การทำนายผลผลิตอ้อยโดยใช้เทคนิคการเรียนรู้ของเครื่อง: กรณีศึกษาโรงงานน้ำตาลแห่งหนึ่งในประเทศไทย

ปริญญานิพนธ์

ของ

ภูษณิศา เจริญอึ้ง

6 ส.ค. 2562

เสนอต่อบัณฑิตวิทยาลัย มหาวิทยาลัยศรีนครินทรวิโรฒ เพื่อเป็นส่วนหนึ่งของการศึกษา

ตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

กรกฎาคม 2561

ลิขสิทธิ์เป็นของมหาวิทยาลัยศรีนครินทรวิโรฒ

A435504

การทำนายผลผลิตอ้อยโดยใช้เทคนิคการเรียนรู้ของเครื่อง: กรณีศึกษาโรงงานน้ำตาลแห่งหนึ่งในประเทศไทย

บทคัดย่อ

ของ

ภูษณิศา เจริญอึ้ง

เสนอต่อบัณฑิตวิทยาลัย มหาวิทยาลัยศรีนครินทรวิโรฒ เพื่อเป็นส่วนหนึ่งของการศึกษา

ตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

กรกฎาคม 2561

ภูษณิศา เจริญอึ้ง (2561). *การทำนายผลผลิตอ้อยโดยใช้เทคนิคการเรียนรู้ของเครื่อง: กรณีศึกษาโรงงานน้ำตาลแห่งหนึ่งในประเทศไทย*. ปริญญานิพนธ์ วท.ม. (เทคโนโลยีสารสนเทศ). กรุงเทพฯ: บัณฑิตวิทยาลัย มหาวิทยาลัยศรีนครินทรวิโรฒ.
อาจารย์ที่ปรึกษาปริญญานิพนธ์: ผู้ช่วยศาสตราจารย์ ดร.ประดิษฐ์ มิตราปิยานุรักษ์.

งานวิจัยนี้นำเสนอการประยุกต์ใช้เทคนิคการเรียนรู้ของเครื่อง (Machine Learning) สำหรับการทำนายเกรดของปริมาณผลผลิตอ้อยและการทำนายค่าปริมาณผลผลิตอ้อยในหน่วยตันต่อไร่ของแต่ละแปลงอ้อย ข้อมูลที่นำมาใช้ในงานวิจัยนี้มาจากการสำรวจและบันทึกข้อมูลโดยโรงงานน้ำตาลแห่งหนึ่งในประเทศไทย โดยข้อมูลนำเข้าที่ใช้ในการทำนาย ได้แก่ ชนิดดิน ขนาดพื้นที่ปลูก ระยะร่องปลูก ปริมาณผลผลิตในฤดูกาลก่อนหน้า ข้อมูลเกี่ยวกับอ้อยที่ปลูก เช่น พันธุ์อ้อยและชนิดอ้อย วิธีการให้น้ำ แหล่งน้ำ วิธีการให้ปุ๋ย ชนิดปุ๋ย วิธีกำจัดวัชพืชและปริมาณน้ำฝน

งานวิจัยนี้ใช้เทคนิคการทำนายแบบ Random Forest และ Gradient Boosting Tree ร่วมกับเทคนิคการปรับหาค่าพารามิเตอร์ที่ดีที่สุดของแบบจำลอง (Hyper-Parameter Tuning) และเทคนิควิธีการคัดเลือกลักษณะเฉพาะ (Feature Selection) มาใช้ในการพัฒนาแบบจำลองการทำนาย เพื่อเปรียบเทียบความถูกต้องในการทำนายของแต่ละเทคนิค

ผลลัพธ์ค่าความถูกต้อง (Accuracy) ของการทำนายเกรดของปริมาณผลผลิตอ้อยด้วยเทคนิค Random Forest และ Gradient Boosting Tree ที่นำเสนอในงานวิจัยนี้ คือ 71.88% และ 71.71% ตามลำดับ งานวิจัยนี้เปรียบเทียบค่าความถูกต้องกับการทำนายที่ใช้เป็นบรรทัดฐาน (Baseline) จำนวน 2 ตัว ได้แก่ (1) การทำนายผลผลิตโดยใช้ค่าผลผลิตที่ได้จากปีการผลิตก่อนหน้าในแปลงเดียวกัน และ (2) การทำนายผลผลิตของฝ่ายสำรวจของโรงงานที่เป็นกรณีศึกษา โดยผลลัพธ์ค่าความถูกต้องของการทำนายที่ใช้เป็นบรรทัดฐานของทั้งสอง มีค่าเท่ากับ 51.52% และ 65.50% ตามลำดับ ส่วนผลลัพธ์ค่าความผิดพลาดเฉลี่ยกำลังสอง (Root Mean Square Error) ของการทำนายค่าปริมาณผลผลิตอ้อยด้วยเทคนิค Random Forest และ Gradient Boosting Tree ที่นำเสนอในงานวิจัยนี้ มีค่าเท่ากับ 2.18 และ 2.19 ตัน/ไร่ ตามลำดับ โดยผลลัพธ์ดังกล่าวของการทำนายที่ใช้เป็นบรรทัดฐานทั้งสองแบบ คือ 5.05 และ 2.62 ตัน/ไร่ ตามลำดับ

ผลลัพธ์ดังกล่าวแสดงให้เห็นว่าเทคนิคการทำนายโดยใช้เทคนิคการเรียนรู้ของเครื่องที่นำเสนอในงานวิจัยนี้สามารถให้ค่าความถูกต้องที่มากกว่า และค่าความผิดพลาดเฉลี่ยกำลังสองที่น้อยกว่าค่าของการทำนายบรรทัดฐาน จึงสามารถนำไปประยุกต์ใช้เป็นตัวช่วยในการตัดสินใจในการวางแผนการผลิตของโรงงานน้ำตาลที่เหมาะสมกว่าการทำนายผลผลิตของฝ่ายสำรวจของโรงงาน

SUGARCANE YIELD PREDICTION USING MACHINE LEARNING TECHNIQUE: CASE STUDY OF A SUGAR FACTORY IN THAILAND

AN ABSTRACT

BY

PHUSANISA CHAROEN-UNG

Presented in Partial Fulfillment of the Requirements for the

Master of Science Degree in Information Technology

At Srinakharinwirot University

July 2018

Phusanisa Charoen-Ung. (2018). *Sugarcane Yield Prediction using Machine Learning Technique: Case Study of a Sugar Factory in Thailand.* Master's thesis, M.Sc. (Information Technology). Bangkok: Graduate School, Srinakharinwirot University. Advisor Committee: Asst. Prof. Pradit Mittrapiyanuruk.

This research presents a machine learning based method to predict sugarcane yield grade and sugarcane yield value of an individual plot. The dataset used in this work was obtained from a set of sugarcane plots around a sugar mill in Thailand. The features used in the predictions consisted of soil type, plot area, groove width, plot yield from last year, cane class, cane type, water resource type, irrigation method, epidemic control method, fertilizer type, fertilizer formula and rain volume.

Our proposed method was based on two supervised machine learning algorithms: (1) Random Forest (RF) and (2) Gradient Boosting Tree (GBT). A forward feature selection was also proposed in conjunction with hyper-parameter tuning for develop a training model. The evaluation of the prediction performances of several proposed models that combined different configurations of algorithms, feature selection and hyper-parameter tuning. The prediction performances of the two non-machine-learning baselines were combined: (1) baseline predictions based on the actual yield from the last year of the same plot, referred to as Baseline-1, and (2) baseline predictions based on the target yield of each plot provided by human experts, referred to as Baseline-2.

In terms of the yield grade prediction, the accuracy of the RF based model and accuracy of the GBT-based model were 71.88% and 71.62%, respectively. Meanwhile, the accuracies of Baseline-1 and Baseline-2 were 51.52% and 65.50%, respectively. For the yield value prediction, the root mean squred error (RMSE) of the RF based model and the RMSE of our GBT based model were 2.18 and 2.19 ton/rai, respectively. Meanwhile, the RMSE values of Baseline-1 and Baseline-2 were 5.05 and 2.62 ton/rai, respectively.

Based on the aforementioned results, the proposed machine learning based yield prediction method outperformed the prediction provided by human experts. Therefore, this method can be used to help the decision-making of the operational management of sugar mills.

ปริญญานิพนธ์

เรื่อง

การทำนายผลผลิตอ้อยโดยใช้เทคนิคการเรียนรู้ของเครื่อง:

กรณีศึกษาโรงงานน้ำตาลแห่งหนึ่งในประเทศไทย

ของ

ภูษณิศา เจริญอิ้ง

ได้รับอนุมัติจากบัณฑิตวิทยาลัยให้นับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ของมหาวิทยาลัยศรีนครินทรวิโรฒ

.....คณบดีบัณฑิตวิทยาลัย

(ผู้ช่วยศาสตราจารย์ นายแพทย์ฉัตรชัย เอกปัญญาสกุล)

วันที่ 7 เดือน กรกฎาคม พ.ศ. 2561

อาจารย์ที่ปรึกษาปริญญานิพนธ์

.....ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์ ดร.ประดิษฐ์ มิตราปิยานุรักษ์)

คณะกรรมการสอบปากเปล่า

.....ประธาน

(ผู้ช่วยศาสตราจารย์ ดร.วราภรณ์ วิยานนท์)

.....กรรมการ

(อาจารย์ ดร. อัครินทร์ ไพบูลย์พานิช)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.นุวีย์ วิวัฒนวัฒนา)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.จันตรี ผลประเสริฐ)

.....กรรมการ

(อาจารย์ ดร. ศุภชัย ไทยเจริญ)

.....กรรมการ

(อาจารย์ ดร. วีรยุทธ เจริญเรืองกิจ)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ประดิษฐ์ มิตราปิยานุรักษ์)

ประกาศคุณูปการ

ปริญญานิพนธ์นี้สำเร็จลุล่วงได้ด้วยความกรุณาช่วยเหลือ และความเอาใจใส่อย่างดียิ่ง ตลอดจนการให้คำแนะนำ และข้อคิดเห็นที่เป็นประโยชน์สำหรับการปรับแก้ไขข้อบกพร่อง จาก คณะกรรมการผู้ควบคุมปริญญานิพนธ์ทุกท่าน ผู้วิจัยขอกราบขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร. ประดิษฐ์ มิตราปิยานุรักษ์ ที่กรุณาเป็นที่ปรึกษา และสละเวลาให้ความช่วยเหลือตลอดจนชี้แนะแนวทางในสิ่งที่เป็นประโยชน์ต่อการศึกษา และการทำปริญญานิพนธ์นี้ด้วยความเอาใจใส่ตลอดมา ทำให้ปริญญานิพนธ์ฉบับนี้มีความสมบูรณ์ยิ่งขึ้น ผู้วิจัยจึงขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ ที่นี้

ขอกราบขอบพระคุณคณาจารย์ และกรรมการบริหารหลักสูตรสาขาวิชาเทคโนโลยีสารสนเทศ คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒทุกท่าน ที่ได้กรุณาประสิทธิ์ประสาทความรู้ต่างๆให้แก่ผู้วิจัย ตลอดจนให้ความช่วยเหลือในการทำวิจัยครั้งนี้

ขอขอบคุณทุนสนับสนุนการเข้าร่วมประชุมและเสนอผลงานฯ จากบัณฑิตวิทยาลัย มหาวิทยาลัยศรีนครินทรวิโรฒ ประจำปีงบประมาณ 2561

ขอขอบคุณเพื่อนๆ น้องๆ สาขาวิชาเทคโนโลยีสารสนเทศที่คอยให้ความช่วยเหลือ ให้คำปรึกษา และคำแนะนำที่ดีให้กับผู้วิจัยเสมอมา

นอกจากนี้ยังมีผู้ที่ให้ความร่วมมือช่วยเหลืออีกหลายท่าน ซึ่งผู้วิจัยไม่สามารถกล่าวนามในที่นี้ได้หมด จึงขอขอบคุณทุกท่านเหล่านั้นไว้ ณ โอกาสนี้ด้วย

เหนือสิ่งอื่นใดขอกราบขอบพระคุณบิดา มารดา และญาติพี่น้องของผู้วิจัยที่ให้กำลังใจกับผู้วิจัยมาโดยตลอด และให้การสนับสนุนในทุกๆ ด้านอย่างดีที่สุดเสมอมา

ภูษณิศา เจริญอึ้ง

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

บัญชีตาราง

บัญชีตาราง (ต่อ)

ตาราง หน้า

บัญชีภาพประกอบ

บัญชีภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของปัญหา

โรงงานน้ำตาลจำเป็นต้องทราบถึงปริมาณอ้อยที่เข้าสู่โรงงานในแต่ละปีการผลิต โดยทางโรงงานน้ำตาลจะทำการสำรวจการปลูกอ้อยจากแปลงของชาวไร่ที่มาทำสัญญาส่งอ้อยให้กับโรงงาน เพื่อใช้ประเมินปริมาณผลผลิตอ้อยที่คาดว่าจะได้รับ ซึ่งในการลงสำรวจอ้อยในแต่ละพื้นที่นั้นมีฝ่ายสำรวจไร่ไปทำการสำรวจปัจจัยต่าง ๆ ที่เกี่ยวกับการปลูกอ้อยและประเมินปริมาณผลผลิตอ้อยในแต่ละแปลง

ผลผลิตอ้อยที่ฝ่ายสำรวจประเมินมานั้นจะถูกนำไปใช้วางแผนการหีบอ้อยของโรงงาน การจัดระบบคิวรับอ้อยให้เหมาะสม การทำสัญญาส่งอ้อยและการปล่อยสินเชื่อให้ชาวไร่ เป็นต้น ถ้าผลผลิตอ้อยที่ได้น้อยเกินไป ฝ่ายสำรวจของโรงงานจำเป็นต้องจัดหาให้เพียงพอต่อความต้องการ นอกจากนี้โรงงานน้ำตาลจำเป็นต้องวางแผนการหีบอ้อยให้ได้มากที่สุด และให้ทันกับกำหนดการเปิดและปิดหีบที่ทางสำนักงานคณะกรรมการอ้อยและน้ำตาลทรายกำหนดไว้

ดังนั้นการทำนายผลผลิตอ้อยจึงจำเป็นอย่างยิ่งสำหรับการวางแผนงานดังกล่าว ในปัจจุบันการประเมินผลผลิตอ้อยใช้จากประสบการณ์ของชาวไร่และนักสำรวจ โดยใช้วิธีการคำนวณแบบง่าย คือการใช้จำนวนพื้นที่ที่มีการปลูกอ้อยมาคูณกับผลผลิตเฉลี่ยต่อไร่ ซึ่งผลผลิตเฉลี่ยต่อไร่นี้เป็นค่าประมาณที่มาจากประสบการณ์ของชาวไร่และนักสำรวจ แต่ปริมาณอ้อยที่ได้นี้มีความคลาดเคลื่อนอยู่มากทำให้การวางแผนงานต่าง ๆ ของโรงงานมีความผิดพลาด ดังนั้นการที่โรงงานน้ำตาลมีแบบจำลองการทำนาย (Predicitve model) ที่มีประสิทธิภาพมาช่วยในการทำนายจึงเป็นประโยชน์ต่อการวางแผนในการบริหารจัดการกับการหีบอ้อยของโรงงานน้ำตาลอย่างมาก

ในงานวิจัยนี้พัฒนาแบบจำลองการทำนายผลผลิตอ้อย (Sugarcane Yield Prediction) โดยใช้เทคนิคการเรียนรู้ของเครื่อง (Machine Learning) แบบจำลองนี้อยู่ในรูปของโปรแกรมคอมพิวเตอร์ที่รับข้อมูลอินพุตจากฐานข้อมูลแล้วให้ผลลัพธ์การทำนายเป็นเกรดที่แสดงระดับของปริมาณผลผลิตอ้อยของแต่ละแปลงและผลลัพธ์การทำนายเป็นค่าตัวเลขปริมาณผลผลิตอ้อยในหน่วยตันต่อไร่ของแต่ละแปลง

1.2 วัตถุประสงค์ของการวิจัย

(1) เพื่อศึกษาและประยุกต์ใช้เทคนิคการเรียนรู้ของเครื่อง (Machine Learning) มาสร้างแบบจำลองการทำนายสำหรับใช้ในการทำนายปริมาณผลผลิตอ้อยของแต่ละแปลง เพื่อนำไปใช้เพิ่มประสิทธิภาพในการวางแผนบริหารจัดการกับปริมาณอ้อยของโรงงานน้ำตาล

(2) เพื่อวิเคราะห์แบบจำลองการทำนายว่าข้อมูลที่นำมาใช้งานนั้น มีตัวแปรใดบ้างที่มีผลต่อความถูกต้องในการทำนาย

(3) เพื่อศึกษาและเปรียบเทียบเทคนิคการเรียนรู้ของเครื่องแบบต่าง ๆ ที่มีผลต่อความถูกต้องในการทำนายปริมาณผลผลิตอ้อย

1.3 ขอบเขตของการวิจัย

งานวิจัยนี้พัฒนาแบบจำลองการทำนายโดยใช้เทคนิคการเรียนรู้ของเครื่อง (Machine Learning) ที่ทำงานในรูปแบบโปรแกรมคอมพิวเตอร์ โดยใช้ข้อมูลที่มาจากการสำรวจจากพื้นที่การปลูกและที่ตั้งแปลงอ้อยของชาวไร่ที่ทำสัญญาการส่งอ้อยไว้กับโรงงานน้ำตาลแห่งหนึ่ง ข้อมูลที่ใช้ในการทำนายประกอบด้วยข้อมูลการปลูกอ้อย ได้แก่ ข้อมูลชนิดอ้อย พันธ์อ้อย ชนิดดิน วิธีการให้น้ำ ปริมาณน้ำฝน ข้อมูลปุ๋ย ระยะร่องปลูก ปริมาณผลผลิตในปีก่อนหน้า

งานวิจัยนี้สามารถให้ผลลัพธ์การทำนายเป็นเกรดที่แสดงระดับของปริมาณผลผลิตอ้อยของแต่ละแปลงและผลลัพธ์การทำนายเป็นค่าของปริมาณผลผลิตอ้อยในหน่วยตันต่อไร่ของแต่ละแปลง โดยมีการวัดความถูกต้องจากแบบจำลองที่พัฒนาเพื่อเปรียบเทียบกับค่าผลผลิตอ้อยที่ชาวไร่เจ้าของแปลงนั้น ๆ ได้มาส่งอ้อยให้กับโรงงานและเกรดของปริมาณอ้อยต่อไร่ที่ได้จัดกลุ่มไว้

เทคนิคการเรียนรู้ของเครื่องที่ประยุกต์ในงานวิจัยนี้ ประกอบด้วยเทคนิค Random Forest [1] และ Gradient Boosting Tree [2] โดยนำไปใช้ทั้งในรูปแบบของปัญหา Classification และปัญหา Regression โดยงานวิจัยนี้ศึกษาความถูกต้องในการทำนายของแต่ละเทคนิคแล้วนำมาเปรียบเทียบกันเพื่อศึกษาสมรรถนะของแต่ละอัลกอริทึม

การพิจารณาปัญหาการทำนายผลผลิตอ้อยในรูปของ Classification นั้น มีการแปลงค่าผลผลิตอ้อยต่อไร่ 3 กลุ่ม ได้แก่ (1) ผลผลิตน้อย (2) ผลผลิตปานกลาง และ (3) ผลผลิตมาก และการวัดค่าความถูกต้องถูกวัดในรูปแบบของค่าความถูกต้อง (Accuracy) ของแต่ละข้อมูลทดสอบว่าถูกต้องตรงกับค่าผลผลิตจริง (Label) หรือไม่

ส่วนการพิจารณาปัญหาการทำนายผลผลิตอ้อยในรูปแบบ Regression ระบบทำนายผลลัพธ์ออกมาเป็นค่าตัวเลขต่อเนื่อง (Continuous value) ของปริมาณอ้อย (หน่วยตันต่อไร่) และการวัดค่าความถูกต้องจะวัดค่าความผิดพลาด (Error) ระหว่างค่าผลผลิตจริง กับ ค่าที่ระบบทำนายของแต่ละข้อมูลทดสอบและคำนวณค่าทางสถิติของ Error ออกมาในรูปของ Root Mean Square Error (RMSE)

สำหรับเครื่องมือที่ใช้ในงานวิจัยนี้ ผู้วิจัยได้เลือกใช้เครื่องมือในการพัฒนาโปรแกรมคอมพิวเตอร์ โดยใช้ภาษาโปรแกรม Python ร่วมกับไลบรารีต่าง ๆ ได้แก่ NumPy, Panda, Matplotlib, Scikit-Learn [3] และ XGBoost [4] โดยผู้วิจัยได้เลือกใช้ Python Distribution ที่ชื่อ Anaconda3[1] ซึ่งเป็น Distribution สำหรับการทำงานด้าน Data Science ที่ประกอบไปด้วย IDE และเครื่องมือต่าง ๆ สำหรับการเขียนโปรแกรมภาษา Python รวมทั้ง Package ของไลบรารีต่าง ๆ ที่จำเป็น ซึ่งผู้วิจัยเลือกใช้ IDE คือ Spyder

1.4 วิธีดำเนินการวิจัย

ในงานวิจัยนี้ ผู้วิจัยได้ดำเนินการงานวิจัยตามขั้นตอนดังนี้

(1) การทบทวนวรรณกรรมและงานวิจัย (Literature Review) ที่เกี่ยวข้องที่สามารถนำความรู้มาประยุกต์ใช้งานได้

(2) การแปลงรูปแบบข้อมูลให้อยู่ในรูปแบบที่เหมาะสมเพื่อใช้กับซอฟท์แวร์ที่นำมาใช้ทำแบบจำลอง

(3) ศึกษาการทำ Feature Selection และ Hyper-Parameter Tuning โดยใช้ Grid Search ร่วมกับการใช้อัลกอริทึมต่าง ๆ คือ Random Forest และ Gradient Boosting Tree (โดยใช้ไลบรารี XGBoost)

(4) สร้างแบบจำลองการทำนายในรูปโปรแกรม โดยใช้อัลกอริทึมต่าง ๆ คือ Random Forest และ Gradient Boosting Tree (โดยใช้ไลบรารี XGBoost)

(5) หาค่าความถูกต้อง (Accuracy) และ Root Mean Square Error (RMSE) จากอัลกอริทึมแต่ละแบบและทำการเปรียบเทียบผลลัพธ์ของวิธีต่าง ๆ

(6) ประเมินผลและสรุปงานวิจัย

---

[1] https://www.anaconda.com/distribution/

1.5 ประโยชน์ที่คาดว่าจะได้รับจากการวิจัย

ประโยชน์จากงานวิจัยนี้คือสามารถนำผลลัพธ์การทำนายไปช่วยเพิ่มประสิทธิภาพของการวางแผนบริหารจัดการกับปริมาณผลผลิตอ้อยที่เข้าโรงงานน้ำตาลได้อย่างถูกต้องมากขึ้นและทำให้ทราบถึงปัจจัยหลักที่ส่งผลต่อผลผลิตอ้อยต่อไร่

การทำนายผลผลิตตันอ้อยให้มีความถูกต้องมากขึ้นนั้นส่งผลให้ฝ่ายสำรวจของโรงงานน้ำตาลทราบถึงปริมาณอ้อยที่จะได้ทั้งหมดและสามารถใช้เพื่อจัดหาอ้อยเข้าโรงงานให้เพียงพอกับความต้องการทั้งฤดูกาลหีบอ้อยและใช้จัดการระบบคิวอ้อยได้เหมาะสมเพื่อให้ปริมาณอ้อยที่เข้ามาหีบในแต่ละวันเหมาะสมกับกำลังการผลิตของเครื่องจักร ฝ่ายผลิตสามารถวางแผนการบำรุงรักษาเครื่องจักรให้มีกำลังการผลิตของเครื่องจักรได้อย่างต่อเนื่องตามกำหนดเวลาตลอดฤดูกาลหีบอ้อย เพิ่มความรวดเร็วในกระบวนการทำงานของฝ่ายสินเชื่อในการพิจารณาการทำสัญญาส่งอ้อยและช่วยการคำนวณยอดสินเชื่อที่ชาวไร่สามารถทำสัญญากู้กับทางโรงงานน้ำตาลได้ อีกทั้งยังมีประโยชน์ต่อเกษตรกรชาวไร่โดยสามารถเพิ่มผลผลิตอ้อยจากการใช้ปัจจัยต่าง ๆ ที่มีผลกับผลผลิตตันอ้อยเป็นแนวทางในการปลูกได้

บทที่ 2
แนวคิด ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 การเรียนรู้ของเครื่อง (Machine learning)

เป็นศาสตร์ทางวิทยาการคอมพิวเตอร์ที่เกี่ยวข้องกับการพัฒนาโปรแกรมคอมพิวเตอร์เพื่อให้คอมพิวเตอร์สามารถเรียนรู้ได้จากข้อมูลที่รับเข้าไป แล้วคอมพิวเตอร์สามารถให้ผลลัพธ์ออกมาจากการเรียนรู้รูปแบบของข้อมูลที่ป้อนเข้าไป จนสามารถสร้างเป็นแบบจำลองที่สามารถทำนายข้อมูลได้หรือช่วยในการตัดสินใจได้ โดยเทคนิคการเรียนรู้ด้วยเครื่อง (Machine learning) สามารถจำแนกออกเป็น 3 ประเภทตามรูปแบบของการช่วยเหลือ (supervision) ระหว่างการเรียนรู้ได้ดังนี้

2.1.1 Supervised learning (การเรียนรู้แบบมีผู้สอน)

เป็นเทคนิคการเรียนรู้ของเครื่องจากข้อมูลสอน (training data) ที่ประกอบด้วยข้อมูลที่รับเข้ามาและมีคำตอบที่เป็นผลลัพธ์ (label) โดยการทำนายผลลัพธ์สามารถแบ่งได้ 2 ประเภท ได้แก่

(1) ปัญหาการทำนายคลาสหรือประเภทของข้อมูลอินพุต (Classification) คือเทคนิคการทำนายที่มี label และผลของการทำนายเป็นข้อมูลชนิดหรือประเภทของข้อมูล (Category) เช่น โปรแกรมการทำนาย spam email ซึ่งจะแยกแยะว่าอีเมล์หนึ่ง ๆ เป็น spam หรือไม่ใช่ spam

(2) ปัญหาการทำนายค่าจากข้อมูลอินพุต (Regression) คือเทคนิคการทำนายที่มี label และผลของการทำนายเป็นค่าตัวเลขต่อเนื่อง (Continuous value) เช่น โปรแกรมการทำนายราคาบ้าน

2.1.2 Unsupervised learning (การเรียนรู้แบบไม่มีผู้สอน)

เป็นเทคนิคการเรียนรู้ของเครื่องที่ข้อมูลสอน (training data) ไม่มี label โดยเทคนิคเป็นการค้นหาโครงสร้างและความสัมพันธ์ของข้อมูลที่ซ่อนอยู่ โดยเทคนิค Unsupervised learning หลัก ๆมีดังนี้

(1) Clustering เป็นการจัดกลุ่มของข้อมูลบนพื้นฐานของความเหมือน (similarities) และความแตกต่าง (differences) ของข้อมูล

(2) Dimensionality Reduction เป็นเทคนิคที่ทำการแปลงข้อมูลให้อยู่ในรูปแบบที่ง่ายขึ้นโดยที่ข้อมูลใหม่ไม่สูยเสียไปจากข้อมูลเดิมมากนัก ตัวอย่างเช่นเทคนิค PCA (Principle Component Analysis)

(3) Association Rules Learning เป็นเทคนิคหากฎของความสัมพันธ์ของข้อมูล เช่น การประยุกต์ใช้ทำ Market Basket Analysis

2.1.3 Reinforcement learning (การเรียนรู้แบบเสริมกำลัง)

เป็นเทคนิคการเรียนรู้ของเครื่อง ซึ่งมีรูปแบบของตัวแทน (agent) ที่เรียนรู้จากการสังเกตุสิ่งแวดล้อม (environment) แล้วเลือกที่จะกระทำ (action) เพื่อให้ได้รับผลตอบแทน (reward) โดยเป้าหมายการเรียนรู้คือหากลวิธี (policy) ที่ดีที่สุดที่ทำให้ได้รับ reward มากสุด

2.2 ทฤษฎีพื้นฐานของเทคนิค Decision Tree และ Ensemble of Decision Trees

2.2.1 ทฤษฎีเกี่ยวกับ Decision Tree

เทคนิคการทำนายแบบ Decision Tree เป็นแบบจำลองการทำนายประเภท Supervised learning ที่มีการสร้างการทำนายในรูปแบบเชิงลำดับชั้น (hierarchy) ของการตัดสินใจแบบ if-else โดยทำการจัดกลุ่มข้อมูลตามคุณลักษณะต่าง ๆ ซึ่งสามารถเห็นลักษณะโครงสร้างคล้ายกับต้นไม้ โดยมีการสร้างกฎต่าง ๆ ขึ้นจากข้อมูลเรียนรู้ (training data) เพื่อใช้ในการตัดสินใจ ซึ่ง Decision Tree นั้นมีรูปร่างเหมือนต้นไม้กลับหัว โดยประกอบด้วยโหนด (node) ที่ระบุเงื่อนไขของการตัดสินใจจากการตรวจสอบคุณลักษณะ (feature หรือ attribute) ของข้อมูลอินพุต โดยโหนดแรกสุดเป็น root node และแต่ละกิ่ง (branch) จะแทนผลลัพธ์การตัดสินใจตามเงื่อนไขของคุณลักษณะของข้อมูลอินพุต จนกระทั่งถึงโหนดสุดท้าย (leaf node)

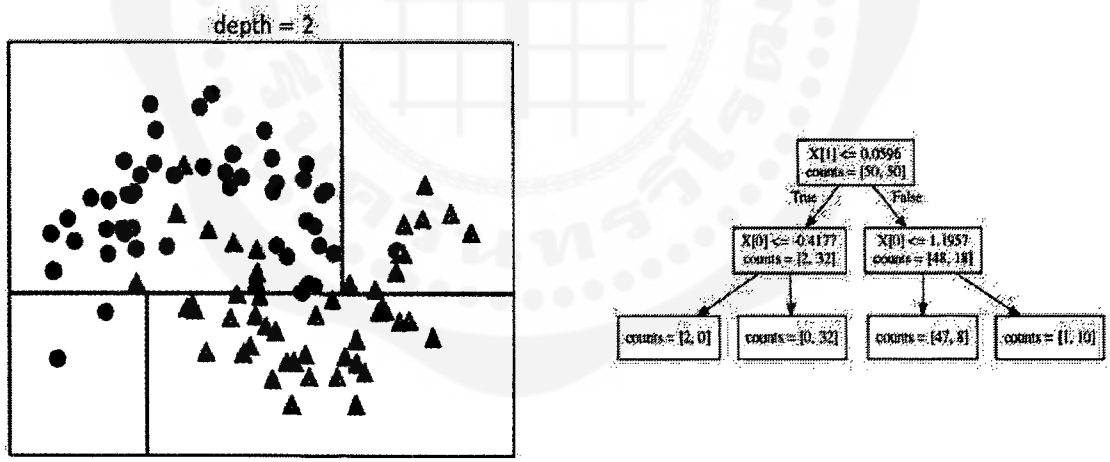
การสร้าง Decision Tree จากข้อมูลเรียนรู้ สามารถอธิบายได้ดังนี้ ถ้าสมมติข้อมูลเรียนรู้ประกอบด้วย ข้อมูลอินพุตจำนวน n ข้อมูล แทนด้วย $\{x_1, x_2, x_3, \dots, x_n\}$ และคำตอบหรือ label ของแต่ละข้อมูลอินพุต แทนด้วย $\{y_1, y_2, y_3, \dots, y_n\}$ และสมมติว่าข้อมูลอินพุต x_j หนึ่ง ๆ จะประกอบด้วยค่าของคุณลักษณะ (feature หรือ attribute) จำนวน p ค่าแทนด้วย $x_j^{(1)}, x_j^{(2)}, \dots, x_j^{(p)}$

การสร้าง Decision Tree จากชุดของข้อมูลเรียนรู้ จะเริ่มจาก Decision Tree ที่ไม่มีโหนดอะไร (Empty tree) หลังจากนั้นจะเริ่มสร้างโหนดการตัดสินใจโดยเลือก feature ในบรรดา p คุณลักษณะ (สมมติว่า feature ที่เลือกคือ k) และเลือกค่า threshold สมมติแทนด้วย t ที่จะใช้เป็นเงื่อนไขการตัดสินใจในรูปแบบ $x^{(k)} \geq t$ ใช่หรือไม่ โดยเงื่อนไขในการเลือกคือเมื่อแบ่งข้อมูลออกตามเงื่อนไขดังกล่าวแล้ว จะทำให้ค่า node impurity ของข้อมูลหลังการแบ่งตามเงื่อนไขมีค่าน้อยที่สุด หลังจากนั้นจะวิ่งไล่แบบเวียนเกิด (Recurse) วิเคราะห์ข้อมูลข้อมูลเรียนรู้ที่แบ่งอยู่ในแต่ละโหนดที่สร้างใหม่ เพื่อทำการสร้างโหนดย่อยและแบ่งข้อมูลไปเรื่อย ๆ จนกว่า จะถึงเงื่อนไขการหยุดสร้างโหนด

(Stoppping criteria) ซึ่งสามารถกำหนดด้วยพารามิเตอร์ต่าง ๆ เช่น Depth หรือ จำนวนลำดับชั้นของ Tree ที่มากสุด เป็นต้น

การทำนายผลลัพธ์เมื่อกำหนดข้อมูลอินพุตหนึ่ง ๆ มาให้ อัลกรอริทึมจะวิ่งไล่ไปตามกิ่งก้านของ Decision Tree ตามผลลัพธ์การตัดสินใจของเงื่อนไขในแต่ละโหนด โดยใช้ค่าของข้อมูลอินพุตตาม feature และค่า Threshold ที่ระบุในโหนดนั้น ๆ โดยจะวิ่งไล่ไปเรื่อย ๆ ไปจนถึง leaf node หลังจากนั้นการทำนายผลลัพธ์จะแตกต่างกันในกรณีของ Classification และ Regression โดยกรณี Classification ผลลัพธ์การทำนายจะตอบเป็น class label ของข้อมูลเรียนรู้ที่ปรากฏมากสุด (Majority) ใน leaf node นั้น ๆ ส่วนผลลัพธ์การทำนายในกรณีของ Regression จะตอบเป็นค่าเฉลี่ยของ value label ของข้อมูลเรียนรู้ที่อยู่ใน leaf node นั้น ๆ

ตัวอย่างของ Decision Tree ซึ่งแสดง Decision Tree ที่ถูกแบ่งสองลำดับ (depth = 2) โดยลำดับที่ 1 (depth = 1) ใช้ decision คือ X[1] <= 0.0596 และแบ่งในลำดับที่ 2 (depth = 2) โดยใช้ decision ตามที่แสดงในภาพประกอบ 1 โดยรูปทางซ้ายเป็นชุดข้อมูล 2 มิติ (X[0] และ X[1]) ส่วนรูปทางขวาแสดง Decision Tree Diagram



ภาพประกอบ 1 ตัวอย่างการทำ Decision Tree

(ที่มา: Introduction to Machine Learning with Python (p. 73), by Sarah Guido and Andreas Müller, 2016, O'Reilly Media)

โดยอัลกอริทึม Decision Tree ที่ใช้ในงานวิจัยนี้เป็นอัลกอริทึมที่อยู่ในไลบรารี scikit-learn ซึ่งใช้เทคนิค CART (Classification and Regression Tree) โดยพารามิเตอร์ที่สามารถกำหนดได้เพื่อควบคุมความซับซ้อนของ Decision Tree ได้แก่

- max_depth : จำนวนลำดับชั้นที่มากสุดของต้นไม้
- min_samples_leaf : จำนวน Sample ที่ต้องมีเป็นอย่างน้อยที่ leaf node หนึ่ง ๆ
- min_samples_split : จำนวน Sample ที่ต้องมีเป็นอย่างน้อยที่ node หนึ่ง ๆ ที่จะอนุญาตให้แตกกิ่งก้านต่อได้

2.2.2 ทฤษฎีเกี่ยวกับ Random Forest

เทคนิคการทำนายแบบ Random Forest เป็นแบบจำลองการทำนายประเภท Supervised learning ที่มีการสร้างการทำนายในรูปแบบชุดของ Decision Tree หลาย ๆ ต้น (Ensemble of Decision Trees) มาช่วยในการทำนายผลลัพธ์ โดย Decision Tree แต่ละต้นถูกสร้างเป็นอิสระต่อกัน โดยใช้ข้อมูล training data ย่อยที่ถูกสุ่มเลือกจากชุดข้อมูล training data เดิม โดยการสุ่มเลือกเป็นแบบ bootstrap (random sampling with replacement) นั้นคืออาจมีข้อมูลที่ถูกเลือกซ้ำได้

นอกจากนี้ ในแต่ละลำดับของการแตกโหนดของ Decision Tree แต่ละต้น เทคนิค Random Forest จะสุ่มเลือกชุดของ feature ที่จะใช้ในพิจารณาแตกกิ่งก้าน (ไม่ได้พิจารณาจากทุก ๆ feature) ซึ่งจะส่งผลทำให้ tree แต่ละต้นมีลักษณะไม่เหมือนกัน

ในการทำนาย input หนึ่ง ๆ ใช้ผลการทำนายของต้นไม้แต่ละต้นเหล่านี้มาร่วมกันตัดสินใจ สำหรับ Random forest ที่ใช้ในงานวิจัยนี้มาจาก scikit-learn ซึ่งแตกต่างจาก Random forest แบบเดิม [1] คือผลลัพธ์ของการทำนายแบบ Classification มาจากการเฉลี่ยค่าความน่าจะเป็นของต้นไม้แต่ละต้น สำหรับพารามิเตอร์หลักที่ปรับได้ในการสร้างแบบจำลองทำนายแบบ Random Forest ได้แก่

- n_estimators : จำนวนต้นไม้ทั้งหมดในตัวทำนาย ซึ่งโดยทั่วไปยิ่งมีค่ามากยิ่งทำให้การทำนายยิ่งถูกต้อง แต่จะใช้เวลานานขึ้นในการคำนวณ
- max_features : จำนวนของ feature ย่อยที่ถูกสุ่มในการสร้าง Decision Tree แต่ละต้น
- max_depth : จำนวนลำดับชั้นที่มากสุดของต้นไม้แต่ละต้น
- min_samples_leaf : จำนวน Sample ที่ต้องมีเป็นอย่างน้อยที่ leaf node หนึ่ง ๆ

ซึ่งจากการสร้างแบบจำลองแบบสุ่มเลือกข้อมูลและชุดของ feature มาสร้าง tree จะทำให้มีข้อมูลส่วนหนึ่งที่ไม่ถูกเลือกมาใช้งาน ข้อมูลกลุ่มนี้ถูกเรียกว่า Out-of-Bag (OOB) เราสามารถใช้ข้อมูล

OOB นี้ในการประเมินประสิทธิภาพการทำนายของตัวทำนายได้ โดยค่าประสิทธิภาพการทำนายดังกล่าวจะเรียกว่า OOB Score

2.2.3 ทฤษฎีเกี่ยวกับ Gradient Boosting Tree

เทคนิคการทำนายแบบ Gradient Boosting Tree เป็นแบบจำลองการทำนายประเภท Supervised learning ที่ใช้เทคนิค Decision Trees จำนวนหลาย ๆ ต้น (Ensemble of Decision Trees) มาช่วยทำการทำนาย แต่จะแตกต่างจาก Random Forest ตรงที่ Decision Tree แต่ละต้นถูกสร้างแบบตามลำดับ (Sequential) นั้นคือ Decision Tree ตัวใหม่ที่ถูกสร้างเพิ่มเข้ามาจะพยายามแก้ไขผลลัพธ์การทำนายที่ผิดพลาดของ Decision Tree ตัวก่อนหน้า และการทำนายผลลัพธ์จากข้อมูลอินพุตหนึ่ง ๆ จะเกิดจากผลลัพธ์ของต้นไม้ที่ถูกสร้างไว้แต่ละต้นใน Ensembler มารวมกัน

สำหรับการสร้างตัวทำนาย Gradient Boosting Tree มีพารามิเตอร์หลักที่ปรับได้ในการสร้างแบบจำลอง ได้แก่

- n_estimators : จำนวนต้นไม้ทั้งหมดในตัวทำนาย
- max_depth : จำนวนลำดับชั้นของต้นไม้แต่ละต้น
- learning_rate : ค่าที่ใช้ควบคุมระดับความเข้มข้นที่ต้นไม้แต่ละต้นเข้าไปแก้ไขต้นไม้ก่อนหน้านี้
- min_child_weight : ค่าผลรวมของ weight ของแต่ละ sample ที่ต้องมีเป็นอย่างน้อยที่ child (leaf) node หนึ่ง ๆ โดยที่ถ้าแต่ละ sample ของข้อมูลมี weight เท่ากับ 1 ค่านี้จะหมายถึงจำนวน sample ที่ต้องมีเป็นอย่างน้อยที่ child node หนึ่ง ๆ หรือกล่าวอีกอย่างคือจำนวนของ samples ที่ต้องมีเป็นอย่างน้อยที่จะยอมให้แตก node ใหม่

ในงานวิจัยนี้ได้ใช้ไลบรารี XGBoost [4] (eXtreme Gradient Boosting)[2] ซึ่งเป็นไลบรารีสำหรับเทคนิค Gradient Boosting Tree ที่พัฒนาโดย Tianqi Chen เน้นเรื่องความเร็วในการทำงาน (Speed) และเพิ่มประสิทธิภาพ (Performance) ของแบบจำลอง

---

[2] https://github.com/dmlc/xgboost และ Documentation: https://xgboost.readthedocs.io/en/latest/

2.3 ทฤษฎีเกี่ยวกับการวัดประสิทธิผลของการทำนาย

ในงานวิจัยนี้มีตัวชี้วัดที่ใช้ในการวัดประสิทธิผลของการทำนายแบ่งตามปัญหาแบบ Classification และ Regression ซึ่งสามารถอธิบายได้ดังนี้

2.3.1 การวัดความถูกต้องของการทำนายในปัญหาแบบ Classification

การวัดความถูกต้องในการทำนายของปัญหา Classification จะแสดงในรูปของ Confusion Matrix สมมติคือเมทริกซ์ C โดยที่ค่า C[i, j] (แถวที่ i คอลัมน์ที่ j ของเมทริกซ์ C) หมายถึงจำนวน Sample ที่จริง ๆ แล้วเป็นคลาส i แต่การทำนายตอบเป็นคลาส j ดังนั้นจำนวน Sample ที่ทำนายถูกต้องจะแสดงอยู่ในแนว diagonal ของ Confusion Matrix

นอกจากนี้ สามารถคำนวณค่าความถูกต้องในการทำนายในรูปของ ค่า Classification Accuracy ค่า Precision ค่าRecall และค่า F1-Score ซึ่งนิยามได้ดังนี้

ค่า Classification Accuracy สามารถแสดงได้ดังสมการที่ (1)

$$Accuracy = \frac{N_{correct}}{N} \tag{1}$$

โดยที่ $N_{correct}$ คือจำนวน sample ใน Test Set ที่ทำนายถูกต้อง และ N คือจำนวน sample ทั้งหมดใน Test Set

ส่วนค่า Precision ค่า Recall และค่า F1-Score สามารถแสดงได้ดังสมการที่ (2) สมการที่ (3) และ สมการที่ (4) ตามลำดับดังนี้

$$Precision = \frac{TP}{TP+FP} \tag{2}$$

$$Recall = \frac{TP}{TP+FN} \tag{3}$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall} \tag{4}$$

โดยที่ TP คือ จำนวนของ Sample ที่ทำนายถูกต้อง

FP คือ จำนวนของ Sample ที่คำตอบที่ถูกต้องเป็น Negative แต่ทำนายเป็น Positive

FN คือ จำนวนของ Sample ที่คำตอบที่ถูกต้องเป็น Positive แต่ทำนายเป็น Negative

2.3.2 การวัดความถูกต้องของการทำนายในปัญหาแบบ Regression

การวัดความถูกต้องของการทำนายในปัญหาแบบ Regression จะใช้ตัวชี้วัดที่นิยามโดยอ้างอิงจากค่าความแตกต่างระหว่างค่าจริง (Actual value) กับค่าที่อัลกอริทึมทำนายออกมาได้ (Predicted value) โดยทั่วไปตัวชี้วัดที่นิยมใช้มีดังต่อไปนี้

(1) ค่าความคาดเคลื่อนเฉลี่ยกำลังสอง (Root Mean Square Error หรือ RMSE) สามารถนิยามดังสมการที่ (5)

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{5}$$

โดยที่ y_i และ $\hat{y}_i$ คือค่าจริงที่ถูกต้อง และค่าที่ประมาณได้ของอินพุตลำดับที่ i ตามลำดับ และ n คือจำนวนข้อมูลอินพุตในการทดสอบ

(2) ค่าเฉลี่ยความคลาดเคลื่อนสัมบูรณ์ (Mean Absolute Error หรือ MAE) สามารถนิยามดังสมการที่ (6)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{6}$$

โดยที่ y_i และ $\hat{y}_i$ คือค่าจริงที่ถูกต้อง และค่าที่ประมาณได้ของอินพุตลำดับที่ i ตามลำดับ และ n คือจำนวนข้อมูลอินพุตในการทดสอบ

(3) ร้อยละของค่าความคลาดเคลื่อนสัมบูรณ์ (Mean Absolute Percentage Error หรือ MAPE) นิยามเป็นค่าร้อยละของค่าเฉลี่ยของอัตราส่วนระหว่างความคลาดเคลื่อนเทียบกับค่าจริงดังสมการที่ (7)

$$MAPE = \frac{100}{n} \times \sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{7}$$

โดยที่ y_i และ $\hat{y}_i$ คือค่าจริงที่ถูกต้อง และค่าที่ประมาณได้ของอินพุตลำดับที่ i ตามลำดับ และ n คือจำนวนข้อมูลอินพุตในการทดสอบ

(4) ค่า R^2 หรือ coefficient of determination สามารถนิยามดังสมการที่ (8)

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{8}$$

โดยที่ y_iและ $\hat{y}_i$ คือค่าจริงที่ถูกต้อง และค่าที่ประมาณได้ของอินพุตลำดับที่ i ตามลำดับ และ n คือจำนวนข้อมูลอินพุตในการทดสอบ และ $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$

ถ้าค่า R^2 ยิ่งมีค่าเข้าใกล้ 1 หมายความว่าแบบจำลองการทำนายยิ่งมีความแม่นยำ

2.4 การทำนายผลการผลิตแบบดั้งเดิมโดยวิธี Crop Cutting

Crop Cutting [5] คือการประมาณผลผลิตต่อไร่แบบสุ่ม เป็นวิธีการที่โรงงานน้ำตาลส่วนใหญ่ใช้ในการประมาณผลผลิตต่อไร่ที่จะได้ ซึ่งจะมีการสำรวจพื้นที่ปลูกอ้อย และทำการสุ่มตัดอ้อยมาชั่งมาจากหลายพื้นที่ ๆ ทำการสุ่มตัวอย่างไว้ โดยเฉพาะพื้นที่ ๆ โรงงานมีความรับผิดชอบชาวไร่อยู่และทำสัญญาส่งอ้อยเข้าโรงงาน ซึ่งจะทำการวัดพื้นที่แล้วทำการตัดอ้อยมาเพื่อชั่งน้ำหนัก แล้วทำการเฉลี่ยน้ำหนักอ้อยที่ได้ต่อไร่ แต่วิธีนี้ยังมีปัญหาคือไม่ครอบคลุมพื้นที่ทั้งหมดที่โรงงานดูแล และความหนาแน่นของอ้อยแต่ละพื้นที่ไม่เท่ากัน ซึ่งบางปีการผลิตไม่มีการสุ่มตัวอย่างทำ Crop Cutting ทำให้ใช้ค่าเดิมที่มีและอาจไม่เหมาะสมนำมาใช้ในปีการผลิตใหม่ ซึ่งโรงงานน้ำตาลที่ทำการศึกษาข้อมูลนั้นมีค่า Yield เฉลี่ยจากวิธี Crop Cutting ที่ 8 ตันต่อไร่

2.5 งานวิจัยที่เกี่ยวข้อง

การทบทวนวรรณกรรมของงานวิจัยนี้ ได้ทำการศึกษางานวิจัยที่เกี่ยวข้องกับการทำนายผลผลิตทางเกษตรกรรม โดยเน้นการเรียนรู้ของเครื่องด้วยเทคนิคต่าง ๆ มีรายละเอียดดังต่อไปนี้

2.5.1 บทความวิจัยเรื่อง Accurate prediction of sugarcane yield using a random forest algorithm โดย Yvette Everingham และคณะ (2016) [6]

งานวิจัยนี้นำเสนอการทำนายผลผลิตอ้อยในบริเวณเมือง Tully ประเทศออสเตรเลีย โดยใช้ข้อมูลที่เก็บในช่วงปี 1992-2013 โดยทำนายข้อมูลเป็น 3 ช่วงคือช่วงปลูก (เริ่มตั้งแต่วันที่ 1 เดือนตุลาคมของก่อนปีการเก็บเกี่ยว) ช่วงก่อนการเก็บเกี่ยว (เริ่มตั้งแต่วันที่ 1 เดือนมกราคมของปีการเก็บเกี่ยว) และช่วงเก็บเกี่ยว (เริ่มตั้งแต่วันที่ 1 เดือนมีนาคมของปีการเก็บเกี่ยว) โดยใช้ feature ในการทำนายประกอบด้วยรูปแบบการเพาะปลูก สภาพอากาศตามฤดูกาล ปริมาณน้ำฝนสะสม อุณหภูมิสูงสุด อุณหภูมิต่ำสุดในแต่ละช่วงเวลา ปริมาณการแผ่รังสี (radiation) อุณหภูมิพื้นผิวทะเลเฉลี่ย 3 เดือน และดัชนีชีวมวล (biomass index) ที่คำนวณได้จาก Agricultural Production Systems sIMulator (APSIM)

งานวิจัยนี้แบ่งการทำนายออกเป็น 2 ปัญหาคือปัญหา Classification และ Regression โดยปัญหา Classification แบ่งเป็น 2 กลุ่มคือกลุ่มที่มี Yield สูงกว่าและต่ำกว่าค่า Median ของ Yield โดยการวัดความถูกต้องของการทำนายนั้นวัดความถูกต้องบนข้อมูลที่เป็น Out-of-bag (OOB) sample ซึ่งผลลัพธ์ที่ได้จาก Classification ทั้ง 3 ช่วงคือ 86.36%, 95.45% และ 99.45% ตามลำดับ ส่วนผลลัพธ์ที่ได้จาก Regression ทั้ง 3 ช่วงเป็นค่า RMSE คือ 8.00, 7.32 และ 6.33 t/Ha ตามลำดับ

2.5.2 บทความวิจัยเรื่อง Random Forests for Global and Regional Crop Yield Prediction โดย Jig Han Jeong และคณะ (2016) [7]

งานวิจัยนี้นำเสนอการทำนายผลผลิตของพืชไร่ 3 ประเภทได้แก่ ข้าวสาลี (wheat) ข้าวโพด (maize grain) และมันฝรั่ง (potato) โดยใช้ feature หลักมาจากสภาพภูมิอากาศ (Climate) และข้อมูล Biophysics ซึ่ง features ที่นำมาใช้เปรียบเทียบผลผลิตของข้าวสาลีและข้าวโพดคือ Averaged monthly temperature, Annual evapotranspiration, Summer day length, Maximum monthly temperature, Mean coldest quarter temperature, Minimum monthly temperature, Mean warmest quarter temperature, Nitrogen fertilizer application rate, Growing season precipitation, Annual precipitation และ Year ส่วน features ที่นำมาใช้ของมันฝรั่งและข้าวโพดคือ Bulk density (soil), Clay content (soil), Hydraulic conductivity, Average maximum daily temperature, Average minimum daily temperature, Annual precipitation, Averaged seasonal radiation, Saturated water content, irrigation, Latitude และ Elevation

งานวิจัยนี้ใช้เทคนิคการเรียนรู้ด้วยเครื่องด้วยวิธี Random Forests (RF) เปรียบเทียบกับ Multiple Linear Regression (MLR) ผลลัพทธ์การทำนายพบว่าได้ RMSE ที่ได้จาก Random Forest อยู่ในช่วง 6-14% ของค่าเฉลี่ยของ Yield ในขณะที่ MLR ให้ RMSE ในช่วง 14-49%

2.5.3 บทความวิจัยเรื่อง Rice crop yield prediction in India using support vector machines โดย Niketa Gandhi และคณะ (2016) [8]

งานวิจัยนี้ใช้เทคนิคการเรียนรู้ด้วยเครื่องศึกษาการทำนายผลผลิตข้าวจากข้อมูลภายใต้สภาพภูมิอากาศที่แตกต่างกัน โดยใช้เทคนิค SMO Classification และ Support Vector Machines (SVM) ในการทำนาย โดยใช้โปรแกรม WEKA ในการคำนวณ

ข้อมูลที่นำมาใช้ในงานวิจันมาจาก 27 รัฐของประเทศอินเดีย ซึ่งข้อมูลหลักที่นำมาทำนายผลผลิตข้าวคือ อุณหภูมิต่ำสุด-อุณหภูมิสูงสุดและอุณหภูมิเฉลี่ยของพื้นที่ปลูก พื้นที่การผลิตและผลผลิตอ้อยช่วงเดือนมิถุนายนถึงพฤศจิกายน แล้วทำการคำนวณหาค่า Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Relative Absolute Error (RAE) และ Root Relative Squared Error (RRSE) แล้วนำผลที่ได้เปรียบเทียบกัน ซึ่งผลที่ได้คือ SVM มีความถูกต้องในการทำนายสูงกว่า

2.5.4 บทความวิจัยเรื่อง Data mining of agricultural yield data: A comparison of regression models โดย Georg Ruß และคณะ (2009) [9]

งานวิจัยนี้ทำนายผลผลิตพืช (yield prediction) โดยศึกษาเปรียบเทียบผลการทำนายจาก 4 เทคนิค คือ multi-layer perceptions (MLPs), radial basis function network (RBF), Regression Tree และ Support Vector Regression โดยใช้ features หลักดังต่อไปนี้

- ข้อมูลการใช้ปุ๋ย Nitrogen - N1, N2, N3
- Red Edge Inflection Point (REIP) - REIP32, REIP49
- Soil Conductivity – EM38
- Yield

โดยเปรียบเทียบความถูกต้องด้วยค่า RMSE และ MAE ของแต่ละเทคนิค ซึ่งสรุปได้ว่าเทคนิค Support Vector Regression นั้นให้ค่า RMSE และ MAE ที่ดีที่สุด

2.5.5 บทความวิจัยเรื่อง Predictive ability of machine learning methods for massive crop yield prediction โดย Alberto Gonzalez-Sanchez และคณะ (2014) [10]

งานวิจัยนี้กล่าวถึงการทำนายผลผลิตทางการเกษตรด้วยการเรียนรู้ของเครื่อง เนื่องจากปัญหาที่พบมักมาจากข้อมูลไม่เพียงพอ งานวิจัยนี้จึงเปรียบเทียบความถูกต้องของการทำนายกับข้อมูล 10 ชุด โดยเทคนิคที่นำมาใช้คือ Multiple linear regression, M5-Prime regression trees, perceptron multilayer neural networks, Support vector regression และ K-nearest neighbor โดย features ที่ใช้ทำในการทำนายคือ

- PA Planting area (ha)
- IWD Applied irrigation water depth (cm)
- SR Solar radiation (kWh m–2)
- RF Rainfall (mm)
- MaxT Maximum temperature (°C)
- AvgT Average temperature (°C)
- MinT Minimum temperature (°C)
- SDC Season-duration cultivar

โดยวัดความถูกต้องดังต่อไปนี้ Root Mean Square Error (RMS), Root Relative Squared Error (RRSE), Normalized Mean Absolute Error (MAE) และ Correlation Factor (R)

ผลของงานวิจัยพบว่าเทคนิค M5-Prime และ k-nearest neighbor มีค่า RMSE, RRSE, MAE ที่ต่ำใกล้เคียงกัน และมีค่า Correlation ที่สูงพอกัน งานวิจัยนี้จึงสรุปว่า M5-Prime นั้นเหมาะสมกับการทำนายมากที่สุด

2.5.6 บทความวิจัยเรื่อง Attribute selection impact on linear and nonlinear regression models for crop yield prediction โดย Alberto Gonzalez-Sanchez และคณะ (2014) [11]

งานวิจัยนี้เปรียบเทียบเทคนิคการทำนายที่ทำงานร่วมกับอัลกอริทึมการเลือกตัวแปร (Attribute Selection) ที่เพื่อนำมาใช้ในการทำนายผลผลิตพืชไร่ โดย feature หรือ attribute ที่ใช้ในงานวิจัยมีดังนี้

- SP - farm location
- IWD - Irrigation water depth applied (mm)
- SGR Solar radiation (M-Joules/m2)
- RF Rainfall (mm)
- MaxT Maximal temperature (^{o}C)
- MinT Minimal temperature (^{o}C)
- RH Relative humidity in leafs (%)

โดยงานวิจัยนี้สร้างแบบจำลองโดยการใช้ Multiple linear regression , M5 regression trees , artificial neural networks (ANN) และ Stepwise linear regression (SLR) เพื่อทำนายผลผลิตจากข้อมูลที่ทำการคัดเลือก attribute ไว้และเปรียบเทียบค่าความถูกต้องจากค่า Root Relative Squared Error (RRSE), Relative Mean Absolute Error (RMAE) และ correlation factor (R)

โดยผลการศึกษาพบว่าเทคนิค ANN ที่ทำงานกับ attribute ที่อัลกอริทึมเลือกมามีค่า RRSE และ RMAE เฉลี่ยต่ำสุดและค่าเฉลี่ยของ correlation สูงที่สุด

2.5.7 บทความวิจัยเรื่อง From spreadsheets to sugar content modeling: A data mining approach โดย Monique Pires Gravina de Oliveira และคณะ (2017) [12]

งานวิจัยนี้ทำการทำนายผลผลิตน้ำตาลที่ได้จากผลผลิตอ้อย โดยใช้เทคนิคการทำเหมืองข้อมูล (Data Mining) และอัลลอริทึมที่ใช้คือ Support Vector Regression (SVR), Random Forests (RF) และ Regression Trees ร่วมกับการทำ feature Selection ด้วยอัลกอริทึม RReliefF [16] โดยข้อมูลที่นำมาใช้เป็น feature นั้นมาจากข้อมูล 4 ส่วนหลัก ๆดังนี้

- ข้อมูลดิน ได้แก่ Soil, Texture, Fertility, Soil density , Chemical properties , Derived chemical properties

- ข้อมูลอากาศ ได้แก่ Sum of degree days (Sum DD), Accumulated precipitation (Sum Ppt), Mean maximum air temperature (Mean Max Temp), Mean minimum air temperature (Mean Min Temp)

- ข้อมูล Agricultural ได้แก่ Cake Filter , Fertilization rates

- ข้อมูล Crop ได้แก่ Variety , Days cycle , Number of harvests, Total Recoverable Sugar

มีการทำ parameter tuning ด้วยใช้วิธี grid search จนได้แบบจำลองที่ให้ค่าความถูกต้องที่ดีที่สุด โดยใช้ซอฟท์แวร์ด้านสถิติคือ R, version 3.1.1 ซึ่งงานวิจัยนี้เน้นการทำ feature Selection ด้วยอัลกอริทึม RReliefF เพื่อหาความสัมพันธ์ของตัวแปรที่มีผลกับการทำนายและตัวแปรที่เป็น bias กับแบบจำลอง

ซึ่งผลการทำนายของงานวิจัยนี้คือเทคนิค Random Forest ทำงานร่วมกับ RReliefF ได้ความถูกต้องกว่าเทคนิคอื่น ๆ ซึ่งมีค่า MAE เท่ากับ 2.02 kg/Mg

2.5.8 บทความวิจัยเรื่อง The effect of tuning, feature engineering, and feature selection in data mining applied to rainfed sugarcane yield modelling โดย Felipe F. Bocca และคณะ (2016) [13]

งานวิจัยนี้เป็นการทำนายผลผลิตอ้อย โดยศึกษาผลจากการทำ Feature Selection, Hyper Parameter Tuning และ Feature Engineering มีผลต่อความถูกต้องของการทำนาย โดยงานวิจัยนี้ใช้เทคนิคการเรียนรู้ของเครื่องทั้งหมด 5 เทคนิค ได้แก่ Support Vector Machine (SVM), Random Forest (RF), Regression Tree (RT), Neural Network (NN) และ Boosted Regression Trees (BRT) เพื่อนำมาเปรียบเทียบผลการทำนายที่ได้

นอกจากนี้ งานวิจัยได้นำเสนอเทคนิคการทำ feature engineering ด้วยการคำนวณค่าจาก feature เดิมให้เป็น feature ใหม่เพื่อนำมาใช้ทำนายโดยการคำนวณอัตราการใช้ปุ๋ยแต่ละชนิดและแยก feature เกี่ยวกับข้อมูลอากาศ มีการใช้อัลกอริธึม RReliefF [16] ในการทำ feature selection และหาค่า feature importance ของแต่ละ feature โดยมีการทำ Hyper Parameter Tuning ด้วย Grid Search

จากการศึกษาของงานวิจัยนี้สรุปได้ว่าค่า Mean Absolute Error (MAE) จากเทคนิคต่าง ๆ อยู่ระหว่าง 4.57 - 8.80 Mg ha^{-1} เมื่อทำ Tuning เหลือ 1.17 Mg ha^{-1} และเมื่อทำ Feature Engineering สามารถลด MAE เหลือ 0.64 Mg ha^{-1} ซึ่งแบบจำลองการทำนายมีผลดีขึ้นถึง 40%

2.5.9 บทความวิจัยเรื่อง Harvest scheduling algorithm to equalize supplier benefits: A case study from the Thai sugar cane industry โดย Thuankaewsing, Surached และคณะ (2016) [14]

งานวิจัยนี้นำเสนอเรื่องการจัดตารางการเก็บเกี่ยว (Harvest Scheduling) ของโรงงานน้ำตาลในภาคตะวันออกเฉียงเหนือในประเทศไทย โดยมีส่วนของการทำนายผลผลิตอ้อยเพื่อใช้ในการจัดตารางเวลาการเก็บเกี่ยว ซึ่งในงานวิจัยนี้ใช้เทคนิค neural network ในการทำนาย โดย Feature ที่ใช้ในการทำนายได้แก่

- ประเภทอ้อย
- พันธุ์อ้อย
- ประเภทดิน
- วิธีการให้น้ำ
- วันที่ปลูกและวันที่เก็บเกี่ยว
- อายุของอ้อยที่จะทำการเก็บเกี่ยว
- ปริมาณน้ำฝนเฉลี่ยต่อวันและปริมาณน้ำฝนสะสม
- อุณหภูมิเฉลี่ยน้อยสุดและอุณหภูมิเฉลี่ยมากสุด
- ความสามารถในการปลูกอ้อยของชาวไร่ซึ่งจัดเป็น 3 ระดับคือ A, B และ C

จากงานวิจัยพบว่าผลลัพธ์การทำนายได้ MAPE มีค่าประมาณ 10.98%

2.5.10 วิทยานิพนธ์เรื่อง การสร้างแบบจำลองนิวโร-ฟัซซี่ เพื่อการประมาณการผลผลิตอ้อย โดย ดิลก ภิญโญศรี (2552) [15]

งานวิจัยนี้นำเสนอการประมาณผลผลิตอ้อยโดยใช้ Adaptive Neuro-Fuzzy Inference systems (ANFIS) และ General Regression Neural Network (GRNN) โดยใช้แปลงอ้อยตัวอย่างทั้งหมดจำนวน 1,446 แปลง ใช้เทคนิค Decision Tree และ k-means มาจัดกลุ่มข้อมูลเป็น 5 กลุ่ม ซึ่ง features หลักที่นำมาใช้คือ

- ค่าความเป็นกรดด่าง (pH)
- ปริมาณไนโตรเจน (N)
- ปริมาณฟอสฟอรัส (P)
- ปริมาณโพแทสเซียม (K)
- ปริมาณน้ำฝน

โดยผลการทำนายเมื่อทำการเปรียบเทียบค่า RMSE และ MAPE แล้วสรุปได้ว่าเทคนิค GRNN ให้ผลลัพธ์ค่าความถูกต้องมากกว่า ANFIS และเมื่อจัดกลุ่มโดยเทคนิค k-means ร่วมด้วยยิ่งทำให้ตัวแบบที่สร้างด้วยเทคนิค GRNN ยิ่งมีประสิทธิภาพมากขึ้น โดยค่า RMSE และ MAPE มีค่าลดลง

2.5.11 บทความวิจัยเรื่อง Feature selection for wheat yield prediction โดย Georg Ruß และ Rudolf Kruse (2010) [17]

งานวิจัยนี้ทำนายผลผลิตข้าวสาลี โดยผู้วิจัยเน้นการใช้เทคนิค Forward Feature Selection มาใช้ในการคัดเลือก feature ซึ่งการทำนายผลผลิตเป็นแบบ Regression ใช้เทคนิค Regression Tree และ Support Vector Regression ในการทำนาย โดย feature ที่นำมาใช้คือ

- Nitrogen Fertilizer – N1 , N2 , N3
- Vegetation – REIP32 , REIP49
- Electric Conductivity – EM38
- Yield
- TRACFORCE

โดยผู้วิจัยสรุปว่าการนำ Feature selection มาใช้ร่วมกับการทำ Regression ทั้งเทคนิค Regression Tree และ เทคนิค Support Vector Regression นั้นทำให้ความผิดพลาด (error) จากการทำนายลดลงทั้งคู่

2.5.12 บทความวิจัยเรื่อง Attribute selection impact on linear and nonlinear regression models for crop yield prediction โดย GONZALEZ-SANCHEZ, Alberto และคณะ (2014) [18]

งานวิจัยนี้ใช้หลายเทคนิคในการทำนายผลผลิตพืชเพื่อเปรียบเทียบค่าความถูกต้องที่ได้ โดยเทคนิคที่ใช้คือ Multiple linear regression, Stepwise linear regression, M5- Regression trees และ Artificial Neural Networks (ANN) เพื่อหาแบบจำลองในการทำนายที่ดีที่สุด

โดยข้อมูลที่นำมาใช้มาจากสองส่วนคือข้อมูลด้านเกษตรกรรและข้อมูลเกี่ยวกับสภาพอากาศ ซึ่งข้อมูลเกี่ยวกับเกษตรกรรมนั้นได้มาจากระบบ Spriter-GIS ที่ทำการจัดเก็บข้อมูลพื้นที่การปลูก, ชนิดพืช, วิธีการให้น้ำ, วิธีการเก็บผลผลิต, วันที่เริ่มต้นและสิ้นสุดการปลูก, ผลผลิตที่ได้ ส่วนข้อมูลสภาพอากาศได้แก่ ปริมาณฝน, ค่ารังสีจากดวงอาทิตย์และอุณหภูมิ นั้นมาจาก National Meteorological Service (SMN)

โดยงานวิจัยนี้เปรียบเทียบความถูกต้องจากสามค่าคือ Root Relative Squared Error (RRSE), Relative Mean Absolute Error (RMAE) และ correlation ซึ่งผลที่ได้คือเทคนิค ANN ให้ค่าความถูกต้องที่ดีที่สุดมีค่า RRSE เท่ากับ 86.04% , RMAE เท่ากับ 8.75% และ correlation คือ 0.63

บทที่ 3
ระเบียบวิธีการดำเนินงานวิจัย

จากการทบทวนทฤษฎีและผลงานวิจัยที่เกี่ยวข้องนั้น ผู้วิจัยได้กำหนดระเบียบวิธีการดำเนินงานวิจัย เพื่อใช้สร้างแบบทำนายผลผลิตอ้อยที่เหมาะสมและสามารถวัดประสิทธิผลการทำนายได้ โดยมีขั้นตอนของการดำเนินงานวิจัยดังนี้

(1) การเก็บรวบรวมข้อมูล (Data Collection) เพื่อใช้เป็นข้อมูลสร้างแบบจำลองในการทำนายผลผลิตอ้อย

(2) การตรวจสอบความถูกต้องของข้อมูลและจัดรูปแบบข้อมูลเพื่อใช้ในการสร้างแบบจำลองในการทำนายผลผลิตอ้อย (Data Preprocessing)

(3) การสร้างแบบจำลองในการทำนายปริมาณอ้อย โดยใช้เทคนิคของการเรียนรู้ของเครื่องแบบต่าง ๆ

(4) เปรียบเทียบและวัดประสิทธิภาพความถูกต้องของข้อมูลที่ได้จากการทำนาย

(5) อภิปราย สรุปผลงานวิจัยและข้อเสนอแนะ

ในบทนี้ ผู้วิจัยอธิบายขั้นตอนการเก็บรวบรวมข้อมูล (Data Collection) ในหัวข้อ 3.1 และอธิบายขั้นตอนการทำ Data preprocessing ในหัวข้อ 3.2 หลังจากนั้นอธิบายการสร้างแบบจำลองการทำนายปริมาณผลผลิตอ้อยเบื้องต้นในหัวข้อ 3.3

ผู้วิจัยจะอธิบายรายละเอียดของการพัฒนาระบบการทำนายเกรดของผลผลิตอ้อย (Yield grade prediction) ในบทที่ 4 และอธิบายรายละเอียดของการพัฒนาระบบการทำนายปริมาณผลผลิตอ้อย (Yield value prediction) ในบทที่ 5 หลังจากนั้นจะอธิบายสรุปผลงานวิจัยและข้อเสนอแนะในบทที่ 6

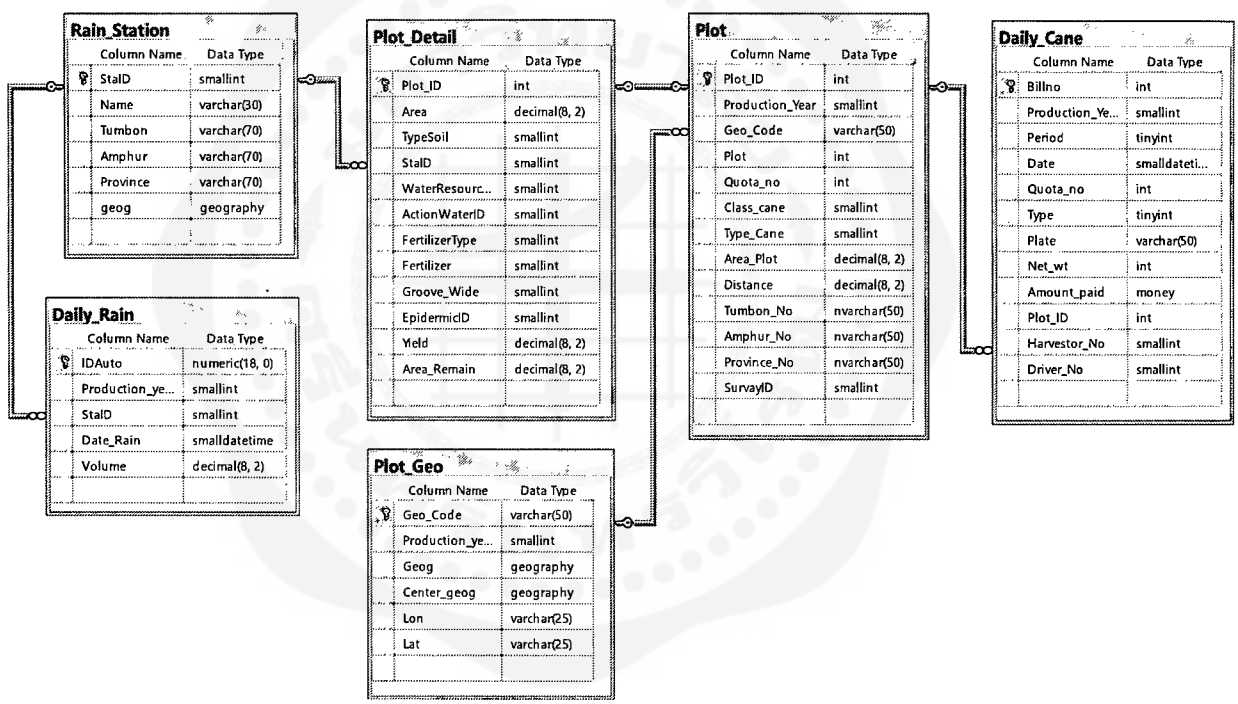
3.1 การเก็บรวบรวมข้อมูล (Data Collection)

ข้อมูลที่นำมาใช้งานวิจัยนี้ถูกเก็บรวมรวมโดยใช้โปรแกรมซอฟต์แวร์ที่ทางโรงงานน้ำตาลที่เป็นกรณีศึกษานั้นได้พัฒนาขึ้นเอง โดยนำมาจากสองระบบซึ่งเชื่อมโยงกันด้วยสองโปรแกรม มีรายละเอียดดังต่อไปนี้

3.1.1 โปรแกรมมีชื่อว่า GIS-System ซึ่งเป็นโปรแกรมที่เก็บข้อมูลของแปลงอ้อยจากการที่พนักงานฝ่ายสำรวจออกไปสำรวจไร่ ทำการเก็บพิกัดโดยระบบ GPS และเก็บข้อมูลการปลูกอ้อยของแปลงที่ทำการสำรวจ แล้วทำการบันทึกข้อมูลลงในโปรแกรม

3.1.2 โปรแกรมมีชื่อว่า Cane-Accounting เป็นโปรแกรมที่เก็บข้อมูลการส่งอ้อยเข้ามาโรงงานของชาวไร่ ซึ่งมีการบันทึกว่าอ้อยที่ส่งเข้ามานั้นเป็นของแปลงอ้อยเลขที่ใด ทำให้สามารถทราบได้ว่าแปลงอ้อยที่ทำการสำรวจนั้นมียอดส่งอ้อยได้ตามที่ประเมินไว้หรือไม่

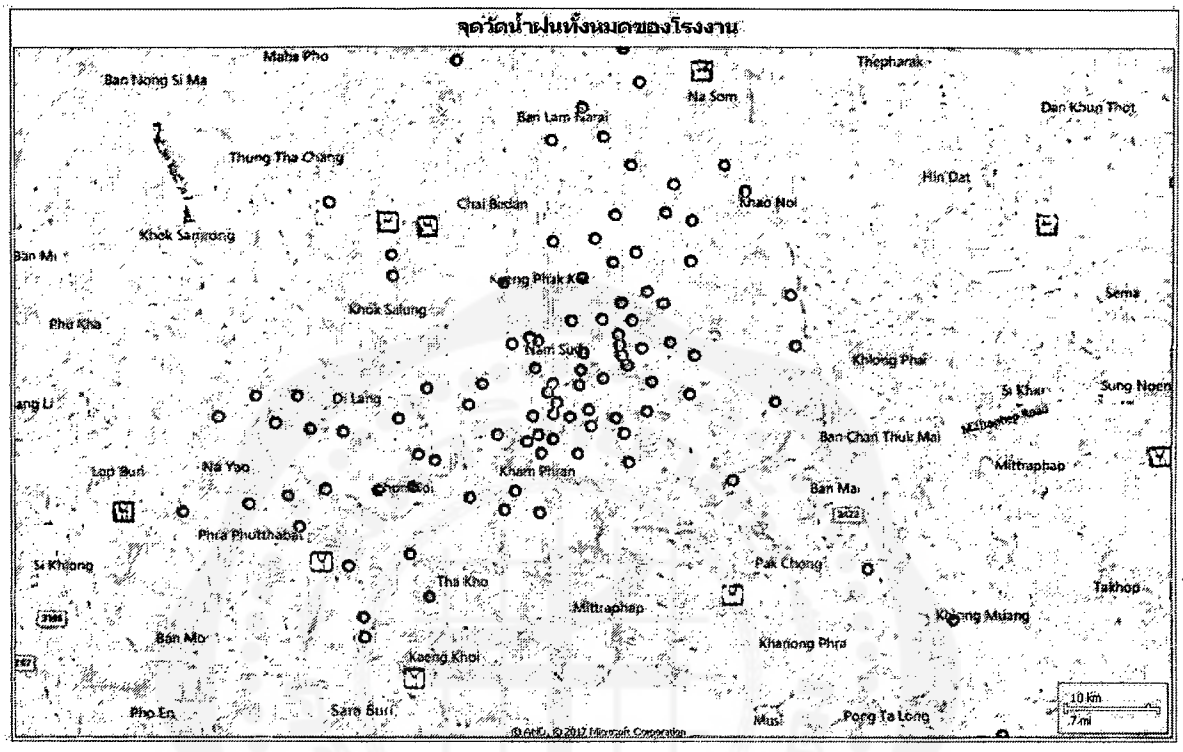
ซึ่งทั้งสองโปรแกรมมีฐานข้อมูลที่ใช้ในการเก็บข้อมูลของทั้งสองโปรแกรมคือ Microsoft SQL Server 2016 โดยโครงสร้างของฐานข้อมูลบางส่วนแสดงได้ดัง E-R Diagram ดังภาพประกอบ 2



ภาพประกอบ 2 E-R Diagram บางส่วนของโปรแกรม GIS-System และ Cane-Accounting

A433504

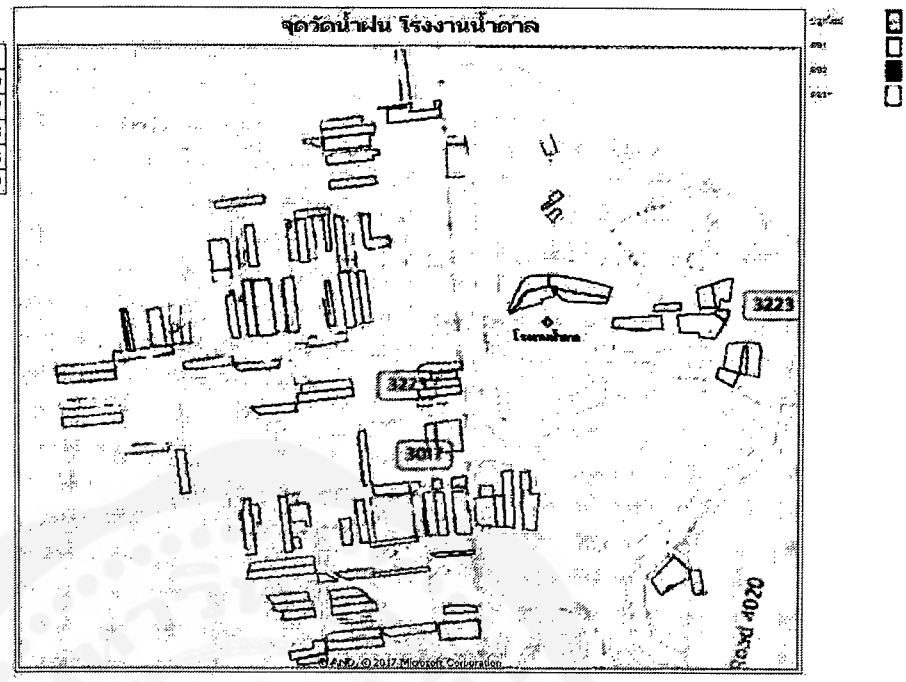
สำหรับข้อมูลน้ำฝนนั้นได้มาจากการติดตั้งจุดวัดน้ำฝนของโรงงานน้ำตาลทั้งหมด 100 จุด ครอบคลุมพื้นที่แปลงอ้อยที่โรงงานรับผิดชอบชาวไร่ที่มีแปลงอยู่ แสดงแผนที่ตั้งได้ดังภาพประกอบ 3



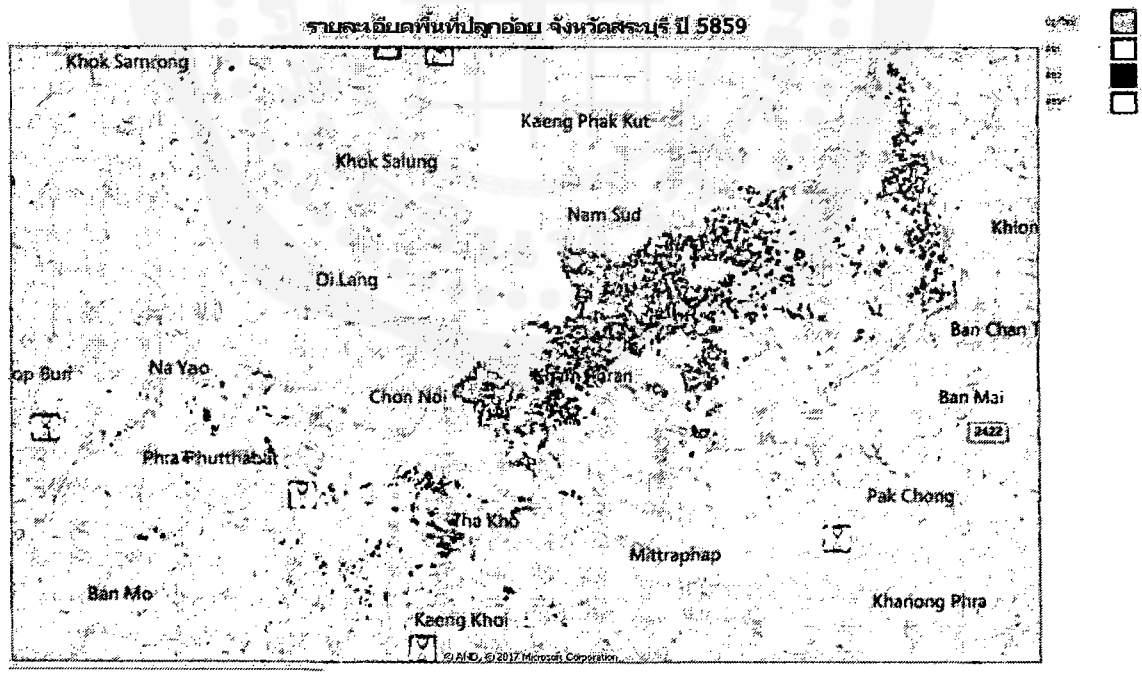
ภาพประกอบ 3 แผนที่แสดงจุดวัดน้ำฝนทั้งหมดของโรงงานน้ำตาล

แต่ละแปลงอ้อยจะถูกคำนวณระยะทางจากแปลงอ้อยถึงจุดวัดน้ำฝนที่ใกล้ที่สุด โดยใช้ข้อมูลเชิง spatial ที่เก็บจากการวัด GPS นอกจากนี้ข้อมูลที่อยู่ในฐานข้อมูลสามารถนำมาแสดงตัวอย่างแปลงอ้อยที่อยู่รอบจุดวัดน้ำฝนที่ทำการเก็บข้อมูลไว้ได้ แสดงได้ดังภาพประกอบ 4 และแสดงตัวอย่างแปลงอ้อยที่โรงงานน้ำตาลทำการสำรวจข้อมูล แสดงได้ดังภาพประกอบ 5

ตำบล	ชนิดอ้อย	พื้นที่ปลูก	จำนวน
[illegible]	[illegible]	764.57	39
	[illegible]	661.01	37
	[illegible]	571.77	28
	[illegible]	478.50	25
รวม		2477.85	129



ภาพประกอบ 4 แผนที่แสดงแปลงอ้อยรอบจุดวัดน้ำฝน



ภาพประกอบ 5 แผนที่แปลงอ้อยของโรงงานน้ำตาล

จากที่กล่าวมาจึงทำการ Query ข้อมูลจากตารางที่มีความสัมพันธ์กันออกมา ในงานวิจัยนี้ข้อมูลที่ใช้ในการทดลองที่ได้จากการรวมข้อมูลจากสองโปรแกรมมีทั้งหมด 12,521 รายการ ซึ่งมาจากข้อมูลจากปีการผลิต 2558/2559 และข้อมูลจากปีการผลิต 2559/2560 สามารถสรุปรายละเอียดจำนวนข้อมูลได้ดังนี้

1) ข้อมูลแปลงอ้อยที่มีการส่งอ้อยเข้าโรงงานของปีการผลิต 2558/2559 มีทั้งหมด 7,451 รายการ

2) ข้อมูลแปลงอ้อยที่มีการส่งอ้อยเข้าโรงงานของปีการผลิต 2559/2560 มีทั้งหมด 5,070 รายการ

ซึ่งข้อมูลการจัดเกรดของปริมาณอ้อยและยอดประเมินอ้อยต่อตันที่ระบุในฟิลด์ TargetGrade, YieldGrade, TargetOldGrade และ YieldOldGrade ใช้เกณฑ์จากการทำ Crop Cutting และโรงงานน้ำตาลที่เป็นกรณีศึกษาใช้กำหนดเป็นเกณฑ์ในการทำงาน แสดงรายละเอียดได้ดังตารางที่ 1 ดังนี้

ตาราง 1 รายละเอียดของการแบ่งเกรดของปริมาณอ้อยต่อไร่

เกรด	รายละเอียด
เกรด 1	ผลผลิตต่ำกว่า 7 ตันต่อไร่
เกรด 2	ผลผลิตอยู่ระหว่าง 7-12 ตันต่อไร่
เกรด 3	ผลผลิตมากกว่า 12 ตันต่อไร่

โดยข้อมูลที่ได้นั้นสามารถแสดงรายละเอียดของแต่ละฟิลด์และสามารถระบุประเภทของข้อมูลว่าเป็น Category หรือ Continuous แสดงได้ดังตารางที่ 2 และแสดงตัวอย่างของข้อมูลได้ดังตารางที่ 3

ตาราง 2 รายละเอียดของข้อมูลที่นำมาใช้ในงานวิจัย

ชื่อ	ประเภท	รายละเอียด
Class_cane	Category	ประเภทอ้อยที่ปลูก คือ อ้อยปลูกใหม่, อ้อยตอ1, อ้อยตอ2 และอ้อยตอ3 ขึ้นไป
Type_Cane	Category	พันธุ์อ้อยที่มี คือ LK92-11, K84-200, K99-72 และ ขอนแก่น3
WaterType	Category	แหล่งน้ำที่มี คือ น้ำฝน, คลองธรรมชาติและชลประทาน
ActionWater	Category	วิธีการให้น้ำ คือ น้ำฝนและน้ำราด
Epidemic	Category	วิธีการกำจัดวัชพืช คือ ใช้ยาคุมและใช้ยาฆ่า
FertilizerType	Category	ประเภทปุ๋ยที่ให้ คือ ปุ๋ยเคมีและปุ๋ยอินทรีย์
Fertilizer	Category	สูตรปุ๋ยที่ให้ คือ 46-0-0, 15-15-15, 16-16-16 และ 25-7-7
TypeSoil	Category	ชนิดดินที่ปลูก คือ ดินร่วน, ดินดำและดินแดง
GrooveWide	Category	ระยะห่างของร่อง คือ 120, 130, 140 และ 150 cm.
TargetGrade	Category	เกรดของปริมาณอ้อยในแต่ละแปลงจากการทำนายของพนักงาน
TargetOldGrade	Category	เกรดของปริมาณอ้อยในแต่ละแปลงจากการทำนายของพนักงานจากฤดูกาลก่อนหน้า
YieldGrade	Category	เกรดของปริมาณอ้อยในแต่ละแปลงในฤดูการผลิตนั้น
YieldOldGrade	Category	เกรดของปริมาณอ้อยในแต่ละแปลงจากฤดูกาลก่อนหน้า
FarmerContract Grade	Category	การจัดเกรดชาวไร่จากแผนกการเงินโดยใช้ข้อมูลการทำสัญญาส่งอ้อยและความสามารถในการคืนเงินจากฤดูกาลก่อนหน้า
Area_Remain	Continuous	ขนาดพื้นที่คงเหลือของแต่ละแปลงอ้อย (ไร่)
Rain_Vol	Continuous	ปริมาณอ้อยสะสมในฤดูการผลิตนั้น (มิลลิเมตร)
Distance	Continuous	ระยะทางจากแปลงอ้อยถึงโรงงานน้ำตาล
ContractsArea	Continuous	สัญญาการส่งอ้อยของชาวไร่ในฤดูการผลิตนั้น (ตันต่อไร่)
YieldOld	Continuous	ปริมาณอ้อยที่ส่งในแต่ละแปลงจากฤดูกาลก่อนหน้า (ตัน)
YieldTarget	Continuous	ปริมาณอ้อยในแต่ละแปลงจากการทำนายของพนักงานในฤดูการผลิตนั้น (ตัน)
YieldTargetOld	Continuous	ปริมาณอ้อยในแต่ละแปลงจากการทำนายของพนักงานในฤดูการก่อนหน้า (ตัน)
Yield	Continuous	ปริมาณอ้อยทั้งหมดที่ส่งในแต่ละแปลงในฤดูการผลิตนั้น (ตัน)

ตาราง 3 แสดงตัวอย่างของข้อมูล

Class Cane	Type Cane	Distance	Rain Vol	Type Soil	Water Type	Action Water Type	Fertilizer Type	Fertilizer	Epidemic Type	Groove Wide	Area Remain	Yield Target	Yield	Yield Old	Yield Target Old	Yield Grade	Yield Old Grade	Target Grade	Target Old Grade	Contracts Area	Farmer Contract Grade
4	1	34.3	752.5	3	1	5	1	6	2	130	56.7	8	1.52	9.53	9	1	2	2	2	38.84	1
2	14	40.85	791	3	1	1	1	6	2	140	4.31	8	6.23	5.8	9	1	1	2	2	80.97	3
4	14	41.04	791	3	3	1	1	6	2	140	2.57	9	10.66	8.98	9	2	2	2	2	135.8	3
1	14	41.15	791	3	1	5	1	6	2	140	11.39	8	6	7.94	7	1	2	2	2	30.64	3
1	14	41.1	791	3	1	1	1	6	2	140	2.66	8	12.56	8.8	9	3	2	2	2	131.2	3
4	14	40.94	791	3	1	1	1	6	2	140	4.66	9	5.75	5.48	7	1	1	2	2	74.89	3
2	14	41.63	791	13	3	1	1	6	2	140	13.88	9	8.29	7.72	9	2	2	2	2	71.11	3
2	14	39.74	791	13	1	1	1	6	2	140	4.03	9	12.39	10.22	8	3	2	2	2	244.91	3
4	1	40.99	791	13	1	5	1	6	2	140	19.34	9	7.49	10.59	8	2	2	2	2	51.03	3
3	1	40.18	791	13	1	1	1	6	2	140	16.28	9	11.73	8.24	7	2	2	2	2	60.63	3
1	14	39.36	791	3	1	5	1	6	2	140	9.36	9	9.18	6.98	7	2	1	2	2	37.29	3
4	14	40.18	791	3	1	1	1	6	2	140	8.71	9	6.4	8.64	9	1	2	2	2	40.07	3
3	1	42.92	791	13	1	1	1	6	2	140	17.61	9	9.06	11.14	9	2	2	2	2	56.05	3
1	1	41.19	791	3	1	5	1	6	2	140	9.94	9	7.82	7.56	7	2	2	2	2	99.3	3
1	14	41.26	791	13	1	5	1	6	2	140	32.04	9	8.83	5.03	6	2	1	2	1	30.81	3
3	1	41.43	791	13	1	5	1	6	2	120	15.81	10	9.1	14.69	8	2	3	2	2	16.7	3

3.2 การประมวลผลข้อมูลเบื้องต้น (Data Preprocessing)

จากข้อมูลที่ได้มานั้นต้องนำมาตรวจสอบข้อมูลก่อนนำไปใช้งาน เมื่อนำมาตรวจสอบพบว่ามีข้อมูลที่บางฟิลด์หรือ feature ในข้อมูลที่ขาดหายไป (missing value) โดยจะปรากฏเป็นค่า Null โดยผู้วิจัยพบว่าสาเหตุหลักของข้อมูลที่ขาดหายไป เกิดมาจากการป้อนข้อมูลที่ไม่ถูกต้อง

ผู้วิจัยได้แก้ปัญหาดังกล่าว โดยกำหนดมีการทำขั้นตอน data cleaning โดยการแทนค่าข้อมูลที่ขาดหายไปที่เป็นมี feature เป็นประเภท Continuous ด้วยค่ากลางของข้อมูลใน feature นั้นและแทนข้อมูลที่ขาดหายไปที่ใน feature ประเภท Category ด้วยค่า Mode ของแต่ละ feature นั้น ๆ โดยตารางที่ 4 แสดงค่า Median และค่า Mode ของข้อมูลในส่วนของ feature เป็นชนิด Category

ตาราง 4 ค่า Median และค่า Mode ของข้อมูลที่เป็นชนิด Category

Features	Median	Mode
Class_Cane	2	2
Type_Cane	1	1
TypeSoil	3	3
WaterType	1	1
ActionWaterType	5	5
FertilizerType	1	1
Fertilizer	6	6
EpidemicType	2	2
GrooveWide	130	130

โดยผู้วิจัยพบว่ามี 3 feature ที่มี missing values คือ Type_Cane มี 41 รายการ, FertilizerType มี 45 รายการ และ Fertilizer มี 43 รายการและส่วนย่อยของโค้ด (code snippet) ภาษา Python ที่จัดการกับปัญหา missing value นี้ สามารถแสดงได้ดังนี้

```
from scipy.stats import mode
#Impute the values:
data['Type_Cane'].fillna(mode(data['Type_Cane']).mode[0],
inplace=True)
data['FertilizerType'].fillna(mode(data['FertilizerType']).mode[0],
                                    inplace=True)
data['Fertilizer'].fillna(mode(data['Fertilizer']).mode[0],
inplace=True)
```

จากข้อมูลที่ผ่านขั้นตอน Data cleaning จะนำข้อมูลมาผ่านขั้นตอน Data transformation โดยทำการแปลงข้อมูลใน features ชนิด Category โดยใช้เทคนิค one-hot-encoding เพื่อให้ได้ dummy variables (ใช้ฟังก์ชัน get_dummies() ของ pandas) เพื่อนำไปใช้กับการทำ regression เพื่อทำนายค่า yield ที่เป็นการทำนายเชิงปริมาณ ส่วนย่อยของโค้ด (code snippet) ของขั้นตอนนี้ สามารถแสดงได้ดังนี้

```
X_dummy=pd.get_dummies(X,columns=['ActionWaterType', 'Class_Cane',
        'EpidemicType','Fertilizer','FertilizerType','GrooveWide',
        'TypeSoil','Type_Cane','WaterType','YieldOldGrade',
        'TargetGrade','TargetOldGrade','FarmerContractGrade'])
```

หลังจากการประมวลผลข้อมูลดังคำอธิบายข้างต้น ข้อมูลจะถูกแบ่งเป็น Training dataset และ Test dataset ในอัตราส่วนคือ 70% ต่อ 30% โดยใช้คำสั่งใน scikit-learn คือ `train_test_split` ในการทำ และสร้างไฟล์ CSV เพื่อนำไปใช้งานวิจัยต่อไป ส่วนย่อยของโค้ด (code snippet) ของขั้นตอนนี้ สามารถได้ดังนี้

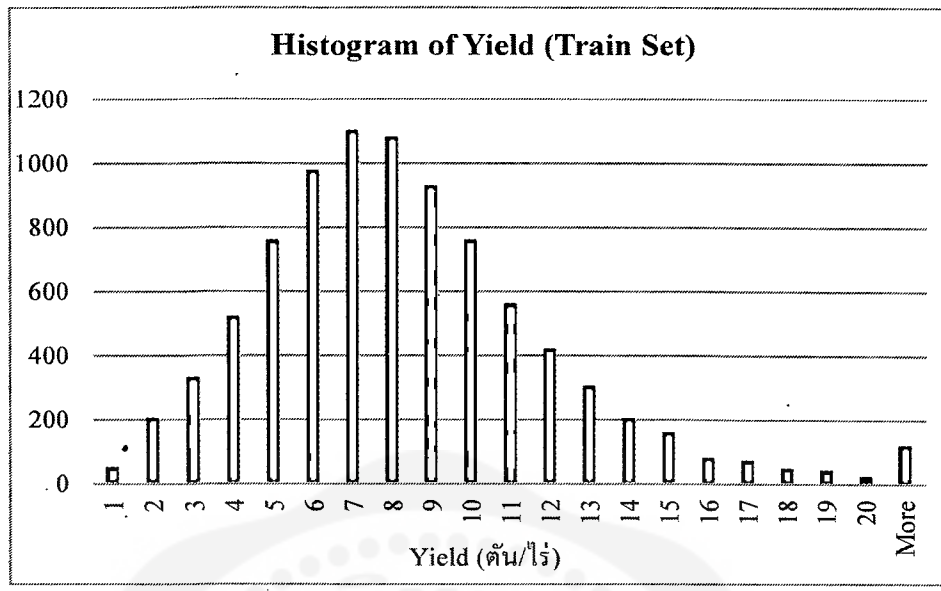
```
X_train,X_test, y_train, y_test = train_test_split(DataAllDummy, y,
                                    test_size=0.30, random_state=0)
X_train.to_csv('TrainAll.csv', sep=',')
X_test.to_csv('TestAll.csv', sep=',')
```

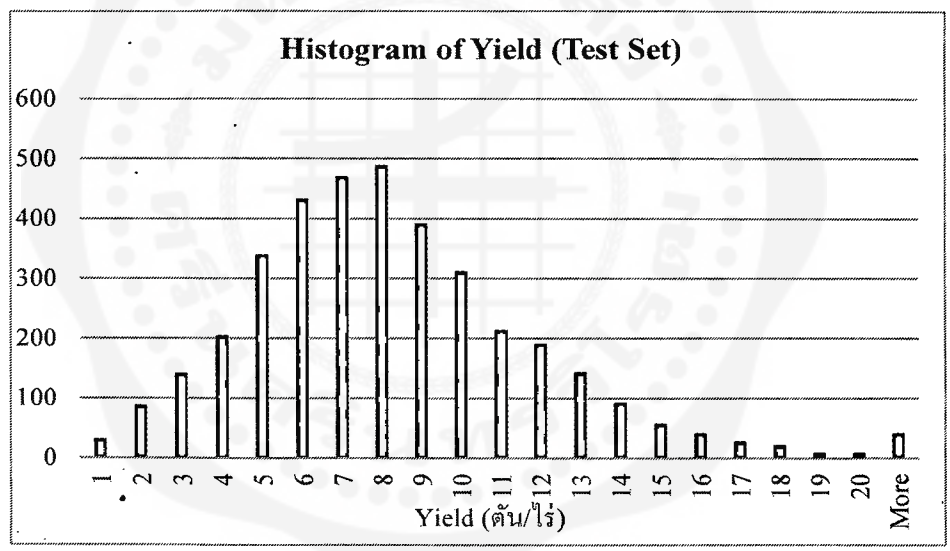
ในงานวิจัยนี้ เราจะทำการพัฒนาแบบจำลองการทำนายในวิธีต่าง ๆ รวมทั้งขั้นตอน Feature selection และขั้นตอน Hyper-parameter tunning โดยใช้ Training dataset หลังจากนั้นเมื่อได้แบบจำลองมาแล้ว เราจะนำมาวัดประสิทธิผลในการทำนายบน Test dataset ถ้ากล่าวโดยสรุปผู้วิจัยพบว่าสัดส่วนของจำนวนข้อมูล Training data และ Test data ที่แบ่งได้คือ 8,765 รายการและ3,756 รายการตามลำดับ ซึ่งสามารถแสดงสรุปได้ดังตารางที่ 5

ตาราง 5 จำนวนข้อมูลของ Training dataset และ Test dataset

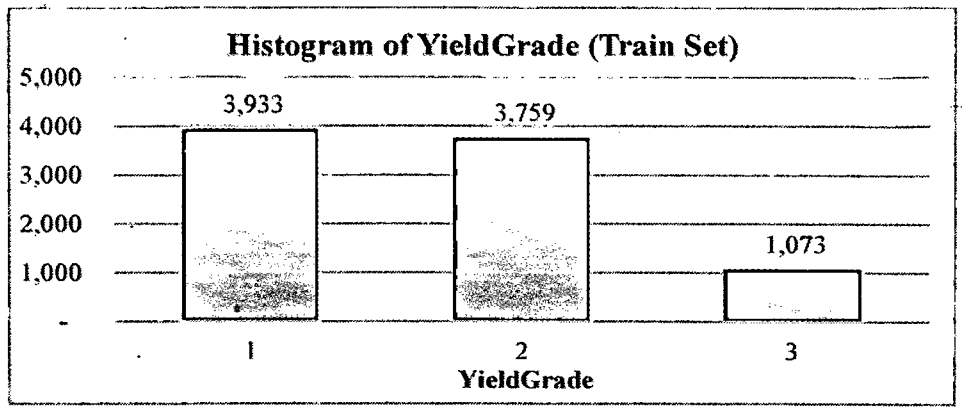
รายละเอียด	จำนวน (รายการ)
ข้อมูลใน Training dataset	8,765
ข้อมูลใน Training dataset	8,765

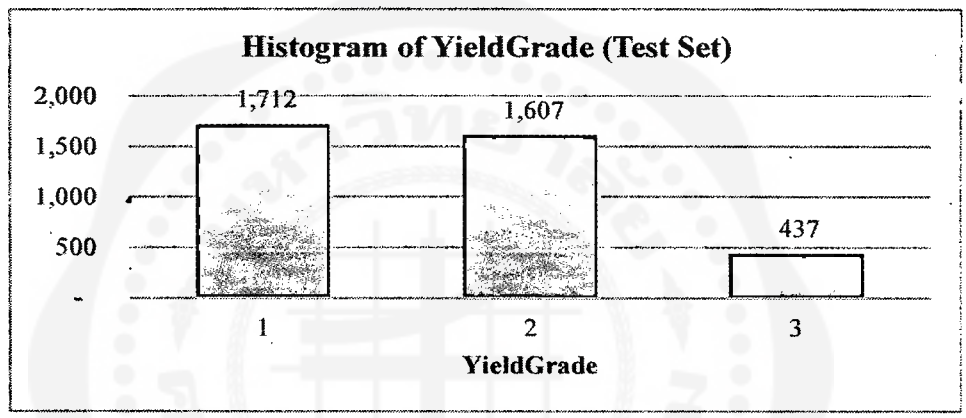
โดยการกระจายของข้อมูลของใน Training dataset และ Test dataset สามารถแสดงในรูปของกราฟ histogram ของ Yield ได้ดังภาพประกอบ 6 และกราฟ histogram ของ YieldGrade ได้ดังภาพประกอบ 7 ตามลำดับ ซึ่งจากกราฟทั้งสองจะเห็นได้ว่าการกระจายของข้อมูล Yield และ YieldGrade บน Training dataset และบน Test dataset มีลักษณะคล้ายกัน ซึ่งหมายความว่าแบบจำลองที่พัฒนาจากข้อมูลใน Training dataset น่าจะนำไปใช้ในการทำนายข้อมูลใน Test dataset เนื่องจากมีค่าเป้าหมาย (label) ที่การกระจายอยู่ในชุดข้อมูลทั้งสองคล้าย ๆ กัน โดยเมื่อเรานำ distribution ของ YieldGrade จาก Train Set และ Test Set มาแสดงคู่กัน ดังภาพประกอบ 8 ยิ่งแสดงให้เห็นได้ชัดเจนมากขึ้นว่า YieldGrade ของ Train และ Test dataset นั้นมีการกระจายตัวคล้ายกัน



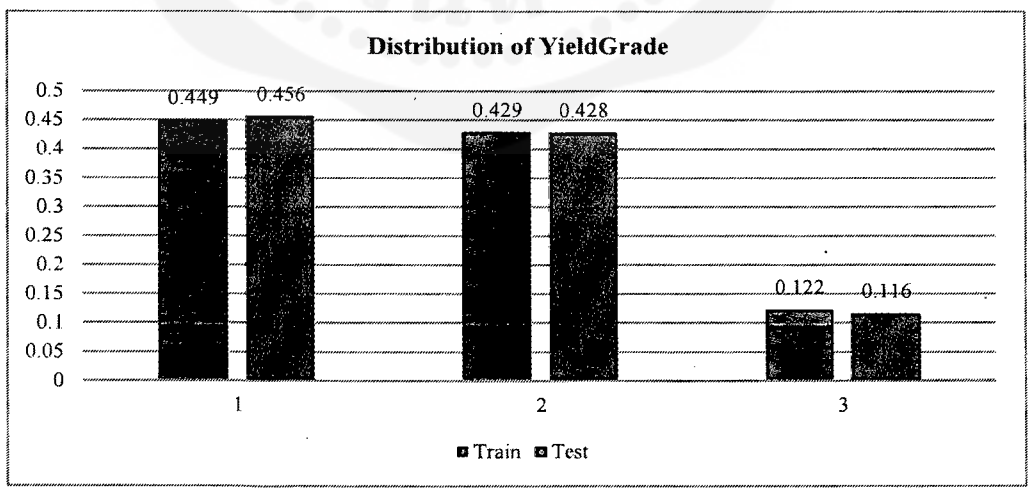


ภาพประกอบ 6 Histogram แสดงการกระจายของข้อมูล Yield





ภาพประกอบ 7 Histogram แสดงการกระจายของข้อมูล YieldGrade



ภาพประกอบ 8 Histograms แสดง distribution ของ YieldGrade จาก Train Set และ Test Set

3.3 การสร้างแบบจำลองการทำนายปริมาณผลผลิตอ้อยเบื้องต้น

จากข้อมูล Training dataset และ Test dataset ที่ได้ในหัวข้อที่แล้ว ผู้วิจัยได้เริ่มทดลองสร้างแบบจำลองทำนายผลผลิตอ้อยและเกรดของผลผลิตอ้อยต่อไร่ โดยใช้เทคนิคการเรียนรู้ของเครื่องที่หลากหลายและใช้ทุก feature ที่มีในการทำนายและใช้ค่า default parameter ของไลบรารี Scikit-Learn ของแต่ละเทคนิคเพื่อเปรียบเทียบค่าความถูกต้องที่ได้ เพื่อผู้วิจัยจะสามารถเลือกเทคนิคที่จะนำมาพัฒนาระบบต่อไปเพื่อให้ได้ค่าความถูกต้องที่ดีที่สุด รายละเอียดของการสร้างแบบจำลองการทำนายในเบื้องต้นสามารถอธิบายแยกแบ่งเป็น 2 ส่วนดังนี้

3.3.1 แบบจำลองการทำนายเกรดของผลผลิตอ้อยเบื้องต้น

ในการพัฒนาแบบจำลองการทำนายเกรดของผลผลิตอ้อยเบื้องต้น ผู้วิจัยได้เลือกเทคนิคการทำนายต่อไปนี้ ซึ่งเป็นฟังก์ชันที่มีอยู่บนไลบรารี Scikit-Learn เพื่อทำการเปรียบเทียบประสิทธิผลเบื้องต้น

- K-Nearest Neighbors (KNN)
- Support Vector Machine (SVM)
- Gaussian Naïve Bayes
- AdaBoost
- Random Forest (RF)
- Gradient Boosting Tree (GBT)

โดยเทคนิค K-Nearest-Neighbors และ Support Vector Machine จะมีการทำ Feature Scaling ด้วย ฟังก์ชัน `StandardScaler` กับข้อมูลก่อนนำไปใช้กับเทคนิคดังกล่าว

นอกจากนี้ผู้วิจัยได้ทดลองใช้ไลบรารี XGBoost (eXtreme Gradient Boosting) ซึ่งเป็นไลบรารีสำหรับเทคนิค Gradient Boosting Tree

ส่วนย่อยของโค้ด (code snippet) ของการสร้างแบบจำลองโดยใช้เทคนิคต่าง ๆ สามารถแสดงได้ดังนี้

```
# K-Nearest-Neighbors Classifer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
XN_std = StandardScaler().fit_transform(train_selFeat.values)
XS_std = StandardScaler().fit_transform(test_selFeat.values)
t0= time()
clf = KNeighborsClassifier()
clf = clf.fit(XN_std, y_train)
print("Accuracy Train: %.2f%%" %
      (accuracy_score(y_train, clf.predict(XN_std))*100.0))
print ("Runtime = %s" %(round(time()-t0,3)))
```

```
pred = clf.predict(XS_std)
print("Accuracy Test: %.2f%%" % (accuracy_score(pred, y_test)*100.0))

#Support vector Classifer
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
XN_std = StandardScaler().fit_transform(train_selFeat.values)
XS_std = StandardScaler().fit_transform(test_selFeat.values)
t0= time()
clf = SVC(kernel="rbf")
clf = clf.fit(XN_std, y_train)
trainPredict = clf.predict(XN_std)
print("Accuracy Train: %.2f%%" %
       (accuracy_score(trainPredict, y_train)*100.0))
print ("Runtime = %s" %(round(time()-t0,3)))
pred = clf.predict(XS_std)
print("Accuracy Test: %.2f%%" % (accuracy_score(pred, y_test)*100.0))

#Gaussian Naive Bayes (GaussianNB)
from sklearn.naive_bayes import GaussianNB
t0= time()
GaussianNB(priors=None)
clf = GaussianNB()
clf = clf.fit(train_selFeat, y_train)
trainPredict = clf.predict(train_selFeat)
print("Accuracy Train: %.2f%%" %
    (accuracy_score(trainPredict, y_train)*100.0))
print ("Runtime = %s" %(round(time()-t0,3)))
pred = clf.predict(test_selFeat)
print("Accuracy Test: %.2f%%" % (accuracy_score(pred, y_test)*100.0))

# AdaBoost Classifier
from sklearn.ensemble import AdaBoostClassifier
t0= time()
clf = AdaBoostClassifier(n_estimators=20, learning_rate=2.0)
clf = clf.fit(train_selFeat, y_train)
trainPredict = clf.predict(train_selFeat)
print("Accuracy Train: %.2f%%" %
       (accuracy_score(trainPredict, y_train)*100.0))
print ("Runtime = %s" %(round(time()-t0,3)))
pred = clf.predict(test_selFeat)
print("Accuracy Test: %.2f%%" % (accuracy_score(pred, y_test)*100.0))

# Random Forests Classifer
from sklearn.ensemble import RandomForestClassifier
t0= time()
rf = RandomForestClassifier()
rf = rf.fit(train_selFeat, y_train)
trainPredict = rf.predict(train_selFeat)
print("Accuracy Train: %.2f%%" %
       (accuracy_score(trainPredict, y_train)*100.0))
print ("Runtime = %s" %(round(time()-t0,3)))
pred = rf.predict(test_selFeat)
print("Accuracy Test: %.2f%%" % (accuracy_score(pred, y_test)*100.0))


# GradientBoostingClassifier (Scikit-Learn)
from sklearn.ensemble import GradientBoostingClassifier
t0= time()
```

```
gb = GradientBoostingClassifier()
gb = gb.fit(train_selFeat, y_train)
trainPredict = gb.predict(train_selFeat)
print("Accuracy Train: %.2f%%" %
       (accuracy_score(trainPredict, y_train)*100.0))
print ("Runtime = %s" %(round(time()-t0,3)))
pred = gb.predict(test_selFeat)
print("Accuracy Test: %.2f%%" % (accuracy_score(pred, y_test)*100.0))

# XGBoost
import xgboost as xgb
train_selFeat = train_selFeat.as_matrix()
test_selFeat = test_selFeat.as_matrix()
t0= time()
xgb1 = xgb.XGBClassifier()
xgb1 = xgb1.fit(train_selFeat, y_train)
trainPredict = xgb1.predict(train_selFeat)
print("Accuracy Train: %.2f%%" %
      (accuracy_score(trainPredict, y_train)* 100.0))
print ("Runtime = %s" %(round(time()-t0,3)))
pred = xgb1.predict(test_selFeat)
print("Test Accuracy: %.2f%%" % (accuracy_score(pred, y_test)* 100.0))
```

ผลลัพธ์การทำนายบน Test dataset ของเทคนิคต่าง ๆ แสดงได้ดังตารางที่ 6 โดยเวลาที่แสดงในตารางเป็นเวลาที่รวมเวลาในการสร้างแบบจำลองจากข้อมูลใน Training dataset (จำนวน 8,765 รายการ) ซึ่งเวลาที่วัดได้ดังกล่าวทดสอบบนเครื่องคอมพิวเตอร์ notebook ที่มี CPU รุ่น Intel(R) Core(TM) i7-6500 CPU ความเร็วสัญญาณนาฬิกา 2.50 GH หน่วยความจำ RAM ขนาด 12 GB หน่วยความจำสำรองแบบ SSD ขนาด 250 GB และใช้ระบบปฏิบัติการMicrosoft Windows 10

ตาราง 6 แสดงค่า accuracy และเวลาที่ใช้ของแต่ละเทคนิค

Method	KNN	SVC	Naïve Bayes	AdaBoost	Random Forest	Graindiant Boosting Tree	XGBoost
Accuracy	64.55%	60.88%	58.87%	65.50%	*68.41%*	*70.33%*	*71.07%*
Time (วินาที)	2.18	4.96	0.02	0.22	*0.16*	*2.56*	*0.99*

จากผลการทดลองจะเห็นได้ว่า เทคนิคการทำนายแบบ Gradient Boosting Tree จะให้ค่า accuracy ที่สูงที่สุดและลำดับต่อมาคือ Random Forest แต่ Gradient Boosting Tree ใช้เวลาสูงกว่า Random Forest ส่วน Naïve Bayes นั้นให้ค่าต่ำที่สุด นอกจากนี้จะเห็นได้ว่าเทคนิคการทำนายโดยใช้ Gradient Boosting Tree ที่ใช้ไลบรารี XGBoost จะให้ค่า accuracy ที่สูงกว่าและใช้เวลาในการทำงานที่เร็วกว่า Gradient Boosting Tree ของไลบรารี Scikit-Learn

ดังนั้นในการพัฒนาเทคนิคการทำนายเกรดของผลผลิตอ้อย ซึ่งจัดเป็นปัญหาแบบ Classification ผู้วิจัยจึงเลือกใช้เทคนิค Random Forest และเทคนิค Gradient Boosting Tree ใช้ไลบรารี XGBoost ในการพัฒนาระบบต่อไป ซึ่งรายละเอียดของเทคนิคการทำนายดังกล่าวจะอธิบายในบทที่ 4

3.3.2 แบบจำลองการทำนายค่าปริมาณผลผลิตอ้อยเบื้องต้น

ในหัวข้อนี้ผู้วิจัยได้ดำเนินกระบวนการเช่นเดียวกันกับวิธีที่อธิบายในหัวข้อ 3.3.1 ในการพัฒนาแบบจำลองการทำนายค่าปริมาณของผลผลิตอ้อยเบื้องต้น โดยผู้วิจัยได้เลือกเทคนิคการทำนายต่อไปนี้ ซึ่งเป็นฟังก์ชันที่มีอยู่บนไลบรารี Scikit-Learn เพื่อทำการเปรียบเทียบประสิทธิผลเบื้องต้น

- Support Vector Machine (SVM)
- Linear Regression
- Random Forest
- Gradient Boosting Tree

โดยใช้ข้อมูลที่ผ่านการแปลงข้อมูลให้เป็น Dummy features ที่ได้จัดทำไว้ เนื่องจาก features ที่มีเป็นแบบ Category ซึ่งอธิบายไว้ในหัวข้อ 3.2 นอกจากนี้ จะประยุกต์ใช้ Feature Scaling ด้วยฟังก์ชัน `StandardScale` กับข้อมูลก่อนนำไปใช้กับเทคนิค Support Vector Machine และเช่นเดียวกับสิ่งที่อธิบายในหัวข้อ 3.3.1 ผู้วิจัยได้ทดลองใช้ไลบรารี XGBoost (eXtreme Gradient Boosting) ซึ่งเป็นไลบรารีสำหรับเทคนิค Gradient Boosting Tree มาเปรียบเทียบกับ Gradient Boosting Tree ของไลบรารี Scikit-Learn ด้วย แสดงส่วนย่อยของโค้ด (code snippet) ของการสร้างแบบจำลองโดยใช้เทคนิคต่าง ๆ ได้ดังนี้

```
#Support Vector Regression (SVR)
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
XN_std = StandardScaler().fit_transform(train_selFeat.values)
XS_std = StandardScaler().fit_transform(test_selFeat.values)
t0 =time()
clf = SVR(kernel="rbf" , gamma=0.1)
clf = clf.fit(XN_std, y_train)
print ("rmse train: ", np.sqrt(mean_squared_error(y_train,
   clf.predict(XN_std)))
print ("run in %.2fs" % (time() - t0))
print ("rmse test: ", np.sqrt(mean_squared_error(y_test,
   clf.predict(XS_std),)))
print ("R2 Variance ", r2_score(y_test, clf.predict(XS_std))
print ("MAPE: : %.2f" % mean_absolute_percentage_error(y_test,
   clf.predict(XS_std)))
```

```
#linear Regression
from sklearn import linear_model
t0 =time()
regr = linear_model.LinearRegression()
regr = regr.fit(train_selFeat, y_train)
print ("rmse train: ", np.sqrt(mean_squared_error(y_train,
   regr.predict(train_selFeat))))
print ("Runtime = %s" %(round(time()-t0,3)))
print ("rmse test: ", np.sqrt(mean_squared_error(y_test,
   regr.predict(test_selFeat))))
print ("R2 Variance ", r2_score(y_test, regr.predict(test_selFeat),))
print ("MAPE: : %.2f" % mean_absolute_percentage_error(y_test,
   regr.predict(test_selFeat)))

#RandomForestRegressor
from sklearn.ensemble import RandomForestRegressor
t0 =time()
rf = RandomForestRegressor()
rf.fit(train_selFeat, y_train)
pred = rf.predict(test_selFeat)
print ("rmse train: : %.2f" %  np.sqrt(mean_squared_error(y_train,
   rf.predict(train_selFeat))))
print ("Runtime = %s" %(round(time()-t0,3)))
print ("rmse test: : %.2f" % np.sqrt(mean_squared_error(y_test,
   rf.predict(test_selFeat))))
print('R2 Variance score: %.2f' % r2_score(y_test, pred))
print ("MAPE: : %.2f" % mean_absolute_percentage_error(y_test, pred))

# GradientBoosting Regressor
from sklearn.ensemble import GradientBoostingRegressor
t0 =time()
clf = GradientBoostingRegressor()
clf = clf.fit(train_selFeat, y_train)
pred = clf.predict(test_selFeat)
print ("rmse train: : %.2f" %  np.sqrt(mean_squared_error(y_train,
   clf.predict(train_selFeat)))
print ("Runtime = %s" %(round(time()-t0,3)))
print ("rmse test: : %.2f" % np.sqrt(mean_squared_error(y_test,
   clf.predict(test_selFeat)))
print ("MAE: : %.2f" % mean_absolute_error(y_test, pred))
print ("MAPE: : %.2f" % mean_absolute_percentage_error(y_test, pred))
print('Variance score: %.2f' % r2_score(y_test, pred))

# XGBoost
import xgboost as xgb
train_selFeat = train_selFeat.as_matrix()
test_selFeat = test_selFeat.as_matrix()
t0 =time()
xgb1 = xgb.XGBRegressor()
xgb1.fit(train_selFeat, y_train)
pred = xgb1.predict(test_selFeat)
print ("Runtime = %s" %(round(time()-t0,3)))
print ("rmse train: : %.2f" %  np.sqrt(mean_squared_error(y_train,
   xgb1.predict(train_selFeat))))
print ("rmse test: : %.2f" % np.sqrt(mean_squared_error(y_test,
   xgb1.predict(test_selFeat))))
print ("MAE: : %.2f" % mean_absolute_error(y_test, pred))
print ("MAPE: : %.2f" % mean_absolute_percentage_error(y_test, pred))
print('Variance score: %.2f' % r2_score(y_test, pred))
```

ผลลัพธ์การทำนายบน Test dataset ของเทคนิคต่าง ๆ แสดงได้ดังตารางที่ 7 โดยเวลาที่แสดงในตารางเป็นเวลาที่ใช้ในการสร้างแบบจำลองจากข้อมูล Training dataset ซึ่งเวลาที่วัดได้ดังกล่าวทดสอบบนเครื่องคอมพิวเตอร์เดียวกันกับหัวข้อ 3.3.2

ตาราง 7 แสดงค่า RMSE, MAPE, R^2 และเวลาที่ใช้ของแต่ละเทคนิค

Method	SVR	Linear Regression	Random Forest	Gradient Boosting Tree	XGBoost
RMSE (ตัน/ไร่)	2.96	3.01	*2.47*	*2.37*	*2.35*
MAPE	36.45	39.22	*33.28*	*33.88*	*33.82*
R^2	0.46	0.44	*0.61*	*0.65*	*0.66*
Time (วินาที)	5.61	0.03	*0.64*	*0.69*	*0.31*

จากผลการทดลองจะเห็นได้ว่าเทคนิคการทำนายโดยใช้ XGBoost ให้ค่า RMSE ที่ต่ำที่สุด และลำดับต่อมาคือ Gradient Boosting Tree และ Random Forest แต่ Random Forest ให้ค่า MAPE ต่ำกว่าเทคนิค Gradient Boosting Tree และ XGBoost ส่วน Support Vector Machineและ Linar Regression นั้นให้ค่า R^2 ต่ำกว่า 0.5 อีกทั้ง Linear Regression ให้ค่า MAPE สูงกว่าเทคนิคอื่น ดังนั้นผู้วิจัยจึงไม่เลือกทั้งสองเทคนิคนี้มาพัฒนาต่อ

นอกจากนี้จะเห็นได้ว่าเทคนิคการทำนายแบบ Gradient Boosting Tree ที่ใช้ไลบรารี XGBoost มีค่าผิดพลาด (RMSE และ MAPE) ที่ต่ำกว่าและมีค่า R^2 ที่สูงกว่าเทคนิค Gradient Boosting Tree ที่ใช้ไลบรารี Scikit-Learn ดังนั้นในการพัฒนาเทคนิคการทำนายค่าปริมาณผลผลิตอ้อย ซึ่งจัดเป็นปัญหาแบบ Regression ผู้วิจัยจึงเลือกใช้เทคนิค Random Forest และเทคนิค Gradient Boosting Tree ใช้ไลบรารี XGBoost ในการพัฒนาระบบต่อไป ซึ่งรายละเอียดของเทคนิคการทำนายดังกล่าวจะอธิบายในบทที่ 5

3.3.3 บทสรุปของการสร้างแบบจำลองการทำนายปริมาณผลผลิตอ้อยเบื้องต้น

จากผลลัพธ์ที่อธิบายในหัวข้อ 3.3.1 และ 3.3.2 ผู้วิจัยขอแยกการนำเสนอรายละเอียดการพัฒนาระบบการทำนายเป็น 2 ส่วนคือส่วน Classification ซึ่งเป็นการทำนายเกรดของผลผลิตอ้อย และส่วน Regression ซึ่งเป็นการทำนายค่าผลผลิตอ้อยในหน่วยตันต่อไร่ แต่เนื่องจากข้อมูลที่นำมาใช้นั้น feature ส่วนใหญ่เป็นแบบ Category ดังแสดงในตารางที่ 2 ดังนั้นการทำนายผลผลิตอ้อยต่อไร่ซึ่งเป็นการทำนายเชิงปริมาณอาจได้ผลไม่ดีนัก

นอกจากนี้จากการทบทวนงานวิจัยที่เกี่ยวข้อง ผู้วิจัยพบว่างานวิจัยส่วนใหญ่ที่ทำนายค่าปริมาณผลผลิต นั้นเหมาะกับการทำนายในกรณีที่ข้อมูลประกอบด้วย feature ประเภท Continuous มากกว่า ดังตัวอย่างเช่นงานวิจัย [6], [7], [10], [11] และ [12] ซึ่งเป็นงานวิจัยที่ทำนายปริมาณผลผลิตพืช ผู้วิจัยพบว่าใน dataset ของงานวิจัยดังกล่าวจะมี feature แบบ Continuous หลายตัวที่ถูกนำมาใช้ในการทำนาย เช่น ค่าความชื้น อุณหภูมิ ปริมาณปุ๋ย สภาพของดินและอายุของพืชที่พร้อมเก็บเกี่ยว เป็นต้น

ดังนั้นงานวิจัยนี้จึงเน้นในการทำปัญหา Classification คือการทำนายเกรดของผลผลิตอ้อยต่อไร่ โดยรายละเอียดจะอธิบายในบทที่ 4 และแสดงการทำปัญหา Regression คือการทำนายผลผลิตอ้อยในหน่วยตันต่อไร่เพื่อเปรียบเทียบผลการทำนายที่ได้ โดยแสดงรายละเอียดได้ในบทที่ 5

บทที่ 4

การพัฒนาระบบการทำนายเกรดของผลผลิตอ้อย

ในบทนี้ ผู้วิจัยอธิบายรายละเอียดของระบบการทำนายเกรดของผลผลิตอ้อย (Yield grade prediction) ซึ่งจัดเป็นปัญหาแบบ Classification ดังที่กล่าวมาแล้วในหัวข้อที่ 3.3 ผู้วิจัยได้เลือกพัฒนาระบบโดยใช้อัลกอริทึม Random Forest Classification ที่ถูกพัฒนาบนไลบรารี Scikit-Learn และใช้อัลกอริทึม Gradient Boosting Tree Classification ของไลบรารี XGBoost โดยเนื้อหาของงานวิจัยที่นำเสนอในบทนี้ได้ตีพิมพ์เผยแพร่เป็นภาษาอังกฤษในบทความวิจัย [19], [20] ในการประชุมวิชาการระดับนานาชาติ

4.1 การทำนายที่ใช้เป็นบรรทัดฐานสำหรับการทำนายเกรดของผลผลิตอ้อย (Baseline for Yield Grade Prediction)

การเปรียบเทียบความถูกต้องในการทำนายของเทคนิคการทำนายที่ได้ถูกพัฒนาในงานวิจัยนี้ ผู้วิจัยจะนำไปเปรียบเทียบอ้างอิงกับค่าความถูกต้องการทำนายที่ใช้เป็นบรรทัดฐาน (Baseline) จำนวน 2 ตัว เพื่อใช้เป็นตัวชี้วัดว่าเทคนิคการทำนายที่ผู้วิจัยได้พัฒนาขึ้นมานี้ มีประสิทธิผลในการทำนายที่ดีพอ โดยการทำนายที่ใช้เป็นบรรทัดฐาน (Baseline) ทั้ง 2 แบบนี้จะไม่ได้ใช้เทคนิคการเรียนรู้ของเครื่อง (non machine learning) ซึ่งสามารถอธิบายได้ดังนี้

(1) การทำนายผลผลิตโดยใช้ค่าผลผลิตที่ได้จากปีการผลิตก่อนหน้าในแปลงเดียวกัน โดยต่อจากนี้ผู้วิจัยขออ้างถึงโดยเรียกว่า Baseline-1

(2) การทำนายผลผลิตของฝ่ายสำรวจของโรงงานที่เป็นกรณีศึกษา โดยต่อจากนี้ผู้วิจัยขออ้างถึงโดยเรียกว่า Baseline-2

หรือกล่าวโดยเฉพาะเราจะใช้ข้อมูลในฟิลด์ `YieldOldGrade` เป็นค่าที่ได้จากการทำนายของ Baseline-1 และเราจะใช้ข้อมูลในฟิลด์ `TargetGrade` เป็นค่าที่ได้จากการทำนายของ Baseline-2 โดยส่วนย่อยของโค้ด (code snippet) ที่ใช้ในการคำนวณค่า Accuracy ของ Baseline-1 และ Baseline-2 สามารถแสดงได้ดังนี้

```
print("Accuracy BaseLine Yield - YieldOld: %.2f%%" %
      (accuracy_score(Data['YieldOldGrade'],
      Data['YieldGrade'])*100.0))
print("Accuracy BaseLine Target - Yield: %.2f%%" %
      (accuracy_score(Data['TargetGrade'],
      Data['YieldGrade'])*100.0))
```

โดยเราพบว่าผลลัพธ์ค่า Accuracy ของการทำนายเกรดของผลผลิตอ้อยของ Baseline-1 และ Baseline-2 เมื่อพิจารณาบน Test dataset จะมีค่าเท่ากับ 51.52% และ 65.50% ตามลำดับ

4.2 Random forest based yield grade prediction

งานวิจัยในส่วนนี้ใช้ Random Forest Classifier เพื่อทำนายเกรดของผลผลิตอ้อย โดยผู้วิจัยได้นำเสนอแบบจำลองการทำนายหลาย ๆ แบบ แล้วทำการเปรียบเทียบค่าความถูกต้องในการทำนาย ซึ่งรายละเอียดของแต่ละแบบจำลองสามารถอธิบายในหัวข้อย่อย 4.2.1 – 4.2.4

4.2.1 Random Forest Classifier with Scikit-Learn's default parameters

แบบจำลองการทำนายที่นำเสนอในหัวข้อนี้ เราจะกำหนดค่าพารามิเตอร์ของ Random Forest Classifier เป็นค่า default ที่กำหนดไว้บนไลบรารี Scikit-Learn และเราใช้ feature ทุกตัวที่มีในชุดข้อมูล รายละเอียดของค่าพารามิเตอร์ต่าง ๆ สามารถแสดงได้ดังนี้

bootstrap=True	class_weight=None	criterion='gini'
max_depth=None	max_features='auto'	max_leaf_nodes=None
min_impurity_split=1e-07	min_samples_leaf=1	min_samples_split=2
min_weight_fraction_leaf=0.0	n_estimators=10	n_jobs=1
oob_score=True	random_state=None	warm_start=False

โดยเราพบว่าผลลัพธ์ค่า Accuracy บน Train Set มีค่า 97.67% และค่า Accuracy บน Test Set มีค่า 68.29% โดยมีค่า OOB accuracy score คือ 64.61%

เนื่องจากค่า Accuracy บน Train Set มีค่าสูงมาก (97.67%) และมีความแตกต่างของค่า Accuracy บน Train Set และ Test set ที่ค่อนข้างมาก แบบจำลองดังกล่าวจึงเกิดปัญหา overfitting ซึ่งผู้วิจัยได้นำเสนอวิธีที่จะจัดการกับปัญหา overfitting นี้ในการพัฒนาแบบจำลองในหัวข้อถัดไป

4.2.2 Random Forest Classifier with separate parameters tuning

แบบจำลองการทำนายที่นำเสนอในหัวข้อนี้ จะทำปรับค่าพารามิเตอร์หลัก ๆ ของเทคนิค Random Forest ดังต่อไปนี้

- n_estimators
- max_depth
- min_samples_leaf

โดยแบบจำลองที่นำเสนอในหัวข้อนี้ ผู้วิจัยได้ทดลองปรับเปลี่ยนค่าพารามิเตอร์ทีละตัว แล้วทำการเทรนแบบจำลองและวัดผลค่าความถูกต้องในการทำนายของแบบจำลองโดยใช้เทคนิค K-fold cross-validation โดยในงานวิจัยนี้เราได้กำหนด K = 5 แล้วเราจะเลือกค่าที่ดีที่สุดที่ของแต่ละพารามิเตอร์ที่ทำให้ ค่า cross-validation score มีค่ามากสุด โดยค่า cross-validation score ที่ใช้ในงานวิจัยนี้คือค่าเฉลี่ยของ accuracy ในแต่ละ fold ของ cross-validition สำหรับช่วงของค่าพารามิเตอร์แต่ละตัวที่ทำการค้นหาสามารถแสดงได้ดังนี้

n_estimators	100-1000 โดย step ที่ละ 100 (หรือ 100, 200, 300, 400, ...,1000)
max_depth	5-50 โดย step ที่ละ 5 (หรือ 5, 10, 15,..., 50)
min_samples_leaf	2-50 โดย step ที่ละ 2 (หรือ 2, 4, 6,..., 50)

ส่วนย่อยของโค้ด (code snippet) สำหรับการพัฒนาแบบจำลองนี้สามารถแสดงได้ดังนี้

```
from sklearn.ensemble import RandomForestClassifier as Clf
from sklearn.model_selection import cross_val_score
# n_estimators
RangeNo = 100*np.linspace(1.0, 10.0, 10, dtype = integer)
DataScore = []
for C in RangeNo:
    DataScore.append(cross_val_score(Clf(n_estimators = C),X,z,
                                     cv = 5,scoring = 'accuracy'))
DataScore = np.array(DataScore)
mean = np.mean(DataScore,1)
std = np.std(DataScore,1)
```

```
n_estimators = (np.argmax(mean)+1) * 100
plt.gca()
plt.plot(RangeNo,mean,'o-',color='#117733')
plt.fill_between(RangeNo,mean-std,mean+std,color='#AAFFCC')
plt.ylabel('Score')
plt.xlabel('n_estimators')
plt.show()

#min_samples_leaf
RangeNo = np.linspace(2.0, 50.0, 25, dtype = integer)
DataScore2 = []
for C in RangeNo:
  DataScore2.append(cross_val_score(Clf(min_samples_leaf = C),X,z,
                                    cv = 5,scoring = 'accuracy'))
DataScore2 = np.array(DataScore2)
mean2 = np.mean(DataScore2,1)
std2 = np.std(DataScore2,1)
min_samples_leaf =  (np.argmax(mean2)+1) * 2
plt.gca()
plt.plot(RangeNo,mean2,'o-',color='#117733')
plt.fill_between(RangeNo,mean2-std2,mean2+std2,color='#AAFFCC')
plt.ylabel('Score')
plt.xlabel('min_samples_leaf')
plt.show()

# max_depth
RangeNo = np.linspace(5.0, 50.0, 10, dtype = integer)
DataScore3 = []
for C in RangeNo:
    DataScore3.append(cross_val_score(Clf(max_depth = C),X,z,
                                    cv = 5,scoring = 'accuracy'))
DataScore3 = np.array(DataScore3)
mean3 = np.mean(DataScore3,1)
std3 = np.std(DataScore3,1)
max_depth = (np.argmax(mean3)+1) * 5
plt.gca()
plt.plot(RangeNo,mean3,'o-',color='#117733')
plt.fill_between(RangeNo,mean3-std3,mean3+std3,color='#AAFFCC')
plt.ylabel('Score')
```

```
plt.xlabel('max_depth')
plt.show()


print('n_estimators' , n_estimators)
print('min_samples_leaf' , min_samples_leaf)
print('max_depth', max_depth)


from sklearn.ensemble import RandomForestClassifier as Rafo
clf_rf = Rafo(n_estimators = n_estimators,  min_samples_leaf =
    min_samples_leaf,  max_depth = max_depth, oob_score=True)
clf_rf = clf_rf.fit(train_selFeat, y_train)
print("Accuracy Train: %.2f%%" %
    (accuracy_score(clf_rf.predict(train_selFeat), y_train)*100.0))
print("Accuracy Test: %.2f%%" %
    (accuracy_score(clf_rf.predict(test_selFeat), y_test)*100.0))
```

ผลลัพธ์ของค่า cross validation score ของแบบจำลองเมื่อปรับค่าของพารามิเตอร์ n_estimators, max_depth และ min_sample_leaf เป็นค่าต่าง ๆ สามารถแสดงรายละเอียดของ cross validation score (mean score และ standard deviation score) ของแต่ละพารามิเตอร์ได้ดังตารางที่ 8, 9 และ 10 ตามลำดับและสามารถแสดงในรูปของกราฟดังภาพประกอบ 9 จากข้อมูลดังกล่าวเราพบว่าค่าของพารามิเตอร์ที่ทำให้ค่า cross validation score มีค่ามากสุดแสดงได้ดังนี้

- min_samples_leaf = 20
- n_estimators = 200
- max_depth = 5

และเมื่อทำการเทรนแบบจำลองอีกครั้งโดยใช้ค่าของพารามิเตอร์แต่ละตัวที่ค้นพบดังกล่าว และเมื่อทดสอบแบบจำลองที่เทรนใหม่นี้ เราพบว่าค่า Accuracy บน Train Set มีค่า 72.03% และค่า Accuracy บน Test Set มีค่า 70.91% โดยมีค่า OOB accuracy score คือ 70.62% ซึ่งจะเห็นได้ว่า accuracy มีค่าดีขึ้นเมื่อเทียบกับแบบจำลองที่ใช้ค่า default ของ parameter ต่าง ๆ นอกจากนี้เราพบว่าแบบจำลองนี้ไม่มีปัญหา overfitting เนื่องจากค่า Accuracy บน Train Set และค่า Accuracy บน Test Set ไม่ต่างกันมากนัก

ตาราง 8 รายละเอียดค่า mean และ standard deviation ของ cross validation score ของพารามิเตอร์ n_estimators

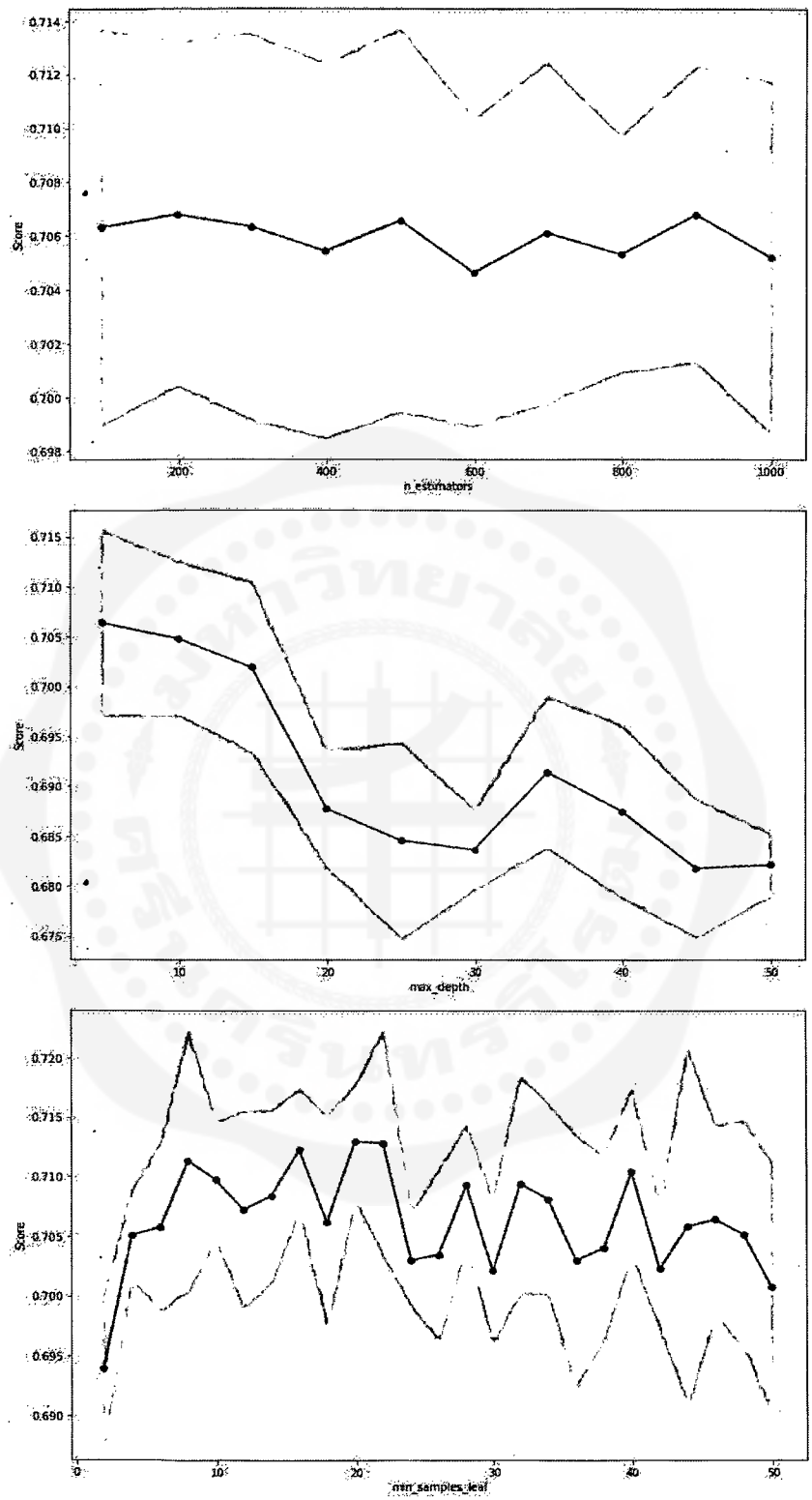
n_estimators	mean	std	n_estimators	mean	std
100	0.70633	0.01087	600	0.70462	0.00863
200	*0.70679*	*0.00851*	700	0.70611	0.00990
300	0.70634	0.01030	800	0.70531	0.01067
400	0.70542	0.00955	900	0.70678	0.00990
500	0.70656	0.00857	1000	0.70519	0.01116

ตาราง 9 รายละเอียดค่า mean และ standard deviation ของ cross validation score ของพารามิเตอร์ max_depth

max_depth	mean	std
5	*0.70633*	*0.01163*
10	0.70474	0.00415
15	0.70188	0.00525
20	0.68774	0.01035
25	0.68454	0.00636
30	0.68363	0.00741
35	0.69139	0.00301
40	0.68751	0.01235
45	0.68180	0.01248
50	0.68214	0.00989

ตาราง 10 รายละเอียดค่า mean และ standard deviation ของ cross validation score ของพารามิเตอร์ min_samples_leaf

min_samples_leaf	mean	std	min_samples_leaf	mean	std
2	0.6939	0.00578	28	0.7093	0.01213
4	0.70508	0.01254	30	0.70211	0.00917
6	0.70577	0.00710	32	0.70942	0.0127
8	0.71135	0.00730	34	0.70816	0.00542
10	0.70976	0.00415	36	0.70302	0.00479
12	0.70725	0.00443	38	0.70405	0.00748
14	0.70839	0.00722	40	0.71044	0.01399
16	0.71226	0.01251	42	0.70234	0.00929
18	0.70622	0.00323	44	0.70588	0.01057
20	*0.71295*	*0.00763*	46	0.70645	0.00571
22	0.71284	0.00462	48	0.70519	0.01253
24	0.70302	0.01289	50	0.70074	0.00588
26	0.70348	0.00287			



ภาพประกอบ 9 กราฟแสดงผลค่า mean score และ standard deviation ของ cross validation score ของแต่ละพารามิเตอร์จากเทคนิค Random Forest

4.2.3 Random Forest Classifier with Hyper-parameter tuning

แบบจำลองการทำนายที่นำเสนอในหัวข้อนี้จะขยายต่อจากแบบจำลองที่นำเสนอในหัวข้อที่ 4.2.2 เพื่อทำการลดปัญหา overfitting ที่ได้กล่าวไว้ในห้วข้อก่อนหน้านี้ หรือกล่าวโดยเฉพาะเราจะใช้เทคนิคในการปรับพารามิเตอร์ (Hyper-parameter tuning) แบบ Grid search มาช่วยในการหาค่าที่เหมาะสมให้กับแบบจำลอง ซึ่งเทคนิค Grid search นี้จะปรับค่าพารามิเตอร์ต่าง ๆ หลายตัวพร้อมกัน แล้วทำการเทรนแบบจำลองและวัดผลค่าความถูกต้องในการทำนายของแบบจำลองโดยใช้เทคนิค K-fold cross-validation (K = 5) แล้วอัลกอริทึมจะเลือกชุดของค่าพารามิเตอร์ที่ดีที่สุดที่ทำให้ ค่า cross-validation score มีค่ามากสุด โดยการทำงานตามคำอธิบายด้านบนสามารถทำได้โดยเรียกซ้ฟังก์ชัน `GridSearchCV` ของไลบรารี Scikit-Learn

สำหรับพารามิเตอร์ของเทคนิค Random Forest ที่จะพิจารณาในขั้นตอน Grid Search จะเหมือนกับแบบจำลองที่พัฒนาในหัวข้อที่แล้ว นั้นคือประกอบด้วยพารามิเตอร์ n_estimators, max_depth และ min_samples_leaf โดยช่วงของค่าพารามิเตอร์แต่ละตัวที่ทำการค้นหาสามารถแสดงได้ดังนี้

n_estimators	100-400 โดย step ที่ละ 100 (หรือ 100, 200, 300, 400)
max_depth	5-15 โดย step ที่ละ 5 (หรือ 5, 10, 15)
min_samples_leaf	10-18 โดย step ที่ละ 2 (หรือ 10, 12, 14, 16, 18)

ส่วนย่อยของโค้ด (code snippet) สำหรับการพัฒนาแบบจำลองนี้สามารถแสดงได้ดังนี้

```
from sklearn.model_selection import GridSearchCV
parameters = {'max_depth':np.arange(5,20,5),
              'n_estimators': np.arange(100,500,100),
              'min_samples_leaf': np.arange(10,20,2)}

clf_rf = RandomForestClassifier()
clf_grid = GridSearchCV(clf_rf, parameters,scoring='accuracy', cv=5)
clf_grid.fit(train_selFeat,y_train)
```

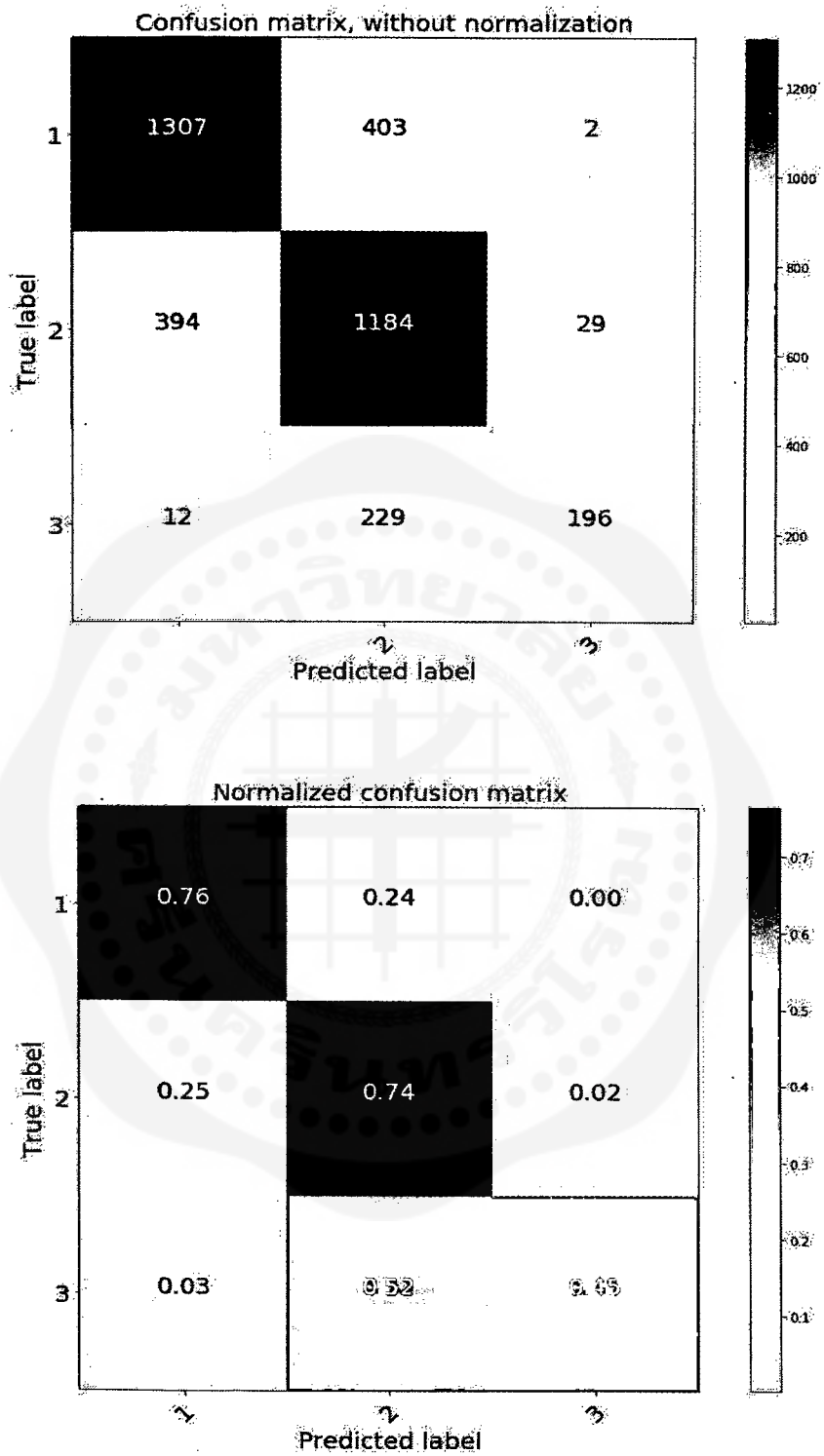
```
print('The parameters that would give best accuracy is : ')
print(clf_grid.best_params_)
print('The best accuracy achieved after parameter tuning via grid
   search is : ', clf_grid.best_score_)
clf_grid = clf_grid.fit(train_selFeat, y_train)
print("Accuracy grid_search Train: %.2f%%" %
   (accuracy_score(clf_grid.predict(train_selFeat), y_train)*100.0))
print("Accuracy grid_search Test: %.2f%%" %
   (accuracy_score(clf_grid.predict(test_selFeat), y_test)*100.0))
```

ผลลัพธ์ของการปรับจูนแบบจำลองโดยใช้ Grid search และ Cross validation จะทำให้เราได้แบบจำลองที่มีค่า cross valdation score มากสุด โดยค่าของของพารามิเตอร์ของแบบจำลองดังกล่าว แสดงดังนี้

- max_depth = 10
- min_samples_leaf = 14
- n_estimators = 300

โดยค่า cross valdation score ที่มีค่ามากสุด (ค่าที่อ่านได้จากแอตทริบิวต์ "best_score_" ของแบบจำลอง) คือ 71.89% และเมื่อนำแบบจำลองดังกล่าวไปทดสอบ training dataset กับ test dataset เราจะได้ค่า accuracy เท่ากับ 75.31% และ 71.79% ตามลำดับ ซึ่งจะเห็นได้ว่าค่า accuracy บน test dataset ของแบบจำลองนี้ (71.79%) จะมีค่ามากกว่าค่าที่ได้จากแบบจำลองที่พัฒนาในหัวข้อที่แล้ว (ค่า accuracy เท่ากับ 70.91%) ประมาณ 1%

นอกจากนี้เราสามารถแสดง Confusion matrix และ Normalized confusion matrix ของแบบจำลอง Random Forest ที่ผ่านการทำ Hyper-paramter tuning แบบ grid search เมื่อทดสอบกับ Test dataset ของผลลัพธ์การทำนายเกรดของแบบจำลองที่พัฒนาขึ้นได้ดังภาพประกอบ 10



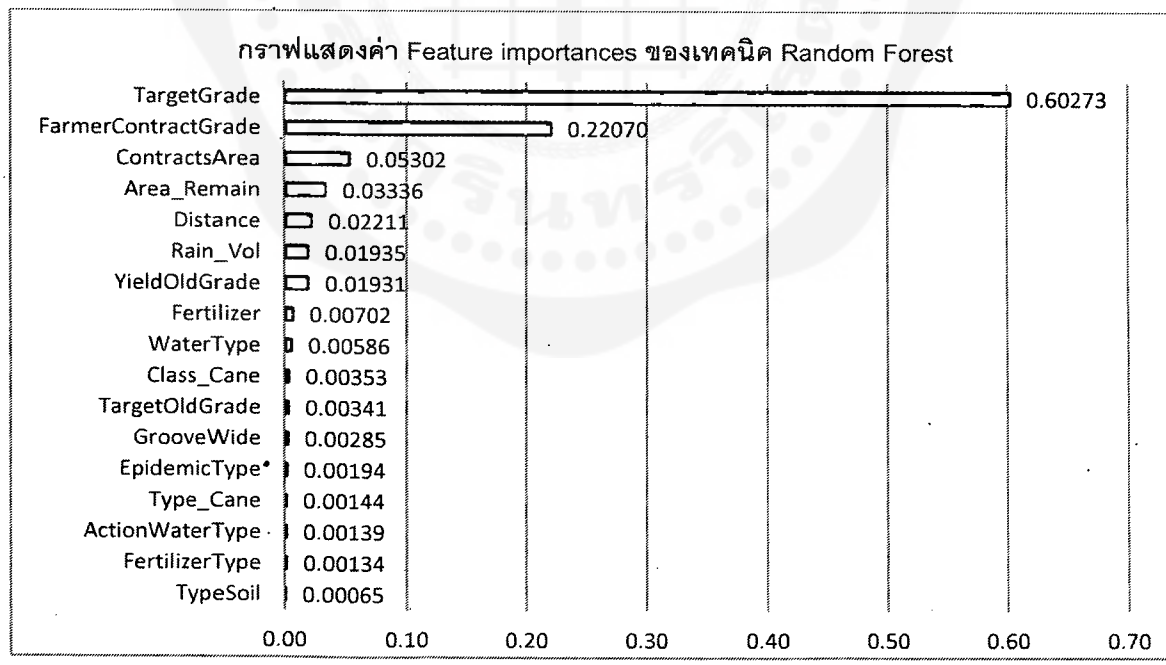
ภาพประกอบ 10 Confusion matrix และ Normalized confusion matrix ของแบบจำลอง Random Forest ที่ทำ Hyper-paramter tuning

ส่วนย่อยของโค้ด (code snippet) สำหรับส่วนคำนวณ confusion matrix จากผลลัพธ์จากการทำนายเทียบกับคลาสที่เป็นคำตอบจริง (ใช้ฟังก์ชัน `confusion_matrix` ของ Scikit-Learn) และส่วนของการแสดง confusion matrix (ฟังก์ชัน `plot_confusion_matrix`) สามารถแสดงได้ดังนี้

```
import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
    print(cm)
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title, fontsize=16)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45, fontsize=16)
    plt.yticks(tick_marks, classes, fontsize=16)
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]),
       range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black",
                 fontsize=16)
    plt.tight_layout()
    plt.ylabel('True label', fontsize=16)
    plt.xlabel('Predicted label', fontsize=16)

# Calculate the confusion matrix
cnf_matrix = confusion_matrix(y_test, pred)
# Plot the confusion matrix
np.set_printoptions(precision=3)
Class = [1,2,3]
```

```
# Plot non-normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, Class,
                      title='Confusion matrix, without normalization')
# Plot normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, Class, normalize=True,
                      title='Normalized confusion matrix')
```

นอกจากนี้เราสามารถคำนวณหาค่าความสำคัญของแต่ละ feature ที่มีต่อแบบจำลอง (feature importances) โดยใช้แอตทริบิวต์ `"feature_importances_"` ของแบบจำลอง (ชนิด `RandomForestClassifier`) ใน Scikit-Learn โดยที่ถ้าค่า feature importance ของ feature ใด ยิ่งมีค่ามากจะแสดงว่า feature นั้นจะมีความสำคัญต่อการทำนายของแบบจำลอง โดยผลลัพธ์ค่า feature importance ของ feature ต่าง ๆ ของแบบจำลองที่พัฒนาในหัวข้อนี้สามารถแสดงได้ดังภาพประกอบ 11 ซึ่งจะเห็นได้ว่า TargetGrade จะเป็น feature ที่มีความสำคัญมากสุด ด้วยค่า importance คือ 0.60273 และ FarmerContractGrade จะสำคัญเป็นลำดับรองลงไปด้วยค่า importance คือ 0.22070



ภาพประกอบ 11 กราฟแสดงค่า feature importances ของแต่ละ features ที่ของแบบจำลอง Random Forest ที่ทำ Hyper-paramter tuning

สำหรับส่วนย่อยของโค้ด (code snippet) ที่ทำการคำนวณค่า feature importance ซึ่งใช้ไลบรารี pandas มาช่วยในการประมวลผล สามารถแสดงได้ดังนี้

```
import pandas as pd
feature_importances = pd.DataFrame(clf_rf.feature_importances_,
                                   index = train_selFeat.columns,
                                   columns=['importance']).
                                   sort_values('importance',
                                   ascending=False)
```

4.2.4 Random Forest Classifier with Forward Feature Selection and Hyper-parameter tuning

แบบจำลองการทำนายที่ใช้เทคนิค Random Forest Classifier ที่ได้นำเสนอในหัวข้อก่อนหน้านี้ จะใช้ทุก features ที่มีอยู่ในข้อมูลมาใช้ในการทำนาย ในหัวข้อย่อยนี้ผู้วิจัยจะนำเสนอการพัฒนาแบบจำลองที่ใช้เทคนิค forward feature selection เพื่อหา features หลักที่มีผลต่อการทำนาย แล้วเลือกใช้เฉพาะ feature เหล่านั้นร่วมกับเทคนิค hyper-parameter tuning ในการสร้างแบบจำลอง

อัลกอริทึมในส่วนของการเทรนแบบจำลองที่นำเสนอในหัวข้อนี้ สามารถแสดงในรูปแบบ Pseudo-code โดยอ้างอิงชื่อฟังก์ชันที่สำคัญของไลบรารี Scikit-Learn ได้ดัง **Algorithm 1** โดยเทคนิค forward feature selection ที่นำเสนอในหัวข้อนี้ได้ดัดแปลงจากแนวคิดที่นำเสนอในงานวิจัย [17]

Algorithm 1: RF model training with Forward Feature Selection and Hyper-parameter tuning
Input: Training data: X=features, Y=class label, F=set of features
Output: Trained model: best_clf and best hyper-parameters: best_params

```
clf_1, params_1=GridSearchCV(RandomForestClassifier(),X[F],Y)
impVal = clf_1.feature_importance()
F=sort_descending(F, impVal)
S0={}
for i=1 to |F|
        Si = Si-1 ∪ F[i]
        clf = RandomForestClassifier(params_1)
        ValScore[i]=cross_validation_score(clf,X[Si],Y)
end for
k = argmaxi ValScore[i]
best_clf,best_params=GridSearchCV(RandomForestClassifier(),X[Sk],Y)
```

โดยคำอธิบายของอัลกอริทึมของการสร้างแบบจำลองโดยใช้เทคนิค forward feature selection สามารถอธิบายได้ดังนี้

i. โดยอ้างอิง Pseudo-code ที่บรรทัด 1: อัลกอริทึมจะทำการเทรนแบบจำลองของ Random Forest Classification โดยใช้ทุก feature และ ทำ Hyper-parameter tuning ด้วยวิธี Grid Search โดยเราจะได้แบบจำลองที่แทนด้วย clf_1 และ ค่าของ parameter ของแบบจำลองที่ดีที่สุดจะอยู่ที่ params_1

ii. โดยอ้างอิง Pseudo-code ที่บรรทัด 2: อัลกอริทึมจะทำการคำนวณหาค่า feature importance ของแต่ละ features โดยกำหนดให้เก็บไว้ที่ตัวแปร impVal

iii. โดยอ้างอิง Pseudo-code ที่บรรทัด 3: อัลกอริทึมจะทำการเรียงลำดับค่าความสำคัญของแต่ละ features จากมากไปน้อย โดยกำหนดให้เก็บไว้ที่ตัวแปร F

iv. โดยอ้างอิง Pseudo-code ที่บรรทัด 5-9: อัลกอริทึมจะวนลูปเพิ่ม feature เข้ามาทีละตัวลงในเซ็ต S_i ตามลำดับค่า feature importance ที่ได้จากขั้นตอนที่แล้ว หลังจากนั้นเราจะเทรนแบบจำลองจากข้อมูลที่ใช้เฉพาะชุด feature ในเซ็ต S_i นี้ และใช้ค่าพารามิเตอร์ clf_1 ที่ได้ในขั้นตอนแรก (บรรทัด 1) ในการเทรนแบบจำลอง Random Forest โดยการประเมินผลว่าแบบจำลองที่ถูกสร้างจากชุด feature S_i นี้มีค่า accuracy เป็นอย่างไร เราจะใช้เทคนิค K-fold cross validation ซึ่งจะทำให้เราได้ค่า cross validation score (ค่าเฉลี่ยของ accuracy ในแต่ละ fold) โดยกำหนดให้เก็บไว้ที่ตัวแปร ValScore[i]

v. โดยอ้างอิง Pseudo-code ที่บรรทัด 10: เราจะเลือกเซ็ตของ feature ที่ค่า cross validation score มีค่ามากสุด โดยสมมิตว่าเซ็ตของ feature ดังกล่าวคือ S_k

vi. โดยอ้างอิง Pseudo-code ที่บรรทัด 11: ทำการเทรนแบบจำลองใหม่จากข้อมูลที่ใช้เฉพาะชุด feature ในเซ็ต S_k นี้ โดยในขั้นตอนการเทรนแบบจำลองนี้ เราจะใช้เทคนิค Hyper-parameter tuning แบบ Grid Search ร่วมกับการทำ Cross Validation เพื่อค้นหาค่าพารามิเตอร์ที่ทำให้ได้แบบจำลองที่มีค่า Cross Validation Score สูงสุด

โดยพารามิเตอร์ที่จะพิจารณาในขั้นตอน Grid Search นี้ ประกอบด้วยพารามิเตอร์ n_estimators, max_depth, min_samples_leaf และ max_features โดยช่วงของค่าพารามิเตอร์แต่ละตัวที่ทำการค้นหาสามารถแสดงได้ดังนี้

n_estimators	100, 200, 300, 400, 500
max_depth	5, 10
min_samples_leaf	10, 12, 14, 16, 18
max_features	'None', 4, 5, 6

ส่วนย่อยของโค้ด (code snippet) จาก algorithm ในส่วนของการทำ forward feature selection and Hyper-parameter tuning สามารถแสดงได้ดังนี้

```
from sklearn.ensemble import RandomForestClassifier as Rafo
from sklearn.model_selection import cross_val_score as crovasco
X,z = Train[Features], Train.YieldGrade
XX,zz = Test[Features], Test.YieldGrade

rafo = Rafo(n_estimators = 300,  min_samples_leaf = 14,
            max_features  =4, max_depth = 10)
rafo.fit(X,z)

rafofi = rafo.feature_importances_
feature_importances = pd.DataFrame(rafo.feature_importances_,
                      index = X.columns,

columns=['importance']).sort_values('importance',
          ascending=False)

Xsort = X.iloc[:,rafo.feature_importances_.argsort()[:-17:-1]]
X.columns = X.columns.str.strip()
plt.barh(np.arange(17),rafo.feature_importances_)
plt.yticks(np.arange(17),X.columns,rotation=45)
plt.show()

riang = rafo.feature_importances_.argsort()[::-1]
RafoCross = []

for i in range(1,18):
```

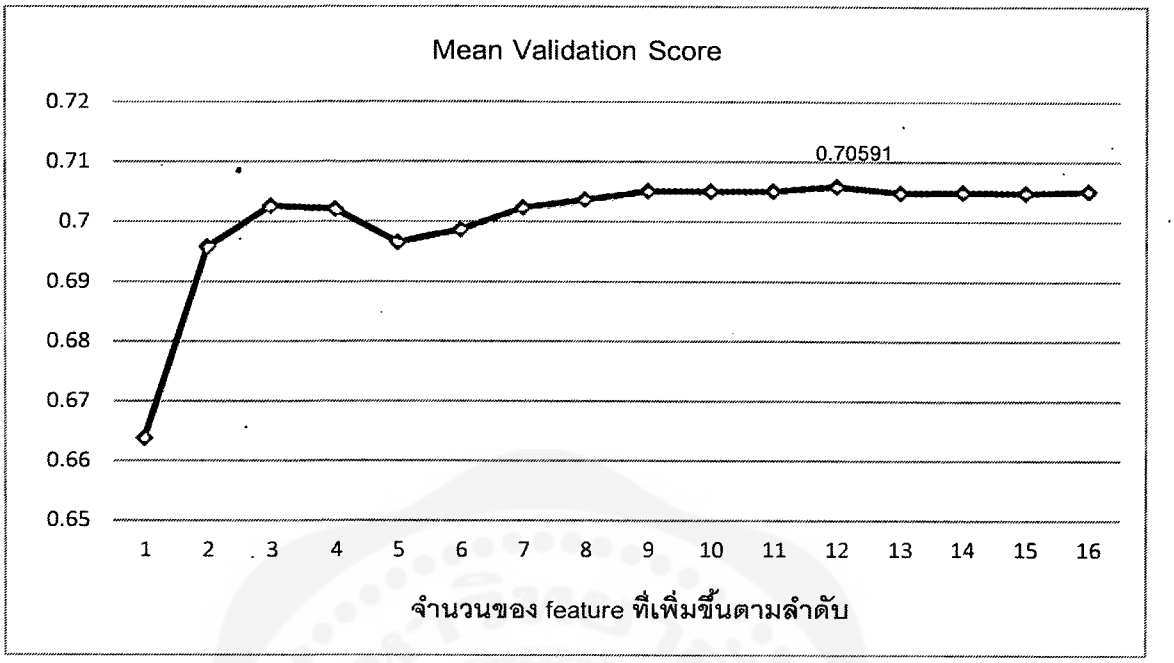
```
        RafoCross.append(crovasco(Rafo(max_depth=10, max_features=
                          None, min_samples_leaf= 14, n_estimators=
                          300), X.iloc[:,riang[:i]],z,cv=5))
RafoCross = np.array(RafoCross)
mean = RafoCross.mean(1)
std = RafoCross.std(1)

plt.plot(np.arange(1,18),mean,'o-',color='#117733')
plt.fill_between(np.arange(1,18),mean-std,mean+std,color='#AAFFCC')
plt.show()

#กำหนด feature จากจำนวนของ feature ที่ให้ค่า mean มาที่สุด
k=np.argmax(RafoCross.mean(1))+1
train_selFeat_new= X.iloc[:,riang[:k]]
test_selFeat_new= XX.iloc[:,riang[:k]]
print (test_selFeat_new.head)

y_train = z
y_test = zz

parameters = {'min_samples_leaf':[10,12,14,16,20],
              'max_depth': [5, 10],
              'n_estimators': [100, 200, 300, 400, 500],
              'max_features': ['None',4,5,6]}

from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
clfg = GridSearchCV(RandomForestClassifier(), param_grid=params)
clfg.fit(train_selFeat_new, y_train)

# summarize results
print("Best: %f using %s" % (clfg.best_score_, clfg.best_params_))
print("Accuracy grid_search Train: %.2f%%" % (accuracy_score(y_train,
     clfg.predict(train_selFeat_new))*100.0))
print("Accuracy grid_search Test: %.2f%%" % (accuracy_score(y_test,
     clfg.predict(test_selFeat_new))*100.0))
```

ผลลัพธ์ของอัลกอริทึมจะพบว่า ค่า mean cross validation score ของแต่ละเซ็ตจากการเพิ่ม features เข้าไปทีละตัวจนครบทั้งหมด พบว่าแบบจำลองที่ใช้ feature เรียงตามลำดับความสำคัญถึง

ลำดับที่ 12 จะได้ค่า cross validation score มากที่สุด คือ 0.70591 สามารถแสดงรายละเอียดดังตารางที่ 11 และแสดงกราฟดังภาพประกอบ 12

ตาราง 11 แสดงค่า feature importance และค่า mean validation score จากการเพิ่มทีละ features

Feature name	Feature importance value	Mean Validation Score (after adding the feature)
TargetGrade	**0.60273**	0.66374
FarmerContractGrade	0.22070	0.69577
ContractsArea	0.05302	0.70264
Area_Remain	0.03336	0.70208
Distance	0.02211	0.69649
Rain_Vol	0.01935	0.69864
YieldOldGrade	0.01931	0.70232
Fertilizer	0.00702	0.70367
WaterType	0.00586	0.70511
Class_Cane	0.00353	0.70503
TargetOldGrade	0.00341	0.70511
GrooveWide	0.00285	**0.70591**
EpidemicType	0.00194	0.70487
Type_Cane	0.00144	0.70495
ActionWaterType	0.00139	0.70487
FertilizerType	0.00134	0.70511
TypeSoil	0.00065	0.70407



ภาพประกอบ 12 กราฟแสดงค่า mean (cross) validation score จากการเพิ่มทีละ features

จากผลลัพธ์ที่ได้นั้น feature ที่ถูกเลือกใช้ทั้งหมด 12 ตัวตามลำดับที่ได้มาได้แก่ TargetGrade, FarmerContractGrade, ContractsArea, Area_Remain, Distance, Rain_Vol, YieldOldGrade, Fertilizer, WaterType, Class_Cane, TargetOldGrade และ GrooveWide

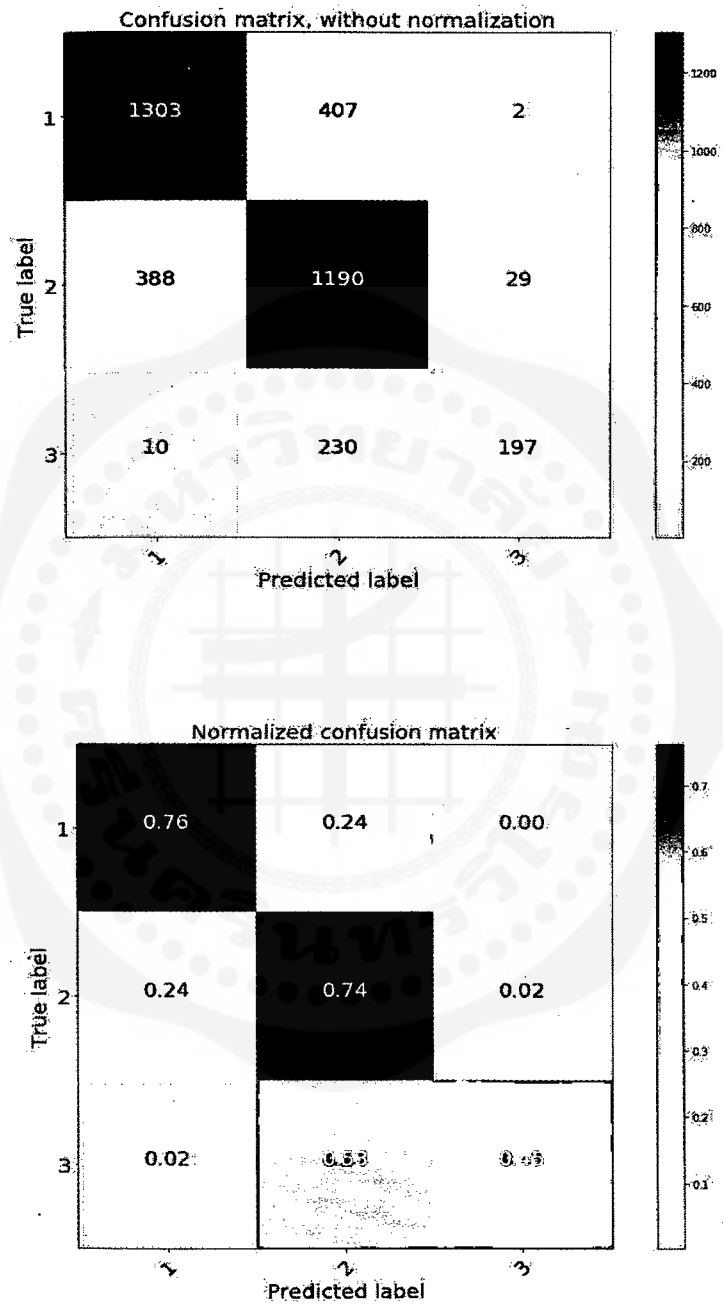
โดยที่ผลของการทำ Grid Search เมื่อเลือกใช้เพียง 12 feature นี้ จะได้ค่า cross valdation score ที่มีค่ามากสุด (ค่าที่อ่านได้จากแอตทริบิวต์ "best_score_" ของแบบจำลอง) คือ 71.83% โดยค่าของของพารามิเตอร์ของแบบจำลองดังกล่าว แสดงดังนี้

- n_estimators = 100
- max_depth = 10
- min_samples_leaf = 14
- max_features = 6

เมื่อนำแบบจำลองดังกล่าวไปทดสอบ training dataset กับ test dataset เราจะได้ค่า accuracy เท่ากับ คือ 75.83% และ 71.88% ตามลำดับ โดย confusion matrix และ normalized confusion matrix ของผลที่ทำนายออกมาได้จาก test set สามารถแสดงได้ดังภาพประกอบ 13

จากผลลัพธ์ดังกล่าวจะเห็นได้ว่าค่า accuracy บน test dataset ของแบบจำลองนี้ (71.88%) จะมีค่ามากกว่าค่าที่ได้จากแบบจำลองที่พัฒนาในหัวข้อที่แล้วเล็กน้อย (71.79%) แต่แบบจำลองนี้จะ

มีข้อดีกว่าคือค่า n_estimators = 100 หมายถึงใช้จำนวน decision tree ในแบบจำลองที่น้อยกว่าและใช้ feature ในการทำนายที่น้อยกว่า แต่ได้ค่า Accuracy ใกล้เคียงกัน



ภาพประกอบ 13 Confusion matrix และ Normalized confusion matrix จากเทคนิค Yield grade prediction using Random Forest with Forward Feature Selection and Hyper-parameter tuning

4.3 Gradient Boosting Tree based yield grade prediction

ในหัวข้อนี้เราจะนำเสนอการพัฒนาแบบจำลองการทำนายโดยใช้เทคนิค Gradient Boosting Tree Classifier ของไลบรารี XGBoost เพื่อทำนายค่าตัวแปร YieldGrade โดยผู้วิจัยได้นำเสนอแบบจำลองการทำนายหลาย ๆ แบบ ในลักษณะเดียวกับการสร้างแบบจำลองการทำนายโดยใช้ Random Forest Classifier ในหัวข้อ 4.2 แล้วจึงทำการเปรียบเทียบค่าความถูกต้องในการทำนาย ซึ่งรายละเอียดของแต่ละแบบจำลองสามารถอธิบายในหัวข้อย่อย 4.3.1 – 4.3.3

4.3.1 Gradient Boosting Tree Classifier with default parameters

แบบจำลองการทำนายที่นำเสนอในหัวข้อนี้ เรากำหนดค่าพารามิเตอร์ของ Gradient Boosting Tree Classifier เป็นค่า default ที่กำหนดไว้บนไลบรารี XGBoost (XGBClassifier) และเราใช้ feature ทุกตัวที่มีในชุดข้อมูล รายละเอียดของค่าพารามิเตอร์ต่าง ๆ สามารถแสดงได้ดังนี้

base_score=0.5	colsample_bylevel=1	colsample_bytree=1
gamma=0	learning_rate=0.1	max_delta_step=0
max_depth=3	min_child_weight=1	n_estimators=100
missing=None	objective='binary:logistic'	scale_pos_weight=1
seed=0	scale_pos_weight=1	subsample=1

โดยเราพบว่าผลลัพธ์ค่า Accuracy บน Train Set มีค่า 72.74% และ ค่า Accuracy บน Test Set มีค่า 71.07% เนื่องจากความแตกต่างของค่า Accuracy บน Train Set และ Test set ที่ค่อนข้างน้อย แบบจำลองดังกล่าวจึงไม่เกิดปัญหา overfitting อย่างเช่นในกรณีที่ของแบบจำลอง Random Forest Classifier ที่ใช้ค่า default parameter ในหัวข้อ 4.2.1

4.3.2 Gradient Boosting Tree Classifier with separate parameters tuning

แบบจำลองการทำนายที่นำเสนอในหัวข้อนี้ ทำการปรับค่าพารามิเตอร์หลัก ๆของเทคนิค Gradient Boosting Tree Classifier ดังต่อไปนี้

- n_estimators
- max_depth
- learning_rate
- min_child_weight

โดยเราใช้วิธีเดียวกับกรณีที่ใช้เทคนิค Random Forest ในหัวข้อ 4.2.2 นั้นคือทดลองปรับเปลี่ยนค่าพารามิเตอร์ทีละตัว แล้วทำการเทรนแบบจำลองและวัดผลค่าความถูกต้องในการทำนายของแบบจำลองโดยใช้เทคนิค K-fold cross-validation (K=5) แล้วเราจะเลือกค่าที่ดีที่สุดที่ของแต่ละพารามิเตอร์ที่ทำให้ ค่า cross-validation score มีค่ามากสุด

สำหรับช่วงของค่าพารามิเตอร์แต่ละตัวที่ทำการค้นหาสามารถแสดงได้ดังนี้

n_estimators	100-1000 โดย step ที่ละ 100 (หรือ 100, 200, 300, 400, ...,1000)
max_depth	2, 4, 6, 8, 10
learning_rate	0.1, 0.2, 0.3, 0.4, 0.5
min_child_weight	1, 2, 3, 4, 5

ส่วนย่อยของโค้ด (code snippet) สำหรับการพัฒนาแบบจำลองนี้สามารถแสดงได้ดังนี้

```
# n_estimators
RangeNo = 100*np.linspace(1.0, 10.0, 10, dtype = integer)
DataScore = []
for C in RangeNo:
    DataScore.append(cross_val_score(xgb.XGBClassifier(n_estimators=C)
    ,X,z, cv=5, scoring='accuracy'))
DataScore = np.array(DataScore)
mean = np.mean(DataScore,1)
std = np.std(DataScore,1)
plt.gca()
plt.plot(RangeNo,mean,'o-',color='#117733')
plt.fill_between(RangeNo,mean-std,mean+std,color='#AAFFCC')
plt.ylabel('Score')
plt.xlabel('n_estimators')
plt.show()


# learning_rate
RangeNo = np.linspace(0.1, 0.5, 5)
DataScore2 = []
```

```
for C in RangeNo:

    DataScore2.append(cross_val_score(xgb.XGBClassifier(learning_rate=
    C),X,z,cv=5,scoring='accuracy'))
DataScore2 = np.array(DataScore2)
mean2 = np.mean(DataScore2,1)
std2 = np.std(DataScore2,1)
plt.gca()
plt.plot(RangeNo,mean2,'o-',color='#117733')
plt.fill_between(RangeNo,mean2-std2,mean2+std2,color='#AAFFCC')
plt.ylabel('Score')
plt.xlabel('learning_rate')
plt.show()

# max_depth
RangeNo = np.linspace(2.0, 10.0, 5, dtype = integer)
DataScore3 = []
for C in RangeNo:
    DataScore3.append(cross_val_score(xgb.XGBClassifier(max_depth=C),
    X,z,cv=5,scoring='accuracy'))
khanae3 = np.array(DataScore3)
mean3 = np.mean(DataScore3,1)
std3 = np.std(DataScore3,1)
plt.gca()
plt.plot(RangeNo,mean3,'o-',color='#117733')
plt.fill_between(RangeNo,mean3-std3,mean3+std3,color='#AAFFCC')
plt.ylabel('Score')
plt.xlabel('max_depth')
plt.show()

# min_child_weight
RangeNo = np.linspace(1.0, 5.0, 5, dtype = integer)
DataScore4 = []
for C in RangeNo:
    DataScore3.append(cross_val_score(xgb.XGBClassifier(min_child_weight =
    C),X,z,cv=5,scoring='accuracy'))
khanae4 = np.array(DataScore4)
mean4 = np.mean(DataScore4,1)
std4 = np.std(DataScore4,1)
plt.gca()
```

```
plt.plot(RangeNo,mean3,'o-',color='#117733')
plt.fill_between(RangeNo,mean4-std4,mean4+std4,color='#AAFFCC')
plt.ylabel('Score')
plt.xlabel('min_child_weight')
plt.show()
```

ผลลัพธ์ของค่า cross validation score ของแบบจำลองเมื่อปรับค่าของพารามิเตอร์ n_estimators, learning_rate, max_depth และ min_child_weight เป็นค่าต่าง ๆ สามารถแสดงรายละเอียดของ cross validation score (mean score และ standard deviation score) ของแต่ละพารามิเตอร์ได้ดังตาราง 12, 13, 14 และ 15 ตามลำดับและสามารถแสดงในรูปของกราฟดังภาพประกอบ 14 จากข้อมูลดังกล่าวเราพบว่าค่าของพารามิเตอร์ที่ทำให้ค่า cross validation score มีค่ามากสุดแสดงได้ดังนี้

- n_estimators = 200
- learning_rate = 0.2
- max_depth = 4
- min_child_weight=2

เมื่อทำการเทรนแบบจำลองอีกครั้งโดยใช้ค่าของพารามิเตอร์แต่ละตัวที่ค้นพบดังกล่าวและเมื่อทดสอบแบบจำลองที่เทรนใหม่นี้ เราพบว่าค่า Accuracy บน Train Set มีค่า 72.23% และค่า Accuracy บน Test Set มีค่า 70.91% เมื่อเปรียบเทียบค่านี้กับค่า Accuracy บน Test set ของแบบจำลองที่ใช้ค่า default parameter ของ ไลบรารี XGBoost (71.07%) จะเห็นได้ว่าการเทรนแบบจำลองการปรับพารามิเตอร์ทีละตัวจะให้ผลลัพธ์ค่า accuracy ที่น้อยลง

ส่วนย่อยของโค้ด (code snippet) ที่ทำงานในส่วนนี้สามารถแสดงได้ดังนี้

```
import xgboost as xgb
xgb1 = xgb.XGBClassifier(learning_rate=0.2,n_estimators=200,
                         max_depth=4, min_child_weight=2 )
xgb1.fit(train_selFeat, y_train)
print("Train Accuracy: %.2f%%" %
   (accuracy_score(xgb1.predict(train_selFeat), y_train)* 100.0))
print("Test Accuracy: %.2f%%" %
   (accuracy_score(xgb1.predict(test_selFeat), y_test) * 100.0))
```

ตาราง 12 รายละเอียดค่า mean score และ standard deviation score ของ n_estimators ของ Gradient Boosting Tree

n_estimators	mean	std
100	0.71158	0.00458
200	**0.71512**	**0.00452**
300	0.71113	0.00552
400	0.70713	0.00468
500	0.70611	0.00697
600	0.70303	0.00810
700	0.70086	0.00728
800	0.69858	0.00484
900	0.69607	0.00358
1000	0.69549	0.00374

ตาราง 13 รายละเอียดค่า mean score และ standard deviation score ของ max_depth ของ Gradient Boosting Tree

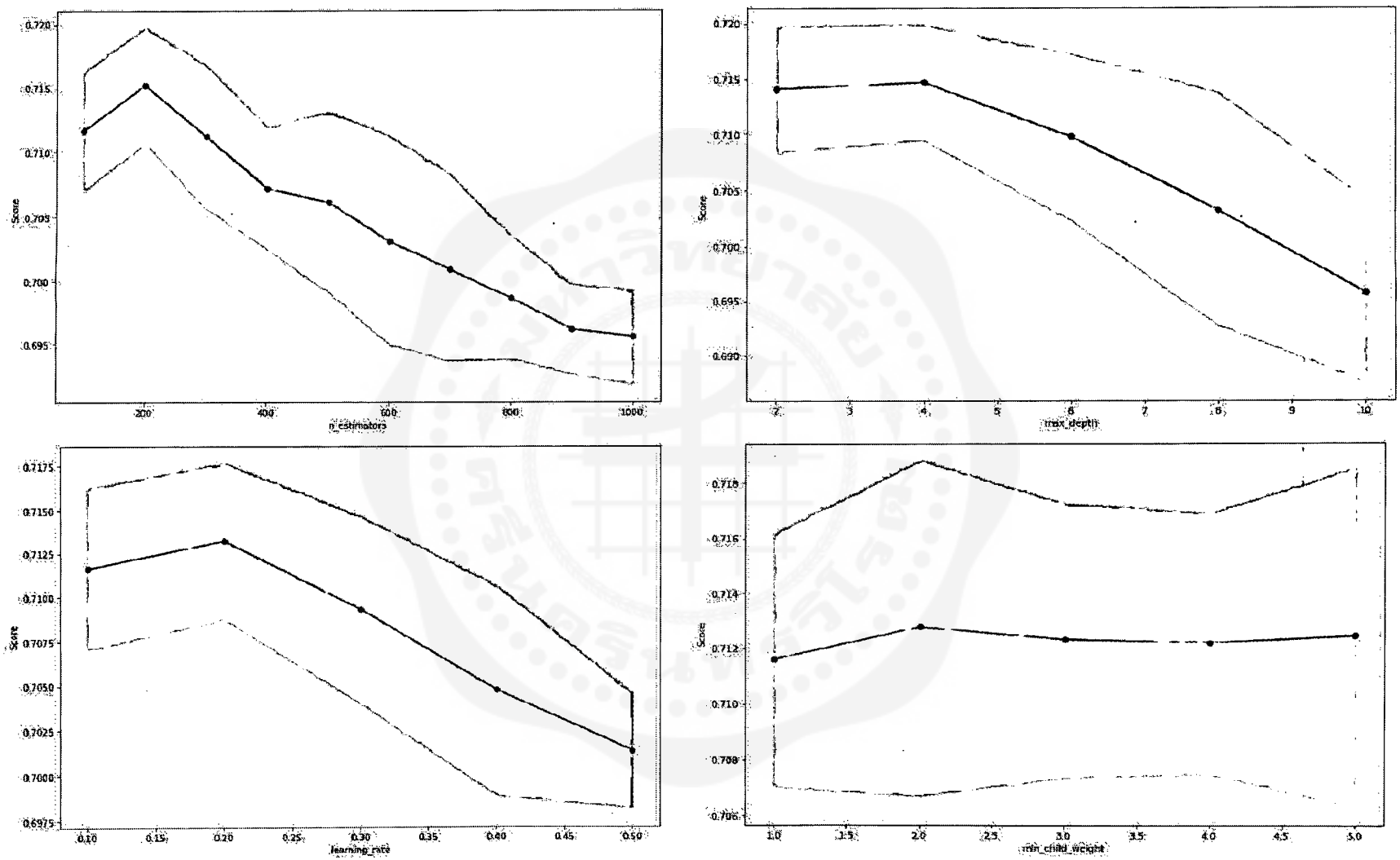
max_depth	mean	std
2	0.71398	0.00563
4	**0.71455**	**0.0052**
6	0.70964	0.00742
8	0.70314	0.01045
10	0.69572	0.00827
12	0.68968	0.00691
14	0.69047	0.00704
16	0.68819	0.00375
18	0.68865	0.00453
20	0.68831	0.00397

ตาราง 14 แสดงรายละเอียดค่า mean score และ standard deviation score ของ learning_rate ของ Gradient Boosting Tree

learning_rate	mean	std
0.1	0.71158	0.00458
0.2	**0.71318**	**0.00444**
0.3	0.70930	0.0053
0.4	0.70473	0.00588
0.5	0.70131	0.00318

ตาราง 15 แสดงรายละเอียดค่า mean score และ standard deviation score ของ min_child_weight ของ Gradient Boosting Tree

min_child_weight	mean	std
1	0.71158	0.00458
2	**0.71272**	**0.00606**
3	0.71227	0.00493
4	0.71215	0.00469
5	0.71238	0.00609



ภาพประกอบ 14 ค่า Cross Validation Score (mean score และ standard deviation score) ของแต่ละพารามิเตอร์จากเทคนิค Gradient Boosting Tree

4.3.3 Gradient Boosting Tree Classifier with hyper-parameter tuning

แบบจำลองการทำนายที่นำเสนอในหัวข้อนี้จะใช้เทคนิคในการปรับพารามิเตอร์ (Hyper-parameter tuning) แบบ Grid search มาช่วยในการหาค่าที่เหมาะสมในการเทรนแบบจำลองเพื่อลดปัญหาการเกิด overfitting เช่นเดียวกับวิธีที่นำเสนอในหัวข้อ 4.2.3 สำหรับเทคนิค Random Forest

เทคนิค Grid search นี้จะปรับค่าพารามิเตอร์ต่าง ๆ หลายตัวพร้อมกันแล้วทำการเทรนแบบจำลองและวัดผลค่าความถูกต้องในการทำนายของแบบจำลองโดยใช้เทคนิค K-fold cross-validation (K = 5) โดยอัลกอริทึมจะเลือกชุดของค่าพารามิเตอร์ที่ดีที่สุดที่ทำให้ ค่า cross-validation score มีค่ามากสุด

สำหรับพารามิเตอร์และช่วงของค่าพารามิเตอร์ของเทคนิค Gradient Boosting ที่จะพิจารณาในขั้นตอน Grid Search สามารถแสดงได้ดังนี้

n_estimators	100-500 โดย step ที่ละ 100 (หรือ 100, 200, 300, 400)
max_depth	2-10 โดย step ที่ละ 2 (หรือ 2,4,6,8)
learning_rate	0.1-0.4 โดย step ที่ละ 0.1 (หรือ 0.1, 0.2, 0.3, 0.4)
min_child_weight	2-10 โดย step ที่ละ 2 (หรือ 2,4,6,8)

ส่วนย่อยของโค้ด (code snippet) สำหรับการพัฒนาแบบจำลองนี้สามารถแสดงได้ดังนี้

```
from sklearn.model_selection import GridSearchCV
parameters = {'max_depth':np.arange(2,10,2),
              'n_estimators': np.arange(100,500,100),
              'min_child_weight': np.arange(2,10,2),
              'learning_rate': np.arange(0.1,0.4,0.1)}
xgb1 = xgb.XGBClassifier()
clf_grid = GridSearchCV(xgb1, parameters,scoring='accuracy', cv=5)
clf_grid.fit(train_selFeat,y_train)
print('The parameters combination that would give best accuracy is : ')
print(clf_grid.best_params_)
```

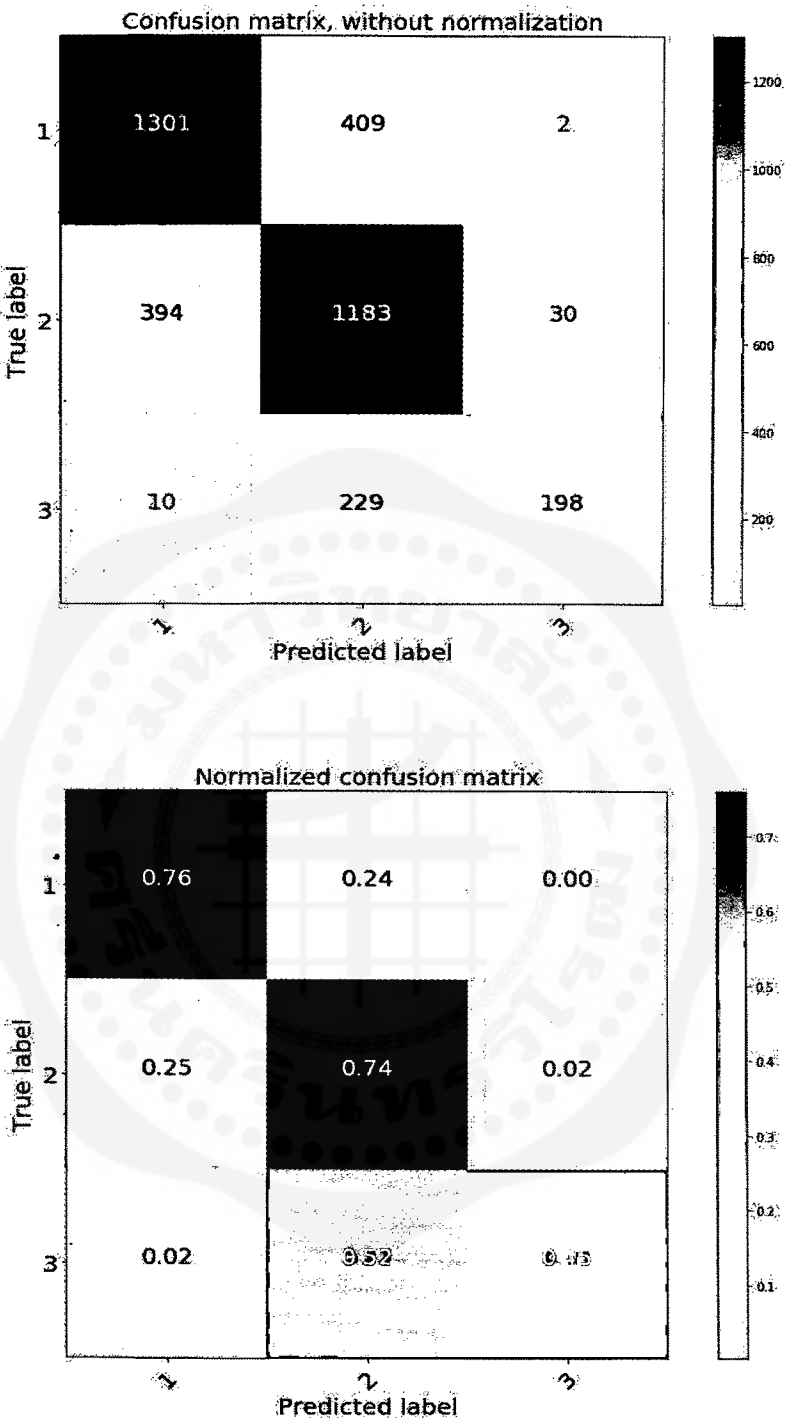
```
print('The best accuracy achieved after parameter tuning via grid
      search is : ', clf_grid.best_score_)
print ("Use Random Forests Classifer and Gridsearch")
clf_grid = clf_grid.fit(train_selFeat, y_train)
print("Accuracy grid_search Train: %.2f%%" %
      (accuracy_score(clf_grid.predict(train_selFeat),
      y_train)*100.0))
print("Accuracy grid_search Test: %.2f%%" %
      (accuracy_score(clf_grid.predict(test_selFeat), y_test)*100.0))
```

ผลลัพธ์ของการปรับจูนแบบจำลองโดยใช้ Grid search และ Cross validation จะทำให้เราได้แบบจำลองที่มีค่า cross valdation score มากสุด โดยค่าของของพารามิเตอร์ของแบบจำลองดังกล่าวแสดงดังนี้

- n_estimators = 100
- learning_rate = 0.1
- max_depth = 4
- min_child_weight = 4

โดยค่า cross valdation score ที่มีค่ามากสุด (ค่าที่อ่านได้จากแอตทริบิวต์ "best_score_" ของแบบจำลอง) คือ 71.54% และเ มื่อนำแบบจำลองดังกล่าวไปทดสอบ training dataset กับ test dataset เราจะได้ค่า accuracy เท่ากับ 74.28% และ 71.71% ตามลำดับ ซึ่งจะเห็นได้ว่าค่า accuracy บน test dataset ของแบบจำลองนี้ (71.71%) จะมีค่ามากกว่าค่าที่ได้จากแบบจำลองที่พัฒนาในหัวข้อที่ 4.3.1และ4.3.2 ซึ่งได้ค่า accuracy เท่ากับ 71.07% และ 70.91% ตามลำดับ

โดย Confusion matrix ของผลลัพธ์การทำนายเกรดของแบบจำลองที่พัฒนาขึ้นนี้บน Test dataset แสดงได้ดังภาพประกอบ 15

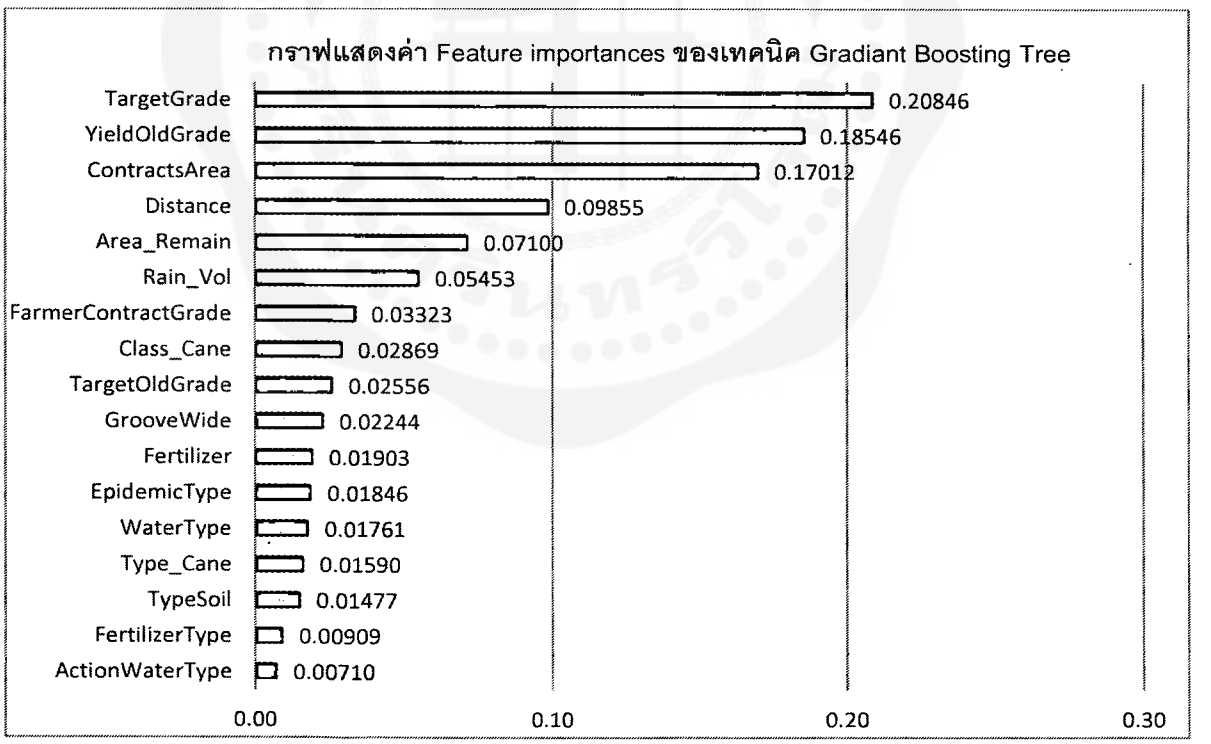


ภาพประกอบ 15 Confusion matrix และ Normalized confusion matrix จากเทคนิค Gradient boosting tree เมื่อทำ Hyper-parameter tunning

นอกจากนี้เราสามารถคำนวณหาค่าความสำคัญของแต่ละ feature ที่มีต่อแบบจำลอง (feature importances) โดยใช้แอตทริบิวต์ `"feature_importances_"` ของแบบจำลอง (ชนิด `XGBoostClassifer`) ในไลบรารี XGBoost โดยที่ถ้าค่า feature importance ของ feature ใด ยิ่งมีค่ามากจะแสดงว่า feature นั้นจะมีความสำคัญต่อการทำนายของแบบจำลอง โดยผลลัพธ์ค่า feature importance ของ feature ต่าง ๆ ของแบบจำลองที่พัฒนาในหัวข้อนี้ สามารถแสดงได้ดังภาพประกอบ 16 ซึ่งจะเห็นได้ว่า TargetGrade จะเป็น feature ที่มีความสำคัญมากสุด (ค่า importance = 0.208476)

ส่วนย่อยของโค้ด (code snippet) ที่ทำการคำนวณค่า feature importance ซึ่งใช้ไลบรารี pandas มาช่วยในการประมวลผล สามารถแสดงได้ดังนี้

```
import pandas as pd
feature_importances = pd.DataFrame(xgb1.feature_importances_,
      index = Train[Features].columns,
      columns=['importance']).sort_values('importance',
      ascending=False)
```



ภาพประกอบ 16 แสดงค่า feature importances ของแต่ละ features ของ Gradient Boosting Tree

บทที่ 5
การพัฒนาระบบการทำนายปริมาณของผลผลิตอ้อย

ในบทนี้ ผู้วิจัยจะอธิบายรายละเอียดของระบบการทำนายค่าปริมาณของผลผลิตอ้อย (Yield value prediction) ซึ่งจัดเป็นปัญหาแบบ Regression ดังที่กล่าวมาแล้วในหัวข้อที่ 3.3 ผู้วิจัยได้เลือกที่พัฒนาระบบโดยใช้อัลกอริทึม Random Forest Regression ที่ถูกพัฒนาบนไลบรารี Scikit-Learn และใช้อัลกอริทึม Gradient Boosting Tree Regression ของไลบรารี XGBoost

5.1 การทำนายที่ใช้เป็นบรรทัดฐานสำหรับการทำนายปริมาณของผลผลิตอ้อย (Baseline for Yield Value Prediction)

การเปรียบเทียบความถูกต้องในการทำนายของเทคนิคการทำนายที่ได้ถูกพัฒนาในงานวิจัยนี้ ผู้วิจัยจะนำไปเปรียบเทียบอ้างอิงกับค่าความถูกต้องการทำนายที่ใช้เป็นบรรทัดฐาน (Baseline) จำนวน 2 ตัว เช่นเดียวกันกับวิธีที่ได้อธิบายไว้ในหัวข้อ 4.1 โดยการทำนายที่ใช้เป็นบรรทัดฐาน (Baseline) ทั้ง 2 แบบนี้จะไม่ได้ใช้เทคนิคการเรียนรู้ของเครื่อง (non machine learning) ซึ่งสามารถอธิบายได้ดังนี้

(1) การทำนายผลผลิตโดยใช้ค่าผลผลิตที่ได้จากปีการผลิตก่อนหน้าในแปลงเดียวกัน โดยต่อจากนี้ผู้วิจัยขออ้างถึงโดยเรียกว่า Baseline-1

(2) การทำนายผลผลิตของฝ่ายสำรวจของโรงงานที่เป็นกรณีศึกษา โดยต่อจากนี้ผู้วิจัยขออ้างถึงโดยเรียกว่า Baseline-2

หรือกล่าวโดยเฉพาะเราจะใช้ข้อมูลในฟิลด์ `YieldOld` เป็นค่าที่ได้จากการทำนายของ Baseline-1 และเราจะใช้ข้อมูลในฟิลด์ `YieldTarget` เป็นค่าที่ได้จากการทำนายของ Baseline-2

เนื่องจากในหัวข้อนี้เป็นปัญหาแบบ Regression เราจะใช้ตัวชี้วัดที่แสดงถึงค่าความคลาดเคลื่อน (Error) ระหว่างค่าจริงของ Yield (actual yield value) กับค่าที่ทำนายได้ของ Yield (predicted yield value) ซึ่งได้อธิบายไปแล้วในหัวข้อ 2.3.2 ตัวอย่างของตัวชี้วัดที่นิยมใช้ เช่น RMSE, MAE, MAPE และ R^2 เป็นต้น สามารถแสดงส่วนย่อยของโค้ด (code snippet) ที่ใช้ในการคำนวณค่า RMSE ของ Baseline-1 และ Baseline-2 ได้ดังนี้

```
print ("Baseline_1")
print("rmse BaseLine Yield - YieldOld: %.2f" %
   np.sqrt(mean_squared_error(Data['Yield'], Data['YieldOld'])))
print ("Baseline_2")
print("rmse BaseLine Target - Yield: %.2f" %
   np.sqrt(mean_squared_error(Data['Yield'], Data['YieldTarget'])))
```

โดยเราพบว่าผลลัพธ์ค่า RMSE ของการทำนายค่าปริมาณของผลผลิตอ้อยของ Baseline-1 และ Baseline-2 เมื่อพิจารณาบน Test dataset จะมีค่าเท่ากับ 5.02 และ 2.60 ตัน/ไร่ ตามลำดับ

5.2 Random forest based yield value prediction

งานวิจัยในส่วนนี้ใช้ Random Forest Regressor เพื่อทำนายค่าปริมาณผลผลิตอ้อย โดยผู้วิจัยได้นำเสนอแบบจำลองการทำนายหลาย ๆ แบบ แล้วทำการเปรียบเทียบค่าความคลาดเคลื่อน (Error) ในการทำนาย ซึ่งรายละเอียดของแต่ละแบบจำลองสามารถอธิบายในหัวข้อย่อย 5.2.1-5.2.4

5.2.1 Random Forest Regression with default parameters

แบบจำลองการทำนายที่นำเสนอในหัวข้อนี้ เราจะกำหนดค่าพารามิเตอร์ของ Random Forest Regressor เป็นค่า default ที่กำหนดไว้บนไลบรารี Scikit-Learn และเราใช้ feature ทุกตัวที่มีในชุดข้อมูล รายละเอียดของค่าพารามิเตอร์ต่าง ๆ สามารถแสดงได้ดังนี้

```
bootstrap=True,  criterion='mse',  max_depth=None,max_features='auto',
max_leaf_nodes=None,   min_impurity_split=1e-07,   min_samples_leaf=1,
min_samples_split=2,  min_weight_fraction_leaf=0.0,  n_estimators=10,  n
n_jobs=1,         oob_score=False,         random_state=None,verbose=0,
warm_start=False
```

โดยเราพบว่าผลลัพธ์ของแบบจำลองนี้ จะได้ค่า RMSE บน Training dataset และ Test dataset คือ 1.03 และ 2.47 ตัน/ไร่ ตามลำดับ และได้ค่า MAE คือ 1.713 ตัน/ไร่ และ MAPE คือ 33.28% และ มีค่า R^2 คือ 0.61 โดยค่าดังกล่าวเป็นค่าที่ได้จากการทดสอบบน Test dataset

เนื่องจากค่า RMSE บน Training dataset มีค่าต่ำกว่าค่า RMSE บน Test dataset ค่อนข้างมาก แบบจำลองดังกล่าวจึงเกิดปัญหา overfitting ซึ่งผู้วิจัยได้นำเสนอวิธีที่จะจัดการกับปัญหา overfitting นี้ในการพัฒนาแบบจำลองในหัวข้อถัดไป

5.2.2 Random Forest Regression with manual parameters tuning

แบบจำลองการทำนายที่นำเสนอในหัวข้อนี้ จะทำการปรับค่าพารามิเตอร์หลัก ๆ ของเทคนิค Random Forest Regression ได้แก่ n_estimators, max_depth และ min_samples_leaf เพื่อพยายามที่จะลดปัญหา overfitting ที่เกิดขึ้น โดยเราได้ทดลองเปลี่ยนค่าพารามิเตอร์เหล่านี้เป็นค่าที่ได้จากการปรับพารามิเตอร์ของแบบจำลองที่ได้จากการทำ Hyper-parameter tuning ของ Random Forest Classifier ที่ได้อธิบายไปในหัวข้อ 4.2.3 ซึ่งค่าพารามิเตอร์ดังกล่าวจะมีค่าดังต่อไปนี้

- n_estimators = 300
- min_samples_leaf = 14
- max_depth = 10

ส่วนย่อยของโค้ด (code snippet) สำหรับการพัฒนาแบบจำลองนี้สามารถแสดงได้ดังนี้

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators = 300,
                           min_samples_leaf = 14,
                           max_depth = 10)
rf.fit(train_selFeat, y_train)
print ("rmse train: : %.2f" %
       np.sqrt(mean_squared_error(y_train,
rf.predict(train_selFeat))))
print ("rmse test: : %.2f" %
       np.sqrt(mean_squared_error(y_test, rf.predict(test_selFeat))))
print ("MAE: : %.2f" % mean_absolute_error(y_test,
       rf.predict(test_selFeat)))
print ("MAPE: : %.2f" % mean_absolute_percentage_error(y_test,
       rf.predict(test_selFeat)))
print ("Variance score: %.2f" % r2_score(y_test,
       rf.predict(test_selFeat)))
```

ผลลัพธ์การทำนายของแบบจำลองนี้ จะได้ค่า RMSE เมื่อทดสอบบน training dataset และ test dataset เท่ากับ 2.09 และ 2.20 ตัน/ไร่ ตามลำดับ และได้ค่า MAE เท่ากับ 1.61 ตัน/ไร่ และ MAPE เท่ากับ 32.25% โดยมีค่า R^2 เท่ากับ 0.660 (ค่า MAE, MAPE และ R^2 ดังกล่าวเป็นค่าที่ได้จากการทดสอบบน Test dataset

จะเห็นได้ว่าค่า RMSE ของ Train dataset และ Test dataset มีค่าใกล้เคียงกันและค่า RMSE บน test dataset ของแบบจำลองนี้ จะมีค่าน้อยกว่าค่าที่ได้จากแบบจำลองที่พัฒนาในหัวข้อที่

แล้ว (ค่า RMSE เท่ากับ 2.47 ตัน/ไร่) แสดงว่าการปรับค่าพารามิเตอร์ดังกล่าวสามารถช่วยลดปัญหา overfitting ได้ในระดับหนึ่ง

5.2.3 Random Forest Regression with Hyper-parameter tuning

แบบจำลองการทำนายที่นำเสนอในหัวข้อนี้ เราจะใช้เทคนิคในการปรับพารามิเตอร์ (Hyper-parameter tuning) แบบ Grid search มาช่วยในการหาค่าที่เหมาะสมให้กับแบบจำลอง เพื่อลดปัญหา overfitting ซึ่งเทคนิค Grid search นี้จะปรับค่าพารามิเตอร์ต่าง ๆ หลายตัวพร้อมกัน แล้วทำการเทรนแบบจำลองและวัดผลค่าความถูกต้องในการทำนายของแบบจำลองโดยใช้เทคนิค K-fold cross-validation (K = 5) แล้วอัลกอริทึมจะเลือกชุดของค่าพารามิเตอร์ที่ดีที่สุดที่ทำให้ ค่า cross-validation score มีค่ามากสุด โดยในกรณีของ Random Forest Regression เราจะกำหนดให้ใช้ค่า Negative Mean Squared Error เป็นค่า Score ของการทำ Cross Validation โดยการทำงานตามคำอธิบายด้านบนสามารถทำได้โดยเรียกใช้ฟังก์ชัน GridSearchCV ของไลบรารี Scikit-Learn

สำหรับพารามิเตอร์ของเทคนิค Random Forest Regression ที่จะพิจารณาในขั้นตอน Grid Search จะเหมือนกับแบบจำลองที่พัฒนาสำหรับปัญหา Classification ที่ได้อธิบายไปในหัวข้อ 4.2.3 นั้นคือประกอบด้วยพารามิเตอร์ n_estimators, max_depth และ min_samples_leaf โดยช่วงของค่าพารามิเตอร์แต่ละตัวที่ทำการค้นหาสามารถแสดงได้ดังนี้

n_estimators	100, 200, 300, 400, 500, 600, 700, 800, 900
max_depth	5, 10, 15
min_samples_leaf	10, 12, 14, 16, 18

ส่วนย่อยของโค้ด (code snippet) สำหรับการพัฒนาแบบจำลองนี้สามารถแสดงได้ดังนี้

```
from sklearn.model_selection import GridSearchCV
parameters = {'max_depth':np.arange(5,20,5),
              'n_estimators': np.arange(100,1000,100),
              'min_samples_leaf': np.arange(10,20,2)}
clf_rf = RandomForestRegressor()
```

```
clf_grid = GridSearchCV(clf_rf, parameters,
                        scoring='neg_mean_squared_error', cv=5)
clf_grid.fit(train_selFeat,y_train)
clf_grid.predict(test_selFeat)
print ("Best: %f using %s" % (grid.best_score_, grid.best_params_))
print ("rmse train: : %.2f" %
       np.sqrt(mean_squared_error(y_train,
grid.predict(train_selFeat))))
print ("rmse test: : %.2f" %
       np.sqrt(mean_squared_error(y_test,
grid.predict(test_selFeat))))
print ("MAE: : %.2f" %
       mean_absolute_error(y_test, clf_grid.predict(test_selFeat)))
print ("MAPE: : %.2f" % mean_absolute_percentage_error(y_test,
                                clf_grid.predict(test_selFeat)))
print ("Variance score: %.2f" %
       r2_score(y_test, clf_grid.predict(test_selFeat)))
```

ผลลัพธ์ของการปรับจูนแบบจำลองโดยใช้ Grid search และ Cross validation จะทำให้เราได้แบบจำลองที่มีค่า cross valdation score มากสุด เนื่องจากเรากำหนด Negative Mean Squared Error เป็นค่า Score ของการทำ Cross validation ดังนั้นแบบจำลองที่ได้จะมีค่าเฉลี่ยของ Mean Squared Error ในแต่ละ fold ที่น้อยที่สุด โดยค่าของของพารามิเตอร์ของแบบจำลองดังกล่าว แสดงดังนี้

- max_depth = 10
- n_estimators = 200
- min_samples_leaf = 20

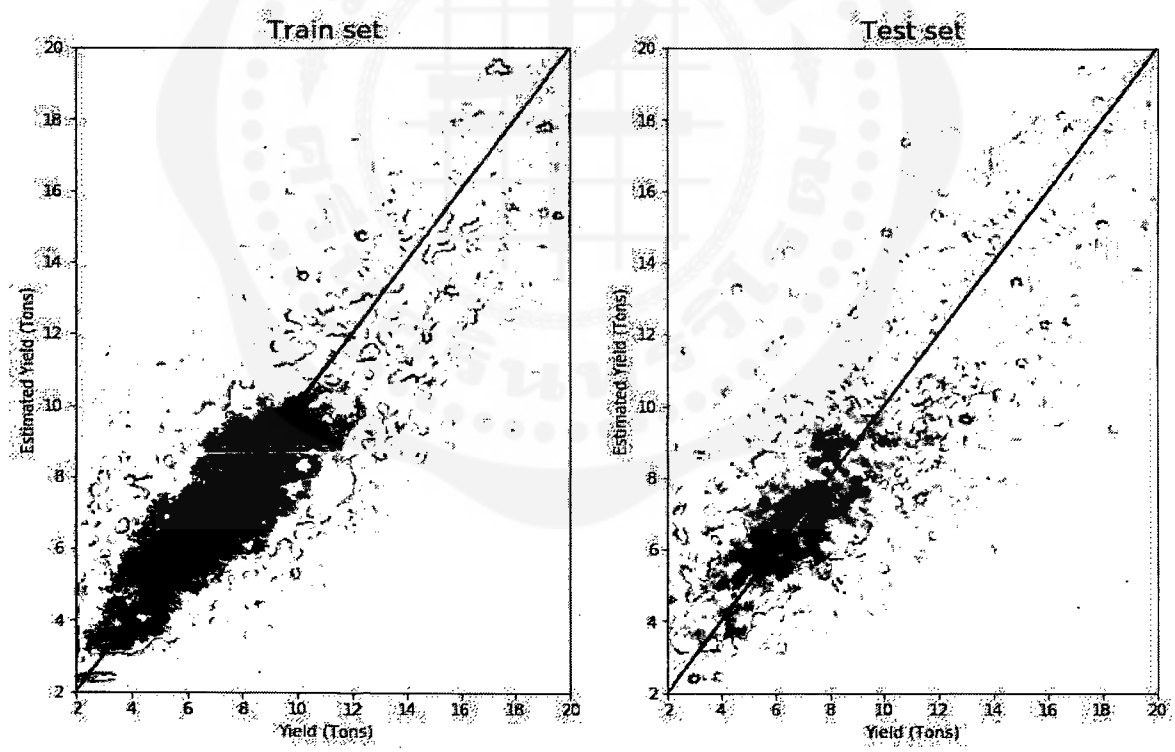
เมื่อนำแบบจำลองดังกล่าวไปทดสอบ training dataset กับ test dataset เราจะได้ค่า RMSE เท่ากับ 2.15 และ 2.18 ตัน/ไร่ ตามลำดับ และได้ค่า MAE เท่ากับ 1.61 ตัน/ไร่ และ MAPE เท่ากับ 32.24% โดยมีค่า R^2 เท่ากับ 0.70 (ค่า MAE, MAPE แล R^2ดังกล่าวเป็นค่าที่ได้จากการทดสอบบน Test dataset)

จะเห็นได้ว่าค่า RMSE ของ Train dataset และ Test dataset มีค่าใกล้เคียงกัน และค่า RMSE บน test dataset ของแบบจำลองนี้ (ค่า RMSE เท่ากับ 2.18 ตัน/ไร่) จะมีค่าน้อยกว่าค่าที่ได้จากแบบจำลองที่พัฒนาในหัวข้อ 5.2.1 (ค่า RMSE เท่ากับ 2.47 ตัน/ไร่) แสดงว่าการปรับค่าพารามิเตอร์โดยใช้ Hyper-parameter tuning ดังกล่าวสามารถช่วยลดปัญหา overfitting ได้

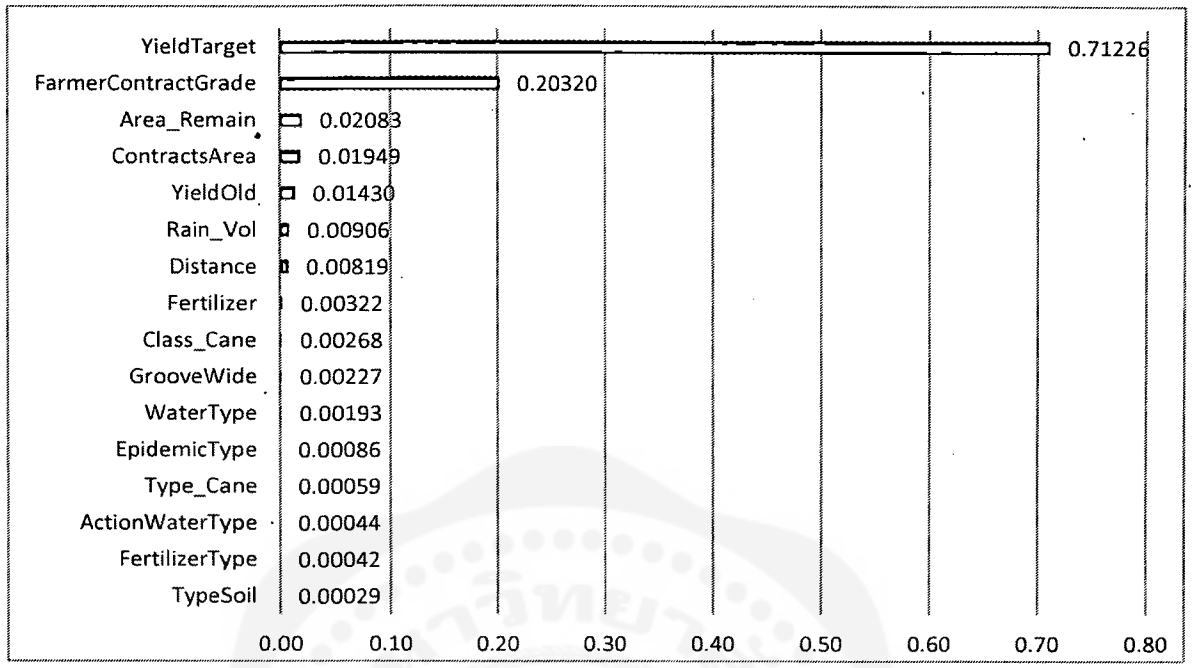
นอกจากนี้เมื่อเปรียบเทียบกับแบบจำลองที่พัฒนาในหัวข้อ 5.2.2 เราจะพบว่าแบบจำลองนี้มีความแม่นยำกว่า เนื่องจากค่า RMSE ต่ำกว่า และค่า R^2 สูงกว่า

ผู้วิจัยได้แสดงผลลัพธ์การทำนายในรูปของคู่ของค่าข้อมูลจริง (แกน x) กับค่าข้อมูลจริงที่ทำนายได้ (แกน y) เมื่อทดสอบบน Train dataset และ Test dataset ดังภาพประกอบ 17 ซึ่งจะแสดงให้เห็นถึงการกระจายของค่าข้อมูลที่ทำนายเทียบกับเส้นตรง x=y ที่เป็นการทำนายในอุดมคติ (perfect prediction) ที่การทำนายถูกต้องหมดไม่มีค่าความคลาดเคลื่อน

นอกจากนี้เราได้คำนวณค่า Feature importances ของแบบจำลองที่พัฒนาในหัวข้อนี้ โดยผลลัพธ์ค่า feature importance ของ feature ต่าง ๆ สามารถแสดงได้ดังภาพประกอบ 18 ซึ่งจะเห็นได้ว่า YieldTarget จะเป็น feature ที่มีความสำคัญมากสุด ด้วยค่า importance=0.71226 และ FarmerContractGrade จะสำคัญเป็นลำดับรองลงไปด้วยค่า importance=0.20320 ซึ่งสอดคล้องกับแบบจำลองที่ใช้ Random Forest สำหรับปัญหาการทำนายเกรดผลผลิตที่ได้อธิบายไปในหัวข้อ 4.2.3



ภาพประกอบ 17 กราฟแสดงคู่ของค่าข้อมูลจริง (แกน x) และค่าข้อมูลที่ทำนายได้ (แกน y) ของ Train dataset และ Test dataset จากเทคนิค Random Forest Regression with Hyper-parameter Tuning



ภาพประกอบ 18 ค่า feature importance ของแบบจำลอง Random Forest Regression with Hyper-parameter tuning

5.2.4 Random Forest Regression with Dummy Features

ในหัวข้อนี้เราจะเทรนแบบจำลองการทำนายโดยการเปลี่ยนข้อมูลในส่วนของ feature ที่มีชนิดเป็น Category ให้เป็นข้อมูลที่เป็น Dummy feature ด้วยวิธี one-hot-encoding และใช้ค่าพารามิเตอร์ที่ได้จากการทำ Hyper-parameter tuning ในหัวข้อ 5.2.3 เพื่อที่จะศึกษาว่าค่าประสิทธิผลในการทำนายระหว่างกรณีที่ไม่ใช้และกรณีที่ใช้ Dummy feature ว่ามีผลต่างกันอย่างไร

จากผลการทดลอง เราจะได้ผลลัพธ์ค่า RMSE เมื่อทดสอบ train dataset และ test dataset คือ 2.15 และ 2.18 ตัน/ไร่ ตามลำดับ และได้ค่า MAE = 1.61 ตัน/ไร่ และ MAPE = 32.27% โดยมีค่า R^2 = 0.70 (ค่า MAE, MAPE และ R^2 ดังกล่าวเป็นค่าที่ได้จากการทดสอบบน Test dataset)

เมื่อเปรียบเทียบผลลัพธ์ของค่าดังกล่าวกับผลลัพธ์ที่ได้เมื่อไม่ได้ใช้ Dummy feature ในหัวข้อ 5.2.3 (RMSE = 2.18 ตัน/ไร่, MAE = 1.61 ตัน/ไร่, MAPE = 32.24% และค่า R^2 = 0.70) จะเห็นได้ว่าค่าต่าง ๆ มีค่าใกล้เคียงกัน ยกเว้นค่า MAPE มีค่าเพิ่มขึ้นเล็กน้อย แสดงว่าการใช้ dummy feature ด้วยวิธี one-hot-encoding ของข้อมูลชุดนี้ ไม่มีผลทำให้การทำนายดีขึ้น (ค่า RMSE ลดลง) ซึ่งสอดคล้องกับแนวคิดที่กล่าวไว้ในหัวข้อ 3.3.3 ว่าชุดของข้อมูลที่นำมาใช้ในงานวิจัยนี้จะมี feature

ส่วนใหญ่เป็นแบบ Category ที่จะส่งผลให้การทำนายค่าผลผลิตอ้อยต่อไร่ซึ่งเป็นการทำนายเชิงปริมาณอาจได้ผลไม่ดีมากนัก

5.3 Gradient boosting tree based yield value prediction

ในหัวข้อนี้เรานำเสนอการพัฒนาแบบจำลองการทำนายค่าปริมาณผลผลิตอ้อยโดยใช้เทคนิค Gradient Boosting Tree Regression ของไลบรารี XGBoost เพื่อทำนายค่าตัวแปร Yield โดยผู้วิจัยได้นำเสนอแบบจำลองการทำนายหลาย ๆ แบบ ในลักษณะเดียวกับการสร้างแบบจำลองการทำนายโดยใช้ Random Forest Regressor ในหัวข้อ 5.2 แล้วจึงทำการเปรียบเทียบค่าความถูกต้องในการทำนาย ซึ่งรายละเอียดของแต่ละแบบจำลองสามารถอธิบายในหัวข้อย่อย 5.3.1 – 5.3.4

5.3.1 Gradient Boosting Tree Regression with default parameters

แบบจำลองการทำนายที่นำเสนอในหัวข้อนี้ เราจะกำหนดค่าพารามิเตอร์ของ Gradient Boosting Tree Regressor เป็นค่า default ที่กำหนดไว้บนไลบรารี XGBoost (XGBRegressor) และเราใช้ feature ทุกตัวที่มีในชุดข้อมูล รายละเอียดของค่าพารามิเตอร์ต่าง ๆ สามารถแสดงได้ดังนี้

```
base_score=0.5, colsample_bylevel=1, colsample_bytree=1, gamma=0,
learning_rate=0.1, max_delta_step=0, max_depth=3, min_child_weight=1,
missing=None, n_estimators=100, nthread=-1, objective='reg:linear',
reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=0, silent=True,
subsample=1
```

โดยเราพบว่าผลลัพธ์ของแบบจำลองนี้ จะได้ค่า RMSE บน Training dataset และ Test dataset คือ 2.10 และ 2.35 ตัน/ไร่ ตามลำดับและได้ค่า MAE คือ 1.61 ตัน/ไร่ และ MAPE คือ 33.82 %และมีค่า R^2 คือ 0.66 โดยค่าดังกล่าวเป็นค่าที่ได้จากการทดสอบบน Test dataset

จากค่า RMSE บน Training dataset และ Test dataset ดังกล่าว เราพบว่าค่าทั้งสองไม่แตกต่างกันมาก จึงไม่เกิดปัญหา overfitting อย่างเช่น แบบจำลองที่ใช้ Random Forest Regression ที่เทรนแบบจำลองโดยใช้ค่า default parameters ที่ได้นำเสนอไปในหัวข้อ 5.1.1

5.3.2 Gradient Boosting Tree Regression with manual set parameters

แบบจำลองการทำนายที่นำเสนอในหัวข้อนี้ จะทำการปรับค่าพารามิเตอร์หลัก ๆ ของเทคนิค Gradient Boosting Tree (XGBoost) ได้แก่ n_estimators, max_depth, learning_rate และ min_child_weight เพื่อดูว่าประสิทธิผลในการทำนายมีการเปลี่ยนแปลงอย่างไร โดยเราได้ทดลองเปลี่ยนค่าพารามิเตอร์เหล่านี้เป็นค่าที่ได้จากการปรับพารามิเตอร์ของแบบจำลองที่ได้จากการทำ

Hyper-parameter tuning ของ Gradient Boosting Tree Classifier ที่ได้อธิบายไปในหัวข้อ 4.3.3 ซึ่งค่าพารามิเตอร์ดังกล่าวจะมีค่าดังต่อไปนี้

- n_estimators = 100
- learning_rate = 0.1
- max_depth = 4
- min_child_weight = 4

ส่วนย่อยของโค้ด (code snippet) สำหรับการพัฒนาแบบจำลองนี้สามารถแสดงได้ดังนี้

```
import xgboost as xgb
xgb1 = xgb.XGBRegressor(max_depth = 4, n_estimators=100,
                        learning_rate = 0.1, min_child_weight = 4)
xgb1.fit(train_selFeat, y_train)
print ("rmse train: : %.2f" %
       np.sqrt(mean_squared_error(y_train,
xgb1.predict(train_selFeat))))
print ("rmse test: : %.2f" %
       np.sqrt(mean_squared_error(y_test,
xgb1.predict(test_selFeat))))
print ("MAE: : %.2f" % mean_absolute_error(y_test,
      xgb1.predict(train_selFeat)))
print ("MAPE: : %.2f" % mean_absolute_percentage_error(y_test,
      xgb1.predict(train_selFeat)))
print ("Variance score: %.2f" % r2_score(y_test,
      clf_grid.predict(test_selFeat)))
```

ผลลัพธ์การทำนายของแบบจำลองนี้ ได้ค่า RMSE เมื่อทดสอบบน training dataset และ test dataset เท่ากับ 2.01 และ 2.20 ตัน/ไร่ ตามลำดับและได้ค่า MAE เท่ากับ 1.60 ตัน/ไร่ และ MAPE เท่ากับ 32.03% โดยมีค่า R^2 เท่ากับ 0.70 (ค่า MAE, MAPE และ R^2 ดังกล่าวเป็นค่าที่ได้จากการทดสอบบน Test dataset)

จากผลลัพธ์เห็นได้ว่าค่า RMSE และ ค่า MAPE บน test dataset ของแบบจำลองนี้ มีค่าน้อยกว่าค่าที่ได้จากแบบจำลองที่พัฒนาในหัวข้อ 5.3.1 (ค่า RMSE =2.35 ตัน/ไร่, ค่า MAPE=33.82%) แสดงว่าการปรับค่าพารามิเตอร์ดังกล่าวสามารถช่วยให้แบบจำลองทำการทำนายได้แม่นยำมากขึ้น

5.3.3 Gradient Boosting Tree Regression with hyper-parameter tuning

แบบจำลองการทำนายที่นำเสนอในหัวข้อนี้ เราจะใช้เทคนิคในการปรับพารามิเตอร์ (Hyper-parameter tuning) แบบ Grid search มาช่วยในการหาค่าที่เหมาะสมของแบบจำลอง Gradient Boosting Tree Regression เช่นเดียวกับในกรณีของแบบจำลอง Random Forest Regression ในหัวข้อ 5.2.3

กล่าวโดยเฉพาะเราจะใช้เทคนิค Grid search ร่วมกับ K-Fold Cross Validation (K=5) โดยการใช้ฟังก์ชัน `GridSearchCV` ของไลบรารี Scikit Learn เพื่อจะปรับหาพารามิเตอร์ที่ดีที่สุดของแบบจำลองแบบ `XGBRegressor` ของไลบรารี XGBoost

สำหรับพารามิเตอร์ของเทคนิค Gradient Boosting Tree Regression ที่จะพิจารณาในขั้นตอน Grid Search และช่วงของค่าพารามิเตอร์แต่ละตัวที่ทำการค้นหาสามารถแสดงได้ดังนี้

n_estimators	100, 200, 300, 400
max_depth	2, 4, 6, 8
learning_rate	0.1, 0.2, 0.3, 0.4
min_child_weight	2, 3, 4, 5, 6, 7, 8, 9

ส่วนย่อยของโค้ด (code snippet) สำหรับการพัฒนาแบบจำลองนี้สามารถแสดงได้ดังนี้

```
parameters = {'max_depth':np.arange(2,10,2),
              'n_estimators': np.arange(100,500,100),
              'min_child_weight': np.arange(2,10,1),
              'learning_rate': np.arange(0.1,0.4,0.1)}
xgb1 = xgb.XGBRegressor()
grid = GridSearchCV(estimator=xgb1,param_grid=parameters, cv=5)
grid.fit(train_selFeat, y_train)
print ("Best: %f using %s" % (grid.best_score_, grid.best_params_))
print ("rmse train: : %.2f" %   np.sqrt(mean_squared_error(y_train,
      grid.predict(train_selFeat))))
print ("rmse test: : %.2f" % np.sqrt(mean_squared_error(y_test,
      grid.predict(test_selFeat),)))
```

```
print ("MAE: : %.2f" % mean_absolute_error(y_test,
      grid.predict(train_selFeat)))
print ("MAPE: : %.2f" % mean_absolute_percentage_error(y_test,
      grid.predict(train_selFeat)))
print ("Variance score: %.2f" % r2_score(y_test,
      clf_grid.predict(test_selFeat)))
```

ผลลัพธ์ของการปรับจูนแบบจำลองโดยใช้ Grid search และ Cross validation จะทำให้เราได้แบบจำลองที่มีค่าของของพารามิเตอร์ของแบบจำลอง ดังต่อไปนี้

- max_depth=5
- n_estimators=100
- learning_rate=0.1
- min_child_weight=3

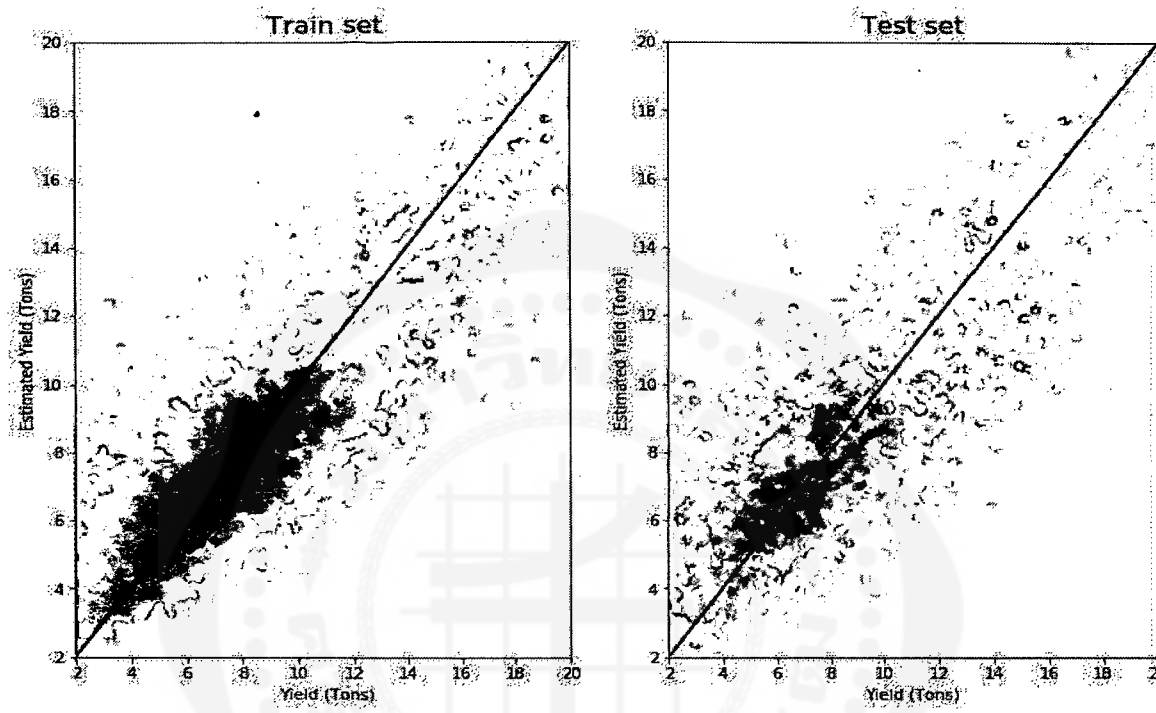
เมื่อนำแบบจำลองดังกล่าวไปทดสอบ training dataset กับ test dataset เราจะได้ค่า RMSE เท่ากับ 1.87 และ 2.19 ตัน/ไร่ ตามลำดับ และได้ค่า MAE เท่ากับ 1.60 ตัน/ไร่ และ MAPE เท่ากับ 31.91% โดยมีค่า R^2 เท่ากับ 0.70 (ค่า MAE, MAPE และ R^2 ดังกล่าวเป็นค่าที่ได้จากการทดสอบบน Test dataset)

เมื่อเปรียบเทียบกับแบบจำลองที่พัฒนาในหัวข้อ 5.3.1 เราจะพบว่าแบบจำลองนี้มีความแม่นยำกว่า เนื่องจากค่า RMSE ต่ำกว่า และค่า R^2 สูงกว่า (RMSE และ R^2 ของแบบจำลองในหัวข้อ 5.3.1 เท่ากับ 2.35 ตัน/ไร่ และ0.66 ตามลำดับ) แสดงว่าการปรับค่าพารามิเตอร์โดยใช้ Hyper-parameter tuning จะช่วยทำให้แบบจำลองมีความแม่นยำมากขึ้น แต่ถ้าเปรียบเทียบกับแบบจำลองที่พัฒนาในหัวข้อ 5.3.2 ความแม่นยำของทั้งสองแบบจำลองยังไม่แตกต่างกันมากนัก

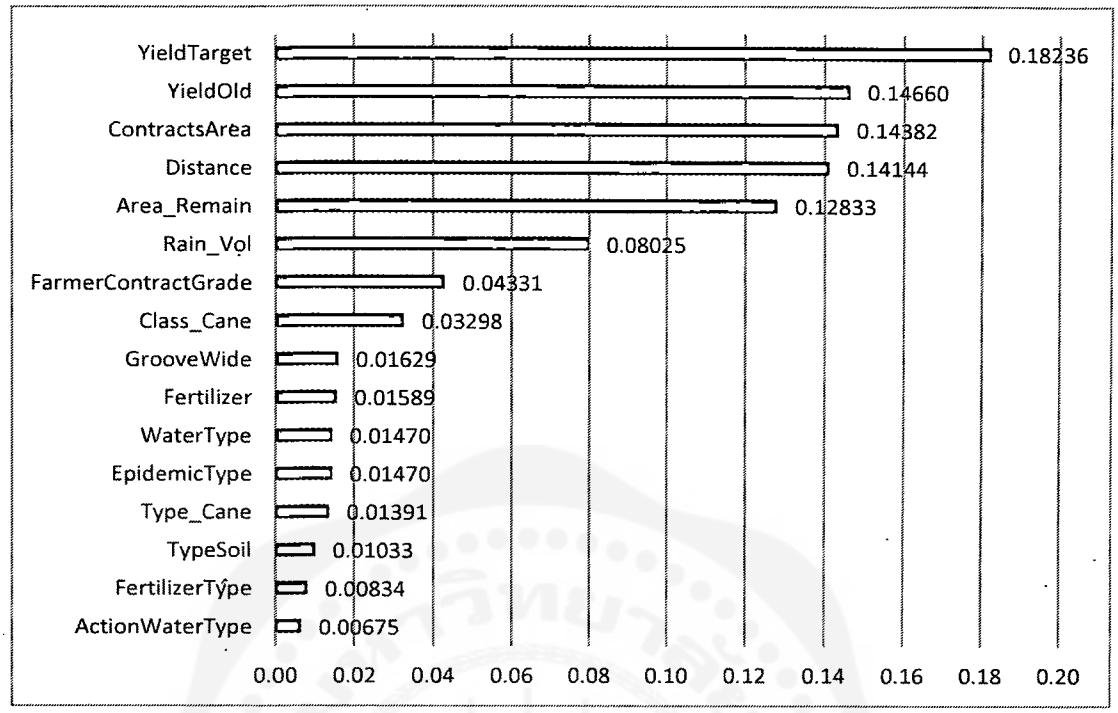
นอกจากนี้ ผู้วิจัยได้นำเสนอผลลัพธ์ประกอบอื่น ๆ ในลักษณะเดียวกับสิ่งที่ได้นำเสนอในหัวข้อ 5.2.3 กล่าวคือผู้วิจัยได้แสดงผลลัพธ์การทำนายในรูปของคู่ของค่าข้อมูลจริง (แกน x) กับค่าข้อมูลจริงที่ทำนายได้ (แกน y) เมื่อทดสอบบน Train dataset และ Test dataset ดังภาพประกอบ 19 ซึ่งจะแสดงให้เห็นถึงการกระจายของค่าข้อมูลที่ทำนายเทียบกับเส้นตรง x=y ที่เป็นการทำนายในอุดมคติ (perfect prediction) ที่การทำนายถูกต้องหมดไม่มีค่าความคลาดเคลื่อน

นอกจากนี้ เราได้คำนวณค่า Feature importances ของแบบจำลองที่พัฒนาในหัวข้อนี้ โดยผลลัพธค่า feature importance ของ feature ต่าง ๆ สามารถแสดงได้ดังภาพประกอบ 20 ซึ่งจะเห็นได้ว่าแบบจำลองที่ใช้ Gradient Boosting Tree สำหรับทำ Regression จะมีจำนวน Feature ที่มีค่า importance สูง ๆ อยู่มากกว่ากรณีของแบบจำลองที่ Random Forest ที่ได้อธิบายไปในหัวข้อ 5.2.3

โดย Feature ที่มีค่าสูง 3 ลำดับแรก ได้แก่ YieldTarget, YieldOld และ ContractArea ซึ่งมีค่า feature importance เท่ากับ 0.18236, 0.14466 และ 0.14382 ตามลำดับ ซึ่งสอดคล้องกับแบบจำลองที่ใช้ Gradient Boosting Tree สำหรับปัญหาการทำนายเกรดผลผลิตที่ได้อธิบายไปในหัวข้อ 4.3.3



ภาพประกอบ 19 กราฟแสดงคู่ของค่าข้อมูลจริง (แกน x) และ ข้อมูลที่ทำนายได้ (แกน y) ของ Training dataset และ Test dataset จากเทคนิค Gradient Boosting Tree Regression with hyper-parameter tuning



ภาพประกอบ 20 Feature importances จากเทคนิค Gradient Boosting Tree Regression with hyper-parameter tuning

5.3.4 Gradient Boosting Tree Regression with Dummy Features

ในหัวข้อนี้ เราทดลองสร้างแบบจำลองในลักษณะเดียวกับสิ่งที่ได้นำเสนอในหัวข้อ 5.1.4 กล่าวคือเราจะเทรนแบบจำลองการทำนายโดยการเปลี่ยนข้อมูลในส่วนของ feature ที่มีชนิดเป็น Category ให้เป็นข้อมูลที่เป็น Dummy feature ด้วยวิธี one-hot-encoding และ ใช้ค่าพารามิเตอร์ที่ได้จากการทำ Hyper-parameter tuning ในหัวข้อ 5.3.3 เพื่อที่จะศึกษาว่าค่าประสิทธิผลในการทำนายระหว่างกรณีที่ไม่ใช้และกรณีที่ใช้ Dummy feature ว่ามีผลต่างกันอย่างไร

จากผลการทดลองเราได้ผลลัพธ์ค่า RMSE เมื่อทดสอบ train dataset และ test dataset คือ 1.87 และ 2.20 ตัน/ไร่ ตามลำดับและได้ค่า MAE = 1.59 ตัน/ไร่ และ MAPE = 31.95% โดยมีค่า R^2 = 0.70 (ค่า MAE, MAPE และ R^2 ดังกล่าวเป็นค่าที่ได้จากการทดสอบบน Test dataset)

เมื่อเปรียบเทียบผลลัพธ์ของค่าดังกล่าวกับผลลัพธ์ที่ได้เมื่อไม่ได้ใช้ Dummy feature ในหัวข้อ 5.3.3 (RMSE = 2.19 ตัน/ไร่, MAE = 1.60 ตัน/ไร่, MAPE = 31.91% และ ค่า R^2 = 0.70) จะเห็นได้ว่าค่าต่าง ๆ มีค่าใกล้เคียงกัน แสดงว่าการใช้ dummy feature ด้วยวิธี one-hot-encoding ของข้อมูลชุดนี้ ไม่มีผลทำให้การทำนายดีขึ้น (ค่า RMSE ลดลง) เช่นเดียวกับข้อสรุปที่กล่าวไว้ในหัวข้อ 5.3.3

บทที่ 6
สรุปผลงานวิจัยและข้อเสนอแนะ

งานวิจัยนี้นำเสนอการทำนายปริมาณผลผลิตอ้อย โดยนิยามเป็นปัญหา 2 แบบ ได้แก่ ปัญหาการทำนายเกรดผลผลิตอ้อยของแต่ละแปลง (Yield grade prediction) ซึ่งจัดเป็นปัญหาแบบ Classification และปัญหาการทำนายค่าปริมาณผลผลิตอ้อยในหน่วยตันต่อไร่ของแต่ละแปลง (Yield value prediction) ซึ่งจัดเป็นปัญหาแบบ Regression โดยในการพัฒนาระบบการทำนายแต่ละแบบ ผู้วิจัยได้เลือกใช้เทคนิคการเรียนรู้ของเครื่อง (Machine learning) จำนวน 2 วิธี คือ เทคนิค Random Forest และเทคนิค Gradient Boosting Tree แล้วทำการเปรียบเทียบผลการทำนายที่ได้ รวมทั้งเปรียบเทียบผลที่ได้กับค่า Baseline ที่เป็นการทำนายโดยไม่ได้ใช้เทคนิคการเรียนรู้ของเครื่อง จำนวน 2 แบบได้แก่ (1) Baseline-1 ซึ่งเป็นการทำนายผลผลิตโดยใช้ค่าผลผลิตที่ได้จริงจากปีการผลิตก่อนหน้าในแปลงเดียวกัน และ (2) Baseline-2 จะเป็นการทำนายผลผลิตของฝ่ายสำรวจของโรงงานที่เป็นกรณีศึกษา

ในบทนี้ ผู้วิจัยจะอธิบายสรุปผลการวิจัยของการพัฒนาระบบการทำนายเกรดของปริมาณอ้อย (Yield grade prediction) ในหัวข้อย่อยที่ 6.1 และจะอธิบายสรุปผลการวิจัยของระบบการทำนายค่าปริมาณผลผลิตอ้อย (Yield value prediction) ในหัวข้อย่อยที่ 6.2 และจะอภิปรายเกี่ยวกับปัญหาและข้อเสนอแนะ ในหัวข้อย่อยที่ 6.3

6.1 สรุปผลของระบบการทำนายเกรดของปริมาณอ้อย (Yield grade prediction)

งานวิจัยในส่วนนี้ผู้วิจัยได้พัฒนาอัลกอริทึมหรือแบบจำลองการทำนายที่แตกต่างกันจำนวน 3 แบบ ซึ่งสามารถสรุปได้ดังนี้

(1) Random Forest with hyper-parameter tuning

อัลกอริทึมหรือแบบจำลองการทำนายที่ใช้เทคนิค Random Forest Classification และใช้เทคนิค Hyper-parameter tuning แบบ Grid search ที่ทำงานร่วมกับเทคนิค K-Fold Cross Validation เพื่อให้ได้ชุดของพารามิเตอร์ที่นำไปสู่แบบจำลองที่มีความถูกต้องในการทำนายมากที่สุด รายละเอียดของแบบจำลองการทำนายแบบนี้อธิบายอยู่ในหัวข้อที่ 4.2.3

(2) Random Forest with forward feature selection and hyper-parameter tuning

อัลกอริทึมหรือแบบจำลองการทำนายที่ใช้เทคนิค Random Forest Classification และใช้เทคนิค Forward Feature Selection เพื่อให้ได้ชุดของ Feature ที่เหมาะสมต่อการสร้างแบบจำลองที่สุด หลังจากนั้นจะใช้เทคนิค Hyper-parameter tuning แบบ Grid search ที่ทำงานร่วมกับเทคนิค K-Fold Cross Validation เพื่อให้ได้ชุดของพารามิเตอร์ที่เมื่อประยุกต์กับข้อมูลที่มีเฉพาะ Feature ที่เลือกมาแล้วนำไปสู่แบบจำลองที่มีความถูกต้องในการทำนายมากที่สุด รายละเอียดของแบบจำลองการทำนายแบบนี้อธิบายอยู่ในหัวข้อที่ 4.2.4

(3) Gradient Boosting Tree with hyper-parameter tuning

อัลกอริทึมหรือแบบจำลองการทำนายที่ใช้เทคนิค Gradient Boosting Tree Classification โดยการใช้เทคนิค Hyper-parameter tuning แบบ Grid search ที่ทำงานร่วมกับเทคนิค K-Fold Cross Validation เพื่อให้ได้ชุดของพารามิเตอร์ที่นำไปสู่แบบจำลองที่มีความถูกต้องในการทำนายมากที่สุด รายละเอียดของแบบจำลองการทำนายแบบนี้อธิบายอยู่ในหัวข้อที่ 4.3.3

การประเมินผลความแม่นยำในการทำนายจะใช้ตัวชี้วัดคือค่าความถูกต้อง (Accuracy) ในการทำนายเกรด ซึ่งผลลัพธ์ค่า Accuracy ในการทำนายข้อมูลใน Test dataset ของ Baseline-1, Baseline-2 และทั้งสามแบบจำลองข้างต้น สามารถสรุปได้ดังตารางที่ 16 และผลลัพธ์ในรูปของ Confusion Metrix สามารถสรุปได้ดังตารางที่ 17 ถึง 21

ตาราง 16 สรุปค่า accuracy ของ baseline และของแบบจำลองการทำนายแบบต่าง ๆ

Method	Accuracy
Baseline-1	51.52%
Baseline-2	65.50%
Random Forest with hyper-parameter tuning	*71.79%*
Random Forest with forward feature selection and hyper-parameter tuning	*71.88%*
Gradient Boosting Tree with hyper-parameter tuning	*71.71%*

จากผลการทดลองเราพบว่าแบบจำลองการทำนายที่เราพัฒนาขึ้นทั้ง 3 แบบ มีค่า Accuracy ประมาณ 72% ซึ่งมีค่าสูงกว่า Baseline-1และ Baseline-2 ซึ่งทำให้เราสรุปได้ว่าแบบจำลองการทำนายโดยใช้เทคนิคการเรียนรู้ของเครื่องที่เราพัฒนาขึ้น มีความถูกต้องแม่นยำกว่าการทำนายโดยการประมาณผลผลิตของฝ่ายสำรวจของโรงงาน (หรือ Baseline-2 ซึ่งมีค่า accuracy ประมาณ 65.5%) โดยแบบจำลองการทำนายที่เราพัฒนาขึ้นจะมีความถูกต้องกว่าประมาณ 6.5%

ตาราง 17 Confusion matrix ของ Baseline-1

Actual class	Predicted Class		
	Grade 1	Grade 2	Grade 3
Grade 1	**926 (0.541)**	647 (0.378)	139 (0.081)
Grade 2	497 (0.310)	**862 (0.536)**	248 (0.154)
Grade 3	75 (0.172)	225 (0.515)	**137 (0.313)**

ตาราง 18 Confusion matrix ของ Baseline-2

Actual class	Predicted Class		
	Grade 1	Grade 2	Grade 3
Grade 1	**885 (0.517)**	825 (0.482)	2(0.001)
Grade 2	403 (0.251)	**1182 (0.736)**	21 (0.013)
Grade 3	10 (0.023)	230 (0.526)	**197 (0.451)**

ตาราง 19 Confusion matrix ของแบบจำลอง Random Forest with hyper-parameter tuning

Actual class	Predicted Class		
	Grade 1	Grade 2	Grade 3
Grade 1	**1307 (0.764)**	403 (0.235)	2 (0.001)
Grade 2	394 (0.245)	**1184(0.737)**	29 (0.018)
Grade 3	12 (0.027)	229 (0.524)	**196 (0.449)**

ตาราง 20 Confusion matrix ของแบบจำลอง Random Forest with forward feature selection and hyper-parameter tuning

Actual class	Predicted Class		
	Grade 1	Grade 2	Grade 3
Grade 1	**1303 (0.761)**	407 (0.238)	2 (0.001)
Grade 2	388 (0.241)	**1190 (0.741)**	29 (0.018)
Grade 3	10 (0.023)	230(0.526)	**197 (0.451)**

ตาราง 21 Confusion matrix ของแบบจำลอง Gradient Boosting Tree with hyper-parameter tuning

Actual class	Predicted Class		
	Grade 1	Grade 2	Grade 3
Grade 1	**1301 (0.760)**	409 (0.239)	2 (0.001)
Grade 2	394 (0.245)	**1183 (0.736)**	30 (0.019)
Grade 3	10 (0.023)	229(0.524)	**198 (0.453)**

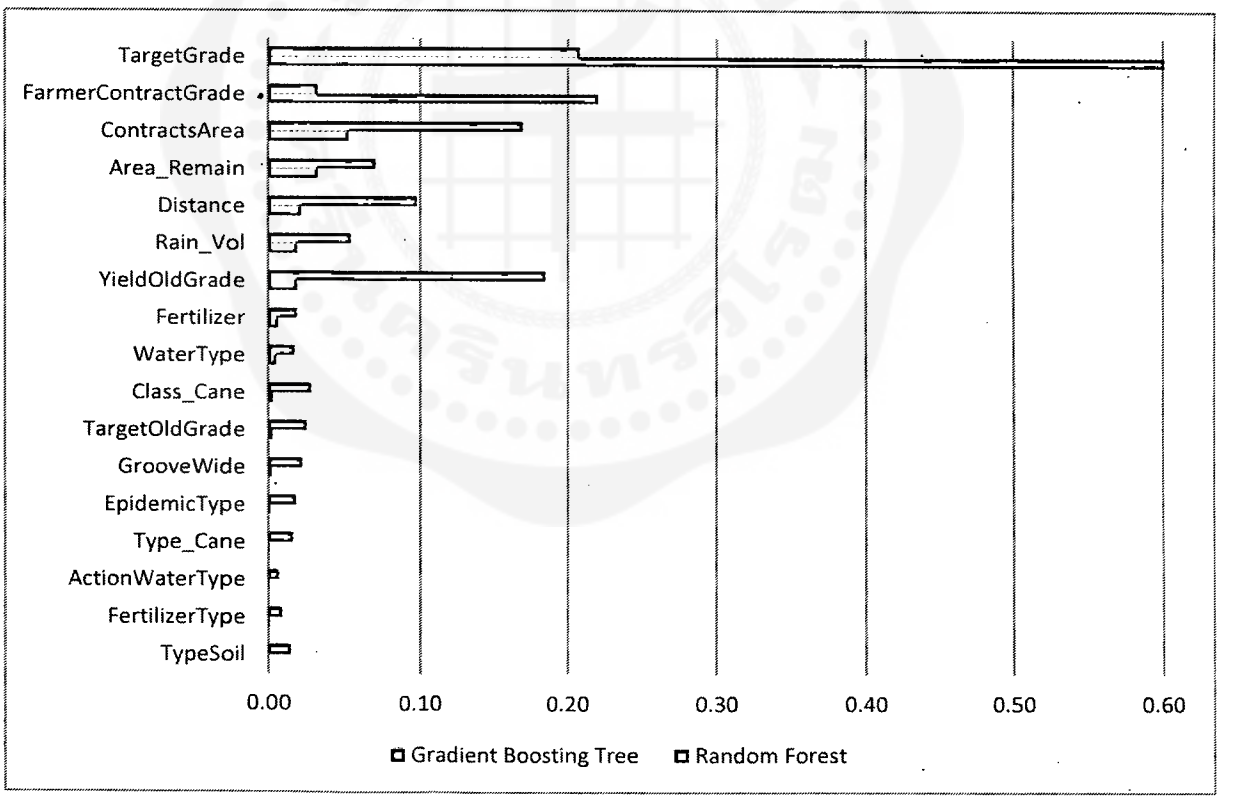
จากตารางที่ 19 ถึง 21 ที่แสดง confusion matrix ของเทคนิคการทำนายแบบต่าง ๆ เราพบว่าแบบจำลองการทำนายที่เราพัฒนาขึ้นทั้ง 3 แบบ ทำนายข้อมูลของแปลงอ้อยที่อยู่ในเกรด 3 ได้ไม่แม่นยำ โดยมีค่า accuracy ประมาณ 45% เท่านั้น แต่เมื่อพิจารณา confusion matrix ของ Baseline-1 และ Baseline-2 เราพบว่าการทำนายแปลงอ้อยที่อยู่ในเกรด 3 ของ Baseline ก็มีค่า Accuracy ต่ำเช่นกัน ส่วนการทำนายแปลงอ้อยที่อยู่ในเกรด 1 และ 2 เราพบว่าแบบจำลองการทำนายที่เราพัฒนาขึ้นจะมีค่า accuracy ถูกต้องเกิน 70% ซึ่งสูงกว่า Baseline ทั้งคู่ ดังนั้นถ้าเราสามารถปรับปรุงแบบจำลองให้สามารถทำนายแปลงอ้อยที่อยู่ในเกรด 3 ให้มีค่า Accuracy สูงมากขึ้นจะทำให้เราได้แบบจำลองที่มีความแม่นยำมากขึ้น

ถ้าเราพิจารณาถึงค่า Feature importance ของแต่ละ feature ในข้อมูลที่มีผลต่อแบบจำลองการทำนายที่เทคนิค Random Forest และ เทคนิค Gradient Boosting Tree ข้อมูลค่า Feature importance ซึ่งได้นำเสนอในหัวข้อ 4.2.3 และในหัวข้อ 4.3.3 สามารถแสดงคู่กันได้ดังตารางที่ 22 และแสดงในรูปของกราฟเปรียบเทียบดังภาพประกอบ 21 โดยในตารางที่ 22 ได้ระบุชื่อ Feature ที่มี

ความสำคัญตามลำดับที่ 1 ถึง 10 (Rank 1, 2, ..., 10) ซึ่งจะเห็นได้ว่า "TargetGrade" เป็น feature ที่มีความสำคัญเป็นอันดับ1 ทั้งแบบจำลองที่ใช้ Random Forest และที่ใช้ Gradient Boosing Tree

แต่อย่างไรก็ตามเนื่องจาก "TargetGrade" จะถูกใช้เป็นผลลัพธ์การทำนายของ Baseline-2 ผู้วิจัยจึงตั้งข้อสังเกตว่า เราสามารถพิจารณาได้ว่าแบบจำลองที่เราพัฒนาขึ้นนี้จะมาช่วยเพิ่มความถูกต้องในการทำนายของฝ่ายสำรวจของโรงงาน โดยใช้ข้อมูลของ feature อื่น ๆ มาประกอบ แล้วทำให้ค่า accuracy ในการทำนายเพิ่มสูงขึ้นประมาณ 6.5%

นอกจากนี้ ข้อสังเกตที่ผู้วิจัยได้พบจากภาพประกอบ 21 และตารางที่ 22 คือ แบบจำลองที่ใช้เทคนิค Random Forest จะมี Feature ที่มีสำคัญสูงอยู่เพียง 2 features ในขณะที่แบบจำลองที่ใช้เทคนิค Gradient Boosting Tree จะมี Feature ที่มีสำคัญสูงพอ ๆ กัน อยู่ถึง 4-5 feature ผู้วิจัยจึงคิดว่าอาจจะเป็นไปได้ที่เราจะสามารถนำผลลัพธ์การทำนายของทั้ง 2 แบบจำลองมาผสมกันกันเพื่อทำนายผลลัพธ์ที่มีค่า Accuracy สูงยิ่งขึ้น



ภาพประกอบ 21 แสดงค่า Feature Importances ของ Random Forest และ Gradiant Boosting Tree ของ Classification

ตาราง 22 แสดง Feature importance ของ Random Forest และ Gradient boosting tree ส่วน Classification

Feature name	Feature importance value		Mean Validation Score (with forward feature selection)
	Random Forest	Gradient Boosting Tree	
TargetGrade	**0.60273 (Rank 1)**	**0.20846 (Rank 1)**	0.66374
FarmerContractGrade	0.22070 (Rank 2)	0.03323 (Rank 7)	0.69577
ContractsArea	0.05302 (Rank 3)	0.17012 (Rank 3)	0.70264
Area_Remain	0.03336 (Rank 4)	0.07100 (Rank 5)	0.70208
Distance	0.02211 (Rank 5)	0.09855 (Rank 4)	0.69649
Rain_Vol	0.019351 (Rank 6)	0.05453 (Rank 6)	0.69864
YieldOldGrade	0.01931 (Rank 7)	0.18546 (Rank 2)	0.70232
Fertilizer	0.007022 (Rank8)	0.01903	0.70367
WaterType	0.005858 (Rank 9)	0.01761	0.70511
Class_Cane	0.003528 (Rank 10)	0.02869 (Rank 8)	0.70503
TargetOldGrade	0.00341	0.02556 (Rank 9)	0.70511
GrooveWide	0.00285	0.02244 (Rank 10)	**0.70591**
EpidemicType	0.00194	0.01846	0.70487
Type_Cane	0.00144	0.0159	0.70495
ActionWaterType	0.00139	0.0071	0.70487
FertilizerType	0.00134	0.00909	0.70511
TypeSoil	0.00065	0.01477	0.70407

6.2 สรุปผลของระบบการทำนายค่าปริมาณผลผลิตอ้อย (Yield value prediction)

งานวิจัยในส่วนนี้ผู้วิจัยได้พัฒนาอัลกอริทึมหรือแบบจำลองการทำนายที่แตกต่างกันจำนวน 2 แบบ สามารถสรุปได้ดังนี้

(1) Random Forest with hyper-parameter tuning

อัลกอริทึมหรือแบบจำลองการทำนายที่ใช้เทคนิค Random Forest Regression และใช้เทคนิค Hyper-parameter tuning แบบ Grid search ที่ทำงานร่วมกับเทคนิค K-Fold Cross Validation เพื่อให้ได้ชุดของพารามิเตอร์ที่นำไปสู่แบบจำลองที่มีค่าความคลาดเคลื่อน (Error) ในการทำนายที่มีค่าน้อยที่สุด รายละเอียดของแบบจำลองการทำนายแบบนี้อธิบายอยู่ในหัวข้อที่ 5.2.3

(2) Gradiant Boosting Tree (XGBoost) with hyper-parameter tuning

อัลกอริทึมหรือแบบจำลองการทำนายที่ใช้เทคนิค Gradient Boosting Tree Regression โดยการใช้เทคนิค Hyper-parameter tuning แบบ Grid search ที่ทำงานร่วมกับเทคนิค K-Fold Cross Validation เพื่อให้ได้ชุดของพารามิเตอร์ที่นำไปสู่แบบจำลองที่มีค่าความคลาดเคลื่อน (Error) ในการทำนายที่มีค่าน้อยที่สุด รายละเอียดของแบบจำลองการทำนายแบบนี้อธิบายอยู่ในหัวข้อที่ 5.3.3

การประเมินผลความแม่นยำในการทำนายค่าปริมาณผลผลิตอ้อยจะใช้ตัวชี้วัด RMSE, MAPE และ R^2 (ดูคำอธิบายในหัวข้อ 2.3) ซึ่งตัวชี้วัดเหล่านี้จะอิงกับค่าความแตกต่างระหว่างค่า Yield จริง ๆ กับค่า Yield ที่ทำนายของแต่ละแปลงอ้อย นอกจากนี้เราได้พัฒนาแบบจำลองการทำนายโดยใช้เทคนิค Linear Regression มาเป็น Baseline อีกตัวในการประเมินผลความแม่นยำ

ผลลัพธ์ค่า RMSE, MAPE และ R^2 ในการทำนายข้อมูลใน Test dataset ของ Baseline-1, Baseline-2, Linear Regreion และแบบจำลองที่เราพัฒนาจากทั้งสองอัลกอริทึม สามารถสรุปได้ดังตารางที่ 23

ตาราง 23 สรุปค่า RMSE, MAPE, R^2 ของ baseline และแบบจำลองการทำนายที่ได้พัฒนาขึ้นในงานวิจัยนี้

Regression Method	RMSE (ตัน/ไร่)	MAPE (%)	R^2
Baseline-1	5.02	49.09	0.33
Baseline-2	2.60	39.68	0.62
Linear Regression	3.01	39.22	0.44
Random Forest with hyper-parameter tuning	*2.18*	*32.24*	*0.70*
Gradiant Boosting Tree (XGBoost) with hyper-parameter tuning	*2.19*	*31.91*	*0.70*

จากผลการทดลองเราพบว่าแบบจำลองการทำนายที่เราพัฒนาขึ้นทั้ง 2 แบบ มีค่า RMSE (2.18 และ 2.19 ตัน/ไร่ ตามลำดับ) และค่า MAPE (ประมาณ 32%) ต่ำกว่าค่าของ Baseline-1 และ Baseline-2 ซึ่งทำให้เราสรุปได้ว่าแบบจำลองการทำนายโดยใช้เทคนิคการเรียนรู้ของเครื่องที่เราพัฒนาขึ้น มีความถูกต้องแม่นยำกว่าการทำนายโดยการประมาณผลผลิตของฝ่ายสำรวจของโรงงาน (หรือ Baseline-2 ซึ่งมีค่า RMSE ประมาณ 2.60 ตัน/ไร่) โดยแบบจำลองการทำนายที่เราพัฒนาขึ้นจะมี RMSE ที่ต่ำกว่าประมาณ 0.42 ตันต่อไร่ ถ้าพิจารณาที่ค่า R^2 เราพบว่า ค่า R^2 ของแบบจำลองการทำนายที่เราพัฒนาขึ้นทั้ง 2 แบบมีค่าสูงกว่า Baseline-1 และ Baseline-2 ซึ่งทำให้เราสรุปได้ว่าแบบจำลองการทำนายของเรามีความแม่นยำมากกว่า

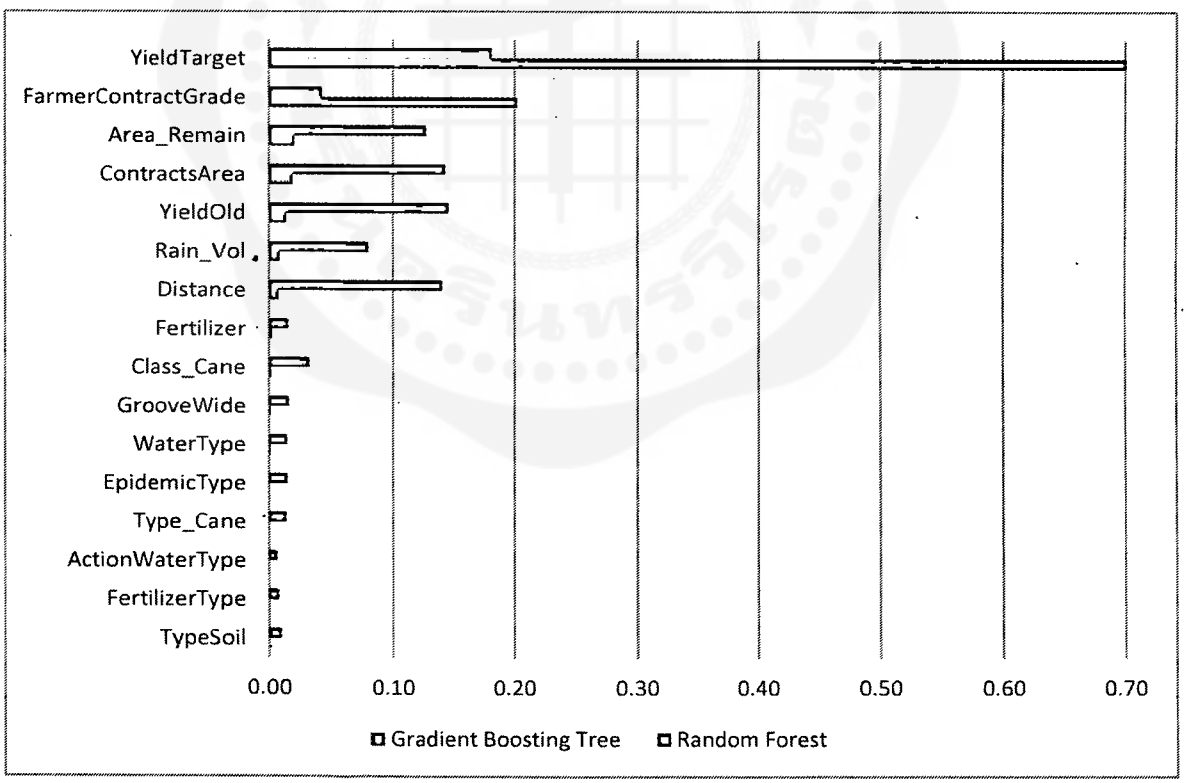
เมื่อเปรียบเทียบกับแบบจำลองที่ใช้เทคนิค Linear Regression ที่เป็นเทคนิคที่ใช้โดยทั่วไปเพื่อทำนายเชิงปริมาณนั้น ผู้วิจัยพบว่าทั้งแบบจำลองการทำนายที่ใช้ Random Forest และ Gradiant Boosting Tree จะให้ค่า RMSE ต่ำกว่าและได้ค่า R^2 ที่สูงกว่าเช่นกัน จึงทำให้สรุปได้ว่าแบบจำลองที่พัฒนาในงานวิจัยนี้สามารถทำนายได้ถูกต้องมากกว่า ดังนั้นการพัฒนาแบบจำลองนี้สามารถช่วยเพิ่มการทำนายผลผลิตของฝ่ายสำรวจของโรงงานได้

ถ้าเราพิจารณาถึงค่า Feature importance ของแต่ละ feature ในข้อมูลที่มีผลต่อแบบจำลองการทำนายที่เทคนิค Random Forest และ เทคนิค Gradient Boosting Tree ข้อมูลค่า Feature importance ซึ่งได้นำเสนอในหัวข้อ 5.2.3 และในหัวข้อ 5.3.3 สามารถแสดงคู่กันได้ดังตารางที่ 6-9 และแสดงในรูปของกราฟเปรียบเทียบดังภาพประกอบ 6-2 โดยในตารางที่ 6-9 ได้ระบุชื่อ Feature ที่มี

ความสำคัญตามลำดับที่ 1 ถึง 10 (Rank 1, 2, ..., 10) ซึ่งจะเห็นได้ว่า "YieldTarget" เป็น feature ที่มีความสำคัญเป็นอันดับ 1 ต่อทั้งแบบจำลองที่ใช้ Random Forest และที่ใช้ Gradient Boosing Tree

เช่นเดียวกับกรณีของการทำนายเกรดของปริมาณผลผลิตที่ได้อธิบายในหัวข้อที่ 6.2 ผู้วิจัยพบว่าเนื่องจาก "YieldTarget" จะถูกใช้เป็นผลลัพธ์การทำนายของ Baseline-2 ผู้วิจัยจึงตั้งข้อสังเกตว่า เราสามารถพิจารณาได้ว่าแบบจำลองที่เราพัฒนาขึ้นนี้จะมาช่วยเพิ่มความถูกต้องในการทำนายของฝ่ายสำรวจของโรงงาน โดยใช้ข้อมูลของ feature อื่น ๆ มาประกอบ แล้วทำให้ค่า RMSE ในการทำนายมีค่าลดลงจาก 2.60ตัน/ไร่ มาอยู่ที่ค่า 2.18 และ 2.19 ตัน/ไร่

นอกจากนี้ ข้อสังเกตที่ผู้วิจัยได้พบจากภาพประกอบ 22 และ ตารางที่ 24 คือ แบบจำลองที่ใช้เทคนิค Random Forest จะมี Feature ที่มีสำคัญสูงอยู่เพียง 2 features ในขณะที่แบบจำลองที่ใช้เทคนิค Gradient Boosting Tree จะมี Feature ที่มีสำคัญสูงพอ ๆ กัน อยู่ถึง 4-5 feature ผู้วิจัยจึงคิดว่าอาจจะเป็นไปได้ที่เราจะสามารถนำผลลัพธ์การทำนายของทั้ง 2 แบบจำลองมาผสมกันกันเพื่อทำนายผลลัพธ์ที่ทำให้ได้ค่าความผิดพลาด (RMSE และ MAPE) ลดลง



ภาพประกอบ 22 แสดงค่า Feature Importances ของ Random Forest และ Gradiant Boosting Tree ของ Regression

ตาราง 24 แสดง Feature importance ของ Random Forest และ Gradient boosting tree ของ Regression

Feature name	Feature importance value	
	Random Forest	Gradient Boosting Tree
YieldTarget	**0.71226 (Rank 1)**	**0.18236 (Rank 1)**
FarmerContractGrade	0.20320 (Rank 2)	0.04331 (Rank 7)
Area_Remain	0.02083 (Rank 3)	0.12833 (Rank 5)
ContractsArea	0.01949 (Rank 4)	0.14382 (Rank 3)
YieldOld	0.01430 (Rank 5)	0.14660 (Rank 2)
Rain_Vol	0.00906 (Rank 6)	0.08025 (Rank 6)
Distance	0.00819 (Rank 7)	0.14144 (Rank 4)
Fertilizer	0.00322 (Rank 8)	0.015892 (Rank 10)
Class_Cane	0.00268 (Rank 9)	0.032976 (Rank 8)
GrooveWide	0.00227 (Rank 10)	0.016289 (Rank 9)
WaterType	0.00193	0.01471
EpidemicType	0.00086	0.01470
Type_Cane	0.00059	0.01391
ActionWaterType	0.00044	0.00675
FertilizerType	0.00042	0.00834
TypeSoil	0.00029	0.01033

6.3 ปัญหาและข้อเสนอแนะ

งานวิจัยนี้มีจุดประสงค์คือพัฒนาแบบจำลองการทำนายปริมาณอ้อยในรูปแบบการจัดเกรดและปริมาณอ้อยในแต่ละแปลง โดยใช้ข้อมูลที่มาจากการสำรวจพื้นที่การปลูกอ้อยและวิธีการปลูกอ้อยของชาวไร่ที่ทำสัญญาการส่งอ้อยไว้กับโรงงานน้ำตาล ข้อมูลที่นำมาใช้นั้นมีข้อมูลเพียง 12,521 รายการเท่านั้นจากการเก็บข้อมูลจากปีการผลิตสองปี ซึ่งอาจเป็นไปได้ว่า ขนาดหรือจำนวนข้อมูลที่นำมาใช้ในการพัฒนาแบบจำลองยังมีไม่มากพอที่จะสร้างแบบจำลองที่ดีในการทำนายได้

นอกจากนี้ จากการทบทวนงานวิจัยที่เกี่ยวข้อง ผู้วิจัยพบว่าในหลาย ๆ งานวิจัยที่เกี่ยวกับการทำนายผลผลิตพืชนั้น จะมีข้อมูลอุณหภูมิเป็น feature หลักที่มีผลต่อการทำนายและข้อมูลเกี่ยวกับลักษณะของดินที่ปลูก เช่น ค่า pH ของดินและค่าความชื้นของดิน เป็นต้น จะมีผลต่อการทำนายผลผลิตพืชต่าง ๆ เช่นกัน แต่ในงานวิจัยนี้มีข้อจำกัดเรื่องการใช้ข้อมูลที่มีในระบบของโรงงานเท่านั้นในการทำนาย

ดังนั้นด้วยสาเหตุที่ไม่มีข้อมูล feature ที่จำเป็นต่อการทำนาย จึงส่งผลทำให้ค่าจากการทำนายที่ได้ผลยังไม่ถูกต้องแม่นยำ อีกทั้ง feature ที่มีส่วนใหญ่ในข้อมูลเป็น feature แบบ category ซึ่งไม่เหมาะนำมาใช้ทำนายข้อมูลเชิงปริมาณคือปริมาณผลผลิตต่อไร่

ผู้วิจัยจะนำปัญหานี้ไปเสนอแนะให้กับทางโรงงานน้ำตาลทำการเก็บข้อมูลอื่น ๆ เพิ่มเติมเพื่อนำมาใช้และปรับปรุงงานต่อไป

บรรณานุกรม

บรรณานุกรม

1. Breiman, L. (2001). Random Forest. *Machine Learning*. 45(1):5-32.
2. Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*. 29(5):1189-1232.
3. Pedregosa; *et al.* (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 12: 2825-2830.
4. Chen, Tianqi; & Guestrin, Carlos. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16*. 785-794.
5. สำนักงานเศรษฐกิจการเกษตร. (2559). *คู่มือเศรษฐกิจอาสา(ศกอ.).[ออนไลน์].* สืบค้นเมื่อ 20 กรกฎาคม 2560, จาก http://www.oae.go.th/
6. EVERINGHAM, Yvette; *et al.* (2016). Accurate prediction of sugarcane yield using a random forest algorithm. *Agronomy for Sustainable Development*. 36(27):1-9.
7. JEONG, Jig Han; et al. (2016). Random Forests for Global and Regional Crop Yield Predictions. *Plos One*. 11(6):1-15.
8. GANDHI, Niketa; et al (2016). Rice crop yield prediction in India using support vector machines. *2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. 1-5.
9. Ruß, Georg. (2009). Data Mining of Agricultural Yield Data: A Comparison of Regression Models. *Advances in Data Mining. Applications and Theoretical Aspects Lecture Notes in Computer Science*. 24-37.
10. Gonzalez-Sanchez, Alberto; Frausto-Solis, Juan; & Ojeda-Bustamante, Waldo. (2014). Predictive ability of machine learning methods for massive crop yield prediction. *Spanish Journal of Agricultural Research*. 12(2):313-328.
11. Gonzalez-Sanchez, Alberto; Frausto-Solis, Juan; & Ojeda-Bustamante, Waldo. (2014). Attribute Selection Impact on Linear and Nonlinear Regression Models for Crop Yield Prediction. *The Scientific World Journal, 2014*. 1-10.

12. DE OLIVEIRA; et al (2017). From spreadsheets to sugar content modeling: A data mining approach. *Computers and Electronics in Agriculture*. 132:14-20.

13. Bocca, Felipe F.; & Luiz Henrique Antunes Rodrigues (2016). The effect of tuning, feature engineering, and feature selection in data mining applied to rainfed sugarcane yield modelling. *Computers and Electronics in Agriculture*. 128:67-76.

14. Thuankaewsing, Surached; et al. (2015). Harvest scheduling algorithm to equalize supplier benefits: A case study from the Thai sugar cane industry. *Computers and Electronics in Agriculture*. 110:42-55.

15. ดิลก ภิญโญศรี. (2552). การสร้างแบบจำลองนิวโร-ฟัซซี่ เพื่อการประมาณการผลผลิตอ้อย. *วิทยานิพนธ์ วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมอุตสาหการ)*. ขอนแก่น: สำนักวิทยาบริการ มหาวิทยาลัยขอนแก่น.

16. Robnik-Šikonja, Marko; & Igor Kononenko. (1997). An adaptation of Relief for attribute estimation in regression. In *Machine Learning: Proceedings of the Fourteenth International Conference (ICML'97)*. 296-304.

17. Ruß, Georg; & Rudolf Kruse (2009). Feature Selection for Wheat Yield Prediction. *Research and Development in Intelligent Systems XXVI*. 465-478.

18. Gonzalez-Sanchez, Alberto; Frausto-Solis, Juan; & Ojeda-Bustamante, Waldo. (2014). Attribute Selection Impact on Linear and Nonlinear Regression Models for Crop Yield Prediction. *The Scientific World Journal, 2014*. 1-10.

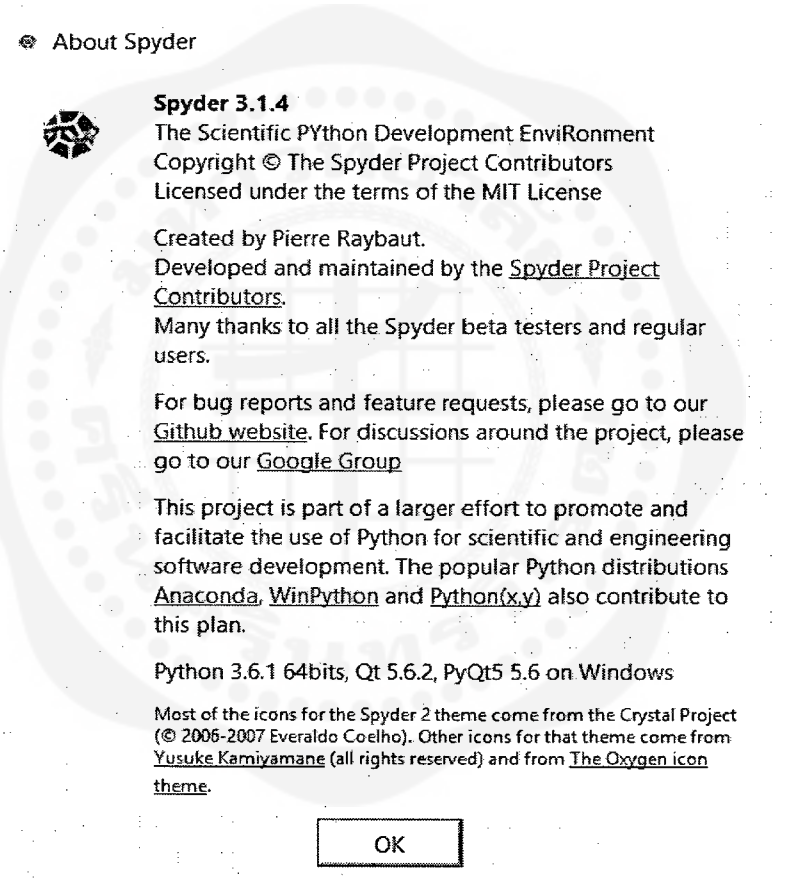
19. Charoen-Ung, Phusanisa; & Mittrapiyanuruk, Pradit. (2018). Sugarcane Yield Grade Prediction using Random Forest and Gradient Boosting Tree Techniques. *The 15th International Joint Conference on Computer Science and Software Engineering (JCSSE2018)*.

20. Charoen-Ung, Phusanisa; & Mittrapiyanuruk, Pradit. (2018). Sugarcane Yield Grade Prediction Using Random Forest with Forward Feature Selection and Hyper-parameter Tuning. *International Conference on Computing and Information Technology*. 33-42.

ภาคผนวก

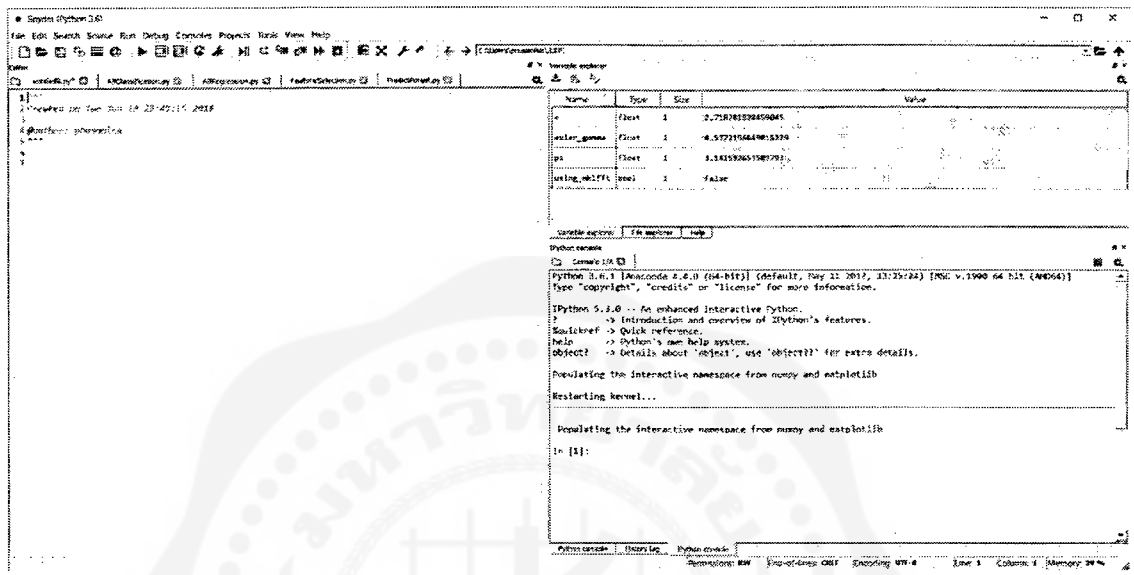
ภาคผนวก

งานวิจัยนี้พัฒนาแบบจำลองการทำนายผลผลิตอ้อยโดยใช้เทคนิคการเรียนรู้ของเครื่องโดยพัฒนาโปรแกรมโดยใช้ IDE Editor ของ Anaconda3 ชื่อ Spyder เวอร์ชั่น 3.1.4 มีรายละเอียดดังภาพประกอบ 1 นี้



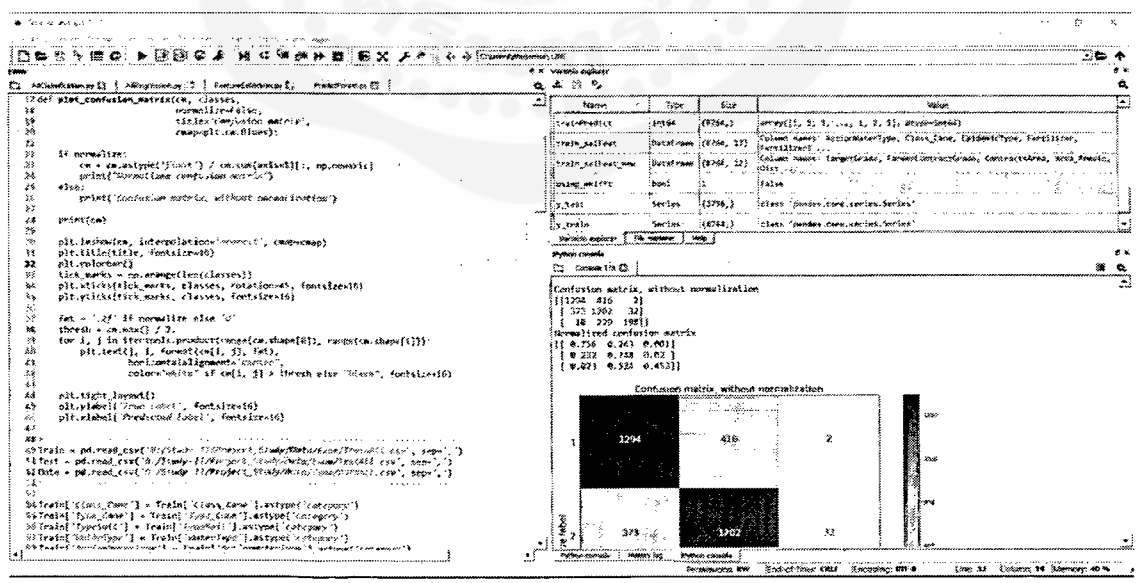
ภาพประกอบ 1 รายละเอียด Spyder 3.1.4

เมื่อเข้าโปรแกรมจะมีหน้าจอดังภาพประกอบ 2 ซึ่งประกอบด้วยสามส่วนหลัก คือ Editor, Explorer และ Console



ภาพประกอบ 2 หน้าจอหลักของโปรแกรม Spyder

ตัวอย่างเมื่อทำการเขียนโปรแกรมภาษา Python และรันดูผลลัพธ์จะได้ดังภาพประกอบ 3



ภาพประกอบ 3 ตัวอย่างการแสดงผลของโปรแกรม Spyder

ประวัติย่อผู้วิจัย

ประวัติย่อผู้วิจัย

ชื่อ ชื่อสกุล	ภูษณิศา เจริญอึ้ง
วันเดือนปีเกิด	25 สิงหาคม 2521
สถานที่เกิด	จังหวัดอุตรดิตถ์
สถานที่อยู่ปัจจุบัน	126 ตำบลศรีพนมมาศ อำเภอลับแล จังหวัดอุตรดิตถ์
ตำแหน่งหน้าที่การงานปัจจุบัน	นักวิเคราะห์ข้อมูล
สถานที่ทำงานปัจจุบัน	กลุ่มน้ำตาลไทยรุ่งเรือง
ประวัติการศึกษา	
พ.ศ. 2541	วิทยาศาสตรบัณฑิต สาขาวิชาคณิตศาสตร์
	จาก มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี
พ.ศ. 2554	บริหารธุรกิจมหาบัณฑิต สาขาการจัดการ
	จาก มหาวิทยาลัยรามคำแหง
พ.ศ. 2560	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
	จาก มหาวิทยาลัยศรีนครินทรวิโรฒ