การพยากรณ์สินค้าคงค้างด้วยการเรียนรู้ของเครื่องสำหรับข้อมูลไม่สมดุล

น.ส.ธิรดา ธาดาจิรสกุล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมอุตสาหการ ภาควิชาวิศวกรรมอุตสาหการ
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2565

Backorder Prediction Using Machine Learning for Imbalanced Data Classification

Miss Thirada Thadajirasakul

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Industrial Engineering
Department of Industrial Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2022

หัวข้อวิทยานิพนธ์	การพยากรณ์สินค้าคงค้างด้วยการเรียนรู้ของเครื่องสำหรับข้อมูลไม่สมดุล
โดย	น.ส.ธิรดา ธาดาจิรสกุล
สาขาวิชา	วิศวกรรมอุตสาหการ
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	อาจารย์ ดร.ปุณณมี สัจจกมล

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.ปวีณา เชาวลิตวงศ์)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(อาจารย์ ดร.ปุณณมี สัจจกมล)

..... กรรมการ
(รองศาสตราจารย์ ดร.นระเกณฑ์ พุ่มชูศรี)

..... กรรมการภายนอกมหาวิทยาลัย
(รองศาสตราจารย์ ดร.จันทร์ศิริ สิงห์เถื่อน)

ธิรดา ธาดาจิรสกุล : การพยากรณ์สินค้าคงค้างด้วยการเรียนรู้ของเครื่องสำหรับข้อมูลไม่สมดุล. (Backorder Prediction Using Machine Learning for Imbalanced Data Classification) อ.ที่ปรึกษาหลัก : อ. ดร.ปุณณมี สัจจกมล

การใช้การเรียนรู้ของเครื่องในการพยากรณ์สินค้าคงค้างกับข้อมูลที่มีรายการสินค้าเป็นจำนวนมากจึงเป็นเรื่องที่จำเป็น ซึ่งในความเป็นจริงข้อมูลที่เจอมักมีความไม่สมดุลทำให้ประสิทธิภาพในการพยากรณ์ด้วยการเรียนรู้ของเครื่องลดลง การพยากรณ์สินค้าคงค้างที่ไม่ถูกต้องนั้นส่งผลต่อความไว้วางใจของผู้ซื้อและทำให้เสียค่าใช้จ่ายถึงร้อยละ 10 ของรายได้ งานวิจัยฉบับนี้จึงได้ศึกษาการปรับสมดุลข้อมูลด้วยวิธี Threshold Moving และการปรับระดับข้อมูลด้วยวิธีสุ่มเพื่อสร้างตัวแบบที่มีประสิทธิภาพและมีความสามารถในการพยากรณ์ข้อมูลกลุ่มน้อยสูง โดยวิธีการปรับระดับข้อมูลมี 4 วิธีได้แก่ การปรับลดข้อมูลด้วยวิธี NearMiss-3, การปรับลดข้อมูลด้วยวิธี OSS, การปรับเพิ่มข้อมูลด้วยวิธี SMOTE และการปรับลดผสมกับเพิ่มข้อมูลด้วยวิธี OSS ผสม SMOTE โดยอัลกอริทึมที่ใช้ได้แก่ LOGIST, FOREST และ XGBoost นอกจากนี้มีการใช้การตรวจสอบแบบไขว้แบบ 5 กลุ่มกับตัวแบบเพื่อป้องกันการเกิด Overfitting ในวิจัยฉบับนี้มีการวัดประสิทธิภาพของตัวแบบด้วย AUROC, F1 score และ G-Mean ซึ่งผลที่ได้จากงานวิจัยฉบับนี้คือการจัดการข้อมูลด้วย Threshold Moving ด้วยการวัดประสิทธิภาพ G-Mean นั้นให้น้ำหนักกับข้อมูลกลุ่มน้อยมากกว่า F1 score และให้ผลลัพธ์ดีกว่า AUROC โดยวิธีการที่ให้ผลลัพธ์ดีที่สุดคือการจัดการข้อมูลด้วย Threshold Moving ด้วยการวัดประสิทธิภาพ G-Mean สำหรับอัลกอริทึม Forest ซึ่งได้ค่าประมาณ 0.8737

สาขาวิชา วิศวกรรมอุตสาหการ ลายมือชื่อนิสิต .....

ปีการศึกษา 2565 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6370131521 : MAJOR INDUSTRIAL ENGINEERING

KEYWORD: Threshold Moving, Imbalanced data, Backorder, Sampling methods, Ensemble learning

Thirada Thadajirasakul : Backorder Prediction Using Machine Learning for Imbalanced Data Classification. Advisor: PUNNAMEE SACHAKAMOL, Ph.D.

It is essential to use machine learning for predicting products' backorder to deal with massive data of SKU. Naturally, real world data is usually imbalanced data which is affect to the efficiency of machine learning. Mistaken predicting products' backorder negatively affects customer's service level and decrease 10 percent of their revenue. This research has studied adjusting data by Threshold Moving and sampling methods for creating efficient model and high forecast proficiency in minority class model. There are 4 methods for adjusting data including NearMiss-3 for undersampling dataset, One-Sided Selection (OSS) for undersampling dataset, SMOTE for oversampling dataset, and combining OSS and SMOTE dataset. LOGIST, FOREST and XGBoost are used as algorithms and Stratified 5-Fold Cross-Validation is used to prevent overfitting. In this research, AUROC, F1 score and G-Mean are used as the efficiency measurements. The result obtained from this research study is Threshold Moving with the G-Mean metric gives more weight to the minority data group compared to F1 score and provides better results than AUROC. The most effective method is using Threshold Moving with G-Mean metric for the Forest algorithm, achieving an approximate value of 0.8737.

Field of Study: Industrial Engineering Student's Signature .....

Academic Year: 2022 Advisor's Signature .....

กิตติกรรมประกาศ

งานวิจัยฉบับนี้สำเร็จได้ดีเนื่องจากได้รับความกรุณาอย่างสูงจาก อ. ดร. ปุณณมี สัจจกมล อาจารย์ที่ปรึกษาวิทยานิพนธ์ที่ให้คำแนะนำและแนวทางในการทำวิจัยที่ถูกต้องตามระเบียบ รวมทั้งให้กำลังใจในการทำวิทยานิพนธ์และให้ความช่วยเหลือในการแก้ปัญหาอย่างดีตลอดมา

ขอขอบพระคุณ รศ. ดร. ปวีณา เชาวลิตวงศ์ ประธานกรรมการการสอบวิทยานิพนธ์ รศ. ดร. นระเกณฑ์ พุ่มชูศรี กรรมการสอบวิทยานิพนธ์และ รศ. ดร. จันทร์ศิริ สิงห์เถื่อน กรรมการสอบวิทยานิพนธ์ จากมหาวิทยาลัยเกษตรศาสตร์ กรุณาสละเวลาตรวจสอบและแก้ไขข้อบกพร่อง พร้อมทั้งคำแนะนำในด้าน ต่าง ๆ จนวิทยานิพนธ์ฉบับนี้มีความถูกต้องและชัดเจน

ขอขอบพระคุณญาดา ศรีวิบูลย์และคุณเตชินท์ อานนท์วัฒนาที่ให้คำปรึกษาและเป็นกำลังใจให้ตลอดการทำวิทยานิพนธ์

สุดท้ายนี้ขอขอบพระคุณครอบครัวและเพื่อนๆ ที่ช่วยสนับสนุนและผลักดันผู้ทำวิจัยเสมอมา

ธิรดา ธาดาจิรสกุล

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูป

หน้า

บทที่ 1
บทนำ

ในบทนี้จะกล่าวถึงรายละเอียดโดยรวมของวิทยานิพนธ์ ความเป็นมาของการทำวิทยานิพนธ์ และขอบเขตที่ผู้วิจัยได้ทำ โดยมีรายละเอียดดังหัวข้อต่อไปนี้

1. ความเป็นมาและความสำคัญของปัญหา
2. วัตถุประสงค์
3. ขอบเขตของวิทยานิพนธ์
4. ประโยชน์ที่คาดว่าจะได้รับ

1.1 ความเป็นมาและความสำคัญของปัญหา

การจัดการห่วงโซ่อุปทานคือการวางแผนสินค้าคงคลัง รวมถึงการตัดสินใจซื้อสินค้าในปริมาณและช่วงเวลาที่เหมาะสมโดยพิจารณาจากปัจจัยต่างๆ แต่เนื่องจากความต้องการของผู้ซื้อมาแบบไม่แน่นอนส่งผลให้เกิดความยุ่งยากในการคาดการณ์ความต้องการซึ่งทำให้ระบบการจัดการห่วงโซ่อุปทานแบบเดิมมีประสิทธิภาพน้อยลงในหลาย ๆ ด้าน เช่น การคาดการณ์อุปสงค์ที่ไม่ถูกต้องหรือการจัดประเภทผลิตภัณฑ์ที่มีความเสี่ยงเป็นสินค้าคงค้างผิด การพยากรณ์ความเสี่ยงในการเกิดสินค้าคงค้างจึงเป็นเรื่องที่ท้าทาย (de Santis et al., 2017) เมื่อผู้ซื้อปลายทางต้องการสินค้า แต่ไม่มีสินค้าในคลัง หรือสินค้าอยู่ในการผลิต ทำให้ไม่สามารถตอบสนองความต้องการ และเกิดการรอสินค้าหรือที่เรียกว่าการเกิดสินค้าคงค้าง (Backorder) (Hajek & Abedin, 2020; Malviya et al., 2021) การพยากรณ์สินค้าคงค้างที่ไม่ถูกต้องส่งผลให้การจัดการสินค้าคงคลังแย่ลง ทำให้ผู้ขายต้องเสียค่าใช้จ่ายมากถึงร้อยละ 10 ของรายได้ การเกิดสินค้าคงค้างบ่อยครั้ง สะท้อนถึงการผลิตที่ไม่เพียงพอ อาจเปิดโอกาสให้ผู้ซื้อหาสินค้าทดแทนจากที่อื่นและลดความไว้วางใจที่มีต่อบริษัท (Hajek & Abedin, 2020) ส่งผลให้บางบริษัทในปัจจุบันมีการใช้การใช้การเรียนรู้ของเครื่อง (Machine Learning) เข้ามาพยากรณ์เสี่ยงในการเกิดสินค้าคงค้าง (Islam & Amin, 2020)

การลดลงของอุปสรรคในการดำเนินการที่เกี่ยวข้องกับการเรียนรู้ของเครื่องไม่ว่าจะเป็นด้านราคา การประมวลผล หรือการเข้าถึงแหล่งข้อมูล ทำให้การเรียนรู้ของเครื่องถูกนำไปใช้กับงานที่หลากหลายตั้งแต่อุตสาหกรรมด้านการผลิต การเกษตร การบิน การบริการรวมถึงการพยากรณ์ความเสี่ยงในการเกิดสินค้าคงค้างที่ผู้จัดทำต้องการศึกษา (de Santis et al., 2017; Malviya et al., 2021) ซึ่งจากงานศึกษาก่อนหน้าพบว่าข้อมูลมีลักษณะไม่สมดุลกันระหว่างรายการสินค้าที่เป็นสินค้าคงค้าง (ข้อมูลกลุ่มน้อย) กับรายการสินค้าที่ไม่เป็นสินค้าคงค้าง(ข้อมูลกลุ่มมาก) หรือที่เรียกว่าข้อมูล

ไม่สมดุล (Imbalanced Data) ส่งผลให้ประสิทธิภาพของการเรียนรู้ด้วยเครื่องลดลง เนื่องจากมองข้ามข้อมูลกลุ่มน้อย หรือมองว่าเป็นข้อมูลที่ผิดพลาด จึงพยากรณ์ข้อมูลกลุ่มน้อยว่าเป็นข้อมูลกลุ่มมาก กล่าวคือไม่สามารถพยากรณ์ข้อมูลกลุ่มน้อยได้ เพื่อแก้ปัญหาดังกล่าวงานศึกษาก่อนหน้าจึงได้ใช้วิธีการปรับระดับข้อมูลแบบต่างๆ ไม่ว่าจะเป็นการเพิ่มจำนวนข้อมูลกลุ่มน้อยให้มีขนาดเท่ากับข้อมูลกลุ่มมาก หรือ การลดจำนวนข้อมูลกลุ่มมากให้มีขนาดเท่ากับข้อมูลกลุ่มน้อยเช่นวิธี SMOTE ,RUS และ CBUS ก่อนนำข้อมูลมาเรียนรู้ด้วยเครื่องเพื่อพยากรณ์ความเสี่ยงการเกิดสินค้าคงค้าง (de Santis et al., 2017; Hajek & Abedin, 2020; Islam & Amin, 2020; Malviya et al., 2021)

จากข้อมูลของบริษัทแห่งหนึ่งที่ผู้จัดทำได้ศึกษาซึ่งเป็นข้อมูลขนาดใหญ่ โดยแต่ละรายการสินค้าประกอบด้วยข้อมูลจำนวนสินค้าคงคลัง เวลาในการขนส่ง สินค้าที่อยู่ระหว่างขนส่ง ยอดขายคาดการณ์ ปริมาณการขายย้อนหลัง จำนวนสินค้าค้างส่ง ปัจจัยเสี่ยงต่างๆ รวมถึงความเสี่ยงในการเกิดสินค้าคงค้างซึ่งมีลักษณะข้อมูลไม่สมดุลในอัตราส่วน 1 ต่อ 137 หรือหมายถึงมีข้อมูลกลุ่มน้อยน้อยกว่าร้อยละ 1 ของข้อมูลทั้งหมด โดยผู้จัดทำได้นำข้อมูลเหล่านี้มาสร้างตัวแบบในการพยากรณ์ความเสี่ยงในการเกิดสินค้าคงค้างด้วยการเรียนรู้ของเครื่อง เพื่อลดการตัดสินใจที่ผิดพลาด และทราบความเสี่ยงล่วงหน้า โดยตัวแบบที่มีความแม่นยำจะช่วยให้ผู้จัดการสินค้าคงคลังจัดการสินค้าได้อย่างเหมาะสม อาจลดระยะเวลารอสินค้าและลดค่าใช้จ่ายในการจัดเก็บสินค้า แต่เนื่องจากปัญหาข้อมูลที่ไม่สมดุลที่ส่งผลเสียต่อการเรียนรู้ของเครื่อง ผู้วิจัยจึงได้จัดการกับปัญหาโดยศึกษาการแก้ปัญหาข้อมูลไม่สมดุลด้วยวิธีการปรับระดับข้อมูลได้แก่ SMOTE, NearMiss-3, OSS และ OSS ผสมกับ SMOTE ซึ่งเป็นวิธีการที่ได้รับความนิยมในการแก้ปัญหาข้อมูลไม่สมดุล แต่เนื่องจากข้อมูลที่มีความไม่สมดุลมากอาจส่งผลให้มีการสร้างข้อมูลรบกวนเมื่อใช้วิธีการปรับเพิ่มระดับข้อมูลอย่างวิธี SMOTE ขณะเดียวกันอาจมีการลบข้อมูลที่สำคัญเมื่อใช้วิธีการปรับระดับลดข้อมูลอย่างวิธี NearMiss-3 และ OSS ด้วยเหตุนี้ผู้วิจัยจึงได้ศึกษาวิธี Threshold-moving ร่วมด้วย ซึ่งเป็นอีกหนึ่งวิธีที่ช่วยแก้ปัญหาด้านข้อมูลไม่สมดุลด้วยแนวคิดการหาจุดแบ่งข้อมูลที่เหมาะสมหลังจากฝึกข้อมูลและได้ตัวแบบที่มีประสิทธิภาพแล้ว ทำให้ไม่ต้องสร้างหรือสูญเสียข้อมูลที่สำคัญและยังใช้เวลาน้อยกว่าการปรับระดับเพิ่มข้อมูล ซึ่งทั้งวิธีการผสมการปรับระดับข้อมูล และ วิธี Threshold-moving นั้นยังไม่พบในงานศึกษาด้านการพยากรณ์ความเสี่ยงการเกิดสินค้าคงค้าง โดยอัลกอริทึมที่ใช้สร้างตัวแบบในวิทยานิพนธ์นี้ล้วนเป็นอัลกอริทึมที่เหมาะสมกับการจำแนกกลุ่มที่มีความไม่สมดุลซึ่งได้แก่ Random forest ,eXtreme Gradient Boosting และ Linear logistic regression นอกจากนี้ข้อมูลที่ไม่สมดุลยังส่งผลต่อตัวชี้วัดบางตัวเช่น accuracy (ความแม่นยำ) ที่ค่าความแม่นยำขึ้นอยู่กับข้อมูลกลุ่มมาก เนื่องจากมองว่าข้อมูลกลุ่มน้อยเป็นข้อมูลที่ผิดพลาดแม้ว่าจะพยากรณ์ข้อมูลกลุ่มน้อยผิดก็ยังคงได้ค่าความแม่นยำที่สูงเนื่องจากพยากรณ์ข้อมูลกลุ่มมากได้ ดังนั้นผู้วิจัยจึงได้ใช้ตัววัดประสิทธิภาพตัวแบบที่เหมาะสมกับข้อมูลที่ไม่สมดุลด้วย โดยสำหรับวิธีปรับระดับข้อมูลจะใช้ AUROC ส่วนวิธี

Threshold-moving จะวัดประสิทธิภาพของการแบ่งเกณฑ์ด้วย F1 score และ G-mean และยังใช้ F1 score กับวิธีปรับระดับข้อมูลเพื่อเปรียบเทียบผลลัพธ์ที่ได้จากการจัดการข้อมูลไม่สมดุลทั้งสองวิธี และใช้การเปรียบเทียบหลังการทดสอบรวมด้วยวิธี Games-Howell เพื่อดูความแตกต่างระหว่างวิธีการย่อยเพื่อสรุปผลและเลือกตัวแบบที่เหมาะสมที่สุดกับชุดข้อมูล

1.2 วัตถุประสงค์

1. เพื่อสร้างตัวแบบพยากรณ์การเกิดสินค้าคงค้างด้วยการเรียนรู้ของเครื่องที่เหมาะสมกับข้อมูลไม่สมดุล

1.3 ขอบเขตของวิทยานิพนธ์

1. ชุดข้อมูลที่ใช้ศึกษาคือข้อมูลของบริษัทแห่งหนึ่งซึ่งสามารถเข้าถึงได้จาก ชุดข้อมูลของ Kaggle ในหัวข้อ Backorder Dataset Predict Backorder Product (ZINJAD, 2021)

2. ศึกษาวิธีการจัดการกับปัญหาข้อมูลไม่สมดุลด้วย SMOTE, NearMiss-3, OSS, OSS ผสมกับ SMOTE และ Threshold-moving

3. ศึกษาการสร้างตัวแบบด้วยอัลกอริทึม Random Forest, eXtreme Gradient Boosting และ Linear logistic regression

4. วัดประสิทธิภาพของตัวแบบด้วย AUROC และเลือกเกณฑ์ในการแบ่งข้อมูลที่เหมาะสมด้วย F1 score และ G-Mean สำหรับวิธี Threshold-moving

5. ศึกษาความแตกต่างระหว่างวิธีการด้วยการเปรียบเทียบหลังการทดสอบรวมด้วยการทดสอบของ Games-Howell

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถประยุกต์ตัวแบบพยากรณ์กับชุดข้อมูลอื่นที่ไม่สมดุล

2. ตัวแบบพยากรณ์สามารถช่วยให้ผู้จัดการสินค้าคงคลังจัดการสินค้าได้อย่างเหมาะสม ลดระยะเวลารอสินค้าและลดค่าใช้จ่ายในการจัดเก็บสินค้า รวมถึงฝ่ายผลิตสามารถควบคุมการผลิตได้ดียิ่งขึ้น

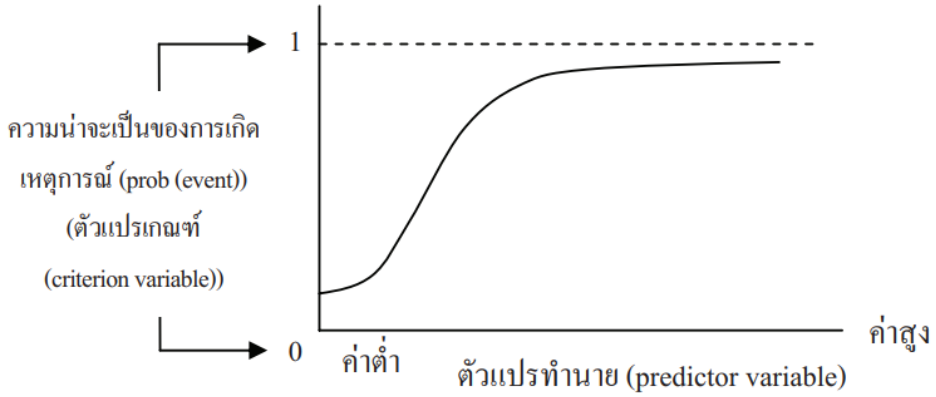
บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

วิทยานิพนธ์เป็นการศึกษาการพยากรณ์สินค้าคงค้างด้วยการเรียนรู้ของเครื่องสำหรับข้อมูลไม่สมดุล เพื่อเข้าใจการจัดการข้อมูลไม่สมดุลและการสร้างตัวแบบเพื่อพยากรณ์ ในบทนี้จะกล่าวถึงทฤษฎีที่เกี่ยวข้องกับวิทยานิพนธ์ ประกอบด้วยหัวข้อดังต่อไปนี้

1. อัลกอริทึม
2. การวัดประสิทธิภาพตัวแบบ
3. การปรับเพิ่ม / ลดระดับข้อมูล
4. Threshold-moving
5. การปรับ boxplot
6. การตรวจสอบแบบไขว้
7. Hyperparameter Tuning
8. การวิเคราะห์ความแปรปรวนแบบ 2 ตัวประกอบ
9. การเปรียบเทียบหลังการทดสอบรวม
10. การทบทวนวรรณกรรมที่เกี่ยวข้อง

2.1 อัลกอริทึม

2.1.1 Logistic Regression (LOGIST) หรือการถดถอยโลจีสติกส์ เป็นอัลกอริทึมที่ง่ายและนิยมใช้กับจำแนกผลลัพธ์ที่มีสองค่า นอกจากนี้ยังไวกับความไม่สมดุลของข้อมูลอีกด้วย ซึ่งหลักการของอัลกอริทึมคือการหาค่าสัมประสิทธิ์ที่ถ่วงน้ำหนักตัวแปรอิสระแต่ละตัว จากนั้นแปลงผลลัพธ์ออกมาให้มีค่าอยู่ระหว่าง 0 กับ 1 ของสมการที่มีรูปแบบคล้ายตัว s ดังรูปที่ 2.1 (de Santis et al., 2017; ยุทธ ไกยวรรณ์, 2555)



รูปที่ 2.1 กราฟผลลัพธ์ของการถดถอยโลจีสติกส์ (ยุทธ ไกยวรรณ์, 2555)

โดยในการถดถอยโลจีสติกส์จะถูกใช้ในรูปแบบของความน่าจะเป็นของความสำเร็จหารด้วยความน่าจะเป็นของความล้มเหลวหรือเรียกอีกอย่างว่า log odds ซึ่งฟังก์ชันโลจีสติกส์แสดงโดยสูตรดังสมการที่ (1) และ (2)

$$\text{LOGIST}(\eta) = \frac{1}{1+\exp(-\eta)} \quad (1)$$

$$\text{Ln}\left(\frac{\eta}{1-\eta}\right) = \beta_0 + \beta_1 x_1 + \ldots + \beta_k K_k \quad (2)$$

จากสมการที่ (2) เมื่อ $\text{LOGIST}(\eta)$ คือตัวแปรตาม , x คือตัวแปรอิสระ , β คือสัมประสิทธิ์หน้าตัวแปรซึ่งประมาณจาก maximum likelihood estimation (MLE)

การคำนวณนี้จะใช้ค่าสัมประสิทธิ์ที่แตกต่างกัน และทำซ้ำหลายครั้งเพื่อให้ได้ค่าพารามิเตอร์ที่เหมาะสมที่สุด จากนั้นค่าพารามิเตอร์จะถูกนำมาใช้ในการพยากรณ์ สำหรับการจำแนกกลุ่มออกเป็น 2 กลุ่ม เมื่อคำนวณได้ค่าความน่าจะเป็นน้อยกว่า 0.5 จะพยากรณ์ว่าเป็นกลุ่ม 0 ขณะเดียวกันหากได้ความน่าจะเป็นมากกว่า 0.5 จะพยากรณ์ว่าเป็น 1 ซึ่งมีลักษณะดังรูปที่ 2.1

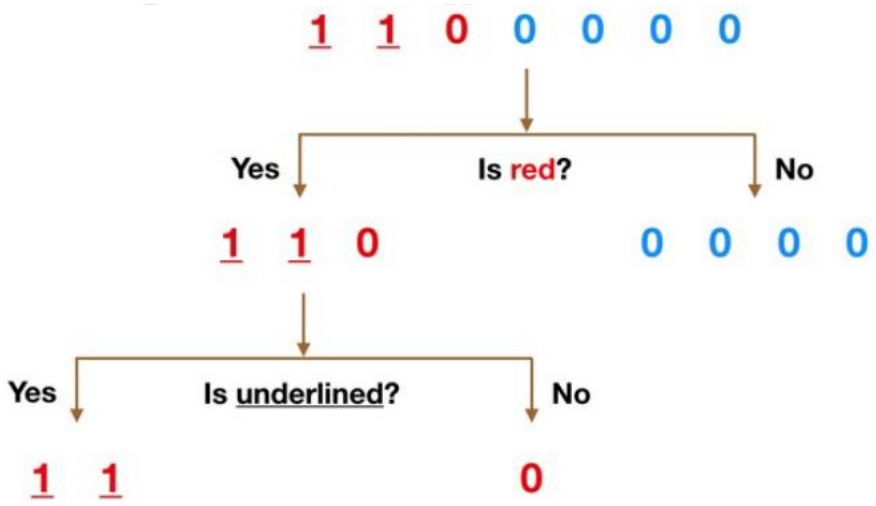
การถดถอยโลจิสติกส์มักใช้สำหรับการคาดการณ์และแก้ปัญหาการจำแนกประเภท ทำให้มีการนำไปใช้ประโยชน์ในหลายด้านเช่น ด้านการตรวจจับการฉ้อโกง ตัวแบบการถดถอยโลจีสติกส์สามารถช่วยระบุความผิดปกติของข้อมูลได้จากพฤติกรรมหรือลักษณะเฉพาะบางอย่างที่มีความสัมพันธ์ที่สูงกับการฉ้อโกง ซึ่งเป็นประโยชน์อย่างยิ่งต่อการธนาคารและสถาบันการเงินอื่นๆ ในการปกป้องลูกค้าของตน นอกจากนี้ในด้านการแพทย์ก็ได้มีการใช้ตัวแบบเพื่อพยากรณ์ความน่าจะเป็นของการเกิดโรคหรือความเจ็บป่วยด้วย

การถดถอยโลจีสติกส์สำหรับการเรียนรู้ของเครื่องนั้นจัดอยู่ในประเภทของการเรียนรู้ด้วยเครื่องแบบมีผู้สอน (supervised machine learning) ทั้งยังมีลักษณะของการจำแนกข้อมูลออกเป็นกลุ่ม ทว่าการพยากรณ์ด้วยการเรียนรู้ของเครื่องจะมีการคำนวณที่ต่างจากเดิมเล็กน้อย โดยจากเดิมที่เป็นการหาฟังก์ชัน log ที่มากที่สุดเพื่อประมาณค่าสัมประสิทธิ์หน้าตัวแปร จะเปลี่ยนค่าฟังก์ชั่น log ให้อยู่ในรูปแบบติดลบเพื่อใช้เป็นฟังก์ชันการสูญเสีย (loss function) จากนั้นใช้กระบวณการ gradient descent ในการหาค่าที่สูงที่สุด (global maximum) จากการเปลี่ยนแปลงนี้หากภายในตัวแบบมีตัวแปรอิสระที่ใช้พยากรณ์มาก อาจทำให้เกิดปัญหา Overfitting ได้จึงควรใช้การ Regularization เพื่อลดปัญหานี้ (IBM, n.d.)

จากลักษณะการพยากรณ์ของอัลกอริทึมนี้จึงเหมาะกับวิทยานิพนธ์ฉบับนี้ที่จำแนกว่าสินค้าใดที่จะเป็นสินค้าคงค้างหรือไม่เป็นสินค้าคงค้าง และยังใช้เป็นอัลกอริทึมพื้นฐานเพื่อเปรียบเทียบกับตัวแบบอื่นอีกด้วย

2.1.2 Random forest (FOREST) เป็นอัลกอริทึมที่ใช้เทคนิค Ensemble Learning ด้วยวิธี Bootstrap Aggregation (Bagging) คือการรวมตัวแบบ Decision Tree ซึ่งแต่ละตัวแบบจะสุ่มข้อมูล และคุณลักษณะแบบแทนที่ (random with replacement) จากนั้นพยากรณ์กลุ่มด้วยการโหวตว่าข้อมูลอยู่กลุ่มไหนมากที่สุด ทำให้ผลลัพธ์ที่ได้ดีกว่า Decision Tree และลดการเกิด Overfitting (TITIPATA, n.d.; ชิตพงษ์ กิตตินราดร, 2563)

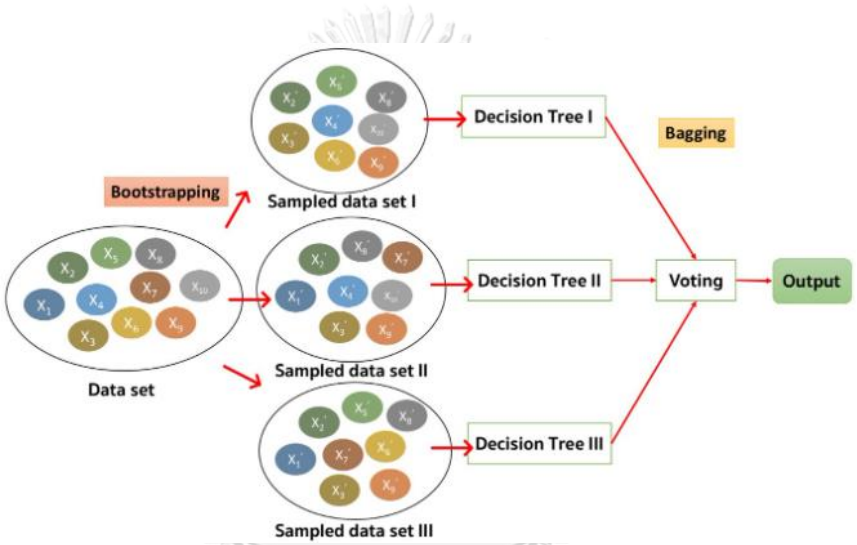
จากข้อความโดยสรุปข้างต้น การที่จะสร้างตัวแบบ Random forest ได้นั้นต้องสร้างตัวแบบ Decision Tree หลายตัวแบบซึ่งมีหลักการในการจำแนกข้อมูลอย่างง่ายดังรูปที่ 2.2



รูปที่ 2.2 หลักการจำแนกข้อมูลอย่างง่าย (Yiu, 2019)

จากรูปที่ 2.2 ซึ่งต้องการจำแนกข้อมูลตัวเลขในแถวบนสุด ในขั้นแรกสีเป็นคุณลักษณะที่สามารถใช้แบ่งชุดข้อมูลได้อย่างชัดเจน จึงใช้คำถามที่ว่า มันเป็นสีแดงหรือไม่ (Is red?) ในการแยกข้อมูลเป็นสองกลุ่ม จะเห็นได้ว่าในกลุ่มของไม่ใช่ (No) นั้น เป็นข้อมูลที่เหมือนกันทั้งหมดจึงไม่สามารถแบ่งกลุ่มได้อีก ทว่าในกลุ่มของใช่ (Yes) นั้นยังมีการแบ่งกลุ่มได้อีก จึงแบ่งกลุ่มอีกครั้งด้วยคำถาม มีการขีดเส้นหรือไม่ (Is underlined?) ซึ่งผลที่ได้คือสามารถแบบกลุ่มชุดข้อมูลทั้งหมดได้อย่างสมบูรณ์ (Yiu, 2019)

จากหลักการ Decision Tree ที่กล่าวไปข้างต้นนั้นนับเป็นหนึ่งตัวแบบซึ่ง Random forest เป็นการรวมผลการพยากรณ์ของตัวแบบ Decision Tree หลายตัวแบบ ดังรูปที่ 2.3



รูปที่ 2.3 หลักการทำงานของ FOREST (Daroontham, 2561)

จากรูปตัวอย่างซึ่งมีชุดข้อมูลซึ่งมีคุณลักษณะ 10 อย่าง จำนวน n ข้อมูล ถูกแบ่งข้อมูลออกเป็น 3 ชุด โดยในแต่ละชุดนั้นจะไม่ใช้คุณลักษณะ และ จำนวนข้อมูลทั้งหมดทั้งหมด นอกจากนี้คุณลักษณะ และ จำนวนข้อมูลในแต่ละชุดนั้นจะถูกสุ่มมาแตกต่างกันด้วย ซึ่งในแต่ละชุดข้อมูลจะทำการสร้างตัวแบบ Decision Tree และพยากรณ์ผลลัพธ์ หากคำตอบใดถูกพยากรณ์มากที่สุดก็จะถือว่าเป็นผลลัพธ์ของการพยากรณ์ วิธีนี้เป็นแนวคิดแบบภูมิปัญญาฝูงชน เนื่องจากแต่ละตัวแบบ Decision Tree เป็นอิสระต่อกันทำให้สามารถป้องกันความผิดพลาดของกันและกันได้ เนื่องจากหากมีบางตัวแบบที่พยากรณ์ผิด แต่ตัวแบบอื่นๆนั้นพยากรณ์ถูกผลลัพธ์ก็จะไปในทิศทางที่ถูกต้องได้ ดังนั้นหากอยากให้ตัวแบบ Random forest พยากรณ์ได้ดีในชุดข้อมูลที่ใช้นั้นควรมีบางคุณลักษณะที่มีความสัมพันธ์กับตัวแปรตามบ้างจะได้ผลลัพธ์ที่ดีกว่าแบบสุ่ม และ ในแต่ละตัวแบบ Decision Tree ควรมีค่าสัมพันธ์กันในระดับต่ำ (Daroontham, 2561)

2.1.3 eXtreme Gradient Boosting (XGBoost) เป็นอัลกอริทึมที่มีความยืดหยุ่น และให้ประสิทธิภาพดีที่สุดกลุ่มหนึ่งในปัจจุบัน เนื่องจากให้ค่า AUROC ที่ดีและใช้เวลาในการฝึกน้อยเมื่อเทียบกับอัลกอริทึมอื่น อัลกอริทึมนี้ใช้เทคนิค Ensemble Learning บนพื้นฐาน Decision Tree ด้วยวิธี Boosting หรือก็คือการนำข้อผิดพลาดของตัวแบบก่อนมาฝึกให้กับตัวแบบใหม่จนได้ตัวแบบที่ดีที่สุด นอกจากนี้ยังเป็นอัลกอริทึมที่ได้รับความนิยมเนื่องจากสามารถใช้งานได้หลากหลาย เช่น การแก้ปัญหาการถอถอย, การจัดกลุ่ม, การจัดอันดับ และการพยากรณ์ตามที่ผู้ใช้กำหนด ทั้งยังรองรับหลายภาษา เช่น C++, Python, R, Java, Scala และ Julia (TITIPATA, n.d.; ชิตพงษ์ กิตตินราดร, 2563)

แม้ว่า XGBoost และ Gradient Boosting Machines (GBMs) จะมีวิธีการที่คล้ายคลึงกัน แต่ XGBoost ได้ปรับปรุงเพื่อเพิ่มประสิทธิภาพระบบและอัลกอริทึม โดยการเพิ่มประสิทธิภาพของระบบนั้น XGBoost มีการใช้งานแบบขนานเพื่อปรับปรุงระยะเวลาการเรียนรู้ ทั้งยังใช้พารามิเตอร์ max_depth ในการหยุดการแบ่งต้นไม้แทนแบบเดิม และทำงานแบบย้อนกลับซึ่งช่วยเพิ่มประสิทธิภาพการคำนวณได้อย่างมาก นอกจากนี้ยังถูกออกแบบมาให้ใช้ทรัพยากรฮาร์ดแวร์อย่างมีประสิทธิภาพ ในส่วนของการเพิ่มประสิทธิภาพอัลกอริทึมนั้นได้ปรับความซับซ้อนของตัวแบบผ่านวิธี LASSO (L1) และ Ridge (L2) regularization เพื่อป้องกันการเกิด Overfitting ทั้งยังสามารถจัดการกับข้อมูลที่หายไป และรับมือกับรูปแบบการกระจายตัวของข้อมูลได้อย่างมีประสิทธิภาพ มีการใช้อัลกอริทึม Weighted Quantile Sketch เพื่อค้นหาจุดแบ่งที่เหมาะสมที่สุดระหว่างชุดข้อมูลแบบถ่วงน้ำหนัก ทั้งยังมีการตรวจสอบแบบไขว้ในแต่ละครั้งที่เรียนรู้อีกด้วย (Morde, 2019)

2.2 การวัดประสิทธิภาพตัวแบบ

2.2.1 Confusion Matrix เป็นพื้นฐานการวัดประสิทธิภาพของการจัดกลุ่ม ซึ่งเกิดจากการนำข้อมูลสำหรับฝึกไปสร้างตัวแบบ และนำตัวแบบนั้นมาใช้กับข้อมูลชุดทดสอบ ผลที่เกิดจากตัวแบบพยากรณ์เทียบกับข้อมูลจริงจะมีลักษณะตามตารางที่ 1 ตัวแปรดังกล่าวใช้คำนวณค่า Accuracy, Recall, Precision ฯลฯ

ตารางที่ 1 Confusion Matrix

	Positive prediction	Negative prediction
Positive class	True Positive (TP)	False Negative (FN)
Negative class	False Positive (FP)	True Negative (TN)

โดยที่ TP (True Positive) คือจำนวนข้อมูลที่ทำนายถูกต้องในกลุ่มมาก ,TN (True Negative) คือจำนวนข้อมูลที่ทำนายถูกต้องในกลุ่มน้อย ,FP (False Positive) คือจำนวนข้อมูลที่ทำนายผิดว่าอยู่ในกลุ่มมาก และFN (False Negative) คือจำนวนข้อมูลที่ทำนายผิดว่าอยู่ในกลุ่มน้อย

ตัวอย่างเช่น ผลการพยากรณ์สินค้าคงค้างจากข้อมูลชุดทดสอบ 100 รายการ ประกอบด้วยสินค้าที่มีโอกาสเป็นสินค้าคงค้าง 50 รายการ และสินค้าที่ไม่เป็นสินค้าคงค้างอีก 50 รายการ แสดงผล Confusion Matrix ดังตารางที่ 2

ตารางที่ 2 ตัวอย่าง Confusion Matrix

	พยากรณ์ว่าไม่เป็นสินค้าคงค้าง	พยากรณ์ว่าเป็นสินค้าคงค้าง
ข้อมูลจริงไม่เป็นสินค้าคงค้าง	43 (TP)	7 (FN)
ข้อมูลจริงเป็นสินค้าคงค้าง	3 (FP)	47 (TN)

จากตารางที่ 3 สรุปผลการพยากรณ์ได้ว่า จากข้อมูลสินค้าที่ไม่เป็นสินค้าคงค้าง 50 รายการ ตัวแบบพยากรณ์ว่าไม่เป็นสินค้าคงค้าง 43 รายการ และพยากรณ์ผิดว่าเป็นสินค้าคงค้าง 7 รายการ ในส่วนของข้อมูลสินค้าคงค้าง 50 รายการนั้นพยากรณ์ถูกว่าเป็นสินค้าคงค้าง 47 รายการ และพยากรณ์ผิดว่าไม่เป็นสินค้าคงค้าง 3 รายการ

2.2.2 การคำนวณค่า Recall ใช้วัดประสิทธิภาพของตัวแบบว่ามีโอกาสจำแนกข้อมูลกลุ่มที่ต้องการสินค้าคงค้างมากเพียงใด หรืออาจกล่าวได้ว่าจากข้อมูลจริงที่ส่งมา 50 รายการ มีการตอบถูกกี่รายการเมื่อเทียบกับข้อมูลจริง โดยคำนวณจากสมการที่ (3)

$$\text{Recall} = TP / TP+FN \qquad (3)$$

จากตารางที่ 2 และสมการที่ (3) สามารถคำนวณค่า Recall ของข้อมูลที่เป็นสินค้าคงค้างและข้อมูลที่ไม่เป็นสินค้าคงค้างได้ดังนี้

Recall ของข้อมูลที่ไม่เป็นสินค้าคงค้าง = $43 / 43 + 7 = 0.86$

Recall ของข้อมูลที่เป็นสินค้าคงค้าง = $47 / 47 + 3 = 0.94$

จากค่า Recall ที่ได้ข้างต้นนั้นหมายถึงตัวแบบสามารถพยากรณ์ข้อมูลที่เป็นสินค้าคงค้างได้ดีกว่าข้อมูลที่ไม่เป็นสินค้าคงค้าง ดังนั้นตัวแบบนี้เหมาะกับการพยากรณ์ข้อมูลสินค้าคงค้างมากกว่า

2.2.3 การคำนวณค่า Precision ใช้วัดร้อยละในการค้นหาเทียบกับความผิดพลาดที่เกิดขึ้นรอบข้างของตัวแบบ หรืออาจกล่าวได้ว่าจากการพยากรณ์คำตอบในกลุ่มนั้น พยากรณ์ถูกกี่รายการเมื่อเทียบกับรายการที่พยากรณ์ว่าเป็นกลุ่มนั้น โดยคำนวณจากสมการที่ (4)

$$\text{Precision} = TP / TP+FP \quad (4)$$

จากตารางที่ 3 และสมการที่ 3 สามารถคำนวณค่า Precision ของข้อมูลที่เป็นสินค้าคงค้างและข้อมูลที่ไม่เป็นสินค้าคงค้างได้ดังนี้

Precision ของข้อมูลที่ไม่เป็นสินค้าคงค้าง = $43 / 43 + 3 = 0.93$

Precision ของข้อมูลที่เป็นสินค้าคงค้าง = $47 / 47 + 7 = 0.87$

จากค่า Precision ที่ได้ข้างต้นนั้นหมายถึงตัวแบบมีการพยากรณ์ข้อมูลที่ไม่เป็นสินค้าคงคลังได้ง่ายกว่าข้อมูลที่เป็นสินค้าคงค้าง เนื่องจากพยากรณ์ผิดไป 3 รายการเมื่อเทียบกับอีกกลุ่มที่ผิดไป 7 รายการ (RPG, 2561)

2.2.4 F1 score : จากหัวข้อ 2.2.1 และ 2.2.2 จะเห็นได้ว่าหากพิจารณาทั้งสองส่วนพร้อมกันในบางครั้งอาจทำได้ยาก เนื่องจากเป็นไปได้ว่าตัวแบบหนึ่งให้ค่า Recall ที่ดีแต่ Precision น้อย กลับกันอาจมีอีกตัวแบบที่ค่า Precision ดีแต่ Recall น้อย ดังนั้น F1 score จึงเป็นการรวม Precision และ Recall เข้าเป็นตัวชี้วัดเดียว โดยคำนวณจากสมการที่ (5) ในขณะเดียวกัน F1 score ยังถูกออกแบบมาให้ทำงานได้ดีกับชุดข้อมูลไม่สมดุล (Korstanje, 2021)

$$\text{F1 score} = 2^* \frac{\text{Precision}^* \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5)$$

2.2.5 AUROC (Area Under the Receiver Operating Characteristics) เป็นหนึ่งในการประเมิณที่สำคัญที่สุดในการวัดประสิทธิภาพการจำแนกข้อมูล หากค่าของ AUCROC สูงก็หมายความว่าตัวแบบมีประสิทธิภาพในการจำแนกดี ใช้เป็นข้อสรุปของ Receiver Operating Characteristics (ROC) ซึ่งเป็นการหาจุดตัดในการจำแนกข้อมูลเพื่อให้ได้ค่า Recall และความจำเพาะ (Specificity) ที่ดีที่สุด โดยกราฟสร้างจากการวาดจุดความน่าจะเป็นของค่า True Positive Rate (TPR) หรือ Recall ที่ได้กล่าวไปในหัวข้อก่อนหน้า เทียบกับค่า False Positive Rate (FPR) ที่หลักเกณฑ์ต่างๆ และแยกสัญญาณออกจากสัญญาณรบกวน ซึ่งค่าความจำเพาะสามารถคำนวณได้ดังสมการที่ (6) ,ค่า

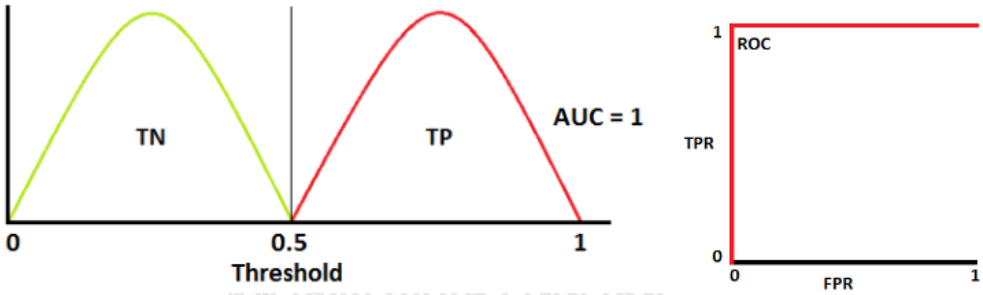
False Positive Rate (FPR) สามารถคำนวณได้ดังสมการที่ (7) และ AUROC สามารถคำนวณได้ดังสมการที่ (8)

$$\text{ความจำเพาะ} = \frac{TN}{FP + TN} \tag{6}$$

$$\text{False Positive Rate} = \frac{FP}{FP + TN} \tag{7}$$

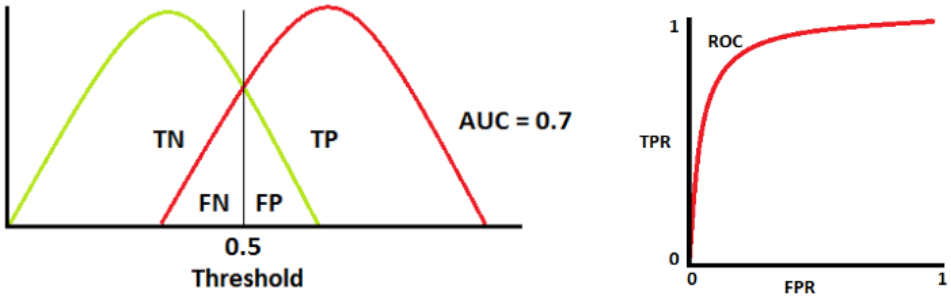
$$AUROC = \frac{1 + \frac{TP}{TP + FP} - \frac{FP}{FP + TN}}{2} \tag{8}$$

เมื่อได้ค่า AUROC เท่ากับ 1 หมายความว่าตัวแบบสามารถจัดกลุ่มข้อมูลได้ถูกต้องทั้งหมด ในทางกลับกันหากได้ค่าเท่ากับ 0 จะหมายถึงตัวแบบพยากรณ์ผิดกลุ่มทั้งหมด โดยกราฟที่ได้จะมีลักษณะดังรูปที่ 2.4



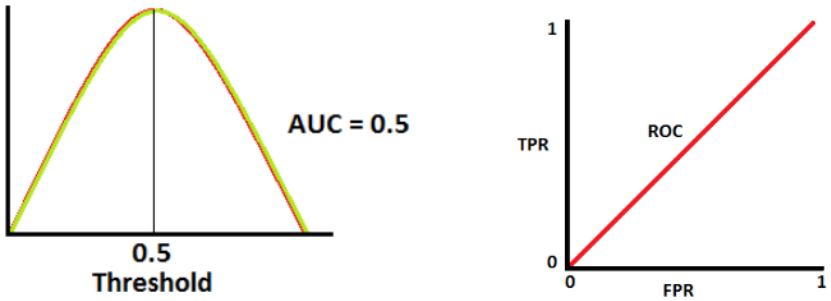
รูปที่ 2.4 กราฟ ROC เมื่อ AUROC เท่ากับ 1 (Narkhede, 2018)

เมื่อได้ค่า AUROC มากกว่า 0.5 แต่น้อยกว่า 1 หมายความว่ามีโอกาสสูงที่จะจัดกลุ่มข้อมูลได้ เนื่องจากมีจำนวน True Positive และ True Negative มากกว่า False negative และ False positives โดยกราฟที่ได้จะมีลักษณะดังรูปที่ 2.5



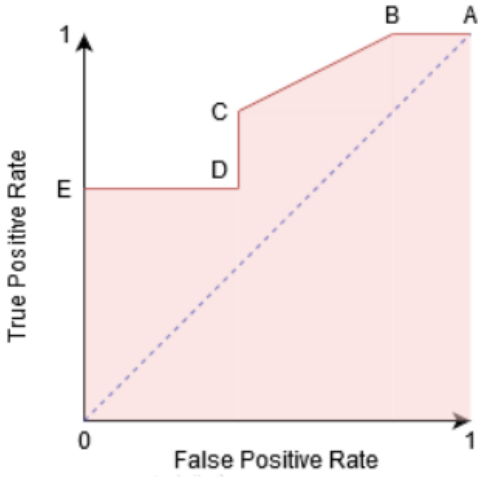
รูปที่ 2.5 กราฟ ROC เมื่อ AUROC มากกว่า 0.5 แต่น้อยกว่า 1 (Narkhede, 2018)

เมื่อได้ค่า AUROC เท่ากับ 0.5 หมายความว่าตัวแบบไม่สามารถจัดกลุ่มข้อมูลได้ เช่นตัวแบบมีการพยากรณ์ข้อมูลแบบสุ่ม หรือ ตัวแบบมีการพยากรณ์ข้อมูลเพียง 1 กลุ่มสำหรับทุกข้อมูล โดยกราฟที่ได้จะมีลักษณะดังรูปที่ 2.6



รูปที่ 2.6 กราฟ ROC เมื่อ AUROC เท่ากับ 0.5 (Narkhede, 2018)

จากกราฟที่กล่าวมาข้างต้นทั้ง 3 ลักษณะ ค่าแกน X ที่สูงกว่าบ่งชี้ว่าจำนวน False positive มากกว่าจำนวน True negative ในขณะที่ค่าแกน Y ที่สูงกว่าบ่งชี้ว่าจำนวน True positive มากกว่า False negative ดังนั้น การเลือกเกณฑ์ขึ้นอยู่กับความสามารถในการสร้างสมดุลระหว่าง False positive กับ False negative ซึ่งเกณฑ์ที่ต่างกันทำให้ประสิทธิภาพตัวแบบต่างกันดังรูปที่ 2.7 โดยตารางที่ 3 แสดง Confusion Matrix ของรูปที่ 2.7



รูปที่ 2.7 ตัวอย่างกราฟ ROC (Bhandari, 2022)

ตารางที่ 3 ค่า Confusion Matrix ของแต่ละจุด

จุด	A	B	C	D	E
TP	5	5	4	3	3
TN	0	1	3	3	5
FP	5	4	2	2	0
FN	0	0	1	2	2

จากรูปที่ 2.7 และตารางที่ 3 จุด A คือจุดที่มีค่า Recall สูงที่สุดและค่าความจำเพาะต่ำที่สุด ซึ่งหมายความว่าตัวแบบสามารถจัดกลุ่มข้อมูลในกลุ่มบวกได้ถูกทั้งหมด แต่ไม่สามารถจัดกลุ่มข้อมูลกลุ่มลบได้ ส่วนของจุด B แม้ว่าจะมีค่า Recall ที่เท่ากับจุด A แต่มีค่าความจำเพาะสูงกว่าทำให้เกณฑ์ B ดีกว่า A ในส่วนของจุด C เมื่อเทียบกับจุด D ที่มีค่าความจำเพาะเท่ากัน จุด C มีค่า Recall ที่สูงกว่า ซึ่งหมายความว่าตัวแบบจัดกลุ่มกลุ่มลบผิดและถูกเท่ากัน (TN และ FP เท่ากัน) แต่สามารถจัดกลุ่มในกลุ่มบวกได้มากกว่าดังนั้นจุด C จึงดีกว่า ลำดับสุดท้าย จุด E มีค่าจำเพาะที่สูงที่สุดซึ่งหมายความว่าตัวแบบสามารถจัดกลุ่มข้อมูลกลุ่มลบได้ทั้งหมด จากผลของเกณฑ์ที่ได้กล่าวไปข้างต้นนั้นสามารถเลือกได้หลายจุดตามความเหมาะสมของการนำตัวแบบไปใช้งาน อาจเลือกจุดใดจุดหนึ่งระหว่างจุด B และ C ในการยอมรับว่าจะให้ตัวแบบทำนายข้อมูลกลุ่มลบหรือบวกมากกว่ากัน เนื่องจากค่า Recall และ ค่าความจำเพาะเป็นสัดส่วนผกผันซึ่งกันและกัน เมื่อลดเกณฑ์ลงก็จะได้ค่ากลุ่มบวกเพิ่มมากขึ้น ค่า Recall เพิ่มขึ้นแต่ค่าความจำเพาะจะลดลง ในทำนองเดียวกันเพื่อเพิ่มเกณฑ์ก็จะได้ค่ากลุ่มลบมากขึ้น ความจำเพาะมากขึ้นแต่ค่า Recall ลดลง นอกจากนี้สามารถเลือกจุด E

หากความถูกต้องในการพยากรณ์ข้อมูลในกลุ่มลบมีความสำคัญมากกว่ากลุ่มบวก โดยในวิทยานิพนธ์ฉบับนี้จะใช้ AUROC ในการเลือกพารามิเตอร์ที่สร้างตัวแบบที่ดีที่สุด ซึ่งจะบอกภาพรวมของความถูกต้องในการจำแนกข้อมูล (Bhandari, 2022; de Santis et al., 2017; Sarakarn & Munpolsri, 2021)

2.2.6 การวัดประสิทธิภาพที่เหมาะกับข้อมูลไม่สมดุลมีหลายตัวชี้วัดขึ้นอยู่กับผลลัพธ์ที่ต้องการ โดยหัวข้อที่ 2.2.4 F1 score จะเห็นได้ว่าตัวชี้วัดนี้ให้ความสำคัญกับ Recall และ Precision เท่ากัน จึงเหมาะกับงานที่ให้ความสำคัญกับข้อมูลกลุ่มบวก (Positive class) มากกว่า ขณะที่ให้ความสำคัญกับ False Negatives และ False Positives เท่ากัน หรือหากให้ความสำคัญกับข้อมูลกลุ่มบวกเท่ากับกลุ่มลบ (Negative class) การใช้ตัวชี้วัด G-mean ก็จะเหมาะกับข้อมูลที่มีความไม่สมดุลสูงเช่น มีข้อมูลกลุ่มมากอยู่ร้อยละ 80 ถึง 90 ของข้อมูล ซึ่งตัวชี้วัดทั้งสองนี้จะจำแนกกลุ่มอย่างชัดเจน ในขณะเดียวกันหากต้องการพยากรณ์ความน่าจะเป็นโดยที่ต้องการผลลัพธ์เป็นการจำแนกกลุ่ม ตัวชี้วัด AUROC ในหัวข้อที่ 2.2.5 จะเหมาะกับงานที่ให้ความสำคัญกับข้อมูลกลุ่มบวกและข้อมูลกลุ่มลบเท่ากัน แต่หากต้องการให้ความสำคัญกับข้อมูลกลุ่มบวกมากกว่าควรใช้ตัวชี้วัด Precision-Recall AUC ซึ่งเป็นพื้นที่ใต้กราฟระหว่าง Precision และ Recall (Brownlee, 2020a)

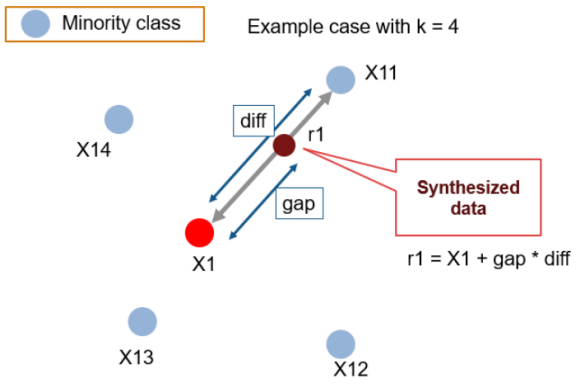
จากข้อมูลดังกล่าวผู้วิจัยจึงเลือกใช้ตัวชี้วัด AUROC ในส่วนของข้อมูลที่ใช้วิธีการปรับระดับของข้อมูลเนื่องจากระดับของข้อมูลที่เท่ากันและลักษณะข้อมูลที่เป็นการพยากรณ์สินค้าคงค้างหากให้ความสำคัญกับข้อมูลที่ไม่เป็นสินค้าคงค้างน้อยเกินไปอาจทำให้เกิดการจัดเก็บสินค้ามากขึ้น ส่งผลต่อค่าใช้จ่ายในการจัดเก็บที่มากขึ้น ทั้งนี้จะใช้ F1 score กับการปรับระดับข้อมูลด้วยเนื่องจากบางชุดข้อมูลที่เปรียบเทียบไม่ได้มีการปรับระดับข้อมูล ส่วนในการจัดการกับข้อมูลไม่สมดุลด้วยวิธี Threshold-moving จะใช้ตัวชี้วัด F1 score เนื่องจากข้อมูลไม่ได้มีระดับเท่ากันจึงให้ความสำคัญกับข้อมูลกลุ่มบวกมากกว่า นอกจากนี้ยังใช้ G-mean ที่เหมาะกับข้อมูลไม่สมดุลสูงหรือก็คือข้อมูลที่เป็นสินค้าคงค้างในวิทยานิพนธ์ฉบับนี้ สุดท้ายนี้เนื่องจากวิธีการจัดการข้อมูลที่ต่างกันจึงพิจารณาจาก Confusion Matrix เพิ่มเติมเพื่อหาวิธีการที่เหมาะสมที่สดกับชุดข้อมูล

2.3 การปรับเพิ่ม / ลดระดับข้อมูล

วิธีการปรับเพิ่ม / ลดระดับข้อมูลเป็นวิธีที่ใช้บ่อยที่สุดในการจัดการกับปัญหาข้อมูลไม่สมดุล ซึ่งวิธีนี้เป็นขั้นตอนก่อนเข้าสู่การเรียนรู้ด้วยเครื่อง โดยมีหลักการคือการสร้างข้อมูลขึ้นมาใหม่เพื่อให้มีความสมดุลมากยิ่งขึ้น เช่น การปรับเพิ่มระดับข้อมูลคือการข้อมูลกลุ่มน้อยขึ้นมาเพิ่ม ส่วนการปรับลดระดับข้อมูลคือการลบข้อมูลกลุ่มมาก หรืออาจทำทั้งสองวิธีการร่วมกัน ซึ่งวิธีการที่กล่าวมากข้างต้น

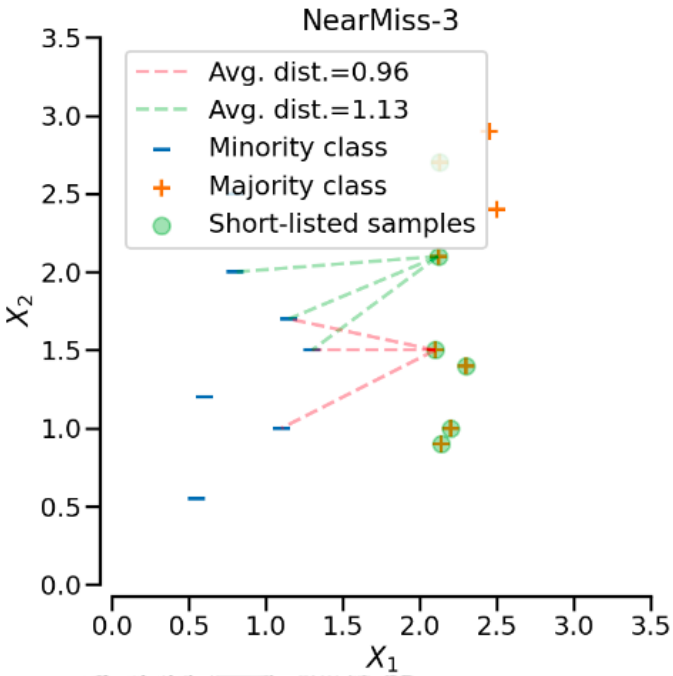
ยังสามารถทำได้หลากหลายไม่ว่าจะด้วยวิธีการสุ่ม หรือวิธีฮิวริสติก โดยในวิทยานิพนธ์ฉบับนี้มีวิธีการหลัก 3 วิธีได้แก่

2.3.1 Synthetic Minority Oversampling Technique (SMOTE) เป็นเทคนิคการสังเคราะห์ข้อมูลกลุ่มน้อยด้วยอัลกอริทึมของ k-Nearest Neighbor (kNN) ซึ่งตำแหน่งข้อมูลที่สังเคราะห์นั้นมีขั้นตอนในการคำนวณคือ ขั้นตอนแรกให้สุ่มตัวอย่างจากข้อมูลกลุ่มน้อย (จุดสีแดง) จากนั้นระบุ kNN (จุดสีฟ้า) และเลือก 1 ตัว (X11) เพื่อระบุเวกเตอร์ระหว่างจุดข้อมูลปัจจุบันกับเพื่อนบ้านที่เลือก (diff) ต่อมาจึงคูณเวกเตอร์ด้วยตัวเลขสุ่มระหว่าง 0 ถึง 1 (gap) แล้วนำมาบวกกับจุดสีแดงก็จะได้เป็นจุดใหม่ (r1) ดังรูปที่ 2.8 ด้วยวิธีการสังเคราะห์ข้อมูลนั้นทำให้ไม่เกิดปัญหา Overfitting เมื่อเทียบกับการสร้างข้อมูลซ้ำกับข้อมูลที่มีอยู่ แต่อย่างไรก็ตามการเพิ่มระดับข้อมูลนั้นส่งผลกับขนาดข้อมูลรวม อาจทำให้การเรียนรู้ของเครื่องช้าลงได้งานวิจัยฉบับนี้กำหนดให้ค่า k คือ 5 (พุทธิพร ธนธรรมเมธี และ เยาวเรศ ศิริสถิตย์กุล,2562;SATPATHY, 2021)



รูปที่ 2.8 หลักการ SMOTE (SATPATHY, 2021)

2.3.2 Near Miss Undersampling-3 (NearMiss-3) เป็นหนึ่งในสามรูปแบบของวิธี Near Miss Undersampling ที่มีการใช้อย่างแพร่หลายในการลดระดับข้อมูลเนื่องจากวิธีนี้สามารถป้องกันข้อมูลสูญหายได้ โดยมีหลักการคือ การปรับลดข้อมูลกลุ่มใหญ่ด้วยอัลกอริทึม kNN ซึ่งเป็นการเลือกเก็บข้อมูลกลุ่มใหญ่จากระยะทางที่สั้นที่สุดระหว่างข้อมูลกลุ่มน้อย โดยวิธีนี้จะเลือกเก็บเฉพาะข้อมูลกลุ่มใหญ่ที่อยู่ในขอบเขตการตัดสินใจ การเลือกเก็บข้อมูลในรูปแบบที่ 3 ประกอบไปด้วย 2 ขั้นตอน คือ ขั้นตอนแรก เลือกข้อมูลกลุ่มน้อย k ตัว จากนั้นจะเลือกกำจัดข้อมูลกลุ่มมากจากระยะทางเฉลี่ยมากที่สุดของ kNN ดังรูปที่ 2.9 นอกจากนี้รูปแบบที่ 3 ยังรับผลกระทบจากข้อมูลรบกวนน้อยเนื่องจากมีการเลือกข้อมูลตัวอย่างในขั้นแรก งานวิจัยฉบับนี้กำหนดให้ค่า k คือ 3 (Brownlee, 2020b; YAĞCI, 2021)



รูปที่ 2.9 หลักการ NearMiss-3 (YAĞCI, 2021)

2.3.3 One-Sided Selection for Undersampling (OSS) เป็นหนึ่งในเทคนิคในการปรับลดข้อมูลกลุ่มใหญ่ด้วยการเลือกเก็บผสมกับการเลือกทิ้งข้อมูล วิธีนี้เป็นการรวมกฎของ วิธี Tomek Links ซึ่งจะลบข้อมูลกลุ่มใหญ่ที่คลุมเคลือภายใต้ขอบเขตการตัดสินใจ และลบข้อมูลที่เป็นข้อมูลรบกวน จากนั้นวิธี the Condensed Nearest Neighbor (CNN) จะลบข้อมูลกลุ่มใหญ่ที่ซ้ำซ้อนออก (Brownlee, 2020b)

2.4 Threshold-moving

Threshold-moving เป็นหนึ่งในวิธีการในการจัดกลุ่มสำหรับข้อมูลไม่สมดุลด้วยแนวคิดที่เรียบง่ายและมีประสิทธิภาพ โดยมีลักษณะเด่นที่แตกต่างจากวิธีอื่นเนื่องจากเป็นการหาเกณฑ์ที่เหมาะสมหลังจากที่สร้างตัวแบบเรียนรู้ แนวคิดนี้เกิดจากการหลีกเลี่ยงการปรับระดับข้อมูลที่อาจส่งผลต่อความน่าจะเป็นของการพยากรณ์ที่ได้จากกลุ่มชุดข้อมูลฝึก ตัวแบบที่ได้จากการปรับข้อมูลนั้นอาจใช้ไม่ได้ผลกับข้อมูลฝึกและข้อมูลทดสอบที่ไม่สมดุลในอนาคต นอกจากนี้หากเป็นการปรับเพิ่มข้อมูลก็อาจทำให้เกิดปัญหา Overfitting ตามมาได้ หรือหากมีการใช้การตรวจสอบแบบไขว้ก็เป็นไปได้ว่าอาจมีบางข้อมูลที่ถูกทำซ้ำ หรือ สร้างขึ้นมาในหลาย fold ดังนั้นชุดข้อมูลในแต่ละ fold อาจจะไม่ได้เป็นอิสระต่อกัน

การหาหลักเกณฑ์ที่เหมาะสมสำหรับข้อมูลที่ไม่สมดุลนั้นอาจทำให้ผลลัพธ์ในการพยากรณ์ดีขึ้น จากเดิมที่เกณฑ์ในแต่ละอัลกอริทึมสำหรับการจัดกลุ่มส่วนใหญ่ตั้งไว้คือ 0.5 หมายถึง หากพยากรณ์ความน่าจะเป็นได้เท่ากับ หรือ มากกว่าข้อมูลจะถูกจัดไปอยู่ในข้อมูลกลุ่มหนึ่ง เช่นเดียวกันหากความน่าจะเป็นได้น้อยกว่า 0.5 ข้อมูลจะถูกจัดกลุ่มไปยังข้อมูลอีกกลุ่ม ซึ่งการหาเกณฑ์ที่เหมาะสมนั้นเริ่มจากการสร้างตัวแบบด้วยชุดข้อมูลฝึก และทำนายกับชุดข้อมูลทดสอบแล้ว จากนั้นปรับเกณฑ์โดยการค้นหาผลลัพธ์ประสิทธิภาพของแต่ละค่าเกณฑ์ในช่วงที่ตั้งไว้ หรือ สามารถคำนวณหาได้โดยตรงเพื่อให้ได้เกณฑ์ที่มีประสิทธิภาพมากที่สุด เกณฑ์ที่ได้ก็จะถูกนำมาใช้กับข้อมูลในอนาคต (Brownlee, 2021; Rosen, 2020)

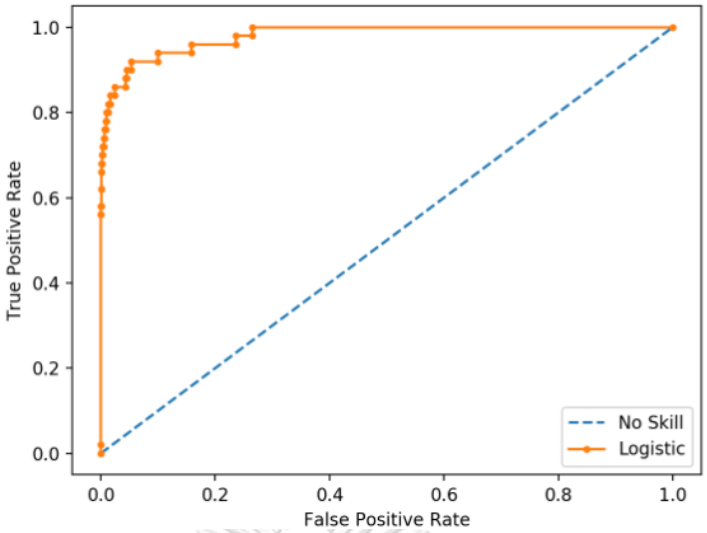
2.4.1 การหาเกณฑ์ที่เหมาะสมด้วยเส้นโค้ง ROC

เส้นโค้ง ROC เป็นเส้นที่ทำให้เข้าใจความสัมพันธ์ระหว่าง TPR และ FPR ดังที่ได้กล่าวไปแล้วในหัวข้อที่ 2.2.5 ซึ่งตัววัดประสิทธิภาพสำหรับหาเกณฑ์ที่เหมาะสมด้วยเส้นโค้ง ROC ประกอบไปด้วยค่า Geometric mean (G-Mean) ซึ่งสามารถคำนวณได้จากสมการที่ (10) และ Youden's J statistic ซึ่งคำนวณได้จากสมการที่ (11)

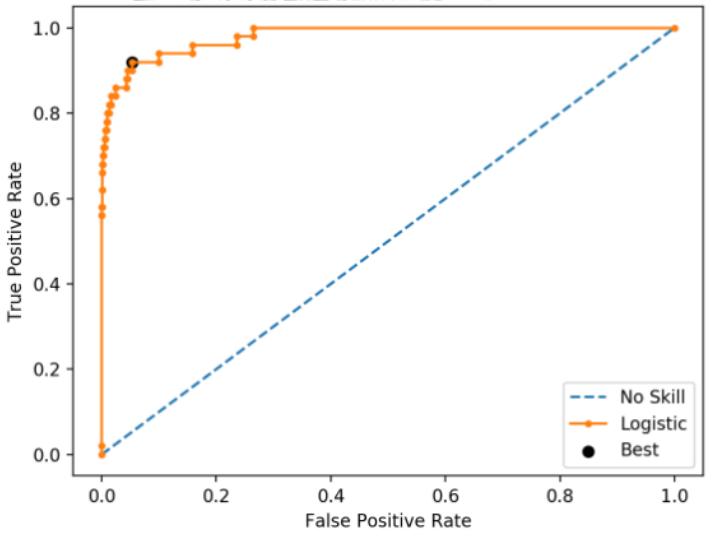
$$\text{G-Mean} = \text{sqrt}(\text{Recall} * \text{Specificity}) \tag{10}$$

$$\text{Youden's J statistic} = \text{TPR} - \text{FPR} \tag{11}$$

จากตัวอย่างการสร้างตัวแบบด้วยอัลกอริทึมการถดถอยโลจีสติกส์นั้นได้กราฟ ROC ดังรูปที่ 2.10 เมื่อดูจากเส้นสีส้มนั้นไม่อาจบอกได้แน่ชัดว่าที่จุดใดคือเกณฑ์ที่เหมาะสมของการจัดกลุ่ม ซึ่งเมื่อนำตัววัดประสิทธิภาพที่ได้กล่าวไปข้างต้นนั้นมาใช้หาเกณฑ์ที่เหมาะสมที่สุดจะได้ผลดังรูปที่ 2.11 โดยจุดสีดำคือจุดที่เป็นเกณฑ์ที่ดีที่สุด



รูปที่ 2.10 กราฟ ROC (Brownlee, 2021)

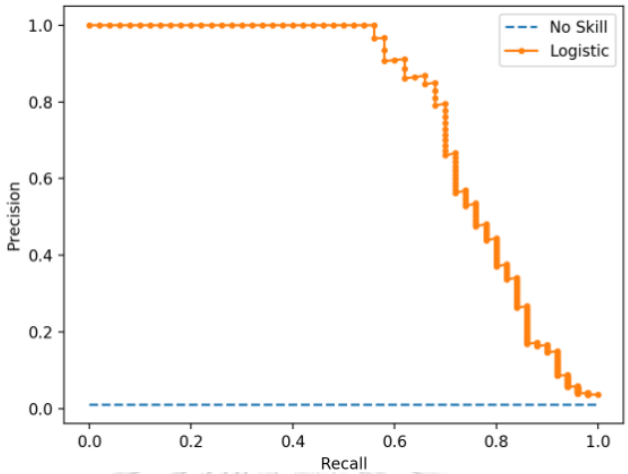


รูปที่ 2.11 จุดที่ดีที่สุดบนกราฟ ROC (Brownlee, 2021)

2.4.2 การหาเกณฑ์ที่เหมาะสมด้วยเส้นโค้ง Precision-Recall

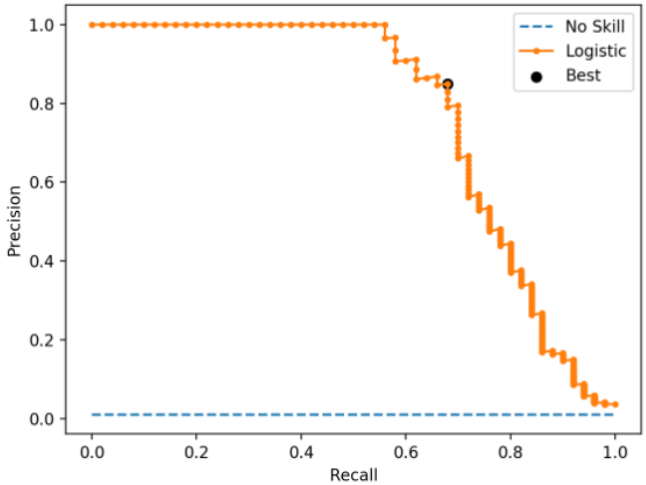
เส้นโค้ง Precision-Recall ต่างจากเส้นโค้ง ROC ตรงที่จะเน้นเพียงประสิทธิภาพของข้อมูลกลุ่มน้อย สามารถสร้างเส้นโค้งจากการคำนวณค่า Precision และ Recall จากจุดเกณฑ์ที่ต่างกัน โดยค่า Recall จะแสดงอยู่บนแกน X ส่วนค่า Precision จะแสดงอยู่บนแกน Y ดังรูปที่ 2.12

เส้นประสีฟ้าในแนวนอนหมายถึงค่า Precision ในข้อมูลกลุ่มบวก ส่วนตำแหน่งจุดเกณฑ์ที่มีประสิทธิภาพมากที่สุดจะอยู่มุมขวาบนของกราฟ



รูปที่ 2.12 กราฟ Precision-Recall (Brownlee, 2021)

การหาจุดเกณฑ์ที่ดีที่สุดสำหรับการสมดุลค่า Precision และ Recall คือ F1 score ที่ได้กล่าวในหัวข้อที่ โดยจุดเกณฑ์ที่ให้ผลลัพธ์ F1 score ดีที่สุดคือจุดสีดำด้านขวาบนดังรูป 2.13

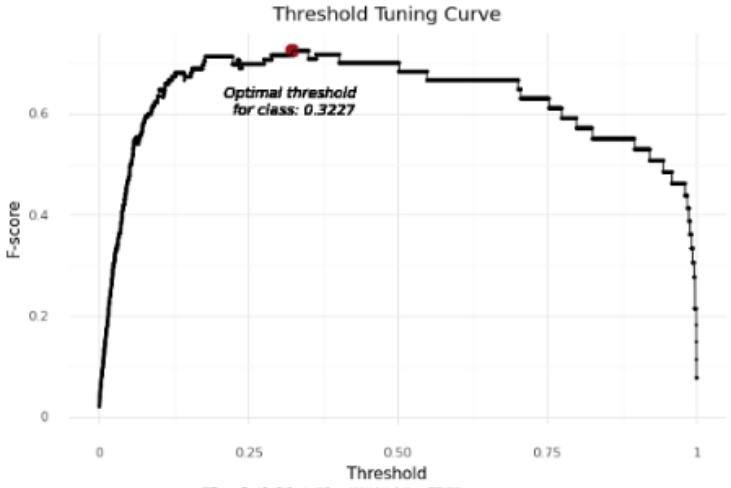


รูปที่ 2.13 จุดที่ดีที่สุดบนกราฟ Precision-Recall (Brownlee, 2021)

2.4.3 การหาเกณฑ์ที่เหมาะสมด้วยการปรับค่าเกณฑ์

การปรับค่าเกณฑ์เป็นวิธีการทั่วไปในการค่าจุดเกณฑ์ที่เหมาะสมที่สุด ซึ่งมีข้อดีคือสามารถปรับความละเอียดของการเลื่อนจุดภายในตำแหน่งที่ต้องการ แต่อาจมีค่าใช้จ่ายสูง เช่นการใช้โค้ด

np.arrange(0.0, 1.0, 0.0001) ซึ่งจะทำการวัดประสิทธิภาพจากจุดเกณฑ์ 10,000 จุด จากนั้นจะได้ผลลัพธ์ดังรูปที่ 2.14 (Brownlee, 2021)



รูปที่ 2.14 กราฟ Threshold Tuning Curve (Brownlee, 2021)

2.5 การปรับ Boxplot

Boxplot เป็นเครื่องมือที่ใช้อธิบายข้อมูลทางสถิติของชุดข้อมูล ไม่ว่าจะเป็นการกระจายตัว ค่าเฉลี่ยตำแหน่งข้อมูล รวมถึงข้อมูลที่เป็น Outlier อย่างไรก็ตาม เมื่อข้อมูลเบ้จะส่งผลต่อการบอกค่า Outlier โดยข้อมูลที่ปกติจะถูกบอกว่าไม่ปกติ ดังนั้นจึงมีการปรับ Boxplot เพื่อให้ตรวจสอบค่า Outlier สำหรับข้อมูลที่เบ้ ซึ่งในปี ค.ศ. 2008 M. Hubert และ E. Vandervieren ได้ปรับค่าด้วย medcouple (MC) ดังสมการที่ (12)

$$MC = med_{x_i \le Q2 \le x_j} h\left(x_i x_j\right) \in [-1,1] \tag{12}$$

เมื่อ Q2 หมายถึงค่ากลาง (Median) และ $X_i \neq X_j$ ส่วนฟังก์ชัน *h* เป็นไปดังสมการที่ (13)

$$h\left(x_i x_j\right) = \frac{(x_j - Q2) - (Q2 - x_i)}{x_j - x_i} \tag{13}$$

ค่า MC เป็นค่าบวกหมายถึงข้อมูลเบ้ขวา ในทางตรงข้ามหากเป็นค่าลบก็จะหมายถึงข้อมูลเบ้ซ้าย และข้อมูลที่สมมาตรจะได้ค่าเป็นศูนย์ โดยค่า MC จะนำมาคำนวณการสร้างหนวดหรือขอบที่

ข้อมูลจะเป็นค่า Outlier จากเดิมที่สามารถคำนวณได้จากสมการที่ (14) ด้วยการแทนที่ฟังก์ชัน $h_l(MC)$ และ $h_u(MC)$ ดังแสดงในสมการที่ (15)

$$[Q1\text{-}1.5IQR\ ;Q3 + 1.5IQR] \tag{14}$$

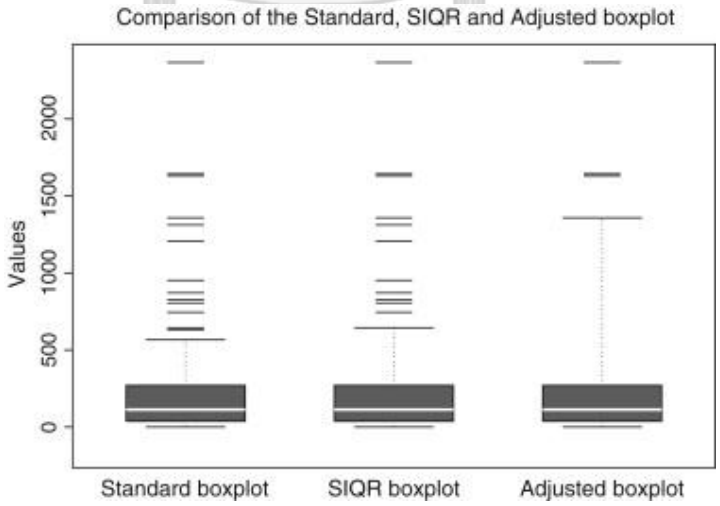
$$[Q1\text{-}h_l(MC)\ IQR\ ;Q3 + h_u(MC)IQR] \tag{15}$$

จากการทดลองสรุปได้ว่า หาก $-0.6 \leq MC \leq 0.6$ จะสามารถประมาณการหา Outlier สำหรับข้อมูลเบ้ขวาได้ดังสมการที่ (16) และสำหรับข้อมูลเบ้ซ้ายดังสมการที่ (17)

$$[Q1\text{-}1.5e^{-4\,MC}\ IQR\ ;Q3 + 1.5e^{3\,MC}IQR] \tag{16}$$

$$[Q1\text{-}1.5e^{-3\,MC}\ IQR\ ;Q3 + 1.5e^{4\,MC}IQR] \tag{17}$$

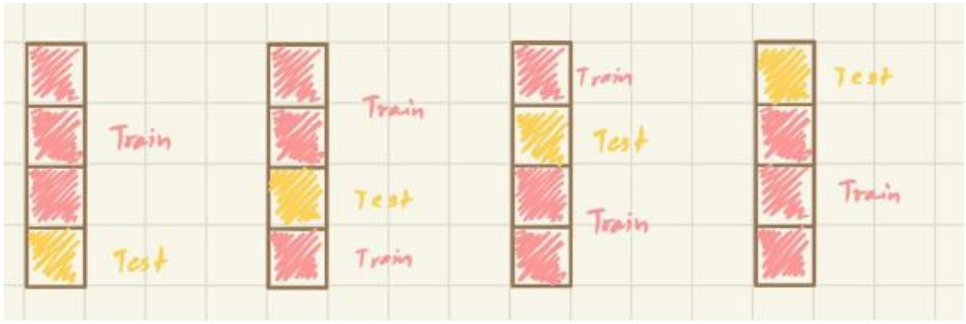
รูปที่ 2.15 แสดงลักษณะของ Boxplot ที่มีการปรับให้ทำงานสำหรับข้อมูลที่เบ้ (Adjusted boxplot) เทียบกับ Boxplot แบบมาตรฐาน (Standard boxplot) และ semi-interquartile range Boxplot (SIQR boxplot) (Hubert & Vandervieren, 2008)



รูปที่ 2.15 การเปรียบเทียบ Boxplot (Hubert & Vandervieren, 2008)

2.6 การตรวจสอบแบบไขว้ (Cross-validation)

การตรวจสอบแบบไขว้เป็นวิธีทางสถิติที่ใช้ประเมินทักษะของตัวแบบการเรียนรู้ด้วยเครื่อง โดยใช้กระบวณการสุ่มตัวอย่างจากข้อมูลที่มีจำกัดออกเป็น k ส่วนเพื่อให้ข้อมูลกระจายเท่า ๆ กัน ดังรูปที่ 2.16 แสดงการตรวจสอบแบบไขว้ 4 กลุ่ม (4 Fold Cross Validation) โดยเป็นข้อมูลฝึก 3 ส่วนและข้อมูลทดสอบ 1 ส่วน ส่งผลให้สามารถประเมินทักษะของตัวแบบที่มีต่อข้อมูลที่ไม่เคยเห็นมาก่อนได้ (stackpython, 2020)



รูปที่ 2.16 4 Fold Cross Validation (stackpython, 2020)

การเลือกใช้ค่า k ควรเลือกใช้อย่างระมัดระวังและให้เหมาะสมกับชุดข้อมูลที่มี หากเลือกผิดอาจสื่อถึงทักษะของตัวแบบที่ไม่ถูกต้องเช่น คะแนนที่มีความแปรปรวนสูงนั้นหมายถึงตัวแบบมีผลที่แตกต่างกันมากตามแต่ละข้อมูลที่ใช้ฝึก โดยการเลือกค่า k มีอยู่ 3 วิธีดังนี้

1. ค่า k ที่เลือกควรทำให้ข้อมูลสำหรับฝึกและทดสอบมีขนาดใหญ่พอที่จะเป็นตัวแทนทางสถิติของชุดข้อมูล
2. การกำหนดค่า k เท่ากับ 5 หรือ 10 ซึ่งเป็นค่าที่พบจากการทดลองทั่วไปที่ส่งผลให้ตัวแบบมีอคติต่ำ (bias) และมีความแปรปรวนน้อย
3. ค่าของ k ถูกกำหนดเป็น n โดยที่ n คือขนาดของชุดข้อมูลเพื่อให้แต่ละข้อมูลทดสอบมีโอกาสที่จะถูกทดสอบในทุกช่วงของชุดข้อมูล ซึ่งเรียกว่า leave-one-out cross-validation (Brownlee, 2018)

2.7 Hyperparameter Tuning

Hyperparameter Tuning คือการปรับค่าพารามิเตอร์ที่ผู้ทดลองกำหนดก่อนเริ่มการเรียนรู้ ซึ่งค่าที่เหมาะสมจะส่งผลดีต่อประสิทธิภาพของตัวแบบ บางครั้งจึงถูกเรียกว่า Hyperparameter Optimization โดยค่าพารามิเตอร์ที่กำหนดนั้นไม่สามารถรู้ล่วงหน้าได้ว่าเป็นค่าที่ดีที่สุด ดังนั้นผู้ทดลองควรลองค่าที่เป็นไปได้ทั้งหมดเพื่อทราบค่าที่เหมาะสมที่สุด

การทำ Hyperparameter Tuning สามารถทำด้วยการปรับค่าและเปรียบเทียบผลของตัวแบบทุกการผสมของพารามิเตอร์ หรือการปรับค่าด้วยตัวเองจนกว่าจะเจอชุดพารามิเตอร์ที่ส่งผลตามที่คาดไว้ วิธีนี้เรียกว่า Traditional Hyperparameter Tuning ซึ่งการปรับค่าด้วยตัวเองอาจใช้เวลาและทรัพยากรจำนวนมาก ปัจจุบันจึงมีฟังก์ชัน GridSearchCV ที่ช่วยในการวนซ้ำ Hyperparameter ที่กำหนดไว้ (ทินกร ม้าลายทอง, 2564)

GridSearchCV เป็นฟังก์ชันที่มาในแพ็คเกจ model_selection ของ Scikit-learn หรือเรียกว่าการค้นหาแบบกริด มีหลักการทำงานคือการลองใช้พารามิเตอร์ที่กำหนดไว้ล่วงหน้าทุกชุดและประเมินประสิทธิภาพของตัวแบบแต่ละชุด ซึ่งชุดพารามิเตอร์ที่มีประสิทธิภาพดีที่สุดจะถือว่าดีที่สุด นอกจากนี้ยังสามารถเพิ่มการตรวจสอบแบบไขว้สำหรับการสร้างตัวแบบในฟังก์ชันนี้ได้ด้วย โดยฟังก์ชันนี้สามารถเพิ่มประสิทธิภาพของตัวแบบได้ดี แต่หากมีชุดพารามิเตอร์ที่มากเกินไปอาจใช้เวลาในการทำงานที่ค่อนข้างนานเมื่อเทียบกับฟังก์ชันอื่นอย่าง RandomizedSearchCV ที่อาจจะให้ประสิทธิภาพดีน้อยกว่าแต่ก็ใช้เวลาที่น้อยกว่าเช่นกัน ซึ่งฟังก์ชัน GridSearchCV มีส่วนประกอบดังนี้

1. estimator: ตัวอย่างตัวแบบที่ต้องการตรวจสอบ Hyperparameter
2. params_grid: ชุด Hyperparameter ที่ต้องการทดลอง
3. scoring: การประเมินที่ต้องการใช้วัดประสิทธิภาพตัวแบบ
4. cv: จำนวนของการตรวจสอบแบบไขว้ที่ต้องลองสำหรับ Hyperparameter แต่ละชุด
5. verbose: สามารถตั้งค่าเป็น 1 เพื่อพิมพ์รายละเอียดขณะที่กำลังใช้งานฟังก์ชัน
6. n_jobs: จำนวนของกระบวนการที่ต้องการทำงานพร้อมกันสำหรับงานนี้ หากเป็น -1 ก็จะใช้เครื่องประมวลผลที่มีทั้งหมด (Team, 2023)

2.8 การวิเคราะห์ความแปรปรวนแบบ 2 ตัวประกอบ (Two – way analysis of variance)

การวิเคราะห์ความแปรปรวน (ANOVA) เป็นการทดสอบทางสถิติที่ใช้ในการวิเคราะห์ความแตกต่างระหว่างค่าเฉลี่ยของกลุ่มที่มากว่าสองกลุ่มขึ้นไป หากมีตัวแปรอิสระ 2 ตัวแปรหลายระดับและต้องการทราบว่าทั้งสองส่งผลต่อตัวแปรตามอย่างไรควรใช้การวิเคราะห์ความแปรปรวนแบบ 2 ตัวประกอบ แต่หากตัวแปรอิสระหนึ่งมีลักษณะเชิงคุณภาพ ส่วนอีกตัวแปรมีลักษณะเชิงปริมาณให้ใช้ ANCOVA แทน โดยการวิเคราะห์ความแปรปรวนจะใช้ F-test เพื่อดูนัยสำคัญซึ่งจะเปรียบเทียบค่า

ความแปรปรวนในแต่ละกลุ่มกับค่าความแปรปรวนรวม หากความแปรปรวนภายในกลุ่มน้อยกว่าความแปรปรวนระหว่างกลุ่มจะส่งผลให้ค่า F สูง โดยการวิเคราะห์ความแปรปรวนแบบ 2 ตัวประกอบจะทดสอบ 3 สมมติฐานหลัก (H_0) ในเวลาเดียวกัน ประกอบด้วย

1. ไม่มีความแตกต่างในค่าเฉลี่ยของกลุ่มที่ระดับใดๆ ของตัวแปรอิสระตัวแรก
2. ไม่มีความแตกต่างในค่าเฉลี่ยของกลุ่มที่ระดับใดๆ ของตัวแปรอิสระตัวที่สอง
3. ผลของตัวแปรอิสระตัวหนึ่งไม่ได้ขึ้นอยู่กับผลของตัวแปรอิสระอีกตัวหนึ่ง

เพื่อให้ผลลัพธ์ของการวิเคราะห์ความแปรปรวนแบบ 2 ตัวประกอบมีความน่าเชื่อถือ ข้อมูลที่จะนำมาวิเคราะห์ควรผ่านการตรวจสอบสมมติฐานเบื้องต้นซึ่งประกอบด้วย 6 สมมติฐานดังนี้

1. ตัวแปรตามที่วัดควรเป็นตัวแปรต่อเนื่อง กล่าวคือมีลักษณะเป็นช่วงเวลาหรืออัตราส่วน สามารถแสดงความมากหรือน้อยได้ สามารถคำนวณทางคณิตศาสตร์ได้ ตัวอย่างเช่น เวลาวัดเป็นชั่วโมง ความฉลาดวัดเป็นคะแนน IQ หรือน้ำหนักวัดเป็นกิโลกรัม เป็นต้น
2. ตัวแปรอิสระ 2 ตัวควรประกอบด้วยกลุ่ม 2 กลุ่มหรือมากกว่า ตัวอย่างเช่น ตัวแปรเพศอาจประกอบด้วยเพศชายและเพศหญิง ตัวแปรอาชีพอาจประกอบด้วย ศัลยแพทย์ แพทย์ พยาบาล ทันตแพทย์ และนักบำบัด เป็นต้น
3. ไม่มีความสัมพันธ์ระหว่างการสังเกตในแต่ละกลุ่มหรือระหว่างกลุ่มด้วยกันเอง ตัวอย่างเช่น ต้องมีผู้เข้าร่วมที่แตกต่างกันในแต่ละกลุ่ม โดยไม่มีผู้เข้าร่วมมากกว่าหนึ่งกลุ่ม
4. ไม่ควรมีค่าผิดปกติ (outliers) เนื่องจากส่งผลให้ความแม่นยำของการวิเคราะห์ความแปรปรวนลดลง ตัวอย่างเช่น ในการศึกษาคะแนน IQ ของนักเรียน 100 คน โดยที่คะแนนเฉลี่ยคือ 108 และมีความแตกต่างเพียงเล็กน้อยระหว่างนักเรียน แต่มีนักเรียนคนหนึ่งมีคะแนน 156 ซึ่งเรียกว่าเป็นค่าที่ผิดปกติ
5. ตัวแปรตามควรกระจายตัวแบบปกติ แต่หากละเมิดสมมติฐานนี้เล็กน้อยผลการวิเคราะห์ยังคงให้ผลลัพธ์ที่ถูกต้อง
6. ความแปรปรวนแต่ละกลุ่มเท่ากัน หรือเรียกว่าความเป็นเอกพันธ์ของความแปรปรวน (Homogeneity of Variance) โดยสามารถทดสอบผ่าน Levene's test แต่หากมีการละเมิดสมมติฐานนี้ ผลลัพธ์จะยังคงถูกต้องเมื่อกลุ่มตัวอย่างแต่ละกลุ่มเท่ากัน หรือกลุ่มตัวอย่างแต่ละกลุ่มไม่เท่ากันแต่แปรปรวนของกลุ่มสูงสุดต่อต่ำสุดไม่เกิน 10 ต่อ 1(Lund; Pornprasertmanit, 2015)

ตัวอย่างข้อมูลที่จะนำมาวิเคราะห์ความแปรปรวนแบบ 2 ตัวประกอบมีรายละเอียดดังรูปที่ 2.17 เป็นการทดลองปลูกต้นไม้จำนวนทั้งหมด 40 ต้นโดยมีตัวแปรอิสระคือ Sunlight Exposure

และ Watering Frequency ซึ่งแต่ละตัวแปรประกอบด้วย 4 และ 2 ระดับตามลำดับโดยทดลองซ้ำทั้งหมด 5 ครั้ง ดังกรอบสีแดงแสดงความสูงของต้นไม้ 5 ต้นภายใต้เงื่อนไข Sunlight Exposure ระดับ None และเงื่อนไข Watering Frequency ระดับ Daily

	Sunlight Exposure			
Watering Frequency	None	Low	Medium	High
Daily	4.8	5	6.4	6.3
	4.4	5.2	6.2	6.4
	3.2	5.6	4.7	5.6
	3.9	4.3	5.5	4.8
	4.4	4.8	5.8	5.8
Weekly	4.4	4.9	5.8	6
	4.2	5.3	6.2	4.9
	3.8	5.7	6.3	4.6
	3.7	5.4	6.5	5.6
	3.9	4.8	5.5	5.5

รูปที่ 2.17 ตัวอย่างข้อมูลสำหรับ Two way ANOVA (Bevans, 2022)

การวิเคราะห์ความแปรปรวนแบบ 2 ตัวประกอบนั้นสามารถทำได้ในหลายซอฟต์แวร์เช่น R, Python, SPSS, Stata รวมถึง Excel โดยรูปที่ 2.18 แสดงผลของการวิเคราะห์ด้วย Excel ได้ผลว่าค่า p-value ของตัวแปรอิสระ Sunlight Exposure มีนัยสำคัญที่ระดับ α เท่ากับ 0.05 แต่ตัวแปรอิสระ Watering Frequency และ interaction ไม่มีนัยสำคัญที่ระดับ α เท่ากับ 0.05 หมายความว่ามีเพียงตัวแปร Sunlight Exposure ที่ส่งผลต่อต้นไม้ ส่วน interaction หมายถึงผลของ Sunlight Exposure จะส่งผลสม่ำเสมอในแต่ละระดับของ Watering Frequency (Bevans, 2022)

G	H	I	J	K	L	M
SUMMARY	None	Low	Medium	High	Total	
Daily						
Count	5	5	5	5	20	
Sum	20.7	24.9	28.6	28.9	103.1	
Average	4.14	4.98	5.72	5.78	5.155	
Variance	0.378	0.232	0.447	0.412	0.775237	
Weekly						
Count	5	5	5	5	20	
Sum	20	26.1	30.3	26.6	103	
Average	4	5.22	6.06	5.32	5.15	
Variance	0.085	0.137	0.163	0.317	0.722632	
Total						
Count	10	10	10	10		
Sum	40.7	51	58.9	55.5		
Average	4.07	5.1	5.89	5.55		
Variance	0.211222	0.18	0.303222	0.382778		
ANOVA						
Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Sample (Watering)	0.00025	1	0.00025	0.000921	0.975975	4.149097
Columns (Sunlight)	18.76475	3	6.254917	23.04898	3.9E-08	2.90112
Interaction	1.01075	3	0.336917	1.241517	0.310898	2.90112
Within	8.684	32	0.271375			
Total	28.45975	39				

รูปที่ 2.18 ตัวอย่างผล Two way ANOVA (Bevans, 2022)

2.9 การเปรียบเทียบหลังการทดสอบรวม

การเปรียบเทียบหลังการทดสอบรวม (Posteriori Comparison) หรือ การเปรียบเทียบภายหลัง (Post hoc Comparison) เป็นการทดสอบหลังจากที่ผลของการวิเคราะห์ความแปรปรวนบ่งชี้ว่าค่าเฉลี่ยของกลุ่มไม่เท่ากันทั้งหมด แต่การวิเคราะห์ความแปรปรวนนั้นไม่ได้ระบุความแตกต่างต่างระหว่างคู่ ดังนั้นจึงใช้การเปรียบเทียบหลังการทดสอบรวมเพื่อระบุความแตกต่างรายคู่ของกลุ่มตัวอย่างอย่างน้อย 3 กลุ่มขึ้นไป (สุพัฒน์ สุกมลสันต์, 2560)

การเปรียบเทียบหลังการทดสอบรวมมีหลายวิธีขึ้นอยู่กับแต่ละชุดข้อมูล หรือผลลัพธ์ที่ต้องการ โดยในกรณีที่มีการวิเคราะห์ความแปรปรวนแบบ 2 ตัวประกอบควรเลือกวิธีที่สามารถควบคุมอัตราการคลาดเคลื่อนต่อการทดลอง (Experiment – wise Error Rate หรือ α_{EW})

กล่าวคือควรเลือกวิธีที่สามารถความคุมให้ α_{EW} = α = 0.05 โดย α_{EW} สามารถคำนวณได้ดังสมการที่ (18)

$$\alpha_{EW} = \frac{\alpha * n_{sig}}{n_{Exp}} \quad (18)$$

โดยที่ α_{EW} คืออัตราความคลาดเคลื่อนต่อการทดลอง α คือระดับนัยยะสำคัญที่ผู้วิจัยกำหนด n_{sig} คือจำนวนการทดลองที่มีการเปรียบเทียบแล้วพบว่าค่าเฉลี่ยของกลุ่มตัวอย่างแตกต่างกัน n_{Exp} คือจำนวนการทดลองทั้งหมด

นอกจากนี้ก่อนการเปรียบเทียบหลังการทดสอบรวมควรผ่านการตรวจสอบสมมติฐานเบื้องต้น เพื่อให้ทราบลักษณะของข้อมูลและเลือกใช้การทดสอบได้ถูกต้อง โดยสมมติฐานที่ใช้มีลักษณะเดียวกันกับการวิเคราะห์ความแปรปรวนแบบ 2 ตัวประกอบ และวิธีนี้ไม่คงทนต่อการละเมิดการกระจายตัวแบบปกติ

การทดสอบ Games-Howell เป็นหนึ่งในการเปรียบเทียบหลังการทดสอบรวมแบบไม่ใช้พารามิเตอร์สำหรับทำการเปรียบเทียบหลายรายการสำหรับกลุ่มตัวอย่างตั้งแต่สองกลุ่มขึ้นไป ซึ่งมีลักษณะคล้ายกับการทดสอบของ Tukey แตกต่างกับตรงการทดสอบ Games-Howell สามารถใช้ได้เมื่อความแปรปรวนแต่ละกลุ่มไม่เท่ากัน หรือเท่ากันก็ได้ ขนาดของกลุ่มตัวอย่างเท่ากัน หรือแตกต่างกันก็ได้ แต่หากขนาดตัวอย่างในแต่ละกลุ่มน้อยกว่าหรือเท่ากับ 5 จะส่งผลให้ $\alpha_{EW} > \alpha$ ที่กำหนด นอกจากนี้การทดสอบ Games-Howell และการทดสอบของ Tukey มักจะรายงานผลลัพธ์ที่คล้ายคลึงกันเมื่อข้อมูลมีความแปรปรวนเท่ากันและมีขนาดตัวอย่างเท่ากัน (Schlegel, 2020; สุพัฒน์ สุกมลสันต์, 2560)

โดยการทดสอบ Games-Howell กำหนดดังสมการที่ 19

$$\bar{x}_i - \bar{x}_j > q_{\sigma,k,df} \quad (19)$$

โดยที่ σ เท่ากับความคลาดเคลื่อนมาตรฐาน สามารถคำนวณดังสมการที่ 20

$$\sigma = \sqrt{\frac{1}{2}\left(\frac{s_i^2}{n_i} + \frac{s_j^2}{n_j}\right)} \quad (20)$$

ระดับแห่งความเป็นอิสระ (Degrees of Freedom) คำนวณด้วยวิธี Welch ดังสมการที่ 21

$$\frac{\left(\frac{s_i^2}{n_i}+\frac{s_j^2}{n_j}\right)^2}{\frac{\left(\frac{s_i^2}{n_i}\right)^2}{n_i-1}+\frac{\left(\frac{s_j^2}{n_j}\right)^2}{n_j-1}} \tag{21}$$

ค่า t-value คำนวณจากการทดสอบ t-test ของ Welch ดังสมการที่ 22

$$t=\frac{\bar{x}_i-\bar{x}_j}{\sqrt{\frac{s_i^2}{n_i}+\frac{s_j^2}{n_j}}} \tag{22}$$

ดังนั้น ช่วงความเชื่อมั่นสามารถคำนวณดังสมการที่ 23

$$\bar{x}_i-\bar{x}_j \pm t\sqrt{\frac{1}{2}\left(\frac{s_i^2}{n_i}+\frac{s_j^2}{n_j}\right)} \tag{23}$$

สุดท้าย ค่า p-values จะถูกคำนวณโดยใช้ช่วงของ Tukey ดังสมการที่ 24

$$q_{t^*\sqrt{2},k,df} \tag{24}$$

2.10 การทบทวนวรรณกรรมที่เกี่ยวข้อง

ปัญหาข้อมูลไม่สมดุล เป็นปัญหาที่พบได้กับข้อมูลในโลกของความเป็นจริงเช่น การตรวจจับมลภาวะ การจัดการความเสี่ยง การตรวจจับการฉ้อโกงและการวินิจฉัยทางการแพทย์ ข้อมูลที่ไม่สมดุลคือจำนวนข้อมูลที่อยู่ในแต่ละกลุ่มที่สนใจมีอัตราส่วนที่ต่างกัน เช่นอัตราส่วนที่วินิจฉัยว่าเป็นมะเร็งหรือข้อมูลกลุ่มน้อย (Minority class) กับไม่เป็นมะเร็งหรือข้อมูลกลุ่มมาก (Majority class) อัตราส่วนที่พบในการเรียนรู้แบบมีผู้สอน (Supervised Learning) อาจเป็น 1:100, 1:1,000 และ 1:10,000 ซึ่งการเรียนรู้ของเครื่องทั่วไปให้ผลความแม่นยำที่ดีแต่เป็นผลการพยากรณ์เฉพาะข้อมูลกลุ่มมาก เนื่องจากมองข้อมูลกลุ่มน้อยเป็นข้อมูลที่ผิดปกติจึงถูกมองข้ามในการเรียนรู้และพยากรณ์

ไม่ได้ ส่งผลต่อนโยบายสินค้าคงค้าง และส่งผลเสียต่อสินค้าคงคลัง (de Santis et al., 2017; Hajek & Abedin, 2020)

งานวิจัยของ de Santis et al. (2017) ได้จัดการกับปัญหาดังกล่าวด้วยการปรับข้อมูลด้วยวิธีสุ่ม (Sampling) เพื่อสร้างความสมดุลของแต่ละกลุ่ม เนื่องจากทำให้ประสิทธิภาพของอัลกอริทึมในการเรียนรู้ดีขึ้น โดยวิธีที่ใช้ปรับข้อมูลได้แก่ Random Under-Sampling (RUS) และ The Synthetic Minority Over-sampling Technique (SMOTE) ซึ่งใช้ร่วมกับอัลกอริทึม Classification Tree นอกจากนี้ยังใช้วิธี Ensemble Learning ได้แก่ Random Forest (FOREST) ,Gradient Tree Boosting (GBOOST) และ Blagging (BLAG) เพื่อจัดกลุ่มด้วยข้อมูลที่ไม่ได้ปรับสมดุล ส่วนอัลกอริทึมพื้นฐานที่ใช้สำหรับเปรียบเทียบคือ Logistic Regression (LOGIST) เนื่องจากเป็นอัลกอริทึมที่ได้รับความนิยมและง่ายกับการจัดกลุ่มที่มี 2 กลุ่ม โดยตัวแบบทั้งหมดมีการใช้วิธีการตรวจสอบแบบไขว้แบบ 5 กลุ่ม (Stratified 5-Fold Cross-Validation) เพื่อหลีกเลี่ยงการเกิด Overfitting ส่วนของการวัดประสิทธิภาพตัวแบบนั้นใช้ Area Under the ROC Curve (AUROC) เนื่องจากวิธีนี้ระบุได้ว่าตัวใดที่เป็นตัวรบกวน และยังวัดความสามารถของการจำแนกข้อมูลได้ จากงานวิจัยได้ผลว่าตัวแบบที่ใช้อัลกอริทึม CART ที่มีการปรับข้อมูลให้สมดุลมีผลลัพธ์ดีกว่าข้อมูลที่ไม่มีการปรับสมดุล โดยการปรับเพิ่มข้อมูลด้วยวิธี SMOTE ให้ผลลัพธ์ที่ดีกว่าการปรับลดข้อมูลด้วยวิธี RUS ส่วนตัวแบบที่ไม่ได้มีการปรับสมดุลพบว่าวิธี GBOOST ให้ผลลัพธ์ที่ดีที่สุดซึ่งวิธี BLAG ให้ผลลัพธ์ดีประมาณกัน (de Santis et al., 2017)

นอกจากการสร้างตัวแบบในการพยากรณ์สินค้าคงค้าง และวัดประสิทธิภาพของตัวแบบด้วยความแม่นยำแล้ว งานวิจัยของ Hajek and Abedin (2020) มีแนวคิดในการนำความแม่นยำของการพยากรณ์มาแปลงเป็นผลกำไรดังสมการที่ (25) โดยประยุกต์สมการเข้ากับตัวแบบเพื่อให้ตัวแบบเรียนรู้และหาตัวแบบที่ได้ผลกำไรจากการพยากรณ์มากที่สุด

$$\pi = b\left(\frac{TN}{TN+FP}\right) - c\left(\frac{FN}{FN+TP}\right) \quad (25)$$

โดยที่ "π" คือผลกำไร TP (True Positive) คือจำนวนข้อมูลที่ทำนายถูกต้องในกลุ่มมาก TN (True Negative) คือจำนวนข้อมูลที่ทำนายถูกต้องในกลุ่มน้อย FP (False Positive) คือจำนวนข้อมูลที่ทำนายผิดว่าอยู่ในกลุ่มมาก FN (False Negative) คือจำนวนข้อมูลที่ทำนายผิดว่าอยู่ในกลุ่มน้อย "b" คืออัตรากำไรจากการขายและ "c" คือค่าใช้จ่ายคงคลัง

เทคนิคที่ใช้สำหรับการปรับสมดุลจะเป็นการดัดแปลง Clustering-Based Under-Sampling (CBUS) และใช้กับอัลกอริทึมพื้นฐานของการจัดกลุ่มอย่าง LOGIST และ k-Nearest

Neighbor (kNN) นอกจากนี้ยังใช้กับอัลกอริทึมที่มีการศึกษาไม่นานมานี้อย่าง C4.5 Decision Tree, FOREST, Support Vector Machine (SVM) และ Multi-Layer Neural Network (NN) นอกจากวิธีสุ่ม CBUS แล้วยังใช้วิธีสุ่มอื่นเพื่อปรับสมดุลของข้อมูลได้แก่ RUS และ SMOTE สำหรับอัลกอริทึมทั้งหมด และใช้เทคนิค EasyEnsemble สำหรับ C4.5 Decision Tree นอกจากการวัดประสิทธิภาพด้วยผลกำไรกับอัลกอริทึมที่กล่าวมาข้างต้นเทียบกันแล้ว ยังวัดกับอัลกอริทึม eXtreme Gradient Boosting (XGBoost) ด้วย นอกจากนี้ยังสนใจเรื่องเวลาในการเรียนรู้ของชุดข้อมูลสอนของแต่ละตัวแบบการพยากรณ์อีกด้วย ซึ่งจากงานวิจัยได้ผลว่า มีเพียง EasyEnsemble และ XGBoost เท่านั้นที่ทำงานได้ดีในแง่ของการวัดผลกำไร โดยตัวแบบที่ใช้วิธีสุ่ม CBUS สำหรับอัลกอริทึม FOREST ได้ผลกำไรประมาณร้อยละ 4 ซึ่งเป็นผลที่ดีที่สุด และยังได้ค่า AUROC ดีที่สุดอีกด้วย แต่สำหรับตัวแบบอื่นที่ให้ค่า AUROC ดีนั้นไม่สามารถวัดผลผ่านตัวชี้วัดกำไรได้ ส่วนในด้านของเวลาพบว่า CBUS ที่วัดประสิทธิภาพด้วยผลกำไรใช้เวลาเรียนรู้เร็วที่สุด (Hajek & Abedin, 2020)

นอกจากการพยากรณ์สินค้าคงค้างด้วยอัลกอริทึมและวิธีวัดประสิทธิภาพของตัวแบบของงานวิจัยที่กล่าวมาข้างต้นแล้ว งานวิจัยของ Malviya et al. (2021) ยังใช้ C5.0 Decision Tree, Bayesian network และ Discriminant analysis พยากรณ์และวัดประสิทธิภาพเปรียบเทียบกับอัลกอริทึมของงานวิจัยก่อนหน้าอย่าง FOREST , SVM, NN และ LOGIST ด้วยวิธีการวัดประสิทธิภาพของตัวแบบที่หลากหลาย ได้แก่ Precision, Recall, F Measure, AUROC, Gini Coefficient และ Accuracy โดยในงานวิจัยนี้ไม่ได้กล่าวถึงการทำให้ข้อมูลสมดุลด้วยวิธีสุ่ม หรือการหลีกเลี่ยงการเกิด Overfitting ด้วยวิธีการตรวจสอบแบบไขว้ จากงานวิจัยได้ผลว่าทุกอัลกอริทึมให้ผลของ Accuracy ดีมาก แต่ผลของ AUROC จะได้ผลดีในกลุ่ม Ensemble Learning แต่กลับกันหากวัดด้วย F Measure ตัวแบบที่ให้ผลลัพธ์ดีคือ C5.0 , SVM และ LOGIST (Malviya et al., 2021)

งานวิจัยของ Islam and Amin (2020) เป็นงานวิจัยล่าสุดที่พบเกี่ยวกับการพยากรณ์สินค้าคงค้างด้วยการใช้การเรียนรู้ของเครื่อง โดยมีวัตถุประสงค์ต่างจากวิจัยที่กล่าวมาก่อนหน้าด้วยการสร้างแนวทางการตัดสินใจว่าสินค้านี้ควรเป็นสินค้าคงค้างหรือไม่ เพื่อที่จะสามารถช่วยตัดสินใจกับข้อมูลขณะที่ไม่มีเครื่องคอมพิวเตอร์ได้ โดยงานวิจัยนี้จะไม่ใช้คุณลักษณะของข้อมูล (Attributes) ที่มีทั้งหมดแต่จะเลือกเฉพาะตัวที่มีผลกับการพยากรณ์มาก โดยในแต่ละคุณลักษณะจะมีการแบ่งช่วงเพื่อเรียนรู้ โดยอัลกอริทึมที่ใช้เรียนรู้นั้นเป็นเทคนิค Ensemble Learningได้แก่ Distributed Random Forest (DRF) และ Gradient Boosting Machine (GBM) ทั้งสองอัลกอริทึมมีการใช้วิธีสุ่ม ทั้ง RUS และ SMOTE และวัดประสิทธิภาพด้วย LogLoss , AUROC และ Mean per class error จากงานวิจัยได้ผลว่าการแบ่งช่วงของคุณลักษณะให้ประสิทธิภาพที่ดีขึ้นร้อยละ 20 และ DRF ให้ผลดีกว่า GBM (Islam & Amin, 2020)

จากงานวิจัยที่กล่าวมาข้างต้นซึ่งจัดการกับปัญหาความไม่สมดุลของข้อมูลด้วยการปรับระดับข้อมูลทั้งหมด ทว่ายังมีวิธีอื่นที่สามารถแก้ไขปัญหาได้โดยไม่ได้ลดข้อมูลที่อาจเป็นประโยชน์หรือสร้างข้อมูลใหม่ที่อาจซ้ำกับข้อมูลเดิมในกลุ่มอื่นนั้นคือวิธี Thershold Moving ซึ่งเป็นวิธีการหลังการฝึกข้อมูลที่ให้ประสิทธิภาพดี และยังทำได้ง่าย โดยในงานวิจัยของ Yu et al. (2015) ศึกษาการจัดการกับข้อมูลที่ไม่สมดุลภายใต้อัลกอริทึม Support Vector Machine (SVM) ได้แก่ Random oversampling (ROS), Random undersampling (RUS), SMOTE, CS-SVM/z-SVM (การให้น้ำหนักแต่ละกลุ่ม) และ optimized SVM decision threshold adjustment strategy (SVM-OTHR) นอกจากนี้ยังใช้วิธี Bagging ensemble learning ร่วมกับ threshold adjustment (EnSVM-OTHR) โดยชุดข้อมูลที่ทดลอง Keel 30 ชุดข้อมูลนั้นวัดประสิทธิภาพด้วย F-measure และ G-mean ได้ผลว่าสำหรับข้อมูลที่มีอัตราส่วนความไม่สมดุลสูง การปรับเพิ่มข้อมูลจะให้ผลลัพธ์ที่ดีกว่าการปรับลดข้อมูล เนื่องจากการปรับลดข้อมูลจะยิ่งสูญเสียข้อมูลเมื่อมีความไม่สมดุลสูง ดังนั้นวิธีนี้จึงเหมาะกับข้อมูลที่ไม่สมดุลต่ำ ขณะเดียวกันในข้อมูลที่ไม่สมดุลต่ำการปรับเพิ่มข้อมูลอาจทำให้เกิดการ Overfitting นอกจากนี้ SVM-OTHR ยังให้ผล F-measure ดีที่สุดถึง 12 ชุดข้อมูล และ 11 ชุดข้อมูลสำหรับการวัดผลด้วย G-mean และเมื่อพิจารณาวิธี EnSVM-OTHR ร่วมด้วยพบว่าประสิทธิภาพดีขึ้น (Yu et al., 2015)

นอกจากมีการใช้ Thershold Moving กับอัลกอริทึม SVM แล้วงานวิจัย Collell et al. (2018) ได้ศึกษาวิธี Probability Threshold Bagging (PT-bagging) กับอัลกอริทึม Decision Tree (DT) และ NN เนื่องจากอัลกอริทึมดังกล่าวไม่เสถียรและเหมาะกับการทำ Bagging โดยผู้วิจัยได้ศึกษาวิธี PT-bagging เทียบกับ Exactly Balancing (EB), Roughly Balancing (RB), SMOTE และ Random Balance (RNB) บนชุดข้อมูลHDDT 14 ชุดข้อมูล, Keel 19 ชุดข้อมูลและ UCI 3 ชุดข้อมูล ทำการฝึกชุดข้อมูล 5 ครั้งและใช้การตรวจสอบแบบไขว้ 2 กลุ่ม จากนั้นวัดประสิทธิภาพด้วย macro-accuracy, macro F1 score และ area under the PR curve (AUCPR) นอกจากนี้ยังใช้วิธีทางสถิติ Friedman test เพื่อทดสอบความแตกต่างระหว่างวิธีการ posthoc Nemenyi test เพื่อระบุความแตกต่างที่เป็นคู่ และ paired Wilcoxon rank sums test เพื่อเปรียบเทียบ 2 วิธีการโดยตรง โดยผลของงานวิจัยนี้คือการปรับระดับข้อมูลแต่ละวิธีจะเหมาะกับวิธีการวัดประสิทธิภาพที่ต่างกัน ซึ่งการปรับลดระดับข้อมูลมีประสิทธิภาพดีกว่าการปรับเพิ่มข้อมูล ในส่วนของการวัดประสิทธิภาพด้วย macro F1 score วิธี PT-bagging ให้ผลลัพธ์ดีกว่าวิธี SMOTE ยิ่งไปกว่านั้นยังให้ค่า Recall ที่ไม่แตกต่างกันระหว่างข้อมูลกลุ่มมาก และ ข้อมูลกลุ่มน้อย (Collell et al., 2018)

ตารางที่ 4 สรุปวิจัยที่เกี่ยวข้องกับการพยากรณ์ความเสี่ยงในการเกิดสินค้าคงค้าง

ผู้วิจัย	de Santis et al. (2017)	Hajek and Abedin (2020)	Malviya et al. (2021)	Islam and Amin (2020)
การจัดการข้อมูลไม่สมดุล	SMOTE, RUS	SMOTE, RUS, CBUS	-	SMOTE, RUS
อัลกอริทึม	FOREST, GBOOST, Blagging, LOGIST	LOGIST, kNN, C4.5 Decision Tree, SVM, NN, EasyEnsemble, XGBoost	FOREST, SVM, NN, LOGIST, C5.0, Decision Tree, Bayesian network, Discriminant analysis	Distributed Random Forest (DRF), Gradient Boosting Machine (GBM)
การวัดประสิทธิ-ภาพ	AUROC	AUROC, Expected profit [%], Training times	Precision, Recall, F Measure, AUROC, Gini Coefficient, Accuracy	precision, recalls, F-measure, AUROC, Misclassification error, Specificity, accuracy

ตารางที่ 5 สรุปวิจัยที่เกี่ยวข้องกับ Threshold Moving

ผู้วิจัย	H. Yu et al. (2015)	G. Collell et al. (2018)
การจัดการข้อมูลไม่สมดุล	Thershold Moving, ROS, RUS, SMOTE, CS-SVM/z-SVM	PT-bagging,Exactly Balancing (EB), Roughly Balancing (RB), SMOTE , Random Balance (RNB)
อัลกอริทึม	SVM	Decision Tree, NN
การวัดประสิทธิภาพ	F-measure, G-mean	macro-accuracy, macro F1 score, AUCPR

จากการทบทวนวรรณกรรมได้ข้อสรุปว่า วิจัยที่เกี่ยวข้องกับการพยากรณ์ความเสี่ยงในการเกิดสินค้าคงค้างสรุปดังตารางที่ 4 ว่างานวิจัยนิยมใช้วิธีปรับระดับข้อมูลในการจัดการปัญหาข้อมูลที่

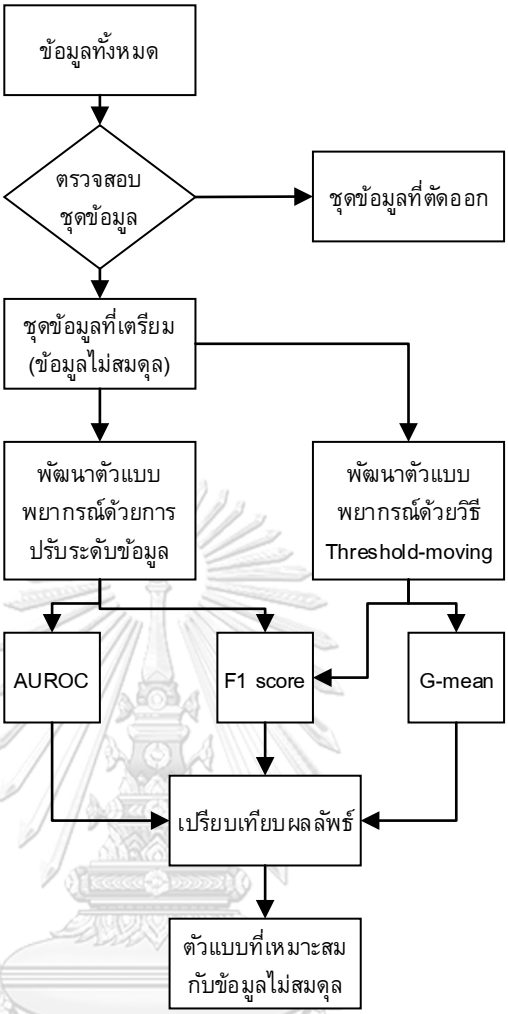
ไม่สมดุลเนื่องจากทำให้ประสิทธิภาพของตัวแบบดีขึ้น และมักใช้การตรวจสอบแบบไขว้เพื่อเลี่ยงการเกิด Overfitting โดยกลุ่มอัลกอริทึมที่มีประสิทธิภาพที่ดีมักจะอยู่ในกลุ่ม Ensemble Learning ส่วนการวัดประสิทธิภาพที่นิยมใช้ได้แก่ AUROC ทว่าการปรับระดับข้อมูลนั้นก็มีข้อเสียที่อาจทำให้สูญเสียข้อมูลที่สำคัญ หรือ อาจมีการสร้างกลุ่มข้อมูลที่ซ้ำกันซึ่งวิธีการนี้อาจไม่ได้ผลกับข้อมูลจริงเพราะเป็นการเปลี่ยนแปลงข้อมูลก่อนการฝึกข้อมูล ในทางตรงกันข้ามวิธี Threshold Moving สรุปดังตารางที่ 5 เป็นวิธีการหลังจากการฝึกข้อมูลที่สามารถแก้ปัญหาข้อมูลที่ไม่สมดุลได้ ทั้งยังให้ประสิทธิภาพดีกว่าการปรับระดับข้อมูลในบางงานวิจัย ดังนั้นวิทยานิพนธ์ฉบับนี้จึงศึกษาการแก้ปัญหาข้อมูลไม่สมดุลด้วยวิธีการปรับระดับข้อมูลที่ได้รับความนิยมสำหรับการพยากรณ์สินค้าคงค้าง และวิธีการ Threshold Moving ที่จากการศึกษาวิจัยก่อนหน้ายังไม่ได้มีการประยุกต์ใช้กับข้อมูลสินค้าคงค้าง โดยในวิธีการปรับระดับข้อมูลประกอบด้วยวิธี Near Miss Undersampling-3 (NearMiss-3) ซึ่งเป็นการปรับลดจำนวนข้อมูลด้วยการเลือกเก็บข้อมูลจนจำนวนข้อมูลของทั้งสองกลุ่มเท่ากัน วิธีสุ่ม One-Sided Selection for Undersampling (OSS) เป็นการเลือกเก็บข้อมูล ผสมกับเลือกทิ้งข้อมูล โดยวิธีนี้จำนวนข้อมูลที่ปรับลดของข้อมูลกลุ่มมากอาจจะไม่เท่ากับข้อมูลกลุ่มน้อย แต่ก็ทำให้ข้อมูลมีความสมดุลมากขึ้น และวิธีสุดท้ายที่ได้รับความนิยมในงานวิจัยอย่าง SMOTE นอกจากนี้ยังใช้วิธีการปรับเพิ่มด้วยวิธี SMOTE ผสมกับปรับลดข้อมูลด้วยวิธี OSS ซึ่งในวิจัยที่กล่าวมาก่อนหน้ายังไม่ได้มีการศึกษาสำหรับการพยากรณ์สินค้าคงค้าง และใช้ AUROC ในการวัดประสิทธิภาพ ส่วนวิธี Threshold Moving มีการหาเกณฑ์ด้วย G-Mean บน AUROC และการปรับเกณฑ์ F1 score ภายในช่วง 0 ถึง 1 ส่วนอัลกอริทึมที่ใช้ในการพยากรณ์ได้แก่ LOGIST , FOREST และ XGBoost ซึ่งแต่ละอัลกอริทึมมีการปรับเปลี่ยนค่าพารามิเตอร์ภายในด้วย Grid Search เพื่อให้ได้ตัวแบบที่มีประสิทธิภาพมากที่สุด นอกจากนี้ยังหลีกเลี่ยงการเกิด Overfitting ด้วยการตรวจสอบแบบไขว้แบบ 5 กลุ่ม (Stratified 5-fold cross-validation)

บทที่ 3
วิธีดำเนินการวิจัย

ในการดำเนินงานจัดทำวิทยานิพนธ์เรื่องการพยากรณ์สินค้าคงค้างด้วยการเรียนรู้ของเครื่องสำหรับข้อมูลไม่สมดุล จากทฤษฎีและงานวิจัยที่เกี่ยวข้อง ทำให้ผู้วิจัยได้รับทราบถึงข้อมูลต่างๆ ที่เป็นประโยชน์ และสามารถนำมาประยุกต์ใช้ได้กับการจัดทำวิทยานิพนธ์ครั้งนี้ โดยหลังจากการศึกษานั้น ผู้วิจัยสามารถทำการแบ่งขั้นตอนการดำเนินงานวิจัย ออกเป็น 4 ส่วน ดังนี้

1. การตรวจสอบชุดข้อมูล
2. การเตรียมชุดข้อมูล
3. การพัฒนาตัวแบบการพยากรณ์
4. การเปรียบเทียบผลการดำเนินงาน

การดำเนินการของผู้วิจัยทั้ง 4 ขั้นตอน มีภาพรวมของการดำเนินการดังรูปที่ 3.1 ซึ่งขั้นตอนแรกคือการตรวจสอบชุดข้อมูลเพื่อดูลักษณะของข้อมูลทั้งด้านของคุณลักษณะที่ใช้พยากรณ์รวมถึงลักษณะของผลลัพธ์ จากนั้นจะทำการเตรียมข้อมูลเพื่อให้สามารถใช้กับการเรียนรู้ของเครื่องได้ดี ไม่ว่าจะเป็นการเปลี่ยนข้อมูลเชิงคุณภาพเป็นตัวเลข การตัดข้อมูลที่ผิดปกติ รวมถึงการตัดบางคุณลักษณะออกเป็นต้น เมื่อได้ข้อมูลที่เตรียมแล้วจะทำการพัฒนาตัวแบบการพยากรณ์ โดยจะศึกษาการจัดการกับข้อมูลไม่สมดุล 2 รูปแบบคือวิธีการปรับระดับข้อมูลและวิธี Threshold-moving โดยทั้ง 2 วิธีนี้จะสร้างตัวแบบด้วยอัลกอริทึม Random Forest, eXtreme Gradient Boosting และ Linear logistic regression ตามค่าพารามิเตอร์ที่ได้กำหนด โดยตัววัดประสิทธิภาพสำหรับวิธีการปรับระดับข้อมูลคือ AUROC และ F1 score ส่วนตัววัดประสิทธิภาพสำหรับวิธี Threshold-moving คือ F1 score และ G-mean เมื่อได้ผลลัพธ์จากทั้ง 3 ตัววัดประสิทธิภาพแล้ว จะทำการเปรียบเทียบตัวแบบโดยสำหรับ F1 score สามารถเปรียบเทียบผลของวิธีการปรับระดับข้อมูลและวิธี Threshold-moving ได้ ส่วนตัวแบบที่ดีที่สุดจะเปรียบเทียบจาก Confusion Matrix ในส่วนของความถูกต้องในการพยากรณ์ข้อมูลกลุ่มน้อย



รูปที่ 3.1 วิธีดำเนินการวิจัย

3.1 การตรวจสอบชุดข้อมูล

ในงานวิจัยฉบับนี้ได้ใช้ข้อมูลของบริษัทแห่งหนึ่งซึ่งสามารถเข้าถึงได้จาก ชุดข้อมูลของ Kaggle ในหัวข้อ Backorder Dataset Predict Backorder Product (ZINJAD, 2021) ซึ่งเป็นข้อมูลในอดีตที่ต้องการพยากรณ์ความเสี่ยงในการเกิดสินค้าคงค้างล่วงหน้า ประกอบด้วยคุณลักษณะที่ใช้พยากรณ์ทั้งหมด 22 คุณลักษณะ และผลลัพธ์ 1 คุณลักษณะ (X23) แบ่งข้อมูล 2 กลุ่มคือเป็นสินค้าคงค้างค้าง กับ ไม่เป็นสินค้าคงค้าง แสดงตัวอย่างข้อมูลดังตารางที่ 6 คุณลักษณะดังกล่าวมีดังนี้

X1 : รหัสสินค้า

X2 : จำนวนสินค้าคงคลัง

X3 : เวลาขนส่งจากต้นทางไปยังปลายทาง (วัน)

X4 : จำนวนสินค้าที่อยู่ระหว่างขนส่ง

X5-7 : คาดการณ์ยอดขายในอีก 3,6,9 เดือน

X8-11 : ปริมาณการขายในช่วงเวลา 1,3,6,9 เดือนที่ผ่านมา

X12 : จำนวนขั้นต่ำที่จัดเก็บ

X13 : จำนวนสินค้าที่ค้างจากแหล่งที่มา

X14-15 : ประสิทธิภาพของแหล่งที่มาในช่วง 6,12 เดือนที่ผ่านมา

X16 : จำนวนสินค้าที่ค้างส่ง

X17-22 : ลักษณะสินค้าทั่วไป

X23 : ความต้องการสินค้าคงค้าง

ตารางที่ 6 ตัวอย่างข้อมูล

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12
1111597	353	52	4	0	0	0	11	34	68	102	21
1111613	0	17	2	1	4	7	2	4	8	9	0
1111620	1	8	1	1	1	1	0	1	1	1	0
1111622	117	8	80	0	150	225	20	49	118	186	36
1111624	36	8	3	0	0	0	4	9	22	32	1
1111643	59	8	3	0	0	0	4	10	31	51	8
1111651	143	8	1	16	50	118	18	49	106	284	27
1111655	-55	8	2	428	526	606	25	100	205	387	52
1111659	30	8	14	42	84	126	7	56	107	148	24
1111670	-499	12	350	3452	5044	7188	747	2550	3572	4745	401

X1	X13	X14	X15	X16	X17	X18	X19	X20	X21	X22	X23
1111597	No	0	0.83	0.89	0	Yes	No	Yes	Yes	No	No
1111613	No	0	0.79	0.83	0	No	No	No	Yes	No	No
1111620	No	0	0.98	0.95	0	No	No	No	Yes	No	No
1111622	No	0	0.83	0.89	0	No	No	No	Yes	No	No
1111624	No	0	1	0.99	0	No	No	No	Yes	No	No
1111643	No	0	0.88	0.92	0	No	No	No	Yes	No	No
1111651	No	0	0.81	0.86	0	No	No	No	Yes	No	No
1111655	No	0	-99	-99	56	No	No	No	Yes	No	Yes
1111659	No	0	0.98	0.92	0	No	No	No	Yes	No	Yes
1111670	No	0	0.49	0.72	525	No	No	No	Yes	No	Yes

ตารางที่ 7 และ 8 แสดงลักษณะของคุณลักษณะที่เป็นข้อมูลเชิงปริมาณโดย สดมภ์จำนวน จะใช้ดูว่าคุณลักษณะมีข้อมูลที่หายไปหรือไม่ สดมภ์ความหลากหลายใช้ดูค่าในแต่ละคุณลักษณะว่ามีจำนวนซ้ำกันเท่าใด ส่วนสดมภ์ที่เหลือใช้ดูลักษณะข้อมูลอื่นๆ เช่น การกระจายตัวของข้อมูล ส่วนตารางที่ 9 แสดงลักษณะของคุณลักษณะที่เป็นข้อมูลเชิงคุณภาพ ซึ่งข้อมูลส่วนใหญ่คือลักษณะส่วนใหญ่ของคุณลักษณะนั้น และแสดงจำนวนในสดมภ์ถัดมา

ตารางที่ 7 ลักษณะข้อมูลเชิงปริมาณ 1

	จำนวน	ความหลากหลาย	mean	std	ความเบ้
รหัสสินค้า	1929935	1929935	2258963.19	765186.65	0.22
จำนวนสินค้าคงคลัง	1929935	15903	496.57	29573.43	340.22
เวลาขนส่ง	1814318	32	7.88	7.05	4.56
สินค้าที่อยู่ระหว่างขนส่ง	1929935	5543	43.06	1295.42	168.98
คาดการณ์ยอดขาย 3 เดือน	1929935	8293	178.54	5108.77	142.78
คาดการณ์ยอดขาย 6 เดือน	1929935	11788	345.47	9831.56	138.82
คาดการณ์ยอดขาย 9 เดือน	1929935	14523	506.61	14345.43	142.68
ปริมาณขาย 1 เดือน	1929935	6088	55.37	1884.38	193.72
ปริมาณขาย 3 เดือน	1929935	11149	174.66	5188.86	141.81
ปริมาณขาย 6 เดือน	1929935	15813	341.57	9585.03	138.93
ปริมาณขาย 9 เดือน	1929935	19581	523.58	14733.27	135.44
จำนวนขั้นต่ำที่จัดเก็บ	1929935	5909	52.78	1257.97	130.96
สินค้าที่ค้างจากแหล่งที่มา	1929935	874	2.02	229.61	414.27
ประสิทธิภาพแหล่งที่มา 6 เดือน	1929935	102	-6.90	26.60	-3.17
ประสิทธิภาพแหล่งที่มา 12 เดือน	1929935	102	-6.46	25.88	-3.30
จำนวนสินค้าที่ค้างส่ง	1929935	686	0.65	35.43	149.62

ตารางที่ 8 ลักษณะข้อมูลเชิงปริมาณ 2

	min	25%	50%	75%	max
รหัสสินค้า	1026827	1594002.5	2076486	3044510.5	3526994
จำนวนสินค้าคงคลัง	-27256	4	15	80	12334404
เวลาขนส่ง	0	4	8	9	52
สินค้าที่อยู่ระหว่างขนส่ง	0	0	0	0	489408
คาดการณ์ยอดขาย 3 เดือน	0	0	0	4	1510592
คาดการณ์ยอดขาย 6 เดือน	0	0	0	12	2461360
คาดการณ์ยอดขาย 9 เดือน	0	0	0	20	3777304
ปริมาณขาย 1 เดือน	0	0	0	4	741774
ปริมาณขาย 3 เดือน	0	0	1	15	1105478
ปริมาณขาย 6 เดือน	0	0	2	31	2146625
ปริมาณขาย 9 เดือน	0	0	4	47	3205172
จำนวนขั้นต่ำที่จัดเก็บ	0	0	0	3	313319
สินค้าที่ค้างจากแหล่งที่มา	0	0	0	0	146496
ประสิทธิภาพแหล่งที่มา 6 เดือน	-99	0.63	0.82	0.96	1
ประสิทธิภาพแหล่งที่มา 12 เดือน	-99	0.66	0.81	0.95	1
จำนวนสินค้าที่ค้างส่ง	0	0	0	0	12530

ตารางที่ 9 ลักษณะข้อมูลเชิงคุณภาพ

ลักษณะสินค้าทั่วไป	จำนวน	ความหลากหลาย	ข้อมูลส่วนใหญ่	จำนวนข้อมูลส่วนใหญ่
potential_issue (X17)	1929935	2	No	1928946
deck_risk (X18)	1929935	2	No	1494482
oe_constraint (X19)	1929935	2	No	1929643
ppap_risk (X20)	1929935	2	No	1697383
stop_auto_buy (X21)	1929935	2	Yes	1859391
rev_stop (X22)	1929935	2	No	1929096
ความต้องการสินค้าคงค้าง	1929935	2	No	1915954

จากตารางที่ 7 พบว่ามีเพียงคุณลักษณะเวลาขนส่งที่ข้อมูลขาดหายไป และคุณลักษณะรหัสสินค้าที่มีจำนวนเท่ากับความหลากหลายซึ่งหมายถึงไม่มีรหัสสินค้าที่ซ้ำกันเลย ควรตัดคุณลักษณะนี้ออกก่อนฝึกข้อมูลเนื่องจากไม่ส่งผลต่อการพยากรณ์ ส่วนของคุณลักษณะประสิทธิภาพแหล่งที่มา 6,12 เดือน มีตัวเลข -99 ซึ่งอาจเกิดจากความผิดพลาดในการบันทึกข้อมูล เช่นเดียวกับจำนวนสินค้าคงคลังที่เป็นจำนวนติดลบ ซึ่งข้อมูลนี้ส่งผลต่อลักษณะข้อมูลโดยรวมอาจส่งผลเสียกับการฝึกข้อมูลได้ นอกจากนี้ข้อมูลเชิงปริมาณยังมีลักษณะการกระจายตัวที่ไม่สมมาตรโดยดูจากความเบ้ดังตารางที่ 7 ซึ่งส่วนนี้การตัดค่า Outlier จึงต้องใช้วิธีที่เหมาะสมกับข้อมูล ส่วนตารางที่ 8 พบว่าคุณลักษณะส่วนใหญ่มีความเอนเอียงไปที่ค่าใดค่าหนึ่ง และในทุกคุณลักษณะมีค่าเพียง 2 ค่า นอกจากนี้ในส่วนของคุณลักษณะที่ต้องการพยากรณ์นั้น มีความไม่สมดุลของข้อมูลอยู่ 1 ต่อ 137

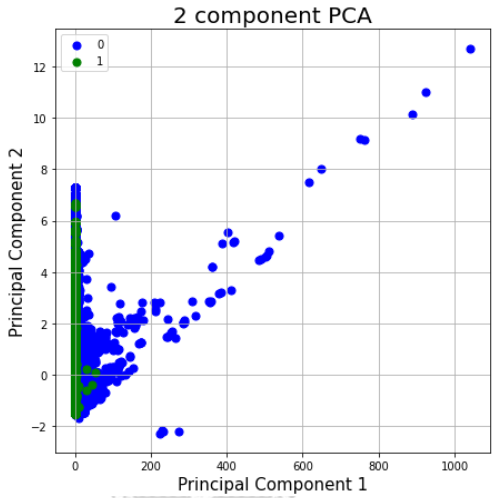
3.2 การเตรียมชุดข้อมูล

จากหัวข้อ 3.1 พบว่าข้อมูลปัจจุบันยังไม่สามารถนำไปฝึกได้ เนื่องจากมีข้อมูลที่ผิดปกติ ข้อมูลที่หายไป รวมถึงข้อมูลที่เป็นข้อความอยู่ ดังนั้นผู้วิจัยจึงได้ปรับเปลี่ยนข้อมูลดังต่อไปนี้

1. เปลี่ยนข้อมูลเชิงคุณภาพเป็นตัวเลขโดยค่า Yes ให้เป็นเลข 1 และค่า No ให้เป็นเลข 0
2. ลบรหัสสินค้าที่ข้อมูลเชิงปริมาณแสดงค่า 0 ในทุกคุณลักษณะ
3. ลบรหัสสินค้าที่คุณลักษณะประสิทธิภาพของแหล่งที่มาในช่วง 6,12 เดือนที่ผ่านมา แสดงค่า -99
4. เติมข้อมูลที่หายไปของคุณลักษณะเวลาขนส่งด้วยค่าเฉลี่ย

5. ลบคุณลักษณะรหัสสินค้าออกจากชุดข้อมูลก่อนนำไปใช้ฝึก
6. ตัดรหัสสินค้าที่พบว่าเป็น Outlier จากวิธีการ Medcouple

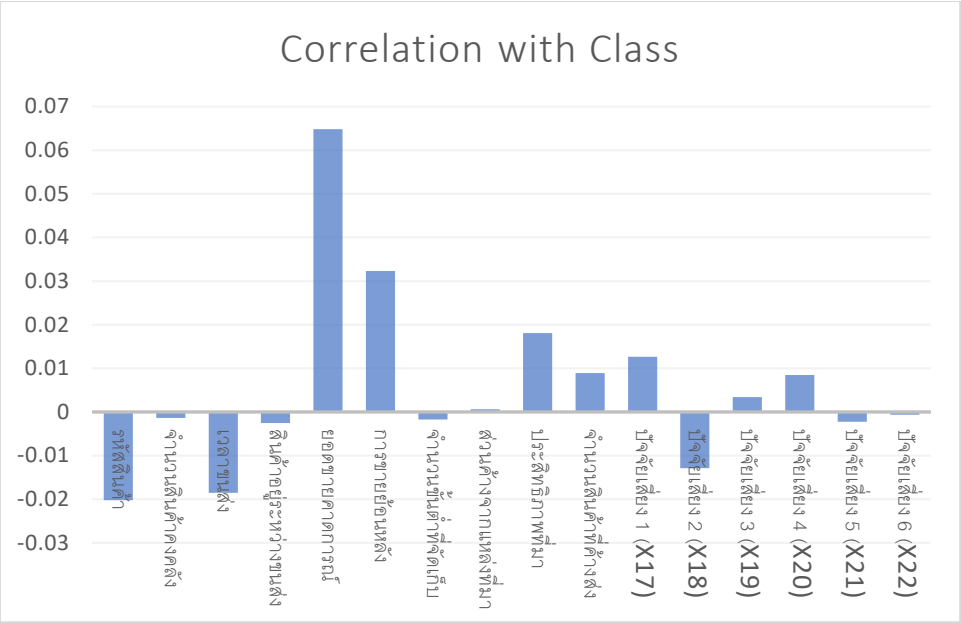
หลังจากตัดข้อมูลและเติมค่าที่หายไปแล้วจึงดูการกระจายของข้อมูล เพื่อเข้าใจลักษณะของข้อมูลมากขึ้น ด้วยการทำ Principal Component Analysis (PCA) ซึ่งมีหลักการคือลดมิติของข้อมูลจากเดิมที่มี 22 มิติให้เหลือ 2 หรือ 3 มิติดังรูปที่ 3.2



รูปที่ 3.2ลักษณะข้อมูลแบบ 2 มิติ

หลังจากทำการลดระดับมิติเหลือ 2 มิติ พบว่าเหลือข้อมูลอยู่ทั้งหมดร้อยละ 42.7 ของข้อมูลมีการกระจายตัวดังรูป โดยใน Principal Component 1 มีคุณลักษณะที่สำคัญได้แก่คุณลักษณะที่ X4 ถึง X11 ส่วน Principal Component 2 มีคุณลักษณะที่สำคัญคือ X14 และ X15 จากที่กล่าวมาข้างต้นสามารถสรุปได้ว่า สินค้าคงค้างมักกระจายตัวอยู่ใน Principal Component 1 ที่มีค่าน้อยกว่าใน Principal Component 2 มีการกระจายอยู่ในช่วงค่าที่ต่ำถึงปานกลาง และแม้ว่าข้อมูลที่เป็นสินค้าคงค้างจะกระจายตัวเป็นกลุ่มแต่ก็ยากที่จะหาจุดแบ่งที่ชัดเจน ในการจำแนกข้อมูล

ลักษณะความสัมพันธ์ของคุณลักษณะกับสินค้าคงค้างหรือค่าสหสัมพันธ์นั้นมีความสำคัญกับการดูข้อมูล แม้ว่าในการเรียนรู้ของเครื่องค่าสหสัมพันธ์อาจไม่ได้มีความจำเป็นมาก แต่ในบางครั้งหากข้อมูลมีคุณลักษณะจำนวนมาก การตัดค่าคุณลักษณะที่มีค่าสหสัมพันธ์กับตัวแปรที่ต้องการทำนายน้อยก็จะช่วยให้ระยะเวลาเรียนรู้เร็วมากขึ้น และยังลดความซับซ้อนในการเรียนรู้อีกด้วย โดยข้อมูลชุดนี้มีค่าสหสัมพันธ์ระหว่างคุณลักษณะกับสินค้าคงค้างดังรูปที่ 3.3



รูปที่ 3.3 ค่าสหสัมพันธ์ระหว่างคุณลักษณะกับสินค้าคงค้าง

จากรูปความสัมพันธ์ระหว่างคุณลักษณะกับสินค้าคงค้าง ซึ่งคุณลักษณะที่มีความคล้ายคลึงกันจะทำ PCA ให้เหลือ 1 คุณลักษณะ ได้แก่ คุณลักษณะคาดการณ์ยอดขายในอีก 3,6,9 เดือน เปลี่ยนเป็นคุณลักษณะยอดขายขาดการณ์ ,คุณลักษณะปริมาณการขายในช่วงเวลา 1,3,6,9 เดือนที่ผ่านมา เปลี่ยนเป็นคุณลักษณะการขายย้อนหลัง และคุณลักษณะประสิทธิภาพของแหล่งที่มาในช่วง 6,12 เดือนที่ผ่านมา เปลี่ยนเป็นคุณลักษณะประสิทธิภาพที่มา ซึ่งพบว่าคุณลักษณะทั้งหมดมีความสัมพันธ์ในระดับที่น้อยมากกับสินค้าคงค้าง ดังนั้นทุกคุณลักษณะจึงถูกใช้ในการฝึกเพื่อประสิทธิภาพที่ดีที่สุด

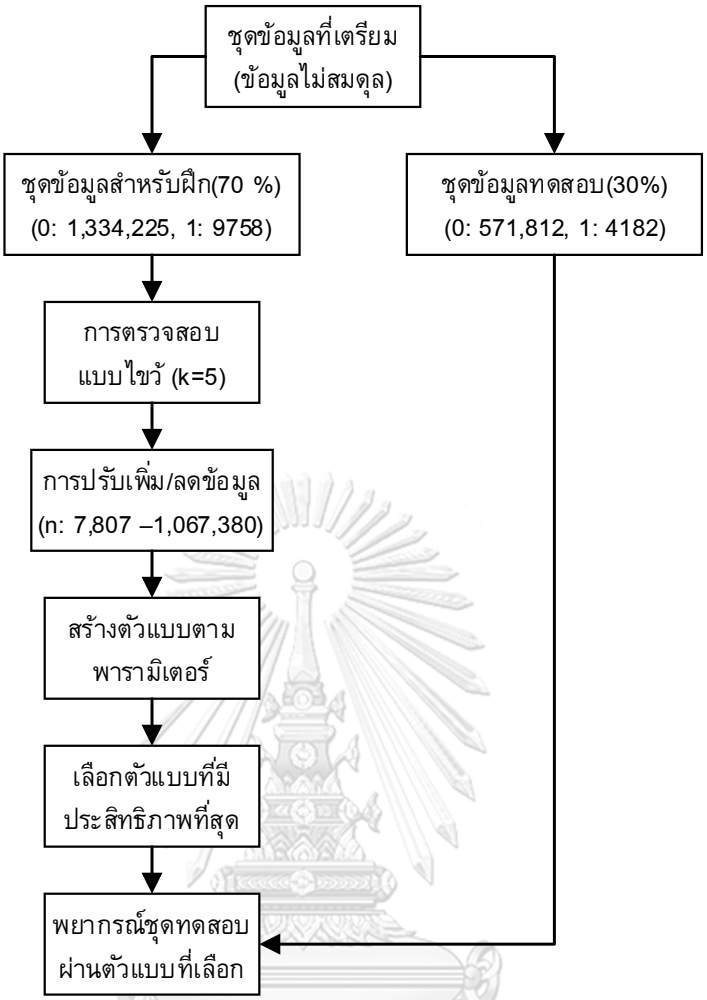
จากตารางที่ 7 และ 8 สังเกตได้ว่าในแต่ละคุณลักษณะมีความแตกต่างกันในเรื่องของขนาดและช่วง ตัวอย่างเช่นคุณลักษณะจำนวนสินค้าคงคลังที่มีขนาดตั้งแต่หลักหมื่นถึงหลักสิบล้าน ขณะเดียวกันคุณลักษณะเวลาขนส่งมีขนาดหลักหน่วยถึงหลังสิบ ซึ่งสิ่งนี้เป็นปัญหากับอัลกอริทึมการเรียนรู้ด้วยเครื่องส่วนใหญ่ที่มีหลักการใช้ระยะห่าง Eucledian ระหว่างจุดในการคำนวณ หรือการจำแนกกลุ่มด้วยวิธี Gradient Descendent เช่น Neural Network ทว่าในอัลกอริทึมกลุ่มที่ใช้ต้นไม้เป็นพื้นฐาน เช่น Random Forest นั้นกลับไม่ได้ส่งผลมาก ดังนั้นจึงควรปรับขนาดของแต่ละคุณลักษณะให้อยู่ในช่วงที่เท่ากันหากอัลกอริทึมที่ใช้มีหลักการดังข้างต้นที่กล่าวมา แต่เนื่องจากการปรับขนาดนั้นสามารถทำได้หลายวิธีไม่ว่าจะเป็น RobustScaler หรือ StandardScaler และ Normalizer ที่ได้รับความนิยมและยังถูกใช้ในงานวิจัยในส่วนของการทบทวนวรรณกรรม ในบทความของ Geller (2019) ซึ่งได้ทดลองใช้การปรับขนาดวิธีต่างๆกับอัลกอริทึมที่หลากหลายกับชุดข้อมูลฝึก

จากการทดลองพบว่าสำหรับชุดข้อมูลที่ไม่สมดุล การปรับขนาดไม่ได้ส่งผลต่ออัลกอริทึม FOREST และ LOGIST ในกรณีที่ค่า AUROC มากกว่า 0.9 ในทางกลับกันหากค่า AUROC มีค่าน้อยกว่าหรือเท่ากับ 0.8 การปรับขนาดด้วยวิธี StandardScaler จะส่งผลให้ตัวแบบที่ฝึกด้วยอัลกอริทึม LOGIST มีประสิทธิภาพเพิ่มขึ้นร้อยละ 40 ทั้งยังมากกว่าวิธี Normalizer ส่วนอัลกอริทึม FOREST นั้นการปรับขนาดยังคงไม่ส่งผลกับประสิทธิภาพตัวแบบ (Geller, 2019) ดังนั้นในวิทยานิพนธ์ฉบับนี้จึงใช้วิธี StandardScaler ในการปรับขนาดเพื่อเพิ่มประสิทธิภาพตัวแบบ

3.3 การพัฒนาตัวแบบการพยากรณ์

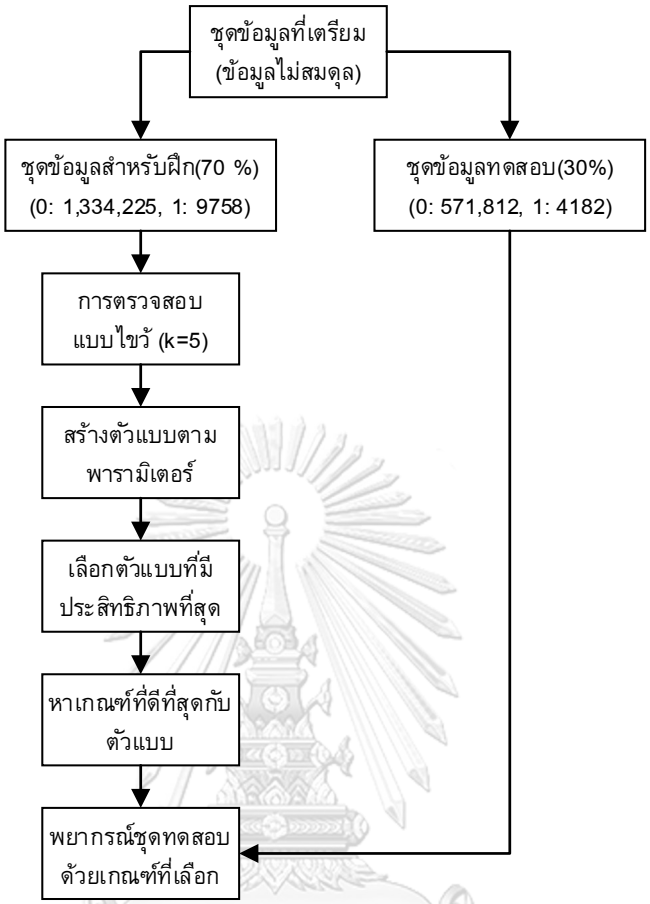
หลังจากที่ผู้วิจัยได้เตรียมชุดข้อมูลสำหรับการเรียนรู้ของเครื่องด้วยการตัดข้อมูลที่ผิดปกติออกจากชุดข้อมูล จากนั้นเติมข้อมูลในแต่ละคุณลักษณะที่หายไป แปลงข้อมูลของคุณลักษณะที่เป็นข้อความให้อยู่ในรูปตัวเลข และ ปรับข้อมูลให้มีช่วงที่เท่ากันตามที่แสดงในหัวข้อที่ 3.2 ขั้นต่อมาคือการแบ่งข้อมูลออกเป็น 2 ส่วนคือข้อมูลสำหรับฝึกร้อยละ 70 และข้อมูลสำหรับทดสอบร้อยละ 30 และใช้การตรวจสอบแบบไขว้แบบ 5 กลุ่มกับข้อมูลสำหรับฝึกเพื่อหลีกเลี่ยงการเกิด Overfitting โดยในวิทยานิพนธ์ฉบับนี้ใช้วิธีการเพื่อจัดการกับความไม่สมดุลของข้อมูล 2 วิธี

วิธีแรกเป็นวิธีปรับระดับข้อมูลดังที่ได้กล่าวในหัวข้อที่ 2.3 เพื่อปรับข้อมูลก่อนเรียนรู้ด้วยเครื่อง โดยวิทยานิพนธ์ฉบับนี้มีการศึกษาอยู่ 5 วิธีเพื่อนำมาเปรียบเทียบกัน ประกอบด้วยชุดข้อมูลเดิมที่ไม่ใช้วิธีสุ่ม, การปรับลดข้อมูลด้วยวิธี NearMiss-3, การปรับลดข้อมูลด้วยวิธี OSS, การปรับเพิ่มข้อมูลด้วยวิธี SMOTE และการปรับลดและเพิ่มข้อมูลด้วยวิธี OSS ผสมกับ SMOTE โดยในแต่ละวิธีได้ใช้อัลกอริทึมที่กล่าวในหัวข้อที่ 2.1 ในการฝึกข้อมูล โดยแต่ละอัลกอริทึมมีการปรับค่าพารามิเตอร์ตามที่กำหนดไว้ในตารางที่ 10 ซึ่งพารามิเตอร์แต่ละค่าจะถูกผสมและคำนวณด้วยวิธี Grid Search จากนั้นใช้ AUROC และ F1 score ในการวัดประสิทธิภาพตัวแบบ ตัวแบบที่ให้ค่า AUROC และ F1 score ดีที่สุดจะถูกนำมามาพยากรณ์กับข้อมูลสำหรับทดสอบ ดังรูปที่ 3.4 โดยการวัดประสิทธิภาพด้วย F1 score เพื่อสามารถนำมาเปรียบเทียบกับวิธี Threshold-moving



รูปที่ 3.4 วิธีการสร้างตัวแบบด้วยการปรับระดับข้อมูล

วิธีที่สองเป็นวิธี Threshold-moving ดังที่ได้กล่าวในหัวข้อที่ 2.4 โดยเริ่มต้นจากการฝึกข้อมูลด้วยอัลกอริทึมที่มีการปรับค่าพารามิเตอร์ ตามที่กำหนดไว้ในตารางที่ 10 ซึ่งพารามิเตอร์แต่ละค่าจะถูกผสมและคำนวณด้วยวิธี Grid Search จากนั้นใช้ AUROC และ F1 score ในการวัดประสิทธิภาพตัวแบบ ตัวแบบที่ให้ค่า AUROC ดีที่สุดจะถูกนำมาวาดกราฟ ROC และหาเกณฑ์ที่ดีที่สุดด้วย G-Mean ส่วนตัวแบบที่ให้ค่า F1 score ดีที่สุดจะถูกนำมาวาดกราฟ Precision-Recall และหาเกณฑ์ที่ดีที่สุดด้วยการปรับเกณฑ์ภายในช่วง F1 score ตั้งแต่ 0 ถึง 1 ทั้งหมด 10,000 จุด จากนั้นนำเกณฑ์ที่เหมาะสมที่ให้ค่าดีที่สุดไปใช้กับชุดข้อมูลทดสอบและวัดประสิทธิภาพตัวแบบด้วย F1 score ดังรูปที่ 3.5 โดยการฝึกข้อมูลสำหรับทุกตัวแบบทำงานผ่าน Google Colaboratory และใช้ Sklearn เวอร์ชั่น 1.0.2 และแต่ละตัวแบบจะทำการสุ่มแบ่งข้อมูลฝึกและทดสอบที่ต่างกันเพื่อทำซ้ำอีก 30 ครั้ง



รูปที่ 3.5 วิธีการสร้างตัวแบบด้วย Threshold-moving

3.3.1 การปรับค่าพารามิเตอร์ด้วยฟังก์ชัน GridSearchCV เพื่อหาค่าที่เหมาะสมและส่งผลดีต่อประสิทธิภาพของตัวแบบดังที่กล่าวไว้ในหัวข้อที่ 2.7 ผู้วิจัยได้กำหนดชุดพารามิเตอร์โดยอ้างอิงจาก de Santis et al. (2017) และ Liu et al. (2022) ซึ่งแต่ละอัลกอริทึมมีการกำหนดและการปรับค่าพารามิเตอร์ดังนี้

3.3.1.1 อัลกอริทึม LOGIST มีการปรับและกำหนดค่าพารามิเตอร์ดังนี้

พารามิเตอร์ Regularization – C คือ ความผกผันของ Regularization ค่าที่ใช้ต้องเป็นค่าบวก โดยยิ่งค่า C มากจะหมายถึงตัวแบบจะให้น้ำหนักกับข้อมูลชุดฝึกมากไม่ได้เผื่อว่าข้อมูลจริงอาจมีความซับซ้อนของข้อมูล ดังนั้นค่า C ที่น้อยกว่าจะหมายถึงในชุดข้อมูลฝึกไม่ได้เป็นตัวแทนของข้อมูลทั้งหมด โดยการปรับค่าตั้งแต่ 1 ขึ้นไปนั้นอ้างอิงจาก de Santis et al. (2017) และปรับค่าจำนวนที่น้อยกว่า 1 เพิ่มเติมด้วยจำนวนเท่ากัน

พารามิเตอร์ max_iter คือจำนวนการวนซ้ำสูงสุด ซึ่งในวิทยานิพนธ์ฉบับนี้ใช้ 1,000

พารามิเตอร์อื่นจะใช้ค่าที่โปรแกรมกำหนดซึ่งมีดังนี้ penalty='l2', dual=False, tol=0.0001, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None

3.3.1.2 อัลกอริทึม FOREST มีการปรับและกำหนดค่าพารามิเตอร์ดังนี้

พารามิเตอร์ max_depth สำหรับอัลกอริทึม FOREST คือความลึกสูงสุดของต้นไม้ หากไม่ได้กำหนด โหนดจะถูกขยายจนกว่าใบทั้งหมดจะเหลือ 1 ใบหรือจนกว่าทุกใบจะมีตัวอย่างน้อยกว่า min_samples_split ซึ่งอ้างอิงและปรับใช้จาก de Santis et al. (2017)

พารามิเตอร์ min_samples_leaf คือจำนวนตัวอย่างขั้นต่ำที่ต้องอยู่ที่โหนดปลายสุด จุดแยกที่ความลึกใด ๆ จะได้รับการพิจารณาก็ต่อเมื่อเหลือตัวอย่างการฝึกอย่างน้อย min_samples_leaf ไว้ในแต่ละกิ่งทางซ้ายและขวา ซึ่งอ้างอิงและปรับใช้จาก de Santis et al. (2017)

พารามิเตอร์ n_estimators คือจำนวนต้นไม้ในป่า โดยวิทยานิพนธ์ฉบับนี้กำหนดไว้ที่ 10 ซึ่งอ้างอิงและปรับใช้จาก de Santis et al. (2017) ซึ่งช่วยลดระยะเวลาในการฝึกลง

พารามิเตอร์อื่นจะใช้ค่าที่โปรแกรมกำหนดซึ่งมีดังนี้ criterion='gini', min_samples_split=2, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None

3.3.1.2 อัลกอริทึม XGBoost มีการปรับและกำหนดค่าพารามิเตอร์ดังนี้

พารามิเตอร์ max_depth สำหรับอัลกอริทึม XGBoost คือความลึกสูงสุดของต้นไม้สำหรับการเรียนรู้พื้นฐาน ซึ่งอ้างอิงและปรับใช้จาก (Liu et al., 2022)

พารามิเตอร์ min_child_weight คือน้ำหนักรวม (hessian) ขั้นต่ำของตัวอย่าง เมื่อขนาดตัวอย่างในโหนดต่ำกว่าเกณฑ์ที่กำหนดตัวแบบจะหยุดแยกโหนด ซึ่งอ้างอิงและปรับใช้จาก (Liu et al., 2022)

พารามิเตอร์ n_estimators คือจำนวนรอบการ Boost หรือก็คือจำนวนต้นไม้ ซึ่งในวิทยานิพนธ์ฉบับนี้กำหนดให้ใช้ค่าเดียวกับอัลกอริทึม FOREST คือ 10

พารามิเตอร์ nthread คือจำนวนการฝึกคู่ขนาน ซึ่งในวิทยานิพนธ์ฉบับนี้กำหนดค่าคือ 1

พารามิเตอร์อื่นจะใช้ค่าที่โปรแกรมกำหนดซึ่งมีดังนี้ learning_rate: float = 0.1, verbosity: int = 1, silent: Any | None = None, objective: str = "binary:logistic", booster: str = 'gbtree', n_jobs: int = 1, gamma: int = 0, max_delta_step: int = 0, subsample: int = 1, colsample_bytree: int = 1, colsample_bylevel: int = 1, colsample_bynode: int = 1, reg_alpha: int = 0, reg_lambda: int = 1, scale_pos_weight: int = 1, base_score: float = 0.5, random_state: int = 0, seed: Any | None = None, missing: Any | None = None

จากรายละเอียดการปรับค่าพารามิเตอร์ที่กล่าวมาข้างต้น สามารถสรุปได้ดังตารางที่ 10

ตารางที่ 10 การปรับค่าพารามิเตอร์

	Parameters	Range
LOGIST	Regularization - C	0.001, 0.01, 0.1, 1, 10, 100, 1000
FOREST	max_depth	7,9,12
	min_samples_leaf	3, 4, 5
XGBoost	max_depth	7,9,12
	min_child_weight	1,5,10

3.4 การเปรียบเทียบผลการดำเนินงาน

จากผลการดำเนินงานตามหัวข้อที่ 3.3 จะได้ผลลัพธ์ทั้งหมด 450 ค่าสำหรับ AUROC ผลลัพธ์ 540 ค่าสำหรับ F1 score และผลลัพธ์ 90 ค่าสำหรับ G-mean จากนั้นผู้วิจัยจะนำผลการทดลองมาวิเคราะห์ความแปรปรวนแบบ 2 ตัวประกอบเพื่อสรุปว่าวิธีการแก้ปัญหาความไม่สมดุลนั้นส่งผลต่อการพยากรณ์หรือไม่ โดยตัวแปรอิสระที่ 1 คือวิธีการแก้ปัญหาความไม่สมดุล ประกอบด้วย 6 ระดับได้แก่ ข้อมูลเดิม, NM, OSS, SMOTE, OSSSMOTE และ Threshold-moving ส่วนตัวแปรอิสระที่ 2 คืออัลกอริทึม ประกอบด้วย 3 ระดับได้แก่ LOGIST, FOREST และ XGBoost โดยจะทำการทดสอบนัยสำคัญที่ระดับ α = 0.05 หากตัวแปรอิสระส่งผลต่อผลลัพธ์อย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05 ผู้วิจัยจะทำการเปรียบเทียบหลังการทดสอบรวมด้วยการทดสอบ Games-Howell เพื่อดูความแตกต่างระหว่างระดับสำหรับแต่ละรูปแบบการวัดประสิทธิภาพ เมื่อได้ผลลัพธ์แล้วผู้วิจัยจะทำการเปรียบเทียบ independent-samples t-test เพื่อหาวิธีการที่เหมาะสมที่สุดกับชุดข้อมูล

บทที่ 4
ผลการดำเนินวิจัย

จากการดำเนินการวิจัยเรื่องการพยากรณ์สินค้าคงค้างด้วยการเรียนรู้ของเครื่องสำหรับข้อมูลไม่สมดุล ผู้วิจัยได้ผลการดำเนินการจากบทที่ 3 โดยแบ่งออกเป็น 2 ส่วนดังนี้

1. ผลการพัฒนาตัวแบบการพยากรณ์
2. ผลการเปรียบเทียบตัวแบบการพยากรณ์

4.1 ผลการพัฒนาตัวแบบการพยากรณ์

จากการพัฒนาตัวแบบการพยากรณ์ด้วยการปรับระดับข้อมูลดังรูปที่ 3.3 ผลการวัดประสิทธิภาพตัวแบบด้วย AUROC ได้ผลลัพธ์ข้อมูลสำหรับฝึกเป็นไปตามตารางที่ 11 และตารางที่ 12 แสดงผล AUROC จากข้อมูลสำหรับทดสอบ โดยข้อมูลภายในตารางแสดงค่าเฉลี่ยของ AUROC และตามด้วยค่าเบี่ยงเบนมาตรฐานของแต่ละรูปแบบ

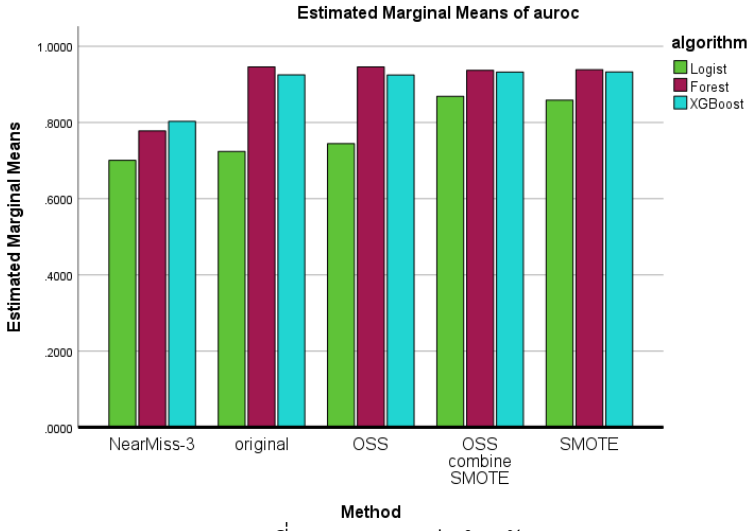
ตารางที่ 11 ผลลัพธ์ AUROC ของการปรับระดับข้อมูลสำหรับข้อมูลฝึก

AUROC (ข้อมูลฝึก)	LOGIST	FOREST	XGBoost
ข้อมูลเดิม	0.7248 ± 0.0073	0.9449 ± 0.0007	0.9204 ± 0.0025
NearMiss-3	0.6966 ± 0.0146	0.7910 ± 0.0064	0.8064 ± 0.0061
OSS	0.7455 ± 0.0121	0.9445 ± 0.0008	0.9224 ± 0.0020
SMOTE	0.8586 ± 0.0069	0.9355 ± 0.0008	0.9308 ± 0.0012
OSS ผสมSMOTE	0.8697 ± 0.0085	0.9354 ± 0.0009	0.9298 ± 0.0014

ตารางที่ 12 ผลลัพธ์ AUROC ของการปรับระดับข้อมูลสำหรับข้อมูลทดสอบ

AUROC (ข้อมูลทดสอบ)	LOGIST	FOREST	XGBoost
ข้อมูลเดิม	0.7241 ± 0.0074	0.9460 ± 0.0020	0.9251 ± 0.0039
NearMiss-3	0.7006 ± 0.0154	0.7778 ± 0.0128	0.8028 ± 0.0067
OSS	0.7444 ± 0.0161	0.9458 ± 0.0019	0.9246 ± 0.0028
SMOTE	0.8587 ± 0.0073	0.9384 ± 0.0019	0.9328 ± 0.0029
OSS ผสมSMOTE	0.8687 ± 0.0087	0.9370 ± 0.0020	0.9323 ± 0.0038

เมื่อนำผลลัพธ์ AUROC ของการปรับระดับข้อมูลสำหรับข้อมูลทดสอบมาวาดเป็นกราฟแท่งเพื่อวิเคราะห์เบื้องต้นจะได้ดังรูปที่ 4.1 โดยในแนวตั้งจะแสดงค่าของ AUROC ส่วนแกนในแนวนอนแสดงวิธีการปรับระดับข้อมูล และกราฟแท่งแต่ละแท่งแสดงผลตามอัลกอริทึมที่กำหนดซึ่งได้ผลวิเคราะห์เบื้องต้นดังนี้ กราฟแท่ง FOREST แสดงผลว่าเป็นอัลกอริทึมที่ให้ผลลัพธ์เฉลี่ยดีที่สุด โดยการปรับระดับข้อมูลด้วยวิธี NearMiss-3 ให้ผลลัพธ์แย่ที่สุด ส่วนข้อมูลเดิมและการปรับระดับข้อมูลด้วยวิธีอื่นให้ผลลัพธ์ที่ใกล้เคียงกัน กราฟแท่ง XGBoost แสดงผลว่าเป็นอัลกอริทึมที่ให้ผลลัพธ์เฉลี่ยดีรองลงมาโดยการปรับระดับข้อมูลด้วยวิธี NearMiss-3 ให้ผลลัพธ์แย่ที่สุด แต่ให้ผลลัพธ์ดีกว่าการปรับระดับข้อมูลด้วยวิธี NearMiss-3 ของอัลกอริทึม FOREST ส่วนข้อมูลเดิมและการปรับระดับข้อมูลด้วยวิธีอื่นให้ผลลัพธ์ที่ใกล้เคียงกัน กราฟแท่ง LOGIST แสดงผลว่าเป็นอัลกอริทึมที่ให้ผลลัพธ์แย่ที่สุดโดยการปรับระดับข้อมูลด้วยวิธี NearMiss-3 ให้ผลลัพธ์แย่ที่สุด รองลงมาคือการไม่ปรับระดับข้อมูล การปรับระดับข้อมูลด้วยวิธี OSS และการปรับระดับข้อมูลด้วยวิธี SMOTE ตามลำดับ ส่วนการปรับระดับข้อมูลด้วยวิธี OSS ผสมกับ SMOTE ให้ผลลัพธ์ดีที่สุด



รูปที่ 4.1 กราฟแท่งสำหรับ AUROC

จากการพัฒนาตัวแบบการพยากรณ์ด้วยการปรับระดับข้อมูลดังรูปที่ 3.3 ผลการวัดประสิทธิภาพตัวแบบด้วย F1 score ได้ผลลัพธ์ข้อมูลสำหรับฝึกเป็นไปตามตารางที่ 13 และตารางที่ 14 แสดงผล F1 score จากข้อมูลสำหรับทดสอบ โดยข้อมูลภายในตารางแสดงค่าเฉลี่ยของ F1 score และตามด้วยค่าเบี่ยงเบนมาตรฐานของแต่ละรูปแบบ

ตารางที่ 13 ผลลัพธ์ F1 score ของการปรับระดับข้อมูลสำหรับข้อมูลฝึก

F1(ข้อมูลฝึก)	LOGIST	FOREST	XGBoost
ข้อมูลเดิม	0.0022 ± 0.0005	0.0279 ± 0.0030	0.1139 ± 0.0067
NearMiss-3	0.0219 ± 0.0016	0.0336 ± 0.0008	0.0377 ± 0.0017
OSS	0.0031 ± 0.0010	0.0347 ± 0.0052	0.1261 ± 0.0102
SMOTE	0.0365 ± 0.0005	0.1227 ± 0.0018	0.1664 ± 0.0057
OSS ผสมSMOTE	0.0374 ± 0.0008	0.1388 ± 0.0046	0.1920 ± 0.0050

ตารางที่ 14 ผลลัพธ์ F1 score ของการปรับระดับข้อมูลสำหรับข้อมูลทดสอบ

F1(ข้อมูลทดสอบ)	LOGIST	FOREST	XGBoost
ข้อมูลเดิม	0.0020 ± 0.0009	0.0284 ± 0.0044	0.1200 ± 0.0056
NearMiss-3	0.0220 ± 0.0008	0.0320 ± 0.0024	0.0373 ± 0.0024
OSS	0.0023 ± 0.0011	0.0341 ± 0.0062	0.1275 ± 0.0113
SMOTE	0.0366 ± 0.0004	0.1236 ± 0.0030	0.1625 ± 0.0116
OSS ผสมSMOTE	0.0369 ± 0.0010	0.1318 ± 0.0054	0.1839 ± 0.0114

จากการพัฒนาตัวแบบการพยากรณ์ด้วยวิธี Threshold-moving ดังรูปที่ 3.4 ผลการวัดประสิทธิภาพตัวแบบด้วย G-Mean และ F1 score สำหรับฝึกเป็นไปตามตารางที่ 15 และตารางที่ 16 แสดงผล G-Mean และ F1 score สำหรับทดสอบ โดยข้อมูลภายในตารางแสดงค่าเฉลี่ย และตามด้วยค่าเบี่ยงเบนมาตรฐานของแต่ละรูปแบบ

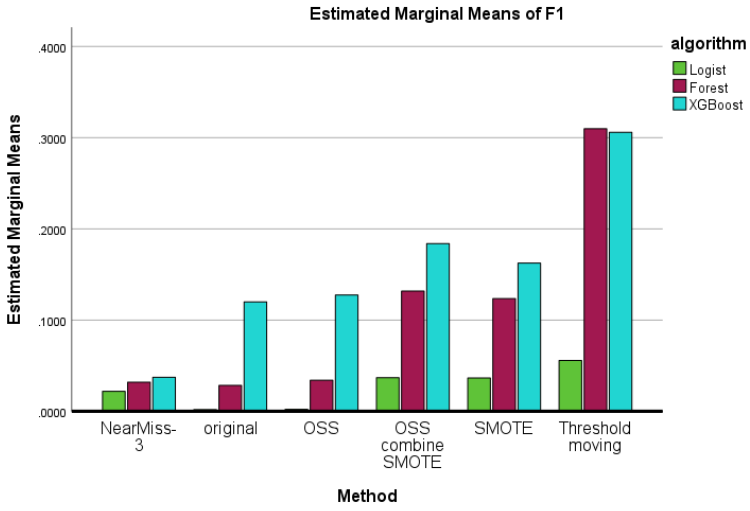
ตารางที่ 15 ผลลัพธ์ Threshold-moving สำหรับข้อมูลฝึก

Threshold-moving	LOGIST	FOREST	XGBoost
G-Mean	0.6533 ± 0.0062	0.9059 ± 0.0029	0.8671 ± 0.0025
F1 score	0.0557 ± 0.0022	0.3998 ± 0.0090	0.3958 ± 0.0518

ตารางที่ 16 ผลลัพธ์ Threshold-moving สำหรับข้อมูลทดสอบ

Threshold-moving	LOGIST	FOREST	XGBoost
G-Mean	0.6526 ± 0.0062	0.8737 ± 0.0041	0.8595 ± 0.0037
F1 score	0.0557 ± 0.0022	0.3098 ± 0.0063	0.3059 ± 0.0128

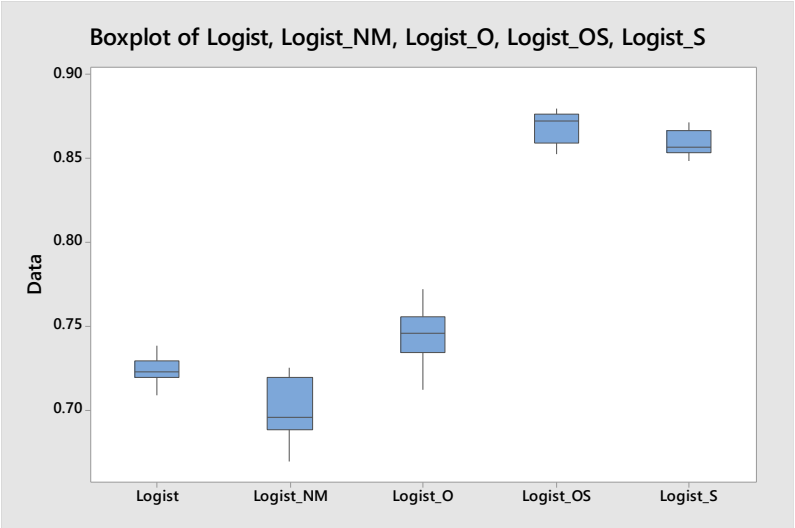
เมื่อนำผลลัพธ์ F1 score ของการจัดการกับข้อมูลสำหรับข้อมูลทดสอบมาวาดเป็นกราฟแท่งเพื่อวิเคราะห์เบื้องต้นจะได้ดังรูปที่ 4.2 โดยในแนวตั้งจะแสดงค่าของ F1 score ส่วนแกนในแนวนอนแสดงวิธีการจัดการกับข้อมูลประกอบด้วยการปรับระดับข้อมูลวิธีต่างๆ และ Threshold-moving โดยกราฟแท่งแต่ละแท่งแสดงผลตามอัลกอริทึมที่กำหนดซึ่งได้ผลวิเคราะห์เบื้องต้นดังนี้ กราฟแท่ง XGBoost แสดงผลว่าเป็นอัลกอริทึมที่ให้ผลลัพธ์เฉลี่ยดีที่สุด โดยการปรับระดับข้อมูลด้วยวิธี NearMiss-3 ให้ผลลัพธ์แย่ที่สุด รองลงมาคือการไม่ปรับระดับข้อมูล การปรับระดับข้อมูลด้วยวิธี OSS และการปรับระดับข้อมูลด้วยวิธี SMOTE และการปรับระดับข้อมูลด้วยวิธี OSS ผสมกับ SMOTE ตามลำดับ ส่วน Threshold-moving ให้ผลลัพธ์ดีที่สุด กราฟแท่ง FOREST แสดงผลว่าเป็นอัลกอริทึมที่ให้ผลลัพธ์เฉลี่ยดีรองลงมาโดยการปรับระดับข้อมูลด้วยวิธี NearMiss-3 OSS และการไม่ปรับระดับข้อมูลให้ผลลัพธ์แย่ที่สุด รองลงมาคือการปรับระดับข้อมูลด้วยวิธี SMOTE และวิธี OSS ผสมกับ SMOTE ส่วนผลลัพธ์ที่ดีที่สุดคือ Threshold-moving กราฟแท่ง LOGIST แสดงผลว่าเป็นอัลกอริทึมที่ให้ผลลัพธ์เฉลี่ยแย่ที่สุด โดยการปรับระดับข้อมูลด้วยวิธี OSS และการไม่ปรับระดับข้อมูลให้ผลลัพธ์แย่ที่สุด รองลงมาคือการปรับระดับข้อมูลด้วยวิธี NearMiss-3 ส่วนการปรับระดับข้อมูลด้วยวิธี SMOTE และการปรับระดับข้อมูลด้วย OSS ผสมกับ SMOTE ให้ผลลัพธ์ดีขึ้นเล็กน้อย ส่วน Threshold-moving ให้ผลลัพธ์ดีที่สุด



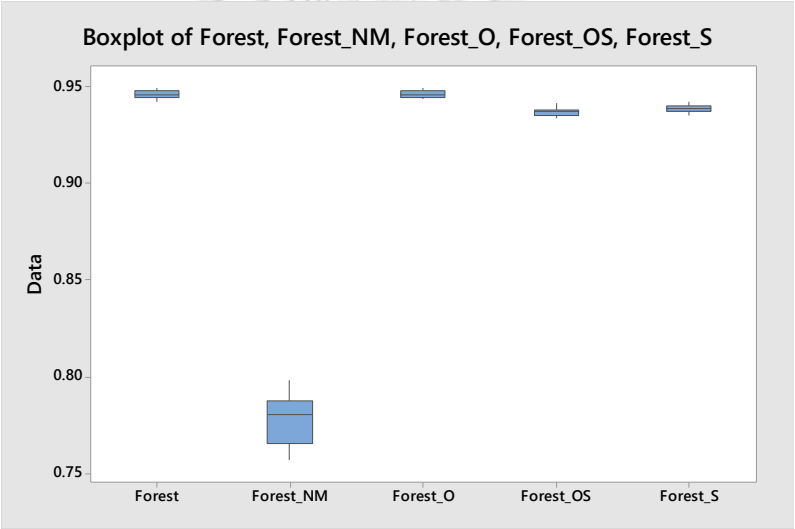
รูปที่ 4.2 กราฟแท่งสำหรับ F1 score

4.2 ผลการเปรียบเทียบตัวแบบการพยากรณ์

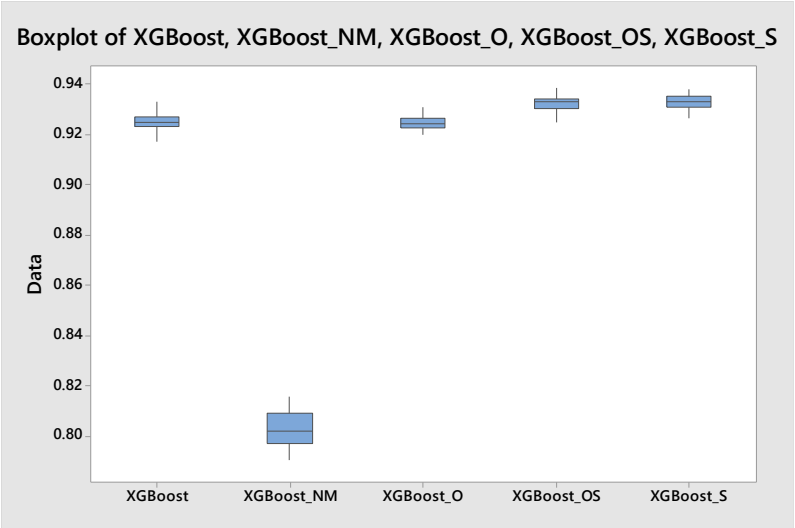
จากผลการพัฒนาตัวแบบการพยากรณ์ในหัวข้อที่ 4.1 ที่ได้มีการวิเคราะห์เบื้องต้นพบว่าอัลกอริทึม และการปรับระดับข้อมูลอาจส่งผลต่อการพยากรณ์ที่ต่างกัน ดังนั้นเพื่อวิเคราะห์ว่าอัลกอริทึม หรือวิธีการจัดการกับข้อมูลนั้นส่งผลต่อการพยากรณ์ ผู้วิจัยจึงวิเคราะห์ความแปรปรวนแบบสองทาง โดยก่อนการวิเคราะห์ผู้วิจัยได้ตรวจสอบสมมติฐานด้านข้อมูลผิดปกติของผลลัพธ์ AUROC แสดงดังรูปที่ 4.3 ถึง 4.5 ส่วนการทดสอบความเป็นเอกพันธ์ของความแปรปรวนของผลลัพธ์ AUROC แสดงดังตารางที่ 17 นอกจากนี้ผลการตรวจสอบสมมติฐานด้านข้อมูลผิดปกติของผลลัพธ์ F1 score แสดงดังรูปที่ 4.6 ถึง 4.8 ส่วนการทดสอบความเป็นเอกพันธ์ของความแปรปรวนของผลลัพธ์ F1 score แสดงดังตารางที่ 18



รูปที่ 4.3 Boxplot of LOGIST (AUROC)



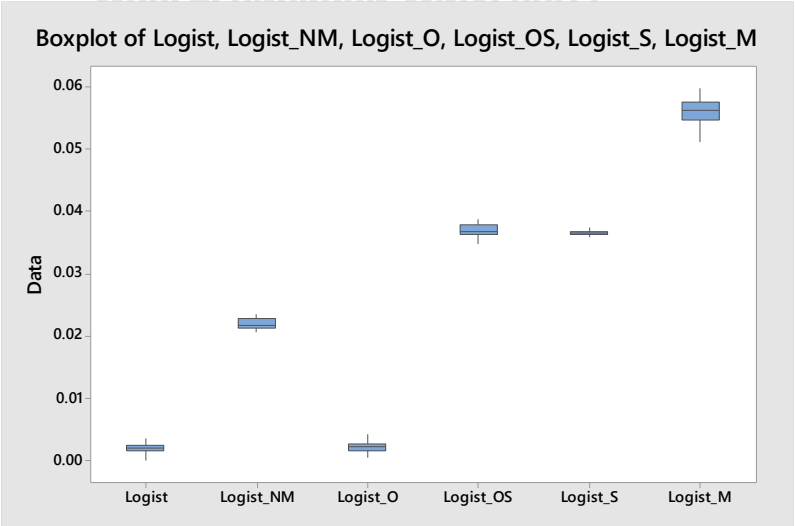
รูปที่ 4.4 Boxplot of FOREST (AUROC)



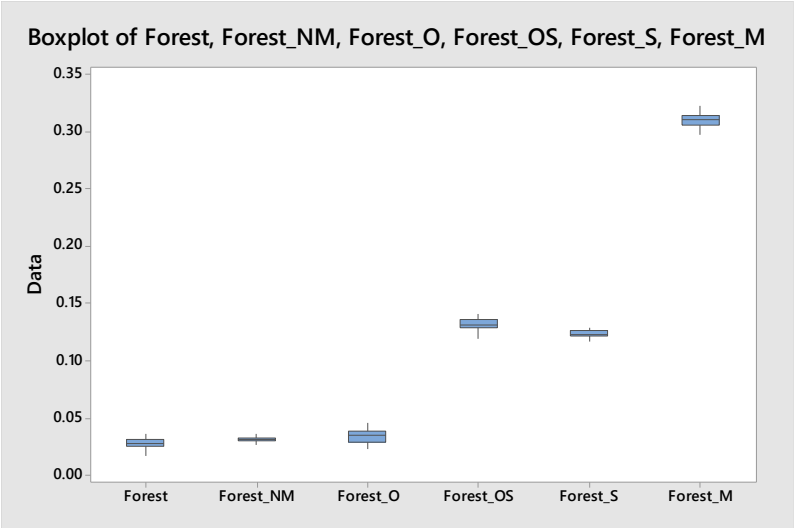
รูปที่ 4.5 Boxplot of XGBoost (AUROC)

ตารางที่ 17 การทดสอบของ Levene สำหรับ AUROC

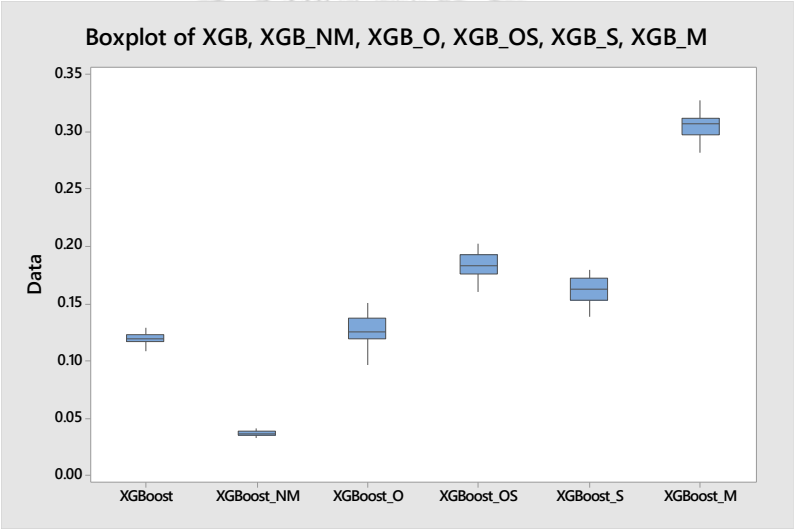
		Levene Statistic	df1	df2	Sig.
AUROC	Based on Mean	31.099	14	435	<.001
	Based on Median	21.572	14	435	<.001
	Based on Median and with adjusted df	21.572	14	156.872	<.001
	Based on trimmed mean	31.017	14	435	<.001



รูปที่ 4.6 Boxplot of LOGIST (F1 score)



รูปที่ 4.7 Boxplot of FOREST (F1 score)



รูปที่ 4.8 Boxplot of XGBoost (F1 score)

ตารางที่ 18 การทดสอบของ Levene สำหรับ F1 score

		Levene Statistic	df1	df2	Sig.
F1	Based on Mean	25.110	17	522	<.001
	Based on Median	23.629	17	522	<.001
	Based on Median and with adjusted df	23.629	17	185.328	<.001
	Based on trimmed mean	24.925	17	522	<.001

จากการตรวจสอบสมมติฐานด้านข้อมูลผิดปกติของผลลัพธ์ AUROC และ F1 score พบว่าผลลัพธ์ทั้งสองไม่มีข้อมูลที่ผิดปกติจึงไม่ละเมิดสมมติฐาน แต่จากการทดสอบความเป็นเอกพันธ์ของความแปรปรวนด้วยการทดสอบของ Levene พบว่าผลลัพธ์แต่ละอันมีความแปรปรวนที่ไม่เท่ากัน โดยสังเกตจากค่า p-value ที่น้อยกว่าระดับนัยสำคัญทางสถิติที่ α = 0.05 ซึ่งถือเป็นการละเมิดสมมติฐานก่อนการวิเคราะห์ แต่เนื่องจากขนาดของข้อมูลมีขนาดเท่ากันส่งผลให้ผลจากการวิเคราะห์ยังคงถูกต้องแม้ละเมิดสมมติฐาน โดยผลการวิเคราะห์ความแปรปรวนสองทางของ AUROC แสดงดังตารางที่ 19 และผลการวิเคราะห์ความแปรปรวนสองทางของ F1 score แสดงดังตารางที่ 20

ตารางที่ 19 ผลการวิเคราะห์ความแปรปรวนสองทางของ AUROC

Source of Variation	SS	df	MS	F	P-value	F crit
การปรับข้อมูล	1.3738	4	0.3434	5377.86	0.0000	2.3924
อัลกอริทึม	1.6141	2	0.8071	12637.89	0.0000	3.0165
Interaction	0.3981	8	0.0498	779.19	0.0000	1.9597
Within	0.0278	435	0.0001			
Total	3.4138	449				

ตารางที่ 20 ผลการวิเคราะห์ความแปรปรวนสองทางของ F1 score

Source of Variation	SS	df	MS	F	P-value	F crit
การปรับข้อมูล	2.2530	5	0.4506	10919.07	0.0000	2.2313
อัลกอริทึม	1.5704	2	0.7852	19026.34	0.0000	3.0130
Interaction	0.7715	10	0.0771	1869.40	0.0000	1.8488
Within	0.0215	522	0.0000			
Total	4.6164	539				

จากผลการวิเคราะห์ความแปรปรวนสองทางของ AUROC ในตารางที่ 19 สรุปได้ว่าในการศึกษาอิทธิพลร่วมระหว่างการปรับข้อมูลกับอัลกอริทึม (Interaction) ตอผล AUROC พบวา P-value เท่ากับศูนย์ซึ่งน้อยกว่าระดับนัยยะสำคัญที่ใช้ทดสอบที่ 0.05 แสดงว่ามีอิทธิพลร่วมระหว่างการปรับข้อมูลกับอัลกอริทึม

จากผลการวิเคราะห์ความแปรปรวนสองทางของ F1 score ในตารางที่ 20 สรุปได้ว่าในการศึกษาอิทธิพลร่วมระหว่างการปรับข้อมูลกับอัลกอริทึม (Interaction) ตอผล F1 score พบวา P-value เท่ากับศูนย์ซึ่งน้อยกว่าระดับนัยยะสำคัญที่ใช้ทดสอบที่ 0.05 แสดงว่ามีอิทธิพลร่วมระหว่างการปรับข้อมูลกับอัลกอริทึม

เนื่องจากการวิเคราะห์ที่ได้กล่าวมาข้างต้นมีอิทธิร่วมระหว่างการปรับข้อมูลกับอัลกอริทึม ซึ่งหมายถึงอิทธิพลของการปรับข้อมูลขึ้นอยู่กับอัลกอริทึม หรืออาจหมายถึงอิทธิพลของอัลกอริทึมขึ้นอยู่กับการปรับข้อมูล ดังนั้นผู้วิจัยจึงไม่สามารถตรวจสอบอิทธิพลหลักได้ ดังนั้นจึงทำการตรวจสอบละเอียดถึงอิทธิพลของการปรับข้อมูลในแต่ละอัลกอริทึมด้วยการวิเคราะห์ความแปรปรวนแบบ 1 ตัวประกอบ หากการปรับข้อมูลแต่ละวิธีส่งผลแตกต่างกันต่อ AUROC หรือ F1 score ผู้วิจัยจะทำการเปรียบเทียบหลังการทดสอบรวมด้วยการทดสอบ Games-Howell เพื่อตรวจสอบความแตกต่างรายคู่ของการปรับข้อมูล โดยผลการวิเคราะห์ความแปรปรวนแบบ 1 ตัวประกอบ และผลการทดสอบ Games-Howell แสดงดังตารางที่ 21 ถึง ถึงตารางที่ 32 ส่วนผลการวิเคราะห์ความแปรปรวนแบบ 1 ตัวประกอบและการเปรียบเทียบหลังการทดสอบรวมด้วยการทดสอบ Games-Howell สำหรับ G-Mean แสดงดังตารางที่ 33 และ 34 ตามลำดับ

ตารางที่ 21 ผลการวิเคราะห์ความแปรปรวนของ AUROC สำหรับ LOGIST

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	.743	4	.186	1361.287	<.001
Within Groups	.020	145	.000		
Total	.762	149			

จากตารางที่ 21 ค่าความน่าจะเป็นในการยอมรับสมมติฐาน Sig. ที่โปรแกรมคำนวณมาให้คือน้อยกว่า 0.001 มีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 จึงสรุปผลได้ว่าการปรับข้อมูลแต่ละวิธีการแตกต่างกันสำหรับ LOGIST ที่ระดับนัยยะสำคัญ 0.05 ผู้วิจัยจึงทำการเปรียบเทียบหลังการทดสอบรวมด้วยการทดสอบ Games-Howell และได้ผลดังตารางที่ 22

ตารางที่ 22 ผลการทดสอบ Games-Howell ของ AUROC สำหรับ LOGIST

(I) Method	(J) Method	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
ข้อมูลเดิม	NearMiss-3	.0234767*	.0031226	<.001	.014573	.032380
	OSS	-.0203700*	.0032364	<.001	-.029608	-.011132
	OSSSMOTE	-.1446433*	.0020853	<.001	-.150520	-.138767
	SMOTE	-.1346100*	.0018876	<.001	-.139924	-.129296
NearMiss-3	ข้อมูลเดิม	-.0234767*	.0031226	<.001	-.032380	-.014573
	OSS	-.0438467*	.0040770	<.001	-.055326	-.032367
	OSSSMOTE	-.1681200*	.0032398	<.001	-.177319	-.158921
	SMOTE	-.1580867*	.0031162	<.001	-.166974	-.149199
OSS	ข้อมูลเดิม	.0203700*	.0032364	<.001	.011132	.029608
	NearMiss-3	.0438467*	.0040770	<.001	.032367	.055326
	OSSSMOTE	-.1242733*	.0033496	<.001	-.133794	-.114753
	SMOTE	-.1142400*	.0032303	<.001	-.123463	-.105017
OSSSMOTE	ข้อมูลเดิม	.1446433*	.0020853	<.001	.138767	.150520
	NearMiss-3	.1681200*	.0032398	<.001	.158921	.177319
	OSS	.1242733*	.0033496	<.001	.114753	.133794
	SMOTE	.0100333*	.0020757	<.001	.004183	.015884
SMOTE	ข้อมูลเดิม	.1346100*	.0018876	<.001	.129296	.139924
	NearMiss-3	.1580867*	.0031162	<.001	.149199	.166974
	OSS	.1142400*	.0032303	<.001	.105017	.123463
	OSSSMOTE	-.0100333*	.0020757	<.001	-.015884	-.004183

จากตารางที่ 22 ค่า Sig. คือความน่าจะเป็นในการยอมรับสมมติฐานของความแตกต่าง ค่า 95% Confidence Interval คือขอบเขตช่วงความเชื่อมั่น 95% ของค่าเฉลี่ยในแต่ละกลุ่ม เมื่อค่า Sig. ที่โปรแกรมคำนวณมีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 หมายถึงทั้งสองกลุ่มแตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05 ดังนั้นจากตารางที่ 22 จึงสรุปได้ว่าสำหรับอัลกอริทึม

LOGIST แต่ละวิธีของการปรับระดับข้อมูลส่งผลต่อ AUROC แตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05

ตารางที่ 23 ผลการวิเคราะห์ความแปรปรวนของ AUROC สำหรับ FOREST

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	.647	4	.162	4542.778	<.001
Within Groups	.005	145	.000		
Total	.652	149			

จากตารางที่ 23 ค่าความน่าจะเป็นในการยอมรับสมมติฐาน Sig. ที่โปรแกรมคำนวณมาให้คือน้อยกว่า 0.001 มีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 จึงสรุปผลได้ว่าการปรับข้อมูลแต่ละวิธีการแตกต่างกันสำหรับ FOREST ที่ระดับนัยยะสำคัญ 0.05 ผู้วิจัยจึงทำการเปรียบเทียบหลังการทดสอบรวมด้วยการทดสอบ Games-Howell และได้ผลดังตารางที่ 24

ตารางที่ 24 ผลการทดสอบ Games-Howell ของ AUROC สำหรับ FOREST

		Mean Difference			95% Confidence Interval	
(I) Method	(J) Method	(I-J)	Std. Error	Sig.	Lower Bound	Upper Bound
ข้อมูลเดิม	NearMiss-3	.1681700*	.0023579	<.001	.161336	.175004
	OSS	.0002100	.0005011	.993	-.001201	.001621
	OSSSMOTE	.0090533*	.0005091	<.001	.007620	.010487
	SMOTE	.0075633*	.0004973	<.001	.006163	.008963
NearMiss-3	ข้อมูลเดิม	-.1681700*	.0023579	<.001	-.175004	-.161336
	OSS	-.1679600*	.0023573	<.001	-.174793	-.161127
	OSSSMOTE	-.1591167*	.0023591	<.001	-.165954	-.152280
	SMOTE	-.1606067*	.0023565	<.001	-.167438	-.153775
OSS	ข้อมูลเดิม	-.0002100	.0005011	.993	-.001621	.001201
	NearMiss-3	.1679600*	.0023573	<.001	.161127	.174793

	OSSSMOTE	.0088433*	.0005067	<.001	.007417	.010270
	SMOTE	.0073533*	.0004949	<.001	.005960	.008747
OSSSMOTE	ข้อมูลเดิม	-.0090533*	.0005091	<.001	-.010487	-.007620
	NearMiss-3	.1591167*	.0023591	<.001	.152280	.165954
	OSS	-.0088433*	.0005067	<.001	-.010270	-.007417
	SMOTE	-.0014900*	.0005030	.034	-.002906	-.000074
SMOTE	ข้อมูลเดิม	-.0075633*	.0004973	<.001	-.008963	-.006163
	NearMiss-3	.1606067*	.0023565	<.001	.153775	.167438
	OSS	-.0073533*	.0004949	<.001	-.008747	-.005960
	OSSSMOTE	.0014900*	.0005030	.034	.000074	.002906

*. The mean difference is significant at the 0.05 level.

จากตารางที่ 24 ค่า Sig. คือความน่าจะเป็นในการยอมรับสมมติฐานของความแตกต่าง ค่า 95% Confidence Interval คือขอบเขตช่วงความเชื่อมั่น 95% ของค่าเฉลี่ยในแต่ละกลุ่ม เมื่อค่า Sig. ที่โปรแกรมคำนวณมีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 หมายถึงทั้งสองกลุ่มแตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05 ดังนั้นจากตารางที่ 24 จึงสรุปได้ว่าสำหรับอัลกอริทึม FOREST แต่ละวิธีของการปรับระดับข้อมูลส่งผลต่อ AUROC แตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05 ยกเว้นการปรับระดับข้อมูลด้วยวิธี OSS กับข้อมูลเดิมที่ให้ผลลัพธ์ไม่แตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05

ตารางที่ 25 ผลการวิเคราะห์ความแปรปรวนของ AUROC สำหรับ XGBoost

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	.382	4	.096	5280.453	<.001
Within Groups	.003	145	.000		
Total	.385	149			

จากตารางที่ 25 ค่าความน่าจะเป็นในการยอมรับสมมติฐาน Sig. ที่โปรแกรมคำนวณมาให้คือน้อยกว่า 0.001 มีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 จึงสรุปผลได้ว่าการปรับข้อมูลแต่ละวิธีการแตกต่างกันสำหรับ XGBoost ที่ระดับนัยยะสำคัญ 0.05 ผู้วิจัยจึงทำการเปรียบเทียบหลังการทดสอบรวมด้วยการทดสอบ Games-Howell และได้ผลดังตารางที่ 26

ตารางที่ 26 ผลการทดสอบ Games-Howell ของ AUROC สำหรับ XGBoost

(I) Method	(J) Method	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
ข้อมูลเดิม	NearMiss-3	.1223033*	.0014141	<.001	.118291	.126316
	OSS	.0005400	.0008748	.972	-.001931	.003011
	OSSSMOTE	-.0072067*	.0009892	<.001	-.009992	-.004422
	SMOTE	-.0077100*	.0008871	<.001	-.010214	-.005206
NearMiss-3	ข้อมูลเดิม	-.1223033*	.0014141	<.001	-.126316	-.118291
	OSS	-.1217633*	.0013249	<.001	-.125553	-.117974
	OSSSMOTE	-.1295100*	.0014030	<.001	-.133494	-.125526
	SMOTE	-.1300133*	.0013330	<.001	-.133822	-.126204
OSS	ข้อมูลเดิม	-.0005400	.0008748	.972	-.003011	.001931
	NearMiss-3	.1217633*	.0013249	<.001	.117974	.125553
	OSSSMOTE	-.0077467*	.0008568	<.001	-.010165	-.005328
	SMOTE	-.0082500*	.0007366	<.001	-.010324	-.006176
OSSSMOTE	ข้อมูลเดิม	.0072067*	.0009892	<.001	.004422	.009992
	NearMiss-3	.1295100*	.0014030	<.001	.125526	.133494
	OSS	.0077467*	.0008568	<.001	.005328	.010165
	SMOTE	-.0005033	.0008693	.978	-.002956	.001949
SMOTE	ข้อมูลเดิม	.0077100*	.0008871	<.001	.005206	.010214
	NearMiss-3	.1300133*	.0013330	<.001	.126204	.133822
	OSS	.0082500*	.0007366	<.001	.006176	.010324
	OSSSMOTE	.0005033	.0008693	.978	-.001949	.002956

จากตารางที่ 26 ค่า Sig. คือความน่าจะเป็นในการยอมรับสมมติฐานของความแตกต่าง ค่า 95% Confidence Interval คือขอบเขตช่วงความเชื่อมั่น 95% ของค่าเฉลี่ยในแต่ละกลุ่ม เมื่อค่า Sig. ที่โปรแกรมคำนวณมีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 หมายถึงทั้งสองกลุ่มแตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05 ดังนั้นจากตารางที่ 26 จึงสรุปได้ว่าสำหรับอัลกอริทึม

XGBoost แต่ละวิธีของการปรับระดับข้อมูลส่งผลต่อ AUROC แตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05 ยกเว้นการปรับระดับข้อมูลด้วยวิธี OSS กับข้อมูลเดิม และ การปรับระดับข้อมูลด้วยวิธี OSS ผสม SMOTE กับการปรับระดับข้อมูลด้วย SMOTE ที่ให้ผลลัพธ์ไม่แตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05

ตารางที่ 27 ผลการวิเคราะห์ความแปรปรวนของ F1 score สำหรับ LOGIST

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	.068	5	.014	9411.394	<.001
Within Groups	.000	174	.000		
Total	.068	179			

จากตารางที่ 27 ค่าความน่าจะเป็นในการยอมรับสมมติฐาน Sig. ที่โปรแกรมคำนวณมาให้คือน้อยกว่า 0.001 มีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 จึงสรุปผลได้ว่าการปรับข้อมูลแต่ละวิธีการแตกต่างกันสำหรับ LOGIST ที่ระดับนัยยะสำคัญ 0.05 ผู้วิจัยจึงทำการเปรียบเทียบหลังการทดสอบรวมด้วยการทดสอบ Games-Howell และได้ผลดังตารางที่ 28

ตารางที่ 28 ผลการทดสอบ Games-Howell ของ F1 score สำหรับ LOGIST

		Mean Difference			95% Confidence Interval	
(I) Method	(J) Method	(I-J)	Std. Error	Sig.	Lower Bound	Upper Bound
ข้อมูลเดิม	NearMiss-3	-.0200333*	.0002276	<.001	-.020704	-.019362
	OSS	-.0002467	.0002518	.923	-.000989	.000495
	OSSSMOTE	-.0349267*	.0002564	<.001	-.035683	-.034171
	SMOTE	-.0345900*	.0001823	<.001	-.035138	-.034042
	Threshold Moving	-.0537700*	.0004398	<.001	-.055087	-.052453
NearMiss-3	ข้อมูลเดิม	.0200333*	.0002276	<.001	.019362	.020704
	OSS	.0197867*	.0002374	<.001	.019086	.020487

	OSSSMOTE	-.0148933*	.0002423	<.001	-.015609	-.014178
	SMOTE	-.0145567*	.0001619	<.001	-.015042	-.014072
	Threshold Moving	-.0337367*	.0004317	<.001	-.035034	-.032439
OSS	ข้อมูลเดิม	.0002467	.0002518	.923	-.000495	.000989
	NearMiss-3	-.0197867*	.0002374	<.001	-.020487	-.019086
	OSSSMOTE	-.0346800*	.0002652	<.001	-.035462	-.033898
	SMOTE	-.0343433*	.0001944	<.001	-.034929	-.033758
	Threshold Moving	-.0535233*	.0004449	<.001	-.054854	-.052193
OSSSMOTE	ข้อมูลเดิม	.0349267*	.0002564	<.001	.034171	.035683
	NearMiss-3	.0148933*	.0002423	<.001	.014178	.015609
	OSS	.0346800*	.0002652	<.001	.033898	.035462
	SMOTE	.0003367	.0002004	.554	-.000267	.000941
	Threshold Moving	-.0188433*	.0004476	<.001	-.020181	-.017506
SMOTE	ข้อมูลเดิม	.0345900*	.0001823	<.001	.034042	.035138
	NearMiss-3	.0145567*	.0001619	<.001	.014072	.015042
	OSS	.0343433*	.0001944	<.001	.033758	.034929
	OSSSMOTE	-.0003367	.0002004	.554	-.000941	.000267
	Threshold Moving	-.0191800*	.0004097	<.001	-.020425	-.017935
Threshold Moving	ข้อมูลเดิม	.0537700*	.0004398	<.001	.052453	.055087
	NearMiss-3	.0337367*	.0004317	<.001	.032439	.035034
	OSS	.0535233*	.0004449	<.001	.052193	.054854
	OSSSMOTE	.0188433*	.0004476	<.001	.017506	.020181
	SMOTE	.0191800*	.0004097	<.001	.017935	.020425

*. The mean difference is significant at the 0.05 level.

จากตารางที่ 28 ค่า Sig. คือความน่าจะเป็นในการยอมรับสมมติฐานของความแตกต่าง ค่า 95% Confidence Interval คือขอบเขตช่วงความเชื่อมั่น 95% ของค่าเฉลี่ยในแต่ละกลุ่ม เมื่อค่า Sig. ที่โปรแกรมคำนวณมีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 หมายถึงทั้งสองกลุ่มแตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05 ดังนั้นจากตารางที่ 28 จึงสรุปได้ว่าสำหรับอัลกอริทึม LOGIST แต่ละวิธีของการปรับระดับข้อมูลส่งผลต่อ F1 score แตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05 ยกเว้นการปรับระดับข้อมูลด้วยวิธี OSS กับข้อมูลเดิม และ การปรับระดับข้อมูลด้วยวิธี OSS ผสม SMOTE กับการปรับระดับข้อมูลด้วย SMOTE ที่ให้ผลลัพธ์ไม่แตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05

ตารางที่ 29 ผลการวิเคราะห์ความแปรปรวนของ F1 score สำหรับ FOREST

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	1.772	5	.354	15016.773	<.001
Within Groups	.004	174	.000		
Total	1.776	179			

จากตารางที่ 29 ค่าความน่าจะเป็นในการยอมรับสมมติฐาน Sig. ที่โปรแกรมคำนวณมาให้คือน้อยกว่า 0.001 มีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 จึงสรุปผลได้ว่าการปรับข้อมูลแต่ละวิธีการแตกต่างกันสำหรับ FOREST ที่ระดับนัยยะสำคัญ 0.05 ผู้วิจัยจึงทำการเปรียบเทียบหลังการทดสอบรวมด้วยการทดสอบ Games-Howell และได้ผลดังตารางที่ 30

ตารางที่ 30 ผลการทดสอบ Games-Howell ของ F1 score สำหรับ FOREST

					95% Confidence Interval	
(I) Method	(J) Method	Mean Difference (I-J)	Std. Error	Sig.	Lower Bound	Upper Bound
ข้อมูลเดิม	NearMiss-3	-.0036000*	.0009081	.003	-.006307	-.000893
	OSS	-.0057167*	.0013876	.002	-.009821	-.001613
	OSSSMOTE	-.1034067*	.0012786	<.001	-.107180	-.099633
	SMOTE	-.0952067*	.0009731	<.001	-.098088	-.092325

	Threshold Moving	-.2813533*	.0014101	<.001	-.285526	-.277181
NearMiss-3	ข้อมูลเดิม	.0036000*	.0009081	.003	.000893	.006307
	OSS	-.0021167	.0012041	.504	-.005735	.001502
	OSSSMOTE	-.0998067*	.0010767	<.001	-.103033	-.096581
	SMOTE	-.0916067*	.0006865	<.001	-.093634	-.089579
	Threshold Moving	-.2777533*	.0012300	<.001	-.281451	-.274055
OSS	ข้อมูลเดิม	.0057167*	.0013876	.002	.001613	.009821
	NearMiss-3	.0021167	.0012041	.504	-.001502	.005735
	OSSSMOTE	-.0976900*	.0015034	<.001	-.102123	-.093257
	SMOTE	-.0894900*	.0012538	<.001	-.093234	-.085746
	Threshold Moving	-.2756367*	.0016167	<.001	-.280401	-.270872
OSSSMOTE	ข้อมูลเดิม	.1034067*	.0012786	<.001	.099633	.107180
	NearMiss-3	.0998067*	.0010767	<.001	.096581	.103033
	OSS	.0976900*	.0015034	<.001	.093257	.102123
	SMOTE	.0082000*	.0011320	<.001	.004831	.011569
	Threshold Moving	-.1779467*	.0015241	<.001	-.182442	-.173451
SMOTE	ข้อมูลเดิม	.0952067*	.0009731	<.001	.092325	.098088
	NearMiss-3	.0916067*	.0006865	<.001	.089579	.093634
	OSS	.0894900*	.0012538	<.001	.085746	.093234
	OSSSMOTE	-.0082000*	.0011320	<.001	-.011569	-.004831
	Threshold Moving	-.1861467*	.0012786	<.001	-.189967	-.182326
Threshold Moving	ข้อมูลเดิม	.2813533*	.0014101	<.001	.277181	.285526
	NearMiss-3	.2777533*	.0012300	<.001	.274055	.281451
	OSS	.2756367*	.0016167	<.001	.270872	.280401

	OSSSMOTE	.1779467*	.0015241	<.001	.173451	.182442
	SMOTE	.1861467*	.0012786	<.001	.182326	.189967

*. The mean difference is significant at the 0.05 level.

จากตารางที่ 30 ค่า Sig. คือความน่าจะเป็นในการยอมรับสมมติฐานของความแตกต่าง ค่า 95% Confidence Interval คือขอบเขตช่วงความเชื่อมั่น 95% ของค่าเฉลี่ยในแต่ละกลุ่ม เมื่อค่า Sig. ที่โปรแกรมคำนวณมีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 หมายถึงทั้งสองกลุ่มแตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05 ดังนั้นจากตารางที่ 30 จึงสรุปได้ว่าสำหรับอัลกอริทึม FOREST แต่ละวิธีของการปรับระดับข้อมูลส่งผลต่อ F1 score แตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05 ยกเว้นการปรับระดับข้อมูลด้วยวิธี OSS กับ การปรับระดับข้อมูลด้วยวิธี NearMiss-3 ที่ให้ผลลัพธ์ไม่แตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05

ตารางที่ 31 ผลการวิเคราะห์ความแปรปรวนของ F1 score สำหรับ XGBoost

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	1.184	5	.237	2443.604	<.001
Within Groups	.017	174	.000		
Total	1.201	179			

จากตารางที่ 31 ค่าความน่าจะเป็นในการยอมรับสมมติฐาน Sig. ที่โปรแกรมคำนวณมาให้คือน้อยกว่า 0.001 มีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 จึงสรุปผลได้ว่าการปรับข้อมูลแต่ละวิธีการแตกต่างกันสำหรับ XGBoost ที่ระดับนัยยะสำคัญ 0.05 ผู้วิจัยจึงทำการเปรียบเทียบหลังการทดสอบรวมด้วยการทดสอบ Games-Howell และได้ผลดังตารางที่ 32

ตารางที่ 32 ผลการทดสอบ Games-Howell ของ F1 score สำหรับ XGBoost

					95% Confidence Interval	
(I) Method	(J) Method	Mean Difference (I-J)	Std. Error	Sig.	Lower Bound	Upper Bound
ข้อมูลเดิม	NearMiss-3	.0826733*	.0009396	<.001	.079872	.085474

	OSS	-.0074533*	.0022311	.021	-.014144	-.000762
	OSSSMOTE	-.0638433*	.0022468	<.001	-.070583	-.057104
	SMOTE	-.0424600*	.0022724	<.001	-.049278	-.035642
	Threshold Moving	-.1859067*	.0024721	<.001	-.193340	-.178473
NearMiss-3	ข้อมูลเดิม	-.0826733*	.0009396	<.001	-.085474	-.079872
	OSS	-.0901267*	.0021127	<.001	-.096533	-.083721
	OSSSMOTE	-.1465167*	.0021292	<.001	-.152973	-.140060
	SMOTE	-.1251333*	.0021562	<.001	-.131673	-.118594
	Threshold Moving	-.2685800*	.0023658	<.001	-.275761	-.261399
OSS	ข้อมูลเดิม	.0074533*	.0022311	.021	.000762	.014144
	NearMiss-3	.0901267*	.0021127	<.001	.083721	.096533
	OSSSMOTE	-.0563900*	.0029374	<.001	-.065047	-.047733
	SMOTE	-.0350067*	.0029571	<.001	-.043722	-.026292
	Threshold Moving	-.1784533*	.0031132	<.001	-.187632	-.169274
OSSSMOTE	ข้อมูลเดิม	.0638433*	.0022468	<.001	.057104	.070583
	NearMiss-3	.1465167*	.0021292	<.001	.140060	.152973
	OSS	.0563900*	.0029374	<.001	.047733	.065047
	SMOTE	.0213833*	.0029689	<.001	.012634	.030133
	Threshold Moving	-.1220633*	.0031245	<.001	-.131275	-.112852
SMOTE	ข้อมูลเดิม	.0424600*	.0022724	<.001	.035642	.049278
	NearMiss-3	.1251333*	.0021562	<.001	.118594	.131673
	OSS	.0350067*	.0029571	<.001	.026292	.043722
	OSSSMOTE	-.0213833*	.0029689	<.001	-.030133	-.012634
	Threshold Moving	-.1434467*	.0031429	<.001	-.152712	-.134181

Threshold Moving	ข้อมูลเดิม	.1859067*	.0024721	<.001	.178473	.193340
	NearMiss-3	.2685800*	.0023658	<.001	.261399	.275761
	OSS	.1784533*	.0031132	<.001	.169274	.187632
	OSSSMOTE	.1220633*	.0031245	<.001	.112852	.131275
	SMOTE	.1434467*	.0031429	<.001	.134181	.152712

*. The mean difference is significant at the 0.05 level.

จากตารางที่ 32 ค่า Sig. คือความน่าจะเป็นในการยอมรับสมมติฐานของความแตกต่าง ค่า 95% Confidence Interval คือขอบเขตช่วงความเชื่อมั่น 95% ของค่าเฉลี่ยในแต่ละกลุ่ม เมื่อค่า Sig. ที่โปรแกรมคำนวณมีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 หมายถึงทั้งสองกลุ่มแตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05 ดังนั้นจากตารางที่ 32 จึงสรุปได้ว่าสำหรับอัลกอริทึม XGBoost แต่ละวิธีของการปรับระดับข้อมูลส่งผลต่อ F1 score แตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05

ตารางที่ 33 ผลการวิเคราะห์ความแปรปรวนของ G-Mean

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	.919	2	.459	19777.053	<.001
Within Groups	.002	87	.000		
Total	.921	89			

จากตารางที่ 33 ค่าความน่าจะเป็นในการยอมรับสมมติฐาน Sig. ที่โปรแกรมคำนวณมาให้คือน้อยกว่า 0.001 มีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 จึงสรุปผลได้ว่าการปรับข้อมูลแต่ละวิธีการแตกต่างกันสำหรับ LOGIST ที่ระดับนัยยะสำคัญ 0.05 ผู้วิจัยจึงทำการเปรียบเทียบหลังการทดสอบรวมด้วยการทดสอบ Games-Howell และได้ผลดังตารางที่ 34

ตารางที่ 34 ผลการทดสอบ Games-Howell ของ G-Mean

(I) algorithm	(J) algorithm	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval Lower Bound	95% Confidence Interval Upper Bound
LOGIST	FOREST	-.2211040*	.0013633	<.001	-.224397	-.217811
	XGBoost	-.2068653*	.0013279	<.001	-.210078	-.203652
FOREST	LOGIST	.2211040*	.0013633	<.001	.217811	.224397
	XGBoost	.0142388*	.0010119	<.001	.011804	.016673
XGBoost	LOGIST	.2068653*	.0013279	<.001	.203652	.210078
	FOREST	-.0142388*	.0010119	<.001	-.016673	-.011804

*. The mean difference is significant at the 0.05 level.

จากตารางที่ 34 ค่า Sig. คือความน่าจะเป็นในการยอมรับสมมติฐานของความแตกต่าง ค่า 95% Confidence Interval คือขอบเขตช่วงความเชื่อมั่น 95% ของค่าเฉลี่ยในแต่ละกลุ่ม เมื่อค่า Sig. ที่โปรแกรมคำนวณมีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 หมายถึงทั้งสองกลุ่มแตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05 ดังนั้นจากตารางที่ 34 จึงสรุปได้ว่าสำหรับวิธี Threshold Moving ด้วยการวัดประสิทธิภาพ G-Mean แต่ละอัลกอริทึมส่งผลแตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05

จากการสรุปผลการเปรียบเทียบที่ได้กล่าวมาก่อนหน้า จะเห็นว่าในส่วนของ AUROC สำหรับอัลกอริทึม FOREST นั้นผลลัพธ์ที่ดีที่สุดมาจากการปรับระดับข้อมูลด้วยวิธี OSS และข้อมูลเดิมซึ่งหากดูจาก Confusion Matrix และการทบทวนวรรณกรรมจะพบว่าทั้ง 2 วิธีนี้ไม่สามารถพยากรณ์ข้อมูลส่วนน้อยได้เมื่อเทียบกับวิธีการที่เหลือ และหากดูจากรูปที่ 4.1 จะเห็นว่าสำหรับอัลกอริทึม XGBoost การปรับระดับข้อมูลที่ให้ผลลัพธ์ดีที่สุดคือ SMOTE และผลลัพธ์ยังใกล้เคียงกับอัลกอริทึม FOREST อีกด้วยผู้วิจัยจึงได้ทำการทดสอบอิทธิพลของอัลกอริทึมต่อการปรับระดับข้อมูลด้วย SMOTE ด้วยการวิเคราะห์ความแปรปรวนแบบ 1 ตัวประกอบ ซึ่งแสดงดังตารางที่ 35

ตารางที่ 35 ผลการวิเคราะห์ความแปรปรวนของ AUROC สำหรับ SMOTE

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	.119	2	.059	2747.707	<.001
Within Groups	.002	87	.000		
Total	.121	89			

จากตารางที่ 35 ค่าความน่าจะเป็นในการยอมรับสมมติฐาน Sig. ที่โปรแกรมคำนวณมาให้คือน้อยกว่า 0.001 มีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 จึงสรุปผลได้ว่าผลของแต่ละอัลกอริทึมแตกต่างกันสำหรับการปรับระดับข้อมูลด้วย SMOTE ที่ระดับนัยยะสำคัญ 0.05 ผู้วิจัยจึงทำการเปรียบเทียบหลังการทดสอบรวมด้วยการทดสอบ Games-Howell และได้ผลดังตารางที่ 36

ตารางที่ 36 ผลการทดสอบ Games-Howell ของ AUROC สำหรับ SMOTE

(I) algorithm	(J) algorithm	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval Lower Bound	95% Confidence Interval Upper Bound
LOGIST	FOREST	-.0797800*	.0013719	<.001	-.083147	-.076413
	XGBoost	-.0741367*	.0014295	<.001	-.077623	-.070650
FOREST	LOGIST	.0797800*	.0013719	<.001	.076413	.083147
	XGBoost	.0056433*	.0006345	<.001	.004111	.007176
XGBoost	LOGIST	.0741367*	.0014295	<.001	.070650	.077623
	FOREST	-.0056433*	.0006345	<.001	-.007176	-.004111

*. The mean difference is significant at the 0.05 level.

จากตารางที่ 36 ค่า Sig. คือความน่าจะเป็นในการยอมรับสมมติฐานของความแตกต่าง ค่า 95% Confidence Interval คือขอบเขตช่วงความเชื่อมั่น 95% ของค่าเฉลี่ยในแต่ละกลุ่ม เมื่อค่า Sig. ที่โปรแกรมคำนวณมีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 หมายถึงทั้งสองกลุ่มแตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05 ดังนั้นจากตารางที่ 36 จึงสรุปได้ว่าสำหรับการปรับระดับข้อมูลด้วย SMOTE อัลกอริทึมส่งผลแตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05

ส่วนการวัดประสิทธิภาพด้วย F1 score นั้นจะเห็นได้ว่าการจัดการข้อมูลด้วยวิธี Threshold Moving ให้ผลลัพธ์ดีที่สุดในแต่ละอัลกอริทึมซึ่งผลลัพธ์ของอัลกอริทึม XGBoost ยังใกล้เคียงกับ FOREST อีกด้วยผู้วิจัยจึงได้ทำการทดสอบอิทธิพลของอัลกอริทึมต่อการวิธีการจัดการข้อมูลด้วย Threshold Moving ด้วยการวิเคราะห์ความแปรปรวนแบบ 1 ตัวประกอบ ซึ่งแสดงดังตารางที่ 37

ตารางที่ 37 ผลการวิเคราะห์ความแปรปรวนของ F1 score สำหรับ Threshold Moving

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	1.271	2	.636	9188.728	<.001
Within Groups	.006	87	.000		
Total	1.277	89			

จากตารางที่ 37 ค่าความน่าจะเป็นในการยอมรับสมมติฐาน Sig. ที่โปรแกรมคำนวณมาให้คือน้อยกว่า 0.001 มีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 จึงสรุปผลได้ว่าผลของแต่ละอัลกอริทึมแตกต่างกันสำหรับการจัดการข้อมูลด้วย Threshold Moving ที่ระดับนัยยะสำคัญ 0.05 ผู้วิจัยจึงทำการเปรียบเทียบหลังการทดสอบรวมด้วยการทดสอบ Games-Howell และได้ผลดังตารางที่ 38

ตารางที่ 38 ผลการทดสอบ Games-Howell ของ F1 score สำหรับ Threshold Moving

(I) algorithm	(J) algorithm	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval Lower Bound	95% Confidence Interval Upper Bound
LOGIST	FOREST	-.2540372*	.0012253	<.001	-.257032	-.251042
	XGBoost	-.2501877*	.0023624	<.001	-.256004	-.244371
FOREST	LOGIST	.2540372*	.0012253	<.001	.251042	.257032
	XGBoost	.0038495	.0025989	.310	-.002462	.010161
XGBoost	LOGIST	.2501877*	.0023624	<.001	.244371	.256004
	FOREST	-.0038495	.0025989	.310	-.010161	.002462

*. The mean difference is significant at the 0.05 level.

จากตารางที่ 38 ค่า Sig. คือความน่าจะเป็นในการยอมรับสมมติฐานของความแตกต่าง ค่า 95% Confidence Interval คือขอบเขตช่วงความเชื่อมั่น 95% ของค่าเฉลี่ยในแต่ละกลุ่ม เมื่อค่า Sig. ที่โปรแกรมคำนวณมีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 หมายถึงทั้งสองกลุ่มแตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05 ดังนั้นจากตารางที่ 38 จึงสรุปได้ว่าสำหรับการจัดการข้อมูลด้วย Threshold Moving นั้น แต่ละอัลกอริทึมส่งผลแตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05 ยกเว้นอัลกอริทึม FOREST และอัลกอริทึม XGBoost ที่ส่งผลไม่แตกต่างกันอย่างมีนัยยะสำคัญทางสถิติที่ระดับ 0.05

นอกจากนี้หากวิเคราะห์เพิ่มเติมโดยการตรวจสอบผล Confusion Matrix ของการปรับระดับข้อมูลด้วย SMOTE สำหรับอัลกอริทึม FOREST ยังให้ผลใกล้เคียงกับผล Confusion Matrix ของการจัดการข้อมูลด้วย Threshold Moving สำหรับการวัดประสิทธิภาพด้วย G-Mean ผู้วิจัยจึงทำการเปรียบเทียบ independent-samples t-test สำหรับจำนวนข้อมูลที่ทำนายถูกต้องในกลุ่มน้อย (TN) ของทั้งสองวิธีการ โดยผล Confusion Matrix แสดงดังตารางที่ 39 และผลการเปรียบเทียบ independent-samples t-test แสดงดังตารางที่ 40

ตารางที่ 39 Confusion Matrix ของ SMOTE และ G-Mean

Variable	Mean	StDev
TP (SMOTE)	479567	1191
FN (SMOTE)	43448	1191
FP (SMOTE)	821.37	33.04
TN (SMOTE)	3116.6	33
TP (G-Mean)	468806	3290
FN (G-Mean)	54209	3290
FP (G-Mean)	583.87	42.83
TN (G-Mean)	3354.1	42.8

ตารางที่ 40 ผลการเปรียบเทียบ independent-samples t-test

	TN (G-Mean)	TN (SMOTE)
Mean	3354.133	3116.633
Variance	1834.464	1091.62
Observations	30	30
Hypothesized Mean Difference	0	
df	54	
t Stat	24.0481	
P(T<=t) one-tail	7.96E-31	
t Critical one-tail	1.673565	
P(T<=t) two-tail	1.59E-30	
t Critical two-tail	2.004879	

จากตารางที่ 40 ค่าความน่าจะเป็นในการยอมรับสมมติฐาน P(T<=t) two-tail ที่โปรแกรมคำนวณมาให้คือ 1.59E-30 มีค่าน้อยกว่า α ที่ผู้วิจัยกำหนดคือ 0.05 จึงสรุปผลได้ว่าการจัดการข้อมูลด้วย Threshold Moving สำหรับการวัดประสิทธิภาพด้วย G-Mean แตกต่างกับการปรับระดับข้อมูลด้วย SMOTE ที่ระดับนัยยะสำคัญ 0.05 ส่วนการจัดการข้อมูลด้วย Threshold Moving สำหรับการวัดประสิทธิภาพด้วย F1 score นั้นพบว่าวิธีการนี้ให้น้ำหนักของจำนวนข้อมูลที่ทำนายถูกต้องในกลุ่มน้อย (TN) น้อยกว่าการวัดประสิทธิภาพด้วย G-Mean ซึ่งแสดงดังตารางที่ 41

ตารางที่ 41 Confusion Matrix ของ F1 score

Variable	Mean	StDev
TP	520414	331
FN	2601.2	331.2
FP	2739	75.4
TN	1199	75.4

จากผลการดำเนินการที่กล่าวมาข้างต้น สรุปได้ว่าอิทธิพลร่วมของวิธีการจัดการข้อมูลและอัลกอริทึมนั้นส่งผลต่อการพยากรณ์ ในด้านการวัดประสิทธิภาพด้วย AUROC การปรับระดับข้อมูลส่งผลต่ออัลกอริทึมดังนี้ สำหรับอัลกอริทึม LOGIST การปรับระดับข้อมูลด้วย OSS ผสม SMOTE ให้

ผลลัพธ์ที่ดีที่สุด รองลงมาคือการปรับระดับข้อมูลด้วย SMOTE, OSS, ข้อมูลเดิม และ NearMiss-3 ตามลำดับ และสำหรับอัลกอริทึม FOREST ข้อมูลเดิมและการปรับระดับข้อมูลด้วย OSS ให้ผลลัพธ์ดีที่สุดและไม่แตกต่างกัน รองลงมาคือการปรับระดับด้วย SMOTE, OSS ผสม SMOTE และ NearMiss-3 ตามลำดับ ส่วนอัลกอริทึม XGBoost การปรับระดับข้อมูลด้วย SMOTE และ OSS ผสม SMOTE ให้ผลลัพธ์ดีที่สุดและไม่แตกต่างกัน รองลงมาคือการปรับระดับข้อมูลด้วย OSS และข้อมูลเดิมซึ่งให้ผลไม่แตกต่างกัน ส่วนการปรับระดับข้อมูลด้วย NearMiss-3 ให้ผลลัพธ์ที่แย่ที่สุด โดยผลลัพธ์ที่ดีที่สุดสำหรับ AUROC คืออัลกอริทึม FOREST ที่ไม่มีการปรับข้อมูล ซึ่งได้ค่าประมาณ 0.9460

ส่วนการวัดประสิทธิภาพด้วย F1 score การปรับระดับข้อมูลส่งผลต่ออัลกอริทึมดังนี้ สำหรับอัลกอริทึม LOGIST การจัดการข้อมูลด้วย Threshold Moving ให้ผลลัพธ์ดีที่สุด รองลงมาคือการปรับระดับข้อมูลด้วย SMOTE และ OSS ผสม SMOTE ซึ่งให้ผลลัพธ์ไม่แตกต่างกัน ถัดมาคือการปรับระดับข้อมูลด้วย NearMiss-3 ส่วนการปรับระดับข้อมูลด้วย OSS และข้อมูลเดิมให้ผลที่แย่ที่สุดและไม่แตกต่างกัน ในส่วนของอัลกอริทึม FOREST การจัดการข้อมูลด้วย Threshold Moving ให้ผลลัพธ์ดีที่สุด รองลงมาคือการปรับระดับข้อมูลด้วย SMOTE และ OSS ผสม SMOTE ตามลำดับ ถัดมาคือการปรับระดับข้อมูลด้วย OSS และ NearMiss-3 ซึ่งให้ผลไม่แตกต่างกัน ส่วนข้อมูลเดิมให้ผลลัพธ์แย่ที่สุด และสำหรับอัลกอริทึม XGBoost การจัดการข้อมูลด้วย Threshold Moving ให้ผลลัพธ์ดีที่สุด รองลงมาคือการปรับระดับข้อมูลด้วย SMOTE, OSS ผสม SMOTE, OSS, ข้อมูลเดิม และ NearMiss-3 ตามลำดับ โดยการจัดการข้อมูลด้วย Threshold Moving สำหรับอัลกอริทึม FOREST และ XGBoost ให้ผลไม่แตกต่างกัน ซึ่งมีค่าอยู่ที่ประมาณ 0.3098 ส่วนการจัดการข้อมูลด้วย Threshold Moving ด้วยการวัดประสิทธิภาพ G-Mean นั้นอัลกอริทึม FOREST ให้ผลลัพธ์ดีที่สุดซึ่งมีค่าอยู่ที่ประมาณ 0.8737 นอกจากนี้ยังเป็นวิธีที่เหมาะสมกับชุดข้อมูลในวิทยานิพนธ์อีกด้วย

บทที่ 5
สรุปผล

5.1 สรุปผล

การลดลงของอุปสรรคในการดำเนินการเรียนรู้ของเครื่องไม่ว่าจะเป็นด้านราคา การประมวลผลทำให้ปัจจุบันมีการนำมาใช้พยากรณ์สินค้าคงค้างเพื่อสร้างความพึงพอใจให้กับลูกค้าและลดค่าใช้จ่ายที่เกิดจากสินค้าคงค้าง แต่ด้วยลักษณะของข้อมูลที่มีความไม่สมดุลกันนั้นส่งผลเสียต่อการเรียนรู้ของเครื่อง เพื่อให้ได้ตัวแบบพยากรณ์ที่เหมาะสมจึงต้องศึกษาการจัดการปัญหานี้ไม่ว่าจะเป็นการจัดการกับข้อมูลก่อนผ่านการเรียนรู้ของเครื่องอย่างวิธีการปรับระดับข้อมูลซึ่งเป็นวิธีที่ได้รับความนิยม ทว่าวิธีนี้อาจลดข้อมูลที่เป็นประโยชน์หรือสร้างข้อมูลใหม่เป็นข้อมูลรบกวน จึงได้มีการศึกษาวิธีของ Thershold-Moving ด้วย ซึ่งผู้วิจัยได้สร้างตัวแบบผ่านทั้ง 2 วิธีนี้ด้วยอัลกอริทึมที่เหมาะสมกับข้อมูลไม่สมดุลอย่าง Random Forest, eXtreme Gradient Boosting และ Linear logistic regression และเพื่อให้ได้ผลลัพธ์ที่ดีที่สุดผู้วิจัยจึงได้มีการปรับค่าพารามิเตอร์ในแต่ละอัลกอริทึมและได้มีการตรวจสอบแบบไขว้เพื่อให้ตัวแบบที่สร้างสามารถใช้ได้กับข้อมูลจริง สำหรับวิธีการปรับระดับข้อมูลได้ใช้ตัววัดประสิทธิภาพ AUROC ซึ่งได้รับความนิยมในการทบทวนวรรณกรรมที่เกี่ยวกับการพยากรณ์สินค้าคงค้าง เป็นวิธีที่ให้ความสำคัญกับทั้งข้อมูลกลุ่มมากและกลุ่มน้อย ค่าที่ได้จะเป็นการบอกภาพรวมของตัวแบบ โดยสำหรับวิทยานิพนธ์ฉบับนี้ผลลัพธ์ที่ดีที่สุดสำหรับ AUROC คืออัลกอริทึม FOREST ที่ไม่มีการปรับข้อมูล ซึ่งได้ค่าประมาณ 0.9460 รองลงมาคือการปรับเพิ่มระดับข้อมูลด้วย SMOTE ซึ่งได้ค่าประมาณ 0.9384 ทว่าวิธีการที่ไม่ปรับข้อมูลนั้นพยากรณ์ข้อมูลกลุ่มมากเป็นส่วนใหญ่ ซึ่งการที่มีค่า AUROC ที่สูงนั้นอาจมาจากการที่ภาพรวมของตัวแบบทั้งหมดดี แต่จุดเกณฑ์ที่ใช้แบ่งข้อมูลยังไม่เหมาะสม นอกจากนี้ยังใช้ตัวชี้วัด F1 score กับการปรับระดับข้อมูลอีกด้วย โดยตัววัดประสิทธิภาพนี้จะให้ความสำคัญกับข้อมูลกลุ่มน้อยมากกว่าและให้น้ำหนักของการพยากรณ์ผิดพลาดของทั้งสองกลุ่มเท่ากัน ซึ่งผลลัพธ์ F1 score ที่ได้จากการปรับระดับข้อมูลนั้นน้อยกว่าวิธีของ Thershold-Moving ซึ่งมีค่าอยู่ที่ประมาณ 0.3098 ซึ่งเมื่อดูจากจากการพยากรณ์ข้อมูลกลุ่มน้อยแล้วพบว่า F1 score ให้น้ำหนักกับข้อมูลกลุ่มน้อยน้อยกว่าการวัดประสิทธิภาพ G-mean สำหรับวิธีของ Thershold-Moving ซึ่งตัววัดประสิทธิภาพนี้ให้ความสำคัญกับข้อมูลกลุ่มมากและกลุ่มน้อยเท่ากันเช่นเดียวกับ AUROC และยังเหมาะกับข้อมูลที่ไม่สมดุล โดยผล G-mean ที่ดีที่สุดมาจากอัลกอริทึม FOREST ซึ่งมีค่าอยู่ที่ประมาณ 0.8737 และยังเป็นตัวแบบที่เหมาะสมกับวิทยานิพนธ์ฉบับนี้ ทั้งนี้ตัวแบบที่ได้ยังสามารถใช้พยากรณ์การเกิดสินค้าคงค้างได้ในอนาคตเนื่องจากผลลัพธ์ไม่เกิด Overfitting ระหว่างข้อมูลฝึกกับทดสอบ และยังสังเกตได้ว่าการให้ความสำคัญของ AUROC และ G-mean มีลักษณะคล้ายคลึงกันดังนั้นผล Confusion Matrix ของทั้งสองจึงต่างกันเล็กน้อย

ส่วนการเปรียบเทียบผลลัพธ์กับงานวิจัยที่เกี่ยวกับการพยากรณ์สินค้าคงค้างอื่นนั้นพบว่าผลลัพธ์ AUROC ที่ดีที่สุดของ de Santis et al. (2017) อยู่ที่ประมาณ 0.9482 ซึ่งมากกว่าวิทยานิพนธ์ฉบับนี้ อาจเป็นเพราะมีการแบ่งข้อมูลสำหรับฝึกที่มากกว่า หรือการปรับค่าพารามิเตอร์บางตัวที่แตกต่างกัน อีกทั้งตัวแบบที่ดีที่สุดของวิทยานิพนธ์ฉบับนี้อยู่ในรูปแบบค่า G-mean จึงไม่สามารถสรุปได้ชัดเจนนัก แต่เมื่อเทียบกับงานวิจัยอื่นพบว่าวิทยานิพนธ์ฉบับนี้ให้ผลลัพธ์ AUROC ที่ดีกว่าไม่ว่าจะเป็นการปรับระดับข้อมูลด้วย SMOTE หรือวิธี SMOTE ผสม OSS โดยผลลัพธ์ AUROC ที่ดีที่สุดของ Hajek and Abedin (2020) อยู่ที่ประมาณ 0.9157 แต่ค่าที่ได้จากงานวิจัยนี้มีการสมมติค่าใช้จ่ายส่วนอื่นขึ้นมาส่งผลให้อาจมีการให้น้ำหนักของข้อมูลที่แตกต่างกัน ส่วนผลลัพธ์ AUROC ที่ดีที่สุดของMalviya et al. (2021) อยู่ที่ประมาณ 0.9000 นอกจากนี้ยังสังเกตได้ว่าสำหรับชุดข้อมูลที่มีความไม่สมดุลมาก การปรับเพิ่มข้อมูลไม่ว่าจะเป็น SMOTE หรือ OSS ผสม SMOTE นั้นให้ผลลัพธ์ที่ดีกว่าการปรับลดข้อมูลอย่าง OSS และ NearMiss-3 ซึ่งสอดคล้องกับงานวิจัยของ de Santis et al. (2017) และวิจัยของ Yu et al. (2015) เนื่องจากการปรับลดข้อมูลทำให้เสียข้อมูลที่สำคัญสำหรับการฝึกข้อมูลเป็นจำนวนมาก นอกจากนี้ยังพบว่ากลุ่ม Ensemble Learning ให้ผลลัพธ์ดีกว่า LOGIST ซึ่งสอดคล้องกับงานวิจัยของ Malviya et al. (2021)

5.2 ข้อจำกัด

แม้ว่าปัจจุบันมีการเข้าถึงการเรียนรู้ของเครื่องได้มากขึ้นทั้งด้านของการประมวลผล และราคาที่ไม่แพงนัก แต่การรับมือกับชุดข้อมูลที่มีขนาดใหญ่ยังคงเป็นอุปสรรคในบ้างด้าน ทั้งด้านของเวลาที่ใช้ฝึกข้อมูลค่อนข้างนานในบางการจัดการกับข้อมูล ส่งผลให้ต้องชั่งน้ำหนักในด้านของค่าใช้จ่ายกับผลลัพธ์ที่ได้เช่น ข้อมูลฝึกที่มีขนาดใหญ่ มีการฝึกซ้ำของข้อมูลมาก หรือวิธีที่ใช้มีความซับซ้อนมากจะให้ผลลัพธ์ที่ดีกว่าแต่แลกกับเวลาในการฝึกที่นาน

5.3 งานในอนาคต

แม้ว่าการจัดการข้อมูลด้วย Threshold Moving จะสามารถพยากรณ์ข้อมูลกลุ่มน้อยได้ดีแต่ก็แลกกับการพยากรณ์ข้อมูลกลุ่มมากเป็นข้อมูลกลุ่มน้อยมากขึ้น ซึ่งก็เป็นเรื่องที่ควรให้เกิดขึ้นน้อย เพราะการพยากรณ์ว่าเป็นสินค้าคงค้างทั้งที่ไม่ได้เป็น (FN) ส่งผลต่อ ค่าใช้จ่ายคงคลัง อาจจะใช้วิธีการอื่นในการจัดการกับข้อมูลไม่สมดุล เช่น Cost-Sensitive รวมถึง Neural Network แม้ว่างานวิจัยจากการทบทวนวรรณกรรมที่เกี่ยวข้อง Neural Network จะมีประสิทธิภาพไม่ดีนักกับข้อมูลที่ไม่สมดุลแต่ Deep Neural Network อาจจะทำให้การพยากรณ์มีประสิทธิภาพมากขึ้น นอกจากนี้อัลกอริทึมกลุ่ม Boosting อื่นนอกเหนือจาก XGBoost ก็ให้ผลลัพธ์ที่ดีกับข้อมูลไม่สมดุลเช่นเดียวกัน

บรรณานุกรม

Bevans, R. (2022). *Two-way ANOVA | When and How to Use it, With Examples*. Retrieved 15 Jun from https://www.scribbr.com/statistics/two-way-anova/

Bhandari, A. (2022). *AUC-ROC Curve in Machine Learning Clearly Explained*. Retrieved 10 Jun from https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/

Brownlee, J. (2018). *A Gentle Introduction to k-fold Cross-Validation*. Retrieved 15 Jun from https://machinelearningmastery.com/k-fold-cross-validation/

Brownlee, J. (2020a). *Tour of Evaluation Metrics for Imbalanced Classification*. Retrieved 27 Aug from https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/

Brownlee, J. (2020b). *Undersampling Algorithms for Imbalanced Classification*. Retrieved 2 Feb from https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/

Brownlee, J. (2021). *A Gentle Introduction to Threshold-Moving for Imbalanced Classification*. Retrieved 15 Jun from https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/

Collell, G., Prelec, D., & Patil, K. R. (2018). A simple plug-in bagging ensemble based on threshold-moving for classifying binary and multiclass imbalanced data. *Neurocomputing*, 275, 330-340.

Daroontham, W. (2561). เจาะลึก *Random Forest !!!— Part 2 of "*รู้จัก *Decision Tree, Random Forest,* และ *XGBoost!!!* Retrieved 10 Jun from https://medium.com/@witchapongdaroontham/%E0%B9%80%E0%B8%88%E0%B8%B2%E0%B8%B0%E0%B8%A5%E0%B8%B6%E0%B8%81-random-forest-part-2-of-%E0%B8%A3%E0%B8%B9%E0%B9%89%E0%B8%88%E0%B8%B1%E0%B8%81-decision-tree-random-forest-%E0%B9%81%E0%B8%A5%E0%B8%B0-xgboost-79b9f41a1c1c

de Santis, R. B., de Aguiar, E. P., & Goliatt, L. (2017). Predicting material backorders in inventory management using machine learning. 2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI),

Geller, S. (2019). *Normalization vs Standardization — Quantitative analysis*. Retrieved 15 Jun from https://towardsdatascience.com/normalization-vs-standardization-quantitative-analysis-a91e8a79cebf

Hajek, P., & Abedin, M. Z. (2020). A profit function-maximizing inventory backorder prediction system using big data analytics. *IEEE Access*, 8, 58982-58994.

Hubert, M., & Vandervieren, E. (2008). An adjusted boxplot for skewed distributions. *Computational statistics & data analysis*, 52(12), 5186-5201.

IBM. (n.d.). *What is logistic regression?* Retrieved 10 June from https://www.ibm.com/topics/logistic-regression

Islam, S., & Amin, S. H. (2020). Prediction of probable backorder scenarios in the supply chain using Distributed Random Forest and Gradient Boosting Machine learning techniques. *Journal of Big Data*, 7(1), 1-22.

Korstanje, J. (2021). *The F1 score*. Retrieved 10 Jun from https://towardsdatascience.com/the-f1-score-bec2bbc38aa6

Liu, W., Chen, Z., & Hu, Y. (2022). XGBoost algorithm-based prediction of safety assessment for pipelines. *International Journal of Pressure Vessels and Piping*, 197, 104655. https://doi.org/https://doi.org/10.1016/j.ijpvp.2022.104655

Lund, A. L. M. (n.d.). *Two-way ANOVA in SPSS Statistics*. Retrieved 1 Jun from https://statistics.laerd.com/spss-tutorials/two-way-anova-using-spss-statistics.php

Malviya, L., Chittora, P., Chakrabarti, P., Vyas, R. S., & Poddar, S. (2021). Backorder prediction in the supply chain using machine learning. *Materials Today: Proceedings*. https://doi.org/https://doi.org/10.1016/j.matpr.2020.11.558

Morde, V. (2019). *XGBoost Algorithm: Long May She Reign!* Retrieved 10 Jun from https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

Narkhede, S. (2018). *Understanding AUC - ROC Curve*. Retrieved 10 Jun from https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Pornprasertmanit, S. (2015). *Introduction (and Intermediate) Statistics in Psychology*.

Retrieved 1 Jun from https://sunthud.com/media/Teaching/IntroStat/Fall15/Ch08_OneWayANOVA.pdf

Rosen, D. B. (2020). *How To Deal With Imbalanced Classification, Without Re-balancing the Data*. Retrieved 15 Jun from https://towardsdatascience.com/how-to-deal-with-imbalanced-classification-without-re-balancing-the-data-8a3c02353fe3

RPG, B. B. (2561). *Confusion Matrix | Recall | Precision | Accuracy* เห้ยยมันคืออะไรว้าาา !? Retrieved 2 Feb from https://medium.com/bigdataeng/confusion-matrix-recall-precision-accuracy-%E0%B9%80%E0%B8%AB%E0%B9%89%E0%B8%A2%E0%B8%A2%E0%B8%A1%E0%B8%B1%E0%B8%99%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3%E0%B8%A7%E0%B9%89%E0%B8%B2%E0%B8%B2%E0%B8%B2-fc3e1fcfc579

Sarakarn, P., & Munpolsri, P. (2021). OPTIMAL CUT-OFF POINTS FOR RECEIVER OPERATING CHARACTERISTIC (ROC) CURVE ANALYSIS IN DEVELOPING TOOLS OF HEALTH INNOVATIONS: EXAMPLE USING STATA. *Thai Bulletin of Pharmaceutical Sciences*, 16(1), 93-108.

SATPATHY, S. (2021). *Overcoming Class Imbalance using SMOTE Techniques*. Retrieved 10 Jun from https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/

Schlegel, A. (2020). *Games-Howell Post-Hoc Multiple Comparisons Test with Python*. Retrieved 1 Jun from https://aaronschlegel.me/games-howell-post-hoc-multiple-comparisons-test-python.html

stackpython. (2020). *Machine Learning Ep.2 : Cross Validation*. Retrieved 15 Jun from https://stackpython.medium.com/machine-learning-ep-2-cross-validation-70cbefb2dda4

Team, G. L. (2023). *Hyperparameter Tuning with GridSearchCV*. Retrieved Jun 2 from https://www.mygreatlearning.com/blog/gridsearchcv/

TITIPATA. (n.d.). *[ML] Bagging หรือ Boosting คืออะไร ทำงานอย่างไร?* Retrieved 2 Feb from https://tupleblog.netlify.app/bagging-boosting/

YAĞCI, H. E. (2021). *Under-Sampling Methods for Imbalanced Data (ClusterCentroids, RandomUnderSampler, NearMiss)*. Retrieved 10 Jun from

https://hersanyagci.medium.com/under-sampling-methods-for-imbalanced-data-clustercentroids-randomundersampler-nearmiss-eae0eadcc145

Yiu, T. (2019). *Understanding Random Forest*. Retrieved 10 Jun from https://towardsdatascience.com/understanding-random-forest-58381e0602d2

Yu, H., Mu, C., Sun, C., Yang, W., Yang, X., & Zuo, X. (2015). Support vector machine-based optimized decision threshold adjustment strategy for classifying imbalanced data. *Knowledge-Based Systems*, 76, 67-78.

ZINJAD, S. (2021). *Backorder Dataset*. Retrieved 15 Mar from https://www.kaggle.com/datasets/ztrimus/backorder-dataset

ชิตพงษ์ กิตตินราดร. (2563). *Machine Learning* บทที่ 11: *Boosting*. Retrieved 2 Feb from https://guopai.github.io/ml-blog11.html

ทินกร ม้าลายทอง. (2564). เพิ่มประสิทธิภาพของ *Machine Learning Model* ด้วย *Hyperparameter Optimization*. Retrieved May 7 from https://bigdata.go.th/big-data-101/machine-learning-model-hyperparameter-optimization/

พุทธิพร ธนธรรมเมธี, & เยาวเรศ ศิริสถิตย์กุล. (2562). เทคนิคการจำแนกข้อมูลที่พัฒนาสำหรับชุดข้อมูลที่ไม่สมดุลของภาวะข้อเข่าเสื่อมในผู้สูงอายุ. *Thai Science and Technology Journal*, 1164-1178.

ยุทธ ไกยวรรณ์. (2555). หลักการและการใช้การวิเคราะห์การถดถอยโลจิสติคสำหรับการวิจัย. วารสารวิจัยมหาวิทยาลัยเทคโนโลยีราชมงคลศรีวิชัย, 4(1), 1-12.

สุพัฒน์ สุกมลสันต์. (2560). การเปรียบเทียบก่อนและ หลังการทดสอบรวมเพื่อ การวิจัย. วารสารวิชาการมหาวิทยาลัยราชภัฏบุรีรัมย์มนุษยศาสตร์และสังคมศาสตร์, 9(2), 51-70.

บทที่ 6
ภาคผนวก

```
#เตรียมข้อมูล
import numpy as np
import pandas as pd
cols=range(1,24)
path1 = "/content/drive/MyDrive/data/data/datamc10drop.csv"
df = pd.read_csv(path1,usecols=cols)
from sklearn.preprocessing import StandardScaler
qty_related = ['national_inv', 'in_transit_qty','lead_time', 'forecast_3_month',
               'forecast_6_month', 'forecast_9_month', 'min_bank',
               'local_bo_qty', 'pieces_past_due', 'sales_1_month',
               'sales_3_month', 'sales_6_month', 'sales_9_month',]
df[qty_related] = StandardScaler().fit_transform(df[qty_related])
df = np.clip(df, -5, 5)

#แบ่งชุดข้อมูลสำหรับฝึกและทดสอบ วัดผลด้วย AUROC : scoring='roc_auc',วัดผลด้วย F1 score
: scoring='f1'
from sklearn.model_selection import train_test_split
X = df.drop(['went_on_backorder','sku'],axis=1).values
y = df['went_on_backorder'].values
X_train,X_test,y_train,y_test = train_test_split(X,y,stratify=y,
test_size=0.3,train_size=0.7)

#สำหรับ LogisticRegression
from sklearn.linear_model import LogisticRegression
from numpy import mean
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

#สำหรับข้อมูลเดิมใช้โค้ดนี้ก่อนคำสั่ง grid_search.fit(X_train, y_train)
param_grid = {'C':[ 0.001, 0.01, 0.1,1, 10, 100, 1000]}
clf = LogisticRegression(max_iter=1000)
cv = StratifiedKFold(n_splits=5)
grid_search = GridSearchCV(estimator = clf, param_grid = param_grid,
scoring='roc_auc',
                    cv = cv , n_jobs = -1)

#สำหรับการปรับระดับข้อมูลด้วยNearMiss ใช้โค้ดนี้ก่อนคำสั่ง grid_search.fit(X_train, y_train)
from imblearn.under_sampling import NearMiss
from imblearn.pipeline import Pipeline

model = LogisticRegression(max_iter=1000)
under = NearMiss(version=3, n_neighbors_ver3=3)
steps = [ ('u', under), ('m', model)]
pipeline = Pipeline(steps=steps)
cv = StratifiedKFold(n_splits=5)
param_grid = {'m__C': [ 0.001, 0.01, 0.1,1, 10, 100, 1000]}
grid_search = GridSearchCV(estimator = pipeline, param_grid = param_grid,
scoring='roc_auc',
                    cv = cv , n_jobs = -1)

#สำหรับการปรับระดับข้อมูลด้วย OSS ใช้โค้ดนี้ก่อนคำสั่ง grid_search.fit(X_train, y_train)
from imblearn.under_sampling import OneSidedSelection
from imblearn.pipeline import Pipeline

model = LogisticRegression(max_iter=1000)
```

```
under = OneSidedSelection(n_neighbors=1, n_seeds_S=500)
steps = [ ('u', under), ('m', model)]
pipeline = Pipeline(steps=steps)
cv = StratifiedKFold(n_splits=5)
param_grid = {'m__C': [ 0.001, 0.01, 0.1,1, 10, 100, 1000]}
grid_search = GridSearchCV(estimator = pipeline, param_grid = param_grid,
scoring='f1',
                        cv = cv , n_jobs = -1)
```

#สำหรับการปรับระดับข้อมูลด้วย OSS ผสม SMOTE ใช้โค้ดนี้ก่อนคำสั่ง grid_search.fit(X_train, y_train)

```
from imblearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import OneSidedSelection

model = LogisticRegression(max_iter=1000)
over = SMOTE()
under = OneSidedSelection(n_neighbors=1, n_seeds_S=500)
steps = [ ('u', under),('o', over), ('m', model)]
pipeline = Pipeline(steps=steps)
cv = StratifiedKFold(n_splits=5)
param_grid = {'m__C': [ 0.001, 0.01, 0.1,1, 10, 100, 1000]}
grid_search = GridSearchCV(estimator = pipeline, param_grid = param_grid,
scoring='f1',
                        cv = cv , n_jobs = -1)
```

#สำหรับการปรับระดับข้อมูลด้วย SMOTE ใช้โค้ดนี้ก่อนคำสั่ง grid_search.fit(X_train, y_train)

```
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline
```

```
param_grid = {'m__C': [ 0.001, 0.01, 0.1,1, 10, 100, 1000]}
model= LogisticRegression(max_iter=1000)
cv = StratifiedKFold(n_splits=5)
over = SMOTE()
steps = [ ('o', over), ('m', model)]
pipeline = Pipeline(steps=steps)
grid_search = GridSearchCV(estimator = pipeline, param_grid = param_grid,
scoring='roc_auc',
                           cv = cv , n_jobs = -1)

grid_search.fit(X_train, y_train)
cv_score = grid_search.best_score_
test_score = grid_search.score(X_test, y_test)
print(f'Cross-validation score: {cv_score}\nTest score: {test_score}')

grid_search.best_params_
best_grid = grid_search.best_estimator_
print(best_grid)

from sklearn.metrics import confusion_matrix
y_pred = grid_search.best_estimator_.predict(X_test)
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
grid_search.cv_results_

#สำหรับ RandomForest
from sklearn.ensemble import RandomForestClassifier
from numpy import mean
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

#สำหรับข้อมูลเดิมใช้โค้ดนี้ก่อนคำสั่ง grid_search.fit(X_train, y_train)
param_grid = {
    'max_depth': [12,9,7],
    'min_samples_leaf': [3, 4, 5]}
clf = RandomForestClassifier(n_estimators=10)
cv = StratifiedKFold(n_splits=5)
grid_search = GridSearchCV(estimator = clf, param_grid = param_grid,
scoring='roc_auc',
                           cv = cv , n_jobs = -1)

#สำหรับการปรับระดับข้อมูลด้วยNearMiss ใช้โค้ดนี้ก่อนคำสั่ง grid_search.fit(X_train, y_train)
from imblearn.pipeline import Pipeline
from imblearn.under_sampling import NearMiss

model = RandomForestClassifier(n_estimators=10)
under = NearMiss(version=3, n_neighbors_ver3=3)
steps = [('u', under), ('m', model)]
pipeline = Pipeline(steps=steps)
cv = StratifiedKFold(n_splits=5)
param_grid = {
    'm__max_depth': [7,9,12],
    'm__min_samples_leaf':  [3, 4,5]}
grid_search = GridSearchCV(estimator = pipeline, param_grid = param_grid,
scoring='roc_auc',
                           cv = cv , n_jobs = -1)

#สำหรับการปรับระดับข้อมูลด้วย OSS ใช้โค้ดนี้ก่อนคำสั่ง grid_search.fit(X_train, y_train)
```

```
from imblearn.under_sampling import OneSidedSelection
from imblearn.pipeline import Pipeline

model = RandomForestClassifier(n_estimators=10)
under = OneSidedSelection(n_neighbors=1, n_seeds_S=500)
steps = [('u', under), ('m', model)]
pipeline = Pipeline(steps=steps)
cv = StratifiedKFold(n_splits=5)
param_grid = {
    'm__max_depth': [7,9,12],
    'm__min_samples_leaf':  [3, 4,5]}
grid_search = GridSearchCV(estimator = pipeline, param_grid = param_grid,
scoring='f1',
                           cv = cv , n_jobs = -1)

#สำหรับการปรับระดับข้อมูลด้วย OSS ผสม SMOTE ใช้โค้ดนี้ก่อนคำสั่ง grid_search.fit(X_train,
y_train)
from imblearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import OneSidedSelection
```



```
model = RandomForestClassifier(n_estimators=10)
over = SMOTE()
under = OneSidedSelection(n_neighbors=1, n_seeds_S=500)
steps = [('u', under),('o', over),  ('m', model)]
pipeline = Pipeline(steps=steps)
cv = StratifiedKFold(n_splits=5)
param_grid = {
    'm__max_depth': [7,9,12],
    'm__min_samples_leaf':  [3,4,5]}
```

```
grid_search = GridSearchCV(estimator = pipeline, param_grid = param_grid,
scoring='f1',
                           cv = cv , n_jobs = -1)

#สำหรับการปรับระดับข้อมูลด้วย SMOTE ใช้โค้ดนี้ก่อนคำสั่ง grid_search.fit(X_train, y_train)
from imblearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE
model = RandomForestClassifier(n_estimators=10)
# define pipeline
over = SMOTE()
steps = [('o', over),  ('m', model)]
pipeline = Pipeline(steps=steps)
# define evaluation procedure
cv = StratifiedKFold(n_splits=5)
param_grid = {
    'm__max_depth': [7,9,12],
    'm__min_samples_leaf':  [3, 4,5]}
# evaluate model
grid_search = GridSearchCV(estimator = pipeline, param_grid = param_grid,
scoring='roc_auc',
                           cv = cv , n_jobs = -1)

grid_search.fit(X_train, y_train)
cv_score = grid_search.best_score_
test_score = grid_search.score(X_test, y_test)
print(f'Cross-validation score: {cv_score}\nTest score: {test_score}')

grid_search.best_params_
best_grid = grid_search.best_estimator_
print(best_grid)
```

```
from sklearn.metrics import confusion_matrix
y_pred = grid_search.best_estimator_.predict(X_test)
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
grid_search.cv_results_

#สำหรับ XGBoost
from xgboost import XGBClassifier
from numpy import mean
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

#สำหรับข้อมูลเดิมใช้โค้ดนี้ก่อนคำสั่ง grid_search.fit(X_train, y_train)
param_grid = {
        'min_child_weight': [1, 5, 10],
        'max_depth': [7,9,12]}
clf = XGBClassifier(learning_rate=0.1, n_estimators=10,
objective='binary:logistic',nthread=1)
cv = StratifiedKFold(n_splits=5)
grid_search = GridSearchCV(estimator = clf, param_grid = param_grid,
scoring='roc_auc',
                        cv = cv , n_jobs = -1)

#สำหรับการปรับระดับข้อมูลด้วยNearMiss ใช้โค้ดนี้ก่อนคำสั่ง grid_search.fit(X_train, y_train)
from imblearn.pipeline import Pipeline
from imblearn.under_sampling import NearMiss
```

```
model = XGBClassifier(learning_rate=0.1, n_estimators=10,
objective='binary:logistic',nthread=1)
under = NearMiss(version=3, n_neighbors_ver3=3)
steps = [ ('u', under), ('m', model)]
pipeline = Pipeline(steps=steps)
cv = StratifiedKFold(n_splits=5)
param_grid = {
        'm__min_child_weight': [1,5,10],
        'm__max_depth': [7,9,12]}
grid_search = GridSearchCV(estimator = pipeline, param_grid = param_grid,
scoring='roc_auc',
                        cv = cv , n_jobs = -1)

#สำหรับการปรับระดับข้อมูลด้วย OSS ใช้โค้ดนี้ก่อนคำสั่ง grid_search.fit(X_train, y_train)
from imblearn.under_sampling import OneSidedSelection
from imblearn.pipeline import Pipeline

model = XGBClassifier(learning_rate=0.1, n_estimators=10,
objective='binary:logistic',nthread=1)
under = OneSidedSelection(n_neighbors=1, n_seeds_S=500)
steps = [ ('u', under), ('m', model)]
pipeline = Pipeline(steps=steps)
cv = StratifiedKFold(n_splits=5)
param_grid = {
        'm__min_child_weight': [1,5,10],
        'm__max_depth': [7,9,12]}
grid_search = GridSearchCV(estimator = pipeline, param_grid = param_grid,
scoring='f1',
                        cv = cv , n_jobs = -1)
```

```
#สำหรับการปรับระดับข้อมูลด้วย OSS ผสม SMOTE ใช้โค้ดนี้ก่อนคำสั่ง grid_search.fit(X_train, y_train)
from imblearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import OneSidedSelection

model = XGBClassifier(learning_rate=0.1, n_estimators=10,
objective='binary:logistic',nthread=1)
over = SMOTE()
under = OneSidedSelection(n_neighbors=1, n_seeds_S=500)
steps = [ ('u', under),('o', over), ('m', model)]
pipeline = Pipeline(steps=steps)
cv = StratifiedKFold(n_splits=5)
param_grid = {
        'm__min_child_weight': [1, 5, 10],
        'm__max_depth': [7,9,12]}
grid_search = GridSearchCV(estimator = pipeline, param_grid = param_grid,
scoring='f1',
                        cv = cv , n_jobs = -1)
```



```
#สำหรับการปรับระดับข้อมูลด้วย SMOTE ใช้โค้ดนี้ก่อนคำสั่ง grid_search.fit(X_train, y_train)
from imblearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE
model = XGBClassifier(learning_rate=0.1, n_estimators=10,
objective='binary:logistic',nthread=1)
over = SMOTE()
steps = [ ('o', over), ('m', model)]
pipeline = Pipeline(steps=steps)
cv = StratifiedKFold(n_splits=5)
param_grid = {
        'm__min_child_weight': [1, 5, 10],
```

```
        'm__max_depth': [7,9,12]}
grid_search = GridSearchCV(estimator = pipeline, param_grid = param_grid,
scoring='roc_auc',
                           cv = cv , n_jobs = -1)
grid_search.fit(X_train, y_train)
cv_score = grid_search.best_score_
test_score = grid_search.score(X_test, y_test)
print(f'Cross-validation score: {cv_score}\nTest score: {test_score}')

grid_search.best_params_
best_grid = grid_search.best_estimator_
print(best_grid)

from sklearn.metrics import confusion_matrix
y_pred = grid_search.best_estimator_.predict(X_test)
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
grid_search.cv_results_

#Threshold Moving หลังจากฝึกข้อมูลด้วยชุดข้อมูลเดิม
# roc curve for model
from sklearn.datasets import make_classification
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot

yhat = grid_search.best_estimator_.predict_proba(X_test)
yhat = yhat[:, 1]
fpr, tpr, thresholds = roc_curve(y_test,yhat)
pyplot.plot([0,1], [0,1], linestyle='--', label='No Skill')
```

```
pyplot.plot(fpr, tpr, marker='.', label='Model')
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.legend()
pyplot.show()

from numpy import sqrt
from numpy import argmax
gmeans = sqrt(tpr * (1-fpr))
ix = argmax(gmeans)
print('Best Threshold=%f, G-Mean=%.4f' % (thresholds[ix], gmeans[ix]))
pyplot.plot([0,1], [0,1], linestyle='--', label='No Skill')
pyplot.plot(fpr, tpr, marker='.', label='Model')
pyplot.scatter(fpr[ix], tpr[ix], marker='o', color='black', label='Best')
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.legend()
pyplot.show()

from sklearn.metrics import f1_score
thresholds = arange(0, 1, 0.001)
def to_labels(pos_probs, threshold):
  return (pos_probs >= threshold).astype('int')
yhat = grid_search.best_estimator_.predict_proba(X_test)
probs = yhat[:, 1]
scores = [f1_score(y_test, to_labels(probs, t)) for t in thresholds]
ix = argmax(scores)
print('Threshold=%.5f, F-Score=%.5f' % (thresholds[ix], scores[ix]))
```

ประวัติผู้เขียน

ชื่อ-สกุล	ธิรดา ธาดาจิรสกุล
วัน เดือน ปี เกิด	25 มกราคม 2541
สถานที่เกิด	สมุทรปราการ,ประเทศไทย
วุฒิการศึกษา	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ที่อยู่ปัจจุบัน	99/1003 หมู่บ้านทรัพย์ดินทอง ซอยเทพารักษ์66 ถนนเทพารักษ์ ตำบลบางเมือง อำเภอเมือง จังหวัดสมุทรปราการ ประเทศไทย