# การทำนายความยากจนจากข้อมูลสำมะโนครัวประชากรด้วยเทคนิคการเรียนรู้ของเครื่องจักร

# Predicting Poverty from Census Data Using Machine Learning Techniques


นางสาวภัทรนันท์ ทุนทวีทรัพย์

Pattaranan Tuntaweesap

นางสาวรุจิรัตน์ สุธิมนต์รัตน์

Rujirat Sutimonrat


**โครงงานนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิทยาศาสตรบัณฑิต**

**ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์**

**มหาวิทยาลัยศรีนครินทรวิโรฒ ปีการศึกษา พ.ศ. 2561**

**คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ**

**ชื่อหัวข้อโครงงาน** การทำนายความยากจนจากข้อมูลสำมะโนครัวประชากรด้วยเทคนิคการเรียนรู้ของเครื่องจักร

Predicting Poverty from Census Data Using Machine Learning Techniques


**นิสิต** นางสาวภัทรนันท์ ทุนทวีทรัพย์ 58102010820

นางสาวรุจิรัตน์ สุธิมนต์รัตน์ 58102010823


**ปริญญา** วิทยาศาสตรบัณฑิต (วท.บ.)

**ภาควิชา** วิทยาการคอมพิวเตอร์

**อาจารย์ที่ที่ปรึกษาโครงงาน** ผู้ช่วยศาสตราจารย์ด็อกเตอร์จันตรี ผลประเสริฐ

ลงชื่อ.....

(ผู้ช่วยศาสตราจารย์ด็อกเตอร์จันตรี ผลประเสริฐ)

อาจารย์ที่ปรึกษาโครงงาน

# บทคัดย่อ

งานวิจัยนี้นำเสนอวิธีการเรียนรู้ของเครื่องสำหรับการทำนายระดับความยากจนในครัวเรือนโดยอาศัยการผสมผสานระหว่างกระบวนการปรับแต่งคุณลักษณะเฉพาะของข้อมูล (Feature Engineering) , เทคนิคการสุ่มเพิ่มตัวอย่างกลุ่มน้อย (Synthetic Minority Over-sampling Technique) และการใช้โมเดลการเรียนรู้ของเครื่องจักรแบบต้นไม้ในการวิเคราะห์ข้อมูลสำมะโนประชากร

งานวิจัยนี้ทดสอบประสิทธิภาพของแบบจำลองที่นำเสนอผ่านข้อมูลสำมะโนประชากรจากประเทศคอสตาริกา โดยนำเสนอการใช้กระบวนการปรับแต่งคุณลักษณะเฉพาะของข้อมูลเพื่อสร้างคุณลักษณะเฉพาะประจำครัวเรือน โดยแก้ปัญหาความไม่สมดุลของข้อมูลสำมะโนประชากรโดยใช้วิธีการสุ่มเพิ่มตัวอย่างแบบกลุ่มน้อย จากนั้นทำการปรับไฮเปอร์พารามิเตอร์ (hyperparameter) ของแต่ละแบบจำลองเพื่อเพิ่มประสิทธิภาพของโมเดลในการทำนายความยากจน จากผลการทดลองพบว่าโมเดลการเรียนรู้ของเครื่องจักรแบบต้นไม้ชนิด Gradient Boosting มีประสิทธิภาพดีที่สุดในการทำนายความยากจนในระดับครัวเรือนโดยให้ค่าความถูกต้อง (Accuracy) เท่ากับ 67.6% ความแม่นยำ (Precision) เท่ากับ 0.46 และคะแนนมาโคร F1 (macro F1) เฉลี่ยเท่ากับ 0.43 ซึ่งมีค่าสูงกว่าโมเดลมาตรฐาน (baseline) ซึ่งใช้โมเดลแบบป่าสุ่ม (Random Forest) ซึ่งได้คะแนนมาโคร F1 เฉลี่ยเท่ากับ 0.32 โดยโมเดลที่นำเสนอนี้ระบุคุณสมบัติที่สำคัญที่สุดสามประการที่มีผลต่อประสิทธิภาพของแบบจำลองอันประกอบไปด้วย จำนวนปีในสถานศึกษา คุณภาพของหลังคา และอัตราส่วนของสมาชิกในบ้านที่ต้องพึ่งพาต่อสมาชิกในบ้านที่ต้องทำงานที่มีค่าความสำคัญ (feature importance) เท่ากับ 0.0286, 0.0267 และ 0.0206 ตามลำดับ จากการทดลองพบว่าเทคนิคการสุ่มเพิ่มตัวอย่างกลุ่มน้อยมีส่วนช่วยในการเพิ่มประสิทธิภาพของโมเดลในการระบุความยากจน การทำงานในอนาคตจะมุ่งเน้นไปที่การปรับไฮเปอร์พารามิเตอร์เพื่อปรับปรุงประสิทธิภาพของโมเดล

## Abstract

In this paper, we propose a machine learning method for household poverty level prediction based on a combination of feature engineering, Synthetic Minority Over-sampling Technique (SMOTE) and tree-based classification models.

We test the performance of our proposed system over census data from Costa Rica. A feature engineering method is applied to construct a group feature of the household. Then, after applying SMOTE, hyperparameter tuning of each classification model is implemented. Our proposed model employing gradient boosting classifier yields the best accuracy equal to 67.6% with precision equal to 0.46 and macro F1 score equal to 0.43. Three most important features contributing to the performance of the proposed model are roof quality of the house, years of education and Ratio of non-working age to working age whose values are equal to 0.0286, 0.0267 and 0.0206, respectively. Our performance is superior to the baseline model using random forest whose F1 score is equal to 0.32. Performance enhancement is partly due to SMOTE which randomly resamples minority data classes to improve the classification performance of the proposed model. Future work will focus on hyperparameter tuning to improve the performance of the model.

# กิติกรรมประกาศ

การทำนายความยากจนจากข้อมูลสำมะโนครัวประชากรด้วยเทคนิคการเรียนรู้ของเครื่องจักร (Predicting Poverty from Census Data Using Machine Learning Techniques) ผู้วิจัยได้มุ่งมั่นศึกษาค้นคว้าอย่างต่อเนื่องจนได้รับความรู้และความเข้าใจในเรื่องที่ศึกษาจนสามารถทำนายความยากจนจากข้อมูลสำมะโนครัวประชากรได้อย่างมีประสิทธิภาพ โครงงานนี้สามารถสำเร็จลุล่วงเพราะได้รับความเมตตาและความช่วยเหลือจากบุคคลหลายๆท่าน คณะผู้จัดทำขอขอบพระคุณบุคคลทุกท่านที่ให้ความรู้คำแนะนำ แนวทางแก้ไขปัญหา และให้ความอนุเคราะห์เพื่อการศึกษาโครงงานนี้

ขอขอบพระคุณ ผศ.ดร.จันตรี ผลประเสริฐ อาจารย์ในที่ปรึกษาโครงงานผู้ซึ่งให้คำแนะนำคำปรึกษา ความใส่ใจและความช่วยเหลืออย่างเต็มที่รวมไปถึงการตรวจสอบแก้ไข ตลอดจนให้คำปรึกษาในทุกๆรายละเอียดของขั้นตอนการทำงานจนโครงงานบรรลุวัตถุประสงค์ตามเป้าหมายที่ตั้งไว้

ขอขอบคุณ ดร.สุทธิพงศ์ ธัชยพงษ์ หัวหน้าทีมวิจัยการวิเคราะห์ยุทธศาสตร์ด้วยปัญญาประดิษฐ์ (SAI) ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (เนคเทค) ที่ให้คำชี้แนะเกี่ยวกับแนวทางการใช้เทคโนโลยีการเรียนรู้ของเครื่องจักรในการวิเคราะห์ผู้ยากจน

คณะผู้จัดทำ

# สารบัญ

# สารบัญรูป

# สารบัญตาราง

# บทที่ 1
# บทนำ

## 1.1 ที่มาและความสำคัญของปัญหา

ความยากจนเป็นหนึ่งในปัจจัยสำคัญที่ส่งผลต่อสุขภาพและการเจริญเติมโตของประชากรซึ่งเป็นสาเหตุหลักของความไม่มั่นคงทางสังคมและเป็นหนึ่งในสาเหตุที่ทำให้สูญเสียศักยภาพของมนุษย์ การขจัดความยากจนยังคงเป็นความท้าทายที่สำคัญและเป็นเป้าหมายหลักของการพัฒนาประเทศอย่างยั่งยืน แนวทางหนึ่งที่ภาครัฐใช้ในการแก้ปัญหาความยากจนคือการพยายามที่จะระบุจำนวนผู้ยากจนที่แน่นอนรวมไปถึงปัจจัยที่ส่งผลต่อความยากจนเหล่านั้นเพื่อจะได้หามาตรการช่วยเหลือได้อย่างตรงจุดและมีประสิทธิภาพ โดยมีงานวิจัยที่พยายามระบุผู้ยากจนและปัจจัยที่ส่งผลต่อความยากจนโดยการใช้ข้อมูลหลายๆประเภท เช่น ข้อมูลการใช้โทรศัพท์ [1], ข้อมูล remote sensing [2], และข้อมูลสำมะโนประชากร [3] โดยงานวิจัยนี้สนใจการระบุความยากจนจากข้อมูลสำมะโนประชากร เนื่องจากเป็นข้อมูลที่มีความละเอียด มีหลายปัจจัย อย่างไรก็ตามการระบุความยากจนจากข้อมูลสำมะโนประชากรยังมีความไม่แม่นยำเนื่องจากสาเหตุหลายประการ อาทิ เช่น ข้อมูลไม่ครบ ข้อมูลมีความผิดพลาด และข้อมูลมีความหลากหลายเนื่องจากข้อมูลที่เก็บมานั้นมีหลายประเภท อาทิ เช่น ข้อมูลที่อยู่อาศัย ข้อมูลสุขอนามัย ข้อมูลการศึกษา รายได้และอื่นๆ ทำให้การวิเคราะห์ทำได้ยาก มีความไม่แน่นอนสูง

ในปัจจุบันเทคโนโลยีการเรียนรู้ของเครื่อง (Machine learning) กำลังเป็นที่นิยมและแพร่หลาย โดยเทคโนโลยีการเรียนรู้ของเครื่องสามารถหาความสัมพันธ์ของข้อมูลที่มีปริมาณมากและมีความซับซ้อนได้ถ้ามีข้อมูลที่มีประโยชน์และมีจำนวนเพียงพออย่างไรก็ตามมีงานวิจัย (reference) ที่นำเทคโนโลยีการเรียนรู้ของเครื่องมาระบุความยากจนโดยการวิเคราะห์ข้อมูลสำมะโนประชากร ซึ่งยังไม่แม่นยำเนื่องมาจากสาเหตุหลายๆประการ อาทิ เช่น การไม่สมดุลกันของข้อมูล (Imbalance dataset) ข้อมูลไม่ครบเป็นต้น

ในงานวิจัยนี้สนใจที่จะนำเทคโนโลยีการเรียนรู้ของเครื่องแบบต้นไม้มาใช้ในการวิเคราะห์ข้อมูลสำมะโนประชากรเพื่อระบุความยากจนในระดับครัวเรือน โดยนำเสนอการใช้กระบวนการปรับแต่งคุณลักษณะเฉพาะของข้อมูลเพื่อสร้างคุณลักษณะเฉพาะประจำครัวเรือน โดยแก้ปัญหาความไม่สมดุลของข้อมูลสำมะโนประชากรโดยใช้วิธีการการสุ่มเพิ่มตัวอย่างแบบกลุ่มน้อย (Synthetic Minority Oversampling Technique, SMOTE) จากนั้นทำการปรับไฮเปอร์พารามิเตอร์ (hyperparameter) ของแต่ละแบบจำลองแบบต้นไม้เพื่อเพิ่มประสิทธิภาพของโมเดลในการทำนายความยากจน หลังจากทำการปรับไฮเปอร์พารามีเตอร์แล้วจะนำโมเดลที่ได้ไปทดสอบประสิทธิภาพกับชุดข้อมูล test dataset เพื่อวัดประสิทธิภาพของโมเดลในรูปของ confusion matrix และ macro f1-score ต่อไป

**1.2 วัตถุประสงค์ของงานวิจัย**

1. เพื่อศึกษาเทคนิคการเรียนรู้ของเครื่องในการวิเคราะห์ข้อมูลสำมะโนครัวประชากร เพื่อระบุความยากจนในระดับครัวเรือน
2. การใช้เทคนิคการเรียนรู้ของเครื่องในการวิเคราะห์หาปัจจัยสำคัญที่ส่งผลต่อความยากจน
3. เพื่อศึกษาการใช้วิธีการสุ่มเพิ่มตัวอย่างกลุ่มน้อยในการแก้ปัญหาความไม่สมดุลกันของข้อมูล

**1.3 ขอบเขตของโครงงาน**

1. ใช้ dataset ใน Kaggle Competition ซึ่งเป็นข้อมูลสำมโนครัวประชากรของประเทศคอสตาริกา [3]
2. ใช้โมเดลเทคโนโลยีการเรียนรู้ของเครื่องแบบต้นไม้ในการระบุความยากจน เช่น Random Forest, gradient booting, Adaboost เป็นต้น
3. ใช้วิธีการสุ่มเพิ่มตัวอย่างกลุ่มน้อยในการแก้ปัญหาความไม่สมดุลกันของข้อมูล
4. ใช้ค่า Macro F1 Score และ Confusion Matrix ในการวัดประสิทธิภาพของโมเดล

**1.4 ประโยชน์ที่คาดว่าจะได้รับ**

1. เรียนรู้ Data science process
2. ได้ทดสอบการใช้เทคนิคการเรียนรู้ของเครื่องจักร ในการวิเคราะห์ความความยากจน
3. เพื่อหาโมเดลเพื่อใช้ทำนายความยากจนจากข้อมูลประชากร
4. สามารถทำนายความยากจนได้อย่างแม่นยำเพื่อให้รัฐบาลนำไปช่วยแก้ไขปัญหาความยากจนได้อย่างมีประสิทธิภาพ

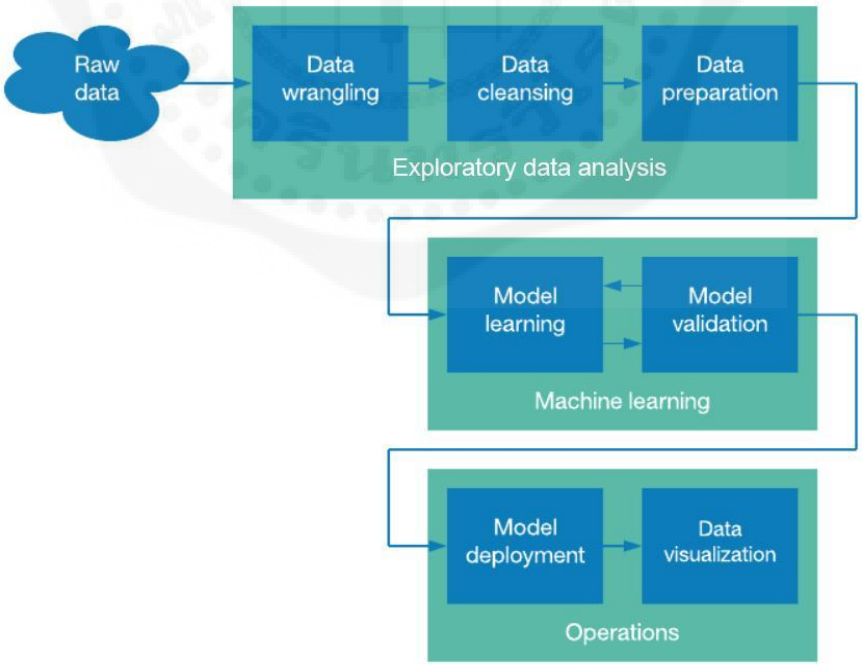
# บทที่ 2
# องค์ความรู้ที่เกี่ยวข้อง

ในบทนี้เริ่มต้นโดยกล่าวถึงองค์ความรู้เกี่ยวกับ Data Science จากนั้นจะอธิบายในส่วนขององค์ความรู้เกี่ยวกับ Machine Learning องค์ความรู้เกี่ยวกับข้อมูลที่ไม่สมดุล (Imbalanced Datasets) และองค์ความรู้เกี่ยวกับ Synthetic Minority Over-samplingTechnique (SMOTE) จากนั้นจะกล่าวถึงการวัดประสิทธิภาพของโมเดล และงานวิจัยที่ใช้ในการศึกษา

## 2.1 องค์ความรู้เกี่ยวกับ Data Science [4]

Data Science หมายถึง ศาสตร์การศึกษาวิเคราะห์หาความสัมพันธ์ของข้อมูล เพื่อนำมาใช้ให้เกิดประโยชน์ เช่น ช่วยในการตัดสินใจ ทำนายหรือคาดคะเน เป็นต้น กระบวนการทาง Data Science สามารถอธิบายได้ดังรูปที่1 โดยจากรูปจะเห็นได้ว่ากระบวนการ Data Science สามารถแบ่งออกได้เป็นขั้นตอนหลัก ทั้งหมด 3 ขั้นตอน

1. Exploratory data analysis (EDA)
2. Machine learning
3. Operations




รูปที่ 1 : The data science pipeline

(ที่มา : https://www.ibm.com/developerworks/library/ba-intro-data-science-1/index.html,2018)

**2.1.1. Exploratory data analysis (EDA) [5]**

Exploratory data analysis (EDA) เป็นวิธีการวิเคราะห์ชุดข้อมูลเพื่อสรุปลักษณะสำคัญของข้อมูล เป็นกระบวนการที่สำคัญในการดำเนินการตรวจสอบข้อมูลเบื้องต้นเพื่อหารูปแบบ ค้นหาจุดผิดปกติ ทดสอบสมมติฐานและเพื่อตรวจสอบสมมติฐานด้วยสถิติสรุปและการใช้ภาพกราฟิก เช่นงาน Predictive ลักษณะการทำงานคือ สำรวจข้อมูลในมุมต่างๆ ในทุกๆตัวแปร หรือเปรียบเทียบกันระหว่างตัวแปร วิธีการทำ EDA ก็มีหลากหลายวิธี เช่น

การตรวจสอบค่าสูญหาย (Missing Values)

ข้อมูลสูญหาย คือ ค่าสังเกตหรือค่าของตัวแปรที่ต้องการทราบค่าแต่ไม่สามารถทราบค่าได้ อย่างไรก็ตามมีวิธีการทำให้ข้อมูลที่ถูกลดจำนวนลงสามารถทำการวิเคราะห์ได้ เช่น การวัดตัวแปร 5 ตัว จากการเก็บข้อมูลทุกๆเดือนเป็นเวลา 6 เดือน พบว่า แต่ละตัวแปรมีค่าขาดหายแบบสุ่ม 5% คาดว่า จำนวนของข้อมูลที่สมบูรณ์น่าจะมีประมาณ 17% วิธีการแก้ไขคือ การแทนค่า (imputation) ค่าที่ขาดหาย

การตรวจสอบค่าผิดปกติ (Outlier/Extreme)

ข้อมูลที่มีค่าแตกต่างกันทั้งมากกว่าและน้อยกว่าจากข้อมูลในชุดเดียวกัน วิธีการแก้ไขคือ ตัดข้อมูลออกจากการวิเคราะห์ (Drop the case) หรือ วิเคราะห์โดยมีค่าผิดปกติ, วิเคราะห์โดยไม่มีค่าผิดปกติ

Data wrangling

ขั้นตอนการจัดการข้อมูลดิบเพื่อให้มีประโยชน์สำหรับการวิเคราะห์ข้อมูลหรือเพื่อฝึกแบบจำลองการเรียนรู้ของเครื่อง ส่วนนี้ของ EDA อาจรวมถึงการจัดหาข้อมูลจากชุดข้อมูลอย่างน้อยหนึ่งชุด การทำให้ข้อมูลเป็นบรรทัดฐานเพื่อให้ข้อมูลรวมจากชุดข้อมูลหลายชุดมีความสอดคล้องกันและการแยกวิเคราะห์ข้อมูลในโครงสร้างบางส่วนหรือ เก็บข้อมูลเพื่อใช้งานต่อไป พิจารณาชุดข้อมูลสาธารณะจากเว็บไซต์ข้อมูลแห่งชาติที่เปิดอยู่ ข้อมูลนี้อาจมีอยู่เป็นไฟล์ spreadsheet ที่คุณต้องส่งออกเป็นรูปแบบที่ยอมรับได้มากขึ้นในภาษา data science (CSV หรือ JavaScript Object Notation) สุดท้ายข้อมูลอาจมาจากหลายแหล่งซึ่งต้องการให้คุณเลือกรูปแบบทั่วไปสำหรับชุดข้อมูลที่เป็นผลลัพธ์

Data cleansing

หลังจากที่รวบรวมชุดข้อมูล ขั้นตอนถัดไปคือ Data cleansing ชุดข้อมูลมักมีปัญหา คือปัญหาทั่วไปรวมทั้งค่าที่หายไป (หรือค่ามากเกินไป) ตัวคั่นที่ไม่ถูกต้องหรือไม่ดี (ซึ่งแยกข้อมูลออก) บันทึกที่ไม่สอดคล้องกันหรือพารามิเตอร์ไม่เพียงพอ ในบางกรณีข้อมูลไม่สามารถซ่อมแซมได้และต้องลบออก ในกรณีอื่น ๆ สามารถแก้ไขด้วยตนเองหรือโดยอัตโนมัติ เมื่อชุดข้อมูลถูกต้องแล้ว ขั้นตอนต่อไปคือเพื่อให้แน่ใจว่ามีความถูกต้องตามหลักไวยากรณ์ ในชุดข้อมูลที่มีข้อมูลตัวเลข จะมีข้อผิดพลาดที่ต้องได้รับการตรวจสอบอย่างรอบคอบ คุณสามารถค้นพบค่าผิดปกติเหล่านี้ได้จากการวิเคราะห์ทางสถิติโดยดูที่ค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐาน การค้นหาค่าผิดปกติเป็นวิธีที่สองในการData cleansing เพื่อให้มั่นใจว่าข้อมูลมีความสม่ำเสมอและถูกต้อง

Data preparation

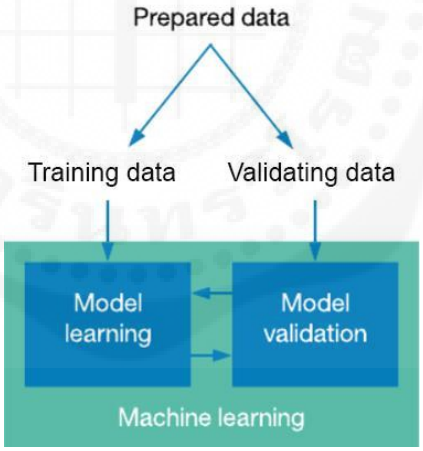
ขั้นตอนสุดท้ายในด้าน EDA คือ Data preparation (หรือการประมวลผลล่วงหน้า) ขั้นตอนนี้อนุมานว่ามีชุดข้อมูลไม่พร้อมสำหรับการประมวลผลโดยอัลกอริทึมการเรียนรู้ด้วยเครื่อง เช่นในบางกรณีการทำให้เป็นบรรทัดฐานของข้อมูลจะมีประโยชน์ เมื่อใช้ normalization คุณจะแปลงคุณลักษณะอินพุตเพื่อแจกจ่ายข้อมูลให้เท่ากันในช่วงที่ยอมรับได้สำหรับอัลกอริทึมการเรียนรู้ของเครื่อง

### 2.1.2. Machine learning [4]

Machine learning คือขั้นตอนหนึ่งที่ทำให้คอมพิวเตอร์มีความสามารถในการเรียนรู้ด้วยตนเองเมื่อมีข้อมูลเข้าสามารถทำนายหรือตัดสินใจได้โดยปราศจากการทำงานตามลำดับคำสั่งโปรแกรม

Model learning ขั้นตอนการประมวลผลข้อมูล ในรูปแบบหนึ่งอัลกอริทึมสามารถประมวลผลข้อมูลได้ด้วยผลิตภัณฑ์ข้อมูลใหม่อันเป็นผล แต่ในแง่การผลิตรูปแบบการเรียนรู้ของเครื่องคือผลิตภัณฑ์ที่นำมาใช้เพื่อให้ข้อมูลเชิงลึกหรือเพิ่มมูลค่า (เช่นให้สามารถทำนายตลาดประกันภัยได้)

Model validation วิธีการทั่วไปในการตรวจสอบแบบจำลองคือการสำรองข้อมูลการ train จำนวนเล็กน้อยที่จะทดสอบกับโมเดลสุดท้าย คุณใช้ข้อมูลการ train เพื่อฝึกแบบจำลองการเรียนรู้ของเครื่องและข้อมูลการทดสอบจะถูกใช้เมื่อโมเดลสมบูรณ์เพื่อตรวจสอบความถูกต้องของข้อมูลที่มองไม่เห็น ดังรูปที่ 2




รูปที่ 2 : ข้อมูลการ train เทียบกับข้อมูลการทดสอบสำหรับการตรวจสอบแบบจำลอง

(ที่มา : https://www.ibm.com/developerworks/library/ba-intro-data-science-1/index.html,2018)

**2.1.3. Operations [4]**

ขั้นตอนที่นำข้อมูลที่ได้จากการทำกระบวนการ EDA, Machine learning มาแสดงเป็นกราฟหรือรูปภาพต่างๆ เพื่อให้ผู้อื่นสามารถเข้าใจได้ง่าย

Model deployment เมื่อการเรียนรู้ด้วยเครื่องเป็นแบบจำลองที่ใช้เปรียบเทียบกับข้อมูลในอนาคต จะปรับใช้โมเดลในสภาพแวดล้อมเพื่อนำไปใช้กับข้อมูลใหม่

Model visualization ในข้อมูลขนาดเล็กไม่จำเป็นต้องเป็นแบบจำลองที่ผลิตด้วยการเรียนรู้ด้วยเครื่องสถานการณ์นี้เป็นรูปแบบการดำเนินงานที่พบมากที่สุดในdata science คือแบบจำลองที่ให้วิธีในการตอบคำถามเกี่ยวกับชุดข้อมูลต้นฉบับ ตัวเลือกสำหรับการสร้างภาพมีมากมายและสามารถผลิตได้จากภาษาการเขียนโปรแกรม

## 2.2 องค์ความรู้เกี่ยวกับ Machine Learning [6]

เป็นสาขาหนึ่งของปัญญาประดิษฐ์ที่พัฒนามาจากการศึกษาการรู้จำแบบเกี่ยวข้องกับการศึกษาและการสร้างอัลกอริทึมที่สามารถเรียนรู้ข้อมูลและทำนายข้อมูลได้ อัลกอริทึมนั้นจะทำงานโดยอาศัยโมเดลที่สร้างมาจากชุดข้อมูลตัวอย่างขาเข้าเพื่อการทำนายหรือตัดสินใจในภายหลัง แทนที่จะทำงานตามลำดับของคำสั่งโปรแกรมคอมพิวเตอร์ แบ่งออกเป็น 3 ประเภท

1. Supervised learning

การนำข้อมูล output ที่ถูกต้องให้ Machine ได้ทำการเรียนรู้ โดยอ้างอิงจาก Output หรือ Label ที่ได้ให้ข้อมูลไป สร้าง Algorithm/Model เพื่อทำนายผลลัพธ์ให้ โดยจะ Train Machine ไปจนกว่าจะได้ Output ที่ถูกต้องแม่นยำ ตัวอย่างการ Algorithm ที่เป็น Supervised Learning ได้แก่ Linear Regression, Decision Tree, Random Forest, KNN, Logistic Regression

2. Unsupervised learning

ตรงกันข้ามกันกับ Supervised learning จะไม่มีการระบุข้อมูล Output หรือ Label ของผลลัพธ์ที่ต้องการให้กับทาง Machine ได้เรียนรู้ แค่ระบุว่าต้องการ Ouput ออกมาเป็นกี่กลุ่ม แล้ว Machine Learning จะทำการแบ่งกลุ่มข้อมูลที่ใกล้เคียงกันให้ นิยมมากในการใช้แบ่ง Customer Segmentation ตัวอย่าง Unsupervised Learning ได้แก่ Dimensionality reduction, Clustering, Manifold learning

3. Reinforcement learning

เป็นด้านหนึ่งของ machine learning หรือ artificial intelligence (AI) ที่ใช้สำหรับพัฒนา robot (หรือ agent) ให้สามารถตัดสินใจภายใต้แต่ละสถานการณ์เพื่อนำมาซึ่งผลลัพธ์ที่ดีที่สุด โดยที่ robot นั้นจะไม่ได้ถูกบอกให้รู้ถึงกฎเกณฑ์ในการเลือกกระทำสิ่งใดภายใต้สถานการณ์ใดโดยตรง แต่ robot จะพยายามพัฒนาระบบความคิดการตัดสินใจเองจากการทดลองผิดลองถูกและเรียนรู้ไปเรื่อยๆ ตัวอย่างเช่น การเล่นหมากรุก จะต้องมีการทำนายล่วงหน้าว่าจะสามารถเกิดอะไรขึ้นได้บ้าง ซึ่งการเดินแต่ละครั้งอาจจะไม่เป็นผลดีต่อครั้งนั้นแต่อาจมีผลดีในครั้งต่อจากนั้นก็ได้

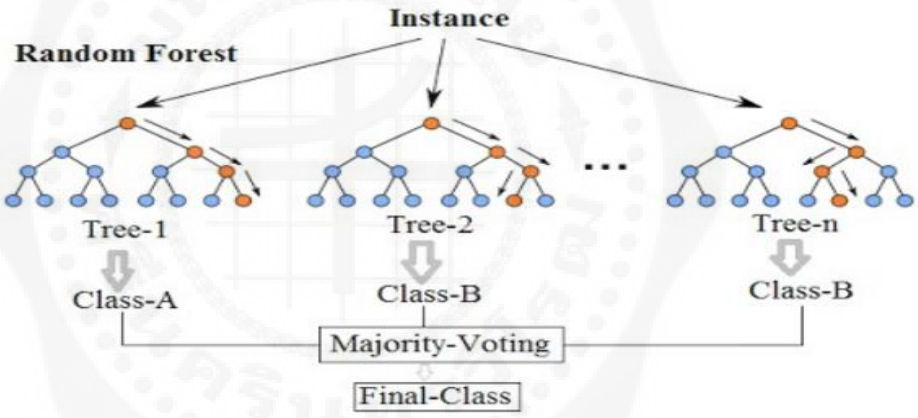
โดยงานวิจัยนี้จะเน้นไปที่ Supervised learning โดยโมเดลที่สนใจประกอบด้วย Random Forest, Gradient boosting, XGBoost, AdaBoost และ Light Gradient Boosting

### 2.2.1 องค์ความรู้เกี่ยวกับ Bagging [7]

Bagging ย่อมาจาก Bootstrap Aggregation โดย bagging tree นี้ จะทำการสร้าง หลายๆ subset แบบสุ่มจาก train data แล้วแต่ละชุดจะถูกเรียนรู้ด้วย Tree ผลลัพธ์ที่ได้คือ จะได้ Tree ที่หลากหลาย เนื่องจากเรียนรู้จากชุดข้อมูลแบบสุ่ม (random subset)

### 2.2.2 องค์ความรู้เกี่ยวกับ Random Forest [8]

แนวคิดของ Random Forest นี้คือการสร้างโมเดลด้วยวิธีการ Decision Tree ขึ้นมาหลายๆ โมเดลโดยวิธีการสุ่มตัวแปร แล้วนำผลที่ได้แต่ละโมเดลมารวมกันพร้อมนับจำนวนผลที่มีจำนวนฝั่งกันมากที่สุด สกัดออกมาเป็นผลลัพธ์สุดท้าย วิธีการของ Decision Tree คือเทคนิคที่ให้ผลลัพธ์ในลักษณะเป็นโครงสร้างของต้นไม้ ภายในต้นไม้จะประกอบไปด้วยโหนด(node) ซึ่งแต่ละโหนดจะมีเงื่อนไขของคุณลักษณะเป็นตัวทดสอบ กิ่งของต้นไม้(branch) แสดงถึงค่าที่เป็นไปได้ของคุณลักษณะที่ถูกเลือกทดสอบ และใบ (leaf) เป็นสิ่งที่อยู่ล่างสุดของต้นไม้ แสดงถึงกลุ่มของข้อมูล(class) ก็คือผลลัพธ์ที่ได้จากการพยากรณ์




รูปที่ 3 : Random Forest

(ที่มา : https://www.semanticscholar.org/paper/Detection-and-classification-of-breast-cancer-from-Ghongade-Wakde/04f3e376672318a7c8bfe12059069d033c688951)

Hyper-parameter ของ Random forest [8]

n_estimators : integer, optional (default=10) จำนวน tree ใน Random Forest จำนวน tree ที่มากขึ้นจะทำให้ model performance ดีขึ้นจนถึงจุดนึงที่ performance เริ่มจะนิ่งจนจำนวน tree ไม่มีผลต่อ model performance

oob_score : bool (default=False) เป็นการระบุว่าจะใช้ data ในส่วนที่ไม่ถูก sample ไปทำ training โดยแต่ละ decision tree ใน Random Forest สำหรับคำนวณ error ซึ่งเทียบเท่ากับการทำ validation จาก training set โดยที่ไม่ต้องทำแบ่งข้อมูลสำหรับทำ validation นั่นเอง ซึ่งจะมีประโยชน์อย่างมากตอนที่เรามี dataset ไม่ใหญ่มาก และไม่อยากแบ่งข้อมูลสำหรับ validation set อีก

min_samples_split : int, float, optional (default=2) ตัวอย่างจำนวนขั้นต่ำที่จำเป็นในการแบ่งโหนดภายใน หาก int ให้พิจารณา min_samples_split เป็นตัวเลขขั้นต่ำ หากเป็น float min_samples_split จะเป็นเศษส่วนและ ceil (min_samples_split * n_samples) คือตัวอย่างจำนวนขั้นต่ำสำหรับการแยกแต่ละครั้ง

min_samples_leaf : int, float, optional (default=1) การระบุจำนวนข้อมูลขั้นต่ำใน leaf node ของแต่ละ decision tree หากมีจำนวนข้อมูลต่ำกว่า min_samples_leaf ให้หยุด split node นั้นๆเป็นการลด overfitting จะกำหนดค่าmin_samples_leaf ให้สอดคล้องตามขนาดของ data set ตัวอย่างเช่น ตั้งแต่ 2–100 ตามขนาดข้อมูล

min_weight_fraction_leaf : float, optional (default=0) เศษส่วนน้ำหนักที่ต่ำสุดของผลรวมของน้ำหนักรวมที่จำเป็นต้องอยู่ที่ลีฟโหนด คล้าย min_samples_leaf

max_leaf_nodes : int หรือ None, optional (default=None) โหนดที่ดีที่สุดถูกกำหนดเพื่อเป็นการ ลดความสัมพันธ์ที่ไม่ชัดเจน ถ้าไม่มีการกำหนดจำนวนโหนดลีฟจะมีไม่จำกัด

min_impurity_decrease, max_impurity_decrease : float, optional (default=0) โหนดจะถูกแบ่ง ถ้าการแยกนี้มีค่าที่ลดลง มากกว่า หรือเท่ากับค่านี้

min_impurity_split : float, (default=1e-7) เกณฑ์สำหรับการหยุดการเจริญเติบโตของต้นไม้ โหนดจะแตกถ้ามีค่าอยู่เหนือขีดจำกัด มิฉะนั้นจะเป็นลีฟโหนด

max_features (0.0–1.0) : int, float, string หรือ None, optional (default=”auto”) การระบุว่าแต่ละ decision tree ใน Random Forest จะสามารถสุ่มหยิบ feature ได้มากที่สุดกี่เปอร์เซ็นต์ (0–100%) ซึ่งการไม่เซตค่าดังกล่าวสูงจนเกินไป จะเป็นการลดความสัมพันธ์กันเองของ tree (ลด correlation) และลดโอกาสการเกิด overfit ของ model

max_depth : integer หรือ None, optional (default=None) จำนวน level ที่มากที่สุด ของ node ที่จะทำการ split observation ซึ่งจะทำ tuning ค่า max_depth ไม่ให้มากเกินไป เพื่อป้องกันปัญหา overfitting

n_jobs : int หรือ None, optional (default=None) จำนวนของงานที่ต้องรันแบบ parallel ให้เหมาะสมต่อการวิเคราะห์ (ไม่มีหมายถึง 1)

random_state : int, RandomState instance หรือ None, optional (default=None) ถ้าเป็น int หรือ random_state เป็นเมล็ดพันธุ์ที่ใช้โดยเครื่องสุ่มตัวเลข ถ้าเป็น RandomState instance, random_state เป็นตัวสร้างตัวเลขสุ่ม ถ้าเป็น None ตัวสร้างตัวเลขสุ่มคือ RandomState instance ที่ใช้โดย np.random

verbose : int, optional (default=0) เป็นควบคุมการ verbosity ในขณะการทำนายให้เหมาะสม

warm_start : bool, optional (default=False) เมื่อตั้งค่าเป็น True ให้นำโซลูชันของการโทรก่อนหน้านี้กลับมาใช้ใหม่เพื่อเพิ่มและเพิ่มตัวประมาณค่าลงในวงดนตรีมิฉะนั้นก็พอดีกับป่าใหม่ทั้งหมด

class_weight : dict, list of dicts, “balanced”, “balanced_subsample” หรือ None, optional (default=None) ค่าweightที่เกี่ยวข้องกับคลาสในรูปแบบ {class_label: weight} หากเป็น None จะมีการสนับสนุนค่าค่าหนึ่งของweight สำหรับปัญหาหลายเอาท์พุต list ของ dicts สามารถจัดเรียงตามลำดับเช่นเดียวกับคอลัมน์ของ y

criterion : string, optional (default=”gini”) cost function ที่จะใช้ สำหรับ regression tree จะมีค่า default เป็น ‘mse’ หรือ mean square error ส่วน classification tree จะ มีค่า default เป็น ‘gini’ ซึ่งสามารถจะเลือกเป็น ‘entropy’ ก็ได้

จาก Hyper-parameter ทั้งหมดที่กล่าวมาข้างต้น ตัวที่ส่งผลต่อโมเดลมากที่สุดคือ max_depth

### 2.2.2 องค์ความรู้เกี่ยวกับ Boosting [9]

Boosting เป็นพื้นฐานของ AdaBoost หรือ Gradient Boosting ในไลบรารี่เช่น XGBoost และ LightGBM boosting คือการนำ weak classifier หรือ classifier ที่มีความแม่นยำต่ำมาทำนายข้อมูลที่มี จากนั้นจะให้ weak classifier ตัวใหม่มาแก้ไข error ที่มี โดยผลรวมของ classifier จะเกิดเป็น classifier ใหม่ขึ้นมา จะทำแบบนี้ไปจนได้โมเดลที่ดีที่สุดจากผลรวมของ classifier ถ้ามองภาพรวมการทำงานของ Boosting ก็เหมือนการทำงานเป็นทีม โดยการเอา classifier ที่ไม่ได้ดีมากมารวมกันจนทำนายข้อมูลที่ซับซ้อนมากๆได้ ข้อเสียของการใช้ boosting คือต้องรันหลายครั้งและเป็นลำดับกว่าจะได้โมเดลที่ต้องการ ต่างจาก Bagging ที่สามารถสุ่มข้อมูลได้แล้วเทรนโมเดลได้พร้อมๆกันเลย แต่ข้อดีของ library เช่น XGBoost นั้น เราจะสามารถเลือก ชนิดของ weak learner หรือ weak classifier นี้ได้ด้วย ไม่ว่าจะเป็นแบบต้นไม้ (tree) หรือแบบเส้นตรง (linear) และในหลายๆครั้งนั้นเทคนิค boosting สามารถทำนายข้อมูลที่มีความซับซ้อนมากๆได้มากกว่าการใช้ bagging

### 2.2.3 องค์ความรู้เกี่ยวกับ XGBoost [10]

XGBoost หรือ Extreme Gradient Boosting ถูกแต่งเสริมเติมแต่งจาก Gradient boosting เป็น model ที่นำเอา Decision Tree มา train ต่อๆกันหลายๆ tree โดยที่แต่ละ decision tree จะเรียนรู้จาก error ของ tree ก่อนหน้า ทำให้ความแม่นยำของในการทำ prediction จะ แม่นยำมากขึ้นเรื่อยๆ เมื่อมีการเรียนรู้ของ tree ต่อเนื่องกันจนมีความลึกมากพอ และ model จะหยุดเรียนรู้เมื่อไม่เหลือ pattern ของ error จาก tree ก่อนหน้าให้เรียนรู้แล้ว

### 2.2.4 องค์ความรู้เกี่ยวกับ AdaBoostClassifier [11]

Adaptive Boosting Algorithm หรือ AdaBoost เป็นเทคนิคในกลุ่ม Boosting ที่มีการนำมาใช้พัฒนาต่อเนื่อง มีงานวิจัยการนำไปประยุกต์ใช้งานใน machine learning ด้านต่างๆ เช่น Face Detection โมเดลนี้ที่ได้รับความนิยมมาก เพราะมีความยืดหยุ่นปรับใช้ได้กับทุก learning algorithm

และใช้เทคนิคของการคำนวณซ้ำ แล้วปรับค่าน้ำหนัก ทำให้ลด Bias Error ลงได้ และไม่ทำให้เกิดการ over fitting เกินไป แนวคิดหลักของ AdaBoost คือการเปลี่ยน weak model ให้กลายเป็น strong model ประมวลผลวิเคราะห์ข้อมูลกลุ่ม data set เดี่ยวกัน ใช้โมเดลเดียวกัน เริ่มต้นจากการ classify แบบธรรมดา เพื่อทำการ observe ค่า error ที่เกิด ก่อนเริ่ม ทำการ adjust weight ในกลุ่มข้อมูล เพื่อทำการ classify ต่อเนื่อง จนกว่าได้ strong model ที่มี accuracy ดีที่สุด ออกมา

**2.2.5 องค์ความรู้เกี่ยวกับ Gradient Boosting [10]**

การสร้างแต่ละ Tree จะเป็นแบบ Sequence โดย input แต่ละ Tree จะเป็น output จาก Tree ก่อนหน้า โดย concept คือ GBT จะทำการสร้างแต่ละ Tree เพื่อลดค่า error ที่เกิดจาก Tree ก่อนหน้า โดยวิธี Gradient Descend

Y(actual) = Y(predict) + Error< - (สร้าง Tree เพื่อ predict Error)

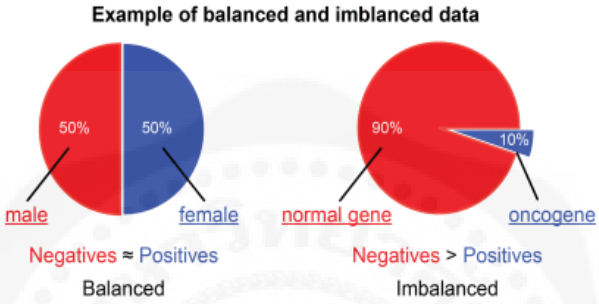
แล้วนำผลลัพธ์ที่ได้มารวมกัน ก็จะทำให้ได้ค่าใกล้เคียงกับ Y(actual)

**2.2.6 องค์ความรู้เกี่ยวกับ Light Gradient Boosting Machine (LightGBM) [12]**

Light Gradient Boosting ตัวนี้ใช้เทคนิค leaf-wise growth คือแตก tree ไปตามใบต่างๆ นอกจากนี้ยังใช้ ฮิสโตแกรม ในการประมาณการ data หาจุดตัดแต่ละ node แทนที่จะรันสแกนบนเดต้าทั้งฐาน ทำให้สามารถรันโมเดลได้เร็ว แม้ dataset จะมีขนาดใหญ่

### 2.3 องค์ความรู้เกี่ยวกับข้อมูลที่ไม่สมดุล (Imbalanced Datasets) [13]

ข้อมูลที่ไม่สมดุล หมายถึง ข้อมูลที่มีการกระจายตัวที่ไม่เท่าเทียมกัน หรือข้อมูลซึ่งอัตราจำนวนสมาชิกในกลุ่มหลักและกลุ่มรองมีจำนวนไม่เท่ากัน เช่น 100:1, 1000:1 หรือ 10000:1 เป็นต้น ยกตัวอย่างเช่นจากรูปที่ 4 ทางขวา เป็นข้อมูลประชากรเกี่ยวกับยีนส์ที่ปกติกับยีนส์ที่ไม่ปกติ ปัญหานี้คำตอบที่เราต้องการ คือ คนมียีนส์ไม่ปกติ (Positive) หรือ คนที่มียีนส์ปกติ (Negative) ซึ่งข้อมูลประชากรที่มียีนส์ปกติ (กลุ่มหลัก) อาจจะมีข้อมูลหลายหมื่นคน แต่ข้อมูลประชากรคนที่มียีนส์ไม่ปกติอาจมีเพียงประมาณหลักร้อยคน (กลุ่มรอง) ดังนั้น ถ้านำข้อมูลทั้งสองกลุ่มมาแบ่งกลุ่มข้อมูล พร้อมกันทั้งหมด เราก็จะพบว่า คำตอบของการพยากรณ์เพื่อการแบ่งกลุ่มข้อมูลทุกๆ ข้อมูลจะถูกจัดเป็นกลุ่มประชากรที่มียีนส์ปกติทั้งหมด (Negative Class) เนื่องจากข้อมูลของคนที่มียีนส์ปกติมากกว่าคนที่มียีนส์ที่ผิดปกติ ส่วนรูปทางซ้าย คือ ตัวอย่างของข้อมูลที่สมดุลกันระหว่างเพศชายและเพศหญิง

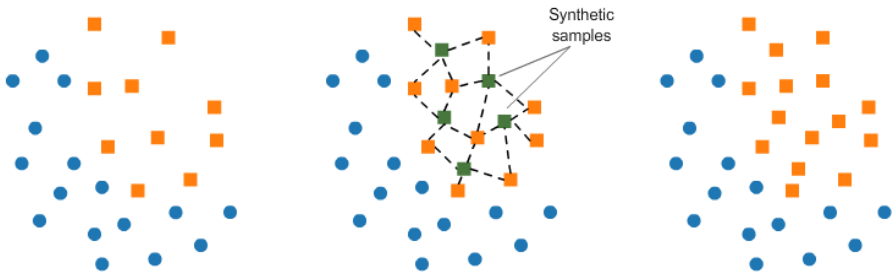



รูปที่ 4 : imbalanced datasets

(ที่มา : https://blog.goodaudience.com/how-to-handle-imbalanced-datasets-d67176dfb0aa)

### 2.4 องค์ความรู้เกี่ยวกับ Synthetic Minority Over-samplingTechnique (SMOTE) [14]

Synthetic Minority Over-sampling Technique: SMOTE เป็นเทคนิคที่ใช้ในการแก้ปัญหาที่ต้องการจำแนกข้อมูลไม่สมดุล ซึ่งข้อมูลมีจำนวนตัวอย่างแตกต่างกันมากในแต่ละคลาส เมื่อทำการจำแนกประเภท จะทำให้มีการเรียนรู้แต่ข้อมูลกลุ่มที่มาก ผลที่ได้ก็จะจำแนกไปในข้อมูลกลุ่มมากวิธี SMOTE เป็นวิธีการเพิ่มจำนวนข้อมูลประเภทที่มีข้อมูลน้อย ให้เพิ่มปริมาณข้อมูลใกล้เคียงกับประเภทที่มีมากที่สุดโดยสุ่มค่าขึ้นมาหนึ่งค่าและหาค่าระยะห่างระหว่างค่าที่เลือกกับทุกๆ ค่า เลือกค่าที่ใกล้เคียงที่สุด จากรูปที่4 จะเห็นได้ว่าข้อมูลสีส้มมีจำนวนที่น้อยกว่าข้อมูลฝั่งสีฟ้า ทำให้สีส้มมีการเพิ่มจำนวนข้อมูลที่ใกล้เคียงกันขึ้นมา(ข้อมูลสีเขียว) เมื่อทำการเพิ่มข้อมูลเสร็จจะเห็นได้ว่าข้อมูลสีส้มจะมีจำนวนที่เพิ่มขึ้นให้ใกล้เคียงหรือเท่ากันกับสีฟ้า



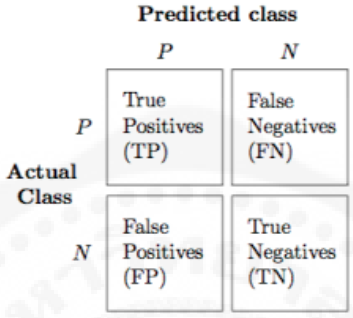

รูปที่ 5 : SMOTE

(ที่มา : https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets)

## 2.5 การวัดประสิทธิภาพของโมเดล [15]

### 2.5.1 องค์ความรู้เกี่ยวกับ Confusion Matrix

การจำลองรูปแบบการอธิบายประสิทธิภาพของโมเดลของ Binary Classifier จะใช้ Confusion Matrix เป็นเครื่องมือในการอธิบาย และ คำนวณผลการทำนายของโมเดลในเรื่องต่างๆ โดยความหมายของ Confusion Matrix จะอธิบายได้ดังรูปที่ 6




รูปที่ 6: Confusion Matrix

(ที่มา : https://www.auoychai.com/big-datadata-analytic/%E0%B8%A/)

จากรูปที่ 6 Actual Class คือ คือ รายการ Label ของข้อมูลที่เป็นคำตอบสำหรับตอนทำโมเดล Predicted Class คือ รายการผลการทำนายของโมเดล ค่าในตำแหน่งต่างๆของ Matrix จะเป็นได้ดังนี้ ตัวอย่างอธิบายรูปที่ 6 คือการทำนายว่าเป็นโรคมะเร็งหรือไม่ โดยสมมุติว่าค่าที่คาดหวังหรือ Actual / Positive คือ ไม่เป็นมะเร็ง

TP (True Positive) คือ ผลการทำนายทายได้ถูกต้องตามค่าที่คาดหวัง (Actual) ที่เป็น Positive กรณีนี้คือ Actual ไม่เป็นมะเร็ง แล้วทำนายว่า ไม่เป็นมะเร็ง

FP (False Positive) คือ ผลทำนายไม่ถูกต้อง ตามค่าที่คาดหวัง ที่เป็น Negative กรณีนี้คือ Actual เป็นมะเร็ง แล้ว ผลทำทายว่าไม่เป็นมะเร็ง

TN (True Negative) คือ ผลการทำนายทายได้ถูกต้องตามค่าที่คาดหวัง ที่เป็น Negative กรณีนี้คือ Actual เป็นมะเร็ง แล้ว ผลทำนายว่าเป็นมะเร็ง

FN (False Negative) คือ ผลการทำนายไม่ถูกต้องตามค่าที่คาดหวังที่เป็น Positive กรณีนี้คือ Actual ไม่เป็นมะเร็ง แล้วผลทำนายว่าเป็นมะเร็ง

การแทนค่าในแต่ละตัววัด ก็นับเอาตามผลการการทดสอบโมเดลที่ทำนายได้เทียบกับผลจริงของ Data Test

Precision คือความแม่นยำ Precision = TP / (TP + FP) ซึ่งสรุปได้ว่าถ้าต้องการค่าความแม่นยำเป็น 100% ก็ต้องให้ค่า FP ที่ทายผิดแบบเสียหายเชิงคุณภาพมีค่าเป็น ศูนย์ คำถามคือในงานจริง Precision ต้องเป็นเท่าไร บางงานที่ อาจจะต้องเป็น 100% แต่ก็ต้องแลกกับการทำโมเดลให้ดี หรือ ใช้เวลาในการพัฒนานาน หรือ ใช้ Algorithm ที่เหมาะสม เช่นผลการทำนายเกี่ยวข้องกับชีวิต ขณะที่บางงานอาจจะมีการกำหนดเปอร์เซ็นต์ error ให้ยอมรับได้ แต่ทางปฏิบัติอาจจะต้องเอาผลการทำงานมาตรวจสอบและใช้ค่าบ้างค่าใน Feature หนึ่งมาเป็นตัวกลั่นกรองสำหรับการเลือกรายการนั้นมาตรวจสอบซ้ำด้วยคนทำงานอีกครั้ง

Recall คือความอ่อนไหวโดยมีสูตรในการคำนวณดังนี้ Recall = TP / (TP + FN)ซึ่งสรุปได้ว่า ถ้าความอ่อนไหว 100% ก็ต้องให้ค่า FN ที่ทายผิดแบบเสียหายเชิงปริมาณมีค่าเป็น ศูนย์ นั่นเอง แต่ถ้าหากค่า Recall = 95% นั้นหมายความจะมีรายการที่ทำนายผิด แบบรายการจริงไม่เป็น แต่ผลทำนายว่าเป็น ตามตัวอย่างข้างต้น ถึง 5% ในเชิงปฏิบัติเราสรุปได้ว่า การโยนของดีเข้าไปในกลุ่มของเสียถึง 5% ที่ต้องถูกโยนทิ้ง

Accuracy คือความถูกต้อง ซึ่งความถูกต้องนี้คือ ทำนายถูกทั้งแบบ Positive และ Negative โดยดูจากสูตรแล้ว Accuracy = (TP + TN) / Total Data Test สรุปได้ว่า ถ้าทายถูกทั้งหมด แล้วค่า Accuracy = 100% แต่ในทางปฏิบัติจริงไม่มีทางที่จะเป็นได้ในลักษณะนี้

### 2.5.2 องค์ความรู้เกี่ยวกับ F1 Score

ค่าน้ำหนักเฉลี่ยของ Recall และ Precision (F1 = 2 * (Precision * Recall / Precision + Recall)) สรุปแบบตามแนวทางทฤษฎี สู่การปฏิบัติคือค่าที่ดีที่สุดคือ 1 และ แย่สุดคือ 0

## 2.6 งานวิจัยที่เกี่ยวข้อง

Transfer Learning from Deep Features for Remote Sensing and Poverty Mapping (2015) [2]

ผู้แต่ง Michael Xie, Neal Jean, Marshall Burke, David Lobell and Stefano Ermone

งานวิจัยนี้กล่าวถึงการเรียนรู้เกี่ยวกับเครื่องจักรเพื่อดึงตัวบ่งชี้ทางเศรษฐกิจและสังคมในระดับอุดมศึกษาจากขั้นสูงภาพถ่ายจากดาวเทียม ความท้าทายหลักคือข้อมูลการ train หายากมากทำให้ยากที่จะใช้เทคนิคสมัยใหม่ เช่น Convolutional Neural Networks (CNN) ดังนั้นเราจึงเสนอวิธีการเรียนรู้การถ่ายโอนข้อมูลโดยใช้ความเข้มของแสงในเวลา โดยแบบจำลอง CNN เต็มรูปแบบเพื่อคาดการณ์แสงในเวลาการคืนจากภาพกลางวันพร้อมกับการเรียนรู้คุณลักษณะที่เป็นประโยชน์สำหรับการทำนายความยากจน แบบจำลองนี้จะเรียนรู้เกี่ยวกับตัวกรองที่ระบุภูมิประเทศที่แตกต่างกันและโครงสร้างที่มนุษย์สร้างขึ้นรวมทั้งถนนอาคารและพื้นที่เกษตรกรรมโดยไม่มีการควบคุมใด ๆ นอกเหนือจากแสงยามค่ำ เราแสดงให้เห็นว่าคุณลักษณะที่เรียนรู้เหล่านี้เป็นข้อมูลที่ให้ข้อมูลเกี่ยวกับการทำแผนที่ความยากจนแม้จะใกล้เคียงกับข้อมูลที่ได้จากการสำรวจในฟิลด์ที่คาดการณ์ไว้

Mapping poverty using mobile phone and satellite data (2017) [1]

ผู้แต่ง Jessica E. Steele, Pa˚l Roe Sundsøy, Carla Pezzulo , Victor A. Alegana , Tomas J. Bird1 , Joshua Blumenstock, Johannes Bjelland , Kenth Engø-Monsen, Yves-Alexandre de Montjoye , Asif M. Iqbal , Khandakar N. Hadiuzzaman, Xin Lu, Erik Wetter, Andrew J. Tatem and Linus Bengtsson

งานวิจัยนี้กล่าวถึงการประเมินค่าความยากจนแบบดั้งเดิม โดยใช้ข้อมูลจากผู้ให้บริการโทรศัพท์มือถือและข้อมูลภูมิสารสนเทศที่มีอยู่อย่างแพร่หลาย โมเดลที่รวมแหล่งข้อมูลเหล่านี้ให้ผลดีที่สุดในการคาดการณ์ (สูงสุด $r^2$ = 0.78) และมีข้อผิดพลาดต่ำสุด แต่โดยทั่วไปโมเดลที่ใช้ข้อมูลมือถือจะให้ผลลัพธ์ที่เท่ากันซึ่งจะนำเสนอศักยภาพในการวัดความยากจนได้บ่อยขึ้นและในระดับละเอียดมากขึ้น การแบ่งแยกรูปแบบในพื้นที่เมืองและชนบทให้ความสำคัญกับการใช้ข้อมูลมือถือในเขตเมืองและข้อมูลที่แตกต่างกันในบริบทที่แตกต่างกัน ผลการวิจัยชี้ให้เห็นถึงความเป็นไปได้ในการประมาณและติดตามอย่างต่อเนื่องในอัตราความยากจนที่มีความละเอียดเชิงพื้นที่สูงในประเทศที่มีขีดจำกัดในการรองรับวิธีการเก็บข้อมูลแบบเดิม

Costa Rican Household Poverty Level Prediction (2018) [3]

ผู้แต่ง Will Koehrsen

งานวิจัยนี้กล่าวถึงมีวัตถุประสงค์เพื่อคาดการณ์ความยากจนในระดับครัวเรือน ข้อมูลที่ได้รับเกี่ยวกับระดับปัจเจกบุคคลกับแต่ละบุคคลที่มีคุณลักษณะเฉพาะ และข้อมูลเกี่ยวกับครอบครัวของพวกเขา ต้องทำนายทุกคนในชุดทดสอบ แต่ "มีเพียงหัวหน้าครัวเรือนเท่านั้นที่ถูกนำมาใช้ในการให้คะแนน" ซึ่งหมายความว่าต้องการทำนายความยากจนในครัวเรือน เพื่อแสดงขั้นตอนการสร้างแบบจำลองครั้งแรกจะใช้ตัวจำแนกประเภท Random Forest Class ช่วยให้สามารถสร้างพื้นฐาน ใช้มาตรการ F1 Macro เพื่อประเมินประสิทธิภาพ งานวิจัยนี้ได้ลองใช้โมเดลขั้นพื้นฐานอย่างหนึ่งแล้วคือ Random Forest Classifier ซึ่งมีมาโคร F1 ที่ดีที่สุดเท่ากับ 0.34 และ Gradient Boosting Machine ที่ดีที่สุดเท่ากับ 0.43 แต่โดยรวมยังไม่แม่นยำมาก อาจมีวิธีปรับปรุงประสิทธิภาพ แต่โดยรวมอาจไม่มีข้อมูลเพียงพอที่จะบรรลุ ประเด็นสำคัญที่ต้องจดจำคือในตอนท้ายของความสำเร็จหรือความล้มเหลวของโครงการวิทยาศาสตร์ข้อมูลจะเน้นเรื่องคุณภาพและปริมาณข้อมูลที่มีอยู่

Predicting Poverty - World Bank (2018) [16]

ผู้แต่ง World Bank

ธนาคารโลกมีเป้าหมายที่จะยุติความยากจนจนสิ้นปี 2030 จุดสำคัญสำหรับเป้าหมายนี้คือเทคนิคในการกำหนดกลยุทธ์การลดความยากจนที่ใช้และกลยุทธ์ใดที่ไม่เหมาะสม แต่การวัดการลดความยากจนจำเป็นต้องมีการวัดความยากจนในตอนแรกและปรากฏว่าการวัดความยากจนเป็นเรื่องยาก ธนาคารโลกช่วยให้ประเทศกำลังพัฒนาสามารถวัดความยากจนด้วยการทำแบบสำรวจเชิงสำรวจแบบเจาะลึกกับกลุ่มย่อยของประชากรในประเทศ ในการวัดความยากจนการสำรวจส่วนใหญ่เหล่านี้รวบรวมข้อมูลรายละเอียดเกี่ยวกับการบริโภคของครัวเรือนทุกอย่างตั้งแต่อาหารและพฤติกรรมการขนส่งไปจนถึงการเข้าถึงการรักษาพยาบาลและการแข่งขันกีฬา เพื่อให้ได้ภาพที่ชัดเจนเกี่ยวกับสถานะความยากจนของครัวเรือน และนำ XGBoost, LightGBM และ CatBoost มาใช้ในการทำนายความยากจนและหาเทคนิคที่เหมาะสมที่สุดที่ได้ผลลัพธ์ที่ดีที่สุด

# บทที่ 3
# การดำเนินงานวิจัย

ในบทนี้จะเริ่มต้นโดยกล่าวถึงชุดข้อมูล (Dataset) ที่ใช้ จากนั้นจะอธิบายในส่วนของการทำ Exploratory Data Analysis (EDA) และกล่าวถึงวิธีการที่นำเสนอ แผนการดำเนินงาน อุปกรณ์และเครื่องมือที่ใช้พัฒนา

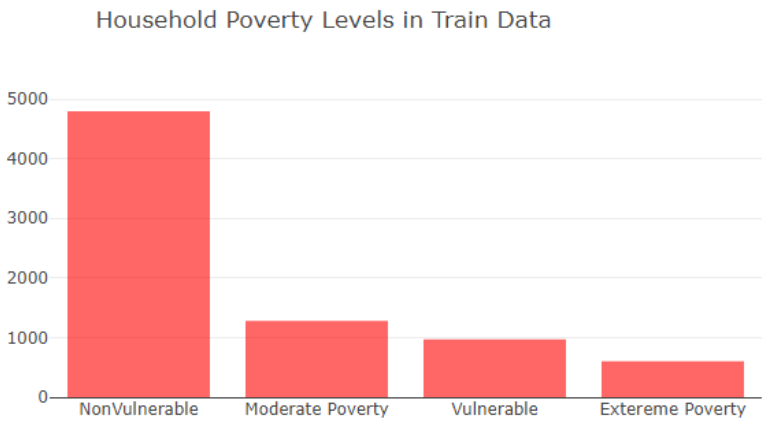
## 3.1 ชุดข้อมูล (Dataset) ที่ใช้ [3]

ชุดข้อมูลสำมะโนประชากรของประเทศคอสตาริกาที่เราสนใจนั้นมีทั้งหมด 9557 แถวและ 143 คอลัมน์ (มีคุณลักษณะเฉพาะจำนวน 143 ชนิด โดยมีคุณลักษณะเฉพาะที่เป็นข้อมูลชนิด Float64 จำนวน 8 ตัว คุณลักษณะเฉพาะที่เป็นข้อมูลชนิด Int64 จำนวน 130 ตัวและคุณลักษณะเฉพาะที่เป็นข้อมูลชนิด Object จำนวน 5 ตัว) โดยรูปที่ 7 แสดงตัวอย่างของชุดข้อมูลแถวแต่ละแถวแสดงถึงบุคคลหนึ่งคนและแต่ละคอลัมน์เป็นคุณลักษณะเฉพาะสำหรับแต่ละบุคคลหรือสำหรับแต่ละครัวเรือนของแต่ละคน ยกตัวอย่างเช่น idhogar คือตัวระบุครัวเรือนแต่ละครัวเรือน parentesco1 คือระบุว่าบุคคลนี้เป็นหัวหน้าครัวเรือนหรือไม่ (0-ไม่เป็น, 1-เป็น)

| idhogar | parentesco1 | Target |
|---|---|---|
| 0172ab1d9 | 0 | 3 |
| 0172ab1d9 | 0 | 2 |
| 0172ab1d9 | 0 | 3 |
| 0172ab1d9 | 1 | 3 |
| 0172ab1d9 | 0 | 2 |

รูปที่ 7: ตัวอย่างคอลัมน์จากคอลัมน์ทั้งหมด

คอลัมน์ Target แสดงถึงระดับความยากจนในระดับ1-4 โดยคลาส 1 คือ Extreme Poverty หรือระดับที่ 1 คือครัวเรือนที่มีความเสี่ยงสูงที่จะยากจน คลาส 2 Vulnerable คือครัวเรือนที่มีความเสี่ยงที่จะยากจนพอสมควร คลาส 3 คือครัวเรือนที่มีความเสี่ยงเล็กน้อยที่จะยากจน และคลาส 4 NonVulnerable คือครัวเรือนที่ไม่มีความเสี่ยงที่จะยากจน โดยในชุดข้อมูลจะมีคลาส 1 จำนวน 604 แถว คลาส 2 จำนวน 967 แถว คลาส 3 จำนวน 1,278 แถว และคลาส 4 จำนวน 4,796 แถวดังแสดงในรูปที่ 8



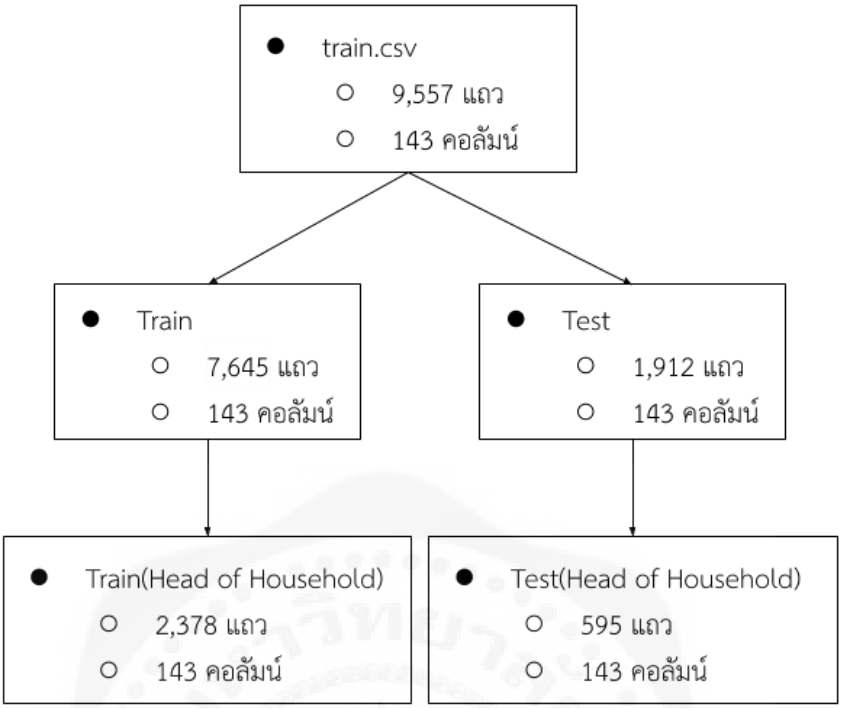

รูปที่ 8 : ระดับความยากจนของครัวเรือน

ขั้นตอนหนึ่งที่สำคัญที่สุดในการวิเคราะห์ข้อมูลสำรวจคือการหาค่าที่ขาดหายไปในข้อมูลและวิธีจัดการกับข้อมูล ค่าที่ขาดหายไปจะต้องได้รับการเติมเต็มก่อนที่เราจะใช้โมเดลการเรียนรู้ด้วยเครื่อง จากตารางที่ 1 คือจำนวนข้อมูลที่หายไปในแต่ละคอลัมน์ โดยคอลัมน์ rez_esc ซึ่งหมายถึงจำนวนปีหลังจบการศึกษา มีข้อมูลหายไปจำนวน 7,926 ค่า (83.31 เปอร์เซ็นต์) คอลัมน์ v18q1 ซึ่งหมายถึงจำนวน ของแท็บเล็ตในครัวเรือน มีข้อมูลหายไปจำนวน 7,342 ค่า (76.82 เปอร์เซ็นต์) คอลัมน์ v2a1 ซึ่งหมายถึงค่าเช่ารายเดือน มีข้อมูลหายไปจำนวน 6,860 ค่า (71.77 เปอร์เซ็นต์) คอลัมน์ SQBmeaned ซึ่งหมายถึงปีการศึกษาโดยเฉลี่ยสำหรับมีอายุเกิน 18 ปีในครัวเรือนยกกำลังสอง มีข้อมูลหายไปจำนวน 5 ค่า (0.05 เปอร์เซ็นต์) คอลัมน์ Meaneduc ซึ่งหมายถึงปีการศึกษาโดยเฉลี่ยสำหรับมีอายุเกิน 18 ปีในครัวเรือน มีข้อมูลหายไปจำนวน 5 ค่า (0.05 เปอร์เซ็นต์) ข้อมูลที่หายไปเราจะทำการแทนค่า 0 ลงไปแทน

ตารางที่ 1: จำนวนข้อมูลที่หายไปในแต่ละ column

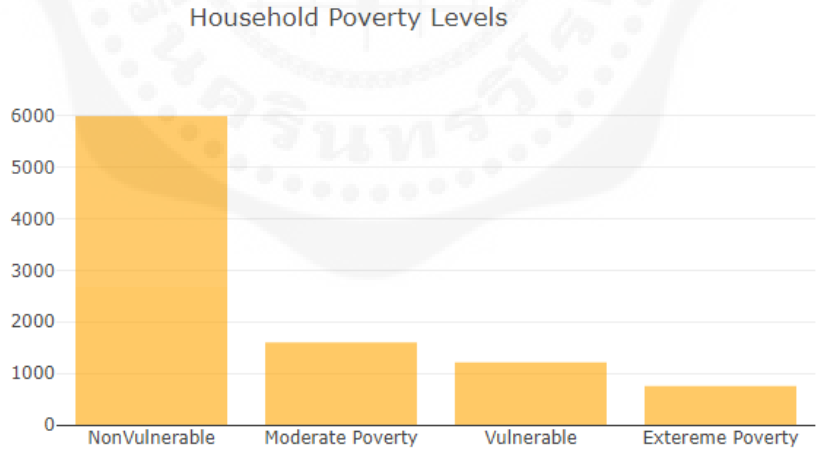
| | Total | Percent |
|---|---|---|
| rez_esc | 7926 | 83.31 |
| v18q1 | 7342 | 76.82 |
| v2a1 | 6860 | 71.77 |
| SQBmeaned | 5 | 0.05 |
| meanduc | 5 | 0.05 |

### 3.2 Exploratory Data Analysis (EDA)



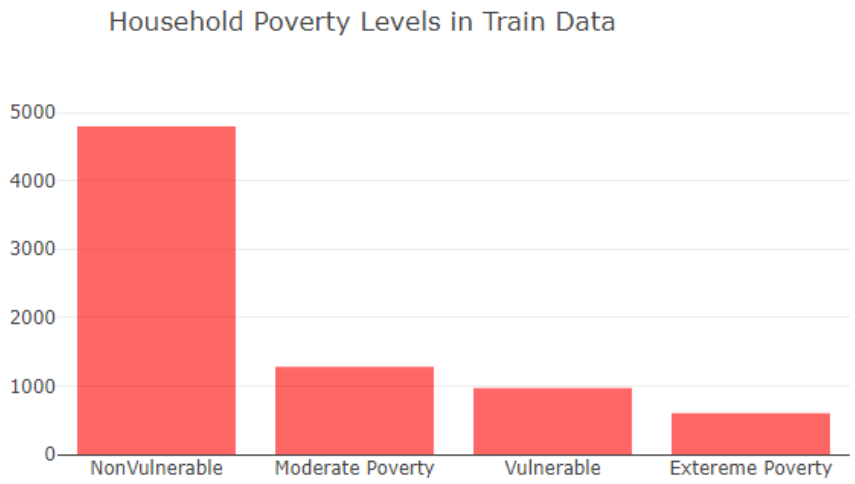

รูปที่ 9 : การจัดเตรียมข้อมูล

รูปที่ 9 คือการจัดเตรียมข้อมูล โดยข้อมูลทั้งหมดหรือ train.csv จะทำการแบ่งเป็น train dataset คิดเป็น 80% และ test dataset คิดเป็น 20% แล้วจะดึงเอาเฉพาะหัวหน้าครอบครัวมาคิด



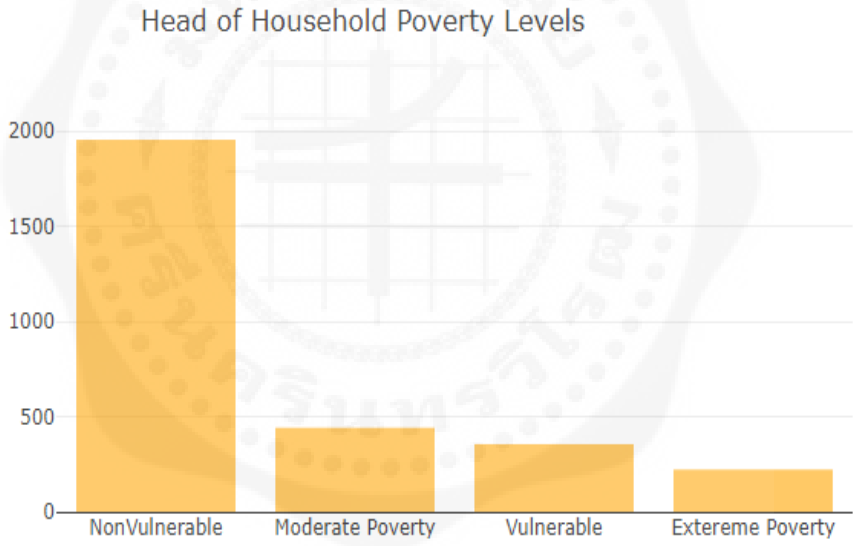

รูปที่ 10 : ข้อมูลระดับความยากจนในครัวเรือนทั้งหมด

รูปที่ 10 แสดงข้อมูลระดับความยากจนในครัวเรือน ซึ่งถูกแบ่งออกเป็น 4 ระดับ คือ NonVulnerable มีข้อมูลทั้งหมด 5996 แถว Moderate Poverty มีข้อมูลทั้งหมด 1597 แถว Vulnerable มีข้อมูลทั้งหมด 1209 แถว และExtreme Poverty มีทั้งหมด 755 แถว



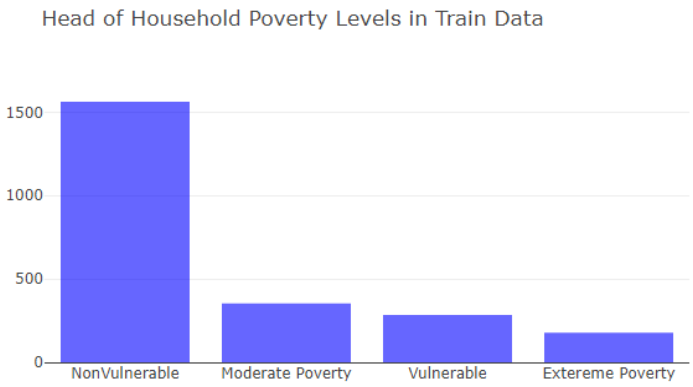

รูปที่ 11 : ข้อมูลระดับความยากจนในครัวเรือนของ Train Data

รูปที่ 11 แสดงข้อมูลระดับความยากจนในครัวเรือนของ Train Data ซึ่งถูกแบ่งออกเป็น 4 ระดับ คือ NonVulnerable มีข้อมูลทั้งหมด 4796 แถว Moderate Poverty มีข้อมูลทั้งหมด 1278 แถว Vulnerable มีข้อมูลทั้งหมด 967 แถว และExtreme Poverty มีทั้งหมด 604 แถว



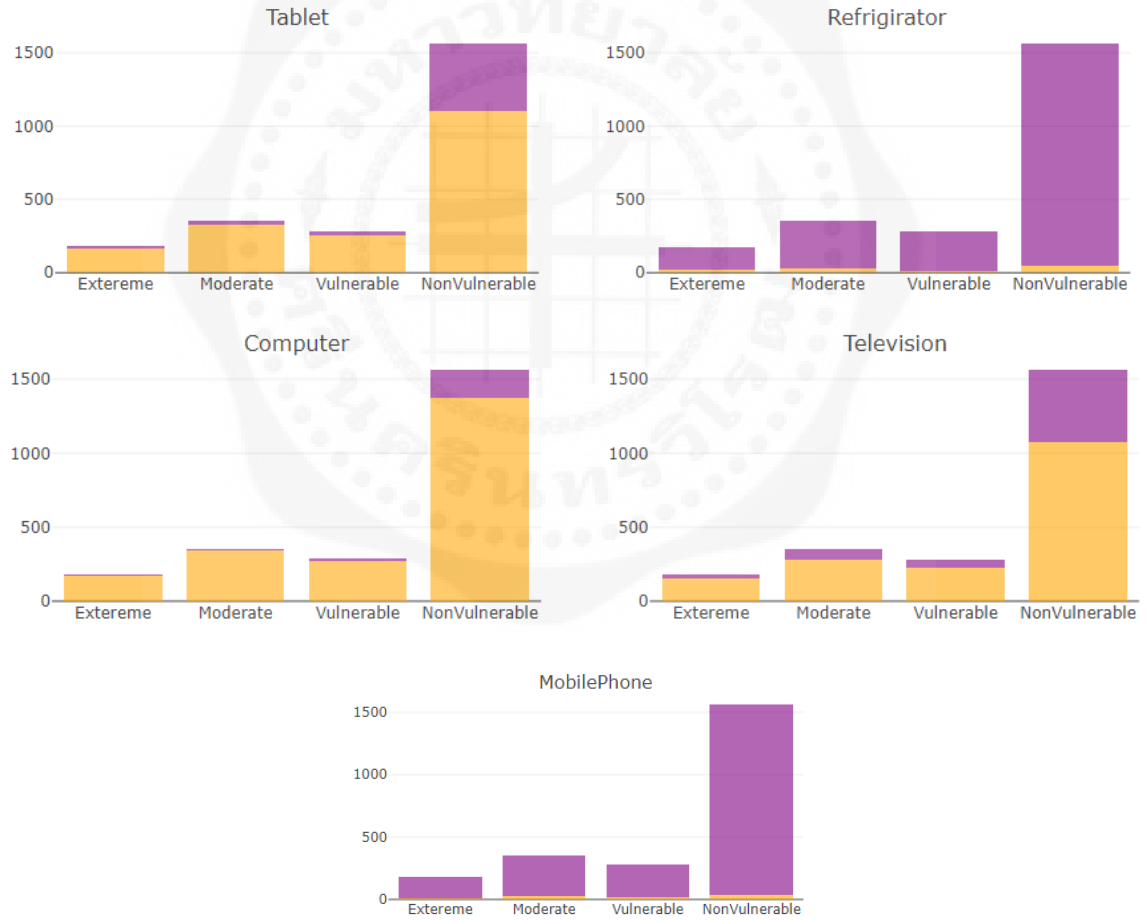

รูปที่ 12: ข้อมูลระดับความยากจนของหัวหน้าครอบครัวในครัวเรือนทั้งหมด

รูปที่ 12 แสดงข้อมูลระดับความยากจนของหัวหน้าครอบครัวในครัวเรือน ซึ่งถูกแบ่งออกเป็น 4 ระดับ คือ NonVulnerable มีข้อมูลทั้งหมด 1954 แถว Moderate Poverty มีข้อมูลทั้งหมด 442 แถว Vulnerable มีข้อมูลทั้งหมด 355 แถว และExtreme Poverty มีทั้งหมด 222 แถว




รูปที่ 13 : ข้อมูลระดับความยากจนของหัวหน้าครอบครัวในครัวเรือนของ Train Data

รูปที่ 13 แสดงข้อมูลระดับความยากจนของหัวหน้าครอบครัวในครัวเรือน ซึ่งถูกแบ่งออกเป็น 4 ระดับ คือ NonVulnerable มีข้อมูลทั้งหมด 1563 แถว Moderate Poverty มีข้อมูลทั้งหมด 363 แถว Vulnerable มีข้อมูลทั้งหมด 284 แถว และExtreme Poverty มีทั้งหมด 178 แถว



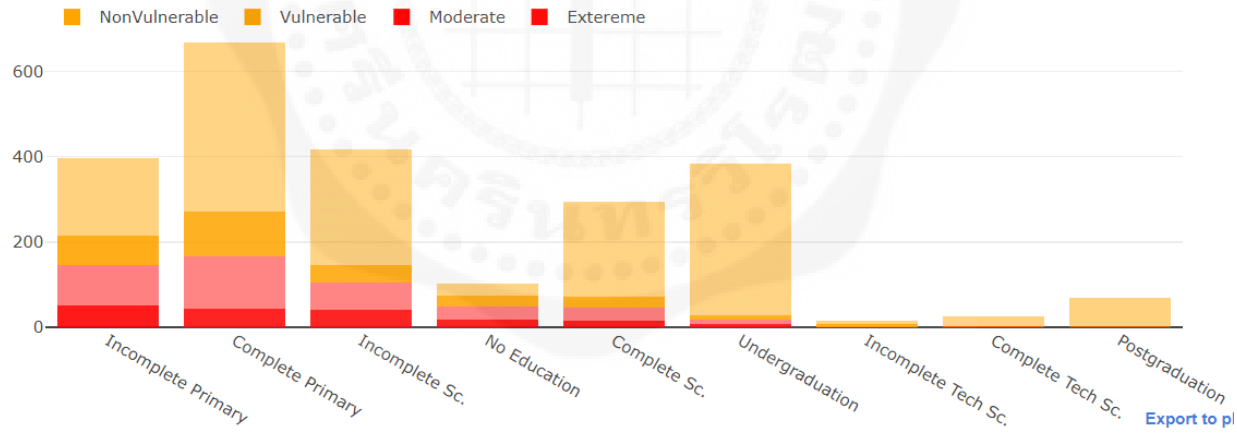

รูปที่ 14 : จำนวนสินทรัพย์ของหัวหน้าครอบครัวในครัวเรือน

รูปที่ 14 จะแบ่งเป็นสีม่วงหรือข้างบน คือ มีระบุไว้ (1) และสีเหลืองหรือข้างล่าง คือ ไม่มีระบุไว้ (0) เพื่อทำให้รู้จำนวนสินทรัพย์ของหัวหน้าครอบครัวในครัวเรือน สามารถสรุปได้เป็นตารางที่ 2

ตารางที่ 2 : แสดงจำนวนสินทรัพย์ของแต่ละคลาส

| สินทรัพย์ | NonVulnerable | | Vulnerable | | Moderate | | Extereme | |
|---|---|---|---|---|---|---|---|---|
| | ระบุ | ไม่ระบุ | ระบุ | ไม่ระบุ | ระบุ | ไม่ระบุ | ระบุ | ไม่ระบุ |
| Tablet | 458 | 1105 | 32 | 252 | 22 | 331 | 11 | 167 |
| Refrigirator | 1513 | 50 | 269 | 15 | 326 | 27 | 157 | 21 |
| Computer | 193 | 1370 | 11 | 273 | 5 | 348 | 3 | 175 |
| Television | 488 | 1075 | 60 | 224 | 68 | 285 | 23 | 155 |
| MobilePhone | 1530 | 33 | 262 | 22 | 326 | 27 | 166 | 12 |

จากตารางที่ 2 สามารถสรุปได้ว่า Refrigirator และ MobilePhone ไม่น่าจะสามารถนำมาใช้ในการระบุความยากจนได้เพราะ คนส่วนใหญ่มีใช้เป็นจำนวนมาก ทำให้อาจบอกได้ว่าเป็นสิ่งของที่ต้องใช้ในชีวิตประจำวัน จึงไม่สามารถที่จะแบ่งว่าคนยากจนหรือไม่ แต่ส่วน Tablet, Computer และTelevision ไม่ใช่สิ่งของที่คนจำนวนมากมี จึงสามารถบอกได้ว่าคนส่วนใหญ่ที่ไม่มีความเสี่ยงที่จะยากจนเท่านั้นที่มี จึงสามารถนำเอา Tablet, Computer และTelevision มาใช้ในการระบุความยากจนได้




รูปที่ 15 : ระดับการศึกษาของหัวหน้าครอบครัว

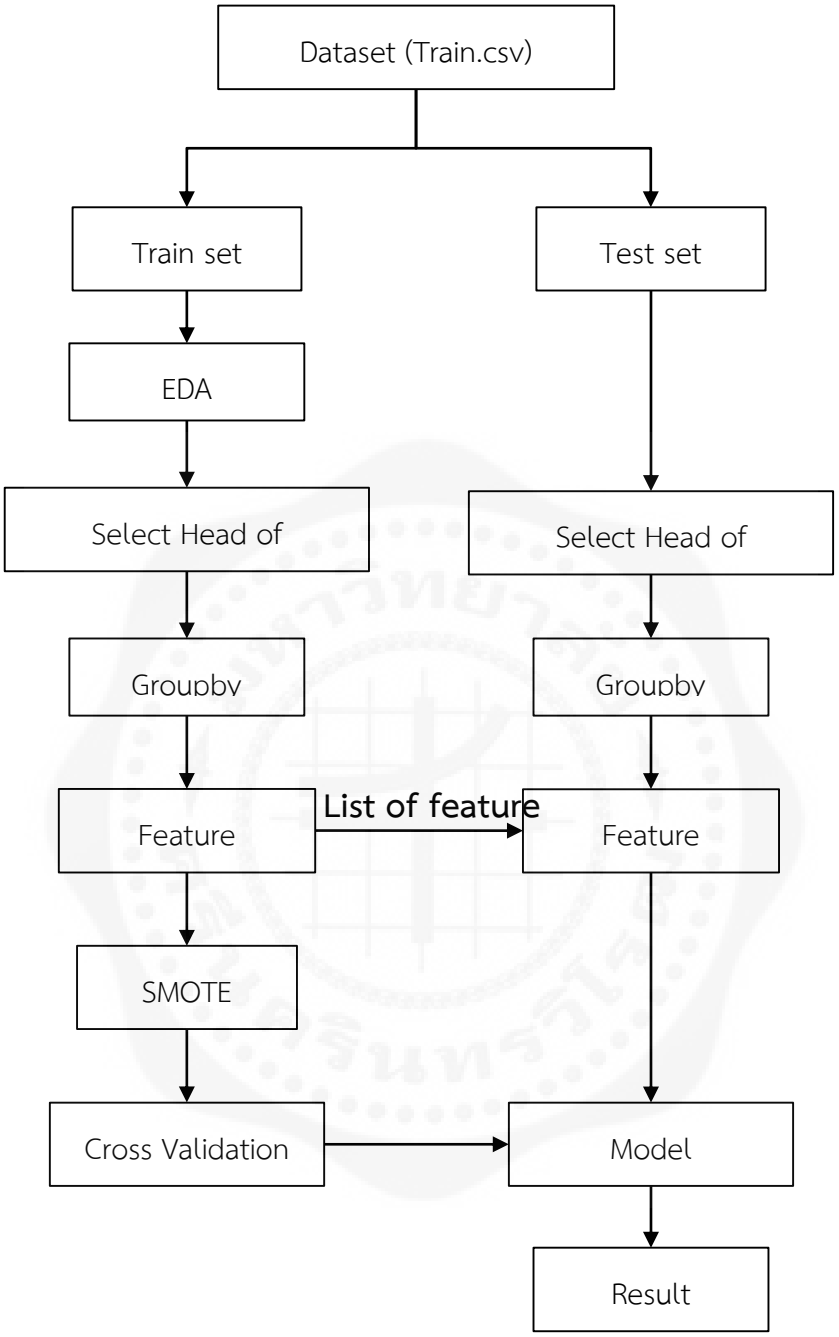
รูปที่ 15 เป็นรูปภาพที่บอกระดับการศึกษาของหัวหน้าครอบครัว จะเห็นได้ว่า แท่งแรก คือแท่งที่บอกถึงคนที่ไม่จบชั้นประถมศึกษา แท่งที่ 2 คือแท่งที่บอกถึงคนที่ไม่มีการศึกษา แท่งที่ 3 คือแท่งที่บอกถึงคนที่จบประถมศึกษา แท่งที่ 4 คือแท่งที่บอกถึงคนที่ไม่จบการศึกษาระดับมัธยมศึกษา แท่งที่ 5 คือแท่งที่บอกถึงคนที่จบการศึกษาระดับมัธยมศึกษา แท่งที่ 6 คือแท่งที่บอกถึงคนที่จบระดับปริญญาตรีและอุดมศึกษา แท่งที่ 7 คือแท่งที่บอกถึงคนที่ไม่จบมัธยมศึกษาทางเทคนิค แท่งที่ 8 คือแท่งที่บอกถึงคนที่จบมัธยมศึกษาทางเทคนิค แท่งที่ 9 คือแท่งที่บอกถึงคนที่มีการศึกษาระดับสูงกว่าปริญญาตรี สามารถสรุปได้เป็นตารางที่ 3

ตารางที่ 3 : แสดงระดับการศึกษาของแต่ละคลาส

| ระดับการศึกษา | NonVulnerable | Vulnerable | Moderate | Extereme |
|---|---|---|---|---|
| Incomplete Primary | 184 | 69 | 93 | 52 |
| Complete Primary | 398 | 104 | 124 | 43 |
| Incomplete Sc. | 270 | 43 | 62 | 42 |
| No Education | 28 | 27 | 31 | 17 |
| Complete Sc. | 223 | 25 | 32 | 15 |
| Undergraduation | 356 | 12 | 10 | 7 |
| Incomplete Tech Sc. | 11 | 4 | - | 1 |
| Complete Tech Sc. | 25 | - | - | 1 |
| Postgraduation | 67 | - | - | 1 |

จากตารางที่ 3 สามารถสรุปได้ว่า คนส่วนใหญ่ที่ไม่มีความเสี่ยงที่จะยากจนจะจบการศึกษาในระดับการศึกษาที่สูงกว่าปริญญาตรี (Postgraduation)

**3.3 วิธีการที่นำเสนอ**



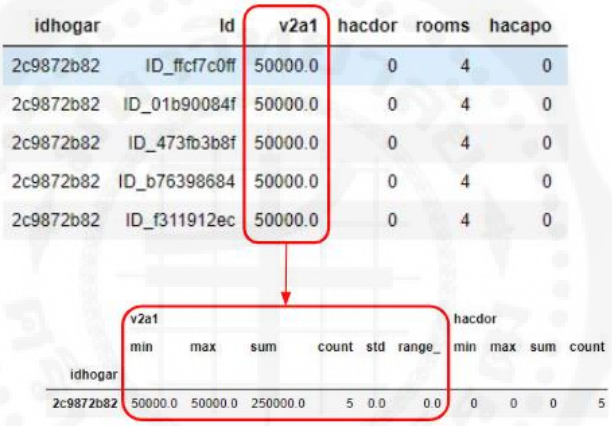

รูปที่ 16 : วิธีการดำเนินงาน

รูปที่ 16 แสดงถึงวิธีการทดลองเพื่อหาโมเดลการเรียนรู้ของเครื่องจักรที่เอามาใช้ในการระบุผู้ยากจน โดยในขั้นแรกผู้วิจัยแบ่งข้อมูลสำมะโนประชากรออกเป็นสองชุดคือ Train และ Test เป็นอัตราส่วน 80:20 โดย Train set มีข้อมูลจำนวน 7,645 แถว และ Test set มีข้อมูลจำนวน 1,912 แถว ต่อมานำข้อมูล Train มาทำมาทำ Exploratory Data Analysis เพื่อวิเคราะห์หาความสัมพันธ์ของข้อมูล จากนั้นนำข้อมูลแต่ละชุดมาหาหัวหน้า

ครอบครัว และทำการ feature engineering เพื่อรวมครัวเรือนเดียวกันเข้าไว้ด้วยกันโดยมีการทำงานของโค้ดดังรูปที่ 17 เพื่อหาค่าเฉลี่ยของแต่ละครัวเรือน ดังรูปที่ 18 เช่นเอา feature v2a1 (ค่าเช่ารายเดือน) มาทำการหาค่าเช่ารายเดือนต่ำสุดของครัวเรือน (min), ค่าเช่ารายเดือนสูงสุดของครัวเรือน (max), ผลรวมค่าเช่ารายเดือนของครัวเรือน (sum), ค่าเช่ารายเดือนเฉลี่ยของครัวเรือน (mean), ค่าเบี่ยงเบนของค่าเช่ารายเดือนของครัวเรือน (std), ผลต่างค่าเช่ารายเดือนสูงสุดและต่ำสุดของครัวเรือน (range) หลังจากกระบวนการนี้จำนวน feature จาก 143 เพิ่มขึ้นเป็น 828

```
#Groupby the household
range_ = lambda x: x.max() - x.min()
range_.__name__ = 'range_'
ind_agg_train = train.groupby('idhogar').agg(['min', 'max', 'sum', 'count', 'std', range_])
ind_agg_train.loc[['2c9872b82']]
```

รูปที่ 17 : โค้ดการทำ Groupby




รูปที่ 18: การ Groupby

จากนั้นนำข้อมูลในแต่ละส่วนที่ได้จากกระบวนการข้างต้นมาหาความสัมพันธ์ในรูปของสัมประสิทธิ์ความสัมพันธ์ (correlation coefficient) ดังแสดงในรูปที่ 18 จากรูปจะทำ Feature Selection เพื่อลดขนาดของคุณลักษณะเฉพาะ โดยคุณลักษณะเฉพาะที่มีค่าสัมประสิทธิ์ความสัมพันธ์เกิน 0.95 จะถูกตัดทิ้งออกไป ซึ่งหลังจากกระบวนการนี้คุณลักษณะเฉพาะจากทั้งหมด 828 ค่าจะถูกตัดเหลือเพียง 713

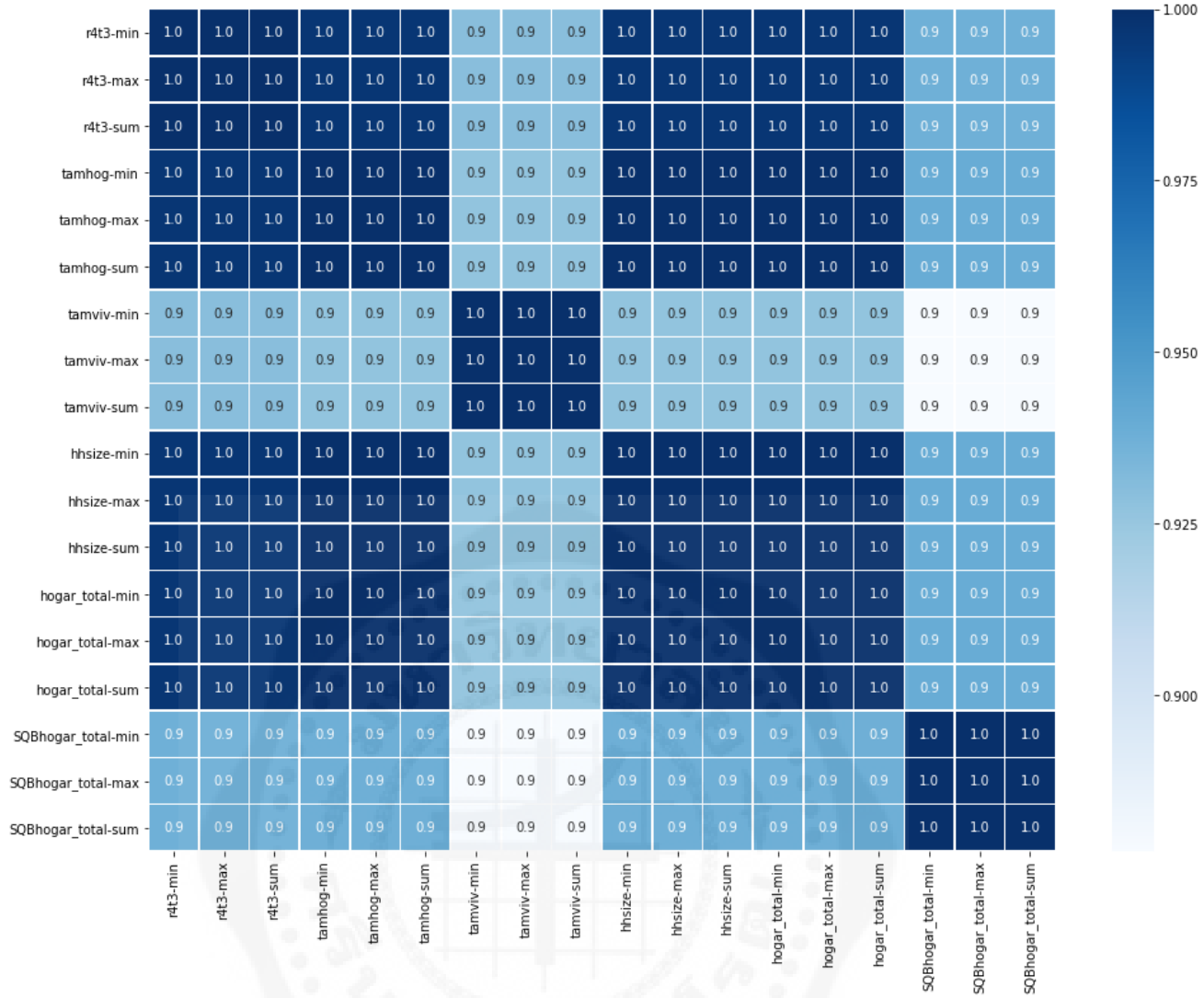
```
# Create correlation matrix
corr_matrix = ind_agg_train.corr()

# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))

# Find index of feature columns with correlation greater than 0.95
to_drop = [column for column in upper.columns if any(abs(upper[column]) > 0.95)]
```

```
fig = plt.figure(figsize=(16,12))
sns.heatmap(corr_matrix.loc[corr_matrix['tamhog-sum'].abs() > 0.9, corr_matrix['tamhog-sum'].abs() > 0.9],
            annot=True, cmap = 'Blues', fmt='.1f',linewidths=.5);
```

รูปที่ 19 : โค้ดการทำ Feature Selection




รูปที่ 20: Feature Selection

โดยในส่วนของ Test จะใช้ Feature ที่ได้จากการทำ Feature Selection ของ Train มาทำ Feature Selection ของ Test ต่อมา Train set จะเข้ากระบวนการสุ่มเพิ่มตัวอย่างกลุ่มน้อยโดยมีโค้ดการทำงานดังรูปที่ 21 เพื่อแก้ไขปัญหาข้อมูลที่ไม่สมดุลที่เกิดขึ้น โดยค่าก่อนทำวิธีการสุ่มเพิ่มตัวอย่างกลุ่มน้อยและหลังทำวิธีการสุ่มเพิ่มตัวอย่างกลุ่มน้อยดูได้จากตารางที่ 4 จะเห็นว่ากลุ่ม extreme poverty (คลาส 1) ของเดิมมี 178 และหลังจากสุ่มเพิ่มตัวอย่างกลุ่มน้อยแล้วมี 1563 กลุ่ม moderate poverty (คลาส 2) ของเดิมมี 284 และหลังจากสุ่มเพิ่มตัวอย่างกลุ่มน้อยแล้วมี 1563 กลุ่มvulnerable households (คลาส 3) ของเดิมมี 353 และหลังจากสุ่มเพิ่มตัวอย่างกลุ่มน้อยแล้วมี 1563 ส่วนกลุ่ม non vulnerable households (คลาส 4) มีเท่าเดิมคือ 1563 เนื่องจากเป็นกลุ่มที่มีจำนวนมากที่สุดจึงไม่ได้ถูกสุ่มเพิ่มแต่อย่างใด

```
sm = SMOTE(random_state=42)
X_train_res, y_train_res = sm.fit_sample(X_train, y_train.ravel())

print('After OverSampling, the shape of train_X: {}'.format(X_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_res.shape))

print('Resampled dataset shape %s' % Counter(y_train_res))
```

รูปที่ 21 : โค้ดการทำ SMOTE

ตารางที่ 4 : การทำวิธีการสุ่มเพิ่มตัวอย่างกลุ่มน้อยของแต่ละคลาส

| | **ก่อนทำวิธีการสุ่มเพิ่มตัวอย่างกลุ่มน้อย** | **หลังทำวิธีการสุ่มเพิ่มตัวอย่างกลุ่มน้อย** |
|---|---|---|
| Shape of train | (2378,713) | (6252,713) |
| non vulnerable households (class4) | 1563 | 1563 |
| vulnerable households (class3) | 353 | 1563 |
| moderate poverty (class2) | 284 | 1563 |
| extreme poverty (class1) | 178 | 1563 |

จากนั้นเข้าสู่กระบวนการ Cross-validation เพื่อใช้ในการทดสอบประสิทธิภาพของโมเดล โดยจะแบ่งกลุ่มทั้งหมด 10 ส่วนโดยที่แต่ละส่วนมีจำนวนข้อมูลเท่ากัน หลังจากนั้นข้อมูลหนึ่งส่วนจะใช้เป็นตัวทดสอบประสิทธิภาพของโมเดล ทำวนไปจนครบจำนวนที่แบ่งไว้และนำข้อมูลทั้งสองชุดที่ได้จาก train และ test เข้าสู่การทำโมเดลเพื่อวัดประสิทธิภาพของโมเดล โดยตารางที่ 5-9 แสดงตัวอย่างการทำ hyperparameter tuning ของโมเดล Random Forest โมเดล Gradient Boosting โมเดล LightGBM โมเดล XGBoost และโมเดล AdaBoost ตามลำดับ

ตารางที่ 5: Hyperparameter ของ Random Forest model

| Hyperparameter | Value | Hyperparameter | Value |
|---|---|---|---|
| bootstrap | True | min_impurity_split | None |
| class_weight | None | min_samples_leaf | 2,3 |
| criterion | 'gini' | min_samples_split | 2,3,4,5 |
| max_depth | 100,110,200,250 | min_weight_fraction_leaf | 0.0 |
| max_features | 5,10,20,30 | n_estimators | 900,1000,1500 |
| max_leaf_nodes | None | n_jobs | -1 |
| min_impurity_decrease | 0.0 | oob_score | False |
| verbose | 0 | warm_start | False |

จากตารางที่ 5 ค่า hyperparameter ที่จูนประกอบไปด้วย

max_depth มีค่าเท่ากับ 100

max_features มีค่าเท่ากับ 20

min_samples_leaf มีค่าเท่ากับ 2

min_samples_split มีค่าเท่ากับ 2

n_estimators มีค่าเท่ากับ 900

ตารางที่ 6: Hyperparameter ของ Gradient Boosting model

| Hyperparameter | Value | Hyperparameter | Value |
|---|---|---|---|
| criterion | 'friedman_mse' | min_impurity_split | None |
| init | None | min_samples_leaf | 1 |
| learning_rate | 0.05,0.1,0.5 | min_samples_split | 2 |
| loss | 'deviance' | min_weight_fraction_leaf | 0.0 |
| max_depth | 2,3,4,5 | n_estimators | 400,700 |
| max_features | None | min_impurity_decrease | 0.0 |
| max_leaf_nodes | 10,50,100 | tol | 0.0001 |
| n_iter_no_change | None | verbose | 0 |
| presort | 'auto' | warm_start | False |
| subsample | 1.0 | validation_fraction | 0.1 |
| random_state | 10 | | |

จากตารางที่ 6 ค่า hyperparameter ที่จูนประกอบไปด้วย

learning_rate มีค่าเท่ากับ 0.1

max_depth มีค่าเท่ากับ 4

max_leaf_nodes มีค่าเท่ากับ 100

n_estimators มีค่าเท่ากับ 700

ตารางที่ 7 : Hyperparameter ของ LightGBM model

| Hyperparameter | Value | Hyperparameter | Value |
|---|---|---|---|
| boosting_type | gbdt | class_weight | balanced |
| colsample_bytree | 1.0 | n_jobs | -1 |
| learning_rate | 0.01,0.05,0.1 | importance_type | split |
| min_split_gain | 0.0 | objective | None |
| max_depth | 5,10,20,30 | n_estimators | 100,400,500,600 |
| min_child_weight | 0.001 | random_state | 10 |
| min_child_samples | 10,20,24,25 | num_leaves | 20,30,40,49,50 |
| gamma | 0 | reg_lambda | 0.0 |
| scale_pos_weight | 1 | reg_alpha | 0.0 |
| subsample | 1.0 | subsample_freq | 0 |
| silent | True | subsample_for_bin | 200000 |

จากตารางที่ 7 ค่า hyperparameter ที่จูนประกอบไปด้วย

learning_rate มีค่าเท่ากับ 0.05

max_depth มีค่าเท่ากับ 30

min_child_samples มีค่าเท่ากับ 24

n_estimators มีค่าเท่ากับ 100

num_leaves มีค่าเท่ากับ 49

ตารางที่ 8 : Hyperparameter ของ XGBoost model

| Hyperparameter | Value | Hyperparameter | Value |
|---|---|---|---|
| base_score | 0.5 | max_delta_step | 0 |
| booster | 'gbtree' | n_jobs | -1 |
| learning_rate | 0.01,0.05,0.1 | nthread | None |
| colsample_bylevel | 1 | objective | 'multi : softprob' |
| max_depth | 5,10,20,30 | n_estimators | 400,500,600,700 |
| min_child_weight | 1 | random_state | 0 |
| missing | None | colsample_bytree | 1 |
| gamma | 0 | subsample | 1 |
| scale_pos_weight | 1 | reg_alpha | 0.2,0.3,0.4 |
| seed | None | reg_lambda | 1 |
| silent | True | | |

จากตารางที่ 8 ค่า hyperparameter ที่จูนประกอบไปด้วย

learning_rate มีค่าเท่ากับ 0.05

max_depth มีค่าเท่ากับ 10

n_estimators มีค่าเท่ากับ 400

reg_alpha มีค่าเท่ากับ 0.3

ตารางที่ 9 : Hyperparameter ของ AdaBoost model

| Hyperparameter | Value |
|---|---|
| **algorithm** | SAMME.R |
| **base_estimator** | None |
| **learning_rate** | 0.01,0.05,0.1 |
| **n_estimators** | 300,500,700,900 |
| **random_state** | 42 |

จากตารางที่ 9 ค่า hyperparameter ที่จูนประกอบไปด้วย

learning_rate มีค่าเท่ากับ 0.01

n_estimators มีค่าเท่ากับ 500

**3.4 แผนการดำเนินงาน**

จากตารางที่ 10 แสดงถึงแผนการดำเนินงานตั้งแต่การเลือกหัวข้อโครงการตลอดจนถึงการสรุปผลการดำเนินงาน

ตารางที่ 10: ตารางการดำเนินงาน

| ขั้นตอน | ปี | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 2561 | | | | | 2562 | | | | |
| | ส.ค. | ก.ย. | ต.ค. | พ.ย. | ธ.ค. | ม.ค. | ก.พ. | มี.ค. | เม.ย. | พ.ค. |
| 1) การวางแผนโครงการ | | | | | | | | | | |
| ประชุมเพื่อเลือกหัวข้อ | x | | | | | | | | | |
| ศึกษาโมเดลที่ใช้ในการวิเคราะห์ข้อมูล | | x | x | | | | | | | |
| 2) การวิเคราะห์ข้อมูล | | | | | | | | | | |
| วิเคราะห์ข้อมูล และ ทำ pre-processing | | | x | x | x | x | | | | |
| 3) การออกแบบโมเดล | | | | | | | | | | |
| กำหนดวัตถุประสงค์และขอบเขตของโมเดล | | | | | x | x | | | | |
| 4) การพัฒนาโมเดล | | | | | | | | | | |
| ทำการพัฒนาโมเดล | | | | | x | x | | | | |
| ทำการทดสอบและปรับปรุง | | | | | | | x | x | | |
| 5) สรุปและประเมินผล | | | | | | | | | | |
| ทำการปรับปรุงระบบอีกครั้ง | | | | | | | x | x | | |
| จัดทำคู่มือ และเอกสารที่เกี่ยวข้อง | | | | | | | | x | x | |
| สรุปผลการดำเนินงาน | | | | | | | | | x | |
| นำเสนอโมเดลที่สมบูรณ์ | | | | | | | | | | x |

## 3.5 อุปกรณ์และเครื่องมือที่ใช้พัฒนา

1. Hardware

   คอมพิวเตอร์

   - ระบบปฏิบัติการ: Windows 10 Pro [10] (64-bit)
   - หน่วยประมวลผล: Intel Core i7-4510U @ 2.60 GHz
   - หน่วยความจำ: RAM 8 GB

2. Software
   - Python3
   - Anaconda Navigator
   - Sklearn version 0.20.2

3. Server
   - ระบบปฏิบัติการ: IBM X System
   - หน่วยประมวลผล: xeon 32 cores
   - หน่วยความจำ: 128 GB

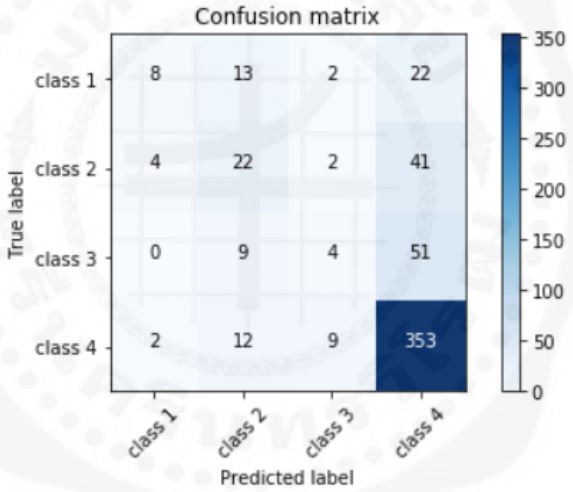
# บทที่ 4

# ผลการทดลอง

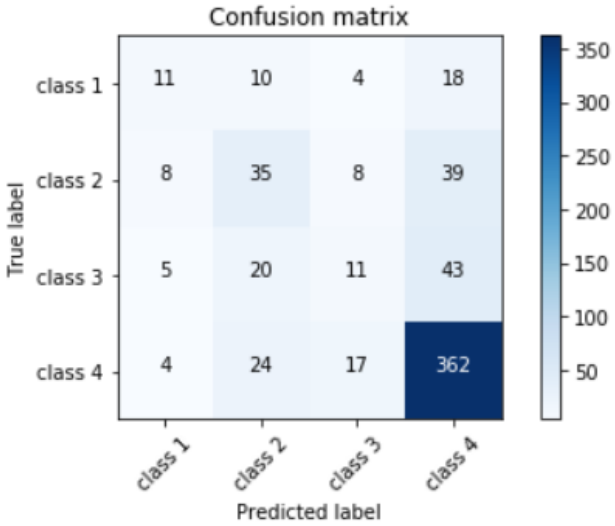
ในงานวิจัยชิ้นนี้ คณะผู้จัดทำได้ทำการทดสอบประสิทธิภาพของโมเดล โดยการเปรียบเทียบประสิทธิภาพของแต่ละโมเดล โดยใช้ตัวชี้วัด ได้แก่ Confusion Matrix, Macro F1 Score, Precision, Recall, Accuracy

## 4.1 ผลการทดสอบประสิทธิภาพของแต่ละโมเดล

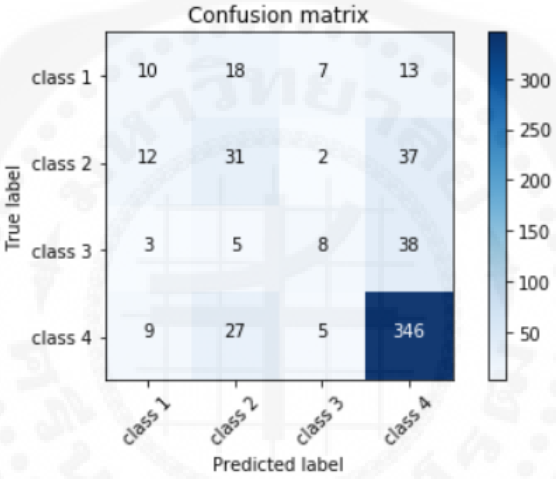
จากกระบวนการที่ได้กล่าวมาข้างต้นจะได้ค่า Confusion matrix ดังในรูปที่ 22-26 โดยรูปที่ 22 คือ Confusion matrix ของ Random Forest model รูปที่ 23 คือ Confusion matrix ของ Gradient Boosting model รูปที่ 24 เป็น Confusion matrix ของ LightGBM model รูปที่ 25 เป็น Confusion matrix ของ XGBoost model และรูปที่ 26 เป็น Confusion matrix ของ AdaBoost model ซึ่งจะเห็นได้ว่าค่า Confusion matrix ของ Gradient Boosting ในทุกๆคลาสมีความแม่นยำสูงกว่าโมเดลอื่นๆ



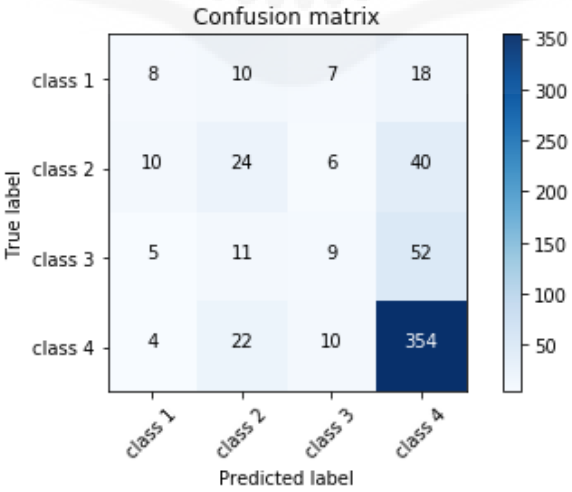

รูปที่ 22: Confusion matrix ของ Random Forest model



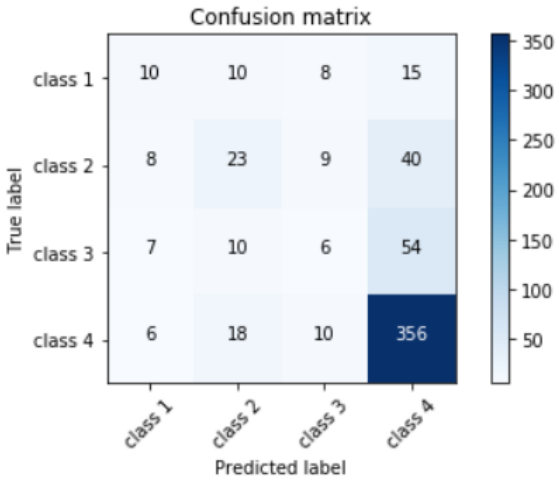

รูปที่ 23: Confusion matrix ของ Gradient Boosting model




รูปที่ 24 : Confusion matrix ของ LightGBM model




รูปที่ 25 : Confusion matrix ของ XGBoost model




รูปที่ 26 : Confusion matrix ของ AdaBoost model

**4.2 ผลการเปรียบเทียบประสิทธิภาพของโมเดลระหว่างโมเดลที่มีและไม่มีการทำ SMOTE**

ตารางที่ 11: การเปรียบเทียบประสิทธิภาพ

| | Groupby + SMOTE | Groupby + No SMOTE | No Groupby + SMOTE | No Groupby + No SMOTE | Baseline |
|---|---|---|---|---|---|
| 10 fold CVF1 | 0.87 | 0.32 | 0.88 | 0.35 | 0.34 |
| precision | 0.40 | 0.39 | 0.37 | 0.38 | - |
| recall | 0.36 | 0.31 | 0.34 | 0.32 | - |
| f1-score | **0.36** | 0.30 | 0.34 | 0.32 | - |
| Accuracy | 0.667 | 0.66 | 0.65 | 0.663 | - |

จากตารางที่ 11 คือการเปรียบเทียบประสิทธิภาพของ Random Forest model เนื่องจากต้องการรู้ว่าเทคนิคการสุ่มเพิ่มตัวอย่างกลุ่มน้อย (SMOTE) และการทำ Groupby มีผลต่อการวิเคราะหรือไม่ โดยจะเปรียบเทียบทั้งหมด 4 แบบ คือ

1. ทำ Groupby และ SMOTE
2. ทำ Groupby แต่ไม่ทำ SMOTE
3. ไม่ทำ Groupby แต่ทำ SMOTE
4. ไม่ทำ Groupby และ SMOTE

จากการทดลองพบว่าเทคนิคการสุ่มเพิ่มตัวอย่างกลุ่มน้อยมีส่วนช่วยในการเพิ่มประสิทธิภาพของโมเดลในการระบุความยากจน โดยการใช้โมเดล Random Forest ที่มีการสุ่มเพิ่มตัวอย่างกลุ่มน้อย (Random Forest) ให้ค่า 10 fold Cross validation F1-Score เท่ากับ 0.87 ค่าความถูกต้อง (Accuracy) เท่ากับ 66.7% ค่าความแม่นยำ (Precision) เท่ากับ 0.40 และค่ามาโคร F1 (macro F1) เฉลี่ยเท่ากับ 0.36 ซึ่งสูงกว่าโมเดลป่าสุ่มที่ไม่มีการสุ่มเพิ่มตัวอย่างกลุ่มน้อย (Baseline)

**4.3 ผลการเปรียบเทียบประสิทธิภาพของแต่ละโมเดล**

ตารางที่ 12: การเปรียบเทียบประสิทธิภาพแต่ละโมเดล

| | RandomForest | LightGBM | Gradient Boosting | AdaBoost | XGBoost | Baseline (RandomForest w/o SMOTE) |
|---|---|---|---|---|---|---|
| 10 fold CV F1-Score | 0.87 | 0.87 | 0.88 | 0.55 | 0.89 | 0.32 |
| precision | 0.40 | 0.43 | 0.46 | 0.40 | 0.42 | 0.39 |
| recall | 0.36 | 0.38 | 0.42 | 0.41 | 0.38 | 0.31 |
| F1-Score | 0.36 | 0.39 | **0.43** | 0.40 | 0.39 | 0.30 |
| Accuracy | 0.667 | 0.672 | 0.676 | 0.62 | 0.669 | 0.66 |

จากตารางที่ 12 พบว่าโมเดลการเรียนรู้ของเครื่องจักรแบบต้นไม้ชนิด Gradient Boosting มีประสิทธิภาพดีที่สุดในการทำนายความยากจนในระดับครัวเรือนเมื่อเทียบกับโมเดลอื่นๆ เช่น Random Forest, LightGBM, Adaboost และ XGBoost โดยให้ค่าความถูกต้อง (Accuracy) เท่ากับ 67.6% ค่าความแม่นยำ (Precision) เท่ากับ 0.46 และค่ามาโคร F1 (macro F1) เฉลี่ยเท่ากับ 0.43 ซึ่งมีค่าสูงกว่าโมเดลมาตรฐาน (Baseline) ซึ่งใช้โมเดลแบบ Random Forest และไม่มีการสุ่มเพิ่มตัวอย่างกลุ่มน้อย (SMOTE) ซึ่งได้ค่าความถูกต้องเท่ากับ 66% ค่าความแม่นยำเท่ากับ 0.39 และค่ามาโคร F1 เฉลี่ยเท่ากับ 0.30 และโมเดลที่มีประสิทธิภาพด้อยที่สุดเมื่อเทียบกับโมเดลอื่นๆคือ Random Forest โดยให้ค่าความถูกต้อง (Accuracy) เท่ากับ 66.7% ค่าความแม่นยำ (Precision) เท่ากับ 0.40 และค่ามาโคร F1 (macro F1) เฉลี่ยเท่ากับ 0.36

**4.4 ผลการแสดงคุณลักษณะที่สำคัญของแต่ละโมเดล**

จากตารางที่ 13-17 คือตารางที่แสดงถึงคุณลักษณะที่สำคัญ (Feature Importance) 3 ลำดับแรก ของแต่ละโมเดล โดยตารางที่ 13 คือ คุณลักษณะที่สำคัญสามลำดับแรกของโมเดล Radom Forest ตารางที่ 14 คือ คุณลักษณะที่สำคัญสามลำดับแรกของโมเดล Gradient Boosting ตารางที่ 15 คือ คุณลักษณะที่สำคัญสามลำดับแรกของโมเดล LightGBM ตารางที่ 16 คือ คุณลักษณะที่สำคัญสามลำดับแรกของโมเดล XGBoost และตารางที่ 17 คือ คุณลักษณะที่สำคัญสามลำดับแรกของโมเดล AdaBoost ซึ่งจะเห็นได้ว่า มี 3 คุณลักษณะที่สำคัญที่มีเหมือนกันมากที่สุดคือ อัตราส่วนของคนที่พึ่งพาอาศัยต่อคนที่ทำงาน, อายุเฉลี่ยของสมาชิกในบ้าน และปีการศึกษาโดยเฉลี่ยสำหรับคนที่อายุเกิน 18 ปี ซึ่งอาจจะบอกได้ว่าคุณลักษณะ 3 คุณลักษณะนี้อาจจะมีผลต่อการวิเคราะห์ความยากจน

ตารางที่ 13 : คุณลักษณะที่สำคัญสามลำดับแรกของโมเดล Radom Forest

| Random Forest Model | |
|---|---|
| **Feature** | **Importances** |
| อัตราส่วนของคนที่พึ่งพาอาศัยต่อคนที่ทำงาน | 0.0145 |
| อายุเฉลี่ยของสมาชิกในบ้าน | 0.0131 |
| ปีการศึกษาโดยเฉลี่ยสำหรับคนที่อายุเกิน 18 ปี | 0.0126 |

ตารางที่ 14 : คุณลักษณะที่สำคัญสามลำดับแรกของโมเดล Gradient Boosting

| Gradient Boosting Model | |
|---|---|
| **Feature** | **Importances** |
| จำนวนปีในสถานศึกษา | 0.0286 |
| คุณภาพของหลังคา | 0.0267 |
| อัตราส่วนของคนที่พึ่งพาอาศัยต่อคนที่ทำงาน | 0.0206 |

ตารางที่ 15 : คุณลักษณะที่สำคัญสามลำดับแรกของโมเดล LightGBM

| LightGBM Model | |
|---|---|
| **Feature** | **Importances** |
| อายุเฉลี่ยของสมาชิกในบ้าน | 0.0134 |
| ปีการศึกษาโดยเฉลี่ยสำหรับคนที่อายุเกิน 18 ปี | 0.0063 |
| ค่าเช่ารายเดือน | 0.0056 |

ตารางที่ 16: คุณลักษณะที่สำคัญสามลำดับแรกของโมเดล XGBoost

| XGBoost Model | |
|---|---|
| **Feature** | **Importances** |
| อายุเฉลียของสมาชิกในบ้าน | 0.068803 |
| ปีการศึกษาโดยเฉลียสำหรับคนทีอายุเกิน 18 ปี | 0.040040 |
| จำนวนห้องภายในบ้าน | 0.027688 |

ตารางที่ 17 : คุณลักษณะที่สำคัญสามลำดับแรกของโมเดล AdaBoost

| AdaBoost Model | |
|---|---|
| **Feature** | **Importances** |
| อัตราส่วนของคนทีพึงพาอาศัยต่อคนทีทำงาน | 0.084 |
| จำนวนคนทีอายุน้อยกว่า 12 ปี | 0.070 |
| หัวหน้าครอบครัวเพศชายทีได้รับการศึกษาตามเกณฑ์ | 0.048 |

# บทที่ 5
# สรุปผลการทดลอง

## 5.1 สรุปผลการดำเนินโครงงาน

งานวิจัยนี้ได้นำเสนอการใช้วิธีการเรียนรู้ของเครื่องสำหรับการทำนายระดับความยากจนในครัวเรือนโดยอาศัยการผสมผสานระหว่างกระบวนการปรับแต่งคุณลักษณะเฉพาะของข้อมูล, เทคนิคการสุ่มเพิ่มตัวอย่างกลุ่มน้อยและการใช้โมเดลการเรียนรู้ของเครื่องจักรแบบต้นไม้ในการวิเคราะห์ข้อมูลสำมะโนประชากร จากผลการทดลองพบว่าโมเดลแบบต้นไม้ชนิด Gradient Boosting มีประสิทธิภาพมากที่สุดในการทำนายความยากจนโดยมีค่า Accuracy = 0.68, precision = 0.46, recall = 0.42, F1-score = 0.43 (คะแนน F1-score ของ โมเดล baseline มีค่าเท่ากับ 0.32) โดยพบว่าสาเหตุหลักที่ส่งผลต่อความยากจนคือ คุณภาพของพื้นบ้าน อายุเฉลี่ยของครัวเรือนและจำนวนสมาชิกในครัวเรือน

## 5.2 ปัญหาและอุปสรรค

1. ข้อมูลที่ใช้มีปัญหาเรื่องข้อมูลไม่สมดุลกัน (Imbalanced Dataset)
   **การแก้ไข :** ใช้ SMOTE ในการทำให้ข้อมูลสมดุลกัน
2. ใช้เวลาในการ Tune Hyperparameters ในแต่ละ model นาน

## 5.3 ข้อเสนอแนะ

1. เนื่องจากผลลัพธ์ที่ได้ไม่ดีเท่าที่ควร ควรหาโมเดลอื่นเพิ่มในการวิเคราะห์
2. ลองปรับใช้กับข้อมูลสำมะโนครัวประชากรของประเทศไทย

# บรรณานุกรม


[1] Jessica E. Steele , Pal Roe Sundsoy , Carla Pezzulo , Victor A. Alegana , Tomas J. Bird , Joshua Blumenstock , Johannes Bjelland , Kenth Engo-Monsen , Yves-Alexandre de Montjoye , Asif M. Iqbal , Khandakar N. Hadiuzzaman , Xin Lu , Erik Wetter , Andrew J. Tatem and Linus Bengtsson, "Mapping poverty using mobile phone and satellite data," *The Royal Society Publishing,* p. 10, 2560.

[2] Michael Xie, Neal Jean, Marshall Burke, David Lobell, Stefano Ermon, "Transfer Learning from Deep Features for Remote Sensing and Poverty Mapping," Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, pp. 2923-3935, 2558.

[3] W. Koehrsen, "Costa Rican Household Poverty Level Prediction," สิงหาคม 2561. [Online]. Available: https://www.kaggle.com/c/costa-rican-household-poverty-prediction/data.

[4] M. T. Jones., "Data and analytics," 1 กุมภาพันธ์ 2561. [Online]. Available: https://www.ibm.com/developerworks/library/ba-intro-data-science-1/index.html.

[5] อ. ภิรมย์รื่น, "Big Data&Data Analytic," Auoychai, 5 เมษายน 2561. [Online]. Available: https://www.auoychai.com/big-datadata-analytic/. [Accessed 27 พฤศจิกายน 2561].

[6] V. Minaphinant, "Machine Learning," Finnomena, 28 กุมภาพันธ์ 2561. [Online]. Available: https://blog.finnomena.com/machine-learning-%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3-fa8bf6663c07?fbclid=IwAR1O5bedKA1YgPq4bhq-tKPG0JbIHzMMGUz9NsHIwj-mlgblmAABy9NPK_U. [Accessed 23 พฤษศจิกายน 2561].

[7] Siraunz, “Ensemble method,” 7 กันยายน 2558. [Online]. Available: http://analyticsth.blogspot.com/2015/09/ensemble-method.html

[8] ปรเมษฐ์ ธันวานนท์ และคณะ, "Predict stock price trends in Stock Exchange of Thailand using," JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY, p. 12, มิถุนายน 2559.

[9] INRIA, "scikit-learn Machine Learning in Python," ธันวาคม 2561. [Online]. Available: https://scikit-learn.org/stable/index.html [Accessed 7 มกราคม 2562]

[10] TITIPATA, "Bagging-Boosting," 21 สิงหาคม 2561. [Online]. Available: https://tupleblog.github.io/bagging-boosting/.

[11] Gabriel Tseng, “Gradient Boosting and XGBoost,” 13 เมษายน 2561.[Online]. Available: https://medium.com/@gabrieltseng/gradient-boosting-and-xgboost-c306c1bcfaf5

[12] Chaipat N, “Adaptive Boosting Algorithm,” 24 พฤษภาคม 2560. [Online]. Available: http://cway-quantlab.blogspot.com/2017/05/adaptive-boosting-algorithm-1.html

[13] ML Blog Team, “Lessons Learned From Benchmarking Fast Machine Learning Algorithms,” 25 กรกฎาคม 2560. [Online]. Available: https://blogs.technet.microsoft.com/machinelearning/2017/07/25/lessons-learned-benchmarking-fast-machine-learning-algorithms/?fbclid=IwAR0S9bi4FHZLsH3QCJNLvsPL10KH8ogODju32L_fTJOwz5k3Dhts_I3ZLdc

[14] P. Palwisut, "Improving Decision Tree Technique in Imbalanced Data Sets," Information Technology Journal, pp. 54-63, มิถุนายน 2559.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-Sampling Technique," Journal of Artificial Intelligent Research, pp. 321-357, 2002.

[16] อวยชัย ภิรมย์รื่น, “การวัดประสิทธิภาพ Model ของ Machine Learning แบบ Classification,” 5 เมษายน 2561. [Online]. Available: https://www.auoychai.com/big-datadata-analytic/%E0%B8%AA%E0%B8%A3%E0%B8%B8%E0%B8%9B%E0%B9%80%E0%B8%A3%E0%B8%B7%E0%B9%88%E0%B8%AD%E0%B8%87%E0%B9%80%E0%B8%81%E0%B8%B5%E0%B9%88%E0%B8%A2%E0%B8%A7%E0%B8%81%E0%B8%B1%E0%B8%9A%E0%B8%81%E0%B8%B2%E0%B8%A3/

[17] "Pover-T Tests: Predicting Poverty," DRIVENDATA, 28 กุมภาพันธ์ 2561. [Online]. Available: https://www.drivendata.org/competitions/50/worldbank-poverty-prediction/page/97/. [Accessed 23 พฤษศจิกายน 2561].

[18] V. Shukla, "How to handle imbalanced datasets," 29 กรกฎาคม 2561. [Online]. Available: https://blog.goodaudience.com/how-to-handle-imbalanced-datasets-d67176dfb0aa.

# ภาคผนวก ก.

## การติดตั้ง packages

**ติดตั้ง pandas**

pip install pandas

**ติดตั้ง scikit-learn**

pip install -U scikit-learn

**ติดตั้ง numpy**

pip install numpy

**ติดตั้ง plotly**

pip install plotly

**ติดตั้ง lightgbm**

pip install lightgbm

**ติดตั้ง seaborn**

pip install seaborn

**ติดตั้ง random**

pip install random

**ติดตั้ง matplotlib**

pip install matplotlib

**ติดตั้ง collections**

pip install collections

**ติดตั้ง imbalanced-learn**

pip install -U imbalanced-learn

**ติดตั้ง xgboost**

pip install xgboost

# ภาคผนวก ข.

## โค้ดที่ใช้งาน

**ชื่อไฟล์ : EDA**

**ภาษาที่ใช้ : Python**

**คำอธิบาย : วิเคราะห์ข้อมูลเพื่อสรุปลักษณะสำคัญของข้อมูล และแสดงเป็นรูปภาพหรือกราฟ**

```
# อ่านข้อมูล
df = pd.read_csv('train.csv')
df.head()

#อ่านรายละเอียด
df.info()

#อ่านคอลัมน์
df.columns

#หา missing data
missing_data = pd.DataFrame({'total_missing': df.isnull().sum(), 'perc_missing':
    (df.isnull().sum()/9557)*100})
missing_data = missing_data.sort_values(by=['total_missing'], ascending = False)
missing_data.head()

#แก้ไขปัญหา missing data โดยใช้ fillna() และตรวจหา missing data
df = df.fillna(0)
print ("Top Columns having missing values")
missmap = df.isnull().sum().to_frame()
missmap = missmap.sort_values(0, ascending = False)
missmap.head()

#หา Outlier
num_cols = [ 'v2a1','SQBescolari', 'SQBage', 'SQBhogar_total', 'SQBedjefe',
        'SQBhogar_nin']
```

```
plt.figure(figsize=(18,9))
df[num_cols].boxplot()
plt.title("Outlier", fontsize=20)
plt.show()

#แสดงกราฟข้อมูลทั้งหมด
target = df['Target'].value_counts().to_frame()
levels = ["NonVulnerable", "Moderate Poverty", "Vulnerable", "Extereme Poverty"]
trace = go.Bar(y=target.Target, x=levels, marker=dict(color='orange', opacity=0.6))
layout = dict(title="Household Poverty Levels", margin=dict(l=200), width=800,
    height=400)
data = [trace]
fig = go.Figure(data=data, layout=layout)
iplot(fig)

#แยกข้อมูลเป็นชุดtrain และชุดtest
y = df['Target']
train, test = train_test_split(df, test_size=0.2,stratify=y)
print(train.shape)
print(test.shape)

#แสดงกราฟข้อมูลทั้งหมดของชุดtrain
target = train['Target'].value_counts().to_frame()
levels = ["NonVulnerable", "Moderate Poverty", "Vulnerable", "Extereme Poverty"]
trace = go.Bar(y=target.Target, x=levels, marker=dict(color='red', opacity=0.6))
layout = dict(title="Household Poverty Levels in Train Data", margin=dict(l=200),
    width=800, height=400)
data = [trace]
fig = go.Figure(data=data, layout=layout)
iplot(fig)
 #แสดงกราฟสินทรัพย์ในครัวเรือน
def compare_plot(col, title):
```

```
    tr1 = train[train['Target'] == 1][col].value_counts().to_dict()
    tr2 = train[train['Target'] == 2][col].value_counts().to_dict()
    tr3 = train[train['Target'] == 3][col].value_counts().to_dict()
    tr4 = train[train['Target'] == 4][col].value_counts().to_dict()
    levels = ['Extereme', 'Moderate', 'Vulnerable', 'NonVulnerable']
    trace1 = go.Bar(y=[tr1[0], tr2[0], tr3[0], tr4[0]], name="Not Present", x=levels,
     marker=dict(color="orange", opacity=0.6))
    trace2 = go.Bar(y=[tr1[1], tr2[1], tr3[1], tr4[1]], name="Present", x=levels,
     marker=dict(color="purple", opacity=0.6))
    return trace1, trace2
tr1, tr2 = compare_plot("v18q", "Tablet")
tr3, tr4 = compare_plot("refrig", "Refrigirator")
tr5, tr6 = compare_plot("computer", "Computer")
tr7, tr8 = compare_plot("television", "Television")
tr9, tr10 = compare_plot("mobilephone", "MobilePhone")
titles = ["Tablet", "Refrigirator", "Computer", "Television", "MobilePhone"]
fig = tools.make_subplots(rows=3, cols=2, print_grid=False, subplot_titles=titles)
fig.append_trace(tr1, 1, 1)
fig.append_trace(tr2, 1, 1)
fig.append_trace(tr3, 1, 2)
fig.append_trace(tr4, 1, 2)
fig.append_trace(tr5, 2, 1)
fig.append_trace(tr6, 2, 1)
fig.append_trace(tr7, 2, 2)
fig.append_trace(tr8, 2, 2)
fig.append_trace(tr9, 3, 1)
fig.append_trace(tr10, 3, 1)

fig['layout'].update(height=1000, title="What do Households Own", barmode="stack",
    showlegend=False)
iplot(fig)
```

```
#แสดงกราฟการศึกษาในครัวเรือน และกราฟสถานะของสมาชิกในครอบครัว
def find_prominent(row, mats):
    for c in mats:
        if row[c] == 1:
            return c
    return
def combine2(starter, colname, title, replacemap, plotme = True):
    mats = [c for c in train.columns if c.startswith(starter)]
    train[colname] = train.apply(lambda row : find_prominent(row, mats), axis=1)
    train[colname] = train[colname].apply(lambda x : replacemap[x] if x != None else x )
    om1 = train[train['Target'] == 1][colname].value_counts().to_frame()
    om2 = train[train['Target'] == 2][colname].value_counts().to_frame()
    om3 = train[train['Target'] == 3][colname].value_counts().to_frame()
    om4 = train[train['Target'] == 4][colname].value_counts().to_frame()
    trace1 = go.Bar(y=om1[colname], x=om1.index, name="Extereme",
     marker=dict(color='red', opacity=0.9))
    trace2 = go.Bar(y=om2[colname], x=om2.index, name="Moderate",
     marker=dict(color='red', opacity=0.5))
    trace3 = go.Bar(y=om3[colname], x=om3.index, name="Vulnerable",
     marker=dict(color='orange', opacity=0.9))
    trace4 = go.Bar(y=om4[colname], x=om4.index, name="NonVulnerable",
     marker=dict(color='orange', opacity=0.5))
    data = [trace1, trace2, trace3, trace4]
    layout = dict(title=title, legend=dict(y=1.1, orientation="h"), barmode="stack",
     margin=dict(l=50), height=400)
    fig = go.Figure(data=data, layout=layout)
    if plotme:
        iplot(fig)
flr = {"instlevel1": "No Education", "instlevel2": "Incomplete Primary", "instlevel3":
     "Complete Primary",
       "instlevel4": "Incomplete Sc.", "instlevel5": "Complete Sc.", "instlevel6": "Incomplete
     Tech Sc.",
```

```
        "instlevel7": "Complete Tech Sc.", "instlevel8": "Undergraduation", "instlevel9":
      "Postgraduation"}
combine2("instl", "education_details", "Education Details of Family Members", flr)

flr = {"estadocivil1": "< 10 years", "estadocivil2": "Free / Coupled union", "estadocivil3":
      "Married",
        "estadocivil4": "Divorced", "estadocivil5": "Separated", "estadocivil6": "Widow",
        "estadocivil7": "Single"}
combine2("estado", "status_members", "Status of Family Members", flr)
flr = {"lugar1": "Central", "lugar2": "Chorotega", "lugar3": "PacÃƒÂ-fico central",
        "lugar4": "Brunca", "lugar5": "Huetar AtlÃƒÂ¡ntica", "lugar6": "Huetar Norte"}
combine2("lugar", "region", "Region of the Households", flr, plotme=False)

#กราฟแสดงเพศและอายุ
def agbr(col):
    temp1 = train[train['Target'] == 1][col].value_counts()
    trace1 = go.Bar(x=temp1.index, y=temp1.values, marker=dict(color="red",
     opacity=0.89), name="Extereme")
    temp2 = train[train['Target'] == 2][col].value_counts()
    trace2 = go.Bar(x=temp2.index, y=temp2.values, marker=dict(color="orange",
     opacity=0.79), name="Moderate")
    temp3 = train[train['Target'] == 3][col].value_counts()
    trace3 = go.Bar(x=temp3.index, y=temp3.values, marker=dict(color="purple",
     opacity=0.89), name="Vulnerable")
    temp4 = train[train['Target'] == 4][col].value_counts()
    trace4 = go.Bar(x=temp4.index, y=temp4.values, marker=dict(color="green",
     opacity=0.79), name="NonVulnerable")
    return [trace1, trace2, trace3, trace4]
    layout = dict(height=400)
    fig = go.Figure(data=[trace1, trace2, trace3, trace4], layout=layout)
    iplot(fig)
```

```
titles = ["Total Persons", "< 12 Yrs", ">= 12 Yrs", "Total Males", "Males < 12 Yrs", "Males
    >= 12 Yrs", "Total Females", "Females < 12 Yrs", "Females >= 12 Yrs"]
fig = tools.make_subplots(rows=3, cols=3, print_grid=False, subplot_titles=titles)
res = agbr('r4t1')
for x in res:
    fig.append_trace(x, 1, 1)
res = agbr('r4t2')
for x in res:
    fig.append_trace(x, 1, 2)
res = agbr('r4t3')
for x in res:
    fig.append_trace(x, 1, 3)
res = agbr('r4h1')
for x in res:
    fig.append_trace(x, 2, 1)
res = agbr('r4h2')
for x in res:
    fig.append_trace(x, 2, 2)
res = agbr('r4h3')
for x in res:
    fig.append_trace(x, 2, 3)
res = agbr('r4m1')
for x in res:
    fig.append_trace(x, 3, 1)
res = agbr('r4m2')
for x in res:
    fig.append_trace(x, 3, 2)
res = agbr('r4m3')
for x in res:
    fig.append_trace(x, 3, 3)
fig['layout'].update(height=900, showlegend=False, title="Gender and Age Distributions")
iplot(fig)
```

**ชื่อไฟล์ : Project**

**ภาษาที่ใช้ : Python**

**คำอธิบาย : นำข้อมูลที่วิเคราะห์แล้วเข้าสู่โมเดลต่างๆเพื่อเปรียบเทียบประสิทธิภาพของแต่ละโมเดล**

**การเตรียมข้อมูล**

```
# อ่านข้อมูล
df = pd.read_csv('train.csv')

#แยกข้อมูลเป็นชุดtrain และชุดtest
train, test= train_test_split(df, test_size=0.2,stratify=y)

#ทำการเลือกแต่หัวหน้าครอบครัว
heads_train = train.loc[train['parentesco1'] == 1].copy()

# สร้างฟังก์ชัน range
range_ = lambda x: x.max() - x.min()
range_.__name__ = 'range_'

# ทำการ GroupBy
ind_agg_train = heads_train.groupby('idhogar').agg(['min', 'max', 'sum', 'count', 'std',
    range_])

# เปลี่ยนชื่อคอลัมน์
new_col = []
for c in ind_agg_train.columns.levels[0]:
    for stat in ind_agg_train.columns.levels[1]:
        new_col.append(f'{c}-{stat}')

ind_agg_train.columns = new_col
ind_agg_train.head()

# สร้าง correlation matrix
corr_matrix = ind_agg_train.corr()
```

```
# เลือก upper triangle ของ correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))

# หาคอลัมน์ที่มีค่า correlation มากกว่า 0.95
to_drop = [column for column in upper.columns if any(abs(upper[column]) > 0.95)]
print(f'There are {len(to_drop)} correlated columns to remove.')

# ลบคอลัมน์ที่มีค่า correlation มากกว่า 0.95
ind_heads_train = ind_agg_train.drop(columns = to_drop)
# ทำการรวม household id
final_train = heads_train.merge(ind_heads_train, on = 'idhogar', how = 'left')
#ทำการจับคู่ข้อมูลให้เป็นถ้า yes ให้เป็น 1 no ให้เป็น 0
mapping = {"yes": 1, "no": 0}
# ใช้ operation เดียวกันทั้ง train และ test
for df in [X_train,X_test]:
    # เติมค่าด้วยการจับคู่ให้ถูกต้อง
    df['dependency'] = df['dependency'].replace(mapping).astype(np.float64)
    df['edjefa'] = df['edjefa'].replace(mapping).astype(np.float64)
    df['edjefe'] = df['edjefe'].replace(mapping).astype(np.float64)
X_train[['dependency', 'edjefa', 'edjefe']].describe()

#แก้ไขปัญหา missing data โดยใช้ fillna()
X_train = X_train.fillna(0)
X_test = X_test.fillna(0)

#ทำการ SMOTE ให้ข้อมูลเท่ากัน
sm = SMOTE(random_state=42)
X_train_res, y_train_res = sm.fit_sample(X_train, y_train.ravel())
print('After OverSampling, the shape of train_X: {}'.format(X_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_res.shape))
print('Resampled dataset shape %s' % Counter(y_train_res))
```

```
#กำหนดตัวแปรค่า f1_score
scorer = make_scorer(f1_score, greater_is_better=True, average = 'macro')
```

**การ hyperparameter tuning**

```
# ทำการhyperparameter tuning โดยใช้ GridSearchCV
from sklearn.model_selection import GridSearchCV
# Create the parameter grid based on the results of random search
param_grid = {
    'bootstrap': [True],
    'max_depth': [100,110,200,250],
    'max_features': [5,10,20,30],
    'min_samples_leaf': [2, 3],
    'min_samples_split': [2,3,4,5],
    'n_estimators': [900,1000,1500]
}
# สร้าง based model
rf = RandomForestClassifier()
# ทำ grid search เพื่อหาค่าพารามิเตอร์ที่เหมาะสมที่สุด
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                           cv = 10, n_jobs = -1, verbose = 2)
grid_search.fit(X_train_res,y_train_res)
grid_search.best_params_
```

**โมเดลที่ใช้**

```
#โมเดล RandomForest
model1 =  RandomForestClassifier(bootstrap= True,max_depth=100,
max_features=20,min_samples_split=2,min_samples_leaf = 2, n_estimators=900,
random_state=10, n_jobs = -1)

#โมเดล LightGBM
model2 = lgb.LGBMClassifier(max_depth= 30,min_child_samples=24,
num_leaves=49,learning_rate=0.05,class_weight = 'balanced', random_state = 10)
```

```
#โมเดล GradientBoosting
 model3 = GradientBoostingClassifier(learning_rate= 0.1, max_depth = 10,
 n_estimators=700, random_state=10)

#โมเดล AdaBoost
model4 = AdaBoostClassifier(learning_rate=0.01,n_estimators=500, random_state=42)

#โมเดล XGBoost
model5 = xgb.XGBClassifier(max_depth=10, learning_rate=0.05,reg_alpha=0.3,
n_jobs=-1, n_estimators=400)

# fit โมเดล
model1.fit(X_train_res, y_train_res)
# predict โมเดล
preds = model1.predict(X_test_res)
```

**การวัดประสิทธิภาพของแต่ละโมเดล**

```
# หา 10 fold cross validation
cv_score = cross_val_score(model1, X_train_res, y_train_res, cv = 10, scoring = scorer)
print(f'10 Fold Cross Validation F1 Score = {round(cv_score.mean(), 4)} with std =
{round(cv_score.std(), 4)}')

#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, predictions))

from sklearn.metrics import confusion_matrix
cnf_matrix = confusion_matrix(y_test, predictions)

from sklearn.metrics import classification_report
target_names = ['class 1', 'class 2', 'class 3','class 4']
```

```
print(classification_report(y_test, predictions, target_names=target_names))


#หา feature_importances
features = list(X_train.columns)
feature_importances = pd.DataFrame({'feature': features,

    'importance':model1.feature_importances_}).sort_values('importance',
                        ascending=False)
feature_importances.head(20)
```