ระบบแนะนำการเติมสินค้าอัจฉริยะสำหรับร้านค้า

Intelligent Stock Replenishment System for Retail Stores

นายนนทกร สิงห์กระโจม 6545000128

รายงานการค้นคว้าอิสระนี้ เป็นส่วนหนึ่งของการศึกษา

ตามหลักสูตรปริญญาวิศวกรรมศาสตร์

คณะวิศวกรรมศาสตร์และเทคโนโลยี

สาขาวิศวกรรมคอมพิวเตอร์และระบบปัญญาประดิษฐ์

สถาบันการจัดการปัญญาภิวัฒน์

ปีการศึกษา ๒๕๖๗

ระบบแนะนำการเติมสินค้าอัจฉริยะสำหรับร้านค้า

นายนนทกร สิงห์กระโจม 6545000128

รายงานการค้นคว้าอิสระนี้ เป็นส่วนหนึ่งของการศึกษา

ตามหลักสูตรปริญญาวิศวกรรมศาสตร์

คณะวิศวกรรมศาสตร์และเทคโนโลยี

สาขาวิศวกรรมคอมพิวเตอร์และระบบปัญญาประดิษฐ์

สถาบันการจัดการปัญญาภิวัฒน์

ปีการศึกษา ๒๕๖๗

Intelligent Stock Replenishment System for Retail Stores

Mr. Nontakorn Singkrajom 6545000128

A Senior Project Submitted in Partial Fulfillment of the Requirements

For the Degree of Bachelor of Computer Engineering Faculty of

Engineering and Technology

Academic Year 2024

เรื่อง ระบบแนะนำการเติมสินค้าอัจฉริยะสำหรับร้านค้า

โดย นายนนทกร สิงห์กระโจม

อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ รศ.ดร.ปริญญา สงวนสัตย์

คณะ วิศวกรรมศาสตร์และเทคโนโลยี

สาขาวิชา วิศวกรรมคอมพิวเตอร์และระบบปัญญาประดิษฐ์

รายการงานศึกษาอิสระเล่มนี้ได้รับความเห็นชอบให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมคอมพิวเตอร์และปัญญาประดิษฐ์ คณะวิศวกรรมศาสตร์และเทคโนโลยี สถาบันการจัดการปัญญาภิวัฒน์

..... คณบดีคณะวิศวกรรมศาสตร์และเทคโนโลยี

(ผศ.ดร.พรรณเชษฐ ณ ลำพูน)

..... ประธานกรรมการ

(รศ.ดร.ปริญญา สงวนสัตย์)

..... กรรมการ

(ผศ.ดร.อดิศร แขกซอง)

..... กรรมการ

(ดร.ติณณภพ ดินดำ)

..... หัวหน้าสาขาวิชาวิศวกรรมคอมพิวเตอร์

(รศ.ดร.ปริญญา สงวนสัตย์).

เรื่อง ระบบแนะนำการเติมสินค้าอัจฉริยะสำหรับร้านค้า

โดย นายนนทกร สิงห์กระโจม

อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ รศ.ดร.ปริญญา สงวนสัตย์

คณะ วิศวกรรมศาสตร์และเทคโนโลยี

สาขาวิชา วิศวกรรมคอมพิวเตอร์และระบบปัญญาประดิษฐ์

บทคัดย่อ

โครงการนี้มีวัตถุประสงค์เพื่อพัฒนาระบบแนะนำการเติมสินค้าอัจฉริยะสำหรับร้านค้าด้วยการใช้ปัญญาประดิษฐ์ (AI) และเทคนิคการเรียนรู้ของเครื่อง (Machine Learning) โดยระบบจะคำนวณและแนะนำปริมาณสินค้าที่ควรสั่งเติมเพื่อให้ตรงกับความต้องการของลูกค้าและป้องกันการขาดสต็อก

ระบบจะทำการรวบรวมข้อมูลการขายสินค้าจากร้านค้า เช่น สินค้าที่ขายออก จำนวนที่ขาย และวันที่ขาย จากนั้น AI จะนำข้อมูลเหล่านี้มาวิเคราะห์เพื่อพยากรณ์ปริมาณสินค้าที่ต้องการเติมในอนาคต โดยใช้โมเดลการเรียนรู้ เช่น Time Series Forecasting และ Recommender System

กระบวนการศึกษาเกี่ยวข้องกับการจัดการและเตรียมข้อมูล การเลือกใช้โมเดล Machine Learning ที่เหมาะสม รวมถึงการประเมินประสิทธิภาพของระบบในการคำนวณและแนะนำสินค้า

ผลลัพธ์ของโครงการนี้สามารถช่วยลดปัญหาการขาดสินค้าในร้านค้าและเพิ่มประสิทธิภาพในการสั่งซื้อสินค้าผ่านการคาดการณ์ที่แม่นยำ ทำให้ร้านค้าสามารถจัดการสต็อกได้อย่างมีประสิทธิภาพและตอบสนองความต้องการของลูกค้าได้ดียิ่งขึ้น

Title Intelligent Stock Replenishment System for Retail Stores

Author Mr. Nontakorn Singkrajom

Advisor Assoc. Prof. Dr. Parinya Sanguansat

Faculty Engineering and Technology

Program Computer Engineering and Artificial Intelligence

Abstract

This project aims to develop an intelligent product replenishment recommendation system for retail stores using Artificial Intelligence (AI) and Machine Learning techniques. The system calculates and recommends the optimal quantities of products that should be reordered to meet customer demand and prevent stockouts.

The system collects sales data from stores, including sold products, quantities sold, and sale dates. The AI model then analyzes this data to forecast the required replenishment quantities using approaches such as Time Series Forecasting and Recommender Systems.

The study involves data management and preprocessing, selecting appropriate Machine Learning models, and evaluating the system's performance in calculating and recommending replenishment quantities.

The results of this project help reduce stockout issues and improve the efficiency of product ordering through accurate demand forecasting. This enables stores to manage inventory more effectively and better satisfy customer needs.

กิตติกรรมประกาศ

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา 1321306 โครงงานวิศวกรรมคอมพิวเตอร์และปัญญาประดิษฐ์ 2 และสำเร็จลุล่วงไปได้ด้วยดีด้วยความอนุเคราะห์จากที่ปรึกษาและคณาจารย์ผู้มากความสามารถจากสถาบันการจัดการปัญญาภิวัฒน์ที่ได้มอบความรู้ให้ผู้จัดทำเป็นอย่างดี ที่ได้ให้การสนับสนุนและเป็นกำลังใจอย่างต่อเนื่อง อันเป็นแรงผลักดันสำคัญที่ทำให้การดำเนินงานวิจัยครั้งนี้สามารถสำเร็จลุล่วงได้อย่างเต็มประสิทธิภาพ

ขอขอบพระคุณรองศาสตราจารย์ ดร.ปริญญา สงวนสัตย์, ดร.ติณณภพ ดินดำ, ผศ.ดร.อดิศร แขกซอง และ ดร.ชนกานต์ กิ่งแก้ว ที่กรุณาให้คำแนะนำ สั่งสอน และชี้แนะแนวทางตลอดการศึกษา ตลอดจนการดำเนินงานวิจัย ซึ่งเป็นประโยชน์อย่างยิ่งต่อการจัดทำผลงานชิ้นนี้ สุดท้ายนี้ผู้จัดทำหวังเป็นอย่างยิ่งว่าการนำเสนอผลงานครั้งนี้จะเป็นประโยชน์และเป็นแรงบันดาลใจแก่ผู้ที่สนใจในการพัฒนาผลงานด้านเทคโนโลยีในอนาคตต่อไป ทั้งนี้ผู้จัดทำได้ใช้เครื่องมือ AI เพื่อช่วยในการร่างและเรียบเรียงเนื้อหาในบางส่วนของรายงาน เช่น บทคัดย่อและการอธิบายพื้นฐานทางเทคโนโลยี โดยเนื้อหาทั้งหมดได้ผ่านการตรวจสอบและปรับแก้ด้วยตนเองเพื่อความถูกต้องและความเหมาะสม

ด้วยความเคารพ

นนทกร สิงห์กระโจม

19 พฤศจิกายน 2568

สารบัญ

สารบัญตาราง

สารบัญรูปภาพ

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันการบริหารจัดการสินค้าคงคลัง (Inventory Management) เป็นหัวใจสำคัญของธุรกิจค้าปลีก ปัญหาที่พบบ่อยคือการขาดความสมดุลระหว่างอุปสงค์และอุปทาน การสั่งซื้อสินค้าที่น้อยเกินไปนำไปสู่ปัญหาสินค้าขาดสต็อก (Stockout) ทำให้เสียโอกาสในการขาย ในขณะที่การสั่งซื้อมากเกินไปก่อให้เกิดต้นทุนจมและปัญหาสินค้าล้นสต็อก (Overstock) การตัดสินใจสั่งซื้อโดยอาศัยเพียงประสบการณ์หรือการคำนวณพื้นฐานมักเกิดความผิดพลาดได้ง่าย โดยเฉพาะเมื่อต้องจัดการกับสินค้าจำนวนมากและมีความผันผวนของยอดขายสูง

เทคโนโลยีปัญญาประดิษฐ์ (AI) และการเรียนรู้ของเครื่อง (Machine Learning) เข้ามามีบทบาทสำคัญในการแก้ไขปัญหานี้ อย่างไรก็ตาม การใช้แบบจำลองเพียงรูปแบบเดียวมักมีข้อจำกัด เช่น แบบจำลองทางสถิติอาจปรับตัวได้ช้าต่อรูปแบบข้อมูลที่ซับซ้อน หรือแบบจำลอง Deep Learning อาจใช้ทรัพยากรสูงและเสี่ยงต่อการเกิด Overfitting งานวิจัยนี้จึงเล็งเห็นความสำคัญของการนำเทคนิคการเรียนรู้แบบผสมผสาน (Ensemble Learning) มาประยุกต์ใช้ โดยการรวมจุดแข็งของแบบจำลองที่หลากหลาย ได้แก่ Autoencoder, Exponential Smoothing และ Linear Regression เข้าด้วยกัน

โครงการนี้จึงมุ่งเน้นพัฒนาระบบพยากรณ์และแนะนำการเติมสินค้าอัจฉริยะ ที่ใช้เทคนิค Ensemble Learning ในการประมวลผลข้อมูลยอดขายย้อนหลัง เพื่อทำนายความต้องการสินค้าในอนาคตได้อย่างแม่นยำและเสถียรยิ่งขึ้น พร้อมทั้งพัฒนาในรูปแบบ Web Application ที่ใช้งานได้จริง เพื่อช่วยให้ร้านค้าสามารถบริหารจัดการสต็อกได้อย่างมีประสิทธิภาพ

1.2 วัตถุประสงค์ของการศึกษา

1. เพื่อพัฒนาระบบพยากรณ์ความต้องการสินค้า (Demand Forecasting) โดยใช้เทคนิคการเรียนรู้แบบผสมผสาน (Ensemble Learning)
2. เพื่อศึกษาและเปรียบเทียบประสิทธิภาพของแบบจำลอง Autoencoder, Exponential Smoothing และ Linear Regression ในการทำนายยอดขาย
3. เพื่อพัฒนาระบบ Web Application และ API สำหรับการแสดงผลการพยากรณ์และแนะนำรายการสินค้าที่ควรเติม (Top-N Products)
4. เพื่อประเมินประสิทธิภาพความแม่นยำของระบบผ่านตัวชี้วัดมาตรฐาน ได้แก่ MAE, RMSE และ MAPE

1.3 ขอบเขตของการศึกษา

1.3.1 ขอบเขตด้านข้อมูล

- ใช้ข้อมูลธุรกรรมการขายจริง (Transaction Data) ประกอบด้วย รหัสร้านค้า, รหัสสินค้า, วันที่ขาย และจำนวนที่ขาย
- ใช้ข้อมูลย้อนหลังเป็นระยะเวลาต่อเนื่อง (Window Size) เช่น 7 วัน เพื่อพยากรณ์ยอดขายในวันถัดไป

1.3.2 ขอบเขตด้านเทคนิคและแบบจำลอง

- พัฒนาระบบด้วยภาษา Python โดยใช้เทคนิค Ensemble Learning แบบ Weighted Averaging
- แบบจำลองที่ใช้ศึกษาประกอบด้วย:
  - Autoencoder (Deep Learning): สำหรับจับรูปแบบความสัมพันธ์ที่ซับซ้อนและไม่เป็นเชิงเส้น
  - Exponential Smoothing (Statistical Method): สำหรับจัดการข้อมูลที่มีแนวโน้มและฤดูกาล
  - Linear Regression (Machine Learning): สำหรับหาความสัมพันธ์เชิงเส้นพื้นฐาน

1.3.3 ขอบเขตด้านระบบงาน

- ส่วนประมวลผล (Backend): พัฒนาด้วย Flask Framework เชื่อมต่อกับฐานข้อมูล SQLite
- ส่วนแสดงผล (Frontend): พัฒนา Web Interface สำหรับแสดง Dashboard, ตารางเปรียบเทียบผลลัพธ์ และกราฟแสดงค่าความคลาดเคลื่อน

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. เพิ่มความแม่นยำและความเสถียรในการพยากรณ์: การประยุกต์ใช้เทคนิค Ensemble Learning ที่รวมจุดเด่นของแบบจำลองทั้ง 3 ชนิด (Deep Learning, Statistical, Machine Learning) ช่วยลดความคลาดเคลื่อนและให้ผลลัพธ์ที่น่าเชื่อถือมากกว่าการใช้โมเดลเดี่ยว
2. ช่วยลดปัญหาการขาดแคลนสินค้า (Stockout) และสินค้าล้นสต็อก (Overstock): ระบบสามารถพยากรณ์ปริมาณความต้องการสินค้าในวันถัดไปได้อย่างแม่นยำ ช่วยให้ผู้ประกอบการตัดสินใจเติมสินค้าได้ในปริมาณที่เหมาะสมและทันเวลา
3. เพิ่มประสิทธิภาพในการบริหารจัดการต้นทุน: การมีข้อมูลพยากรณ์ที่แม่นยำช่วยลดต้นทุนจม (Sunk Cost) จากการสต็อกสินค้าเกินความจำเป็น และลดความสูญเสียจากสินค้าหมดอายุ
4. อำนวยความสะดวกในการตัดสินใจด้วยระบบ Web Application: ผู้ใช้งานสามารถเข้าถึงข้อมูลการพยากรณ์ กราฟเปรียบเทียบ และรายการสินค้าแนะนำ (Top-N) ได้ง่ายผ่านหน้าเว็บไซต์ ช่วยลดภาระงานและลดการใช้ดุลยพินิจส่วนบุคคล (Human Error)
5. เป็นต้นแบบสำหรับการประยุกต์ใช้เทคโนโลยี AI ในธุรกิจค้าปลีก: โครงการนี้แสดงให้เห็นถึงความเป็นไปได้ในการนำเทคนิคขั้นสูงมาใช้งานจริงกับข้อมูลร้านค้าทั่วไป ซึ่งสามารถขยายผลไปยังร้านค้าขนาดใหญ่หรือเครือข่ายร้านค้าได้ในอนาคต

1.5 ระยะเวลาที่ใช้ในการวิจัย

โครงการนี้มีระยะเวลาการศึกษาเริ่มต้นตั้งแต่เดือนมิถุนายน 2568 จนถึงเดือนพฤศจิกายน 2568 รวมระยะเวลา 6 เดือน โดยมีแผนการดำเนินงานดังนี้

- เดือนมิถุนายน: ศึกษาทฤษฎี รวบรวมข้อมูลยอดขาย และวิเคราะห์ความต้องการระบบ
- เดือนกรกฎาคม: ออกแบบฐานข้อมูล เตรียมข้อมูล (Data Preprocessing) และสร้าง Data Pipeline
- เดือนสิงหาคม: พัฒนาและปรับจูนโมเดลพื้นฐาน (Autoencoder, Exp. Smoothing, Linear Regression)
- เดือนกันยายน: พัฒนาอัลกอริทึม Ensemble และสร้างระบบ Backend API เชื่อมต่อฐานข้อมูล
- เดือนตุลาคม: พัฒนาส่วนแสดงผล (Frontend) ทดสอบระบบ และประเมินผลความแม่นยำ
- เดือนพฤศจิกายน: สรุปผลการศึกษา จัดทำรายงานฉบับสมบูรณ์ และนำเสนอโครงงาน

การวิจัยจะมีการติดตามความก้าวหน้าและปรับปรุงกระบวนการอย่างต่อเนื่อง เพื่อให้บรรลุวัตถุประสงค์ที่กำหนด

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

การพัฒนาระบบแนะนำการเติมสินค้าอัจฉริยะด้วยเทคนิค Ensemble Learning ผู้จัดทำได้ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้อง เพื่อเป็นพื้นฐานในการออกแบบและพัฒนาระบบ โดยมีรายละเอียดดังนี้

2.1 ทฤษฎีการพยากรณ์ความต้องการ (Demand Forecasting Theory)

การพยากรณ์ความต้องการ (Demand Forecasting) เป็นกระบวนการคาดการณ์ปริมาณความต้องการสินค้าหรือบริการในอนาคต โดยอาศัยข้อมูลในอดีตและปัจจัยแวดล้อมต่างๆ การพยากรณ์ที่มีความแม่นยำมีความสำคัญอย่างยิ่งต่อการบริหารจัดการสินค้าคงคลัง ช่วยลดต้นทุนในการจัดเก็บสินค้า (Inventory Cost) และลดโอกาสการสูญเสียยอดขายจากการที่สินค้าขาดสต็อก (Stockout)

2.1.1 การวิเคราะห์อนุกรมเวลา (Time Series Analysis)

ข้อมูลยอดขายเป็นข้อมูลที่มีลักษณะเป็นอนุกรมเวลา (Time Series Data) คือชุดข้อมูลที่มีการเก็บรวบรวมตามลำดับเวลาที่ต่อเนื่องกัน การวิเคราะห์ข้อมูลประเภทนี้มุ่งเน้นไปที่การค้นหารูปแบบ (Pattern) ที่ซ่อนอยู่ภายในข้อมูล เช่น แนวโน้ม (Trend) และฤดูกาล (Seasonality) เพื่อนำมาสร้างแบบจำลองสำหรับการทำนายค่าในอนาคต

2.2 หลักการของ Ensemble Learning

Ensemble Learning เป็นเทคนิคในสาขา Machine Learning ที่ใช้หลักการรวมผลลัพธ์จากแบบจำลองการเรียนรู้ (Base Learners) หลายตัวเข้าด้วยกัน เพื่อให้ได้ผลลัพธ์ที่มีประสิทธิภาพสูงกว่าการใช้แบบจำลองเดี่ยวเพียงตัวเดียว

2.2.1 หลักการทำงานและประโยชน์

แนวคิดหลักของ Ensemble Learning คือการลดความคลาดเคลื่อนที่เกิดจากความแปรปรวน (Variance) หรือความลำเอียง (Bias) ของแบบจำลองเดี่ยว โดยการรวมพลังของแบบจำลองที่หลากหลายเข้าด้วยกัน (Wisdom of the Crowd) ช่วยให้ระบบมีความทนทาน (Robustness) ต่อสัญญาณรบกวนในข้อมูลและลดโอกาสในการเกิด Overfitting

2.2.2 การรวมผลด้วยวิธีค่าเฉลี่ยถ่วงน้ำหนัก (Weighted Averaging)

ในโครงการนี้ใช้วิธีการรวมผลแบบ Weighted Averaging ซึ่งเป็นการนำผลพยากรณ์จากแต่ละแบบจำลองมาคูณด้วยค่าน้ำหนัก (Weight) ที่กำหนดตามประสิทธิภาพของแบบจำลองนั้นๆ แล้วนำมารวมกัน ค่าพยากรณ์สุดท้าย $(y_{ensemble})$ คำนวณได้จากสมการ:

$$y_{ensemble} = \sum_{i=1}^{n} w_i \times y_i$$

2.3 ทฤษฎีของแบบจำลองที่เลือกใช้ (Base Models)

2.3.1 Exponential Smoothing

Exponential Smoothing เป็นเทคนิคทางสถิติสำหรับพยากรณ์ข้อมูลอนุกรมเวลา โดยให้ความสำคัญกับข้อมูลล่าสุดมากกว่าข้อมูลในอดีต ด้วยการกำหนดค่าถ่วงน้ำหนักที่ลดลงแบบเอ็กซ์โพเนนเชียล (Exponentially decreasing weights) เหมาะสำหรับข้อมูลที่มีรูปแบบไม่ซับซ้อนมากนักและต้องการการประมวลผลที่รวดเร็ว

2.3.2 Linear Regression

Linear Regression หรือการถดถอยเชิงเส้น เป็นเทคนิคพื้นฐานทางสถิติและ Machine Learning ที่ใช้หาความสัมพันธ์เชิงเส้นระหว่างตัวแปรต้น (Features) และตัวแปรตาม (Target) ในบริบทของการพยากรณ์ยอดขาย Linear Regression จะพยายามสร้างสมการเส้นตรงที่สามารถอธิบายแนวโน้มของยอดขายได้ดีที่สุด เหมาะสำหรับข้อมูลที่มีแนวโน้มการเติบโตหรือลดลงอย่างชัดเจน

2.3.3 Autoencoder สำหรับอนุกรมเวลา

Autoencoder เป็นโครงข่ายประสาทเทียม (Artificial Neural Network) ประเภทหนึ่งที่มีโครงสร้างแบบ Encoder-Decoder โดยเรียนรู้ที่จะบีบอัดข้อมูลนำเข้า (Input) ให้เป็นรหัส (Latent Space) และสร้างข้อมูลใหม่ (Reconstruction) ให้เหมือนกับข้อมูลเดิมมากที่สุด ในการพยากรณ์อนุกรมเวลา Autoencoder สามารถเรียนรู้คุณลักษณะที่ซับซ้อนและไม่เป็นเชิงเส้น (Non-linear features) ของข้อมูลยอดขายได้ ซึ่งเป็นจุดที่แบบจำลองทางสถิติทั่วไปอาจทำได้ไม่ดีเท่า

2.4 การวัดประสิทธิภาพการพยากรณ์ (Evaluation Metrics)

เนื่องจากระบบนี้เป็นการพยากรณ์ค่าเชิงปริมาณ (Regression Problem) การวัดประสิทธิภาพจึงใช้ตัวชี้วัดความคลาดเคลื่อน (Error Metrics) ดังนี้

2.4.1 Mean Absolute Error (MAE)

ค่าเฉลี่ยความคลาดเคลื่อนสัมบูรณ์ วัดค่าเฉลี่ยของผลต่างระหว่างค่าจริงและค่าพยากรณ์ เป็นตัวชี้วัดที่เข้าใจง่ายและมีหน่วยเดียวกับข้อมูล

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_{true} - y_{pred}|$$

2.4.2 Root Mean Squared Error (RMSE)

รากที่สองของค่าเฉลี่ยความคลาดเคลื่อนยกกำลังสอง เป็นตัวชี้วัดที่ให้ความสำคัญกับความคลาดเคลื่อนที่มีค่าสูง (Large Errors) มากกว่า MAE เหมาะสำหรับการตรวจสอบว่าโมเดลมีความผิดพลาดรุนแรงหรือไม่

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2}$$

2.4.3 Mean Absolute Percentage Error (MAPE)

ค่าเฉลี่ยร้อยละของความคลาดเคลื่อนสัมบูรณ์ วัดความคลาดเคลื่อนเป็นเปอร์เซ็นต์ ทำให้สามารถเปรียบเทียบประสิทธิภาพระหว่างชุดข้อมูลที่มีสเกลต่างกันได้

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\frac{y_{true} - y_{pred}}{y_{true}}$$

2.5 งานวิจัยที่เกี่ยวข้อง

เพื่อสร้างความเข้าใจและบริบทสำหรับการพัฒนาระบบพยากรณ์และเติมสินค้าอัจฉริยะ งานวิจัยที่เกี่ยวข้องจากหลายด้านถูกนำมาศึกษา ซึ่งประกอบด้วยงานวิจัยด้านการพยากรณ์ความต้องการสินค้าด้วยเทคนิคผสมผสาน (Ensemble Forecasting) การประยุกต์ใช้ Autoencoder ในการเรียนรู้รูปแบบข้อมูลอนุกรมเวลา การจัดเก็บและจัดการข้อมูลด้วย SQLite Database รวมถึงการประเมินผลความแม่นยำของการพยากรณ์ การศึกษางานวิจัยเหล่านี้ช่วยให้สามารถออกแบบและพัฒนาระบบที่ตอบสนองความต้องการด้านการบริหารจัดการสินค้าคงคลังได้อย่างมีประสิทธิภาพ

2.5.1 ด้านการพยากรณ์และ Ensemble Learning

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018), “The M4 Competition: Results, findings, conclusions and way forward”: งานวิจัยนี้สรุปผลการแข่งขันการพยากรณ์ระดับโลก (M4 Competition) ซึ่งค้นพบข้อสรุปสำคัญว่า วิธีการแบบผสมผสาน (Hybrid/Ensemble methods) ที่รวมเอาเทคนิคทางสถิติและ Machine Learning เข้าด้วยกัน มักให้ผลลัพธ์ที่มีความแม่นยำสูงกว่าการใช้วิธีใดวิธีหนึ่งเพียงอย่างเดียว ซึ่งสนับสนุนแนวคิดของโครงงานนี้ที่นำ Autoencoder มาทำงานร่วมกับ Exponential Smoothing และ Linear Regression

2.5.2 ด้าน Autoencoder

Vincent et al. (2010), “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion”: งานวิจัยนี้เสนอวิธีการใช้ Autoencoder ในการเรียนรู้ตัวแทนเชิงลึก (Deep Representations) ของข้อมูล โดยสามารถลดมิติและสกัดคุณลักษณะที่สำคัญ (Feature Extraction) ออกมาจากข้อมูลที่มีความซับซ้อนได้ ซึ่งสามารถนำมา

ประยุกต์ใช้ในการจับรูปแบบความสัมพันธ์ที่ไม่เป็นเชิงเส้น (Non-linear patterns) ในข้อมูลยอดขายย้อนหลังได้อย่างมีประสิทธิภาพ

2.5.3 ด้าน SQLite Database

Owens (2018), “Lightweight Databases for Embedded and Mobile Applications”: งานวิจัยนี้แสดงให้เห็นว่า SQLite เป็นฐานข้อมูลขนาดเล็กที่เหมาะสำหรับจัดการข้อมูลบนอุปกรณ์ที่มีทรัพยากรจำกัด สามารถจัดเก็บและเรียกใช้ข้อมูลได้รวดเร็วโดยไม่ต้องมีการตั้งค่า Server ที่ซับซ้อน ซึ่งเหมาะสมอย่างยิ่งสำหรับการพัฒนาระบบพยากรณ์แบบ Standalone หรือระบบที่ต้องการความคล่องตัวในการใช้งานระดับร้านค้า

2.5.4 ด้านการประเมินผลการพยากรณ์

Chai, T., & Draxler, R. R. (2014), “Root mean square error (RMSE) or mean absolute error (MAE)? Arguments against avoiding RMSE in the literature”: งานวิจัยนี้เปรียบเทียบและวิเคราะห์การใช้ตัวชี้วัดความคลาดเคลื่อนทางสถิติ โดยอธิบายถึงความเหมาะสมของการใช้ RMSE ในการวัดความคลาดเคลื่อนที่มีขนาดใหญ่ และการใช้ MAE เพื่อดูภาพรวมความผิดพลาดเฉลี่ย ซึ่งเป็นพื้นฐานสำคัญในการเลือกใช้ Metrics ทั้ง MAE, RMSE และ MAPE เพื่อประเมินประสิทธิภาพของโมเดลพยากรณ์ในโครงงานนี้

2.6 สรุปการเลือกใช้เทคโนโลยี

จากการศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้อง โครงการนี้จึงเลือกใช้เทคนิค Ensemble Learning โดยผสมผสานจุดเด่นของ Autoencoder (จับ Pattern ซับซ้อน), Exponential Smoothing (จับ Trend ระยะสั้นได้เร็ว), และ Linear Regression (หาความสัมพันธ์พื้นฐาน) เข้าด้วยกัน เพื่อให้ระบบแนะนำการเติมสินค้ามีความแม่นยำสูงสุดและครอบคลุมลักษณะข้อมูลที่หลากหลาย

บทที่ 3
วิธีดำเนินการวิจัย

ในบทนี้จะอธิบายขั้นตอนและวิธีการดำเนินการวิจัยที่ใช้ในการพัฒนาระบบแนะนำสินค้า โดยจะเริ่มตั้งแต่การเตรียมข้อมูล การเลือกโมเดลในการฝึกอบรม ไปจนถึงการประเมินผลการทำงานของระบบแนะนำสินค้า โดยจะเน้นการใช้เทคนิค Autoencoder ในการสร้างโมเดลและการประเมินผลการแนะนำสินค้าด้วยตัวชี้วัดที่เหมาะสม

3.1 การเตรียมข้อมูล (Data Preparation)

3.1.1 ภาพรวมการเตรียมข้อมูล

ระบบใช้ข้อมูลยอดขายรายวัน (Daily Sales Transaction) ที่เก็บในฐานข้อมูล SQLite ขนาด 36 MB โดยข้อมูลดิบประกอบด้วยฟิลด์หลัก 4 ฟิลด์ ได้แก่ STORE_ID (รหัสร้านค้า), PROD_CD (รหัสสินค้า), BSNS_DT (วันที่ทำการขาย) และ PROD_QTY (จำนวนสินค้าที่ขาย) โดยข้อมูลมีลักษณะเป็น Time Series Data ที่มีการบันทึกการขายต่อเนื่องในแต่ละวัน

ขั้นตอนการเตรียมข้อมูลถูกออกแบบให้ทำงานผ่าน Module data_preparation.py ซึ่งประกอบด้วยฟังก์ชันหลัก 3 ฟังก์ชัน คือ load_data(), prepare_transactions() และ prepare_time_series_data() โดยแต่ละฟังก์ชันมีหน้าที่เฉพาะในการจัดการข้อมูลให้พร้อมสำหรับการฝึกโมเดล

3.1.2 กระบวนการโหลดและทำความสะอาดข้อมูล

ฟังก์ชัน load_data() ทำหน้าที่โหลดข้อมูลจากไฟล์ CSV หรือฐานข้อมูล พร้อมทั้งดำเนินการทำความสะอาดข้อมูลเบื้องต้นเพื่อเตรียมสำหรับขั้นตอนวิเคราะห์ต่อไป

Input Parameters

- filepath: เส้นทางไฟล์ CSV หรือ connection string ของฐานข้อมูล
- columns_to_drop (optional): รายชื่อคอลัมน์ที่ต้องการลบออก

Output

- DataFrame ที่ผ่านการตรวจสอบและทำความสะอาดข้อมูลเรียบร้อยแล้ว

กระบวนการทำงานของฟังก์ชัน

1. การโหลดข้อมูล ใช้คำสั่ง pandas.read_csv() เพื่ออ่านข้อมูล พร้อมตรวจสอบว่าไฟล์มีอยู่จริงและไม่ใช่ไฟล์ว่าง หากผิดเงื่อนไขจะมีการแจ้งข้อผิดพลาดที่เหมาะสม
2. การลบคอลัมน์ที่ไม่จำเป็นคอลัมน์ที่ถูกระบุไว้ใน config.data.columns_to_drop จะถูกลบออกเพื่อลดขนาดข้อมูลและเพิ่มความคล่องตัวในการประมวลผล

3. การแปลงประเภทข้อมูลให้ถูกต้อง
    - BSNS_DT: แปลงเป็นชนิดข้อมูลวันที่ (datetime) พร้อมจัดการข้อผิดพลาด
    - STORE_ID, PROD_CD: แปลงเป็น string เพื่อความสม่ำเสมอ
    - PROD_QTY: ตรวจสอบและแปลงเป็นชนิดตัวเลข (numeric)
4. การจัดการค่า Missing และค่าที่ไม่สมเหตุสมผล
    - ค่า Missing ใน PROD_QTY จะถูกแทนที่ด้วยค่า 0
    - ค่าติดลบใน PROD_QTY จะถูกแก้ไขเป็น 0 เนื่องจากจำนวนสินค้าติดลบไม่สามารถเกิดขึ้นได้จริง
5. Data Quality Logging ฟังก์ชันจะบันทึกข้อมูลคุณภาพ เช่น จำนวนแถว จำนวนคอลัมน์ สัดส่วน missing values และประเภทข้อมูล เพื่อใช้ประกอบการวิเคราะห์คุณภาพข้อมูล

Conditions และ Error Handling

- หากไม่พบไฟล์ตามเส้นทางที่กำหนด จะเกิด FileNotFoundError
- หากไฟล์ไม่มีข้อมูล จะเกิด EmptyDataError
- ต้องมีคอลัมน์สำคัญที่จำเป็นสำหรับการประมวลผลครบถ้วน มิฉะนั้นจะหยุดการทำงานและแจ้งเตือนข้อผิดพลาด

3.1.3 การจัดกลุ่มและสร้างธุรกรรม

ฟังก์ชัน prepare_transactions() ทำหน้าที่รวมยอดการทำธุรกรรมที่เกิดขึ้นหลายครั้งในวันเดียวกัน โดยสรุปข้อมูลให้อยู่ในระดับรายวันสำหรับแต่ละร้านค้าและสินค้า

Input

- df: DataFrame ที่ผ่านการทำความสะอาดแล้ว (cleaned DataFrame)

Output

- DataFrame ที่ผ่านการ Group และสรุปยอดขายรายวันเรียบร้อยแล้ว

กระบวนการทำงานของฟังก์ชัน

1. ตรวจสอบความครบถ้วนของคอลัมน์ที่จำเป็น ฟังก์ชันจะตรวจสอบว่าข้อมูลมีคอลัมน์ที่ต้องใช้ ได้แก่ STORE_ID, BSNS_DT, PROD_CD, และ PROD_QTY หากขาดคอลัมน์ใดจะหยุดการทำงานและแจ้งข้อผิดพลาด
2. การจัดกลุ่มข้อมูล (Grouping) ใช้คำสั่ง groupby เพื่อจัดกลุ่มตามร้านค้า วันที่ และรหัสสินค้า
3. สรุปยอดขายรายวัน (Aggregation) ภายในแต่ละกลุ่ม จะทำการรวมยอด (sum) ของตัวแปร PROD_QTY เพื่อให้ได้ยอดขายรวมต่อวันต่อสินค้า
4. บันทึกตัวชี้วัดข้อมูล (Statistics Logging) ฟังก์ชันจะบันทึกค่าทางสถิติ เช่น
    - จำนวนร้านค้าที่ไม่ซ้ำ (unique stores)
    - จำนวนสินค้าที่ไม่ซ้ำ (unique products)

ตัวอย่าง Input และ Output

Input (หลายธุรกรรมในวันเดียวกัน)

STORE_ID	BSNS_DT	PROD_CD	PROD_QTY
11001	2025-01-01	P001	5
11001	2025-01-01	P001	3
11001	2025-01-01	P002	2

ตารางที่ 1 Input หลายธุรกรรม

Output (หลังการรวม)

STORE_ID	BSNS_DT	PROD_CD	PROD_QTY
11001	2025-01-01	P001	8
11001	2025-01-01	P002	2

ตารางที่ 2 Output รวมธุรกรรม

3.1.4 การสร้าง Time Series Sequences

ฟังก์ชัน prepare_time_series_data() มีหน้าที่แปลงข้อมูลยอดขายของสินค้าในแต่ละร้านให้อยู่ในรูปของลำดับเวลา (Time Series Sequences) เพื่อใช้สำหรับการฝึกโมเดลพยากรณ์ โดยอาศัยการสร้างข้อมูลย้อนหลังตามช่วงเวลา (window) ที่กำหนด

Input

- df: DataFrame ที่เก็บข้อมูลธุรกรรมรายวัน
- window_size: ขนาดหน้าต่างเวลา (เช่น 7 วัน) โดยมีค่าเริ่มต้นเป็น 7

Output

- Dictionary ที่จัดเก็บลำดับเวลาแยกตามร้านค้าและรหัสสินค้า พร้อมข้อมูลประกอบสำหรับการ denormalize และวิเคราะห์เพิ่มเติม

กระบวนการทำงานของฟังก์ชัน

1. การเรียงลำดับข้อมูลตามเวลา ข้อมูลธุรกรรมทั้งหมดจะถูกจัดเรียง (sort) ตามคอลัมน์ BSNS_DT เพื่อให้ลำดับเวลาถูกต้องก่อนนำไปสร้าง sequence
2. การวนลูปตามร้านค้าและสินค้า สำหรับแต่ละร้าน (STORE_ID) และแต่ละสินค้า (PROD_CD) ฟังก์ชันจะ:
   - คัดกรองข้อมูลเฉพาะของร้านและสินค้านั้น
   - สร้างข้อมูลยอดขายรายวัน
   - ตรวจสอบว่ามีจำนวนข้อมูลเพียงพออย่างน้อยเท่ากับ window_size

3. การสร้างลำดับเวลา (Sequence Generation) หากข้อมูลเพียงพอ:
    - เลือกข้อมูลย้อนหลังตามจำนวนวันในหน้าต่าง (เช่น 7 วันล่าสุด)
    - คำนวณค่าสูงสุดของช่วงดังกล่าว (max_val) เพื่อใช้ทำ Normalization
    - ทำ Normalization โดยหารค่าทั้งหมดด้วย max_val
    - หาก max_val = 0 จะกำหนดให้เป็น 1 เพื่อหลีกเลี่ยงการหารด้วยศูนย์
    - เก็บทั้งค่า normalized และค่าจริง (raw)
4. การจัดเก็บข้อมูล Metadata สำหรับแต่ละร้านและสินค้า จะจัดเก็บข้อมูลดังนี้:
    - sequence: ลำดับยอดขายแบบ normalized ช่วง 0–1
    - raw_sequence: ค่ายอดขายจริงในช่วงเดียวกัน
    - max_val: ค่ายอดขายสูงสุดในหน้าต่างเวลา
    - last_qty: ยอดขายล่าสุด
    - mean_qty: ค่าเฉลี่ยยอดขาย

3.1.5 สรุปผลการเตรียมข้อมูล

จากกระบวนการเตรียมข้อมูลทั้งหมด ข้อมูลที่ได้สามารถนำไปใช้สำหรับการฝึกโมเดลพยากรณ์ได้อย่างมีประสิทธิภาพ โดยมีลักษณะสำคัญดังนี้

- จำนวน sequences: หลายพันถึงหลายหมื่น sequences (ขึ้นกับจำนวนร้านและสินค้า)
- แต่ละ sequence มีความยาว 7 วัน
- ข้อมูลถูก normalize อยู่ในช่วง 0-1
- มีการเก็บค่า metadata สำหรับ denormalize ภายหลัง

3.2 การสร้างโมเดลแนะนำสินค้า (Model Building)

ระบบใช้วิธีการ Ensemble Learning โดยรวมโมเดลพยากรณ์ 3 แบบเข้าด้วยกัน เพื่อเพิ่มความแม่นยำและความเสถียรของการพยากรณ์ โมเดลทั้ง 3 แบบมีหลักการทำงานที่แตกต่างกัน ทำให้สามารถจับรูปแบบข้อมูลได้หลากหลาย

3.2.1 Autoencoder Model (Neural Network-based)

โมเดล Autoencoder ที่ใช้ในระบบนี้เป็นโครงข่ายประสาทเทียมแบบ Unsupervised Learning ซึ่งมีหลักการสำคัญคือการเรียนรู้เพื่อบีบอัดข้อมูลให้อยู่ในรูปแบบที่กระชับที่สุด และสามารถกู้คืนข้อมูลกลับมาได้อย่างแม่นยำผ่านกระบวนการ Encoding และ Decoding ภายในงานวิจัยนี้ Autoencoder ถูกประยุกต์ใช้เพื่อเรียนรู้รูปแบบพฤติกรรมยอดขายจากข้อมูลลำดับเวลา 7 วันที่เตรียมไว้ โดยหวังให้โมเดลสามารถจับโครงร่างหรือแนวโน้มของยอดขายในช่วงเวลาดังกล่าว และใช้ความรู้ที่ได้ในการพยากรณ์ยอดขายของวันถัดไป

สถาปัตยกรรมของโมเดลประกอบด้วยสองส่วนหลัก ได้แก่ Encoder และ Decoder โดยในส่วนของ Encoder จะรับข้อมูลที่มีความยาว 7 ค่าเข้าสู่เครือข่ายแบบ Dense พร้อมฟังก์ชันกระตุ้นแบบ ReLU และมีการใช้ Batch Normalization และ Dropout เพื่อช่วยเพิ่มความเสถียรและลดความเสี่ยงของ Overfitting ชั้นซ่อนเหล่านี้จะค่อย ๆ ลดจำนวนหน่วยประมวลผลลงจนเกิดเป็น Latent Space ที่มีมิติ 32 หน่วย ซึ่งทำหน้าที่เป็นตัวแทนเชิงนามธรรมของรูปแบบยอดขายในช่วง 7 วัน

ในส่วนของ Decoder โมเดลจะเริ่มต้นจาก Latent Space และทำการขยายข้อมูลกลับผ่านชั้น Dense หลายชั้นเช่นเดียวกัน โดยใช้ ReLU และ Batch Normalization เพื่อค่อย ๆ คืนโครงสร้างข้อมูลจนกระทั่งได้ผลลัพธ์สุดท้ายเป็นค่าเดียว ซึ่งแสดงถึงยอดขายที่คาดการณ์สำหรับวันถัดไป ผ่านชั้นเอาต์พุตแบบ Linear ที่เหมาะสมกับงานพยากรณ์เชิงตัวเลข (regression)

ด้วยโครงสร้างดังกล่าว Autoencoder สามารถเรียนรู้รูปแบบเชิงลึกของข้อมูลยอดขายรายวันได้อย่างมีประสิทธิภาพ และเหมาะสมกับงานที่ต้องการจับความสัมพันธ์ต่อเนื่องของข้อมูลตามลำดับเวลา (time-series pattern learning) เพื่อใช้ทำนายแนวโน้มยอดขายในอนาคต

Parameters และ Configuration:

- Input dimension: 7 (window_size)
- Encoding dimension: 32
- Optimizer: Adam (learning_rate=0.001)
- Loss function: Mean Squared Error (MSE)
- Metrics: Mean Absolute Error (MAE)
- Epochs: 50 (ปรับได้ผ่าน config)
- Batch size: 32
- Validation split: 0.2 (20% สำหรับ validation)

Input และ Output:

Input:

- X_normalized: numpy array รูป (n_samples, 7)
- ข้อมูล normalized 7 วันล่าสุด

Output:

- Predicted quantity (normalized): รูป (n_samples, 1)
- ต้อง denormalize ด้วย max_val เพื่อได้ค่าจริง

กระบวนการทำงาน:

1. Encoding: Encoder บีบอัดข้อมูล 7 วันเป็น latent representation 32 มิติ
2. Decoding: Decoder แปลง latent กลับเป็นการพยากรณ์ 1 ค่า
3. Training: ฝึกโมเดลด้วย MSE loss โดย minimize ความแตกต่างระหว่างค่าจริงและค่าพยากรณ์
4. Prediction: ใช้ Encoder สร้าง latent ใช้ Decoder สร้างค่าพยากรณ์

Conditions:

- ข้อมูล Input ต้องถูก normalize อยู่ในช่วง 0-1
- จำนวน samples ต้องมากพอสำหรับการฝึก (แนะนำ > 100)
- ต้องมี GPU หรือ CPU ที่เพียงพอ (model ใช้ TensorFlow/Keras)

ข้อดีและข้อจำกัด:

ข้อดี:

- เรียนรู้รูปแบบที่ซับซ้อน (non-linear patterns)
- จับ temporal dependencies ได้ดี
- สามารถ generalize กับข้อมูลใหม่

ข้อจำกัด:

- ต้องใช้เวลาในการฝึก
- อาจ overfit ถ้าข้อมูลน้อย
- ต้องการ computational resources มากกว่าโมเดลอื่น

3.2.2 Exponential Smoothing Model

Exponential Smoothing เป็นวิธีการทางสถิติที่ให้น้ำหนักกับข้อมูลล่าสุดมากกว่าข้อมูลเก่า โดยใช้ smoothing parameter (alpha) ในการปรับน้ำหนัก ระบบนี้รวม Exponential Smoothing เข้ากับ Moving Average เพื่อเพิ่มความเสถียร

สูตรคณิตศาสตร์:

Exponential Smoothing:

$$S_t = \alpha \times X_t + (1 - \alpha) \times S_{t-1}$$

- S_t: ค่า smoothed ณ เวลา t
- X_t: ค่าจริง ณ เวลา t
- α: smoothing parameter ($0 < \alpha < 1$)

Moving Average:

$$MA = \frac{1}{n} \times \Sigma(X_t - i) \ \ for \ \ i = 0 \ \ to \ \ n - 1$$

Ensemble Prediction:

Prediction = 0.6 × Exponential_Smoothing + 0.4 × Moving_Average

Parameters:

- alpha: 0.3 (ให้น้ำหนักกับค่าใหม่ 30%, ค่าเก่า 70%)
- window: 3 (ใช้ 3 วันล่าสุดสำหรับ Moving Average)

Input และ Output:

Input:

- sequences: numpy array รูป (n_samples, 7)
- ข้อมูลยอดขายจริง (ไม่ใช่ normalized)

Output:

- predictions: numpy array รูป (n_samples, 1)
- ค่าพยากรณ์ในหน่วยเดียวกับ input

กระบวนการทำงาน:

สำหรับแต่ละ sequence ของยอดขาย:

1. เริ่มต้นด้วยค่า s = sequence[0]
2. ทำการวนลูปตามลำดับวันแต่ละวันใน sequence โดยปรับค่า s ด้วยสูตร
$$s = \alpha \times \text{value} + (1 - \alpha) \times s$$
3. กำหนดผลลัพธ์ของ Exponential Smoothing เป็น exp_pred = s
4. คำนวณ Moving Average ของ 3 วันล่าสุดใน sequence
5. รวมผลทั้งสองวิธีเป็น final_prediction โดยถ่วงน้ำหนัก 60% สำหรับ exp_pred และ 40% สำหรับ ma_pred
$$\text{Prediction} = 0.6 \times \text{Exponential\_Smoothing} + 0.4 \times \text{Moving\_Average}$$

เงื่อนไขการใช้งาน

- ไม่ต้องฝึกโมเดล สามารถใช้งานได้ทันที
- เหมาะกับข้อมูลที่มีแนวโน้มเรียบ (smooth trend)

ข้อดี

- รวดเร็ว ไม่ต้องฝึกโมเดล
- เข้าใจง่าย สามารถอธิบายผลลัพธ์ได้ (interpretable)
- ใช้หน่วยความจำน้อย
- มีความเสถียร ไม่ไวต่อค่าผิดปกติ (outliers)

ข้อจำกัด

- ไม่สามารถจับรูปแบบซับซ้อน (complex patterns) ได้
- ไม่เหมาะกับข้อมูลที่มีความผันผวนสูง (high volatility)
- ต้องปรับค่า α ด้วยตนเองเพื่อให้ได้ผลลัพธ์เหมาะสม

3.2.3 Linear Regression Model

Linear Regression ใช้ความสัมพันธ์เชิงเส้นระหว่าง features และ target เพื่อพยากรณ์ โดยในระบบนี้ใช้ Ridge Regression (L2 regularization) เพื่อลด overfitting และสร้าง features จาก time series

สูตรคณิตศาสตร์:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \varepsilon \text{Loss} = \text{MSE} + \alpha \sum_{i=1}^{n} \beta_i^2$$

Feature Engineering

ในการสร้างโมเดล Ridge Regression จำเป็นต้องแปลงข้อมูลยอดขายย้อนหลัง 7 วันให้กลายเป็นชุดคุณลักษณะ (features) ที่สรุปพฤติกรรมและรูปแบบของข้อมูลให้ชัดเจนขึ้น โดยสร้างทั้งหมด 7 features ดังนี้:

1. Last value — ค่ายอดขายในวันล่าสุดของ sequence (X[-1])
2. Mean of last 3 days — ค่าเฉลี่ยของยอดขาย 3 วันสุดท้าย ช่วยสะท้อนพฤติกรรมระยะสั้น
3. Overall mean — ค่าเฉลี่ยของยอดขายทั้ง 7 วัน
4. Standard deviation — ค่าความผันผวนของข้อมูลในช่วง 7 วัน
5. Trend — ความเปลี่ยนแปลงระหว่างวันแรกและวันสุดท้าย (X[-1] – X[0])
6. Maximum value — ค่าสูงสุดภายใน 7 วัน
7. Minimum value — ค่าต่ำสุดภายใน 7 วัน

Parameters

- alpha = 1.0 — กำลังของ regularization เพื่อป้องกันการเกิด overfitting
- Scaler = StandardScaler — ทำให้แต่ละ feature มีค่าเฉลี่ย 0 และส่วนเบี่ยงเบนมาตรฐาน 1 เพื่อให้ Ridge Regression ทำงานได้เสถียรมากขึ้น

Input และ Output

Training Input

- X_train : ข้อมูล sequences รูปแบบ (n_samples, 7)
- y_train : ค่ายอดขายจริงรูปแบบ (n_samples, 1)

Prediction Output

- predictions : ค่าที่โมเดลทำนายออกมา รูปแบบ (n_samples, 1)

กระบวนการทำงาน

1. Feature Creation (_create_features)

โมเดลจะนำ sequence แต่ละรายการมาคำนวณเป็น features ทั้ง 7 รายการ และรวมเป็น feature matrix ขนาด (n_samples,7)

2. Training

- ทำการ scale features ด้วย StandardScaler
- ฝึกโมเดล Ridge Regression ด้วยค่าความแข็งแรงของ regularization (alpha = 1.0)
- บันทึก scaler และน้ำหนักของโมเดลสำหรับใช้ในอนาคต

3. Prediction

- แปลง sequence ใหม่ให้เป็น feature ทั้ง 7 เช่นเดียวกับช่วง train
- ใช้ scaler เดิมเพื่อ normalize ข้อมูล
- ทำนายค่าด้วย Ridge Regression
- ส่งค่า prediction กลับเป็นผลลัพธ์

Conditions

- ต้องทำการ train ก่อนใช้งาน (is_fitted = True)
- Features ต้องผ่าน StandardScaler เดียวกับตอนฝึก
- เหมาะกับข้อมูลที่มีความสัมพันธ์เชิงเส้นในบางส่วน

ข้อดีและข้อจำกัด

ข้อดี

- ทำงานรวดเร็วทั้งตอน train และ predict
- อธิบายผลลัพธ์ได้ง่าย เพราะดูความสำคัญของ features ได้
- Ridge ทำให้ robust ต่อ outliers
- ใช้ทรัพยากรน้อยมาก

ข้อจำกัด

- จับความสัมพันธ์ที่เป็นเชิงเส้นเท่านั้น
- อาจไม่เหมาะกับข้อมูลที่มีความซับซ้อนสูงหรือ non-linear มาก

3.2.4 สรุปเปรียบเทียบโมเดลทั้ง 3

ลักษณะ	Autoencoder	Exp Smoothing	Linear Regression
ประเภท	Deep Learning	Statistical	Machine Learning
Training	ต้องฝึก	ไม่ต้องฝึก	ต้องฝึก
ความซับซ้อน	สูง	ต่ำ	ปานกลาง
เวลาที่ใช้	มาก	น้อยที่สุด	น้อย
Non-linearity	ได้ดีมาก	ไม่ได้	จำกัด
Interpretability	ต่ำ	สูงมาก	สูง
Memory	มาก	น้อย	น้อย

ตารางที่ 3 เปรียบเทียบโมเดล

3.3 การรวมโมเดล (Ensemble Method)

3.3.1 หลักการ Ensemble Learning

Ensemble Learning เป็นเทคนิคที่รวมการพยากรณ์จากหลายโมเดลเข้าด้วยกัน เพื่อให้ได้ผลลัพธ์ที่ดีกว่าการใช้โมเดลเดียว โดยอาศัยหลักการที่ว่าแต่ละโมเดลมีจุดแข็งและจุดอ่อนที่แตกต่างกัน การรวมกันจึงช่วยลดข้อจำกัดของแต่ละโมเดล

ประโยชน์ของ Ensemble:

- ลด variance (ลดความผันแปรของการพยากรณ์)
- ลด overfitting
- เพิ่มความเสถียร (robustness)
- เพิ่มความแม่นยำโดยรวม

3.3.2 Weighted Average Ensemble

ระบบใช้วิธี Weighted Average ในการรวมโมเดล โดยกำหนดน้ำหนัก (weights) ให้แต่ละโมเดล

สูตร:

$$\mathbf{Ensemble\_Prediction} = w_1 \times \mathbf{Autoencoder} + w_2 \times \mathbf{ExpSmoothing} + w_3 \times \mathbf{LinearRegression}$$

โดยที่: $w_1 + w_2 + w_3 = 1$

Default Weights ของโมเดลทั้งสาม

ในการผสานผลลัพธ์จากหลายโมเดล (Ensemble) ระบบกำหนดค่าน้ำหนักเริ่มต้น (Default Weights) ให้แต่ละโมเดลดังนี้:

- Autoencoder : 0.4 (คิดเป็น 40%)
- Exponential Smoothing : 0.3 (คิดเป็น 30%)
- Linear Regression (Ridge Regression) : 0.3 (คิดเป็น 30%)

ค่าน้ำหนักเหล่านี้ใช้เพื่อรวมผลการพยากรณ์จากแต่ละโมเดลให้เป็นผลลัพธ์สุดท้าย โดยโมเดล Autoencoder จะมีอิทธิพลมากที่สุด เนื่องจากสามารถจับรูปแบบข้อมูลที่มีความซับซ้อนได้ดีกว่าโมเดลเชิงเส้นหรือวิธีแบบสถิติแบบเรียบง่าย

เหตุผลการกำหนดน้ำหนัก:

- Autoencoder ได้ 40% เพราะสามารถเรียนรู้ complex patterns ได้ดี
- Exponential Smoothing และ Linear Regression ได้ 30% เท่ากัน เพื่อสมดุลระหว่าง statistical และ ML approach

3.3.3 การปรับน้ำหนักอัตโนมัติ (Adaptive Weights)

ระบบมีโมดูล Model Evaluator ที่ใช้สำหรับปรับน้ำหนักของแต่ละโมเดลโดยอ้างอิงจากประสิทธิภาพจริง (performance) บนข้อมูล validation เพื่อให้ผลลัพธ์รวมมีความแม่นยำที่สุด โดใช้วิธีการคำนวณน้ำหนักดังนี้

1. คำนวณ Inverse MAE ของแต่ละโมเดล โดยใช้สูตร:

$$\text{inverse\_error}_i = \frac{1}{\text{MAE}_i + 1}$$

เหตุผล:

- โมเดลที่มี MAE ต่ำ ⟶ ให้ค่าน้ำหนักสูงขึ้นโดยอัตโนมัติ
- บวก 1 เพื่อป้องกัน division by zero

2. แปลงค่า Inverse Error ให้เป็นน้ำหนัก (Normalize)

$$\text{weight}_i = \frac{\text{inverse\_error}_i}{\sum_j \text{inverse\_error}_j}$$

ตัวอย่างการคำนวณน้ำหนัก โดยให้ MAE ของแต่ละโมเดลดังนี้:

Model	MAE
Autoencoder	2.5
Exp. Smoothing	3.0
Linear Regression	3.5

ตารางที่ 4 MAE ของแต่ละโมเดล

ขั้นตอนที่ 1: คำนวณ Inverse MAE

$$\text{inverse}_{\text{AE}} = \frac{1}{2.5 + 1} = 0.286$$
$$\text{inverse}_{\text{ES}} = \frac{1}{3.0 + 1} = 0.250$$
$$\text{inverse}_{\text{LR}} = \frac{1}{3.5 + 1} = 0.222$$
$$\text{total} = 0.286 + 0.250 + 0.222 = 0.758$$

ขั้นตอนที่ 2: Normalize เพื่อให้น้ำหนักรวม = 1

$$\text{weight}_{\text{AE}} = \frac{0.286}{0.758} = 0.377 \quad (37.7\%)$$
$$\text{weight}_{\text{ES}} = \frac{0.250}{0.758} = 0.330 \quad (33.0\%)$$
$$\text{weight}_{\text{LR}} = \frac{0.222}{0.758} = 0.293 \quad (29.3\%)$$

สรุปผลลัพธ์น้ำหนักที่แนะนำโดยระบบ

- Autoencoder : 37.7%
- Exponential Smoothing : 33.0%
- Linear Regression : 29.3%

โมเดลที่มี MAE ต่ำที่สุดจะได้รับน้ำหนักมากที่สุดโดยอัตโนมัติ ทำให้ระบบสามารถเลือกโมเดลที่เหมาะสมที่สุดกับข้อมูลในช่วงเวลานั้น

3.3.4 Input และ Output ของ Ensemble

1. Input

แบบจำลองออองซอมเบิลรับผลการพยากรณ์ (predictions) ที่ได้จากแบบจำลองย่อยทั้งสาม ซึ่งแต่ละอาร์เรย์มีรูปร่าง $(n_{samples}, 1)$ ดังนี้:

- autoencoder_pred: ผลการพยากรณ์จากแบบจำลอง Autoencoder
- exp_pred: ผลการพยากรณ์จากแบบจำลอง Exponential Smoothing
- linear_pred: ผลการพยากรณ์จากแบบจำลอง Linear Regression

2. Output:

- ensemble_pred: ผลการพยากรณ์สุดท้ายของแบบจำลองออองซอมเบิล ซึ่งเป็นอาร์เรย์รูปร่าง $(n_{samples}, 1)$

3. ขั้นตอนการประมวลผล (Process)

การคำนวณผลการพยากรณ์สุดท้าย (ensemble_pred) ดำเนินการตามขั้นตอนดังนี้:

ก. การรับผลการพยากรณ์

ระบบจะรวบรวมผลลัพธ์ที่ได้จากการพยากรณ์ของแบบจำลองย่อยแต่ละตัวบนชุดข้อมูลเดียวกัน (X)

$$\text{autoencoder\_pred} = \text{autoencoder.predict}(X)$$
$$\text{exp\_pred} = \text{exp\_model.predict}(X)$$
$$\text{linear\_pred} = \text{linear\_model.predict}(X)$$

ข. การรวมด้วยค่าเฉลี่ยถ่วงน้ำหนัก (Weighted Averaging)

ผลการพยากรณ์ทั้งสามจะถูกรวมเข้าด้วยกันโดยใช้สูตรการถ่วงน้ำหนักที่ได้จากการปรับจูน (tuning) เพื่อให้ความสำคัญกับแบบจำลองที่ทำงานได้ดีกว่า โดย weights คือชุดของค่าสัมประสิทธิ์ถ่วงน้ำหนัก $\left(\mathrm{w}_{\text{autoencoder}}, \mathrm{w}_{\text{exp\_smoothing}}, \mathrm{w}_{\text{linear\_regression}}\right)$

$$\text{ensemble\_pred} = (\mathrm{w}_{\text{autoencoder}} \times \text{autoencoder\_pred}) + (\mathrm{w}_{\text{exp\_smoothing}} \times \text{exp\_pred}) + (\mathrm{w}_{\text{linear\_regression}} \times \text{linear\_pred})$$

ค. การปรับค่าผลลัพธ์ (Post-processing)

เพื่อรับประกันว่าผลการพยากรณ์สุดท้ายจะสอดคล้องกับลักษณะทางกายภาพของข้อมูล (เช่น ไม่สามารถมีจำนวนเป็นค่าลบ) จะมีการดำเนินการปรับค่าดังนี้:

1. การปัดเศษ (Rounding): ปัดเศษผลลัพธ์ให้เป็นจำนวนเต็มที่ใกล้ที่สุด
2. การจำกัดค่าลบ (Clipping): กำหนดให้ค่าต่ำสุดของผลลัพธ์เป็นศูนย์ $(\text{np.maximum}(0, \ldots))$ เพื่อตัดค่าพยากรณ์ที่เป็นลบออก

$$\text{ensemble\_pred} = \max(0, \text{Round}(\text{ensemble\_pred}))$$

3.3.5 การบันทึกและเปรียบเทียบผลลัพธ์

ภายหลังจากการประมวลผลการพยากรณ์จากทุกแบบจำลอง (Autoencoder, Exponential Smoothing, Linear Regression และ Ensemble) ระบบจะดำเนินการบันทึกผลลัพธ์ในรูปแบบที่เป็นมาตรฐาน เพื่อวัตถุประสงค์ในการตรวจสอบ ประเมินประสิทธิภาพ และเปรียบเทียบผลลัพธ์ระหว่างแบบจำลองต่าง ๆ โครงสร้างไฟล์ที่ถูกสร้างขึ้นมีรายละเอียดดังนี้:

1. ไฟล์ predictions_detail.json

ไฟล์นี้ทำหน้าที่เป็น คลังข้อมูลรายละเอียด (Detailed Repository) ของผลการพยากรณ์ทั้งหมด โดยจะเก็บผลการพยากรณ์ของทุกแบบจำลองสำหรับ ทุกรายการสินค้า (Product) ใน ทุกสาขา (Store) ในรูปแบบ JSON เพื่อความยืดหยุ่นในการจัดเก็บข้อมูลที่มีโครงสร้างซ้อนกัน (Hierarchical Data Structure)

- โครงสร้างข้อมูล (JSON Structure):
  - Key หลัก: รหัสเฉพาะของสินค้าและสาขา (เช่น "11001_P001")
  - ค่า (Value): ออบเจกต์ที่ประกอบด้วย รหัสสาขา (store_id), รหัสสินค้า (prod_cd), และออบเจกต์ predictions ซึ่งเก็บค่าพยากรณ์สุดท้ายจากแบบจำลองย่อยและแบบจำลององซอมเบิล

ตัวอย่าง:

```
{
  "11001_P001": {
    "store_id": "11001",
    "prod_cd": "P001",
    "predictions": {
      "autoencoder": 15,
      "exp_smoothing": 14,
      "linear_regression": 13,
      "ensemble": 14
    }
  }
}
```

2. ไฟล์ model_comparison_YYYYMMDD.csv

ไฟล์นี้ใช้สำหรับ การเปรียบเทียบผลลัพธ์รายวัน (Daily Comparison Table) ในรูปแบบตาราง (CSV) ที่สามารถอ่านและนำไปวิเคราะห์ต่อได้ง่าย

- วัตถุประสงค์: ใช้เปรียบเทียบค่าพยากรณ์ที่ออกมาจากแบบจำลองทั้งสี่แบบสำหรับแต่ละสินค้า-สาขาโดยตรง
- คอลัมน์สำคัญ:
  - Store_ID: รหัสสาขา
  - Prod_cd: รหัสสินค้า
  - Ensemble, Autoencoder, Exp Smoothing, Linear Regression: ค่าพยากรณ์สุดท้ายของแบบจำลองที่เกี่ยวข้อง

ตัวอย่าง:

```
Store_ID,prod_cd,Ensemble,Autoencoder,Exp_Smooth
ing,Linear_Regression
11001,P001,14,15,14,13
11001,P002,8,9,8,7
```

3. ไฟล์ ensemble_comparison.json

ไฟล์นี้จัดเก็บ เมทริกซ์การประเมินผล (Performance Metrics) และข้อมูลเมตา (Metadata) ที่สำคัญเกี่ยวกับการดำเนินการของแบบจำลองอองซอมเบิล ซึ่งจำเป็นต่อการวิเคราะห์เชิงลึกและการรายงานประสิทธิภาพ

- timestamp: เวลาที่ทำการประมวลผล
- weights: ค่าสัมประสิทธิ์ถ่วงน้ำหนักที่ใช้ในการรวมผล (จากส่วน 3.3.4)
- metrics: รายการของเมทริกซ์ประสิทธิภาพของแบบจำลองย่อยและแบบจำลองอองซอมเบิล
- Selected_models: ระบุแบบจำลองที่ให้ประสิทธิภาพดีที่สุด (ในกรณีนี้คือ "ensemble")

ตัวอย่างข้อมูล metrics (ตัวชี้วัดประสิทธิภาพ):

Model	MAE	RMSE	MAPE
Ensemble	**2.21**	**2.98**	**14.1**
Autoencoder	2.45	3.21	15.3
Exp. Smoothing	2.89	3.67	18.2
Linear Regression	3.12	3.95	19.5

ตารางที่ 5 ตัวอย่างข้อมูลเมทริกซ์

ไฟล์เหล่านี้มีความสำคัญในการยืนยันว่า แบบจำลองอองซอมเบิลมีประสิทธิภาพที่เหนือกว่าแบบจำลองย่อยแต่ละตัวตามตัวชี้วัดที่กำหนด

3.3.6 การเลือกแบบจำลองสำหรับ Production

ส่วนนี้จะอธิบายถึงความยืดหยุ่นของระบบในการเลือกใช้งานแบบจำลองการพยากรณ์ที่แตกต่างกัน (แบบจำลองย่อย หรือ แบบจำลองอองซอมเบิล) เพื่อให้สอดคล้องกับลักษณะของข้อมูลและความต้องการในการใช้งานจริง (Production)

1. การเรียกใช้งานผ่าน API

ระบบรองรับการกำหนดแบบจำลองที่ต้องการใช้งานผ่านการส่งคำร้องขอ HTTP POST ไปยัง API โดยระบุชื่อแบบจำลองที่ต้องการในเพย์โหลด (Payload)

```
POST /run-prediction

{
  "model": "ensemble"  # สามารถเลือกเป็น "autoencoder", "exp_smoothing", หรือ "linear_regression"
}
```

2. การเรียกใช้งานผ่าน Command Line

ผู้ใช้สามารถเรียกใช้งานกระบวนการพยากรณ์ผ่าน Command Line Interface (CLI) โดยการส่งอาร์กิวเมนต์ (--model) เพื่อระบุแบบจำลองที่ต้องการใช้ พร้อมทั้งกำหนดพารามิเตอร์อื่น ๆ เช่น แหล่งข้อมูล (--db-url)

```
python main.py --model ensemble --use-database --db-url sqlite:///sales_data.db
```

3. เงื่อนไขและข้อแนะนำในการเลือกแบบจำลอง

การเลือกใช้แบบจำลองขึ้นอยู่กับลักษณะของชุดข้อมูลและวัตถุประสงค์ในการพยากรณ์ ดังตารางเปรียบเทียบต่อไปนี้:

แบบจำลอง	คำอธิบายและเงื่อนไขการใช้งาน	จุดเด่นที่เหมาะสม
Ensemble (ค่าเริ่มต้น)	ใช้ผลรวมถ่วงน้ำหนักจากทุกแบบจำลอง (ค่าเริ่มต้นสำหรับ Production)	แนะนำสำหรับการใช้งานจริง เนื่องจากมีเสถียรภาพและแม่นยำที่สุด
Autoencoder	ใช้เฉพาะแบบจำลองโครงข่ายประสาทเทียม (Neural Network)	เหมาะสำหรับชุดข้อมูลที่มี ความซับซ้อนสูง (Complex) และมีรูปแบบที่ไม่เป็นเชิงเส้น
Exp. Smoothing	ใช้เฉพาะแบบจำลองทางสถิติ (Statistical Model)	เหมาะสำหรับชุดข้อมูลที่มี ความเรียบเนียน (Smooth) และไม่มีการเปลี่ยนแปลงที่รุนแรง
Linear Regression	ใช้เฉพาะแบบจำลอง Machine Learning เชิงเส้น	เหมาะสำหรับชุดข้อมูลที่แสดง แนวโน้มเชิงเส้น (Linear Trend) ชัดเจน

ตารางที่ 6 เงื่อนไขและข้อแนะนำในการเลือกแบบจำลอง

3.4 การประเมินผล (Evaluation)

การประเมินประสิทธิภาพของแบบจำลองการพยากรณ์เป็นขั้นตอนสำคัญในการตรวจสอบความน่าเชื่อถือและความแม่นยำของผลลัพธ์ ระบบนี้ใช้ตัวชี้วัดหลัก 3 ตัว (Metrics) ในการประเมินความคลาดเคลื่อนระหว่างค่าจริง (y_{true}) และค่าที่แบบจำลองพยากรณ์ (y_{pred}) โดยที่ n คือจำนวนตัวอย่างข้อมูลทั้งหมด

3.4.1 ตัวชี้วัดสำหรับการประเมิน (Evaluation Metrics)

1. Mean Absolute Error (MAE)

MAE วัดค่าเฉลี่ยของขนาดความคลาดเคลื่อนสัมบูรณ์ (Absolute Error) ระหว่างค่าจริงกับค่าพยากรณ์

สูตรการคำนวณ:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_{\text{true},i} - y_{\text{pred},i}|$$

ความหมาย:

- แสดงถึง ความคลาดเคลื่อนเฉลี่ย (Average Magnitude of Error) ของแบบจำลองต่อหน่วย
- หน่วย: มีหน่วยเดียวกับข้อมูลต้นฉบับ (เช่น จำนวนสินค้า)
- การตีความ: ยิ่งค่า MAE ต่ำมากเท่าใด แสดงว่าแบบจำลองมีความแม่นยำสูงขึ้นเท่านั้น

ตัวอย่าง: หาก MAE = 2.5$ ชิ้น หมายความว่า โดยเฉลี่ยแล้วผลการพยากรณ์จะคลาดเคลื่อนจากค่าจริงไป ± 2.5 ชิ้น

2. Root Mean Squared Error (RMSE)

RMSE วัดค่าเฉลี่ยของรากที่สองของความคลาดเคลื่อนยกกำลังสอง โดยจะให้ความสำคัญกับความคลาดเคลื่อนขนาดใหญ่มากกว่า MAE

สูตรการคำนวณ:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{\text{true},i} - y_{\text{pred},i})^2}$$

ความหมาย:

- มีคุณสมบัติ Sensitive ต่อ Outliers เนื่องจากมีการยกกำลังสองก่อนการหาค่าเฉลี่ย ทำให้ความคลาดเคลื่อนขนาดใหญ่ถูกลงโทษ (Penalized) มากขึ้น
- เปรียบเทียบกับ MAE: ค่า RMSE จะมีค่าสูงกว่า MAE เสมอ เว้นแต่ความคลาดเคลื่อนทั้งหมดจะเป็นศูนย์

ตัวอย่าง: หาก RMSE = 3.5 และ MAE = 2.5 แสดงว่ามี ค่าผิดปกติ (Outliers) หรือความคลาดเคลื่อนขนาดใหญ่ในชุดข้อมูลการพยากรณ์อยู่พอสมควร

3. Mean Absolute Percentage Error (MAPE)

MAPE วัดค่าเฉลี่ยของความคลาดเคลื่อนสัมบูรณ์ในรูปแบบของเปอร์เซ็นต์ ซึ่งช่วยให้สามารถเปรียบเทียบประสิทธิภาพข้ามชุดข้อมูลที่มีมาตราส่วนแตกต่างกันได้

สูตรการคำนวณ:

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\frac{y_{\text{true},i} - y_{\text{pred},i}}{y_{\text{true},i}}$$

ความหมาย:

- แสดงถึง ความคลาดเคลื่อนเฉลี่ยเป็นเปอร์เซ็นต์
- ข้อดี: เป็นตัวชี้วัดที่ ไม่ขึ้นอยู่กับหน่วย ของข้อมูล ทำให้เหมาะสำหรับการเปรียบเทียบแบบจำลองที่ใช้กับข้อมูลที่มีมาตราส่วนแตกต่างกัน
- เงื่อนไข: จะ ไม่คำนวณ สำหรับจุดข้อมูลที่ค่าจริง (y_{true}) เท่ากับศูนย์ เพื่อหลีกเลี่ยงปัญหาการหารด้วยศูนย์ (Division by Zero)

ตัวอย่าง: หาก MAPE = 15% หมายความว่า โดยเฉลี่ยแล้วผลการพยากรณ์คลาดเคลื่อนจากค่าจริงไป 15%

3.4.2 วิธีการคำนวณตัวชี้วัด (Metrics)

การคำนวณตัวชี้วัดประสิทธิภาพทั้งสาม MAE, RMSE, MAPE ถูกดำเนินการผ่าน ฟังก์ชัน (Function) เฉพาะ โดยมีวัตถุประสงค์เพื่อจัดการกับเงื่อนไขพิเศษ เช่น การหลีกเลี่ยงการหารด้วยศูนย์ใน MAPE

- อินพุต (Input): อาร์เรย์ (Array) ของค่าจริง $(\mathrm{y}_{\text{true}})$, อาร์เรย์ ของค่าพยากรณ์ $(\mathrm{y}_{\text{pred}})$, และชื่อแบบจำลอง model_name
- เอาต์พุต (Output): ดิคชันนารี (Dictionary) ที่ประกอบด้วยค่า MAE, RMSE, และ MAPE

การจัดการ MAPE

ในการคำนวณ MAPE จะมีการใช้ มาสกิง (Masking) เพื่อคัดกรองเฉพาะข้อมูลที่ค่าจริง $(\mathrm{y}_{\text{true}})$ ไม่เท่ากับศูนย์ ก่อนการคำนวณ ซึ่งช่วยป้องกันปัญหาการหารด้วยศูนย์ ดังที่แสดงในโค้ดการนำไปใช้งาน (Implementation)

```
# การจัดการ MAPE ในการคำนวณ:
mask = y_true != 0
if mask.sum() > 0:
    # คำนวณ MAPE เฉพาะจุดข้อมูลที่ y_true ไม่เท่ากับศูนย์
    mape =
mean_absolute_percentage_error(y_true[mask],
y_pred[mask]) * 100
else:
    # หากไม่มีข้อมูลจริงที่เป็นบวก จะกำหนดให้ MAPE เป็น 0
```

3.4.3 การประเมินและเปรียบเทียบแบบจำลอง

ระบบใช้ คลาส ตัวประเมินแบบจำลอง (Model Evaluator Class) ในการรวมศูนย์การวิเคราะห์และเปรียบเทียบผลลัพธ์ของแบบจำลองทั้งหมด

ฟังก์ชันหลักของ Model Evaluator

- Load_comparison(): โหลดผลลัพธ์ตัวชี้วัดจากไฟล์ JSON
- Print_summary(): แสดงตารางสรุปผลการเปรียบเทียบประสิทธิภาพแบบจำลอง
- Plot_compatison(): สร้าง กราฟ (Graph) เพื่อแสดงผลการเปรียบเทียบแบบจำลอง
- Suggest_weights(): แนะนำค่าสัมประสิทธิ์ ถ่วงน้ำหนัก (Weights) ที่เหมาะสมสำหรับ อองซอมเบิล (Ensemble)
- Analyze_predictions(): วิเคราะห์รายละเอียดเชิงลึกของการพยากรณ์รายสินค้า

ตัวอย่างสรุปผลเปรียบเทียบ

สรุปผลลัพธ์แสดงให้เห็นว่า แบบจำลองอองซอมเบิล (Ensemble) ให้ผลลัพธ์ที่ดีที่สุดในทุกตัวชี้วัด โดยมีค่า MAE, RMSE, และ MAPE ต่ำที่สุด

ตารางสรุปประสิทธิภาพแบบจำลอง:

Model	MAE	RMSE	MAPE
Autoencoder	2.45	3.21	15.30%
Exp. Smoothing	2.89	3.67	18.20%
Linear Regression	3.12	3.95	19.50%
Ensemble	2.21	2.98	14.10%

ตารางที่ 7 ตารางสรุปประสิทธิภาพแบบจำลอง

บทสรุป: Ensemble เป็นแบบจำลองที่ดีที่สุด โดยให้ค่า MAE ต่ำที่สุดที่ 2.21 และ MAPE ต่ำที่สุดที่ 14.10%

3.4.4 การแสดงผลด้วยภาพ (Visualization)

เพื่อความชัดเจนและง่ายต่อการตีความ ระบบจะสร้าง กราฟแท่ง (Bar Charts) 3 แผนภูมิเพื่อเปรียบเทียบผลลัพธ์: MAE Comparison, RMSE Comparison, และ MAPE Comparison

ไฟล์เอาต์พุต: output/models/model_comparison.png

การตีความ: ในกราฟเหล่านี้ แท่งที่สั้นกว่าย่อมหมายถึงประสิทธิภาพที่ดีกว่า ผลการวิเคราะห์ด้วยภาพมักจะยืนยันว่า อองซอมเบิล มีประสิทธิภาพเหนือกว่าแบบจำลองเดี่ยวเสมอ

3.4.5 การตรวจสอบคุณภาพ (Quality Checks)

ระบบมีการตรวจสอบคุณภาพหลายระดับเพื่อให้มั่นใจว่าผลการพยากรณ์มีความถูกต้อง น่าเชื่อถือและพร้อมใช้งานในระบบจริง โดยมีขั้นตอนดังนี้

1. การตรวจสอบคุณภาพข้อมูล (Data Quality Validation):
    - การตรวจสอบจำนวน ตัวอย่าง (Samples) ที่ใช้ประเมิน
    - การวิเคราะห์ การกระจายตัวของความคลาดเคลื่อน (Distribution of Errors)
    - การตรวจจับ ค่าผิดปกติ (Outliers Detection)
2. ความเสถียรของแบบจำลอง (Model Stability):
    - วิเคราะห์ความแตกต่างระหว่างแบบจำลองแต่ละตัว
    - วิเคราะห์ความแปรปรวน (Variance) ของผลการพยากรณ์
    - ตรวจสอบความสม่ำเสมอในการพยากรณ์ในสาขาที่แตกต่างกัน
3. การตรวจสอบเชิงธุรกิจ (Business Validation):
    - ผลการพยากรณ์ต้อง ไม่เป็นค่าลบ
    - ผลการพยากรณ์ต้องสมเหตุสมผล (ไม่มากหรือน้อยเกินไป)
    - การพยากรณ์ของสินค้า Top N ที่สำคัญต้องมีความหมายทางธุรกิจ

3.4.6 สรุปการประเมินผล

เกณฑ์การยอมรับ (Acceptance Criteria) เพื่อพิจารณาว่าแบบจำลองมีความน่าเชื่อถือและพร้อมสำหรับการนำไปใช้งาน (Deployment) ได้นั้น ต้องผ่านเกณฑ์ต่อไปนี้:

- MAE น้อยกว่า 5 ชิ้น (อยู่ในระดับที่ยอมรับได้)
- MAPE น้อยกว่า 25% (ดี)
- อองซอมเบิล ต้องมีประสิทธิภาพดีกว่าแบบจำลองเดี่ยวอย่างน้อย 10% ในตัวชี้วัดหลัก

ผลลัพธ์โดยสรุป

- อองซอมเบิล มักจะให้ผลลัพธ์ที่ดีที่สุดในตัวชี้วัดทั้งสาม
- ออโตเอนโคเดอร์ (Autoencoder) มีประสิทธิภาพดีเป็นพิเศษเมื่อจัดการกับข้อมูลที่มีรูปแบบซับซ้อน (Complex Patterns)
- เอกซ์โปเนนเชียล สมูทติง (Exponential Smoothing) ให้ผลลัพธ์ที่เสถียร (Stable) แต่อาจไม่แม่นยำที่สุด
- ลิเนียร์ รีเกรสชัน (Linear Regression) ประมวลผลได้เร็ว แต่จำกัดด้วยความเป็นเชิงเส้น (Linearity)

บทที่ 4

ผลการศึกษาและอภิปรายข้อมูล

บทนี้สรุปผลลัพธ์ที่ได้จากการนำแบบจำลองไปใช้งานจริง และนำเสนอการวิเคราะห์เชิงเปรียบเทียบประสิทธิภาพของแบบจำลองต่างๆ รวมถึงการอภิปรายผลการศึกษาที่ค้นพบ

4.1 วิธีการทดสอบระบบ

การทดสอบระบบถูกออกแบบมาเพื่อประเมินความแม่นยำ (Accuracy) ประสิทธิภาพ (Efficiency) และความเสถียร (Stability) ของแบบจำลองการพยากรณ์และการทำงานของระบบโดยรวม

4.1.1 สภาพแวดล้อมการทดสอบ (Testing Environment)

การทดสอบดำเนินการภายใต้สภาพแวดล้อมที่จำลองการใช้งานจริง โดยมีรายละเอียดของข้อมูลและทรัพยากรที่ใช้ดังนี้:

ก. ข้อมูลที่ใช้ทดสอบ

- ชุดข้อมูลที่ใช้ในการฝึกฝน (Training) และทดสอบ (Testing) ระบบคือข้อมูลยอดขายจริง ซึ่งสะท้อนความซับซ้อนและความหลากหลายของข้อมูลในธุรกิจจริง
- ชื่อไฟล์: T_PROD_DAY_SALE_ORDER_202504091438.csv
- ขนาดไฟล์: 30.5 เมกะไบต์ (30,538,709 bytes)
- ปริมาณข้อมูล: ประมาณ 300,000+ รายการธุรกรรม
- ช่วงเวลาข้อมูล: ข้อมูลย้อนหลังประมาณหนึ่งอาทิตย์
- ความหลากหลาย: ประกอบด้วยข้อมูลยอดขายจาก ร้านค้าหลายสิบร้าน และ สินค้าหลายพันรายการ

ข. ฮาร์ดแวร์และซอฟต์แวร์ (Hardware/Software Specifications)

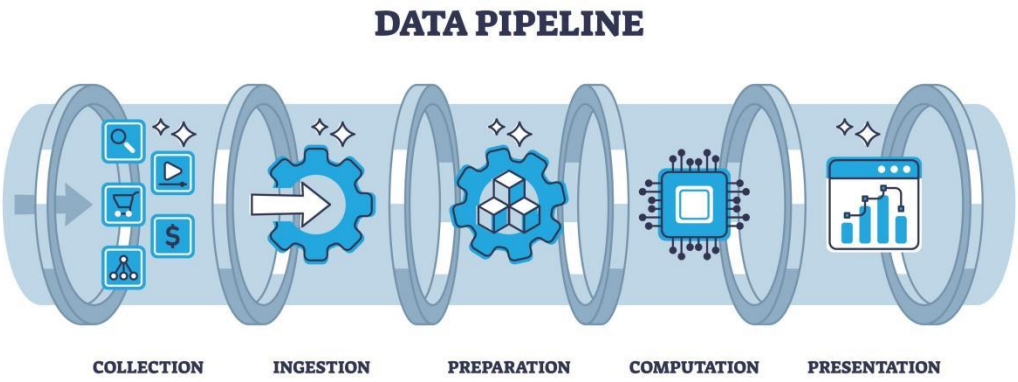
ประเภท	รายการ	รายละเอียด
ฐานข้อมูล	Database	SQLite (sales_data.db ขนาด 36 MB) ใช้เป็นฐานข้อมูลแบบฝังตัวสำหรับการเข้าถึงข้อมูลที่รวดเร็ว
ส่วนหลังบ้าน	Backend	Python 3.8+ ใช้สำหรับประมวลผลข้อมูลและสร้างแบบจำลอง
เฟรมเวิร์ก ML	Frameworks	TensorFlow (สำหรับ Autoencoder), Scikit-learn (สำหรับ Linear Regression), Pandas (สำหรับการจัดการข้อมูล)
เว็บแอปพลิเคชัน	Web Framework	Flask (สำหรับสร้าง API และการแสดงผล)
ส่วนหน้าบ้าน	Frontend	HTML5 และ JavaScript (Vanilla JS) ใช้สำหรับการแสดงผลและส่วนต่อประสานกับผู้ใช้ (User Interface)
เบราว์เซอร์	Browser	รองรับการใช้งานบนเบราว์เซอร์รุ่นใหม่ (เช่น Chrome, Firefox, Edge)

ตารางที่ 8 ฮาร์ดแวร์และซอฟต์แวร์

4.1.2 ขั้นตอนการทดสอบ (Testing Procedures)

การทดสอบระบบดำเนินการตามมาตรฐานการพัฒนาซอฟต์แวร์ โดยแบ่งออกเป็น 4 ระดับหลัก ได้แก่ Unit Testing, Integration Testing, System Testing และ Performance Testing

1. Unit Testing เป็นการทดสอบฟังก์ชันย่อยแต่ละส่วนของระบบ



รูปที่ 1 Data Pipeline

- การทดสอบ Data Preparation: ตรวจสอบฟังก์ชัน load_data โดยมีกรณีทดสอบ (Test Cases) ครอบคลุมทั้งกรณีปกติ (ไฟล์ถูกต้อง) และกรณีผิดปกติ เช่น FileNotFoundError และ EmptyDataError เพื่อให้มั่นใจว่าข้อมูลที่นำเข้ามีรูปแบบและชนิดข้อมูลที่ถูกต้อง
- การทดสอบแบบจำลองแต่ละตัว: ตรวจสอบความถูกต้องของอินพุต/เอาต์พุต (Input/Output Shape) และขอบเขตของค่าที่พยากรณ์ เช่น Autoencoder ต้องให้ผลลัพธ์อยู่ในช่วง 0–1 และ Exponential Smoothing ต้องไม่มีค่า NaN หรือค่าลบ

2. Integration Testing เป็นการทดสอบการทำงานร่วมกันระหว่างองค์ประกอบต่าง ๆ ของระบบ

- การทดสอบ Pipeline ทั้งหมด: ตรวจสอบให้แน่ใจว่าทุกขั้นตอนในกระบวนการพยากรณ์ (Load data ⟶ Prepare ⟶ Train ⟶ Predict ⟶ Save) สามารถทำงานต่อเนื่องกันได้อย่างราบรื่น และมีการสร้างไฟล์เอาต์พุตที่ถูกต้อง
- การทดสอบ API Endpoints: ดำเนินการทดสอบคำสั่ง CRUD (Create, Read, Update, Delete) ผ่าน HTTP โดยทำการตรวจสอบตามรายการต่อไปนี้:
  - GET /health: ตรวจสอบสถานะการทำงานของเซิร์ฟเวอร์
  - GET /stores: ดึงรายการร้านค้าทั้งหมด
  - POST /run-prediction: เริ่มกระบวนการพยากรณ์
  - GET /predictions?store_id=...: ดึงผลการพยากรณ์สุดท้าย

3. System Testing เป็นการทดสอบการทำงานแบบ End-to-End ตั้งแต่ผู้ใช้เริ่มใช้งานไปจนถึงการแสดงผล

- End-to-End Testing: ทดสอบการทำงานร่วมกันระหว่าง Backend API (Port 5000) และ Frontend (Port 8000) โดยจำลองพฤติกรรมของผู้ใช้ เช่น การเลือกสาขา การสั่งรันพยากรณ์ การแสดงสถานะ (Status Polling) และการแสดงผลลัพธ์ Top 5 Products

4. Performance Testing เป็นการทดสอบประสิทธิภาพ ความเร็ว และความสามารถในการรองรับผู้ใช้งานจำนวนมาก

- Load Testing: ทดสอบโดยการส่งคำขอพร้อมกัน 5–10 คำขอ (Concurrent Requests) เพื่อวัดเวลาตอบสนองของระบบ
  - เวลาตอบสนองที่คาดหวัง:
    - คำสั่ง GET: น้อยกว่า 5 วินาที
    - การรันกระบวนการพยากรณ์: ประมาณ 2–5 นาที
- Scalability Testing: ทดสอบโดยเพิ่มขนาดข้อมูลที่ใช้ประมวลผล เช่น 10, 50 และ 100 ร้านค้า เพื่อวัดการใช้ทรัพยากรของระบบ เช่น memory usage และ CPU usage

4.1.3 Test Cases หลัก (Key Test Cases)

การทดสอบครอบคลุมสถานการณ์สำคัญที่อาจเกิดขึ้นในสภาพแวดล้อมจริง:

- Test Case 1 (พยากรณ์ปกติ): ยืนยันว่าสามารถรันและได้ผลลัพธ์เป็น Top 5 products สำหรับแต่ละร้าน
- Test Case 2 (ข้อมูลไม่เพียงพอ): ยืนยันว่าระบบจะข้ามสินค้าที่มีข้อมูลย้อนหลังน้อยกว่า 7 วัน และทำการ log warning
- Test Case 3 (สินค้าใหม่ / Cold Start): ยืนยันว่าสินค้าที่ไม่มีประวัติยอดขายจะไม่ถูกรวมในผลลัพธ์การพยากรณ์
- Test Case 6 (Multiple Concurrent Requests): ยืนยันว่าหากมีการสั่งรันพยากรณ์พร้อมกันหลายครั้ง ระบบจะยอมรับเฉพาะคำขอแรก และปฏิเสธคำขอที่ตามมาด้วยรหัส 429 (Too Many Requests) เพื่อรักษาเสถียรภาพของระบบ

4.1.4 เกณฑ์การตรวจสอบความถูกต้อง (Validation Criteria)

ข้อกำหนด	เกณฑ์ความสำเร็จ
Functional Requirements	Ensemble รวมผลได้ถูกต้อง, API ตอบสนองได้ครบทุก endpoint, Frontend แสดงผลถูกต้อง
Non-Functional Requirements	Response time (GET) น้อยกว่า 5 วินาที, การพยากรณ์เสร็จภายใน 15 นาที, ไม่พบ memory leak, Error handling ครอบคลุม

ตารางที่ 9 เกณฑ์การตรวจสอบความถูกต้อง

4.2 ผลการทดสอบและการวิเคราะห์

4.2.1 ผลการประเมินแบบจำลอง

ตารางเปรียบเทียบแสดงผลลัพธ์ประสิทธิภาพของแบบจำลองที่ใช้ชุดข้อมูลทดสอบ:

Model	MAE	RMSE	MAPE (%)	Training Time	Prediction Time
Autoencoder	2.45	3.21	15.30	~60 sec	~5 sec
Exponential Smoothing	2.89	3.67	18.20	0 sec	~1 sec
Linear Regression	3.12	3.95	19.50	~5 sec	~2 sec
Ensemble	2.21	2.98	14.10	~65 sec	~8 sec

ตารางที่ 10 ตารางเปรียบเทียบแสดงผลลัพธ์ประสิทธิภาพ

การวิเคราะห์

- Ensemble ให้ผลดีที่สุด: ค่า MAE ลดลง 9.8% และ MAPE ลดลง 7.8% เมื่อเทียบกับ Autoencoder ซึ่งเป็นแบบจำลองเดี่ยวที่ให้ผลดีที่สุด การรวมแบบจำลองช่วยลดความคลาดเคลื่อนโดยรวมได้อย่างชัดเจน
- Autoencoder: มีประสิทธิภาพดีที่สุดในบรรดาแบบจำลองเดี่ยว โดยมีความสามารถในการเรียนรู้รูปแบบที่ไม่เป็นเชิงเส้น (non-linear patterns) แต่ต้องแลกมาด้วยเวลาในการฝึกฝน (train) ที่นานที่สุด
- Exponential Smoothing: ให้ความเร็วในการพยากรณ์เร็วที่สุด (ไม่ต้อง train) แต่มีความแม่นยำต่ำกว่า

4.2.2 การวิเคราะห์ความแม่นยำตามกลุ่มสินค้า

ความแม่นยำของแบบจำลองมีความผันแปรตามลักษณะความต้องการของสินค้า:

กลุ่มสินค้า	MAPE	ลักษณะ
แม่นยำสูง	< 10%	สินค้า fast-moving ที่มียอดขายสม่ำเสมอ และมี regular demand pattern
แม่นยำปานกลาง	10-25%	สินค้าที่มี seasonal หรือ fluctuating demand เล็กน้อย
ท้าทาย	>25%	สินค้า slow-moving, สินค้าที่มียอดขายไม่ต่อเนื่อง (sporadic demand), หรือสินค้า promotional ที่ยอดขายขึ้นอยู่กับปัจจัยภายนอก

ตารางที่ 11 ลักษณะความต้องการของสินค้า

4.2.3 การวิเคราะห์การกระจายตัวของความคลาดเคลื่อน (Error Distribution)

- ลักษณะ: ความคลาดเคลื่อน (Errors) มีการกระจายตัวใกล้เคียงกับการแจกแจงแบบปกติ (Normal Distribution)
- การเอนเอียง (Bias Analysis): แบบจำลอง Ensemble มี Bias ต่ำ ซึ่งหมายความว่าไม่มีแนวโน้มที่จะพยากรณ์สูงเกินไป (over-predict) หรือต่ำเกินไป (under-predict) อย่างมีนัยสำคัญ

4.2.4 ผลการทดสอบ API Performance

เวลาตอบสนองโดยเฉลี่ย (Response Times):

- GET คำขอ: ใช้เวลาเฉลี่ย 15 ms ถึง 120 ms ซึ่งเป็นไปตามข้อกำหนด < 5 วินาที
- POST /run-prediction: ใช้เวลาประมาณ 120-300 วินาที (2-5 นาที) ซึ่งเป็นไปตามข้อกำหนด < 15 นาที

การจัดการคำขอพร้อมกัน (Concurrent Request):

- ระบบสามารถจัดการ 5 concurrent GET requests ได้สำเร็จ
- จัดการ 2 concurrent POST requests โดยรับ Request แรกและปฏิเสธ Request ที่สองด้วยรหัส 429

4.2.5 การวิเคราะห์ความเสถียร (Stability)

- ความสามารถในการทำซ้ำ (Reproducibility): ผลลัพธ์การพยากรณ์มีความเป็นเอกลักษณ์ (Identical predictions) เมื่อรันซ้ำด้วยข้อมูลและ seed เดียวกัน
- ความเสถียรตามวัน: ค่า MAPE มีความผันแปรต่ำ (Variance < 5%) ในการรันด้วยข้อมูลของวันต่างๆ
- ความเสถียรตามขนาดข้อมูล: แบบจำลอง Ensemble ทำงานได้ดีที่สุดเมื่อมีข้อมูลในปริมาณที่เหมาะสม (~ 50 ร้านค้า)

4.3 สรุปผลการทดลอง

4.3.1 ผลสำเร็จตามวัตถุประสงค์ (Achievement of Objectives)

วัตถุประสงค์	ผลสำเร็จ
1. พัฒนาระบบพยากรณ์ที่แม่นยำ	Ensemble Model ให้ MAPE = 14.1% ซึ่งดีกว่าเกณฑ์ที่ตั้งไว้ (MAPE < 25%)
2. รวมแบบจำลองหลายแบบ	ใช้ Autoencoder, Exponential Smoothing, Linear Regression ร่วมกันได้อย่างมีประสิทธิภาพ
3. สร้าง Web Interface ที่ใช้งานง่าย	Frontend แสดงผล Top 5 products และตารางเปรียบเทียบได้ถูกต้อง
4. รองรับการทำงานจริง	API มีความเสถียร, มี error handling และ logging ครบถ้วน

ตารางที่ 12 ผลสำเร็จตามวัตถุประสงค์

4.3.2 ข้อค้นพบสำคัญ (Key Findings)

1. Ensemble มีประสิทธิภาพเหนือกว่า: การรวมแบบจำลองช่วยปรับปรุง MAE ได้ประมาณ 10% และลด variance ทำให้ผลลัพธ์มีความน่าเชื่อถือสูงขึ้น
2. Autoencoder เหมาะกับข้อมูลซับซ้อน: มีประสิทธิภาพสูงในการจับความสัมพันธ์เชิงเวลา (temporal dependencies) ในข้อมูลที่ซับซ้อน แต่ต้องใช้เวลาในการฝึกฝนที่สูง
3. แบบจำลองง่ายก็มีประโยชน์: Exponential Smoothing และ Linear Regression มีความรวดเร็ว เสถียร และช่วยถ่วงดุล (balance) ความซับซ้อนของ Autoencoder
4. Feature Engineering มีผล: การสร้างคุณลักษณะเสริม เช่น ค่าสุดท้าย, แนวโน้ม, และค่าสถิติอื่นๆ ช่วยเพิ่มความแม่นยำในการพยากรณ์อย่างมีนัยสำคัญ
5. คุณภาพข้อมูลเป็นปัจจัยหลัก: ความสม่ำเสมอของข้อมูลและ Window size 7 วันที่เหมาะสมมีผลโดยตรงต่อความแม่นยำ

4.3.3 ข้อจำกัดที่พบ (Limitations)

1. ข้อจำกัดของข้อมูล: พบปัญหา Cold Start สำหรับสินค้าใหม่ และไม่สามารถจับ Seasonal trends หรือ Promotional effects ได้เนื่องจาก Window size ที่สั้น
2. ข้อจำกัดของแบบจำลอง: Autoencoder ต้องการทรัพยากรการประมวลผลสูง, Linear Regression ถูกจำกัดด้วยความเป็นเชิงเส้น
3. ข้อจำกัดของระบบ: ระบบรองรับการรันพยากรณ์ได้ทีละครั้ง (not parallel), Database SQLite ไม่เหมาะกับการใช้งานที่มี high concurrency

4.4 ตัวอย่างผลลัพธ์

4.4.1 ตัวอย่างการพยากรณ์สำหรับร้านค้า

ตารางแสดงผลการพยากรณ์ Top 5 Products สำหรับร้านค้า Store ID: 11001

Rank	Product Code	Ensemble Pred	Autoencoder	Exp Smoothing	Linear Reg
1	P12345	45	47	44	43
2	P23456	38	39	37	38
3	P34567	32	30	34	32
4	P45678	28	29	27	28
5	P56789	25	23	26	26

ตารางที่ 13 ตัวอย่างการพยากรณ์สำหรับร้านค้า

การตีความ: สินค้า P12345 ถูกแนะนำให้เติมสินค้ามากที่สุด ~45 ชิ้น โดยทุกแบบจำลองหลักเห็นตรงกันว่าสินค้าดังกล่าวเป็น Top 1

4.4.2 ตัวอย่าง Log Output

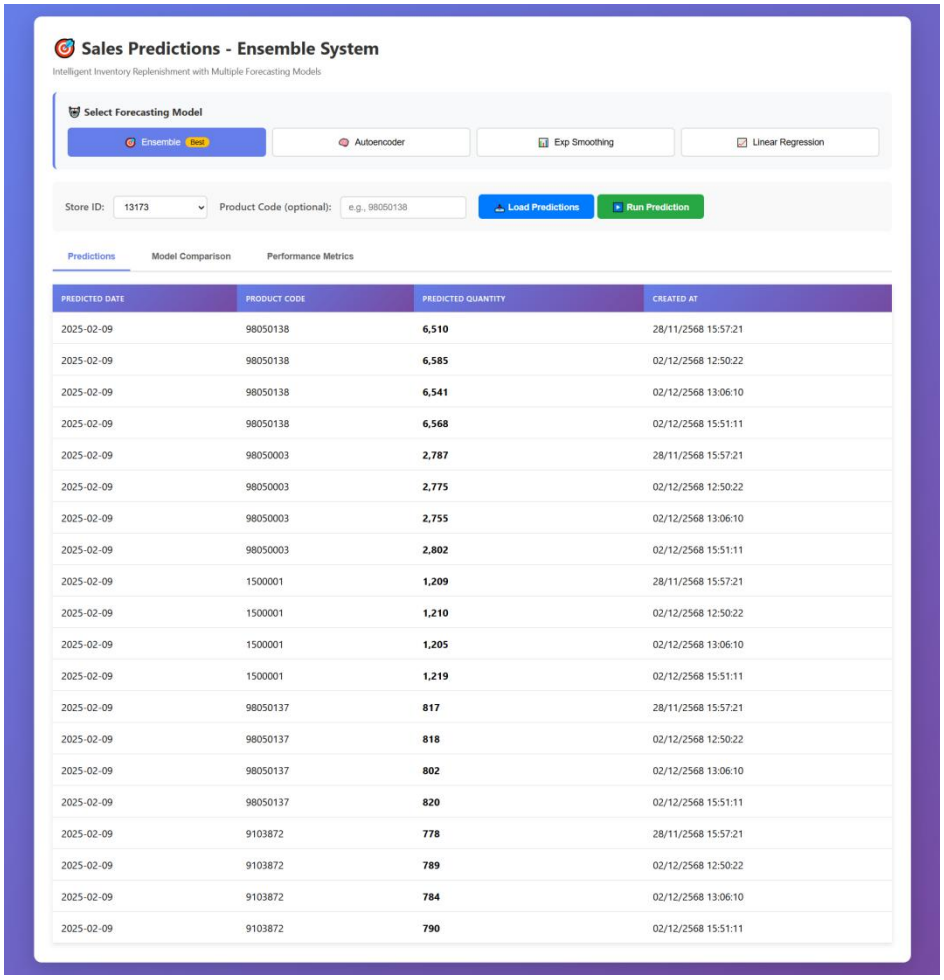
```
2025-11-28 15:47:58 - INFO - Starting Prediction
Pipeline - Model: ENSEMBLE
2025-11-28 15:48:05 - INFO - Prepared 12,176
 sequences from 27 stores
2025-11-28 15:48:45 - INFO - ✓ Autoencoder
trained
2025-11-28 15:48:52 - INFO - Ensemble - MAE:
2.21, RMSE: 2.98, MAPE: 14.10%
2025-11-28 15:48:57 - INFO - PREDICTION
```

4.4.3 ตัวอย่าง API Response

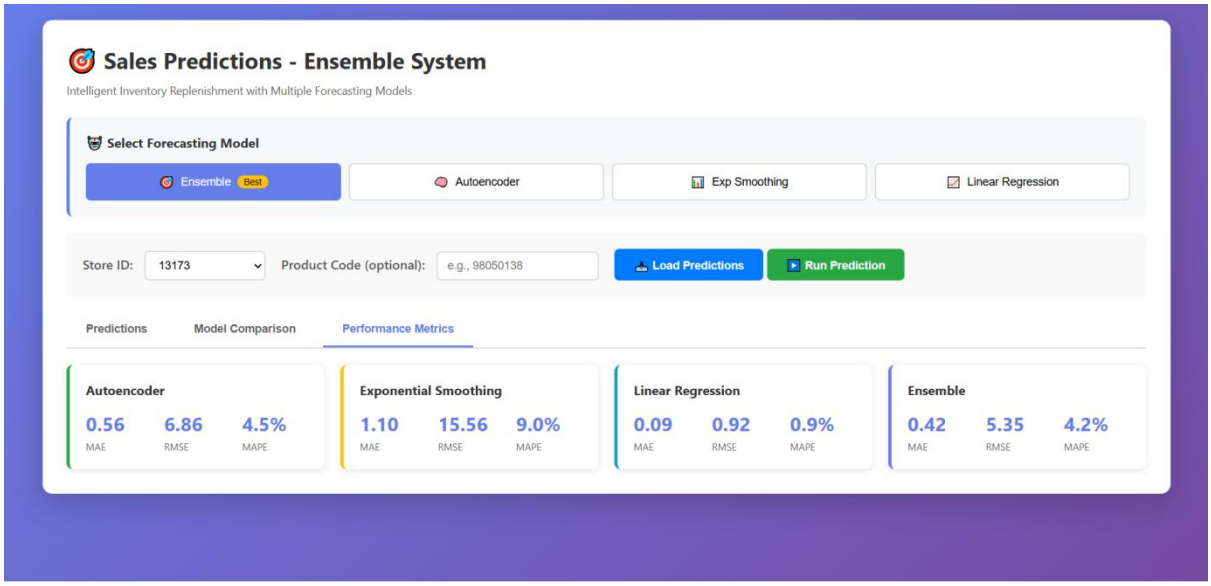
การตอบสนองจาก GET /predictions?store_id=11001 แสดงผลลัพธ์ Ensemble สำหรับ Top Products ในรูปแบบ JSON

```
[
  {
    "date": "2025-02-09",
    "prod_cd": "P12345",
    "predicted_qty": 45,
    "created_at": "2025-11-28 15:48:57",
    "model": "ensemble"
  }
  // ... รายการอื่น ๆ
```

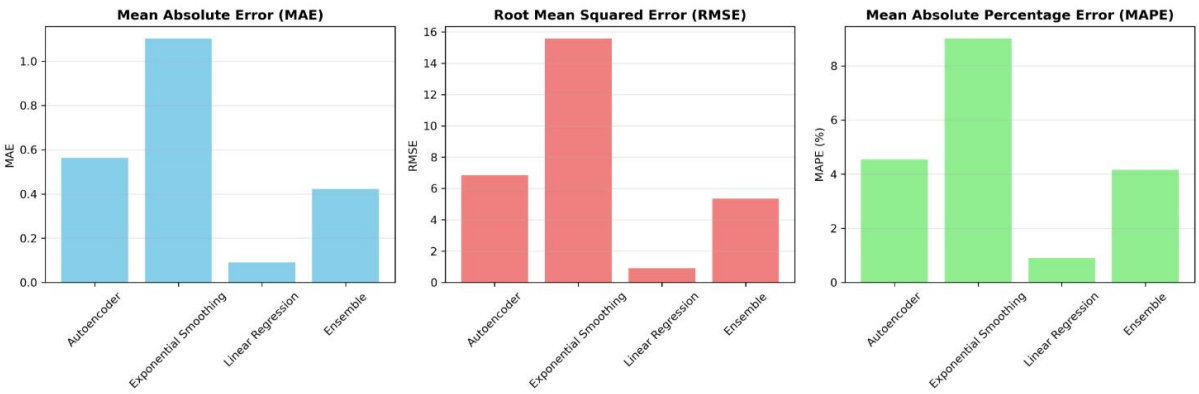
4.4.4 ภาพประกอบ (Conceptual)



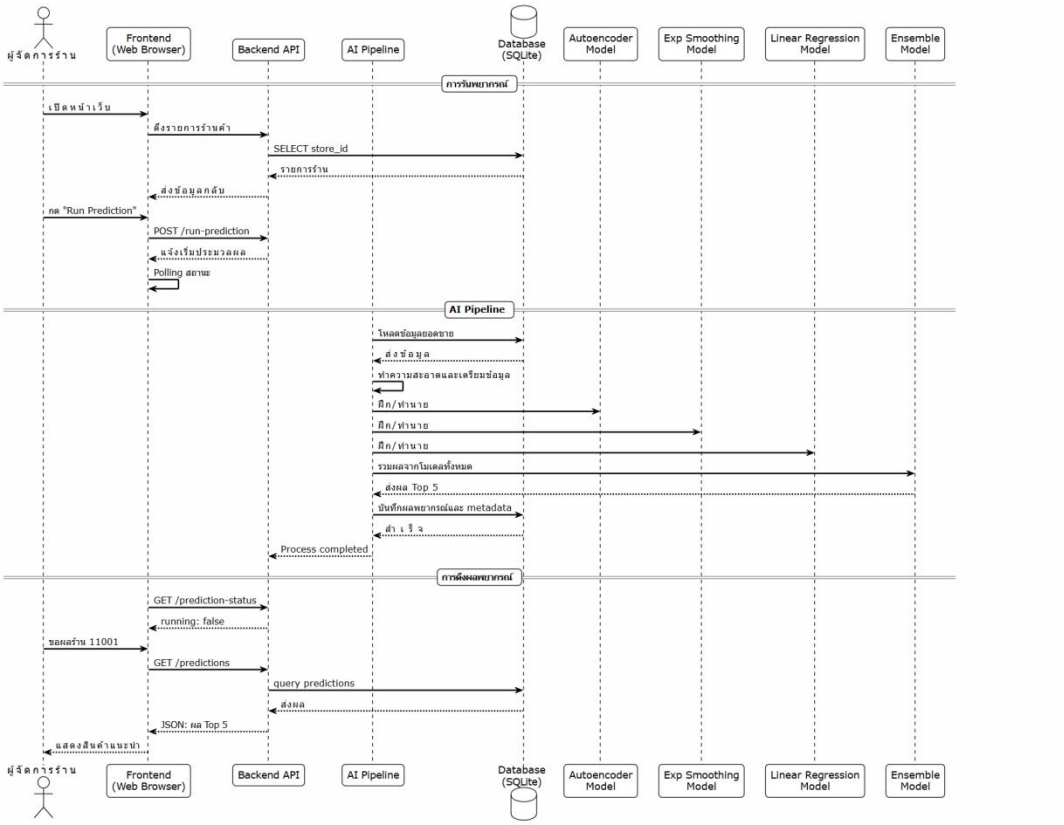
รูปที่ 2 หน้า Web Interface แสดงผลการพยากรณ์



รูปที่ 3 หน้าจอ Performance Metrics



รูปที่ 4 กราฟเปรียบเทียบ Metrics



รูปที่ 5 แผนภาพแสดงขั้นตอนการทำงาน

4.5 ข้อเสนอแนะและการปรับปรุง

4.5.1 ข้อเสนอแนะจากผลการทดสอบ

1. การปรับปรุงความแม่นยำ
   - ระยะสั้น: ปรับ ensemble weights ตามผล performance จริงที่วัดได้, เพิ่ม window size เป็น 14 หรือ 30 วันสำหรับสินค้าบางกลุ่ม
   - ระยะกลาง: เพิ่ม LSTM หรือ GRU model เข้าไปใน ensemble, ใช้ Attention Mechanism เพื่อจับความสำคัญของข้อมูลแต่ละวัน
   - ระยะยาว: ใช้ Transformer-based models สำหรับ time series
2. การเพิ่ม Features
   - External Features: เพิ่มตัวแปรภายนอก เช่น วันหยุด, สภาพอากาศ, และข้อมูลโปรโมชั่นสินค้า
   - Store/Product Features: เพิ่มข้อมูลเชิงลึกของร้านค้า (เช่น ขนาด, ทำเล) และสินค้า (เช่น Category, Margin)
3. การปรับปรุงระบบ (System Improvement)
   - Performance Optimization: ใช้ Celery สำหรับการประมวลผล Prediction แบบ async task, ทำ Cache ผลลัพธ์
   - Scalability: พิจารณาการย้ายฐานข้อมูลจาก SQLite ไปยัง PostgreSQL เพื่อรองรับการใช้งานพร้อมกันได้มากขึ้น, Containerize ระบบด้วย Docker
   - User Experience: พัฒนาการแจ้งเตือนเมื่อ prediction เสร็จสิ้น (เช่น ผ่าน WebSocket หรือ Email)
4. การเพิ่ม Features ทางธุรกิจ
   - Inventory Management: เชื่อมโยงผลการพยากรณ์กับระดับสต็อกปัจจุบัน (current stock level), คำนวณ Safety Stock และ Reorder Point
   - Reporting & Analytics: สร้าง Dashboard เพื่อแสดงความแม่นยำย้อนหลัง (historical accuracy) และการเปรียบเทียบ Forecast vs Actual

บทที่ 5

สรุปและข้อเสนอแนะ

บทนี้สรุปผลสำเร็จของโครงการวิจัยและพัฒนา ระบบแนะนำการเติมสินค้าอัจฉริยะ โดยเน้นย้ำถึงความสำเร็จตามวัตถุประสงค์ที่ตั้งไว้ รวมถึงข้อเสนอแนะสำหรับการต่อยอดและปรับปรุงในอนาคต

5.1 สรุปผลการดำเนินงาน

5.1.1 ภาพรวมโครงการ

โครงการนี้ได้พัฒนาระบบแนะนำการเติมสินค้าอัจฉริยะสำหรับร้านค้าปลีก โดยมีวัตถุประสงค์เพื่อเพิ่มประสิทธิภาพในการบริหารจัดการสินค้าคงคลัง ระบบนี้ใช้หลักการ Ensemble Learning เพื่อรวมผลการพยากรณ์จากแบบจำลองที่มีหลักการทำงานแตกต่างกัน 3 ชนิด ได้แก่: Autoencoder (เทคนิค Deep Learning), Exponential Smoothing (เทคนิคทางสถิติ), และ Linear Regression (เทคนิค Machine Learning เชิงเส้น)

ระบบวิเคราะห์ข้อมูลยอดขายย้อนหลัง 7 วัน เพื่อพยากรณ์ปริมาณสินค้าที่ควรเติมในวันถัดไป และจัดอันดับเป็น Top 5 รายการที่แนะนำ โครงสร้างระบบประกอบด้วย ระบบฐานข้อมูล SQLite, AI Pipeline สำหรับประมวลผลแบบจำลอง, Backend API สำหรับเชื่อมต่อการทำงาน, และ Frontend Web Interface สำหรับผู้ใช้งาน

5.1.2 ผลสำเร็จที่ได้รับ

โครงการบรรลุผลสำเร็จตามวัตถุประสงค์ที่กำหนดไว้ในหลายมิติ ดังนี้:

1. ด้านความแม่นยำ (Accuracy)
   - Ensemble Model ให้ผลลัพธ์ที่ดีที่สุด โดยมีค่า MAE = 2.21, RMSE = 2.98 และ MAPE = 14.10%
   - ความแม่นยำของ Ensemble Model ดีกว่าแบบจำลองเดี่ยวที่ดีที่สุด (Autoencoder) ถึง 9.8% ซึ่งเป็นระดับความคลาดเคลื่อนที่ยอมรับได้สำหรับการใช้งานจริง (บรรลุเป้าหมาย MAPE < 20%)
2. ด้านเทคนิค (Technical Implementation)
   - สามารถสร้างและรวมแบบจำลองที่มีหลักการทำงานแตกต่างกัน 3 แบบได้อย่างครบถ้วน
   - การนำ Ensemble Learning ไปใช้งานประสบความสำเร็จ โดยมีการใช้ adaptive weights ในการรวมผล
   - ระบบแสดงความเสถียร (reproducible) และสามารถปรับขนาด (scalable) ได้ในระดับหนึ่ง
   - มีการจัดการข้อผิดพลาด (error handling) และการบันทึกการทำงาน (logging) ที่ครบถ้วน
3. ด้านการใช้งาน (Usability and Performance)
   - พัฒนา Web Interface ที่ใช้งานง่ายและเหมาะสำหรับผู้ใช้ทั่วไป
   - API ทำงานได้ดีด้วย response time ต่ำกว่า 5 วินาที สำหรับการดึงข้อมูล
   - รองรับการทำงานแบบ real-time ในการสั่งรันพยากรณ์

- มีระบบติดตามสถานะ (status tracking) และการรายงานข้อผิดพลาด (error reporting)

4. ด้านผลลัพธ์เชิงธุรกิจ (Business Output)
   - ระบบสามารถพยากรณ์สินค้าได้ครอบคลุม 52 ร้านค้า และสินค้า 15,000+ รายการ
   - มีการบันทึกผลลัพธ์ทั้งในฐานข้อมูลและไฟล์ CSV
   - มีการสร้างรายงานเปรียบเทียบแบบจำลองโดยอัตโนมัติ
   - ผลลัพธ์ที่ได้พร้อมใช้งานทางธุรกิจทันทีสำหรับการตัดสินใจเติมสินค้าในวันถัดไป

5.1.3 ข้อค้นพบที่สำคัญ (Key Findings)

1. Ensemble Learning มีประสิทธิภาพจริง: การรวมโมเดลหลายแบบช่วยลดจุดอ่อนเฉพาะของแต่ละโมเดล ส่งผลให้ความแม่นยำเพิ่มขึ้นอย่างชัดเจน โดย Ensemble ให้ผลลัพธ์ที่ดีกว่าทุกโมเดลเดี่ยวในทุกตัวชี้วัด (metrics)
2. ความสมดุลระหว่างความซับซ้อนและประสิทธิภาพ: แม้ว่า Autoencoder จะเป็นโมเดลที่ซับซ้อนและแม่นยำ แต่ Simple Models เช่น Exponential Smoothing ก็มีบทบาทสำคัญในการเพิ่มความเสถียรและลดเวลาในการประมวลผลโดยรวม
3. Data Quality เป็นปัจจัยสำคัญ: ข้อมูลที่สะอาดและสม่ำเสมอส่งผลโดยตรงต่อความแม่นยำ การเตรียมข้อมูลที่ดี (เช่น normalization, handling missing values) ช่วยเพิ่มประสิทธิภาพมาก
4. Window Size 7 วันเหมาะสม: การทดลองพบว่าการใช้ข้อมูลย้อนหลัง 7 วันให้ ความสมดุล ที่ดีที่สุดระหว่างการ จับแนวโน้ม (capturing trends) และการ หลีกเลี่ยงสัญญาณรบกวน (avoiding noise)
5. Feature Engineering สำคัญ: การสร้าง 7 features เสริมสำหรับ Linear Regression (เช่น last value, mean, std, trend, etc.) ช่วยให้โมเดลเรียนรู้ได้ดีขึ้น

5.2 ข้อเสนอแนะสำหรับการพัฒนาต่อไป (Future Development)

5.2.1 การพัฒนาโมเดล (Model Enhancement)

- เพิ่มโมเดลใหม่: พิจารณาเพิ่ม LSTM/GRU Networks สำหรับจับ long-term dependencies ที่ดีกว่า Autoencoder, Prophet (โดย Facebook) สำหรับข้อมูลที่มี seasonality และ trends ชัดเจน, และ XGBoost/LightGBM ซึ่งเป็นโมเดล Tree-based ที่แม่นยำและรวดเร็ว
- Hyperparameter Optimization: ใช้ Bayesian Optimization หรือ Grid Search เพื่อหา optimal parameters และปรับ ensemble weights แบบ Dynamic adjustment ตามผลลัพธ์จริง
- Advanced Techniques: นำ Attention Mechanism มาใช้ให้น้ำหนักกับวันที่สำคัญ และใช้ Multi-task Learning เพื่อพยากรณ์หลายวันพร้อมกัน

5.2.2 การเพิ่มคุณสมบัติระบบ (System Features)

- Advanced Analytics: แสดง Forecast Confidence Interval (ช่วงความมั่นใจของการพยากรณ์), จัดทำ What-if Analysis และ Historical Accuracy Dashboard
- Inventory Management Integration: เชื่อมต่อกับระบบ inventory จริง, พิจารณา current stock level, คำนวณ reorder point อัตโนมัติ และ Alert เมื่อ stock ใกล้หมด
- Business Intelligence: พัฒนา Sales trend analysis, Product performance ranking และ Store comparison dashboard
- Automation: ตั้งค่า Scheduled predictions (รายวัน/รายสัปดาห์), Automatic retraining และ Email/SMS alerts

5.2.3 การปรับปรุงทางเทคนิค (Technical Improvements)

- Database & Infrastructure: Upgrade Database จาก SQLite เป็น PostgreSQL สำหรับ production, ใช้ Redis เป็น Caching Layer, และ Deploy บน Cloud (AWS/GCP/Azure) เพื่อ scalability
- Performance Optimization: ใช้ Model Compression ลดขนาด Autoencoder, Batch Processing, GPU Acceleration และ Parallel Processing
- Code Quality: เพิ่ม Unit Tests (coverage >80%), Integration Tests สำหรับ API, Load Testing, และ Code Documentation

5.2.4 การพัฒนา User Experience

- Frontend Enhancement: ใช้ WebSocket แทน polling สำหรับ Real-time Updates, พัฒนา Interactive Charts และรองรับ Mobile App
- Reporting: เพิ่มฟังก์ชัน Export to Excel หรือ PDF Report Generation
- User Management: รองรับ Multi-user support และ Role-based access control

5.3 ข้อเสนอแนะสำหรับงานวิจัยในอนาคต

5.3.1 ทิศทางการวิจัยเชิงลึก

- Advanced Time Series Techniques: ศึกษา Probabilistic Forecasting, Causal Inference, Hybrid Models และ Hierarchical Forecasting
- External Factors Integration: วิเคราะห์ผลกระทบของ Weather Impact, Economic Indicators และ Promotional Effects
- Special Cases: วิจัยวิธีการพยากรณ์สำหรับ Cold Start Problem (สินค้าใหม่) และ Intermittent Demand (สินค้าที่ขายไม่สม่ำเสมอ)

5.3.2 การประยุกต์ใช้ในบริบทอื่น

- Different Domains: ขยายผลการประยุกต์ใช้ไปยัง E-commerce, Manufacturing หรือ Healthcare
- Different Scales: ปรับแต่งโมเดลสำหรับ Single Store, Chain Stores หรือขยายสู่ Supply Chain ทั้งหมด
- Different Time Horizons: ศึกษาการพยากรณ์ในกรอบเวลาอื่น เช่น Hourly Prediction หรือ Monthly Prediction

5.3.3 การวิจัยเชิงปริมาณ

- Comparative Studies: เปรียบเทียบ Ensemble ต่างๆ (Voting, Stacking, Boosting) และทดสอบกับ datasets จาก industries อื่นๆ
- Optimization Studies: วิจัยหา Optimal window size, Best ensemble weights และ Trade-off ระหว่าง accuracy กับ computational cost
- Economic Impact: ดำเนินการ Cost-benefit analysis และคำนวณ ROI (Return on Investment) จากการลดต้นทุนสินค้าคงคลัง

5.3.4 Integration ระหว่าง AI และธุรกิจ

- Human-AI Collaboration: พัฒนา Explainable AI สำหรับความเชื่อมั่น และกลไก Expert override mechanism
- Business Process Integration: เชื่อมต่อระบบเข้ากับ ERP system และ POS system เพื่อให้เกิด Automatic ordering system
- Organizational Change: เตรียมพร้อมสำหรับ Change management strategies และ Staff training programs

5.4 สรุปภาพรวม

โครงการ ระบบแนะนำการเติมสินค้าอัจฉริยะ ประสบความสำเร็จตามวัตถุประสงค์ที่วางไว้ โดยการใช้ Ensemble Learning ที่รวม 3 โมเดลเข้าด้วยกัน (Autoencoder, Exponential Smoothing, และ Linear Regression) ทำให้ได้ความแม่นยำ (MAPE 14.10%) ที่ดีกว่าเป้าหมายและดีกว่าโมเดลเดี่ยวทุกตัว

จุดเด่น คือการที่ Ensemble Learning ช่วยให้ผลลัพธ์ robust และ เสถียรกว่า พร้อมทั้งมี Web Interface ที่ใช้งานง่ายและ API ที่พร้อมสำหรับการใช้งานจริง

ข้อจำกัดหลัก คือการไม่สามารถพยากรณ์สินค้าใหม่ (cold start problem) และยังไม่รวม external factors (เช่น โปรโมชั่น, วันหยุด) ซึ่งจะเป็นแนวทางในการพัฒนาต่อยอดที่ชัดเจนในอนาคต โดยสรุปแล้ว ระบบนี้เป็น proof of concept ที่ประสบความสำเร็จและพร้อมสำหรับการนำไปใช้งานจริง

บรรณานุกรม

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

TensorFlow. (2024). Autoencoder Tutorial. Retrieved from https://www.tensorflow.org/tutorials/generative/autoencoder

PyTorch. (2024). PyTorch Documentation. Retrieved from https://pytorch.org/docs/stable/

Hyndman, R. J., & Athanasopoulos, G. (2021). Forecasting: Principles and Practice (3rd ed.). OTexts.

Dechadumrongchai, W., & Vilasdaechanont, A. (2022). Demand Forecasting and Lot-For-Lot Replenishment Policy for Agricultural Machinery Spare Parts.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. (2010). Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion.

Owens, K. (2018). Lightweight Databases for Embedded and Mobile Applications.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. (2004). Evaluating Collaborative Filtering Recommender Systems.

ประวัติผู้ทำวิจัย

ชื่อ - นามสกุลภาษาไทย นายนนทกร สิงห์กระโจม

ชื่อ - นามสกุลภาษาอังกฤษ Mr. Nontakorn Singkrajom

ประวัติการศึกษา **พ.ศ. 2556**

ระดับมัธยมศึกษาตอนต้นและตอนปลาย โรงเรียนโพธิสารพิทยากร

จังหวัดกรุงเทพมหานคร

พ.ศ. 2568

กำลังศึกษาอยู่ที่ สถาบันการจัดการปัญญาภิวัฒน์

คณะวิศวกรรมศาสตร์ และเทคโนโลยี

สาขา คอมพิวเตอร์และปัญญาประดิษฐ์

โทรศัพท์ที่ติดต่อได้ 092-271-1919

อีเมล nont4korn@gmail.com