การพัฒนาแบบจำลองทำนายอนุกรมเวลาโดยใช้อัลกอริทึมเชิงเส้นและไม่เชิงเส้นเพื่อพยากรณ์มลพิษจากฝุ่นละอองอนุภาคเล็ก

นายอนุพงษ์ บรรจงการ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคมและคอมพิวเตอร์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2565

THE DEVELOPMENT OF TIME SERIES PREDICTIVE MODEL USING LINEAR AND NONLINEAR ALGORITHMS TO FORECAST PARTICLE POLLUTION

ANUPONG BANJONGKAN

A Thesis Submitted in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy in
Telecommunication and Computer Engineering
Suranaree University of Technology
Academic Year 2022

การพัฒนาแบบจำลองทำนายอนุกรมเวลาโดยใช้อัลกอริทึมเชิงเส้นและไม่เชิงเส้นเพื่อพยากรณ์มลพิษจากฝุ่นละอองอนุภาคเล็ก

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาดุษฎีบัณฑิต

คณะกรรมการสอบวิทยานิพนธ์

.....
(ดร.กีระชาติ สุขสุทธิ์)
ประธานกรรมการ

.....
(รศ.ดร.กิตติศักดิ์ เกิดประสพ)
กรรมการ (อาจารย์ที่ปรึกษาวิทยานิพนธ์)

.....
(รศ.ดร.นิตยา เกิดประสพ)
กรรมการ

.....
(ผศ.ดร.นันทวุฒิ คะอังกุ)
กรรมการ

.....
(ดร.รติพร จันทร์กลั่น)
กรรมการ

.....
(รศ.ดร.ฉัตรชัย โชติษฐยางกูร)
รองอธิการบดีฝ่ายวิชาการและประกันคุณภาพ

.....
(รศ.ดร.พรศิริ จงกล)
คณบดีสำนักวิชาวิศวกรรมศาสตร์

อนุพงษ์ บรรจงการ: การพัฒนาแบบจำลองทำนายอนุกรมเวลาโดยใช้อัลกอริทึมเชิงเส้นและไม่เชิงเส้นเพื่อพยากรณ์มลพิษจากฝุ่นละอองอนุภาคเล็ก (THE DEVELOPMENT OF TIME SERIES PREDICTIVE MODEL USING LINEAR AND NONLINEAR ALGORITHMS TO FORECAST PARTICLE POLLUTION) อาจารย์ที่ปรึกษา: รองศาสตราจารย์ ดร.กิตติศักดิ์ เกิดประสพ, 207 หน้า.

คำสำคัญ: การเรียนรู้ของเครื่อง/แบบจำลองอนุกรมเวลา/ฝุ่นละอองอนุภาคเล็ก/ANFIS/ARIMA

ปัญหามลพิษทางอากาศเป็นปัญหาที่ทั่วโลกให้ความสำคัญ เพราะเป็นปัญหาที่ส่งผลกระทบต่อเศรษฐกิจและสังคมโดยกว้าง โดยเฉพาะอย่างยิ่งสุขภาพของประชาชนที่มลพิษทางอากาศส่งผลอันตรายที่ก่อให้เกิดการเสียชีวิตก่อนวัยอันควรด้วยปัจจัยจากโรคร้ายต่าง ๆ เช่น โรคมะเร็งปอด โรคหัวใจ หรือโรคหลอดเลือดสมอง เป็นต้น สารมลพิษทางอากาศที่งานวิจัยนี้ให้ความสนใจคือ Particulate Matter: PM ที่มีขนาดเส้นผ่านศูนย์กลางน้อยกว่า 2.5 ไมโครเมตร เรียกว่า PM2.5 ดังนั้น การตระหนักรู้ถึงข้อมูลฝุ่น PM2.5 แบบทันทีและแบบล่วงหน้า เป็นประเด็นที่สำคัญอย่างยิ่งต่อการบริหารจัดการปัญหาฝุ่น PM2.5

งานวิจัยนี้ได้นำเสนออัลกอริทึมผสมระหว่างอัลกอริทึมเชิงเส้นและไม่เชิงเส้น สำหรับพัฒนาแบบจำลองอนุกรมเวลาเพื่อพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ด้วยการพัฒนาแบบจำลองผ่านกระบวนการเรียนรู้ของเครื่อง โดยใช้ชุดข้อมูลฝุ่น PM2.5 จากสถานีวัดคุณภาพอากาศภาคพื้นในเขตพื้นที่ของจังหวัดระยอง เป็นชุดข้อมูลสำหรับการวิจัย ในส่วนของกระบวนการเตรียมข้อมูล งานวิจัยนี้ได้เลือกใช้วิธีการเติมข้อมูลที่หายไป (Missing Values) ด้วยเทคนิคเชิงเวลาและพื้นที่เรียกว่า Temporal and Spatial Average Value: TSA เพื่อให้ได้ชุดข้อมูลอนุกรมเวลาฝุ่น PM2.5 ที่สมบูรณ์ที่สุด สำหรับการทดสอบและประเมินประสิทธิภาพแบบจำลองอนุกรมเวลาผสมที่นำเสนอ โดยเปรียบเทียบประสิทธิภาพกับแบบจำลองอนุกรมเวลาเดี่ยวเชิงเส้นและไม่เชิงเส้น ได้แก่ Autoregressive Integrated Moving Average: ARIMA, Artificial Neural Network: ANN, Long Short-Term Memory: LSTM และ Adaptive Neuro-Fuzzy Inference System: ANFIS ผ่านมาตรวัดความคาดเคลื่อน 5 มาตรวัด ได้แก่ MAE, MAPE, RMSE, %RMSE และ Coefficient of Determination: R^2 สุดท้ายปรับปรุงประสิทธิภาพแบบจำลองผ่านกระบวนการ Hyperparameters Tunning ด้วยเทคนิคการทำ Optimization

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนักศึกษา.....

ปีการศึกษา 2565 ลายมือชื่ออาจารย์ที่ปรึกษา.....

ANUPONG BANJONGKAN: THE DEVELOPMENT OF TIME SERIES PREDICTIVE MODEL USING LINEAR AND NONLINEAR ALGORITHMS TO FORECAST PARTICLE POLLUTION. THESIS ADVISOR: ASSOC. PROF. KITTISAK KERDPRASOP, Ph.D. 207 PP.

Keyword: ANFIS/ARIMA/Machine Learning/Particulate Matter/Time Series Model

The problem of air pollution is a problem that is a global concern because it is a problem that affects the economy and society in general. A special concern is public health where air pollution has a dangerous effect that causes premature death from factors relating to various serious diseases such as lung cancer, heart disease, stroke etc. The air pollutant that this research is interested in is Particulate Matter: PM with a diameter of less than 2.5 micrometers, known as PM2.5. Therefore, real-time and early awareness of PM2.5 information is a very important key for the management of PM2.5 problems.

This research presents a linear and nonlinear hybrid algorithm for modeling the PM2.5 forecasting model. The PM2.5 data used in this research are the average daily air quality measurement recorded from ground-based stations in the Rayong province. As part of the data preparation process, a Temporal and Spatial Average value: TSA technique was used to handle the missing values problem. The performance of the proposed hybrid model was compared against three standard time series models including Autoregressive Integrated Moving Average: ARIMA, Artificial Neural Network: ANN, Long Short-Term Memory: LSTM, and Adaptive Neuro-Fuzzy Inference System: ANFIS. The assessment to consider the error in forecasting are Mean Absolute Error: MAE, Mean Absolute Percentage Error: MAPE, Root Mean Square Error: RMSE, Percentage Root Mean Square Error: %RMSE, and Coefficient of Determination: R^2. Finally, the performance of the hybrid model was improved through the process of hyperparameter tunning with optimization techniques.

School of Computer Engineering
Academic Year 2022

Student's Signature Anupong Banjongkan
Advisor's Signature Kittisak Kerdprasop

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สำเร็จลุล่วงด้วยดี ผู้วิจัยขอกราบขอบพระคุณ บุคคล และกลุ่มบุคคลต่าง ๆ ที่ได้กรุณาให้คำปรึกษา แนะนำ ช่วยเหลืออย่างดียิ่ง ทั้งในด้านวิชาการ และด้านการดำเนินงานวิจัย ดังต่อไปนี้

รองศาสตราจารย์ ดร.กิตติศักดิ์ เกิดประสพ และรองศาสตราจารย์ ดร.นิตยา เกิดประสพ ที่ให้คำปรึกษาในการดำเนินงานวิจัย การจัดการรูปแบบ และช่วยตรวจทานความถูกต้องของวิทยานิพนธ์ คุณปราณี กฐินใหม่ เลขานุการสาขาวิชาวิศวกรรมคอมพิวเตอร์ ที่ให้ความช่วยเหลือในการประสานงานด้านเอกสารระหว่างศึกษา

ขอขอบคุณนักศึกษาบัณฑิตสาขาวิชาวิศวกรรมคอมพิวเตอร์ ทุกท่านที่ให้คำปรึกษา ช่วยตรวจทานความถูกต้องและช่วยเหลือด้วยดีมาโดยตลอด

และขอขอบคุณภรรยา คุณญาณิกา บรรจงการ และครอบครัวของผู้วิจัยที่ค่อยให้กำลังใจและสนับสนุนในเรื่องต่าง ๆ ตลอดระยะเวลาการศึกษา

นอกจากนี้ขอขอบคุณครู อาจารย์ทั้งในอดีตและปัจจุบันที่ให้ความรู้แก่ผู้วิจัยจนประสบความสำเร็จในชีวิต

ท้ายที่สุดที่จะลืมมิได้ ขอกราบขอบพระคุณ บิดา มารดา ที่ให้กำเนิด อบรม เลี้ยงดูด้วยความรัก และส่งเสริมการศึกษาเป็นอย่างดีโดยตลอด ทำให้ผู้วิจัยมีความรู้ ความสามารถ มีจิตใจที่เข้มแข็ง รวมทั้งเป็นกำลังใจที่ยิ่งใหญ่แก่ผู้วิจัย จนทำให้ผู้วิจัยประสบความสำเร็จในชีวิตเรื่อยมา

อนุพงษ์ บรรจงการ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

บทที่

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหาการวิจัย

ปัญหามลพิษทางอากาศเป็นปัญหาที่ทุกประเทศต่างให้ความสำคัญ เพราะเป็นปัญหาที่ส่งผลกระทบโดยตรงต่อคุณภาพชีวิตของประชาชนโดยกว้าง โดยเฉพาะมลพิษทางอากาศในเขตเมืองใหญ่และเขตเมืองอุตสาหกรรม จากข้อมูลปี ค.ศ. 2016 องค์การอนามัยโลก รายงานว่ามีประชาชนจำนวน 4.2 ล้านคนทั่วโลกเสียชีวิตก่อนวัยอันควรจากปัญหามลพิษทางอากาศ (World Health Organization [WHO], 2016) และในประเทศไทยคาดการณ์ว่าจะมีผู้เสียชีวิตจากผลกระทบของมลพิษทางอากาศประมาณ 50,000 คน (กรมควบคุมมลพิษ [คพ], 2020) สารมลพิษทางอากาศที่ถูกกำหนดให้เป็นอันตรายต่อประชาชนและต้องมีการเฝ้าระวังได้แก่ ฝุ่นละอองขนาดเล็ก (Particulate Matter: PM) ก๊าซโอโซน (Ozone: O_3), ก๊าซไนโตรเจนไดออกไซด์ (Nitrogen Dioxide: NO_2) ก๊าซซัลเฟอร์ไดออกไซด์ (Sulfur Dioxide: SO_2) และก๊าซคาร์บอนมอนอกไซด์ (Carbon Monoxide: CO) โดยเฉพาะอย่างยิ่งสารมลพิษทางอากาศ PM เป็นสารมลพิษที่ทั่วโลกให้ความสำคัญและเร่งแก้ไขเพราะมีอันตรายต่อร่างกายทั้งในระยะสั้น และระยะยาวมากที่สุดประเภทหนึ่ง

PM คือฝุ่นละอองขนาดเล็กที่ลอยอยู่ในอากาศและสามารถเข้าสู่ร่างกายผ่านระบบทางเดินหายใจ โดยฝุ่นละอองขนาดเล็กจะแบ่งเป็น 2 ชนิด คือ ฝุ่นละอองขนาดเล็กที่มีขนาดเส้นผ่านศูนย์กลางน้อยกว่าหรือเท่ากับ 10 ไมครอน เรียกว่าฝุ่น PM10 และ ฝุ่นละอองขนาดเล็กที่มีขนาดเส้นผ่านศูนย์กลางน้อยกว่าหรือเท่ากับ 2.5 ไมครอน เรียกว่าฝุ่น PM2.5 ในงานวิจัยนี้จะให้ความสำคัญกับฝุ่น PM2.5 ซึ่งเป็นฝุ่นละอองขนาดเล็กที่มีความอันตรายมากเมื่อผ่านเข้าสู่ระบบทางเดินหายใจในปริมาณมากเกินมาตราฐานกำหนดที่ค่าเฉลี่ย 25 mg/m^3 ต่อ 24 ชม. แบบต่อเนื่อง จะส่งผลกระทบต่อสุขภาพและก่อให้เกิดโรค เช่น โรคหอบหืด ระบบทางเดินหายใจอักเสบ ส่งผลต่อการทำงานของปอด และโรคมะเร็งปอด เป็นต้น (Bălă, Râjnoveanu, Tudorache, Motișan, & Oancea, 2021) โดยที่ประเทศไทยมีรายงานค่าฝุ่น PM2.5 เฉลี่ยตลอดปี พ.ศ. 2564 ที่ 20.2 mg/m^3 ถูกจัดเป็นประเทศที่มีคุณภาพอากาศย่ำแย่ลำดับที่ 45 ของโลก (IQAir, 2021, p. 9)

ดังนั้น การจัดการกับปัญหาของฝุ่น PM2.5 จึงเป็นสิ่งที่จำเป็นเร่งด่วน ซึ่งวิธีการจัดการกับปัญหาฝุ่น PM2.5 มีอยู่ 2 ระดับคือ ระดับที่ 1 กำจัดต้นกำเนิดของฝุ่น PM2.5 และระดับที่ 2 คือการหลีกเลี่ยงฝุ่น PM2.5 จากวิธีการจัดการกับปัญหาฝุ่น PM2.5 ในระดับที่ 2 นั้น การแจ้งเตือนปริมาณฝุ่น PM2.5 รวมถึงการพยากรณ์ฝุ่น PM2.5 ล่วงหน้าในบริเวณต่าง ๆ ของเมืองเป็นสิ่งสำคัญเพราะจะ

ทำให้ประชาชนได้รับรู้ถึงสถานการณ์ของฝุ่น PM2.5 อย่างทันท่วงที และสามารถวางแผนการดำเนินกิจกรรมประจำวันเพื่อให้สอดคล้อง ป้องกันและหลีกเลี่ยงฝุ่น PM2.5 ได้

ประเทศไทยมีหน่วยงานที่รับผิดชอบในการดูแลและแก้ไขปัญหาทางด้านมลพิษ คือ กรมควบคุมมลพิษ สังกัดกระทรวงทรัพยากรธรรมชาติและสิ่งแวดล้อม กรมควบคุมมลพิษมีสถานีวัดภาคพื้นสำหรับวัดค่าคุณภาพอากาศกระจายอยู่ทั่วประเทศ โดยเฉพาะในเขตเมืองใหญ่และเขตเมืองอุตสาหกรรม เช่น จังหวัดกรุงเทพมหานคร จังหวัดเชียงใหม่ และจังหวัดระยอง เป็นต้น นอกจากนี้ กรมควบคุมมลพิษยังสร้างระบบสำหรับรายงานผลรวมถึงการพยากรณ์ผลของค่าปริมาณสารมลพิษต่าง ๆ ของแต่ละพื้นที่ทั่วประเทศไทย โดยใช้โปรแกรมจำลองที่ชื่อ Weather Research and Forecasting model coupled with Chemistry หรือเรียกสั้น ๆ ว่า WRF-CHEM (สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ [สวทช], 2021) เพื่อนำผลลัพธ์จากการจำลองมาใช้เป็นข้อมูลสำหรับวิเคราะห์ บริหารจัดการ และแจ้งเตือนค่ามลพิษทางอากาศล่วงหน้า ซึ่งการจำลองค่ามลพิษทางอากาศผ่านโปรแกรม WRF-CHEM จะต้องใช้ทรัพยากรคำนวณและระยะเวลาประมวลผลสูงมากเพื่อให้ผลลัพธ์

งานวิจัยนี้ ได้นำเสนอการพัฒนาแบบจำลองอนุกรมเวลาสำหรับพยากรณ์ฝุ่น PM2.5 ด้วยกระบวนการเรียนรู้ของเครื่อง โดยเลือกใช้ข้อมูลบันทึกของค่าฝุ่น PM2.5 ของพื้นที่เขตเมืองอุตสาหกรรมมาใช้เป็นชุดข้อมูลสำหรับการทดลองวิจัย การวิจัยศึกษาและพัฒนาแบบจำลองอนุกรมเวลาสำหรับพยากรณ์ค่ามลพิษทางอากาศหรือคุณภาพอากาศด้วยกระบวนการเรียนรู้ของเครื่องมีอย่างแพร่หลายในช่วงตลอด 10 ปีที่ผ่านมา ยกตัวอย่างเช่น งานวิจัยของ Du, Li, Yang, and Horng (2021) นำเสนอแบบจำลอง Deep Air Quality Forecasting Framework: DAQFF ที่เป็นการทำงานร่วมกันระหว่าง One Dimensional Convolution Neural Network: 1D-CNN และ Bi-directional Long Short-term Memory Network: Bi-LSTM พยากรณ์ค่าฝุ่น PM2.5 ของเมืองปักกิ่ง ประเทศจีน ผลของงานวิจัยแบบจำลอง DAQFF ให้ผลพยากรณ์มีความแม่นยำสูงที่ค่าความคลาดเคลื่อน RMSE เท่ากับ 8.20 สำหรับการพยากรณ์แบบหนึ่งช่วงเวลาล่วงหน้า งานวิจัยของ Chen and Chiu (2021) พัฒนาแบบจำลองเชิงสถิติและแบบจำลองการเรียนรู้สำหรับพยากรณ์ค่าคุณภาพอากาศรายวันของประเทศไต้หวัน โดยใช้ข้อมูลจาก 16 แห่ง ผลการวิจัยพบว่าแบบจำลอง Autoregressive model with Exogenous variables and Generalized Autoregressive Conditional Heteroskedasticity errors: ARX-GARCH ให้ผลพยากรณ์ดีที่สุดที่ความแม่นยำ 86.38% งานวิจัยของ Espinosa, Palma, Jiménez, Kamińska, Sciavicco, and Lucena-Sánchez (2021) เปรียบเทียบและประเมินประสิทธิภาพแบบจำลอง 1D-CNN, Recurrent Neural Network: RNN, Random Forest: RF, Lasso Regression: Lasso และ Support Vector Machines: SVM โดยการสร้างแบบจำลองและพยากรณ์ค่า NO_2 หนึ่งวันล่วงหน้า โดยใช้ข้อมูลจากสถานีวัดคุณภาพ

อากาศภาคพื้นของเมือง Wrocław ประเทศโปแลน จากผลการวิจัยแบบจำลอง LSTM ในกลุ่มอัลกอริทึม RNN ให้ประสิทธิภาพต่อผลการพยากรณ์โดยรวมดีที่สุด ส่วนงานวิจัยของ Zhang, Li, Lam, and Han (2020) นำเสนอแบบจำลองเรียนรู้เชิงลึกที่ทำงานร่วมกันระหว่างอัลกอริทึม CNN กับ LSTM เรียกว่า Deep-AIR สำหรับพยากรณ์คุณภาพอากาศโดยพยากรณ์สารมลพิษทั้ง 5 ชนิด ผลการวิจัยแสดงให้เห็นว่าแบบจำลอง Deep-AIR ให้ประสิทธิภาพดีกว่าแบบจำลองที่เป็น Baseline โดยให้ค่าความคลาดเคลื่อน Mean Average Percentage Error (MAPE) เท่ากับ 15.7 และ 27.1 สำหรับชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบตามลำดับ

สำหรับงานวิจัยที่พัฒนาแบบจำลองสำหรับพยากรณ์ฝุ่น PM หรือพยากรณ์คุณภาพอากาศในประเทศไทยในช่วงหลายปีที่ผ่านมามีงานวิจัยที่น่าสนใจ เช่น งานวิจัยของ Kanabkaew (2013) พัฒนาแบบจำลอง Simple Linear Regress: LR และ Multiple Linear Regress: MLRโดยใช้ข้อมูลจากดาวเทียม Aerosol Optical Depth: AOD ซึ่งพิจารณาเฉพาะข้อมูลของจังหวัดเชียงใหม่เป็นชุดข้อมูลสำหรับงานวิจัย โดยแบบจำลอง MLR ให้ประสิทธิภาพดีที่สุดผ่านมาตรวัด R^2 ที่ 0.77 และ 0.71 ของชุดข้อมูลฝุ่น PM2.5 และ PM10 ตามลำดับ งานวิจัยของ Tongprasert and Ongsomwang (2022) พัฒนาแบบจำลอง Geographically Weighted Regression: GWR และ Mixed-effect Model: MEM สำหรับพยากรณ์ฝุ่น PM ในเขตพื้นที่กรุงเทพมหานคร และเขตพื้นที่นอกเมืองใหญ่แถวภาคกลางตอนบน โดยใช้ชุดข้อมูลจากดาวเทียม AOD ผลการวิจัยพบว่าแบบจำลอง GWR มีความเหมาะสมที่สุดต่อการนำไปใช้พยากรณ์ฝุ่น PM ทั้งในเขตเมืองและนอกเมือง สำหรับงานวิจัยของ Srikamdee and Onpans (2019) พัฒนาและศึกษาเปรียบเทียบประสิทธิภาพแบบจำลอง Artificial Neural Network: ANN, LR และ Genetic Programming สำหรับพยากรณ์คุณภาพอากาศในบริเวณภาคเหนือของประเทศไทยในช่วงหน้าแล้ง จากงานวิจัยรายงานว่าแบบจำลอง LR และ Genetic Programming ให้การพยากรณ์แบบสองสถานะ คือ คุณภาพอากาศดีและคุณภาพอากาศไม่ดี มีความแม่นยำที่ 97.65% และ 78.70% ตามลำดับ ตัวอย่างสุดท้ายเป็นงานวิจัยของ Wongsathan (2018) เป็นการพัฒนาแบบจำลองโครงข่ายเชิง Fuzzy ที่เรียกว่า Adaptive Neuro-Fuzzy Inference System: ANFIS สำหรับพยากรณ์ฝุ่น PM10 ในจังหวัดเชียงใหม่และพื้นที่ใกล้เคียง โดยใช้ข้อมูลทางอุตุนิยมวิทยาและ Hotspots สำหรับพัฒนาแบบจำลอง ผลการวิจัยพบว่าแบบจำลอง ANFIS ให้ประสิทธิภาพการพยากรณ์ฝุ่น PM10 ดีว่าแบบจำลองอื่น ๆ ที่ได้พัฒนาก่อนหน้า

งานวิจัยนี้นำเสนอเทคนิคแบบใหม่โดยใช้แบบจำลองอนุกรมเวลาผสมระหว่างแบบจำลองอนุกรมเวลาเชิงเส้นและไม่เชิงเส้นสำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ในพื้นที่อุตสาหกรรมจังหวัดระยอง โดยพัฒนาผ่านกระบวนการเรียนรู้ของเครื่อง ในกระบวนการเตรียมข้อมูล งานวิจัยนี้เลือกใช้เทคนิคการเติมข้อมูลของชุดข้อมูลอนุกรมเวลาด้วยวิธี Temporal and Spatial Average Value

เพื่อให้เกิดความสมบูรณ์สูงสุดของชุดข้อมูลอนุกรมเวลาสำหรับนำมาใช้ในการวิจัย การประเมินและเปรียบเทียบประสิทธิภาพแบบจำลองอนุกรมเวลาที่นำเสนอ งานวิจัยนี้ได้พัฒนาแบบจำลองอนุกรมเวลาเดียวที่นิยมและมีประสิทธิภาพที่ดีกับชุดข้อมูลอนุกรมเวลา ได้แก่ แบบจำลองอนุกรมเวลาเชิงเส้น Autoregressive Integrated Moving Average: ARIMA และแบบจำลองอนุกรมเวลาไม่เชิงเส้นในกลุ่มของแบบจำลองโครงข่าย ได้แก่ ANN, LSTM และ ANFIS

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อศึกษาและพัฒนาอัลกอริทึมสำหรับสร้างแบบจำลองอนุกรมเวลาสำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ที่มีความแม่นยำสูงโดยพัฒนาแบบจำลองอนุกรมเวลาผ่านกระบวนการเรียนรู้ของเครื่อง

1.2.2 พัฒนาแบบจำลองอนุกรมเวลาที่สามารถหาความสัมพันธ์แบบเชิงเส้นและไม่เชิงเส้นภายในสายข้อมูลอนุกรมเวลา โดยใช้อัลกอริทึมผสมสำหรับพัฒนาแบบจำลองอนุกรมเวลาระหว่างแบบจำลองอนุกรมเวลาเชิงเส้นและไม่เชิงเส้น

1.2.3 ปรับปรุงประสิทธิภาพแบบจำลองอนุกรมเวลาผสมที่นำเสนอ ผ่านกระบวนการ Hyperparameter Tuning ด้วยเทคนิคการทำ Optimizations เพื่อให้ได้แบบจำลองอนุกรมเวลาที่มีประสิทธิภาพสูงสำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า

1.3 ขอบเขตของการวิจัย

1.3.1 งานวิจัยนี้ใช้ข้อมูลบันทึกเฉลี่ยรายวัน (24 ชม.) ของค่าฝุ่น PM2.5 ที่บันทึกตั้งแต่ปี ค.ศ. 2011 ถึง 2022 ของแต่ละสถานีวัดคุณภาพอากาศภาคพื้นทั่วประเทศไทย จากเว็บไซต์ของกรมควบคุมมลพิษ กระทรวงทรัพยากรธรรมชาติและสิ่งแวดล้อม สามารถดาวน์โหลดข้อมูลที่ URL http://air4thai.pcd.go.th/webV3/#/History

1.3.2 งานวิจัยนี้ได้เลือกใช้ชุดข้อมูลฝุ่น PM2.5 จากสถานีวัดคุณภาพอากาศภาคพื้น ของพื้นที่จังหวัดระยอง โดยพิจารณาเฉพาะสถานีวัดฯ รหัส 30T ในช่วงปี ค.ศ. 2017 ถึง 2021 มาทำการวิจัยเท่านั้น

1.3.3 งานวิจัยนี้พัฒนาอัลกอริทึมผสมระหว่าง อัลกอริทึมเชิงเส้นกับอัลกอริทึมไม่เชิงเส้นสำหรับหาความสัมพันธ์ภายในของสายข้อมูลอนุกรมเวลา

1.3.4 เครื่องมือที่ใช้ทำการวิจัยเพื่อพัฒนาอัลกอริทึม และการสร้างแบบจำลองอนุกรมเวลาที่นำเสนอ ผ่านกระบวนการเรียนรู้ของเครื่อง ได้แก่

(1) โปรแกรมทางภาษาที่ใช้พัฒนา: Python version 3.10.5 และ Matlab version R2022a

(2) เครื่องมือร่วมพัฒนา: Kernel-based Virtual Machine, Jupyter Notebook, และ Visual Studio Code

(3) เครื่องคอมพิวเตอร์แม่ข่าย: CentOS7 x86_64, 32 CPU-Cores, 64 GByte of RAM without GPU support

1.4 ประโยชน์ที่คาดว่าจะได้รับ

งานวิจัยนี้ได้นำเสนอการพัฒนาแบบจำลองอนุกรมเวลาสำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้าผ่านกระบวนการเรียนรู้ของเครื่อง โดยใช้ทรัพยากรคำนวณและระยะเวลาในการสร้างแบบจำลองน้อยกว่าวิธีการพัฒนาแบบจำลองผ่านทางโปรแกรมจำลองสภาพอากาศ WRF-CHEM อย่างมาก และประสิทธิภาพของแบบจำลองอนุกรมเวลาผสมที่ได้นำเสนอมีประสิทธิภาพที่ดีต่อการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ผู้วิจัยคาดหวังว่าวิธีการพยากรณ์ฝุ่น PM2.5 ล่วงหน้าที่นำเสนอจะเป็นอีกหนึ่งวิธีที่ช่วงเสริมวิธีการพยากรณ์ค่าฝุ่น PM2.5 เดิมของหน่วยงานที่รับผิดชอบให้มีประสิทธิภาพยิ่งขึ้น และคาดหวังว่าจะสามารถนำไปประยุกต์ใช้สำหรับการพยากรณ์ฝุ่น PM2.5 หรือสารมลพิษอื่น ๆ ในพื้นที่สนใจ ด้วยอุปกรณ์ทางโครงข่าย Internet of Things: IoT

นอกจากนี้แบบจำลองอนุกรมเวลาผสมที่ได้นำเสนอ สามารถหาความสัมพันธ์เชิงเส้นและไม่เชิงเส้นของสายข้อมูลอนุกรมเวลาได้เป็นอย่างดี จึงทำให้สามารถนำไปประยุกต์ใช้กับชุดข้อมูลอนุกรมเวลาในด้านอื่น ๆ ได้ด้วยเช่นกัน

บทที่ 2
ปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

มลพิษทางอากาศเป็นปัญหาทางด้านสิ่งแวดล้อมปัญหาหนึ่งที่ทั่วโลกให้ความสำคัญ เพราะมลพิษทางอากาศมีผลกระทบโดยตรงต่อสุขภาพของประชาชน และยังส่งผลต่อการพัฒนาและขยายเขตเมืองที่มีผลต่อเศรษฐกิจของประเทศ จากการรายงานข้อมูลในปี ค.ศ. 2016 ขององค์กรอนามัยโลกพบว่ามีประชาชนเสียชีวิตก่อนวัยอันควรจากผลกระทบของมลพิษทางอากาศจำนวน 4.2 ล้านคนทั่วโลก (World Health Organization [WHO], 2016) และในประเทศไทยมีการคาดการณ์ว่าจะมีผู้เสียชีวิตก่อนวัยอันควรจากมลพิษทางอากาศต่อปีประมาณ 50,000 คน ซึ่งเป็นรายงานของกรมควบคุมมลพิษ กระทวงทรัพยากรธรรมชาติและสิ่งแวดล้อมประเทศไทย ในปี ค.ศ. 2020 (กรมควบคุมมลพิษ [คพ], 2020) สาเหตุการเสียชีวิตก่อนวัยอันควรที่เป็นผลกระทบจากมลพิษทางอากาศ ซึ่งองค์กรอนามัยโลกได้แจกแจงดังนี้ จำนวน 43% เกิดจากโรคปอดอุดกั้นเรื้อรัง (Chronic Obstructive Pulmonary Disease) 29% เกิดจากโรคมะเร็งปอด (Lung Cancer) 25% เกิดจากโรคหัวใจล้มเหลว (Heart Disease) 24% เกิดจากโรคหลอดเลือดสมอง (Stroke) และ 17% เกิดจากการติดเชื้อทางเดินหายใจส่วนล่างเฉียบพลัน (Disease From Acute Lower Respiratory Infection) สารมลพิษทางอากาศ (Air Pollutants) ที่องค์กรอนามัยโลกได้ให้ความสำคัญมีจำนวน 5 ชนิด ได้แก่ 1) ฝุ่นละอองขนาดเล็ก (Particulate Matter: PM) 2) ก๊าซโอโซน (O_3) 3) ก๊าซคาร์บอนมอนอกไซด์ (CO) 4) ก๊าซไนโตรเจนไดออกไซด์ (NO_2) และ 5) ก๊าซซัลเฟอร์ไดออกไซด์ (SO_2) โดยงานวิจัยนี้มุ่งสนใจที่สารมลพิษทางอากาศฝุ่นละอองขนาดเล็ก ซึ่งแบ่งเป็น 2 ประเภท คือ ประเภทที่หนึ่งฝุ่นละอองขนาดไม่เกิน 2.5 ไมครอน (PM2.5) เป็นฝุ่นที่มีเส้นผ่านศูนย์กลางไม่เกิน 2.5 ไมครอน เกิดจากควันไอเสียทั้งของยานพาหนะบนท้องถนน การเผาไร่นาทางการเกษตร ไฟป่า และเกิดจากการดำเนินงานของโรงงานอุตสาหกรรม ซึ่งฝุ่น PM2.5 สามารถเข้าไปถึงถุงลมในปอดได้ เป็นผลให้เกิดโรคในระบบทางเดินหายใจ และโรคปอดต่าง ๆ ส่งผลให้ปอดเสื่อมประสิทธิภาพลง และประเภทที่สองฝุ่นละอองขนาดไม่เกิน 10 ไมครอน (PM10) เป็นฝุ่นที่มีขนาดเส้นผ่านศูนย์กลางไม่เกิน 10 ไมครอน เกิดจากการเผาไหม้ต่าง ๆ ในที่โล่ง กระบวนการอุตสาหกรรม ฝุ่นจากการก่อสร้าง ซึ่งส่งผลกระทบต่อสุขภาพไม่ต่างจากฝุ่น PM2.5 เนื่องจากเมื่อหายใจเข้าไปสามารถสะสมในระบบทางเดินหายใจก่อให้เกิดโรคร้ายต่าง ๆ ได้เช่นกัน

สารมลพิษทางอากาศทั้ง 5 ชนิดที่กล่าวมา มีความสัมพันธ์ต่อคุณภาพอากาศ และดัชนีคุณภาพอากาศของประเทศไทย ซึ่งระบุโดยกรมควบคุมมลพิษ แบ่งเป็น 5 ระดับ คือ ค่าคุณภาพ-

ตารางที่ 2.1 เกณฑ์ดัชนีคุณภาพอากาศของประเทศไทย[1]

AQI	ความหมาย	สี	คำอธิบาย
0-25	อากาศดีมาก	ฟ้า	คุณภาพอากาศดีมาก เหมาะสำหรับกิจกรรมกลางแจ้งและการท่องเที่ยว
26-50	อากาศดี	เขียว	คุณภาพอากาศดี สามารถทำกิจกรรมกลางแจ้งและการท่องเที่ยวได้ตามปกติ
51-100	อากาศปานกลาง	เหลือง	ประชาชนทั่วไป: สามารถทำกิจกรรมกลางแจ้งได้ตามปกติ, ผู้ที่ต้องดูแลสุขภาพเป็นพิเศษ: หากมีอาการเบื้องต้น เช่น ไอ หายใจลำบาก ระคายเคืองตา ควรลดระยะเวลาการทำกิจกรรมกลางแจ้ง
101-200	เริ่มมีผลกระทบต่อสุขภาพ	ส้ม	ประชาชนทั่วไป: ควรเฝ้าระวังสุขภาพ ถ้ามีอาการเบื้องต้น เช่น ไอ หายใจลำบาก ระคายเคืองตา ควรลดระยะเวลาการทำกิจกรรมกลางแจ้ง หรือใช้อุปกรณ์ป้องกันตนเองหากมีความจำเป็น, ผู้ที่ต้องดูแลสุขภาพเป็นพิเศษ: ควรลดระยะเวลาการทำกิจกรรมกลางแจ้งหรือใช้อุปกรณ์ป้องกันตนเองหากมีความจำเป็น ถ้ามีอาการทางสุขภาพ เช่น ไอ หายใจลำบาก ตาอักเสบ แน่นหน้าอก ปวดศีรษะ หัวใจเต้นไม่เป็นปกติ คลื่นไส้อ่อนเพลีย ควรปรึกษาแพทย์
>201	มีผลกระทบต่อสุขภาพ	แดง	ทุกคนควรหลีกเลี่ยงกิจกรรมกลางแจ้งหลีกเลี่ยงพื้นที่ที่มีมลพิษทางอากาศสูง หรือใช้อุปกรณ์ป้องกันตนเองหากมีความจำเป็น หากมีอาการทางสุขภาพควรปรึกษาแพทย์

[1]หมายเหตุ ข้อมูลจากกรมควบคุมมลพิษ (http://air4thai.pcd.go.th/webV3/#/AQIInfo)

อากาศอยู่ที่ช่วง 0 - 25 เป็นอาการดีมาก ช่วง 26 - 50 เป็นคุณภาพอากาศดี ช่วง 51 - 100 เป็นคุณภาพอากาศปานกลาง ช่วง 101 - 200 เป็นคุณภาพอากาศแย่เริ่มมีผลกระทบต่อสุขภาพ และค่าดัชนีคุณภาพอากาศเกิน 201 ขึ้นไป จะแสดงถึงคุณภาพอากาศเลวร้ายไม่ควรอยู่ในพื้นที่ดังกล่าว เพราะมีผลต่อสุขภาพโดยตรง (ดังตารางที่ 2.1) และสมการที่ (2-1) แสดงสูตรคำนวณดัชนีคุณภาพอากาศจากสารมลพิษทางอากาศที่ต้องการพิจารณา

$$AQI = \frac{i_{max} - i_{min}}{x_{max} - x_{min}}(x - x_{min}) + i_{min} \quad (2\text{-}1)$$

โดยที่ AQI คือ ค่าดัชนีคุณภาพอากาศของสารมลพิษที่พิจารณา, x เป็นความเข้มข้นของสารมลพิษที่พิจารณาจากการตรวจวัด ส่วนค่า x_{min} และ x_{max} เป็นค่าต่ำสุดและค่าสูงสุดของช่วงความเข้มข้นสารมลพิษที่พิจารณา ตามลำดับ สุดท้าย i_{min} และ i_{max} แทนค่าต่ำสุดและสูงสุดของช่วงดัชนีคุณภาพอากาศที่ตรงกับช่วงความเข้มข้นของสารมลพิษที่พิจารณา

จากค่าดัชนีของสารมลพิษทางอากาศที่คำนวณได้ สารมลพิษทางอากาศประเภทใดมีค่าดัชนีสูงสุด จะใช้เป็นดัชนีคุณภาพอากาศ ณ ช่วงเวลานั้น ความเข้มข้นของสารมลพิษทางอากาศเทียบเท่ากับค่าดัชนีคุณภาพอากาศของสารมลพิษทางอากาศทั้ง 5 ชนิด แสดงในตารางที่ 2.2

ตารางที่ 2.2 ค่าความเข้มข้นของสารมลพิษทางอากาศที่เทียบเท่ากับค่าดัชนีคุณภาพอากาศ[1]

AQI	สี	$PM_{2.5}$	PM_{10}	O_3	CO	NO_2	SO_2
		เฉลี่ย 24 ชม. ต่อเนื่อง		**เฉลี่ย 8 ชม. ต่อเนื่อง**		**เฉลี่ย 1 ชม. ต่อเนื่อง**	
0-25	ฟ้า	0-25	0-50	0-35	0-4.4	0-60	0-100
26-50	เขียว	26-37	51-80	36-50	4.5-6.4	61-106	101-200
51-100	เหลือง	38-50	81-120	51-70	6.5-9.0	107-170	201-300
101-200	ส้ม	51-90	121-180	71-120	9.1-30.0	171-340	301-400
200 ขึ้นไป	แดง	91 ขึ้นไป	181 ขึ้นไป	121 ขึ้นไป	30.1 ขึ้นไป	341 ขึ้นไป	401 ขึ้นไป

[1]หมายเหตุ ข้อมูลจากกรมควบคุมมลพิษ (http://air4thai.pcd.go.th/webV3/#/AQIInfo)

วิธีการจัดการกับปัญหามลพิษทางอากาศแบ่งเป็น 2 ส่วนที่จะต้องใช้ร่วมกัน คือ ส่วนแรกเป็นการกำจัดมลพิษทางอากาศ และส่วนที่สองเป็นการหลีกเลี่ยงมลพิษทางอากาศ โดยที่วิธีกำจัด คือ การจัดการกับต้นกำเนิดของสารมลพิษทางอากาศ เช่น การแก้ไขปัญหารถติดในเมืองใหญ่เพื่อช่วยลดก๊าซคาร์บอนมอนน็อกไซด์และฝุ่น PM การควบคุมมาตราฐานการผลิตในโรงงานอุตสาหกรรมเพื่อลดก๊าซโอโซนภาคพื้น หรือการลดการเผาในที่แจ้งทางการเกษตรเพื่อลดแหล่งกำเนิดฝุ่น PM เป็นต้น สำหรับวิธีการหลีกเลี่ยง คือ การหลีกเลี่ยงเข้าไปในพื้นที่ที่มีมลพิษทางอากาศ หรือหากจำเป็นก็สามารถใช้เครื่องมือหรืออุปกรณ์ป้องกันสารมลพิษทางอากาศได้อย่างทันท่วงที เช่น การใส่หน้ากากอนามัยเมื่ออยู่ในพื้นที่ที่มีฝุ่น PM เกินค่ามาตราฐานกำหนด เป็นต้น

งานวิจัยนี้ ได้ให้ความสำคัญกับสารมลพิษทางอากาศประเภทฝุ่น PM โดยเฉพาะฝุ่น PM2.5 ซึ่งเป็นปัญหาที่ปัจจุบันประเทศไทยเร่งดำเนินการแก้ไขอย่างเร่งด่วน ดังนั้น ความสามารถในการวิเคราะห์ข้อมูลแบบทันต่อเวลาและความสามารถในการพยากรณ์ฝุ่น PM2.5 ด้วยความแม่นยำเป็นสิ่งที่จำเป็นอย่างยิ่งต่อการรับมือกับปัญหาฝุ่น PM2.5 ซึ่งงานวิจัยนี้ได้เสนอเครื่องมือสำหรับพยากรณ์ฝุ่น

PM2.5 ด้วยแบบจำลองอนุกรมเวลาที่ถูกพัฒนาผ่านกระบวนการเรียนรู้ของเครื่องด้วยอัลกอริทึมผสมเชิงเส้นและไม่เชิงเส้น ที่มีประสิทธิภาพต่อการหาความสัมพันธ์แบบเชิงเส้นและไม่เชิงเส้นในสายข้อมูลอนุกรมเวลา ซึ่งทำให้แบบจำลองอนุกรมเวลาผสมมีประสิทธิภาพที่ดีต่อการพยากรณ์ฝุ่น PM2.5

2.1 ข้อมูลอนุกรมเวลา

ข้อมูลอนุกรมเวลา เป็นข้อมูลที่มีลำดับและความสัมพันธ์ตามช่วงเวลา เช่น ชั่วโมง วัน เดือน หรือปี เป็นต้น ซึ่งจะเห็นได้ว่าข้อมูลอนุกรมเวลาเป็นข้อมูลที่มันจะพบเจอโดยทั่วไปในชีวิตประจำวัน เช่น ข้อมูลตลาดหลักทรัพย์ ข้อมูลยอดขายในแต่ละไตรมาส หรือ ข้อมูลสภาพอากาศรายวัน เป็นต้น องค์ประกอบของข้อมูลอนุกรมเวลาสามารถแบ่งออกได้ 4 องค์ประกอบ ดังนี้

1) แนวโน้ม (Trend Component) เป็นรูปแบบระยะยาวของอนุกรมเวลา แนวโน้มสามารถแสดงรูปแบบระยะยาวที่เพิ่มขึ้น (เป็นบวก) หรือรูปแบบระยะยาวที่ลดลง (เป็นลบ) และมีแนวโน้มระยะยาวเป็นค่าคงทีเมื่ออนุกรมเวลาเป็นข้อมูลในรูปแบบคงที่ (Stationary)

2) วัฏจักรหรือวงรอบ (Cyclical Component) เป็นรูปแบบที่ข้อมูลแสดงการเคลื่อนไหวขึ้นและลงเป็นรอบ ๆ ตลอดข้อมูลอนุกรมเวลา ระยะเวลาของวัฏจักรส่วนใหญ่จะมีค่ามากกว่า 1 ปี และจะขึ้นอยู่กับประเภทของธุรกิจหรืออุตสาหกรรมที่กำลังวิเคราะห์

3) ฤดูกาล (Seasonal Component) เป็นรูปแบบการเปลี่ยนแปลงของข้อมูลที่เกิดคงที่ตามช่วงฤดูกาลและสามารถพยากรณ์ได้ ตัวอย่างเช่น ยอดขายไอศกรีมเพิ่มขึ้นในฤดูร้อนและลดลงในช่วงฤดูหนาว หรือฝุ่นละอองขนาดเล็ก PM จะมีความหน้าแน่นมากในช่วงหน้าแล้งและลดลงในช่วงหน้าฝน เป็นต้น

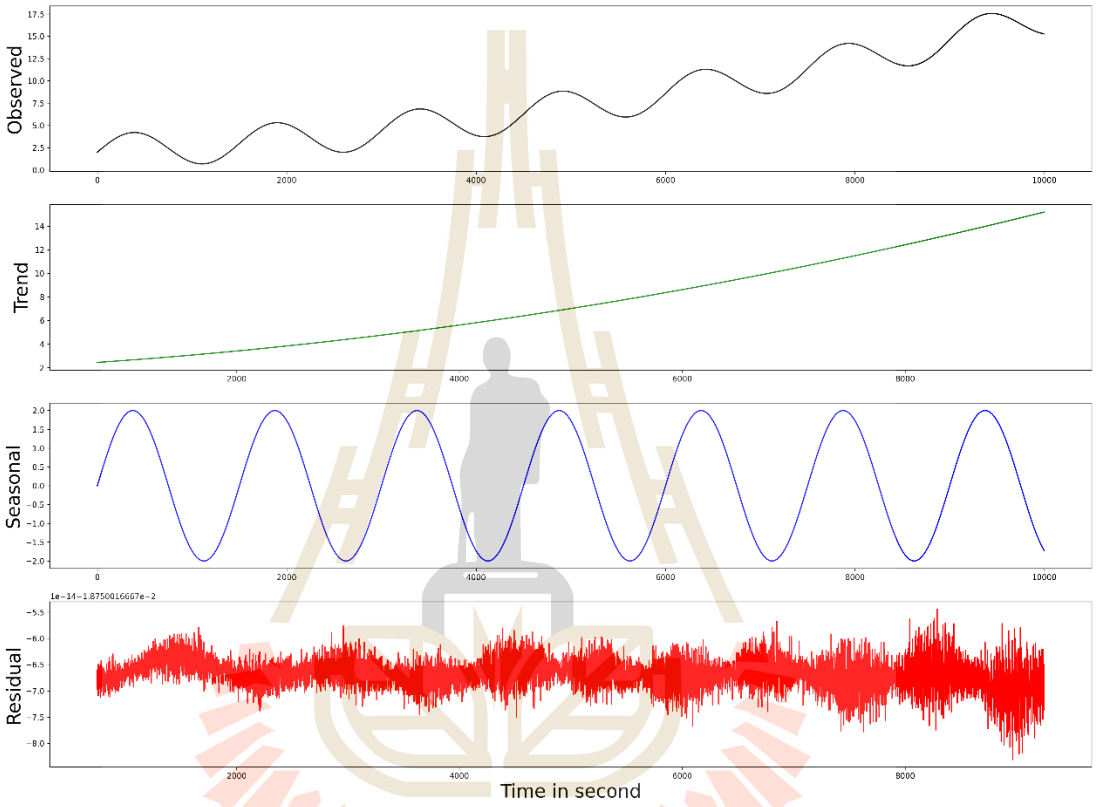
4) ความไม่ปกติ (Irregular Component) องค์ประกอบนี้คาดเดาไม่ได้ ทุกอนุกรมเวลามีองค์ประกอบที่คาดเดาไม่ได้ ซึ่งค่าของข้อมูลที่มีความไม่ปกติจะเป็นลักษณะแบบสุ่ม โดยส่วนมากความไม่ปกติจะเกิดจากเหตุการณ์ที่ไม่คาดคิด เช่น แผ่นดินไหว สงคราม หรือเหตุการณ์ทางการเมือง เป็นต้น อาจจะเรียกความไม่ปกติในข้อมูลอนุกรมเวลาว่าสัญญาณรบกวนขาว (White Noise) จากรูปที่ 2.1 แสดงลักษณะกราฟองค์ประกอบทั้ง 4 ของข้อมูลอนุกรมเวลาที่สร้างขึ้นจากสมการ Sine Wave โดยแสดงรูปกราฟแนวโน้ม กราฟฤดูกาล และกราฟความไม่ปกติ ตามลำดับ

ข้อมูลอนุกรมเวลาอาจจะประกอบด้วยองค์ประกอบใดองค์ประกอบหนึ่ง หรือทุกองค์ประกอบรวมกันก็ได้ โดยการรวมกันขององค์ประกอบของอนุกรมเวลา สามารถรวมกันได้ด้วยการบวกหรือการคูณ ดังสมการที่ (2-2) และ (2-3) ตามลำดับ

$$y_t = T + C + S + I \tag{2-2}$$

$$y_t = T \times C \times S \times I \quad (2\text{-}3)$$

โดยที่ y_tคือ อนุกรมเวลา ณ เวลา t, T เป็นค่าแนวโน้ม, C เป็นค่าวัฏจักรหรือวงรอบ, S เป็นค่าฤดูกาล, และ I คือ ค่าความไม่ปกติ



รูปที่ 2.1 แสดงองค์ประกอบของข้อมูลอนุกรมเวลาสมการ Sine Wave

โดยที่ y_tคือ อนุกรมเวลา ณ เวลา t, T เป็นค่าแนวโน้ม, C เป็นค่าวัฏจักรหรือวงรอบ, S เป็นค่าฤดูกาล, และ I คือ ค่าความไม่ปกติ

2.2 แบบจำลองเชิงสถิติ

แบบจำลองทางสถิติ เป็นเครื่องมือที่นิยมใช้เพื่อการทำนายหรือพยากรณ์ข้อมูลมานานหลายทศวรรษ เพราะในอดีตคอมพิวเตอร์ยังไม่มีประสิทธิภาพสูงเหมือนปัจจุบัน ทำให้การนำกระบวนการเรียนรู้ของเครื่องมาใช้งานทำได้อย่างไม่เต็มประสิทธิภาพ แบบจำลองทางสถิติที่ใช้สำหรับพยากรณ์ข้อมูลอนุกรมเวลามีด้วยกันหลายแบบจำลอง เช่น แบบจำลอง Auto Regressive: AR, แบบจำลอง Moving Average: MA, แบบจำลอง Auto Regressive Moving Average: ARMA, และแบบจำลอง

Auto Regressive Integrated Moving Average: ARIMA ซึ่งในกลุ่มของแบบจำลองเชิงสถิติ สำหรับแบบจำลองที่มีประสิทธิภาพดีที่สุดต่อการพยากรณ์ข้อมูลอนุกรมเวลาคือแบบจำลองอนุกรมเวลา ARIMA

2.2.1 Autoregressive Integrated Moving Average: ARIMA

แบบจำลอง ARIMA เป็นแบบจำลองที่รวมกันระหว่าง 3 กระบวนการ ได้แก่ กระบวนการที่ 1) Auto Regressive, AR เป็นแบบจำลองที่ใช้สมการความสัมพันธ์เชิงเส้นในการพยากรณ์ โดยใช้คุณสมบัติของข้อมูลย้อนหลังที่เรียกว่า Lag Time ซึ่งจะมีความคล้ายกันกับแบบจำลอง Multiple Regression แต่แบบจำลอง AR มีการใช้ค่า Lag Time เป็นตัวแปรทำนาย (Predictor Variable) ดังสมการ (2-4)

$$y_t = \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \alpha_3 y_{t-3} + \cdots + \alpha_p y_{t-p} + c + \varepsilon_t \qquad (2\text{-}4)$$

โดยที่ y_t คือ ข้อมูลอนุกรมเวลา ที่เวลา t, y_{t-i} ข้อมูลอนุกรมเวลาย้อนหลัง (Lag Time) ที่เวลา $t-i$ โดยที่ i เท่ากับ 1 ถึง p, α_i เป็นค่าสัมประสิทธิ์ประจำตำแหน่ง i โดยที่ i เท่ากับ 1 ถึง p, c เป็นค่าคงที่ และ ε_tคือ ตัวแปรสุ่มคลาดเคลื่อน (White Noise) ที่เวลา t

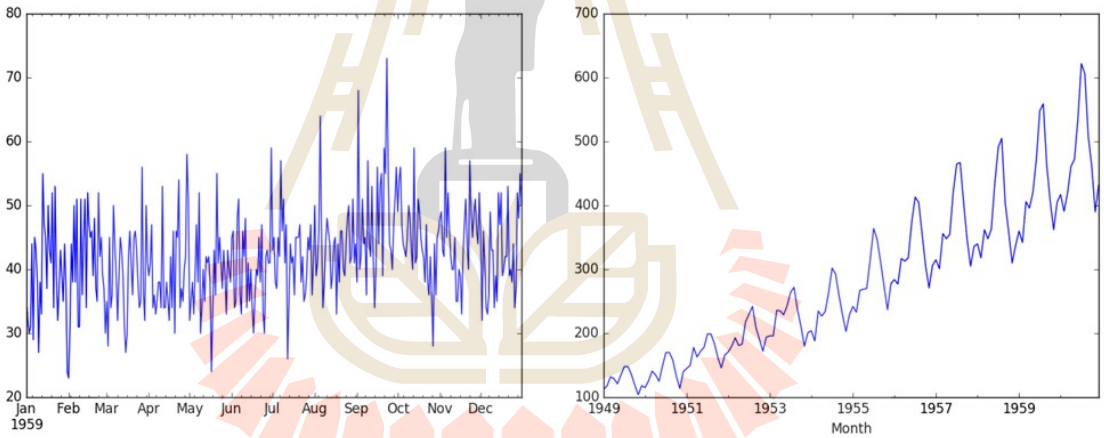
กระบวนการที่ 2) Moving Average, MA เป็นแบบจำลองสมการเชิงเส้นและพยากรณ์ค่าในอนาคตโดยใช้ข้อมูลในอดีต (ข้อมูลย้อนหลัง) เหมือนกันกับแบบจำลอง AR แต่แบบจำลอง MA พิจารณาความสัมพันธ์เชิงเส้นผ่านค่าความคาดเคลื่อนของแต่ละช่วงเวลา ดังสมการ (2-5) แตกต่างกันกับ Moving Average Smoothing ซึ่งแบบจำลอง MA นี้ใช้สำหรับพยากรณ์ค่าในอนาคตแต่ Moving Average Smoothing ใช้สำหรับการคาดคะเนแนวโน้มของข้อมูล (Trend และ Cycle) ดังนั้นเมื่อรวม AR กับ MA เข้าด้วยกันจะได้แบบจำลอง ARMA ซึ่งมีความคล้ายกันกับแบบจำลอง ARIMA แต่แบบจำลอง ARIMA มีกระบวนการในการทำ Differencing สำหรับแปลงข้อมูลอนุกรมเวลาเพิ่มเข้ามา

$$y_t = \beta_1 \varepsilon_{t-1} + \beta_2 \varepsilon_{t-2} + \beta_3 \varepsilon_{t-3} + \cdots + \beta_q \varepsilon_{t-q} + c + \varepsilon_t \qquad (2\text{-}5)$$

โดยที่ y_t คือ ข้อมูลอนุกรมเวลา ที่เวลา t, ε_{t-i} คือ ค่าความคาดเคลื่อนย้อนหลัง (Lag Time) ที่เวลา $t-i$ โดยที่ i เท่ากับ 1 ถึง q, β_i เป็นค่าสัมประสิทธิ์ประจำตำแหน่ง i โดยที่ i เท่ากับ 1 ถึง q, c เป็นค่าคงที่ และ ε_tคือ ตัวแปรสุ่มคลาดเคลื่อน (White Noise) ที่เวลา t

กระบวนการที่ 3) เรียกว่า Integrated, I เป็นกระบวนการทำ Differencing หรือเรียกสั้น ๆ ว่า Differ เพื่อช่วยจัดการในเรื่องของความเป็น Stationary ของข้อมูลอนุกรมเวลา ซึ่ง

เป็นข้อกำหนดหลักที่สำคัญของการใช้แบบจำลองเชิงสถิติอย่าง AR และ MA ซึ่งขั้นตอน Differ ในส่วนของ Integrated ของแบบจำลอง ARIMA ทำให้ ARIMA มีประสิทธิภาพดีกว่าแบบจำลอง ARMA ข้อมูลอนุกรมเวลาที่มีความเป็น Stationary หมายความว่าข้อมูลอนุกรมเวลานี้จะต้องไม่มีค่าแนวโน้ม (Trend) ไม่มีค่าฤดูกาล (Seasonality) มีระดับของกราฟค่อนข้างคงที่ (Constant Level) และมีค่าความแปรปรวนคงที่ (Constant Variance) หรืออาจจะกล่าวได้ว่าข้อมูลอนุกรมเวลาที่มีความเป็น Stationary ข้อมูลในอนาคตจะต้องมีความคล้ายกันกับข้อมูลในอดีตในเชิงของความน่าจะเป็น รูปที่ 2.2 แสดงกราฟข้อมูลอนุกรมเวลาที่เปรียบเทียบระหว่างข้อมูลอนุกรมเวลาที่มีกว่าเป็น Stationary และไม่เป็น Stationary โดยความเป็น Stationary ของข้อมูลอนุกรมเวลาสามารถตรวจสอบได้ทั้งการใช้เครื่องมือทางสถิติ เช่น Box-Jenkins Method, Augmented Dickey–Fuller (ADF) Test, Regression Method, Smoothing Technique หรือจะใช้วิธีวาดกราฟแล้วพิจารณาก็ได้เช่นเดียวกัน (Data Visualization)



(a) (b)

รูปที่ 2.2 แสดงกราฟข้อมูลอนุกรมเวลาที่เป็น Stationary (a) กับที่ไม่เป็น Stationary (b)

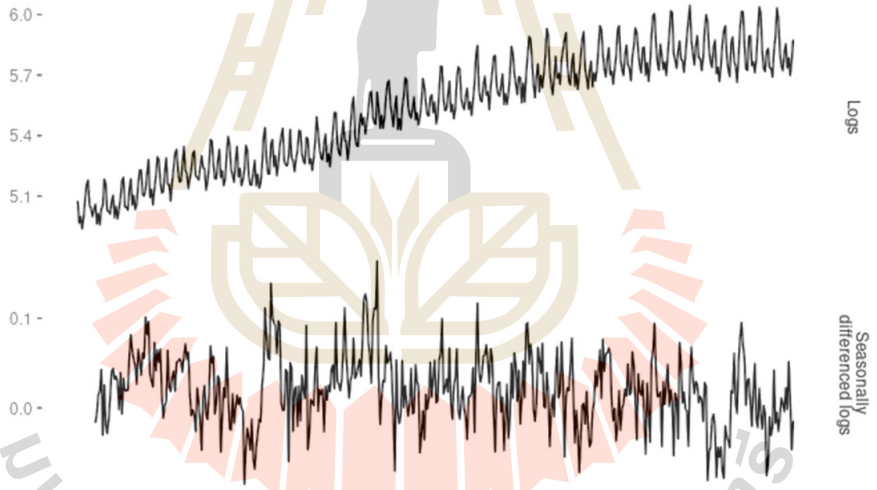
จากที่ได้กล่าวข้างต้นความเป็น Stationary ของข้อมูลอนุกรมเวลานั้นมีความสำคัญต่อการพยากรณ์ทางสถิติ ดังนั้นจะต้องทำข้อมูลอนุกรมเวลาที่ไม่เป็น Stationary ให้มีความเป็น Stationary โดยในขั้นตอน Integrated ของแบบจำลอง ARIMA จะใช้เทคนิค Differencing หรือ Differ เพื่อแปลงข้อมูลอนุกรมเวลาให้มีความเป็น Stationary โดยการขจัดค่าแนวโน้ม หรือ ค่าฤดูกาล เทคนิค Differ คือการหาค่าส่วนต่างระหว่างสองค่าของสายข้อมูล โดยที่จะใช้ค่าตาม Lag Time มาเป็นตัวลบ ดังสมการ (2-6) เช่น ถ้าต้องการที่จะขจัดค่าแนวโน้มจะต้องทำ Lag-1 Differencing และถ้าต้องการขจัดค่าฤดูกาล โดยมีทั้งหมด S ฤดู จะต้องทำ Lag-S Differencing รูปที่ 2.3 แสดงกราฟข้อมูล

อนุกรมเวลาหลังการทำ Differ เพื่อแปลงข้อมูลให้มีความเป็น Stationary โดยการทำ Lag-1 Differencing เพื่อขจัดค่าแนวโน้ม

$$y'_t = y_t - y_{t-s} \tag{2-6}$$

โดยที่ y'_t ข้อมูลอนุกรมเวลาหลังการทำ Differ ที่ตำแหน่งเวลา t, y_t ข้อมูลอนุกรมเวลาที่ตำแหน่งเวลา t, และ y_{t-s} ข้อมูลอนุกรมเวลาที่ตำแหน่งเวลา $t-s$ โดยที่ s คือ Order ของการทำ Differ ดังนั้นเมื่อรวมสามขั้นตอนของแบบจำลอง ARIMA จะได้สมการดัง (2-7)

$$\begin{aligned} y_t = {} & \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \alpha_3 y_{t-3} + \cdots + \alpha_p y_{t-p} + \beta_1 \varepsilon_{t-1} \\ & + \beta_2 \varepsilon_{t-2} + \beta_3 \varepsilon_{t-3} + \cdots + \beta_q \varepsilon_{t-q} + c + \varepsilon_t \end{aligned} \tag{2-7}$$

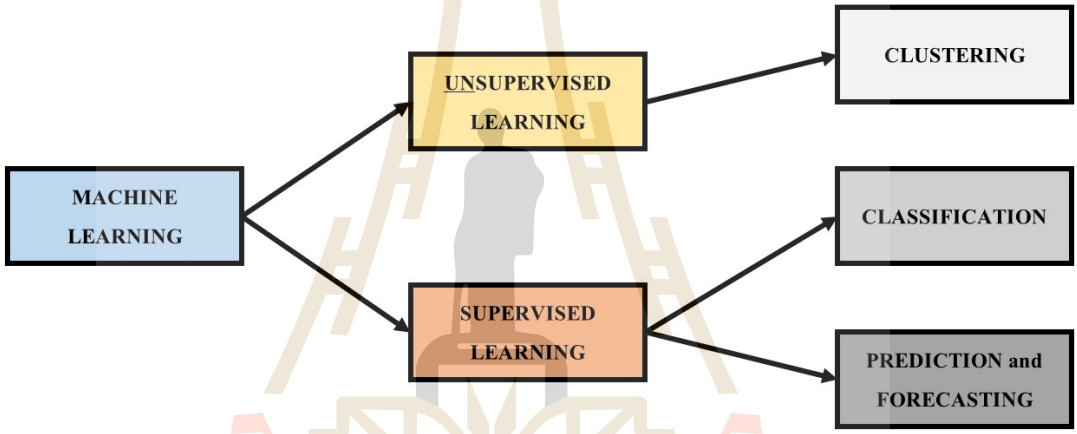


รูปที่ 2.3 แสดงกราฟก่อนและหลังการทำ Lag-1 Differencing

ตัวแปรหรือพารามิเตอร์ที่จะต้องพิจารณาและต้องถูกกำหนดเมื่อใช้แบบจำลอง ARIMA คือ พารามิเตอร์ p, d, และ q โดยที่ p จะเป็นค่าที่แสดงถึงจำนวน Lag Time ที่ใช้ในส่วนของ AR พิจารณาจาก Partial Auto Correlation Function (PACF) ส่วนตัวแปร q จะเป็นค่าที่แสดงถึงจำนวน Lag Time ของค่าความคลาดเคลื่อนที่ใช้ในส่วนของ MA ซึ่งจะพิจารณาจาก Auto Correlation Function (ACF) และตัวแปรสุดท้ายที่ต้องพิจารณาคือ d เป็นตัวแปรในส่วนของขั้นตอน Integrated ซึ่งค่าของตัวแปร d จะบ่งบอกถึงการทำ Differencing เพื่อแปลงข้อมูลอนุกรมเวลาให้มีความเป็น Stationary

2.3 การเรียนรู้ของเครื่อง

การเรียนรู้ของเครื่อง เป็นกระบวนการที่นำชุดข้อมูลที่สนใจ (Dataset) เป็นข้อมูลนำเข้า ให้ระบบคอมพิวเตอร์ใช้เป็นข้อมูลสำหรับเรียนรู้ เพื่อสร้างแบบจำลองทางคณิตศาสตร์ผ่านอัลกอริทึมที่กำหนด และใช้แบบจำลองที่ได้แก้ปัญหาในรูปแบบต่าง ๆ เช่น การจัดกลุ่ม, จำแนก, ทำนาย หรือพยากรณ์ ข้อดีของการสร้างแบบจำลองผ่านกระบวนการเรียนรู้ของเครื่องคือ แบบจำลองจะถูกสร้างโดยอัตโนมัติบนพื้นฐานของข้อมูล และแบบจำลองสามารถแก้ปัญหาได้หลากหลาย โดยไม่ยึดติดกับปัญหาใดปัญหาหนึ่ง รวมทั้งแบบจำลองมีความยืดหยุ่นสูงเพราะสามารถปรับแบบจำลองได้ด้วยความสามารถของการเรียนรู้ผ่านชุดข้อมูล



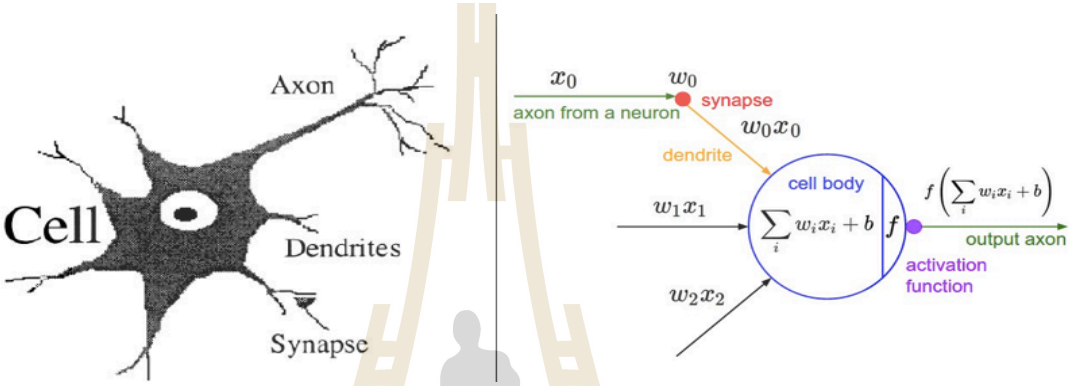
รูปที่ 2.4 แสดงประเภทของกระบวนการเรียนรู้ของเครื่อง

อัลกอริทึมสำหรับกระบวนการเรียนรู้ของเครื่อง แบ่งออกเป็น 2 กลุ่มใหญ่ ๆ ตามลักษณะของการเรียนรู้ แสดงดังรูปที่ 2.4 คือ แบบมีการแนะนำการเรียนรู้ (Supervised Learning) ซึ่งจะถูกกำหนดตัวแปรเป้าหมายไว้ (Target Variable) ขณะทำการเรียนรู้ และแบบไม่มีการแนะนำการเรียนรู้ (Unsupervised Learning) ซึ่งจะเรียนรู้ผ่านข้อมูลทั้งหมดโดยไม่มีการระบุตัวแปรเป้าหมาย โดยที่ กระบวนการเรียนรู้แบบมีการแนะนำการเรียนรู้ จะเป็นกลุ่มของแบบจำลองที่สร้างมาเพื่อใช้ในการจำแนกข้อมูล (Classification) หรือเพื่อทำนายผลข้อมูล (Prediction) ส่วนกระบวนการเรียนรู้แบบไม่มีการแนะนำการเรียนรู้ จะนำไปประยุกต์ใช้กับการจัดกลุ่มข้อมูล (Clustering) เพื่อค้นหาคุณสมบัติบ่งชี้ในชุดข้อมูลนั้น ๆ โดยงานวิจัยนี้ ให้ความสนใจที่กระบวนการเรียนรู้แบบมีการแนะนำการเรียนรู้

2.3.1 Artificial Neural Network: ANN

โครงข่ายประสาทเทียม เป็นอัลกอริทึมที่ทำงานเลียนแบบหลักการทำงานของสมองมนุษย์ ซึ่งสมองของมนุษย์มีความสามารถในการจดจำ (Recognize) เรียนรู้ (Learning) จำแนก

(Classify) หรือทำนาย (Predict) เพราะสมองของมนุษย์ทุกคนจะมีเซลล์ประสาทหลายล้านเซลล์ และทุก ๆ เซลล์ประสาทจะเชื่อมต่อกัน โดยที่ตัวเซลล์ประสาทแต่ละตัวจะมี Nucleus ค่อยควบคุมการทำงานของเซลล์ประสาท และมี Dendrites เป็นตัวรับข้อมูลเข้ามายังเซลล์ประสาท ส่วน Axon จะเป็นตัวส่งผ่านสัญญาณหรือข้อมูลไปยังเซลล์ประสาทอื่น ๆ โดยเชื่อมต่อกับ Synapse ของเซลล์ประสาทปลายทาง



(a) (b)

รูปที่ 2.5 แสดงการเปรียบเทียบ (a) เซลล์ประสาทของสมองมนุษย์ กับ (b) เซลล์ประสาทเทียม (Agatonovic-Kustrin & Beresford, 2000)

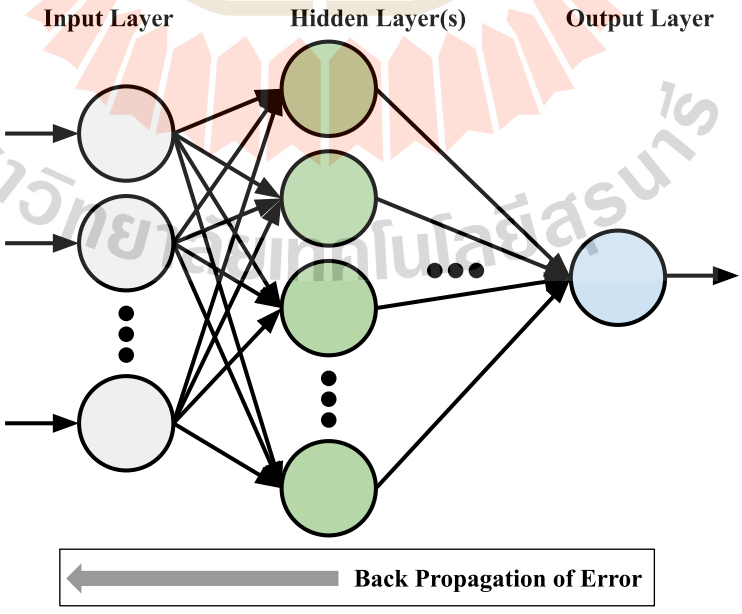
สำหรับอัลกอริทึม ANN จะเป็นโครงข่ายของกลุ่มเซลล์ประสาท (Neural) ที่มาเชื่อมต่อกัน โดย Neural แต่ละตัวจะเปรียบเสมือนเซลล์ประสาทของสมองมนุษย์ รูปที่ 2.5 แสดงการเปรียบเทียบเซลล์ประสาทของสมองมนุษย์กับ ANN โดยที่รูป 2.5 (b) แสดงหลักการทำงานของ Neural Node ของ ANN โดยมีหลักการทำงานคือ รับข้อมูลนำเข้า (Input) x_i มาคูณกับค่าน้ำหนักประจำเส้นสัญญาณ w_i ส่งค่าผลลัพธ์มายัง Neural Node เพื่อรวมค่าของข้อมูลที่รับเข้ามาทั้งหมดแล้วบวกกับค่า Bias ประจำ Neural Node จากนั้นปรับค่าที่ได้ผ่าน Activation Function หรือเรียกอีกอย่างว่า Transfer Function เพื่อปรับข้อมูลเป็นข้อมูลส่งออก (Output) ดังสมการ (2-8) สำหรับ Activation Function ที่ได้รับความนิยม ได้แก่ Sigmoid Function, Linear Function, Tanh Function หรือ ReLU Function เป็นต้น

$$y = f(\sum_i x_i w_i + b) \tag{2-8}$$

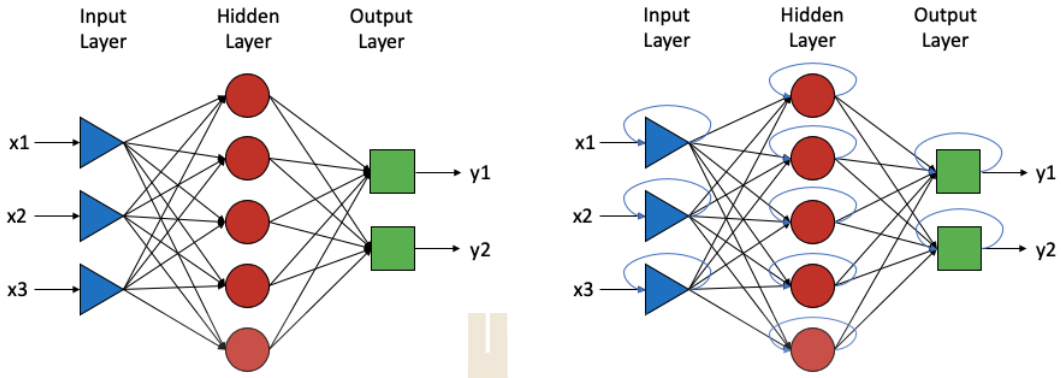
โครงสร้างของโครงข่าย ANN โดยพื้นฐานจะต้องประกอบด้วย 3 ชั้น ดังนี้ 1) ชั้นรับข้อมูล (Input Layer) ทำหน้าที่รับข้อมูลนำเข้าของแต่ละตัวแปร Input 2) ชั้นคำนวณข้อมูล (Hidden Layer) โดยเป็นชั้นที่ทำกระบวนการประมวลผล 3) ชั้นผลลัพธ์ (Output Layer) ทำหน้าที่

รวมผลลัพธ์ของชั้นที่ 2 เพื่อคำนวณเป็นผลลัพธ์สุดท้าย ซึ่งการกำหนดจำนวน Neural Node ในแต่ละชั้นนั้นขึ้นอยู่กับการใช้งาน โดยต้องคำนึงถึงความสอดคล้องกับข้อมูลหรือปัญหา เพื่อให้อัลกอริทึม ANN ทำงานได้มีประสิทธิภาพดีที่สุด แสดงดังรูปที่ 2.6

หลักการทำงานของอัลกอริทึม ANN จะเป็นการทำงานแบบส่งข้อมูลไปยัง Neural Node ในแต่ละชั้นแบบแพร่กระจาย (Propagation) ซึ่งมีกระบวนการส่งข้อมูลผ่านโครงข่าย 2 แบบ คือ แบบ Feedforward Propagation และ Feedback Propagation โดยกระบวนการ Feedforward Propagation จะเป็นการส่งและคำนวณค่าของข้อมูลไปในทิศทางข้างหน้าอย่างเดียว แต่กระบวนการ Feedback Propagation จะทำการคำนวณค่าของข้อมูลในสองทิศทางคือไปข้างหน้าและย้อนกลับ ซึ่งแต่ละ Neural Node สามารถนำค่าของข้อมูลส่งออกกลับมาร่วมคำนวณกับค่าน้ำหนักประจำเส้นสัญญาณและค่าของข้อมูลนำเข้าได้ ทำให้ Feedback Propagation มีความไวของการเรียนรู้ดีกว่า Feedforward Propagation ส่วนข้อดีของ Feedforward Propagation คือจะมีการทำงานที่เร็วกว่า Feedback Propagation ซึ่งแสดงโครงสร้างของโครงข่าย ANN รวมถึงกระบวนการทำงานแบบ Feedforward Propagation และแบบ Feedback Propagation ในรูปที่ 2.7 นอกจากโครงข่าย ANN จะส่งข้อมูลนำเข้าแบบแพร่กระจายไปข้างหน้าแล้วนั้น โครงข่าย ANN จะต้องมีการตรวจสอบความถูกต้องของการคำนวณผ่านกระบวนการ Back Propagation นำเสนอโดย Rumelhart ในปี 1985 (Rumelhart, Hinton, & Williams, 1985) ซึ่งเป็นกระบวนการที่ใช้ในการปรับค่าน้ำหนักประจำเส้นสัญญาณและค่า Bias ประจำ Neural Node โดยผ่าน Gradient Descent: GD Algorithm



รูปที่ 2.6 แสดงโครงข่ายพื้นฐานของ ANN

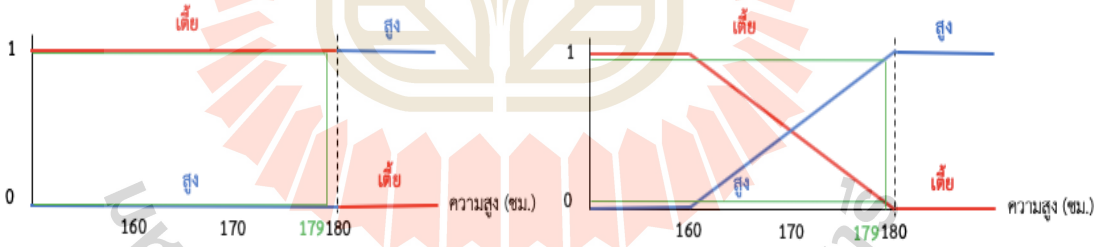


(a) (b)

รูปที่ 2.7 แสดงการทำงานแบบ (a) Feedforward Propagation และ (b) Feedback Propagation

2.3.2 Adaptive Neuro-Fuzzy Inference System: ANFIS

วัตถุประสงค์หลักของ ANFIS คือการรวมเอาข้อดีของ Fuzzy Inference System กับ Neural Network มารวมกัน คือ การนำเอาประโยชน์ของ Fuzzy Inference System ที่มีความสามารถในการสร้างเหตุผล (Rule) เพื่อใช้ในการตัดสินใจ มาผนวกกับความสามารถในการเรียนรู้ด้วยชุดข้อมูล โดยใช้อัลกอริทึม Neural Network เพื่อให้ได้ผลลัพธ์สำหรับการตัดสินใจที่ดีที่สุด หลักการทำงานของ ANFIS จะประกอบด้วย 2 ส่วนที่สำคัญ ส่วนแรกคือ Fuzzy Inference System: FIS และส่วนที่สองคือ Artificial Neural Network: ANN



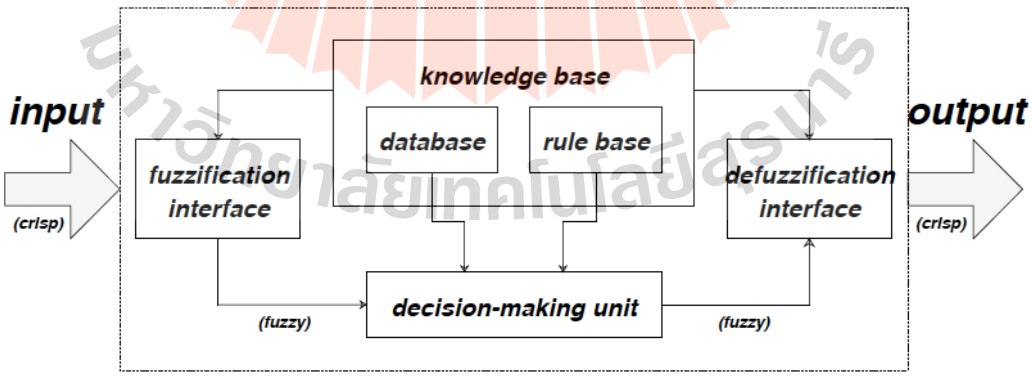
(a) (b)

รูปที่ 2.8 แสดงเซตแบบ (a) Crisp Set และ (b) Fuzzy Set สำหรับระบุความสูงของคน

Fuzzy Set เป็นเซตของข้อมูลที่ไม่มีความแน่นอน (Infinite-valued) ถูกนำเสนอในปี ค.ศ. 1965 โดยศาสตราจารย์ L.A. Zadeh (Zadeh, 1965) ซึ่งมีความสอดคล้องกับปัญหาบนพื้นฐานของความเป็นจริงมากกว่าหลักการของเซตแบบดั้งเดิมในทางคณิตศาสตร์ ที่เป็น Classical Set หรือ Crisp Set ซึ่งเป็นเซตของข้อมูลที่มีความแน่นอน ส่วนใหญ่จะเป็นข้อมูลแบบ “จริงหรือเท็จ” (Bi-valued) ถ้ากล่าวถึงปัญหาอย่างง่าย สามารถใช้ Crisp Set อธิบายได้ เช่น ม้าเป็นสัตว์บกจริงหรือไม่? โดยที่คำตอบเป็น Bi-valued คือ จริงหรือเท็จ เป็นต้น ส่วนปัญหาที่ต้องใช้ Fuzzy Set

ในการอธิบาย จะเป็นปัญหาที่มีความไม่แน่นอนของคำตอบ เช่น ความสูงของมนุษย์ ถ้ากล่าวว่า คนที่มีความสูงตั้งแต่ 180 ซม. ขึ้นไป เป็นคนสูง และคนที่มีความสูงต่ำกว่า 180 ซม. ทั้งหมดเป็นคนเตี้ย ถ้าใช้หลักการของ Crisp Set อธิบาย เซตของคนสูงคือคนที่มีความสูงตั้งแต่ 180 ซม. ขึ้นไป และเซตของคนเตี้ย คือคนที่มีความสูงต่ำกว่า 180 ซม. ทั้งหมด แต่ในความเป็นจริง อาจจะไม่สามารถกล่าวได้ว่าคนที่มีความสูง 179 ซม. เป็นคนเตี้ยเหมือนกันกับคนที่สูง 150 ซม. ในลักษณะเช่นนี้ อาจทำให้เกิดการตัดสินใจที่ผิดพลาด Fuzzy System ได้เสนอวิธีการแก้ไขปัญหาลักษณะนี้ โดยใช้ฟังก์ชันสมาชิก (Membership Function) มาช่วยอธิบายความเป็นสมาชิกของข้อมูล โดยค่าของฟังก์ชันสมาชิกจะเป็นค่าแบบต่อเนื่องตั้งแต่ 0-1 จากรูปที่ 2.8 ถ้าใช้ Crisp Set ระบุคนที่สูง 179 ซม. กับคนที่สูง 150 ซม. ถูกจัดเป็นคนเตี้ยเหมือนกัน แต่ถ้าใช้ Fuzzy Set โดยผ่านฟังก์ชันสมาชิก จะได้ว่าคนที่มีความสูง 179 ซม. จัดเป็นคนสูงเพราะมีค่าความสูงที่ 0.9 และค่าความเตี้ยอยู่ที่ 0.1 ส่วนคนที่มีความสูง 150 ซม. จะถูกจัดเป็นคนเตี้ยเพราะมีค่าความสูงที่ 0.0 และมีค่าความเตี้ยที่ 1.0

Fuzzy Inference System: FIS อาจจะเรียกกันในชื่ออื่น ๆ ดังต่อไปนี้ Fuzzy Rule-based System, Fuzzy Associative Memories หรือ Fuzzy Controller โดยที่ FIS จะใช้ If-then Rules หรือเรียกอีกอย่างว่า Fuzzy Condition Statement ในการสร้างกระบวนการตัดสินใจผ่านการประมวลผลเชิงตรรกะ (Inference) โดยกฎ (Rule) จะประกอบด้วย 2 ส่วน คือ เหตุการณ์ที่เกิดก่อน (Antecedent หรือ Premise) และ เหตุการณ์ที่จะเกิดตามมาภายหลัง (Consequent) เช่น IF x is A and y is B then z is C คือ ถ้ามีค่าตัวแปร x เป็น “A” และค่าตัวแปร y เป็น “B” จะทำให้ ค่าตัวแปร z เป็น “C” โดยที่ค่า A, B และ C เป็นค่าทางความรู้สึกหรือค่าทางภาษา (Linguistic Values)

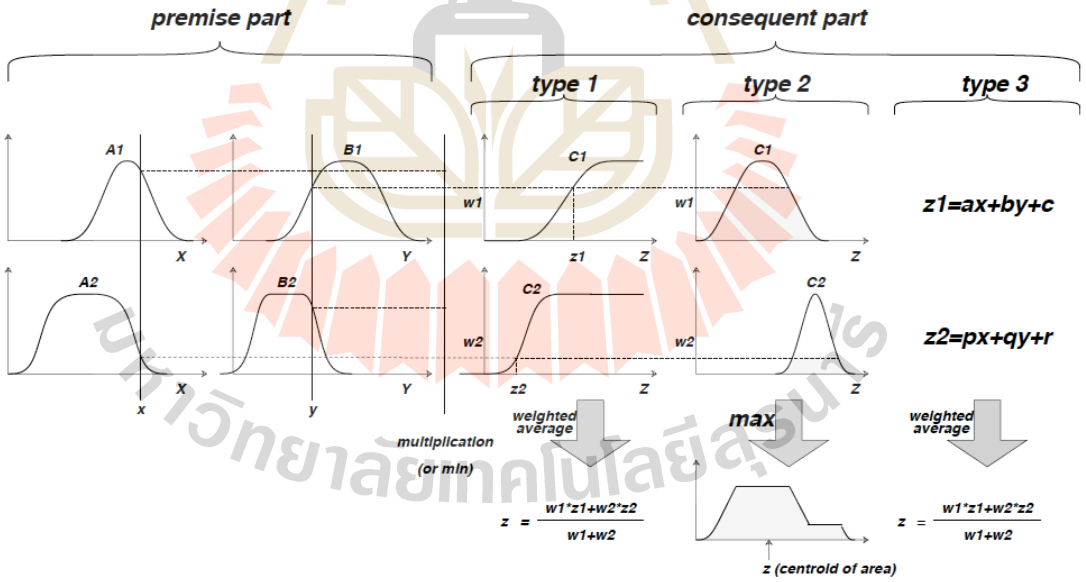


รูปที่ 2.9 แสดงโครงสร้างของ Fuzzy Inference System: FIS (Jang, 1993)

โครงสร้างของ FIS แสดงดังรูปที่ 2.9 ซึ่งจะประกอบไปด้วย 5 หน่วยที่ทำงานร่วมกันดังนี้ หน่วย Rule Base ทำหน้าที่เก็บกฎของ If-then Rules หน่วย Database เป็นฐานข้อมูลสำหรับ

ใช้ระบุฟังก์ชันสมาชิกของ Fuzzy Set ที่ใช้ใน Fuzzy Rule โดยในส่วนนี้จะใช้องค์ความรู้ของมนุษย์เป็นผู้กำหนด หน่วย Decision-making Unit เป็นหน่วยในการตัดสินใจโดยผ่านกระบวนการประมวลผลเชิงตรรกะ ตามกฎของ Fuzzy Rules หน่วย Fuzzification Interface มีหน้าที่ทำการแปลง Crisp Inputs ไปเป็นค่าทางความรู้สึกหรือค่าทางภาษา (Linguistic Values) และหน่วย Defuzzification Interface เป็นกระบวนการสุดท้ายในการแปลงค่าของ Fuzzy ไปเป็น Crisp Output ปกติแล้วในส่วนของ Rule Base กับ Database จะรวมกันเรียกว่า Knowledge Base

กระบวนการให้เหตุผลทาง Fuzzy (Fuzzy Reasoning) ของ FIS เป็นกระบวนการประมวลผลเชิงตรรกะบนพื้นฐานของ Fuzzy If-then Rules โดยจะมี 4 กระบวนการสำคัญดังต่อไปนี้ 1) กระบวนการทำการเปรียบเทียบค่าของ Input Values กับฟังก์ชันสมาชิก ในส่วนของเหตุการณ์ที่เกิดขึ้นก่อน เพื่อให้ได้ค่าความรู้สึกหรือค่าทางภาษา (Linguistic Values) 2) กระบวนการทำการรวมค่าที่ได้จากฟังก์ชันสมาชิก โดยปกติจะใช้ การคูณ (Multiplication) หรือใช้ค่าน้อยสุด (Min) เพื่อกำหนดค่าน้ำหนักสำหรับแต่ละกฎของ Fuzzy Rule 3) กระบวนการสร้างเหตุการณ์ที่เกิดตามมาภายหลังของแต่ละกฎของ Fuzzy Rules ตามค่าน้ำหนัก และ 4) กระบวนการทำการรวมค่าของเหตุการณ์ที่เกิดตามมาภายหลังเพื่อเป็น Crisp Output ขั้นตอนนี้เรียกว่า Defuzzification



รูปที่ 2.10 แสดงกระบวนการให้เหตุผลเชิง Fuzzy (Jang, 1993)

สำหรับประเภทของกระบวนการให้เหตุผลทาง Fuzzy มีอยู่ด้วยกัน 3 ประเภท (รูปที่ 2.10) ได้แก่ Type 1 คือ ค่าของการให้เหตุผล (Fuzzy Output) เป็นค่าเฉลี่ยของน้ำหนักในแต่ละกฎของ Crisp Output โดยขึ้นอยู่กับค่าความเข้มของน้ำหนักและค่าผลลัพธ์จากฟังก์ชันสมาชิก และ

ฟังก์ชันสมาชิกจะเป็นแบบ Monotonically Non-decreasing เท่านั้น Type 2 คือ ค่าของการให้เหตุผล (Fuzzy Output) จะใช้ค่ากลางของพื้นที่ (Center of Area), Bisector of Area, Mean of Maxima หรือ Maximum ซึ่งพื้นที่เกิดจากผลรวมที่คำนวณจากค่าน้อยสุดของค่าความเข้มของน้ำหนักเทียบกับฟังก์ชันสมาชิกของแต่ละกฎของ Fuzzy Rule และ Type 3 คือ เป็นการให้เหตุผลทาง Fuzzy ที่นำเสนอโดย Takagi และ Sugeno (Takagi and Sugeno, 1985) โดยค่าของการให้เหตุผล (Fuzzy Output) สามารถกำหนดได้สองแบบคือ Zero-order และ Single-order Polynomial โดยที่ถ้าเป็นแบบ Zero-order Polynomial ค่าผลลัพธ์จะเป็นค่าคงที่ (Singleton Constant) และถ้าเป็นแบบ Single-order Polynomial ผลลัพธ์จะเป็นค่าเฉลี่ยของผลลัพธ์ของแต่ละกฎ ซึ่งใช้สมการเชิงเส้นในการคำนวณ สำหรับฟังก์ชันสมาชิกพื้นฐานที่ใช้กับ FIS ที่นิยมได้แก่ Bell-shaped Membership Function, Gaussian Membership Function, Triangular Membership Function และ Trapezoidal Membership Function เป็นต้น

Adaptive Nero-Fuzzy Inference System: ANFIS เป็นแบบจำลองที่รวมเอาข้อดีของกระบวนการให้เหตุผลของ FIS มาทำงานรวมกันกับ Artificial Neural Network: ANN ซึ่งมีข้อดีในการเรียนรู้ข้อมูล และผู้ที่นำเสนอเทคนิคนี้ได้แก่ Jyh-Shing Roger Jan ในปี ค.ศ. 1933 (Jang, 1993)

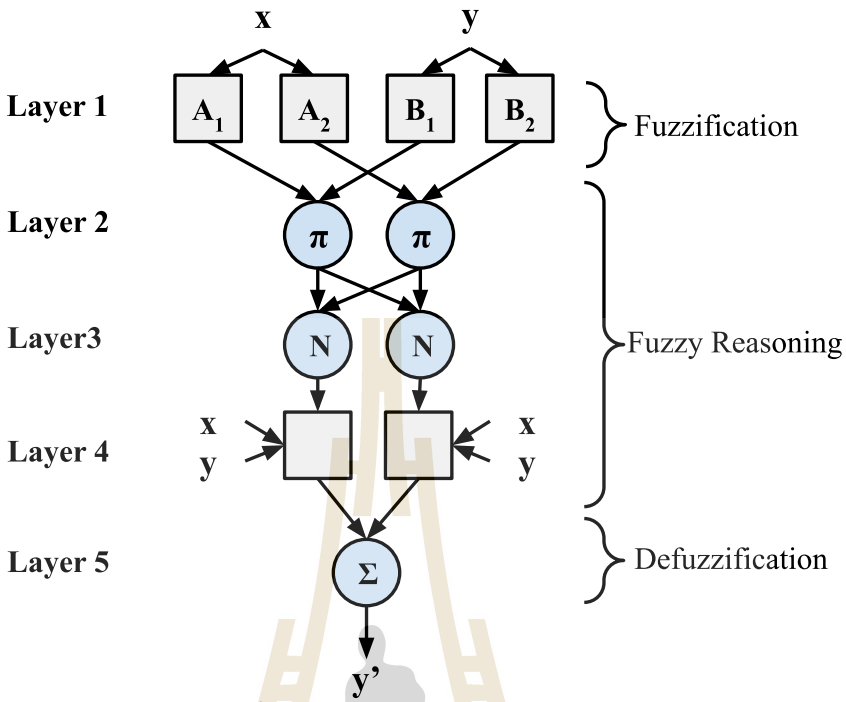
โดยปกติการทำงานของ FIS จะต้องทำงานร่วมกันกับผู้เชี่ยวชาญที่จะต้องใช้องค์ความรู้ในการสร้างกฎของ Fuzzy และกำหนดฟังก์ชันสมาชิกที่เหมาะสม ซึ่งในส่วนตรงนี้ ANFIS นำความสามารถของ ANN มาช่วยในการกำหนดค่าของกฎและปรับค่าน้ำหนักของแต่ละกฎผ่านการเรียนรู้จากชุดข้อมูลเพื่อให้มีความเหมาะสม และให้ได้ประสิทธิภาพในการทำนายผลข้อมูลหรือการจำแนกข้อมูลที่ดีที่สุด

ANFIS จะนิยมใช้ FIS ในรูปแบบที่ 3 (FIS Type 3) ซึ่งเป็นการใช้ Fuzzy If-then Rules และการให้เหตุผลทาง Fuzzy ของ Takagi และ Sugeno โดยแสดงตัวอย่างรูปแบบของ Fuzzy If-then Rules จำนวน 2 กฎ เมื่อมีสองตัวแปร Input คือ x กับ y และมีตัวแปร Output หนึ่งตัวคือ z

$$Rule\ 1: If\ x\ is\ A_1\ and\ y\ is\ B_1, Then\ f_1 = p_1x + q_1y + r_1,$$

$$Rule\ 2: If\ x\ is\ A_2\ and\ y\ is\ B_2, Then\ f_2 = p_2x + q_2y + r_2$$

โครงสร้างของ ANFIS จะมีทั้งหมด 5 ชั้นหลัก ๆ แสดงดังรูปที่ 2.11 โดยแต่ละชั้นจะทำหน้าที่ ดังต่อไปนี้



รูปที่ 2.11 แสดงโครงสร้างของ ANFIS

ชั้นที่ 1 จะเป็นชั้นที่มีการทำงานร่วมกันกับฟังก์ชันสมาชิก (แสดงเป็นรูปสี่เหลี่ยม) โดยจะเทียบค่าข้อมูลเข้า (Input) กับฟังก์ชันสมาชิกเพื่อให้ได้ค่าระดับทางความรู้สึกหรือค่าทางภาษา (Linguistic Values) เพื่อนำไปร่วมเป็นค่าน้ำหนักประจำโหนดในชั้นถัดไป แทนด้วยสมการที่ (2-9)

$$O_i^1 = \mu_{A_i}(x) \tag{2-9}$$

โดยที่ O_i^1 เป็นค่าข้อมูลออก (Output) ของค่าตัวแปร x ตำแหน่งที่ i ซึ่งคำนวณผ่านฟังก์ชันสมาชิกประจำโหนด ส่วน μ_{A_i} แทนฟังก์ชันสมาชิกประจำโหนดในชั้นที่หนึ่ง

ชั้นที่ 2 ทุกโหนดในชั้นนี้ (แสดงเป็นรูปวงกลม) จะทำหน้าที่รวมค่าน้ำหนักหรือค่าความเข้มข้นของข้อมูล ที่เข้ามายังโหนดโดยใช้การคูณ (Multiplication) เพื่อให้ได้ค่าน้ำหนักรวม w_i ดังสมการที่ (2-10)

$$w_i = \mu_{A_i}(x) \times \mu_{B_i}(y), i = 1, 2 \tag{2-10}$$

โดยที่ w_i เป็นค่าน้ำหนักรวมประจำโหนดที่ i ของชั้นที่สอง

ชั้นที่ 3 ทุกโหนดในชั้นนี้ (แสดงเป็นรูปวงกลม) จะทำหน้าที่ในการ Normalize ค่าน้ำหนักประจำโหนดจาก w_i เป็น $\overline{w_i}$ ดังสมการที่ (2-11)

$$\overline{w_i} = \frac{w_i}{w_1+w_2}, i = 1,2 \quad (2\text{-}11)$$

ชั้นที่ 4 (แสดงเป็นรูปสี่เหลี่ยม) เป็นชั้นที่ทำงานเกี่ยวกับส่วนของเหตุการณ์ที่เกิดขึ้นภายหลัง (Consequent) โดยจะทำหน้าที่คำนวณข้อมูลออก (Output) ผ่านค่าน้ำหนักประจำโหนด ดังสมการ (2-12)

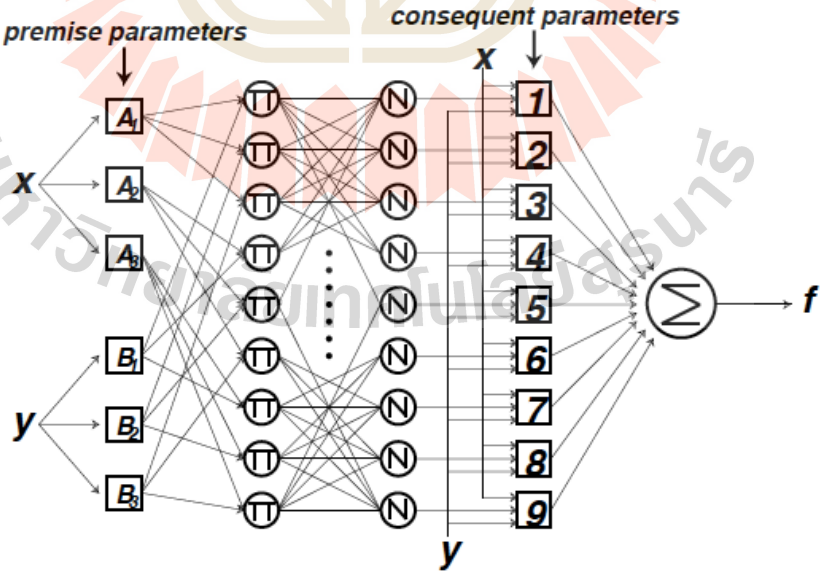
$$O_i^4 = \overline{w_i}\,(p_i x + q_i y + r_i), i = 1,2 \quad (2\text{-}12)$$

โดยที่ O_i^4 เป็นค่า Output ของแต่ละกฎของ Fuzzy Rules ในชั้นที่สี่, $\overline{w_i}$ เป็นค่าน้ำหนักประจำโหนดที่ผ่านการ Normalize, และ $\{p_i, q_i, r_i\}$ เป็นเซตของพารามิเตอร์ในส่วนของเหตุการณ์ที่เกิดขึ้นภายหลัง

ชั้นที่ 5 ในชั้นนี้จะมีเพียงแค่หนึ่งโหนดเท่านั้น (แสดงเป็นรูปวงกลม) และทำหน้าที่คำนวณผลรวมค่าน้ำหนักในแต่ละกฎของ Fuzzy Rules ให้ออกมาเป็นค่าผลลัพธ์ ดังสมการที่ (2-13)

$$O_i^5 = \sum_i \overline{w}_i f_i, i = 1,2 \quad (2\text{-}13)$$

โดยที่ O_i^5 เป็นค่าผลลัพธ์สุดท้าย (Output) ของโครงข่าย ANFIS

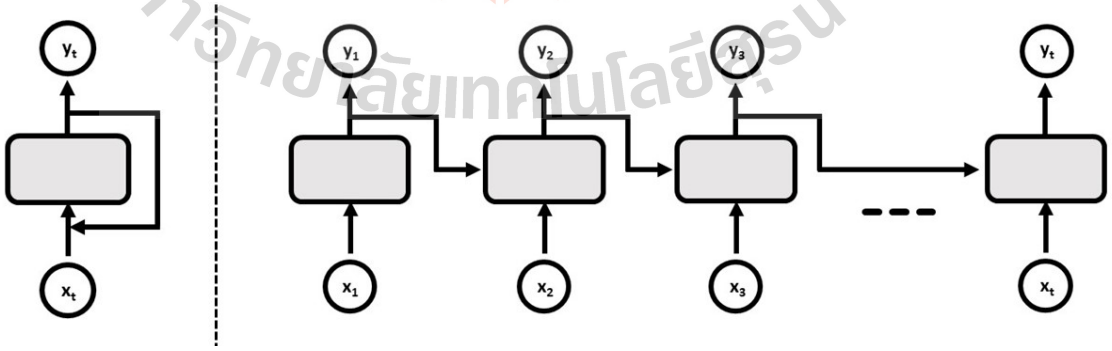


รูปภาพที่ 2.12 แสดงโครงสร้าง ANFIS Type-3 (2 ข้อมูลนำเข้า, 3 ฟังก์ชันสมาชิก, 9 Fuzzy Rules) (Jang, 1993)

ANFIS นำกระบวนการเรียนรู้ของ ANN ผ่านชุดข้อมูลในชั้นที่ 1 และ 4 (แสดงเป็นรูปสี่เหลี่ยม) ซึ่งเป็นชั้นที่มีการกระทำกับฟังก์ชันสมาชิก จะเห็นได้ว่าเป็นลักษณะของกระบวนการหาพารามิเตอร์ที่เหมาะสม (Hyper-parameters) ส่วนในชั้นอื่น ๆ ที่เหลือ (2, 3 และ 5) ซึ่งเป็นโหนดที่แสดงด้วยรูปวงกลมจะเป็นค่าคงที่ (สมการ) ไม่มีการเปลี่ยนแปลง เพื่อให้เห็นภาพได้ชัดเจนยิ่งขึ้น รูปภาพที่ 2.12 เป็นการแสดงโครงสร้างการทำงานทั้ง 5 ชั้นของ ANFIS ประเภทที่ 3 (Type 3) กับข้อมูลเข้า (Input) จำนวนสองตัวแปร x กับ y และมีหนึ่งตัวแปรข้อมูลผลลัพธ์ (Output) โดยตัวแปร Input แต่ละตัวแปร จะมี 3 ฟังก์ชันสมาชิก ดังนั้น จะทำให้มีจำนวนกฎของ Fuzzy Rules จำนวนทั้งหมด 9 กฎ (3 ยกกำลัง 2)

2.3.3 Long Short-Term Memory: LSTM

โครงข่ายประสาทเทียม Long Short Term Memory (LSTM) นำเสนอโดย Hochreiter & Schmidhuber (1997) เป็นโครงข่ายประสาทเทียมรูปแบบหนึ่งที่นำมาประมวลผลกับข้อมูลลำดับ (Sequence Data) เช่น ข้อมูลภาษา หรือข้อมูลอนุกรมเวลา เป็นต้น LSTM เป็นโครงข่ายประสาทเทียมที่จัดอยู่ในกลุ่มของการเรียนรู้เชิงลึก (Deep Learning) โดยตัว LSTM ถูกพัฒนาบนพื้นฐานของโครงข่ายประสาทเทียมแบบ RNN ซึ่งเป็นโครงข่ายที่มีการนำเอาข้อมูลขาออก (Output) วนกลับมาใช้ร่วมเป็นข้อมูลนำเข้า (Input) ดังรูปที่ 2.13 (a) โดยที่รูปที่ 2.13 (b) เป็นรูปแสดงโครงข่ายประสาทเทียมของ RNN แบบกางออก เพื่อใช้คำนวณข้อมูลลำดับ ปัญหาหลักของโครงข่ายประสาทเทียม RNN คือ ไม่สามารถจับ Long-term Dependency ซึ่งเป็นคุณสมบัติที่สำคัญของข้อมูลลำดับ เนื่องจากปัญหา Gradient Vanishing (Szandała, 2020) จากกระบวนการ Backpropagation ที่มีหลายลำดับชั้น ซึ่งค่า Gradient ที่จะนำไปใช้ร่วมกับค่า Learning Rate เพื่อปรับปรุงค่าน้ำหนัก (Weights) มีค่าน้อยมาก ๆ



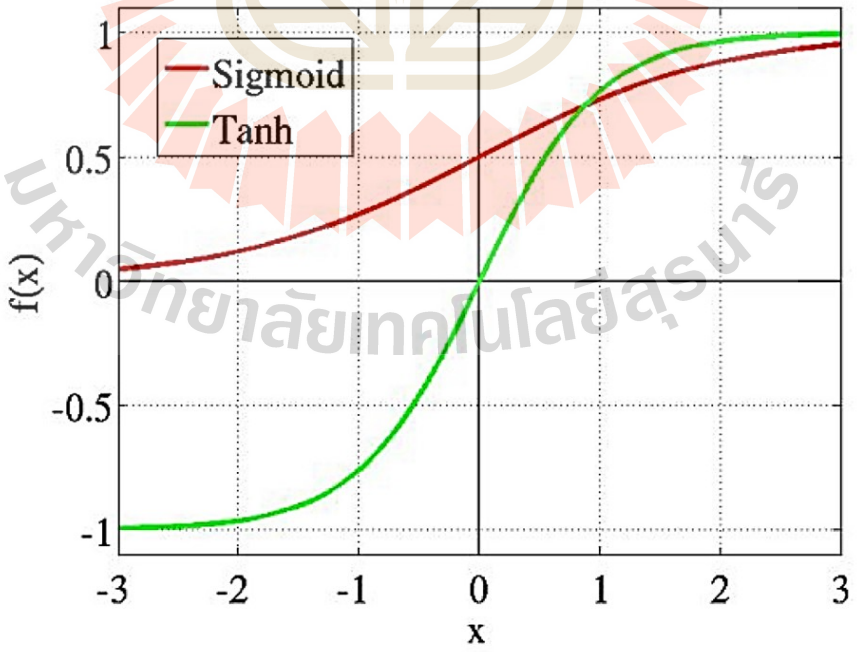
(a) (b)

รูปที่ 2.13 แสดง (a) รูปแบบ RNN และ (b) โครงข่าย RNN ที่ใช้กับข้อมูลลำดับ

ดังนั้น LSTM ถูกพัฒนามาเพื่อแก้ปัญหาของ RNN โดยการเพิ่มหน่วยความจำเพื่อเก็บ Long-term Dependency เรียกว่า Cell State หรือ Memory State อาจจะกล่าวได้ว่า Cell State เป็นช่องทางการนำ Gradient ก็ได้ นอกจากนี้ LSTM ยังถูกพัฒนาบน 3 หลักการร่วมกัน คือ 1) ข้อมูลบางข้อมูลควรจะมีการถูกลืมได้บ้าง 2) ข้อมูลบางข้อมูลอาจจะเป็นข้อมูลปนเปื้อน (Noise) ก็ไม่ควรนำมาพิจารณา และ 3) ข้อมูลบางข้อมูลควรมีการขยาย (Scale) หรือกรอง (Filter) ซึ่งแนวคิดตามหลักการทั้งสามดังกล่าวทำให้เกิดการออกแบบ Forget Gate (f), Input Gate (i) และ Output Gate (o) ตามลำดับ โดยที่ Gate ทั้งสามแบบจะมี Activation Function กำหนดอยู่ โดย LSTM มาตราฐานจะกำหนด Sigmoid Function ดังสมการที่ (2-14) เป็นตัวควบคุมข้อมูลให้อยู่ในช่วง [0, 1] คือ ถ้าค่าเป็น 0 หมายความว่า ไม่ต้องส่งค่านั้นออกไป และถ้าค่าเป็น 1 ก็จะเป็นการส่งค่าออกไปยังโหนดต่อไป และจะมีการขยายหรือกรองข้อมูลในส่วนของ input และส่วนของ output ด้วย Tanh function ดังสมการที่ (2-15) รูปที่ 2.14 แสดงเส้นกราฟเปรียบเทียบระหว่าง Sigmoid Function กับ Tanh Function

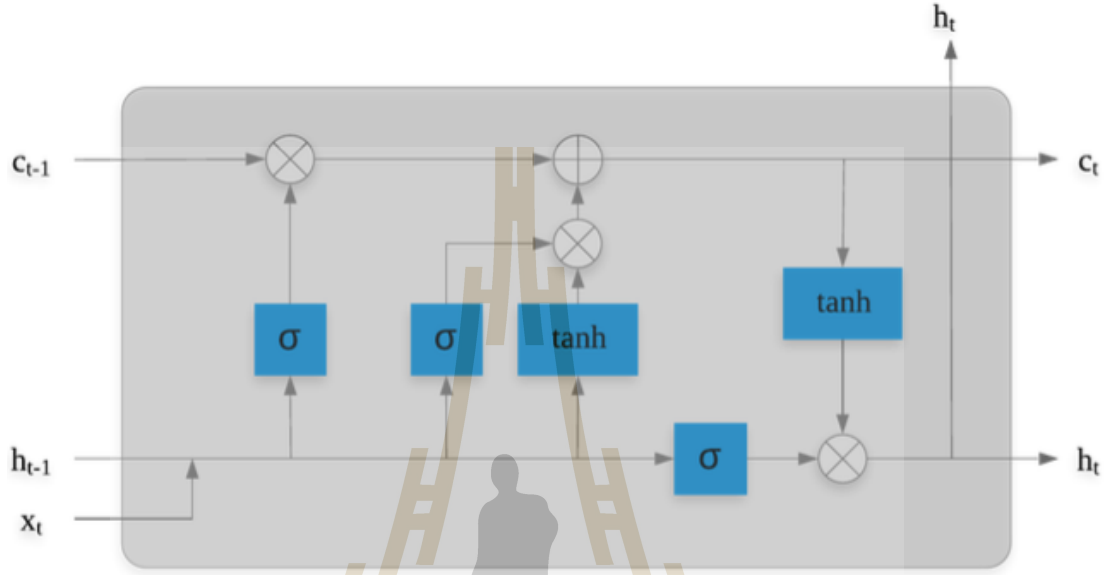
$$Sigmoid(x) = \frac{1}{1+e^{-x}} \tag{2-14}$$

$$Tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2-15}$$

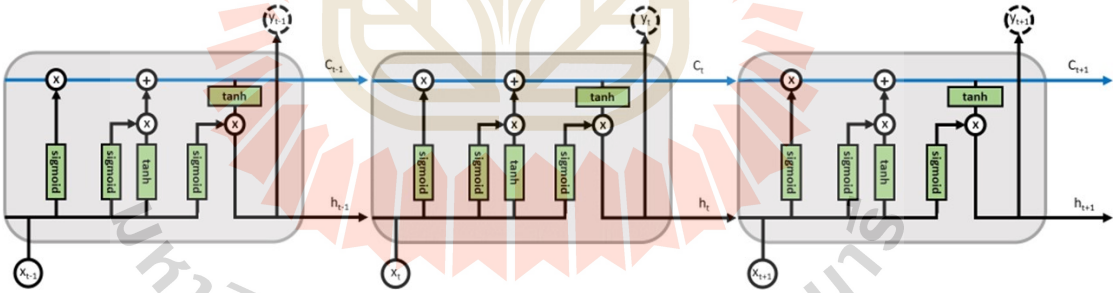


รูปที่ 2.14 แสดงกราฟเปรียบเทียบระหว่าง Sigmoid Function กับ Tanh Function (Szandała, 2020)

จากรูปที่ 2.15 แสดงโครงสร้างภายในหนึ่งหน่วยย่อยของโครงข่ายประสาทเทียม LSTM และเมื่อนำโครงข่ายประสาทเทียม LSTM ต่อกับเป็นลำดับ (Sequence) เพื่อทำงานกับข้อมูลลำดับ จะได้ดังรูปที่ 2.16



รูปที่ 2.15 แสดงโครงสร้างภายในหนึ่งหน่วยย่อยของ LSTM (Zhang, Li, Lam, & Han, 2020)



รูปที่ 2.16 แสดงโครงสร้างของ LSTM

พิจารณาการส่งผ่านข้อมูลภายในโครงข่ายประสาทเทียม LSTM จะมีการทำงานอยู่ 3 ส่วน ตามประเภทของ Call Gate ดังนี้ ส่วนที่หนึ่ง Forget Gate เป็นขั้นตอนที่จะพิจารณาว่าจะเลือกเก็บหรือทิ้งข้อมูลที่อยู่ในหน่วยความจำหรือ Cell State การตัดสินใจในขั้นตอนนี้จะควบคุมผ่าน Sigmoid Function โดยใช้ข้อมูลจาก h_{t-1} และ x_t เมื่อผลลัพธ์ที่ได้เป็น 0 หมายถึงการเลือกทิ้งหรือลืมข้อมูล C_{t-1} และถ้าผลลัพธ์ที่ได้เป็น 1 หมายถึงการเลือกเก็บหรือจำข้อมูล C_{t-1} โดยแสดงในรูปแบบสมการ ดังสมการ (2-16)

$$f_t = Sigmoid(W_f \times [h_{t-1}, x_t] + B_f) \quad (2\text{-}16)$$

โดยที่ f_t คือค่าผลลัพธ์ของ Forget Gate, W_f คือค่าน้ำหนักของ Forget Gate, B_f คือค่า Bias ของ Forget Gate, และ $\{h_{t-1}, x_t\}$ คือค่าผลพยากรณ์ก่อนหน้า และค่าข้อมูลนำเข้า ณ เวลา t ตามลำดับ

ส่วนที่สอง Input Gate เป็นขั้นตอนที่พิจารณาข้อมูลใหม่ที่จะเก็บไว้ในหน่วยความจำหรือ Cell State (Update) โดยขั้นตอนนี้จะมีอยู่ 2 ขั้นตอนย่อย ขั้นตอนย่อยที่หนึ่งคือการกำหนดข้อมูลที่จะ update ด้วย Sigmoid Function โดยใช้ข้อมูลนำเข้า h_{t-1} และ x_t (สมการที่ 2-17) ต่อมานำข้อมูลนำเข้า h_{t-1} และ x_t มาผ่าน Tanh Function เพื่อปรับค่าข้อมูลให้อยู่ในช่วง [-1, 1] (สมการที่ 2-18) ในขั้นตอนย่อยที่สองจะเป็นการ update ข้อมูลในหน่วยความจำดังสมการที่ (2-19)

$$i_t = Sigmoid(W_i \times [h_{t-1}, x_t] + B_i) \quad (2\text{-}17)$$

$$C'_t = Tanh(W_C \times [h_{t-1}, x_t]) + B_C \quad (2\text{-}18)$$

$$C_t = (f_t \times C_{t-1}) + (i_t \times C'_t) \quad (2\text{-}19)$$

โดยที่ i_t คือค่าสำหรับกำหนดการ Update Cell State, C'_t คือค่าของข้อมูลนำเข้าสำหรับ Update Cell State, C_t คือค่าผลลัพธ์ของการ Update ข้อมูลของ Input Gate, และ $\{W_i, W_c, B_i, B_c\}$ คือค่าน้ำหนักและค่า Bias ของ Input Gate ตามลำดับ

ส่วนที่สาม Output Gate เป็นขั้นตอนสุดท้ายเพื่อพิจารณาว่าข้อมูลที่อยู่ในหน่วยความจำ Cell State จะมีอิทธิพลต่อข้อมูลพยากรณ์ h_t หรือ y_t หรือไม่ โดยจะคำนวณจากข้อมูลในหน่วยความจำที่ผ่านการกรองด้วย Tanh Function เพื่อให้ค่าอยู่ในช่วง [-1, 1] คูณกับค่าของ h_{t-1} และ x_t ที่ผ่านการ Activated ด้วย Sigmoid Function ดังสมการที่ (2-20) และ (2-21)

$$o_t = Sigmoid(W_o[h_{t-1}, x_t] + B_o) \quad (2\text{-}20)$$

$$h_t = o_t \times Tanh(C_t) \quad (2\text{-}21)$$

โดยที่ o_t คือค่าพิจารณาอิทธิพลของข้อมูลในหน่วยความจำ มีผลต่อการพยากรณ์หรือไม่, h_t คือผลพยากรณ์, และ $\{W_o, B_o\}$ คือค่าน้ำหนักและค่า Bias ของ Output Gate ตามลำดับ

นอกจากโครงข่ายประสาทเทียม LSTM มาตราฐานตามที่ได้กล่าวไปแล้ว ก็ยังมีการนำโครงข่ายประสาทเทียม Gated Recurrent Unit (GRU) ที่นำโครงข่ายประสาทเทียม LSTM มาตราฐานไปปรับปรุงโดยการรวม Forget Gate กับ Input Gate เพื่อให้การ Update ข้อมูลในหน่วยความจำ Cell State กระทำเพียงขั้นตอนเดียว ทำให้ลดความซับซ้อนของแบบจำลองและระยะเวลาในการประมวลผล (Chung, Gulcehre, Cho & Bengio, 2014)

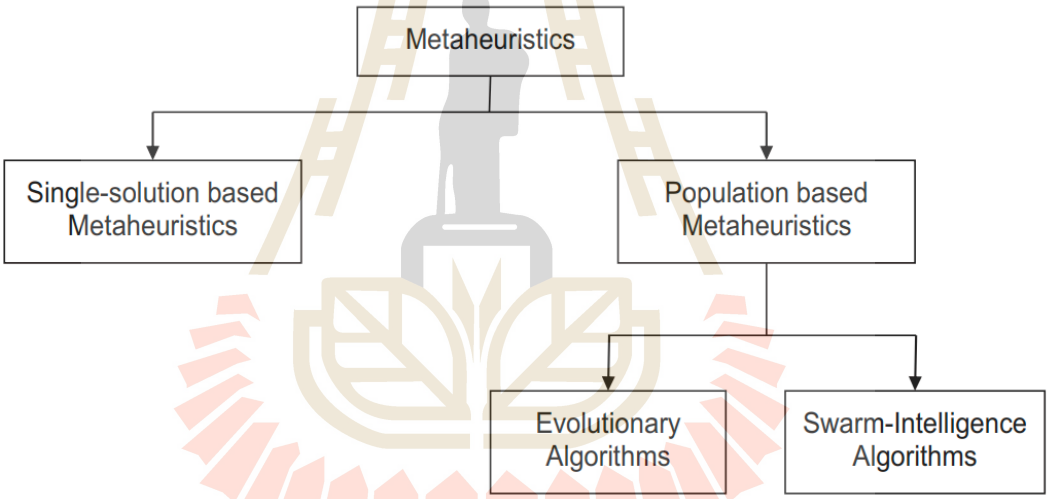
2.4 Metaheuristic Algorithms

ในช่วง 10 ปี ที่ผ่านมา ได้มีการพัฒนาและนำเสนอ Metaheuristic Algorithms จำนวนไม่น้อยกว่า 500 อัลกอริทึม (Rajwar, Deep, & Das, 2023) สำหรับใช้แก้ปัญหางานด้าน Optimization ซึ่งสามารถจำแนก Metaheuristic Algorithm ออกเป็น 2 รูปแบบ คือ รูปแบบที่หนึ่ง Single-solution based Metaheuristic และรูปแบบที่สอง Population based Metaheuristic โดย Single-solution based Metaheuristic จะเป็นการหาหรือปรับปรุง Solution สำหรับคำตอบของปัญหาแบบ Local Optimum ซึ่งจะได้ Solution ที่แน่นอนและแม่นยำ แต่บางกรณีอาจจะติดปัญหาเรื่อง Local Optimum Trap ส่วน Population based Metaheuristic เป็นการหาหรือปรับปรุง Solution สำหรับคำตอบของปัญหาในหลาย Solution พร้อมกัน (Parallel Process) เรียกว่า Global Optimum ซึ่งวิธีนี้จะช่วยแก้ปัญหาของ Local Optimum Trap ของการหา Solution แบบ Single-solution based Metaheuristic แต่ Solution สำหรับคำตอบของปัญหาที่ได้จาก Population based Metaheuristic อาจจะไม่ใช่ Solution ที่ดีที่สุด และใช้เวลานานในการค้นหา Solution รวมถึงใช้ Computing Power ที่สูงด้วยเช่นกัน รูปที่ 2.17 แสดงประเภท Metaheuristic Algorithms ที่ได้กล่าวไปข้างต้น จากข้อมูลเชิงสถิติพบว่างานวิจัยส่วนใหญ่จะใช้ Metaheuristic Algorithms แบบ Population Based Metaheuristic Algorithms ในการแก้ปัญหา โดยที่ได้รับความนิยมสูงสุดคือ อัลกอริทึม Particle Swarm Optimization: PSO และ Genetic Algorithm: GA

2.4.1 Genetic Algorithm: GA

Genetic Algorithm หรือ ระเบียบวิธีเชิงพันธุกรรม เป็นอัลกอริทึมที่ถูกจัดอยู่ในกลุ่มของ Population Based Metaheuristic Algorithm แบบ Evolutionary Algorithm นำเสนอโดย John Holland ในปี ค.ศ. 1975 (Holland, 1975) ซึ่ง GA มีกระบวนการหาคำตอบของปัญหาตามหลักการดำรงเผ่าพันธุ์ของสิ่งมีชีวิต กล่าวคือ สิ่งมีชีวิตที่แข็งแรงเท่านั้นจะสามารถอยู่รอดและ

ดำรงเผ่าพันธุ์ต่อไป โดยกระบวนการทำงานของของ GA สามารถแบ่งออกเป็น 3 ขั้นตอนหลัก ดังนี้ ขั้นตอนแรก เป็นขั้นตอนของการกำหนดประชากรในที่นี้คือ Solution ที่ทำให้ได้คำตอบของปัญหา ขั้นตอนที่สอง เป็นขั้นตอนคำนวณค่า Fitness ของแต่ละ Solution เพื่อหา Solution ที่จะทำให้ได้คำตอบของปัญหาดีที่สุด หากในขั้นตอนนี้ ได้ Solution ที่ดีที่สุดตามความต้องการแล้ว ก็จะหยุดกระบวนการทำงานของ GA ถ้าไม่ได้ก็จะทำขั้นตอนที่สามคือ การทำกระบวนการทางพันธุกรรม ซึ่งเป็นกระบวนการสำคัญของ GA มีอยู่ 3 กระบวนการย่อย ได้แก่ กระบวนการคัดเลือก (Selection) กระบวนการผสมพันธุ์ (Crossover) และกระบวนการกลายพันธุ์ (Mutation) ซึ่งทั้งสามกระบวนการย่อยนี้จะช่วยปรับปรุง Solution ให้ได้ Solution ที่ให้คำตอบของปัญหาดียิ่งขึ้น ต่อจากนั้นจะส่งกลับไปยังขั้นตอนที่สอง และจะทำงานเป็นวงรอบจนกว่าจะเป็นไปตามเงื่อนไขของการสิ้นสุดการทำงานที่กำหนด โดยแสดงกระบวนการทำงานของ GA ในรูปที่ 2.18



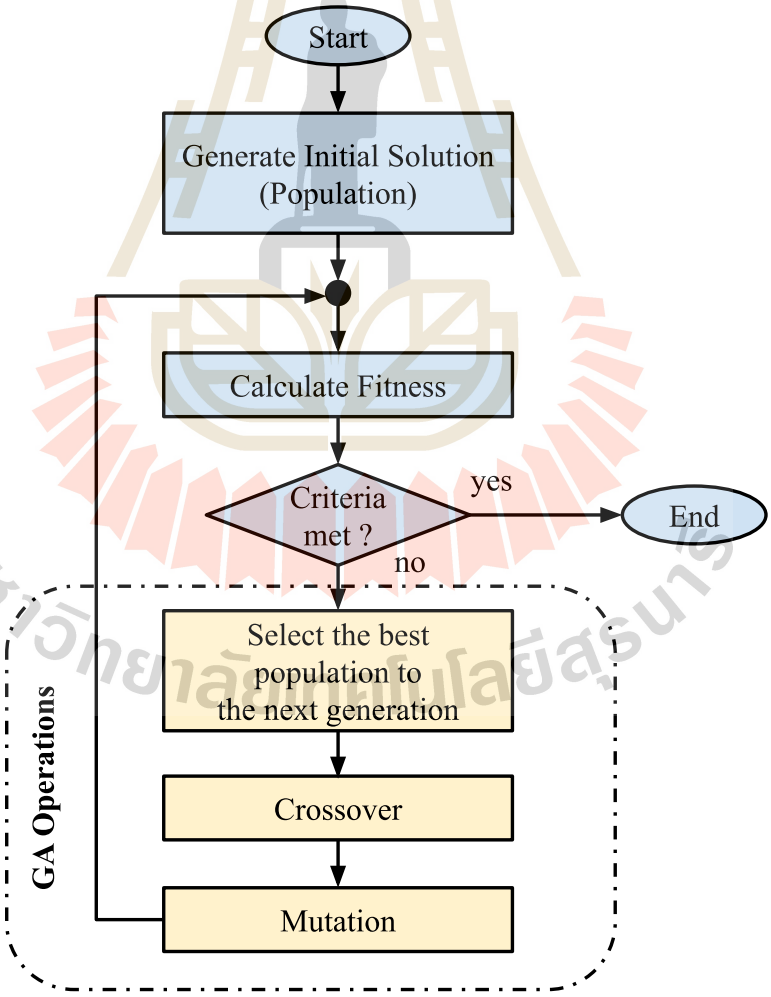
รูปที่ 2.17 ประเภทของ Metaheuristic Algorithm (Katoch, Chauhan, & Kumar, 2021)

กระบวนการคัดเลือกของระเบียนวิธีเชิงพันธุกรรม เป็นกระบวนการเปลี่ยนประชากร หรือกลุ่ม Solution จากรุ่นเก่าไปสู่รุ่นใหม่ โดยหลักการจะเลือกประชากรที่มีคุณภาพ กล่าวอีกนัยหนึ่งคือ เลือก Solution ที่ให้คำตอบของปัญหาดีที่สุด โดยประเมินผ่าน Fitness Function สำหรับระเบียบวิธีเชิงพันธุกรรมจะมีกระบวนการคัดเลือกหลายวิธี แต่วิธีที่ได้รับความนิยมและมีประสิทธิภาพที่สุดวิธีหนึ่งคือ วิธีคัดเลือกแบบ Roulette Wheel Selection ซึ่งกระบวนการคัดเลือกแบบ Roulette Wheel Selection จะสอดคล้องตามหลักการคัดเลือกทางธรรมชาติ คือ จะอาศัยหลักของความน่าจะเป็นในการเลือกประชากร หรือ Solution โดยกำหนดความน่าจะเป็นที่จะถูกเลือกให้กับประชากร หรือ Solution ตามความแข็งแรงของประชากร หรือประสิทธิภาพของ

Solution กล่าวคือ ถ้าประชากรที่มีความแข็งแรงมากก็จะมีโอกาสถูกเลือกสูง ตรงกันข้ามถ้าประชากรมีความแข็งแรงน้อยก็จะมีโอกาสถูกเลือกต่ำ สมการ (2-22) แสดงความน่าจะเป็นของการเลือกประชากรผ่านกระบวนการคัดเลือกแบบ Roulette Wheel Selection และรูปที่ 2.19 แสดงรูปเทียบเคียงกงล้อ Roulette ในวิธีการคัดเลือกแบบ Roulette Wheel Selection

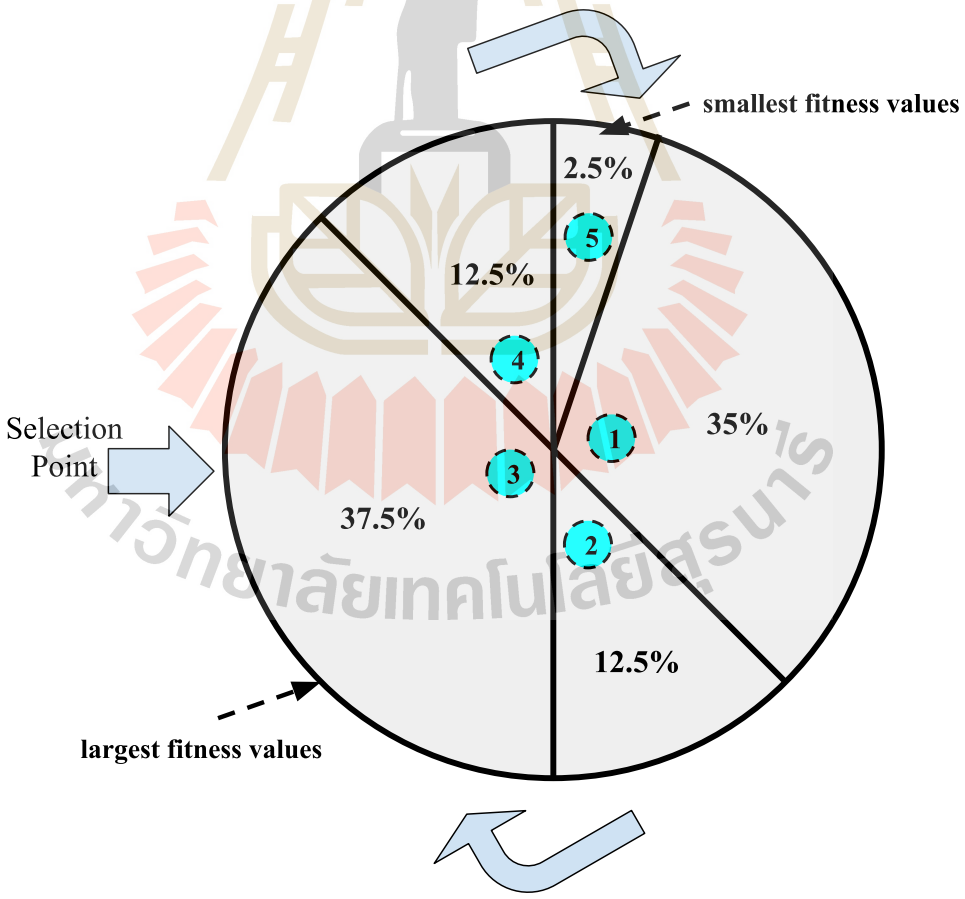
$$p_i = \frac{f_i}{TF} \tag{2-22}$$

โดยที่ p_i คือ ค่าความน่าจะเป็นการถูกคัดเลือกของประชากรหรือ Solution i โดยที่ i มีค่าตั้งแต่ 1 ถึงจำนวนประชากรทั้งหมด ส่วน f_i คือ ค่า Fitness ของประชากรหรือ Solution i โดยที่ i มีค่าตั้งแต่ 1 ถึงจำนวนประชากรทั้งหมด และ TF คือผลรวมของค่า Fitness ของประชากรหรือ Solution ทั้งหมด

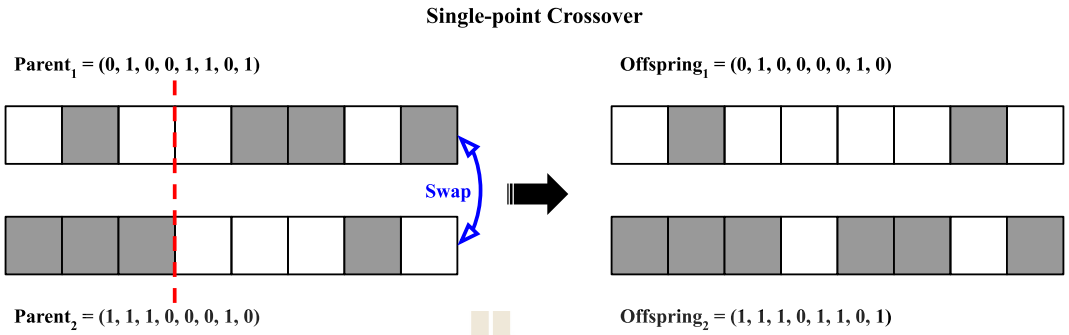


รูปที่ 2.18 แสดงกระบวนการทำงาน GA

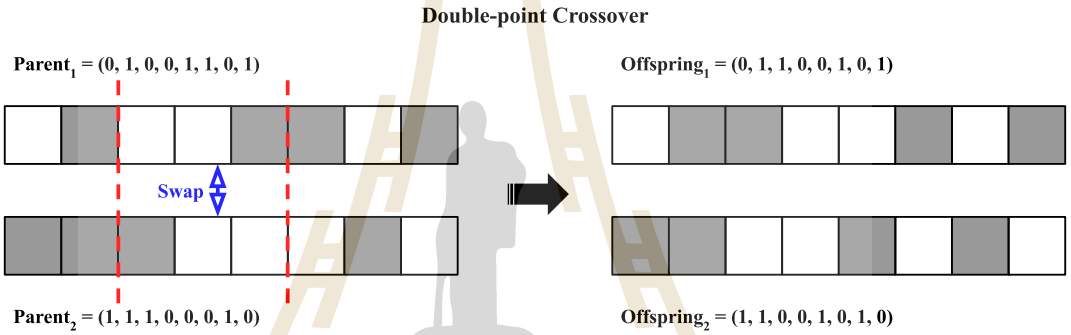
กระบวนการผสมพันธุ์ เป็นกระบวนการสำหรับสร้างประชากรรุ่นใหม่ หรือ Solution ใหม่ ๆ จากการคัดเลือกประชากรที่มีความแข็งแรงในรุ่นเดิมให้เป็น Parents ด้วยกระบวนการคัดเลือก จากนั้นนำคู่ Parents มาดำเนินกระบวนการผสมพันธุ์ให้เกิดเป็นประชากรรุ่นใหม่ เรียกว่า Offspring สำหรับกระบวนการผสมพันธุ์ของระเบียบวิธีเชิงพันธุกรรมที่นิยมใช้ ได้แก่ Single Point Crossover, Double Point Crossover และ Uniform Crossover โดยที่กระบวนการผสมพันธุ์แบบ Single และ Double Point Crossover จะบ่งบอกถึงจำนวนจุดตัดบนประชากรที่เป็นคู่ Parents สำหรับการผสมพันธุ์ให้เกิด Offspring โดยที่ Single และ Double Point Crossover จะมีจำนวนจุดตัดที่ 1 และ 2 จุดตัดตามลำดับ แสดงรูปตัวอย่างจุดตัดและการสร้าง Offspring ของ Single และ Double Point Crossover ของปัญหาที่เป็นแบบ Binary Digit ในรูปที่ 2.20 และ 2.21 ตามลำดับ ส่วน Uniform Crossover จะเป็นการผสมพันธุ์ระหว่างคู่ Parents เพื่อให้เกิด Offspring แบบซุ่ม กล่าวคือ เป็นการซุ่มเลือกโครโมโซมของของคู่ Parents มาสร้างเป็น Offspring ส่วนมากจะใช้เทคนิคการซุ่มเลือกแบบโยนเหรียญ (Tossing of a Coin)



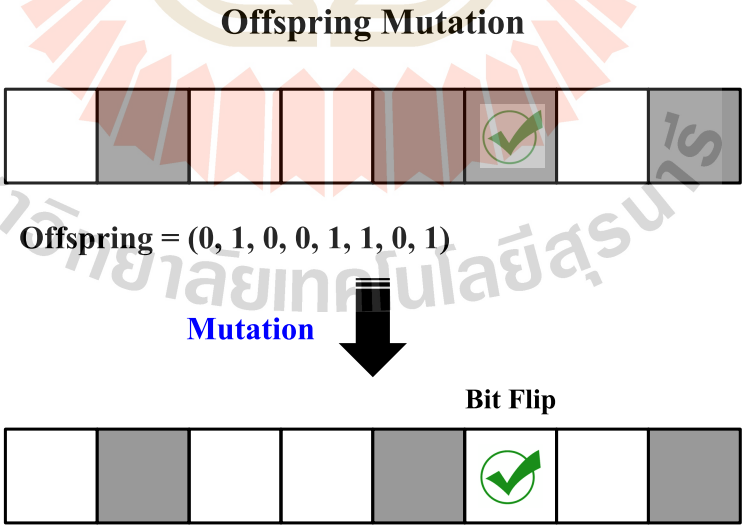
รูปที่ 2.19 แสดงการคัดเลือกของระเบียบวิธีเชิงพันธุกรรมแบบ Roulette Wheel Selection



รูปที่ 2.20 กระบวนการผสมพันธุ์แบบ Single-point Crossover ของ GA



รูปที่ 2.21 กระบวนการผสมพันธุ์แบบ Double-point Crossover ของ GA



รูปที่ 2.22 กระบวนการกลายพันธุ์ของ GA

กระบวนการกลายพันธุ์ เป็นกระบวนการหนึ่งที่เกิดขึ้นในธรรมชาติในการดำรงเผ่าพันธุ์ของสิ่งมีชีวิต ซึ่งจะมีโอกาสเกิดน้อยมาก กระบวนการกลายพันธุ์ในระเบียบวิธีเชิงพันธุกรรมจะเป็นการเปลี่ยน Offspring ให้แตกต่างไปจาก Parents โดยการซุ่ม Flip โครโมโซมของ Offspring แสดงดังรูปที่ 2.22

2.4.2 Particle Swarm Optimization: PSO

อัลกอริทึมในกลุ่ม Swarm Intelligent เป็นอัลกอริทึมที่ได้รับแรงบันดาลใจจากพฤติกรรมของสัตว์หรือแมลงในการหาแหล่งอาหาร ซึ่งอัลกอริทึมที่ได้รับความนิยมในกลุ่ม Swarm Intelligent ได้แก่ Particle Swarm Optimization: PSO, Ant Colony Optimization: ACO, Salp Swarm Algorithm: SSA, Grey Wolf Optimization: GWO เป็นต้น และในช่วงหนึ่งทศวรรษที่ผ่านมา จากข้อมูลทางสถิติพบว่า PSO เป็นอัลกอริทึมที่ได้รับความนิยมสูงสุด (Rajwar, Deep, & Das, 2023)

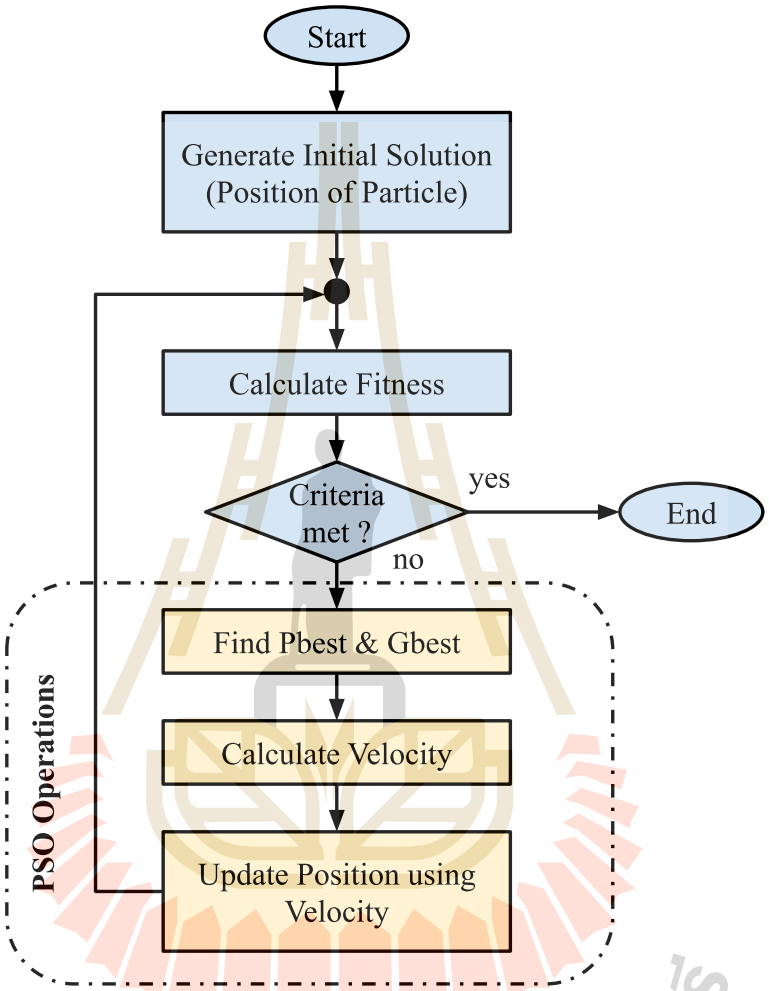
PSO เป็น Search Algorithm ที่ใช้สำหรับหา Solution ที่ดีที่สุดเพื่อให้ได้คำตอบของปัญหา ทั้งแบบ Single Objective และ Multi Objective ซึ่งถูกนำเสนอใน ปี ค.ศ. 1995 โดย Kennedy และ Eberhart (1995) หลักการทำงานของ PSO ได้แนวความคิดมาจากพฤติกรรมในการหาแหล่งอาหารของฝูงนกหรือฝูงปลาตามธรรมชาติ โดยที่สัตว์นั้นจะอยู่รวมกันเป็นฝูง และเมื่อเวลาหาแหล่งอาหารก็จะออกหากันเป็นฝูงเช่นกัน โดยพฤติกรรมการหาแหล่งอาหารจะมีลักษณะ ดังนี้ คือ สัตว์แต่ละตัวที่อยู่ในฝูงก็จะหาและพบแหล่งอาหารที่ดีที่สุดสำหรับตัวเอง เรียกว่า Personal Best หรือ Particle Best และสัตว์เหล่านี้ก็จะมีการแลกเปลี่ยนข้อมูลระหว่างกันในฝูงเพื่อให้ได้ข้อมูลของแหล่งอาหารที่ดีที่สุดสำหรับทั้งฝูง ซึ่งจะเรียกว่า Global Best หรือ Social Best จากนั้นสัตว์ทั้งฝูงก็จะทยอยเคลื่อนที่ไปยังตำแหน่งของแหล่งอาหารที่ดีที่สุดผ่านอิทธิพลของสองค่าดังกล่าวคือ Personal Best และ Global Best ตามลำดับ แสดงสมการการเคลื่อนที่ของฝูงสัตว์ในการหาแหล่งอาหาร ดัง (2-23) และ (2-24) ซึ่งเป็นสมการของการหาความเร็วเคลื่อนที่และการเปลี่ยนตำแหน่งไปยังแหล่งอาหารนั้นของฝูงสัตว์ ตามลำดับ

$$v_{i,t+1} = (w \times v_{i,t}) + \left[c_1 \times r_1 \times \left(Pbest_{i,t} - x_{i,t}\right)\right] + \left[c_2 \times r_2 \times \left(Gbest_t - x_{i,t}\right)\right] \quad (2\text{-}23)$$

$$x_{i,t+1} = x_{i,t+1} + v_{i,t+1} \quad (2\text{-}24)$$

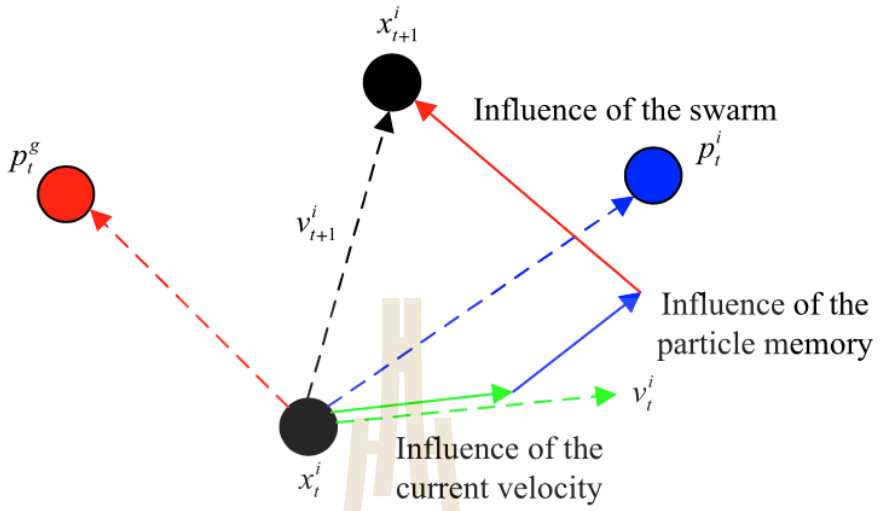
โดยที่ v_i คือความเร็วเคลื่อนที่ของ Particle i, x_i คือตำแหน่งของ Particle i, $Pbest_i$ และ $Gbest$ คือ ตำแหน่งที่ดีที่สุดของ Particle i และตำแหน่งที่ดีที่สุดของทั้งฝูง ตามลำดับ

, $\{w, c_1, c_2\}$ ค่าคงที่สำหรับถ่วงน้ำหนัก, และ $\{r_1, r_2\}$ คือค่าสุ่มระหว่าง [0, 1] สำหรับปรับระยะการเคลื่อนที่ของ Particle



รูปที่ 2.23 แสดงกระบวนการทำงาน PSO

กระบวนการทำงานของ PSO แสดงดังรูปที่ 2.24 โดยแบ่งขั้นตอนการทำงานหลัก 3 ขั้นตอน ดังนี้ ขั้นตอนแรกเป็นการสุ่มสร้าง Particle และตำแหน่งเริ่มต้นของ Particle จากนั้นส่งไปยังขั้นตอนที่สองเพื่อคำนวณค่า Fitness ของแต่ละ Particle หากในขั้นตอนนี้ ได้ Solution ที่ดีที่สุดตามความต้องการแล้ว ก็จะหยุดกระบวนการทำงานของ PSO ถ้าไม่ได้ก็จะทำขั้นตอนที่สามคือ การปรับตำแหน่งและความเร็วเคลื่อนที่ของ Particle Swarm ตามสมการที่ (2-23) และ (2-24) และแสดงการเปลี่ยนตำแหน่งของ Particle ผ่านอิทธิพลของ $Pbest_i$, $Gbest$, และความเร็วเคลื่อนที่เดิมของ Particle ดังรูปที่ 2.23



รูปที่ 2.24 แสดงการเคลื่อนที่ของ Particle ของ PSO (Wang, Tan, & Liu, 2017)

2.5 มาตรวัดประสิทธิภาพ

แบบจำลองการเรียนรู้ของเครื่องในกลุ่มของการทำนายและพยากรณ์ข้อมูล การประเมินประสิทธิภาพจึงเป็นการวัดความคลาดเคลื่อนของผลลัพธ์จากค่าที่แท้จริง (Error) หรือค่าเบี่ยงเบนของผลลัพธ์ต่อค่าที่แท้จริง (Deviation) ดังสมการที่ (2-25) และแสดงการวัดความคลาดเคลื่อนของแบบจำลองที่อยู่ในลักษณะสมการเส้นตรงได้ดังรูปที่ 2.25 โดยที่มาตรวัดประสิทธิภาพที่ใช้จะเป็นมาตรวัดทางสถิติพื้นฐาน ที่นิยมได้แก่ ค่ากลางของความคลาดเคลื่อนกำลังสอง (Mean Square Error: MSE) แสดงดังสมการที่ (2-26) และ ค่ากลางของความคลาดเคลื่อนสัมบูรณ์ (Mean Absolute Error: MAE) แสดงดังสมการที่ (2-27)

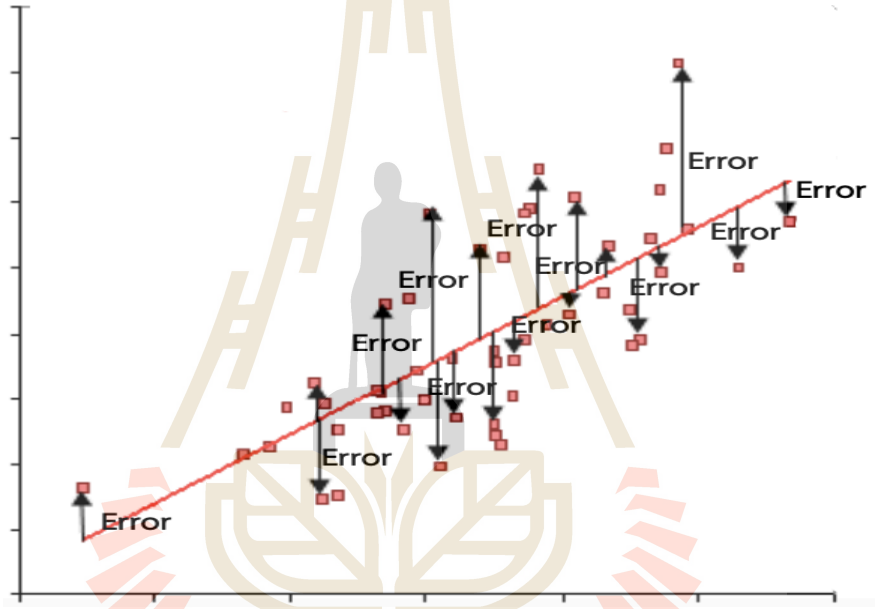
$$Error = Actual\ Value - Predicted\ Value \tag{2-25}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - x'_i)^2 \tag{2-26}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - x'_i| \tag{2-27}$$

โดยที่ x_i คือ ค่าที่แท้จริงในลำดับที่ i และ x'_i คือ ค่าของการทำนายในลำดับที่ i ค่า MSE และ MAE กำจัดเครื่องหมายของค่าความคลาดเคลื่อนโดย MSE ใส่กำลังสองให้กับค่าความคลาดเคลื่อน ส่วน MAE ใส่สัมบูรณ์ให้กับค่าความคลาดเคลื่อน เพราะค่าความคลาดเคลื่อนเป็นข้อมูลตัวเลขแบบเวกเตอร์ จึงไม่สามารถนำเครื่องหมายมาคำนวณได้ ตารางที่ 2.3 แสดงตัวอย่างค่าความ

คลาดเคลื่อนของผลลัพธ์จากการทำนายต่อค่าที่แท้จริงของ MSE เทียบกับ MAE ซึ่งข้อดีของการใช้มาตรวัดประสิทธิภาพ MSE จะทำให้แสดงผลของความคลาดเคลื่อนอย่างชัดเจนและตัวสมการสามารถประยุกต์ใช้งานได้สะดวกกว่า MAE เมื่อต้องทำงานกับคอมพิวเตอร์ ส่วนข้อดีของ MAE คือแสดงค่าความคลาดเคลื่อนที่แท้จริง ส่วนมากจะนิยมใช้ในการอ้างอิงทางสถิติ สำหรับมาตรวัดประสิทธิภาพความคลาดเคลื่อนที่นิยมใช้วัดประสิทธิภาพแบบจำลองการเรียนรู้ของเครื่อง ได้แก่ ราก ที่สองของค่ากลางความคลาดเคลื่อนกำลังสอง (Root Mean Square Error: RMSE) และ ค่ากลางของเปอร์เซ็นต์ความคลาดเคลื่อนสัมบูรณ์ (Mean Absolute Percentage Error: MAPE)



รูปที่ 2.25 แสดงความคลาดเคลื่อนของผลลัพธ์พยากรณ์ผ่านสมการเส้นตรง

ตารางที่ 2.3 แสดงตัวอย่างค่าความคลาดเคลื่อน MSE และ MAE

No.	Actual	Predicted	Error	$Error^2$	|Error|
1	10	15	-5	25	5
2	5	4	1	1	1
3	25	22	3	9	3
4	13	14	-1	1	1
5	10	12	-2	4	2
		รวม		44	12
				MSE = 8.8	**MAE = 2.4**

2.5.1 มาตรวัดความคลาดเคลื่อน RMSE

มาตรวัดความคลาดเคลื่อน RMSE เป็นการนำเอา MSE มาลดทอนค่าคลาดเคลื่อนส่วนเกินจากการใส่กำลังสอง โดยนำรากที่สองมาใส่ MSE แสดงดังสมการ (2-28)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - x'_i)^2} \quad (2\text{-}28)$$

โดยที่ x_i คือ ค่าที่แท้จริงในลำดับที่ i และ x'_i คือ ค่าของการทำนายในลำดับที่ i

2.5.2 มาตรวัดความคลาดเคลื่อน MAPE

มาตรวัดความคลาดเคลื่อน MAPE เป็นการนำเอา MAE มาปรับค่าให้อยู่ในรูปผลลัพธ์ของเปอร์เซ็นต์ เพื่อทำให้สะดวกและเข้าใจง่ายในการอ่านผล โดยแสดงดังสมการ (2-29)

$$MAPE = \left(\frac{1}{n}\sum_{i=1}^{n}\left|\frac{x_i - x'_i}{x_i}\right|\right) \times 100 \quad (2\text{-}29)$$

โดยที่ x_i คือ ค่าที่แท้จริงในลำดับที่ i และ x'_i คือ ค่าของการทำนายในลำดับที่ i

2.6 งานวิจัยที่เกี่ยวข้อง

ในช่วง 10 ปีที่ผ่านมาปัญหาด้านสิ่งแวดล้อมเป็นสิ่งที่ผู้คนทั่วโลกต่างให้ความสนใจโดยเฉพาะอย่างยิ่งปัญหามลพิษทางอากาศ ซึ่งเป็นปัญหาที่สำคัญมากเพราะกระทบทั้งด้านเศรษฐกิจและสังคมโดยกว้าง กล่าวอีกนัยหนึ่งก็คือ เป็นปัญหาที่มีผลกระทบต่อการพัฒนาของเมืองและมีผลโดยตรงต่อสุขภาพของประชาชน ดังนั้นจึงมีหลายงานวิจัยที่มุ่งพัฒนาเครื่องมือสำหรับใช้พยากรณ์ค่ามลพิษทางอากาศล่วงหน้า เพื่อใช้เป็นข้อมูลในการบริหารจัดการและแจ้งเตือนต่อประชาชนที่ได้รับผลกระทบอย่างทันท่วงที

Huang and Kuo (2018) ระบุว่าฝุ่น PM2.5 เป็นปัญหาที่มีผลกระทบต่อระบบทางเดินหายใจ และยังนำไปสู่โรคอื่น ๆ เช่น โรคหอบหืด โรคมะเร็งปอด และโรคหัวใจ เป็นต้น ดังนั้นจึงได้นำเสนอแบบที่เรียกว่า APNet ซึ่งเป็นแบบจำลองที่พัฒนาด้วยกระบวนการ Deep Learning ในงานวิจัยนี้ได้ใช้อัลกอริทึม Convolutional Neural Network: CNN และ Long Short-Term Memory: LSTM ทำงานร่วมกันในการสร้างแบบจำลองสำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า 1 ชม. และศึกษาเชิงเปรียบเทียบประสิทธิภาพแบบจำลอง APNet กับแบบจำลอง Support Vector Machine: SVM, Random Forest: RF, Decision Tree: DT, Multi-Layer Perceptron: MLP, CNN, and LSTM งานวิจัยนี้ใช้ข้อมูลฝุ่น PM2.5 ของเมืองปักกิ่ง ประเทศจีน โดยเลือกใช้ข้อมูลตั้งแต่

ปี 2010 ถึง 2014 ผลที่ได้จากการวิจัยแสดงให้เห็นว่าแบบจำลอง CNN-LSTM (APNet) สามารถพยากรณ์ฝุ่น PM2.5 ล่วงหน้า 1 ชม. ได้ดีที่สุดเมื่อเปรียบเทียบกับแบบจำลอง Baseline แบบจำลอง APNet ให้ค่า MAE, RMSE, Pearson Correlation Coefficient: R, และ Index of Agreement: IA ที่ 14.63, 24.22, 0.95 และ 0.97 ตามลำดับ

งานวิจัยของ Qi, Wang, Song, Hu, Li, and Zhang (2018) ได้นำเสนอแบบจำลองที่เรียกว่า the Deep Air Learning: DAL ซึ่งเป็นแบบจำลองบนพื้นฐานของการเรียนรู้เชิงลึก โดยใช้อัลกอริทึม Autoencoder แบบจำลอง DAL เป็นแบบจำลองที่มีความสามารถในการทำ Interpolation, Prediction และ Feature Analysis กับข้อมูลคุณภาพอากาศ งานวิจัยนี้ได้ใช้กระบวนการ Semi-Supervised Learning ด้วย Neural Network อัลกอริทึม เพื่อทำการหา Feature ของข้อมูลคุณภาพอากาศที่มีความเชื่อมโยงแบบ Spatiotemporal ชุดข้อมูลสำหรับงานวิจัยนี้ได้ใช้ข้อมูลคุณภาพอากาศและข้อมูลอุตุนิยมวิทยาร่วมกัน ซึ่งเป็นข้อมูลของเมืองปักกิ่ง ประเทศจีน จากผลการประเมินและทดสอบประสิทธิภาพแบบจำลอง DAL พบว่ามีประสิทธิภาพต่อการพยากรณ์คุณภาพอากาศล่วงหน้าแบบหนึ่งช่วงและหลายช่วงเวลาดีกว่าแบบจำลองเปรียบเทียบอื่น ๆ เช่น Logistic Regression: LR, Neural Network, Autoencoder, ARIMA และ RNN เป็นต้น

Mao and Lee (2019) นำเสนอและพัฒนาแบบจำลองสำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า 1 วัน ด้วยอัลกอริทึม Convolutional Neural Network: CNN เพราะผู้วิจัยให้ความเห็นว่า CNN สามารถสกัด Sequential Features ของข้อมูลอนุกรมเวลาได้ดี โดยเลือกให้ 24 Lag Time เป็นข้อมูลนำเข้า ข้อมูลที่ใช้เป็นข้อมูลคุณภาพอากาศและข้อมูลอุตุนิยมวิทยารายวัน จำนวน 77 สถานีวัดของประเทศไต้หวัน จากผลการวิจัยแบบจำลอง CNN สามารถพยากรณ์ฝุ่น PM2.5 ล่วงหน้า 1 วัน โดยมีค่าความคลาดเคลื่อน (RMSE) ที่ 7.25 mg/m^3 และผู้วิจัยยังให้ความเห็นว่า Convolution Layer ของ CNN มีประสิทธิภาพที่ดีในการสกัด Sequential Features ของข้อมูลอนุกรมเวลา

Joharestani, Cao, Ni, Bashir, and Talebiesfandarani (2019) ได้ทำการศึกษาตัวแปรพยากรณ์ที่สำคัญซึ่งคำนวณผ่าน Random Forest: RF และ Extreme Gradient Boosting: XGBoost จากนั้นนำตัวแปรพยากรณ์ที่ได้ไปสร้างแบบจำลองสำหรับพยากรณ์ฝุ่น PM2.5 ด้วย Deep Neural Network โดยใช้ข้อมูลคุณภาพอากาศข้อมูลดาวเทียมและข้อมูลแผนที่ของเขตเมือง Tehran ประเทศอิหร่าน สำหรับการดำเนินงานวิจัย ซึ่งงานวิจัยนี้เป็นการพัฒนาและทดสอบประสิทธิภาพแบบจำลองที่ได้นำเสนอ ผลการวิจัยพบว่าแบบจำลองมีประสิทธิภาพต่อการพยากรณ์ฝุ่น PM2.5 ล่วงหน้าดีที่สุดผ่านมาตรวัด R^2 เท่ากับ 0.81, MAE เท่ากับ 9.93 μg/m^3, and RMSE เท่ากับ 13.58 μg/m^3 โดยใช้อัลกอริทึม XGboost ในการทำ Feature Selection และผลการวิจัยยังพบอีกว่าข้อมูลจากดาวเทียมไม่ส่งผลต่อประสิทธิภาพของแบบจำลองสำหรับงานวิจัยนี้ Li, Xie, Ren, Guo, Yang, and Xu (2020) ได้พัฒนาแบบจำลองสำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ผ่านการทำงาน

ร่วมกันระหว่างแบบจำลอง CNN และ Attention-based Network LSTM: AC-LSTM โดยในส่วนของแบบจำลอง CNN จะทำหน้าที่ Feature Extraction เพื่อสกัด Feature ในรูปแบบของ Spatiotemporal จากนั้นใช้ส่วนของ AC-LSTM พยากรณ์ผลต่อไป โดยงานวิจัยนี้ได้ใช้ชุดข้อมูลฝุ่น PM2.5 จากสถานีวัดภาคพื้น 9 สถานีร่วมกับชุดข้อมูลอุตุนิยมวิทยา ซึ่งเป็นข้อมูลของเมือง Taiyuan ประเทศจีน ผลการวิจัยพบว่าแบบจำลองที่นำเสนอให้ประสิทธิภาพต่อการพยากรณ์ฝุ่น PM2.5 ดีที่สุดเมื่อเปรียบเทียบกับแบบจำลอง Baseline เช่น Support Vector Regression: SVR, Random Forest Regression: RFR, และ Multi-Layer Perceptron: MLP เป็นต้น โดยให้ค่าความคลาดเคลื่อน MAE และ RMSE ต่ำที่สุด ที่ 8.98 และ 14.83 ตามลำดับ

Srijiranon and Eiamkanitchat (2018) เป็นงานวิจัยที่ได้ใช้ข้อมูลของประเทศไทย ซึ่งเป็นข้อมูลคุณภาพอากาศ และข้อมูลอุตุนิยมวิทยาทางภาคเหนือ โดยได้เลือกใช้ตัวแปรพยากรณ์ที่เป็นสารมลพิษทั้ง 5 ชนิด และข้อมูลอุตุนิยมวิทยา 6 ตัวแปร งานวิจัยนี้ได้นำเสนอแบบจำลองโครงข่าย Collective Neural Networks System สำหรับคำนวณผลพยากรณ์ และทำการจำแนกผลพยากรณ์เป็น อากาศดี ปลานกลาง และอากาศเสีย ผ่านค่า Max Function โดยผลการพยากรณ์ฝุ่น PM10 ให้ประสิทธิภาพของความถูกต้องในการพยากรณ์เท่ากับ 92.51% ซึ่งดีกว่าแบบจำลองที่เป็น Baseline 0.18%

Wongsathan (2018) ได้ทำการวิจัยเพื่อศึกษาเปรียบเทียบประสิทธิภาพแบบจำลองอนุกรมเวลาผสมที่นำเสนอ Neural Network-ARIMA กับแบบจำลอง ANFIS และแบบจำลองเดี่ยวอื่น ๆ โดยได้ใช้ข้อมูลคุณภาพอากาศของจังหวัดเชียงใหม่ ประเทศไทย เป็นชุดข้อมูลในการดำเนินงานวิจัย จากผลการวิจัยพบว่าแบบจำลองอนุกรมเวลาที่นำเสนอมีประสิทธิภาพดีกว่าแบบจำลองเปรียบเทียบอื่น ๆ

งานวิจัยของ Amnuaylojaroen (2022) ได้นำเสอนแบบจำลอง Multivariate Linear Regression สำหรับพยากรณ์ฝุ่น PM2.5 ในพื้นที่ภาคเหนือของประเทศไทย โดยเลือกใช้ข้อมูลคุณภาพอากาศจากสถานีวัดภาคพื้นที่ตั้งอยู่ที่ จังหวัดเชียงใหม่ จังหวัดลำปาง และจังหวัดน่าน รวมถึงยังได้ใช้ข้อมูลดาวเทียมและข้อมูลอุตุนิยมวิทยาร่วมด้วย ผลจากการวิจัยพบว่าแบบจำลองที่นำเสนอให้ประสิทธิภาพเฉลี่ยของการประเมินจากทั้ง 3 สถานี ผ่านมาตรวัดค่าสัมประสิทธิ์ความเชื่อมั่น R^2 และค่าความคลาดเคลื่อน RMSE เท่ากับ 0.50 และ 56.21 ตามลำดับ

สำหรับงานวิจัยนี้นำเสนอแบบจำลองอนุกรมเวลาผสมระหว่างแบบจำลองอนุกรมเวลาเชิงเส้นและแบบจำลองอนุกรมเวลาไม่เชิงเส้น ซึ่งเป็นแบบจำลองอนุกรมเวลารูปแบบใหม่โดยเป็นการทำงานร่วมกันของแบบจำลองอนุกรมเวลาเชิงเส้น ARIMA และแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS โดยใช้ชุดข้อมูลฝุ่น PM2.5 ของจังหวัดระยอง ประเทศไทย แสดงตารางเปรียบเทียบงานวิจัยที่เกี่ยวข้อง ดังตารางที่ 2.4

ตารางที่ 2.4 สรุปเปรียบเทียบงานวิจัยที่เกี่ยวข้องกับการพยากรณ์มลพิษทางอากาศ

กระบวนการทำงาน	งานวิจัยที่เกี่ยวข้อง								
	ก	ข	ค	จ	ง	ฉ	ช	ซ	*ฌ
ชุดข้อมูลที่นำมาใช้ในงานวิจัย									
ข้อมูลคุณภาพอากาศ	X	X	X	X	X	X	X	X	X
ข้อมูลอุตุนิยมวิทยา		X	X		X	X		X	
ข้อมูลดาวเทียม (AOD)				X				X	
ข้อมูลประเทศไทย						X	X	X	X
เทคนิคและอัลกอริทึมที่ใช้ในงานวิจัย									
Multivariate	X	X	X	X	X	X	X	X	X
Statistics Model							X	X	X
Machine Learning				X		X			X
Deep Learning	X	X	X		X				
Hybrid Model			X				X		X
ลักษณะการพยากรณ์และเป้าหมาย									
Single Step	X	X	X	X	X	X	X	X	X
Multiple Step			X						
PM	X	X		X	X	X	X	X	X
AQI			X						
วัตถุประสงค์ของงานวิจัย									
ประเมินประสิทธิภาพ	X	X	X	X	X	X	X	X	X
เสนอแนวคิดใหม่	X		X		X		X		X
ประยุกต์ใช้กับข้อมูลจริง	X	X	X	X	X	X	X	X	X

หมายเหตุ งานวิจัยที่เกี่ยวข้องประกอบด้วย

ก แทนงานวิจัยของ C. Huang and P. Kuo (2018)

ข แทนงานวิจัยของ Z. Qi, et al (2018)

ค แทนงานวิจัยของ Y. Mao and S. Lee (2019)

ง แทนงานวิจัยของ M. Z. Joharestani, et al (2019)

จ แทนงานวิจัยของ S. Li, et al (2020)

ฉ แทนงานวิจัยของ K. Srijiranon and N. Eiamkanitchat (2018)

ซ แทนงานวิจัยของ R. Wongsathan (2018)

ซ แทนงานวิจัยของ T. Amnuaylojaroen (2022)

*ฌ แทนงานวิจัยของวิทยานิพนธ์ฉบับนี้

บทที่ 3
วิธีดำเนินงานวิจัย

งานวิจัยนี้มีวัตถุประสงค์ที่จะพัฒนาและปรับปรุงแบบจำลองสำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า โดยใช้กระบวนการเรียนรู้ของเครื่อง โดยนำเสนอและพัฒนาแบบจำลองอนุกรมเวลาผสมระหว่างแบบจำลองอนุกรมเวลาเชิงเส้นและแบบจำลองอนุกรมเวลาไม่เชิงเส้นเพื่อให้มีประสิทธิภาพต่อการหาความสัมพันธ์แบบเชิงเส้นและไม่เชิงเส้นของสายข้อมูลอนุกรมเวลา นอกจากนี้เสนอการปรับปรุงแบบจำลองอนุกรมเวลาผสมผ่านกระบวนการ Hyperparameters Tuning ด้วยการทำ Optimization Techniques ได้แก่ Genetic Algorithm: GA และ Particle Swarm Optimization: PSO สุดท้ายดำเนินการประเมินและเปรียบเทียบประสิทธิภาพแบบจำลองอนุกรมเวลาที่นำเสนอกับแบบจำลองอนุกรมเวลาเดี่ยวอื่น ๆ ได้แก่ แบบจำลองอนุกรมเวลาเชิงเส้น Auto Regressive Integrated Moving Average: ARIMA, แบบจำลองอนุกรมเวลาไม่เชิงเส้นในกลุ่มของแบบจำลองโครงข่าย Artificial Neural Network: ANN, Long Short Term Memory: LSTM และ Adaptive Neuro-Fuzzy Inference System: ANFIS

กระบวนการดำเนินงานวิจัยแบ่งเป็น 3 ขั้นตอนหลัก ดังรูปที่ 3.1 ซึ่งเป็นกรอบแนวคิดของการวิจัย คือ 1) ขั้นตอนการเตรียมข้อมูลวิจัย (Data Preparation) 2) ขั้นตอนการพัฒนาแบบจำลอง (Modeling หรือ Model Training) รวมถึงการปรับปรุงแบบจำลอง (Model Tuning) และสุดท้ายเป็นขั้นตอนการประเมินและเปรียบเทียบประสิทธิภาพแบบจำลอง (Model Evaluating หรือ Model Testing)

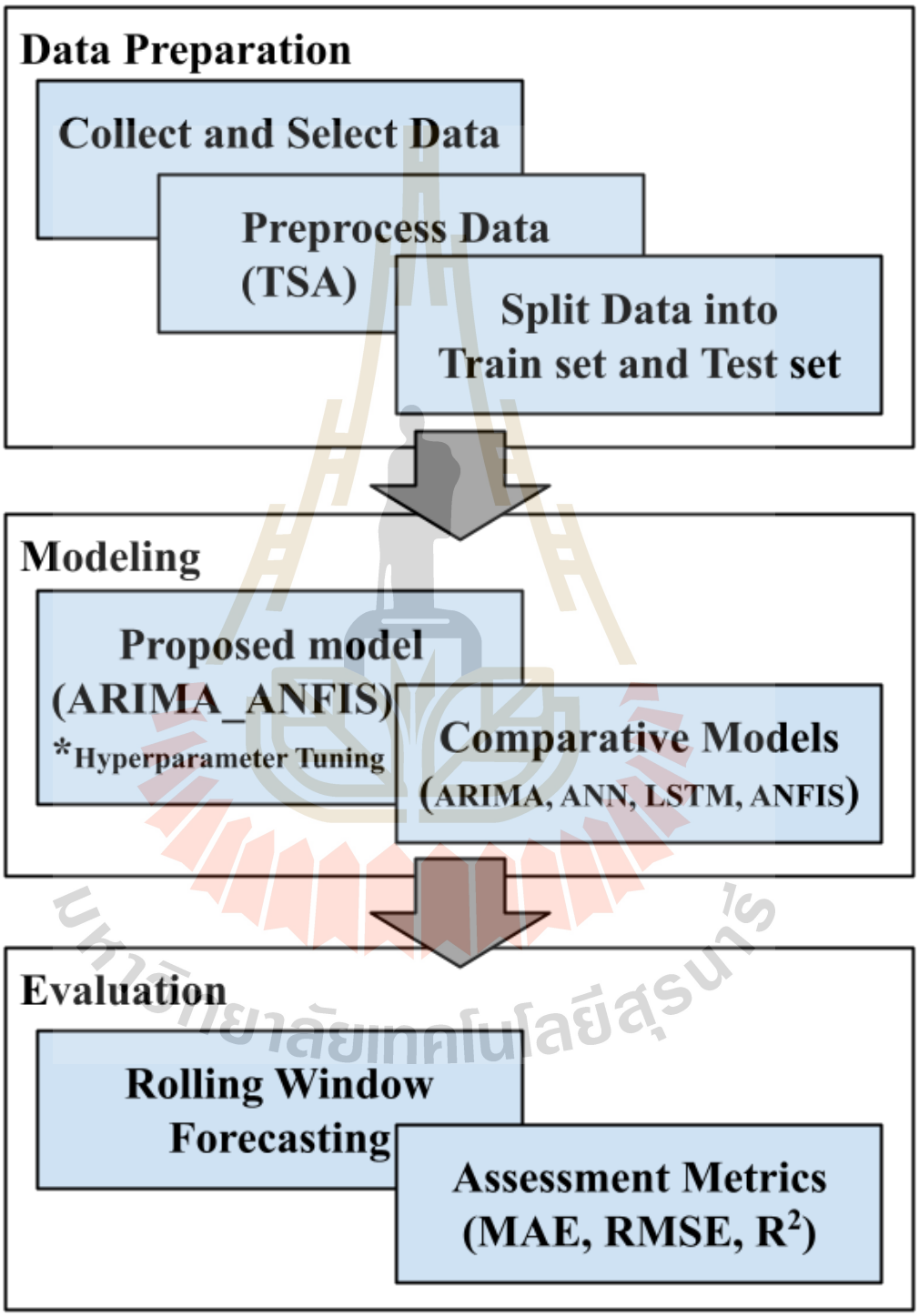
3.1 กระบวนการวิจัย

3.1.1 ขั้นตอนการเตรียมข้อมูลวิจัย

ในขั้นตอนนี้จะเป็นการนำข้อมูลดิบมาประมวลผลเพื่อให้ได้ชุดข้อมูลอนุกรมเวลาเหมาะสมต่อการนำไปทำการวิจัยในขั้นตอนการพัฒนาแบบจำลอง ซึ่งในขั้นตอนการเตรียมข้อมูลวิจัยนี้ จะมีกระบวนการสำคัญกระบวนการหนึ่ง คือ การเติมข้อมูล (Data Imputation) เพื่อจัดการปัญหาของข้อมูลที่ขาดหายไป (Missing Values) ทำให้ได้ชุดข้อมูลอนุกรมเวลาที่มีความสมบูรณ์มากที่สุด

งานวิจัยนี้ได้เลือกใช้เทคนิคการเติมข้อมูลแบบ Temporal and Spatial Average Value: TSA หลักการคือ ถ้าข้อมูลขาดช่วง 1 ช่วงเวลา (1 วัน) จะเติมข้อมูลโดยค่าเฉลี่ยของข้อมูลก่อนหน้า 3 ช่วงเวลา (Temporal Average Value) ดังสมการ (3-1) และถ้าข้อมูลขาดช่วงตั้งแต่ 2

ช่วงเวลาขึ้นไป จะเติมข้อมูลโดยค่าเฉลี่ยจากสถานีวัดคุณภาพอากาศอื่น ๆ (Spatial Average Value) ที่อยู่ในพื้นที่สนใจเดียวกัน ดังสมการ (3-2)



รูปที่ 3.1 แสดงกรอบแนวคิดการดำเนินงานวิจัย

$$y_{temporal} = \frac{1}{d}\sum_{i=1}^{d} y_{t-i} \quad (3\text{-}1)$$

$$y_{spatial} = \frac{1}{s}\sum_{j=1}^{s} y_j \quad (3\text{-}2)$$

โดยที่ $y_{temporal}$ และ $y_{spatial}$ คือข้อมูลฝุ่น PM2.5 ที่ถูกเติมด้วยวิธี Temporal Average Value และ Spatial Average Value ตามลำดับ ส่วน y_{t-i} คือข้อมูลฝุ่น PM2.5 ย้อนหลังตาม Lag Time ที่ i และ y_j คือข้อมูลฝุ่น PM2.5 ที่สถานีวัดคุณภาพอากาศลำดับที่ j

$$y_t = (linear_t + nonlinear_t) + Noise_t \quad (3\text{-}3)$$

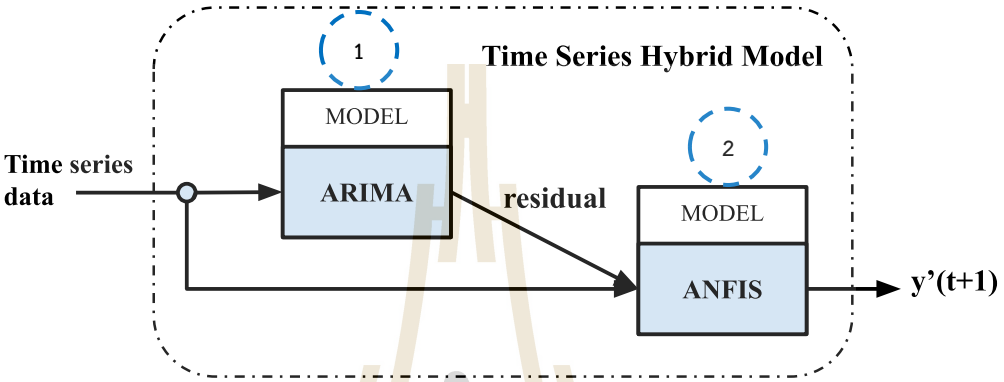
โดยที่ y_t คือ ข้อมูลอนุกรมเวลา ที่เวลา t, $linear_t$ ความสัมพันธ์เชิงเส้นของข้อมูลอนุกรมเวลา ที่เวลา t, $nonlinear_t$ ความสัมพันธ์ไม่เชิงเส้นของข้อมูลอนุกรมเวลา ที่เวลา t, และ $Noise_t$ เป็นค่าสุ่มของข้อมูลอนุกรมเวลา ที่เวลา t (White Noise)

3.1.2 ขั้นตอนการพัฒนาแบบจำลอง

ขั้นตอนการพัฒนาแบบจำลองนี้จะเป็นการพัฒนาแบบจำลองอนุกรมเวลาสำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ผ่านกระบวนการเรียนรู้ของเครื่อง โดยงานวิจัยนี้นำเสนอแบบจำลองอนุกรมเวลาผสมระหว่างแบบจำลองอนุกรมเวลาเชิงเส้น ARIMA ซึ่งเป็นแบบจำลองทางสถิติ ทำงานร่วมกับแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS ซึ่งเป็นแบบจำลองโครงข่ายเชิง Fuzzy ที่เป็นอัลกอริทึมผสมระหว่าง Neural Network: NN กับ Fuzzy Inference System: FIS โดยเรียกแบบจำลองอนุกรมเวลาผสมว่า Auto Regressive Integrated Moving Average Adaptive Neuro-Fuzzy Inference System: ARIMA-ANFIS ซึ่งแบบจำลองอนุกรมเวลาผสม ARIMA-ANFIS ที่นำเสนอจะมีความสามารถหาความสัมพันธ์เชิงเส้นและไม่เชิงเส้นของสายข้อมูลอนุกรมเวลาซึ่งสอดคล้องกับองค์ประกอบเชิงความสัมพันธ์ของสายอนุกรมเวลา (Zhang, 2003) แสดงดังสมการ (3-3) จึงทำให้แบบจำลองอนุกรมเวลาผสมที่นำเสนอมีประสิทธิภาพที่ดีต่อการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า

กระบวนการย่อสำหรับพัฒนาแบบจำลองอนุกรมเวลาผสม ARIMA-ANFIS หลังจากได้ชุดข้อมูลอนุกรมเวลาฝุ่น PM2.5 ที่ผ่านกระบวนการจัดเตรียมชุดข้อมูลแล้ว จะนำชุดข้อมูลเรียนรู้มาทำการวิเคราะห์เพื่อหาค่าพารามิเตอร์ p, d, และ q ซึ่งเป็นพารามิเตอร์หลักที่ต้องกำหนดก่อนการสร้างแบบจำลองของแบบจำลองอนุกรมเวลาเชิงเส้น ARIMA หลังจากนั้นสร้างแบบจำลองอนุกรมเวลาเชิงเส้น ARIMA ตามค่าพารามิเตอร์ p, d, และ q ที่กำหนดพร้อมกับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า จากนั้นเป็นกระบวนการพัฒนาแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS โดยใช้ชุดข้อมูล

เรียนรู้ร่วมกับผลการพยากรณ์ฝุ่น PM2.5 ของแบบจำลองอนุกรมเวลาเชิงเส้น ARIMA โดยหลักการทำงานร่วมกันระหว่างแบบจำลองอนุกรมเวลาเชิงเส้น ARIMA กับแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS แสดงดังรูปที่ 3.2



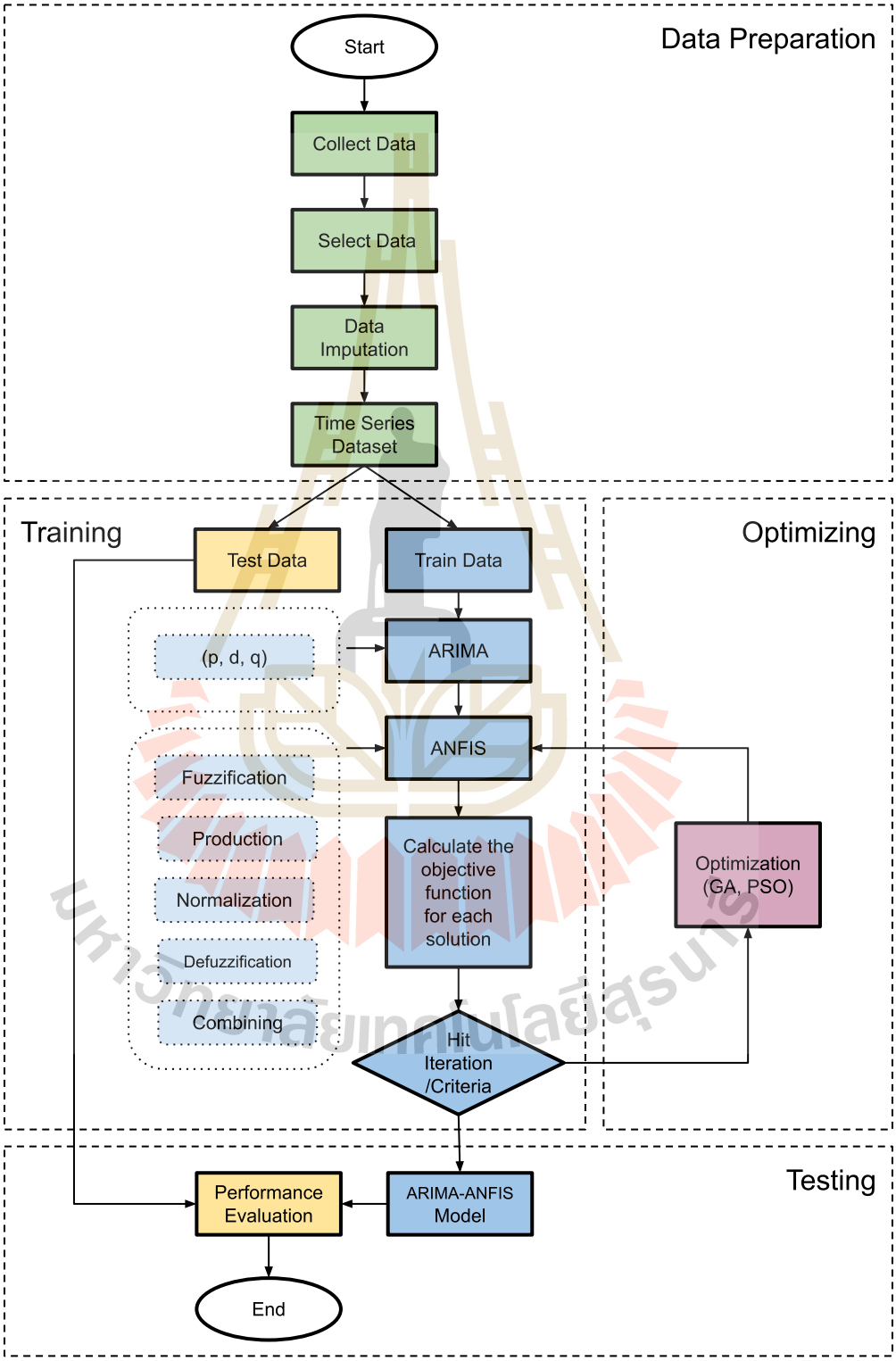
รูปที่ 3.2 แสดงขั้นตอนการทำงานของแบบจำลองอนุกรมเวลาผสม ARIMA-ANFIS

สำหรับการปรับปรุงแบบจำลองอนุกรมเวลาผสม จะทำผ่านกระบวนการ Hyperparameter Tuning โดยใช้ Optimization Techniques วัตถุประสงค์ของการทำในขั้นตอนย่อนี้ 1) ต้องการปรับปรุงประสิทธิภาพแบบจำลองอนุกรมเวลาผสม ARIMA-ANFIS ให้มีประสิทธิภาพสูงสุดสำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า 2) เพื่อป้องกันปัญหา Local Optimum Trap ที่มักจะเกิดกับกลุ่มของแบบจำลองโครงข่ายที่ให้ Gradient Descent: GD สำหรับเรียนรู้ชุดข้อมูลผ่านกระบวนการ Backpropagation งานวิจัยนี้เลือก Optimization Techniques ในกลุ่มของ Metaheuristic Population-based Algorithms ได้แก่ GA และ PSO รูปที่ 3.3 แสดงขั้นตอนการพัฒนาแบบจำลองอนุกรมเวลาผสม ARIMA-ANFIS และการปรับปรุงประสิทธิภาพแบบจำลองด้วย Optimization Techniques

3.1.3 ทดสอบประสิทธิภาพแบบจำลอง

ในขั้นตอนนี้เป็นขั้นตอนสุดท้ายของการวิจัย เพื่อที่จะทดสอบและประเมินประสิทธิภาพแบบจำลองอนุกรมเวลาผสม ARIMA-ANFIS ต่อการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ผู้วิจัยได้เลือกทำการประเมิน และเปรียบเทียบประสิทธิภาพแบบจำลองอนุกรมเวลาผสม ที่นำเสนอกับแบบจำลองอนุกรมเวลาเดี่ยว ได้แก่ แบบจำลองอนุกรมเวลาเชิงเส้น ARIMA, แบบจำลองอนุกรมเวลาไม่เชิงเส้น ANN, แบบจำลองอนุกรมเวลาไม่เชิงเส้น LSTM, และแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS ผ่านมาตรวัดประสิทธิภาพความคลาดเคลื่อน Mean Absolute Error: MAE, Mean Absolute Percentage Error: MAPE, Root Mean Square Error: RMSE, Percentage Root

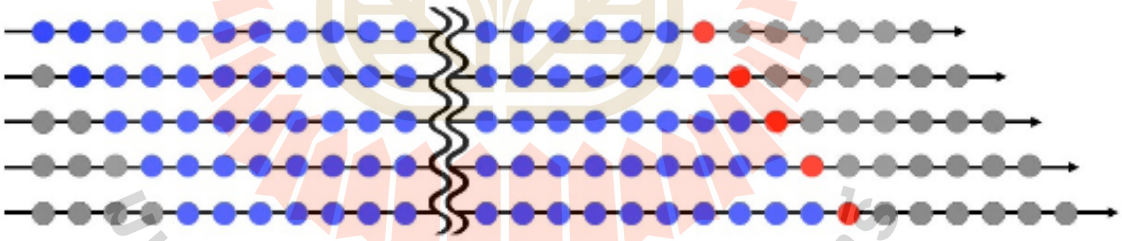
Mean Square Error: %RMSE และค่าสัมประสิทธิ์การตัดสินใจ Coefficient of Determination: R^2 แสดงดังตารางที่ 3.1



รูปที่ 3.3 แสดงขั้นตอนการพัฒนาแบบจำลองอนุกรมเวลาผสม ARIMA-ANFIS

ตารางที่ 3.1 มาตรวัดประสิทธิภาพแบบจำลองอนุกรมเวลา

Matric	Equation	Note
MAE	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$	
MAPE	$\frac{100}{n}\sum_{i=1}^{n}\left\lvert\frac{y_i - \hat{y}_i}{y_i}\right\rvert$	
RMSE	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$	y_i คือค่าข้อมูลจริง ที่ตำแหน่ง i $\hat{y}_i$ คือค่าพยากรณ์ ที่ตำแหน่ง i $\bar{y}$ คือค่าเฉลี่ยของข้อมูลจริง
%RMSE	$\frac{100}{\bar{y}} \times RMSE$	
R^2	$1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$	

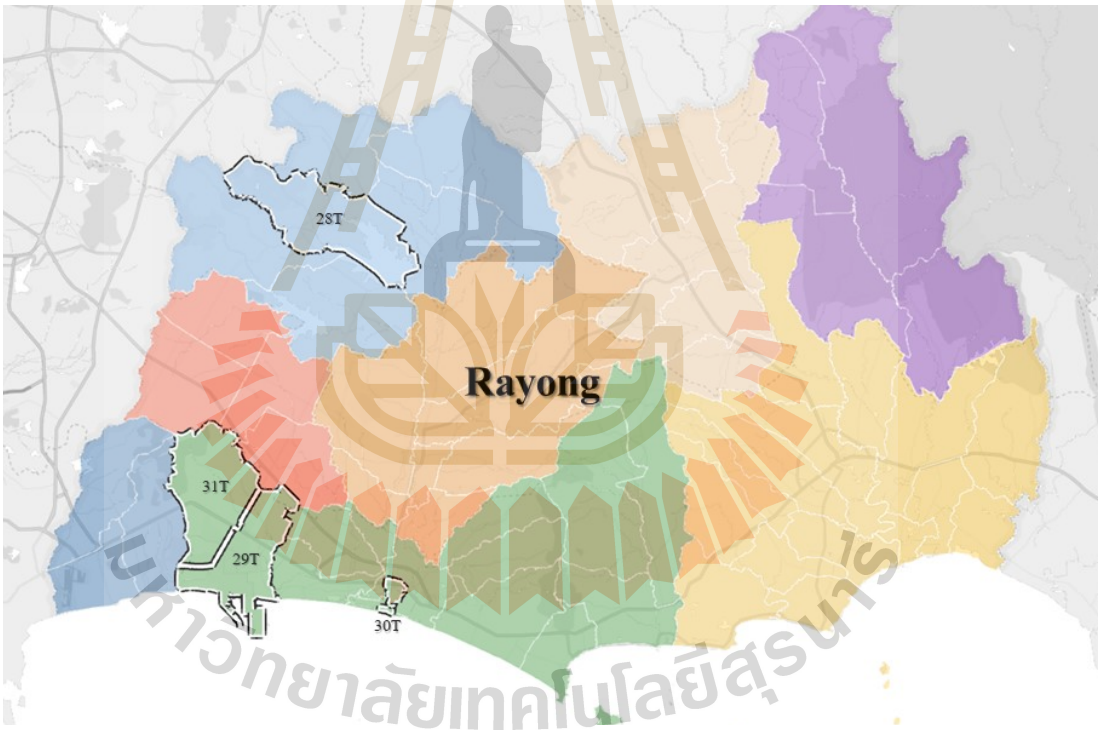


รูปที่ 3.4 แสดงการพยากรณ์แบบ Rolling Window (Chen & Chiu, 2021)

การประเมินและทดสอบประสิทธิภาพของแบบจำลองอนุกรมเวลาที่พัฒนาทั้งอนุกรมเวลาผสมที่นำเสนอและอนุกรมเวลาเดี่ยวสำหรับเปรียบเทียบประสิทธิภาพ ผู้วิจัยได้วางแผนการทดสอบโดยการพยากรณ์ฝุ่น PM2.5 หนึ่งวันล่วงหน้า โดยทดสอบจำนวนทั้งหมดตามชุดข้อมูลทดสอบรวม 365 วัน ซึ่งรูปแบบการพยากรณ์ของแบบจำลองอนุกรมเวลาจะทดสอบด้วยเทคนิค Rolling Window แสดงดังรูปที่ 3.4 ตามรูปวงกลมสีน้ำเงินแทนข้อมูลนำเข้าแบบจำลองอนุกรมเวลาและวงกลมสีแดงแทนผลการพยากรณ์

3.2 ข้อมูลที่ใช้ในงานวิจัย

งานวิจัยนี้ได้เลือกใช้ข้อมูลอนุกรมเวลาฝุ่น PM2.5 จากสถานีวัดคุณภาพอากาศภาคพื้นของประเทศไทย ซึ่งข้อมูลมีการรวบรวมจัดเตรียมและเปิดเผยโดยกรมควบคุมมลพิษ กระทรวงทรัพยากรธรรมชาติและสิ่งแวดล้อม ประเทศไทย งานวิจัยนี้สนใจที่ชุดข้อมูลอนุกรมเวลาฝุ่น PM2.5 ในพื้นที่เขตเมืองอุตสาหกรรมเพราะเป็นเขตพื้นที่ที่ทางรัฐบาล ฯ ให้ความสำคัญและมีการเฝ้าระวังเกี่ยวกับปัญหามลพิษเป็นพิเศษ ผู้วิจัยจึงได้เลือกชุดข้อมูลอนุกรมเวลาฝุ่น PM2.5 จากสถานีวัดคุณภาพอากาศภาคพื้นของจังหวัดระยอง มีจำนวนทั้งหมด 4 สถานี โดยมีรหัสประจำสถานี ดังนี้ 28T, 29T, 30T และ 31T ซึ่งตั้งอยู่ที่ สำนักงานสาธารณสุขอำเภอปลวกแดง, โรงพยาบาลส่งเสริมสุขภาพตำบลมาบตาพุด, สำนักงานเกษตรจังหวัดระยอง, และศูนย์วิจัยพืชไร่ระยอง ตามลำดับแสดงแผนที่ของแต่ละตั้งสถานี ดังรูปที่ 3.5



รูปที่ 3.5 แสดงแผนที่ตั้งสถานีวัดคุณภาพอากาศภาคพื้นในเขตจังหวัดระยอง

จากชุดข้อมูลดิบของอนุกรมเวลาฝุ่น PM2.5 ซึ่งเป็นข้อมูลบันทึกเฉลี่ยรายวันของฝุ่น PM2.5 ตั้งแต่ ปี ค.ศ. 2011 - 2022 โดยผู้วิจัยเลือกข้อมูลอนุกรมเวลาฝุ่น PM2.5 เฉพาะช่วงเวลาปี ค.ศ. 2017 - 2021 รวมทั้งสิ้นเป็นเวลา 5 ปี เพราะเป็นช่วงเวลาที่ชุดข้อมูลดิบมีความสมบูรณ์มากที่สุด สำหรับงานวิจัยนี้ได้แบ่งชุดข้อมูลอนุกรมเวลาฝุ่น PM2.5 ออกเป็น 2 ส่วน คือ ส่วนของชุดข้อมูล

สำหรับเรียนรู้ โดยเลือกช่วงเวลาตั้งแต่ ปี ค.ศ. 2017 - 2020 และส่วนที่สองเป็นชุดข้อมูลสำหรับทดสอบ คือ ข้อมูลฝุ่น PM2.5 ในปี ค.ศ. 2021 โดยคิดเป็นอัตราส่วนระหว่างข้อมูลเรียนรู้และข้อมูลทดสอบเท่ากับ 4 ต่อ 1 รูปที่ 3.6 แสดงข้อมูลดิบของชุดข้อมูลอนุกรมเวลาฝุ่น PM2.5 สำหรับงานวิจัยนี้

	A	B	C	D	E	F	G	H	I	J
1	Date	02T	05T	10T	11T	12T	59T	61T	03T	50T
2	1/1/2021	27	20	22	25	22	20	25	24	25
3	1/2/2021	32	25	26	27	27	23	26	29	31
4	1/3/2021	46	37	33	41	40	38	29	44	44
5	1/4/2021	39	31	32	36	38	36	28	46	41
6	1/5/2021	50	31	31	32	44	28	24	67	41
7	1/6/2021	57	32	36	36	39	32	25	85	36
8	1/7/2021	37	26	30	30	31	25	23	62	32
9	1/8/2021	26	21	24	23	24	21	20	26	25
10	1/9/2021	23	19	13	20	22	18	18	19	20
11	1/10/2021	30	27	20	25	26	25	22	35	27
12	1/11/2021	35	27	20	31	29	28	23	37	32
13	1/12/2021	33	26	19	29	28	26	22	36	31
14	1/13/2021	54	48	32	49	46	47	31	63	51
15	1/14/2021	76	71	40	68	64	69	42	99	71
16	1/15/2021	83	83	56	79	71	74	49	112	79

รูปที่ 3.6 แสดงตัวอย่างของชุดข้อมูลดิบของอนุกรมเวลาฝุ่น PM2.5

3.3 เครื่องมือที่ใช้ในงานวิจัย

เครื่องมือที่ใช้ในงานวิจัยนี้ ประกอบด้วยฮาร์ดแวร์และซอฟต์แวร์ ดังนี้

3.3.1 เครื่องคอมพิวเตอร์แม่ข่าย โดยมีรายละเอียดดังนี้

(1) หน่วยประมวลผลกลาง: Intel(R) Xeon(R) Silver 4314 CPU @ 2.40GHz (x2)

(2) หน่วยความจำหลัก: 64 GBytes

(3) หน่วยจัดเก็บข้อมูล: 1 TBytes

3.2.1 ระบบปฏิบัติการและโปรแกรมประยุกต์ ประกอบด้วย

(1) ระบบปฏิบัติการ: CentOS 7.9.2009

(2) โปรแกรมภาษาที่ใช้พัฒนา: Python 3.10.5 และ Matlab R2022a

(3) เครื่องมือร่วมในการพัฒนา: Kernel-based Virtual Machine, Jupyter Notebook, และ Visual Studio Code

บทที่ 4
การทดสอบและอภิปรายผล

งานวิจัยนี้เป็นการพัฒนาแบบจำลองอนุกรมเวลาผสมระหว่างแบบจำลองอนุกรมเวลาเชิงเส้นกับแบบจำลองอนุกรมเวลาไม่เชิงเส้นเพื่อให้เกิดประสิทธิภาพสูงสุดสำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า โดยเลือกพัฒนาแบบจำลองอนุกรมเวลาเชิงเส้นด้วยแบบจำลองทางสถิติ Autoregressive Integrated Moving Average: ARIMA ส่วนแบบจำลองอนุกรมเวลาไม่เชิงเส้น งานวิจัยนี้นำเสนอแบบจำลองในกลุ่มของแบบจำลองโครงข่ายประสาทเทียม 3 รูปแบบ ได้แก่ 1) แบบจำลองโครงข่ายประสาทเทียมมาตรฐาน Artificial Neural Network: ANN 2) แบบจำลองโครงข่ายประสาทเทียมเรียนรู้เชิงลึก Long Short-Term Memory: LSTM และ 3) แบบจำลองโครงข่ายประสาทเทียมเชิงฟัซซี Adaptive Neuro-Fuzzy Inference System: ANFIS

การทดลองวิจัยขั้นที่หนึ่ง เป็นการพัฒนาแบบจำลองอนุกรมเวลาเดี่ยว (Stand Alone) ที่เป็นอนุกรมเวลาเชิงเส้นและไม่เชิงเส้น และทดสอบเปรียบเทียบเพื่อหาค่าพารามิเตอร์และโครงสร้างโครงข่ายที่เหมาะสมที่สุดของแบบจำลองอนุกรมเวลาเดี่ยวที่ศึกษาทั้งหมด โดยใช้กระบวนการ Approximate Search แบบ Trial and Error การทดลองในขั้นที่สองของงานวิจัย คือการพัฒนาแบบจำลองอนุกรมเวลาผสมระหว่างแบบจำลองอนุกรมเวลาเชิงเส้นกับแบบจำลองอนุกรมเวลาไม่เชิงเส้น โดยเลือกแบบจำลองอนุกรมเวลาที่ให้ประสิทธิภาพโดยรวมสูงที่สุดของแบบจำลองอนุกรมเวลาเชิงเส้นและไม่เชิงเส้นจากการทดลองในขั้นที่หนึ่ง หลังจากนั้นเป็นการทดลองปรับปรุงประสิทธิภาพแบบจำลองอนุกรมเวลาผสมโดยใช้เทคนิค Global Optimization ที่เลือกใช้อัลกอริทึมในกลุ่มของ Population based Metaheuristics ได้แก่ Genetic Algorithm: GA และ Particle Swarm Optimization: PSO เพื่อช่วยปรับค่าพารามิเตอร์ของแบบจำลองอนุกรมเวลาผสมให้ได้ค่าที่เหมาะสมที่สุด ซึ่งจะส่งผลให้แบบจำลองอนุกรมเวลาผสมมีความแม่นยำสูงสุดต่อการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า

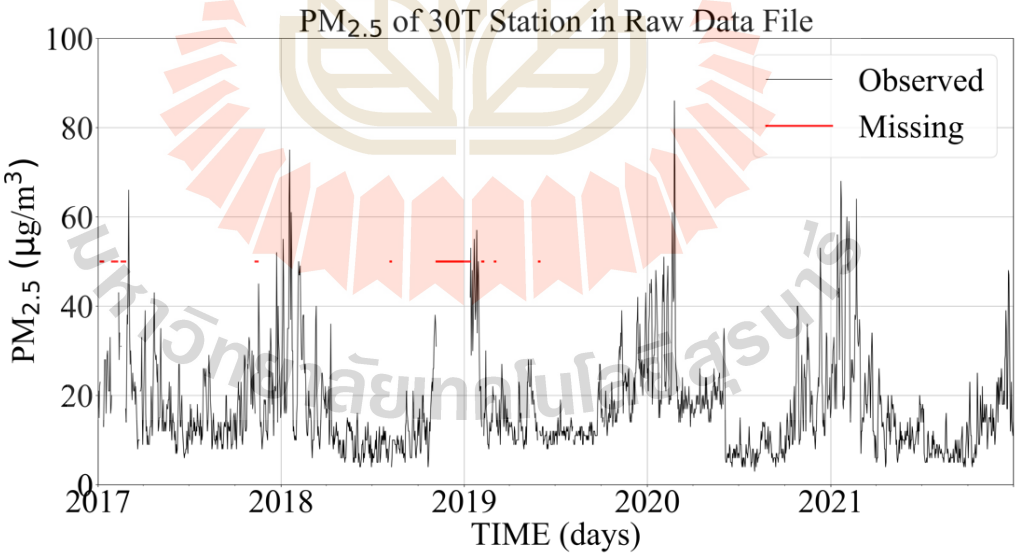
4.1 ข้อมูลสำหรับการวิจัย

งานวิจัยนี้เลือกใช้ข้อมูลบันทึกค่าเฉลี่ยของฝุ่น PM2.5 แบบรายวัน ของสถานีวัดคุณภาพอากาศภาคพื้น ตั้งอยู่ที่ ตำบลท่าประดู่ อำเภอเมือง จังหวัดระยอง รหัสสถานี 30T ซึ่งเป็นชุดข้อมูลเป็นของกรมควบคุมมลพิษ กระทรวงทรัพยากรธรรมชาติและสิ่งแวดล้อม ประเทศไทย โดยดำเนินการรวบรวมข้อมูลตั้งแต่ปี ค.ศ. 2011 ถึง ค.ศ. 2022 และเผยแพร่แบบสาธารณะบนเว็บไซต์

ของกรมควบคุมมลพิษ กระทรวงทรัพยากรธรรมชาติและสิ่งแวดล้อม ประเทศไทย สำหรับงานวิจัยนี้เลือกใช้ข้อมูลอนุกรมเวลาฝุ่น PM2.5 ช่วงเวลาตั้งแต่ปี ค.ศ. 2017 ถึง ค.ศ. 2021 รวมทั้งหมด 5 ปี โดยแบ่งข้อมูลจำนวน 4 ปี ได้แก่ ปี ค.ศ. 2017 ถึง ค.ศ. 2020 เป็นข้อมูลสำหรับชุดข้อมูลเรียนรู้ และข้อมูลส่วนที่เหลือจำนวน 1 ปี คือ ปี ค.ศ. 2021 นำเป็นข้อมูลสำหรับชุดข้อมูลทดสอบ โดยคิดเป็นอัตราส่วนร้อยละ 80 สำหรับชุดข้อมูลเรียนรู้และร้อยละ 20 สำหรับชุดข้อมูลทดสอบ

4.1.1 การเตรียมชุดข้อมูลสำหรับการวิจัย

เนื่องจากความสมบูรณ์หรือความต่อเนื่องของข้อมูลอนุกรมเวลาเป็นสิ่งสำคัญอย่างยิ่งเพราะมีผลต่อประสิทธิภาพของแบบจำลองอนุกรมเวลา โดยส่วนใหญ่ชุดข้อมูลอนุกรมเวลาที่เก็บจากข้อมูลจริง (Real-world Data) มักจะมีข้อมูลไม่สมบูรณ์หรือขาดความต่อเนื่องเป็นผลจากปัญหาของการเก็บบันทึกข้อมูล เช่น อุปกรณ์ตรวจวัดเสีย หรือมีการปิดระบบสำหรับซ่อมบำรุงอุปกรณ์ เป็นต้น เรียกข้อมูลที่ขาดช่วงไปว่า Missing Value หากเพิกเฉยต่อค่า Missing Value ในข้อมูลอนุกรมเวลาอาจจะทำให้ความสัมพันธ์เชิงเวลาของข้อมูลผิดเพี้ยนไปได้ สำหรับข้อมูลอนุกรมเวลาฝุ่น PM2.5 ที่ใช้กับงานวิจัยนี้ก็เช่นเดียวกัน ชุดข้อมูลที่เลือกใช้มีจำนวนทั้งหมด 1,826 ข้อมูล และมีข้อมูลที่เป็น Missing Value จำนวน 117 ข้อมูล คิดเป็นร้อยละ 6.40 ของข้อมูลทั้งหมด รูปที่ 4.1 แสดงกราฟข้อมูลดิบ PM2.5 ของสถานีวัดคุณภาพอากาศภาคพื้น รหัส 30T โดยเส้นสีแดงแสดงข้อมูลในช่วงที่เป็น Missing Values



รูปที่ 4.1 แสดงกราฟข้อมูล PM2.5 ที่สถานีวัด 30T โดยเส้นสีแดงแสดงถึง Missing Values

ดังที่กล่าวไปข้างต้น ปัญหาของ Missing Value ที่อยู่ในข้อมูลอนุกรมเวลาเป็นสิ่งสำคัญและจำเป็นอย่างยิ่งที่ต้องจัดการเพื่อให้ได้ข้อมูลอนุกรมเวลาที่มีความสมบูรณ์ที่สุด งานวิจัยนี้ได้

ศึกษาเปรียบเทียบการเติมข้อมูลโดยวิธี Mean, Median, Most-frequency, Last Observed Carry Forward: LOCF, และ Temporal and Spatial Average Value: TSA เพื่อให้ได้ชุดข้อมูลอนุกรมเวลาฝุ่น PM2.5 ที่มีสายข้อมูลที่สมบูรณ์สำหรับดำเนินการวิจัยขั้นต่อไป

การทดสอบ ประเมิน และเปรียบเทียบประสิทธิภาพวิธีการเติมข้อมูล โดยเลือกข้อมูลอนุกรมเวลาฝุ่น PM2.5 ในปี ค.ศ. 2021 เป็นชุดข้อมูลทดสอบ เพราะมีข้อมูลครบตลอดทั้งปี โดยนำชุดข้อมูลทดสอบทำการจำลองให้เกิด Missing Value ด้วยวิธีการสุ่มเลือกให้มีค่า Missing Value คิดเป็นร้อยละ 20 ของข้อมูลทั้งหมด ข้อมูลอนุกรมเวลาฝุ่น PM2.5 ปี ค.ศ. 2021 มีจำนวนข้อมูลทั้งหมด 365 ข้อมูล ดังนั้นจะมีข้อมูลที่เป็น Missing Value จำนวน 73 ข้อมูล แสดงการเปรียบเทียบระหว่างข้อมูลอนุกรมเวลาฝุ่น PM2.5 ของชุดข้อมูลทดสอบที่สมบูรณ์กับข้อมูลอนุกรมเวลาฝุ่น PM2.5 ที่จำลองให้มี Missing Value ดังรูปที่ 4.2 หลังการนั้นเลือกเติมข้อมูลด้วยวิธีการ Mean, Median, Most-frequency, LOCF, และ TSA ตามลำดับ สำหรับวิธีการเติมข้อมูลด้วย TSA กำหนดค่าตัวแปร d และค่าตัวแปร s เท่ากับ 3 ซึ่งเป็นการกำหนดจำนวนวันย้อนหลังและจำนวนสถานีวัดคุณภาพอากาศภาคพื้นที่ใช้ร่วมพิจารณาสำหรับเติมข้อมูลที่ขาดหายไป และประเมินประสิทธิภาพโดยการวัดความถูกต้องของข้อมูลที่เติมผ่านค่าความคลาดเคลื่อน Mean Square Error: MSE

รูปที่ 4.2 แสดงเปรียบเทียบข้อมูล PM2.5 ของปี ค.ศ. 2021 ระหว่างข้อมูลที่สมบูรณ์กับข้อมูลที่จำลองค่า Missing Value

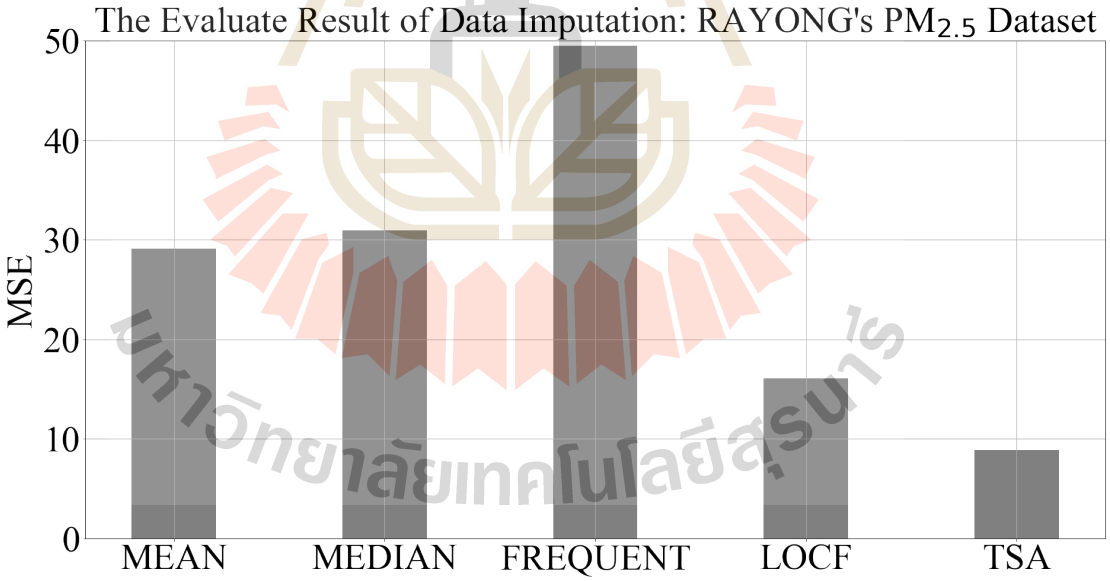
4.1.2 ผลการประเมินประสิทธิภาพของวิธีเติมข้อมูล

ผลการประเมินประสิทธิภาพของวิธีเติมข้อมูล จากการทดลองพบว่าวิธีเติมข้อมูลด้วยเทคนิค TSA ให้ประสิทธิภาพดีที่สุด โดยมีค่าความคลาดเคลื่อน MSE ที่ 8.91 และการเติมข้อมูล

ด้วยเทคนิค LOCF ให้ประสิทธิภาพที่ดีรองลงมา โดยมีค่าความคลาดเคลื่อน MSE ที่ 16.09 ส่วนการเติมข้อมูลด้วยเทคนิค Mean กับ Median ให้ประสิทธิภาพใกล้เคียงกันที่ค่า MSE เท่ากับ 29.07 และ 30.96 ตามลำดับ ส่วนการเติมข้อมูลด้วยเทคนิค Most-frequent ให้ประสิทธิภาพต่ำที่สุด ซึ่งมีค่าความคลาดเคลื่อน MSE เท่ากับ 49.46 แสดงดังตารางที่ 4.1 ส่วนรูปที่ 4.3 แสดงกราฟแท่งเปรียบเทียบค่าความคลาดเคลื่อนของวิธีการเติมข้อมูลด้วยเทคนิคต่าง ๆ

ตารางที่ 4.1 ค่าความคลาดเคลื่อน MSE ของแต่ละเทคนิคการเติมข้อมูล

ลำดับที่	เทคนิคการเติมข้อมูล	MSE
1	Mean	29.0737
2	Median	30.9616
3	Most-Frequency	49.4684
4	Last Observed Carry Forward	16.0958
5	Temporal and Spatial Average Value	**8.9133**



รูปที่ 4.3 กราฟแท่งแสดงค่าความคลาดเคลื่อน MSE ของการเติมข้อมูลด้วยเทคนิคต่าง ๆ

ผู้วิจัยมีความเห็นว่า เนื่องจากลักษณะการเคลื่อนไหวหรือการเปลี่ยนแปลงของข้อมูลอนุกรมเวลาฝุ่น PM2.5 มีค่าค่อนข้างต่ำ โดยส่วนใหญ่มีการเปลี่ยนแปลงค่าข้อมูลอย่างช้า ๆ ทำให้ค่าความสัมพันธ์ระหว่างข้อมูลที่อยู่ติดกันมีค่าสูง จึงทำให้วิธีการเติมข้อมูลที่อิงกับข้อมูลในช่วงเวลาใกล้เคียงกันกับข้อมูลที่เป็น Missing Value อย่างเทคนิค LOCF และ TSA มีประสิทธิภาพสูงกว่าการ

เติมข้อมูลด้วยเทคนิคเชิงสถิติอย่าง Mean, Median, และ Most-Frequency รูปที่ 4.4 แสดงกราฟข้อมูลอนุกรมเวลา PM2.5 แก้ปัญหา Missing Value ผ่านวิธีเติมข้อมูลด้วยเทคนิคต่าง ๆ ตามที่ได้ทดสอบไปข้างต้น

4.2 ผลประเมินประสิทธิภาพการทดลองขั้นที่หนึ่ง - พัฒนาแบบจำลองอนุกรมเวลาเดี่ยว

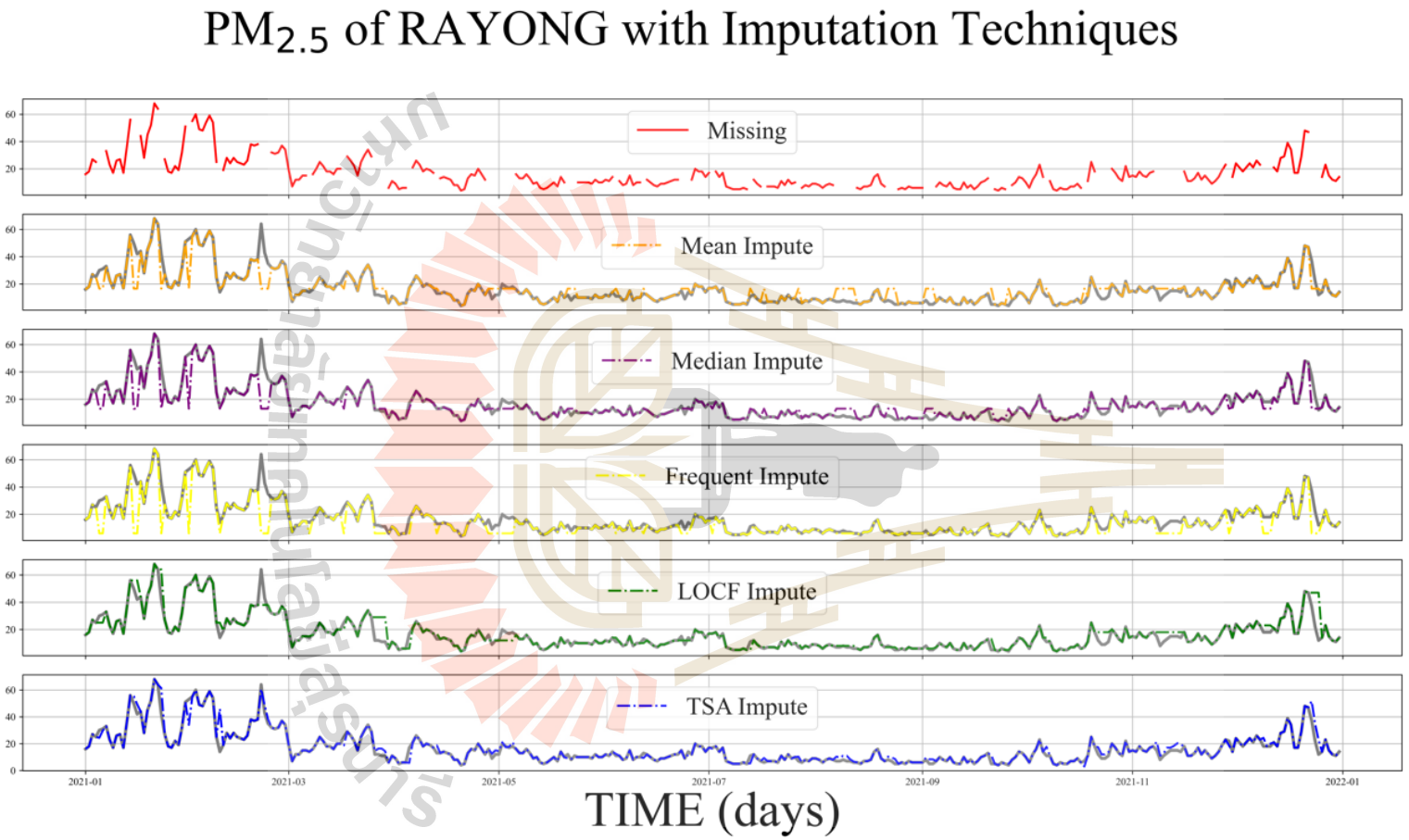
การทดลองขั้นแรกของงานวิจัยเป็นการพัฒนาแบบจำลองอนุกรมเวลาแบบเชิงเส้นและไม่เชิงเส้นด้วยแบบจำลองเดี่ยวสำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า โดยเลือกพัฒนาแบบจำลองอนุกรมเวลาเชิงเส้นที่เป็นแบบจำลองเชิงสถิติ ARIMA และแบบจำลองอนุกรมเวลาไม่เชิงเส้นในกลุ่มของแบบจำลองโครงข่ายประสาทเทียม 3 ประเภท ได้แก่ 1) แบบจำลองโครงข่ายมาตราฐาน ANN 2) แบบจำลองโครงข่ายการเรียนรู้เชิงลึก LSTM และ 3) แบบจำลองโครงข่ายเชิง Fuzzy ANFIS

4.2.1 ผลประเมินประสิทธิภาพแบบจำลองอนุกรมเวลา ARIMA

แบบจำลองอนุกรมเวลา ARIMA เป็นแบบจำลองในกลุ่มของ Parametric Model ดังนั้น จะต้องมีการกำหนดค่าพารามิเตอร์ที่สำคัญของแบบจำลองก่อนการสร้างแบบจำลอง ซึ่งค่าพารามิเตอร์ที่สำคัญของแบบจำลองอนุกรมเวลา ARIMA ได้แก่ ค่า p, d, และ q โดยที่ค่าพารามิเตอร์ p จะเป็นตัวกำหนดเทอมของสมการ AR, ค่าพารามิเตอร์ q จะเป็นตัวกำหนดเทอมของสมการ MA, และสุดท้ายค่าพารามิเตอร์ d เป็นตัวกำหนด Order สำหรับการทำ Data Differencing เพื่อให้ชุดข้อมูลอนุกรมเวลาอยู่ในรูปแบบของ Stationary กล่าวคือ สายข้อมูลอนุกรมเวลาจะไม่มีค่าของแนวโน้มชัดเจน และค่าความแปรปรวนจะค่อนข้างคงที่ตลอดสายข้อมูล

ปกติแล้วการหาค่าพารามิเตอร์ p, d, และ q ของแบบจำลองอนุกรมเวลา ARIMA จะต้องนำชุดข้อมูลอนุกรมเวลามาทำการวิเคราะห์ด้วยกราฟ Auto Correlation Function: ACF สำหรับพิจารณาหาค่าพารามิเตอร์ p ที่เหมาะสม และหาค่าพารามิเตอร์ q ที่เหมาะด้วยการวิเคราะห์กราฟ Partial Auto Correlation Function: PACF สุดท้ายค่าพารามิเตอร์ d จะต้องใช้เครื่องมือทางสถิติ เช่น Augmented Dickey–Fuller test เพื่อทดสอบความเป็น Stationary ของข้อมูลอนุกรมเวลา จากนั้นพิจารณากำหนดค่าพารามิเตอร์ d เพื่อเป็นการระบุ Order สำหรับการทำ Data Differencing

สำหรับงานวิจัยนี้ ได้เลือกใช้เครื่องมือสำหรับการหาค่าพารามิเตอร์ของแบบจำลองอนุกรมเวลา ARIMA ด้วยวิธีอัตโนมัติจากการใช้ฟังก์ชัน “auto_arima” ของไลบรารี “pmdarima” ในภาษา Python โดยที่ฟังก์ชัน “auto_arima” จะใช้วิธีการปรับค่าพารามิเตอร์ p, d, และ q ด้วย Greedy Algorithm และมี Cost Function คือค่า Akaike Information Criterion (AIC) สำหรับเป็นเกณฑ์ในการเลือกชุดพารามิเตอร์ p, d, และ q ที่เหมาะสมที่สุด



รูปที่ 4.4 กราฟเปรียบเทียบข้อมูลอนุกรมเวลาฝุ่น PM2.5 ที่ผ่านการเติมข้อมูลด้วยเทคนิค Mean, Median, Most-frequent, LOCF, และ TSA

จากผลการทดลองพบว่าค่าพารามิเตอร์ที่เหมาะสมที่สุดสำหรับแบบจำลองอนุกรมเวลา ARIMA ที่ใช้ข้อมูลฝุ่น PM2.5 ของสถานีวัดคุณภาพอากาศ รหัส 30T โดยมีค่าพารามิเตอร์ p, d, และ q คือ 2, 0, และ 2 ตามลำดับ และมีค่า Coefficient, Intercept, และ Sigma ดังรูปที่ 4.5 แสดงผลการค้นหาค่าพารามิเตอร์ของฟังก์ชัน “auto_arima” แสดงดังตารางที่ 4.2

ผู้วิจัยได้นำข้อมูลฝุ่น PM2.5 ของสถานีวัดคุณภาพอากาศรหัส 30T วิเคราะห์พบเบื้องต้นพบว่าค่าของฝุ่น PM2.5 จะมีค่าสูงในช่วงหน้าแล้งโดยจะมีค่าที่เริ่มสูงขึ้นตั้งแต่เดือนพฤศจิกายนและมีค่าสูงที่สุดในช่วงปลายเดือนมกราคมและเริ่มลดลงเมื่อเข้าสู่ฤดูฝน แสดงดังรูปที่ 4.6 ส่วนรูปที่ 4.7 แสดงการ Decompose ข้อมูลอนุกรมเวลา ของข้อมูลฝุ่น PM2.5 จากรูปพบว่าข้อมูลไม่มีแนวโน้ม แต่จะมีฤดูกาลที่ในช่วงหน้าแล้งจะมีค่าฝุ่น PM2.5 สูงในทุก ๆ ปี

ตารางที่ 4.2 แสดงผลการค้นหาค่าพารามิเตอร์ p, d, และ q ของแบบจำลองอนุกรมเวลา ARIMA ผ่านฟังก์ชัน “auto_arima”

Performing Stepwise Search to Minimize AIC			
ARIMA(p, d, q)	**Intercept**	**AIC**	**Time (Sec)**
2, 0, 2	**/**	**9302.419**	**0.91**
0, 0, 0	/	11222.283	0.02
1, 0, 0	/	9378.323	0.11
0 ,0 , 1	/	10118.006	0.16
0, 0, 0		13110.343	0.01
1, 0, 2	/	9342.923	0.28
2, 0, 1	/	9361.376	0.44
3, 0, 2	/	9303.619	1.16
2, 0, 3	/	9303.512	1.28
1, 0, 1	/	9369.685	0.17
1, 0, 3	/	9333.011	0.46
3, 0, 1	/	9307.775	1.11
3, 0, 3	/	9306.374	1.07
2, 0, 2		9310.245	0.26

การทดสอบความเป็น Stationary ของชุดข้อมูลอนุกรมเวลาฝุ่น PM2.5 ผู้วิจัยใช้เครื่องมือ “adfuller” จากไลบรารี “statsmodels.tsa.stattools” ของภาษา Python โดยได้

ผลลัพธ์จากการทดสอบ ดังนี้ ค่า ADF เท่ากับ -3.5284, P-Value เท่ากับ 0.0072, และ Critical Values ที่ 1%, 5%, และ 10% มีค่าเท่ากับ -3.4339, -2.8631, และ -2.5676 ตามลำดับ โดยแสดงค่าดังตารางที่ 4.3 จากผลสรุปของการทดสอบ Augmented Dickey Fuller Test พบว่าข้อมูลอนุกรมเวลาฝุ่น PM2.5 ของสถานีวัดคุณภาพอากาศรหัส 30T มีความเป็น Stationary ซึ่งสอดคล้องกับการกำหนดค่าพารามิเตอร์ของแบบจำลองอนุกรมเวลา ARIMA (2, 0, 2) ที่ได้จากการหาพารามิเตอร์ด้วยฟังก์ชัน “auto_arima” คือ ได้ค่า d = 0 นั่นแสดงว่าข้อมูลมีความเป็น Stationary จึงไม่ต้องทำกระบวนการ Data Differencing

SARIMAX Results

Dep. Variable:	PM2dot5	No. Observations:	1461
Model:	ARIMA(2, 0, 2)	Log Likelihood	-4645.210
Date:	Sat, 28 Jan 2023	AIC	9302.419
Time:	23:51:32	BIC	9334.141
Sample:	01-01-2017	HQIC	9314.252
	- 12-31-2020		
Covariance Type:	opg		

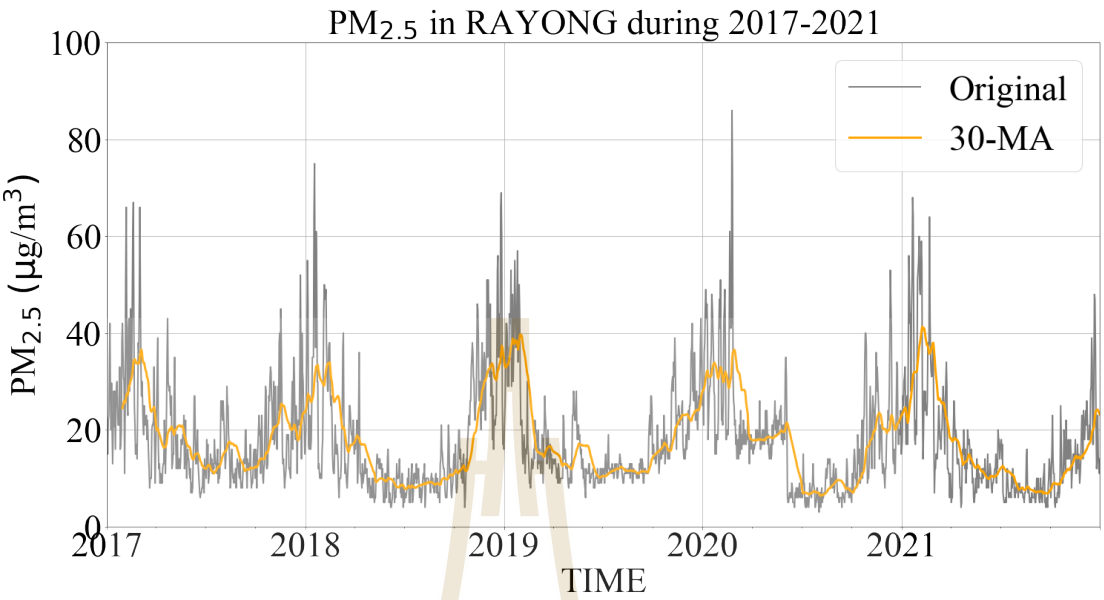
	coef	std err	z	P>|z|	[0.025	0.975]
const	18.5561	3.454	5.373	0.000	11.787	25.325
ar.L1	1.5535	0.034	45.987	0.000	1.487	1.620
ar.L2	-0.5594	0.032	-17.364	0.000	-0.623	-0.496
ma.L1	-0.6695	0.036	-18.408	0.000	-0.741	-0.598
ma.L2	-0.2312	0.022	-10.569	0.000	-0.274	-0.188
sigma2	33.7830	0.663	50.937	0.000	32.483	35.083

Ljung-Box (L1) (Q):	0.01	Jarque-Bera (JB):	2767.57
Prob(Q):	0.92	Prob(JB):	0.00
Heteroskedasticity (H):	0.62	Skew:	0.96
Prob(H) (two-sided):	0.00	Kurtosis:	9.46

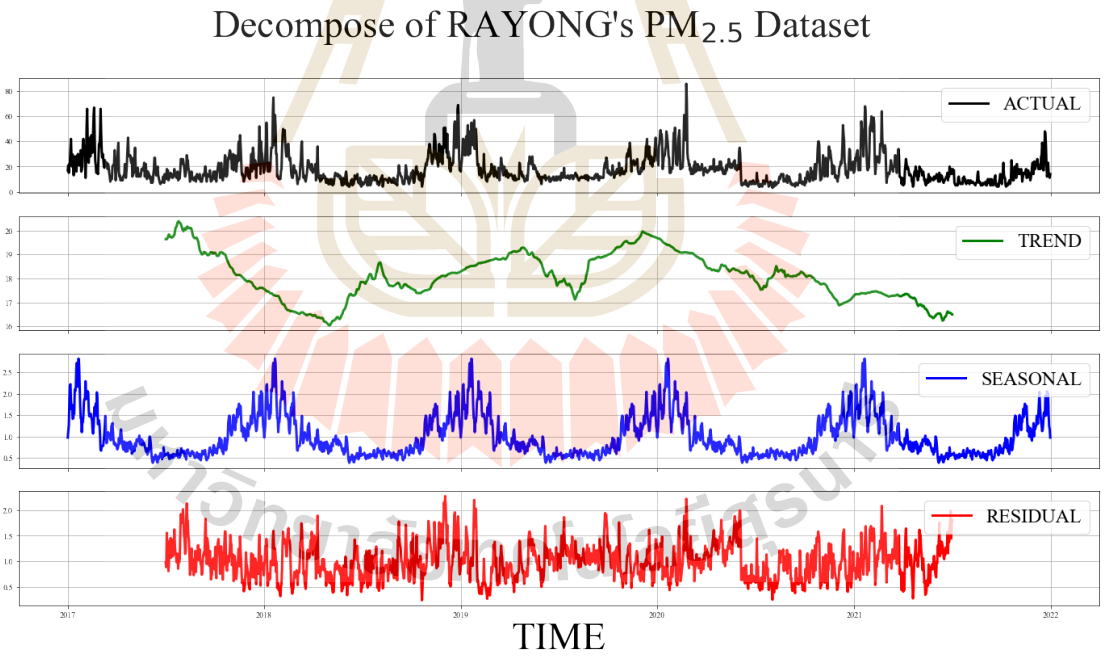
รูปที่ 4.5 แสดงรายละเอียดของแบบจำลองอนุกรมเวลา ARIMA(2, 0, 2)

ตารางที่ 4.3 แสดงผลลัพธ์ของ Augmented Dickey Fuller Test

No.	Name		Values
1	ADF		3.5284
2	P-Value		0.0072
3	Number of Lag		20
4	Number of Observations Used for ADF Regression and Critical Values Calculation		1805
5	Critical Values:	1%	-3.4339
		5%	-2.8631
		10%	-2.5676
6	Stationary		Yes

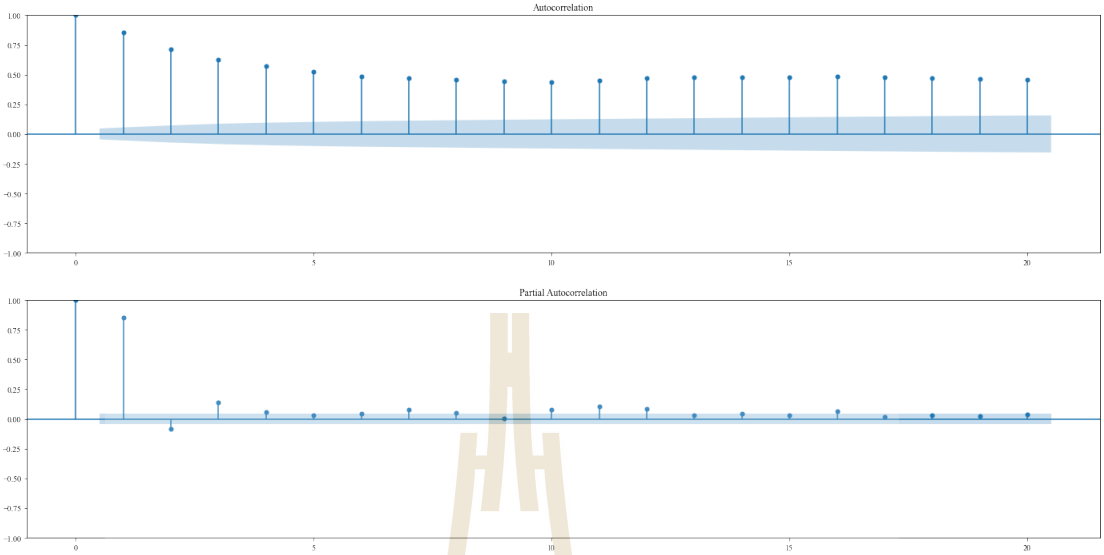


รูปที่ 4.6 แสดงกราฟข้อมูลอนุกรมเวลาฝุ่น PM2.5 ของสถานี 30T ช่วงปี ค.ศ. 2017 - 2021



รูปที่ 4.7 แสดงกราฟ Decompose ข้อมูลอนุกรมเวลาฝุ่น PM2.5 ของสถานี 30T

Auto Correlation Function: ACF และ Partial Auto Correlation Function: PACF กราฟ เป็นเครื่องมือที่ใช้วิเคราะห์ความสัมพันธ์ของข้อมูลอนุกรมเวลา และใช้สำหรับพิจารณาเลือกพารามิเตอร์ p และ q ของแบบจำลองอนุกรมเวลา ARIMA รูปภาพที่ 4.8 แสดงกราฟ ACF และ PACF ข้อมูลอนุกรมเวลาฝุ่น PM2.5ของสถานีวัดคุณภาพอากาศรหัส 30T

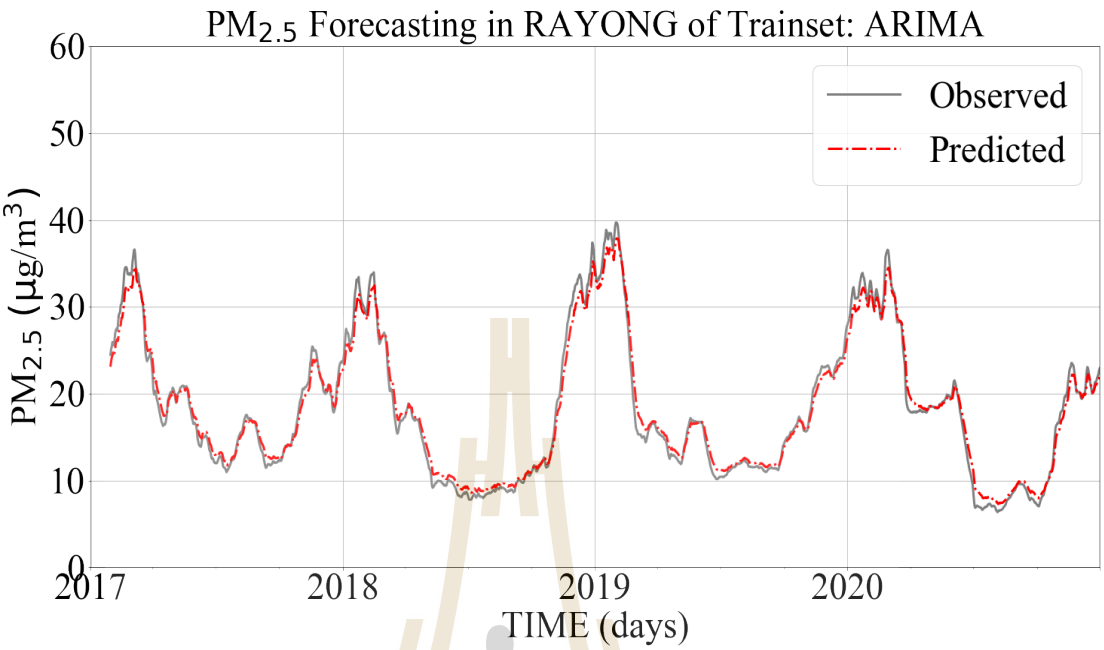


รูปที่ 4.8 แสดง ACF และ PACF กราฟ ของข้อมูลอนุกรมเวลาฝุ่น PM2.5 ของสถานีวัดคุณภาพอากาศรหัส 30T

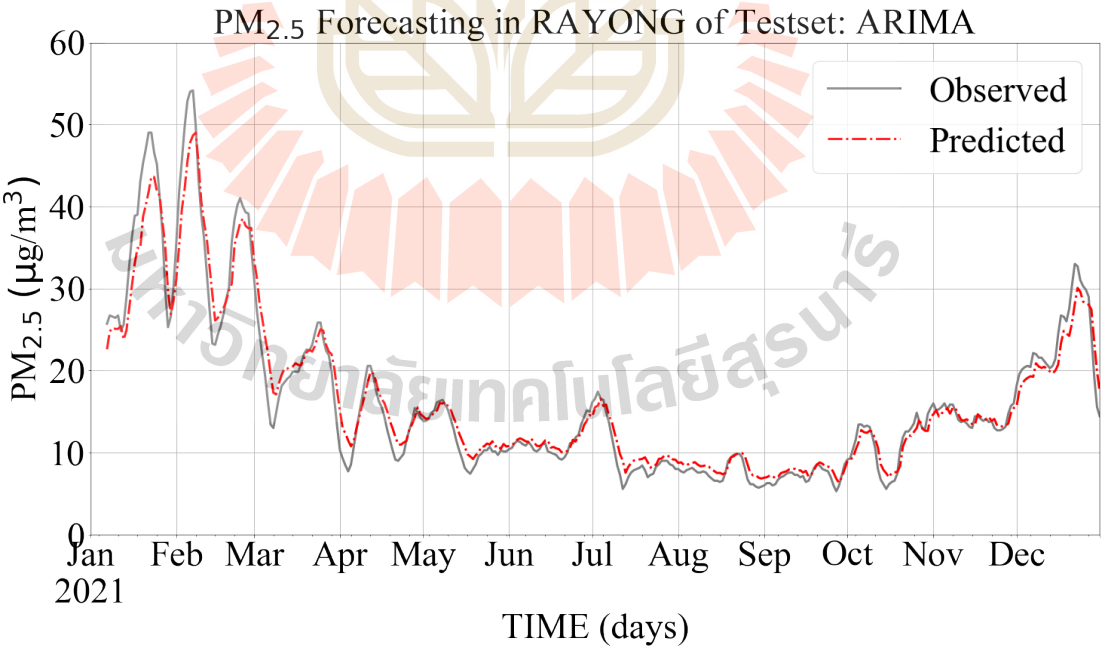
ผลการวัดประสิทธิภาพการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ของแบบจำลองอนุกรมเวลา ARIMA(2, 0, 2) ด้วยมาตรวัดค่าสัมประสิทธิ์การตัดสินใจ R^2 และกลุ่มของมาตรวัดค่าความคลาดเคลื่อน ได้แก่ MAE, MAPE, RMSE, และ %RMSE กับชุดข้อมูลเรียนรู้ และชุดข้อมูลทดสอบของชุดข้อมูลอนุกรมเวลาฝุ่น PM2.5 ได้ผลลัพธ์ ดังนี้ 1) ผลการวัดประสิทธิภาพของชุดข้อมูลเรียนรู้ให้ค่า R^2, MAE, MAPE, RMSE, และ %RMSE เท่ากับ 0.73, 3.90, 23.54, 5.81, และ 31.76 ตามลำดับ 2) ผลการวัดประสิทธิภาพของชุดข้อมูลทดสอบให้ค่า R^2, MAE, MAPE, RMSE, และ %RMSE เท่ากับ 0.79, 3.66, 26.17, 5.42, และ 32.86 ตามลำดับ รูปที่ 4.9 และ 4.10 แสดงกราฟผลการพยากรณ์ค่าฝุ่น PM2.5 ล่วงหน้า ด้วยแบบจำลองอนุกรมเวลา ARIMA(2, 0, 2) ของชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบ ตามลำดับ

4.2.2 ผลประเมินประสิทธิภาพแบบจำลองอนุกรมเวลา ANN

สำหรับการพัฒนาแบบจำลองอนุกรมเวลาไม่เชิงเส้น ด้วยแบบจำลองโครงข่ายประสาทเทียมมาตรฐาน ANN สำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ผู้วิจัยได้พัฒนาและประเมินประสิทธิภาพแบบจำลองอนุกรมเวลา ANN ด้วยเทคนิค Approximate Search แบบ Trial and Error โดยกำหนดพารามิเตอร์ของโครงข่ายประสาทเทียม ดังนี้ 1) พิจารณาจำนวน Neural Node ตั้งแต่ 32 ถึง 256 Node โดยแบ่งเป็นชุดของจำนวน Neural Node ดังนี้ 32, 64, 128, และ 256 Nodes 2) พิจารณาจำนวนชั้นของ Hidden Layer ตั้งแต่ 1 ถึง 4 ชั้น โดยแบ่งทดสอบดังนี้ 1, 2, และ 4 ชั้น สำหรับการกำหนดโครงข่ายประสาทเทียมของแบบจำลองอนุกรมเวลา ANN จะพิจารณาทั้ง 2 ข้อกำหนดร่วมกัน จึงทำให้ได้ชุดโครงข่ายประสาทเทียมของแบบจำลองอนุกรมเวลา ANN



รูปที่ 4.9 กราฟแสดงผลการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ด้วยแบบจำลองอนุกรมเวลาเชิงเส้น ARIMA ของชุดข้อมูลเรียนรู้



รูปที่ 4.10 กราฟแสดงผลการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ด้วยแบบจำลองอนุกรมเวลาเชิงเส้น ARIMA ของชุดข้อมูลทดสอบ

จะพิจารณาทั้ง 2 ข้อกำหนดร่วมกัน จึงทำให้ได้ชุดโครงข่ายประสาทเทียมของแบบจำลองอนุกรมเวลา ANN สำหรับทดสอบจำนวน 12 โครงข่าย ในรูปแบบ “ANN_x_y” โดยที่ x คือจำนวน Neural Node และ y คือจำนวน Hidden Layer แจกแจงได้ดังนี้ ANN_32_1, ANN_64_1, ANN_128_1, ANN_256_1, ANN_16_2, ANN_32_2, ANN_64_2, ANN_128_2, ANN_8_4, ANN_16_4, ANN_32_4, และ ANN_64_4 สำหรับจำนวนพารามิเตอร์ที่ต้องปรับปรุงค่าของแต่ละแบบจำลองอนุกรมเวลา ANN แสดงดังตารางที่ 4.4 ในส่วนของ Activation Function ของ Neural Node ผู้วิจัยเลือกใช้ Rectified Linear Unit function: ReLU ซึ่งเป็น Activation Function ที่กรองเอาเฉพาะค่าบวกสำหรับนำส่งเป็นข้อมูลออกจาก Neural Node

ตารางที่ 4.4 แสดงจำนวนพารามิเตอร์ปรับค่าได้ของแต่ละแบบจำลองอนุกรมเวลา ANN

Topology	Parameter	Topology	Parameter
1-Hidden Layer			
ANN_32_1	161	ANN_64_1	321
ANN_128_1	641	ANN_256_1	1,281
2-Hidden Layer			
ANN_16_2	353	ANN_32_2	1,217
ANN_64_2	4,481	ANN_128_2	17,153
4-Hidden Layer			
ANN_8_4	257	ANN_16_4	897
ANN_32_4	3,329	ANN_64_4	12,801

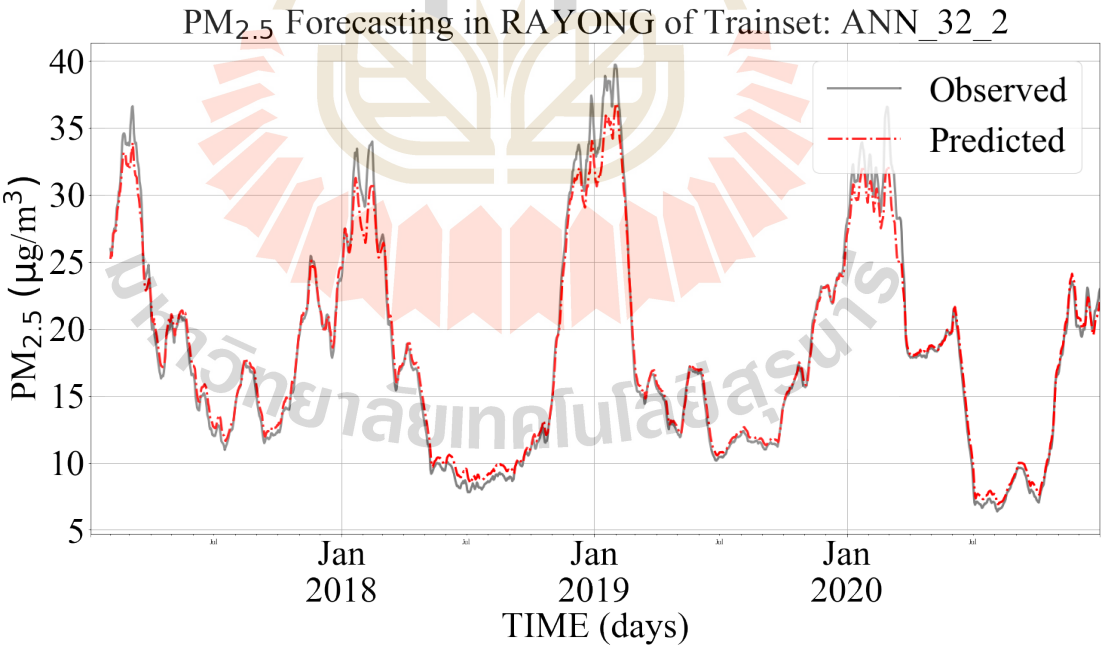
พิจารณาจากกราฟการเรียนรู้ของแต่ละแบบจำลองอนุกรมเวลา ANN ดังรูปที่ 4.11 โดยที่กำหนดจำนวนรอบการเรียนรู้ที่ 1,000 รอบ พบว่าแบบจำลองอนุกรมเวลา ANN_32_4 ให้อัตราการเรียนรู้ดีที่สุด โดยมีค่าความคลาดเคลื่อน MSE เท่ากับ 27.02 และผลการประเมินประสิทธิภาพแบบจำลองอนุกรมเวลา ANN แต่ละโครงข่ายทั้งของชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบ ผ่านมาตรวัด R^2, MAE, MAPE, RMSE, และ %RMSE ดังตารางที่ 4.5 พบว่าแบบจำลองอนุกรมเวลา ANN_32_4 เป็นแบบจำลองที่ให้ประสิทธิภาพโดยรวมดีที่สุดกับชุดข้อมูลเรียนรู้ ส่วนผลการประเมินประสิทธิภาพแบบจำลองอนุกรมเวลา ANN กับชุดข้อมูลทดสอบพบว่าแบบจำลองอนุกรมเวลา ANN_32_2 มีค่าประสิทธิภาพโดยรวมดีที่สุด ดังนั้น การพัฒนาและค้นหาแบบจำลองอนุกรมเวลา ANN ที่มีค่าประสิทธิภาพโดยรวมดีที่สุดสำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า คือแบบจำลองอนุกรมเวลา ANN_32_2 โดยให้ค่าประเมินประสิทธิภาพผ่านมาตรวัด R^2, MAE, MAPE,

RMSE, และ %RMSE กับชุดข้อมูลเรียนรู้ ดังนี้ 0.77, 3.64, 21.99, 5.36, และ 29.31 ตามลำดับ และให้ค่าประเมินประสิทธิภาพผ่านมาตรวัด R^2, MAE, MAPE, RMSE, และ %RMSE กับชุดข้อมูลทดสอบ ดังนี้ 0.78, 3.78, 26.95, 5.65, และ 34.47 ตามลำดับ นอกจากนี้แบบจำลองอนุกรมเวลา ANN_32_2 ยังให้ค่า Overfitting ที่ 9.09% และใช้เวลาสำหรับการเรียนรู้ประมาณ 19 นาที รูปที่ 4.12 และ 4.13 กราฟแสดงผลการพยากรณ์ฝุ่น PM2.5 ล่วงหน้าด้วยแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANN_32_2 ของชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบ ตามลำดับ

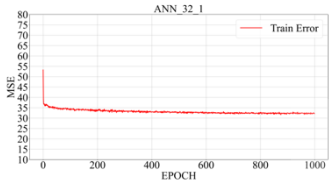
สำหรับการคำนวณค่า Overfitting จะพิจารณาจากความคลาดเคลื่อน RMSE ดังสมการที่ (4-1) (รติพร จันทร์กลั่น, 2560: 73)

$$Overfitting = \frac{|RMSE_{train} - RMSE_{test}|}{RMSE_{train}} \times 100 \quad (4\text{-}1)$$

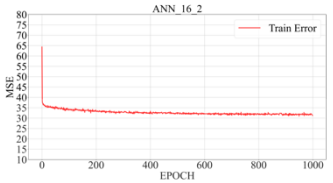
โดยที่ $RMSE_{train}$ และ $RMSE_{test}$ เป็นค่าความคลาดเคลื่อน RMSE ของแบบจำลองอนุกรมเวลาที่กระทำกับชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบ ตามลำดับ



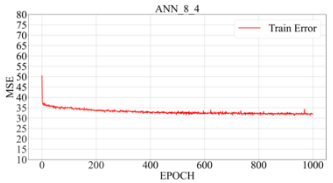
รูปที่ 4.11 กราฟแสดงผลการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ด้วยแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANN_32_2 ของชุดข้อมูลเรียนรู้



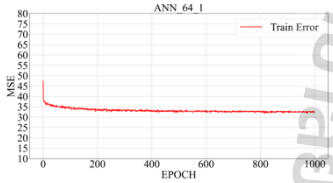
ANN_32_1



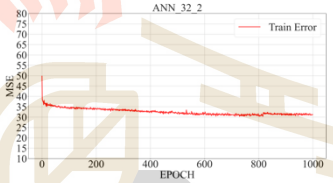
ANN_16_2



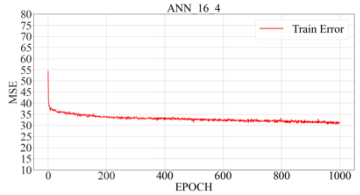
ANN_8_4



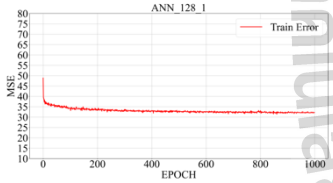
ANN_64_1



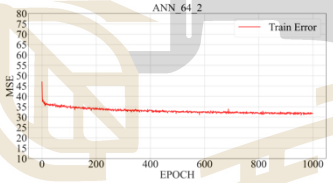
ANN_32_2



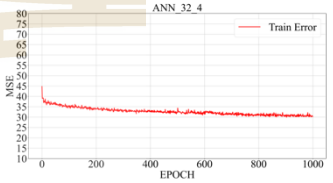
ANN_16_4



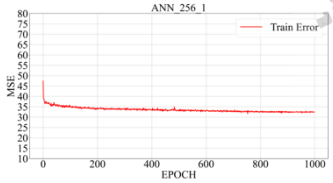
ANN_128_1



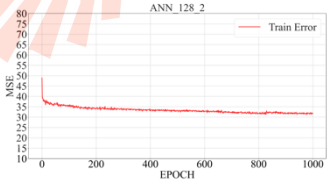
ANN_64_2



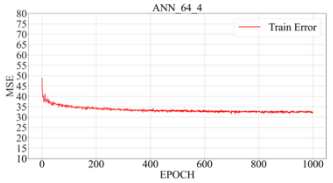
ANN_32_4



ANN_256_1



ANN_128_2

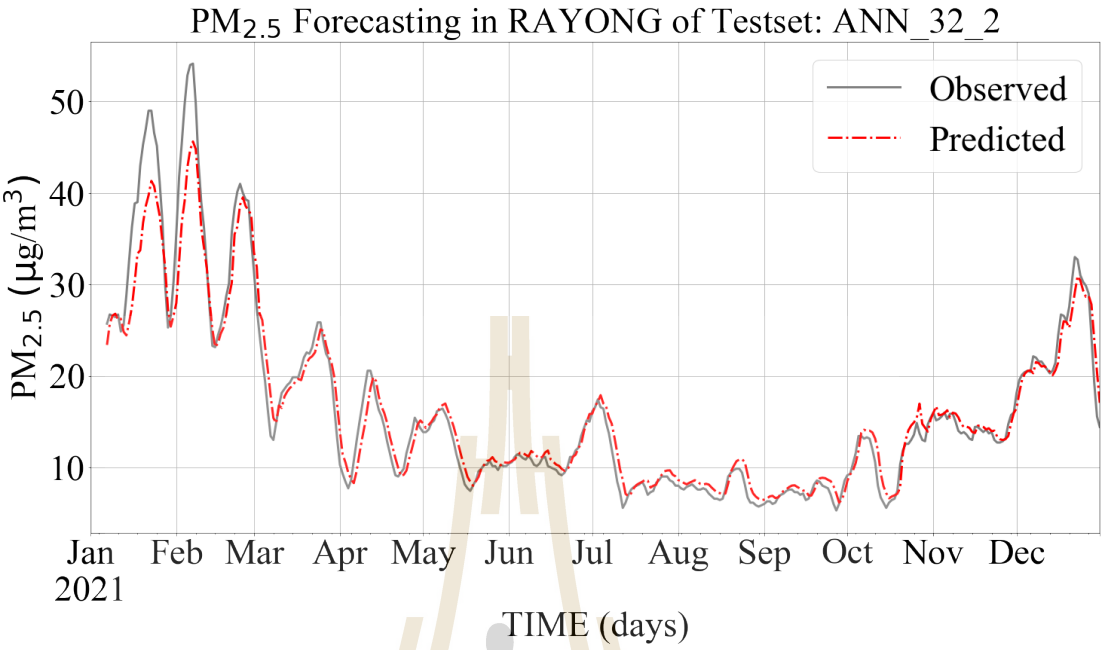


ANN_64_4

รูปที่ 4.12 กราฟแสดงการเรียนรู้ของแบบจำลองอนุกรมเวลา ANN

ตารางที่ 4.5 ผลประเมินประสิทธิภาพแบบจำลองอนุกรมเวลา ANN

	ANN_Node_Layer											
Topology	32_1	64_1	128_1	256_1	16_2	32_2	64_2	128_2	8_4	16_4	32_4	64_4
	Train Set											
R^2	0.7532	0.7355	0.7490	0.7488	0.7580	0.7730	0.7638	0.7754	0.7350	0.7616	0.7868	0.7629
MAE	3.7613	3.9980	3.8102	3.7887	3.7478	3.6461	3.7355	3.6291	4.1638	3.7827	3.4205	3.6000
MAPE	23.0334	25.2397	23.3809	23.1450	22.9251	21.9931	23.0612	22.9917	27.1444	23.4322	20.1983	21.0567
RMSE	5.5931	5.7906	5.6413	5.6436	5.5390	5.3639	5.4718	5.3357	5.7961	5.4976	5.1986	5.4821
%RMSE	30.5626	31.6417	30.8259	30.8382	30.2669	29.3104	29.8998	29.1561	31.6719	30.0408	28.4067	29.9559
Time (m)	15	15	15	17	15	19	21	21	16	16	22	23
	Test Set											
R^2	0.7756	0.7695	0.7816	0.7823	0.7343	0.7889	0.7156	0.7376	0.7492	0.7527	0.7571	0.7470
MAE	3.9033	4.0136	3.8648	3.8870	4.1553	3.7821	4.2242	4.2674	4.2605	4.1143	4.0176	4.0688
MAPE	27.4204	29.7884	27.6013	27.5730	27.9483	26.9558	28.6843	30.8258	31.8384	28.9181	27.6169	27.1505
RMSE	5.6447	5.7215	5.5694	5.5597	6.1432	5.6516	6.3552	6.1043	5.9676	5.9261	5.8733	5.9938
%RMSE	34.2249	34.6905	33.7685	33.7095	37.2470	34.4792	38.5329	37.0112	36.1828	35.9308	35.6105	36.3416
Overfitting	0.9226	1.1928	1.2734	1.4856	10.9073	9.0912	16.1450	14.4040	2.9593	7.7935	12.9779	9.3347



รูปที่ 4.13 กราฟแสดงผลการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ด้วยแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANN_32_2 ของชุดข้อมูลทดสอบ

4.2.3 ผลประเมินประสิทธิภาพแบบจำลองอนุกรมเวลา LSTM

สำหรับการพัฒนาแบบจำลองอนุกรมเวลาไม่เชิงเส้น ด้วยแบบจำลองโครงข่ายประสาทเทียมการเรียนรู้เชิงลึก LSTM สำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ผู้วิจัยได้พัฒนาและประเมินประสิทธิภาพแบบจำลองอนุกรมเวลา LSTM ด้วยเทคนิค Approximate Search แบบ Trial and Error โดยกำหนดพารามิเตอร์ของโครงข่ายประสาทเทียม ดังนี้ 1) จำนวน Neural Node ตั้งแต่ 16 ถึง 128 โดยแบ่งเป็นชุดของจำนวน Neural Node ดังนี้ 16, 32, 64, และ 128 Nodes 2) จำนวนชั้นของ Hidden Layer ตั้งแต่ 1 ถึง 4 ชั้น โดยแบ่งทดสอบดังนี้ 1, 2, และ 4 ชั้น สำหรับการกำหนดโครงข่ายของแบบจำลองอนุกรมเวลา LSTM จะพิจารณาทั้ง 2 ข้อกำหนดร่วมกัน จึงทำให้ได้ชุดโครงข่ายประสาทเทียมของแบบจำลองอนุกรมเวลา LSTM สำหรับทดสอบจำนวน 12 โครงข่าย ในรูปแบบ “LSTM_x_y” โดยที่ x คือจำนวน Neural Node และ y คือจำนวน Hidden Layer แจกแจงได้ดังนี้ LSTM_16_1, LSTM_32_1, LSTM_64_1, LSTM_128_1, LSTM_8_2, LSTM_16_2, LSTM_32_2, LSTM_64_2, LSTM_4_4, LSTM_8_4, LSTM_16_4, และ LSTM_32_4 โดยจำนวนพารามิเตอร์ที่ต้องปรับปรุงค่าของแต่ละโครงข่ายแสดงดังตารางที่ 4.6 สำหรับ Activation Function ของ Neural Node ผู้วิจัยเลือกใช้ Rectified Linear Unit function: ReLU ซึ่งเป็น Activation Function ที่กรองเอาเฉพาะค่าบวกสำหรับนำส่งเป็นข้อมูลออกจาก Neural Node

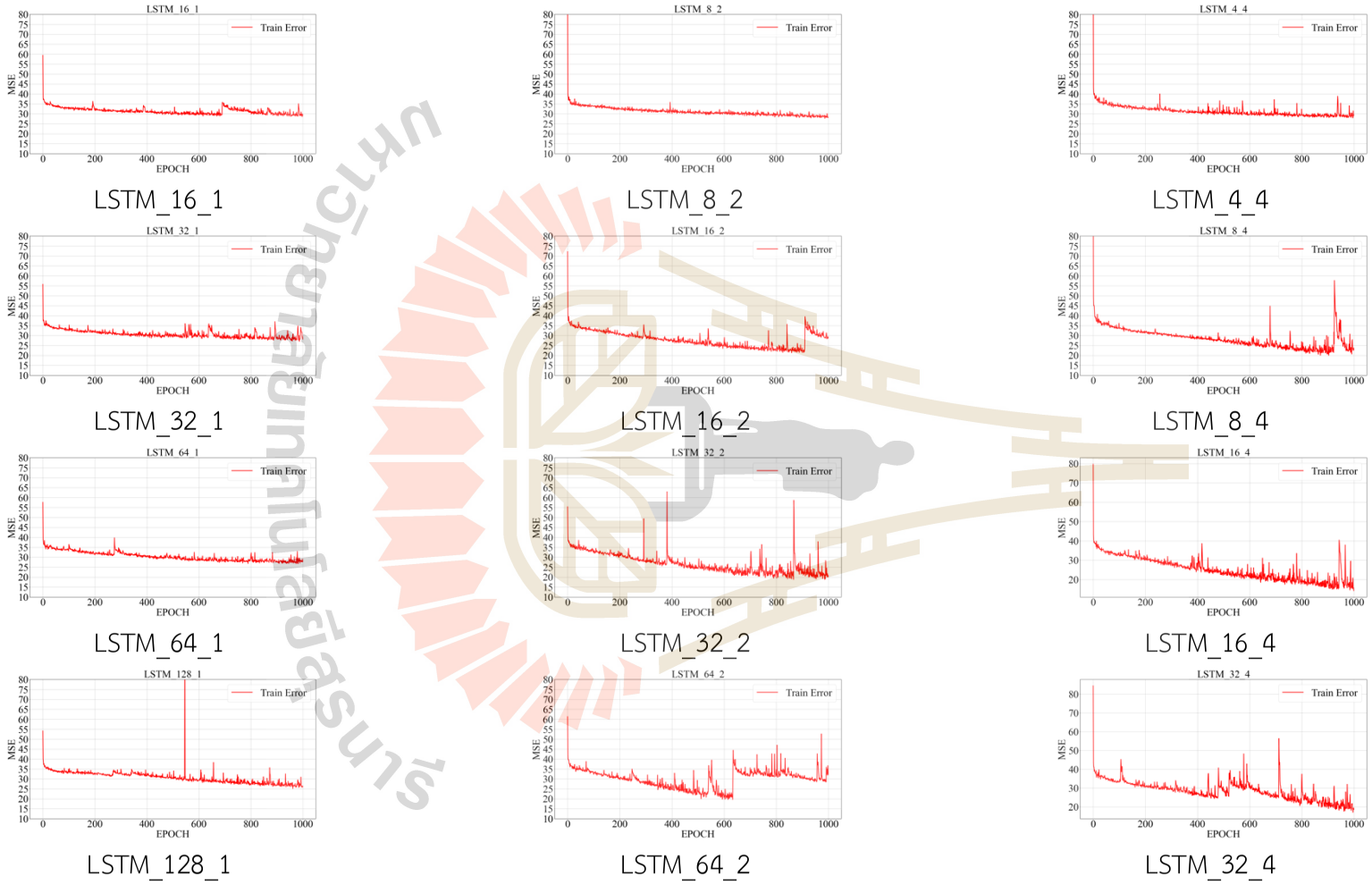
ตารางที่ 4.6 แสดงจำนวนพารามิเตอร์ปรับค่าได้ของแต่ละแบบจำลองอนุกรมเวลา LSTM

Topology	Parameter	Topology	Parameter
1-Hidden Layer			
LSTM_16_1	1,169	LSTM_32_1	4,385
LSTM_64_1	16,9961	LSTM_128_1	66,689
2-Hidden Layer			
LSTM_8_2	873	LSTM_16_2	3,281
LSTM_32_2	12,705	LSTM_64_2	49,985
4-Hidden Layer			
LSTM_4_4	533	LSTM_8_4	1,961
LSTM_16_4	7,505	LSTM_32_4	29,345

พิจารณาจากกราฟการเรียนรู้ของแต่ละแบบจำลองอนุกรมเวลา LSTM โดยที่กำหนดจำนวนรอบการเรียนรู้ที่ 1,000 รอบ พบว่าแบบจำลองอนุกรมเวลา LSTM_32_2 ให้อัตราการเรียนรู้ดีที่สุด โดยมีค่าความคลาดเคลื่อน MSE เท่ากับ 21.10 ดังรูปที่ 4.14 และผลการประเมินประสิทธิภาพแบบจำลองอนุกรมเวลา LSTM แต่ละโครงข่ายทั้งของชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบ ผ่านมาตรวัด R^2, MAE, MAPE, RMSE, และ %RMSE ดังตารางที่ 4.7 พบว่าแบบจำลองอนุกรมเวลา LSTM_32_2 เป็นแบบจำลองที่ให้ค่าประสิทธิภาพดีที่สุดในทุกมาตรวัดกับชุดข้อมูลเรียนรู้ แต่หากพิจารณาร่วมกับค่า Overfitting พบว่ามีค่าที่สุดถึง 36.51% ทำให้แบบจำลองอนุกรมเวลา LSTM_32_2 ไม่เหมาะสมต่อการนำไปใช้งาน ส่วนผลการประเมินประสิทธิภาพแบบจำลองอนุกรมเวลา LSTM กับชุดข้อมูลทดสอบพบว่าแบบจำลองอนุกรมเวลา LSTM_4_4 ให้ค่าประสิทธิภาพดีที่สุดในทุกมาตรวัด และแบบจำลองอนุกรมเวลา LSTM_4_4 ยังให้ค่า Overfitting ที่ต่ำเพียง 8.53% โดยที่มีค่าประเมินประสิทธิภาพผ่านมาตรวัด R^2, MAE, MAPE, RMSE, และ %RMSE กับชุดข้อมูลเรียนรู้ ดังนี้ 0.7853, 3.4781, 20.1383, 5.2168, และ 28.5062 ตามลำดับ และให้ค่าประเมินประสิทธิภาพผ่านมาตรวัด R^2, MAE, MAPE, RMSE, และ %RMSE กับชุดข้อมูลทดสอบ ดังนี้ 0.7743, 3.8332, 25.2997, 5.6619, และ 34.3291 ตามลำดับ รูปที่ 4.15 และ 4.16 กราฟแสดงผลการพยากรณ์ฝุ่น PM2.5 ล่วงหน้าด้วยแบบจำลองอนุกรมเวลาไม่เชิงเส้น LSTM_4_4 ของชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบ ตามลำดับ

4.2.4 ผลประเมินประสิทธิภาพแบบจำลองอนุกรมเวลา ANFIS

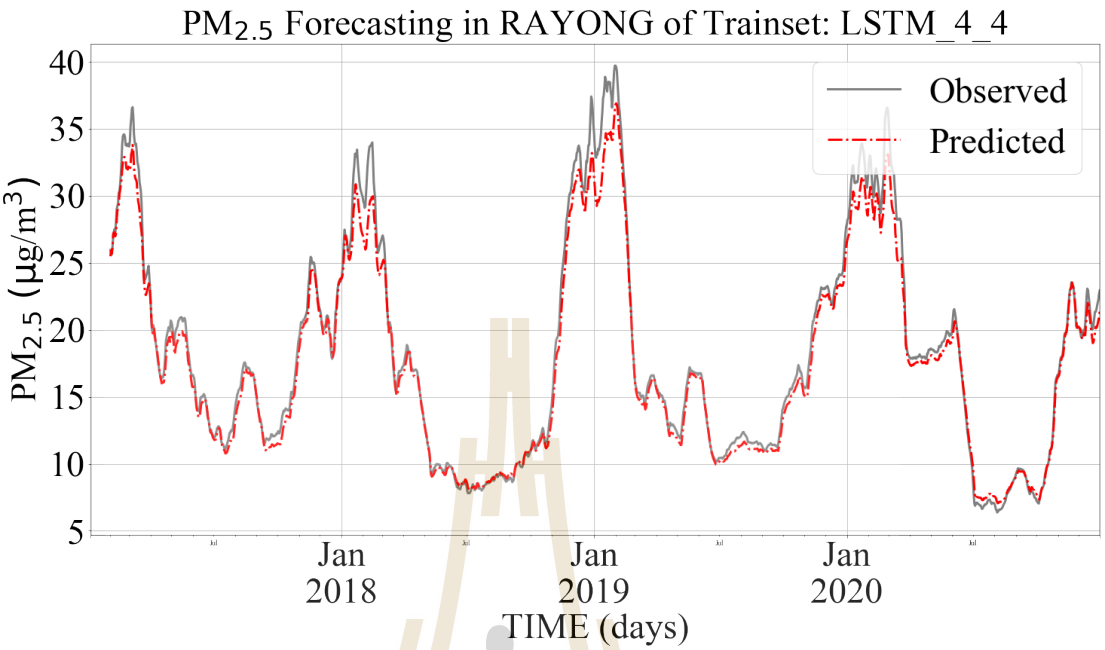
การพัฒนาแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS เพื่อใช้พยากรณ์ฝุ่น PM2.5 ล่วงหน้า ผู้วิจัยได้พัฒนาและประเมินประสิทธิภาพแบบจำลองอนุกรมเวลา ANFIS ด้วยเทค-



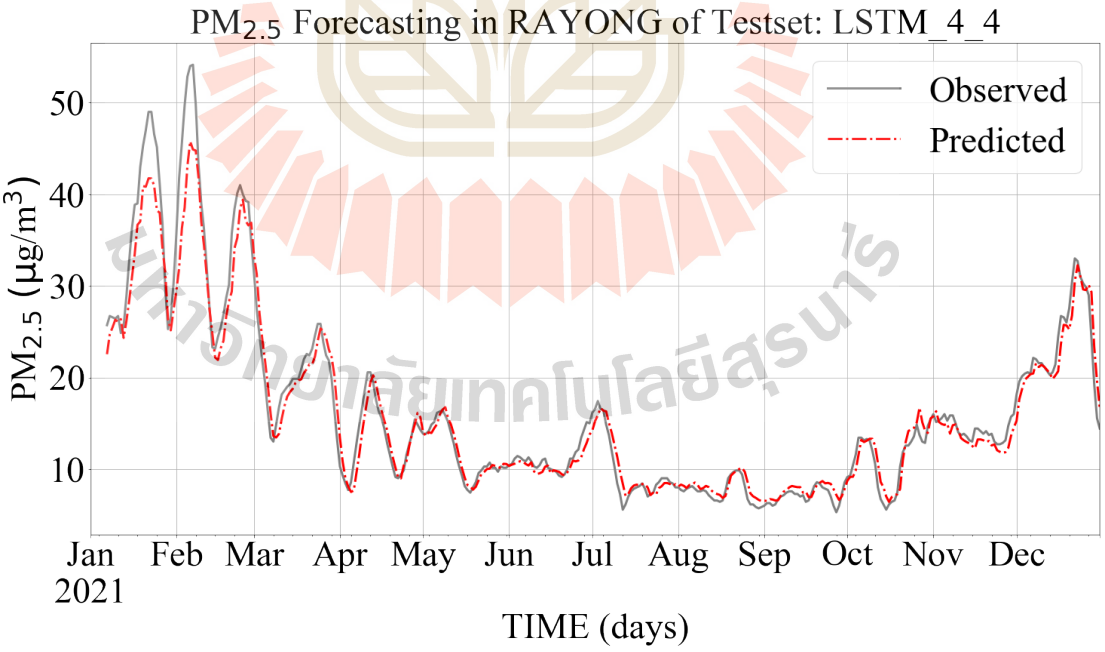
รูปที่ 4.14 กราฟแสดงการเรียนรู้ของแบบจำลองอนุกรมเวลา LSTM

ตารางที่ 4.7 ผลประเมินประสิทธิภาพแบบจำลองอนุกรมเวลา LSTM

LSTM_Node_Layer												
Topology	16_1	32_1	64_1	128_1	8_2	16_2	32_2	64_2	4_4	8_4	16_4	32_4
Train Set												
R^2	0.7795	0.7663	0.7658	0.8022	0.8261	0.7703	0.8335	0.7937	0.7853	0.8306	0.7800	0.7220
MAE	3.5522	3.6128	3.6146	3.3743	3.2167	3.5869	3.0179	3.3248	3.4781	3.1326	3.5585	3.8849
MAPE	21.1843	21.7930	21.0372	20.2985	19.9173	21.2768	18.4051	19.5262	20.1383	18.8675	21.9128	22.9991
RMSE	5.2869	5.4427	5.4493	5.0074	4.6956	5.3965	4.5944	5.1136	5.2168	4.6336	5.2809	5.9364
%RMSE	28.8892	29.7407	29.7766	27.3620	25.6581	29.4883	25.1054	27.9425	28.5062	25.3194	28.8568	32.4385
Time (m)	33	34	36	39	43	52	55	61	75	76	91	96
Test Set												
R^2	0.7553	0.7623	0.7203	0.7387	0.7547	0.7386	0.7230	0.7364	0.7743	0.7165	0.74077	0.7136
MAE	4.0256	4.0633	4.1953	4.1644	4.1166	4.0964	4.3612	4.3507	3.8332	4.2877	4.1134	4.2765
MAPE	27.0278	28.3141	27.7139	29.0756	28.1822	27.6140	29.6335	31.0845	25.2997	27.8742	28.1290	29.3905
RMSE	5.8955	5.8104	6.3023	6.0924	5.9019	6.0926	6.2720	6.1183	5.6619	6.3456	6.0682	6.3773
%RMSE	35.7452	35.2296	38.2121	36.9394	35.7840	36.9407	38.0280	37.0964	34.3291	38.4744	36.7922	38.6667
Overfitting	11.5113	6.7564	15.6545	21.6685	25.6900	12.8994	36.5125	19.6476	8.5324	36.9478	14.9064	7.4269



รูปที่ 4.15 กราฟแสดงผลการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ด้วยแบบจำลองอนุกรมเวลาไม่เชิงเส้น LSTM_4_4 ของชุดข้อมูลเรียนรู้



รูปที่ 4.16 กราฟแสดงผลการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ด้วยแบบจำลองอนุกรมเวลาไม่เชิงเส้น LSTM_4_4 ของชุดข้อมูลทดสอบ

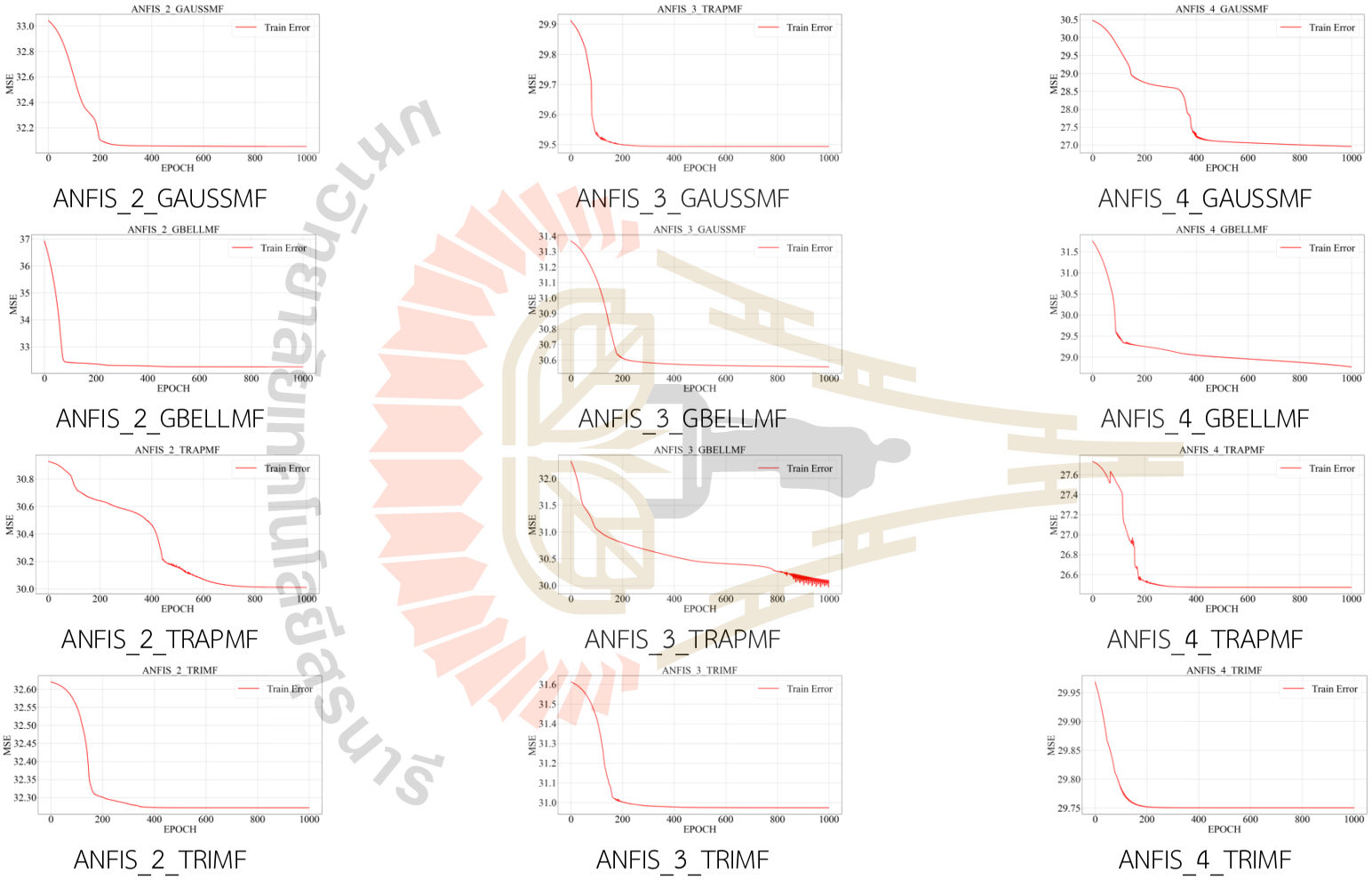
นิค Approximate Search แบบ Trial and Error โดยกำหนดพารามิเตอร์ของโครงข่ายที่ต้องการปรับค่า ดังนี้ 1) จำนวนสมาชิกฟังก์ชัน ที่จำนวน 2, 3, และ 4 ต่อข้อมูลนำเข้า โดยจำนวนสมาชิกฟังก์ชันจะแสดงถึงเกณฑ์ของระดับข้อมูลนำเข้าในรูปแบบของตัวแปรทางภาษา (Linguistic Value) เช่น จำนวนสมาชิกฟังก์ชันเท่ากับ 2 แสดงระดับข้อมูลนำเข้าฝุ่น PM2.5 ในรูปแบบตัวแปรทางภาษา คือ {“ระดับปลอดภัย”, “ระดับอันตราย”} หรือ จำนวนสมาชิกฟังก์ชันเท่ากับ 3 แสดงระดับข้อมูลนำเข้าฝุ่น PM2.5 ในรูปแบบตัวแปรทางภาษา คือ {“ระดับปลอดภัย”, “ระดับมีความเสี่ยง”, “ระดับอันตราย”} เป็นต้น และ 2) ชนิดของสมาชิกฟังก์ชัน ได้แก่ Gaussian membership function (gaussmf), Generalized bell-shaped membership function (gbellmf), Trapezoidal membership function (trapmf), และ Triangular membership function (trimf) สำหรับการกำหนดพารามิเตอร์ของแบบจำลองอนุกรมเวลา ANFIS จะพิจารณาทั้ง 2 ข้อกำหนดร่วมกัน จึงทำให้ได้รูปแบบของการกำหนดพารามิเตอร์ของแบบจำลองอนุกรมเวลา ANFIS เป็น “ANFIS_x_y” โดยที่ x คือจำนวนสมาชิกฟังก์ชันต่อข้อมูลนำเข้า และชนิดของสมาชิกฟังก์ชัน ซึ่งแจกแจงได้ทั้งหมด 12 รูปแบบ ดังนี้ ANFIS_2_GAUSSMF, ANFIS_2_GBELLMF, ANFIS_2_TRAPMF, ANFIS_2_TRIMF, ANFIS_3_GAUSSMF, ANFIS_3_GBELLMF, ANFIS_3_TRAPMF, ANFIS_3_TRIMF, ANFIS_4_GAUSSMF, ANFIS_4_GBELLMF, ANFIS_4_TRAPMF, และ ANFIS_4_TRIMF โดยจำนวนพารามิเตอร์ที่ต้องปรับปรุงค่ารวมถึงจำนวน Fuzzy Rule ของแต่ละแบบจำลองอนุกรมเวลา ANFIS แสดงดังตารางที่ 4.8 แบบจำลองอนุกรมเวลา ANFIS ใช้อัลกอริทึม Grid Partition ในการกำหนด Fuzzy Rule ผ่านชุดข้อมูลอนุกรมเวลาฝุ่น PM2.5 เนื่องจากเป็นอัลกอริทึมที่รองรับ Fuzzy Inference System: FIS ในรูปแบบของ Takagi-Sugeno ซึ่งเป็นชนิดของ FIS ที่ใช้ในแบบจำลองอนุกรมเวลา ANFIS และใช้กระบวนการเรียนรู้ผ่านชุดข้อมูลเหมือนกับแบบจำลองโครงข่ายประสาทเทียม คือ Gradient Descent Algorithm

พิจารณาจากกราฟการเรียนรู้ของแต่ละแบบจำลองอนุกรมเวลา ANFIS ดังรูปที่ 4.17 โดยที่กำหนดจำนวนรอบการเรียนรู้ที่ 1,000 รอบ พบว่าแบบจำลองอนุกรมเวลา ANFIS_4_TRAPMF ให้อัตราการเรียนรู้ดีที่สุด ที่ค่าความคลาดเคลื่อน MSE เท่ากับ 26.47 แต่จากการพิจารณาค่า Overfitting จากตารางแสดงผลการประเมินประสิทธิภาพ ตารางที่ 4.9 พบว่าแบบจำลองอนุกรมเวลา ANFIS_4_TRAPMF มีการ Overfitting กับชุดข้อมูลที่สูงมาก คือ 1,632.43% จึงทำให้แบบจำลองอนุกรมเวลา ANFIS_4_TRAPMF ไม่เหมาะสมต่อการนำไปใช้งานสำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า จากผลการประเมินประสิทธิภาพแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS กับชุดข้อมูลทดสอบ ผ่านมาตรวัด R^2, MAE, MAPE, RMSE, และ %RMSE พบว่าแบบจำลองอนุกรมเวลา ANFIS_2_GBELLMF ให้ค่าประสิทธิภาพโดยรวมดีที่สุด และมีค่า Overfitting เพียง 2.13% โดยที่แบบจำลองอนุกรมเวลา ANFIS_2_GBELLMF ให้ค่าประเมิน

ประสิทธิภาพผ่านมาตรวัด R^2, MAE, MAPE, RMSE, และ %RMSE กับชุดข้อมูลเรียนรู้ ดังนี้ 0.74, 3.81, 22.96, 5.67, และ 31.03 ตามลำดับ ส่วนกับชุดข้อมูลทดสอบแบบจำลองอนุกรมเวลา ANFIS_2_GBELLMF ให้ค่าประเมินประสิทธิภาพผ่านมาตรวัด R^2, MAE, MAPE, RMSE, และ %RMSE ดังนี้ 0.78, 3.81, 26.55, 5.55, และ 33.69 ตามลำดับ สำหรับค่า Overfitting ของแบบจำลองอนุกรมเวลา ANFIS_2_GBELLMF มีค่าเท่ากับ 2.13% ส่วนเวลาที่ใช้สำหรับการเรียนรู้ของแบบจำลอง ANFIS_2_GBELLMF ประมาณ 23 วินาที รูปที่ 4.18 และ 4.19 กราฟแสดงผลการพยากรณ์ฝุ่น PM2.5 ล่วงหน้าด้วยแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS_2_GBELLMF ของชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบ ตามลำดับ

ตารางที่ 4.8 แสดงจำนวนพารามิเตอร์ปรับค่าได้ของแต่ละแบบจำลองอนุกรมเวลา ANFIS

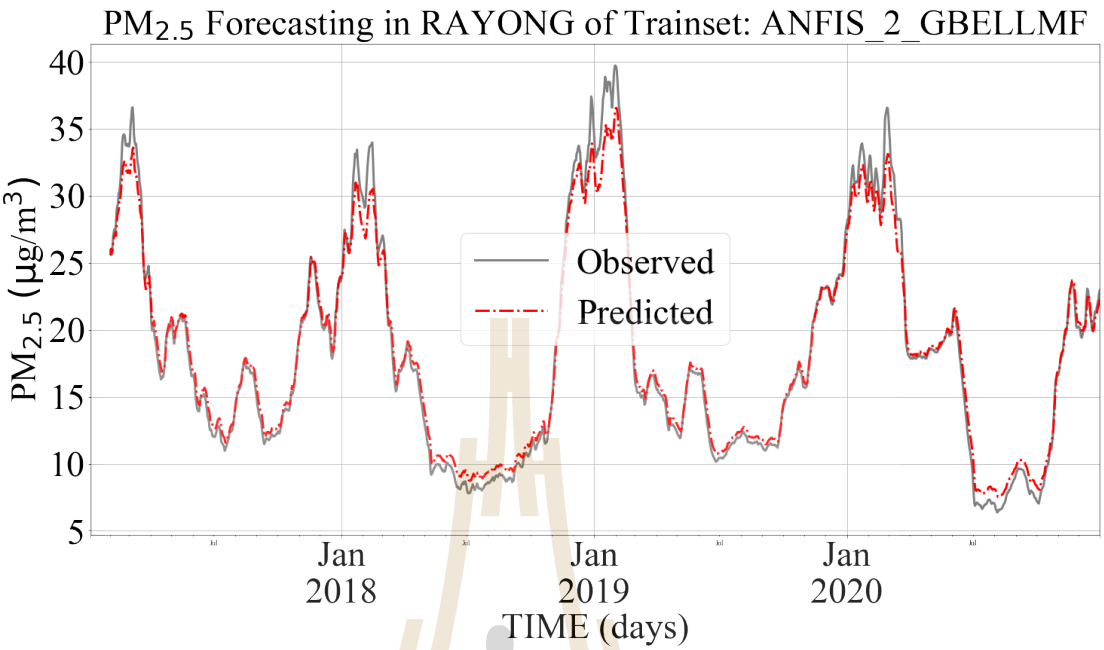
Topology	Trainable	Topology	Trainable
2 Membership Functions			
ANFIS_2_GAUSSMF	20 (with 8 rules)	ANFIS_2_TRAPMF	56 (with 8 rules)
ANFIS_2_GBELLMF	26 (with 8 rules)	ANFIS_2_TRIMF	26 (with 8 rules)
3 Membership Functions			
ANFIS_3_GAUSSMF	45 (with 27 rules)	ANFIS_3_TRAPMF	114 (with 27 rules)
ANFIS_3_GBELLMF	54 (with 27 rules)	ANFIS_3_TRIMF	54 (with 27 rules)
4 Membership Functions			
ANFIS_4_GAUSSMF	88 (with 64 rules)	ANFIS_4_TRAPMF	304 (with 64 rules)
ANFIS_4_GBELLMF	100 (with 64 rules)	ANFIS_4_TRIMF	100 (with 64 rules)



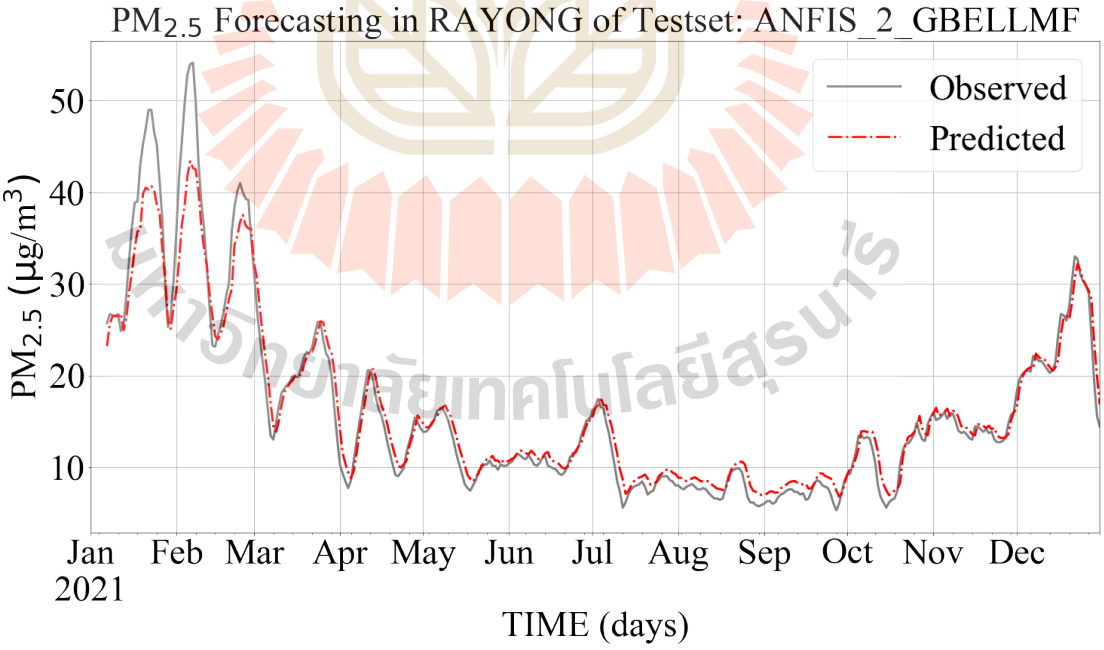
รูปที่ 4.17 แสดงกราฟการเรียนรู้แต่ละโครงข่ายของอนุกรมเวลา ANFIS

ตารางที่ 4.9 ผลทดสอบประสิทธิภาพแบบจำลองอนุกรมเวลา ANFIS

Topology	ANFIS_#MF_MF											
	GAUSSMF			GBELLMF			TRAPMF			TRIMF		
	2	3	4	2	3	4	2	3	4	2	3	4
	Train Set											
R^2	0.7472	0.7590	0.7873	0.7456	0.7636	0.7731	0.7633	0.7673	0.7912	0.7454	0.7557	0.7653
MAE	3.7927	3.7144	3.5546	3.8150	3.7331	3.6168	3.7162	3.6241	3.4970	3.8183	3.7495	3.6588
MAPE	22.8903	22.6947	22.1585	22.9620	22.8150	22.2401	22.6411	22.3173	21.6621	22.9353	22.7983	22.4190
RMSE	5.6614	5.5277	5.1923	5.6790	5.4741	5.3635	5.4782	5.4308	5.1451	5.6807	5.5653	5.4543
%RMSE	30.9360	30.2050	28.3723	31.0318	29.9121	29.3079	29.9345	29.6755	28.1143	31.0415	30.4109	29.8043
Time (s)	17.2199	48.2177	137.718	22.4654	56.1988	148.172	36.5213	275.321	1293.98	18.3447	49.6003	141.365
	Test Set											
R^2	0.7806	0.7474	-45.972	0.7825	0.7046	0.7300	0.7285	-1.1102	-54.930	0.7792	0.7263	0.7173
MAE	3.8213	4.0240	10.5823	3.8154	4.2119	4.1271	4.0628	5.5037	10.1235	3.8428	4.1196	4.1303
MAPE	26.5122	27.2142	39.1807	26.5537	27.6972	27.2777	26.9879	32.0078	46.8572	26.6266	27.3891	27.0092
RMSE	5.5824	5.9901	81.6860	5.5576	6.4774	6.1923	6.2099	17.3137	89.1353	5.5997	6.2347	6.3370
%RMSE	33.8473	36.3189	495.272	33.6966	39.2736	37.5447	37.6519	104.975	540.438	33.9520	37.8021	38.4222
Overfitting	1.3952	8.3653	1473.20	2.1373	18.3289	15.4519	13.3581	218.804	1632.43	1.4264	12.0273	16.1824



รูปที่ 4.18 กราฟแสดงผลการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ด้วยแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS_2_GBELLMF ของชุดข้อมูลเรียนรู้



รูปที่ 4.19 กราฟแสดงผลการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ด้วยแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS_2_GBELLMF ของชุดข้อมูลทดสอบ

4.2.5 ผลเปรียบเทียบประสิทธิภาพแบบจำลองอนุกรมเวลาเดี่ยว

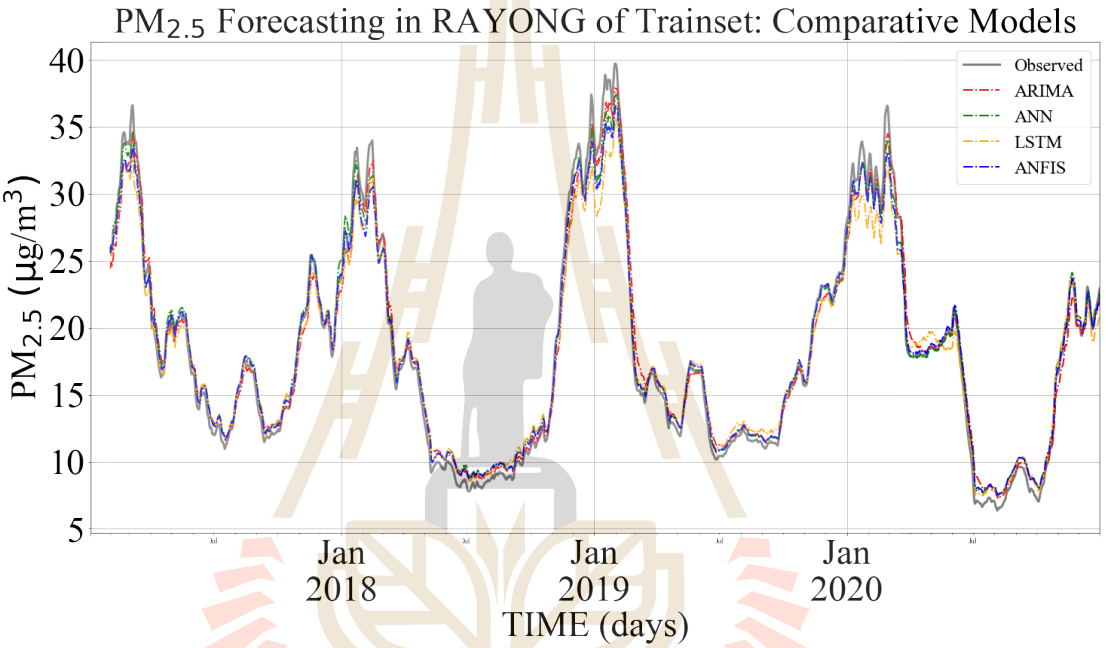
จากการพัฒนาและประเมินประสิทธิภาพแบบจำลองอนุกรมเวลา เพื่อหาแบบจำลองอนุกรมเวลาที่มีประสิทธิภาพที่ดีที่สุดสำหรับพยากรณ์ฝุ่น PM2.5 ของแบบจำลองอนุกรมเวลาเดี่ยวทั้งแบบเชิงเส้น (ARIMA) และแบบไม่เชิงเส้น (ANN, LSTM, และ ANFIS) พบว่าพารามิเตอร์ของแบบจำลองอนุกรมเวลาเดี่ยวที่มีประสิทธิภาพที่สุดของแต่ละแบบจำลอง คือ ARIMA(2, 0, 2) ANN_128_1, LSTM_16_4 และ ANFIS_2_GBELLMF

ตารางที่ 4.10 ตารางเปรียบเทียบประสิทธิภาพแบบจำลองอนุกรมเวลาเดี่ยว

	Model			
Metric	ARIMA	ANN	LSTM	ANFIS
		Train Set		
R^2	0.7329	0.7451	0.7351	**0.7456**
MAE	3.9074	3.8395	**3.7923**	3.8150
MAPE	23.5494	23.5489	**21.7317**	22.9620
RMSE	5.8133	5.6848	5.7951	**5.6790**
%RMSE	31.7681	31.0635	31.6664	**31.0318**
Time	N/A	930.1931	5412.6155	**26.0240**
		Test Set		
R^2	**0.7931**	0.7810	0.7738	0.7825
MAE	**3.6687**	3.8556	3.7754	3.8154
MAPE	**26.1751**	27.4962	25.1952	26.5537
RMSE	**5.4206**	5.5764	5.6674	5.5576
%RMSE	**32.8660**	33.8105	34.3625	33.6966
Overfitting	6.7551	**0.4731**	2.2034	2.1373

จากตารางที่ 4.10 แสดงผลการประเมินประสิทธิภาพของแบบจำลองอนุกรมเวลาเชิงเส้นและไม่เชิงเส้นเปรียบเทียบกัน พบว่าแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANN และ LSTM ให้ประสิทธิภาพดีกว่าแบบจำลองอนุกรมเวลา ARIMA และ ANFIS ในการประเมินกับชุดข้อมูลเรียนรู้ ส่วนผลการประเมินประสิทธิภาพกับชุดข้อมูลทดสอบ แบบจำลองอนุกรมเวลาเชิงเส้น ARIMA ให้ประสิทธิภาพดีที่สุด แต่ก็พบว่ามีค่า Overfitting สูงที่สุดเช่นกัน รูปที่ 4.20 และ 4.21 เป็นกราฟเปรียบเทียบผลพยากรณ์ฝุ่น PM2.5 ล่วงหน้าของชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบ ตามลำดับ

หากพิจารณาเรื่องของเวลาสำหรับการเรียนรู้ในขั้นตอนการสร้างแบบจำลองพบว่าแบบจำลองอนุกรมเวลาเชิงเส้น ARIMA ไม่ต้องใช้เวลาในส่วนนี้ แต่แบบจำลองอนุกรมเวลาเชิงเส้น ARIMA จะมีข้อจำกัดที่ต้องระบุพารามิเตอร์ p, d, และ q ก่อนที่สร้างแบบจำลอง ซึ่งเป็นลักษณะของ Parametric Model ส่วนแบบจำลองอนุกรมเวลาไม่เชิงเส้นทั้งสามแบบจำลอง ผลการทดสอบพบว่าแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS ใช้เวลาสำหรับเรียนรู้น้อยที่สุด ส่วนแบบจำลองอนุกรมเวลาไม่เชิงเส้น LSTM ใช้เวลาเรียนรู้มากที่สุด โดยคิดเป็น 200 เท่า ของแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS



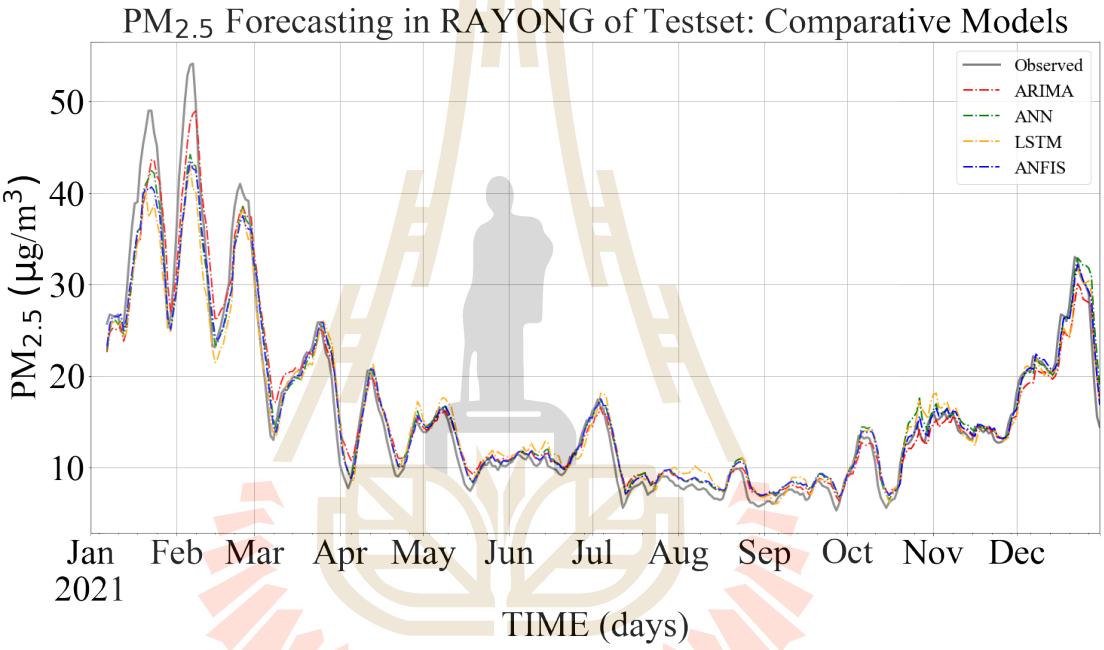
รูปที่ 4.20 กราฟเปรียบเทียบผลพยากรณ์ฝุ่น PM2.5 ล่วงหน้าของแบบจำลองอนุกรมเวลาเชิงเส้นและไม่เชิงเส้นกับชุดข้อมูลเรียนรู้

4.3 ผลประเมินประสิทธิภาพการทดลองขั้นที่สอง – พัฒนาแบบจำลองอนุกรมเวลาผสม

4.3.1 พัฒนาและประเมินประสิทธิภาพแบบจำลองผสม

การทดลองขั้นที่สองของงานวิจัยเป็นการพัฒนาแบบจำลองอนุกรมเวลาแบบผสมระหว่างเชิงเส้นและไม่เชิงเส้น โดยการนำแบบจำลองอนุกรมเวลาเชิงเส้น ARIMA ทำงานร่วมกันกับแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS เนื่องจากผลจากการทดลองขั้นที่หนึ่งบ่งชี้ว่าแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS ให้ประสิทธิภาพโดยรวมดีกว่าแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANN และ LSTM โดยเฉพาะอย่างยิ่งเมื่อพิจารณาเรื่องของเวลาในการเรียนรู้ของแบบจำลองเป็นสำคัญ

การนำแบบจำลองอนุกรมเวลาเชิงเส้น ARIMA และแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS มาทำงานร่วมกันเป็นแบบจำลองอนุกรมเวลาผสม โดยใช้หลักของการนำผลลัพธ์ที่ได้จากการพยากรณ์ของแบบจำลองอนุกรมเวลาเชิงเส้น ARIAM (ซึ่งเสมือนกับการได้ความสัมพันธ์เชิงเส้นของข้อมูลอนุกรมเวลา) มาใช้เป็นข้อมูลนำเข้าแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS เพื่อเพิ่มประสิทธิภาพผลการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ซึ่งเป็นการคำนึงถึงความสัมพันธ์แบบเชิงเส้นและไม่เชิงเส้นของข้อมูลอนุกรมเวลา แสดงดังรูปที่ 4.22

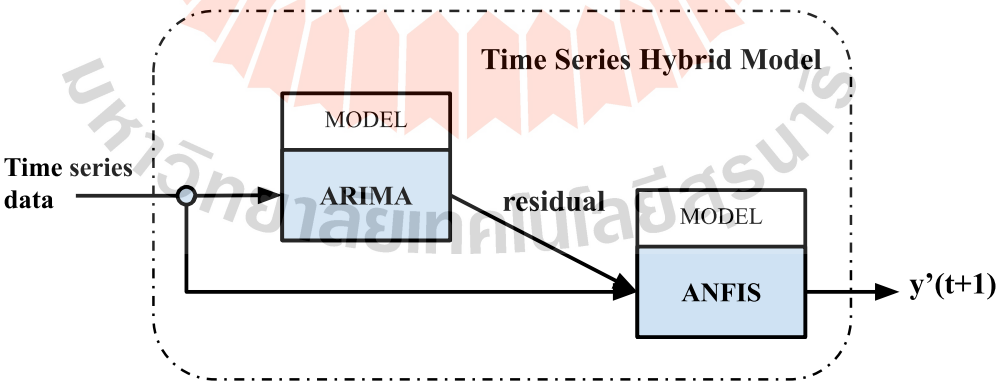


รูปที่ 4.21 กราฟเปรียบเทียบผลพยากรณ์ฝุ่น PM2.5 ล่วงหน้าของแบบจำลองอนุกรมเวลาเชิงเส้นและไม่เชิงเส้นกับชุดข้อมูลทดสอบ

การทดลองขั้นที่สอง จะทดสอบการนำพัฒนาแบบจำลองอนุกรมเวลาผสม 3 รูปแบบ ดังนี้ แบบที่ 1 ใช้ผลการพยากรณ์จากแบบจำลองอนุกรมเวลาเชิงเส้น ARIMA เป็นหนึ่งในข้อมูลนำเข้าแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS แบบที่ 2 ใช้ค่าความคลาดเคลื่อน (Error หรือ Residual) จากการพยากรณ์ของแบบจำลองอนุกรมเวลาเชิงเส้น ARIMA เป็นหนึ่งในข้อมูลนำเข้าแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS และ แบบที่ 3 ใช้ทั้งสองข้อมูลคือผลการพยากรณ์และความคลาดเคลื่อนของผลพยากรณ์จากแบบจำลองอนุกรมเวลาเชิงเส้น ARIAM เป็นหนึ่งในข้อมูลนำเข้าแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS

ตารางที่ 4.11 ตารางเปรียบเทียบประสิทธิภาพแบบจำลองอนุกรมเวลาผสม ARIMA-ANFIS

Metric	ARIMA-ANFIS		
	Type1 (y'_{ARIMA})	Type2 (error)	Type3 (y'_{ARIMA} + error)
	Train Set		
R^2	0.7604	0.8078	**0.8269**
MAE	3.7330	3.4937	**3.3720**
MAPE	22.5967	22.4431	**21.9067**
RMSE	5.5107	4.9358	**4.6846**
%RMSE	30.1123	26.9709	**25.5983**
Learning (s)	41.8711	**41.7565**	92.0241
	Test Set		
R^2	0.7601	**0.8096**	0.7740
MAE	3.9357	**3.4483**	3.5891
MAPE	26.3186	25.5670	**25.5365**
RMSE	5.8372	**5.1995**	5.6649
%RMSE	35.3921	**31.5252**	34.3473
Overfitting	5.9251	**5.3414**	20.9255

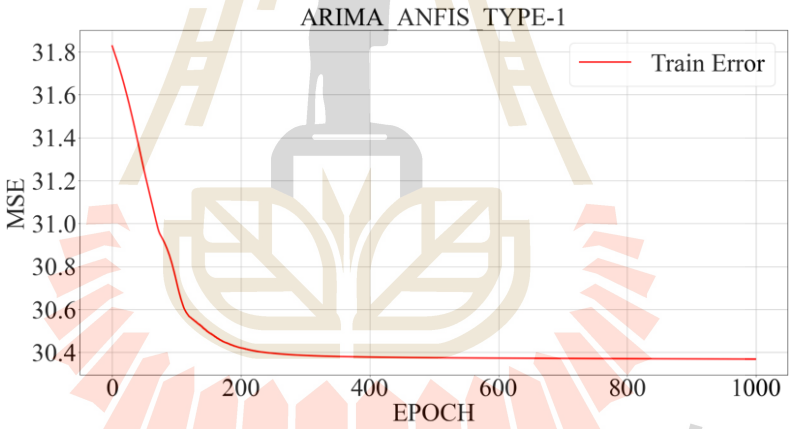


รูปที่ 4.22 แสดงโครงสร้างของแบบจำลองอนุกรมเวลาผสม

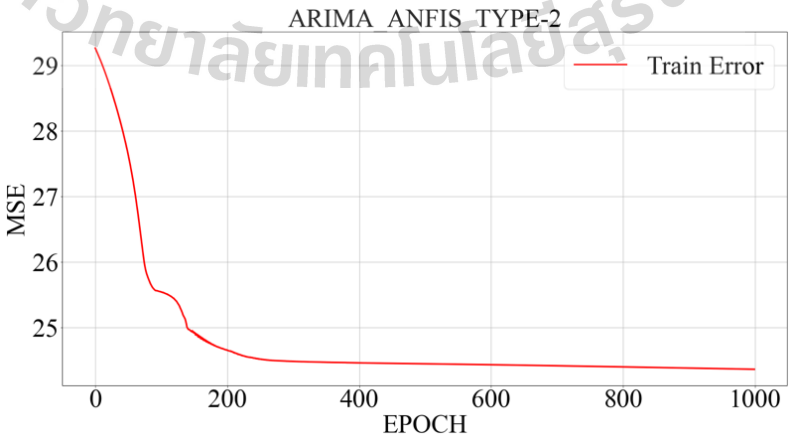
ซึ่งแบบจำลองอนุกรมเวลาเชิงเส้นและไม่เชิงเส้นต้นแบบที่นำมาใช้ร่วมกันเพื่อทำงานแบบอนุกรมเวลาผสมจะเลือกแบบจำลองอนุกรมเวลาเดี่ยวที่มีประสิทธิภาพดีที่สุดผ่านการ

ทดสอบขั้นที่หนึ่ง ผลการพัฒนาและประเมินประสิทธิภาพการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ของทั้งสามแบบจำลองอนุกรมเวลาผสม แสดงดังตารางที่ 4.11

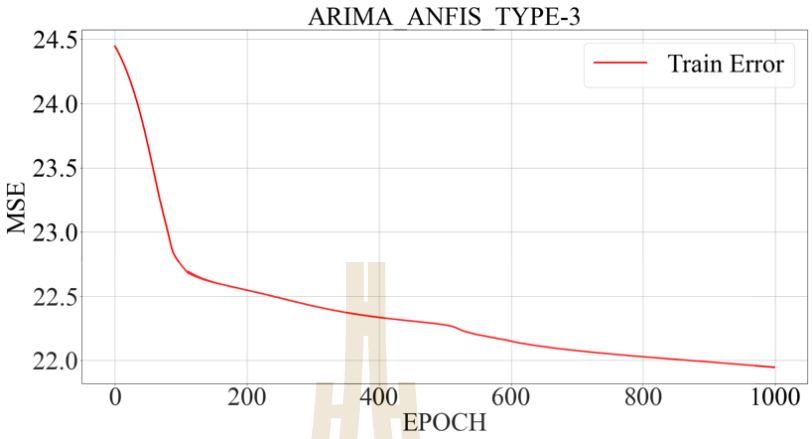
ผลการทดสอบแบบจำลองอนุกรมเวลาผสมระหว่างแบบจำลองอนุกรมเวลาเชิงเส้น ARIMA กับแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS พบว่า แบบจำลองอนุกรมเวลาผสมรูปแบบที่ 3 ให้ประสิทธิภาพสำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้าในชุดข้อมูลเรียนรู้ดีที่สุดในทุกมาตรวัด แต่พบว่าแบบจำลองอนุกรมเวลาผสมรูปแบบที่ 3 มีค่า Overfitting ที่สูงถึง 20.92% อีกทั้งยังใช้เวลาในขั้นตอนเรียนรู้สูงที่สุด คือ 92.02 วินาที ส่วนแบบจำลองอนุกรมเวลาผสมรูปแบบที่ 2 ให้ประสิทธิภาพการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ในชุดข้อมูลทดสอบดีที่สุด และยังมีค่า Overfitting ต่ำที่สุดเช่นกัน ส่วนระยะเวลาในขั้นตอนการเรียนรู้พบว่าแบบจำลองอนุกรมเวลาผสมแบบที่ 2 ใช้เวลาสำหรับเรียนรู้ข้อมูลที่ 41.75 วินาที รูปที่ 4.23 แสดงกราฟการเรียนรู้ของแบบจำลองอนุกรมเวลาผสมในแต่ละรูปแบบ ส่วนรูปที่ 4.24 และ 4.25 แสดงกราฟเปรียบเทียบผลการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ของแบบจำลองอนุกรมเวลาผสมในแต่ละรูปแบบของชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบ ตามลำดับ



(a)

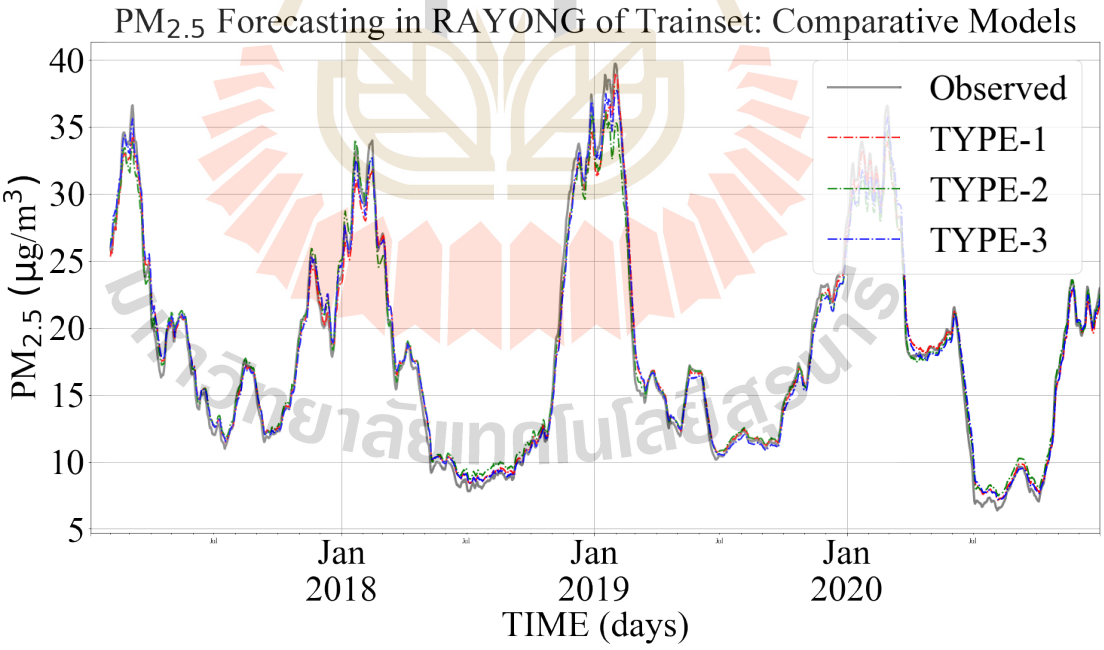


(b)

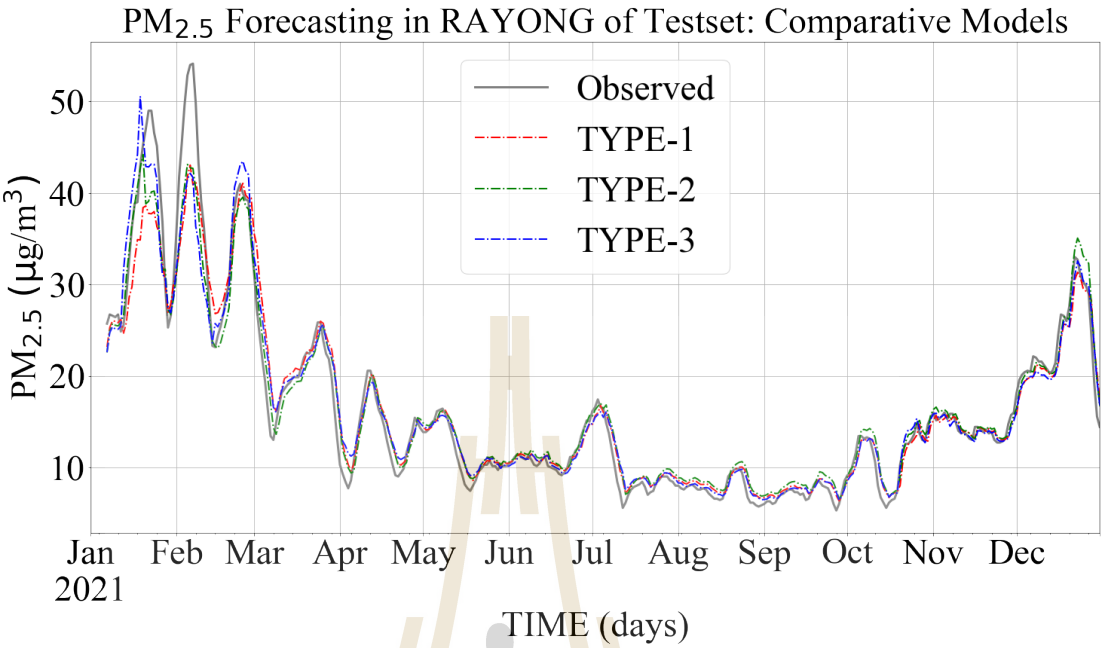


(c)

รูปที่ 4.23 แสดงกราฟการเรียนรู้ของแบบจำลองอนุกรมเวลาผสม (a) แบบที่ 1, (b) แบบที่ 2, และ (c) แบบที่ 3



รูปที่ 4.24 กราฟเปรียบเทียบผลพยากรณ์ฝุ่น PM2.5 ล่วงหน้าของแบบจำลองอนุกรมเวลาผสมกับชุดข้อมูลเรียนรู้



รูปที่ 4.25 กราฟเปรียบเทียบผลพยากรณ์ฝุ่น PM2.5 ล่วงหน้าของแบบจำลองอนุกรมเวลาผสมกับชุดข้อมูลเรียนรู้

4.3.2 ปรับปรุงประสิทธิภาพแบบจำลองอนุกรมเวลาผสม

แบบจำลองอนุกรมเวลาผสมแบบที่ 2 คือ นำค่าความคลาดเคลื่อน (Error หรือ Residual) จากผลการพยากรณ์ของแบบจำลองอนุกรมเวลาเชิงเส้น ARIMA เป็นหนึ่งในข้อมูลนำเข้าแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS มาปรับปรุงประสิทธิภาพต่อด้วยการใช้ Global Optimization เพื่อปรับปรุงค่าพารามิเตอร์ Antecedent และ Consequent สำหรับงานวิจัยนี้เลือกใช้ Metaheuristic Algorithm ที่เป็นแบบ Population Based ได้แก่ GA และ PSO โดยการกำหนดค่าพารามิเตอร์ของ GA และ PSO Algorithm แสดงดังตารางที่ 4.12

ตารางที่ 4.12 แสดงค่าพารามิเตอร์ของ Global Optimization

GA		PSO	
Population Size	128, 256, 512, 1024	**Swarm Size**	128, 256, 512, 1024
Selection	Roulette	**Self-adjust**	1
Crossover	Two-point	**Social-adjust**	1
Mutation Rate	0.1	**Inertia Rate**	[0.1, 1.1]
Epoch	1000	**Epoch**	1000

ผลประเมินประสิทธิภาพจากการทดลองปรับปรุงค่าพารามิเตอร์ของแบบจำลองอนุกรมเวลาผสม ด้วยกระบวนการ Global Optimization พบกว่าแบบจำลองอนุกรมเวลาผสมมีประสิทธิภาพการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ดีขึ้นประมาณ 7.58% โดยคำนวณผ่านสมการที่ (4-2) (รติพร จันทร์กลั่น, 2560: 72) เมื่อทดสอบกับชุดข้อมูลทดสอบแต่ต้องแลกกับเวลาที่ใช้ในขั้นตอนการเรียนรู้ที่สูงขึ้น โดยที่แบบจำลองอนุกรมเวลาผสมที่ปรับปรุงประสิทธิภาพด้วย PSO อัลกอริทึม ที่กำหนดจำนวน Swarm เท่ากับ 256 ให้ประสิทธิภาพโดยรวมดีที่สุดผ่านมาตรวัด R^2, MAE, MAPE, RMSE, และ %RMSE กับชุดข้อมูลเรียนรู้ ดังนี้ 0.79, 3.54, 22.64, 5.11, และ 27.94 ตามลำดับ ส่วนกับชุดข้อมูลทดสอบให้ค่าประเมินประสิทธิภาพผ่านมาตรวัด R^2, MAE, MAPE, RMSE, และ %RMSE ดังนี้ 0.83, 3.38, 25.19, 4.80, และ 28.74 ตามลำดับ สำหรับค่า Overfitting มีค่าเท่ากับ 4.08% ส่วนเวลาที่ใช้สำหรับการเรียนรู้ของแบบจำลองอนุกรมเวลาผสมที่ปรับปรุงประสิทธิภาพด้วย PSO อัลกอริทึม ที่กำหนดจำนวน Swarm เท่ากับ 256 ประมาณ 72 นาที เปรียบเทียบดังตารางที่ 4.13 และรูปที่ 4.26 แสดงกราฟการเรียนรู้ของแบบจำลองอนุกรมเวลาผสมที่ปรับปรุงด้วย GA และ PSO อัลกอริทึม จากรูปของกราฟการเรียนรู้แสดงให้เห็นว่า เมื่อกำหนดขนาดของคำตอบ คือ Population และ Swarm มีค่าต่ำกว่า 256 จะทำให้ผลการปรับปรุงประสิทธิภาพแบบจำลองอนุกรมเวลาผสมอาจจะไม่ได้ดีที่สุด เมื่อกำหนดจำนวนรอบการเรียนรู้ที่ 1,000 รอบ

$$Speedup = \frac{|Model_{std} - Model_{imp}|}{Model_{std}} \times 100 \quad (4\text{-}2)$$

โดยที่ $Model_{std}$ คือแบบจำลองอนุกรมเวลาก่อนปรับปรุง $Model_{imp}$ และ คือแบบจำลองอนุกรมเวลาหลังการปรับปรุง

รูปที่ 4.27 และ 4.28 แสดงกราฟเปรียบเทียบผลพยากรณ์ฝุ่น PM2.5 ล่วงหน้าของแบบจำลองอนุกรมเวลาผสมที่ปรับปรุงด้วย GA และ PSO โดยกำหนดขนาดของคำตอบเท่ากับ 256 กับชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบ ตามลำดับ

4.4 ผลประเมินประสิทธิภาพการพยากรณ์ฝุ่น PM2.5 ในรูปแบบดัชนีความรุนแรง

การรายงานค่าฝุ่น PM2.5 ส่วนใหญ่ หน่วยงานที่รับผิดชอบหรือหน่วยงานที่เกี่ยวข้อง เช่น กรมควบคุมมลพิษ กรมอุตุนิยมวิทยา หรือหน่วยงานสากลอย่าง World Air Quality Index Project จะรายงานค่าฝุ่น PM2.5 เป็น ค่าดัชนีความรุนแรง เพราะเป็นการสื่อความหมายที่ทำให้ประชาชนเข้าใจได้ง่ายกว่าการรายงานเป็นค่าตัวเลข ดังนั้น สำหรับการรายงานฝุ่น PM2.5 ตามดัชนีความรุนแรงจะต้องเปรียบเทียบค่าฝุ่น PM2.5 ที่ตรวจวัดได้ผ่านตารางแสดงระดับความรุนแรงของค่าฝุ่น PM2.5 ดังตารางที่ 4.14

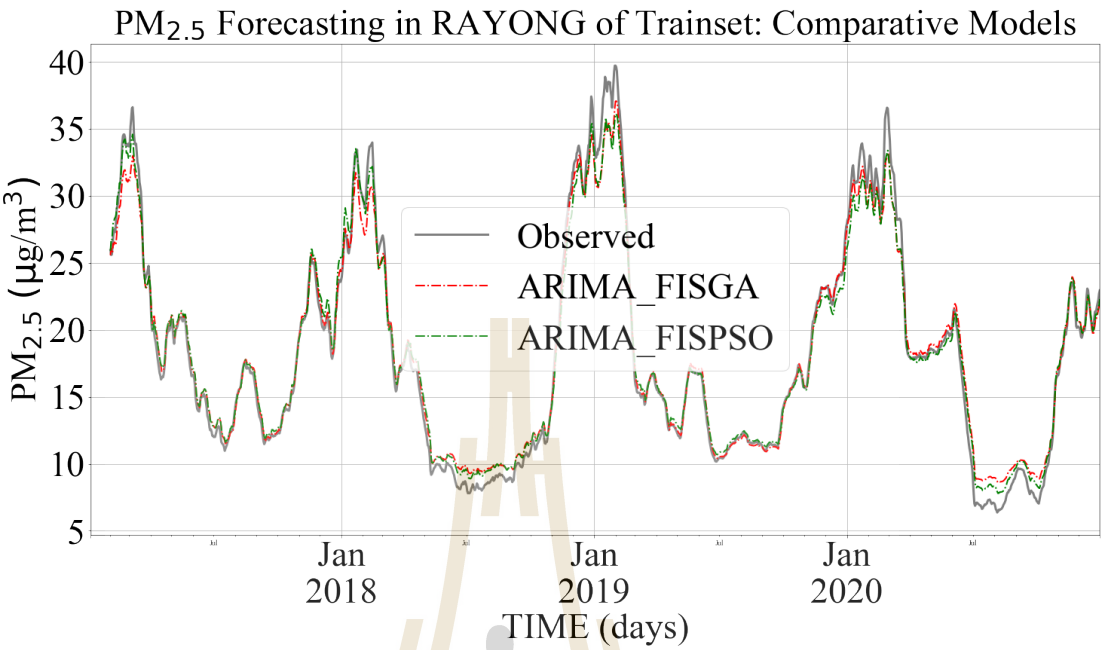
ตารางที่ 4.13 ตารางเปรียบเทียบประสิทธิภาพแบบจำลองอนุกรมเวลาผสมที่ปรับปรุง

Optimized	GA				PSO			
Size	128	256	512	1024	128	256	512	1024
				Train Set				
R^2	0.7688	0.7851	0.7614	0.7674	0.7908	**0.7937**	0.7667	0.7935
MAE	3.6737	3.6419	3.8121	3.7857	3.6585	**3.5493**	3.7738	3.5861
MAPE	22.784	23.314	23.832	24.2581	23.812	**22.649**	23.576	23.204
RMSE	5.4137	5.2194	5.5001	5.4303	5.1502	**5.1142**	5.4381	5.1162
%RMSE	29.582	28.520	30.054	29.6731	28.142	**27.945**	29.715	27.956
Learning (m)	**36**	68	135	282	40	72	80	296
				Test Set				
R^2	0.8177	0.8212	0.7668	0.7910	0.8235	**0.8306**	0.7712	0.8174
MAE	3.5944	3.6219	3.9672	3.8709	3.6044	**3.3898**	3.8853	3.6069
MAPE	26.405	27.667	29.013	29.0876	28.300	**25.195**	28.412	27.272
RMSE	5.0883	5.0386	5.7549	5.4483	5.0066	**4.8053**	5.7006	5.0925
%RMSE	30.851	30.549	34.892	33.0338	30.356	**28.741**	34.563	30.877
Overfitting	6.0100	3.4650	4.6315	**0.3314**	2.7878	4.0847	4.1956	0.4624

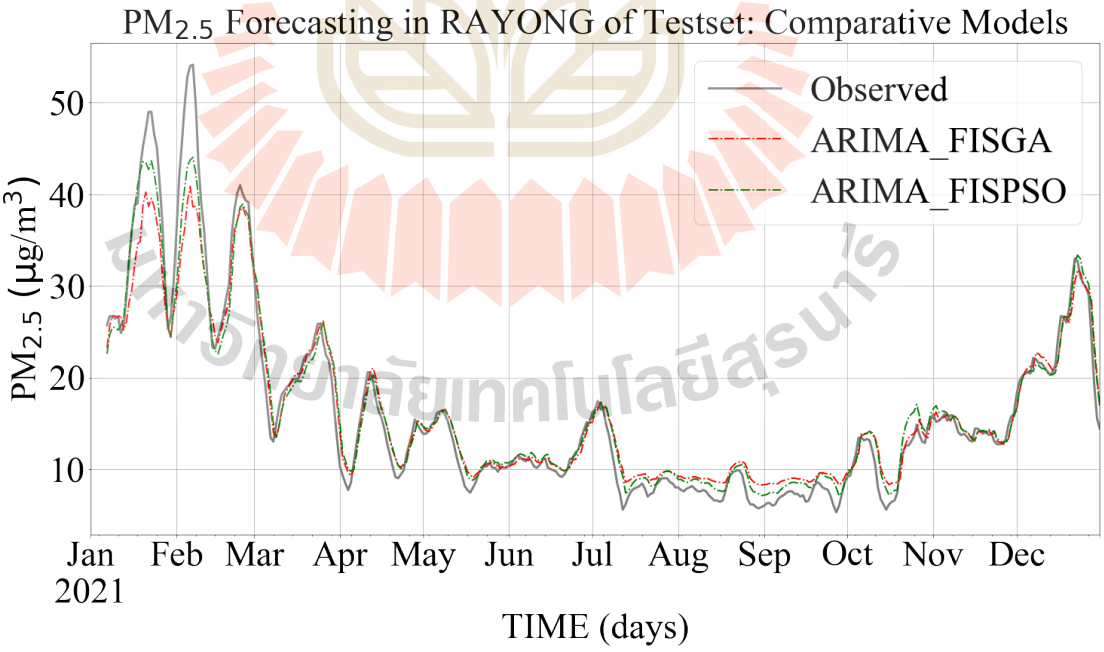
GA (Population Size: 128, 256, 512, and 1024)

PSO (Swarm Size: 128, 256, 512, and 1024)

รูปที่ 4.26 แสดงกราฟการเรียนรู้ของแบบจำลองอนุกรมเวลาผสมที่ปรับปรุงประสิทธิภาพด้วย GA และ PSO อัลกอริทึม



รูปที่ 4.27 กราฟเปรียบเทียบผลพยากรณ์ฝุ่น PM2.5 ล่วงหน้าของแบบจำลองอนุกรมเวลาผสมที่ปรับปรุงด้วย Global Optimization กับชุดข้อมูลเรียนรู้



รูปที่ 4.28 กราฟเปรียบเทียบผลพยากรณ์ฝุ่น PM2.5 ล่วงหน้าของแบบจำลองอนุกรมเวลาผสมที่ปรับปรุงด้วย Global Optimization กับชุดข้อมูลทดสอบ

ตารางที่ 4.14 แสดงดัชนีระดับความรุนแรงของฝุ่น PM2.5 (เฉลี่ย 24 ชั่วโมงต่อเนื่อง)(1)

ระดับที่	PM2.5	สีที่ใช้	ความหมาย
1	0 - 25	ฟ้า	คุณภาพอากาศดีมาก
2	26 - 37	เขียว	คุณภาพอากาศดี
3	38 - 50	เหลือง	ปานกลาง
4	51 - 90	ส้ม	เริ่มมีผลกระทบต่อสุขภาพ
5	> 91	แดง	มีผลกระทบต่อสุขภาพ

(1)<u>หมายเหตุ</u> ข้อมูลจากกรมควบคุมมลพิษ (http://air4thai.pcd.go.th/webV3/#/AQIInfo)

```
PM2.5 Index Y2021  (365,)
 [1 1 2 1 2 2 2 1 1 2 2 1 2 4 3 3 3 2 3 4 4 4 3 2 1 1 1 1 2 4 4 4 4 3 3 4 4
 4 1 1 1 2 1 2 1 1 1 2 3 2 3 4 3 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 2 2 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 2 3 2 1 1 2 3 3 2 1 1 1 1 1 1 1 1]

PM2.5 Forecast Index Y2021  (365,)
 [1 1 1 2 1 2 2 2 1 1 1 2 2 4 4 3 2 3 3 3 3 4 3 2 2 1 1 1 1 3 3 3 3 3 3 3 3
 3 2 2 1 1 2 1 2 1 1 1 2 2 2 4 4 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 2 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 1 3 3 3 2 1 1 1 1 1 1 1]
```

รูปที่ 4.29 แสดงข้อมูลอนุกรมเวลาฝุ่น PM2.5 ในรูปแบบของดัชนีความรุนแรง

การประเมินประสิทธิภาพการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ในรูปแบบดัชนีความรุนแรงของงานวิจัยนี้ จะแปลงข้อมูลค่าฝุ่น PM2.5 ที่พยากรณ์ได้เป็นค่าดัชนีความรุนแรงตามตารางที่ 4.14 จากนั้นเปรียบเทียบกับค่าจริงของฝุ่น PM2.5 ที่แปลงข้อมูลเป็นค่าดัชนีความรุนแรง เช่นกัน เพื่อคำนวณเป็นค่าความแม่นยำของผลการพยากรณ์

ผลการประเมินจากการนำแบบจำลองอนุกรมเวลาผสมที่ปรับปรุงประสิทธิภาพด้วย PSO อัลกอริทึมมาพยากรณ์ฝุ่น PM2.5 ล่วงหน้าในชุดข้อมูลทดสอบ (ของข้อมูลฝุ่น PM2.5 ปี ค.ศ. 2021)

ที่พิจารณาเป็นดัชนีความรุนแรง มีความถูกต้องอยู่ที่ 87.39% รูปที่ 4.29 แสดงผลการแปลงข้อมูลอนุกรมเวลาฝุ่น PM2.5 ในปี ค.ศ. 2021 จากข้อมูลค่าฝุ่น PM2.5 เป็นข้อมูลดัชนีระดับความรุนแรง

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

ปัญหามลพิษทางอากาศเป็นปัญหาที่มีความสำคัญ เพราะส่งผลกระทบโดยตรงต่อคุณภาพชีวิตของประชาชนโดยกว้าง โดยเฉพาะมลพิษทางอากาศในเขตเมืองใหญ่ และเขตเมืองอุตสาหกรรม โดยเฉพาะอย่างยิ่งปัญหาฝุ่น PM2.5 ซึ่งมีความอันตรายต่อระบบทางเดินหายใจของมนุษย์และเป็นสาเหตุให้เกิดการเสียชีวิตก่อนวัยอันควรจากโรคร้ายต่าง ๆ เช่น โรคมะเร็งปอด โรคหัวใจล้มเหลวเฉียบพลัน หรือโรคหลอดเลือดสมอง เป็นต้น ดังนั้น การรับทราบข้อมูลของฝุ่น PM2.5 แบบทันท่วงทีและแบบล่วงหน้า นั้นเป็นสิ่งที่สำคัญต่อการบริหารจัดการกับปัญหาฝุ่น PM2.5 อย่างยิ่ง งานวิจัยนี้จึงมุ่งพัฒนาแบบจำลองที่มีประสิทธิภาพสำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้าด้วยกระบวนการเรียนรู้ของเครื่อง โดยการนำเสนออัลกอริทึมแบบใหม่ที่เป็นการผสมระหว่างอัลกอริทึมแบบเชิงเส้น Autoregressive Integrated Moving Average: ARIMA กับอัลกอริทึมแบบไม่เชิงเส้น Adaptive Neuro-Fuzzy Inference System: ANFIS โดยใช้ชุดข้อมูลทดสอบจริงของค่าฝุ่น PM2.5 จังหวัดระยอง

5.1 สรุปขั้นตอนการดำเนินงานวิจัย

งานวิจัยนี้มีความประสงค์ที่จะช่วยสร้างเครื่องมือแบบจำลองทางคณิตศาสตร์สำหรับใช้พยากรณ์ค่าฝุ่น PM2.5 ล่วงหน้า เพื่อนำข้อมูลฝุ่น PM2.5 จากผลพยากรณ์ไปใช้ประโยชน์ต่อการรับมือและวางแผนจัดการปัญหาฝุ่น PM2.5 ต่อไป โดยงานวิจัยนี้ได้พัฒนาแบบจำลองอนุกรมเวลาผสมระหว่างแบบจำลองอนุกรมเวลาเชิงเส้น ARIMA กับแบบจำลองไม่เชิงเส้น ANFIS ผ่านกระบวนการเรียนรู้ของเครื่อง โดยได้ดำเนินงานวิจัย เป็นลำดับขั้นตอนดังนี้

1) ศึกษาปัญหามลพิษทางอากาศและผลกระทบ โดยเฉพาะอย่างยิ่งปัญหามลพิษทางอากาศที่เกิดจากฝุ่น PM2.5 ทั้งที่เป็นข้อมูลในประเทศไทย และต่างประเทศ

2) ปริทัศน์วรรณกรรมที่เกี่ยวข้องกับการพยากรณ์ข้อมูลอนุกรมเวลา โดยเฉพาะอย่างยิ่งวรรณกรรมที่เกี่ยวข้องกับการพยากรณ์ฝุ่น PM10 และ PM2.5 ล่วงหน้า

3) ศึกษาการพัฒนาแบบจำลองอนุกรมเวลาผ่านกระบวนการเรียนรู้ของเครื่อง ด้วยอัลกอริทึมแบบต่าง ๆ ได้แก่ แบบจำลองอนุกรมเวลาเชิงเส้นด้วยอัลกอริทึม ARIMA และแบบจำลองอนุกรมเวลาไม่เชิงเส้นด้วยอัลกอริทึมกลุ่มโครงข่ายประสาทเทียม คือ Artificial Neural Network: ANN, Long Short-Term Memory: LSTM, และ ANFIS

4) ออกแบบการทดสอบและพัฒนาแบบจำลองอนุกรมเวลาผสมสำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ด้วยชุดข้อมูล PM2.5 ของจังหวัดระยอง จากนั้นปรับปรุงแบบจำลองอนุกรมเวลาผสมด้วยการปรับค่า Hyperparameter ผ่านการทำ Optimization ด้วย Metaheuristic Algorithm – Genetic Algorithm: GA และ Particle swarm optimization: PSO

5) ทดสอบประเมินประสิทธิภาพของแบบจำลองอนุกรมเวลาผสม ด้วยมาตรวัดสัมประสิทธิ์การตัดสินใจ (Coefficient of Determination: R^2), มาตรวัดความคลาดเคลื่อน Mean Absolute Error: MAE, มาตรวัดความคลาดเคลื่อน Mean Absolute Percentage Error: MAPE, มาตรวัดความคลาดเคลื่อน Root Mean Square Error: RMSE, และมาตรวัดความคลาดเคลื่อน Percentage Root Mean Square Error: %RMSE นอกจากนี้ ยังได้นำค่า Overfitting และระยะเวลาในขั้นตอนการเรียนรู้ของแบบจำลองอนุกรมเวลามาพิจารณาร่วมด้วย

5.2 สรุปผลการวิจัย

การศึกษาและพัฒนาแบบจำลองอนุกรมเวลาผสมระหว่างแบบจำลองอนุกรมเวลาเชิงเส้นและไม่เชิงเส้นสำหรับพยากรณ์ฝุ่น PM2.5 โดยส่วนแรกเป็นการเตรียมข้อมูลอนุกรมเวลาฝุ่น PM2.5 ซึ่งงานวิจัยนี้ได้เลือกใช้วิธีการเติมข้อมูล Missing Value ด้วยเทคนิค Temporal and Spatial Average Value: TSA จากผลการประเมินประสิทธิภาพ พบกว่ามีประสิทธิภาพดีกว่าการเติมข้อมูล Missing Value ด้วยเทคนิคทางสถิติ ได้แก่ เติมข้อมูลด้วยค่าเฉลี่ย (Mean) เติมข้อมูลด้วยค่ามัธยฐาน (Median) และเติมข้อมูลด้วยค่านิยม (Most-frequency) โดยให้ประสิทธิภาพดีกว่าอย่างน้อยที่ 20.16 MSE

สำหรับการพัฒนาแบบจำลองอนุกรมเวลาเดี่ยว - พบว่าค่าพารามิเตอร์ p, d, และ q ที่ดีที่สุดของแบบจำลองอนุกรมเวลา ARIMA คือ ARIMA(2, 0, 2) ส่วนในกลุ่มแบบจำลองอนุกรมเวลาไม่เชิงเส้น จากผลการทดลองพบว่าแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANN, LSTM, และ ANFIS ที่ให้ผลประเมินประสิทธิภาพดีที่สุด คือ ANN_32_2, LSTM_4_4, และ ANFIS_2_GBELLMF ตามลำดับ โดยที่แบบจำลองอนุกรมเวลาไม่เชิงเส้นที่มีประสิทธิภาพโดยรวมดีที่สุด สำหรับพยากรณ์ฝุ่น PM2.5 คือ ANFIS_2_GBELLMF โดยเฉพาะอย่างยิ่งประสิทธิภาพด้านเวลาการเรียนรู้ข้อมูล แบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS_2_GBELLMF ใช้เวลาในระดับวินาทีเท่านั้น ต่างจากแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANN_32_2 และ LSTM_4_4 ซึ่งใช้เวลาในระดับนาทีและชั่วโมงสำหรับเรียนรู้ข้อมูล ตามลำดับ

การพัฒนาแบบจำลองอนุกรมเวลาผสมด้วยแบบจำลองอนุกรมเวลาเชิงเส้นและไม่เชิงเส้น – งานวิจัยนี้ได้พัฒนาแบบจำลองอนุกรมเวลาผสมด้วยแบบจำลองอนุกรมเวลาเชิงเส้น ARIMA(2, 0, 2) กับแบบจำลองอนุกรมเวลาไม่เชิงเส้น ANFIS_2_GBELLMF โดยเรียกแบบจำลองอนุกรมเวลาผสมนี้

ว่า ARIMA-ANFIS ผลจากการประเมินประสิทธิภาพพบว่า แบบจำลองอนุกรมเวลาผสม ARIMA-ANFIS ที่ใช้ค่า Residual จากผลการพยากรณ์ของอนุกรมเวลาเชิงเส้น ARIMA เป็นหนึ่งในข้อมูลข้าวของอนุกรมเวลาไม่เชิงเส้น ANFIS ให้ประสิทธิภาพดีที่สุด (ในบทที่ 4 จะอ้างอิงตามอนุกรมเวลาผสม ARIMA-ANFIS แบบที่ 2) โดยที่แบบจำลองอนุกรมเวลาผสม ARIMA-ANFIS จะมีประสิทธิภาพดีกว่าแบบจำลองอนุกรมเวลาเดี่ยวอย่างน้อยที่สุด 4.08%

หลังจากนั้นงานวิจัยนี้ได้ปรับปรุงประสิทธิภาพแบบจำลองอนุกรมเวลาผสม ARIMA-ANFIS ผ่านกระบวนการ Hyperparameter Optimization ด้วยอัลกอริทึม GA และ PSO ผลการปรับปรุงประสิทธิภาพพบกว่าแบบจำลองอนุกรมเวลาผสม ARIMA-ANFIS ที่ปรับปรุงด้วยอัลกอริทึม PSO ให้ประสิทธิภาพเพิ่มขึ้นโดยรวมดีที่สุดที่ 7.85% แต่ต้องแลกกับเวลาที่ใช้เรียนรู้ข้อมูลของแบบจำลองอนุกรมเวลาผสมหลายเท่าตัว

สำหรับการประเมินผลสุดท้ายของงานวิจัยนี้ เป็นการประเมินผลผ่านการพยากรณ์ฝุ่น PM2.5 ล่วงหน้าด้วยค่าดัชนีความรุนแรง ซึ่งเป็นการรายงานผลในรูปแบบมาตราฐานของหน่วยงานที่รับผิดชอบเกี่ยวกับมลพิษทางอากาศ ทั้งในและต่างประเทศ จากผลการประเมินประสิทธิภาพแบบจำลองอนุกรมเวลาผสม ARIMA-ANFIS (PSO) ในการพยากรณ์ดัชนีความรุนแรงของฝุ่น PM2.5 ล่วงหน้า พบว่ามีความถูกต้องเฉลี่ยที่ 87.39%

5.3 ปัญหาและข้อเสนอแนะ

การพัฒนาแบบจำลองอนุกรมเวลาสำหรับพยากรณ์ข้อมูลล่วงหน้า ความสมบูรณ์ของชุดข้อมูลอนุกรมเวลานั้นมีความสำคัญมากต่อประสิทธิภาพของแบบจำลองอนุกรมเวลา สำหรับงานวิจัยนี้พบปัญหาเรื่องความไม่สมบูรณ์ของข้อมูลที่มีข้อมูล Missing Value ประมาณร้อยละ 6.4 และ Missing Value จะเป็นช่วงติดต่อกัน จำเป็นที่จะต้องให้ความสำคัญ ซึ่งงานวิจัยนี้ได้ใช้วิธีการเติมข้อมูลด้วยเทคนิค TSA

แบบจำลองอนุกรมเวลาเชิงเส้น ARIMA เป็นแบบจำลองอนุกรมเวลาทางสถิติที่มีประสิทธิภาพสูง ให้ผลการพยากรณ์ที่แม่นยำไม่ต่างจากแบบจำลองอนุกรมเวลาอื่น ๆ แต่มีจุดด้อยเรื่องการใช้งาน เนื่องจากเป็นแบบจำลองประเภท Parametric Model ที่จะต้องกำหนดค่าพารามิเตอร์ที่สำคัญ 3 ค่า ได้แก่ p, d, และ q ปกติการหาค่าที่เหมาะสมของชุดพารามิเตอร์ดังกล่าว จะต้องใช้กระบวนการวิเคราะห์ข้อมูลผ่าน Auto Correlation Function: ACF และ Partial Autocorrelation Function: PACF ร่วมถึงต้องวิเคราะห์ความเป็น Stationary ของชุดข้อมูลอนุกรมเวลาอีกด้วย สำหรับงานวิจัยนี้ได้เลือกใช้วิธีการหาค่าพารามิเตอร์ p, d, และ q ของแบบจำลองอนุกรมเวลา ARIMA ผ่าน Auto Function ของภาษา Python ซึ่งอาจจะส่งผลให้ไม่ได้ค่าพารามิเตอร์ p, d, และ q ที่เหมาะสมที่สุด แต่ส่งผลให้มีความสะดวกและใช้เวลาดำเนินงานเร็วขึ้น

จากผลการพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ของแบบจำลองอนุกรมเวลาผสม ARIMA-ANFIS (PSO) กับชุดข้อมูลทดสอบ ซึ่งมีช่วงเวลา 1 ปี พบว่าแบบจำลองอนุกรมเวลาที่นำเสนอให้ผลการพยากรณ์ที่ดีในช่วงฤดูฝน ซึ่งเป็นช่วงเวลาที่มีค่าปริมาณฝุ่น PM2.5 ต่ำและมีความผันผวนน้อย แต่เมื่อแบบจำลองอนุกรมเวลาที่นำเสนอพยากรณ์ฝุ่น PM2.5 ในหน้าแล้ง ซึ่งเป็นช่วงเวลาที่ปริมาณฝุ่น PM2.5 มีค่าสูงและค่อนข้างมีความผันผวน กลับพบค่าความคลาดเคลื่อนที่สูงขึ้น แสดงให้เห็นว่าแบบจำลองอนุกรมเวลาผสม ARIMA-ANFIS (PSO) ยังมีจุดที่ต้องปรับปรุงอยู่ โดยทางผู้วิจัยแนะนำแนวทางโดยการเพิ่มข้อมูลเรียนรู้ที่เกี่ยวข้องกับช่วงเวลา เช่น ข้อมูลฤดูกาล หรือข้อมูลของเดือน เป็นต้น อีกแนวทางหนึ่ง คือ พัฒนาแบบจำลองอนุกรมเวลาผสม ARIMA-ANFIS (PSO) สองแบบจำลอง โดยแบบจำลองอนุกรมเวลาผสมที่หนึ่ง สำหรับพยากรณ์ฝุ่น PM2.5 ช่วงที่มีค่าสูง (ช่วงหน้าแล้ง) และแบบจำลองอนุกรมเวลาผสมที่สอง สำหรับพยากรณ์ฝุ่น PM2.5 ช่วงที่มีปริมาณค่าต่ำ (ช่วงหน้าฝน)

งานศึกษาวิจัยในอนาคตจะศึกษาและนำกระบวนการสำหรับการเตรียมข้อมูลอนุกรมเวลาฝุ่น PM2.5 ที่มีประสิทธิภาพเพื่อขยายงานให้ครอบคุมจังหวัดอื่น ๆ ของประเทศไทย

รายการอ้างอิง

กรมควบคุมมลพิษ, (2020). โครงการศึกษาแหล่งกำเนิดและแนวทางการ จัดการฝุ่นละอองขนาดไม่เกิน 2.5 ไมครอน ในพื้นที่กรุงเทพและปริมณฑล. *กรมควบคุมมลพิษ กระทรวงทรัพยากรธรรมชาติและสิ่งแวดล้อม*, เข้าถึงจาก https://www.pcd.go.th/wp-content/uploads/2020/06/pcdnew-2020-06-05_02-34-12_147817.pdf

รติพร จันทร์กลั่น. (2560). **การสร้างแบบจำลองด้วยเทคนิคการเรียนรู้ของเครื่องเพื่อคาดการณ์ปริมาณน้ำท่า**. (วิทยานิพนธ์ปริญญาดุษฎีบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์). นครราชสีมา: มหาวิทยาลัยเทคโนโลยีสุรนารี

สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ, (2021). คพ. ผนึก สวทช. ใช้ 'ซูเปอร์คอมพิวเตอร์'คาดการณ์ฝุ่น PM 2.5 รู้ล่วงหน้า 3 วัน. *สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ*, เข้าถึงจากhttps://www.nstda.or.th/home/news_post/super-computer

Agatonovic-Kustrin, S., & Beresford, R. (2000). Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis*, *22*(5), 717–727.

Amnuaylojaroen, T. (2022). Prediction of PM2.5 in an Urban Area of Northern Thailand Using Multivariate Linear Regression Model. *Advances in Meteorology*, *2022*, 1–9.

Bălă, G.-P., Râjnoveanu, R.-M., Tudorache, E., Motișan, R., & Oancea, C. (2021). Air pollution exposure—The (in)visible risk factor for respiratory diseases. *Environmental Science and Pollution Research*, *28*(16), 19615–19628.

Chen, C. W. S., & Chiu, L. M. (2021). Ordinal Time Series Forecasting of the Air Quality Index. *Entropy*, *23*(9), 1167.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling* (arXiv:1412.3555). arXiv.

Du, S., Li, T., Yang, Y., & Horng, S.-J. (2021). Deep Air Quality Forecasting Using Hybrid Deep Learning Framework. *IEEE Transactions on Knowledge and Data Engineering*, *33*(6), 2412–2424.

Espinosa, R., Palma, J., Jiménez, F., Kamińska, J., Sciavicco, G., & Lucena-Sánchez, E. (2021). A time series forecasting based multi-criteria methodology for air quality prediction. *Applied Soft Computing*, *113*, 107850.

Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor. (2nd Edition, MIT Press, 1992.)

Huang, C.-J., & Kuo, P.-H. (2018). A Deep CNN-LSTM Model for Particulate Matter (PM2.5) Forecasting in Smart Cities. *Sensors*, *18*(7), 2220.

IQAir. (2021). World Air Quality Report: Region & City PM2.5 Ranking. *IQAir*, Available From https://www.iqair.com/world-most-polluted-cities/world-air-quality-report-2021-en.pdf

Jang, J.-S. R. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, *23*(3), 665–685.

Kanabkaew, T. (2013). Prediction of Hourly Particulate Matter Concentrations in Chiangmai, Thailand Using MODIS Aerosol Optical Depth and Ground-Based Meteorological Data. *2*, *6*, EnvironmentAsia.

Katoch, S., Chauhan, S.S., & Kumar, V. (2021) A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80, 8091–8126.

Kennedy, J. & Eberhart, R. C. (1995). Particle Swarm Optimization. In IEEE International Conference on Neural Network, 1942-1948.

Li, S., Xie, G., Ren, J., Guo, L., Yang, Y., & Xu, X. (2020). Urban PM2.5 Concentration Prediction via Attention-Based CNN–LSTM. *Applied Sciences*, *10*(6), 1953.

Mao, Y., & Lee, S. (2019). Deep Convolutional Neural Network for Air Quality Prediction. *Journal of Physics: Conference Series*, *1302*(3), 032046.

Qi, Z., Wang, T., Song, G., Hu, W., Li, X., & Zhang, Z. (2018). Deep Air Learning: Interpolation, Prediction, and Feature Analysis of Fine-Grained Air Quality. *IEEE Transactions on Knowledge and Data Engineering*, *30*(12), 2285–2297.

Rajwar, K., Deep, K., & Das, S. (2023). An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges. Artif Intell Rev, 1–71.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533–536.

Srijiranon, K., & Eiamkanitchat, N. (2018). Collective Neural Networks System for PM_{10} Classification in the North of Thailand. *2018 22nd International Computer Science and Engineering Conference (ICSEC)*, 1–4.

Srikamdee, S., & Onpans, J. (2019). Forecasting Daily Air Quality in Northern Thailand Using Machine Learning Techniques. *2019 4th International Conference on Information Technology (InCIT)*, 259–263.

Szandała, T. (2021). Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks. In A. K. Bhoi, P. K. Mallick, C.-M. Liu, & V. E. Balas (Eds.), *Bio-inspired Neurocomputing* (Vol. 903, pp. 203–224). Springer Singapore.

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-15*(1), 116–132.

Tongprasert, P., & Ongsomwang, S. (2022). A Suitable Model for Spatiotemporal Particulate Matter Concentration Prediction in Rural and Urban Landscapes, Thailand. *Atmosphere*, *13*(6), 904.

Wang, D., Tan, D. & Liu, L. (2018). Particle swarm optimization algorithm: an overview. *Soft Comput*, 22, 387–408.

Wongsathan, R. (n.d.). *The Hybrid Neural Networks-ARIMA/X Models and ANFIS Model for PM-10 Forecasting: A Case Study of Chiang Mai, Thailand's High Season*.

Wongsathan, R. (2018). Improvement of PM-10 Forecast Using ANFIS Model with an Integrated Hotspots. *Science & Technology Asia*, *23*, 6271.

World Health Organization. (2016). Ambient Air Pollution: A Global Assessment of Exposure and Burden of Disease. *World Health Organization*, Available from https://apps.who.int/iris/handle/10665/250141.

Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control*, 8, 338-353.

Zamani Joharestani, M., Cao, C., Ni, X., Bashir, B., & Talebiesfandarani, S. (2019). PM2.5 Prediction Based on Random Forest, XGBoost, and Deep Learning Using Multisource Remote Sensing Data. *Atmosphere*, *10*(7), 373.

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, *50*, 159–175.

Zhang, Q., Li, V. O., Lam, J. C., & Han, Y. (n.d.). *Deep-AIR: A Hybrid CNN-LSTM Framework for Fine-Grained Air Pollution Forecast*.

ภาคผนวก ก
รหัสต้นฉบับของโปรแกรม

1. โปรแกรมสำหรับพัฒนาแบบจำลองอนุกรมเวลา ARIMA สำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ด้วยภาษา Python

```
################################## BEGIN ARIMA ###################################
from IPython.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams["font.family"] = "Times New Roman"

method = "ARIMA"
city = "RAYONG"
pm_type = "PM2dot5"

## Load dataset
working_path = "/home/anupong/works/2022-trimester-0265/thesis/"
file_name = pm_type + "_" + city + "_" + method + "_DATASET.csv"
data_path = working_path + "dataset/" + pm_type + "/csv/use/" + city + "/" + method + "/"
pm_data = data_path + file_name
missing_values = ["n/a", "na", "--", "null", "?"]
data = pd.read_csv(pm_data, na_values = missing_values)
print("\nMissing values: \n", data.isnull().sum() + data.isna().sum()) #Check missing values
print("\nData Size: ", len(data))

## Set time index to dataframe
data["Date"] = pd.date_range(start = "1/1/2017", periods = len(data), freq = "D")
data = data.set_index(["Date"])
data.columns = [pm_type]

## Plot to overview the dataset
pm_str = "PM" + r"$\mathrm{_{2.5}}$"
pm_mgpm3 = "PM" + r"$\mathrm{_{2.5}}$ " + r"$\mathrm{(\mu}$" + "g/m" + r"$\mathrm{^{3})}$"
```

```
plt.figure(figsize = (24, 12))
data[pm_type].plot(label = "Original", linewidth = 2, color = "gray")
data[pm_type].rolling(window = 30).mean().plot(label = "30-MA", linewidth = 3, color = "orange")
plt.title(pm_str + " in " + city + " during 2017-2021", fontsize = 48)
plt.legend(fontsize = 48)
plt.xlabel("TIME", fontsize = 48)
plt.ylabel(pm_mgpm3, fontsize = 48)
plt.xticks(fontsize = 48)
plt.yticks(fontsize = 48)
plt.ylim(0, 100)
plt.grid(True)
plt.show()
plt.savefig(data_path + pm_type + "_" + city + "_DATASET.png", dpi = 300)

## Decompose the time-series data
from statsmodels.tsa.seasonal import seasonal_decompose

results = seasonal_decompose(data[pm_type], model = "multiplicable", period = 365)
fig, axs = plt.subplots(4, sharex = True, sharey = False, figsize = (24, 12))
fig.suptitle("Decompose of " + city + "\'s " + pm_str + " Dataset", fontsize = 48)

axs[0].plot(data[pm_type], label = "ACTUAL", linewidth = 3, color = "black")
axs[0].grid(True)
axs[0].legend(fontsize = 24)
axs[1].plot(results.trend, label = "TREND", linewidth = 3, color = "green")
axs[1].grid(True)
axs[1].legend(fontsize = 24)
axs[2].plot(results.seasonal, label = "SEASONAL", linewidth = 3, color = "blue")
axs[2].grid(True)
axs[2].legend(fontsize = 24)
axs[3].plot(results.resid, label = "RESIDUAL", linewidth = 3, color = "red")
axs[3].grid(True)
axs[3].legend(fontsize = 24)
plt.xlabel("TIME", fontsize = 48)
plt.show()
```

```
## Check stationary of the time-series data
from statsmodels.tsa.stattools import adfuller

def ad_test(dataset):
        dftest = adfuller(dataset, autolag = "AIC")
        print("1. ADF: ", dftest[0])
        print("2. P-Value: ", dftest[1])
        print("3. Number of Lag: ", dftest[2])
        print("4. Number of Observations Used for ADF Regression and Critical \
                Values Calculation: ", dftest[3])
        print("5. Critical Values: ")

        for key, val in dftest[4].items():
                print("\t", key, ": ", val)

        if(dftest[1] <= 0.05) & (dftest[4]["5%"] > dftest[0]):
                print("\nThe data is Stationary :) ")
        else:
                print("\nThe data is Non-stationary :(")

ad_test(data[pm_type])

## Plot ACF and PACF graphs
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import statsmodels.api as sm
import warnings

warnings.filterwarnings("ignore")

fig = plt.figure(figsize = (24, 12))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(data[pm_type].dropna(), lags = 20, ax = ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(data[pm_type].dropna(), lags = 20, ax = ax2)
```

```
## Find a suitable arima model configuration
from pmdarima import auto_arima

# Split the dataset into train and test set
one_year_test = 365
train = data.iloc[:-(one_year_test)]
test = data.iloc[-(one_year_test):]
auto_fit = auto_arima(train[pm_type], trace = True, parallel = False, stepwise = True, \
        suppress_warnings = True)
print(auto_fit.summary())

## Create and train a ARIMA model
from statsmodels.tsa.arima.model import ARIMA

regex = "ARIMAX\(([0-9]+), ([0-9]+), ([0-9]+)"
summary_string = str(auto_fit.summary())
param = re.findall(regex, summary_string)
p, d, q = int(param[0][0]) , int(param[0][1]) , int(param[0][2])
print("ARIMA: ", p, d, q)

arima = ARIMA(train[pm_type], order = (p, d, q))
model = arima.fit()
print(model.summary())

# Save train prediction
train["Predict"] = model.predict()
train.columns = ["Observe", "Predict"]
train["Predict"].to_csv(data_path + pm_type + "_" + city + "_" + method + \
        "_STD_TRAIN_PRED.csv", encoding = "utf-8")

## Run the ARIMA model forecasting - Rolling Forecast
pred = []
for i in range(len(test)):
        to_train = data.iloc[:-(one_year_test) + (i)]
        arima = ARIMA(to_train[pm_type], order = (p, d, q))
        model = arima.fit()
```

```
        pred.append(model.forecast()[0]) #single-step forecasting

test["Predict"] = pred
test.columns = ["Observe", "Predict"]
test["Predict"].to_csv(data_path + pm_type + "_" + city + "_" + method + \
        "_STD_TEST_PRED.csv", encoding = "utf-8")

## Evaluate and plot the results
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from statistics import mean
from math import sqrt

## Evaluation of train set
#MAPE function
def MAPE(y_actual, y_predicted):
        mape = np.mean(np.abs((y_actual - y_predicted) / y_actual)) * 100
        return mape

#%RMSE function
def PRMSE(y_actual, rmse):
        prmse = (rmse / np.mean(y_actual)) * 100
        return prmse

#R^2
rsqure = r2_score(train["Observe"], train["Predict"])
#MAE
mae = mean_absolute_error(train["Observe"], train["Predict"])
#MAPE
mape = MAPE(train["Observe"], train["Predict"])
#RMSE
rmse = sqrt(mean_squared_error(train["Observe"], train["Predict"]))
#%RMSE
prmse = PRMSE(train["Observe"], rmse)
```

```
#Save the metric results to csv file
result_path = working_path + "result/metrics/PM/" + pm_type
matrics = pd.DataFrame([pm_type + "_" + city + "_" + method + \
          "_STD_TRAIN", rsqure, mae, mape, rmse, prmse])
matrics.to_csv(result_path + "/" + pm_type + \
          "_STD_EVALU_RESULTS.csv", mode = "a", index = False, header = False)

#Show results
print("\nR^2 of train: ", rsqure)
print("MAE of train: ", mae)
print("MAPE of train: ", mape)
print("RMSE of train: ", rmse)
print("%RMSE of train: ", prmse)

## Evaluation of test set
#R^2
rsqure = r2_score(test["Observe"], test["Predict"])
#MAE
mae = mean_absolute_error(test["Observe"], test["Predict"])
#MAPE
mape = MAPE(test["Observe"], test["Predict"])
#RMSE
rmse = sqrt(mean_squared_error(test["Observe"], test["Predict"]))
#%RMSE
prmse = PRMSE(test["Observe"], rmse)

#Save the metric results to csv file
result_path = working_path + "result/metrics/PM/" + pm_type
matrics = pd.DataFrame([pm_type + "_" + city + "_" + method + \
          "_STD_TEST", rsqure, mae, mape, rmse, prmse])
matrics.to_csv(result_path + "/" + pm_type + \
          "_STD_EVALU_RESULTS.csv", mode = "a", index = False, header = False)

#Show results
print("\nR^2 of test: ", rsqure)
print("MAE of test: ", mae)
```

```
print("MAPE of test: ", mape)
print("RMSE of test: ", rmse)
print("%RMSE of test: ", prmse)

# Plot Train
plt.figure(figsize = (24, 12))
plt.title(pm_str + " Forecasting in " + city + " of Trainset: " + method, fontsize = 48)
train["Observe"].rolling(window = 30).mean().plot(label="Observed", linewidth = 3, color = "gray")
train["Predict"].rolling(window = 30).mean().plot(label="Predicted", linewidth = 3, \
        color = "red", linestyle = "dashdot")
plt.xlabel("TIME (days)", fontsize = 48)
plt.ylabel(pm_mgpm3, fontsize = 48)
plt.xticks(fontsize = 48)
plt.yticks(fontsize = 48)
plt.ylim(0, 60)
plt.grid(True)
plt.legend(fontsize = 48)

# Plot Test
plt.figure(figsize = (24, 12))
plt.title(pm_str + " Forecasting in " + city + " of Testset: " + method, fontsize = 48)
#test["Observe"].plot(label = "Observed", linewidth = 3, color = "gray")
#test["Predict"].plot(label = "Predicted", linewidth = 3, color = "red", linestyle = "dashdot")
test["Observe"].rolling(window = 7).mean().plot(label="Observed", linewidth = 3, color = "gray")
test["Predict"].rolling(window = 7).mean().plot(label="Predicted", linewidth = 3, \
        color = "red", linestyle = "dashdot")
plt.xlabel("TIME (days)", fontsize = 48)
plt.ylabel(pm_mgpm3, fontsize = 48)
plt.xticks(fontsize = 48)
plt.yticks(fontsize = 48)
plt.ylim(0, 60)
plt.grid(True)
plt.legend(fontsize = 48)
################################## END ARIMA ##################################
```

2. โปรแกรมสำหรับพัฒนาแบบจำลองอนุกรมเวลา ANN สำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ด้วยภาษา Python

```
################################## BEGIN ANN ##################################
from IPython.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams["font.family"] = "Times New Roman"

method = "ANN"
city = "RAYONG"
pm_type = "PM2dot5"

## Load dataset
working_path = "/home/anupong/works/2022-trimester-0265/thesis/"
file_name = pm_type + "_" + city + "_" + method + "_DATASET.csv"
data_path = working_path + "dataset/" + pm_type + "/csv/use/" + city + "/" + method + "/"
pm_data = data_path + file_name
missing_values = ["n/a", "na", "--", "null", "?"]
data = pd.read_csv(pm_data, na_values = missing_values)
print("\nMissing values: \n", data.isnull().sum() + data.isna().sum()) #Check missing values
print("\nData Size: ", len(data))

## Data preparation
# train test split
one_year_test = 365
train_observe = data.iloc[:-(one_year_test)]
test_observe = data.iloc[-(one_year_test):]

train_data = train_observe.to_numpy()
test_data = test_observe.to_numpy()
train_data = train_data.reshape((len(train_data),))
```

```
test_data = test_data.reshape((len(test_data),))

## Transform time-series data to supervised learning data
from keras.preprocessing.sequence import TimeseriesGenerator

n_predictor = 3
n_target = 1
generator = TimeseriesGenerator(train_data, train_data, \
        length = n_predictor, batch_size = n_target)

## Create and train ANN model
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers as opt
from timeit import default_timer as timer

number_ann_node = 64
model_name = method + "_" + str(number_ann_node) + "_2"

# define model
model = Sequential(name = model_name)
model.add(Dense(number_ann_node, activation = "relu", input_dim = n_predictor))
model.add(Dense(number_ann_node, activation = "relu"))
model.add(Dense(1))
model.compile(optimizer = opt.Adam(learning_rate = 0.001, beta_1 = 0.9, beta_2 = 0.999), \
        loss = "mse")
model.summary()

# fit model (learning process)
stime = timer()
model.fit(generator, epochs = 1000, verbose = 0)
etime = timer()
print("Time to training is ", (etime - stime), " Seconds.")

loss_per_epoch = model.history.history["loss"]
plt.figure(figsize = (24, 12))
```

```
plt.title(model_name, fontsize = 48)
plt.plot(range(len(loss_per_epoch)), loss_per_epoch, label = "Train Error", linewidth = 3, \
        color = "red")
plt.legend(fontsize = 48)
plt.xlabel("EPOCH", fontsize = 48)
plt.ylabel("MSE", fontsize = 48)
plt.xticks(fontsize = 48)
plt.yticks(fontsize = 48)
plt.grid(True)

## Forecasting of train-set
pred = []

first_eval_batch = train_data[:n_predictor]
current_batch = first_eval_batch.reshape(1, n_predictor)

for i in range(len(train_observe) - n_predictor):
        #get the prediction value for the first batch
        current_pred = model.predict(current_batch)[0]

        #append the prediction into the array
        pred.append(current_pred)
        actual = train_data[i + n_predictor]

        #use the prediction to update the batch and remove the first value
        current_batch = np.append(current_batch[:, 1:], [[actual]], axis = 1)

train = pd.DataFrame(pred, columns = ["Predict"])
train_observe = data.iloc[n_predictor:-(one_year_test)]
train_observe.reset_index(drop = True, inplace = True)
train["Observe"] = train_observe
train = train.astype(float)
train["Predict"].to_csv(data_path + pm_type + "_" + city + "_" + method + \
        "_STD_TRAIN_PRED.csv", encoding = "utf-8")
```

```
## Forecasting of test-set
pred = []

#select last three values from train-set to be predictor for fist predection
first_eval_batch = train_data[-n_predictor:]
current_batch = first_eval_batch.reshape(1, n_predictor)

for i in range(len(test_observe)):
        # get the prediction value for the first batch
        current_pred = model.predict(current_batch)[0]

        # append the prediction into the array
        pred.append(current_pred)
        actual = test_data[i]

        # use the prediction to update the batch and remove the first value
        current_batch = np.append(current_batch[:, 1:], [[actual]], axis = 1)

test = pd.DataFrame(pred, columns = ["Predict"])
test_observe.reset_index(drop = True, inplace = True)
test["Observe"] = test_observe
test = test.astype(float)
test["Predict"].to_csv(data_path + pm_type + "_" + city + "_" + method + \
        "_STD_TEST_PRED.csv", encoding = "utf-8")

## Evaluate and plot the results
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from statistics import mean
from math import sqrt
```

```
## Evaluation of train set
#MAPE function
def MAPE(y_actual, y_predicted):
        mape = np.mean(np.abs((y_actual - y_predicted) / y_actual)) * 100
        return mape

#%RMSE function
def PRMSE(y_actual, rmse):
        prmse = (rmse / np.mean(y_actual)) * 100
        return prmse

#R^2
rsqure = r2_score(train["Observe"], train["Predict"])
#MAE
mae = mean_absolute_error(train["Observe"], train["Predict"])
#MAPE
mape = MAPE(train["Observe"], train["Predict"])
#RMSE
rmse = sqrt(mean_squared_error(train["Observe"], train["Predict"]))
#%RMSE
prmse = PRMSE(train["Observe"], rmse)

#Save the metric results to csv file
result_path = working_path + "result/metrics/PM/" + pm_type
matrics = pd.DataFrame([pm_type + "_" + city + "_" + method + \
        "_STD_TRAIN", rsqure, mae, mape, rmse, prmse])
matrics.to_csv(result_path + "/" + pm_type + \
        "_STD_EVALU_RESULTS.csv", mode = "a", index = False, header = False)

#Show results
print("\nR^2 of train: ", rsqure)
print("MAE of train: ", mae)
print("MAPE of train: ", mape)
print("RMSE of train: ", rmse)
print("%RMSE of train: ", prmse)
```

```
## Evaluation of test set
#R^2
rsqure = r2_score(test["Observe"], test["Predict"])
#MAE
mae = mean_absolute_error(test["Observe"], test["Predict"])
#MAPE
mape = MAPE(test["Observe"], test["Predict"])
#RMSE
rmse = sqrt(mean_squared_error(test["Observe"], test["Predict"]))
#%RMSE
prmse = PRMSE(test["Observe"], rmse)

#Save the metric results to csv file
result_path = working_path + "result/metrics/PM/" + pm_type
matrics = pd.DataFrame([pm_type + "_" + city + "_" + method + \
          "_STD_TEST", rsqure, mae, mape, rmse, prmse])
matrics.to_csv(result_path + "/" + pm_type + \
          "_STD_EVALU_RESULTS.csv", mode = "a", index = False, header = False)

#Show results
print("\nR^2 of test: ", rsqure)
print("MAE of test: ", mae)
print("MAPE of test: ", mape)
print("RMSE of test: ", rmse)
print("%RMSE of test: ", prmse)

pm_str = "PM" + r"$\mathrm{_{2.5}}$"
pm_mgpm3 = "PM" + r"$\mathrm{_{2.5}}$ " + r"$\mathrm{(\mu}$" + "g/m" + r"$\mathrm{^{3})}$"

## Set date to index in train
train["Date"] = pd.date_range(start = "1/4/2017", periods = len(train), freq = "D")
train = train.set_index(["Date"])

# Plot Train
plt.figure(figsize = (24, 12))
plt.title(pm_str + " Forecasting in " + city + " of Trainset: " + model_name, fontsize = 48)
```

```
train["Observe"].rolling(window = 30).mean().plot(label="Observed", linewidth = 3, color = "gray")
train["Predict"].rolling(window = 30).mean().plot(label="Predicted", linewidth = 3, \
        color = "red", linestyle = "dashdot")
plt.xlabel("TIME (days)", fontsize = 48)
plt.ylabel(pm_mgpm3, fontsize = 48)
plt.xticks(fontsize = 48)
plt.yticks(fontsize = 48)
plt.grid(True)
plt.legend(fontsize = 48)

## Set date to index in test
test["Date"] = pd.date_range(start = "1/1/2021", periods = len(test), freq = "D")
test = test.set_index(["Date"])

# Plot Test
plt.figure(figsize = (24, 12))
plt.title(pm_str + " Forecasting in " + city + " of Testset: " + model_name , fontsize = 48)
test["Observe"].rolling(window = 7).mean().plot(label="Observed", linewidth = 3, color = "gray")
test["Predict"].rolling(window = 7).mean().plot(label="Predicted", linewidth = 3, \
        color = "red", linestyle = "dashdot")
plt.xlabel("TIME (days)", fontsize = 48)
plt.ylabel(pm_mgpm3, fontsize = 48)
plt.xticks(fontsize = 48)
plt.yticks(fontsize = 48)
plt.grid(True)
plt.legend(fontsize = 48)
################################### END ANN ###################################
```

3. โปรแกรมสำหรับพัฒนาแบบจำลองอนุกรมเวลา LSTM สำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ด้วยภาษา Python

```
################################## BEGIN LSTM ##################################
from IPython.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams["font.family"] = "Times New Roman"

method = "LSTM"
city = "RAYONG"
pm_type = "PM2dot5"

## Load dataset
working_path = "/home/anupong/works/2022-trimester-0265/thesis/"
file_name = pm_type + "_" + city + "_" + method + "_DATASET.csv"
data_path = working_path + "dataset/" + pm_type + "/csv/use/" + city + "/" + method + "/"
pm_data = data_path + file_name
missing_values = ["n/a", "na", "--", "null", "?"]
data = pd.read_csv(pm_data, na_values = missing_values)
print("\nMissing values: \n", data.isnull().sum() + data.isna().sum()) #Check missing values
print("\nData Size: ", len(data))

## Data preparation
# train test split
one_year_test = 365
train_observe = data.iloc[:-(one_year_test)]
test_observe = data.iloc[-(one_year_test):]
train_data = train_observe.to_numpy()
test_data = test_observe.to_numpy()
```

```
## Transform time-series data to supervised learning data
from keras.preprocessing.sequence import TimeseriesGenerator
# define genarator (Lag t-1, t-2, t-3)
n_predictor = 3
n_target = 1
generator = TimeseriesGenerator(train_data, train_data, length = n_predictor, \
            batch_size = n_target)

## Create and train LSTM model
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras import optimizers as opt
from timeit import default_timer as timer

number_lstm_node = 16
model_name = method + "_" + str(number_lstm_node) + "_4"

# define model
model = Sequential(name = model_name)
model.add(LSTM(number_lstm_node, activation = \
            "relu", input_shape = (n_predictor, n_target), return_sequences = True))
model.add(LSTM(number_lstm_node, activation = "relu", return_sequences = True))
model.add(LSTM(number_lstm_node, activation = "relu", return_sequences = True))
model.add(LSTM(number_lstm_node, activation = "relu"))
model.add(Dense(1))
model.compile(optimizer = opt.Adam(learning_rate = 0.001, beta_1 = 0.9, beta_2 = 0.999), \
            loss = "mse")
model.summary()

# fit model (learning process)
stime = timer()
model.fit(generator, epochs = 1000, verbose = 0)
etime = timer()
print("Time to training is ", (etime - stime), " Seconds.")
```

```
loss_per_epoch = model.history.history["loss"]
plt.figure(figsize = (24, 12))
plt.title(model_name, fontsize = 48)
plt.plot(range(len(loss_per_epoch)), loss_per_epoch, label = "Train Error", linewidth = 3, \
        color = "red")
plt.legend(fontsize = 48)
plt.xlabel("EPOCH", fontsize = 48)
plt.ylabel("MSE", fontsize = 48)
plt.xticks(fontsize = 48)
plt.yticks(fontsize = 48)
plt.grid(True)

## Forecasting of train-set
pred = []

first_eval_batch = train_data[:n_predictor]
current_batch = first_eval_batch.reshape(1, n_predictor, n_target)

for i in range(len(train_observe) - n_predictor):
        #get the prediction value for the first batch
        current_pred = model.predict(current_batch)[0]

        #append the prediction into the array
        pred.append(current_pred)
        actual = train_data[i + n_predictor]

        #use the prediction to update the batch and remove the first value
        current_batch = np.append(current_batch[:, 1:, :], [[actual]], axis = 1)

train = pd.DataFrame(pred, columns = ["Predict"])
train_observe = data.iloc[n_predictor:-(one_year_test)]
train_observe.reset_index(drop = True, inplace = True)
train["Observe"] = train_observe
train = train.astype(float)
train["Predict"].to_csv(data_path + pm_type + "_" + city +  "_" + method + \
        "_STD_TRAIN_PRED.csv", encoding = "utf-8")
```

```
## Forecasting of test-set
pred = []

#select last three values from train-set to be predictor for fist predection
first_eval_batch = train_data[-n_predictor:]
current_batch = first_eval_batch.reshape(1, n_predictor, n_target)

for i in range(len(test_observe)):
        # get the prediction value for the first batch
        current_pred = model.predict(current_batch)[0]

        # append the prediction into the array
        pred.append(current_pred)
        actual = test_data[i]

        # use the prediction to update the batch and remove the first value
        current_batch = np.append(current_batch[:, 1:, :], [[actual]], axis = 1)

test = pd.DataFrame(pred, columns = ["Predict"])
test_observe.reset_index(drop = True, inplace = True)
test["Observe"] = test_observe
test = test.astype(float)
test["Predict"].to_csv(data_path + pm_type + "_" + city + "_" + method + \
        "_STD_TEST_PRED.csv", encoding = "utf-8")

## Evaluate and plot the results
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from statistics import mean
from math import sqrt
```

```
## Evaluation of train set
#MAPE function
def MAPE(y_actual, y_predicted):
        mape = np.mean(np.abs((y_actual - y_predicted) / y_actual)) * 100
        return mape

#%RMSE function
def PRMSE(y_actual, rmse):
        prmse = (rmse / np.mean(y_actual)) * 100
        return prmse
#R^2
rsqure = r2_score(train["Observe"], train["Predict"])
#MAE
mae = mean_absolute_error(train["Observe"], train["Predict"])
#MAPE
mape = MAPE(train["Observe"], train["Predict"])
#RMSE
rmse = sqrt(mean_squared_error(train["Observe"], train["Predict"]))
#%RMSE
prmse = PRMSE(train["Observe"], rmse)

#Save the metric results to csv file
result_path = working_path + "result/metrics/PM/" + pm_type
matrics = pd.DataFrame([pm_type + "_" + city + "_" + method + \
        "_STD_TRAIN", rsqure, mae, mape, rmse, prmse])
matrics.to_csv(result_path + "/" + pm_type + \
        "_STD_EVALU_RESULTS.csv", mode = "a", index = False, header = False)

#Show results
print("\nR^2 of train: ", rsqure)
print("MAE of train: ", mae)
print("MAPE of train: ", mape)
print("RMSE of train: ", rmse)
print("%RMSE of train: ", prmse)
```

```
## Evaluation of test set
#R^2
rsqure = r2_score(test["Observe"], test["Predict"])
#MAE
mae = mean_absolute_error(test["Observe"], test["Predict"])
#MAPE
mape = MAPE(test["Observe"], test["Predict"])
#RMSE
rmse = sqrt(mean_squared_error(test["Observe"], test["Predict"]))
#%RMSE
prmse = PRMSE(test["Observe"], rmse)

#Save the metric results to csv file
result_path = working_path + "result/metrics/PM/" + pm_type
matrics = pd.DataFrame([pm_type + "_" + city + "_" + method + \
          "_STD_TEST", rsqure, mae, mape, rmse, prmse])
matrics.to_csv(result_path + "/" + pm_type + \
          "_STD_EVALU_RESULTS.csv", mode = "a", index = False, header = False)

#Show results
print("\nR^2 of test: ", rsqure)
print("MAE of test: ", mae)
print("MAPE of test: ", mape)
print("RMSE of test: ", rmse)
print("%RMSE of test: ", prmse)

pm_str = "PM" + r"$\mathrm{_{2.5}}$"
pm_mgpm3 = "PM" + r"$\mathrm{_{2.5}}$ " + r"$\mathrm{(\mu}$" + "g/m" + r"$\mathrm{^{3})}$"

## Set date to index in train
train["Date"] = pd.date_range(start = "1/4/2017", periods = len(train), freq = "D")
train = train.set_index(["Date"])

# Plot Train
plt.figure(figsize = (24, 12))
plt.title(pm_str + " Forecasting in " + city + " of Trainset: " + model_name, fontsize = 48)
```

```
train["Observe"].rolling(window = 30).mean().plot(label="Observed", linewidth = 3, color = "gray")
train["Predict"].rolling(window = 30).mean().plot(label="Predicted", linewidth = 3, \
        color = "red", linestyle = "dashdot")
plt.xlabel("TIME (days)", fontsize = 48)
plt.ylabel(pm_mgpm3, fontsize = 48)
plt.xticks(fontsize = 48)
plt.yticks(fontsize = 48)
plt.grid(True)
plt.legend(fontsize = 48)
## Set date to index in test
test["Date"] = pd.date_range(start = "1/1/2021", periods = len(test), freq = "D")
test = test.set_index(["Date"])

# Plot Test
plt.figure(figsize = (24, 12))
plt.title(pm_str + " Forecasting in " + city + " of Testset: " + model_name , fontsize = 48)
test["Observe"].rolling(window = 7).mean().plot(label="Observed", linewidth = 3, color = "gray")
test["Predict"].rolling(window = 7).mean().plot(label="Predicted", linewidth = 3, \
        color = "red", linestyle = "dashdot")
plt.xlabel("TIME (days)", fontsize = 48)
plt.ylabel(pm_mgpm3, fontsize = 48)
plt.xticks(fontsize = 48)
plt.yticks(fontsize = 48)
plt.grid(True)
plt.legend(fontsize = 48)
################################## END LSTM ##################################
```

4. โปรแกรมสำหรับพัฒนาแบบจำลองอนุกรมเวลา ANFIS สำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ด้วยภาษา Matlab และ Python

```
################################## BEGIN ANFIS ###################################
%% MATLAB-PART %%
%% MODELING %%
clc; close all;

%% Load data from csv file into array
method = 'ANFIS'
city = 'RAYONG';
pmType = 'PM2dot5';
numMF = 2;
inputMF = 'gaussmf';
outputMF = 'constant';
topology = '2_GAUSSMF'

fileName = strcat(pmType, '_', city,'_', method, '_DATASET.csv');
dataPath = strcat('/home/anupong/works/2022-trimester-0265/thesis/dataset/', \
           pmType, '/csv/use/', city, '/', method, '/');
dataFile = strcat(dataPath, fileName)
rawData = readtable(dataFile, 'VariableNamingRule', 'preserve');
dataset = table2array(rawData);

%% Split the dataset into train/test dataset
oneYearTest = 365;
numTrain = size(dataset, 1) - oneYearTest;
trainData = dataset(1:numTrain, :);
testData = dataset(numTrain + 1:end, :);

%% Create Sugino-FIS architechture
fisOption = genfisOptions('GridPartition');
fisOption.NumMembershipFunctions = numMF;
fisOption.InputMembershipFunctionType = inputMF;
fisOption.OutputMembershipFunctionType = outputMF;
```

```
%fisOption = genfisOptions('SubtractiveClustering');

fis = genfis(trainData(:, 1:end - 1), trainData(:, end), fisOption)

%% Tune Sugino-FIS parameters with ANFIS
rng('shuffle');
epoch = 1000;
[in, out, rule] = getTunableSettings(fis)
tune = tunefisOptions('Method', 'anfis', 'OptimizationType', 'tuning', 'UseParallel', false);
tune.MethodOptions.EpochNumber = epoch;                 %Default is 10.
tune.MethodOptions.ErrorGoal = 0.1;                     %Default is 0.
tune.MethodOptions.InitialStepSize = 0.01;              %Default is 0.01.
tune.MethodOptions.StepSizeIncreaseRate = 1.1;          %Default is 1.1.
tune.MethodOptions.StepSizeDecreaseRate = 0.9;          %Default is 0.9.
tune.MethodOptions.OptimizationMethod = 1;              %Default is 1.
%tune.MethodOptions.ValidationData = [];                %Default is empty array.

%Display options
tune.MethodOptions.DisplayErrorValues = 1;              %Default is 1.
tune.MethodOptions.DisplayStepSize = 1;                 %Default is 1.
tune.MethodOptions.DisplayFinalResults = 1;             %Default is 1.

%% Capture the time of learning process
tStart = tic;
[myfis, summary] = tunefis(fis, [in; out], trainData(:, 1:end - 1), trainData(:, end), tune)
elabTime = toc(tStart)

%% Save the ANFIS model
modelFile = strcat(dataPath, pmType, '_', city, '_', method, '_', topology, '_MODEL.mat');
save(modelFile);

%% EVALUATEING %%
clc; close all;

%% Load ANFIS model and all variales
```

```
method = 'ANFIS'
city = 'RAYONG';
pmType = 'PM2dot5';
topology = '2_GBELLMF'
fileName = strcat(pmType, '_', city,'_', method, '_', topology, '_MODEL.mat');
dataPath = strcat('/home/anupong/works/2022-trimester-0265/thesis/dataset/', \
           pmType, '/csv/use/', city, '/', method, '/');
modelFile = strcat(dataPath, fileName)
load(modelFile);

%% Evaluate the ANFIS models
predictTrain = evalfis(myfis, trainData(:, 1:end - 1));
predictTest = evalfis(myfis, testData(:, 1:end - 1));

%% Calculate the assessment of prediction
%R^2
rsqureTrain = 1 - (sum((trainData(:, end) - predictTrain).^2) / \
           sum((trainData(:, end) - mean(trainData(:, end))).^2));
rsqureTest = 1 - (sum((testData(:, end) - predictTest).^2) / \
           sum((testData(:, end) - mean(testData(:, end))).^2));
%RMSE
rmseTrain = sqrt(immse(predictTrain, trainData(:, end)));
rmseTest = sqrt(immse(predictTest, testData(:, end)));
%PRMSE
prmseTrain = (100 / mean(trainData(:, end))) * rmseTrain;
prmseTest= (100 / mean(testData(:, end))) * rmseTest;
%MAE
maeTrain = mae((trainData(:, end) - predictTrain));
maeTest = mae((testData(:, end) - predictTest));
%MAPE
mapeTrain = mean(abs((trainData(:, end) - predictTrain) ./ trainData(:, end))) * 100;
mapeTest = mean(abs((testData(:, end) - predictTest) ./ testData(:, end))) * 100;

%% Keep results
learningTime = elabTime
trainResults = [rsqureTrain; maeTrain; mapeTrain; rmseTrain; prmseTrain]
```

```
testResults = [rsqureTest; maeTest; mapeTest; rmseTest; prmseTest]

%% Save forecasted results to csv file
saveTrain = strcat(dataPath, pmType, '_', city, '_', method, '_', topology, '_TRAIN.csv');
saveTest = strcat(dataPath, pmType, '_', city, '_', method, '_', topology, '_TEST.csv');
saveLearning = strcat(dataPath, pmType, '_', city, '_', method, '_', topology, '_LEARNING.csv');
saveTrainResults = strcat(dataPath, pmType, '_', city, '_', method, '_', topology, \
        '_TRAIN_RESULTS.csv');
saveTestResults = strcat(dataPath, pmType, '_', city, '_', method, '_', topology, \
        '_TEST_RESULTS.csv');
saveLearningTime = strcat(dataPath, pmType, '_', city, '_', method, '_', topology, \
        '_LEARNING_TIME.txt');

writematrix(predictTrain, saveTrain);
writematrix(predictTest, saveTest);
writematrix(summary.tuningOutputs.trainError, saveLearning);
writematrix(trainResults, saveTrainResults);
writematrix(testResults, saveTestResults);
writematrix(learningTime, saveLearningTime);

%% Save matlab workspace
resultsPath = strcat('/home/anupong/works/2022-trimester-0265/thesis/result/matlab/PM/', \
        pmType, '/');
saveName = [datestr(now, 'yyyy-mmm-dd_HHMMSS'), '-', pmType, '_', city, '_', \
        method, '_', topology, '_RESULTS.mat'];
strSave = append(resultsPath, saveName)
save(strSave);


## PYTHON-PART ##
from IPython.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))

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

```
plt.rcParams["font.family"] = "Times New Roman"

method = "ANFIS"
city = "RAYONG"
pm_type = "PM2dot5"
topology = "2_GBELLMF"

## Load dataset to show
working_path = "/home/anupong/works/2022-trimester-0265/thesis/"
data_path = working_path + "dataset/" + pm_type + "/csv/use/" + city + "/" + method + "/"
file_name = pm_type + "_" + city + "_" + method + "_DATASET.csv"
pm_data = data_path + file_name
missing_values = ["n/a", "na", "--", "null", "?"]
data = pd.read_csv(pm_data, na_values = missing_values)
print("\nMissing values: \n", data.isnull().sum() + data.isna().sum()) #Check missing values
print("\nData Size: ", len(data))

## Create ANFIS and Training with the ANFIS in Matlab
import os

code_dir = "code/matlab/PM/"
matlib_file = pm_type + "_" + city + "_" + method + "_" + topology + "_MODEL.m"
run = "time matlab -nodisplay -nosplash -nodesktop -r"
cmd = run + " \"run(\'" + working_path + code_dir + pm_type + "/" + \
          city + "/" + matlib_file + "\'); exit;\""
os.system(cmd)

## train test split and collect the result data
one_year_test = 365
train_observe = data.iloc[:-(one_year_test)]
test_observe = data.iloc[-(one_year_test):]
test_observe.reset_index(drop = True, inplace = True)

train = pd.DataFrame()
test = pd.DataFrame()
train["Observe"] = train_observe["t"]
```

```
test["Observe"] = test_observe["t"]

model_name = method + "_" + topology
file_prefix = pm_type + "_" + city + "_" + method + "_" + topology
train_time = np.loadtxt(data_path + file_prefix + "_LEARNING_TIME.txt")
learning = pd.read_csv(data_path + file_prefix + "_LEARNING.csv", header = None)
learning["MSE"] = pow(learning, 2)
learning.columns = ["RMSE", "MSE"]
train_pred = pd.read_csv(data_path + file_prefix + "_TRAIN.csv", header = None)
test_pred = pd.read_csv(data_path + file_prefix + "_TEST.csv", header = None)
train["Predict"] = train_pred
test["Predict"] = test_pred

## Plot learning graph
print("\nTime to training is ", train_time, " Seconds.")

# plot learning
plt.figure(figsize = (24, 12))
plt.title(model_name, fontsize = 48)
plt.plot(range(len(learning)), learning["MSE"], label = "Train Error", linewidth = 3, color = "red")
plt.legend(fontsize = 48)
plt.xlabel("EPOCH", fontsize = 48)
plt.ylabel("MSE", fontsize = 48)
plt.xticks(fontsize = 48)
plt.yticks(fontsize = 48)
plt.grid(True)

## Evaluate and plot the results
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from statistics import mean
from math import sqrt

## Evaluation of train set
#MAPE function
```

```
def MAPE(y_actual, y_predicted):
        mape = np.mean(np.abs((y_actual - y_predicted) / y_actual)) * 100
        return mape

#%RMSE function
def PRMSE(y_actual, rmse):
        prmse = (rmse / np.mean(y_actual)) * 100
        return prmse

#R^2
rsqure = r2_score(train["Observe"], train["Predict"])
#MAE
mae = mean_absolute_error(train["Observe"], train["Predict"])
#MAPE
mape = MAPE(train["Observe"], train["Predict"])
#RMSE
rmse = sqrt(mean_squared_error(train["Observe"], train["Predict"]))
#%RMSE
prmse = PRMSE(train["Observe"], rmse)

#Save the metric results to csv file
result_path = working_path + "result/metrics/PM/" + pm_type
matrics = pd.DataFrame([pm_type + "_" + city + "_" + method + \
        "_STD_TRAIN", rsqure, mae, mape, rmse, prmse])
matrics.to_csv(result_path + "/" + pm_type + \
        "_STD_EVALU_RESULTS.csv", mode = "a", index = False, header = False)

#Show results
print("\nR^2 of train: ", rsqure)
print("MAE of train: ", mae)
print("MAPE of train: ", mape)
print("RMSE of train: ", rmse)
print("%RMSE of train: ", prmse)

## Evaluation of test set
#R^2
```

```
rsqure = r2_score(test["Observe"], test["Predict"])
#MAE
mae = mean_absolute_error(test["Observe"], test["Predict"])
#MAPE
mape = MAPE(test["Observe"], test["Predict"])
#RMSE
rmse = sqrt(mean_squared_error(test["Observe"], test["Predict"]))
#%RMSE
prmse = PRMSE(test["Observe"], rmse)

#Save the metric results to csv file
result_path = working_path + "result/metrics/PM/" + pm_type
matrics = pd.DataFrame([pm_type + "_" + city + "_" + method + \
          "_STD_TEST", rsqure, mae, mape, rmse, prmse])
matrics.to_csv(result_path + "/" + pm_type + \
          "_STD_EVALU_RESULTS.csv", mode = "a", index = False, header = False)

#Show results
print("\nR^2 of test: ", rsqure)
print("MAE of test: ", mae)
print("MAPE of test: ", mape)
print("RMSE of test: ", rmse)
print("%RMSE of test: ", prmse)

pm_str = "PM" + r"$\mathrm{_{2.5}}$"
pm_mgpm3 = "PM" + r"$\mathrm{_{2.5}}$ " + r"$\mathrm{(\mu}$" + "g/m" + r"$\mathrm{^{3})}$"

## Set date to index in train
train["Date"] = pd.date_range(start = "1/4/2017", periods = len(train), freq = "D")
train = train.set_index(["Date"])

# Plot Train
plt.figure(figsize = (24, 12))
plt.title(pm_str + " Forecasting in " + city + " of Trainset: " + model_name, fontsize = 48)
train["Observe"].rolling(window = 30).mean().plot(label = "Observed", linewidth = 3, color = "gray")
train["Predict"].rolling(window = 30).mean().plot(label = "Predicted", linewidth = 3, \
```

```
            color = "red", linestyle = "dashdot")
plt.xlabel("TIME (days)", fontsize = 48)
plt.ylabel(pm_mgpm3, fontsize = 48)
plt.xticks(fontsize = 48)
plt.yticks(fontsize = 48)
plt.grid(True)
plt.legend(fontsize = 48)

## Set date to index in test
test["Date"] = pd.date_range(start = "1/1/2021", periods = len(test), freq = "D")
test = test.set_index(["Date"])

# Plot Test
plt.figure(figsize = (24, 12))
plt.title(pm_str + " Forecasting in " + city + " of Testset: " + model_name , fontsize = 48)
test["Observe"].rolling(window = 7).mean().plot(label = "Observed", linewidth = 3, color = "gray")
test["Predict"].rolling(window = 7).mean().plot(label = "Predicted", linewidth = 3, \
            color = "red", linestyle = "dashdot")
plt.xlabel("TIME (days)", fontsize = 48)
plt.ylabel(pm_mgpm3, fontsize = 48)
plt.xticks(fontsize = 48)
plt.yticks(fontsize = 48)
plt.grid(True)
plt.legend(fontsize = 48)
################################## END ANFIS ##################################
```

5. โปรแกรมสำหรับปรับปรุงพารามิเตอร์ของแบบจำลอง ARIMA_ANFIS สำหรับพยากรณ์ฝุ่น PM2.5 ล่วงหน้า ผ่านกระบวนการ Optimization ด้วยภาษา Matlab

```
%%%%%%%%%%%%%%%%%%%%%%%% GA-PART %%%%%%%%%%%%%%%%%%%%%%%%
%% MODELING %%
clc; close all;

%% Load data from csv file into array
opt = 'GA'
method = 'ARIMA_ANFIS'
city = 'RAYONG';
pmType = 'PM2dot5';
numMF = 2;
inputMF = 'gbellmf';
outputMF = 'constant';

fileName = strcat(pmType, '_', city, '_', method, '_RESID_DATASET.csv');
dataPath = strcat('/home/anupong/works/2022-trimester-0265/thesis/dataset/', \
           pmType, '/csv/use/', city, '/', method, '/');
dataFile = strcat(dataPath, fileName)
rawData = readtable(dataFile, 'VariableNamingRule', 'preserve');
dataset = table2array(rawData);

%% Split the dataset into train/test dataset
oneYearTest = 365;
numTrain = size(dataset, 1) - oneYearTest;
trainData = dataset(1:numTrain, :);
testData = dataset(numTrain + 1:end, :);

%% Create Sugino-FIS architechture
fisOption = genfisOptions('GridPartition');
fisOption.NumMembershipFunctions = numMF;
fisOption.InputMembershipFunctionType = inputMF;
fisOption.OutputMembershipFunctionType = outputMF;
```

```
%fisOption = genfisOptions('SubtractiveClustering');

fis = genfis(trainData(:, 1:end - 1), trainData(:, end), fisOption)

%%Tune Sugino-FIS parameters with GA.
numberOfWorkers = 32;
pool = parpool(numberOfWorkers);
mpiprofile on;

rng('shuffle');
epoch = 1000;
[in, out, rule] = getTunableSettings(fis)
tune = tunefisOptions('Method', 'ga', 'OptimizationType', 'tuning', 'UseParallel', true);
tune.MethodOptions.PopulationSize = 512%Default is 50 when nvars <= 5, 200 otherwise.
tune.MethodOptions.SelectionFcn = @selectionroulette;     %Default is 'selectionstochunif'.
tune.MethodOptions.CrossoverFcn = @crossovertwopoint;   %Default is 'crossoverscattered'.
muRate = 0.01;                                            %Default is 0.01.
tune.MethodOptions.MutationFcn = {@mutationuniform, muRate};%Default is 'mutationgaussian'.
tune.MethodOptions.MaxGenerations = epoch;                %Default is 100*nvars.
tune.MethodOptions.FitnessLimit = 0.1;                    %Default is -Inf.
%tune.MethodOptions.MaxStallGenerations = 20;             %Default is 50.

%K-Fold Cross-Validation.
rng('default');
tune.KFoldValue = 0;                                      %Default is 0.
tune.ValidationWindowSize = 2;                            %Default is 5.
tune.ValidationTolerance = 0.05;                  %Default is 0.1, value in the range [0,1].

%% Capture the time of learning process
tStart = tic;
[myfis, summary] = tunefis(fis, [in; out], trainData(:, 1:end - 1), trainData(:, end), tune)
elabTime = toc(tStart)

%% Save the ANFIS model
filePrefix = strcat(pmType, '_', city, '_ARIMA_FIS', opt);
modelFile = strcat(dataPath, filePrefix, '_MODEL.mat');
```

```
save(modelFile);

%% EVALUATING %%
clc; close all;

%% Load ARIMA_ANFIS model and all variales
opt = 'GA'
method = 'ARIMA_ANFIS'
city = 'RAYONG';
pmType = 'PM2dot5';
filePrefix = strcat(pmType, '_', city, '_ARIMA_FIS', opt);
fileName = strcat(filePrefix, '_MODEL.mat');
dataPath = strcat('/home/anupong/works/2022-trimester-0265/thesis/dataset/', \
            pmType, '/csv/use/', city, '/', method, '/');
modelFile = strcat(dataPath, fileName)
load(modelFile);

%% Evaluate the ANFIS models
predictTrain = evalfis(myfis, trainData(:, 1:end - 1));
predictTest = evalfis(myfis, testData(:, 1:end - 1));

%% Calculate the assessment of prediction
%R^2
rsqureTrain = 1 - (sum((trainData(:, end) - predictTrain).^2) / \
            sum((trainData(:, end) - mean(trainData(:, end))).^2));
rsqureTest = 1 - (sum((testData(:, end) - predictTest).^2) / \
            sum((testData(:, end) - mean(testData(:, end))).^2));
%RMSE
rmseTrain = sqrt(immse(predictTrain, trainData(:, end)));
rmseTest = sqrt(immse(predictTest, testData(:, end)));
%PRMSE
prmseTrain = (100 / mean(trainData(:, end))) * rmseTrain;
prmseTest= (100 / mean(testData(:, end))) * rmseTest;
%MAE
maeTrain = mae((trainData(:, end) - predictTrain));
maeTest = mae((testData(:, end) - predictTest));
```

```
%MAPE
mapeTrain = mean(abs((trainData(:, end) - predictTrain) ./ trainData(:, end))) * 100;
mapeTest = mean(abs((testData(:, end) - predictTest) ./ testData(:, end))) * 100;

%% Keep results
learningTime = elabTime
trainResults = [rsqureTrain; maeTrain; mapeTrain; rmseTrain; prmseTrain]
testResults = [rsqureTest; maeTest; mapeTest; rmseTest; prmseTest]

%% Save forecasted results to csv file
saveTrain = strcat(dataPath, filePrefix, '_TRAIN_PRED.csv');
saveTest = strcat(dataPath, filePrefix, '_TEST_PRED.csv');
%saveLearning = strcat(dataPath, filePrefix, '_LEARNING.csv');
saveTrainResults = strcat(dataPath, filePrefix, '_TRAIN_RESULTS.csv');
saveTestResults = strcat(dataPath, filePrefix, '_TEST_RESULTS.csv');
saveLearningTime = strcat(dataPath, filePrefix, '_LEARNING_TIME.txt');

writematrix(predictTrain, saveTrain);
writematrix(predictTest, saveTest);
%writematrix(summary.tuningOutputs.scores, saveLearning);
writematrix(trainResults, saveTrainResults);
writematrix(testResults, saveTestResults);
writematrix(learningTime, saveLearningTime);

%% Save matlab workspace
resultsPath = strcat('/home/anupong/works/2022-trimester-0265/thesis/result/matlab/PM/', \
            pmType, '/');
saveName = [datestr(now, 'yyyy-mmm-dd_HHMMSS'), '-', filePrefix, '_RESULTS.mat'];
strSave = append(resultsPath, saveName)
save(strSave);
```

```
%%%%%%%%%%%%%%%%%%%%%%% PSO-PART %%%%%%%%%%%%%%%%%%%%%%%
%% MODELING %%
clc; close all;

%% Load data from csv file into array
opt = 'PSO'
method = 'ARIMA_ANFIS'
city = 'RAYONG';
pmType = 'PM2dot5';
numMF = 2;
inputMF = 'gbellmf';
outputMF = 'constant';

fileName = strcat(pmType, '_', city, '_', method, '_RESID_DATASET.csv');
dataPath = strcat('/home/anupong/works/2022-trimester-0265/thesis/dataset/', \
            pmType, '/csv/use/', city, '/', method, '/');
dataFile = strcat(dataPath, fileName)
rawData = readtable(dataFile, 'VariableNamingRule', 'preserve');
dataset = table2array(rawData);

%% Split the dataset into train/test dataset
oneYearTest = 365;
numTrain = size(dataset, 1) - oneYearTest;
trainData = dataset(1:numTrain, :);
testData = dataset(numTrain + 1:end, :);

%% Create Sugino-FIS architechture
fisOption = genfisOptions('GridPartition');
fisOption.NumMembershipFunctions = numMF;
fisOption.InputMembershipFunctionType = inputMF;
fisOption.OutputMembershipFunctionType = outputMF;

%fisOption = genfisOptions('SubtractiveClustering');

fis = genfis(trainData(:, 1:end - 1), trainData(:, end), fisOption)
```

```
%%Tune Sugino-FIS parameters with PSO.
numberOfWorkers = 32;
pool = parpool(numberOfWorkers);
mpiprofile on;

rng('shuffle');
epoch = 1000;
[in, out, rule] = getTunableSettings(fis)
tune = tunefisOptions('Method', 'particleswarm', 'OptimizationType', 'tuning', 'UseParallel', true);
tune.MethodOptions.SwarmSize = 512          %Default is min(100,10*nvars),
                                            %where nvars is the number of variables.
tune.MethodOptions.MaxIterations = epoch;   %Default is 200*nvars.
tune.MethodOptions.ObjectiveLimit = 0.1;    %Default is -Inf.
tune.MethodOptions.MaxStallIterations = 50; %Default is 20.

%K-Fold Cross-Validation.
rng('default');
tune.KFoldValue = 0;                        %Default is 0.
tune.ValidationWindowSize = 2;              %Default is 5.
tune.ValidationTolerance = 0.05;            %Default is 0.1, value in the range [0,1].

%% Capture the time of learning process
tStart = tic;
[myfis, summary] = tunefis(fis, [in; out], trainData(:, 1:end - 1), trainData(:, end), tune)
elabTime = toc(tStart)

%% Save the ANFIS model
filePrefix = strcat(pmType, '_', city, '_ARIMA_FIS', opt);
modelFile = strcat(dataPath, filePrefix, '_MODEL.mat');
save(modelFile);

%% EVALUATING %%
clc; close all;

%% Load ANFIS model and all variales
opt = 'PSO'
```

```
method = 'ARIMA_ANFIS'
city = 'RAYONG';
pmType = 'PM2dot5';
filePrefix = strcat(pmType, '_', city, '_ARIMA_FIS', opt);
fileName = strcat(filePrefix, '_MODEL.mat');
dataPath = strcat('/home/anupong/works/2022-trimester-0265/thesis/dataset/', \
           pmType, '/csv/use/', city, '/', method, '/');
modelFile = strcat(dataPath, fileName)
load(modelFile);

%% Evaluate the ANFIS models
predictTrain = evalfis(myfis, trainData(:, 1:end - 1));
predictTest = evalfis(myfis, testData(:, 1:end - 1));

%% Calculate the assessment of prediction
%R^2
rsqureTrain = 1 - (sum((trainData(:, end) - predictTrain).^2) / \
           sum((trainData(:, end) - mean(trainData(:, end))).^2));
rsqureTest = 1 - (sum((testData(:, end) - predictTest).^2) / \
           sum((testData(:, end) - mean(testData(:, end))).^2));
%RMSE
rmseTrain = sqrt(immse(predictTrain, trainData(:, end)));
rmseTest = sqrt(immse(predictTest, testData(:, end)));
%PRMSE
prmseTrain = (100 / mean(trainData(:, end))) * rmseTrain;
prmseTest = (100 / mean(testData(:, end))) * rmseTest;
%MAE
maeTrain = mae((trainData(:, end) - predictTrain));
maeTest = mae((testData(:, end) - predictTest));
%MAPE
mapeTrain = mean(abs((trainData(:, end) - predictTrain) ./ trainData(:, end))) * 100;
mapeTest = mean(abs((testData(:, end) - predictTest) ./ testData(:, end))) * 100;

%% Keep results
learningTime = elabTime
trainResults = [rsqureTrain; maeTrain; mapeTrain; rmseTrain; prmseTrain]
```

```
testResults = [rsqureTest; maeTest; mapeTest; rmseTest; prmseTest]

%% Save forecasted results to csv file
saveTrain = strcat(dataPath, filePrefix, '_TRAIN_PRED.csv');
saveTest = strcat(dataPath, filePrefix, '_TEST_PRED.csv');
%saveLearning = strcat(dataPath, filePrefix, '_LEARNING.csv');
saveTrainResults = strcat(dataPath, filePrefix, '_TRAIN_RESULTS.csv');
saveTestResults = strcat(dataPath, filePrefix, '_TEST_RESULTS.csv');
saveLearningTime = strcat(dataPath, filePrefix, '_LEARNING_TIME.txt');

writematrix(predictTrain, saveTrain);
writematrix(predictTest, saveTest);
%writematrix(summary.tuningOutputs.scores, saveLearning);
writematrix(trainResults, saveTrainResults);
writematrix(testResults, saveTestResults);
writematrix(learningTime, saveLearningTime);

%% Save matlab workspace
resultsPath = strcat('/home/anupong/works/2022-trimester-0265/thesis/result/matlab/PM/', \
            pmType, '/');
saveName = [datestr(now, 'yyyy-mmm-dd_HHMMSS'), '-', filePrefix, '_RESULTS.mat'];
strSave = append(resultsPath, saveName)
save(strSave);
```

ภาคผนวก ข

บทความวิจัยที่ได้รับการตีพิมพ์เผยแพร่ในระหว่างศึกษา

รายชื่อบทความวิจัยที่ได้รับการตีพิมพ์เผยแพร่ในระหว่างศึกษา

อนุพงษ์ บรรจงการ, นิตยา เกิดประสพ และกิตติศักดิ์ เกิดประสพ. (2560). การวิเคราะห์ข้อมูลบันทึกรายวันของระบบคำนวณสมรรถนะสูงเพื่อเพิ่มประสิทธิภาพด้วยโครงข่าย เบย์เซียน: กรณีศึกษาระบบคำนวณสมรรถนะสูงของเนคเทค. The 2nd Santapol Knowledge Research to Development, วิทยาลัยสันตพล, จังหวัดอุดรธานี, ประเทศไทย, วันที่ 25 พฤศจิกายน 2560.

Anupong Banjongkan, Nittaya Kerdprasop and Kittisak Kerdprasop. (2018). Bayesian Decision Network Approaches for HPC Log Analysis: Case Study of NECTEC HPC System. The International Conference on Engineering and Applied Sciences (TICESA'2018), Bangkok, Thailand. February 22 – 24, 2018.

Anupong Banjongkan, Nittaya Kerdprasop and Kittisak Kerdprasop. (2018). Performance Evaluation of Ensemble Decision Tree Techniques with HPC-workload Dataset. 4th International Conference on Computer, Communication and Control Technology (I4CT'2018). Krabi, Thailand. March 20 – 22, 2018.

Anupong Banjongkan, Watthana Pongsena, Ratiporn Chanklan, Nittaya Kerdprasop, and Kittisak Kerdprasop. (2018). Multi-label Classification of High Performance Computing Workload with Variable Transformation. International Journal of Machine Learning and Computing. Vol. 8, No. 6, pp. 536 – 541. December 2018.

Anupong Banjongkan, Watthana Pongsena, Nittaya Kerdprasop, and Kittisak Kerdprasop. (2020). A Comparative Study of Learning Techniques with Convolutional Neural Network Based on HPC-Workload Dataset. International Journal of Machine Learning and Computing. Vol. 10, No. 1, pp. 10 – 17. January 2020.

รายชื่อบทความวิจัยที่ได้รับการตีพิมพ์เผยแพร่ในระหว่างศึกษา (ต่อ)

Anupong Banjongkan, Watthana Pongsena, Nittaya Kerdprasop, and Kittisak Kerdprasop. (2021). A Study of Job Failure Prediction at Job Submit-State and Job Start-State in High-Performance Computing System: Using Decision Tree Algorithms. Journal of Advances in Information Technology. Vol. 12, No. 2, pp. 84 – 92. May 2021.

Anupong Banjongkan, Anusara Hirunyawanakul, Nittaya Kerdprasop and Kittisak Kerdprasop. (2023). PM2.5 Forecasting Model based on Linear and Non-linear Hybrid Algorithm. 15th International Conference on Knowledge and Smart Technology (KST'2023), Phuket, Thailand, February 21 – 24, 2023.

Anupong Banjongkan, Nittaya Kerdprasop, and Kittisak Kerdprasop. (2023). Renewable Energy Forecasting with Hybrid Nonlinear Model (ANFIS): Case Study of Wind Speed in Thailand. International Journal of Smart Grid and Clean Energy. Vol. 12, No. 2, pp. 19 – 29. April 2023.

การวิเคราะห์ข้อมูลบันทึกรายวันของระบบคำนวณสมรรถนะสูงเพื่อเพิ่มประสิทธิภาพด้วยโครงข่ายเบย์เซียน: กรณีศึกษาระบบคำนวณสมรรถนะสูงของเนคเทค

HPC LOG ANALYSIS FOR EFFICIENCY IMPROVEMENT USING BAYESIAN NETWORK: A CASE STUDY OF NECTEC HPC

อนุพงษ์ บรรจงการ [1], นิตยา เกิดประสพ และ กิตติศักดิ์ เกิดประสพ [2]

ANUPONG BANJONGKAN, NITTAYA KERDPRASOP AND KITTISAK KERDPRASOP

บทคัดย่อ

ประสิทธิภาพ (Efficiency) ของระบบคำนวณสมรรถนะสูง (Hight Performance Computing: HPC) เป็นสิ่งที่ต้องให้ความสำคัญอย่างยิ่ง เนื่องจากระบบคำนวณสมรรถนะสูงใช้พลังงานไฟฟ้าสูงมากสำหรับขับเคลื่อนระบบฯ ซึ่งประสิทธิภาพของระบบฯ ขึ้นอยู่กับอัตราความสำเร็จของงานที่ประมวลผล นั่นหมายความว่าถ้าระบบฯ มีอัตราความสำเร็จของงานที่ประมวลผลสูงก็แสดงถึงระบบฯ มีประสิทธิภาพสูง สะท้อนถึงการใช้พลังงานไฟฟ้าอย่างคุ้มค่าในการขับเคลื่อนระบบฯ ในทางตรงกันข้าม ถ้าระบบฯ มีอัตราความสำเร็จของงานที่ประมวลผลต่ำก็แสดงถึงระบบฯ มีประสิทธิภาพต่ำ ซึ่งสะท้อนถึงการใช้พลังงานไฟฟ้าอย่างไม่คุ้มค่าในการขับเคลื่อนระบบฯ ดังนั้นงานวิจัยนี้ ต้องการวิเคราะห์และทำนายลักษณะงานที่มีโอกาสประมวลผลสำเร็จมากที่สุดบนระบบคำนวณสมรรถนะสูง เพื่อเพิ่มอัตราความสำเร็จของงานที่ประมวลผล ซึ่งอัตราความสำเร็จของงานเป็นปัจจัยหลักต่อประสิทธิภาพของระบบคำนวณสมรรถนะสูง ผู้วิจัยได้นำเสนอกระบวนการทำเหมืองข้อมูล (Data mining) ผ่านแบบจำลองโครงข่ายเบย์เซียน (Bayesian Network) ร่วมกับผู้เชี่ยวชาญ (Expert) ด้านระบบคำนวณสมรรถนะสูง โดยใช้ข้อมูลบันทึกรายวัน (Log) ของระบบคำนวณสมรรถนะสูงเป็นชุดข้อมูลสำหรับการวิจัย ซึ่งงานวิจัยนี้ได้รับการอนุเคราะห์ข้อมูลบันทึกรายวันของระบบคำนวณสมรรถนะสูงของเนคเทค (NECTEC) เพื่อใช้เป็นชุดข้อมูลสำหรับการวิจัย

ผลการวิจัย 1) แสดงให้เห็นว่าแบบจำลองโครงข่ายเบย์เซียนที่ผ่านการปรับโครงสร้างตามความเห็นของผู้เชี่ยวชาญฯ มีประสิทธิภาพดีที่สุด โดยที่ให้ค่าความถูกต้อง (Accuracy) ต่อการทำนายผลข้อมูลสูงถึง 79% จากการประเมินแบบแบ่งข้อมูลเรียนรู้ 70% และข้อมูลทดสอบ 30% ของข้อมูลทั้งหมด 2) ผลการวิเคราะห์และทำนายผลข้อมูลผ่านการประมวลผลเชิงตรรกะ (Inference) ของแบบจำลองโครงข่ายเบย์เซียนที่นำเสนอ แสดงให้เห็นถึงลักษณะงานที่มีโอกาสประมวลผลสำเร็จมากที่สุดบนระบบคำนวณสมรรถนะสูงของเนคเทค คือ งานที่ใช้หน่วยประมวลผล 4 – 8 CPU, ใช้หน่วยความจำ 1 – 4 GB และเลือกใช้แถวคอยแบบ SHORT กล่าวคือเป็นงานขนาดกลาง และใช้เวลาในการประมวลระยะสั้น

ข้อมูลที่ได้จากการวิจัย จะใช้เป็นข้อมูลเพื่อเพิ่มอัตราความสำเร็จของงานในระบบฯ ทำให้ระบบคำนวณสมรรถนะสูงของเนคเทคมีประสิทธิภาพสูงขึ้น และสะท้อนต่อการใช้พลังงานไฟฟ้าสำหรับขับเคลื่อนระบบฯ อย่างคุ้มค่าที่สุด

คำสำคัญ : โครงข่ายเบย์เซียน, การประเมินประสิทธิภาพ, ข้อมูลบันทึกรายวัน, เนคเทค

[1] นักศึกษาปริญญาเอก สาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

[2] ที่ปรึกษาวิทยานิพนธ์ อาจารย์ประจำสาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี
Email: banjongkan@gmail.com

ABSTRACT

The performance of High Performance Computing (HPC) is a crucial need as it is a highly power consumption system. The performance of an HPC system depends on the success rate of the applications processed. This means that the high success rate of processing, the high performance of the HPC system. On the other hand, the low success rate of processing, the low performance of the HPC system. This reflects the efficiently and inefficiently use of electric power to drive the HPC system. Therefore, the main propose of this research is to analyze and predict the characteristics of the job that are most likely to be a successful job in order to increase the job success rate of HPC systems. In this research, the researchers have presented the data mining techniques through a Bayesian Network model incorporated with expert, who have highly experiences in HPC area. The dataset used in this research is log data of a real-world HPC system, which have been supported by NECTEC. The results of our experiments demonstrate that firstly, the Bayesian network model, which restructured by HPC expertise perform the best accuracy for predicting results (approximately 79%) evaluated by dividing the dataset into 70% for training and 30% for testing the model. Secondly, the results from using the proposed model for analyzing and predicting the characteristics of a job that is most likely to be the successful job on an HPC system are the job that using 4 – 8 CPUs, using 1 – 4 GB of RAM, and selecting a SHORT queue. In other words, the job has to be a medium job and using a short period of time for processing this job. The information obtained from this research will increase the job success rate, which indicate to the higher efficiently use of electric power of the NECTEC HPC system. This also reflect on the worthwhile power consumption for the system.

Keywords: Bayesian Network, Performance Evaluation, Log, NECTEC

บทนำ

ระบบคำนวณสมรรถนะสูง (Hight Performance Computing: HPC) คือระบบที่นำเครื่องคอมพิวเตอร์หลายๆ เครื่องมาทำงานร่วมกันเพื่อเพิ่มกำลังการคำนวณ โดยที่ทุกๆ เครื่องคอมพิวเตอร์ที่อยู่ในระบบฯ จะเชื่อมต่อถึงกันผ่านโครงข่ายภายใน (Interconnection Network) และมีโปรแกรมตัวกลาง (Middle-ware) ค่อยบริหารจัดการทรัพยากรของระบบคำนวณสมรรถนะสูง ทำให้เสมือนเป็นเครื่องคอมพิวเตอร์เครื่องเดียว (Zhang et al., 2012) (Uthayopas et al., 1998) ระบบคำนวณสมรรถนะสูงถูกนำมาใช้คำนวณงานทางด้านวิทยาศาสตร์และวิศวกรรมศาสตร์ ซึ่งเป็นงานที่มีความซับซ้อนสูงและมีขนาดใหญ่ เช่น งานด้านสภาพอากาศภูมิอากาศ (Deconinck et al., 2017), งานด้านพันธุวิศวกรรม (Ahmed et al., 2015), งานด้านฟิสิกส์พลังงานสูง (O'Brien et al., 2014) และงานด้านวิศวกรรมการบินและอวกาศ (Gao et al., 2014) เป็นต้น

การบริการระบบคำนวณสมรรถนะสูงต้องให้ความสำคัญต่อประสิทธิภาพ (Efficiency) ของระบบฯ เป็นอย่างมาก เพราะมีผลต่อพลังงานไฟฟ้าที่ใช้สำหรับขับเคลื่อนระบบฯ โดยที่ประสิทธิภาพของระบบคำนวณสมรรถนะสูงนั้นประเมินได้จากอัตราความสำเร็จของงานที่ประมวลผล (Yuan et al., 2012) ผู้วิจัยขอยกตัวอย่างความสำคัญของอัตราความสำเร็จของงานที่ประมวลผล ต่อ

ประสิทธิภาพของระบบคำนวณสมรรถนะสูง ดังนี้ สมมุติระบบคำนวณสมรรถนะสูงที่สนใจสามารถให้บริการได้เต็มความสามารถของทรัพยากร 100% ต่อช่วงเวลาที่สนใจ แต่พบว่ามีอัตราความสำเร็จของงานที่ประมวลผลเพียงแค่ 30% ของงานที่เข้าประมวลผลทั้งหมด จากผลดังกล่าวสามารถเปรียบได้ว่าระบบคำนวณสมรรถนะสูงที่สนใจต้องสูญเปล่าพลังงานไฟฟ้าที่ใช้ขับเคลื่อนระบบฯ ถึง 70% ของพลังงานไฟฟ้าที่ใช้ขับเคลื่อนระบบฯ ทั้งหมด จากที่ผู้วิจัยได้ยกตัวอย่างจะพบว่าประสิทธิภาพของระบบคำนวณสมรรถนะสูงนั้นมีผลต่อพลังงานไฟฟ้าที่ใช้สำหรับขับเคลื่อนระบบฯ กล่าวคือ ถ้าระบบคำนวณสมรรถนะสูงมีอัตราความสำเร็จของงานที่ประมวลผลสูงก็แสดงถึงระบบฯ มีประสิทธิภาพสูง สะท้อนถึงการใช้พลังงานไฟฟ้าอย่างคุ้มค่าในการขับเคลื่อนระบบฯ ในทางตรงกันข้าม ถ้าระบบการคำนวณสมรรถนะสูงมีอัตราความสำเร็จของงานที่ประมวลผลต่ำก็แสดงถึงระบบฯ มีประสิทธิภาพต่ำ ซึ่งสะท้อนถึงการใช้พลังงานไฟฟ้าอย่างไม่คุ้มค่าในการขับเคลื่อนระบบฯ ดังนั้น งานวิจัยนี้มุ่งเป้าไปที่การหาวิธีเพิ่มอัตราความสำเร็จของงานที่ประมวลผล ซึ่งเป็นปัจจัยหลักต่อประสิทธิภาพของระบบคำนวณสมรรถนะสูง

ผู้วิจัยได้นำเสนอกระบวนการทำเหมืองข้อมูล (Data mining) ผ่านแบบจำลองโครงข่ายเบย์เซียน (Bayesian Network) (Friedman et al., 1997) ร่วมกับผู้เชี่ยวชาญ (Expert) ด้านระบบคำนวณสมรรถนะสูง โดยใช้ข้อมูลบันทึกรายวัน (Log) ของระบบคำนวณสมรรถนะสูงเป็นชุดข้อมูลสำหรับการวิจัย ซึ่งงานวิจัยนี้ ได้รับการอนุเคราะห์ข้อมูลบันทึกรายวันของระบบคำนวณสมรรถนะสูงจากศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติหรือเนคเทค (NECTEC) ผ่านโครงการ National e-Science Infrastructure Consortium มาใช้เป็นชุดข้อมูลสำหรับการทำวิจัย ซึ่งข้อมูลดังกล่าวเป็นชุดข้อมูลที่เริ่มบันทึกตั้งแต่กลางปี 2555 ถึงกลางปี 2560 ทำให้มีขนาดของข้อมูลจำนวน 389,765 แถว และมีจำนวน 27 ตัวแปร

วัตถุประสงค์

1. สร้างแบบจำลองโครงข่ายเบย์เซียนที่มีประสิทธิภาพสูงโดยใช้เครื่องมือ GeNIe ร่วมกับผู้เชี่ยวชาญฯ โดยมีข้อมูลบันทึกรายวันของระบบคำนวณสมรรถนะสูงของเนคเทคเป็นชุดข้อมูลสำหรับวิจัย

2. ใช้แบบจำลองฯ ที่นำเสนอ วิเคราะห์และทำนายผลข้อมูลของช่วงข้อมูลที่สนใจ ของชุดข้อมูลบันทึกรายวันของระบบคำนวณสมรรถนะสูงของเนคเทค เพื่อหาลักษณะงานที่มีโอกาสประมวลผลสำเร็จมากที่สุดบนระบบฯ เพื่อเพิ่มอัตราความสำเร็จของงาน ซึ่งเป็นปัจจัยหลักต่อประสิทธิภาพของระบบคำนวณสมรรถนะสูง

การทบทวนวรรณกรรมและทฤษฎีที่เกี่ยวข้อง

วรรณกรรมที่ใช้การวิเคราะห์ข้อมูลบันทึกรายวันส่วนใหญ่มักจะวิเคราะห์หาต้นหตุของปัญหา (Root Cause) ที่เกิดขึ้นกับอุปกรณ์หรือโปรแกรม (Zawawy et al., 2010) (Anamika el al., 2015) เพื่อแก้ไขปัญหาที่เกิดขึ้นได้ตรงจุดและให้อุปกรณ์หรือโปรแกรมสามารถกลับมาทำงานได้อย่างปกติโดยเร็วที่สุด มีงานวิจัยหลายงาน (Sîrbu & Babaoglu, 2015) (Sheng et al., 2017) ที่นำข้อมูลบันทึกรายวันของอุปกรณ์ของระบบคำนวณสมรรถนะสูงมาวิเคราะห์ผลข้อมูลเพื่อทำนายถึงโอกาสที่อุปกรณ์ซึ่งเป็นส่วนประกอบของระบบฯ จะมีปัญหาล่วงหน้า และมีงานวิจัย (Yoon et al., 2015) ที่ใช้ข้อมูลบันทึกรายวันจากระบบคำนวณสมรรถนะสูงมาศึกษาและวิเคราะห์ปัญหาที่ทำให้งานประมวลผลไม่สำเร็จบนระบบฯ โดยใช้วิธีทางสถิติ

งานวิจัยนี้ มีทฤษฎีหลักที่เกี่ยวข้องคือ 1) ทฤษฎีการทำเหมือนข้อมูลผ่านโครงข่ายเบย์เซียน และ 2) ทฤษฎีของ Confusion Matrix ซึ่งใช้สำหรับคำนวณตัววัดประสิทธิภาพความถูกต้อง (Accuracy) และ ตัววัดประสิทธิภาพ ROC (Receiver Operating Characteristic) ของแบบจำลองฯ ในงานวิจัยนี้

1. โครงข่ายเบย์เซียน

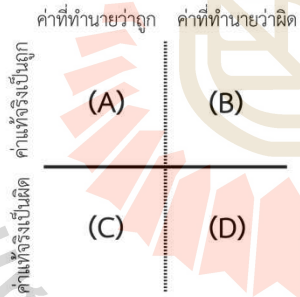
โครงข่ายเบย์เซียน เป็นโครงข่ายสำหรับวิเคราะห์และทำนายผลข้อมูลผ่านความน่าจะเป็นตามทฤษฎีพื้นฐาน

ของเบย์ ดังสมการที่ *(1)*

$$P(h|e) = \frac{P(e|h)P(h)}{P(e)} \qquad (1)$$

โดยที่ $P(h|e)$ แทนค่า Posterior, $P(e|h)$ แทนค่า Likelihood, $P(h)$ แทนค่า Prior และ $P(e)$ แทนค่า Probability of Evidence

แบบจำลองโครงข่ายเบย์เซียนแสดงผ่านกราฟไม่เป็นวง (Directed Acyclic Graph: DAC) และประมวลผลเชิงตรรกะหรือการอนุมาน (Inference) เพื่อแสดงความสัมพันธ์ของข้อมูลผ่านตารางความน่าจะเป็นแบบมีเงื่อนไข (Conditional Probability Table: CPT) โดยพิจาณาตามคุณสมบัติของ Markov Blanket เพื่อลดความซับซ้อนและระยะเวลาการประมวล จากการทำงานดังกล่าวทำให้แบบจำลองโครงข่ายเบย์เซียนเป็นหนึ่งในเครื่องมือการทำเหมืองข้อมูลที่มีประสิทธิภาพสูงและได้รับการยอมรับจนถูกนำไปใช้งานในหลายสาขา เช่น งานด้านการแพทย์ (Loghmanpour et al., 2016) งานด้านสิ่งแวดล้อม (VARKEY et al., 2013) หรืองานด้านความมั่นคง (Shin et al., 2015) เป็นต้น



ภาพประกอบที่ 1 Confusion Matrix สำหรับจำแนกข้อมูล 2 กลุ่ม

2. การประเมินประสิทธิภาพ

การประเมินประสิทธิภาพแบบจำลองฯ ในงานวิจัยนี้ ผู้วิจัยเลือกใช้ตัววัดประสิทธิภาพ 2 ตัววัด คือ ตัววัดประสิทธิภาพความถูกต้อง และ ตัววัดประสิทธิภาพ ROC โดยที่ตัววัดทั้ง 2 (Sokolova et al., 2006) ถูกคำนวณจาก Confusion Matrix (Caelen et al., 2017) แสดงดังภาพประกอบที่ 1. โดยที่ (A) คือกลุ่มข้อมูลที่ทำนายว่าถูกตรงกับค่าแท้จริงของข้อมูลเป็นถูก (True Positive), (B) คือกลุ่มข้อมูลที่ทำนายว่าผิดตรงกันข้ามกับค่าแท้จริงของข้อมูลเป็นถูก (False Negative), (C) คือกลุ่มข้อมูลที่ทำนายว่าถูกตรงกันข้ามกับค่าแท้จริงของข้อมูลเป็นผิด (False Positive) และ (D) คือกลุ่มข้อมูลที่ทำนายว่าผิดตรงกับค่าแท้จริงของข้อมูลเป็นผิด (True Negative) โดยมีเส้นประแนวแกนตั้งเป็นเส้นบรรทัดฐาน (Threshold) สำหรับจำแนกข้อมูล 2 กลุ่ม

2.1 ตัวประสิทธิภาพวัดความถูกต้อง (Accuracy)

ตัววัดประสิทธิภาพความถูกต้องใช้ประเมินประสิทธิภาพแบบจำลองในการทำนายหรือการจัดกลุ่มข้อมูล โดยมีค่าอยู่ระหว่าง 0 ถึง 1 ซึ่งถ้าค่าของตัววัดประสิทธิภาพความถูกต้องเข้าใกล้ 1 แสดงถึงแบบจำลองที่ประเมินมีประสิทธิภาพสูง ตรงกันข้าม ถ้าค่าของตัววัดประสิทธิภาพความถูกต้องเข้าใกล้ 0 แสดงถึงแบบจำลองที่ประเมินมีประสิทธิภาพต่ำ ซึ่งสามารถคำนวณตัววัดประสิทธิภาพความถูกต้อง ดังสมการที่ *(2)*

$$\text{Accuracy} = \frac{(A)+(D)}{(A)+(B)+(C)+(D)} \qquad (2)$$

2.2 ตัววัดประสิทธิภาพ ROC

ตัววัดประสิทธิภาพ ROC (Receiver Operating Characteristic) แสดงผลในรูปของกราฟเส้น โดยมีแกน y แทนค่าของกลุ่มข้อมูลที่ทำนายว่าถูกตรงกับค่าแท้จริงของข้อมูลเป็นถูก (True Positive) ต่อกลุ่มข้อมูลที่มีค่าแท้จริงเป็นถูกทั้งหมด เรียกว่าค่า True Positive Rate: TPR ดังสมการที่ *(3)* และมีแกน x แทนค่าของกลุ่มข้อมูลที่ทำนายว่าถูกตรงกันข้ามกับค่าแท้จริงของข้อมูลเป็นผิด (False Positive) ต่อกกลุ่มข้อมูลที่มีค่าแท้จริงเป็นผิดทั้งหมด เรียกว่าค่า False Positive Rate: FPR ดังสมการที่ *(4)*

$$TPR = \frac{(A)}{(A)+(B)} \qquad (3)$$

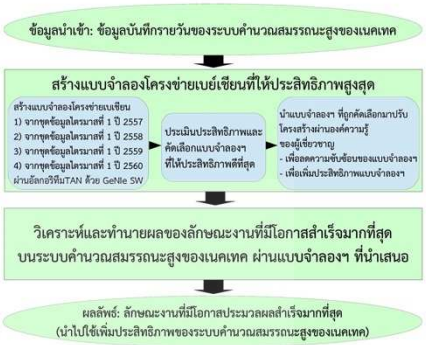
$$FPR = \frac{(D)}{(C)+(D)} \quad (4)$$

จากกราฟของตัววัดประสิทธิภาพ ROC จะใช้ค่าของพื้นที่ใต้กราฟ (Area Under Curve) หรือ AUC เป็นตัวประเมินประสิทธิภาพ โดยที่ค่าของ AUC จะมีค่าอยู่ระหว่าง 0 – 1 ถ้าค่า AUC ของตัววัดประสิทธิภาพ ROC มีค่าเข้าใกล้ 1 แสดงว่าแบบจำลองที่ประเมินมีประสิทธิภาพสูง แต่ถ้าค่าของ AUC ของตัววัดประสิทธิภาพ ROC มีค่าต่ำกว่า 0.5 แสดงว่าประสิทธิภาพของแบบจำลองที่ประเมินมีประสิทธิภาพต่ำ

กรอบแนวคิดที่ใช้ในการดำเนินงานวิจัย

ผู้วิจัยได้ศึกษาทฤษฎีและบททวนวรรณกรรมที่เกี่ยวข้อง พบว่ายังไม่มีวรรณกรรมใดที่ใช้วิธีการทำเหมืองข้อมูลผ่านแบบจำลองโครงข่ายเบย์เซียนบนชุดข้อมูลบันทึกรายวันของระบบคำนวณสมรรถนะสูง เพื่อวิเคราะห์และทำนายผลของข้อมูลที่สนใจ ดังนั้นผู้วิจัยจึงยึดถือเอาการทำเหมืองข้อมูลผ่านแบบจำลองโครงข่ายเบย์เซียน ซึ่งเป็นเครื่องมือที่มีความรวดเร็วและมีประสิทธิภาพสูงต่อการทำนายผลข้อมูลตามกระบวนการการทำเหมืองข้อมูล มาใช้เป็นเครื่องมือสำหรับวิเคราะห์และทำนายคุณลักษณะของงานที่มีโอกาสประมวลผลสำเร็จมากที่สุด บนชุดข้อมูลบันทึกรายวันของระบบคำนวณสมรรถนะสูงของเนคเทค เพื่อเพิ่มอัตราความสำเร็จของงานบนระบบฯ ซึ่งเป็นปัจจัยหลักต่อประสิทธิภาพของการให้บริการระบบการคำนวณสมรรถนะสูง

งานวิจัยนี้ ได้แบ่งแนวคิดของการวิจัยเป็นสองส่วนหลัก คือ 1) สร้างแบบจำลองโครงข่ายเบย์เซียนที่ให้ประสิทธิภาพสูงสุดเพื่อนำไปใช้ในส่วนที่ 2 ของงานวิจัย การสร้างแบบจำลองฯ ที่ให้ประสิทธิภาพสูงสุด ใช้วิธีคัดเลือกแบบจำลองโครงข่ายเบย์เซียนที่มีประสิทธิภาพดีที่สุดจาก 4 แบบจำลองฯ ซึ่งแต่ละแบบจำลองฯ สร้างจากชุดข้อมูลไตรมาสที่ 1 (มกราคม, กุมภาพันธ์ และมีนาคม) ของปี 2557, 2558, 2559 และ 2560 จากข้อมูลบันทึกรายวันของระบบคำนวณสมรรถนะสูงของเนคเทค



ภาพประกอบที่ 2 กรอบแนวคิดกระบวนการวิจัย

ตามลำดับ ซึ่งแต่ละชุดข้อมูลเป็นช่วงข้อมูลที่สนใจโดยเลือกผ่านคำแนะนำของผู้เชี่ยวชาญ จากนั้นนำแบบจำลองโครงข่ายเบย์เซียนที่คัดเลือกมาปรับโครงสร้างของแบบจำลองเพื่อลดความซับซ้อนและเพิ่มประสิทธิภาพให้ได้แบบจำลองที่มีประสิทธิภาพสูงสุด 2) นำแบบจำลองโครงข่ายเบย์เซียนจากการวิจัยในส่วนที่ 1 มาใช้เป็นแบบจำลองสำหรับวิเคราะห์และทำนายผลผ่านกระบวนการประมวลผลเชิงตรรกะของทฤษฎีโครงข่ายเบย์เซียน บนชุดข้อมูลบันทึกรายวันของระบบคำนวณสมรรถนะสูงของเนคเทค เพื่อหาลักษณะงานที่มีโอกาสประมวลผลสำเร็จสูงที่สุด เพื่อเพิ่มอัตราความสำเร็จของงานในระบบฯ ซึ่งเป็นปัจจัยหลักต่อประสิทธิภาพของการให้บริการระบบคำนวณสมรรถนะสูง โดยกรอบแนวคิดของการวิจัยแสดงดังภาพประกอบที่ 2

วิธีการดำเนินการวิจัย

เครื่องมือและชุดข้อมูลสำหรับวิจัย

1. เครื่องมือ

งานวิจัยนี้ ผู้วิจัยเลือกใช้เครื่องมือสำหรับสร้างและวิเคราะห์ผลข้อมูลผ่านแบบจำลองโครงข่ายเบย์เซียนที่มีชื่อว่า GeNie เวอร์ชัน 2.2.2017.0 (32-bit academic) ถูกพัฒนาบนพื้นฐานของภาษาคอมพิวเตอร์ C++ โดยบริษัท BAYESFUSION, LLC ซึ่งเป็นเครื่องมือที่ได้รับความนิยมอย่างกว้างขวางสำหรับงานที่เกี่ยวข้องกับการทำ

เหมืองข้อมูลผ่านโครงข่ายเบย์เซียน

สำหรับการสร้างแบบจำลองโครงข่ายเบย์เซียนด้วยเครื่องมือ GeNIe ผู้วิจัยเลือกใช้อัลกอริทึม TAN (Tree Augmented Naive Bayes) ในขั้นตอนการเรียนรู้ชุดข้อมูลเพื่อสร้างโครงสร้างของแบบจำลองฯ ซึ่ง อัลกอริทึม TAN เป็นอัลกอริทึมแบบประมาณค่า (Approximate Algorithm) นิยมใช้กับชุดข้อมูลที่มีขนาด 1,000 แถวขึ้นไป เพื่อลดระยะเวลาของการประมวลผล ซึ่งเหมาะสมสำหรับใช้กับชุดข้อมูลของงานวิจัยนี้

2. ชุดข้อมูล

ชุดข้อมูลที่ใช้สำหรับงานวิจัย เป็นข้อมูลบันทึกรายวันของระบบคำนวณสมรรถนะสูงของเนคเทค โดยข้อมูลทั้งหมดมีระยะเวลาบันทึกรายวันประมาณ 5 ปี เริ่มเก็บบันทึกตั้งแต่กลางปี 2555 จนถึงกลางปี 2560 และมีจำนวนตัวแปรในชุดข้อมูลทั้งหมด 27 ตัวแปร

ตารางที่ 1 แสดงรายละเอียดของตัวแปรที่สนใจ

No.	ชื่อตัวแปร	ความหมาย	ชนิด
1	QUEUE_TYPE	ประเภทของระบบแถวคอย	ตั้งต้น
2	CPU_USAGE	จำนวนหน่วยประมวลผลที่ใช้	ตั้งต้น
3	FINISH_STATUS	สถานะเมื่อสิ้นสุดการประมวลผล	ตั้งต้น
4	CPU_TIME	จำนวนเวลาที่ใช้ประมวลผลคูณกับจำนวนหน่วยประมวลผลที่ใช้	ตั้งต้น
5	MEMORY_USAGE	ขนาดหน่วยความจำที่โปรแกรมใช้งาน	ตั้งต้น
6	VMEMORY_USAGE	ขนาดหน่วยความจำที่โปรแกรมและสภาพแวดล้อมของโปรแกรมใช้งาน	ตั้งต้น
7	WALL_TIME	จำนวนเวลาที่รอในระบบแถวคอยบวกกับจำนวนเวลาที่ใช้ประมวลผล	ตั้งต้น
8	QUEING_TIME	จำนวนเวลาที่รอในระบบแถวคอย	สังเคราะห์

กระบวนการวิจัย

กระบวนการวิจัย ผู้วิจัยได้แบ่งเป็น 3 ขั้นตอนหลักดังนี้ 1) ขั้นตอนปรับชุดข้อมูลดิบและเลือกช่วงข้อมูลที่สนใจเพื่อนำมาเป็นชุดข้อมูลสำหรับวิจัยในขั้นตอนวิจัยถัดไป 2) ขั้นตอนการสร้างแบบจำลองโครงข่ายเบย์เซียนเพื่อให้ได้ประสิทธิภาพสูงที่สุด โดยใช้ชุดข้อมูลสำหรับวิจัยจากขั้นตอนวิจัยก่อนหน้า 3) ขั้นตอนวิเคราะห์และทำนายผลข้อมูลที่สนใจผ่านแบบจำลองโครงข่ายเบย์เซียนที่นำเสนอในขั้นตอนวิจัยที่ 2)

ตารางที่ 2 แสดงการแบ่งช่วงข้อมูลตัวเลขของ 6 ตัวแปร

ประเภท	ตัวแปร	ช่วงข้อมูล
จำนวนหน่วยประมวลผล	CPU_USAGE	1) 0 CPU 2) 1 ถึง 4 CPU 3) 4 ถึง 8 CPU 4) 8 ถึง 16 CPU 5) 16 ถึง 32 CPU 6) มากกว่า 32 CPU
ขนาดของหน่วยความจำ	MEMORY_USAGE, VMEMORY_USAGE	1) 0 Gb 2) น้อยกว่า 1 Gb 3) 1 ถึง 4 Gb 4) 4 ถึง 8 Gb 5) 8 ถึง 12 Gb 6) มากกว่า 12 Gb
เวลา	QUEING_TIME, CPU_TIME, WALL_TIME	1) 0 ชม. 2) น้อยกว่า 3 ชม. 3) 3 ถึง 6 ชม. 4) 6 ถึง 12 ชม. 5) 12 ถึง 24 ชม. 6) มากกว่า 24 ชม.

1. ขั้นตอนปรับชุดข้อมูลดิบและเลือกชุดข้อมูล

1.1 ปรับชุดข้อมูลดิบ การปรับชุดข้อมูลดิบเพื่อให้เกิดความเหมาะสมและเพื่อให้สามารถนำชุดข้อมูลไปใช้กับเครื่องมือ GeNIe ได้อย่างมีประสิทธิภาพที่สุด โดยได้ปรับชุดข้อมูลดิบดังนี้ 1) ทำการคัดเลือกเฉพาะตัวแปรที่สนใจโดยผ่านคำแนะนำจากผู้เชี่ยวชาญ ผู้วิจัยได้คัดเลือกตัวแปรที่สนใจจำนวน 8 ตัวแปร โดยที่ 7 ตัวแปร คือ QUEUE_TYPE, FINISH_STATUS, MEMORY_USAGE, VMEMORY_USAGE, CPU_TIME, QUEING_TIME และ WALL_TIME เป็นตัวแปรตั้งต้นของชุดข้อมูล และอีก 1 ตัว คือ CPU_USAGE แปรเป็นตัวแปรสังเคราะห์ แสดงดังตารางที่ 1 2) ปรับข้อมูลของตัวแปรที่มีค่าเป็นข้อมูลตัวเลขให้เป็นค่าของช่วงข้อมูลที่สนใจ โดยมีจำนวนตัวแปรที่มีค่า

เป็นข้อมูลตัวเลขทั้งหมด 6 ตัวแปร คือ CPU_USAGE, MEMORY_USAGE, VMEMORY_USAGE, CPU_TIME, QUEING_TIME และ WALL_TIME โดยแสดงการแบ่งช่วงข้อมูลที่สนใจของแต่ละตัวแปร ดังตารางที่ 2

1.2 เลือกชุดข้อมูลจำนวน 4 ชุด เพื่อใช้สำหรับวิจัย ในขั้นตอนวิจัยถัดไป ผู้วิจัยได้เลือกชุดข้อมูลผ่านคำแนะนำของผู้เชี่ยวชาญ โดยเลือกตามช่วงข้อมูลไตรมาสที่ 1 ของปี 2557, 2558, 2559 และ 2560 โดยมีขนาด 2,960 แถว, 5,685 แถว, 6,232 แถว และ 4,039 แถว ตามลำดับ เพราะเนื่องจากช่วงไตรมาสที่ 1 ของทุกปี จะมีจำนวนงานที่เข้าประมวลผลบนระบบคำนวณสมรรถนะสูงของเนคเทคมากที่สุด

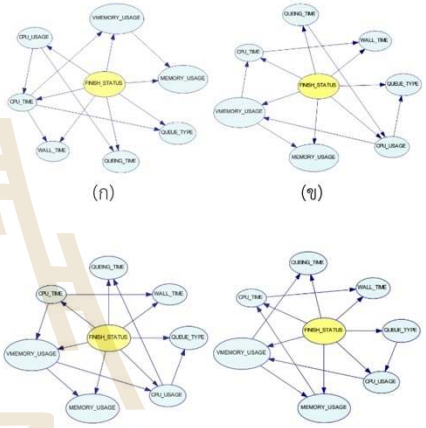
2. ขั้นตอนการสร้างแบบจำลองโครงข่ายเบย์เซียนให้ได้ประสิทธิภาพสูงที่สุด

2.1 สร้างแบบจำลองโครงข่ายเบย์เซียน สำหรับขั้นตอนนี้ผู้วิจัยได้สร้างแบบจำลองฯ โดยใช้ชุดข้อมูลทั้ง 4 ชุดที่ได้จากขั้นตอนวิจัยก่อนหน้า คือ ชุดข้อมูลไตรมาสที่ 1 ของปี 2557, 2558, 2559 และ 2560 โดยใช้เครื่องมือ GeNIe และเลือกใช้อัลกอริทึม TAN สำหรับขั้นตอนของการเรียนรู้ชุดข้อมูลเพื่อสร้างโครงสร้างแบบจำลองโครงข่ายเบย์เซียน หลังจากผ่านกระบวนการดังกล่าว จะได้โครงสร้างของแต่ละแบบจำลองฯ แสดงในรูปแบบของแบบจำลองกราฟ ดังภาพประกอบที่ 3

2.2 ประเมินประสิทธิภาพและคัดเลือกแบบจำลองที่มีประสิทธิภาพดีที่สุด สำหรับขั้นตอนนี้เป็นขั้นตอนการประเมินประสิทธิภาพของแบบจำลองโครงข่ายเบย์เซียนทั้ง 4 ของขั้นตอนวิจัยก่อนหน้า เพื่อคัดเลือกแบบจำลองฯ ที่มีประสิทธิภาพดีที่สุด ไปใช้ในขั้นตอนวิจัยต่อไป

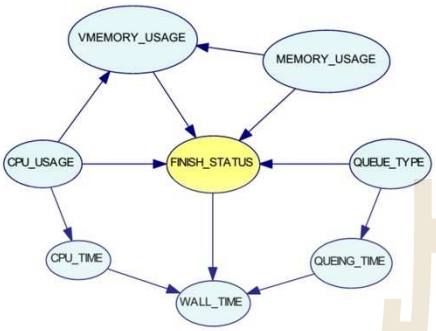
ผู้วิจัยเลือกวิธีประเมินประสิทธิภาพแบบจำลองฯ 3 รูปแบบ คือ 1) ประเมินโดยใช้ข้อมูลเรียนรู้ (Train Set) และข้อมูลทดสอบ (Test Set) ตัวเดียวกัน 2) ประเมินโดยใช้วิธี K-fold Cross Validation โดยกำหนดค่า K เท่ากับ 10 3) ประเมินโดยแบ่งข้อมูลออกเป็น 2 ส่วน ส่วนแรกเป็นข้อมูลเรียนรู้ มีขนาด 70% ของชุดข้อมูล และส่วนที่สองคือข้อมูลที่เหลือจากส่วนแรก (30%) ให้เป็นข้อมูลทดสอบ โดยที่ประสิทธิภาพของแบบจำลองฯ ถูกประเมินจากค่าของ 2 ตัววัดประสิทธิภาพ คือ 1) ค่าความถูกต้องของตัววัดประสิทธิภาพความถูกต้อง และ 2) ค่า AUC ของตัววัดประสิทธิภาพ ROC

ผลจากการประเมินประสิทธิภาพ พบว่า แบบจำลองโครงข่ายเบย์เซียนจากชุดข้อมูลไตรมาสที่ 1 ปี 2560 มีประสิทธิภาพดีที่สุด ในทุกรูปแบบการประเมิน แสดงดังตารางที่ 3 ดังนั้น ผู้วิจัยได้เลือกแบบจำลองฯ ดังกล่าว เพื่อใช้ในขั้นตอนวิจัยถัดไป



ภาพประกอบที่ 3 (ก) แบบจำลองฯ จากชุดข้อมูลไตรมาสที่ 1 ปี 2557, (ข) แบบจำลองฯ จากชุดข้อมูลไตรมาสที่ 1 ปี 2558, (ค) แบบจำลองฯ จากชุดข้อมูลไตรมาสที่ 1 ปี 2559 และ (ง) แบบจำลองฯ จากข้อมูลไตรมาสที่ 1 ปี 2560

2.3 นำแบบจำลองฯ ที่คัดเลือกมาปรับโครงสร้างแบบจำลองฯ ตามคำแนะนำของผู้เชี่ยวชาญฯ เพื่อ 1) ลดความซับซ้อนของแบบจำลองฯ และ 2) คาดหวังที่จะเพิ่มประสิทธิภาพของแบบจำลองฯ ให้มีประสิทธิภาพดียิ่งขึ้น

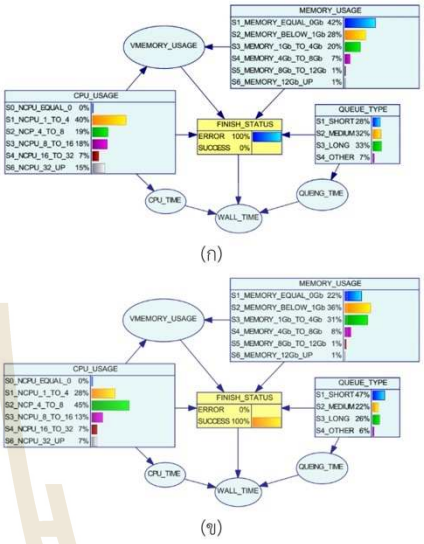


ภาพประกอบที่ 4 แบบจำลองฯ ที่ปรับโครงสร้างผ่านคำแนะนำของผู้เชี่ยวชาญฯ

หลังจากการปรับโครงสร้างแบบจำลองโครงข่ายเบย์เซียนของชุดข้อมูลไตรมาสที่ 1 ปี 2560 ซึ่งทำให้ได้แบบจำลองฯ ที่มีความซับซ้อนลดลง ดังภาพประกอบที่ 4 และผู้วิจัยได้ประเมินประสิทธิภาพแบบจำลองฯ ดังกล่าวเหมือนกับขั้นตอนวิจัยที่ 2.2 พบว่า แบบจำลองฯ ที่ได้ปรับโครงสร้างผ่านคำแนะนำของผู้เชี่ยวชาญฯ มีประสิทธิภาพดีที่สุดในทุกรูปแบบการประเมิน แสดงดังตารางที่ 3 จากขั้นตอนการวิจัยนี้ทำให้ได้แบบจำลองโครงข่ายเบย์เซียนที่ให้ประสิทธิภาพสูงสุด เพื่อนำไปใช้วิเคราะห์และทำนายผลข้อมูลที่สนใจ ในขั้นตอนวิจัยถัดไป

3. ขั้นตอนวิเคราะห์และทำนายผลข้อมูลที่สนใจผ่านแบบจำลองโครงข่ายเบย์เซียน

ขั้นตอนวิจัยนี้ เป็นการวิเคราะห์ผลของชุดข้อมูลไตรมาสที่ 1 ปี 2560 ของข้อมูลบันทึกรายวันของระบบคำนวณสมรรถนะสูงของเนคเทค ผ่านแบบจำลองโครงข่ายเบย์เซียนที่นำเสนอจากขั้นตอนวิจัยก่อนหน้า โดยใช้กระบวนการประมวลผลเชิงตรรกะตามหลักการทำงานของโครงข่ายเบย์เซียน เพื่อทำนายลักษณะงานที่มีโอกาสประมวลผลสำเร็จมากที่สุดบนระบบคำนวณสมรรถนะสูงของเนคเทค เพื่อเพิ่มอัตราความสำเร็จของงานในระบบฯ ซึ่งเป็นปัจจัยหลักต่อประสิทธิภาพของการบริการระบบคำนวณสมรรถนะสูง



ภาพประกอบที่ 5 (ก) การวิเคราะห์และทำนายผลข้อมูลผ่านแบบจำลองฯ ที่นำเสนอ เมื่อกำหนดโอกาสการประมวลผลสำเร็จของงานเป็น 0% และ (ข) การวิเคราะห์และทำนายผลข้อมูลผ่านแบบจำลองฯ ที่นำเสนอ เมื่อกำหนดโอกาสการประมวลผลสำเร็จของงานเป็น 100%

สำหรับวิธีการวิเคราะห์ผลของชุดข้อมูลดังกล่าว ผู้วิจัย กำหนดให้ตัวแปร FINISH_STATUS เป็นตัวแปรหลักฐานเชิงประจักษ์ (Evidence) แล้วเลือกพิจารณาถึงโอกาสที่งานจะประมวลผลสำเร็จ 0%, 43%, 80% และ 100% โดยที่โอกาสที่งานจะประมวลผลสำเร็จ 43% เป็นค่าตั้งต้นที่ได้จากการป้อนชุดข้อมูล และผู้วิจัยกำหนดตัวแปรเป้าหมาย (Target) 3 ตัวแปร คือ CPU_USAGE MEMORY_USAGE และ QUEUE_TYPE เพื่อต้องการศึกษาผลกระทบต่อ 3 ตัวแปรดังกล่าว เมื่อเลือกพิจารณาโอกาสที่งานจะประมวลผลสำเร็จในตัวแปร FINISH_STATUS

ตารางที่ 3 ผลการประเมินประสิทธิภาพแบบจำลอง

No.	Bayesian Network	Accuracy			AUC of ROC (Avg.)		
		Train 100% Test 100%	10-fold Cross Validation	Train 70% Test 30%	Train 100% Test 100%	10-fold Cross Validation	Train 70% Test 30%
1	Q1_2014: TAN	0.7493	0.7419	0.6813	0.8348	0.8165	0.7179
2	Q1_2015: TAN	0.7012	0.6882	0.6377	0.7748	0.7601	0.6794
3	Q1_2016: TAN	0.8148	0.8097	0.7155	0.8853	0.8784	0.8071
4	Q1_2017: TAN	0.8274	0.8207	0.7764	0.9054	0.8941	0.8642
5	*Q1_2017: Expert*	*0.8344*	*0.8265*	*0.7879*	*0.9142*	*0.8971*	*0.8840*

ตารางที่ 4 ผลการวิเคราะห์และทำนายลักษณะงาน

Job Success Rate	ตัวแปรเป้าหมาย		
	CPU_USAGE	MEMORY_USAGE	QUEUE_TYPE
0%	1 - 4 (40%)	0GB (42%)	LONG (33%), MEDIUM (32%)
43%	1 - 4 (35%), 4 - 8 (30%)	0 - 1GB (65%)	SHORT (36%), LONG (30%)
80%	1 - 4 (31%), 4 - 8 (39%)	1 - 4GB (63%)	SHORT (42%)
100%	4 - 8 (45%)	1 - 4GB (67%)	SHORT (47%)

ผู้วิจัยเลือกพิจารณาเฉพาะ 3 ตัวแปร เพื่อแทนลักษณะของงาน ตามคำแนะนำของผู้เชี่ยวชาญฯ เพราะว่าตัวแปร CPU_USAGE, MEMORY_USAGE และ QUEUE_TYPE คือจำนวนหน่วยประมวลผล, ขนาดของหน่วยความจำ และประเภทของแถวคอย ตามลำดับ ซึ่งเป็นค่าตั้งต้นที่ผู้ใช้งานจำเป็นต้องระบุก่อนส่งงานเข้าประมวลผลในระบบฯ ดังนั้นงานวิจัยนี้สามารถใช้แบบจำลองฯ ที่นำเสนอวิเคราะห์และทำนายลักษณะงานที่มีโอกาสประมวลผลสำเร็จมากที่สุดผ่าน 3 ตัวแปรดังกล่าว ซึ่งให้ผลของการทำนายดังตารางที่ 4 ภาพประกอบที่ 5 แสดงภาพตัวอย่างการวิเคราะห์ผลข้อมูลผ่านแบบจำลองโครงข่ายเบย์เซียนที่นำเสนอบนเครื่องมือ GeNIe

ผลการวิจัย

ผลการวิจัยแบ่งออกเป็น 2 ส่วนตามกรอบแนวคิดการวิจัย คือ

1. ผลการประเมินประสิทธิภาพแบบจำลองโครงข่ายเบย์เซียนของชุดข้อมูลไตรมาสที่ 1 ปี 2560 มีประสิทธิภาพดีที่สุด โดยมีค่าความถูกต้องที่ 78% จากการประเมินประสิทธิภาพแบบแบ่งข้อมูลเรียนรู้ 70% และข้อมูลทดสอบ 30% จากข้อมูลทั้งหมด และมีค่าความถูกต้องเพิ่มขึ้นเป็น 79% เมื่อนำแบบจำลองโครงข่ายเบย์เซียนของชุดข้อมูลดังกล่าว มาปรับโครงสร้างแบบจำลองฯ โดยผ่านคำแนะนำจากผู้เชี่ยวชาญฯ

2. ผลการทำนายลักษณะงานที่มีโอกาสประมวลผลสำเร็จมากที่สุดบนระบบคำนวณประสิทธิภาพสูงของเนคเทค ผ่านแบบจำลองโครงข่ายเบย์เซียนที่นำเสนอ โดยพิจารณาโอกาสของความสำเร็จในการประมวลผลของงานที่ 0%, 43%, 80% และ 100% แสดงดังตารางที่ 4 ซึ่งผลการวิจัยพบว่า

2.1 ลักษณะงานที่ใช้หน่วยประมวลผล 1 – 4 CPU ใช้หน่วยความจำ 0GB และใช้แถวคอยแบบ LONG หรือ MEDIUM มีโอกาสประมวลผลสำเร็จ 0% บนระบบคำนวณสมรรถนะสูงของเนคเทค

2.2 ลักษณะงานที่ใช้หน่วยประมวลผล 1 – 4

CPU หรือ 4 – 8 CPU ใช้หน่วยความจำ 0 – 1GB และใช้แถวคอยแบบ SHORT หรือ LONG มีโอกาสประมวลผลสำเร็จ 43% บนระบบคำนวณสมรรถนะสูงของเนคเทค

2.3 ลักษณะงานที่ใช้หน่วยประมวลผล 1 – 4 CPU หรือ 4 – 8 CPU ใช้หน่วยความจำ 1 - 4GB และใช้แถวคอยแบบ SHORT มีโอกาสประมวลผลสำเร็จ 80% บนระบบคำนวณสมรรถนะสูงของเนคเทค

2.4 ลักษณะงานที่ใช้หน่วยประมวลผล 4 – 8 CPU ใช้หน่วยความจำ1 - 4GB และใช้แถวคอยแบบ SHORT มีโอกาสประมวลผลสำเร็จ 100% บนระบบคำนวณสมรรถนะสูงของเนคเทค

สรุปผล

งานวิจัยนี้สามารถสรุปผลการวิจัยได้ดังนี้ คือ

1. สร้างแบบจำลองโครงข่ายเบย์เซียนที่มีประสิทธิภาพสูงสำหรับใช้วิเคราะห์และทำนายผลข้อมูลที่สนใจ จากการวิจัยพบว่าแบบจำลองฯ ของชุดข้อมูลไตรมาสที่ 1 ปี 2560 จากข้อมูลบันทึกรายวันของระบบคำนวณสมรรถนะสูงของเนคเทค เป็นแบบจำลองฯ ที่มีประสิทธิภาพดีที่สุด ซึ่งมีค่าความถูกต้องที่ 78% และเพิ่มเป็น 79% เมื่อนำแบบจำลองฯ ดังกล่าวมาปรับโครงสร้างผ่านคำแนะนำของผู้เชี่ยวชาญฯ จากประเมินประสิทธิภาพแบบแบ่งข้อมูลเรียนรู้ 70% และข้อมูลทดสอบ 30% ของข้อมูลทั้งหมด

2. ใช้แบบจำลองฯ ที่นำเสนอ วิเคราะห์และทำนายผลข้อมูลของชุดข้อมูลไตรมาสที่ 1 ปี 2560 เพื่อหาลักษณะงานที่มีโอกาสประมวลผลสำเร็จมากที่สุดบนระบบคำนวณสมรรถนะสูงของเนคเทค ซึ่งแบบจำลองสามารถทำนายผลได้สอดคล้องกับความเห็นของผู้เชี่ยวชาญฯ คือ งานที่มีโอกาสประมวลผลสำเร็จ 100% บนระบบคำนวณสมรรถนะสูงของเนคเทค คืองานที่ใช้หน่วยประมวลผล 4 – 8 CPU ใช้หน่วยความจำ 1 – 4 GB และใช้แถวคอยแบบ SHORT ส่วนงานที่ไม่มีโอกาสประมวลผลสำเร็จบนระบบคำนวณสมรรถนะสูงของเนคเทค คืองานที่ใช้หน่วยประมวลผล 1 – 4 CPU ใช้หน่วยความจำ 0 GB และใช้แถวคอยแบบ LONG หรือ MEDIUM

อภิปรายผล

1. แบบจำลองโครงข่ายเบย์เซียนที่ปรับโครงสร้างตามคำแนะนำของผู้เชี่ยวชาญฯ ซึ่งเป็นแบบจำลองฯ ที่นำเสนอในงานวิจัยนี้ สามารถรองรับข้อมูลที่มีลักษณะไม่แน่นอน (Uncertain) โดยที่ประสิทธิภาพของแบบจำลองฯ มีความถูกต้องที่ 79% ของการคำนายผลข้อมูล จากการประเมินประสิทธิภาพแบบแบ่งข้อมูลเรียนรู้ 70% และข้อมูลทดสอบ 30% ซึ่งถือเป็นค่าที่สูงและยอมรับได้สำหรับกระบวนการทำเหมืองข้อมูล ดังนั้นแบบจำลองฯ ที่นำเสนอสามารถรองรับ หรือนำใช้ได้กับทุกช่วงข้อมูลที่สนใจได้อย่างมีประสิทธิภาพ

2. สาเหตุที่แบบจำลองโครงข่ายเบย์เซียนของชุดข้อมูลไตรมาสที่ 1 ปี 2560 มีประสิทธิภาพจากการประเมินดีกว่าแบบจำลองฯ ของข้อมูลชุดอื่นๆ เพราะว่าชุดข้อมูลดังกล่าว เป็นข้อมูลที่ล่าสุดกว่าหรือใหม่กว่าข้อมูลในชุดข้อมูลอื่นๆ กล่าวคือเป็นข้อมูลที่มีความสอดคล้องกับทรัพยากรของระบบคำนวณสมรรถนะสูงของเนคเทค ในปัจจุบันที่สุด

3. ผลการทำนายลักษณะงานที่มีโอกาสประมวลผลสำเร็จมากที่สุดบนระบบคำนวณสมรรถนะสูงของเนคเทค ผ่านแบบจำลองโครงข่ายเบย์เซียนที่นำเสนอ พบว่าเป็นงานที่มีขนาดกลาง คือใช้หน่วยประมวลผล 4 – 8 CPU ใช้หน่วยความจำ 1 – 4 GB และใช้แถวคอยแบบ SHORT แต่ผลการทำนายผ่านแบบจำลองฯ ก็ยังไม่สามารถระบุได้ชัดถึงโอกาสประมวลผลสำเร็จของงานที่ใช้จำนวนหน่วยประมวลผลตั้งแต่ 32 CPU ขึ้นไป เพราะมีจำนวนของตัวอย่างที่อยู่ในชุดข้อมูลน้อยเกินไป

ข้อเสนอแนะ

ผลของการทำนายจากแบบจำลองฯ ที่นำเสนอบ่งชี้ว่างานขนาดกลางมีโอกาสประมวลผลสำเร็จมากที่สุดบนระบบคำนวณสมรรถนะสูงของเนคเทคนั้นจะขึ้นอยู่กับชุดข้อมูลตามช่วงข้อมูลที่สนใจ กล่าวคือถ้าเปลี่ยนชุดข้อมูลที่ใช้สำหรับวิเคราะห์และทำนายผลก็อาจจะให้ผลของการทำนายที่แต่งต่างไป โดยความเป็นจริงแล้วผู้ใช้งานก็สามารถประมวลผลงานที่มีขนาดใหญ่และมีโอกาสประมวล

ผลสำเร็จสูงเช่นกัน ถ้าผู้ใช้งานมีความเข้าใจต่อลักษณะงานของตนเอง และกำหนดตัวแปรอื่นๆ ให้เหมาะสม

กิตติกรรมประกาศ

ผู้วิจัยขอขอบคุณข้อมูลสำหรับใช้ในการวิจัย จากภาคีโครงสร้างพื้นฐานระดับชาติด้าน e-Science ของเนคเทคURL :http://www.e-science.in.th

เอกสารอ้างอิง

Zhang, K., Zhang, X., Zhao, Z., & Wang, H. (2012). **Construction and performance test of a PC cluster.** 2012 2Nd International Conference On Applied Robotics For The Power Industry (CARPI), 799. doi:10.1109/CARPI.2012.6356443

Uthayopas, P., Angskun, T., & Maneesilp, J. (1998). **Building a parallel computer from cheap PCs: SMILE cluster experiences.** Proceedings of the Second Annual National Symposium on Computational Science and Engineering, 1998-Mar.

Deconinck, W., Bauer, P., Diamantakis, M., Hamrud, M., Kühnlein, C., Maciel, P., & ... Wedi, N. P. (2017). **Atlas : A library for numerical weather prediction and climate modelling.** Computer Physics Communications, 220188-204. doi:10.1016/j.cpc.2017.07.006

Ahmed, M., Ahmad, I., & Ahmad, M. (2015). **A survey of genome sequence assembly techniques and algorithms using high-performance computing.** Journal Of Supercomputing, 71(1), 293-339. doi:10.1007/s11227-014-1297-4

O'Brien, B., Walker, R., & Washbrook, A. (2014). **Leveraging HPC resources for High Energy Physics.** Journal Of Physics: Conference Series, 513(3), 1. doi:10.1088/1742-6596/513/3/032104

Gao, Y., Liu, Y., Wang, C., Li, X., & Ou, G. (2012). **Design and Evaluation of a High Performance Distributed Expert System (HPDES) for Aerospace Ground Verification System.** Procedia Computer Science, 9 (Proceedings of the International Conference on Computational Science, ICCS 2012), 1380-1389. doi:10.1016/j.procs.2012.04.152

Yuan, Y., Wu, Y., Wang, Q., Yang, G., & Zheng, W. (2012). **Job failures in high performance computing systems: A large-scale empirical study.** Computers And Mathematics With Applications, 63(Advances in context, cognitive, and secure computing), 365-377. doi:10.1016/j.camwa.2011.07.040

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). **Bayesian Network Classifiers.** Machine Learning, (29), 131. doi:10.1023/A:1007465528199

Zawawy, H., Kontogiannis, K., & Mylopoulos, J. (2010). **Log filtering and interpretation for root cause analysis.** 2010 IEEE International Conference on Software Maintenance, Software Maintenance (ICSM), 2010 IEEE International Conference on, 1. doi:10.1109/ICSM.2010.5609556

Anamika, Gayatri, B., Kanhaiya, & Poonam (2015). **Log Aggregator for Better Root-Cause-Analysis.** (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (2) , 2015, 1100-1102

Sîrbu A., & Babaoglu O. (2015). **A Holistic Approach to Log Data Analysis in High-Performance Computing Systems: The Case of IBM Blue Gene/Q. Parallel Processing Workshops. Euro-Par 2015.** Lecture Notes in Computer Science, vol 9523. doi:10.1007/978-3-319-27308-2_51

Sheng, D., Rinku, G., Marc, S., Eric, P., & Franck, C. (2017). **LOGAIDER: A Tool for Mining Potential Correlations of HPC Log Events.** 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Cluster, Cloud and Grid Computing (CCGRID), 2017 17th IEEE/ACM International Symposium on, CCGRID, 442. doi:10.1109/CCGRID.2017.18

Yoon, JunWeon; Hong, TaeYoung; Kong, Ki-Sik; Park, ChanYeol; (2015) **Improving the Job Success Rate through Analysis of User Logs in HPC**. Journal of Digital Contents Society Volume 16, Issue 5, 2015, pp.691-697. doi:10.9728/dcs.2015.16.5.691

Loghmanpour, N. A., Kormos, R. L., Kanwar, M. K., Teuteberg, J. J., Murali, S., & Antaki, J. F. (2016). **A Bayesian Model to Predict Right Ventricular Failure Following Left Ventricular Assist Device Therapy.** JACC: Heart Failure, 4(9), 711.

VARKEY, D. A., PITCHER, T. J., McALLISTER, M. K., & SUMAILA, R. S. (2013). **Bayesian Decision-Network Modeling of Multiple Stakeholders for Reef Ecosystem Restoration in the Coral Triangle.** Conservation Biology, (3), 459. doi:10.1111/cobi.12036

Shin, J., Son, H., Khalil ur, R., & Heo, G. (2015). **Development of a cyber security risk model using Bayesian networks.** Reliability Engineering And System Safety, 134208-217. doi:10.1016/j.ress.2014.10.006

Caelen, O. (2017). **A Bayesian interpretation of the confusion matrix.** Annals Of Mathematics & Artificial Intelligence,81(3/4), 429-450. doi:10.1007/s10472-017-9564-8

Sokolova, M., Japkowicz, N., & Szpakowicz, S. (2006). **Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation.** Lecture Notes In Computer Science, (4304), 1015-1021.

Bayesian Decision Network Approaches for HPC Log Analysis: Case Study of NECTEC HPC System

Anupong Banjongkan[a,*], Nittaya Kerdprasop[b], Kittisak Kerdprasop[c]
[a]School of Computer Engineering, Suranaree University of Technology, Thailand
E-mail address: banjongkan@gmail.com

[b]Data Engineering Research Unit,
[c]Knowledge Engineering Research Unit,
School of Computer Engineering, Suranaree University of Technology, Thailand
E-mail address: nittaya@sut.ac.th, kerdpras@sut.ac.th

Abstract

Manually of log analysis is very difficult and inefficient. Especially, log of high performance computing (HPC) system, that huge and complex associations of parameter. Moreover, the process to make a decision respect on a HPC log dataset to do something for improve the system efficiency is difficult as well. Sometime, it must be use the expert consult. The efficiency of HPC system is depend on the job success rate. Therefore, this research proposes a Bayesian network model to find suitable characteristic of a job that run in NECTEC HPC system. While, we extend a propose model to be a Bayesian decision network model to make a decision for improve the system efficiency. We assumed the way to improve the system efficiency is hold the event of HPC training for their users. The HPC training can help users to well understand and cleanly about policy and structure of NECTEC HPC that affect to the job success rate in a system. The experimental results show the performance of a proposed model at 79% of accuracy. The best characteristic of a job is a medium job, that uses 4 – 8 CPU, 1 – 4 GB of memory and run in SHORT queue. A Bayesian decision network of propose model can make a decision when we assumed the expected of system efficiency is "High", "Medium", and "Low" represent to the job success rate at 70%, 50%, and 30% respectively.

Keywords: Bayesian Network, Decision Network, HPC, Log, NECTEC

1. Introduction

High Performance Computing (HPC) system is a system that compose of many computes are connected via high speed of inter-connection network and working together for increasing the computing power. HPC system has a middle-ware for manage the resources of system that make it like a single system view (Zhang. et al.,

2012: Uthayopas. et al., 1998). HPC system is difference to traditional computer system, most of HPC system run on 24 x 7 to solved complex and big problems in various fields such as weather forecast (Deconinck. et al., 2017), genome sequencing (Ahmed. et al., 2015), high energy physics (O'Brien. et al., 2014), aerospace engineering (Geo. et al., 2014) and so on.

The efficiency of the HPC systems can evaluate at two levels. The first level evaluates by system capacity based on system utilization (Yang. et al., 2013), suppose a HPC system has total of computing resources to service 10,000 CPU Hour per year but the actual service is 9,000 CPU Hour per year. So, that mean this HPC system can service 90% of system capacity per year. On the other word, this HPC system has 90% of system utilization per year. The efficiency of this level is responsible of operator or administrator of the HPC system to keep availability and reliability of the system. Second level evaluated by focus on the job success rate in the system (Yuan. et al, 2012). In the second level of efficiency is depend on users of the system. If users had submitted a proper job to the HPC system, then job had a high chance of success. So, the job success rate is a very important key of the HPC service (Etinski. et al., 2010) because the HPC system has the high-power consumption. If the efficiency in second level is low that mean the service of HPC system are wasteful. For example, HPC system has 100% of system utilization but the job success rate in the system is 30%, that mean this HPC system useless 70% of electricity. There are many ways to improve the job success rate such as optimize the queue system, manage user's priority, hold the event of HPC training to make users to understand about policy and system. This research focusing on the system efficiency in second level and assume the way to improve the job success is hold the event of HPC training.

The National Electronic and Computer Technology Center (NECTEC) under the Ministry of Science and Technology of Thailand is one of government sector at provided HPC service through National e-Science Infrastructure Consortium Project. NECTEC HPC system serves to government employees such as researcher, student and users in computational area. This research was received the HPC log from ATOM cluster computer of NECTEC. The ATOM cluster computer has 580 CPU cores and record the HPC log since 2012. So, the HPC log has 389,765 records and 27 attributes.

As direct to analyze the HPC log dataset is difficult and inefficient since the log are huge and complex associations of parameter. (Lin. et al., 2013). Sometime, it must be use the expert consult. Log is a dataset that provided by system or software. It records about information in production runtime. The log away used meaningless and short messages to reduce the size of log file. Most of log analysis for investigate the root cause of problems that give to an operator or administrator can fix the system

(Zawawy. et al., 2010: Anamika. el al., 2015). There are many research (Sîrbu. & Babaoglu., 2015: Sheng. et al., 2017) analyzed the HPC log for predict the chance of errors in the system. The research of (Yoon. et al., 2015) used statistic model to analysis the HPC log about job error rate in the HPC system.

This research used data mining process through propose the Bayesian Network model (Friedman. et al., 1997) to analyze the NECTEC HPC log dataset to find suitable characteristic of a job. Moreover, we extend a propose model to a Bayesian decision network to make a decision for improve the system efficiency. The objectives and goals of this research is 1) Provide easy and fast mining model to analyze the real HPC log dataset. 2) Extend a model to decision model to make a decision respect on a dataset for improve the system efficiency. 3) Bring a propose model to be a typical model for another HPC systems.

2. Methods

This research used datamining technique with Bayesian Network and Decision Network for analyzed the NECTEC HPC log dataset to find suitable characteristic of a job and show how to make a decision to improve the system efficiency. The performance evaluation of the model, we used accuracy and ROC graph are evaluators.

2.1 Bayesian Network

The Bayesian Network is one of high performance datamining model as well know and widely used in many research areas such as medical science (Loghmanpour. et al., 2016), environment (Varkey. et al., 2013), security (Shin. et al., 2015), and so no. The Bayesian Network based on Bayes' Theory that show in formula (1). The model had represented in direct acyclic graph (DAG) and inference to the association of data via conditional probability table (CPT) of nodes with Markov chains property (Markov Blanket) to reduce the complexity and consumption time.

$$P(h|e) = \frac{P(e|h)P(h)}{P(e)} \quad (1)$$

where $P(h|e)$ is posterior, $P(e|h)$ is likelihood, $P(h)$ is prior and $P(e)$ is probability of evidence.

2.2 Decision Network

The Decision Network sometime call Bayesian Decision Network or Influence Diagrams. The Decision Network is an extension of Bayesian Network based on decision theory. For the decision theory used the value of expected utility (EU) that combine of utility theory and probability theory as shown in formula (2).

$$EU(A) = \sum_i U(O_i|A) \times P(O_i|A) \quad (2)$$

where $EU(A)$ is expected of the action A, $U(O_i|A)$ is utility function over every possible outcome of action A, $P(O_i|A)$ is a probability for each outcome that given by action A.

The Decision Network has two new nodes is utility and decision node based on Bayesian Network as shown in figure 1. The utility node is constraining utility function to calculate a criterion of decision that depend on inference of Bayesian Network. The function of decision node is decision respect on utility value.

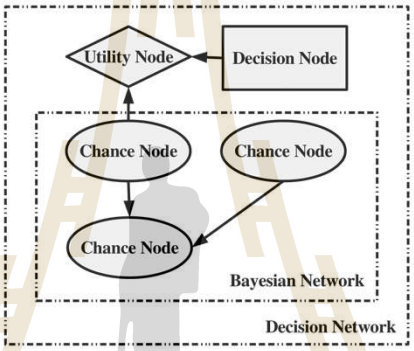


Fig. 1: Example of Bayesian Decision Network

2.3 Performance Evaluation

In this research we used two evaluators for evaluated the performance of propose model. The evaluators are accuracy and Receiver Operating Characteristic (ROC) the both calculated from confusion matrix (Caelen. et al., 2017) as shown in figure 2.

		Actual Value	
		Positive	Negative
Predicted Value	Positive	(A)	(B)
	Negative	(C)	(D)

Fig. 2: Confusion Matrix

As follow in figure 2. (A) is the data was predicted "True" according to the actual value of data is "True" as call "True Positive". (B) is the data was predicted "False" but the actual value of data is "True" as call "False Negative". (C) is the data was

predicted “True” but the actual value of data is “False” as call “False Positive”. (D) is the data was predicted “False” according to the actual of data is “False” as call “True Negative”.

2.3.1 Accuracy used for evaluating the model to predicted class of data. The value of accuracy is between 0 – 1. So, the value is nearly 1 that mean the model is high performance. If the value is nearly 0 that mean the model is low performance. The formula (3) show accuracy calculation from values in the confusion matrix.

$$Accuracy = \frac{(A)+(D)}{(A)+(B)+(C)+(D)} \quad (3)$$

2.3.2 Receiver Operating Characteristic: ROC show the result in graph format, where y axis represents the true positive divide by the total of data that actual value is true as call “True Positive Rate: TPR”. The x axis represents the false positive divide by the total of data that actual value is false as call “False Positive Rate: FPR”. For calculate the TPR and FPR show in formula (4) and (5) respectively. Furthermore, it easy to understand the performance of the model that can uses the area under curve (AUC) of ROC graph. The value of AUC of ROC is between 0 – 1. So, the value nearly 1 that mean the model is high performance. If the value is nearly 0.5 that mean the model is low performance.

$$TPR = \frac{(A)}{(A)+(B)} \quad (4)$$

$$FPR = \frac{(D)}{(c)+(D)} \quad (5)$$

3. Experimental and Results

3.1 Tools

There are many software to modeling and learning with Bayesian Network such as GeNIe, Netica, Bayesian Lab, Hugin and AgenaRisk. In this research used GeNIe software version 2.2.2017.0 (32-bit academic) that developed by BAYESFUSION, LLC company based on C++ language. This Bayesian software is well known and free of charge for academic license.

3.2 Dataset

The Dataset was used in this research is the HPC log from NECTEC HPC system, that start record in 2012. The size of dataset is 389.765 records with 27 attributes. In this research selected 7 attributes and include 1 synthesis attribute based on expert knowledge. The selected attributes as shown in tables 1.

Table 1: Show the collected attributes from dataset

No.	Name	Description	Type
1	QUEUE_TYPE	Type of queue in NECTEC HPC system.	Original
2	CPU_USAGE	The number of CPU that job required.	Original
3	FINISH_STATUS	The end state of job.	Original
4	CPU_TIME	The computation time × number of CPU that job require.	Original
5	MEMORY_USAGE	The main memory usage of job.	Original
6	VMEMORY_USAGE	The main memory usage of job includes with environment of job.	Original
7	WALL_TIME	The time of waiting in queue + computation time.	Original
8	QUEING_TIME	The time of waiting in queue.	Synthesis

3.3 Experiment

First, we pre-processed raw dataset for suitable to the GeNIe software. After that we selected a target dataset and used it to build the Bayesian network model. Next, adjust the model by expert to reduce the complexity of model. We evaluated the performance of both models. Finally, we can find the suitable characteristic of a job, that run in NECTEC HPC system by inference of the propose model. Finally, we scale a model to Bayesian Decision Network. The research workflow as shown in figure 3.

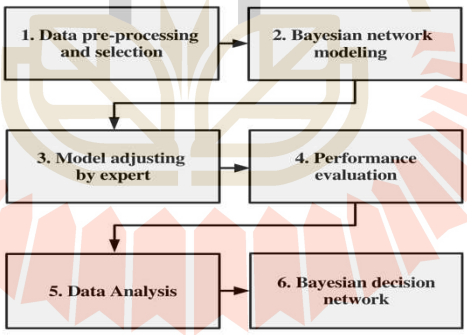


Fig. 3: The research workflow

3.3.1 Data pre-processing and selection. This process for prepared raw dataset such as eliminate missing values and synthesis some attributes if necessary. We selected the first quarter (Q1) of 2017 (January, February, March) from raw dataset

and synthesized a “QUEING_TIME” attribute to be a target dataset. The size of target dataset is 4,039 records. The reason for selected Q1 of 2017 be a dataset because this period is presently and highly work load in the system.

3.3.2 Bayesian network modeling. From previous process we got a dataset for build the Bayesian network model. We interested 8 attributes QUEUE_TYPE, FINISH_STATUS, CPU_USAGE, MEMORY_USAGE, VMEMORY_USAGE, CPU_TIME, QUEING_TIME, and WALL_TIME. There are 6 attributes are continuous values that cannot build the Bayesian network model. It must has discretized the values before use. The table 2 show discretize values of 6 attributes. After that, we used the dataset to build the Bayesian network model with Tree Augmented Naïve Bayes (TAN). So, this is the approximate algorithm that appropriate to the datasets are size more than 1,000 records. In this research, we focus on the “FINISH_STATUS” attribute this represent to the job success rate.

Table 2: Show discretize values of 6 attributes

Attribute	State (#CPU)	Attribute	State (GB)	Attribute	State (Hour)
CPU_USAGE	1) 0	MEMORY_USAGE, VMEMORY_USAGE	1) 0	QUEING_TIME, CPU_TIME, WALL_TIME	1) 0
	2) 1 – 4		2) 1		2) 1 – 3
	3) 4 – 8		3) 1 – 4		3) 3 – 6
	4) 8 – 16		4) 4 – 8		4) 6 – 12
	5) 16 – 32		5) 8 – 12		5) 12 - 24
	6) > 32		6) > 12		6) > 24

3.3.3 Model adjusting by expert. We used expert knowledge to adjust the model from the previous process. We would like to reduce the complexity of model. While, the performance of the model is increases. The figure 4 (a) is the Bayesian network model before expert adjust, (b) is the Bayesian network model after expert adjust.

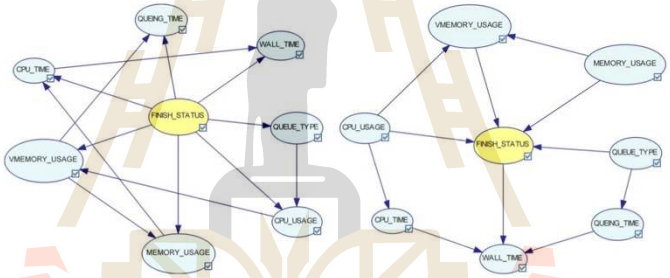
Table 3: Performance evaluation of both models

Bayesian Network	Accuracy			AUC of ROC graph (Avg.)		
	Train 100% Test 100%	10-fold Cross Validation	Train 70% Test 30%	Train 100% Test 100%	10-fold Cross Validation	Train 70% Test 30%
TAN	0.8274	0.8207	0.7764	0.9054	0.8941	0.8642
Expert	**0.8344**	**0.8365**	**0.7879**	**0.9142**	**0.8917**	**0.8840**

3.3.4 Performance evaluation. We evaluated the performance of both models using two evaluators is accuracy and ROC. For the ROC used the value of AUC of the graph. We set three manners is 1) train/test 100% of dataset, 2) 10-fold cross

validation, and 3) train 70% and test 30% of dataset to evaluate the propose models. The results as shown in table 3. The propose model by expert adjusted outperform the propose model by TAN algorithm.

3.3.5 Data Analysis. This process used a Bayesian network model was adjusted by expert to find the characteristics of a job that suitable for run in the NECTEC HPC system. We used the dataset is Q1 of 2017 to be an input of the model. So, we used manual sensitivity analysis by set the "FINISH_STATUS" is evident and set three attributes "CPU_USAGE", "MEMORY_USAGE", "QUEUE_TYPE" are target. Then, we interested the "FINISH_STATUS" at success state are 0%, 43%, 80% and 100%. The results show jobs likely to success rate 1) 0% that uses 1 – 4 CPU, 0 GB of memory and run in LONG or MEDIUM queue. 2) 43% that uses 1 – 4 CPU or 4 – 8 CPU, 0 – 1 GB of memory and run in SHORT or LONG queue. 3) 80% that uses 4 – 8 CPU or 1 – 4 CPU, 1 – 4 GB of memory and run in SHORT queue. 4) 100% that uses 4 – 8 CPU, 1 – 4 GB of memory and run in SHORT queue. The results as shown in table 4 and example manually sensitivity analysis on the propose model that show in figure 5.



(a) (b)

Fig. 4: (a) Bayesian network by TAN algorithm, (b) Bayesian network by expert

Table 4: The results of characteristic of job respect on percentage of success rate

FINISH_STATUS (Success state)	Target Attributes		
	CPU_USAGE	MEMORY_USAGE	QUEUE_TYPE
0%	1 – 4 CPU (40%)	0 GB (42%)	LONG (33%) MEDIUM (32%)
43%	1 – 4 CPU (35%) 4 – 8 CPU (30%)	0 – 1 GB (65%)	SHORT (36%) LONG (30%)
80%	4 – 8 CPU (39%) 1 – 4 CPU (31%)	1 – 4 GB (63%)	SHORT (42%)
100%	4 – 8 CPU (45%)	1 – 4 GB (67%)	SHORT (47%)

3.3.6 Bayesian decision network. The final process in this research, we extended the exiting Bayesian network model to Bayesian decision network model. This process shows Bayesian decision network make a decision for improve the system efficiency depend on utility function in utility node. In this research, the utility node represents to the expected of system efficiency. We set the expected of system efficiency to “High”, “Medium”, and “Low”, that are representing to the job success rate at 70%, 50%, and 30% respectively. We assumed the way to improve the system efficiency, that hold the event of HPC training on the decision node. The dataset is Q1 in 2017 of NECTEC HPC log. The Bayesian decision network make a decision respect on “High”, “Medium”, and “Low” of expected of system efficiency. Figure 6 show the example of Bayesian decision network model when decision node defined the expected of system efficiency to “Low”.

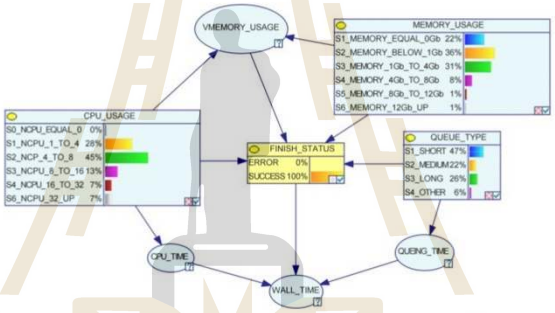


Fig. 5: The example of manually sensitivity on the model at 100% of job success rate

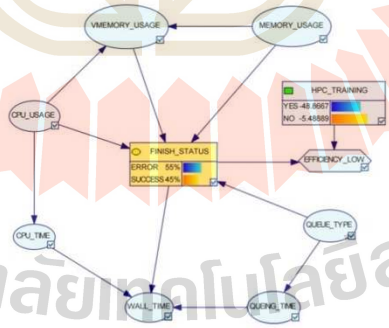


Fig. 6: Decision network model when expected of system efficicecy is “Low”

3.4 Results

3.4.1 Bayesian network model had been adjusted by expert has highly performance at 79% of accuracy when used train 70% and test 30% of dataset. The best of characteristic of a job based on NECTEC HPC system that uses 4 – 8 CPU, 1 – 4 GB of memory and run in SHORT queue. While, the characteristic of a job must be prevented uses 1 – 4 CPU, of memory 0 GB and run in LONG or MEDIUM queue.

3.4.2 Bayesian decision network make a decision "Yes" on decision node when set the expected of system efficiency on utility node is "High", make a decision "Yes" when set expected of system efficiency is "Medium", and make a decision "No" when set expected of system efficiency is "Low". The results as shown in table 5.

Table 5: The results of Bayesian decision network

Utility Node	Decision Node		
Expected of efficiency	Decision Values		Make a Decision
	Yes	No	
High	64.666662	-16.466666	Yes
Medium	5.6444399	-10.977777	Yes
Low	-48.866671	-5.4888887	No

4. Conclusions

This research shows about mining in large and complex dataset with high performance model. We propose a Bayesian network model to analyze the NECTEC HPC log dataset. We find suitable characteristic of a job that likely high success when run in the NECTEC HPC system, that is a medium job as uses 4 – 8 CPU, 1 – 4 GB of memory and run in the SHORT queue. Moreover, we show how to extended a propose model to be a Bayesian decision network to make a decision for improve the system efficiency. When, we assumed to hold the event of HPC training that can improve the system efficiency. The propose model can make a decision respect on a dataset when we set expected of system efficiency to "High", "Medium", and "Low".

This research can be typically of another HPC systems to uses Bayesian network and Bayesian decision network for model, analysis, predict, and decision based on HPC log dataset.

5. Acknowledgements

The authors acknowledge "National e-Science Infrastructure Consortium" of NECTEC for providing the HPC log dataset that have contributed to the research results reported within this paper. URL: http://www.e-science.in.th

6. References

Zhang, K., Zhang, X., Zhao, Z., & Wang, H. (2012). *Construction and performance test of a PC cluster.* 2012 2Nd International Conference on Applied Robotics for The Power Industry (CARPI), 799. doi:10.1109/CARPI.2012.6356443

Uthayopas, P., Angskun, T., & Maneesilp, J. (1998). *Building a parallel computer from cheap PCs: SMILE cluster experiences.* Proceedings of the Second Annual National Symposium on Computational Science and Engineering, 1998-Mar.

Deconinck, W., Bauer, P., Diamantakis, M., Hamrud, M., Kühnlein, C., Maciel, P., & ... Wedi, N. P. (2017). *Atlas: A library for numerical weather prediction and climate modelling.* Computer Physics Communications, 220188-204. doi:10.1016/j.cpc.2017.07.006

Ahmed, M., Ahmad, I., & Ahmad, M. (2015). *A survey of genome sequence assembly techniques and algorithms using high-performance computing.* Journal Of Supercomputing, 71(1), 293-339. doi:10.1007/s11227-014-1297-4

O'Brien, B., Walker, R., & Washbrook, A. (2014). *Leveraging HPC resources for High Energy Physics.* Journal Of Physics: Conference Series, 513(3), 1. doi:10.1088/1742-6596/513/3/032104

Gao, Y., Liu, Y., Wang, C., Li, X., & Ou, G. (2012). *Design and Evaluation of a High Performance Distributed Expert System (HPDES) for Aerospace Ground Verification System.* Procedia Computer Science, 9 (Proceedings of the International Conference on Computational Science, ICCS 2012), 1380-1389. doi:10.1016/j.procs.2012.04.152

Yang, X., Zhou, Z., Wallace, S., Lan, Z., Tang, W., Coghlan, S., & Papka, M. E. (2013). *Integrating dynamic pricing of electricity into energy aware scheduling for HPC systems.* Proceedings Of The International Conference On High Performance Computing, Networking, Storage & Analysis, 1. doi:10.1145/2503210.2503264

Yuan, Y., Wu, Y., Wang, Q., Yang, G., & Zheng, W. (2012). *Job failures in high performance computing systems: A large-scale empirical study.* Computers And Mathematics With Applications, 63(Advances in context, cognitive, and secure computing), 365-377. doi:10.1016/j.camwa.2011.07.040

Etinski. M., Corbalan. J., Labarta. J., Valero. M. (2010). *Optimizing Job Performance Under a Given Power Constraint in HPC Centers.* Green Computing Conference, pp. 257-267.

Lin, X., Wang, P., & Wu, B. (2013). *Log analysis in cloud computing environment with Hadoop and Spark.* 2013 5th IEEE International Conference on Broadband Network & Multimedia Technology, Broadband Network & Multimedia Technology (IC-BNMT), 2013 5th IEEE International Conference on, 273.

doi:10.1109/ICBNMT.2013.6823956

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). *Bayesian Network Classifiers.* Machine Learning, (29), 131. doi:10.1023/A:1007465528199

Zawawy, H., Kontogiannis, K., & Mylopoulos, J. (2010). *Log filtering and interpretation for root cause analysis.* 2010 IEEE International Conference on Software Maintenance, Software Maintenance (ICSM), 2010 IEEE International Conference on, 1. doi:10.1109/ICSM.2010.5609556

Anamika, Gayatri, Kanhaiya, & Poonam. (2015). *Log Aggregator for Better Root-Cause-Analysis.* (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (2), 2015, 1100-1102

Sîrbu A., & Babaoglu O. (2015). *A Holistic Approach to Log Data Analysis in High-Performance Computing Systems: The Case of IBM Blue Gene/Q. Parallel Processing Workshops. Euro-Par 2015.* Lecture Notes in Computer Science, vol 9523. doi:10.1007/978-3-319-27308-2_51

Sîrbu A., & Babaoglu O. (2015). *A Holistic Approach to Log Data Analysis in High-Performance Computing Systems: The Case of IBM Blue Gene/Q. Parallel Processing Workshops. Euro-Par 2015.* Lecture Notes in Computer Science, vol 9523. doi:10.1007/978-3-319-27308-2_51

Yoon, JunWeon; Hong, TaeYoung; Kong, Ki-Sik; Park, ChanYeol; (2015). *Improving the Job Success Rate through Analysis of User Logs in HPC.* Journal of Digital Contents Society Volume 16, Issue 5, 2015, pp.691-697. doi:10.9728/dcs.2015.16.5.691

Loghmanpour, N. A., Kormos, R. L., Kanwar, M. K., Teuteberg, J. J., Murali, S., & Antaki, J. F. (2016). *A Bayesian Model to Predict Right Ventricular Failure Following Left Ventricular Assist Device Therapy.* JACC: Heart Failure, 4(9), 711.

VARKEY, D. A., PITCHER, T. J., McALLISTER, M. K., & SUMAILA, R. S. (2013). *Bayesian Decision-Network Modeling of Multiple Stakeholders for Reef Ecosystem Restoration in the Coral Triangle.* Conservation Biology, (3), 459. doi:10.1111/cobi.12036

Shin, J., Son, H., Khalil ur, R., & Heo, G. (2015). *Development of a cyber security risk model using Bayesian networks.* Reliability Engineering And System Safety, 134208-217. doi:10.1016/j.ress.2014.10.006

Caelen, O. (2017). *A Bayesian interpretation of the confusion matrix.* Annals Of Mathematics & Artificial Intelligence,81(3/4), 429-450. doi:10.1007/s10472-017-9564-8

Performance Evaluation of Ensemble Decision Tree Techniques with HPC-workload Dataset

Anupong Banjongkan[1], Nittaya Kerdprasop[2] and Kittisak Kerdprasop[3]
[1,2,3]*School of Computer Engineering, Suranaree University of Technology, Thailand*
[2]*Data Engineering Research Unit, Suranaree University of Technology.*
[3]*Knowledge Engineering Research Unit, Suranaree University of Technology.*
banjongkan@gmail.com

***Abstract*—Analyze log file manually is an inefficient manner. This is one challenge operation in High Performance Computing (HPC) domain because HPC log file, or HPC-workload log, is huge with a lot of parameters. However, the HPC-workload log contain important information that can be useful in improving the HPC performance. This research proposes classification of data mining technique and descriptive analysis to analyze and predict the job success in HPC system based on realistic HPC-workload dataset from the National Electronic and Computer Center (NECTEC) of Thailand. We adopted ensemble technique using the decision tree classification methods: C5.0, Classification and Regression Tree (CART), and Chi-square Automatic Interaction Detector (CHAID). The dataset was separated into train set (70%) and test set (30%) for building and validating the model, respectively. The hold out train-test method is the best evaluation scheme to prevent the overfitting. The hold-out evaluation results showed that the best performance was the ensemble model of C5.0 at 86.05% of accuracy and the area under curve of Receiver Operating Characteristic (ROC graph) as high as 0.934. However, the best performance improvement is CHAID ensemble with the increased 5.25% of accuracy compared to the non-ensemble model. Meanwhile, the results of descriptive analysis through Pearson correlation coefficient and Web-graph of counting statistic showed some interesting relation among variables. Mostly job success in NECTEC HPC system had occurred in small (older) compute nodes that used short queue type. The most important variables affecting job success in the propose analysis models respective to the NECTEC HPC-workload dataset are "CPU_USAGE", "VMEMORY_USAGE", "EXECUTE_TIME", and "CPU_CONSUMPTION_TIME".**

***Index Terms*—Decision Tree Classifier; High Performance Computing; Ensemble Model; Performance Evaluation.**

I. Introduction

In the last decade, the computer technology is fast growing in all dimensions, for example, the CPU technology, storage technology and network technology. All of these have invoked current applications be able to transfer, process, and storage huge data causing the so called Big Data. So, research in the field of Data science has become more and more important, especially, in the part of Knowledge Discovery and Data mining or the KDD domain. KDD can help data scientists to handle the challenge of Big Data [1], that is to extract some useful information and use them to improve daily operations in each domain.

High Performance Computing (HPC) is one of the domain that has the challenge of Big Data collective from the HPC log or HPC-workload log from HPC scheduler. The big log file contains many objects as well as complex variable relations. Analyzing HPC-workload log is important [2] because it can help discovering some useful information potentially applicable to improve the system efficiency. Traditional analysis of the HPC-workload log is manually done by system operator or administrator. It is inefficient in that the success of the analysis process depends on experience of person who is an administrator of the HPC system. Furthermore, it takes more time to analyzed HPC log file manually event by the experienced administrator.

This research proposes a data mining technique through classification methods to analyze the HPC-workload log. We selected the common three decision tree (DT) methods that are well known and widely used to be a base classifier of our ensemble model. The most advantage of DT is the comprehensible process to interpret the result [3]. This is because DT uses rule-base or logic-base approach to analyze the data like a human thinking. In our analysis, we used real dataset from the National Electronic and Computer Technology Center (NECTEC) of Thailand. This dataset is the HPC-workload log of medium HPC system. NECTEC's HPC system has provided the computing resources to Thai researchers since the mid-2012 until present.

The main objectives of this research are 1) to analyze the real HPC-workload dataset through classification mining method, 2) to perform comparative performance evaluation of the three DT classifiers: C5.0 [4-5], Classification and Regression Tree (CART) [6], and Chi-square Automatic Interaction Detector (CHAID) [7], and 3) to improve the decision tree classifiers through ensemble model by boosting technique [8-10].

Next section is literature review. In section III, we explain DT classification method and evaluator. Our dataset is shown in section IV. Section V is experimental setting, results, and discussion. The last section is conclusion of this research.

II. Literature Review

A. HPC Analysis

The performance of HPC system needs constantly monitoring from the system administrator. Mostly, it can be observed from log file, for example, from compute node log

file, storage log file, network equipment log file, or any combination of log files. The log file records the events that have occurred in the system in time series. So, some researcher [11-13] proposed methods to analyze log file to find a root cause problem of HPC performance. In [14], researcher reviewed the classification models that had been applied for analyzing the network traffic in HPC system. In our research, we focus on analyzing the HPC performance based on HPC-workload log. Such analysis can find the suitable characteristic of a job, learn uses' behavior, and find the appropriate parameter configuration of queue in the HPC system [15-18].

B. Classification as Data Analysis Technique

Many researchers had applied classification models to analyze and compare performance in many domains of datasets. For example, in [19] the authors compared performance between two groups of classification methods, nonlinear and rule-base method, using facebook and tweeter datasets. In [20], the authors studied and evaluated performance of C4.5 method with imbalance dataset and also applied ensemble model to improve the model performance. The work in [21] modified the KNN method and compared performance with standard KNN using K-fold cross-validation. In [22-24] authors performed performance evaluation and compared classification models between linear, non-linear and rule-base within the education data mining domain. From literature review, we have find that there is on research using classification model to analyze performance of HPC system with realistic HPC-workload log to find the characteristics of jobs that are likely to run successfully in the HPC system.

III. CLASSIFICATION & ASSESSMENT

Classification is one of the mostly used data mining technique to learn general patterns from data such that the patterns can correctly classify future data with unknown class label. It is a supervised learning algorithm. Common classification algorithms can be separated into two phases; the first phase is building the model structure via learning process, and the last phase is using the model from the first phase to predict the unseen dataset.

Classification techniques can be divided into three groups: 1) linear classification such as logistic regression and linear discriminate analysis, 2) nonlinear classification such as support vector machine, artificial neural network and k-nearest neighbor, and 3) rule-base or logic-base classification such as decision tree. In this research, we are interested in rule-base classification. We studied three DT methods, (C5.0, CART, CHAID) to build predictive models and compare their performances on classifying realistic HPC-workload dataset. After that, we create ensemble models by boosting technique to improve the model performance. The concept of ensemble model is shown in figure 1.

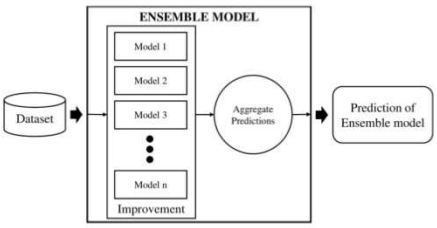


Figure 1: Concept of Ensemble Classification Model

A. Decision Tree Modeling

Decision tree modeling method is a manner to build classifier from tree structure and making class decision with rulesets. For this research, we used three popular DT methods including C5.0, CART, and CHAID. C5.0 or See5 method is an improvement of C4.5 method. C5.0 uses information gain value in splitting process and supports splitting more than two subgroups. CART method builds tree structure and ruleset with the Gini coefficient value. CART supports only two splitting subgroups. CHAID uses the Chi-square statistic value to split data into subgroups and it supports more than two subgroups.

B. Tree Model Assessment

This research uses two evaluators to evaluate the performance of the proposed models, that is, accuracy and Receiver Operating Characteristic, or ROC graph. We choose accuracy and ROC graph to be model evaluators because we would like to observe the overall performance of the model as well as the sensitivity and specificity of the models. So, both evaluators are appropriate and can be calculated from confusion matrix as shown in figure 2.

	Positive Predicted	**Negative Predicted**
Positive Class	True Positive (TP)	False Negative (FN)
Negative Class	False Positive (FP)	True Negative (TN)

Figure 2: Confusion Matrix of a Two-class Problem

Accuracy is mostly use for observing the overall performance of the model. Accuracy is a ratio of the number of objects that model can predict correctly to the number of total objects as shown in formula (1). The value of accuracy stays in the range 0 to 1. The value nearly 1 means that the model has high accuracy performance, but if the value converging to 0 means the model has poor predictive performance.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (1)$$

where: TP = Number of objects that the model predicts as "True" and real class is "True".
TN = Number of objects that the model predicts as "False" and real class is "False".
FP = Number of objects that the model predicts as "True" but real class is "False".
FN = Number of objects that the model predicts as "False" but real class is "True".

ISSN: 2180 – 1843 e-ISSN: 2289-8131 Vol. X No. X

$$True\ Positive\ Rate\ (TPR) = \frac{TP}{TP+FN} \quad (2)$$

$$False\ Positive\ Rate\ (FPR) = \frac{FP}{FP+TN} \quad (3)$$

We also use value of area under curve of ROC graph (or AUC) to assess the performance of model at the specific class of response variable. The ROC graph composes of True Positive Rate (TPR) against False Positive Rate (FPR), represented in y-axis and x-axis, respectively. TPR and FPR are also calculated from confusion matrix, as shown in formula (2) and (3), respectively. The value of AUC of ROC graph is in the range 0 to 1 as well, where 0.5 is baseline of the ROC graph and the value nearly 1 reflects high performance of the model. The AUC lower than 0.5 means unacceptable performance of the model. The figure 3 shows the structure of ROC graph.

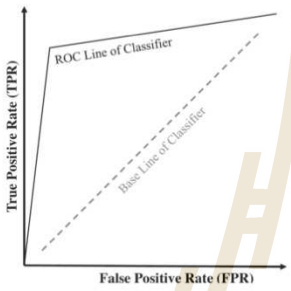


Figure 3: Structure of Receiver Operating Characteristic Graph

IV. HPC Dataset

This research uses realistic dataset from the PBS-scheduler log file of HPC system called "HPC-Workload log". This HPC system is a medium cluster computer of NECTEC. It composes of 580 CPU cores, 2.7 TByte of memory and 50 TByte of disk storages. The NECTEC HPC has been providing and serving computing power to researchers in Thailand since the mid-2012 until present. The data recorded in NECTEC HPC-workload log contain 421,659 objects with 27 variables.

```
1 01/01/2017 00:37:22;E;92995                .nectec.or.th;user=c1280hd
  group=p128 jobname=Zr-S-Ads queue=long ctime=1483099421 qtime=148
  3099421 etime=1483099421 start=1483099422 owner=c1280
  .nectec.or.th exec_host=sodium-0-0.ib/11+sodium-0-0.ib/10+sodium
  -0-0.ib/9+sodium-0-0.ib/8 Resource_List.ncpus=1 Resource_List.need
  nodes=1:ppn=4 Resource_List.nodect=1 Resource_List.nodes=1:ppn=4 R
  esource_List.walltime=336:00:00 session=12466 end=1483205842 Exit_
  status=0 resources_used.cput=116:10:54 resources_used.mem=1747740k
  b resources_used.vmem=5130256kb resources_used.walltime=29:33:40
2 01/01/2017 11:13:47;Q;93015                .nectec.or.th;queue=long
3 01/01/2017 11:13:48;S;93015                .nectec.or.th;user=c1290hg
  group=p129 jobname=Ge-Defect-TS2 queue=long ctime=1483244027 qtim
  e=1483244027 etime=1483244027 start=1483244028 owner=c1290hg
       .nectec.or.th exec_host=sodium-0-0.ib/11+sodium-0-0.ib/10+s
  odium-0-0.ib/9+sodium-0-0.ib/8 Resource_List.ncpus=1 Resource_List
  .neednodes=sodium-0-0.ib:ppn=4 Resource_List.nodect=1 Resource_Lis
  t.nodes=1:ppn=4 Resource_List.walltime=336:00:00
```

Figure 4: Example of Record in Realistic HPC Workload Dataset

From raw dataset (figure 4), we pre-process data by selecting only recent records from the year 2017 (size is 17,081 objects) and using only 10 variables as advised by the HPC expert (shown in table 1). In this research, we focus on finish status of jobs that run in the NECTEC HPC system. So, the response variable of our model is "FINISH_STATUS"; other variables are used as predictors.

Table 1
Variables in HPC-workload Dataset

Variable Name	Description	Scale Type
CPU_USAGE	Number of CPU at job required	Numeric
MEMORY_USAGE	Memory usage when job running	Numeric
VMEMORY_USAGE	Memory usage when job running + job's environment	Numeric
CPU_CONSUMPTION	Computation time × number of CPU at job required	Numeric
QUEUEING_TIME	Time when job waiting in queue	Numeric
EXECUTE_TIME	Computation time	Numeric
WALL_TIME	Time when job waiting in queue + computation time	Numeric
EXECUTE_HOST	Compute node at job running	Categorical
QUEUE_TYPE	Queue type in HPC system	Categorical
FINISH_STATUS	Job's finish status (Success or Error)	Binary

V. Experimentation, Results & Discussion

A. *Experimentation*

We setup experimentation of this research composing of four phases as shown in figure 5. The first phase or initial state is planning and literature review. Second phase is data collection and data pre-processing to make appropriate for our experiment. Third phase is experimental state. This research uses descriptive analysis to explain a relation of each pairwise of variables, and predictive analysis to predict class label of objects. The descriptive analysis has been performed based on the Pearson correlation coefficient and Web-graph of counting statistic to show and explain a relation of each pairwise of variables. The Pearson correlation coefficient is to be used on quantitative variables, whereas, Web-graph is for qualitative variables. In part of predictive analysis, we propose three DTs to classify the HPC-workload dataset. The final phase is a process of performance comparison of each DT model.

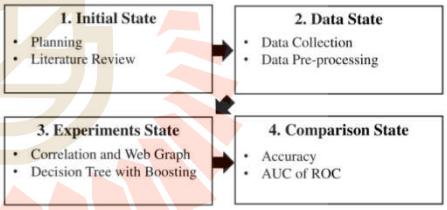


Figure 5: Research Workflow

The success or unsuccess jobs on the NECTEC HPC system are the main focus of our analysis with data classification techniques. So, the response variable of model in the learning process is "FINISH_STATUS" variable, while, others are predictor variables. We split the HPC-workload dataset into two data-subsets. The 70% of original dataset is to be a training data, whereas, the remaining 30% data-subset is to be used as a test

set. We are using a train set to build a classifier model and using a test set to validate the model for the overfitting avoidance.

After that, we validate the results of performance comparison among three DT classifiers by repeating the experiment with four subsets. We divide the original dataset (HPC-workload dataset of 2017) to quarter 1, quarter 2, quarter 3, and quarter 4 of 2017.

	CPU_USAGE	MEMORY_USAGE	VMEMORY_USAGE	CPU_CONSUMPTION	QUEUEING_TIME	EXECUTE_TIME	WALL_TIME
CPU_USAGE	1	0.206	0.353	0.400	0.118	0.042	0.111
MEMORY_USAGE	0.206	1	0.846	0.152	-0.042	0.068	0.025
VMEMORY_USAGE	0.353	0.846	1	0.289	-0.077	0.128	0.048
CPU_CONSUMPTION	0.400	0.152	0.289	1	-0.021	0.606	0.454
QUEUEING_TIME	0.118	-0.042	-0.077	-0.021	1	-0.034	0.635
EXECUTE_TIME	0.042	0.068	0.128	0.606	-0.034	1	0.750
WALL_TIME	0.111	0.025	0.048	0.454	0.635	0.750	1

Figure 6: Pearson Correlation Coefficient Matrix

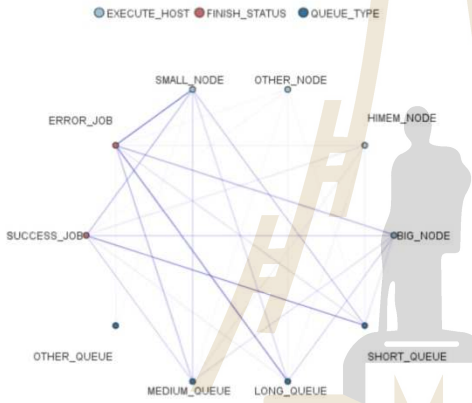


Figure 7: Web-graph Showing Relations among Variables

B. Results

Figure 6 shows the relations of variables in the group of quantitative variables through Pearson correlation coefficient. The results show that "CPU_USAGE" variable has relation with the "CPU_CONSUMPTION" and "VMEMORY_USAGE" variables; "MEMORY_USAGE" variable has strong relation with "VMEMORY_USAGE" and "QUEUEING_TIME" variables; "EXECUTE_TIME" has a strong relation with "WALL_TIME" variable. These results of Pearson correlation coefficient analysis are in accordance to the common sense characteristics of most HPC systems. The Web-graph results as shown in figure 7 show relation of variables in group of qualitative variables by counting statistic between 20 to 5,000 times. It represents the facts that mostly job success occurred on small compute nodes in the older computer machine in the NECTEC HPC system (SMALL_NODE state of "EXECUTE_HOST" variable) with short queue type (SHORT_QUEUE state of "QUEUE_TYPE" variable). Job unsuccess (error) occurred mostly on big compute nodes installed in the new computer machine in the NECTEC HPC system (BIG_NODE state of "EXECUTE_HOST" variable) with long (LONG_QUEUE state of "QUEUE_TYPE" variable) or medium (MEDIUM_QUEUE state of "QUEUE_TYPE" variable) queue type.

Table 2
The Results of Performance Evaluation of C5.0, CART, and CHAID

Classifiers	Accuracy					
	Training Set			Test Set		
	Standard	Boosting	Improve	Standard	Boosting	Improve
C5.0	0.873	0.907	3.93%	**0.840**	**0.860**	**2.42%**
CART	0.775	0.819	5.64%	**0.775**	**0.815**	**5.11%**
CHAID	0.771	0.812	5.55%	**0.758**	**0.798**	**5.25%**
	AUC of ROC					
	Training Set			Test Set		
	Standard	Boosting	Improve	Standard	Boosting	Improve
C5.0	0.944	0.963	2.01%	**0.909**	**0.934**	**2.75%**
CART	0.817	0.872	6.73%	**0.821**	**0.868**	**5.72%**
CHAID	0.859	0.882	2.68%	**0.850**	**0.872**	**2.59%**

The performance comparison of the three-based models is shown in table 2. We consider particularly at the result of test set. The results of standard model, which is the single tree, show that C5.0 outperforms others at 84% of accuracy and AUC of ROC as high as 0.909. At the same time, accuracy of CART and CHAID of standard model is 77.5% and 75.8% respectively. We improve the model performance through ensemble model with boosting technique. The results show that C5.0 is again better than other models at 86% of accuracy, whereas, accuracy of CART and CHAID is up to 81.5% and 79.8%, respectively. For, AUC of ROC graph of C5.0, CART and CHAID are 0.934, 0.868 and 0.872, respectively. So, the best of performance improvement is CHAID with the 5.25% of accuracy increase compared to the standard model. The second best improvement is CART with the accuracy increase at 5.11%. The last improvement is C5.0 with only 2.42% of accuracy increase. While, the performance improvement in terms of AUC of ROC graph of C5.0, CART and CHAID can be improved up to 2.57%, 5.72% and 2.59% as compared to the standard model, respectively. The performance improvement is graphically shown in figure 8.

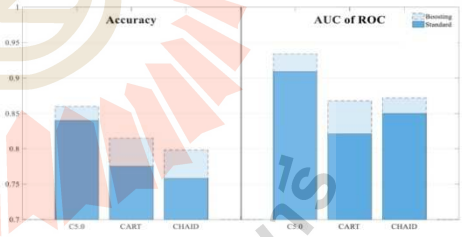
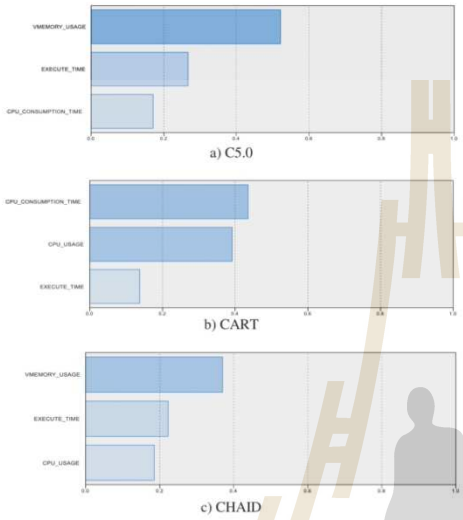


Figure 8: Performance Comparison of the Standard Single Tree Models and the Boosting Models

Figure 9 shows predictor variable importance on predicting the response variable (job success) of C5.0, CART, and

ISSN: 2180 – 1843 e-ISSN: 2289-8131 Vol. X No. X

CHAID. The results show that "CPU_USAGE", "MEMORY_USAGE", "EXECUTE_TIME" and "CPU_CONSUMPTION_TIME" are the most importance variables that may affect a job success in the NECTEC HPC system with respect to the historical NECTEC HPC-workload dataset.



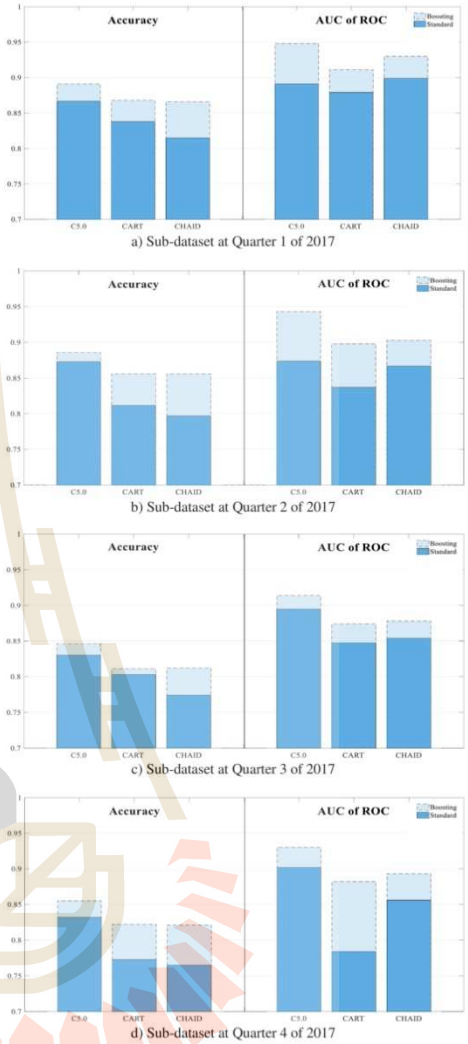
a) C5.0

b) CART

c) CHAID

Figure 9: The Predictor Importance Chart of DT Classifiers

From the results, we can say that a classifier from C5.0 method is the best job success predictor. While, CART and CHAID are in the second best group of predictors showing almost the same performance based on NECTEC HPC-workload dataset. Figure 10 shows validate results of test set using four quarters data subsets of the year 2017. The maximum accuracy of C5.0, CART, and CHAID in quarter 1 is 89.1%, 86.8%, and 86.6%, respectively. While the maximum AUC of ROC is 0.948, 0.911, and 0.930, respectively. In quarter 2 of 2017, the maximum accuracy of C5.0, CART, and CHAID is at 88.6%, 85.6%, and 85.6%, respectively. While the maximum AUC of ROC is 0.943, 0.898, and 0.903, respectively. In quarter 3 of 2017, the maximum accuracy of C5.0, CART, and CHAID is at 84.6%, 81.1%, and 81.2%, respectively. While the maximum AUC of ROC is 0.914, 0.878, and 0.878, respectively. In quarter 4 of 2017, maximum accuracy of C5.0, CART, and CHAID at 85.5%, 82.2%, and 82.1%, respectively. While the maximum AUC of ROC is 0.930, 0.882, and 0.893, respectively. From all results, C5.0 shows the best performance, while CART and CHAID are pretty much the same performance. The results of validation sub-set support our conclusion.



a) Sub-dataset at Quarter 1 of 2017

b) Sub-dataset at Quarter 2 of 2017

c) Sub-dataset at Quarter 3 of 2017

d) Sub-dataset at Quarter 4 of 2017

Figure 10: The Performance Comparison based on Test Set in each Quarter

C. Discussion

For descriptive analysis that uses Pearson correlation coefficient and Web-graph of counting statistic to explain the relation of quantitative and qualitative variables, respectively, the result of Pearson correlation coefficient among variables in the HPC-workload dataset is as expected with administrator's common sense. The result of Web-graph shows that mostly job

success had occurred on older computer machines more frequent than on the new computer machines. Perhaps this is because users of NECTEC HPC system stilled familiar to working with older computer machines or maybe the newly installed computer machines had a problem. Web-graph of counting statistic also shows that small job (using short queue type) has more success rate than big job (using medium or long queue type). The best performance of three proposed models is C5.0 that outperforms others on both of standard and ensemble models. The best of overall performance improvement after applying boosting technique with 10 component models is CHAID.

VI. Conclusion

This research studies classification model based on rule-base method. We had selected three common used DT methods including C5.0, CART, and CHAID. These DT methods were trained to build model through the learning process based on train set of HPC-workload dataset. Then, the model can predict class label of object on unseen test set of HPC-workload. The maximum accuracy performance of C5.0, CART, and CHAID after using boosting technique is 86%, 81.5%, and 79.8%, respectively. The maximum AUC of ROC graph of C5.0, CART and CHAID is 0.934, 0.868, and 0.872, respectively. The best improvement after using boosting technique of ensemble model is CHAID. It improved 5.25% compared to the standard model in terms of accuracy. The predictor variables that show high impact to response variable in proposed DT models are "CPU_USAGES", "MEMORY_USAGAE", "EXECUTE_TIME", and "CPU_CONSUMPTION_TIME" with respect to the NECTEC HPC-workload dataset.

Furthermore, this research uses descriptive analysis through Pearson correlation coefficient to study relations among qualitative variables of our HPC-workload dataset and using Web-graph of counting statistic to observe the relations of qualitative variables. The result shows that most success jobs on NECTEC HPC system are run on small (older) compute nodes with short queue type and most unsuccess (error) jobs are run on big (new) compute nodes with medium or long queue type.

Acknowledgment

The authors acknowledge "National e-Science Infrastructure Consortium" of NECTEC for providing the HPC-workload as a dataset used in this research. (URL: http://www.e-science.in.th.)

References

[1] N. W. Grady, "Knowledge Discovery in Data Science KDD meets Big Data," *IEEE Int. Conf. on Big Data 2016*, USA, pp. 1603-1608, Dec 2016.
[2] O. Adam, et al., "Advances and Challenges in Log Analysis," *Communications of the ACM Journal*, Vol. 55, Issue 2, pp. 55-61, Feb 2012.
[3] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *Informatica Journal*, Vol. 31, Issue 3, pp. 249-268, Oct 2007.
[4] B. Tomasz, et al., "A Method for Classification of Network Traffic Based on C5.0 Machine Learning Algorithm," *IEEE Int. Conf. on ICNC'12*, USA, pp. 237-241, Jan 2012.
[5] G. P. Siknun and I. S. Sitanggang, "Web-based Classification Application for Forest Fire Data Using the Shiny Framework and The C5.0 Algorithm," *Proc. Environmental Sciences (2nd LISAT-FSEM2015)*, Vol. 33, pp. 332-339, 2016.
[6] M. Khandelwal, et al., "Classification and Regression Tree Technique in Estimating Peak Particle Velocity Caused by Blasting," *Journal of Engineering with Computers*, Vol. 33, Issue 1, pp. 45-53, Jan 2017.
[7] M. Ramaswami and R. Bhaskaran, "A CHAID Based Performance Prediction Model in Educational Data Mining," *IJCSI International Journal of Computer Science Issues*, Vol. 7, Issue 1, No. 1, pp. 10-18, Jan 2010.
[8] C. D. Sutton, "Classification and Regression Trees, Bagging, and Boosting," *Handbook of Statistics*, Vol. 24, pp. 303-329, Feb 2005.
[9] Y. Xu, et al, "An Efficient Tree Classifier Ensemble-Based Approach for Pedestrian Detection," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 41, No. 1, pp. 107-117, Feb 2011.
[10] L. Rokach, "Ensemble-based classifiers," *An International Science and Engineering Journal (Artif Intell Rev)*, Vol. 33, Issue 1–2, pp. 1–39, Feb 2010.
[11] R. E. Banfield, et al., "A Comparison of Decision Tree Ensemble Creation Techniques" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, No. 1, pp. 173-180, Jan 2007.
[12] E. Chuah, et al., "Diagnosing the root-causes of failures from cluster log files," *Int. Conf. HiPC2010*, India, Dec 2010.
[13] Y. Orcun, et al., "On the Root Causes of Cross-Application I/O Interference in HPC Storage Systems," *IEEE Int. Conf. IPDPS2016*, USA, pp. 750-759, May 2016.
[14] J. Manish and H. T. Hassn, "A Review of Network Traffic Analysis and Prediction Techniques," *Tech. Report in computer science*, 2015.
[15] J. Klinkenberg, et al., "Data Mining-based Analysis of HPC Center Operations," *IEEE Int. Conf. on Cluster Computing 2017*, USA, pp. 766-773, Sep 2017.
[16] S. He, et al., "Experience Report: System Log Analysis for Anomaly Detection," *IEEE Int. Conf. 27th ISSRE2016*, Canada, pp. 207-218, Oct 2016.
[17] X. Fu, et al., "LogMaster: Mining Event Correlations in Logs of Large-Scale Cluster Systems," IEEE Int. Conf. 31st SRDS2012, USA, pp. 71-80, Oct 2012.
[18] A. Sirbu and O. Babaoglu, "A Holistic Approach to Log Data Analysis in High-Performance Computing Systems: The Case of IBM Blue Gene/Q," *Tech. Report in computer science*, Oct 2014.
[19] G. Kesavaraj, S. Sukumaran, "A Study on Classification Techniques in Data Mining," *in Proc. 4th ICCCNT*, India, July 2013.
[20] M. Galar, et al., "A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches," *IEEE Trans. Syst., Man, Cybern.* C. USA, Vol. 4, pp. 463-484, Jul 2012.
[21] Okfalisa, et al., "Comparative Analysis of K-Nearest Neighbor and Modified K-Nearest Neighbor Algorithm for Data Classification," *Int. Conf. 2nd ICITISEE2017*, Indonesia, pp. 294-298, Nov 2017.
[22] M. Agaoglu, "Predicting Instructor Performance Using Data Mining Techniques in Higher Education," *IEEE J. ACCESS.*, Vol. 4, pp. 2379-2387, June 2016.
[23] S. Senthil and W. M. Lin, "Applying Classification Techniques to Predict Students' Academic Results," *Int. Conf. ICCTAC2017*, India, Mar 2017.
[24] "Predicting Students Performance Using Decision Trees: Case of an Algerian University," *IEEE Int. Conf. on Mathematics and information Technology*, Algeria, pp. 113-121, Dec 2017.

International Journal of Machine Learning and Computing, Vol. 8, No. 6, December 2018

Multi-label Classification of High Performance Computing Workload with Variable Transformation

Anupong Banjongkan, Watthana Pongsena, Ratiporn Chanklan, Nittaya Kerdprasop, and Kittisak Kerdprasop

***Abstract*—High Performance Computing (HPC) log analysis is an active research domain. The challenge is how to extract the useful information from the HPC log file because the information resulting from the analysis can be used as a new knowledge to re-configure the HPC system for improving its efficiency. The traditional manner of HPC log analysis is considered inefficient in the sense that it is time-consuming and requires specific knowledge and skills of system administrator. In this research, we empirical study the application of machine learning techniques to perform an HPC log analysis task. We apply machine learning techniques that are different in their learning schemes including C5.0, Support Vector Machine (SVM), and Artificial Neuron Network (ANN) to analyze and predict the job status on the HPC system. We also propose a novel technique, which is called "Grouping & Combining". Grouping means reducing the class labels of the target variable. Doing so the time-consuming for analyzing is reduced. Then, the class labels of the target variable are combined with another variable such that the efficiency of the interpretability could be increased. The dataset used in our experiment is the real-world data obtained from the HPC system of the National Electronics and Computer Technology Center, or NECTEC, Thailand. According to the experimental results, the C5.0 model has the highest prediction accuracy at 88.74%. In contrast, the ANN model shows the best robustness. In addition, the experimental results show that the proposed Grouping & Combining technique can be efficiently used for handling the multi-label classification as it helps increasing the accuracy, consuming less time, and improving interpretability of the learned model.**

***Index Terms*—High performance computing workload, log analysis, multi-label classification, performance evaluation.**

I. Introduction

In the last decade, the computer technologies including hardware, software, and data storage are rapidly growing. Nowadays, the price of a computer is inverse with the performance of the equipment. In other words, today we can buy a cheaper computer equipment with higher performance than in the past. For this reason, the high performance computing (HPC) [1] or super computer is wildly used, and the performance of HPC is scaling up very fast. However, operating the HPC system is highly electricity consumption. Therefore, many researchers pay attention to the issue of improving performance of HPC [2]-[4].

The HPC log analysis is one effective way to remedy the power consumption problem. In general, data in a log file is the time series of the events that occur in the system. Currently, many researchers attempt to analyze the log, particularly the HPC log. The results from HPC log analysis may reveal characteristics of the system or uncover some useful information that can be used to improve the efficiency of the system. The traditional manner of the log analysis that manually performed by a human is inefficient. It takes a lot of time and requires the expert knowledge and high level of skills. Therefore, many researchers apply the data mining techniques for analyzing the log [5]-[7].

In this research, the data mining techniques are utilized to develop a model based on the workload dataset for predicting the job finish status of the HPC system. The job finish status is an important information. It can be used to fine-tune the system leading to the increase in the efficiency of the system. For the comparative study of the classification method, we select the three popular classifiers: C5.0 [8]-[10], support vector machine (SVM) [11]-[13], and artificial neural network (ANN) [14], [15]. The performance of the models is evaluated and compared based on the HPC-workload dataset, which is collected from the production HPC system of National Electronics and Computer Technology Center of Thailand (NECTEC). The log file is created by the PBS scheduler software that recorded the job information such as job wall time, job computation time, job finish status, and so on. The dataset consists of approximately 421,459 records with 27 variables. Besides performing log analysis with the job finish status as a single main target, we also consider enhancing ability of the model by applying the multi-label classification technique.

Multi-label classification is one of the challenges to many researchers in the machine learning (ML) domain [16]-[18]. The major difficulty is the multi-class value of the target variable. It can affect the classification performance. Indeed, the target variable of the dataset in this research contains a high multi-class value. We thus propose a technique to handle the problem of multi-label classification by transforming the target variable. This technique is called "Grouping & Combining".

The contributions of this research are as follows.

1) We demonstrate empirically the efficiency on analyzing the HPC-workload log file of the three famous ML techniques including C5.0, SVM, and ANN.

2) We propose a novel technique that can be efficiently used to handle high multi-label classification problem.

In the next section, we illustrate background knowledge and the existing work, which are related to this research. In Section III, we describe the classifier methods and the proposed technique. The HPC-workload dataset is described in Section IV. Section V demonstrates the experimentation. The last two sections (VI and VII) are the experimental

Manuscript received August 23, 2018; revised October 20, 2018.

The authors are with the School of Computer Engineering, Suranaree University of Technology (SUT), Thailand (e-mail: banjongkan@gmail.com, watthana.p@sskru.ac.th, arc_angle@hotmail.com, nittaya@sut.ac.th, kerdpras@sut.ac.th).

doi: 10.18178/ijmlc.2018.8.6.742

results and conclusions, respectively.

II. Background and Related Works

Log analysis is a popular research topic since the log file contains much information about hardware or software which is ordered in the time series fashion. Data mining is the efficient technique for extracting useful information from the log file. This research uses the data mining for conducting the comparative study of the three classifiers that can be used for analyzing and predicting the HPC-workload dataset. We focus on high multi-label classification issue.

A. Log Analysis

Q. Cao *et al.* [19] uses machine learning techniques to detect the abnormal sign from a cyber attack on web services. The decision tree (DT) and hidden Markov model (HMM) work together in this research. The dataset is collected from the industry sector. It contains around 4.6M records. The proposed method gives high accuracy (93.54%) for detecting the abnormal sign from cyber attack.

The existing work in [20] concerns about the quality of service (QoS). In order to keep high availability and reliability of service, the maintenance operation is very important. The problem is a high frequency of maintenance operation that leads to a high cost. Meanwhile, low frequency of maintenance operation is a risk. Therefore, this research fine-tunes optimum of the frequency of the maintenance operation. They use the classification-based ML technique to predict the system failure based on the data log file.

B. High Performance Computing Log Analysis

The research in [21] uses HMM method with frequency based strategy to focus only important log messages to predict the job remaining time in the supercomputing system. The maximum accuracy of the prediction is as high as 75%, and the error on job remaining time prediction is less than 200 seconds.

Y. Liang *et al.* [22] use the tagged logs from the BlueGene machine to discover the correlations between the fatal and non-fatal events. Then, they use these correlations for predicting the failures.

B. H. Park *et al.* [23] develop a framework for a deep monitoring of the HPC system. The framework composes of many tools, such as Cassandra (a highly scalable), the NoSQL distributed database (high performance column-oriented), and the Apache Spark (a real-time distributed in-memory analytics engine). The root cause analysis of the system failure is the focus of their research.

Yoo *et al.* [24] conduct a comparative study of classification methods including DT, Random Forest, Naive Bayes, and SVM with HPC-workload dataset from the Genepool scientific cluster at NERSC. Their research aims to find the patterns of unsuccessful job status. The result of the comparative study shows that the Random Forest method is the best with 99.8% accuracy, assessed with the 5-fold cross-validation method.

Although the existing research has widely studied on the log analysis, there is still a lack of research that applies the machine learning technique with the HPC-workload dataset. In addition, the existing research described in the literature review attempt to address the binary-class labels where the job status can be either success, or unsuccess. However, in the real world, the job finish status of the HPC system can be varied with miscellaneous multi-class values. Hence, the previous research may not have the ability for defining or expecting the root cause when the job finish status is in unsuccess state. This is a gap that our research aims to address.

III. Methods

This research uses the classification data mining technique to analyze and predict the job finish status in HPC-workload dataset. The classification is a supervised machine learning technique. There are so many algorithms available for the classification task. Those algorithms can be separated into three groups: 1) Linear, such as linear discriminant analysis (LDA) method, 2) Non-linear, such as SVM and ANN, 3) Rule-based or logic-based, such as DT. In this research, we study the non-linear and rule-based classifiers using C5.0, SVM, and ANN to conduct the comparative study.

In this work, we also propose a technique called "Grouping & Combining" that can be used to handle the high multi-label classification by transforming a single target variable to be a group of variables. This research uses IBM SPSS MODELER software for learning dataset and constructing the models. The simulation software runs on the machine with Intel Core-i5, CPU speed is 1.6 GHz and memory capacity is 8 GB.

A. Variable Transformation

We propose a "Grouping & Combining" technique to handle high multi-label classification as a main idea to transform the target variable in order to improve the prediction accuracy, and also to enhance the interpretability of the result. The "Grouping & Combining" technique composes of two steps described as follows.

The first step is called the grouping. This step reduces the wide range of multi-class values of the target variable in order to improve the prediction accuracy. We group the many possible values of target variable into the binary-class where the target class values consist of only "SUCCESS" and "UNSUCCESS". The advantage of this grouping is time reduction. However, the interpretability of the final result is also reduced. For example, if the model predicts the job finish status as "UNSUCCESS", we only know that the job is an error. But the root cause of that error cannot be identified by such result. Therefore, we have to perform the next step.

The second step is called the combining step. The objective of this step is to improve both accuracy and interpretability. The technique in this step is to create the new target variable and its class labels by combining the class labels of the original target variable with the class labels of another predictor variable. There are various ways to select the suitable predictor variable for the combination, such as making a choice based on the knowledge of expert, selecting from importance analysis of variables, or selecting from some statistical analysis techniques such as correlation coefficient analysis and factor component analysis (FCA). In this research, we rely on the knowledge of expert to select a predictor variable. Consequently, the "QUEUE_TPYE" is

selected, then its values are combined with the values of the target variable as demonstrated in Table I.

TABLE I: CLASS VALUES OF TARGET VARIABLE AFTER APPLYING "GROUPING & COMBINING" TECHNIQUE

Queue Type	Finish Status	New Target Variable
SHORT	SUCCESS	SHORT_SUCCESS
	UNSUCCESS	SHORT_UNSUCCESS
MEDIUM	SUCCESS	MEDIUM_SUCCESS
	UNSUCCESS	MEDIUM_UNSUCCESS
LONG	SUCCESS	LONG_SUCCESS
	UNSUCCESS	LONG_UNSUCCESS
OTHER	SUCCESS	OTHER_SUCCESS
	UNSUCCESS	OTHER_UNSUCCESS

B. Classification Techniques

C5.0 is a classifier in a group of rule-based or logic-based machine learning method. It is inherited from the C4.5 algorithm. Thus, C5.0 has all functions of C4.5. Moreover, it includes the new useful functions, such as boosting and cost-sensitive tree. The C5.0 uses information-gain as a criterion for splitting the tree branch. The advantage of the C5.0 is that it works very well with a big dataset because the nature of the algorithm uses less memory and it has high tolerance against the missing values. However, the disadvantages of this technique are that it supports only the categorical target variable and it does not work well with high multi-label classification.

Support vector machine (SVM) is an algorithm that builds classifier by searching for an optimum plane that can separate data with different class labels of target variable. The optimal plane is found from the applying the proper kernel function to transform data from the regular plane to the hyperplane. The popular kernel function of SVM is linear, radial basis function (RBF), polynomial and sigmoid. The original design of SVM is for the binary-label classification. Currently, SVM was developed to handle multi- label classification. The advantage of this method is that it can handle high multi-label classification and outlier. Meanwhile, the drawback is that it is difficult to fine-tune the appropriate parameters to achieve the best performance of the SVM model. In this research, we use the RBF kernel function and set the Gamma=1.0, C=3.

Artificial neural network (ANN) is developed from the concept of a human brain functioning. Normally, the ANN is composed of three main layers and it is called "Multilayer Perceptron". The first layer is the input layer. The second layer is the hidden layer. The hidden layer may contain more than one layer. The last layer is the output layer. The ANN works by propagating data into the input layer through the hidden layer. This process does multiply the input value with the weight, then, plus with the bias value of the hidden layer. The result is called "net value". Next process brings a net value into a transfer function for computing the final result, which is the output. This research configures multilayer perceptron as one input layer with 9 nodes, one hidden layer with 10 nodes, and one output layer with 1 node.

C. Quality Assessments

The accuracy is used to evaluate the performance of the classifier. Accuracy can be calculated from the confusion matrix as showed in Fig. 1. Besides the accuracy, we also consider another two important aspects of the performance including time-consuming and interpretability.

	Positive Predict	Negative Predict
Positive Actual	True Positive (TP)	False Negative (FN)
Negative Actual	False Positive (FP)	True Negative (TN)

Fig. 1. Confusion matrix of a two-class problem.

Generally, the accuracy is used to evaluate the overall predictive performance of the model. The accuracy is a ratio of the number of objects that the model can predict correctly divided by the number of totals objects as demonstrated in equation (1). The value of accuracy stays in the range between 0 and 1. The value nearly 1 means that the model has high accuracy performance, while the value converges to 0 means that the model has poor predictive performance.

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \quad (1)$$

where: TP = Number of objects that the model predicts as "True" and real class is "True".

TN = Number of objects that the model predicts as "False" and real class is "False".

FP = Number of objects that the model predicts as "True" but real class is "False".

FN = Number of objects that the model predicts as "False" but real class is "True".

The time-consuming is the time that spends for building the model and the time used in the predicting process. The less value is defined as better performance. In the perspective of interpretability, we define three levels including poor, neutral, and good. The description of each interpretability level is explained in Table II.

TABLE II: INTERPRETABILITY CRITERION

Level	Description
Good	Receive the specific information, can track to the root cause
Neutral	Receive the scope of information, can expect to the root cause
Poor	Receive the general information, cannot track to the root cause

IV. HIGH PERFORMANCE COMPUTING DATASET

There are many log files in the HPC system since the system consists of many hardware and software modules, such as a log file of compute node, a log file of network equipment, the HPC-workload log from scheduler software, and others. This research pays attention to the job success rate in the HPC system. Thus, the HPC-workload log is appropriate to be a dataset for our experimentation.

A. Data Collection

The HPC-workload dataset in this research is collected from the production HPC system of NECTEC. The name of this HPC system is "Atom cluster computer". It is a medium size HPC system that has totally 580 CPUs, 2.7 terabytes of memory, and 50 terabytes of disk storages. The Atom cluster computer has provided high computing power for many researchers in Thailand since 2012 to present. The total size

of HPC-workload dataset is 421,659 records with 27 variables. Figure 2 shows the raw form of the HPC-workload log. In the figure, only the three records of HPC-workload log are illustrated.

```
1 01/01/2017 00:37:22;E;92995             .nectec.or.th;user=c1280hd
   group=p128 jobname=Zr-S-Ads queue=long ctime=1483099421 qtime=148
  3099421 etime=1483099421 start=1483099422 owner=c1280
    .nectec.or.th exec_host=sodium-0-0.ib/11+sodium-0-0.ib/10+sodium
  -0-0.ib/9+sodium-0-0.ib/8 Resource_List.ncpus=1 Resource_List.need
  nodes=1:ppn=4 Resource_List.nodect=1 Resource_List.nodes=1:ppn=4 R
  esource_List.walltime=336:00:00 session=12466 end=1483205842 Exit_
  status=0 resources_used.cput=116:10:54 resources_used.mem=1747740k
  b resources_used.vmem=5130256kb resources_used.walltime=29:33:40
2 01/01/2017 11:13:47;Q;93015             .nectec.or.th;queue=long
3 01/01/2017 11:13:48;S;93015             .nectec.or.th;user=c1290hg
   group=p129 jobname=Ge-Defect-TS2 queue=long ctime=1483244027 qtim
  e=1483244027 etime=1483244027 start=1483244028 owner=c1290hg
        .nectec.or.th exec_host=sodium-0-0.ib/11+sodium-0-0.ib/10+s
  odium-0-0.ib/9+sodium-0-0.ib/8 Resource_List.ncpus=1 Resource_List
  .neednodes=sodium-0-0.ib:ppn=4 Resource_List.nodect=1 Resource_Lis
  t.nodes=1:ppn=4 Resource_List.walltime=336:00:00
```

Fig. 2. Example of record in realistic hpc workload dataset.

B. Data pre-Processing

Based on the raw data of the HPC-workload dataset, we select only data records during the year 2017. The dataset consists of 17,018 records. In addition, we select 9 variables from 27 variables for experimentation in this research. These selections follow the advice given by the knowledge expert who is the administrator of this system. The selected data attributes are 8 predictor variables and one target variable. The target field contains the number which has 32 possible values of exit code of job running in the system. Table III shows the details of these 9 variables.

Then, we split the dataset into four subsets. Each subset corresponds to each quarter of the year 2017. Therefore, the sizes of the four subsets (called quarter1, quarter2, quarter3, and quarter4) are 4,045, 5,007, 3,452 and 4,577 records, respectively. This data separation is a simple way to cross-check and validate the performance of the models.

TABLE III: DETAILS OF NINE VARIABLES IN THE DATASET

Feature	Description	Data Type
Queue Type	Queue system type in HPC	Categorical
Execute Host	Compute node that job running	Categorical
Finish Status	Exit code when job ending	Categorical
CPU Usage	Number of CPU that job requires	Numeric
Memory Usage	Memory space that job requires	Numeric
VMemory Usage	Memory space while job running	Numeric
Queueing Time	Time when job waiting in a queue	Numeric
Execute Time	The computation time of job	Numeric
CPU Time	Computation time x Number of CPU	Numeric

V. EXPERIMENTATIONS

We repeatedly perform the same set of experimentation steps on each of the four data-subsets. These steps are graphically shown in Fig. 3. In addition, we generate the three different scenarios:

1) "Raw" is a test case that target variable has not been transformed. The number of possible class values are 32.

2) "Grouping" is a test case that the target variable has been transformed by grouping the related class values into one group. The number of possible class values are 2.

3) "Grouping & Combining" is a test case that target variable has been transformed by combining related class values into one group and also grouping the class value with another predictor variable. The number of possible class values are 8.

Then, each test case is used for building and testing the performance of the models. In this process, we use 70% of data to build three different classifiers based on the three ML techniques including C5.0, SVM, and ANN. Then, the next process is class labeling or predicting process using the rest 30% of data. We evaluate the performance of models by the three measurements: the accuracy, time-consuming, and interpretability. Finally, the results based on each test case are analyzed.

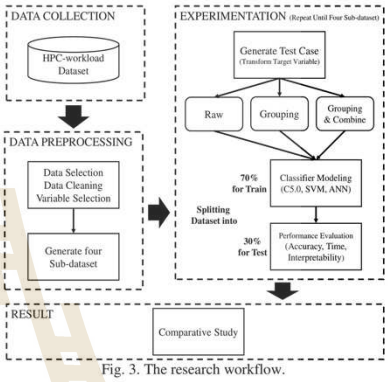


Fig. 3. The research workflow.

VI. EXPERIMENTAL RESULTS AND DISCUSSIONS

A. Experimental Results

The results on the first data subset, which is the Quarter1 in Table IV, show that the accuracy of C5.0 evaluated on the "Grouping & Combining" test case shows the highest accuracy performance at 88.74%. However, in the "Raw" test case, C5.0 cannot make the rule set. For the "Grouping" test case, the accuracy of C5.0 is 87.16%.

The SVM models yield the accuracy at 71.85%, 67.35% and 74.88% in "Raw", "Grouping" and "Grouping & Combining" test case, respectively. For the ANN classifier, the accuracy results are 77.49%, 75.91% and 80.26% in "Raw", "Grouping" and "Grouping & Combining" test case, respectively.

The results of the second data subset, which is the Quarter 2 in Table IV, show that the accuracy of C5.0 with "Grouping & Combining" test case is the best at 88.38%, while in "Raw" test case, C5.0 also cannot make the rule set. For the "Grouping" test case, the accuracy of C5.0 is 87.46%. SVM perform poorly at 69.88%, 70.01% and 72.63% of accuracy in the "Raw", "Grouping" and "Grouping & Combining" test case, respectively. For the ANN, the predicting accuracy is 77.08%, 73.16% and 76.05% accuracy in "Raw", "Grouping" and "Grouping & Combining" test case, respectively.

The results of the third data subset, which is the Quarter 3 in Table IV, show that the accuracy of C5.0 with "Grouping & Combining" test case is the best at 85.19%, while in the

"Raw" test cast, C5.0 cannot make the rule set. For the "Grouping" test case, the accuracy of C5.0 is 84.46%. The SVM models predict with the accuracy at 65.99%, 66.27% and 75.96% in "Raw", "Grouping" and "Grouping & Combining" test case, respectively. For the ANN, the accuracy results are 72.94%, 71.75% and 79.52% in the "Raw", "Grouping" and "Grouping & Combining" test case, respectively.

The results of the fourth data subset, which is the Quarter 4 in Table IV, show that the accuracy of C5.0 with the "Grouping & Combining" test case is the best at 84.21%, while in the "Raw" test cast, C5.0 also cannot make the rule set. For the "Grouping" test case, the accuracy of C5.0 is 83.14%. The SVM models give the prediction accuracy approximately 77.78%, 72.52% and 73.31% in the "Raw", "Grouping" and "Grouping & Combining" test case, respectively. For the ANN, the results are 79.66%, 72.81% and 74.03% of accuracy in the "Raw", "Grouping" and "Grouping & Combining" test case, respectively.

TABLE IV: ACCURACY OF THE MODELS ON EACH QUARTER

Classifier	Target Variable Class Label Transformation		
	Raw	Grouping	Grouping & Combining
QUARTER 1			
C5.0	n/a	87.163%	88.748%
SVM	71.857%	67.353%	74.881%
ANN	77.493%	75.911%	80.269%
QUARTER 2			
C5.0	n/a	87.467%	88.386%
SVM	69.886%	70.013%	72.638%
ANN	77.088%	73.163%	76.051%
QUARTER 3			
C5.0	n/a	84.461%	85.192%
SVM	65.998%	66.271%	75.961%
ANN	72.939%	71.755%	79.525%
QUARTER 4			
C5.0	n/a	83.142%	84.218%
SVM	77.788%	66.271%	73.314%
ANN	79.659%	71.755%	74.032%

TABLE V: TIME-CONSUMING OF THE THREE CLASSIFIERS

Dataset (Size)	Target Variable Class Label Transformation		
	Raw	Grouping	Grouping & Combining
Q1 (4,045)	14 Mins 49 Secs	8 Secs	8 Secs
Q2 (5,007)	15 Mins 5 Secs	9 Secs	11 Secs
Q3 (3,452)	15 Mins 23 Secs	6 Secs	7 Secs
Q4 (4,577)	14 Mins 39 Secs	7 Secs	7 Secs

TABLE VI: INTERPRETABILITY PERFORMANCE OF THE PROPOSED TRANSFORMATION TECHNIQUES

Transformation Technique	Accuracy	Time	Interpretability
Grouping & Combining	Good	Good	Neutral
Grouping	Good	Good	Poor
Raw	Poor	Neutral	Good

Meanwhile, the time-consuming of the machine learning process including the time for building the models and predicting the results has been observed. The result shows that the "Raw" test case takes longer time than the other test cases. The average time-consuming is around 15 minutes. For the "Grouping" and "Grouping & Combining" test cases, the average time-consuming is around 9 seconds as shown in Table V. The unit of dataset size is the number of records.

The interpretability of the results obtained from different kinds of models using various variable transformation techniques is summarized and shown in Table VI. In terms of all three criteria which are accuracy, time-consuming, and interpretability, it can be noticed that the "Grouping & Combining" transformation technique yields the best performance.

B. Discussions

One of the main contributions of this research is the proposal of a technique that provides classifiers the ability for handling the multi-label classification task. This research is a comparative study through the three classifiers which are C5.0, SVM, and ANN based on the realistic HPC-workload dataset. The result shows that C5.0 is the best classifiers in the "Grouping" and "Grouping & Combining" test cases. However, this classifier cannot return the result in the "Raw" test case. This means that the C5.0 cannot be used to handle the high multi-label classification data because the nature of the C5.0 algorithm supports only binary-label classification. Therefore, it is not suitable for the large dataset, which often contains a complex rule set or a deep tree. However, the C5.0 performs very well in the normal or low multi-class test cases.

For the non-linear classifiers including the SVM and ANN, the ANN is better than the SVM in terms of the predicting accuracy for all test cases. The performance of the SVM model is increased when choosing the proper kernel function and properly fine-tuning the gamma and C parameters according to the dataset characteristics.

VII. CONCLUSIONS

This research demonstrates the technique based on the data mining approach to analyze the HPC-workload dataset from production HPC system of the NECTEC. The target variable is the job finish status. It contains a wide range of multi-class values. The terms multi-class means the target variable has more than two values, and normally the values are much more than two. In this research, we propose a novel variable transformation technique called "Grouping & Combining", which can be effectively used for solving the high multi-label classification problem. The experimental results show that the proposed technique is sufficient to handle the high multi-label classification as it performs good results in terms of the predicting accuracy, time-consuming, and interpretability of the model. Furthermore, the results of the comparative study performed on different kinds of classifiers including the C5.0, SVM, and ANN show that the C5.0 model is potentially to be the best classifier as it shows the predicting accuracy as high as 88.74%. Meanwhile, the ANN is a classifier that is more likely to be the most robustness when considering from all kinds of test cases. For the future work, we plan to scale the experimentation to cover more classification methods and used other data domains.

ACKNOWLEDGMENT

The authors would like to acknowledge the "National e-Science Infrastructure Consortium" of NECTEC for providing the HPC-workload as a dataset used in this research (URL: http://www.escience.in.th). The first author has been supported by scholarship from Suranaree University

of Technology (SUT). The second author has been supported by scholarship from the Ministry of Science and Technology, Thailand. The third, fourth, and fifth authors are researchers of the Data and Knowledge Engineering Research Unit, which has been fully supported by research grant from SUT.

REFERENCES

[1] P. Uthayopas, T. Angskun, and J. Maneesilp, "Building a Parallel computer from cheap PCs: SMILE cluster experiences," in *Proc. the Second Annual National Symposium on Computational Science and Engineering*, p. 10.

[2] C.-H. Hsu and W.-C. Feng, "A power-aware run-time system for high performance computing," in *Proc. the 2005 ACM/IEEE Conference on Supercomputing*, 2005, p. 1.

[3] B. Schroeder and G. A. Gibson, "A large-scale study of failures in high performance computing systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 4, pp. 337–350, Oct. 2010.

[4] W. Huang, J. Liu, B. Abali, and D. K. Panda, "A case for high performance computing with virtual machines," A*CM Digital Library*, 2006, p. 125.

[5] A. Oliner, A. Ganapathi, and W. Xu, "Advances and challenges in log analysis," *Communications of the ACM*, vol. 55, no. 2, p. 55, Feb. 2012.

[6] E. Chuah *et al.*, "Enabling dependability-driven resource use and message log-analysis for cluster system diagnosis," in *Proc. IEEE 24th International Conference on High Performance Computing*, 2017, pp. 317–327.

[7] J. Klinkenberg, C. Terboven, S. Lankes, and M. S. Muller, "Data mining-based analysis of HPC center operations," in *Proc. 2017 IEEE International Conference on Cluster Computing*, pp. 766–773.

[8] Z. Sun, P. Leinenkugel, H. Guo, C. Huang, and C. Kuenzer, "Extracting distribution and expansion of rubber plantations from Landsat imagery using the C5.0 decision tree method," *Journal of Applied Remote Sensing*, vol. 11, no. 2, p. 026011, May 2017.

[9] T. Bujlow, T. Riaz, and J. M. Pedersen, "A method for classification of network traffic based on C5.0 machine learning algorithm," in *Proc. 2012 International Conference on Computing, Networking and Communications*, 2012, pp. 237–241.

[10] S. Pang and J. Gong, "C5.0 classification algorithm and application on individual credit evaluation of banks," *Systems Engineering-Theory & Practice*, vol. 29, no. 12, pp. 94–104, Dec. 2009.

[11] M. R. Kapadia and D. C. N. Paunwala, "Analysis of SVM kernels for content based image retrieval system," in *Proc. 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing*, p. 6.

[12] S. Porwal, D. S. A. Akbar, and D. S. C. Jain, "Leakage detection and prediction of location in a smart water grid using SVM classification," in *Proc. 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing*, p. 5.

[13] D. P. Kaucha, P. W. C. Prasad, A. Alsadoon, A. Elchouemi, and S. Sreedharan, "Early detection of lung cancer using SVM classifier in biomedical image processing," in *Proc. 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering*, p. 6.

[14] G. C. Jaiswal, M. S. Ballal, and D. R. Tutakne, "ANN Based Methodology for Determination of Distribution Transformer Health Status," in *Proc. 2017 7th International Conference on Power Systems*, pp. 133-138.

[15] S. A. Chandran and M. J. R. Panicker, "An Efficient Multi-Label Classification System Using Ensemble of Classifiers," in *Proc. 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies*, 2017, p. 4.

[16] Y. Feng, J. Jones, Z. Chen, and C. Fang, "An empirical study on software failure classification with multi-label and problem-transformation techniques," in *Proc. the IEEE 11th International Conference on Software Testing, Verification and Validation*, 2018, pp. 320–330.

[17] K.-H. Lo and H.-T. Lin, "Cost-sensitive encoding for label space dimension reduction algorithms on multi-label classification," in *Proc. 2017 Conference on Technologies and Applications of Artificial Intelligence*, p. 6.

[18] R. Kiran, B. R. Lakshmikantha, and R. V. Parimala, "Optimal placement of PMUs and analytics on PMU data using ANN technique," in *Proc. 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing*, 2017, pp. 922-926.

[19] Q. Cao, Y. Qiao, and Z. Lyu, "Machine learning to detect anomalies in web log analysis," in *Proc. 2017 3rd IEEE International Conference on Computer and Communications*, p. 5.

[20] J. Wang, C. Li, S. Han, and X. Zhou, "Predictive maintenance based on even-log analysis: A case study," *IBM Journal of Research and Development*, pp. 121–132, 2017.

[21] X. Chen, C.-D. Lu, and K. Pattabiraman, "Predicting job completion times using system logs in supercomputing clusters," in *Proc. 2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop*, 2013, pp. 1–8.

[22] Yinglung Liang, Yanyong Zhang, A. Sivasubramaniam, M. Jette, and R. Sahoo, "BlueGene/L failure analysis and prediction models," in *Proc. International Conference on Dependable Systems and Networks*, 2006, pp. 425–434.

[23] B. H. Park, S. Hukerikar, R. Adamson, and C. Engelmann, "Big data meets HPC log analytics: Scalable approach to understanding systems at extreme scale," in *Proc. 2017 IEEE International Conference on Cluster Computing*, Aug. 2017.

[24] W. Yoo, A. Sim, and K. Wu, "Machine learning based job status prediction in scientific clusters," in *Proc. 2016 SAI Computing Conference*, 2016, pp. 44–53.

Anupong Banjongkan is a Ph.D. student in computer engineering program with School of computer engineering, Suranaree University of Technology (SUT), Thailand. He graduated with a B.S. of computer science and master of engineering in electrical engineering from King Mongkut's University of Technology North Bangkok (KMUTNB), Thailand, in 2007 and 2011, respectively. His current research of interest includes high performance computing, machine learning, and knowledge discovery.

Watthana Pongsena is a Ph.D. student in School of Computer Engineering, Suranaree University of Technology (SUT), Thailand. He received his B.E. and M.E. in computer engineering from Suranaree University of Technology, Thailand, in 2008 and 2012, respectively. His research of interest includes software engineering, data mining, artificial intelligence, and human-computer interaction.

Ratiporn Chanklan is currently a researcher with the Data and Knowledge Engineering Research Unit at Suranaree University of Technology (SUT), Thailand. She received his bachelor degree in computer engineering from SUT in 2013, the master degree in computer engineering from SUT in 2014, and a doctoral degree in computer engineering from SUT in 2018. Her current research of interest includes classification, data mining, artificial intelligence.

Nittaya Kerdprasop is an associate professor and the head of Data Engineering Research Unit, School of Computer Engineering, SUT, Thailand. She received her B.S. in radiation techniques from Mahidol University, Thailand in 1985, M.S. in computer science from the Prince of Songkla University, Thailand in 1991 and Ph.D. in computer science from Nova Southeastern University, U.S.A. in 1999. Her research of interest includes data mining, logic and constraint programming.

Kittisak Kerdprasop is an associate professor at the School of Computer Engineering, SUT, and a Chair of the school. He received his bachelor degree in mathematics from Srinakarinwirot University, Thailand in 1986, MS in computer science from the Prince of Songkla University, Thailand, in 1991 and Ph.D. in computer science from Nova Southeastern University, U.S.A. in 1999. His current research includes machine learning and artificial intelligence.

International Journal of Machine Learning and Computing, Vol. 10, No. 1, January 2020

A Comparative Study of Learning Techniques with Convolutional Neural Network Based on HPC-Workload Dataset

Anupong Banjongkan, Watthana Pongsena, Nittaya Kerdprasop, and Kittisak Kerdprasop

***Abstract*—High-Performance Computing or HPC is a computer system that has high computing power. The HPC supports various computational domains. A huge amount of jobs from a large group of users prefer to complete their jobs in this kind of system. Therefore, managing the jobs or job scheduling is very important since it involves the overall system efficiency. The analysis of an HPC-workload log file is a solution to improve system efficiency. Because some information may appear in the log file, this information can help the system scheduler to make an appropriate decision for job scheduling in the HPC system. This research proposed predictive models for predicting the job status at the finishing state in the HPC system. The model can be used as a tool for monitoring the jobs in the HPC system. We develop and build the three models including HPC-CNN, HPC-AlexNet, and HPC-VGG16 based on the two different learning techniques, which comprise Initial and Transfer Learning of Convolutional Neural Network based on the HPC-work load dataset. Moreover, the three state-of-the-art Machine Learning methods: Classification and Regression Tree (CART), Artificial Neural Network (ANN), and Support Vector Machine (SVM) are used as the baseline models for performance comparison. The results show that the model that performs the best predictive performance is the proposed HPC-CNN model. It archives 76.48% accuracy of the prediction followed with the CART model (75.60%), while the SVM model performs lowest the accuracy at 66.80%.**

***Index Terms*—Convolutional Neural network, machine learning, transfer learning, high-performance computing, HPC-workload log.**

I. Introduction

The HPC systems provide computing power in many computation domains [1]-[5]. The type of jobs, which are computed on the HPC system is diverse since it combines various computing domains from different users. Therefore, the job management or job scheduling as called job scheduler [6]-[8] is very important for the HPC system. The system efficiency of the HPC system can be evaluated from the job success rate in the system. In other words, the performance of the scheduler affects the overall system efficiency of the HPC system. The efficiency of the system can be evaluated as the power that the system consumes and the job success rate. This means that the HPC system that has a high job success rate indicating the high efficiency of the system. Whereas, the low job success rate indicates the poor efficiency of the system.

Job scheduler like a brain of the HPC system. It is a middle-ware for receiving the job from users. Then, it sends the job to appropriate computing resources with the best strategy. In the job scheduling process, the scheduler records all events that occur in the system with numeric or string to the file as called the HPC-workload log file. This file can be used as source information for a system administrator to investigates or tracks the problems when problems occur in the system. Moreover, the HPC-workload log file may contain some hidden information that the administrator can be used to improve the efficiency of the system. For the traditional HPC-workload log analysis, an administrator manually analyzes using basic statistical methods based on their knowledge. Analyzing that data in this manner may be inefficient since it takes a long time to process, event there is no flexibility to be used for the generic model.

In the last decade, data mining techniques have been widely applied in the log analysis domain [9]. This research proposed the classifier models using Deep Learning with different learning techniques of CNN based on the HPC-workload dataset. The proposed models for predicting the job status at the finishing state in the HPC system. Meanwhile, the three state-of-the-art models of Machine Learning including CART, ANN, and SVM are created based on the same dataset that can be used as the baseline models. This research uses the HPC-workload log file as a dataset. The raw dataset contains 421,459 records. Each record consists of 27 attributes. The dataset is collected from the production of the HPC system named "Atom Computer Cluster" of National Electronics and Computer Technology Center (NECTEC) in Thailand. This HPC is operated under the National e-Science Infrastructure Consortium project. It provides the computing resources to support various computational projects in Thailand since the middle of 2012.

The main objective of this research is (i) to propose the developing and modeling classifier models using Deep Learning with different learning techniques of the CNN based on the HPC-workload dataset, (ii) to propose the comparative study of the performance of the models based on Deep Learning techniques and the models based on Machine Learning techniques including CART, ANN and SVM, and (iii) to demonstrate the advantages as well as disadvantages of the proposed models based on the real-world dataset.

Manuscript received March 27, 2019; revised December 29, 2019.

Anupong Banjongkan, Nittaya Kerdprasop, and Kittisak Kerdprasop are with the School of Computer Engineering, Suranaree University of Technology (SUT), Thailand (e-mail: banjongkan@gmail.com, nittaya@sut.ac.th, kerdpras@sut.ac.th).

Watthana Pongsena is with the School of Computer Engineering, SUT, and also with the Sisaket Rajabhat University, Thailand (e-mail: watthana.p@sskru.ac.th).

doi: 10.18178/ijmlc.2020.10.1.891

In the next section, we illustrate the existing works that relate to our research. Section III, the methodology and the dataset are explained in this section. The experimentation of this research is described in section IV. Section V illustrates the results. Section VI and VII illustrate the discussion and conclusion of this research, respectively.

II. Literature Review

The log file is a time series event-based record of the systems or applications while the process is online. The contents in a log file consist of many types of messages, such as only text, only numeral, or the combination of text and numeral. Analysis of the log file to extract useful information that investigates the root cause of the problem in order to find the suitable configuration of the system, the characteristic of the user's behavior, and etc. Currently, machine learning techniques play an important role in the log analysis domain. In this section, we describe the existing works that related to the use of a machine learning technique in the log analysis.

A. Log Analysis using Machine Learning

The existing works in network log analysis, D. J. Arndt and Zincir-Heywood [10] conduct a comparative study of the three classifier models to classify binary-class problems of encrypted network traffic (SSH encrypted or Non-SSH encrypted). The models are built based on machine learning methods, which include C4.5, K-means, and K-mean with Multi-Objective Genetic Algorithm (MOGA). This research shows C4.5 classifier archiving in overall performance. Meanwhile, the K-mean with MOGA gives the highest accuracy in some test cases as well as reduces time complexity of K-mean. Bujlow *et al.* [11] propose a classifier model with a decision tree method. The C5.0 method is applied to create the model for classifying the seven types of network traffic (Skype, FTP, P2P, Web, Web radio, Game, and SSH). The dataset in this research is a real-world dataset that is collected by their Volunteer-Based System. The result shows that their classifier has a better performance than the previous work. The performance in terms of accuracy of their model is approximately 99.3 - 99.9%.

Cao and Qiao [12] develop an Abnormal Detection System (ADS) for predicting the cyber-attack of the web (normal access or abnormal access) through the two levels of machine learning techniques. For the first level, they create three classifiers: logistic regression, decision tree, and support vector machine to label the data. For the second level, they choose the dataset, which is labeled from the best model according to the first level. Then, the classifier model is built with the Hidden Markov Model (HMM) based on the chosen dataset. The results in terms of performance comparison show that the proposed model archives the highest accuracy at 93.54%. The dataset is collected from the industrial.

Ertam and Kaya [13] conduct a comparative study of the classifiers for classifying the package permission, which composes of Allow, Deny, Drop and Reset-Both. The SVM algorithm is applied to build the model with different kernel functions including Linear, Polynomial, Radial Basis Function (RBF) and Sigmoid function. The dataset is a firewall log, which is taken from the firewall device of the Firat University. The result shows that the SVM classifier model using an RBF function overcomes other kernel functions with the best F_1 score at 76.4%.

B. High-Performance Computing Log Analysis

Hsu and Feng [14] propose a prototype of power awareness in the HPC system. The main objective of the research is to help the HPC system to reduce power consumption. This research uses the β-adaption Algorithm with Dynamic Voltage and Frequency Scaling technology to control the CPU workload in the system. Computer profiling (Real-time log) is used as a dataset. For experimentation, the model runs using benchmark applications. The result demonstrates that the proposed method reduces the power consumption of the HPC system around 20% for sequential Benchmark test cases and 25% for the parallel benchmark test case.

Taerat, *et al.* [15] conduct research using descriptive analysis to explain the characteristics of the HPC system based on system failures. The HPC log file of the IBM Blue Gene/L system of Louisiana Tech University is used as a dataset. The result shows in terms of the enumerated information, such as the severity level of failures, time to repair (TTR) or mean time to failures (MTTF). The conclusion of the analysis assumes a time to repair (TTR) as 10 minutes. Then, the results suggest that the system has a mean time to failure (MTTF) at 5.89 hours, or around 4 times a day.

Pelaez *et al.* [16] develop a system failure detection through the improvement of the clustering algorithm. The proposed method so-called Decentralized Online Clustering (DOC). The system is built based on a case study of the Ranger supercomputer of the Texas Advanced Computing Center. The result illustrates that the performance of the system failure detection is not different compared to the baseline. Meanwhile, the proposed model reduces approximately 2% of the overall overhead (CPU, memory and network bandwidth).

Klinkenberg *et al.* [17] propose a monitoring system for predicting system failures for the HPC system of the RWTH Aachen University. The first phase of the research uses a descriptive statistic to identify the events through the characteristics of the event. In the second phase, a comparative study of classification methods: logistic regression, decision tree, random forest, SVM, and multilayers perceptron based on preprocessing data in the first phase. The performance evaluation using 10-fold cross-variation demonstrates that the proposed model archives 98% precision and 91% recall.

Yoo, Sim, and Wu [18] conduct a comparative study using six methods of machine learning including decision tree, random forest, logistic regression, and naïve bayes to build the classifier models for predicting the job unsuccess at running state in the HPC system. The dataset is an HPC-workload log file of Genepool Scientific Cluster Computer of the NERSC. The result shows that the model based on the random forest method outperforms other models. The performance of the classifier archives 99.8% accuracy, 83.6% recall and 94.8% precision.

In conclusion, the literature review we mentioned above

demonstrates that machine learning techniques are widely applied in many log analysis domains, especially for HPC log analysis. However, in this research, we proposed classifier models using deep learning techniques with different learning techniques of the convolutional neural network.

III. Research Methodology

The classification technique is a technique in machine learning. It is a supervised learning technique to classify or predict binary or multi-label classification problems. Currently, deep learning is a subset of machine learning that becomes a popular technique in the artificial intelligent domain. This research applies deep learning techniques with the CNN method to develops classifier models for predicting the job status at the finishing state in the HPC system.

A. Convolutional Neural Network

The Convolutional Neural Network (CNN) also known as ConvNet is an algorithm, which uses the process of the neural network. The architecture and process of neural networks mimic the process of the human brain. Therefore, the ConvNet is a popular algorithm in deep learning technique that has been applied in many domains, such as the self-driving car system [19], medical science [20], [21], and environmental science [22].

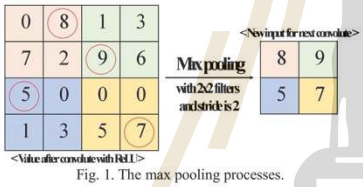


Fig. 1. The max pooling processes.

The architecture of the ConvNet composes of the input layer (receive input data), the hidden layer (computational process), and the output layer (classify or predict the result). In a part of the hidden layer of the ConvNet, it is different from the normal neural network. Therefore, it can be separated into two main procedures. The first procedure is the process of convolution to maintain the value with Rectified Linear Unit (ReLU). The second procedure reduced features using pooling techniques (select one feature in the region). Then, the network re-processes the two procedures again until finish convolution loop. Fig. 1 shows the example of the max pooling technique.

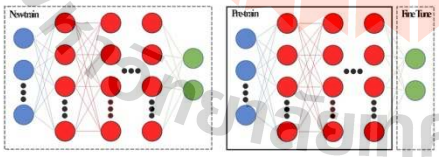


Fig. 2. Initial learning (a) and transfer learning (b).

There are two techniques to build a classifier model through the ConvNet algorithm. The first technique is the initial learning technique [23], [24]. This technique creates all-new architecture as well as initial learning the data from zero at the model learning state. The second technique is the transfer learning technique [25], [26]. This technique uses the pre-train network with fine-tune technique and modifies the pre-train network according to the dataset. The transfer learning technique reduces the model learning time in the learning state. Generally, this technique suits for using with the general image. Fig. 2 shows a comparison of the initial learning and transfer learning techniques. In this research, we use AlexNet and VGG16 as the pre-train network. Table I shows the characteristics of the pre-train network.

TABLE I: Property of the Pre-train Network

Pre-train	Layers	Hyper Parameter	Input Size
AlexNet	8	61M	227×227
VGG16	16	138M	224×224

B. The State-of-The-Art Machine Learning

Machine learning algorithms are divided into two groups according to the learning process including supervised and unsupervised learning. The supervised learning means the target variable must be defined at the learning state while unsupervised learning the target variable has not to be defined at the learning state. Mostly, the algorithms that propose classification and regression tasks are grouped as a supervised learning technique. Meanwhile, the algorithms that propose a clustering task are grouped as an unsupervised learning technique. In this paper, we use the three state-of-the-art machine learning algorithms including Classification and Regression Tree (CART), Artificial Neural Network (ANN), and Support Vector Machine (SVM) building as the baseline models.

The CART is a tree based-algorithm [27]. The CART algorithm supports the model for classification as well as a regression task. In other words, this algorithm can be used with the dataset that is a categorical and continuous type of target variable. Therefore, the learning data to create the tree structure rule of the CHART algorithm are the Gini index value and variance reduction criterion for classification and regression task, respectively.

Artificial Neural Network (ANN) [28] is an algorithm developed from the motivation of the human brain works. Typically, the ANN architecture composes of three parts including the input layer, hidden layer, and an output layer. The multilayer perceptron is a basis of ANN architecture (one input layer, one hidden layer, and one output layer). The process of the ANN algorithm sends the data into the input layer, and then, propagates the data into the hidden layer. At the same time, the input values are computed by multiplying the weight including the bias values. The result is called "net input". Then, the activate function is taken to the net input. Finally, the result is processed in the output layer for classifying the data.

	Positive Predict	Negative Predict
Positive Actual	True Positive (TP)	False Negative (FN)
Negative Actual	False Positive (FP)	True Negative (TN)

Fig. 3. The confusion matrix of a two-class problem.

Support Vector Machine (SVM) [29] is an algorithm that

finds the appropriate line to separate the data in a hyperplane. The line is defined from a mathematical function called kernel function. The popular SVM kernel function is Linear, Radial Basis Function, Polynomial, and Sigmoid. Previously, the native SVM supports only binary-label classification problems. Presently, modern SVM can be used to handle the multi-label classification problem as well as increasing the robustness to the outliner. However, finding the appropriate kernel function of the SVM algorithm is a difficult task.

C. Assessments

To evaluate the performance of the models, we select the four evaluators including accuracy, recall, precision, and F-measure. All evaluators are computed from the confusion matrix table. Fig. 3 illustrates the example of the confusion matrix of a two-class problem.

The true positive (TP) is the number of the predicted value is "True", and the actual value is "True". The false negative (FN) is the number of the predicted value is "False", while the actual value is "True". The false positive (FP) is the number of the predicted value is "True", while the actual value is "False". The true negative (TN) is the number of the predicted value is "False", and the actual value is "False".

The accuracy (1) is an evaluator that assesses the overall performance of the model. The recall (2) regards the model performance based on the actual value point view. Meanwhile, the precision (3) observe the model performance base on the predicted value point of view. The F-measure or F_1 score (4) is a harmonic mean of precision and recall.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

```
1 01/01/2017 00:37:22;E;92995                .nectec.or.th;user=c1280hd
  group=p128 jobname=Zr-S-Ads queue=long ctime=1483099421 qtime=148
  3099421 etime=1483099421 start=1483099422 owner=c1280
    .nectec.or.th exec_host=sodium-0-0.ib/11+sodium-0-0.ib/10+sodium
  -0-0.ib/9+sodium-0-0.ib/8 Resource_List.ncpus=1 Resource_List.need
  nodes=1:ppn=4 Resource_List.nodect=1 Resource_List.nodes=1:ppn=4 R
  esource_List.walltime=336:00:00 session=12466 end=1483205842 Exit_
  status=0 resources_used.cput=116:10:54 resources_used.mem=1747740k
  b resources_used.vmem=5130256kb resources_used.walltime=29:33:40
2 01/01/2017 11:13:47;Q;93015                .nectec.or.th;queue=long
3 01/01/2017 11:13:48;S;93015                .nectec.or.th;user=c1290hg
  group=p129 jobname=Ge-Defect-TS2 queue=long ctime=1483244027 qtim
  e=1483244027 etime=1483244027 start=1483244028 owner=c1290hg
        .nectec.or.th exec_host=sodium-0-0.ib/11+sodium-0-0.ib/10+s
  odium-0-0.ib/9+sodium-0-0.ib/8 Resource_List.ncpus=1 Resource_List
  .neednodes=sodium-0-0.ib:ppn=4 Resource_List.nodect=1 Resource_Lis
  t.nodes=1:ppn=4 Resource_List.walltime=336:00:00
```

Fig. 4. The example of some records in the raw HPC-workload log file.

D. Dataset and Tools

This research uses the dataset, which is collected from the National Electronics and Computer Technology Center, Thailand or NECTEC. The dataset is an HPC-workload log from the PBS/Torque scheduler in a production computer cluster called "Atom computer cluster". Atom computer cluster is a medium size HPC system in Thailand that composes of 580 computing elements, 2.7 TBytes of memory, and 50 TBytes of the storage capacity. This system provides free-computing resources for the research in Thailand since mid-2012. The raw HPC-workload log file contains 421,659 records. Each record composes of 27 attributes. Fig. 4 illustrates the example of the raw data of the HPC-workload log file.

In this research, we use MATLAB software version R2018b to build the models through different learning techniques of the CNN network. In addition, we use the IBM SPSS Modeler 18.0 for creating the baseline classifier models through the machine learning methods. Moreover, we use Python 3.4 to handle the raw HPC-workload log in the data preprocess state. All experimentation is run on the working station computer (Intel Xeon Silver 4116 CPU, 2.10 GHz, 24 GB of memory without GPU).

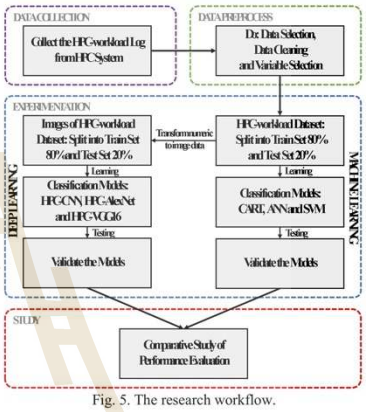


Fig. 5. The research workflow.

IV. Experimentation

In this research, we divide the experimental process into four main parts including data collection, data preprocessing, experimentation, and analyzing the results. The data collection process has already been described in section III. The experimental results and discussions are presented in sections V and VI, respectively. In this section, we describe the data preprocessing and experimentation parts.

TABLE II: Details of All Attributes in the Dataset

Attribute	Description	Type
Queue Type	Queue type in HPC	Categorical
Execute Host	Compute node at job running	Categorical
Limit Wall Time	Time limit which depends on queue type	Continuous
CPU Usage	Number of CPU at the job requires	Continuous
Memory Usage	Memory space at job requires	Continuous
VMemory Usage	Memory space while job running	Continuous
Queueing Time	Time of job waiting in the queue	Continuous
Wall Time	Total time of job stay in HPC	Continuous
Execute Time	The computation time of the job	Continuous
CPU Time	Computation time × CPU Usage	Continuous
Finish Status	Exit code at job ending	Categorical

A. Data Preprocess

After the raw data is collected from the system, we prepare the dataset through the data preprocessing process. This

process makes a suitable dataset for the experimental process. This dataset is a good quality one since, in the year 2016, the HPC system has a little downtime (around 6%). Then, we clean the data by eliminating outliers and missing values. Next, we select 11 out of 27 attributes using expert knowledge. There are 10 predictor variables including "Queue Type", "Execute Host", "CPU Usage", "Memory Usage", "VMemory Usage", "Queueing Time", "Execute Time", "CPU Time", "Limit Wall Time", and "Wall Time". The "Finish Status" is a target variable in this research. The target variable is a binary-class problem that composes of "success" and "error" state. Table II shows details of all attributes in the dataset.

B. Classifiers Modelling

In this process, we use the HPC-workload dataset that is already prepared according to the previous process. We separate the experimentation into two phases. The first phase according to the main objective (i) of this research that is to model the classifier models for predicting the job status at the finishing state of the HPC system. The models are built through the different learning techniques of the CNN network. The second phase according to the objective (ii) of this research that builds the three baseline models through the machine learning methods, which include CART, ANN, and SVM.

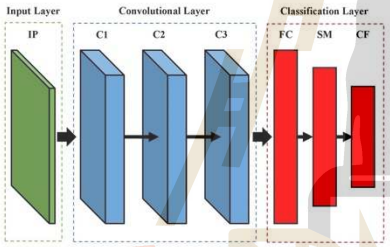


Fig. 6. The HPC-CNN architecture.

In the first phase of the experiment, we propose the three classifier models to predict the job status at the finishing state of the HPC system based on the HPC-workload dataset. A deep learning technique is used to build models. The HPC-CCN is the proposed model, which is modeled through the initial learning technique of the CNN network. The network architecture and configuration of the HPC-CNN are defined using expert knowledge as showed in Fig. 6. The other two proposed models are HPC-AlexNet and HPC-VGG16. These models are built using the transfer learning technique of the CNN network. The AlexNet and VGG16 are used as a pre-train network for HPC-AlexNet and HPC-VGG16, respectively. In the transfer learning process, we fine-tune the three layers of the output port of the pre-train networks. As the input of the proposed models must be an image, we perform an extra-process for transforming the HPC-workload dataset into an image dataset. In this process, the categorical value of the predictor variable is changed to be a nominal value. Then, the numeric value in a dataset is normalized to 0 - 255. At the end of this process, the HPC-workload dataset is ready transformed into the image data. The image data is created one by one from rows in the HPC-workload dataset. We duplicate nine times of row to be 10×10 pixels image data (Fig. 7). Fig. 8 shows an example of the image data after the transformation process is done. The color channel of the image data for the HPC-CNN model is 1 channel (grayscale), while the proposed models, which are modeled from the pre-train network (HPC-AlexNet and HPC-VGG16) are 3 channels (RGB). After the HPC-workload image dataset is created, we randomly split the dataset into 80% train-set and 20% test-set. The three proposed models (HPC-CNN, HPC-AlexNet, and HPC-VGG16) are built from the train-set with the same configuration as shown in Table III. The accuracy, recall, precision, and F-measure score are used to evaluate the performance evaluation of the proposed models. Then, the model, which perform the best performance is selected in order to compare its performance with the baseline models.

	1	2	3	4	5	6	7	8	9	10
1	36	125	113	81	0	44	127	11	255	76
2	36	125	113	81	0	44	127	11	255	76
3	36	125	113	81	0	44	127	11	255	76
⋮	⋮				⋮				⋮	
10	36	125	113	81	0	44	127	11	255	76

Fig. 7. The example of a 2D array 10×10 for creating the image data.

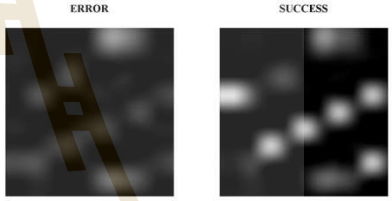


Fig. 8. The example of grayscale image data with a label (x50).

TABLE III: THE CONFIGURATION OF THE LEARNING PROCESS

Parameter	Value
Optimizer	sgdm
Mini Batch Size	100
Max Epochs	3
Initial Learning	0.1
Validation Frequency	10
Validation Patience	Inf

In the second phase, we create the baseline models for our comparative study with the proposed model. The three machine learning methods including CART, ANN, and SVM are used to build the baseline classifier models. We also randomly split the dataset into 80% train-set and 20% test-set. For the ANN configuration, it composes of one input layer with 10 neural nodes, one hidden layer with 7 neural nodes, and the output layer has 1 neural node. For SVM, the RBF kernel function is applied with the Gamma = 0.1 and $C = 3$.

V. RESULTS

From the experimentation, the three CNN network models HPC-CNN, HPC-AlexNet and HPC-VGG16 that are the classifier model, which are used to predict the job status at the finishing state in the HPC system. The performance

evaluation of the models illustrates that the HPC-CNN model archives the highest accuracy at 73.55%. In a part of the two models, which are built using the transfer learning technique, HPC-AlexNet and HPC-VGG16 provide the accuracy of the prediction at 57.35% and 42.65%, respectively. For the performance in terms of recall, precision, and F-measure, only HPC-CNN returns the results. The results are 59.69% recall, 73.35% precision, and 65.79% F-measure score as shown in Table IV. The model building time of the three models shows that the HPC-CNN model takes less time of 5 minutes and 13 seconds while the HPC-VGG16 takes the longest time at 11 hours (Table V). Fig. 9 shows the confusion matrix of the HPC-CNN model.

TABLE IV: THE PERFORMANCE OF THE THREE PROPOSED MODELS

Model	Learning Techniques	Evaluators			
		Accuracy	Recall	Precision	F_1
HPC-CNN	Initial Learning	**0.735**	**0.596**	**0.733**	**0.658**
HPC-AlexNet	Transfer Learning	0.573	0	n/a	n/a
HPC-VGG16	Transfer Learning	0.426	0	n/a	n/a

TABLE V: TIME CONSUMPTION AT THE LEARNING PROCESS

Model	Time Consumption
HPC-CNN	**5 min 13 sec**
HPC-AlexNet	93 min 3 sec
HPC-VGG16	698 min 50 sec

We increase the epoch at the learning state of HPC-CNN up to 18 epochs (Fig. 10). As a result, the accuracy of the model increases up to 76.49%. Table VI illustrates the performance of the HPC-CNN model compared with the baseline models. The results show that the HPC-CNN model (76.5% accuracy) outperforms the others as shown in Table VI.

TABLE VI: THE PERFORMANCE COMPARISON OF PROPOSED VS BASELINE

Machine Learning	Accuracy
CART	0.754
ANN	0.729
SVM	0.668
Deep Learning	Accuracy
HPC-CNN	**0.765**

VI. DISCUSSION

The result shows that the performance of the HPC-AlexNet and HPC-VGG16 models are very poor as they cannot return the results of recall, precision, and F-measure. Based on this result, it could be concluded that the models, which are built using the transfer learning technique of CNN from the pre-train networks (AlexNet and VGG16) are unsuitable for the HPC-workload dataset. This could be because the network architecture of the pre-train networks is inconsistent with the input data. In other words, there are some unnecessary of the hidden layers (convolutional part) since the HPC-workload is a low dimensional dataset. This conclusion seems to be supported by the result that the HPC-CNN network archives higher accuracy than the HPC-AlexNet and HPC-VGG16. This is possibly because it has only three hidden layers.

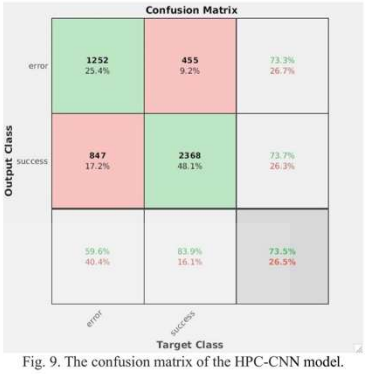


Fig. 9. The confusion matrix of the HPC-CNN model.

VII. CONCLUSION

This research proposed classifier models to predict the job status at the finishing state of the HPC system based on the HPC-workload dataset. The two learning techniques including initial and transfer learning of the CNN network is utilized to model the proposed models. The HPC-CNN network uses the initial learning technique of the CNN network. Meanwhile, HPC-AlexNet and HPC-VGG16 use the transfer learning technique. The AlexNet and VGG16 network is used as the pre-train network. The performance comparison of three proposed models demonstrates that the HPC-CNN model archives the highest accuracy at 76.5%. Moreover, this research is a comparative study of the proposed model with the three state-of-the-art machine learning methods including CART, ANN, and SVM. The results show that the proposed HPC-CNN network outperforms the others with 76.49% accuracy, while the baseline models CART, ANN, and SVM provide the accuracy of the prediction at 75.4%, 72.9%, and 66.8%, respectively.

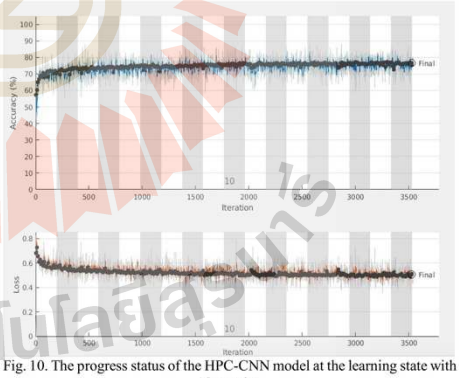


Fig. 10. The progress status of the HPC-CNN model at the learning state with 18 epochs.

For the future work, we will apply a grid search or random search to find the best CNN configurations based on the HPC-workload dataset and increase the scale of the dataset in

order to enhance the performance of the model.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

The first author is designing the research framework, organizing the experimentation steps and preparing the manuscript. The second author helps to validate the manuscript. The third author had approved the final version. The last author takes part in the experimentation design.

Acknowledgment

The authors would like to acknowledge the "National e-Science Infrastructure Consortium" of NECTEC for providing the HPC-workload as a dataset that we use in this research (URL: http://www.escience.in.th). The first author has been supported by a scholarship from the Suranaree University of Technology (SUT). The second author has been supported by a scholarship from the Ministry of Science and Technology, Thailand. The third, fourth, and fifth authors are researchers of the Data and Knowledge Engineering Research Unit, which has been fully supported by a research grant from SUT.

References

[1] V. Sipkova, L. Hluchy, M. Dobrucky, J. Bartok, and B. M. Nguyen, "Manufacturing of weather forecasting simulations on high-performance infrastructures," in *Proc. 2016 IEEE 12th International Conference on e-Science (e-Science)*, Baltimore, MD, USA, 2016, pp. 432–439.

[2] S. Sehrish, J. Kowalkowski, M. Paterno, and C. Green, "Python and HPC for high energy physics data analyses," in *Proc. the 7th Workshop on Python for High-Performance and Scientific Computing*, Denver, CO, USA, 2017, pp. 1–8.

[3] R. Dolezal, T. C. Ramalho, T. C. C. França, and K. Kuca, "Parallel flexible molecular docking in computational chemistry on high-performance computing clusters," in *Computational Collective Intelligence*, M. Núñez, N. T. Nguyen, D. Camacho, and B. Trawiński, Eds. Cham: Springer International Publishing, 2015, vol. 9330, pp. 418–427.

[4] A. Kawalia *et al.*, "Leveraging the power of high-performance computing for next generation sequencing data analysis: Tricks and twists from a high throughput exome workflow," *PLOS ONE*, vol. 10, no. 5, p. e0126321, May 2015.

[5] E. J. Nielsen and B. Diskin, "High-performance aerodynamic computations for aerospace applications," *Parallel Computing*, vol. 64, pp. 20–32, May 2017.

[6] A. Reuther *et al.*, "Scheduler technologies in support of high-performance data analysis," in *Proc. 2016 IEEE High-Performance Extreme Computing Conference (HPEC)*, Waltham, MA, USA, 2016, pp. 1–6.

[7] M. Etinski, J. Corbalan, J. Labarta, and M. Valero, "Parallel job scheduling for power constrained HPC systems," *Parallel Computing*, vol. 38, no. 12, pp. 615–630, Dec. 2012.

[8] Z. R. M. Azmi, K. A. Bakar, M. S. Shamsir, N. W. Manan, and A. H. Abdullah, "Scheduling grid jobs using priority rule algorithms and gap filling techniques," *International Journal of Advanced Science and Technology*, vol. 37, p. 16, 2011.

[9] A. Oliner, A. Ganapathi, and W. Xu, "Advances and challenges in log analysis," *Communications of the ACM*, vol. 55, no. 2, p. 55, Feb. 2012.

[10] D. J. Arndt and A. N. Zincir-Heywood, "A comparison of three machine learning techniques for encrypted network traffic analysis," in *Proc. 2011 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, Paris, France, 2011, pp. 107–114.

[11] T. Bujlow, T. Riaz, and J. M. Pedersen, "A method for classification of network traffic based on C5.0 machine learning algorithm," in *Proc. 2012 International Conference on Computing, Networking and Communications (ICNC)*, Maui, HI, USA, 2012, pp. 237–241.

[12] Q. Cao, Y. Qiao, and Z. Lyu, "Machine learning to detect anomalies in web log analysis," in *Proc. 2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, Chengdu, 2017, pp. 519–523.

[13] F. Ertam and M. Kaya, "Classification of firewall log files with multiclass support vector machine," in *Proc. 2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, Antalya, 2018, pp. 1–4.

[14] C.-H. Hsu and W.-C. Feng, "A power-aware run-time system for high-performance computing," in *Proc. ACM/IEEE SC 2005 Conference (SC'05)*, Seattle, WA, USA, 2005, pp. 1–1.

[15] N. Taerat, N. Naksinehaboon, C. Chandler, J. Elliott, G. Ostrouchov, and S. L. Scott, "Using Log Information to Perform Statistical Analysis on Failures Encountered by Large-Scale HPC Deployments," in *Proc. In High Availability and Performance Computing Workshop*, 2008, p. 6.

[16] A. Pelaez, A. Quiroz, J. C. Browne, E. Chuah, and M. Parashar, "Online failure prediction for HPC resources using decentralized clustering," in *Proc. 2014 21st International Conference on High-Performance Computing (HiPC)*, Goa, India, 2014, pp. 1–9.

[17] J. Klinkenberg, C. Terboven, S. Lankes, and M. S. Muller, "Data mining-based analysis of HPC center operations," in *Proc. 2017 IEEE International Conference on Cluster Computing (CLUSTER)*, Honolulu, HI, USA, 2017, pp. 766–773.

[18] W. Yoo, A. Sim, and K. Wu, "Machine learning based job status prediction in scientific clusters," in *Proc. 2016 SAI Computing Conference (SAI)*, London, United Kingdom, 2016, pp. 44–53.

[1] A. Shustanov and P. Yakimov, "CNN design for real-time traffic sign recognition," *Procedia Engineering*, vol. 201, pp. 718–725, 2017.

[19] A. Pal, U. Garain, A. Chandra, R. Chatterjee, and S. Senapati, "Psoriasis skin biopsy image segmentation using deep convolutional neural network," *Computer Methods and Programs in Biomedicine*, vol. 159, pp. 59–69, Jun. 2018.

[20] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Computers and Electronics in Agriculture*, vol. 145, pp. 311–318, Feb. 2018.

[21] C. Zhang, J. Yan, C. Li, H. Wu, and R. Bie, "End-to-end learning for image-based air quality level estimation," *Machine Vision and Applications*, vol. 29, no. 4, pp. 601–615, May 2018.

[22] N. Kondo, W. Chinsatit, and T. Saitoh, "Pupil center detection for infrared irradiation eye image using CNN," in *Proc. 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, Kanazawa, 2017, pp. 100–105.

[23] Q. Le, O. Boydell, B. Mac Namee, and M. Scanlon, "Deep learning at the shallow end: Malware classification for non-domain experts," *Digital Investigation*, vol. 26, pp. S118–S126, Jul. 2018.

[24] C. Boufenar, A. Kerboua, and M. Batouche, "Investigation on deep learning for off-line handwritten Arabic character recognition," *Cognitive Systems Research*, vol. 50, pp. 180-195, Aug. 2018.

[25] D. Han, Q. Liu, and W. Fan, "A new image classification method using CNN transfer learning and web data augmentation," *Expert Systems with Applications*, vol 95, pp. 43-56, Apr. 2018.

[26] B. Choubin, H. Darabi, O. Rahmati, F. Sajedi-Hosseini, and B. Kløve, "River suspended sediment modeling using the CART model: A comparative study of machine learning techniques," *Science of The Total Environment*, vol. 615, pp. 272–281, Feb. 2018.

[27] H. Li, F. Chung, and S. Wang, "An SVM based classification method for homogeneous data," *Applied Soft Computing*, vol. 36, pp. 228–235, Nov. 2015.

[28] R. Jafari-Marandi, S. Davarzani, M. S. Gharibdousti, and B. K. Smith, "An optimum ANN-based breast cancer diagnosis: Bridging gaps between ANN learning and decision-making goals," *Applied Soft Computing*, vol. 72, pp. 108–120, Nov. 2018.

Anupong Banjongkan is a Ph.D. student in computer engineering program with the School of computer engineering, Suranaree University of Technology (SUT), Thailand. He graduated with B.S. of computer science and master of engineering in electrical engineering at the King Mongkut's University of Technology North Bangkok (KMUTNB), Thailand, in 2007 and 2011,

respectively. His current research of interest includes high-performance computing, machine learning, and knowledge discovery.

Watthana Pongsana is a Ph.D. student in the School of Computer Engineering, Suranaree University of Technology (SUT), Thailand. He received his B.E. and M.E. in computer engineering from Suranaree University of Technology, Thailand, in 2008 and 2012. His research of interest includes software engineering, data mining, artificial intelligence, and human-computer interaction.

Nittaya Kerdprasop is an associate professor and the head of Data Engineering Research Unit, School of Computer Engineering, SUT, Thailand. She received her B.S. in radiation techniques from Mahidol University, Thailand, in 1985, M.S. in computer science from the Prince of Songkla University, Thailand, in 1991 and Ph.D. in computer science from Nova Southeastern University, U.S.A., in 1999. Her research of interest includes data mining, logic and constraint programming.

Kittisak Kerdprasop is an associate professor at the School of Computer Engineering, SUT, and a chair of the School. He received his bachelor degree in mathematics from Srinakarinwirot University, Thailand, in 1986, MS in computer science from the Prince of Songkla University, Thailand, in 1991 and Ph.D. in computer science from Nova Southeastern University, U.S.A., in 1999. His current research includes machine learning and artificial intelligence.

Journal of Advances in Information Technology Vol. 12, No. 2, May 2021

A Study of Job Failure Prediction at Job Submit-State and Job Start-State in High-Performance Computing System: Using Decision Tree Algorithms

Anupong Banjongkan, Watthana Pongsena, Nittaya Kerdprasop, and Kittisak Kerdprasop
School of Computer Engineering, Suranaree University of Technology (SUT), Thailand
Email: banjongkan@gmail.com, watthana.p@sskru.ac.th, {nittaya, kerdpras}@sut.ac.th

***Abstract*—In High-Performance Computing (HPC) system, job failure is a major problem because it means the losses in computation time, resources, and power. Job failure also degrades significantly overall efficiency of the HPC system. In this paper, we propose two sets of models to predict job failure at two points of submission: job submit-state and job start-state. The models can be used as guiding tools for HPC-user to make efficient decision on managing their job submisison on the HPC system. The tools are thus for improving the efficiency of the HPC system at the job level. In the evaluation stage, we conduct a comparative study in order to compare performance of the job failure predictive models developed based on the decision-tree induction techniques including C5.0, Classification and Regression Tree (CART), and Chi-square Automatic Interaction Detector (CHAID). The datasets used for training and testing the models are the two workload logs collected from the HPC system at the National Electronics and Computer Technology Center (NECTEC), Thailand, and the Los Alamos National Laboratory (LANL), USA. To predict failure at the job submit-state and at the job start-state, the results show that the models built from C5.0 algorithm provide the highest accuracy of prediction (around 85% for the NECTEC dataset and 87% for the LANL dataset). The experimental results regarding prediction at different job states reveal that failure forecasting at the job start-state is slightly more accurate than making prediction at the job submit-state (accuracy improvement is around 1.45% for the NECTEC dataset and 0.46% for the LANL dataset). However, when considering both criteria of the performance of the models and the overhead of job waiting time, job failure prediction modeling at the job submit-state provides the best efficiency.**

***Index Terms*—decision tree, high-performance computing, job failure prediction, workload log**

I. Introduction

High-Performance Computing (HPC) is a computer system that combines many computers working together as a single system. Each of computer machines is referred to as a computing node. To achieve a single system characteristic, a group of computers communicate with each other via the internal network system. The HPC system has a special software named a scheduler to prioritize and manage the jobs in the queue to appropriately and efficiently be processed in the computing nodes. The HPC system uses central storage to guarantee correctness by making all computing nodes to access the same data. The two important aspects of most HPC systems are reliability and scalability.

To deliver reliable service, the HPC system operates with the redundancy principle such that it can resist the system failure. The second feature of HPC system, named scalability, refers to the ability to expand the system hardware to handle jobs that need extremely large number of computing nodes. The fundamental idea of combining many computers to work together as a single cluster makes the HPC system naturally effective in expanding its computing resources such that its computing power is unlimited. Based on these two important features, HPC systems are thus installed and operated in many computational laboratories worldwide.

Most of the jobs processed in the HPC system are related to advanced computational science and engineering tasks [1]. These jobs have common characteristics of high-memory consumption and complex calculation, such as jobs from the astronomical computing group, particle and high energy physics, the forecasting of climate, and so on. Most HPC systems are established as either the organization's computing resource center or open to public as a computing service with hourly service fees. Based on the vast areas of service, HPC systems must be able to support a variety of tasks. Variety issue also includes the different number of users, the variance of computational domains, and the variety of computational applications. To handle efficiently diverse tasks in the HPC systems, there are three levels of efficiency to be considered (as shown in Fig. 1).

The efficiency development at the system level is the maintenance of computing resources to be able to work or provide computational services at all time [2], [3]. At scheduling level, the HPC system must manage the submitted jobs as much as possible according to the full capacity of computing resources [4], [5]. The last one is at the job level which makes the jobs processed in the HPC

Manuscript received July 20, 2020; revised January 19, 2021.

doi: 10.12720/jait.12.2.84-92

system run through the finish-state with a high success rate [6], [7]. For the efficiency development of the HPC system at the system level and the scheduling level, it is the responsibility of the system administrator. Meanwhile, efficiency at the job level is the direct responsibility of users of the HPC system.

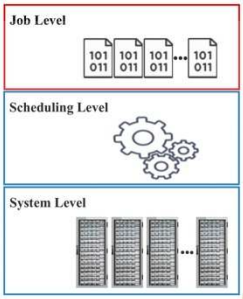


Figure 1. The level of efficiency development of the HPC system.

This research aims to improve the computing efficiency of the HPC system at the job level. Unsuccessful processing or job failure while processing in the HPC system is the most important problem for users because most jobs submitted to the HPC system is a large and complex job requiring long processing time and high usage of computing resources. If the job processing fails at running state, that means the waste of time, computing resources, and electrical power [8]. This research proposes a job failure predictive model while the job is processing at the job submit-state and the job start-state in the HPC system. The model can be further developed as a guiding tool for the HPC-users in a soft recommendation system or as a notifier in a monitoring system. We expect that such tool would be very useful for many HPC-users. For example, it is a decision-making helper for new users who lack experience of setting parameters while submitting a job to be processed in the HPC system, or it can issue notification when an error occurs by providing some helpful message for users to correctly handle the erroneous situation.

Machine Learning (ML) technique was used to create the job failure predictive model to predict a job failure while processing in the HPC system. We select three Decision Tree (DT) algorithms: C5.0, Classification and Regression Tree (CART), and Chi-square Automatic Interaction Detector (CHAID) to create the models based on the workload logs or job logs of the HPC systems. The first dataset is a workload log from the National Electronics and Computer Technology Center (NECTEC), Thailand. The second dataset belongs to the HPC system at the Los Alamos National Laboratory (LANL), USA, which is publicly available at the Parallel Workloads Archive [9].

The contributions of this research are as follows.

- This research proposes a framework to improve the computing efficiency of the HPC system at the job level by creating the model to predict job failure while the job is processed in the HPC system. The model can help users to submits a job to the HPC system in an effective way.
- We evaluate and compare the performance of several models to find the best one. The predictive tree-based models were created by using the algorithms C5.0, CART, and CHAID trained on the workload logs of the HPC systems.
- We perform a comparative study to assess model's performance at the job submit-state and the job start-state to find a suitable position to create the job failure predictive model.

The rest of this research is organized as follows. Section II presents a literature review. Section III describes theory and algorithms used in this research. Section IV explains details of the dataset, experimentation setting and results. Sections V and VI are discussion and conclusion, respectively.

II. Literature Review

The improve in computing efficiency of the HPC system can be done at 3 levels: the system level, the scheduling level, and the job level. Many research works propose the development of computing efficiency of the HPC system at the system level by preventing system from failure using analytical method to find patterns of system defects. Such patterns are in a form of a predictive model to forecast system failure in advance.

In 2006, Schroeder and Gibson [10] proposed the idea to investigate the root causes of the problem that occur on the HPC system with the objective to estimate the Mean Time Between Failure (MTBF) and Mean Time to Repair (MTTR). They used static tools to analyze the defect. The dataset used in their research was collected from 20 HPC systems at the Los Alamos National Laboratory, USA. The analysis results showed that the number of faults in the HPC system in one year was in a very wide range from 20 times to 1,000 times and defects mostly occurred in the large HPC system.

In 2016, Chuah *et al.* [11] also studied patterns of defects or system failure on the HPC system called Ranger Supercomputer. They introduced the CRUMEL (Correlating Resource Usage data and MEssage Logs) framework, which used data analysis principles based on correlation relationships between the system log and the workload dataset. The data from two sources were connected via timestamp. They reported the results that the CRUMEL tool could identify the pattern of defects. Their framework could also show the relationship of fault events on the system.

In 2007, Liang *et al.* [12] analyzed a system log of the IBM BlueGene/L system to predict system failures. The dataset was recorded in a period of 142 days. They tested and compared the efficiency of the models for predicting the system failure of the HPC system. The models were created by Rule-based Method, Support Vector Machine (SVM) and k-Nearest Neighbors (kNN). The results showed that the best performance in terms of time complexity and accuracy was from the model of the kNN algorithm.

In the same year, Gujrati *et al.* [13] also used Rule-based Method and statistical tools to create models to predict abnormal events in the IBM Blue Gene/L system. Their tools consisted of three processes: event processing, based prediction, and the Meta-learning prediction. The dataset was collected from the system log of two IBM Blue Gene/L systems. They reported that experimental results provided by the proposed tool could increase performance of system failures prediction by up to 3 times compared to their previous research.

In 2018, Soysal *et al.* [14] proposed a method to predict how long the job is to be processed in the HPC system (called wall-time). Their modeling method was based on the automated machine learning using 15 algorithms trained with the workload dataset of the HPC system that was collected from the Parallel Workloads Archive. The results showed that the models built from automated machine learning apprpach provided better performance than human predictions up to 7 times.

In 2012, Zhang *et al.* [15] proposed a descriptive analysis method using statistical tools and clustering algorithm to analyze the workload log data of the HPC system with the objective to find a suitable form of resource usage for the jobs to be processed in the HPC system. Their analysis scheme also employed information obtained from the experiment to create a tool for suggesting job submission to the system, called a knowledge-based recommendation system. The system is for users who lack experience in submitting job to the HPC system. Users were satisfied with the system that showed accuracy as high as 64.2%.

In 2012, Yuan *et al.* [16] studied the nature of a job that had not been successfully processed in the HPC system. They used statistical method to analyze the workload log dataset of the HPC system. The data were collected from 10 public datasets compiled from 8 HPC systems. The results showed that the unsuccessful jobs had much effect toward the overall efficiency of the HPC system in terms of service quality as well as wasted computation time.

From the literature review, we found that researchers focused on developing methods to improve the computing efficiency of the HPC system at two levels: the system level and the job level. The majority of research work concerned improvement the computing efficiency of the HPC system at the system level. Some researchers performed log analysis at the system level to find the root cause of the problems in the HPC system. Many recent works aimed at creating the predictive model to forecast the system failure of the HPC system. The predictive model was created by using statistical tools and machine learning techniques. There was some research work aiming at improving the computing efficiency of the HPC system at the job level by trying to find the appropriate form of requesting computing resources for the job to be processed in the HPC system. However, there are some existing limitations such as the model could not be applied to the real world HPC system because the model was built from the jobs at the finish-state and such state is not so useful for applying the model to the actual situation that job finish-state has not been reached yet. Therefore, we propose a new study framework by performing a comparative study of the job failure predictive model where jobs are at the submit-state and the start-state.

III. Background Theory and Algorithms

A. The Job State in the HPC System

The job state is the various statuses of job that has been processed through the HPC system (as shown in Fig. 2). There are three possible states; the first one is the state at which the job has been sent or submitted to the HPC system (called job submit-state), the second state is the state at which the job begins to be processed on the HPC system (called job start-state), and the last one is the state at which the job is processed completly and successfully (called job finish-state).

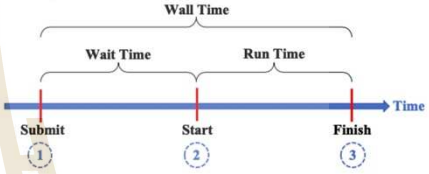


Figure 2. The three job states in the HPC system.

The time period from the job submit-state to the job start-state is called the wait-time. During this time, the job is waiting for its processing in the queue system. The next period from the job start-state to the job finish-stae, which is the end of job processing in the HPC system, is called the run-time or execute-time of the job. The whole duration of the job in the HPC system is called the wall-time (as in equation 1). It is a time frame from the point that users submitting their job until they receive the result.

$$\textit{Wall Time} = \textit{Wait Time} + \textit{Run Time} \quad (1)$$

We are interested in evaluating and comparing the performance of models to predict job failure while processing in the HPC system. We analysis job failure where the job is at the submit-state and the start-state with the main purpose of finding the most efficient model. The job failure predictive model can help the HPC-user making good decision on submitting job with the most efficient configuration of job processing in the HPC system. In some cases, if users know in advance at the submit-state that the job to be sent to the HPC system is likely to fail, they can avoid the useless computation and not to lose the job wall-time. Also, in the situation of the job that is in the start-state, the model can help users saving time for job processing (job run-time).

B. Decision Tree

Usually, the purpose of analyzing data from log files [17], [18] whether it is system log, network log, or workload log data, is to find the cause of problems (root cause) or to find defect that causes suboptimal performance of the system. Such analysis has be done regularly using the manual style that requires the

knowledge of experts to make decision about which part of data should be used, which analysis tool should be employed, and how to interpret the analysis results. So, it can be seen that it is an inefficient method of data analysis because it takes a long time for getting the correct result and it is limited by experience of the expert. Therefore, in this research we propose to use machine learning technique to create a model to predict job failure while processing in the HPC system. Machine learning technique is more efficient than manual analysis because it works in an automatic way; thus, limitations in time and expertise of the human analyst are eliminated.

This research adopts the decision tree algorithm as a modeling method to predict job failures while processing in the HPC system. The advantagees of a decision tree algorithm are that it has a simple work process, accuracy of the model is high, and the model is easy for interpretation, which is especially useful for root cause analysis. The decision tree model can classify or predict the target attribute with the logic-based concept, which is like the reasoning generally made by humans [19]. The structure of the decision tree is in the form of an upside-down tree (as illustrated in Fig. 3). The top node of the decision tree is the root node which is the node that has only the branching out lines. The node that has both the input output lines is called the internal node. The node at the end of the tree structure, in which there are only the input lines, is the leaf node. The leaf node is responsible for showing the final result of the classification or prediction of the decision tree model.

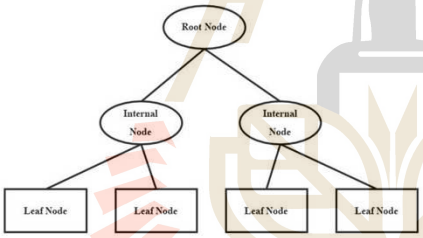


Figure 3. The decision tree structure.

This research builds the tree-based model to predict job failure while processing in the HPC system using three algorithms: C5.0, CART, and CHAID. The reasons for choosing theses three algorithms are due to their efficiency and successful application in many domains.

C5.0 algorithm [20], [21] was developed from the C4.5 algorithm. The main extension from the previous version is that C5.0 algorithm has some new features such as Boosting and Cost-sensitive tree. The C5.0 algorithm uses the node splitting criteria from calculating a value called Information Gain. One prominent advantage of C5.0 algorithm is its robustness against missing values in the dataset. The C5.0 algorithm only supports the categorical target variables. While, the CART algorithm [21], [22] is an outstanding algorithm that be able to support both data classification and data prediction as the target data can be either categorical or continuous. For classification, branching criteria for decision tree construction of the CART algorithm is the Gini index value, whereas in prediction, CART uses variance reduction value. The CHAID algorithm [23], [24] works like CART, but it has the advantage of being able to support branching of decision trees in more than two subgroups. It uses the statistical criteria based on the chi-square value for branching.

C. Assessment

This research generates job failure models based on three different algorithms. Performance of the obtained models are to be evaluated and compared using the four assessment matrices: accuracy, recall, precision, and F_1 score. The computation of each assessment metric can be done by observing values from a confusion matrix. Structure of confusion matrix is shown in Table I, while computation of various assesments are summarized in Table II.

TABLE I. THE CONFUSION MATRIX OF BINARY LABEL CLASSIFICATION

	Predict as Positive	Predict as Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

True Positive (TP) = The number of data that their actual class is "true" and the model predicts correctly as "true".

False Negative (FN) = The number of data that their actual class is "true", but the model predicted incorrectly as "false".

False Positive (FP) = The number of data that their actual class is "false", but the model predicted incorrectly as "true".

True Negative (TN) = The number of data that their actual class is "false" and the model predicts correctly as "false".

TABLE II. THE MODEL ASSESSMENT METRICS

No.	Metric Name	Formula
1	Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$
2	Recall	$\frac{TP}{TP+FN}$
3	Precision	$\frac{TP}{TP+FP}$
4	F_1 score	$2\times\frac{Recall\times Precision}{Recall+Precision}$

Accuracy is an assessment metric that considers the overall classification accuracy (for both "true" and "false" classes) of the model. It may not be a good metric is the dataset is imbalance in that number of data in one class significantly outnumbers data in other class. For such imbalance cases, we can use other assessment metrics such as recall (or sensitivity), precision, and F_1 score for a specific class of interest (which is normally called a positive class). The recall is used to evaluate the model performance from a perspective of the power to predict correctly as much as possible the data from the class of interest. While precision is the assessment of the model

from the aspect of correctness that a good model should not incorrectly predict data in negative class (those that are out of interest) to be a positive class. The F_1 score is the harmonic mean of the values from recall and precision. The range of these values are from 0 to 1. The value 1 is the most desirable measurement.

IV. Experimentation and Results

A. Dataset

The data used in this research are the workload log or job log of the HPC system. It is the result of recording the activity related to the jobs processed on the HPC system, which are recorded by the scheduler. The dataset used in this research is the HPC-workload logs from two HPC systems. The first one is the public dataset obtained from the Parallel Workloads Archive which is the workload log of the large cluster computers of the Los Alamos National Laboratory (LANL) in the United States. The cluster computers consist of the Origin 2000 computers, a total of 2,048 nodes and use the LSF Scheduler. The dataset of LANL contains the data from December 1999 to April 2000, with a data size of 122,233 elements. The second dataset is a workload log of a small cluster computer that has approximately 500 processing units of the X86_64 computer. This cluster computer belongs to the National Electronic and Computer Technology Center (NECTEC) of Thailand. This dataset was recorded by PBS/Torque scheduler with a total data size of 87,046 elements.

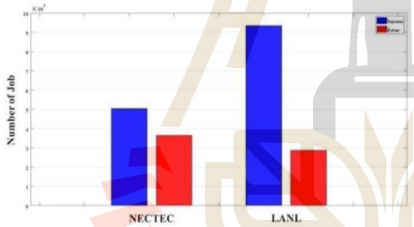


Figure 4. Histogram of the job success (blue) and error (red) in the dataset.

Fig. 4 shows the proportion between number of job completed with success and those completed with error status in the two datasets. The job success per job failure of the LANL dataset is around 76.5% to 23.5%. This dataset shows imbalance between the majority class of job success and the minority class of job failure. The NECTEC dataset has proportion between job success and job failure at 58% to 42%. The amount of data in the two classes are quite balance.

The first step prior to the modeling process is the attribute selection. This step corresponds to the main aim of this research to model HPC log at the job submit-state and job start-state. For creating a model to predict success/failure of a job at the job submit-state, we selcet three attributes to be used as predictors to predict the target value of job status as either success or failure. The three predictor attributes are "User ID", "CPU Request" and "Queue Type". This selection is based on the real situation that they are basic information that users need to specify before submitting the job into the HPC system. To create a model to predict at the job start-state the final job status, we additional attribute from the workload log regarding a job being processed on the HPC system. Therefore, on modeling at the job start-state, four attributes are used as predictors to predict success/failure of a job. These predictor attributes are "User ID", "CPU Request", "Queue Type" and "Wait Time". The target attribute for both job at submit-state and job at start-state modeling is "Finished Status". Summary of data attributes is presented in Table III.

The meaning of each attribute is as follows. "User ID" is the unique id of HPC-user. "CPU Request" is the number of processor elements that the user requires to use for the job. "Queue Type" is the queue system in the HPC system, which relates to the limitation of job run time. "Wait Time" is the period of a job waiting in the queue. "Finished Status" is the job status at the job finish-state.

TABLE III. The Attributes Used in This Research

No.	Attribute at Job Submit-state	Attribute at Job Start-state	Data Type	Attribute Type
1	User ID	User ID	Nominal	Predictor
2	CPU Request	CPU Request	Numeric	Predictor
3	Queue Type	Queue Type	Nominal	Predictor
4	n/a	Wait Time	Numeric	Predictor
5	Finished Status	Finished Status	Binary	Target

B. Experimentation

The experimentation steps in this research are shown in Fig. 5. The data collection is the procedure for collecting the HPC-workload logs from data sources. After that, it is the data pre-processing step. At this step, the data cleaning was performed on the HPC-workload log from NECTEC. Data cleaning is unnecessary for the LANL dataset as it is already in the Standard Workload Format (SWF). After that, we split randomly the cleased data into three sub-datasets of each HPC-workload log. Each sub-dataset consists of 1,000 elements with five attributes: "User ID", "CPU Request", "Queue Type", "Wait Time", and "Finished Status". This research sets the target attribute to be "Finished Status". There are two distinct values in the target attribute representing the job status (either success or error) after the completion of HPC processing. The reason to use different sub-dataset to create the model because we would like to observe the model robustness when the dataset is changed.

The next steps are modeling and evaluating performance of each model on predicting final status of the job while it is at the processing stage in the HPC system. The three decision tree learning algorithms (C5.0, CART, and CHAID) are applied to create the job failure predictive model. In this experiment, we use 70% of the dataset to create the model, while the remaining 30% is for testing the model's performance. The last step of our experimentation is the part of model evaluation and comparison. The total scenarioes to test the model are 32 cases (2 workload log x 3 sub-datasets x 3 DT algorithms x 2 states of the job).

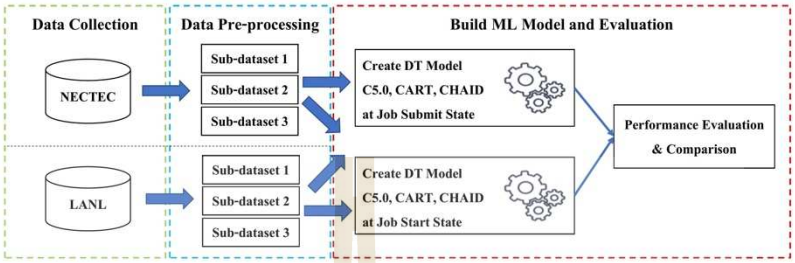
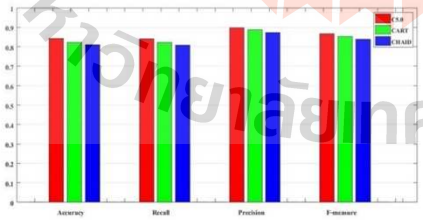


Figure 5. The research workflow.

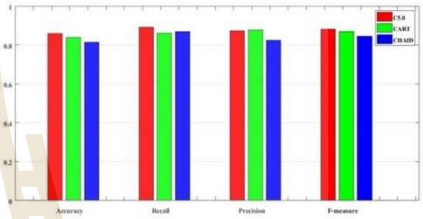
C. Results

The modeling evaluation results for the models built from the NECTEC dataset where the jobs are at the submit-state are as follows. The results of the job failure predictive model of C5.0 algorithm show the average accuracy, average recall, average precision, and average F_1 score at 0.8435, 0.8399, 0.8984, and 0.8670, respectively, with the variance of the F_1 score at 0.65e-3. The results of the job failure predictive model of the CART algorithm show the average accuracy, average recall, average precision, and average F_1 score at 0.8234, 0.8218, 0.8870, and 0.8526, respectively, with the variance of the F_1 score at 0.26e-3. Lastly, the results of the job failure predictive model of the CHAID algorithm show the average accuracy, average recall, average precision, and average F_1 score at 0.8100, 0.8078, 0.8728, and 0.8388, respectively, with the variance of the F_1 score at 0.011e-3.

The modeling results for the case of building predictive models with the NECTEC dataset while the jobs are at the start state are as follows, The results of the job failure predictive model of C5.0 algorithm show the average accuracy, average recall, average precision, and average F_1 score at 0.8580, 0.8924, 0.8734, and 0.8817, respectively, with the variance of the F_1 score at 0.33e-3. Next, the results of the job failure predictive model of the CART algorithm show the average accuracy, average recall, average precision, and average F_1 score at 0.8381, 0.8604, 0.8771, and 0.8686, respectively, with the variance of the F_1 score at 0.25e-3. Lastly, the results of the job failure predictive model of the CHAID algorithm show the average accuracy, average recall, average precision, and average F_1 score at 0.8153, 0.8697, 0.8237, and 0.8446, respectively, with the variance of the F_1 score at 0.88e-3.



(a) job submit-state



(b) job start-state

Figure 6. Performance of models built from the NECTEC dataset at different HPC processing stages.

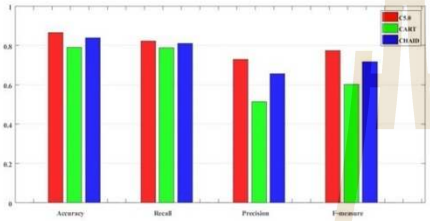
The performances of models built from the C5.0, CART, and CHAID algorithms using the NECTEC dataset are demonstrated in Fig. 6. The models built from jobs at submit-state and jobs at start-state are shown in the upper and lower graphs, respectively.

The results of the experiment using the LANL dataset at the job submit state show that the job failure predictive model of C5.0 algorithm having the average accuracy, average recall, average precision, and average F_1 score at 0.8654, 0.8230, 0.7298, and 0.7734, respectively, with the variance of the F_1 score at 0.69e-3. The results of the job failure predictive model of the CART algorithm show the average accuracy, average recall, average precision, and average F_1 score at 0.7893, 0.7876, 0.5151, and 0.6010, respectively, with the variance of the F_1 score at0.061e-3. Lastly, the results of the job failure predictive model of the CHAID algorithm show the average accuracy, average recall, average precision, and average F_1 score at 0.8390, 0.8096, 0.6551, and 0.7166, respectively, with the variance of the F_1 score at 0.022e-3.

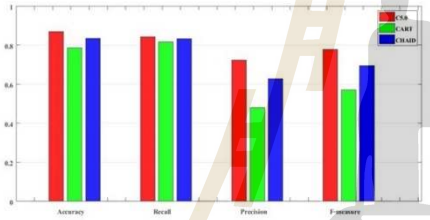
The results of the experiment in the case of LANL dataset at the job start state are as follows. The job failure predictive model of C5.0 algorithm shows the average accuracy, average recall, average precision, and average F_1 score at 0.8700, 0.8422, 0.7227, and 0.7777, respectively, with the variance of the F1 score at 0.87e-3. The results of the job failure predictive model of the CART algorithm show the average accuracy, average recall, average precision, and average F_1 score at 0.7861, 0.8171, 0.4796, and 0.5712, respectively, with the variance of the F_1 score at 0.0189. Lastly, the results of the job failure predictive

model of the CHAID algorithm show the average accuracy, average recall, average precision, and average F_1 score at 0.8345, 0.8332, 0.6282, and 0.6955, respectively, with the variance of the F_1 score at 0.094e-3.

The performance comparison of the predictive models of C5.0, CART, CHAID algorithms built from the LANL dataset are shown in Fig. 7. The upper graph correspomds to models built from jobs at submit-state, whereas the lower graph shows performance of models built from jobs at start-state.



(a) job submit-state



(b) job start-state

Figure 7. Performance of models built from the LANL dataset at different HPC processing stages.

V. Discussion

The overall performance of all the predictive models to forecast job failure when processing in the HPC system gives the average accuracy of the model not less than 78%. The performance of the predictive model of the C5.0 algorithm is better than the predictive model of other algorithms in almost every test case. The C5.0 achieves the best average accuracy of 85.8% and 87% for the NECTEC and the LANL datasets, respectively. While, the predictive model of CART shows the lowest performance with 83.81% and 78.93% of average accuracy for the NECTEC dataset and the LANL dataset, respectively.

The models to predict job failure while processing in the HPC system built from decision tree algorithms can give high accuracy in every test case of the two HPC-workload datasets. But, the precision value of the model is quite low in the test case of the LANL dataset. The reason for this is that the LANL dataset has a high imbalance ratio between majority and minority classes. The low accuracy is from the predictive models built with the DT algorithms being unable to predict correctly the minority class.

The results regarding the performances of the predictive models at two different job states are the case of NECTEC workload log (Table IV) and LANL workload log (Table V) show that the predictive model being built while jobs are at start-state can predict the job failure at the end of the HPC processing slightly better than the model built from the jobs at job submit-state. This is because at the start-state there exist more useful information than while the jobs are at the submit-state. However, the predictive model to job failure prediction in the HPC system at job submit-state shows the best efficiency when considering the performance of the model together with the overhead of job waiting time. In both datasets, there are jobs waiting in the queue of the HPC system more than 80%.

TABLE IV. Performance of the Best Predictive Model (C5.0) Built from NECTEC Workload-Log at Different Stages: Job Submit-State vs Job Start-State

NECTEC	Accuracy (avg)	Recall (avg)	Precision (avg)	F_1 Score (avg)
Submit-state	0.8435	0.8339	0.8984	0.8670
Start-state	0.8580	0.8924	0.8771	0.8817

TABLE V. Performance of the Best Predictive Model (C5.0) Built from LANL Workload-Log at Different Stages: Job Submit-State vs Job Start-State

LANL	Accuracy (avg)	Recall (avg)	Precision (avg)	F_1 Score (avg)
Submit-state	0.8654	0.8230	0.7298	0.7734
Start-state	0.8700	0.8422	0.7227	0.7777

VI. Conclusion

This research proposed the predictive modeling framework to create a model for forecasting job failure while processing in the HPC system. The model was created through the machine learning technique using three decision tree algorithms: C5.0, CART, and CHAID. The obtained model can be applied to help users make an efficient justification while their job is running in the HPC system. The datasets used in this work are the two HPC-workload logs that were collected from the operation of the HPC systems at NECTEC, Thailand, and LANL, USA. The results of the comparative study regarding the model performance show that the job failure predictive model built by the C5.0 algorithm has the best performance with an average accuracy of 85.8% and 87% using NECTEC dataset and LANL dataset, respectively. The C5.0 model also has robustness on data instability.

In the part of the comparative study when models are built at different job states, we found that performance of the model built from jobs while those jobs were at the start-state shows better accuracy than the model built from jobs running at the job submit-state. This finding is agree upon the two datasets. However, when considering the model accuracy with the tradeoff regarding the overhead of waiting time, we suggest that the position of the job at submit-state is more suitable to be applied for creating the job failure predictive model than the position of the job at start-state.

Conflict of Interest

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

The first author is responsible for designing the research framework, organizing the experimentation steps and preparing the draft manuscript. The second author helps correcting the draft manuscript. The third author helps editing manuscript to be as appeared in the final version. The last author takes part in the conceptual design, experimentation setup and confirming correctness of the results.

ACKNOWLEDGMENT

The authors would like to acknowledge the "National e-Science Infrastructure Consortium" of NECTEC for providing the HPC-workload log as a dataset that we use in this research (URL: http://www.escience.in.th). The first author has been supported by a scholarship from the Suranaree University of Technology (SUT). The second author has been supported by a scholarship from the Ministry of Science and Technology, Thailand. The third and fourth authors are researchers of the Data and Knowledge Engineering Research Unit, which has been fully supported by a research grant from SUT.

REFERENCES

[1] P. Uthayopas, T. Angskun, and J. Maneesilp, "Building a parallel computer from cheap PCs: SMILE cluster experiences," in *Proc. the Second Annual National Symposium on Computational Science and Engineering*, 1998, p. 10.

[2] T. Pitakrat, D. Okanović, A. V. Hoorn, and L. Grunske, "Hora: Architecture-Aware online failure prediction," *Journal of System and Software*, vol. 137, pp. 669-685, Mar. 2018.

[3] B. Mohammed, I. Awan, H. Ugail, and M. Younas, "Failure prediction using machine learning in a virtualised HPC system and application," *Cluster Comput.*, vol. 22, no. 2, pp. 471-485, Jun. 2019.

[4] M. Stillwell, F. Vivien, and H. Casanova, "Dynamic fractional resource scheduling versus batch scheduling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 3, pp. 521-529, Mar. 2012.

[5] A. Reuther, *et al.*, "Scalable system scheduling for HPC and big data," *Journal of Parallel and Distributed Computing*, vol. 111, pp. 76-92, Jan. 2018.

[6] R. L. F. Cunha, E. R. Rodrigues, L. P. Tizzei, and M. A. S. Netto, "Job placement advisor based on turnaround predictions for HPC hybrid clouds," *Future Generation Computer System*, vol. 67, pp. 35-46, Feb. 2017.

[7] B. Silva, M. A. S. Netto, and R. L. F. Cunha, "JobPruner: A machine learning assistant for exploring parameter spaces in HPC applications," *Future Generation Computer System*, vol. 83, pp. 144-157, Jun. 2018.

[8] J. Kunkel and M. F. Dolz, "Understanding hardware and software metrics with respect to power consumption," *Sustainable Computing: Informatics and System*, vol. 17, pp. 43-54, Mar. 2018.

[9] D. G. Feitelson, D. Tsafrir, and D. Krakov, "Experience with using the parallel workloads archive," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2967-2982, Oct. 2014.

[10] B. Schroeder and G. A. Gibson, "A large-scale study of failures in high-performance computing system," in *Proc. International Conference on Dependable Systems and Networks*, 2006, p. 10.

[11] E. Chuah, A. Jhumka, J. C. Browne, N. Gurumdimma, S. Narasimhamurthy, and B. Barth, "Using message logs and resource use data for cluster failure diagnosis," in *Proc. IEEE 23rd International Conference on High Performance Computing*, Hyderabad, India, 2016, pp. 232-241.

[12] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo, "Failure prediction in IBM BlueGene/L event logs," in *Proc. Seventh IEEE International Conference on Data Mining*, Omaha, NE, USA, 2007, pp. 583-588.

[13] P. Gujrati, Y. Li, Z. Lan, R. Thakur, and J. White, "A Meta-learning failure predictor for blue Gene/L system," in *Proc. International Conference on Parallel Processing*, Xian, China, 2007.

[14] M. Soysal, M. Berghoff, and A. Streit, "Analysis of job metadata for enhanced wall time prediction," *Job Scheduling Strategies for Parallel Processing*, vol. 11332, 2018.

[15] H. Zhang, H. You, B. Hadri, and M. Fahey, "HPC usage behavior analysis and performance estimation with machine learning techniques," in *Proc. 18th International Conference on Parallel and Distributed Processing Techniques and Applications*, 2012, p. 7.

[16] Y. Yuan, Y. Wu, Q. Wang, G. Yang, and W. Zheng, "Job failures in high performance computing system: A large-scale empirical study," *Computers & Mathematics with Applications*, vol. 63, no. 2, pp. 365-377, Jan. 2012.

[17] S. Sabato, E. Yom-Tov, A. Tsherniak, and S. Rosset, "Analyzing system logs: A new view of what's important," in *Proc. the 2nd USENIX Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*, 2007, p. 7.

[18] A. Oliner, A. Ganapathi, and W. Xu, "Advances and challenges in log analysis," *Commun. ACM*, vol. 55, no. 2, p. 55, Feb. 2012.

[19] A. Trabelsi, Z. Elouedi, and E. Lefevre, "Decision tree classifiers for evidential attribute values and class labels," *Fuzzy Sets and System*, vol. 366, pp. 46-62, Jul. 2019.

[20] S. Pang and J. Gong, "C5.0 classification algorithm and application on individual credit evaluation of banks," *System Engineering - Theory & Practice*, vol. 29, no. 12, pp. 94-104, Dec. 2009.

[21] M. Hassoon, M. S. Kouhi, M. Zomorodi-Moghadam, and M. Abdar, "Rule optimization of boosted C5.0 classification using Genetic algorithm for liver disease prediction," in *Proc. International Conference on Computer and Applications*, Doha, United Arab Emirates, 2017, pp. 299-305.

[22] S. Abdul Kareem, S. Raviraja, N. A Awadh, A. Kamaruzaman, and A. Kajindran, "Classification and regression tree in prediction of survival of AIDS patients," *MJCS*, vol. 23, no. 3, pp. 153-165, Dec. 2010.

[23] W. A. V. Clark, M. C. Deurloo, and F. M. Dieleman, "Modeling categorical data with chi square automatic interaction detection and correspondence analysis," *Geographical Analysis*, vol. 23, no. 4, pp. 332-345, Sep. 2010.

[24] M. Ramaswami and R. Bhaskaran, "A CHAID based performance prediction model in educational data mining," *International Journal of Computer Science Issues*, vol. 7, no. 1, p. 9, 2010.

Anupong Banjongkan is a Ph.D. student in a computer engineering program with the School of computer engineering, Suranaree University of Technology, Thailand. He graduated in B.S. of Computer Science and Master of Engineering in Electrical Engineering at King Mongkut's University of Technology North Bangkok, Thailand, in 2007 and 2011, respectively. His current research of interest includes High Performance Computing, Machine Learning, and Knowledge Discovery.

Watthana Pongsana is a Ph.D. student, School of Computer Engineering, Suranaree University of Technology (SUT), Thailand. He received his B.E. and M.E. in computer engineering from Suranaree University of Technology, Thailand, in 2008 and 2012. His research of interest includes Software Engineering, Data Mining, Artificial Intelligence, and Human-Computer Interaction.

Nittaya Kerdprasop is an associate professor and the head of the Data Engineering Research Unit, School of Computer Engineering, SUT, Thailand. She received her B.S. in radiation techniques from Mahidol University, Thailand, in 1985, M.S. in computer science from the Prince of Songkla University, Thailand, in 1991 and Ph.D. in computer science from Nova Southeastern University, U.S.A., in 1999. Her research of interest includes Data Mining, Logic and Constraint Programming.

Kittisak Kerdprasop is an associate professor at the School of Computer Engineering, SUT, and a Chair of the School. He received his bachelor's degree in Mathematics from Srinakarinwirot University, Thailand, in 1986, MS in computer science from the Prince of Songkla University, Thailand, in 1991 and Ph.D. in computer science from Nova Southeastern University, U.S.A., in 1999. His current research includes Machine Learning and Artificial Intelligence.

$PM_{2.5}$ Forecasting Model based on Linear and Non-linear Hybrid Algorithm

Anupong Banjongkan
School of Computer Engineering
Suranaree University of Technology, Thailand
Email: banjongkan@gmail.com

Nittaya Kerdprasop
School of Computer Engineering
Suranaree University of Technology, Thailand
Email: nittaya@sut.ac.th

Anusara Hirunyawanakul
Data Science and Computation
Faculty of Science, Energy, and Environment
King Mongkut's University of Technology North Bangkok, Rayong Campus, Thailand
Email: anusara.h@sciee.kmutnb.ac.th

Kittisak Kerdprasop
School of Computer Engineering
Suranaree University of Technology, Thailand
Email: kerdpras@sut.ac.th

2023 15th International Conference on Knowledge and Smart Technology (KST) | 978-1-6654-7712-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/KST57286.2023.10086907

Abstract—**Air pollution is one of the harmful problems that the world has focused on and needs to be solved urgently because air pollution has a direct impact on humans leading to premature death caused by various diseases such as asthma inflammatory respiratory disease, lung cancer, and so on. The air pollutants, especially tiny particulate matter (PM), are currently receiving attention because they are a major problem in many large and populated cities around the world. This paper proposed a time-series model for forecasting $PM_{2.5}$ in advance through a machine learning process with a linear and non-linear hybrid algorithm. A hybrid algorithm that brings together the capabilities of autoregressive integrated moving average (ARIMA) and the adaptive-neuro fuzzy inference system (ANFIS) is used to find the linear and nonlinear correlation of the $PM_{2.5}$ time-series data. The proposed model is called ARIMA-FIS which uses the gradient descent (GD) method in the learning process. The dataset used in this research is the daily recorded of $PM_{2.5}$ values in Rayong province, which is the industrial city in Thailand. The results showed that the ARIMA-FIS model had the best performance in forecasting $PM_{2.5}$ in advance with the least error at 3.46 of mean absolute error (MAE) and 5.11 of root mean square error (RMSE). The proposed model gave the percentage of RMSE almost 3% better than the other standard time-series models.**

Keywords- ANFIS; Hybrid Model; $PM_{2.5}$; Time-series.

I. INTRODUCTION

Air pollution is one of the environmental problems that the world pays attention, because air pollution has a direct impact on public health. It also affects the development and expansion of urban that has great impact on the country's economy. The report from the World Health Organization (WHO) in 2016 sated that 4.2 million people around the world died prematurely from the effects of air pollution [1]. Thailand is ranked 45th among the countries with the worst air quality [2]. It is estimated that in Thailand there will be approximately 50,000 premature deaths from air pollution per year [3]. Air pollution is estimated to be the cause of lung diseases for as much as 43% and the major cause of stroke for almost 24%. [4]. There are five types of air pollutants that the WHO has focused on including particulate matter (PM), ozone (O_3), nitrogen dioxide (NO_2), carbon monoxide (CO), and sulfur dioxide (SO_2). The air quality levels, or air quality index (AQI) are measured from the values of these five air pollutants [5], The severity of air quality is represented by color, i.e., air quality is in the blue and green range are 0-25 and 26-50, respectively. The blue and green ranges are the air quality that is safe for living things. The air quality in yellow range is 51-90, which indicates that air quality is bad and beginning to affect living things. The air quality in the range 101-200 and more than 200, which are represented by orange and red color, respectively, are the poor air quality not suitable for living.

This paper focuses on air pollution, especially $PM_{2.5}$, which is a problem that Thailand and countries with large cities around the world are needed to fix urgently. The ability to analyze data in real time and the ability to forecast $PM_{2.5}$ values in advance with accuracy is essential to combating air pollution problems. This research presents a tool for forecast $PM_{2.5}$ in advance, with a mathematical model developed through a machine learning (ML) process with a hybrid algorithm that able to capture the linear and non-linear correlation of time-series data [6]. The correlation components of the time-series data are shown as in equation (1).

$$y_t = (y_{linear} + y_{non\text{-}linear}) + \text{White Noise} \quad (1)$$

Where y_t is the time-series data, y_{linear} and $y_{non\text{-}linear}$ represent the linear and non-linear correlation patterns of the time-series data respectively. The white noise is non-pattern correlation of time-series data.

Over the past decade, the topic of research related to air quality forecasting, especially the forecast of $PM_{2.5}$, has drawn a lot of attention from many researchers. The development of tools for forecasting air quality with machine learning and deep learning techniques contribute many interesting research work as follows.

C.-J. Huang and P.-H. Kuo [7] presents a model called APNet, a model developed with the deep learning technique. The deep learning network that used in their research are convolutional neural network (CNN) and LSTM. They are used to model $PM_{2.5}$ data in Beijing city, China. The proposed model (APNet) got high performance with forecasting $PM_{2.5}$ in an hour ahead when compared with other baseline models. R. Wongsathan [8] presented a time-series model to forecast PM_{10} in Chiang Mai, Thailand. He used machine learning technique with ANFIS algorithm to build the forecasting model. The hotspots and meteorology data (wind speed, temperature, pressure) were used in his research. The result shown that the proposed model has the best performance in comparison with the other existing models at MAE and RMSE 5.8 and 7.3, respectively. S. Du et al. [9] presented a model to forecast $PM_{2.5}$ using deep learning with a combination of algorithms including one-dimensional CNN (1D-CNNs) and bi-directional LSTM (Bi-LSTM) which are called the deep air quality forecasting framework (DAQFF). The DAQFF can be used for single-step and multi-step forecasting. The data used in their research are Beijing air quality dataset from UCI machine learning repository and Urban air quality dataset from the Urban Air Project of Microsoft research. The results showed that the DAQFF model achieved the best performance in both single-step and multi-step for $PM_{2.5}$ forecasting of the two datasets when compare with standard model (SVR, LSTM, GRU, RNN, and CNN).

J. Amanollahi and S. Ausati [10] studied and developed linear, non-linear and hybrid models for forecasting the $PM_{2.5}$ in Tehran city, Iran. The meteorological data were used in their research. The results showed that the best performance model was the ANFIS model, which had the efficiency of the train process with the measurement of R^2, RMSE, and MAE at 0.99, 0.47, and 0.1305, respectively. The efficiency of the test process showed R^2, RMSE, and MAE at 0.82, 3.29, and 2.16, respectively. S. Jeya and L. Sankari [11] developed $PM_{2.5}$ forecasting model using Bi-LSTM algorithm. The dataset used in their research is $PM_{2.5}$ data from ground base station and the meteorological data of Beijing, China. The data are collected from UCI repository. They selected the data in duration of 1 January 2010 to 31 December 2014 for building the model. The Bi-LSTM model achieved high performance with RMSE of 9.86, MAE of 7.53, and MAPE of 0.16, which is the efficiency of one hour ahead of $PM_{2.5}$ forecasting.

C. Guo, et al. [12] proposed the forecasting model to predict $PM_{2.5}$ in an hour ahead. Their research used the ensemble technique with RNN, LSTM, and GRU algorithms. They extracted the important predictor variables with the Pearson correlation coefficient. The dataset is the $PM_{2.5}$ of three ground base stations with the meteorological data of Shanghai city, China. The result shown that the wind direction is most effect to the $PM_{2.5}$ as well as the proposed model outperform other baseline models with MAE of 6.19 and MAPE of 16.20%. H. Xie, et al. [13] proposed the $PM_{2.5}$ forecasting model of Wuxi city, China. The proposed model was developed from the deep learning with two algorithms, CNN to extract the feature of data and GRU for predictive model. Their research used AQI and meteorological data. The results showed that the model CNN-GRU yield the best performance at accuracy (computed from 100 - MAPE) 76.90% and 70.05% when tested in January and May dataset, respectively. R. Janarthanan, et al. [14] developed $PM_{2.5}$ forecasting model with combination algorithm using Support Vector Regression (SVR) and LSTM. At the data pre-processing step, the Grey Level Co-occurrence Matrix (GLCM) algorithm was used to prepare the dataset. The dataset is the air quality data collected from National Air Monitoring Program (NAMP), India. The results showed that the proposed model performed the best in comparison with previous studies, with RMSE of 10.9.

E. Isaev, et al. [15] presented a machine learning model for forecasting air pollution in Bishkek city, Kyrgyzstan at the time scale of one hour in advance. A comparative study of model performance using random forest (RF), extreme gradient boost (XGboost), artificial neuron network (ANN), K-nearest neighbors (KNN), decision tree (DT), lasso regression (LaR), and linear regression (LR). The data used in their research were air quality and meteorological data collected from the Kyrgyzhydromet. The results showed that the air pollution forecasting model developed with the RF algorithm outperform other models at RMSE of 28.

The main objectives of this research are the same as those mentioned above. We want to develop a highly efficient time-series model for forecasting air quality in advance. This research focuses on the $PM_{2.5}$. It is like [8] and [10], who developed a model based on the ANFIS algorithm. However, this research presents a model for forecasting $PM_{2.5}$ in advance with different methods and datasets. We apply [9] in the data preparation concept to our research. This research developed a time-series model for forecasting $PM_{2.5}$ through a machine learning with a hybrid algorithm. We combine the advantage of the ARIMA statistical time-series model with the Hybrid non-linear ANFIS algorithm, which is able to capture the correlation of linear and nonlinear time-series data. This research uses $PM_{2.5}$ from Rayong province, Thailand as data for modeling and comparative study of efficiency with other standard time-series models. This research has three main contributions:

- Proposing a highly efficient time-series model with new techniques. It uses a machine learning with a hybrid algorithm between the ARIMA statistical time-series model and the non-linear ANFIS algorithm.
- Appling the temporal and spatial average value (TSA) method to fill in the missing data in the dataset.
- Performing comparative study of the proposed time-series model with the standard time-series models including ARIMA, ANFIS, and LSTM.

The rest of the paper is organized as follows. Section II describes the ARIMA statistical time-series model and the ANFIS algorithm. Section III outlines the data preparation and research process. Section IV is the results of the research experimentation. The last section discusses the conclusion of the research.

II. METHODOLOGY

A. Autoregressive Integrated Moving Average

The Autoregressive Integrated Moving Average (ARIMA) is a statistical model that combines three processes [16], [17]. The first process is auto regressive or AR, which is a model that uses linear equations to forecast future values. The AR model uses relation of the data in time-series as called lag time to build the forecasting model. The equation of AR looks like a multiple regression (MR) model but the AR model is based on the lag time of the data, where the amount of lag time is determined by the partial auto correlation function (PACF) graph. The second part of ARIMA is a moving average (MA). The MA is a linear equation model that predicts future values using past data. It is the same as the AR model, but the MA model considers a linear correlation through individual time intervals. The number of terms is determined by the auto correlation function (ACF) dependent on the current and past error terms. So, when AR and MA are combined, the equation (2) is the equation of the ARMA model.

$$y_t = \sum_{i=1}^{p} \alpha_i y_{t-i} + \sum_{i=1}^{q} \beta_i \varepsilon_{t-i} + c + \varepsilon_t \qquad (2)$$

Where, y_t is a time-series data at time t. ε_{t-i} is error terms at time i, i is in the range of 1 to q. β_i is the coefficient at the time i, i is ranging from 1 to q. c is a constant value, and ε_t is a white noise of time-series data.

The last part of ARIMA is Integrated, represent by i. This part transforms the time-series data to stationary via the differencing time-series process or differ. The equation (3) shows the example of the single order differ process. The stationary means that this time-series data must have no trend, no seasonal, has a relatively constant level of graph and has a constant variance. It can be said that the future data must be similar to the data in the past, in terms of probability.

$$y'_t = y_t - y_{t-1} \qquad (3)$$

Where, y'_t is the transformed data at time t, y_t is the time-series data at time t, and y_{t-1} is the time-series data at one lag time.

B. Adaptive Nero-Fuzzy Inference System

Adaptive Nero-Fuzzy Inference System (ANFIS) is a model that takes the advantages of the fuzzy inference system (FIS) with the artificial neural network (ANN), which combines the benefit of fuzzy reasoning with capacity of learning data. The person who proposed this technique was Jyh-Shing Roger Jang in 1933 [18]. ANFIS prefers to use of FIS type-3, which is the Takagi and Sugeno's fuzzy reasoning by the fuzzy If-Then rules. For example, given that the inputs are x and y and output is z, the fuzzy If-Then rules can be shown below.

Rule 1: If x is A_1 and y is B_1, Then $f_1 = p_1x + q_1y + r_1$,

Rule 2: If x is A_2 and y is B_2, Then $f_2 = p_2x + q_2y + r_2$.

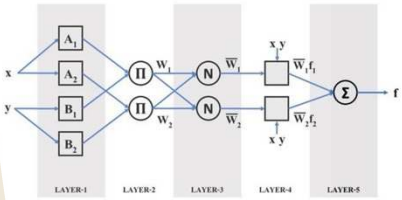


Figure 1. The ANFIS structure.

The structure of ANFIS has five main layers as shown in Figure 1. Each layer performs the following functions: the first layer is the layer that interacts with member functions. It compares the input values with member functions to transform them into linguistic values, and then combines them into the node weights in the next layer of the network. The second layer consists of nodes to combine the weight values. It uses multiplication to obtain the total weight. The third layer is used to normalize the node's weight. The fourth layer is a layer that deals with the consequent part of the fuzzy rules. The last is the fifth layer, which has only one node. This layer is responsible for calculating the sum of each output in the fourth layer. Then, the output in this layer is a result of ANFIS.

C. Assessment

To evaluate the efficiency of a time-series model for forecasting $PM_{2.5}$ in advance, this research selects three evaluation metrics including root mean square error (RMSE), mean absolute error (MAE), and percentage of root mean square error (%RMSE) to evaluate the performance of the models. The equations of these metrics are shown in Table I.

TABLE I. MODEL PERFORMANCE EVALUATION METRIC.

Matric	Equation	Note
MAE	$\frac{1}{n}\sum_{i=1}^{n}\lvert x_i - x'_i\rvert$	x is observed value. x' is predicted value. n is the number of sample data. $\bar{x}$ is an average value of sample data.
RMSE	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - x'_i)^2}$	
%RMSE	$\frac{100}{\bar{x}}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - x'_i)^2}$	

III. Data and Experimentation

A. Dataset

This research used the $PM_{2.5}$ data from ground base air quality measurement stations. It locates in Rayong province, Thailand. There are four stations (28T, 29T, 30T and 31T) in the area. The $PM_{2.5}$ data were prepared and provided by the Pollution Control Department, Ministry of Natural Resources and Environment of Thailand.

At the preliminary exploratory step, we found that the $PM_{2.5}$ in Rayong province were high during late January until early May. This is shown in Figure 2, which is the data from the 30T ground base air quality measurement station.

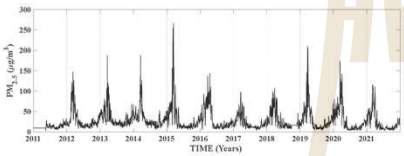


Figure 2. The $PM_{2.5}$ in Rayong province at station 30T(mid-2011 to 2021).

This research selected the $PM_{2.5}$ data from 30T ground base air quality measurement station as a base dataset. We used the data in duration of 2012 to 2021 because they are the most complete. The temporal and spatial average value (TSA) data imputation technique [19] was used to handle the missing values. As shown in the equations (4) and (5), the spatial average value is used when the data is missing more than one period in a row, otherwise the temporal average value is to be considered.

$$y_{temp} = \frac{1}{d}\sum_{i=1}^{d} y_{t-i} \quad (4)$$

$$y_{spat} = \frac{1}{s}\sum_{j=1}^{s} y_j \quad (5)$$

Where, y_{temp} and y_{spat} are the $PM_{2.5}$ values that are imputed with the temporal average value and spatial average value method, respectively. y_{t-i} is the $PM_{2.5}$ value at lag time i, y_j is the $PM_{2.5}$ value at ground base air quality measurement station j. d is the number of day's lag time as well as s is the number of ground base air quality measurement stations that are to be accounted.

We compare the efficiency of the TSA data imputation method with other techniques including the use of mean value and median value imputation techniques. The 2021 of $PM_{2.5}$ data were simulated to lost 20% of the total data by random method. The TSA data imputation method was found to be the most effective. The error measurement with RMSE metric is 2.25, while the data imputation using the mean and the median techniques for data imputation, the RMSE is 6.20 and 5.78, respectively. The comparison of the $PM_{2.5}$ data after using the data imputation process through all three techniques is shown in Figure 3.

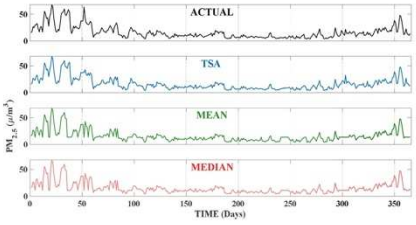


Figure 3. Comparison of $PM_{2.5}$ in 2021 after using TSA, Mean, and Median imputation methods.

B. Experimentation

This research aims to develop a highly efficient time-series model for forecasting $PM_{2.5}$ in advance by introducing a new technique through machine learning with a hybrid algorithm between ARIMA statistical model and non-linear ANFIS algorithm. The proposed model gains benefit from the optimization of the correlation of linear and non-linear of time-series data. Moreover, this research studies and performs comparison with three groups of standard models including the ARIMA model representing the statistical method, the ANFIS model representing the machine learning method, and the LSTM model representing the deep learning method. The research workflow is shown in Figure 4 and the parameter settings are listed in Table II.

TABLE II. The Configuration of the Standand Models.

ARIMA		ANFIS		LSTM	
library	auto_arima	*genfis*	subtractive clustering	*node*	128
p	2	*step size*	1.1	*function*	ReLU
d	1	*learning*	gradient descent	*optimizer*	adam
q	2	*epoch*	1000	*epoch*	100

The research process is divided into three parts, where the first part deals with the data collection for modeling and experimentation. In this research, the $PM_{2.5}$ data were collected from the website of the Pollution Control Department, Ministry of Natural Resources and Environment, Thailand. We select the data of Rayong province, Thailand, with records of $PM_{2.5}$ from 2012 to 2021 (10 years) and clean it with the TSA technique to impute missing values. After that, the dataset is divided into two parts. The first part is for learning, using the data from 2012 to 2020. Second part is the data for testing, which is the data in 2012. The second part of the research workflow is the

process of developing a time-series model for forecasting $PM_{2.5}$ in advance.

This research presents a time-series model for forecasting $PM_{2.5}$, known as ARIMA-FIS, which is a hybrid algorithm between ARIMA with ANFIS.

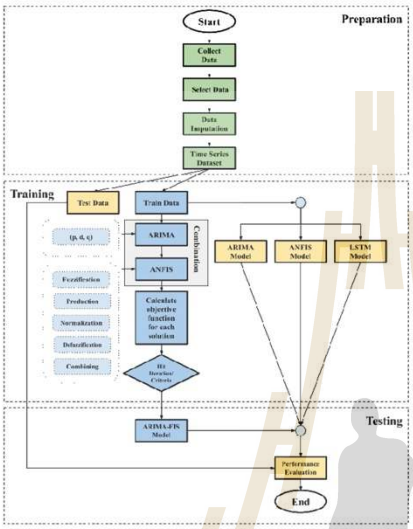


Figure 4. The research workflow.

The ARIMA-FIS works with two sub-processes. Initially, the ARIMA model has been used to forecast the $PM_{2.5}$ value, and the forecasting result is used as one of the inputs of the ANFIS model as shown in Figure 5.

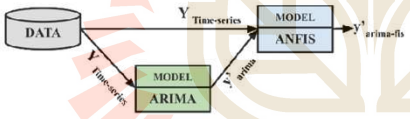


Figure 5. The ARIMA-FIS topology.

At the same time, the ARIMA, ANFIS and LSTM time-series models are created in this work. They represent a group of various kinds of techniques: statistical models, machine learning model, and deep learning model (the configuration of standard models is shown in Table II). These techniques are to be used as a base model for comparing predictive performance with the proposed ARIMA-FIS time-series model.

The final part of the research is the model evaluation and comparison process. We are using the test dataset for forecasting the $PM_{2.5}$ a day in advance. It will be tested for 365 days (year 2021). The testing process using the rolling window technique [20]. We perform test and evaluation on all models, including those presented with ARIMA-FIS and standard models (ARIMA, ANFIS, and LSTM).

IV. Results

We are using $PM_{2.5}$ from 2012 to 2020, accounting for 90% of the total data, as model training data. The remaining 10% is used as test data, which are $PM_{2.5}$ in 2021. The proposed time-series model for forecasting $PM_{2.5}$ is ARIMA-FIS. It shows (in Table III) the best performance compared to other three standard time-series models (ARIMA, ANFIS and LSTM). The ARIMA-FIS time-series model shows performance score with the MAE, RMSE, and %RMSE metrics of 3.46, 5.11, and 30.99, respectively. While the results of the three standard time-series models are as follows, the ARIMA model gives the MAE, RMSE, and %RMSE metrics of 3.69, 5.49 and 33.31, respectively. The ANFIS model produces the MAE, RMSE, and %RMSE metrics of 3.74, 5.47, and 33.61, respectively. Finally, the LSTM model yields the MAE, RMSE, and %RMSE metrics of 4.03, 5.54, and 33.62, respectively. The performance comparison is also graphically shown in Figure 6.

TABLE III. Results of Model Evaluation on Forcasting $PM_{2.5}$ in the Year 2021.

NO.	Model	MAE	RMSE	%RMSE
1	ARIMA	3.69	5.49	33.31
2	ANFIS	3.74	5.47	33.18
3	LSTM	4.03	5.54	33.61
4	ARIMA-FIS	**3.46**	**5.11**	**30.99**

Although the forecasting performance of the ARIMA-FIS model is better than the other standard time-series models, the difference of forecasting results in terms of errors is not too much. The time consumption (Table IV) of the proposed and three standard time-series models at the training (learning) process is significant. It can be seen that ARIMA-FIS takes the learning time less than LSTM and quite comparable to the ANFIS model.

TABLE IV. The Time Consumptions of Training Process.

NO.	Model	Learning Time Consumption (Sec.)
1	ARIMA	None training process
2	ANFIS	63.28
3	LSTM	918.51
4	ARIMA-FIS	71.03

V. Conclusion

This research proposed a high-performance time-series model for forecasting $PM_{2.5}$ for a day in advance using a machine learning process that combines the ARIMA statistical model with a hybrid

non-linear ANFIS model as called the ARIMA-FIS. We used the real-world data as the dataset, which is a daily record of $PM_{2.5}$ values in Rayong province, Thailand. The results showed that the ARIMA-FIS time-series model was effective in correlating both linear and nonlinear time-series data. This makes it highly accurate in forecasting $PM_{2.5}$ in advance. It gave the best performance over MAE, RMSE, and %RMSE metrics based on performance comparisons with the ARIMA, ANFIS, and LSTM standard time-series models, providing nearly 3% more accurate than the standard time-series models.

In the future work, we will study and improve hyper-parameters of the ARIMA-FIS model with the optimization techniques and expand the research dataset to cover other provinces in Thailand.

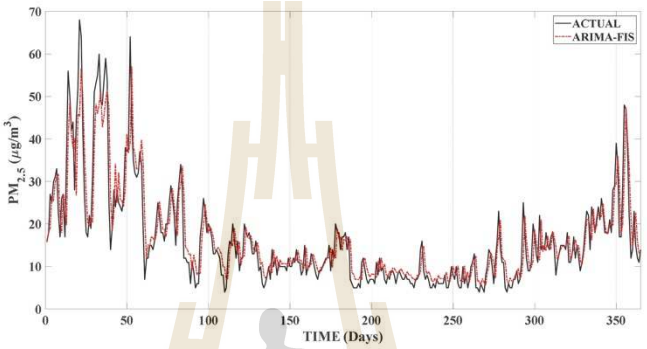


Figure 6. The $PM_{2.5}$ forecast results of the ARIMA-FIS model compare with the observed values.

REFERENCES

[1] WHO, "Ambient air pollution: A global assessment of exposure and burden of disease," World Health Organization, 2016, https://apps.who.int/iris/handle/10665/250141.

[2] IQAir, "World Air Quality Report. Region and City $PM_{2.5}$ Ranking," World Air Quality, 2021, https://www.iqair.com/world-air-quality-report.

[3] J. Nikam, D. Archer, and C. Nopsert, "Air Quality in Thailand: Understanding the regulatory context," SEI Working Paper, Stockholm Environment Institute, 2021, https://www.sei.org/publications/air-quality-thailand-regulatory-context.

[4] G.-P. Bălă, R.-M. Râjnoveanu, E. Tudorache, R. Motișan, and C. Oancea, "Air pollution exposure—the (in)visible risk factor for respiratory diseases," Environ Sci Pollut Res, vol. 28, no. 16, pp. 19615–19628, Apr. 2021, doi: 10.1007/s11356-021-13208-x.

[5] WHO, "WHO global air quality guidelines: particulate matter ($PM_{2.5}$ and PM_{10}), ozone, nitrogen dioxide, sulfur dioxide and carbon monoxide," World Health Organization, 2021, https://apps.who.int/iris/handle/10665/345329.

[6] P. S. G. de Mattos Neto, G. D. C. Cavalcanti, and F. Madeiro, "Nonlinear combination method of forecasters applied to PM time-series," Pattern Recognition Letters, vol. 95, pp. 65–72, Aug. 2017, doi: 10.1016/j.patrec.2017.06.008.

[7] C.-J. Huang and P.-H. Kuo, "A Deep CNN-LSTM Model for Particulate Matter ($PM_{2.5}$) Forecasting in Smart Cities," Sensors, vol. 18, no. 7, p. 2220, Jul. 2018, doi: 10.3390/s18072220.

[8] R. Wongsathan, "Improvement of PM-10 Forecast Using ANFIS Model with an Integrated Hotspots," Science & Technology Asia, vol. 23, p. 6271, 2018, doi: 10.14456/SCITECHASIA.2018.25.

[9] S. Du, T. Li, Y. Yang, and S.-J. Horng, "Deep Air Quality Forecasting Using Hybrid Deep Learning Framework," IEEE Trans. Knowl. Data Eng., vol. 33, no. 6, pp. 2412–2424, Jun. 2021, doi: 10.1109/TKDE.2019.2954510.

[10] J. Amanollahi and S. Ausati, "$PM_{2.5}$ concentration forecasting using ANFIS, EEMD-GRNN, MLP, and MLR models: a case study of Tehran, Iran," Air Qual Atmos Health, vol. 13, no. 2, pp. 161–171, Feb. 2020, doi: 10.1007/s11869-019-00779-5.

[11] S. Jeya and L. Sankari, "Air Pollution Prediction by Deep Learning Model," in 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, May 2020, pp. 736–741. doi: 10.1109/ICICCS48265.2020.9120932.

[12] C. Guo, G. Liu, and C.-H. Chen, "Air Pollution Concentration Forecast Method Based on the Deep Ensemble Neural Network," Wireless Communications and Mobile Computing, vol. 2020, pp. 1–13, Oct. 2020, doi: 10.1155/2020/8854649.

[13] H. Xie, L. Ji, Q. Wang, and Z. Jia, "Research of $PM_{2.5}$ Prediction System Based on CNNs-GRU in Wuxi Urban Area," IOP Conf. Ser.: Earth Environ. Sci., vol. 300, no. 3, p. 032073, Jul. 2019, doi: 10.1088/1755-1315/300/3/032073.

[14] R. Janarthanan, P. Partheeban, K. Somasundaram, and P. Navin Elamparithi, "A deep learning approach for prediction of air quality index in a metropolitan city," Sustainable Cities and Society, vol. 67, p. 102720, Apr. 2021, doi: 10.1016/j.scs.2021.102720.

[15] E. Isaev, B. Ajikeev, U. Shamyrkanov, K. Kalnur, K. Maisalbek, and R. C. Sidle, "Impact of Climate Change and Air Pollution Forecasting Using Machine Learning Techniques in Bishkek," Aerosol Air Qual. Res., vol. 22, no. 3, p. 210336, 2022, doi: 10.4209/aaqr.210336.

[16] Q. Abdulgader, "Time-series Forecasting Using Arima Methodology with Application on Census Data in Iraq," SJUOZ, vol. 4, no. 2, pp. 258–268, Dec. 2016, doi: 10.25271/2016.4.2.116.

[17] S. Mehrmolaei and M. R. Keyvanpour, "Time-series forecasting using improved ARIMA," in 2016 Artificial Intelligence and Robotics (IRANOPEN), Qazvin, Iran, Apr. 2016, pp. 92–97. doi: 10.1109/RIOS.2016.7529496.

[18] J.-S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," IEEE Trans. Syst., Man, Cybern., vol. 23, no. 3, pp. 665–685, Jun. 1993, doi: 10.1109/21.256541.

[19] T. Kim, J. Kim, W. Yang, H. Lee, and J. Choo, "Missing Value Imputation of Time-Series Air-Quality Data via Deep Neural Networks," IJERPH, vol. 18, no. 22, p. 12213, Nov. 2021, doi: 10.3390/ijerph182212213.

[20] C. W. S. Chen and L. M. Chiu, "Ordinal Time-series Forecasting of the Air Quality Index," Entropy, vol. 23, no. 9, p. 1167, Sep. 2021, doi: 10.3390/e23091.

International Journal of Smart Grid and Clean Energy

Renewable Energy Forecasting with Hybrid Nonlinear Model (ANFIS): Case Study of Wind Speed in Thailand

Anupong Banjongkan*, Nittaya Kerdprasop and Kittisak Kerdprasop
School of Computer Engineering, Suranaree University of Technology, Nakhon Ratchasima, 30000, Thailand.

*Correspondence. banjongkan@gmail.com(A.B.)
Manuscript submitted January 20, 2023; revised February 20, 2023; accepted 31 March, 2023; published April 1, 2023.
doi: 10.12720/sgce.12.2.19-29

Abstract: Renewable energy has been a hot topic recently, especially wind power which has grown considerably in the past decade. The forecast of wind speed in advance is important information for wind power plant management. In this paper, a high-efficiency time series model for forecasting wind speed day-ahead is proposed, developed from the nonlinear hybrid model called Adaptive Neuro-Fuzzy Inference System (ANFIS). It brings together the advantages of fuzzy and neural network learning. In addition, a comparative study was done with Autoregressive Integrated Moving Average (ARIMA) and other nonlinear time series models including Artificial Neural Network (ANN) and Long Short-Term Memory (LSTM) models. The realistic data from the meteorological data of Chaiyaphum province, Thailand were used in this research. The dataset was split into learning and testing data in the ratio of 75% and 25%, respectively. The result shows that the forecasting performance of the ANFIS model was comparable to the ARIMA model. Both models achieve high accuracy than other neural network models. The proposed model achieves high efficiency at 22.89 MAPE and 0.41 of R^2. Interestingly, the ANFIS model has a learning time faster than ANN and LSTM models by at least 100 times.

Key words: ANFIS, fuzzy, neural network, renewable energy, time series

1. Introduction

Due to the shortage of main energy resources like petroleum, the price of fuel energy has risen as well as the often control through resource-rich countries in the Middle East. Alternative energy sources with unlimited supply called renewable energy such as wind power, solar radiation, and sea wave power have been receiving continuous attention over the past decade, especially wind power. The growth of wind power plants or wind turbine farms is widespread in each country around the world.

According to [1] reported, in 2021 there is a growth in wind power plants worldwide. It has a total electricity generation capacity of 874 Gigawatt, 13% more than the previous year. The total electricity capacity from wind power plants worldwide in 2022 is expected to reach 955 Gigawatt. Forecasting information in advance is an important part of management and planning. It makes maximized efficiency in managing wind power plants such as planning for the installation of the new wind turbine node and planning for maintenance. The hardware-based wind speed forecasting tools known as physical models have limitations in the setup process and forecasting accuracy [2–3] because the nature of wind speed is random and chaotic. The wind speed records are time series data, which means the value of wind speed depends on time. It says today's value depends on past values. As explained in [4], the components of time series can be

defined in terms of correlation of the time series data as shown in Eq. (1)

$$y_t = Linear_t + Nonlinear_t + e_t \quad (1)$$

where y_t is the actual time series value, $Linear_t$ is linear correlation part, $Nonlinear_t$ is nonlinear correlation part, and e_t is the error part. So, many researchers have developed tools for forecasting wind speed through machine learning with various time series modeling techniques.

The models developed through a neural network proposed by [5] were time series model to be used for forecasting wind energy with ANN technique using feed-forward back-propagation networks with Radial Basis Function (RBF), and Adaptive Linear Element networks were tested with wind data from two sources in the state of North Dakota, USA. Such research found that the time series model RBF neural networks performed well on the site Kulm dataset. The backpropagation network performed well on the site Hann dataset. The research work of [6] presented the improvement of ANN time series models for wind power forecasting based on an optimization technique. Two methods, namely ANN-LM (Levenberg Marquardt) and ANN-PSO (Particle Swarm Optimization) were tested on the collected wind datasets from the IST-University of Lisbon automatic weather station. The multivariate data (wind speed, temperature, humidity, and pressure) were processed in their research. The results showed that the ANN-LM model achieved the best forecasting results. In papers [7–9] the authors used the hybrid technique via a statistical ARIMA model working with the neural network (ANN) model. The results of those work reported that the hybrid models can achieve high accuracy in wind speed forecasting. The statistical model studied by [10] performed well in forecasting long-term wind speed in the Zhangye area with ES-ARMA and ES-GARCH (Generalized Autoregressive Conditional Heteroskedasticity) models, in which ES is Eliminating the Seasonal technique. The results of that research showed that the ES-GARCH time series model gave better long-term wind energy forecasting efficiency than ES-ARMA, with ES-GARCH giving the lowest MAE of 0.30 of the winter data and ES-ARMA also providing the lowest MAE of 0.34 for the winter dataset. The work proposed in [11] also used a hybrid nonlinear model with optimization techniques including the PSO, Genetic Algorithm (GA), and Differential Evolution (DE) to forecast the wind speed of different locations in Malaysia. The overall result showed that the proposed method with PSO and GA outperformed ANFIS standalone and ANFIS-DE. The work of [12] used the regional atmospheric modeling system (RAMS) to simulate 168-h low-level wind forecasts over some areas of Thailand. The results showed good forecasting but consume much computing power.

This research proposed the nonlinear hybrid time series model for forecasting wind speed. Machine learning technique with the ANFIS algorithm is the key to this research sharing the same research scheme as adopted in [11]. However, this research uses a different architecture and fuzzy rule base creation method. Especially, this research focuses on the dataset of Thailand. The comparative study with other time series models are also presented in this research.

The rest of the paper is organized as follows: section2 presents the methods used in the research. The dataset, experiment setup, and testing technique were discussed in section3. The last two sections (4 and 5) are discussing the experimental results and concluding the research, respectively.

2. Methodology

2.1. ARIMA–linear time aeries model

ARIMA is a statistical model developed from the auto-regressive (AR) and moving average (MA) models by adjusting the time series data to stationary. The letter I (Integrate) in the ARIMA model was used for processing the data with the differencing technique. This time series model is parametric. The values of parameters p, d, and q are shown in Eq. (2). The order of an ARIMA model can be determined by using the Autocorrelation Function (ACF) and the Partial Autocorrelation Function (PACF), proposed by Box and

Jenkins [13].

$$y_t = \sum_{i=1}^{p} \alpha_i y_{t-i} + \sum_{j=1}^{q} \beta_j e_{t-j} + \varepsilon_t \quad (2)$$

where α_i is ith autoregressive parameter, β_j is jth moving average parameter, and ε_t is the error at time t.

2.2. ANN–nonlinear time series model

ANN is a basic neural network with the concept of imitating the working process of the human brain to apply to learn information through the machine learning process. The basic ANN is called a Multi-layer Perceptron (MLP), which uses the back-propagation process to learn the data, presented by Rumelhart in 1985 [14]. Fig. 1 shows the simple structure of MLP such that there are three layers of the network structure. The first layer is the input layer, which is responsible for receiving input data. The second layer is the hidden layer, which is the data processing layer that can be modified to contain more than one layer. Finally, is an output layer where each neural node of the hidden and output layer will be processed through Eq. (3).

$$y_i = f(x_i W_i + B_i) \quad (3)$$

where y_i, x_i, W_i, B_i, and $f()$ are the output, input, weight, bias, and activation function of neural node i, respectively.

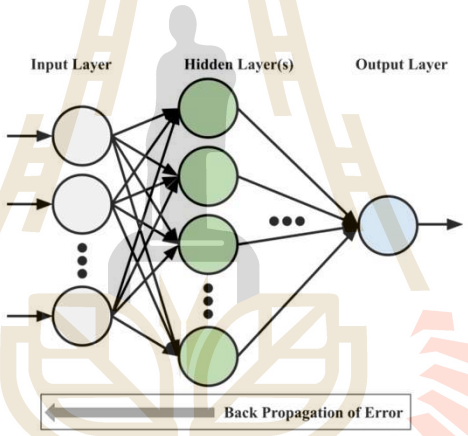


Fig. 1. The multilayer perceptron topology.

2.3. LSTM–nonlinear deep learning model

LSTM is a deep learning neural network improved from the recurrent neural network (RNN). It was proposed by S. Hochreiter and J. Schmidhuber in 1997 [15]. The LSTM was developed to have the ability to memorize and selectively forget some correlations of a time series. Such ability enables the LSTM to capture important patterns of long-term correlation.

An important mechanism that makes LSTM work well with time series data is the "cell state" instead of symbol C in Fig. 2, which is an important part of finding long-term patterns. Due to the complexity of the LSTM structure, it requires high computing power and takes a lot of time to learn the data.

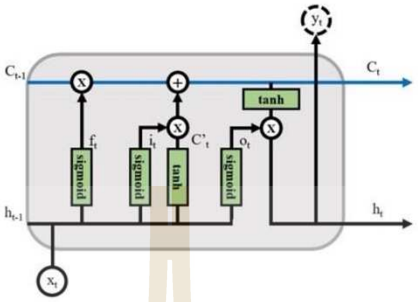


Fig. 2. Show the component inside the LSTM neural node.

2.4. ANFIS–nonlinear hybrid time series model

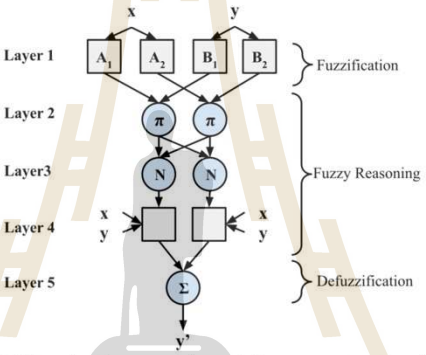


Fig. 3. The adaptive neuro-fuzzy inference system topology.

ANFIS is a nonlinear hybrid model that is a combination of neural networks and fuzzy inference systems proposed by J. S. R. Jang [16]. The advantages of fuzzy inference systems are that they can interpret information by mapping input space to output space through the fuzzy if-then rule as illustrated in Eqs. (4) and (5). The ANFIS takes less time in its learning process than other network models. Moreover, ANFIS brings the capabilities of the neural network to the learning process, making ANFIS robust to the uncertainty of data.

Fig. 3 shows the working process of the ANFIS model that is composed of five layers. ANFIS can learn from data like a general neural network. It uses back-propagation to calculate the error to optimize parameters. The parameter after the "If" condition is called the antecedent parameters (a, b, and c) which are parameters of the membership function, and the parameter after "Then" is called the consequence parameters (p, q, and r) which are the parameters of the output equation.

$$If\ x\ is\ A_1\ and\ y\ is\ B_1, Then\ f_1 = p_1x + q_1y + r_1, \quad (4)$$

$$If\ x\ is\ A_2\ and\ y\ is\ B_2, Then\ f_2 = p_2x + q_2y + r_2 \quad (5)$$

For the fuzzy If-then rules generation process [17], three algorithms can be used including Fuzzy C-mean Clustering (FCM), Grid Partitioning (GP), and Subtractive Clustering (SC).

2.5. Model assessment

Performance evaluation of the time series model for this research adopts five measurement metrics: MAE, MAPE, RMSE, %RMSE, and R^2 [18]. If y_t is the observed value at a time t and y'_t is the forecasted value at the

same time t, then the error (e_t) of forecasting is defined as Eq. (6). The Mean Absolute Error (MAE) can be calculated as shown in Eq. (7). The Mean Absolute Percentage Error (MAPE) can be computed as in Eq. (8). The Root Mean Square Error (RMSE) computation is shown in Eq. (9). The Percentage of Root Mean Square Error (%RMSE) can also be computed as in Eq. (10). These four metrics assess errors of forecasting. The coefficient of determination (R^2) as shown in Eq. (11) is used to compare the performance of the time series models.

$$e_t = y_t - y_t' \tag{6}$$

$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n}|e_t| \tag{7}$$

$$\text{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{e_t}{y_t}\right| \times 100 \tag{8}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(e_t)^2} \tag{9}$$

$$\%\text{RMSE} = \frac{RMSE}{\bar{y}} \times 100 \tag{10}$$

$$R^2 = 1 - \frac{\sum_{t=1}^{n}(e_t)^2}{\sum_{t=1}^{n}(y_t - \bar{y})^2} \tag{11}$$

3. Experimentation

3.1. Dataset

This research uses the meteorological data collected from the open data of the Thailand Meteorological Department. This is a realistic data recorded from the weather ground-base station, which is located in Chaiyaphum province, Thailand. Chaiyaphum is the area that has large private wind turbine farms. This research uses only wind speed data, which are recorded as the average daily wind speed. We used the data from the years 2017-2020 by selecting the data from 2017-2019 for learning and the rest in the year 2020 for model testing. Fig. 4 shows the wind speed data of Chaiyaphum, used in this research.

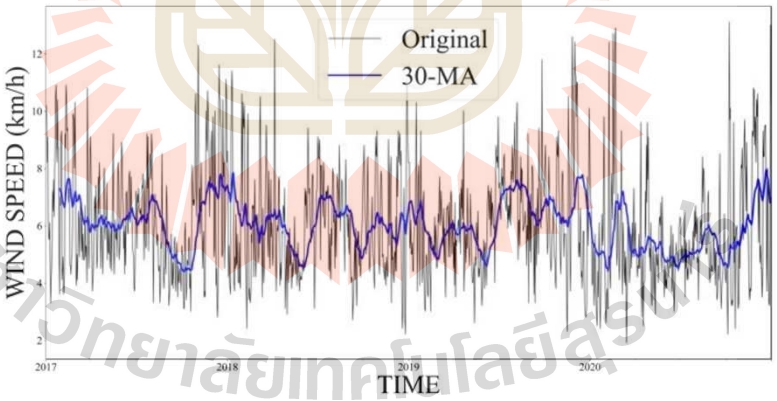


Fig. 4. Wind speed of CHAIYAPHUM site during 2017-2020 (raw and 30 days moving average values represented by gray and blue, respectively).

3.2. Research workflow

The research process is shown in Fig. 5. Firstly, it is the data preparation phase to perform data collection to extract meteorological data of the Chaiyaphum weather station. The second step is to select only the wind speed feature from 2017 to 2020. The Last Observation Carried Forward (LOCF) data imputation technique [19] has been used to complete the dataset. Finally, the dataset has been split into 75% for model training and 25% for testing. That is using data from 2017 to 2019 as a training set, while data in the year 2020 are used as a test set.

The next step of the research workflow is model building. This research develops a highly efficient time series model for forecasting wind speed. This research presents model development through machine learning techniques using the nonlinear hybrid model ANFIS algorithm. The univariate technique has been used. ANFIS time series modeling determines the input using the lag times of the wind speed time series data. We use lag time at y_{t-1}, y_{t-2}, and y_{t-3}, which are predictors and y'_t is the output. While the number and type of membership along with the fuzzy rule define via the subtractive clustering algorithm.

After that, ARIMA, ANN, and LSTM time series modeling have been constructed as a model for performance comparison against the proposed time series model. The ARIMA model is a statistical model which is known as parametric modeling in the sense that it requires three parameter variables including p, d, and q. This research uses the “auto_arima” function in the "pmdarma" library of python to find the p, d, and q values automatically, where the values of the parameters in the ARIMA model are p = 2, d = 0, and q = 1. For the time series modeling of the neural networks group (ANN and LSTM), this research defines the structure of the ANN and LSTM based on the performance of the computer used to execute the models as well as using a trial-and-error manner to decide the suitable neural network structure for this research. The structure of the time series model of the ANN is that there are three inputs (y_{t-1}, y_{t-2}, y_{t-3}) with two hidden layers, each layer with 128 neural nodes, and one output node (y'_t). The LSTM structure is defined to have the same three inputs as well as one output as the ANN model, and the number of LSTM neural nodes is 128.

Optimizing the non-parametric parameter values of the ANN and LSTM models is done through the back-propagation process with a gradient descent algorithm. The activation function of neural nodes of both models is the Rectified Linear Unit (ReLU) function.

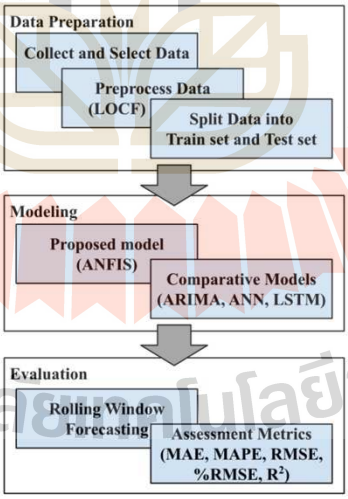


Fig. 5. The research workflow.

The final step of the research is a process to test the efficiency of the nonlinear hybrid model. The unseen data (test set) has been used with the sliding windows technique [20] to feed the input of the model. Five metrics (namely MAE, MAPE, RMSE, %RMSE, and R^2) have been used to evaluate the forecasting performance of the time series model. The comparative time series models (ARIMA, ANN, and LSTM) have been evaluated with the same procedure as the proposed model.

4. Results and Discussion

4.1. The results

The assessment of developing a time series model for forecasting wind speed in Chaiyaphum Province, Thailand, is the performance evaluation of the proposed time series nonlinear hybrid model: ANFIS against other three time series model: ARIMA, ANN, and LSTM based on five measurement metrics. Table 1 shows the results of model evaluation assessed with the training dataset. The results show that the ARIMA model yields MAE, MAPE, RMSE, %RMSE, and R^2 efficiency of 1.06, 18.76, 1.40, 22.67, and 0.42, respectively. While the time series model of ANN gives the forecasting performance as follows: MAE = 1.02, MAPE = 17.81, RMSE = 1.34, %RMSE = 21.77, and R^2 = 0.46. The forecasting performance of LSTM is MAE = 0.88, MAPE = 15.71, RMSE = 1.16, %RMSE = 18.82, and R^2 = 0.60. Lastly, the results of the proposed ANFIS model return the MAE, MAPE, RMSE, %RMSE, and R^2 values as 1.0477, 18.5201, 1.3791, 22.3598, and 0.4359, respectively.

Table 1. The forecasting performance of the trainset

	MAE	MAPE	RMSE	%RMSE	R-Square
ARIMA	1.0668	18.7673	1.4006	22.6709	0.4228
ANN	1.0206	17.8113	1.3428	21.7704	0.4653
LSTM	0.8860	15.7191	1.16092	18.8212	0.6003
ANFIS	**1.0477**	**18.5201**	**1.3791**	**22.3598**	**0.4359**

The results of performance evaluation on the test dataset (Table 2) are that the ARIMA model shows MAE, MAPE, RMSE, %RMSE, and R^2 efficiency of 1.14, 22.43, 1.55, 27.80, and 0.41, respectively. While the time series model of ANN gives the set of results performance MAE = 1.19, MAPE = 23.54, RMSE = 1.59, %RMSE = 28.56, and R^2 = 0.37. The LSTM model gives the set of results performance as MAE = 1.37, MAPE = 28.04, RMSE = 2.02, %RMSE = 36.14, and R^2 = 0.008. Lastly, the results of the proposed ANFIS model return the MAE, MAPE, RMSE, %RMSE, and R^2 values as 1.15, 22.89, 1.55, 27.81, and 0.41, respectively.

Table 2. The forecasting performance of the test set

	MAE	MAPE	RMSE	%RMSE	R-Square
ARIMA	1.1463	22.4311	1.5561	27.8098	0.4119
ANN	1.1904	23.5462	1.5981	28.5605	0.3797
LSTM	1.3790	28.0452	2.0202	36.1047	0.0088
ANFIS	**1.1516**	**22.8986**	**1.5566**	**27.8182**	**0.4115**

4.2. Discussion

From the performance testing results of wind speed forecasting of the proposed ANFIS time series model and other three comparative models. We found that the proposed ANFIS model shows its forecasting efficiency comparable to the statistical ARIMA time series model, and more efficient than both neural network models (ANN and LSTM). The proposed ANFIS model shows robustness to uncertain data better than the ARIMA model. This is because the ARIMA model needs to define the parameters p, d, and q appropriately before running the model.

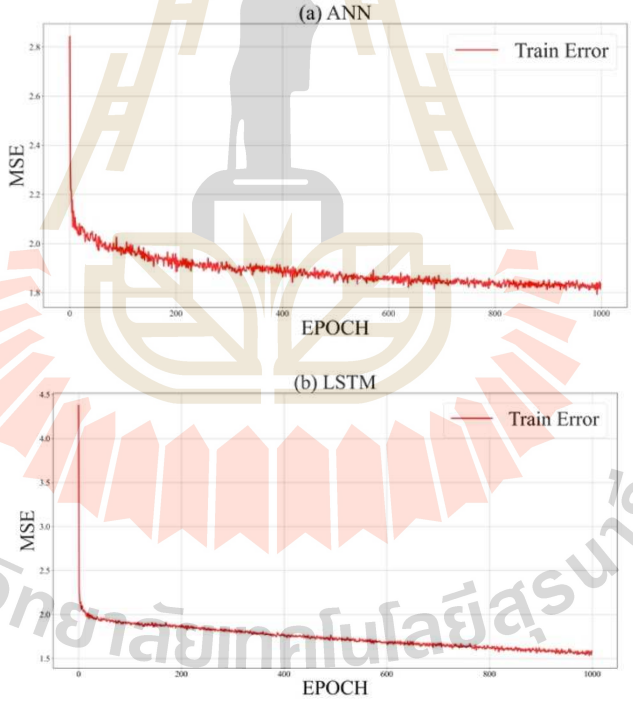
The results show that the forecast within the training set (seen data) of the model in the neural network group performs better than the statistical model. While performing with the test set (unseen data), the neural

network models show lower forecasting performance than the statistical time series model, which means the neural network models tend to be over-fitting. This can be obviously seen in the LSTM model that its coefficient of determination value of 0.0088 in the unseen data.

To consider in the aspect of the learning process of the proposed nonlinear hybrid model (ANFIS), and the comparative model in the neural network group. The result in Table 3 shows that the proposed model has a learning efficiency of about 100 times faster than other neural network models because of the less complex neural structure of ANFIS. Fig. 6 shows the learning performance of the ANFIS model and comparative models. The ANFIS model achieves stable learning error because the ANFIS has initialized hyperparameters through the fuzzy rule that is built on subtractive clustering algorithm in the fuzzy reasoning process. The forecasting of four models compared to the actual observed values is graphically shown in Fig. 7.

Table 3. The number of tuning parameters and time consumption of the neural time series model

	Number of Parameters	Number of Neural Nodes	Time Consumption (Seconds)
ANN	17,153	256	1106.2978
LSTM	66,689	128	1901.3514
ANFIS	**20 with 2 fuzzy rules**	**22**	**9.4320**



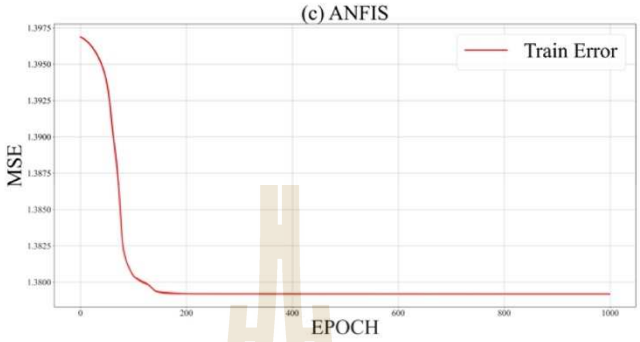


Fig. 6. The graphs showing training errors of (a) ANN, (b) LSTM, and (c) ANFIS.

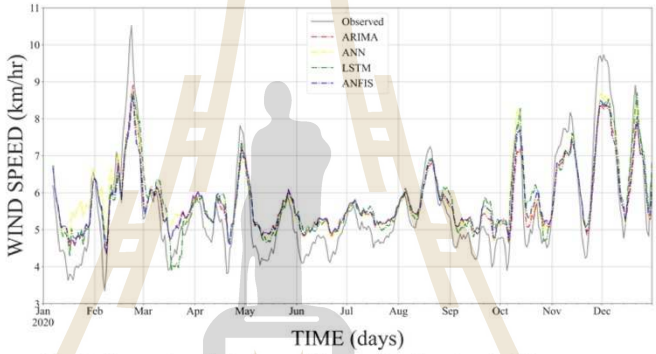


Fig. 7. The results of wind speed forecasting day ahead of the test set.

5. Conclusion

This research presents a time series model for forecasting wind speed using machine learning with the nonlinear hybrid algorithm (ANFIS). The meteorological data used the experimentation are the open data available from the Thai Meteorological Department. The dataset is partitioned into a training set and a test set at a ratio of 75% and 25%, respectively. For performance comparison, ARIMA statistical time series model and ANN and LSTM neural network models have also been developed to compare against the proposed ANFIS model.

For efficiency evaluation of the time series model proposed in this research, the assessment has been done in two aspects. First, the efficiency of forecasting is estimated from MAE, MAPE, RMSE %RMSE, and R2. Then, the structure and the learning speed of the models are considered to reflect the complexity of the model. The results of the model performance evaluation reveal that the proposed ANFIS time series model is slightly better than the neural network model (ANN and LSTM) and the accuracy in wind speed forecasting of ANFIS is similar to the ARIMA model. Efficiency in terms of model building time of ANFIS is the best among neural network models, with at least 100 times better learning speed when measured in seconds.

The results of performance comparisons in this research show that the proposed nonlinear hybrid time series model performs well for wind speed forecasting. It is useful for wind power plants. Our future work is

the improvement of the ANFIS model based on the hyper-parameter tuning of the nonparametric model using global optimization techniques.

Conflict of Interest

The authors declare that they have no conflicts of interest.

Author Contributions

A.B. is responsible for the design, the model development, the data analysis as well as the manuscript preparation. N.K. helps proving the research framework and revising the manuscript. K.K. provides critical feedback and helps proving the algorithms and discussing the experimental results. All authors had approved the final version.

References

[1] World Wind Energy Association (November 2022). *WWEA Half-year Report 2022*. Bonn, Germany. [Online]. Available: https://wwindea.org/wp-content/uploads/2022/11/WWEA_HYR2022.pdf (November 20, 2022).

[2] Olson, J.B., Kenyon, J.S., Djalalova, I., *et al.* (2019). Improving wind energy forecasting through numerical weather prediction model development. *Bulletin of the American Meteorological Society*, 100(11), 2201–2220.

[3] Ye, H., Yang, B., Han, Y., *et al.* (2022). Wind speed and power prediction approaches: classifications, methodologies, and comments. *Frontiers in Energy Research*, 10.

[4] Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50, 159–175.

[5] Li, G., & Shi, J. (2010). On comparing three artificial neural networks for wind speed forecasting. *Applied Energy*, 87(7), 2313–2320.

[6] Nazaré, G., Castro, R., & Gabriel Filho, L. R. A. (2020). Wind power forecast using neural networks: Tuning with optimization techniques and error analysis. *Wind Energy*, 23, 810–824.

[7] Nair, K. R., Vanitha, V., & Jisma, M. (2017). Forecasting of wind speed using ANN, ARIMA and hybrid models, *Proceedings of 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, 170–175.

[8] Singh, P. K., Singh, N., Negi, R. (2019), Wind power forecasting using hybrid ARIMA-ANN technique. In Hu, Yu-Chen, Tiwari, Shailesh, & Mishra, Krishn K., *et al.* (Eds) *Ambient Communications and Computer Systems*. 904, 209–220.

[9] Liu, H., Tian, H., Li, Y. (2012). Comparison of two new ARIMA-ANN and ARIMA-Kalman hybrid methods for wind speed prediction. *Applied Energy*, 98, 415–424.

[10] Guo, Z., Dong, Z., Wang, J., *et al.* (2010). The forecasting procedure for long-term wind speed in the Zhangye area. *Mathematical Problems in Engineering*, 2010, 1–17.

[11] Hossain, M., Mekhilef, S., Afifi, F., *et al.* (2018). Application of the hybrid ANFIS models for long-term wind power density prediction with extrapolation capability. *PLoS ONE*, 13(4).

[12] Pan-Aram, R., Viryasiri, P. (2016). Surface level wind forecast simulation over the land-sea area and mountain-valley area in Thailand. *Proceedings of The Asian Conference on Sustainability, Energy and the Environment 2016: Official Conference Proceedings*, Kobe, Japan.

[13] Box, G. E. P., Jenkins, G. M., & Reinsel, G. C. (1986). *Time Series Analysis: Forecasting and Control. San Francisco*. CA: Holden-Day.

[14] Rumelhart, D. E., Hinton, G. F., & Williams, R. G. (1986). Learning representations by back-propagating

errors. *Nature*, 323, 533–536.

[15] Hochreiter, S., & Schmidhuber, J. (1997). LSTM can solve hard long time lag problems. In Mozer, M. C., Jordan, M. and Petsche, J. T. (Eds.) *Advances in Neural Information Processing Systems*, Cambridge, Massachusetts: MIT Press.

[16] Jang, J. S. R. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(3): 665–685.

[17] Mola, M, Amiri-Ahouee, R. (2021). ANFIS model based on fuzzy C-mean, grid partitioning and subtractive clustering to detection of stator winding inter-turn fault for PM synchronous motor. *International Transactions on Electrical Energy Systems*, 31(3).

[18] Thorp, K. R., Marek, G. W., DeJonge K. C., *et al.* (2020). Comparison of evapotranspiration methods in the DSSAT cropping system model: II. Algorithm performance. *Computers and Electronics in Agriculture*, 177, 105679.

[19] Overall, J. E., Tonidandel, S., Starbuck, R. R. (2009). Last-observation-carried-forward (LOCF) and tests for difference in mean rates of change in controlled repeated measurements designs with dropouts. *Social Science Research*, 38, 492–503.

[20] Vafaeipour, M., Rahbari, O., Rosen, M.A., *et al.* (2014). Application of sliding window technique for prediction of wind velocity time series. *International Journal of Energy and Environmental Engineering*, 5, 1–7.

ประวัติผู้เขียน

นายอนุพงษ์ บรรจงการ เกิดเมื่อวันที่ 20 มกราคม พ.ศ. 2527 ที่ จังหวัดจันทบุรี จบการศึกษาระดับมัธยมศึกษา ที่โรงเรียนเบญจมราชูทิศ จังหวัดจันทบุรี ปีการศึกษา 2545 ได้เข้าศึกษาต่อระดับปริญญาตรีในสาขาวิชาวิทยาการคอมพิวเตอร์ประยุกต์ คณะวิทยาศาสตร์ประยุกต์ สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือและสำเร็จการศึกษาระดับปริญญาตรีปี พ.ศ. 2550 ในปี พ.ศ. 2554 ได้สำเร็จการศึกษาระดับปริญญาโทในสาขาวิชาวิศวกรรมไฟฟ้า แขนงวิศวกรรมคอมพิวเตอร์จากมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ ภายหลังสำเร็จการศึกษาในระดับปริญญาโทได้เข้าทำงานตำแหน่งผู้ช่วยวิจัยของศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยี กระทรวงวิทยาศาสตร์ ณ ขณะนั้น โดยปฏิบัติงานตำแหน่งผู้ช่วยวิจัยของศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ ระหว่างปี พ.ศ. 2555 – 2558 และได้เข้าศึกษาต่อในระดับปริญญาเอกในสาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีสุรนารีในปีการศึกษา 2560

ในระหว่างการศึกษาได้รับการตีพิมพ์เผยแพร่บทความวิชาการทั้งในระดับชาติและระดับประเทศ ซึ่งรายละเอียดของบทความวิชาการที่ตีพิมพ์เผยแพร่แสดงดังภาคผนวก ข