การวิเคราะห์การบุกรุกบนระบบเครือข่ายโดยใช้เทคนิคการเรียนรู้ของเครื่อง

ANALYSIS OF NETWORK INTRUSION DETECTION SYSTEM USING MACHINE LEARNING TECHNIQUES

อภิชิต ปัตถาวะโร

บัณฑิตวิทยาลัย มหาวิทยาลัยศรีนครินทรวิโรฒ

2561

การวิเคราะห์การบุกรุกบนระบบเครือข่ายโดยใช้เทคนิคการเรียนรู้ของเครื่อง

อภิชิต ปัตถาวะโร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ
ปีการศึกษา 2561

ANALYSIS OF NETWORK INTRUSION DETECTION SYSTEM USING MACHINE LEARNING TECHNIQUES

APICHIT Pattawaro

A Thesis Submitted in partial Fulfillment of Requirements

for MASTER OF SCIENCE (Information Technology)

Faculty of Science Srinakharinwirot University

2018

ปริญญานิพนธ์

เรื่อง

การวิเคราะห์การบุกรุกบนระบบเครือข่ายโดยใช้เทคนิคการเรียนรู้ของเครื่อง

ของ

อภิชิต ปัตถาวะโร

ได้รับอนุมัติจากบัณฑิตวิทยาลัยให้นับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ของมหาวิทยาลัยศรีนครินทรวิโรฒ

..... คณบดีบัณฑิตวิทยาลัย

(ผู้ช่วยศาสตราจารย์ นายแพทย์ฉัตรชัย เอกปัญญาสกุล)

.....

คณะกรรมการสอบปากเปล่าปริญญานิพนธ์

..... ที่ปรึกษาหลัก ..... ประธาน

(ผู้ช่วยศาสตราจารย์ ดร.จันตรี ผลประเสริฐ) (ดร.วีรยุทธ เจริญเรืองกิจ)

..... กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ประจักษ์ เฉิดโฉม)

ชื่อเรื่อง	การวิเคราะห์การบุกรุกบนระบบเครือข่ายโดยใช้เทคนิคการเรียนรู้ของเครื่อง
ผู้วิจัย	อภิชิต ปัตถาวะโร
ปริญญา	วิทยาศาสตรมหาบัณฑิต
ปีการศึกษา	2561
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร. จันตรี ผลประเสริฐ

ในงานวิจัยนี้ นำเสนอเทคนิคการตรวจจับรูปแบบการบุกรุกบนเครือข่ายที่ผิดปกติ โดยผสมผสานเทคนิคของการเลือกคุณลักษณะ (Feature Selection) การแบ่งกลุ่มรูปแบบของข้อมูล (K-Means clustering) และการจำแนกรูปแบบของข้อมูลที่ผิดปกติโดยใช้เทคนิคการจำแนกข้อมูลแบบต้นไม้ (Tree-based models classification) ผู้วิจัยได้ทดสอบประสิทธิภาพของโมเดลโดยใช้ชุดข้อมูล NSL-KDD สำหรับทดสอบ (KDDTest+) ซึ่งหลักการสร้างโมเดลจะใช้วิธีการเลือกคุณลักษณะตามค่าอัตราเฉลี่ยของคุณลักษณะ (Attribute Ratio) เพื่อลดจำนวนคุณลักษณะที่ไม่จำเป็นออก หลังจากนั้นจึงใช้การแบ่งกลุ่มรูปแบบของข้อมูล (K-Means clustering) เพื่อให้รูปของข้อมูลที่เหมือนกันอยู่ในกลุ่มเดียวกัน โดยงานวิจัยนี้ใช้การแบ่งกลุ่มเพียง 2 กลุ่มเท่านั้น จากนั้นนำข้อมูลที่ได้จากการแบ่งกลุ่มมาเข้าโมเดลโดยใช้การทำไฮเปอร์พารามิเตอร์จูนนิ่ง เพื่อหาค่าคะแนนสูงสุดของแต่ละพารามิเตอร์ในแต่ละกลุ่ม ซึ่งจากการทดสอบประสิทธิภาพของโมเดลกับชุดข้อมูล KDDTest+ พบว่าโมเดลที่นำเสนอมีค่าความแม่นยำเท่ากับ 84.41% โดยมีอัตราการตรวจจับเท่ากับ 86.36% และอัตราการเตือนภัยผิดพลาดเท่ากับ 18.20% โดยประสิทธิภาพของโมเดลที่ผู้วิจัยได้นำเสนอมีค่าความแม่นยำดีกว่าการใช้เครือข่ายประสาทเทียมแบบโครงข่ายประสาทเทียม (RNN) นอกจากนี้โมเดลของเราใช้คุณลักษณะเพียง 77 คุณลักษณะเท่านั้นจาก 122 คุณลักษณะ หรือคิดเป็น 63.11% เมื่อเปรียบเทียบกับชุดข้อมูล NSL-KDD แบบเต็ม พบว่าโมเดลให้ค่าความแม่นยำใกล้เคียงกันแต่ใช้คุณลักษณะน้อยกว่า

คำสำคัญ : ไฮบริดโมเดล การแบ่งกลุ่ม การจำแนก ชุดข้อมูล NSL-KDD

Title	ANALYSIS OF NETWORK INTRUSION DETECTION SYSTEM USING MACHINE LEARNING TECHNIQUES
Author	APICHIT Pattawaro
Degree	MASTER OF SCIENCE
Academic Year	2018
Thesis Advisor	Chantri Polprasert

This paper proposes an anomaly-based network intrusion detection system based on a combination of feature selection, K-Means clustering and tree-based models classification. The performance of the proposed system was tested with a NSL-KDD dataset using a KDDTest^{+} dataset. A feature selection method based on attribute ratio (AR) was applied to construct a reduced feature subset of NSL-KDD dataset. After the application of K-Means clustering and hyperparameter tuning of each classification model corresponding with each cluster was implemented. There were only two clusters, the proposed model obtained accuracy equal to 84.41% with a detection rate equal to 86.36% and a false alarm rate equal to 18.20% for the KDDTest^{+} dataset. The performance of the proposed model outperformed those obtained using the recurrent neural network (RNN) based deep neural network. In addition, due to feature selection, the proposed model employed only seventy-seven out of one hundred and twenty-two features (63.11%) to achieve this level of performance comparable to those using a full number of features to train the model.

Keyword : Hybrid Clustering, Classification NSL-KDD network security

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ ด้วยความกรุณาจากคณะกรรมการผู้ควบคุมปริญญานิพนธ์ทุกท่านที่ช่วยให้ข้อเสนอแนะที่เป็นประโยชน์ต่องานวิจัยนี้และผู้วิจัยขอกราบขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.จันตรี ผลประเสริฐ ที่กรุณาเป็นอาจารย์ที่ปรึกษาและให้ความช่วยเหลือ คำแนะนำและแนวทางที่เป็นประโยชน์ต่อการทำปริญญานิพนธ์ที่ดีเสมอมา ทำให้ปริญญานิพนธ์ฉบับนี้มีความสมบูรณ์ยิ่งขึ้น ขอกราบขอบพระคุณอาจารย์และกรรมการบริหารหลักสูตรสาขาวิชาเทคโนโลยีสารสนเทศ คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒที่ได้กรุณาให้ความรู้ในด้านต่างๆให้แก่ผู้วิจัย จึงเป็นที่มาของปริญญานิพนธ์ฉบับนี้

ขอขอบพระคุณทุนสนับสนุนการเข้าร่วมประชุมและนำเสนอผลงานทางวิชาการ จากบัณฑิตวิทยาลัย มหาวิทยาลัยศรีนครินทรวิโรฒ ประจำปีงบประมาณ 2562 ขอขอบคุณเพื่อนในรุ่น 1 และ 2 ทุกคน ที่ให้ความช่วยเหลือ ตลอดจนคำแนะนำที่เป็นประโยชน์ทั้งการเรียนและการทำวิจัยในครั้งนี้ ท้ายที่สุดขอกราบขอบพระคุณบิดา มารดา ที่เป็นกำลังใจสำคัญและให้โอกาสทางการศึกษาอันมีค่ายิ่งต่อผู้วิจัย

อภิชิต ปัตถาวะโร

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูปภาพ

หน้า

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหาที่ทำการวิจัย

ปัจจุบันระบบ Network มีบทบาทสำคัญในการดำเนินชีวิตของมนุษย์มากขึ้น ทั้งการทำงาน สังคมและธุรกิจ ซึ่งโครงข่าย Network ในปัจจุบันยังคงไม่มีความปลอดภัย และมีรูปแบบของการบุกรุกหรือโจมตีที่หลากหลาย จากบุคคลที่ไม่พึงประสงค์ ส่งผลโดยตรงกับผู้ใช้งานทั่วไป ทั้งทางด้านความเป็นส่วนตัว รวมถึงความปลอดภัยต่างๆ เช่นด้าน สังคม การทำธุรกรรม การโจมตีที่ทำให้ธุรกิจต้องหยุดชะงัก จึงต้องมีระบบรักษาความปลอดภัยทางไซเบอร์ (Cyber Security) เกิดขึ้น เพื่อตรวจตราและป้องกันภัยคุกคามทางไซเบอร์ที่มีเกิดขึ้นในปัจจุบัน

ระบบรักษาความปลอดภัยทางไซเบอร์ (Cyber Security) เป็นเทคโนโลยีและกระบวนการที่ถูกออกแบบมาเพื่อป้องกันคอมพิวเตอร์, ระบบโครงข่าย, โปรแกรมและข้อมูลจากการโจมตี การเข้าถึงเปลี่ยนแปลงหรือการทำลายโดยไม่ได้รับอนุญาต ระบบรักษาความปลอดภัยทางไซเบอร์ประกอบไปด้วยการรักษาความปลอดภัยทางเครือข่าย เช่นไฟร์วอลล์ และโฮสต์ (คอมพิวเตอร์) เช่น ซอฟต์แวร์ป้องกันไวรัส และยังมีระบบตรวจจับการบุกรุก (IDS) ซึ่งอยู่ในไฟร์วอลล์ IPS (intrusion prevention system)และซอฟต์แวร์ป้องกันไวรัสรุ่นใหม่ได้นำมาใช้ในการตรวจจับการบุกรุก

IDS (intrusion detection system) คือระบบตรวจจับการบุกรุก ช่วยในการค้นหา ตรวจสอบและระบุการทำสำเนาหรือการเปลี่ยนแปลงและการทำลายระบบสารสนเทศโดยไม่ได้รับอนุญาต การละเมิดด้านความปลอดภัยทางไซเบอร์ทั้งจากการบุกรุกภายนอก (การโจมตีจากภายนอกองค์กร) และการบุกรุกภายใน (การโจมตีจากภายในองค์กร)

หลักการวิเคราะห์ของ IDS ที่ใช้อยู่ในปัจจุบันมีอยู่ 3 ประเภทคือ

1 misuse-based (ในบางครั้งถูกเรียกว่า signature based)

2 anomaly-based

3 ผสม misuse-based กับ anomaly-based

เทคนิค misuse-based ถูกออกแบบมาเพื่อตรวจจับการโจมตีโดยใช้ลายเซ็น (signatures) มันมีประสิทธิภาพในการตรวจจับชนิดของการโจมตีที่เคยเกิดขึ้นมาแล้ว เพื่อให้มีความผิดพลาดที่น้อยที่สุด โดยเชื่อมโยงฐานข้อมูลของการโจมตีที่เคยเกิดขึ้นในอดีต ดังนั้นระบบนี้จึงมีประสิทธิการในการตรวจจับการโจมตีที่เคยเกิดขึ้นมาแล้ว แต่เทคนิค Misuse-based มีข้อเสียคือไม่สามารถใช้ตรวจจับการโจมตีแบบใหม่ๆ (zero-day) ได้

เทคนิค anomaly-based เป็นเทคนิคที่ใช้ตรวจจับพฤติกรรมที่ผิดปกติของรูปแบบข้อมูลที่อยู่ในระบบโครงข่ายและทำหน้าที่ระบุความผิดปกติของข้อมูลนั้นด้วย มันจึงมีความน่าสนใจเพราะสามารถตรวจจับการโจมตีแบบ zero-day ได้ ซึ่งพฤติกรรมของรูปแบบข้อมูลที่ผิดปกติจะไม่สามารถระบุได้ว่าเป็นการโจมตีและเป็นรูปแบบที่ไม่เคยพบมาก่อน ซึ่งอาจเป็นอันตรายหรือไม่เป็นอันตรายก็ได้ ดังนั้นการตรวจจับความผิดปกติจึงมีความความสำคัญมากต่อแอพพลิเคชั่นหรือระบบโครงข่าย ซึ่งยากต่อการจะตรวจหาช่องโหว่นี้ นอกจากนี้ ข้อมูลที่ได้จากเทคนิค anomaly-based ยังสามารถนำมาใช้เพื่อกำหนดลายเซ็นสำหรับการตัวตรวจจับแบบ misuse แต่ข้อเสียหลักของเทคนิค anomaly-based คือการแจ้งเตือนมีความผิดพลาดสูง False Alarm Rates (FARs) เนื่องจากพฤติกรรมของข้อมูลเป็นพฤติกรรมที่ระบบไม่ False Positive Rate เคยรู้จักมาก่อน ซึ่งอาจถูกมองว่าเป็นความผิดปกติ ในทางปฏิบัติจะมีระบบผสม เพื่อจะแก้ปัญหาทั้ง ทั้ง zero-day และ False Alarm Rates เป็นเทคนิคการรวม misuse-based และ anomaly-based เข้าด้วยกัน ซึ่งถูกเรียกว่าแบบผสม เพื่อเพิ่มอัตราการตรวจจับและลดอัตราความผิดพลาด (FARs) สำหรับการโจมตีที่ไม่รู้จัก

สำหรับ anomaly-based ได้มีเทคนิคการใช้การเรียนรู้ของเครื่องเช่น Support vector machine (SVM) , K-Nearest Neighbors (KNN) มาทำการตรวจจับรูปแบบข้อมูลที่ผิดปกติ แต่พบว่าประสิทธิภาพในการตรวจจับยังไม่ดีเท่าที่ควร ในการทบทวนงานวิจัยยังไม่พบวิธีการตรวจจับรูปแบบข้อมูลที่ผิดปกติ ที่มีความแม่นยำเพียงพอ ส่วนใหญ่จะใช้วิธีแบบไฮบริด ซึ่งเป็นการรวมอัลกอลิทึมมากกว่า1แบบ เช่น Clustering กับ Classification ทำงานรวมกัน ซึ่งวิธีนี้ทำให้ประสิทธิภาพของการตรวจจับการโจมตีในรูปแบบของ Accuracy, Recall, Precision, False Alarm Rates ดีขึ้น

ในงานวิจัยนี้เราจะมุ่งเน้นศึกษาการเรียนรู้ของเครื่องแบบ Hybrid เนื่องจากการใช้โมเดลแบบ classifier อย่างเดียว ยังคงมีความผิดพลาดในการตรวจจับข้อมูลที่เป็น anomaly-based เพื่อปรับปรุงเทคนิคในการตรวจจับข้อมูลที่เป็น anomaly-based ดังกล่าว ให้มีความแม่นยำมากขึ้น เราจึงนำเสนอเทคนิคการเรียนรู้ของเครื่องแบบไฮบริด โดยใช้ Clustering และ Classification มาทำงานร่วมกัน เพื่อวิเคราะห์รูปแบบชุดข้อมูลและทำนายว่ารูปแบบข้อมูลไหนที่เป็นรูปแบบปกติและรูปแบบข้อมูลไหนที่เป็นรูปแบบของการโจมตี

ในงานวิจัยนี้เลือกใช้ชุดข้อมูล NSL-KDD ซึ่งเป็นชุดข้อมูลที่ใช้สำหรับวิเคราะห์รูปแบบของการโจมตีและถูกพัฒนามาจากชุดข้อมูล KDD-Cup99 ที่มีข้อผิดพลาดหลายด้านทั้งข้อมูลที่ซ้ำกันจนทำให้เกิดการเอียงเอนของการประมวลผลข้อมูล และทำให้ข้อมูลมากเกินความจำเป็น ซึ่งชุดข้อมูล NSL-KDD เป็นที่ยอมรับในหลายงานวิจัยและให้ผลที่แม่นยำมากกว่า ประกอบกับการทำ Data Preprocessing โดยจะใช้เทคนิคของ One Hot Encoding ทำ Normalization และ Feature Selection แบบ (Attribute Ratio) เพื่อปรับแต่งและลดขนาดของชุดข้อมูล

จากนั้นทำการสร้างโมเดลโดยใช้เทคนิคการเรียนรู้ของเครื่องแบบไฮบริดซึ่งประกอบด้วย supervised learning และ unsupervised learning โดย supervised learning ใช้อัลกอลิทึมแบบ Tree-based (Random Forest, Adaboost , XGBoost) ซึ่ง Tree-based เป็นอัลกอลิทึมที่เหมาะกับการสร้างโมเดลแบบ classification เพราะมีความไวต่อการเปลี่ยนแปลงในการเทรนข้อมูลและรูปแบบการทำงานของ Tree-based เป็นการทำงานแบบแยกย่อย ส่งผลให้มีจำนวนของโมเดลเพิ่มขึ้น เพื่อแยกกันทำงาน ทำให้ช่วยลดเวลาในการเทรนข้อมูล ในส่วนของ unsupervised learning ใช้อัลกอลิทึมแบบ K-Means Clustering ซึ่งเป็นเทคนิคการเรียนรู้แบบไม่มีผู้สอน ใช้สำหรับจัดกลุ่มรูปแบบข้อมูลที่มีลักษณะคล้ายกันให้อยู่ในกลุ่มเดียวกัน เพื่อตรวจสอบข้อมูลในกรณีที่พบรูปแบบข้อมูลที่ไม่เคยรู้จักมาก่อน นำมาจัดกลุ่มให้อยู่ในกลุ่มข้อมูลที่มีรูปแบบคล้ายกัน และทำการ classifier ด้วย Tree-based จากนั้นทำการตรวจวัดความถูกต้องของโมเดลโดยใช้ Evaluation Matric (Accuracy, TPR, FPR, AUC (Area Under Curve)) เพื่อเปรียบเทียบประสิทธิภาพของโมเดลและหาโมเดลที่ดีที่สุด ที่สามารถนำมาวิเคราะห์การโจมตี

1.2 ความมุ่งหมายของงานวิจัย

1. ศึกษาการนำเทคโนโลยี ML (Machine Learning) แบบไฮบริด มาใช้ในการตรวจสอบการโจมตีบนเครือข่าย

2. ศึกษาการทำ Feature engineering เพื่อนำมาช่วยในกระบวนการลดขนาดของข้อมูลโดยตัดข้อมูลที่ไม่จำเป็นออก เพื่อให้โมเดลสามารถทำงานได้เร็วขึ้นและมีความแม่นยำมากขึ้น เนื่องจากการลดข้อมูลที่เป็นปัจจัยให้เกิดข้อผิดพลาด

3. เปรียบเทียบประสิทธิภาพของโมเดล เพื่อหาโมเดลที่ดีที่สุด ที่สามารถนำมาวิเคราะห์การโจมตีได้ดีที่สุด

1.3 ขอบเขตของการวิจัย

1. ใช้ชุดข้อมูลสาธารณะชื่อ NSL-KDD ในการทดสอบประสิทธิภาพของโมเดลการเรียนรู้ของเครื่อง

2. ทำการจำแนกรูปแบบของข้อมูลออกเป็น 2 คลาสคือ Normal และ Anomaly

3. ใช้ Clustering แบบ K-Mean Clustering

4. จะใช้ Classification model แบบTree-based (Random Forest, Adaboost, XGBoost)

5. ทดสอบประสิทธิภาพของงานโดยใช้ Evaluation Matric (Accuracy, True Positive Rate (TPR), False Positive Rate (FPR), Area Under the Curve (AUC))

1.4 วิธีดำเนินการวิจัย

1. ทำการทบทวนงานวิจัยที่เกี่ยวข้อง **(**Literature Review)

2. ศึกษาการจัดเตรียมรูปแบบข้อมูลให้เหมาะสมกับงานวิจัย

3. ศึกษาการทำ One Hot Encoding, Feature Selection**,** การ normalization data เพื่อปรับปรุง ประสิทธิภาพการประมวลผลในการทำเหมืองข้อมูลและการเรียนรู้ของเครื่องจักร

4. ศึกษาการเรียนรู้ของเครื่องแบบ Unsupervised Learning (K-Means Clustering) Supervise Learning (Random Forest, Adaboost, XGBoost) และแบบไฮบริด

5. ศึกษาเครื่องมือวัดประสิทธิภาพ Confusion Matrix, Precision and Recall, ROC Curve เพื่อนำาวัดความถูกต้องของโมเดล

6. นำผลที่ได้จากการศึกษามาสร้างแบบจำลองเพื่อทำนายความถูกต้องโดยใช้โปรแกรมและวัดค่าความถูกต้อง

7. รวบรวมผลที่ได้มาเปรียบเทียบความถูกต้องกับงานวิจัยเดิมที่มีอยู่

8. สรุปผลและเขียนต้นฉบับของบทความวิจัย

1.5 ประโยชน์ที่คาดว่าจะได้รับจากการวิจัย

1. ได้ผลของการตรวจจับ IDS แบบ Anomaly-based ที่มีความแม่นยำขึ้น

2. สามารถลดเวลาและประหยัดทรัพยากรณ์ในการประมวลผลข้อมูลจากการเรียนรู้ของเครื่องโดยใช้ Feature Engineering

บทที่ 2
ทฤษฎี สมมุติฐาน และกรอบแนวความคิดของโครงการวิจัย

2.1 การเรียนรู้ของเครื่อง (Machine Learning)

เทคนิคการเรียนรู้ของเครื่อง (Machine Learning) เป็นศาสตร์ทางคอมพิวเตอร์ที่เรียนรู้ความสัมพันธ์ของข้อมูล เพื่อเอามาใช้ประโยชน์เช่นการทำนายสินค้าที่คนจะซื้อจากข้อมูลประวัติการค้นหาโดยใช้อัลกอริทึมสร้างโมเดลหรือหาความสัมพันธ์จากชุดข้อมูลและทำนายหรือจำแนกประเภทของข้อมูล การเรียนรู้ของเครื่องสามารถให้ข้อมูลเชิงลึกที่สำคัญต่อการใช้งานที่หลากหลาย การเรียนรู้ของเครื่องสามารถแบ่งได้เป็น 3 ประเภทคือ supervised learning, unsupervised learning และ reinforcement learning

2.1.1 เทคนิคการเรียนรู้แบบมีผู้สอน (Supervised Learning)

เป็นเทคนิคการเรียนรู้ของเครื่องที่จะหาความสัมพันธ์หรือโมเดลระหว่างข้อมูลตัวอย่างที่มีให้และผลลัพธ์ที่ต้องการ โดยถ้าโมเดลที่ได้ให้ผลลัพธ์เป็นข้อมูลแบบต่อเนื่อง (Continuous) จะเรียกว่าวิธีการถดถอย (regression) และถ้าโมเดลที่ได้ให้ผลลัพธ์เป็นแบบหลายประเภท จะเรียกวิธีการนี้ว่าการจำแนก (classification) ซึ่งการ classification มีอยู่หลายรูปแบบเช่น Instance-based ตัวอย่างเช่น KNN หรือเป็นแบบ Tree-based เช่น Random Forest, Adaboost, Gradient Boosting ซึ่งในงานวิจัยนี้จะมุ่งไปที่การจำแนก (Classification) แบบ Tree-based

ในการทำ NIDS (Network Intrusion Detection System) งานวิจัยส่วนใหญ่มักจะใช้ tree-based เพราะให้ผลดี เนื่องจากประสิทธิภาพในการจัดหมวดหมู่ที่ดี มีความสามารถในการปรับขนาด สามารถทำงานแบบใช้ต้นไม้หลายๆต้นมาช่วยกันทำนายผลลัพธ์ได้ (Ensemble of Decision Trees)

2.1.2 เทคนิคการเรียนรู้แบบไม่มีผู้สอน (Unsupervised Learning)

จะตรงกันข้ามกับ supervised learning คือการเรียนรู้แบบนี้จะไม่มีการระบุผลลัพท์ (Label) ที่ต้องการไว้ และให้โมเดลหาความสัมพันธ์จากข้อมูลที่มีอยู่เอง จึงถูกเรียกว่าการเรียนรู้แบบไม่มีผู้สอน Unsupervised learning ซึ่ง Unsupervised learning มี 2 แบบคือ

2.1.2.1 แบบ Clustering เป็นการจัดลักษณะข้อมูลเป็นกลุ่มตามความคล้ายคลึงกันของข้อมูล

2.1.2.2 แบบ Association rule learning เป็นการหาความสัมพันธ์ของข้อมูล

ในงานวิจัยนี้จะใช้การทำ Clustering เพื่อนำมาใช้ตรวจจับรูปแบบของ Packet ที่ไม่เคยเจอมาก่อนในการตรวจจับรูปแบบที่ผิดปกติ anomaly-based

2.1.3 เทคนิคการเรียนรู้แบบเสริมกำลัง (Reinforcement learning)

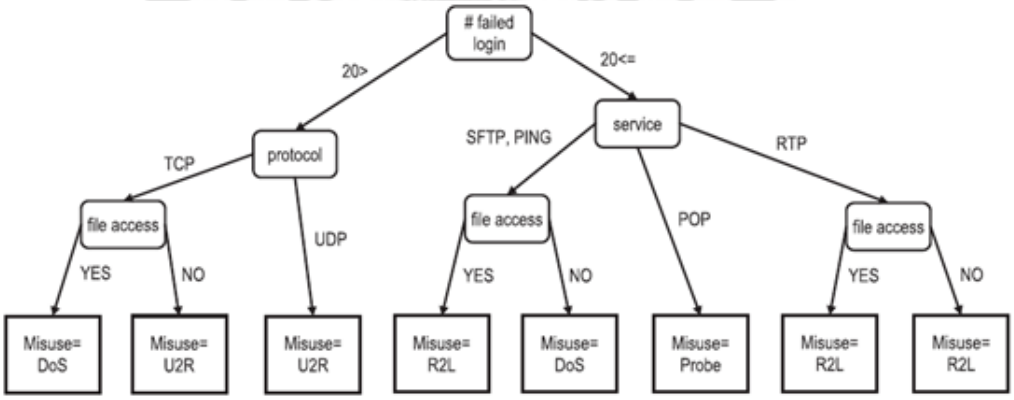
เป็นการพัฒนาระบบให้ดีขึ้นโดยอาศัยการปฏิสัมพันธ์กับสิ่งรอบข้าง ซึ่งลักษณะการทำงานจะคล้าย supervised learning แต่แทนที่จะใช้ Label ในการเทรน เปลี่ยนเป็นการใช้ฟังค์ชั่น reward (การให้รางวัล) เพื่อปรับปรุงประสิทธิภาพการทำงานแทน

2.2 Tree-based models

โมเดลต้นไม้ (Tree-based models) เป็นอัลกอลิทึมที่เหมาะกับการนำมาใช้สร้างโมเดลสำหรับ classification เพราะมีความไวต่อการเปลี่ยนแปลงในการเทรนข้อมูลและมีประสิทธิภาพมากเมื่อถูกนำมาใช้ทำการสุ่มแบบ subspace (แยกย่อย) ซึ่งจะส่งผลให้โมเดลมีจำนวนมากขึ้นและในแต่ละโมเดลจะทำหน้าที่ประมวลผลคุณลักษณะที่แยกย่อยออกจากชุดข้อมูลหลัก จึงทำให้ช่วยลดเวลาในการเทรนข้อมูล

2.2.1 ทฤษฎีเกี่ยวกับ Random Forest

Random Forest (ป่าสุ่ม) นิยมนำมาใช้งานกับการเรียนรู้ของเครื่องในช่วงทศวรรษที่ผ่านมา เนื่องจากประสิทธิภาพในการจัดหมวดหมู่ที่ดี มีความสามารถในการปรับขนาดได้และใช้งานง่าย Random Forest ถือเป็นกลุ่มของต้นไม้ตัดสินใจ (decision tree) แนวคิดนี้เกิดขึ้นเพื่อลดจุดอ่อนของการเรียนรู้แบบ ต้นไม้ตัดสินใจ (decision tree) และลดการทำให้เกิด overfitting ตัวอย่างของ Decision Tree ดังแสดงในภาพประกอบที่ 1



ภาพประกอบ 1 ตัวอย่าง Decision Tree ที่ใช้กับชุดข้อมูล NSL-KDD

ที่มา: A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection (p. 1164), by Buczak, A. L., & Guven, E, 2016

ใน Random forest นั้นจะประกอบด้วยต้นไม้หลายต้นและจะสุ่มเลือกคุณลักษณะของชุดข้อมูลมาเทรน จากนั้นจะนำผลที่ได้จากต้นไม้แต่ละต้นมาให้น้ำหนักโดยการโหวต โดยนำผลโหวตที่มากที่สุดมาเป็นคำตอบ

2.2.2 ทฤษฎีเกี่ยวกับ Adaboost

เป็นวิธีการเพิ่มความถูกต้องแบบ Boosting ซึ่งได้รับความนิยม หลักการทำงานของ Adaboost คือพยายามปรับค่าน้ำหนักของชุดข้อมูลแบบเทรน โดยสร้างตัวที่จะจำแนกในแต่ละรอบ กำหนดให้ Si คือตัวที่ใช้จำแนกประเภทของข้อมูล โดยให้ i = [1....n] ซึ่งเริ่มสร้างตัวจำแนกที่ S0 จากชุดข้อมูลเทรนนิ่ง ซึ่งการจำแนกครั้งนี้อาจจะยังไม่ถูกต้องที่สุด ลำดับต่อมาจึงเริ่มสร้างตัวจำแนกที่ S1 แล้วทำการเพิ่มค่าน้ำหนักให้กับข้อมูลที่ถูกจำแนกประเภทไม่ถูกต้องอีกครั้ง จากนั้นสร้างตัวจำแนกที่ S2 อีกทำซ้ำไปจนถึงตัวจำแนกที่ Sn จึงหยุดการเทรน ซึ่งจะพบว่าวิธีการนี้มีมีการจำแนกหลายรอบ เพื่อเพิ่มน้ำหนักให้ที่ชุดข้อมูลเทรนจำแนกไม่ถูกต้อง เมื่อนำตัวจำแนกตั้งแต่ S0 จนถึง S1 มาใช้ร่วมกัน ซึ่งที่คาดว่าจะได้รับคือการจำแนกข้อมูลที่ถูกต้องมากกว่าการใช้ตัวจำแนกเพียงตัวเดียว

2.2.3 ทฤษฎีเกี่ยวกับ Gradient Boosting

เป็น Boosting อีกเวอร์ชั่นหนึ่งที่ได้รับการออกแบบมาเป็นอย่างดี ให้ความแม่นยำสูง และยังสามารถจัดการกับข้อมูลที่กระจัดกระจายได้แบบคล่องตัว แต่เนื่องด้วยสถาปัตยกรรมแบบเรียงลำดับของมัน ทำให้ไม่เหมาะสำหรับการทำงานแบบคู่ขนาน ดังนั้นมันจึงไม่เหมาะกับชุดข้อมูลที่มีขนาดใหญ่ ซึ่งจะนิยมให้ Random forest มากกว่า Gradient Boosting จะใช้ Decision Trees จำนวนหลายๆต้น (Ensemble of Decision Trees) มาสร้างโมเดล โดยแต่ละต้นจะถูกสร้างแบบตามลำดับ เพื่อแก้ไขผลลัพท์ของการทำนายที่ผิดพลาดก่อนหน้าและนำผลลัพท์ของการทำนายสุดท้ายมารวมกันเป็นผลลัพท์ ซึ่งในงานวิจัยนี้จะใช้ไลบรารี่ XGBoost [2] ในการทำวิจัย

2.3 K-Means Clustering

คือหนึ่งในอัลกอริทึมที่ใช้เทคนิคการเรียนรู้แบบไม่มีผู้สอนและนิยมใช้ในการจัดกลุ่ม เนื่องจากง่ายและไม่ยุ่งยาก โดย K-Means จะแบ่ง (Partition) ของข้อมูลออกเป็น K กลุ่ม โดยจุดศูนย์กลาง (centroid) ของกลุ่มจะแทนที่ด้วยค่าเฉลี่ยในแต่ละกลุ่มโดยการวัดระยะห่างของข้อมูลในกลุ่มเดียวกัน เริ่มแรกขั้นตอนของการจัดกลุ่มโดยการหาค่าเฉลี่ยแบบ K ต้องกำหนดจำนวนกลุ่ม (K) ที่ต้องการก่อน และกำหนดจุดศูนย์กลางเริ่มต้นจำนวน K จุด สิ่งสำคัญในการกำหนดจุดศูนย์กลางเริ่มต้นของแต่ละกลุ่มนี้ ควรจะถูกกำหนดด้วยวิธีที่เหมาะสม เพราะตำแหน่งจุดศูนย์กลางเริ่มต้นที่แตกต่างกันทำให้ได้ผลลัพธ์สุดท้ายแตกต่างกัน ดังนั้นควรจะกำหนดจุดศูนย์กลางนี้ให้ห่างจากจุดศูนย์กลางอื่นๆ

จากนั้นจะแบ่งกลุ่มข้อมูลและความสัมพันธ์กับจุดศูนย์กลางที่ใกล้มากที่สุด โดยแต่ละจุดจะถูกกำหนดไปยังจุดศูนย์กลางที่ใกล้เคียงที่สุดจนครบหมดทุกจุด และคำนวณจุดศูนย์กลางใหม่ โดยการหาค่าเฉลี่ยทุกข้อมูลที่อยู่ในกลุ่ม หากจุดศูนย์กลางในแต่ละกลุ่มถูกเปลี่ยนตำแหน่ง จะได้จุดมีความสัมพันธ์กับกลุ่มใหม่และใกล้กับจุดศูนย์กลางใหม่ ทำซ้ำไปเรื่อย ๆ จะสังเกตได้ว่าผลลัพธ์จากการทำซ้ำแบบนี้ทำให้จุดศูนย์กลางเปลี่ยนตำแหน่งทุกรอบ จนกระทั่งจุดศูนย์กลางจำนวน K จุด ไม่มีการเปลี่ยนแปลงจึงจะสิ้นสุดกระบวนการ

2.4 Feature Selection

Feature Selection มีความสำคัญในการเพิ่มประสิทธิภาพของของ Data Mining Algorithm เนื่องจากข้อมูลส่วนใหญ่ประกอบด้วยคุณลักษณะที่ไม่เกี่ยวข้อง ซ้ำซ้อนหรือข้อมูลที่ไม่จำเป็น การเลือกคุณลักษณะ (Feature Selection) คือการคัดเลือกคุณลักษณะที่มีคุณภาพตามเกณฑ์ที่กำหนดและเทคนิคที่สำคัญที่ใช้บ่อยในการทำ Data Mining คือการลดรูปข้อมูล ซึ่งจะช่วยลดจำนวนของ Feature ที่ไม่เกี่ยวข้อง ซ้ำซ้อนและไม่จำเป็นออก และนำมาซึ่งผลที่เห็นได้ชัดสำหรับแอ็พพลิเคชัน ความเร็วในการประมวลผล ช่วยในการปรับปรุงการเทรนข้อมูลให้มีความแม่นยำมากขึ้น

ในงานวิจัยนี้ผู้วิจัยใช้ Feature Selection แบบ AR (Attribute Ratio) เพื่อทำการคัดเลือกคุณลักษณะที่จำเป็นต้องใช้ โดยจากการทบทวนวรรณกรรม [5] พบว่า Feature Selection แบบ AR ให้ความถูกต้อง สูงกว่า Feature Selection แบบ CFS (Correlation-based Feature Selection), IG (Information Gain) และ GR(Gain Ratio) โดยใช้ J45 Decision tree classifier กับ K-Fold cross validation เป็นตัววัดความแม่นยำ โดยค่า Attribute Ratio ของ feature j (AR(j)) สามารถคำนวณได้ดังสมการที่ (1)

$$\mathrm{AR(j)} = \underset{i \in [0, N-1]}{Max\ (CR_i(j))} \quad (1)$$

โดย CRi(j) คือ Class Ratio สำหรับ feature j บน label i ($\mathbf{i \in [0, N-1]}$) โดย N คือจำนวนของ label

CRi(j) สำหรับ feature j ที่มีค่าเป็น Numeric สามารถคำนวณได้ดังสมการ 2

$$CR_i(j) = \frac{AVG_{i,j}}{AVG_{T,j}} \quad (2)$$

โดยค่าเฉลี่ยของคุณลักษณะในแต่ละคลาส ($AVG_{i,j}$) สามารถคำนวณได้ดังสมการที่ 3

$$AVG_{i,j} = \frac{C_{i,j}}{N_{i,j}} \quad (3)$$

Ci,j คือผลรวมของ Feature j ที่มี label =i,

Ni,j คือจำนวน record ของ Feature j ที่มี label = i

ส่วน $AVG_{T,j}$ มาจากผลรวมของ Feature j ทั้งหมด $\left(\sum_i C_{i,j}\right)$ หาร ด้วย จำนวน record ของ Feature j ที่เป็นทั้งหมด (N) ดังแสดงในสมการที่ 4

$$AVG_{T,j} = \frac{\sum_i C_{i,j}}{N} \quad (4)$$

สูตรการหาค่า CR หรับ Binary ดังแสดงในสมการที่ 5

$$CR_i(j) = \frac{Freq(1)_{i,j}}{Freq(0)_{i,j}} \quad (5)$$

$Freq(0)_{i,j}$ คือจำนวน record ของ label i $(i \in [0, N-1])$ ที่มี Feature j มีค่าเป็น 0

$Freq(1)_{i,j}$ คือจำนวน record ของ label i $(i \in [0, N-1])$ ที่มี Feature j มีค่าเป็น 1

2.5 Evaluation Metrics

เราใช้ Confusion Matrix ในการประเมินประสิทธิภาพของโมเดลในการจำแนกข้อมูล โดยประเมินความสามารถในการจำแนกข้อมูลจากการทดสอบ ซึ่งมีรายละเอียดดังนี้

Pred.\True.	Actual Positive Class	Actual Negative Class
Positive Prediction	True Positive (TP)	False Positive (FP)
Negative Prediction	False Negative (FN)	True Negative (TN)

จาก Confusion Matrix สามารถอธิบายได้โดย

True Positive (TP) คือ จำนวนข้อมูลที่อยู่ในคลาส ถูก และถูกทำนายว่า ถูก

False Positive (FP) คือ จำนวนข้อมูลที่อยู่ในคลาส ผิด แต่ถูกทำนายว่า ถูก

True Negative (TN) คือ จำนวนข้อมูลที่อยู่ในคลาส ผิด และถูกทำนายว่า ผิด

False Negative (FN) คือ จำนวนข้อมูลที่อยู่ในคลาส ถูก แต่ถูกทำนายว่า ผิด

ค่าความถูกต้อง (accuracy) เป็นการประเมินประสิทธิภาพการจำแนกประเภทข้อมูลโดยการนำตัวแปรใน Confusion Matrix มาคำนวนหาค่าความถูกต้องดังสมการที่ (6)

$$\text{Accuracy} = \frac{TP + TN}{(\text{TP} + \text{FN} + \text{TN} + \text{FP})} \tag{6}$$

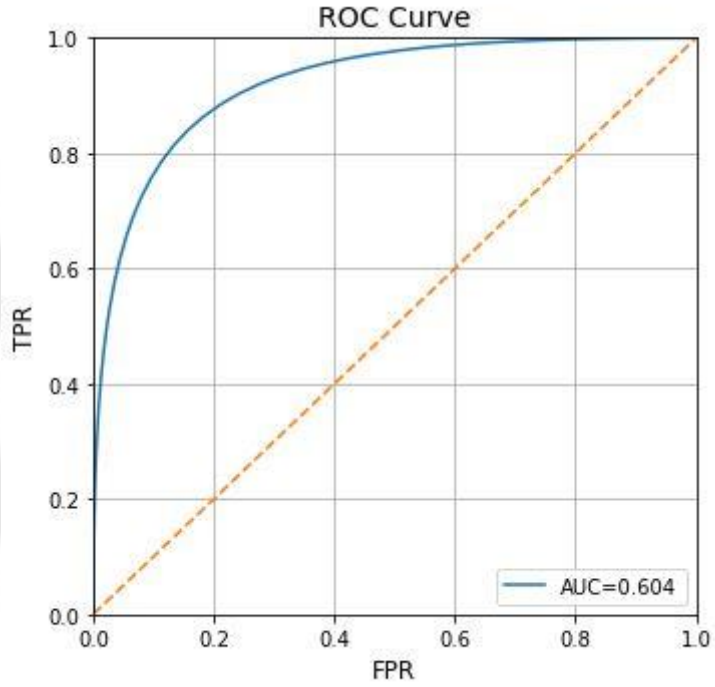
ค่า Precision เป็นการวัดความแม่นยำของการทำนายโมเดล โดยหาจากอัตราส่วนของการทำนายข้อมูลที่อยู่ในคลาส Positive ได้ถูกต้อง เปรียบเทียบกับจำนวนข้อมูลที่ถูกทำนายว่าเป็นคลาส Positive ทั้งหมด ดังสมการที่ (7)

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})} \tag{7}$$

ค่า Recall เป็นการวัดความถูกต้องของโมเดล โดยหาจากอัตราส่วนของการทำนายข้อมูลที่อยู่ในคลาส Positive ได้ถูกต้อง เปรียบเทียบกับจำนวนข้อมูลค่าจริงของคลาส Positive ทั้งหมด ดังสมการที่ (8)

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \quad (8)$$

ROC (Receiver Operating Characteristic) Curve เป็นอีกวิธีหนึ่งที่สามารถนำมาใช้ในการตรวจสอบความถูกต้อง โดยดูจากจุดตัดที่เหมาะสมของ True-Positive Rate (sensitivity) กับ False Positive Rate (1–Specificity) โดยค่าจุดตัด (cut-off point) ณ จุดต่างๆ จะแสดงความหมายตามภาพประกอบที่ 2



ภาพประกอบ 2 ตัวอย่าง ROC Curve

ที่มา: https://towardsdatascience.com/

โมเดลที่มีประสิทธิภาพควรมีค่า True Positive Rate สูง และมี False Positive Rate ต่ำ ซึ่งจะส่งผลให้ ROC Curve เข้าประชิดมุมบนซ้ายมากสุด นอกจากนี้ ROC Curve ยังช่วยเปรียบเทียบประสิทธิภาพของการ classifier โดยดูจากพื้นที่ใต้เส้นโค้ง AUC (Area Under the Curve) ซึ่งถ้าพื้นที่ใต้ Curve มากและเส้น Curve เข้าใกล้ 1 แสดงว่ามีประสิทธิภาพสูง

2.6 การทบทวนวรรณกรรม/สารสนเทศ (information) ที่เกี่ยวข้อง

งานวิจัย [1] มุ่งเน้นให้เห็นถึงข้อเสียของ Dataset KDD CUP 99 ซึ่งงานวิจัยนี้ไม่แนะนำให้ใช้ KDD CUP 99 Dataset ในการทำวิจัย เนื่องจากมีข้อมูลใน training set และ test set ที่ซ้ำ ซึ่งจะทำให้ค่า accuracy ที่สูงเกินจริง หรือจะเกิดการ over fitting งานวิจัยนี้เสนอให้ใช้ชุดข้อมูล NSL-KDD แทนซึ่งได้แก้ไขข้อผิดพลาดของ KDD CUP 99 แต่ก็ยังไม่สมบรูณ์ เ มื่อเทียบกับ network จริงที่มีอยู่

งานวิจัย [3] นำเสนอ วิเคราะห์รูปแบบของ traffic ที่ผิดปกติด้วยการนำเอาอัลกอลิทึมใน Machine Learning หลายแบบเอามารวมกัน เพื่อสร้าง Model ซึ่งทำงานบน Spark แต่ใช้ Dataset เป็น KDD CUP 99 ซึ่งเป็น Dataset ที่มีข้อผิดพลาด และอาจทำให้ค่าความแม่นยำของ Model ลดลง

งานวิจัย [4] นำเสนอการตรวจจับความผิดปกติด้วยวิธี multi-level network โดยใช้กฎที่น่าเชื่อถือตรวจจับพฤติกรรมที่ผิดปกติและสร้างโมเดล สามารถตรวจจับการโจมตีที่แน่นอน (เช่น DoS, R2L และ Probe) โดยใช้คุณลักษณะ DWT(discrete wavelet transform) ซึ่งงานวิจัยนี้มีวิธีการทำ multi-level ที่น่าสนใจ เช่นการสร้างกฎเพื่อตรวจจับผลลัพธ์ว่าปกติหรือผิดปกติ สร้างโมเดลที่ตรวจจับพฤติกรรมที่ผิดปกติโดยเฉพาะ

ในงานวิจัย [5] นำเสนอบททบทวนวรรณกรรมของการทำเหมืองข้อมูล (Data Mining (DM)) และการเรียนรู้ของเครื่อง (ML) ที่มุ่งเน้นการหาตัวอย่างของบทความที่อธิบายถึงการใช้เทคนิค DM และ ML ที่แตกต่างกันใน Cyber Security Domain ทั้งการตรวจจับแบบ misuse based และ anomaly based detection พร้อมทั้งอธิบายลักษณะเฉพาะของ ML และ DM สำหรับ Cyber Security ข้อดีและข้อเสียของ ML และ DM ในแต่ละแบบ

ในงานวิจัย [6] งานวิจัยนี้นำเสนอวิธีการเลือกคุณลักษณะโดยใช้ AR (Attribute Ratio)และเปรียบเทียบกับตัวเลือกคุณลักษณะแบบอื่นคือ CFS (Correlation-based Feature Selection), IG (Information Gain),และ GR(Gain Ratio) ซึ่งให้ค่าความแม่นยำถึง 99.794% จากชุดข้อมูลแบบเทรน โดยใช้เพียง 22 คุณลักษณะซึ่งช่วยลดจำนวนคุณลักษณะที่ไม่จำเป็นออก ซึ่งจะถูกนำมาช่วยในงานวิจัยนี้ เนื่องจากงานวิจัยนี้มีการทำ OHE (One Hot Encoding) ซึ่งมีคุณลักษณะที่มากกว่าเดิมถึง 3 เท่า

ในงานวิจัย [7] การสำรวจความผิดปกติโดยใช้เทคนิคการทำเหมืองข้อมูล งานวิจัยอธิบายการตรวจจับความผิดปกติโดยใช้การทำเหมืองข้อมูล และมุ่งอธิบายอัลกอลิทึมที่ใช้ในการตรวจจับความผิดปกติทั้งแบบ Clustering แบบ Classification และแบบไฮบริด จากงานวิจัยนี้พบว่าการใช้เทคนิคแบบไฮบริด supervised และ unsupervised ให้ผลในการตรวจจับความผิดปกติได้แม่นยำได้ดีกว่า อีกทั้งยังปิดข้อด้อยของอัลกอลทึมแบบเดี่ยว ซึ่งจะนำมาใช้กับงานวิจัยนี้

ในงานวิจัย [8] นำเสนอการประมาณการสูญเสียบนเครือข่ายสายส่งไฟฟ้า (power line) โดยใช้ hybrid model ที่รวมกันระหว่าง clustering method (K-Medoids) และ Gradient Boosting โดยบทความนี้แสดงให้เห็นว่าการใช้ hybrid model สามารถประมาณการสูญเสียบนเครือข่ายได้แม่นยำกว่า ML model แบบอื่นๆอาทิ เช่น Random Forest, Neural Network โดยความคลาดเคลื่อนของแต่ละโมเดลจะถูกพิจารณาโดยใช้ RMSE, MAE และ MAPE

ในงานวิจัย [9] นำเสนอการใช้ Recurrent Neural Network (RNN) ในการตรวจจับการโจมตี โดยจากการทดสอบบนฐานข้อมูล NSL-KDD พบว่า RNN IDS มีประสิทธิภาพสูงกว่า ML แบบอื่นๆ (naive bayes (NB), RF) ในการตรวจจับการโจมตีทั้งบนฐานข้อมูล KDDTest+ และ KDDTest-21 ทั้งในการวิเคราะห์การโจมตีแบบ Binary classes และแบบ Multiple classes อย่างไรก็ตามวิธีนี้ต้องใช้เวลาค่อนข้างนานในการ Train ฐานข้อมูลซึ่งอาจไม่เหมาะสมในการนำไปใช้งานในสภาพแวดล้อมจริง

บทที่ 3
วิธีดำเนินการวิจัย

ในการวิจัยครั้งนี้ ผู้วิจัยได้ดำเนินการวิจัยโดยเริ่มจากการจำแนกข้อมูลของชุดข้อมูล NSL-KDD เพื่อวิเคราะห์รูปแบบของข้อมูล เช่นประเภทของคุณลักษณะในชุดข้อมูล NSL-KDD จำแนกประเภทการโจมตีที่มีอยู่ในชุดข้อมูล ทั้งชุดข้อมูลแบบเทรนและแบบทดสอบ ชนิดของคุณลักษณะในชุดข้อมูลแบบเทรน จำนวนของ Normal และ Anomaly ในชุดข้อมูล NSL-KDD

จากนั้นออกแบบอัลกอลิทึมและแบบจำลองที่จะใช้ในการสร้างโมเดล เพื่อเปรียบเทียบหาโมเดลที่ดีที่สุด ที่นำมาใช้ตรวจจับรูปแบบการบุกรุก โดยมีรายละเอียดดังต่อไปนี้

3.1 จำแนกชุดข้อมูล NSL-KDD

NSL-KDD Dataset เป็นชุดข้อมูลที่ถูกนำมาใช้วิเคราะห์รูปแบบการบุกรุกของแพ็กเก็ตบนระบบโครงข่าย ซึ่งเดิมทีชุดข้อมูลดังกล่าว ถูกพัฒนามาจากชุดข้อมูล (DARPA) 1998 และ DARPA 1999 ของสำนักงานโครงการวิจัยขั้นสูงของกลาโหมสหรัฐอเมริกา [5]

โดยชุดข้อมูล DARPA 1998 ถูกสร้างขึ้นโดยกลุ่มระบบและเทคโนโลยีระบบดิจิทัลของสถาบันเทคโนโลยีแห่งมลรัฐแมสซาชูเซตส์ห้องปฏิบัติการลินคอล์น (MIT / LL) โดยการสร้างเครือข่ายจำลองข้อมูล ซึ่งข้อมูลถูกรวบรวมมาจากข้อมูล TCP/IP network ข้อมูล log ของโมดูล Solaris Basic Security และ Solaris file system ที่เก็บ User และ root และระบบปฏิบัติการ (OS) ซึ่งข้อมูลทั้งหมดใช้เวลารวบรวมถึง 9 สัปดาห์ โดย 7 สัปดาห์แรกถูกจัดให้เป็นชุดข้อมูลสำหรับเทรน และ 2 สัปดาห์ที่เหลือถูกใช้เป็นชุดข้อมูลำหรับการทดสอบ ซึ่งชุดข้อมูลถูกจำลองการโจมตีในทั้งชุดข้อมูลที่ใช้สำหรับการเทรน และสำหรับทดสอบ

ชุดข้อมูล KDD 1999 ถูกสร้างขึ้นสำหรับการแข่งขัน KDD Cup ในปี 1999 ซึ่งชุดข้อมูลนี้ถูกอ้างอิงมาจากชุดข้อมูล DARPA 1998 ในส่วน TCP/IP และคุณลักษณะพื้นฐานที่ถูกบันทึกจากไฟล์ pcap รวมทั้งหมด 41 คุณลักษณะ มี record โดยประมาณ 4 ล้าน record

ชุดข้อมูล KDD Cup 99 ที่ถูกใช้กันอย่างกว้างขวางในการตรวจจับรูปแบบการโจมตีของระบบโครงข่าย แต่ [1] Tavallaee et al. และคณะได้ทำการวิเคราะห์ทางสถิติกับชุดข้อมูลนี้และพบประเด็นสำคัญสองประเด็นที่ส่งผลต่อประสิทธิภาพของระบบที่ได้รับการประเมินผลและส่งผลต่อการตรวจจับแพ็กเก็ตที่ผิดปกติทำให้มีผลที่ผิดพลาด

เพื่อแก้ปัญหาของชุดข้อมูล KDD Cup 99 พวกเขาจึงได้นำเสนอชุดข้อมูล NSL-KDD ซึ่งมีชุดข้อมูลที่ถูกต้องและสมบูรณ์กว่า การเปรียบเทียบข้อดีของชุดข้อมูล NSL-KDD ที่ดีกว่าชุดข้อมูล KDD Cup 99 มีรายละเอียดดังนี้

1 ชุดข้อมูล NSL-KDD ไม่มีข้อมูลที่ซ้ำกันในชุดข้อมูลที่เป็น Training Set ดังนั้นการจำแนกข้อมูลจะไม่เอนเอียงไปทางข้อมูลที่ซ้ำกันบ่อยๆ

2 จำนวนของ record ที่ถูกเลือกจากระดับความยากของแต่ละกลุ่มมีความผกผันกับเปอร์เซนต์ของ record ในชุดข้อมูล KDD cup 1999 ผลที่ได้มาคืออัตราการจำแนกรูปแบบของการเรียนรู้ของเครื่องมีช่วงที่กว้างขึ้น ซึ่งทำให้การประมวลผลมีประสิทธิภาพมากขึ้น

3 ทำให้มันสามารถรันเพื่อทดสอบบนชุดข้อมูลที่สมบูรณ์ โดยไม่จำเป็นต้องสุ่มเลือกเป็นชิ้นส่วนย่อยๆ ดังนั้นผลของงานวิจัยที่แตกต่างกัน จะมีความสอดคล้องและเปรียบเทียบได้

NSL-KDD Dataset มีชุดข้อมูลทั้งหมด 4 ชุดคือ

1. KDDTrain+ มี record จำนวน 125,973 record

2. KDDTrain+_20Percent มี record จำนวน 25,192 record

3. KDDTest+ มี record จำนวน 22,544 record

4. KDDTest-21 มี record จำนวน 11,850 record เป็นชุดข้อมูลที่ถูกสร้างขึ้นมา โดยแยกจาก KDDTest+ โดยใช้หลักการสร้างโมเดลการเรียนรู้ของเครื่อง (Machine Learning Model) จำนวน 21 Model มาทำนายชุดข้อมูล KDDTest+ ว่า record ไหนเป็นรูปแบบปกติ (Normal) และ record ไหนเป็นรูปแบบของการโจมตี (Attack) และตัดจำนวน record ที่ 21 โมเดลนี้ทำนายถูกทั้งหมดออก และนำข้อมูลที่ทำนายผิดที่เหลือมาเป็นชุดข้อมูลนี้

ในงานวิจัยนี้จะใช้ชุดข้อมูล KDDTrain+, KDDTest+ ในการทำงานวิจัย ชุดข้อมูล NSL-KDD ประกอบด้วย 41 คุณลักษณะดังที่แสดงในตารางที่ 1

ตาราง 1 คุณลักษณะของชุดข้อมูล NSL-KDD

No.	Features	Types	No.	Features	Types
1	duration	continuous	22	is_guest_login	symbolic
2	protocol_type	symbolic	23	count	continuous
3	service	symbolic	24	srv_count	continuous
4	flag	symbolic	25	serror_rate	continuous
5	src_bytes	continuous	26	srv_serror_rate	continuous

ตาราง 1 ต่อ

No.	Features	Types	No.	Features	Types
6	dst_bytes	continuous	27	rerror_rate	continuous
7	land	symbolic	28	srv_rerror_rate	continuous
8	wrong_fragment	continuous	29	same_srv_rate	continuous
9	urgent	continuous	30	diff_srv_rate	continuous
10	hot	continuous	31	srv_diff_host_rate	continuous
11	num_failed_logins	continuous	32	dst_host_count	continuous
12	logged_in	symbolic	33	dst_host_srv_count	continuous
13	num_compromised	continuous	34	dst_host_same_srv_rate	continuous
14	root_shell	continuous	35	dst_host_diff_srv_rate	continuous
15	su_attempted	continuous	36	dst_host_same_src_port_rate	continuous
16	num_root	continuous	37	dst_host_srv_diff_host_rate	continuous
17	num_file_creations	continuous	38	dst_host_serror_rate	continuous
18	num_shells	continuous	39	dst_host_srv_serror_rate	continuous
19	num_access_files	continuous	40	dst_host_rerror_rate	continuous
20	num_outbound_cmds	continuous	41	dst_host_srv_rerror_rate	continuous
21	is_host_login	symbolic			

จากตารางที่ 1 ผู้วิจัยจะอธิบายบางคุณลักษณะที่ใช้บ่อยในการเชื่อมต่อบนระบบโครงข่ายและinternet โดยมีรายละเอียดดังนี้

Duration คือระยะเวลาของการเชื่อมต่อ (มีหน่วยเป็นวินาที)

Protocol_type ประเภทของโพรโทคอลที่ใช้เชื่อมต่อประกอบด้วย tcp, udp, icmp

Service คือบริการที่ใช้เชื่อมต่อไปยังปลายทาง เช่น http, telnet, ftp เป็นต้น

Flag คือสถานะปกติ หรือมีข้อผิดพลาดของการเชื่อมต่อ

Src_bytes คือจำนวนไบต์ของข้อมูลที่ถูกส่งจากต้นทางไปที่ปลายทาง

Dst_bytes คือจำนวนไบต์ของข้อมูลที่จากปลายทางไปยังต้นทาง

จากตัวอย่างที่กล่าวมาข้างต้นมีการใช้บ่อยในการวิเคราะห์ปัญหาทางด้านโครงข่าย

ในชุดข้อมูล NSL-KDD มีการจำแนกข้อมูลที่เป็นการโจมตี 4 ประเภท สามารถแบ่งได้ดังแสดงในตารางที่ 2

ตาราง 2 การจำแนกชนิดของการโจมตี

No	Category	Attacks
1	DoS (Denial of Service)	Neptune, pod, smurf, teardrop, table,warezmaster,apache2, mail bomb, back, process
2	Probe	multihop, http tunnel, ftp_write, root kit, ps buffer overflow, xterm
3	R2L (Remote to Local)	named, snmpgetattack, xlock, send mail, guess_passwd
4	U2R (Unauthorized to Root)	mscan, ipsweep, nmap, port sweep, satan, saint

จากตารางที่ 2 สามารถอธิบายประเภทของการโจมตีได้ดังนี้

1. DoS คือ เป็นการโจมตีที่ไม่ให้เครื่องเป้าหมายสามารถทำงานได้ โดยการสะกัดกั้นการใช้งานปกติ ไม่ให้เข้าถึงเครื่องเป้าหมายได้ เช่นการส่งแพ็กเก็ตจำนวนมากไปยังเครื่องเป้าหมาย ทำให้คิวของการให้บริการของเครื่องเป้าหมายเต็ม ไม่สามารถให้บริการได้

2. Probing เป็นลักษณะของการสแกนพอร์ตและการตรวจหาข้อมูลที่อยู่บนเครือข่าย เพื่อหาช่องโหว่ของเป้าหมายและ นำมาเป็นข้อมูลในใช้การโจมตี โดยการโจมตีลักษณะนี้นิยมกันในหมู่แฮคเกอร์ เช่นการทำ port scan เป็นต้น

3. Remote to Local (R2L) เป็นลักษณะของการพยายามเข้าถึงเครื่องเป้าหมายโดยไม่ได้รับอนุญาตในการเข้าถึง เพื่อทำลาย เปลี่ยนแปลงหรือแก้ไขระบบงานของเป้าหมาย

4. User to Root (U2R) เป็นการที่ผู้ใช้งานพยายามเข้าใช้งานสิ่งที่ไม่ได้รับอนุญาตในการเข้าถึงข้อมูลหรือระบบ โดยใช้สิทธิ root

จากชุดข้อมูล NSL_KDD ทั้ง KDDTrain+ และ KDDTest+ สามารถจำแนกข้อมูลที่เป็น Normal และ Anomaly โดยมีรายละเอียดตามตารางที่ 3 และ 4

ตาราง 3 การจำแนกประเภทของชุดข้อมูลสำหรับเทรนของ NSL-KDD (KDDTrain+)

Classify	Number	Percent
Normal	67343	53.46
Anomaly	58630	46.54

สำหรับชุดข้อมูล KDDTest+ สามารถจำแนกชุดข้อมูลค่า Normal และ Anomaly ดังที่แสดงในตารางที่ 4

ตาราง 4 การจำแนกประเภทของชุดข้อมูลสำหรับทดสอบของ NSL-KDD (KDDTest+)

Classify	**Number**	**Percent**
Normal	9711	43.08
Anomaly	12833	56.92

NSL-KDD มีคุณลักษณะ 3 ชนิด คือ Numeric, Nominal และ Binary โดยคุณลักษณะที่ 7, 12, 14, 15, 21 และ 22 เป็น binary ส่วนคุณลักษณะที่เหลือเป็นชนิด numeric ดังที่แสดงในตารางที่ 5

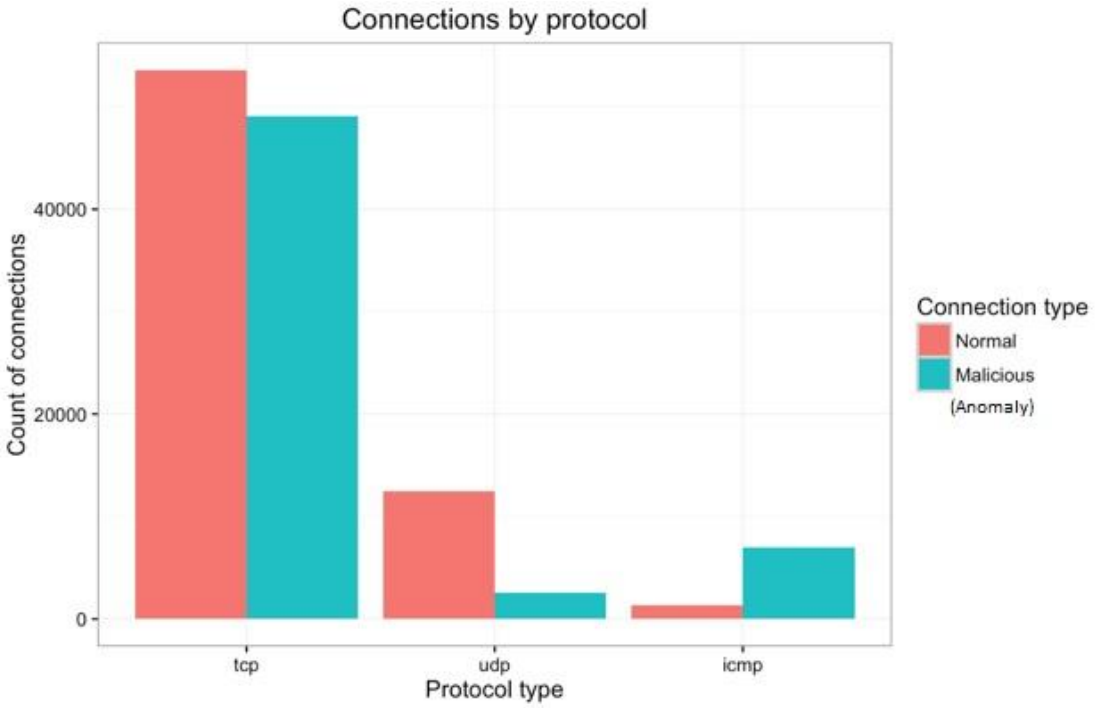
ตาราง 5 ชนิดของคุณลักษณะในชุดข้อมูลแบบเทรน

Type	**Features**
Nominal	Protocol_type(2), Service(3), Flag(4)
Binary	Land(7), logged_in(12), root_shell(14), su_attempted(15), is_host_login(21),, is_guest_login(22)

ตาราง 5 (ต่อ)

Type	Features
Numeric	duration(1), src_bytes(5), dst_bytes(6),wrong_fragment(8), urgent(9), hot(10),num_failed_logins(11),num_compromised(13), num_root(16), num_file_creations(17), num_shells(18), num_access_files(19), num_outbound_cmds(20), count(23) srv_count(24), serror_rate(25), srv_serror_rate(26), rerror_rate(27), srv_rerror_rate(28), same_srv_rate(29) diff_srv_rate(30), srv_diff_host_rate(31), dst_host_count(32), dst_host_srv_count(33), dst_host_same_srv_rate(34), dst_host_diff_srv_rate(35), dst_host_same_src_port_rate(36), dst_host_srv_diff_host_rate(37), dst_host_serror_rate(38), dst_host_srv_serror_rate(39), dst_host_rerror_rate(40), dst_host_srv_rerror_rate(41)

จากตารางที่ 5 จะพบว่า มีคุณลักษณะที่เป็น Binary อยู่ 6 คุณลักษณะ เป็น Nominal 3 คุณลักษณะ และเป็น Numeric 32 คุณลักษณะ ในส่วนของ Nominal ของชุดข้อมูล KDDTrain+ สามารถจำแนกข้อมูลของคุณลักษณะ Protocol_type โดยใช้กราฟแท่งดังแสดงในภาพประกอบที่ 6 ซึ่งจะจำแนกชนิดของ normal และ anomaly

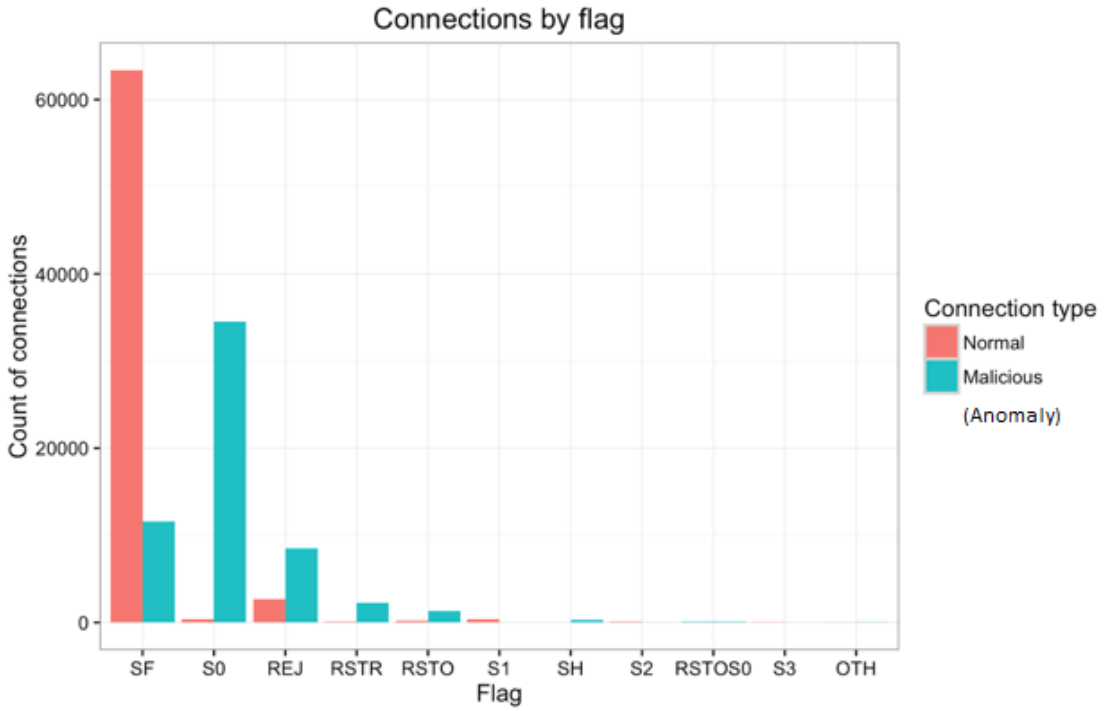


ภาพประกอบ 3 การจำแนกประเภทของโพรโทคอลในคุณลักษณะ Protocol_type (KDDTrain+)

ที่ ม า : https://nycdatascience.com/blog/student-works/network-intrusion-detection-2

จากภาพประกอบที่ 3 อินเทอร์เน็ตโพรโทคอลที่อยู่ในคุณลักษณะ Protocol type จะประกอบด้วย (TCP, UDP และ ICMP) โดย ในระบบโครงข่ายและอินเตอร์เนตส่วนใหญ่จะใช้ TCP โพรโทคอลในการเชื่อมต่อ เช่น HTTP, FTP, SMTP, Telnet, SSH เป็นต้น ซึ่งอยู่ในรูปแบบของ normal และ anomaly ในสัดส่วนที่ใกล้เคียงกัน ในส่วนของการเชื่องต่อแบบ UDP เช่น DNS, DHCP ส่วนใหญ่จะมีอัตราส่วนที่ปลอดภัยกว่า TCP เพราะมีอัตราส่วนของ Anomaly ที่น้อยกว่ามาก แต่การเชื่อมต่อแบบ ICMP เช่น ping, traceroute จากอัตราส่วนบนชุดข้อมูลจะพบว่า ICMP เป็นการเชื่อมต่อที่เป็นอันตราย ซึ่ง ICMP ถูกใช้บ่อยในการโจมตีแบบ DoS

ในส่วนของภาพประกอบที่ 4 แสดงรูปแบบของข้อมูลที่เป็น normal และ anomaly ในคุณลักษณะของ flag โดยมีรายละเอียดดังนี้

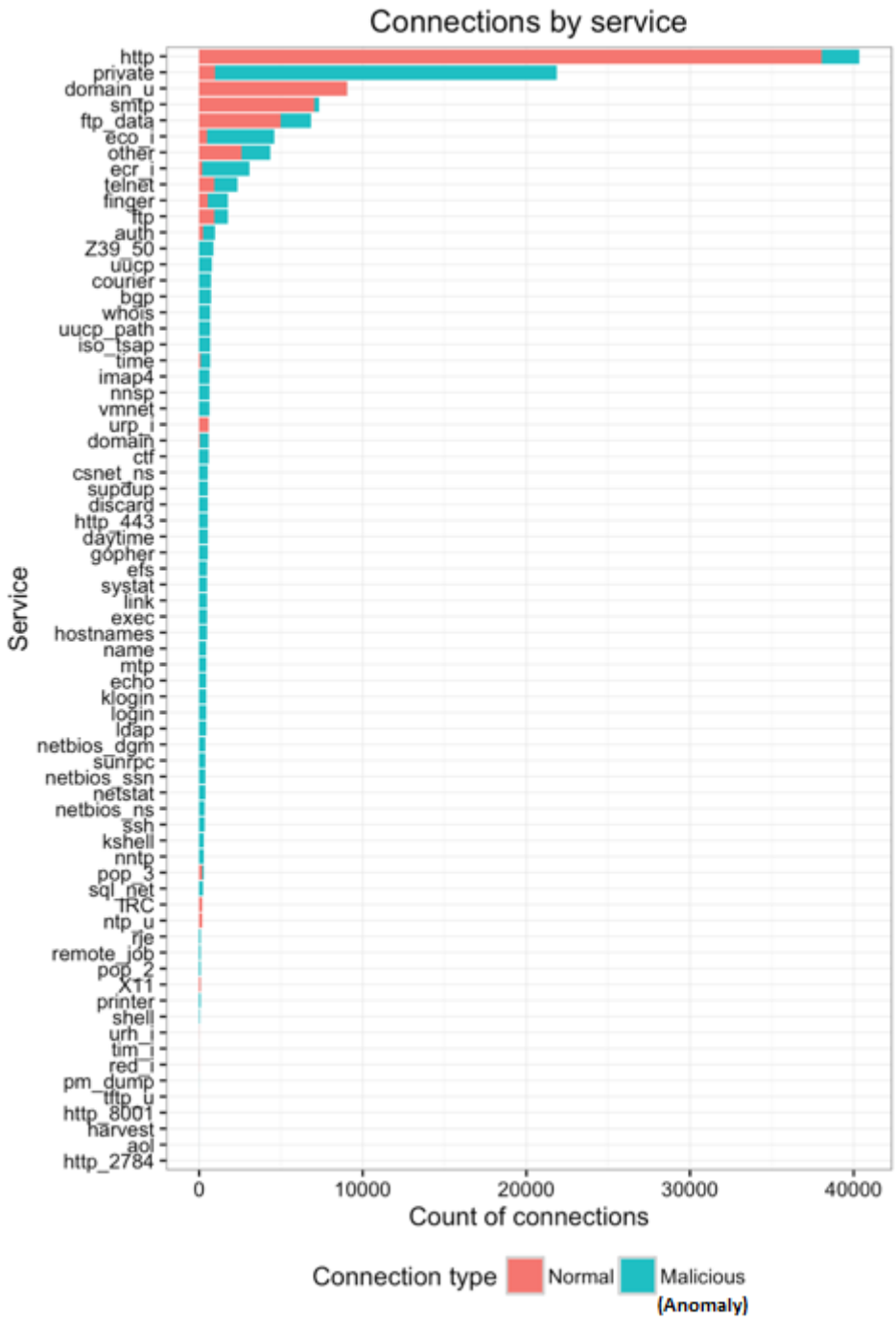


ภาพประกอบ 4 รูปแบบข้อมูลที่เป็น normal และ anomaly ในคุณลักษณะของ flag (KDDTrain+)

ที่มา : https://nycdatascience.com/blog/student-works/network-intrusion-detection-2

แต่ละการเชื่อมต่อคุณลักษณะ flag ซึ่งจะแสดงสถานะที่เกี่ยวข้องการเชื่อมต่อเช่น เริ่มต้นส่งแต่ไม่ได้รับการตอบกลับ (S0) ปฏิเสธ (REJ) รีเซ็ตการเชื่อมต่อโดยผู้ริเริ่ม (RSTOS), เสร็จสมบูรณ์แล้ว (SF) เป็นต้น จากกราฟแท่งที่แสดงในภาพประกอบที่ 7 จะพบว่า ในส่วนของ SF flag จะแสดงให้เห็นถึงการรับส่งข้อมูลที่เสร็จสิ้นตามปกติ ซึ่งพบว่าส่วนใหญ่การเชื่อมต่อลักษณะนี้จะไม่เป็นอันตราย ในส่วนของ S0 จะพบว่าเป็นสถานะ flag ที่อันตราย ซึ่งเข้าใจได้เนื่องจากเป็นการเริ่มต้นส่ง SYN ไปแต่ไม่ได้รับการตอบกลับมา ในส่วนของสถานะ REJ และ RST ถึงแม้พบว่ามีจำนวนน้อยก็ตามแต่ถ้าสถานะนี้เกิดขึ้นก็ถูกมองว่าไม่ปกติได้เช่นกันซึ่งสอดคล้องกับภาพประกอบที่ 4

นอกเหนือจากคุณลักษณะของ Protocal_type และ flag แล้ว ในชุดข้อมูล KDDTrain+ ยังมีส่วนคุณลักษณะ Service ที่ให้บริการในการเชื่อมต่อ เช่น HTTP, SMTP, Private เป็นต้น ดังแสดงในภาพประกอบที่ 5



ภาพประกอบ 5 รูปแบบข้อมูล normal และ anomaly ในคุณลักษณะ Service (KDDTrain+)

ที่ ม า : https://nycdatascience.com/blog/student-works/network-intrusion-detection-2

จากประกอบที่ 5 สามารถอธิบายบาง Service การเชื่อมต่อได้ดังนี้

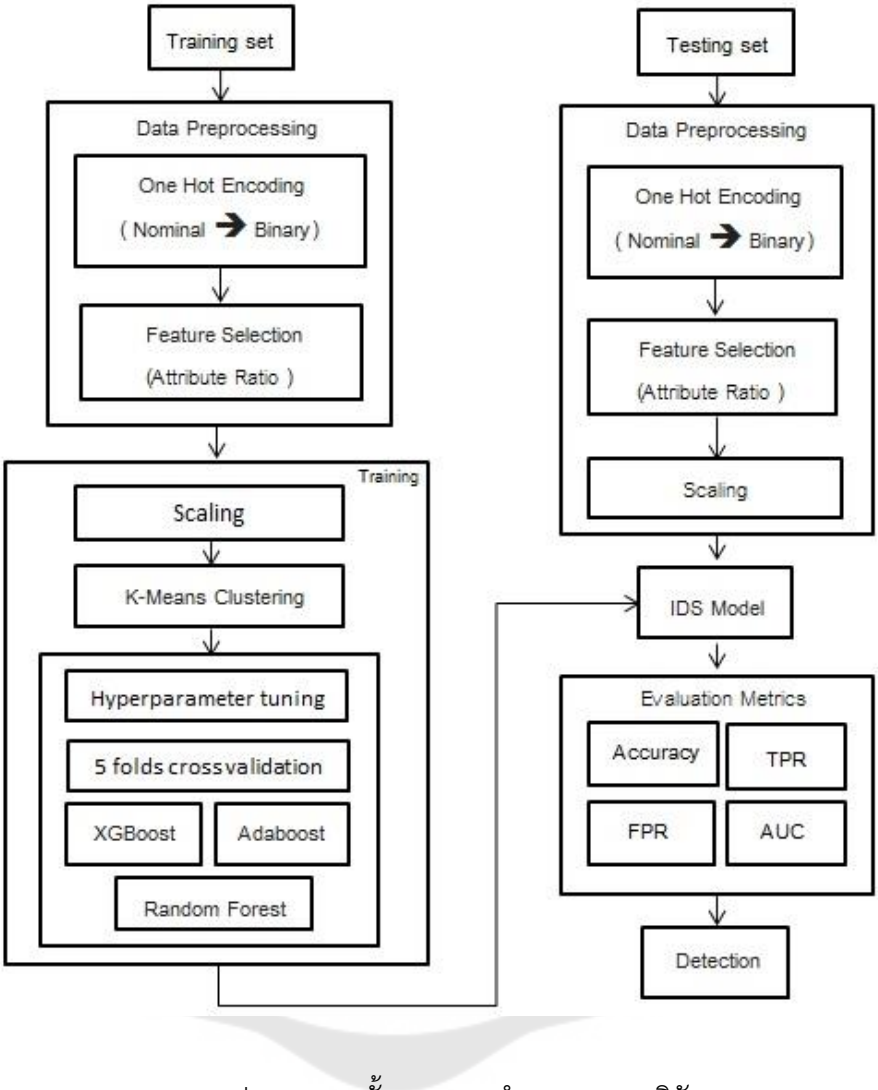
HTTP เป็นการเชื่อมต่อที่ใช้มากในการเชื่อมต่อเว็บไซต์ผ่านพอร์ทเน็ตเวิร์คหมายเลข 80 ซึ่งลักษณะการใช้งานส่วนใหญ่จะเป็นการเชื่อมต่อแบบปกติ และจากชุดข้อมูลพบว่ามี normal และ anomaly อยู่ในอัตราส่วนที่ใกล้เคียงกัน

Private เป็นการเชื่อมต่อภายใน โดยส่วนใหญ่การโจมตีจะเกิดจากเน็ตเวิร์คภายในเอง ซึ่งเป็นผลให้การเชื่อมต่อแบบ private กลับมีอัตราส่วน anomaly มากที่สุดเมื่อเทียบกับ normal

SMTP เป็นการเชื่อมต่อสำหรับเมล โดยใช้พอร์ทเน็ตเวิร์คหมายเลข 25 ในการเชื่อมต่อและใช้ชนิดของโปโตคอล (Protocol Type) ทั้ง TCP และ UDP เป็นการเชื่อมต่อแบบปกติ ซึ่งพบว่ามีข้อมูลที่เป็น normal และ anomaly อยู่ในอัตราส่วนที่ใกล้เคียงกันซึ่งมี anomaly เป็นจำนวนที่มากกว่า

ส่วนของ FTP เป็นการเชื่อมต่อสำหรับ รับ ส่ง ไฟล์ข้อมูล โดยใช้พอร์ทเน็ตเวิร์คหมายเลข 21 ซึ่งจะคล้ายกับ HTTP และ FTP ซึ่งส่วนใหญ่จะถูกใช้สำหรับการเชื่อมต่อปกติจากชุดข้อมูลพบว่ามี normal และ anomaly อยู่ในอัตราส่วนที่ใกล้เคียงกันโดยมี anomaly เป็นจำนวนที่มากกว่า เป็นต้น

3.2 ออกแบบอัลกอลิทึมและแบบจำลองที่จะใช้ในการสร้างโมเดล



ภาพประกอบ 6 ขั้นตอนการทำงานของงานวิจัย

จากภาพประกอบที่ 6 งานวิจัยนี้จะใช้ชุดข้อมูล KDDTrain+ สร้างโมเดลและใช้ชุดข้อมูล KDDTest+ สำหรับทดสอบความถูกต้องของโมเดลและในการทดลองจะต้องปรับข้อมูลให้เข้ากับโมเดลโดยสร้างอีก 1 คุณลักษณะ เพื่อทำการจำแนกการโจมตีที่มีให้อยู่ในคุณลักษณะของลาเบล

จากนั้นแทนค่าของลาเบลที่จำแนกแล้วเป็นตัวเลข 2 Label โดยแทนค่า Normal = 0 นอกนั้นจะ Anomaly = 1 ซึ่งการทำ Label จะทำให้ง่ายต่อการนำมาทดสอบกับการเรียนรู้ของเครื่องได้หลากหลายอัลกอลิทึมดังแสดงในภาพประกอบที่ 7

	label	label2	label2-index
0	normal	normal	0
1	normal	normal	0
2	neptune	anomaly	1
3	normal	normal	0
4	normal	normal	0
5	neptune	anomaly	1
6	neptune	anomaly	1
7	neptune	anomaly	1
8	neptune	anomaly	1
9	neptune	anomaly	1

ภาพประกอบ 7 การทำ Label ของชุดข้อมูลทดสอบ

หลังจากสร้างคุณลักษณะ label2 และ label2-index แล้ว จากนั้นทำขั้นตอน Data Preprocessing ตามที่กล่าวมาขั้นต้น โดยชุดข้อมูล NSL-KDD มีชนิดของคุณลักษณะ 3 ชนิด คือ Numeric, Nominal และ Binary ซึ่งคุณลักษณะ Nominal จะเป็นตัวหนังสือ ดังแสดงในภาพประกอบที่ 8

	protocol_type	service	flag
0	tcp	ftp_data	SF
1	udp	other	SF
2	tcp	private	S0

ภาพประกอบ 8 ตัวอย่างชนิดของคุณลักษณะ Nominal บนชุดข้อมูลแบบเทรน (KDDTrain+) ก่อนทำ One Hot Encoding

เนื่องจากคุณลักษณะ Nominal มีข้อมูลเป็นตัวหนังสือ จึงไม่สามารถนำมาสร้างโมเดล ผู้วิจัยได้ใช้ One Hot Encoding (OHE) ในการแปลงค่าข้อมูลจากตัวหนังสือเป็นตัวเลข Binary ทำให้ค่าคุณลักษณะเพิ่มขึ้น เพราะการทำ OHE จะนำค่าที่มีอยู่ในคุณลักษณะ Nominal มาสร้างคุณลักษณะใหม่และแทนค่า record ที่ตัวมันเป็น 1 ถ้าไม่มีเป็น 0 ดังแสดงในภาพประกอบที่ 9

	flag_RSTO	flag_RSTOS0	flag_RSTR	flag_S0	flag_S1
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	1	0

ภาพประกอบ 9 ตัวอย่าง record ที่ 2 บนชุดข้อมูลแบบเทรน (KDDTrain+) หลังทำ One Hot Encoding

จากการทำ One Hot Ending จะมีข้อเสียคือถ้าคุณลักษณะที่เป็น Nominal มีค่ามากและหลากหลาย One Hot Encoding จะสร้างคุณลักษณะเท่าที่มีในค่านั้นๆ ทำให้มีคุณลักษณะมากตาม และเมื่อรวมคุณลักษณะของ Numeric และ Binary ที่มีอยู่รวมเป็น 122 คุณลักษณะและจะส่งผลต่อเวลาในการประมวลผลของ CPU และ Memory ในการสร้าง Model และทำนายความแม่นยำ ซึ่งในบางคุณลักษณะไม่มีความจำเป็นในการในใช้สร้างโมเดลจึงทำให้เสียเวลาในการประมวลผลด้วย

ดังนั้นทางผู้วิจัยจึงได้ใช้ Feature Engineering ในการทำงานวิจัยนี้ โดยเลือกใช้เทคนิค Feature Selection (Attribute Ratio) เพื่อเลือกคุณลักษณะที่จำเป็นต้องใช้เท่านั้น นำมาสร้างโมเดล โดยการหาค่าอัตราเฉลี่ยสูงสุดของคุณลักษณะในแต่ละคลาส ดังที่กล่าวไว้ในหัวข้อที่ 2.4 ซึ่งหลังจากผ่านกระบวนการ AR แล้ว คุณลักษณะจะแสดงค่าของ AR ดังแสดงในภาพประกอบที่ 10

Index	Feature_name	AR_Value
10	num_shells	326.114
4	urgent	173.04
9	num_file_creations	62.2336
115	flag_SF	51
6	num_failed_logins	46.0386
5	hot	40.7745
34	protocol_type_tcp	16.3333
31	logged_in	10.5698
2	dst_bytes	9.15485
1	src_bytes	8.46406

ภาพประกอบ 10 ตัวอย่างของคุณลักษณะที่ถูกเลือกโดย AR

จากนั้นทำการกำหนดค่า AR เพื่อตัดคุณลักษณะที่ไม่จำเป็นออก โดยจากการทดลองพบว่าการตัดค่า AR ที่ 0.01 ให้ค่าความแม่นยำและเวลาในการเทรน (Accuracy) เหมาะสมที่สุด ดูรายละเอียดเพิ่มเติมจากภาคผนวก ซึ่งหลังจากทำ Feature Selection แล้ว จะเหลือค่าคุณลักษณะที่จำเป็นต้องใช้อยู่ที่ 77 คุณลักษณะจาก 122 คุณลักษณะ

เนื่องจากค่าในแต่ละคุณลักษณะมีค่าที่กระจายกัน มีความแตกต่างกันระหว่างค่าสูงสุดและต่ำสุดและในบางคุณลักษณะมีช่วงของข้อมูลที่กว้างมาก แต่บางคุณลักษณะมีช่วงของข้อมูลที่แคบ ดังนั้นผู้วิจัยจึงใช้วิธีปรับขนาดของค่าในแต่ละคุณลักษณะให้อยู่ในช่วงเดียวกันโดยใช้ฟังค์ชั่น scale ของ sklearn ใน library preprocessing เพื่อปรับขนาดของข้อมูลให้อยู่ในช่วงเดียวกันดังแสดงในสมการที่ 6

$$\acute{x}_i = \frac{x_i - \bar{x}}{s.d.} \tag{6}$$

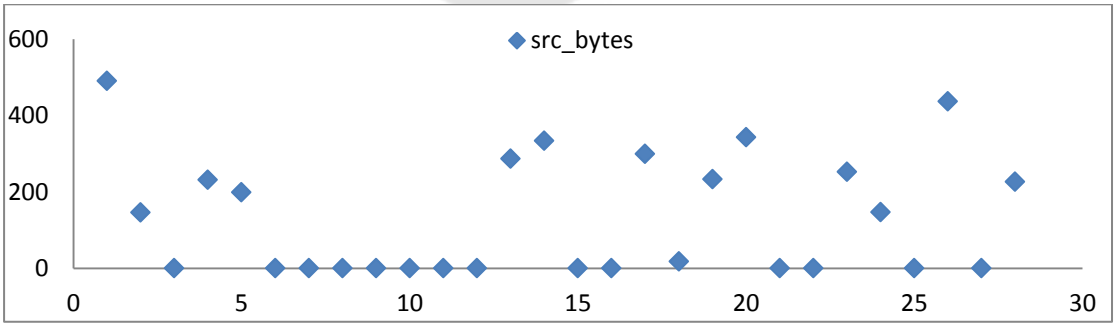
โดย $\acute{x}_i$ คือ ค่าของ record ที่ถูกทำ scale ข้อมูล

x_i คือ ค่าของ record ในคุณลักษณะที่ลำดับของ i

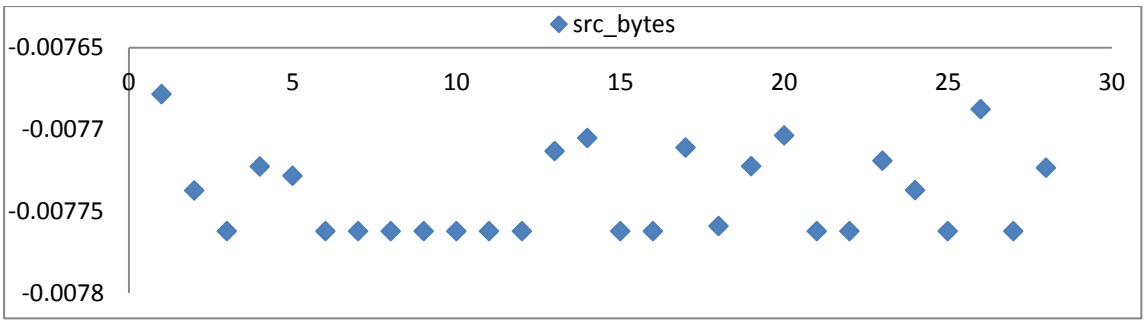
$\bar{x}$ คือ ค่าเฉลี่ยของคุณลักษณะ

standard deviation: s.d. คือ ค่าเบี่ยงเบนมาตราฐาน เป็นการวัดการกระจายของกลุ่มข้อมูล ซึ่งนำมาใช้แจกแจงความน่าจะเป็น

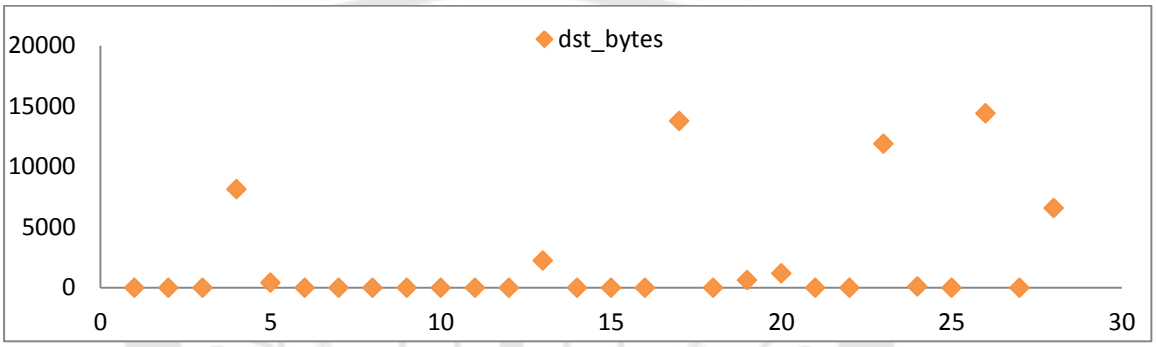
หลังจากทำ scale ข้อมูลแล้ว ค่าที่อยู่ในแต่ละคุณลักษณะจะได้ผลที่อยู่ใน range เดียวกัน ดังแสดงตัวอย่างเปรียบเทียบคุณลักษณะ src_bytes ตั้งแต่ record ที่ 1 ถึง record ที่ 28 ในภาพประกอบที่ 11 และ 12 และคุณลักษณะ dst_bytes ตั้งแต่ record ที่ 1 ถึง record ที่ 28 ดังแสดงในภาพประกอบที่ 13 และ 14



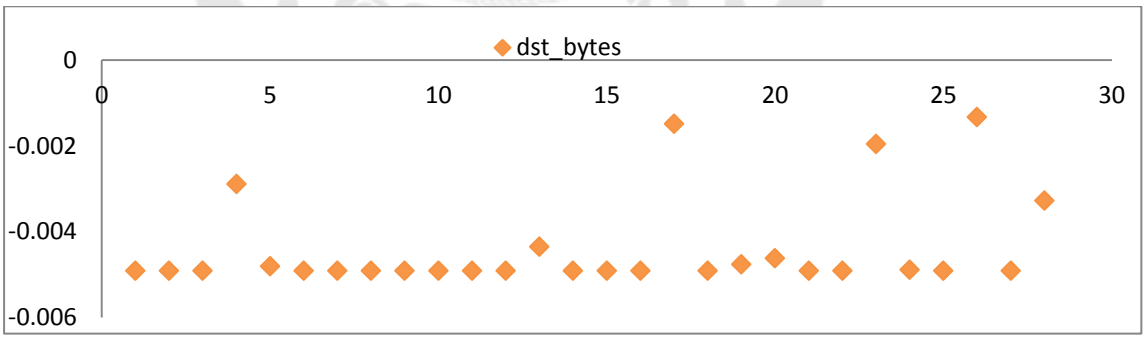
ภาพประกอบ 11 กราฟเชิงเส้นค่าของในคุณลักษณะของ src_bytes ก่อนทำ scaling



ภาพประกอบ 12 กราฟเชิงเส้นค่าของในคุณลักษณะของ src_bytes หลังทำ scaling

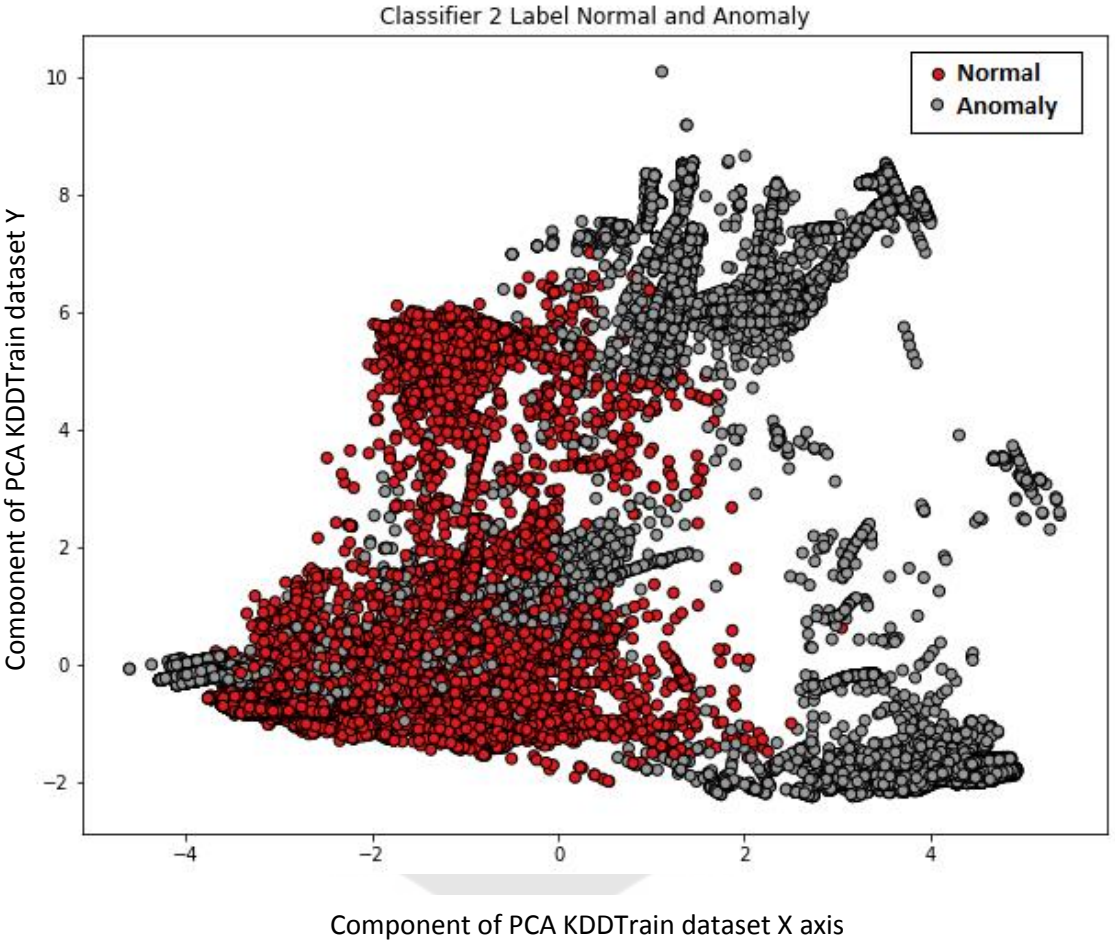


ภาพประกอบ 13 กราฟเชิงเส้นค่าของในคุณลักษณะของ dst_bytes ก่อนทำ scaling



ภาพประกอบ 14 กราฟเชิงเส้นค่าของในคุณลักษณะของ dst_bytes หลังทำ scaling

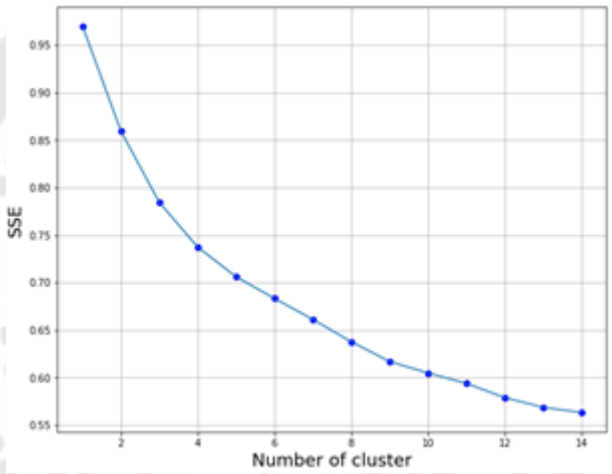
หลังจากทำขั้นตอน scale ข้อมูลเสร็จ ทางผู้วิจัยได้ใช้ PCA เพื่อทำการลดรูปของคุณลักษณะทั้งหมด เพื่อ plot กราฟเปรียบเทียบจำนวนข้อมูลระหว่าง normal และ anomaly ในชุดข้อมูลแบบเทรน โดยพบว่าจำนวนของข้อมูลทั้ง 2 ไม่ต่างกัน ดังแสดงในภาพประกอบที่ 15 จึงต้องการจัดกลุ่มข้อมูลโดยใช้ K-Means Clustering



ภาพประกอบ 15 กราฟการจำแนกข้อมูลที่เป็น Normal และ Anomaly

จากนั้นนำชุดข้อมูลมาทำการจัดกลุ่มข้อมูลที่มีลักษณะคล้ายกันโดยใช้ K-Means Clustering ซึ่งเป็น Unsupervised learning มีข้อดีคือช่วยลดความผิดพลาดในการตรวจจับรูปแบบของแพ็กเก็ตที่ไม่เคยรู้จัก (zero-day) ให้อยู่ในกลุ่มที่คล้ายกันก่อน แต่สิ่งหนึ่งที่เป็นปัญหาสำคัญสำหรับ K-Means Clustering คือการหาค่า K ที่เหมาะสมกับชุดข้อมูล

การเลือกค่า K (Choosing K) เพื่อที่จะใช้ในการจับกลุ่มให้ได้ผลลัพท์ในการจับกลุ่มที่ดีที่สุด ซึ่งโดยทั่วไปยังไม่มีวิธีการที่แน่นอนในการกำหนดค่า K แต่สามารถหาค่าประมาณได้โดยใช้เทคนิคข้อศอก (elbow method) [10] แนวคิดของข้อศอกคือการรัน K-Means clustering กับชุดข้อมูลโดยกำหนดช่วงของค่า K ตั้งแต่ 1 ถึง 15 สำหรับแต่ละค่าของ K จะคำนวณค่า sum of squared errors (SSE) และพล็อตกราฟเชิงเส้นของ SSE ซึ่งในแต่ละค่าของ K กราฟเชิงเส้นจะมีรูปร่างเหมือนแขนและจะเห็น "ข้อศอก" แต่จากภาพประกอบที่ 16 จะพบว่าเป็นการยากที่จะเห็นกราฟลักษณะในรูปของข้อศอก แต่สามารถสังเกตุได้จากจำนวนของ SSE ที่ drop ลงมากที่สุดในแต่ละจำนวน K ที่เพิ่มขึ้น



ภาพประกอบ 16 กราฟข้อศอกที่ใช้ในการกำหนดค่า K ในงานวิจัย

จากภาพประกอบที่ 16 แสดง SSE ของ K-Means Clustering โดยกำหนดช่วงของ K อยู่ที่ 1 – 15 ซึ่งพารามิเตอร์ของ K-Means ที่ใช้คือ n_init=25, max_iter=100, tol=0.0001, random_state = 3425 โดยสามารถอธิบายพารามิเตอร์ข้างต้นได้ดังนี้

n_init: จำนวนครั้งของอัลกอริทึม K-Means ที่รันด้วยจุดเซนทรอยด์ที่แตกต่างกัน ผลลัพธ์สุดท้ายจะเป็นผลลัพธ์ที่ดีที่สุดของ n_init ที่ทำงานติดต่อกันในแง่ของความเฉื่อยชา

max_iter: จำนวนสูงสุดของการทำซ้ำของอัลกอริทึม K-Means สำหรับการรัน 1 ครั้ง

tol: ความคลาดเคลื่อนยินยอมสัมพัทธ์กับการพิจารณาถึงความเฉื่อยที่ลู่เข้า

random_state: กำหนดจำนวนที่ใช้สุ่มสำหรับจุดเซนทรอยด์เริ่มต้น

ในการหาค่า K ที่เหมาะสม จากภาพประกอบที่ 16 จะพบว่าค่าของ SSE ลดลงสูงสุด (23.89%) เมื่อ K เพิ่มขึ้นจาก 1 เป็น 2 จากนั้นค่า SSE จะค่อยๆ ลด เมื่อจำนวน K เพิ่มขึ้น เช่นค่าของ SSE ลดลงเหลือ 9.30% เมื่อจำนวน K เพิ่มขึ้นจาก 2 เป็น 3 และเมื่อจำนวน K มากกว่า 10 ค่า SSE จะลดลงเรื่อยๆ จนเข้าใกล้เส้นตรงซึ่งไม่จำเป็นสำหรับนำมาใช้ในการจัดกลุ่ม

จากภาพประกอบที่ 16 พบว่า ค่า SSE ที่ drop ลงมากที่สุดคือจาก 1 มาที่ 2 ดังนั้นผู้วิจัยจึงนำค่า K = 2 มาสร้างโมเดล K-Means Clustering เพื่อจัดกลุ่มรูปแบบของข้อมูลที่คล้ายกัน โดยใช้ตัวอย่าง code ตามภาพประกอบที่ 17

```
# K-Means Clustering
from sklearn.cluster import KMeans
k = 2
km = KMeans(n_clusters = k, n_init=25, max_iter=100,
tol=0.0001, random_state=3425)
km.fit(df_train_std)
```

ภาพประกอบ 17 code การทำ K-Means Clustering ที่ K=2

จากการนำชุดข้อมูลเข้าโมเดล K-Means Clustering ทำให้ได้ผลการจัดกลุ่มอยู่ที่

Cluster 0 มีจำนวนของข้อมูลอยู่ที่ 47377 record

Cluster 1 มีจำนวนของข้อมูลอยู่ที่ 78596 record

หลังจากที่ได้กลุ่มข้อมูลที่ต้องการแล้ว นำข้อมูลกลุ่มทั้ง 2 มาเข้าโมเดลต้นไม้ โดยในงานวิจัยนี้จะใช้ Random Forest , Adaboost และ XGBoost ในการสร้างโมเดลและปรับแต่งค่าภายในพารามิเตอร์ของโมเดลโดยใช้การ Tuning แบบ RandomizedSearchCV เพื่อให้ได้ผลที่ดีที่สุดในการนำมาทดสอบความแม่นยำกับชุดข้อมูลแบบทดสอบ โดยจะใช้ Evaluation Metrics เป็นตัววัดความแม่นยำ เมื่อได้ Model ที่ต้องการแล้ว จากนั้นนำชุดข้อมูลที่ใช้ทดสอบ KDDTest+ มาทำการทดสอบ เพื่อดูเปรียบเทียบประสิทธิภาพของโมเดล เพื่อหาโมเดลที่ดีที่สุด ที่สามารถนำมาวิเคราะห์การโจมตีได้ดีที่สุด

บทที่ 4
ผลการดำเนินงานวิจัย

ผู้วิจัยเปรียบเทียบประสิทธิภาพของการเรียนรู้ของเครื่องแบบไฮบริดที่นำเสนอในแง่ของ Accuracy ,TPR, FPR และ AUC กับแนวคิดของ RNN ที่เป็น Baseline ของงานวิจัยนี้และ Classifier ชนิดต้นไม้ (Random Forest, Adaboost, XGBoost) โดยใช้ชุดข้อมูล KDDTrain+ ทำการเทรนข้อมูลเพื่อสร้างโมเดลและประเมินประสิทธิภาพของโมเดลโดยใช้ชุดข้อมูล KDDTest+ ผู้วิจัยใช้วิธีของ Feature Selection แบบ AR มาเลือกคุณลักษณะ โดยใช้ค่า AR ของ Feature Selection เป็นตัวกำหนดและเลือกคุณลักษณะ โดยคุณลักษณะที่มีค่า AR น้อยกว่า 0.01 จะถูกตัดออกจากการนำมาสร้าง model ผู้วิจัยใช้ Cluster ที่ค่า K=2 โดยแบ่งข้อมูลเป็น 2 Cluster สำหรับการทำ K-Means Clustering ตามที่นำเสนอในภาพประกอบที่ 16 ซึ่งพบว่าที่ K=2 มีค่าความชันของ SSE ลดลงมากที่สุด หลังจากที่ได้กลุ่มของรูปแบบข้อมูลที่คล้ายกันแล้ว จากนั้นผู้วิจัยนำเข้าโมเดลต้นไม้ทั้ง 3 แบบคือ Random Forest, Adaboost, XGBoost มาทำ Classification และผู้วิจัยได้ทำ hyperparameter tuning โดยใช้อัลกอริทึมแบบ RandomizedSearchCV เพื่อเลือกพารามิเตอร์ที่ดีที่สุด (best score) มาสร้างโมเดล

4.1 K-Means Clustering + Random Forest

หลังจากได้ทำการจับกลุ่มชุดข้อมูลเป็น 2 กลุ่มและทำการ Scale ข้อมูลแล้ว ผู้วิจัยได้เริ่มทำวิจัยโดยนำชุดข้อมูลทั้ง 2 กลุ่ม เข้าอัลกอลิทึมแบบป่าสุ่มก่อน (Random Forest) โดยในแต่ละชุดของข้อมูลจะถูกทำ Tuning โดยใช้ hyperparameter ในการ Tuning และผู้วิจัยใช้ 5-fold cross-validation เพื่อเทรนและตรวจสอบประสิทธิภาพโมเดลของผู้วิจัยในแต่ละกลุ่ม

```
param_grid = {
 'max_depth': [80, 90, 100],
 'max_features': [2, 3],
 'min_samples_leaf': [3, 4, 5],
 'min_samples_split': [8, 10, 12],
 'n_estimators': [100, 200, 300],
 'bootstrap': [True, False],
 'criterion': ["gini", "entropy"]
 }
```

ภาพประกอบ 18 พารามิเตอร์ที่ใช้สำหรับการทำ Tuning ในอัลกอลิทึมแบบป่าสุ่ม

จากภาพประกอบที่ 18 ผู้วิจัยได้อธิบายพารามิเตอร์ที่ใช้สำหรับ tuning โดยมีรายละเอียดดังนี้

max_depth: จำนวนสูงสุดลำดับชั้นของต้นไม้

max_features: จำนวนคุณลักษณะที่ถูกพิจารณาเพื่อหาการแบ่งคุณลักษณะที่ดีที่สุด

min_samples_leaf: จำนวนของการสุ่มที่ต้องการจะอยู่ที่โหนด

min_samples_split: จำนวนของการสุ่มขั้นต่ำที่ถูกแบ่งภายในหนึ่งโหนด

n_estimators: จำนวนของต้นไม้ทั้งหมดที่ใช้ในโมเดล

bootstrap: สุ่มตัวอย่าง bootstrap ที่จะถูกใช้เมื่อสร้างต้นไม้

criterion: ฟังค์ชั่นที่ใช้วัดคุณภาพของการแบ่งในหนึ่งครั้ง

จาก parameter tuning และ seed ค่าไว้ที่ 3425 และหลังจากการทำ Tuning ทำให้ได้ผลคะแนนสูงสุดของแต่ละพารามิเตอร์ที่จะนำมาสร้างโมเดลดังแสดงในตารางที่ 6

ตาราง 6 ค่า Best Score ของ hyperparameter ของ Random Forest ในแต่ละ Cluster

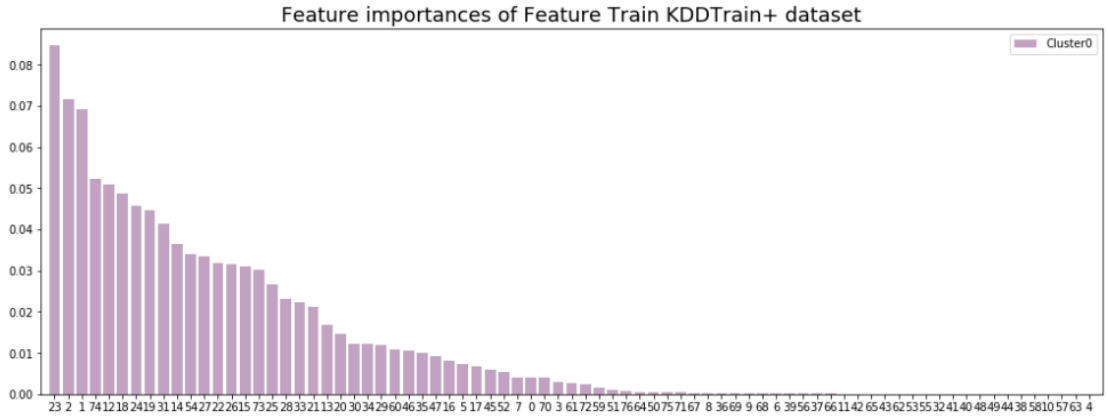
Cluster	n_estimators	min_samples_split	min_samples_leaf	max_features	max_depth	criterion	bootstrap
0	100	10	3	3	100	gini	False
1	100	10	3	3	100	gini	False

จากผล Best Score ในตารางที่ 6 เมื่อนำมาทดสอบร่วมกับชุดข้อมูล KDDTrain+ และ KDDTest+ ทำให้ได้ประสิทธิภาพของโมเดลดังแสดงในตารางที่ 7 ซึ่งจากการแบ่งกลุ่มของข้อมูลที่ K=2 ในการสร้างและทดสอบโมเดลบนชุดข้อมูล KDDTrain+ ให้ค่าความแม่นยำเท่ากับ 99.67% ค่า True Positive Rate เท่ากับ 99.91% ค่า False Positive Rate เท่ากับ 0.55% ค่า Area Under The Curve เท่ากับ 0.999 และทดสอบโมเดลกับชุดข้อมูล KDDTest+ ให้ค่าความแม่นยำเท่ากับ 73.63% ค่า True Positive Rate เท่ากับ 90.60% ค่า False Positive Rate เท่ากับ 36.79% และค่า Area Under The Curve เท่ากับ 0.918

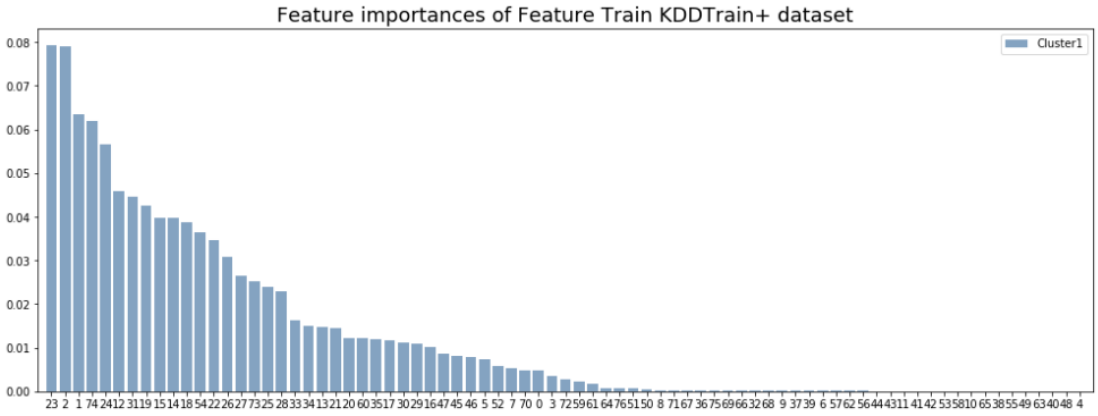
ตาราง 7 ค่าของ Accuracy, TPR ,FPR และ AUC ใน K-Means Clustering กับ Random Forest Classifier

Number of K in K-Means	**KDDTrain+**	**KDDTest+**
K-Means K=2	Accuracy = 99.67%	Accuracy = 73.63%
	TPR = 99.91%	TPR =90.60%
	FPR = 0.55%	FPR =36.79%
	AUC = 0.999	AUC =0.918

จากชุดข้อมูล KDDTrain+ เมื่อถูกแบ่งเป็น 2 กลุ่ม สามารถคำนวนหาค่าคุณลักษณะที่สำคัญ (Feature Importance) ที่มีผลต่อโมเดล โดยใช้ library ของ Feature Importance ใน Random Forest Sklearn โดยถ้าคุณลักษณะไหนมีค่า feature importance มาก แสดงว่ามีความสำคัญต่อการทำนายความถูกต้องของโมเดลเช่นกัน ค่าของ Feature Importance ในแต่ละคุณลักษณะถูกแสดงตามภาพประกอบ 19 และ 20



ภาพประกอบ 19 Feature Importance ของ KDDTrain+ feature train cluster 0



ภาพประกอบ 20 Feature Importance ของ KDDTrain+ feature train cluster 1

จากค่าของ Feature Importance ในแต่ละคุณลักษณะที่แสดงในภาพประกอบที่ 19 และ 20 แสดงรายการของค่าคุณลักษณะที่มากที่สุด 10 อันดับแรกของทั้งสองกลุ่ม ดังแสดงในตารางที่ 8 พบว่า cluster ทั้ง 2 กลุ่ม มีคุณลักษณะ dst_host_same_srv_rate ที่มีค่า Feature Importance สูงสุด ซึ่งมีค่าเท่ากับ 0.085 สำหรับ cluster 0 และ 0.079 สำหรับ cluster 1 โดยคุณลักษณะดังกล่าวมีความสำคัญกับการทำนายผลให้กับโมเดลที่ใช้ Random Forest สร้างโมเดล

ตาราง 8 ลำดับของค่า Feature Importance 10 อันดับแรก ของ Random Forest

Seq. No	Feature no.	Feature Name	Cluster 0	Feature no.	Feature Name	Cluster 1
1	23	dst_host_same_srv_rate	0.084602	23	dst_host_same_srv_rate	0.079218
2	2	dst_bytes	0.071585	2	dst_bytes	0.078885
3	1	src_bytes	0.069186	1	src_bytes	0.063323
4	74	flag_SF	0.052301	74	flag_SF	0.061813
5	12	count	0.050753	24	dst_host_diff_srv_rate	0.056525
6	18	same_srv_rate	0.048771	12	count	0.045768
7	24	dst_host_diff_srv_rate	0.045673	31	logged_in	0.044548
8	19	diff_srv_rate	0.044742	19	diff_srv_rate	0.042662
9	31	logged_in	0.041448	15	srv_serror_rate	0.039623
10	14	serror_rate	0.036546	14	serror_rate	0.039613

4.2 K-Means Clustering + Adaboost

จากนั้นผู้วิจัยได้นำชุดข้อมูลทั้ง 2 กลุ่ม เข้าอัลกอลิทึมแบบ Adaboost โดยในแต่ละชุดของข้อมูลจะถูก Tuning โดยใช้ hyperparameter ในการ Tuning และผู้วิจัยใช้ 5-fold cross-validation เพื่อเทรนและตรวจสอบประสิทธิภาพโมเดลของผู้วิจัยในแต่ละกลุ่ม ซึ่งพารามิเตอร์ที่ใช้สำหรับ Tuning มีรายละเอียดตามภาพประกอบที่ 21

```
param_grid = {
    'learning_rate':[0.25,0.5,0.75,1],
    'n_estimators':[100,250,500,650],
     }
```

ภาพประกอบ 21 พารามิเตอร์ที่ใช้สำหรับการทำ Tuning ในอัลกอลิทึมแบบ Adaboost

เนื่องจากพารามิเตอร์ที่ใช้ทำการ Tuning ของ Adaboost มีน้อยผู้วิจัยจึงสามารถทำการ Tuning ได้เพียง 2 พารามิเตอร์ จากภาพประกอบที่ 21 สามารถได้อธิบายพารามิเตอร์ที่ใช้สำหรับ Tuning ได้ โดยมีรายละเอียดดังนี้

n_estimators: จำนวนของต้นไม้ทั้งหมดที่ใช้ในโมเดล

learning_rate: พารามิเตอร์นี้ช่วยลดปัญหา over-fitting ซึ่งจะควบคุมการหดตัวของขั้นตอนและปัจจัยการให้น้ำหนักเพื่อให้ถูกต้องเมื่อมีการเพิ่มต้นไม้ใหม่ลงในแบบจำลอง

ในแต่ละชุดของการทำ hyperparameter ผู้วิจัยใช้ 5-fold cross-validation เพื่อตรวจสอบประสิทธิภาพโมเดลของผู้วิจัยในแต่ละกลุ่ม หลังจากการทำ Tuning ทำให้ได้ผลคะแนนสูงสุดของแต่ละพารามิเตอร์ที่จะนำมาสร้างโมเดลดังแสดงในตารางที่ 9

ตาราง 9 จำนวนค่า Best Score ของ hyperparameter ของ Adaboost ในแต่ละ Cluster

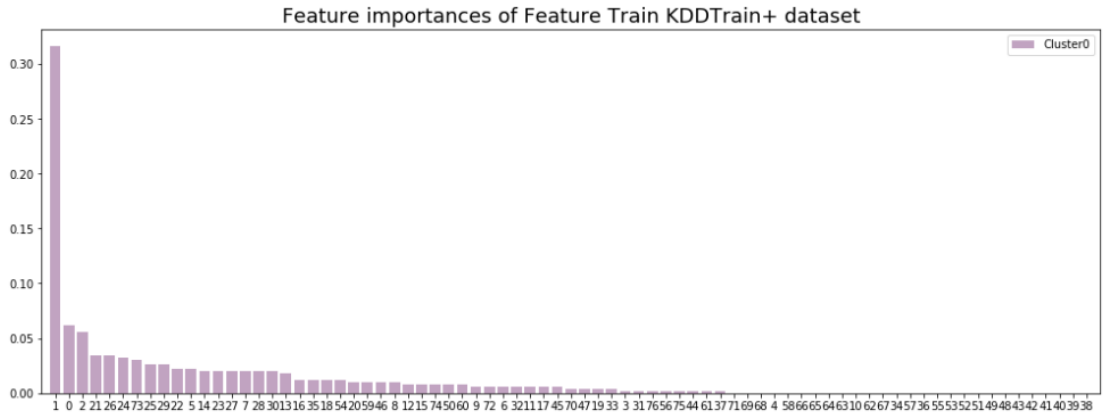
Cluster	n_estimators	learning_rate
0	500	1
1	500	1

จากผล Best Score ในตารางที่ 9 เมื่อนำมาทดสอบร่วมกับชุดข้อมูล KDDTrain+ และ KDDTest+ ทำให้ได้ประสิทธิภาพของโมเดลดังแสดงในตารางที่ 10 ซึ่งจากการแบ่งกลุ่มของข้อมูลที่ K=2 ในการสร้างและทดสอบโมเดลบนชุดข้อมูล KDDTrain+ ให้ค่าความแม่นยำเท่ากับ 99.59% ค่า True Positive Rate เท่ากับ 99.78% ค่า False Positive Rate เท่ากับ 0.57% ค่า Area Under The Curve เท่ากับ 0.999 และทดสอบโมเดลกับชุดข้อมูล KDDTest+ ให้ค่าความแม่นยำเท่ากับ 72.90% ค่า True Positive Rate เท่ากับ 88.16% ค่า False Positive Rate เท่ากับ 37.08% และค่า Area Under The Curve เท่ากับ 0.877

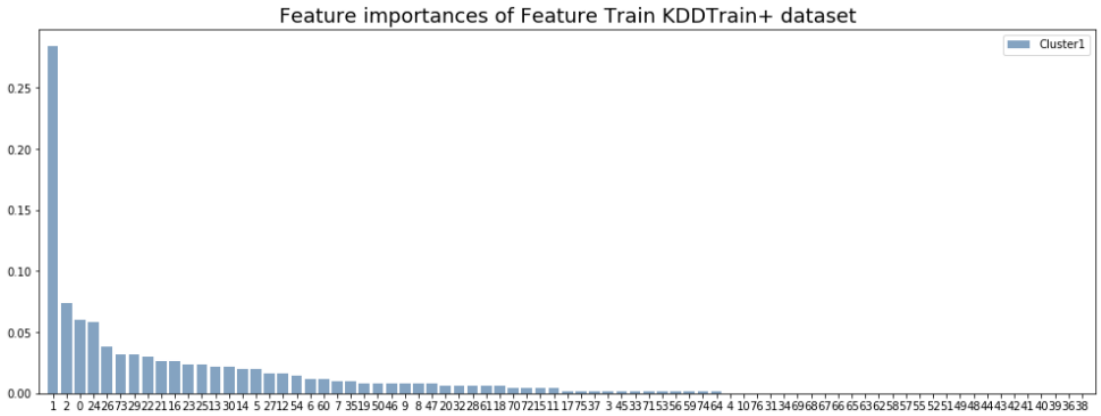
ตาราง 10 ค่าของ Accuracy, TPR ,FPR และ AUC ใน K-Means Clustering กับ Adaboost Classifier

Number of K in K-Means	**KDDTrain+**	**KDDTest+**
K-Means K=2	Accuracy = 99.59%	Accuracy = 72.90%
	TPR = 99.78%	TPR = 88.16%
	FPR = 0.57%	FPR = 37.08%
	AUC = 0.999	AUC = 0.877

จากชุดข้อมูล KDDTrain+ สามารถคำนวนหาค่าคุณลักษณะที่สำคัญ (Feature Importance) ที่มีผลต่อโมเดล โดยใช้ library ของ Feature Importance ใน Adaboost Sklearn ถ้าคุณลักษณะไหนมีค่า Feature Importance มากจะมีความสำคัญต่อการทำนายความแม่นยำของโมเดลเช่นกัน ค่าของ Feature Importance ในแต่ละ feature train ของ Cluster ทั้งสองกลุ่ม สามารถแสดงได้ดังภาพประกอบที่ 22 และ 23



ภาพประกอบ 22 ค่า Feature Importance ของ cluster 0 ใน Adaboost อัลกอลิทึม



ภาพประกอบ 23 ค่า Feature Importance ของ cluster 1 ใน Adaboost อัลกอลิทึม

จากค่าของ Feature Importance ในแต่ละคุณลักษณะที่แสดงในภาพประกอบที่ 22 และ 23 แสดงรายการของค่าคุณลักษณะที่มากที่สุด 10 อันดับแรกของทั้งสองกลุ่ม ดังแสดงในตารางที่ 11 พบว่า cluster ทั้ง 2 กลุ่ม มีคุณลักษณะ src_bytes ที่มีค่า Feature Importance สูงสุด ซึ่งมีค่าเท่ากับ 0.316 สำหรับ cluster 0 และ 0.284 สำหรับ cluster 1 โดยคุณลักษณะดังกล่าวมีความสำคัญกับการทำนายผลให้กับโมเดลที่ใช้ Adaboost สร้างโมเดล

ตาราง 11 ลำดับของค่า Feature Importance 10 อันดับแรกของ Adaboost อัลกอลิทึม

Seq. No	Feature no.	Feature Name	Cluster 0	Feature no.	Feature Name	Cluster 1
1	1	src_bytes	0.316	1	src_bytes	0.284
2	0	duration	0.062	2	dst_bytes	0.074
3	2	dst_bytes	0.056	0	duration	0.06
4	21	dst_host_count	0.034	24	dst_host_diff_srv_rate	0.058
5	26	dst_host_srv_diff_host_rate	0.034	26	dst_host_srv_diff_host_rate	0.038
6	24	dst_host_diff_srv_rate	0.032	37	service_auth	0.032
7	73	flag_S0	0.03	29	dst_host_rerror_rate	0.032
8	25	dst_host_same_src_port_rate	0.026	22	dst_host_srv_count	0.03
9	29	dst_host_rerror_rate	0.026	21	dst_host_count	0.026
10	22	dst_host_srv_count	0.022	16	rerror_rate	0.026

4.3 K-Means Clustering + XGBoost

ในการสร้างโมเดลโดยใช้ อัลกอลิทึมแบบ XGBoost ผู้วิจัยได้ใช้พารามิเตอร์สำหรับ tuning ของ XGBoost เพื่อหาค่า คะแนนที่ดีที่สุด (best score) ดังแสดงในภาพประกอบที่ 24

```
param_grid = {
        'n_estimators': [100, 200],
        'max_depth': [5, 10, 15, 20],
        'learning_rate': [0.001, 0.01, 0.1, 0.2, 0.3],
        'subsample': [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
        'colsample_bytree': [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
        'colsample_bylevel': [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
        'min_child_weight': [0.4, 0.5, 1.0, 3.0, 5.0, 7.0, 10.0],
        'gamma': [0, 0.25, 0.5, 1.0],
        'reg_lambda': [0.1, 1.0, 5.0, 10.0, 50.0, 100.0]
        }
```

ภาพประกอบ 24 แสดงรายการพารามิเตอร์ที่ใช้สำหรับการทำ Tuning ของ XGBoost อัลกอลิทึม

จากภาพประกอบที่ 24 ผู้วิจัยได้อธิบายพารามิเตอร์ที่ใช้ tuning โดยมีรายละเอียดดังนี้

n_estimators: จำนวนของต้นไม้ทั้งหมดที่ใช้ในโมเดล

max_depth: จำนวนสูงสุดของลำดับชั้นของต้นไม้ ค่าที่สูงขึ้นจะทำให้รูปแบบที่ซับซ้อนมากขึ้น

learning_rate: พารามิเตอร์นี้ช่วยลดปัญหา over-fitting ซึ่งจะควบคุมการหดตัวของขั้นตอนและปัจจัยการให้น้ำหนักเพื่อให้ถูกต้องเมื่อมีการเพิ่มต้นไม้ใหม่ลงในแบบจำลอง

subsample: ส่วนหนึ่งของตัวอย่างที่จะถูกสุ่มเลือกสำหรับแต่ละต้น

colsample_bytree: ส่วนของคอลัมน์ที่จะถูกสุ่มเลือกสำหรับแต่ละต้น

colsample_bylevel: อัตราส่วนของลำดับของคอลัมน์สำหรับการแบ่งแต่ละคุณลักษณะในแต่ละชั้น

min_child_weight: ผลรวมขั้นต่ำของน้ำหนักในข้อมูลทั้งหมดที่ต้องใช้ในใบไม้ ซึ่งพารามิเตอร์นี้ถูกใช้เพื่อควบคุม over-fitting ของโมเดล

gamma: การลดการสูญเสียขั้นต่ำที่ต้องถูกแยกออก

reg_lambda: ถูกใช้จัดการในส่วนหนึ่งของกระบวนการแนะนำข้อมูลเพิ่มเติมเพื่อแก้ปัญหาที่ไม่ถูกต้องหรือเพื่อป้องกัน Over-Fitting ของฟังค์ชันการสูญเสียใน XGBoost

ในแต่ละชุดของการทำ hyperparameter ผู้วิจัยใช้ 5-fold cross-validation เพื่อตรวจสอบประสิทธิภาพโมเดลของผู้วิจัยในแต่ละกลุ่ม โดยค่า best score ของแต่ละพารามิเตอร์หลังจากการทำ Tuning จะได้ผลคะแนนสูงสุดที่จะนำมาสร้างโมเดล ดังแสดงในตารางที่ 12

ตาราง 12 จำนวนค่า Best Score ของ hyperparameter ของ XGBoost ในแต่ละ Cluster

Cluster	sub sample	reg_ lambda	n_ estimators	min_child _weight	max _depth	learning _rate	gamma	Colsample _bytree	colsample _bylevel
0	0.6	5	200	0.4	5	0.2	0	1	0.6
1	0.6	5	200	0.4	5	0.2	0	1	0.6

จาก Best Score ในตารางที่ 12 เมื่อนำมาทดสอบร่วมกับชุดข้อมูล KDDTrain+ และ KDDTest+ สามารถวัดประสิทธิภาพของโมเดล โดยจากการแบ่งกลุ่มข้อมูลที่ K=2 ในการสร้างและทดสอบโมเดลบนชุดข้อมูล KDDTrain+ ให้ค่าความแม่นยำเท่ากับ 99.85% ค่า True Positive Rate เท่ากับ 99.87% ค่า False Positive Rate เท่ากับ 0.18% ค่า Area Under The Curve เท่ากับ 0.998 และทดสอบโมเดลกับชุดข้อมูล KDDTest+ ให้ค่าความแม่นยำเท่ากับ 84.41% ค่า True Positive Rate เท่ากับ 86.36% ค่า False Positive Rate เท่ากับ 18.20% และค่า Area Under The Curve เท่ากับ 0.923 ดังแสดงในตารางที่ 13

ตาราง 13 ค่าของ Accuracy, TPR ,FPR และ AUC ใน K-Means Clustering กับ XGboost Classifier

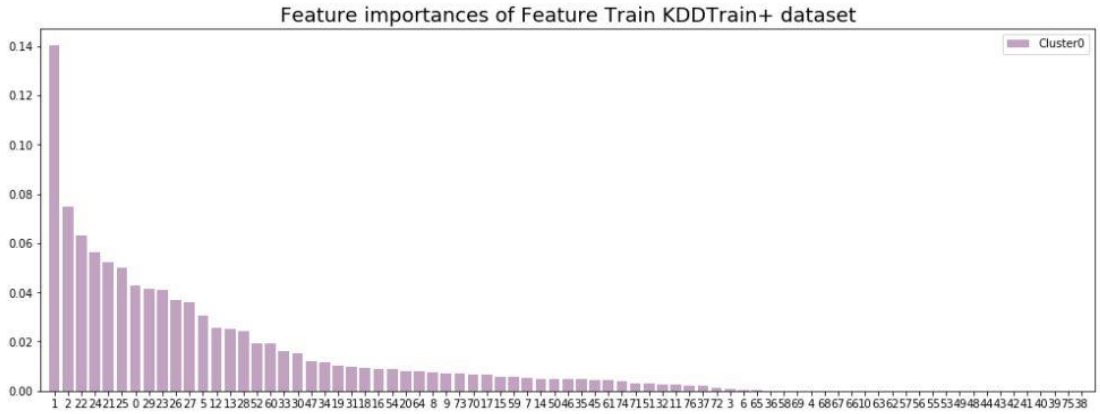
Number of K in K-Means	KDDTrain+	KDDTest+
K-Means K=2	Accuracy = 99.85%	Accuracy =84.41%
	TPR = 99.87%	TPR = 86.36%
	FPR = 0.18%	FPR = 18.20%
	AUC = 0.998	AUC = 0.923

จากชุดข้อมูล KDDTrain+ เมื่อถูกแบ่งเป็น 2 กลุ่ม สามารถคำนวนหาค่าคุณลักษณะที่สำคัญ (Feature Importance) ที่มีผลต่อโมเดล โดยใช้ library ของ Feature Importance ใน Sklearn กับ XGBoost โดยถ้าคุณลักษณะไหนมีค่า feature importance มาก แสดงว่ามีความสำคัญต่อการทำนายความถูกต้องของโมเดลเช่นกัน ค่าของ Feature Importance ในแต่ละคุณลักษณะถูกแสดงในตารางที่ 14 ซึ่งจะแสดงค่าของ Feature Importance ที่มากที่สุด 10 อันดับแรกของทั้งสองกลุ่ม จากตารางที่ 14 ทั้งสองกลุ่มแสดงรูปแบบคล้ายกัน ซึ่งคุณลักษณะ src_bytes ให้ความสำคัญสูงสุดสำหรับทั้งสองกลุ่ม ซึ่งมีค่าเท่ากับ 0.140 สำหรับ cluster 0 และ 0.133 สำหรับ cluster 1 โดยคุณลักษณะดังกล่าวมีความสำคัญกับการทำนายผลให้กับโมเดลที่ใช้ XGBoost สร้างโมเดล

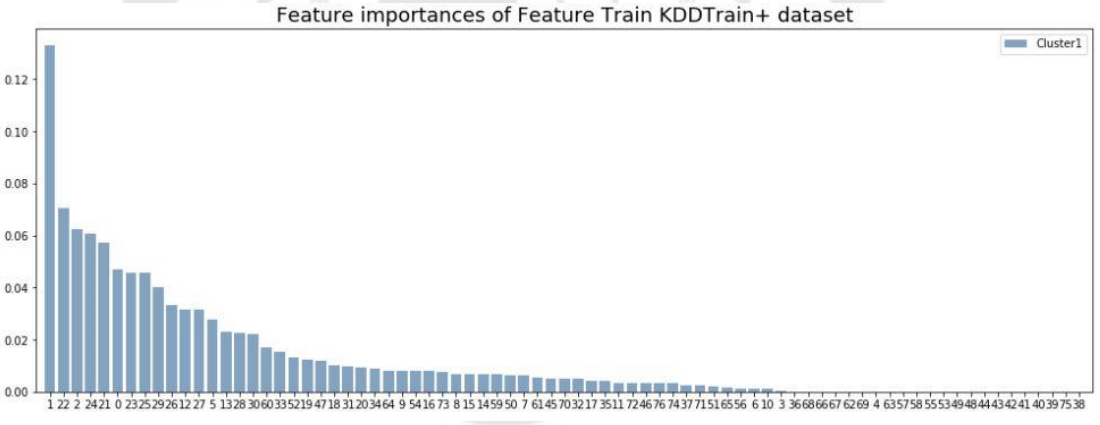
ตาราง 14 ลำดับของค่า Feature Importance 10 อันดับแรกของ XGBoost

Seq. No	Feature no.	Feature Name	Cluster 0	Feature no.	Feature Name	Cluster 1
1	1	src_bytes	0.140	1	src_bytes	0.133
2	2	dst_bytes	0.075	22	dst_host_srv_count	0.070
3	22	dst_host_srv_count	0.063	2	dst_bytes	0.062
4	24	dst_host_diff_srv_rate	0.056	24	dst_host_diff_srv_rate	0.061
5	21	dst_host_count	0.052	21	dst_host_count	0.057
6	25	dst_host_same_src_port_rate	0.050	0	duration	0.047
7	0	duration	0.043	23	dst_host_same_srv_rate	0.046
8	29	dst_host_rerror_rate	0.041	25	dst_host_same_src_port_ra	0.046
9	23	dst_host_same_srv_rate	0.041	29	dst_host_rerror_rate	0.040
10	26	dst_host_srv_diff_host_rate	0.037	26	dst_host_srv_diff_host_rate	0.033

จากที่อธิบายในตารางที่ 14 ผู้วิจัยได้แสดงกราฟ Feature Importance สำหรับ XGBoost ทั้ง 2 cluster ได้แสดงในภาพประกอบที่ 25 และ 26



ภาพประกอบ 25 Feature Importance ของ KDDTrain+ cluster 0 โดยใช้ XGBoost



ภาพประกอบ 26 Feature Importance ของ KDDTrain+ cluster 1 โดยใช้ XGBoost

จากตารางที่ 13 พบว่าอัลกอลิทึมแบบ XGBoost ให้ค่าความแม่นยำสูงสุด โดยสามารถเปรียบเทียบค่าของ Accuracy, TPR ,FPR และ AUC ของโมเดลต้นไม้ทั้ง 3 ได้ ดังแสดงในตารางที่ 15

ตาราง 15 เปรียบเทียบค่าของ Accuracy, TPR ,FPR และ AUC ของไฮบริดโมเดลที่ K-Means =2 ทำงานโมเดลแบบต้นไม้

Classifiers	Number of K	KDDTrain+	KDDTest+
XGBoost	**K-Means K=2**	**Accuracy = 99.85%**	**Accuracy =84.41%**
		TPR = 99.87%	**TPR = 86.36%**
		FPR = 0.18%	**FPR = 18.20%**
		AUC = 0.998	**AUC = 0.923**
Random Forest	K-Means K=2	Accuracy = 99.67%	Accuracy = 73.63%
		TPR = 99.91%	TPR =90.60%
		FPR = 0.55%	FPR =36.79%
		AUC = 0.999	AUC =0.918
Adaboost	K-Means K=2	Accuracy = 99.59%	Accuracy = 72.90%
		TPR = 99.78%	TPR = 88.16%
		FPR = 0.57%	FPR = 37.08%
		AUC = 0.999	AUC = 0.877

จากตารางที่ 15 พบว่าไฮบริดโมเดลที่ K-Means = 2 ทำงานร่วมกับ XGBoost Classifier ให้ค่าความแม่นยำสูงสุด เนื่องจาก XGBoost มีจำนวนของ Hyperparameter ที่มีความหลากหลายกว่าโมเดลแบบอื่นและในการทำ tuning ได้เพิ่มพารามิเตอร์ในส่วนของ learning rate และ reg_lambda ซึ่งมีคุณสมบัติช่วยควบคุมและลด over fitting ในโมเดล ส่งผลให้ XGBoost Classifier ให้ผลลัพธ์ที่ดีที่สุดเมื่อเปรียบเทียบกับโมเดลแบบต้นไม้แบบอื่น

เมื่อนำค่า K ของ K-Means ที่ K = 2 มาเปรียบเทียบกับค่า K ที่ 4, 6, 8, 10 เพื่อทดสอบแนวทางการเลือกค่า K (Choosing K) โดยใช้เทคนิคข้อศอก (elbow method) ดังที่กล่าวไปในหัวข้อที่ 3.2 เพื่อที่จะพิสูจน์ว่าเทคนิค ข้อศอก สามารถกำหนดค่า K ที่เหมาะสมได้หรือไม่ ซึ่งผลของการทดสอบโมเดลกับการกำหนดค่า K ที่ 4, 6, 8, 10 สามารถแสดงได้ดังตารางที่ 16

ตาราง 16 ค่าของ Accuracy, TPR ,FPR และ AUC ในค่า K ที่เท่ากับ 2, 4, 6, 8, 10 ใน K-Means Clustering กับ XGBoost Classifier

Number of K in K-Means	**KDDTrain+**	**KDDTest+**
K-Means K=10	Accuracy = 99.77%	Accuracy = 83.56%
	TPR = 99.83%	TPR =83.49%
	FPR = 0.28%	FPR =16.34%
	AUC = 0.997	AUC =0.828
K-Means K=8	Accuracy = 99.77%	Accuracy = 83.23%
	TPR = 99.86%	TPR = 83.89%
	FPR = 0.30%	FPR = 17.72%
	AUC = 0.998	AUC = 0.826
K-Means K=6	Accuracy = 99.75%	Accuracy = 80.32%
	TPR = 99.79%	TPR = 78.37%
	FPR = 0.28%	FPR = 15.99%
	AUC = 0.998	AUC = 0.788
K-Means K=4	Accuracy = 99.83%	Accuracy = 83.69%
	TPR = 99.84%	TPR = 83.90%
	FPR = 0.18%	FPR = 16.62%
	AUC = 0.998	AUC = 0.830
K-Means K=2	**Accuracy = 99.85%**	**Accuracy =84.41%**
	TPR = 99.87%	**TPR = 86.36%**
	FPR = 0.18%	**FPR = 18.20%**
	AUC = 0.998	**AUC = 0.923**
No Clustering	Accuracy = 99.90%	Accuracy = 82.66%
	TPR = 99.92%	TPR = 87.32%
	FPR = 0.11%	FPR = 22.60%
	AUC = 0.999	AUC = 0.829

จากตารางที่ 16 ผู้วิจัยทำเปรียบเทียบประสิทธิภาพของโมเดล โดยกำหนดค่า K ที่ 2, 4, 6, 8, 10 และไม่มีการทำ cluster พบว่าแบบโมเดลที่ค่า K=2 ให้ผลลัพท์ที่ดีที่สุด

จากการวิจัยพบว่าประสิทธิภาพของโมเดลที่เรานำเสนอในด้านความแม่นยำเมื่อเปรียบเทียบกับแบบจำลอง RNN [9] และตัวจำแนกประเภทต้นไม้ (Random Forest, Adaboost, XGBoost) ในตารางที่ 17 พบว่าไฮบริดโมเดลที่ K-Means+XGBoost ได้รับความแม่นยำสูงสุดเมื่อเทียบกับโมเดลอื่น ทั้งชุดข้อมูล KDDTrain+ และ KDDTest+ โดยใช้การจัดกลุ่ม (Cluster) ที่ K=2 ทำงานร่วมกับการจำแนกแบบ XGBoost โดยใช้การ Tuning แบบ RandomizedSearchCV ซึ่งข้อมูลจะถูกเทรนในกลุ่มของข้อมูลที่ถูกจัดกลุ่มโดยมีลักษณะคล้ายกันในแต่ละ Cluster ซึ่งจะช่วยเพิ่มประสิทธิภาพการตรวจจับรูปแบบของเพ็คเกตที่ไม่รู้จักมาก่อน เมื่อเทียบกับการเทรนข้อมูลที่ไม่มีการแบ่งกลุ่มของข้อมูล ซึ่งอาจเป็นหนึ่งในสาเหตุสำคัญที่ทำให้โมเดลของผู้วิจัยมีค่าความแม่นยำที่สูงกว่าโมเดล RNN

ตาราง 17 เปรียบเทียบค่าแม่นยำของโมเดลต้นไม้กับโมเดล RNN

Model	**KDDTrain+**	**KDDTest+**
RNN	99.81%	83.28%
K-Means+XGBoost	**99.85%**	**84.41%**
K-Means+Random Forest	99.67%	73.63%
K-Means+Adaboost	99.61%	72.90%

นอกจากนี้การเลือกใช้ Feature Selection แบบ AR ในการสร้างโมเดลมีความสำคัญเช่นกัน ซึ่งเกณฑ์ของค่า AR ที่เลือกใช้จะช่วยลดจำนวนของคุณลักษณะให้เหลือ 77 คุณลักษณะ จากทั้งหมด 122 คุณลักษณะ หรือ 63.11% จากคุณลักษณะของชุดข้อมูลแบบเทรนทั้งหมด เพื่อให้เกิดประสิทธิภาพที่ดีเทียบเท่ากับการใช้คุณลักษณะ 122 คุณลักษณะ ซึ่งใช้เวลาในการเทรนมากกว่า รายละเอียดเพิ่มเติมสามารถดูได้จากภาคผนวก

บทที่ 5
สรุปผลการวิจัย อภิปรายผล และข้อเสนอแนะ

ในการวิจัยเรื่องการวิเคราะห์การบุกรุกบนระบบเครือข่ายโดยใช้เทคนิคการเรียนรู้ของเครื่อง ผู้วิจัยได้ทำการวัดประสิทธิภาพของโมเดล โดยใช้ค่า Accuracy ค่า True Positive Rate ค่า False Positive Rate และพื้นที่ใต้โค้ง Area Under The Curve เป็นเกณฑ์ในการวัดประสิทธิภาพของโมเดล จากผลที่ได้การดำเนินงานแล้ว สามารถสรุปผลการดำเนินงานได้ดังต่อไปนี้

5.1 สรุปผลการวิจัย

ในงานวิจัยนี้นำเสนอการสร้างโมเดลแบบไฮบริดโดยใช้ unsupervised learning (K-Means Clustering) ทำงานร่วมกับ supervise learning (XGBoost, Random Forest, Adaboost) ผู้วิจัยได้ทำการทดสอบประสิทธิภาพโมเดลกับชุดข้อมูล NSL-KDD ทั้ง KDDTrain+ และ KDDTest+ และใช้แนวทางของ Feature selection แบบ AR มาใช้ลดจำนวนของคุณลักษณะของชุดข้อมูล NSL-KDD และหลังจากทำ K-Means Clustering แล้ว ผู้วิจัยมีการปรับค่าพารามิเตอร์ของโมเดลในแต่ละกลุ่ม (Tuning) เพื่อให้ได้ค่าพารามิเตอร์ที่เหมาะสมของโมเดลแต่ละกลุ่ม ผู้วิจัยได้ทำการทดสอบโมเดลที่สร้างขึ้นกับชุดข้อมูล KDDTest+ ซึ่งได้ค่าความแม่นยำสูงสุดเท่ากับ 84.41% และมีอัตราการตรวจจับ (detection rate) เท่ากับ 86.36% อัตราการเตือนภัยผิดพลาด (false alarm rate)เท่ากับ 18.20% และ AUC เท่ากับ 0.923 นอกจากนี้ประสิทธิภาพของโมเดลที่ผู้วิจัยนำเสนอในแง่ของความแม่นยำให้ผลดีกว่าเมื่อเปรียบเทียบกับ RNN-based deep neural network. จากการทำ Feature selection ทำให้โมเดลของเราใช้คุณลักษณะเพียง 77 คุณลักษณะจากคุณลักษณะทั้งหมด 122 คุณลักษณะ เพื่อให้บรรลุเป้าหมายการปฏิบัติงานของโมเดลเทียบกับชุดข้อมูลเทรนแบบเต็มรูปแบบ

5.2 อภิปรายผล

จากการทดลองสร้างโมเดลแบบไฮบริดโดยใช้ K-Means clustering ทำงานร่วมกับ Tree based classification ทั้ง 3 แบบ พบว่าไฮบริดโมเดลแบบ K-Means+XGBoost ให้ผลความแม่นยำสูงสุดเมื่อเปรียบเทียบกับโมเดลต้นไม้ด้วยกัน และเมื่อนำผลไปเปรียบเทียบกับแบบจำลอง RNN ซึ่งเป็น Baseline โมเดล พบว่าไฮบริดโมเดลที่ K-Means+XGBoost มีความแม่นยำสูงกว่าเช่นกัน ทั้งชุดข้อมูล KDDTrain+ และ KDDTest+ ซึ่งเกิดจากการจัดกลุ่ม (Cluster) ที่ K=2 ทำงานร่วมกับการจำแนกแบบ XGBoost โดยใช้การ Tuning แบบ RandomizedSearchCV และในพารามิเตอร์ของ XGBoost ที่ใช้ในการทำ tuning นั้น มีส่วนของ learning rate และ reg_lambda ซึ่งมีคุณสมบัติช่วยควบคุมและลดการเกิด over fitting ในโมเดล ในส่วนของ K-Means นั้นยังข้อดีอีกประการหนึ่ง คือข้อมูลที่มีลักษณะคล้ายกันจะถูกจับกลุ่มให้อยู่ด้วยกัน ซึ่งจะช่วยเพิ่มประสิทธิภาพการตรวจจับรูปแบบของแพ็คเกตที่ไม่รู้จักมาก่อน zero- day เมื่อเทียบกับการเทรนข้อมูลที่ไม่มีการแบ่งกลุ่มของข้อมูล ซึ่งอาจเป็นหนึ่งในสาเหตุสำคัญที่ทำให้โมเดลของผู้วิจัยมีค่าความแม่นยำที่สูงกว่าโมเดล RNN

5.3 ข้อเสนอแนะ

งานวิจัยนี้นำเสนอการทำงานของ Machine Learning ในการจำแนกรูปแบบของข้อมูลบนเครือข่ายเป็น Normal และ Anomaly บนชุดข้อมูล NSL-KDD ซึ่งการจำแนกเป็นแบบ 2 คลาส นอกจากนี้ชุดข้อมูล NSL-KDD ยังสามารถจำแนกรูปแบบข้อมูลเป็นแบบ 5 คลาสได้อีกด้วย คือ Normal, DoS, Probe, R2L และ U2R หากต้องการทำงานวิจัยในการจำแนกการโจมตีบนเครือข่ายแบบ 5 คลาส สามารถต่อยอดจากงานวิจัยนี้ได้ และในชุดข้อมูล NSL-KDD ยังมีชุดข้อมูล KDDTest-21 ที่มี record จำนวน 11,850 record ซึ่งเป็นชุดข้อมูลที่ถูกสร้างขึ้นมา โดยข้อมูลถูกแยกจาก KDDTest+ ซึ่งใช้หลักการสร้างโมเดลจำนวน 21 โมเดลมาทำนายชุดข้อมูล KDDTest+ ว่า record ไหนเป็นรูปแบบ Normal และ record ไหนเป็นรูปแบบเป็น Anomaly จากนั้นตัดจำนวนของ record ที่โมเดลทั้ง 21 ตัวทำนายถูกออก เหลือไว้เฉพาะ record ที่ทำนายผิดและนำมาสร้างเป็นชุดข้อมูล KDDTest-21 ซึ่งชุดข้อมูลดังกล่าวเป็นชุดข้อมูลที่เหมาะกับการใช้ทดสอบการโจมตีแบบ Zero-day เพราะโมเดลที่ถูกสร้างจากชุดข้อมูล KDDTrain+ เมื่อนำมาทดสอบกับ KDDTest-21 จะได้ค่าความแม่นยำที่ต่ำ เนื่องจาก KDDTest-21 มีข้อมูลที่โมเดลสร้างจาก KDDTrain+ ไม่รู้จัก ซึ่งสามารถนำงานวิจัยที่ผู้วิจัยนำเสนอมาต่อยอดกับ ชุดข้อมูล KDDTest-21 ได้เช่นกัน

บรรณานุกรม

1. Tavallaee, M., et al. *A detailed analysis of the KDD CUP 99 data set*. in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*. 2009. IEEE.
2. Chen, T. and C. Guestrin. *Xgboost: A scalable tree boosting system*. in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016. ACM.
3. Yao, H., Y. Liu, and C. Fang, *An Abnormal Network Traffic Detection Algorithm Based on Big Data Analysis.* International Journal of Computers, Communications & Control, 2016. **11**(4).
4. Ji, S.-Y., et al., *A multi-level intrusion detection method for abnormal network behaviors.* Journal of Network and Computer Applications, 2016. **62**: p. 9-17.
5. Buczak, A.L. and E. Guven, *A survey of data mining and machine learning methods for cyber security intrusion detection.* IEEE Communications Surveys & Tutorials, 2016. **18**(2): p. 1153-1176.
6. Choi, S.-H. and H.-S. Chae. *Feature Selection using Attribute Ratio in NSL-KDD data*. in *International Conference Data mining, Civil and Mechanical Engineering (ICDMSME'2014), Bali (Indonesia), Fecb*. 2014.
7. Agrawal, S. and J. Agrawal, *Survey on anomaly detection using data mining techniques.* Procedia Computer Science, 2015. **60**: p. 708-713.
8. Wang, S., P. Dong, and Y. Tian, *A Novel Method of Statistical Line Loss Estimation for Distribution Feeders Based on Feeder Cluster and Modified XGBoost.* Energies, 2017. **10**(12): p. 2067.
9. Yin, C., et al., *A deep learning approach for intrusion detection using recurrent neural networks.* IEEE Access, 2017. **5**: p. 21954-21961.
10. Bholowalia, P. and A. Kumar, *EBK-means: A clustering technique based on elbow method and k-means in WSN.* International Journal of Computer Applications, 2014. **105**(9).

ภาคผนวก

งานวิจัยนี้พัฒนาโมเดลโดยใช้ภาษา Python เวอร์ชั่น 3 ซึ่งสามารถดู code ทั้งหมดได้จาก https://drive.google.com/drive/folders/1 hbSdytkbaALLNJqb-Xsn9 rp6 6 lxGaQgj?usp =sharing ทั้งนี้ผู้วิจัยได้แสดงตัวอย่างบาง code ที่สำคัญเพื่อใช้ประกอบงานวิจัย โดยมีรายละเอียดดังนี้

1. Code PCA ใช้สำหรับนำชุดข้อมูลมาลดรูปเพื่อ plot กราฟ วิเคราะห์ตำแหน่งรูปแบบของชุดข้อมูลเพื่อแยก normal และ anomaly ตามที่กล่าวไว้ในหัวข้อที่ 3.2 ดังแสดงในภาพประกอบ 27

```
import matplotlib.pyplot as plt
X_reduced = PCA(n_components=2).fit_transform(df_train_std_PCA)

# Plot the training points
plt.figure(2, figsize=(10, 8))
plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=y, cmap=plt.cm.Set1,
            edgecolor='k')
plt.title('Plot PCA Analyst NSL_KDD 2 Label')
plt.xlabel('Component of PCA KDDTrain dataset X axis', fontsize=16)
plt.ylabel('Component of PCA KDDTrain dataset y axis', fontsize=16)
plt.show()
```

ภาพประกอบ 27 Code PCA สำหรับ plot กราฟวิเคราะห์ตำแหน่งข้อมูล

2. Code สำหรับเลือกค่า K ให้เหมาะสมกับโมเดล เพื่อนำค่า K ไปใช้ในการแบ่งกลุ่มชุดข้อมูลที่มีลักษณะคล้ายกันให้อยู่ในกลุ่มเดียวกัน ตามที่กล่าวไว้ในหัวข้อที่ 3.2 ดังแสดงในภาพประกอบ 28

```
from sklearn.cluster import KMeans
from scipy import cluster
import matplotlib.pyplot as plt
from matplotlib import pyplot
sse = {}
for k in range(1, 15):
    kmeans = KMeans(n_clusters=k, n_init=25, max_iter=100, tol=0.0001, random_state=3425).fit(df_train_std)
    df_train_std["clusters"] = kmeans.labels_
    #print(data["clusters"])
    sse[k] = kmeans.inertia_
plt.figure(2, figsize=(10, 8))
plt.plot(list(sse.keys()), list(sse.values()),'bo')
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Number of cluster", fontsize=18)
plt.ylabel("SSE", fontsize=18)
plt.grid(True)
plt.show()
```

ภาพประกอบ 28 Code สำหรับเลือกค่า K

3. Code สำหรับการทำ K-Means Clustering หลังจากได้ค่า K=2 ดังแสดงในภาพที่ 29 เป็นการทำ cluster กับชุดข้อมูล KDDtrain+ ซึ่งพบว่า cluster 0 มีจำนวนข้อมูลใน cluster เท่ากับ 47377 และ cluster 1 มีจำนวนข้อมูลใน cluster เท่ากับ 78596 ตามที่กล่าวไว้ในหัวข้อที่ 3.2

```
from sklearn.cluster import KMeans
k = 2
km = KMeans(n_clusters = k, n_init=25, max_iter=100, tol=0.0001, random_state=3425)
t0 = time()
km.fit(df_train_std)
tt = time()-t0
print ("Clustered in {} seconds".format(round(tt,3)))
pandas.Series(km.labels_).value_counts()
```

```
Clustered in 7.771 seconds

1    78596
0    47377
dtype: int64
```

```
labels = km.labels_
label_names = list(map(
    lambda x: pandas.Series([labels[i] for i in range(len(km.labels_)) if km.labels_[i]==x]),
    range(k)))
```

ภาพประกอบ 29 ตัวอย่าง Code สำหรับการทำ K Means Clustering

4. Code สำหรับหาค่า AR ในคุณลักษณะประเภท numeric ดังแสดงในภาพที่ 30 ตามที่กล่าวไว้ในหัวข้อที่ 3.2

```
# AR value Numeric
df_train_avg_all = df_train_num.mean()/511
df_normal = df_train_avg[(df_train_avg['label5-index'] == 0)]
df_normal_avg = df_normal[Numberic].mean()/511
df_dos = df_train_avg[(df_train_avg['label5-index'] == 1)]
df_dos_avg = df_dos[Numberic].mean()/511
df_probe = df_train_avg[(df_train_avg['label5-index'] == 2)]
df_probe_avg = df_probe[Numberic].mean()/511
df_R2L = df_train_avg[(df_train_avg['label5-index'] == 3)]
df_R2L_avg = df_R2L[Numberic].mean()/511
df_U2R = df_train_avg[(df_train_avg['label5-index'] == 4)]
df_U2R_avg = df_U2R[Numberic].mean()/511

df_avg = pd.concat([df_normal_avg, df_dos_avg, df_probe_avg, df_R2L_avg, df_U2R_avg, df_train_avg_all], axis=1)
df_avg.rename(columns={0:'Normal_AVG', 1:'DoS_AVG', 2:'Probe_AVG', 3:'R2L_AVG', 4:'U2R_AVG', 5:'Total_AVG'}, inplace=True)

df_AR_Normal = df_avg['Normal_AVG']/df_avg['Total_AVG']
df_AR_DoS = df_avg['DoS_AVG']/df_avg['Total_AVG']
df_AR_Probe = df_avg['Probe_AVG']/df_avg['Total_AVG']
df_AR_R2L = df_avg['R2L_AVG']/df_avg['Total_AVG']
df_AR_U2R = df_avg['U2R_AVG']/df_avg['Total_AVG']

df_AR = pd.concat([df_AR_Normal, df_AR_DoS, df_AR_Probe, df_AR_R2L, df_AR_U2R], axis=1)
df_AR.rename(columns={0:'Normal_AR', 1:'DoS_AR', 2:'Probe_AR', 3:'R2L_AR', 4:'U2R_AR'}, inplace=True)
df_ARi = df_AR.max(axis=1)
```

ภาพประกอบ 30 Code สำหรับหาค่า AR ในคุณลักษณะประเภท numeric

5. Code สำหรับหาค่า AR ในคุณลักษณะประเภท binary ดังแสดงในภาพที่ 31

```
# AR value Binary
col_name = ['row','count0_0', 'count0_1','div0','count1_0', 'count1_1','div1','count2_0', 'count2_1', 'div2'
                        ,'count3_0', 'count3_1', 'div3','count4_0', 'count4_1', 'div4']
df = pd.DataFrame(columns = col_name)
for i in range (1,85):
    tmp_df = df_train_one_v[[i,85]]
    tmp_df0 = tmp_df.loc[tmp_df[85] == 0]
    tmp_df1 = tmp_df.loc[tmp_df[85] == 1]
    tmp_df2 = tmp_df.loc[tmp_df[85] == 2]
    tmp_df3 = tmp_df.loc[tmp_df[85] == 3]
    tmp_df4 = tmp_df.loc[tmp_df[85] == 4]
    tmp0_0 = tmp_df0.loc[tmp_df0[i] == 0]
    tmp0_1 = tmp_df0.loc[tmp_df0[i] == 1]
    count0_0,tmp = tmp0_0.shape
    count0_1,tmp = tmp0_1.shape
    tmp1_0 = tmp_df1.loc[tmp_df1[i] == 0]
    tmp1_1 = tmp_df1.loc[tmp_df1[i] == 1]
    count1_0,tmp = tmp1_0.shape
    count1_1,tmp = tmp1_1.shape
    tmp2_0 = tmp_df2.loc[tmp_df2[i] == 0]
    tmp2_1 = tmp_df2.loc[tmp_df2[i] == 1]
    count2_0,tmp = tmp2_0.shape
    count2_1,tmp = tmp2_1.shape
    tmp3_0 = tmp_df3.loc[tmp_df3[i] == 0]
    tmp3_1 = tmp_df3.loc[tmp_df3[i] == 1]
    count3_0,tmp = tmp3_0.shape
    count3_1,tmp = tmp3_1.shape
    tmp4_0 = tmp_df4.loc[tmp_df4[i] == 0]
    tmp4_1 = tmp_df4.loc[tmp_df4[i] == 1]
    count4_0,tmp = tmp4_0.shape
    count4_1,tmp = tmp4_1.shape
    while True:
        try:
            div0 = count0_1/count0_0
            break
        except ZeroDivisionError:
            break
            div0 = -1
    while True:
        try:
            div1 = count1_1/count1_0
            break
        except ZeroDivisionError:
            break
            div1 = -1
    while True:
        try:
            div2 = count2_1/count2_0
            break
        except ZeroDivisionError:
            break
            div2 = -1
    while True:
        try:
            div3 = count3_1/count3_0
            break
        except ZeroDivisionError:
            break
            div3 = -1
    while True:
        try:
    while True:
        try:
            div4 = count4_1/count4_0
            break
        except ZeroDivisionError:
            break
            div4 = -1

    tmp_df = pd.DataFrame([[i,count0_0,count0_1,div0,count1_0,count1_1,div1,count2_0,count2_1,div2,count3_0,count3_1,div3
                        ,count4_0,count4_1,div4]], columns = col_name)
    df = df.append(tmp_df)
divDF = df[["div0","div1","div2","div3","div4"]]
df['HighScore'] = df[["div0","div1","div2","div3","div4"]].max(axis=1)
df_trainfull_ohe = pd.concat([df_train_ohe, df_train['label5-index']],axis = 1)
tmp = df.loc[:,('row','HighScore')]
a = list(df_trainfull_ohe.head())
del a[-1]
tmp['feature_name'] = a
ar = df[["div0","div1","div2","div3","div4"]]
df_AR_ohe = ar.max(axis=1)
fea_name = tmp['feature_name']
CJ_Echo2 = pd.concat([fea_name, df_AR_ohe], axis=1)
CJ_Echo11= CJ_Echo2.reset_index(drop=True)
CJ_Echo11.columns = range(CJ_Echo11.shape[1])
```

ภาพประกอบ 31 Code สำหรับหาค่า AR ในคุณลักษณะประเภท binary

6. แสดงตัวอย่าง code ในการทำ tuning แบบ hyperparameter กับ Random Forest ตามที่กล่าวไว้ในหัวข้อที่ 4.1 ดังแสดงในภาพประกอบที่ 32

```
param_dist = {
'max_depth': [80, 90, 100],
'max_features': [2, 3],
'min_samples_leaf': [3, 4, 5],
'min_samples_split': [8, 10, 12],
'n_estimators': [100, 200, 300],
'bootstrap': [True, False],
'criterion': ["gini", "entropy"]
}

rf = RandomForestClassifier()
rf_random = RandomizedSearchCV(rf, param_distributions=param_dist, n_iter = 20,
                               n_jobs=-1, cv = 5, verbose=2, random_state=3425)
# Fit the random search model
rf_random.fit(features_train,labels_train);
pred = rf_random.predict(features_test)
rf_model.append(rf_random)
print(colors.BLUE)
print ('Best Score is:',rf_random.best_params_ )
print(colors.ENDC)
best_score.append(rf_random.best_params_)
print ("Random Forests Classifer")
acc = accuracy_score(pred, labels_test)
print ("Accuacy is {}.".format(round(acc,4)))
print('-------------------------------------------------------')
print(classification_report(pred, labels_test, digits = 3))
print('-------------------------------------------------------')
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
cnf_matri = confusion_matrix(labels_test, pred)
print(colors.Red)
print("confusion_matrix as below")
print(cnf_matri)
print(colors.ENDC)
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
fpr_rt_lm, tpr_rt_lm, _ = roc_curve(labels_test, pred)
roc_auc = auc(fpr_rt_lm, tpr_rt_lm)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_rt_lm, tpr_rt_lm, 'cornflowerblue', label='ROC curve (area = %0.3f)' % roc_auc)
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
print(colors.Red)
print("Time for loop is",time() - t0)
print(colors.ENDC)
print(colors.Purple  + "##########################################################" + colors.ENDC)
```

ภาพประกอบ 32 ตัวอย่าง code การทำ tuning แบบ hyperparameter+ Randon Forest

7. แสดงตัวอย่าง code ในการทำ tuning แบบ hyperparameter กับ Adaboost ตามที่กล่าวไว้ในหัวข้อที่ 4.2 ดังแสดงในภาพประกอบที่ 33

```
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [5, 10, 15, 20],
    'learning_rate': [0.001, 0.01, 0.1, 0.2, 0,3],
    'subsample': [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
    'colsample_bytree': [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
    'colsample_bylevel': [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
    'min_child_weight': [0.4, 0.5, 1.0, 3.0, 5.0, 7.0, 10.0],
    'gama': [0, 0.25, 0.5, 1.0],
    'reg_lambda': [0.1, 1.0, 5.0, 10.0, 50.0, 100.0]
    }

clf = xgb.XGBClassifier(seed=3425)
gx_search = RandomizedSearchCV(clf, param_grid, n_iter=20,
                       n_jobs=-1, verbose=2, cv=5,
                       scoring='neg_log_loss', random_state=42)

gx_search.fit(features_train,labels_train);
pred = gx_search.predict(features_test)

print(colors.BLUE)
print ('Best Score is:',gx_search.best_params_ )
print(colors.ENDC)
best_score.append(gx_search.best_params_)
print ("XGBoost Classifer")
acc = accuracy_score(pred, labels_test)
print ("Accuacy is {}.".format(round(acc,4)))
print('-------------------------------------------------------')
print(classification_report(pred, labels_test, digits = 3))
print('-------------------------------------------------------')
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
cnf_matri = confusion_matrix(labels_test, pred)
print(colors.Red)
print("confusion_matrix as below")
print(cnf_matri)
print(colors.ENDC)
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
fpr_gx_lm, tpr_xg_lm, _ = roc_curve(labels_test, pred)
roc_auc = auc(fpr_xg_lm, tpr_xg_lm)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_xg_lm, tpr_xg_lm, 'cornflowerblue', label='ROC curve (area = %0.3f)' % roc_auc)
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
print(colors.Red)
print("Time for loop is",time() - t0)
print(colors.ENDC)
print(colors.Purple  + "#########################################################" + colors.ENDC)
```

ภาพประกอบ 33 ตัวอย่าง code การทำ tuning แบบ hyperparameter+ Adaboost

8. แสดงตัวอย่าง code ในการทำ tuning แบบ hyperparameter กับ XGBoost ตามที่กล่าวไว้ในหัวข้อที่ 4.3 ดังแสดงในภาพประกอบที่ 34

```
param_grid ={'learning_rate':[0.25,0.5,0.75,1.],
             'n_estimators':[100,250,500,650],
             }
ABC = AdaBoostClassifier()
ABC_search = RandomizedSearchCV(ABC, param_grid, n_jobs=-1, n_iter=10, cv=5)

# Fit the random search model
ABC_search.fit(features_train,labels_train);

pred = ABC_search.predict(features_test)
ABC_model.append(ABC_search)
T0_labels_test.append(labels_test)
T0_pred.append(pred)

print(colors.BLUE)
print ('Best Score is:',ABC_search.best_params_ )
print(colors.ENDC)
best_score.append(ABC_search.best_params_)
print ("Random Forests Classifer")
acc = accuracy_score(pred, labels_test)
print ("Accuacy is {}.".format(round(acc,4)))
print('-----------------------------------------------------')
print(classification_report(pred, labels_test, digits = 3))
print('-----------------------------------------------------')
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
cnf_matri = confusion_matrix(labels_test, pred)
print(colors.Red)
print("confusion_matrix as below")
print(cnf_matri)
print(colors.ENDC)
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
fpr_abc_lm, tpr_abc_lm, _ = roc_curve(labels_test, pred)
roc_auc = auc(fpr_abc_lm, tpr_abc_lm)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_abc_lm, tpr_abc_lm, 'cornflowerblue', label='ROC curve (area = %0.3f)' % roc_auc)
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
print(colors.Purple  + "#########################################################" + colors.ENDC)
```

ภาพประกอบ 34 ตัวอย่าง code การทำ tuning แบบ hyperparameter+ XGBoost

9. แสดงตารางทดสอบหาค่าความแม่นยำของ AR โดยพบว่าค่า AR ตั้งแต่ 0.01 – No AR มีค่าความแม่นยำใกล้เคียงกัน แต่ค่า AR 0.08 – No AR ใช้เวลาในการ train ที่มากกว่า ดังนั้นผู้วิจัยจึงเลือกใช้ค่า AR ที่ 0.01 ในการทำงานวิจัยนี้ ดังแสดงในตารางที่ 18

ตาราง 18 ตารางทดสอบหาค่าความแม่นยำของการกำหนดค่า AR

No	AR	Accuracy	Feature	Time/s
1	0.1	83.48%	50	146.639
2	0.01	84.41%	77	203
3	0.08	84.67%	89	290.088
4	No AR	84.84	122	317.813

10. แสดงตัวอย่าง code สำหรับ plot กราฟของ Feature Importance ที่ cluster 0 ของ XGBoost ดังแสดงในภาพประกอบที่ 35

```
clf0 = xgb.XGBClassifier(subsample=0.6, silent=False, reg_lambda=5.0, n_estimators=200, min_child_weight=0.4, max_depth=5,
    learning_rate=0.2, gamma=0, colsample_bytree=1.0, colsample_bylevel=0.6,n_iter=20,
    n_jobs=-1, verbose=2, cv=5, random_state=42, seed=3425)
clf0.fit(T_features_train[0],T_labels_train[0])
from matplotlib import pyplot
import matplotlib.pyplot as plt
plt.figure(2, figsize=(11, 6))
plt.bar(range(len(clf0.feature_importances_)), clf0.feature_importances_, label="Cluster0")
plt.legend()

plt.title('Cluster')
plt.xlabel('Features')
pyplot.show()
```

ภาพประกอบ 35 code สำหรับ plot กราฟของ Feature Importance cluster 0

11. แสดงตัวอย่าง code สำหรับ plot กราฟของ Feature Importance ที่ cluster 1 ของ XGBoost ดังแสดงในภาพประกอบที่ 36

```
clf1 = xgb.XGBClassifier(subsample=0.6, silent=False, reg_lambda=5.0, n_estimators=200, min_child_weight=0.4, max_depth=5,
    learning_rate=0.2, gamma=0, colsample_bytree=1.0, colsample_bylevel=0.6,n_iter=20,
    n_jobs=-1, verbose=2, cv=5, random_state=42, seed=3425)
clf1.fit(T_features_train[1],T_labels_train[1])

plt.figure(2, figsize=(11, 6))
plt.bar(range(len(clf1.feature_importances_)), clf1.feature_importances_, color="c", label="Cluster1")
plt.legend()

plt.title('Cluster')
plt.xlabel('Features')
pyplot.show()
```

ภาพประกอบ 36 code สำหรับ plot กราฟของ Feature Importance cluster 1

ประวัติผู้เขียน

ชื่อ-สกุล	Apichit Pattawaro
วัน เดือน ปี เกิด	2 October 1979
สถานที่เกิด	Udonthani
วุฒิการศึกษา	Srinakharinwirot University
ที่อยู่ปัจจุบัน	223/7 Udon dusadee rd, Mueang, Udon Thani 41000.