การใช้ TensorFlow เพื่อการพยากรณ์ทางธุรกิจ

อาณัติ ลีมัคเดช*

คณะพาณิชยศาสตร์และการบัญชี มหาวิทยาลัยธรรมศาสตร์

***Correspondence:** arnat@tbs.tu.ac.th

บทคัดย่อ

บทความนี้แนะนำการใช้แบบจำลองโครงข่ายประสาทเทียมเพื่อการพยากรณ์ข้อมูลทางธุรกิจ โดยแนะนำการใช้งานเฟรมเวิร์ค TensorFlow ที่พัฒนาโดย Google ทีมีจุดเด่นมากมาย เช่น ความสามารถในการสร้างแบบจำลองผ่านกระบวนการเรียนรู้ของเครื่องได้หลายหลาย การใช้ฮาร์ดแวร์ร่วมกันหลายเครื่องเพื่อประมวลผลข้อมูลขนาดใหญ่ และยังสามารถใช้ได้โดยไม่มีค่าใช้จ่าย ผู้อ่านจะเข้าใจการใช้งานเพื่อการพยากรณ์ข้อมูลทางธุรกิจผ่านแพลทฟอร์ม Colab ตลอดจนข้อจำกัดในการใช้งาน

คำสำคัญ: โครงข่ายประสาทเทียม, เฟรมเวิร์ค TensorFlow, สมการเชิงคณิตศาสตร์, การพยากรณ์ข้อมูลทางธุรกิจ

The Use of Tensorflow for Business Forecasting

Arnat Leemakdej*

Thammasat Business School, Thammasat University

***Correspondence:** arnat@tbs.tu.ac.th

Abstract

The article introduces the use of artificial neural network models for business data forecasting, highlighting the use of the TensorFlow framework developed by Google, which has many advantages, such as the ability to create models through various machine learning processes, the use of multiple hardware to process large data sets, and it is also available at no cost. Readers will understand how to use it for business data forecasting through the Colab platform as well as its limitations.

Keywords: Artificial Neural Network (ANN); TensorFlow framework; Mathematical equations; Business data forecasting

1. บทนำ

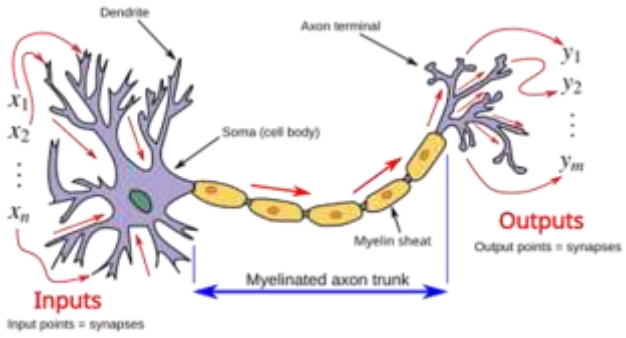
การพยากรณ์ (Forecasting) มีบทบาทสำคัญอย่างยิ่งต่อการตัดสินใจในหลากหลายอุตสาหกรรม ไม่ว่าจะเป็นการคาดการณ์ยอดขายในธุรกิจ การวิเคราะห์ทางการแพทย์ (Mao et al., 2011) การทำนายราคาหุ้นในตลาดทุน (Al Akashi, 2020) หรือการพยากรณ์สภาพอากาศ ล้วนอาศัยเทคนิคการพยากรณ์ทั้งสิ้น บทความนี้มีวัตถุประสงค์เพื่อนำเสนอแนวทางการใช้ TensorFlow ซึ่งเป็นเฟรมเวิร์คโอเพนซอร์สสำหรับสร้างแบบจำลองหรือโมเดลพยากรณ์ที่มีประสิทธิภาพผ่านกระบวนการเรียนรู้ของเครื่อง (Machine Learning) โดยอาศัยแนวคิดของโครงข่ายประสาทเทียม Artificial Neural Network (ANN)

ผู้อ่านจะเข้าใจหลักการทำงานของโครงข่ายประสาทเทียมและการแปลงให้เป็นสมการเชิงคณิตศาสตร์ การใช้งาน TensorFlow เพื่อสร้างแบบจำลองตั้งแต่กระบวนการเตรียมข้อมูล การกำหนดพารามิเตอร์หรือโครงสร้างของแบบจำลอง การฝึกแบบจำลอง (Train) และการพยากรณ์โดยใช้แบบจำลองที่สร้างขึ้น

2. โครงข่ายประสาทเทียม (Artificial Neural Network: ANN)

ANN เป็นแบบจำลองทางคณิตศาสตร์ที่เลียนแบบกลไกการเชื่อมต่อกันของเซลล์ประสาท (Neural) ของสมองมนุษย์ การที่มนุษย์มองเห็นสิ่งมีชีวิตตัวหนึ่งแล้วบอกได้ว่าเป็นสิ่งมีชีวิตใดได้ทันที เป็นเพราะมนุษย์มีการเรียนรู้ และสร้างฟังก์ชันสำหรับแปลความหมายของภาพที่เห็นเก็บไว้แล้ว เมื่อได้รับสัญญาณภาพจากดวงตา เห็นว่ามี 4 ขา หูตั้ง ฯลฯ จึงบอกได้ว่าสิ่งมีชีวิตที่เห็นคือสุนัข เป็นต้น กลไกลักษณะนี้ถูกนำมาจำลองโดยใช้หลักคณิตศาสตร์ นักวิเคราะห์จะทำการนำข้อมูลในอดีต หรือข้อมูลที่ถูกต้องมาป้อนเป็นข้อมูลตั้งต้นให้แบบจำลองเรียนรู้ (Train) เมื่อแบบจำลองได้ถูก Train แล้ว ผู้วิเคราะห์จะสามารถใส่ข้อมูลใหม่เข้าสู่แบบจำลองและได้รับผลลัพธ์การพยากรณ์กลับออกมาได้

การทำงานของโครงข่ายเซลล์ประสาท คือการที่เซลล์ประสาทแต่ละตัวรับสัญญาณ Input ผ่าน Dendrites เข้าสู่ Cell Body และส่งสัญญาณ Output ต่อไปยังเซลล์ประสาทตัวอื่นผ่าน Axon ดังแสดงภาพที่ 1



ภาพที่ 1 การทำงานของเซลล์ประสาท (Wikimedia commons, 2025)

สามารถสร้างเป็นสมการเชิงคณิตศาสตร์ที่เป็นตัวแทนของเซลล์ประสาท 1 ตัวได้ดังนี้

(1) กำหนดให้ x คือตัวแปร Input หรือสัญญาณที่ส่งเข้าเซลล์ประสาท สัญญาณแต่ละตัวนั้นมีความสำคัญไม่เท่ากันจึงมีการถ่วงน้ำหนักโดยนำไปคูณกับค่าคงที่ w

x1 => w1 * x1

x2 => w2 * x2

(2) การรวมสัญญาณ Input คือการรวมสัญญาณถ่วงน้ำหนักเข้าด้วยกัน แต่จะมีการเพิ่มค่าคงที่เข้าไปในสมการอีก 1 ตัวเพื่อให้สมการมีความยืดหยุ่นและรองรับข้อมูลที่ซับซ้อนได้มากขึ้น เช่นการรับมือกับกรณีที่ Input มีค่าเท่ากับ

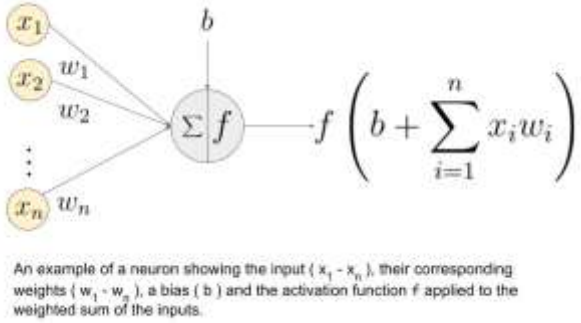
0 ทุกตัว หากไม่มีค่าคงที่นี้จะส่งผลให้คำนวณ Output ออกมาเป็น 0 เสมอ ทำให้ค่าน้ำหนักที่คำนวณได้คลาดเคลื่อนจากที่ควรเป็น ค่าคงที่นี้เรียกว่าไบแอส (bias: b)

(w1 * x1) + (w2 * x2) + b

(3) แปลงผลรวมในข้อ (2) ไปเป็นผลลัพธ์หรือ Output โดยนำไปผ่านฟังก์ชันทางคณิตศาสตร์ เรียกฟังก์ชันในการแปลงนี้ว่า Activation Function ซึ่งผู้วิเคราะห์สามารถเลือกใช้ Activation Function แตกต่างกัน เพื่อให้ค่าผลลัพธ์อยู่ในช่วงที่ต้องการได้ เช่นการตอบคำถามในลักษณะ ใช่หรือไม่ใช่ ก็ควรให้ผลลัพธ์ออกมาเป็น 0 หรือ 1 เท่านั้น เป็นต้น

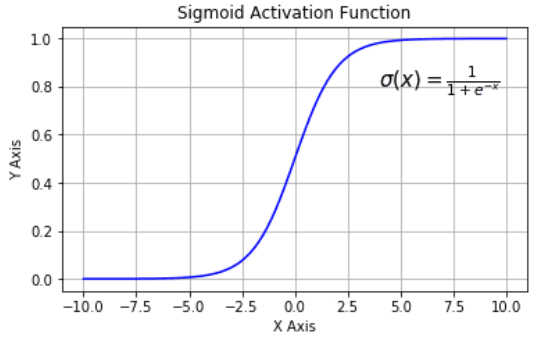
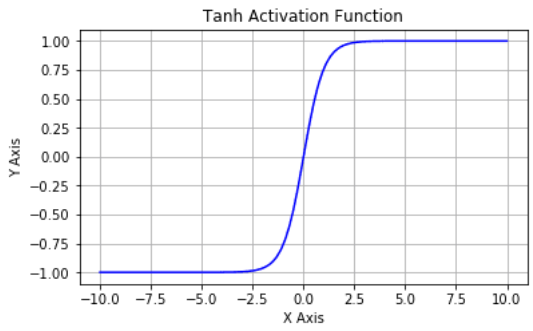
y = f (w1 * x1 + w2 * x2 + b)

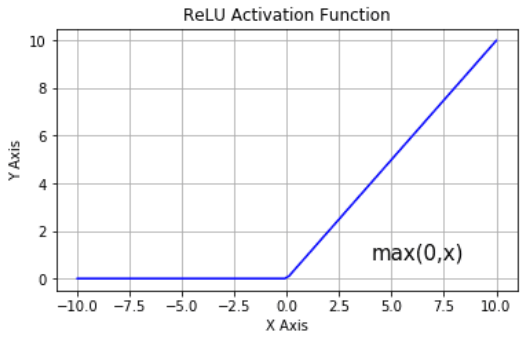
รูปแบบของสมการคณิตศาสตร์ที่มีจำนวน input เป็นจำนวน n ตัว สามารถแสดงได้ดังรูปที่ 2



ภาพที่ 2 สมการคณิตศาสตร์จำลองการทำงานของเซลล์ประสาท 1 ตัว (learnopencv, 2017)

Activation Function ที่สำคัญและถูกใช้งานอย่างแพร่หลายได้แก่

ชื่อฟังก์ชัน	แผนภาพแสดงลักษณะ Output	คำอธิบาย
Sigmoid (Rumelhart, D. E et al., 1986)		$f(x)=\frac{1}{(1+e)^{-x}}$ ผลลัพธ์จะอยู่ในช่วง 0 ถึง 1 เท่านั้น
Hyperbolic tangent (Rumelhart, D. E et al., 1986) (Tanh)		$f(x)=\tanh(x)$ ผลลัพธ์จะอยู่ในช่วง -1 ถึง 1 เท่านั้น

ชื่อฟังก์ชัน	แผนภาพแสดงลักษณะ Output	คำอธิบาย
Rectified Linear Unit (Nair, V., & Hinton, G. E. (2010) (ReLU)		$f(x) = \max(0, x)$ เป็นฟังก์ชันที่ถูกใช้มากที่สุด ผลลัพธ์จะเป็นบวกเท่านั้น

ภาพที่ 3 แสดงการเปรียบเทียบ Activation Function ที่ใช้งานแพร่หลาย (Learnopencv, 2017)

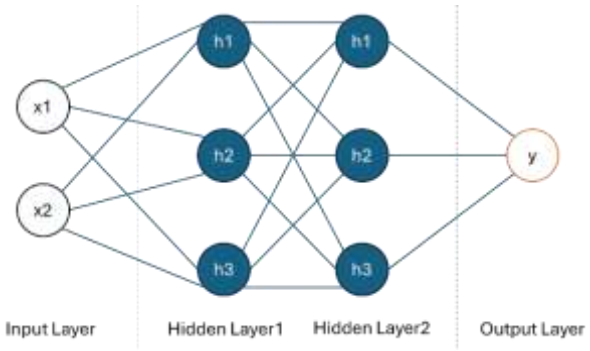
การจะเลือกใช้ Activation แบบใด ขึ้นอยู่กับว่าต้องการแปรความหมาย Input ไปเป็น Output ลักษณะใด

• Sigmoid จะใช้ในกรณีต้องการ Output เป็นความน่าจะเป็น (Probability) หรือตอบคำถามว่าใช่หรือไม่ใช่ (1= ใช่, 0=ไม่ใช่)

• Tanh จะนิยมนำไปใช้ในจำแนก classification ที่มี 2 คลาส หรือตอบคำถามทิศทางเช่น หุ้นจะขึ้น หรือจะตก เป็นต้น

• ReLU ใช้ในการประมาณค่าทั่วไป มีจุดเด่นในการทำงานได้เร็วและแก้ปัญหา

ที่กล่าวมาข้างต้นเป็นการทำงานของเซลล์ประสาทเพียง 1 ตัว หรือ 1 Node ในแบบจำลอง หากจำลองการเชื่อมต่อกันของเซลล์ประสาทโดย Output ของเซลล์ประสาทตัวหนึ่งจะกลายเป็น Input ของเซลล์ประสาทอีกตัวต่อเนื่องกันไป ตัวอย่างโครงข่ายที่เกิดขึ้นจากการเชื่อมต่อ Node หลายตัวเข้าด้วยกันแสดงดังภาพที่ 4



ภาพที่ 4 แสดงลักษณะการเชื่อมต่อของ Neural Network

โครงข่ายสามารถแบ่งเป็น Layer ย่อย ๆ ได้เป็น

(1) Input Layer สำหรับรับค่าตัวแปร Input ผู้ใช้ต้องกำหนดว่ามีกี่ตัวแปร Input โดยปกติจำนวน Input Node จะเท่ากับจำนวนตัวแปรต้นในแบบจำลอง

(2) Output Layer สำหรับส่งออกค่าตัวแปร Output ผู้ใช้ต้องกำหนดว่ามีกี่ตัวแปร Output โดยปกติอาจมี 1 ตัวแปร หรือมากกว่าก็ได้ ตามสมมติฐานของแบบจำลอง

(3) Hidden Layer ส่วนประสานระหว่าง Input Layer และ Output Layer ซึ่งสามารถมีได้มากกว่า 1 ชั้น แล้วแต่ความซับซ้อนของปัญหา (โดยปกติประมาณ 2 ชั้นก็เพียงพอ) ในชั้นนี้มักใช้ Activation Function เป็น ReLU โดยมักมี

จำนวน Node ในแต่ละชั้นอยู่ระหว่างจำนวน Input Node และ Output Node หรือหากมากกว่าไม่เกินสองเท่าของจำนวน Input Node

กระบวนการฝึก (Training) คือกระบวนการที่ทำให้ทราบค่าน้ำหนักและค่าไบแอสทั้งหมดในโครงข่ายที่ทำให้เกิดความคลาดเคลื่อนรวม (Loss Function หรือ Objective Function) ต่ำที่สุด ความคลาดเคลื่อนนี้คำนวณจากความแตกต่างระหว่างผลลัพธ์ที่แบบจำลอง Machine Learning ทำนายได้ (predicted values) กับผลลัพธ์ที่แท้จริง (actual values) Loss Function ที่นิยมใช้มี 2 สูตร ได้แก่

Mean Squared Error (MSE): คำนวณค่าเฉลี่ยของผลต่างยกกำลังสองระหว่าง predicted values และ actual values

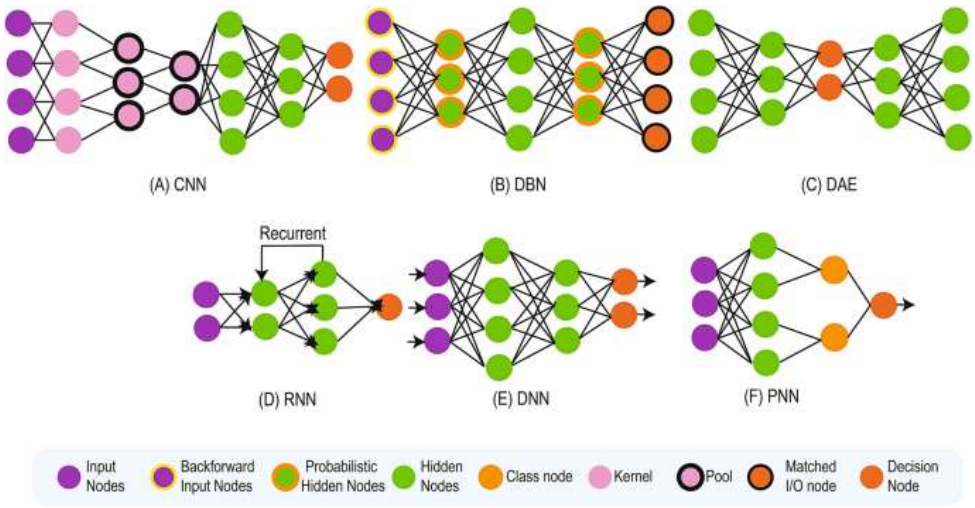
$$MSE = (1/n) * \Sigma(y\_true - y\_pred)^2$$

Mean Absolute Error (MAE): คำนวณค่าเฉลี่ยของผลต่างสัมบูรณ์ระหว่าง predicted values และ actual values

$$MAE = (1/n) * \Sigma|y\_true - y\_pred|$$

การวนลูปปรับค่าน้ำหนักและไบแอสซ้ำ ๆ โดยมีเป้าหมายทำให้ Loss Function มีค่าต่ำที่สุด ที่สุดแล้วจะทำให้ทราบค่าน้ำหนักในแต่ละเส้นและไบแอสในแต่ละโหนด การนำไปพยากรณ์คือการส่ง Input เข้าสู่แบบจำลองที่ฝึกเสร็จแล้ว (Trained) เมื่อทราบค่าน้ำหนักและไบแอสทั้งหมดจึงได้ผลลัพธ์ y ออกมาในทันที

รูปแบบการเชื่อมต่อของ Node เป็นไปได้หลายแบบ ทำให้เกิดประเภทของโครงข่ายที่มีชื่อเรียกเฉพาะแตกต่างกันและมีจุดเด่นจุดด้อยในการนำไปใช้งานแตกต่างกันออกไป ดังแสดงภาพที่ 5



ภาพที่ 5 แสดงการเชื่อมต่อของ Neural Network รูปแบบต่าง ๆ (Noor, M. B. T. et al., 2020)

ประเภทของโครงข่ายที่สำคัญและถูกใช้งานอย่างแพร่หลายได้แก่

DNN (Deep Neural Network): โครงข่ายประสาทเทียมเชิงลึก

- **ลักษณะ:** โครงสร้างพื้นฐานที่สุดของโครงข่ายประสาทเทียม มีหลายชั้น (layer) เชื่อมต่อกันแบบ feedforward (ข้อมูลไหลไปข้างหน้า)
- **การทำงาน:** เรียนรู้ความสัมพันธ์ระหว่างข้อมูลนำเข้าและข้อมูลส่งออกผ่านการปรับค่าน้ำหนัก (weight) และค่าไบแอส (bias) ในแต่ละชั้น
- **จุดเด่น:** สามารถเรียนรู้ฟังก์ชันที่ซับซ้อนได้ดี เมื่อมีข้อมูลจำนวนมาก และมีจำนวนชั้นที่เหมาะสม

- **ข้อจำกัด:** ไม่เหมาะกับข้อมูลแบบ sequential data เช่น ข้อความ หรือ เสียง
- **การใช้งาน:** การจำแนกประเภททั่วไป การทำนาย การวิเคราะห์ข้อมูลที่ไม่เป็นลำดับ

RNN (Recurrent Neural Network): โครงข่ายประสาทเทียมแบบวนซ้ำ

- **ลักษณะ:** มีการเชื่อมต่อแบบวนซ้ำ (recurrent connection) ทำให้สามารถเก็บข้อมูลในอดีตไว้ในหน่วยความจำ (memory) เพื่อนำมาใช้ในการประมวลผลข้อมูลปัจจุบัน
- **การทำงาน:** ประมวลผลข้อมูลทีละส่วนตามลำดับ โดยใช้ข้อมูลที่ผ่านมาและข้อมูลปัจจุบันในการทำนายหรือจำแนก
- **จุดเด่น:** เหมาะกับข้อมูลลำดับ (sequential data) เช่น ข้อความ เสียง วิดีโอ ข้อมูลอนุกรมเวลา
- **ข้อจำกัด:** ไม่เหมาะกับข้อมูลที่มีระยะเวลาหรือความยาวมาก ๆ
- **การใช้งาน:** การแปลภาษา การสร้างข้อความ การวิเคราะห์ความรู้สึก

CNN (Convolutional Neural Network): โครงข่ายประสาทเทียมแบบคอนโวลูชัน

- **ลักษณะ:** ใช้ layer ที่เรียกว่า convolutional layer เพื่อดึงคุณลักษณะ (feature) จากข้อมูลโดยการเลื่อนตัวกรอง (filter) ไปทั่วข้อมูล
- **การทำงาน:** เรียนรู้รูปแบบ (pattern) ที่ซับซ้อนในข้อมูล โดยเฉพาะอย่างยิ่งข้อมูลภาพ โดยการใช้ convolutional layer และ pooling layer
- **จุดเด่น:** เหมาะกับข้อมูลที่มีโครงสร้างเชิงพื้นที่ (spatial structure) เช่น ภาพ หรือ เสียง (ในรูปแบบ spectrogram) มีประสิทธิภาพในการดึงคุณลักษณะที่สำคัญ
- **ข้อจำกัด:** ไม่เหมาะกับข้อมูลแบบ Unstructured data เช่นข้อความข่าว หรือ อีเมล
- **การใช้งาน:** การตรวจจับวัตถุ การจำแนกประเภทภาพ การประมวลผลภาพทางการแพทย์

3. ทำความรู้จัก TensorFlow

3.1 TensorFlow คืออะไร

TensorFlow คือ เฟรมเวิร์คโอเพนซอร์สสำหรับพัฒนา AI เช่นการสร้างแบบจำลองด้วย Machine Learning (ML) โดยใช้ภาษา Python ที่มีจุดเด่นมากมาย เช่น ความสามารถในการรองการสร้างแบบจำลองได้หลายหลาย การใช้ฮาร์ดแวร์ร่วมกันหลายเครื่องเพื่อประมวลผลข้อมูลขนาดใหญ่ และยังสามารถใช้ได้โดยไม่มีค่าใช้จ่าย ผู้ใช้งานมีทางเลือกในการติดตั้ง TensorFlow ที่เครื่องคอมพิวเตอร์ของตนเอง หรือใช้งานแบบออนไลน์ผ่านเว็บไซต์ https://colab.research.google.com/

ผู้ริเริ่มพัฒนา TensorFlow คือ Google Brain Team โดยในตอนแรกมีจุดประสงค์เพื่อใช้ภายในบริษัท Google จนต่อมาได้เปิดให้สาธารณชนได้ใช้งานภายใต้ Apache License 2.0 เมื่อวันที่ 9 พฤศจิกายน ค.ศ. 2015

3.2 รูปแบบคำสั่งใช้งาน TensorFlow

ผู้ใช้งานสามารถใช้ชุดคำสั่งในภาษา Python เพื่อดำเนินการสิ่งต่าง ๆ ในกระบวนการสร้างแบบจำลองได้ โดย TensorFlow ได้จัดเตรียม API หรือชุดคำสั่งสำหรับการตั้งค่าและกำหนดการทำงานต่าง ๆ ใน ANN เช่น จัดการโครงสร้าง Model และการกำหนด Layer ต่าง ๆ ชุดคำสั่งที่สำคัญได้แก่

การกำหนดโครงสร้างแบบจำลอง

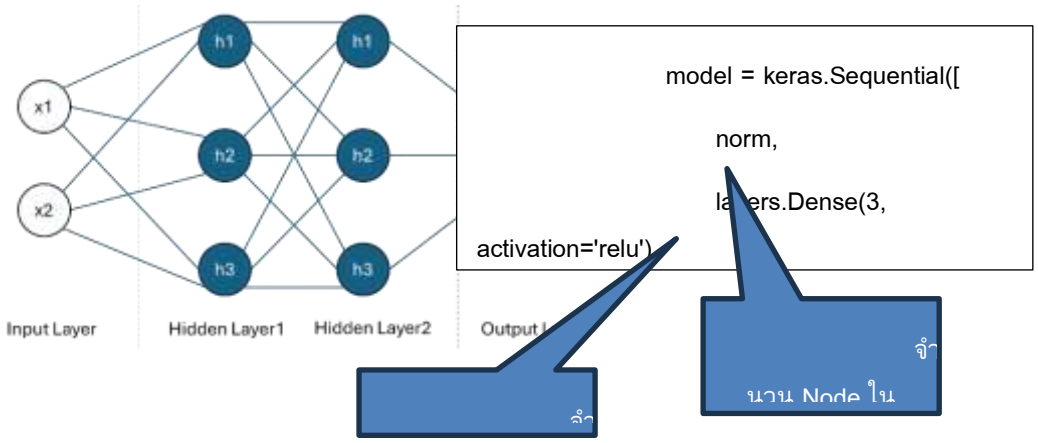
ผู้วิเคราะห์ต้องเป็นผู้กำหนดจำนวนตัวแปรต้น (Input) จำนวนตัวแปรตาม (Output) และจำนวน Node และจำนวนชั้น ของ Hidden Layer ด้วยตนเอง

รูปแบบคำสั่งพื้นฐานคือ layers.Dense(n, activation='relu') (tf.keras.layers.Dense, 2025) มีพารามิเตอร์ที่สำคัญคือ

- n จำนวนโหนดใน layer นั้น ๆ

- activation ใช้ระบุ Activation Function ในตัวอย่างคือ ReLU
  (relu: ReLU / sigmoid: Sigmoid / tanh: Tanh)

ตัวอย่างการแปลงโครงสร้างด้านล่าง จะใช้คำสั่งลักษณะดังนี้



ภาพที่ 6 แสดงตัวอย่างการตั้งค่าโครงสร้างแบบจำลองของ TensorFlow

การกำหนดค่าการฝึก (Training Configuration)

ทำได้โดยใช้ model.compile (tf.keras.Model, 2025) ที่มีพารามิเตอร์สำคัญคือ

- optimizer ระบุ Optimizer ที่ใช้ คำว่า Optimizer หมายถึง อัลกอริทึมที่ใช้ในกระบวนการปรับค่าน้ำหนักและไบแอสซึ่งมีหลายวิธีการ โดยปกติมักใช้อัลกอริทึมชื่อ Adam (Kingma, Diederik P., and Jimmy Lei Ba, 2015)
- loss ระบุ Loss Function
- metrics ระบุวิธีการวัดประสิทธิภาพของโมเดล เช่นการใช้ R-Squared: R2

```
# Compile model
model. compile (optimizer='adam',
                loss='mean_squared_error',
                metrics=['accuracy'])
```

การฝึกแบบจำลอง

ใช้คำสั่ง model.fit (tf.keras.Model, 2025) มีพารามิเตอร์ที่สำคัญคือ

- train_features ข้อมูล input สำหรับฝึก (x_train)
- train_labels ข้อมูล output สำหรับฝึก (y_train)
- epochs จำนวนลูปในการวนปรับค่า โดยปกติจะใช้ค่าเริ่มต้นประมาณ 50-100 ผู้ใช้งานอาจใช้ฟีเจอร์ Early Stopping คือการหยุด Train เมื่อมีค่า Loss ไม่ดีขึ้นหรือแย่ลง เป็นจำนวนกี่ epochs ติดต่อกัน (tf.keras.callbacks.EarlyStopping, 2025)

```
history_dnn = dnn_model.fit(
    train_features,
    train_labels,
    validation_split=0.2,
    verbose=0, epochs=100)
```

การวัดผลแบบจำลอง

ใช้คำสั่ง model.evaluate (tf.keras.Model, 2025) มีพารามิเตอร์ที่สำคัญคือ

- test_features ข้อมูล input สำหรับทดสอบ (x_test)
- test_labels ข้อมูล output สำหรับทดสอบ (y_test)

```
model. evaluate (test_features, test_labels, verbose=0)
```

การพยากรณ์โดยใช้แบบจำลอง

ใช้คำสั่ง model.predict (tf.keras.Model, 2025) โดยพารามิเตอร์คือค่า input

```
model. predict (x1,x2,x3, … ,xn)
```

3.3 เงื่อนไขและข้อจำกัดในการใช้งาน TensorFlow

TensorFlow เฟรมเวิร์คโอเพนซอร์ส จึงสามารถติดตั้งใช้งานได้ฟรีโดยไม่มีค่าใช้จ่าย อย่างไรก็ตามผู้ใช้งานยังมีข้อที่ต้องพิจารณาบางประการ เช่น

- ผู้ใช้งานควรมีพื้นฐานภาษา Python จึงจะเข้าใจรูปแบบการปรับตั้งค่าต่าง ๆ ได้
- รูปแบบคำสั่งของ TensorFlow แต่ละเวอร์ชัน มีความแตกต่างกัน จึงต้องเลือกเวอร์ชันใช้งานให้ถูกต้อง และอาจต้องปรับแก้ไข Code ใหม่ทั้งหมดกรณีที่ต้องการใช้งานฟีเจอร์ในเวอร์ชันที่สูงกว่า
- กรณีที่ติดตั้งใช้งานที่เครื่องคอมพิวเตอร์ของตนเอง ควรจะมีประสิทธิของเครื่องที่สูงเพียงพอ เช่น การมีหน่วยความจำอย่างน้อย 12 GB หรือการมีการ์ดจอช่วยในการประมวลผล (Install TensorFlow with pip, 2025) นอกจากนี้ผู้ใช้ยังต้องมีความรู้ในการจัดการ Library ของ Python ที่ติดตั้งบนเครื่องเป็นอย่างดี เพราะอาจเกิดความไม่สอดคล้องของเวอร์ชันหรือติดตั้งไม่สมบูรณ์ จนทำให้ไม่สามารถใช้งานได้
- ในกรณีที่ใช้งานออนไลน์ผ่าน Colab โดยไม่มีค่าใช้จ่าย จะมีขนาดทรัพยากรจำกัดคือ vCPU 2 หน่วย, Ram 12 GB และ Hard Disk 100 GB หากต้องการทรัพยากรที่มากขึ้น ต้องจ่ายค่าสมาชิกรายเดือน

4. ขั้นตอนการสร้างแบบจำลองโดยใช้ TensorFlow

4.1 การเตรียมข้อมูลสำหรับใช้สร้างแบบจำลอง

การเตรียมข้อมูลมีความสัมพันธ์กับสมมติฐานที่ใช้ในการสร้างแบบจำลอง ตัวอย่างเช่น การพยากรณ์ปริมาณการใช้น้ำประปาของแต่ละหมู่บ้านแบบ Non-Linear โดยมีข้อมูลในอดีตเป็นอนุกรมเวลาของปริมาณการใช้น้ำ(ลูกบาสก์เมตร) และจำนวนผู้ใช้น้ำ (ราย) ซึ่งได้จากการเก็บข้อมูลการใช้น้ำประปาในพื้นที่จริง

สมมติฐานในการสร้างแบบจำลองคือ ปริมาณการใช้น้ำ (q) ขึ้นอยู่กับปริมาณการใช้น้ำในอดีต 3 ช่วงเวลาก่อนหน้า (q1 q2 q3) จำนวนผู้ใช้น้ำ (u) และจำนวนผู้ใช้น้ำในอดีต 3 ช่วงเวลาก่อนหน้า (u1 u2 u3) สามารถเขียนเป็นฟังก์ชันทางคณิตศาสตร์คือ

$$q=f\ (q1,q2,q3,u,u1,u2,u3)$$

จากข้อมูล q, u จะถูกนำมาจัดเตรียมเป็น CSV กำหนดชื่อไฟล์คือ “house_pq.csv” ซึ่งมีคอลัมน์ข้อมูลเป็น q, q1, q2, q3, u, u1, u2, u3 ดังแสดงภาพที่ 7 เพื่อนำไปใช้สร้างแบบจำลองต่อไป

	A	B	C	D	E	F	G	H
1	q	q1	q2	q3	u	u1	u2	u3
2	1287795	1193320	1228374	1163164	74398	74098	73710	73476
3	6248	5876	6407	5801	659	647	641	635
4	5095	4723	4906	4458	391	391	391	389
5	33756	30394	30410	30755	2314	2300	2293	2285
6	172749	160637	164403	147025	12091	12002	11980	11863
7	243415	226572	236822	221847	16044	15984	15859	15808
8	20021	17289	18777	16339	1420	1421	1412	1401
9	7013	6243	6981	5967	592	585	580	578
10	11567	9201	10187	8441	888	880	870	867
11	5676	5300	5608	4958	462	471	476	474
12	34688	33477	35830	34359	2706	2697	2683	2653
13	6644	6406	6592	6797	585	583	581	572

ภาพที่ 7 แสดงตัวอย่างการเตรียมข้อมูลไฟล์ CSV สำหรับสร้างแบบจำลอง

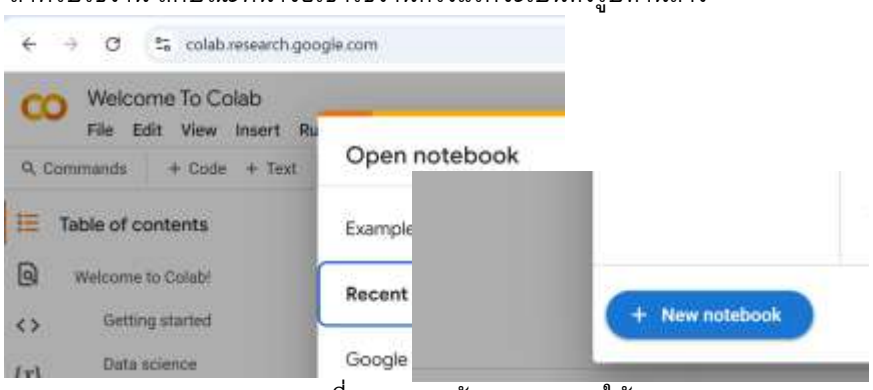
4.2 การติดตั้งใช้งาน TensorFlow

การติดตั้งใช้งานสามารถทำได้ 2 แบบ คือ

- การติดตั้งที่เครื่องคอมพิวเตอร์ของตนเอง
  - ติดตั้ง Python
  - ใช้คำสั่ง pip install tensorflow (สำหรับ CPU-only) หรือ pip install tensorflow-gpu (สำหรับ GPU support)
  - ใช้งานผ่าน IDE เช่น VS Code โดยต้องติดตั้ง Plugins ภาษา Python
- การใช้งานออนไลน์ผ่านเว็บไซต์ https://colab.research.google.com/
  - Colab คือแพลตฟอร์มออนไลน์ที่ใช้อย่างแพร่หลายในวงการ data science ซึ่งผู้ใช้สามารถใช้งานภาษา Python และ TensorFlow ได้โดยไม่ต้องติดตั้งโปรแกรมใดเพิ่มเติม
  - หน้าจอทำงานมีลักษณะเป็น IDE ที่รองรับการเขียนภาษา Python ได้ทันทีในลักษณะ Jupyter Notebook (รูปแบบการแบ่ง Code ออกเป็นส่วน ๆ แทรกคำอธิบายและกดประมวลผลทีละส่วนได้)
  - เชื่อมต่อกับ Google Drive เพื่อนำเข้าหรือบันทึกไฟล์ได้โดยง่าย
  - ด้วยข้อดีต่าง ๆ จึงแนะนำให้ใช้งาน TensorFlow โดยมีขั้นตอนดังนี้

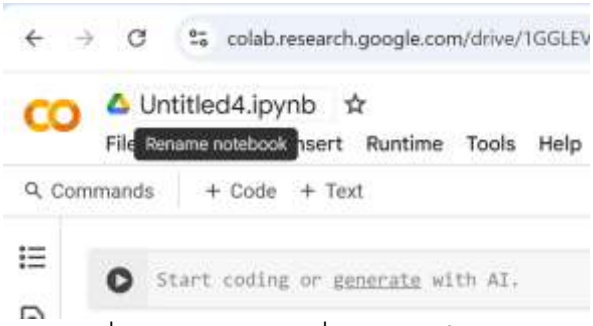
4.3 เริ่มใช้งาน Colab

- เข้าใช้งานที่ https://colab.research.google.com/ และกด + New notebook เพื่อเริ่มสร้าง Session สำหรับใช้งาน ลักษณะหน้าจอเข้าใช้งานครั้งแรกจะเป็นดังรูปด้านล่าง



ภาพที่ 8 แสดงหน้าแรกของการใช้งาน Colab

- ตั้งชื่อ Session ตามต้องการ ที่ส่วนบนของหน้าจอ ดังแสดงในรูปด้านล่าง โดยการกดที่ชื่อ Untitled…ipynb เพื่อแก้ไข



ภาพที่ 9 แสดงส่วนบันทึกชื่อ Session ใน Colab

- วิธีการใช้งานโดยทั่วไปคือ สามารถพิมพ์ Code ภาษา Python ได้ที่ช่อง Start coding และกด  เพื่อรัน Code ที่พิมพ์ลงในช่องดังกล่าว
- สามารถกด + Code เพื่อเพิ่มช่องสำหรับพิมพ์ Code เพิ่มเติม และกด  แยกกันแต่ละช่องได้ โดยตัวแปรที่ประกาศในช่องก่อนหน้าหากผ่านการรันแล้ว จะถูกนำมาใช้ในช่องถัด ๆ ไปได้

4.4 เลือกเวอร์ชันของ TensorFlow และ Import Library ที่จำเป็น

โดยปกติในแพลทฟอร์ม Colab จะมี TensorFlow ติดตั้งเป็น Library พื้นฐานมาให้อยู่แล้ว สามารถใช้งานออนไลน์ได้ทันที อย่างไรก็ตามเวอร์ชันของ TensorFlow ที่ถูกติดตั้งจะถูกอัพเดทเวอร์ชันเพิ่มขึ้นเรื่อย ๆ ซึ่งจะมีรูปแบบคำสั่งแตกต่างกันออกไป ในบทความนี้จะใช้ TensorFlow เวอร์ชัน 2.15.0 จึงต้องเริ่มจากกระบวนการกำหนดเวอร์ชันก่อนเป็นลำดับแรก

- ระบุเวอร์ชัน TensorFlow เป็น 2.15.0 โดยพิมพ์คำสั่งด้านล่าง และกดรัน

```
!pip install tensorflow==2.15.0
```

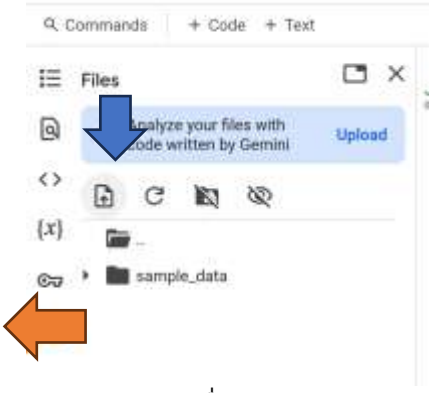
- อิมพอร์ทไลบรารี่ที่จำเป็น

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

# Make NumPy printouts easier to read.
np.set_printoptions(precision=3, suppress=True)

import tensorflow as tf

from tensorflow import keras
from keras import layers
from keras import backend as K

print(tf.__version__)
```

เมื่อกดรันจะได้ผลลัพธ์เป็นการแสดงตัวเลข 2.15.0 ซึ่งเป็นเวอร์ชันของ TensorFlow ที่เลือก

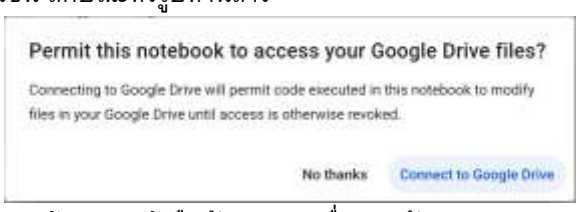
4.5 ตั้งค่าการเชื่อมต่อ Google Drive

- กดที่สัญลักษณ์โฟลเดอร์ด้านซ้ายสุด
- กดที่สัญลักษณ์ Google Drive



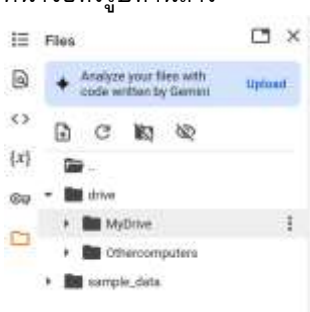
ภาพที่ 10 แสดงส่วนสำหรับเชื่อมต่อ Google Drive ของ Colab

- อนุญาตให้เว็บไซต์ Colab เข้าถึง Google Drive ได้ โดยกดที่ปุ่ม “Connect to Google Drive” ในกล่องข้อความที่แสดงขึ้น ลักษณะดังรูปด้านล่าง



ภาพที่ 11 แสดงข้อความแจ้งยืนยันอนุญาตเชื่อมต่อกับ Google Drive ของ Colab

- หลังเชื่อมต่อ Google Drive เรียบร้อยจะปรากฏ Directory drive/MyDrive ที่มีไฟล์ต่าง ๆ บน Google Drive อยู่ภายใต้ ตัวอย่างหน้าจอดังรูปด้านล่าง



ภาพที่ 12 แสดงตัวอย่างโฟลเดอร์ที่แสดงผล หลังเชื่อมต่อ Google Drive ของ Colab

4.6 นำเข้าข้อมูลและแบ่งช่วงข้อมูลสำหรับฝึกและทดสอบ

- ระบุที่อยู่ของไฟล์ CSV ภายใน Drive ที่จัดเตรียมไว้แล้วให้ถูกต้อง ในกรณีนี้คือไฟล์ “house_pq.csv” ที่เตรียมไว้ในขั้นตอน 4.1

```
# ระบุชื่อไฟล์และโฟลเดอร์ใน Google Drive ให้ถูกต้อง
url = '/content/drive/MyDrive/water/house_pq.csv'
raw_dataset = pd.read_csv(url,na_values='?', comment='\t',sep=',', skipinitialspace=True)
dataset = raw_dataset.copy()

# ลบข้อมูลที่เป็น n/a ออก
dataset = dataset.dropna()
```

```
# แบ่งข้อมูลสำหรับ Train 80% ข้อมูลสำหรับ Test คือที่เหลือ
train_dataset = dataset.sample(frac=0.8, random_state=0)
test_dataset = dataset.drop(train_dataset.index)
train_features = train_dataset.copy()
test_features = test_dataset.copy()

# แยกตัวแปร input และ output ออกจากกัน โดย input อยู่ที่ features และ output อยู่ที่ labels
train_labels = train_features.pop('q')
test_labels = test_features.pop('q')

# Normalize ข้อมูลให้อยู่ในช่อง 0 ถึง 1 เพื่อทำให้สร้างแบบจำลองได้รวดเร็วและแม่นยำ
normalizer = tf.keras.layers.Normalization()
normalizer.adapt(train_features)
```

4.7 สร้างฟังก์ชันสำหรับคำนวณความแม่นยำ R^2

ฟังก์ชันนี้เป็นฟังก์ชันที่สร้างขึ้นมาเพิ่มเติม โดยแสดงให้เห็นว่าตัวแปรอิสระในแบบจำลองสามารถอธิบายความแปรปรวนของตัวแปรตามได้มากแค่ไหน (Cohen, J., 1988) มีค่าอยู่ระหว่าง 0 ถึง 1 โดยยิ่งเข้าใกล้ 1 จะหมายถึงแบบจำลองสามารถทำนายผลลัพธ์ได้แม่นยำ และอธิบายความสัมพันธ์ในข้อมูลได้ดี

```
def coeff_determination(y_true, y_pred):
  SS_res =  K.sum(K.square( y_true-y_pred ))
  SS_tot = K.sum(K.square( y_true - K.mean(y_true) ) )
  return ( 1 - SS_res/(SS_tot + K.epsilon()) )
```

4.8 กำหนดโครงสร้างแบบจำลอง และการกำหนดค่าการฝึก

ใช้ keras.Sequential (tf.keras.Sequential, 2025) และใส่ข้อมูลทีละ layers จาก Code ด้านล่าง

- ชั้นแรกเป็น input layer โดยใช้ข้อมูลที่ทำการ Normalize แล้ว จำนวนตัวแปร input ของตัวอย่าง คือ 7 ตัวแปร
- ชั้นถัดมาเป็นการระบุ hidden layer โดยมี 2 ชั้น แต่ละชั้นมีจำนวน Node คือ 14 (โดยปกติไม่ควรมีจำนวนเกิน 2 เท่าของตัวแปร input) และ Activation Function เป็น ReLU
- ชั้นสุดท้ายคือ output layer ในกรณีตัวอย่างมี output ตัวเดียว
- ใช้ Loss Function เป็น MSE
- Optimizer เป็น Adam
- วัดความแม่นยำโดยใช้ฟังก์ชัน coeff_determination หรือหมายถึงค่า R^2

```
model = keras.Sequential([
  normalizer,
  layers.Dense(14, activation='relu'),
  layers.Dense(14, activation='relu'),
  layers.Dense(1)
])
model.compile(loss="mean_squared_error",
```

4.9 ฝึกแบบจำลอง

ใช้ model.fit

- ระบุ epochs เป็นจำนวน 100 รอบ เมื่อรันคำสั่งจะต้องรอสักครู่ให้ TensorFlow ทำกระประมวลผลเพื่อหาค่าน้ำหนักและไบแอสดังที่กล่าวไว้ช่วงแรกของบทความ
- หลังจากสร้างแบบจำลองเสร็จให้แสดงค่าต่าง ๆ ในช่วง epoch หลัง ๆ ค่า loss มักจะลดลงเรื่อย ๆ และค่า R^2 จะดีขึ้นเรื่อย ๆ เช่นกัน ตัวอย่างผลลัพธ์การแสดงค่าเป็นดังรูปที่ 13

```
history_dnn = model.fit(
  train_features,
  train_labels,
  validation_split=0.2,
  verbose=0, epochs=100)
hist = pd.DataFrame(history_dnn.history)
hist['epoch'] = history_dnn.epoch
hist.tail()
```

	loss	coeff_determination	val_loss	val_coeff_determination	epoch
95	257948256.0	0.994703	307902016.0	0.994743	95
96	264336480.0	0.994294	290354688.0	0.994519	96
97	272808800.0	0.994585	274657760.0	0.994982	97
98	253937984.0	0.994966	325987936.0	0.994581	98
99	257195744.0	0.994661	302604960.0	0.994242	99

ภาพที่ 13 ตัวอย่างผลลัพธ์การวนปรับค่าน้ำหนักของแบบจำลอง

4.10 ทดสอบแบบจำลองโดยใช้ model.evaludate

ผลลัพธ์คือค่า Loss และ R^2

```
model.evaluate(test_features, test_labels, verbose=0)
```

```
[217531040.0, 0.9945358633995056]
```

4.11 ใช้แบบจำลองเพื่อทำนายค่า

ใช้ model.predict

- พารามิเตอร์เป็น input แต่ละตัว ในกรณีตัวอย่างคือ q1, q2, q3, u, u1, u2, u3, ตามลำดับ
- ผลลัพธ์ที่ได้คือค่า output ในที่นี้คือปริมาณการใช้น้ำ q ที่ได้จากการพยากรณ์

```
model.predict(np.array([2260000,2170000,2080000,11150,10700,10300,9900]))
```

```
1/1 [==============================] - 0s 108ms/step
array([[2096332.4]], dtype=float32)
```

4.12 การบันทึกแบบจำลอง (Save and load models in Tensorflow, 2025)

1. การบันทึกแบบจำลองหมายถึงการบันทึกค่าน้ำหนักและไบแอสที่ผ่านการฝึกเสร็จแล้วเก็บไว้ในไฟล์ เมื่อโหลดไฟล์นี้เข้าสู่โครงสร้างแบบจำลอง จะเป็นการกำหนดค่าน้ำหนักและไบแอสตามที่บันทึกไว้ทันที ด้วยความสามารถนี้ทำให้ผู้ใช้ไม่ต้องทำการฝึกแบบจำลองทุกครั้งที่จะใช้งาน และการพยากรณ์จะให้ผลลัพธ์คงเดิมทุกครั้ง

2. การบันทึกแบบจำลองใช้รูปแบบคำสั่ง model.save_weight โดยระบุชื่อไฟล์ที่ต้องการบันทึก ไฟล์จะถูกบันทึกเข้า Google Drive มีนามสกุลเป็น .h5

3. กระบวนการโหลดแบบจำลองจะทำให้ไม่ต้องเสียเวลาฝึกแบบจำลองอีก จึงไม่ต้องใช้คำสั่ง model.fit แต่ใช้คำสั่ง model.load_weight แทนโดยระบุที่อยู่ของไฟล์ .h5 ที่ทำการบันทึกไว้

```
# บันทึกแบบจำลอง
# ต้องผ่านขั้นตอน 4.5-4.10 ก่อน จึงจะใช้คำสั่งได้
model.save_weights('ชื่อไฟล์.h5', overwrite=True)

# โหลดแบบจำลอง
# ต้องผ่านขั้นตอน 4.5-4.9 ก่อน และมีไฟล์ .h5 แล้ว จึงจะใช้คำสั่งได้
weight = '/content/drive/MyDrive/ชื่อโฟลเดอร์/ชื่อไฟล์.h5'
model.load_weights(weight, skip_mismatch=False)
```

5. สรุปผล

บทความนี้ได้นำเสนอแนวทางการสร้างแบบจำลองพยากรณ์โดยใช้โครงข่ายประสาทเทียม (ANN) ร่วมกับแพลตฟอร์ม TensorFlow ตั้งแต่หลักการทำงานเบื้องต้น การแปลงแนวคิดสู่สมการเชิงคณิตศาสตร์ ไปจนถึงขั้นตอนการใช้งาน TensorFlow อย่างละเอียด ความรู้เหล่านี้จะเป็นพื้นฐานสำคัญในการพัฒนาแบบจำลองพยากรณ์ของตนเองที่มีประสิทธิภาพสูงตรงตามความต้องการของผู้ใช้ และสามารถนำไปประยุกต์ใช้ได้จริงในหลากหลายสถานการณ์

ความสามารถของ TensorFlow ที่ไม่ได้กล่าวถึงยังมีอีกมาก เช่นการประมวลผลภาพ หรือเสียง การใช้รูปแบบโครงข่ายอื่น ๆ เช่น RNN ในการสร้างแบบจำลอง ผู้อ่านสามารถศึกษาข้อมูลเพิ่มเติมได้ที่ https://www.tensorflow.org/tutorials และ https://keras.io/examples/

บรรณานุกรม

Al Akashi, F. H. A. (2022). Stock market index prediction using artificial neural network. *Journal of Information Technology Research*, *15*(1), 1–16. https://doi.org/10.4018/JITR.299918

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences (2nd ed.)*. Lawrence Erlbaum Associates. from https://doi.org/10.4324/9780203771587

Colab.research. (2025). Retrieved 11 Jun 2025, from https://colab.research.google.com/

Install TensorFlow with pip (2025). Retrieved 11 Jun 2025, from https://www.tensorflow.org/install/pip

Kingma, Diederik P., and Jimmy Lei Ba. Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. from https://arxiv.org/pdf/1412.6980

Keras (2025). Retrieved 11 Jun 2025, from https://keras.io/examples/

Learnopencv (2017). Retrieved 11 Jun 2025, from https://learnopencv.com/wp-content/uploads/2017/10/neuron-diagram.jpg

Mao, W. B., Lyu, J. Y., Vaishnani, D. K., Lyu, Y. M., Gong, W., Xue, X. L., Shentu, Y. P., & Ma, J. (2020). Application of artificial neural networks in detection and diagnosis of gastrointestinal and liver tumors. *World Journal of Clinical Cases, 8*(18), 3971–3977. https://doi.org/10.12998/wjcc.v8.i18.3971

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (pp. 807-814). from https://www.cs.toronto.edu/~hinton/absps/reluICML.pdf

Noor, M. B. T., Zenia, N. Z., Kaiser, M. S., Mamun, S. A., & Mahmud, M. (2020). Application of deep learning in detecting neurological disorders from magnetic resonance images: a survey on the detection of Alzheimer's disease, Parkinson's disease and schizophrenia. *Brain informatics*, *7*, 1-21. https://doi.org/10.1186/s40708-020-00112-2

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533-536. from https://doi.org/10.1038/323533a0

System Hardware Requirements for TensorFlow in 2025. Retrieved 11 Jun 2025, from https://www.proxpc.com/blogs/system-hardware-requirements-for-tensorflow-in-2025

Save and load models in Tensorflow (2025). Retrieved 11 Jun 2025, from https://www.geeksforgeeks.org/save-and-load-models-in-tensorflow/

TensorFlow Core (2025). Retrieved 11 Jun 2025, from https://www.tensorflow.org/tutorials

tf.keras.Sequential (2025). Retrieved 11 Jun 2025, from https://www.tensorflow.org/api_docs/python/tf/keras/Sequential

tf.keras.layers.Dense. (2025). Retrieved 11 Jun 2025, from https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense

tf.keras.Model. (2025). Retrieved 11 Jun 2025, from https://www.tensorflow.org/api_docs/python/tf/keras/Model

tf.keras.callbacks.EarlyStopping (2025). Retrieved 11 Jun 2025, from https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping

Wikimedia commons (2025). Retrieved 11 Jun 2025, from https://commons.wikimedia.org/wiki/File:Neuron3.svg