การทำนายเบี้ยประกันสุขภาพ: การศึกษาการวิเคราะห์ด้วยอัลกอริทึมการเรียนรู้ของเครื่อง

HEALTH INSURANCE PREMIUM PREDICTION: AN ANALYTICAL STUDY USING MACHINE LEARNING ALGORITHMS

จิดาภา อาดัม

บัณฑิตวิทยาลัย มหาวิทยาลัยศรีนครินทรวิโรฒ

2567

การทำนายเบี้ยประกันสุขภาพ: การศึกษาการวิเคราะห์ด้วยอัลกอริทึมการเรียนรู้ของเครื่อง

จิดาภา อาดัม

สารนิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการข้อมูล
คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ
ปีการศึกษา 2567

HEALTH INSURANCE PREMIUM PREDICTION: AN ANALYTICAL STUDY USING MACHINE LEARNING ALGORITHMS

JIDAPA ADAM

An Master's Project Submitted in Partial Fulfillment of the Requirements

for the Degree of MASTER OF SCIENCE

(Data Science)

Faculty of Science, Srinakharinwirot University

2024

สารนิพนธ์

เรื่อง

การทำนายเบี้ยประกันสุขภาพ: การศึกษาการวิเคราะห์ด้วยอัลกอริทึมการเรียนรู้ของเครื่อง

ของ

จิดาภา อาดัม

ได้รับอนุมัติจากบัณฑิตวิทยาลัยให้นับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการข้อมูล

ของมหาวิทยาลัยศรีนครินทรวิโรฒ

(รองศาสตราจารย์ นายแพทย์ฉัตรชัย เอกปัญญาสกุล)

คณบดีบัณฑิตวิทยาลัย

คณะกรรมการสอบปากเปล่าสารนิพนธ์

..... ที่ปรึกษาหลัก

(ผู้ช่วยศาสตราจารย์ ดร.วราภรณ์ วิยานนท์)

..... ประธาน

(ผู้ช่วยศาสตราจารย์ ดร.จันตรี ผลประเสริฐ)

..... กรรมการ

(อาจารย์ ดร.เรืองศักดิ์ ตระกูลพุทธิรักษ์)

ชื่อเรื่อง	การทำนายเบี้ยประกันสุขภาพ: การศึกษาการวิเคราะห์ด้วยอัลกอริทึมการเรียนรู้ของเครื่อง
ผู้วิจัย	จิดาภา อาดัม
ปริญญา	วิทยาศาสตรมหาบัณฑิต
ปีการศึกษา	2567
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร. วราภรณ์ วิยานนท์

งานวิจัยนี้มีวัตถุประสงค์เพื่อพัฒนาและเปรียบเทียบแบบจำลองการเรียนรู้ของเครื่องสำหรับการทำนายเบี้ยประกันสุขภาพ โดยใช้ข้อมูลลักษณะทางกายภาพคือ อายุ สภาวะของโรคเบาหวาน ปัญหาเรื่องความดันโลหิต การปลูกถ่ายอวัยวะ มีโรคเรื้อรังหรือไม่ ส่วนสูง น้ำหนัก มีการแพ้หรือไม่ ประวัติการเป็นมะเร็งของครอบครัว และจำนวนครั้งการผ่าตัด โดยเลือกใช้แบบจำลอง 6 ประเภท ได้แก่ Support Vector Regression (SVR), Lasso Regression, Ridge Regression, Decision Tree, Random Forest และ XGBoost พร้อมปรับจูนไฮเปอร์พารามิเตอร์เพื่อเพิ่มประสิทธิภาพด้วยวิธี GridSearchCV ผลการทดลองพบว่า Random Forest ให้ผลลัพธ์ดีที่สุด โดยมีค่า MAPE และ RMSE ต่ำสุด และค่า R^2 สูงสุด แสดงถึงความสามารถในการอธิบายข้อมูลได้ดีที่สุด นอกจากนี้ ตัวแปรที่มีผลต่อแบบจำลองมากที่สุดในทุกกรณีคือ อายุ ซึ่งสอดคล้องกับสมมติฐาน อย่างไรก็ตาม โมเดลประเภท Tree-based จะให้ประสิทธิภาพในการทำนายเบี้ยประกันสุขภาพสูงกว่าโมเดลเชิงเส้น (Linear Models) เนื่องจากข้อมูลเบี้ยประกันมีลักษณะความสัมพันธ์แบบไม่เชิงเส้น (Nonlinear Relationship) และมีปฏิสัมพันธ์ระหว่างตัวแปร (Feature Interaction) งานวิจัยนี้สามารถนำไปต่อยอดโดยใช้ชุดข้อมูลหรือแบบจำลองอื่นเพิ่มเติม และสามารถประยุกต์ใช้ในภาคธุรกิจประกันสุขภาพเพื่อเพิ่มประสิทธิภาพในการคำนวณเบี้ยประกันได้ในอนาคต

คำสำคัญ : การทำนายเบี้ยประกันสุขภาพ, การเรียนรู้ของเครื่อง, การจูนไฮเปอร์พารามิเตอร์

Title	HEALTH INSURANCE PREMIUM PREDICTION: AN ANALYTICAL STUDY USING MACHINE LEARNING ALGORITHMS
Author	JIDAPA ADAM
Degree	MASTER OF SCIENCE
Academic Year	2024
Thesis Advisor	Assistant Professor Waraporn Viyanon , Ph.D.

This research aims to develop and compare machine learning models for predicting health insurance premiums based on physical characteristics are age, Diabetes, BloodPressureProblems, AnyTransplants, AnyChronicDiseases, Height, Weight, KnowAllergies, HistoryOfCancerinFamily and NumberOfMajorSurgeries. Six models were investigated: Support Vector Regression (SVR), Lasso Regression, Ridge Regression, Decision Tree, Random Forest, and XGBoost. Hyperparameter tuning was applied to optimize each model's performance by using GridSearchCV. The experimental results indicate that the Random Forest model yielded the best performance, achieving the lowest MAPE and RMSE, as well as the highest R-squared (R^2) value, thereby demonstrating its superior ability to capture and explain the variance in the data. Furthermore, age was identified as the most influential feature in predicting insurance premiums across all models, aligning with the initial hypothesis. Tree-based models showed better prediction performance than linear models, due to the nonlinear relationships in insurance premium data and interactions among features. This study provides a foundation for further research using alternative datasets or more advanced models and holds potential for practical applications in the health insurance industry to improve the accuracy and efficiency of premium estimation.

Keyword : Health Insurance Premium Prediction, Machine Learning, Hyperparameter Tuning

กิตติกรรมประกาศ

การค้นคว้าอิสระนี้สำเร็จลุล่วงได้ด้วยความกรุณาจาก ผศ.ดร.วราภรณ์ วิยานนท์ ผู้เป็นอาจารย์ที่ปรึกษาการค้นคว้าอิสระที่คอยให้คำแนะนำ ปรึกษา เสนอแนวคิดในการแก้ไขปัญหาของการค้นคว้าอิสระ ตลอดจนข้อบกพร่องต่างๆ ผู้ศึกษาจึงขอกราบขอบพระคุณเป็นอย่างสูง ขอขอบคุณ คณะวิทยาศาสตร์ ภาควิชาวิทยาการคอมพิวเตอร์ ที่ให้โอกาสและสนับสนุนทรัพยากรต่าง ๆ ในการดำเนินการศึกษาในครั้งนี้

ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ และ ผู้ปกครอง ที่คอยเป็นกำลังใจให้เสมอตลอดการทำการค้นคว้าอิสระนี้และคอยสนับสนุนจนการค้นคว้าอิสระนี้สำเร็จลุล่วงได้ด้วยดี และขอบคุณเพื่อน ๆ ทุกคน ที่คอยเป็นกำลังใจให้เสมอ

จิดาภา อาดัม

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูปภาพ

หน้า

บทที่ 1
บทนำ

1.1 ภูมิหลัง

ประกันสุขภาพถือได้ว่าเป็นหนึ่งในประกันภัยที่มีความต้องการมากที่สุดในโลกเนื่องจากการเจ็บป่วยที่อาจจะเกิดขึ้นตอนไหนก็ได้ในช่วงวัยของชีวิตเป็นความเสี่ยงในรูปแบบหนึ่งที่อาจจะมาพร้อมกับค่าใช้จ่ายมหาศาล จากข้อมูลจาก จะเห็นได้ว่าตลาดการเติบโตของประกันสุขภาพนั้นมีการเพิ่มขึ้นๆในทุกๆปี นับตั้งแต่จากการเกิดโรคระบาดของโควิดในปี 2019 ประกันสุขภาพก็ยิ่งมีแนวโน้มเติบโตมากยิ่งขึ้นทั้งจากปัจจัยการเกิดโรคระบาดที่ผ่านมา และการหันมาดูแลสุขภาพมากยิ่งขึ้น [1]

ในประเทศไทย ถึงแม้ว่าจะมีสวัสดิการขั้นพื้นฐานจากรัฐบาลอย่างโครงการ 30 บาทรักษาได้ทุกโรคนั้น หรือสิทธิ์ประกันสังคมก็ตาม แต่ในการเจ็บป่วยแต่ละครั้งอาจจะเป็นได้ตั้งแต่เล็กน้อยไปจนถึงขั้นรุนแรงปฏิเสธไม่ได้เลยว่าหากอยากจะเข้ารับการรักษาที่รวดเร็วการรับการบริการจากเอกชนก็ถือได้ว่าเป็นหนึ่งในตัวเลือกที่สะดวกสบาย ดังนั้นในปัจจุบันการมีประกันสุขภาพสามารถสร้างความมั่นคงให้กับสถานภาพทางการเงินได้ระดับหนึ่งแต่หากจะต้องเลือกก็จะต้องดูค่าเบี้ยประกันต่อความคุ้มครองให้เหมาะสมด้วย สิ่งที่สำคัญที่สุดคือ จะต้องประเมินก่อนว่าเบี้ยประกันภัยที่ต้องจ่ายนั้น จะอยู่ที่เท่าไรเพื่อที่จะได้สามารถประเมินงบประมาณการเงินได้ ซึ่งก่อนการจะซื้อประกันสุขภาพจะต้องดูวิธีการเลือกซื้อคือ 1) ประเมินความเสี่ยงสุขภาพ 2) พิจารณาแผนความคุ้มครอง 3) ประเมินศักยภาพในการจ่ายเบี้ยประกัน 4) ตรวจสอบโรงพยาบาลที่ประกันรับเคลม 5) ค่ารักษาแบบแยกจ่ายหรือเหมาจ่าย [2] ในงานวิจัยของนี้จะเน้นไปที่การประเมินเบี้ยประกันสุขภาพเบื้องต้นจากข้อมูลโดยจะนำ การเรียนรู้ของเครื่อง (Machine Learning) มาช่วยในการทำนายเบี้ยประกันสุขภาพจากข้อมูลที่มี ซึ่งลักษณะข้อมูลที่จะนำมาใช้ในการเรียนรู้จะเป็นลักษณะข้อมูลเชิงกายภาพ เช่น อายุ เพศ น้ำหนัก เป็นต้น ซึ่งการเรียนรู้ของเครื่อง มีแบบจำลอง (Model) ของการเรียนรู้ที่หลากหลาย โดยจะเลือกใช้แบบจำลอง ทั้งหมด 6 แบบจำลอง ดังนี้ 1) Support Vector Regression (SVR) 2) Lasso Regression 3) Ridge Regression 4) Decision Tree 5) Random Forest และ 6) XGBoots (eXtream Gradient Boosting) ในงานวิจัยนี้จะทำการเปรียบเทียบผลลัพธ์ระหว่าง แบบจำลองต่าง ๆ เพื่อหาแบบจำลองที่ดีที่สุดสำหรับการทำนายเบี้ยประกันสุขภาพออกมา ซึ่งจะสามารถนำไปใช้ต่อยอดในแง่ของการนำมาประเมินเบี้ยประกันสุขภาพเองในเบื้องต้นหรืออาจนำไปปรับใช้กับธุรกิจประกันสุขภาพเพื่อลดเวลาในการประเมินเบี้ยประกันสุขภาพให้มี

ความรวดเร็วยิ่งขึ้น และการนำการเรียนรู้ของเครื่องมาใช้ในการทำนายเบี้ยประกันจะทำให้สามารถจะสามารถวิเคราะห์ความสัมพันธ์ที่ซับซ้อนและไม่เชิงเส้นระหว่างปัจจัยต่าง ๆ ได้ดีกว่าวิธีการคำนวณด้วยวิธีทางสถิติและสามารถปรับเปลี่ยนได้ตลอดเวลา

1.2 ความมุ่งหมายของงานวิจัย

ในการวิจัยครั้งนี้ผู้วิจัยได้ตั้งความมุ่งหมายไว้ดังนี้

1. สร้างแบบจำลองที่สามารถทำนายเบี้ยประกันสุขภาพจากข้อมูลได้
2. สรุปผลของแบบจำลองที่ทำนายเบี้ยประกันได้แม่นยำที่สุด
3. ศึกษาปัจจัยที่มีผลต่อเบี้ยประกันสุขภาพ

1.3 ความสำคัญของการวิจัย

การวิจัยนี้มีขึ้นเพื่อศึกษาปัจจัยของข้อมูลที่นำมาใช้ในการคิดวิเคราะห์หาเบี้ยประกันภัยสุขภาพ จะทำให้ผู้ใช้งานได้สามารถประเมินเบี้ยประกันสุขภาพของตัวเองได้คร่าว ๆ จากงานวิจัยว่าหากมีลักษณะข้อมูลประมาณนี้ เบี้ยประกันสุขภาพจะเป็นเท่าไร ทำให้สามารถประเมินค่าใช้จ่ายที่ต้องเตรียมไว้สำหรับการซื้อประกันสุขภาพล่วงหน้าได้อย่างเหมาะสม นอกจากนี้ยังช่วยเพิ่มความเข้าใจให้กับผู้บริโภคเกี่ยวกับโครงสร้างการคิดเบี้ยประกัน ทำให้เกิดความโปร่งใสและสร้างความเชื่อมั่นต่อระบบประกันสุขภาพมากยิ่งขึ้น

ในแง่ของวงการธุรกิจประกัน สามารถนำไปปรับใช้กับกระบวนการประเมินเบี้ยประกันสุขภาพให้มีความครอบคลุม และสะท้อนถึงความเสี่ยงของแต่ละบุคคลได้อย่างแม่นยำมากยิ่งขึ้น ข้อมูลเชิงลึกที่ได้จากการวิเคราะห์สามารถต่อยอดไปสู่การพัฒนาแบบจำลองการคำนวณเบี้ยประกันที่เหมาะสมกับพฤติกรรมและลักษณะเฉพาะของกลุ่มเป้าหมายแต่ละประเภท ส่งผลให้บริษัทประกันสามารถนำเสนอผลิตภัณฑ์ที่มีความยืดหยุ่นและตรงกับความต้องการของลูกค้ามากขึ้น อีกทั้งยังช่วยลดความเสี่ยงทางธุรกิจและสร้างความได้เปรียบทางการแข่งขันในตลาดประกันสุขภาพในอนาคต

นอกจากนี้ ผลการศึกษายังสามารถนำไปประยุกต์ใช้ในเชิงนโยบายหรือการออกแบบผลิตภัณฑ์ประกันสุขภาพในอนาคต โดยเฉพาะในด้านการวางกลยุทธ์ทางการตลาด เช่น การกำหนดกลุ่มเป้าหมายเฉพาะตามปัจจัยเสี่ยงที่ตรวจพบจากข้อมูล เช่น อายุ เพศ ประวัติการเจ็บป่วย พฤติกรรมการใช้ชีวิต หรือแม้กระทั่งปัจจัยด้านภูมิภาคและสภาพแวดล้อมทางสังคมและเศรษฐกิจ ซึ่งเป็นองค์ประกอบที่มีผลต่อการพิจารณาเบี้ยประกันอย่างมีนัยสำคัญ ยิ่งไปกว่านั้น การ

นำเทคนิคการวิเคราะห์ข้อมูล เช่น การเรียนรู้ของเครื่อง หรือการสร้างแบบจำลองทางสถิติ มาช่วยในการวิเคราะห์ข้อมูลจำนวนมาก จะช่วยให้การคาดการณ์เบี้ยประกันมีความแม่นยำและเป็นกลางมากขึ้น ลดการพึ่งพาการประเมินจากประสบการณ์ของเจ้าหน้าที่เพียงอย่างเดียว และยังสามารถอัปเดตปรับเปลี่ยนแบบจำลองได้ตามข้อมูลใหม่ที่เพิ่มเข้ามาในอนาคต ซึ่งส่งผลดีต่อทั้งผู้ให้บริการประกันภัยและผู้เอาประกันในระยะยาว

ในภาพรวม การวิจัยนี้ไม่เพียงแต่ช่วยส่งเสริมความเข้าใจในเชิงลึกเกี่ยวกับปัจจัยที่มีผลต่อเบี้ยประกันสุขภาพ แต่ยังเป็นแนวทางสำคัญที่สามารถต่อยอดไปสู่การพัฒนาระบบการประเมินความเสี่ยงอย่างเป็นระบบ โปร่งใส และมีประสิทธิภาพ ซึ่งจะนำไปสู่ระบบประกันสุขภาพที่ตอบโจทย์ทั้งในระดับบุคคลและระดับอุตสาหกรรมได้อย่างยั่งยืน

1.4 ขอบเขตของการวิจัย

งานวิจัยนี้จะใช้ การเรียนรู้ของเครื่อง ในการทำนายเบี้ยประกันสุขภาพจากชุดข้อมูล โดยชุดข้อมูลที่ใช้ในการศึกษาจะเป็นชุดข้อมูลสาธารณะ (Open source) ที่ได้จากเว็บไซต์ Kaggle [3] ชุดข้อมูลนี้จะมีข้อมูลทั้งหมด 986 ข้อมูล โดยรายละเอียดของชุดข้อมูลจะถูกพูดถึงในบทที่ 3 ซึ่งข้อมูลชุดนี้มาจากประเทศอินเดียดังนั้นตัวค่าเบี้ยประกันที่ทำนายได้จะไม่อยู่ในสกุลเงิน ไทยบาท แต่เพื่อให้เกิดความแม่นยำของตัวแบบจำลอง งานวิจัยนี้จะไม่ทำการเปลี่ยนแปลงข้อมูลใดจากข้อมูลดิบที่ได้มา

แบบจำลองที่ใช้ในการศึกษามีทั้งหมด 6 แบบจำลอง ดังนี้ ดังนี้ 1) Support Vector Regression (SVR) 2) Lasso Regression 3) Ridge Regression 4) Decision Tree 5) Random Forest และ 6) XGBoots (eXtream Gradient Boosting) โดยจะนำผลของแบบจำลองทั้งหมดมาเปรียบเทียบกันว่าแบบจำลองใดสามารถให้การทำนายค่าเบี้ยประกันสุขภาพได้ดีที่สุด ซึ่งทำการเปรียบเทียบผลระหว่างก่อนการปรับจูนไฮเปอร์พารามิเตอร์และหลังการปรับจูนไฮเปอร์พารามิเตอร์เพื่อให้ได้แบบจำลองที่มีประสิทธิผลดีที่สุด

ซึ่งปัจจัยที่มีผลต่อเบี้ยประกันภัยนั้นก็จะแตกต่างกันไปในแต่ละประเทศ พื้นที่ หรือบริษัทประกันภัยใดๆก็ตาม งานวิจัยนี้จะศึกษาการทำนายตามข้อมูลที่ได้มาซึ่งเป็นข้อมูลทั่ว ๆ ไปไม่ได้เจาะจงว่าเป็นข้อมูลจากบริษัทใดดังนั้นหากนำข้อมูลใหม่ที่ไม่ได้อยู่ในกรอบข้อมูลเดียวกับที่การเรียนรู้ของเครื่อง ได้เรียนรู้มาก็จะส่งผลให้ความแม่นยำของแบบจำลองนั้นต่ำลงแต่อย่างไรก็ตามหากในอนาคตได้มีโอกาสหาข้อมูลมาให้ การเรียนรู้ของเครื่อง ได้เรียนรู้ได้มากยิ่งขึ้นก็จะยิ่งทำให้มีประสิทธิภาพมากขึ้นตามไปด้วย

1.5 สมมุติฐานในการวิจัย

1. ตัวแปรที่มีผลต่อเบี้ยประกันที่สุดจะเป็นอายุ เนื่องจากตามหลักแล้วสุขภาพร่างกายของมนุษย์จะเสื่อมสภาพลงตามอายุขัยดังนั้นยิ่งผู้ทำประกันมีอายุเยอะก็จะส่งผลให้เบี้ยประกันภัยสูงตามด้วย

2. แบบจำลองที่คาดว่าน่าจะให้ผลแม่นยำที่สุดคือแบบจำลองในกลุ่มของ Regression เนื่องจากผลของการทำนายเป็นตัวเลขก็จะเหมาะสมกับกลุ่มแบบจำลอง Regression และข้อมูลที่นำมาใช้มีจำนวนไม่มากและไม่ซับซ้อน

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถเข้าใจลำดับขั้นตอนในการสร้างแบบจำลองการวิเคราะห์เบี้ยประกันสุขภาพตั้งแต่การรวบรวมและเตรียมข้อมูล การสำรวจข้อมูล การเลือกแบบจำลองที่เหมาะสม ตลอดจนการสร้างแบบจำลอง และ การประเมินประสิทธิภาพ

2. สามารถประเมินเบี้ยประกันสุขภาพของตนเองได้ในเบื้องต้นจากข้อมูลส่วนบุคคล เช่น อายุ เพศ พฤติกรรมการใช้ชีวิต และประวัติสุขภาพ

3. มีความเข้าใจเกี่ยวกับการคำนวณเบี้ยประกันสุขภาพมากขึ้น

4. ช่วยลดความสับสน และสร้างความโปร่งใสในการตัดสินใจเลือกผลิตภัณฑ์ประกัน

ในบทนี้ผู้วิจัยได้กล่าวถึงที่มาและความสำคัญของงานวิจัยเพื่อให้ได้ข้าใจว่างานวิจัยนี้มีจุดประสงค์อย่างไร ประโยชน์ที่จะได้รับจากงานวิจัยนี้คืออะไร ในบทถัดไปจะกล่าวถึงข้อมูลงานวิจัยที่เกี่ยวข้องที่ใช้ในการทำงานวิจัยนี้

บทที่ 2
ทบทวนวรรณกรรม

ในบทนี้กล่าวถึงแนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้องกับการทำนายค่าเบี้ยประกันสุขภาพ ซึ่งครอบคลุมถึงแนวคิดพื้นฐานของระบบประกันสุขภาพ ปัจจัยที่มีผลต่อการกำหนดเบี้ยประกันสุขภาพ เช่น อายุ เพศ ดัชนีมวลกาย พฤติกรรมการสูบบุหรี่ เป็นต้น ตลอดจนเทคนิคทางวิทยาการข้อมูลและการเรียนรู้ของเครื่อง ที่นำมาใช้ในการสร้างแบบจำลองเพื่อทำนายค่าเบี้ยประกัน เช่น Support Vector Regression, Lasso Regression, Ridge Regressiom, Decision Tree, Random Forest, และ XGBoost โดยกล่าวถึงหลักการทำงาน ข้อดีข้อจำกัด และกรณีศึกษาการใช้งาน เพื่อใช้เป็นกรอบแนวคิดในการเลือกเทคนิคที่เหมาะสมในการพัฒนาแบบจำลองการทำนายเบี้ยประกันสุขภาพในงานวิจัยนี้ โดยผู้วิจัยได้ศึกษาเอกสารและงานวิจัยที่เกี่ยวข้องทั้งหมด 5 งานหลักซึ่งทั้งหมด 5 งานนี้มีการทำวิจัยที่เกี่ยวข้องกับการทำนายเบี้ยประกันสุขภาพ

2.1 แบบจำลองที่เกี่ยวข้อง

หัวข้อนี้จะกล่าวถึงแบบจำลองที่เกี่ยวข้องที่ใช้ในการทำงานวิจัยนี้โดยจะกล่าวถึงสมมติฐาน และข้อจำกัดในแต่ละแบบจำลองเพื่อให้ได้เข้าใจหลักการทำงานและนำไปใช้ได้อย่างเหมาะสม

2.1.1 Support Vector Regression

Support Vector Regression (SVR) เป็นเทคนิคใน Machine Learning ที่ขยายแนวคิดของ Support Vector Machine (SVM) เพื่อนำไปใช้กับปัญหาการพยากรณ์ค่าต่อเนื่อง โดยมีเป้าหมายหลักคือการสร้างฟังก์ชันที่สามารถคาดการณ์ค่าของตัวแปรตาม (Dependent Variable) ให้แม่นยำภายในค่าความคลาดเคลื่อนที่กำหนดไว้

โดยหลักการของ SVR คือการหาฟังก์ชันที่สามารถประมาณค่าความสัมพันธ์ระหว่างตัวแปรต้นและตัวแปรตามในลักษณะที่ให้ค่าความคลาดเคลื่อนระหว่างค่าที่พยากรณ์ได้กับค่าจริงมีขนาดน้อยที่สุดภายใต้ขอบเขตที่ยอมรับได้ (epsilon-insensitive tube) ซึ่งถือเป็นข้อได้เปรียบในแง่ของความสามารถในการควบคุมความซับซ้อนของแบบจำลอง ทั้งนี้ SVR ใช้ kernel trick ในการแปลงข้อมูลให้อยู่ในมิติที่สูงขึ้นเพื่อให้สามารถแบ่งแยกหรือสร้างสมการได้อย่างมีประสิทธิภาพมากยิ่งขึ้น อย่างไรก็ตามข้อจำกัดของ SVR คือการที่ต้องเลือกตัวแปรหลายค่า เช่น ค่า C (ค่าควบคุมความซับซ้อน), epsilon (ค่าความอดทนต่อข้อผิดพลาด), และ kernel ต่าง ๆ ซึ่ง

ต้องอาศัยการปรับจูนอย่างเหมาะสมและใช้เวลาในการฝึกสูงเมื่อทำงานกับข้อมูลขนาดใหญ่หรือมีมิติสูง สมมติฐานของ SVR คือมีความสัมพันธ์ไม่เชิงเส้นหรือเชิงเส้นบางรูปแบบที่สามารถเรียนรู้ได้จากการแปลงข้อมูลผ่าน kernel functions [4]

2.1.2 Lasso Regression

Lasso ย่อมาจาก Least Absolute Shrinkage and Selection Operator เป็นแบบจำลองเชิงเส้นที่นิยมใช้เมื่อต้องการให้แบบจำลองเลือกเฉพาะ feature ที่สำคัญจริง ๆ โดยจะประมาณค่าสัมประสิทธิ์ (coefficients) ที่เป็นศูนย์สำหรับ feature ที่ไม่จำเป็น ทำให้แบบจำลองมีความ sparse (เลือกเฉพาะตัวแปรสำคัญ) โดยที่สมการของ Lasso จะมีการใช้ค่าปรับโทษแบบ L1-norm ที่มีลักษณะเป็นการบวกค่าสัมบูรณ์ของค่าสัมประสิทธิ์เข้าไปใน loss function ดังสมการ (1)

$$\min_{w} \frac{1}{2n_{samples}} ||Xw-y||_2^2 + \alpha||w||_1 \qquad (1)$$

โดยที่ $\alpha||w||_1$ คือ เทอมค่าปรับโทษแบบ L1-norm ซึ่งมี α เป็นค่าสัมประสิทธิ์คงที่ทำหน้าที่ควบคุมความแรงของการลงโทษ โดยค่าตั้งต้นจะอยู่ที่ 1 การมีเทอมนี้ช่วยให้สามารถลดค่าสัมประสิทธิ์ของบาง feature ให้เหลือศูนย์ได้ ส่งผลให้แบบจำลองสามารถเลือกเฉพาะตัวแปรที่สำคัญจริง ๆ (feature selection) ได้โดยอัตโนมัติ และจุดเด่นหลัก ๆ ของ Lasso คือ ลดตัวแปร (feature) แบบอัตโนมัติ ลด overfitting ซึ่งข้อจำกัดของ Lasso ก็คือ กรณีที่ตัวแปรมีความสัมพันธ์กันสูง Lasso อาจสุ่มเลือกลบตัวใดตัวหนึ่งออกซึ่งนั้นอาจจะทำให้การทำนายผิดพลาดไปได้ [5]

2.1.3 Ridge Regression

Ridge Regression เป็นเทคนิคการถดถอยเชิงเส้นที่เพิ่มการลงโทษแบบ L2 เข้าไปในสมการ (2)

$$\min_{w} ||Xw-y||_2^2 + \alpha||w||_2^2 \qquad (2)$$

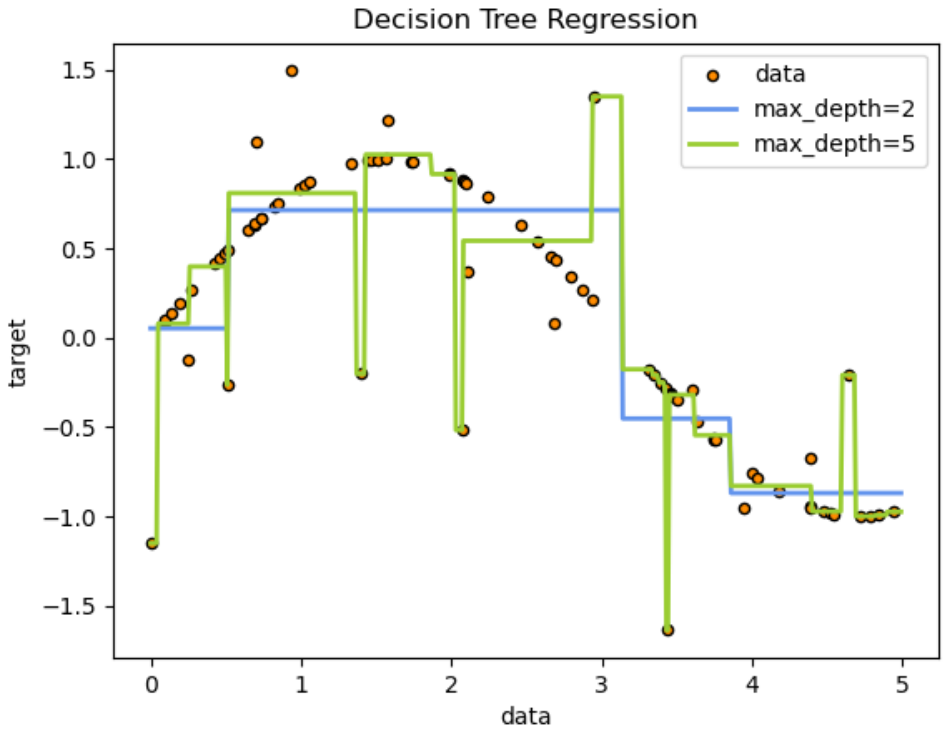
โดยที่ $\alpha||w||_2^2$ คือ เทอมค่าปรับโทษแบบ L2-norm ซึ่งมี α เป็นค่าสัมประสิทธิ์คงที่ทำหน้าที่ควบคุมความแรงของการลงโทษเช่นเดียวกับ Lasso Regression แต่จะแตกต่างกันในส่วนที่ว่า L2-norm นั้นมีการยกกำลังสอง โดยจะช่วยควบคุมไม่ให้ค่าสัมประสิทธิ์มีขนาดใหญ่

จนเกินไป ซึ่งจะทำให้แบบจำลองซับซ้อนน้อยลงและลดความเสี่ยงของการ overfitting ได้ดี โดยต่างจาก Lasso ที่ใช้การลงโทษแบบ L1 ซึ่งสามารถลดค่าสัมประสิทธิ์ให้เหลือศูนย์เพื่อทำ feature selection ได้ แต่ Ridge จะไม่ตัดค่าตัวแปรให้เป็นศูนย์แต่จะปรับให้เล็กลงแทน ทำให้เหมาะกับกรณีที่ตัวแปรหลายตัวมีความสัมพันธ์กัน (multicollinearity) และไม่ต้องการตัดตัวแปรใดตัวแปรหนึ่งทิ้ง

ข้อดีของ Ridge Regression คือสามารถใช้งานกับชุดข้อมูลที่มีตัวแปรจำนวนมาก และสามารถจัดการกับปัญหา multicollinearity ได้ดี อีกทั้งยังช่วยให้แบบจำลองมีความเสถียรและแม่นยำมากขึ้นเมื่อเจอกับข้อมูลใหม่ แต่ข้อจำกัดคือไม่สามารถใช้ในการเลือกตัวแปรสำคัญได้เหมือน Lasso และต้องมีการปรับค่าตัวแปร α อย่างเหมาะสม ซึ่งอาจต้องใช้เทคนิคอย่าง cross-validation ในการช่วยเลือกค่าที่ดีที่สุด[6]

2.1.4 Decision Tree

แบบจำลอง Decision Tree เป็นแบบจำลองที่สามารถใช้กับการจำแนก (Classifier) และการถดถอย (Regression) โดยเรียนรู้กฎ if-then-else จากข้อมูล เพื่อทำนายเป้าหมายแบบแบ่งช่วง (piecewise constant) ซึ่งการตัดสินใจแต่ละครั้งจะพิจารณาจากเกณฑ์เช่น Gini impurity หรือ entropy เพื่อลดความไม่แน่นอนของข้อมูลในแต่ละโหนดลงเรื่อย ๆ แบบจำลองนี้มีข้อดีเรื่องความสามารถในการทำงานกับข้อมูลที่เป็นเชิงหมวดหมู่และเชิงตัวเลขผสมกันได้ดี เข้าใจง่าย และใช้ในระบบที่ต้องการการอธิบายแบบจำลอง (interpretability) ได้เป็นอย่างดี อย่างไรก็ตาม ข้อจำกัดของ Decision Tree คือความโน้มเอียงที่จะเกิด overfitting ได้ง่ายถ้าไม่มีการควบคุม เช่น ไม่จำกัดความลึกของต้นไม้หรือตัวแยกขั้นต่ำ อีกทั้งยังอ่อนไหวต่อข้อมูล noise หรือข้อมูลที่มีความผันผวนสูง ทำให้ต้นไม้เปลี่ยนไปมากเพียงแค่มีการเปลี่ยนแปลงข้อมูลเล็กน้อย ภาพประกอบท 1 จะแสดงตัวอย่างการทำงานของ Decision Tree [7]



ภาพประกอบ 1 ตัวอย่างการทำงานของ Decision Tree แบบ Regression

ที่มา: Decision tree. 2025. https://scikit-learn.org/stable/modules/tree.html

จากภาพประกอบ 1 แสดงตัวอย่างผลการทำนายจาก Decision Tree Regression โดยใช้ต้นไม้ตัดสินใจที่มีความลึก (depth) ต่างกันบนข้อมูลตัวอย่าง ซึ่งแสดงให้เห็นถึงผลกระทบของพารามิเตอร์ max_dept ต่อความสามารถของแบบจำลองในการเรียนรู้รูปแบบของข้อมูล

2.1.5 Random Forest

Random Forest เป็นแบบจำลองที่ใช้วิธีการ Ensemble คือวิธีการที่รวมการพยากรณ์จากแบบจำลองพื้นฐานหลายตัว (base estimators) ที่สร้างจากอัลกอริทึมเดียวกัน เพื่อเพิ่มความสามารถในการทำนายและความทนทานของแบบจำลอง ให้ดีกว่าใช้แบบจำลองเดียว โดยที่ Random Forest ก็สามารถใช้ได้กับปัญหาแบบ การจำแนก (Classifier) และการถดถอย (Regression) ได้ โดยแต่ละต้นไม้จะถูกฝึกด้วยข้อมูลที่สุ่มมาจากชุดข้อมูลต้นฉบับ และในแต่ละจุดแบ่ง (split) ก็จะสุ่มเลือกเฉพาะบาง feature มาใช้ตัดสิน ทำให้ต้นไม้แต่ละต้นมีความแตกต่างกัน และเมื่อนำผลลัพธ์ของทุกต้นไม้มาเฉลี่ยกันหรือโหวตเสียงข้างมาก จะได้ผลลัพธ์ที่แม่นยำและ

เสถียรกว่าแบบจำลองต้นไม้เดี่ยว ๆ มาก ช่วยลดความเสี่ยงของการ overfitting ได้ดี เพราะความผิดพลาดของแต่ละต้นไม้จะเฉลี่ยกันออกไป

ข้อดีของ Random Forest คือมีความแม่นยำสูง ใช้งานได้ดีทั้งกับข้อมูลที่เป็นเชิงตัวเลขหรือเชิงหมวดหมู่ ไม่ต้องปรับตัวแปรมากก็ให้ผลดีในระดับหนึ่ง และยังสามารถดูความสำคัญของตัวแปร (feature importance) ได้ด้วย ซึ่งเหมาะกับการวิเคราะห์ข้อมูลเพื่อหาปัจจัยสำคัญในการพยากรณ์ แต่ข้อจำกัด คือแบบจำลองมีความซับซ้อนและตีความยาก เพราะไม่ได้มีสูตรชัดเจนแบบสมการทางคณิตศาสตร์เหมือน Linear Regression หรือ Decision Tree เดี่ยว ๆ นอกจากนี้การฝึกและพยากรณ์ก็อาจใช้เวลาและทรัพยากรค่อนข้างมากโดยเฉพาะกับข้อมูลขนาดใหญ่หรือเมื่อต้นไม้มีจำนวนเยอะมาก และแม้จะลด overfitting ได้ดี แต่ก็ยังมีความเสี่ยงหากข้อมูลมี noise เยอะหรือตัวแปรไม่เหมาะสม [8]

2.1.6 eXtream Gradient Boosting (XGBoots)

XGBoost หรือ Extreme Gradient Boosting เป็นแบบจำลองแบบ ensemble ที่พัฒนามาจากแนวคิดของ Gradient Boosting โดยจะสร้างต้นไม้แบบทีละต้นโดยให้ต้นไม้ใหม่เรียนรู้จากข้อผิดพลาดของต้นไม้ก่อนหน้าเรื่อย ๆ ซึ่งทำให้แบบจำลองมีความสามารถในการจับความซับซ้อนของข้อมูลได้ดีและมีความแม่นยำสูง จุดเด่นของ XGBoost คือความเร็วในการฝึกแบบจำลองและประสิทธิภาพที่ดีกว่า Gradient Boosting แบบเดิม เพราะถูกออกแบบมาให้สามารถทำงานแบบขนาน (parallel) ได้ และยังมีระบบ regularization ที่ช่วยลดการ overfitting อีกทั้งยังรองรับการทำงานแบบ out-of-core สำหรับข้อมูลที่ไม่สามารถเก็บในหน่วยความจำได้ทั้งหมด

ในด้านกลไกการทำงาน XGBoost ใช้การเรียนรู้แบบเพิ่มพูน (additive learning) ด้วยการสร้างต้นไม้แต่ละต้นโดยใช้ ค่าการประมาณอันดับที่สอง (second-order gradient) ซึ่งช่วยให้การหาค่าที่เหมาะสมเป็นไปอย่างรวดเร็วและมีประสิทธิภาพ อีกทั้งยังมีการออกแบบอัลกอริทึมเพื่อค้นหาจุดแบ่งข้อมูลที่เหมาะสม โดยมีทั้งแบบ exact greedy คือพิจารณาทุกจุดแบ่งข้อมูลที่เป็นไปได้เพื่อหาจุดที่ดีที่สุด และ approximate คือใช้การประมาณค่าเพื่อลดเวลาคำนวณ เช่น การพิจารณาเฉพาะจุดแบ่งที่เลือกไว้บางส่วนซึ่งผู้ใช้งานสามารถเลือกใช้งานตามบริบทได้ XGBoost ยังมีการออกแบบให้รองรับการเรียนรู้จากข้อมูลที่มีค่าหาย (missing values) ได้อย่างมีประสิทธิภาพผ่าน sparsity-aware algorithm คือการออกแบบให้รู้จักและใช้ประโยชน์จากข้อมูลที่มีค่าว่างหรือศูนย์จำนวนมาก และสามารถกำหนดเส้นทางเริ่มต้นให้ข้อมูลที่มีค่าหายว่าควรเข้ากิ่งไหนของต้นไม้ (default direction) ของ node ในแต่ละต้นไม้ได้อัตโนมัติ

อย่างไรก็ตาม XGBoost มีข้อจำกัดในบางด้าน เช่น กระบวนการฝึกแบบจำลองที่อาจซับซ้อนและใช้เวลามากเมื่อมี ไฮเปอร์พารามิเตอร์จำนวนมาก โดยเฉพาะอย่างยิ่งการจูนพารามิเตอร์ (tuning) ที่มีผลต่อความลึกของต้นไม้, อัตราการเรียนรู้ (learning rate) และจำนวนต้นไม้ ซึ่งต้องใช้การทดลองซ้ำหลายครั้งเพื่อให้ได้แบบจำลองที่ดีที่สุด นอกจากนี้ในกรณีที่มีตัวแปรจำนวนมาก การใช้ exact greedy algorithm อาจใช้ทรัพยากรสูง และแม้ว่าระบบจะมีการการหาจุดแบ่งข้อมูลโดยใช้การประมาณ เพื่อประหยัดเวลาและหน่วยความจำ (approximate split finding) ที่ใช้เทคนิค weighted quantile sketch คือ เทคนิคที่ใช้ในการประมาณค่าควอนไทล์จากข้อมูลที่มีการถ่วงน้ำหนัก เพื่อลดเวลาและหน่วยความจำ แต่ก็มีโอกาสที่จะพลาดจุดแบ่งที่ดีที่สุดได้

โดยสรุป XGBoost สามารถให้ผลลัพธ์ที่แม่นยำมาก โดยเฉพาะกับข้อมูลขนาดใหญ่หรือการแข่งขันด้านแบบจำลองพยากรณ์ต่าง ๆ ที่มักเห็น XGBoost เป็นหนึ่งในตัวเลือกหลัก ยังรองรับการจัดการกับ missing values ได้อัตโนมัติ และมีเครื่องมือวิเคราะห์ feature importance รวมถึงการปรับแต่งตัวแปรที่หลากหลายเพื่อให้เหมาะกับข้อมูล แต่ข้อจำกัดของ XGBoost คือแบบจำลองค่อนข้างซับซ้อนและต้องการการปรับจูนตัวแปรหลายตัว เช่น learning rate, max depth, และจำนวนต้นไม้ ซึ่งถ้าไม่จูนให้เหมาะสมอาจทำให้ประสิทธิภาพของแบบจำลองลดลงหรือเกิด overfitting ได้ นอกจากนี้เนื่องจากความซับซ้อนในการฝึกและพยากรณ์ ก็อาจใช้เวลาหรือทรัพยากรสูงกว่าระบบทั่วไป และไม่เหมาะเท่าไรนักถ้าข้อมูลมีขนาดเล็กหรือไม่ซับซ้อน [9]

2.2 วิธีการประเมินแบบจำลอง

สำหรับวิธีการประเมินแบบจำลองได้เลือกใช้ทั้งหมด 4 วิธีคือ 1) ค่าสัมประสิทธิ์การถดถอยกำลังสอง (R^2) 2) ค่าเฉลี่ยของความผิดพลาดแบบสัมบูรณ์ (MAE) 3) ค่าความคลาดเคลื่อนสัมบูรณ์เฉลี่ยร้อยละ (MAPE) 4) รากที่สองของค่าเฉลี่ยความผิดพลาดกำลังสอง (RMSE) ซึ่งโดยแต่ละวิธีการประเมินก็จะมีหลักการและสมการดังต่อไปนี้

2.2.1 ค่าสัมประสิทธิ์การถดถอยกำลังสอง (R^2)

ค่าสัมประสิทธิ์การถดถอยกำลังสอง หรือ (R^2) เป็นค่าที่ใช้วัดว่า สมการถดถอยสามารถอธิบายความแปรปรวนของข้อมูลได้มากน้อยแค่ไหน โดยค่าจะอยู่ในช่วง 0 ถึง 1 โดยที่ ถ้าค่า R^2 เท่ากับ 1 หมายความว่า แบบจำลองสามารถอธิบายข้อมูลได้สมบูรณ์แบบ แต่ถ้าค่า R^2 เท่ากับ 0 หมายความว่า แบบจำลองไม่สามารถอธิบายความแปรปรวนของข้อมูลได้เลย ดังนั้นค่า R^2 ที่ดีนั้นควรจะเป็นค่าที่เข้าใกล้ 1 มากที่สุด[10] อธิบายค่า R^2 ดังสมการ (3)

$$R^2=1-\frac{SS_{res}}{SS_{tot}} \quad (3)$$

โดยที่

$SS_{res} = \sum (y_i-\hat{y}_i)^2$ คือ Residual Sum of Squares (ความคลาดเคลื่อนของแบบจำลอง)

$SS_{res} = \sum (y_i-\bar{y}_i)^2$ คือ Total Sum of Squares (ความแปรปรวนรวมจากค่าเฉลี่ย)

2.2.2 ค่าเฉลี่ยของความผิดพลาดแบบสัมบูรณ์ (MAE)

ค่าเฉลี่ยของความผิดพลาดแบบสัมบูรณ์ (MAE) คือค่าที่บอกว่าโดยเฉลี่ยแล้วแบบจำลอง ทำนายผิดจากค่าจริงมากแค่ไหน โดยวัดจากความผิดพลาดแบบ “สัมบูรณ์” ซึ่งก็คือไม่สนใจเครื่องหมาย บวกหรือลบ เอาแค่ขนาดของความผิดพลาดถ้าค่า MAE ที่ต่ำจะบ่งบอกได้ว่าแบบจำลองนั้นมีการทำนายผิดจากข้อมูลนั้นน้อยซึ่งก็คือแบบจำลองมีความแม่นยำมาก ถ้าค่า MAE เท่ากับ 0 แสดงว่าแบบจำลองนั้นไม่มีความผิดพลาดเลยคือแบบจำลองสามารถทำนายผลได้ตรง 100% ซึ่งในความเป็นจริงนั้นเป็นไปได้ยาก และค่า MAE จะไม่ติดลบได้เนื่องจากสมการเป็นค่าสัมบูรณ์ซึ่งผลลัพธ์ที่ได้จะเป็นบวกเท่านั้น[10] สมการ (4) อธิบายค่า MAE ดังนี้

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i-\hat{y}_i\right| \quad (4)$$

โดยที่

y_i คือ ค่าจริง (Observed / Actual value)

$\hat{y}_i$ คือ ค่าที่แบบจำลองทำนายได้ (Predicted value)

$\left|y_i-\hat{y}_i\right|$ คือ ค่าความผิดพลาดแบบสัมบูรณ์ (Absolute Error)

n คือ จำนวนจุดข้อมูลทั้งหมด

2.2.3 ค่าความคลาดเคลื่อนสัมบูรณ์เฉลี่ยร้อยละ (MAPE)

ค่าความคลาดเคลื่อนสัมบูรณ์เฉลี่ยร้อยละ (Mean Absolute Percentage Error: MAPE) หรือเรียกอีกชื่อว่า Mean Absolute Percentage Deviation (MAPD) เป็นตัวชี้วัดที่ใช้ประเมินประสิทธิภาพของแบบจำลองในการแก้ปัญหาแบบ Regression โดยมีแนวคิดในการวัดข้อผิดพลาดเชิงสัมพัทธ์ (Relative Error) กล่าวคือ MAPE จะคำนึงถึงสัดส่วนของความคลาดเคลื่อนเทียบกับค่าจริง ทำให้ มีความไวต่อความผิดพลาดเชิงสัมพัทธ์ ไม่เปลี่ยนแปลงเมื่อมีการปรับขนาดค่าทั้งหมดของตัวแปรเป้าหมาย (Target Variable) เช่น การคูณค่าทั้งหมดด้วยค่าคงที่หนึ่ง จะไม่กระทบกับค่า MAPE จึงเหมาะสำหรับใช้เปรียบเทียบแบบจำลองที่ทำงานกับข้อมูลที่มีขนาดต่างกันหรือเมื่อต้องการสื่อสารผลการทำนายในเชิงเปอร์เซ็นต์ [10] สมการ (5) อธิบายค่า MAPE ดังนี้

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i-\hat{y}_i}{y_i}\right| \times 100 \qquad (5)$$

โดยที่

y_i คือ ค่าจริง (Observed / Actual value)

$\hat{y}_i$ คือ ค่าที่แบบจำลองทำนายได้ (Predicted value)

n คือ จำนวนจุดข้อมูลทั้งหมด

2.2.4 รากที่สองของค่าเฉลี่ยความผิดพลาดกำลังสอง (RMSE)

รากที่สองของค่าเฉลี่ยความผิดพลาดกำลังสอง (Root Mean Square Error : RMSE) คือค่าถอดรากที่สองของค่าความคลาดเคลื่อนเฉลี่ยกำลังสอง (MSE) เพื่อให้หน่วยของข้อมูลตรงกับข้อมูลจริงซึ่งหลักการก็จะเหมือนกับ MSE[10] คือค่าเฉลี่ยของผลต่างระหว่างค่าจริง (Actual) กับค่าที่แบบจำลองทำนาย (Predicted) ที่ถูกยกกำลังสองซึ่งจะไม่เหมาะกับข้อมูลที่มีผิดปกติมาก (outlier) เนื่องจากจะทำให้ค่าที่ผิดปกติมีอิทธิพลสูงเกินไป RMSE ดังสมการ (6)

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2} \qquad (6)$$

2.3 งานวิจัยที่เกี่ยวข้อง

ผู้วิจัยได้ศึกษาเกี่ยวงานวิจัยของ Kashish Bhatia และคณะ[11], Sudhir Panda และคณะ[12] , Keshav Kaushik และคณะ[13] และ Mounika Karakavalasa และคณะ[14] ซึ่งทั้งสี่งานวิจัยที่เกี่ยวที่กล่าวมานั้นได้ใช้ชุดข้อมูล จาก Kaggle ซึ่งมี 1,338 ข้อมูลตัวอย่าง[15] ตัวแปรที่ใช้ประกอบด้วยอายุ เพศ ดัชนีมวลกาย (BMI) พฤติกรรมการสูบบุหรี่ และจำนวนบุตร เป็นต้น ซึ่งข้อมูลนั้นเป็นข้อมูลมาจากประกันสุขภาพใน 4 ภูมิภาคของประเทศสหรัฐอเมริกา

Kashish Bhatia และคณะ ได้ใช้เทคนิค K-fold cross-validation สำหรับการแยกข้อมูลเป็นกลุ่มย่อยเพื่อการฝึกและทดสอบ และมีการสร้างตัวแปรใหม่ เป็นการจำแนกผู้สูบบุหรี่และไม่สูบบุหรี่ และ จำแนกคนที่มีภาวะอ้วนและไม่มีภาวะอ้วนจากค่าดัชนีมวลกาย ซึ่งแบบจำลองที่ใช้จะใช้แบบจำลอง Linear Regression เพื่อดูความสัมพันธ์ระหว่างตัวแปรต่าง ๆ และเบี้ยประกันสุขภาพ ซึ่งผลลัพธ์ที่ได้จากงานวิจัยของ Kashish Bhatia และคณะ ให้ความแม่นยำของแบบจำลองอยู่ที่ 81.3% โดยใช้สัดส่วนการฝึกและทดสอบแบบจำลองที่ 70 ต่อ 30 โดยสรุปจากการงานวิจัยของ Kashish Bhatia มีการใช้แบบจำลองเพียงแบบเดียวคือแบบจำลอง Linear Regression โดยตัวแปรที่ส่งผลต่อเบี้ยประกันสุขภาพมากที่สุดคือ พฤติกรรมการสูบบุหรี่ ที่ให้ค่าสัมประสิทธิ์ความสัมพันธ์อยู่ที่ 0.78 มากกว่าตัวแปรอื่น ๆ และ Kashish Bhatia และคณะ ได้ทำการสร้างเว็บไซต์สำหรับการคำนวณเบี้ยประกันเบื้องต้น

งานวิจัยของ Sudhir Panda และคณะ ได้ศึกษาเรื่องการทำนายเบี้ยประกันสุขภาพที่ใช้การเรียนรู้ของเครื่อง โดยใช้แบบจำลอง Regression ที่หลากหลายคือ 1) Linear Regression 2) Multiple Linear Regression 3) Ridge Regression 4) Lasso Regression และ 5) Polynomial Regression เพื่อทำการเปรียบเทียบประสิทธิภาพในแต่ละแบบจำลองว่าแบบไหนให้ความแม่นยำที่มากที่สุด โดยงานวิจัยของ Sudhir Panda และคณะ ใช้ค่า ค่าสัมประสิทธิ์การถดถอยกำลังสอง (R^2) และค่า รากที่สองของค่าเฉลี่ยความผิดพลาดกำลังสอง (RMSE) เป็นตัววัดประสิทธิผลของแบบจำลอง และมีการใช้สัดส่วนการฝึกและทดสอบแบบจำลองที่ 80 ต่อ 20 ซึ่งผลลัพธ์ที่ได้จากงานวิจัยของ Sudhir Panda และคณะ แบบจำลองที่มีประสิทธิภาพดีที่สุดคือแบบจำลอง Polynomial Regression โดยให้ค่า R^2 เท่ากับ 0.80 ค่า RMSE เท่ากับ 5100.53 และค่าความแม่นยำ เท่ากับ 80.97%ซึ่งแบบจำลอง Polynomial Regression จะทำงานได้ดีกว่าแบบจำลองอื่นในกรณีที่ข้อมูลมีความสัมพันธ์แบบไม่เชิงเส้น โดยแบบจำลอง Ridge และ Lasso Regression ให้ค่าความแม่นยำอยู่ที่ 75.86% และ 75.82% ตามลำดับ ส่วนแบบจำลอง Simple Linear Regression จะให้ค่าความแม่นยำต่ำที่สุดที่ 62.86%

งานวิจัยของ Keshav Kaushik และคณะ ได้ศึกษาเรื่องการทำนายเบี้ยประกันสุขภาพที่ใช้ Machine Learning โดยใช้แบบจำลอง Linear Regression เปรียบเทียบกับแบบจำลอง Artificial Neural Network (ANN) เพื่อศึกษาว่าหากมีการนำเรื่องของการเรียนรู้มาใช้ในการทำนายจะให้ผลได้ดีกว่าการเรียนรู้ของเครื่องแบบธรรมดาหรือไม่ โดยจะมีการแบ่งสัดส่วนการฝึกและทดสอบแบบจำลองที่ 80 ต่อ 20 แบบจำลอง Artificial Neural Network (ANN) จะใช้โครงสร้างแบบจำลอง 5 เลเยอร์ Dense พร้อมฟังก์ชัน Relu และ Adam Optimizer โดยปรับตัวแปร ของ batch size และ epochs โดยงานวิจัยนี้จะใช้ตัววัดประสิทธิผลของแบบจำลองทั้งหมด 4 แบบ คือ 1) ค่าสัมประสิทธิ์การถดถอยกำลังสอง (R^2) 2) ค่าเฉลี่ยของความผิดพลาดแบบสัมบูรณ์ (MAE) 3) รากที่สองของค่าเฉลี่ยความผิดพลาดกำลังสอง (RMSE) และ 4) ค่าความคลาดเคลื่อนเฉลี่ยกำลังสอง (MSE) โดยผลลัพธ์ที่ได้แบบจำลอง ANN จะให้ผลลัพธ์ที่ดีกว่าโดยมีค่าตัววัดผลประสิทธิภาพ R^2 เท่ากับ 0.92 RMSE เท่ากับ 0.27, MSE เท่ากับ 0.073 และ MAE เท่ากับ 0.143 ในส่วนของแบบจำลอง Linear Regression จะให้ค่า R^2 เท่ากับ 0.75 RMSE เท่ากับ 0.499, MSE เท่ากับ 0.249 และ MAE เท่ากับ 0.345 โดยสรุปแล้วการนำการเรียนรู้เชิงลึกมาใช้กับการทำนายค่าเบี้ยประกันให้ประสิทธิผลที่ดีกว่าเนื่องจากการเรียนรู้เชิงลึกสามารถวิเคราะห์แบบจำลองได้รวดเร็วและแม่นยำกว่าการเรียนรู้ของเครื่อง ในอนาคตสามารถเพิ่มความซับซ้อนของข้อมูลเข้าไปเพื่อให้แบบจำลองสามารถทำนายค่าเบี้ยประกันได้แม่นยำมากกว่าเดิม

งานวิจัยของ Mounika Karakavalasa และคณะ ได้ศึกษาเรื่องการทำนายเบี้ยประกันสุขภาพที่ใช้ Machine Learning โดยใช้แบบจำลอง XGBoost, CatBoost, ElasticNetCV, Lasso Regression, Ridge Regression, Voting Regression และ Stacking Regression เพื่อทำการเปรียบเทียบแบบจำลองสองแบบหลักคือ แบบ Regression และ แบบ Tree สำหรับแบบจำลอง XGBoost, CatBoost โดยงานวิจัยนี้จะใช้ตัววัดประสิทธิผลของแบบจำลองคือ ค่ารากที่สองของค่าเฉลี่ยความผิดพลาดกำลังสอง (RMSE) และ ค่าคะแนนการอธิบายความแปรปรวน (Explained Variance Score) ซึ่งเป็นค่าชี้วัดที่ใช้ประเมินประสิทธิภาพของแบบจำลอง แบบ Regression โดยเฉพาะในการทำนายค่าที่ต่อเนื่อง (Continuous Values) เพื่อวัดว่าแบบจำลองสามารถอธิบายความแปรปรวนของข้อมูลจริงได้ดีเพียงใดเมื่อเทียบกับค่าที่ทำนายได้ งานวิจัยของ Mounika Karakavalasa และคณะ ไม่ได้บอกในเรื่องของการแบ่งสัดส่วนการฝึกและทดสอบแบบจำลอง ซึ่งผลที่ได้พบว่า แบบจำลอง Stacking Regression ให้ผลลัพธ์ดีที่สุด โดยได้ค่า RMSE เท่ากับ 0.313 และค่าคะแนนการอธิบายความแปรปรวนเท่ากับ 0.89 เมื่อเปรียบเทียบกับแบบจำลองอื่น เช่น XGBoost, CatBoost, และ ElasticNetCV พบว่า Stacking Regression มีความแม่นยำสูงสุด ซึ่ง

จากงานวิจัยนี้ที่ได้ทำการเปรียบเทียบผลระหว่างแบบจำลองหลัก พบว่าแบบจำลองที่ใช้แบบ Regression จะให้ผลลัพธ์ดีกว่าแบบจำลองที่ใช้หลักการของแบบ Tree ซึ่งอาจจะเป็นเพราะว่าข้อมูลมีความสัมพันธ์เป็นแบบเชิงเส้นไม่ได้มีความซับซ้อนมากแบบจำลองแบบ Regression จึงให้ผลลัพธ์ที่ดีกว่า

งานวิจัยของ Ugochukwu Orji และ คณะ [16] มีการใช้ชุดข้อมูลที่แตกต่างไปจากงานวิจัยที่ได้ศึกษาไปข้างต้น โดยชุดข้อมูลนั้นมาจาก Kaggle แต่จะมีตัวแปรทั้งหมด 11 ตัว[3] และตัวชุดข้อมูลนี้มีค่าเบี้ยประกันที่อ้างอินมาจากสกลุเงินรูปีของปรุเทศอินเดีย โดยงานวิจัยนี้ได้ใช้การเรียนรู้เครื่องในการทำนายค่าเบี้ยประกันสุขภาพผ่านสามแบบจำลองคือ 1) Xtream Gradient Boosting (XGBoots) 2) Gradient Boosting Machine (GBM) และ 3) Random Forest (RF) และใช้การวัดผลจำลองทั้งหมด 4 วิธีคือ 1) ค่าสัมประสิทธิ์การถดถอยกำลังสอง (R^2) 2) ค่าเฉลี่ยของความผิดพลาดแบบสัมบูรณ์ (MAE) 3) รากที่สองของค่าเฉลี่ยความผิดพลาดกำลังสอง (RMSE) 4) ค่าเฉลี่ยความผิดพลาดแบบสัมบูรณ์เชิงเปอร์เซ็นต์ (MAPE) โดย Ugochukwu Orji และ คณะ ได้ทำการปรับตัวแปรและเลือกตัวแปรที่ให้ผลที่ดีที่สุดมาใช้กับแบบจำลองทั้งสามจากนั้นจึงค่อยนำตัวแปรเหล่านั้นไปทดสอบกับชุดข้อมูลทดสอบเพื่อดูผลลัพธ์ผ่านการวัดผลทั้ง 4 แบบที่กล่าวไปข้างต้น จากการวัดผลทำให้ได้ผลลัพธ์ดังต่อไปนี้ XGBoots ให้ค่าวัดผลของ ค่า R^2 อยู่ที่ 86.47% ซึ่งมีค่าเข้าใกล้ 1 มากที่สุดในบรรดาสามแบบจำลอง และให้ค่า RMSE ที่ 2231.524 ซึ่งเป็นค่าที่ต่ำที่สุดในบรรดาสามแบบจำลอง ในขณะเดียวกันแบบจำลอง Random Forest ให้ MAE และ ค่า MAPE ต่ำสุด ซึ่งค่าจากการวัดผลนี่ควรจะมีค่าที่ต่ำดังนั้นจากงานวิจัยของ Ugochukwu Orji และ คณะ จึงสรุปได้ว่าแบบจำลองที่สามารถทำนายค่าเบี้ยประกันสุขภาพได้อย่างมีประสิทธิภาพจะเป็นแบบจำลอง XGBoots และ Random Forest จากงานวิจัยของ Ugochukwu Orji และ คณะ ได้ทำการศึกษาแบบจำลองที่ใช้ Tree เป็นหลักไม่ได้มีการเปรียบเทียบกับแบบจำลองแบบ Regression

2.4 สรุปงานวิจัยที่เกี่ยวข้อง

จากการที่ศึกษางานวิจัยที่เกี่ยวกับการทำนายเบี้ยประกันสุขภาพโดยใช้การเรียนรู้ของเครื่องทั้ง 5 งานวิจัย โดยสรุป 4 ใน 5 ของงานวิจัยมีการใช้ชุดข้อมูลที่เหมือนกันแต่มีการใช้แบบจำลองที่ใช้ในการฝึกแบบจำลองที่แตกต่างกันไปแต่โดยรวมแล้วจะเน้นการใช้แบบจำลองที่มีหลักการแบบ Regression และ แบบ Tree นำมาเปรียบเทียบกัน โดยผลลัพธ์ที่ได้จะพบว่าแบบจำลองที่ใช้หลักการการแบบ Regression จะให้แบบจำลองที่มีประสิทธิผลดีกว่าและใน

ขณะเดียวกันในแบบจำลองที่ใช้หลักการแบบ Regression จะพบว่าแบบจำลองแบบ Regression ที่ไม่ใช่แบบ linear จะให้ผลลัพธ์ที่ดีกว่าเนื่องจากมีการเพิ่มความซับซ้อนเข้าไปจากแบบ linear regression และในขณะเดียวกันงานวิจัยของ Ugochukwu Orji และ คณะ ที่มีการใช้ชุดข้อมูลที่แตกต่างจากงานวิจัยอื่นมีการทำแบบจำลองเปรียบเทียบแค่ในส่วนของแบบจำลองที่ใช้หลักแบบ Tree ไม่ได้มีการเปรียบเทียบกับแบบ Regression ซึ่งถ้าหากดูแนวโน้มแล้วแบบจำลองที่ใช้หลักการแบบ Regression น่าจะให้ผลลัพธ์ที่ดีกว่าสำหรับชุดข้อมูลของงานวิจัย Ugochukwu Orji และคณะ แต่อย่างไรก็ตามข้อสมมติฐานนี้ต้องได้รับการทดสอบกับชุดข้อมูลนี้ถึงจะตอบได้ว่าแบบจำลองแบบ Regression ให้ผลลัพธ์ที่ดีสำหรับการทำนายค่าเบี้ยประกันสุขภาพ นอกจากนี้จากงานวิจัยทั้ง 5 งานพบว่าตัวแปรที่มีความสำคัญต่อค่าเบี้ยประกันสุขภาพมากที่สุดคือตัวแปรอายุก็จะตรงกับสมมติฐานที่ตั้งไว้ว่าอายุนั้นมีความสัมพันธ์ในด้านสุขภาพอย่างเห็นได้ชัดจึงทำให้ค่าเบี้ยประกันสุขภาพนั้นมีแนวโน้มที่เพิ่มขึ้นตามอายุไปด้วยเช่นกัน แต่อย่างไรก็ตามตัวแปรอื่นก็มีผลต่อค่าเบี้ยทำนายด้วยเช่นกัน ซึ่งสามารถระบุได้ว่าตัวแปรใดบ้างที่มีความสำคัญก็จะสามารถทำให้การทำนายแม่นยำยิ่งขึ้น โดยจะสรุปข้อแตกต่างในแต่ละงานวิจัยดังตาราง 1

ตาราง 1 การเปรียบเทียบข้อแตกต่างในแต่ละงานวิจัย

งานวิจัย	แหล่งข้อมูล	วิธีการสร้างแบบจำลอง	วิธีการประเมิน	ผลและค่าประเมิน
Kashish Bhatia และคณะ[11]	ข้อมูลจาก Kaggle จำนวนข้อมูลทั้งหมด 1,338 ข้อมูล[15] ทั้งหมด 7 ตัวแปร 1) อายุ 2) เพศ 3) ดัชนีมวลกาย (BMI) 4) พฤติกรรมการสูบบุหรี่ 5) จำนวนบุตร 6) ภูมิภาคที่อยู่อาศัย 7) ค่าเบี้ยประกัน	Linear Regression	เปอร์เซ็นต์ความแม่นยำ (Accuracy)	Linear Regression ที่มีเปอร์เซ็นต์ความแม่นยำ 81.3
Sudhir Panda และคณะ[12]	ข้อมูลจาก Kaggle จำนวนข้อมูลทั้งหมด 1,338 ข้อมูล[15]	1) Linear Regression 2) Multiple Linear Regression	1) ค่าสัมประสิทธิ์การถดถอยกำลังสอง (R^2) 2) รากที่สองของค่าเฉลี่ยความ	แบบจำลอง Polynomial Regression ให้ผลที่ดีที่สุดตามค่าประเมินดังนี้

งานวิจัย	แหล่งข้อมูล	วิธีการสร้างแบบจำลอง	วิธีการประเมิน	ผลและค่าประเมิน
	ทั้งหมด 7 ตัวแปร 1) อายุ 2) เพศ 3) ดัชนีมวลกาย (BMI) 4) พฤติกรรมการสูบบุหรี่ 5) จำนวนบุตร 6) ภูมิภาคที่อยู่อาศัย 7) ค่าเบี้ยประกัน	3) Polynomial Regression 4) Ridge Regression 5) Lasso Regression	ผิดพลาดกำลังสอง (RMSE) 3) เปอรเซ็นต์ความแม่นยำ (Accuracy)	1) ค่าสัมประสิทธิ์การถดถอยกำลังสอง เท่ากับ 0.8 2) รากที่สองของค่าเฉลี่ยความผิดพลาดกำลังสอง เท่ากับ 5100.53 3) เปอรเซ็นต์ความแม่นยำ เท่ากับ 80.97
Keshav Kaushik และคณะ[5]	ข้อมูลจาก Kaggle จำนวนข้อมูลทั้งหมด 1,338 ข้อมูล[15] ทั้งหมด 7 ตัวแปร 1) อายุ 2) เพศ 3) ดัชนีมวลกาย (BMI) 4) พฤติกรรมการสูบบุหรี่ 5) จำนวนบุตร	1) Linear Regression 2) ANN (Artificial Neuron Network)	1) ค่าสัมประสิทธิ์การถดถอยกำลังสอง (R^2) 2) ค่าเฉลี่ยของความผิดพลาดแบบสัมบูรณ์ (MAE) 3) รากที่สองของค่าเฉลี่ยความผิดพลาดกำลังสอง (RMSE) 4) ค่าความคลาดเคลื่อนเฉลี่ยกำลังสอง (MSE)	แบบจำลอง ANN ให้ผลที่ดีที่สุดตามค่าประเมินดังนี้ 1) ค่าสัมประสิทธิ์การถดถอยกำลังสอง เท่ากับ 0.93 2) ค่าเฉลี่ยของความผิดพลาดแบบสัมบูรณ์ เท่ากับ 0.14 3) รากที่สองของค่าเฉลี่ยความผิดพลาดกำลังสอง เท่ากับ 0.27 4) ค่าความคลาดเคลื่อนเฉลี่ย

งานวิจัย	แหล่งข้อมูล	วิธีการสร้างแบบจำลอง	วิธีการประเมิน	ผลและค่าประเมิน
	6) ภูมิภาคที่อยู่อาศัย 7) ค่าเบี้ยประกัน			กำลังสอง เท่ากับ 0.072
Mounika Karakavalasa และคณะ[6]	ข้อมูลจาก Kaggle จำนวนข้อมูลทั้งหมด 1,338 ข้อมูล[15] มีตัวแปรทั้งหมด 7 ตัวแปร 1) อายุ 2) เพศ 3) ดัชนีมวลกาย (BMI) 4) พฤติกรรมการสูบบุหรี่ 5) จำนวนบุตร 6) ภูมิภาคที่อยู่อาศัย 7) ค่าเบี้ยประกัน	1) XGBoots 2) LightGBM 3) Elastic-Net 4) Lasso Regression 5) Ridge Regression 6) Votting Regression 7) CatBoots Regression 8) Stacking Regression	1) รากที่สองของค่าเฉลี่ยความผิดพลาดกำลังสอง (RMSE) 2) ค่าอธิบายความแปรปรวน (Explained variance score)	1) แบบจำลอง LightGBM ดีที่สุดสำหรับการประเมินผ่านค่ารากที่สองของค่าเฉลี่ยความผิดพลาดกำลังสองมีค่าเท่ากับ 0.78 2) แบบจำลอง Stacking Regression ดีที่สุดสำหรับการประเมินผ่านค่าอธิบายความแปรปรวนมีค่าเท่ากับ 0.89
Ugochukwu Orji และ คณะ[7]	ข้อมูลจาก Kaggle จำนวนข้อมูลทั้งหมด 986 ข้อมูล [3] มีตัวแปรทั้งหมด 11 ตัวแปร	1) Xtream Gradient Boosting (XGBoots) 2) Gradient Boosting Machine (GBM)	1) ค่าสัมประสิทธิ์การถดถอยกำลังสอง (R^2) 2) ค่าเฉลี่ยของความผิดพลาดแบบสัมบูรณ์ (MAE)	1) แบบจำลอง XGBoots ดีที่สุดสำหรับการประเมินผ่านค่ารากที่สองของค่าเฉลี่ยความผิดพลาดกำลังสองมีค่าเท่ากับ 86.47

งานวิจัย	แหล่งข้อมูล	วิธีการสร้างแบบจำลอง	วิธีการประเมิน	ผลและค่าประเมิน
	1) อายุ 2) สภาวะของโรคเบาหวาน 3) ปัญหาเรื่องความดันโลหิต 4) การปลูกถ่ายอวัยวะ 5) มีโรคเรื้อรังหรือไม่ 6) ส่วนสูง 7) น้ำหนัก 8) มีการแพ้หรือไม่ 9) ประวัติการเป็นมะเร็งของครอบครัว 10) จำนวนครั้งการผ่าตัดใหญ่ 11) เบี้ยประกัน	3) Random Forest (RF)	3) รากที่สองของค่าเฉลี่ยความผิดพลาดกำลังสอง (RMSE) 4) ค่าเฉลี่ยความผิดพลาดแบบสัมบูรณ์เชิงเปอร์เซ็นต์ (MAPE)	2) แบบจำลอง Random Forest ดีที่สุดสำหรับการประเมินผ่านค่าเฉลี่ยของความผิดพลาดแบบสัมบูรณ์มีค่าเท่ากับ 1,380 3) แบบจำลอง XGBoots ดีที่สุดสำหรับการประเมินค่าเฉลี่ยความผิดพลาดกำลังสองมีค่าเท่ากับ 2231 4) แบบจำลอง Random Forest ดีที่สุดสำหรับการประเมินผ่านค่าเฉลี่ยความผิดพลาดแบบสัมบูรณ์เชิงเปอร์เซ็นต์มีค่าเท่ากับ 5.8

2.5 แนวทางสำหรับงานวิจัย

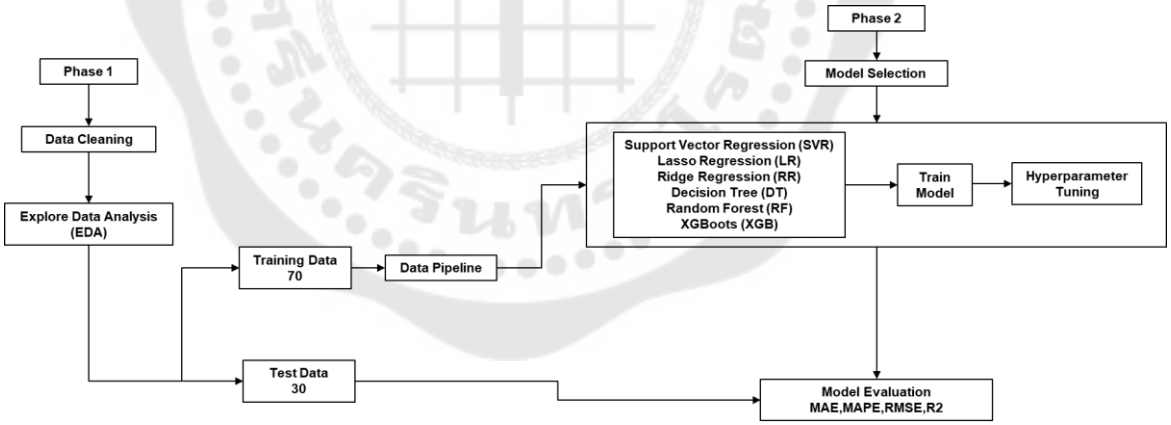
จากการศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้อง พบว่าการเรียนรู้ของเครื่องเป็นแนวทางที่มีศักยภาพในการนำมาทำนายค่าเบี้ยประกันสุขภาพ เนื่องจากสามารถจัดการกับข้อมูลที่หลากหลายและซับซ้อนได้อย่างมีประสิทธิภาพ โดยแบบจำลองที่ได้รับความนิยม ได้แก่ Linear Regression ซึ่งเข้าใจง่ายและเหมาะสำหรับความสัมพันธ์เชิงเส้น, Random Forest ที่สามารถจัดการข้อมูลเชิงซ้อนและลดความเอนเอียงของข้อมูล, และ XGBoost ที่เด่นด้านความแม่นยำและประสิทธิภาพในการทำงาน ในงานวิจัยนี้ ผู้วิจัยจะนำชุดข้อมูลของ Ugochukwu Orji และ คณะ ที่ใช้ในการวิจัยไปต่อยอดโดยเพิ่มแบบจำลองในส่วนของ Regression ได้แก่ Support Vector Regression, Lasso Regression และ Ridge Regression เข้าไปเพื่อให้เกิดการเปรียบเทียบในแต่ละแบบจำลองให้มากยิ่งขึ้นและจะเพิ่มในส่วนของแบบจำลอง Decision Tree เข้ามาและยังคงแบบจำลอง Random Forest และ XGBoots คงไว้อยู่ โดยผู้วิจัยจะนำแบบจำลองเหล่านี้มาเปรียบเทียบสมรรถนะเพื่อพัฒนาแบบจำลองที่เหมาะสมที่สุดสำหรับการทำนายค่าเบี้ยประกัน โดยบทที่ 3 จะนำเสนอวิธีการดำเนินงานวิจัย ตั้งแต่การเตรียมข้อมูล การเลือกคุณลักษณะ การสร้างและฝึกแบบจำลอง ตลอดจนการประเมินผล

บทที่ 3
วิธีดำเนินการวิจัย

ในบทนี้จะกล่าวถึงวิธีดำเนินการวิจัยเพื่อให้ดำเนินงานอย่างเป็นระบบ โดยเริ่มจากการวางแผนกระบวนการสร้างแบบจำลอง จากนั้นดำเนินการทำความสะอาดและเตรียมข้อมูลเพื่อให้เหมาะสมต่อการวิเคราะห์ ก่อนเข้าสู่ขั้นตอนการวิเคราะห์ข้อมูลเชิงสำรวจเพื่อทำความเข้าใจโครงสร้างและลักษณะของข้อมูลอย่างลึกซึ้ง ต่อมาได้ดำเนินการสร้างแบบจำลองด้วยการเรียนรู้ของเครื่อง และทำการปรับจูนค่าไฮเปอร์พารามิเตอร์เพื่อเพิ่มประสิทธิภาพของแบบจำลอง สุดท้ายคือการประเมินผลลัพธ์ของแบบจำลองเพื่อให้มั่นใจว่าแบบจำลองที่ได้มีความแม่นยำและเหมาะสมต่อการนำไปใช้งานในบริบทที่เกี่ยวข้อง

3.1 **แผนการสร้างแบบจำลอง** (Experiment Workflow)

ก่อนเริ่มการศึกษา ผู้วิจัยได้วางแผนการสร้างแบบจำลองอย่างละเอียด เพื่อกำหนดแนวทางและขั้นตอนที่ชัดเจนสำหรับการดำเนินการวิจัย



ภาพประกอบ 2 แผนผังการสร้างแบบจำลอง

โดยจากภาพประกอบ 2 จะแบ่งการสร้างแบบจำลองออกเป็น 2 ขั้นตอนใหญ่ ขั้นตอนแรกจะทำการศึกษาข้อมูลที่จะนำมาใช้เพื่อให้เข้าใจถึงคุณลักษณะของข้อมูลว่าจะต้องมีการจัดการข้อมูลให้พร้อมใช้งานใน มีข้อมูลสูญหายหรือไม่ และมีคุณลักษณะใดบ้างที่สำคัญสำหรับการวิเคราะห์ จากนั้นจะทำการจัดการข้อมูลให้พร้อมใช้งาน เช่น การทำความสะอาดข้อมูล (data cleaning) การปรับสเกล การแปลงรูปแบบข้อมูล (data transformation) การเลือกตัวแปร (feature

selection) ที่เหมาะสม รวมถึงการทำ data pipeline เพื่อป้องกันข้อมูลรั่วไหล เมื่อได้ข้อมูลมาแล้วจากขั้นตอนที่ 1 ในขั้นตอนที่ 2 นี้จะเริ่มการฝึก (training) ข้อมูล โดยใช้ข้อมูลที่เตรียมไว้มาทดลองสร้างแบบจำลองการเรียนรู้ของเครื่องหลายประเภท และมีการทดลองปรับค่าตัวแปร (hyperparameter tuning) เพื่อหาพารามิเตอร์ที่เหมาะสมในแต่ละแบบจำลอง จากนั้นทำการประเมินประสิทธิภาพของแบบจำลองแต่ละแบบ เมื่อได้ผลการประเมินแล้วจะเลือกแบบจำลองที่ให้ผลลัพธ์ดีที่สุดเพื่อนำไปใช้ในกระบวนถัดไป

3.2 **การทำความสะอาดข้อมูลและเตรียมข้อมูล** (Data Cleaning and Preprocessing)

ชุดข้อมูลที่นำมาใช้มีชื่อว่า The medical insurance cost ชุดข้อมูลนี้มาจากเว็บไซต์ KAGGLE [(3)] ซึ่งเป็นเว็บไซต์สำหรับชุดข้อมูลแบบเปิด (Open Source) มีจำนวนข้อมูลทั้งหมด 986 ข้อมูล มีตัวแปรทั้งหมด 11 ตัวแปร แสดงดังตาราง 2 โดยจะนำชุดข้อมูลไปเก็บไว้ใน google drive แล้วทำการดึงข้อมูลจาก google drive มายัง google colab เพื่อทำการแสดงผลข้อมูลและสร้างแบบจำลองต่อไปโค้ดสำหรับการดึงข้อมูลแสดงดังภาพประกอบ 3

ตาราง 2 รายละเอียดของชุดข้อมูล

ลำดับ	ชื่อตัวแปร	คำอธิบาย	ชนิดข้อมูล
1	Age	อายุ	ตัวเลข
2	Diabetes	สภาวะของโรคเบาหวาน (เป็น/ไม่เป็น)	ตัวเลข
3	BloodPressureProblems	ปัญหาเรื่องความดันโลหิต (มี/ไม่มี)	ตัวเลข
4	AnyTransplants	การปลูกถ่ายอวัยวะ (เคย/ไม่เคย)	ตัวเลข
5	AnyChronicDiseases	มีโรคเรื้อรังหรือไม่(มี/ไม่มี)	ตัวเลข
6	Height	ส่วนสูง	ตัวเลข
7	Weight	น้ำหนัก	ตัวเลข
8	KnowAllergies	มีการแพ้หรือไม่ (มี/ไม่มี)	ตัวเลข
9	HistoryOfCancerInFamily	ประวัติการเป็นมะเร็งของครอบครัว (มี/ไม่มี)	ตัวเลข
10	NumberOfMajorSurgeries	จำนวนครั้งการผ่าตัดใหญ่	ตัวเลข
11	Premium Price (INR)	เบี้ยประกัน (รูปีอินเดีย)	ตัวเลข

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[3] data_path = "/content/drive/MyDrive/Project_Jidapa/Dataset"
    data = pd.read_csv(f"{data_path}/2. Medicalpremium for paper 3.csv")
```

```
# Check data type of each column
data.dtypes
```

ภาพประกอบ 3 โค้ดสำหรับการดึงข้อมูล

ชุดข้อมูลที่ได้นำมาศึกษานั้นมีแหล่งมาจากประเทศอินเดียฉะนั้นค่าเบี้ยประกันภัยจะอยู่ในสกุลเงินรูปี ของประเทศอินเดีย จากนั้นผู้วิจัยได้มีการทำความสะอาดข้อมูลในเรื่องดูว่ามีค่าว่างในชุดข้อมูลหรือไม่ แสดงดังภาพประกอบ 4 ดูในเรื่องชนิดของข้อมูลว่าสอดคล้องกับที่ควรจะเป็นหรือไม่ แสดงดังภาพประกอบ 5 หลังจากทำการตรวจสอบทั้งสองอย่างพบว่าชุดข้อมูลของนั้นไม่มีค่าว่างและชนิดของข้อมูลก็ถูกต้อง จากนั้นจึงมาดูในเรื่องการเตรียมข้อมูลก่อนนำไปฝึกเนื่องจากข้อมูลที่จะต้องนำไปฝึกนั้นต้องเป็นตัวเลขทั้งหมดจึงจำเป็นต้องทำการ แปลงรหัสข้อมูล (encoding) สำหรับตัวแปร Diabetes, BloodPressureProblems, AnyTransplants, AnyChronicDiseases, KnowAllergies, HistoryOfCancerInFamily ข้อมูลก่อนนำไปใช้ซึ่งข้อมูลได้ทำการ แปลงรหัสข้อมูลเรียบร้อยแสดงตัวอย่างข้อมูลดังภาพประกอบ 6

```
[ ] # Show null values
    print(data.isnull().sum())
```

```
Age                        0
Diabetes                   0
BloodPressureProblems      0
AnyTransplants             0
AnyChronicDiseases         0
Height                     0
Weight                     0
KnownAllergies             0
HistoryOfCancerInFamily    0
NumberOfMajorSurgeries     0
PremiumPrice               0
dtype: int64
```

ภาพประกอบ 4 ตรวจสอบค่าว่างในชุดข้อมูล

```
# Check data type of each column
data.dtypes
```

	0
Age	int64
Diabetes	int64
BloodPressureProblems	int64
AnyTransplants	int64
AnyChronicDiseases	int64
Height	int64
Weight	int64
KnownAllergies	int64
HistoryOfCancerInFamily	int64
NumberOfMajorSurgeries	int64
PremiumPrice	int64

dtype: object

ภาพประกอบ 5 ตรวจสอบชนิดข้อมูล

```
[ ] data
```

	Age	Diabetes	BloodPressureProblems	AnyTransplants	AnyChronicDiseases	Height	Weight	KnownAllergies	HistoryOfCancerInFamily	NumberOfMajorSurgeries	PremiumPrice
0	45	0	0	0	0	155	57	0	0	0	25000
1	60	1	0	0	0	180	73	0	0	0	29000
2	36	1	1	0	0	158	59	0	0	1	23000
3	52	1	1	0	1	183	93	0	0	2	28000
4	38	0	0	0	1	166	88	0	0	1	23000
...	...	...	...	...	...	...	...	...	...	...	...
981	18	0	0	0	0	169	67	0	0	0	15000
982	64	1	1	0	0	153	70	0	0	3	28000
983	56	0	1	0	0	155	71	0	0	1	29000
984	47	1	1	0	0	158	73	1	0	1	39000
985	21	0	0	0	0	158	75	1	0	1	15000

986 rows × 11 columns

ภาพประกอบ 6 ตัวอย่างของชุดข้อมูล

3.3 **การวิเคราะห์ข้อมูลเชิงสำรวจ** (Exploratory Data Analysis)

ผู้วิจัยได้ทำการสำรวจข้อมูลเบื้องต้นจากชุดข้อมูลเพื่อดูลักษณะของข้อมูลทั้งทางสถิติและทางเชิงพรรณนาโดยจะดูว่าในแต่ละ ตัวแปร มีการอธิบายข้อมูลเชิงสถิติอย่างไรบ้างแสดงดังภาพประกอบ 7

```
[ ] data.describe()
```

	Age	Diabetes	BloodPressureProblems	AnyTransplants	AnyChronicDiseases	Height	Weight	KnownAllergies	HistoryOfCancerInFamily	NumberOfMajorSurgeries	Premium Price (INR)
count	986.000000	986.000000	986.000000	986.000000	986.000000	986.000000	986.000000	986.000000	986.000000	986.000000	986.000000
mean	41.745436	0.419878	0.468560	0.055781	0.180527	168.182556	76.950304	0.215010	0.117647	0.667343	24336.713996
std	13.963371	0.493789	0.499264	0.229615	0.384821	10.098155	14.265096	0.411038	0.322353	0.749205	6248.184382
min	18.000000	0.000000	0.000000	0.000000	0.000000	145.000000	51.000000	0.000000	0.000000	0.000000	15000.000000
25%	30.000000	0.000000	0.000000	0.000000	0.000000	161.000000	67.000000	0.000000	0.000000	0.000000	21000.000000
50%	42.000000	0.000000	0.000000	0.000000	0.000000	168.000000	75.000000	0.000000	0.000000	1.000000	23000.000000
75%	53.000000	1.000000	1.000000	0.000000	0.000000	176.000000	87.000000	0.000000	0.000000	1.000000	28000.000000
max	66.000000	1.000000	1.000000	1.000000	1.000000	188.000000	132.000000	1.000000	1.000000	3.000000	40000.000000

ภาพประกอบ 7 อธิบายชุดข้อมูลด้วยค่าทางสถิติ

จากภาพประกอบ 7 แสดงเป็นตารางสรุปค่าทางสถิติ (Descriptive Statistics) ของข้อมูลจำนวน 986 แถว โดยค่าทางสถิติที่แสดง ได้แก่ count (จำนวนข้อมูลที่ไม่เป็นค่าว่าง), mean (ค่าเฉลี่ย), std (ส่วนเบี่ยงเบนมาตรฐาน), min (ค่าน้อยที่สุด), 25% (ควอไทล์แรก), 50% (มัธยฐาน), 75% (ควอไทล์ที่สาม), และ max (ค่ามากที่สุด) ซึ่งค่าต่าง ๆ เหล่านี้ช่วยให้เข้าใจการกระจายตัวและลักษณะของข้อมูลได้อย่างชัดเจน โดยสามารถอธิบายได้อย่างละเอียดดังนี้

ทุกคอลัมน์มี count เท่ากับ 986 แสดงว่าไม่มีข้อมูลว่างในชุดข้อมูลนี้เลย โดยตัวแปรแรกคือ Age (อายุ) มีค่าเฉลี่ยอยู่ที่ประมาณ 41.75 ปี แสดงว่าค่าเฉลี่ยของกลุ่มตัวอย่างอยู่ในช่วงวัยกลางคน ค่าส่วนเบี่ยงเบนมาตรฐานประมาณ 13.96 บ่งบอกว่ามีความกระจายของอายุอยู่พอสมควร โดยอายุต่ำสุดคือ 18 ปี และสูงสุดคือ 66 ปี ควอไทล์แรก (25%) อยู่ที่ 30 ปี มัธยฐานอยู่ที่ 42 ปี และควอไทล์ที่สามอยู่ที่ 53 ปี ซึ่งแสดงว่ากลุ่มข้อมูลมีการกระจายที่ค่อนข้างสมดุล

ตัวแปร Diabetes, BloodPressureProblems, AnyTransplants, AnyChronicDiseases, KnownAllergies, และ HistoryOfCancerInFamily มีค่า min เท่ากับ 0 และ max เท่ากับ 1 เนื่องมาจากตัวแปรเหล่านี้เป็นข้อมูลประเภทใช่/ไม่ใช่ หรือมี/ไม่มี โดยค่าเฉลี่ยของ Diabetes เท่ากับ 0.419 แสดงว่าประมาณ 41.9% ของกลุ่มตัวอย่างมีภาวะเบาหวาน ในขณะที่ BloodPressureProblems มีค่าเฉลี่ยประมาณ 0.469 หรือประมาณ 46.9% ของกลุ่มตัวอย่างมีปัญหาเรื่องความดันโลหิต AnyTransplants มีค่าเฉลี่ยต่ำเพียง 0.055 แสดงว่ามีเพียงประมาณ 5.5% ที่เคยรับการปลูกถ่ายอวัยวะ ส่วน AnyChronicDiseases มีค่าเฉลี่ย 0.180 แสดงว่าประมาณ

18% ของกลุ่มตัวอย่างมีโรคเรื้อรัง KnownAllergies มีค่าเฉลี่ย 0.215 ซึ่งหมายถึงมีคนที่มีประวัติอาการแพ้ประมาณ 21.5% ขณะที่ HistoryOfCancerInFamily อยู่ที่ 0.117 หรือ 11.7% แสดงว่ามีประวัติมะเร็งในครอบครัวไม่มากนัก

สำหรับข้อมูลเชิงปริมาณอย่าง Height (ส่วนสูง) มีค่าเฉลี่ยอยู่ที่ 168.18 เซนติเมตร ส่วนเบี่ยงเบนมาตรฐานคือ 10.10 เซนติเมตร ส่วนสูงต่ำสุดคือ 145 ซม. และสูงสุดคือ 188 ซม. ควอไทล์แรกอยู่ที่ 161 ซม. มัธยฐานอยู่ที่ 170 ซม. และควอไทล์ที่สามอยู่ที่ 176 ซม. แสดงว่ากลุ่มตัวอย่างส่วนใหญ่อยู่ในช่วงความสูงมาตรฐานของผู้ใหญ่เพศชาย/หญิงในเอเชีย

น้ำหนัก (Weight) มีค่าเฉลี่ยอยู่ที่ 76.9 กิโลกรัม ส่วนเบี่ยงเบนมาตรฐานอยู่ที่ 14.27 กิโลกรัม ค่าน้ำหนักต่ำสุดคือ 51 กก. และสูงสุดคือ 132 กก. โดยมีควอไทล์แรกที่ 67 กก., มัธยฐานที่ 75 กก. และควอไทล์ที่สามที่ 87 กก. บ่งชี้ว่ามีการกระจายของน้ำหนักที่ค่อนข้างกว้าง และมีบางคนที่น้ำหนักสูงมาก

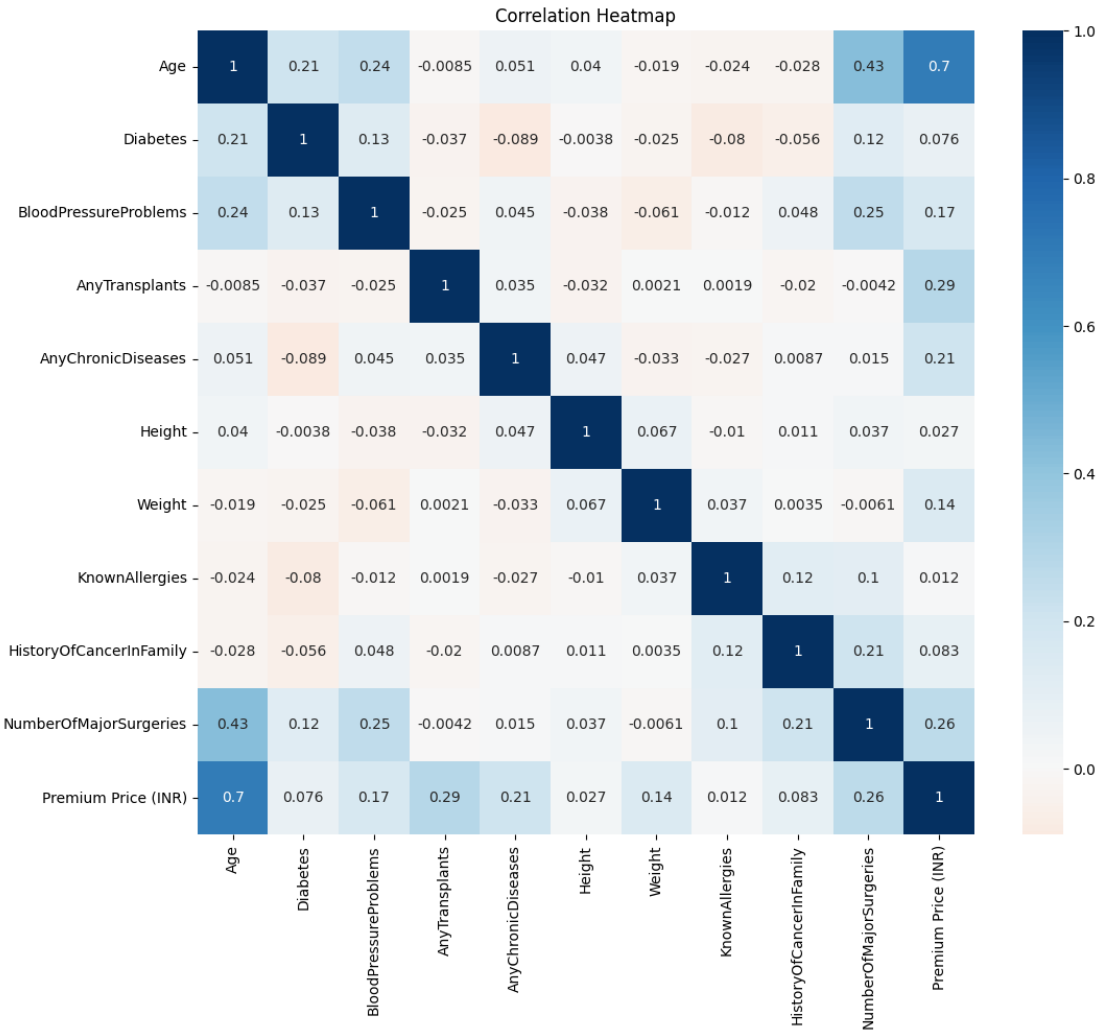
จำนวนการผ่าตัดใหญ่ (NumberOfMajorSurgeries) มีค่าเฉลี่ยประมาณ 0.667 และมีค่าสูงสุดถึง 3 ครั้ง มัธยฐานอยู่ที่ 1 ซึ่งแปลว่ากลุ่มตัวอย่างส่วนใหญ่เคยผ่านการผ่าตัดใหญ่มาแล้วอย่างน้อย 1 ครั้ง

สุดท้ายคือ Premium Price (INR) หรือค่าเบี้ยประกันสุขภาพ (อินเดียรูปี) มีค่าเฉลี่ยอยู่ที่ 24,336.71 รูปีอินเดีย มีส่วนเบี่ยงเบนมาตรฐานค่อนข้างสูงคือ 6,248.18 รูปี แสดงว่ามีความหลากหลายของราคาในกลุ่มตัวอย่าง โดยมีค่าต่ำสุดที่ 15,000 รูปี และสูงสุดที่ 40,000 รูปี ค่าควอไทล์แรกอยู่ที่ 21,000 มัธยฐานอยู่ที่ 24,000 และควอไทล์ที่สามอยู่ที่ 28,000 แสดงว่ากลุ่มคนส่วนใหญ่จ่ายค่าเบี้ยประกันในช่วง 2 หมื่นต้นถึงปลายรูปี โดยมีบางกลุ่มที่จ่ายสูงกว่าเฉลี่ยอย่างมีนัยสำคัญ

จากการดูค่าทางสถิติเบื้องต้นทำให้ทราบว่าตัวแปรแต่ละตัวแปรมีลักษณะอย่างไรแต่ยังไม่สามารถเห็นภาพได้ว่าแต่ละตัวแปรนั้นมีความสัมพันธ์กันอย่างไร ดังนั้นจะใช้ค่าสัมประสิทธิ์ความสัมพันธ์ (Correlation) ดูความสัมพันธ์ของแต่ละตัวแปรกับค่าเบี้ยประกัน ซึ่งจะแสดงผ่านกราฟแผนที่ความร้อนแสดงดังภาพประกอบ 8 และ 9

```python
# Calculate the correlation matrix
correlation_matrix = data.corr()

# Create a heatmap with green and blue tones
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='RdBu', center=0)  # Use RdBu colormap for blue and red tones
plt.title('Correlation Heatmap')
plt.show()
```

ภาพประกอบ 8 โค้ดสำหรับการสร้างค่าสัมประสิทธิ์ความสัมพันธ์และแผนที่ความร้อน



ภาพประกอบ 9 แผนที่ความร้อนแสดงสัมประสิทธิ์ความสัมพันธ์

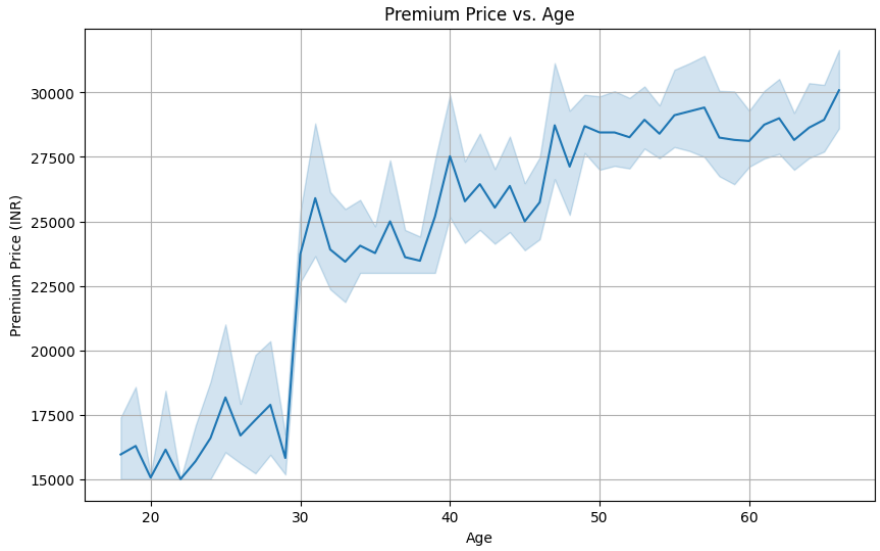
จากภาพประกอบ 9 จะเห็นได้ค่าว่าค่าความสัมพันธ์ระหว่าง Premium Price (INR) กับอายุมีค่าความสัมพันธ์อยู่ที่ 0.7 ซึ่งมากที่สุดในบรรดา ตัวแปร อื่นๆ แสดงว่าสอง ตัวแปร นี้มีความสัมพันธ์ที่เกี่ยวข้องกันอย่างมาก ถัดมานอกจาก ตัวแปร อายุแล้วค่าความสัมพันธ์ของ ตัวแปรที่รองลงมาก็คือ AnyTranplants มีค่าความสัมพันธ์อยู่ที่ 0.29 และรองลงมาอีก ตัวแปร คือ NumberOfMajorSurgeries ค่าความสัมพันธ์อยู่ที่ 0.26 กล่าวคือยิ่งค่าความสัมพันธ์มีค่าเป็นบวกและมีค่าเข้าใกล้หนึ่งแสดงว่าทั้งสองตัวแปรมีความเกี่ยวข้องกันซึ่งสามารถนำค่านี้ไปใช้ในการทำ feature selection เพื่อเลือก ตัวแปร มาใช้ในการทำแบบจำลองต่อไปเพื่อให้ได้ผลลัพธ์ที่ดีที่สุดแต่จากภาพประกอบ 9 ค่าความสัมพันธ์ของแต่ละ ตัวแปร ที่มีต่อ Premium Price (INR) มีค่าเป็นบวกทั้งหมดอาจกล่าวได้ว่าทุกๆ ตัวแปร ที่แสดงล้วนมีความเกี่ยวข้องหรือส่งผลต่อค่า Premium Price

(INR) ดังนั้นในงานวิจัยนี้ผู้วิจัยอาจจะยังไม่เลือกใช้ feature selection เข้ามาเกี่ยวข้องก่อนนำชุดข้อมูลไปฝึกแบบจำลอง

จากการสำรวจข้อมูล กราฟถัดมาจะแสดงให้เห็นถึงความสัมพันธ์ระหว่างอายุและเบี้ยประกันสุขภาพ ที่ตั้งสมมติฐานไว้ว่า ถ้าอายุยังน้อยเบี้ยประกันก็จะมีค่าที่ต่ำไปตามกันและขณะเดียวกัน ถ้าอายุเยอะเบี้ยประกันก็จะมีค่าเบี้ยที่สูงตามกันไป แสดงให้เห็นดังภาพประกอบ 10 และ 11

```
[ ]  #Line chart between Premium Price and Age
     plt.figure(figsize=(10, 6))
     sns.lineplot(x='Age', y='Premium Price (INR)', data=data)
     plt.title('Premium Price vs. Age')
     plt.xlabel('Age')
     plt.ylabel('Premium Price (INR)')
     plt.grid(True)
     plt.show()
```

ภาพประกอบ 10 โค้ดการสร้างกราฟเส้นของอายุและค่าเบี้ยประกันสุขภาพ



ภาพประกอบ 11 กราฟเส้นแสดงความสัมพันธ์ระหว่างเบี้ยประกันสุขภาพและอายุ

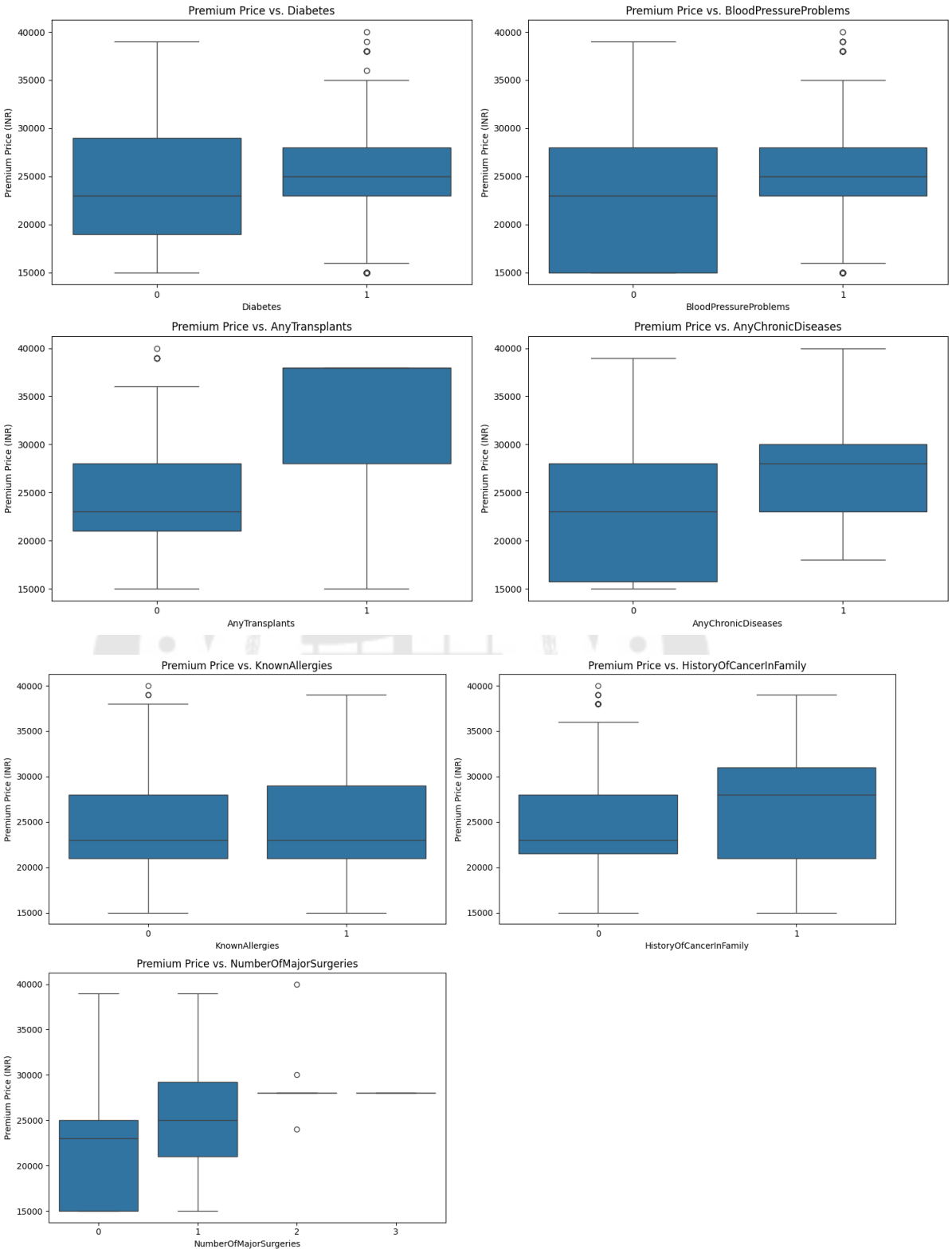
จากภาพประกอบ 11 แสดงให้เห็นอย่างชัดเจนเลยว่าเมื่ออายุน้อยค่าเบี้ยประกันสุขภาพก็จะมีค่าที่ต่ำ และเมื่ออายุเพิ่มขึ้นค่าเบี้ยประกันก็จะเพิ่มขึ้นตามแสดงดังภาพประกอบ 11 แต่จากกราฟจะสังเกตได้ว่าช่วงอายุ ประมาณ 20 ปลายปลายจนถึง 30 ต้นๆ ค่าของเบี้ยประกันมีการเปลี่ยนแปลงอย่างสูง แสดงว่า ตัวแปรอายุ มีการเปลี่ยนแปลงอย่างเป็นนัยยะสำคัญ เมื่ออายุ 30 ปี

เป็นต้นไป และกราฟที่แสดงความสัมพันธ์นั้นไม่ได้มีความเป็นเส้นตรงหรือแสดงให้เห็นรูปแบบที่ชัดเจนของการทำค่าเบี้ยประกันสุขภาพกับอายุแต่อย่างไรก็ตามยังมีตัวแปรอื่นที่จะต้องนำมาพิจารณาด้วย

นอกจากตัวแปรอายุแล้วจะมาดูว่าตัวแปรตัวอื่นมีความสัมพันธ์กับค่าเบี้ยประกันสุขภาพอย่างไรบ้างโดยจะใช้กราฟกล่อง (box plot) เพื่อดูการกระจายตัวของข้อมูลและดูว่าชุดข้อมูลในงานวิจัยมีความผิดปกติที่ใดบ้าง (outlier) แสดงดังภาพประกอบ 12 และ 13

```python
# Create a figure with 2 rows and 2 columns of subplots (4 plots total)
fig, axes = plt.subplots(2, 2, figsize=(15, 10))  # Adjust figsize as needed

# Flatten the axes array for easier indexing
axes = axes.ravel()

# Iterate through the first four columns and create box plots
for i in range(4):
    sns.boxplot(x=columns_to_plot[i], y='Premium Price (INR)', data=data, ax=axes[i])
    axes[i].set_title(f'Premium Price vs. {columns_to_plot[i]}')
    axes[i].set_xlabel(columns_to_plot[i])
    axes[i].set_ylabel('Premium Price (INR)')

plt.tight_layout() # Ensures proper spacing between plots
plt.show()

# Create a second figure with 2 rows and 2 columns of subplots (4 plots total)
fig, axes = plt.subplots(2, 2, figsize=(15, 10))
axes = axes.ravel()

# Iterate through the remaining three columns and create box plots
for i in range(3):
    sns.boxplot(x=columns_to_plot[i+4], y='Premium Price (INR)', data=data, ax=axes[i])
    axes[i].set_title(f'Premium Price vs. {columns_to_plot[i+4]}')
    axes[i].set_xlabel(columns_to_plot[i+4])
    axes[i].set_ylabel('Premium Price (INR)')
axes[3].axis('off') # Removing the last subplot as we only need 3 for the remaining columns.

plt.tight_layout()
plt.show()
```

ภาพประกอบ 12 โค้ดการสร้างกราฟกล่อง



ภาพประกอบ 13 กราฟกล่องระหว่างค่าเบี้ยประกันและตัวแปร

จากภาพประกอบ 13 แสดงให้เห็นความสัมพันธ์ระหว่างค่าเบี้ยประกันในแต่ละตัวแปรซึ่งตัวแปร Diabetes BloodPressureProblems AnyTransplants AnyChronicDiseases KnowAllergies HistoryOfCancerInFamily NumberOfMajorSurgeries จะเป็นตัวแปรที่แสดงว่ามี หรือ ไม่มี ภาวะหรือปัญหาสุขภาพต่าง ๆ ยกเว้น ตัวแปร NumberOfMajorSurgerie ที่จะบอกจำนวนของการผ่าตัดว่าเคยมาแล้วกี่ครั้งซึ่งสูงสุดคือ 3 ครั้ง

เริ่มจากตัวแปร Diabetes จากภาพประกอบ 13 กราฟระหว่าง Premium Price และ Diabetes จะเห็นว่า ผู้ที่มีภาวะเบาหวาน (ค่า เท่ากับ 1) มีค่ากลางของเบี้ยประกัน (median) สูงกว่ากลุ่มที่ไม่มีเล็กน้อย แต่ช่วง interquartile (Q1-Q3) ของทั้งสองกลุ่มยังมีการทับซ้อนกันพอสมควร ซึ่งหมายความว่าเบี้ยประกันอาจสูงขึ้นบ้างสำหรับผู้ที่เป็นเบาหวาน แต่ไม่ใช่ความแตกต่างที่ชัดเจนมาก ดังนั้นตัวแปร Diabetes อาจจะไม่ได้ส่งผลต่อค่าเบี้ยประกันมากนัก

ตัวแปร BloodPressureProblems ลักษณะของกลุ่มนี้คล้ายกับตัวแปร Diabetes คือ กลุ่มที่มีปัญหาความดัน (ค่า เท่ากับ 1) มีเบี้ยประกันเฉลี่ยและค่ากลางสูงกว่ากลุ่มที่ไม่มีเล็กน้อย อย่างไรก็ตาม การกระจายของข้อมูลยังค่อนข้างทับซ้อนกัน และมี outliers จำนวนหนึ่งทั้งสองฝั่งถัดมา

ตัวแปร AnyTransplants จากภาพประกอบ 13 กราฟระหว่าง Premium Price และ AnyTransplants พบความแตกต่างชัดเจนระหว่างกลุ่มที่เคยปลูกถ่ายอวัยวะ (ค่า เท่ากับ 1) กับกลุ่มที่ไม่เคย (ค่า เท่ากับ 0) โดยกลุ่มที่เคยปลูกถ่ายมีเบี้ยประกันสูงกว่าอย่างมีนัยสำคัญ ทั้งในแง่ของค่ากลาง และช่วงการกระจายของข้อมูล ซึ่งสะท้อนถึงความเสี่ยงทางการแพทย์ที่สูงขึ้น และต้นทุนในการประกันที่มากขึ้น แสดงว่าตัวแปร AnyTransplants มีผลต่อค่าเบี้ยประกันอย่างมีนัยยะสำคัญ

ตัวแปร AnyChronicDiseases ผู้ที่มีโรคเรื้อรังแสดงให้เห็นถึงแนวโน้มที่มีเบี้ยประกันสูงกว่าโดยค่ากลางของเบี้ยประกันในกลุ่มที่มีโรคเรื้อรังสูงกว่ากลุ่มไม่มีอย่างชัดเจน และยังมีการกระจายข้อมูลกว้างขึ้นซึ่งบ่งชี้ว่าเบี้ยประกันในกลุ่มนี้มีความหลากหลายมากขึ้นตามประเภทและความรุนแรงของโรค

ตัวแปร KnowAllergies สำหรับผู้ที่มีหรือไม่มีอาการแพ้ ค่าเบี้ยประกันไม่มีความแตกต่างอย่างชัดเจน ทั้งค่ากลางและช่วงการกระจายค่อนข้างใกล้เคียงกันซึ่งอาจบ่งชี้ว่าอาการแพ้ทั่วไปไม่ได้ส่งผลต่อเบี้ยประกันมากนัก

ตัวแปร HistoryOfCancerInFamily ผู้ที่มีประวัติมะเร็งในครอบครัวมีแนวโน้มค่าเบี้ยประกันสูงขึ้นเล็กน้อยเมื่อเทียบกับผู้ที่ไม่มีประวัติ แต่ความแตกต่างไม่มาก และช่วงการกระจายยัง

ทับซ้อนกันสูง แสดงว่าความเสี่ยงจากพันธุกรรมอาจถูกรวมไว้ในลักษณะอื่น หรือยังไม่ถูกใช้ในการกำหนดเบี้ยอย่างจริงจัง

ตัวแปร NumberOfMajorSurgeries ผู้ที่มีประวัติมะเร็งในครอบครัวมีแนวโน้มค่าเบี้ยประกันสูงขึ้นเล็กน้อยเมื่อเทียบกับผู้ที่ไม่มีประวัติ แต่ความแตกต่างไม่มาก และช่วงการกระจายยังทับซ้อนกันสูง แสดงว่าความเสี่ยงจากพันธุกรรมอาจจะไม่ได้นำมาพิจารณาในการคิดค่าเบี้ยประกัน

จากภาพประกอบ 13 สรุปแล้วตัวแปรที่มีผลต่อค่าเบี้ยประกันอย่างมีนัยสำคัญคือตัวแปร AnyTransplants AnyChronicDiseases และ NumberOfMajorSurgeries ในส่วนของตัวแปร Diabetes BloodPressureProblems และ HistoryOfCancerInFamily มีผลต่อค่าเบี้ยประกันปานกลาง และสุดท้ายตัวแปร KnowAllergies ค่อนข้างที่จะไม่ค่อยมีผลต่อค่าเบี้ยประกันสุขภาพเท่าไร แต่อย่างไรก็ตามการสำรวจข้อมูลทำให้ได้เห็นความสัมพันธ์ของตัวแปรได้เห็นภาพมากยิ่งขึ้นซึ่งก็จะช่วยให้วิเคราะห์ผลได้อย่างเป็นเหตุเป็นผลกัน

3.4 การสร้างแบบจำลอง (Machine Learning Model)

การสร้างแบบจำลองในงานวิจัยนี้ดำเนินการผ่านแพลตฟอร์ม Google Colab โดยเริ่มต้นจากการติดตั้งไลบรารีที่จำเป็น ซึ่งประกอบด้วยโมดูลสำหรับใช้งาน XGBoost ซึ่งจะต้องทำการติดตั้งก่อนถึงจะเรียกใช้งานจาก Scikit-learn ได้ และไลบรารีจาก Scikit-learn ที่เกี่ยวข้องกับการแบ่งข้อมูล ได้แก่ train_test_split, GridSearchCV, cross_val_score และ KFold ต่อมาผู้วิจัยได้เรียกใช้งานโมดูลสำหรับสร้างแบบจำลองต่าง ๆ ได้แก่ LinearRegression, Lasso, Ridge, SVR, DecisionTreeRegressor, RandomForestRegressor XGBRegressor และ XGBClassifier ถัดมาจะเป็นไลบรารีเกี่ยวข้องกับการประมวลผลข้อมูลเบื้องต้น ได้แก่ scale, StandardScaler, TransformedTargetRegressor, make_pipeline และ Pipeline และสุดท้ายคือไลบรารีที่ที่ใช้สำหรับการประเมินผลลัพธ์ของโมแบบจำลองได้แก่ mean_squared_error, r2_score, mean_absolute_error และ mean_absolute_percentage_error ไลบรารีที่กล่าวมาข้างต้นทั้งหมดแสดงดังภาพประกอบ 14

```
[ ] !pip install xgboost
    !pip install --upgrade scikit-learn xgboost
```

Show hidden output

```
[ ] # Import libraries
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LinearRegression, Lasso
    from sklearn.preprocessing import PolynomialFeatures, scale
    from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
    import xgboost as xgb
    from xgboost import XGBRegressor
    from xgboost import XGBClassifier
    from sklearn.preprocessing import StandardScaler
    from sklearn.linear_model import Ridge
    from sklearn.svm import SVR
    from sklearn.ensemble import RandomForestRegressor
    from sklearn.tree import DecisionTreeRegressor
    from sklearn.model_selection import GridSearchCV
    from sklearn.model_selection import cross_val_score
    from sklearn import metrics
    from sklearn.model_selection import KFold
    from sklearn.pipeline import make_pipeline
    from sklearn.metrics import mean_absolute_percentage_error
    from sklearn.pipeline import Pipeline
    from sklearn.pipeline import Pipeline
    from sklearn.compose import TransformedTargetRegressor
    from sklearn.model_selection import GridSearchCV, cross_val_score
    from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, r2_score
```

ภาพประกอบ 14 library ที่จำเป็นสำหรับการสร้างแบบจำลอง

ผู้วิจัยได้ทำการแบ่งสัดส่วนของการฝึกและทดสอบข้อมูลอยู่ที่ 70 ต่อ 30 และมีการตั้งค่า seed ไว้ที่ค่าหนึ่งสำหรับใช้ใน random state เพื่อให้ผลลัพธ์ในทุก ๆ รอบยังคงเหมือนเดิมแสดงดังภาพประกอบ 15

```
[ ] # Set seed for reproducibility
    seed = 42
```

แบ่งสัดส่วนของ train 70 test 30

```
[ ] # Split data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=seed)
```

```
[ ] print(X.shape)
    print(y.shape)
    # print(X_train_scaled.shape)
    print(X_train.shape)
    print(y_train.shape)
    # print(X_test_scaled.shape)
    print(X_test.shape)
    # print(X_test_scaled.shape)
    print(y_test.shape)
```

```
(986, 10)
(986,)
(690, 10)
(690,)
(296, 10)
(296,)
```

ภาพประกอบ 15 โค้ดสำหรับการแบ่งข้อมูล

โดยผู้วิจัยจะทำการศึกษาประสิทธิภาพของแบบจำลองทั้งหมด 6 แบบคือ ดังนี้ 1) Support Vector Regression (SVR) 2) Lasso Regression 3) Ridge Regression 4) Decision Tree 5) Random Forest และ 6) XGBoots (eXtream Gradient Boosting) ซึ่งทั้ง 6 แบบจำลองจะผ่านการฝึกและทดสอบและประเมินผลที่ได้ออกมา 1 รอบก่อนหลังจากนั้นจะทำการปรับค่าพารามิเตอร์ของแต่ละแบบจำลองแล้วทำการเปรียบเทียบประสิทธิภาพของแบบจำลองหลังจากนั้นจึงทำการประเมินผลแบบจำลองเพื่อหาแบบจำลองที่มีประสิทธิภาพที่สุด ซึ่งในแต่ละแบบจำลองก็จะมีความต่างในด้านของการป้อนพารามิเตอร์ต่างๆเข้าไป ซึ่งจะต้องมีหลักการในการใส่ค่าเหล่านั้น ซึ่งก่อนที่จะเริ่มทำการฝึกข้อมูลนั้นในแต่ละแบบจำลองได้มีการสร้าง data pipeline เพื่อป้องกันการรั่วไหลของชุดข้อมูลฝึกไปยังชุดข้อมูลทดสอบ

Support Vector Regression (SVR) สามารถกำหนดพารามิเตอร์ เช่น ค่า C ซึ่งเป็นค่าปรับสมดุลระหว่างความซับซ้อนของแบบจำลอง กับพารามิเตอร์ที่ป้องกันการ overfitting, epsilon (**ε**) ที่กำหนดช่วงของค่าที่สามารถมองข้าม error ได้, kernel ใช้ในการกำหนดรูปแบบของฟังก์ชัน เช่น linear, poly, rbf, และ gamma ซึ่งสามารถควบคุมความซับซ้อนของแบบจำลองได้ โค้ดสำหรับการสร้าง Support Vector Regression (SVR) แสดงดังภาพประกอบ 16

```
#1. SVR before Tuning

# Define the pipeline for X
pipeline = Pipeline([
    ('scaler_x', StandardScaler()),
    ('svr', SVR())
])

# Wrap the pipeline with TransformedTargetRegressor to scale y
model = TransformedTargetRegressor(
    regressor=pipeline,
    transformer=StandardScaler()
)

# Fit the model on the training data
model.fit(X_train, y_train)

# Predict on the test set
y_pred_svr = model.predict(X_test)
```

ภาพประกอบ 16 โค้ดสำหรับการสร้าง Support Vector Regression (SVR)

จากภาพประกอบ 16 แบบจำลอง SVR เริ่มจากกการปรับขนาดข้อมูล (scaling) ทั้งในส่วนของตัวแปรต้น (features) และตัวแปรตาม (target variable) โดยใช้เทคนิค StandardScaler ซึ่งเป็นการแปลงข้อมูลให้อยู่ในรูปแบบที่มีค่าเฉลี่ย (mean) เท่ากับศูนย์ และส่วนเบี่ยงเบนมาตรฐาน (standard deviation) เท่ากับหนึ่ง เพื่อให้ข้อมูลทุกตัวแปรอยู่ในมาตราส่วนเดียวกัน ซึ่งเป็นสิ่งสำคัญอย่างยิ่งต่อประสิทธิภาพของแบบจำลอง SVR เนื่องจาก SVR มีความไวต่อสเกลของข้อมูล โดยจะทำงานบนพื้นฐานของการคำนวณระยะห่างระหว่างจุดข้อมูล (distance-based methods) ดังนั้น หากตัวแปรใดมีค่ามากหรือน้อยเกินไปเมื่อเทียบกับตัวแปรอื่น จะส่งผลต่อการเรียนรู้ของแบบจำลองในลักษณะที่ไม่สมดุล ซึ่งอาจทำให้แบบจำลองเกิดความเอนเอียง (bias) ต่อบางตัวแปรโดยไม่ตั้งใจ รวมถึงอาจส่งผลต่อกระบวนการหาค่าที่ดีที่สุดของพารามิเตอร์ (optimization process) ให้ทำงานได้ช้าหรือไม่เสถียรได้ นอกจากนี้ยังได้มีการปรับขนาดของตัวแปรตามด้วย เนื่องจาก SVR ไม่สามารถจัดการกับ target variable ที่มีสเกลสูงได้ดี การปรับสเกลของ target จึงช่วยให้แบบจำลองสามารถเรียนรู้ความสัมพันธ์ในข้อมูลได้อย่างมีประสิทธิภาพมากขึ้น ก่อนที่จะทำการแปลงผลการพยากรณ์กลับมาเป็นค่าจริงในหน่วยเดิมผ่านกระบวนการ inverse transform เพื่อให้สามารถตีความผลลัพธ์ได้อย่างถูกต้อง หลังจากขั้นตอนการปรับสเกลแล้ว จึงมีการแบ่งข้อมูลเป็นชุดฝึก (training set) และชุดทดสอบ (test set) แล้วนำชุดข้อมูลฝึกที่ผ่านการปรับขนาดมาใช้ในการฝึกแบบจำลอง SVR และทำนายค่าผลลัพธ์จากข้อมูลทดสอบในรูปแบบที่ปรับขนาดแล้ว ก่อนจะนำค่าที่ได้มาแปลงกลับเพื่อให้ได้ค่าผลลัพธ์ในรูปแบบดั้งเดิม ซึ่งสามารถใช้เปรียบเทียบกับข้อมูลจริงได้อย่างตรงไปตรงมา

แบบจำลอง Lasso Regression จะมีการเพิ่มพจน์การลงโทษ (Penalty Term) จาก Linear Regression โดยการกำหนดค่าแอลฟา (alpha) ซึ่งสามารถทำการปรับเปลี่ยนค่านี้เพื่อดูผลของแบบจำลองได้ โค้ดสำหรับการสร้าง Lasso Regression แสดงดังภาพประกอบ 17

```
# 2. Lasso Regression

# Define the pipeline
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('lasso', Lasso(alpha=0.1))
])

# Fit the pipeline on the training data
pipeline.fit(X_train, y_train)

# Predict on the test set
y_pred_lasso = pipeline.predict(X_test)
```

ภาพประกอบ 17 โค้ดสำหรับการสร้าง Lasso Regression

จากภาพประกอบ 17 แบบจำลอง Lasso Regression เริ่มจากกการปรับขนาดข้อมูล (scaling) ด้วย StandardScaler เช่นเดียวกับแบบจำลอง SVR เนื่องจาก Lasso Regression มีความไวต่อขนาดของค่าตัวแปร หากข้อมูลไม่ได้รับการปรับสเกล ค่าความสำคัญของแต่ละตัวแปรที่แบบจำลองเรียนรู้อาจผิดเพี้ยน และอาจส่งผลต่อการเลือกตัวแปร (feature selection) ได้ หลังจากการ scaling แล้ว ข้อมูลถูกแบ่งเป็นชุดฝึก (training set) และชุดทดสอบ (test set) และใช้ข้อมูลฝึกที่ผ่านการปรับขนาดในการฝึกแบบจำลอง Lasso โดยมีการกำหนดค่า alpha = 0.1 ซึ่งเป็นค่าคงที่ที่ควบคุมความแรงของการทำ regularization ยิ่งค่า alpha สูง การลงโทษต่อค่าพารามิเตอร์ยิ่งมาก และแบบจำลองจะลดจำนวนตัวแปรที่ใช้ลง (ทำให้พารามิเตอร์บางตัวกลายเป็นศูนย์) ซึ่งช่วยในการป้องกันการฟิตมากเกินไป (overfitting) และช่วยในกระบวนการเลือกตัวแปรที่มีนัยสำคัญ จากนั้นจึงนำข้อมูลทดสอบที่ผ่านการ scaling แล้วไปใช้ในการพยากรณ์ผลลัพธ์โดยแบบจำลอง Lasso ที่ได้รับการฝึกเรียบร้อยแล้ว เพื่อประเมินความสามารถของแบบจำลองในการพยากรณ์ข้อมูลใหม่

แบบจำลอง Ridge Regression จะมีการสร้างแบบจำลองที่คล้ายกับแบบจำลอง Lasso โดยส่วนที่แตกต่างกันคือสมการ Regularization โค้ดสำหรับการสร้าง Ridge Regression แสดงดังภาพประกอบ 18

```
# 3. Ridge regression before Tuning

# Define the pipeline
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('ridge', Ridge(alpha=0.1))
])

# Fit the pipeline on the training data
pipeline.fit(X_train, y_train)

# Make predictions on the scaled test set
y_pred_ridge = pipeline.predict(X_test)
```

ภาพประกอบ 18 โค้ดสำหรับการสร้าง Ridge Regression

จากภาพประกอบ 18 แบบจำลอง Ridge Regression เริ่มจากกการปรับขนาดข้อมูล (scaling) ด้วย StandardScaler เช่นเดียวกับแบบจำลอง SVR และ Lasso Regression ด้วยเหตุผลเดียวกัน หลังจากการ scaling แล้ว ข้อมูลถูกแบ่งเป็นชุดฝึก (training set) และชุดทดสอบ (test set) และใช้ข้อมูลฝึกที่ผ่านการปรับขนาดในการฝึกแบบจำลอง Lasso โดยมีการกำหนดค่า

alpha = 0.1 ซึ่งเป็นค่าคงที่ จากนั้นจึงนำข้อมูลทดสอบที่ผ่านการ scaling แล้วไปใช้ในการพยากรณ์ผลลัพธ์โดยแบบจำลอง Ridge ที่ได้รับการฝึกเรียบร้อยแล้ว เพื่อประเมินความสามารถของแบบจำลองในการพยากรณ์ข้อมูลใหม่ ซึ่งข้อแตกต่างระหว่าง Ridge และ Lasso Regression คือ Ridge จะไม่ไปทำให้ค่าสัมประสิทธิ์บางตัวเป็นศูนย์ (Feature Selection) เหมือนกับ Lasso, Ridge จะลดขนาดของสัมประสิทธิ์ลง ทำให้สามารถป้องกัน overfitting ได้โดยยังคงใช้ข้อมูลทั้งหมดเหมือนเดิม

แบบจำลอง Decision Tree สามารถกำหนดค่าพารามิเตอร์ต่าง ๆ เช่น max_depth เพื่อควบคุมความลึกของต้นไม้, min_samples_split และ min_samples_leaf เพื่อกำหนดจำนวนข้อมูลขั้นต่ำที่ใช้ในการแบ่งและสร้างใบไม้ของต้นไม้ เป็นต้น และหากมีการใช้ Decision Tree สำหรับปัญหาแบบ Regression จะต้องใช้ DecisionTreeRegressor ในการสร้างแบบจำลอง โค้ดสำหรับการสร้าง Decision Tree แสดงดังภาพประกอบ 19

```
# 4.Decision Tree model

# Define the pipeline
pipeline = Pipeline([('dt', DecisionTreeRegressor(random_state=seed))])

# Fit the pipeline on the training data
pipeline.fit(X_train, y_train)

# Make predictions on the test set
y_pred_dt = pipeline.predict(X_test)
```

ภาพประกอบ 19 โค้ดสำหรับการสร้าง Decision Tree

แบบจำลอง Random Forest ซึ่งเป็นการรวมกันของหลาย ๆ Decision Tree สามารถกำหนดค่าพารามิเตอร์ เช่น n_estimators ซึ่งระบุจำนวนต้นไม้ที่ใช้, max_depth ที่กำหนดความลึกสูงสุดของแต่ละต้นไม้ และ bootstrap เพื่อกำหนดว่าจะใช้การสุ่มตัวอย่างซ้ำหรือไม่ ซึ่งช่วยลดปัญหา overfitting และเพิ่มความแม่นยำของแบบจำลองได้ และเช่นเดียวกัน หากมีการใช้ Random Forestสำหรับปัญหาแบบ Regression จะต้องใช้ RandomForestRegressor สำหรับการสร้างแบบจำลองโค้ดสำหรับการสร้าง Random Forest แสดงดังภาพประกอบ 20

```python
# 5. Random forest

# Define the pipeline
pipeline = Pipeline([('rf', RandomForestRegressor(random_state=seed))])

# Fit the pipeline on the training data
pipeline.fit(X_train, y_train)

# Make predictions on the test set
y_pred_rf = pipeline.predict(X_test)
```

ภาพประกอบ 20 โค้ดสำหรับการสร้าง Random Forest

แบบจำลอง XGBoost ที่เป็นแบบจำลองแบบเดียวที่ใช้หลักการแบบ Tree-Based Algorithm ซึ่งเป็นอัลกอริทึมที่ใช้ Gradient Boosting โดยแบบจำลอง XGBoost สามารถกำหนดค่าพารามิเตอร์ เช่น max_depth, learning_rate, และ n_estimators ได้ แบบจำลอง XGBoost ถ้าใช้กับปัญหาแบบ Regression จะต้องใช้ XGBRegressor สำหรับการสร้างแบบจำลอง โค้ดสำหรับการสร้าง XGBoost แสดงดังภาพประกอบ 21

```python
# 6. XGBoot

# Define the pipeline
pipeline = Pipeline([('xgb', XGBRegressor(random_state=seed))])

# Fit the pipeline on the training data
pipeline.fit(X_train, y_train)

# Make predictions on the test set
y_pred_xgb = pipeline.predict(X_test)
```

ภาพประกอบ 21 โค้ดสำหรับการสร้าง XGBoost

จะเห็นได้ว่าจากแบบจำลอง Decision Tree, RandomForest และ XGBoost จะไม่ต้องทำการปรับขนาดข้อมูลก่อนเหมือนกับแบบจำลอง SVR, Lasso Regression และ Ridge Regression เนื่องจากแบบจำลองดังกล่าวใช้การเปรียบเทียบค่าเป็นหลักในการตัดสินใจและสร้างกฎการแยกข้อมูลค่าของแต่ละตัวแปรทำให้สามารถจัดการกับข้อมูลดิบที่มีขนาดต่างกันได้อย่างมีประสิทธิภาพ โดยไม่จำเป็นต้อง preprocessing ข้อมูลในขั้นตอนการปรับขนาดก่อนใช้งาน และในแบบจำลอง Decision Tree, RandomForest และ XGBoost จะไม่ได้มีการกำหนดค่าพารามิเตอร์ใด ๆ เลยโดย

ใช้ค่าตั้งต้นจากตัวแบบจำลองเท่านั้น สรุปค่าของพารามิเตอร์ในแต่ละแบบจำลองก่อนการปรับจูนค่าไฮเปอร์พารามิเตอร์ ดังตาราง 3

ตาราง 3 ค่าไฮเปอร์พารามิเตอร์ที่ใช้ก่อนการปรับแบบจำลอง

แบบจำลอง	ค่าไฮเปอร์พารามิเตอร์
Support Vector Regression	Kernel [rbf]
Lasso Regression	Alpha [0.1]
Ridge Regression	Alpha [0.1]
Decision Tree	ไม่ได้ระบุใช้ค่าตั้งต้น
Random Forest	ไม่ได้ระบุใช้ค่าตั้งต้น
XGBoots	ไม่ได้ระบุใช้ค่าตั้งต้น

3.5 การปรับจูนค่าไฮเปอร์พารามิเตอร์ (Hyperparameter Tuning)

การปรับจูนค่าไฮเปอร์พารามิเตอร์ของแต่ละแบบจำลอง จะช่วยเพิ่มความแม่นยำและลดปัญหา Overfitting หรือ Underfitting โดยใช้กระบวนการทดลองอย่างเป็นระบบ (Grid Search หรือ Random Search) ควบคู่ไปกับการประเมินผลบนชุด Validation หรือ Cross-Validation เพื่อเลือกค่าที่ดีที่สุดสำหรับแต่ละพารามิเตอร์ โดยในแต่ละแบบจำลองก็จะมีการปรับค่าพารามิเตอร์ที่แตกต่างกันไปและในแต่ละบบจำลองจะมีการทำ data pipeline เช่นเดียวกับการทำแบบจำลองก่อนการปรับจูนค่าพารามิเตอร์

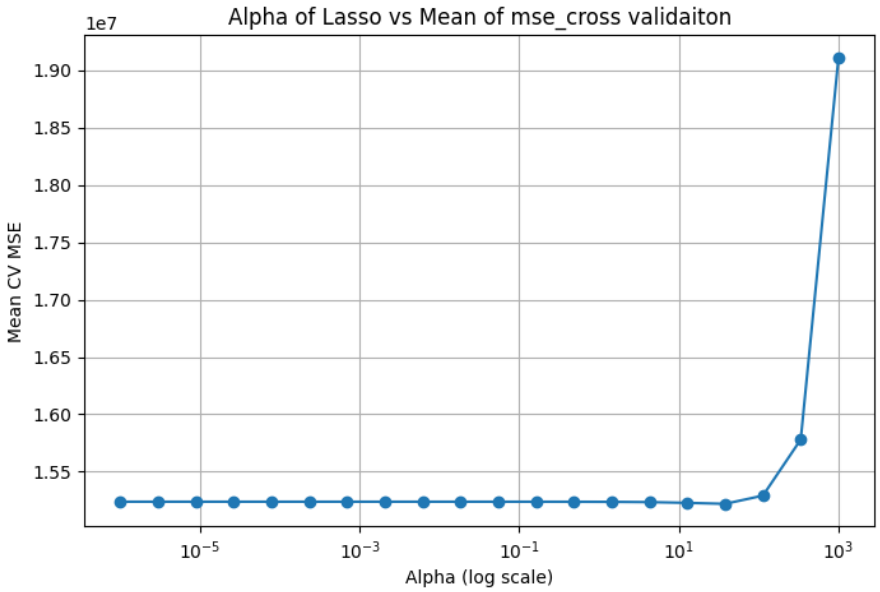
แบบจำลอง Support Vector Regression (SVR) เนื่องจาก SVR อ่อนไหวต่อช่วงค่าของข้อมูล โดยข้อมูลฝึกถูกใช้ในการ fit และ transform ในขณะที่ข้อมูลทดสอบถูก transform ตามค่าที่ได้จากข้อมูลฝึก จากนั้นแบบจำลอง SVR ถูกสร้างขึ้นโดยยังไม่มีการกำหนดค่าไฮเปอร์พารามิเตอร์และกำหนดช่วงค่าของพารามิเตอร์ที่ต้องการปรับแต่งผ่าน GridSearchCV ซึ่งครอบคลุม kernel function ได้แก่ linear, rbf และ poly รวมถึงค่าปรับสมดุลระหว่าง bias-variance tradeoff หรือค่า C ที่กำหนดเป็น 0.1, 1 และ 10 รวมถึงพารามิเตอร์ gamma ที่สามารถเป็น ‘scale’ หรือ ‘auto’ และค่า epsilon ที่ใช้กำหนดช่วงค่าที่ยอมให้ข้อผิดพลาดอยู่ภายใน ซึ่งถูกกำหนดเป็น 0.01, 0.1 และ 1 โดยหลักการเลือกค่าปรับจูนไฮเปอร์พารามิเตอร์ของ SVR มาจากการเลือกช่วงค่าแบบ log scale

หลักๆ 3 ค่า จะไม่ได้ทำการทดสอบค่า C หรือ ค่า epsilon หลังจากนั้น GridSearchCV จะทำการค้นหาชุดพารามิเตอร์ที่เหมาะสมที่สุดโดยใช้ 5-fold cross-validation และใช้ค่าความคลาดเคลื่อนเชิงกำลังสองเฉลี่ย (Mean Squared Error: MSE) เป็นตัววัดผล ซึ่งค่า MSE จะถูกแปลงเป็นค่าลบอัตโนมัติเนื่องจากเป็นค่าที่ใช้ minimization ในกระบวนการ tuning เมื่อ GridSearchCV ทำงานเสร็จสิ้นจะได้พารามิเตอร์ที่ดีที่สุดจากการทดลองค่าต่าง ๆ และพารามิเตอร์ที่ดีที่สุดนี้จะถูกนำมาใช้ในการสร้างแบบจำลอง SVR ใหม่ที่มีการใช้ค่าพารามิเตอร์ที่มีการปรับแต่งแล้ว โค้ดสำหรับการปรับจูนค่าไฮเปอร์พารามิเตอร์ของแบบจำลอง Support Vector Regression (SVR) แสดงดังภาพประกอบ 22

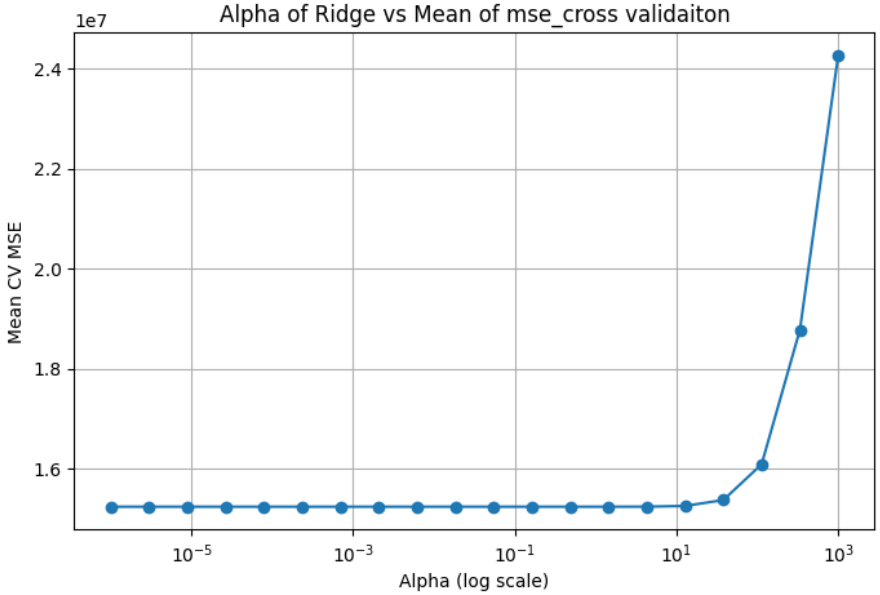
```python
#1. SVR with Tuning Parameter

# Define the pipeline for X
pipeline = Pipeline([
    ('scaler_x', StandardScaler()),
    ('svr', SVR())
])

# Wrap the pipeline with TransformedTargetRegressor to scale y
model = TransformedTargetRegressor(
    regressor=pipeline,
    transformer=StandardScaler()
)

# Define parameter grid with correct prefix for the inner SVR
param_grid = {
    'regressor__svr__kernel': ['linear', 'rbf', 'poly'],
    'regressor__svr__C': [0.1,1,10],
    'regressor__svr__gamma': ['scale', 'auto'],
    'regressor__svr__epsilon': [0.01, 0.1, 1]
}

# Create GridSearchCV object
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='neg_mean_squared_error')

# Fit the GridSearchCV object
grid_search.fit(X_train, y_train)

# Best hyperparameters
best_params = grid_search.best_params_
print("Best Hyperparameters:", best_params)

# Get the best model
best_model = grid_search.best_estimator_

# Cross-validation score using best model
cv_scores = cross_val_score(best_model, X_train, y_train, cv=5, scoring='neg_mean_squared_error')
print("Cross-validation scores (neg-MSE):", -cv_scores)
print("Mean cross-validation score (MSE):", -np.mean(cv_scores))

# Fit the best model on full training data
best_model.fit(X_train, y_train)

# Predict on the test set
y_pred_best_svr = best_model.predict(X_test)
```

ภาพประกอบ 22 โค้ดการปรับจูนค่าไฮเปอร์พารามิเตอร์ของแบบจำลอง SVR

แบบจำลอง Lasso Regression และ Ridge Regression สามารถปรับค่าพารามิเตอร์แอลฟา (alpha) โดยการที่จะกำหนดค่าเหล่านี้จะมาจากการทดลองดูค่าเฉลี่ยของค่า mse cross validation เทียบกับค่า แอลฟา ในช่วงตั้งแต่ 10^{-6} ถึง 10^{3} แสดงให้เห็นดังภาพประกอบ 23 และ 24



ภาพประกอบ 23 ค่าเฉลี่ยของค่า mse cross validation เทียบกับค่า แอลฟา ของ Lasso



ภาพประกอบ 24 ค่าเฉลี่ยของค่า mse cross validation เทียบกับค่า แอลฟา ของ Ridge

ซึ่งจากภาพทั้งสองจะเห็นได้ว่าช่วงตั้งแต่ 10^{-6} ถึง 10^{2} นั้นค่าเฉลี่ยไม่ได้มีการเปลี่ยนแปลงมากนั้น ดังนั้นสำหรับช่วงของค่า แอลฟา จะกำหนดตั้งแต่ 0.001 , 0.01, 0.1, 1, 10, 100 หลังจากกำหนดช่วงค่าของ alpha แล้ว จึงสร้างแบบจำลอง Lasso และ Ridge โดยการแบ่งข้อมูลเพื่อทำ cross-validation ด้วย KFold โดยกำหนดให้มีการแบ่งข้อมูลเป็น 5 ส่วน (5 folds) และทำการสลับข้อมูลแบบสุ่ม (shuffle) เพื่อให้มั่นใจได้ว่าแต่ละชุดข้อมูลมีการกระจายตัวอย่างสม่ำเสมอ ซึ่งเป็นการลดปัญหาอคติจากการแบ่งชุดข้อมูล จากนั้นใช้ GridSearchCV ซึ่งจะทำการฝึกแบบจำลอง ซ้ำหลายครั้งตามจำนวนพารามิเตอร์ในตารางที่กำหนด (param_grid) และประเมินผลแต่ละแบบจำลองด้วยค่าความคลาดเคลื่อนเฉลี่ยกำลังสอง (Mean Squared Error: MSE) ที่ถูกทำให้เป็นค่าลบเพื่อให้เข้ากับเกณฑ์การเพิ่มค่าคะแนนในกระบวนการค้นหา เมื่อการค้นหาเสร็จสิ้น จะได้ค่าของ alpha ที่เหมาะสมที่สุด (best alpha) ซึ่งนำมาใช้เพื่อสร้างแบบจำลองที่ให้ผลลัพธ์ดีที่สุด โดยไม่เกิด overfitting หรือ underfitting โค้ดสำหรับการปรับจูนค่าไฮเปอร์พารามิเตอร์ของแบบจำลอง Lasso Regression และ Ridge Regression แสดงดังภาพประกอบ 24 และ 25

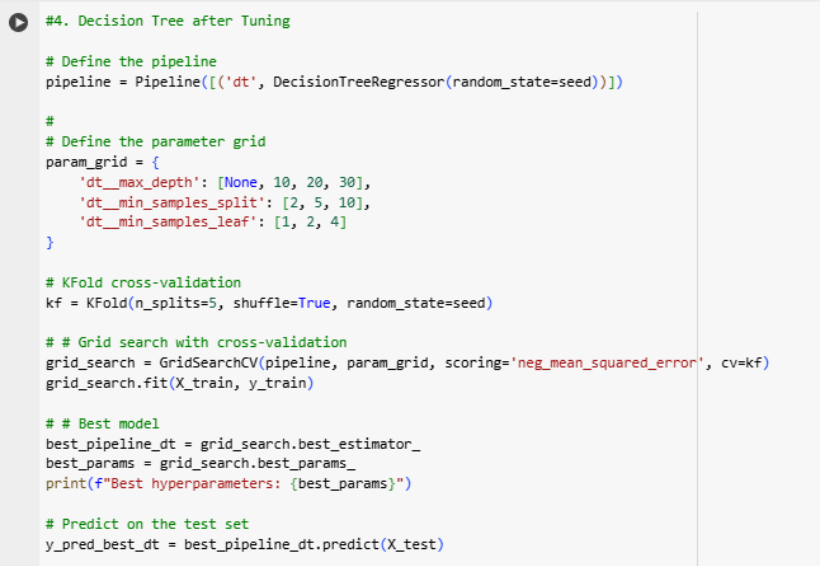
```python
#2. Lasso after tuning

# Define the pipeline
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('lasso', Lasso())
])

# Define the parameter grid (use 'lasso__alpha' to refer to the alpha parameter in the pipeline)
param_grid = {
    'lasso__alpha': [0.001, 0.01, 0.1, 1, 10, 100]
}

# Create a KFold cross-validator
kf = KFold(n_splits=5, shuffle=True, random_state=seed)

# Grid search with cross-validation
grid_search = GridSearchCV(pipeline, param_grid, scoring='neg_mean_squared_error', cv=kf)
grid_search.fit(X_train, y_train)

# Best model
best_pipeline_lasso = grid_search.best_estimator_
best_alpha = grid_search.best_params_['lasso__alpha']
print(f"Best alpha: {best_alpha}")

# Predict on the test set
y_pred_best_lasso = best_pipeline_lasso.predict(X_test)
```

ภาพประกอบ 25 โค้ดการปรับจูนค่าไฮเปอร์พารามิเตอร์ของแบบจำลอง Lasso

```python
#3. Ridge after tuning

# Define the pipeline
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('ridge', Ridge())
])

# Define the parameter grid (use 'lasso__alpha' to refer to the alpha parameter in the pipeline)
param_grid = {
    'ridge__alpha': [0.001, 0.01, 0.1, 1, 10, 100]
}

# Create a KFold cross-validator
kf = KFold(n_splits=5, shuffle=True, random_state=seed)

# Grid search with cross-validation
grid_search = GridSearchCV(pipeline, param_grid, scoring='neg_mean_squared_error', cv=kf)
grid_search.fit(X_train, y_train)

# Best model
best_pipeline_ridge = grid_search.best_estimator_
best_alpha = grid_search.best_params_['ridge__alpha']
print(f"Best alpha: {best_alpha}")

# Predict on the test set
y_pred_best_ridge = best_pipeline_ridge.predict(X_test)
```

ภาพประกอบ 26 โค้ดการปรับจูนค่าไฮเปอร์พารามิเตอร์ของแบบจำลอง Ridge

แบบจำลอง Decision Tree จะมีการปรับค่าพารามิเตอร์คือ ค่า max_depth ซึ่งเป็นค่าความลึกสูงสุดของต้นไม้ หากกำหนดค่านี้ต่ำเกินไปจะทำให้แบบจำลองเรียนรู้ได้ไม่เพียงพอ (underfitting) ในขณะที่ค่าที่สูงเกินไปอาจทำให้แบบจำลองซับซ้อนและเกิด overfitting ได้ ค่า min_samples_split กำหนดจำนวนตัวอย่างขั้นต่ำที่ต้องมีเพื่อให้ node หนึ่งสามารถถูกแยก (split) ได้ ซึ่งมีผลต่อความซับซ้อนของต้นไม้ และ min_samples_leaf คือจำนวนตัวอย่างขั้นต่ำที่ต้องมีในใบไม้แต่ละใบ (leaf node) เพื่อป้องกันไม่ให้แบบจำลองเรียนรู้จาก noise หรือข้อมูลที่ผิดปกติในตัวอย่างเพียงไม่กี่รายการ โดยการเลือกค่าพารามิเตอร์นั้นจะทำการเลือกจากค่าตั้งต้นของตัว libraries แล้วปรับให้ครอบคลุมโดยในแต่ละชนิดพารามิเตอร์จะใช้ทั้งหมดสามค่า จากนั้นจะสร้างแบบจำลอง Decision Tree Regressor โดยการแบ่งชุดข้อมูลฝึกผ่าน KFold cross-validation จำนวน 5 ส่วน ซึ่งจะทำการสลับลำดับข้อมูลแบบสุ่มก่อนแบ่ง เพื่อให้การประเมินผลของแบบจำลองมีความเสถียรและแม่นยำยิ่งขึ้น ขั้นตอนต่อไปคือทำ GridSearchCV ซึ่งจะทำการทดลองฝึกแบบจำลอง Decision Tree ด้วยทุกชุดค่าพารามิเตอร์ที่กำหนดไว้ใน param_grid และประเมินผลด้วยคะแนน ค่าความคลาดเคลื่อนเฉลี่ยกำลังสอง (Mean Squared Error - MSE) โดยใช้ค่าที่มีเครื่องหมายลบเพื่อให้สามารถเปรียบเทียบและค้นหาค่าที่ให้ผลดีที่สุดได้อย่างถูกต้อง ผลลัพธ์ของ Grid Search จะให้ค่าพารามิเตอร์ที่เหมาะสมที่สุด (best_params) และแบบจำองที่ได้จากการฝึกด้วยพารามิเตอร์ดังกล่าว (best_dt_model) โค้ดสำหรับการปรับจูนค่าไฮเปอร์พารามิเตอร์ของแบบจำลอง Decision Tree แสดงดังภาพประกอบ 27



```python
#4. Decision Tree after Tuning

# Define the pipeline
pipeline = Pipeline([('dt', DecisionTreeRegressor(random_state=seed))])

#
# Define the parameter grid
param_grid = {
    'dt__max_depth': [None, 10, 20, 30],
    'dt__min_samples_split': [2, 5, 10],
    'dt__min_samples_leaf': [1, 2, 4]
}

# KFold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=seed)

# # Grid search with cross-validation
grid_search = GridSearchCV(pipeline, param_grid, scoring='neg_mean_squared_error', cv=kf)
grid_search.fit(X_train, y_train)

# # Best model
best_pipeline_dt = grid_search.best_estimator_
best_params = grid_search.best_params_
print(f"Best hyperparameters: {best_params}")

# Predict on the test set
y_pred_best_dt = best_pipeline_dt.predict(X_test)
```

ภาพประกอบ 27 โค้ดการปรับจูนค่าไฮเปอร์พารามิเตอร์ของแบบจำลอง Decision Tree

แบบจำลอง Random Forest จะมีการปรับค่าพารามิเตอร์เหมือนกับแบบจำลอง Decision Tree แต่จะมีการเพิ่มเติมในส่วนของค่า n_estimators คือ จำนวนของต้นไม้ที่สามารถระบุได้ว่าต้องการให้มีต้นไม้กี่ต้นในแบบจำลอง เช่นเดียวกันกับแบบจำลองอื่น ๆ มีการแบ่งชุดข้อมูลฝึกผ่าน KFold cross-validation จำนวน 5 ส่วน และใช้ GridSearchCV ในการช่วยหาค่าพารามิเตอร์ที่ดีที่สุด โค้ดสำหรับการปรับจูนค่าไฮเปอร์พารามิเตอร์ของแบบจำลอง Decision Tree แสดงดังภาพประกอบ 28

```python
#5. Random forest after Tuning

# Define the pipeline
pipeline = Pipeline([('rf', RandomForestRegressor(random_state=seed))])

# Define the parameter grid for hyperparameter tuning
param_grid = {
    'rf__n_estimators': [50, 100, 200],
    'rf__max_depth': [None, 10, 20],
    'rf__min_samples_split': [2, 5, 10],
    'rf__min_samples_leaf': [1, 2, 4]
}

# Create a Random Forest Regression model
rf = RandomForestRegressor(random_state=seed)

# Create a KFold object for cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=seed)

# Perform GridSearchCV with k-fold cross-validation
grid_search = GridSearchCV(pipeline, param_grid, scoring='neg_mean_squared_error', cv=kf)
grid_search.fit(X_train, y_train)

# # Best model
best_pipeline_rf = grid_search.best_estimator_
best_params = grid_search.best_params_
print(f"Best hyperparameters: {best_params}")

# Predict on the test set
y_pred_best_rf = best_pipeline_rf.predict(X_test)
```

ภาพประกอบ 28 โค้ดการปรับจูนค่าไฮเปอร์พารามิเตอร์ของแบบจำลอง Random Forest

แบบจำลอง XGBoots สามารถปรับค่าพารามิเตอร์บางค่าได้เหมือนกับแบบจำลอง Decision Tree และ Random Forest ซึ่งในแบบจำลอง XGBoots ในงานวิจัยนี้จะมีการปรับค่าพารามิเตอร์ต่อไปนี้ จำนวนต้นไม้ (n_estimators), ความลึกของต้นไม้แต่ละต้น (max_depth), อัตราการเรียนรู้ (learning_rate), สัดส่วนข้อมูลที่ใช้ในการฝึกแต่ละต้นไม้ (subsample) และสัดส่วนของฟีเจอร์ที่ใช้ในการสร้างต้นไม้แต่ละต้น (colsample_bytree) จากนั้นได้สร้างแบบจำลองโดยใช้เป็น XGBoost Regressor และเช่นเดียวกันมีการใช้ KFold cross-validation จำนวน 5 ส่วน และใช้ GridSearchCV ในการช่วยหาค่าพารามิเตอร์ที่ดีที่สุด โค้ดสำหรับการปรับจูนค่าไฮเปอร์พารามิเตอร์ของแบบจำลอง XGBoots แสดงดังภาพประกอบ 29

```python
# 6. XGBoot after Tuning Parameter

# Define the pipeline
pipeline = Pipeline([('xgb', XGBRegressor(random_state=seed))])

# Define the parameter grid for hyperparameter tuning
param_grid = {
    'xgb__n_estimators': [100, 200, 300],
    'xgb__max_depth': [3, 5, 7],
    'xgb__learning_rate': [0.01, 0.1, 0.2],
    'xgb__subsample': [0.8, 0.9, 1.0],
    'xgb__colsample_bytree': [0.8, 0.9, 1.0]
}

# Create a KFold object for cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=seed)

# Perform GridSearchCV with k-fold cross-validation
grid_search = GridSearchCV(pipeline, param_grid, scoring='neg_mean_squared_error', cv=kf)
grid_search.fit(X_train, y_train)

# # Best model
best_pipeline_xgb = grid_search.best_estimator_
best_params = grid_search.best_params_
print(f"Best hyperparameters: {best_params}")

# Predict on the test set
y_pred_best_xgb = best_pipeline_xgb.predict(X_test)
```

ภาพประกอบ 29 โค้ดการปรับจูนค่าไฮเปอร์พารามิเตอร์ของแบบจำลอง XGBoots

โดยสรุปในแต่ละแบบจำลองก็จะมีการปรับค่าพารามิเตอร์ที่แตกต่างกันไปและใช้การแบ่งชุดข้อมูลด้วย KFold cross-validation จำนวน 5 ส่วน และใช้ GridSearchCV ในการหาค่าพารามิเตอร์ที่ดีที่สุดในแต่ละแบบจำลองซึ่งสามารถสรุปค่าพารามิเตอร์ได้ดังตาราง 4

ตาราง 4 ค่าไฮเปอร์พารามิเตอร์ที่ใช้ในการปรับแบบจำลอง

แบบจำลอง	ไฮเปอร์พารามิเตอร์
Support Vector Regression	Kernel [linear,rbf,poly], C [0.1, 1, 10] Gamma [scale, auto] Epsilon [0.01, 0.1, 1]
Lasso Regression	Alpha [0.001, 0.01, 0.1, 1, 10, 100]
Ridge Regression	Alpha [0.001, 0.01, 0.1, 1, 10, 100]
Decision Tree	max_depth [None, 10, 20, 30]

แบบจำลอง	ไฮเปอร์พารามิเตอร์
	min_samples_split [2, 5, 10]
	min_samples_leaf [1, 2, 4]
Random Forest	n_estimators [50, 100, 200]
	max_depth [None, 10, 20]
	min_samples_split [2, 5, 10]
	min_samples_leaf [1, 2, 4]
XGBoots	n_estimators [100, 200, 300]
	max_depth [3, 5, 7]
	learning_rate [0.01, 0.1, 0.2]
	subsample [0.8, 0.9, 1.0]
	colsample_bytree [0.8, 0.9, 1.0]

3.6 การประเมินผลลัพธ์ของแบบจำลอง (Model Evaluation)

การประเมินผลลัพธ์ของแต่ละแบบจำลองจะทำการประเมินทั้งก่อนการจูนไฮเปอร์พารามิเตอร์และหลังการจูนไฮเปอร์พารามิเตอร์ ผู้วิจัยได้ใช้วิธีในการประเมินแบบจำลองทั้งหมด 4 วิธีคือ 1) ค่าสัมประสิทธิ์การถดถอยกำลังสอง (R^2) 2) ค่าเฉลี่ยของความผิดพลาดแบบสัมบูรณ์ (MAE) 3) ค่าความคลาดเคลื่อนสัมบูรณ์เฉลี่ยร้อยละ (MAPE) และ 4) รากที่สองของค่าเฉลี่ยความผิดพลาดกำลังสอง (RMSE) ตัวอย่างโค้ดสำหรับการประเมินในแต่ละแบบจำลองแสดงดังภาพประกอบ 30

```
# Evaluate the model
mape = mean_absolute_percentage_error(y_test, y_pred_lasso)
rmse = np.sqrt(mean_squared_error(y_test, y_pred_lasso))
mae = mean_absolute_error(y_test, y_pred_lasso)
r2 = r2_score(y_test, y_pred_lasso)

# Print metrics
print("Lasso Regression Model")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Absolute Percentage Error (MAPE): {mape * 100:.2f}%")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2): {r2}")
```

ภาพประกอบ 30 ตัวอย่างโค้ดการประเมินผลลัพธ์ในแต่ละแบบจำลอง

นอกจากนี้แล้วเนื่องจากมีแบบจำลองหลายแบบจำลองและมีค่าการประเมินหลายค่า ผู้วิจัยได้ทำการสร้างโค้ดเพื่อหาแบบจำลองที่ดีที่สุดในแต่ละการประเมินเพื่อให้ง่ายต่อการวิเคราะห์ถัดไปมีทั้งในส่วนของก่อนการจูนไฮเปอร์พารามิเตอร์และหลังการจูนไฮเปอร์พารามิเตอร์โค้ดและผลัพธ์แสดงดังภาพประกอบ 31 และ 32

```python
# Create a dictionary to store the evaluation metrics for each model

model_results = {
    'SVR': {
        'MAE': mean_absolute_error(y_test, y_pred_svr),
        'MAPE': mean_absolute_percentage_error(y_test, y_pred_svr),
        'RMSE': np.sqrt(mean_squared_error(y_test, y_pred_svr)),
        'R2': r2_score(y_test, y_pred_svr)
    },
    'Lasso': {
        'MAE': mean_absolute_error(y_test, y_pred_lasso),
        'MAPE': mean_absolute_percentage_error(y_test, y_pred_lasso),
        'RMSE': np.sqrt(mean_squared_error(y_test, y_pred_lasso)),
        'R2': r2_score(y_test, y_pred_lasso)
    },
    'Ridge': {
        'MAE': mean_absolute_error(y_test, y_pred_ridge),
        'MAPE': mean_absolute_percentage_error(y_test, y_pred_ridge),
        'RMSE': np.sqrt(mean_squared_error(y_test, y_pred_ridge)),
        'R2': r2_score(y_test, y_pred_ridge)
    },
    'Decision Tree': {
        'MAE': mean_absolute_error(y_test, y_pred_dt),
        'MAPE': mean_absolute_percentage_error(y_test, y_pred_dt),
        'RMSE': np.sqrt(mean_squared_error(y_test, y_pred_dt)),
        'R2': r2_score(y_test, y_pred_dt)
    },
    'Random Forest': {
        'MAE': mean_absolute_error(y_test, y_pred_rf),
        'MAPE': mean_absolute_percentage_error(y_test, y_pred_rf),
        'RMSE': np.sqrt(mean_squared_error(y_test, y_pred_rf)),
        'R2': r2_score(y_test, y_pred_rf)
    },
    'XGBoost': {
        'MAE': mean_absolute_error(y_test, y_pred_xgb),
        'MAPE': mean_absolute_percentage_error(y_test, y_pred_xgb),
        'RMSE': np.sqrt(mean_squared_error(y_test, y_pred_xgb)),
        'R2': r2_score(y_test, y_pred_xgb)
    }
}
```

```python
# Find the best model for each metric
best_models = {}
for metric in ["MAE", "MAPE", "RMSE", "R2"]:
  if metric == "R2":
    best_model_name = max(model_results, key=lambda model: model_results[model][metric])
  else:
    best_model_name = min(model_results, key=lambda model: model_results[model][metric])
  best_models[metric] = best_model_name

# Print the best model for each metric
print("Best Models for Each Metric:")
for metric, model_name in best_models.items():
  print(f"{metric}: {model_name}")
```

```
Best Models for Each Metric:
MAE: Decision Tree
MAPE: Random Forest
RMSE: Random Forest
R2: Random Forest
```

ภาพประกอบ 31 โค้ดหาแบบจำลองที่ดีที่สุดก่อนการจูนไฮเปอร์พารามิเตอร์

```python
# Find the best model for each metric after tuning
best_tuned_models = {}
for metric in ["MAE", "MAPE", "RMSE", "R2"]:
    if metric == "R2":
        best_model_name = max(tuned_model_results, key=lambda model: tuned_model_results[model][metric])
    else:
        best_model_name = min(tuned_model_results, key=lambda model: tuned_model_results[model][metric])
    best_tuned_models[metric] = best_model_name

# Print the best model for each metric after tuning
print("\nBest Tuned Models for Each Metric:")
for metric, model_name in best_tuned_models.items():
    print(f"{metric}: {model_name}")
```

```
Best Tuned Models for Each Metric:
MAE: Decision Tree After Tuning
MAPE: Random Forest After Tuning
RMSE: Random Forest After Tuning
R2: Random Forest After Tuning
```

ภาพประกอบ 32 โค้ดหาแบบจำลองที่ดีที่สุดหลังการจูนไฮเปอร์พารามิเตอร์

ในบทนี้ได้แสดงให้เห็นถึงกระบวนการสร้างแบบจำลองของงานวิจัยอย่างเป็นลำดับ ตั้งแต่การเตรียมข้อมูล การวิเคราะห์ข้อมูลเบื้องต้น การสร้างแบบจำลอง การปรับจูนไฮเปอร์พารามิเตอร์ไปจนถึงการประเมินผล โดยในบทถัดไปจะนำเสนอผลลัพธ์ที่ได้จากการสร้างแบบจำลองทั้งก่อนการปรับจูนไฮเปอร์พารามิเตอร์และหลังการปรับจูนไฮเปอร์พารามิเตอร์ รวมถึงการเปรียบเทียบประสิทธิภาพของแบบจำลองที่ใช้ในการวิจัยครั้งนี้

บทที่ 4
ผลการศึกษา

ในบทนี้จะกล่าวถึงผลการศึกษาของแบบจำลองได้แก่ ผลลัพธ์ก่อนการปรับจูนไฮเปอร์พารามิเตอร์ ผลลัพธ์หลังการปรับจูนไฮเปอร์พารามิเตอร์ รวมถึง ไฮเปอร์พารามิเตอร์ที่ดีสุดในแต่ละแบบจำลอง เปอร์เซ็นต์การปรับปรุงพารามิเตอร์ของก่อนและหลัง โดยผลการศึกษาที่ได้นั้นจะการประเมินผลผ่านตัวประเมินทั้ง 4 แบบ ได้แก่ R^2, MAE, MAPE และ RMSE โดยศึกษาและวิเคราะห์ผลว่าแบบจำลองใดที่ให้ผลลัพธ์ที่ดีที่สุด

4.1 ผลลัพธ์ก่อนการปรับจูนไฮเปอร์พารามิเตอร์

ก่อนที่จะทำการปรับปรุงพารามิเตอร์จะมาดูในส่วนของผลลัพธ์ก่อนการปรับค่าพารามิเตอร์ต่าง ๆ ในแต่ละแบบจำลองโดยจะใช้ค่าประเมินทั้ง 4 แบบในการดูผลลัพธ์ แสดงดังตาราง 5

ตาราง 5 ผลลัพธ์ของแต่ละแบบจำลองก่อนการปรับจูนไฮเปอร์พารามิเตอร์

แบบจำลอง	R^2	MAE	MAPE	RMSE
Support Vector Regression	0.71	2,448.53	10.20%	3,564.11
Lasso Regression	0.70	2,629.23	11.05%	3,584.07
Ridge Regression	0.70	2,629.34	11.05%	3,584.16
Decision Tree	0.57	1,172.30	5.39%	4,344.54
Random Forest	0.82	1,210.14	5.22%	2,836.40
XGBoots	0.76	1,449.12	6.19%	3,239.96

จากตาราง 5 แสดงให้เห็นว่าแบบจำลองที่ดีที่สุดสำหรับแต่ละตัวชี้วัดแตกต่างกันไป โดยเมื่อพิจารณาค่าความผิดพลาดแบบสัมบูรณ์หรือ Mean Absolute Error (MAE) ซึ่งเป็นค่าที่บอกถึงความผิดพลาดเฉลี่ยระหว่างค่าทำนายกับค่าจริงในเชิงสัมบูรณ์ พบว่าแบบจำลอง Decision Tree ให้ค่าที่ต่ำที่สุด แสดงว่าแบบจำลองนี้สามารถทำนายค่าที่ใกล้เคียงกับค่าจริงได้ดีในแง่ของความ

ผิดพลาดเฉลี่ย อย่างไรก็ตามเมื่อพิจารณาตัวชี้วัดที่สำคัญอื่น ๆ เช่น Mean Absolute Percentage Error (MAPE) และ Root Mean Squared Error (RMSE) ซึ่งเป็นค่าที่ใช้วัดความผิดพลาดโดยให้ความสำคัญกับค่าผิดพลาดที่สูงมากกว่าค่าที่ต่ำ Random Forest กลับเป็นแบบจำลองที่ให้ค่าความผิดพลาดต่ำที่สุดในสองตัวชี้วัดนี้ ซึ่งหมายความว่าแบบจำลอง Random Forest สามารถลดข้อผิดพลาดในการทำนายได้ดีแม้ว่าจะมีค่าผิดปกติหรือข้อมูลที่มีการกระจายตัวสูง และสุดท้ายเมื่อพิจารณาค่า R-squared (R^2) ซึ่งใช้วัดว่าแบบจำลองสามารถอธิบายความแปรปรวนของข้อมูลได้ดีเพียงใด Random Forest ก็เป็นแบบจำลองที่มีค่ามากที่สุดอีกเช่นกัน ซึ่งหมายความว่าแบบจำลองนี้สามารถจับความสัมพันธ์ระหว่างพารามิเตอร์อิสระและพารามิเตอร์ตามได้ดีกว่าแบบจำลองอื่น ๆ ดังนั้นโดยสรุปแม้ว่า Decision Tree จะให้ค่าความผิดพลาดสัมบูรณ์ที่ต่ำที่สุด แต่ในภาพรวม Random Forest เป็นแบบจำลองที่ดีที่สุดเพราะสามารถลดข้อผิดพลาดได้ดีที่สุดทั้งในแง่ของ MAPE และ RMSE รวมถึงสามารถอธิบายข้อมูลได้ดีที่สุดในแง่ของค่า R^2 ซึ่งเป็นผลมาจากการใช้หลาย Decision Trees มาช่วยกันทำนายทำให้สามารถลดปัญหา Overfitting และเพิ่มความแม่นยำของแบบจำลองได้ดีกว่าแบบจำลองเดี่ยวอื่น ๆ ซึ่งจากสมมติฐานที่ตั้งไว้ว่าแบบจำลองแบบ Regression น่าจะให้ผลลัพธ์ที่ดีกว่าแบบจำลองแบบ Tree จากผลลัพธ์นี้แสดงให้เห็นว่ายังไม่สอดคล้องกับสมมติฐานที่ตั้งไว้ ดังนั้นถัดไปจะไปดูผลลัพธ์กันในส่วนเมื่อทำการปรับปรุงพารามิเตอร์แล้วจะให้ผลลัพธ์ที่แตกต่างกันไปหรือไม่

4.2 ผลลัพธ์หลังการปรับจูนไฮเปอร์พารามิเตอร์

โดยหลังจากการปรับพารามิเตอร์ต่าง ๆ ในแต่ละแบบจำลองโดยใช้ GridSearchCV ในการหาค่าพารามิเตอร์ที่ดีที่สุดในแต่ละแบบจำลองมาซึ่งแสดงผลของค่าพารามิเตอร์ที่ดีที่สุดในแต่ละแบบจำลองดังตาราง 6

ตาราง 6 ค่าไฮเปอร์พารามิเตอร์ที่ดีที่สุดในแต่ละแบบจำลอง

แบบจำลอง	ไฮเปอร์พารามิเตอร์ที่ดีที่สุด
Support Vector Regression	Kernel [rbf], C [1] Gamma [scale] Epsilon [0.1]
Lasso Regression	Alpha [10]

แบบจำลอง	ไฮเปอร์พารามิเตอร์ที่ดีที่สุด
Ridge Regression	Alpha [1]
Decision Tree	max_depth [None] min_samples_split [10] min_samples_leaf [4]
Random Forest	n_estimators [100] max_depth [None] min_samples_split [10] min_samples_leaf [2]
XGBoots	n_estimators [100] max_depth [5] learning_rate [0.1] subsample [0.9] colsample_bytree [0.8]

ซึ่งค่าตัวแปรที่ดีที่สุดในแต่ละแบบจำลองนั้นก็จะนำมาใช้ในการทดสอบข้อมูลอีกครั้งซึ่งผลลัพธ์ที่ได้หลังจากการปรับปรุงตัวแปรแสดงดังตาราง 7

ตาราง 7 ผลลัพธ์หลังจากการจูนไฮเปอร์พารามิเตอร์

แบบจำลอง	R^2	MAE	MAPE	RMSE
Support Vector Regression	0.71	2,448.53	10.20%	3,564.12
Lasso Regression	0.70	2,633.99	11.05%	3,591.09
Ridge Regression	0.70	2,629.34	11.05%	3,584.16
Decision Tree	0.84	1,107.30	4.73%	2,632.10
Random Forest	0.87	1,107.89	4.57%	2,416.46
XGBoots	0.82	1,459.35	6.00%	2,797.95

จากผลการเปรียบเทียบแบบจำลองก่อนและหลังการจูนไฮเปอร์พารามิเตอร์พบว่า การจูนไฮเปอร์พารามิเตอร์ ส่งผลอย่างมีนัยสำคัญต่อประสิทธิภาพของแบบจำลอง โดยเฉพาะในกรณีของแบบจำลอง Decision Tree, Random Forest และ XGBoost ซึ่งสามารถลดค่าความผิดพลาดลงได้อย่างมากเมื่อเทียบกับก่อนการจูนไฮเปอร์พารามิเตอร์ ในขณะที่แบบจำลองประเภท SVR, Lasso และ Ridge การจูนไฮเปอร์พารามิเตอร์นั้นไม่ได้ส่งผลต่อผลลัพธ์มากนัก หรือแทบไม่มีการเปลี่ยนแปลง ซึ่งอาจมีสาเหตุมาจากลักษณะของข้อมูลที่ไม่ซับซ้อนเพียงพอ ขาดความสัมพันธ์แบบไม่เชิงเส้น หรือข้อมูลไม่มีปัญหา multicollinearity ที่ชัดเจน อีกทั้งแบบจำลองเหล่านี้ล้วนมีพื้นฐานเป็นแบบจำลองเชิงเส้นหรือกึ่งเชิงเส้น จึงมีข้อจำกัดในการอธิบายความแปรปรวนของข้อมูลที่มีลักษณะซับซ้อนมากขึ้น

ก่อนการจูนไฮเปอร์พารามิเตอร์ (Before Tuning) Random Forest มีค่าความผิดพลาดต่ำที่สุดทั้งในแง่ของ MAPE (5.22%) และ RMSE (2,836.40) รวมถึงให้ค่า R^2 สูงสุดที่ 0.82 ซึ่งบ่งบอกว่ามีความสามารถในการอธิบายความแปรปรวนของข้อมูลได้ดีที่สุดในกลุ่มแบบจำลอง ก่อนการจูนไฮเปอร์พารามิเตอร์ Decision Tree มีค่า MAE ต่ำที่สุด (1,172.30) แต่ยังคงมีค่า MAPE และ RMSE สูงกว่าที่ควร แสดงให้เห็นว่าแม้จะมีค่าความผิดพลาดเฉลี่ยต่อจุดต่ำ แต่ยังมีข้อผิดพลาดสะสมที่สูง SVR, Lasso และ Ridge มีค่าความผิดพลาดที่ค่อนข้างสูง โดยเฉพาะในกรณีของ Lasso และ Ridge ที่มีค่าความผิดพลาดแทบไม่แตกต่างกันเลย ซึ่งอาจเป็นผลจากความคล้ายคลึงกันของแบบจำลองเชิงโครงสร้าง XGBoost มีค่าความผิดพลาดอยู่ในระดับปานกลาง โดยดีกว่า Lasso และ Ridge แต่ยังด้อยกว่า Random Forest

หลังการปรับจูนไฮเปอร์พารามิเตอร์ (After Tuning) Decision Tree ให้ผลลัพธ์ที่ดีขึ้น โดยที่ค่า MAPE ลดลงจาก 5.39% เหลือ 4.73% และค่า RMSE ลดลงจาก 4,344.54 เหลือเพียง 2,632.1 ส่งผลให้ R^2 เพิ่มขึ้นจาก 0.57 เป็น 0.84 ซึ่งหมายความว่าแบบจำลองสามารถอธิบายข้อมูลได้ดีขึ้น หลังการปรับจูนไฮเปอร์พารามิเตอร์ Random Forest กลายเป็นแบบจำลองที่ดีที่สุด โดยมีค่าความผิดพลาดต่ำที่สุดในทุกด้าน ได้แก่ MAPE (4.57%), RMSE (2,416.46) และ R^2 (0.87) ซึ่งหมายความว่าหลังการปรับแต่ง Random Forest สามารถลดข้อผิดพลาดลงได้มากที่สุด และอธิบายข้อมูลได้ดีที่สุด XGBoost มีประสิทธิภาพดีขึ้น โดยค่า MAPE และ RMSE ลดลง รวมถึงค่า R^2 เพิ่มขึ้นจาก 0.76 เป็น 0.82 แต่ในขณะเดียวกันแบบจำลอง SVR, Lasso และ Ridge ยังคงมีค่าความผิดพลาดใกล้เคียงเดิม และจากการปรับตัวแปรก็ไม่ได้ส่งผลต่อการเปลี่ยนแปลงผลลัพธ์แต่อย่างใด ซึ่งอาจแสดงให้เห็นว่าแบบจำลองเหล่านี้ไม่เหมาะกับข้อมูลชุดนี้ โดยที่ ตาราง 8 จะแสดง

ให้เห็นถึงเปอร์เซ็นต์การปรับปรุงพารามิเตอร์ในแต่ละแบบจำลองผ่านค่า R^2 เพื่อที่จะเห็นถึงความเปลี่ยนแปลงผลลัพธ์ของแต่ละแบบจำลองหลังการจูนไฮเปอร์พารามิเตอร์

ตาราง 8 เปอร์เซ็นต์การปรับปรุงพารามิเตอร์

แบบจำลอง	R^2 ก่อนการจูนไฮเปอร์พารามิเตอร์	R^2 หลังการจูนไฮเปอร์พารามิเตอร์	เปอร์เซ็นต์การปรับปรุง
Support Vector Regression	0.71	0.71	0.00
Lasso Regression	0.70	0.70	0.00
Ridge Regression	0.70	0.70	0.00
Decision Tree	0.57	0.84	47.85
Random Forest	0.82	0.87	6.16
XGBoots	0.76	0.82	8.00

จากตาราง 8 จะเห็นได้ว่า แบบจำลอง SVR Lasso และ Ridge Regression มีค่า R^2 ก่อนและหลังปรับแต่งเท่ากัน (0.71, 0.70 และ 0.70 ตามลำดับ) ซึ่งสาเหตุที่เป็นไปได้แบบจำลองอาจเข้าสู่จุดอิ่มตัว หรือข้อมูลไม่เหมาะกับการเรียนรู้เพิ่มเติมจากพารามิเตอร์ที่ปรับ เช่น มีความเป็นเชิงเส้นสูง หรือขาดความซับซ้อน ในขณะเดียวกันแบบจำลอง Decision Tree มีเปอร์เซ็นต์การปรับปรุงที่สูงที่สุดแสดงให้เห็นว่าการจูนไฮเปอร์พารามิเตอร์ส่งผลต่อแบบจำลอง Decision Tree อย่างมาก ซึ่งการเลือกปรับค่าที่เหมาะสมจะช่วยควบคุม overfitting ได้ดี แบบจำลอง Random Forest และ XGBoots ก็แสดงให้เห็นถึงการเปลี่ยนแปลงที่ดีขึ้นเมื่อทำการปรับจูนค่าไฮเปอร์พารามิเตอร์ สรุปจากผลลัพธ์แสดงให้เห็นว่าแบบจำลองในกลุ่ม Decision Tree, Random Forest และ XGBoost มีแนวโน้มได้รับประโยชน์อย่างมีนัยสำคัญจากการปรับจูนไฮเปอร์พารามิเตอร์ โดยสามารถเพิ่มค่าความแม่นยำของแบบจำลองได้อย่างชัดเจน สะท้อนถึงศักยภาพในการเรียนรู้รูปแบบความสัมพันธ์ที่มีความซับซ้อนในข้อมูลได้ดีกว่า เมื่อเปรียบเทียบกับแบบจำลองเชิงเส้นหรือแบบจำลองที่มีการกำกับด้วย regularization อย่าง Lasso, Ridge และ SVR ซึ่งพบว่าไม่มีการเปลี่ยนแปลงของประสิทธิภาพอย่างมีนัยสำคัญหลังการจูนพารามิเตอร์ ทั้งนี้อาจเนื่องมาจากข้อจำกัดของแบบจำลอง

ในการรองรับความซับซ้อนของข้อมูล หรือการที่ลักษณะข้อมูลไม่สอดคล้องกับสมมติฐานพื้นฐานของแบบจำลองเชิงเส้น

ในบทนี้ผู้วิจัยได้แสดงผลลัพธ์ทั้งก่อนการปรับจูนไฮเปอร์พารามิเตอร์และหลังการปรับจูนไฮเปอร์พารามิเตอร์ รวมถึงแสดงเปอร์เซ็นต์การปรับปรุงพารามิเตอร์ในแต่ละแบบจำลอง โดยในบทสุดท้ายผู้วิจัยจะทำการสรุปผล อภิปรายผล และข้อเสนอแนะสำหรับงานวิจัยนี้

บทที่ 5
สรุปผล อภิปรายผล และข้อเสนอแนะ

ในบทนี้จะกล่าวถึงการสรุปผลการวิจัยโดยอ้างอิงจากผลลัพธ์ที่ได้นำเสนอไว้ในบทที่ 4 เพื่อวิเคราะห์และประเมินว่าแบบจำลองที่พัฒนาขึ้นสามารถตอบโจทย์วัตถุประสงค์ของการวิจัยได้มากน้อยเพียงใด รวมถึงการอภิปรายผลให้สอดคล้องกับข้อมูลและผลลัพธ์ พร้อมทั้งนำเสนอข้อเสนอแนะทั้งในด้านการปรับปรุงงานวิจัย และแนวทางการต่อยอดเพื่อพัฒนางานวิจัยในอนาคตให้มีประสิทธิภาพและครอบคลุมยิ่งขึ้น

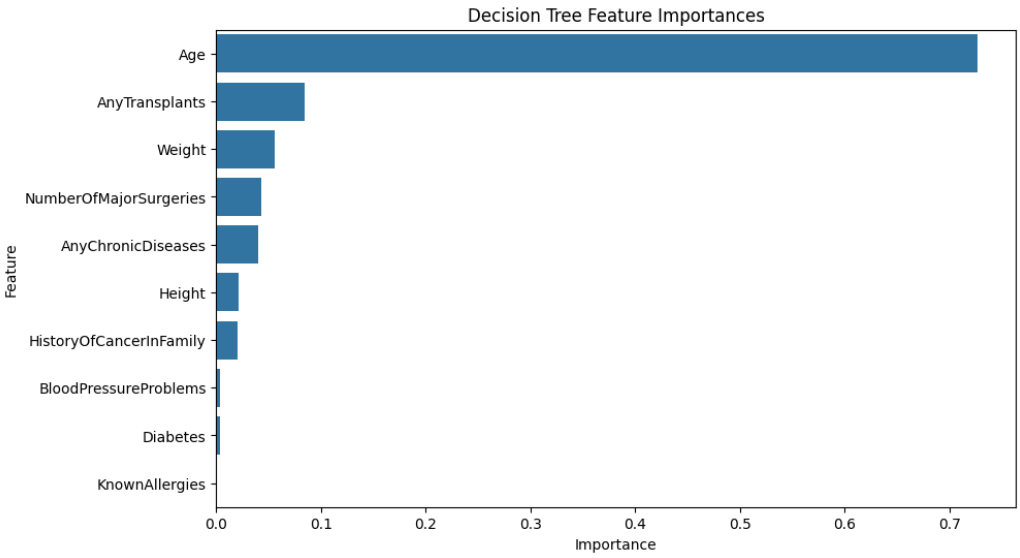
5.1 สรุปผล

จากผลลัพธ์หลังจากการปรับตัวแปร วิเคราะห์ผลแล้วพบว่า แบบจำลองที่ดีที่สุดในแต่ละตัวชี้วัดมีดังนี้ แบบจำลองที่ให้ค่า MAE ต่ำที่สุดคือ Decision Tree มีค่าอยู่ที่ 1,107.30 หมายความว่าโดยเฉลี่ยแล้ว การพยากรณ์ของ Decision Tree หลังการปรับค่าตัวแปรมีค่าความผิดพลาดต่ำที่สุดต่อข้อมูลแต่ละจุด ซึ่งอาจเป็นเพราะการแบ่งโครงสร้างของต้นไม้อย่างเหมาะสมทำให้ลดข้อผิดพลาดเฉลี่ยได้ดี แบบจำลองที่ให้ค่า MAPE ต่ำที่สุดคือ Random Forest โดยมีค่าอยู่ที่ 4.57% ซึ่งบ่งบอกว่าแบบจำลองนี้สามารถทำนายค่าที่ใกล้เคียงกับค่าจริงได้ดีในเชิงสัดส่วนเมื่อเปรียบเทียบกับแบบจำลองอื่น ๆ โดยรวมแล้ว Random Forest จึงมีความแม่นยำสูงที่สุดในเชิงของข้อผิดพลาดเชิงเปอร์เซ็นต์ แบบจำลองที่ให้ค่า RMSE ต่ำที่สุด คือ Random Forest มีค่าอยู่ที่ 2,416.46 แสดงว่าแบบจำลองนี้สามารถลดความผิดพลาดโดยรวมลงได้ดีที่สุด และแบบจำลองที่ให้ค่า R^2 สูงที่สุด คือ Random Forest มีค่าอยู่ที่ 0.87 แสดงให้เห็นว่าแบบจำลองสามารถอธิบายความแปรปรวนของข้อมูลได้ดีที่สุด จากตัวชี้วัดทั้ง 4 พบว่าแบบจำลองแบบ Random Forest ให้ผลลัพธ์ที่ดีถึง 3 ใน 4 ของตัวชี้วัดทั้งหมดซึ่งอาจสรุปได้ว่าสำหรับการทำนายเบี้ยประกันสุขภาพแบบจำลอง Random Forest นั้นมีความเหมาะสมที่สุด และจากวิจัยของ Ugochukwu Orji และคณะ[7] ที่นำมาใช้เป็นงานวิจัยอ้างอิงหลักของเราพบว่าแบบจำลอง Random Forest นั้นให้ค่า MAPE อยู่ที่ 5.8% ซึ่งค่าจากงานวิจัยของเราอยู่ที่ 4.57% แสดงว่าผลลัพธ์ของงานวิจัยของเรานั้นให้ผลลัพธ์ที่ดีในระดับหนึ่งเมื่อทำการเทียบเคียงกับงานวิจัยอื่นแต่อย่างไรก็ตามการที่จะบอกได้ว่าผลลัพธ์ของงานวิจัยของเรานั้นดีหรือไม่นั้นอาจจะต้องดูในเรื่องที่ว่าธุรกิจประเภทนี้นั้นมีค่ายอมรับเปอร์เซ็นต์ความผิดพลาดอยู่ที่เท่าไรจึงจะสามารถอ้างอิงได้ว่างานวิจัยของเรานั้นสามารถให้ผลลัพธ์ที่อยู่ในเกณฑ์ได้หรือไม่ และจากสมมติฐานตอนต้นที่ตั้งไว้ว่าแบบจำลองแบบ Regression จะให้ผลลัพธ์ที่

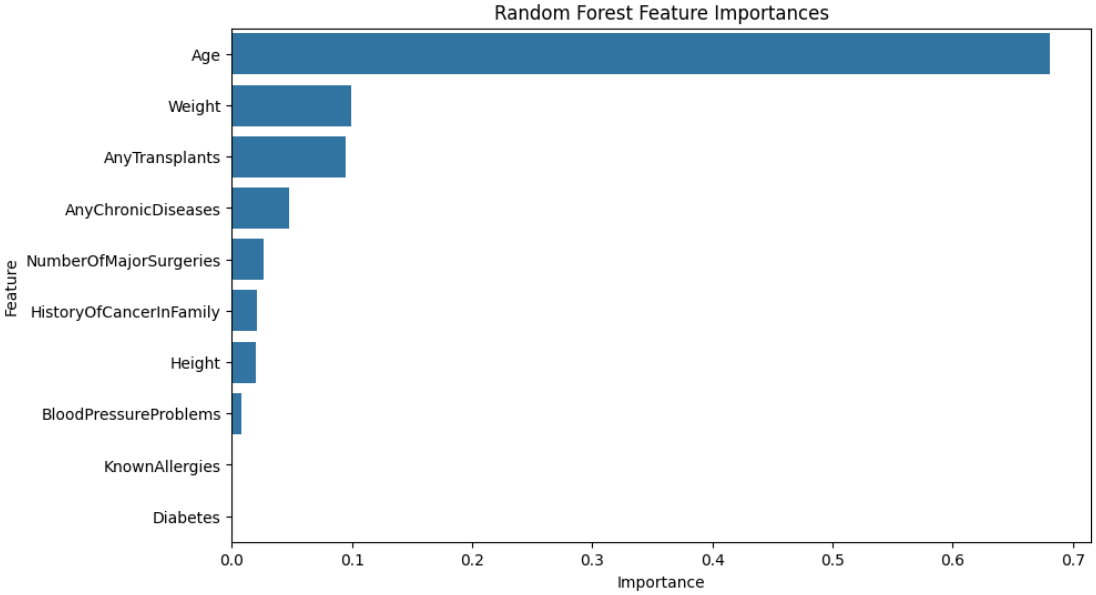
ดีกว่านั้นไม่เป็นไปตามที่ตั้งไว้ซึ่งสาเหตุอาจจะมาจากการที่แบบจำลองแบบ Tree-based สามารถจัดการปัญหาที่ซับซ้อนได้ดีกว่าแบบ Regression ซึ่งจากผลลัพธ์ได้แสดงให้เห็นว่า Random Forest ที่อยู่ในกลุ่มแบบจำลองแบบ Tree-based นั้นให้ผลลัพธ์ที่ดีที่สุดเนื่องมาจาก Random Forest เป็นแบบจำลองที่อยู่บนพื้นฐานของการรวมผลลัพธ์จากหลาย ๆ ต้นไม้ (Decision Trees) ผ่านเทคนิค Bagging ซึ่งมีความสามารถในการจับความสัมพันธ์ที่ไม่เป็นเชิงเส้นระหว่างตัวแปรต้นกับตัวแปรตามได้อย่างยืดหยุ่นและไม่จำเป็นต้องตั้งสมมติฐานเชิงเส้นเหมือนกับแบบ Regression ที่มีข้อจำกัดในการอธิบายเฉพาะความสัมพันธ์ที่เป็นเชิงเส้น (Linear Relationship) เท่านั้น โดยเฉพาะในปัญหาที่มีลักษณะข้อมูลซับซ้อน ซึ่งจากชุดข้อมูลที่ใช้ในงานวิจัยนี้จะเห็นได้ว่าการทำนายเบี้ยประกันสุขภาพนั้นมีการใช้ตัวแปรหลายตัวแปรมาใช้ในการทำนาย ซึ่งตัวแปรเหล่านี้มักมีลักษณะการปฏิสัมพันธ์ (Interaction) กันและกัน เช่น บุคคลคนหนึ่งอาจจะมีประวัติมีปัญหาทั้งในด้านเบาหวาน ความดัน อยู่ในคนคนเดียวกันซึ่งจะทำให้มีความเสี่ยงสูงขึ้นแบบทวีคูณ ไม่ใช่แค่เพิ่มขึ้นตามแนวเส้นตรง ทำให้แบบจำลองเชิงเส้นอย่าง Linear Regression ไม่สามารถจับรูปแบบเหล่านี้ได้ดีเท่าที่ควร ในขณะที่ Random Forest สามารถเรียนรู้กฎเชิงตรรกะซ้อนกันแบบลำดับชั้น (Hierarchical Rules) เพื่อแบ่งกลุ่มข้อมูลออกตามเงื่อนไขของตัวแปรต้นในหลายระดับจึงทำให้การทำนายนั้นครอบคลุมมากกว่าแบบที่ใช้ Linear Regression ดังนั้นการเลือกแบบจำลองให้สอดคล้องกับข้อมูลของและการปรับจูนไฮเปอร์พารามิเตอร์ให้เหมาะสมก็จะช่วยให้ได้ผลลัพธ์ที่ดีขึ้นตามไปด้วย

5.2 อภิปรายผล

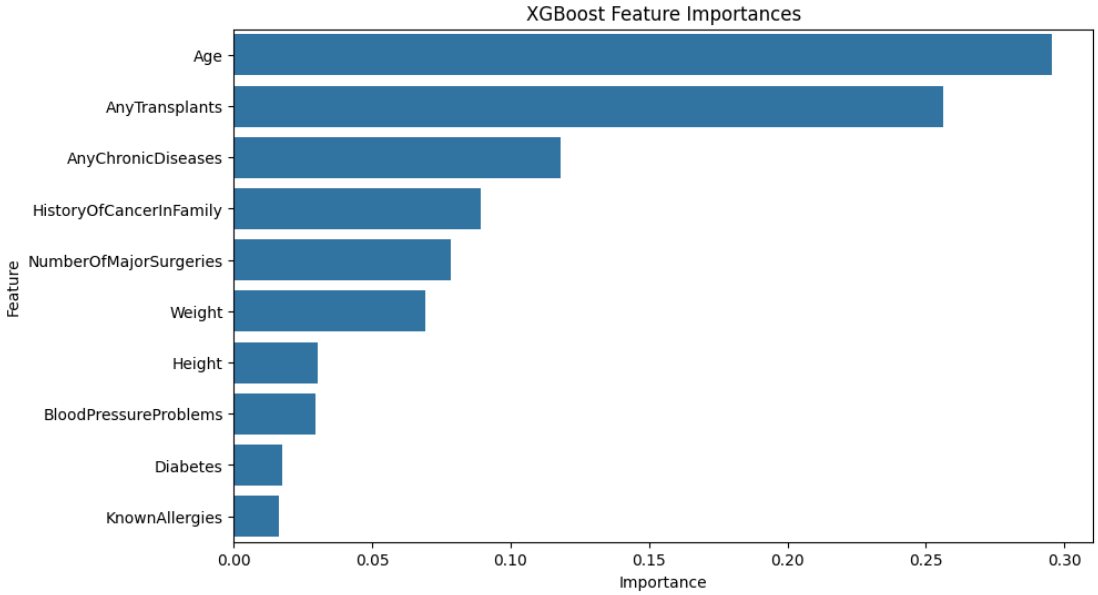
จากผลสรุป แบบจำลอง Random Forest นั้นมีความเหมาะสมที่สุดสำหรับชุดข้อมูลนี้และการทำนายผลเบี้ยประกันสุขภาพซึ่งแบบจำลองในกลุ่มต้นไม้นั้นสามารถให้ผลลัพธ์ที่ดีขึ้นหลังจากการปรับค่าตัวแปรซึ่งทั้งแบบจำลอง Decision Tree, Random Forest และ XGBoots นั้น สามารถดูในเรื่องของ Feature Importance ว่าสอดคล้องกับสมมติฐานแรกที่ตั้งไว้หรือไหมซึ่งจะแสดงดังภาพประกอบ 33, 34 และ 35



ภาพประกอบ 33 Feature Importances ของแบบจำลอง Decision Tree

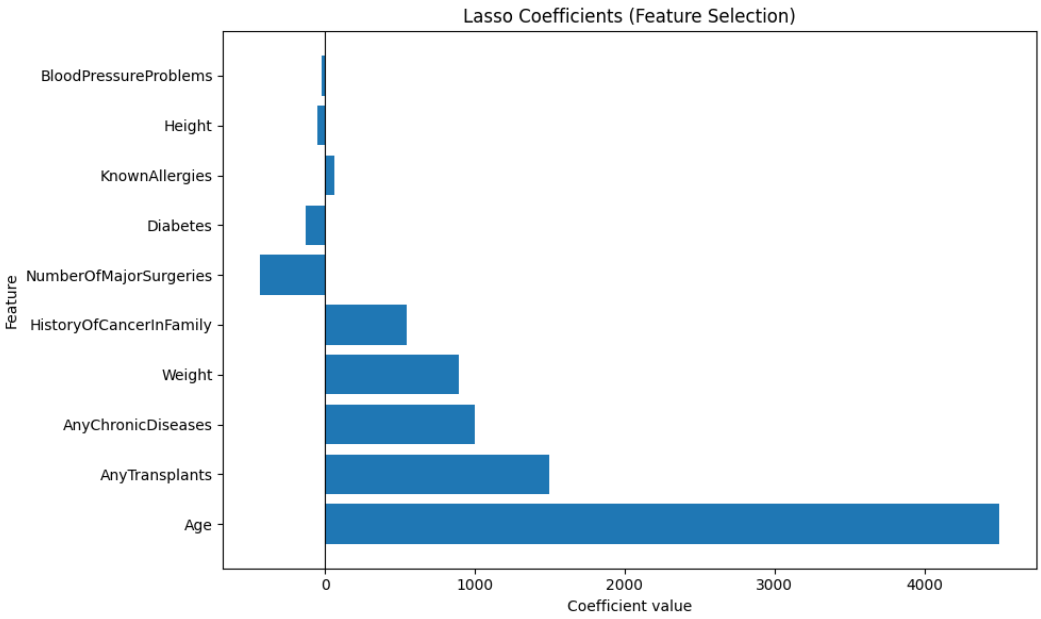


ภาพประกอบ 34 Feature Importances ของแบบจำลอง Random Forest

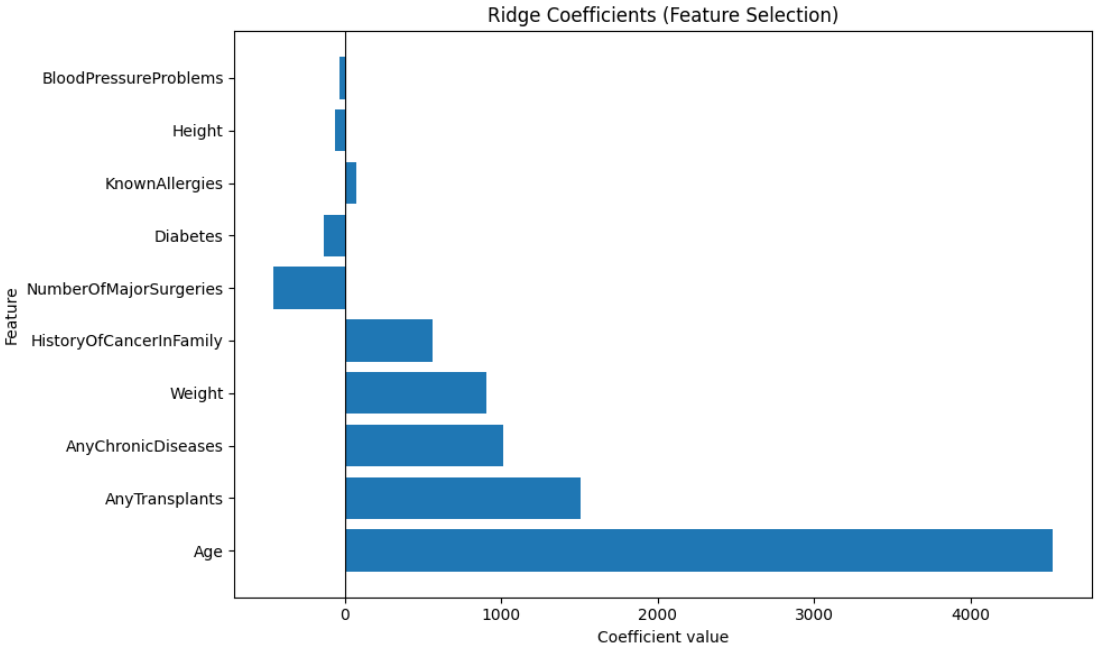


ภาพประกอบ 35 Feature Importances ของแบบจำลอง XGBoots

จากภาพที่ 31, 32 และ33 แสดงให้เห็นว่าตัวแปรที่มีผลมากที่สุดคือ อายุ ก็จะตรงกับสมมติฐานที่ตั้งไว้ ซึ่งในขณะเดียวกันในแต่ละแบบจำลองตัวแปรอื่น ๆ นั้นมีสัดส่วนของความสำคัญที่แตกต่างกันไป ดังนั้นหากต้องการแบบจำลองที่มีประสิทธิภาพสูงสุดในทุกด้าน: ควรเลือก Random Forest ที่ผ่านการปรับตัวแปรแล้วเนื่องจากให้ค่าความผิดพลาดต่ำที่สุดและมีความสามารถในการอธิบายข้อมูลได้ดีที่สุด (R^2 สูงที่สุด) หากให้ความสำคัญกับค่าความผิดพลาดเฉลี่ยต่ำที่สุด ควรเลือก Decision Tree หลังการปรับตัวแปร เพราะมีค่า MAE ต่ำที่สุด ซึ่งหมายถึงข้อผิดพลาดเฉลี่ยต่อข้อมูลแต่ละจุดน้อยที่สุด แม้ว่าจะไม่ได้ดีที่สุดในแง่ของ MAPE และ RMSE ในขณะเดียวกันหากต้องการแบบจำลองที่สมดุลและมีความยืดหยุ่น: XGBoost เป็นตัวเลือกที่ดีเนื่องจากมีประสิทธิภาพดีขึ้นมากจากการปรับค่าตัวแปรและเป็นแบบจำลองที่มีความสามารถในการเรียนรู้ที่ลึกขึ้นมากว่าแบบจำลองต้นไม้ทั่วไป และในส่วนของ SVR ให้ผลลัพธ์ไม่ดีเพราะไม่สามารถจัดการกับข้อมูลที่ซับซ้อนหรือมี noise ได้ดีนัก, Lasso ใช้ L1 regularization ซึ่งอาจตัดตัวแปรที่สำคัญออกทำให้สูญเสียข้อมูลที่จำเป็น, และ Ridge ใช้ L2 regularization ที่ช่วยลด overfitting ซึ่งในแง่ของผลลัพธ์ระหว่าง Lasso และ Ridge นั้นค่อนข้างที่จะให้ผลลัพธ์ใกล้เคียงกันทั้งในแง่ของค่า R^2 ในส่วนของตัวแปรที่สองแบบจำลองนี้ให้ความสำคัญยังคงเป็นตัวแปรอายุตามที่คาดการณ์ไว้ซึ่งจะแสดงให้เห็นดังภาพประกอบ 36 และ 37



ภาพประกอบ 36 Feature Importances ของแบบจำลอง Lasso Regression



ภาพประกอบ 37 Feature Importances ของแบบจำลอง Ridge Regression

จากภาพประกอบ 36 แสดงค่าสัมประสิทธิ์ของ Lasso Regression จะเห็นได้ชัดว่าแบบจำลองได้ทำการคัดเลือกตัวแปรบางตัวออกโดยการกำหนดค่าสัมประสิทธิ์ของตัวแปรที่ไม่มีความสำคัญให้เป็นศูนย์หรือใกล้ศูนย์ เช่น BloodPressureProblems, Height และ KnownAllergies ซึ่งบ่งชี้ว่า Lasso มองว่าตัวแปรเหล่านี้ไม่มีบทบาทสำคัญในการทำนายค่าของเป้าหมาย และสามารถตัดทิ้งได้โดยไม่กระทบต่อประสิทธิภาพของแบบจำลอง ในขณะที่ตัวแปรที่มีค่าสัมประสิทธิ์สูง เช่น Age, AnyTransplants และ AnyChronicDiseases ถือเป็นตัวแปรสำคัญที่มีผลต่อการทำนายและถูกเลือกใช้อย่างชัดเจน ซึ่งสะท้อนถึงคุณสมบัติหลักของ Lasso ที่สามารถทำ Feature Selection ได้ในตัวโดยใช้การลงโทษแบบ L1 (L1 regularization)

ในทางกลับกัน จากภาพประกอบ 37 Ridge Regression ซึ่งใช้การลงโทษแบบ L2 จะไม่ทำการกำจัดตัวแปรออกอย่างเด็ดขาด แต่จะลดขนาดของสัมประสิทธิ์ทั้งหมดให้อยู่ในระดับที่สมดุลมากขึ้น เพื่อป้องกันปัญหา overfitting ดังนั้นทุกตัวแปรยังคงมีค่าสัมประสิทธิ์อยู่ แม้ว่าบางตัวเช่น BloodPressureProblems หรือ KnownAllergies จะมีค่าน้อยมากก็ตาม ซึ่งแสดงว่า Ridge ยังคงพิจารณาข้อมูลจากทุกตัวแปร เพียงแต่ลดอิทธิพลของตัวแปรที่ไม่สำคัญลง ทำให้เหมาะกับกรณีที่เชื่อว่าทุกตัวแปรอาจมีส่วนเกี่ยวข้องเล็กน้อยแต่ไม่ควรถูกตัดทิ้ง

5.3 ข้อเสนอแนะ

SVR, Lasso และ Ridge ไม่ได้ผลลัพธ์ที่ดีขึ้นหลังการปรับจูนไฮเปอร์พารามิเตอร์ แสดงให้เห็นว่าอาจไม่ใช่แบบจำลองที่เหมาะสมกับชุดข้อมูลนี้ หรืออาจต้องใช้วิธีการปรับแต่งที่ซับซ้อนขึ้น เช่น การเลือกตัวแปรที่เหมาะสมกว่านี้ การปรับตัวแปรมีผลอย่างมากกับ Decision Tree, Random Forest และ XGBoost โดยสามารถลดข้อผิดพลาดได้อย่างมีนัยสำคัญ ซึ่งแสดงให้เห็นถึงความสำคัญของการเลือกค่าตัวแปรที่เหมาะสม หากอยากให้ประสิทธิภาพการทำงานของแบบจำลองมีประสิทธิภาพยิ่งขึ้นกว่าอาจจะต้องมองหาชุดข้อมูลที่มีจำนวนข้อมูลมากขึ้นเพื่อให้แบบจำลองได้เกิดการเรียนรู้ที่มากขึ้นซึ่งจะทำให้แบบจำลองมีประสิทธิภาพและในขณะเดียวกันหากใช้ชุดข้อมูลที่มากขึ้นมีความซับซ้อนอาจจะใช้แบบจำลองที่เหมาะสมกับความซับซ้อนของข้อมูลเช่น neural network เป็นต้น และจากเหตุการณ์การแพร่ระบาดของเชื้อไวรัสโควิด ในช่วงปลายปี 2019 ซึ่งส่งผลกระทบอย่างรุนแรงต่อระบบสุขภาพและเศรษฐกิจทั่วโลก รวมถึงภาคธุรกิจประกันภัยที่ต้องเผชิญกับภาระค่าใช้จ่ายในการเคลมที่สูงเกินกว่าที่ระบบการประเมินความเสี่ยงแบบดั้งเดิมจะสามารถรองรับได้ ได้สะท้อนให้เห็นถึงข้อจำกัดสำคัญของกระบวนการตั้งเบี้ยประกันสุขภาพที่อาศัยการวิเคราะห์ข้อมูลสถิติจากอดีตเป็นหลักซึ่งไม่สามารถตอบสนองต่อเหตุการณ์ไม่

คาดฝันที่เกิดขึ้นแบบฉับพลันได้อย่างมีประสิทธิภาพ ซึ่งการประยุกต์ใช้เทคโนโลยีที่มีความสามารถในการเรียนรู้และปรับตัวได้อย่างอัตโนมัติอย่างระบบ AutoML (Automated Machine Learning) มีคุณสมบัติเด่นในการสร้างและปรับแต่งแบบจำลองการเรียนรู้ของเครื่องได้อย่างต่อเนื่อง โดยระบบดังกล่าวสามารถวิเคราะห์ข้อมูลจากหลากหลายแหล่ง ทั้งข้อมูลสุขภาพส่วนบุคคล เช่น อายุ เพศ โรคประจำตัว พฤติกรรมการใช้ชีวิต และประวัติการเคลม รวมถึงข้อมูลภายนอก เช่น แนวโน้มโรคระบาด สภาพแวดล้อม หรือข้อมูลจากหน่วยงานสาธารณสุข เพื่อประมวลผลและแบ่งกลุ่มความเสี่ยงอย่างแม่นยำ และสามารถตั้งเบี้ยประกันได้อย่างยืดหยุ่นและสอดคล้องกับความเสี่ยงเฉพาะรายในแต่ละช่วงเวลา อีกทั้งยังสามารถตรวจจับความผิดปกติของข้อมูล (Anomaly Detection) ได้แบบเรียลไทม์ ซึ่งเป็นสิ่งจำเป็นอย่างยิ่งในสถานการณ์ที่มีการเปลี่ยนแปลงของความเสี่ยงตลอดเวลา เช่นในช่วงการระบาดของโรคติดต่อร้ายแรงอย่างโควิด และด้วยความสามารถของ AutoML ในการเรียนรู้จากข้อมูลขนาดใหญ่และซับซ้อน จึงช่วยให้บริษัทประกันสามารถตั้งเบี้ยได้แม่นยำมากขึ้น ลดความเสี่ยงจากการขาดทุน ตอบสนองต่อสถานการณ์ฉุกเฉินได้อย่างมีประสิทธิภาพ

บรรณานุกรม

1. Statista. Health insurance - worldwide.; 2024 [November 9]. https://www.statista.com/outlook/fmo/insurances/non-life-insurances/health-insurance/worldwide
2. Kasikornbank. รู้จักประกันสุขภาพ (health insurance). 2024 [November 9]. https://www.kasikornbank.com/th/personal/insure/article/pages/what-is-health-insurance.aspx
3. Tejashvi14. Medical insurance premium prediction. 2021 2024 [October 15]; [Indicate that it's a data set by including "[Data set]"]. https://www.kaggle.com/datasets/tejashvi14/medical-insurance-premium-prediction
4. Support vector machines. 2025 [April 5]. https://scikit-learn.org/stable/modules/svm.html
5. Lasso regression. 2025 [April 5]. https://scikit-learn.org/stable/modules/linear_model.html#lasso
6. Guido ACMS. Ridge regression Schanafelt D, บรรณาธิการ. Introduction to machine learning with python. 1. United States of America: O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472; 2017. 49.
7. Decision tree. 2025 [April 5]. https://scikit-learn.org/stable/modules/tree.html
8. Random forest. 2025 [April 5]. https://scikit-learn.org/stable/modules/ensemble.html#forest
9. Tianqi Chen CG. Xgboost: A scalable tree boosting system. 2016.
10. Regression metrics. 2025 [April 6]. https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics
11. Kashish Bhatia MK. Health insurance cost prediction using machine learning. International Conference for Emerging Technology (INCET); May 27-29; Belgaum. 2022.
12. Sudhir Panda SP, Biswajit Purkayastha, Dolly Das, Manomita Chakraborty, Saroj Kumar Biswas. Health insurance cost prediction using regression models. International

Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON); 26-27 May; India. 2022.

13. Keshav Kaushik AB, Ashutosh Dhar Dwivedi ,and Rajani Singh Machine learning-based regression framework to predict health insurance premiums. International Journal of Environmental Research and Public Health. 2022;19(13).
14. N Venkata Sailaja MK, Meera Katkam, Devipriya M, Sreeja M, and Dr.D N Vasundhara. Hybrid regression model for medical insurance cost prediction and recommendation. IEEE International Conference on Intelligent Systems, Smart and Green Technologies (ICISSGT); November 13-14; Isakhapatnam, India. 2021.
15. Teertha. Us health insurance dataset. 2020 2024 [October 15]; [Indicate that it's a data set by including "[Data set]"]. https://www.kaggle.com/datasets/teertha/ushealthinsurancedataset
16. Ugochukwu Orji EU. Machine learning for an explainable cost prediction of medical insurance. Machine Learning with Applications. 2024;15.

ประวัติผู้เขียน