เอกสารประกอบการสอน

รหัสวิชา CPE5007

ปัญญาประดิษฐ์

โดย

พรภวิษย์ บุญศรีเมือง

สาขาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์และเทคโนโลยีอุตสาหกรรม

1. โมเดลแรก: การสร้างและจัดการข้อมูลพื้นฐาน

(Basic Data Creation and Manipulation)

สำหรับโมเดลแรก ควรเริ่มต้นจากสิ่งที่ง่ายที่สุดและเป็นพื้นฐานของการประมวลผลข้อมูลในวิศวกรรมคอมพิวเตอร์ ซึ่งก็คือ **การสร้างและจัดการกับข้อมูลตัวเลขในรูปแบบของเวกเตอร์และเมทริกซ์** MATLAB ถูกสร้างขึ้นมาเพื่อทำงานกับเมทริกซ์โดยเฉพาะ

- **ตัวแปร (Variables):** การกำหนดค่าให้กับตัวแปรชนิดต่างๆ (เช่น x = 5, name = 'LISA')

- **เวกเตอร์ (Vectors):**
  - การสร้างเวกเตอร์แถว (row vector) และเวกเตอร์คอลัมน์ (column vector) (v = [1 2 3], w = [4;5;6])
  - การเข้าถึงข้อมูลในเวกเตอร์ (indexing) (v(2))
  - การดำเนินการทางคณิตศาสตร์กับเวกเตอร์ (element-wise operations, vector addition/subtraction)

- **เมทริกซ์ (Matrices):**
  - การสร้างเมทริกซ์ (M = [1 2; 3 4])
  - การเข้าถึงข้อมูลในเมทริกซ์ (indexing) (M(1,2))
  - การดำเนินการทางคณิตศาสตร์กับเมทริกซ์ (matrix multiplication, element-wise multiplication, transpose)

- **ฟังก์ชันพื้นฐาน:**
  - length(), size() เพื่อดูขนาดของเวกเตอร์/เมทริกซ์
  - zeros(), ones(), rand() สำหรับสร้างเมทริกซ์ที่มีค่าเริ่มต้น
  - sum(), mean(), max(), min() สำหรับการคำนวณสถิติเบื้องต้น

Model of Simulation

- **พื้นฐาน:** การเข้าใจเวกเตอร์และเมทริกซ์เป็นรากฐานสำคัญสำหรับงานวิศวกรรมคอมพิวเตอร์เกือบทุกแขนง ไม่ว่าจะเป็นการประมวลผลสัญญาณ ภาพ การควบคุม หรือแม้แต่ Machine Learning
- **ง่ายต่อการมองเห็นผลลัพธ์:** ผู้เรียนสามารถเห็นผลลัพธ์ของการดำเนินการต่างๆ ได้ทันทีใน Command Window ของ MATLAB
- **ใช้ประโยชน์จากจุดแข็งของ MATLAB:** MATLAB ถูกออกแบบมาเพื่อจัดการกับเมทริกซ์โดยเฉพาะ ทำให้การเรียนรู้แนวคิดเหล่านี้เป็นไปอย่างราบรื่น

---

2. ซอฟต์แวร์แรก: โปรแกรมประมวลผลข้อมูลอย่างง่าย (Simple Data Processing Program)

หลังจากที่เข้าใจการจัดการข้อมูลพื้นฐานแล้ว ซอฟต์แวร์แรกที่ควรสร้างคือ **โปรแกรมที่รับข้อมูลเข้า ประมวลผล และแสดงผลลัพธ์** โดยใช้สคริปต์ (.m file) ใน MATLAB

แนวคิดที่ควรเน้น:

- **สคริปต์ M-file:** การเขียนโค้ดลงในไฟล์ .m เพื่อให้สามารถรันซ้ำได้
- **การรับข้อมูลเข้า (Input):**
    - ใช้ฟังก์ชัน input() เพื่อรับค่าจากผู้ใช้ (num = input('Enter a number: ');)
- **การประมวลผลเงื่อนไข (Conditional Statements):**
    - if-else-end เพื่อตัดสินใจตามเงื่อนไข (if num > 0 ... else ... end)
- **การวนซ้ำ (Loops):**
    - for loop สำหรับการทำงานซ้ำตามจำนวนครั้งที่กำหนด (for i = 1:N ... end)
    - while loop สำหรับการทำงานซ้ำจนกว่าเงื่อนไขจะไม่เป็นจริง (while condition ... end)

- **การแสดงผลลัพธ์ (Output):**
  - disp() เพื่อแสดงข้อความหรือค่าตัวแปร
  - fprintf() เพื่อจัดรูปแบบการแสดงผลที่ซับซ้อนขึ้น

- **การสร้างกราฟ (Basic Plotting):**
  - plot() เพื่อสร้างกราฟ 2D อย่างง่าย (เช่น plot(x, y))
  - xlabel(), ylabel(), title(), legend() เพื่อเพิ่มองค์ประกอบกราฟ

- **การเขียนฟังก์ชัน (Basic Function Creation):**
  - การสร้างฟังก์ชัน .m ไฟล์ของตัวเอง เพื่อให้โค้ดเป็นระเบียบและนำกลับมาใช้ใหม่ได้ (function output = myFunction(input) ... end)

(Concept):

1. **รับข้อมูลตัวเลขจากผู้ใช้:** เช่น รับค่าคะแนนนักเรียน
2. **ประมวลผล:** ตรวจสอบว่าคะแนนผ่านเกณฑ์ที่กำหนดหรือไม่ (เช่น มากกว่า 50)
3. **แสดงผลลัพธ์:** แสดงข้อความว่า "ผ่าน" หรือ "ไม่ผ่าน"
4. **(เพิ่มเติม) สร้างกราฟ:** ถ้ามีชุดข้อมูลคะแนนหลายคน อาจจะให้วาดกราฟแท่ง (bar plot) หรือกราฟเส้น (line plot) ของคะแนนเหล่านั้น

ทำไมถึงเป็นซอฟต์แวร์นี้?

- **ครบวงจรพื้นฐาน:** โปรแกรมนี้ครอบคลุมองค์ประกอบพื้นฐานทั้งหมดของการเขียนโปรแกรม: Input, Process, Output รวมถึงการควบคุมการไหลของโปรแกรมด้วยเงื่อนไขและการวนซ้ำ
- **เห็นประโยชน์จริง:** ผู้เรียนสามารถเห็นว่าโค้ดที่เขียนขึ้นมาสามารถทำงานบางอย่างที่เป็นประโยชน์ได้จริง
- **ปูทางสู่ความซับซ้อน:** เป็นจุดเริ่มต้นที่ดีในการทำความเข้าใจโครงสร้างของโปรแกรม ก่อนที่จะก้าวไปสู่การพัฒนาซอฟต์แวร์ที่ซับซ้อนขึ้น เช่น การประมวลผลภาพ หรือระบบควบคุม

สรุปภาพรวม:

เริ่มต้นด้วย **พื้นฐานข้อมูล** ใน MATLAB (เวกเตอร์, เมทริกซ์) เพื่อให้คุ้นเคยกับสภาพแวดล้อมและวิธีการจัดการข้อมูลของ MATLAB จากนั้นจึงขยับไปที่ **การเขียนโปรแกรมพื้นฐาน** โดยใช้สคริปต์และฟังก์ชัน เพื่อให้ผู้เรียนสามารถสร้างโปรแกรมที่รับข้อมูล ประมวลผล และแสดงผลได้ การเรียนรู้แบบนี้จะช่วยให้ผู้เรียนมีรากฐานที่แข็งแกร่งในการต่อยอดไปสู่หัวข้อที่ซับซ้อนขึ้นในวิศวกรรมคอมพิวเตอร์ต่อไป

II.โมเดล AI แรก: การถดถอยเชิงเส้นอย่างง่าย

(Simple Linear Regression)

แนวคิดหลัก: การถดถอยเชิงเส้นเป็นการทำนายค่าตัวเลข (ค่า Y) จากค่าตัวเลขอีกค่าหนึ่ง (ค่า X) โดยสมมติว่ามีความสัมพันธ์เชิงเส้นตรงระหว่างกัน คือ

เราจะหา **สมการเส้นตรงที่ดีที่สุด** (Y=mX+c) ที่ใช้อธิบายชุดข้อมูลของเราได้ โมเดลนี้เป็นพื้นฐานของ Machine Learning หลายๆ ตัวและเข้าใจง่าย เหมาะสำหรับผู้เริ่มต้น

Linear Regression?

- **พื้นฐาน AI ที่เข้าใจง่าย:** เป็นหนึ่งในอัลกอริธึม Machine Learning ที่ง่ายที่สุดในการทำความเข้าใจแนวคิดพื้นฐาน เช่น การเรียนรู้จากข้อมูล, การทำนาย, และการวัดประสิทธิภาพของโมเดล

- **เห็นภาพชัดเจน:** สามารถแสดงผลลัพธ์เป็นเส้นตรงบนกราฟ 2 มิติได้ ทำให้เห็นการทำงานของโมเดลได้ทันที

- **MATLAB พื้นฐาน:** อาศัยการจัดการเวกเตอร์ เมทริกซ์ และการพล็อตกราฟ

ซอฟต์แวร์ AI แรก: โปรแกรมทำนายค่าด้วย Linear Regression ใน MATLAB

1. **สร้างข้อมูลตัวอย่าง:** จำลองสถานการณ์ที่เรามีข้อมูลคู่ X และ Y
2. **ฝึกโมเดล (Train Model):** ค้นหาสมการเส้นตรง (m และ c) ที่เหมาะสมที่สุดกับข้อมูลของเรา
3. **ทำนายค่าใหม่ (Prediction):** ใช้โมเดลที่ได้ไปทำนายค่า Y จากค่า X ใหม่
4. **แสดงผลด้วยกราฟ:** แสดงจุดข้อมูลและเส้นตรงที่โมเดลเรียนรู้ได้ เพื่อให้เห็นภาพการทำงาน

โมเดล AI แรกสำหรับ Data Communication: การจำแนกประเภทข้อมูล/สถานะเครือข่าย (Classification)

แนวคิดหลัก: ใน Data Communication เรามักจะต้องรู้ว่าข้อมูลที่รับส่งเป็นข้อมูลประเภทใด (เช่น เสียง, วิดีโอ, ข้อความ) หรือสถานะของเครือข่ายเป็นอย่างไร (เช่น ปกติ, มีความแออัด, มีการโจมตี) **โมเดลการจำแนกประเภท (Classification Model)** จะเรียนรู้จากชุดข้อมูลที่มีการระบุประเภทหรือสถานะไว้ล่วงหน้า (labeled data) เพื่อให้สามารถทำนายประเภทหรือสถานะของข้อมูลใหม่ที่ไม่เคยเห็นมาก่อนได้

Classification?

- **ปัญหาที่พบบ่อยในการสื่อสารข้อมูล:** การจำแนกประเภทเป็นงานพื้นฐานและสำคัญใน Data Communication เช่น

- **การระบุประเภทของ Traffic:** แยกแยะว่า Packet ที่วิ่งในเครือข่ายเป็นของโปรแกรม Video Streaming, Web Browse หรือ VoIP (Voice over IP)
- **การตรวจจับความผิดปกติ (Anomaly Detection):** ระบุว่า Traffic มีลักษณะผิดปกติที่อาจบ่งชี้ถึงการโจมตีทางไซเบอร์หรือไม่
- **การจัดลำดับความสำคัญ (QoS - Quality of Service):** จำแนกประเภทของ Traffic เพื่อจัดสรร Bandwidth ให้เหมาะสม

- **มองเห็นผลลัพธ์ชัดเจน:** ผลลัพธ์ของการจำแนกคือ "ประเภท" ที่ชัดเจน เช่น "Voice", "Video", "Attack", "Normal"
- **ต่อยอดได้ง่าย:** เป็นรากฐานสำหรับงานที่ซับซ้อนขึ้น เช่น Intrusion Detection Systems (IDS), Network Performance Monitoring

ซอฟต์แวร์ AI แรก: โปรแกรมจำแนกสถานะเครือข่ายอย่างง่ายใน MATLAB

เราจะใช้ **Support Vector Machine (SVM)** ซึ่งเป็นอัลกอริธึม Classification ยอดนิยมและมีประสิทธิภาพ เพื่อสร้างโมเดลที่สามารถจำแนกสถานะเครือข่ายได้จากคุณลักษณะบางอย่าง

สถานการณ์จำลอง: เราจะสร้างข้อมูลสมมติที่แสดงคุณลักษณะ 2 อย่างของเครือข่าย (เช่น **Latency** และ **Packet Loss Rate**) และสถานะของเครือข่ายว่าอยู่ในสภาวะ **"ปกติ"** หรือ **"มีปัญหา"** โมเดล SVM จะเรียนรู้จากข้อมูลนี้เพื่อจำแนกสถานะของเครือข่ายใหม่ๆ

```
% --- AI Model for Data Communication: Simple %Classification with SVM in MATLAB ---

% 1. สร้างข้อมูลตัวอย่าง (Simulate Sample Data for Network Status)

% สมมติว่ามี 2 คุณลักษณะ (features): Latency (หน่วงเวลา) และ Packet
```

```
% Loss Rate (อัตราการสูญหายของแพ็กเก็ต)
% และมี 2 Class (สถานะ): 0 = 'Normal', 1 = 'Problematic'
% ข้อมูลสถานะปกติ (Normal)
X_normal = [
    5, 0.1;
    7, 0.2;
    6, 0.15;
    8, 0.3;
    4, 0.05
];
Y_normal = zeros(size(X_normal, 1), 1); % Class 0: Normal

% ข้อมูลสถานะมีปัญหา (Problematic)
X_problem = [
    20, 1.5;
    25, 2.0;
    18, 1.0;
    30, 2.5;
    22, 1.8
];
Y_problem = ones(size(X_problem, 1), 1); % Class 1: Problematic
```

```
% รวมข้อมูลทั้งหมด

X = [X_normal; X_problem]; % คุณลักษณะทั้งหมด (Latency, Packet Loss Rate)

Y = [Y_normal; Y_problem]; % คลาส/สถานะทั้งหมด

fprintf('--- ข้อมูลตัวอย่างสำหรับสถานะเครือข่าย ---\n');

fprintf('  Latency (ms) | Packet Loss (%%) | Status (0=Normal, 1=Problem)\n');

disp([X, Y]);

fprintf('\n');

% 2. ฝึกโมเดล Support Vector Machine (SVM) (Train the SVM % Model)

% เราจะใช้ฟังก์ชัน fitcsvm ของ MATLAB สำหรับการฝึกโมเดล SVM

% 'KernelFunction', 'linear' หมายถึงการใช้เส้นตรงในการแบ่งข้อมูล

% 'Standardize', true หมายถึงการปรับข้อมูลให้อยู่ใน scale เดียวกัน ซึ่งช่วยให้ %โมเดลทำงานได้ดีขึ้น

svm_model = fitcsvm(X, Y, 'KernelFunction', 'linear', 'Standardize', true);

fprintf('--- ผลลัพธ์การฝึกโมเดล SVM ---\n');

fprintf('โมเดล SVM ถูกสร้างขึ้นเรียบร้อยแล้ว\n');

fprintf('\n');

% 3. ทำนายสถานะเครือข่ายใหม่ (Predict New Network Status)

% สมมติว่าเรามีข้อมูลสถานะเครือข่ายใหม่ที่ต้องการจำแนก
```

```
X_new_1 = [7, 0.25]; % ค่า Latency และ Packet Loss ที่ควรจะเป็น Normal

X_new_2 = [28, 2.2]; % ค่า Latency และ Packet Loss ที่ควรจะเป็น Problematic

X_new_3 = [12, 0.7]; % ค่ากลางๆ ลองดูว่าโมเดลจะจำแนกอย่างไร

[label_1, score_1] = predict(svm_model, X_new_1);

[label_2, score_2] = predict(svm_model, X_new_2);

[label_3, score_3] = predict(svm_model, X_new_3);

fprintf('--- การทำนายสถานะเครือข่ายใหม่ ---\n');

fprintf('ข้อมูล 1 (Latency: %.1f, Packet Loss: %.2f) -> ทำนาย: %s\n', X_new_1(1), X_new_1(2), getStatusString(label_1));

fprintf('ข้อมูล 2 (Latency: %.1f, Packet Loss: %.2f) -> ทำนาย: %s\n', X_new_2(1), X_new_2(2), getStatusString(label_2));

fprintf('ข้อมูล 3 (Latency: %.1f, Packet Loss: %.2f) -> ทำนาย: %s\n', X_new_3(1), X_new_3(2), getStatusString(label_3));

fprintf('\n');

% 4. แสดงผลด้วยกราฟ (Visualize the Results)

figure;

gscatter(X(:,1), X(:,2), Y, 'br', 'xo'); % พล็อตจุดข้อมูลจริง แยกสีตาม Class

hold on;

% พล็อตเส้นแบ่ง (Decision Boundary) ของ SVM

% สร้าง grid ของจุดเพื่อประเมินค่า

h = 0.1; % step size of the grid
```

```
[x1_grid, x2_grid] = meshgrid(min(X(:,1))-1:h:max(X(:,1))+1, ...
                              min(X(:,2))-0.1:h:max(X(:,2))+0.1);
grid_points = [x1_grid(:), x2_grid(:)];
[~, scores] = predict(svm_model, grid_points);
% scores(:,1) for class 0, scores(:,2) for class 1
% Decision boundary is where score for class 0 - score for class 1 = 0
contour(x1_grid, x2_grid, reshape(scores(:,2) - scores(:,1),
size(x1_grid)), [0 0], 'k-', 'LineWidth', 2);
% พล็อตจุดทำนายใหม่
scatter(X_new_1(1), X_new_1(2), 150, 'g*', 'DisplayName', sprintf('
ทำนาย: %s', getStatusString(label_1)));
scatter(X_new_2(1), X_new_2(2), 150, 'm^', 'DisplayName', sprintf('
ทำนาย: %s', getStatusString(label_2)));
scatter(X_new_3(1), X_new_3(2), 150, 'co', 'DisplayName', sprintf('
ทำนาย: %s', getStatusString(label_3)));
xlabel('Latency (ms)');
ylabel('Packet Loss Rate (%)');
title('การจำแนกสถานะเครือข่ายด้วย SVM');
legend('Normal', 'Problematic', 'Decision Boundary', 'Location',
'best');
grid on;
hold off;
fprintf('--- กราฟแสดงผลลัพธ์ถูกสร้างขึ้น (พร้อมเส้นแบ่งข้อมูล) ---\n');
```

```
% ฟังก์ชันช่วยแปลง label ตัวเลขเป็นข้อความ
function status_str = getStatusString(label)
    if label == 0
        status_str = 'Normal';
    else
        status_str = 'Problematic';
    end
end
##########################################################################
##########################################################################
########################################
```

การทำงานของโค้ด:

1. **สร้างข้อมูลตัวอย่าง:** เราสร้าง X ซึ่งเป็นเมทริกซ์ของ **คุณลักษณะ (features)** (Latency และ Packet Loss Rate) และ Y ซึ่งเป็น **label/class** (0 สำหรับ 'Normal', 1 สำหรับ 'Problematic')

2. **ฝึกโมเดล SVM:**
   - เราใช้ฟังก์ชัน **fitcsvm(X, Y, ...)** ซึ่งเป็นฟังก์ชันใน Machine Learning Toolbox ของ MATLAB สำหรับการสร้างโมเดล SVM
   - 'KernelFunction', 'linear' หมายถึงโมเดลจะพยายามหาเส้นตรงเพื่อแบ่งข้อมูลออกเป็นคลาสต่างๆ
   - 'Standardize', true เป็นการปรับขนาดข้อมูลให้เท่ากัน ซึ่งเป็นขั้นตอนสำคัญในการเตรียมข้อมูลสำหรับ AI เพื่อให้โมเดลทำงานได้ดีขึ้น

3. **ทำนายสถานะเครือข่ายใหม่:**
    - เราใช้ฟังก์ชัน **predict(svm_model, X_new)** เพื่อให้โมเดล SVM ที่เราฝึกมาแล้วทำนาย label (สถานะ) ของข้อมูลเครือข่ายใหม่ๆ ที่เราป้อนเข้าไป
    - getStatusString เป็นฟังก์ชันเล็กๆ ที่เราสร้างขึ้นมาเพื่อช่วยแปลงตัวเลข 0 หรือ 1 ให้เป็นข้อความ "Normal" หรือ "Problematic" เพื่อให้อ่านง่ายขึ้น

4. **แสดงผลด้วยกราฟ:**
    - **gscatter(X(:,1), X(:,2), Y, ...):** พล็อตจุดข้อมูลจริง โดยจะใช้สีและสัญลักษณ์ต่างกันตาม Class (Normal/Problematic)
    - **contour(...):** ส่วนนี้ซับซ้อนขึ้นเล็กน้อย ใช้สำหรับพล็อต **Decision Boundary (เส้นแบ่งการตัดสินใจ)** ของโมเดล SVM ซึ่งแสดงให้เห็นว่าโมเดลใช้เกณฑ์ใดในการแบ่งข้อมูลออกเป็น Class ต่างๆ
    - พล็อตจุดที่เราทำนายขึ้นมาใหม่ เพื่อให้เห็นว่าจุดเหล่านั้นตกอยู่ในพื้นที่ของ Class ใดตามการตัดสินใจของโมเดล

โมเดลและซอฟต์แวร์นี้:

- **แนวคิด Classification:** เข้าใจว่า AI สามารถจำแนกประเภทสิ่งต่างๆ ได้
- **การเตรียมข้อมูลสำหรับ AI:** การจัดรูปแบบข้อมูลเป็นคุณลักษณะและ Label
- **การใช้โมเดล SVM:** รู้จัก fitcsvm และ predict ใน MATLAB
- **Decision Boundary:** การทำความเข้าใจว่าโมเดลแบ่งข้อมูลออกเป็น Class อย่างไร
- **การประยุกต์ใช้ใน Data Communication:** เห็นภาพว่า AI สามารถช่วยในการตรวจจับปัญหาหรือจำแนก Traffic ในเครือข่ายได้อย่างไร

ในการสื่อสารข้อมูล (Data Communication) **Latency (ความหน่วง)** ของโมเดล AI หมายถึง **เวลาที่โมเดล AI ใช้ในการประมวลผลและตอบสนอง** ตั้งแต่ได้รับข้อมูลเข้า จนกระทั่งส่งผลลัพธ์ออกมา

ความหน่วง (Latency) ของโมเดล AI ใน Data Communication

ความหน่วง (Latency) ของโมเดล AI ในบริบทของการสื่อสารข้อมูล คือ **ระยะเวลาที่ใช้ตั้งแต่ข้อมูลเข้ามายังโมเดล AI จนกระทั่งโมเดลประมวลผลเสร็จสิ้นและส่งผลลัพธ์ออกมา**

- **อินพุต (Input):** ข้อมูลเครือข่ายเข้ามาที่โมเดล (เช่น แพ็กเก็ต, สถิติการใช้งาน)
- **ประมวลผล (Processing):** โมเดล AI (เช่น โมเดลจำแนกประเภท, โมเดลทำนาย) ใช้เวลาในการวิเคราะห์ข้อมูลนั้น
- **เอาต์พุต (Output):** โมเดลส่งผลลัพธ์การประมวลผล (เช่น การจำแนกประเภททราฟฟิก, การตรวจจับการโจมตี)

Impotent

ความหน่วงของโมเดล AI มีความสำคัญอย่างยิ่งในการสื่อสารข้อมูล เนื่องจาก:

1. **การตอบสนองแบบเรียลไทม์:** ในแอปพลิเคชันที่ต้องการการตอบสนองทันที เช่น ระบบเครือข่ายอัจฉริยะ (SDN), การตรวจจับการบุกรุกแบบเรียลไทม์ (Real-time Intrusion Detection Systems), หรือการจัดสรรทรัพยากรเครือข่ายแบบไดนามิก (Dynamic Resource Allocation) ความหน่วงที่สูงเกินไปจะทำให้ระบบทำงานล่าช้าและไม่มีประสิทธิภาพ
2. **ประสบการณ์ผู้ใช้:** สำหรับบริการที่ผู้ใช้โต้ตอบโดยตรง เช่น Video Conferencing หรือ Gaming ความหน่วงของ AI ที่เข้าไปช่วยปรับปรุงคุณภาพบริการ (QoS) จะต้องต่ำมาก มิฉะนั้นจะส่งผลเสียต่อประสบการณ์ของผู้ใช้

3. **ความน่าเชื่อถือของระบบ:** ในระบบควบคุมเครือข่ายหรือระบบความปลอดภัย การตัดสินใจที่ล่าช้าจากโมเดล AI อาจทำให้ไม่สามารถป้องกันปัญหาหรือการโจมตีได้อย่างทันท่วงที

ปัจจัยที่ส่งผลต่อ Latency ของโมเดล AI:

- **ความซับซ้อนของโมเดล:** โมเดลที่ซับซ้อน (เช่น Deep Neural Networks ขนาดใหญ่) มักจะใช้เวลาประมวลผลนานกว่าโมเดลที่เรียบง่าย
- **ขนาดของข้อมูล:** ข้อมูลที่ต้องประมวลผลยิ่งมาก ยิ่งใช้เวลานานขึ้น
- **ฮาร์ดแวร์:** ประสิทธิภาพของ CPU/GPU ที่ใช้ในการรันโมเดลมีผลอย่างมากต่อความเร็วในการประมวลผล
- **ประสิทธิภาพของโค้ด/ไลบรารี:** การเขียนโค้ดที่เหมาะสมและการใช้ไลบรารีที่ปรับปรุงมาอย่างดีช่วยลด Latency ได้

<u>สรุป</u>

คือ **Latency ของโมเดล AI ใน Data Communication คือ "ความเร็ว" ที่โมเดลตอบสนองต่อเหตุการณ์ในเครือข่าย ยิ่งต่ำยิ่งดี** เพื่อให้ระบบสามารถทำงานได้อย่างรวดเร็ว มีประสิทธิภาพ และตอบสนองต่อการเปลี่ยนแปลงได้ทันท่วงที