โครงงานคณิตศาสตร์

เรื่อง

การทำนายราคาทองด้วย Tensorflow และโครงข่ายประสาทเทียม LSTM

Gold Price Prediction Using TensorFlow and Long Short-Term Memory (LSTM) Neural Network

นำเสนอโดย

นายนันทิพัฒน์ ชลภิญโญสกุล รหัสนักศึกษา 6604105312

นางสาวพิชญา สารมะโน รหัสนักศึกษา 6604105313

นางสาวเพชรรัตน์ ขวัญหวัน รหัสนักศึกษา 6604105314

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา 10305492 โครงงาน

ภาคเรียนที่ 2 ปีการศึกษา 2568

สาขาวิชาคณิตศาสตร์ คณะวิทยาศาสตร์

มหาวิทยาลัยแม่โจ้

โครงงานคณิตศาสตร์
เรื่อง
การทำนายราคาทองด้วย Tensorflow และโครงข่ายประสาทเทียม LSTM

Gold Price Prediction Using TensorFlow and Long Short-Term Memory (LSTM) Neural Network

นำเสนอโดย

นายนันทิพัฒน์ ชลภิญโญสกุล	รหัสนักศึกษา 6604105312
นางสาวพิชญา สารมะโน	รหัสนักศึกษา 6604105313
นางสาวเพชรรัตน์ ขวัญหวัน	รหัสนักศึกษา 6604105314

..... อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์ ดร.บุรัสกร นันทดิลก)

..... อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์ ดร.อุไรลักษมณ์ สิงห์ทอง)

ชื่อ–สกุล นายนันทิพัฒน์ ชลภิญโญสกุล รหัสนักศึกษา 6604105312
นางสาวพิชญา สารมะโน รหัสนักศึกษา 6604105313
นางสาวเพชรรัตน์ ขวัญหวัน รหัสนักศึกษา 6604105314

อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร.บุรัสกร นันทดิลก
ผู้ช่วยศาสตราจารย์ ดร.อุไรลักษมณ์ สิงห์ทอง

หัวข้อโครงงาน

การทำนายราคาทองด้วย Tensorflow
และโครงข่ายประสาทเทียม LSTM

Gold Price Prediction Using TensorFlow
and Long Short-Term Memory (LSTM) Neural Network

บทคัดย่อ

งานวิจัยนี้มีวัตถุประสงค์เพื่อพัฒนาแบบจำลองสำหรับการทำนายราคาทองด้วย Tensorflow และโครงข่ายประสาทเทียม LSTM และเทคนิคด้าน Machine Learning และ Deep Learning เพื่อศึกษารูปแบบและแนวโน้มของข้อมูลราคาทองคำในอดีต ซึ่งเป็นข้อมูลประเภทอนุกรมเวลา (Time Series Data) การศึกษานี้ได้รวบรวมข้อมูลราคาทองคำย้อนหลังจากแหล่งข้อมูลที่เชื่อถือได้ จากนั้นทำการเตรียมข้อมูล (Data Preprocessing) โดยการทำความสะอาดข้อมูล การจัดรูปแบบข้อมูล และการปรับสเกลข้อมูลเพื่อให้เหมาะสมต่อการนำไปสร้างแบบจำลอง ในการพัฒนาแบบจำลองได้ใช้โครงข่ายประสาทเทียมแบบ Recurrent Neural Network (RNN) และ Long Short-Term Memory (LSTM) ซึ่งเป็นเทคนิคที่เหมาะสมสำหรับการวิเคราะห์ข้อมูลที่มีลำดับเวลา โดยแบบจำลองสามารถเรียนรู้ความสัมพันธ์ของข้อมูลในอดีตเพื่อนำมาพยากรณ์แนวโน้มราคาทองคำในอนาคต นอกจากนี้ยังใช้เทคนิค Dropout เพื่อลดปัญหาการเกิด Overfitting ของโมเดล ผลการทดลองพบว่าแบบจำลองสามารถทำนายแนวโน้มราคาทองคำได้ในระดับที่น่าพอใจ โดยทำการประเมินประสิทธิภาพของโมเดลด้วยตัวชี้วัด ได้แก่ Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) และ Mean Absolute Percentage Error (MAPE) ซึ่งช่วยแสดงให้เห็นถึงความคลาดเคลื่อนระหว่างค่าที่พยากรณ์กับค่าจริง จากผลการศึกษาแสดงให้เห็นว่าการประยุกต์ใช้ภาษา Python ร่วมกับเทคนิค Machine Learning และ Deep Learning สามารถนำมาใช้ในการวิเคราะห์และทำนายราคาทองคำได้อย่างมีประสิทธิภาพ และสามารถนำไปประยุกต์ใช้ในการวางแผนการลงทุนหรือการวิเคราะห์แนวโน้มทางเศรษฐกิจในอนาคตได้

กิตติกรรมประกาศ

โครงงานคณิตศาสตร์ เรื่อง วิธีทำนายตลาดหุ้นโดยใช้ Google Tensorflow และโครงข่ายประสาทเทียม LSTM สำเร็จลุล่วงไปได้ด้วยดี เนื่องจากได้รับความอนุเคราะห์และการสนับสนุนจากหลายฝ่าย คณะผู้จัดทำขอกราบขอบพระคุณอาจารย์ที่ปรึกษาเป็นอย่างสูง ที่ได้ให้คำแนะนำ ให้คำปรึกษา ตลอดจนตรวจสอบและแก้ไขข้อบกพร่องต่าง ๆ ด้วยความเอาใจใส่ ทำให้โครงงานฉบับนี้มีความสมบูรณ์มากยิ่งขึ้น คณะผู้จัดทำขอขอบพระคุณคณาจารย์ทุกท่านที่ได้ถ่ายทอดความรู้และประสบการณ์อันมีคุณค่า ตลอดระยะเวลาที่ศึกษา ซึ่งเป็นพื้นฐานสำคัญในการจัดทำโครงงานครั้งนี้ รวมทั้งขอขอบคุณเพื่อน ๆ และผู้ที่มีส่วนเกี่ยวข้องทุกท่านที่ได้ให้ความช่วยเหลือ ให้คำแนะนำ และให้กำลังใจจนทำให้การดำเนินงานครั้งนี้สำเร็จลุล่วงไปได้ด้วยดี

คณะผู้จัดทำหวังเป็นอย่างยิ่งว่าโครงงานฉบับนี้จะเป็นประโยชน์ต่อผู้ที่สนใจศึกษาเกี่ยวกับการวิเคราะห์ข้อมูลและการทำนายราคาทองคำ และสามารถนำไปศึกษาเพิ่มเติม หรือต่อยอดในประเด็นที่ได้เสนอแนะไว้ในโครงงานฉบับนี้ ให้บรรลุผลอย่างมีประสิทธิภาพต่อไป หากมีข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้ และขออภัยมา ณ ที่นี้

คณะผู้จัดทำ

1 บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ทองคำเป็นสินทรัพย์ที่มีความสำคัญในระบบเศรษฐกิจและการลงทุนทั่วโลก ราคาทองคำมีการเปลี่ยนแปลงอยู่ตลอดเวลาเนื่องจากปัจจัยทางเศรษฐกิจ การเมือง และตลาดการเงิน การทำนายราคาทองคำจึงเป็นเรื่องสำคัญสำหรับนักลงทุนและนักวิเคราะห์ทางการเงิน ในปัจจุบัน เทคโนโลยีด้านปัญญาประดิษฐ์ได้ถูกนำมาใช้ในการวิเคราะห์ข้อมูลจำนวนมากเพื่อช่วยในการคาดการณ์แนวโน้มของตลาด หนึ่งในเทคนิคที่ได้รับความนิยมคือการใช้โครงข่ายประสาทเทียมแบบ Long Short-Term Memory (LSTM) ซึ่งเป็นส่วนหนึ่งของโครงข่ายประสาทเทียมแบบลำดับเวลา นอกจากนี้ยังมีเครื่องมือสำหรับพัฒนาโมเดลปัญญาประดิษฐ์ เช่น TensorFlow ซึ่งเป็นไลบรารีสำหรับการสร้างและฝึกโมเดล Machine Learning และ Deep Learning ดังนั้น งานวิจัยนี้จึงมุ่งเน้นการนำเทคนิค LSTM มาประยุกต์ใช้กับ TensorFlow เพื่อพัฒนาโมเดลสำหรับทำนายราคาทองคำจากข้อมูลในอดีต

1.2 จุดประสงค์ของการศึกษา

1. เพื่อศึกษาการทำงานของโครงข่ายประสาทเทียมแบบ LSTM
2. เพื่อพัฒนาโมเดลสำหรับทำนายราคาทองคำโดยใช้ TensorFlow
3. เพื่อเปรียบเทียบผลการทำนายที่ได้จากแบบจำลองกับการเปลี่ยนแปลงของราคาจริงในอดีต

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้แบบจำลอง (Model) ที่สามารถใช้ในการทำนายราคาทองคำในระยะสั้นหรือระยะกลาง ซึ่งสามารถนำไปประยุกต์ใช้ในการตัดสินใจลงทุนได้
2. เป็นแนวทาง (Guideline) สำหรับนักลงทุนหรือผู้สนใจในการวิเคราะห์และประเมินความเสี่ยงของการลงทุนในทองคำโดยใช้หลักการทางวิทยาศาสตร์ข้อมูล
3. เพิ่มทักษะ (Skill Enhancement) ในการประยุกต์ใช้ภาษา Python และเทคนิค Machine Learning ในการแก้ปัญหางานวิจัยหรือปัญหาทางธุรกิจที่เกี่ยวข้องกับการวิเคราะห์อนุกรมเวลา (TimeSeries Data)

1.4 ขอบเขตในการศึกษา

ขอบเขตด้านข้อมูล (Data Scope)

1. ช่วงเวลาข้อมูล: กำหนดช่วงเวลาที่ชัดเจน เช่น ข้อมูลย้อนหลัง 5 ปี หรือ 10 ปี (ระบุปีเริ่มต้นและปีสิ้นสุด)
2. ข้อมูลที่ใช้: ข้อมูลราคาทองคำย้อนหลัง (เช่น ราคาเปิด-ปิด, ราคาสูงสุด-ต่ำสุด) เป็นรายวันหรือรายเดือน จากแหล่งข้อมูลที่เชื่อถือได้ (เช่น สมาคมค้าทองคำ, เว็บไซต์การเงินสากล)

ขอบเขตด้านเทคนิค (Technical Scope)

1. ภาษาโปรแกรม: ใช้ Python เป็นภาษาหลักในการพัฒนา
2. ไลบรารีหลัก: ใช้ไลบรารีที่เกี่ยวข้องกับ Data Science และ Machine Learning เช่น Pandas,NumPy, Matplotlib, Scikit-earn, หรือ TensorFlow/Keras (ขึ้นอยู่กับประเภทของแบบจำลอง) เช่น Regression หรือ Deep Learning
3. แบบจำลอง (Model): เน้นการใช้แบบจำลอง Machine Learning สำหรับการวิเคราะห์อนุกรมเวลา เช่น ARIMA, SARIMA, Prophet, หรือแบบจำลองพื้นฐานอย่าง Linear Regression, Random Forest หรือโครงข่ายประสาทเทียม (RNN/LSTM)

ขอบเขตด้านผลลัพธ์ (Output Scope)

1. ผลลัพธ์คือค่าทำนายราคาทองคำ (เช่น ราคาปิด) ในอนาคต
2. มีการแสดงผลลัพธ์เป็นกราฟเปรียบเทียบระหว่างราคาจริงกับราคาที่ทำนายได้
3. มีการประเมินประสิทธิภาพของแบบจำลองด้วยตัวชี้วัดที่เหมาะสม เช่น RMSE (Root Mean Squared Error), MAE (Mean Absolute Error) หรือ R^2 Score

1.5 สมมุติฐานการศึกษา

1. แบบจำลองการเรียนรู้ของเครื่องที่พัฒนาขึ้นโดยใช้โปรแกรม Python สามารถทำนายแนวโน้มราคาทองคำในอนาคตได้อย่างมีความแม่นยำในระดับที่ยอมรับได้ (เช่น มีค่า RMSE ต่ำกว่าเกณฑ์ที่กำหนดไว้)
2. ปัจจัยทางสถิติของราคาทองคำในอดีต (เช่น แนวโน้ม, ฤดูกาล, ความสัมพันธ์กับค่าในอดีต) มีอิทธิพลอย่างมีนัยสำคัญ ต่อราคาในอนาคต ซึ่งแบบจำลอง Machine Learning สามารถเรียนรู้ได้

1.6 เครื่องมือที่ใช้ในการวิจัย

1. ภาษา Python เป็นภาษาคอมพิวเตอร์ที่นิยมใช้ในการพัฒนาโปรแกรมด้านวิทยาการข้อมูล (Data Science) และปัญญาประดิษฐ์ เนื่องจากมีไลบรารีจำนวนมากที่ช่วยในการวิเคราะห์ข้อมูลและสร้างแบบจำลอง Machine Learning

2. TensorFlow เป็นไลบรารีสำหรับพัฒนา Machine Learning และ Deep Learning ที่พัฒนาโดย Google ใช้สำหรับสร้างและฝึกแบบจำลองโครงข่ายประสาทเทียม เช่น LSTM เพื่อใช้ในการทำนายข้อมูลอนุกรมเวลา

3. Keras เป็นไลบรารีสำหรับสร้างโมเดล Deep Learning ที่ทำงานร่วมกับ TensorFlow ช่วยให้สามารถสร้างโครงข่ายประสาทเทียมได้ง่ายและสะดวกยิ่งขึ้น

4. Pandas เป็นไลบรารีในภาษา Python ที่ใช้สำหรับจัดการและวิเคราะห์ข้อมูล เช่น การอ่านไฟล์ข้อมูล การจัดรูปแบบข้อมูล และการเตรียมข้อมูลก่อนนำไปใช้ในการฝึกโมเดล

5. NumPy เป็นไลบรารีสำหรับการคำนวณเชิงตัวเลขในภาษา Python ใช้สำหรับจัดการข้อมูลในรูปแบบอาร์เรย์และช่วยเพิ่มประสิทธิภาพในการคำนวณ

6. Matplotlib เป็นไลบรารีสำหรับสร้างกราฟและการแสดงผลข้อมูลในรูปแบบภาพ เช่น กราฟเส้น เพื่อใช้เปรียบเทียบราคาทองคำจริงกับราคาที่โมเดลทำนาย

7. แหล่งข้อมูลราคาทองคำ ข้อมูลราคาทองคำที่ใช้ในการวิจัยนี้ถูกดึงมาจาก Yahoo Finance โดยใช้ไลบรารี yfinance ในภาษา Python ซึ่งข้อมูลประกอบด้วยราคาปิดของทองคำในแต่ละวัน เพื่อนำมาใช้ในการฝึกแบบจำลอง

1.7 นิยามศัพท์เฉพาะ

-Machine Learning (การเรียนรู้ของเครื่อง) คือสาขาหนึ่งของปัญญาประดิฐษ์ ที่เน้นการพัฒนาโปรแกรมให้สามารถเรียนรู้จากข้อมูลได้โดยไม่ต้องมีการเขียนโปรแกรมอย่าง ชัดเจนในการตัดสินใจ
-Time Series (อนุกรมเวลา) คือชุดข้อมูลที่มีการจัดเรียงตามลำดับเวลา เช่น ราคาหุ้นรายวัน หรืออุณหภูมิรายชั่วโมง
-RMSE (Root Mean Squared Error) (รากของค่าเฉลี่ยความคลาดเคลื่อนกำลังสอง) เป็นตัวชี้วัดความแม่นยำที่ใช้ ประเมินความแตกต่างระหว่างค่าที่ทำนายได้และค่าจริง โดยแสดงผลในหน่วยเดียวกับตัวแปรทำนาย
-API (Application Programming Interface) คือชุดของข้อกำหนดที่ช่วยให้ โปรแกรมซอฟต์แวร์สองโปรแกรมสามารถสื่อสารกันได้ ใช้ในการดึงข้อมูลราคา ทองคำแบบอัตโนมัติ
-Features (ฟีเจอร์/คุณลักณะ) คือตัวแปรอิสระ (Independent Variables) หรือข้อมูล นำเข้าที่ใช้ในการฝึกฝนแบบจำลอง เพื่อทำนายค่าเป้าหมาย (ราคาทองคำ)

2 ความรู้พื้นฐาน

2.1 ความรู้พื้นฐานเกี่ยวกับภาษา Python

Python เป็นภาษาการเขียนโปรแกรมระดับสูง (High-level Programming Language) ที่ถูกพัฒนาโดย Guido van Rossum และเปิดตัวครั้งแรกในปี ค.ศ. 1991 ภาษา Python ถูกออกแบบให้มีโครงสร้างที่อ่านง่ายและเข้าใจได้สะดวก มีไวยากรณ์ที่ไม่ซับซ้อน จึงเหมาะสำหรับผู้เริ่มต้นเรียนรู้การเขียนโปรแกรม รวมถึงถูกนำไปใช้ในงานด้านต่าง ๆ เช่น การพัฒนาเว็บ การวิเคราะห์ข้อมูล ปัญญาประดิษฐ์ และระบบอัตโนมัติ ภาษา Python เป็นภาษาแบบ Interpreted Language ซึ่งหมายถึงการทำงานของโปรแกรมจะถูกแปลและประมวลผลทีละคำสั่งขณะรันโปรแกรม ทำให้สามารถทดสอบและแก้ไขโค้ดได้อย่างรวดเร็ว อีกทั้งยังสามารถใช้งานได้บนหลายระบบปฏิบัติการ เช่น Windows, macOS และ Linux

2.2 ความรู้พื้นฐานเกี่ยวกับโครงสร้างพื้นฐานของภาษา Python

2.2.1 ตัวแปรและชนิดข้อมูล

-ตัวแปร (Variable) คือพื้นที่สำหรับเก็บข้อมูลในหน่วยความจำ โดยภาษา Python ไม่จำเป็นต้องประกาศชนิดข้อมูลล่วงหน้า เนื่องจากเป็นภาษาแบบ Dynamic Typing
-ชนิดข้อมูลพื้นฐานที่ใช้บ่อย ได้แก่

- Integer (int) สำหรับเก็บจำนวนเต็ม
- Float สำหรับเก็บจำนวนทศนิยม
- String (str) สำหรับเก็บข้อความ
- Boolean (bool) สำหรับเก็บค่าความจริง ได้แก่ True และ False

นอกจากนี้ยังมีโครงสร้างข้อมูลอื่น ๆ เช่น List, Tuple, Dictionary และ Set ซึ่งใช้สำหรับจัดเก็บข้อมูลหลายค่า

2.2.2 โครงสร้างควบคุมการทำงานของโปรแกรม

ภาษา Python มีโครงสร้างสำหรับควบคุมลำดับการทำงานของโปรแกรม เช่น การกำหนดเงื่อนไขและการวนซ้ำ

1. โครงสร้างเงื่อนไข (Conditional Statements) ใช้คำสั่ง if, elif, และ else เพื่อตรวจสอบเงื่อนไขก่อนทำงาน
2. โครงสร้างการวนซ้ำ (Loop) ใช้คำสั่ง for และ while เพื่อทำงานซ้ำหลายครั้งตามเงื่อนไขที่กำหนด

โครงสร้างเหล่านี้ช่วยให้โปรแกรมสามารถตัดสินใจและทำงานได้อย่างมีประสิทธิภาพ

2.2.3 ฟังก์ชัน (Functions)

ฟังก์ชันคือชุดคำสั่งที่ถูกจัดกลุ่มไว้เพื่อทำงานเฉพาะอย่าง และสามารถเรียกใช้งานซ้ำได้หลายครั้ง การสร้างฟังก์ชันในภาษา Python ใช้คำสั่ง def ซึ่งช่วยให้โปรแกรมมีความเป็นระเบียบและลดการเขียนโค้ดซ้ำซ้อน

2.3 ความรู้พื้นฐานเกี่ยวกับไลบรารีพื้นฐานของ Python

ไลบรารี (Library) คือชุดคำสั่งหรือโมดูลที่ถูกรวบรวมไว้เพื่อช่วยอำนวยความสะดวกในการพัฒนาโปรแกรม โดยภาษา Python มีทั้งไลบรารีมาตรฐานที่มาพร้อมกับภาษา และไลบรารีเพิ่มเติมที่สามารถติดตั้งภายหลังได้

2.3.1 ไลบรารีมาตรฐาน

ไลบรารีมาตรฐานเป็นไลบรารีที่ติดตั้งมาพร้อมกับ Python โดยไม่ต้องติดตั้งเพิ่มเติม ตัวอย่างเช่น

- math ใช้สำหรับการคำนวณทางคณิตศาสตร์ เช่น รากที่สอง ค่าไตรโกณมิติ และค่าคงที่ทางคณิตศาสตร์
- random ใช้สำหรับการสุ่มตัวเลขหรือข้อมูลต่าง ๆ
- datetime ใช้สำหรับจัดการข้อมูลเกี่ยวกับวันที่และเวลา
- os ใช้สำหรับจัดการไฟล์และติดต่อกับระบบปฏิบัติการ

2.3.2 ไลบรารีเพิ่มเติมที่นิยมใช้

- NumPy ใช้สำหรับการคำนวณเชิงตัวเลขและการจัดการอาร์เรย์หลายมิติ
- Pandas ใช้สำหรับการวิเคราะห์และจัดการข้อมูลในรูปแบบตาราง
- Matplotlib ใช้สำหรับการสร้างกราฟและการแสดงผลข้อมูล
- TensorFlow ใช้สำหรับพัฒนาโมเดลปัญญาประดิษฐ์และการเรียนรู้ของเครื่อง

ไลบรารีเหล่านี้ทำให้ภาษา Python สามารถนำไปประยุกต์ใช้ในงานหลากหลายสาขา เช่น วิทยาการข้อมูล วิศวกรรมซอฟต์แวร์ และการพัฒนาระบบอัตโนมัติ

2.4 ความรู้พื้นฐานเกี่ยวกับการจัดทำและเตรียมข้อมูล (Data Preprocessing)

การจัดทำและเตรียมข้อมูล หรือ Data Preprocessing เป็นขั้นตอนสำคัญในกระบวนการวิเคราะห์ข้อมูลในสาขา Data Science และ Machine Learning โดยมีจุดประสงค์เพื่อปรับปรุงคุณภาพของข้อมูลก่อนนำไปใช้ในการวิเคราะห์หรือสร้างแบบจำลอง เนื่องจากข้อมูลดิบที่ได้จากแหล่งต่าง ๆ มักมีปัญหา เช่น ข้อมูลสูญหาย ข้อมูลซ้ำ หรือข้อมูลที่มีรูปแบบไม่เหมาะสม ดังนั้นการเตรียมข้อมูลจึงเป็นขั้นตอนที่ช่วยให้ข้อมูลมีความถูกต้อง เป็นระเบียบ และสามารถนำไปใช้ประโยชน์ได้อย่างมีประสิทธิภาพ

ขั้นตอนสำคัญของการเตรียมข้อมูล

1. การทำความสะอาดข้อมูล (Data Cleaning)
   การทำความสะอาดข้อมูลเป็นขั้นตอนในการตรวจสอบและแก้ไขข้อผิดพลาดของข้อมูล เช่น การจัดการค่าที่หายไป (Missing Values) การลบข้อมูลที่ซ้ำกัน (Duplicate Data) และการแก้ไขข้อมูลที่มีรูปแบบไม่ถูกต้อง การทำความสะอาดข้อมูลช่วยให้ข้อมูลมีความถูกต้องและลดความคลาดเคลื่อนในการวิเคราะห์

2. การรวมข้อมูล (Data Integration)
   การรวมข้อมูลเป็นกระบวนการนำข้อมูลจากหลายแหล่งมารวมกันให้อยู่ในรูปแบบเดียวกัน เช่น การรวมข้อมูลจากฐานข้อมูลหลายระบบ หรือการรวมข้อมูลจากไฟล์หลายประเภท เพื่อให้สามารถนำข้อมูลทั้งหมดมาวิเคราะห์ร่วมกันได้อย่างมีประสิทธิภาพ

3. การแปลงข้อมูล (Data Transformation)
   การแปลงข้อมูลเป็นการปรับรูปแบบของข้อมูลให้เหมาะสมกับการใช้งาน เช่น การปรับสเกลของข้อมูล (Normalization หรือ Standardization) การแปลงข้อมูลประเภทข้อความให้เป็นตัวเลข หรือการจัดกลุ่มข้อมูลเพื่อให้ง่ายต่อการวิเคราะห์

4. การลดขนาดข้อมูล (Data Reduction)
   การลดขนาดข้อมูลเป็นการลดจำนวนข้อมูลหรือจำนวนตัวแปรโดยยังคงข้อมูลสำคัญไว้ เช่น การเลือกคุณลักษณะที่สำคัญ (Feature Selection) หรือการลดมิติของข้อมูล (Dimensionality Reduction) ซึ่งช่วยลดเวลาในการประมวลผลและเพิ่มประสิทธิภาพในการวิเคราะห์ข้อมูล

เครื่องมือที่ใช้ในการเตรียมข้อมูล

ในการจัดทำและเตรียมข้อมูล มักใช้เครื่องมือและภาษาการเขียนโปรแกรมที่ช่วยในการจัดการข้อมูล เช่น

- Python สำหรับการพัฒนาโปรแกรมและวิเคราะห์ข้อมูล
- Pandas สำหรับการจัดการข้อมูลในรูปแบบตาราง
- NumPy สำหรับการคำนวณเชิงตัวเลข
- Scikit-learn สำหรับการเตรียมข้อมูลและการสร้างโมเดล Machine Learning

2.5 ความรู้พื้นฐานเกี่ยวกับโครงข่ายประสาทเทียมเชิงลึก (Deep Learning)

การใช้เทคนิค Deep Learning โดยเฉพาะโมเดล LSTM สามารถช่วยในการวิเคราะห์และทำนายราคาทองคำได้อย่างมีประสิทธิภาพ เนื่องจากสามารถเรียนรู้รูปแบบของข้อมูลอนุกรมเวลาในอดีตได้ดี เมื่อผสานกับการประเมินผลด้วยตัวชี้วัด เช่น MAE, RMSE และ MAPE จะช่วยให้สามารถประเมินความแม่นยำของโมเดลและเลือกโมเดลที่เหมาะสมสำหรับการพยากรณ์ราคาทองคำได้

องค์ประกอบสำคัญของโมเดลที่ใช้ในงานนี้ ได้แก่

1. LSTM (Long Short-Term Memory)
   Long Short-Term Memory เป็นโครงสร้างพิเศษของ RNN ที่สามารถจดจำข้อมูลในระยะยาวได้ดี ทำให้เหมาะสำหรับการพยากรณ์ข้อมูลอนุกรมเวลา เช่น ราคาหุ้น ราคาน้ำมัน และราคาทองคำ โมเดล LSTM สามารถเรียนรู้แนวโน้มและรูปแบบของราคาทองคำในอดีต เพื่อนำมาคาดการณ์ราคาที่อาจเกิดขึ้นในอนาคต

2. Dropout
   เทคนิค Dropout ใช้เพื่อลดปัญหา Overfitting โดยการสุ่มปิดการทำงานของบางโหนดในโครงข่ายประสาทเทียมระหว่างการฝึกโมเดล วิธีนี้ช่วยให้โมเดลสามารถเรียนรู้ข้อมูลได้อย่างทั่วไปและไม่จดจำข้อมูลฝึกมากเกินไป

3. Dense Layer
   Dense Layer เป็นเลเยอร์ที่เชื่อมต่อโหนดทั้งหมดเข้าด้วยกัน โดยทำหน้าที่รวบรวมข้อมูลจากเลเยอร์ก่อนหน้าและคำนวณผลลัพธ์สุดท้าย ซึ่งในกรณีของการทำนายราคาทองคำ เลเยอร์นี้จะใช้สำหรับคำนวณค่าราคาที่โมเดลพยากรณ์ออกมา

2.6 การวัดผลและประเมินประสิทธิภาพของโมเดล (Evaluation Metrics)

หลังจากสร้างและฝึกโมเดลแล้ว จำเป็นต้องมีการประเมินประสิทธิภาพของโมเดลเพื่อวัดความแม่นยำของผลการพยากรณ์ โดยใช้ตัวชี้วัดทางสถิติที่เรียกว่า Evaluation Metrics ตัวชี้วัดที่นิยมใช้ในการประเมินผลของโมเดลพยากรณ์ ได้แก่

- Mean Absolute Error (MAE) ใช้วัดค่าเฉลี่ยของความคลาดเคลื่อนระหว่างค่าที่พยากรณ์กับค่าจริง ยิ่งมีค่าน้อยยิ่งแสดงว่าโมเดลมีความแม่นยำสูง
- Root Mean Squared Error (RMSE) ใช้วัดความคลาดเคลื่อนของการพยากรณ์ โดยให้ความสำคัญกับค่าความผิดพลาดที่มีขนาดมาก
- Mean Absolute Percentage Error (MAPE) เป็นค่าร้อยละของความคลาดเคลื่อนเฉลี่ย ทำให้สามารถประเมินความแม่นยำของโมเดลได้ง่ายในรูปแบบเปอร์เซ็นต์

3 วิธีการดำเนินงาน

การทำนายราคาทองคำในงานวิจัยนี้ดำเนินการโดย Tensorflow และโครงข่ายประสาทเทียม LSTM (Long Short-Term Memory) ซึ่งเหมาะสำหรับการวิเคราะห์ข้อมูลลำดับเวลา โดยกระบวนการดำเนินงานประกอบด้วยขั้นตอนสำคัญดังต่อไปนี้

1. **การดึงข้อมูล (Data Collection)** ทำการดึงข้อมูลราคาทองคำจากแหล่งข้อมูลออนไลน์ โดยใช้ไลบรารี `yfinance` ซึ่งสามารถดึงข้อมูลราคาปิด (Close) ของทองคำในแต่ละวันมาใช้ในการวิเคราะห์

2. **การทำความสะอาดข้อมูล (Data Cleaning)** จัดระเบียบข้อมูลที่ได้จากการดึงข้อมูล เช่น การแก้ไขหัวตารางที่ซ้อนกัน การเลือกเฉพาะคอลัมน์ราคาปิด และการจัดรูปแบบข้อมูลวันที่ เพื่อให้ง่ายต่อการนำไปใช้งาน

3. **การปรับขนาดข้อมูล (Data Scaling)** ใช้เทคนิค `MinMaxScaler` เพื่อปรับค่าราคาทองคำให้อยู่ในช่วงระหว่าง 0 ถึง 1 ซึ่งช่วยให้แบบจำลองปัญญาประดิษฐ์สามารถเรียนรู้ข้อมูลได้อย่างมีประสิทธิภาพมากขึ้น

4. **การเตรียมข้อมูลสำหรับโมเดล (Data Preprocessing)** จัดรูปแบบข้อมูลให้เป็นลักษณะลำดับเวลา (Sequence) โดยใช้ข้อมูลย้อนหลังจำนวนหลายวัน เช่น 7 วัน เพื่อใช้เป็นข้อมูลนำเข้าในการทำนายราคาทองคำในอนาคต

5. **การสร้างและฝึกโมเดล (Model Training)** สร้างแบบจำลอง LSTM โดยใช้โครงข่ายประสาทเทียมหลายชั้น พร้อมทั้งกำหนดพารามิเตอร์การเรียนรู้ เช่น จำนวน Epoch และ Optimizer เพื่อฝึกให้โมเดลสามารถเรียนรู้รูปแบบการเปลี่ยนแปลงของราคาทองคำ

6. **การพยากรณ์ราคา (Prediction)** นำโมเดลที่ผ่านการฝึกแล้วมาใช้ในการทำนายราคาทองคำในอนาคต เช่น การพยากรณ์ราคาล่วงหน้า 1 ถึง 3 วัน

7. **การแสดงผลและวิเคราะห์ผลลัพธ์ (Visualization and Evaluation)** แสดงผลการพยากรณ์ด้วยกราฟเพื่อเปรียบเทียบราคาจริงกับราคาที่โมเดลทำนาย พร้อมทั้งประเมินประสิทธิภาพของโมเดลโดยใช้ตัวชี้วัด เช่น MAE, RMSE และ MAPE

3.1 การดึงข้อมูล

```
!pip install yahoo_fin yfinance
```

```
Requirement already satisfied: yahoo_fin in /usr/local/lib/python3.12/dist-packages (0.8.9.1)
Requirement already satisfied: yfinance in /usr/local/lib/python3.12/dist-packages (0.2.66)
Requirement already satisfied: requests-html in /usr/local/lib/python3.12/dist-packages (from yahoo_fin) (0.10.0)
Requirement already satisfied: feedparser in /usr/local/lib/python3.12/dist-packages (from yahoo_fin) (6.0.12)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from yahoo_fin) (2.32.4)
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (from yahoo_fin) (2.2.2)
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.12/dist-packages (from yfinance) (2.0.2)
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.12/dist-packages (from yfinance) (0.0.12)
Requirement already satisfied: platformdirs>=2.0.0 in /usr/local/lib/python3.12/dist-packages (from yfinance) (4.9.4)
Requirement already satisfied: pytz>=2022.5 in /usr/local/lib/python3.12/dist-packages (from yfinance) (2025.2)
Requirement already satisfied: frozendict>=2.3.4 in /usr/local/lib/python3.12/dist-packages (from yfinance) (2.4.7)
Requirement already satisfied: peewee>=3.16.2 in /usr/local/lib/python3.12/dist-packages (from yfinance) (4.0.1)
Requirement already satisfied: beautifulsoup4>=4.11.1 in /usr/local/lib/python3.12/dist-packages (from yfinance) (4.13.5)
Requirement already satisfied: curl_cffi>=0.7 in /usr/local/lib/python3.12/dist-packages (from yfinance) (0.14.0)
Requirement already satisfied: protobuf>=3.19.0 in /usr/local/lib/python3.12/dist-packages (from yfinance) (5.29.6)
Requirement already satisfied: websockets>=13.0 in /usr/local/lib/python3.12/dist-packages (from yfinance) (15.0.1)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.12/dist-packages (from beautifulsoup4>=4.11.1->yfinance) (2.8.3)
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.12/dist-packages (from beautifulsoup4>=4.11.1->yfinance) (4.15.0)
Requirement already satisfied: cffi>=1.12.0 in /usr/local/lib/python3.12/dist-packages (from curl_cffi>=0.7->yfinance) (2.0.0)
Requirement already satisfied: certifi>=2024.2.2 in /usr/local/lib/python3.12/dist-packages (from curl_cffi>=0.7->yfinance) (2026.2.25)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas->yahoo_fin) (2.9.0.post0)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas->yahoo_fin) (2025.3)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests->yahoo_fin) (3.4.5)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests->yahoo_fin) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests->yahoo_fin) (2.5.0)
Requirement already satisfied: sgmllib3k in /usr/local/lib/python3.12/dist-packages (from feedparser->yahoo_fin) (1.0.0)
Requirement already satisfied: pyquery in /usr/local/lib/python3.12/dist-packages (from requests-html->yahoo_fin) (2.0.1)
Requirement already satisfied: fake-useragent in /usr/local/lib/python3.12/dist-packages (from requests-html->yahoo_fin) (2.2.0)
Requirement already satisfied: parse in /usr/local/lib/python3.12/dist-packages (from requests-html->yahoo_fin) (1.21.1)
Requirement already satisfied: bs4 in /usr/local/lib/python3.12/dist-packages (from requests-html->yahoo_fin) (0.0.2)
Requirement already satisfied: w3lib in /usr/local/lib/python3.12/dist-packages (from requests-html->yahoo_fin) (2.4.0)
Requirement already satisfied: pyppeteer>=0.0.14 in /usr/local/lib/python3.12/dist-packages (from requests-html->yahoo_fin) (0.0.25)
Requirement already satisfied: pycparser in /usr/local/lib/python3.12/dist-packages (from cffi>=1.12.0->curl_cffi>=0.7->yfinance) (3.0)
Requirement already satisfied: pyee in /usr/local/lib/python3.12/dist-packages (from pyppeteer>=0.0.14->requests-html->yahoo_fin) (13.0.1)
Requirement already satisfied: appdirs in /usr/local/lib/python3.12/dist-packages (from pyppeteer>=0.0.14->requests-html->yahoo_fin) (1.4.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from pyppeteer>=0.0.14->requests-html->yahoo_fin) (4.67.3)
```

ติดตั้ง `yahoo_fin` และ `yfinance` เพื่อทำการดึงราคาทองจากออนไลน์

```
#import
import numpy as np
import time as tm
import datetime as dt#ใช้จัดการเรื่องวันที่และเวลา
import tensorflow as tf #เป็นแพลตฟอร์มสำหรับสร้างและฝึกสอนโมเดล

# Data preparation
from yahoo_fin import stock_info as yf_yahoo_fin # เป็นไลบรารีที่ดึงข้อมูลงบการเงินและราคาหุ้น/สินค้าโภคภัณฑ์จาก Yahoo Finance
import yfinance as yf # ตัวนี้เป็นที่นิยมมากในการดึงข้อมูลราคาย้อนหลัง
from sklearn.preprocessing import MinMaxScaler#โมเดล
from collections import deque #เป็นโครงสร้างข้อมูลแบบ "คิว" (Queue) ที่มีความพิเศษคือถ้าข้อมูลเต็มแล้วใส่ใหม่เข้าไป ข้อมูลเก่าสุดจะหลุดออกไปเอง

# AI
from keras.models import Sequential#แพลตฟอร์มหลักที่ใช้สร้างและเทรนโมเดล Machine Learning ของ Google
from keras.layers import Dense, LSTM, Dropout#ใช้สร้างโมเดลแบบ "เรียงลำดับ"

# Graphics library
import matplotlib.pyplot as plt
```

นำเข้าไลบรารี (Import Libraries) สำหรับเขียนโปรแกรม AI พยากรณ์ราคาหุ้น โดยแบ่งกลุ่มตามหน้าที่ได้ดังนี้

1. การจัดการข้อมูลและเวลา
2. การดึงข้อมูลหุ้น
3. การเตรียมข้อมูลและ AI
4. การแสดงผลกราฟ

```
# SETTINGS

# Window size or the sequence length, 7 (1 week)
N_STEPS = 7

# Lookup steps, 1 is the next day, 3 = after tomorrow
LOOKUP_STEPS = [1, 2, 3]

# Stock ticker, GOOGLฆฆ
STOCK = 'GC=F'

# Current date
date_now = tm.strftime('%Y-%m-%d')
date_3_years_back = (dt.date.today() - dt.timedelta(days=1100)).strftime('%Y-%m-%d')
```

เตรียมให้ AI เรียนรู้จากราคาทองคำย้อนหลัง 3 ปี โดยใช้ข้อมูลรายสัปดาห์ (7 วัน) มาทายราคาล่วงหน้า 1-3 วัน

3.2 การทำความสะอาดข้อมูล

1. การโหลดข้อมูล (Load Data) จากอินเทอร์เน็ต

ในขั้นตอนนี้เป็นการดึงข้อมูลราคาทองคำจากแหล่งข้อมูลออนไลน์ โดยใช้ภาษา Python ร่วมกับไลบรารี yfinance ซึ่งสามารถเชื่อมต่อกับฐานข้อมูลของ Yahoo Finance เพื่อดึงข้อมูลราคาทองคำย้อนหลังตามช่วงเวลาที่กำหนด

```
# LOAD DATA
# Using yfinance for more reliable data fetching

init_df = None
attempt = 0#ตั้งค่าว่าตอนนี้พยายามไปกี่ครั้งแล้ว
max_retries = 3#อนุญาตให้ลองใหม่ได้dujiv[]

while attempt < max_retries:
    try:
        # Use yfinance.download instead of yahoo_fin.stock_info.get_data
        init_df = yf.download(
            STOCK,
            start=date_3_years_back,
            end=date_now,
            interval='1d')
        print(f"Data successfully loaded on attempt {attempt + 1}.")
        break # Exit loop if successful
    except Exception as e: #ถ้าใน try เกิดปัญหา.ให้มาทำตรงนี้แทน
        print(f"Error loading data on attempt {attempt + 1}: {e}")
        attempt += 1
        if attempt < max_retries:
            print(f"Retrying in 5 seconds...")
            tm.sleep(5) # Wait for 5 seconds before retrying
        else:
            print("Max retries reached. Could not load data.")
            raise # Re-raise the last exception if all retries fail

if init_df is not None and not init_df.empty:#เป็นการตรวจสอบว่าตัวแปรinit_dfมีข้อมูลอยู่จริง
    print(f"Loaded {len(init_df)} rows of data.")
else:
    print("No data was loaded.")
```

2 การดึงข้อมูลแบบป้องกันข้อผิดพลาด (Retry Mechanism)

เพื่อป้องกันปัญหาที่อาจเกิดขึ้นระหว่างการดึงข้อมูลจากอินเทอร์เน็ต เช่น การเชื่อมต่อเครือข่ายขัดข้องหรือเซิร์ฟเวอร์ไม่ตอบสนอง โปรแกรมจึงถูกออกแบบให้มี กลไกการลองดึงข้อมูลใหม่ (Retry Mechanism

โดยใช้คำสั่ง while ร่วมกับ try...except ในภาษา Python เพื่อจัดการข้อผิดพลาดที่อาจเกิดขึ้น

โปรแกรมกำหนดจำนวนครั้งสูงสุดในการพยายามดึงข้อมูลไว้ที่ 3 ครั้ง (`max_retries = 3`) หากไม่สามารถดึงข้อมูลได้สำเร็จในครั้งแรก โปรแกรมจะไม่หยุดทำงานทันที แต่จะทำการลองดึงข้อมูลใหม่ พร้อมทั้งหน่วงเวลาเป็นระยะเวลา 5 วินาที (`tm.sleep(5)`) ก่อนเริ่มการทำงานในรอบถัดไป

3. แหล่งข้อมูล

การดึงข้อมูลใช้ฟังก์ชัน `yf.download` จากไลบรารี `yfinance` เพื่อดาวน์โหลดข้อมูลราคาทองคำจาก Yahoo Finance โดยกำหนดช่วงเวลาข้อมูลย้อนหลัง 3 ปี และใช้ข้อมูลแบบรายวัน (`interval='1d'`)

4. การตรวจสอบความถูกต้องของข้อมูล

หลังจากสิ้นสุดกระบวนการดึงข้อมูล โปรแกรมจะทำการตรวจสอบว่าข้อมูลถูกโหลดมาได้ อย่างถูกต้องหรือไม่ โดยใช้เงื่อนไข

```
if init_df is not None and not init_df.empty
```

หากข้อมูลถูกโหลดสำเร็จ โปรแกรมจะแสดงจำนวนแถวของข้อมูลที่ได้รับ โดยใช้คำสั่ง `len(init_df)` เพื่อยืนยันว่าข้อมูลพร้อมสำหรับการนำไปใช้ ในขั้นตอนการวิเคราะห์และการสร้างแบบจำลองต่อไป

```
init_df.info()

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 758 entries, 2023-03-09 to 2026-03-13
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   (Close, GC=F)   757 non-null    float64
 1   (High, GC=F)    757 non-null    float64
 2   (Low, GC=F)     757 non-null    float64
 3   (Open, GC=F)    757 non-null    float64
 4   (Volume, GC=F)  758 non-null    int64
dtypes: float64(4), int64(1)
memory usage: 35.5 KB
```

ผลการตรวจสอบข้อมูล

เตรียมข้อมูล (Data Preprocessing)

การเตรียมข้อมูลเป็นขั้นตอนสำคัญก่อนนำข้อมูลเข้าสู่กระบวนการเรียนรู้ของโมเดลปัญญาประดิษฐ์ประเภท LSTM โดยมีวัตถุประสงค์เพื่อจัดรูปแบบข้อมูลให้เหมาะสมสำหรับการฝึกโมเดล ซึ่งประกอบด้วยขั้นตอนหลักดังนี้

```
# --- Data Preprocessing for LSTM ---

# Ensure 'Close' column exists and is numeric
if 'Close' not in init_df.columns:
    raise KeyError("The 'Close' column is missing from the loaded data.")

# Create a new DataFrame 'df' for further processing
df = init_df[['Close']].copy()

# Add 'future' columns that contain the 'Close' price N_STEPS days in the future for each LOOKUP_STEP
for step in LOOKUP_STEPS:
    df[f'future_{step}'] = df['Close'].shift(-step)

# Drop rows with NaN values (which are introduced by the shift operation at the end of the dataframe)
df.dropna(inplace=True)

# Select features (only 'Close' for now)
feature_cols = ['Close']

# Scale data
scaler = MinMaxScaler()#บีบข้อมูล
# Fit and transform the feature columns (scaler is fitted on a single feature)
df[feature_cols] = scaler.fit_transform(df[feature_cols])

# Also scale the target (future) columns using the same scaler
target_cols = [f'future_{step}' for step in LOOKUP_STEPS]
# Corrected line: Iterate and transform each target column individually
for col in target_cols:
    df[col] = scaler.transform(df[[col]]) # Pass a single-column DataFrame for transformation แปลราคาทองให้เป็น0:1

# Now, create sequences for the LSTM model
X = []
y = []
for index in range(len(df) - N_STEPS + 1):
    X.append(df[feature_cols].iloc[index : index + N_STEPS].values)
    y.append(df[target_cols].iloc[index + N_STEPS - 1].values) # Target is the future price at the end of the sequence

X = np.array(X)
y = np.array(y)

print(f"Shape of X (sequences): {X.shape}")
print(f"Shape of y (targets): {y.shape}")

# Store the last N_STEPS sequence for future prediction
# This sequence will be used for making actual predictions after the model is trained
last_sequence = df[feature_cols].iloc[-N_STEPS:].values
```

```
Shape of X (sequences): (748, 7, 1)
Shape of y (targets): (748, 3)
```

1) การสร้างเฉลย (Labeling)

ในขั้นตอนนี้ใช้คำสั่ง `shift(-step)` เพื่อเลื่อนข้อมูลราคาปิด (Close) ไปยังตำแหน่งก่อนหน้าเพื่อสร้างค่าราคาในอนาคต เช่น 1 วัน 2 วัน และ 3 วันข้างหน้า โดยค่าดังกล่าวจะถูกนำมาใช้เป็นคำตอบหรือเฉลย (Label) สำหรับให้โมเดลเรียนรู้ในการทำนายค่าราคาในอนาคต

2) การปรับขนาดข้อมูล (Scaling)
เนื่องจากข้อมูลราคาทองคำมีค่าค่อนข้างสูง เช่น หลักหมื่นหรือหลักแสน จึงมีการใช้เทคนิค `MinMaxScaler` เพื่อปรับขนาดข้อมูลให้อยู่ในช่วงระหว่าง 0 ถึง 1 ซึ่งช่วยให้โมเดล LSTM สามารถเรียนรู้รูปแบบของข้อมูลได้อย่างมีประสิทธิภาพและรวดเร็วมากยิ่งขึ้น

3) การจัดรูปแบบข้อมูลแบบลำดับ (Windowing)
เพื่อให้เหมาะสมกับการทำงานของโมเดล LSTM ซึ่งออกแบบมาสำหรับข้อมูลลำดับเวลา จึงมีการจัดข้อมูล (Sequence) เพื่อใช้เป็นโจทย์
สร้างตัวแปร `X`: สำหรับเก็บข้อมูลราคาย้อนหลัง 7 วัน เพื่อใช้เป็นข้อมูลนำเข้า (Input) ของโมเดล
สร้างตัวแปร `y`: สำหรับเก็บค่าราคาในอนาคตซึ่งใช้เป็นผลลัพธ์ (Target) ที่โมเดลต้องทำนาย
จากผลลัพธ์ของการเตรียมข้อมูลพบว่ามีชุดข้อมูลทั้งหมด 748 ชุด โดยแต่ละชุดประกอบด้วยข้อมูลย้อนหลัง 7 วัน เพื่อใช้ในการทำนายราคาล่วงหน้า 3 ค่าในอนาคต

4) การเก็บข้อมูลชุดสุดท้ายสำหรับการพยากรณ์
ในขั้นตอนสุดท้ายมีการเก็บข้อมูลราคาล่าสุดย้อนหลัง 7 วันไว้ในตัวแปร `last_sequence` เพื่อนำไปใช้สำหรับการทำนายค่าจริงในอนาคต หลังจากที่โมเดล LSTM ผ่านกระบวนการฝึก (Training) เรียบร้อยแล้ว

การจัดระเบียบตารางข้อมูล (Data Cleaning)
การจัดระเบียบข้อมูลเป็นขั้นตอนสำคัญเพื่อเตรียมข้อมูลให้มีรูปแบบที่เหมาะสมก่อนนำไปวิเคราะห์หรือใช้ในการฝึกโมเดลปัญญาประดิษฐ์ โดยในขั้นตอนนี้มีการปรับโครงสร้างของตารางข้อมูลให้เรียบง่ายและสะดวกต่อการใช้งาน ซึ่งประกอบด้วยขั้นตอนดังต่อไปนี้

```
import pandas as pd

# If columns are MultiIndex, flatten them to a single level
# Example: ('Price', 'Close') becomes 'Close'
if isinstance(init_df.columns, pd.MultiIndex):
    # Correctly extract the column names from the first level of the MultiIndex
    init_df.columns = init_df.columns.get_level_values(0) #เช็กว่าถ้าหัวตารางมันซ้อนกันอยู่ให้ดึงมาแค่ชั้นเดียว

# Keep only the 'Close' column as this seems to be the primary focus for subsequent steps
init_df = init_df[['Close']]

# Create the column 'date' based on index column
init_df['date'] = init_df.index
```

1) การแก้ปัญหาหัวตารางซ้อน (Flatten MultiIndex)
ในบางกรณีข้อมูลที่ดึงมาจากไลบรารี `yfinance` จะมีหัวตาราง (Column Header) แบบสองชั้นหรือที่เรียกว่า MultiIndex เช่น ชั้นบนอาจเป็นคำว่า `Price` และชั้นล่างเป็น `Close` เป็นต้น เพื่อให้สามารถเรียกใช้งานคอลัมน์ได้ง่ายขึ้น จึงมีการใช้คำสั่ง `get_level_values(0)` เพื่อยุบหัวตารางให้เหลือเพียงชั้นเดียว ทำให้โครงสร้างของข้อมูลมีความเรียบง่ายและลดความสับสนในการเรียกใช้งาน

2) การเลือกเฉพาะข้อมูลที่จำเป็น

หลังจากปรับโครงสร้างตารางข้อมูลแล้ว จะทำการเลือกเฉพาะคอลัมน์ที่จำเป็นสำหรับการวิเคราะห์ โดยในงานนี้เลือกใช้เฉพาะคอลัมน์ `Close` ซึ่งเป็นราคาปิดของสินทรัพย์ ส่วนข้อมูลอื่น ๆ เช่น `Open`, `High`, `Low` และ `Volume` จะถูกตัดออก เพื่อลดความซับซ้อนของข้อมูลและทำให้โมเดลสามารถมุ่งเน้นการเรียนรู้จากราคาปิดเพียงอย่างเดียว

3) การดึงข้อมูลวันที่ออกมาใช้งาน

โดยปกติแล้วข้อมูลวันที่จะถูกเก็บไว้ในรูปแบบดัชนี (Index) ของตารางข้อมูล เพื่อให้สามารถนำข้อมูลวันที่ไปใช้งานได้สะดวก เช่น การสร้างกราฟหรือการวิเคราะห์ข้อมูลตามช่วงเวลา จึงมีการสร้างคอลัมน์ใหม่ชื่อ `date` โดยคัดลอกค่าจากดัชนีของตารางมาเก็บไว้ในคอลัมน์ดังกล่าว

```
init_df.head()
```

Price Date	Close	date
2023-03-09	1829.300049	2023-03-09
2023-03-10	1862.000000	2023-03-10
2023-03-13	1911.699951	2023-03-13
2023-03-14	1906.199951	2023-03-14
2023-03-15	1926.599976	2023-03-15

เช็คข้อมูลส่วนต้น

```
init_df.tail()
```

Price Date	Close	date
2026-03-09	5091.500000	2026-03-09
2026-03-10	5229.700195	2026-03-10
2026-03-11	5167.399902	2026-03-11
2026-03-12	5115.799805	2026-03-12
2026-03-13	NaN	2026-03-13

เช็คข้อมูลส่วนท้าย

พล็อตกราฟจากข้อมูลที่โหลดมา

```
# Let's preliminary see our data on the graphic
plt.style.use(style='ggplot')
plt.figure(figsize=(16,10))
plt.plot(init_df['Close'][0:])
plt.xlabel("days")
plt.ylabel("price")
plt.legend([f'Actual price for {STOCK}'])
plt.show()
```

กราฟจากข้อมูล

3.3 การเตรียมข้อมูล

ปรับสเกลข้อมูล (Data Scaling)

การปรับสเกลข้อมูลเป็นขั้นตอนสำคัญในการเตรียมข้อมูลก่อนนำไปฝึกโมเดลปัญญาประดิษฐ์ เนื่องจากข้อมูลราคาทองคำมีค่าค่อนข้างสูงและมีช่วงค่าที่กว้าง การปรับสเกลจึงช่วยให้โมเดลสามารถเรียนรู้รูปแบบของข้อมูลได้อย่างมีประสิทธิภาพมากขึ้น โดยในขั้นตอนนี้ประกอบด้วยกระบวนการดังต่อไปนี้

```
# Scale data for ML engine
init_df.dropna(inplace=True)
scaler = MinMaxScaler()
init_df['Close'] = scaler.fit_transform(np.expand_dims(init_df['Close'].values, axis=1))
```

1) การกำจัดค่าว่าง (dropna)

ในขั้นตอนแรกมีการตรวจสอบและลบแถวของข้อมูลที่มีค่าหายไปหรือค่า `NaN` ออกจากตารางข้อมูล โดยใช้คำสั่ง `dropna()` เพื่อให้ข้อมูลที่นำไปใช้ในการวิเคราะห์มีความสมบูรณ์ และป้องกันไม่ให้เกิดข้อผิดพลาดในระหว่างการคำนวณของโมเดลปัญญาประดิษฐ์

2) การเตรียมเครื่องมือสำหรับการปรับสเกล (MinMaxScaler)

มีการเรียกใช้เครื่องมือ `MinMaxScaler` ซึ่งเป็นวิธีการปรับขนาดข้อมูลให้ค่าทั้งหมดอยู่ในช่วงระหว่าง 0 ถึง 1 โดยเครื่องมือนี้จะทำหน้าที่แปลงค่าราคาทองคำจริง เช่น หลักพันหรือหลักหมื่นดอลลาร์ ให้กลายเป็นตัวเลขที่มีช่วงค่ามาตรฐาน ซึ่งช่วยให้โมเดล LSTM สามารถประมวลผลข้อมูลได้ง่ายและมีประสิทธิภาพมากขึ้น

3) การปรับขนาดและแปลงข้อมูล (fit_transform)

ก่อนทำการปรับสเกลข้อมูล มีการใช้คำสั่ง `np.expand_dims` เพื่อจัดรูปแบบข้อมูลจากลักษณะเวกเตอร์หนึ่งมิติให้กลายเป็นข้อมูลสองมิติ (2D) เนื่องจากเครื่องมือสำหรับการปรับสเกลกำหนดให้ข้อมูลต้องอยู่ในรูปแบบคอลัมน์ จากนั้นใช้คำสั่ง `fit_transform` เพื่อให้เครื่องมือเรียนรู้ค่าต่ำสุดและค่าสูงสุดของข้อมูลราคาปิด (Close) และทำการแปลงค่าทั้งหมดให้อยู่ในช่วงระหว่าง 0 ถึง 1 ในขั้นตอนเดียว

เรียกดูข้อมูลจัดเตรียม

init_df

Price Date	Close	date
2023-03-09	0.003627	2023-03-09
2023-03-10	0.012965	2023-03-10
2023-03-13	0.027157	2023-03-13
2023-03-14	0.025587	2023-03-14
2023-03-15	0.031412	2023-03-15
...	...	...
2026-03-06	0.950797	2026-03-06
2026-03-09	0.935205	2026-03-09
2026-03-10	0.974670	2026-03-10
2026-03-11	0.956879	2026-03-11
2026-03-12	0.942144	2026-03-12

757 rows × 2 columns

การเตรียมข้อมูลสำหรับปัญญาประดิษฐ์ (AI)

ขั้นตอนการเตรียมข้อมูลสำหรับโมเดลปัญญาประดิษฐ์เป็นกระบวนการสำคัญก่อนนำข้อมูลเข้าสู่การฝึกโมเดล LSTM โดยมีการจัดรูปแบบข้อมูลให้อยู่ในลักษณะที่เหมาะสมกับการเรียนรู้ของโมเดล ซึ่งประกอบด้วยขั้นตอนดังต่อไปนี้

```
#Prepare data for the engine
def PrepareData(days):
  df = init_df.copy()
  df['future'] = df['Close'].shift(-days)#คือการดึงราคาปิดในอีกวันข้างหน้า
  last_sequence = np.array(df[['Close']].tail(days))
  df.dropna(inplace=True)
  sequence_data = []
  sequences = deque(maxlen=N_STEPS)

  for entry, target in zip(df[['Close', 'date']].values, df['future'].values):
      sequences.append(entry)
      if len(sequences) == N_STEPS:
          sequence_data.append([np.array(sequences), target])

  last_sequence = list([s[:len(['Close'])] for s in sequences]) + list(last_sequence)
  last_sequence = np.array(last_sequence).astype(np.float32)

  # construct the X's and Y's
  X, Y = [], []
  for seq, target in sequence_data:
      X.append(seq)
      Y.append(target)

  # convert to numpy arrays
  X = np.array(X)
  Y = np.array(Y)

  return df, last_sequence, X, Y
```

1) การสร้างคำตอบ (Labeling)

ในขั้นตอนนี้มีการใช้คำสั่ง `df['future'] = df['Close'].shift(-days)` เพื่อเลื่อนค่าราคาปิด (Close) ในอนาคตขึ้นมาใช้เป็นค่าคำตอบของโมเดล โดยแนวคิดคือ หากโมเดลเห็นข้อมูลราคาย้อนหลังในช่วงเวลาหนึ่ง เช่น 7 วัน โมเดลจะต้องเรียนรู้ที่จะทำนายราคาที่จะเกิดขึ้นในอีกจำนวนวันที่กำหนดไว้ในอนาคต

2) การสร้างชุดข้อมูลด้วยเทคนิค Windowing
มีการใช้โครงสร้างข้อมูล `deque` ร่วมกับตัวแปร `N_STEPS` เพื่อสร้างชุดข้อมูลแบบลำดับเวลา โดยโปรแกรมจะเลื่อนหน้าต่างข้อมูล (window) ไปทีละช่วงเวลาและเก็บข้อมูลราคาย้อนหลัง 7 วันเป็นชุดข้อมูลนำเข้า (Input) พร้อมกับค่าราคาในอนาคตที่ใช้เป็นคำตอบ (Target) จนครบทั้งชุดข้อมูล
ตัวแปร `X` จะใช้เก็บข้อมูลราคาย้อนหลัง 7 วันสำหรับการทำนาย
ตัวแปร `Y` จะใช้เก็บค่าราคาเป้าหมายในอนาคตที่โมเดลต้องทำนาย

3) การเตรียมข้อมูลชุดล่าสุด (last_sequence)
ในขั้นตอนสุดท้ายมีการเก็บข้อมูลราคาล่าสุดย้อนหลัง 7 วันไว้ในตัวแปร `last_sequence` เพื่อนำไปใช้ในการพยากรณ์ค่าจริงในอนาคต หลังจากที่โมเดล LSTM ผ่านกระบวนการฝึก (Training) เรียบร้อยแล้ว

ฟังก์ชันนี้จะเปลี่ยนตารางราคาธรรมดา ให้กลายเป็น คู่โจทย์-คำตอบ (X, Y) จำนวนมาก เพื่อเอาไปป้อนให้โมเดล LSTM

3.4 การสร้างและฝึกโมเดลปัญญาประดิษฐ์

ในขั้นตอนนี้เป็นการสร้างและฝึกโมเดลปัญญาประดิษฐ์ประเภท LSTM (Long Short-Term Memory) เพื่อใช้ในการเรียนรู้รูปแบบการเปลี่ยนแปลงของราคาทองคำจากข้อมูลในอดีต และนำไปใช้สำหรับการพยากรณ์ราคาในอนาคต โดยมีรายละเอียดดังต่อไปนี้

```
#Machine learning model
def GetTrainedModel(x_train, y_train):
  model = Sequential()
  model.add(LSTM(60, return_sequences=True, input_shape=(N_STEPS, len(['Close']))))
  model.add(Dropout(0.3))
  model.add(LSTM(120, return_sequences=False))
  model.add(Dropout(0.3))
  model.add(Dense(20))
  model.add(Dense(1))

  BATCH_SIZE = 8
  EPOCHS = 80 # Reduced for faster execution จากเดิม80

  model.compile(loss='mean_squared_error', optimizer='adam')

  model.fit(x_train, y_train,
          batch_size=BATCH_SIZE,
          epochs=EPOCHS,
          verbose=1)

  model.summary()

  return model
```

1) โครงสร้างของโมเดล (Model Architecture)
โมเดลที่ใช้เป็นโครงข่ายประสาทเทียมแบบ LSTM จำนวน 2 ชั้น โดยชั้นแรกประกอบด้วยหน่วยประมวลผลจำนวน 60 หน่วย และชั้นที่สองมีจำนวน 120 หน่วย เพื่อช่วยให้โมเดลสามารถเรียนรู้และจดจำรูปแบบของข้อมูลลำดับเวลาได้อย่างมีประสิทธิภาพมากขึ้น

นอกจากนี้ยังมีการใช้เทคนิค `Dropout` ที่กำหนดค่าเป็น `0.3` เพื่อสุ่มละเว้นข้อมูลบางส่วนระหว่างกระบวนการฝึกโมเดล ซึ่งช่วยลดปัญหาการจดจำข้อมูลมากเกินไป (Overfitting) และทำให้โมเดลสามารถนำความรู้ไปใช้กับข้อมูลใหม่ได้ดีขึ้น

ในส่วนสุดท้ายของโมเดลมีการใช้ชั้น `Dense` ซึ่งทำหน้าที่เป็นชั้นเอาต์พุตสำหรับสรุปผลลัพธ์ออกมาเป็นค่าราคาที่ต้องการพยากรณ์

2) การตั้งค่าการเรียนรู้ของโมเดล

ในการฝึกโมเดลมีการกำหนดฟังก์ชันการสูญเสีย (Loss Function) เป็น `Mean Squared Error (MSE)` ซึ่งใช้วัดความคลาดเคลื่อนระหว่างค่าที่โมเดลพยากรณ์กับค่าจริง โดยค่าความคลาดเคลื่อนที่ต่ำแสดงว่าโมเดลสามารถพยากรณ์ค่าได้ใกล้เคียงกับความเป็นจริงมากขึ้น

นอกจากนี้ยังใช้ตัวปรับพารามิเตอร์ `Adam Optimizer` ซึ่งช่วยให้กระบวนการเรียนรู้ของโมเดลมีประสิทธิภาพและรวดเร็วขึ้น

ทั้งนี้ได้กำหนดจำนวนรอบการเรียนรู้ (Epochs) เท่ากับ 80 รอบ เพื่อให้โมเดลได้เรียนรู้จากข้อมูลชุดเดิมซ้ำหลายครั้งและสามารถปรับค่าพารามิเตอร์ให้เหมาะสมมากยิ่งขึ้น

3) การเริ่มกระบวนการฝึกโมเดล (Training)

ขั้นตอนสุดท้ายเป็นการเริ่มกระบวนการฝึกโมเดลโดยใช้คำสั่ง `model.fit` ซึ่งเป็นคำสั่งสำหรับป้อนข้อมูลชุดฝึกเข้าสู่โมเดล

โดยข้อมูล `x_train` จะใช้เป็นข้อมูลนำเข้า (Input) หรือโจทย์สำหรับการเรียนรู้

ส่วนข้อมูล `y_train` จะใช้เป็นค่าคำตอบ (Target) ที่โมเดลต้องพยายามเรียนรู้และทำนายให้ใกล้เคียงที่สุด

กระบวนการฝึกจะดำเนินการตามจำนวนรอบที่กำหนดไว้เพื่อให้โมเดลสามารถเรียนรู้รูปแบบของข้อมูลได้อย่างมีประสิทธิภาพ

การพยากรณ์และสรุปผลลัพธ์

หลังจากที่โมเดลปัญญาประดิษฐ์ได้ผ่านกระบวนการฝึก (Training) แล้ว ขั้นตอนถัดไปคือการนำโมเดลมาใช้ในการพยากรณ์ราคาทองคำในอนาคต พร้อมทั้งจัดรูปแบบผลลัพธ์ให้อยู่ในลักษณะที่สามารถนำไปวิเคราะห์และแสดงผลได้อย่างชัดเจน โดยมีรายละเอียดดังต่อไปนี้

```
# GET PREDICTIONS
predictions = []
historical_prediction_data_list = [] # List to store (date, prediction) for historical predictions

for step in LOOKUP_STEPS:
  df_current_step, last_sequence_current_step, x_train_current_step, y_train_current_step = PrepareData(step)
  # Ensure x_train has the correct shape for the model (N_STEPS, num_features)
  x_train_current_step = x_train_current_step[:, :, :len(['Close'])].astype(np.float32)

  model_current_step = GetTrainedModel(x_train_current_step, y_train_current_step)

  # If this is for step = 1, generate historical predictions using the trained model
  if step == 1:
    predicted_on_train_scaled = model_current_step.predict(x_train_current_step)
    historical_predictions_unscaled = scaler.inverse_transform(predicted_on_train_scaled).flatten()

    # Determine the dates for these historical predictions
    # Y targets (y_train_current_step) correspond to the 'future' price for a given date in df_current_step
    # The index for y_train[i] is df_current_step.index[N_STEPS - 1 + i]
    historical_prediction_dates = df_current_step.index[N_STEPS - 1 : N_STEPS - 1 + len(historical_predictions_unscaled)]

    for i, pred_val in enumerate(historical_predictions_unscaled):
        historical_prediction_data_list.append((historical_prediction_dates[i], pred_val))

  # Predict for the actual future using the last sequence from the current step's data
  last_sequence = last_sequence_current_step[-N_STEPS:]
  last_sequence = np.expand_dims(last_sequence, axis=0)
  prediction = model_current_step.predict(last_sequence)
  predicted_price = scaler.inverse_transform(prediction)[0][0]

  predictions.append(round(float(predicted_price), 2))

# After the loop, reconstruct final_df with actuals, historical predictions, and future predictions

# Define df_actual from init_df for plotting
df_actual = init_df.copy()
df_actual['Close_actual'] = scaler.inverse_transform(df_actual[['Close']]).flatten()
df_actual.drop(columns=['Close'], inplace=True)
```

```python
# 1. Create DataFrame for historical predictions
df_historical_predictions = pd.DataFrame(historical_prediction_data_list, columns=['Date', 'predicted_close_hist']).set_index('Date')
df_historical_predictions.index = pd.to_datetime(df_historical_predictions.index) # Ensure datetime index

# 2. Create DataFrame for future predictions
# `date_now` is a string 'YYYY-MM-DD'. `LOOKUP_STEPS` is [1, 2, 3].
# The future dates are `date_now` + 1 day, `date_now` + 2 days, etc.
future_dates = [dt.datetime.strptime(date_now, '%Y-%m-%d') + dt.timedelta(days=s) for s in LOOKUP_STEPS]
df_future_predictions = pd.DataFrame({'predicted_close': predictions}, index=future_dates)
df_future_predictions.index = pd.to_datetime(df_future_predictions.index) # Ensure datetime index

# 3. Get actual prices (df_actual from kernel state is already unscaled and has 'Close_actual')
df_actual_for_final = df_actual.copy()
df_actual_for_final.index = pd.to_datetime(df_actual_for_final.index) # Ensure datetime index

# 4. Combine all data into final_df
# Start with actual data
final_df = df_actual_for_final.copy()

# Merge historical predictions into final_df
# Use a left merge to keep all actual dates and add historical predictions where available
final_df = pd.merge(final_df, df_historical_predictions, left_index=True, right_index=True, how='left')

# Populate the 'predicted_close' column: start with historical predictions
final_df['predicted_close'] = final_df['predicted_close_hist']
final_df.drop(columns=['predicted_close_hist'], inplace=True) # Remove the temporary column

# Add future predictions to the 'predicted_close' column
# This will update predicted_close for future dates and add new rows if they don't exist
for date, pred_val in df_future_predictions['predicted_close'].items():
    final_df.loc[date, 'predicted_close'] = pred_val
    # For new future dates, 'Close_actual' will remain NaN as no actual data exists

final_df.sort_index(inplace=True) # Ensure chronological order for plotting
```

```python
# --- Display results ---
if len(predictions) > 0:
    predictions_list = [str(d) + '$' for d in predictions]
    predictions_str = ', '.join(predictions_list)
    message = f'{STOCK} prediction for upcoming {len(LOOKUP_STEPS)} days ({predictions_str})'
    print(message)
```

1) การพยากรณ์ราคา (Prediction)

โปรแกรมจะทำการวนลูปตามจำนวนวันที่ต้องการพยากรณ์ ซึ่งกำหนดไว้ใน `LOOKUP_STEPS` เช่น 1,2,3 วันข้างหน้า โดยในแต่ละรอบจะมีการเรียกใช้โมเดลที่ผ่านการฝึกเพื่อพยากรณ์ราคาทองคำในอนาคต จากนั้นใช้คำสั่ง `inverse_transform` เพื่อแปลงค่าที่ได้จากโมเดลซึ่งอยู่ในรูปแบบข้อมูลที่ถูกปรับสเกล (ช่วงค่า 0 ถึง 1) ให้กลับมาเป็นค่าราคาทองคำจริงในหน่วยดอลลาร์สหรัฐ เพื่อให้สามารถนำไปตีความและใช้งานได้ง่าย ตีความและใช้งานได้ง่าย

2) การสร้างตารางเปรียบเทียบผลลัพธ์

มีการสร้างโครงสร้างข้อมูลแบบ `DataFrame` เพื่อใช้เก็บข้อมูลสำหรับการเปรียบเทียบผลลัพธ์ โดยประกอบด้วยข้อมูลสำคัญสามส่วน ได้แก่ (1.) ราคาจริงในอดีต (Actual)

(2.) ราคาที่โมเดลเคยพยากรณ์จากข้อมูลในอดีต (Historical Prediction) ใช้สำหรับประเมินความแม่นยำของโมเดล

(3.) ราคาที่พยากรณ์ในอนาคต (Future Prediction) เป็นค่าที่ต้องการนำไปใช้ในการวิเคราะห์แนวโน้มของราคาทองคำ

3) การจัดการข้อมูลวันที่ในอนาคต (Future Dates)
โปรแกรมมีการคำนวณวันที่ในอนาคตโดยใช้คำสั่ง `timedelta` เพื่อเพิ่มจำนวนวันจากวันที่ล่าสุดในชุดข้อมูลเดิม ทำให้สามารถระบุได้อย่างชัดเจนว่าค่าราคาที่โมเดลพยากรณ์ออกมานั้นเป็นราคาของวันใดในอนาคต

4) การแสดงผลลัพธ์ของการพยากรณ์
ในขั้นตอนสุดท้ายโปรแกรมจะแสดงผลลัพธ์การพยากรณ์ออกมาในรูปแบบข้อความ
โดยใช้คำสั่ง `print` เพื่อสรุปค่าราคาทองคำที่โมเดลคาดการณ์ในช่วงวันข้างหน้า เช่น
`GC=F prediction for upcoming 3 days ($2XXX.X, $2XXX.X, $2XXX.X)`
ซึ่งช่วยให้ผู้ใช้งานสามารถเห็นผลการพยากรณ์ได้อย่างชัดเจนและเข้าใจได้ง่าย

```
Model: "sequential_20"
```

Layer (type)	Output Shape	Param #
lstm_40 (LSTM)	(None, 7, 60)	14,880
dropout_40 (Dropout)	(None, 7, 60)	0
lstm_41 (LSTM)	(None, 120)	86,880
dropout_41 (Dropout)	(None, 120)	0
dense_40 (Dense)	(None, 20)	2,420
dense_41 (Dense)	(None, 1)	21

```
Total params: 312,605 (1.19 MB)
Trainable params: 104,201 (407.04 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 208,404 (814.08 KB)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 332ms/step
GC=F prediction for upcoming 3 days (5138.34$, 5231.41$, 5077.38$)
```

การสร้างกราฟวิเคราะห์ผล (Data Visualization)

เพื่อให้สามารถวิเคราะห์ประสิทธิภาพของแบบจำลองปัญญาประดิษฐ์และมองเห็นแนวโน้มของราคาทองคำได้อย่างชัดเจน จึงมีการสร้างกราฟสำหรับแสดงผลลัพธ์ของการพยากรณ์ โดยใช้เครื่องมือสำหรับการสร้างกราฟในภาษา Python ซึ่งช่วยให้สามารถเปรียบเทียบค่าราคาจริงกับค่าที่โมเดลพยากรณ์ได้อย่างเป็นรูปธรรม รายละเอียดของการแสดงผลมีดังนี้

```
# --- 4. วาดกราฟ (เวอร์ชันแยกเส้นแดงออกจากเส้นน้ำเงิน) ---
plt.style.use('ggplot')
plt.figure(figsize=(16, 8))

# 1. เส้นราคาจริง (สีน้ำเงิน)
plt.plot(final_df.index[-60:], final_df['Close_actual'].tail(60),
         label=f'Actual Price ({STOCK})', color='royalblue', linewidth=2)

# 2. เส้นทำนายย้อนหลัง (สีส้ม) - ใช้ค่าจากช่วง x_train เดิม
plt.plot(final_df.index[-60:-3], final_df['predicted_close'].tail(60).iloc[:-3],
         label='AI Historical Prediction', color='orange', linestyle='--', alpha=0.7)

# 3. เส้นทำนายอนาคต (สีแดง) - ตัดจุดเชื่อมออกเพื่อให้ลอยแยกจากเส้นน้ำเงิน
# เราดึงเฉพาะ 3 วันล่าสุดที่เป็นค่าทำนายอนาคตเพียวๆ
forecast_only = final_df['predicted_close'].tail(3)

plt.plot(forecast_only.index, forecast_only.values,
         label='Future 3-Day Forecast', color='crimson', linewidth=3,
         marker='o', markersize=8, linestyle='-')

# การตั้งค่ากราฟเพิ่มเติม
plt.title(f'Gold Price Analysis & Forecast: {STOCK}', fontsize=18)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Price (USD)', fontsize=12)
plt.legend(loc='upper left')
plt.grid(True, which='both', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

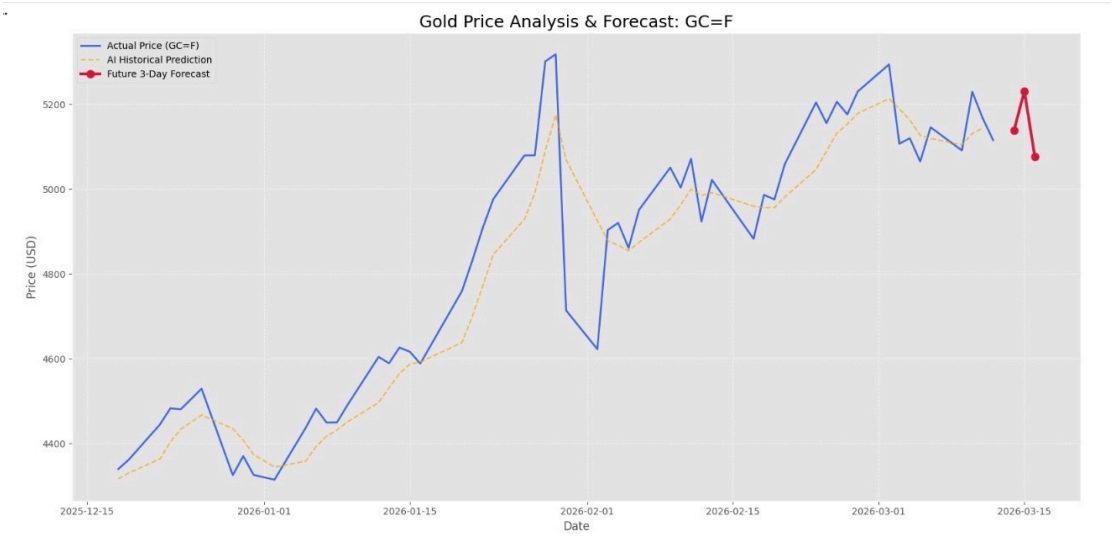
1) การกำหนดรูปแบบและขนาดของกราฟ

ในการสร้างกราฟมีการกำหนดรูปแบบกราฟโดยใช้สไตล์ `ggplot` ซึ่งช่วยให้กราฟมีความเป็นระเบียบและมีเส้นตารางสีเทาเพื่อช่วยในการอ่านค่าข้อมูล นอกจากนี้ยังมีการกำหนดขนาดของกราฟให้มีขนาดใหญ่ (16 × 8) เพื่อให้สามารถมองเห็นรายละเอียดของแนวโน้มราคาทองคำได้อย่างชัดเจน

2) การแยกเส้นข้อมูลเพื่อการเปรียบเทียบ
กราฟที่สร้างขึ้นมีการแยกเส้นข้อมูลออกเป็นสามประเภทเพื่อให้สามารถวิเคราะห์ข้อมูลได้ง่ายขึ้น ได้แก่

- **Actual Price (สีน้ำเงิน)** แสดงราคาทองคำจริงย้อนหลังประมาณ 60 วัน เพื่อใช้เป็นข้อมูลอ้างอิงของแนวโน้มราคาจริงในตลาด
- **AI Historical Prediction (สีส้ม)** แสดงค่าที่โมเดลปัญญาประดิษฐ์เคยพยากรณ์จากข้อมูลในอดีต โดยแสดงเป็นเส้นประ เพื่อใช้เปรียบเทียบกับราคาจริงว่าการพยากรณ์มีความใกล้เคียงเพียงใด
- **Future 3-Day Forecast (สีแดง)** แสดงค่าราคาทองคำที่โมเดลพยากรณ์สำหรับ 3 วันข้างหน้า ซึ่งเป็นข้อมูลสำคัญสำหรับการวิเคราะห์แนวโน้มในอนาคต โดยมีการเพิ่มความหนาของเส้นและใช้จุดมาร์กเกอร์รูปวงกลมเพื่อให้สังเกตได้ง่าย

3) การแสดงรายละเอียดเพิ่มเติมบนกราฟ
เพื่อให้กราฟสามารถตีความได้ง่าย มีการเพิ่มองค์ประกอบสำคัญ เช่น การแสดง *Legend* หรือคำอธิบายสัญลักษณ์ไว้ที่มุมบนซ้ายของกราฟ รวมทั้งมีการกำหนดชื่อกราฟและระบุหน่วยของราคาทองคำเป็นดอลลาร์สหรัฐ (USD) เพื่อให้ผู้ใช้งานสามารถเข้าใจผลลัพธ์ได้อย่างชัดเจน



การวัดประสิทธิภาพของโมเดล (Evaluation)

หลังจากที่แบบจำลองปัญญาประดิษฐ์ได้ผ่านกระบวนการฝึกแล้ว ขั้นตอนสำคัญถัดไปคือการประเมินประสิทธิภาพของโมเดล เพื่อพิจารณาว่าโมเดลสามารถพยากรณ์ราคาทองคำได้แม่นยำมากน้อยเพียงใด โดยใช้ตัวชี้วัดทางสถิติในการประเมินผลลัพธ์ที่ได้จากการพยากรณ์ ซึ่งมีรายละเอียดดังต่อไปนี้

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

# 1. ทำนายค่าจากข้อมูลชุดทดสอบ (X_test)
# Using x_train_current_step for prediction as no separate test set is defined
y_pred_scaled = model_current_step.predict(x_train_current_step)

# 2. แปลงค่ากลับเป็นราคาทองจริง (Unscale)
# y_train_current_step contains the actual scaled values for the training set
y_true = scaler.inverse_transform(y_train_current_step.reshape(-1, 1)).flatten()
y_pred = scaler.inverse_transform(y_pred_scaled).flatten()

# 3. คำนวณค่า Error ต่างๆ
mae = mean_absolute_error(y_true, y_pred)
mse = mean_squared_error(y_true, y_pred)
rmse = np.sqrt(mse)

# 4. คำนวณเปอร์เซ็นต์ความคลาดเคลื่อน (MAPE) และความแม่นยำ (Accuracy)
# Add a small epsilon to y_true to avoid division by zero
epsilon = 1e-10
mape = np.mean(np.abs((y_true - y_pred) / (y_true + epsilon))) * 100 # Added epsilon for stability
accuracy = 100 - mape

print(f"--- ผลการทดสอบความแม่นยำ (Error Metrics) ---")
print(f"Mean Absolute Error (MAE): {mae:.2f} $")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f} $")
print(f"ค่าความคลาดเคลื่อนเฉลี่ย (MAPE): {mape:.2f} %")
print(f"เปอร์เซ็นต์ความแม่นยำ (Accuracy): {accuracy:.2f} %")
# The error metrics have been calculated and are presented above.
```

```
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
--- ผลการทดสอบความแม่นยำ (Error Metrics) ---
Mean Absolute Error (MAE): 50.75 $
Root Mean Squared Error (RMSE): 77.88 $
ค่าความคลาดเคลื่อนเฉลี่ย (MAPE): 1.65 %
เปอร์เซ็นต์ความแม่นยำ (Accuracy): 98.35 %
```

1) การแปลงค่าข้อมูลกลับสู่หน่วยจริง (Unscale)

ก่อนที่จะทำการคำนวณค่าความผิดพลาดของการพยากรณ์ จำเป็นต้องแปลงค่าข้อมูลจากรูปแบบที่ถูกปรับสเกลในช่วง 0 ถึง 1 ให้กลับมาเป็นค่าราคาทองคำจริงในหน่วยดอลลาร์สหรัฐเสียก่อน เพื่อให้สามารถตีความความคลาดเคลื่อนของการพยากรณ์ได้ในหน่วยของเงินจริง

2) ตัวชี้วัดความคลาดเคลื่อนของการพยากรณ์ (Error Metrics)
ในการประเมินประสิทธิภาพของโมเดล มีการใช้ตัวชี้วัดความคลาดเคลื่อนที่สำคัญสามประเภท ได้แก่

- **MAE (Mean Absolute Error)** เป็นค่าความคลาดเคลื่อนสัมบูรณ์เฉลี่ย ซึ่งแสดงให้เห็นว่าโดยเฉลี่ยแล้วโมเดลพยากรณ์ราคาทองคำคลาดเคลื่อนไปจากค่าจริงกี่ดอลลาร์
- **RMSE (Root Mean Squared Error)** เป็นค่ารากที่สองของค่าเฉลี่ยกำลังสองของความคลาดเคลื่อน ซึ่งจะให้ความสำคัญกับค่าความผิดพลาดที่มีขนาดใหญ่เป็นพิเศษ ทำให้สามารถสะท้อนผลกระทบของความผิดพลาดขนาดใหญ่ได้ชัดเจนยิ่งขึ้น
- **MAPE (Mean Absolute Percentage Error)** เป็นค่าความคลาดเคลื่อนเฉลี่ยในรูปแบบเปอร์เซ็นต์ ซึ่งช่วยให้สามารถเข้าใจระดับความคลาดเคลื่อนของการพยากรณ์ได้ง่าย เช่น การพยากรณ์ผิดพลาดประมาณ 1%

3) การคำนวณค่าความแม่นยำของโมเดล
เพื่อให้ผลลัพธ์สามารถตีความได้ง่าย จึงมีการคำนวณค่าความแม่นยำของโมเดลจากสมการ

$$Accuracy = 100 - MAPE$$

ซึ่งช่วยให้สามารถสรุประดับความแม่นยำของการพยากรณ์ออกมาในรูปแบบเปอร์เซ็นต์ เช่น ความแม่นยำ 98.5% เป็นต้น นอกจากนี้ยังมีการกำหนดค่า *epsilon* ซึ่งเป็นค่าตัวเลขขนาดเล็กมาก เพื่อป้องกันปัญหาการหารด้วยศูนย์ที่อาจทำให้โปรแกรมเกิดข้อผิดพลาดในระหว่างการคำนวณ

4 สรุปและอภิปราย

4.1 สรุปผล

จากการทดลองพบว่า การใช้โครงข่ายประสาทเทียมแบบ LSTM สามารถใช้ในการทำนายราคาทองคำได้อย่างมีประสิทธิภาพ โดยโมเดลสามารถเรียนรู้รูปแบบของข้อมูลในอดีตและนำมาคาดการณ์แนวโน้มของราคาทองคำในอนาคตได้

4.2 อภิปรายผล

หลังจากทำการฝึกโมเดลแล้ว สามารถนำโมเดลไปทำนายราคาทองคำและเปรียบเทียบกับราคาจริงได้ โดยแสดงผลในรูปแบบกราฟ ผลลัพธ์แสดงให้เห็นว่าโมเดลสามารถทำนายแนวโน้มของราคาทองคำได้ใกล้เคียงกับข้อมูลจริง

4.3 ข้อเสนอแนะ

1. ควรเพิ่มตัวแปรทางเศรษฐกิจอื่น ๆ เช่น ค่าเงินดอลลาร์ หรืออัตราดอกเบี้ย
2. ควรใช้ข้อมูลที่มีระยะเวลายาวขึ้นเพื่อเพิ่มความแม่นยำของโมเดล
3. อาจทดลองใช้โมเดลอื่น เช่น GRU หรือ Transformer เพื่อเปรียบเทียบประสิทธิภาพ

เอกสารอ้างอิง

[1] Sazonov, D. (2021). *How to Predict Stock Market Using Google TensorFlow and LSTM Neural Network*. Medium. สืบค้นจาก https://medium.com/@dmytrosazonov/how-to-predict-stock-market-using-google-tensorflow-and-lstm-neural-network-81ccc41a22a8

[2] Python Software Foundation. (2024). *Welcome to Python.org*. สืบค้นจาก https://www.python.org

[3] QuantStart. (2024). *Python Libraries for Quantitative Trading*. สืบค้นจาก https://www.quantstart.com/articles/python-libraries-for-quantitative-trading/

[4] Google Finance. (2024). *Gold Futures (GCW00:COMEX)*. สืบค้นจาก https://g.co/finance/GCW00:COMEX

[5] Google Share. (2024). *Online Resource for Financial Data*. สืบค้นจาก https://share.google/PGdkeFECX6fdQPVUJ