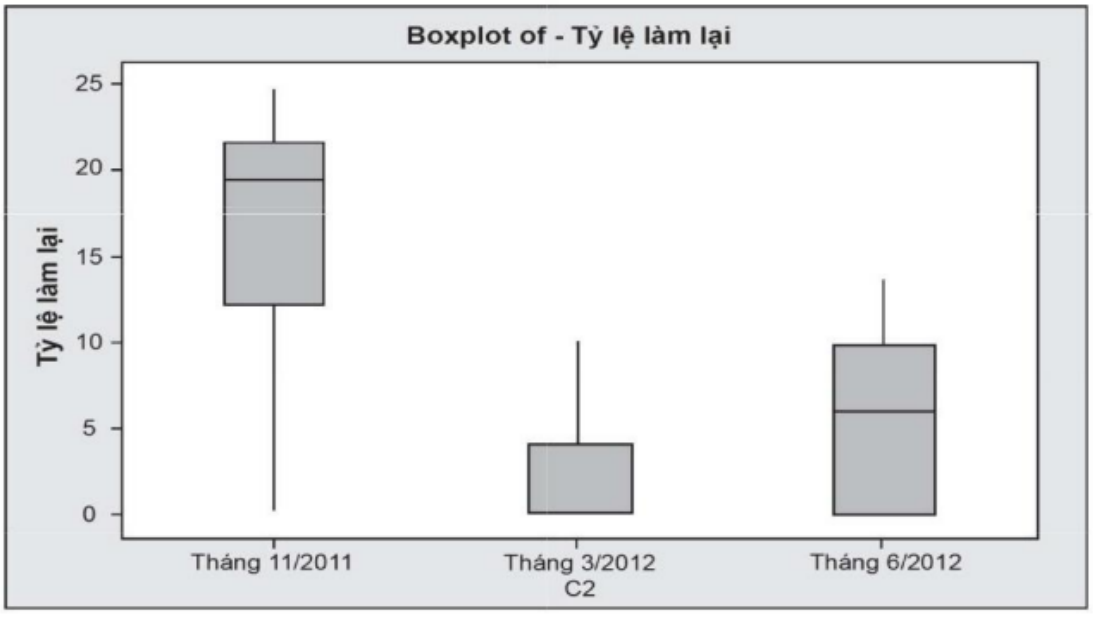
Trung bình tỉ lệ làm lại là 15,8%, trung vị là 19%.

Dữ liệu có xu hướng nghiêng nhiều về phía trên giá trị trung bình.

Khoảng số phân tử = Q3 - Q1 = 20,6 - 14,9 = 5,7

Khoảng cách giữa giá trị lớn nhất và nhỏ nhất là 23,2 - 0 = 23,2.

Nhìn chung, tỉ lệ làm lại cao và quá trình có sự dao động lớn, không ổn định, kiểm soát chất lượng kém. Tuy nhiên, biểu đồ hộp giúp nhìn trực quan hơn khi so sánh 3 giai đoạn hoặc khu vực khác nhau như hình dưới đây:



Hình 2.20.b: Biểu đồ hộp tỷ lệ làm lại ở 3 giai đoạn

Nhận xét:

Với ba lần thu thập dữ liệu về tỉ lệ làm lại vào thời điểm tháng 11/2011, tháng 3/2012 và tháng 6/2012, dữ liệu vào thời điểm tháng 11/2011 cho thấy quá trình kiểm soát lỗi kém vì xu hướng tập trung của dữ liệu (trung vị) ở mức cao, độ dao động lớn.

Kiểm soát chất lượng vào thời điểm tháng 3/2012 là tốt nhất vì dữ liệu về tỉ lệ làm lại tập trung ở mức thấp, dao động ở phạm vi hẹp.

Bài tập về Gradient Descent

Hãy sử dụng phương pháp Gradient Descent để tìm giá trị tối ưu x^* của hàm số sau:

$$f(x) = x^4 - 3x^3 + 2x^2 + 1$$

Thực hiện phương pháp trên bằng 2 cách:

1. Thực hiện theo công thức tính Gradient Descent trên giấy:

$$x_{t+1} = x_t - \eta f'(x_t)$$

2. Code trên Python, so sánh với kết quả của cách 1 có giống nhau hay không?

Bài tập KNN

Bài tập 1

Bài toán: Dự đoán lớp của một điểm dữ liệu mới dựa trên tập dữ liệu cho trước bằng thuật toán KNN.

Tập dữ liệu:

Index	Feature 1	Feature 2	Class
1	1	2	A
2	2	3	A
3	3	3	B
4	6	5	B
5	7	8	B
6	8	8	B
7	7	6	A

Điểm dữ liệu mới cần dự đoán:

$$\text{New Point} = (4, 4)$$

Yêu cầu:

1. Tính khoảng cách Euclidean từ điểm dữ liệu mới đến tất cả các điểm dữ liệu trong tập dữ liệu.
2. Sắp xếp các khoảng cách này và chọn K điểm gần nhất (giả sử K=3).
3. Dự đoán lớp của điểm dữ liệu mới dựa trên lớp của K điểm gần nhất.

Bước 1: Tính Khoảng Cách Euclidean

Khoảng cách Euclidean giữa hai điểm (x_1, y_1) và (x_2, y_2) được tính bằng công thức:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Tính khoảng cách từ điểm dữ liệu mới $(4, 4)$ đến tất cả các điểm trong tập dữ liệu:

1. Đến điểm $(1, 2)$:

$$d = \sqrt{(4-1)^2 + (4-2)^2} = \sqrt{3^2 + 2^2} = \sqrt{9+4} = \sqrt{13} \approx 3.61$$

2. Đến điểm $(2, 3)$:

$$d = \sqrt{(4-2)^2 + (4-3)^2} = \sqrt{2^2 + 1^2} = \sqrt{4+1} = \sqrt{5} \approx 2.24$$

3. Đến điểm $(3, 3)$:

$$d = \sqrt{(4-3)^2 + (4-3)^2} = \sqrt{1^2 + 1^2} = \sqrt{2} \approx 1.41$$

4. Đến điểm $(6, 5)$:

$$d = \sqrt{(4-6)^2 + (4-5)^2} = \sqrt{2^2 + 1^2} = \sqrt{4+1} = \sqrt{5} \approx 2.24$$

5. Đến điểm $(7, 8)$:

$$d = \sqrt{(4-7)^2 + (4-8)^2} = \sqrt{3^2 + 4^2} = \sqrt{9+16} = \sqrt{25} = 5$$

6. Đến điểm $(8, 8)$:

$$d = \sqrt{(4-8)^2 + (4-8)^2} = \sqrt{4^2 + 4^2} = \sqrt{16 + 16} = \sqrt{32} \approx 5.66$$

7. Đến điểm $(7, 6)$:

$$d = \sqrt{(4-7)^2 + (4-6)^2} = \sqrt{3^2 + 2^2} = \sqrt{9 + 4} = \sqrt{13} \approx 3.61$$

Bước 2: Sắp Xếp và Chọn K Điểm Gần Nhất (K=3)

Sắp xếp các khoảng cách:

1. $(3, 3)$ với khoảng cách ≈ 1.41 (Class B)
2. $(2, 3)$ với khoảng cách ≈ 2.24 (Class A)
3. $(6, 5)$ với khoảng cách ≈ 2.24 (Class B)

Bước 3: Dự Đoán Lớp

Dựa trên lớp của 3 điểm gần nhất:

- Class B: 2 điểm
- Class A: 1 điểm

Vậy, dự đoán lớp của điểm dữ liệu mới $(4, 4)$ là **Class B**.

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier

# Dữ liệu
X = np.array([[1, 2], [2, 3], [3, 3], [6, 5], [7, 8], [8, 8],
y = np.array(['A', 'A', 'B', 'B', 'B', 'B', 'A'])

# Điểm dữ liệu mới
new_point = np.array([[4, 4]])

# Khởi tạo và huấn luyện mô hình KNN với K = 3
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X, y)

# Dự đoán lớp của điểm dữ liệu mới
predicted_class = knn.predict(new_point)
print(f"Lớp dự đoán của điểm dữ liệu {new_point} là: {predict
```

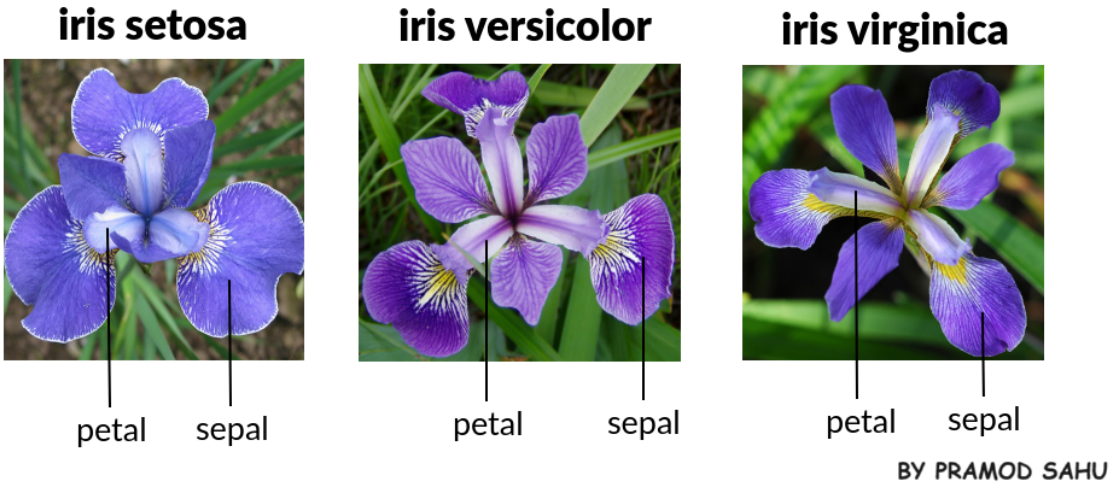
```
Lớp dự đoán của điểm dữ liệu [[4 4]] là: B
```

Bài tập 2: Bài Toán Phân Loại Hoa Iris

Mô tả bài toán:
Bài toán phân loại hoa Iris sử dụng tập dữ liệu Iris, gồm 150 mẫu hoa thuộc ba loài khác nhau: Setosa, Versicolor, và Virginica. Mỗi mẫu được mô tả bằng bốn đặc trưng:

1. Chiều dài đài hoa (sepal length)
2. Chiều rộng đài hoa (sepal width)
3. Chiều dài cánh hoa (petal length)
4. Chiều rộng cánh hoa (petal width)

Mục tiêu là sử dụng các đặc trưng này để dự đoán loài hoa.



Các Bước Thực Hiện

1. **Tải và khám phá tập dữ liệu Iris**
2. **Chia tập dữ liệu thành tập huấn luyện và tập kiểm tra**
3. **Chuẩn hóa dữ liệu**
4. **Áp dụng thuật toán KNN để huấn luyện mô hình**
5. **Đánh giá mô hình trên tập kiểm tra**
6. **Thử nghiệm với các giá trị khác nhau của K**

1. Khám phá dữ liệu

```
# Import các thư viện cần thiết
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris

# Tải tập dữ liệu Iris
iris = load_iris()
X = iris.data
y = iris.target

# Chuyển dữ liệu thành DataFrame để dễ dàng khám phá
df = pd.DataFrame(data=np.c_[X, y], columns=iris.feature_name
print(df.head())
```

2. Chia Tập Dữ Liệu Thành Tập Huấn Luyện và Tập Kiểm Tra

```
from sklearn.model_selection import train_test_split

# Chia tập dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, t
est_size=0.3, random_state=42)
```

Sử dụng
`train_test_split` để chia dữ liệu, trong đó 70% dữ liệu dùng để huấn luyện và 30% để kiểm tra.

3. Chuẩn Hóa Dữ Liệu

```
from sklearn.preprocessing import StandardScaler

# Chuẩn hóa dữ liệu
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Sử dụng `StandardScaler` để chuẩn hóa dữ liệu, giúp các đặc trưng có giá trị trung bình bằng 0 và độ lệch chuẩn bằng 1.

4. Áp Dụng Thuật Toán KNN Để Huấn Luyện Mô Hình

```
from sklearn.neighbors import KNeighborsClassifier

# Khởi tạo mô hình KNN với K = 3
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

5. Đánh Giá Mô Hình Trên Tập Kiểm Tra

```
from sklearn.metrics import accuracy_score, classification_re

# Dự đoán trên tập kiểm tra
y_pred = knn.predict(X_test)

# Đánh giá độ chính xác và hiển thị báo cáo phân loại
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_tes
```

6. Thử Nghiệm Với Các Giá Trị Khác Nhau Của K

```
accuracy_list = []
k_range = range(1, 21)

# Thử nghiệm với các giá trị khác nhau của K
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy_list.append(accuracy_score(y_test, y_pred))

# Vẽ đồ thị độ chính xác theo K
import matplotlib.pyplot as plt
```

```
plt.plot(k_range, accuracy_list)
plt.xlabel('Value of K for KNN')
plt.ylabel('Testing Accuracy')
plt.title('Accuracy vs. K Value')
plt.show()
```