

Agentic AI

สำหรับนักการเงิน

พื้นฐานทางเทคนิคและการประยุกต์



ศาสตราจารย์ ดร. อาณัติ สัมคเดช

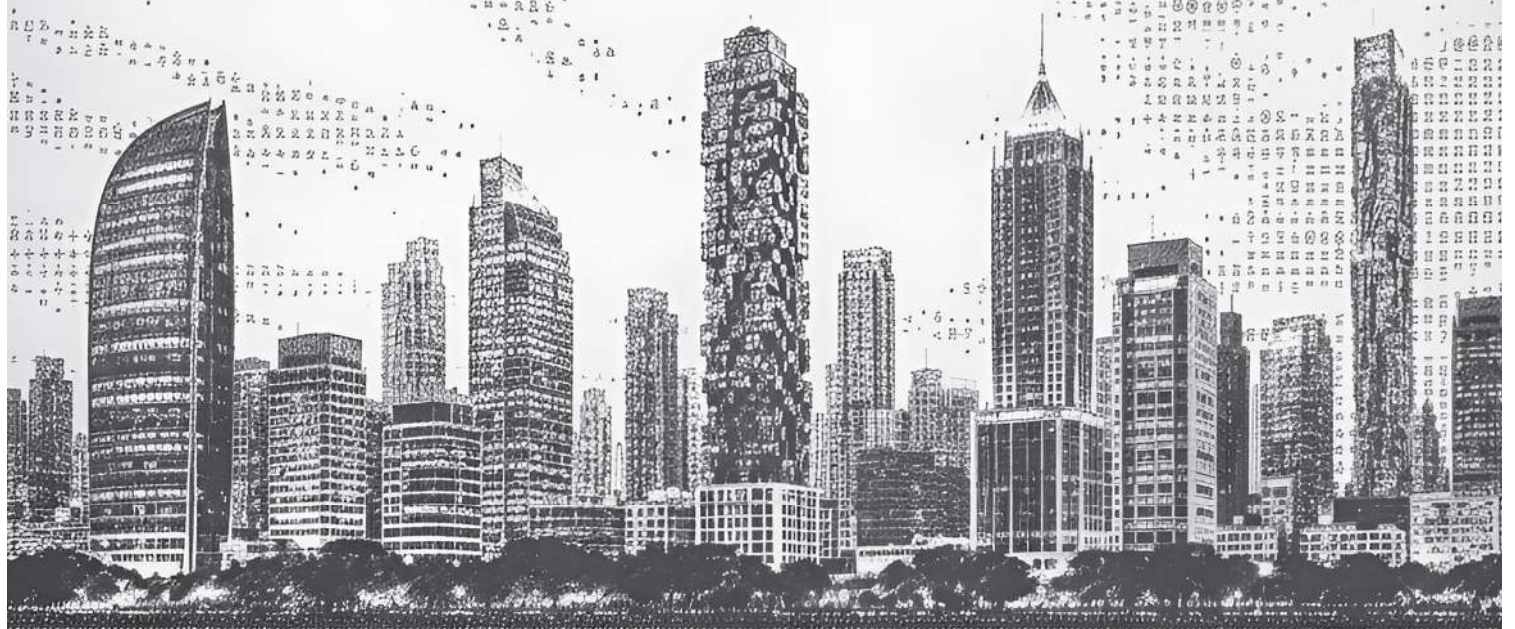
คณะพาณิชยศาสตร์และการบัญชี มหาวิทยาลัยธรรมศาสตร์

Agentic AI

สำหรับนักการเงิน

ศาสตราจารย์ ดร. อาณัติ ลีมคเดช

คณะพาณิชยศาสตร์และการบัญชี มหาวิทยาลัยธรรมศาสตร์



Agentic AI สำหรับนักการเงิน

ศาสตราจารย์ ดร.อาณัติ ลี้มัคเดช

ราคา 790 บาท

ข้อมูลทางบรรณานุกรม

ศาสตราจารย์ ดร.อาณัติ ลี้มัคเดช

Agentic AI สำหรับนักการเงิน

จำนวน 624 หน้า

ราคา 790 บาท

ISBN 978-616-92877-1-1

พิมพ์ที่ ห้างหุ้นส่วนจำกัด เอส.ออฟเซ็ทกราฟฟิคดีไซน์

สำนักพิมพ์ พาร์คพรอพ - นนทบุรี : 2569

พิมพ์ที่

ห้างหุ้นส่วนจำกัด เอส.ออฟเซ็ทกราฟฟิคดีไซน์

63 ประชาอุทิศ 75 แขวงทุ่งครุ กรุงเทพฯ 10140

จัดทำและจัดจำหน่ายโดย

บริษัท พาร์คพรอพ จำกัด

9/29 หมู่ที่ 9 อาคารเดอะพาร์ค

ตำบลบางม่วง อำเภอบางใหญ่ นนทบุรี 11140

โทร 099-0027008, 081-7107490

Email: thepark929@gmail.com

ผู้เขียน

ที่ปรึกษา

ศ.ดร.อาณัติ ลี้มัคเดช

ชัยโย เตโชนิมิต (ด้านโปรแกรมมิ่ง)

อรรธรณ ลิมกังวาพมงคล, วณิดา ประทีปเสน (ด้านธุรกิจ)

ศิลปกรรม

ภาพปก

เอกอนันต์ ศิริเลิศฤทัย

สร้างโดยผู้เขียนด้วยความช่วยเหลือของ ChatGPT (OpenAI)

และปรับแต่งโดยผู้เขียน

สงวนลิขสิทธิ์ในประเทศไทยตาม พ.ร.บ. ลิขสิทธิ์ โดยบริษัท พาร์คพรอพ จำกัด
ห้ามการลอกเลียนไม่ว่าส่วนหนึ่งส่วนใดของหนังสือเล่มนี้ นอกจากจะได้รับอนุญาต

คำนำ

การใช้ AI ในงานการเงินขององค์กร ไม่ใช่ทางเลือก แต่เป็นเครื่องมือสำคัญที่เลี่ยงไม่ได้ เนื่องจาก Agentic AI เป็นการรวมพลังความเข้าใจในโมเดลภาษา การคำนวณ และกฎเกณฑ์การเงิน เช่น การคำนวณอัตราส่วนทางการเงินต่างๆ ด้วยการเขียน Prompt เพียงไม่กี่บรรทัด ปัญหาสำคัญของผู้บริหารคือจะเริ่มจุดไหนก่อน และจะเอาชนะแรงเฉื่อยในองค์กรได้อย่างไร ตำราเล่มนี้จะช่วยชี้ทางออก

ก่อนที่จะนำ AI มาใช้ องค์กรจำเป็นต้องปูพื้นฐานให้คนในทีมเห็นก่อนว่า AI ทำอะไรได้บ้าง จากเดิมที่คนในทีมฝ่าย Quant อาจเห็นว่า AI ใช้ในการคำนวณด้วยวิธีเครือข่ายประสาทเทียมอย่างเดียว ใช้ในการพยากรณ์ค่าที่เป็นตัวเลขเท่านั้น มาสู่การสร้างโมเดลภาษาที่เรียกว่า LLM (Large Language Model) จนเป็นจุดกำเนิดของ AI แบบ ChatGPT ซึ่งเรียกความสามารถ AI ในการสร้างเนื้อหาใหม่ ไม่ว่าจะเป็นคำอธิบาย รูปภาพ หรือเสียงเพลง นี่ว่า Generative AI ในยุคแรก โมเดลภาษาเหล่านี้คือแบบจำลองความน่าจะเป็นล้วน ๆ ทุกคำตอบที่ออกมาเกิดจากการพยากรณ์ความน่าจะเป็นของคำที่จะอยู่ถัดไปที่ AI เรียนรู้จากคลังข้อมูลต่างๆ ที่รวบรวมบนเครือข่ายอินเทอร์เน็ต ดังนั้นข้อมูลใหม่ๆ ที่ไม่เคยเห็นมาก่อน Generative AI ก็จะมี “ตอบผิดอย่างมั่นใจ” เกิดเป็นปัญหาข้อมูลหลอน (Hallucination) ทำให้หลายคนไม่เชื่อใจที่จะนำ AI มาใช้ในองค์กร

เพื่อแก้ปัญหาดังกล่าว จึงมีการฝึก AI ให้ตอบเฉพาะกรอบของข้อมูลที่กำหนดให้ อันไหนไม่รู้ให้ตอบว่าไม่รู้ ห้ามเดา เรียกว่าเทคนิค RAG (Retrieval Augmented Generation) จนกระทั่งมาถึงยุคปัจจุบันที่กำหนดให้ AI สามารถวางแผนหาคำตอบ สร้างทีม AI ค้นคำตอบในรูปแบบต่างๆ แล้วมาประสานงานกันเพื่อหาคำตอบที่ดีที่สุด เรียกว่า Agentic AI

ด้วยความสามารถนี้ ทำให้ผมเห็นว่า Agentic AI เสมือนทีมงานในองค์กรที่ทำงานฉับไว และหากองค์กรไหนตามไม่ทัน ก็อาจถูกคู่แข่งทิ้งห่างอย่างไม่เห็นฝุ่น การเรียนรู้ Agentic AI ทั้งในเชิงเทคนิคและการประยุกต์กับงานการเงิน จึงเป็นสิ่งจำเป็นสำหรับผู้บริหารสายการเงิน

ตำราเล่มนี้ถูกพัฒนาขึ้นเพื่อใช้สอนในคณะพาณิชยศาสตร์และการบัญชี มหาวิทยาลัยธรรมศาสตร์ เนื้อหาทั้งหมดถูกออกแบบเพื่อให้ผู้เรียนสามารถใช้เครื่องมือ AI ที่สมัครขอใช้บริการฟรีได้ ยกเว้นบทที่ 10 บทเดียวที่จำเป็นต้องใช้ Claude Cowork ที่มีค่าใช้จ่าย

ดังนั้นผู้เรียนสามารถฝึกปฏิบัติไปกับโครงการทางการเงินที่ตำราเล่มนี้ใช้เป็นตัวอย่างได้แทบทั้งหมด โดยไม่มีค่าใช้จ่ายเพิ่ม

โครงสร้างของตำราแบ่งเป็นสองส่วนใหญ่ๆ คือพื้นฐานเทคนิค ตั้งแต่ บทที่ 1 ถึง 7 พูดถึงการพัฒนาปัญญาประดิษฐ์ พร้อมกิจกรรมให้ผู้เรียนลงมือพัฒนาการใช้งานไปด้วย ส่วนที่สองเป็นการลงมือพัฒนาโครงการทางการเงิน โดยบทที่ 8 พูดถึงลักษณะการจัดทำโครงการให้เป็นมาตรฐานเดียวกัน ตามด้วย บทที่ 9 ถึง 14 เป็นโครงการที่ผมคำนึงถึงการนำไปใช้งานได้เลย

ตำราเล่มนี้เน้นการพัฒนาทักษะผู้เรียนเป็นลำดับขั้น จึงขอให้ผู้เรียนค่อยๆ เรียนไปที่ละบท

ผมเพิ่มบทที่ 15 ขึ้นมาก่อนปิดต้นฉบับ หลังจากพูดคุยกับคนในธุรกิจ พบว่าอุปสรรคสำคัญของการใช้ AI ในองค์กรคือความกังวลเรื่องการรั่วไหลของข้อมูล จึงแนะนำการติดตั้ง Local LLM ซึ่งปัจจุบันมีหลายค่ายและมีพัฒนาการอย่างรวดเร็ว ระหว่างที่เขียนนี้ก็มีคอลิบรี colibri ที่เป็นโมเดลภาษาระดับ 744 พันล้านพารามิเตอร์แต่สามารถรันบนเครื่อง Notebook รุ่นใหม่ที่มีหน่วยความจำสูงได้

อาจารย์ที่นำตำราเล่มนี้ในห้องเรียนอาจข้ามบทนี้ไปได้ เพราะจะมีพัฒนาการใหม่ ออกมาตลอดเวลา ผมเพียงต้องการย้ำว่าปัญหาการรั่วไหลของข้อมูลไม่ควรถูกใช้เป็นข้ออ้างในการนำ AI มาใช้ในองค์กรอีกต่อไป

ตำราเล่มนี้ถูกสร้างโดยมีเครื่องมือ AI Claude เป็นผู้ช่วยสำคัญ โดยโค้ดแทบทั้งหมดในเล่มถูกเขียนด้วย AI แต่ผ่านการสอบทานจาก **คุณชัยโย เตโชนิมิต** CTO ของบริษัท เวโลพาร์ค ซึ่งผมต้องขอขอบคุณเป็นอย่างมาก

พร้อมกันนั้นก็สร้างความมั่นใจให้ผมเห็นว่าผู้บริหารการเงิน ที่อาจไม่คุ้นชินกับการเขียนโปรแกรมสามารถใช้ AI เป็นผู้ช่วยได้ ด้วยการสั่งงานโดยภาษามนุษย์ เพราะเนื้อหาส่วนกิจกรรมในตำราเล่มนี้เป็นบทพิสูจน์อย่างดี Claude ยังเป็นเครื่องมือสร้างภาพ Diagram ทั้งหมดที่อยู่ในตำราเล่มนี้ แต่ความสามารถในการจัดรูปแบบของ Claude หรือโปรแกรม AI อื่นๆ ยังห่างไกลจากการจัดรูปแบบมืออาชีพ เรายังต้องใช้ทักษะการจัดหน้าและรสนิยมของผู้จัดรูปแบบมือฉมัง **คุณเอกอนันต์ ศิริเลิศฤทัย** ที่ทำให้ตำราเล่มนี้น่าอ่าน

ผมขอขอบคุณ **คุณอรรธรณ ลิ้มกังวาฬมงคล** และ **คุณวนิดา ประทีปเสน** ที่ช่วยประสานงานด้านธุรกิจ การจัดพิมพ์และจัดจำหน่ายตำราเล่มนี้ด้วย

สุดท้าย ผมเชื่อว่า Agentic AI ยังไม่สามารถแย่งงานนักการเงินได้ เรายังต้องการวิจรณ์ญาณทางการเงินของผู้บริหาร ความสามารถในการเจรจาต่อรอง แต่ AI จะเปลี่ยนวิธีการทำงานของผู้บริหารในสายการเงินแน่นอน โดยผู้บริหารจะต้องรู้วิธีการสั่งงาน AI ให้ทำงานน่าเบื่อ ซ้ำซาก แทน เช่นการวนคำนวณค่า NPV ภายใต้ความเป็นไปได้ร้อยแปดพันเก้า การร่างรายงานตามฟอร์แมตมาตรฐาน การตรวจสอบกฎระเบียบ เพื่อเอาเวลาไปใช้ในงานเชิงกลยุทธ์ การตัดสินใจ มากขึ้น และตำราเล่มนี้จะเป็นเครื่องมือสำคัญของท่าน

ศาสตราจารย์ ดร. อาณัติ ลิ้มคเดช

คณะพาณิชยศาสตร์และการบัญชี

มหาวิทยาลัยธรรมศาสตร์



ไฟล์ประกอบทุกบทเรียนในหนังสือเล่มนี้
ทั้งโปรแกรมและข้อมูลสามารถดาวน์โหลดได้ที่
<http://tinyurl.com/AgenticFinance>

คำนิยม

AI มีพัฒนาการอย่างรวดเร็ว และมีความจำเป็นที่สถาบันการศึกษาจำเป็นต้องปรับตัว เพื่อพัฒนาบัณฑิตที่ตอบโจทย์ภาคธุรกิจที่มีการแข่งขันสูงในปัจจุบัน

คณะพาณิชยศาสตร์และการบัญชี มหาวิทยาลัยธรรมศาสตร์เองได้ตั้งเป้าการพัฒนา AI ในทุกระดับของคณะ ไม่เพียงแต่นักศึกษา อย่างไรก็ตาม องค์ความรู้ด้านนี้ยังมีการกระจาย จัดกระจาย ทำให้การเรียนรู้เกิดขึ้นที่ละส่วน ที่ผู้เรียนต้องนำไปปะติดปะต่อเอง

ตำรา **Agentic AI สำหรับนักการเงิน** ที่ ศ.ดร.อาณัติ ลิ้มคเดช พัฒนาขึ้น แก้ Pain Point นี้ เพราะเป็นการบูรณาการความรู้ทางด้าน AI โดยชี้ให้เห็นพัฒนาการเป็นลำดับ ตั้งแต่การเขียน Prompt ใน Generative AI ไปจนกระทั่งการสร้าง Workflow ของทีม Multi-Agent ใน Agentic AI พร้อมกันนี้ยังได้พัฒนาโครงงานขึ้นมาเป็นภาคปฏิบัติ เพื่อให้ผู้เรียนได้ทดลองพัฒนาโครงการจริง โดยเน้นการต่อยอดความรู้ที่ละชั้น

ตำราเล่มนี้จึงเป็นประโยชน์ทั้งสำหรับนักศึกษา และผู้บริหารการเงินที่ต้องการ Upskill ให้ทันกับเทคโนโลยี AI ที่กำลังเปลี่ยนแปลงอย่างรวดเร็ว

รศ.ดร.สุรัตน์ ทิรมาภิบาล

คณบดี คณะพาณิชยศาสตร์และการบัญชี
มหาวิทยาลัยธรรมศาสตร์

คำนิยม

ตลอดหลายทศวรรษที่ผ่านมา เทคโนโลยีได้เข้ามาเปลี่ยนแปลงอุตสาหกรรมการเงินอย่างต่อเนื่อง ตั้งแต่ระบบ Core Banking การทำธุรกรรมผ่าน Mobile Banking ไปจนถึง Cloud และ Data Analytics แต่การมาถึงของ **Generative AI** และต่อยอดสู่ **Agentic AI** อาจเป็นอีกก้าวสำคัญที่ไม่ได้เพียงเปลี่ยน “เครื่องมือ” ในการทำงาน หากกำลังเปลี่ยน “วิธีการทำงาน” ของภาคการเงินอย่างสิ้นเชิง

หนังสือ **Agentic AI สำหรับนักการเงิน** เล่มนี้ มาถูกเวลา เพราะไม่ได้อธิบายเพียงแนวคิดของ AI แต่ค่อยๆ พาผู้อ่านตั้งแต่พื้นฐานของ Generative AI ไปจนถึงการออกแบบ AI Agent, Multi-Agent, Workflow Automation และโครงการที่สามารถนำไปประยุกต์ใช้กับงานการเงินจริงได้อย่างเป็นระบบ จึงเหมาะทั้งสำหรับผู้เริ่มต้นศึกษา และผู้ที่ต้องการนำ AI ไปสร้างคุณค่าให้กับองค์กร

ผมเห็นด้วยกับ ศ.ดร.อาณัติ ที่มองว่า AI ไม่ได้มาแทนที่มนุษย์ แต่กำลังกลายเป็น “เพื่อนร่วมงานดิจิทัล” ที่ช่วยรับผิดชอบงานซ้ำๆ และงานที่ต้องประมวลผลข้อมูลจำนวนมาก เพื่อให้มนุษย์มีเวลามากขึ้นกับการคิดเชิงกลยุทธ์ การตัดสินใจ และการสร้างสรรค์สิ่งใหม่ๆ

ผมนึกถึงสำนวนจีนที่ว่า 三个人干五个人的活 · 拿四个人的工资 หรือ “ทีมขนาด 3 คน แต่สร้างผลงานได้เทียบเท่า 5 คน ด้วยต้นทุนใกล้เคียง 4 คน” แม้จะเป็นเพียงคำเปรียบเทียบ แต่ก็สะท้อนเป้าหมายของหลายองค์กรในยุคที่ต้องยกระดับผลิตภาพและสร้างคุณค่าจากทรัพยากรที่มีอยู่

ท้ายที่สุดแล้ว คุณค่าของ Agentic AI ไม่ได้อยู่ที่ความสามารถของเทคโนโลยี แต่อยู่ที่ว่าเราจะนำเทคโนโลยีนั้นไปสร้างคุณค่าให้กับผู้คน ลูกค้า และองค์กรได้มากเพียงใด ผมหวังว่าหนังสือเล่มนี้จะช่วยให้ผู้อ่านไม่เพียงเข้าใจ Agentic AI แต่ยังสามารถนำความรู้นั้นไปออกแบบวิธีการทำงานรูปแบบใหม่ เพื่อสร้างคุณค่าและยกระดับศักยภาพขององค์กรได้อย่างแท้จริง

ผศ.ดร.กรินทร์ บุญเลิศวิชย์

รองผู้จัดการใหญ่

ธนาคารกสิกรไทย

สารบัญ

Agentic AI สำหรับนักการเงิน

บทที่ 1 Generative AI และ Foundation Model สำหรับงานการเงิน	2
1.1 จากโครงข่ายประสาทเทียม สู่ Deep Learning และ Generative AI	3
1.2 word2vec: ทำให้คอมพิวเตอร์เข้าใจความหมายของคำ	6
1.3 กรณีศึกษา: การพยากรณ์ราคาหุ้นไทยด้วย word2vec และ Deep Learning	9
1.3.1 คำถามวิจัยและสมมติฐานตลาดมีประสิทธิภาพ (Efficient Market Hypothesis: EMH)	9
1.3.2 ขั้นตอนและสถาปัตยกรรมของแบบจำลอง	10
1.3.3 ผลการศึกษาและนัยทางการเงิน	10
1.4 จาก word2vec สู่ Large Language Models (LLM)	12
1.4.1 กลไก Attention: หัวใจของ Transformer	13
1.4.2 การฝึกล่วงหน้า การปรับแต่ง และการขยายขนาด	14
1.5 หลักการทำงานของ LLM ในระดับแนวคิด	16
1.6 การประยุกต์ Generative AI ในงานการเงิน	18
1.6.1 การย่อและวิเคราะห์เอกสารการเงิน	18
1.6.2 การวิเคราะห์ความรู้สึกและสัญญาณจากข่าว	19
1.6.3 ผู้ช่วยบริการลูกค้าและการร่างเอกสาร	19
1.6.4 ตัวอย่าง: การใช้งานอื่นๆ และข้อควรระวัง	20
1.7 ลงมือทำ: ใช้ TensorFlow บน Google Colab พยากรณ์อุณหภูมิล่วงหน้า	21
1.7.1 ขั้นที่ 0: รู้จัก TensorFlow, Keras และ Google Colab22	
1.7.2 ขั้นที่ 1: เปิดสมุดบันทึกใหม่บน Colab	22
1.7.3 ขั้นที่ 2 (ทางเลือก): เลือกใช้ GPU ให้ฝึกเร็วขึ้น	23
1.7.4 ขั้นที่ 3: พิมพ์และรันเซลล์โค้ดแรก	24
1.7.5 ขั้นที่ 4: เตรียมข้อมูลอุณหภูมิ	25
1.7.6 ขั้นที่ 5: ทำให้เป็นมาตรฐานและแบ่งชุดข้อมูล	27
1.7.7 ขั้นที่ 6: สร้างหน้าต่างข้อมูล (Windowing)	29
1.7.8 ขั้นที่ 7: ประกอบแบบจำลองด้วย Keras	29
1.7.9 ขั้นที่ 8: คอมไพล์และฝึกแบบจำลอง	31
1.7.10 ขั้นที่ 9: ประเมินและพยากรณ์	33

1.7.11	ชั้นที่ 10: ตีความผลและเชื่อมโยงสู่การเงิน	35
1.7.12	ทำให้แบบจำลองดีขึ้น: การหยุดฝึกอัตโนมัติและการลด Overfitting	36
1.7.13	พยากรณ์ล่วงหน้าหลายชั่วโมง	37
1.8	ข้อจำกัดของ Generative AI และการแก้ไขด้วย RAG	42
1.8.1	ข้อจำกัดสำคัญที่ต้องระวัง	43
1.8.2	RAG: ให้คำตอบยึดกับเอกสารจริง	44
	สรุปท้ายบท	46
	กิจกรรมท้ายบท	47
	หนังสือและเอกสารอ้างอิง	49

บทที่ 2 การใช้งาน LLM และ Prompt Engineering แบบ No-Code 50

2.1	โครงสร้างของ Prompt ที่ดี	51
2.1.1	ทำความเข้าใจแต่ละองค์ประกอบ	52
2.1.2	ตัวอย่างเปรียบเทียบ: Prompt อ่อนกับ Prompt ที่ดี	53
2.1.3	ข้อผิดพลาดที่พบบ่อยในแต่ละองค์ประกอบ	54
2.2	เทคนิคสำคัญ	55
2.2.1	Few-shot Prompting: สอนด้วยตัวอย่าง	55
2.2.2	Chain-of-thought: ขอให้คิดเป็นขั้นตอน	56
2.2.3	การบังคับอ้างอิงและการระบุความไม่แน่นอน	57
2.2.4	เทคนิคเสริมที่ควรมี	57
2.2.5	ตัวอย่าง: Few-shot สำหรับจัดหมวดค่าใช้จ่าย	58
2.2.6	ตัวอย่าง: Chain-of-thought คำนวณอัตราส่วนทางการเงิน	58
2.2.7	Prompt เทมเพลตที่นำกลับมาใช้ซ้ำได้	59
2.2.8	ควบคุมโทน ความยาว และรูปแบบด้วยคำสั่งธรรมดา	59
2.2.9	ลงมือทำ: ใช้โปรแกรมจริง	60
2.3	ผลลัพธ์แบบมีโครงสร้าง	62
2.3.1	รูปแบบที่นิยมและเมื่อไรควรใช้	62
2.3.2	เจาะลึก JSON และการออกแบบ Schema	63
2.3.3	วินัยของ Schema: enum ฟิลด์บังคับ และการจัดการค่าว่าง	64
2.3.4	กฎคำสั่งให้ JSON เสถียร	64
2.3.5	การตรวจรับและแปลงผลลัพธ์	65
2.3.6	ตัวอย่าง: สกัดหลายรายการด้วย JSON ช้อน	66

2.3.7	เลือก JSON หรือ CSV	66
2.3.8	ส่งต่อผลลัพธ์เข้าสู่ขั้นถัดไป	67
2.4	ข้อควรระวังด้านความปลอดภัยและการรักษาความลับ	68
2.4.1	แนวปฏิบัติของ PDPC สิงคโปร์	68
2.4.2	พ.ร.บ. คุ้มครองข้อมูลส่วนบุคคล พ.ศ. 2562 ของไทย	69
2.4.3	ความเสี่ยงเฉพาะของ Generative AI	70
2.4.4	ตัวอย่าง: ปกปิดข้อมูลก่อน-หลัง	70
2.4.5	เช็กलिस्ट: จากหลักการสู่การปฏิบัติ	71
2.5	Prompt สร้างที่ปรึกษาการลงทุน	72
2.5.1	แยกโครงสร้างของ Prompt ตามหัวข้อประกอบ	72
2.5.2	หลักการเด่นที่ควรเรียนรู้	73
2.5.3	ตัวอย่าง: ที่ปรึกษาการขายประกัน	74
2.5.4	เดินตามแปดเฟสการค้นพบ	76
2.5.5	ตัวอย่าง: บทสนทนาจากค้นพบสู่การให้คำแนะนำ	77
2.5.6	Guardrails และคำสงวนสิทธิ์	77
	สรุปท้ายบท	78
	กิจกรรมท้ายบท	79
	หนังสือและเอกสารอ้างอิง	82

บทที่ 3 แนวคิดและสถาปัตยกรรมของ Agentic AI 84

3.1	จาก Generative AI สู่ Agentic AI และแนวทางประยุกต์ทางการเงิน	85
3.1.1	เอเจนต์ทำงานอย่างไร	86
3.1.2	แนวคิดจากหนังสือ The Agentic Bank	87
3.1.3	เอเจนต์เดียวกับระบบที่มียูเอไอ	89
3.1.4	บันไดวุฒิภาวะและแนวทางประยุกต์ทางการเงิน	89
3.2	วงจรการทำงานของ Agent	91
3.2.1	สี่ขั้นของวงจร: รับรู้ วางแผน ลงมือ ทบทวน	91
3.2.2	ReAct: การให้เหตุผลสลับกับการลงมือ	92
3.2.3	ตัวอย่าง: เอเจนต์ช่วยงานกระต่ายอดบัญชี	93
3.2.4	องค์ประกอบที่ทำให้วงจรทำงานได้ดี	93
3.2.5	สามรูปแบบการวางแผนที่พบบ่อย	94
3.2.6	เมื่อวงจรสะดุด: การจัดการข้อผิดพลาดและทางตัน	94

3.2.7	วงจรเอเจนต์ต่างจากระบบอัตโนมัติเดิมอย่างไร	95
3.2.8	ตัวอย่าง: ร่องรอยการทำงานที่ละเอียดรอบ	95
3.2.9	รู้ได้อย่างไรว่าเอเจนต์ทำงานได้ดี	96
3.2.10	ออกแบบเป้าหมายให้เอเจนต์บรรลุได้	96
3.3	องค์ประกอบสี่ส่วนของ Agent	97
3.3.1	องค์ประกอบที่ 1: โมเดล (Model) - สมongผู้ตัดสินใจ	98
3.3.2	เลือกโมเดลให้เหมาะกับงาน	98
3.3.3	องค์ประกอบที่ 2: เครื่องมือ (Tools) - มือและขาที่ลงมือทำ	98
3.3.4	องค์ประกอบที่ 3: หน่วยความจำ (Memory) - จดจำบริบทและบทเรียน	99
3.3.5	หน่วยความจำในทางปฏิบัติ: จำอะไร ลืมอะไร	99
3.3.6	องค์ประกอบที่ 4: คำสั่งกำกับ (Instructions) - เป้าหมายและกติกา	100
3.3.7	รูปแบบคำสั่งกำกับที่ดีในงานการเงิน	100
3.3.8	สี่องค์ประกอบทำงานร่วมกับวงจรอย่างไร	101
3.3.9	ตัวอย่าง: ผู้ช่วยวิเคราะห์การเงิน	101
3.3.10	ข้อผิดพลาดที่พบบ่อยในการออกแบบองค์ประกอบ	101
3.3.11	ตัวอย่าง: การปรับปรุงเอเจนต์ที่ละองค์ประกอบ	102
3.4	เครื่องมือ (Tools) และโปรโตคอลการเชื่อมต่อ	103
3.4.1	การเรียกใช้เครื่องมือทำงานอย่างไร	103
3.4.2	โปรโตคอลการเชื่อมต่อมาตรฐาน	104
3.4.3	ลงมือทำ: เชื่อมเครื่องมือเข้ากับเอเจนต์แบบ No-Code	104
3.4.4	ตัวอย่างคำขอเรียกเครื่องมือแบบมีโครงสร้าง	106
3.4.5	เครื่องมือกับการดึงข้อมูล (RAG) ในงานการเงิน	107
3.4.6	สิทธิ์และความปลอดภัยของการเชื่อมต่อ	107
3.4.7	ออกแบบเครื่องมือที่ดีและปลอดภัย	108
3.4.8	เลือกชุดเครื่องมือและทดสอบก่อนใช้จริง	109
3.4.9	ความปลอดภัยจากการแทรกคำสั่งผ่านข้อมูลนำเข้า	109
3.4.10	ต้นทุน ความเร็ว และความน่าเชื่อถือของเครื่องมือ	109
3.5	ระดับความเป็นอิสระและการกำกับโดยมนุษย์	110
3.5.1	บันไดระดับความเป็นอิสระ	110
3.5.2	การกำกับโดยมนุษย์ (Human-in-the-loop)	111
3.5.3	ธรรมาภิบาลของเอเจนต์ในงานการเงิน	112
3.5.4	ความเสี่ยงของความเป็นอิสระสูงและการบริหารจัดการ	112
3.5.5	กรณีศึกษา: กำหนดระดับความเป็นอิสระให้เอเจนต์บริหารสภาพคล่อง	112

3.5.6	ร่องรอยการตรวจสอบ (Audit Trail) ในทางปฏิบัติ	113
3.5.7	บริบทการกำกับดูแลในประเทศไทย	113
3.5.8	จากกฎสู่วัฒนธรรมการกำกับที่ดี	114
3.5.9	บทบาทใหม่ของนักการเงินในยุคเอเจนต์	114
3.5.10	ลำดับขั้นการนำเอเจนต์ไปใช้จริงอย่างปลอดภัย	115
	สรุปท้ายบท	116
	กิจกรรมท้ายบท	116
	หนังสือและเอกสารอ้างอิง	119

บทที่ 4 ระบบ Multi-Agent และการประสานงาน 120

4.1	ทำไมต้องมีทีมเอเจนต์	121
4.1.1	ข้อจำกัดของเอเจนต์เดี่ยว	121
4.1.2	ประโยชน์ของการแบ่งเป็นทีมเอเจนต์	122
4.1.3	ทีมเอเจนต์เทียบกับทีมพนักงาน	123
4.1.4	ความเข้าใจผิดที่พบบ่อยเกี่ยวกับระบบทีมเอเจนต์	123
4.1.5	เมื่อใดควรใช้ทีมเอเจนต์ และเมื่อใดไม่ควร	124
4.1.6	สัญญาณว่าถึงเวลาแยกเป็นทีมเอเจนต์	124
4.1.7	ต้นทุนที่ต้องแลก และการเริ่มต้นที่ดี	125
4.1.8	ตัวอย่าง: การแบ่งงานในงานการเงินที่พบบ่อย	125
4.2	รูปแบบการประสานงานที่นิยมกัน	126
4.2.1	รูปแบบที่ 1: ท่อต่อเนื่อง (Sequential Pipeline)	127
4.2.2	รูปแบบที่ 2: ผู้จัดการสั่งงานลูกทีม (Orchestrator-Worker)	128
4.2.3	รูปแบบที่ 3: ทำงานขนานแล้วรวมผล (Parallel + Aggregator)	128
4.2.4	รูปแบบที่ 4: กำหนดเส้นทางและส่งต่อ (Router / Handoff)	129
4.2.5	รูปแบบที่ 5: เสนอ-ค้าน-ตรวจทาน (Proposer / Critic)	130
4.2.6	รูปแบบที่ 6: ลำดับชั้น (Hierarchical Teams)	131
4.2.7	รูปแบบที่ 7: กระดานความรู้ร่วม (Shared Memory / Blackboard)	132
4.2.8	การสื่อสารและการจัดการสถานะระหว่างเอเจนต์	133
4.2.9	การจัดการข้อผิดพลาดในระบบทีมเอเจนต์	134
4.2.10	ตัวอย่าง: รูปแบบเดียวกันในงานการเงินจริง	134
4.2.11	การผสมผสานหลายรูปแบบในระบบเดียว	135
4.2.12	ต้นทุนและความซับซ้อนของการประสานงาน	135

4.2.13	เมื่อเลือกรูปแบบผิด: บทเรียนที่พบบ่อย	136
4.2.14	ออกแบบการสื่อสารระหว่างเอเจนต์ให้ดี	136
4.2.15	การจัดการสถานะร่วมและความสอดคล้องของข้อมูล	137
4.2.16	การติดตามและสังเกตการณ์ระบบ (Observability)	137
4.2.17	ความปลอดภัยในระบบทีมเอเจนต์	138
4.2.18	รายละเอียดและรูปแบบย่อยของแต่ละรูปแบบ	138
4.2.19	กรณีศึกษา: งานจัดพอร์ตด้วยสามรูปแบบ	139
4.2.20	บทบาทของเอเจนต์ผู้ประสานงาน	139
4.2.21	การออกแบบให้ระบบขยายขนาดได้	140
4.2.22	เครื่องมือและเฟรมเวิร์กสำหรับสร้างระบบทีมเอเจนต์	140
4.2.23	จากต้นแบบสู่การใช้งานจริง	141
4.2.24	ความสัมพันธ์กับวงจรการทำงานของเอเจนต์	142
4.2.25	การประเมินและปรับปรุงระบบทีมเอเจนต์	142
4.2.26	ตัวอย่าง: ปัญหาการประสานงานและวิธีแก้	143
4.2.27	บทบาทของมนุษย์ในระบบทีมเอเจนต์	143
4.2.28	การประสานงานในงานที่ต้องตอบสนองตามเวลาจริง	144
4.2.29	ข้อพิจารณาเชิงจริยธรรมและความรับผิดชอบ	144
4.2.30	หลักคิดสำคัญของการประสานงานทีมเอเจนต์	145
4.2.31	กรณีศึกษา: ระบบเฝ้าระวังความเสี่ยงพอร์ต	145
4.2.32	เลือกรูปแบบให้เหมาะกับงาน	146
4.3	ตัวอย่างการแบ่งบทบาทในงานวิเคราะห์หุ้น	147
4.3.1	การแบ่งบทบาทของเอเจนต์ทั้งสี่	148
4.3.2	เตรียมการ: เชื่อม กับ MCP Server	149
4.3.3	เอเจนต์ที่ 1: Data Collector - ดึงข้อมูลจริงผ่าน MCP	155
4.3.4	เอเจนต์ที่ 2: Quant Analyst - คำนวณตัวชี้วัด	157
4.3.5	เอเจนต์ที่ 3: Risk Reviewer - ตรวจสอบและคัดค้าน	158
4.3.6	เอเจนต์ที่ 4: Report Writer - เรียบเรียงรายงาน	159
4.3.7	Orchestrator: ประกอบทุกเอเจนต์เข้าด้วยกัน	160
4.3.8	ปรับให้ทำงานขนานเพื่อความเร็ว	161
4.3.9	เพิ่มการจัดการข้อผิดพลาดให้แข็งแรง	161
4.3.10	ทดสอบทีมเอเจนต์ก่อนใช้จริง	162
4.3.11	ต่อยอด: เพิ่มเอเจนต์เปรียบเทียบคู่แข่ง	163
4.3.12	อ่านและตรวจสอบร่องรอยการทำงานของทีม	163

4.3.13	ตัวอย่าง: ผลลัพธ์ของแต่ละชั้นโดยละเอียด	164
4.3.14	เลือกและกำหนดค่าโมเดลให้เหมาะกับแต่ละเอเจนต์	166
4.3.15	ข้อควรระวังด้านข้อมูลและการปฏิบัติตามกฎเกณฑ์	166
4.3.16	สรุปขั้นตอนการสร้างทีมเอเจนต์วิเคราะห์หุ้น	167
4.3.17	ผลลัพธ์: บทวิเคราะห์ที่ได้	167
4.3.18	ตัวอย่างบทวิเคราะห์ฉบับเต็มที่ทีมผลิตได้	168
4.3.19	กำหนดสัญญาการส่งต่อด้วยโครงสร้างข้อมูล	170
4.3.20	เพิ่มการตรวจสอบสัญญาที่รอยต่อ	171
4.3.21	ตัวอย่างโค้ดเอเจนต์เปรียบเทียบคู่แข่ง	172
4.3.22	บทเรียนสำคัญจากการสร้างทีมเอเจนต์	173
4.3.23	นำแนวทางไปปรับใช้กับงานการเงินอื่น	174
4.4	ความเสี่ยงและการควบคุมคุณภาพ	175
4.4.1	ความเสี่ยงสำคัญของระบบทีมเอเจนต์	175
4.4.2	กลไกควบคุมคุณภาพ	175
4.4.3	ความเสี่ยงเฉพาะของงานการเงิน	177
4.4.4	การติดตามและตอบสนองเมื่อเกิดปัญหา	177
4.4.5	คุณภาพคือวัฒนธรรม ไม่ใช่กลไก	177
	สรุปท้ายบท	178
	กิจกรรมท้ายบท	179
	หนังสือและเอกสารอ้างอิง	181

บทที่ 5 Python เบื้องต้นสำหรับงานอัตโนมัติทางการเงิน (Low-Code) 182

5.1	พื้นฐานภาษา Python	183
5.1.1	ทำไมต้องเรียน Python ในงานการเงิน	183
5.1.2	เริ่มต้นใช้งาน: Google Colab	184
5.1.3	ตัวแปรและชนิดข้อมูลพื้นฐาน	186
5.1.4	การแปลงชนิดข้อมูล	187
5.1.5	การคำนวณและตัวดำเนินการ	188
5.1.6	โครงสร้างข้อมูล: list และ dict	188
5.1.7	การควบคุมการทำงาน: เงื่อนไข การวนซ้ำ for และ while	190
5.1.8	ฟังก์ชัน: รวมโค้ดที่ใช้ซ้ำ	192
5.1.9	การจัดรูปแบบข้อความด้วย f-string	192

5.1.10	นำเข้าไลบรารีเพื่อใช้เครื่องมือสำเร็จรูป	193
5.1.11	ความคิดเห็นและสไตล์การเขียนที่ดี	193
5.1.12	ข้อผิดพลาดที่พบบ่อยและวิธีแก้	193
5.1.13	ตัวอย่างรวม: เครื่องคิดเลขการลงทุนอย่างง่าย	194
5.2	ตัวอย่าง: การคำนวณ NPV ด้วย Python	197
5.3	pandas สำหรับจัดการข้อมูลการเงิน	197
5.3.1	DataFrame คืออะไร	197
5.3.2	ขั้นที่ 1: นำเข้าและสร้างข้อมูล	198
5.3.3	ขั้นที่ 2: ดูและเลือกข้อมูล	199
5.3.4	ขั้นที่ 3: คำนวณคอลัมน์ใหม่และกรองข้อมูล	199
5.3.5	ขั้นที่ 4: จัดกลุ่มและสรุปผล	200
5.3.6	ขั้นที่ 5: คำนวณผลตอบแทนและบันทึกผล	200
5.3.7	เรียงลำดับและจัดการข้อมูลที่ขาดหาย	201
5.3.8	รวมข้อมูลจากหลายตาราง	202
5.3.9	ทำงานกับวันที่และข้อมูลอนุกรมเวลา	202
5.4	ใช้ AI ช่วยเขียนโค้ดอย่างปลอดภัย	203
5.4.1	ขั้นตอนการทำงานกับ AI	204
5.4.2	เขียนคำขอให้ได้โค้ดที่ดี	204
5.4.3	อ่านและทำความเข้าใจโค้ด	205
5.4.4	ทดสอบกับข้อมูลที่รู้คำตอบ	205
5.4.5	ปรับปรุงโค้ดที่ละรอบกับ AI	206
5.4.6	ให้ AI อธิบายโค้ดเพื่อเรียนรู้	206
5.4.7	ตัวอย่าง: Debug โค้ดที่ AI เขียนผิด	206
5.4.8	บันทึกและอธิบายโค้ดเพื่อการตรวจสอบ	207
5.4.9	ข้อจำกัดของ AI ในการเขียนโค้ด	207
5.4.10	รายการตรวจสอบความปลอดภัย	208
5.5	Python ในฐานะเครื่องมือของ Agent	209
5.5.1	ทำไมเอเจนต์ต้องใช้ Python	209
5.5.2	ตัวอย่าง: เอเจนต์ใช้ Python คำนวณ	210
5.5.3	รันโค้ดของเอเจนต์อย่างปลอดภัย	211
5.5.4	เชื่อมโยงกับเครื่องมือและ MCP	212
5.5.5	เมื่อใดควรให้เอเจนต์ใช้โค้ด และเมื่อใดไม่ควร	212
5.5.6	ตัวอย่าง: เอเจนต์วิเคราะห์ข้อมูลด้วย pandas	212

5.5.7	ข้อจำกัดและสิ่งที่ต้องระวัง	213
5.5.8	ตรวจสอบร่องรอยการรันโค้ดของเอเจนต์	213
5.5.9	จากผู้ให้ข้อมูลผู้ออกแบบเครื่องมือ	213
	สรุปท้ายบท	214
	กิจกรรมท้ายบท	215
	หนังสือและเอกสารอ้างอิง	217

บทที่ 6 การสร้าง Workflow อัตโนมัติด้วย n8n (No-Code/Low-Code) 218

6.1	แนวคิดพื้นฐานของ n8n	219
6.1.1	No-Code/Low-Code คืออะไร	219
6.1.2	ทำไมงานการเงินจึงเหมาะกับเวิร์กโฟลว์อัตโนมัติ	220
6.1.3	รู้จักหน้าจอตว์แก้ไขของ n8n	221
6.1.4	โหนด (Node): หน่วยพื้นฐานของการทำงาน	221
6.1.5	ประเภทของโหนดที่ใช้บ่อยในงานการเงิน	222
6.1.6	ตัวกระตุ้น (Trigger): จุดเริ่มของเวิร์กโฟลว์	223
6.1.7	การเชื่อมโหนดและการไหลของข้อมูล	223
6.1.8	ข้อมูลและ Expression: ส่งค่าระหว่างโหนด	224
6.1.9	การจัดการข้อมูลหลายรายการ	225
6.1.10	จัดรูปแบบข้อมูลด้วยโหนด Set และ Code	225
6.1.11	ข้อมูลรับรองและประวัติการรัน	226
6.1.12	เมื่อใดควรใช้ n8n และเมื่อใดควรเขียนโค้ดเอง	226
6.1.13	ข้อดีและข้อจำกัดของแนวทาง No-Code	227
6.1.14	ภาพรวมระบบนิเวศของเวิร์กโฟลว์อัตโนมัติ	227
6.1.15	จัดระเบียบและตั้งชื่อเวิร์กโฟลว์ให้ดี	228
6.1.16	การนำเวิร์กโฟลว์กลับมาใช้ซ้ำและแบ่งปัน	228
6.1.17	กรณีใช้งานเวิร์กโฟลว์ในงานการเงิน	228
6.1.18	จาก n8n สู่อัตโนมัติที่ใหญ่ขึ้น	229
6.1.19	n8n เทียบกับเครื่องมืออัตโนมัติอื่น	229
6.1.20	ติดตั้งและเริ่มใช้งาน n8n	230
6.1.21	การทดสอบและแก้ปัญหาเวิร์กโฟลว์เบื้องต้น	234
6.1.22	เคล็ดลับสำหรับผู้เริ่มต้นใช้ n8n	235
6.2	เตรียม LINE Official Account และ Messaging API	238

6.2.1	ทำไมต้องใช้ LINE Official Account	238
6.2.2	ขั้นที่ 1: สมัคร LINE Official Account	239
6.2.3	ขั้นที่ 2: เปิดใช้งาน Messaging API	240
6.2.4	ขั้นที่ 3: ออก Channel Access Token	242
6.2.5	ขั้นที่ 4: เก็บโทเคนเป็น Credential ใน n8n	244
6.2.6	ทำความเข้าใจวิธีส่งข้อความผ่าน LINE ใน n8n	246
6.3	ตัวอย่างเวิร์กโฟลว์: แจ้งเตือนเมื่อหุ้นถึงราคาเป้าหมาย	247
6.3.1	ภาพรวมและการวางแผน	247
6.3.2	เข้าใจข้อมูลที่ไหลผ่านเวิร์กโฟลว์นี้	248
6.3.3	เตรียมข้อมูลรับรองก่อนเริ่ม	249
6.3.4	ขั้นที่ 1: เพิ่มตัวกระตุ้นตามเวลา	249
6.3.5	ขั้นที่ 2: ดึงราคาหุ้นด้วย HTTP Request	250
6.3.6	ขั้นที่ 3: ตรวจสอบว่าราคาถึงเป้าหมายหรือยัง	251
6.3.7	ขั้นที่ 4: ส่งข้อความแจ้งเตือน	252
6.3.8	ทดสอบและเปิดใช้งานจริง	254
6.3.9	ปัญหาที่พบบ่อยและวิธีแก้	254
6.3.10	ปรับแต่งข้อความแจ้งเตือนให้มีประโยชน์	255
6.3.11	ต่อยอดเวิร์กโฟลว์	255
6.3.12	ออกแบบเงื่อนไขการแจ้งเตือนให้ฉลาดขึ้น	255
6.3.13	บันทึกประวัติราคาเพื่อวิเคราะห์ภายหลัง	256
6.3.14	ขยายเป็นการติดตามหลายหุ้นพร้อมกัน	256
6.3.15	เลือกช่องทางแจ้งเตือนให้เหมาะกับการใช้งาน	257
6.3.16	สรุปขั้นตอนการสร้างเวิร์กโฟลว์นี้	257
6.3.17	ข้อควรระวังด้านความปลอดภัยของเวิร์กโฟลว์นี้	258
6.4	การเชื่อม AI กับข้อมูลภายนอก	259
6.4.1	ทำไมต้องเชื่อม AI กับข้อมูลจริง	259
6.4.2	ตั้งค่าโหมด AI ในเวิร์กโฟลว์	260
6.4.3	ตัวอย่าง: สรุปข่าวการเงินอัตโนมัติ	261
6.4.4	เชื่อมโยงกับเครื่องมือและเอเจนต์	261
6.4.5	ตรวจสอบและควบคุมคุณภาพผลลัพธ์ของ AI	262
6.4.6	ต่อยอด: จากสรุปสู่การวิเคราะห์	262
6.4.7	ตัวอย่างกรณีใช้งาน AI ในงานการเงิน	263
6.4.8	สรุปแนวทางผสาน AI กับข้อมูลในเวิร์กโฟลว์	263

6.5	การจัดการข้อผิดพลาดและขั้วรวมขา	264
6.5.1	การจัดการข้อผิดพลาด	264
6.5.2	ขั้วรวมขาของเวอร์กโฟลว์	265
6.5.3	ทดสอบ ติดตาม และดูแลเวอร์กโฟลว์ต่อเนื่อง	266
6.5.4	บทบาทของมนุษย์และการกำหนดความรับผิดชอบ	267
	สรุปท้ายบท	268
	กิจกรรมท้ายบท	268
	หนังสือและเอกสารอ้างอิง	271

บทที่ 7 การสร้างและใช้งาน AI Agent 272

ด้วย OpenClaw และ Browser Automation

7.1	OpenClaw คืออะไร	273
7.1.1	OpenClaw ทำงานอย่างไร	273
7.1.2	เอเจนต์ต่างจากเวอร์กโฟลว์อย่างไร	274
7.1.3	ความเหมือนของทั้งสองเครื่องมือ	275
7.1.4	เมื่อใดควรใช้เอเจนต์ เมื่อใดควรใช้เวอร์กโฟลว์	275
7.1.5	ตัวอย่าง: การใช้ OpenClaw ในงานการเงิน	276
7.1.6	ความเข้าใจผิดที่พบบ่อยเกี่ยวกับเอเจนต์	276
7.1.7	สิ่งที่ต้องเตรียมก่อนเริ่มใช้ OpenClaw	277
7.1.8	บทบาทใหม่ของคนทำงานการเงินในยุคเอเจนต์	277
7.1.9	OpenClaw ในภาพรวมของเครื่องมือเอเจนต์	278
7.2	สถาปัตยกรรมสามชั้น	279
7.2.1	ภาพรวมของสามชั้น	279
7.2.2	ชั้น Channel: ช่องทางรับ-ส่ง	280
7.2.3	ชั้น Brain: สมองที่คิดและตัดสินใจ	280
7.2.4	ชั้น Body: มือที่ลงมือทำ	280
7.2.5	คำขอไหลผ่านสามชั้นอย่างไร	280
7.2.6	ทำไมต้องแยกเป็นสามชั้น	281
7.2.7	เปรียบเทียบสามชั้นกับการทำงานของมนุษย์	282
7.2.8	ข้อพิจารณาในการออกแบบแต่ละชั้น	283
7.2.9	ติดตามการทำงานข้ามสามชั้น	283
7.2.10	เริ่มต้นออกแบบเอเจนต์จากสามชั้น	283

7.3	แนวคิด Skill	284
7.3.1	Skill คืออะไร	285
7.3.2	ทำไม Skill จึงเป็น Low-Code อย่างแท้จริง	285
7.3.3	Skill เทียบกับโค้ดและเวิร์กโฟลว์	286
7.3.4	ตัวอย่าง Skill: หาราคาประเมินที่ดิน	286
7.3.5	ตัวอย่าง: Skill สร้างงบการเงิน	287
7.3.6	เอเจนต์ทำตาม Skill อย่างไร	288
7.3.7	การเขียน Skill ที่ดี	289
7.3.8	ข้อผิดพลาดที่พบบ่อยในการเขียน Skill	290
7.3.9	ทดสอบ Skill ก่อนใช้งานจริง	290
7.3.10	จาก Skill สู่งานจริง	291
7.3.11	วงจรของ Skill: สร้าง ใช้ซ้ำ แบ่งปัน	291
7.3.12	ความเชื่อมโยงกับ Claude Skills	292
7.3.13	ข้อจำกัดของแนวคิด Skill	292
7.3.14	Skill ในบริบทของงานการเงิน	293
7.4	ลงมือทำ: ติดตั้งและใช้ OpenClaw จริงด้วย Gemini	298
7.4.1	ขั้นที่ 1: ติดตั้ง OpenClaw	298
7.4.2	ขั้นที่ 2: เชื่อม OpenClaw กับ Gemini (ฟรี)	299
7.4.3	ขั้นที่ 3: ส่งงานเอเจนต์ครั้งแรก	302
7.4.4	ขั้นที่ 4: ลองสร้างและใช้ Skill ง่ายๆ	304
7.5	Browser Automation: เมื่อเว็บไม่มี API	307
7.5.1	API คืออะไร และทำไมบางเว็บจึงไม่มี	307
7.5.2	เอเจนต์ควบคุมเบราว์เซอร์อย่างไร	308
7.5.3	เปรียบเทียบกับกรดิงข้อมูลผ่าน API	309
7.5.4	ความท้าทายและความเปราะบาง	310
7.5.5	แนวทางทำให้ Browser Automation น่าเชื่อถือขึ้น	311
7.5.6	เมื่อใดไม่ควรใช้ Browser Automation	311
7.5.7	ตัวอย่าง: ดึงข้อมูลจากเว็บราชการ	312
7.5.8	บทบาทในโครงการบทที่ 11-13	312
7.6	ความปลอดภัยและความรับผิดชอบ	314
7.6.1	กฎความปลอดภัยสำหรับเอเจนต์ที่เข้าถึงระบบ	315
7.6.2	ระวัง Prompt Injection จากเนื้อหาเว็บ	315
7.6.3	ข้อพึงปฏิบัติทางกฎหมายและจริยธรรม	316

7.6.4	สร้างวัฒนธรรมการใช้เอเจนต์อย่างรับผิดชอบ	317
7.6.5	สรุปหลักความปลอดภัยสำหรับเอเจนต์	317
	สรุปท้ายบท	318
	กิจกรรมท้ายบท	319
	หนังสือและเอกสารอ้างอิง	321

บทที่ 8 หลักการสร้างโครงงาน Agentic AI ทางการเงิน 322

8.1	โครงสร้างมาตรฐานของทุกโครงงาน	323
8.1.1	หกอองค์ประกอบและคำถามนำ	324
8.1.2	ลำดับงานและร่องรอยที่ต้องเก็บ	325
8.1.3	ตัวอย่าง: แมปกรอบหกอองค์ประกอบเข้ากับโครงงานจริง	327
8.1.4	โครงงานที่ดีต่างจากโครงงานที่อ่อนอย่างไร	328
8.2	การเตรียมสภาพแวดล้อม	329
8.2.1	เครื่องมือที่ต้องเตรียม	329
8.2.2	ลงมือทำ: เปิด Google Colab และติดตั้งไลบรารี xAI	330
8.2.3	การจัดการกุญแจลับและข้อมูล	332
8.2.4	รายการตรวจสอบความพร้อมก่อนเริ่มโครงงาน	333
8.3	การตรวจสอบและการคำนวณซ้ำ	334
8.3.1	แยกให้ชัด: Verification กับ Reperformance	334
8.3.2	ลงมือทำ: คำนวณซ้ำค่า NPV	335
8.3.3	การตรวจสอบข้อเท็จจริงกับแหล่งปฐมภูมิ	336
8.3.4	ทำให้ผลลัพธ์ทำซ้ำได้ (Reproducibility)	337
8.3.5	การวิเคราะห์ความอ่อนไหวของสมมติฐาน	337
8.3.6	บันทึกการตรวจสอบที่ส่งมอบได้	339
8.4	การใช้ xAI: อธิบายแบบจำลองกล่องดำ	339
8.4.1	Blackbox คืออะไร และเมื่อใดต้องทำรายงาน xAI	340
8.4.2	ภูมิทัศน์เครื่องมือ xAI	341
8.4.3	SHAP: ส่วนเสริมของแต่ละตัวแปร	342
8.4.4	LIME: แบบจำลองตัวแทนเฉพาะจุด	345
8.4.5	SHAP กับ LIME ต่างกันอย่างไร	346
8.4.6	ขั้นตอนจัดทำรายงาน xAI ประกอบโครงงาน	347
8.4.7	ข้อควรระวังในการใช้ xAI	347

8.5	เอกสารกำกับและความเชื่อถือทางวิชาการ	348
8.5.1	เอกสารกำกับโมเดล (Model Card)	349
8.5.2	ตัวอย่าง: เอกสารกำกับโมเดลที่ครอบคลุม	350
8.5.3	ความเชื่อถือทางวิชาการ	351
8.5.4	คำชี้แจงการใช้ AI และร่องรอยการตรวจสอบ	351
8.5.5	การอ้างอิงการใช้เครื่องมือ AI ในงานวิชาการ	352
	สรุปท้ายบท	354
	กิจกรรมท้ายบท	354
	หนังสือและเอกสารอ้างอิง	357

บทที่ 9 โครงการที่ 1: วิเคราะห์งบการเงินและสร้าง MD&A ด้วย Claude Skills 358

9.1	แนวคิด Skill กับงานที่ทำซ้ำ	359
9.1.1	Skill คืออะไร และเหมาะกับงานแบบใด	359
9.1.2	ตัวอย่างโครงสร้าง Skill วิเคราะห์งบการเงิน	361
9.1.3	การเปิดเผยข้อมูลแบบสามระดับ	362
9.1.4	Skill กับ Plugin ต่างกันอย่างไร	363
9.2	ขั้นตอนการทำโครงการ	365
9.2.1	กำหนดโจทย์และเลือกบริษัท	365
9.2.2	เตรียมและเข้าถึงข้อมูล	366
9.2.3	ลงมือทำ: เรียกใช้ Skill ใน Claude	367
9.2.4	สกัดตัวเลขและคำนวณอัตราส่วน	371
9.2.5	ตรวจสอบและคำนวณซ้ำ	372
9.2.6	ปรับปรุง Skill เมื่อผลยังไม่ได้มาตรฐาน	374
9.3	โครงร่าง MD&A ที่แนะนำ	375
9.3.1	ส่วนที่ 1: ภาพรวมธุรกิจและภาวะอุตสาหกรรม	376
9.3.2	ส่วนที่ 2: ผลการดำเนินงาน	376
9.3.3	ส่วนที่ 3: ฐานะการเงินและสภาพคล่อง	377
9.3.4	ส่วนที่ 4: กระแสเงินสดและการจัดหาเงินทุน	377
9.3.5	ส่วนที่ 5: อัตราส่วนสำคัญและการวิเคราะห์แนวโน้ม	378
9.3.6	ส่วนที่ 6: ความเสี่ยง ปัจจัยไม่แน่นอน และแนวโน้ม	379
9.3.7	ส่วนที่ 7: ประเมินการทางบัญชีที่สำคัญและข้อจำกัด	379

9.3.8 ตัวอย่าง: หลักการเขียนรายงาน MD&A ที่ดี	380
9.3.9 ตัวอย่าง: MD&A ฉบับย่อและขั้นตอนให้ AI ร่างแล้วตรวจทาน	381
9.4 การตรวจสอบและข้อควรระวัง	383
สรุปท้ายบท	384
กิจกรรมท้ายบท	384
หนังสือและเอกสารอ้างอิง	387

บทที่ 10 โครงการที่ 2: วิเคราะห์ความเป็นไปได้ของโครงการและ NPV ด้วย Claude Cowork 388

10.1 Cowork กับงานหลายขั้นตอน	389
10.1.1 Cowork คืออะไร	389
10.1.2 Cowork ต่างจากแชตทั่วไปอย่างไร	390
10.1.3 สภาพแวดล้อมการทำงานของ Cowork	391
10.1.4 เหมาะกับงานประเมินโครงการอย่างไร	392
10.1.5 ลงมือทำ: ตั้งค่าและมอบหมายงานใน Cowork ที่ละขั้น	392
10.1.6 ตัวอย่างบทสนทนาการมอบหมายงาน	394
10.1.7 การจัดระเบียบโฟลเดอร์โครงการ	395
10.1.8 ความปลอดภัย สิทธิ์ และการกำกับ	396
10.1.9 การทำงานหลายงานพร้อมกันและข้อจำกัด	397
10.1.10 ข้อจำกัดและการใช้อย่างมีวิจารณญาณ	397
10.1.11 ทางเลือก: ทำงานในแอป Office ด้วย Claude for Microsoft	398
10.2 ขั้นตอนการประเมินโครงการ	399
10.2.1 กำหนดขอบเขตและสมมติฐาน	399
10.2.2 ให้ Cowork สร้างแบบจำลองกระแสเงินสด	400
10.2.3 คำนวณ NPV, IRR และระยะคืนทุน แล้วตรวจสอบสูตร	401
10.2.4 สรุปข้อเสนอแนะและจัดทำเอกสารกำกับ	402
10.2.5 ปัญหาที่พบบ่อยและวิธีแก้	403
10.3 ตัวอย่างการรอบการคำนวณ NPV	404
10.3.1 โครงสร้างแบบจำลองกระแสเงินสด	405
10.3.2 คำนวณ NPV, IRR และระยะคืนทุน	406
10.4 การวิเคราะห์ความอ่อนไหวและสถานการณ์	408
10.4.1 การวิเคราะห์ความอ่อนไหว (Sensitivity)	409

10.4.2	การวิเคราะห์จุดคุ้มทุนของสมมติฐาน	410
10.4.3	การวิเคราะห์สถานการณ์ (Scenario)	411
10.4.4	การนำเสนอผลความเสี่ยงต่อผู้บริหาร	412
10.5	การตรวจสอบด้วย xAI	413
10.5.1	เมื่อใดที่การพยากรณ์กลายเป็นกล่องดำ	414
10.5.2	การประยุกต์ SHAP และ LIME กับการพยากรณ์รายได้	414
10.5.3	ตัวอย่าง: การอ่านผล SHAP และการเลือกเครื่องมือ	415
10.5.4	การบันทึกผล xAI ในเอกสารกำกับ	416
	สรุปท้ายบท	418
	กิจกรรมท้ายบท	418
	หนังสือและเอกสารอ้างอิง	421

บทที่ 11 โครงการที่ 3: ค้นข้อมูลและส่งคำสั่งซื้อขายหุ้นผ่านเว็บด้วย n8n

11.1	สถาปัตยกรรมเวิร์กโฟลว์	423
11.1.1	n8n คืออะไร และเหมาะกับงานนี้อย่างไร	424
11.1.2	ทำไมเลือก n8n และข้อพิจารณาในการเลือกเครื่องมือ	424
11.1.3	องค์ประกอบของเวิร์กโฟลว์	425
11.1.4	บันไดสามขั้น: จากโมเดลภาษาสู่ AI Agent	426
11.1.5	สถาปัตยกรรมเวิร์กโฟลว์ซื้อขาย	426
11.1.6	ติดตั้ง n8n แบบ Self-Host (ตามบทที่ 6)	427
11.1.7	ลงมือทำ: สร้างเวิร์กโฟลว์ที่ละขั้น	428
11.1.8	การตั้งค่าโหนด AI Agent และ Credentials	430
11.1.9	ตัวอย่าง: การกำหนดค่าโหนดและการเชื่อมข้อมูล	433
11.1.10	การทดสอบและ Execution Log	434
11.2	กฎความเสี่ยงที่ต้องฝังในเวิร์กโฟลว์	434
11.2.1	เกณฑ์ความมั่นใจ	435
11.2.2	เพดานขนาดรายการ	435
11.2.3	เพดานต่อวัน	436
11.2.4	การอนุมัติของมนุษย์	438
11.2.5	การเรียงลำดับและการรวมกฎทั้งสี่ด้าน	441
11.3	การส่งคำสั่งผ่านเว็บอย่างปลอดภัย	441
11.3.1	เรียกใช้ API ทางกรอก่อนเสมอ	442

11.3.2	การยืนยันตัวตนและการรักษาความลับ	444
11.3.3	Browser Automation เมื่อไม่มี API	445
11.3.4	ป้องกันการทำการซ้ำและการยืนยันก่อนส่ง	446
11.3.5	การบันทึกร่องรอยเพื่อตรวจสอบย้อนหลัง	447
11.3.6	การจำกัดอัตราและการจัดการคิวคำขอ	448
11.3.7	การเฝ้าระวังและการแจ้งเตือน	448
11.3.8	การปฏิบัติตามกฎหมายและกฎของหน่วยงานกำกับ	448
11.3.9	การทยอยเปิดใช้งานอย่างปลอดภัย	449
11.3.10	ลงมือทำ: ตั้งค่าการส่งคำสั่งอย่างปลอดภัย	450
11.3.11	ตัวอย่างวงจรคำสั่งครบหนึ่งรอบ	452
11.4	การจัดการข้อผิดพลาด	454
11.4.1	ออกแบบให้ล้มเหลวอย่างปลอดภัย	454
11.4.2	การจัดการข้อผิดพลาดใน n8n	455
11.4.3	การกู้คืนและการกลับมาทำงานหลังหยุด	456
	สรุปท้ายบท	456
	กิจกรรมท้ายบท	457
	หนังสือและเอกสารอ้างอิง	459

บทที่ 12 โครงการที่ 4: การวิเคราะห์สินเชื่อกะด้วยข้อมูลกรมที่ดิน และภาพถ่ายดาวเทียม

12.1	ภาพรวมและเครื่องมือที่ใช้	462
12.1.1	โครงการนี้ทำอะไร ทำไมต้องเอเจนต์ และบทบาทของมนุษย์	462
12.1.2	อินพุตและเอาต์พุตของระบบ	462
12.1.3	สถาปัตยกรรมห้าชั้น	463
12.1.4	เครื่องมือที่ใช้และการเตรียม	464
12.2	ลงมือ: ดึงข้อมูลราคาประเมินและทำเล	466
12.2.1	ดึงราคาประเมินด้วย OpenClaw (Browser Automation)	466
12.2.2	ดึงราคาประเมินด้วย Engine Python (วิธีที่ใช้ในโครงการนี้)	469
12.2.3	รวมข้อมูลและจัดการกรณีข้อมูลไม่ครบ	474
12.2.4	คุณภาพ ความสดใหม่ และการตรวจสอบข้อมูล	474
12.2.5	การกระหายอดและการจัดการข้อมูลที่ขัดแย้งกัน	475
12.2.6	ตัวอย่างชุดข้อมูลที่ดึงได้ครบหนึ่งกรณี	475

12.3	ลงมือ: ประเมินมูลค่าหลักประกันและคำนวณ LTV	476
12.3.1	รวมข้อมูลและปรับมูลค่าหลักประกัน	476
12.3.2	คำนวณและตีความ LTV เทียบเพดาน	478
12.3.3	ความอ่อนไหวของ LTV และระยะปลอดภัย	480
12.3.4	ความเสี่ยงหลักประกันและความสามารถชำระหนี้	482
12.4	ลงมือ: ด้านกำกับ ความเป็นธรรม การอธิบายผล และการอนุมัติ	485
12.4.1	การอธิบายผล (Explainability)	485
12.4.2	ความเป็นธรรมและการไม่เลือกปฏิบัติ	486
12.4.3	ด้านกำกับ เพดาน LTV และการออกแบบคำสั่ง	488
12.4.4	การอนุมัติของมนุษย์ ร่องรอย และการเชื่อมต่อระบบจริง	489
12.5	ประกอบทุกขั้นตอนเข้าด้วยกันและสรุป	492
12.5.1	ข้อจำกัดและความเสี่ยงของระบบ	494
12.5.2	การจับคู่กับกรอบองค์ประกอบของบทที่ 8	494
	สรุปท้ายบท	495
	กิจกรรมท้ายบท	495
	หนังสือและเอกสารอ้างอิง	499

บทที่ 13 โครงการที่ 5: รายงานวิเคราะห์หุ้นด้วยทีม Agent 500

13.1	การแบ่งบทบาทของทีม Agent	501
13.1.1	ผู้เก็บข้อมูล (Data Collector)	502
13.1.2	นักวิเคราะห์เชิงปริมาณ (Quantitative Analyst)	502
13.1.3	นักวิเคราะห์ข่าว (News Analyst)	503
13.1.4	ผู้ตรวจทาน (Reviewer หรือ Critic)	503
13.1.5	ผู้เขียนรายงาน (Report Writer)	503
13.2	กระบวนการวิเคราะห์	504
13.2.1	ขั้นที่ 1: กำหนดหุ้นเป้าหมายและกรอบการวิเคราะห์	504
13.2.2	ขั้นที่ 2: ผู้เก็บข้อมูลดึงข้อมูลจากเว็บ	506
13.2.3	ขั้นที่ 3: คำนวณอัตราส่วนและตรวจซ้ำ	509
13.2.4	ขั้นที่ 4: ประเมิน Sentiment ข่าวและเหตุการณ์สำคัญ	514
13.2.5	ขั้นที่ 5: ผู้ตรวจทานด้านข้อสรุปก่อนเขียนรายงาน	518
13.2.6	การประสานงานและการส่งต่อระหว่าง Agent	520
13.2.7	การจัดการเมื่อ Agent ให้ผลขัดแย้งกัน	521

13.2.8	การออกแบบคำสั่งและเครื่องมือของแต่ละ Agent	521
13.2.9	การทดสอบและปรับปรุงทีม Agent	522
13.2.10	สรุปกระบวนการและการเชื่อมโยงทั้งห้าขั้น	523
13.3	ตัวอย่างผลลัพธ์มีโครงสร้างจาก Agent วิเคราะห์	524
13.3.1	โครงสร้างของรายงาน	524
13.3.2	ผลลัพธ์เชิงโครงสร้าง (Schema)	525
13.3.3	การเรียบเรียงรายงาน	526
13.3.4	การเขียนแหล่งอ้างอิงในภาคผนวก	532
13.3.5	ข้อสังเกตเรื่องการประเมินมูลค่าและการเปรียบเทียบ	533
13.3.6	การนำรายงานไปใช้และข้อควรระวัง	534
13.4	การตรวจสอบและการลดอคติ	535
13.4.1	การตรวจสอบความถูกต้อง	535
13.4.2	อคติที่พบบ่อยและวิธีลด	536
13.4.3	ความรับผิดชอบและข้อจำกัดของระบบ	536
	สรุปท้ายบท	537
	กิจกรรมท้ายบท	537
	หนังสือและเอกสารอ้างอิง	541

บทที่ 14 โครงการที่ 6: การตัดสินใจจ่ายเงินปันผล **542**

ด้วยระบบทีม Agent

14.1	ทำไมการตัดสินใจปันผลจึงเป็นโจทย์หลายมิติ	543
14.2	โครงสร้างทีม Agent	545
14.2.1	บทบาทของแต่ละเอเจนต์	546
14.2.2	การประสานงานและการไหลของข้อมูล	549
14.2.3	การออกแบบคำสั่งของแต่ละเอเจนต์	550
14.2.4	การประกอบทีมที่ละเอียด	551
14.2.5	การจัดการเมื่อข้อมูลไม่พอหรือ Agent ทำงานไม่สำเร็จ	552
14.3	กรอบการตัดสินใจและตัวอย่างที่ตรวจสอบได้	553
14.3.1	แนวคิดพื้นฐานนโยบายปันผลและสัญญา	553
14.3.2	เกณฑ์และการนำเสนอ	554
14.3.3	ตัวอย่าง: ตัวเลขที่ตรวจสอบได้	556
14.4	การสังเคราะห์และการตรวจทาน	559

14.4.1	การสังเคราะห์และการระบุเป้าหมายที่ขัดแย้งกัน	559
14.4.2	การตรวจทานก่อนนำเสนอ	561
14.5	การบูรณาการความรู้ของตำรา	563
14.5.1	การ Prompt และ Skill (บทที่ 2, 9)	563
14.5.2	การประสานงานหลายเอเจนต์ (บทที่ 4)	564
14.5.3	การคำนวณที่ตรวจสอบได้ (บทที่ 5, 10)	564
14.5.4	การดึงข้อมูลเว็บ (บทที่ 7, 11, 13)	565
14.5.5	ธรรมาภิบาลและการตรวจสอบ (บทที่ 8)	565
14.5.6	ภาพรวมการบูรณาการ	566
14.5.7	การใช้ AI เชิงเอเจนต์อย่างรับผิดชอบ	568
	กิจกรรมท้ายบท	569
	หนังสือและเอกสารอ้างอิง	572

บทที่ 15 การใช้โมเดลภาษาแบบรันในเครื่อง (Local LLM) 574

15.1	ทำไมงานบางงานจึงต้องใช้โมเดลในเครื่อง	575
15.2	รู้จัก Ollama และ Qwen	576
15.3	ชนิดของโมเดล: อ่านข้อความ รูปภาพ หรือฟังเสียง	577
15.4	เลือกขนาดโมเดลให้เหมาะกับเครื่อง	578
15.5	ลงมือทำ: ติดตั้ง Ollama และรัน Qwen	578
15.5.1	ติดตั้ง Ollama	578
15.5.2	ใช้งานผ่านแอป Ollama (หน้าจอกกราฟิก)	580
15.5.3	ดาวน์โหลดและรันโมเดล Qwen	581
15.6	ใช้ Qwen ในเครื่องกับ OpenCLAW	582
15.6.1	สลับ Brain ระหว่าง Qwen ในเครื่องกับ Gemini บนคลาวด์	583
15.7	ใช้ Qwen ในเครื่องกับ n8n	584
15.8	สถาปัตยกรรมระบบ: วาง Brain ไว้ในเครื่องหรือบนคลาวด์	586
15.9	ข้อจำกัดและข้อควรระวัง	587
	สรุปท้ายบท	587
	กิจกรรมท้ายบท	588
	หนังสือและเอกสารอ้างอิง	591
	ดัชนี	592
	เกี่ยวกับผู้เขียน	594

บทที่ 1

Generative AI และ Foundation Model สำหรับงานการเงิน

บทแรกนี้พาผู้เรียนเริ่มต้นการเดินทางเพื่อเรียนรู้ Agentic AI จากแนวคิดพื้นฐานที่สุดของปัญญาประดิษฐ์ คือ “โครงข่ายประสาทเทียม” ไปจนถึงเทคโนโลยีที่กำลังเปลี่ยนวิธีทำงานของวงการการเงินในวันนี้ คือ Generative AI และ Foundation Models เป้าหมายไม่ใช่เพียงให้ผู้เรียน “รู้จัก” เทคโนโลยีเหล่านี้ แต่ให้ “เข้าใจที่มา” ว่าแต่ละแนวคิดต่อยอดกันอย่างไร และที่สำคัญที่สุดคือได้ “ลงมือทำ” ด้วยตนเอง ผ่านกรณีศึกษาจริงจากงานวิจัยด้านการเงินไทย และการสร้างแบบจำลองพยากรณ์ด้วย TensorFlow บน Google Colab ผู้ที่ไม่คุ้นเคยกับการเขียนโปรแกรมคอมพิวเตอร์ ขอให้พยายามอ่าน และทำความเข้าใจด้วยตนเองไปก่อน และพยายามทำกิจกรรมตามที่หนังสือแนะนำทุกขั้นตอน เพราะภาษา Python ที่ใช้ในโปรแกรมเหล่านี้จะมีลักษณะใกล้เคียงกับภาษามนุษย์มาก เมื่อคุ้นเคยในระดับหนึ่ง เราจะมาศึกษาไวยากรณ์ของภาษา Python อย่างละเอียดในบทที่ 5 อย่งไรก็ตาม เนื่องจากหนังสือเล่มนี้ยึดหลัก Low Code เพราะผู้เรียนส่วนใหญ่คือนักการเงิน ประจวบกับเครื่องมือ AI สามารถช่วยมนุษย์เขียนโปรแกรมตั้งแต่ต้นได้แล้วด้วยการสั่งงานโดยภาษาธรรมชาติ (Prompting) ที่มีลักษณะแทบจะเหมือนกับภาษาคนทุกประการ สำคัญของการโค้ดดิ้งในหนังสือเล่มนี้จึงไม่ได้มุ่งพัฒนาผู้เรียนไปเป็นโปรแกรมเมอร์ แต่เรียนจำเป็นต้องเข้าใจศักยภาพและข้อจำกัดของโค้ดดิ้ง เพื่อจะได้รับการพัฒนาเป็นนักการเงินที่สามารถสั่งคอมพิวเตอร์ให้เขียนโปรแกรม และทำตามที่ต้องการได้โดยตรงด้วยการมี AI เป็นผู้ช่วย ไม่ต้องพึ่งพาโปรแกรมเมอร์เช่นในอดีต

วัตถุประสงค์การเรียนรู้

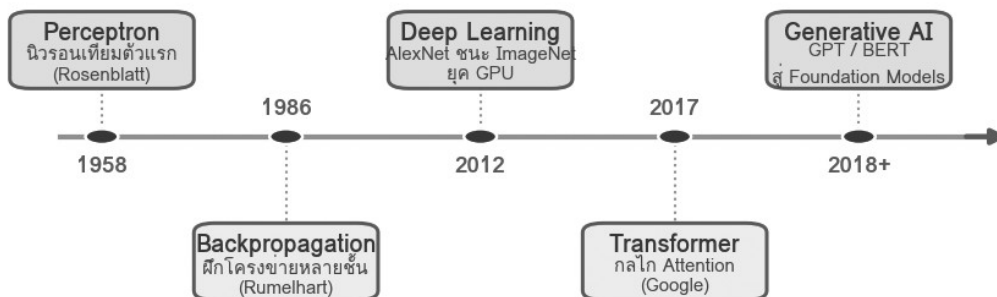
- 1 อธิบายวิวัฒนาการจากโครงข่ายประสาทเทียมสู่ Deep Learning และ Generative AI ได้
- 2 อธิบายแนวคิด word2vec ว่าทำให้คอมพิวเตอร์เข้าใจความหมายของคำได้อย่างไร และเชื่อมโยงกับกรณีศึกษาการพยากรณ์ราคาหุ้นของไทย
- 3 อธิบายได้ว่า LLM ต่อยอดจาก word2vec มาเป็นแบบจำลองภาษาขนาดใหญ่ได้อย่างไร และทำงานในระดับแนวคิดอย่างไร
- 4 ยกตัวอย่างการประยุกต์ Generative AI ในงานการเงินได้หลากหลาย
- 5 สร้างและฝึกแบบจำลองพยากรณ์อนุกรมเวลาด้วย TensorFlow บน Google Colab ได้ด้วยตนเอง
- 6 อธิบายข้อจำกัดของ Generative AI และวิธีแก้ปัญหาด้วย RAG

1.1

จากโครงข่ายประสาทเทียมสู่ Deep Learning และ Generative AI

เรื่องราวของ Generative AI ไม่ได้เริ่มต้นเมื่อไม่กี่ปีก่อน แต่ยิ่งรึกมากกว่าครึ่งศตวรรษ จุดเริ่มต้นคือความพยายามจำลองการทำงานของสมองมนุษย์ด้วยคณิตศาสตร์แนวคิด “**เพอร์เซปตรอน**” (Perceptron) ของ Frank Rosenblatt ในปี 1958 เสนอหน่วยคำนวณที่เลียนแบบเซลล์ประสาทหนึ่งเซลล์ ต่อมาในปี 1986 การค้นพบวิธี “**แพร่ย้อนกลับ**” (Backpropagation) ทำให้เราฝึกโครงข่ายที่มีหลายชั้นได้สำเร็จ และในปี 2012 เมื่อพลังการประมวลผลของ GPU เพิ่มขึ้นมาก โครงข่ายขนาดใหญ่ที่เรียกว่า Deep Learning ก็แสดงศักยภาพจนชนะการแข่งขันจำแนกภาพ ImageNet อย่างถล่มทลาย จากนั้นในปี 2017 สถาปัตยกรรม Transformer พร้อมกลไก Attention ได้เปิดประตูสู่ยุคของแบบจำลองภาษาขนาดใหญ่ และนำมาสู่ Generative AI ที่เราใช้กันในปัจจุบัน

วิวัฒนาการจากโครงข่ายประสาทเทียมสู่ Generative AI

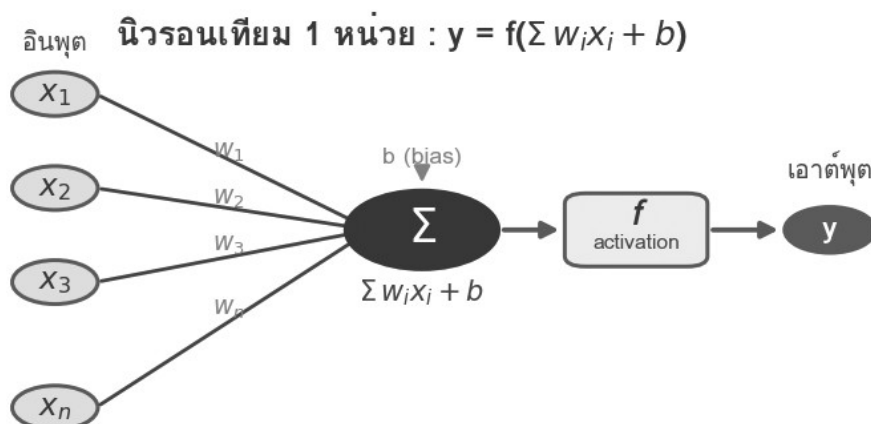


รูปที่ 1.1 เส้นเวลาแสดงวิวัฒนาการจากโครงข่ายประสาทเทียมสู่ Generative AI

สิ่งที่น่าสนใจคือ แต่ละก้าวกระโดดไม่ได้ลบล้างแนวคิดเดิม แต่ต่อยอดจากมัน เพอร์เซปตรอนยังคงเป็นหน่วยพื้นฐานในโครงข่ายสมัยใหม่ การแพร่ย้อนกลับยังเป็นวิธีฝึกหลักจนทุกวันนี้ และ word2vec ที่เราจะกล่าวถึงต่อไปก็เป็นบรรพบุรุษทางความคิดของแบบจำลองภาษาขนาดใหญ่ที่เรียกว่า LLM เช่น ChatGPT การเข้าใจลำดับวิวัฒนาการนี้จึงช่วยให้เราเห็นว่า Generative AI ไม่ใช่ความบังเอิญที่เกิดขึ้นลอยๆ แต่เป็นผลของการสะสมความรู้อย่างเป็นระบบมาหลายทศวรรษ

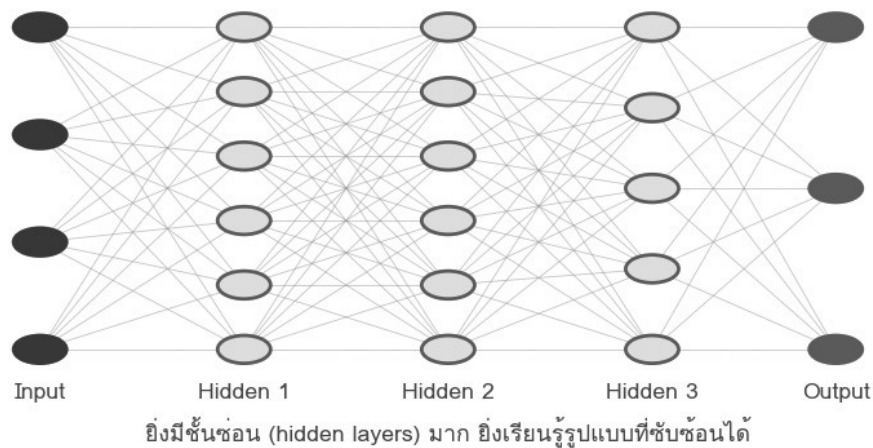
นิวรอนเทียม: หน่วยคิดที่เล็กที่สุด

หัวใจของโครงข่ายประสาทเทียมคือ “**นิวรอนเทียม**” (Artificial Neuron) ซึ่งทำงานง่ายมาก คือรับค่าอินพุตหลายตัว ($x_1, x_2, \dots$) คูณด้วยน้ำหนัก (weight, w) ของแต่ละเส้น แล้วบวกรวมกันพร้อมค่าคงที่ที่เรียกว่า bias (b) จากนั้นส่งผลรวมผ่าน “**ฟังก์ชันกระตุ้น**” (Activation Function) เพื่อให้ได้เอาต์พุต ดังสมการ $y = f(\sum w_i x_i + b)$ การ “**เรียนรู้**” ของโครงข่ายก็คือการค่อยๆ ปรับค่า w และ b ให้คำตอบที่ทำนายใกล้เคียงคำตอบจริงมากที่สุด



รูปที่ 1.2 โครงสร้างนิวรอนเทียมหนึ่งหน่วยและสมการพื้นฐาน

เมื่อนำนิรอนจำนวนมากมาเรียงต่อกันเป็นชั้นๆ และเชื่อมโยงระหว่างชั้น เราจะได้ “**โครงข่ายประสาทเทียมเชิงลึก**” (Deep Neural Network) ยังมีชั้นซ่อน (Hidden Layers) มาก โครงข่ายยังสามารถเรียนรู้รูปแบบที่ซับซ้อนและเป็นนามธรรมได้มากขึ้น เช่น เริ่มจากเส้นและขอบในภาพ ไปสู่รูปทรง และวัตถุทั้งชิ้น



รูปที่ 1.3 โครงข่ายประสาทเทียมเชิงลึกที่มีหลายชั้นซ่อน

ฟังก์ชันกระตุ้น (Activation Function) มีบทบาทสำคัญอย่างยิ่ง เพราะหากไม่มีการซ่อนชั้นหลายๆ ชั้นจะไม่มีประโยชน์ เนื่องจากการบวกและคูณเชิงเส้นหลายชั้นยุบรวมเป็นชั้นเดียวได้ ฟังก์ชันกระตุ้นที่ไม่เป็นเชิงเส้น เช่น sigmoid, tanh และที่นิยมที่สุดในปัจจุบันคือ ReLU (Rectified Linear Unit) ทำให้โครงข่าย “**โค้งงอ**” เพื่อจับความสัมพันธ์ที่ไม่ใช่เส้นตรงได้ ReLU ทำงานง่ายมาก คือถ้าค่าที่รับเข้ามาเป็นลบให้ตอบศูนย์ ถ้าเป็นบวกให้ตอบค่าเดิม ความเรียบง่ายนี้เองที่ช่วยให้ฝึกโครงข่ายลึกๆ ได้เร็วและเสถียร เพราะสามารถจับความสัมพันธ์ที่ไม่ใช่เส้นตรงได้

คำถามที่น่าสนใจคือ เหตุใด Deep Learning จึงเพิ่งมาประสบความสำเร็จอย่างก้าวกระโดดในช่วงทศวรรษ 2010 ทั้งที่แนวคิดมีมานาน คำตอบคือการมาบรรจบกันของสามปัจจัย ปัจจัยแรกคือ “**ข้อมูล**” จำนวนมหาศาลจากยุคอินเทอร์เน็ต ปัจจัยที่สองคือ “**พลังประมวลผล**” โดยเฉพาะหน่วยประมวลผลกราฟิก (GPU) ที่คำนวณเมทริกซ์ขนาดใหญ่ได้เร็ว และปัจจัยที่สามคือ “**อัลกอริทึม**” ที่ดีขึ้น เช่น เทคนิคป้องกัน Overfitting คือป้องกันการอธิบายเฉพาะข้อมูลที่เอามาเทรนได้ดีเกินไปจนขาดความยืดหยุ่นที่จะนำไปทำนายชุดข้อมูลที่ระบบไม่เคยเห็นมาก่อน และฟังก์ชันกระตุ้นที่เหมาะสม เมื่อสามปัจจัยนี้มาพร้อมกัน โครงข่ายเชิงลึกจึงแสดงความสามารถที่ซ่อนอยู่ออกมา

คำว่า “Generative” หรือ “**เชิงสร้าง**” หมายถึงแบบจำลองที่ไม่ได้เพียงจำแนกหรือทำนายตัวเลข แต่สามารถ “**สร้าง**” ข้อมูลใหม่ที่มีลักษณะคล้ายข้อมูลที่เคยเรียนรู้ เช่น สร้างข้อความ รูปภาพ หรือเสียง ส่วน “Foundation Model” คือแบบจำลองขนาดใหญ่ที่ฝึกบนข้อมูลมหาศาลแบบครอบคลุมกว้าง เช่น GPT-4o, Llama, Claude Mythos จนสามารถนำไปปรับใช้กับงานปลายทางได้ ลากหลายโดยไม่ต้องฝึกใหม่ทั้งหมด

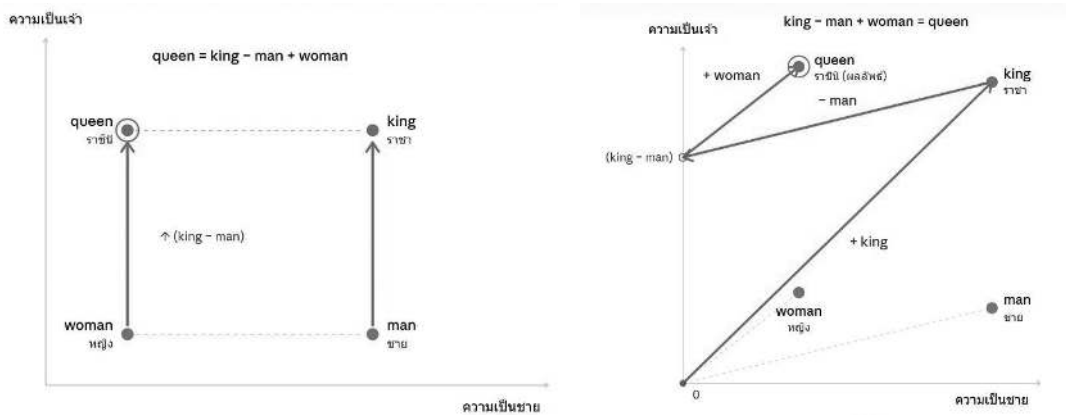
1.2

word2vec: ทำให้คอมพิวเตอร์เข้าใจ ความหมายของคำ

คอมพิวเตอร์ยุคดั้งเดิมเข้าใจตัวเลขแต่ไม่เข้าใจคำ คำถามสำคัญคือ เราจะแปลงคำอย่าง “**ทำไม**” หรือ “**ขาดทุน**” ให้เป็นตัวเลขที่สื่อความหมายได้อย่างไร คำตอบที่สำคัญที่สุดคำตอบหนึ่งมาจากงานของ Mikolov et al (2013) ที่ชื่อ word2vec แนวคิดนี้ตั้งอยู่บน “**สมมติฐานเชิงการกระจาย**” (Distributional Hypothesis) ที่ว่า คำที่ปรากฏในบริบทคล้ายกัน มักมีความหมายใกล้เคียงกัน เช่น คำว่า “**ธนาคาร**” กับ “**สินเชื่**” มักอยู่ในประโยคแวดล้อมคล้ายๆ กัน

word2vec แทนแต่ละคำด้วยเวกเตอร์จำนวนจริง (Real Number) หลายสิบหรือหลายร้อยมิติ คำที่ความหมายใกล้เคียงกันจะมีเวกเตอร์อยู่ใกล้กันในปริภูมิ และที่น่าทึ่งคือความสัมพันธ์เชิงความหมายกลายเป็นการคำนวณเชิงเวกเตอร์ได้ เช่น ทิศทางจากคำเชิงลบไปคำเชิงบวกมีรูปแบบที่สอดคล้องกัน

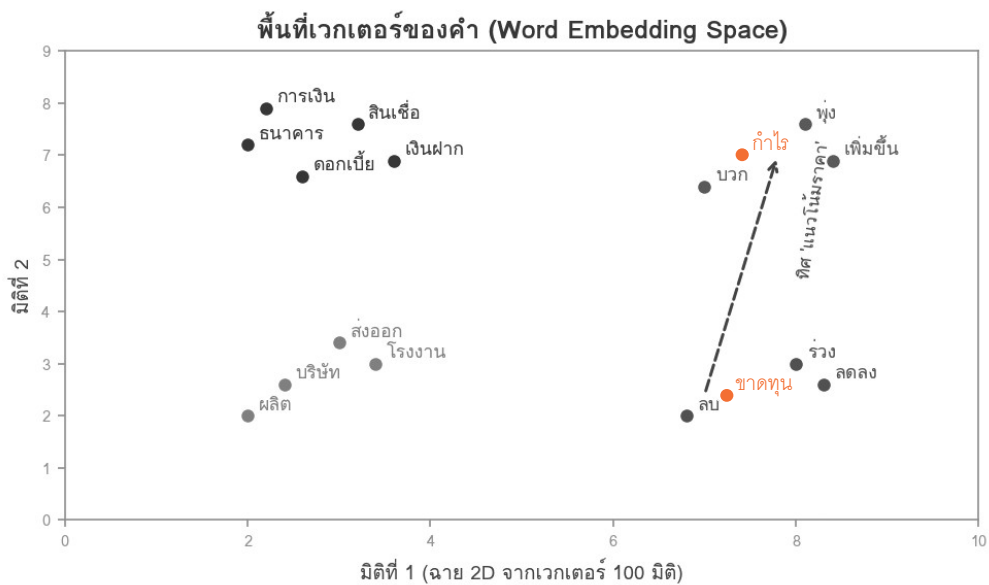
สิ่งที่ทำให้ word2vec กลายเป็นแนวคิดที่ปฏิวัติวงการคือ ความสัมพันธ์เชิงความหมายปรากฏเป็นการคำนวณเชิงเวกเตอร์ที่ “**บวกลบ**” ได้ ตัวอย่างคลาสสิกในภาษาอังกฤษคือ เวกเตอร์ของ king ลบด้วย man แล้วบวก woman จะได้ผลเท่ากับเวกเตอร์ของ queen ซึ่งสามารถเขียนความสัมพันธ์ในเชิงสมการได้ว่า เวกเตอร์ 2 มิติ ที่ใช้แทนคำ 4 คำเหล่านี้ คือ $queen = king - man + woman$ ที่เราสามารถแสดงความสัมพันธ์ของสมการนี้ในรูปแบบที่ 1.4 จะเห็นว่าตัวอย่างของคำที่แสดงด้วย 2 มิตินี้ คือ มิติที่วัด ระดับความเป็นชายและมิติที่วัดระดับความเป็นเจ้า



รูปที่ 1.4 ความสัมพันธ์ $queen = king - man + woman$ ด้วยเวกเตอร์ 2 มิติ

ในบริบทการเงินก็เช่นกัน เราอาจพบว่าทิศทางจากคำว่า “ขาดทุน” ไป “กำไร” มีรูปแบบคล้ายกับทิศทางจาก “ลดลง” ไป “เพิ่มขึ้น” นั่นหมายความว่าแบบจำลองไม่ได้เพียงจำคำ แต่จับ “โครงสร้างความหมาย” ของภาษาได้ในระดับหนึ่ง นอกจากนี้ศัพท์ทางการเงินมีได้มีอยู่เพียง 2 คำนี้ แต่มีมากมายหลายพันคำ ยังมีคำมากเท่าไร จำนวนมิติของเวกเตอร์ที่จะแทนที่คำจะมีมากขึ้น คำถามคือ เราจะทราบได้อย่างไรว่าในบริบทหนึ่งที่เราสนใจ ควรใช้เวกเตอร์กี่มิติ และแต่ละมิติสะท้อนอะไร และเมื่อรู้แล้ว เราจะต้องใช้ค่าเท่าไร แทนในมิติเหล่านั้น

รูปที่ 1.5 แสดงตัวอย่างการฉายเวกเตอร์ขนาด 100 มิติ บนฉากสองมิติเพื่อให้เห็นการเกาะกลุ่มของคำในโดเมนการเงิน

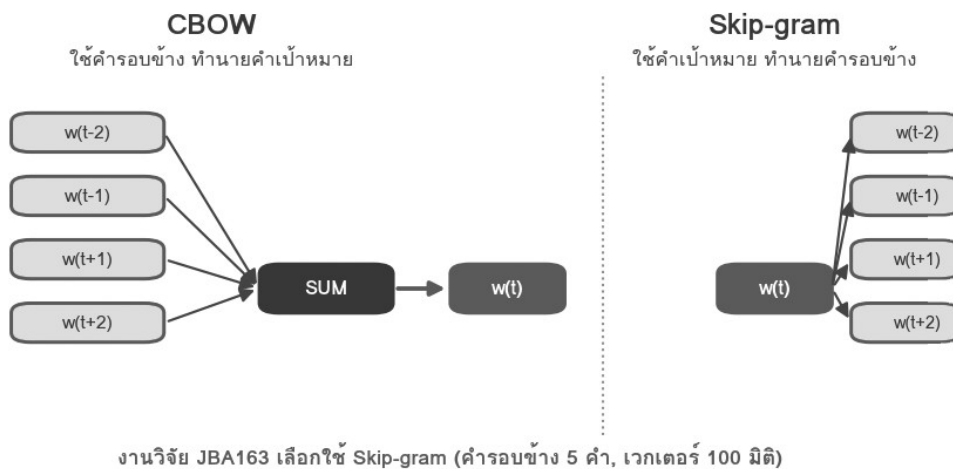


รูปที่ 1.5 ภาพฉาย 2 มิติของพื้นที่เวกเตอร์ของคำ คำความหมายใกล้เคียงกันเกาะกลุ่มกัน

คำตอบว่าจำนวนมิติควรมีเท่าไร ใช้อะไรวัดมิติ ตลอดจนค่าของแต่ละมิติควรเป็นเท่าไร ได้จากการ “ฝึกทำนาย” บนข้อความจำนวนมาก เริ่มจากสุ่มค่าเวกเตอร์ให้ทุกคำ แล้วให้แบบจำลองพยายามทำนายคำจากบริบท เมื่อทำนายผิด ก็ปรับค่าเวกเตอร์ทีละน้อย ด้วยหลักการเดียวกับ Backpropagation ทำซ้ำนับล้านครั้งจนเวกเตอร์ของคำที่ใช้ในบริบทคล้ายกันค่อยๆ เลื่อนเข้ามาใกล้กัน จำนวนมิติของเวกเตอร์ (เช่น 100 มิติในกรณีศึกษาที่จะกล่าวถึง) เป็นตัวกำหนดว่าแบบจำลองมี “พื้นที่” ให้บรรยายความหมายของคำได้ละเอียดเพียงใด

วิธีฝึกเวกเตอร์ของคำด้วยเทคนิค CBOW และ Skip-gram

word2vec ฝึกเวกเตอร์ได้สองแบบ แบบแรกคือ CBOW (Continuous Bag of Words) ใช้คำรอบข้างเพื่อทำนายคำตรงกลาง ส่วนแบบที่สองคือ Skip-gram ทำกลับกัน คือใช้คำตรงกลางเพื่อทำนายคำรอบข้าง โดยทั่วไป Skip-gram ทำงานได้ดีกับคำที่พบบ่อยซึ่งเหมาะกับข้อมูลข่าวการเงินที่มีศัพท์เฉพาะจำนวนมาก



รูปที่ 1.6 เปรียบเทียบสถาปัตยกรรม CBOW และ Skip-gram ในการสร้าง Word Vector

กิจกรรม 1.1

ถ้าเราป้อนพาดหัวข่าว “บริษัทประกาศกำไรสุทธิเพิ่มขึ้น” ให้ระบบที่ฝึกด้วย word2vec เวกเตอร์ของคำว่า “กำไร” ควรอยู่ใกล้คำใดมากกว่า ระหว่าง “เติบโต” กับ “ล้มละลาย” เพราะเหตุใด ลองอธิบายโดยใช้แนวคิดสมมติฐานเชิงการกระจาย

กรณีศึกษา: การพยากรณ์ราคาหุ้นไทยด้วย word2vec และ Deep Learning

เพื่อให้เห็นภาพการประยุกต์จริงในบริษัทไทย เราจะศึกษางานวิจัยของ Leemakdej (2019) เรื่อง “การพยากรณ์ราคาหุ้นด้วยวิธีดีพเลิร์นนิง” (Stock Price Prediction with Deep Learning) งานวิจัยนี้จะช่วยให้ผู้เรียนเห็นภาพความเชื่อมโยงแนวคิดทั้งหมดในหัวข้อ 1.1 และ 1.2 และตอบคำถามสำคัญทางการเงินว่าด้วยเรื่อง “ตลาดหุ้นมีประสิทธิภาพจริงหรือไม่” หรือ Efficient Market Hypothesis ด้วย

1.3.1 คำถามวิจัยและสมมติฐานตลาดมีประสิทธิภาพ (Efficient Market Hypothesis: EMH)

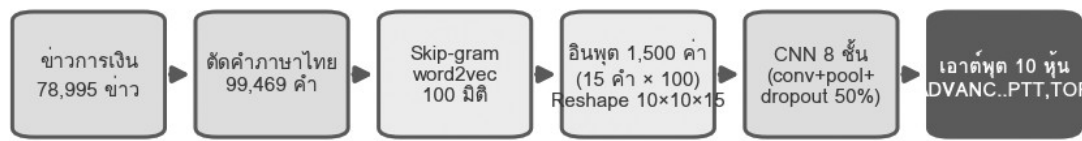
ทฤษฎี EMH ที่ Fama เสนอในปี 1970 แบ่งระดับประสิทธิภาพของข้อมูลออกเป็นสามระดับ ระดับแรกคือ “ระดับอ่อน” (Weak Form) ที่ราคาในอดีตไม่สามารถใช้ทำนายราคาในอนาคตได้ ซึ่งปฏิเสธการวิเคราะห์เชิงเทคนิค (Technical Analysis) เพื่อทำนายราคาหุ้นในอนาคต ระดับที่สองคือ “ระดับกึ่งแกร่ง” (Semi-strong Form) ที่เมื่อข้อมูลกลายเป็นสาธารณะ ราคาจะปรับตัวสะท้อนทันที นักลงทุนจึงทำกำไรผิดปกติจากข่าวสาธารณะไม่ได้ และระดับที่สามคือ “ระดับแกร่ง” (Strong Form) ที่แม้แต่ข้อมูลภายในก็ทำกำไรไม่ได้ งานวิจัยนี้มุ่งทดสอบระดับกึ่งแกร่งเป็นหลัก เพราะวิเคราะห์ว่านักลงทุนสามารถทำกำไรจากข่าวที่เปิดเผยเป็นสาธารณะแล้วได้หรือไม่

งานวิจัยนี้แตกต่างจากงานวิจัยทดสอบ EMH อื่น เพราะการทดสอบระดับกึ่งแกร่งในอดีตมักใช้ราคาปิดรายวัน ซึ่งอาจ “หายาบ” เกินกว่าจะจับโอกาสที่เกิดและหายไปภายในไม่กี่นาที ในความเป็นจริง นักเก็งกำไรในตลาดหุ้น (Speculator) จำนวนมากไม่ได้ถือสถานะข้ามวัน แต่ซื้อขายในช่วงสั้นๆ หลังข่าวออก งานวิจัยจึงออกแบบการทดสอบโดยใช้ข้อมูลระหว่างวัน (Intraday) และวัดผลตอบแทนในกรอบเวลา 5, 10 และ 30 นาที รวมถึงการถือทั้งวัน เพื่อดูว่าโอกาสทำกำไรผิดปกตินี้ยังมีอยู่ในกรอบเวลาใดบ้าง

อีกประเด็นที่น่าสนใจจากข้อมูลคือ ข่าวการเงินส่วนใหญ่มักถูกประกาศนอกเวลาทำการซื้อขาย สะท้อนว่าบริษัทต้องการลดผลกระทบต่อความผันผวนของราคา ทำให้เมื่อตลาดเปิด ราคามักปรับตัวรับข่าวอย่างรวดเร็ว นี่คือเหตุผลที่กรอบเวลาสั้นๆ หลังเปิดตลาด หรือหลังการประกาศข่าวจึงเป็นช่วงที่น่าศึกษาเป็นพิเศษ

1.3.2 ขั้นตอนและสถาปัตยกรรมของแบบจำลอง

ผู้วิจัยรวบรวมข่าวการเงินจำนวน 78,995 ข่าว ในช่วงมีนาคมถึงสิงหาคม 2016 พบคำทั้งหมด 99,469 คำ จากนั้นตัดคำภาษาไทยและแปลงเป็นเวกเตอร์ด้วย Skip-gram โดยใช้คำรอบข้าง 5 คำ และกำหนดให้แต่ละคำมีเวกเตอร์ขนาด 100 มิติ เพื่อจำกัดความซับซ้อน ผู้วิจัยใช้เฉพาะ “พาดหัวข่าว” และเลือกคำ 15 คำแรก ทำให้ได้อินพุตขนาด 1,500 คำ (15 คำ × 100 มิติ) แล้วจัดรูปใหม่ให้เป็น Tensor คล้ายภาพขนาด 10×10×15 ก่อนป้อนเข้าโครงข่ายแบบ Convolutional Neural Network



ฝึกข้อมูล ม.ค.-มิ.ย. 2016 | ทดสอบนอกตัวอย่าง ก.ค. 2016 | loss = MSE, optimizer = Adam

รูปที่ 1.7 กระบวนการทำนายราคาหุ้นด้วย word2vec + Deep Learning
ที่มา: เรียบเรียงจาก Leemakdej (2019)

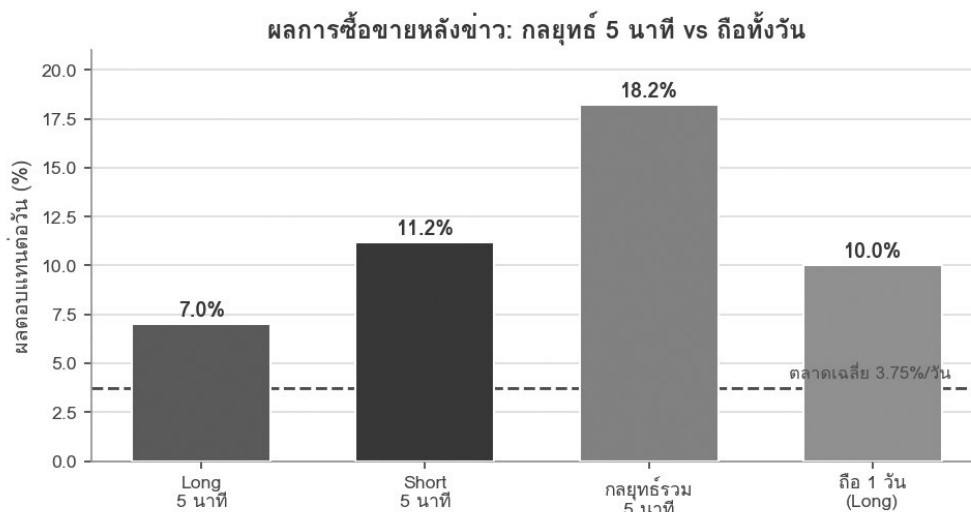
เหตุใดจึงจัดข้อมูลข่าวให้เป็นเทนเซอร์คล้ายภาพ คำตอบคือเพื่อใช้ประโยชน์จากโครงข่ายแบบ Convolutional Neural Network (CNN) ซึ่งเดิมออกแบบมาเพื่อใช้อธิบายรูปภาพ โดยแบ่งพื้นที่ภาพใหญ่เป็นกรอบย่อยเพื่อหาลักษณะเฉพาะ เช่นการทอดนให้เข้าใจการแยกประเภทสัตว์อาจแบ่งกรอบย่อยในภาพใหญ่ให้แสดงส่วนหู ตา จมูก ปาก เท้า หาง แปลงเป็นตัวเลขที่สามารถคำนวณได้ที่ละส่วน เมื่อนำตัวเลขที่ดีความจากกรอบย่อยมารวมกันแล้ววิเคราะห์ต่อด้วยโมเดลก็จะสามารถระบุได้ว่าภาพนี้เป็นสัตว์อะไร ผู้วิจัยอาศัยแนวคิดเดียวกันนี้ให้ฟิลเตอร์ของ CNN จับ “รูปแบบของกลุ่มคำ” ที่อยู่ใกล้กันในเวกเตอร์ข่าว การเลือกใช้เฉพาะพาดหัวข่าว 15 คำแรกเพื่อจำกัดความซับซ้อน เพราะพาดหัวมักสรุปสาระสำคัญของข่าวไว้แล้ว และความยาวพาดหัวส่วนใหญ่อยู่ที่ราว 12 ถึง 15 คำ

1.3.3 ผลการศึกษาและนัยทางการเงิน

ตารางที่ 1.1 สรุปผลตอบแทนของกลยุทธ์การซื้อขายหลังข่าวในช่วงเวลาต่างๆ ผลที่โดดเด่นคือ กลยุทธ์ซื้อขายภายใน 5 นาทีหลังข่าวให้ผลตอบแทนรวมเฉลี่ยสูงถึงประมาณ 18.2% ต่อวัน (รวมทั้งฝั่งซื้อและฝั่งขายชอร์ต) เทียบกับผลตอบแทนเฉลี่ยของตลาดในเดือนนั้นที่ราว 3.75% ต่อวัน ซึ่งบ่งชี้ว่าตลาดอาจไม่ได้มีประสิทธิภาพระดับกึ่งแกร่งอย่างสมบูรณ์ในกรอบเวลานั้นๆ

ตารางที่ 1.1 สรุปผลตอบแทนของกลยุทธ์ซื้อขายหลังข่าวถูกประกาศเป็นข้อมูลสาธารณะ

กลยุทธ์	ค่าเฉลี่ยต่อรายการ	ผลตอบแทนต่อวัน (โดยประมาณ)
ซื้อ (Long) ภายใน 5 นาที	0.131%	~7.0%
ขายชอร์ต (Short) ภายใน 5 นาที	0.209%	~11.2%
รวมสองกลยุทธ์ 5 นาที	-	~18.2%
ถือทั้งวัน (ฝั่งซื้อ)	10.17%	~10.0%
ผลตอบแทนเฉลี่ยตลาด (ก.ค. 2016)	-	~3.75%



รูปที่ 1.8 เปรียบเทียบผลตอบแทนต่อวันของกลยุทธ์ต่างๆ กับค่าเฉลี่ยตลาด

น่าสังเกตว่าผลตอบแทนส่วนใหญ่ของกลยุทธ์ 5 นาทีมาจากฝั่ง **“ขายชอร์ต”** ทั้งที่ตลาดในเดือนกรกฎาคม 2016 เป็นขาขึ้นชัดเจน ผู้วิจัยตีความว่าแบบจำลองสามารถมองเห็นแนวโน้มราคาบนความผันผวนระยะสั้น และจับจังหวะการปรับฐานของราคาเพื่อทำกำไรได้ ซึ่งเป็นการแสดงว่าข้อมูลข่าวสารสาธารณะยังมีพลังพยากรณ์ในกรอบเวลาด้านมาก ชัดกับสมมติฐานตลาดมีประสิทธิภาพระดับกึ่งแกร่ง

ในเชิงระเบียบวิธี งานวิจัยแนวนี้นแบ่งออกเป็นสองสำนักหลัก สำนักแรกคือแนวทาง **“อารมณ์ตลาด”** (Sentiment) ที่นับความถี่ของคำเชิงบวกและเชิงลบเพื่อคำนวณคะแนนอารมณ์ ส่วนสำนักที่สองซึ่งงานวิจัยนี้ใช้ คือแนวทางใช้โครงข่ายประสาทเทียมแปลงข้อความไม่มีโครงสร้างเป็นเวกเตอร์ แล้วให้โครงข่ายเรียนรู้ความสัมพันธ์ระหว่างเวกเตอร์ข่าวกับราคาในอนาคต ข้อได้เปรียบของแนวทางหลังคือไม่ต้องกำหนดล่วงหน้าว่าคำใดบวกหรือลบ แต่ให้แบบจำลองค้นพบรูปแบบเอง

งานวิจัยยังเสนอแนวทางพัฒนาต่อไปหลายประการ เช่น การเพิ่มจำนวนชั้นเพื่อเพิ่มความแม่นยำ การขยายให้ครอบคลุมหุ้นมากขึ้น และการ “ผสม” ข้อมูลข่าวเข้ากับการวิเคราะห์ปัจจัยพื้นฐานและทางเทคนิค ประเด็นเหล่านี้สะท้อนว่า แม้ผลลัพธ์จะชี้ว่าแบบจำลองสามารถทำนายราคาหุ้นได้ แต่การนำไปใช้จริงยังต้องการการพัฒนาและการตรวจสอบอีกมาก ซึ่งเป็นบทเรียนสำคัญสำหรับผู้เรียนที่จะลงมือสร้างแบบจำลองของตนเองในหัวข้อ 1.7

ข้อพึงระวังในการตีความ

ผลตอบแทนที่สูงในงานวิจัยเป็นค่าก่อนหักต้นทุนการซื้อขาย ค่าธรรมเนียมสภาพคล่อง และ Slippage คือเมื่อทำการซื้อหรือขายจริงอาจไม่ได้ในราคาที่วางแผนไว้ เพราะความลึกของตลาดซึ่งวัดผ่าน Limit Order ที่รอการจับคู่ซื้อขายยังมีไม่มากพอ อีกทั้งทดสอบในกรอบเวลาสั้นและตลาดขาขึ้น การนำไปใช้จริงต้องตรวจสอบความทนทานในหลายสภาวะตลาด และพึงระลึกว่าผลในอดีตไม่รับประกันผลในอนาคต นี่คือเหตุผลที่ตลอดหนังสือเล่มนี้ เราจะเน้นการตรวจสอบซ้ำและการมีมนุษย์กำกับเสมอ

กิจกรรม 1.2

ลองสรุปด้วยคำพูดของตนเองว่า งานวิจัยนี้ใช้แนวคิด word2vec (หัวข้อ 1.2) และโครงข่ายเชิงลึก (หัวข้อ 1.1) ตรงจุดใดบ้าง แล้วเขียนผังกระบวนการ 5 ขั้นตอนจากข่าวดิบจนถึงการจับสัญญาณซื้อขาย

1.4

จาก word2vec สู่ Large Language Models (LLM)

word2vec เป็นก้าวสำคัญของการพัฒนาคอมพิวเตอร์ให้สามารถประมวลผลภาษามนุษย์ แต่มีข้อจำกัดใหญ่ คือให้เวกเตอร์ “คงที่” หนึ่งเวกเตอร์ต่อหนึ่งคำเสมอ ไม่ว่าคำนั้น

จะปรากฏในบริบทใด เช่น คำว่า “กรุงเทพ” ในประโยค “เปิดบัญชีธนาคารกรุงเทพ” กับ “เดินทางไปกรุงเทพ...” ควรมีความหมายต่างกัน แต่ word2vec ให้เวกเตอร์เดียวกัน แบบจำลองภาษาสมัยใหม่แก้ปัญหา นี้ด้วยการสร้างเวกเตอร์ที่ **“ขึ้นกับบริบท”** (Contextual Embedding)



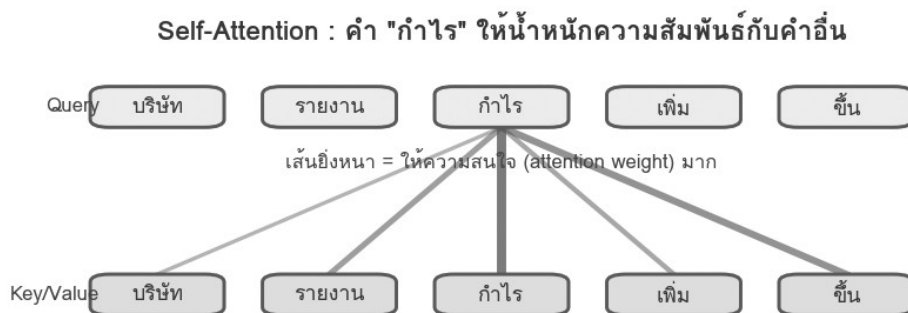
รูปที่ 1.9 ความต่างระหว่างเวกเตอร์คงที่ของ word2vec กับเวกเตอร์ตามบริบทของ LLM

1.4.1 กลไก Attention: หัวใจของ Transformer

ก่อนยุค Transformer แบบจำลองลำดับอย่าง RNN และ LSTM ประมวลผลข้อความทีละคำตามลำดับ ทำให้ฝึกช้าและจำบริบทไกลๆ ได้ยาก ปี 2017 สถาปัตยกรรม Transformer เสนอกลไก **“Self-attention”** ที่ให้ทุกคำในประโยคมองเห็นและ **“ให้น้ำหนักความสนใจ”** แก่คำอื่น ๆ ได้พร้อมกันทั้งประโยค ทำให้แบบจำลองเข้าใจว่าคำใดสัมพันธ์กับคำใด และประมวลผลแบบขนานได้รวดเร็ว

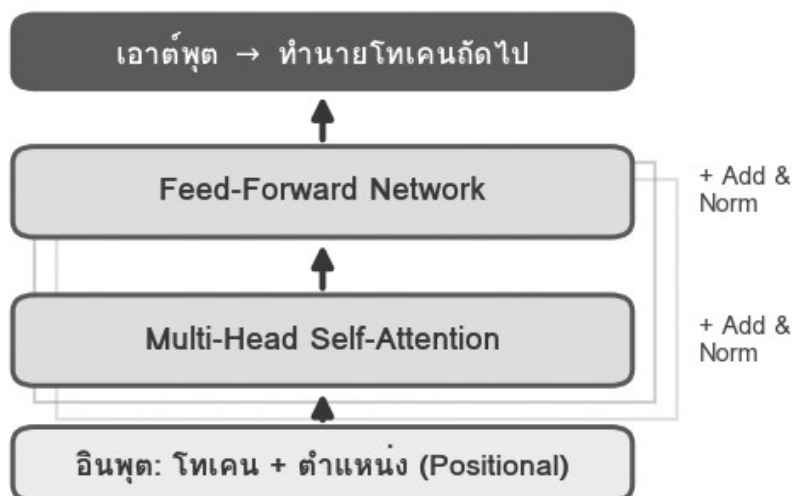
กลไก Self-attention ทำงานโดยให้แต่ละคำสร้างเวกเตอร์สามชนิด ได้แก่ **“คำถาม”** (Query) ที่บอกว่าคำนี้กำลังมองหาอะไร **“กุญแจ”** (Key) ที่บอกว่าคำนี้มีอะไรให้ และ **“ค่า”** (Value) ที่เป็นเนื้อหาจริง แบบจำลองจับคู่ Query กับ Key ของทุกคำเพื่อคำนวณน้ำหนักความสนใจ แล้วนำน้ำหนักนั้นไปถ่วงรวม Value ผลคือคำแต่ละคำได้ **“จุดจับ”** บริบทจากคำที่เกี่ยวข้องมากที่สุด เช่น เมื่อใช้คำสรรพนามว่า **“มัน”** ในประโยคหนึ่ง คอมพิวเตอร์จะรู้ว่าอ้างถึงคำนามใดในประโยคก่อนหน้า

เนื่องจากกลไกนี้ไม่ได้รับรู้ลำดับของคำโดยอัตโนมัติ Transformer จึงเติม **“การเข้ารหัสตำแหน่ง”** (Positional Encoding) เพื่อบอกแบบจำลองว่าคำใดมาก่อนมาหลัง และยังใช้ **“หลายหัว”** (Multi-head) คือทำ Attention พร้อมกันหลายชุด เพื่อให้แต่ละหัวจับความสัมพันธ์คนละแบบ เช่น หัวหนึ่งจับความสัมพันธ์ทางไวยากรณ์ อีกหัวจับความสัมพันธ์เชิงความหมาย



รูปที่ 1.10 กลไก Self-attention: คำหนึ่งให้น้ำหนักความสัมพันธ์กับคำอื่นในประโยค

Transformer หนึ่งบล็อกประกอบด้วยชั้น Multi-head Self-attention และชั้น Feed-forward พร้อมกลไกช่วยฝึก เช่น Residual Connection และ Layer Normalization เมื่อนำบล็อกเหล่านี้มาซ้อนกันหลายสิบชั้นและฝึกบนข้อความมหาศาล เราจะได้แบบจำลองภาษาขนาดใหญ่ที่เรียกว่า LLM (Large Language Model)



รูปที่ 1.11 โครงสร้างบล็อกหลักของ Transformer ที่ซ้อนกันหลายชั้นใน LLM

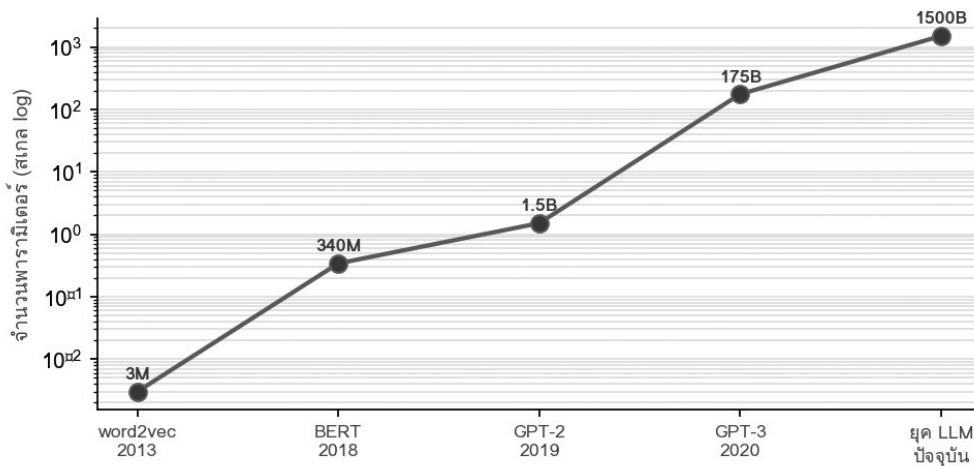
1.4.2 การฝึกล่วงหน้า การปรับแต่ง และการขยายขนาด

LLM แบ่งการฝึกเป็นสองช่วงหลัก ช่วงแรกคือ **“การฝึกล่วงหน้า”** (Pre-training) ให้แบบจำลองทำนายคำถัดไปบนข้อความจำนวนมากจนเข้าใจไวยากรณ์ ข้อเท็จจริง และรูปแบบการให้เหตุผลในระดับหนึ่ง ช่วงที่สองคือ **“การปรับแต่ง”** (Fine-tuning) และการปรับด้วยผลป้อนกลับจากมนุษย์ เพื่อให้แบบจำลองทำตามคำสั่งและปลอดภัยขึ้น

น่าสังเกตว่าการฝึกล่วงหน้าใช้หลักการเดียวกับ word2vec อย่างน่าทึ่ง คือ **“การทำนายคำจากบริบท”** เพียงแต่ LLM ทำในสเกลที่ใหญ่กว่ามาก และทำนายทั้งประโยค ต่อเนื่องแทนที่จะเป็นคำเดียว ความต่อเนื่องของแนวคิดนี้ ตั้งแต่ word2vec จนถึง LLM แสดงให้เห็นว่าความก้าวหน้าทางเทคโนโลยีมักเป็นการต่อยอดแนวคิดพื้นฐานที่ดี ไม่ใช่ การกระโดดข้ามแบบไร้ที่มา

ส่วนการปรับด้วยผลป้อนกลับจากมนุษย์ (มักเรียกย่อว่า RLHF) เป็นขั้นตอนที่ทำให้แบบจำลองจาก **“เครื่องเต็มประโยค”** กลายเป็น **“ผู้ช่วยที่ทำตามคำสั่ง”** ได้จริง โดยให้มนุษย์ช่วยจัดอันดับคำตอบที่ดีกว่า แล้วใช้สัญญาณนั้นปรับพฤติกรรมของแบบจำลอง ขั้นตอนนี้สำคัญต่อความปลอดภัยและความน่าเชื่อถือ ซึ่งเป็นคุณสมบัติที่งานการเงินให้ความสำคัญสูง

สิ่งที่ทำให้ LLM ทำงานอย่างน่าทึ่งคือการ **“ขยายขนาด”** ทั้งจำนวนพารามิเตอร์ ปริมาณข้อมูล และพลังประมวลผล เมื่อขนาดโตถึงระดับหนึ่ง แบบจำลองแสดงความสามารถใหม่ๆ ที่ไม่ได้ถูกออกแบบไว้โดยตรง เช่น การสรุปความ การแปล และการตอบคำถามเชิงเหตุผล รูปที่ 1.12 แสดงการเติบโตของขนาดโมเดลตั้งแต่ word2vec จนถึงยุค LLM ปัจจุบัน



รูปที่ 1.12 การเติบโตของจำนวนพารามิเตอร์ของแบบจำลองภาษา (สเกล log)

ปรากฏการณ์ **“ความสามารถที่ผุดขึ้น”** (Emergent Abilities) นี้เป็นเหตุผลหนึ่งที่ทำให้ LLM กลายเป็น **“Foundation Model”** อย่างแท้จริง คือเป็นฐานรากที่นำไปปรับใช้กับงานปลายทางจำนวนมากได้โดยไม่ต้องฝึกใหม่ตั้งแต่ต้น ในทางปฏิบัติ องค์กรการเงินจึงไม่จำเป็นต้องสร้างแบบจำลองขนาดใหญ่เอง แต่สามารถนำ Foundation Model ที่มีอยู่มาปรับแต่งด้วยข้อมูลและบริบทของตน ซึ่งประหยัดทรัพยากรมหาศาล

อย่างไรก็ตาม การขยายขนาดมีต้นทุนและข้อจำกัด แบบจำลองยิ่งใหญ่อิ่งใช้พลังประมวลผลและพลังงานมาก อีกทั้งยังมี **“หน้าต่างบริบท”** (Context Window) จำกัด คือ รับข้อความเข้าได้เพียงจำนวนโทเคนหนึ่ง ซึ่งเป็นข้อจำกัดสำคัญเมื่อต้องวิเคราะห์เอกสารยาวๆ และเป็นเหตุผลหนึ่งที่เราต้องใช้เทคนิคอย่าง RAG

เชื่อมโยงสู่การเงิน

ความสามารถในการเข้าใจบริบทของ LLM ทำให้มันอ่านรายงานการเงินที่ยาวและซับซ้อนได้ เช่น แยกแยะว่า **“ความเสี่ยง”** ในย่อหน้าหนึ่งหมายถึงความเสี่ยงด้านเครดิตหรือด้านตลาด ซึ่งเป็นสิ่งที่ word2vec แบบคงที่ทำได้ เราจะนำความสามารถนี้ไปใช้จริงในบทที่ 9 เรื่องการวิเคราะห์งบการเงินและ MD&A

1.5

หลักการทำงานของ LLM ในระดับแนวคิด

แม้ภายในจะซับซ้อน แต่หลักการทำงานของ LLM ในระดับแนวคิดเข้าใจได้ไม่ยาก เริ่มจากการ **“แบ่งโทเคน”** (Tokenization) คือซอยข้อความเป็นชิ้นย่อยๆ ที่เรียกว่าโทเคน แล้วแทนแต่ละโทเคนด้วยรหัสตัวเลขและเวกเตอร์ตามบริบท

ข้อความ: "ผลประกอบการไตรมาสได้โต"



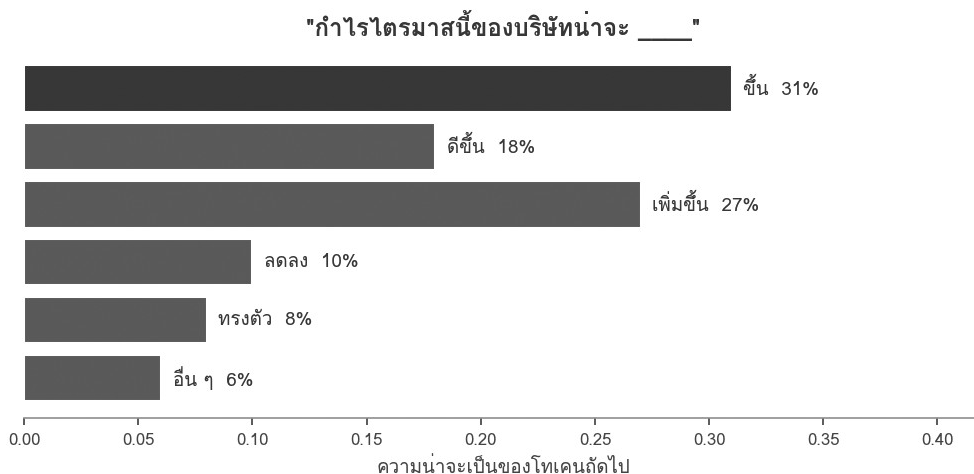
ผล 8120	ประกอบ 4471	การ 203	ไตรมาส 9982	ได้ 55	โต 7310
------------	----------------	------------	----------------	-----------	------------

แต่ละโทเคนถูกแทนด้วยรหัสตัวเลข (token id) แล้วแปลงเป็นเวกเตอร์

รูปที่ 1.13 ตัวอย่างการแบ่งโทเคนของข้อความภาษาไทยและการแทนด้วยรหัสตัวเลข

เหตุที่ต้องแบ่งเป็นโทเคนแทนที่จะใช้คำเต็มหรือตัวอักษรเดี่ยว ก็เพื่อสร้างสมดุลระหว่างขนาดของคลังคำกับความสามารถในการจัดการคำที่ไม่เคยพบ โทเคนบางตัวอาจเป็นคำเต็ม บางตัวเป็นเพียงส่วนของคำ การออกแบบเช่นนี้ทำให้แบบจำลองประกอบคำใหม่ๆ หรือคำเฉพาะทางการเงินที่ไม่เคยเห็นได้ จากการต่อโทเคนย่อยเข้าด้วยกัน

จากนั้นเวกเตอร์เหล่านี้จะไหลผ่านชั้น Transformer หลายชั้น และในขั้นสุดท้าย แบบจำลองจะคำนวณ “ความน่าจะเป็นของโทเคนถัดไป” จากคลังคำทั้งหมด แล้วเลือกโทเคนถัดไปจากการแจกแจงนั้น เมื่อทำซ้ำทีละโทเคน ข้อความที่สมบูรณ์ก็ค่อยๆ ถูกสร้างขึ้น พารามิเตอร์อย่าง Temperature และ Top-p ใน Foundation Model ที่อนุญาตให้ผู้ใช้งานปรับแต่งได้ว่าจะเลือกแบบ “ปลอดภัย” ตามความน่าจะเป็นสูงสุด หรือเปิดให้แบบจำลองคอมพิวเตอร์สร้างสรรค์ความหลากหลายมากขึ้น



รูปที่ 1.14 การทำนายโทเคนถัดไปจากการแจกแจงความน่าจะเป็น

พารามิเตอร์การถอดรหัสนี้มีนัยเชิงปฏิบัติโดยตรง สำหรับงานการเงินที่ต้องการความถูกต้องและสม่ำเสมอ เช่น การสกัดตัวเลขจากงบการเงิน เรายังตั้ง Temperature ต่ำเพื่อให้คำตอบเสถียรและทำซ้ำได้ ส่วนงานที่ต้องการความหลากหลาย เช่น การระดมความคิดเชิงกลยุทธ์ อาจตั้ง Temperature สูงขึ้น การเข้าใจกลไกนี้ช่วยให้ผู้ใช้ปรับแบบจำลองให้เหมาะสมกับงานได้อย่างมีประสิทธิภาพ ไม่ใช่การลองผิดลองถูกอย่างไร้ทิศทาง

อนึ่ง การแบ่งโทเคนในภาษาไทยมีความท้าทายเป็นพิเศษ เพราะภาษาไทยเขียนติดกัน โดยไม่มีการเว้นวรรคระหว่างคำ แบบจำลองจึงต้องเรียนรู้ขอบเขตของคำจากข้อมูล ซึ่งบางครั้งอาจตัดคำผิด ส่งผลต่อความเข้าใจ นี่เป็นอีกเหตุผลที่ผู้ใช้งานการเงินในไทยควรตรวจสอบผลลัพธ์อย่างรอบคอบ โดยเฉพาะกับศัพท์เฉพาะทางหรือชื่อเฉพาะ

ความเข้าใจนี้สำคัญมากต่อผู้ทำงานการเงิน เพราะอธิบายทั้ง “ประโยชน์” และ “ข้อจำกัด” ของ LLM ได้พร้อมกัน ประโยชน์คือมันสร้างข้อความที่ลื่นไหลและเข้ากับบริบทได้ดี แต่ข้อจำกัดคือมันทำนายจาก “รูปแบบทางภาษา” ไม่ใช่จากการเปิดดูข้อเท็จจริง จึงอาจสร้างตัวเลขหรือข้ออ้างที่ฟังดูน่าเชื่อถือแต่ผิดได้

การประยุกต์ Generative AI ในงานการเงิน

คำถามที่ผู้เรียนควรถามเสมอเมื่อพิจารณานำ AI มาใช้คือ **“งานนี้เหมาะกับ AI หรือไม่”** โดยทั่วไป Generative AI เหมาะกับงานที่เกี่ยวข้องกับภาษาและการประมวลผลข้อความปริมาณมาก งานที่มีรูปแบบซ้ำๆ และงานที่ผลลัพธ์เป็นการร่างหรือคัดกรองเบื้องต้น ซึ่งมนุษย์จะตรวจสอบต่อ ในทางกลับกัน งานที่ต้องการความถูกต้องสมบูรณ์ของตัวเลข ความรับผิดชอบกฎหมาย หรือการตัดสินใจที่มีผลกระทบสูง ควรให้ AI เป็นเพียงผู้ช่วย ไม่ใช่ผู้ตัดสินใจ

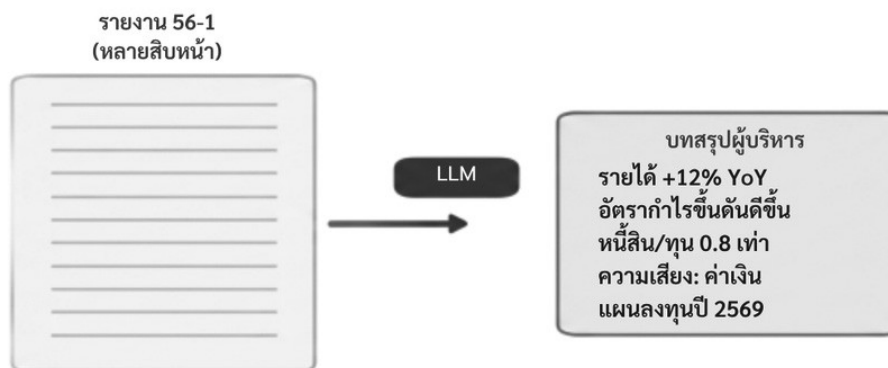
เมื่อเข้าใจหลักการแล้ว มาดูว่าจะนำ Generative AI ไปใช้ในงานการเงินได้อย่างไรบ้าง รูปที่ 1.15 รวบรวมตัวอย่างการใช้งานที่พบบ่อย ตั้งแต่งานวิเคราะห์เอกสาร งานบริการลูกค้า ไปจนถึงงานเขียนโค้ดช่วยวิเคราะห์



รูปที่ 1.15 ตัวอย่างการประยุกต์ Generative AI ในงานการเงิน

1.6.1 การย่อและวิเคราะห์เอกสารการเงิน

รายงานประจำปี แบบ 56-1 One Report และคำอธิบายและการวิเคราะห์ของฝ่ายจัดการ (MD&A) มักยาวหลายสิบหน้า LLM ช่วยย่อให้เหลือประเด็นสำคัญ เช่น แนวโน้มรายได้ อัตรากำไร โครงสร้างหนี้สิน และปัจจัยเสี่ยง ทำให้ผู้วิเคราะห์ประหยัดเวลาในการคัดกรองเบื้องต้น อย่างไรก็ตาม ทุกตัวเลขที่ได้ต้องตรวจสอบย้อนกลับกับเอกสารต้นฉบับเสมอ



รูปที่ 1.16 การใช้ LLM ย่อรายงานยาวให้เหลือประเด็นสำคัญ โดยต้องตรวจสอบกับต้นฉบับ

ลองพิจารณาตัวอย่างที่เป็นรูปธรรม สมมติว่านักวิเคราะห์ต้องอ่านรายงานประจำปีของบริษัทจดทะเบียน 30 แห่งภายในสัปดาห์เดียว แทนที่จะอ่านทุกหน้า นักวิเคราะห์อาจให้ LLM ช่วยสกัดประเด็นมาตรฐานจากแต่ละรายงาน เช่น การเปลี่ยนแปลงของรายได้และกำไร ปัจจัยเสี่ยงที่ฝ่ายจัดการระบุ และทิศทางการลงทุน แล้วจึงใช้เวลาที่ประหยัดได้ไปเจาะลึกเฉพาะบริษัทที่น่าสนใจ นี่คือการนำ AI เพื่อ “ขยายศักยภาพ” ของผู้เชี่ยวชาญ ไม่ใช่แทนที่การตัดสินใจของมนุษย์

1.6.2 การวิเคราะห์ความรู้สึกและสัญญาณจากข่าว

การวิเคราะห์ความรู้สึก (Sentiment Analysis) ของข่าวและโซเชียลมีเดียเป็นการประยุกต์ที่เชื่อมโยงโดยตรงกับกรณีศึกษาในหัวข้อ 1.3 LLM สามารถอ่านพาดหัวข่าวหรือบทวิเคราะห์แล้วประเมินว่าโทนโดยรวมเป็นบวก กลาง หรือลบต่อหลักทรัพย์ใดหลักทรัพย์หนึ่ง ข้อได้เปรียบของ LLM เหนือวิธีนับคำแบบเดิมคือเข้าใจบริบทและการประชดประชันที่อาจปรากฏในเนื้อหาได้ดีกว่า แต่ก็ยังต้องระวังการตีความผิดในบริบทเฉพาะของตลาดประเทศไทย

1.6.3 ผู้ช่วยบริการลูกค้าและการร่างเอกสาร

ในงานบริการ ธนาคารและบริษัทหลักทรัพย์ใช้ผู้ช่วยสนทนาเพื่อตอบคำถามพื้นฐานของลูกค้าได้ตลอด 24 ชั่วโมง เช่น การสอบถามผลิตภัณฑ์ หรือขั้นตอนการทำธุรกรรม ส่วนงานภายใน LLM ช่วยร่างบันทึก อีเมล และเอกสารเบื้องต้นได้รวดเร็ว อย่างไรก็ตาม ทุกการประยุกต์เกี่ยวกับลูกค้าและการปฏิบัติตามกฎเกณฑ์ต้องมีการออกแบบให้รัดกุมเป็นพิเศษ เพื่อป้องกันการให้คำแนะนำที่ผิดพลาดหรือขัดต่อกฎหมาย และต้องมีมนุษย์ตรวจสอบก่อนนำไปใช้จริงเสมอ

1.6.4 ตัวอย่าง: การใช้งานอื่น ๆ และข้อควรระวัง

นอกจากการสรุปเอกสารแล้ว Generative AI ยังช่วยงานได้อีกมาก ดังตารางที่ 1.2 ซึ่งสรุปกรณีใช้งาน ประโยชน์ และข้อควรระวังของแต่ละงาน

ตารางที่ 1.2 การใช้งาน Generative AI ในงานการเงินและข้อควรระวัง

กรณีใช้งาน	ประโยชน์หลัก	ข้อควรระวัง
วิเคราะห์ความรู้สึกของข่าว (Sentiment)	จับอารมณ์ตลาดได้เร็ว	อาจตีความประชด/บริบทเฉพาะผิด
ผู้ช่วยตอบลูกค้า (Chatbot)	บริการ 24 ชม. ลดภาระพนักงาน	ต้องกันการให้คำแนะนำที่ผิดกฎ
ร่างเอกสาร/อีเมล/สัญญา	ร่างเร็ว ปรับสำนวนได้	ต้องมีผู้เชี่ยวชาญตรวจสอบก่อนใช้
ช่วยเขียนโค้ด/สูตร Excel	เร่งงานวิเคราะห์เชิงปริมาณ	ต้องทดสอบผลลัพธ์ทุกครั้ง
ตรวจจบบรูปแบบทุจริต	อ่านข้อมูลปริมาณมากได้	ความเป็นส่วนตัวและอคติของข้อมูล

ลองดูตัวอย่างงานตรวจจบบูจริตให้ลึกขึ้น ธนาคารมีรายการธุรกรรมจำนวนมหาศาลในแต่ละวัน การให้แบบจำลองช่วยอ่านรูปแบบของธุรกรรม เช่น ความถี่ ช่วงเวลา สถานที่ และความสัมพันธ์ระหว่างบัญชี ช่วยคัดกรองธุรกรรมที่น่าสงสัยขึ้นมาให้เจ้าหน้าที่ตรวจสอบก่อน อย่างไรก็ตาม ระบบเช่นนี้ต้องออกแบบอย่างระมัดระวัง เพราะการแจ้งเตือนผิดพลาด (False Positive) มากเกินไปจะรบกวนลูกค้า ขณะที่การพลาดธุรกรรมทุจริตจริงก็มีต้นทุนสูง อีกทั้งยังต้องคำนึงถึงความเป็นส่วนตัวและการไม่เลือกปฏิบัติต่อกลุ่มลูกค้าบางกลุ่ม

กฎเหล็กสำหรับงานการเงิน

Generative AI เป็น “ผู้ช่วยร่างและคัดกรอง” ที่ขยัน แต่ไม่ใช่ “แหล่งความจริง” ผู้ใช้ต้องตรวจสอบข้อเท็จจริง ตัวเลข และความสอดคล้องกับกฎเกณฑ์เสมอ โดยเฉพาะงานที่เกี่ยวกับเงินของผู้อื่นและการปฏิบัติตามกฎหมาย

กิจกรรม 1.3

เลือกงานการเงินที่คุณสนใจหนึ่งงานจากตารางที่ 1.2 แล้วร่าง “คำสั่ง” (Prompt) ที่ชัดเจนเพื่อให้ผู้ช่วย AI ทำงานนั้น พร้อมระบุว่า คุณจะ “ตรวจสอบ” ผลลัพธ์อย่างไร เก็บ Prompt ไว้เปรียบเทียบกับเทคนิคการเขียน Prompt ในบทที่ 2

ลงมือทำ: ใช้ TensorFlow บน Google Colab พยากรณ์อุณหภูมิล่วงหน้า

หัวข้อนี้คือหัวใจการลงมือปฏิบัติของบทนี้ เราจะสร้างแบบจำลองพยากรณ์ “อนุกรมเวลา” (Time Series) ด้วย TensorFlow ตั้งแต่ศูนย์ บนเครื่องมือฟรีที่ชื่อ Google Colab โจทย์ที่เลือกคือการพยากรณ์อุณหภูมิในอนาคตจากข้อมูลสภาพอากาศที่บันทึกตามช่วงเวลา เหตุผลที่ใช้ข้อมูลอุณหภูมิเป็นตัวอย่างก็เพราะมันมี “รูปแบบตามเวลา” ที่ชัดเจน (รอบกลางวัน-กลางคืน และรอบฤดูกาล) เห็นภาพง่าย และที่สำคัญ เทคนิคเดียวกันนี้ใช้กับอนุกรมเวลาทางการเงินได้โดยตรง เช่น ราคาหุ้น อัตราแลกเปลี่ยน หรือกระแสเงินสด ซึ่งเป็นแนวคิดเดียวกับกรณีศึกษาในหัวข้อ 1.3

สิ่งที่คุณจะได้ทำในหัวข้อนี้

- เปิดและตั้งค่า Google Colab สำหรับงาน Machine Learning
- โหลดและสำรวจข้อมูลอนุกรมเวลา (อุณหภูมิรายชั่วโมง)
- เตรียมข้อมูล : ทำให้เป็นมาตรฐาน แบ่งชุดฝึก/ตรวจสอบ/ทดสอบ และสร้างหน้าต่างข้อมูล
- สร้าง ฟังก์ชัน และประเมินแบบจำลองด้วย Keras (TensorFlow)
- พยากรณ์อุณหภูมิและตีความผล แล้วเชื่อมโยงกับการพยากรณ์ทางการเงิน

เตรียมตัวก่อนเริ่ม

สิ่งที่คุณต้องมีคือบัญชี Google และเว็บเบราว์เซอร์เท่านั้น ไม่ต้องติดตั้งโปรแกรมใดๆ ในเครื่อง เพราะ Colab รันบนคลาวด์ทั้งหมด แนะนำให้เปิดบทเรียนนี้คู่กับหน้าจอ Colab แล้วพิมพ์ตามไปที่ละขั้น (อย่าเพิ่งกดล่อกวาง การพิมพ์เองช่วยให้เข้าใจได้มากขึ้น)

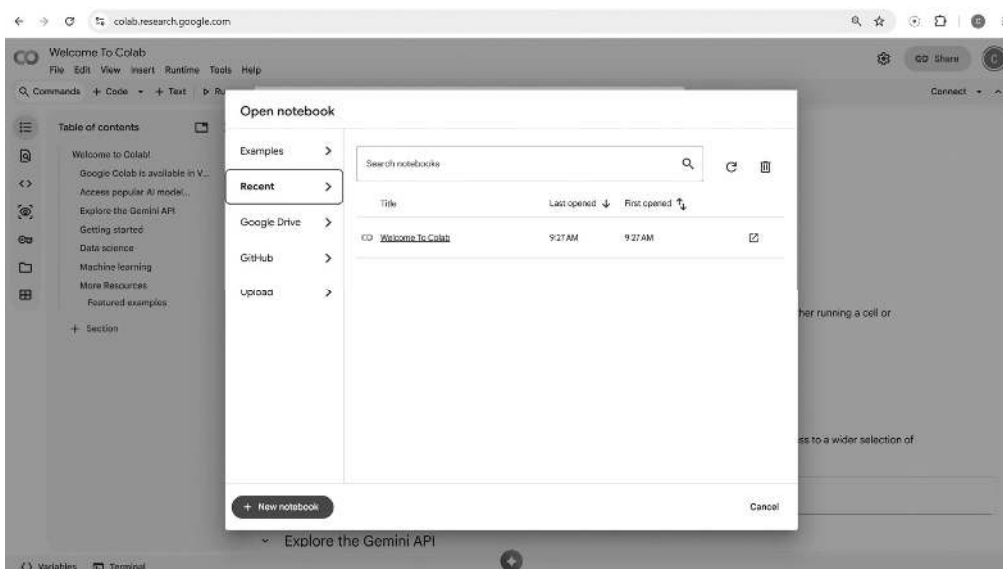
ก่อนลงมือ เรามาดูความเข้าใจกรอบคิดของงานนี้กันสักเล็กน้อย สิ่งที่เรากำลังจะทำคือ **“การเรียนรู้แบบมีผู้สอน”** (Supervised Learning) ซึ่งหมายความว่าเราจะป้อนทั้ง **“โจทย์”** (อุณหภูมิ 24 ชั่วโมงย้อนหลัง) และ **“เฉลย”** (อุณหภูมิชั่วโมงถัดไป) ให้แบบจำลองดูจำนวนมาก เพื่อให้มันค่อยๆ จับรูปแบบความสัมพันธ์ระหว่างโจทย์กับเฉลย เมื่อฝึกเสร็จเราจะทดสอบว่าแบบจำลองทำโจทย์ที่ **“ไม่เคยเห็นเฉลย”** ได้ดีเพียงใด งานพยากรณ์อุณหภูมิและพยากรณ์ราคาสินทรัพย์จึงเป็นปัญหาประเภทเดียวกันในเชิงโครงสร้าง คือการทำนายค่าตัวเลขในอนาคตจากข้อมูลในอดีต

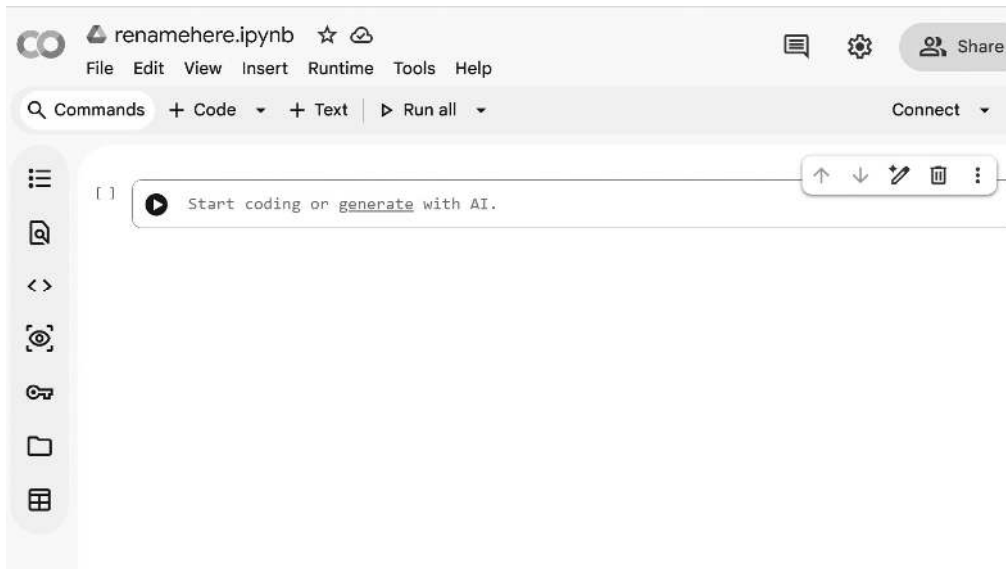
1.7.1 ขั้นที่ 0: รู้จัก TensorFlow, Keras และ Google Colab

TensorFlow คือไลบรารีโอเพนซอร์สสำหรับสร้างและฝึกแบบจำลอง Machine Learning ที่พัฒนาโดย Google และเป็นเครื่องมือเดียวกับที่งานวิจัยพยากรณ์ราคาหุ้นในหัวข้อ 1.3 ใช้ ส่วน Keras คือส่วนต่อประสานระดับสูงของ TensorFlow ที่ทำให้เราประกอบโครงข่ายได้ง่ายเหมือนต่อเลโก้ และ Google Colab (ชื่อเต็มคือ Colaboratory บางครั้งเรียกกันว่า Google Studio) คือสมุดบันทึก (Notebook) ที่รันโค้ด Python บนเซิร์ฟเวอร์ของ Google ผ่านเบราว์เซอร์ พร้อมตัวเร่งฮาร์ดแวร์อย่าง GPU ให้ใช้ฟรี

1.7.2 ขั้นที่ 1: เปิดสมุดบันทึกใหม่บน Colab

เปิดเบราว์เซอร์ไปที่ colab.research.google.com แล้วลงชื่อเข้าใช้ด้วยบัญชี Google จากนั้นเลือกเมนู File แล้วเลือก New notebook (หรือในกล่อง Open notebook ที่ปรากฏขึ้นให้กดปุ่ม New notebook ที่มุมล่างซ้าย) ระบบจะเปิดสมุดบันทึกเปล่าที่มี **“เซลล์”** (Cell) พร้อมข้อความ **“Start coding or generate with AI”** ให้เริ่มพิมพ์โค้ด ดังรูปที่ 1.17

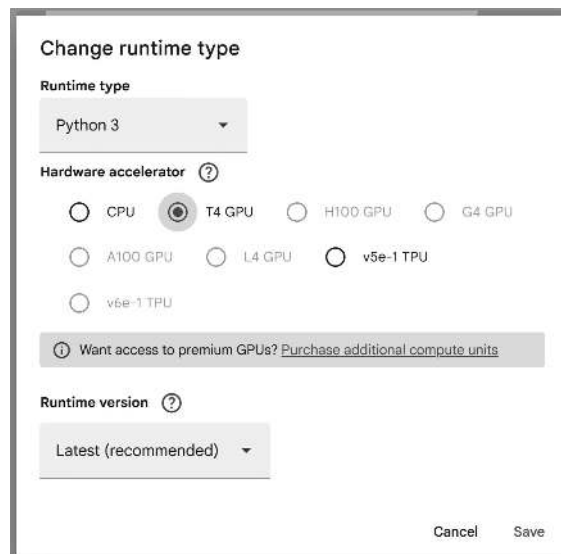




รูปที่ 1.17 การสร้างสมุดบันทึกใหม่ ภาพบนคือกล่อง Open notebook (เรียกจากเมนู File) ที่มีปุ่ม New notebook ส่วนภาพล่างคือสมุดบันทึกเปล่าที่ได้ พร้อมเซลล์แรกสำหรับพิมพ์โค้ด

1.7.3 ขั้นที่ 2 (ทางเลือก): เลือกใช้ GPU ให้ฝึกเร็วขึ้น

สำหรับตัวอย่างเล็กๆ ในบทนี้การใช้ CPU ประมวลผล ก็เพียงพอ แต่ถ้าต้องการให้การฝึกเร็วขึ้น ให้เลือกเมนู Runtime แล้วเลือก Change runtime type จากนั้นในหัวข้อ Hardware accelerator เลือก T4 GPU แล้วกด Save (ช่อง Runtime type ปล่อยเป็น Python 3 และ Runtime version เป็น Latest) ดังรูปที่ 1.18

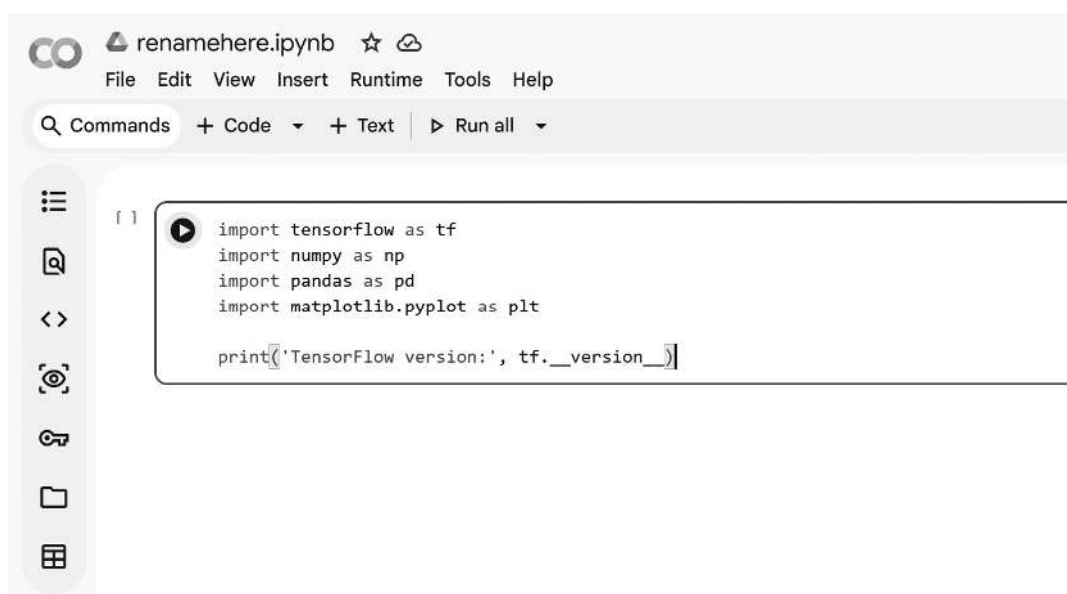


รูปที่ 1.18 กล่อง Change runtime type สำหรับเลือกตัวเร่งฮาร์ดแวร์ (Hardware accelerator) เป็น T4 GPU เพื่อเร่งการฝึกแบบจำลอง

1.7.4 ขั้นที่ 3: พิมพ์และรันเซลล์โค้ดแรก

คลิกในเซลล์แล้วพิมพ์โค้ดเรียกใช้ไลบรารีที่จำเป็น จากนั้นกดปุ่มรัน (สัญลักษณ์สามเหลี่ยมทางซ้ายของเซลล์) หรือกด Shift+Enter เพื่อส่งทำงาน ดังรูปที่ 1.19

เซลล์ที่ 1: เรียกใช้ไลบรารีที่จำเป็น
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
print('TensorFlow version:', tf.__version__)



รูปที่ 1.19 การพิมพ์โค้ดในเซลล์และกดปุ่มรัน

เมื่อรันสำเร็จ เซลล์จะมีหมายเลขลำดับและเครื่องหมายถูกสีเขียวทางซ้าย และผลลัพธ์จะปรากฏใต้เซลล์ เช่น **“TensorFlow version: 2.20.0”** ดังรูปที่ 1.20 หากขึ้นข้อความแสดงข้อผิดพลาดสีแดงให้อ่านบรรทัดสุดท้ายซึ่งมักบอกสาเหตุ แล้วแก้ไขก่อนรันใหม่

```
[1] ✓ 6s
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

print('TensorFlow version:', tf.__version__)

... TensorFlow version: 2.20.0
```

รูปที่ 1.20 ผลลัพธ์ที่ได้เซลล์เมื่อรันสำเร็จ แสดง “TensorFlow version: 2.20.0” พร้อมหมายเลขลำดับและเครื่องหมายถูกสีเขียว

แก้ปัญหาที่พบบ่อยใน Colab

- NameError: ใช้ตัวแปรก่อนสร้าง มักเกิดจากยังไม่ได้รันเซลล์ก่อนหน้านี้ ให้รันจากบนลงล่างตามลำดับ
- ModuleNotFoundError: ไม่พบไลบรารี ให้ติดตั้งด้วยคำสั่ง !pip install ชื่อไลบรารีในเซลล์
- การเชื่อมต่อหลุด: หากทิ้งไว้นานเกินไป ให้เชื่อมต่อใหม่แล้วรันทุกเซลล์อีกครั้ง (Runtime > Run all)
- ค่าเปลี่ยนไปมาแบบไม่คาดคิด: ให้รีสตาร์ทรันไทม์เพื่อล้างตัวแปรเก่าทั้งหมดแล้วเริ่มใหม่

เคล็ดลับการทำงานกับ Colab

แต่ละเซลล์รันแยกกันได้ แต่ใช้ตัวแปรร่วมกันตามลำดับที่รัน ควรรันจากบนลงล่าง ถ้าผลแปลกๆ ให้เลือก รันไทม์ > เริ่มรันไทม์ใหม่ (Restart) แล้วรันใหม่ทั้งหมด Colab จะตัดการเชื่อมต่อหากไม่ใช้งานนาน ควรบันทึกงานไว้ใน Google Drive เป็นระยะ

1.7.5 ขั้นที่ 4: เตรียมข้อมูลอุณหภูมิ

ในงานจริงเรามักโหลดข้อมูลจากไฟล์ CSV ที่บันทึกตัวแปรสภาพแวดล้อม เช่น อุณหภูมิ ความชื้น ความกดอากาศ ตามช่วงเวลา ชุดข้อมูลสภาพอากาศที่นิยมใช้สอนคือ ชุด Jena Climate ซึ่งบันทึกทุก 10 นาทีเป็นเวลาหลายปี ในบทนี้เพื่อให้รันได้ทุกที่ เราจะสร้างข้อมูลอุณหภูมิรายชั่วโมงจำลองที่มีรูปแบบเหมือนของจริง คือมีรอบฤดูกาลรายปีและ

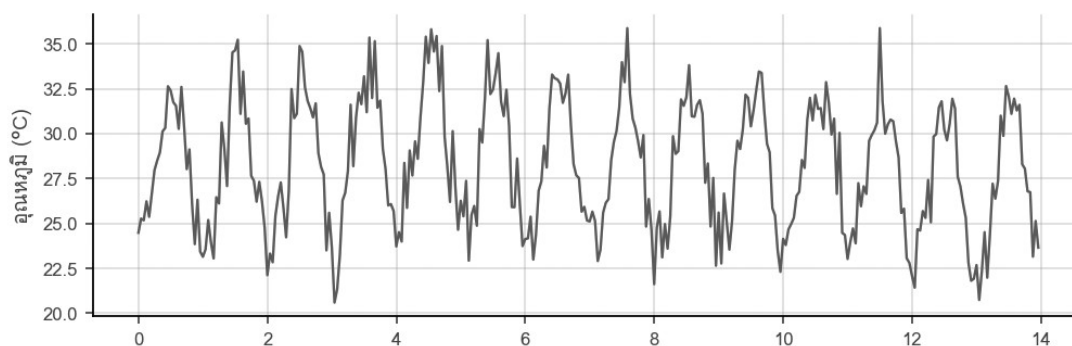
รอบกลางวัน-กลางคืน ซ้อนด้วยความผันผวนแบบสุ่ม ได้ด้านล่างสร้างข้อมูลประมาณสองปีและแสดงกราฟช่วงตัวอย่าง

```
# เซลล์ที่ 2: สร้าง/โหลดข้อมูลอุณหภูมิรายชั่วโมง
hours = np.arange(0, 24*365*2)
day, year = 24, 24*365
yearly = 9.0*np.sin(2*np.pi*(hours-24*80)/year) # รอบฤดูกาล
daily = 4.5*np.sin(2*np.pi*(hours-7)/day) # รอบกลางวัน-กลางคืน
noise = np.random.normal(0, 1.4, size=hours.size)
temp = 13.0 + yearly + daily + noise # อุณหภูมิ (C)

df = pd.DataFrame({'temp': temp})
df.head()
```

ขั้นตอนแรกที่ดีของงานข้อมูลคือ “ดูข้อมูลด้วยตา” ก่อนเสมอ ลองวาดกราฟอุณหภูมิช่วง 14 วันเพื่อดูรอบกลางวัน-กลางคืน ดังโค้ดและรูปที่ 1.21

```
# เซลล์ที่ 3: วาดกราฟข้อมูลช่วงตัวอย่าง 14 วัน
plt.figure(figsize=(10,3))
plt.plot(temp[24*120 : 24*120 + 24*14])
plt.xlabel('ชั่วโมง'); plt.ylabel('อุณหภูมิ (°C)')
plt.title('อุณหภูมิรายชั่วโมง ช่วง 14 วัน'); plt.show()
```

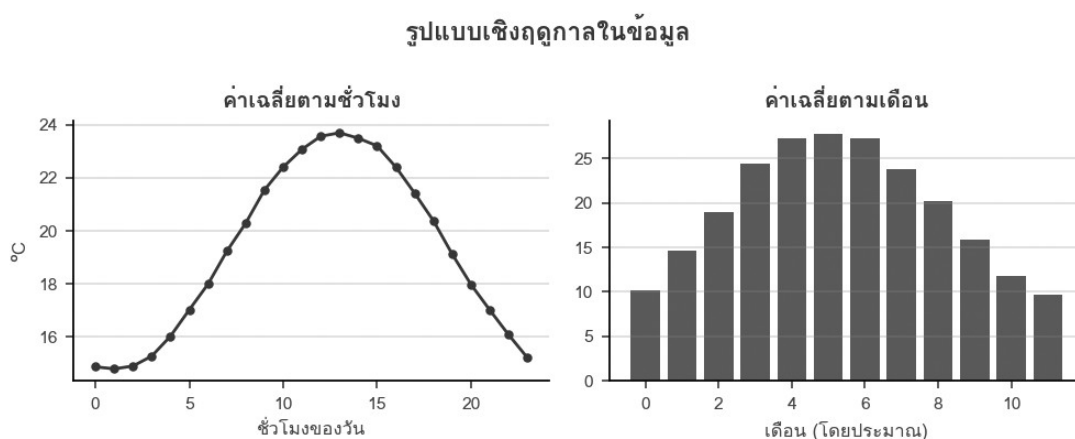


รูปที่ 1.21 กราฟอุณหภูมิรายชั่วโมง เห็นรอบกลางวัน-กลางคืนชัดเจน

นอกจากดูกราฟแล้ว ควรดูสรุปเชิงสถิติของข้อมูลด้วย คำสั่ง `df.describe()` จะแสดงค่าเฉลี่ย ค่าต่ำสุด-สูงสุด และส่วนเบี่ยงเบนมาตรฐาน ซึ่งช่วยให้เราตรวจพบความผิดปกติ เช่น ค่าที่หายไปหรือค่าที่เป็นไปไม่ได้ ก่อนนำข้อมูลไปฝึก การ **“รู้จักข้อมูล”** ของตนอย่างถ่องแท่นับเป็นนิสัยที่ดีของนักวิเคราะห์ข้อมูลทุกคน

```
# เซลล์ที่ 3ก: ดูสรุปเชิงสถิติของข้อมูล
print(df.describe())
print('จำนวนค่าที่หายไป:', df['temp'].isna().sum())
```

เราควรเข้าใจ **“รูปแบบตามฤดูกาล”** ของข้อมูลด้วย เพราะแบบจำลองจะเรียนรู้รูปแบบเหล่านี้ ลองหาค่าเฉลี่ยอุณหภูมิตามชั่วโมงของวันและตามเดือน จะเห็นว่ากลางวันร้อนกว่ากลางคืน และบางเดือนร้อนกว่าบางเดือน ดังรูปที่ 1.22



รูปที่ 1.22 รูปแบบเชิงฤดูกาล: ค่าเฉลี่ยตามชั่วโมงของวัน (ซ้าย) และตามเดือน (ขวา)

1.7.6 ขั้นที่ 5: ทำให้เป็นมาตรฐานและแบ่งชุดข้อมูล

ก่อนฝึก เราต้อง **“ทำให้เป็นมาตรฐาน”** (Normalize) คือปรับข้อมูลให้มีค่าเฉลี่ยใกล้เคียงศูนย์และความแปรปรวนใกล้เคียงหนึ่ง เพื่อให้โครงข่ายเรียนรู้ได้เสถียรขึ้น ข้อควรระวังสำคัญคือ ต้องคำนวณค่าเฉลี่ยและส่วนเบี่ยงเบนจาก **“ชุดฝึกเท่านั้น”** แล้วนำไปใช้กับทุกชุด เพื่อป้องกันการรั่วไหลของข้อมูลอนาคต (Data Leakage) จากนั้นแบ่งข้อมูลตามลำดับเวลาเป็นชุดฝึก 70% ชุดตรวจสอบ 15% และชุดทดสอบ 15%

เซลล์ที่ 4: ทำให้เป็นมาตรฐานและแบ่งชุดข้อมูลตามเวลา
n = len(temp)
i_train, i_val = int(n*0.7), int(n*0.85)
mu = temp[:i_train].mean() # ใช้ค่าจากชุดฝึกเท่านั้น
sd = temp[:i_train].std()
temp_n = (temp - mu) / sd # normalize
train = temp_n[:i_train]
val = temp_n[i_train:i_val]
test = temp_n[i_val:]
print(len(train), len(val), len(test))

ทำไมต้องแบ่งตามเวลา ไม่ใช่สุ่ม

งานอนุกรมเวลาต้องแบ่งข้อมูลตามลำดับเวลา (อดีตไว้ฝึก อนาคตไว้ทดสอบ) ห้ามสุ่มสลับ เพราะถ้าใช้ข้อมูลอนาคตมาฝึก แบบจำลองจะ “แอบเห็นอนาคต” ทำให้ผลทดสอบดีเกินจริง หลักการนี้สำคัญมากในงานการเงิน เพราะเราต้องการรู้ว่าแบบจำลองทำนาย “อนาคตที่ยังไม่เคยเห็น” ได้จริงหรือไม่

ก่อนสร้างแบบจำลองที่ซับซ้อน เรามักกำหนด “เส้นฐาน” (Baseline) ง่ายๆ ไว้เปรียบเทียบ เส้นฐานที่นิยมที่สุดสำหรับอนุกรมเวลาคือ “ทำนายว่าค่าถัดไปเท่ากับค่าล่าสุด” (Naive Forecast) ถ้าแบบจำลองที่เราสร้างไม่สามารถเอาชนะเส้นฐานง่ายๆ นี้ได้ ก็แสดงว่ายังไม่คุ้มค่าที่จะใช้ การมีเส้นฐานช่วยให้เราประเมินคุณค่าที่แท้จริงของแบบจำลองอย่างชัดเจน

เซลล์ที่ 4ก: เส้นฐานอย่างง่าย (naive forecast)
naive_pred = test[:-2] # ใช้ค่าล่าสุดเป็นค่าทำนายถัดไป
naive_true = test[2:]
naive_mae = np.mean(np.abs(naive_true - naive_pred)) * sd
print('Baseline MAE (°C):', round(naive_mae, 3))

เมื่อมีเส้นฐานแล้ว เป้าหมายของเราคือสร้างแบบจำลองที่ให้ MAE ต่ำกว่าเส้นฐานนี้ อย่างมีนัยสำคัญ

1.7.7 ขั้นที่ 6: สร้างหน้าต่างข้อมูล (Windowing)

แบบจำลองพยากรณ์อนุกรมเวลาเรียนรู้จาก “หน้าต่าง” ของอดีต เพื่อทำนายค่าถัดไป ในตัวอย่างนี้เราใช้อุณหภูมิ 24 ชั่วโมงย้อนหลังเป็นอินพุต เพื่อทำนายอุณหภูมิของชั่วโมงถัดไป (Label) แล้วเลื่อนหน้าต่างไปที่ละก้าวเพื่อสร้างตัวอย่างฝึกจำนวนมากจากอนุกรมเดียว ดังรูปที่ 1.23



เลื่อนหน้าต่างทีละก้าวเพื่อสร้างตัวอย่างฝึกจำนวนมากจากอนุกรมเดียว

รูปที่ 1.23 การสร้างหน้าต่างข้อมูล (Windowing) สำหรับอนุกรมเวลา

```
# เซลล์ที่ 5: สร้างหน้าต่างข้อมูล (X = 24 ชม.ย้อนหลัง, y = ชม.ถัดไป)
```

```
def make_windows(series, w=W):
```

```
    X, y = [], []
```

```
    for i in range(len(series) - w - 1):
```

```
        X.append(series[i:i+w])
```

```
        y.append(series[i+w+1])
```

```
X_train, y_train = make_windows(train)
```

```
X_val, y_val = make_windows(val)
```

```
X_test, y_test = make_windows(test)
```

```
print('รูปทรงของ X_train:', X_train.shape)
```

1.7.8 ขั้นที่ 7: ประกอบแบบจำลองด้วย Keras

ตอนนี้ถึงขั้นประกอบโครงข่ายซึ่งต้องใช้จินตนาการพอสมควร เราจะใช้แบบ Sequential ซึ่งต่อชั้นเรียงกันจากบนลงล่าง ประกอบด้วยชั้น Dense (เชื่อมต่อเต็ม) ที่มีฟังก์ชันกระตุ้น ReLU สองชั้น มีชั้น Dropout เพื่อลด Overfitting และปิดท้ายด้วยชั้น Dense(1) ที่ให้เอาต์พุตเป็นอุณหภูมิหนึ่งค่า ดังรูปที่ 1.24 (ผู้ที่สนใจอาจเปลี่ยนชั้นแรกเป็น LSTM เพื่อจับลำดับ

เวลาได้ดีขึ้น) เนื่องจากโครงข่ายนี้เกิดจากจินตนาการ ผู้เรียนสามารถกำหนดโครงข่ายขึ้นมาตามใจก่อน แล้วค่อยๆ พิจารณาจากเส้นโค้งการเรียนรู้ของแบบจำลองที่จะกล่าวต่อไป เพื่อปรับโครงข่ายให้เหมาะกับการพยากรณ์มากที่สุด



รูปที่ 1.24 โครงสร้างแบบจำลองพยากรณ์ที่ประกอบด้วยชั้น Dense และ Dropout

เซลล์ที่ 6: ประกอบแบบจำลอง
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout, Input
model = Sequential([
Input(shape=(W,)),
Dense(64, activation='relu'),
Dense(32, activation='relu'),
Dropout(0.2),
Dense(1) # เอาต์พุต: อุณหภูมิที่ทำนาย
])
model.summary()

คำสั่ง `model.summary()` จะแสดงโครงสร้างและจำนวนพารามิเตอร์ของแต่ละชั้น ลองสังเกตว่าพารามิเตอร์ส่วนใหญ่อยู่ที่ชั้นแรก เพราะรับอินพุต 24 ค่าเชื่อมเข้า 64 หน่วย

เหตุใดจึงเลือกใช้ชั้น Dense เป็นตัวอย่างเริ่มต้น คำตอบคือเพื่อความเรียบง่ายและเข้าใจง่าย แต่สำหรับงานอนุกรมเวลายังมีสถาปัตยกรรมอื่นที่ออกแบบมาเพื่อจับลำดับเวลาโดยเฉพาะ ตารางที่ 1.3 เปรียบเทียบทางเลือกที่พบบ่อย ผู้เรียนสามารถทดลองเปลี่ยนชั้นแรกเป็น LSTM หรือ Conv1D ในกิจกรรมท้ายหัวข้อเพื่อสังเกตความต่างของผลลัพธ์

ตารางที่ 1.3 เปรียบเทียบสถาปัตยกรรมที่นิยมใช้สำหรับงานพยากรณ์อนุกรมเวลา

สถาปัตยกรรม	จุดเด่น	เหมาะกับ
Dense (เชื่อมต่อเต็ม)	เรียบง่าย ฝึกเร็ว เข้าใจง่าย	อนุกรมสั้น เริ่มต้นเรียนรู้
LSTM / GRU (โครงข่ายวกซ้ำ)	จับความสัมพันธ์ระยะยาวตามเวลาได้ดี	อนุกรมที่มีความจำยาว
Conv1D (คอนโวลูชัน 1 มิติ)	จับรูปแบบเฉพาะถิ่นได้เร็ว	สัญญาณที่มีลวดลายซ้ำ
Transformer	จับความสัมพันธ์ไกลๆ และประมวลผลขนาน	ข้อมูลปริมาณมากและซับซ้อน

1.7.9 ชั้นที่ 8: คอมไพล์และฝึกแบบจำลอง

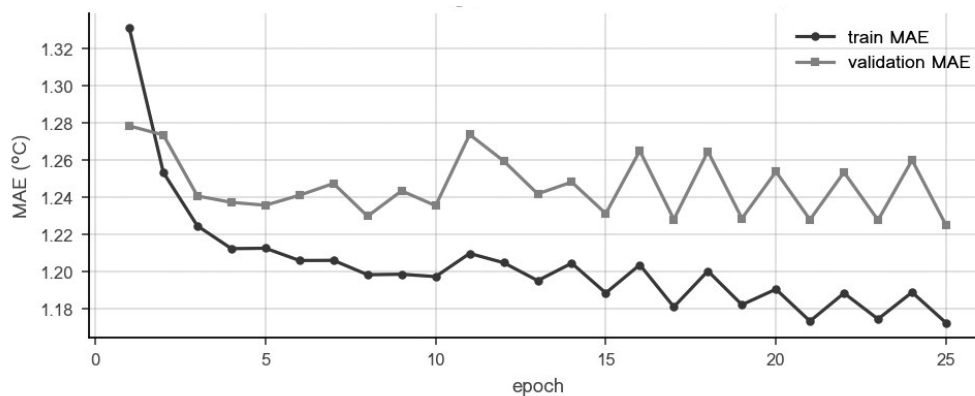
ก่อนฝึก เราต้อง **“คอมไพล์”** เพื่อกำหนดสามสิ่ง คือ ตัวปรับค่า (optimizer) ที่นิยมคือ Adam, ฟังก์ชันสูญเสีย (loss) สำหรับงานพยากรณ์ตัวเลขนิยมใช้ Mean Squared Error และตัววัดผล (metric) ในที่นี้ใช้ Mean Absolute Error (MAE) เพราะดีความเป็นหน่วยองศา จากนั้นเรียก `model.fit` เพื่อเริ่มฝึก โดยส่งชุดตรวจสอบไปด้วยเพื่อเฝ้าดูว่าไม่เกิด Overfitting หรือไม่

เซลล์ที่ 7: คอมไพล์และฝึก
<code>model.compile(optimizer='adam', loss='mse', metrics=['mae'])</code>
<code>history = model.fit(</code>
<code> X_train, y_train,</code>
<code> validation_data=(X_val, y_val),</code>
<code> epochs=25, batch_size=256, verbose=1)</code>

ระหว่างฝึก Colab จะแสดงความคืบหน้าทีละ epoch พร้อมค่า loss และ val_loss ที่ควรลดลงเรื่อยๆ (ดูตัวอย่างผลลัพธ์ในรูปที่ 1.19) เมื่อฝึกเสร็จ เราวาด **“เส้นโค้งการเรียนรู้”** หรือ Learning Curve ของแบบจำลอง เพื่อดูพัฒนาการ ดังโค้ดและรูปที่ 1.25

คำว่า epoch หมายถึงการที่แบบจำลองได้ดูข้อมูลฝึกครบทั้งชุดหนึ่งรอบ การฝึก 25 epoch จึงหมายความว่าแบบจำลองได้เรียนรู้จากข้อมูลทั้งหมด 25 รอบ ส่วน batch_size คือจำนวนตัวอย่างที่ป้อนเข้าในแต่ละครั้งของการปรับค่า การใช้ batch ช่วยให้ฝึกได้เร็ว และเสถียรมากกว่าการป้อนทีละตัวอย่าง การเลือกจำนวน epoch ที่เหมาะสมเป็นศิลปะอย่างหนึ่ง น้อยเกินไปแบบจำลองจะเรียนรู้ไม่พอ (underfitting) มากเกินไปอาจจำข้อมูลฝึกจนเกิด overfitting

```
# เซลล์ที่ 8: วาดเส้นโค้งการเรียนรู้
plt.plot(history.history['mae'], label='train MAE')
plt.plot(history.history['val_mae'], label='val MAE')
plt.xlabel('epoch'); plt.ylabel('MAE (°C)')
plt.legend(); plt.show()
```



รูปที่ 1.25 เส้นโค้งการเรียนรู้: ค่าผิดพลาดลดลงเมื่อฝึกมากขึ้น

อ่านเส้นโค้งการเรียนรู้อย่างไร

- ถ้าทั้ง train และ val MAE ลดลงและเข้าใกล้กัน แสดงว่าเรียนรู้ได้ดี
- ถ้า train ลดลงเรื่อยๆ แต่ val เริ่มสูงขึ้น แสดงว่า Overfitting ควรลดความซับซ้อน เพิ่ม Dropout หรือหยุดฝึกเร็วขึ้น
- ถ้าทั้งคู่สูงและไม่ลด แสดงว่า Underfitting อาจต้องเพิ่มชั้น/หน่วย หรือฝึกนานขึ้น

1.7.10 ขั้นที่ 9: ประเมินและพยากรณ์

ขั้นตอนที่ห้าคือทดสอบกับ “ชุดทดสอบ” ที่แบบจำลองไม่เคยเห็น เพื่อประเมินความสามารถจริง เราใช้ `model.evaluate` เพื่อดู MAE บนชุดทดสอบ แล้ว `model.predict` เพื่อพยากรณ์ จากนั้นแปลงค่ากลับจากสเกลมาตรฐานเป็นหน่วยองศาเพื่อให้ดีดูความง่าย

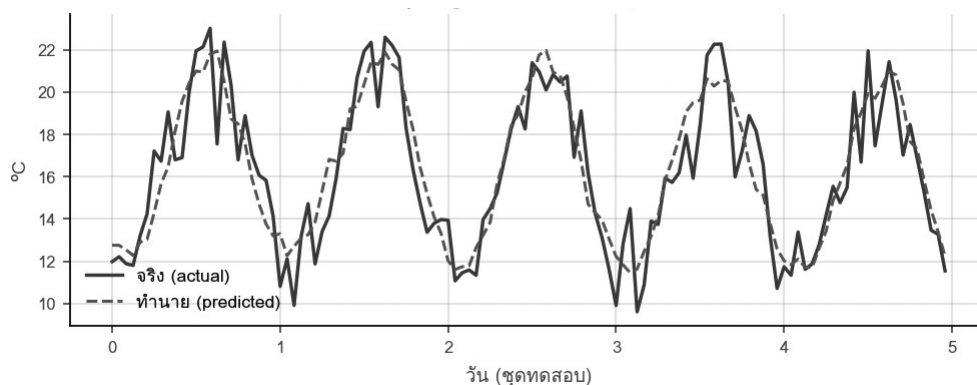
```
# เซลล์ที่ 9: ประเมินและพยากรณ์
loss, mae = model.evaluate(X_test, y_test, verbose=0)
print('Test MAE (สเกลมาตรฐาน):', round(mae,3))

pred = model.predict(X_test).flatten()
pred_C = pred * sd + mu          # แปลงกลับเป็น °C

act_C = y_test * sd + mu

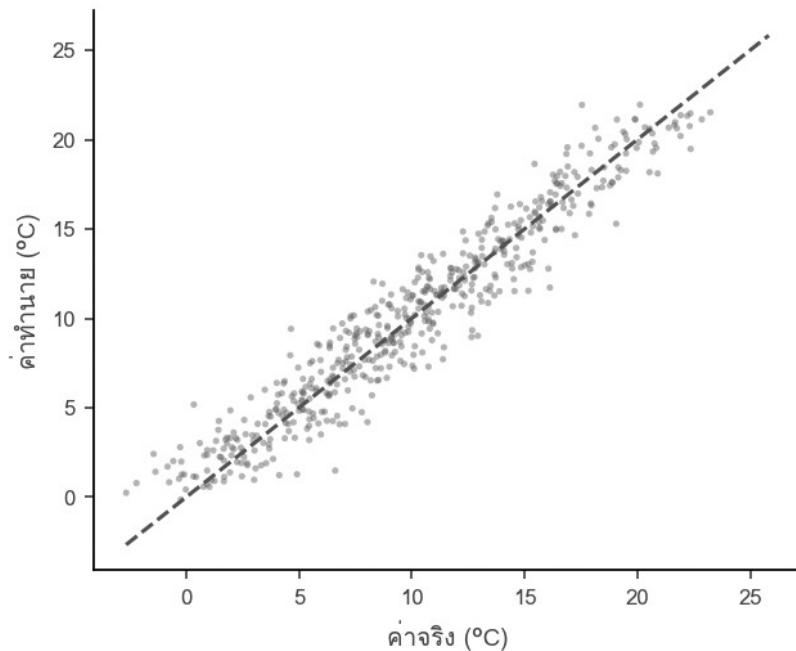
plt.figure(figsize=(10,3))
plt.plot(act_C[:24*5], label='จริง')
plt.plot(pred_C[:24*5], '--', label='ทำนาย')
plt.legend(); plt.show()
```

รูปที่ 1.26 แสดงผลการทำนายเทียบกับค่าจริงในชุดทดสอบ จะเห็นว่าเส้นทำนาย (เส้นประ) เกาะตามเส้นจริงได้ดี โดยมีค่าความคลาดเคลื่อนเฉลี่ย (MAE) ประมาณ 1.2 องศาเซลเซียส ซึ่งถือว่าน่าพอใจสำหรับแบบจำลองอย่างง่าย



รูปที่ 1.26 ผลการพยากรณ์อุณหภูมิช่วงหน้า 1 ชั่วโมงเทียบกับค่าจริง

อีกมุมมองที่ช่วยประเมินคุณภาพคือกราฟกระจาย (Scatter) ระหว่างค่าจริงกับค่าทำนาย ยิ่งจุดเกาะใกล้เส้นทแยงมุม 45 องศา แสดงว่าทำนายแม่นยำ ดังรูปที่ 1.27



รูปที่ 1.27 กราฟกระจายค่าจริงกับค่าทำนาย ยิ่งใกล้เส้นทแยงยิ่งแม่นยำ

ค่าที่เราใช้วัดคุณภาพคือ MAE (Mean Absolute Error) ซึ่งคือค่าเฉลี่ยของขนาดความคลาดเคลื่อน ดีความง่ายว่า **“โดยเฉลี่ยทำนายพลาดกี่องศา”** อีกตัววัดที่นิยมคือ RMSE (Root Mean Squared Error) ที่ลงโทษความผิดพลาดขนาดใหญ่หนักกว่า การเลือกตัววัดควรสอดคล้องกับเป้าหมายของงาน เช่น หากความผิดพลาดครั้งใหญ่ๆ มีผลเสียรุนแรง RMSE อาจเหมาะกว่า นอกจากนี้ ควรเทียบผลกับ **“เส้นฐาน”** (Baseline) ง่ายๆ เสมอ เช่น **“ทำนายว่าชั่วโมงถัดไปเท่ากับชั่วโมงนี้”** ถ้าแบบจำลองที่เราสร้างไม่ดีกว่าเส้นฐานนี้ ก็แปลว่ายังไม่คุ้มที่จะใช้

ในงานจริง เรามักมีตัวแปรมากกว่าหนึ่ง เช่น นอกจากอุณหภูมิยังมีความชื้น ความกดอากาศ และความเร็วลม การใช้หลายตัวแปร (Multivariate) มักช่วยให้พยากรณ์แม่นยำขึ้น เพราะแบบจำลองมีข้อมูลประกอบมากกว่า ในทำนองเดียวกัน การพยากรณ์ราคาสินทรัพย์ทางการเงินก็มักใช้หลายปัจจัยร่วมกัน ทั้งราคาในอดีต ปริมาณการซื้อขาย ตัวชี้วัดทางเทคนิค และเวกเตอร์ของข่าวอย่างในกรณีศึกษาของ Leemakdej (2019) การปรับโค้ดให้รับหลายตัวแปรทำได้โดยเปลี่ยนรูปทรงของอินพุตให้เป็นสองมิติ (จำนวนชั่วโมง $\times$ จำนวนตัวแปร)

1.7.11 ขั้นที่ 10: ดีความผลและเชื่อมโยงสู่การเงิน

เราเพิ่งสร้างแบบจำลองที่เรียนรู้ “รูปแบบตามเวลา” จากอดีตเพื่อทำนายอนาคต กระบวนการทั้งหมด ตั้งแต่การเตรียมข้อมูล การทำให้เป็นมาตรฐาน การสร้างหน้าต่าง การฝึก ไปจนถึงการประเมินดี ยชุดทดสอบ คือกระบวนการมาตรฐานเดียวกับการพยากรณ์อนุกรมเวลาทางการเงิน เพียงเปลี่ยนข้อมูลอินพุตจากอุณหภูมิเป็นราคาสินทรัพย์ อัตราดอกเบี้ย หรือกระแสเงินสด นี่คือการเชื่อมโยงไปยังกรณีศึกษาในหัวข้อ 1.3 ที่ใช้ TensorFlow ในลักษณะเดียวกันเพื่อทำนายทิศทางราคาหุ้นจากเวกเตอร์ของข่าว

เพื่อให้เห็นความเชื่อมโยงชัดเจน ตารางที่ 1.4 จับคู่ขั้นตอนในบทเรียนนี้กับขั้นตอนในงานวิจัยพยากรณ์ราคาหุ้น จะเห็นว่าโครงสร้างของงานแทบจะเหมือนกันทุกชั้น ต่างเพียงชนิดของข้อมูลและรายละเอียดของสถาปัตยกรรม

ตารางที่ 1.4 ตารางขั้นตอนการพยากรณ์อุณหภูมิกับการพยากรณ์ราคาหุ้น

ขั้นตอน	บทเรียนนี้ (อุณหภูมิ)	งานวิจัยราคาหุ้น (Leemakdej 2019)
ข้อมูลดิบ	อุณหภูมิรายชั่วโมง	ข่าวการเงิน 78,995 ข่าว
แปลงเป็นตัวเลข	ค่าที่วัดได้โดยตรง	แปลงคำเป็นเวกเตอร์ด้วย word2vec
สร้างอินพุต	หน้าต่าง 24 ชั่วโมง	15 คำ $\times$ 100 มิติ = 1,500 คำ
แบบจำลอง	Dense/LSTM ด้วย Keras	CNN 8 ชั้น ด้วย TensorFlow
เอาต์พุต	อุณหภูมิชั่วโมงถัดไป	ทิศทางราคาหุ้น 10 ตัว
ประเมินผล	MAE บนชุดทดสอบ	ผลตอบแทนกลยุทธ์ในเดือนทดสอบ

เมื่อได้แบบจำลองที่พอใจแล้ว เรายังต้องการบันทึกไว้ใช้ภายหลังโดยไม่ต้องฝึกใหม่ Keras บันทึกแบบจำลองได้ด้วยคำสั่งเดียว และโหลดกลับมาใช้ได้ทันที ดังโค้ดด้านล่าง ในงานจริงเราอาจบันทึกลง Google Drive เพื่อเก็บถาวร

# เซลล์ที่ 10: บันทึกและโหลดแบบจำลอง	
model.save('temp_forecaster.keras')	# บันทึก
from tensorflow.keras.models import load_model	
loaded = load_model('temp_forecaster.keras')	# โหลดกลับมาใช้
print('โหลดแบบจำลองสำเร็จ')	

จุดสำคัญที่ผู้เรียนควรนึกกลับไปคิดคือ ความ “ง่าย” ของการสร้างแบบจำลองด้วย Keras ไม่ได้แปลว่าการนำไปใช้จริงจะง่ายตามไปด้วย ความท้าทายที่แท้จริงของงานพยากรณ์ทางการเงินไม่ได้อยู่ที่การเขียนโค้ดไม่กี่บรรทัด แต่อยู่ที่การเข้าใจข้อมูล การออกแบบการทดสอบที่ชัดเจน การตีความผลอย่างมีวิจารณญาณ และการตระหนักถึงข้อจำกัดทักษะเหล่านี้คือสิ่งที่แยกผู้ใช้เครื่องมืออย่างผิวเผินออกจากผู้เชี่ยวชาญที่ใช้เครื่องมืออย่างเข้าใจ

ข้อควรระวังเมื่อย้ายไปใช้กับข้อมูลการเงิน

- ข้อมูลการเงินมักมีสัญญาณรบกวนสูงและไม่นิ่ง (Non-stationary) มากกว่าข้อมูลอุณหภูมิ จึงทำนายยากกว่ามาก
- ระวัง Data Leakage อย่างเข้มงวด
- แบ่งข้อมูลตามเวลาเสมอ และอย่าใช้ข้อมูลอนาคตมาช่วยในการพยากรณ์ แต่ใช้เพื่อตรวจสอบเท่านั้น
- ผลตอบแทนที่ดีในการทดสอบย้อนหลังต้องหักต้นทุนซื้อขายและทดสอบในลายช่วงตลาดก่อนซื้อ

1.7.12 ทำให้แบบจำลองดีขึ้น: การหยุดฝึกอัตโนมัติและการลด Overfitting

ในทางปฏิบัติ เรามักไม่รู้ล่วงหน้าว่าควรฝึกกี่ epoch จึงพอดี ฝึกน้อยเกินไป Underfitting ฝึกมากเกินไป Overfitting เทคนิคที่ช่วยได้คือ “การหยุดฝึกอัตโนมัติ” (Early Stopping) ซึ่งจะเฝ้าดูค่า val_loss และหยุดฝึกเองเมื่อค่าไม่ดีขึ้นต่อเนื่องหลายรอบ พร้อมคืนค่าพารามิเตอร์ที่ดีที่สุดให้ ดังโค้ดด้านล่าง

เซลล์ที่ 11: ใช้ EarlyStopping เพื่อหยุดฝึกที่จุดที่ดีที่สุด
from tensorflow.keras.callbacks import EarlyStopping
es = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)
history = model.fit(
X_train, y_train,
validation_data=(X_val, y_val),
epochs=100, batch_size=256,
callbacks=[es], verbose=1)

นอกจาก Early Stopping แล้ว ยังมีเทคนิคลด Overfitting อื่นๆ เช่น การเพิ่มชั้น Dropout (อย่างที่เราใช้ไปแล้ว) การลดความซับซ้อนของแบบจำลอง และการเพิ่มข้อมูลฝึก ในงานการเงินที่ข้อมูลมักมีสัญญาณรบกวนสูง การควบคุม Overfitting ยิ่งสำคัญ เพราะแบบจำลองที่ “จำ” สัญญาณรบกวนในอดีตจะทำนายอนาคตได้แย่

1.7.13 พยากรณ์ล่วงหน้าหลายชั่วโมง

จนถึงตอนนี้เราทำนายเพียงชั่วโมงถัดไป แต่ในงานจริงเรามักอยากรู้แนวโน้มล่วงหน้าหลายช่วงเวลา การพยากรณ์หลายช่วงเวลาทำได้สองแนวทางหลัก แนวทางแรกคือ “ทำนายซ้ำ” (Recursive) คือนำค่าที่ทำนายได้ป้อนกลับเป็นอินพุตเพื่อทำนายช่วงเวลาถัดไป แนวทางที่สองคือออกแบบให้แบบจำลองมีเอาต์พุตหลายค่าพร้อมกัน คือเปลี่ยนชั้นสุดท้ายจาก Dense(1) เป็น Dense(k) เมื่อ k คือจำนวนชั่วโมงที่ต้องการทำนาย

เซลล์ที่ 14: ตัวอย่างทำนายแบบหลายก้าวด้วยวิธีทำนายซ้ำ
def forecast_multi(seed_window, steps=12):
win = list(seed_window) # อินพุตเริ่มต้น 24 ค่า
preds = []
for _ in range(steps):
x = np.array(win[-W:]).reshape(1, W)
yhat = model.predict(x, verbose=0)[0, 0]
preds.append(yhat)
win.append(yhat) # ป้อนค่าที่ทำนายกลับเข้าไป
return np.array(preds) * sd + mu
future = forecast_multi(X_test[0], steps=12)
print('พยากรณ์ 12 ชม.ข้างหน้า (°C):', np.round(future, 1))

ข้อสังเกตสำคัญคือ ยิ่งทำนายไกลขึ้น ความแม่นยำยิ่งลดลง เพราะความคลาดเคลื่อนสะสมจากการป้อนค่าที่ทำนาย (ซึ่งมีความผิดพลาดอยู่แล้ว) กลับเข้าไปทำนายต่อ ปรากฏการณ์นี้เป็นจริงทั้งในการพยากรณ์อากาศและการพยากรณ์ทางการเงิน และเป็นเหตุผลที่การพยากรณ์ราคาสินทรัพย์ระยะยาวจึงยากกว่าระยะสั้นมาก สอดคล้องกับที่งานวิจัยของ Leemakdej (2019) ที่เน้นกรอบเวลาสั้นเพียงไม่กี่นาทียี่หลังข่าว

กิจกรรม 1.4

จากกิจกรรม 1.3 เปลี่ยนความยาวหน้าต่าง W จาก 24 เป็น 48 หรือ 72 ชั่วโมง แล้วเปรียบเทียบ MAE

- เพิ่มจำนวน epoch หรือเปลี่ยนขั้นแรกเป็น LSTM(64) แล้วสังเกตเส้นโค้งการเรียนรู้
- ลองพยากรณ์ “ล่วงหน้าหลายชั่วโมง” แทนชั่วโมงเดียว แล้วอภิปรายว่าทำไมความแม่นยำจึงลดลงเมื่อทำนายไกลขึ้น
- ดาวน์โหลดข้อมูลราคาปิดของหุ้นหรือดัชนีที่สนใจ แล้วปรับโค้ดนี้ให้พยากรณ์ราคาวันถัดไป พร้อมอภิปรายข้อจำกัด

พยากรณ์ราคาหุ้นจริง แล้วค้นหา “กัปเดต”

ของอนุกรมไม่นิ่ง (Non-Stationary) ด้วยตัวเอง

เทคนิคพยากรณ์อนุกรมเวลาแบบเดียวกับตัวอย่างอุณหภูมิ นำมาใช้กับราคาหุ้นได้ แต่จะเจอ “กัปเดต” สำคัญที่ผู้เริ่มต้นพลาดกันบ่อย เราจะลองสองแบบบนข้อมูลจริงของหุ้น GOOGL (ราคาปิดปรับปันผลรายวัน เตรียมไว้ให้แล้วในไฟล์ `googl_daily.csv` ซึ่งดึงจากผู้ให้บริการข้อมูลการเงินรายเดียวกับที่จะใช้ในบทที่ 4) คือ แบบ ก ทำนายราคาโดยตรง และ แบบ ข แปลงเป็นอัตราผลตอบแทนก่อนแล้วจึงทำนาย ขอให้คุณรันทั้งสองแบบ สังเกตผล แล้วตัดสินใจด้วยตัวเองว่าแบบใดน่าจะเชื่อถือกว่าและเพราะอะไร โดยยังไม่ต้องด่วนสรุป (เฉลยอยู่ท้ายกิจกรรม)



ที่เก็บไฟล์ข้อมูล/โปรแกรม สำหรับหนังสือเล่มนี้

แม้ตำราเล่มนี้จะสนับสนุนให้ผู้เรียนป้อนคำสั่งภาษาคอมพิวเตอร์ด้วยตนเอง แต่เพื่อความสะดวกหรือเพื่อการตรวจสอบ ผู้เรียนสามารถดาวน์โหลดไฟล์โปรแกรมตัวอย่าง ตลอดจนไฟล์ข้อมูลที่เกี่ยวข้องได้จาก <http://tinyurl.com/AgenticFinance> ซึ่งจะมีการแบ่งไฟล์เดอร์แยกแต่ละบท

กุญแจอยู่ที่คำว่า “อนุกรมนิ่ง” (Stationary) คืออนุกรมที่คุณสมบัติทางสถิติ เช่น ค่าเฉลี่ยและความแปรปรวน คงที่ตามเวลา โดยทั่วไปแบบจำลองเรียนรู้ได้ดีกับอนุกรมนิ่งมากกว่า ก่อนลงมือ ลองตั้งคำถามกับตัวเองว่า ระหว่าง “ราคาหุ้น” กับ “อัตราผลตอบแทน

รายวัน” (Return ซึ่งคือการเปลี่ยนแปลงของราคาในแต่ละวัน) อันไหนน่าจะใกล้เคียง
อนุกรมหนึ่งมากกว่า และการเลือกพยากรณ์ตัวใดจะให้ผลที่น่าเชื่อถือกว่า เก็บคำตอบไว้
ในใจ แล้วไปพิสูจน์กับข้อมูลจริงในสองแบบด้านล่าง

หมายเหตุสำหรับ Colab: ก่อนรัน ให้อัปโหลดไฟล์ googl_daily.csv เข้าสู่ Colab
โดยคลิกไอคอนไฟล์เดอร์ (Files) ทางซ้าย แล้วลากไฟล์เข้าไป หรือใช้เมนูอัปโหลด จากนั้น
จึงรันเซลล์ด้านล่าง

```
# เซลล์: โหลดข้อมูลราคาหุ้นจริง (GOOGL)
import pandas as pd, numpy as np
df = pd.read_csv('googl_daily.csv', parse_dates=['date']).sort_values('date')
price = df['adj_close'].values.astype('float32')
print('จำนวนวัน:', len(price), '|', df['date'].min().date(), 'ถึง',
      df['date'].max().date())

# เซลล์: ดูข้อมูลด้วยตา — สังเกตว่าราคาเคลื่อนไหวอย่างไรตามเวลา
import matplotlib.pyplot as plt
plt.figure(figsize=(10,3)); plt.plot(df['date'], price)
plt.title('ราคาปิดปรับ GOOGL รายวัน'); plt.ylabel('USD'); plt.show()
```

แบบ ก: ทำนาย “ราคา” โดยตรง

วิธีที่ดีเป็นธรรมชาติที่สุดคือ ใช้ราคา 20 วันย้อนหลังทำนายราคาวันถัดไป โดยทำให้
เป็นมาตรฐานดี ย่กว่าจากชุดฝึกเท่านั้น (กันการรั่วไหลของข้อมูลอนาคต) แล้วเทียบกับ
“เส้นฐาน” ที่ง่ายที่สุดคือ การทำนายว่า **“ราคาพรุ่งนี้เท่ากับราควันนี้”**

```
# เซลล์: แบบ ก — ทำนายราคาโดยตรง
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Input, Dense, Dropout

W = 20
def make_windows(s, w=W):
    X = [s[i:i+w] for i in range(len(s)-w)]
    y = [s[i+w] for i in range(len(s)-w)]
    return np.array(X), np.array(y)
```

<code>i_tr = int(len(price)*0.8)</code>
<code>mu, sd = price[:i_tr].mean(), price[:i_tr].std()</code> <code># ใช้ค่าจากชุดฝึกเท่านั้น</code>
<code>p_n = (price - mu) / sd</code>
<code>X, y = make_windows(p_n)</code>
<code>split = i_tr - W</code>
<code>X_tr, y_tr, X_te, y_te = X[:split], y[:split], X[split:], y[split:]</code>
<code>model = Sequential([Input((W,)), Dense(64,'relu'), Dense(32,'relu'),</code>
<code>Dropout(0.2), Dense(1)])</code>
<code>model.compile('adam', 'mse', metrics=['mae'])</code>
<code>model.fit(X_tr, y_tr, validation_split=0.1, epochs=20, batch_size=32,</code>
<code>verbose=0)</code>
<code>pred = model.predict(X_te, verbose=0).flatten()</code>
<code>mae_model = np.mean(np.abs(pred - y_te)) * sd</code>
<code>mae_naive = np.mean(np.abs(X_te[:, -1] - y_te)) * sd</code> <code># เส้นฐาน: พรุ่งนี้ = วันนี้</code>
<code>print('MAE โมเดล = %.2f USD MAE เส้นฐาน(เมื่อวาน) = %.2f USD' % (mae_model,</code>
<code>mae_naive))</code>

ตัวอย่างผลลัพธ์: (ตัวเลขอาจต่างเล็กน้อยในแต่ละรอบ): MAE โมเดล 3.98 USD ขณะที่ MAE เส้นฐาน 3.81 USD เมื่อวาดกราฟค่าทำนายที่บราคจริง เส้นจะ **“แนบสนิท”** และค่าผิดพลาดราว 4 ดอลลาร์บนหุ้นราคาหลายร้อยดอลลาร์ดูน้อยมากจนเหมือนแบบจำลองแม่นยำ จดค่า MAE ทั้งสอง (โมเดลและเส้นฐาน) ไว้เปรียบเทียบกับแบบ ข แล้วลองตอบในใจว่าแบบจำลองเอาชนะเส้นฐาน **“พรุ่งนี้เท่ากับวันนี้”** ได้หรือไม่

แบบ ข: แปลงเป็น “อัตราผลตอบแทน” ก่อนแล้วจึงทำนาย
 แทนที่จะทำนายราคา เราแปลงเป็นผลตอบแทนแบบ log ก่อน คือ $r_t = \ln(P_t / P_{t-1})$ ซึ่งมักมีพฤติกรรมทางสถิติต่างจากราคา แล้วใช้ผลตอบแทน 20 วันย้อนหลังทำนายผลตอบแทนวันถัดไป คราวนี้เส้นฐานที่เหมาะสมคือ **“ทำนายว่าผลตอบแทนวันถัดไปเท่ากับศูนย์”** และเราเพิ่มตัวชี้วัด **“ความแม่นยำทาง”** คือทายถูกไหมว่าวันถัดไปขึ้นหรือลง

เซลล์: แบบ ข — แปลงเป็นผลตอบแทน (log return) แล้วค่อยทำนาย
ret = np.diff(np.log(price)).astype('float32') # log return รายวัน
i_tr2 = int(len(ret)*0.8)
m2, s2 = ret[:i_tr2].mean(), ret[:i_tr2].std()
r_n = (ret - m2) / s2
Xr, yr = make_windows(r_n)
sp = i_tr2 - W
Xr_tr, yr_tr, Xr_te, yr_te = Xr[:sp], yr[:sp], Xr[sp:], yr[sp:]
m = Sequential([Input((W,)), Dense(64,'relu'), Dense(32,'relu'), Dropout(0.2), Dense(1)])
m.compile('adam', 'mse', metrics=['mae'])
m.fit(Xr_tr, yr_tr, validation_split=0.1, epochs=20, batch_size=32, ver- bose=0)
pr = m.predict(Xr_te, verbose=0).flatten() * s2 + m2
tr = yr_te * s2 + m2
mae_ret = np.mean(np.abs(pr - tr))
mae_zero = np.mean(np.abs(tr)) # เส้นฐาน: ผลตอบแทน = 0
dir_acc = np.mean(np.sign(pr) == np.sign(tr)) * 100 # ความแม่นยำทิศทาง ขึ้น/ลง
print('MAE โมเดล = %.5f MAE เส้นฐาน(0) = %.5f ทิศทางถูก = %.1f%%' % (mae_ret, mae_zero, dir_acc))

ตัวอย่างผลลัพธ์: MAE โมเดล 0.0136 ขณะที่ MAE เส้นฐาน(0) 0.0133 และ ความแม่นยำทิศทาง 51.8% บันทึกตัวเลขเหล่านี้ไว้ แล้วไปตอบคำถามท้ายกิจกรรม

คำถามท้ายกิจกรรม (ลองตอบก่อนเปิดเฉลย)

- 1) แบบ ข ให้ค่า MAE (0.01) ต่ำกว่าแบบ ก (4) มาก สรุปได้หรือไม่ว่าแบบ ข แม่นยำกว่า เพราะอะไร
- 2) ในแต่ละแบบ แบบจำลองเอาชนะ “เส้นฐาน” ได้หรือไม่ (เส้นฐานของแบบ ก คือ “พรั้งนี้เท่ากับวันนี้” ส่วนของแบบ ข คือ “ผลตอบแทนพรั้งนี้เท่ากับศูนย์”)
- 3) ระหว่างราคากับอัตราผลตอบแทน อันไหนใกล้เคียงอนุกรมหนึ่งมากกว่า และ เหตุใดจึงควรพยากรณ์ตัวที่หนึ่งกว่า

เฉลยและบทเรียนสำคัญ

ข้อ 1: สรุปแบบนั้นไม่ได้ เพราะ MAE ของสองแบบอยู่คนละหน่วย - แบบ ก วัดเป็นดอลลาร์ (4) ส่วนแบบ ข วัดเป็นอัตราผลตอบแทน (0.01) ตัวเลข 0.01 ที่ดูต่ำกว่าจึงไม่ได้แปลว่าแบบ ข แม่นยำกว่า สิ่งที่ถูกต้องคือเทียบแต่ละแบบกับ “**เส้นฐาน**” ของตัวเอง ในแบบ ก โมเดลได้ MAE 3.98 แต่เส้นฐาน “**พรงนี้เท่ากับวันนี้**” ได้ 3.81 (ต่ำกว่า) แปลว่าโมเดลยังเอาชนะเส้นฐานไม่ได้ ทั้งที่ค่า 4 ดอลลาร์บนหุ้นราคาหลายร้อยดอลลาร์ดู “น้อย” จนหลอกตา สาเหตุคือราคาดังนี้เป็นอนุกรมไม่นิ่ง (Non-Stationary) คล้ายการเดินสุ่ม (Random Walk) โมเดลจึงได้แค่ “**ลอคค่าล่าสุด**”

ข้อ 2-3: “**อัตราผลตอบแทน**” (log return) ใกล้เคียงอนุกรมนิ่ง (Stationarity) มากกว่าราคา จึงเหมาะนำมาพยากรณ์ แต่เมื่อทำกฎวิธีในแบบ ข กลับพบว่าแบบจำลองแทบเอาชนะเส้นฐาน “**ผลตอบแทนเท่ากับศูนย์**” (MAE 0.0136 เทียบ 0.0133) ไม่ได้ และทายทิศทางถูก 51.8% พอๆ กับการเดาสุ่ม สอดคล้องกับสมมติฐานตลาดมีประสิทธิภาพรูปแบบอ่อน (Weak-form EMH) ในหัวข้อ 1.3 ที่ว่าราคาในอดีตเพียงอย่างเดียวทำนายผลตอบแทนในอนาคตได้ยากมาก

กับดักที่ต้องระวัง: (1) ค่าความผิดพลาดต่ำไม่ได้แปลว่าแบบจำลองเก่ง ต้องเทียบกับเส้นฐานเสมอ (2) ข้อมูลการเงินมักไม่นิ่ง ควรแปลงเป็นผลตอบแทนให้นิ่งก่อนพยากรณ์

1.8

ข้อจำกัดของ Generative AI และการแก้ไขด้วย RAG

Generative AI สามารถช่วยงานได้มาก แต่ผู้ใช้งานการเงินต้องเข้าใจข้อจำกัดของมันอย่างถ่องแท้ เพราะการตัดสินใจทางการเงินมีเดิมพันสูง ข้อจำกัดเหล่านี้ไม่ใช่ “**ข้อบกพร่อง**” ที่จะหายไปเอง แต่เป็นผลโดยตรงจากวิธีที่แบบจำลองทำงาน คือทำนายโทเคนถัดไปจากรูปแบบทางภาษา ตามที่อธิบายในหัวข้อ 1.5

1.8.1 ข้อจำกัดสำคัญที่ต้องระวัง

เพื่อให้เห็นภาพชัดเจน เราจะอธิบายข้อจำกัดแต่ละข้อพร้อมตัวอย่างในบริบทการเงิน เริ่มจากปัญหาที่อันตรายที่สุดสำหรับงานการเงิน นั่นคือการก่อกวนข้อมูล ตามด้วยข้อจำกัดด้านอื่นๆ

- ข้อมูลหลอน (Hallucination): แบบจำลองอาจสร้างข้อมูลที่ฟังดูน่าเชื่อถือและสมเหตุสมผล แต่เป็นเท็จ เช่น ตัวเลขกำไรสุทธิ ชื่อรายงาน หรือมาตราของกฎหมายที่ไม่มีอยู่จริง เพราะมันทำนายคำที่ **“น่าจะตามมา”** ไม่ได้ตรวจสอบข้อเท็จจริง
- ข้อมูลหยุดนิ่งตามเวลาฝึก (Knowledge Cutoff): แบบจำลองรู้เฉพาะข้อมูลถึงช่วงที่ฝึก จึงไม่ทราบราคาล่าสุด ประกาศของหน่วยงานกำกับล่าสุด หรือเหตุการณ์ที่เพิ่งเกิด
- ไม่เชื่อมกับข้อมูลภายในองค์กร: โดยลำพังแบบจำลองไม่รู้จักรงบการเงิน นโยบายสินเชื่อ หรือข้อมูลลูกค้าเฉพาะของสถาบันคุณ จึงตอบคำถามเฉพาะองค์กรไม่ได้หากไม่ป้อนข้อมูลให้
- ความน่าเชื่อถือของการคำนวณและการอ้างอิง: แบบจำลองภาษาไม่ใช่เครื่องคิดเลข อาจคำนวณดอกเบี้ยทบต้นหรือ NPV ผิด และอาจอ้างแหล่งข้อมูลที่ฟังดูจริงแต่ไม่มีอยู่
- อคติและความเป็นส่วนตัว: แบบจำลองอาจสะท้อนอคติที่ฝังอยู่ในข้อมูลฝึก ซึ่งอ่อนไหวมากในงานเช่นการอนุมัติสินเชื่อ อีกทั้งการป้อนข้อมูลส่วนบุคคลหรือข้อมูลลับเข้าสู่บริการภายนอกยังมีความเสี่ยงด้านความเป็นส่วนตัว
- หน้าต่างบริบท (Context Window) มีขนาดจำกัด: แบบจำลองรับข้อความเข้าได้จำกัด จึงไม่สามารถ **“อ่าน”** เอกสารยาวมากๆ ทั้งหมดในคราวเดียว ต้องใช้เทคนิคแบ่งชิ้นและสืบค้นมาช่วย

รูปที่ 1.28 แสดงปัญหา Hallucination อย่างเป็นรูปธรรม เมื่อถามถึงตัวเลขกำไรของบริษัทหนึ่ง แบบจำลองที่ไม่ได้เชื่อมกับข้อมูลจริงอาจ **“แต่ง”** ตัวเลขขึ้นมาตอบอย่างมั่นใจ

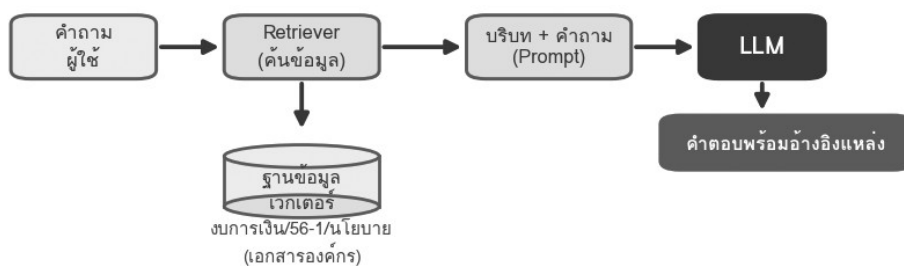


รูปที่ 1.28 ปัญหา Hallucination: โมเดลตอบมั่นใจแต่ข้อมูลผิดเพราะไม่ได้เปิดดูข้อมูลจริง

1.8.2 RAG: ให้คำตอบยึดกับเอกสารจริง

เทคนิคที่ใช้แก้ปัญหาข้อมูลหลอนได้อย่างมีประสิทธิภาพคือ RAG (Retrieval-Augmented Generation) หรือ “การสร้างคำตอบโดยเสริมด้วยการสืบค้น” แนวคิดหลักคือ แทนที่จะให้แบบจำลองตอบจากความจำล้วนๆ เราให้มัน “ค้นเอกสารที่เกี่ยวข้องก่อน” แล้วจึงสร้างคำตอบโดยอ้างอิงเอกสารเหล่านั้น เปรียบเหมือนการเปลี่ยนจากข้อสอบ “ปิดตำรา” เป็น “เปิดตำรา”

กระบวนการ RAG มีขั้นตอนหลักดังนี้ เริ่มจากแปลงเอกสารองค์กร เช่น งบการเงินแบบ 56-1 และนโยบายภายใน ให้เป็นเวกเตอร์แล้วจัดเก็บใน “ฐานข้อมูลเวกเตอร์” (Vector Database) เมื่อมีคำถามเข้ามา ระบบ “ตัวสืบค้น” (Retriever) จะค้นหาส่วนของเอกสารที่ใกล้เคียงคำถามมากที่สุด นำมาประกอบเป็นบริบทร่วมกับคำถาม แล้วส่งให้ LLM สร้างคำตอบที่อ้างอิงจากเอกสารจริง ดังรูปที่ 1.29

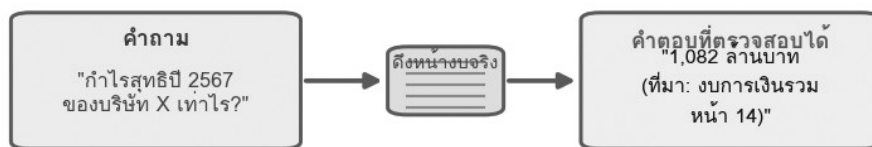


รูปที่ 1.29 สถาปัตยกรรม RAG: ค้นเอกสารที่เกี่ยวข้องก่อน แล้วจึงให้ LLM สร้างคำตอบ

ลองดูตัวอย่างที่เป็นรูปธรรม สมมติว่านักวิเคราะห์ถามว่า “บริษัทมีนโยบายจ่ายเงินปันผลอย่างไร และปีล่าสุดจ่ายเท่าไร” ในระบบ RAG ที่ดี ตัวสืบค้นจะไปดึงย่อหน้าเรื่องนโยบายเงินปันผลจากแบบ 56-1 และตารางการจ่ายปันผลจากงบการเงินมาเป็นบริบท จากนั้น LLM จะเรียบเรียงคำตอบโดยอ้างอิงข้อมูลทั้งสองส่วน พร้อมระบุที่มา ผลคือคำตอบที่ทั้งครบถ้วนและตรวจสอบได้ ต่างจากการถาม LLM เปล่าๆ ที่อาจได้ตัวเลขที่ฟังดูสมเหตุสมผลแต่ไม่ตรงกับเอกสารจริง

ผลลัพธ์คือคำตอบที่ “ตรวจสอบย้อนกลับได้” เพราะอ้างอิงถึงแหล่งที่มา เช่น เลขหน้าของงบการเงิน ทำให้ลดการก่อกวนข้อมูลลงอย่างมาก และยังแก้ปัญหาข้อมูลหลุดนิ่งตามเวลาฝึกได้ด้วย เพราะเราอัปเดตฐานเอกสารได้ตลอดเวลาโดยไม่ต้องฝึกแบบจำลองใหม่ รูปที่ 1.30 แสดงตัวอย่างการใช้ RAG ตอบคำถามจากงบการเงินจริง

RAG ช่วยตอบคำถามจากเอกสารจริง



ลดการแต่งข้อมูล เพราะให้โมเดลอ้างอิงเอกสารต้นฉบับ

รูปที่ 1.30 ตัวอย่าง RAG ในงานการเงิน: ตอบคำถามพร้อมอ้างอิงหน้าของงบการเงิน

เบื้องหลังของ RAG มีรายละเอียดที่ผู้ใช้งานควรเข้าใจเพื่อให้ระบบทำงานได้ดี ขั้นแรกคือการ **“แบ่งเอกสารเป็นชิ้น”** (Chunking) เพราะเอกสารการเงินยาวเกินกว่าจะใส่ทั้งหมดในหน้าต่างบริบทของแบบจำลอง การแบ่งชิ้นที่ดีควรรักษาความต่อเนื่องของเนื้อหา เช่น ไม่ตัดกลางตารางหรือกลางย่อหน้าสำคัญ ขั้นที่สองคือการแปลงแต่ละชิ้นเป็นเวกเตอร์ด้วยแบบจำลอง Embedding ซึ่งใช้หลักการเดียวกับ word2vec แต่ทำในระดับประโยคหรือย่อหน้า เพื่อให้ค้นหาด้วยความหมายได้ ไม่ใช่แค่จับคู่คำตรงตัว

เมื่อมีคำถามเข้ามา ระบบจะแปลงคำถามเป็นเวกเตอร์เช่นกัน แล้ววัดความใกล้เคียงกับเวกเตอร์ของทุกชิ้นในฐานข้อมูล เพื่อเลือกชิ้นที่เกี่ยวข้องที่สุดมาประกอบเป็นบริบทคุณภาพของคำตอบจึงขึ้นกับสองสิ่งสำคัญ คือคุณภาพของการค้น (Retrieval) และคุณภาพของเอกสารต้นทาง หากค้นผิดชิ้น หรือเอกสารต้นทางมีข้อมูลผิด คำตอบก็จะผิดตามไปด้วย แม้แบบจำลองภาษาจะเก่งเพียงใด

ด้วยเหตุนี้ การ **“ประเมินผล”** ระบบ RAG จึงสำคัญมาก เราควรตรวจสอบทั้งว่าระบบค้นเอกสารที่ถูกต้องมาได้หรือไม่ และคำตอบสุดท้ายตรงกับเอกสารที่อ้างอิงจริงหรือไม่ ในงานการเงินที่มีเดิมพันสูง การให้ระบบ **“แสดงแหล่งอ้างอิง”** ทุกครั้งเป็นแนวปฏิบัติที่ดี เพราะช่วยให้ผู้ใช้ตรวจสอบย้อนกลับได้ และเพิ่มความรับผิดชอบต่อคำตอบ

อย่างไรก็ตาม RAG ไม่ใช่ยาครอบจักรวาล มันช่วยเรื่องการเข้าถึงข้อมูลที่ต้องการ แต่ไม่ได้แก้ปัญหายุ่งยาก เช่น หากคำถามต้องการการคำนวณที่ซับซ้อน เราควรให้แบบจำลองเรียกใช้ **“เครื่องมือ”** คำนวณแทนการให้มันคำนวณเอง หากงานต้องการรูปแบบเฉพาะขององค์กรมากๆ การปรับแต่ง (Fine-tuning) อาจเหมาะกว่า และไม่ว่าจะใช้เทคนิคใด การมีมนุษย์ตรวจสอบในงานที่มีเดิมพันสูงยังคงจำเป็นเสมอ ตารางที่ 1.5 สรุปวิธีบรรเทาข้อจำกัดและความเหมาะสมของแต่ละวิธี

ตารางที่ 1.5 ตารางเปรียบเทียบวิธีแก้ข้อจำกัดของ Generative AI

วิธีบรรเทา	เหมาะกับปัญหา	ข้อสังเกต
RAG (เสริมด้วยการสืบค้น)	กู้ข้อมูล/ข้อมูลล่าสุด /เอกสารองค์กร	คุณภาพขึ้นกับการค้น และคุณภาพเอกสาร
เรียกใช้เครื่องมือ (Tools)	การคำนวณ ดึงราคาเรียลไทม์	ต้องออกแบบสิทธิ์และความปลอดภัย
การปรับแต่ง (Fine-tuning)	รูปแบบงาน/น้ำเสียงเฉพาะองค์กร	ใช้ข้อมูลและทรัพยากรมาก
มนุษย์ตรวจสอบ (Human Review)	งานที่มีเดิมพันสูงทุกประเภท	จำเป็นเสมอในงานการเงิน

กิจกรรม 1.5

ถ้าผู้บริหารถามเหตุผลของธนาคารว่า “ยอดสินเชื่อกำลังของลูกค้ายานี้ **เดือนนี้เท่าไร**” เพราะเหตุใด RAG จึงจำเป็น และข้อมูลใดบ้างที่ต้องอยู่ในฐานข้อมูลเวกเตอร์ ลองออกแบบขั้นตอนการทำงานคร่าวๆ

สรุปท้ายบท

บทที่ 1 ได้พาผู้เรียนเห็นภาพรวมของวิวัฒนาการ AI เริ่มจากนิเวศน์ที่ยึมนหน่วยเดียว ขยายเป็นโครงข่ายเชิงลึก เรียนรู้การแทนความหมายของคำด้วย word2vec ผ่านกรณีศึกษา การพยากรณ์ราคาหุ้นไทย ต่อยอดสู่ LLM และหลักการทำงานของมัน สำรวจการประยุกต์ ในงานการเงิน ลงมือสร้างแบบจำลองพยากรณ์ด้วย TensorFlow ด้วยตนเอง เข้าใจข้อจำกัด และปิดท้ายการแก้ปัญหาข้อมูลหลอนที่ Generative AI แต่งขึ้นเองด้วย RAG ผู้เรียนที่เข้าใจ ภาพรวมนี้จะมีรากฐานที่มั่นคงสำหรับการเจาะลึกเครื่องมือและโครงงานในบทต่อไป ไป ขอย้ำหลักการสำคัญสามข้อที่จะเป็นเข็มทิศตลอดเล่ม

- ข้อแรก เทคโนโลยีเหล่านี้เป็นเครื่องมือขยายศักยภาพของมนุษย์ ไม่ใช่สิ่งทดแทนวิจารณญาณ
- ข้อสอง ในงานการเงินที่มีเดิมพันสูง การตรวจสอบความถูกต้องและการมีมนุษย์กำกับเป็นสิ่งที่ไม่ได้
- ข้อสาม การเข้าใจ “**หลักการทำงาน**” ของเครื่องมืออย่างถ่องแท้ คือสิ่งที่ทำให้เราเชื่อมั่นได้อย่างปลอดภัยและสร้างสรรค์ เหนือกว่าการใช้ตามกระแสอย่างผิวเผิน

กิจกรรมท้ายบท

คำถามต่อไปนี้ออกแบบเพื่อวัดความเข้าใจในแนวคิดหลักของบท ลองตอบด้วยความเข้าใจของตนเองก่อน แล้วจึงเทียบกับแนวตอบที่ให้ไว้

- 1 อธิบายความเชื่อมโยงจากนิเวศวิทยาหนึ่งหน่วย ไปสู่ Deep Learning และ Generative AI ว่าแต่ละชั้นต่อยอดกันอย่างไร พร้อมยกตัวอย่างเหตุการณ์สำคัญในเส้นเวลา

แนวตอบ

นิเวศวิทยาคำนวณ $y = f(\sum w_i x_i + b)$ เมื่อนำมาเรียงหลายชั้นและฝึกด้วย Backpropagation จะได้โครงข่ายเชิงลึกที่เรียนรู้รูปแบบซับซ้อนได้ (Deep Learning) เมื่อขยายขนาดและใช้สถาปัตยกรรม Transformer ที่มีกลไก attention จึงเกิดแบบจำลองที่ “สร้าง” ข้อมูลใหม่ได้ (Generative AI) เหตุการณ์สำคัญ เช่น Perceptron (1958), Backpropagation (1986), Deep Learning ชนะ ImageNet (2012), Transformer (2017), และยุค GPT/BERT (2018 เป็นต้นมา)

- 2 word2vec ทำให้คอมพิวเตอร์ “เข้าใจความหมายของคำ” ได้อย่างไร และในกรณีศึกษาการพยากรณ์ราคาหุ้นไทย ผู้วิจัยนำ word2vec ไปใช้ร่วมกับโครงข่ายเชิงลึกอย่างไร

แนวตอบ

word2vec แทนคำด้วยเวกเตอร์หลายมิติโดยอาศัยสมมติฐานเชิงการกระจาย คำที่อยู่ในบริบทคล้ายกันจะมีเวกเตอร์ใกล้เคียงกัน ในกรณีศึกษาของ Leemakdej (2016) ผู้วิจัยใช้ Skip-gram (คำรอบข้าง 5 คำ เวกเตอร์ 100 มิติ) แปลงพาดหัวข่าว 15 คำเป็นอินพุต 1,500 คำ จัดรูปเป็นเทนเซอร์ $10 \times 10 \times 15$ แล้วป้อนเข้า CNN 8 ชั้นเพื่อทำนายทิศทางราคาหุ้นสภาพคล่องสูง 10 ตัวใน SET ผลพบว่ากลยุทธ์ซื้อขายภายใน 5 นาทีหลังข่าวให้ผลตอบแทนรวมราว 18.2% ต่อวัน บ่งชี้ว่าตลาดอาจไม่ได้มีประสิทธิภาพระดับที่แข็งแกร่งในกรอบเวลาดังนั้น

3 เหตุใด LLM จึงเข้าใจบริบทได้ดีกว่า word2vec จงอธิบายบทบาทของกลไก Self-attention และความหมายของเวกเตอร์ตามบริบท

แนวตอบ

word2vec ให้เวกเตอร์คงที่หนึ่งเวกเตอร์ต่อคำ ไม่ว่าบริบทใด จึงแยกความกำกวมไม่ได้ ส่วน LLM ใช้ Self-attention ให้ทุกคำในประโยคให้น้ำหนักความสัมพันธ์กับคำอื่นได้พร้อมกัน ทำให้สร้าง “เวกเตอร์ตามบริบท” ที่เปลี่ยนตามคำแวดล้อม คำเดียวกันจึงมีความหมายต่างกันได้ตามประโยค ทำให้เข้าใจข้อความซับซ้อนอย่างรายงานการเงินได้ดีขึ้น

4 ในการสร้างแบบจำลองพยากรณ์อนุกรมเวลาด้วย TensorFlow เหตุใดจึงต้องแบ่งข้อมูลตามลำดับเวลา และเหตุใดจึงต้องคำนวณค่าทำให้เป็นมาตรฐานจากชุดฝึกเท่านั้น

แนวตอบ

ต้องแบ่งตามเวลา (อดีตฝึก อนาคตทดสอบ) เพื่อประเมินว่าทำนายอนาคตที่ยังไม่เคยเห็นได้จริงหรือไม่ ถ้าสับสลับ แบบจำลองอาจ “แอบเห็นอนาคต” ทำให้ผลดีเกินจริง ส่วนการคำนวณค่าเฉลี่ย/ส่วนเบี่ยงเบนจากชุดฝึกเท่านั้น ก็เพื่อป้องกัน Data Leakage ไม่ให้ข้อมูลจากชุดทดสอบ (อนาคต) รั่วไหลเข้าสู่กระบวนการฝึก ซึ่งเป็นหลักการสำคัญอย่างยิ่งในงานพยากรณ์ทางการเงิน

หนังสือและเอกสารอ้างอิง

- Abadi, M., et al. (2016). TensorFlow: A System for Large-Scale Machine Learning. USENIX OSDI.
- Fama, E. F. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. *Journal of Finance*, 25(2), 383-417.
- Google. (n.d.). Google Colaboratory. Rsetrieved from <https://colab.research.google.com>
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.
- Leemakdej, Arnat (2019). Stock Price Prediction with Deep Learning. *วารสารบริหารธุรกิจ*, 42(163), 23-32.
- Lewis, P., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *NeurIPS*.
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *ICLR Workshop*.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *JMLR*, 15(1), 1929-1958.
- Temsamani, D. (2025). *The Agentic Bank: How AI and Intelligent Systems Are Redefining Finance*. (ISBN 9798899655616).
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, ., & Polosukhin, I. (2017). Attention Is All You Need. *NeurIPS*.

ตำรา **Agentic AI สำหรับนักการเงิน** ที่ ศ.ดร.อาณัติ สัมภคเดช พัฒนาขึ้น แก้ Pain Point นี้ เพราะเป็นการบูรณาการความรู้ทางด้าน AI โดยชี้ให้เห็นพัฒนาการเป็นลำดับขั้นตั้งแต่การเขียน Prompt ใน Generative AI ไปจนถึงการสร้าง Workflow ของทีม Multi-Agent ใน Agentic AI พร้อมกันนี้ยังได้พัฒนาโครงงานขึ้นมาเป็นภาคปฏิบัติ เพื่อให้ผู้เรียนได้ทดลองพัฒนาโครงการจริง โดยเน้นการต่อยอดความรู้ที่ละขั้น

รศ.ดร.สุรัตน์ ทิรชาภิบาล

คณบดี คณะพาณิชยศาสตร์และการบัญชี มหาวิทยาลัยธรรมศาสตร์

หนังสือ **Agentic AI สำหรับนักการเงิน** เล่มนี้ มาถูกเวลา เพราะไม่ได้อธิบายเพียงแนวคิดของ AI แต่ค่อยๆ พาผู้อ่านตั้งแต่พื้นฐานของ Generative AI ไปจนถึงการออกแบบ AI Agent, Multi-Agent, Workflow Automation และโครงงานที่สามารถนำไปประยุกต์ใช้กับงานการเงินจริงได้อย่างเป็นระบบ จึงเหมาะทั้งสำหรับผู้เริ่มต้นศึกษา และผู้ที่ต้องการนำ AI ไปสร้างคุณค่าให้กับองค์กร

ผศ.ดร.กรินทร์ บุญเลิศวนิชย์

รองผู้จัดการใหญ่ ธนาคารกรุงไทย

ในเล่มนี้คุณจะได้เรียนรู้

- 15 บท จากโครงข่ายประสาทเทียมและ Generative AI สู่ทีม Multi-Agent
- โครงงานภาคปฏิบัติ 6 โครงงาน: วิเคราะห์งบการเงินและ MD&A · NPV/IRR · ส่งคำสั่งซื้อขายผ่าน n8n · สินเชื่อเคหะ · วิเคราะห์หุ้น · นโยบายเงินปันผล
- กรอบธรรมาภิบาล การตรวจสอบซ้ำ และ Human-in-the-loop กำกับทุกโครงงาน
- แนวทาง Low-Code ด้วย n8n, OpenClaw, Claude และ Ollama พร้อมไฟล์ประกอบดาวน์โหลดได้ - ไม่จำเป็นต้องเป็นโปรแกรมเมอร์

เหมาะสำหรับนักศึกษา MBA และการเงิน นักวิเคราะห์ ผู้ตรวจสอบ และผู้บริหารสถาบันการเงิน

ISBN 978-616-92877-1-1



9 786169 287711

ราคา 790 บาท