การวิเคราะห์ประกันรถยนต์ด้วยการเรียนรู้ของเครื่อง

ยศหรินทร์ มนพลับ1*, ศิริสรรพ เหล่าหะเกียรติ2

บทคัดย่อ

งานวิจัยนี้เป็นการศึกษาความสัมพันธ์ระหว่างตัวแปรต้นต่างๆ เช่นประวัติการเคลม และระยะเวลาที่ทำประกัน กับตัวแปรตาม นั้นคือจำนวนเคลมที่เกิดขึ้น โดยได้ใช้การเรียนรู้ของเครื่อง 7 โมเดล นั้นคือ Catboost, Regressor, XGBoost Regressor, LightGBM Regressor, Poisson Regressor, Negative binomial Regressor, MLP Regressor และ Tabnet Regressor ได้มีการนำข้อมูลมาประมวลผลด้วยการทำ Catboost encoding และ standard scaler และได้ทำ pre-processing ด้วยการแบ่งข้อมูลออกเป็น 3 ส่วนได้แก่ Test Validation และ Train นอกจากนี้เพื่อให้โมเดลมีประสิทธิภาพมากขึ้น ได้มีการทำ Hyperparameter tuning ในทุกโมเดล และในงานวิจัยนี้มีการทำ Feature importance โดยโมเดล XGboost, Catboost, LightGBM และ Random forest โดยจุดประสงค์ของงานนี้คือการสร้างโมเดลที่จะทำนายจำนวนเคลมได้อย่างแม่นยำ เพื่อให้บริษัทประกันสามารถคำนวณค่าใช้จ่ายได้ และศึกษาว่าตัวแปรต้นตัวใดส่งผลกับจำนวณเคลมมากที่สุด โดยผลงานวิจัยคือ โมเดลที่ดีที่สุดหากดูจากคะแนน MAE คือ Tabnet regressor โดยมี MAEเท่ากับ 0.0787 และหากดูจากคะแนน R_squared และ MSE คือ Catboost regressor โดยมีคะแนนเท่ากับ 0.0339 และ 0.0557 ตามลำดับ ส่วนตัวแปรต้นที่ส่งผลต่อจำนวณเคลมมากที่สุด คือ ระยะเวลาที่กรมธรรม์ส่งผล (Area) อายุรถ (Region) ประวัติการเคลม (BonusMalus) และ ระยะเวลาของกรมธรรม์ (Exposure)

คำสำคัญ : การเรียนรู้ของเครื่อง, ประกันภัยรถยนต์, การปรับจูนไฮเปอร์พารามิเตอร์,

1 หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิทยาการข้อมูล คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ กรุงเทพฯ 10110

2 คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ กรุงเทพฯ 10110

* Corresponding author:Tel.: 0806501333 E-mail address: yodharin.monplub@g.swu.ac.th

MOTOR CLAIM ANALYSIS WITH MACHINE LEARNING MODELS

Yodharin Monplub*, Sirisup Laohakiat[2]

Abstract

This research investigates the relationship between various independent variables—such as claim history and insurance duration—and the dependent variable, which is the number of claims made. Seven machine learning models were used in the study: CatBoost Regressor, XGBoost Regressor, LightGBM Regressor, Poisson Regressor, Negative Binomial Regressor, MLP Regressor, and TabNet Regressor. The data preprocessing included CatBoost encoding and standard scaling, and the dataset was divided into three parts: Training, Validation, and Testing. To improve model performance, hyperparameter tuning was applied to all models. Additionally, feature importance analysis was conducted using the XGBoost, CatBoost, LightGBM, and Random Forest models. The objective of this study is to build a model that can accurately predict the number of insurance claims, enabling insurance companies to better estimate costs, and to identify which independent variables have the most influence on the number of claims. The findings show that the best-performing model based on Mean Absolute Error (MAE) is the TabNet Regressor, with an MAE of 0.0787. When considering R-squared and Mean Squared Error (MSE), the best model is the CatBoost Regressor, with scores of 0.0339 and 0.0557, respectively. The most influential features affecting the number of claims are the duration of the insurance policy (Exposure), vehicle age (Region), claim history (BonusMalus), and the policy duration (Area).

Keywords : Machine Learning, Motor insurance, Hyperparameter tuning

[1] Data Science, Faculty of Science, Srinakharinwirot University, Bangkok, 10110, Thailand

[2] Faculty of Science, Srinakharinwirot University, Bangkok, 10110, Thailand

* Corresponding author:Tel.: 0806501333 E-mail address: yodharin.monplub@g.swu.ac.th

บทนำ

ที่มาและความสำคัญของปัญหา

ปัจจุบันในวงการประกันภัยและคณิตศาสตร์ประกันภัย มีการนำ Generalize Linear Model (GLM)(1) มาคำนวณจำนวนเคลมของประกันภัยรถยนต์ โดยการคำนวณเหล่านี้จะทำให้การเก็บเบี้ยสอดคล้องกับจำนวนเคลมที่แท้จริง และทำให้สามารถเก็บเงินทุนสำรอง (reserve)ได้เหมาะสมมากขึ้น

ข้อดีของ GLM คือ

- มีค่า Betas ที่บอกความสัมพันธ์ระหว่างตัวแปรต้นต่างๆและตัวแปรตาม โดยค่าตัวแปรต้นเปลี่ยนไป 1 หน่วย ตัวแปรตามจะเปลี่ยนไป Beta หน่วย

- มีค่า P-Value ที่บอกระดับนัยสำคัญของ Beta แต่ละตัว โดยถ้าค่า P-value ต่ำกว่า 0.05 จะแปรว่าตัวแปรมีนัยสำคัญทางสถิติ และถ้าค่า P-value มากกว่า 0.05 จะแปรว่าตัวแปรไม่มีนัยสำคัญทางสถิติ

ในขณะเดียวกัน GLM ก็มีข้อเสีย ที่โมเดลการเรื่องรู้ของเครื่องโมเดลอื่นๆอาจไม่มี นั้นคือ

- สมมติฐานที่ว่าตัวแปรต้นและตัวแปรตามต้องมีความสัมพันธ์เชิงเส้นกัน ซึ่งสมมติฐานนี้อาจไม่ได้เป็นจริงเสมอในชุดข้อมูลของวงการประกันภัย

- GLM เป็นโมเดลที่มีความซับซ้อนน้อย อาจให้ค่าทำนายที่ไม่ดีเท่าโมเดลที่มีความซับซ้อนมาก

งานวิจัยนี้จัดทำขึ้นเพื่อศึกษาการเรียนรู้ของเครื่องตัวอื่น(2) และเปรียบเทียบระหว่างการเรียนรู้ของเครื่องตัวอื่นกับ GLM โดยจะดูว่าโมเดลใดให้ค่าทำนายที่ดีกว่า รวมถึงดูการทำ Feature Importance ของแต่ละโมเดลว่าให้ค่าใกล้เคียงกับหรือไม่

ในงานวิจัยนี้มีการใช้โมเดลเรียนรู้ของเครื่องแบบ boosting แบบ ensemble (3) และ แบบ neural network และ แบบ GLM มาทำนายจำนวนเคลม รวมถึงได้มีการทำ Feature importance เพื่อศึกษาว่าตัวแปรต้นตัวใดส่งผลต่อจำนวนเคลมมากที่สุด สิ่งที่คาดหวังจากงานวิจัยนี้คือ เพื่อที่จะสามารถระบุได้ว่าโมเดลแบบใด ตัวใด สามารถทำนายจำนวนเคลมได้ดีที่สุด และดูว่าตัวแปรต้นตัวใดส่งผลกระทบกับจำนวนเคลมมากน้อยเท่าใด

วัตถุประสงค์ของการวิจัย

- เพื่อสนับสนุนให้ใช้โมเดลการเรียนรู้ของเครื่องกล (machine learning model)(4) ที่หลากหลายมากขึ้นในการคำนวณค่าเบี้ยประกันรถยนต์

- เพื่อศึกษาว่าโมเดลตัวใดทำนายค่าเคลมได้ดีกว่าโมเดลตัวอื่น

- เพื่อระบุว่าตัวแปรต้นตัวใดส่งผลต่อจำนวนเคลมมากที่สุด

ประโยชน์ที่คาดว่าจะได้รับจากการวิจัย

- เพื่อให้บริษัทประกันเก็บค่าเงินทุนสำรอง (reserve)(5) ได้อย่างเหมาะสมกับความเสี่ยงมากขึ้น นั้นคือสอดคล้องกับค่าเคลมในอนาคตมากขั้น

- เพื่อให้บริษัทประกันเก็บเบี้ยลูกค้าได้อย่างยุติธรรมมากขึ้น โดยถ้าเสี่ยงมากค่าเบี้ยจะมาก และถ้าเสี่ยงน้อยค่าเบี้ยจะน้อย

ขอบเขตของการวิจัย

งานวิจัยนี้ใช้เทคนิคการเรียนรู้ของเครื่อง (machine learning) และใช้ภาษา python เป็นภาษาในการเขียนโค้ด มีขอบเขตดังนี้

- มีการทำ hyperparameter tuning แต่ทำอย่างจำกัด เนื่องจากทรัพยากรคอมพิวเตอร์มีจำกัด
- มีการพยายามเลือกโมเดลที่สามารถเทรนด้วยการ์ดจอได้ เพราะว่าจะเทรนได้เร็วกว่าและทำ hyperparameter tuning ได้

วิธีดำเนินการวิจัย

- ใช้ข้อมูลเคลมจาก Kaggle ชื่อข้อมูลชุด French Motor Claims Datasets ที่ https://www.kaggle.com/datasets/floser/french-motor-claims-datasets-fremtpl2freq
- มีการทำ data cleaning โดยการเช็คว่าในข้อมูลมีค่าว่าง ค่า Outlier หรือค่าผิดปกติอื่นหรือไม่ และทำการจัดการกับค่าเหล่านั้นด้วยวิธีที่เหมาะสม
- มีการทำ data preprocessing ด้วยการทำ Standard Scaler สำหรับ คอลัมน์ที่เป็นตัวเลข และทำ Catboost encoding สำหรับคอลัมน์หมวดหมู่
- มีการแบ่งข้อมูลออกเป็น 3 ส่วน ได้แก่ Test Train และ Validation data
- มีการสร้างโมเดล และปรับจูน Hyperparameter
- มีการทำ feature importance ด้วย Catboost Xgboost Lightgbm และ Poisson Regressor
- นำค่าตัวชี้วัดต่างๆ นั้นคือ MAE MSE และ R squared มาเปรียบเทียบกันในตารางและดูว่าโมเดลใดให้ค่าดีที่สุด

งานวิจัยที่เกี่ยวข้อง

การทบทวนวรรณกรรมของงานวิจัยนี้ได้ทำการศึกษางานวิจัยที่เกี่ยวข้องกับการนำการเรียนรู้ของเครื่องกลประยุกต์ใช้กับการประกันรถยนต์ดังนี้:

- "Predictive Modeling of Insurance Claims Using Machine Learning Approach for Different Types of Motor Vehicles" โดย V. Selvakumar, Dipak Kumar Satpathi, P. T. V. Praveen Kumar และ V. V. Haragopal

  วัตถุประสงค์หลักของเอกสารวิจัยนี้คือการสร้างแบบจำลองทางคณิตศาสตร์ที่เหมาะสมซึ่งช่วยในการคาดการณ์จำนวนเงินเรียกร้องค่าสินไหมทดแทนจากบุคคลที่สามสำหรับประเภทยานพาหนะต่างๆ โดยอิงจากลักษณะเฉพาะของข้อมูล การทำนายจำนวนเงินเรียกร้องค่าสินไหมทดแทนจากบุคคลที่สามสำหรับประเภทยานพาหนะต่างๆ ถือเป็นงานที่ท้าทาย และมีการศึกษาวิจัยเชิงประจักษ์จำนวนไม่มากที่ได้ทำการทำนายค่าเคลม

  ในการศึกษานี้ มีการรวบรวมข้อมูลประวัติศาสตร์แบบอนุกรมเวลาประจำปีเป็นระยะเวลา 34 ปี ได้มีสร้างแบบจำลองการทำนายด้วยการเรียนรู้ของเครื่องเพื่อสร้างแบบจำลองจำนวนเงินเรียกร้องค่าสินไหมทดแทนสำหรับประเภทยานพาหนะต่างๆ ได้มีการ

แสดงให้เห็นถึงความเป็นไปได้ในการใช้การเรียนรู้ของเครื่องหลายแบบ เช่น แบบจำลองการถดถอยเชิงเส้น (Regression) แบบจำลองการปรับให้เรียบแบบเอ็กซ์โพเนนเชียล (Exponential Smoothing Model) ค่าเฉลี่ยเคลื่อนที่เชิงบูรณาการถดถอยอัตโนมัติ (ARIMA) เครือข่ายประสาทเทียม (ANN) และแบบจำลอง ARIMA-ANN แบบผสม เพื่อคาดการณ์จำนวนเงินเรียกร้องค่าสินไหมทดแทนสำหรับประเภทยานพาหนะต่างๆ

ข้อมูลดังกล่าวได้รับการวิเคราะห์ เปรียบเทียบ และการวิเคราะห์เชิงประจักษ์แสดงให้เห็นว่าเครือข่ายประสาทเทียม (ANN) เป็นแบบจำลองเชิงทำนายที่ดีกว่าเมื่อเทียบกับแบบจำลองอนุกรมเวลาอื่นๆ ที่อิงตามเมตริกการประเมินประสิทธิภาพ RMSE และ MAPE โดยมีความแปรปรวนน้อยกว่า

ดังนั้น การใช้การเรียนรู้ของเครื่องเพื่อคาดการณ์จำนวนเงินเรียกร้องค่าสินไหมทดแทนจากบุคคลที่สามจะช่วยให้บริษัทประกันภัยในอินเดียจัดทำแบบจำลองที่ดีกว่า ทำให้การจ่ายเคลมแม่นยำขึ้นและบริหารค่าเคลมของรถยนต์แต่ละหมวดหมู่ได้ดีกว่า

- "Generalized Linear Models for Dependent Frequency and Severity of Automobile Insurance Claims" โดย by Carina Clemente, Gracinda R. Guerreiro, และ Jorge M.

  โดยทั่วไป จำนวนเคลมและจำนวนเงินเรียกร้องค่าสินไหมทดแทนจะถือว่าเป็นอิสระต่อกันในธุรกิจประกันภัยวินาศภัย บทความนี้จะสำรวจว่าสมมติฐานความเป็นอิสระต่อกันนี้จะยังเป็นจริงอยู่ไหมเมื่อมีการนำระบบ rating (rating factor) มาใช้

  วิธีการทำการวิจัยประกอบด้วยการใช้แบบจำลองแบบ GLM มาทำนายความถี่ของเคลม (frequency) และความรุนแรงของเคลม (severity) โดยโมเดลความสัมพันธ์ระหว่างความถี่และความรุนแรงโดยใช้ Covariate ใน GLM โมเดล

  การทำแบบนี้นอกจากใช้งานง่ายแล้ว ข้อดีคือ เมื่อมีการใช้ Poisson regression ร่วมกับ Log-link สำหรับแบบจำลอง frequency เบี้ยประกันภัยบริสุทธิ์ที่ได้จะเป็นผลคูณของความถี่ frequency และ ความถี่ severity ซึ่งสามารถโมเดลได้โดยใช้การตัดแปลงความถี่ของความน่าจะเป็น

- Machine Learning in Forecasting Motor Insurance Claims โดย by Thomas Poufinas,Periklis Gogas,Theophilos Papadimitrio และ Emmanouil Zaganidis

  การทำนายค่าเคลมที่แม่นยำเป็นสิ่งสำคัญในการดูเงินเข้าเงินออกของบริษัท การตั้งราคา รวมไปจนถึงการกำไรของบริษัทประกันด้วย การทำนายค่าเคลมถูกใช้เป็นตัวแปรต้นในการสร้างแผนธุรกิจ การดูว่าจะรับความเสี่ยงเท่าใด ไปจนถึงการคำนวณว่าจะต้องถือเงินทุนสำรองเท่าไหร่ วิธีการคำนวณแบบเก่าคือคำนวณค่าความถี่ (frequency) และความรุนแรง (severity) แยกกัน แต่ในงานวิจัยนี้มีการเสนอวิธีการทำรานค่าเคลมแบบใหม่ โดยมีชุดตัวแปรสองตัวแปร คือ ข้อมูลอากาศ และข้อมูลยอดขาย มีการนำวิธีการเรียนรู้ของเครื่องหลายวิธี คือ Support Vector Machines—SVM, Decision Trees, Random Forests, และ Boosting มาทำนายค่าเคลมต่อหนึ่งคันต่อสามเดือน สุดท้าย มีการเรียกดูว่าตัวแปรตัวไหนส่งผลต่อค่าเคลมมากที่สุด

  ชุดข้อมูลของเรามาจากบริษัทประกันภัยในกรุงเอเธนส์ ประเทศกรีซ ในปี 2008 ถึง 2020 ตัวแปรสามตัวที่ส่งผลต่อค่าเคลมมากที่สุดคือยอดขายรถใน3เดือนที่แล้ว และ9เดือนที่แล้ว และอุณหภูมิต่ำที่สุดใน Elefsina (หนึ่งในสถานีตรวจอากาศในกรุงเอเธนส์) ในบรรดาโมเดลที่สร้าง โมเดล Random forest และ XGboost เป็นโมเดลที่ดีที่สุด

  สามารถสรุปได้ว่าข้อมูลยอดขายรถและข้อมูลอุณหภูมิสามารถทำนายค่าเคลมของประกันรถยนต์ได้ดีที่สุด

- Motor Insurance Claim Status Prediction using Machine Learning Techniques โดย by Endalew Alamir, Teklu Urgessa, Ashebir Hunegnaw และ Tiruveedula Gopikrishna

  การทำนายค่าเคลมเป็นปัญหาเบสิกที่บริษัทประกันต้องเจอ โดยบริษัทประกันมักพบเจอกันปัญหาค่าเคลมเพิ่มขึ้น โดยเกิดจากการฉ้อโกงการเคลมเป็นส่วนใหญ่ เป็นเรื่องยากที่บริษัทประกันจะแยกได้ว่าเคลมใดเป็นเคลมฉ้อโกง ดังนั้นงานวิจัยนี้จึงมีเป้าหมายเพื่อแยกว่าเคลมใดคือเคลมฉ้อโกง

  ในการจัดเตรียมข้อมูล มีการใช้ Missing value ratio Z- Score encoding techniques และ entropy เพื่อจัดเตรียมข้อมูล จากนั้นมีการทำ k-fold cross validation เพื่อแบ่งข้อมูลเป็น test และ train โมเดลที่ใช้คือ Random Forest และ Multiclass-SVM มีการคำนวณค่า Accuracy Precision Recall และ F1 โดยค่า Accuracy ของ Random forest และMulticlass-SVM สูงถึง 98.36% และ 98.17% ตามลำดับ สามารถสรุปได้ว่าโมเดล Random forest ได้คะแนนสูงกว่าเล็กน้อย

- Modelling Motor Insurance Claim Frequency and Severity Using Gradient Boosting โดย Carina Clemente, Gracinda R. Guerreiroi และ Jorge M. Bravo

  การทำนายค่าความถี่การเคลมและจำนวนเงินต่อหนึ่งเคลมเป็นเรื่องสำคัญในวงการประกันวินาศภัย โดยเฉพาะอย่างยิ่งการทำการรับประกัน (underwriting) การกำหนดอัตรา (ratemaking) และการสำรอง (reserving) โมเดล GLM มีกำหนดว่าตัวแปรต้นและตัวแปรตามต้องมีความสัมพันธ์เชิงเส้นกัน ซึ่งในบางทีความสัมพันธ์นี้อาจไม่เป็นจริง งานวิจัยนี้จึงได้ใช้ Gradient boost ที่ไม่ได้บังคับว่าตัวแปรต้นและตัวแปรตามต้องมีความสัมพันธ์เชิงเส้นกัน และเอามาเปรียบเทียบกับ GLM

  ผลที่ได้คือในกี่ทำนายความถี่การเคลม Gradient boost ให้ผลลัพธ์ที่ดีกว่าGLM แบบ Poisson regression แต่ในจำนวนเงินต่อหนึ่งเคลม GLM กลับทำงานได้ดีกว่า gradient boost
  สามารถสรุปได้ว่า gradient boost เป็นโมเดลที่ตัวแปรต้นและตัวแปรตามไม่ต้องมีความสัมพันธ์เชิงเส้นกัน และ gradient boost สามารถจับความสัมพันธ์ที่ซับซ้อนได้ เพราะฉะนั้น gradient boost ถือเป็นอุปกรณ์สำคัญสำหรับวงการประกันภัยและการบริหารความเสี่ยง

ทฤษฎีที่เกี่ยวข้อง

ทฤษฎีที่เกี่ยวข้องกับแบบจำลอง Machine learning แบบ gradient boost

Gradient Boosting (GB) เป็นเทคนิคหนึ่งในการสร้างโมเดลเชิงทำนาย (predictive model) ซึ่งอยู่ในกลุ่มของวิธีการ “ensemble learning” หมายถึงการผสานโมเดลย่อยหลาย ๆ ตัวเข้าด้วยกันเพื่อเพิ่มประสิทธิภาพในการพยากรณ์หรือการจำแนก (classification) โดย Gradient Boosting จะค่อย ๆ สร้างโมเดลย่อยขึ้นมาแบบทีละขั้นตอน (stage-wise) แล้วปรับปรุงให้ดีขึ้นเรื่อย ๆ ตามแนวคิดของการไล่ระดับ (gradient) ของฟังก์ชันความสูญเสีย (loss function) ทำให้ได้โมเดลสุดท้ายที่สามารถทำนายได้แม่นยำกว่าโมเดลที่ฝึกเพียงตัวเดียว เริ่มต้นด้วยการสร้างโมเดลพื้นฐาน (base model)

แนวคิดหลักของ Gradient Boosting

- เริ่มต้นด้วยโมเดลพื้นฐาน (Base Model)

  โดยทั่วไป โมเดลพื้นฐาน (หรือ weak learner) ที่นิยมใช้คือ ต้นไม้การตัดสินใจ (Decision Tree) ขนาดเล็กที่มีความลึกต่ำ (shallow tree) เช่น ต้นไม้ที่มีความลึก 1-5 ระดับ โดยเหตุผลที่ใช้ต้นไม้ขนาดเล็กคือเพื่อป้องการ overfit และทำให้ฝึกได้เร็ว

- คำนวณค่าความผิดพลาด (Residuals)

  เมื่อได้โมเดลพื้นฐานแล้ว เราจะทดสอบโมเดลนั้นกับข้อมูลฝึก (training data) และคำนวณความคลาดเคลื่อนระหว่างค่าที่แท้จริง (y) กับค่าที่โมเดลทำนาย ($\hat{\mathbf{y}}$)

  ความคลาดเคลื่อนนี้เรียกว่า “residuals” ซึ่งในกรณีของปัญหาการพยากรณ์เชิงตัวเลข (regression) จะคิดเป็น $\boldsymbol{r_i = y_i - \hat{y}_i}$

- สร้างโมเดลใหม่เพื่อพยากรณ์ค่าความผิดพลาด

  จาก residuals ที่ได้ เราจะสร้างโมเดลใหม่ (weak learner ตัวที่สอง) เพื่อทำนาย residuals เหล่านี้ และ หากเป็นปัญหาการจำแนก (classification) มักจะใช้แนวคิดของการเรียนรู้จากค่าลบของเกรเดียนต์ (negative gradient of loss function) เป็นตัวแทนของ “residual” เช่นกัน

- รวมโมเดลเข้าด้วยกัน (Update Model)

  เมื่อฝึกโมเดลใหม่เสร็จแล้ว ก็จะนำโมเดลนี้มาผสานรวมกับโมเดลเดิม โดยมีการถ่วงน้ำหนัก (learning rate หรือ shrinkage) เพื่อควบคุมไม่ให้โมเดลใหม่ “แทรกแซง” โมเดลเก่ามากเกินไป โดย กระบวนการรวมมักจะอยู่ในรูปของ

  $$\hat{y}_{new} = \hat{y}_{old} + v * h_{new}(x)$$

  โดย

  V คือ learning rate ที่เป็นตัวกำหนดความสำคัญของโมเดลใหม่ $\boldsymbol{h_{new}(x)}$

  $\hat{y}_{new}$ คือ **การคาดการณ์ใหม่** ของโมเดลรวมหลังจากที่ได้เพิ่มโมเดลย่อยตัวล่าสุดเข้าไป

  $\hat{y}_{old}$ คือ **การคาดการณ์เดิม** ของโมเดลรวมจากรอบการวนซ้ำก่อนหน้า ซึ่งเป็นผลรวมของการคาดการณ์จากโมเดลย่อยทั้งหมดที่สร้างขึ้นมาแล้ว

  $h_{new}(x)$ คือ โมเดลย่อย (weak learner) ตัวใหม่ที่เพิ่งถูกสร้างขึ้นมาในรอบการวนซ้ำปัจจุบัน โมเดลนี้ถูกฝึกเพื่อ ทำนายค่าความผิดพลาด (residuals) หรือ negative gradient ของโมเดลรวมเดิม

- ทำซ้ำ (Iterative Process)

  ทำขั้นตอนการคำนวณ residuals -> สร้างโมเดลใหม่ -> รวมเข้าไปในโมเดลเดิม ซ้ำไปเรื่อย ๆ ตามจำนวนรอบ (iterations) ที่

กำหนด หรือจนกว่าโมเดลจะไม่สามารถลดค่าความผิดพลาดได้อีก โดยแต่ละรอบจะได้โมเดลย่อย (weak learner) มาเสริมความสามารถของโมเดลก่อนหน้า จนสุดท้ายได้ “strong learner” ที่มีความแม่นยำสูงขึ้น

รายละเอียดเพิ่มเติมในมุมทางคณิตศาสตร์

- Loss Function

  Gradient Boosting จะกำหนดฟังก์ชันความสูญเสีย (loss function) ไว้ เช่น Mean Squared Error (MSE) หรือ Log Loss (Binary Cross-Entropy) แล้วประเมินว่าโมเดลที่สร้างขึ้นในแต่ละรอบสามารถลด loss ได้มากน้อยแค่ไหน โดยในแต่ละรอบของการเรียนรู้ เราจะมอง residuals เป็นค่าลบของเกรเดียนต์ของ loss function เพื่อบอกทิศทางที่โมเดลควรปรับค่าเพื่อให้ loss ลดลง

- Line Search

  บ่อยครั้งเพื่อความแม่นยำในการหาค่าน้ำหนัก (learning rate) หรือค่าพารามิเตอร์อื่นๆ เราอาจทำ line search เพื่อหาค่าที่เหมาะสมในการอัปเดตโมเดลแต่ละครั้ง

- Regularization

  นอกจาก learning rate จะช่วยลดความรุนแรงของการอัปเดตแล้ว บางครั้งเรายังใช้วิธีอย่างการกำหนดความลึกสูงสุดของต้นไม้ (max depth) การกำหนดจำนวนใบสูงสุด (max leaves) หรือกำหนดค่า minimum samples split เพื่อป้องกันไม่ให้ต้นไม้ overfit เกินไป

- อีกเทคนิคหนึ่งก็คือการสุ่มตัวอย่างข้อมูล (subsample) หรือการสุ่มตัวอย่างฟีเจอร์ (feature subsampling) ในแต่ละรอบของการสร้างต้นไม้คล้ายกับแนวคิดของ Random Forest เพื่อช่วยลดความสัมพันธ์ (correlation) ระหว่างต้นไม้แต่ละต้น

Gradient Boosting เป็นเทคนิคที่ทรงพลังในการปรับปรุงความสามารถในการทำนายของโมเดล เพราะการเสริมโมเดล (boost) ทีละเล็กละน้อยบนฐานของ weak learner หลายตัว ทำให้โมเดลสุดท้ายมีความแม่นยำสูง อย่างไรก็ตาม ต้องใช้งานอย่างระมัดระวัง ทั้งในด้านการเลือกรูปแบบต้นไม้ (depth) การตั้งค่า learning rate และการกำหนดจำนวนรอบที่เหมาะสม เพื่อหลีกเลี่ยงปัญหา overfitting และการใช้ทรัพยากรที่มากเกินจำเป็น

ด้วยหลักการเหล่านี้ Gradient Boosting จึงได้รับความนิยมอย่างแพร่หลายในงานวิจัยและอุตสาหกรรม ตั้งแต่การวิเคราะห์ข้อมูลเชิงธุรกิจ งานด้านการเงิน การตลาด การประมวลผลภาษาธรรมชาติ ไปจนถึงงานด้านภาพและเสียง เพราะมีความยืดหยุ่นสูงและมักให้ผลลัพธ์ที่ดีเยี่ยมเมื่อได้รับการปรับแต่งอย่างเหมาะสม

Gradient Boosting มี hyperparameter หลายอย่าง โดยตาราง 1 แสดงถึงตัวอย่าง hyperparameter นั้น

ตาราง 1 hyperparameter ของ gradient boost

hyperparameter	description
n_estimators	จำนวน weak learners (โดยมากคือ decision trees) ที่จะสร้างต่อเนื่องกัน โดย ยิ่งมีมากอาจช่วยให้

	โมเดลเรียนรู้ได้ละเอียดขึ้น แต่ก็ใช้เวลาฝึก (train) มากขึ้น และเสี่ยงต่อการ overfitting หาก learning rate สูง
learning_rate	ค่าที่ใช้ควบคุม “ความแรง” ในการอัปเดตโมเดลใหม่ในแต่ละรอบ ค่ายิ่งเล็ก การปรับโมเดลในแต่ละรอบจะค่อยเป็นค่อยไป ทำให้ต้องใช้ n_estimators สูงขึ้น แต่ลดโอกาส overfitting ค่ายิ่งใหญ่ โมเดลจะเรียนรู้เร็ว แต่เสี่ยง overfitting
max_depth	ความลึกสูงสุดของแต่ละต้นไม้ (weak learner) ต้นไม้ลึกมากอาจจับความซับซ้อนของข้อมูลได้สูง แต่มีโอกาส overfitting มากขึ้น

ทฤษฎีที่เกี่ยวข้องกับแบบจำลอง Machine learning แบบ CatBoost (Categorical Boosting)

CatBoost (Categorical Boosting) เป็นเฟรมเวิร์ก Gradient Boosting ที่ถูกพัฒนาโดย Yandex โดยมีจุดเด่นในการจัดการกับตัวแปรเชิงหมวดหมู่ (categorical features) ได้อย่างมีประสิทธิภาพ ซึ่งเป็นหนึ่งในปัญหาหลักเมื่อเราทำงานกับชุดข้อมูลในโลกความเป็นจริง อัลกอริทึมของ CatBoost จะใช้วิธีการ “Ordered Target Statistics” หรือ “Ordered Boosting” ที่ช่วยลดปัญหาการรั่วไหลของข้อมูล (target leakage) ในขั้นตอนการเข้ารหัส (encoding) นอกจากนี้ยังปรับปรุงกระบวนการเรียนรู้ด้วยแนวคิดอย่าง symmetric decision trees ที่ทำให้การสร้างต้นไม้มีความสมดุลและเสถียรกว่า ในแง่ของประสิทธิภาพ CatBoost ยังมีการปรับแต่งงานคำนวณเพื่อให้สามารถรันบน GPU ได้ดี และมีการควบคุมค่าพารามิเตอร์ที่เอื้อต่อการป้องกัน overfitting รวมถึงสนับสนุนการ tune ค่าต่าง ๆ ได้หลายวิธี

CatBoost เหมาะกับงานที่มีตัวแปรเชิงหมวดหมู่จำนวนมาก เช่น งานการตลาด การวิเคราะห์ลูกค้า การพยากรณ์ยอดขาย ไปจนถึงงานที่เกี่ยวกับการประมวลผลภาษาธรรมชาติ (NLP) โดยที่ตัวโมเดลจะสามารถเรียนรู้แบบ “end-to-end” กับฟีเจอร์เชิงหมวดหมู่ได้เลย ไม่จำเป็นต้องแปลงหรือเข้ารหัสล่วงหน้าหลายขั้นตอนเหมือนการใช้ One-Hot Encoding หรือ Label Encoding แบบทั่วไป ทำให้ลดเวลาในการเตรียมข้อมูล (feature engineering) และยังคงรักษาความแม่นยำในการทำนายได้ดี ทั้งนี้ CatBoost ยังมาพร้อมกับฟังก์ชันการวิเคราะห์ feature importance และเครื่องมือ visualizations ที่ช่วยให้ผู้ใช้งานเข้าใจโมเดลและนำไปประยุกต์ใช้ได้อย่างมีประสิทธิภาพในหลากหลายโดเมนของอุตสาหกรรม

แนวคิดหลักของ Catboost

- รองรับ Categorical Features โดยตรง

  เป็นหนึ่งในไม่กี่โมเดลที่สามารถรับค่าข้อมูลแบบ string หรือ category ได้ทันทีโดยไม่ต้องแปลงค่า โดยใช้เทคนิคที่เรียกว่า **Ordered Target Statistics** แทนการเข้ารหัสแบบดั้งเดิม ทำให้ประหยัดเวลา preprocessing และลดโอกาสการ leakage ของข้อมูล

- ป้องกัน Overfitting ได้ดี

  ใช้เทคนิค Ordered Boosting แทนการ boosting แบบดั้งเดิม เพื่อหลีกเลี่ยง target leakage และ overfitting

- สามารถใช้งานทั้ง Regression และ Classification

  รองรับทั้งงานพยากรณ์ตัวเลข (regression) และการจำแนกประเภท (classification) และ รองรับหลายประเภทของ loss function

- ความเร็วในการฝึกและทำนาย

  ถึงแม้ว่า CatBoost จะไม่เร็วเท่า LightGBM ในบางกรณี แต่มันถูกปรับให้สามารถใช้ทั้ง CPU และ GPU ได้อย่างมีประสิทธิภาพ

- ใช้งานง่ายและมีเอกสารครบถ้วน

  API ที่ใช้งานง่ายคล้ายกับ scikit-learn และมี document และ tutorial ที่เข้าใจง่าย

ตาราง 2 ข้างล่างแสดงถึงพารามิเตอร์ของ Catboost regressor โดยพารามิเตอร์ที่สำคัญได้แก่ iteration learning_rate และ depth

ตาราง *2* hyperparameter ของ catboost

hyperparameter	description
iterations	จำนวนรอบของ boosting
learning_rate	ความเร็วในการเรียนรู้
depth	ความลึกของแต่ละ decision tree
cat_features	รายการของคอลัมน์ที่เป็น categorical
loss_function	รูปแบบของ loss (เช่น RMSE, Logloss)
early_stopping_rounds	หยุดเทรนหาก performance ไม่ดีขึ้นในรอบที่กำหนด
eval_metric	รูปแบบของ loss (เช่น RMSE, Logloss)
verbose	ระดับการแสดงผลระหว่างเทรน

ทฤษฎีที่เกี่ยวข้องกับแบบจำลอง Machine learning แบบ XGBoost (Extreme Gradient Boosting)

XGBoost (Extreme Gradient Boosting) เป็นไลบรารีที่พัฒนาขึ้นเพื่อเสริมประสิทธิภาพของวิธีการ Gradient Boosting โดยเน้นด้านความเร็วและความยืดหยุ่นในการใช้งาน อัลกอริทึมของ XGBoost ได้รับการปรับปรุงให้ทำงานแบบขนาน (parallel) และรองรับงานประมวลผลแบบกระจาย (distributed computing) ทำให้สามารถจัดการกับข้อมูลขนาดใหญ่ (big data) ได้อย่างมีประสิทธิภาพ ยิ่งไปกว่านั้น XGBoost ยังมีการปรับใช้เทคนิค Regularization เช่น L1 และ L2 เพื่อลดโอกาสในการเกิด overfitting รวมถึงมีการจัดการ missing values โดยอัตโนมัติในขั้นตอนการสร้างต้นไม้ ซึ่งช่วยให้การเรียนรู้ข้อมูลที่มีช่องว่างหรือข้อมูลที่ไม่สมบูรณ์เป็นไปได้สะดวกขึ้น

XGBoost ยังมีระบบที่เอื้อให้ทำการ tuning ค่าพารามิเตอร์ได้หลากหลาย เช่น learning rate, max depth, subsample และ colsample_bytree เพื่อลดความซับซ้อนของโมเดลและเพิ่มความสามารถในการทำนายให้อยู่ในระดับที่เหมาะสมกับงานแต่ละประเภท นอกจากนี้ยังมีฟังก์ชันในการวิเคราะห์ความสำคัญของฟีเจอร์ (feature importance) ที่ช่วยให้นักวิเคราะห์และนักวิจัยเข้าใจถึงปัจจัยที่มีผล

ต่อการตัดสินใจของโมเดลได้ชัดเจน XGBoost จึงได้รับความนิยมอย่างแพร่หลายในการแข่งขันวิเคราะห์ข้อมูล (data competition) และการทำโครงการต่าง ๆ ทั้งในภาคธุรกิจและภาคอุตสาหกรรมทั่วโลก

แนวคิดหลักของ XGBoost

- Gradient Tree Boosting

  XGBoost ใช้การสร้างต้นไม้ (Decision Trees) หลาย ๆ ต้น และปรับค่าด้วย gradient descent ของ loss function โดยทุกต้นไม้ใหม่จะช่วยลดข้อผิดพลาดของต้นไม้ก่อนหน้า

- Regularization

  XGBoost เพิ่มการควบคุมพารามิเตอร์ (Regularization) เช่น L1, L2 เพื่อ ป้องกัน Overfitting

- Parallelization

  สามารถฝึกแบบขนาน (parallel) ได้ ทำให้ รวดเร็วมาก แม้ในข้อมูลขนาดใหญ่

- Sparsity Aware

  รองรับข้อมูลที่มีช่องว่าง (missing values หรือ sparse data) ได้อย่างดี

- Weighted Quantile Sketch

  ใช้เทคนิคเฉพาะในการหาจุดแบ่งของ decision tree ได้แม่นยำและเร็ว

ตาราง 3 ข้างล่างแสดงถึงพารามิเตอร์ของ XGboost โดยพารามิเตอร์ที่สำคัญคือ n_estimators max_depth และ learning_rate

ตาราง *3* hyperparameter ของ XGboost

hyperparameter	description
n_estimators	จำนวนต้นไม้ (รอบ boosting)
max_depth	ความเร็วในการเรียนรู้
learning_rate	ขนาดของก้าวที่ใช้ในแต่ละรอบ
subsample	สัดส่วนของข้อมูลที่ใช้ฝึกในแต่ละรอบ
colsample_bytree	สัดส่วนของฟีเจอร์ที่ใช้ในแต่ละต้นไม้
reg_alpha	L1 regularization (ค่า alpha)
reg_lambda	L2 regularization (ค่า lambda)
objective	ฟังก์ชันเป้าหมาย เช่น reg:squarederror, binary:logistic

ทฤษฎีที่เกี่ยวข้องกับแบบจำลอง Machine learning แบบ CatBoost (Categorical Boosting)

LightGBM (Light Gradient Boosting Machine) เป็นไลบรารี Gradient Boosting ที่ถูกพัฒนาโดย Microsoft โดยมีจุดเด่นเรื่องความเร็วในการประมวลผลและความสามารถในการจัดการกับข้อมูลที่มีขนาดใหญ่ได้ดี เทคนิคสำคัญของ LightGBM คือการใช้วิธี “Gradient-based One-Side Sampling (GOSS)” และ “Exclusive Feature Bundling (EFB)” ซึ่งช่วยลดปริมาณการคำนวณและความซ้ำซ้อนในการฝึกโมเดล นอกจากนี้ LightGBM ยังใช้โครงสร้างต้นไม้ในรูปแบบ “leaf-wise” แทนการแบ่งตามระดับ (level-wise) ซึ่งทำให้สามารถลดค่าความผิดพลาดได้มากขึ้นในแต่ละรอบการสร้างต้นไม้ ส่งผลให้โมเดลสามารถเรียนรู้ได้เร็วและมีประสิทธิภาพสูง

LightGBM มีการประมวลผลในรูปแบบของการสร้างฮิสโตแกรม (histogram-based decision tree) ที่ช่วยบริหารจัดการหน่วยความจำได้อย่างมีประสิทธิภาพ อีกทั้งยังรองรับการประมวลผลแบบขนาน (parallel) และแบบกระจาย (distributed computing) ได้เช่นเดียวกับ XGBoost และ CatBoost นอกจากนี้ยังมีตัวเลือกมากมายให้ผู้ใช้งานปรับแต่ง (hyperparameter tuning) ทั้งในเรื่องของความลึกของต้นไม้ (max depth) จำนวนใบสูงสุด (max leaves) ค่า learning rate และอื่น ๆ ซึ่งหากตั้งค่าได้อย่างเหมาะสม ก็จะช่วยลดโอกาสเกิด overfitting และเพิ่มความแม่นยำในการทำนายได้เป็นอย่างดี ทำให้ LightGBM ได้รับความนิยมในงานที่ต้องรับมือกับปริมาณข้อมูลขนาดใหญ่ และงานที่ต้องการความเร็วในการทำนาย (inference) สูงในระดับอุตสาหกรรม

แนวคิดของ LightGBM

- Leaf-wise Tree Growth

  แทนที่จะเติบโตต้นไม้แบบระดับต่อระดับ (level-wise) LightGBM จะเลือกแตก "ใบไม้ (leaf)" ที่ลด loss ได้มากที่สุดก่อน โดยส่งผลให้โมเดลแม่นยำขึ้น แต่ก็มีโอกาส overfitting ได้มากขึ้นถ้าไม่ควบคุมพารามิเตอร์ให้ดี

- Histogram-based Algorithm

  แทนที่จะใช้ค่าตัวเลขต่อเนื่องตรง ๆ LightGBM จะทำ binning ของ feature เป็น histogram เพื่อลดเวลาในการคำนวณและใช้หน่วยความจำน้อยลง

- Gradient-based One-Side Sampling (GOSS)

  ใช้เฉพาะตัวอย่างที่มี gradient สูงร่วมกับการสุ่มเลือกตัวอย่างที่มี gradient ต่ำ เพื่อรักษาความแม่นยำและลดต้นทุนในการคำนวณ

- Exclusive Feature Bundling (EFB)

  รวม feature ที่ไม่เปิดใช้งานพร้อมกัน (mutually exclusive) เข้าด้วยกันเป็น feature เดียวเพื่อประหยัดหน่วยความจำ

ตาราง 5 ข้างล่างแสดงถึงพารามิเตอร์ของ LightGBM โดยพารามิเตอร์ที่สำคัญคือ n_estimators learning_rate และ max_depth

ตาราง 4 hyperparameter ของ LightGBM

ตาราง *4* hyperparameter ของ LightGBM

พารามิเตอร์	ความหมาย

n_estimators	จำนวนรอบของ boosting
learning rate	ความเร็วในการเรียนรู้
max_depth	ความลึกของต้นไม้
num_leaves	จำนวนใบในแต่ละต้นไม้ (ควบคุมความซับซ้อน)
min_data_in_leaf	จำนวนข้อมูลขั้นต่ำในแต่ละใบ
subsample	สัดส่วนของข้อมูลที่ใช้ในแต่ละรอบ
colsample_bytree	สัดส่วนของ feature ที่สุ่มเลือกในแต่ละต้น
reg_alpha / reg_lambda	Regularization L1 / L2
objective	เช่น regression, binary, multiclass
boosting_type	ปกติใช้ gbdt, หรือ dart, goss

ทฤษฎีที่เกี่ยวข้องกับแบบจำลอง Neural network แบบ TabNetRegressor

TabNet เป็นโมเดลการเรียนรู้เชิงลึก (Deep Learning) ที่ออกแบบมาเฉพาะสำหรับข้อมูลเชิงตาราง (Tabular Data) ซึ่งมีลักษณะการทำงานที่แตกต่างจากโมเดลการถดถอย (Regression) ทั่วไป โดย TabNet นั้นจะใช้เทคนิคที่เรียกว่า **Attentive Feature Selection** เพื่อเลือกเฉพาะคุณลักษณะที่มีความสำคัญสูงต่อการพยากรณ์ ซึ่งช่วยเพิ่มประสิทธิภาพของโมเดลและช่วยให้เกิดการเรียนรู้ที่มีความแม่นยำสูงในข้อมูลที่มีโครงสร้างซับซ้อน โมเดล TabNet ถูกใช้อย่างแพร่หลายในการแก้ปัญหาการพยากรณ์เชิงถดถอย

TabNet ใช้โครงสร้างที่เรียกว่า **Decision Step** ซึ่งแบ่งเป็นหลายชั้น (Layers) ในการคำนวณ โดยมีส่วนที่เรียกว่า **Attentive Transformer** ที่จะปรับน้ำหนักคุณลักษณะ (Feature Weights) และเลือกเฉพาะคุณลักษณะที่สำคัญในแต่ละรอบของการเรียนรู้ ทำให้ TabNet มีความสามารถในการทำงานคล้ายคลึงกับทั้ง Deep Neural Network และ Decision Tree-based Models

สถาปัตยกรรมของ TabNet (แนวคิดเบื้องหลัง)

- Feature Transformer + Attentive Transformer

  ใช้ **transformer** ภายในสำหรับเรียนรู้จากข้อมูลตาราง และมี layer ที่ทำหน้าที่ “เลือกดูฟีเจอร์สำคัญ” ผ่านกลไก **attention**

- Sparse Feature Selection

  TabNet จะเลือก feature แต่ละตัวแบบไม่ใช้ทั้งหมดในแต่ละรอบ ⟶ ทำให้โมเดล **ตีความได้** และลด overfitting

- Decision Steps

  ข้อมูลจะไหลผ่าน “ขั้นตอนการตัดสินใจ (decision steps)” หลายขั้น เพื่อปรับปรุงการเรียนรู้แบบลำดับ

- Explainability

  TabNet สามารถบอกได้ว่าในแต่ละ sample ใช้ feature ไหนมากที่สุดในการตัดสินใจ

ตาราง 5 แสดงถึงพารามิเตอร์ของ Tabnet Regressor โดยพารามิเตอร์ที่ถูกปรับจูนในงานวิจัยนี้คือ n_d และ n_a

ตาราง *5* hyperparameter ของ Tabnet

พารามิเตอร์	ความหมาย
n_d, n_a	ขนาดของการฝึกในแต่ละ block
n_steps	จำนวน decision steps
Gamma	ควบคุมความต่อเนื่องของ attention
lambda_sparse	ค่าควบคุมความเบาบางของ attention
mask_type	รูปแบบของ attention เช่น `"entmax"`

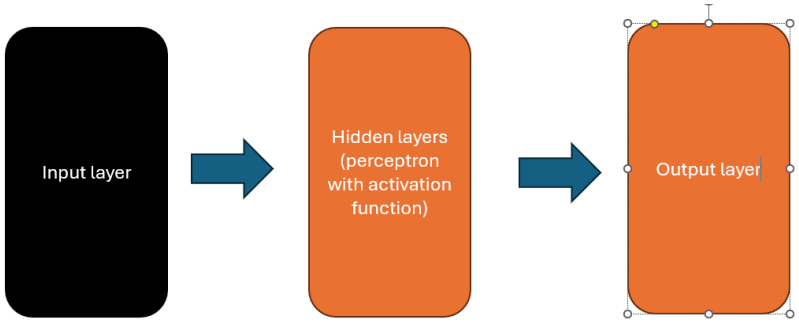
สรุปแล้ว TabNet Regressor คือโมเดล Deep Learning ที่ถูกออกแบบมาสำหรับข้อมูลตารางโดยเฉพาะ ซึ่งสามารถเรียนรู้ข้อมูลได้ลึก มีความสามารถในการ “เลือก” ฟีเจอร์ที่สำคัญได้เองในแต่ละ sample ทำให้โมเดล ตีความได้ และ ยืดหยุ่นมาก

แม้จะใช้ทรัพยากรสูงกว่า XGBoost/LightGBM แต่ในบางกรณีที่ข้อมูลมีความซับซ้อน หรือเมื่อเราต้องการการตีความแบบละเอียด TabNet เป็นอีกทางเลือกที่น่าสนใจมาก

ทฤษฎีที่เกี่ยวข้องกับแบบจำลอง Neural network แบบ MLPRegressor

MLPRegressor (Multilayer Perceptron Regressor) เป็นโมเดลโครงข่ายประสาทเทียม (Artificial Neural Network) ที่ถูกออกแบบมาสำหรับการพยากรณ์ค่าต่อเนื่อง (Regression) ซึ่งได้รับการศึกษาอย่างแพร่หลายในงานวิจัยต่าง ๆ โดยเฉพาะอย่างยิ่งในงานที่ต้องการความสามารถในการประมวลผลข้อมูลที่ซับซ้อนและมีความสัมพันธ์ไม่เป็นเชิงเส้น (Non-linear Relationship)

MLP regressor จะมีโครงสร้างดังภาพประกอบ 1 โดยมีรายละเอียดดังด้านล่าง



ภาพประกอบ 1 โครงสร้างของ MLP regressor

- Input layer: รับข้อมูลเข้าจากชุดข้อมูล

- Hidden Layers: ทำหน้าที่แปลงข้อมูลให้มีความซับซ้อนและมีมิติมากขึ้น โดยการใช้นิวรอนในแต่ละชั้นที่มีฟังก์ชันการเปิดใช้งาน (Activation Function) เช่น ReLU หรือ Tanh เพื่อจัดการกับความไม่เป็นเชิงเส้น (Non-linear) ของข้อมูล

- Output Layer: ให้ผลลัพธ์เป็นค่าตัวเลขสำหรับปัญหาการถดถอย

โดยในการประมาณค่าพารามิเตอร์ จะใช้เทคนิค Back propagation นั้นคือเป็นการที่ใช้ในโครงข่ายประสาทเทียม (Neural Networks) โดยเฉพาะในโครงข่ายแบบหลายชั้น เช่น Multilayer Perceptron (MLP) เพื่อปรับค่า "น้ำหนัก" (weights) ของการเชื่อมโยงระหว่างนิวรอนให้เหมาะสม ทำให้โมเดลสามารถพยากรณ์ได้แม่นยำขึ้น โดยกระบวนการนี้ทำงานผ่านสองขั้นตอนหลัก ๆ คือ "การคำนวณค่าความผิดพลาด" (Error Calculation) และ "การปรับปรุงค่า" (Weight Update)

ตารางที่ 7 แสดงถึงพารามิเตอร์ของ MLP regressor โดยพารามิเตอร์ที่สำคัญที่สุดคือ hidden_layer_sizes

ตาราง 1 hyperparameter ของ MLP regressor

พารามิเตอร์	ความหมาย
hidden_layer_sizes	โครงสร้างของ hidden layers เช่น (100,) หรือ (64, 64)
activation	ฟังก์ชันกระตุ้น เช่น 'relu', 'tanh', 'logistic'
solver	วิธีการ optimize เช่น 'adam', 'sgd', 'lbfgs'
alpha	ค่าปรับ L2 regularization
learning_rate	'constant', 'invscaling', 'adaptive'
max_iter	จำนวนรอบการฝึกสูงสุด
early_stopping	หยุดฝึกถ้า validation score ไม่ดีขึ้น
random_state	กำหนด seed เพื่อให้ reproducible

ทฤษฎีที่เกี่ยวข้องกับแบบจำลอง Machine learning แบบ Random Forest

Random Forest เป็นหนึ่งในเทคนิคการเรียนรู้แบบรวม (Ensemble Learning) ที่ได้รับความนิยมสูงมากในงานด้านวิทยาการข้อมูล (Data Science) และการวิเคราะห์เชิงพยากรณ์ (Predictive Analytics) ด้วยจุดเด่นในเรื่องของความแม่นยำ ความยืดหยุ่น (robustness) และการป้องกันการเกิด overfitting ได้ดีกว่าโมเดลต้นไม้การตัดสินใจ (Decision Tree) แบบเดี่ยว ๆ โดยแกนหลักของ Random Forest คือการสร้างต้นไม้การตัดสินใจหลาย ๆ ต้น แล้วให้โหวตผลลัพธ์หรือเฉลี่ยผลการพยากรณ์เพื่อให้ได้ข้อสรุปสุดท้าย

หลักการเรียนรู้แบบรวม (Ensemble Learning) ของ Random forest มีวิธีการทำงานคือมีการสร้าง Decision tree หรือต้นไม้ตัดสินใจ จำนวนมาก โดยการสร้างตามขั้นตอนต่อไปนี้

- สำหรับ Decision tree แต่ละต้น จะมีการทำ Bootstrapping กับชุดข้อมูลฝึก (Training Data) เพื่อสร้างข้อมูลใหม่ (Bootstrapped dataset) จากนั้นจึงนำข้อมูลใหม่ไปสร้าง Decision Tree

- มีการทำ Feature Randomness นั้นคือจากชุดข้อมูลใหม่ (Bootstrapped dataset) เลือกตัวแปรต้นเพียงบางตัวแปรไปใช้ทำ Decision tree

- มีการทำ Bootstrapping และการทำ Feature Randomness ทำเพื่อให้ Decision tree แต่ละต้นเกิดความแตกต่างกัน (Variation)

- เมื่อมี Decision tree จำนวนมากแล้ว เวลาทำนายให้ทำการหาค่าเฉลี่ยสำหรับโมเดลแบบ regression และให้ทำการโหวตสำหรับโมเดลแบบ classification

นอกจากนี้ยังมีรายละเอียดอื่นๆคือ

- สามารถหา Out-of-Bag (OOB) Error ได้

  ในขณะสร้างแต่ละต้นไม้ เราได้สุ่มชุดข้อมูลย่อย (Bootstrap) ขึ้นจากข้อมูลฝึกเดิม ซึ่งจะมีบางตัวอย่างไม่ถูกเลือกไปฝึกในต้นไม้นั้น ๆ โดยตัวอย่างที่ไม่ถูกสุ่มไปในชุดข้อมูลย่อยของต้นไม้ต้นใด ๆ เรียกว่า Out-of-Bag (OOB) samples สามารถนำกลับมาตรวจสอบค่าความผิดพลาดได้ โดยไม่ต้องแบ่งชุดทดสอบ (Test Set) แยกจากภายนอก หากจำนวนต้นไม้มากพอ

สามารถคำนวณความสำคัญของตัวแปร (Feature Importance) ได้ โดย Random Forest สามารถบอกได้ว่าตัวแปร (feature) ใดมีผลอย่างไรต่อการทำนายโมเดลโดยดูได้จาก

- **Mean Decrease in Impurity (MDI)**: วัดว่าการแตกแขนงโดยใช้ feature นั้นช่วยลดความไม่บริสุทธิ์ (impurity) ของโหนดในต้นไม้มากเพียงใด เฉลี่ยทุกต้น

- **Mean Decrease in Accuracy (MDA)**: วัดจากการสลับค่าของ feature นั้น ๆ แบบสุ่ม แล้วดูว่าความแม่นยำของโมเดลลดลงมากน้อยแค่ไหน

Random forest มี hyperparameter หลายอย่าง โดยตาราง 8 แสดงถึงตัวอย่าง hyperparameter นั้น

ตาราง *8* hyperparameter ของ random forest

hyperparameter	Description
n_estimators	จำนวน Decision tree ที่ใช้ โดยส่วนมากถ้าค่ามากจะให้ผลการทำนายที่ดีกว่า แต่ก็จะทำให้เวลาในการฝึกโมเดลนานและขนานโมเดลใหญ่
criterion	ฟังก์ชันที่ใช้วัดคุณภาพของการแบ่ง (split) ในแต่ละโหนด ถ้าเป็นโจทย์ regression มักเป็นMAEหรือ MSE และถ้าเป็นโจทย์ classification มักเป็น gini หรือ entropy
max_depth	ความลึกสูงสุดที่อนุญาตให้ต้นไม้แตกได้ หากตั้งค่านี้ให้ต่ำ ต้นไม้จะตื้นและอาจช่วยป้องกันการ Overfitting

min_samples_split	จำนวนน้อยที่สุดของตัวอย่าง (samples) ในโหนดหนึ่ง ๆ ที่ต้องมีเพื่อให้สามารถแตกโหนดต่อได้
max_features	จำนวน (หรือสัดส่วน) ของตัวแปร (features) ที่จะพิจารณาในการหาจุดแบ่งที่ดีที่สุด (best split) ในแต่ละโหนด

โดยข้อดีของ Random forest คือ

- ความแม่นยำสูง: Random Forest มักจะให้ค่าความแม่นยำสูงกว่า Decision Tree เดี่ยวอย่างมีนัย
- ป้องกัน Overfitting ได้ดี: การเฉลี่ยผลจากหลายต้นไม้ลดความผันผวนและความเสี่ยงต่อการ fitting ข้อมูลมากเกินไป
- ใช้งานได้หลากหลาย: รองรับทั้งงานจำแนก (classification) และงานพยากรณ์ตัวเลข (regression)
- ตีความได้ในระดับหนึ่ง: สามารถวิเคราะห์ความสำคัญของ feature ได้ แม้จะไม่ชัดเจนเท่ากับ Decision Tree เดี่ยว

ส่วนข้อเสียคือ

- ความเร็วในการทำนาย: ถ้ามีจำนวนต้นไม้มหาศาล (เช่น หลักพันหรือล้านต้น) จะใช้เวลาประมวลผลมากขึ้น แม้จะขนานได้บางส่วน
- ขนาดโมเดลใหญ่: การเก็บต้นไม้จำนวนมากอาจใช้หน่วยความจำมากกว่าโมเดลที่เรียบง่าย

ทฤษฎีที่เกี่ยวข้องกับแบบจำลอง Machine learning แบบ GLM

GLM (Generalized Linear Models) คือรูปแบบทางสถิติที่ขยายจาก Linear Regression (การถดถอยเชิงเส้น) เพื่อสามารถจัดการกับตัวแปรตอบสนองที่มีการแจกแจงแตกต่างกันได้ โดยไม่จำเป็นต้องเป็นการแจกแจงแบบปกติ (normal distribution) โดยมีรายละเอียดดังนี้

- ตัวแปรตามจำเป็นต้องอยู่ใน Linear Exponential Family:

  Linear Exponential Family ในบริบทของ Generalized Linear Models (GLMs) หมายถึงกลุ่มของการแจกแจงความน่าจะเป็นที่สามารถเขียนอยู่ในรูปแบบฟังก์ชันเอ็กซ์โปเนนเชียล (Exponential Family) โดยมีพารามิเตอร์เชิงเส้น (linear predictor) เชื่อมโยงกับพารามิเตอร์ของการแจกแจง ฟังก์ชันความหนาแน่น (probability density function) หรือฟังก์ชันมวล (probability mass function) ของ Exponential Family สามารถเขียนได้ตามโครงสร้างทั่วไปดังนี้

$$f(y{:}\,\theta,\emptyset) = a(y,\emptyset)\exp\left(\frac{y\theta - b(\theta)}{\emptyset}\right)$$

  โดย

Y คือ ค่าที่สังเกตได้ (ค่าของตัวแปรสุ่ม)

Θ หรือtheta คือ พารามิเตอร์แบบ Natural Parameter หรือที่เรียกว่าCanonical Parameter

Φ หรือ phi คือ พารามิเตอร์การกระจาย (Dispersion Parameter) กำหนดความแปรปรวนของข้อมูล5

a(Y,ϕ) คือ ฟังก์ชันส าหรับปรับค่าคงที่ให้ผลรวมความน่าจะเป็นให้เป็น 1 (Normalization)

b(θ) คือ ฟังก์ชัน Log-partition หรือที่เรียกอีกชื่อว่า Cumulant Function

- ช่วยให้ฟังก์ชันมีคุณสมบัติของการแจกแจงความน่าจะเป็นสูตรของ GLM

$$\eta = \beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k$$

$$g(\hat{y}) = \eta \text{ หรือ } \hat{y} = g^{-1}(\eta)$$

โดยแต่ละส่วนของ GLM คือ

$\mathbf{g(x)}$ คือฟังก์ชันลิงก์ (Link Function)

$\hat{y}$ คือค่าทำนายตัวแปรตาม

$\boldsymbol{\eta}$ คือตัวทำนายเชิงเส้น

βคือค่าพารามิเตอร์

X คือตัวแปรต้น

- วิธีการหาค่าพารามิเตอร์

ใช้วิธี Maximum Likelihood Estimation โดยใช้ Iteratively Reweighted Least Squares (IRLS) ซึ่งเป็นอัลกอริทึมเชิงตัวเลข (iterative method) ประเภทหนึ่งที่อัปเดตค่า parameter จากการแก้สมการการถดถอยเชิงเส้นแบบถ่วงน้ำหนักซ้ำหลายรอบ อัลกอริทึมจะหยุดเมื่อค่าพารามิเตอร์นั้นไม่เปลี่ยนไปมาก หรือเมื่อค่าวัดความคลาดเคลื่อน (deviance) หรือตัวชี้วัดความเหมาะสมอื่น ๆ คอนเวิร์จสู่จุดต่ำสุด

วิธีดำเนินการวิจัย

ในบทนี้ผู้วิจัยได้สร้างโมเดลการเรียนรู้ของเครื่องกลจากข้อมูล French Motor Claims Datasets จากเว็ป Kaggle โดยเริ่มจากการศึกษาชุดข้อมูล การทำ EDA การสร้างโมเดล การทำ Hyperparameter tuning การทำ feature importance และการเขียนคะแนนตัวชี้วัดต่างๆ ดังนี้

การศึกษาชุดข้อมูล และการทำ EDA

ข้อมูลมีทั้งหมด 12 คอลัมน์แบ่งเป็น 8 คอลัมน์ชนิดตัวเลขและ 4 คอลัมน์ชนิดหมวดหมู่ รายละเอียดข้อมูลเป็นดังนี้

โดยตารางที่ 8 แสดงถึงชนิดข้อมูลแบบตัวเลข (numeric data) และตารางที่ 9 แสดงถึงชุดข้อมูลแบบหมวดหมู่ (Categorical data) โดยคอลัมน์ที่น่าสนใจคือคอลัมน์ตัวแปรตาม หรือ ClaimNb

ตาราง *9* ข้อมูลแบบตัวเลข (numeric data)

ชื่อคอลัมน์	คำอธิบาย
IDpol	เลขกรมธรรม์
ClaimNb	จำนวนเคลมของเลขกรมธรรม์นั้น
Exposure	ระยะเวลาคุ้มครอง
VehPower	แรงม้าของรถยนต์
VehAge	อายุของรถยนต์
DrivAge	อายุของคนขับ
BonusMalus	ประวัติการเคลม
Density	ความหนาแน่นของประชากร (คนต่อตารางกม)

ตาราง 2 ข้อมูลแบบหมวดหมู่ (Categorical data)

ชื่อคอลัมน์	คำอธิบาย
Area	เขตที่อยู่
VehBrand	ยี่ห้อของรถยนต์
VehGas	ชนิดของน้ำมัน (Diesel/Gasoline)
Region	พื้นที่ในประเทศฝรั่งเศส

ข้อมูลแบบตัวเลข

ได้มีการทำ EDA สำหรับข้อมูลแบบตัวเลข โดยโปรแกรม python ได้ข้อมูลค่าเฉลี่ย จำนวน ส่วนเบี่ยงเบนมาตราฐาน และค่าอื่นๆ ดังภาพประกอบ 2 นี้

ภาพประกอบ 2 แสดงให้เห็นถึงว่าจำนวน ค่าเฉลี่ย ส่วนเบี่ยงแบนมาตรฐาน และค่าอื่นๆของแต่ละตัวแปร โดยคอลัมน์ที่สำคัญคือ ClaimNB เนื่องจากคือตัวแปรตาม โดยมีค่าเฉลี่ยเท่ากับ 0.053247

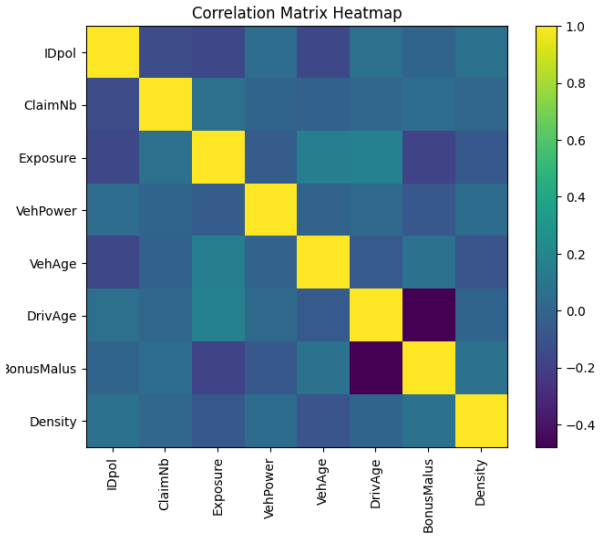
Summary of Numeric Columns:

	IDpol	ClaimNb	Exposure	VehPower
count	6.780130e+05	678013.000000	678013.000000	678013.000000
mean	2.621857e+06	0.053247	0.528750	6.454631
std	1.641783e+06	0.240117	0.364442	2.050906
min	1.000000e+00	0.000000	0.002732	4.000000
25%	1.157951e+06	0.000000	0.180000	5.000000
50%	2.272152e+06	0.000000	0.490000	6.000000
75%	4.046274e+06	0.000000	0.990000	7.000000
max	6.114330e+06	16.000000	2.010000	15.000000

	VehAge	DrivAge	BonusMalus	Density
count	678013.000000	678013.000000	678013.000000	678013.000000
mean	7.044265	45.499122	59.761502	1792.422405
std	5.666232	14.137444	15.636658	3958.646564
min	0.000000	18.000000	50.000000	1.000000
25%	2.000000	34.000000	50.000000	92.000000
50%	6.000000	44.000000	50.000000	393.000000
75%	11.000000	55.000000	64.000000	1658.000000
max	100.000000	100.000000	230.000000	27000.000000

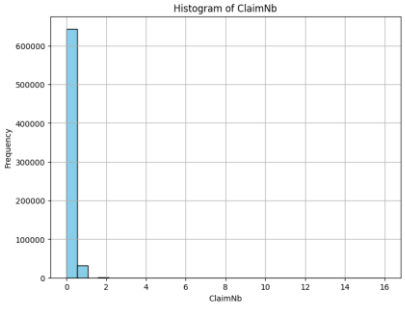
ภาพประกอบ 2 สรุปคอลัมน์ตัวเลข

ภาพประกอบ 3 เป็น Correlation matrix heatmap ที่สร้างออกมาเพื่อดูความสัมพันธ์ระหว่างตัวแปร โดยสีเหลืองแสดงถึงการสัมพันธ์กันอย่างสัมบูรณ์แบบ



ภาพประกอบ 3 Correlation matrix heatmap ของคอลัมน์ตัวเลข

นอกจากนี้ยังได้สร้าง Histogram ของตัวแปรตาม (ClaimNB) ดังภาพประกอบที่ 4 ดังนี้ โดยเคลมส่วนใหญ่มีค่าเท่ากับ 0

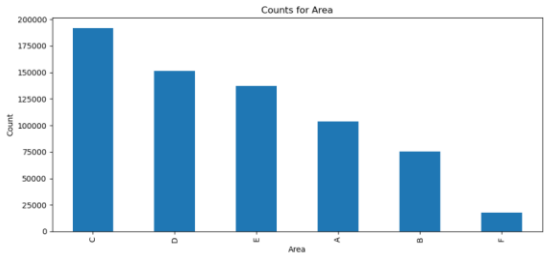


ภาพประกอบ 4 Histogram ของ ClaimNB

ข้อมูลแบบหมวดหมู่

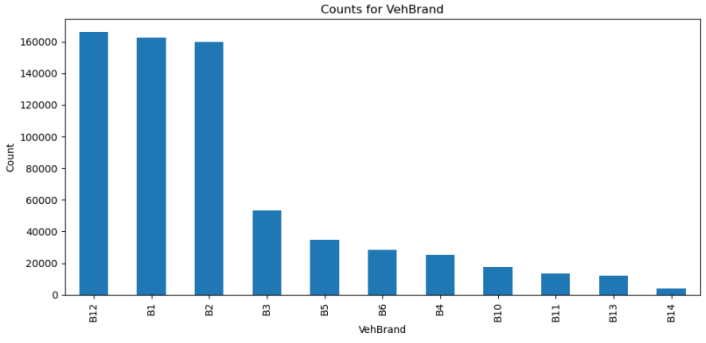
ได้มีการทำ EDA สำหรับข้อมูลแบบหมวดหมู่ โดยโปรแกรม python และนำความถี่มาสร้างเป็นกราฟ ดังนี้

ในภาพประกอบ 5 เป็นกระจายตัวของตัวแปร Area ค่าที่มีความถี่เยอะที่สุดคือ AREA C โดยค่าเหล่านี้เป็นรหัสที่ไม่ได้แสดงถึงพื้นที่ที่แท้จริง



ภาพประกอบ 5 การกระจายตัวของ column AREA

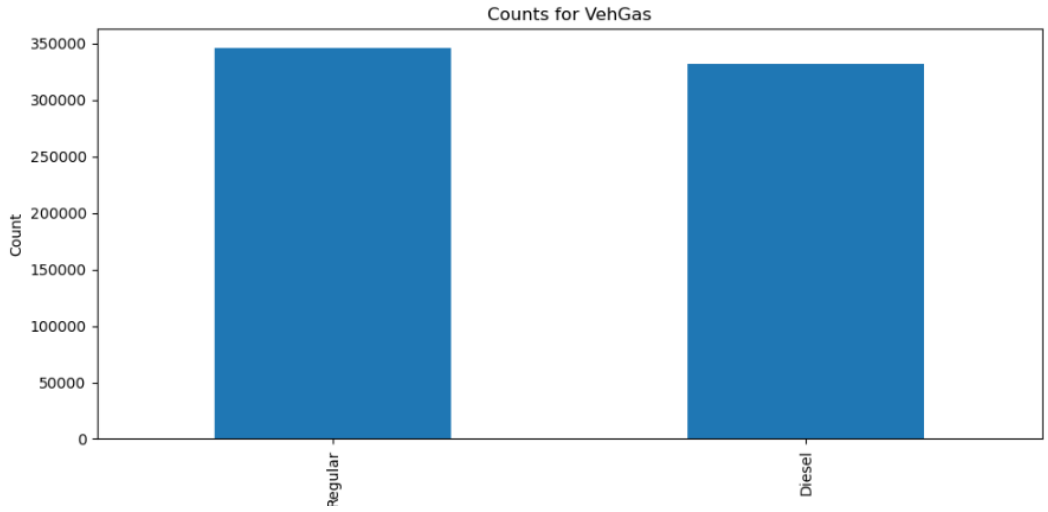
ในภาพประกอบ 6 เป็นกระจายตัวของตัวแปร VehBrand ค่าที่มีความถี่เยอะที่สุดคือ B12 และตามด้วย B1 และ B2 โดยค่าเหล่านี้เป็นรหัสที่ไม่ได้แสดงถึงยี่ห้อที่แท้จริง



ภาพประกอบ 6 การกระจายตัวของ column VehBrand

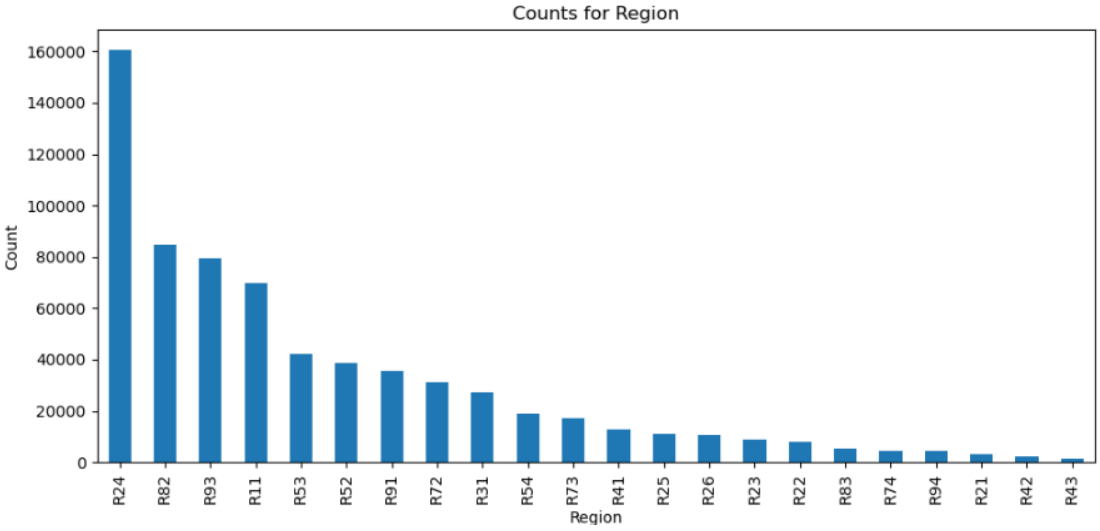
ในภาพประกอบ 7 เป็นกระจายตัวของตัวแปร Vehgas โดยแสดงให้เห็นว่ารถที่ทำประกันมีจำนวนเท่าๆกันระหว่างรถที่ใช้น้ำมันดีเซล

และรถที่ใช้น้ำมันเบนซิน มีจำนวนอยู่ที่ประมาณ325000 คันสำหรับน้ำมันแต่ละแบบ



ภาพประกอบ 7 การกระจายตัวของ column Vehgas

ในภาพประกอบ 8 เป็นกระจายตัวของตัวแปร Region ค่าที่มีความถี่เยอะที่สุดคือ R24 โดยตัวแปรเหล่านี้เป็นรหัสที่ไม่ได้แสดงถึงค่าที่แท้จริง



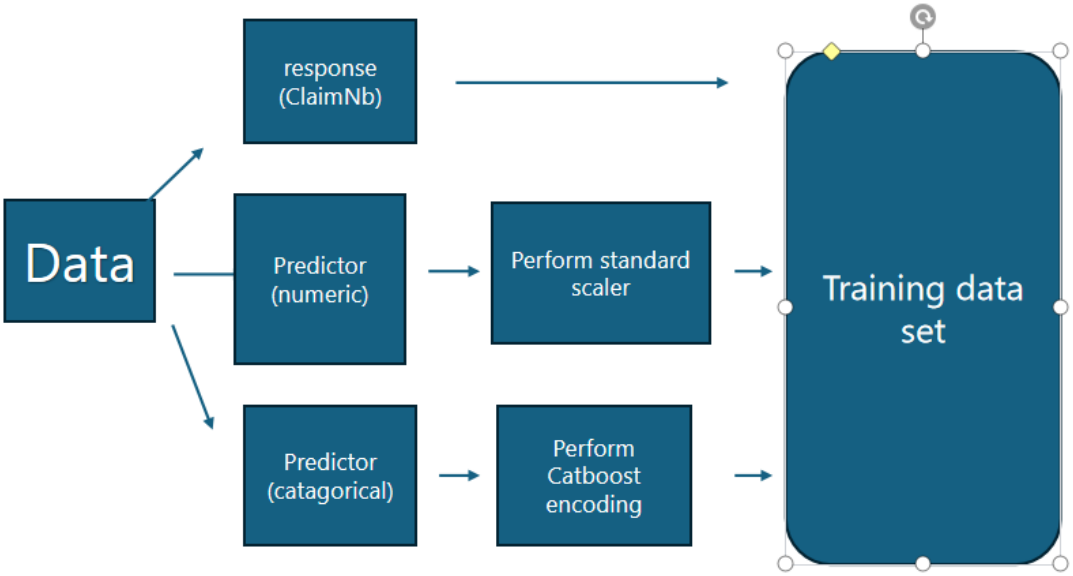
ภาพประกอบ 8 การกระจายตัวของ column Region

การทำData preprocessing การทำ Test-train-validation split และการสร้างโมเดล

ได้มีการทำการจัดเตรียมข้อมูลและการสร้างโมเดล โดยมีรายละเอียดดังข้างล่าง

Data preprocessing

ภาพประกอบ 9 เป็นกระบวนการทำ data preprocessing โดยรายละเอียดตามที่เขียนด้านล่าง



ภาพประกอบ 9 Data preprocessing

การทำ preprocessing ประกอบด้วยการคัดแยกตัวแปรต้นที่เป็นตัวเลขและตัวแปรต้นที่เป็นหมวดหมู่ โดยสำหรับตัวแปรต้นที่เป็นตัวเลข จะมีการทำ standard scaler คอลัมน์ และสำหรับตัวแปรที่เป็นหมวดหมู่ จะทำ catboost encoding

- การทำ Standard scaler สำหรับคอลัมน์ที่เป็นตัวเลข สำหรับทุกคอลัมน์ตัวแปรต้นที่เป็นตัวเลข จะมีการหาค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานของแต่ละคอลัมน์ แล้วนำค่าในคอลัมน์เดิมมาลบค่าเฉลี่ยละหารส่วนเบี่ยงเบนมาตรฐาน

ภาพประกอบ 10 แสดงให้เห็นวิธีการทำ Standard Scaler

X1	X2
0.1	10
0.4	5

→

X1_standardize	X2_standardize
$\frac{0.1-\mu_{x1}}{S.D(x1)}$	$\frac{10-\mu_{x2}}{S.D(x2)}$
$\frac{0.4-\mu_{x1}}{S.D(x1)}$	$\frac{5-\mu_{x2}}{S.D(x2)}$

ภาพประกอบ 10 Standard Scaler

โดยสูตรของค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานเป็นดังภาพประกอบ 11

$$\mu_x = \frac{\sum X}{n}$$

$$S.D(x) = \sqrt{\frac{\sum (X_i - \mu_x)^2}{n}}$$

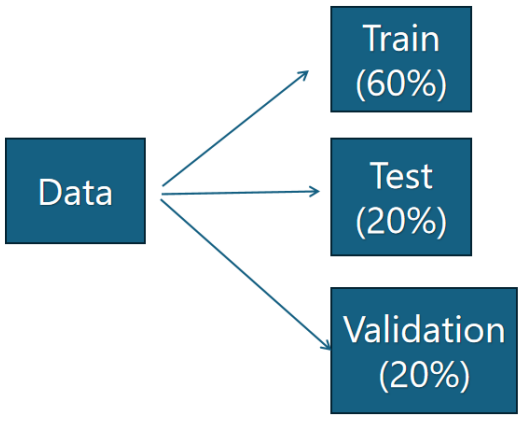
ภาพประกอบ 11 สูตรค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐาน

- CatBoost encoding เป็นเทคนิคเฉพาะที่ใช้จัดการกับข้อมูลประเภทหมวดหมู่ (categorical variables) โดยไม่ต้องแปลงเป็น one-hot encoding หรือ label encoding แบบทั่วไป ซึ่งมักทำให้ข้อมูลมีมิติสูงเกินจำเป็น โดย CatBoost ใช้วิธีที่เรียกว่า Ordered Target Statistics ซึ่งจะคำนวณสถิติเช่นค่าเฉลี่ยของ target ภายใต้แต่ละหมวดหมู่ แต่จะใช้เฉพาะข้อมูลจากลำดับก่อนหน้าเท่านั้นในแต่ละรอบของการฝึกโมเดล เพื่อป้องกันการรั่วไหลของข้อมูลเป้าหมาย (target leakage) ที่อาจทำให้โมเดล overfit

  ข้อดีของการ encoding แบบนี้คือสามารถจัดการกับข้อมูลหมวดหมู่ที่มีค่าซ้ำจำนวนมากหรือค่าที่ไม่เคยเห็นมาก่อนได้อย่างมีประสิทธิภาพ อีกทั้งยังช่วยลดมิติของข้อมูลโดยไม่สูญเสียข้อมูลสำคัญ ทำให้การฝึกโมเดลเร็วขึ้นและแม่นยำมากขึ้น โดยผู้ใช้เพียงแค่ระบุชื่อ column ที่เป็นหมวดหมู่ผ่านพารามิเตอร์ cat_features ใน CatBoost เท่านั้น ส่วนการแปลงค่าจะถูกจัดการโดยอัตโนมัติในเบื้องหลังของโมเดล.

Test-train-validation split

มีการแบ่ง test-train-validation split ดังภาพประกอบ 12 โดยมีคำอธิบายอยู่ด้านล่าง



ภาพประกอบ 12 การทำ test-train-validation split

แบ่งชุดข้อมูลออกเป็นสองส่วน โดยเป็น Train 60% Test 20% และ Validation 20% โดยแบ่งทั้งหมด 5 ครั้ง แต่ละครั้งใช้ seed ต่างกันเพื่อให้ได้ผลต่างกัน

การสร้างโมเดล

โมเดลที่สร้างจะมีทั้งหมด 8 โมเดล นั้นคือ

Model แบบ boosting

- XGboost regressor
- LightGBM regressor
- CatBoost regressor

Model แบบ GLM

- Poisson regressor
- Negative binomial regressor

Model แบบ Neural Network

- Tabnet regressor
- MLP regressor

Model แบบ Ensemble

- Random forest

โดยใน XGboost LightGBM Catboost และ Random forest ได้มีการใช้การ์ดจอ Nvidia เร่งความเร็วในการเทรนโมเดล และ ในทุกโมเดล มีการทำ Hyperparameter tuning โดยใช้ Grid search เพื่อค้นหา hyperparameter ที่เหมาสม

ขั้นตอนการสร้างโมเดลมีดังนี้

- มีการ split test-train-validation data 5 ครั้ง โดยแต่ละครั้งใช้ seed ต่างกันเพื่อให้ได้ผลต่างกัน
- ในแต่ละ seed นำ Train data มาทำ hyperparameter tuning กับ validation data เพื่อให้ได้โมเดลที่ดีที่สุด
- จากนั้นนำโมเดลที่ดีที่สุดพร้อมกับ hyperparameter ที่ดีที่สุดไปคำนวณค่า R squared, MAE และ MSE กับ test data ของ seed นั้นๆ
- สุดท้ายหาค่าเฉลี่ยของ R squared, MAE และ MSE ของทั้ง 5 ครั้งนั้น เป็นผลลัพธ์สุดท้าย
- นอกจากนี้ในโมเดล XGboost Catboost LightGBM Tabnet และ MLP regressor มีการใช้ Early stopping โดย MAE metric เพื่อประหยัดเวลาและทำให้โมเดลไม่เกิดการ Overfit

การทำ Feature importance

นอกจากการสร้างโมเดลแล้ว ได้มีการทำ feature importance ด้วย 4 โมเดล นั้นคือ

- Lightgbm
- Catboost
- Xgboost
- Random forest

โดยสำหรับ Lightgbm Catboost Xgboost และ random forest ได้ใช้คำสั่ง Feature importance ที่ build in ใน โมเดล

ผลการวิจัย

ในงานวิจัยนี้ ได้มีการนำการเรียนรู้ของเครื่องมาเปรียบเทียบประสิทธิภาพการทำนายจำนวนเคลม โดยมีทั้งหมด 7 โมเดล ได้แก่

Model แบบ boosting

- XGboost regressor
- LightGBM regressor
- CatBoost regressor

Model แบบ GLM

- Poisson regressor
- Negative binomial regressor

Model แบบ Neural network

- Tabnet regressor
- MLP regressor

Model แบบ Ensemble

- Random forest

ได้ผลลัพธ์จากการวิจัยดังนี้

ผลจากค่า Hyperparameter tuning และคะแนนตัวชี้วัดต่างๆ

ในทุกโมเดลที่ทำ hyperparameter tuning โดยได้รัน 5 ครั้ง โดยแต่ละครั้งใช้ Seed ต่างกัน สุดท้ายมีการนำค่าคะแนน MAE MSE และ R squared มาเฉลี่ยกัน

ตาราง 9 เป็น Hyperparameterที่จูนได้ของ XGboost ในแต่ละ Seed โดย Hyperparameter ที่ได้ในแต่ละ Seed ไม่เท่ากัน

ตาราง *10* hyperparameter ที่ใช้ของ XGboost

Seed	Hyperparameter
0	Max_depth:6,Learning_rate:0.5, Early_stopping_round:89
1	Max_depth:8, Learning_rate:0.3 , Early_stopping_round:46
2	Max_depth:6, Learning_rate:0.5 , Early_stopping_round:37
3	Max_depth:8, Learning_rate:0.3 , Early_stopping_round:89
4	Max_depth:8, Learning_rate:0.3 , Early_stopping_round:38

ตารางที่ 19 เป็น Hyperparameterที่จูนได้ของ Catboost ในแต่ละ Seed โดย Hyperparameter ที่ได้ในแต่ละ Seed ไม่เท่ากัน เช่นกัน

ตาราง *11* hyperparameter ที่ใช้ของ Catboost

Seed	Hyperparameter
0	Max_depth:6, Learning_rate:0.5, Early_stopping_round:43
1	Max_depth:8, Learning_rate:0.5 , Early_stopping_round:32
2	Max_depth:4, Learning_rate:0.4 , Early_stopping_round:199
3	Max_depth:8, Learning_rate:0.2 , Early_stopping_round:218
4	Max_depth:6, Learning_rate:0.4

	, Early_stopping_round:28

ตารางที่ 20 เป็น Hyperparameter ที่จูนได้ของ LightGBM ในแต่ละ Seed โดย Hyperparameter ที่ได้ในแต่ละ Seed ไม่เท่ากัน เช่นกัน

ตาราง *12* hyperparameter ที่ใช้ของ LightGBM

Seed	Hyperparameter
0	Max_depth:6, Learning_rate:0.5, Early_stopping_round:87
1	Max_depth:6, Learning_rate:0.2 , Early_stopping_round:209
2	Max_depth:6, Learning_rate:0.4 , Early_stopping_round:73
3	Max_depth:6, Learning_rate:0.4 , Early_stopping_round:38
4	Max_depth:4, Learning_rate:0.5 , Early_stopping_round:23

ตารางที่ 21 เป็น Hyperparameter ที่จูนได้ของ Poisson regressor ในแต่ละ Seed โดยมีค่าเท่ากันหมดคือ 0

ตาราง *13* hyperparameter ที่ใช้ของ Poisson regressor

Seed	Hyperparameter
0	Alpha: 0
1	Alpha: 0
2	Alpha: 0
3	Alpha: 0
4	Alpha: 0

ส่วนตารางที่ 22 เป็น Hyperparameter ที่จูนได้ของ Negative Binomial regressor ในแต่ละ Seed โดนมีค่า alpha เท่ากับ 0.01 ใน 4 Seed

ตาราง *14* hyperparameter ที่ใช้ของ Negative Binomial regressor

Seed	Hyperparameter
0	Alpha: 0.01
1	Alpha: 0.01
2	Alpha: 0.01
3	Alpha: 0.000001
4	Alpha: 0.01

ตารางที่ 23 เป็น Hyperparameter ที่จูนได้ของ MLP regressor ในแต่ละ Seed โดยทีมีค่า hidden_layer_sizes เท่ากับ (300,300) ใน 3 seed

ตาราง *14* hyperparameter ที่ใช้ของ MLP regressor

Seed	Hyperparameter
0	hidden_layer_sizes: (300,300)
1	hidden_layer_sizes: (300,300)
2	hidden_layer_sizes: (300,300)
3	hidden_layer_sizes: (100,100)
4	hidden_layer_sizes: (200,)

ตารางที่ 24 เป็น Hyperparameter ที่จูนได้ของ Tabnet regressor ในแต่ละ Seed โดยมีค่าไม่เท่ากับเลยในทุก seed

ตาราง *16* hyperparameter ที่ใช้ของ Tabnet regressor

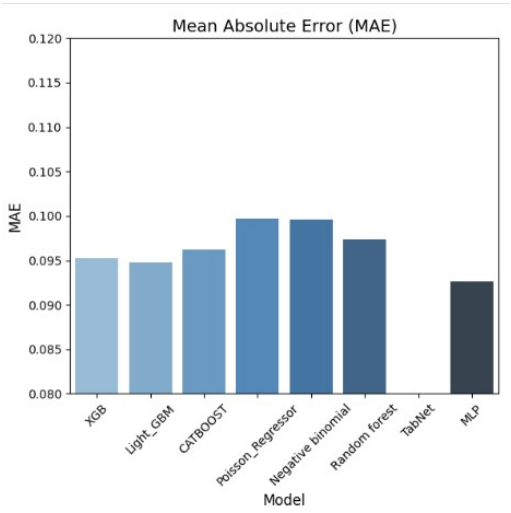
Seed	Hyperparameter
0	N_d: 64, N_a: 64
1	N_d: 32, N_a: 16
2	N_d: 32, N_a: 32
3	N_d: 64, N_a: 16
4	N_d: 32, N_a: 64

ได้มีการคำนวณค่าเฉลี่ยของ MAE MSE และ R-squared จากทั้ง 5 seed โดยหากค่าทำนายติดลบ ได้มีการดึงค่าทำนายให้กลับมาเป็น 0 ได้ผลลัพธ์ดังตารางที่ 25 โดยถ้าดูจาก MAE MSE และ R_squared โมเดลที่ดีที่สุดคือ Tabnet Catboost และ Catboost ตามลำดับ และโมเดล Tabnet มี MAE ต่ำแต่ MSE สูง ซึ่งค้านกับโมเดลอื่นๆ จะมีการอภิปรายปรากฏการณ์นี้ในบทที่ 5

ตาราง *17* สรุปคะแนนแต่ละโมเดล

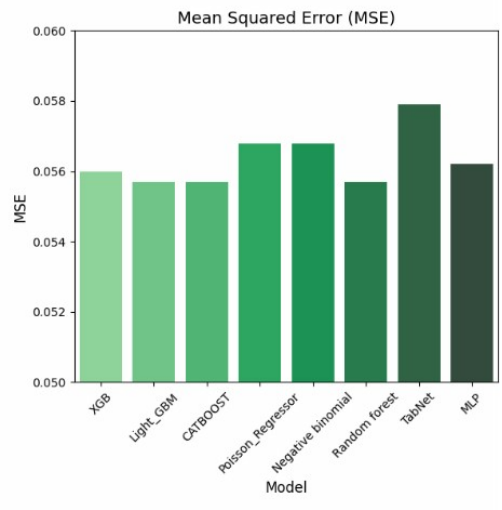
	MAE	MSE	R_squared
XGB	0.0952	0.0560	0.0288
Light_GBM	0.0948	0.0557	0.0336
CATBOOST	0.0962	0.0557	0.0339
Poisson_Regressor	0.0997	0.0568	0.0146
Negative binomial	0.0996	0.0568	0.0149
Random forest	0.0974	0.0557	0.0331
TabNet	0.0787	0.0579	-0.0047
MLP	0.0926	0.0562	0.0248

จากนั้นมีการนำค่า MAE มาแสดงเป็น Bar chart ดังภาพประกอบ 13 และจะเห็นได้ว่าโมเดลที่คะแนนดีคือโมเดล Tabnet และตามด้วยโมเดลจำพวก boosting



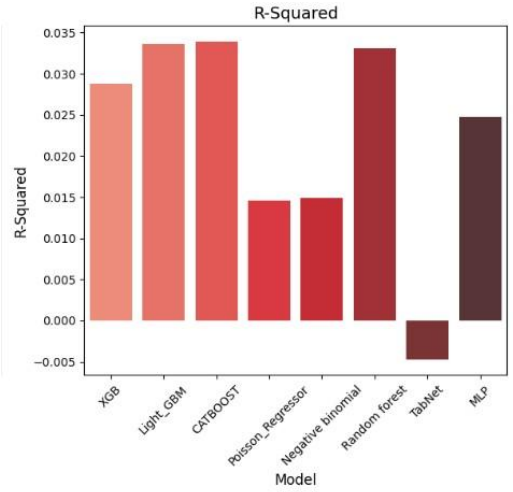
ภาพประกอบ 13 ค่า MAE แต่ละโมเดล

เช่นเดียวกับ MAE จากนั้นมีการนำค่า MSE มาแสดงเป็น Bar chart ดังภาพประกอบ 14 และจะเห็นได้ว่าโมเดลที่คะแนนดีคือโมเดลจำพวก boosting โดยเฉพาะ Catboost Regressor และ LightGBM



ภาพประกอบ 14 ค่า MSE แต่ละโมเดล

สุดท้ายจะเป็น Bar chart ของ R_squared ดังภาพประกอบ 15 โมเดลที่ดีที่สุดคือ Catboost และ LightGBM และโมเดลที่แย่ที่สุดคือ Tabnet ที่ให้ค่า R_squared ติดลบเล็กน้อย



ภาพประกอบ 15 ค่า R-squared แต่ละโมเดล

ผลลัพธ์การวิเคราะห์ฟีเจอร์ด้วย Feature importance

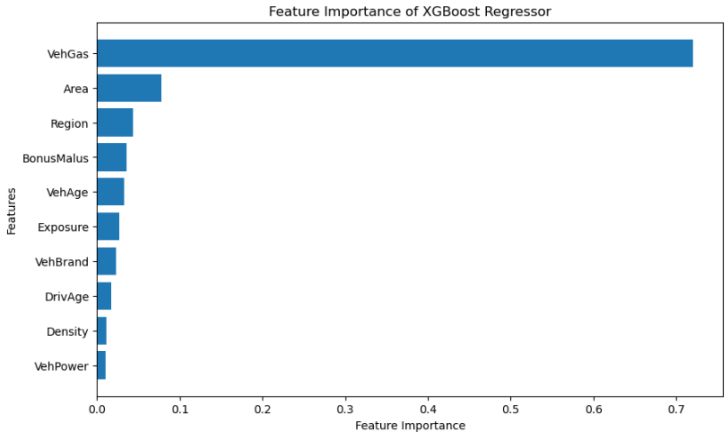
ได้มีการทำ Feature importance ด้วยโมเดล 5 โมเดล นั้นคือ

- Xgboost feature importance
- Lightgbm feature importance
- Catboost feature importance

- Random Forest feature importance

XGboost regressor

ภาพประกอบ 16 คือผลลัพธ์จากทำ Feature importance ด้วย XGboost regressor

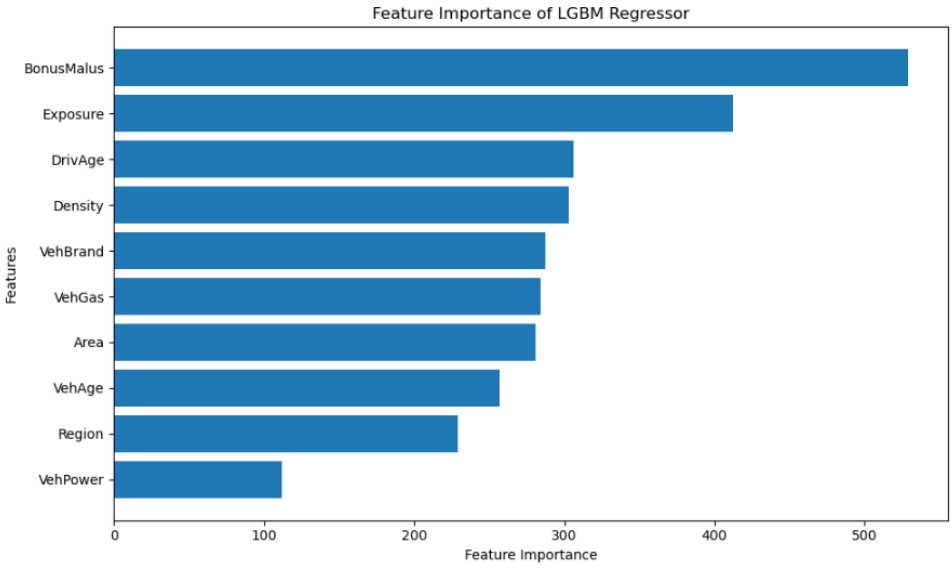


ภาพประกอบ 16 คะแนน feature importance ด้วย XGboost regressor

ค่าตัวแปรต้นที่ทำนายจำนวนการเคลมได้ดีที่สุดคือประวัติการเคลมในอดีต (VehGas) ต่อด้วยระยะเวลาที่กรมธรรม์ส่งผล (Area) และอายุรถ (Region)

Lightgbm regressor

ภาพประกอบ 17 คือผลลัพธ์จากทำ Feature importance ด้วย Lightgbm regressor

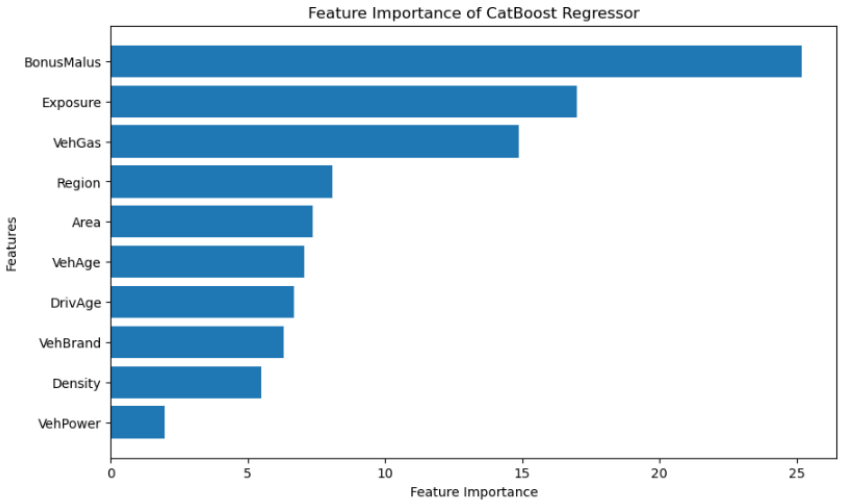


ภาพประกอบ 17 คะแนน feature importance ด้วย XGboost regressor

โดยประวัติการเคลม (BonusMalus) และ ระยะเวลาของกรมธรรม์ (Exposure)ส่งผลต่อตัวแปรตามเยอะที่สุด

Catboost regressor

ภาพประกอบ 18 คือคะแนนการทำ Feature importance ด้วย Catboost regressor

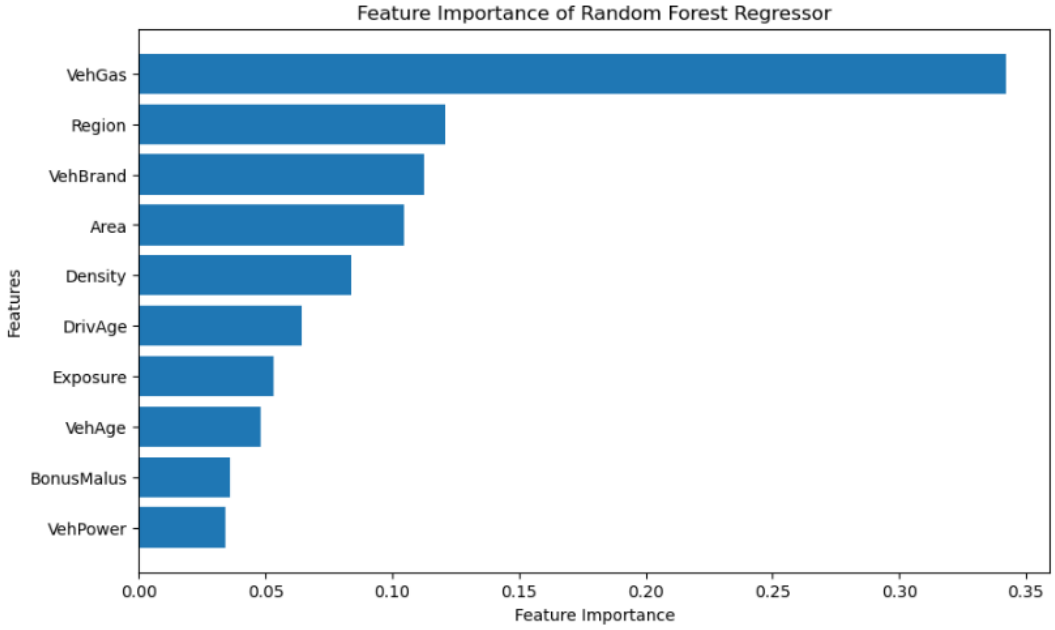


ภาพประกอบ 18 คะแนน feature importance ด้วย Catboost regressor

คะแนน Feature importance ของ Catboost ใกล้เคียงกับของ LightGBM โดยตัวแปรที่คะแนนเยอะที่สุดคือ ประวัติการเคลม (BonusMalus) และ ระยะเวลาของกรมธรรม์ (Exposure)

Random forest regressor

ภาพประกอบที่ 19 คือผลของ feature importance ของ Random forest โดยตัวแปรที่ส่งผลกับตัวแปรตามมากที่สุดคือชนิดน้ำมันที่ใช้

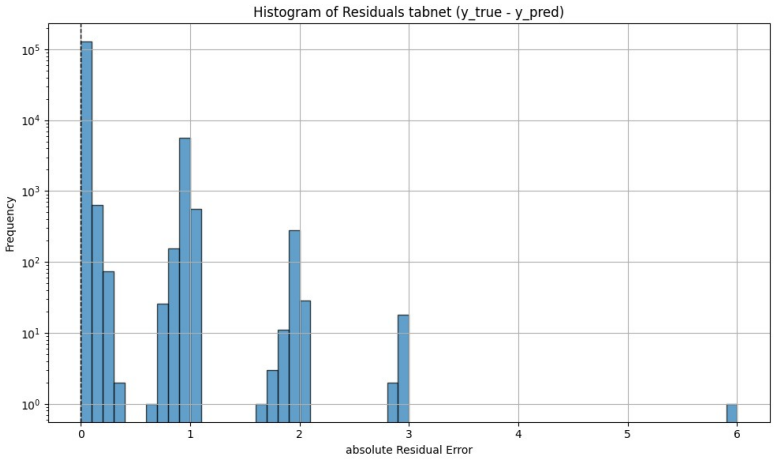


ภาพประกอบ 19 คะแนน feature importance ด้วย Random Forest

สรุปผลการวิจัย อภิปรายผล และข้อเสนอแนะ

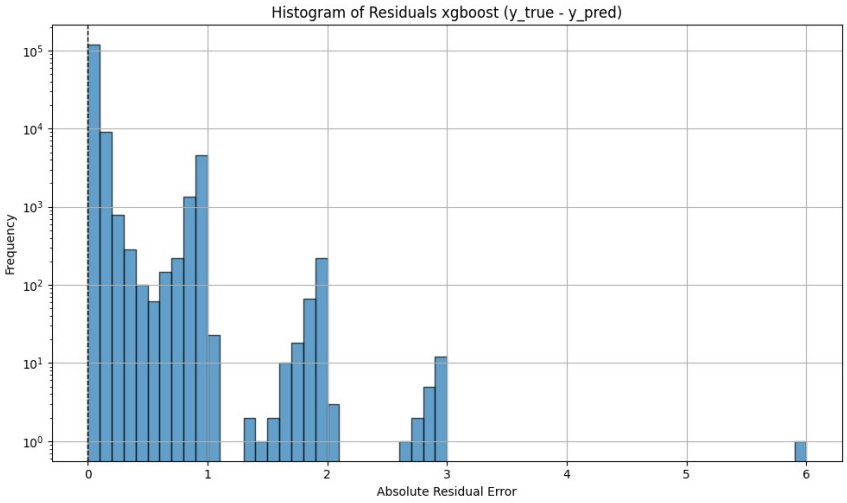
อภิปรายเหตุผลที่โมเดล Tabnet มีค่าMAE ต่ำ และ MSE สูง เมื่อเปรียบเทียบกับโมเดลอื่นๆ

ภาพประกอบ 20 เป็นภาพค่า absolute ของ error ของโมเดล tabnet ที่ใช้ logscale โดยจะเป็นว่าที่ค่า actual เท่ากับ 0 ค่า error กระจุกกันอยู่รอบๆ 0 และสำหรับค่า actual ที่มากกว่า 0 ค่า error จะกระจุกอยู่ที่ 1 2 และ 3 ตามลำดับ



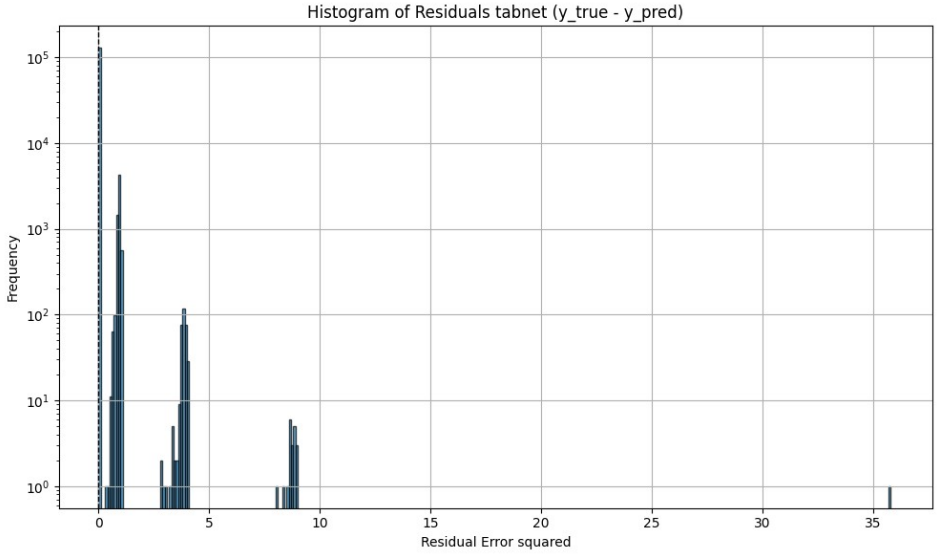
ภาพประกอบ 20 error ของโมเดล tabnet

ภาพประกอบ 21 เป็นภาพของค่า absolute ของ error ของโมเดล XGboost ที่ใช้ logscale โดยจะได้ว่าค่า error ของค่าเคลมแต่ละค่ามีการกระจายตัวมากกว่า



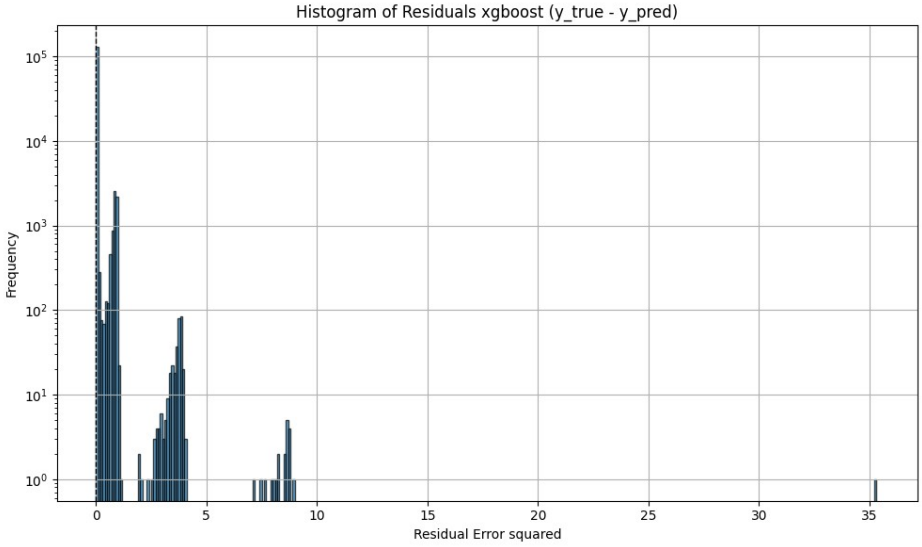
ภาพประกอบ 21 absolute error ของโมเดล XGboost

นอกจากนี้ ในภาพประกอบ 22 เป็นภาพความถี่ของค่า error ยกกำลังสองของโมเดล tabnet โดยค่า error ยกกำลังสองกระจุกกันอยู่ 3 กระจุก



ภาพประกอบ 22 ค่า error ยกกำลังสองของโมเดล tabnet

ในภาพประกอบ 23 เป็นภาพความถี่ของค่า error ยกกำลังสองของโมเดล xgboost โดยค่า error ยกกำลังสองกระจุกกันอยู่ 3 กระจุก เช่นกัน แต่ค่าในแต่ละกระจุกกระจายมากกว่าโมเดล tabnet



ภาพประกอบ 23 ค่า error ยกกำลังสองของโมเดล xgboost

หากจะยกตัวอย่างให้เข้าใจง่าย หากจะเปรียบเทียบอย่างง่าย ยกตัวอย่างให้

- โมเดล 1 เปรียบได้กับ โมเดล Tabnet มี error 5 ตัว (เป็นตัวอย่างคะแนนสมมุติ เพื่อให้เห็นภาพเท่านั้น) โดยทั้ง 5 ตัวคือ1,1,1,1 และ 6
- โมเดล 2 เปรียบได้กับ โมเดล xgboost มี error 5 ตัวเช่นกัน (เป็นตัวอย่างคะแนนสมมุติ เพื่อให้เห็นภาพเท่านั้น) โดยค่าคือ 2.1,2.1,2.1,2.1 และ 2.1

แม้ว่าทั้ง 2 โมเดลดูจะมี error พอๆกัน แต่ถ้าคำนวณ MAE และ MSE แล้ว จะได้ว่า

ตาราง *18* ตัวอย่างคะแนนสมมุติ

	MAE	MSE
โมเดล 1	2	8
โมเดล 2	2.1	4.41

โดยจะเห็นว่าในคะแนนสมมุติ tabnet ค่า MAE ต่ำกว่า แต่ค่า MSE สูงกว่าโมเดล xgboost มาก นี้เปรียบได้ดังสถานการณ์ที่เกิดขึ้นจริงของโมเดล Tabnet

อภิปรายค่า R squared

ในข้อมูลชุดนี้ ค่า R squared ไม่สูงนัก โดยมีค่า R squared ตั้งแต่ -0.0047 ถึง 0.0339 ทำให้เห็นได้ว่าข้อมูลชุดนี้ทำนายได้ยาก แต่ค่าทำนายในทุกโมเดลยกเว้น tabnet ให้ผลดีกว่าการใช้ค่าเฉลี่ยทำนายเล็กน้อย เนื่องจากค่า R squared ยังมีค่าเป็นบวกอยู่

อภิปรายผลการวิจัยอื่นๆ

การเรียนรู้ของเครื่องแบบ boosting เป็นการเรียนรู้ของเครื่องกลที่ซับซ้อนกว่าการเรียนรู้แบบ GLMและในชุดข้อมูลที่ซับซ้อน หลายครั้งการเรียนรู้แบบ boosting จะใช้งานได้ดีการกว่าเรียนรู้แบบGLM หรือ การเรียนรู้แบบ Neural network ผลการวิจัยในงานวิจัยนี้เป็นข้อบ่งชี้ให้เห็นได้อย่างชัดเจนถึงการทำนายนั้นที่แม่นกว่านั้น ทั้งนี้การเรียนรู้ของเครื่องแบบ boosting จะใช้พลังงานคอมพิวเตอร์ (Computational power) มากกว่า ซึ่งเป็นสิ่งที่ต้องจ่ายเพื่อให้ได้ค่านำนายที่ดีกว่า

ข้อเสนอแนะ

จากผลของงานวิจัยนี้ ทางผู้จัดทำได้แสดงให้เห็นว่าการเรียนรู้ของเครื่องแบบ boosting สามารถให้ค่าทำนายที่ดีกว่าการเรียนรู้ของเครื่องกลแบบเชิงเส้น จึงมีข้อเสนอแนะให้มีการปรับใช้การเรียนรู้ของเครื่องแบบ boosting กับวงการประกันภัยมากขึ้น โดยให้คำนึงถึงการนำไปใช้ต่อไปนี้

- การเรียนรู้ของเครื่องกลแบบต้นไม้สามารถใช้ได้ดีกับชุดข้อมูลที่มีขนาดใหญ่ และสามารถใช้การเร่งความเร็วด้วย GPU (GPU acceleration) เพื่อทำให้การเทรนเร็วขึ้นได้ โดยโมเดลหลายโมเดลสามารถเทรนด้วย GPU ได้ ไม่ว่าจะเป็น Catboost, XGBoost LightGBM หรือ Random forest ที่ใช้ cuml ในคอมพิวเตอร์แบบลินุกซ์

- หากทรัพยากรไม่พียงพอที่จะเทรนโมเดล สามารถใช้บริการ Cloud computing ได้ โดยจ่ายเป็นรายชั่วโมง

- หากต้องการศึกษาสถิติแบบอนุมาน (inference statistic) อาจเลือกใช้โมเดลการเรียนรู้ของเครื่องกลแบบ GLM แทน เนื่องจากมีค่า Beta ที่บอกความสัมพันธ์ระหว่างตัวแปรได้

เอกสารอ้างอิง

[1]Banachewicz, K. (2022). *The Kaggle Book: Data Analysis and Machine Learning for Competitive Data Science*. Packt Publishing. https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=3269830

[2]Clemente, C., Guerreiro, G. R., & Bravo, J. M. (2023). Modelling Motor Insurance Claim Frequency and Severity Using Gradient Boosting. *Risks*, *11*(9).

[3] Garrido, J., Genest, C., & Schulz, J. (2016). Generalized linear models for dependent frequency and severity of insurance claims. *Insurance: Mathematics and Economics*, *70*, 205-215. https://doi.org/10.1016/j.insmatheco.2016.06.006

[4]Poufinas, T., Gogas, P., Papadimitriou, T., & Zaganidis, E. (2023). Machine Learning in Forecasting Motor Insurance Claims. *Risks*, *11*(9).

[5] Sugiyama, M. (2016). *Introduction to Statistical Machine Learning*. Morgan Kaufmann. https://www.sciencedirect.com/science/book/9780128021217

[6] ปวริศา, ส. (2561). ตัวแบบการถดถอยที่มีผลกระทบจากค่าศูนย์ประยุกต์ใช้กับจำนวนครั้งของการเรียกร้องค่าสินไหมทดแทนในประกันภัยรถยนต์ภาคสมัครใจ. *วารสารมหาวิทยาลัยศรีนครินทรวิโรฒ (สาขาวิทยาศาสตร์และเทคโนโลยี)*, *10*(20), 57-75. http://ejournals.swu.ac.th/index.php/SWUJournal/article/view/10868/9274