การเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนานสำหรับข้อมูลไม่สมดุล กรณีศึกษาข้อมูลเครดิตเยอรมัน

น.ส.ศศิวิมล ศรีโรจน์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาสถิติ ภาควิชาสถิติ
คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2565

BAGGING HETEROGENEOUS ENSEMBLE LEARNING FOR IMBALANCED DATA: A CASE STUDY OF GERMAN CREDIT DATA

Miss Sasivimol Sriroj

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Statistics
Department of Statistics
FACULTY OF COMMERCE AND ACCOUNTANCY
Chulalongkorn University
Academic Year 2022

หัวข้อวิทยานิพนธ์	การเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนานสำหรับข้อมูลไม่สมดุล กรณีศึกษาข้อมูลเครดิตเยอรมัน
โดย	น.ส.ศศิวิมล ศรีโรจน์
สาขาวิชา	สถิติ
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร.ณัตติฤดี เจริญรักษ์

คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

..... คณบดีคณะพาณิชยศาสตร์และการบัญชี

(ศาสตราจารย์ ดร.วิเลิศ ภูริวัชร)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.อัครินทร์ ไพบูลย์พานิช)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(ผู้ช่วยศาสตราจารย์ ดร.ณัตติฤดี เจริญรักษ์)

..... กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.นัท กุลวานิช)

..... กรรมการภายนอกมหาวิทยาลัย

(รองศาสตราจารย์ ดร.สุพล ดุรงค์วัฒนา)

ศศิวิมล ศรีโรจน์ : การเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนานสำหรับข้อมูลไม่สมดุล กรณีศึกษาข้อมูลเครดิตเยอรมัน. (BAGGING HETEROGENEOUS ENSEMBLE LEARNING FOR IMBALANCED DATA: A CASE STUDY OF GERMAN CREDIT DATA) อ.ที่ปรึกษาหลัก : ผศ. ดร.ณัตติฤดี เจริญรักษ์

งานวิจัยนี้มีวัตถุประสงค์เพื่อสร้างตัวแบบการเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนาน (Bagging Heterogeneous Ensemble) และหาวิธีการลดมิติข้อมูลและวิธีการสุ่มตัวอย่างซ้ำที่เหมาะสมกับข้อมูลเครดิตเยอรมันที่มีอัตราส่วนความไม่สมดุลแตกต่างกัน 3 ค่าคือ 2.3, 10 และ 14 โดยวัดประสิทธิภาพด้วยตัวชี้วัด Accuracy, The area under the curve, F1-score, Precision, Brier score และ Kolmogorov-Smirnov และทดสอบทางสถิติเพื่อแสดงว่าประสิทธิภาพของตัวแบบมีความแตกต่างกัน ที่ระดับนัยสำคัญ 0.05 ผลการศึกษาพบว่าข้อมูลเครดิตเยอรมันที่มีอัตราส่วนความไม่สมดุลต่ำ (IR = 2.3) ตัวแบบ Logistic Regression ที่ใช้เทคนิค Linear Discriminant Analysis (LDA) และ Systematic Minority Over-Sampling Technique (SM) จะมีประสิทธิภาพเฉลี่ยดีที่สุดในการจำแนกประเภท ในส่วนของอัตราส่วนความไม่สมดุลกลาง (IR = 10) และ อัตราส่วนความไม่สมดุลสูง (IR = 14) วิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำที่มีประสิทธิภาพคือ Linear Discriminant Analysis (LDA), Random Under-Sampling (RUS) และ Linear Discriminant Analysis (LDA), Borderline SMOTE (BSM) ตามลำดับ โดยที่การเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนานมีประสิทธิภาพเฉลี่ยดีที่สุด ทั้งในกรณีที่มีและไม่มีวิธีการลดมิติข้อมูลและสุ่มตัวอย่างซ้ำของอัตราส่วนความไม่สมดุลกลางและสูง

สาขาวิชา สถิติ ลายมือชื่อนิสิต .....

ปีการศึกษา 2565 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6480493826 : MAJOR STATISTICS
KEYWORD:

Sasivimol Sriroj : BAGGING HETEROGENEOUS ENSEMBLE LEARNING FOR IMBALANCED DATA: A CASE STUDY OF GERMAN CREDIT DATA. Advisor: Asst. Prof. NUTTIRUDEE CHAROENRUK, Ph.D.

The objective of this study is to develop a bagging heterogeneous ensemble and identify appropriate dimensionality reduction and resampling techniques for three different imbalance ratios (2.3, 10 and 14) in the German credit data. Model performance was evaluated using Accuracy, the area under the curve, F1-score, Precision, Brier score and Kolmogorov-Smirnov and statistical tests showed significant performance differences at 0.05 significance level. The study found that for German credit data with low imbalance ratio (IR = 2.3), the Logistic Regression model using Linear Discriminant Analysis (LDA) and Systematic Majority Over-Sampling (SM) had the best classification performance. For medium imbalance ratio (IR = 10) and high imbalance ratio (IR = 14), the most effective techniques for dimensionality reduction and resampling were Linear Discriminant Analysis (LDA), Random Under-Sampling (RUS), and Linear Discriminant Analysis (LDA), Borderline SMOTE (BSM) respectively. The Bagging Heterogeneous ensemble performed best both in cases with and without resampling and dimensionality reduction for medium and high imbalance ratios.

Field of Study: Statistics Student's Signature .....

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

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้ เสร็จสมบูรณ์ได้ด้วยความช่วยเหลืออย่างดีจากผู้ช่วยศาสตราจารย์ ดร.ณัตติฤดี เจริญรักษ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ที่คอยให้ความรู้ คำแนะนำ ตลอดชี้แนะแนวทางในการศึกษาแก่ผู้วิจัย อีกทั้งยังช่วยแก้ไขข้อบกพร่องต่างๆ ด้วยความเอาใจใส่ รวมทั้งให้องค์ความรู้แนวทางในการค้นคว้า จนกระทั่งงานวิจัยนี้สำเร็จลุล่วงไปได้ด้วยดี ผู้วิจัยขอกราบขอบพระคุณอย่างสูงมา ณ โอกาสนี้

ผู้วิจัยขอขอบพระคุณ ประธานกรรมการสอบวิทยานิพนธ์ ผู้ช่วยศาสตราจารย์ ดร. อัครินทร์ ไพบูลย์พานิช กรรมการสอบวิทยานิพนธ์ ผู้ช่วยศาสตราจารย์ ดร. นัท กุลวานิช และ รองศาสตราจารย์ ดร. สุพล ดุรงค์วัฒนา ที่กรุณาสละเวลามาเป็นกรรมการในการสอบครั้งนี้ ตลอดจนช่วยให้ความรู้ตรวจสอบและช่วยเหลือในการปรับปรุงแก้ไขวิทยานิพนธ์ให้สมบูรณ์ยิ่งขึ้น อีกทั้งขอขอบพระคุณคณาจารย์ประจำภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ที่ได้ให้ความรู้ตลอดระยะเวลาในการศึกษาหลักสูตรวิทยาศาสตรมหาบัณฑิต ทำให้สามารถนำเอาความรู้ที่ได้รับมาประยุกต์ใช้ในการทำวิทยานิพนธ์

สุดท้ายนี้ขอขอบพระคุณครอบครัว และเพื่อนของผู้วิจัย ที่คอยสนับสนุน เป็นกำลังใจที่ดีและคอยให้คำปรึกษาและช่วยเหลือผู้วิจัยเสมอมา

ศศิวิมล ศรีโรจน์

สารบัญ

หน้า

สารบัญรูป

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

ความเป็นมาและความสำคัญของปัญหา (Background and Problem Review)

การให้คะแนนเครดิตคือการประเมินความเสี่ยงที่เกี่ยวข้องกับการให้สินเชื่อแก่บริษัทหรือบุคคลถือเป็นหนึ่งในประเด็นที่สำคัญในอุตสาหกรรมการเงินและเป็นหนึ่งในความเสี่ยงที่สำคัญเกี่ยวข้องกับธนาคารที่ช่วยในการตัดสินใจ ซึ่งธนาคารเข้าถึงความน่าเชื่อถือของผู้สมัครในการให้สินเชื่อโดยใช้รูปแบบการให้คะแนนเครดิต ดังนั้นการพัฒนาตัวแบบการให้คะแนนเครดิตที่มีประสิทธิภาพจึงเป็นเครื่องมือที่สำคัญ

ในความเป็นจริงจำนวนผู้ที่ผิดนัดชำระหนี้ (bad credit) มีจำนวนน้อยกว่าผู้ที่ชำระหนี้ตรงเวลา (good credit) งานวิจัยส่วนมากมีวัตถุประสงค์ที่จะศึกษาเป็นข้อมูลของผู้ที่ผิดชำระหนี้ที่เป็นกลุ่มข้อมูลส่วนน้อย (minority) เมื่อเทียบกับปริมาณของข้อมูลผู้ที่ชำระหนี้ตรงเวลาหรือข้อมูลส่วนมาก (majority) ซึ่งเป็นลักษณะของข้อมูลที่ไม่สมดุล โดยอัตราส่วนความไม่สมดุล (Imbalance ratio) ของคะแนนเครดิตคำนวณจากจำนวนของผู้ชำระหนี้ตรงเวลาหรือกลุ่มข้อมูลส่วนมาก (majority) หารด้วยจำนวนผู้ที่ผิดนัดชำระหนี้หรือกลุ่มข้อมูลส่วนน้อย (minority) การจัดการกับข้อมูลไม่สมดุลเป็นสิ่งที่ท้าทายอย่างมากในการศึกษาด้านการเรียนรู้ของเครื่องจักร (machine learning)

วิธีในการรับมือกับปัญหาความไม่สมดุลของข้อมูลนี้ ถูกพัฒนามาอย่างต่อเนื่อง โดยส่วนใหญ่แบ่งออกเป็น 2 ระดับ ได้แก่ data level คือ การแก้ไขปัญหาก่อนที่จะนำไปสร้างตัวแบบ เพื่อให้จำนวนตัวอย่างของกลุ่ม majority และ minority มีจำนวนที่ใกล้เคียงกันหรือเท่ากัน วิธีการนี้ถูกเรียกว่า การสุ่มตัวอย่างซ้ำ (Resampling) เทคนิคการสุ่มตัวอย่างซ้ำที่รู้จักกันอย่างแพร่หลายเช่น Over-Sampling, Under-Sampling เป็นต้น (Marqués et al., 2013) ได้ใช้เทคนิคการสุ่มตัวอย่างซ้ำ (Resampling) เพื่อจัดการกับค่าอัตราส่วนความไม่สมดุลที่แตกต่างกัน โดยมีค่าอัตราส่วนความไม่สมดุลเท่ากับ 4, 6, 8, 10, 12 และ 14 ของข้อมูลคะแนนเครดิตกับตัวแบบการจำแนกประเภท การถดถอยโลจิสติก (Logistic Regression) และซัพพอร์ตเวคเตอร์แมชชีน (Support Vector Machine)

วิธีการแก้ไขปัญหาข้อมูลไม่สมดุลในระดับถัดมา คือระดับ algorithm level ซึ่งเป็นการใช้เครื่องมือทางสถิติหรืออัลกอริธึมทางการเรียนรู้ของเครื่องจักรเข้ามาช่วยในการจำแนกความแตกต่างระหว่าง majority และ minority เพื่อป้องกันไม่ให้เกิดความเอนเอียงของการจำแนกไปทางกลุ่มใด

กลุ่มหนึ่งมากเกินไป และในระดับ algorithm level ยังรวมถึงการลดมิติข้อมูล (Dimensionality Reduction) เพื่อช่วยปรับปรุงประสิทธิภาพในการเรียนรู้ของ classifier ให้ดีขึ้น นอกจากนี้ในระดับ algorithm level ยังมีการใช้เทคนิคการเรียนรู้แบบรวมกลุ่ม (Ensemble Learning)

(Zhang & Chi, 2021) ได้นำเสนอตัวแบบ Heterogeneous bagging ensemble ที่ประกอบไปด้วยตัวแบบพื้นฐาน 5 ตัวแบบคือ 1. Linear Support Vector Machine (LSVM), 2. Multivariate Discriminant analysis (MDA), 3. K-Nearest Neighborhood (KNN), 4. Decision Tree (DT) และ 5. Logistic Regression (LR) ที่สามารถปรับได้อัตโนมัติตามชุดข้อมูลด้วยตัวชี้วัด AUC โดยประยุกต์ใช้กับข้อมูลคะแนนเครดิต (Lenka et al., 2022) ได้นำเสนอการเปรียบเทียบตัวแบบการจำแนกประเภท สำหรับข้อมูลคะแนนเครดิต โดยการเพิ่มเทคนิคการสุ่มตัวอย่างซ้ำ (Resampling) และเทคนิคการคัดเลือกคุณลักษณะ (Feature selection) เพื่อจัดการกับความไม่สมดุลของข้อมูลคะแนนเครดิตและเพิ่มประสิทธิภาพในการจำแนกประเภทของตัวแบบให้มีความแม่นยำขึ้น

จากงานวิจัยที่ผ่านมาผู้วิจัยพบว่าวิธีการแก้ไขปัญหาข้อมูลไม่สมดุลในแต่ละค่าอัตราส่วนความไม่สมดุล (Imbalance Ratio) ของข้อมูลคะแนนเครดิต ประกอบด้วย 3 ส่วนคือ 1. การสุ่มตัวอย่างซ้ำ (Resampling) 2. การลดมิติข้อมูล (Dimensionality Reduction) 3. การเรียนรู้แบบรวมกลุ่ม (Ensemble) โดยในการเพิ่มประสิทธิภาพของตัวแบบจะใช้เทคนิคการเรียนรู้แบบรวมกลุ่มร่วมกับเทคนิคการลดมิติข้อมูลเพื่อลดปัญหาข้อมูลที่ใช้ในการวิเคราะห์มีมากเกินไป และใช้เทคนิคการสุ่มตัวอย่างซ้ำจัดการกับค่าอัตราส่วนความไม่สมดุลที่แตกต่างกัน เพื่อเพิ่มความแม่นยำและถูกต้องในการจำแนกประเภทของตัวแบบมากยิ่งขึ้น

ในงานวิจัยนี้ได้นำเสนอตัวแบบ Bagging Heterogeneous Ensemble โดยจะใช้ตัวแบบพื้นฐาน 4 ตัวแบบคือ ซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machine), ตัวแบบการถดถอยโลจิสติก (Logistic Regression), ต้นไม้การตัดสินใจ (Decision Tree) และ เพื่อนบ้านใกล้สุด (K-Nearest Neighbors) ร่วมกับเทคนิคการลดมิติข้อมูล (Dimensionality Reduction) และ การสุ่มตัวอย่างซ้ำ (Resampling) ทั้งนี้ในงานวิจัยจะใช้ข้อมูลเครดิตเยอรมันในแต่ละอัตราส่วนความไม่สมดุล (Imbalance Ratio) ที่แตกต่างกัน

วัตถุประสงค์ของการวิจัย (Objectives)

1. เพื่อสร้างตัวแบบ Bagging Heterogeneous Ensemble ที่เหมาะสมกับค่าอัตราความไม่สมดุล (imbalance ratio) ที่แตกต่างกันของข้อมูลคะแนนเครดิต (credit scoring)
2. เพื่อหาเทคนิคการลดมิติข้อมูล (dimensionality reduction) และเทคนิคการสุ่มตัวอย่างซ้ำ (resampling) ที่เหมาะสมกับอัตราส่วนความไม่สมดุล (imbalance ratio) ที่แตกต่างกันของข้อมูลคะแนนเครดิต

สมมติฐานการวิจัย (Research Hypotheses)

ตัวแบบ Bagging Heterogeneous Ensemble ที่ประกอบด้วยเทคนิคการลดมิติข้อมูล (dimensionality reduction) และการสุ่มตัวอย่างซ้ำ (resampling) จะมีความแม่นยำในการจำแนกประเภทดีกว่าตัวแบบการจำแนกประเภทพื้นฐาน 4 ตัวแบบ คือ Support Vector Machine (SVM), Logistic Regression (LR), Decision Tree (DT), K-Nearest Neighborhoods (KNN) และ Random Forest (RF)

ขอบเขตของการวิจัย (Scope of the Study)

1. การศึกษาในครั้งนี้ใช้ข้อมูลเครดิตเยอรมัน (German credit) จาก UCI Machine Learning Repository ประกอบไปด้วยตัวแปรต้น คือ สถานะของบัญชี (Status of existing checking account) , ระยะเวลา (Duration in month), ประวัติของเครดิต (Credit history), วัตถุประสงค์ (Purpose) , วงเงิน (Credit amount) , บัญชีออมทรัพย์/พันธบัตร (Savings account/bonds), ระยะเวลาของการทำงาน ปัจจุบัน (Present employment since), อัตราการผ่อนชำระเป็นเปอร์เซ็นต์ของรายได้ (Installment rate in percentage of disposable income), สถานะและเพศ (Personal status and sex), ลูกหนี้/ผู้ค้ำประกัน (Other debtors / guarantors) , ระยะเวลาการพักอาศัยของที่อยู่ปัจจุบัน (Present residence since), ทรัพย์สิน (Property) , อายุ (Age in years) , แผนการผ่อนชำระ (Other installment plans), ที่อยู่อาศัย (Housing) ,จำนวนเครดิตที่มีอยู่ในธนาคารนี้ (Number of existing credits at this bank), อาชีพ (Job), จำนวนผู้รับผิดชอบในการดูแล (Number of people being liable to provide maintenance for), โทรศัพท์ (Telephone),แรงงานต่างด้าว (foreign worker) และตัวแปรตามคือ ผลการให้คะแนน

สินเชื่อ โดย good แปลว่า มีความน่าจะเป็นในการชำระคืนหนี้ bad แปลว่า มีความน่าจะเป็นในการไม่ชำระหนี้

2. กำหนดอัตราความไม่สมดุล (imbalance ratio) ในการศึกษาครั้งนี้คือ 2.3, 10 และ 14 โดย IR = 2.3 คือ ผู้ที่ชำระหนี้ตรงเวลา 700 คนหารด้วยผู้ที่ผิดนัดชำระหนี้ 300 คน
3. กำหนดอัตราส่วนข้อมูลชุดเรียนรู้ (training set) และข้อมูลชุดทดสอบ (test set) ในการศึกษาครั้งนี้คือ 80% และ 20%
4. กำหนดค่าจำนวนตัวแบบในการเรียนรู้แบบรวมกลุ่ม (ensemble estimators) คือ 20

ประโยชน์ที่คาดว่าจะได้รับ

ผลการศึกษาแสดงให้เห็นถึงประสิทธิภาพในการจำแนกประเภทของตัวแบบ bagging heterogeneous ensemble ที่ประกอบด้วยเทคนิคการลดมิติข้อมูล (dimension reduction) และการสุ่มข้อมูลซ้ำ (resampling) ในแต่ละค่าอัตราความไม่สมดุล (imbalance ratio) ที่แตกต่างกัน

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

บทความนี้เกี่ยวข้องกับตัวแบบการเรียนรู้กลุ่ม (Ensemble Learning) สำหรับข้อมูลที่ไม่สมดุล (Imbalanced learning) จากงานวิจัยการเรียนรู้ข้อมูลไม่สมดุลได้รวบรวมมาไว้ในบทนี้โดยแบ่งเป็น 2 ส่วนคือ

1. อธิบายในส่วนของข้อมูลไม่สมดุลสำหรับข้อมูลคะแนนเครดิต (Credit Scoring)
2. อธิบายในส่วนของการเรียนรู้ของเครื่องจักร (Machine Learning)

คะแนนเครดิต (Credit scoring)

คะแนนเครดิตเป็นแบบจำลองที่ใช้กระบวนการทางสถิติในการจัดการข้อมูลเพื่อกำหนดเป็นค่าคะแนนเครดิตซึ่งใช้เป็นตัววัดความน่าจะเป็นในการชำระหนี้คืน คะแนนเครดิตถูกนำมาใช้อย่างแพร่หลายในการพิจารณาคำขอสินเชื่อ เนื่องจากคุณภาพของผู้กู้จะส่งผลโดยตรงต่อความสามารถในการทำกำไรและความมั่นคงของสถาบันการเงิน การคัดกรองและพิจารณาคำขอของผู้กู้จึงเป็นขั้นตอนสำคัญในการป้องกันความเสี่ยงด้านเครดิต โดยทางสถาบันการเงินจะใช้เป็นขั้นตอนในการพิจารณาเพื่อลดความเสี่ยงและเพิ่มประสิทธิภาพในขั้นตอนการประเมินความเสี่ยงด้านเครดิต นอกจากนี้ยังสามารถใช้เป็น Learning Indicator เพื่อติดตามคุณภาพและความสามารถ ในการชำระหนี้ของ ลูกหนี้ที่จะเกิดการผิดนัดชำระหนี้ได้

ตัวแปรต้นที่ใช้ในการพยากรณ์โอกาสในการชำระหนี้ตรงตามเวลานั้นมีได้หลายตัวแปรเช่น 1) อายุ โดยช่วงวัยกลางคนมีอัตราการผิดนัดชำระหนี้ที่สูงกว่าช่วงวัยรุ่นและวัยสูงอายุ (Debbaut et al., 2014) 2) เพศ โดย 60.3% ของผู้หญิงมีแนวโน้มที่จะเสียค่าธรรมเนียมจากการชำระหนี้ช้าเมื่อเปรียบเทียบกับ 57.4% ของผู้ชาย (Holmes, 2021) 3) เงินฝากธนาคาร พบว่าผู้ที่มีเงินฝากสูงจะมีโอกาสในการชำระหนี้เต็มมากกว่าผู้ที่มีเงินฝากต่ำ (White, 2007) 4) ประเภทที่อยู่อาศัยเป็นตัวช่วยบ่งบอกถึงทรัพย์สินที่ถือครองซึ่งคาดว่าจะมีผลกับคะแนนเครดิต (Lopes, 2008) และ 5) ระยะเวลาในการทำงาน เป็นตัวช่วย บ่งบอกถึงการไม่มีงานทำหรือการมีงานทำซึ่งส่งผลต่อคะแนนเครดิต (Irby, 2021)

โดยในปัจจุบันตัวแบบสำหรับคะแนนเครดิตสามารถสร้างได้ด้วยหลากหลายวิธีเช่น วิธีทางสถิติ (Statistical methods), วิธีตามเงื่อนไข (Rule-based methods) และ วิธีการเรียนรู้ด้วยเครื่องจักร (Machine learning methods) ตัวอย่างตัวแบบการเรียนรู้ด้วยเทคนิคทางสถิติที่มีประสิทธิภาพ คือ ตัวแบบการถดถอยโลจิสติก (Logistic Regression) และการวิเคราะห์การจำแนก

ประเภทเชิงเส้น (Linear Discriminant Analysis) โดยใช้ความสัมพันธ์เชิงเส้นระหว่างตัวแปรต้นและตัวแปรตาม แต่ตัวแบบเหล่านี้ ไม่สามารถวิเคราะห์ความสัมพันธ์ไม่เชิงเส้นระหว่างตัวแปรต้นและตัวแปรตามได้ ตัวอย่างตัวแบบการเรียนรู้ด้วยวิธีตามเงื่อนไขโดยใช้กฎในการแยก รวมเข้ากับการเรียนรู้ด้วยเครื่องจักรสำหรับการทำนายความสามารถในการชำระหนี้ ข้อดีของวิธีตามเงื่อนไขคือสามารถสร้างและทำความเข้าใจได้ง่ายและสามารถจัดการกับปัญหาที่ซับซ้อนได้แต่การใช้วิธีตามเงื่อนไขไม่เหมาะกับข้อมูลที่มีตัวแปรต้นเยอะ และตัวแบบการเรียนรู้ด้วยเครื่องจักรเป็นที่นิยมในปัจจุบัน เทคนิคการเรียนรู้ด้วยเครื่องจักรสามารถสกัดข้อมูลที่สำคัญและสร้างคะแนนเครดิตที่มีประสิทธิภาพได้โดยอัตโนมัติ (Yu et al., 2018) พบว่าเทคนิคการเรียนรู้ด้วยเครื่องจักรมีประสิทธิภาพมากกว่าเทคนิควิธีทางสถิติ อย่างไรก็ตามการเรียนรู้ด้วยเครื่องจักรยังมีข้อจำกัดหลายอย่าง เช่น ต้องมีการปรับ hyperparameter ที่เหมาะสม, ปัญหา Local minima, การเกิดปัญหา overfitting และการคำนวณของการเรียนรู้ด้วยเครื่องจักรอาจมีระยะเวลาที่นาน (De Melo Junior et al., 2019) ได้แสดงการเปรียบเทียบตัวแบบ การจำแนกประเภทข้อมูลคะแนนเครดิต 19 ตัวแบบที่แตกต่างกัน ประกอบไปด้วย 11 ตัวแบบพื้นฐาน (Base classifier), 3 ตัวแบบการเรียนรู้กลุ่มแบบมีค่าใช้จ่าย (Cost-sensitive ensemble) และ 5 ตัวแบบการเรียนรู้กลุ่มแบบไม่สมดุล (Imbalanced ensemble) โดยใช้ชุดข้อมูลคะแนนเครดิต 3 ชุดคือ (1) Australian, (2) German และ (3) Japanese ในแต่ละชุดข้อมูลจะทำการสร้าง 12 อัตราความไม่สมดุล (Imbalanced ratio) ที่แตกต่างกันวัดผลด้วยค่า AUC และ Friedman's test พบว่าตัวแบบต้นไม้ตัดสินใจ (Random forest) และ XGBoost มีประสิทธิภาพเฉลี่ยทุกค่าอัตราส่วนความไม่สมดุลมากที่สุด ข้อมูลคะแนนเครดิตที่ไม่สมดุลสามารถวัดระดับความไม่สมดุลได้จากการคำนวณค่าอัตราส่วนความไม่สมดุล (IR) ซึ่งผลกระทบของระดับความไม่สมดุลส่งผลกับประสิทธิภาพของตัวแบบการเรียนรู้ของเครื่องจักร AI (Marqués et al., 2013) ออกแบบการทดลองโดยมีการสร้างข้อมูลคะแนนเครดิตเป็น 6 อัตราส่วนความไม่สมดุล (Imbalance Ratio) ที่แตกต่างกันคือ 4, 6, 8, 10, 12 และ 14 ของข้อมูลคะแนนเครดิต 5 ชุดข้อมูลโดยคำนวณจากจำนวนของกลุ่มที่มากหารจำนวนของกลุ่มที่น้อย เพื่อหาผลกระทบของการสุ่มตัวอย่างซ้ำ (Resampling) และการจำแนกประเภท (Brown & Mues, 2012) ออกแบบการทดลองโดยการสร้างอัตราความไม่สมดุลของข้อมูลคะแนนเครดิตออกเป็น 8 ค่าที่แตกต่างกัน โดยการคำนวณเปอร์เซ็นในการแบ่งข้อมูลจำนวนส่วนมากและจำนวนส่วนน้อยเพื่อเปรียบเทียบประสิทธิภาพของตัวแบบการจำแนกประเภทต่าง ๆ

การเรียนรู้ของเครื่องจักร (Machine learning)

1.1. การเรียนรู้ด้วยเครื่องจักร มีการใช้เทคนิคต่าง ๆ ในการสร้างตัวแบบการให้คะแนนเครดิต

1.1.1.ซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machine : SVM) เป็นตัวแบบจำแนกประเภทข้อมูลที่ใช้ Hyperplane ในการแบ่งข้อมูลเรียนรู้ที่มีอยู่ 2 ประเภท (เช่น ชำระหนี้ตรงเวลาหรือผิดนัดชำระหนี้) โดยข้อมูลที่อยู่เหนือ Hyperplane จะเป็นประเภทหนึ่งและข้อมูลที่อยู่ใต้ Hyperplane จะเป็นอีกประเภทหนึ่ง โดยปกติข้อมูลอาจมีการกระจายตัวที่การแบ่งข้อมูลทั้งสองประเภทออกจากกันด้วย Hyperplane ไม่สามารถทำได้ วิธีการแก้ไขคือการแปลงข้อมูลเรียกว่า Feature Space โดยใช้ Kernel Function ซึ่งจะสามารถใช้ SVM ในการแบ่งข้อมูลทั้งสองประเภทออกจากกันได้ ตัวอย่าง Kernel Function ที่นิยมคือ (1) Linear Kernel (2) Polynomial Kernel (3) Radial Basis Function และ (4) Sigmoid kernel

1.1.2.ตัวแบบการถดถอยโลจิสติก (Logistic Regression : LR) เป็นการศึกษาความสัมพันธ์ระหว่างตัวแปรต้นกับตัวแปรตาม โดยที่ตัวแปรตามเป็นข้อมูลเชิงคุณภาพ ส่วนตัวแปรต้นสามารถเป็นไปได้ทั้งข้อมูลเชิงคุณภาพและข้อมูลเชิงปริมาณและตัวแบบการถดถอยโลจิสติกได้ถูกนำมาใช้อย่างแพร่หลายในการพยากรณ์โอกาสในการชำระหนี้ โดยตัวแบบที่ใช้ส่วนมากคือ ตัวแบบการถดถอยโลจิสติกทวิ

ตัวแบบการถดถอยโลจิสติกทวิ (Binary Logistic Regression model เป็นการวิเคราะห์ความถดถอยที่ตัวแปรตามเป็นข้อมูลเชิงคุณภาพที่มีค่าได้เพียง 2 ค่า คือ 0 และ 1 ส่วนตัวแปรต้นสามารถเป็นได้ทั้งข้อมูลเชิงคุณภาพและข้อมูลเชิงปริมาณ โดยการศึกษาครั้งนี้กำหนดให้ตัวแปรตาม (y) มีค่า 2 ค่า คือ 0 แทน กลุ่มคนที่ชำระหนี้ตรงเวลา และ 1 แทนกลุ่มคนที่ผิดนัดชำระหนี้ในตัวแบบคะแนนเครดิต

การเขียนตัวแบบโลจิสติกจะอยู่ในรูปของ log ของ odds เรียกว่า logit หรือ logistic response function โดยสมการจิสติกจะอยู่ในรูป log(odds) ซึ่งเรียกว่า logit มีรูปแบบดังนี้

$$logit(P) = \beta_0 + \beta_1 x + \cdots + \beta_k X_k$$

โดยที่ $\beta_0, \beta_1, \ldots, \beta_k$ คือค่าพารามิเตอร์ของตัวแบบ

$P = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_k X_k}}{1+e^{\beta_0+\beta_1 X_1+\cdots+\beta_k X_k}}$ คือความน่าจะเป็นของการเกิดเหตุการณ์ (y=1)

$X_1, X_2, \ldots, X_k$ คือตัวแปรต้นตัวที่ 1,2,...,k

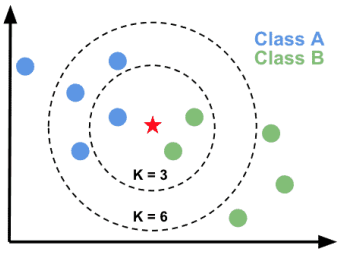
1.1.3.เพื่อนบ้านใกล้สุด (K-Nearest Neighbors : KNN) เป็นวิธีการจำแนกประเภทของข้อมูลโดยการเปรียบเทียบข้อมูลกับข้อมูลตัวอย่างที่เก็บอยู่ในฐานข้อมูล (Instance-based Learning) คือ ข้อมูลเรียนรู้จะถูกเก็บไว้เป็นตัวอย่างในฐานข้อมูลเมื่อต้องการจำแนกประเภทข้อมูลใหม่จะค้นหาตัวอย่างข้อมูลในฐานที่มีลักษณะใกล้เคียงข้อมูลใหม่มากที่สุด (เพื่อนบ้าน) จำนวน K ตัว มาใช้ร่วมตัดสินว่าข้อมูลใหม่นี้ควรมีประเภทอะไร โดยวิธีเพื่อนบ้านใกล้สุดจะใช้หลักการที่ว่าข้อมูลใหม่ที่ต้องการทราบประเภทจะมีลักษณะใกล้เคียงกับข้อมูลใดในฐานข้อมูล ข้อมูลใหม่นั้นก็ควรมีประเภทเดียวกัน สำหรับการหาความเหมือนของข้อมูลในฐานข้อมูลสามารถใช้ตัววัดความเหมือน (Similarity) ได้หลายวิธี เช่น Euclidean Distance หรือ Cosine Similarity และการตัดสินใจร่วมกัน K ตัวนั้น อาจใช้วิธี Majority Vote ซึ่งเป็นวิธีการตัดสินใจจากประเภทข้อมูลที่มีจำนวนมากที่สุดในข้อมูลเพื่อนบ้าน K ตัวนั้น แต่การใช้วิธี Majority Vote ในการตัดสินประเภทของข้อมูลอาจเกิดกรณีที่ตัดสินใจเสมอกัน โดยเฉพาะข้อมูลเพื่อนบ้านที่มีมากกว่า 2 ประเภท ทำให้ไม่สามารถตัดสินประเภทของข้อมูลได้ นอกจากนี้การตัดสินประเภทข้อมูลด้วยวิธี Majority Vote ไม่ได้คำนึงว่าข้อมูลเพื่อนบ้านใดมีความเหมือนกับข้อมูลใหม่ที่จะจำแนกประเภทมากกว่ากัน ผลการจำแนกประเภทอาจขึ้นกับประเภทของเพื่อนบ้านที่อยู่ไกลเป็นส่วนใหญ่ก็ได้ การแก้ไขทำได้โดย ใช้การ Vote แบบถ่วงน้ำหนัก คือการให้น้ำหนักของคะแนน Vote ของเพื่อนบ้านแต่ละข้อมูลเป็นสัดส่วนผกผันกับระยะทางระหว่างข้อมูลเพื่อนบ้านนั้นกับข้อมูลใหม่ที่ต้องการจำแนกประเภทยกกำลังสอง ดังสมการ

$$w_i = \frac{1}{d(X_i, X)^2}$$

โดยที่ w_i คือน้ำหนักของคะแนน Vote ของเพื่อนบ้าน X_i
และ $d(X_i, X)$ คือ ระยะห่างระหว่าง X_i และข้อมูล X ที่ต้องการจำแนกประเภท

รวมทั้งการเลือกค่า K ที่เหมาะสมมีผลอย่างมากต่อความถูกต้องของวิธีการจำแนกประเภท เช่น ในกรณีที่เลือกค่า K น้อยเกินไป เช่น K=1 หรือ 2 การตัดสินประเภทของข้อมูลจะขึ้นกับข้อมูลตัวอย่างในฐานข้อมูลเพียงไม่กี่จำนวน ถ้าข้อมูลเหล่านั้นเป็น Noise ก็จะทำให้การตัดสินประเภทของข้อมูลผิดพลาดได้ ในกรณีที่ K มีค่ามากเกินไป ตัวอย่างของข้อมูลที่ใช้ในการตัดสินอาจไม่มีลักษณะใกล้เคียงเลยกับข้อมูลที่ต้องการ

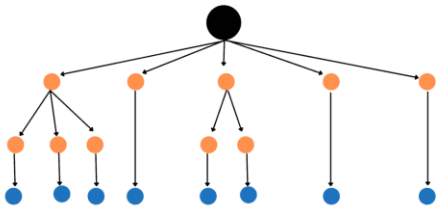
ทราบประเภท จึงทำให้การตัดสินประเภทของข้อมูลผิดพลาดได้เช่นกัน สำหรับการคำนวณความน่าจะเป็นคำนวณได้จากจำนวนประเภทส่วนมากหารด้วยข้อมูลทั้งหมด



รูปที่ 2.1 กลไกของอัลกอรึทึมเพื่อนบ้านใกล้สุด

หมายเหตุ. จาก https://www.jcchouinard.com/k-nearest-neighbors/

1.1.4.ต้นไม้การตัดสินใจ (Decision Tree : DT) เป็นวิธีการจำแนกประเภทของข้อมูลที่ใช้ตัวแบบในรูปต้นไม้การตัดสินใจ แต่ละโหนดของต้นไม้การตัดสินใจสำหรับการจำแนกประเภทของข้อมูลจะประกอบด้วยเงื่อนไขที่ใช้ตัวแปรใดตัวแปรหนึ่งของข้อมูลในการตัดสินใจเลือกโหนดลูกใดลูกหนึ่งเป็นโหนดตัดสินใจ โดยการตัดสินใจจะเริ่มจากโหนดรากของต้นไม้และไล่ไปยังโหนดลูกจนถึงโหนดใบซึ่งจะเป็นโหนดที่จะบอกประเภทของข้อมูลนั้นได้ การตัดสินใจที่โหนดเปรียบได้กับการแบ่ง Data Space ออกเป็นส่วน ๆ ที่ส่วนเล็กลงเรื่อย ๆ จนข้อมูลในส่วนเหล่านั้นมีประเภทเป็นชนิดเดียวกันทั้งหมดหรือเกือบทั้งหมด กล่าวคือข้อมูลไม่มีความหลากหลาย



รูปที่ 2.2 กลไกของอัลกอรึทึมต้นไม้ตัดสินใจ

หมายเหตุ. จาก https://hands-on.cloud/decision-tree-using-python/

การสร้างเงื่อนไขตัดสินใจสำหรับตัวแปรแต่ละประเภทเป็นดังนี้

- ตัวแปรชนิด Nominal แบ่งเป็น 2 กรณีคือ (1) ไม่จำกัดโหนดลูก เป็นเงื่อนไขที่ตัวแปรมีค่าเท่ากับค่าที่เป็นไปได้ตัวใดตัวหนึ่งของตัวแปรนั้น (2) จำกัดโหนดลูก (เช่น

ให้มีค่าเท่ากับสอง สำหรับ Binary Decision Tree) ค่าของตัวแปรที่เป็นไปได้จะต้องถูกจัดเป็นกลุ่มตามจำนวนโหนดลูกที่ต้องการ

- ตัวแปรชนิด Ordinal สามารถทำได้แบบเดียวกับตัวแปรชนิด Nominal ยกเว้นการจับกลุ่มของค่าของตัวแปรเพื่อให้ได้จำนวนกลุ่มเท่ากับจำนวนโหนดลูกตามที่ต้องการจะต้องจับกลุ่มเป็นช่วงของค่าของตัวแปรเรียงตาม Order ของค่าจากน้อยไปมาก เช่น ถ้าตัวแปร Ordinal มีค่าเป็น น้อย ปานกลาง และมาก การจับกลุ่มค่าของตัวแปรนี้เป็น 2 กลุ่ม สามารถทำได้ดังนี้ {น้อย} กับ {ปานกลาง, มาก} หรือ {น้อย, ปานกลาง} กับ {มาก}
- ตัวแปรชนิด Numerical การสร้างเงื่อนไขตัดสินใจสำหรับตัวแปรชนิดนี้จำเป็นต้องแบ่งค่าตัวแปร ซึ่งเป็นค่าต่อเนื่องให้เป็นช่วง (Intervals) โดยมีจำนวนช่วงเท่ากับจำนวนของโหนดลูกที่ต้องการ การกำหนดช่วงอาจทำได้โดยการ แบ่งช่วงที่มีความกว้างเท่ากัน หรือ การแบ่งช่วงที่มีจำนวนข้อมูลในแต่ละช่วงเท่ากัน (Equal Frequencies)

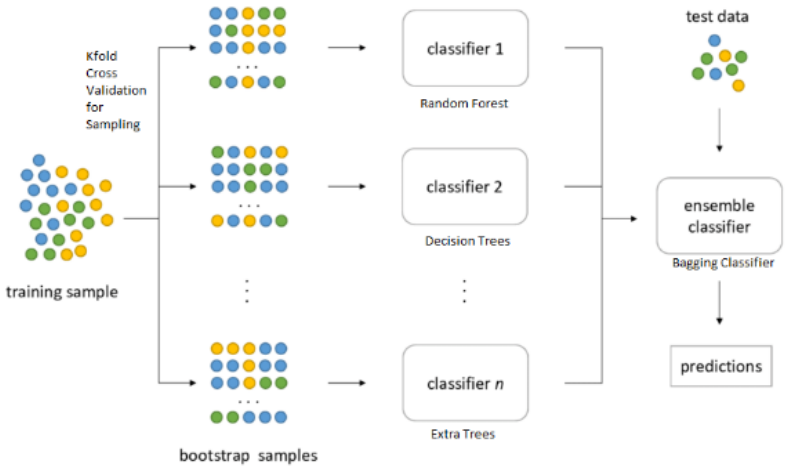
1.1.5.การสุ่มป่าไม้ (Random Forest : RF) เป็นการเรียนรู้ด้วยเครื่องจักรที่นิยมใช้กับการแก้ปัญหา Regression และ Classification โดยการสุ่มป่าไม้คือการต่อยอดจากตัวแบบต้นไม้ตัดสินใจ (Decision Tree) แตกต่างกันที่การสุ่มป่าไม้เป็นการเพิ่มจำนวนต้นไม้ตัดสินใจหลายๆต้น ทำให้การจำแนกประเภทมีประสิทธิภาพสูงขึ้น โดยมีหลักการทำงานคือ จะแบ่งข้อมูลออกไปทำต้นไม้ตัดสินใจโดยแต่ละต้นจะได้รับคุณลักษณะและข้อมูลสำหรับการเรียนรู้ที่แตกต่างกัน เพื่อทำให้แต่ละต้นไม้ตัดสินใจมีความหลากหลายและมีอิสระซึ่งกันและกัน

การทำงานของการสุ่มป่าไม้จะเริ่มต้นจาก

1. ทำการสุ่มเลือกคุณลักษณะและข้อมูลจากชุดข้อมูลทั้งหมดที่มี
2. สร้างต้นไม้ตัดสินใจจากชุดข้อมูลตัวอย่างแต่ละชุดและหาค่าพยากรณ์จากต้นไม้แต่ละต้น
3. เลือกจำนวนต้นไม้ตัดสินใจ จากนั้นทำซ้ำในขั้นตอนที่ 1 และ 2 ในการสร้างต้นไม้
4. การรวมค่าพยากรณ์จากต้นไม้ตัดสินใจแต่ละต้นนั้น สามารถใช้การรวมได้หลายวิธี เช่น วิธี Majority Vote, The Highest Mean Predicted Probability เป็นต้น

(Feng et al., 2018) พบว่าการเรียนรู้แบบรวมกลุ่มมีประสิทธิภาพดีกว่าตัวแบบการเรียนรู้แบบเดี่ยว โดยการเรียนรู้แบบรวมกลุ่ม (Ensemble Learning Model) เป็นการสร้างตัวแบบการเรียนรู้แบบกลุ่มที่มีการใช้ตัวจำแนก (Classifier) มากกว่าหนึ่งตัวในการเรียนรู้แต่ละตัวจำแนก จะมีกระบวนการทำงานของตัวเองและทุกตัวจำแนกจะกระทำกับข้อมูลชุดเดียวกัน เมื่อได้ผลการจำแนกของแต่ละตัวจำแนกแล้ว ก็จะนำผลลัพธ์เหล่านั้นมาผ่านวิธีการรวบรวม (Combination หรือ Vote) และสุดท้ายนำไปตัดสินใจโดยการสร้างตัวแบบหลายตัวเพื่อร่วมกันตัดสินการจำแนกประเภทของข้อมูลสามารถทำได้ 3 ลักษณะคือ 1. Bagging 2. Boosting และ 3. Stacking ซึ่งในการทดลอง นี้จะใช้วิธี Bagging

1.1.6. Bagging เป็นวิธีการที่ตัวแบบแต่ละตัวใน Ensemble Method จะถูกสร้างจากชุดข้อมูลที่สุ่มมาจากชุดข้อมูลเรียนรู้ที่กำหนดให้แบบใส่คืน (Sampling with replacement) กล่าวคือ เมื่อสุ่มข้อมูลขึ้นมาได้หนึ่งตัวจะคืนข้อมูลนั้นกลับเข้าไปยังชุดข้อมูลอย่างเดิม ทำให้ข้อมูลนั้นมีโอกาสถูกสุ่มซ้ำอีกในอนาคต โดยผลลัพธ์จากการสุ่ม n ครั้ง จะได้ข้อมูลที่เป็น subset ของชุดข้อมูลเรียนรู้เดิม ทำให้ตัวแบบถูกสร้างขึ้นจากชุดข้อมูลที่แตกต่างกัน และการที่ตัวแบบหลายตัวถูกสร้างขึ้นจากชุดข้อมูลที่อาจจะไม่เหมือนกันจะใช้การรวมผลลัพธ์ใช้ Majority Vote โดยตัวแบบแต่ละตัวจะมีน้ำหนักเท่ากัน หรือใช้วิธี The Highest Mean ซึ่ง Bagging เป็น Ensemble Method ที่ช่วยลดความผิดพลาดของการจำแนกประเภทข้อมูลโดยมีเป้าหมายที่จะลดความผิดพลาดของการจำแนกประเภทที่ตัวแบบถูกสร้างให้มีความจำเพาะและซับซ้อนมากเกินไป (Complexity) ที่ทำให้เกิดปัญหา Overfitting



รูปที่ 2.3 กลไกของอัลกอรีทึม Boosting Aggregation (Bagging)

หมายเหตุ. จาก *https://medium.com/ml-research-lab/bagging-ensemble-meta-algorithm-for-reducing-variance-c98fffa5489f*

1.2. ข้อมูลเครดิตอาจมีจำนวนตัวแปรต้นที่เยอะและไม่สำคัญทำให้ส่งผลต่อเวลาในการคำนวณการสร้างตัวแบบ แก้ไขโดยการใช้เทคนิคการลดมิติข้อมูล (Dimensionality Reduction) เพื่อช่วยลดระยะเวลาในการสร้างตัวแบบได้

1.2.1.การสลายตัวของค่าเอกพจน์ (Singular Value Decomposition : SVD) เป็นวิธีการแยกส่วนของเมทริกซ์ให้เป็นผลคูณของเมทริกซ์ 3 ตัวที่มีคุณสมบัติพิเศษที่สามารถนำไปวิเคราะห์ข้อมูลได้ดียิ่งขึ้น เมื่อ SVD ถูกนำมาใช้กับเมทริกซ์ที่มีขนาดใหญ่ จะช่วยให้สามารถประมวลผลได้รวดเร็วขึ้น โดยการแปลงเมทริกซ์ข้อมูลอยู่ในรูป

$$X_{n\times d} = U_{n\times n} \sum_{n\times d} (V_{d\times d})^T$$

โดยที่ X คือ เมทริกซ์ข้อมูล

U คือ Left singular vectors

$\sum$ คือ Diagonal of singular values

V^T คือ Right singular vectors

m คือ จำนวนแถว

n คือ จำนวนคอลัมน์

ขั้นตอนการทำงานของการสลายตัวของค่าเอกพจน์สำหรับการลดมิติข้อมูลมีดังนี้ (Anowar et al., 2021)

1. แยกเมทริกซ์ข้อมูล X เป็น 3 เมทริกซ์คือ U, $\sum$ และ $\mathbf{V^T}$
2. การลดมิติข้อมูลทำได้โดยการเลือก K อันดับแรกของ $\sum$ (Diagonal of Singular values)

ตัวอย่างการคำนวณจากข้อมูลคะแนนเครดิตเยอรมัน

ข้อมูลตั้งต้น: $X \in R^{n\ X\ d}$

ผลลัพธ์: $Y \in R^{n\ X\ k}$

$X = \begin{bmatrix} -0.738 & -0.910 & \cdots & 0 \\ 0.256 & 0.444 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ -0.240 & -0.009 & \cdots & 0 \end{bmatrix}$ โดยมีมิติคือ n X d และมีจำนวนกลุ่มคือ 2 กลุ่ม

<u>ขั้นตอนที่ 1</u>

$$\begin{bmatrix} 4.162 & 3.264 & \cdots & 7.440 \\ 3.690 & -2.381 & \cdots & -1.060 \\ \vdots & \vdots & \vdots & \vdots \\ 3.179 & -9.702 & \cdots & 9.312 \end{bmatrix}, \begin{bmatrix} 53.554 & 0 & \cdots & 0 \\ 0 & 38.296 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \begin{bmatrix} 0.006 & 0.005 & \cdots & 0.017 \\ -0.598 & -0.676 & \cdots & 0.011 \\ \vdots & \vdots & \vdots & \vdots \\ 0.007 & -0.010 & \cdots & 0.008 \end{bmatrix}$$

<u>ขั้นตอนที่ 2</u> เลือก K = 1 และคำนวณข้อมูลใหม่ด้วย Y = U * $\sum_k$

$$\begin{bmatrix} 2.229 & 0 & \cdots & 0 \\ 1.976 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1.702 & 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} 4.162 & 3.264 & \cdots & 7.440 \\ 3.690 & -2.381 & \cdots & -1.060 \\ \vdots & \vdots & \vdots & \vdots \\ 3.179 & -9.702 & \cdots & 9.312 \end{bmatrix} * \begin{bmatrix} 53.554 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

ข้อมูลที่ได้หลังจากการลดมิติข้อมูลคือ $\begin{bmatrix} 2.229 \\ 1.976 \\ \vdots \\ 1.702 \end{bmatrix}$

1.2.2.การวิเคราะห์การจำแนกประเภทเชิงเส้น (Linear Discriminant Analysis : LDA) หรือ Fisher Discriminant เป็นวิธีที่ใช้สำหรับการลดมิติของข้อมูล (dimensionality reduction) และการจัดกลุ่มข้อมูล (classification) โดยเป้าหมายของ LDA คือการ

เพิ่มปริมาณการกระจายระหว่างกลุ่มให้มากที่สุดและลดปริมาณการกระจายภายในกลุ่มให้น้อยที่สุด สำหรับการกระจายตัวภายในกลุ่ม (Within-class scatter matrix) ดังสมการ

$$S_W = \sum_{j=1}^{C} \sum_{i=1}^{N_j} (\Gamma_i^j - \mu_j)(\Gamma_i^j - \mu_j)^T$$

และการกระจายตัวระหว่างกลุ่ม (Between-class scatter matrix) ดังสมการ

$$S_B = \sum_{j=1}^{C} (\mu_j - \mu)(\mu_j - \mu)^T$$

โดยที่ S_W คือ การกระจายตัวภายในกลุ่ม

S_B คือ การกระจายตัวระหว่างกลุ่ม

N_j คือ จำนวน Sample ในกลุ่มที่ j

Γ_i^j คือ Sample ที่ i ของกลุ่มที่ j

c คือ จำนวนกลุ่ม

μ_j คือ ค่าเฉลี่ยของกลุ่มที่ j

μ คือ ค่าเฉลี่ยรวมของกลุ่มทั้งหมด

ขั้นตอนการทำงานของการวิเคราะห์การจำแนกประเภทเชิงเส้นสำหรับการลดมิติข้อมูลมีดังนี้ (Anowar et al., 2021)

1. คำนวณ Mean vector และสร้าง 2 Scatter matrices คือ Within-class scatter matrix (d X d) และ Between-class scatter matrix (d X d) โดยที่ d คือ จำนวนมิติข้อมูล
2. คำนวณค่า Eigenvalues (d) และหา Eigenvectors (d X d) ของ Scatter matrices
3. เรียง Eigenvectors (d X d) ตามลำดับของ Eigenvalue (d) จากมากไปน้อย
4. สร้างเมทริกซ์ W (d X k) ด้วย Eigenvectors จากข้อที่ 3 (k ลำดับแรก) โดยที่ $\mathbf{k} = \min(\mathbf{c}-1, \mathbf{d})$
5. แปลง X โดยใช้เมทริกซ์ W จากข้อที่ 4 โดยจะได้ Subspace Y = X * W

ตัวอย่างการคำนวณจากข้อมูลคะแนนเครดิตเยอรมัน

ข้อมูลตั้งต้น: $X \in R^{n\ X\ d}$

ผลลัพธ์: $Y \in R^{n\ X\ k}$

$X = \begin{bmatrix} -0.738 & -0.910 & \cdots & 0 \\ 0.256 & 0.444 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ -0.240 & -0.009 & \cdots & 0 \end{bmatrix}$ โดยมีมิติคือ n X d และมีจำนวนกลุ่มคือ 2 กลุ่ม

Within-class scatter matrix $\begin{bmatrix} 755.505 & 494.581 & \cdots & -16.627 \\ 494.581 & 806.988 & \cdots & -2.705 \\ \vdots & \vdots & \vdots & \vdots \\ -16.627 & -2.705 & \cdots & 27.753 \end{bmatrix}$ โดยมีมิติ d X d

Between-class scatter matrix $\begin{bmatrix} 36.837 & 29.455 & \cdots & -2.682 \\ 29.455 & 23.551 & \cdots & -2.145 \\ \vdots & \vdots & \vdots & \vdots \\ -2.682 & -2.145 & \cdots & 0.195 \end{bmatrix}$ โดยมีมิติ d X d

Eigenvalues ที่ได้จากการคำนวณ $S_W^{-1}S_B$

$$\begin{matrix} 0 \\ 4.11 \times 10^{-1} \\ \vdots \\ 1.602 \times 10^{-19} \\ 7.865 \times 10^{-20} \end{matrix}$$

Eigenvector ที่ได้จาก Eigenvalues ($A\vec{X} = \lambda\vec{X}$) โดยที่ A คือ $S_W^{-1}S_B$ มีมิติ d X d, $\vec{X}$ คือ Eigenvector และ λ คือ Eigenvalues

$$\begin{bmatrix} -0.570 & -0.059 & \cdots & -0.570 \\ 0.059 & -0.072 & \cdots & 0.059 \\ \vdots & \vdots & \vdots & \vdots \\ -0.167 & 0.203 & \cdots & -0.167 \end{bmatrix}$$

เรียงลำดับ Eigenvectors จากมากไปน้อยตามค่าของ Eigenvalues

$$\begin{bmatrix} -0.059 \\ \vdots \\ 0.203 \end{bmatrix}, \begin{bmatrix} 0.081 \\ \vdots \\ 0.175 \end{bmatrix}, \ldots, \begin{bmatrix} -5.707 \times 10^{-1} \\ \vdots \\ -1.673 \times 10^{-1} \end{bmatrix}$$

จากนั้นเลือก Eigenvalue k โดยที่ $k = min(c-1, d)$ ที่มี Eigenvalues สูงที่สุดเพื่อสร้าง Matrix W มิติ d X k

กำหนดให้ k = 1

$$W = \begin{bmatrix} -0.059 \\ \vdots \\ 0.203 \end{bmatrix}$$

Y = X * W

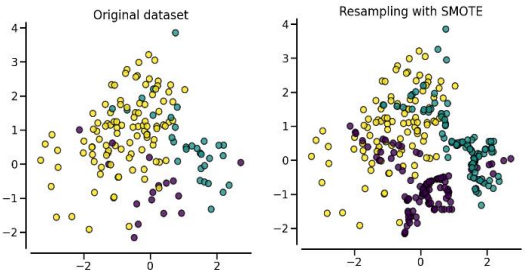
$$\begin{bmatrix} 0.878 \\ \vdots \\ 0.513 \end{bmatrix} = \begin{bmatrix} -0.738 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ -0.240 & \cdots & 0 \end{bmatrix} * \begin{bmatrix} -0.059 \\ \vdots \\ 0.203 \end{bmatrix}$$

ข้อมูลหลังจากการแปลงแล้วคือ $\begin{bmatrix} 0.878 \\ \vdots \\ 0.513 \end{bmatrix}$

(Haixiang et al., 2017) มีเทคนิคต่าง ๆ ในการสกัดคุณลักษณะต่าง ๆ เช่น Principal Component Analysis (PCA), Singular Value Decomposition (SVD)

1.3. ข้อมูลเครดิตที่ไม่สมดุลอาจส่งผลต่อประสิทธิภาพของตัวแบบได้ แก้ไขได้โดยเทคนิคการสุ่มตัวอย่างซ้ำ (Resampling) เพื่อให้จำนวนข้อมูลในแต่ละประเภทมีจำนวนใกล้เคียงกัน การสุ่มอาจทำได้ใน 3 ลักษณะคือ (1) Under-sampling, (2) Oversampling และ (3) Hybrid approach

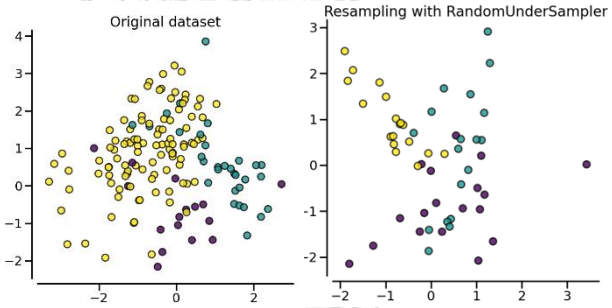
1.3.1. วิธีสังเคราะห์ข้อมูลเพิ่ม (Synthetic Minority Over-Sampling Technique : SMOTE) เป็นการปรับเพิ่มข้อมูลกลุ่มน้อยให้มีจำนวนเพิ่มขึ้น โดยทำการสุ่มค่าข้อมูลที่อยู่ในกลุ่มน้อยขึ้นมา 1 ค่า โดยวิธีเพื่อนบ้านใกล้สุด (K-nearest neighborhood) แล้วทำการคำนวณหาระยะห่างระหว่างจุดด้วยวิธีระยะทางแบบยุคลิด (Euclidean distance) ระหว่างค่าที่สุ่มกับข้อมูลใกล้เคียง และทำการเลือกข้อมูลที่สุ่มขึ้นมาใหม่จากระยะห่างระหว่างจุดที่ใกล้กันมากที่สุด



รูปที่ 2.4 การสังเคราห์ข้อมูลเพิ่ม (SMOTE)

หมายเหตุ. จาก *https://imbalanced-learn.org/stable/over_sampling.html#smote-adasyn*

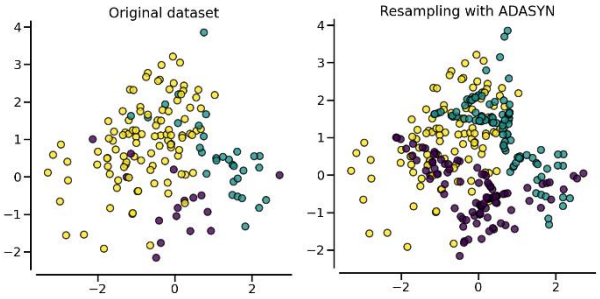
1.3.2. วิธีสุ่มลด (Random Under Sampling : RUS) เป็นเทคนิคที่ใช้จัดการกับข้อมูลที่มีความไม่สมดุล เป็นวิธีการอย่างง่าย โดยการลดจำนวนตัวอย่างข้อมูลแบบสุ่มเพื่อให้มีปริมาณข้อมูลทุกกลุ่มมีจำนวนที่ใกล้เคียงกันแต่ข้อเสียคืออาจทำให้ข้อมูลที่สำคัญหายไปด้วย



รูปที่ 2.5 การสุ่มลด (RUS)

หมายเหตุ. จาก *https://imbalanced-learn.org/stable/under_sampling.html*

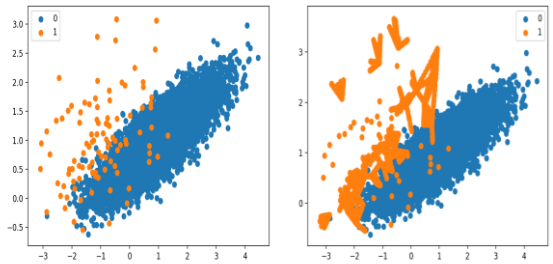
1.3.3. การสุ่มเพิ่มข้อมูลในกลุ่ม (Adaptive Synthetic : ADASYN) เป็นวิธีการสุ่มข้อมูลที่ถูกพัฒนามาจาก SMOTE ซึ่งขั้นตอนการสร้างข้อมูลขึ้นมาใหม่ไม่จำเป็นต้องพิจารณาเฉพาะชุดข้อมูลของกลุ่มน้อยเท่านั้น โดยเทคนิคนี้จะใช้วิธีการแจกแจงแบบถ่วงน้ำหนัก (Weight Distribution) ของชุดข้อมูลในกลุ่มน้อย โดยพิจารณาจากความสำคัญของข้อมูลนั้น ถ้าข้อมูลตัวใดสามารถแบ่งกลุ่มยากก็จะให้น้ำหนักมากกว่า ถ้าข้อมูลตัวใดที่สามารถแบ่งกลุ่มได้ง่ายก็จะให้น้ำหนักน้อย ซึ่งวิธีนี้จะทำให้การสร้างข้อมูลชุดใหม่มีการตัดสินใจแบ่งกลุ่มดีขึ้น



รูปที่ 2.6 การสุ่มเพิ่มข้อมูลในกลุ่ม (ADASYN)

หมายเหตุ. จาก https://imbalanced-learn.org/stable/over_sampling.html#smote-adasyn

1.3.4. วิธีสังเคราะห์ข้อมูลเพิ่มตามขอบข้อมูล (Borderline SMOTE :BSMOTE) เป็นวิธีการสุ่มข้อมูลที่ถูกพัฒนามาจากวิธีสังเคราะห์ข้อมูลเพิ่ม (SMOTE) เทคนิคที่สร้างข้อมูลไปที่บริเวณขอบเท่านั้น โดยเริ่มจากการแบ่งกลุ่มของกลุ่มข้อมูลส่วนน้อย (Minority class) ออกเป็นข้อมูลส่วนน้อย (Minority points), ข้อมูลคลาดเคลื่อน (Noise Points) และข้อมูลที่ขอบ (Border points) โดยใช้การคำนวณข้อมูลที่ใกล้เคียง จากนั้นจะทำสุ่มข้อมูลขึ้นมาใหม่จากข้อมูลที่ขอบเหล่านั้น



รูปที่ 2.7 การสังเคราะห์ข้อมูลเพิ่มตามขอบ (BSMOTE)

หมายเหตุ. จาก https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

(Lenka et al., 2022) ได้นำเสนอการเปรียบเทียบตัวแบบการจำแนกประเภท Ensemble สำหรับข้อมูลคะแนนเครดิต Credit Scoring โดยการเพิ่มเทคนิคการสุ่มตัวอย่างซ้ำ (Resampling) และเทคนิคการคัดเลือกคุณลักษณะ (Feature selection) โดยในส่วนของเทคนิค Resampling ได้ทำการทดลองหลายวิธีคือ SMOTE, ADASYN, BSMOTE และ ROS พบว่า SMOTE ให้ผลลัพธ์เฉลี่ยดีที่สุดและเทคนิค Feature selection ได้ทำการทดลองหลายวิธีคือ Information Gain (IG), Principal Component Analysis (PCA) , Genetic Algorithm (GA) พบว่า Genetic Algorithm (GA) ได้ผลลัพธ์เฉลี่ยที่ดีที่สุด

2.4 การประเมินแบบจำลอง (Model Evaluation)

2.4.1 Confusion Matrix คือตารางแสดงผลการจำแนกด้วยโมเดลที่สร้างขึ้น โดยเปรียบเทียบกับค่ากลุ่มที่แท้จริงของข้อมูล ในรูปแบบของตารางเมทริกซ์ เมื่อ Predicted หมายถึงค่าที่โมเดลทำนาย และ Actual หมายถึงค่ากลุ่มที่แท้จริงของข้อมูล โดยมีรายละเอียดดังนี้

ตารางที่ 2.1 Confusion matrix

		Predicted	
		Positive	Negative
Actual	Positive	True positive (TP)	False negative (FN)
	Negative	False positive (FP)	True negative (TN)

กำหนดให้

TP (True positive) คือ จำนวนตัวอย่างที่เป็น good credit และ ถูกทำนายเป็น good credit

FP (False positive) คือ จำนวนตัวอย่างที่เป็น bad credit และ ถูกทำนายเป็น good credit

FN (False negative) คือ จำนวนตัวอย่างที่เป็น good credit และ ถูกทำนายเป็น bad credit

TN (True negative) คือ จำนวนตัวอย่างที่เป็น bad credit และ ถูกทำนายเป็น bad credit

2.4.2 Accuracy (ACC) หรือความแม่นยำในการจำแนกประเภท ได้แก่ สัดส่วนระหว่างจำนวนข้อมูลทั้งหมดที่จำแนกประเภท ถูกต้องทั้งประเภท Positive และ Negative กับจำนวนข้อมูล ทั้งหมดที่ถูกจำแนกประเภท โดยที่ค่า Accuracy ยิ่งมีค่าเข้าใกล้ 1 แสดงว่าโมเดลมีประสิทธิภาพ ในการทำนายที่แม่นยำขึ้น

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

2.4.3 The area under the curve (AUC) คือการพิจารณาพื้นที่ภายใต้กราฟ ROC (Receiver Operating Characteristic) ซึ่งได้จากการนำผลความแม่นยำในการทำนายของโมเดลมาสร้างเป็นกราฟเส้นโค้งโดยมีจุดเริ่มต้นที่ (0,0) และสิ้นสุดที่จุด (1,1) โดยที่จุดแบ่งเท่ากับ 0 หมายถึงตัวแบบทำนายออกเป็น negative ทั้งหมด เท่ากับค่า TP , FP เป็น 0 ทั้งหมด ในทางตรงกันข้าม ที่จุดแบ่งเท่ากับ 1 หมายถึงตัวแบบทำนายออกมาเป็น positive ทั้งหมด หมายถึง ตัวแบบสามารถทำนาย positive ได้ถูกต้องทั้งหมด

2.4.4 Precision (PRE) ความถูกต้องของการจำแนกประเภทข้อมูลว่าเป็น Positive ได้แก่ สัดส่วนที่ตัวแบบจำแนกประเภทข้อมูลแบบ Positive ได้ถูกต้องเทียบกับจำนวนข้อมูลทั้งหมดที่ถูกจำแนกว่าเป็น Positive

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

2.4.5 The F1-score (F1) เป็นตัวชี้วัดที่พิจารณาจากค่าเฉลี่ยฮาร์โมนิคของ Precision และ Recall

$$F1 - score = \frac{2*Recall*Precision}{Recall + Precision}$$

2.4.6 Brier Score (BS) เป็นเมทริกซ์ที่ใช้ในทางสถิติเพื่อวัดความแม่นยำของการคาดการณ์ความน่าจะเป็น โดยคำนวณจากค่าเฉลี่ยของค่าคลาดเคลื่อนกำลังสอง (Mean-square error) ระหว่างความน่าจะเป็นที่คาดการณ์ได้กับค่าจริง (0 หรือ 1)

$$\mathrm{BS} = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}(\mathrm{P_i} - \mathrm{y_i})^2$$

เมื่อ P_i คือ ค่าทำนาย y_i คือค่าจริงของตัวอย่างที่ i และ N คือ จำนวนตัวอย่าง

2.4.7 Kolmogorov-Smirnov (KS) ใช้เพื่อประเมินความแตกต่างสูงสุดระหว่างฟังก์ชันการแจกแจงความน่าจะเป็นแบบสะสม (Cumulative Distribution Function) ของตัวอย่างที่เป็นบวกและลบ นั่นคือ ผู้ที่ชำระหนี้ตรงเวลา (good credit) และ ผู้ที่ผิดนัดชำระหนี้ (bad credit)

2.5 Statistical significant test

งานวิจัยได้ใช้การทดสอบทางสถิติเพื่อแสดงว่าประสิทธิภาพของตัวแบบมีความแตกต่างกัน การทดลองนี้ใช้สถิติไร้พารามิเตอร์ (nonparametric statistic) เพื่อทดสอบความแตกต่างของตัวแบบทั้งหมดในแต่ละค่าอัตราส่วนความไม่สมดุล ใช้สถิติทดสอบลำดับที่โดยเครื่องหมายของวิลค์อกซัน (Wilcoxon-Signed Rank test) ทดสอบความแตกต่างของประสิทธิภาพระหว่างสองกลุ่ม โดยมีวิธีการคือ จัดอันดับความแตกต่างโดยไม่คำนึงถึงเครื่องหมายจากค่าน้อยสุดไปหามากสุด กรณีที่ผลต่างเท่ากันหลายคู่จะใช้อันดับเฉลี่ยและถ้าผลต่างเป็นศูนย์จะไม่นำมาพิจารณาในการจัดอันดับ จากนั้นแยกอันดับตามเครื่องหมายเป็นกลุ่มอันดับเป็นบวกและกลุ่มที่อันดับเป็นลบ จากนั้นหาผลรวมในแต่ละกลุ่มพร้อมทั้งพิจารณาให้ค่าของผลรวมของอันดับที่น้อยกว่า (โดยไม่คิดเครื่องหมาย) เป็นค่า T ซึ่งถือเป็นค่าที่ได้จากการคำนวณ

$$T = min\left|\sum R_i^+ , \sum R_i^- \right|$$

เกณฑ์ในการตัดสินค่าที่ได้จากการคำนวณจะปฏิเสธสมมติฐาน $\mathbf{H_0}$ เมื่อค่า $\mathbf{T}$ ที่คำนวณได้มีค่าน้อยกว่าหรือเท่ากับค่าวิกฤต $\mathbf{T}$ ที่ระดับนัยสำคัญ 0.05 และ n (ซึ่งในการศึกษานี้มีจำนวนตัวชี้วัด n=6) ที่ได้จากการเปิดตาราง Wilcoxon Matched Pairs Sign-Rank test

และใช้การทดสอบ Friedman ทดสอบความแตกต่างของประสิทธิภาพมากกว่า 2 กลุ่มขึ้นไป เช่น ทดสอบความแตกต่างของตัวแบบทั้งหมดในแต่ละค่าอัตราส่วนความไม่สมดุล กำหนดให้ตัวชี้วัด Accuracy (ACC), The area under the curve (AUC), The F1-score (F1), Precision (PRE), Kolmogorov-Smirnov (KS) และ Brier Score (BS) เป็นบล็อกและตัวแบบต่างๆเป็นทรีทเมนท์ โดยที่ ranked 1 จะถูกกำหนดให้กับตัวแบบที่ดีที่สุดในแต่ละทรีทเมนท์ , ranked 2 เป็นอันดับที่สอง รองลงมา ในแต่ละทรีทเมนท์ตามลำดับ การทดสอบ Friedman test จะดำเนินการใน K ตัวแบบ โดยใช้ chi-square ที่องศาอิสระ K-1 โดยคำนวณจาก rank ของแต่ละ classifier j

$$X^2{}_F = [\frac{12}{bk(k+1)}\sum_{j=1}^{k} R_j^2] - 3b(k+1)$$

เมื่อ b คือจำนวนตัวชี้วัด, k คือจำนวนตัวแบบ และ $\mathrm{R_j}$ คือ อันดับของแต่ละตัวแบบ

เมื่อการทดสอบ Friedman ถูกปฏิเสธสมมติฐานว่าง นั่นหมายถึงประสิทธิภาพของตัวแบบแตกต่างกันในแต่ละค่าอัตราส่วนความไม่สมดุล จากนั้น Post hoc test สามารถนำมาใช้เพื่อกำหนดความแตกต่างอย่างมีนัยสำคัญในประสิทธิของตัวแบบการจำแนกประเภทแต่ละคู่ ตามการทดสอบของ Nemenyi ประสิทธิภาพของตัวแบบจำแนกประเภทสองตัวแบบหรือหลายตัวแบบจะแตกต่างกันอย่างมีนัยสำคัญเมื่ออันดับเฉลี่ย (average rank) แตกต่างกันอย่างน้อย CD (Critical difference) ดังสมการ

$$CD = q_{\alpha,\infty,K}\sqrt{\frac{K(K+1)}{12N}}$$

เมื่อ $q_{\alpha,\infty,K}$ คือ ค่าจากตาราง Studentized range statistic , K คือ จำนวนตัวแบบ และ N คือ จำนวนข้อมูล

บทที่ 3
วิธีการดำเนินการวิจัย

การวิจัยนี้ทำการศึกษาโดยจำลองข้อมูลเครดิตเยอรมันในอัตราส่วนความไม่สมดุลที่แตกต่างกันเพื่อเปรียบเทียบประสิทธิภาพของการเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนาน (Bagging Heterogeneous Ensemble), ตัวแบบการสุ่มป่าไม้ (Random Forest) และตัวแบบพื้นฐาน 4 ตัวแบบ คือ Logistic Regression (LR), Decision Tree (DT), K-Nearest Neighbors (KNN) และ Support Vector Machine (SVM) โดยวัดประสิทธิภาพตัวแบบด้วยตัวชี้วัด Accuracy (ACC), The area under the curve (AUC), The F1-score (F1), Precision (PRE), Kolmogorov-Smirnov (KS) และ Brier Score (BS)

การเตรียมข้อมูลและสร้างชุดข้อมูลที่แตกต่างกัน (Data Preparation and Dataset Version Generation)

ข้อมูลเครดิตเยอรมันมีจำนวนตัวแปรต้น 20 ตัวแปร ดังนี้

ชื่อคุณลักษณะ	ประเภทของข้อมูล	
สถานะของบัญชี (status of existing checking)	ข้อมูลเชิงคุณภาพ	A11: น้อยกว่า 0 DM A12: มากกว่า 0 DM แต่น้อยกว่า 200 DM A13: มากกว่า 200 DM/ มีหนังสือรับรองเงินเดือนอย่างน้อย 1 ปี A14: ไม่มีบัญชีธนาคาร
ระยะเวลา (duration in month)	ข้อมูลเชิงปริมาณ	
ประวัติของเครดิต (credit history)	ข้อมูลเชิงคุณภาพ	A30: ไม่มีประวัติการใช้เครดิต/เงินกู้ A31: เคยมีการขอเงินกู้ที่ธนาคารนี้และชำระตรงตามกำหนดไม่มีค้างชำระ A32: เคยมีการใช้เครดิตกับธนาคารอื่นๆ และชำระตรงตามกำหนดไม่มีค้างชำระ A33: เคยชำระหนี้ล่าช้าในอดีต

		A34: มีเครดิตที่อยู่กับธนาคารอื่นๆ ที่มีประวัติการชำระหนี้ล่าช้า
วัตถุประสงค์ (purpose)	ข้อมูลเชิงคุณภาพ	A40: รถยนต์ใหม่ A41: รถยนต์มือสอง A42: เฟอร์นิเจอร์/ของตกแต่งบ้าน A43: ทีวี A44: เครื่องซักผ้า/ตู้เย็น/เตาอบ A45: ซ่อมแซมบ้าน A46: การศึกษา A47: การเที่ยวพักผ่อน A48: การฝึกอบรม A49: ธุรกิจ A410: อื่นๆ
วงเงิน (credit amount)	ข้อมูลเชิงปริมาณ	
บัญชีออมทรัพย์/พันธบัตร (saving account/bonds)	ข้อมูลเชิงคุณภาพ	A61: น้อยกว่า 100 DM A62: มากกว่า 100 DM แต่น้อยกว่า 500 DM A63: มากกว่า 500 DM แต่น้อยกว่า 1,000 DM A64: มากกว่า 1,000 DM A65: ไม่มีข้อมูล/ไม่มีบัญชีออมทรัพย์
ระยะเวลาของการทำงานปัจจุบัน (present employment since)	ข้อมูลเชิงคุณภาพ	A71: ว่างงาน A72: น้อยกว่า 1 ปี A73: มากกว่า 1 ปีแต่น้อยกว่า 4 ปี A74: มากกว่า 4 ปีแต่น้อยกว่า 7 ปี A75: มากกว่า 7 ปี
อัตราการผ่อนชำระเป็นเปอร์เซ็นต์ของรายได้ (installment rate in percentage of disposable income)	ข้อมูลเชิงปริมาณ	
สถานะและเพศ (personal status and sex)	ข้อมูลเชิงคุณภาพ	A91: เพศชายและเคยหย่าหรือแยกกันอยู่ A92: เพศหญิงและเคยหย่า แยกกันหรือแต่งงาน A93: เพศชายและโสด

		A94: เพศชายและแต่งงาน A95: เพศหญิงและโสด
ลูกหนี้/ผู้ค้ำประกัน (other debtors/ guarantors)	ข้อมูลเชิงคุณภาพ	A101: ไม่มี A102: ผู้สมัครร่วมกัน A103: ผู้ค้ำประกัน
ระยะเวลาการพักอาศัยของที่อยู่ปัจจุบัน (present residence since)	ข้อมูลเชิงปริมาณ	
ทรัพย์สิน (property)	ข้อมูลเชิงคุณภาพ	A121: อสังหาริมทรพย์ A122: เงินกองทุนประกันชีวิตหรือสัญญาอื่นๆกับสกกรณีออมทรัพย์ A123: รถยนต์หรือทรัพย์สินอื่นๆ A124: ไม่ระบุ/ไม่มีทรัพย์สิน
อายุ (age in years)	ข้อมูลเชิงปริมาณ	
แผนการผ่อนชำระ (other installment plans)	ข้อมูลเชิงคุณภาพ	A141: ธนาคาร A142: ร้านค้า A143: ไม่ระบุ
ที่อยู่อาศัย (housing)	ข้อมูลเชิงคุณภาพ	A151: เช่า A152: เป็นเจ้าของ A153: ฟรี
จำนวนเครดิตที่มีอยู่ของธนาคารนี้ (number of existing credits at this bank)	ข้อมูลเชิงปริมาณ	
งาน (jobs)	ข้อมูลเชิงคุณภาพ	A171: ว่างงาน A172: รับจ้าง/เกษตรกร A173: พนักงานบริษัท / ข้าราชการ A174: ผู้จัดการ/ธุรกิจส่วนตัว/ เจ้าหน้าที่ระดับสูง
จำนวนผู้รับผิดชอบในการดูแล (number of people being liable to provide maintenance for)	ข้อมูลเชิงปริมาณ	
โทรศัพท์ (telephone)	ข้อมูลเชิงคุณภาพ	A191: ไม่มีโทรศัพท์ A192: มี และลงทะเบียนในชื่อลูกค้า
แรงงานต่างด้าว (Foreign worker)	ข้อมูลเชิงคุณภาพ	A201: ใช่ A202: ไม่ใช่

ขั้นตอนการเตรียมข้อมูลเป็นอีกขั้นตอนหนึ่งที่สำคัญในกระบวนการเตรียมข้อมูลเพื่อนำไปวิเคราะห์โดยมีตัวแปรต้นทั้งหมด 20 ตัวแปร แบ่งเป็น 2 ประเภทคือ (1) ข้อมูลเชิงปริมาณ และ (2) ข้อมูลเชิงคุณภาพ สำหรับข้อมูลที่มีประเภทเป็นข้อมูลเชิงปริมาณจะทำการแปลงข้อมูลด้วยวิธี Standardization

$$\acute{x} = \frac{x - \mu}{\sigma}$$

โดยที่ $\acute{x}$ คือตัวแปรหลังจากทำ standardizing, x คือตัวแปร, μ คือค่าเฉลี่ยของตัวแปรและ σ คือค่าส่วนเบี่ยงเบนมาตรฐานของตัวแปร และข้อมูลเชิงคุณภาพจะทำการแปลงข้อมูลด้วยวิธีโปรแกรมสอนการเข้ารหัสข้อมูล (Binary Encoding) หลังจากการแปลงข้อมูลแล้วทำให้มีจำนวนมิติของข้อมูลเป็น 1000 x 49 (1000 แถว 49 หลัก)

การสร้างชุดข้อมูลไม่สมดุล (imbalance datasets) ที่มีค่าอัตราส่วนความไม่สมดุล (Imbalance Ratio : IR) ที่แตกต่างกันโดยการสุ่มลบข้อมูลในส่วนของชุดข้อมูลเรียนรู้ (Training set) ที่เป็นข้อมูลกลุ่มน้อยจนได้ค่าอัตราส่วนความไม่สมดุลที่กำหนด คือ

- อัตราส่วนความไม่สมดุลต่ำ (IR = 2.3) คือ ผู้ที่ชำระหนี้ตรงเวลา 700 คน หารด้วยผู้ที่ผิดนัดชำระหนี้ 300 คน
- อัตราส่วนความไม่สมดุลกลาง (IR = 10) คือ ผู้ที่ชำระหนี้ตรงเวลา 700 คน หารด้วยผู้ที่ผิดนัดชำระหนี้ 70 คน
- อัตราส่วนความไม่สมดุลสูง (IR =14) คือ ผู้ที่ชำระหนี้ตรงเวลา 700 คน หารด้วยผู้ที่ผิดนัดชำระหนี้ 50 คน

โดยค่า IR = 14 เป็นค่าอัตราส่วนความไม่สมดุลสูงที่สุดที่ทำให้ไม่เกิดปัญหาในการสร้างตัวแบบสำหรับข้อมูลเครดิตเยอรมัน จากนั้นแบ่งข้อมูลออกเป็น 2 ส่วนสำหรับในแต่ละชุดข้อมูลไม่สมดุลคือข้อมูลที่ใช้สำหรับการเรียนรู้ (Train set) และข้อมูลที่ใช้สำหรับการทดสอบ (Test set) โดยแบ่งในอัตราส่วน 80% และ 20% ตามลำดับ

สร้างตัวแบบการเรียนรู้กลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนาน (Bagging Heterogeneous Ensemble)

ขั้นตอนการสร้างตัวแบบการเรียนรู้กลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนานสามารถแบ่งออกเป็น 2 ขั้นตอน คือ 1) Bootstrap คือการสุ่มตัวอย่างจากข้อมูลชุดเรียนรู้ในอัตราส่วน 95% ของ

ข้อมูลทั้งหมดทำให้เกิดชุดข้อมูลย่อยที่แตกต่างกันในการเรียนรู้แต่ละรอบ โดยทำการ bootstrap 20 รอบ ในแต่ละรอบจะสร้างตัวแบบการเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกัน (Bagging Heterogeneous Ensemble) ด้วยตัวแบบพื้นฐาน 4 ตัวแบบคือ Logistic Regression (LR), K-Nearest Neighbors (KNN), Decision Tree (DT) และ Support Vector Machine (SVM) จากนั้นรวมผลลัพธ์ของตัวแบบการเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันด้วยวิธี Soft Voting และขั้นตอนที่ 2) Aggregation จะเป็นการรวมผลลัพธ์ที่ได้จากตัวแบบการเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันทั้งหมด 20 ตัวแบบที่ได้จากขั้นตอนที่ 1 ด้วยวิธี The highest mean predicted probability ซึ่งเป็นวิธีจำแนกประเภทจากการคำนวณค่าเฉลี่ยความน่าจะเป็นสูงที่สุด

ตัวแบบในการจำแนกประเภทที่ใช้ในการวิจัย (Classifier models)

ในงานวิจัยนี้ได้ใช้ตัวแบบการจำแนกประเภทเพื่อเปรียบเทียบกับการเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนาน 5 ตัวแบบคือ ตัวแบบการสุ่มป่าไม้ (RF), Logistic Regression (LR), K-Nearest Neighbors (KNN), Decision Tree (DT) และ Support Vector Machine (SVM) โดยแต่ละตัวแบบที่นำมาเปรียบเทียบจะทำการหาพารามิเตอร์ที่ดีที่สุดโดยใช้ RandomizedSearchCV ในไลบรารีของ sklearn เพื่อนำพารามิเตอร์ที่ดีที่สุดนี้ไปสร้างตัวแบบในการจำแนกประเภทและวัดประสิทธิภาพด้วยตัวชี้วัดต่างๆ โดยขอบเขตของการหาพารามิเตอร์ที่ดีที่สุดของแต่ละตัวแบบ มีรายละเอียดดังนี้

1. กำหนดขอบเขตพารามิเตอร์ของตัวแบบการสุ่มป่าไม้ (Random Forest) ดังตารางที่ 3.1

ตารางที่ 3.1 ขอบเขตพารามิเตอร์สำหรับการสร้างตัวแบบการสุ่มป่าไม้

ลำดับที่	พารามิเตอร์	ขอบเขต
1	criterion	['gini', 'entropy', 'log_loss']
2	max_features	['auto', 'sqrt', 'log2']
3	min_samples_split	[2, 5, 10]

2. กำหนดขอบเขตพารามิเตอร์ของตัวแบบการถดถอยโลจิสติก (Logistic Regression) ดังตารางที่ 3.2

ตารางที่ 3.2 ขอบเขตพารามิเตอร์สำหรับการสร้างตัวแบบการถดถอยโลจิสติก

ลำดับที่	พารามิเตอร์	ขอบเขต
1	penalty	['l1', 'l2', 'elasticnet', 'none']
2	solver	['newton-cg', 'lbgfs', 'liblinear', 'sag', 'saga']
3	C	[100, 10, 1.0, 0.1, 0.01]

3. กำหนดขอบเขตพารามิเตอร์ของตัวแบบเพื่อนบ้านใกล้สุด (K-Nearest Neighbors) ดังตารางที่ 3.3

ตารางที่ 3.3 ขอบเขตพารามิเตอร์สำหรับการสร้างตัวแบบเพื่อนบ้านใกล้สุด

ลำดับที่	พารามิเตอร์	ขอบเขต
1	n_neighbors	range (10,20)
2	leaf_size	range (25,35)
3	weights	['uniform', 'distance']
4	metric	['euclidean', 'minkowski', 'cityblock']

4. กำหนดขอบเขตพารามิเตอร์ของตัวแบบต้นไม้ตัดสินใจ (Decision Tree) ดังตารางที่ 3.4

ตารางที่ 3.4 ขอบเขตพารามิเตอร์สำหรับการสร้างตัวแบบต้นไม้ตัดสินใจ

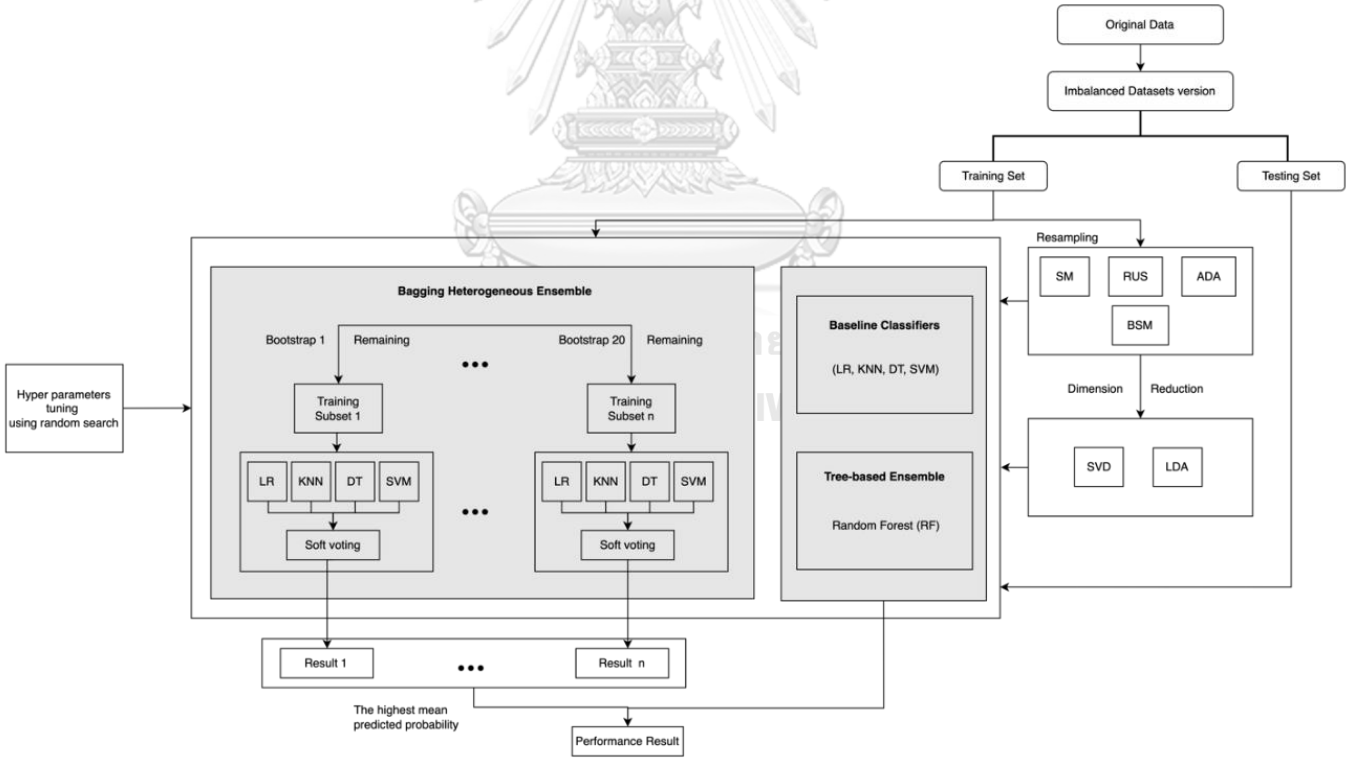
ลำดับที่	พารามิเตอร์	ขอบเขต
1	ccp_alpha	[0, 0.1, 0.01]
2	min_samples_leaf	[1, 2, 3, 4]
3	criterion	['gini', 'entropy']
4	max_depth	[None, 5, 10, 15]
5	max_feature	['auto', 'sqrt', 'log2']

5. กำหนดขอบเขตพารามิเตอร์ของตัวแบบซัพพอร์ตเวคเตอร์แมชชีน (Support Vector Machine) ดังตารางที่ 3.5

ตารางที่ 3.5 ขอบเขตพารามิเตอร์สำหรับการสร้างตัวแบบซัพพอร์ตเวคเตอร์แมชชีน

ลำดับที่	พารามิเตอร์	ขอบเขต
1	C	[100, 10, 1.0, 0.1, 0.01, 0.001, 0.0001]
2	gamma	['scale', 'auto']
3	kernel	['linear', 'poly', 'rbf', 'sigmoid']
4	class_weight	['balanced', 'None']

การเปรียบเทียบตัวแบบ (Classifier evaluation)



รูปที่ 3.8 ขั้นตอนการทำงาน

จากรูปที่ 3.1 สามารถแบ่งออกเป็น 4 การศึกษา โดยแต่ละการศึกษาจะวัดประสิทธิภาพด้วยตัวชี้วัด Accuracy (ACC), The area under the curve (AUC), F1-score (F1), Precision (PRE), Brier score (BS) และ Kolmogorov-Smirnov (KS)

กำหนดให้ (1) Training set $\mathrm{s} = \{(\mathrm{x_i}, \mathrm{y_i}),\ \mathrm{i} = 1, 2, \ldots, \mathrm{n}\}$,

(2) Test set $\mathrm{t} = \{(\mathrm{xt_j}, \mathrm{yt_j}),\ \mathrm{j} = 1, 2, \ldots, \mathrm{m}\}$

(3) $\mathrm{Performance} = \{\mathrm{ACC}, \mathrm{AUC}, \mathrm{F1}, \mathrm{PRE}, \mathrm{BS}, \mathrm{KS}\}$ ประเภทของ y_i และ yt_i ประกอบไปด้วย 0 หรือ 1

การศึกษาที่ 1 : เปรียบเทียบตัวแบบการเรียนรู้แบบรวมกลุ่ม (Bagging Heterogeneous Ensemble), Random Forest (RF) และตัวแบบการจำแนกประเภทพื้นฐาน 4 ตัวแบบ ได้แก่ Logistic Regression (LR), K-Nearest Neighbors (KNN), Decision Tree (DT), Support Vector Machine (SVM) ในแต่ละค่าอัตราส่วนความไม่สมดุล (IR)

<u>ขั้นตอนการทำงาน</u>

1) นำข้อมูลชุดเรียนรู้มาสร้างตัวแบบเพื่อเปรียบเทียบประสิทธิภาพ ประกอบด้วย (1) Bagging Hetero (2) LR (3) KNN (4) DT (5) SVM (6) RF

$$\mathrm{Model_{Bagging\ Hetero}} = \mathrm{train(Bagging\ Hetero, x, y)}$$
$$\mathrm{Model_{LR}} = \mathrm{train(LR, x, y)}$$
$$\mathrm{Model_{KNN}} = \mathrm{train(KNN, x, y)}$$
$$\mathrm{Model_{DT}} = \mathrm{train(DT, x, y)}$$
$$\mathrm{Model_{SVM}} = \mathrm{train(SVM, x, y)}$$
$$\mathrm{Model_{RF}} = \mathrm{train(RF, x, y)}$$

$$\mathrm{y\_pre_{Bagging\ Hetero}},\ \mathrm{y\_prob_{Bagging\ Hetero}} = \mathrm{test(Model_{Bagging\ Hetero}, xt)}$$
$$\mathrm{y\_pre_{LR}}, \mathrm{y\_prob_{LR}} = \mathrm{test(Model_{LR}, xt)}$$
$$\mathrm{y\_pre_{KNN}}, \mathrm{y\_prob_{KNN}} = \mathrm{test(Model_{KNN}, xt)}$$
$$\mathrm{y\_pre_{DT}}, \mathrm{y\_prob_{DT}} = \mathrm{test(Model_{DT}, xt)}$$
$$\mathrm{y\_pre_{SVM}}, \mathrm{y\_prob_{SVM}} = \mathrm{test(Model_{SVM}, xt)}$$
$$\mathrm{y\_pre_{RF}}, \mathrm{y\_prob_{RF}} = \mathrm{test(Model_{RF}, xt)}$$

2) หลังจากนั้นนำตัวแบบที่ได้ไปวัดประสิทธิภาพด้วย Test set โดยการคำนวณตัวชี้วัด ACC, AUC, F1, PRE, KS และ BS

$$\mathrm{Performance_{Bagging\ Hetero}} = \mathrm{Performance(y\_pre_{Bagging\ Hetero}, y\_prob_{Bagging\ Hetero}, yt)}$$

$\text{Performance}_{\text{LR}} = \text{Performance}(\text{y\_pre}_{\text{LR}}, \text{y\_prob}_{\text{LR}}, \text{yt})$
$\text{Performance}_{\text{KNN}} = \text{Performance}(\text{y\_pre}_{\text{KNN}}, \text{y\_prob}_{\text{KNN}}, \text{yt})$
$\text{Performance}_{\text{DT}} = \text{Performance}(\text{y\_pre}_{\text{DT}}, \text{y\_prob}_{\text{DT}}, \text{yt})$
$\text{Performance}_{\text{SVM}} = \text{Performance}(\text{y\_pre}_{\text{SVM}}, \text{y\_prob}_{\text{SVM}}, \text{yt})$
$\text{Performance}_{\text{RF}} = \text{Performance}(\text{y\_pre}_{\text{RF}}, \text{y\_prob}_{\text{RF}}, \text{yt})$

3) หาอันดับตามแต่ละตัวชี้วัดโดยเรียงจากมากไปน้อย จากนั้นหาอันดับเฉลี่ย (mean rank) ในแต่ละตัวแบบ

$\text{Rank}_{\text{Bagging Hetero}} = \text{Rank}(\text{Performance}_{\text{Bagging Hetero}})$
$\text{Rank}_{\text{LR}} = \text{Rank}(\text{Performance}_{\text{LR}})$
$\text{Rank}_{\text{KNN}} = \text{Rank}(\text{Performance}_{\text{KNN}})$
$\text{Rank}_{\text{DT}} = \text{Rank}(\text{Performance}_{\text{DT}})$
$\text{Rank}_{\text{SVM}} = \text{Rank}(\text{Performance}_{\text{SVM}})$
$\text{Rank}_{\text{RF}} = \text{Rank}(\text{Performance}_{\text{RF}})$

Mean_Rank_ACC
$= \text{Mean}_{\text{ACC}}(\text{Rank}_{\text{Bagging Hetero}}, \text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}}, \text{Rank}_{\text{RF}})$
Mean_Rank_AUC
$= \text{Mean}_{\text{AUC}}(\text{Rank}_{\text{Bagging Hetero}}, \text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}}, \text{Rank}_{\text{RF}})$
Mean_Rank_F1
$= \text{Mean}_{\text{F1}}(\text{Rank}_{\text{Bagging Hetero}}, \text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}}, \text{Rank}_{\text{RF}})$
Mean_Rank_PRE
$= \text{Mean}_{\text{PRE}}(\text{Rank}_{\text{Bagging Hetero}}, \text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}}, \text{Rank}_{\text{RF}})$
Mean_Rank_BS
$= \text{Mean}_{\text{BS}}(\text{Rank}_{\text{Bagging Hetero}}, \text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}}, \text{Rank}_{\text{RF}})$
Mean_Rank_KS
$= \text{Mean}_{\text{KS}}(\text{Rank}_{\text{Bagging Hetero}}, \text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}}, \text{Rank}_{\text{RF}})$

4) คำนวณหาอันดับเฉลี่ยเพื่อหาตัวแบบที่มีประสิทธิภาพมากที่สุด

ผลลัพธ์ : $\text{Min}(\text{Mean}(\text{Mean\_Rank}))$

การศึกษาที่ 2 : เปรียบเทียบวิธีการลดมิติข้อมูล (dimensionality reduction) ได้แก่ Singular Value Decomposition (SVD), Linear Discriminant Analysis (LDA) ด้วยตัวแบบการจำแนกประเภทพื้นฐาน 4 ตัวแบบ ในแต่ละค่าอัตราส่วนความไม่สมดุล (IR)

ขั้นตอนการทำงาน

1) ทำการแปลงข้อมูลชุดเรียนรู้ด้วยเทคนิคการลดมิติข้อมูลประกอบไปด้วย 2 วิธีคือ (1) SVD และ (2) LDA

$$x_{SVD} = Transform_{SVD}(x), xt_{SVD} = Transform_{SVD}(xt)$$
$$x_{LDA} = Transform_{LDA}(x, y), xt_{LDA} = Transform_{LDA}(xt)$$

2) นำข้อมูลที่แปลงจากข้อที่ 1 มาสร้างตัวแบบพื้นฐานทั้ง 4 ตัวแบบ (LR, KNN, DT และ SVM) เพื่อเปรียบเทียบประสิทธิภาพเทคนิคการลดมิติข้อมูล

$$Model_{SVD\_LR} = train(LR, x_{SVD}, y), Model_{LDA\_LR} = train(LR, x_{LDA}, y)$$
$$Model_{SVD\_KNN} = train(KNN, x_{SVD}, y), Model_{LDA\_KNN} = train(KNN, x_{LDA}, y)$$
$$Model_{SVD\_DT} = train(DT, x_{SVD}, y), Model_{LDA\_DT} = train(DT, x_{LDA}, y)$$
$$Model_{SVD\_SVM} = train(SVM, x_{SVD}, y), Model_{LDA\_SVM} = train(SVM, x_{LDA}, y)$$

$$y\_pre_{SVD\_LR}, y\_prob_{SVD\_LR} = test(Model_{SVD\_LR}, xt)$$
$$y\_pre_{LDA\_LR}, y\_prob_{LDA\_LR} = test(Model_{LDA\_LR}, xt)$$
$$y\_pre_{SVD\_KNN}, y\_prob_{SVD\_KNN} = test(Model_{SVD\_KNN}, xt)$$
$$y\_pre_{LDA\_KNN}, y\_prob_{LDA\_KNN} = test(Model_{LDA\_KNN}, xt)$$
$$y\_pre_{SVD\_DT}, y\_prob_{SVD\_DT} = test(Model_{SVD\_DT}, xt)$$
$$y\_pre_{LDA\_DT}, y\_prob_{LDA\_DT} = test(Model_{LDA\_DT}, xt)$$
$$y\_pre_{SVD\_SVM}, y\_prob_{SVD\_SVM} = test(Model_{SVD\_SVM}, xt)$$
$$y\_pre_{LDA\_SVM}, y\_prob_{LDA\_SVM} = test(Model_{LDA\_SVM}, xt)$$

3) หลังจากนั้นนำตัวแบบที่มีเทคนิคการลดมิติข้อมูลที่ได้ไปวัดประสิทธิภาพด้วย Test set โดยการคำนวณตัวชี้วัด ACC, AUC, F1, PRE, KS และ BS

$$Performance_{SVD\_LR} = Performance(y\_pre_{SVD\_LR}, y\_prob_{SVD\_LR}, yt)$$
$$Performance_{LDA\_LR} = Performance(y\_pre_{LDA\_LR}, y\_prob_{LDA\_LR}, yt)$$
$$Performance_{SVD\_KNN} = Performance(y\_pre_{SVD\_KNN}, y\_prob_{SVD\_KNN}, yt)$$
$$Performance_{LDA\_KNN} = Performance(y\_pre_{LDA\_KNN}, y\_prob_{LDA\_KNN}, yt)$$
$$Performance_{SVD\_DT} = Performance(y\_pre_{SVD\_DT}, y\_prob_{SVD\_DT}, yt)$$
$$Performance_{LDA\_DT} = Performance(y\_pre_{LDA\_DT}, y\_prob_{LDA\_DT}, yt)$$
$$Performance_{SVD\_SVM} = Performance(y\_pre_{SVD\_SVM}, y\_prob_{SVD\_SVM}, yt)$$
$$Performance_{LDA\_SVM} = Performance(y\_pre_{LDA\_SVM}, y\_prob_{LDA\_SVM}, yt)$$

4) หาอันดับตามแต่ละตัวชี้วัดในแต่ละตัวแบบของเทคนิคการลดมิติข้อมูล (LDA และ SVD) โดยเรียงจากมากไปน้อย จากนั้นหาอันดับเฉลี่ย (mean rank) ในแต่ละเทคนิคการลดมิติข้อมูล

$$Rank_{LR} = Rank(Performance_{SVD\_LR}, Performance_{LDA\_LR})$$
$$Rank_{KNN} = Rank(Performance_{SVD\_KNN}, Performance_{LDA\_KNN})$$

$$\text{Rank}_{\text{DT}} = \text{Rank}(\text{Performance}_{\text{SVD\_DT}}, \text{Performance}_{\text{LDA\_DT}})$$
$$\text{Rank}_{\text{SVM}} = \text{Rank}(\text{Performance}_{\text{SVD\_SVM}}, \text{Performance}_{\text{LDA\_SVM}})$$

$$\text{Mean\_Rank\_ACC}_{\text{SVD}}, \text{Mean\_Rank\_ACC}_{\text{LDA}} = \text{Mean}_{\text{ACC}}(\text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}})$$
$$\text{Mean\_Rank\_AUC}_{\text{SVD}}, \text{Mean\_Rank\_AUC}_{\text{LDA}} = \text{Mean}_{\text{AUC}}(\text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}})$$
$$\text{Mean\_Rank\_F1}_{\text{SVD}}, \text{Mean\_Rank\_F1}_{\text{LDA}} = \text{Mean}_{\text{F1}}(\text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}})$$
$$\text{Mean\_Rank\_PRE}_{\text{SVD}}, \text{Mean\_Rank\_PRE}_{\text{LDA}} = \text{Mean}_{\text{PRE}}(\text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}})$$
$$\text{Mean\_Rank\_BS}_{\text{SVD}}, \text{Mean\_Rank\_BS}_{\text{LDA}} = \text{Mean}_{\text{BS}}(\text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}})$$
$$\text{Mean\_Rank\_KS}_{\text{SVD}}, \text{Mean\_Rank\_KS}_{\text{LDA}} = \text{Mean}_{\text{KS}}(\text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}})$$

5) คำนวณหาอันดับเฉลี่ยเพื่อหาเทคนิคการลดมิติข้อมูลที่มีประสิทธิภาพมากที่สุด

ผลลัพธ์ : $\text{Min}(\text{Mean}(\text{Mean\_Rank}_{\text{SVD}}), \text{Mean}(\text{Mean\_Rank}_{\text{LDA}}))$

การศึกษาที่ 3 : เปรียบเทียบวิธีการสุ่มตัวอย่างซ้ำ (resampling) ได้แก่ Systematic Minority Over-Sampling Technique (SM), Random Under-Sampling (RUS), Adaptive Synthetic (ADA) , Borderline SMOTE (BSM) ด้วยตัวแบบการจำแนกประเภทพื้นฐาน 4 ตัวแบบ ในแต่ละค่าอัตราส่วนความไม่สมดุล (IR)

<u>ขั้นตอนการทำงาน</u>

1) ทำการเพิ่มข้อมูลชุดเรียนรู้ด้วยเทคนิคการสุ่มตัวอย่างซ้ำประกอบไปด้วย 4 วิธีคือ (1) SM (2) RUS (3) ADA และ (4) BSM

$$\text{x}_{\text{SM}} = \text{SM}(\text{x}, \text{y})$$
$$\text{x}_{\text{RUS}} = \text{RUS}(\text{x}, \text{y})$$
$$\text{x}_{\text{ADA}} = \text{ADA}(\text{x}, \text{y})$$
$$\text{x}_{\text{BSM}} = \text{BSM}(\text{x}, \text{y})$$

2) นำข้อมูลที่ได้จากข้อที่ 1 มาสร้างตัวแบบพื้นฐานทั้ง 4 ตัวแบบ (LR, KNN, DT และ SVM) เพื่อเปรียบเทียบประสิทธิภาพเทคนิคการสุ่มตัวอย่างซ้ำ

$$\text{Model}_{\text{SM\_LR}} = \text{train}(\text{LR}, \text{x}_{\text{SM}}, \text{y}), \text{Model}_{\text{SM\_KNN}} = \text{train}(\text{KNN}, \text{x}_{\text{SM}}, \text{y})$$
$$\text{Model}_{\text{SM\_DT}} = \text{train}(\text{DT}, \text{x}_{\text{SM}}, \text{y}), \text{Model}_{\text{SM\_SVM}} = \text{train}(\text{SVM}, \text{x}_{\text{SM}}, \text{y})$$
$$\text{Model}_{\text{RUS\_LR}} = \text{train}(\text{LR}, \text{x}_{\text{RUS}}, \text{y}), \text{Model}_{\text{RUS\_KNN}} = \text{train}(\text{KNN}, \text{x}_{\text{RUS}}, \text{y})$$

$$\text{Model}_{\text{RUS\_DT}} = \text{train}(\text{DT}, \text{x}_{\text{RUS}}, \text{y}), \text{Model}_{\text{RUS\_SVM}} = \text{train}(\text{SVM}, \text{x}_{\text{RUS}}, \text{y})$$
$$\text{Model}_{\text{ADA\_LR}} = \text{train}(\text{LR}, \text{x}_{\text{ADA}}, \text{y}), \text{Model}_{\text{ADA\_KNN}} = \text{train}(\text{KNN}, \text{x}_{\text{ADA}}, \text{y})$$
$$\text{Model}_{\text{ADA\_DT}} = \text{train}(\text{DT}, \text{x}_{\text{ADA}}, \text{y}), \text{Model}_{\text{ADA\_SVM}} = \text{train}(\text{SVM}, \text{x}_{\text{ADA}}, \text{y})$$
$$\text{Model}_{\text{BSM\_LR}} = \text{train}(\text{LR}, \text{x}_{\text{BSM}}, \text{y}), \text{Model}_{\text{BSM\_KNN}} = \text{train}(\text{KNN}, \text{x}_{\text{BSM}}, \text{y})$$
$$\text{Model}_{\text{BSM\_DT}} = \text{train}(\text{DT}, \text{x}_{\text{BSM}}, \text{y}), \text{Model}_{\text{BSM\_SVM}} = \text{train}(\text{SVM}, \text{x}_{\text{BSM}}, \text{y})$$

$$\text{y\_pre}_{\text{SM\_LR}}, \text{y\_prob}_{\text{SM\_LR}} = \text{test}(\text{Model}_{\text{SM\_LR}}, \text{xt})$$
$$\text{y\_pre}_{\text{SM\_KNN}}, \text{y\_prob}_{\text{SM\_KNN}} = \text{test}(\text{Model}_{\text{SM\_KNN}}, \text{xt})$$
$$\text{y\_pre}_{\text{SM\_DT}}, \text{y\_prob}_{\text{SM\_DT}} = \text{test}(\text{Model}_{\text{SM\_DT}}, \text{xt})$$
$$\text{y\_pre}_{\text{SM\_SVM}}, \text{y\_prob}_{\text{SM\_SVM}} = \text{test}(\text{Model}_{\text{SM\_SVM}}, \text{xt})$$

$$\text{y\_pre}_{\text{RUS\_LR}}, \text{y\_prob}_{\text{RUS\_LR}} = \text{test}(\text{Model}_{\text{RUS\_LR}}, \text{xt})$$
$$\text{y\_pre}_{\text{RUS\_KNN}}, \text{y\_prob}_{\text{RUS\_KNN}} = \text{test}(\text{Model}_{\text{RUS\_KNN}}, \text{xt})$$
$$\text{y\_pre}_{\text{RUS\_DT}}, \text{y\_prob}_{\text{RUS\_DT}} = \text{test}(\text{Model}_{\text{RUS\_DT}}, \text{xt})$$
$$\text{y\_pre}_{\text{RUS\_SVM}}, \text{y\_prob}_{\text{RUS\_SVM}} = \text{test}(\text{Model}_{\text{RUS\_SVM}}, \text{xt})$$

$$\text{y\_pre}_{\text{ADA\_LR}}, \text{y\_prob}_{\text{ADA\_LR}} = \text{test}(\text{Model}_{\text{ADA\_LR}}, \text{xt})$$
$$\text{y\_pre}_{\text{ADA\_KNN}}, \text{y\_prob}_{\text{ADA\_KNN}} = \text{test}(\text{Model}_{\text{ADA\_KNN}}, \text{xt})$$
$$\text{y\_pre}_{\text{ADA\_DT}}, \text{y\_prob}_{\text{ADA\_DT}} = \text{test}(\text{Model}_{\text{ADA\_DT}}, \text{xt})$$
$$\text{y\_pre}_{\text{ADA\_SVM}}, \text{y\_prob}_{\text{ADA\_SVM}} = \text{test}(\text{Model}_{\text{ADA\_SVM}}, \text{xt})$$
$$\text{y\_pre}_{\text{BSM\_LR}}, \text{y\_prob}_{\text{BSM\_LR}} = \text{test}(\text{Model}_{\text{BSM\_LR}}, \text{xt})$$
$$\text{y\_pre}_{\text{BSM\_KNN}}, \text{y\_prob}_{\text{BSM\_KNN}} = \text{test}(\text{Model}_{\text{BSM\_KNN}}, \text{xt})$$
$$\text{y\_pre}_{\text{BSM\_DT}}, \text{y\_prob}_{\text{BSM\_DT}} = \text{test}(\text{Model}_{\text{BSM\_DT}}, \text{xt})$$
$$\text{y\_pre}_{\text{BSM\_SVM}}, \text{y\_prob}_{\text{BSM\_SVM}} = \text{test}(\text{Model}_{\text{BSM\_SVM}}, \text{xt})$$

3) หลังจากนั้นนำตัวแบบที่มีเทคนิคสุ่มตัวอย่างซ้ำที่ได้ไปวัดประสิทธิภาพด้วย Test set โดยการคำนวณตัวชี้วัด ACC, AUC, F1, PRE, KS และ BS

$$\text{Performance}_{\text{SM\_LR}} = \text{Performance}(\text{y\_pre}_{\text{SM\_LR}}, \text{y\_prob}_{\text{SM\_LR}}, \text{yt})$$
$$\text{Performance}_{\text{SM\_KNN}} = \text{Performance}(\text{y\_pre}_{\text{SM\_KNN}}, \text{y\_prob}_{\text{SM\_KNN}}, \text{yt})$$
$$\text{Performance}_{\text{SM\_DT}} = \text{Performance}(\text{y\_pre}_{\text{SM\_DT}}, \text{y\_prob}_{\text{SM\_DT}}, \text{yt})$$
$$\text{Performance}_{\text{SM\_SVM}} = \text{Performance}(\text{y\_pre}_{\text{SM\_SVM}}, \text{y\_prob}_{\text{SM\_SVM}}, \text{yt})$$

$$\text{Performance}_{\text{RUS\_LR}} = \text{Performance}(\text{y\_pre}_{\text{RUS\_LR}}, \text{y\_prob}_{\text{RUS\_LR}}, \text{yt})$$
$$\text{Performance}_{\text{RUS\_KNN}} = \text{Performance}(\text{y\_pre}_{\text{RUS\_KNN}}, \text{y\_prob}_{\text{RUS\_KNN}}, \text{yt})$$
$$\text{Performance}_{\text{RUS\_DT}} = \text{Performance}(\text{y\_pre}_{\text{RUS\_DT}}, \text{y\_prob}_{\text{RUS\_DT}}, \text{yt})$$
$$\text{Performance}_{\text{RUS\_SVM}} = \text{Performance}(\text{y\_pre}_{\text{RUS\_SVM}}, \text{y\_prob}_{\text{RUS\_SVM}}, \text{yt})$$

$$\text{Performance}_{\text{ADA\_LR}} = \text{Performance}(\text{y\_pre}_{\text{ADA\_LR}}, \text{y\_prob}_{\text{ADA\_LR}}, \text{yt})$$
$$\text{Performance}_{\text{ADA\_KNN}} = \text{Performance}(\text{y\_pre}_{\text{ADA\_KNN}}, \text{y\_prob}_{\text{ADA\_KNN}}, \text{yt})$$
$$\text{Performance}_{\text{ADA\_DT}} = \text{Performance}(\text{y\_pre}_{\text{ADA\_DT}}, \text{y\_prob}_{\text{ADA\_DT}}, \text{yt})$$
$$\text{Performance}_{\text{ADA\_SVM}} = \text{Performance}(\text{y\_pre}_{\text{ADA\_SVM}}, \text{y\_prob}_{\text{ADA\_SVM}}, \text{yt})$$

$\text{Performance}_{\text{BSM\_LR}} = \text{Performance}(\text{y\_pre}_{\text{BSM\_LR}}, \text{y\_prob}_{\text{BSM\_LR}}, \text{yt})$
$\text{Performance}_{\text{BSM\_KNN}} = \text{Performance}(\text{y\_pre}_{\text{BSM\_KNN}}, \text{y\_prob}_{\text{BSM\_KNN}}, \text{yt})$
$\text{Performance}_{\text{BSM\_DT}} = \text{Performance}(\text{y\_pre}_{\text{BSM\_DT}}, \text{y\_prob}_{\text{BSM\_DT}}, \text{yt})$
$\text{Performance}_{\text{BSM\_SVM}} = \text{Performance}(\text{y\_pre}_{\text{BSM\_SVM}}, \text{y\_prob}_{\text{BSM\_SVM}}, \text{yt})$

4) หาอันดับตามแต่ละตัวชี้วัดในแต่ละตัวแบบของเทคนิคการสุ่มตัวอย่างซ้ำ (SM, RUS, ADA และ BSM) โดยเรียงจากมากไปน้อย จากนั้นหาอันดับเฉลี่ย (mean rank) ในแต่ละเทคนิคการสุ่มตัวอย่างซ้ำ

$$\text{Rank}_{\text{LR}} = \text{Rank}(\text{Performance}_{\text{SM\_LR}}, \text{Performance}_{\text{RUS\_LR}}, \text{Performance}_{\text{ADA\_LR}}, \text{Performance}_{\text{BSM\_LR}})$$
$$\text{Rank}_{\text{KNN}} = \text{Rank}(\text{Performance}_{\text{SM\_KNN}}, \text{Performance}_{\text{RUS\_KNN}}, \text{Performance}_{\text{ADA\_KNN}}, \text{Performance}_{\text{BSM\_KNN}})$$
$$\text{Rank}_{\text{DT}} = \text{Rank}(\text{Performance}_{\text{SM\_DT}}, \text{Performance}_{\text{RUS\_DT}}, \text{Performance}_{\text{ADA\_DT}}, \text{Performance}_{\text{BSM\_DT}})$$
$$\text{Rank}_{\text{SVM}} = \text{Rank}(\text{Performance}_{\text{SM\_SVM}}, \text{Performance}_{\text{RUS\_SVM}}, \text{Performance}_{\text{ADA\_SVM}}, \text{Performance}_{\text{BSM\_SVM}})$$

$$\text{Mean\_Rank\_ACC}_{\text{SM}}, \text{Mean\_Rank\_ACC}_{\text{RUS}}, \text{Mean\_Rank\_ACC}_{\text{ADA}}, \text{Mean\_Rank\_ACC}_{\text{BSM}} = \text{Mean}_{\text{ACC}}(\text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}})$$

$$\text{Mean\_Rank\_AUC}_{\text{SM}}, \text{Mean\_Rank\_AUC}_{\text{RUS}}, \text{Mean\_Rank\_AUC}_{\text{ADA}}, \text{Mean\_Rank\_AUC}_{\text{BSM}} = \text{Mean}_{\text{AUC}}(\text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}})$$

$$\text{Mean\_Rank\_F1}_{\text{SM}}, \text{Mean\_Rank\_F1}_{\text{RUS}}, \text{Mean\_Rank\_F1}_{\text{ADA}}, \text{Mean\_Rank\_F1}_{\text{BSM}} = \text{Mean}_{\text{F1}}(\text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}})$$

$$\text{Mean\_Rank\_PRE}_{\text{SM}}, \text{Mean\_Rank\_PRE}_{\text{RUS}}, \text{Mean\_Rank\_PRE}_{\text{ADA}}, \text{Mean\_Rank\_PRE}_{\text{BSM}} = \text{Mean}_{\text{PRE}}(\text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}})$$

$$\text{Mean\_Rank\_BS}_{\text{SM}}, \text{Mean\_Rank\_BS}_{\text{RUS}}, \text{Mean\_Rank\_BS}_{\text{ADA}}, \text{Mean\_Rank\_BS}_{\text{BSM}} = \text{Mean}_{\text{BS}}(\text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}})$$

$$\text{Mean\_Rank\_KS}_{\text{SM}}, \text{Mean\_Rank\_KS}_{\text{LM}}, \text{Mean\_Rank\_KS}_{\text{ADA}}, \text{Mean\_Rank\_KS}_{\text{BSM}} = \text{Mean}_{\text{KS}}(\text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}})$$

5) คำนวณหาอันดับเฉลี่ยเพื่อหาเทคนิคการสุ่มตัวอย่างซ้ำที่มีประสิทธิภาพมากที่สุด

ผลลัพธ์: $Min(Mean(Mean_{Rank_{SM}}), Mean(Mean_{Rank_{RUS}}), Mean(Mean\_Rank_{ADA}), Mean(Mean\_Rank_{BSM}))$

การศึกษาที่ 4 : เปรียบเทียบตัวแบบการเรียนรู้แบบรวมกลุ่ม (Bagging Heterogeneous Ensemble), Random Forest (RF) และตัวแบบการจำแนกประเภทพื้นฐาน 4 ตัวแบบ ได้แก่ Logistic Regression (LR), K-Nearest Neighbors (KNN), Decision Tree (DT), Support Vector Machine (SVM) โดยมีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำที่มีประสิทธิภาพเฉลี่ยดีที่สุดจากการศึกษาที่ 2 และ 3 ในแต่ละค่าอัตราส่วนความไม่สมดุล (IR)

<u>ขั้นตอนการทำงาน</u>

1) ทำการแปลงข้อมูลชุดเรียนรู้ด้วยเทคนิคการลดมิติข้อมูลที่มีประสิทธิภาพมากที่สุดจากการศึกษาที่ 2

$$\mathrm{x_{trans}} = \mathrm{Transform(x, y)}, \mathrm{xt_{trans}} = \mathrm{Transform(xt)}$$

2) นำข้อมูลจากข้อที่ 1 มาทำการสุ่มเพิ่มข้อมูลด้วยเทคนิคการสุ่มตัวอย่างซ้ำที่มีประสิทธิภาพมากที่สุดจากการศึกษาที่ 3

$$\mathrm{x_{res\_trans}} = \mathrm{Resampling(x_{trans}, y)}$$

3) นำชุดข้อมูลจากข้อที่ 2 มาสร้างตัวแบบเพื่อเปรียบเทียบประสิทธิภาพ ประกอบด้วย (1) Bagging Hetero (2) LR (3) KNN (4) DT (5) SVM (6) RF

$$\mathrm{Model_{Bagging\ Hetero}} = \mathrm{train(Bagging\ Hetero, x_{res\_trans}, y)}$$
$$\mathrm{Model_{LR}} = \mathrm{train(LR, x_{res\_trans}, y)}$$
$$\mathrm{Model_{KNN}} = \mathrm{train(KNN, x_{res\_trans}, y)}$$
$$\mathrm{Model_{DT}} = \mathrm{train(DT, x_{res\_trans}, y)}$$
$$\mathrm{Model_{SVM}} = \mathrm{train(SVM, x_{res\_trans}, y)}$$
$$\mathrm{Model_{RF}} = \mathrm{train(RF, x_{res\_trans}, y)}$$

$$\mathrm{y\_pre_{Bagging\ Hetero}},\ \mathrm{y\_prob_{Bagging\ Hetero}} = \mathrm{test(Model_{Bagging\ Hetero}, xt_{trans})}$$
$$\mathrm{y\_pre_{LR}}, \mathrm{y\_prob_{LR}} = \mathrm{test(Model_{LR}, xt_{trans})}$$
$$\mathrm{y\_pre_{KNN}}, \mathrm{y\_prob_{KNN}} = \mathrm{test(Model_{KNN}, xt_{trans})}$$
$$\mathrm{y\_pre_{DT}}, \mathrm{y\_prob_{DT}} = \mathrm{test(Model_{DT}, xt_{trans})}$$
$$\mathrm{y\_pre_{SVM}}, \mathrm{y\_prob_{SVM}} = \mathrm{test(Model_{SVM}, xt_{trans})}$$
$$\mathrm{y\_pre_{RF}}, \mathrm{y\_prob_{RF}} = \mathrm{test(Model_{RF}, xt_{trans})}$$

4) หลังจากนั้นนำตัวแบบที่ได้ไปวัดประสิทธิภาพด้วย Test set โดยการคำนวณตัวชี้วัด ACC, AUC, F1, PRE, KS และ BS

$$\text{Performance}_{\text{Bagging Hetero}} = \text{Performance}(\text{y\_pre}_{\text{Bagging Hetero}}, \text{y\_prob}_{\text{Bagging Hetero}}, \text{yt})$$
$$\text{Performance}_{\text{LR}} = \text{Performance}(\text{y\_pre}_{\text{LR}}, \text{y\_prob}_{\text{LR}}, \text{yt})$$
$$\text{Performance}_{\text{KNN}} = \text{Performance}(\text{y\_pre}_{\text{KNN}}, \text{y\_prob}_{\text{KNN}}, \text{yt})$$
$$\text{Performance}_{\text{DT}} = \text{Performance}(\text{y\_pre}_{\text{DT}}, \text{y\_prob}_{\text{DT}}, \text{yt})$$
$$\text{Performance}_{\text{SVM}} = \text{Performance}(\text{y\_pre}_{\text{SVM}}, \text{y\_prob}_{\text{SVM}}, \text{yt})$$
$$\text{Performance}_{\text{RF}} = \text{Performance}(\text{y\_pre}_{\text{RF}}, \text{y\_prob}_{\text{RF}}, \text{yt})$$

5) หาอันดับตามแต่ละตัวชี้วัดโดยเรียงจากมากไปน้อย จากนั้นหาอันดับเฉลี่ย (mean rank) ในแต่ละตัวแบบ

$$\text{Rank}_{\text{Bagging Hetero}} = \text{Rank}(\text{Performance}_{\text{Bagging Hetero}})$$
$$\text{Rank}_{\text{LR}} = \text{Rank}(\text{Performance}_{\text{LR}})$$
$$\text{Rank}_{\text{KNN}} = \text{Rank}(\text{Performance}_{\text{KNN}})$$
$$\text{Rank}_{\text{DT}} = \text{Rank}(\text{Performance}_{\text{DT}})$$
$$\text{Rank}_{\text{SVM}} = \text{Rank}(\text{Performance}_{\text{SVM}})$$
$$\text{Rank}_{\text{RF}} = \text{Rank}(\text{Performance}_{\text{RF}})$$

$$\text{Mean\_Rank\_ACC} = \text{Mean}_{\text{ACC}}(\text{Rank}_{\text{Bagging Hetero}}, \text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}}, \text{Rank}_{\text{RF}})$$
$$\text{Mean\_Rank\_AUC} = \text{Mean}_{\text{AUC}}(\text{Rank}_{\text{Bagging Hetero}}, \text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}}, \text{Rank}_{\text{RF}})$$
$$\text{Mean\_Rank\_F1} = \text{Mean}_{\text{F1}}(\text{Rank}_{\text{Bagging Hetero}}, \text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}}, \text{Rank}_{\text{RF}})$$
$$\text{Mean\_Rank\_PRE} = \text{Mean}_{\text{PRE}}(\text{Rank}_{\text{Bagging Hetero}}, \text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}}, \text{Rank}_{\text{RF}})$$
$$\text{Mean\_Rank\_BS} = \text{Mean}_{\text{BS}}(\text{Rank}_{\text{Bagging Hetero}}, \text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}}, \text{Rank}_{\text{RF}})$$
$$\text{Mean\_Rank\_KS} = \text{Mean}_{\text{KS}}(\text{Rank}_{\text{Bagging Hetero}}, \text{Rank}_{\text{LR}}, \text{Rank}_{\text{KNN}}, \text{Rank}_{\text{DT}}, \text{Rank}_{\text{SVM}}, \text{Rank}_{\text{RF}})$$

6) คำนวณหาอันดับเฉลี่ยเพื่อหาตัวแบบที่มีประสิทธิภาพสูงที่สุด

ผลลัพธ์: $\text{Min}(\text{Mean}(\text{Mean\_Rank}))$

หมายเหตุ: ใช้ภาษาคอมพิวเตอร์ไพธอน (Python) ในการเขียนโปรแกรม Google Collaboratory เพื่อจัดการข้อมูลรวมถึงสร้างตัวแบบ โดย code ทั้งหมดอยู่ในส่วนของภาคผนวก ด้านบนเป็นเพียง Pseudocode ที่อธิบายเพื่อให้เห็นถึงขั้นตอนการดำเนินการวิจัย

บทที่ 4
ผลวิจัย

การวิจัยนี้ทำการศึกษาโดยจำลองข้อมูลเครดิตเยอรมันในอัตราส่วนความไม่สมดุลที่แตกต่างกันเพื่อเปรียบเทียบประสิทธิภาพของการเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนาน (Bagging Heterogeneous Ensemble), ตัวแบบการสุ่มป่าไม้ (RF) และตัวแบบพื้นฐาน 4 ตัวแบบ คือ Logistic Regression (LR), Decision Tree (DT), K-Nearest Neighbors (KNN) และ Support Vector Machine (SVM) โดยวัดประสิทธิภาพตัวแบบด้วยตัวชี้วัด Accuracy (ACC), The area under the curve (AUC), The F1-score (F1), Precision (PRE), Kolmogorov-Smirnov (KS) และ Brier Score (BS) และทดสอบทางสถิติเพื่อแสดงว่าประสิทธิภาพของตัวแบบแตกต่างกันที่ระดับนัยสำคัญ 0.05

งานวิจัยนี้ได้จำลองอัตราส่วนความไม่สมดุล 3 ค่า คือ 2.3, 10 และ 14 ของข้อมูลเครดิตเยอรมันและแบ่งการศึกษาเป็น 4 การศึกษา คือ (1) เปรียบเทียบประสิทธิภาพของตัวแบบการเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนาน, ตัวแบบการสุ่มป่าไม้ และตัวแบบการจำแนกประเภทพื้นฐาน 4 ตัวแบบที่ไม่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำในแต่ละอัตราส่วนความไม่สมดุล (2) เปรียบเทียบวิธีการลดมิติข้อมูล 2 วิธี ได้แก่ Singular Value Decomposition (SVD), Linear Discriminant Analysis (LDA) ด้วยตัวแบบการจำแนกประเภทพื้นฐาน 4 ตัวแบบในแต่ละค่าอัตราส่วนความไม่สมดุล (3) เปรียบเทียบวิธีการสุ่มตัวอย่างซ้ำ 4 วิธี ได้แก่ Systematic Minority Over-Sampling Technique (SM), Random Under-Sampling (RUS), Adaptive Synthetic (ADA) , Borderline SMOTE (BSM) ด้วยตัวแบบการจำแนกประเภทพื้นฐาน 4 ตัวแบบ ในแต่ละค่าอัตราส่วนความไม่สมดุล และ (4) เปรียบเทียบตัวแบบการเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนาน, ตัวแบบการสุ่มป่าไม้ และตัวแบบการจำแนกประเภทพื้นฐาน 4 ตัวแบบ ได้แก่ Logistic Regression (LR), K-Nearest Neighbors (KNN), Decision Tree (DT), Support Vector Machine (SVM) ที่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำที่มีประสิทธิภาพเฉลี่ยดีที่สุดจากการทดลองที่ 2 และ 3 ในแต่ละค่าอัตราส่วนความไม่สมดุล

ผลการวิจัยจัดทำในรูปแบบตารางและแผนภาพ โดยแบ่งตามการศึกษาออกเป็น 4 การศึกษา และในแต่การศึกษาจะประกอบด้วยผลลัพธ์ของอัตราส่วนความไม่สมดุลที่ 2.3, 10 และ 14

ผลการเปรียบเทียบประสิทธิภาพของการเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนาน , ตัวแบบการสุ่มป่าไม้ และตัวแบบการจำแนกประเภทพื้นฐาน 4 ตัวแบบ ที่ไม่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำ

ในตารางที่ 4.1 แสดงผลการเปรียบเทียบประสิทธิภาพตัวแบบในแต่ละอัตราส่วนความไม่สมดุล โดยวัดประสิทธิภาพตัวแบบด้วยตัวชี้วัด Accuracy (ACC), The area under the curve (AUC), The F1-score (F1), Precision (PRE), Kolmogorov-Smirnov (KS) และ Brier Score (BS) ได้ผลดังนี้

ตารางที่ 4.1 ประสิทธิภาพของตัวแบบจำแนกประเภทที่ไม่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำ

IR	Performance	Bagging Hetero	RF	LR	KNN	DT	SVM
2.3	ACC	0.770	0.770	0.740	0.755	0.705	0.730
	AUC	0.809	0.770	0.781	0.762	0.651	0.786
	F1	0.854	0.853	0.824	0.846	0.826	0.790
	PRE	0.771	0.774	0.787	0.758	0.705	0.871
	KS	0.498	0.427	0.460	0.386	0.308	0.471
	BS	0.161	0.167	0.171	0.170	0.191	0.165
10	ACC	0.948	0.941	0.941	0.948	0.948	0.941
	AUC	0.738	0.575	0.732	0.660	0.689	0.716
	F1	0.973	0.970	0.969	0.973	0.973	0.970
	PRE	0.948	0.947	0.953	0.948	0.948	0.947
	KS	0.525	0.222	0.537	0.351	0.356	0.446
	BS	0.048	0.055	0.051	0.052	0.054	0.049
14	ACC	0.946	0.946	0.933	0.946	0.940	0.946
	AUC	0.801	0.767	0.785	0.756	0.568	0.778
	F1	0.972	0.972	0.964	0.972	0.969	0.972
	PRE	0.946	0.946	0.964	0.946	0.9463	0.946
	KS	0.619	0.473	0.536	0.385	0.346	0.515
	BS	0.045	0.049	0.055	0.047	0.066	0.046

จากการเปรียบเทียบประสิทธิภาพตัวแบบที่ไม่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำ ผู้วิจัยทำการให้อันดับในแต่ละตัวชี้วัดที่แสดงถึงประสิทธิภาพของตัวแบบ โดยอันดับที่น้อยลงหมายถึงตัวแบบมีประสิทธิภาพมากขึ้นในแต่ละตัวชี้วัด จากนั้นหาอันดับเฉลี่ยของแต่ละตัวแบบที่ไม่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำ ได้ผลลัพธ์ดังตารางที่ 4.2

ตารางที่ 4.2 อันดับในแต่ละตัวชี้วัดและอันดับเฉลี่ยของแต่ละตัวแบบจำแนกประเภทที่ไม่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำ (ตัวหนาแสดงตัวแบบที่มีประสิทธิภาพสูงสุดในแต่ละค่าอัตราส่วนความไม่สมดุล)

IR	Performance	Bagging Hetero	RF	LR	KNN	DT	SVM
2.3	ACC	1.5	1.5	4	3	6	5
	AUC	1	4	3	5	6	2
	F1	1	2	5	3	4	6
	PRE	4	3	2	5	6	1
	KS	1	4	3	5	6	2
	BS	1	3	5	4	6	2
	Mean Rank	**1.583**	2.916	3.666	4.166	5.666	3.000
10	ACC	2	5	5	2	2	5
	AUC	1	6	2	5	4	3
	F1	2	4.5	6	2	2	4.5
	PRE	3	5.5	1	3	3	5.5
	KS	2	6	1	5	4	3
	BS	1	6	3	4	5	2
	Mean Rank	**1.833**	5.500	3.000	3.500	3.333	3.833
14	ACC	2.5	2.5	6	2.5	5	2.5
	AUC	1	4	2	5	6	3
	F1	2.5	2.5	6	2.5	5	2.5
	PRE	3.5	3.5	1	3.5	6	3.5
	KS	1	4	2	5	6	3
	BS	1	4	5	3	6	2
	Mean Rank	**1.916**	3.416	3.666	3.583	5.666	2.750

จากตารางที่ 4.2 แสดงอันดับของตัวแบบในแต่ละตัวชี้วัดและอันดับเฉลี่ยของอัตราส่วนความไม่สมดุลที่ 2.3, 10 และ 14 พบว่า การเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนาน (Bagging Hetero) มีประสิทธิภาพดีที่สุดในทุกตัวแบบที่มีอัตราส่วนความไม่สมดุลที่ 2.3, 10 และ 14 โดยมีอันดับเฉลี่ย 1.583, 1.833 และ 1.916 ตามลำดับ

ในอัตราส่วนความไม่สมดุลที่ 2.3 (IR = 2.3) ตัวแบบที่มีประสิทธิภาพรองลงมาคือตัวแบบการสุ่มป่าไม้ (RF), ตัวแบบซัพพอร์ตเวกเตอร์แมชชีน (SVM), ตัวแบบการถดถอยโลจิสติก (LR), ตัวแบบเพื่อนบ้านใกล้สุด (KNN) และตัวแบบต้นไม้ตัดสินใจ (DT) โดยมีอันดับเฉลี่ย 2.916, 3.000, 3.666, 4.166 และ 5.666 ตามลำดับ

อัตราส่วนความไม่สมดุลที่ 10 (IR = 10) ตัวแบบที่มีประสิทธิภาพรองลงมาคือตัวแบบการถดถอยโลจิสติก (LR), ตัวแบบต้นไม้การตัดสินใจ (DT), ตัวแบบเพื่อนบ้านใกล้สุด (KNN), ตัวแบบซัพ

พอร์ตเวคเตอร์แมชชีน (SVM) และตัวแบบการสุ่มป่าไม้ (RF) โดยมีอันดับเฉลี่ย 3.000, 3.333, 3.500, 3.833 และ 5.500 ตามลำดับ

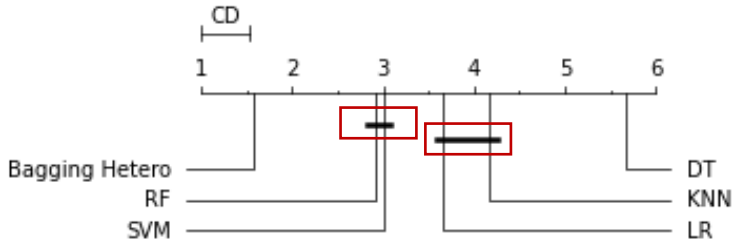
อัตราส่วนความไม่สมดุลที่ 14 (IR = 14) ตัวแบบที่มีประสิทธิภาพรองลงมาคือ ตัวแบบซัพพอร์ตเวกเตอร์แมชชีน (SVM), ตัวแบบการสุ่มป่าไม้ (RF), ตัวแบบเพื่อนบ้านใกล้สุด (KNN), ตัวแบบการถดถอยโลจิสติก (LR) และตัวแบบต้นไม้การตัดสินใจ (DT) โดยมีอันดับเฉลี่ย 2.750, 3.416, 3.583, 3.666 และ 5.666 ตามลำดับ

จากนั้นใช้การทดสอบ Friedman เพื่อแสดงว่าประสิทธิภาพตัวแบบของแต่ละอัตราส่วนความไม่สมดุลแตกต่างกันที่ระดับนัยสำคัญ 0.05 ได้ผลลัพธ์ดังตารางที่ 4.3

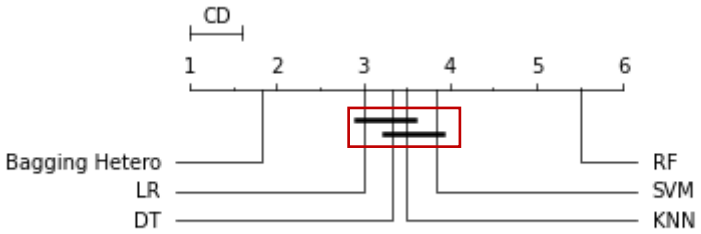
ตารางที่ 4.3 สถิติทดสอบ Friedman ในแต่ละค่าอัตราส่วนความไม่สมดุล

Imbalance Ratio (IR)	Friedman Statistic	p-value
IR 2.3	16.244	0.006
IR 10	13.437	0.019
IR 14	15.611	0.008

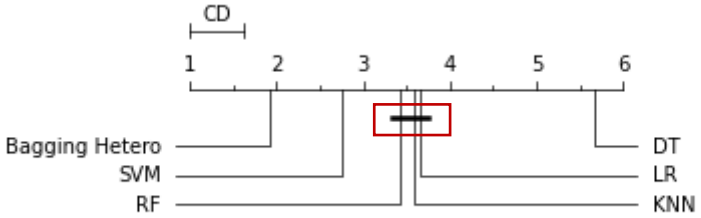
จากตารางที่ 4.3 พบว่าในอัตราส่วนความไม่สมดุลที่ 2.3, 10 และ 14 การเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนาน (Bagging Hetero), ตัวแบบการสุ่มป่าไม้ (RF) และตัวแบบการจำแนกประเภทพื้นฐาน 4 ตัวแบบ ได้แก่ Logistic Regression (LR), K-Nearest Neighbors (KNN), Decision Tree (DT), Support Vector Machine (SVM) ที่ไม่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำมีประสิทธิภาพแตกต่างกันที่ระดับนัยสำคัญ 0.05 จึงนำไปทดสอบ Neymanyi post hoc เพื่อหาว่าตัวแบบใดมีความแตกต่างกัน โดยตัวแบบจะแตกต่างกันเมื่ออันดับเฉลี่ยต่างกันอย่างน้อย Critical Difference (CD) ถ้าตัวแบบที่ไม่แตกต่างกันจะมีเส้นสีดำขีดระหว่างอันดับเฉลี่ยของแต่ละตัวแบบ ดังกรอบสีแดงด้านล่าง และจากการทดสอบ Nemenyi post hoc ได้ค่า critical difference (CD) ในแต่ละอัตราส่วนความไม่สมดุลดังนี้ IR 2.3 (CD = 0.533) ดังรูปที่ 4.1, IR 10 (CD = 0.607) ดังรูปที่ 4.2 และ IR 14 (CD= 0.615) ดังรูปที่ 4.3



รูปที่ 4.9 แผนภาพแสดงอันดับเฉลี่ยของตัวแบบที่ไม่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำของอัตราส่วนความไม่สมดุลที่ 2.3 โดยใช้ Neymanyi post hoc



รูปที่ 4.10 แผนภาพแสดงอันดับเฉลี่ยของตัวแบบที่ไม่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำของอัตราส่วนความไม่สมดุลที่ 10 โดยใช้ Neymanyi post hoc



รูปที่ 4.11 แผนภาพแสดงอันดับเฉลี่ยของตัวแบบที่ไม่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำของอัตราส่วนความไม่สมดุลที่ 14 โดยใช้ Neymanyi post hoc

จากแผนภาพแสดงอันดับเฉลี่ยของตัวแบบที่ไม่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำของอัตราส่วนความไม่สมดุลที่ 2.3, 10 และ 14 โดยใช้ Neymanyi post hoc ดังรูปที่ 4.1, 4.2 และ 4.3 ตามลำดับ พบว่าการเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนาน (Bagging Hetero) มีอันดับเฉลี่ยดีที่สุดในทุกๆ อัตราส่วนความไม่สมดุล

ผลการเปรียบเทียบวิธีการลดมิติข้อมูล ได้แก่ Singular Value Decomposition (SVD) และ Linear Discriminant Analysis (LDA) ด้วยตัวแบบจำแนกประเภทพื้นฐาน 4 ตัวแบบ ในแต่ละค่าอัตราส่วนความไม่สมดุล

ในตารางที่ 4.4 แสดงผลการเปรียบเทียบประสิทธิภาพของวิธีการลดมิติข้อมูลในแต่ละอัตราส่วนความไม่สมดุล โดยวัดประสิทธิภาพตัวแบบด้วยตัวชี้วัด Accuracy (ACC), The area under the curve (AUC), The F1-score (F1), Precision (PRE), Kolmogorov-Smirnov (KS) และ Brier Score (BS) ได้ผลวิจัยดังนี้

ตารางที่ 4.4 ประสิทธิภาพของวิธีการลดมิติข้อมูลในแต่ละตัวแบบจำแนกประเภทพื้นฐานและตัวชี้วัดที่แตกต่างกัน

Performance	model	IR 2.3		IR 10		IR 14	
		SVD	LDA	SVD	LDA	SVD	LDA
ACC	LR	0.730	0.750	0.948	0.935	0.953	0.927
	KNN	0.740	0.740	0.948	0.935	0.947	0.920
	DT	0.705	0.740	0.935	0.942	0.913	0.880
	SVM	0.705	0.740	0.773	0.909	0.627	0.847
AUC	LR	0.790	0.782	0.687	0.718	0.724	0.793
	KNN	0.710	0.756	0.640	0.694	0.740	0.813
	DT	0.555	0.727	0.525	0.553	0.557	0.799
	SVM	0.790	0.782	0.588	0.718	0.754	0.793
F1	LR	0.818	0.830	0.973	0.966	0.976	0.961
	KNN	0.840	0.819	0.973	0.966	0.973	0.958
	DT	0.827	0.826	0.966	0.970	0.955	0.934
	SVM	0.772	0.826	0.868	0.952	0.761	0.914
PRE	LR	0.781	0.797	0.948	0.953	0.953	0.958
	KNN	0.741	0.803	0.948	0.953	0.947	0.958
	DT	0.705	0.783	0.947	0.954	0.945	0.970
	SVM	0.847	0.783	0.966	0.958	0.967	0.969
KS	LR	0.470	0.469	0.526	0.531	0.384	0.553
	KNN	0.330	0.413	0.301	0.349	0.444	0.518
	DT	0.136	0.405	0.075	0.295	0.158	0.495
	SVM	0.423	0.469	0.308	0.531	0.435	0.553
BS	LR	0.170	0.170	0.054	0.055	0.047	0.058
	KNN	0.181	0.180	0.052	0.055	0.048	0.055

	DT	0.213	0.182	0.064	0.056	0.082	0.058
	SVM	0.164	0.174	0.053	0.055	0.047	0.061

จากการเปรียบเทียบวิธีการลดมิติข้อมูล ผู้วิจัยทำการให้อันดับในแต่ละตัวแบบพื้นฐานของแต่ละตัวชี้วัดที่แสดงถึงประสิทธิภาพวิธีการลดมิติข้อมูลในอัตราส่วนความไม่สมดุลที่แตกต่างกัน โดยอันดับที่น้อยลงหมายถึงวิธีการลดมิติข้อมูลมีประสิทธิภาพมากขึ้นแต่ละตัวแบบพื้นฐาน ได้ผลลัพธ์ดังตารางที่ 4.5

ตารางที่ 4.5 อันดับของวิธีการลดมิติข้อมูลในแต่ละตัวแบบจำแนกประเภทพื้นฐานและตัวชี้วัดที่แตกต่างกัน

Performance	model	IR 2.3		IR 10		IR 14	
		SVD	LDA	SVD	LDA	SVD	LDA
ACC	LR	2.0	1.0	1.0	2.0	1.0	2.0
	KNN	1.5	1.5	1.0	2.0	1.0	2.0
	DT	2.0	1.0	2.0	1.0	1.0	2.0
	SVM	2.0	1.0	2.0	1.0	2.0	1.0
AUC	LR	1.0	2.0	2.0	1.0	2.0	1.0
	KNN	2.0	1.0	2.0	1.0	2.0	1.0
	DT	2.0	1.0	2.0	1.0	2.0	1.0
	SVM	1.0	2.0	2.0	1.0	2.0	1.0
F1	LR	2.0	1.0	1.0	2.0	1.0	2.0
	KNN	1.0	2.0	1.0	2.0	1.0	2.0
	DT	1.0	2.0	2.0	1.0	1.0	2.0
	SVM	2.0	1.0	2.0	1.0	2.0	1.0
PRE	LR	2.0	1.0	2.0	1.0	2.0	1.0
	KNN	2.0	1.0	2.0	1.0	2.0	1.0
	DT	2.0	1.0	2.0	1.0	2.0	1.0
	SVM	1.0	2.0	1.0	2.0	2.0	1.0
KS	LR	1.0	2.0	2.0	1.0	2.0	1.0
	KNN	2.0	1.0	2.0	1.0	2.0	1.0
	DT	2.0	1.0	2.0	1.0	2.0	1.0
	SVM	2.0	1.0	2.0	1.0	2.0	1.0
BS	LR	1.5	1.5	1.0	2.0	1.0	2.0
	KNN	2.0	1.0	1.0	2.0	1.0	2.0
	DT	2.0	1.0	2.0	1.0	2.0	1.0

	SVM	1.0	2.0	1.0	2.0	1.0	2.0

จากตารางที่ 4.5 ผู้วิจัยได้หาอันดับเฉลี่ยของวิธีการลดมิติข้อมูลในแต่ละตัวชี้วัดของอัตราส่วนความไม่สมดุลที่แตกต่างกัน เพื่อหาวิธีการลดมิติข้อมูลที่เหมาะสมในอัตราส่วนความไม่สมดุล 2.3, 10 และ 14 ได้ผลลัพธ์ดังตารางที่ 4.6

ตารางที่ 4.6 อันดับเฉลี่ยของวิธีการลดมิติข้อมูลในแต่ละตัวชี้วัดที่แตกต่างกัน (ตัวหนาแสดงวิธีการลดมิติข้อมูลที่มีประสิทธิภาพสูงสุดในแต่ละค่าอัตราส่วนความไม่สมดุล)

Mean Rank	IR 2.3		IR 10		IR 14	
	SVD	LDA	SVD	LDA	SVD	LDA
ACC	1.875	1.125	1.500	1.500	1.250	1.750
AUC	1.500	1.500	2.000	1.000	2.000	1.000
F1	1.500	1.500	1.500	1.500	1.250	1.750
PRE	1.750	1.250	1.750	1.250	2.000	1.000
KS	1.750	1.250	2.000	1.000	2.000	1.000
BS	1.625	1.375	1.250	1.750	1.250	1.750
Average	1.667	**1.333**	1.667	**1.333**	1.625	**1.375**

จากตาราง 4.6 พบว่า LDA มีประสิทธิภาพดีที่สุดในอัตราส่วนความไม่สมดุลที่ 2.3, 10 และ 14 โดยมีอันดับเฉลี่ย 1.333, 1.333 และ 1.375 ตามลำดับ และจากการทดสอบ Wilcoxon signed-rank พบว่าเทคนิค LDA และ SVD มีประสิทธิภาพแตกต่างกันในแต่ละอัตราส่วนความไม่สมดุลที่ระดับนัยสำคัญ 0.05 ดังตารางที่ 4.7

ตารางที่ 4.7 สถิติทดสอบ Wilcoxon signed-rank ในแต่ละอัตราส่วนความไม่สมดุล

Imbalance Ratio (IR)	Statistic	p-value
IR 2.3	54.0	0.018
IR 10	74.0	0.029
IR 14	73.0	0.026

ผลการเปรียบเทียบวิธีการสุ่มตัวอย่างซ้ำ ได้แก่ Systematic Minority Over-Sampling Technique (SM), Random Under-Sampling (RUS), Adaptive Synthetic (ADA), Borderline SMOTE (BSM) ด้วยตัวแบบจำแนกประเภทพื้นฐาน 4 ตัวแบบ ในแต่ละค่าอัตราส่วนความไม่สมดุล

ในตารางที่ 4.8 แสดงผลการเปรียบเทียบประสิทธิภาพของวิธีการสุ่มตัวอย่างซ้ำในแต่ละอัตราส่วนความไม่สมดุล โดยวัดประสิทธิภาพตัวแบบด้วยตัวชี้วัด Accuracy (ACC), The area under the curve (AUC), The F1-score (F1), Precision (PRE), Kolmogorov-Smirnov (KS) และ Brier Score (BS) ได้ผลวิจัยดังนี้

ตารางที่ 4.8 ประสิทธิภาพของวิธีการสุ่มตัวอย่างซ้ำในแต่ละตัวแบบจำแนกประเภทพื้นฐานและตัวชี้วัดที่แตกต่างกัน

Performance	model	IR 2.3				IR 10				IR 14			
		SM	RUS	ADA	BSM	SM	RUS	ADA	BSM	SM	RUS	ADA	BSM
ACC	LR	0.755	0.745	0.755	0.760	0.909	0.948	0.916	0.922	0.913	0.940	0.900	0.920
	KNN	0.680	0.625	0.665	0.675	0.721	0.643	0.695	0.753	0.713	0.520	0.653	0.720
	DT	0.705	0.665	0.680	0.690	0.708	0.948	0.649	0.864	0.867	0.947	0.840	0.853
	SVM	0.755	0.760	0.760	0.745	0.903	0.760	0.896	0.922	0.887	0.753	0.907	0.933
AUC	LR	0.798	0.782	0.795	0.794	0.667	0.738	0.693	0.675	0.784	0.775	0.764	0.793
	KNN	0.748	0.743	0.754	0.747	0.708	0.619	0.676	0.719	0.588	0.664	0.662	0.612
	DT	0.735	0.702	0.683	0.706	0.617	0.690	0.597	0.639	0.518	0.624	0.500	0.724
	SVM	0.790	0.796	0.793	0.797	0.670	0.711	0.661	0.658	0.754	0.763	0.756	0.746
F1	LR	0.826	0.827	0.829	0.833	0.952	0.973	0.956	0.959	0.954	0.969	0.946	0.957
	KNN	0.744	0.699	0.733	0.741	0.831	0.776	0.813	0.855	0.827	0.667	0.783	0.832
	DT	0.779	0.747	0.770	0.756	0.825	0.973	0.782	0.926	0.928	0.973	0.911	0.919
	SVM	0.828	0.840	0.832	0.816	0.948	0.859	0.945	0.959	0.939	0.854	0.950	0.965
PRE	LR	0.829	0.792	0.815	0.816	0.952	0.954	0.952	0.953	0.964	0.952	0.964	0.964
	KNN	0.853	0.806	0.836	0.845	0.972	0.960	0.971	0.966	0.963	0.973	0.959	0.963
	DT	0.825	0.798	0.781	0.850	0.955	0.948	0.951	0.950	0.949	0.947	0.961	0.969
	SVM	0.819	0.792	0.821	0.831	0.952	0.966	0.951	0.953	0.963	0.973	0.964	0.971
KS	LR	0.470	0.461	0.490	0.480	0.385	0.545	0.421	0.408	0.614	0.456	0.502	0.607
	KNN	0.417	0.364	0.426	0.410	0.408	0.430	0.442	0.397	0.254	0.257	0.338	0.282
	DT	0.383	0.351	0.308	0.393	0.327	0.356	0.277	0.416	0.239	0.342	0.262	0.373
	SVM	0.476	0.468	0.466	0.486	0.440	0.423	0.409	0.358	0.468	0.447	0.475	0.498

BS	LR	0.166	0.172	0.170	0.170	0.069	0.050	0.065	0.068	0.074	0.047	0.081	0.070
	KNN	0.211	0.203	0.210	0.215	0.195	0.239	0.205	0.164	0.213	0.271	0.232	0.221
	DT	0.194	0.264	0.232	0.205	0.218	0.054	0.219	0.115	0.139	0.056	0.142	0.139
	SVM	0.172	0.160	0.170	0.170	0.074	0.049	0.083	0.068	0.080	0.047	0.076	0.071

จากการเปรียบเทียบวิธีการสุ่มตัวอย่างซ้ำ ผู้วิจัยทำการให้อันดับในแต่ละตัวแบบพื้นฐานของแต่ละตัวชี้วัดที่แสดงถึงประสิทธิภาพวิธีการสุ่มตัวอย่างซ้ำของอัตราส่วนความไม่สมดุลที่แตกต่างกัน โดยอันดับที่น้อยลงหมายถึงวิธีการสุ่มตัวอย่างซ้ำมีประสิทธิภาพมากขึ้นแต่ละตัวแบบพื้นฐาน ได้ผลลัพธ์ดังตารางที่ 4.9

ตารางที่ 4.9 อันดับของวิธีการสุ่มตัวอย่างซ้ำในแต่ละตัวแบบจำแนกประเภทพื้นฐานและตัวชี้วัดที่แตกต่างกัน

Performance	model	IR 2.3				IR 10				IR 14			
		SM	RUS	ADA	BSM	SM	RUS	ADA	BSM	SVD	LDA	ADA	BSM
ACC	LR	2.5	4.0	2.5	1.0	4.0	1.0	3.0	2.0	3.0	1.0	4.0	2.0
	KNN	1.0	4.0	3.0	2.0	2.0	4.0	3.0	1.0	2.0	4.0	3.0	1.0
	DT	1.0	4.0	3.0	2.0	3.0	1.0	4.0	2.0	2.0	1.0	4.0	3.0
	SVM	3.0	1.5	1.5	4.0	2.0	4.0	3.0	1.0	3.0	4.0	2.0	1.0
AUC	LR	1.0	4.0	2.0	3.0	4.0	1.0	2.0	3.0	2.0	3.0	4.0	1.0
	KNN	2.0	4.0	1.0	3.0	2.0	4.0	3.0	1.0	4.0	1.0	2.0	3.0
	DT	1.0	3.0	4.0	2.0	3.0	1.0	4.0	2.0	3.0	2.0	4.0	1.0
	SVM	4.0	2.0	3.0	1.0	2.0	1.0	3.0	4.0	3.0	1.0	2.0	4.0
F1	LR	4.0	3.0	2.0	1.0	4.0	1.0	3.0	2.0	3.0	1.0	4.0	2.0
	KNN	1.0	4.0	3.0	2.0	2.0	4.0	3.0	1.0	2.0	4.0	3.0	1.0
	DT	1.0	4.0	2.0	3.0	3.0	1.0	4.0	2.0	2.0	1.0	4.0	3.0
	SVM	3.0	1.0	2.0	4.0	2.0	4.0	3.0	1.0	3.0	4.0	2.0	1.0
PRE	LR	1.0	4.0	3.0	2.0	3.5	1.0	3.5	2.0	2.0	4.0	2.0	2.0
	KNN	1.0	4.0	3.0	2.0	1.0	4.0	2.0	3.0	2.5	1.0	4.0	2.5
	DT	2.0	3.0	4.0	1.0	1.0	4.0	2.0	3.0	3.0	4.0	2.0	1.0
	SVM	3.0	4.0	2.0	1.0	3.0	1.0	4.0	2.0	4.0	1.0	3.0	2.0
KS	LR	3.0	4.0	1.0	2.0	4.0	1.0	2.0	3.0	1.0	4.0	3.0	2.0
	KNN	2.0	4.0	1.0	3.0	3.0	2.0	1.0	4.0	4.0	3.0	1.0	2.0
	DT	2.0	3.0	4.0	1.0	3.0	2.0	4.0	1.0	4.0	2.0	3.0	1.0
	SVM	2.0	3.0	4.0	1.0	1.0	2.0	3.0	4.0	3.0	4.0	2.0	1.0
BS	LR	1.0	4.0	2.5	2.5	4.0	1.0	2.0	3.0	3.0	1.0	4.0	2.0

	KNN	3.0	1.0	2.0	4.0	2.0	4.0	3.0	1.0	1.0	4.0	3.0	2.0
	DT	1.0	4.0	3.0	2.0	3.0	1.0	4.0	2.0	2.5	1.0	4.0	2.5
	SVM	4.0	1.0	2.5	2.5	3.0	1.0	4.0	2.0	4.0	1.0	3.0	2.0

จากตารางที่ 4.9 ผู้วิจัยได้หาอันดับเฉลี่ยของวิธีการสุ่มตัวอย่างซ้ำในแต่ละตัวชี้วัดของอัตราส่วนความไม่สมดุลที่แตกต่างกัน เพื่อหาวิธีการสุ่มตัวอย่างซ้ำที่เหมาะสมในอัตราส่วนความไม่สมดุล 2.3, 10 และ 14 ได้ผลลัพธ์ดังตารางที่ 4.10

ตารางที่ 4.10 อันดับเฉลี่ยของวิธีการสุ่มตัวอย่างซ้ำในแต่ละตัวชี้วัดที่แตกต่างกัน (ตัวหนาแสดงวิธีการสุ่มตัวอย่างซ้ำที่มีประสิทธิภาพสูงสุดในแต่ละค่าอัตราส่วนความไม่สมดุล)

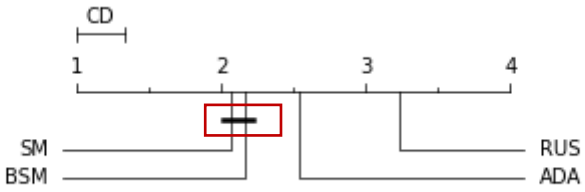
Mean	IR 2.3				IR 10				IR 14			
Rank	SM	RUS	ADA	BSM	SM	RUS	ADA	BSM	SM	RUS	ADA	BSM
ACC	1.875	3.375	2.500	2.250	2.750	2.500	3.250	1.500	2.500	2.500	3.250	1.750
AUC	2.000	3.250	2.500	2.250	2.750	1.750	3.000	2.500	3.000	1.750	3.000	2.250
F1	2.250	3.000	2.250	2.500	2.750	2.500	3.250	1.500	2.500	2.500	3.250	1.750
PRE	1.750	3.750	3.000	1.500	2.125	2.500	2.875	2.500	2.875	2.500	2.750	1.875
KS	2.250	3.500	2.500	1.750	2.750	1.750	2.500	3.000	3.000	3.250	2.250	1.500
BS	2.250	2.500	2.500	2.750	3.000	1.750	3.250	2.000	2.625	1.750	3.500	2.125
Average	**2.062**	3.229	2.541	2.166	2.687	**2.125**	3.020	2.166	2.750	2.375	3.000	**1.875**

จากตารางที่ 4.10 พบว่าวิธีการสุ่มตัวอย่างซ้ำที่ดีที่สุดขึ้นอยู่กับอัตราส่วนความไม่สมดุล โดยที่อัตราส่วนความไม่สมดุลที่ 2.3 วิธี SM ดีที่สุด อัตราส่วนความไม่สมดุลที่ 10 วิธี RUS ดีที่สุด และอัตราส่วนความไม่สมดุลที่ 14 วิธี BSM ดีที่สุด โดยมีค่าเฉลี่ยอันดับ 1.958, 2.125 และ 1.895 ตามลำดับ จากนั้นใช้การทดสอบ Friedman เพื่อแสดงว่าประสิทธิภาพวิธีการสุ่มตัวอย่างซ้ำของแต่ละอัตราส่วนความไม่สมดุลแตกต่างกันที่ระดับนัยสำคัญ 0.05 ได้ผลลัพธ์ดังตารางที่ 4.11

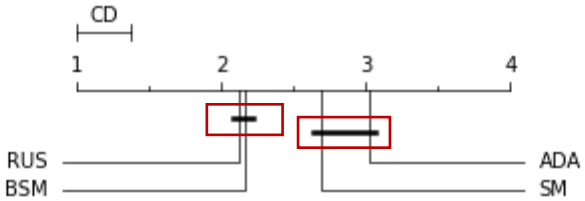
ตารางที่ 4.11 สถิติทดสอบ Friedman ในแต่ละอัตราส่วนความไม่สมดุลที่แตกต่างกัน

Imbalance Ratio (IR)	Friedman Statistic	p-value
IR 2.3	10.706	0.013
IR 10	8.796	0.032
IR 14	10.052	0.018

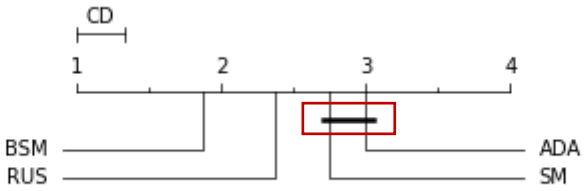
จากตารางที่ 4.11 พบว่าในอัตราส่วนความไม่สมดุลที่ 2.3, 10 และ 14 วิธีการสุ่มตัวอย่างซ้ำ 4 วิธี ได้แก่ Systematic Minority Over-Sampling Technique (SM), Random Under-Sampling (RUS), Adaptive Synthetic (ADA) และ Borderline SMOTE (BSM) มีประสิทธิภาพแตกต่างกันในแต่ละอัตราส่วนความไม่สมดุลที่ระดับนัยสำคัญ 0.05 จึงนำไปทดสอบ Neymanyi post hoc เพื่อหาว่าวิธีการสุ่มตัวอย่างซ้ำใดมีความแตกต่างกัน โดยจะแตกต่างกันเมื่ออันดับเฉลี่ยต่างกันอย่างน้อย Critical Difference (CD) ถ้าวิธีการสุ่มตัวอย่างซ้ำที่ไม่แตกต่างกันจะมีเส้นสีดำขีดระหว่างอันดับเฉลี่ยของแต่ละตัวแบบ ดังกรอบสีแดงด้านล่าง และจากการทดสอบ Nemenyi post hoc ได้ค่า critical difference (CD) ในแต่ละอัตราอัตราส่วนความไม่สมดุลดังนี้ IR 2.3 (CD = 0.331) ดังรูปที่ 4.4, IR 10 (CD = 0.377) ดังรูปที่ 4.5 และ IR 14 (CD= 0.331) ดังรูปที่ 4.6



รูปที่ 4.12 แผนภาพแสดงอันดับเฉลี่ยของการสุ่มตัวอย่างซ้ำของอัตราส่วนความไม่สมดุลที่ 2.3 โดยใช้ Neymanyi post hoc



รูปที่ 4.13 แผนภาพแสดงอันดับเฉลี่ยของการสุ่มตัวอย่างซ้ำของอัตราส่วนความไม่สมดุลที่ 10 โดยใช้ Neymanyi post hoc



รูปที่ 4.14 แผนภาพแสดงอันดับเฉลี่ยของการสุ่มตัวอย่างซ้ำของอัตราส่วนความไม่สมดุลที่ 14 โดยใช้ Neymanyi post hoc

จากแผนภาพแสดงอันดับเฉลี่ยของวิธีการสุ่มตัวอย่างซ้ำของอัตราส่วนความไม่สมดุลที่ 2.3, 10 และ 14 โดยใช้ Neymanyi post hoc ดังรูปที่ 4.4, 4.5 และ 4.6 ตามลำดับ พบว่าในอัตราส่วนความไม่สมดุลต่ำ (IR = 2.3) SM มีอันดับเฉลี่ยดีที่สุดแต่ไม่มีหลักฐานเพียงพอที่จะสรุปว่าแตกต่างกันกับ BSM อัตราส่วนความไม่สมดุลกลาง (IR = 10) RUS มีอันดับเฉลี่ยดีที่สุดแต่ไม่มีหลักฐานเพียงพอที่จะสรุปว่าแตกต่างกันกับ BSM และอัตราส่วนความไม่สมดุลสูง (IR = 14) BSM มีค่าเฉลี่ยดีที่สุดในกรณีที่ไม่มีหลักฐานเพียงพอที่จะสรุปว่าวิธีการสุ่มตัวอย่างซ้ำแตกต่างกัน ผู้วิจัยเลือกจากอันดับเฉลี่ยที่ต่ำที่สุดในแต่ละค่าอัตราส่วนความไม่สมดุลนั้นๆ ดังนั้นจากผลการเปรียบเทียบประสิทธิภาพของวิธีการสุ่มตัวอย่างซ้ำในแต่ละอัตราส่วนความไม่สมดุล SM, RUS และ BSM เป็นวิธีการสุ่มตัวอย่างซ้ำที่มีประสิทธิภาพเฉลี่ยดีที่สุดในอัตราส่วนความไม่สมดุลต่ำ (IR = 2.3), อัตราส่วนความไม่สมดุลกลาง (IR = 10) และ อัตราส่วนความไม่สมดุลสูง (IR = 14) ตามลำดับ

ผลการเปรียบเทียบประสิทธิภาพของการเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนาน , ตัวแบบการสุ่มป่าไม้ และตัวแบบการจำแนกประเภทพื้นฐาน 4 ตัวแบบ ที่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำที่เหมาะสมในแต่ละอัตราส่วนความไม่สมดุลจากผลการศึกษา 4.2 และ 4.3 ตามลำดับ

จากการเปรียบเทียบวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำเพื่อหาวิธีที่มีประสิทธิภาพเฉลี่ยดีที่สุดในแต่ละอัตราส่วนความไม่สมดุล ได้ผลดังนี้ ในอัตราส่วนความไม่สมดุลต่ำ (IR = 2.3) วิธีการลดมิติข้อมูลและวิธีการสุ่มตัวอย่างซ้ำที่มีประสิทธิภาพคือ Linear Discriminant Analysis (LDA) และ Systematic Minority Over-Sampling Technique (SM) อัตราส่วนความไม่สมดุลกลาง (IR = 10) วิธีการลดมิติข้อมูลและวิธีการสุ่มตัวอย่างซ้ำที่มีประสิทธิภาพคือ Linear Discriminant Analysis (LDA) และ Random Under-Sampling (RUS) และ อัตราส่วนความไม่สมดุลสูง (IR = 14) วิธีการลดมิติข้อมูลและวิธีการสุ่มตัวอย่างซ้ำที่มีประสิทธิภาพคือ Linear

Discriminant Analysis (LDA) และ Borderline SMOTE (BSM) จากนั้นนำวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำที่มีประสิทธิภาพเฉลี่ยดีที่สุดในแต่ละอัตราส่วนความไม่สมดุลดังกล่าว ไปใช้เปรียบเทียบประสิทธิภาพตัวแบบการจำแนกประเภทเมื่อมีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำ ได้ผลลัพธ์ดังตารางที่ 4.12

ในตารางที่ 4.12 แสดงผลการเปรียบเทียบประสิทธิภาพตัวแบบในแต่ละอัตราส่วนความไม่สมดุล โดยวัดประสิทธิภาพตัวแบบด้วยตัวชี้วัด Accuracy (ACC), The area under the curve (AUC), The F1-score (F1), Precision (PRE), Kolmogorov-Smirnov (KS) และ Brier Score (BS) ได้ผลดังนี้

ตารางที่ 4.12 ประสิทธิภาพของตัวแบบจำแนกประเภทที่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำ

IR	Performance	Bagging Hetero	RF	LR	KNN	DT	SVM
2.3	ACC	0.720	0.660	0.750	0.740	0.750	0.745
	AUC	0.784	0.693	0.792	0.742	0.767	0.681
	F1	0.787	0.757	0.823	0.819	0.828	0.823
	PRE	0.845	0.762	0.818	0.802	0.801	0.804
	KS	0.453	0.322	0.488	0.409	0.450	0.243
	BS	0.180	0.257	0.173	0.183	0.187	0.250
10	ACC	0.701	0.610	0.662	0.662	0.629	0.701
	AUC	0.741	0.669	0.739	0.720	0.559	0.523
	F1	0.816	0.750	0.785	0.786	0.765	0.818
	PRE	0.980	0.957	0.989	0.979	0.958	0.962
	KS	0.565	0.402	0.532	0.464	0.200	0.149
	BS	0.193	0.291	0.263	0.260	0.355	0.250
14	ACC	0.913	0.926	0.893	0.913	0.900	0.933
	AUC	0.820	0.808	0.788	0.790	0.648	0.785
	F1	0.953	0.961	0.942	0.954	0.946	0.965
	PRE	0.964	0.958	0.956	0.957	0.963	0.952
	KS	0.642	0.642	0.635	0.614	0.387	0.614
	BS	0.065	0.066	0.077	0.071	0.072	0.069

จากการเปรียบเทียบตัวแบบที่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำที่ได้จากผลการศึกษาที่ 4.2 และ 4.3 ผู้วิจัยทำการให้อันดับในแต่ละตัวชี้วัดที่แสดงถึงประสิทธิภาพของตัวแบบ โดยอันดับที่น้อยลงหมายถึงตัวแบบมีประสิทธิภาพมากขึ้นแต่ละตัวชี้วัด จากนั้นหาอันดับเฉลี่ยของแต่ละตัวแบบที่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำ ได้ผลลัพธ์ดังตารางที่ 4.13

ตารางที่ 4.13 อันดับในแต่ละตัวชี้วัดและอันดับเฉลี่ยของแต่ละตัวแบบจำแนกประเภทที่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำ (ตัวหนาแสดงตัวแบบที่มีประสิทธิภาพสูงสุดในแต่ละค่าอัตราส่วนความไม่สมดุล)

IR	Performance	Bagging Hetero	RF	LR	KNN	DT	SVM
2.3	ACC	5	6	1.5	4	1.5	3
	AUC	2	5	1	4	3	6
	F1	5	6	2	4	1	3
	PRE	1	6	2	4	5	3
	KS	2	5	1	4	3	6
	BS	2	6	1	3	4	5
	Mean Rank	2.833	5.666	**1.416**	3.833	2.916	4.333
10	ACC	1.5	6	3.5	3.5	5	1.5
	AUC	1	4	2	3	5	6
	F1	2	6	4	3	5	1
	PRE	2	6	1	3	5	4
	KS	1	4	2	3	5	6
	BS	1	5	4	3	6	2
	Mean Rank	**1.416**	5.166	2.750	3.083	5.166	3.416
14	ACC	3.5	2	6	3.5	5	1
	AUC	1	2	4	3	6	5
	F1	4	2	6	3	5	1
	PRE	1	3	5	4	2	6
	KS	1.5	1.5	3	4.5	6	4.5
	BS	1	2	6	4	5	3
	Mean Rank	**2.000**	2.083	5.000	3.666	4.833	3.416

จากตารางที่ 4.13 แสดงอันดับของตัวแบบที่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำในแต่ละตัวชี้วัดและอันดับเฉลี่ยของอัตราส่วนความไม่สมดุลที่ 2.3, 10 และ 14 พบว่า ตัวแบบที่มีเทคนิคการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำที่ดีที่สุดขึ้นอยู่กับค่าอัตราส่วนความไม่สมดุล โดยอัตราส่วนความไม่สมดุลที่ 2.3 ตัวแบบ LR มีประสิทธิภาพดีที่สุด อัตราส่วนความไม่สมดุลที่ 10 และ 14 ตัวแบบ Bagging Hetero มีประสิทธิภาพดีที่สุด มีอันดับเฉลี่ย 1.416, 1.416 และ 2.000 ตามลำดับ

ในอัตราส่วนความไม่สมดุลที่ 2.3 (IR = 2.3) ตัวแบบที่มีวิธีการลดมิติข้อมูล LDA และการสุ่มตัวอย่างซ้ำ SM ที่ประสิทธิภาพรองลงมาคือ ตัวแบบ Bagging Hetero, ตัวแบบต้นไม้ตัดสินใจ (DT), ตัวแบบเพื่อนบ้านใกล้สุด (KNN), ตัวแบบซัพพอร์ตเวคเตอร์แมชชีน (SVM) และ ตัวแบบการสุ่มป่าไม้ โดยมีอันดับเฉลี่ย 2.833, 2.916, 3.833, 4.333 และ 5.666 ตามลำดับ

อัตราส่วนความไม่สมดุลที่ 10 (IR = 10) ตัวแบบที่มีวิธีการลดมิติข้อมูล LDA และการสุ่มตัวอย่างซ้ำ RUS ที่ประสิทธิภาพรองลงมาคือ ตัวแบบการถดถอยโลจิสติก (LR), ตัวแบบเพื่อนบ้านใกล้สุด (KNN) , ตัวแบบซัพพอร์ตเวคเตอร์แมชชีน (SVM), ตัวแบบการสุ่มป่าไม้ (RF) และ ตัวแบบต้นไม้การตัดสินใจ (DT) โดยมีอันดับเฉลี่ย 2.750, 3.083, 3.416, 5.166 และ 5.166 ตามลำดับ

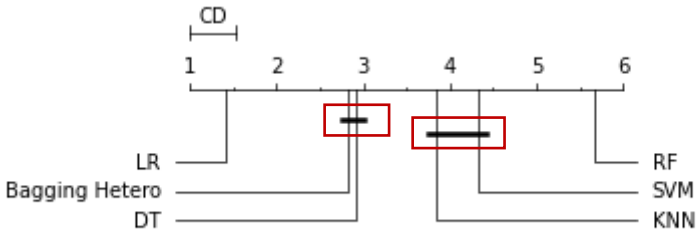
อัตราส่วนความไม่สมดุลที่ 14 (IR = 14) ตัวแบบที่มีวิธีการลดมิติข้อมูล LDA และการสุ่มตัวอย่างซ้ำ BSMที่ประสิทธิภาพรองลงมาคือ ตัวแบบการสุ่มป่าไม้ (RF), ตัวแบบซัพพอร์ตเวคเตอร์แมชชีน (SVM), ตัวแบบเพื่อนบ้านใกล้สุด (KNN), ตัวแบบต้นไม้การตัดสินใจ (DT) และ ตัวแบบการถดถอยโลจิสติก โดยมีอันดับเฉลี่ย 2.083, 3.416, 3.666, 4.833 และ 5.000 ตามลำดับ

จากนั้นใช้การทดสอบ Friedman เพื่อแสดงว่าประสิทธิภาพของตัวแบบที่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำแตกต่างกันที่ระดับนัยสำคัญ 0.05 ได้ผลลัพธ์ดังตารางที่ 4.14

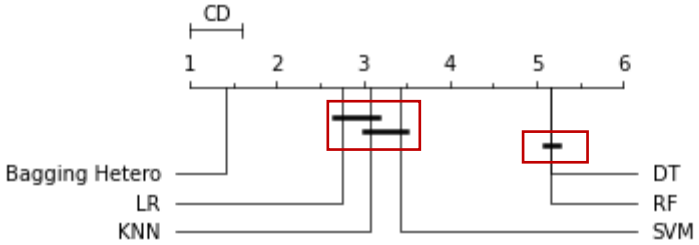
ตารางที่ 4.14 สถิติทดสอบ Friedman ในแต่ละค่าอัตราส่วนความไม่สมดุล

Imbalance Ratio (IR)	Friedman Statistic	p-value
IR 2.3	18.301	0.002
IR 10	18.413	0.002
IR 14	14.468	0.012

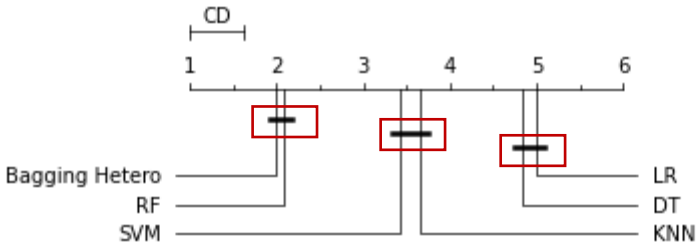
จากตารางที่ 4.14 พบว่าในอัตราส่วนความไม่สมดุลที่ 2.3, 10 และ 14 การเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนาน (Bagging Hetero), ตัวแบบการสุ่มป่าไม้ (RF) และตัวแบบการจำแนกประเภทพื้นฐาน 4 ตัวแบบ ได้แก่ Logistic Regression (LR), K-Nearest Neighbors (KNN), Decision Tree (DT), Support Vector Machine (SVM) ที่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำที่ได้จากการทดลองที่ 2 และ 3 มีประสิทธิภาพแตกต่างกันที่ระดับนัยสำคัญ 0.05 จึงนำไปทดสอบ Neymanyi post hoc เพื่อหาว่าตัวแบบคู่ใดมีความแตกต่างกัน โดยตัวแบบจะแตกต่างกันเมื่ออันดับเฉลี่ยต่างกันอย่างน้อย Critical Difference (CD) ถ้าตัวแบบที่ไม่แตกต่างกันจะมีเส้นสีดำขีดระหว่างอันดับเฉลี่ยของแต่ละตัวแบบ ดังกรอบสีแดงด้านล่าง และจากการทดสอบ Nemenyi post hoc ได้ค่า critical difference (CD) ในแต่ละอัตราส่วนความไม่สมดุลดังนี้ IR 2.3 (CD = 0.533) ดังรูปที่ 4.7, IR 10 (CD = 0.607) ดังรูปที่ 4.8 และ IR 14 (CD= 0.615) ดังรูปที่ 4.9



รูปที่ 4.15 แผนภาพแสดงอันดับเฉลี่ยของตัวแบบที่มีวิธีการลดมิติข้อมูล LDA และการสุ่มตัวอย่างซ้ำ SM ของอัตราส่วนความไม่สมดุลที่ 2.3 โดยใช้ Neymanyi post hoc



รูปที่ 4.16 แผนภาพแสดงอันดับเฉลี่ยของตัวแบบที่มีวิธีการลดมิติข้อมูล LDA และการสุ่มตัวอย่างซ้ำ RUS ของอัตราส่วนความไม่สมดุลที่ 10 โดยใช้ Neymanyi post hoc



รูปที่ 4.17 แผนภาพแสดงอันดับเฉลี่ยของตัวแบบที่มีวิธีการลดมิติข้อมูล LDA และการสุ่มตัวอย่างซ้ำ BSM ของอัตราส่วนความไม่สมดุลที่ 14 โดยใช้ Neymanyi post hoc

จากแผนภาพแสดงอันดับเฉลี่ยของตัวแบบที่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำของอัตราส่วนความไม่สมดุลที่ 2.3, 10 และ 14 โดยใช้ Neymanyi post hoc ดังรูปที่ 4.7, 4.8 และ 4.9 ตามลำดับ พบว่าในอัตราส่วนความไม่สมดุลต่ำ (IR = 2.3) LR มีอันดับเฉลี่ยดีที่สุด อัตราส่วนความไม่สมดุลกลาง (IR = 10) Bagging Hetero มีอันดับเฉลี่ยดีที่สุด และอัตราส่วนความไม่สมดุลสูง (IR = 14) Bagging Hetero มีอันดับเฉลี่ยดีที่สุดแต่ไม่มีหลักฐานเพียงพอที่จะสรุปว่าแตกต่างกันกับ RF

บทที่ 5
สรุปผลการวิจัย และข้อเสนอแนะ

งานวิจัยนี้มีวัตถุประสงค์เพื่อสร้างตัวแบบ Bagging Heterogeneous Ensemble ที่เหมาะสมกับอัตราส่วนความไม่สมดุล (imbalance ratio) ที่แตกต่างกันของข้อมูลคะแนนเครดิต (credit scoring) และหาเทคนิคการลดมิติข้อมูล (dimensionality reduction) และเทคนิคการสุ่มตัวอย่างซ้ำ (resampling) ที่เหมาะสมกับอัตราส่วนความไม่สมดุล (imbalance ratio) ที่แตกต่างกันของข้อมูลคะแนนเครดิต โดยมีการสรุปผลการวิจัยดังนี้

สรุปผลการวิจัย

จากการศึกษาและเปรียบเทียบตัวแบบการจำแนกประเภทข้อมูลเยอรมันเครดิตในแต่ละอัตราส่วนความไม่สมดุลที่แตกต่างกัน และหาเทคนิคการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำที่เหมาะสมในแต่ละค่าอัตราส่วนความไม่สมดุล ในการเพิ่มประสิทธิภาพของตัวแบบจะใช้ตัวแบบการเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนาน ร่วมกับเทคนิคการลดมิติข้อมูล และใช้เทคนิคการสุ่มตัวอย่างซ้ำจัดการกับอัตราส่วนความไม่สมดุลที่แตกต่างกัน โดยวัดประสิทธิภาพของตัวแบบด้วยตัวชี้วัด ACC, AUC, F1, PRE, BS และ KS รวมทั้งการทดสอบทางสถิติเพื่อแสดงว่าประสิทธิภาพของตัวแบบมีความแตกต่างกัน

ผลการศึกษาพบว่าในอัตราส่วนความไม่สมดุล 2.3 วิธีการลดมิติข้อมูลและวิธีการสุ่มตัวอย่างซ้ำที่มีประสิทธิภาพคือ Linear Discriminant Analysis (LDA) และ Systematic Minority Over-Sampling Technique (SM) ตามลำดับ โดยที่การเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบขนานมีประสิทธิภาพเฉลี่ยดีที่สุดในกรณีที่ไม่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำ แต่ในกรณีที่มีวิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำพบว่าไม่ใช่ตัวแบบที่มีประสิทธิภาพมากที่สุด เนื่องจากตัวแบบมีความซับซ้อนมากเกินไป จึงสรุปได้ว่าในค่าอัตราส่วนความไม่สมดุลต่ำ (IR = 2.3) ตัวแบบ Logistic Regression ที่ใช้เทคนิค Linear Discriminant Analysis (LDA) และ Systematic Minority Over-Sampling Technique (SM) จะมีประสิทธิภาพเฉลี่ยดีที่สุดในการจำแนกประเภทในส่วนของอัตราส่วนความไม่สมดุลกลาง (IR = 10) และ อัตราส่วนความไม่สมดุลสูง (IR = 14) วิธีการลดมิติข้อมูลและการสุ่มตัวอย่างซ้ำที่มีประสิทธิภาพคือ Linear Discriminant Analysis (LDA), Random Under-Sampling (RUS) และ Linear Discriminant Analysis (LDA), Borderline SMOTE (BSM) ตามลำดับ โดยที่การเรียนรู้แบบรวมกลุ่มด้วยตัวแบบที่แตกต่างกันแบบ

ขนานมีประสิทธิภาพเฉลี่ยดีที่สุด ทั้งในกรณีที่มีและไม่มีวิธีการลดมิติข้อมูลและสุ่มตัวอย่างซ้ำของอัตราส่วนความไม่สมดุลกลางและสูง เนื่องจากเมื่อข้อมูลมีอัตราส่วนความไม่สมดุลที่มากขึ้น การแก้ไขปัญหาระดับ Algorithm level คือการเรียนรู้แบบรวมกลุ่มรวมกับการลดมิติข้อมูล และ Data level คือการสุ่มตัวอย่างซ้ำ จะช่วยเพิ่มประสิทธิภาพของการจำแนกประเภทของตัวแบบได้

ข้อจำกัด

ผลการศึกษาจากงานวิจัยสามารถใช้ได้เฉพาะกับข้อมูลคะแนนเครดิต (credit scoring)

ข้อเสนอแนะ

ในงานวิจัยนี้ผู้วิจัยได้ศึกษาการเรียนรู้กลุ่มแบบขนาน (Bagging) ด้วยตัวแบบที่แตกต่างกัน ผู้ที่สนใจสามารถศึกษาการเรียนรู้แบบรวมกลุ่ม (Ensemble learning) ประเภทอื่นๆได้ เช่น Boosting หรือ Stacking เป็นต้น ในส่วนของ Data level สามารถเพิ่มเทคนิคการจัดการกับข้อมูลที่ไม่สมดุลด้วยเทคนิค Cost-Sensitive และทำการทดลองการเปรียบเทียบตัวแบบกับชุดข้อมูลไม่สมดุลที่แตกต่างกัน รวมทั้งสามารถใช้การคัดเลือกคุณลักษณะ (Feature Selection) ก่อนสร้างตัวแบบการจำแนกประเภทได้

บรรณานุกรม

Alam, B. (2022). *Decision Tree Python – Easy Tutorial*. Retrieved Dec 1, 2022 from https://hands-on.cloud/decision-tree-python-tutorial/

Anowar, F., Sadaoui, S., & Selim, B. (2021). Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE). *Computer Science Review*, *40*, 100378. https://doi.org/https://doi.org/10.1016/j.cosrev.2021.100378

Brown, I., & Mues, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert systems with applications*, *39*(3), 3446-3453.

De Melo Junior, L. S., Nardini, F. M., Renso, C., & de Macêdo, J. A. F. (2019). An empirical comparison of classification algorithms for imbalanced credit scoring datasets. 2019 18th IEEE international conference on machine learning and applications (ICMLA),

Debbaut, P., Ghent, A. C., & Kudlyak, M. (2014). Are young borrowers bad borrowers? Evidence from the Credit CARD Act of 2009.

Feng, X., Xiao, Z., Zhong, B., Qiu, J., & Dong, Y. (2018). Dynamic ensemble classification for credit scoring using soft probability. *Applied Soft Computing*, *65*, 139-151.

Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert systems with applications*, *73*, 220-239.

Holmes, T. E. (2021). *Credit card race, age, gender and income statistics*. Retrieved Dec 1, 2022 from https://www.creditcards.com/statistics/race-age-gender-statistics/

Irby, L. (2021). *How Long Does It Take Your Credit Score to Improve?* Retrieved Dec 1, 2022 from https://www.thebalancemoney.com/how-long-does-it-take-for-your-credit-score-to-improve-960555

JEAN-CHRISTOPHE-CHOUINARD. (2022). *k-Nearest Neighbors (KNN) in Python*. Retrieved Dec 1, 2022 from https://www.jcchouinard.com/k-nearest-neighbors/

Lenka, S. R., Bisoy, S. K., Priyadarshini, R., & Sain, M. (2022). Empirical analysis of

ensemble learning for imbalanced credit scoring datasets: A systematic review. *Wireless Communications and Mobile Computing*, *2022*.

Lopes, P. (2008). Credit card debt and default over the life cycle. *Journal of Money, Credit and Banking*, *40*(4), 769-790.

Marqués, A. I., García, V., & Sánchez, J. S. (2013). On the suitability of resampling techniques for the class imbalance problem in credit scoring. *Journal of the Operational Research Society*, *64*(7), 1060-1070. https://doi.org/10.1057/jors.2012.120

Patel, A. (2019). *Bagging — Ensemble meta Algorithm for Reducing variance*. Retrieved Dec 1, 2022 from https://medium.com/ml-research-lab/bagging-ensemble-meta-algorithm-for-reducing-variance-c98fffa5489f

PradyaSin. (2019). *Support Vector Machines (SVM).* Retrieved Dec 1, 2022 from https://medium.com/@pradyasin/support-vector-machines-svm-943f9a732a69

Raschka, S. (2014). *Linear Discriminant Analysis*. Retrieved Dec 1, 2022 from https://sebastianraschka.com/Articles/2014_python_lda.html

White, M. J. (2007). Bankruptcy reform and credit cards. *Journal of Economic Perspectives*, *21*(4), 175-199.

Yu, L., Zhou, R., Tang, L., & Chen, R. (2018). A DBN-based resampling SVM ensemble learning paradigm for credit classification with imbalanced data. *Applied Soft Computing*, *69*, 192-202.

Zhang, T., & Chi, G. (2021). A heterogeneous ensemble credit scoring model based on adaptive classifier selection: An application on imbalanced data. *International Journal of Finance & Economics*, *26*(3), 4372-4385.

ภาคผนวก

คำสั่งการวิเคราะห์ข้อมูลด้วยโปรแกรม Python

```
########## Install & Import libraries ##########
!pip install sklearn-genetic
!pip install imbalanced-learn
!pip install scikit-learn
!pip install orange3

import numpy as np
import pandas as pd
import random
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, precision_score,
f1_score, brier_score_loss, make_scorer
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import VotingClassifier
from scipy.stats import ks_2samp
from scipy.stats import mode
from scipy.stats import friedmanchisquare
from scipy import stats
from scipy.stats.distributions import chi2
from numpy import array
from imblearn.pipeline import Pipeline
from sklearn.decomposition import TruncatedSVD
```

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
import scipy.stats as stats
from imblearn.over_sampling import SMOTE, ADASYN, BorderlineSMOTE
from imblearn.under_sampling import RandomUnderSampler

########## Importing Dataset ##########
header_list = [
            "Status of existing checking account", "Duration in month", "Credit history",
            "Purpose", "Credit amount", "Savings account", "Present employment since",
"Installment rate",
            "Personal status and sex", "Other debtors", "Present residence since",
"Property",
            "Age", "Other installment plans", "Housing", "Number of existing credits at
this bank",
            "Job", "Number of people being liable to provide maintenance for",
"Telephone", "foreign worker", 'Risk'
           ]
df = pd.read_csv('data_ger',header=None, sep=" ", names=header_list,
index_col=False)

########## Data preparation steps ##########
#standardized and get dummy (binary encoding)
df_full = df.copy()
# Get binary encoding of columns
df_full = pd.get_dummies(df_full, drop_first=True)
# Get names of numeric columns
cols_num = df.select_dtypes(np.number).columns.tolist()
# Standardized
for col in cols_num:
  if col != 'Risk':
    df_full[col] = (df_full[col] - df_full[col].mean()) / df_full[col].std()
```

```
########## Function ##########
def generate_ir(df=df, ir = 2.3, seed=123):
  dict_value = df['Risk'].value_counts().to_dict()
  max_value = max(dict_value, key=dict_value.get)
  min_value = min(dict_value, key=dict_value.get)
  del_minority = dict_value[max_value] / ir
  if(del_minority < dict_value[min_value]):
    np.random.seed(seed)
    remove_n = dict_value[min_value] - round(del_minority)
    drop_indexs = np.random.choice(df[df['Risk']==min_value].index, remove_n,
replace=False)
    df_subset = df.drop(drop_indexs)
    df_subset = df_subset.reset_index(drop=True)
    # print('IR: ',
df_subset['Risk'].value_counts().to_dict()[max_value]/df_subset['Risk'].value_counts().to
_dict()[min_value])
    return df_subset
  else:
    print('Error')
```



```
def generate_df_IR(df, ir, seed):
  # function generate ir
  df_generate = generate_ir(df, ir, seed)

  # train-test split
  X_train, X_test, y_train, y_test = train_test_split(
            df_generate.drop(['Risk'], axis=1), df_generate['Risk'], test_size=0.2,
random_state=seed*2)
  return X_train, X_test, y_train, y_test
```

```
X_train, X_test, y_train, y_test = train_test_split(
                df_full.drop(['Risk'], axis=1), df_full['Risk'], test_size=0.2,
random_state=rand[0])
X_train10, X_test10, y_train10, y_test10 = generate_df_IR(df_full, 10, rand[1])
X_train14, X_test14, y_train14, y_test14 = generate_df_IR(df_full, 14, rand[2])

display(y_train10.value_counts(), y_test10.value_counts(), y_train14.value_counts(),
y_test14.value_counts())

########## Experiment 1 ##########
#Function
import random
rand = np.random.randint(1000, size=100)

def get_models():
  models = dict() # dict ของแต่ละ models
  models['lr'] = LogisticRegression(random_state=rand[0])
  models['knn'] = KNeighborsClassifier()
  models['cart'] = DecisionTreeClassifier(random_state=rand[0])
  models['svm'] = SVC(probability=True, random_state=rand[0])
  models['rf'] = RandomForestClassifier(random_state=rand[0])
  return models

def get_models_bagging():
  models = dict() # dict ของแต่ละ models
  models['lr'] = LogisticRegression(random_state=rand[0])
  models['knn'] = KNeighborsClassifier()
  models['cart'] = DecisionTreeClassifier(random_state=rand[0])
  models['svm'] = SVC(probability=True, random_state=rand[0])
  return models
```

```
def get_parameters():
  parameters = dict()
  parameters['lr'] = {'penalty':('l1', 'l2', 'elasticnet', 'none'),
                      'solver':('newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'),
                      'C':np.array([100, 10, 1.0, 0.1, 0.01])}
  parameters['knn'] = {'n_neighbors':np.arange(10, 20),
                       'leaf_size':np.arange(25,35),
                       'weights':('uniform', 'distance'),
                       'metric':('euclidean', 'minkowski', 'cityblock')}
  parameters['cart'] = {'ccp_alpha':np.array([0, 0.1, 0.01]),
                        'min_samples_leaf':np.array([1,2,3,4]),
                        'criterion':('gini', 'entropy'),
                        'max_depth':np.array([None, 5, 10, 15]),
                        'max_features':('auto', 'sqrt', 'log2')}
  parameters['svm'] = {'C':np.array([100, 10, 1.0, 0.1, 0.01, 0.001, 0.0001]),
                       'gamma':('scale', 'auto'),
                       'kernel':('linear', 'poly', 'rbf', 'sigmoid'),
                       'class_weight':('balanced', None)}
  parameters['rf'] = {"criterion":("gini", "entropy", "log_loss"), "max_features": ("auto",
"sqrt", "log2"), "min_samples_split" : np.array([2, 5, 10]) }
  return parameters

def get_parameters_bagging():
  parameters = dict()
  parameters['lr'] = {'penalty':('l1', 'l2', 'elasticnet', 'none'),
                      'solver':('newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'),
                      'C':np.array([100, 10, 1.0, 0.1, 0.01])}
  parameters['knn'] = {'n_neighbors':np.arange(10, 20),
                       'leaf_size':np.arange(25,35),
                       'weights':('uniform', 'distance'),
                       'metric':('euclidean', 'minkowski', 'cityblock')}
```

```
  parameters['cart'] = {'ccp_alpha':np.array([0, 0.1, 0.01]),
                        'min_samples_leaf':np.array([1,2,3,4]),
                        'criterion':('gini', 'entropy'),
                        'max_depth':np.array([None, 5, 10, 15]),
                        'max_features':('auto', 'sqrt', 'log2')}
  parameters['svm'] = {'C':np.array([100, 10, 1.0, 0.1, 0.01, 0.001, 0.0001]),
                       'gamma':('scale', 'auto'),
                       'kernel':('linear', 'poly', 'rbf', 'sigmoid'),
                       'class_weight':('balanced', None)}
  return parameters


list_models = get_models()
list_models_bagging = get_models_bagging()
list_parameters = get_parameters()
list_parameters_bagging = get_parameters_bagging()

########## Function bagging hetero ##########
def new_dict_parameter(list_parameters=list_parameters):
    new_parameter = dict()
    for i,j in list_parameters.items():
      for k,l in j.items():
        text = str(i) + '__' + str(k)
        new_parameter[text] = l
    return new_parameter


def bagging_fit(X, y, n_estimators, percent_max_samples=0.95, list_models=
list_models_bagging, list_parameters= list_parameters_bagging):
    new_parameter = new_dict_parameter(list_parameters=list_parameters)
    scoring = 'roc_auc'
    n_examples = len(y)
    estimators = [ VotingClassifier(estimators = [
```

```
                    ('lr', list_models['lr']),
                    ('knn', list_models['knn']),
                    ('cart', list_models['cart']),
                    ('svm', list_models['svm'])
                ], voting='soft') for _ in range(n_estimators)]
    search_estimators = list()
    best_parameter = list()

    for count_number, voting in enumerate(estimators):
        bag = np.random.choice(n_examples,
round(n_examples*percent_max_samples), replace=False)
        clf = RandomizedSearchCV(voting, new_parameter, scoring=scoring, verbose=2,
n_jobs=-1, cv=3,  n_iter=500, random_state=rand[count_number])
        clf.fit(X.iloc[bag].values, y.iloc[bag])
        search_estimators.append(clf)
        best_parameter.append(clf.best_params_)
    return best_parameter, search_estimators

def bagging_predict(X, search_estimators, n_estimators, vote=True):
  if vote:
    all_predictions = [estimator.predict(X.values) for estimator in estimators]
    all_predictions = np.array(all_predictions)
    ypred, _ = mode(all_predictions, axis=0)
    ypred = np.squeeze(ypred)
    y_prob = np.count_nonzero(all_predictions == 2, axis=0)/n_estimators
    return y_prob, ypred
  else:
    all_predictions_proba = [estimator.predict_proba(X.values)[:,1] for estimator in
estimators]
    all_predictions_proba = np.array(all_predictions_proba)
    y_prob = all_predictions_proba.sum(axis=0)/n_estimators
```

```
    ypred = np.where(y_prob < 0.5, 1, 2)
    return y_prob, ypred

########## Function Base model ##########
        def classifier_selection(X_train_, y_train_, X_test_, y_test_, model, parameters,
rand=rand[0]):
  scoring = 'roc_auc'
  clf = RandomizedSearchCV(model, parameters, scoring=scoring ,verbose=2, n_jobs=-
1, cv=3,  n_iter=20, random_state=rand)
  clf.fit(X_train_, y_train_)
  y_probs = clf.predict_proba(X_test_)[:, 1]
  y_pred = clf.predict(X_test_)
  return y_probs, y_pred, clf.best_params_

########## Function performance ##########
def kss(y_real, y_proba):
  df = pd.DataFrame()
  df['real'] = y_real
  df['proba'] = y_proba
  # Recover each class
  class1 = df[df['real'] == 1]
  class2 = df[df['real'] == 2]
  ks = ks_2samp(class1['proba'], class2['proba'])[0]
  return ks

def result_(y_real, y_hat, y_prob):
  return [  accuracy_score(y_real, y_hat), roc_auc_score(y_real, y_prob),
            f1_score(y_real, y_hat), precision_score(y_real, y_hat),
            kss(y_real, y_prob), brier_score_loss(y_real, y_prob)
            ]
```

```
def generate_df_result(input_dict):
  df_result = pd.DataFrame(input_dict, index=['ACC', 'AUC', 'F1', 'PRE', 'KSS', 'BS'])
  df_result_rank = df_result.rank(ascending=False, axis=1)
  df_result_rank = df_result_rank.drop(['BS'], axis=0)
  df_result_rank = pd.concat([df_result_rank,
pd.DataFrame(df_result.loc['BS'].rank(ascending=True, axis=0)).transpose()])
  df_result_rank = pd.concat([df_result_rank,
pd.DataFrame(df_result_rank.mean(axis=0), columns=['Mean']).transpose()])
  return df_result, df_result_rank


def friedman_custom(df_rank):
  tmp = df_rank.iloc[:-1].to_numpy()
  k = tmp.shape[1]
  n = tmp.shape[0]
  ties = 0
  for d in tmp:
      replist, repnum = stats.find_repeats(array(d))
      for t in repnum:
          ties += t * (t*t - 1)
  c = 1 - ties / (k*(k*k - 1)*n)
  ssbn = np.sum(tmp.sum(axis=0)**2)
  chisq = (12.0 / (k*n*(k+1)) * ssbn - 3*n*(k+1)) / c
  f_res = pd.DataFrame({'test':'Friedman','statistic':chisq, 'pvalue':chi2.sf(chisq, k -
1)},index=[0])
  return f_res

########## Experiment 1 IR2.3 ##########
dict_result = dict()
dict_best = dict()

#Bagging Hetero
```

```
n_estimators = 20
best_para, bag_ens = bagging_fit(X_train, y_train, n_estimators=n_estimators,
percent_max_samples=0.95)

y_probs, y_pred = bagging_predict(X=X_test, estimators=bag_ens,
n_estimators=n_estimators, vote=False)
dict_result['Bagging Hetero'] = result_(y_test, y_pred, y_probs)
dict_best['Bagging Hetero'] = best_para

#Base model
for i in list_models:
  # Key result, best parameters
  text = i
  # fit and predict
  y_probs, y_pred, best = classifier_selection(X_train, y_train, X_test, y_test,
list_models[i], list_parameters[i])
  #Calculate performance
  dict_result[text] = result_(y_test, y_pred, y_probs)
  dict_best[text] = best

#Dataframe result
df1, df2 = generate_df_result(dict_result)
test_stat = friedman_custom(df2)

########## Experiment 1 IR10 ##########
dict_result10 = dict()
dict_best10 = dict()

n_estimators = 20
best_para, bag_ens = bagging_fit(X_train10, y_train10, n_estimators=n_estimators,
percent_max_samples=0.95)
```

```
y_probs, y_pred = bagging_predict(X = X_test10, estimators = bag_ens, n_estimators
= n_estimators, vote = False)

dict_result10['Bagging Hetero'] = result_(y_test10, y_pred, y_probs)
dict_best10['Bagging Hetero'] = best_para

#Base model
for i in list_models:
  # Key result, best parameters
  text = i
  # fit and predict
  y_probs, y_pred, best = classifier_selection(X_train10, y_train10, X_test10, y_test10,
list_models[i], list_parameters[i])
  #Calculate performance
  dict_result10[text] = result_(y_test10, y_pred, y_probs)
  dict_best10[text] = best

#Dataframe result
df1_10, df2_10 = generate_df_result(dict_result10)
test_stat = friedman_custom(df2_10)

########## Experiment 1 IR14 ##########
dict_result14 = dict()
dict_best14 = dict()

n_estimators = 20
best_para, bag_ens = bagging_fit(X_train14, y_train14, n_estimators=n_estimators,
percent_max_samples=0.95)
y_probs, y_pred = bagging_predict(X = X_test14, estimators = bag_ens, n_estimators
= n_estimators, vote = False)
```

```
dict_result14['Bagging Hetero'] = result_(y_test14, y_pred, y_probs)
dict_best14['Bagging Hetero'] = best_para

#Base model
for i in list_models:
  # Key result, best parameters
  text = i
  # fit and predict
  y_probs, y_pred, best = classifier_selection(X_train14, y_train14, X_test14, y_test14,
list_models[i], list_parameters[i])
  #Calculate performance
  dict_result14[text] = result_(y_test14, y_pred, y_probs)
  dict_best14[text] = best

#Dataframe result
df1_14, df2_14 = generate_df_result(dict_result14)
test_stat = friedman_custom(df2_14)

########## Experiment 2 ##########
#Dimension reduction
class dimension_reduction:

  def __init__(self, X_train, y_train, X_test, y_test, model_name, model, para):
    self.X_train = X_train
    self.y_train = y_train
    self.X_test = X_test
    self.y_test = y_test
    self.model_name = model_name
    self.model = model
    self.para = para
```

```
  def new_para(self, dict_para):
      new_para_ = dict()
      for i, j in self.para.items():
        dummy = dict()
        for k, l in self.para[i].items():
          text = str(i) + '__' + str(k)
          dummy[text] = l
        for n,o in dict_para.items():
          dummy[n] = o
        new_para_[i] = dummy
      return new_para_

  def singular_value_decomposition(self):
      new_para_ex2 = self.new_para({'svd__n_components':[3, 5, 10, 15, 20, 25, 30]})
      pipe = Pipeline(steps=[("svd", TruncatedSVD(random_state=rand[2])),
(self.model_name, self.model)])
      search = RandomizedSearchCV(pipe, new_para_ex2[self.model_name],
scoring='roc_auc', verbose=2, n_jobs=-1, cv=3,  n_iter=50, random_state=rand[0])
      search.fit(self.X_train, self.y_train)
      pre = search.predict(self.X_test)
      pre_prob = search.predict_proba(self.X_test)[:, 1]
      return search.best_params_, self.result_(self.y_test, pre, pre_prob)

  def linear_discriminant_analysis(self):
      new_para_ex2 = self.new_para({'lda__solver':['svd', 'lsqr']})
      pipe = Pipeline(steps=[("lda", LinearDiscriminantAnalysis(n_components=1)),
(self.model_name, self.model)])
      search = RandomizedSearchCV(pipe, new_para_ex2[self.model_name],
scoring='roc_auc', verbose=2, n_jobs=-1, cv=3,  n_iter=50, random_state= rand[0])
      search.fit(self.X_train, self.y_train)
      pre = search.predict(self.X_test)
```

```
        pre_prob = search.predict_proba(self.X_test)[:, 1]
        return search.best_params_, self.result_(self.y_test, pre, pre_prob)
    def result_(self, y_real, y_pre, y_auc):
        return [accuracy_score(y_real, y_pre), roc_auc_score(y_real, y_auc),
                f1_score(y_real, y_pre), precision_score(y_real, y_pre),
                self.kss(y_real, y_auc), brier_score_loss(y_real, y_auc)
                ]

    def kss(self, y_real, y_proba):
        df = pd.DataFrame()
        df['real'] = y_real
        df['proba'] = y_proba
        class1 = df[df['real'] == 1]
        class2 = df[df['real'] == 2]
        ks = ks_2samp(class1['proba'], class2['proba'])[0]
        return ks

index = ['ACC', 'AUC', 'F1', 'PRE', 'KSS', 'BS']
name_model = ['lr', 'knn', 'dt', 'svm']

def Dataframe_result_ex2(dict_ex2, evaluation_metrics, model_name):
    new_list = list(dict_ex2)
    new_index = [item for sublist in [[*range(i, len(dict_ex2), len(index))] for i in
range(len(evaluation_metrics))] for item in sublist]
    number_ = int(len(new_list)/len(model_name))
    new_dict = {'model':[i for j in evaluation_metrics for i in model_name],
                'performance':[j for j in evaluation_metrics for i in model_name],
                'svd':[round(dict_ex2[new_list[i]][j],3) for j in range(len(evaluation_metrics))
for i in [*range(0, len(dict_ex2), number_)]],
                'lda':[round(dict_ex2[new_list[i]][j],3) for j in range(len(evaluation_metrics))
for i in [*range(1, len(dict_ex2), number_)]]
```

```
  }

  df_result = pd.DataFrame(new_dict)
  list_df_rank = []
  list_df_rank2 = []
  for model in name_model:
    df_temp = df_result[df_result['model'] == model]
    df_temp_rank_asc = df_temp[df_temp['performance'] !=
'BS'][df_temp.columns[2:].tolist()].rank(ascending=False, axis=1)
    df_temp_rank_desc = df_temp[df_temp['performance'] ==
'BS'][df_temp.columns[2:].tolist()].rank(ascending=True, axis=1)
    df_temp_rank = pd.concat([df_temp_rank_asc, df_temp_rank_desc])
    df_temp_rank_final = df_temp_rank.mean(axis=0).to_frame().transpose()
    df_temp_rank_final.insert(0,'Model','')
    df_temp_rank_final['Model'] = 'Mean rank ' + model
    df_temp_rank.insert(0,'Model','')
    df_temp_rank.insert(0,'Performnce','')
    df_temp_rank['Performnce'] = df_result[df_result['model'] == model].iloc[:, 1]
    df_temp_rank['Model'] = df_result[df_result['model'] == model].iloc[:, 0]
    list_df_rank.append(df_temp_rank_final)
    list_df_rank2.append(df_temp_rank)
  df_ex_2_all_rank = pd.concat(list_df_rank, axis=0)
  df_ex_2_rank = pd.concat(list_df_rank2, axis=0)
  return df_result, df_ex_2_all_rank, df_ex_2_rank

########## Experiment 2 IR2.3 ##########
dict_ex2_result23 = dict()
dict_ex2_best23 = dict()
#LogisticRegression
lr_dr = dimension_reduction(X_train, y_train, X_test, y_test, 'lr', list_models['lr'],
list_parameters)
```

```
dict_ex2_best23['lr_svd'], dict_ex2_result23['lr_svd'] =
lr_dr.singular_value_decomposition()
dict_ex2_best23['lr_lda'], dict_ex2_result23['lr_lda'] =
lr_dr.linear_discriminant_analysis()

#KNeighborsClassifier
knn_dr = dimension_reduction(X_train, y_train, X_test, y_test, 'knn', list_models['knn'],
list_parameters)
dict_ex2_best23['knn_svd'], dict_ex2_result23['knn_svd'] =
knn_dr.singular_value_decomposition()
dict_ex2_best23['knn_lda'], dict_ex2_result23['knn_lda'] =
knn_dr.linear_discriminant_analysis()

#DecisionTreeClassifier
dt_dr = dimension_reduction(X_train, y_train, X_test, y_test, 'cart', list_models['cart'],
list_parameters)
dict_ex2_best23['dt_svd'], dict_ex2_result23['dt_svd'] =
dt_dr.singular_value_decomposition()
dict_ex2_best23['dt_lda'], dict_ex2_result23['dt_lda'] =
dt_dr.linear_discriminant_analysis()
```



```
#SVC
svc_dr = dimension_reduction(X_train, y_train, X_test, y_test, 'svm',
list_models['svm'],  list_parameters)
dict_ex2_best23['svc_svd'], dict_ex2_result23['svc_svd'] =
svc_dr.singular_value_decomposition()
dict_ex2_best23['svc_lda'], dict_ex2_result23['svc_lda'] =
svc_dr.linear_discriminant_analysis()

#Result
```

```
df_result_ir23_ex2, df_result_ir23_ex2_rank, result =
Dataframe_result_ex2(dict_ex2_result23, index, name_model)
result_wilcoxon = stats.wilcoxon(df_result_ir23_ex2['svd'], df_result_ir23_ex2['lda'])
df_avg_performance23 = result.groupby(['Performnce']).agg({'svd':'mean', 'lda':'mean'})

########## Experiment 2 IR10 ##########
dict_ex2_result10 = dict()
dict_ex2_best10 = dict()

#LogisticRegression
lr_dr = dimension_reduction(X_train10, y_train10, X_test10, y_test10, 'lr',
list_models['lr'],  list_parameters)
dict_ex2_best10['lr_svd'], dict_ex2_result10['lr_svd'] =
lr_dr.singular_value_decomposition()
dict_ex2_best10['lr_lda'], dict_ex2_result10['lr_lda'] =
lr_dr.linear_discriminant_analysis()

#KNeighborsClassifier
knn_dr = dimension_reduction(X_train10, y_train10, X_test10, y_test10, 'knn',
list_models['knn'],  list_parameters)
dict_ex2_best10['knn_svd'], dict_ex2_result10['knn_svd'] =
knn_dr.singular_value_decomposition()
dict_ex2_best10['knn_lda'], dict_ex2_result10['knn_lda'] =
knn_dr.linear_discriminant_analysis()

#DecisionTreeClassifier
dt_dr = dimension_reduction(X_train10, y_train10, X_test10, y_test10, 'cart',
list_models['cart'],  list_parameters)
dict_ex2_best10['dt_svd'], dict_ex2_result10['dt_svd'] =
dt_dr.singular_value_decomposition()
```

```
dict_ex2_best10['dt_lda'], dict_ex2_result10['dt_lda'] =
dt_dr.linear_discriminant_analysis()

#SVC
svc_dr = dimension_reduction(X_train10, y_train10, X_test10, y_test10, 'svm',
list_models['svm'],  list_parameters)
dict_ex2_best10['svc_svd'], dict_ex2_result10['svc_svd'] =
svc_dr.singular_value_decomposition()
dict_ex2_best10['svc_lda'], dict_ex2_result10['svc_lda'] =
svc_dr.linear_discriminant_analysis()

#Result
df_result_ir10_ex2, df_result_ir10_ex2_rank, result10 =
Dataframe_result_ex2(dict_ex2_result10, index, name_model)
result_wilcoxon = stats.wilcoxon(df_result_ir10_ex2['svd'], df_result_ir10_ex2['lda'])
df_avg_performance10 = result10.groupby(['Performnce']).agg({'svd':'mean',
'lda':'mean'})

########## Experiment 2 IR14 ##########
dict_ex2_result14 = dict()
dict_ex2_best14 = dict()

#LogisticRegression
lr_dr = dimension_reduction(X_train14, y_train14, X_test14, y_test14, 'lr',
list_models['lr'],  list_parameters)
dict_ex2_best14['lr_svd'], dict_ex2_result14['lr_svd'] =
lr_dr.singular_value_decomposition()
dict_ex2_best14['lr_lda'], dict_ex2_result14['lr_lda'] =
lr_dr.linear_discriminant_analysis()

#KNeighborsClassifier
```

```
knn_dr = dimension_reduction(X_train14, y_train14, X_test14, y_test14, 'knn',
list_models['knn'],  list_parameters)
dict_ex2_best14['knn_svd'], dict_ex2_result14['knn_svd'] =
knn_dr.singular_value_decomposition()
dict_ex2_best14['knn_lda'], dict_ex2_result14['knn_lda'] =
knn_dr.linear_discriminant_analysis()

#DecisionTreeClassifier
dt_dr = dimension_reduction(X_train14, y_train14, X_test14, y_test14, 'cart',
list_models['cart'],  list_parameters)
dict_ex2_best14['dt_svd'], dict_ex2_result14['dt_svd'] =
dt_dr.singular_value_decomposition()
dict_ex2_best14['dt_lda'], dict_ex2_result14['dt_lda'] =
dt_dr.linear_discriminant_analysis()

#SVC
svc_dr = dimension_reduction(X_train14, y_train14, X_test14, y_test14, 'svm',
list_models['svm'],  list_parameters)
dict_ex2_best14['svc_svd'], dict_ex2_result14['svc_svd'] =
svc_dr.singular_value_decomposition()
dict_ex2_best14['svc_lda'], dict_ex2_result14['svc_lda'] =
svc_dr.linear_discriminant_analysis()

#Result
df_result_ir14_ex2, df_result_ir14_ex2_rank, result14 =
Dataframe_result_ex2(dict_ex2_result14, index, name_model)
result_wilcoxon = stats.wilcoxon(df_result_ir14_ex2['svd'], df_result_ir14_ex2['lda'])
df_avg_performance14 = result14.groupby(['Performnce']).agg({'svd':'mean',
'lda':'mean'})

########## Experiment 3 ##########
```

```
# Class
class resampling:
  def __init__(self, X_train, y_train, X_test, y_test, model_name, model, para,
number_rand=0):
    self.X_train = X_train
    self.y_train = y_train
    self.X_test = X_test
    self.y_test = y_test
    self.model_name = model_name
    self.model = model
    self.para = para
    self.number_rand = number_rand

  def new_para(self, dict_para):
      new_para_ = dict()
      for i, j in self.para.items():
        dummy = dict()
        for k, l in self.para[i].items():
          text = str(i) + '__' + str(k)
          dummy[text] = l
        for n,o in dict_para.items():
          dummy[n] = o
        new_para_[i] = dummy
      return new_para_

  class preprocessing():
    def __init__(self):
      pass
    def fit(self, X, y=None):
      return self
    def transform(self, X):
```

```
        X_round = X.copy()
        col_round = [ i for i in X.columns if X[i].between(0,1).all() == True]
        X_round[col_round] = round(X_round[col_round])
        return X_round


    def oversampling_smote(self):
        new_para_ex2 = self.new_para({'oversample__k_neighbors':np.arange(3, 20, 2),
                                      'oversample__sampling_strategy':('not majority', 'all', 'auto')})
        pipe = Pipeline(steps=[("oversample", SMOTE(random_state=0)), ("pre",
self.preprocessing()),  (self.model_name, self.model)])
        search = RandomizedSearchCV(pipe, new_para_ex2[self.model_name],
scoring='roc_auc', verbose=2, n_jobs=-1, cv=3,  n_iter=50, random_state=0,
refit="roc_auc")
        search.fit(self.X_train, self.y_train)
        pre = search.predict(self.X_test)
        pre_prob = search.predict_proba(self.X_test)[:, 1]
        return search.best_params_, self.result_(self.y_test, pre, pre_prob)


    def undersampling_Random_Under_Sampler(self):
        new_para_ex2 = self.new_para({'RUS__sampling_strategy':('not majority', 'all',
'auto')})
        pipe = Pipeline(steps=[("RUS",
RandomUnderSampler(random_state=self.number_rand)), ("pre", self.preprocessing()),
(self.model_name, self.model)])
        search = RandomizedSearchCV(pipe, new_para_ex2[self.model_name],
scoring='roc_auc', verbose=2, n_jobs=-1, cv=3,  n_iter=50,
random_state=self.number_rand, refit="roc_auc")
        search.fit(self.X_train, self.y_train)
        pre = search.predict(self.X_test)
        pre_prob = search.predict_proba(self.X_test)[:, 1]
        return search.best_params_, self.result_(self.y_test, pre, pre_prob)
```

```
  def oversampling_ADASYN(self):
      new_para_ex2 = self.new_para({'adasyn__n_neighbors':np.arange(3, 20, 2),
                                    'adasyn__sampling_strategy':('not majority', 'all', 'auto')
                                    })
      pipe = Pipeline(steps=[("adasyn", ADASYN(random_state=0)), ("pre",
self.preprocessing()), , (self.model_name, self.model)])
      search = RandomizedSearchCV(pipe, new_para_ex2[self.model_name],
scoring='roc_auc', verbose=2, n_jobs=-1, cv=3,  n_iter=50, random_state=0,
refit="roc_auc")
      search.fit(self.X_train, self.y_train)
      pre = search.predict(self.X_test)
      pre_prob = search.predict_proba(self.X_test)[:, 1]
      return search.best_params_, self.result_(self.y_test, pre, pre_prob)


  def oversampling_BSMOTE(self):
      new_para_ex2 = self.new_para({'BSMOTE__k_neighbors':np.arange(3, 20, 2),
                                    'BSMOTE__sampling_strategy':('not majority', 'not majority', 'all',
'auto'),
                                    'BSMOTE__m_neighbors':np.arange(3, 20, 2)
                                    })
      pipe = Pipeline(steps=[("BSMOTE", BorderlineSMOTE(random_state=0)), ("pre",
self.preprocessing()), (self.model_name, self.model)])
      search = RandomizedSearchCV(pipe, new_para_ex2[self.model_name],
scoring='roc_auc', verbose=2, n_jobs=-1, cv=3,  n_iter=50, random_state=0,
refit="roc_auc")
      search.fit(self.X_train, self.y_train)
      pre = search.predict(self.X_test)
      pre_prob = search.predict_proba(self.X_test)[:, 1]
      return search.best_params_, self.result_(self.y_test, pre, pre_prob)
```

```
  def result_(self, y_real, y_pre, y_auc):
      return [accuracy_score(y_real, y_pre), roc_auc_score(y_real, y_auc),
              f1_score(y_real, y_pre), precision_score(y_real, y_pre),
              self.kss(y_real, y_auc), brier_score_loss(y_real, y_auc)
              ]


  def kss(self, y_real, y_proba):
      df = pd.DataFrame()
      df['real'] = y_real
      df['proba'] = y_proba

      # Recover each class
      class1 = df[df['real'] == 1]
      class2 = df[df['real'] == 2]

      ks = ks_2samp(class1['proba'], class2['proba'])[0]
      return ks


index=['ACC', 'AUC', 'F1', 'PRE', 'KSS', 'BS']
name_model = ['lr', 'knn', 'dt', 'svm']

def Dataframe_result_ex3(dict_ex3, evaluation_metrics, model_name):
  new_list = list(dict_ex3)
  number_ = int(len(new_list)/len(model_name))
  new_dict = {'model':[i for j in evaluation_metrics for i in model_name],
              'performance':[j for j in evaluation_metrics for i in model_name],
              'smote':[round(dict_ex3[new_list[i]][j],3) for j in
range(len(evaluation_metrics)) for i in [*range(0, len(dict_ex3), number_)]],
              'ros':[round(dict_ex3[new_list[i]][j],3) for j in range(len(evaluation_metrics))
for i in [*range(1, len(dict_ex3), number_)]],
```

```
            'adasyn':[round(dict_ex3[new_list[i]][j],3) for j in
range(len(evaluation_metrics)) for i in [*range(2, len(dict_ex3), number_)]],
            'bsmote':[round(dict_ex3[new_list[i]][j],3) for j in
range(len(evaluation_metrics)) for i in [*range(3, len(dict_ex3), number_)]],
  }

  df_result = pd.DataFrame(new_dict)
  list_df_rank = []
  list_df_rank2 = []
  for model in name_model:
    df_temp = df_result[df_result['model'] == model]
    df_temp_rank_asc = df_temp[df_temp['performance'] !=
'BS'][df_temp.columns[2:].tolist()].rank(ascending=False, axis=1)
    df_temp_rank_desc = df_temp[df_temp['performance'] ==
'BS'][df_temp.columns[2:].tolist()].rank(ascending=True, axis=1)
    df_temp_rank = pd.concat([df_temp_rank_asc, df_temp_rank_desc])
    df_temp_rank_final = df_temp_rank.mean(axis=0).to_frame().transpose()
    df_temp_rank_final.insert(0,'Model','')
    df_temp_rank_final['Model'] = 'Mean rank ' + model
    df_temp_rank.insert(0,'Model','')
    df_temp_rank.insert(0,'performance','')
    df_temp_rank['performance'] = df_result[df_result['model'] == model].iloc[:, 1]
    df_temp_rank['Model'] = df_result[df_result['model'] == model].iloc[:, 0]
    list_df_rank.append(df_temp_rank_final)
    list_df_rank2.append(df_temp_rank)
  df_ex_3_all_rank = pd.concat(list_df_rank, axis=0)
  df_ex_3_rank = pd.concat(list_df_rank2, axis=0)
  return df_result, df_ex_3_all_rank, df_ex_3_rank

########## Experiment 3 IR2.3 ##########
dict_ex3_result23 = dict()
```

```
dict_ex3_best23 = dict()

#LogisticRegression
lr_oversampling = resampling(X_train, y_train, X_test, y_test, 'lr', list_models['lr'],
list_parameters, 2)

dict_ex3_best23['lr_smote'], dict_ex3_result23['lr_smote'] =
lr_oversampling.oversampling_smote()
dict_ex3_best23['lr_ros'], dict_ex3_result23['lr_rus'] = lr_oversampling.
undersampling_Random_Under_Sampler()
dict_ex3_best23['lr_adasyn'], dict_ex3_result23['lr_adasyn'] =
lr_oversampling.oversampling_ADASYN()
dict_ex3_best23['lr_bsmote'], dict_ex3_result23['lr_bsmote'] =
lr_oversampling.oversampling_BSMOTE()

#KNeighborsClassifier
knn_oversampling = resampling(X_train, y_train, X_test, y_test, 'knn',
list_models['knn'],  list_parameters, 2)

dict_ex3_best23['knn_smote'], dict_ex3_result23['knn_smote'] =
knn_oversampling.oversampling_smote()
dict_ex3_best23['knn_ros'], dict_ex3_result23['knn_rus'] = knn_oversampling.
undersampling_Random_Under_Sampler()
dict_ex3_best23['knn_adasyn'], dict_ex3_result23['knn_adasyn'] =
knn_oversampling.oversampling_ADASYN()
dict_ex3_best23['knn_bsmote'], dict_ex3_result23['knn_bsmote'] =
knn_oversampling.oversampling_BSMOTE()

#DecisionTreeClassifier
dt_oversampling = resampling(X_train, y_train, X_test, y_test, 'cart', list_models['cart'],
list_parameters, 2)
```

```
dict_ex3_best23['dt_smote'], dict_ex3_result23['dt_smote'] =
dt_oversampling.oversampling_smote()
dict_ex3_best23['dt_ros'], dict_ex3_result23['dt_rus'] = dt_oversampling.
undersampling_Random_Under_Sampler()
dict_ex3_best23['dt_adasyn'], dict_ex3_result23['dt_adasyn'] =
dt_oversampling.oversampling_ADASYN()
dict_ex3_best23['dt_bsmote'], dict_ex3_result23['dt_bsmote'] =
dt_oversampling.oversampling_BSMOTE()

#SVC
svm_oversampling = resampling(X_train, y_train, X_test, y_test, 'svm',
list_models['svm'],  list_parameters, 2)

dict_ex3_best23['svm_smote'], dict_ex3_result23['svm_smote'] =
svm_oversampling.oversampling_smote()
dict_ex3_best23['svm_ros'], dict_ex3_result23['svm_rus'] = svm_oversampling.
undersampling_Random_Under_Sampler()
dict_ex3_best23['svm_adasyn'], dict_ex3_result23['svm_adasyn'] =
svm_oversampling.oversampling_ADASYN()
dict_ex3_best23['svm_bsmote'], dict_ex3_result23['svm_bsmote'] =
svm_oversampling.oversampling_BSMOTE()

#Result
df_result_ir23_ex3, df_result_ir23_ex3_rank, result23 =
Dataframe_result_ex3(dict_ex3_result23, index, name_model)
df_avg_performance23 = result23.groupby(['performance']).agg({'smote':'mean',
'rus':'mean', 'adasyn':'mean', 'bsmote':'mean'})
test_stat = friedman_custom(df_avg_performance23)

########## Experiment 3 IR10 ##########
```

```
dict_ex3_result10 = dict()
dict_ex3_best10 = dict()

#LogisticRegression
lr_oversampling = resampling(X_train10, y_train10, X_test10, y_test10, 'lr',
list_models['lr'],  list_parameters)

dict_ex3_best10['lr_smote'], dict_ex3_result10['lr_smote'] =
lr_oversampling.oversampling_smote()
dict_ex3_best10['lr_ros'], dict_ex3_result10['lr_rus'] = lr_oversampling.
undersampling_Random_Under_Sampler()
dict_ex3_best10['lr_adasyn'], dict_ex3_result10['lr_adasyn'] =
lr_oversampling.oversampling_ADASYN()
dict_ex3_best10['lr_bsmote'], dict_ex3_result10['lr_bsmote'] =
lr_oversampling.oversampling_BSMOTE()

#KNeighborsClassifier
knn_oversampling = resampling(X_train10, y_train10, X_test10, y_test10, 'knn',
list_models['knn'],  list_parameters, 2)

dict_ex3_best10['knn_smote'], dict_ex3_result10['knn_smote'] =
knn_oversampling.oversampling_smote()
dict_ex3_best10['knn_ros'], dict_ex3_result10['knn_rus'] = knn_oversampling.
undersampling_Random_Under_Sampler()
dict_ex3_best10['knn_adasyn'], dict_ex3_result10['knn_adasyn'] =
knn_oversampling.oversampling_ADASYN()
dict_ex3_best10['knn_bsmote'], dict_ex3_result10['knn_bsmote'] =
knn_oversampling.oversampling_BSMOTE()

#DecisionTreeClassifier
```

```
dt_oversampling = resampling(X_train10, y_train10, X_test10, y_test10, 'cart',
list_models['cart'],  list_parameters)

dict_ex3_best10['dt_smote'], dict_ex3_result10['dt_smote'] =
dt_oversampling.oversampling_smote()
dict_ex3_best10['dt_ros'], dict_ex3_result10['dt_rus'] = dt_oversampling.
undersampling_Random_Under_Sampler()
dict_ex3_best10['dt_adasyn'], dict_ex3_result10['dt_adasyn'] =
dt_oversampling.oversampling_ADASYN()
dict_ex3_best10['dt_bsmote'], dict_ex3_result10['dt_bsmote'] =
dt_oversampling.oversampling_BSMOTE()

#SVC
svm_oversampling = resampling(X_train10, y_train10, X_test10, y_test10, 'svm',
list_models['svm'],  list_parameters)

dict_ex3_best10['svm_smote'], dict_ex3_result10['svm_smote'] =
svm_oversampling.oversampling_smote()
dict_ex3_best10['svm_ros'], dict_ex3_result10['svm_rus'] = svm_oversampling.
undersampling_Random_Under_Sampler()
dict_ex3_best10['svm_adasyn'], dict_ex3_result10['svm_adasyn'] =
svm_oversampling.oversampling_ADASYN()
dict_ex3_best10['svm_bsmote'], dict_ex3_result10['svm_bsmote'] =
svm_oversampling.oversampling_BSMOTE()

#Result
df_result_ir10_ex3, df_result_ir10_ex3_rank, result10 =
Dataframe_result_ex3(dict_ex3_result10, index, name_model)
df_avg_performance10 = result10.groupby(['performance']).agg({'smote':'mean',
'rus':'mean', 'adasyn':'mean', 'bsmote':'mean'})
test_stat = friedman_custom(df_avg_performance10)
```

```
########## Experiment 3 IR14 ##########
dict_ex3_result14 = dict()
dict_ex3_best14 = dict()

#LogisticRegression
lr_oversampling = resampling(X_train14, y_train14, X_test14, y_test14, 'lr',
list_models['lr'],  list_parameters)

dict_ex3_best14['lr_smote'], dict_ex3_result14['lr_smote'] =
lr_oversampling.oversampling_smote()
dict_ex3_best14['lr_ros'], dict_ex3_result14['lr_rus'] = lr_oversampling.
undersampling_Random_Under_Sampler()
dict_ex3_best14['lr_adasyn'], dict_ex3_result14['lr_adasyn'] =
lr_oversampling.oversampling_ADASYN()
dict_ex3_best14['lr_bsmote'], dict_ex3_result14['lr_bsmote'] =
lr_oversampling.oversampling_BSMOTE()

#KNeighborsClassifier
knn_oversampling = resampling(X_train14, y_train14, X_test14, y_test14, 'knn',
list_models['knn'],  list_parameters)

dict_ex3_best14['knn_smote'], dict_ex3_result14['knn_smote'] =
knn_oversampling.oversampling_smote()
dict_ex3_best14['knn_ros'], dict_ex3_result14['knn_rus'] = knn_oversampling.
undersampling_Random_Under_Sampler()
dict_ex3_best14['knn_adasyn'], dict_ex3_result14['knn_adasyn'] =
knn_oversampling.oversampling_ADASYN()
dict_ex3_best14['knn_bsmote'], dict_ex3_result14['knn_bsmote'] =
knn_oversampling.oversampling_BSMOTE()
```

```
#DecisionTreeClassifier
dt_oversampling = resampling(X_train14, y_train14, X_test14, y_test14, 'cart',
list_models['cart'],  list_parameters)

dict_ex3_best14['dt_smote'], dict_ex3_result14['dt_smote'] =
dt_oversampling.oversampling_smote()
dict_ex3_best14['dt_ros'], dict_ex3_result14['dt_rus'] = dt_oversampling.
undersampling_Random_Under_Sampler()
dict_ex3_best14['dt_adasyn'], dict_ex3_result14['dt_adasyn'] =
dt_oversampling.oversampling_ADASYN()
dict_ex3_best14['dt_bsmote'], dict_ex3_result14['dt_bsmote'] =
dt_oversampling.oversampling_BSMOTE()

#SVC
svm_oversampling = resampling(X_train14, y_train14, X_test14, y_test14, 'svm',
list_models['svm'],  list_parameters)

dict_ex3_best14['svm_smote'], dict_ex3_result14['svm_smote'] =
svm_oversampling.oversampling_smote()
dict_ex3_best14['svm_ros'], dict_ex3_result14['svm_rus'] = svm_oversampling.
undersampling_Random_Under_Sampler()
dict_ex3_best14['svm_adasyn'], dict_ex3_result14['svm_adasyn'] =
svm_oversampling.oversampling_ADASYN()
dict_ex3_best14['svm_bsmote'], dict_ex3_result14['svm_bsmote'] =
svm_oversampling.oversampling_BSMOTE()

#Result
df_result_ir14_ex3, df_result_ir14_ex3_rank, result14 =
Dataframe_result_ex3(dict_ex3_result14, index, name_model)
df_avg_performance14 = result14.groupby(['performance']).agg({'smote':'mean',
'rus':'mean', 'adasyn':'mean', 'bsmote':'mean'})
```

```
test_stat = friedman_custom(df_avg_performance14)

########## Experiment 4 ##########
#Function
def get_dimension():
  dimension = dict()
  dimension['svd'] = TruncatedSVD(random_state=0)
  dimension['lda'] = LinearDiscriminantAnalysis(n_components = 1)
  return dimension

def get_parameters_dimension():
  parameters = dict()
  parameters['svd'] = {'n_components':np.array([2, 3, 4, 5])}
  parameters['lda'] = {'solver':['svd', 'lsqr']}
  return parameters

def get_resampling():
  resampling = dict()
  resampling['smote'] = SMOTE(random_state=0)
  resampling['rus'] = RandomUnderSampler(random_state=0)
  resampling['adasyn'] = ADASYN(random_state=0)
  resampling['bsmote'] = BorderlineSMOTE(random_state=0)
  return resampling

def get_parameters_resampling():
  parameters = dict()
  parameters['smote'] = {'k_neighbors':np.arange(3, 20, 2),
                         'sampling_strategy':('all', 'auto', 'not minority')}
  parameters['rus'] = {'sampling_strategy':('all', 'auto', 'not minority')}
  parameters['adasyn'] = {'n_neighbors':np.arange(3, 20, 2),
                          'sampling_strategy':('all', 'auto', 'not minority')}
```

```
  parameters['bsmote'] = {'k_neighbors':np.arange(3, 20, 2),
                          'sampling_strategy':('all', 'auto', 'not minority'),
                          'm_neighbors':np.arange(3, 20, 2)}
  return parameters

def new_append_parameter_final(list_parameters=list_parameters, dict_para1={},
dict_para2={}):
    new_parameter = dict()
    for i,j in list_parameters.items():
      for k,l in j.items():
        text = "bbc__" + str(i) + '__' + str(k)
        new_parameter[text] = l
      for n,o in dict_para1.items():
        new_parameter[n] = o
      for n,o in dict_para2.items():
        new_parameter[n] = o
    return new_parameter

def new_dict_homo(parameters, append_param1, append_param2):
  new_dict = dict()
  for i,j in parameters.items():
    new_dict['bbc__' + str(i)] = j
  new_dict = {**new_dict, **append_param1,**append_param2}
  return new_dict

  class preprocessing():
    def __init__(self):
      pass
    def fit(self, X, y=None):
      return self
    def transform(self, X):
```

```
    X_round = X.copy()
    col_round = [ i for i in X.columns if X[i].between(0,1).all() == True]
    X_round[col_round] = round(X_round[col_round])
    return X_round


def bagging_fit_final(X, y, n_estimators, percent_max_samples=0.8,
list_models=list_models_bagging, list_parameters=list_parameters_bagging,
reduction="", resampling="",
                      para_reduction = {}, para_resampling = {}):

  new_parameter = new_append_parameter_final(list_parameters, para_reduction,
para_resampling)
  scoring = 'roc_auc'
  n_examples = len(y)
  estimators = [ VotingClassifier(
                estimators = [
                    ('lr', list_models['lr']),
                    ('knn', list_models['knn']),
                    ('cart', list_models['cart']),
                    ('svm', list_models['svm'])
                ],
                voting='soft')
            for _ in range(n_estimators)]
  search_estimators = list()
  best_parameter = list()
  for count_number, voting in enumerate(estimators):
    bag = np.random.choice(n_examples,
round(n_examples*percent_max_samples), replace=False)
    pipe = Pipeline(steps=[("rs", resampling), ("pre" , preprocessing()), ("dr",
reduction), ('bbc', voting)])
```

```
        clf = RandomizedSearchCV(pipe, new_parameter, scoring=scoring, verbose=2,
n_jobs=-1, cv=3,  n_iter=500, random_state=rand[count_number])
        clf.fit(X.iloc[bag].values, y.iloc[bag])
        search_estimators.append(clf)
        best_parameter.append(clf.best_params_)
        print("{}/{}".format(count_number+1, n_estimators))

    return best_parameter, search_estimators

def classifier_selection_final(X_train_, y_train_, X_test_, y_test_, model, parameters,
reduction, resampling, para_reduction, para_resampling, number=0):
  param = new_dict_homo(parameters, para_reduction, para_resampling)
  scoring = 'roc_auc'
  pipe = Pipeline(steps=[("rs", resampling), ("pre" , preprocessing()), ("dr", reduction),
('bbc', model)])
  clf = RandomizedSearchCV(pipe, param, scoring=scoring, verbose=2, n_jobs=-1,
cv=3,  n_iter=50, random_state=rand[number])
  clf.fit(X_train_, y_train_)
  y_probs = clf.predict_proba(X_test_)[:, 1]
  y_pred = clf.predict(X_test_)
  return y_probs, y_pred, clf.best_params_

all_dimension = get_dimension()
all_parameter_dimension = get_parameters_dimension()
all_resampling = get_resampling()
all_parameter_resampling = get_parameters_resampling()

########## Experiment 4 IR2.3 ##########
dict_result_ir23_ex4 = dict()
dict_best_ir23_ex4 = dict()
best_ex2 = "lda"
```

```
best_ex3 = "smote"
reduction= all_dimension[best_ex2]
resampling = all_resampling[best_ex3]
para_reduction = { "dr__" + str(k): v for k, v in
all_parameter_dimension[best_ex2].items() }
para_resampling = { "rs__" + str(k): v for k, v in
all_parameter_resampling[best_ex3].items() }

n_estimators = 20

best_para_final, bag_ens_final = bagging_fit_final( X_train, y_train,
n_estimators=n_estimators, percent_max_samples=0.95, reduction=reduction,
resampling=resampling, para_reduction=para_reduction,
para_resampling=para_resampling )
y_prob, y_pred = bagging_predict(X_test, bag_ens_final, n_estimators, False)
dict_result_ir23_ex4['Bagging Hetero'] = result_(y_test, y_pred, y_prob)
dict_best_ir23_ex4['Bagging Hetero'] = best_para_final

#Base model
for i in list_models:
  text = str(i)
  y_prob, y_pred, best = classifier_selection_final(X_train, y_train, X_test, y_test,
list_models[i], list_parameters[i], reduction, resampling, para_reduction,
para_resampling)
  dict_result_ir23_ex4[text] = result_(y_test, y_pred, y_prob)
  dict_best_ir23_ex4[text] = best

#Result
df1_final, df2_final = generate_df_result(dict_result_ir23_ex4)
test_stat = friedman_custom(df2_final)
```

```
########## Experiment 4 IR10 ##########
dict_result_ir10_ex4 = dict()
dict_best_ir10_ex4 = dict()
best_ex2 = "lda"
best_ex3 = "rus"
reduction= all_dimension[best_ex2]
resampling = all_resampling[best_ex3]
para_reduction = { "dr__" + str(k): v for k, v in
all_parameter_dimension[best_ex2].items() }
para_resampling = { "rs__" + str(k): v for k, v in
all_parameter_resampling[best_ex3].items() }

n_estimators = 20

best_para_final, bag_ens_final = bagging_fit_final(X_train10, y_train10,
n_estimators=n_estimators, percent_max_samples=0.95, reduction=reduction,
resampling=resampling, para_reduction=para_reduction,
para_resampling=para_resampling )
y_prob, y_pred = bagging_predict(X_test10, bag_ens_final, n_estimators, False)
dict_result_ir10_ex4['Bagging Hetero'] = result_(y_test10, y_pred, y_prob)
dict_best_ir10_ex4['Bagging Hetero'] = best_para_final

#Base model
for i in list_models:
  text = str(i)
  y_prob, y_pred, best = classifier_selection_final(X_train10, y_train10, X_test10,
y_test10, list_models[i], list_parameters[i], reduction, resampling, para_reduction,
para_resampling)
  dict_result_ir10_ex4[text] = result_(y_test10, y_pred, y_prob)
  dict_best_ir10_ex4[text] = best
```

```
#Result
df1_ir10_final, df2_ir10_final = generate_df_result(dict_result_ir10_ex4)
test_stat = friedman_custom(df2_ir10_final)

########## Experiment 4 IR14 ##########
dict_result_ir14_ex4 = dict()
dict_best_ir14_ex4 = dict()
best_ex2 = "lda"
best_ex3 = "bsmote"
reduction= all_dimension[best_ex2]
resampling = all_resampling[best_ex3]
para_reduction = { "dr__" + str(k): v for k, v in
all_parameter_dimension[best_ex2].items() }
para_resampling = { "rs__" + str(k): v for k, v in
all_parameter_resampling[best_ex3].items() }

n_estimators = 20

best_para_final, bag_ens_final = bagging_fit_final(X_train14, y_train14,
n_estimators=n_estimators, percent_max_samples=0.95, reduction=reduction,
resampling=resampling, para_reduction=para_reduction,
para_resampling=para_resampling )
y_prob, y_pred = bagging_predict(X_test14, bag_ens_final, n_estimators, False)
dict_result_ir14_ex4['Bagging Hetero'] = result_(y_test14, y_pred, y_prob)
dict_best_ir14_ex4['Bagging Hetero'] = best_para_final

#Base model
for i in list_models:
  text = str(i)
```

```
  y_prob, y_pred, best = classifier_selection_final(X_train14, y_train14, X_test14,
y_test14, list_models[i], list_parameters[i], reduction, resampling, para_reduction,
para_resampling)
  dict_result_ir14_ex4[text] = result_(y_test14, y_pred, y_prob)
  dict_best_ir14_ex4[text] = best

#Result
df1_ir14_final, df2_ir14_final = generate_df_result(dict_result_ir14_ex4)
test_stat = friedman_custom(df2_ir14_final)
```

ประวัติผู้เขียน

ชื่อ-สกุล	นางสาว ศศิวิมล ศรีโรจน์
วัน เดือน ปี เกิด	1 พฤษภาคม 2541
สถานที่เกิด	กรุงเทพมหานคร ประเทศไทย
ที่อยู่ปัจจุบัน	24/168 หมู่บ้านมโนรมย์ 5 หทัยราษฎร์ 39 แขวงสามวาตะวันตก เขตคลองสามวา กรุงเทพมหานคร 10510