การทำนายราคาที่พักในกรุงเทพโดยประยุกต์ใช้การเรียนรู้ของเครื่อง

PREDICTING THE ACCOMMODATION PRICE IN BANGKOK BY APPLYING MACHINE LEARNING

ชยพัทธ์ ณัฐศิววงศ์

CHAYAPAT NATSHIVAWONG

การค้นคว้าอิสระนี้เป็นส่วนหนึ่งของการการศึกษาตามหลักสูตร

ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการข้อมูลและการวิเคราะห์

ศูนย์วิเคราะห์ข้อมูลดิจิทัลอัจฉริยะพระจอมเกล้าลาดกระบัง คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าคุณทหารลาดกระบัง

2567

KMITL-2024-SC-M-017-008

PREDICTING THE ACCOMMODATION PRICE IN BANGKOK BY APPLYING MACHINE LEARNING

CHAYAPAT NATSHIVAWONG

AN INDEPENDENT STUDY SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENT FOR THE DEGREE OF SCIENCE IN DATA SCIENCE AND ANALYTICS
KMITL-DIGITAL ANALYTICS AND INTELLIGENCE CENTER SCHOOL OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2024
KMITL-2024-SC-M-017-008

หัวข้อการค้นคว้าอิสระ การทำนายราคาที่พักในกรุงเทพโดยประยุกต์ใช้การเรียนรู้ของเครื่อง
ชื่อนักศึกษา นายชยพัทธ์ ณัฐศิววงศ์
รหัสประจำตัว 65056021
ปริญญา วิทยาศาสตรมหาบัณฑิต (วิทยาการข้อมูลและการวิเคราะห์)
ศูนย์วิเคราะห์ข้อมูลดิจิทัลอัจฉริยะพระจอมเกล้าลาดกระบัง
พ.ศ. 2567
อาจารย์ที่ปรึกษาค้นคว้าอิสระ ผู้ช่วยศาตราจารย์.ดร.ยุวดี กล่อมวิเศษ

บทคัดย่อ

ในปัจจุบัน การท่องเที่ยวได้กลายมาเป็นอุตสาหกรรมที่สร้างรายได้อย่างมหาศาลให้แก่ประเทศ โดยแพลตฟอร์ม Airbnb เป็น 1 ในแพลตฟอร์มสำหรับการนำที่พักมาแชร์เพื่อให้นักท่องเที่ยวเข้ามาพักอาศัยในขณะมาเที่ยวประเทศไทย ผู้วิจัยจึงมีความสนใจในการศึกษาการทำนายราคาที่พักในประเทศไทย เฉพาะจังหวัดกรุงเทพมหานคร โดยใช้ข้อมูลจากเว็บไซต์ insideairbnb ซึ่งข้อมูลที่ผ่านกระบวนการคลีนแล้ว จะมีทั้งหมด 29 คอลัมน์ และ 7924 แถว จากนั้นนำข้อมูลมาสร้างตัวแบบเอ็กซ์จีบูส, ซัพพอร์ตเวกเตอร์แมชชีนและการถดถอยเชิงเส้น สำหรับตัวแบบเอ็กซ์จีบูส และซัพพอร์ตเวกเตอร์แมชชีน ใช้การเลือกตัวแปรเข้าตัวแบบทั้งหมด 3 กรณี ได้แก่ กรณีตัวแปรมีค่า importance มากกว่า 0 กรณีตัวแปรมีค่า importance มากที่สุด 5 อันดับแรก และกรณีตัวแปรมีค่า importance มากที่สุด 10 อันดับแรก สำหรับตัวแบบการถดถอยเชิงเส้น ใช้การเลือกตัวแปรเข้าตัวแบบโดยพิจารณาจากค่า p-value เมื่อพิจารณาค่า MAE, RMSE และ R2 ระหว่างข้อมูลชุดฝึกฝนและชุดทดสอบพบว่า พบปัญหา overfitting ในกรณีที่เลือกตัวแปรที่มีค่า importance มากกว่า 0 ในตัวแบบตัวแบบเอ็กซ์จีบูส และซัพพอร์ตเวกเตอร์แมชชีน รวมถึงกรณีที่เลือกตัวแปรที่มีค่า importance มากที่สุด 10 อันดับแรก ในตัวแบบเอ็กซ์จีบูส เมื่อพิจารณาผลลัพธ์ในการวัดประสิทธิภาพของตัวแบบโดยข้อมูลชุดทดสอบ จะได้ผลลัพธ์ที่ว่า ตัวแบบ ซัพพอร์ตเวกเตอร์แมชชีนที่เลือกใช้ตัวแปรที่มีค่า importance มากที่สุด 10 อันดับแรก มีประสิทธิภาพที่ดีที่สุดในการทำนายราคาที่พักในกรุงเทพ ตัวแปรที่ส่งผลกระทบต่อการทำนายราคาที่พักสูงสุด 3 อันดับแรกคือ คือ จำนวนคนสูงสุดที่ที่พักรองรับ (accommodates), เขตท่องเที่ยว (tourist district) และ จำนวนห้องนอน (bedrooms) ตามลำดับ

คำสำคัญ : การถดถอยเชิงเส้น, การทำนายราคาที่พัก, ซัพพอร์ตเวกเตอร์แมชชีน, เอ็กซ์จีบูส, แอร์บีเอ็นบี

Independent Study Title Predicting the accomodation price in Bangkok by applying machine learning
Student Name Mr. Chayapat Natshivawong
Student Id 65056021
Degree Master of Science (Data Science and Analytics)
KMITL-Digital Analytics and Intelligence Center
Year 2024
Independent Study Advisor Assoc. Prof. Dr. Yuwadee Klomwises

Abstract

At present, tourism has become an industry that generates significant income for the country. The Airbnb platform is one of the platforms for sharing local accommodations for tourists to stay in Thailand. Therefore, we were motivated to study the prediction of accommodation prices in Thailand, with a particular focus on Bangkok, utilizing Airbnb data obtained from insideairbnb website. After the data has been cleaned, it consists of 29 columns and 7,924 rows. This research employs XGBoost, Support Vector Machine, and Linear Regression as base models. In case of the XGBoost and Support Vector Machine models, three distinct scenarios were used for variable selection: variables with an importance value greater than 0, the top 5 variables with the highest importance values, and the top 10 variables with the highest importance values. On the other hand, for the linear regression model, variables were selected based on their p-values. By evaluating the MAE, RMSE, and and R2 values between the training and test datasets, it was found that there is an issue of overfitting in cases where variables with an importance value greater than 0 are selected in the XGBoost and Support Vector Machine models, as well as in cases where the top 10 variables with the highest importance values are selected in the XGBoost model. Further analysis on the test dataset revealed that the Support Vector Machine model, which selects the top 10 variables with the highest importance values, demonstrates the most proficient performance in predicting accommodation prices in Bangkok. Moreover, the top three features that have the most impact on

predicting the accommodation price are the maximum capacity of the listing, tourist district, and the number of bedrooms.

Keywords : Linear Regression, Accommodation prices, Support Vector Machine, Airbnb, XGBoost

กิตติกรรมประกาศ

การศึกษาค้นคว้าอิสระฉบับนี้มิอาจสำเร็จได้ หากปราศจากความช่วยเหลือของหลายๆบุคคล ซึ่งผู้จัดทำได้รับความรับช่วยเหลือไว้ ได้แก่

ขอกราบขอบพระคุณ ผศ.ดร.ยุวดี กล่อมวิเศษ อาจารย์ที่ปรึกษาการค้นคว้าอิสระ อาจารย์ผู้ให้คำปรึกษาตั้งแต่การเลือกหัวข้อ การจัดการข้อมูล การรันตัวแบบ ไปจนถึงแนวทางต่างๆและคำแนะนำในการทำค้นคว้าอิสระให้สำเร็จ

ขอกราบขอบพระคุณ ผศ.ดร.วรางคณา กิ้มปาน และ ดร.จิรภัทร์ หยกรัตนศักดิ์ คณะกรรมสอบค้นคว้าอิสระที่สละเวลามาให้คำแนะนำและข้อเสนอแนะต่างๆที่ทำให้งานค้นคว้าอิสระครั้งนี้ดียิ่งขึ้น

ขอกราบขอบพระคุณ คณาจารย์วิทยาศาสตร์มหาบัณฑิต สาขาวิทยาการข้อมูลและการวิเคราะห์ ที่มอบความรู้ด้านต่างๆให้แก่ผู้จัดทำอันมีประโยชน์ต่อการทำค้นคว้าอิสระ

ขอกราบขอบพระคุณครอบครัว เพื่อน ที่คอยเป็นกำลังใจและให้คำปรึกษาเรื่องต่างๆจนทำให้การค้นคว้าอิสระครั้งนี้ประสบความสำเร็จได้

ชยพัทธ์ ณัฐศิววงศ์

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เศรษฐกิจแบบแบ่งปันคือแนวคิดการทำธุรกิจที่จะทำการจับคู่ระหว่างผู้ให้บริการที่มีทรัพย์สินที่ต้องการปล่อยเช่ากับผู้ใช้บริหารที่มีความต้องการในทรัพย์สิน (Wongjantorn, 2020) ยกตัวอย่าง เช่น Airbnb คือ แพลตฟอร์มสำหรับการที่ใช้ในการเปิดห้องว่างให้เช่าตามเมืองต่างๆ โดย Airbnb จะทำหน้าที่เหมือนแพลตฟอร์มไว้สำหรับเจ้าของที่พักในการปล่อยเช่าเพื่อที่จะหารายได้ ในขณะที่คนเช่าหรือนักท่องเที่ยวก็สามารถได้ที่พักในราคาที่ถูก จากผลวิจัยของมหาลัย Oxford พบว่า Airbnb เป็น 1 ใน แหล่งรายได้ที่สำคัญต่ออุตสาหกรรมท่องเที่ยวของไทย, ในปี 2022 หลังจากสถานการณ์ โควิด-19ได้เบาบางลง ยอดใช้จ่ายจากผู้ใช้ Airbnb ได้สร้างรายได้ให้แก่ประเทศไทยในหลายๆภาคส่วนโดยมีมูลค่าประมาณ 4 หมื่น 1 พันล้านบาท โดย มูลค่าพุ่งสูงขึ้น 5 เท่าจากปี 2021 โดยเฉพาะอย่างยิ่ง กับกรุงเทพ (ทีมข่าวคอร์ปอเรท-การตลาด กรุงเทพธุรกิจ, 2023) จากผลสำรวจของอโกด้า กรุงเทพเป็นอันดับ 1 ของเมืองที่นักท่องเที่ยวจองที่พักมากที่สุดในโลก ซึ่งมันบ่งบอกว่าการท่องเที่ยวในไทยกำลังฟื้นตัวไปในทิศทางที่ดีมากซึ่งมาพร้อมกับโอกาสเติบโต (ฐานเศรษฐกิจ, 2023)

ราคาที่พักจึงเป็น 1 ในปัจจัยสำคัญที่ช่วยส่งเสริมการท่องเที่ยว Airbnb จะมีระบบ ที่ชื่อว่า Smart Pricing สำหรับผู้ปล่อยเช่าเพื่อที่จะสามารถเพิ่มยอดการจองโดยทำการแนะนำราคาที่เหมาะสมโดยการใส่ข้อมูลของที่พักเพื่อทำการวิเคราะห์ราคาที่เหมาะสมออกมา แต่ระบบของ Airbnb ก็ยังมีข้อบกพร่องอยู่เนื่องจาก เทรนด์ของราคาบ้านในแต่ละที่ของโลกนั้นไม่เหมือนกัน จากกงานวิจัยของ (Yang, 2021) ได้กล่าวไว้ว่า การมีสิ่งของใช้จำเป็นในห้องพักเป็น 1 ใน ปัจจัยสำคัญการเพิ่มราคา การเพิ่มสิ่งอำนวยความสะดวกอย่างเช่น อาหารเช้า หรือ อ่างอาบน้ำทำให้สามารถเพิ่มราคาของที่พักในเมืองปักกิ่งได้ ในขณะที่พื้นที่รอบๆที่พักก็ส่งผลกระทบต่อทั้งราคาและจำนวนที่พักในพื้นที่เหล่านั้น จากงานวิจัยในนิวยอร์ก (Zhu, Li, และ Xie, 2020) พบว่า พื้นที่บริเวณที่มีแลนด์มาร์กตั้งอยู่ จะมีราคาที่สูงและจำนวนที่เยอกว่า เมื่อเทียบกับพื้นที่อื่นๆโดยงานวิจัยชิ้นนี้จะโฟกัสไปที่กรุงเทพ เมืองหลวงของไทย เพื่อที่จะเข้าใจเทรนด์ของราคาที่พักในกรุงเทพมหานคร

สำหรับงานวิจัยฉบับนี้ ทางผู้วิจัยจะดึงข้อมูลจากเว็บไซต์ Airbnb ซึ่งเป็นเว็บไซต์ที่รวบรวมข้อมูลที่พักของAirbnbตามเมืองต่างๆทั่วโลก โดยจะทำการดึงข้อมูลของกรุงเทพซึ่งมีการอัพเดทล่าสุดวันที่ 22 กันยายน พ.ศ. 2566 แล้วทำการจัดการและประยุกต์ข้อมูลให้เหมาะสมก่อนทำการนำเข้าไปใช้วิธีการ XGBoost, ซัพพอร์ตเวกเตอร์แมชชีน และ การถดถอยเชิงเส้น ผู้วิจัยหวังว่า ข้อมูลที่ได้จาก

การวิจัยครั้งนี้จะสามารถส่งเสริมและสนับสนุนการตั้งราคาที่พักใน platform Airbnb ของคนไทยให้มีประสิทธิภาพเพื่อที่จะส่งเสริมการท่องเที่ยวในไทยให้ดียิ่งขึ้น

1.2 วัตถุประสงค์ของงานค้นคว้าอิสระ

1) เพื่อเปรียบเทียบวิธีการทำนายราคาที่พัก ใน แพลตฟอร์ม Airbnb ในกรุงเทพ

2) เพื่อพยากรณ์ราคาที่พักของในแพลตฟอร์ม Airbnb ในกรุงเทพโดยใช้ตัวแบบ XGBoost, ซัพพอร์ตเวกเตอร์แมชชีนและ การถดถอยเชิงเส้น

1.3 ขอบเขตการจัดสร้างโครงงาน

ในงานวิจัยครั้งนี้ใช้ข้อมูลทุติยภูมิที่ทำการสำรวจโดยได้จากข้อมูลจากเว็บไซต์ insideairbnb ซึ่งเป็นเว็บไซต์ที่รวบรวมข้อมูลที่พักบนแพลตฟอร์ม Airbnb ตามเมืองต่างๆทั่วโลก โดยจะมีการปรับปรุงทุกๆ3 เดือน โดยข้อมูลที่ใช้จะเป็นชุดข้อมูลของจังหวัด กรุงเทพซึ่งได้รับการอัพเดทล่าสุด ณ วันที่ 22 กันยายน พ.ศ. 2566 โดยผู้วิจัยจะใช้วิธีการ XGBoost, ซัพพอร์ตเวกเตอร์แมชชีน และ การถดถอยเชิงเส้นซึ่งข้อมูลจะถูกแบ่งเป็นข้อมูลเรียนรู้ 80 เปอร์เซ็นต์ และ ข้อมูลทดสอบ 20 เปอร์เซ็นต์ โดยเครื่องมือที่ใช้งานในการจัดการข้อมูลและทำตัวแบบ คือ excel และ python

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) ทราบถึงวิธีที่เหมาะสมในการทำนายราคาที่พักใน แพลตฟอร์ม Airbnb ในกรุงเทพ

2) สามารถพยากรณ์ราคาที่พักจากข้อมูลหรือลักษณะของที่พักเพื่อการตั้งราคาที่เหมาะสมสำหรับดึงดูดนักท่องเที่ยว

บทที่ 2

วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

การศึกษาวิจัยเรื่องการทำนายตัวแปร ราคาที่พักในกรุงเทพมหานครโดยประยุกต์ใช้การเรียนรู้ของเครื่องเป็นหลัก ผู้วิจัยได้ทำรวบรวมแนวคิดทฤษฎีและหลักการต่างๆจากทฤษฎีบทและได้งานวิจัยที่เกี่ยวข้องดังนี้

1) เศรษฐกิจแบบแบ่งปัน 2) ประวัติแอร์บีแอนด์บี 3) การเรียนรู้ของเครื่อง. 4) เอ็กจีบูส

5) ซัพพอร์ตเวกเตอร์แมชชีน 6) การถดถอยเชิงเส้น 7) การเปรียบเทียบประสิทธิภาพการทำนาย

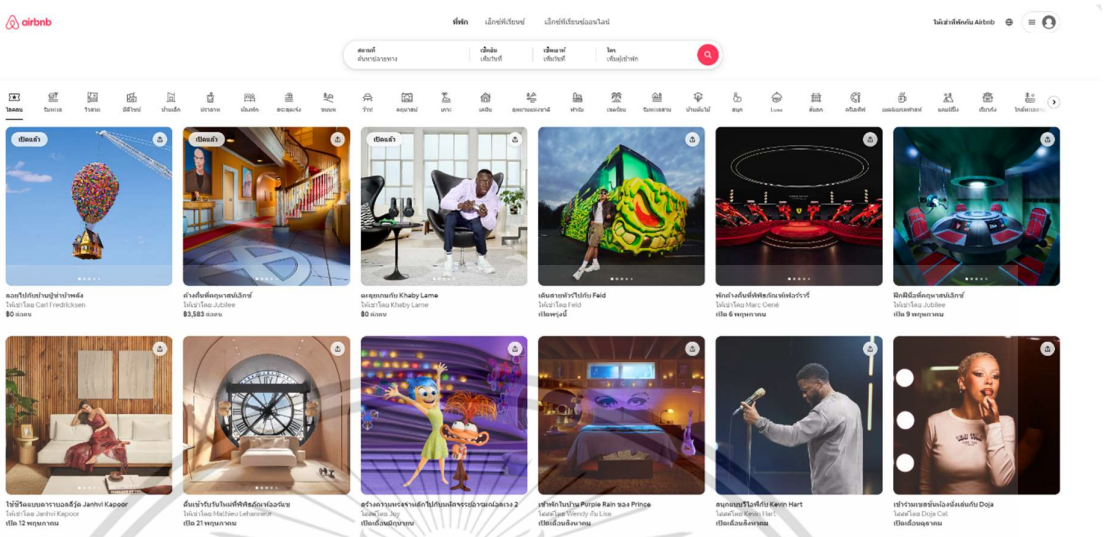
8)งานวิจัยที่เกี่ยวข้อง

2.1 เศรษฐกิจแบบแบ่งปัน

เศรษฐกิจแบบแบ่งปันเป็นแนวคิดการทำธุรกิจแบบ peer to peer โดยเป็นการทำธุรกิจจากการนำทรัพย์สินของผู้ใช้บริการที่ไม่ได้ใช้งานแล้วกับผู้บริโภคที่มีความต้องการในทรัพย์สินนั้น เพื่อนำทรัพย์สินนั้นมาให้ผู้บริโภคยืมหรือใช้บริการ เช่น ที่พัก รถยนต์ แรงงาน โดยจะเป็นการใช้ทรัพย์สินส่วนเกิน (Excess Capacity) ให้มีประโยชน์และประสิทธิภาพสูงสุด โดยการทำธุรกิจแบบแบ่งปันจะเป็นการเพิ่มรายได้ให้แก่ผู้ให้บริการ เพิ่มทางเลือกให้แก่ผู้บริโภค และจะเติบโตไปได้ด้วยการแบ่งปันและร่วมมือของทั้งสองฝ่าย

2.2 แอร์บีเอ็นบี

แอร์บีเอ็นบีเกิดขึ้นเมื่อปี ค.ศ.2007 โดย Brian Chesky และ Joe Gebbia ได้เกิดไอเดียในการทำธุรกิจ ปล่อยให้เช่าขึ้นมา โดยใช้ชื่อ ว่า airbedandbreakfast.com ก่อนภายหลังจะเปลี่ยนมาเป็น airbnb.com ธุรกิจ Airbnb มีจำนวนผู้เข้าพักมากถึง 400 ล้านคนตั้งแต่เปิดตัว โดยในประเทศไทย มีที่พักกับทาง Airbnbมากกว่า 60,000 แห่ง (Komkid, 2015)



รูปที่ 2.1 เว็บเพจของ Airbnb

(ที่มา : https://th.airbnb.com/)

2.3 การเรียนรู้ของเครื่อง

การเรียนรู้ของเครื่องคือการที่เราสั่งคอมพิวเตอร์เรียนรู้การแพทเทิร์นของข้อมูลเพื่อสร้างตัวแบบขึ้นมาโดยการให้ตัวแบบเรียนรู้จากการอินพุตข้อมูลที่ไม่เคยเห็น แล้วทำการเรียนรู้ข้อมูลแล้ววัดประสิทธิภาพของตัวแบบแล้วทำซ้ำวนไปเพื่อให้ได้ตัวแบบที่มีประสิทธิภาพที่ดีที่สุด การเรียนรู้ของเครื่องปัจจุบันแบ่งเป็น 3 แบบ

การเรียนรู้แบบมีผู้สอน (Supervised Learning) เป็นการเรียนรู้ของเครื่องที่ใช้ในการทำนายเหตุการณ์ในอนาคตโดยทำการเรียนรู้จากข้อมูลในอดีตเพื่อเรียนรู้และสร้างตัวแบบในการทำนายขึ้นมา โดยแบ่งเป็น 2 ประเภท 1. การถดถอย (regression) คือ การทำนายผลลัพธ์ที่เป็นตัวเลข เช่น การทำนายราคาที่พัก 2. การจำแนก (classification) คือ การทำนายผลลัพธ์ที่เป็นกลุ่ม เช่น การทำนายว่าคนสอบตกหรือผ่าน การทำนายว่าใครเป็นโรคหรือไม่เป็น

การเรียนรู้แบบไม่มีผู้สอน (Unsupervised Learning) เป็นการเรียนรู้ของเครื่องที่ใช้ในข้อมูลที่ไม่มี label โดยจะทำการเรียนรู้โดยการจับกลุ่มของข้อมูลที่มีลักษณะใกล้เคียงกันให้เป็นกลุ่มเดียวกัน (clustering) เช่น การจำแนกกลุ่มลูกค้า เพื่อทำการจัดกลุ่มลูกค้าและทำการเอา insight ที่ได้มาวิเคราะห์ต่อไป

การเรียนแบบเสริมกำลัง (Reinforcement Learning) เป็นการเรียนรู้ของเครื่องที่จะไม่ได้เรียนรู้จากการอินพุตข้อมูลลงไปให้เรียนรู้ แต่จะเป็นการเรียนรู้จากสภาพแวดล้อม โดยทำการจำลองภายใต้สถานการณ์ต่างๆเพื่อทำการตัดสินใจเลือกเส้นทางที่น่าจะได้ผลลัพธ์ที่ดีที่สุด โดยการจำลองจะช่วยในการพัฒนาการตัดสินใจของตัวเครื่องให้ดียิ่งขึ้น ยกตัวอย่างเช่น AlphaGo ที่แข่งโกะกับคน หรือ self-driving car ระบบรถไร้คนขับ

2.4 เอ็กจีบูส (XGBoost)

XGBoost ย่อมาจาก eXtreme Gradient Boosting โดยจะมีองค์ประกอบหลักดังนี้ (Chen & Guestrin, 2016)

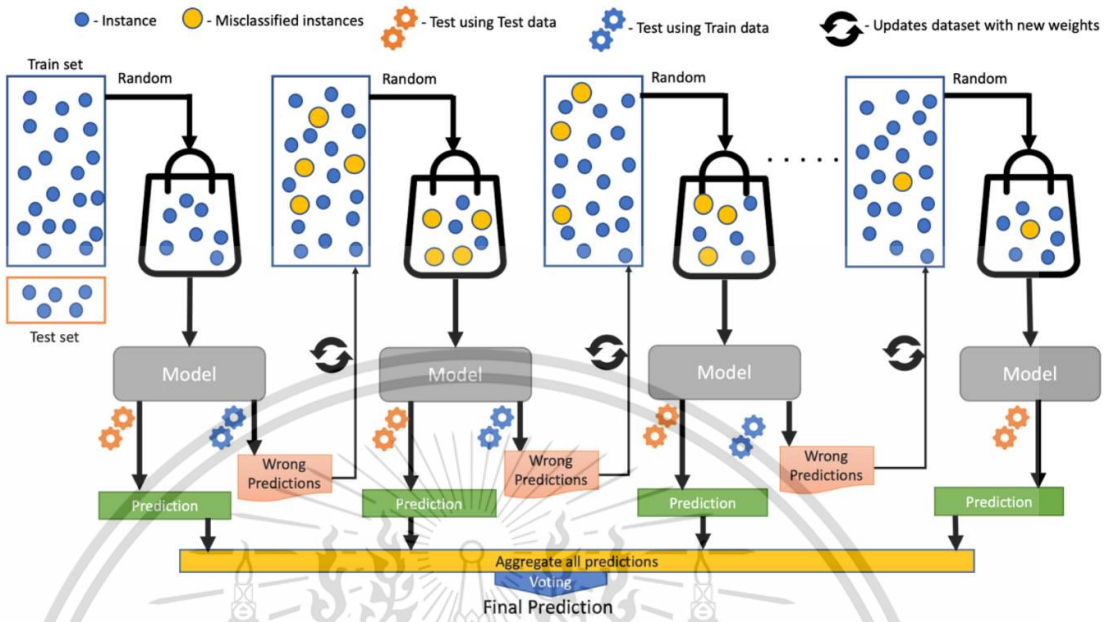
1. Gradient Boosting Framework

เป็นการเทคนิคของการเรียนรู้ของเครื่องที่ใช้ใน regression และ classification โดยมี concept คือ การนำ หลายตัวแบบที่มีมารวมกัน เพื่อสร้างเปSequential Model Training: การฝึกสอนใน XGBoost เกิดขึ้นแบบลำดับชั้น, โดยตัวแบบถัดไปในลำดับจะพยายามปรับปรุงข้อผิดพลาดที่เกิดจากตัวแบบก่อนหน้า แล้วนำไปเรียนรู้เพื่อทำให้ได้ตัวแบบใหม่ที่ประสิทธิภาพดียิ่งขึ้น

Loss Function Optimization: XGBoost มีการลดค่าความผิดพลาดของค่าที่ทำนายและค่าจริง ทุกครั้งที่มีการสร้างตัวแบบขึ้นมา เพื่อเพิ่มประสิทธิภาพให้แก่ตัวแบบ

Regularized Learning: XGBoost มีการรวมฟังก์ชัน Regularization ไว้ในตัวแบบเพื่อที่จะลดปัญหา overfitting ที่อาจเกิดขึ้นได้

Boosting Iterations: ในทุกรอบของการเรียนรู้จากข้อผิดพลาดของตัวแบบ โดยจะมีการใช้hyperparameter อย่าง learning rate ที่ใช้ไว้ปรับความเร็วในการเรียนรู้ของตัวแบบ และ n_estimators ที่คอยกำหนดจำนวนรอบในการเรียนรู้



รูปที่ 2.2 หลักการทำงานของ gradient boosting

(ที่มา : https://dzone.com/articles/XGBoost-a-deep-dive-into-boosting)

2. Decision Tree XGBoost ใช้ decision tree เป็นตัวแบบพื้นฐานในการเรียนรู้โดยในแต่ละ node จะมีการแบ่งเป็น 2 กลุ่ม โดยจะทำวนจนกว่าถึงเป้าหมายที่ตั้งไว้

1. โครงสร้างของต้นไม้

โครงสร้าง เป็นแบบ binary tree ที่ในแต่ละ node จะมี 2 node ที่แบ่งแยก feature ของข้อมูล

2. การเลือกการแยก (Split Selection)

XGBoost ใช้ ค่า gain ในการตัดสินใจเลือก node โดยจะเลือก node ที่มีค่า gain ที่มากที่สุด

3. Regularization ในต้นไม้

สำหรับการ regularization ใน decision tree จะมีการกำหนดค่าต่างๆใน ตัวแบบ decision tree เพื่อทำให้ตัวแบบไม่เกิดปัญหา overfitting เช่น

ค่า Maximum Depth ที่คอยกำหนดความลึกของต้นไม้ที่สามารถเติบโตเพื่อกำหนดความซับซ้อนของตัวแบบ หรือ การทำ Pruning :ซึ่งก็คือการตัดแต่งต้นไม้เพื่อทำให้ต้นไม้หยุดเติบโตในกรณีที่ การเพิ่ม node ไม่ได้ทำให้ตัวแบบมีประสิทธิภาพที่ดีขึ้น

3. Regularization

XGBoost เป็น gradient boosting ที่มีฟังก์ชัน regularization อยู่ในตัวแบบ โดย regularization มีไว้เพื่อป้องกันปัญหา overfitting โดย ในตัวแบบ จะมีทั้ง L1 (Lasso Regression) และ L2 (Ridge Regression) ใน objective function (Tewari, 2021)

L1 Regularization (Lasso): เป็นการทำให้ค่า parameter ที่มีความสำคัญน้อย หรือ แทบไม่มีความสำคัญมีค่าเป็นศูนย์ ทำให้ลดความซับซ้อนให้แก่ตัวแบบ โดยนิยมใช้เวลาต้องการลด จำนวน feature ในตัวแบบ

L2 Regularization (Ridge): มีการเพิ่มบทลงโทษให้แก่ parameter โดยยิ่งมีค่ามาก บทลงโทษยิ่งสูง โดยนิยมใช้เวลาตัวแบบมีความซับซ้อนสูง.

สมการของXGBoost (Yang, 2021) และ (Chen และ Guestrin, 2016)

สมมุติให้ชุดข้อมูลของเราเป็นตัวแปรd โดย $d = \{(x_i, y): \mathrm{i} = 1 \dots n, x_i \in R^m, y \in R\}$ โดยที่ $n =$ จำนวนข้อมูล, $m =$ จำนวน $feature$ และ $y, \hat{y}$ เป็นตัวแปรในฟังก์ชั่น

$$\hat{y} = \phi(x_i) = \sum_{k=1}^{T} f_k(x_i) \tag{2.1}$$

f_k คือการถดถอยแบบต้นไม้ โดย $f_k(x_i)$ จะแทนคะแนนตั้งแต่ ต้นไม้ที่ k จนถึง ข้อมูลที่ T โดยต้องทำการ minimize objective function ข้างล่าง เพื่อให้ ฟังก์ชัน f_k ทำงานได้

$$L(\emptyset) = \Sigma_i l(y, \hat{y}) + \Sigma_k l\Omega(f_k) \tag{2.2}$$

L คือ lost function เพื่อการลดความซับซ้อนของ model โดยจะทำการแอด บทลงโทษ ω เข้าไปในฟังก์ชันด้านล่าง

$$\Omega(fk) = \gamma T + \frac{1}{2}\lambda \sum_{k=1}^{T} w_k^2 + \alpha \sum_{k=1}^{T} |w_k| \tag{2.3}$$

ฟังก์ชัน $\Omega(fk)$ ป้องกัน overfitting ของ model

γ เป็น parameter ที่ใช้ในการควบคุมบทลงโทษของจำนวนต้นไม้ T.

λ ใช้ควบคุมน้ำหนักของใบไม้ w และเป็น coefficient สำหรับควบคุม L2 regularization

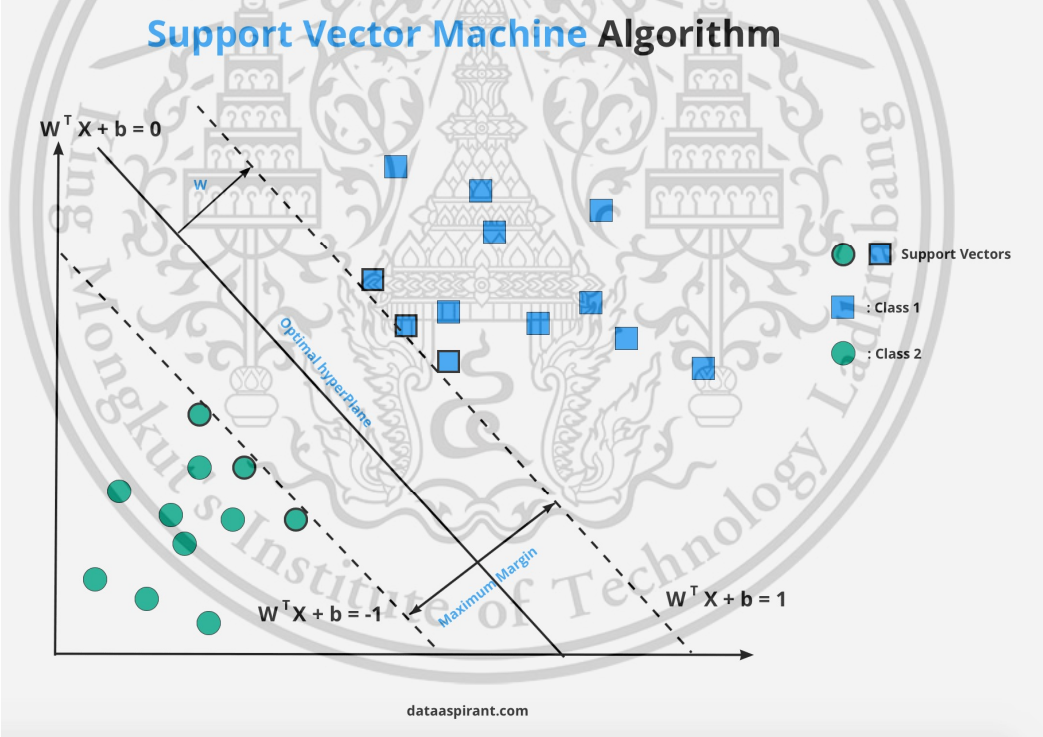
α เป็น coefficient ของ L1 regularization

k เป็นจำนวนของต้นไม้

w_k เป็น คะแนนบนใบไม้

2.5 ซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machine)

เป็นอัลกอริทึมประเภทการเรียนรู้แบบมีผู้สอน(supervised learning) ที่ถูกพัฒนาโดย Vladimir Vapnik (Corinna & Vladimir, 1995) โดยจะเป็นอัลกอริทึมที่มีความยืดหยุ่นและทำงานได้ดี เมื่อใช้งานกับข้อมูลที่จำนวนน้อยและมีความซับซ้อนมาก (กิตตินราดร, 2563).



รูปที่ 2.3 ซัพพอร์ตเวกเตอร์แมชชีน

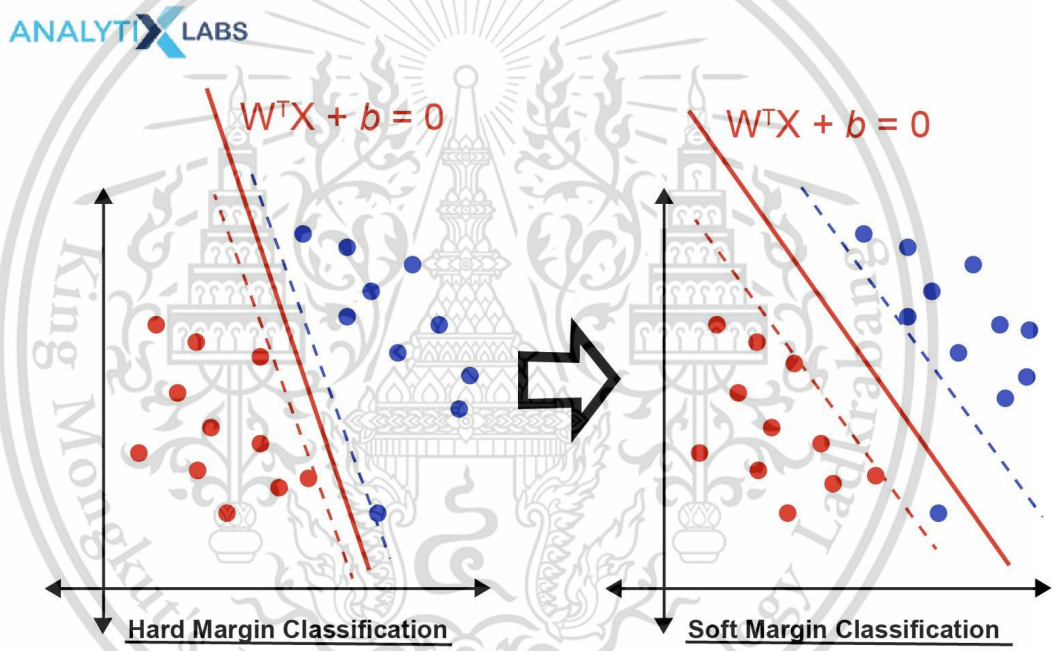
(ที่มา : จาก https://dataaspirant.com/svm-kernels/)

จากรูปที่ 2.3 หลักกการของ ซัพพอร์ตเวกเตอร์แมชชีน คือการที่เราจะแบ่งชุดข้อมูลออกเป็นสอง ประเภท เพื่อที่จะทำการหาเส้น optimal hyperplane ที่ทำให้ระยะห่างระหว่าง เส้น optimal

hyperplane กับเส้นปะ สอง เส้น มีระยะห่างมากที่สุด โดยที่ข้อมูลจะถูกแบ่งออกเป็น 2 ประเภท คือ ผลลัพธ์ที่เป็นบวกและผลลัพธ์ที่เป็นลบ ถ้าผลลัพธ์เป็นบวก ค่า $\hat{y}$ จะเป็น 1 ถ้าผลลัพธ์เป็นลบ ค่า $\hat{y}$จะเป็น -1 โดยของเส้นประอยู่ภายใต้ 2 เงื่อนไข (กิตตินราดร, 2563)

เงื่อนไขที่ 1 คือ ข้อมูลห้ามอยู่ในพื้นที่ระหว่างเส้นประ มีชื่อเรียกว่า Hard Margin Classification

เงื่อนไขที่ 2 คือ ข้อมูลอยู่ในพื้นที่ระหว่างเส้นประได้ มีชื่อเรียกว่า Soft Margin Classification



รูปที่ 2.4 แสดงเงื่อนไข Hard Margin และ Soft Margin ของซัพพอร์ตเวกเตอร์แมชชีน

(ที่มา : จาก https://entri.app/blog/what-is-svm-algorithm-in-machine-learning/)

ฟังก์ชัน สำหรับ SVM for regression (Géron, 2019)

$$h_\theta = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b \quad (2.4)$$

$$h_\theta = w^Tx + b \quad (2.5)$$

เมื่อ W คือ ค่าถ่วงน้ำหนัก (weight)

B คือ ค่าความเอนเอียง (bias)

x = ชุดข้อมูลที่ใช้ในการเทรน

โดยผลลัพธ์หรือค่า y ที่ได้จะมีสองค่า คือ 1และ -1 โดยค่าy จะเป็นไปตามเงื่อนไขดังนี้

$$w^T x + b < 0 : y = -1 \tag{2.6}$$

$$w^T x + b \geq 0 : y = 1 \tag{2.7}$$

เมื่อทำการนิยามเส้น optimal hyperplane กำหนดเส้นประทั้งสองด้านของเส้น optimal hyperplane โดยเส้นประแต่ละด้านคือตำแหน่งที่$h_\theta(x)$ เท่ากับ -1 และ 1 โดยเป้าหมายคือการหาค่า w และ b ที่ทำให้ระยะห่างของเส้นประและเส้น optimal hyperplaneมีค่ามากที่สุดโดยอยู่ภายใต้เงื่อนไข hard margin หรือ soft margin

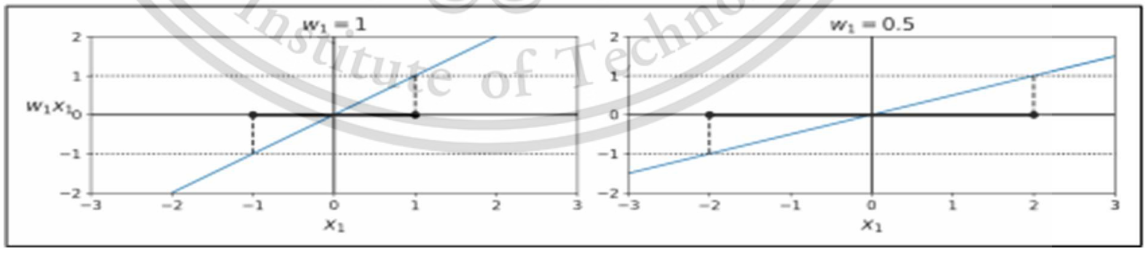
โดยที่ความชันของ $h_\theta(x)$ เท่ากับ norm vector ของ ค่า w โดยดูจากได้จาก สมการ 2.10

$$\frac{\partial}{\partial_x} h_\theta(x) = \sum_{i=1}^{m} |w_i| \tag{2.8}$$

$$\frac{\partial}{\partial_x} h_\theta(x) = |w_1| + |w_2| + \cdots + |w_i| \tag{2.9}$$

$$\frac{\partial}{\partial_x} h_\theta(x) = \|w\|_1 \tag{2.10}$$

ค่า w ยิ่งน้อย ระยะห่างก็จะยิ่งเพิ่มมากขึ้นโดยสังเกตได้จากรูปที่ 2.5



รูปที่ 2.5 แผนภาพแสดงความสัมพันธ์ระหว่าง ค่า w และ ระยะห่าง (Geron, 2019)

ดังนั้น การminimizeค่า w จึงจำเป็นในการเพิ่มระยะห่างให้มากที่สุดโดยที่ยังคงไว้ในเงื่อนไข hard margin และ soft margin สำหรับ hard margin โดยหมายความว่าฟังก์ชันจะต้องมีค่ามากกว่า 1 สำหรับค่าบวก หรือ ฟังก์ชันมีค่าน้อยกว่า -1 สำหรับค่าลบ โดยเราจะกำหนดให้ $t^{(i)}$ = 1 สำหรับค่า

บวก และ $t^{(i)} = -1$ สำหรับค่าลบ จึงเขียนข้อจำกัดได้ว่า $t^{(i)}(w^T x^{(i)}+b) \geq 1$ สำหรับทุกคำตอบ โดย objective function ของ ฟังก์ชัน SVM algorithm โดยภายใต้เงื่อนไข hard margin เป็นดังสมการ 2.11 และ 2.12 (Géron, 2019)

$$\text{Minimize}_{w,b} \frac{1}{2} w^T w \tag{2.11}$$

$$\text{Subject to } t^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 \text{ for i = 1,2,...,m} \tag{2.12}$$

สำหรับ soft margin จะมีค่า slack variable โดย $\zeta^{(i)} \geq 0$ สำหรับแต่ละคำตอบ ค่า $\zeta^{(i)}$ วัดว่า ใช้ข้อมูลจำนวนเท่าใดที่จะอนุญาตให้อยู่ระหว่างเส้นประโดยต้องทำให้ ค่า $\zeta^{(i)}$ เล็กที่สุดเท่าที่เป็นไปได้ ในขณะเดียวกัน ก็ minimize objective function เพื่อเพิ่ม marginให้มากที่สุด โดยจะมีค่า C ซึ่งเป็น hyperparameter ที่ถูกกำหนดขึ้นมาเพื่อกำหนดระดับของ slack variable และ objective function ของ ฟังก์ชัน SVM algorithm โดยภายใต้เงื่อนไข soft margin เป็นดังสมการที่ 2.13 และ 2.14

$$\text{Minimize}_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{m} \zeta^{(i)} \tag{2.13}$$

$$\text{Subject to } t^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 - \zeta^{(i)} \text{ and } \zeta^{(i)} \text{ for I = 1,2,...,m} \tag{2.14}$$

Kernel

คือเทคนิคทางคณิตศาสตร์ที่ใช้ในการทำงานกับข้อมูลที่มีความซับซ้อนมากหรือไม่ได้มีความสัมพันธ์เป็นเส้นตรง โดยจะทำการเปลี่ยนเป้าหมายในการ optimize เพื่อรองรับการ optimize ตัวแปรแบบ polynomial ได้โดยได้ผลลัพธ์แบบเดียวกัน (กิตตินราดร, 2563) โดย objective function ใหม่จะเป็นไปตามสมการนี้

$$\text{Minimize } \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha^{(i)} \alpha^{(j)} t^{(i)} t^{(j)} x^{(i)T} x^{(j)} - \sum_{i=1}^{w} \alpha^{(i)} \tag{2.15}$$

$$\text{Subject to } \alpha^{(i)} \geq 0 \text{ for i =1,2,...,m} \tag{2.16}$$

โดย kernel จะแทนที่ค่า $x^{(i)T} x^{(j)}$ ในสมการ ให้เป็น$\left(x^{(i)T} x^{(j)}\right)^2$ เสมือนการเปลี่ยนรูปฟังก์ชันให้เป็น second-degree polynomial

ฟังก์ชัน kernel จะมีค่าดังสมการ

$$K(a,b) = (a^T b)^2 \quad (2.17)$$

โดย kernel ที่นิยมใช้งานมีดังต่อไปนี้ (กิตตินราดร, 2563)

1.เส้นตรง (Linear)

$$K(a,b) = a^T b \quad (2.18)$$

2.พหุนาม (Polynomial)

$$K(a,b) = (\gamma a^T b + r)^d \quad (2.19)$$

3.เกาเซียน เรเดียนเบซิสฟังก์ชัน (Gaussian RBF)

$$K(a,b) = \mathrm{e}^{(-\gamma \|a-b\|^2)} \quad (2.20)$$

4.ซิกมอยด์ (Sigmoid)

$$K(a,b) = tanh(\gamma a^T b + r)^b \quad (2.21)$$

d = ดีกรีของพหุนาม ยิ่งดีกรีมาก แก้ไขปัญหาได้ซับซ้อนมากขึ้น (อาจทำให้เกิดปัญหาoverfitting)

γ = ปรับความชันของkernel function โดยจะการปรับ γจะมีผลต่อความสามารถในการรับมือข้อมูลที่ไม่มีความสัมพันธ์เป็นเส้นตรง

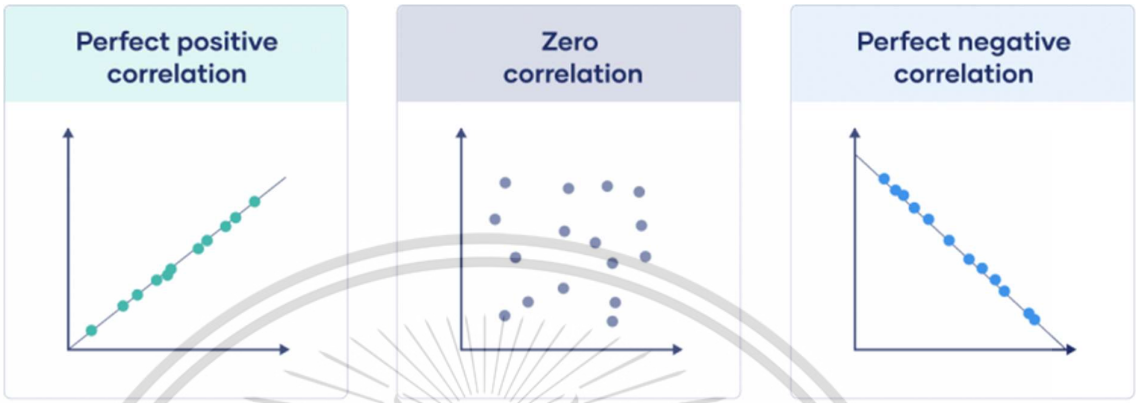
r = term อิสระที่ปรับเปลี่ยนการ mapping ของ feature space เข้าสู่มิติที่สูงขึ้น มันสามารถส่งผลต่อการ covergence ของอัลกอริทึมและรูปทรงของเส้นแบ่งขอบเขตการตัดสินใจได้

2.6 การถดถอยเชิงเส้น (Linear Regression)

การถดถอยเชิงเส้น (Linear Regression) เป็นวิธีการทางสถิติที่ใช้ในการหาความสัมพันธ์ข้อมูลระหว่างตัวแปรตาม (independent variable) และตัวแปรอิสระ (dependent variable) โดยความสัมพันธ์ของข้อมูลจะเป็นเส้นตรงหรือใกล้เคียง (Data Innovation and Governance Institute, 2022)

โดยความสัมพันธ์ระหว่างตัวแปรต้นและตัวแปรตามที่เราได้มา จะเรียกว่า correlation โดยจะมีค่าอยู่ระหว่าง -1 ถึง 1 โดยยิ่งค่าเข้าใกล้ -1 หรือ 1 จะบ่งบอกถึงความสัมพันธ์ที่เป็นเส้นตรงแบบ

ลบหรือ บวก ตามลำดับ หากค่าเข้าใกล้ 0 จะแสดงถึงความสัมพันธ์ที่ไม่ได้เป็นเส้นตรง (Data Innovation and Governance Institute, 2022)



รูปที่ 2.6 รูปแบบของ correlation

(ที่มา : https://www.scribbr.com/statistics/correlation-coefficient/)

การถดถอยเชิงเส้นจะถูกแบ่งเป็น 2 แบบ คือ

การถดถอยเชิงเส้นอย่างง่าย

$$y = \beta_0 + \beta_1 X \tag{2.22}$$

y คือ ตัวแปรตาม (dependent variable)

X คือ ตัวแปรอิสระ (independent variable)

β_0 คือ ค่าคงที่เมื่อ ค่า $x = 0$

β_1 คือ อัตราส่วนการเปลี่ยนแปลงของ y ต่อการเปลี่ยนแปลงของ x (ความชัน)

การถดถอยเชิงเส้นแบบพหุคูณ

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \varepsilon \tag{2.23}$$

y คือ ตัวแปรตาม (dependent variable)

X คือ ตัวแปรอิสระ (independent variable)

β_0 คือ ค่าคงที่เมื่อ ค่า $x = 0$

$\beta_1, \ldots, \beta_n$ คือ ค่าสัมประสิทธิ์การถดถอยเชิงเส้นของตัวแปร $X_1, \ldots, X_n$

ε คือ ค่าความคลาดเคลื่อน

การทดสอบสมมุติฐานสำหรับ β_0

$$H_0: \beta_0 = 0 \quad (2.24)$$

$$H_1: \beta_0 \neq 0 \quad (2.25)$$

Null hypothesis (H0) คือ สมมุติฐานที่ ค่าคงที่ เท่ากับ 0

Alternative hypothesis (H1) คือ สมมุติฐานที่ ค่าคงที่ ไม่เท่ากับ 0

โดยจะทำการดูค่า p-value เพื่อใช้ในการตัดสินใจทางนัยสำคัญ โดยจะทำการเทียบกับค่า significance(α)

p-value ≤ α จะทำการปฏิเสธ null hypothesis และสรุปได้ว่า ค่าคงที่ไม่เท่ากับ 0

p-value > α จะทำการยอมรับ null hypothesis และสรุปได้ว่า ค่าคงที่เท่ากับ 0

การทดสอบสมมุติฐานสำหรับ ตัวแปร $X_1, \ldots, X_n$

$$H_0: \beta_n = 0 \quad (2.26)$$

$$H_1: \beta_n \neq 0 \quad (2.27)$$

Null hypothesis (H0) คือ สมมุติฐานที่ ค่าสัมประสิทธิ์ เท่ากับ 0 นั่นคือตัวแปร X ไม่ควรอยู่ในตัวแบบ

Alternative hypothesis (H1) คือ สมมุติฐานที่ ค่าสัมประสิทธิ์ไม่เท่ากับ 0 นั่นคือตัวแปร X ควรอยู่ในตัวแบบ

โดยจะทำการดูค่า p-value เพื่อใช้ในการตัดสินใจทางนัยสำคัญ โดยจะทำการเทียบกับค่า significance (α)

p-value ≤ จะทำการปฏิเสธ null hypothesis และสรุปได้ว่า ค่าสัมประสิทธิ์ไม่เท่ากับ 0 และค่า Xn มีผลต่อตัวแปรตาม

p-value > α จะทำการยอมรับ null hypothesis และสรุปได้ว่า ค่าสัมประสิทธิ์เท่ากับ 0 และค่า Xn ไม่มีผลต่อตัวแปรตาม

2.7 วิธีวัดประสิทธิภาพของตัวแบบ

การวัดผลของการทำนายราคาที่พักมีเพื่อทำการวัดประสิทธิภาพของเทคนิคที่ใช้ในการทำนาย โดยเป็นการนำค่าของราคาที่ทำนายได้มาเปรียบเทียบกับข้อมูล เพื่อวัดประสิทธิภาพของตัวแบบ โดยค่าที่ใช้วัดประสิทธิภาพ จะมี Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) และ R-squared (Visitora-at, 2019)

Mean Absolute Error (MAE) คือการนำผลบวกของค่าสัมบูรณ์ของค่าความผิดพลาดมาหาค่าเฉลี่ย โดยมีสูตรดังนี้.

$$MAE = \frac{1}{n} \times \sum_{i=1}^{n} |\hat{y}_i - y_i| \quad (2.28)$$

$\hat{y}_i$ = ค่าประมาณของตัวแปรตอบสนอง

y_i = ค่าของตัวแปรตอบสนอง

n = จำนวนชุดข้อมูล

RMSE หรือ Root Mean Squared Error คือการการนำผลบวกของค่ายกกำลังสองของค่าความผิดพลาดมาหาค่าเฉลี่ย แล้วใส่ square root โดยมีสูตรดังนี้.

$$RMSE = \sqrt{\frac{1}{n} \times \sum_{i=1}^{n} (\hat{y}_i - y_i)^2} \quad (2.29)$$

$\hat{y}_i$ = ค่าประมาณของตัวแปรตอบสนอง

y_i = ค่าของตัวแปรตอบสนอง

n = จำนวนชุดข้อมูล

R^2 หรือ R-squared คือค่าความผันแปรของตัวแปรตามที่สามารถอธิบายได้ด้วยตัวแบบมีอยู่เป็นสัดส่วนเท่าใด

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \quad (2.30)$$

$\hat{y}_i$ = ค่าประมาณของตัวแปรตอบสนอง

y_i = ค่าของตัวแปรตอบสนอง

$\bar{y}_i$ = ค่าเฉลี่ยของตัวแปรตอบสนอง

n = จำนวนชุดข้อมูล

2.8 งานวิจัยที่เกี่ยวข้อง

Siqi Yang (2021) ได้ทำการวิจัยเรื่อง ตัวแบบการทำนายราคาที่พักใน Airbnb (Yang, 2021) โดยได้ทำการแบ่ง สิ่งอำนวยความสะดวกเป็น หลายหลายหมวดหมู่ เพื่อทำการ ตรวจสอบว่าสิ่งอำนวยความสะดวกแต่ละอย่างนั้นจะทำ การเพิ่มราคาให้ที่พัก ลดราคา ที่พัก หรือไม่มีผลกระทบกับราคาที่พัก ในกรุงปักกิ่งประเทศจีน โดยได้ทำการทำนายราคาที่พักในAirbnb ในกรุงปักกิ่งโดยใช้เทคนิค XGBoost และ โครงข่ายประสาทเทียม ผลลัพธ์ที่ได้จะพบว่า XGBoost มีประสิทธิภพที่ดีกว่าเมื่อเทียบกับโครงข่ายประสาทเทียมโดย XGBoost ตัวแบบมีค่าr-squared เท่ากับ 0.8112 และ 0.6549 สำหรับชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบตามลำดับ ในขณะที่ ตัวแบบโครงข่ายประสาทเทียม มีค่าr-squared เท่ากับ 0.5087 และ 0.5019 สำหรับชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบตามลำดับ

Ang Zhu and Others (2020) ได้ทำการวิจัยเรื่อง การทำนายที่พักในAirbnbในเมืองนิวยอร์กด้วยการเรียนรู้ของเครื่อง (Zhu, Li, และ Xie, 2020) ทำการแบ่ง พื้นที่แต่ละพื้นที่ของประเทศ สหรัฐอเมริกา เป็น 4 แบบ คือราคาถูก ราคาปานกลาง ราคาแพง และราคาระดับหรู เพื่อดูว่า ในแต่ละพื้นที่นั้นราคาที่พักเป็นอย่างไรโดยจะค้นพบว่า ในพื้นที่รอบแลนด์มาร์คของประเทศอเมริกา เช่นในตัวเมืองนิวยอร์กนั้น จะมี ราคาที่พักอยู่ในราคาแพง และะราคาระดับหรูเท่านั้น

Mohamed Mahyoub and Others (2023) ได้ทำการวิจัยเรื่องทำนายราคาที่พักใน Airbnb ในเมืองลอนดอน (Mahypub, Ataby, Upadhyay, และ Mustafina, 2023) โดยข้อมูลที่ใช้จะเป็นข้อมูลที่พักAirbnb ในลอนดอน อัพเดทล่าสุดเมื่อวันที่ 7 ธันวาคม พ.ศ.2564 โดยจำนวนข้อมูลทั้งหมดจะมี 66641 ชุด เทคนิค ทั้งหมด 4 แบบ ป่าไม้แบบสุ่ม, XGBoost, การถดถอยเชิงเส้นตรง

และ กราเดี้ยนบูสติ้ง โดยจะได้ผลลัพธ์ที่ว่า ตัวแบบ ป่าไม้แบบสุ่ม ทำผลงานได้ดีที่สุดโดยมี ค่า r-squared อยู่ที่ 0.86944 ตามมาด้วยตัวแบบ XGBoost ที่ 0.86601 ซึ่งตรงกับการศึกษาของ (Yang, 2021) ที่ทำการทำนายราคาที่พักในปักกิ่ง โดยใช้วิธี XGBoost โครงข่ายประสาทเทียม โดย ตัวแบบ XGBoost ก็มีผลลัพธ์ที่ดีกว่าโดยค่า r-squared อยู่ที่ 0.8112 และ 0.6549 ของชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบ

Yihao Chen and Others (2021) ได้กล่าวไว้การทำนายราคาอสังหาริมทรัพย์ อย่างบ้านได้กลายมาเป็น 1 ในเทรนด์สำคัญในปัจจุบัน โดยเฉพาะการนำการเทคโนโลยีเข้ามาช่วยในการทำนาย ไม่ว่าจะเป็น การเรียนรู้ของเครื่องหรือ การเรียนรู้ของเครื่องเชิงลึก จากการนำการเรียนรู้ของเครื่องเข้ามาใช้ในการทำนายโดยใช้เทคนิคที่ดังและเทคนิคเชิงลึก มาใช้ โดยจะมี การถดถอยเชิงเส้น, Naïve Bayesian, Back Propagation Neural Network, Support Vector Machine, Deep Neural Network โดยจากผลลัพธ์จะได้ว่า การถดถอยเชิงเส้น จะมีผลลัพธ์ที่แย่ที่สุดเมื่อเทียบกับตัวแบบอื่น โดย SVM จะมีประสิทธิภาพที่ดีที่สุดในตัวแบบทั้งหมด

Jibrin Katun Mohammed and Others (2021) ได้ทำการวิจัยเรื่อง ผลกระทบของโควิด 19 ที่มีผลต่อตลาดราคาบ้าน (Mohammed , Aliyu, Dzukog, และ Olawale, 2021) โดยได้ทำการกล่าวถึงผลกระทบของโควิด 19 ที่มีต่อ ราคาบ้านว่า ในตลาดราคาที่พักและบ้านตอนนี้นั้น มีผลกระทบทั้งในทางบวกและลบ โดย ผลกระทบจะเกิดขึ้นจากการที่ที่พักและบ้านได้ถูกปรับเปลี่ยนจากการรองรับนักท่องเที่ยวกลายเป็น รองรับผู้คนท้องถิ่นแทน โดยมีความต้องการบ้านที่มีสิ่งอำนวยความสะดวกครบครันในแถบชนบทเพิ่มมากขึ้น ในขณะที่ มีผลกระทบทางลบมากมายที่เกิดขึ้นกับตลาดราคาบ้าน ไม่ว่าจะเป็นราคาที่ตกลง อุปสงค์ อุปทานที่แย่ลง การค้างจำนอง หรือ การก่อสร้างที่ล่าช้าจากการล็อกดาวน์ของโควิด 19

V. Raul Perez-Sanchez and Others (2018) ได้ทำการศึกษาวิจัยเรื่อง The What, Where, and Why of Airbnb Price Determinants หรือ ปัจจัยที่มีผลต่อราคาของที่พักใน Airbnb (Perez-Sanchez, Serrano-Estrada, & Marti, 2018) ได้ทำการค้นพบ 5 ปัจจัย ที่ส่งผลต่อราคาที่พักใน Airbnb คือ ราคา,ลักษณะที่พัก (attribute), โฆษณา, สภาพแวดล้อมของที่พัก และ โลเคชันของที่พัก จากงานวิจัย ได้มีการศึกษามาว่า โลเคชันที่พักนั้นมีผลกระทบต่อราคาของที่พักโดยเฉพาะที่พักที่ใกล้กับแหล่งท่องเที่ยวหรือแลนด์มาร์กของเมืองนั้นๆ โดยเฉพาะระยะทางที่ยิ่งใกล้กับแหล่งท่องเที่ยวมากเท่าไร ราคาที่พักก็จะเพิ่มสูงขึ้นมากเท่านั้น

Tarik Dogru and Osman Pekin (2017) ได้ทำการศึกษาวิจัยเรื่อง อะไรที่ผู้เข้าพักให้ค่ามากที่สุดเวลาเข้าพักที่พักของAirbnb (Tarik & Pekin, 2017) โดยได้ทำการค้นพบว่า ประเภทของที่พักที่เป็นบ้านทั้งหลังหรือ เป็นห้องส่วนตัว จะมีราคาที่สูงกว่าห้องแบบแชร์ อยู่ที่ 141% and 28%

ตามลำดับ แล้วสถานะของเจ้าของที่พักก็มีผลกระทบต่อราคาของที่พักโดยหาสถานะของเจ้าของที่พักเป็น superhost จะมีราคาสูงกว่าประมาณ 5% เมื่อเทียบเจ้งของที่พักที่ไม่ได้มีสถานะ superhost และ ผู้เข้าพักยังให้ความสำคัญกับเรื่อง พื้นที่, ความสะอาด, จำนวนรูปภาพ, โลเคชัน, และประสบการณ์ใหม่ๆ

บทที่ 3

วิธีการดำเนินงานวิจัย

งานวิจัยนี้ทำการศึกษาการทำนายราคาที่พักในกรุงเทพมหานครโดยประยุกต์การเรียนรู้ของเครื่องและงานวิจัยที่เกี่ยวข้องมากำหนดขั้นตอนในการศึกษาดังนี้

3.1 การรวบรวมข้อมูล

การทำนายราคาที่พักในกรุงเทพมหานครโดยประยุกต์การเรียนรู้ของเครื่อง โดยการนำข้อมูลของที่พักใน Airbnb ในจังหวัดกรุงเทพมหานคร ประเทศไทย จากเว็บไซต์ www.insideairbnb.com โดยเป็นข้อมูลอัพเดทล่าสุดเมื่อวันที่ 22 กันยายน พ.ศ. 2566 โดยจำนวนข้อมูลจะมีทั้งหมด 20823 ตัวอย่างและ 75 ตัวแปร ต่อมาได้มีขั้นตอนในการจัดการกับข้อมูลเพื่อพร้อมสำหรับการทำนายราคาด้วยการเรียนรู้ของเครื่องดังนี้

ตารางที่ 3.1 Features ทั้งหมดของชุดข้อมูล

ลำดับที่	คอลัมน์	ลำดับที่	คอลัมน์
1	id	10	host_id
2	listing_url	11	host_url
3	scrape_id	12	host_name
4	last_scraped	13	host_since
5	source	14	host_location
6	name	15	host_about
7	description	16	host_response_time
8	neighborhood_overview	17	host_response_rate
9	picture_url	18	host_acceptance_rate

ตารางที่ 3.1 Features ทั้งหมดของชุดข้อมูล (ต่อ)

ลำดับที่	คอลัมน์	ลำดับที่	คอลัมน์
19	host_is_superhost	36	bathrooms
20	host_thumbnail_url	37	bathrooms_text
21	host_picture_url	38	bedrooms
22	host_neighbourhood	39	beds
23	host_listings_count	40	amenities
24	host_total_listings_count	41	price
25	host_verifications	42	minimum_nights
26	host_has_profile_pic	43	maximum_nights
27	host_identity_verified	44	minimum_minimum_nights
28	neighbourhood	45	maximum_minimum_nights
29	neighbourhood_cleansed	46	minimum_maximum_nights
30	neighbourhood_group_cleansed	47	maximum_maximum_nights
31	latitude	48	minimum_nights_avg_ntm
32	longitude	49	maximum_nights_avg_ntm
33	property_type	50	calendar_updated
34	room_type	51	has_availability
35	accommodates	52	availability_30

ตารางที่ 3.1 Features ทั้งหมดของชุดข้อมูล (ต่อ)

ลำดับที่	คอลัมน์	ลำดับที่	คอลัมน์
53	availability_60	65	review_scores_checkin
54	availability_90	66	review_scores_communication
55	availability_365	67	review_scores_location
56	calendar_last_scraped	68	review_scores_value
57	number_of_reviews	69	license
58	number_of_reviews_ltm	70	instant_bookable
59	number_of_reviews_l30d	71	calculated_host_listings_count
60	first_review	72	calculated_host_listings_count_entire_homes
61	last_review	73	calculated_host_listings_count_private_rooms
62	review_scores_rating	74	calculated_host_listings_count_shared_rooms
63	review_scores_accuracy	75	reviews_per_month
64	review_scores_cleanliness	72	calculated_host_listings_count_entire_homes

3.2 การจัดการข้อมูล

3.2.1 การคลีนข้อมูล

ก. ทำการตัวแปรที่ไม่จำเป็นออกไป 32 ตัวแปร แล้วข้อมูลจะเหลือ 43 ตัวแปรและ 20823 ตัวอย่าง

ข. ทำการนำข้อมูล n/a ของตัวแปร host response time และ host acceptance rate ออก

ค. ทำการลบข้อมูล ว่าง ของ ตัวแปร host is superhost

ง. ทำการลบข้อมูล ว่าง ของ ตัวแปร bed , review rating, bedroom, email, phone, work_email

จ. ทำการ drop duplicate

ฉ. ทำการ drop missing value

ช. ทำการคำนวณ correlation matrix

ซ. นำคอลัมน์ที่มีค่า correlation สูงและคอลัมน์ที่ไม่เกี่ยวข้องออก

3.2.2 การแปลงข้อมูลเชิงปริมาณ

ก. ทำการ standardize คอลัมน์ที่เป็นตัวเลขทั้งหมด

ข. ทำการนำ outlier ออกจากคอลัมน์ ราคา (ตัวแปรอิสระ)

ค. ทำการ log transformation คอลัมน์ ราคา

ง. ทำการแทนที่ % ในคอลัมน์ host response rate/ host acceptance rate ด้วยช่องว่าง แล้วนำค่าไปหารด้วย 100

จ. ทำการนำคอลัมน์ property_type และ room_type ออก

3.2.3 การแปลงข้อมูลเชิงคุณภาพ

ก. ทำการแปลง คอลัมน์ verify 1 คอลัมน์ เป็น 3 คอลัมน์ ใหม่ที่มีชื่อตามคำตอบของคอลัมน์เดิม (email, phone, work_email)

ข. ทำการดึงค่าในคอลัมน์ amenities เฉพาะ amenities ที่มีจำนวนเกินครึ่งของจำนวนชุดข้อมูล

ค. ทำการแบ่งประเภทของค่าที่ได้จากคอลัมน์ amenities เป็น 4 ประเภท

i.Comfort & Basics

ii.Appliances & Technology

iii.Safety & Facilities

iv.Convenience

ง. ทำการดึงค่า unique value จาก คอลัมน์ property_type และแบ่งประเภทออกเป็น 4 ประเภท

i.entire_units

ii.rooms_shared

iii.private_rooms

iv.specialty_accomodation

จ. one hot dummy คอลัมน์ที่มีคำตอบเป็นกลุ่ม

3.2.4 การเพิ่ม ตัวแปรจาก source อื่นๆ

ก. ทำการเพิ่ม คอลัมน์ tourist district เข้าไป โดย จะเป็นคอลัมน์ที่มีค่า คือ Tourist Distirct และ Non-tourist distirct โดย จะมีหลักเกณฑ์ว่าที่พักที่อยู่ที่ใกล้แหล่งท่องเที่ยวและแลนด์มาร์กของกรุงเทพจะได้คำตอบเป็น Tourist District

i. Chatu Chak

ii.Phra Nakhon

iii.Samphanthawong

iv.Pra Wet

v.Parthum Wan

vi.Bang Rak

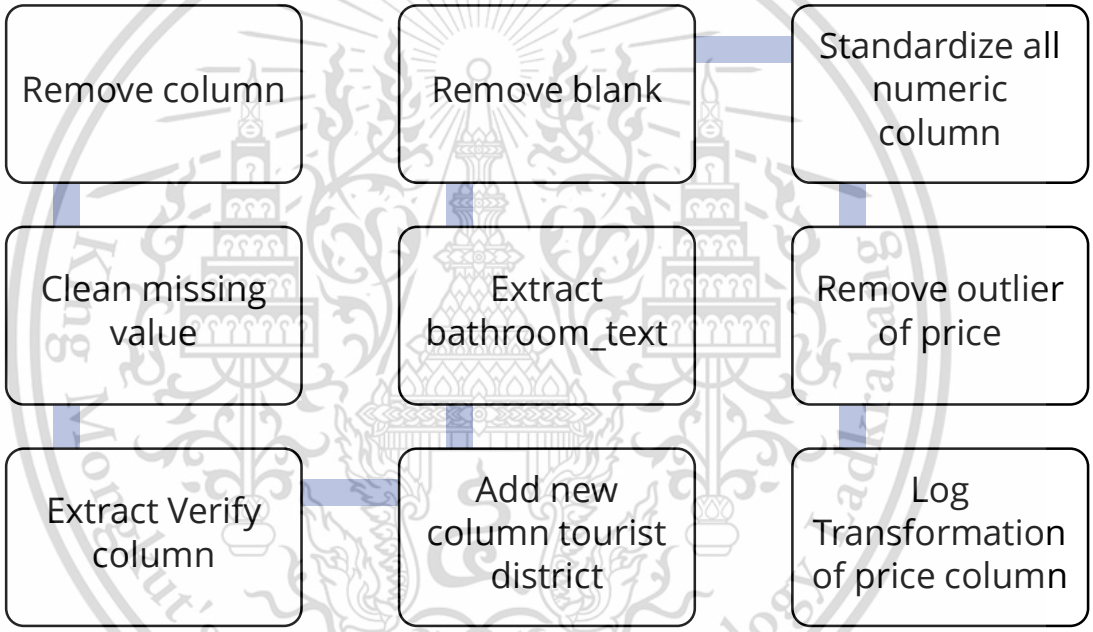
vii.Vadhana

viii.Ratchathewi

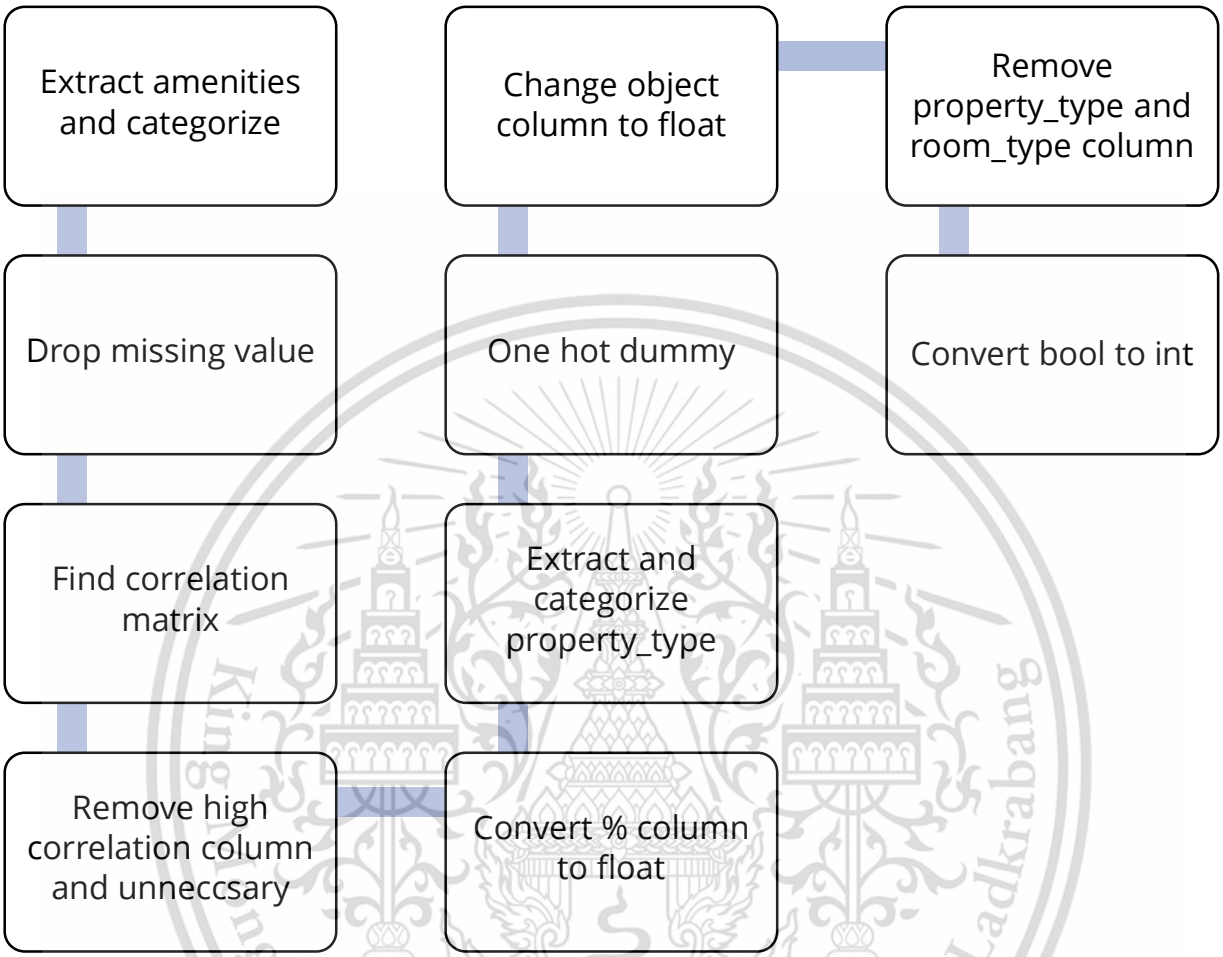
ix.Khlong Toei

x.Klong San

หลังผ่านกระบวนการจัดการข้อมูลแล้ว จะเหลือข้อมูลทั้งหมด 7924 แถว และ 29 คอลัมน์



รูปที่ 3.1 ผังงานแสดงกระบวนการจัดการข้อมูล ส่วนที่ 1



รูปที่ 3.2 ผังงานแสดงกระบวนการจัดการข้อมูล ส่วนที่ 2

ตารางที่ 3.2 ข้อมูลหลังผ่านกระบวนการจัดการข้อมูล

Variable Name	Type of Variable	Description
host_response_rate	Quality	เปอร์เซ็นต์การตอบสนองของเจ้าของที่พัก
host_acceptance_rate	Quality	เปอร์เซ็นต์การตอบตกลงการเข้าพักของเจ้าของที่พัก
host_total_listings_count	Quality	จำนวนที่พักที่เจ้าของมีใน Airbnb
email	Quantity	มีช่องทางการติดต่อด้วยemail 1 = มี 0 = ไม่มี
phone	Quantity	มีช่องทางการติดต่อด้วย phone 1 = มี 0 = ไม่มี
work_email	Quantity	มีช่องทางการติดต่อด้วย Work email 1 = มี 0 = ไม่มี

ตารางที่ 3.2 ข้อมูลหลังผ่านกระบวนการจัดการข้อมูล (ต่อ)

Variable Name	Type of Variable	Description
accommodates	Quality	จำนวนคนสูงสุดที่รองรับได้
bedrooms	Quality	จำนวนห้องนอน
price	Quality	ราคาที่พัก
minimum_nights	Quality	จำวนคืนขั้นต่ำในการเข้าพัก
maximum_nights	Quality	จำนวนคืนสูงสุดในการเข้าพัก
availability_30	Quality	จำนวนวันว่างในช่วง30วัน
number_of_reviews	Quality	จำนวนรีวิว
review_scores_rating	Quality	คะแนนของการรีวิว
Comfort & Basics	Quantity	ประเภทของสิ่งอำนวยความสะดวก(ของใช้พื้นฐาน) 1 = มี 0 = ไม่มี

ตารางที่ 3.2 ข้อมูลหลังผ่านกระบวนการจัดการข้อมูล (ต่อ)

Variable Name	Type of Variable	Description
Appliances & Technology	Quantity	ประเภทของสิ่งอำนวยความสะดวก(เครื่องใช้และเทคโนโลยี) 1 = มี 0 = ไม่มี
Safety & Facilities	Quantity	ประเภทของสิ่งอำนวยความสะดวก(ความปลอดภัยและของอำนวยความสะดวก 1 = มี 0 = ไม่มี
Convenience	Quantity	ประเภทของสิ่งอำนวยความสะดวก(สิ่งอำนวยความสะดวก) 1 = มี 0 = ไม่มี
host_has_profile_pic_t	Quantity	โฮสมีรูปภาพโปรไฟล์ 1 = มี 0 = ไม่มี

ตารางที่ 3.2 ข้อมูลหลังผ่านกระบวนการจัดการข้อมูล (ต่อ)

Variable Name	Type of Variable	Description
host_identity_verified_t	Quantity	โฮสยืนยันตัวตน 1 = ยืนยัน 0 = ไม่ยืนยัน
host_is_superhost_t	Quantity	โฮสเป็นซุปเปอร์โฮส 1 = เป็น 0 = ไม่เป็น
host_response_time_within a day	Quantity	โฮสตอบสนองภายใน 1 วัน 1 = ใช่ 0 = ไม่
host_response_time_within a few hours	Quantity	โฮสตอบสนองภายใน ไม่กี่ชั่วโมง 1 = ใช่ 0 = ไม่
host_response_time_within an hour	Quantity	โฮสตอบสนองภายใน 1 ชั่วโมง 1 = ใช่ 0 = ไม่

ตารางที่ 3.2 ข้อมูลหลังผ่านกระบวนการจัดการข้อมูล (ต่อ)

Variable Name	Type of Variable	Description
Tourist District_tourist distirct	Quantity	เขตท่องเที่ยว 1 = เขตท่องเที่ยว 0 = ไม่ใช่เขตท่องเที่ยว
instant_bookable_t	Quantity	จองได้ทันที 1 = ใช่ 0 = ไม่
category_property_Private Room	Quantity	ห้องเดี่ยว 1 = เป็นห้องเดี่ยว 0 = ไม่เป็นห้องเดี่ยว
category_property_Room or Shared Space	Quantity	แชร์รูม 1 = เป็นแชร์รูม 0 = ไม่เป็นแชร์รูม
category_property_Specialty property_type	Quantity	ห้องแบบพิเศษ 1 = เป็นห้องแบบพิเศษ 0 = ไม่เป็นห้องแบบพิเศษ

3.3 เครื่องมือที่ใช้

ตารางที่ 3.3 เครื่องมือที่ใช้

เครื่องมือ	คำอธิบาย
Microsoft Excel	เครื่องมือในการจัดการข้อมูลและสร้างกราฟ
Pandas	Library Python สำหรับใช้ในการจัดการข้อมูล
Numpy	Library Python สำหรับใช้การคำนวณทางสถิติ
Matplotlib	Library Python สำหรับใช้ในการทำdata visualization
Scikit learn	Library Python สำหรับใช้ในการสร้างตัวแบบการทำนายเช่น XGBoost /SVM / Linear Regression
re	Library Python สำหรับใช้สำหรับการหา regular expression
shap	Library Python สำหรับใช้ในการอธิบายผลลัพธ์ของการเรียนรู้ของเครื่อง
ast	Library Python สำหรับใช้ในการจัดการกับ python code

3.4 การแบ่งข้อมูลเป็นชุดข้อมูลเรียนรู้ (training set) และชุดข้อมูลทดสอบ (test set)

ทำการแบ่งข้อมูลที่ได้เป็นข้อมูลเรียนรู้ (train set)และข้อมูลทดสอบ (test set) โดยจะแบ่งข้อมูล 80% เป็นชุดข้อมูลเรียนรู้และ 20% เป็นชุดข้อมูลทดสอบ โดยชุดข้อมูลเรียนรู้จะไว้ทำการเรียนรู้ตัวแบบหรือแบบจำลอง และชุดข้อมูลทดสอบสำหรับการหาประสิทธิภาพของแบบจำลองที่ผ่านการเรียนรู้

3.5 การทำนายราคาที่พัก

3.5.1 XGBoost

มีการกำหนดค่า hyperparameter เพื่อให้ได้แบบจำลองที่มีประสิทธิภาพมากที่สุด โดย hyperparameterที่ใช้จะมีดังนี้

1. alpha : ไว้ใช้ควบคุม L1 regularization โดยจะทำการตัด features ที่มีความสำคัญน้อยออก ยิ่งมีค่ามาก ค่า coefficient ยิ่งเข้าใกล้ 0 [0, 1, 5, 10, 15]

2. colsample_bytree : ควบคุมส่วนแบ่งของ features ที่ถูกเลือกในแต่ละต้นไม้ เพื่อป้องกันoverfitting และ เพิ่มความสุ่มให้แก่โมเดล ค่าอยู่ในช่วง 0 ถึง 1 [0.1, 0.3, 0.5, 0.7, 0.9]

3. learning_rate : อัตราการเรียนรู้ไว้ป้องกัน overfitting [0.001, 0.01, 0.05, 0.1, 0.2]

4. max_depth : ความลึกสูงสุดของต้นไม้ โดยการเพิ่มค่าจะทำให้ตัวแบบมีความซับซ้อนมากขึ้น หากค่ามีมากเกินไปจะเกิดปัญหา overfitting หากค่าต่ำเกินไป จะเกิดปัญหา underfitting [3, 5, 7, 10, 15]

5. n_estimators : จำนวนต้นไม้ โดยจำนวนต้นไม้เยอะ จะเพิ่มประสิทธิภาพการทำงานของตัวแบบ ซึ่งอาจตามมาด้วยปัญหา overfitting [50, 100, 200, 300, 500]

แล้วทำการหาค่า hyperparameter ที่ดีที่สุดโดยใช้ gridsearchcv ในการค้นหา

ตารางที่ 3.4 ค่า hyperparameterที่ดีที่สุดจากการใช้ gridsearchcv ของ XGBoost

Hyperparameter	Value	Best hyperparameter
alpha	[0, 1, 5, 10, 15]	0
colsample_bytree	[0.1, 0.3, 0.5, 0.7, 0.9]	0.5
learning_rate	[0.001, 0.01, 0.05, 0.1, 0.2]	0.05
max_depth	[3, 5, 7, 10, 15]	10
n_estimators	[50, 100, 200, 300, 500]	500

นำค่า hyperparameter ที่ได้ มารันตัวแบบ XGBoost หาfeature importance ของแต่ละตัวแปร แล้วจะทำการแบ่งออกเป็น 3 กรณีคือ

1. Filtered feature (Importance > 0)
2. ตัวแปรที่มีค่า importance มากที่สุด 5 อันดับแรก (Top 5)
3. ตัวแปรที่มีค่า importance มากที่สุด 10 อันดับแรก (Top 10)

3.5.2 ซัพพอร์ตเวกเตอร์แมชชีน(SVM)

มีการกำหนดค่า hyperparameter ที่จะใช้ดังนี้

1. C: ค่า hyperparmeter สำหรับกำหนดระดับของ slack variable [0.1, 1, 10, 100],
2. Gamma: ปรับความชันของ kernel function [scale, auto]

- Scale $\gamma = \frac{1}{n * var(x)}$ (3.1)
- Auto $\gamma = \frac{1}{n}$ (3.2)

3. Kernel: kernel function [rbf, linear]

ทำการค้นหา hyperparameter ที่ดีที่สุดโดยใช้ gridsearchcv

ตารางที่ 3.5 ค่า hyperparameterที่ดีที่สุดจากการใช้ gridsearchcv ของ SVM

Hyperparameter	Value	Best hyperameter
C	[0.1, 1, 10, 100]	1
gamma	[scale, auto]	scale
kernel	[rbf, linear]	rbf

ทำการสร้างตัวแบบ 3 กรณี

- Filtered feature (Importance > 0)
- ตัวแปรที่มีค่า importance มากที่สุด 5 อันดับแรก (Top 5)
- ตัวแปรที่มีค่า importance มากที่สุด 10 อันดับแรก (Top 10)

แล้วทำการวัดประสิทธิภาพของตัวแบบด้วย ค่า MAE RMSE และ R-squared

3.5.3 การถดถอยเชิงเส้น

ทำการนำข้อมูลที่ผ่านกระบวนการ preprocessing มาใช้วิธี OLS (Ordinary Least Square) เพื่อการประมาณค่า แต่ละ feature ในสมการการถดถอยเชิงเส้น โดยทำการแยกความสำคัญของ feature โดยใช้ ค่า p-value เป็นตัวตัดสิน โดยกำหนดระดับ ความเชื่อมั่นไว้ที่ 95 % (α = 0.05)

$$H_0: \beta_n = 0$$

$$H_1: \beta_n \neq 0$$

โดยหากค่า p-value ของ feature ≤ α จะทำการปฏิเสธ H_0 หาก p-value > α จะทำการยอมรับ H_0

โดยจะเลือก feature มีค่า p-value น้อยกว่า 0.05 มาทำการสร้างตัวแบบการถดถอยเชิงเส้น

บทที่ 4

ผลการวิจัยและอภิปรายผล

ผลการศึกษาการทำนายราคาที่พักในกรุงเทพโดยประยุกต์การเรียนรู้ของเครื่อง โดยใช้ข้อมูลจากเว็บไซต์ insideairbnb อัพเดทล่าสุดเมื่อวันที่ 22 กันยายน พ.ศ.2566 เพื่อทำการศึกษาปัจจัยที่ส่งผลต่อราคาที่พักในกรุงเทพและเปรียบเทียบประสิทธิภาพการทำนายราคาที่พักโดยใช้การเรียนรู้ของเครื่อง ระหว่าง วิธีเอ็กจีบูส (XGBoost), ซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machine) และวิธีการถดถอยเชิงเส้น(Linear Regression)

4.1 การวัดประสิทธิภาพของตัวแบบ XGBoost

4.2 การวัดประสิทธิภาพของตัวแบบ SVM

4.3 การวัดประสิทธิภาพของตัวแบบการถดถอยเชิงเส้น

4.4 การวัดประสิทธิภาพระหว่างตัวแบบ

4.1 การวัดประสิทธิภาพของตัวแบบ XGBoost

เริ่มจากการหาค่า feature importance ของตัวแบบ XGBoost ทั้ง 3 กรณี

- Filtered feature (Importance > 0)
- ตัวแปรที่มีค่า importance มากที่สุด 5 อันดับแรก (Top 5)
- ตัวแปรที่มีค่า importance มากที่สุด 10 อันดับแรก (Top 10)

จากการหาค่า feature importance จะได้ ตัวแปรที่มีค่า importance > 0 ของตัวแบบ XGBoost ตามตารางที่ 4.1

ตารางที่ 4.1 ค่า feature importance ของ XGBoost (Filtered feature)

Feature	Feature Importance
host_response_rate	0.0158
host_acceptance_rate	0.0216
host_total_listings_count	0.0324
email	0.0212
phone	0.0329
work_email	0.0205
accommodates	0.0652
bedrooms	0.1912
minimum_nights	0.0193
maximum_nights	0.0209
availability_30	0.0177
number_of_reviews	0.0189
review_scores_rating	0.0147
Comfort & Basics	0.0191
Appliances & Technology	0.0153
Safety & Facilities	0.0403
Convenience	0.0208
host_has_profile_pic_t	0.0182

ตารางที่ 4.1 ค่า feature importance ของ XGBoost (Filtered feature) (ต่อ)

Feature	Feature Importance
host_identity_verified_t	0.0278
host_is_superhost_t	0.0227
host_response_time_within a day	0.0294
host_response_time_within a few hours	0.0240
host_response_time_within an hour	0.0204
Tourist District_tourist distirct	0.1513
instant_bookable_t	0.0199
category_property_Private Room	0.0555
category_property_Room or Shared Space	0.0367
category_property_Specialty property_type	0.0065

ตารางที่ 4.2 ค่า feature importance ของ XGBoost (Top 5)

Feature	Feature Importance
bedrooms	0.1912
Tourist District_tourist distirct	0.1513
accommodates	0.0652
category_property_Private Room	0.0555
Safety & Facilities	0.0403

จากการหา top 5 ตัวแปรที่มีค่า importance สูงที่สุดของตัวแบบ XGBoost พบว่าประกอบไปด้วยตัวแปรดังต่อไปนี้ bedrooms, Tourist District_tourist district, accommodates, category_property_Private Room และ Safety & Facilities

ตารางที่ 4.3 ค่า feature importance ของ XGBoost (Top 10)

Feature	Feature Importance
bedrooms	0.1912
Tourist District_tourist distirct	0.1513
accommodates	0.0652
category_property_Private Room	0.0555
Safety & Facilities	0.0403
category_property_Room or Shared Space	0.0065
phone	0.0329
host_total_listings_count	0.0324
host_response_time_within a day	0.0294
host_identity_verified_t	0.0278

จากการหา top 10 ตัวแปรที่มีค่า importance สูงที่สุดของตัวแบบ XGBoost พบว่าประกอบไปด้วยตัวแปรดังต่อไปนี้ bedrooms, Tourist District_tourist district, accommodates, category_property_Private Room, Safety & Facilities category_property_Room or Shared Space, phone, host_total_listings_count, host_response_time_within a day, host_identity_verified_t

แล้วทำการหาค่า hyperparameter ที่เหมาะสมกับแต่ละตัวแบบในแต่ละกรณี

ตารางที่ 4.4 ค่า hyperparameter ของ XGBoost (Filtered feature)

Hyperparameter	Value	Best hyperparameter
alpha	[0, 1, 5, 10, 15]	0
colsample_bytree	[0.1, 0.3, 0.5, 0.7, 0.9]	0.5
learning_rate	[0.001, 0.01, 0.05, 0.1, 0.2]	0.05
max_depth	[3, 5, 7, 10, 15]	10
n_estimators	[50, 100, 200, 300, 500]	500

ในกรณีที่ เลือก feature ที่มีค่า importance > 0 สำหรับวิธี XGBoost ได้ค่า hyperparameter ที่เหมาะสมที่สุดดังนี้ alpha = 0, colsample_bytree = 0.5, learning_rate = 0.05, max_depth = 10 และ n_estimators = 500

ตารางที่ 4.5 ค่า hyperparameter ของ XGBoost (Top 5)

Hyperparameter	Value	Best hyperparameter
alpha	[0, 1, 5, 10, 15]	0
colsample_bytree	[0.1, 0.3, 0.5, 0.7, 0.9]	0.5
learning_rate	[0.001, 0.01, 0.05, 0.1, 0.2]	0.05
max_depth	[3, 5, 7, 10, 15]	10
n_estimators	[50, 100, 200, 300, 500]	500

ในกรณีที่ เลือก feature ที่มีค่า importance มากที่สุด 5 อันดับสำหรับวิธี XGBoost ได้ค่า hyperparameter ที่เหมาะสมที่สุดดังนี้ alpha = 0, colsample_bytree = 0.5, learning_rate = 0.05, max_depth = 10 และ n_estimators = 500

ตารางที่ 4.6 ค่า hyperparameter ของ XGBoost (Top 10)

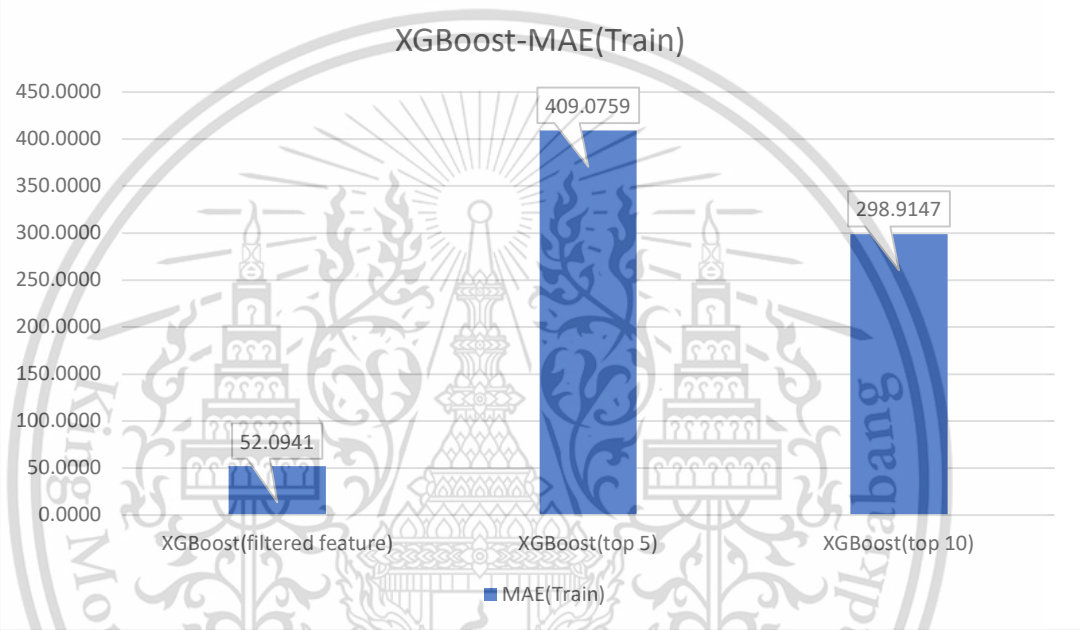
Hyperparameter	Value	Best Hyperparameter
alpha	[0, 1, 5, 10, 15]	0
colsample_bytree	[0.1, 0.3, 0.5, 0.7, 0.9]	0.5
learning_rate	[0.001, 0.01, 0.05, 0.1, 0.2]	0.05
max_depth	[3, 5, 7, 10, 15]	10
n_estimators	[50, 100, 200, 300, 500]	500

ในกรณีที่ เลือก feature ที่มีค่า importance มากที่สุด 10 อันดับสำหรับวิธี XGBoost ได้ค่า hyperparameter ที่เหมาะสมที่สุดดังนี้ alpha = 0, colsample_bytree = 0.5, learning_rate = 0.05, max_depth = 10 และ n_estimators = 500

ทำการวัดประสิทธิภาพของตัวแบบ XGBoost ผ่านค่า MAE, RMSE และ R^2 โดยแบ่งข้อมูลที่วัดประสิทธิภาพเป็นข้อมูลชุดเรียนรู้และข้อมูลชุดทดสอบ

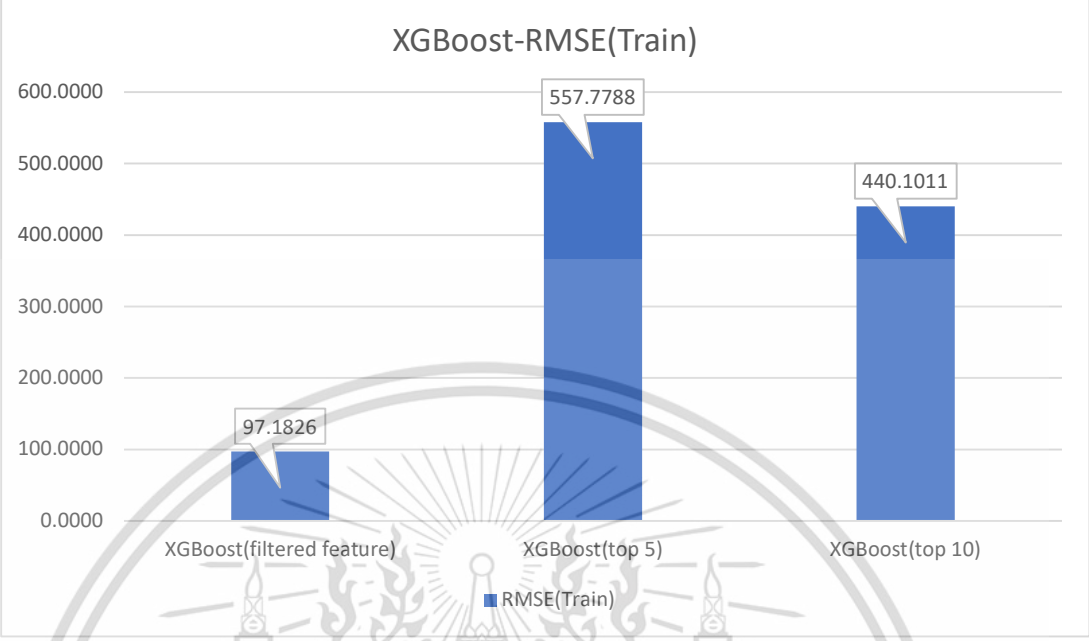
ข้อมูลชุดเรียนรู้

ผลการทดสอบตัวแบบ XGBoost ของชุดข้อมูลเรียนรู้โดยใช้ค่า MAE, RMSE และ R^2 ในการวัดประสิทธิภาพของตัวแบบ XGBoost ทั้ง 3 ตัวแบบ (filtered feature/ top 5/ top 10) โดยเป็นไปตามรูปที่ 4.1, 4.2 และ 4.3



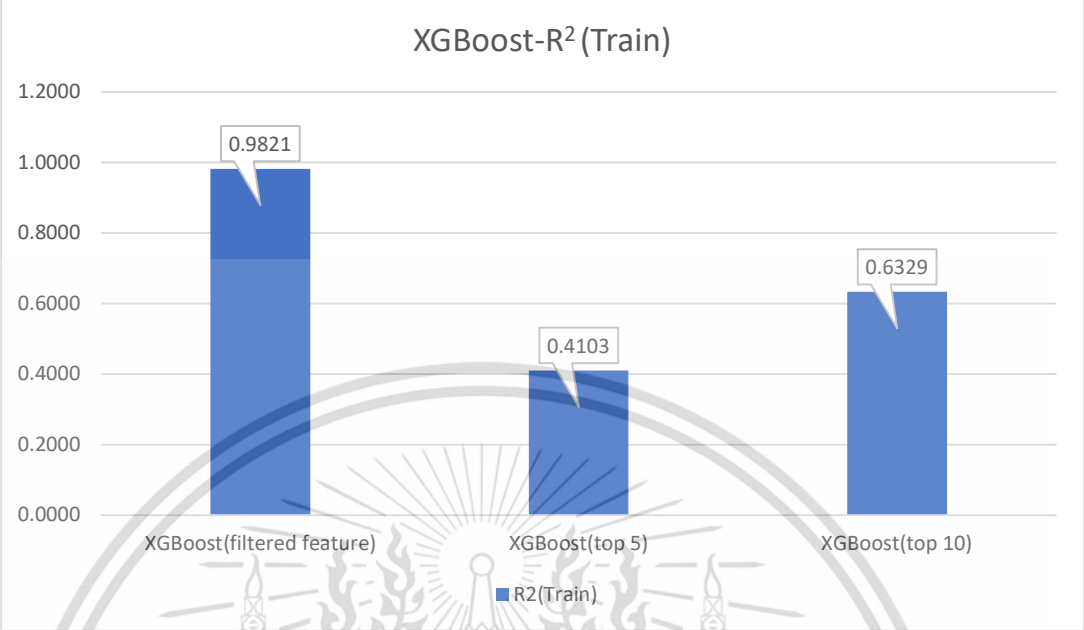
รูปที่ 4.1 แผนภูมิแท่งแสดงค่า MAE ของชุดข้อมูลเรียนรู้ของตัวแบบ XGBoost

จากรูปที่ 4.1 จะพบว่าตัวแบบ XGBoost (filtered feature) จะมีค่า MAE อยู่ที่ 52.0941 โดย XGBoost (top 5) จะมีค่า MAE อยู่ที่ 409.0759และ ตัวแบบ XGBoost (top 10) มีค่า MAE อยู่ที่ 298.9147



รูปที่ 4.2 แผนภูมิแท่งแสดงค่า RMSE ของชุดข้อมูลเรียนรู้ของตัวแบบ XGBoost

จากรูปที่ 4.2 จะพบว่าตัวแบบ XGBoost (filtered feature) จะมีค่า RMSE อยู่ที่ 97.1826 โดย XGBoost (top 5) จะมีค่า RMSE อยู่ที่ 557.7788 และ ตัวแบบ XGBoost (top 10) มีค่า RMSE อยู่ที่ 440.1011

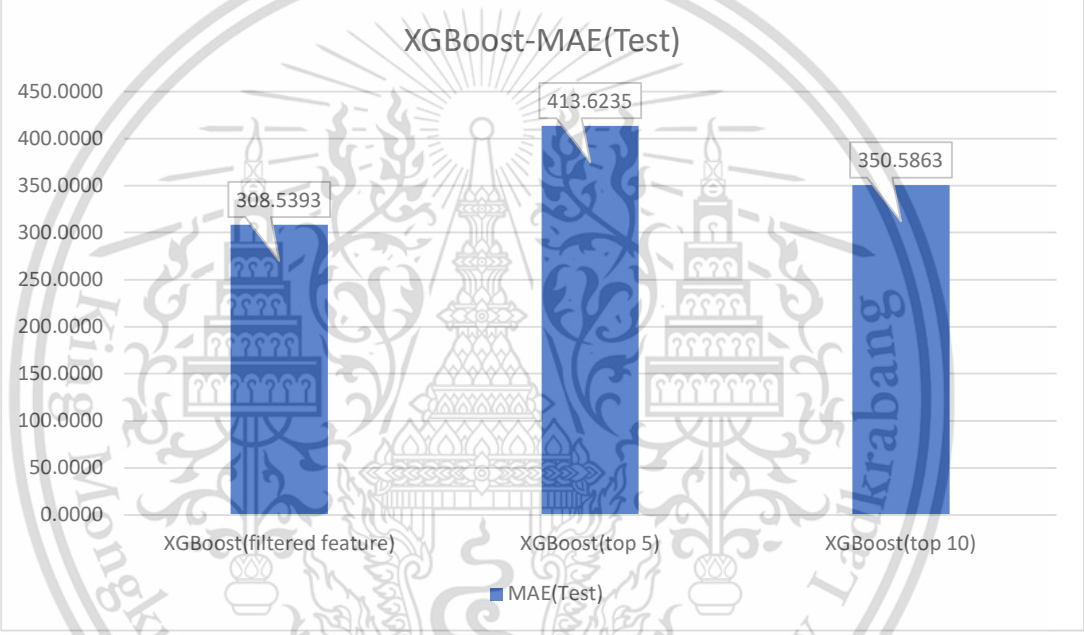


รูปที่ 4.3 แผนภูมิแท่งแสดงค่า R^2 ของชุดข้อมูลเรียนรู้ของตัวแบบ XGBoost

จากรูปที่ 4.3 จะพบว่าตัวแบบ XGBoost (filtered feature) จะมีค่า R^2 อยู่ที่ 0.982099 โดย XGBoost (top 5) จะมีค่า R^2 อยู่ที่ 0.410336 และ ตัวแบบ XGBoost (top 10) มีค่า R^2 อยู่ที่ 0.632899

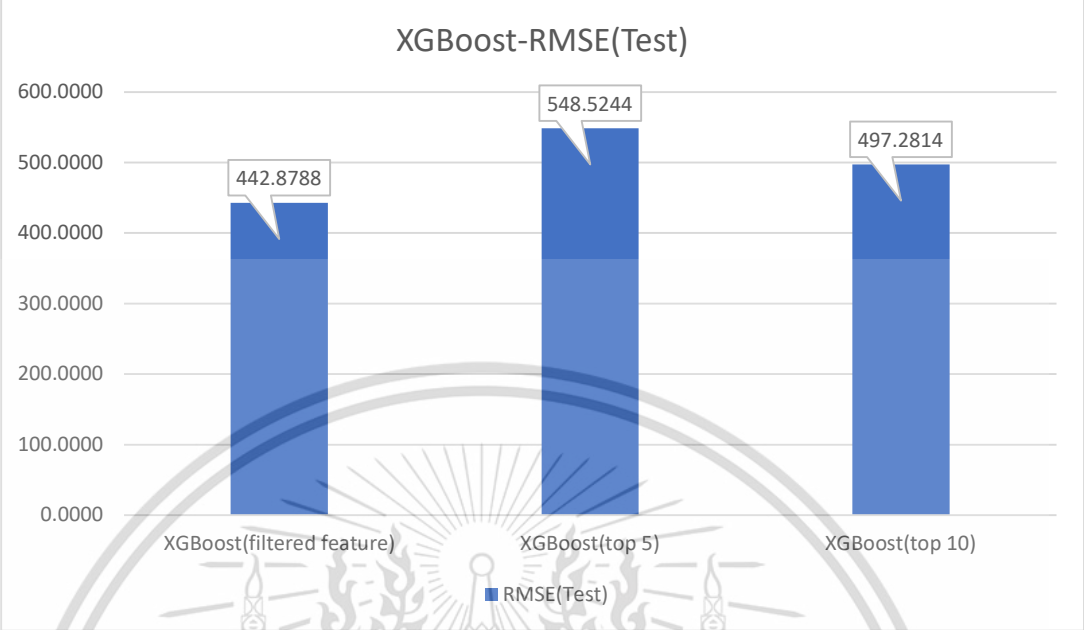
ข้อมูลชุดทดสอบ

ผลการทดสอบตัวแบบ XGBoost ของชุดข้อมูลทดสอบโดยใช้ค่า MAE, RMSE และ R^2 ในการวัดประสิทธิภาพของตัวแบบ XGBoost ทั้ง 3 ตัวแบบ (filtered feature/ top 5/ top 10) โดยเป็นไปตามรูปที่ 4.4, 4.5 และ 4.6



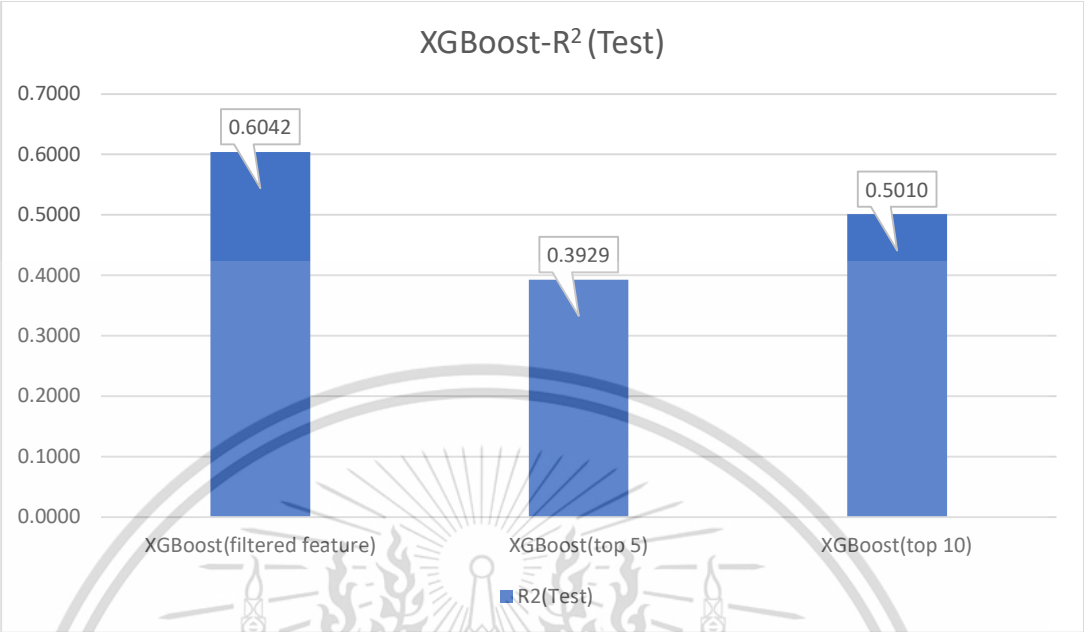
รูปที่ 4.4 แผนภูมิแท่งแสดงค่า MAE ของชุดข้อมูลทดสอบของตัวแบบ XGBoost

จากรูปที่ 4.4 จะพบว่าตัวแบบ XGBoost (filtered feature) จะมีค่า MAE อยู่ที่ 308.5393 โดย XGBoost (top 5) จะมีค่า MAE อยู่ที่ 413.6235 และ ตัวแบบ XGBoost (top 10) มีค่า MAE อยู่ที่ 350.5863



รูปที่ 4.5 แผนภูมิแท่งแสดงค่า RMSE ของชุดข้อมูลทดสอบของตัวแบบ XGBoost

จากรูปที่ 4.5 จะพบว่าตัวแบบ XGBoost (filtered feature) จะมีค่า RMSE อยู่ที่ 442.8788 โดย XGBoost (top 5) จะมีค่า RMSE อยู่ที่ 548.5244 และ ตัวแบบ XGBoost (top 10) มีค่า RMSE อยู่ที่ 497.2814



รูปที่ 4.6 แผนภูมิแท่งแสดงค่า R^2 ของชุดข้อมูลทดสอบของตัวแบบ XGBoost

จากรูปที่ 4.6 จะพบว่าตัวแบบ XGBoost (filtered feature) จะมีค่า R^2 อยู่ที่ 0.6042 โดย XGBoost(top 5) จะมีค่า R^2 อยู่ที่ 0.3929 และ ตัวแบบ XGBoost (top 10) มีค่า R^2 อยู่ที่ 0.5010

ตารางที่ 4.7 เปรียบเทียบค่าวัดประสิทธิภาพของตัวแบบ XGBoost

วิธีประสิทธิภาพตัวแบบ	ตัวแบบ	ค่าวัดประสิทธิภาพตัวแบบ	
		ข้อมูลชุดเรียนรู้	ข้อมูลชุดทดสอบ
MAE	XGBoost (filtered feature)	52.0941	308.5393
	XGBoost (top 5)	409.0759	413.6235
	XGBoost (top 10)	298.9147	350.5863
RMSE	XGBoost (filtered feature)	97.1826	442.8788
	XGBoost (top 5)	557.7788	548.5244
	XGBoost (top 10)	440.1011	497.2814
R^2	XGBoost (filtered feature)	0.9821	0.6042
	XGBoost (top 5)	0.4103	0.3929
	XGBoost (top 10)	0.6329	0.501

จากตารางเปรียบเทียบการวัดประสิทธิภาพของตัวแบบ XGBoost จะเห็นได้ว่า ตัวแบบ XGBoost (filtered feature) และXGBoost (top 5) มีปัญหา overfitting ที่เห็นได้ชัด โดยเฉพาะตัวแบบ XGBoost (filtered feature) ดังนั้น ตัวแบบ XGBoost (top 10) จึงเป็นตัวแบบที่มีประสิทธิภาพที่สุดในตัวแบบ XGBoost ทั้งหมดของชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบ

4.2 การวัดประสิทธิภาพของตัวแบบ SVM

เริ่มจากการหาค่า feature importance ของตัวแบบ XGBoost ทั้ง 3 ตัวแบบ

- Filtered feature (Importance > 0)

- ตัวแปรที่มีค่า importance มากที่สุด 5 อันดับแรก (Top 5)
- ตัวแปรที่มีค่า importance มากที่สุด 10 อันดับแรก (Top 10)

จากการหาค่า feature importance จะได้ ตัวแปรที่มีค่า importance > 0 ของตัวแบบ SVM ตามตารางที่ 4.8

ตารางที่ 4.8 ค่า feature importance ของ SVM (Filtered feature)

Feature	Feature Importance
accommodates	0.2319
Tourist District_tourist distirct	0.2044
bedrooms	0.1806
host_total_listings_count	0.0917
availability_30	0.0675
host_is_superhost_t	0.0667
review_scores_rating	0.0652
number_of_reviews	0.0618
work_email	0.0617
minimum_nights	0.0486
email	0.0445
instant_bookable_t	0.0392

ตารางที่ 4.8 ค่า feature importance ของ SVM (Filtered feature)

Feature	Feature Importance
category_property_Private Room	0.0379
maximum_nights	0.0317
category_property_Room or Shared Space	0.0269
Convenience	0.0204
host_response_rate	0.0191
Safety & Facilities	0.0144
host_response_time_within an hour	0.0125
host_response_time_within a few hours	0.0110
phone	0.01015
host_response_time_within a day	0.0078
Comfort & Basics	0.0048
host_acceptance_rate	0.0039
host_identity_verified_t	0.00127
host_has_profile_pic_t	0.00016

ตารางที่ 4.9 ค่าfeature importance ของ SVM (Top 5)

Feature	Feature Importance
accommodates	0.2319
Tourist District_tourist distirct	0.2044
bedrooms	0.1806
host_total_listings_count	0.0917
availability_30	0.0675

จากการหา top 5 ตัวแปรที่มีค่า importance สูงที่สุดของตัวแบบ SVM พบว่าประกอบไปด้วยตัวแปรดังต่อไปนี้ accommodates, Tourist District_tourist district, bedrooms, host_total_listings_count และ availability_30

ตารางที่ 4.10 ค่า feature importance ของ SVM (Top 10)

Feature	Feature Importance
accommodates	0.2319
Tourist District_tourist distirct	0.2044
bedrooms	0.1806
host_total_listings_count	0.0917
availability_30	0.0675
host_is_superhost_t	0.0667
review_scores_rating	0.0652

ตารางที่ 4.10 ค่า feature importance ของ SVM (Top 10) (ต่อ)

Feature	Feature Importance
number_of_reviews	0.0618
work_email	0.0617
minimum_nights	0.0486

จากการหา top 10 ตัวแปรที่มีค่า importance สูงที่สุดของตัวแบบ SVM พบว่าประกอบไปด้วยตัวแปรดังต่อไปนี้ accommodates, Tourist District_tourist district, bedrooms, host_total_listings_count, availability_30, host_is_superhost_t, review_scores_rating, number_of_reviews, work_email, minimum_nights

นำfiltered features, top 5 features และ top 10 features มาหาค่า hyperparameter ที่ดีที่สุดโดยใช้ gridsearchcv

ตารางที่ 4.11 ค่า hyperparameter ของ SVM (Filtered Feature)

Hyperparameter	Value	Best Hyperparameter
C	[0.1, 1, 10, 100]	10
gamma	[scale, auto]	scale
kernel	[rbf, linear]	rbf

ในกรณีที่ เลือก feature ที่มีค่า importance > 0 สำหรับวิธี SVM ได้ค่า hyperparameter ที่เหมาะสมที่สุดดังนี้ C = 10, gamma = scale และ kernel = rbf

ตารางที่ 4.12 ค่า hyperparameter ของ SVM (Top 5)

Hyperparameter	Value	Best Hyperparameter
C	[0.1, 1, 10, 100]	1
gamma	[scale, auto]	scale
kernel	[rbf, linear]	rbf

ในกรณีที่ เลือก feature ที่มีค่า importance มากที่สุด 5 อันดับสำหรับวิธี SVM ได้ค่า hyperparameter ที่เหมาะสมที่สุดดังนี้ C = 1, gamma = scale และ kernel = rbf

ตารางที่ 4.13 ค่า hyperparameter ของ SVM (Top 10)

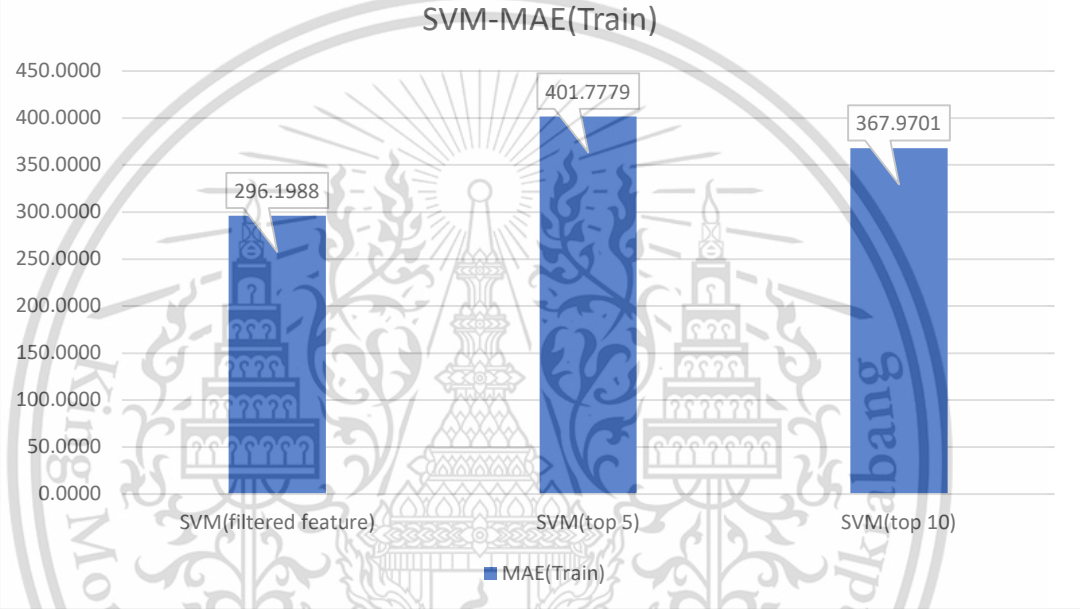
Hyperparameter	Value	Best Hyperparameter
C	[0.1, 1, 10, 100]	1
gamma	[scale, auto]	scale
kernel	[rbf, linear]	rbf

ในกรณีที่ เลือก feature ที่มีค่า importance มากที่สุด 10 อันดับสำหรับวิธี SVM ได้ค่า hyperparameter ที่เหมาะสมที่สุดดังนี้ C = 1, gamma = scale และ kernel = rbf

ทำการวัดประสิทธิภาพของตัวแบบ XGBoost ผ่านค่า MAE, RMSE และ R^2 โดยแบ่งข้อมูลที่วัดประสิทธิภาพเป็นข้อมูลชุดเรียนรู้และข้อมูลชุดทดสอบ

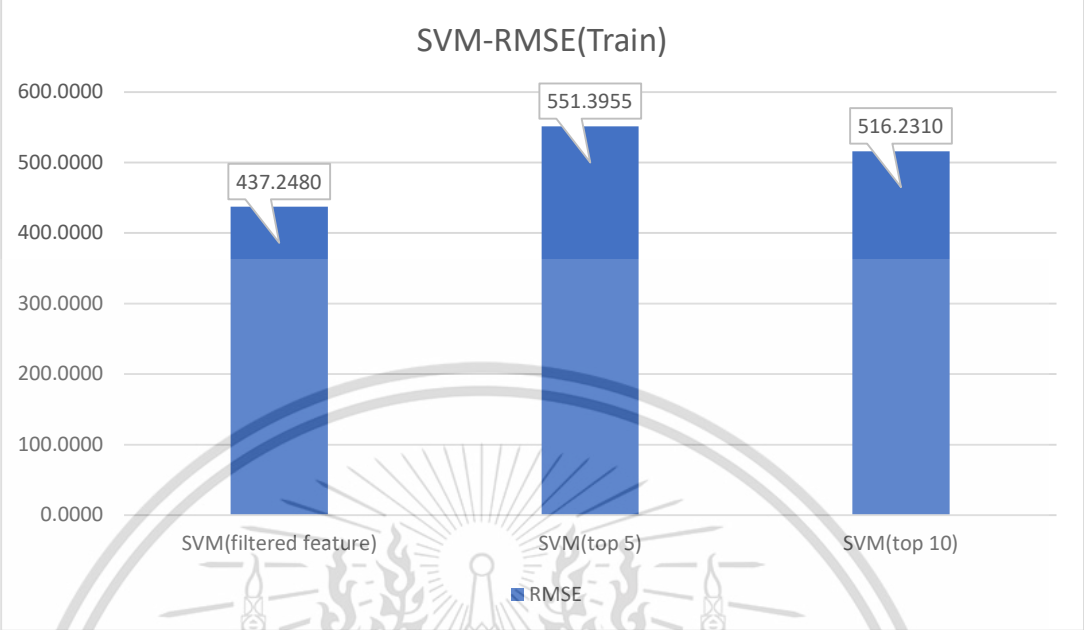
ชุดข้อมูลเรียนรู้

ผลการทดสอบตัวแบบ SVM ของชุดข้อมูลเรียนรู้โดยใช้ค่า MAE, RMSE และ R^2 ในการวัดประสิทธิภาพของตัวแบบ SVM ทั้ง 3 ตัวแบบ (filtered feature/ top 5/ top 10) โดยเป็นไปตามรูปที่ 4.7, 4.8 และ 4.9



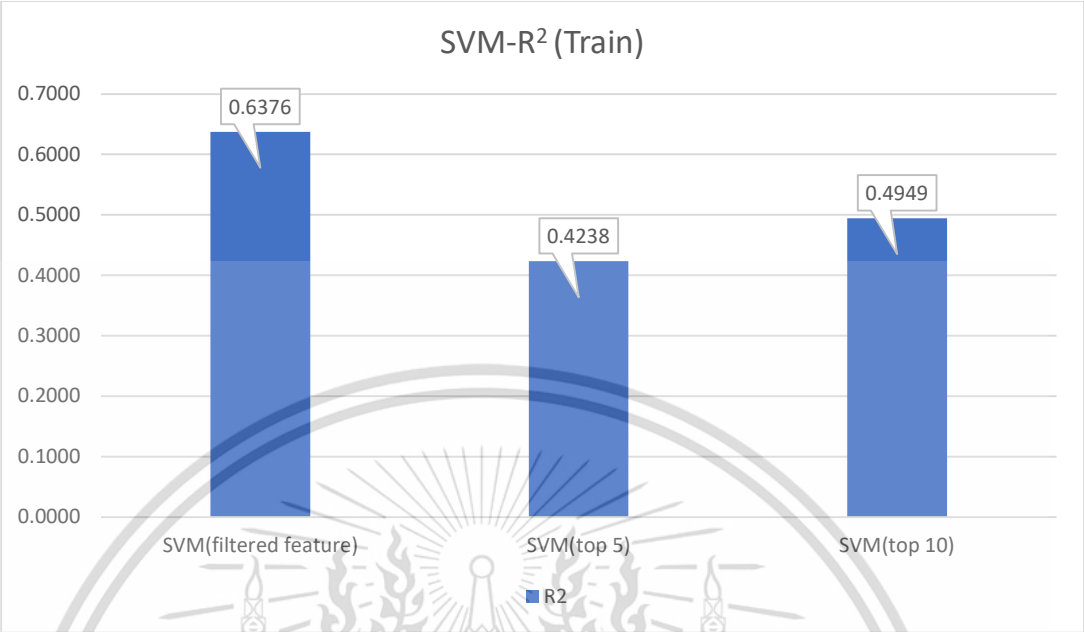
รูปที่ 4.7 แผนภูมิแท่งแสดงค่า MAE ของชุดข้อมูลเรียนรู้ของตัวแบบ SVM

จากรูปที่ 4.7 จะพบว่าตัวแบบ SVM (filtered feature) จะมีค่า MAE ที่ 296.1988 ในขณะที่ SVM (top 5) จะมีค่า MAE อยู่ที่ 401.7779 และ SVM (top 10) จะมีค่า MAE อยู่ที่ 367.9701



รูปที่ 4.8 แผนภูมิแท่งแสดงค่า RMSE ของชุดข้อมูลเรียนรู้ของตัวแบบ SVM

จากรูปที่ 4.8 จะพบว่าตัวแบบ SVM (filtered feature) จะมีค่า RMSE ที่ 437.2480 ในขณะที่ SVM (top 5) จะมีค่า RMSE อยู่ที่ 551.3955 และ SVM (top 10) จะมีค่า RMSE อยู่ที่ 516.2310

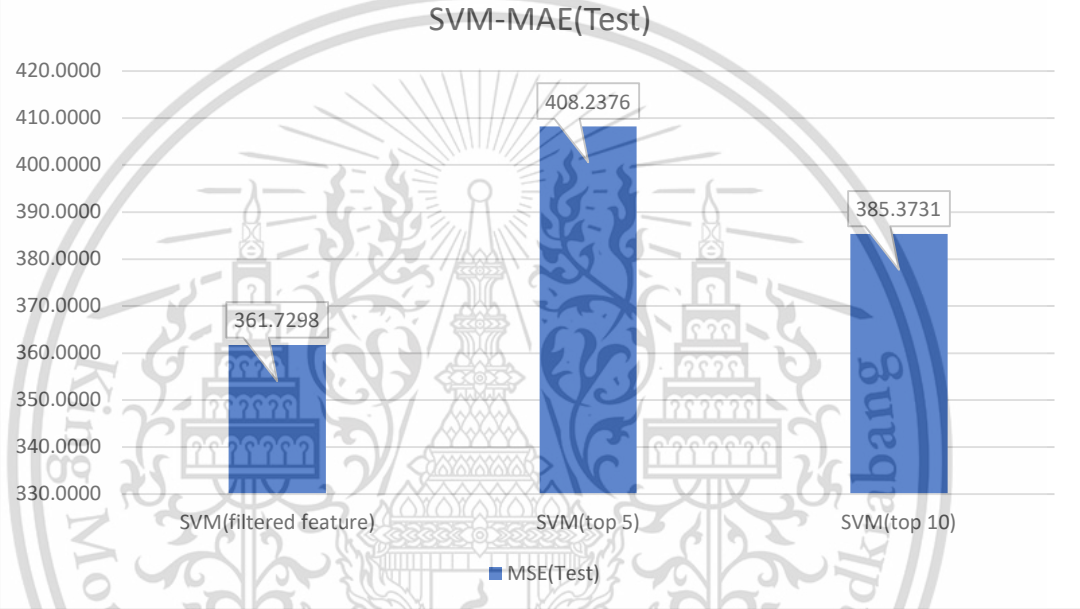


รูปที่ 4.9 แผนภูมิแท่งแสดงค่า R^2 ของชุดข้อมูลเรียนรู้ของตัวแบบ SVM

จากรูปที่ 4.9 จะพบว่าตัวแบบ SVM (filtered feature) จะมีค่า R^2 ที่ 0.6376 ในขณะที่ SVM (top 5) จะมีค่า R^2 อยู่ที่ 0.4238 และ SVM (top 10) จะมีค่า R^2 อยู่ที่ 0.4949

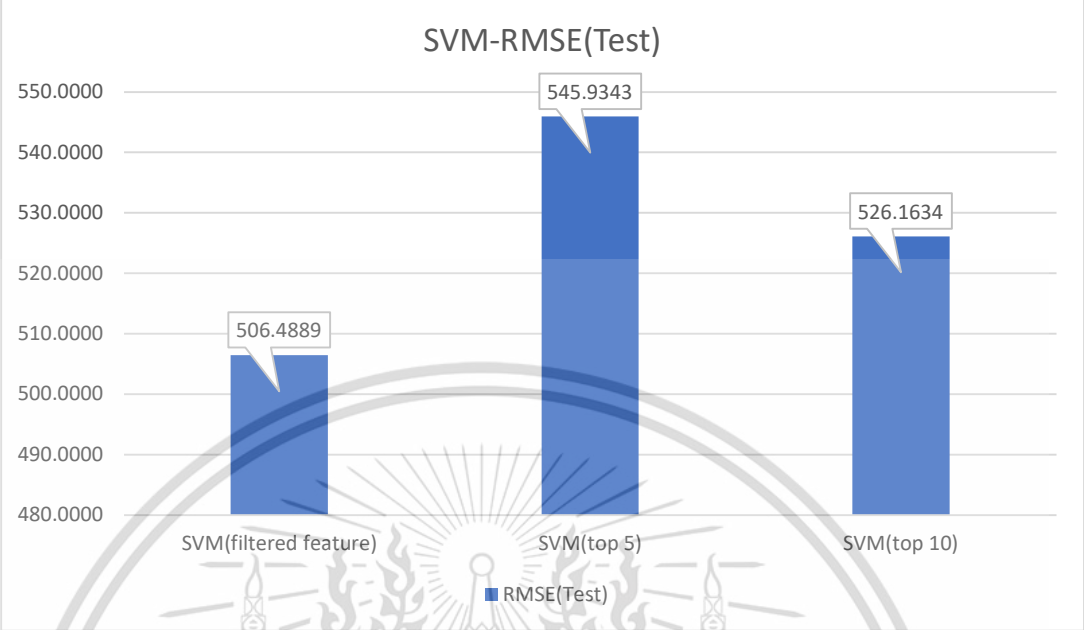
ชุดข้อมูลทดสอบ

ผลการทดสอบตัวแบบ SVM ของชุดข้อมูลทดสอบโดยใช้ค่า MAE, RMSE และ R^2 ในการวัดประสิทธิภาพของตัวแบบ SVM ทั้ง 3 ตัวแบบ (filtered feature/ top 5/ top 10) โดยเป็นไปตามรูปที่ 4.10, 4.11 และ 4.12



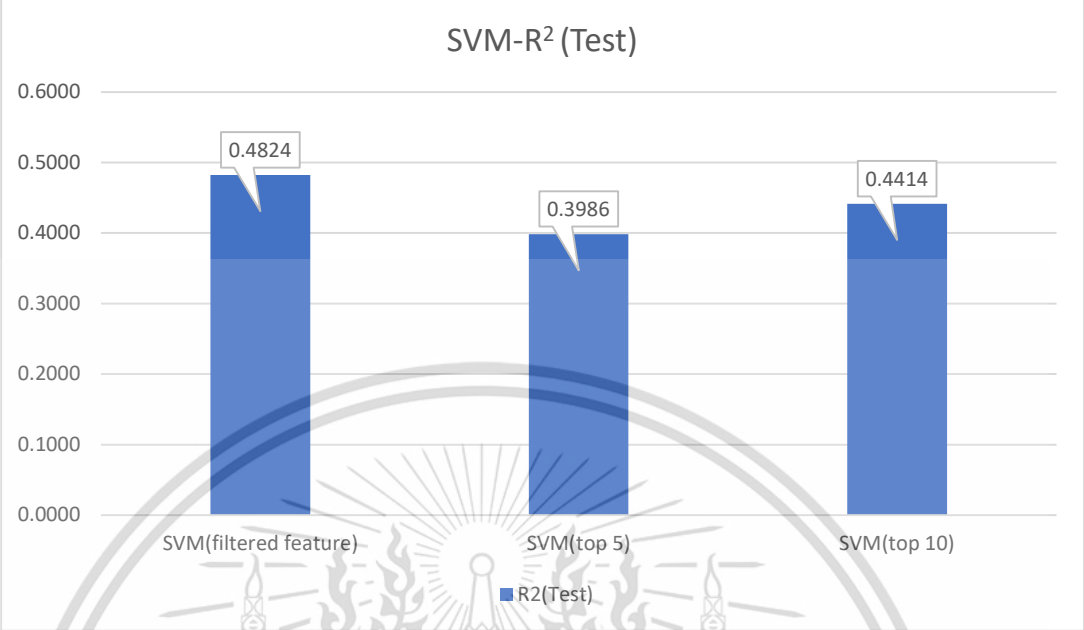
รูปที่ 4.10 แผนภูมิแท่งแสดงค่า MAE ของชุดข้อมูลทดสอบของตัวแบบ SVM

จากรูปที่ 4.10 จะพบว่าตัวแบบ SVM (filtered feature) จะมีค่า MAE ที่ 361.7298 ในขณะที่ SVM (top 5) จะมีค่า MAE อยู่ที่ 408.2376 และ SVM (top 10) จะมีค่า MAE อยู่ที่ 385.3731



รูปที่ 4.11 แผนภูมิแท่งแสดงค่า RMSE ของชุดข้อมูลทดสอบของตัวแบบ SVM

จากรูปที่ 4.11 จะพบว่าตัวแบบ SVM (filtered feature) จะมีค่า RMSE ที่ 506.4889 ในขณะที่ SVM (top 5) จะมีค่า RMSE อยู่ที่ 545.9343 และ SVM (top 10) จะมีค่า RMSE อยู่ที่ 526.1634



รูปที่ 4.12 แผนภูมิแท่งแสดงค่า R^2 ของชุดข้อมูลทดสอบของตัวแบบ SVM

จากรูปที่ 4.12 จะพบว่าตัวแบบ SVM (filtered feature) จะมีค่า R^2 ที่ 0.4824 ในขณะที่ SVM (top 5) จะมีค่า R^2 อยู่ที่ 0.3986และ SVM (top 10) จะมีค่า R^2 อยู่ที่ 0.4414

ตารางที่ 4.14 เปรียบเทียบค่าวัดประสิทธิภาพของตัวแบบ SVM

วิธีประสิทธิภาพตัวแบบ	ตัวแบบ	ค่าวัดประสิทธิภาพตัวแบบ	
		ข้อมูลชุดเรียนรู้	ข้อมูลชุดทดสอบ
MAE	SVM (filtered feature)	296.1988	361.7298
	SVM (top 5)	401.7779	408.2376
	SVM (top 10)	367.9701	385.3731
RMSE	SVM (filtered feature)	437.248	506.4889
	SVM (top 5)	551.3955	545.9342
	SVM (top 10)	516.231	526.1634
R^2	SVM (filtered feature)	0.6376	0.4824
	SVM (top 5)	0.4238	0.3986
	SVM (top 10)	0.4949	0.4414

จากตารางที่ 4.14 ตารางเปรียบเทียบการวัดประสิทธิภาพของตัวแบบ SVM จะเห็นได้ว่า ตัวแบบ SVM (filtered feature) มีปัญหา overfitting :ซึ่งทำให้ไม่ถูกนำไปเลือกใช้ ถึงแม้ว่าจะมีประสิทธิภาพที่ดีที่สุด ในบรรดาตัวแบบทั้งหมด และตัวแบบ SVM (top 10) มีประสิทธิภาพที่ดีสุดในการเปรียบเทียบตัวแบบ SVM ทั้งหมดของชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบ

4.3 การวัดประสิทธิภาพของตัวแบบการถดถอยเชิงเส้น

สำหรับตัวแบบการถดถอยเชิงเส้น จะใช้การเลือกตัวแปรต้นโดยพิจารณาจากค่า p-value โดยจะทำเลือกตัวที่มีค่า p-value น้อยกว่า 0.05 ดังแสดงในตารางที่ 4.15

ตารางที่ 4.15 ค่า p-value

Feature	P-value	การเลือกเข้าตัวแบบ
const	0.00	ถูกเลือก
accommodates	0.00	ถูกเลือก
Tourist District_tourist	0.00	ถูกเลือก
category_property_Private	0.00	ถูกเลือก
host_is_superhost_t	0.00	ถูกเลือก
availability_30	0.00	ถูกเลือก
Safety & Facilities	0.00	ถูกเลือก
bedrooms	0.00	ถูกเลือก
Convenience	0.00	ถูกเลือก
review_scores_rating	0.00	ถูกเลือก
minimum_nights	0.00	ถูกเลือก
phone	0.01	ถูกเลือก
host_response_rate	0.02	ถูกเลือก
email	0.02	ถูกเลือก
host_has_profile_pic_t	0.04	ถูกเลือก
host_total_listings_count	0.06	ไม่ถูกเลือก
category_property_Room or	0.06	ไม่ถูกเลือก
Appliances & Technology	0.07	ไม่ถูกเลือก

ตารางที่ 4.15 ค่า p-value (ต่อ)

Feature	P-value	การเลือกเข้าตัวแบบ
host_response_time_within a day	0.23	ไม่ถูกเลือก
maximum_nights	0.27	ไม่ถูกเลือก
host_response_time_within an hour	0.28	ไม่ถูกเลือก
host_response_time_within a few hours	0.32	ไม่ถูกเลือก
host_identity_verified_t	0.33	ไม่ถูกเลือก
instant_bookable_t	0.46	ไม่ถูกเลือก
number_of_reviews	0.49	ไม่ถูกเลือก
host_acceptance_rate	0.50	ไม่ถูกเลือก
category_property_Special	0.57	ไม่ถูกเลือก
work_email	0.89	ไม่ถูกเลือก
Comfort & Basics	0.94	ไม่ถูกเลือก

เมื่อนำตัวแปรต้นที่มีค่า p-value น้อยกว่า 0.05 จะได้ตัวแบบการถดถอยเชิงเส้นที่มีค่าสัมประสิทธิ์การถดถอยดังนี้

ตารางที่ 4.16 coefficient ของตัวแบบ Linear Regression

Feature	Coefficients
Intercept	1022.01
host_response_rate	-203.56
email	15.07
phone	26.10
accommodates	469.82
bedrooms	61.65
minimum_nights	-24.13
availability_30	31.41
review_scores_rating	29.43
Safety & Facilities	199.22
Convenience	82.83
host_has_profile_pic_t	239.31
host_is_superhost_t	90.80
Tourist District_tourist distirct	368.54
category_property_Private Room	-220.24

จากสัมประสิทธิ์การถดถอยเชิงเส้น สามารถแปลผลได้ดังนี้

β_0 หมายถึง ราคาเฉลี่ยของที่พัก เท่ากับ 1022.01 บาท เมื่อตัวแปรต้นทุกตัวมีค่า =0

β_1 หมายถึง ราคาเฉลี่ยของที่พักลดลง 203.56 บาท เมื่อตัวแปร host_response_rate เพิ่มขึ้น 1 หน่วย และตัวแปรต้นอื่นๆมีค่าคงที่

β_2 หมายถึง ราคาเฉลี่ยของที่พักลดลง 15.07 บาท เมื่อที่พักมี การ verify ด้วย email และตัวแปรต้นอื่นๆมีค่าคงที่

β_3 หมายถึง ราคาเฉลี่ยของที่พักเพิ่มขึ้น 26.10 บาท เมื่อที่พักมี การ verify ด้วย phone และตัวแปรต้นอื่นๆมีค่าคงที่

β_4 หมายถึง ราคาเฉลี่ยของที่พักเพิ่มขึ้น 469.82 บาท เมื่อตัวแปร accommodates เพิ่มขึ้น 1 หน่วย และตัวแปรต้นอื่นๆมีค่าคงที่

β_5 หมายถึง ราคาเฉลี่ยของที่พักเพิ่มขึ้น 61.65 บาท เมื่อตัวแปร bedrooms เพิ่มขึ้น 1 หน่วย และตัวแปรต้นอื่นๆมีค่าคงที่

β_6 หมายถึง ราคาเฉลี่ยของที่พักลดลง 24.13 บาท เมื่อตัวแปร minimum_nights เพิ่มขึ้น 1 หน่วย และตัวแปรต้นอื่นๆมีค่าคงที่

β_7 หมายถึง ราคาเฉลี่ยของที่พักเพิ่มขึ้น 31.41 บาท เมื่อตัวแปร availability_30 เพิ่มขึ้น 1 หน่วย และตัวแปรต้นอื่นๆมีค่าคงที่

β_8 หมายถึง ราคาเฉลี่ยของที่พักเพิ่มขึ้น 29.43 บาท เมื่อตัวแปร review_scores_rating เพิ่มขึ้น 1 หน่วย และตัวแปรต้นอื่นๆมีค่าคงที่

β_9 หมายถึง ราคาเฉลี่ยของที่พักเพิ่มขึ้น 199.22 บาท เมื่อที่พักมี สิ่งอำนวยความสะดวกประเภท Safety & Facilities และตัวแปรต้นอื่นๆมีค่าคงที่

β_{10} หมายถึง ราคาเฉลี่ยของที่พักเพิ่มขึ้น 82.23 บาท เมื่อที่พักมี สิ่งอำนวยความสะดวกประเภท Convenience และตัวแปรต้นอื่นๆมีค่าคงที่

β_{11} หมายถึง ราคาเฉลี่ยของที่พักเพิ่มขึ้น 239.31 บาท เมื่อเจ้าของที่พักมีรูปโปรไฟล์ และตัวแปรต้นอื่นๆมีค่าคงที่

β_{12} หมายถึง ราคาเฉลี่ยของที่พักเพิ่มขึ้น 90.80 บาท เมื่อเจ้าของที่พักมีสถานะ superhost และตัวแปรต้นอื่นๆมีค่าคงที่

β_{13} หมายถึง ราคาเฉลี่ยของที่พักเพิ่มขึ้น 368.54 บาท เมื่อที่พักตั้งอยู่ในเขตท่องเที่ยว และตัวแปรต้นอื่นๆมีค่าคงที่

β_{14} หมายถึง ราคาเฉลี่ยของที่พักลดลง 220.24 บาท เมื่อที่พักมี สิ่งอำนวยความสะดวกประเภท Safety & Facilities และตัวแปรต้นอื่นๆมีค่าคงที่

β_{15} หมายถึง ราคาเฉลี่ยของที่พักเพิ่มขึ้น 199.22 บาท เมื่อที่พักมี สิ่งอำนวยความสะดวกประเภท Safety & Facilities และตัวแปรต้นอื่นๆมีค่าคงที่

ตารางที่ 4.17 เปรียบเทียบค่าวัดประสิทธิภาพของตัวแบบ Linear Regression

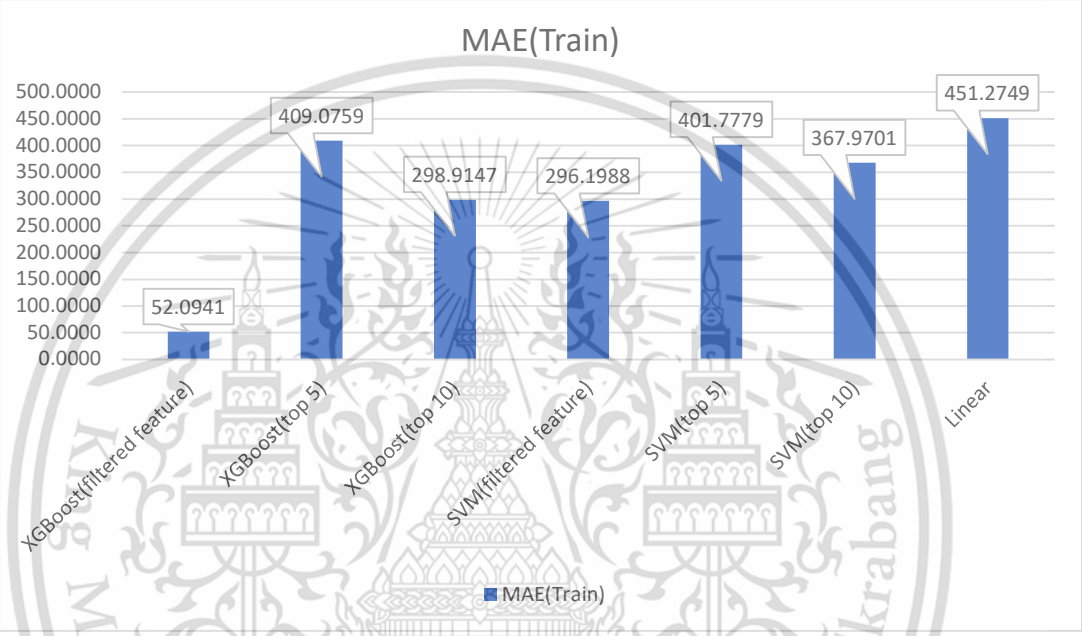
วิธีวัดประสิทธิภาพตัวแบบ	ค่าวัดประสิทธิภาพตัวแบบ	
	ชุดข้อมูลเรียนรู้	ชุดข้อมูลทดสอบ
MAE	451.2749	447.3687
RMSE	653.0655	612.9571
R^2	0.1917	0.2419

จากตารางที่ 4.17 จะเห็นได้ว่า ตัวแบบการถดถอยเชิงเส้นมีประสิทธิภาพที่ไม่ดี หากดูจากค่า R^2 โดยค่า R^2 มีค่าที่ต่ำมากทั้งชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบ ที่ 0.1917 และ 0.2419 ตามลำดับ

4.4 การวัดประสิทธิภาพระหว่างตัวแบบ

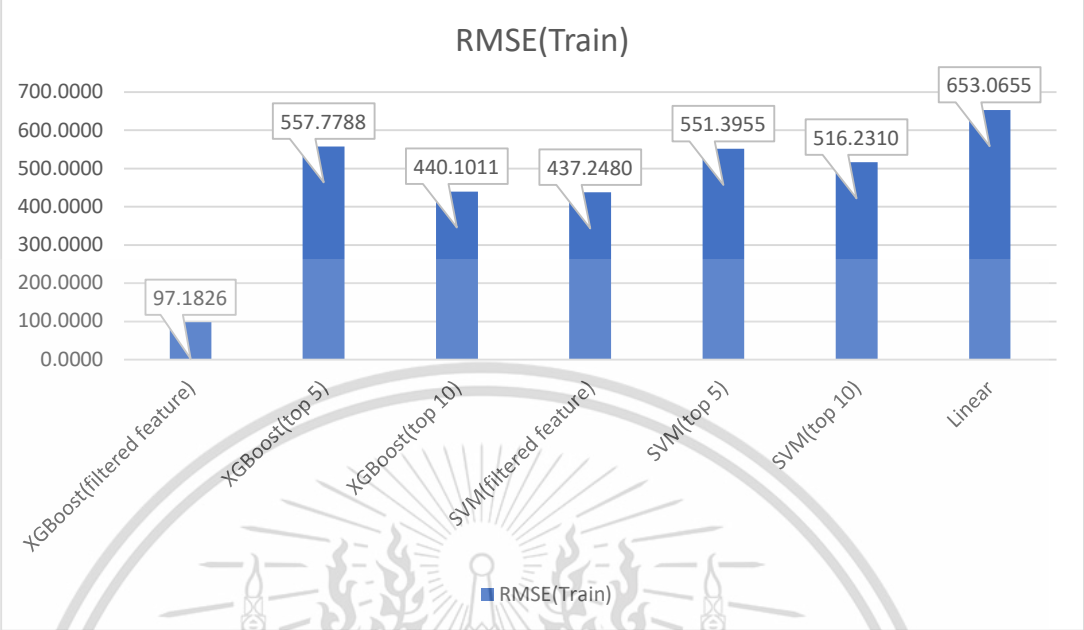
ชุดข้อมูลเรียนรู้

ผลการทดสอบระหว่างแต่ละตัวแบบของชุดข้อมูลเรียนรู้โดยใช้ค่า MAE, RMSE และ R^2 ในการวัดประสิทธิภาพของแต่ละตัวแบบโดยเป็นไปตามรูปที่ 4.13, 4.14 และ 4.15



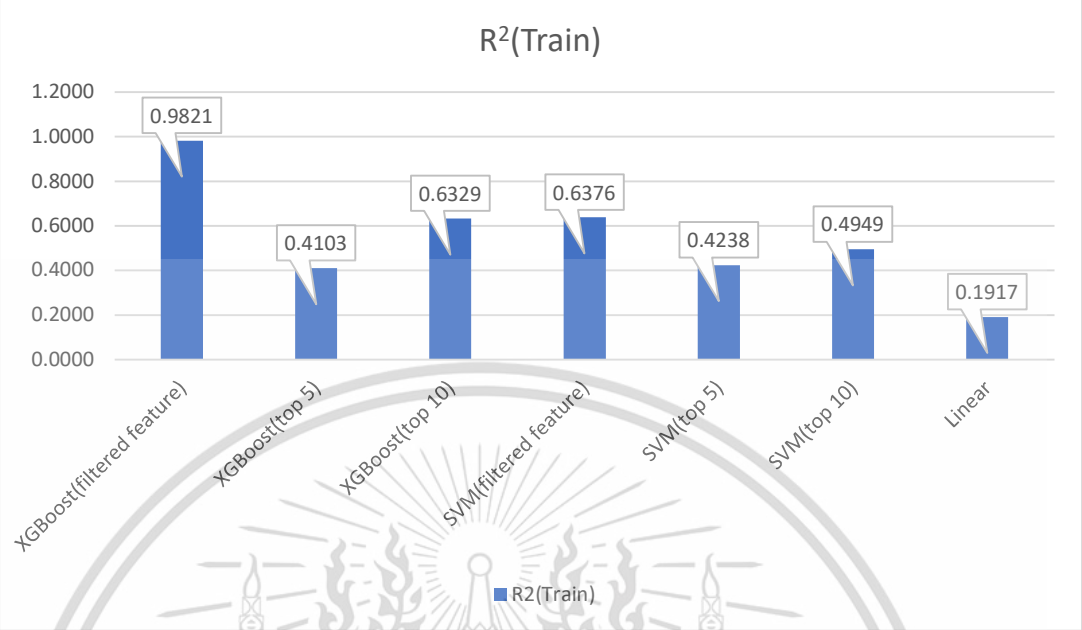
รูปที่ 4.13 แผนภูมิแท่งแสดงค่า MAE ของชุดข้อมูลเรียนรู้ระหว่างตัวแบบ

จากรูปที่ 4.13 การเปรียบเทียบประสิทธิภาพของตัวแบบโดยใช้ MAE เป็นตัววัดประสิทธิภาพของตัวแบบ จะเห็นได้ว่า ตัวแบบ XGBoost (filtered features) มีค่า MAE ที่น้อยที่สุดที่ 52.0941 เมื่อเปรียบเทียบกับตัวแบบอื่นๆ โดยมีตัวแบบSVM (filtered feature) และ XGBoost (top 10) ดีที่สุดรองลงมาตามลำดับที่ 296.1988 และ 298.9147



รูปที่ 4.14 แผนภูมิแท่งแสดงค่า RMSE ของชุดข้อมูลเรียนรู้ระหว่างตัวแบบ

จากรูปที่ 4.14 การเปรียบเทียบประสิทธิภาพของตัวแบบโดยใช้ RMSE เป็นตัววัดประสิทธิภาพของตัวแบบ จะเห็นได้ว่า ตัวแบบ XGBoost (filtered features) มีค่า RMSE ที่น้อยที่สุดที่ 97.1826 เมื่อเปรียบเทียบกับตัวแบบอื่นๆ โดยมีตัวแบบSVM (filtered feature) และ XGBoost (top 10) ที่มีค่า RMSE น้อยลงมาตามลำดับที่ 437.2480 และ 440.1011

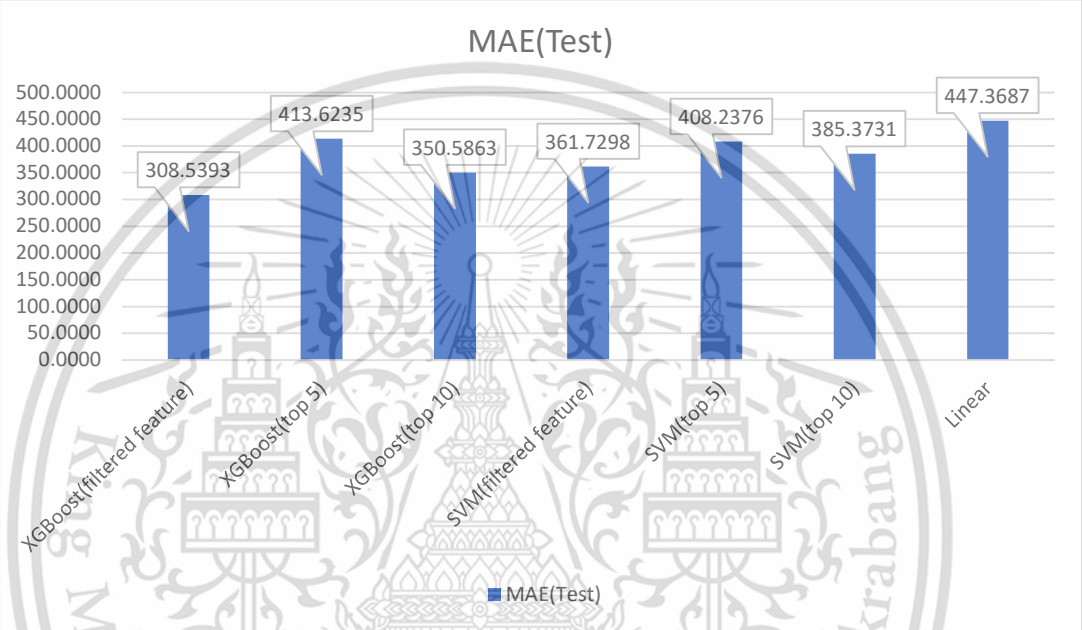


รูปที่ 4.15 แผนภูมิแท่งแสดงค่า R^2 ของชุดข้อมูลเรียนรู้ระหว่างตัวแบบ

จากรูปที่ 4.15 การเปรียบเทียบประสิทธิภาพของตัวแบบโดยใช้ R^2 เป็นตัววัดประสิทธิภาพของตัวแบบ จะเห็นได้ว่า ตัวแบบ XGBoost (filtered features) มีค่า R^2 ที่มากที่สุดที่ 0.9821 เมื่อเปรียบเทียบกับตัวแบบอื่นๆ โดยมีตัวแบบSVM (filtered feature) และ XGBoost (top 10) ดีที่สุดรองลงมาตามลำดับที่ 0.6376 และ 0.6329

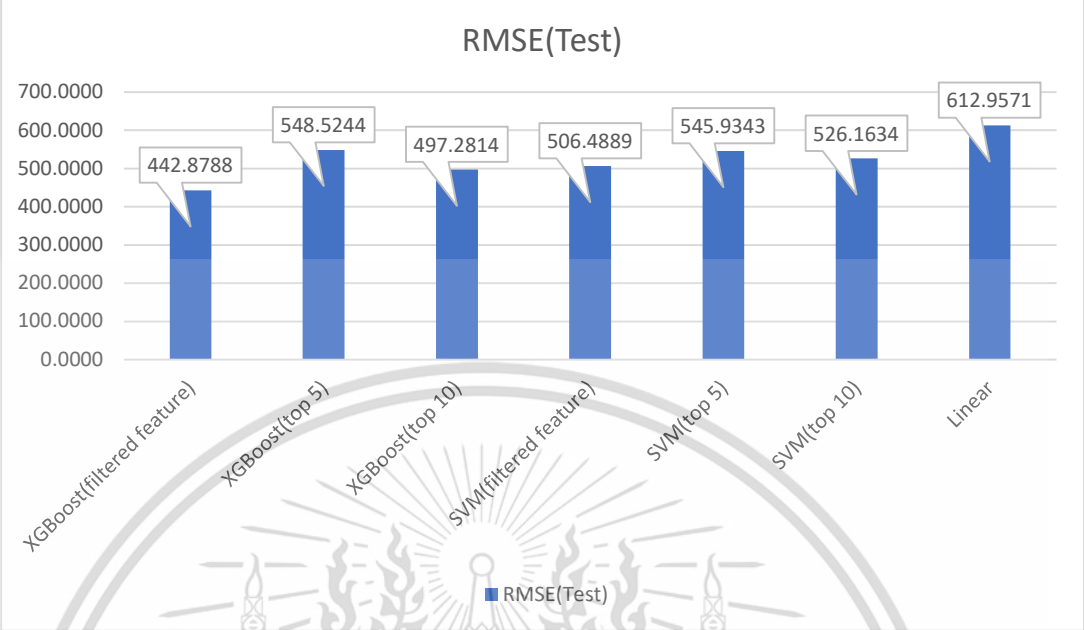
ชุดข้อมูลทดสอบ

ผลการทดสอบระหว่างแต่ละตัวแบบของชุดข้อมูลเรียนรู้โดยใช้ค่า MAE, RMSE และ R^2 ในการวัดประสิทธิภาพของแต่ละตัวแบบโดยเป็นไปตามรูปที่ 4.16, 4.17 และ 4.18



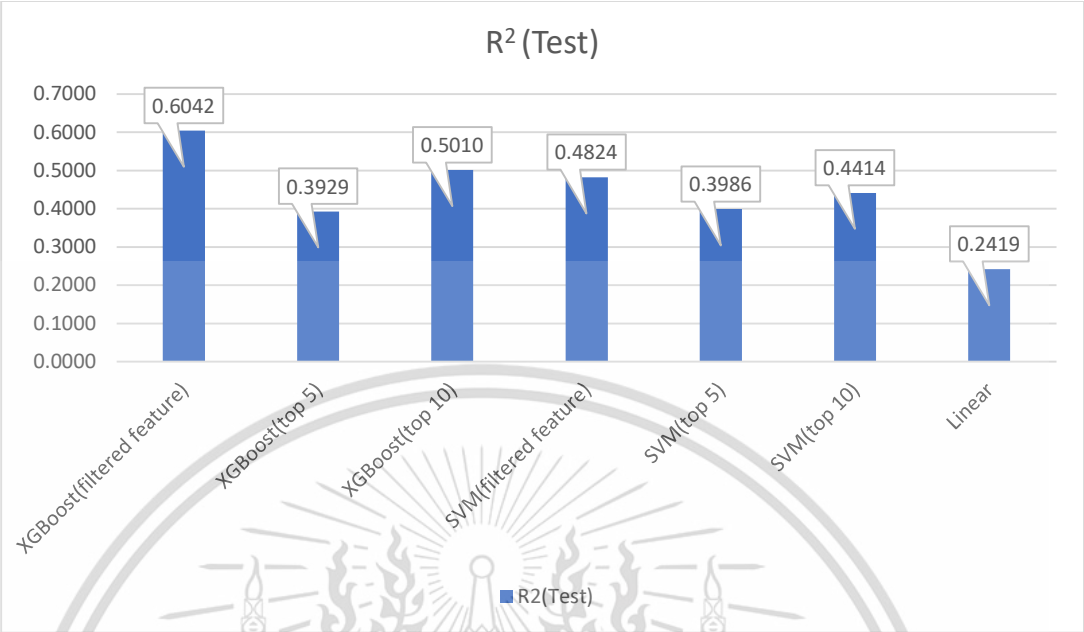
รูปที่ 4.16 แผนภูมิแท่งแสดงค่า MAE ของชุดข้อมูลทดสอบระหว่างตัวแบบ

จากรูปที่ 4.16 การเปรียบเทียบประสิทธิภาพของตัวแบบโดยใช้ MAE เป็นตัววัดประสิทธิภาพของตัวแบบ จะเห็นได้ว่า ตัวแบบ XGBoost (filtered features) มีค่า MAE ที่น้อยที่สุดที่ 308.5393 เมื่อเปรียบเทียบกับตัวแบบอื่นๆ โดยมีตัวแบบ XGBoost (top 10) และ SVM (filtered feature) ดีที่สุดรองลงมาตามลำดับที่ 350.5863 และ 361.7298



รูปที่ 4.17 แผนภูมิแท่งแสดงค่า RMSE ของชุดข้อมูลทดสอบระหว่างตัวแบบ

จากรูปที่ 4.17 การเปรียบเทียบประสิทธิภาพของตัวแบบโดยใช้ RMSE เป็นตัววัดประสิทธิภาพของตัวแบบ จะเห็นได้ว่า ตัวแบบ XGBoost (filtered features) มีค่า RMSE ที่น้อยที่สุดที่ 442.8788 เมื่อเปรียบเทียบกับตัวแบบอื่นๆ โดยมีตัวแบบ XGBoost(top 10) และ SVM (filtered feature) ที่มีค่า RMSE น้อยลงมาตามลำดับที่ 497.2814และ 506.48889



รูปที่ 4.18 แผนภูมิแท่งแสดงค่า R^2 ของชุดข้อมูลทดสอบระหว่างตัวแบบ

จากรูปที่ 4.18 การเปรียบเทียบประสิทธิภาพของตัวแบบโดยใช้ R^2 เป็นตัววัดประสิทธิภาพของตัวแบบ จะเห็นได้ว่า ตัวแบบ XGBoost (filtered features) มีค่า R^2 ที่มากที่สุดที่ 0.6042 เมื่อเปรียบเทียบกับตัวแบบอื่นๆ โดยมีตัวแบบSVM (filtered feature) และ XGBoost (top 10) ดีที่สุดรองลงมาตามลำดับที่ 0.5010 และ 0.4824

ตารางที่ 4.18 เปรียบเทียบค่าวัดประสิทธิภาพของระหว่างตัวแบบ

วิธีประสิทธิภาพตัวแบบ	ตัวแบบ	ค่าวัดประสิทธิภาพตัวแบบ	
		ข้อมูลชุดเรียนรู้	ข้อมูลชุดทดสอบ
MAE	XGBoost (filtered feature)	52.0941	308.5393
	XGBoost (top 5)	409.0759	413.6235
	XGBoost (top 10)	298.9147	350.5863
	SVM (filtered feature)	296.1988	361.7298
	SVM (top 5)	401.7779	408.2376
	SVM (top 10)	367.9701	385.3731
	Linear	451.2749	447.3687
RMSE	XGBoost (filtered feature)	97.1826	442.8788
	XGBoost (top 5)	557.7788	548.5244
	XGBoost (top 10)	440.1011	497.2814
	SVM (filtered feature)	437.2480	506.4889
	SVM (top 5)	551.3955	545.9342
	SVM (top 10)	516.2310	526.1634
	Linear	653.0655	612.9570

ตารางที่ 4.18 เปรียบเทียบค่าวัดประสิทธิภาพของระหว่างตัวแบบ (ต่อ)

วิธีประสิทธิภาพตัวแบบ	ตัวแบบ	ค่าวัดประสิทธิภาพตัวแบบ	
		ข้อมูลชุดเรียนรู้	ข้อมูลชุดทดสอบ
R^2	XGBoost (filtered feature)	0.9821	0.6042
	XGBoost (top 5)	0.4103	0.3929
	XGBoost (top 10)	0.6329	0.5010
	SVM (filtered feature)	0.6376	0.4824
	SVM (top 5)	0.4238	0.3986
	SVM (top 10)	0.4949	0.4414
	Linear	0.1917	0.2419

4.5 การอภิปรายและสรุปผลการทดลอง

4.5.1 ปัญหา overfitting

จากตารางที่ 4.18 ที่ ตัวแบบ XGBoost (filtered feature) มีค่าการวัดประสิทธิภาพของตัวแบบ ไม่ว่าจะเป็น ค่า MAE, RMSE หรือ R^2 ล้วนมีผลลัพธ์ของข้อมูลชุดการเรียนรู้ที่ดีกว่าผลลัพธ์ของข้อมูลชุดการทดสอบเป็นอย่างมาก ซึ่งแสดงให้เห็นถึงปัญหา overfitting ได้อย่างชัดเจน ซึ่ง 1 ในวิธีการแก้ปัญหา overfitting คือ การลดจำนวน feature ภายในตัวแบบ โดย ในงานวิจัยครั้งนี้ ก็ได้มีตัวแบบ XGBoost ที่ทำการเลือกเฉพาะค่า feature ที่มีตัวแปรสูงสุด 5 และ 10 อันดับแรก ในการนำตัวแปรเหล่านั้นมารันเป็นตัวแบบ XGBoost (Top 5) และ XGBoost (Top 10) ตามลำดับ โดยหากทำการเทียบค่าการวัดประสิทธิภาพของตัวแบบ จะพบได้ว่าตัวแบบ XGBoost (Top 10) และ SVM (filtered featured) มีประสิทธิภาพที่ดีที่สุดในบรรดาตัวแบบทั้งหมด เพียงแต่ว่า จากการสังเกตจะพบเห็นถึงปัญหา overfitting ได้ จากความต่างของค่า MAE ของชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบ โดยทั้งสองตัวแบบจะมีค่า MAE ที่มีความต่างกันประมาณ 20% ซึ่งก็แสดงให้เห็นถึงปัญหา overfitting ในตัวแบบ XGBoost (top 10) และ SVM (filtered feature) ด้วยเช่นกัน

4.5.2 ตัวแบบที่มีประสิทธิภาพที่ดีที่สุด

จากตารางที่ 4.18 ที่พบได้ว่ามีปัญหา overfitting ในหลายๆตัวแบบ โดยสังเกตได้จากค่าการวัดประสิทธิภาพของตัวแบบ โดยจะสรุปได้ว่า ตัวแบบ XGBoost (filtered feature) , XGBoost (top 10) และ SVM (filtered feature) เป็นตัวแบบที่มีปัญหา overfitting โดยจะไม่นำตัวแบบเหล่านี้มาใช้ในการพิจารณาตัวแบบที่มีประสิทธิภาพที่ดีที่สุดในการทำนายราคาที่พักในกรุงเทพ โดยจากตารางที่ 4.18 จะพบว่าตัวแบบ SVM (Top 10) ที่มีประสิทธิภาพที่ดีสุดในการทำนายราคาที่พัก โดยจะมีค่า MAE อยู่ที่ 367.9701 และ 385.3731 สำหรับชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบตามลำดับ และมีค่า RMSE อยู่ที่ 516.2310 และ 526.1634 สำหรับชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบตามลำดับ โดยสุดท้ายจะมีค่า R^2 อยู่ที่ 0.4949 และ0.4414 สำหรับชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบตามลำดับ ซึ่งจากค่าการวัดประสิทธิภาพของการวัดตัวแบบ จะได้ออกมาว่าตัวแบบ SVM (Top 10) มีประสิทธิภาพที่ดีที่สุดในการทำนายราคาที่พักในกรุงเทพ โดยสุดท้าย สรุปได้ว่าตัวแบบ SVM (Top 10) เป็นตัวแบบที่มีประสิทธิภาพที่ดีที่สุด โดยตัวแบบ SVM สามารถจัดการกับปัญหา overfitting ได้ดีกว่าตัวแบบ XGBoost แต่ตัวแบบ SVM ก็มีประสิทธิภาพที่ไม่ได้ดีเท่าที่ควรและยังเหลือพื้นที่ในการปรับปรุงประสิทธิภาพตัวแบบให้ดีขึ้นในอนาคต

4.5.3 ตัวแปรที่ส่งผลกระทบต่อราคาของที่พักในแพลตฟอร์ม Airbnb ในกรุงเทพ

จากตารางที่ 4.10 ที่แสดงถึง feature importance ของตัวแบบ SVM (Top 10) จะเห็นได้ว่า feature หลักที่ทั้งตัวแบบมีความสำคัญสูง ได้แก่ accommodates, bedrooms และ Tourist District_tourist distirct โดยจะสรุปได้ว่า ไม่ว่าจะเป็น feature bedrooms หรือ accommodates ล้วนเป็น feature ที่แสดงถึงความจุสูงสุดของที่พักไม่ว่าเป็นจำนวนห้องนอนหรือจำนวนคนสูงสุดที่รองรับได้ของที่พักซึ่งแสดงให้เห็นถึงว่าขนาดของที่พักมีความสำคัญและเป็นปัจจัยหลักในการตั้งราคาที่พัก และ feature Tourist District_tourist distirct เป็น feature ที่แสดงให้เห็นถึงว่าโลเคชันของที่พักนั้นมีความสำคัญต่อราคาที่พักโดยเฉพาะโลเคชันที่อยู่ใกล้กับแลนด์มาร์กหรือสถานที่ท่องเที่ยวชื่อดังของกรุงเทพ ซึ่งสอดคล้องกับงานวิจัยของ (Zhu, Li, และ Xie, 2020) และ (Perez-Sanchez, Serrano-Estrada, และ Marti, 2018) ที่ได้กล่าวไว้ถึงความสำคัญของโลเคชันของที่พักที่ส่งผลต่อของราคาที่พัก โดยเฉพาะอย่างยิ่งการที่ ที่พักใกล้กับแลนด์มาร์กของเมืองจะทำให้ราคาของที่พักมีราคาที่สูง

บทที่ 5

สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

จากงานค้นคว้าอิสระนี้ ได้ทำการศึกษาเรื่อง การทำนายราคาที่พักในกรุงเทพโดยประยุกต์ใช้การเรียนรู้ของเครื่อง โดยได้ทำการนำข้อมูลที่พักจากแพลตฟอร์ม Airbnb โดยข้อมูลมาจากเว็บไซต์ insideairbnb ที่ทำการรวบรวมข้อมูลที่พักของ Airbnb ไว้โดยข้อมูลที่ใช้เป็นข้อมูลของที่พักในกรุงเทพที่อัพเดทล่าสุด เมื่อวันที่ 22 กันยายน พ.ศ.2563 โดยข้อมูลเริ่มต้น มีทั้งหมด 20823 แถว และ 75 คอลัมน์ หลังจากผ่านกระบวนการจัดการข้อมูล จะเหลือข้อมูล ประมาณ 7924 แถว และ 29 คอลัมน์ ได้มีการรันตัวแบบในการทำนายราคาที่พักในกรุงเทพอยู่ 3 ตัวแบบหลัก คือ XGBoost, SVM และ การถดถอยเชิงเส้น โดย จะมีการนำการแบ่งตัวแบบเป็น 3 กรณี สำหรับตัวแบบ XGBoost และ SVM ซึ่งก็คือ 1) ตัวแปรที่มีค่าความสำคัญมากกว่า 0 2) ตัวแปรที่มีค่าความสำคัญสูงสุด 5 อันดับแรก 3) ตัวแปรที่มีค่าความสำคัญสูงสุด 10 อันดับแรก โดยจากผลลัพธ์ที่ได้ จะพบว่า ตัวแบบ XGBoost ที่ใช้ตัวแปรที่มีความสำคัญมากกว่า 0, XGBoost ที่ใช้ตัวแปรที่มีค่าความสำคัญสูงสุด 10 อันดับแรกและ SVM ที่ใช้ตัวแปรที่มีความสำคัญมากกว่า 0 ตัวแบบทั้งสามที่กล่าวมานั้นมีปัญหา overfitting จึงทำให้ตัวแบบที่มีประสิทธิภาพที่ดีที่สุดคือ SVM ที่ใช้ตัวแปรที่มีค่าความสำคัญสูงสุด 10 อันดับแรก) โดยมีค่า R^2 อยู่ที่ 0.4949 และ0.4414 สำหรับชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบตามลำดับ โดยตัวแปรที่มีความสำคัญเป็นอย่างมากต่อการทำนายราคาที่พัก คือ จำนวนคนสูงสุดที่ที่พักรองรับได้ (accommodates), จำนวนห้องนอน (bedrooms) และ การบ่งบอกว่าที่พักเป็นเขตท่องเที่ยว (Tourist District) โดยเป็นการบ่งบอกถึงว่าขนาดของที่พักและโลเคชันของที่พักมีผลกระทบต่อราคาของที่พักเป็นอย่างมาก

5.2 ปัญหาในการทำงานค้นคว้าอิสระ ตามกระบวนการวิทยาศาสตร์ข้อมูล และการปรับปรุง

5.2.1 การเก็บข้อมูล

ข้อมูลที่รวบรวมมาเป็นข้อมูลที่ได้มาแหล่งข้อมูลที่ไม่เป็นทางการที่มีชื่อว่า insideairbnb ซึ่งอาจทำให้ผลลัพธ์ที่ได้จากใช้ข้อมูลนี้มีความคลาดเคลื่อนได้

5.2.2 การจัดการข้อมูล

การสร้าง คอลัมน์ Tourist District มีหลักการในการเลือกเขตจากสถานที่ท่องเที่ยวชื่อดังตามแหล่งออนไลน์ ซึ่งอาจทำให้เกิดความคลาดเคลื่อนได้

5.2.3 การรันข้อมูล

Kernel ของ SVM บางรูปแบบ ใช้เวลาในการรันที่นานจึงทำให้ไม่สมารถประมวลผลได้ และ ก็การรันหาค่า hyperparameter ใช้เวลาในการประมวลผลที่นาน

5.3 ข้อเสนอแนะ

เมื่อพิจารณาจาก feature ที่มีการสร้างเพิ่มเติมคือ Tourist District_tourist distirct ที่บ่งบอกถึงโลเคชันของที่พักว่าอยู่ใกล้กับแลนด์มาร์กหรือสถานที่ท่องเที่ยวชื่อดังในกรุงเทพ พบว่า เป็นตัวแปรที่มีค่า importance ติด อันดับสูงสุด 3 อันดับแรกในทุกตัวแบบ ดังนั้นหากผู้มีสนใจศึกษาค้นคว้าเพิ่มเติม ควรให้ความสำคัญกับการสร้าง feature ใหม่สำหรับทำนายราคาที่พักพร้อมกับการหาตัวแบบใหม่ที่ สามารถทำนายได้แม่นยำมากยิ่งขึ้น รวมไปถึงการเลือกและการปรับจูนค่า hyperparameter ให้รับมือปัญหา overfitting

บรรณานุกรม

กิตตินราดร, ช. (2563, มกราคม). **Support Vector Machines**. Retrieved from https://guopai.github.io/ml-blog08.html

ฐานเศรษฐกิจ. (2023, June 20**). เจ๋ง! "กรุงเทพ"ติดอันดับ 1 เมืองที่นักท่องเที่ยวจองมาพักมากที่สุดในโลก**. Retrieved from ฐานเศรษฐกิจ: https://www.thansettakij.com/business/tourism/568579

ทีมข่าวคอร์ปอเรท-การตลาด กรุงเทพธุรกิจ. (2023, October 7). **วิจัย 'อ็อกซ์ฟอร์ด' ชี้กิจกรรมเกี่ยวกับ Airbnb ทำรายได้ 3 หมื่นล้านในไทยปี 65**. Retrieved from Bangkok Biz: https://www.bangkokbiznews.com/business/business/1092576

Chen, T., & Guestrin, C. (2016, August 13). **XGBoost: A Scalable Tree Boosting System**. KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794. doi:https://doi.org/10.1145/2939672.2939785

Chen, Y., Xue, R., & Yu, Z. (2021). **House price prediction based on machine learning and deep learning methods**. 2021 International Conference on Electronic Information Engineering and Computer Science (EIECS), 699-702.

Corinna, C., & Vladimir, V. (1995). **Support-Vector Networks. Machine Learning**, 20, 273-297. Retrieved from https://link.springer.com/article/10.1007/BF00994018

cway investment. (2018, October 6). **Ensemble Learning Method**. Retrieved from medium: https://medium.com/cw-quantlab/ensemble-learning-method-98359636adf9

Data Innovation and Governance Institute. (2022, August 22). **เจาะลึก "Linear Regression" คืออะไร พร้อมตัวอย่างง่ายๆใน excel**. Retrieved from https://digi.data.go.th/en/blog/linear-regression-en/

Géron, A. (2019). **Hands-On-Machine-Learning-with-Scikit-Learn-Keras-and-Tensorflow**. O'Reilly Media.

Komkid. (2015, May 31). **มาทำความรู้จัก Airbnb สตาร์ทอัพการแบ่งปันที่พักชื่อดัง**. Retrieved from https://startitup.in.th/introducing-airbnb-startup/

Mahypub, M., Ataby, A. A., Upadhyay, Y., & Mustafina, J. (2023). **AIRBNB Price Prediction Using Machine Learning**. 2023 15th International Conference on Developments in eSystems Engineering (DeSE), 166-171.

Minaphinant, V. (2018, February 28). **Machine Learning คืออะไร?** Retrieved from https://medium.com/investic/machine-learning-คืออะไร-fa8bf6663c07

Mohammed , J. K., Aliyu, A. A., Dzukog, A. U., & Olawale, A. A. (2021). **The Impact of COVID-19 on HousingMarket**: A Review of Emerging Literature. InternationalJournalofRealEstateStudie, 66-74.

Perez-Sanchez, V. R., Serrano-Estrada, L., & Marti, P. (2018). **The What, Where, and Why of Airbnb**. Sustainability, 10(12)(4596). doi:https://doi.org/10.3390/su10124596

Ruangsujitwat, N. (2020, January 30). **Sharing Economy คืออะไร?** Retrieved from medium: https://medium.com/@nuttakitruangsujitwat/sharing-economy-คืออะไร-8479f9b23604

Satangmongkol, K. (2019, March 30). **อธิบาย 10 Metrics พื้นฐานสำหรับวัดผลโมเดล Machine Learning**. Retrieved from https://datarockie.com/blog/top-ten-machine-learning-metrics

Tarik, D., & Pekin, O. (2017). **What do guests value most in Airbnb**. Boston Hospitality Review, 5(2), 1-12.

Tewari, U. (2021, November 10). **Regularization — Understanding L1 and L2 regularization for Deep Learning**. Retrieved from Medium: https://medium.com/analytics-vidhya/regularization-understanding-l1-and-l2-regularization-for-deep-learning-a7b9e4a409bf

Tranmer, M., Murphy, J., Elliot, M., & Pampaka, M. (2020). **Multiple Linear Regression**. Cathie Marsh Institute Working Paper 2020-01. Retrieved from https://hummedia.manchester.ac.uk/institutes/cmist/archive-publications/working-papers/2020/multiple-linear-regression.pdf

Visitora-at, P. (2019, October 29). **Metrics พื้นฐานสำหรับวัดประสิทธิภาพของโมเดล Machine Learning**. Retrieved from https://medium.com/@Porntivi/metrics-พื้นฐานสำหรับวัดประสิทธิภาพของโมเดล-machine-learning-c00fcc32fa30

Wongjantorn, P. (2020, February 20). **มารู้จัก Sharing Economy กัน!!** Retrieved from https://medium.com/@panusak.wong/มารู้จัก-sharing-economy-กัน-831ab88dc0a2

Yang, S. (2021). **Learning-based Airbnb Price Prediction Model**. 2021 2nd International Conference on E-Commerce and Internet Technology (ECIT), 283-288.

Zhu, A., Li, R., & Xie, Z. (2020). **Machine Learning Prediction of New York Airbnb**. 2020 Third International Conference on Artificial Intelligence for Industries (AI4I), 1-5.

ภาคผนวก

ภาคผนวก ก

การจัดการข้อมูล

1) ขั้นตอนในการจัดการข้อมูล

Microsoft Excel

1. ทำการตัวแปรที่ไม่จำเป็นออกไป 32 ตัวแปร แล้วข้อมูลจะเหลือ 43 ตัวแปรและ 20823 ตัวอย่าง

2. ทำการนำข้อมูล n/a ของตัวแปร host response time และ host acceptance rate ออก

3. ทำการลบข้อมูล ว่าง ของ ตัวแปร host is superhost

4. ทำการแปลง คอลัมน์ verify 1 คอลัมน์ เป็น 3 คอลัมน์ ใหม่ที่มีชื่อตามคำตอบของคอลัมน์เดิม(email,phone,work_email)

5. ทำการเพิ่ม คอลัมน์ tourist district เข้าไป โดย จะเป็นคอลัมน์ที่มีค่า คือ Tourist Distirct และ Non-tourist distirct โดย จะมีหลักเกณฑ์ว่าที่พักที่อยู่ที่ใกล้แหล่งท่องเที่ยวและแลนด์มาร์กของกรุงเทพจะได้คำตอบเป็น Tourist District

- Chatu Chak
- Phra Nakhon
- Samphanthawong
- Pra Wet
- Parthum Wan
- Bang Rak
- Vadhana
- Ratchathewi
- Khlong Toei
- Klong San

6. ทำการ extract ตัวเลข และ ตัวอักษรใน คอลัมน์ bathroom_textเป็น 2 คอลัมน์ โดยให้ตัวอักษรเป็นคอลัมน์ใหม่ที่ชื่อว่า bath_unit

7. ทำการลบข้อมูล ว่าง ของ ตัวแปร bed , review rating, bedroom, email, phone, work_email

Python

1. ทำการdrop duplicate / ทำการstandardize คอลัมน์ที่เป็นตัวเลขทั้งหมด

2. ทำการนำoutlier ออกจากคอลัมน์ ราคา (ตัวแปรอิสระ)

3. ทำการ log transformation คอลัมน์ ราคา

4. ทำการดึงค่าในคอลัมน์ amenities เฉพาะ amenities ที่มีจำนวนเกินครึ่งของจำนวนชุดข้อมูล

5. ทำการแบ่งประเภทของค่าที่ได้จากคอลัมน์ amenities เป็น 4 ประเภท

   - Comfort & Basics : ["Air conditioning", "Dedicated workspace", "Long term stays allowed", "Essentials", "Bed linens", "Shampoo", "Hangers", "Hot water"],]
   - Appliances & Technology : ["Microwave", "Refrigerator", "Washer", "Hair dryer", "Iron", "Wifi", "TV"]
   - Safety & Facilities : ["Smoke alarm", "Fire extinguisher", "Free parking on premises", "Elevator", "Kitchen"]
   - Convenience : ["Self check-in", "Cooking basics", "Dishes and silverware"]

6. ทำการdrop missing value และ ทำการคำนวณ correlation matrix

คู่คอลัมน์ที่มีค่าcorrelationมากกว่า 0.8

- [(host_total_listings_count, host_listings_count)
- (availability_60, availability_30),
- (availability_90, availability_30),
- (availability_90, availability_60),

- (review_scores_accuracy, review_scores_rating)
- (review_scores_value, review_scores_rating)
- (review_scores_value, review_scores_accuracy)
- (calculated_host_listings_count, host_listings_count)
- (calculated_host_listings_count, host_total_listings_count)
- (calculated_host_listings_count_entire_homes, host_listings_count)
- (calculated_host_listings_count_entire_homes, host_total_listings_count)
- (calculated_host_listings_count_entire_homes, calculated_host_listings_count)

7. นำคอลัมน์ที่มีค่าcorrelationสูงและคอลัมน์ที่ไม่เกี่ยวข้อง

- host_listings_count
- beds
- availability_90
- availability_60
- review_scores_accuracy
- review_scores_cleanliness
- review_scores_checkin
- review_scores_communication
- review_scores_location, review_scores_value
- calculated_host_listings_count
- calculated_host_listings_count_entire_homes
- calculated_host_listings_count_private_rooms
- calculated_host_listings_count_shared_rooms
- availability_365
- number_of_reviews_ltm
- number_of_reviews_l30d
- reviews_per_month
- neighbourhood_cleansed
- bathrooms_text

- bath_unit
- has_availability
- Amenities
- latitude
- longitude

8. ทำการแทนที่ % ในคอลัมน์ host response rate/ host acceptance rate ด้วยช่องว่าง แล้วนำค่าไปหารด้วย 100

9. ทำการดึงค่าunique value จาก คอลัมน์ property_type และแบ่งประเภทออกเป็น 4 ประเภท

- entire_units : [Entire rental unit, Entire townhouse, Entire condo, Entire loft, Entire home/apt, Entire guesthouse, Entire guest suite, Entire cabin, Entire vacation home, Entire bungalow, Entire cottage, Entire chalet, Entire villa, Tiny home, Entire place, Entire serviced apartment, Entire home]
- rooms_shared : [Room in aparthotel, Room in hotel, Room in boutique hotel, Room in hostel, Room in bed and breakfast, Room in serviced apartment]
- private_rooms : [Private room in rental unit, Private room in hostel, Private room in bed and breakfast, Private room in guesthouse, Private room in condo, Private room in home, Private room in townhouse, Private room in serviced apartment, Private room in loft, Private room in guest suite, Private room in tiny home, Private room in vacation home, Private room in farm stay, Private room in resort, Private room in nature lodge, Private room]
- specialty_accomodation : [Casa particular, Treehouse, Farm stay, Barn, Pension, Shipping container, Nature lodge]

10. one hot dummy คอลัมน์ดังนี้

- host_has_profile_pic (t, f)
- host_identity_verified (t, f)

- host_is_superhost (t, f)
- host_response_time (within a day, within a few hours, within an hour, a few days or more)
- Tourist District (tourist distirct, non-tourist district)
- instant_bookable (t, f)
- category_property

11. เปลี่ยนcolumn object กลายเป็น float
12. ทำการนำคอลัมน์ property_type และ room_type ออก
13. แปลงcolumn ที่เป็น bool ให้กลายเป็น int
14. จะเหลือ ข้อมูล 7924 แถว และ 29 คอลัมน์
15. ทำการเช็กค่า feature price เพื่อนำไปเปรียบกับตัววัดประสิทธิภาพภายหลัง โดยทำการ คัดบอกdataframe ขึ้นมา เพื่อทำการ unlog transformation column price เพื่อมาดูโดยค่า count/mean/std/min/max

2) คอลัมน์ที่ลบออกไป 32 คอลัมน์

id

listing_url

scrape_id

last_scraped

source

name

description

neighborhood_overview

picture_url

host_id

host_url

host_name

host_since

host_location

host_about

host_thumbnail_url

host_picture_url

host_neighbourhood

neighbourhood

neighbourhood_group_cleansed

bathrooms

minimum_minimum_nights

maximum_minimum_nights

minimum_maximum_nights

maximum_maximum_nights

minimum_nights_avg_ntm

maximum_nights_avg_ntm

calendar_updated

calendar_last_scraped

license

first_review

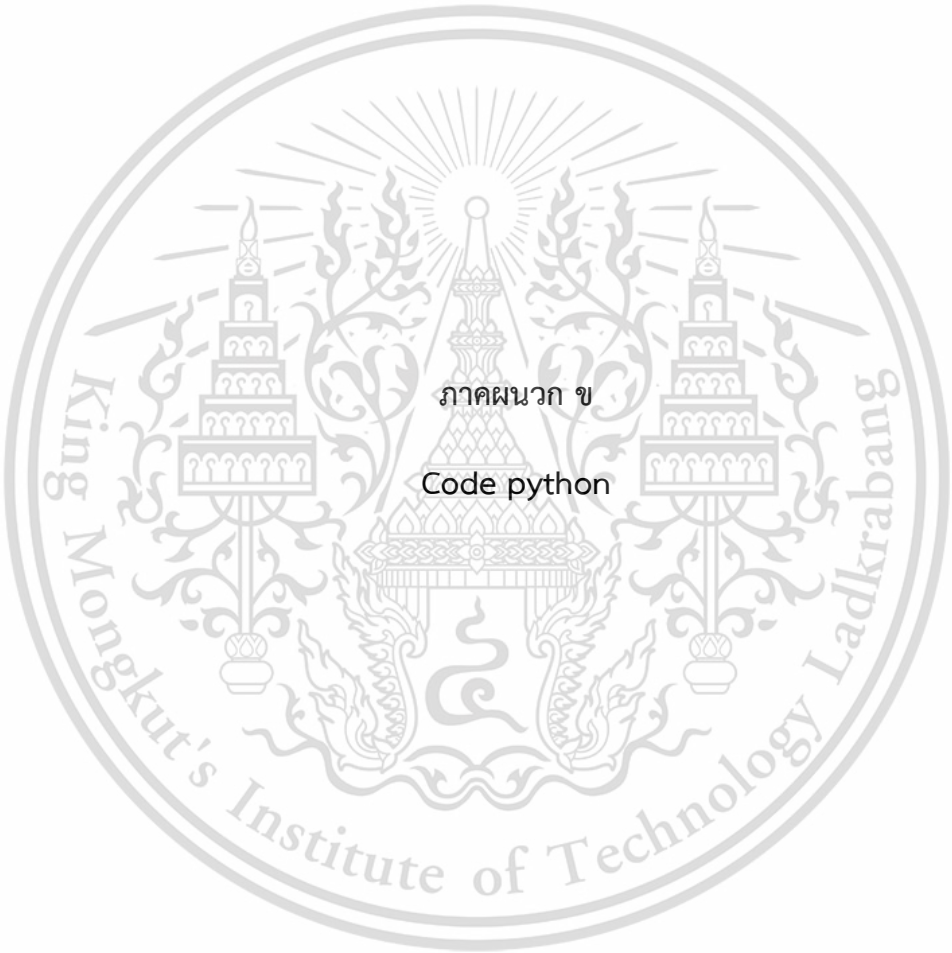
last_review

3) แหล่งอ้างอิงสำหรับการสร้างรายชื่อเขตท่องเที่ยวของ feature tourist district

https://www.lonelyplanet.com/articles/top-things-to-do-in-bangkok

https://www.timeout.com/bangkok/things-to-do/best-things-to-do-in-bangkok

https://www.timeout.com/bangkok/things-to-do/best-things-to-do-in-bangkok



ภาคผนวก ข

Code python

1) ขั้นตอนการจัดการข้อมูล

1.1) Import library ที่จำเป็น

```
!pip install pandas openpyxl
!pip install shap
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from XGBoost import XGBRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from collections import Counter
import re
import XGBoost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
from sklearn.linear_model import Lasso
from sklearn.preprocessing import StandardScaler
from XGBoost import XGBRegressor
from sklearn.model_selection import GridSearchCV, cross_val_score
```

```
from sklearn.metrics import r2_score

import shap

from sklearn.inspection import permutation_importance

import ast

import statsmodels.api as sm

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score

from sklearn.metrics import mean_absolute_error
```

1.2) Import ไฟล์ excel เข้ามา

```
file_path = 'ISs.xlsx'

df = pd.read_excel(file_path)
```

ทำการเช็กและdrop duplicates

```
df = df.drop_duplicates()
```

1.3) ทำการ standardize คอลัมน์ที่เป็น numerical

```
categorical_cols = df.select_dtypes(include=['object', 'category']).columns

numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns

numerical_cols = numerical_cols.drop('price')

scaler = StandardScaler()
```

```
df_scaled = df.copy()

df_scaled[numerical_cols] = scaler.fit_transform(df[numerical_cols])

scaler = StandardScaler()

df_scaled = df.copy()

df_scaled[numerical_cols] = scaler.fit_transform(df[numerical_cols])

df=df_scaled
```

1.4) หาและนำ outlier ออกจาก column price(ตัวแปรอิสระ)

```
# Calculate Q1 and Q3

Q1 = df['price'].quantile(0.25)

Q3 = df['price'].quantile(0.75)

# Calculate the IQR

IQR = Q3 - Q1

# Define the lower and upper bounds for outliers

lower_bound = Q1 - 1.5 * IQR

upper_bound = Q3 + 1.5 * IQR

df = df[(df['price'] >= lower_bound) & (df['price'] <= upper_bound)]

print(df)
```

1.5) ทำการlog transformation คอลัมน์ price

```
df['price'] = np.log1p(df['price'])
```

```
df_scaled=df
```

1.6) หาamenities ที่มีจำวนครั้งมากกว่าครึ่งนึงของข้อมูลทั้งหมด

```
# Extract the 'Amenities' column
amenities_series = df_scaled['Amenities']
# Initialize a counter to count occurrences of each amenity
amenities_counter = Counter()
# Process each row in the amenities column
for amenities in amenities_series:
    # Convert the string representation of the list to an actual list
    amenities_list = ast.literal_eval(amenities)
    # Update the counter with the amenities in this row
    amenities_counter.update(amenities_list)
# Determine the threshold for the minimum number of occurrences
threshold = len(df_scaled) / 2
# Filter the amenities that meet or exceed the threshold
frequent_amenities = [amenity for amenity, count in amenities_counter.items() if
count >= threshold]
frequent_amenities.sort()
frequent_amenities
```

1.7) แบ่ง amenitites ที่ได้มาเป็น category

```
# Define categories and the amenities that belong to them

categories =  {"Comfort & Basics": ["Air conditioning", "Dedicated workspace", "Long term stays allowed", "Essentials", "Bed linens", "Shampoo", "Hangers", "Hot water"],

    "Appliances & Technology": ["Microwave", "Refrigerator", "Washer", "Hair dryer", "Iron", "Wifi", "TV"],

    "Safety & Facilities": ["Smoke alarm", "Fire extinguisher", "Free parking on premises", "Elevator", "Kitchen"],

    "Convenience": ["Self check-in", "Cooking basics", "Dishes and silverware"]

}

# Create a new DataFrame for one-hot encoding

encoded_data = pd.DataFrame()

# Process each category

for category, amenities in categories.items():

    # Create a column for each category, initially set to 0

    encoded_data[category] = 0

    # Update the column based on the presence of any of the category's amenities

    for amenity in amenities:

        encoded_data[category] = encoded_data[category] | df_scaled['Amenities'].apply(lambda x: amenity in x)

# Convert boolean to integer for one-hot encoding

encoded_data = encoded_data.astype(int)
```

```
# Display the first few rows of the new DataFrame

encoded_data.head()
```

1.8) รวมone hot encoded dataframe เข้ากับ dataframe อันเก่า

```
# Concatenating the one-hot encoded DataFrame with the original DataFrame

df_combined = pd.concat([df_scaled, encoded_data], axis=1)

# Displaying the first few rows of the combined DataFrame

print(df_combined.head())
```

1.9) ทำการdrop missing values

```
# Remove rows with any missing values

df_cleaned = df.dropna()
```

1.10) ทำการหาcorrelation matrix

```
# Select only numeric columns from the DataFrame

numeric_df = df.select_dtypes(include=[np.number])

# Calculate the correlation matrix on the numeric columns only

corr_matrix = numeric_df.corr()

# Initialize a list to hold pairs of highly correlated features

high_corr_pairs = []

# Iterate over the correlation matrix
```

```
for i in range(len(corr_matrix.columns)):
    for j in range(i):
        if abs(corr_matrix.iloc[i, j]) > 0.8:
            col_pair = (corr_matrix.columns[i], corr_matrix.columns[j])
            high_corr_pairs.append(col_pair)
# Print or return the list of highly correlated column pairs
print(high_corr_pairs)
```

1.10) ทำการdrop columnที่มีค่า correlation สูงและcolumnที่ไม่จำเป็น

```
columns_to_drop = ['host_listings_count', 'beds', 'availability_90',
'availability_60','review_scores_accuracy', 'review_scores_cleanliness',
    'review_scores_checkin', 'review_scores_communication',
    'review_scores_location', 'review_scores_value','calculated_host_listings_count',
    'calculated_host_listings_count_entire_homes',
    'calculated_host_listings_count_private_rooms',
     'calculated_host_listings_count_shared_rooms', 'availability_365',
'number_of_reviews_ltm',
     'number_of_reviews_l30d','reviews_per_month','neighbourhood_cleansed',
'bathrooms_text', 'bath_unit', 'has_availability', 'Amenities','latitude','longitude']
df.drop(columns_to_drop, axis=1, inplace=True)
```

1.11) แปลงcolumn host_response_rate และ host_acceptance_rate ให้กลายเป็นตัวเลข (50% กลายเป็น 0.05)

```
# Replace percentage sign with nothing and convert to float

df.loc[:, 'host_response_rate'] = df['host_response_rate'].str.replace('%', '').astype(float)

# Convert to fraction

df.loc[:, 'host_response_rate'] = df['host_response_rate'] / 100

# Replace percentage sign with nothing and convert to float

df.loc[:, 'host_acceptance_rate'] = df['host_acceptance_rate'].str.replace('%', '').astype(float)

# Convert to fraction

df.loc[:, 'host_acceptance_rate'] = df['host_acceptance_rate'] / 100
```

1.12) หาunique value ในcolumn property_type

```
unique_values = df['property_type'].unique()

unique_count = df['property_type'].nunique()

print(f"Unique Values in 'property_type': {unique_values}")
```

1.13) แบ่ง ค่าในproperty_type เป็น category

```
# Define Categories

entire_units = ['Entire rental unit', 'Entire townhouse', 'Entire condo', 'Entire loft', 'Entire home/apt', 'Entire guesthouse', 'Entire guest suite', 'Entire cabin', 'Entire vacation
```

```
home', 'Entire bungalow', 'Entire cottage', 'Entire chalet', 'Entire villa', 'Tiny home', 'Entire place', 'Entire serviced apartment', 'Entire home']

rooms_shared = ['Room in aparthotel', 'Room in hotel', 'Room in boutique hotel', 'Room in hostel', 'Room in bed and breakfast', 'Room in serviced apartment']

private_rooms = ['Private room in rental unit', 'Private room in hostel', 'Private room in bed and breakfast', 'Private room in guesthouse', 'Private room in condo', 'Private room in home', 'Private room in townhouse', 'Private room in serviced apartment', 'Private room in loft', 'Private room in guest suite', 'Private room in tiny home', 'Private room in vacation home', 'Private room in farm stay', 'Private room in resort', 'Private room in nature lodge', 'Private room']

specialty_accomodation = ['Casa particular', 'Treehouse', 'Farm stay', 'Barn', 'Pension', 'Shipping container', 'Nature lodge']

# Categorization Function

def categorize_property_type(property_type):

    if property_type in entire_units:

        return 'Entire Unit'

    elif property_type in rooms_shared:

        return 'Room or Shared Space'

    elif property_type in private_rooms:

        return 'Private Room'

    elif property_type in specialty_accomodation:

        return 'Specialty property_type'

    else:
```

```
        return 'Other'

# Apply the Function

df['category_property'] = df['property_type'].apply(categorize_property_type)

# Display the DataFrame

print(df)
```

1.14) ทำdummy variable

```
#get dummies

df_encoded = pd.get_dummies(df, columns=['host_has_profile_pic',
'host_identity_verified','host_is_superhost','host_response_time','Tourist District',
'instant_bookable','category_property'],drop_first=True)
```

1.15) เปลี่ยนcolumn object กลายเป็น float

```
# Select columns that are of type 'object'

object_columns = df_encoded.select_dtypes(include=['object']).columns

print(object_columns)

# Convert all columns of type 'object' that can be converted to 'float'

for col in df_encoded.select_dtypes(include=['object']).columns:
    try:
        df_encoded[col] = df_encoded[col].astype(float)
    except ValueError:
```

```
        pass  # This column cannot be converted to float

# Check the new data types

print(df_encoded.dtypes)
```

drop column property_type และ room_type

```
feature_to_drop = ['property_type', 'room_type']

# Drop specified columns

df_encoded = df_encoded.drop(feature_to_drop, axis=1)

print(df_encoded)
```

1.17) แปลงcolumn ที่เป็น bool ให้กลายเป็น int

```
for column in df_encoded.columns:
    if df_encoded[column].dtype == 'bool':
        df_encoded[column] = df_encoded[column].astype(int)
```

2). การรันตัวแบบต่างๆ

2.1) การรันตัวแบบ XGBoost

2.1.1) การแบ่ง train test data

```
#select feature not 0

Xq = df_encoded.drop(columns = 'price')

yq = df_encoded['price']

# Split data
```

```
Xq_train, Xq_test, yq_train, yq_test = train_test_split(Xq, yq, test_size=0.2, random_state=42)
```

2.1.2) การหาค่า best hyperparameter ผ่าน gridsearchcv

```
# Define the model
model_2_0 = XGBRegressor()
# Define the grid of hyperparameters to search
param_grid = {
    'colsample_bytree': [0.1, 0.3, 0.5, 0.7, 0.9],
    'learning_rate': [0.001, 0.01, 0.05, 0.1, 0.2],
    'max_depth': [3, 5, 7, 10, 15],
    'alpha': [0, 1, 5, 10, 15],
    'n_estimators': [50, 100, 200, 300, 500]
}
# Setup the grid search
grid_search = GridSearchCV(estimator=model_2_0, param_grid=param_grid, cv=3, scoring='neg_mean_squared_error', verbose=1)
# Perform the grid search
grid_search.fit(Xq_train, yq_train)
# Best parameters
print("Best parameters:", grid_search.best_params_)
```

2.1.3) การหาค่า feature importance ของแต่ละตัวแปรสำหรับตัวแบบ XGBoost

```
# Select features and target

Xa = df_encoded.drop(columns ='price')

ya = df_encoded['price']

# Split data

Xa_train, Xa_test, ya_train, ya_test = train_test_split(Xa, ya, test_size=0.2, random_state=42)

# XGBoost regression model

model_1_f = xgb.XGBRegressor(objective='reg:squarederror', alpha= 0, colsample_bytree= 0.5, learning_rate= 0.05, max_depth= 10, n_estimators= 500)

# Fit the model

model_1_f.fit(Xa_train, ya_train)

feature_importances = model_1_f.feature_importances_  # This line is model-specific and might need to be adjusted for a neural network

# Filtering out features with 0.0 importance and including their importance values

filtered_features_with_importance = [(feature, importance) for feature, importance in zip(Xa.columns, feature_importances) if importance > 0.0]

# Printing each feature with its importance

for feature, importance in filtered_features_with_importance:

    print(f"{feature}: {importance}")

threshold = 0 # Example threshold
```

```
important_features = [(feature, importance) for feature, importance in zip(Xa.columns, feature_importances) if importance > threshold]

print(important_features)

threshold = 0

filtered_features = [feature for feature, importance in zip(Xa.columns, feature_importances) if importance > threshold]

print(filtered_features)

# Sorting the filtered features by importance

sorted_filtered_features = sorted(important_features, key=lambda x: x[1], reverse=True)

# Corrected: Extracting names of top 5 features from the filtered list

top_5_feature_names = [feature[0] for feature in sorted_filtered_features[:5]]

# Extracting names of bottom 5 features from the filtered list

bottom_5_feature_names = [feature[0] for feature in sorted_filtered_features[-5:]]

# Sorting the filtered features by importance

sorted_filtered_features = sorted(important_features, key=lambda x: x[1], reverse=True)

# Extracting names of top 10 features from the filtered list

top_10_feature_names = [feature[0] for feature in sorted_filtered_features[:10]]

# Extracting names of bottom 10 features from the filtered list

bottom_10_feature_names = [feature[0] for feature in sorted_filtered_features[-10:]]

print(top_5_feature_names)
```

```
print(top_10_feature_names)
```

2.1.4) การหา best hyperpameter สำหรับตัวแบบ XGBoost (filtered feature)

```
#select feature not 0

Xb = df_encoded[filtered_features]

yb = df_encoded['price']

# Split data

Xb_train, Xb_test, yb_train, yb_test = train_test_split(Xb, yb, test_size=0.2,
random_state=42)

# Define the model

model_2_0 = XGBRegressor()

# Define the grid of hyperparameters to search

param_grid = {

    'colsample_bytree': [0.1, 0.3, 0.5, 0.7, 0.9],

    'learning_rate': [0.001, 0.01, 0.05, 0.1, 0.2],

    'max_depth': [3, 5, 7, 10, 15],

    'alpha': [0, 1, 5, 10, 15],

    'n_estimators': [50, 100, 200, 300, 500]

}

# Setup the grid search
```

```
grid_search = GridSearchCV(estimator=model_2_0, param_grid=param_grid, cv=3, scoring='neg_mean_squared_error', verbose=1)

# Perform the grid search

grid_search.fit(Xb_train, yb_train)

# Best parameters

print("Best parameters:", grid_search.best_params_)
```

2.1.5) การหา best hyperpameter สำหรับตัวแบบ XGBoost (top 5)

```
#select top 5 feature

Xb_5  = df_encoded[top_5_feature_names]

yb_5 = df_encoded['price']

# Split data

Xb_5_train, Xb_5_test, yb_5_train, yb_5_test = train_test_split(Xb_5 , yb_5, test_size=0.2, random_state=42)

# Define the model

model_2_5 = XGBRegressor()

# Define the grid of hyperparameters to search

param_grid = {

    'colsample_bytree': [0.1, 0.3, 0.5, 0.7, 0.9],

    'learning_rate': [0.001, 0.01, 0.05, 0.1, 0.2],

    'max_depth': [3, 5, 7, 10, 15],
```

```
    'alpha': [0, 1, 5, 10, 15],
    'n_estimators': [50, 100, 200, 300, 500]
}
# Setup the grid search
grid_search_5 = GridSearchCV(estimator=model_2_5, param_grid=param_grid, cv=3, scoring='neg_mean_squared_error', verbose=1)
# Perform the grid search
grid_search_5.fit(Xb_5_train, yb_5_train)
# Best parameters
print("Best parameters:", grid_search.best_params_)
```

2.1.6) การหา best hyperpameter สำหรับตัวแบบ XGBoost (top 10)

```
#select top 10
Xb_10 = df_encoded[top_10_feature_names]
yb_10 = df_encoded['price']
# Split data
Xb_10_train, Xb_10test, yb_10_train, yb_10_test = train_test_split(Xb_10, yb_10 , test_size=0.2, random_state=42)
# Define the model
model_2_10 = XGBRegressor()
# Define the grid of hyperparameters to search
```

```
param_grid = {
    'colsample_bytree': [0.1, 0.3, 0.5, 0.7, 0.9],
    'learning_rate': [0.001, 0.01, 0.05, 0.1, 0.2],
    'max_depth': [3, 5, 7, 10, 15],
    'alpha': [0, 1, 5, 10, 15],
    'n_estimators': [50, 100, 200, 300, 500]
}
# Setup the grid search
grid_search_10 = GridSearchCV(estimator=model_2_10, param_grid=param_grid, cv=3,
scoring='neg_mean_squared_error', verbose=1)
# Perform the grid search
grid_search_10.fit(Xb_10_train, yb_10_train)
# Best parameters
print("Best parameters:", grid_search.best_params_)
```

2.1.7) การวัดประสิทธิภาพตัวแบบสำหรับตัวแบบ XGBoost (filtered feature)

```
# Assuming df is your DataFrame
# Select features and target
XX = df_encoded[filtered_features]
yy = df_encoded['price']
# Split data
```

```
XX_train, XX_test, yy_train, yy_test = train_test_split(XX, yy, test_size=0.2, random_state=42)

# XGBoost regression model

model_1_0 = xgb.XGBRegressor(objective='reg:squarederror', alpha= 0, colsample_bytree= 0.5, learning_rate= 0.05, max_depth= 10, n_estimators= 500)

# Fit the model

model_1_0.fit(XX_train, yy_train)

# Predictions for the training set

yy_train_pred = model_1_0.predict(XX_train)

yy_train_pred_unlog = np.exp(yy_train_pred)  # Unlog predictions

yy_train_unlog = np.exp(yy_train)            # Unlog actual values

# Evaluate on the training set

mse_train_unlog = mean_squared_error(yy_train_unlog, yy_train_pred_unlog)

print("Mean Squared Error (unlogged) for the training set: ", mse_train_unlog)

# Predictions for the test set

yy_pred = model_1_0.predict(XX_test)

yy_pred_unlog = np.exp(yy_pred)  # Unlog predictions

yy_test_unlog = np.exp(yy_test)  # Unlog actual values

# Evaluate on the test set

mse_test_unlog = mean_squared_error(yy_test_unlog, yy_pred_unlog)

print("Mean Squared Error (unlogged) for the test set: ", mse_test_unlog)
```

```
# R^2 score for the training set (unlogged)

r2_train_unlog = r2_score(yy_train_unlog, yy_train_pred_unlog)

print("R-squared (unlogged) for the training set: ", r2_train_unlog)

# R^2 score for the test set (unlogged)

r2_test_unlog = r2_score(yy_test_unlog, yy_pred_unlog)

print("R-squared (unlogged) for the test set: ", r2_test_unlog)

rmse_train_unlog = np.sqrt(mse_train_unlog)

rmse_test_unlog = np.sqrt(mse_test_unlog)

print("Root Mean Squared Error (unlogged) for the train set: ", rmse_train_unlog)

print("Root Mean Squared Error (unlogged) for the test set: ", rmse_test_unlog)

# Evaluate on the training set using MAE

mae_train_unlog = mean_absolute_error(yy_train_unlog, yy_train_pred_unlog)

print("Mean Absolute Error (unlogged) for the training set: ", mae_train_unlog)

# Evaluate on the test set using MAE

mae_test_unlog = mean_absolute_error(yy_test_unlog, yy_pred_unlog)

print("Mean Absolute Error (unlogged) for the test set: ", mae_test_unlog)
```

2.1.8) การวัดประสิทธิภาพตัวแบบสำหรับตัวแบบ XGBoost (top 5)

```
#select 5 features

XX_5 = df_encoded[top_5_feature_names]

yy_5 = df_encoded['price']
```

```
# Split data

XX_5_train, XX_5_test, yy_5_train, yy_5_test = train_test_split(XX_5, yy_5, test_size=0.2, random_state=42)

# XGBoost regression model

model_1_5 = xgb.XGBRegressor(objective='reg:squarederror', alpha= 0, colsample_bytree= 0.5, learning_rate= 0.05, max_depth= 10, n_estimators= 500)

# Fit the model

model_1_5.fit(XX_5_train, yy_5_train)

# Predictions for the training set

yy_5_train_pred = model_1_5.predict(XX_5_train)

yy_5_train_pred_unlog = np.exp(yy_5_train_pred)  # Unlog predictions

yy_5_train_unlog = np.exp(yy_5_train)            # Unlog actual values

# Evaluate on the training set

mse_5_train_unlog = mean_squared_error(yy_5_train_unlog, yy_5_train_pred_unlog)

print("Mean Squared Error (unlogged) for the training set: ", mse_5_train_unlog)

# Predictions for the test set

yy_5_pred = model_1_5.predict(XX_5_test)

yy_5_pred_unlog = np.exp(yy_5_pred)  # Unlog predictions

yy_5_test_unlog = np.exp(yy_5_test)  # Unlog actual values

# Evaluate on the test set

mse_5_test_unlog = mean_squared_error(yy_5_test_unlog, yy_5_pred_unlog)
```

```
print("Mean Squared Error (unlogged) for the test set: ", mse_5_test_unlog)

# R^2 score for the training set (unlogged)

r2_5_train_unlog = r2_score(yy_5_train_unlog, yy_5_train_pred_unlog)

print("R-squared (unlogged) for the training set: ", r2_5_train_unlog)

# R^2 score for the test set (unlogged)

r2_5_test_unlog = r2_score(yy_5_test_unlog, yy_5_pred_unlog)

print("R-squared (unlogged) for the test set: ", r2_5_test_unlog)

rmse_5_train_unlog = np.sqrt(mse_5_train_unlog)

rmse_5_test_unlog = np.sqrt(mse_5_test_unlog)

print("Root Mean Squared Error (unlogged) for the train set: ", rmse_5_train_unlog)

print("Root Mean Squared Error (unlogged) for the test set: ", rmse_5_test_unlog)

# Evaluate on the training set using MAE

mae_5_train_unlog = mean_absolute_error(yy_5_train_unlog, yy_5_train_pred_unlog)

print("Mean Absolute Error (unlogged) for the training set: ", mae_5_train_unlog)

# Evaluate on the test set using MAE

mae_5_test_unlog = mean_absolute_error(yy_5_test_unlog, yy_5_pred_unlog)

print("Mean Absolute Error (unlogged) for the test set: ", mae_5_test_unlog)
```

2.1.9) การวัดประสิทธิภาพตัวแบบสำหรับตัวแบบ XGBoost (top 10)

```
#select top 10

XX_10 = df_encoded[top_10_feature_names]
```

```
yy_10 = df_encoded['price']

# Split data

XX_10_train, XX_10_test,  yy_10_train,  yy_10_test = train_test_split(XX_10, yy_10, test_size=0.2, random_state=42)

# XGBoost regression model

model_1_10 = xgb.XGBRegressor(objective='reg:squarederror', alpha= 0, colsample_bytree= 0.5, learning_rate= 0.05, max_depth= 10, n_estimators= 500)

# Fit the model

model_1_10.fit(XX_10_train,yy_10_train)

# Predictions for the training set

yy_10_train_pred = model_1_10.predict(XX_10_train)

yy_10_train_pred_unlog = np.exp(yy_10_train_pred)  # Unlog predictions

yy_10_train_unlog = np.exp(yy_10_train)          # Unlog actual values

# Evaluate on the training set

mse_10_train_unlog = mean_squared_error(yy_10_train_unlog, yy_10_train_pred_unlog)

print("Mean Squared Error (unlogged) for the training set: ", mse_10_train_unlog)

# Predictions for the test set

yy_10_pred = model_1_10.predict(XX_10_test)

yy_10_pred_unlog = np.exp(yy_10_pred)  # Unlog predictions

yy_10_test_unlog = np.exp(yy_10_test)  # Unlog actual values

# Evaluate on the test set
```

```
mse_10_test_unlog = mean_squared_error(yy_10_test_unlog, yy_10_pred_unlog)

print("Mean Squared Error (unlogged) for the test set: ", mse_10_test_unlog)

# R^2 score for the training set (unlogged)

r2_10_train_unlog = r2_score(yy_10_train_unlog, yy_10_train_pred_unlog)

print("R-squared (unlogged) for the training set: ", r2_10_train_unlog)

# R^2 score for the test set (unlogged)

r2_10_test_unlog = r2_score(yy_10_test_unlog, yy_10_pred_unlog)

print("R-squared (unlogged) for the test set: ", r2_10_test_unlog)

rmse_10_train_unlog = np.sqrt(mse_10_train_unlog)

rmse_10_test_unlog = np.sqrt(mse_10_test_unlog)

print("Root Mean Squared Error (unlogged) for the train set: ", rmse_10_train_unlog)

print("Root Mean Squared Error (unlogged) for the test set: ", rmse_10_test_unlog)

# Evaluate on the training set using MAE

mae_10_train_unlog = mean_absolute_error(yy_10_train_unlog,
yy_10_train_pred_unlog)

print("Mean Absolute Error (unlogged) for the training set: ", mae_10_train_unlog)

# Evaluate on the test set using MAE

mae_10_test_unlog = mean_absolute_error(yy_10_test_unlog, yy_10_pred_unlog)

print("Mean Absolute Error (unlogged) for the test set: ", mae_10_test_unlog)
```

2.2) การรันตัวแบบ SVM

2.2.1) การ แบ่ง train test data

```
X = df_encoded.drop(columns ='price')

y = df_encoded['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

2.2.1) การหาค่า best hyperparameter สำหรับตัวแบบ SVM

```
from sklearn.svm import SVR

from sklearn.model_selection import GridSearchCV

from sklearn.metrics import mean_squared_error

# Define SVR model

svr = SVR()

# Define a grid of parameters to search over

param_grid = {

    'C': [0.1, 1, 10, 100],  # Regularization parameter

    'gamma': ['scale', 'auto'],  # Kernel coefficient

    'kernel': ['rbf', 'linear'],  # Type of kernel

}

# Grid search for the best parameters

grid_search = GridSearchCV(svr, param_grid, refit=True, verbose=2)

grid_search.fit(X_train, y_train)

# View the best parameters
```

```
print("Best Parameters: ", grid_search.best_params_)

# Predict with the best estimator

best_estimator = grid_search.best_estimator_

y_pred = best_estimator.predict(X_test)

# Calculate Mean Squared Error

mse = mean_squared_error(y_test, y_pred)

print("Mean Squared Error: ", mse)
```

2.2.2) การหา ค่า feature importance ของแต่ละตัวแปรสำหรับตัวแบบ SVM

```
# Assuming best_params, svr_model, X_train, y_train, X_test, and y_test are already defined

best_params_a = grid_search.best_params_

svr_model = SVR(**best_params_a)

svr_model.fit(X_train, y_train)

# Calculate permutation importance

perm_importance = permutation_importance(svr_model, X_test, y_test, n_repeats=30, random_state=42)

# Get the feature names

feature_names = np.array(X.columns)

# Sort features by importance in descending order

sorted_idx = perm_importance.importances_mean.argsort()[::-1]
```

```
# Extract top 5 and top 10 feature names
top_5_features = feature_names[sorted_idx][:5]
top_10_features = feature_names[sorted_idx][:10]
# Also, print the importance values for the top 5 and top 10 features
top_5_importance_values = perm_importance.importances_mean[sorted_idx][:5]
top_10_importance_values = perm_importance.importances_mean[sorted_idx][:10]
print("Top 5 important features and their importance values:")
for feature, importance in zip(top_5_features, top_5_importance_values):
    print(f"{feature}: {importance}")
print("\nTop 10 important features and their importance values:")
for feature, importance in zip(top_10_features, top_10_importance_values):
    print(f"{feature}: {importance}")
# Identify features with importance greater than 0
important_features_indices = np.where(perm_importance.importances_mean > 0)[0]
important_features = feature_names[important_features_indices]
important_features_importance_values = perm_importance.importances_mean[important_features_indices]
# Sort the important features by their importance values in ascending order
ascending_order_indices = important_features_importance_values.argsort()
sorted_important_features = important_features[ascending_order_indices]
```

```
sorted_important_features_importance_values = important_features_importance_values[ascending_order_indices]

# Print the important features with their importance values in ascending order

print("\nFeatures with importance greater than 0 and their importance values (ascending):")

for feature, importance in zip(sorted_important_features, sorted_important_features_importance_values):

    print(f"{feature}: {importance}")

top_5 = ['accommodates','Tourist District_tourist distirct','bedrooms',

 'host_total_listings_count','availability_30']

top_10 = ['accommodates', 'Tourist District_tourist distirct','bedrooms',

 'host_total_listings_count', 'availability_30', 'host_is_superhost_t'

 ,'review_scores_rating', 'number_of_reviews', 'work_email', 'minimum_nights']
```

2.2.3) การหา best hyperpameter และการวัดประสิทธิภาพสำหรับตัวแบบ SVM (filtered feature)

```
Xa = df_encoded[important_features]

ya = df_encoded['price']

Xa_train, Xa_test, ya_train, ya_test = train_test_split(Xa, ya, test_size=0.2, random_state=42)

best_params_ = grid_search.best_params_

# Assume svr_model and X_test_scaled, y_test are already defined
```

```
svr_model_a = SVR(**best_params_)

svr_model_a.fit(Xa_train, ya_train)

ya_train_pred = svr_model_a.predict(Xa_train)

ya_train_pred_unlog = np.exp(ya_train_pred)  # Unlog predictions

ya_train_unlog = np.exp(ya_train)            # Unlog actual values

mse_a_train = mean_squared_error(ya_train_unlog, ya_train_pred_unlog)

r2_a_train = r2_score(ya_train_unlog, ya_train_pred_unlog)

ya_pred = svr_model_a.predict(Xa_test)

ya_pred_unlog = np.exp(ya_pred)

ya_test_unlog = np.exp(ya_test)

mse_a_test = mean_squared_error(ya_test_unlog, ya_pred_unlog)

r2_a_test = r2_score(ya_test_unlog, ya_pred_unlog)

rmse_a_train = np.sqrt(mse_a_train)

rmse_a_test = np.sqrt(mse_a_test)

mae_a_train = mean_absolute_error(ya_train_unlog, ya_train_pred_unlog)

mae_a_test = mean_absolute_error(ya_test_unlog, ya_pred_unlog)

print("best parameter: ", best_params_)

print("Mean Squared Error (unlogged) for the training set: ", mse_a_train)

print("Mean Squared Error (unlogged) for the test set: ", mse_a_test)

print("R-squared (unlogged) for the training set: ", r2_a_train)

print("R-squared (unlogged) for the test set: ", r2_a_test)
```

```
print("Root Mean Squared Error (unlogged) for the train set: ", rmse_a_train)

print("Root Mean Squared Error (unlogged) for the test set: ", rmse_a_test)

print("Mean Absolute Error (unlogged) for the training set: ", mae_a_train)

print("Mean Absolute Error (unlogged) for the test set: ", mae_a_test)
```

2.2.4) การหา best hyperpameter สำหรับตัวแบบ SVM (top 5)

```
X5 = df_encoded[top_5]

y5 = df_encoded['price']

X5_train, X5_test, y5_train, y5_test = train_test_split(X5, y5, test_size=0.2,
random_state=42)

from sklearn.svm import SVR

# Define SVR model

svr = SVR()

# Define a grid of parameters to search over

param_grid = {

    'C': [0.1, 1, 10, 100],  # Regularization parameter

    'gamma': ['scale', 'auto'],  # Kernel coefficient

    'kernel': ['rbf', 'linear'],  # Type of kernel, now including 'poly'

}

# Grid search for the best parameters

grid_search = GridSearchCV(svr, param_grid, refit=True, verbose=2)
```

```
grid_search.fit(X5_train, y5_train)

# View the best parameters

print("Best Parameters: ", grid_search.best_params_)

# Predict with the best estimator

best_estimator = grid_search.best_estimator_

y5_pred = best_estimator.predict(X5_test)

# Calculate Mean Squared Error

mse = mean_squared_error(y5_test, y5_pred)

print("Mean Squared Error: ", mse)
```

2.2.5) การวัดประสิทธิภาพตัวแบบสำหรับตัวแบบ SVM (top 5)

```
Xb = df_encoded[top_5]

yb = df_encoded['price']

Xb_train, Xb_test, yb_train, yb_test = train_test_split(Xb, yb, test_size=0.2,
random_state=42)

best_params_b = grid_search.best_params_

# Assume svr_model and X_test_scaled, y_test are already defined

svr_model_b = SVR(**best_params_b)

svr_model_b.fit(Xb_train, yb_train)

yb_train_pred = svr_model_b.predict(Xb_train)

yb_train_pred_unlog = np.exp(yb_train_pred)  # Unlog predictions
```

```
yb_train_unlog = np.exp(yb_train)            # Unlog actual values
mse_b_train = mean_squared_error(yb_train_unlog, yb_train_pred_unlog)
r2_b_train = r2_score(yb_train_unlog, yb_train_pred_unlog)
yb_pred = svr_model_b.predict(Xb_test)
yb_pred_unlog = np.exp(yb_pred)
yb_test_unlog = np.exp(yb_test)
mse_b_test = mean_squared_error(yb_test_unlog, yb_pred_unlog)
r2_b_test = r2_score(yb_test_unlog, yb_pred_unlog)
rmse_b_train = np.sqrt(mse_b_train)
rmse_b_test = np.sqrt(mse_b_test)
mae_b_train = mean_absolute_error(yb_train_unlog, yb_train_pred_unlog)
mae_b_test = mean_absolute_error(yb_test_unlog, yb_pred_unlog)
print("Mean Squared Error (unlogged) for the training set: ", mse_b_train)
print("Mean Squared Error (unlogged) for the test set: ", mse_b_test)
print("R-squared (unlogged) for the training set: ", r2_b_train)
print("R-squared (unlogged) for the test set: ", r2_b_test)
print("Root Mean Squared Error (unlogged) for the train set: ", rmse_b_train)
print("Root Mean Squared Error (unlogged) for the test set: ", rmse_b_test)
print("Mean Absolute Error (unlogged) for the training set: ", mae_b_train)
print("Mean Absolute Error (unlogged) for the test set: ", mae_b_test)
```

2.2.6) การหา best hyperpameter สำหรับตัวแบบ SVM (top 10)

```
X10 = df_encoded[top_10]

y10 = df_encoded['price']

X10_train, X10_test, y10_train, y10_test = train_test_split(X10, y10, test_size=0.2, random_state=42)

from sklearn.svm import SVR

# Define SVR model

svr = SVR()

# Define a grid of parameters to search over

param_grid = {

    'C': [0.1, 1, 10, 100],  # Regularization parameter

    'gamma': ['scale', 'auto'],  # Kernel coefficient

    'kernel': ['rbf', 'linear'],  # Type of kernel, now including 'poly'

}

# Grid search for the best parameters

grid_search = GridSearchCV(svr, param_grid, refit=True, verbose=2)

grid_search.fit(X10_train, y10_train)

# View the best parameters

print("Best Parameters: ", grid_search.best_params_)

# Predict with the best estimator

best_estimator = grid_search.best_estimator_
```

```
y10_pred = best_estimator.predict(X10_test)

# Calculate Mean Squared Error

mse = mean_squared_error(y10_test, y10_pred)

print("Mean Squared Error: ", mse)
```

2.2.7) การวัดประสิทธิภาพตัวแบบสำหรับตัวแบบ SVM (top 10)

```
Xc = df_encoded[top_10]

yc = df_encoded['price']

Xc_train, Xc_test, yc_train, yc_test = train_test_split(Xc, yc, test_size=0.2,
random_state=42)

best_params_c = grid_search.best_params_

# Assume svr_model and X_test_scaled, y_test are already defined

svr_model_c = SVR(**best_params_c)

svr_model_c.fit(Xc_train, yc_train)

yc_train_pred = svr_model_c.predict(Xc_train)

yc_train_pred_unlog = np.exp(yc_train_pred)  # Unlog predictions

yc_train_unlog = np.exp(yc_train)            # Unlog actual values

mse_c_train = mean_squared_error(yc_train_unlog, yc_train_pred_unlog)

r2_c_train = r2_score(yc_train_unlog, yc_train_pred_unlog)

yc_pred = svr_model_c.predict(Xc_test)

yc_pred_unlog = np.exp(yc_pred)
```

```
yc_test_unlog = np.exp(yc_test)

mse_c_test = mean_squared_error(yc_test_unlog, yc_pred_unlog)

r2_c_test = r2_score(yc_test_unlog, yc_pred_unlog)

rmse_c_train = np.sqrt(mse_c_train)

rmse_c_test = np.sqrt(mse_c_test)

mae_c_train = mean_absolute_error(yc_train_unlog, yc_train_pred_unlog)

mae_c_test = mean_absolute_error(yc_test_unlog, yc_pred_unlog)

print("Mean Squared Error (unlogged) for the training set: ", mse_c_train)

print("Mean Squared Error (unlogged) for the test set: ", mse_c_test)

print("R-squared (unlogged) for the training set: ", r2_c_train)

print("R-squared (unlogged) for the test set: ", r2_c_test)

print("Root Mean Squared Error (unlogged) for the train set: ", rmse_c_train)

print("Root Mean Squared Error (unlogged) for the test set: ", rmse_c_test)

print("Mean Absolute Error (unlogged) for the training set: ", mae_c_train)

print("Mean Absolute Error (unlogged) for the test set: ", mae_c_test)
```

2.3) การรันตัวแบบการถดถอยเชิงเส้น

2.3.1) การหาค่า p-value ของแต่ละตัวแปร

```
X = df_encoded.drop(columns ='price')

y = df_encoded['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Fit a preliminary model to get p-values

X_train_sm = sm.add_constant(X_train)

model_sm = sm.OLS(y_train, X_train_sm).fit()

p_values = model_sm.pvalues

# Convert p-values into a DataFrame for better visualization

p_values_df = pd.DataFrame(p_values, columns=['p_value'])

# Display the p-values for all features

print(p_values_df)
```

2.3.2) การเลือกตัวแปรที่มีค่า p-value ที่ ≤ 0.05 และการวัดประสิทธิภาพของตัวแบบการถดถอยเชิงเส้น

```
# Set a threshold for p-values, e.g., 0.05 and drop features with higher p-values

high_p_value_features = p_values[p_values > 0.05].index.tolist()

low_p_value_features = p_values[p_values <= 0.05].index.tolist()

X_train_reduced = X_train.drop(high_p_value_features, axis=1)

X_test_reduced = X_test.drop(high_p_value_features, axis=1)

print(high_p_value_features)

print(low_p_value_features)

# Re-fit the linear regression model with the reduced set of features

model = LinearRegression()

model.fit(X_train_reduced, y_train)
```

```
# Print evaluation metrics for unlogged values

# Make predictions and evaluate the model

y_pred_train = model.predict(X_train_reduced)

y_pred_unlogged_train = np.exp(y_pred_train)

y_train_unlogged = np.exp(y_train)

# Compute MSE and R-squared for unlogged values

mse_unlogged_train = mean_squared_error(y_train_unlogged, y_pred_unlogged_train)

rmse_unlogged_train = np.sqrt(mse_unlogged_train)

r2_unlogged_train = r2_score(y_train_unlogged, y_pred_unlogged_train)

# Print evaluation metrics for unlogged values

# Make predictions and evaluate the model

y_pred = model.predict(X_test_reduced)

y_pred_unlogged = np.exp(y_pred)

y_test_unlogged = np.exp(y_test)

# Compute MSE and R-squared for unlogged values

mse_unlogged = mean_squared_error(y_test_unlogged, y_pred_unlogged)

rmse_unlogged = np.sqrt(mse_unlogged)

r2_unlogged = r2_score(y_test_unlogged, y_pred_unlogged)

# Compute MAE for unlogged values for the train set

mae_unlogged_train = mean_absolute_error(y_train_unlogged,
y_pred_unlogged_train)
```

```
# Compute MAE for unlogged values for the test set

mae_unlogged_test = mean_absolute_error(y_test_unlogged, y_pred_unlogged)

print(f"Mean Squared Error for train set: {mse_unlogged_train}")

print(f"Root Mean Squared Error for train set: {rmse_unlogged_train}")

print(f"R-squared for train set: {r2_unlogged_train}")

print(f"Mean Squared Error for test set: {mse_unlogged}")

print(f"Root Mean Squared Error for test set: {rmse_unlogged}")

print(f"R-squared for test set: {r2_unlogged}")

print(f"Mean Absolute Error for train set: {mae_unlogged_train}")

print(f"Mean Absolute Error for test set: {mae_unlogged_test}")
```

2.3.3) การหาค่า coefficient ของตัวแบบการถดถอยเชิงเส้น

```
y_train_unlogged = np.exp(y_train)

# Fit the linear regression model with reduced features

model = LinearRegression()

model.fit(X_train_reduced, y_train_unlogged)

# Coefficients and intercept

coefficients = model.coef_

intercept = model.intercept_

# Displaying the coefficients and intercept

print("Intercept:", intercept)
```

```
print("\nCoefficients:")

for feature, coef in zip(X_train_reduced.columns, coefficients):

    print(f"{feature}: {coef}")
```

ประวัติผู้เขียน

ชื่อ	นาย ชยพัทธ์ ณัฐศิววงศ์
วันเกิด	29 มีนาคม พ.ศ.2542
ที่อยู่ปัจจุบัน	19/351 ถ.ริมคลองบางค้อ แขวงบางค้อ เขตจอมทอง กรุงเทพ 10150
ประวัติการศึกษา	(2564) ศิลปศาสตร์บัณฑิต สาขาเศรษฐศาสตร์ เกรดเฉลี่ย 3.604 จุฬาลงกรณ์มหาวิทยาลัย