การจัดหมวดหมู่ประเภทข้อความเพื่อเพิ่มหัวข้อใหม่โดยใช้การเรียนรู้ของเครื่อง

TEXT CLASSIFICATION FOR NEW INTENT USING MACHINE LEARNING

ธนพัฒน์ สอนโส

สหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา
วิทยาศาสตรบัณฑิต (สถิติประยุกต์)
ภาควิชาสถิติ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา2566

TEXT CLASSIFICATION FOR NEW INTENT USING MACHINE LEARNING

Thanapat Sonso

A COOPERATIVE EDUCATION SUBMITTED IN PARTIAL
FULFILLMENT OF THE REQUIREMENT FOR
THE DEGREE OF BACHELOR OF SCIENCE (APPLIED STATISTICS)
DEPARTMENT OF STATISTICS SCHOOL OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2566

หัวข้อสหกิจศึกษา	การจัดหมวดหมู่ประเภทข้อความเพื่อเพิ่มหัวข้อใหม่โดยใช้การเรียนรู้ของเครื่อง Text Classification for New Intent Using Machine Learning
ชื่อนักศึกษา	นายธนพัฒน์ สอนโส รหัสนักศึกษา 63050627
ปริญญา	วิทยาศาสตรบัณฑิต (สถิติประยุกต์)
ภาควิชา	สถิติ
ปีการศึกษา	2566
อาจารย์ที่ปรึกษา	รศ.ดร.วลัยลักษณ์ อัตธีรวงศ์

คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง (สจล.) อนุมัติให้สหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต (สถิติประยุกต์) ประจำปีการศึกษา 2566

คณะกรรมการสอบ	ลายมือชื่อ
ผศ.ดร.ยุวดี กล่อมวิเศษ ประธานกรรมการ	
คุณกีรติพร ส่วนบุญ กรรมการ	
รศ.ดร.วลัยลักษณ์ อัตธีรวงศ์ กรรมการและอาจารย์ที่ปรึกษา	

หัวข้อสหกิจศึกษา การจัดหมวดหมู่ประเภทข้อความเพื่อเพิ่มหัวข้อใหม่โดยใช้การเรียนรู้ของเครื่อง
ชื่อนักศึกษา นายธนพัฒน์ สอนโส รหัสนักศึกษา 63050627
ปริญญา วิทยาศาสตรบัณฑิต (สถิติประยุกต์)
ภาควิชา สถิติ
คณะ วิทยาศาสตร์
มหาวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง (สจล.)
ปีการศึกษา 2566
อาจารย์ที่ปรึกษา รศ.ดร.วลัยลักษณ์ อัตธีรวงศ์

บทคัดย่อ

การเรียนรู้ของเครื่อง เป็นเครื่องมือที่มีประโยชน์ต่อธุรกิจในยุคปัจจุบันอย่างมาก เนื่องจากการเรียนรู้ของเครื่องช่วยลดภาระงานที่ซ้ำซากของพนักงาน การเรียนรู้ของเครื่องจำเป็นอย่างยิ่งที่ต้องจัดสรรงบประมาณเพื่อนำการเรียนรู้ของเครื่องมาปรับใช้ งานวิจัยนี้มีจุดประสงค์เพื่อสร้างหัวข้อใหม่ๆ เพื่อเพิ่มความสามารถแชทบอทของบริษัท A ที่มีหัวข้อเดิมอยู่แล้ว ให้สามารถตอบคำถามของผู้ใช้งานได้หลากหลายมากขึ้น โดยการศึกษาข้อความที่แชทบอทตอบไม่ได้ เพื่อหาหัวข้อใหม่ที่ควรเพิ่ม ผู้วิจัยได้ใช้ข้อความจากปี พ.ศ. 2566 โดยสร้างฐานข้อมูลเป็นข้อความจำนวน 840 ข้อความ และมี 10 หัวข้อ โดยข้อมูลได้ถูกแบ่งเป็นข้อมูลฝึก 80% และข้อมูลทดสอบ 20% โดยใช้แบบจำลอง 3 แบบได้แก่ แบบจำลองแบบป่าสุ่ม แบบจำลองซัพพอร์ตเวกเตอร์แมชชีน และ แบบจำลองซัพพอร์ตเวกเตอร์แมชชีนแบบ Kernel จากนั้นทำการเปรียบเทียบประสิทธิภาพของทั้ง 3 แบบ โดยใช้ ค่าความแม่นยำโดยรวม ค่าความแม่นยำ ค่าความครบถ้วน และค่าประสิทธิภาพโดยรวม ผลลัพธ์แสดงว่า แบบจำลองแบบป่าสุ่ม ทำงานได้ดีที่สุด ต่อมาผู้วิจัยได้ใช้แบบจำลองแบบป่าสุ่ม เพื่อหาค่าเกณฑ์ที่เหมาะสมเพื่อกรองข้อความที่ไม่อยู่ใน 10 หัวข้อที่กำหนด แต่ในระหว่างใช้งานจริงกลับพบปัญหาเล็กน้อย 2 หัวข้อ ดังนั้นนักวิจัยจึงได้ปรับเพิ่มค่าเกณฑ์ สำหรับทั้ง 2 หัวข้อที่ต้องการการกรองข้อความที่เข้มงวดขึ้น ผลลัพธ์แสดงให้เห็นว่าประสิทธิภาพในการใช้งานจริงนั้นมีประสิทธิภาพสูง

คำสำคัญ : แบบจำลองป่าสุ่ม วิธีซัพพอร์ตเวกเตอร์แมชชีน วิธีซัพพอร์ตเวกเตอร์แมชชีนแบบKernel ค่าเกณฑ์

Title	Text Classification for New Intent Using Machine Learning
Students	Mr. Thanapat Sonso Student ID 63050627
Degree	Bachelor of Science (Applied Statistics)
Department	Statistics
School	Science
University	King Mongkut's Institute of TechnologyLadkrabang (KMITL)
Academic Year	2023
Advisor	Assoc.Prof.Dr. Walailak Atthirawong

Abstract

Machine learning is a very useful tool for business today because it doesn't need people to do the same jobs repeatedly. It is crucial to have the ability to allocate sufficient budgets for the implementation of machine learning.The objective of this research is to generate novel subjects to enhance the capability of Company A's bot, which is currently equipped with existing topics, to respond to a wider range of inquiries from human users. It performs investigation using message data that the algorithm is unable to respond to in order to identify potential new topics. By using text data from 2023, the researcher made a data set with 840 characters and 10 topics. Subsequently, the data was partitioned into 80% training data and 20% test data utilizing three models: the Random Forest Model, Support Vector Machines, and Support Vector Machines using Kernel.A comparison was made between the three models' performance using the metrics of Accuracy, Precision , Recall and F-1. The results indicated that the Random Forest model outperformed the other two models.Subsequently, the researcher employed the Random forest model in order to search for a threshold value that would result in the absence of any text that was not included in all ten themes that the researcher had discovered to be included in all ten topics. It turned out that there were two small problems with the real implementation.The researcher therefore increased the threshold values for both classes that had more stringent problems with the text. The results show that the actual implementation of the model is highly effective.

Keywords : Random Forest ,Support Vector Machines, Support Vector Machines using Kernel, Threshold

กิตติกรรมประกาศ

สหกิจศึกษาเรื่องการจัดหมวดหมู่โดยใช้การเรียนรู้ขอเครื่องฉบับนี้มีความสำเร็จลุล่วงไปได้ด้วยดี เนื่องด้วยได้รับความเมตตากรุณาจาก รศ.ดร.วลัยลักษณ์ อัตธีรวงศ์ อาจารย์ที่ปรึกษาสหกิจ ที่ให้ความดูแลเอาใจใส่ ให้คำแนะนำ ให้ข้อเสนอแนะเสมอมา แนะนำเอกสารต่างๆ ช่วยตรวจทานความถูกต้อง ติดตามลูกศิษย์อย่างใกล้ชิดมาตลอดจนกระทั่งเสร็จสมบูรณ์ จึงขอขอบพระคุณด้วยความเคารพอย่างสูงไว้ ณ โอกาสนี้

ขอขอบพระคุณกรรมการสอบสหกิจศึกษา ผศ.ดร. ยุวดี กล่อมวิเศษ เป็นอย่างยิ่งที่ให้ความเอ็นดู ให้คำแนะนำ และเป็นห่วงจนงานวิจัยนี้สำเร็จลุล่วงไปด้วยดี

ขอบพระคุณผู้ประกอบการเป็นอย่างยิ่งที่ให้โอกาสเข้าไปทำสหกิจศึกษาในครั้งนี้ ทั้งได้ประสบการณ์ในการทำงาน และยังได้ข้อมูลมาทำสหกิจในครั้งนี้

สุดท้ายนี้ขอบพระคุณบิดามารดา ที่สนับสนุนผู้วิจัยทุกๆทาง ให้กำลังใจอยู่ตลอดไม่ห่างจนทำให้สหกิจศึกษาครั้งนี้สำเร็จลุล่วงไปได้ด้วยดี

ธนพัฒน์ สอนโส

สารบัญ

หน้า

บทคัดย่อภาษาไทย ..... ก
บทคัดย่อภาษาอังกฤษ ..... ข
กิตติกรรมประกาศ ..... ง
สารบัญ ..... จ
สารบัญตาราง ..... ซ
สารบัญรูป ..... ฌ
คำย่อ/สัญลักษณ์ ..... ญ
บทที่ 1 บทนำ 1
1.1 ความเป็นมาและความสำคัญ 1
1.2 วัตถุประสงค์ 2
1.3 ขอบเขต 2
1.4 ประโยชน์ที่คาดว่าจะได้รับ 2
บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง 3
2.1 การเรียนรู้ของเครื่อง(Machine Learning) 3
2.1.1 การเรียนรู้แบบมีผู้สอน (Supervised Learning) 3
2.1.2 การเรียนรู้แบบไม่มีผู้สอน (Unsupervised Learning) 4
2.1.3 การเรียนรู้แบบเสริมกำลัง (Reinforcement Learning) 4
2.2 การจัดหมวดหมู่ (Classification) 4
2.2.1 การจัดหมวดหมู่แบบไบนารี (Binary Classification) 4
2.2.2 การจัดหมวดหมู่แบบหลายคลาส (Multi-Class Classification) 5
2.2.3 การจัดหมวดหมู่แบบหลายเลเบล (Multi-Label Classification) 5
2.3 แบบจำลองป่าสุ่ม (Random Forest (RF)) 5
2.3.1 ค่าพารามิเตอร์ (parameter) 6
2.4 แบบจำลอง (Support Vector Machines (SVM)) 6
2.4.1 SVM algorithm 7
2.4.2 Kernel 7
2.4.3 Max-Margin and Support Vectors 9
2.4.3 ค่าพารามิเตอร์ (parameter) หลักที่สามารถปรับได้ใน SVM 9
2.5 K-Fold Cross Validation 10

สารบัญ (ต่อ)

2.6 เมทริกซ์ความสับสน (Confusion Matrix) 11
2.6.1 เมทริกซ์ความสับสนแบบจำแนก 2 ประเภท (Confusion Matrix for Binary Classification) 11
2.6.2 เมทริกซ์ความสับสนแบบจำแนกหลายประเภท (Confusion Matrix for Multi Classification) 13
2.7 ข้อมูลที่ไม่สมดุล (Imbalanced Datasets) 14
2.8 การแปลงข้อความเป็นตัวเลข (Vectorizer) 14
2.9 การกำหนดค่าเกณฑ์โดยใช้เส้นโค้งความแม่นยำและความครบถ้วน (Precision-Recall Curve) 14
2.10 งานวิจัยที่เกี่ยวข้อง 15
บทที่ 3 วิธีการดำเนินงานวิจัย 17
3.1 ขั้นตอนการดำเนินงาน 17
3.2 การจัดเตรียมข้อมูล 17
3.2.1 Label ข้อมูลจากข้อความที่ไม่มีหัวข้อ (Intent) 17
3.2.2 การทำความสะอาดข้อมูล (Cleaning data) 18
3.2.3 การจัดการกับข้อความ (Preprocess text) 18
3.2.3.1 การแทนคำ (Replace) 18
3.2.3.2 การทำความสะอาดข้อความ (Clean text) 19
3.2.4 การแบ่งคำ (Tokenization) 19
3.2.5 การแบ่งข้อมูล (Split data) 20
3.2.6 การเปลี่ยนหัวข้อให้เป็นตัวเลข (LabelEncoder) 21
3.2.7 เปลี่ยนข้อความเป็นตัวเลข (Vectorizer) 22
3.2.8 ใช้ K-Fold Cross Validation 23
3.3 ตัวแปรที่ใช้ในการวิเคราะห์ 23
3.3.1 ตัวแปรตาม (Dependent Variable) 23
3.3.2 ตัวแปรอิสระ (Independent Variable) 23
3.4 การสร้างแบบจำลอง 23
3.5 เครื่องมือที่ใช้ในการวิจัย 24
บทที่ 4 ผลการวิจัยและการอภิปรายผล 25
4.1 การทดสอบประสิทธิภาพของแบบจำลอง 25

สารบัญ (ต่อ)

4.1.1 ประสิทธิภาพของแบบจำลองป่าสุ่ม (Random Forest:RF)..... 25
4.1.2 ประสิทธิภาพแบบจำลอง Support Vector Machines (SVM)..... 27
4.1.3 ประสิทธิภาพแบบจำลอง Support Vector Machines (SVM) แบบ Kernel..... 30
4.1.4 เปรียบเทียบประสิทธิภาพของทุกโมเดล..... 33
4.2 การกำหนดเกณฑ์ (Find Threshold)..... 34
4.3 การใช้งานจริง..... 35
4.3.1 การแก้ปัญหา..... 36
4.4 อภิปรายผล..... 38
บทที่ 4 ผลการวิจัยและการอภิปรายผล..... 39
5.1 สรุปผลการวิจัย..... 39
5.2 ข้อเสนอแนะ..... 39
เอกสารอ้างอิง..... 41
ภาคผนวก..... 43

สารบัญตาราง

ตารางที่ **หน้า**

2.1 ตัวอย่าง K fold cv 11
3.1 ตัวอย่างชื่อหัวข้อที่สร้างใหม่จากข้อมูล “Other” 18
3.2 ตารางตัวอย่างการแทนคำ 19
3.3 ตัวอย่างการทำความสะอาดข้อความ 19
3.4 ตัวอย่างการแบ่งคำ (Tokenization) 19
3.5 ตัวอย่างจำนวนข้อมูลทั้งหมด 20
3.6 ข้อมูลสำหรับฝึกแบบจำลอง 20
3.6 ข้อมูลสำหรับฝึกแบบจำลอง(ต่อ) 21
3.7 ข้อมูลสำหรับทดสอบแบบจำลอง 21
3.8 ตัวอย่างการเปลี่ยนหัวข้อให้เป็นตัวเลข (LabelEncoder) 22
3.9 การอย่างการใช้TF-IDF ของข้อความ “อยาก ขอ สอบถาม ว่า หนังสือ แนว สืบสวนสอบสวน มี ไหม” 10 ตัว 22
3.10 ตัวแปรตามที่ใช้ในการวิเคราะห์ 23
3.11 ตัวแปรอิสระที่ใช้ในการวิเคราะห์ 23
3.12 ไลบรารีที่ใช้ในการสร้างแบบจำลองและวิเคราะห์ 24
4.1 ประสิทธิภาพในการทำนายของแบบจำลองป่าสุ่ม (Random Forest:RF) 27
4.2 ประสิทธิภาพในการทำนายของแบบจำลอง Support Vector Machine (SVM) 30
4.3 ประสิทธิภาพในการทำนายของแบบจำลอง Support Vector Machine (SVM) แบบ Kernel 33
4.4 เปรียบเทียบประสิทธิภาพแบบจำลอง 33
4.5 ค่า Probability ที่ทำให้ค่า F1 ที่คิดแยกแบบไบนารีในแต่ละคลาสสูงที่สุด 35
4.6 การเพิ่มเกณฑ์ (Threshold) 36
4.7 เปรียบเทียบค่าF-1 ของแบบจำลองป่าสุ่มและค่าF-1 ของแบบจำลองแบบป่าสุ่มที่เพิ่มเกณฑ์แล้วในการทดสอบกับข้อมูลทดสอบที่ไม่มีหัวข้อ “อื่นๆ” 37
5.1 ประสิทธิภาพของแต่ละแบบจำลอง 39

สารบัญรูป

รูปที่ **หน้า**

2.1 การทำงานของ Random Forest 5
2.2 ตัวอย่างการแบ่งกลุ่มโดยใช้ Hyperplane 6
2.3 ตัวอย่างการแบ่งข้อมูล 9
2.4 การปรับพารามิเตอร์ C 10
2.5 ตัวอย่างการแบ่งข้อมูลเป็น 5 folds เท่าๆกัน 10
2.6 เมทริกซ์ความสับสนแบบจำแนก2ประเภท 11
2.7 เมทริกซ์ความสับสนแบบจำแนกหลายประเภท 13
2.8 ตัวอย่าง Precision-Recall curve 15
3.1 ขั้นตอนการดำเนินงาน 17
4.1 เมทริกซ์ความสับสนแบบหลายคลาสของแบบจำลองป่าสุ่ม(Random Forest:RF) 25
4.2 เมทริกซ์ความสับสนแบบหลายคลาสของแบบจำลองป่าสุ่ม(Random Forest:RF) แบบเป็นเปอร์เซ็นต์ 26
4.3 เมทริกซ์ความสับสนแบบหลายคลาสของแบบจำลอง Support Vector Machine(SVM) 28
4.4 เมทริกซ์ความสับสนแบบหลายคลาสของแบบจำลอง Support Vector Machine (SVM) แบบเปอร์เซ็นต์ 29
4.5 เมทริกซ์ความสับสนแบบหลายคลาสของแบบจำลอง Support Vector Machine (SVM) แบบ Kernel 31
4.6 เมทริกซ์ความสับสนแบบหลายคลาสของแบบจำลอง Support Vector Machine (SVM) แบบ Kernel รูปแบบเปอร์เซ็นต์ 32
4.7 เส้นโค้งความแม่นยำและความโค้ง 34
4.8 การใช้งานจริงหัวข้อ “โทรศัพท์ 35
4.9 การใช้งานจริงหัวข้อ “เครื่องซักผ้า” 36
4.10 การใช้งานจริงหัวข้อ “โทรศัพท์” หลังจากเพิ่มเกณฑ์ 38

คำย่อ/สัญลักษณ์

คำย่อ/สัญลักษณ์	คำอธิบาย
RF	แบบจำลองป่าสุ่ม Random Forest
SVM	แบบจำลองซัพพอร์ตเวกเตอร์แมชชีน
F-1	ค่าประสิทธิภาพโดยรวมของแบบจำลอง

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันมีการนำเทคโนโลยีต่างๆ มาใช้ในการประกอบอาชีพเป็นจำนวนมาก โดยเฉพาะด้านธุรกิจและอุตสาหกรรม เพื่อให้เกิดความสะดวกสบายมากขึ้น แชทบอท (Chatbot) หรือระบบช่วยตอบคำถามอัตโนมัติให้กับผู้สนทนา หรือลูกค้า จึงเป็นอีกหนึ่งเทคโนโลยีที่ได้รับความนิยมเป็นอย่างมาก ซึ่งถือเป็นเครื่องมือที่สำคัญสำหรับธุรกิจในยุคดิจิทัล เนื่องจากแชทบอทจะช่วยคัดกรองลูกค้า ให้ได้ข้อมูลที่ต้องการไปก่อนที่จะถึงมือของเจ้าหน้าที่ เพื่อแบ่งเบาการทำงานของเจ้าหน้าที่ หรือคนให้น้อยลง แชทบอททำให้ช่วยประหยัดค่าใช้จ่าย ช่วยตอบคำถาม และช่วยให้ผู้ที่มาสอบถามได้รับคำตอบที่ต้องการ และยังช่วยให้ได้ประสิทธิภาพการทำงานมากขึ้นโดยใช้คน หรือ ทรัพยากรน้อยลง

แชทบอทคือ แอปพลิเคชันซอฟต์แวร์ที่ถูกออกแบบให้สามารถพูดคุย หรือสนทนาได้เหมือนกับมนุษย์ผ่านทางข้อความ หรือการพูดคุยโดยมีเทคโนโลยี AI เป็นผู้อยู่เบื้องหลัง รวมถึงเจ้าหน้าที่เสมือน (Virtual agent) เจ้าหน้าที่เสมือนในการโต้ตอบกับลูกค้า (Interactive agents) และผู้ช่วยดิจิทัล (Digital assistants) หรือ AI เชิงสนทนา (Conversational AI) ณ ที่นี้จะใช้เครื่องมือแชทบอทในการสนทนากับลูกค้าผ่านทางระบบข้อความเสียง หรือการโทรเพื่อที่จะลดการใช้เจ้าหน้าที่ตอบคำถาม แชทบอทเรียนรู้จากพฤติกรรมของผู้ใช้งาน และเมื่อเวลาผ่านไปทำให้แชทบอททอบคำถามได้อย่างแม่นยำ และมีประสิทธิภาพในการทำงานสูง

ในปัจจุบันได้มีการนำ AI และ Machine learning เข้ามาใช้เพื่อสร้างประสบการณ์ที่เหมือนกับมีเจ้าหน้าที่ในการตอบคำถาม และพูดคุยกับลูกค้าโดยใช้โมเดลรูปแบบการแบ่งออกเป็นหมวดหมู่ (Classification) ในการช่วยจำแนกคำตอบที่บอทจะตอบได้โดยการใช้ Machine learning นี้ทำให้บอทมีความฉลาดเหมือนมีเจ้าหน้าที่ช่วยตอบคำถามอยู่หลังสายโทรศัพท์

การเรียนรู้ของเครื่อง (Machine Learning) คือหัวใจหลักพื้นฐานของเทคโนโลยี AI Chatbot กระบวนการเรียนรู้ทั้งสองแบบนี้ช่วยให้ AI สามารถเรียนรู้องค์ประกอบ รูปแบบ และลักษณ์ของสิ่งต่าง ๆ ได้ และทำให้ AI สามารถแยกแยะของสองสิ่งออกจากกัน เช่น นกกับเครื่องบิน หรือ สีส้มกับสีแดง การเรียนรู้ของเครื่อง (Machine Learning) คือการเรียนรู้อย่างต่อเนื่องของหุ่นยนต์ และปัญญาประดิษฐ์ ผ่านการป้อนข้อมูลจากมนุษย์ ระบบคอมพิวเตอร์จะสามารถเรียนรู้ความแตกต่างของข้อมูล คาดเดา และสร้างการตัดสินใจด้วยตัวเองได้

หัวข้อที่เข้ามาในแชทบอท (Intent) สำหรับ คำกริยาที่เข้ามาในแชทบอท ในภาษาอังกฤษจะเรียกว่า Intent หลายคนเคยได้ยินประโยคที่บอกว่า Pay attention หรือตั้งใจหน่อย ระวังหน่อยในบริบทของแชทบอท Intent จะแปลได้ว่า ความตั้งใจ หรือจุดประสงค์ของประโยคที่เข้ามา อาทิ อยาก

ซื้อ อยากขาย อยากยกเลิก อยากสมัคร ในที่นี้ต้องการที่จะเพิ่ม หัวข้อ (Intent) ผู้วิจัยจึงใช้การเรียนรู้ของเครื่อง (machine learning) มาช่วยในการเพิ่ม

ปัญหาของบริษัท A คือแชทบอทไม่สามารถตอบปัญหาของผู้ใช้ได้ครบถ้วน ทำให้มีข้อความหลงเหลือแบบไม่มีหัวข้อเป็นจำนวนมาก ผู้วิจัยจึงคิดสร้างแบบจำลอง เพื่อช่วยให้แชทบอทสามารถมีหัวข้อตอบคำถามได้มากขึ้น และอาจจะทำให้เจ้าหน้าที่มีอัตราการทำงานที่ต่ำลง

1.2 วัตถุประสงค์ของงานวิจัย

1) เพื่อเพิ่มหัวข้อที่เข้ามาในแชทบอท (Intent) โดยใช้การเรียนรู้ของเครื่อง (Machine Learning) แบบการจัดหมวดหมู่ (Classification)

2) เพื่อเลือกตัวแบบที่เหมาะสมที่สุด สำหรับการเลือกหัวข้อให้กับข้อความโดยใช้ Confusion matrix หรือ ตาราง Cross tabulation ในการคัดเลือก

1.3 ขอบเขตของงานวิจัย

1.3.1 ขอบเขตด้านข้อมูล

ข้อมูลที่นำเข้ามีลักษณะเป็นข้อความที่บอทไม่สามารถจัดหมวดหมู่ให้ได้ (other) มาหาหมวดหมู่ เป็นข้อมูลของปี พ.ศ.2566

1.3.2 ขอบเขตด้านเวลา

ผู้วิจัยได้ดำเนินการศึกษาวิจัยโดยมีระยะเวลาตั้งแต่วันที่ 1 ธันวาคม พ.ศ.2566 ถึงวันที่ 31 มีนาคม พ.ศ.2567 รวมระยะเวลาทั้งสิ้น 4 เดือน

1.3.3 ขอบเขตด้านเครื่องมือ

เครื่องมือที่ใช้สำหรับวิเคราะห์ข้อมูล

- โปรแกรมภาษาไพธอน (Python) Colab Notebook
- Google sheet Microsoft excel

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้หัวข้อคำตอบใหม่ๆ ให้บอทข้อความเสียงสามารถพูดคุยโต้ตอบกับลูกค้า ได้หลากหลายมากขึ้น เพื่อให้บริษัท A มีประสิทธิภาพในการตอบคำถามลูกค้ามากขึ้น

2) เพื่อเป็นแนวทางในการสร้างหัวข้อใหม่ๆให้บอทข้อความเสียงต่อไป

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

งานวิจัยนี้ผู้วิจัยได้รวบรวมแนวคิด ทฤษฎี และหลักการต่างๆจากเอกสาร และงานวิจัยที่เกี่ยวข้องดังนี้

2.1 การเรียนรู้ของเครื่อง
2.2 การจัดหมวดหมู่
2.3 แบบจำลองป่าสุ่ม
2.4 แบบจำลอง Support Vector Machines (SVM)
2.5 K-Fold Cross Validation
2.6 เมทริกซ์ความสับสน
2.7 ข้อมูลที่ไม่สมดุล
2.8 การแปลงข้อความเป็นตัวเลข
2.9 การกำหนดค่าเกณฑ์โดยใช้เส้นโค้งความแม่นยำและความครบถ้วน
2.10 งานวิจัยที่เกี่ยวข้อง

2.1 การเรียนรู้ของเครื่อง (Machine Learning)

คือถูกใช้งานเสมือนเป็นสมองของ AI (Artificial Intelligence) AI นั้นใช้ Machine Learning ในการสร้างความฉลาด ซึ่งความฉลาดนั้นจะมากจากข้อมูล (Data) โดยที่ผู้วิจัยใส่ข้อมูล (Data) และคำตอบเข้าไป เพื่อที่จะให้ได้ output ที่ต้องการออกมาแบบอัตโนมัติสามารถค้นหา แยกแยะ แบ่งกลุ่ม คาดคะเน และคำนวณความน่าจะเป็น และแก้ไขปัญหาได้อย่างเหมาะสม โดยที่ไม่ต้องมีมนุษย์มากำกับ และมนุษย์ไม่ต้องเขียนโปรแกรมใหม่ เพราะคอมพิวเตอร์สามารถทำงานอัตโนมัติผ่านอัลกอริทึม (Algorithm) ที่ Data Scientist เป็นผู้ออกแบบซึ่งการเรียนรู้ของเครื่อง (Machine Learning) (Vithan, 2561; Matana, 2567) ยังแบ่งได้อีก 3 รูปแบบดังนี้

2.1.1 การเรียนรู้แบบมีผู้สอน (Supervised Learning)

การเรียนรู้แบบมีผู้สอน (Supervised Learning) มีมาแล้วตั้งแต่ปี 1959 ถูกเสนอโดย Arthur Samuel เป็นนักวิทยาศาสตร์คอมพิวเตอร์ชาวอเมริกัน แต่ด้วยเทคโนโลยีหรือระบบประมวลผลในตอนนั้นยังล้าสมัยอยู่ ทำให้ยังไม่เป็นที่นิยม ผิดกับในปัจจุบัน การเรียนรู้แบบมีผู้สอน (Supervised Learning) เป็นกลุ่มของ algorithm ที่เน้นสอน computer โดยการศึกษาจากข้อมูลตัวอย่าง เพื่อทำให้คอมพิวเตอร์สามารถหาคำตอบของปัญหาได้ด้วยตัวเอง หลังจากเรียนรู้จากชุดข้อมูลตัวอย่างที่ได้ป้อนให้ไปแล้วระยะหนึ่ง (Natdanai, 2562)

2.1.2 การเรียนรู้แบบไม่มีผู้สอน (Unsupervised Learning)

เป็นการเรียนรู้ที่ให้เครื่องจักรนั้นสามารถเรียนรู้ได้ด้วยตนเอง โดยไม่ต้องมีค่าเป้าหมายของแต่ละข้อมูล ซึ่งวิธีการคือมนุษย์จะเป็นผู้ใส่ข้อมูลต่าง ๆ และกำหนดสิ่งที่ต้องการจากข้อมูลเหล่านั้น โดยให้เครื่องจักรวิเคราะห์จากการจำแนก และสร้างแบบแผนจากข้อมูลที่ได้รับมา โดยตัวอย่างที่เห็นได้ชัดของ Machine Learning ในกลุ่ม Unsupervised Learning ที่ถูกนำมาประยุกต์ใช้งานในเชิงธุรกิจ คือ ระบบแนะนำผลิตภัณฑ์ ยกตัวอย่างเช่นการแนะนำคลิปวิดีโอใน YouTube ที่ทำการแบ่งหมวดหมู่ของคลิปวีดีโอต่าง ๆ

2.1.3 การเรียนรู้แบบเสริมกำลัง (Reinforcement Learning)

เป็นการเรียนรู้สิ่งต่าง ๆ จากการลองผิดลองถูก ภายใต้แนวคิดที่ว่าจะเลือกกระทำสิ่งที่ทำให้ได้ผลลัพธ์มากที่สุด โดยทำการเรียนรู้จากการลองผิดลองถูก ในสถานการณ์ในอดีตหรือระบบจำลองและพยายามที่จะพัฒนาระบบการตัดสินใจของตัวเองให้ดีขึ้นเรื่อย ๆ โดยที่อาจจะพัฒนาด้วยการพยายามสร้างแบบจำลองสถานการณ์ต่าง ๆ โดยตัวอย่างที่เห็นได้ชัดของ Machine Learning ในกลุ่ม Reinforcement Learning ที่ถูกนำมาประยุกต์ใช้งานในเชิงธุรกิจ คือ AlphaGo ที่สามารถเล่นเกมโกะให้ชนะผู้เล่นระดับโลก ระบบการจัดการ Portfolio ให้ตัดสินใจเลือกอัตราส่วนของสินทรัพย์

2.2 การจัดหมวดหมู่ (Classification)

การจัดหมวดหมู่ (Classification) เป็นแบบจำลองประเภทการเรียนรู้แบบมีผู้สอน (Supervised Learning) การจัดหมวดหมู่ (Classification) จำเป็นจะต้องมี Target ไว้สำหรับให้ตัว Model เรียนรู้จาก Input Data เพื่อหาคำตอบออกมาตาม Target ที่ได้วางเอาไว้ ซึ่งผลลัพธ์จากการวิเคราะห์ข้อมูลด้วย โมเดลการจัดหมวดหมู่ (Classification Model) จะเป็นในรูปแบบของการจำแนกข้อมูลเพื่อให้ได้คำตอบที่เป็นตัวเลือก หรือกลุ่มข้อมูล ตัวอย่างของ Target ของโมเดล เช่น Yes กับ No เป็น กับ ไม่เป็น หรือเป็นกลุ่มคำตอบว่าเป็นกลุ่ม A B หรือ C ซึ่งโมเดลการจัดหมวดหมู่ Classification Model สามารถวัดความแม่นยำของโมเดล (Accuracy) ด้วยการใช้ เมทริกซ์ความสับสน (Confusion Matrix) ถ้าเปรียบเทียบกับ Regression Model ซึ่งเป็นหนึ่งในโมเดลประเภท Supervised Learning Model เช่นเดียวกับ Classification Model แต่มีความแตกต่างกันตรงที่ Regression Model มี Target ของโมเดลเป็น Value ซึ่งออกมาเป็นค่าตัวเลขที่จะเกิดขึ้นในอนาคตจาก Input Data ไม่ใช่เป็นคำตอบที่เป็นตัวเลือกเหมือนของ Classification(สถาบันนวัตกรรมและธรรมมาภิบาลข้อมูล, 2565) โมเดลการจัดหมวดหมู่ (Classification Model) แบ่งออกได้เป็น 3 ประเภทหลัก ได้แก่

2.2.1 การจัดหมวดหมู่แบบไบนารี (Binary Classification)

เป็นรูปแบบการจำแนกที่มีกระบวนการวิเคราะห์ Input Data เพื่อให้ได้ผลลัพธ์ออกมาเป็น Target แค่เพียง 2 ประเภทหรือโมเดลมีเพียง 2 คำตอบเช่น Input รูปภาพเข้าสู่คอมพิวเตอร์ว่ารูปนี้เป็นหมาหรือไม่ ซึ่งคำตอบของโมเดลมีเพียง 2 คำตอบ คือ ใช่หรือไม่

2.2.2 การจัดหมวดหมู่แบบหลายคลาส (Multi-Class Classification)

ลักษณะคล้ายกับการจัดหมวดหมู่แบบไบนารี (Binary Classification) แต่ผลลัพธ์ที่ได้ออกมาเป็น Target มากกว่า 2 คำตอบขึ้นไปตัวอย่างเช่น input data เป็นรูปภาพแล้วให้จำแนกว่ารูปภาพที่เห็นเป็นภาพสัตว์ สิ่งของ หรือไม่ใช่ทั้งคู่ ซึ่งคำตอบของโมเดลมี 3 คำตอบ

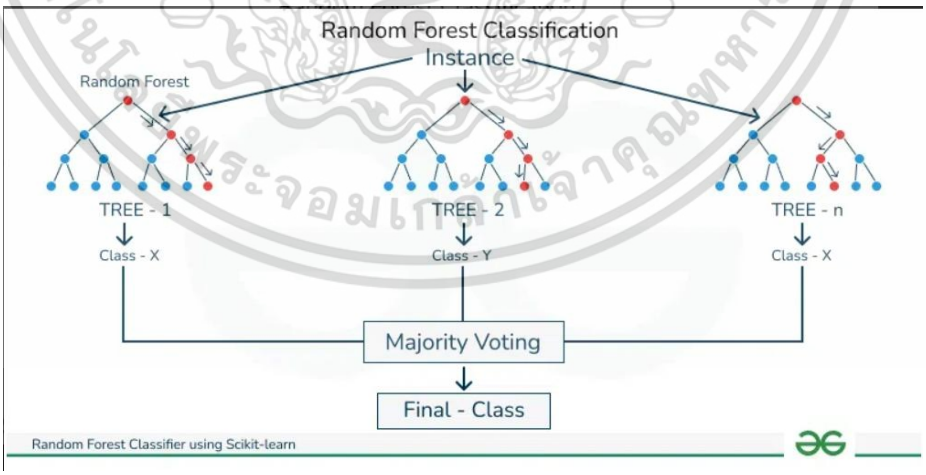
2.2.3 การจัดหมวดหมู่แบบหลายเลเบล (Multi-Label Classification)

Multi-Label ชุดข้อมูลชุดหนึ่งอาจจะมีข้อมูลที่มีคุณสมบัติเหมือนกันแต่สามารถให้ผลลัพธ์ที่มีความแตกต่างกันซึ่งการเรียนรู้ของคอมพิวเตอร์เพื่อจำแนก multi-label จะมีความยากและซับซ้อนกว่าแบบ Multi-Class

ผู้วิจัยได้ใช้การจัดหมวดหมู่แบบหลายคลาส (Multi-Class Classification) และใช้แบบจำลองแบบป่าสุ่ม แบบจำลองซัพพอร์ตเวกเตอร์แมชชีน และแบบจำลองซัพพอร์ตเวกเตอร์แมชชีน แบบ Kernel

2.3 แบบจำลองป่าสุ่ม (Random Forest (RF))

แบบจำลองป่าสุ่มRandom Forest (RF) เป็นอัลกอริทึมที่ใช้สำหรับการจัดหมวดหมู่ (Classification) การถดถอย (Regresstion) และอื่นๆ (Amandp13,2024) Random Forest จะใช้พื้นฐานมาจาก ต้นไม้ ตัดสินใจ (Decision Tree) หลายๆต้น โดยสร้างจากการสุ่มข้อมูลตัวอย่างแบบเลือกและใส่กลับ (Random Sampling With Replacement) เพื่อนำมาสร้างเป็นแบบจำลองต้นไม้แบบไม่ซ้ำกัน และแต่ละต้นจะมีการทำนายผลซึ่งผลของการทำนายจะเลือกจากผลที่มีการโหวตมากที่สุด (ธนัท, 2561) โดยเทคนิคนี้เรียกว่า Bootstrap Aggregation (Bagging)



รูปที่ 2.1 การทำงานของ Random Forest

ที่มา Amandp13 (2567)

2.3.1 ค่าพารามิเตอร์ (Parameter)

ค่าพารามิเตอร์หลักที่สามารถปรับได้มีดังนี้

n_estimator:จำนวนต้นไม้ในการจำลอง ซึ่งถ้ามีเยอะจะยิ่งทำให้การทำนายแม่นแต่แลกมากับการที่จะใช้เวลาในการคำนวนนานขึ้น

max_depth:ความลึกของต้นไม้

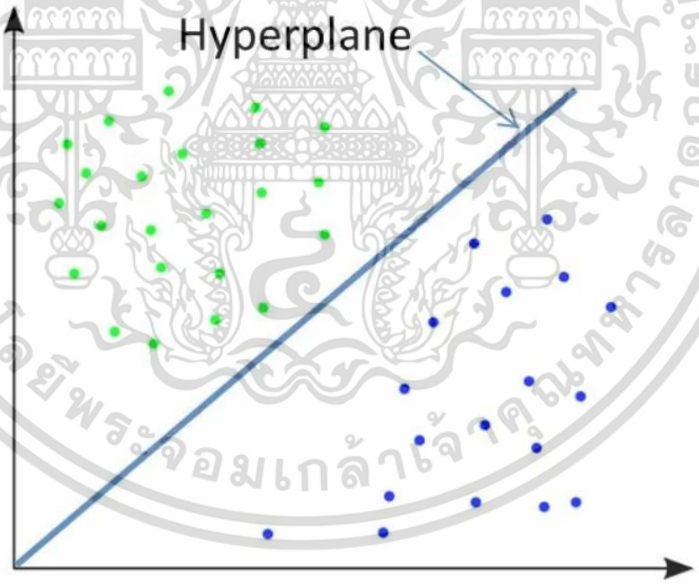
min_samples_leaf:จำนวนข้อมูลขั้นต่ำที่สุ่มมาเป็น leaf node

min_samples_split:จำนวนข้อมูลขั้นต่ำในการแยก node

max_features: จ้านวนของฟีเจอร์ที่ถูกสุ่มมาสร้างต้นไม้ตัดสินใจ (Decision Tree) แต่ละต้น

2.4 แบบจำลอง Support Vector Machines (SVM)

เป็นหนึ่งในโมเดล Machine Learning ที่ใช้ในการจำแนกข้อมูล หรือแบ่งกลุ่มข้อมูลโดยจะสร้างเส้นตรงที่ใช้แบ่งกลุ่มข้อมูล (Hyperplane) และหาเส้นที่ดีที่สุด (PradyaSin, 2562) SVM ใช้ได้หลากหลายรูปแบบเช่น การจัดหมวดหมู่ข้อความ การจัดหมวดหมู่รูปภาพ การตรวจหาสแปม เป็นต้น



รูปที่ 2.2 ตัวอย่างการแบ่งกลุ่มโดยใช้ Hyperplane
ที่มา PradyaSin (2562)

2.4.1 SVM algorithm

SVM ใช้ Hypothesis function แบบเส้นตรง เหมือนกับ Linear regression ดังสมการที่ (2.1)

$$\begin{aligned} h_\theta(x) &= w_1x_1 + w_2x_2 + \cdots + w_nx_n + b \\ &= w^Tx + b \end{aligned} \tag{2.1}$$

โดยถ้าผลลัพธ์เป็นบวก จะทำนาย Class $\hat{y}$ ว่าเป็น 1 ส่วนถ้าเป็นลบ ทำนายว่าเป็น 0 สามารถเขียนวิธีการตัดสินใจตามเงื่อนไขดังกล่าวดังสมการที่ (2.2)

$$\hat{y} = \begin{cases} 0 \text{ if } w^Tx + b < 0, \\ 1 \text{ if } w^Tx + b \geq 0 \end{cases} \tag{2.2}$$

เมื่อนิยามเส้นแบ่งการตัดสินใจแล้ว (เส้นทึบ) จะกำหนดเส้นประทั้งสองด้านของเส้นทึบ โดยเส้นประแต่ละด้านคือตำแหน่งที่ h_θ (x) เท่ากับ -1 และ 1 โดยที่เป้าหมายคือต้องการลด $\|w\|$ เพื่อที่จะให้ได้เส้นขอบเขตให้กว้างที่สุดที่เป็นไปได้ อย่างไรก็ตาม ในเวลาเดียวกันไม่ต้องการให้ขอบเขตเส้นแบ่งนั้นกว้างเกินไปจนกระทั่งครอบคลุมจุดข้อมูล ดังนั้นจึงต้องการให้ฟังก์ชันการตัดสินใจนั้นมีค่ามากกว่า 1 ในด้านที่ผลพยากรณ์เป็น 1 ("ใช่") และน้อยกว่า -1 ในด้านที่ผลพยากรณ์เป็น 0 ("ไม่ใช่") โดยที่จะได้เป้าหมายการ Optimise ของ SVM algorithm ดังสมการที่ (2.3)

$$\begin{aligned} &\text{minimise}_{w,b} \quad \frac{1}{2}w^Tw \\ &\text{subject to} \quad t^{(i)}(w^Tx^{(i)} + b) \geq 1 \end{aligned} \tag{2.3}$$

เป้าหมายนี้ใช้สำหรับ Hard margin SVM แต่ถ้าเป็น Soft margin ที่ต้องการอนุญาตให้พื้นที่เส้นขอบเขตการตัดสินใจนั้นกินบริเวณที่มีจุดข้อมูลอยู่ด้วยได้ ก็ต้องเพิ่มตัวแปรที่เรียกว่า Slack variable โดยระดับของ Slack variable จะถูกกำหนดโดย Hyperparameter C ดังนั้นเป้าหมายการ Optimise สำหรับ Soft margin SVM คือสมการที่ (2.4)

$$\begin{aligned} &\text{minimise}_{w,b} \quad \frac{1}{2}w^Tw + C\sum_{i=1}^{m}\zeta^{(i)} \\ &\text{subject to} \quad t^{(i)}(w^Tx^{(i)} + b) \geq 1 - \zeta^{(i)} \end{aligned} \tag{2.4}$$

2.4.2 Kernel

SVM มีข้อจำกัดคือสามารถสร้างเส้นแบ่งขอบเขตการตัดสินใจแบบเส้นตรงเท่านั้น ซึ่งอาจทำงานได้ไม่ดีถ้าความสัมพันธ์ของข้อมูลนั้นมีความซับซ้อนจนแบ่งด้วยเส้นตรงไม่ได้ วิธีการแก้ปัญหานี้เรียกว่า Kernel

Kernel คือ "ทริค" ทางคณิตศาสตร์ที่ทำให้ Algorithm สามารถ Optimise ค่าตัวแปรแบบ Polynomial ได้ โดยไม่ต้องไปเปลี่ยนรูปแบบและความสัมพันธ์ของ Feature ตั้งต้นโดยรูปแบบของเป้าหมาย Optimise แบบใหม่ที่จะเป็นดังสมการที่ (2.5)

$$\begin{aligned} &\textbf{minimise}_{\alpha} && \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha^{(i)}\alpha^{(j)}t^{(i)}t^{(j)}x^{(i)T}x^{(j)} - \sum_{i=1}^{m}\alpha^{(i)} \\ &\textbf{subject to} && \alpha^{(i)} \geq 0 \end{aligned} \tag{2.5}$$

เมื่อหา Vector $\hat{\alpha}$ ที่ทำให้สมการนี้มีค่าน้อยที่สุดได้แล้ว ก็จะสามารถคำนวนหา Vector $\hat{w}$ และ Intercept $\hat{b}$ ที่ทำให้สมการ Primal problem (2.3) หรือ (2.4) นั้นมีค่าน้อยที่สุด โดยทำดังสมการที่ (2.6 และ 2.7)

$$\hat{w} = \sum_{i=1}^{m}\alpha^{(i)}t^{(i)}x^{(i)} \tag{2.6}$$

$$\hat{b} = \frac{1}{n_s}\sum_{i=1}^{m}(t^{(i)} - \hat{w}^T x^{(i)}) \tag{2.7}$$

ถ้าต้องการทำเหมือนกับว่า Hypothesis function ที่ประกอบด้วยตัวแปร x1 และ x2 นั้นมีรูปร่างฟังก์ชันที่ซับซ้อนขึ้น โดยต้องการให้เป็น Second-degree polynomial จะสามารถเขียน Mapping function (ϕ อ่านว่า Phi) ของสมการนี้ได้ดังสมการที่ (2.8)

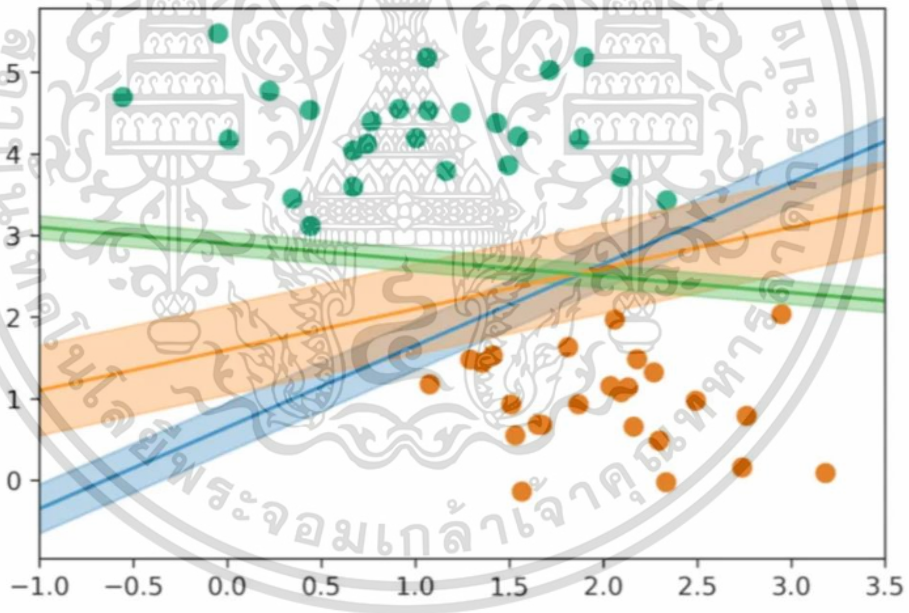
$$\phi(x) = \phi\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1^2 \\ \sqrt{2}\, x_1 x_2 \\ x_2^2 \end{pmatrix} \tag{2.8}$$

ถ้าคำนวน Dot product ของ Vector ขนาด 3 มิติ 2 Vector (แบบเดียวกับที่อยู่ในสมการ (2.5)) โดยสมมุติว่าชื่อ a และ b จะได้ดังสมการที่ (2.9) (ชิตพงษ์, 2563)

$$
\begin{aligned}
\phi(a)^T\phi(b) &= \begin{pmatrix} a_1^2 \\ \sqrt{2}\,a_1a_2 \\ a_2^2 \end{pmatrix}^T \begin{pmatrix} b_1^2 \\ \sqrt{2}\,b_1b_2 \\ b_2^2 \end{pmatrix} \\
&= a_1^2b_1^2 + 2a_1b_1a_2b_2 + a_2^2b_2^2 \\
&= (a_1b_1 + a_2b_2)^2 \\
&= \left[\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}^T \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}\right]^2 \\
&= (a^Tb)^2
\end{aligned}
\tag{2.9}
$$

2.4.3 Max-Margin and Support Vectors

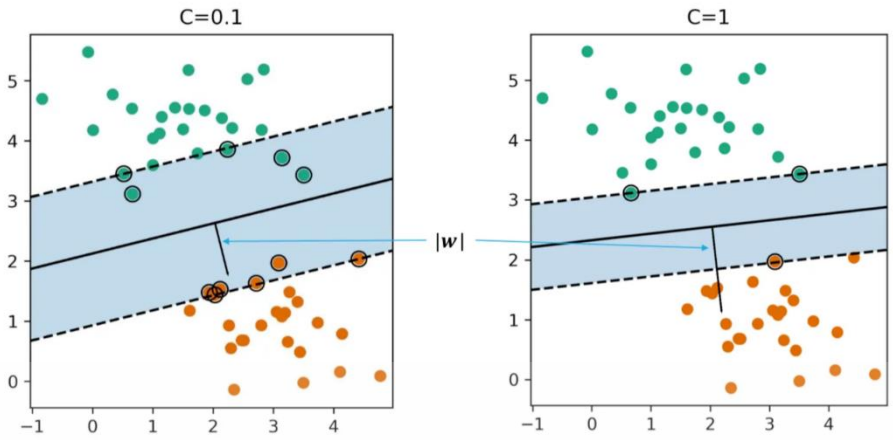
การแบ่งข้อมูลสามารถแบ่งได้หลายเส้น แต่วิธีเลือกว่าจะเอาเส้นไหนคือ จะเลือกเส้นที่มี Margin มากที่สุด คือ เส้นที่มีระยะแบ่งกว้างที่สุด เช่น เส้นสีส้มมีระยะมากที่สุด



รูปที่ 2.3 ตัวอย่างการแบ่งข้อมูล
ที่มา PradyaSin (2562)

2.4.4 ค่าพารามิเตอร์ (Parameter) หลักที่สามารถปรับได้ใน SVM

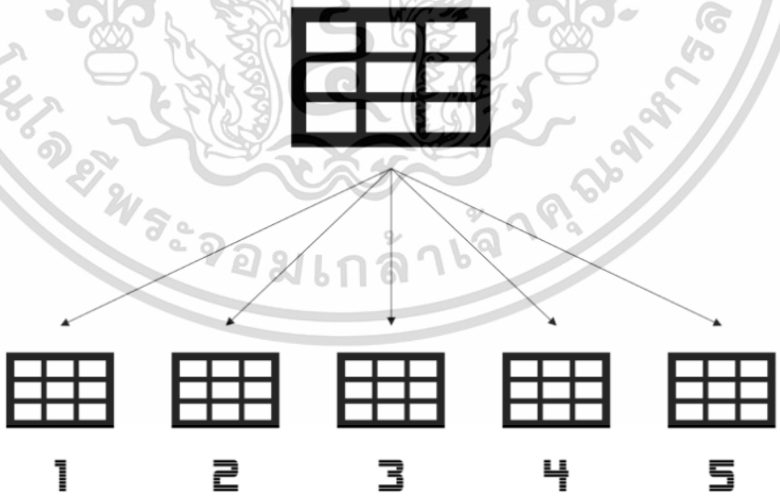
SVM_C: การปรับ parameter C จะทำให้ขนาดของเส้นแบ่งเปลี่ยนแปลงได้ โดยที่ C โดยที่ C มากจะทำให้พื้นที่แคบลง C น้อยจะทำให้พื้นที่กว้างขึ้น



รูปที่ 2.4 การปรับพารามิเตอร์ C
ที่มา PradyaSin (2562)

2.5 K-Fold Cross Validation

เป็นหนึ่งในเทคนิคการทำ Resampling k-fold cv คือการแบ่งข้อมูลเป็น k ส่วนเท่าๆ กันเพื่อสร้างและทดสอบโมเดล (Train + Validate) คำนวณค่าเฉลี่ย Accuracy หรือ Error (i.e. Model Performance) ก่อนที่จะนำโมเดลไปใช้ทำนายข้อมูล Test set รูปที่ 2.5 แสดงการแบ่งข้อมูลเป็น 5 Folds เท่าๆกัน โดยการแบ่งข้อมูลต้องเป็นไปอย่างสุ่ม (Kasidis, 2562)



รูปที่ 2.5 ตัวอย่างการแบ่งข้อมูลเป็น 5 Folds เท่าๆกัน
ที่มา Kasidis (2561)

พอแบ่งข้อมูลเสร็จแล้ว (k=5) จะสร้างและทดสอบโมเดลจนกว่าข้อมูลทุก fold จะถูกนำมาใช้ ถ้า k=5 ต้องเทรนโมเดลทั้งหมด 5 รอบด้วย (Train folds) และทดสอบโมเดลทั้งหมด 5 รอบด้วย (Validation fold)

ในแต่ละ Iteration (รอบ) ต้องบันทึกค่า Validation error ไว้ด้วยเพื่อนำไปสรุปผลหลังจบกระบวนการ Cross validation ทั้งหมด ตัวอย่างตารางที่2.1สามารถคำนวณค่าเฉลี่ย Validation error ได้เท่ากับ (0.12 + .25 + .46 + .25 + .11)/ 5 = 0.238

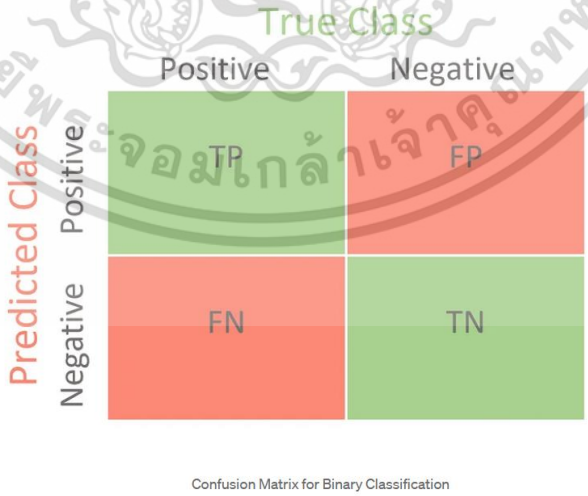
ตารางที่ 2.1 ตัวอย่าง K fold cv

Iteration	Train Folds	Validation Fold	Validation Error
1	{1, 2, 3, 4}	5	0.12
2	{1, 2, 3, 5}	4	0.25
3	{1, 2, 4, 5}	3	0.46
4	{1, 3, 4, 5}	2	0.25
5	{2, 3, 4, 5}	1	0.11

2.6 เมทริกซ์ความสับสน (Confusion Matrix)

คือตารางที่ใช้วัดความสามารถการเรียนรู้ของเครื่อง (Machine Learning) ในโมเดลการจัดหมวดหมู่ (Classification) (Mohajon,2020)

2.6.1 เมทริกซ์ความสับสนแบบจำแนก 2 ประเภท (Confusion Matrix for Binary Classification)



รูปที่ 2.6 เมทริกซ์ความสับสนแบบจำแนก 2 ประเภท
ที่มา Joydwip (2563)

เมทริกซ์ความสับสนแบบจำแนก2ประเภท (Confusion Matrix for Binary Classification) มีคลาสเพียง2 คลาสแบ่งเป็นบวก (Positive) และลบ (Negative)

1. True Positive (TP) คือ สิ่งที่แบบจำลองโมเดลทำนายว่า “จริง” และมีค่าเป็น “จริง”
2. True Negative (TN) คือ สิ่งที่แบบจำลองโมเดลทำนายว่า “ไม่จริง” และมีค่าเป็น “ไม่จริง”
3. False Positive (FP) คือ สิ่งที่จำลองทำนายว่า “จริง” แต่มีค่าเป็น “ไม่จริง”
4. False Negative (FN) คือ สิ่งที่จำลองทำนายว่าา “ไม่จริง” แต่มีค่าเป็น “จริง”

โดยจะมีตัววัดค่าที่ใช้ในการทำงานหลักๆอยู่ 4 ค่า คือ

ค่าความถูกต้อง (Accuracy) เป็นการวัดค่าความถูกต้องโดยรวมของโมเดล โดยจะพิจารณาทุกคลาส ดังสมการที่ (2.10)

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}} \tag{2.10}$$

ค่าความแม่นยำ (Precision) เป็นการวัดความแม่นยำของโมเดลโดยเฉพาะ โดยจะพิจารณาแยกทีละคลาส ดังสมการที่ (2.11)

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{2.11}$$

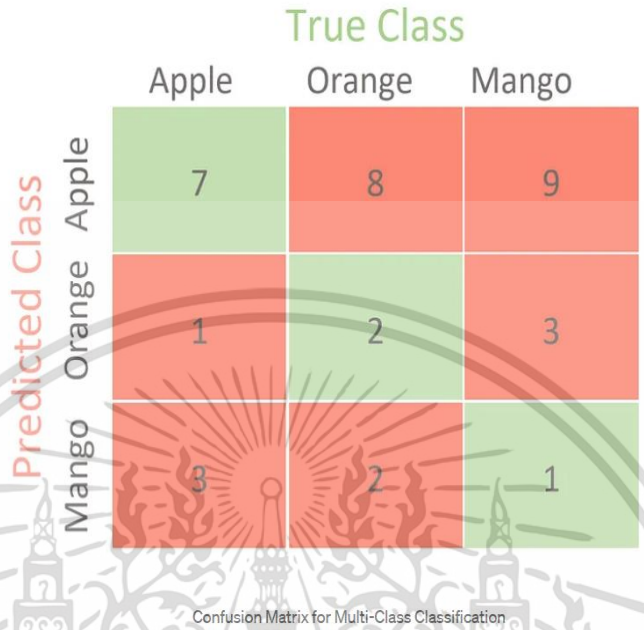
ค่าความครบถ้วน (Recall) เป็นการวัดความถูกต้องของโมเดลเมื่อเอาผลของโมเดลมาเทียบกับคำตอบที่มีมาให้โมเดล ดังสมการที่ (2.12)

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{2.12}$$

ค่าประสิทธิภาพโดยรวม (F1-score) เป็นการรวมกันระหว่าง ค่าความแม่นยำ (Precision) และ ค่าความครบถ้วน (Recall) ทำ ให้เป็นค่าเดียว ดังสมการที่ (2.13)

$$F1-score = 2 \times \frac{Precision \times Recall}{Precison+Recall} = \frac{2TP}{2TP+FP+FN} \tag{2.13}$$

2.6.2 เมทริกซ์ความสับสนแบบจำแนกหลายประเภท (Confusion Matrix for Multi-Class Classification)



รูปที่ 2.7 เมทริกซ์ความสับสนแบบจำแนกหลายประเภท
ที่มา Joydwip (2563)

การคำนวณประสิทธิภาพต่างจากแบบ 2 ประเภทเล็กน้อยเพราะแบบหลายคลาสจะไม่มีการแบ่งเป็นบวก (Positive) และลบ (Negative) แต่จะดูในแต่ละคลาสแทนตัวอย่างการคำนวณหาค่า TP,TN, FP, FN ของคลาสแอปเปิ้ลตามรูปที่ 2.7 คือ

- TP = 7
- TN = (2+3+2+1) = 8
- FP = (8+9)
- FN = (1+3) = 4

ส่วนขั้นตอนการวัดค่าเหมือนกับเมทริกซ์ความสับสนแบบจำแนก 2 ประเภท

2.7 ข้อมูลที่ไม่สมดุล (Imbalanced Datasets)

เป็นการที่จำนวนข้อมูลในแต่ละกลุ่มแตกต่างกันมากเกินปกติ ซึ่งปกติชุดข้อมูลที่ดีควรจะมีความสมดุล (Balance) เพราะถ้าข้อมูลในกลุ่มตัวแปรเป้าหมายมีความไม่สมดุล จะส่งผลต่อความถูกต้องการทำนาย ในที่นี้ผู้วิจัยได้ทำการใช้พารามิเตอร์ Stratify (Soni, 2022) ในขั้นตอนการแบ่งข้อมูล ซึ่งพารามิเตอร์ Stratify แบ่งข้อมูลให้มีสัดส่วนเท่ากันเช่น ตัวแปร y มี 0 อยู่ 25% และ1 อยู่ 75% ถ้าใช้พารามิเตอร์ Stratify จะทำให้ข้อมูลที่แบ่งมีค่า 0 อยู่ที่ 25% และ 1 อยู่ที่ 75% อย่างแน่นอน

2.8 การแปลงข้อความเป็นตัวเลข (Vectorizer)

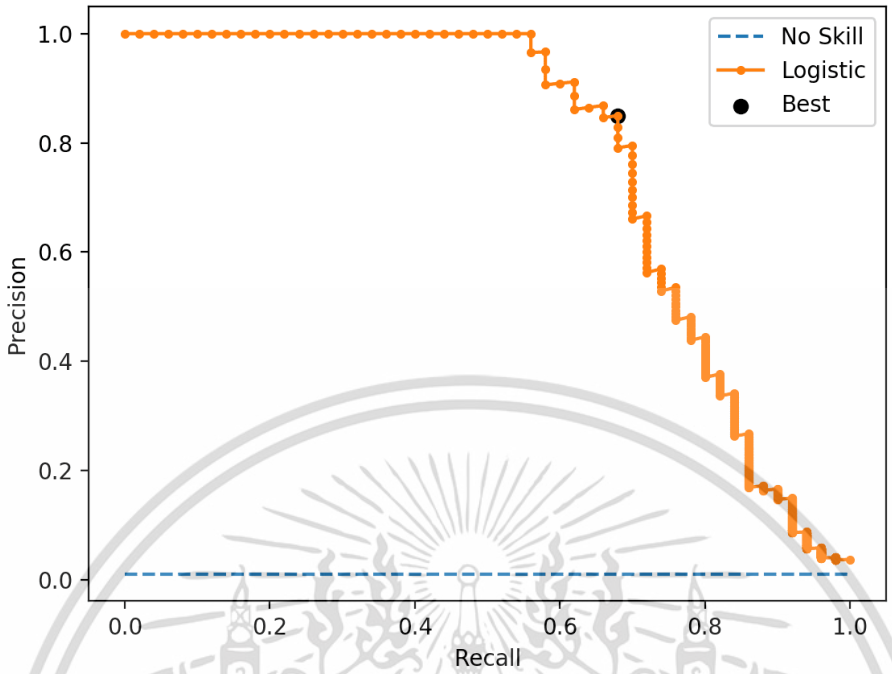
ผู้วิจัยได้ใช้ TFIDF ซึ่งย่อมาจาก Term Frequency-Inverse Document Frequency ซึ่งหมายถึงการเอาจำนวนครั้งที่แต่ละ Word id ปรากฏในแต่ละข้อความหารด้วยจำนวนข้อความทั้งหมดในข้อความนั้นแล้วจึงนำมาคูณกับจำนวนข้อความทั้งหมดหารด้วยจำนวนข้อความที่แต่ละ Word id ปรากฏอยู่แล้วใส่ Log เข้าไปดังรูปสมการที่ (2.14)

$$\mathrm{TFIDF}_w(\text{Term frequency} - \text{Inverse document frequency}) = \mathrm{t}f_{w,d} \times \log\frac{N}{df_w} \tag{2.4}$$

หาก Word id ไหนอยู่ในข้อความเป็นจำนวนมากก็จะมีค่ามากแต่ถ้าหาก word id นั้นไปปรากฏอยู่ในข้อความหลายข้อความมากเกินไปก็ค่าโดยรวมจะลดลง โดยสรุปคือ TFIDF จะช่วย Highlight คำเด่นๆออกมา

2.9 การกำหนดค่าเกณฑ์โดยใช้เส้นโค้งความแม่นยำและความครบถ้วน (Precision-Recall curve)

เส้นโค้งความแม่นยำและความครบถ้วน (Precision-Recall curve) คือเส้นโค้งที่จะแสดงค่าความแม่นยำ (Precision) ในแกนตั้งและค่าความครบถ้วน (Recall) ในแกนนอน โดยที่จะมีจุดที่ทำให้บาลานซ์ของความแม่นยำ (Precision) และค่าความครบถ้วน (Recall) ดีที่สุดจุดนั้นเรียกว่า F1-score ซึ่งปกติแล้วเส้นโค้งความแม่นยำและความครบถ้วน จะใช้ในการจัดหมวดหมู่แบบไบนารี การที่จะใช้เส้นโค้งความแม่นยำและความครบถ้วน ในการจัดหมวดหมู่แบบหลายคลาส จำเป็นที่จะต้องมองทุกคลาสเป็นแบบไบนารี ซึ่งจะวาดแต่ละคลาสได้ทีละคลาสเท่านั้น (Scikit-learn developers, 2024) ซึ่งผู้จะวิจัยจะใช้จุดนี้ ในการกำหนดเกณฑ์ในการนำไปใช้กับข้อมูลจริงดังรูปที่ 2.8



รูปที่ 2.8 ตัวอย่าง Precision-Recall curve

ที่มา Jason (2564)

2.10 งานวิจัยที่เกี่ยวข้อง

สุวพร (2565) ได้ทำการจัดหมวดหมู่แบบผสมสำหรับป่าสุ่มของต้นไม้และต้นไม้ตัดสินใจควบแน่นฝ่ายข้างน้อย สำหรับปัญหาคลาสไม่สมดุล ในการทดสอบจะนำข้อมูลที่สังเคราะห์มา 10 ชุดข้อมูล ผลเฉลี่ยมีค่า Precision, Recall, F-measure และ G-measure เป็น 0.6105, 0.7784, 0.6694, และ 0.6814 ตามลำดับ ซึ่งถือว่ามีคุณภาพที่สูง

ปฐวี (2564) ได้ทำการศึกษาการทำให้เป็นโทเค็น (Tokenization) โดยได้ทำการนำเสนอวิธีการเพิ่มข้อมูล (Data Augmentation) เพื่อเพิ่มความคงทนและประสิทธิภาพโดยใช้การทำให้เป็นโทเคนหลายรูปแบบ (Multi-Tokenization) วัดผลด้วยข้อความภาษาไทย ด้วยการใช้ Multi - Tokenization ผลปรากฏว่าสามารถเพิ่มความคงทนให้ข้อความที่ผ่านการ Tokenization ด้วยคุณภาพไม่ดีได้จริงซึ่งค่า F-1 ก่อนใช้การ Augmentation จะอยู่ที่ 74.3% ส่วนหลังจากใช้จะอยู่ที่ 76.6%

พงศ์ทวัส (2565) ได้ทำการพัฒนาเวิร์กโฟลว์สำหรับการสร้างต้นไม้จัดหมวดหมู่ที่ดีที่สุด ด้วยตัวแบบเชิงเส้นจำนวนเต็มแบบผสม ทำการประเมินประสิทธิภาพบนชุดเยอรมันเครดิต จากการเปรียบเทียบประสิทธิภาพ ระหว่างตัวแบบต้นไม้จำแนกประเภทที่ดีที่สุดกับต้นไม้ตัดสินใจ พบว่าต้นไม้จัดหมวดหมู่ที่ดีที่สุด ให้อัตราความถูกต้องสูงกว่าต้นไม้ตัดสินใจทั้งบนชุดข้อมูลสร้างตัวแบบ

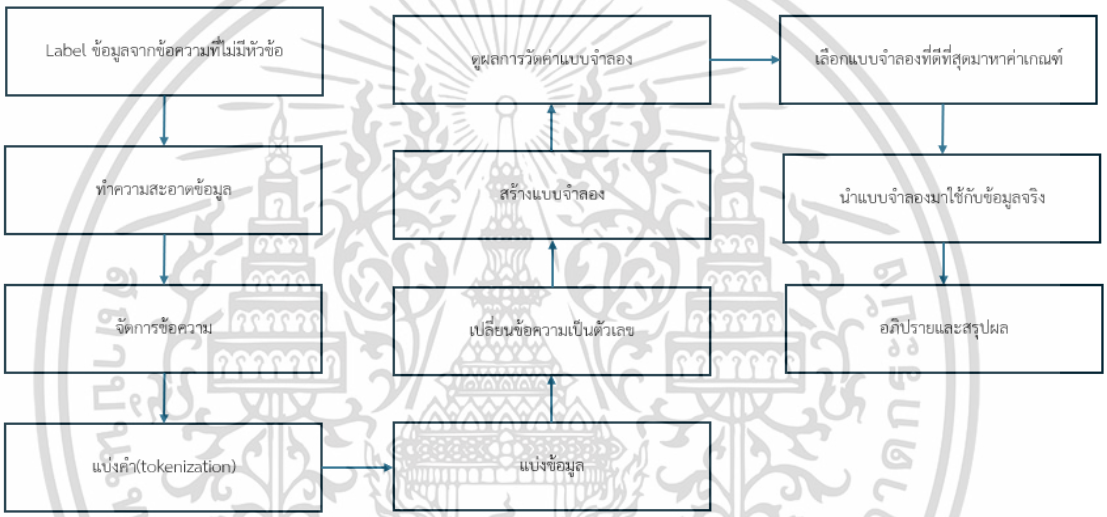
และบนชุดข้อมูลทดสอบ 0.4% ถึง 3.2% ทำให้เห็นว่าตัวแบบต้นไม้ จัดหมวดหมู่ที่ดีที่สุดรองรับข้อมูลสูญหายจำนวนมากและแสดงให้เห็นถึงเวิร์กโฟลว์ที่พัฒนาขึ้น

บทที่ 3
วิธีการดำเนินงานวิจัย

งานวิจัยนี้มีจุดประสงค์เพื่อเพิ่มหัวข้อที่จะเพิ่มหัวข้อที่จะนำเข้าแชทบอท ผู้วิจัยได้นำทฤษฎี แนวคิด และงานวิจัยที่เกี่ยวข้องมากำหนดขั้นตอนในการศึกษาดังนี้

3.1 ขั้นตอนการดำเนินงาน

ในการทำจิวัยครั้งนี้ มีขั้นตอนการดำเนินงาน 11 ขั้นตอนดังนี้



รูปที่ 3.1 ขั้นตอนการดำเนินงาน

3.2 การจัดเตรียมข้อมูล

3.2.1 Label ข้อมูลจากข้อความที่ไม่มีหัวข้อ (Intent)

นำข้อมูลที่เป็นหัวข้อ “Other” ของโมเดลที่มีหัวข้ออยู่แล้ว มาค้นคว้าว่าควรจะมีหัวข้อใหม่ๆ กี่หัวข้อ และทำการจัดข้อความนั้นๆ ให้มาอยู่ในหัวข้อเดียวกัน ได้ทั้งหมด 10 หัวข้อ แสดงดังตารางที่ 3.1

ตารางที่ 3.1 ตัวอย่างชื่อหัวข้อที่สร้างใหม่จากข้อมูล “Other”

ลำดับ	ตัวอย่างข้อความ	ตัวอย่างหัวข้อที่สร้างใหม่
1	สอบถามข้อมูลโทรศัพท์หน่อยครับราคาเท่าไหร่	โทรศัพท์
2	หนังสือเล่มนี้เกี่ยวกับอะไรครับ	หนังสือ
3	รถคันนี้เครื่องกี่ cc คะ	รถยนต์
4	เครื่องบินสวยจังเท่าไหร่ครับ	เครื่องบิน
5	เก้าอี้ทำมาจากอะไรครับ	เก้าอี้
6	คอมเครื่องนี้เล่นเกมได้ไหมครับ	คอมพิวเตอร์
7	พัดลมประหยัดไฟไหมครับ	พัดลม
8	ตุ๊กตาตัวนี้ทำมาจากอะไรครับ	ตุ๊กตา
9	ตู้เย็นนี้รับประกันกี่ปีครับ	ตู้เย็น
10	เครื่องซักผ้าหนักกี่กิโลครับ	เครื่องซักผ้า

จากนั้นบันทึกเป็นไฟล์ Excel และนำมาเข้าโปแกรม Colab เพื่อทำความสะอาดข้อมูลต่อไป

3.2.2 การทำความสะอาดข้อมูล (Cleaning data)

ในการการทำความสะอาดข้อมูล (Cleaning Data) ผู้วิจัยได้ทำการลบข้อมูลซ้ำ(Drop Duplicates) เพื่อที่แบบจำลองจะไม่ได้รับข้อมูลที่ซ้ำซ้อนไปประมวลผล

3.2.3 การจัดการกับข้อความ (Preprocess text)

3.2.3.1 การแทนคำ (Replace)

ในการแทนคำ (Replace) ผู้วิจัยได้ทำการสร้างฟังก์ชั่น (Function) สำหรับการแทนที่ (Replace) บางคำ ซึ่งเป็นคำที่ได้รับข้อความมาผิดหรือลูกค้าเข้าใจผิดบ่อยๆ สมมติตัวอย่างเช่น ควรจะเป็นคำว่า “รถยนต์” แต่ได้รับข้อความมาเป็น “รถใหญ่” ฟังก์ชั่นจะทำการเปลี่ยนเป็นคำว่ารถยนต์ เพื่อที่จะทำให้แบบจำลองเข้าใจได้อย่างถูกต้อง แสดงดังตารางที่ 3.2

ตารางที่3.2 ตัวอย่างการแทนคำ

คำที่แทน	**ข้อความดั้งเดิม**	**ข้อความที่แทนคำ**
แทน “รถใหญ่” เป็น “รถยนต์”	อยากสอบถามเรื่อง**รถใหญ่**คันนี้ครับว่าราคาเท่าไหร่	อยากสอบถามเรื่อง**รถยนต์**คันนี้ครับว่าราคาเท่าไหร่
แทน “ที่นั่ง” เป็น “เก้าอี้”	**ที่นั่ง**อันนี้ทำมาจำวัสดุอะไรครับ	**เก้าอี้**อันนี้ทำมาจากวัสดุอะไรครับ

3.2.3.2 การทำความสะอาดข้อความ (Clean text)

ในการทำความสะอาดข้อความ (Clean text) ผู้วิจัยได้สร้างฟังก์ชั่น (Function) ในการทำความสะอาดข้อความ ตัวฟังก์ชั่นจะทำการลบคำลงท้ายออกเช่น ค่ะ ครับ นะคะ เป็นต้น และทำการลบตัวอักษรพิเศษ ลบอีโมจิ ลบพื้นที่ว่าง แสดงดังตารางที่ 3.3

ตารางที่ 3.3 ตัวอย่างการทำความสะอาดข้อความ

ข้อความดั้งเดิม	**ข้อความที่ผ่านการทำความสะอาด**
สวัสดีค่ะอยากขอสอบถามว่า หนังสือแนวสืบสวนสอบสวน มีไหมคะ 😀	อยากขอสอบถามว่าหนังสือแนวสืบสวนสอบสวนมีไหม
สวัสดีครับเครื่องซักผ้ารุ่นนี้ รับประกันกี่ปีครับ 😄	เครื่องซักผ้ารุ่นนี้รับประกันกี่ปี

3.2.4 การแบ่งคำ (Tokenization)

ในการแบ่งคำ (Tokenization) ผู้วิจัยได้ทำการแบ่งคำโดยใช้ library Pythainlp ซึ่งจะมีพจนานุกรมที่เป็นโมดูลเรียกว่า Newmm จะทำให้มีการแบ่งคำที่ถูกต้อง ดังตารางที่ 3.4

ตารางที่ 3.4 ตัวอย่างการแบ่งคำ (Tokenization)

ข้อความดั้งเดิม	**ข้อความที่ผ่านการแบ่งคำ**
อยากขอสอบถามว่าหนังสือแนวสืบสวนสอบสวนมีไหม	อยาก ขอ สอบถาม ว่า หนังสือ แนว สืบสวน สอบสวน มี ไหม
เครื่องซักผ้ารุ่นนี้รับประกันกี่ปี	เครื่องซักผ้า รุ่น นี้ รับประกัน กี่ ปี

3.2.5 การแบ่งข้อมูล (Split data)

ในการแบ่งข้อมูล (Split data) ผู้วิจัยได้ทำการแบ่งข้อมูลออกเป็นข้อมูลสำหรับฝึกแบบจำลอง 80 เปอร์เซ็นต์ ข้อมูลสำหรับทดสอบแบบจำลอง 20 เปอร์เซ็นต์ เนื่องจากข้อมูลมีความไม่สมดุล เพราะมีบางหัวข้อที่มากกว่าปกติ และบางหัวข้อที่น้อยกว่าปกติ ผู้วิจัยจึงใช้พารามิเตอร์ Stratify มาช่วยซึ่งจะทำให้ค่าสัดส่วนของแต่ละหัวข้อหลังจากทำการแบ่งข้อมูลแล้วเท่ากัน โดยจำนวนข้อมูลทั้งหมด แสดงดังตารางที่ 3.5

ตารางที่ 3.5 ตัวอย่างจำนวนข้อมูลทั้งหมด

ลำดับ	หัวข้อ	จำนวน
1	โทรศัพท์	284
2	หนังสือ	97
3	รถยนต์	96
4	เครื่องบิน	89
5	เก้าอี้	76
6	คอมพิวเตอร์	74
7	พัดลม	50
8	ตุ๊กตา	47
9	ตู้เย็น	17
10	เครื่องซักผ้า	10
รวม		840

ข้อมูลสำหรับฝึกแบบจำลองแสดงดังตารางที่ 3.6

ตารางที่ 3.6 ข้อมูลสำหรับฝึกแบบจำลอง

ลำดับ	หัวข้อ	จำนวน
1	โทรศัพท์	227
2	หนังสือ	78
3	รถยนต์	77
4	เครื่องบิน	71
5	เก้าอี้	61
6	คอมพิวเตอร์	59
7	พัดลม	40

ตารางที่ 3.6 ข้อมูลสำหรับฝึกแบบจำลอง (ต่อ)

ลำดับ	หัวข้อ	จำนวน
8	ตู๊กตา	38
9	ตู้เย็น	13
10	เครื่องซักผ้า	8
รวม		672

ข้อมูลสำหรับทดสอบแบบจำลองแสดงดังตารางที่ 3.7

ตารางที่ 3.7 ข้อมูลสำหรับทดสอบแบบจำลอง

ลำดับ	หัวข้อ	จำนวน
1	โทรศัพท์	57
2	หนังสือ	19
3	รถยนต์	19
4	เครื่องบิน	18
5	เก้าอี้	15
6	คอมพิวเตอร์	15
7	พัดลม	10
8	ตู๊กตา	9
9	ตู้เย็น	4
10	เครื่องซักผ้า	2
รวม		168

3.2.6 การเปลี่ยนหัวข้อให้เป็นตัวเลข (LabelEncoder)

การเปลี่ยนหัวข้อให้เป็นตัวเลข (LabelEncoder) เพื่อให้ข้อมูลที่ไม่ใช่ตัวเลขทำให้เป็นตัวเลข แสดงดังตารางที่ 3.8

ตารางที่ 3.8 ตัวอย่างการเปลี่ยนหัวข้อให้เป็นตัวเลข (LabelEncoder)

ลำดับ	หัวข้อดั้งเดิม	หัวข้อที่ผ่านการเปลี่ยนเป็นตัวเลข
1	โทรศัพท์	3
2	หนังสือ	0
3	รถยนต์	2
4	เครื่องบิน	1
5	เก้าอี้	4
6	คอมพิวเตอร์	8
7	พัดลม	6
8	ตุ๊กตา	5
9	ตู้เย็น	7
10	เครื่องซักผ้า	9

3.2.7 เปลี่ยนข้อความเป็นตัวเลข (Vectorizer)

ในการเปลี่ยนข้อความเป็นตัวเลข (Vectorizer) ผู้วิจัยจะใช้ TF-IDF เพื่อเปรียบเทียบความถี่ของคำถ้าคำไหนอยู่ในข้อความหลายข้อความจะทำให้มีค่ามาก แต่หากว่ามีคำนั้นๆอยู่ในข้อมูลมากเกินไปก็ค่าโดยรวมจะลดลงผู้ วิจัยได้ใช้การนับความถี่แบบแยกเป็นตัวอักษร แสดงดังตารางที่3.9

ตารางที่ 3.9 การอย่างการใช้ TF-IDF ของข้อความ “อยาก ขอ สอบถาม ว่า หนังสือ แนว สืบสวน สอบสวน มี ไหม” 10 ตัว

ลำดับ	ตัวอักษร	ผลลัพธ์
1	“ข”	0.0510
2	“ขอ”	0.0510
3	“ม”	0.0510
4	“ส”	0.1021
5	“สอ”	0.0510
6	“ก”	0.0363
7	“น”	0.1452
8	“ว”	0.2041
9	“วน”	0.1021
10	“อ”	0.1816

3.2.8 ใช้ K-Fold Cross Validation

การใช้ K-Fold Cross Validation ผู้วิจัยได้ใช้การทำ Resempling แบบ K-Fold Cross Validation เพื่อสร้างและทดสอบโมเดล คำนวณค่าเฉลี่ย Accuracy หรือ error (i.e. Model performance) ก่อนที่จะนำโมเดลไปใช้ทำนายข้อมูล Test set

3.3 ตัวแปรที่ใช้ในการวิเคราะห์

ตัวแปรที่ใช้ในการวิเคราะห์เป็นข้อมูลนำเข้าในแบบจำลองการจัดหมวดหมู่ (Classification) ซึ่งประกอบไปด้วยตัวแปรตาม 1 คุณลักษณะและตัวแปรอิสระ 1 คุณลักษณะ ดังนี้

3.3.1 ตัวแปรตาม (Dependent Variable)

ตัวแปรตามที่ใช้ในการวิจัยครั้งนี้คือตัวแปร “Intent” ซึ่งจะมี 10 คลาสคือ 10 หัวข้อใหม่ที่ผู้วิจัยได้ศึกษาค้นคว้ามา

ตารางที่ 3.10 ตัวแปรตามที่ใช้ในการวิเคราะห์

ลำดับ	ชื่อตัวแปร	คำอธิบาย	มาตรวัด
1	Intent	เป็นหัวข้อที่ผู้วิจัยได้คิดขึ้นมาใหม่ 10 หัวข้อ	นามบัญญัติ (Nominal)

3.3.2 ตัวแปรอิสระ (Independent Variable)

ตัวแปรอิสระในครั้งนี้เป็นข้อความที่โดนแปลงเป็นตัวเลขที่นำเข้าแบบจำลองเพื่อเป็นปัจจัยในการทำนาย

ตารางที่ 3.11 ตัวแปรอิสระที่ใช้ในการวิเคราะห์

ลำดับ	ชื่อตัวแปร	คำอธิบาย	มาตรวัด
1	keyword	เป็นข้อความที่โดนแปลงเป็นตัวเลข	นามบัญญัติ (Nominal)

3.4 การสร้างแบบจำลอง

ในการสร้างแบบจำลองผู้วิจัยได้ทำการสร้างแบบจำลอง 3 แบบคือ Random Forest(RF) , Support Vector Machines (SVM) และ Support Vector Machines (SVM) แบบ Kernel โดยข้อมูลจะแบ่งเป็นสองชุด คือข้อมูลชุดฝึกสอน (Training Dataset) 80% และข้อมูลชุดทดสอบ (Testing Dataset) 20% โดยจะเป็นข้อมูลที่เป็นข้อความหัวข้อ “อื่นๆ” ที่ผู้วิจัยได้ทำการจัดหมวดหมู่ให้เป็น 10 หัวข้อใหม่ภายในปี พ.ศ.2566

3.5 เครื่องมือที่ใช้ในการวิจัย

ทั้งในการจัดการข้อมูลเพื่อนำไปใช้ในแบบจำลอง การสร้างแบบจำลอง และการวัดประสิทธิภาพของแบบจำลอง ผู้วิจัยจะดำเนินการด้วยการใช้โปรแกรมภาษาไพทอน (Python 3.10.12) บน Colab Notebook และใช้ไลบรารีที่จำเป็นต่อการสร้างแบบจำลองและวิเคราะห์ แสดงดังตารางที่ 3.12

ตารางที่ 3.12 ไลบรารีที่ใช้ในการสร้างแบบจำลองและวิเคราะห์

Library	**คำอธิบาย**
Pandas	ใช้ในการจัดการและวิเคราะห์ข้อมูล
Numpy	ใช้ในการจัดการกับตัวเลข
Matplotlib	ใช้ในการสร้างกราฟและแสดงผล
Seaborn	ใช้ในการสร้างกราฟและแสดงผล
Sklearn	ใช้ในการทำแบบจำลองรวมถึงการจัดการข้อมูลก่อนทำแบบจำลอง
Pythainlp	ใช้ในการจัดการกับข้อความภาษาไทย
Re	ใช้ในการทำความสะอาดข้อความ

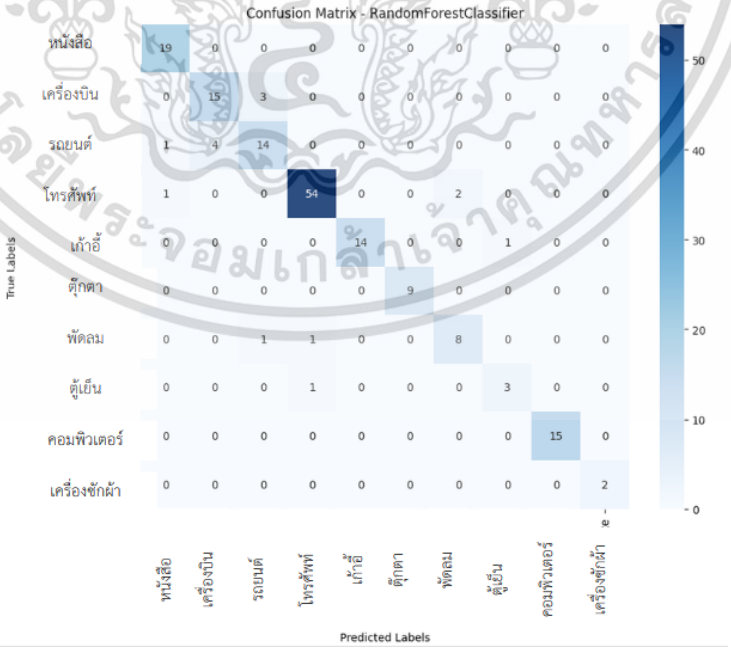
บทที่ 4
ผลการวิจัยและการอภิปรายผล

ในบทนี้จะนำเสนอผลของการเรียนรู้ของเครื่อง (Machine Learning) ในโมเดลการจัดหมวดหมู่ (Classification) ที่จะนำมาจำแนกหัวข้อ โดยใช้ข้อมูลเป็นข้อความในการสร้างแบบจำลอง ทั้ง 3 แบบคือ Random Forest (RF) Support Vector Machines (SVM) และ Support Vector Machines (SVM) แบบ Kernel โดยดูผลการทดสอบประสิทธิภาพในแต่ละแบบ

4.1 การทดสอบประสิทธิภาพของแบบจำลอง

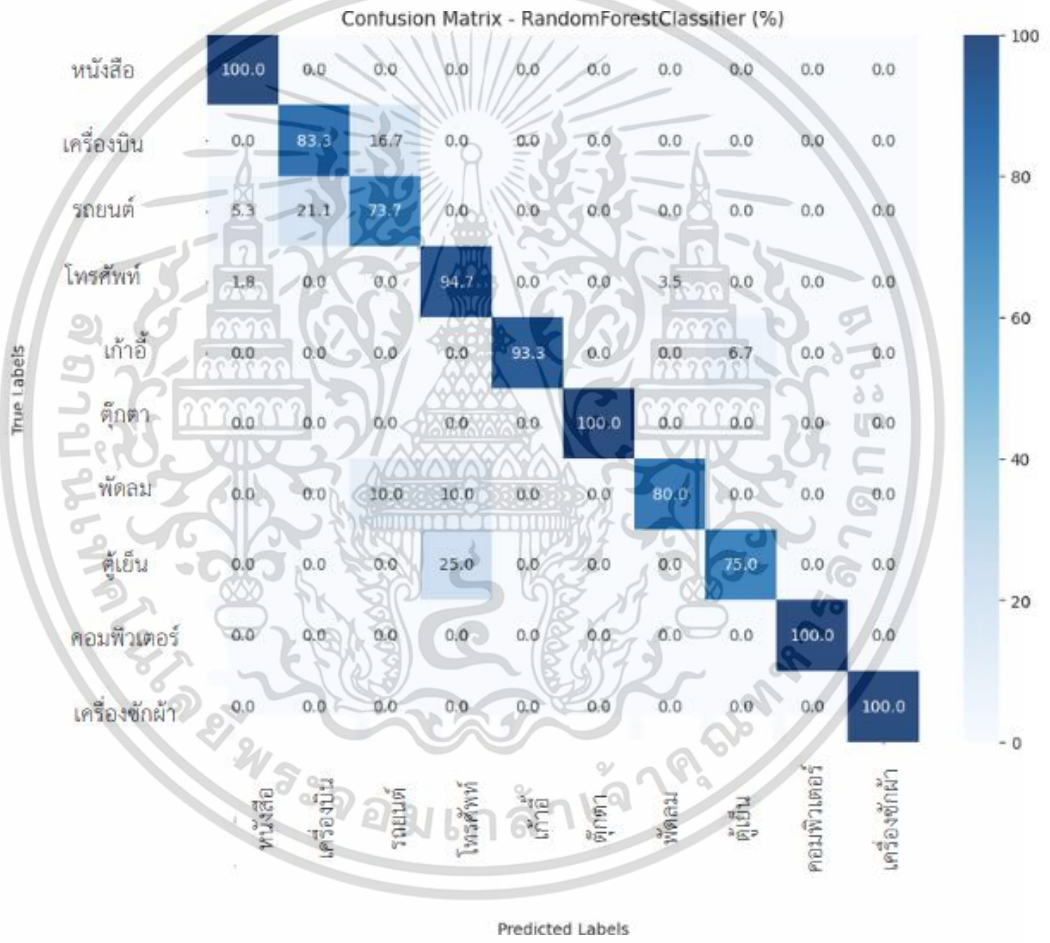
4.1.1 ประสิทธิภาพของแบบจำลองป่าสุ่ม (Random Forest: RF)

ในการสร้างแบบจำลองเพื่อที่จะหาหัวข้อใหม่โดยใช้แบบจำลองป่าสุ่ม โดยกำหนดค่าพารามิเตอร์ โดยใช้ GridSearchCV กำหนดค่า n_estimator ไว้เป็น 100, 200, 500 กำหนดค่า max_depth ไว้เป็น 0, 10, 20, 50 กำหนดค่า min_samples_split ไว้เป็น 2, 5, 10 กำหนดค่า min_samples_leaf ไว้เป็น 1, 2, 4 ผลคือค่าพารามิเตอร์ที่ดีที่สุดของแบบจำลองป่าสุ่ม คือ n_estimator = 100 max_depth = 50 min_samples_split = 10 min_samples_leaf = 1 ผลการทดสอบโดยใช้เมทริกซ์ความสับสนแบบหลายคลาส (Confusion Matrix for Multi-Class) แสดงดังรูปที่ 4.1



รูปที่ 4.1 เมทริกซ์ความสับสนแบบหลายคลาสของแบบจำลองป่าสุ่ม (Random Forest: RF)

จากรูปจะเห็นได้ว่าแบบจำลองป่าสุ่ม (Random Forest: RF) ผลลัพธ์คือ คลาสหนังสือแบบจำลองทำนายถูกทั้งหมด 19 ค่าจากทั้งหมด 19 ค่า คลาสเครื่องบินแบบจำลองทำนายถูกทั้งหมด 15 ค่าจากทั้งหมด 18 ค่า คลาสรถยนต์ทำนายถูก 14 ค่าจากทั้งหมด19 ค่า คลาสโทรศัพท์ทำนายถูก 54 ค่าจากทั้งหมด 57 ค่า คลาสเก้าอี้ทำนายถูก 14 ค่าจากทั้งหมด 15 ค่า คลาสตุ๊กตาทำนายถูก 9 ค่าจากทั้งหมด 9 ค่า คลาสพัดลมทำนายถูก 8 ค่าจากทั้งหมด 10 ค่า คลาสตู้เย็นทำนายถูก3 ค่าจากทั้งหมด 4 ค่า คลาสคอมพิวเตอร์ทำนายถูก 15 ค่า จากทั้งหมด 15 ค่า คลาสเครื่องซักผ้าทำนายถูกทั้งหมด 2 ค่าจากทั้งหมด 2 ค่า ซึ่งการที่จะดูให้ชัดเจนยิ่งขึ้น จึงนำเมทริกซ์ความสับสนแบบหลายคลาสของแบบจำลองป่าสุ่มไปทำเป็นแบบเปอร์เซ็นต์แสดงดังรูปที่ 4.2



รูปที่ 4.2 เมทริกซ์ความสับสนแบบหลายคลาสของแบบจำลองป่าสุ่ม (Random Forest:RF) แบบเป็นเปอร์เซ็นต์

จากรูปจะเห็นได้ว่าโดยรวมแล้วแบบจำลองป่าสุ่ม (Random Forest: RF) มีผลการทำนายที่แม่นยำ โดยคลาสที่ทำนายได้ต่ำที่สุดคือคลาสรถยนต์แต่ก็ยังมีความแม่นยำถึง 73.7 เปอร์เซ็นต์ และความแม่นยำที่มากที่สุดมีถึง 4 คลาสที่มีความแม่นยำถึง 100 เปอร์เซ็นต์คือคลาสหนังสือ ตุ๊กตา คอมพิวเตอร์ และเครื่องซักผ้า ซึ่งถือว่ามีความแม่นยำที่ดี ต่อมาจึงคำนวณประสิทธิภาพการทำนายของแบบจำลองป่าสุ่ม (Random Forest: RF) แสดงดังตารางที่ 4.1

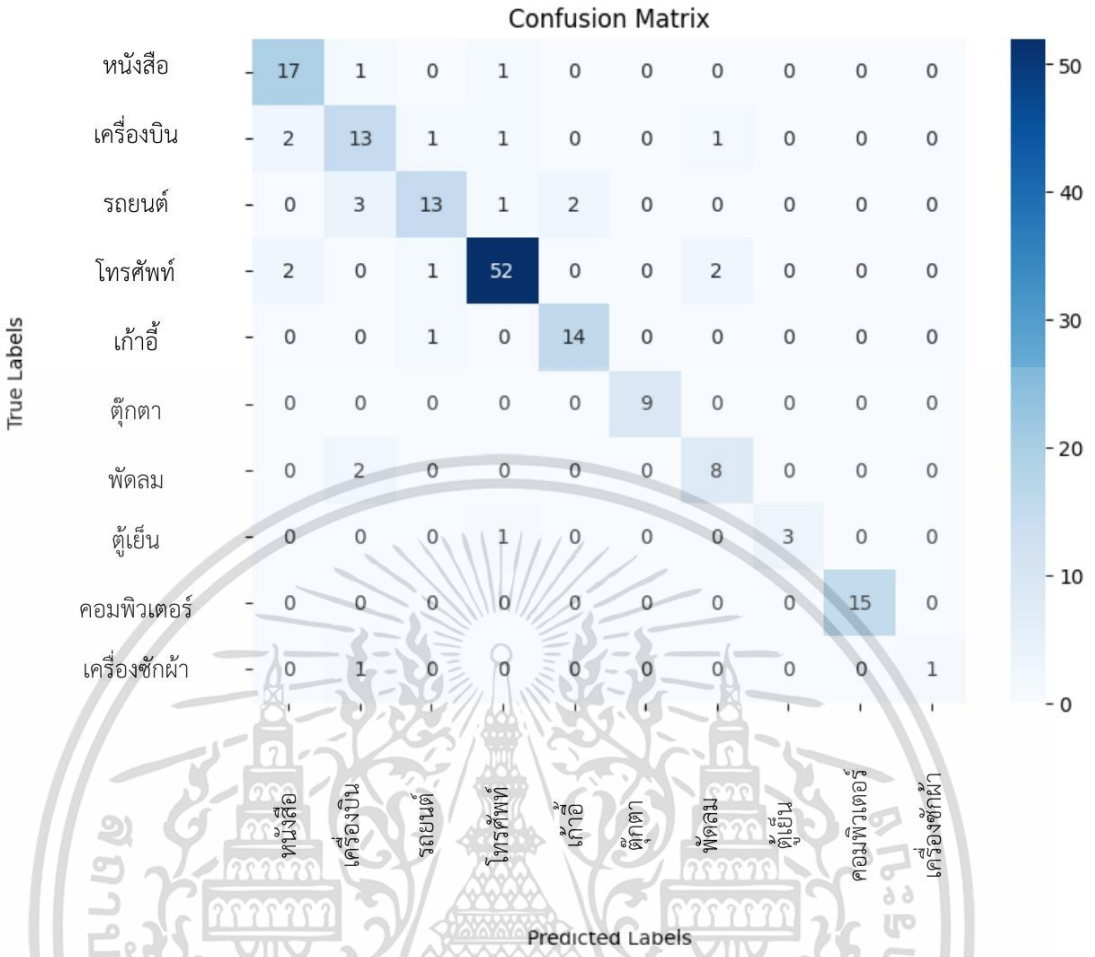
ตารางที่ 4.1 ประสิทธิภาพในการทำนายของแบบจำลองป่าสุ่ม (Random Forest: RF)

คลาสที่ใช้ในการทดสอบ	ประสิทธิภาพในการทำนายของแบบจำลอง			
	Precision	Recall	F1-Score	Accuracy
หนังสือ	0.90	1.00	0.95	0.91
เครื่องบิน	0.79	0.83	0.81	
รถยนต์	0.78	0.74	0.76	
โทรศัพท์	0.96	0.95	0.96	
เก้าอี้	1.00	0.93	0.97	
ตุ๊กตา	1.00	1.00	1.00	
พัดลม	0.80	0.80	0.80	
ตู้เย็น	0.75	0.75	0.75	
คอมพิวเตอร์	1.00	1.00	1.00	
เครื่องซักผ้า	1.00	1.00	1.00	
ค่าเฉลี่ย	0.91	0.91	0.91	

จากตารางที่ 4.1 ประสิทธิภาพในการทำนายของแบบจำลองป่าสุ่ม (Random Forest: RF) จะมีค่า Accuracy อยู่ที่ 91% และค่าความแม่นยำ (Precision) อยู่ที่ 91% ค่าความครบถ้วน(Recall) อยู่ที่ 91% ค่าประสิทธิภาพโดยรวม (F1-Score) อยู่ที่ 91% ซึ่งถือว่าแบบจำลองป่าสุ่ม (Random Forest: RF) สามารถทำนายผลได้มีความแม่นยำสูง

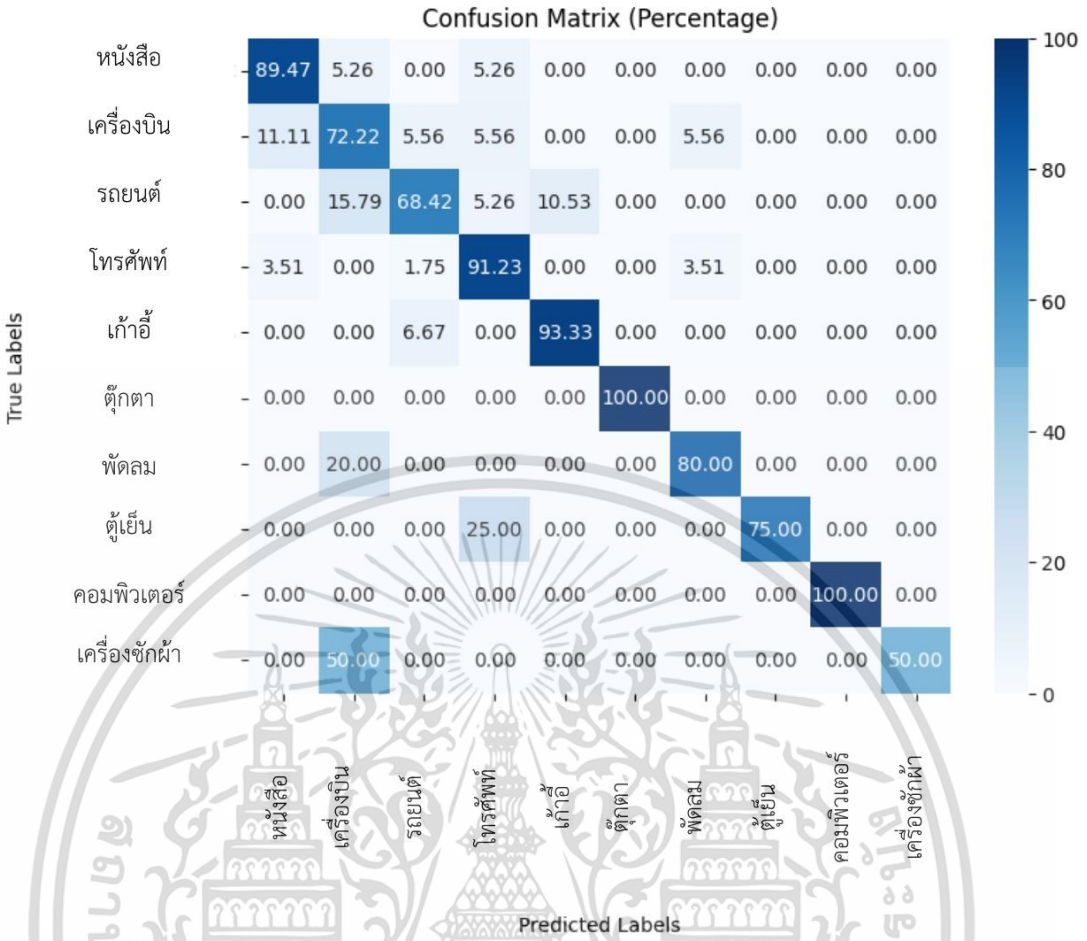
4.1.2 ประสิทธิภาพแบบจำลอง Support Vector Machines (SVM)

ในการสร้างแบบจำลองเพื่อที่จะหาหัวข้อใหม่ โดยใช้แบบจำลอง Support Vector Machines (SVM) โดยกำหนดค่าพารามิเตอร์โดยใช้ GridSearchCV กำหนดค่า svc__C ไว้เป็น 0.1,1,10 ผลคือค่าพารามิเตอร์ที่ดีที่สุดของแบบจำลอง แบบจำลอง Support Vector Machines (SVM) คือ svc__C = 1 ผลการทดสอบโดยใช้เมทริกซ์ความสับสนแบบหลายคลาส (Confusion Matrix for Multi-Class) แสดงดังรูปที่ 4.3



รูปที่ 4.3 เมทริกซ์ความสับสนแบบหลายคลาสของแบบจำลอง Support Vector Machine (SVM)

จากรูปจะเห็นได้ว่าแบบจำลอง Support Vector Machine (SVM) ผลลัพธ์คือ คลาสหนังสือ แบบจำลองทำนายถูกทั้งหมด 17 ค่าจากทั้งหมด 19 ค่า คลาสเครื่องบิน แบบจำลองทำนายถูกทั้งหมด 13 ค่าจากทั้งหมด 18 ค่า คลาสรถยนต์ทำนายถูก 13 ค่าจากทั้งหมด 19 ค่า คลาสโทรศัพท์ทำนายถูก 52 ค่าจากทั้งหมด 57 ค่า คลาสเก้าอี้ทำนายถูก 14 ค่าจากทั้งหมด 15 ค่าคลาสตุ๊กตาทำนายถูก 9 ค่าจากทั้งหมด 9 ค่าคลาสพัดลมทำนายถูก 8 ค่าจากทั้งหมด 10 ค่า คลาสตู้เย็นทำนายถูก 3 ค่าจากทั้งหมด 4 ค่า คลาสคอมพิวเตอร์ทำนายถูก 15 ค่าจากทั้งหมด 15 ค่า คลาสเครื่องซักผ้าทำนายถูกทั้งหมด 1 ค่าจากทั้งหมด 2 ค่า และผู้วิจัยได้นำเมทริกซ์ความสับสนแบบหลายคลาสของแบบจำลอง Support Vector Machine (SVM) ไปทำเป็นรูปแบบเปอร์เซ็นต์เพื่อจะดูให้ชัดเจนยิ่งขึ้นแสดงดังรูปที่ 4.4



รูปที่ 4.4 เมทริกซ์ความสับสนแบบหลายคลาสของแบบจำลอง Support Vector Machine (SVM) แบบเปอร์เซ็นต์

จากรูปจะเห็นได้ว่าทำนายได้ค่อนข้างด้อยกว่า Random Forest(RF) เพราะคลาสเครื่องบินแบบจำลอง Support Vector Machine (SVM) มีความแม่นยำที่ 72.22% แต่แบบจำลอง Random Forest (RF) มีความแม่นยำอยู่ที่ 83.3% คลาสรถยนต์แบบจำลอง SVM มีความแม่นยำที่ 68.42% แต่แบบจำลอง RF มีความแม่นยำอยู่ที่ 73.7% และคลาสเครื่องซักผ้าแบบจำลอง SVM มีความแม่นยำที่ 50% แต่แบบจำลอง RF มีความแม่นยำอยู่ที่100% ต่อมาจึงคำนวณประสิทธิภาพการทำนายของแบบจำลอง Support Vector Machine (SVM) แสดงดังตารางที่ 4.2

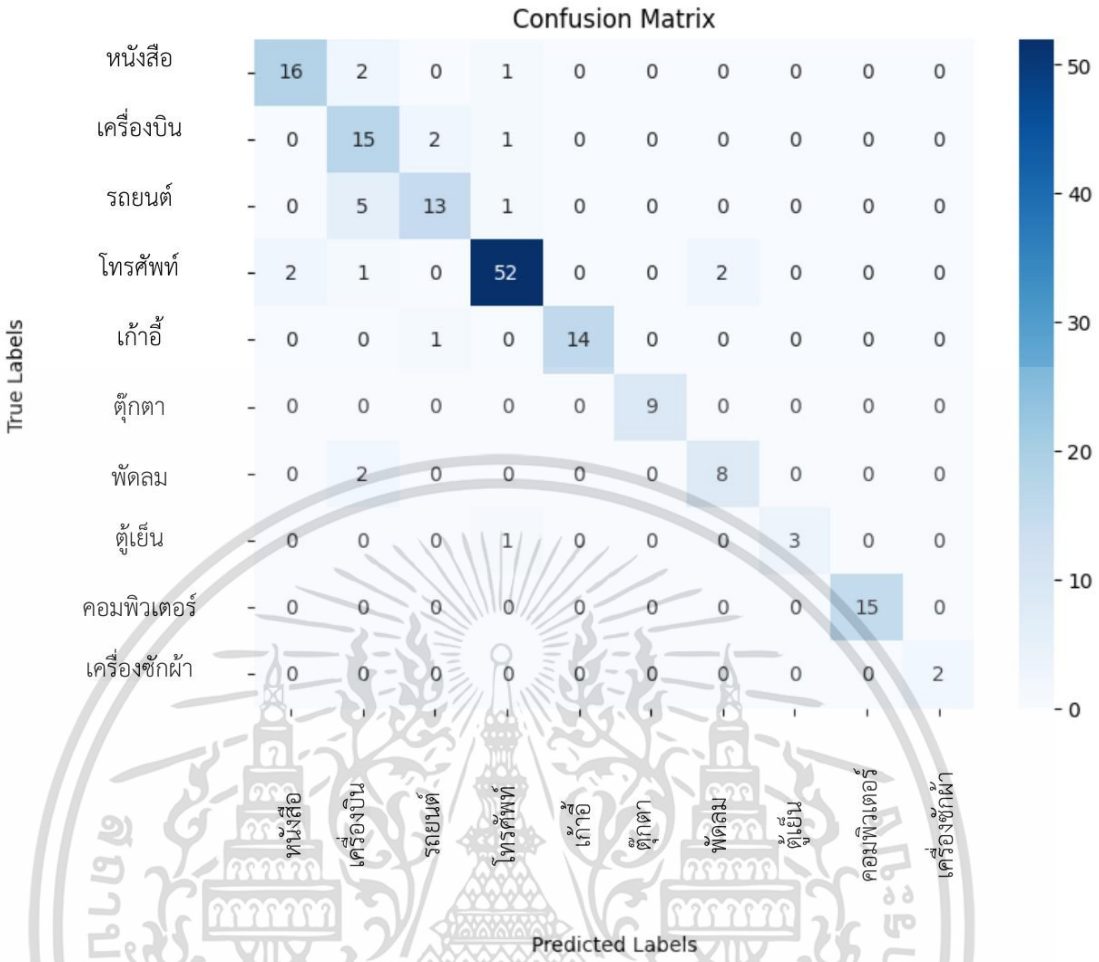
ตารางที่ 4.2 ประสิทธิภาพในการทำนายของแบบจำลอง Support Vector Machine (SVM)

คลาสที่ใช้ในการทดสอบ	**ประสิทธิภาพในการทำนายของแบบจำลอง**			
	Precision	**Recall**	**F1-Score**	**Accuracy**
หนังสือ	0.81	0.89	0.85	0.86
เครื่องบิน	0.65	0.72	0.68	
รถยนต์	0.81	0.68	0.74	
โทรศัพท์	0.93	0.91	0.92	
เก้าอี้	0.88	0.93	0.90	
ตุ๊กตา	1.00	1.00	1.00	
พัดลม	0.73	0.80	0.76	
ตู้เย็น	1.00	0.75	0.86	
คอมพิวเตอร์	1.00	1.00	1.00	
เครื่องซักผ้า	1.00	0.50	0.67	
ค่าเฉลี่ย	0.87	0.86	0.86	

จากตารางที่ 4.2 ประสิทธิภาพในการทำนายของแบบจำลอง Support Vector Machine (SVM) จะมีค่า Accuracy อยู่ที่ 86% และค่าความแม่นยำ (Precision) อยู่ที่ 87% ค่าความครบถ้วน (Recall) อยู่ที่ 87% ค่าประสิทธิภาพโดยรวม (F1-Score) อยู่ที่ 86% ซึ่งถือว่าแบบจำลอง Support Vector Machine (SVM) สามารถทำนายผลได้มีความแม่นยำปานกลาง

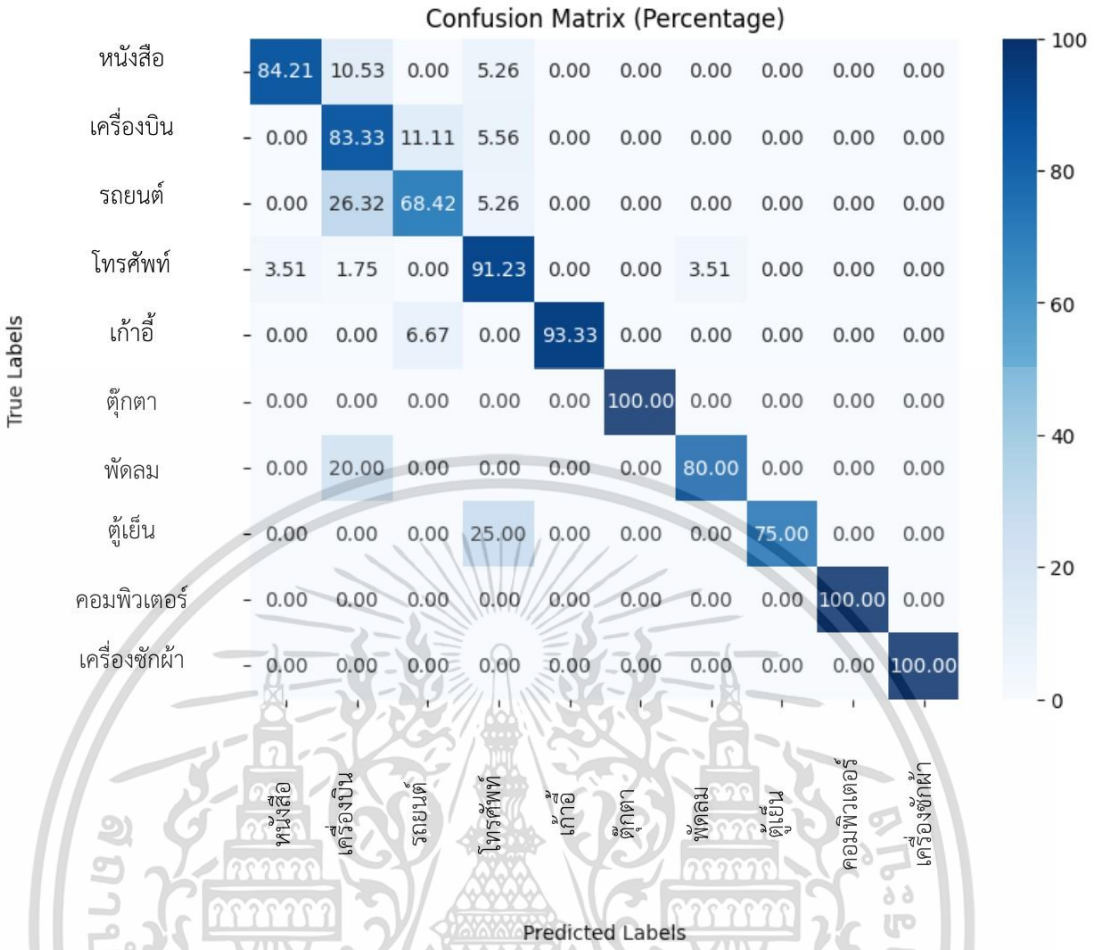
4.1.3 ประสิทธิภาพแบบจำลอง Support Vector Machines (SVM) แบบ Kernel

ในการสร้างแบบจำลองเพื่อที่จะหาหัวข้อใหม่ โดยใช้แบบจำลอง Support Vector Machines (SVM) แบบ Kernel โดยกำหนดค่าพารามิเตอร์โดยใช้ GridSearchCV กำหนดค่า svc__C ไว้เป็น 0.1,1,10 กำหนดค่า SVM_gamma ไว้เป็น 0.1 ผลคือค่าพารามิเตอร์ที่ดีที่สุดของแบบจำลองแบบจำลอง Support Vector Machines (SVM) แบบ Kernel คือ svc__C = 10 SVM_gamm = 0.1 ผลการทดสอบโดยใช้เมทริกซ์ความสับสนแบบหลายคลาส (Confusion Matrix for Multi-Class) เป็นไปดังรูปที่ 4.5



รูปที่ 4.5 เมทริกซ์ความสับสนแบบหลายคลาสของแบบจำลอง Support Vector Machine (SVM) แบบ Kernel

จากรูปจะเห็นได้ว่าแบบจำลอง Support Vector Machine (SVM) แบบ Kernel ผลลัพธ์คือ คลาสหนังสือแบบจำลองทำนายถูกทั้งหมด 16 ค่า จากทั้งหมด 19 ค่า คลาสเครื่องบินแบบจำลองทำนายถูกทั้งหมด 15 ค่าจากทั้งหมด 18 ค่า คลาสรถยนต์ทำนายถูก 13 ค่าจากทั้งหมด 19 ค่า คลาสโทรศัพท์ทำนายถูก 52 ค่าจากทั้งหมด 57ค่า คลาสเก้าอี้ทำนายถูก 14 ค่า จากทั้งหมด 15 ค่าคลาสตุ๊กตาทำนายถูก 9 ค่าจากทั้งหมด 9 ค่า คลาสพัดลมทำนายถูก 8 ค่าจากทั้งหมด 10 ค่า คลาสตู้เย็นทำนายถูก 3 ค่าจากทั้งหมด 4 ค่า คลาสคอมพิวเตอร์ทำนายถูก 15 ค่า จากทั้งหมด 15 ค่า คลาสเครื่องซักผ้าทำนายถูกทั้งหมด 1 ค่า จากทั้งหมด 2 ค่า และผู้วิจัยได้นำเมทริกซ์ความสับสนแบบหลายคลาสของแบบจำลอง Support Vector Machine (SVM) แบบ Kernel ไปทำเป็นรูปแบบเปอร์เซ็นต์เพื่อจะดูให้ชัดเจนยิ่งขึ้นแสดงดังรูปที่ 4.6



รูปที่ 4.6 เมทริกซ์ความสับสนแบบหลายคลาสของแบบจำลอง Support Vector Machine (SVM) แบบ Kernel รูปแบบเปอร์เซ็นต์

จากรูปจะเห็นได้ว่าทำนายได้ค่อนข้างด้อยกว่า Random Forest (RF) เพราะคลาสรถยนต์แบบจำลอง SVM แบบ Kernel มีความแม่นยำที่ 68.42% แต่แบบจำลอง RF มีความแม่นยำอยู่ที่ 73.7% และคลาสโทรศัพท์แบบจำลอง SVM แบบ Kernel มีความแม่นยำที่ 91.23% แต่แบบจำลอง RF มีความแม่นยำอยู่ที่ 94.7% ต่อมาจึงคำนวณประสิทธิภาพการทำนายของแบบจำลอง Support Vector Machine (SVM) แบบเปอร์เซ็นต์ แสดงดังตารางที่ 4.3

ตารางที่ 4.3 ประสิทธิภาพในการทำนายของแบบจำลอง Support Vector Machine (SVM) แบบ Kernel

คลาสที่ใช้ในการทดสอบ	ประสิทธิภาพในการทำนายของแบบจำลอง			
	Precision	Recall	F1-Score	Accuracy
หนังสือ	0.89	0.84	0.86	0.88
เครื่องบิน	0.60	0.83	0.70	
รถยนต์	0.81	0.68	0.74	
โทรศัพท์	0.93	0.91	0.92	
เก้าอี้	1.00	0.93	0.97	
ตุ๊กตา	1.00	1.00	1.00	
พัดลม	0.80	0.80	0.80	
ตู้เย็น	1.00	0.75	0.86	
คอมพิวเตอร์	1.00	1.00	1.00	
เครื่องซักผ้า	1.00	1.00	1.00	
ค่าเฉลี่ย	0.89	0.88	0.88	

จากตารางที่ 4.3 ประสิทธิภาพในการทำนายของแบบจำลอง Support Vector Machine (SVM) แบบ Kernel จะมีค่า Accuracy อยู่ที่ 88% และค่าความแม่นยำ (Precision) อยู่ที่ 89% ค่าความครบถ้วน (Recall) อยู่ที่ 88% ค่าประสิทธิภาพโดยรวม (F1-Score) อยู่ที่ 88% ซึ่งถือว่าแบบจำลอง Support Vector Machine (SVM) Kernel สามารถทำนายผลได้มีความแม่นยำปานกลาง

4.1.4 เปรียบเทียบประสิทธิภาพของทุกโมเดล

เมื่อหาประสิทธิภาพของทุกแบบจำลองได้แล้วจึงนำแบบจำลองมาเปรียบเทียบว่าแบบจำลองไหนมีประสิทธิภาพมากที่สุด ดังตารางที่ 4.4

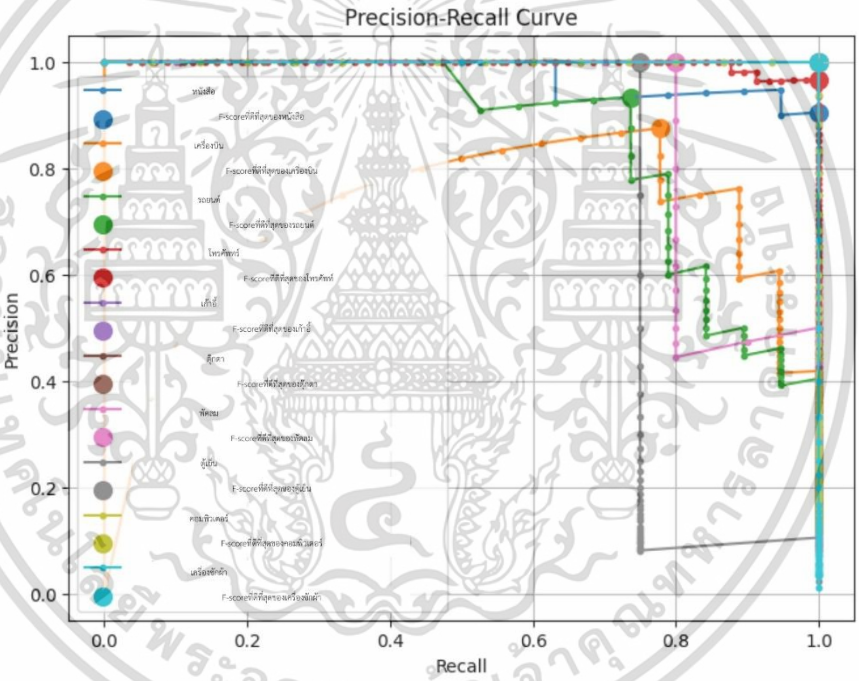
ตารางที่ 4.4 เปรียบเทียบประสิทธิภาพแบบจำลอง

แบบจำลองที่ใช้ในการทดสอบ	Precision	Recall	F1-Score	Accuracy
Random Forest(RF)	0.91	0.91	0.91	0.91
SVM	0.87	0.86	0.86	0.86
SVM แบบ Kernel	0.89	0.88	0.88	0.88

เมื่อดูจากตารางแล้วจะเห็นว่าแบบจำลอง Random Forest (RF) สามารถทำประสิทธิภาพได้ดีกว่าในทุกการวัดผล ผู้วิจัยจึงนำมาใช้ในการกำหนดเกณฑ์ (Threshold) ต่อไป

4.2 การกำหนดเกณฑ์ (Find Threshold)

หลังจากได้โมเดลที่ดีที่สุดแล้วจึงทำไปหาเกณฑ์ (Threshold) ที่ดีที่สุดด้วยเส้นโค้งความเม่นยำและความถูกต้องที่จะหาค่า F-1 โดยมองทุกคลาสเป็นแบบการจัดหมวดหมู่แบบไบนารี เพื่อที่จะจัดหมวดข้อความที่ไม่เกี่ยวกับหัวข้อทั้ง 10 หัวข้อแล้วนำข้อความนั้นไปเป็นหัวข้อ “อื่นๆ” แทนเช่น คำว่า “กินข้าวหรือยัง” “ฝนตกไหม” โดยการกำหนดเกณฑ์นั้นจะใช้ค่า Probability ที่ทำให้ค่า F1 ในแต่ละคลาสมากที่สุดซึ่งค่า F1 ในที่นี้จะเป็นค่า F1 ที่คิดแยกในแต่ละคลาสซึ่งค่าโดยมองแต่ละคลาสเป็นแบบไบนารีทำให้ค่าที่ได้ แสดงดังรูปที่ 4.7



รูปที่ 4.7 เส้นโค้งความแม่นยำและความถูกต้อง

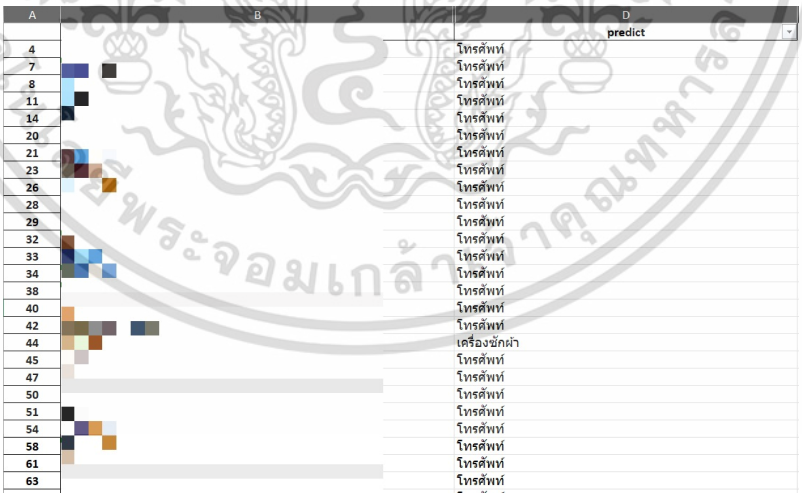
เส้นโค้งความแม่นยำและความถูกต้องได้มาจากการนำแต่ละคลาสมาคิดหาค่า Probability ที่ทำให้ได้ค่า F1 ที่ดีที่สุดซึ่งมองแต่ละคลาสเป็นแบบไบนารีจะแสดงดังตารางที่ 4.5

ตารางที่ 4.5 ค่า Probability ที่ทำให้ค่า F1 ที่คิดแยกแบบไบนารีในแต่ละคลาสสูงที่สุด

คลาสที่ใช้ในการทดสอบ	ค่า F1	ค่า Probability
หนังสือ	0.95	0.1782
เครื่องบิน	0.82	0.3514
รถยนต์	0.82	0.4595
โทรศัพท์	0.98	0.1822
เก้าอี้	1.00	0.1832
ตุ๊กตา	1.00	0.0891
พัดลม	0.88	0.5786
ตู้เย็น	0.85	0.2813
คอมพิวเตอร์	1.00	0.1381
เครื่องซักผ้า	1.00	0.1572

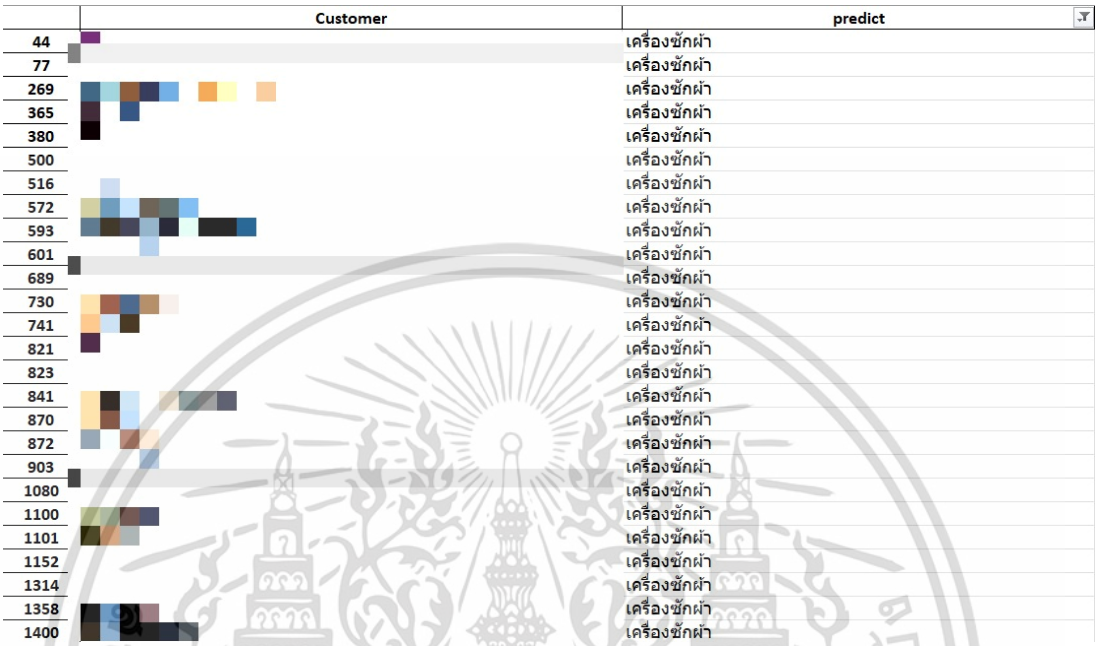
4.3 การใช้งานจริง

หลังจากที่กำหนดเกณฑ์ (Threshold) เสร็จเรียบร้อยแล้วจึงนำโมเดลไปใช้งานจริงซึ่งผลปรากฏว่ามี 1 หัวข้อที่ไม่สามารถใช้งานได้คือหัวข้อ “โทรศัพท์”และ 1 หัวข้อที่ใช้งานได้ไม่ดีคือหัวข้อ “เครื่องซักผ้า” ได้ดังรูปที่ 4.8 และรูปที่ 4.9 ตามลำดับ



รูปที่ 4.8 การใช้งานจริงหัวข้อ “โทรศัพท์”

หัวข้อ “โทรศัพท์” มีปัญหาคือข้อความที่ไม่เกี่ยวข้องกับ 10 หัวข้อที่ผู้วิจัยได้สร้างไว้ มารวมกันอยู่ที่หัวข้อนี้ทำให้การใช้งานจริง มีข้อความที่ควรจะเป็นหัวข้อ “อื่นๆ” อยู่มาก ซึ่งรูปที่ 4.7 ที่ผู้วิจัยระบายสีทึบไว้ด้านซ้าย เกือบทั้งหมดเป็นข้อความที่ไม่เกี่ยวข้องกับหัวข้อ “โทรศัพท์”



รูปที่ 4.9 การใช้งานจริงหัวข้อ “เครื่องซักผ้า”

หัวข้อ “เครื่องซักผ้า” แตกต่างออกไปเล็กน้อย เนื่องจากว่าหัวข้อ “เครื่องซักผ้า”ข้อมูลมีคำว่า “ต่อ” อยู่ทุกคำทำให้การใช้จริงจะมีข้อความไม่มีเกี่ยวกับหัวข้อ “เครื่องซักผ้า” แต่มีคำว่า “ต่อ” เข้ามาในหัวข้อนี้

4.3.1 การแก้ปัญหา

เนื่องจากมีหัวข้อที่มีปัญหา ผู้วิจัยจึงได้ทำการทดลองเพิ่มเกณฑ์ (Threshold) ของทั้งหัวข้อ “โทรศัพท์” และ “เครื่องซักผ้า” เพื่อที่จะทำให้แบบจำลองมีความเข้มงวดกับทั้ง 2 หัวข้อมากยิ่งขึ้นซึ่งพบว่าค่า Probability ที่เพิ่มขึ้น 0.4 สามารถทำนายกับข้อมูลจริงได้ดี ผู้วิจัยจึงเพิ่มค่า Probability เข้าไป 0.4 ดังตารางที่ 4.6

ตารางที่ 4.6 การเพิ่มเกณฑ์ (Threshold)

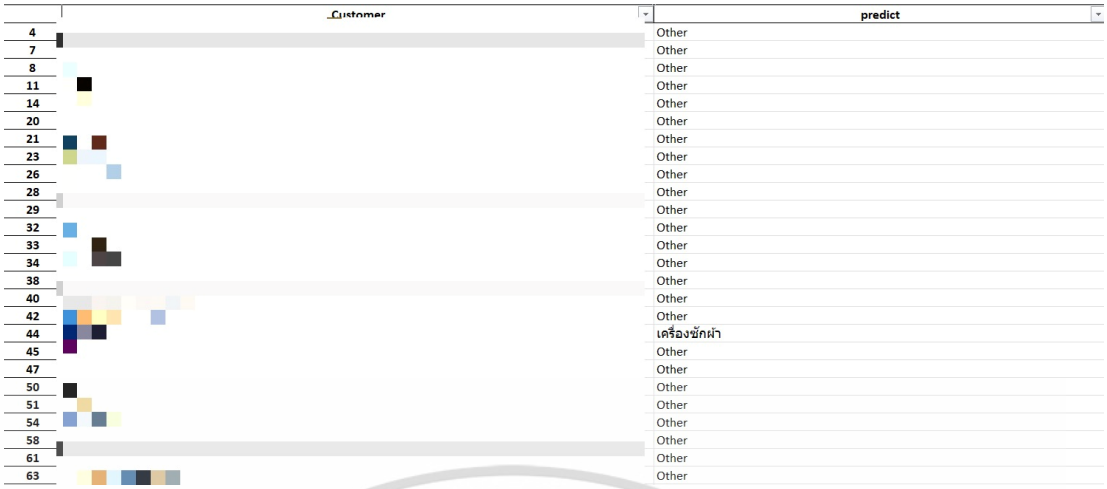
หัวข้อที่เพิ่มเกณฑ์	ค่า Probability ก่อนเพิ่ม	ค่า Probability ที่เพิ่ม	ค่า Probability หลังเพิ่ม
โทรศัพท์	0.1822	0.4	0.5822
เครื่องซักผ้า	0.1522	0.4	0.5522

หลังจากนั้นผู้วิจัยยังนำแบบจำลองและค่าเกณฑ์ที่ได้ตั้งไว้ ไปทดสอบกับข้อมูลทดสอบ พบว่าค่า F-1 ของการกำหนดค่าเกณฑ์ เหมือนกับค่า F-1 ของแบบจำลองแบบป่าสุ่มก่อนที่จะมาทำการหาค่าเกณฑ์ ด้วยเส้นโค้งความแม่นยำและความครบถ้วน ซึ่งถูกต้องเพราะว่าหลังจากการกำหนดค่าเกณฑ์เพิ่มเข้าไปแล้ว ถ้าค่า Probability ของข้อความไหนไม่ถึงเกณฑ์ จะใช้ค่า Probability ที่ดีที่สุดแทนที่จะตัดไปเป็นหัวข้อ “อื่นๆ” แบบการใช้กับข้อความจริง ซึ่งข้อมูลทดสอบไม่มีหัวข้อ “อื่นๆ” ทำให้การทดสอบ F-1 เหมือนกับก่อนที่จะนำไปหาค่าเกณฑ์ ดังตารางที่ 4.7

ตารางที่ 4.7 เปรียบเทียบค่า F-1 ของแบบจำลองป่าสุ่มและค่า F-1 ของแบบจำลองแบบป่าสุ่มที่เพิ่มเกณฑ์แล้วในการทดสอบกับข้อมูลทดสอบที่ไม่มีหัวข้อ “อื่นๆ”

หัวข้อ	ค่า F-1 ของแบบจำลองป่าสุ่ม	ค่า F-1 ของแบบจำลองแบบป่าสุ่มที่เพิ่มเกณฑ์แล้ว
หนังสือ	0.95	0.95
เครื่องบิน	0.81	0.81
รถยนต์	0.76	0.76
โทรศัพท์	0.96	0.96
เก้าอี้	0.97	0.97
ตู้กตา	1.00	1.00
พัดลม	0.80	0.80
ตู้เย็น	0.75	0.75
คอมพิวเตอร์	1.00	1.00
เครื่องซักผ้า	1.00	1.00
ค่าเฉลี่ย	0.91	0.91

หลังจากเพิ่มเกณฑ์และนำไปทดสอบกับข้อมูลทดสอบเรียบร้อยแล้ว ผู้วิจัยได้นำโมเดลไปลองใช้ใหม่ผลปรากฏว่าหัวข้อ “โทรศัพท์”สามารถใช้งานจริงได้ดีดังรูปที่ 4.10 ส่วนหัวข้อ “เครื่องซักผ้า” ยังมีบางข้อความที่ไม่เกี่ยวอยู่บ้างแต่อยู่ในเกณฑ์ที่ยอมรับได้



	Customer	predict
4		Other
7		Other
8		Other
11		Other
14		Other
20		Other
21		Other
23		Other
26		Other
28		Other
29		Other
32		Other
33		Other
34		Other
38		Other
40		Other
42		Other
44		เครื่องซักผ้า
45		Other
47		Other
50		Other
51		Other
54		Other
58		Other
61		Other
63		Other

รูปที่ 4.10 การใช้งานจริงหัวข้อ “โทรศัพท์” หลังจากเพิ่มเกณฑ์

เมื่อนำรูปที่ 4.10 ไปเปรียบเทียบกับรูปที่ 4.8 จะเห็นว่ารูปที่ 4.8 มีหัวข้อ “โทรศัพท์” อยู่เยอะมากแต่ไม่มีสักข้อความที่เป็นหัวข้อ “โทรศัพท์”ที่แท้จริงเลย แต่รูปที่ 4.10 จะเห็นว่าไม่มีหัวข้อ “โทรศัพท์”อยู่แล้ว และ แบบจำลองสามารถทำนายหัวข้อโทรศัพท์ออกมาได้ดี

4.4 อภิปรายผล

จากการวิจัยโดยการนำเข้าชุดข้อมูลแบบจำลองทั้ง 3 แบบพบว่าแบบจำลองมีความแม่นยำสูง โดยแบบจำลองที่ดีที่สุดคือ แบบจำลอง Random Forest (RF) มีประสิทธิภาพดีที่สุดโดยค่าความแม่นยำอยู่ที่ (Precision) 91% ค่าความครบถ้วนอยู่ที่ (Recall) 91% ค่าประสิทธิภาพโดยรวมอยู่ที่ (F1-score) 91% และค่า Accuracy อยู่ที่ 91% ซึ่งถือว่ามีค่าที่สูง หลังจากนั้นเพื่อที่จะนำไปใช้งานจริง โดยที่ข้อมูลที่นำแบบจำลองไปใช้อาจจะมีข้อความที่ไม่เข้าเกณฑ์เช่น “กินข้าวหรือยัง” “วันนี้ฝนตกไหม” ผู้วิจัยจึงนำแบบจำลองแบบ Random Forest (RF) มาทำการกำหนดเกณฑ์ (Threshold) และทำการเพิ่มเกณฑ์ เนื่องจากว่ามีบางคลาสที่ใช้งานจริงมีประสิทธิภาพน้อย จึงเพิ่มเกณฑ์ (Threshold) เข้าไป 0.4 จากนั้นจึงนำแบบจำลองไปใช้กับข้อมูลจริง ซึ่งถือว่ามีความแม่นยำที่สูง

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

งานวิจัยนี้ได้นำข้อมูลหัวข้อ “อื่นๆ” ของโมเดลบริษัทประกันแห่งหนึ่ง มาค้นคว้าว่าควรที่จะมีหัวข้อใหม่ๆอะไรบ้าง เพื่อที่บริษัทจะสามารถนำไปเพิ่มหัวข้อให้บอทสามารถตอบคำถามได้มากขึ้นกว่าเดิม โดยผู้วิจัยได้ใช้แบบจำลองทั้ง 3 แบบคือ Random Forest (RF) Support Vector Machine (SVM) และ Support Vector Machine (SVM) แบบKernel และหาค่าเกณฑ์เพื่อที่จะสามารถใช้กับข้อมูลจริง จึงสามารถสรุปผลการดำเนินงานและข้อเสนอแนะดังนี้

5.1 สรุปผลการวิจัย

การใช้แบบจำลองทั้ง 3 แบบเมื่อดูประสิทธิภาพพบว่าค่าค่อนข้างสูงดังตารางที่ 5.1

ตารางที่ 5.1 ประสิทธิภาพของแต่ละแบบจำลอง

แบบจำลองที่ใช้ในการทดสอบ	Precision	Recall	F1-Score	Accuracy
Random Forest(RF)	0.91	0.91	0.91	0.91
SVM	0.87	0.86	0.86	0.86
SVM แบบ Kernel	0.89	0.88	0.88	0.88

จากตารางที่ 5.1 พบว่าแบบจำลองทุกแบบจำลองมีค่าที่ดีแต่แบบจำลองป่าสุ่ม (Random Forest: RF) มีความโดดเด่นออกมาสูงซึ่งมีค่าความแม่นยำ (Precision) อยู่ที่ 91% ค่าความครบถ้วนอยู่ที่ (Recall) 91% ค่าประสิทธิภาพโดยรวม (F-1) อยู่ที่ 91% และค่าความแม่นยำแบบรวมโดยรวมทั้งโมเดล (Accuracy) อยู่ที่ 91% ทำให้ผู้วิจัยเลือกใช้แบบจำลอง Random Forest (RF) มาทำการกำหนดค่าเกณฑ์ (Threshold) ซึ่งหลังจากที่กำหนดค่าเกณฑ์แล้ว ผู้วิจัยได้นำแบบจำลองไปใช้กับข้อมูลจริง พบว่ามีปัญหาอยู่ 2 หัวข้อคือหัวข้อ “โทรศัพท์” และหัวข้อ “เครื่องซักผ้า”ผู้วิจัยจึงได้ทำการเพิ่มค่าเกณฑ์เข้าไปในหัวข้อ “โทรศัพท์” และหัวข้อ “เครื่องซักผ้า”ผลปรากฏว่าแบบจำลองสามารถใช้งานได้จริงโดยมีประสิทธิภาพที่ดี

5.2 ข้อเสนอแนะ

5.2.1 หากถ้าจะนำไปใช้ต่อ บริษัทสามารถหาข้อมูลทั้ง 10 หัวข้อเพิ่มเติม เพื่อนำมาเพิ่มประสิทธิภาพแบบจำลองได้ โดยการเพิ่มชุดข้อมูลที่เกี่ยวข้องเข้าไปเพิ่มจะทำให้แบบจำลองมีความสามารถที่จะรู้จักกับข้อมูลที่มากขึ้น ทำให้แบบจำลองมีความแม่นยำสูงขึ้นได้

5.2.2 เนื่องจากแบบจำลองใหม่ๆเกิดขึ้นตลอดเวลา ดังนั้นผู้ที่จะนำแนวคิดนี้ไปปรับใช้ควรที่จะค้นคว้าเพิ่มเติมอยู่เสมอ ในอนาคตอาจจะมีแบบจำลองที่ทำงานกับข้อมูลที่เป็นข้อความได้ดีเกิดขึ้นมาใหม่ เพื่อความแม่นยำที่มากขึ้น

เอกสารอ้างอิง

ชิตพงษ์ กิตตินราดร.2563. **Support Vector Machines.**[Online].เข้าถึงได้จาก. https://guopai.github.io/ml-blog08.html

ปฐวี ปราการกมานันท์. 2564. **การเพิ่มข้อมูลสำหรับระบบประมวณภาษาธรรมชาติภาษาไทยโดยใช้การแบ่งโทเค็นที่แตกต่างกัน.** วิทยานิพนธ์. หลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์.

ธนัท จรณะสมบูรณ์. 2561. **การทำนายการซื้อซ้ำของผู้ซื้อโดยใช้เทคนิคการเรียนรู้ของเครื่องจักร.** ปริญญานิพนธ์. หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ. มหาวิทยาลัยศรีนครินทรวิโรฒ

พงศ์ทวัส ฮั่นวัฒนวงศ์. 2565. **การพัฒนาเวิร์กโฟลว์สำหรับตัวแบบต้นไม้จำแนกประเภทที่ดีที่สุด**. วิทยานิพนธ์. หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาสถิติ.จุฬาลงกรณ์มหาวิทยาลัย.

สุวพร หอมจันทร์ดี. 2565. **ป่าสุ่มของต้นไม้ตัดสินใจควบแน่นฝ่ายข้างน้อยสำหรับปัญหาคลาสไม่สมดุล.** วิทยานิพนธ์. หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาคณิตศาสตร์ประยุกต์และวิทยาการคณนา.จุฬาลงกรณ์มหาวิทยาลัย

สถาบันนวัตกรรมและธรรมมาภิบาลข้อมูล. 2565. **เข้าใจใน 5 นาที! Classification Model คืออะไร.**[Online].เข้าถึงได้จาก. DIGI (data.go.th)

Amandp13.2024. **Random Forest Classifier using Scikit-learn**. [Online]. https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/

Kasidis, S. 2562. **อธิบาย K-Fold Cross Validation พร้อมโค้ดตัวอย่างใน R**. [online].เข้าถึงได้จาก.https://datarockie.com/blog/k-fold-cross-validation/comment-page-1/

Matana, W. 2567. **Machine Learning คืออะไร?.** [Online]. เข้าถึงได้จาก. https://www.aware.co.th/machine-learning-คืออะไร/

Mohajon, J. 2020. **Confusion Matrix for Your Multi-Class Machine Learning Model**. [Online]. https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826.

Natdanai, J. 2562. **Supervised Learning (การเรียนรู้แบบมีผู้สอน) คืออะไร.**[Online].เข้าถึงได้จาก.Supervised Learning (การเรียนรู้แบบมีผู้สอน) คืออะไร - GlurGeek.Com.

PradyaSin. 2562. **Support Vector Machines (SVM).** [Online].เข้าถึงได้จาก.https://medium.com/@pradyasin/support-vector-machines-SVM-943f9a732a69.

เอกสารอ้างอิง (ต่อ)

Scikit-learn developers. **Precision-Recall.** [Online].เข้าถึงได้จาก. https://scikit-learn.org/stable/auto_examples/model_selection/plot_Precision_Recall.html.

Soni, K. 2016. **What is Stratify in train_test_split? With example.**[online]. https://dragonforest.in/stratify/#google_vignette

Vithan, M. 2561. **Machine Learning คืออะไร?**. [Online]. เข้าถึงได้จาก .Machine Learning คืออะไร?. เคยสงสัยกันบ้างรึเปล่า ว่า | by Vithan Minaphinant | investic | Medium.

ภาคผนวก

คำสั่งโปรแกรม python ที่ใช้ในการวิจัย

การเตรียมข้อมูล

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
Precision_Recall_fscore_support
from sklearn.model_selection import train_test_split, StratifiedKFold, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelEncoder
import numpy as np
import pythainlp
import re
import string
import nltk
from pythainlp.corpus import thai_stopwords, thai_words
from pythainlp import word_tokenize, Tokenizer
from pythainlp.corpus.common import thai_words
from pythainlp.util import Trie
from sklearn.SVM import LinearSVC
!pip install pythainlp
df = pd.read_excel("/content/drive/MyDrive/File for new intent
project/new_intent_threshole.xlsx")
df = df.drop_duplicates()
import pandas as pd
import re
def replace_words(text):
  Args:
      text: The text to be processed.
  Returns:
      The text with the specified words replaced.
  """
  replacements = {
```

```
    “รถใหญ่”: “รถยนต์”,
    “ที่นั่ง”: “เก้าอี้”
  }
  pattern = re.compile(r'\b(?<!omne)'+'|'.join(replacements.keys())+r'\b',
re.IGNORECASE)
  return pattern.sub(lambda m: replacements[m.group(0)], text, count=1)
df["keyword"] = df["keyword"].astype(str)
df["keyword"] = df["keyword"].apply(replace_words)
words = new_words.union(thai_words())
custom_dictionary_trie = Trie(words)
def clean_text(text):
    # Remove "ค่ะ" and "ครับ" using regular expressions
    text = re.sub(r'\b(?:ค่ะ|ครับ|คะ|ครับผม|เจ้าค่ะ|จ้ะ|จ้า|นะคะ)\b', '', text)
    # Remove other patterns as before
    regex = re.compile(pattern=r"delay{[0-9]+.[0-9]+}|delay{[\d]}|delay{[a-z]+}|\\[u][0-
9]+[a-zA-Z]|\\[xa][0-9]")
    text = re.sub(regex, "", text)
    text = re.sub(r'[^\u0E00-\u0E7Fa-zA-Z0-9\s]', '', text)  # remove special characters
and emojis
    text = re.sub(r'(?<=\d)\s+(?=\d)', '', text)
    text = re.sub(r'\d+', '', text)
    text = re.sub(r'\s+', ' ', text)  # remove extra whitespace
    text = re.sub(r':[a-z_]+:', '', text)  # remove all emoji
    text = text.strip()  # remove leading and trailing whitespace
    return text
# Preprocess the text
def preprocess_text(text):
   ## remove punctuation
   filtered_text = "".join(u for u in text if u not in ("?", ".", ";", ":", "!", '"', "ๆ", "ฯ"))
    filtered_text = word_tokenize(filtered_text, custom_dict=custom_dictionary_trie)
    # filtered_text = att_token.tokenize(filtered_text)
    ## Join Alltexts into a string
    filtered_text = " ".join(word for word in filtered_text)
```

```
    ## Join Text after split text
    filtered_text = " ".join(word for word in filtered_text.split())
    return filtered_text
df["keyword"] = df['keyword'].apply(clean_text)
df["keyword"] = df['keyword'].apply(preprocess_text)
from sklearn.model_selection import train_test_split
## Split data


df['intent'] = df['intent'].astype(str)
X_train, X_test, y_train, y_test  = train_test_split(df['keyword'], df['intent'],
                                                     test_size = 0.2, random_state = 1, shuffle =
True, stratify= df['intent'])
# Get the count of each class in the training set
train_class_counts = y_train.value_counts()
# Get the count of each class in the testing set
test_class_counts = y_test.value_counts()
print("Training Class Counts:")
print(train_class_counts)
print("\nTesting Class Counts:")
print(test_class_counts)
print("\nShape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
```

การรันผลแบบจำลอง

แบบป่าสุ่ม

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
Precision_Recall_fscore_support
from sklearn.model_selection import train_test_split, StratifiedKFold, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelEncoder
import numpy as np
```

```
# Assuming you have X and y defined
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# Use LabelEncoder to convert string labels to integers
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
# Create a pipeline with TF-IDF vectorizer and RandomForestClassifier with class
weights
# Feature engineering (Thai-specific adjustments)
vectorizer = TfidfVectorizer(ngram_range=(1, 3), analyzer='char_wb')  # Include
character n-grams
pipeline_rf = Pipeline([
    ('tfidf', vectorizer),
    ('rf', RandomForestClassifier(class_weight='balanced'))  # Set class_weight to
'balanced'
])
# Define a range of values for hyperparameters (adjust as needed)
param_grid_rf = {
    "rf__n_estimators": [100, 200, 500],
    "rf__max_depth": [None, 10, 20, 50],
    "rf__min_samples_split": [2, 5, 10],
    "rf__min_samples_leaf": [1, 2, 4],
    "tfidf__max_features": [5000, 10000, None]  # Consider feature selection
}
# Use StratifiedKFold for cross-validation
stratified_kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)

# Use GridSearchCV with StratifiedKFold
grid_search_rf = GridSearchCV(pipeline_rf, param_grid_rf, cv=stratified_kfold,
scoring='accuracy')
try:
        grid_search_rf.fit(X_train, y_train_encoded)
```

```
except Exception as e:
    print(f"Error during grid search: {e}")
    # You can print additional information or investigate the error further
    raise  # Raise the exception to see the full traceback
# Get the best hyperparameters
best_params_rf = grid_search_rf.best_params_
best_model_rf = grid_search_rf.best_estimator_
# Predict on the test set
y_pred_rf = best_model_rf.predict(X_test)
# Convert the predicted labels back to original class names
y_pred_rf_original = label_encoder.inverse_transform(y_pred_rf)
# Calculate accuracy
train_accuracy_rf = accuracy_score(y_train_encoded, grid_search_rf.predict(X_train))
test_accuracy_rf = accuracy_score(label_encoder.transform(y_test), y_pred_rf)
print(f"Training Accuracy: {train_accuracy_rf * 100:.2f}%")
print(f"Testing Accuracy: {test_accuracy_rf * 100:.2f}%")
# Calculate Precision, Recall, and F1-score
Precision, Recall, F1_score, _ =
Precision_Recall_fscore_support(label_encoder.transform(y_test), y_pred_rf,
average='weighted')
print(f"Precision: {Precision:.2f}, Recall: {Recall:.2f}, F1-Score: {F1_score:.2f}")
print(f"The best hyperparameters are: {best_params_rf}")
print(f'The best Accuracy of RandomForestClassifier: {grid_search_rf.best_score_}')
# Calculate and print the classification report with string labels
print(classification_report(label_encoder.transform(y_test),
label_encoder.transform(y_pred_rf_original), target_names=label_encoder.classes_))
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
# Calculate confusion matrix
conf_matrix = confusion_matrix(label_encoder.transform(y_test), y_pred_rf)
```

```
# Plot heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
            xticklabels=label_encoder.classes_,
            yticklabels=label_encoder.classes_)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix - RandomForestClassifier')
plt.show()
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
# Calculate confusion matrix
conf_matrix = confusion_matrix(label_encoder.transform(y_test), y_pred_rf)
# Calculate percentages
conf_matrix_percent = conf_matrix.astype('float') / conf_matrix.sum(axis=1)[:,
np.newaxis] * 100
# Plot heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix_percent, annot=True, fmt=".1f", cmap="Blues",
            xticklabels=label_encoder.classes_,
            yticklabels=label_encoder.classes_)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix - RandomForestClassifier (%)')
plt.show()
```

แบบเวกเตอร์ซัพพอตแมชชีน(SVM)

```
# Assuming you have X and y defined
# Split the data into training and testing sets
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# Use LabelEncoder to convert string labels to integers
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
# Create a pipeline with TF-IDF vectorizer and LinearSVC
pipeline_svc = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('svc', LinearSVC())
])
# Define a range of values for hyperparameters (adjust as needed)
param_grid_svc = {'svc__C': [0.1, 1.0, 10.0]}
# Use StratifiedKFold for cross-validation
stratified_kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)
# Use GridSearchCV with StratifiedKFold
grid_search_svc = GridSearchCV(pipeline_svc, param_grid_svc, cv=stratified_kfold,
scoring='accuracy')
try:
    grid_search_svc.fit(X_train, y_train_encoded)
except Exception as e:
    print(f"Error during grid search: {e}")
    # You can print additional information or investigate the error further
    raise  # Raise the exception to see the full traceback
# Get the best hyperparameters
best_params_svc = grid_search_svc.best_params_
best_model_svc = grid_search_svc.best_estimator_

# Predict on the test set
y_pred_svc = best_model_svc.predict(X_test)
```

```
# Convert the predicted labels back to original class names
y_pred_svc_original = label_encoder.inverse_transform(y_pred_svc)
# Calculate accuracy
train_accuracy_svc = accuracy_score(y_train_encoded,
grid_search_svc.predict(X_train))
test_accuracy_svc = accuracy_score(label_encoder.transform(y_test), y_pred_svc)
print(f"Training Accuracy: {train_accuracy_svc * 100:.2f}%")
print(f"Testing Accuracy: {test_accuracy_svc * 100:.2f}%")
# Calculate Precision, Recall, and F1-score
Precision, Recall, F1_score, _ =
Precision_Recall_fscore_support(label_encoder.transform(y_test), y_pred_svc,
average='weighted')
print(f"Precision: {Precision:.2f}, Recall: {Recall:.2f}, F1-Score: {F1_score:.2f}")
print(f"The best hyperparameters are: {best_params_svc}")
print(f'The best Accuracy of LinearSVC: {grid_search_svc.best_score_}')
# Calculate and print the classification report with string labels
print(classification_report(label_encoder.transform(y_test),
label_encoder.transform(y_pred_svc_original), target_names=label_encoder.classes_))
# Calculate confusion matrix for test data
conf_matrix = confusion_matrix(label_encoder.transform(y_test), y_pred_svc)
# Plot heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```


```
# Calculate confusion matrix for test data
conf_matrix = confusion_matrix(label_encoder.transform(y_test), y_pred_svc)
```

```
# Calculate percentages
conf_matrix_percentage = conf_matrix.astype('float') / conf_matrix.sum(axis=1)[:,
np.newaxis] * 100
# Plot heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_percentage, annot=True, fmt='.2f', cmap='Blues',
xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix (Percentage)')
plt.show()
```

แบบเวกเตอร์ซัพพอตแมชชีน(SVM)แบบKernel

```
# Assuming you have X and y defined
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# Use LabelEncoder to convert string labels to integers
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
# Create a pipeline with TF-IDF vectorizer and SVM with RBF Kernel
pipeline_SVM = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('SVM', SVC(Kernel='rbf'))
])
# Define a range of values for hyperparameters (adjust as needed)
param_grid_SVM = {'SVM__C': [0.1, 1.0, 10.0],
                  'SVM__gamma': [0.01, 0.1, 1.0]}
# Use StratifiedKFold for cross-validation
stratified_kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)

# Use GridSearchCV with StratifiedKFold
```

```
grid_search_SVM = GridSearchCV(pipeline_SVM, param_grid_SVM, cv=stratified_kfold,
scoring='accuracy')
try:
    grid_search_SVM.fit(X_train, y_train_encoded)
except Exception as e:
    print(f"Error during grid search: {e}")
    # You can print additional information or investigate the error further
    raise  # Raise the exception to see the full traceback
# Get the best hyperparameters
best_params_SVM = grid_search_SVM.best_params_
best_model_SVM = grid_search_SVM.best_estimator_
# Predict on the test set
y_pred_SVM = best_model_SVM.predict(X_test)
# Convert the predicted labels back to original class names
y_pred_SVM_original = label_encoder.inverse_transform(y_pred_SVM)
# Calculate accuracy
train_accuracy_SVM = accuracy_score(y_train_encoded,
grid_search_SVM.predict(X_train))
test_accuracy_SVM = accuracy_score(label_encoder.transform(y_test), y_pred_SVM)
print(f"Training Accuracy: {train_accuracy_SVM * 100:.2f}%")
print(f"Testing Accuracy: {test_accuracy_SVM * 100:.2f}%")
# Calculate Precision, Recall, and F1-score
Precision, Recall, F1_score, _ =
Precision_Recall_fscore_support(label_encoder.transform(y_test), y_pred_SVM,
average='weighted')
print(f"Precision: {Precision:.2f}, Recall: {Recall:.2f}, F1-Score: {F1_score:.2f}")
print(f"The best hyperparameters are: {best_params_SVM}")
print(f'The best Accuracy of SVM with RBF Kernel: {grid_search_SVM.best_score_}')
```



```
# Calculate and print the classification report with string labels
```

```
print(classification_report(label_encoder.transform(y_test),
label_encoder.transform(y_pred_SVM_original),
target_names=label_encoder.classes_))
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
# Calculate confusion matrix for test data
conf_matrix = confusion_matrix(label_encoder.transform(y_test), y_pred_SVM)
# Plot heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
# Calculate confusion matrix for test data
conf_matrix = confusion_matrix(label_encoder.transform(y_test), y_pred_SVM)
# Calculate percentages
conf_matrix_percentage = conf_matrix.astype('float') / conf_matrix.sum(axis=1)[:,
np.newaxis] * 100
# Plot heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_percentage, annot=True, fmt='.2f', cmap='Blues',
xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix (Percentage)')
plt.show()
```

หาค่าเกณฑ์

```
from sklearn.metrics import F1_score
# Initialize variables to store optimal thresholds
optimal_thresholds = {}
for class_idx in range(len(unique_classes)):
    class_name = label_encoder.classes_[class_idx]
    # Binary classification approach for each class
    y_test_class = (label_encoder.transform(y_test) == class_idx).astype(int)
    y_probs_class = y_probs_rf[:, class_idx]
    # Initialize variables for optimal threshold search
    best_threshold = None
    best_F1 = 0
    # Loop through different thresholds
    for threshold in np.linspace(0, 1, 1000):  # Adjust the range as needed
        y_pred_class = (y_probs_class > threshold).astype(int)
        F1 = F1_score(y_test_class, y_pred_class)
        if F1 > best_F1:
            best_F1 = F1
            best_threshold = threshold
    optimal_thresholds[class_name] = best_threshold
    # Print F1-score for each class
    print(f"F1-score for {class_name}: {best_F1:.4f}")
# Print optimal thresholds
for class_name, threshold in optimal_thresholds.items():
    print(f"Optimal Threshold for {class_name}: {threshold:.4f}")
```

การใช้โมเดล

```
def classify_with_thresholds(y_probs, thresholds, label_encoder):
    y_pred = []
    for i in range(len(y_probs)):
        max_prob = max(y_probs[i])
        max_class = np.argmax(y_probs[i])
        # For "Payment suggestion" class, use a higher threshold
        if label_encoder.classes_[max_class] == "โทรศัพท์":
            threshold = thresholds[label_encoder.classes_[max_class]] + 0.4  # Increase
threshold by 0.1
        elif label_encoder.classes_[max_class] == "เครื่องซักผ้า":
            threshold = thresholds[label_encoder.classes_[max_class]] + 0.4  # Increase
threshold by 0.2
        else:
            threshold = thresholds[label_encoder.classes_[max_class]]
        if max_prob >= threshold:
            y_pred.append(label_encoder.classes_[max_class])
        else:
            y_pred.append("Other")
    return y_pred
y_probs_new = best_model_rf.predict_proba(df_test["cus"])
df_test["predict"] = classify_with_thresholds(y_probs_new, optimal_thresholds,
label_encoder)
```

งานทะเบียนคณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
คำรับรองเล่มโครงงานพิเศษ/ปัญหาพิเศษ/สหกิจศึกษา

วันที่ 16 เดือน พฤษภาคม พ.ศ 2566

ข้าพเจ้า นาย ธนพัฒน์ สอนโส รหัสประจำตัว 63050627
นักศึกษาหลักสูตรวิทยาศาสตร์บัณฑิต สาขาวิชา สถิติประยุกต์ ภาควิชา สถิติ ขอรับรองว่าโครงงานสหกิจศึกษา เรื่อง

ชื่อภาษาไทย การจัดหมวดหมู่ประเภทข้อความเพื่อเพิ่มหัวข้อใหม่โดยใช้การเรียนรู้ของเครื่อง
ชื่อภาษาอังกฤษ TEXT CLASSIFICATION FOR NEW INTENT USING MACHINE LEARNING
ปีการศึกษา 2566

เป็นผลงานวิจัยที่มิได้คัดลอกหรือละเมิดลิขสิทธิ์ของผู้อื่นและได้ผ่านการตรวจสอบความซ้ำซ้อนเรียบร้อยแล้ว และได้แนบเอกสารการตรวจสอบการลอกเลียนงานวรรณกรรมที่ตรวจสอบจากเล่มสหกิจศึกษาฉบับสมบูรณ์แล้ว
โปรแกรมอักขราวิสุทธิ์ 0.49%

ลงชื่อ.....
(นายธนพัฒน์ สอนโส)
นักศึกษา

ข้าพเจ้า รศ.ดร. วลัยลักษณ์ อัตธีรวงศ์ อาจารย์ที่ปรึกษาโครงงานสหกิจศึกษา ได้ตรวจสอบโครงงานพิเศษ/ปัญหาพิเศษ/สหกิจศึกษาของนักศึกษาข้างต้น แล้ว ขอรับรองว่าเป็นผลงานวิจัยของนักศึกษาจริงและมีเนื้อหาสมบูรณ์ จึงลงชื่อไว้เป็นหลักฐาน

ลงชื่อ.....
อาจารย์ที่ปรึกษา

Submission Information

ID	SUBMISSION DATE	SUBMITTED BY	ORGANIZATION	FILENAME	STATUS	SIMILARITY INDEX
3741995	May 16, 2024 at 05:43 AM	63050627@kmitl.ac.th	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง	เล่มสหกิจ 63050627 แก้ไขเพิ่มเติม.pdf	Completed	0.49 %

Match Overview

Show 10 entries Search:

NO.	TITLE	AUTHOR(S)	SOURCE	SIMILARITY INDEX
1	The prototype of detecting risky behavior of drowsiness from video	ฐิติอาภากุล, ไพศาล	วารสารวิทยาศาสตร์และเทคโนโลยีแห่งรัตนโกสินทร์	0.49 %

NO.	TITLE	AUTHOR(S)	SOURCE	SIMILARITY INDEX

Showing 1 to 1 of 1 entries First Previous 1 Next Last