JSET-RMU
Journal of Science, Engineering, and Technology
Rajabhat Maha Sarakham University
https://ph03.tci-thaijo.org/index.php/jsetRMU

บทความวิจัย (Research Article) **Volume 4, No. 2, July-December 2025, ISSN: 2821-9066 (Print)**

ระบบตรวจจับเว็บไซต์ฟิชชิงแบบเรียลไทม์โดยใช้แมชชีนเลิร์นนิง

Web-Based Real-Time Phishing Detection System Integrated with Machine Learning

ปวิณา ชาปัญญา ฉัฐพร จุลลาศรี สาธิต กระเวนกิจ และ จักรชัย โสอินทร์*

Pawina Chapanya, Chartaporn Jullasri, Satit Kravenkit and Chakchai So-In*

สาขาวิชาวิทยาการคอมพิวเตอร์ วิทยาลัยการคอมพิวเตอร์ มหาวิทยาลัยขอนแก่น จังหวัดขอนแก่น 40000

Program of Computer Science, College of Computing, Khon Kaen University,

Khon Kaen Province, 40000

*Corresponding author E-mail: chakso@kku.ac.th

Received: Nov 4, 2025/ **Revised:** Dec 16, 2025/ **Accepted:** Dec 21, 2025

บทคัดย่อ

ปัจจุบันการป้องกันการโจมตีจากฟิชชิงที่วิเคราะห์และคัดกรองจาก URL Blacklist/Whitelist ร่วมกับเทคนิค Machine Learning ได้รับการพิสูจน์ว่ามีประสิทธิภาพสูง อย่างไรก็ตาม ยังมีข้อจำกัดในการประยุกต์โมดูลการเรียนรู้ของเครื่องในสภาพแวดล้อมจริง เพื่อให้สามารถตรวจจับการโจมตีฟิชชิงได้อย่างยืดหยุ่นและมีประสิทธิภาพ ทีมวิจัยจึงพัฒนาระบบตรวจจับเว็บไซต์ฟิชชิงบนเว็บที่ผสานเข้ากับโมดูลการเรียนรู้ของเครื่อง เพื่อทำหน้าที่ตรวจสอบและตัดสินใจอนุญาตหรือปฏิเสธการเข้าถึงเว็บไซต์ ระบบดังกล่าวติดตั้งบน Proxy Server ที่พัฒนาด้วยภาษา Python การตรวจจับอาศัยการวิเคราะห์จาก (1) คุณลักษณะของโดเมน (2) คุณลักษณะของ URL และ (3) คุณลักษณะของพาธและชื่อไฟล์ ระบบถูกออกแบบให้มีส่วนติดต่อผู้ใช้ (UI) สำหรับการฝึกและประเมินโมเดลได้ตามต้องการ พร้อมรองรับอัลกอริทึมหลากหลาย ได้แก่ Random Forest (RF), Support Vector Machine (SVM), Multilayer Perceptron (MLP), Deep Neural Network (DNN) และ Extreme Gradient Boosting (XGBoost) ในประเมินพบว่าอัลกอริทึม Random Forest (RF) ให้ค่าความแม่นยำ Accuracy 96.70% และ F1-score 96.68% สูงที่สุด นอกจากนี้ผลการประเมินความพึงพอใจของผู้ใช้งานจำนวน 10 คน พบว่าอยู่ในระดับมาก 4.73 คะแนน โดยมีจุดเด่นด้านความน่าเชื่อถือและการทำงานที่เสถียรภาพของระบบ 4.91 คะแนน ขณะที่ด้านความสามารถของ UI ในการอัปโหลดและฝึกโมเดล มีค่าเฉลี่ยต่ำอยู่ที่ 4.36 คะแนน

คำสำคัญ: เว็บไซต์ฟิชชิง, แมชชีนเลิร์นนิง, ปัญญาประดิษฐ์, พร็อกซีเซิร์ฟเวอร์, ความมั่นคงปลอดภัยไซเบอร์

ABSTRACT

Phishing attack prevention that relies on URL blacklist/whitelist analysis combined with machine learning techniques has proven highly effective. However, limitations still exist in the practical implementation of machine learning modules under real-world environments, particularly due to the lack of real-time deployment. Therefore, this research developed a web-based phishing detection system integrated with a machine learning module to analyze, verify, and determine whether to allow or deny access to websites. The system is deployed on a proxy server developed in Python and analyzes (1) domain features, (2) URL features, and (3) path and filename characteristics. It is designed with a user interface (UI) that supports model training and evaluation as needed and accommodates multiple algorithms, including Random Forest (RF), Support Vector Machine (SVM), Multilayer Perceptron (MLP), Deep Neural Network (DNN), and Extreme Gradient Boosting (XGBoost). The evaluation results revealed that the Random Forest (RF) algorithm achieved the highest performance, with an accuracy of 96.70% and an F1-score of 96.68%. Moreover, a user satisfaction survey of 10 participants indicated a high level of satisfaction mean = 4.73, particularly in system reliability and stability, 4.91, while the UI capability for model uploading and training received a relatively lower score of 4.36.

Keywords: phishing website, machine learning, artificial intelligence, proxy server, cybersecurity

บทนำ

ปัจจุบันภัยคุกคามจากเว็บไซต์ฟิชชิงยังคงเพิ่มขึ้นอย่างต่อเนื่อง จากรายงานของ Anti-Phishing Working Group (2023) ระบุว่าในปี 2023 มีจำนวนการโจมตีฟิชชิงทั่วโลกกว่า 4.98 ล้านครั้ง เพิ่มขึ้นประมาณ 6 เปอร์เซ็นต์จากปีก่อนหน้า และถือเป็นค่าสูงสุดที่เคยบันทึกไว้ ผู้โจมตีมุ่งขโมยข้อมูลสำคัญขององค์กร ไม่ว่าจะเป็นรหัสผ่าน หมายเลขบัตรเครดิต หรือข้อมูลธุรกิจ ซึ่งสร้างผลกระทบทั้งด้านความปลอดภัย เศรษฐกิจ และชื่อเสียงขององค์กร วิธีการตรวจจับแบบดั้งเดิม เช่น การใช้ Blacklist หรือ Whitelist แม้จะช่วยลดความเสี่ยงได้ในระดับหนึ่ง แต่ยังมีข้อจำกัดทั้งในด้านความแม่นยำและการตอบสนองต่อรูปแบบการโจมตีใหม่ ๆ ที่เปลี่ยนแปลงอยู่ตลอดเวลา (Alkhalil et al., 2021)

ด้วยเหตุนี้จึงมีการนำเทคนิคการเรียนรู้ของเครื่อง (Machine Learning: ML) เข้ามาช่วยเพิ่มความแม่นยำในการตรวจจับฟิชชิง โดยมีการใช้อัลกอริทึมหลากหลาย เช่น Random Forest (RF) ซึ่งเป็นวิธีการเรียนรู้แบบเอนเซมเบิลที่ช่วยลดปัญหา overfitting (Yang et al., 2021; Harinahalli Lokesh & Boregowda, 2021), Support Vector Machine (SVM) ที่สามารถจำแนกข้อมูลได้ทั้งแบบเส้นตรงและไม่เชิงเส้นผ่านการใช้ kernel (Cortes & Vapnik, 1995; Anupam & Kar, 2021), Multilayer Perceptron (MLP) สำหรับงานเรียนรู้เชิงลึกและการจำแนกข้อมูลที่ซับซ้อน (Mushtaq et al., 2024), Deep Neural Network (DNN) ที่สามารถเรียนรู้คุณลักษณะเชิงลึกได้อย่างมีประสิทธิภาพ (Haq et al., 2024) รวมถึง Extreme Gradient Boosting (XGBoost) ซึ่งถูกนำมาใช้อย่างแพร่หลายในงานด้านความปลอดภัยไซเบอร์ (Chen et al., 2022)

ในช่วงไม่กี่ปีที่ผ่านมา งานวิจัยด้านการตรวจจับฟิชชิงมีความก้าวหน้าอย่างต่อเนื่อง โดยเฉพาะการประยุกต์ใช้สถาปัตยกรรมเชิงลึกประเภท Transformer ซึ่งมีความสามารถในการจับความสัมพันธ์เชิงลำดับและโครงสร้างของ URL ได้ละเอียดกว่าวิธีดั้งเดิม งานของ Asiri et al. (2024) ได้นำเสนอ PhishTransformer ซึ่งออกแบบให้แปลง URL เป็นลำดับ

โทเคนก่อนเรียนรู้ความสัมพันธ์ของส่วนประกอบต่าง ๆ ของลิงก์ เพื่อเพิ่มความแม่นยำในการจำแนกฟิชชิงที่มีรูปแบบซับซ้อน โดยเฉพาะการประยุกต์ใช้สถาปัตยกรรม Transformer ร่วมกับโครงข่ายประสาทเทียม เพื่อเพิ่มความสามารถในการตรวจจับฟิชชิงที่ถูกดัดแปลงเชิงรูปแบบ (obfuscation) และ URL ที่มีความหลากหลายสูง งานของ Xu (2021) ได้นำเสนอโมเดล Transformer สำหรับตรวจจับ phishing URLs ซึ่งแสดงให้เห็นว่าการแปลง URL เป็นลำดับ token และเรียนรู้แบบ sequence สามารถช่วยจำแนก phishing ได้อย่างมีประสิทธิภาพ นอกจากนี้ Haq et al. (2024) ยังได้ประเมินประสิทธิภาพของโครงข่ายประสาทเทียมเชิงลึกบนข้อมูลจริงเพื่อสะท้อนพฤติกรรมฟิชชิงในสภาพแวดล้อมที่ใกล้เคียงการใช้งานจริง ผลการทดลองชี้ให้เห็นว่าโมเดล deep learning สามารถเรียนรู้คุณลักษณะเชิงโครงสร้างของ URL ได้อย่างมีประสิทธิภาพ รองรับรูปแบบการโจมตีที่มีความหลากหลาย และช่วยเพิ่มความสามารถในการตรวจจับฟิชชิงที่มีรูปแบบซับซ้อนมากขึ้น

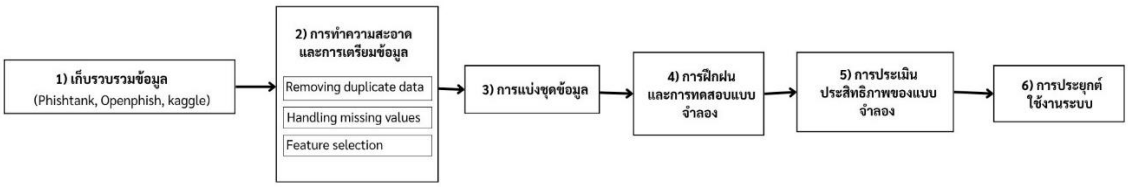
อย่างไรก็ตาม งานวิจัยจำนวนมากยังคงประเมินระบบในลักษณะออฟไลน์หรือในสภาพแวดล้อมจำลอง ทำให้ยังไม่สามารถแสดงให้เห็นประสิทธิภาพการใช้งานจริงบนระบบโครงสร้างพื้นฐานขององค์กร เช่น Proxy Server ที่ต้องรองรับคำขอจำนวนมาก เพื่อแก้ช่องว่างดังกล่าว งานวิจัยนี้ได้พัฒนาระบบตรวจจับเว็บไซต์ฟิชชิงที่ทำงานร่วมกับ Proxy Server แบบเรียลไทม์ โดยตรวจสอบ URL ผ่านสองขั้นตอน ได้แก่ (1) การตรวจสอบด้วย Blacklist/Whitelist และ (2) การจำแนกด้วยโมเดล Machine Learning สำหรับ URL ที่ไม่อยู่ในฐานข้อมูล รวมถึงจัดทำส่วนติดต่อผู้ใช้ (UI) สำหรับการฝึกและประเมินโมเดล และระบบบันทึกเหตุการณ์ตามมาตรฐาน NIST (2006) เพื่อรองรับการใช้งานจริงในองค์กร

ในงานวิจัยนี้ ระบบถูกออกแบบให้รองรับสภาพแวดล้อมองค์กรที่ใช้ Explicit Proxy โดยตรวจสอบเฉพาะ URL และข้อมูลส่วนหัวของ HTTP/HTTPS โดยไม่ถอดรหัสเนื้อหา เพื่อหลีกเลี่ยงผลกระทบต่อความเป็นส่วนตัวและลดภาระการประมวลผล การตรวจจับมุ่งเน้นรูปแบบ URL ที่มีความเสี่ยง เช่น โดเมนปลอมหรือโครงสร้างที่เลียนแบบเว็บไซต์จริง โดยระบบสามารถรองรับปริมาณคำขอได้ประมาณ 1,000 คำขอต่อวินาทีภายใต้แบนด์วิดท์ 1 Gbps และใช้รูปแบบคำขอต่อเนื่องเพื่อสะท้อนพฤติกรรมการใช้งานจริงโดยไม่มีการใช้เทคนิคเร่งประสิทธิภาพเพิ่มเติม อย่างไรก็ตาม ในสภาวะการใช้งานจริง ระบบยังมีข้อจำกัดสำคัญ ได้แก่ การไม่รองรับการตรวจเนื้อหา HTTPS ทำให้ไม่สามารถตรวจจับฟิชชิงที่ซ่อนอยู่ภายในเนื้อหาเว็บได้ทั้งหมด

วิธีการวิจัย

1. วิธีการดำเนินงาน

แสดงขั้นตอนวิธีดำเนินการวิจัยซึ่งประกอบไปด้วย 6 ขั้นตอน



ภาพที่ 1 ขั้นตอนวิธีดำเนินการ

1) การเก็บรวบรวมข้อมูล (Data Collection) ผู้วิจัยได้รวบรวมชุดข้อมูล (Dataset) ของ URLs จากแหล่งข้อมูลสาธารณะและฐานข้อมูลที่เชื่อถือได้ ได้แก่ OpenPhish (2024) Kaggle (2024) และ PhishTank (2024) โดยข้อมูลครอบคลุมทั้งเว็บไซต์ฟิชชิงและเว็บไซต์ปกติ เพื่อใช้เป็นข้อมูลตั้งต้นสำหรับการฝึกและทดสอบแบบจำลองการเรียนรู้ของเครื่อง (Machine Learning Models)

2) การทำความสะอาดและการเตรียมข้อมูล (Data Cleaning and Preprocessing) กระบวนการนี้มุ่งปรับปรุงคุณภาพของข้อมูลโดยเน้นการลบข้อมูลที่ซ้ำซ้อน (Removing Duplicate Data) การจัดการกับค่าที่หายไป (Handling Missing Values) และการเลือกคุณลักษณะที่สำคัญ (Feature Selection) เพื่อเพิ่มความถูกต้องและประสิทธิภาพในการจำแนกเว็บไซต์ฟิชชิง

3) ระบบรองรับการเตรียมข้อมูลก่อนการแบ่งชุดข้อมูลผ่าน UI ได้แก่ Canonicalization และ De-duplication โดยก่อนการแบ่งข้อมูล ระบบจะดำเนินการ canonicalization เพื่อจัดรูปแบบ URL ให้เป็นมาตรฐานก่อนนำเข้าสู่กระบวนการประเมินโมเดล จากนั้นทำ exact deduplication ด้วย hash ของ canonical URL รวมถึง near-duplicate detection (host ร่วมกับ path similarity) เพื่อขจัดข้อมูลซ้ำ และรายงานจำนวนก่อน–หลังเป็นเปอร์เซ็นต์ต่อชุดข้อมูล การประเมินความสามารถทั่วไป (Cross-dataset & Temporal Split) เพื่อทดสอบการทั่วไปของโมเดลได้ดำเนินการทดลอง cross-dataset และ temporal split โดยฝึกบนชุดข้อมูลหนึ่ง (A) และทดสอบบนอีกชุดหนึ่ง (B) รวมทั้งแบ่งตามเวลา (train ก่อน cutoff, test หลัง cutoff) มีการประเมินความสามารถทั่วไปของโมเดลด้วยค่า ROC-AUC ผ่านการทดลองแบบ cross-dataset และ temporal split Ethical, Licensing, and Data Usage Compliance Statement ผู้วิจัยได้ตรวจสอบข้อกำหนดการใช้งานของชุดข้อมูลที่ใช้ (OpenPhish, PhishTank, Kaggle) และยึดตาม ToS/License ที่เกี่ยวข้อง โดยระบุแหล่งที่มาและขอบเขตการใช้งาน รวมทั้งกำหนดวิธีการปกปิดข้อมูล เช่น hashing/anonymization ของ IP และ path เพื่อป้องกันการเปิดเผยข้อมูลส่วนบุคคล และแนะนำให้ขออนุญาตหรือใช้ข้อมูลที่ได้รับสิทธิ์สำหรับการเผยแพร่

a. Hold-out Split แบ่งข้อมูลออกเป็นชุดฝึก (Training Set) ร้อยละ 80 และชุดทดสอบ (Test Set) ร้อย ละ 20 ซึ่งเป็นวิธีมาตรฐานที่ใช้กันทั่วไป (Géron, 2022)

b. K-Fold Cross Validation (K-Fold CV) เทคนิคที่แบ่งข้อมูลออกเป็น k ส่วนเท่า ๆ กัน และสลับกันใช้เป็นชุดทดสอบ/ชุดฝึก เพื่อเพิ่มความน่าเชื่อถือและลดความเอนเอียงของการประเมินแบบจำลอง (Kohavi, 1995; Raschka & Mirjalili, 2020)

4) การฝึกฝนและการทดสอบแบบจำลองในสภาพแวดล้อม (Model Training and Testing)

4.1 การตั้งค่าโมเดล (Model Hyperparameters)

ในการทดลองครั้งนี้ โมเดลแต่ละชนิดได้รับการกำหนดค่าให้เหมาะสมกับลักษณะของข้อมูล URL ฟิชชิง ซึ่งมีโครงสร้างแบบลำดับและมิติข้อมูลสูง โดย Random Forest ถูกตั้งค่าให้ใช้จำนวน trees เท่ากับ 100 พร้อมค่า random_state เท่ากับ 42 เพื่อรักษาความคงที่ของผลลัพธ์และลดโอกาสเกิดการเรียนรู้เฉพาะตัวอย่าง (overfitting) ส่วนโมเดล SVM แบบ LinearSVC กำหนด max_iter ที่ 2000 เพื่อรองรับการทำงานกับเวกเตอร์ TF-IDF ที่มักมีจำนวนคุณลักษณะสูงมาก ขณะที่โมเดล MLP กำหนดชั้นซ่อนจำนวน 100 นิวรอน ใช้ฟังก์ชันกระตุ้นแบบ ReLU และจำนวนรอบการฝึก 300 รอบ ซึ่งเป็นค่าที่เหมาะสมต่อทั้งประสิทธิภาพในการเรียนรู้ความสัมพันธ์ที่ไม่เป็นเชิงเส้น และเวลาในการประมวลผล สำหรับโมเดล DNN ใช้ชั้นซ่อนสามชั้น (256, 128 และ 64 หน่วย) ร่วมกับตัวแก้ปัญหา Adam และอัตราการเรียนรู้เริ่มต้น 0.001 พร้อมการเปิดใช้ early stopping และสัดส่วนข้อมูลตรวจสอบ 0.1 เพื่อควบคุมการเรียนรู้มากเกินไป ด้าน XGBoost กำหนดตัวชี้วัดผลลัพธ์เป็น logloss ซึ่งเหมาะกับโจทย์การจำแนกแบบทวินาม และกำหนด random_state เป็น 42 เช่นเดียวกัน การเลือกค่า Hyperparameters

4.2 สภาพแวดล้อมการทดลอง (Experimental Environment)

การทดลองในงานวิจัยนี้ดำเนินการด้วย Python 3.10 โดยใช้ไลบรารี Scikit-learn, XGBoost และ Pandas สำหรับการเตรียมข้อมูล การฝึกโมเดล และการประเมินประสิทธิภาพ ทั้งหมดประมวลผลบนเครื่องเสมือน (Virtual Machine) บนแพลตฟอร์ม Microsoft Azure ซึ่งกำหนดสเปกเป็น CPU แบบมัลติคอร์และหน่วยความจำเพียงพอต่อการ

ประมวลผล พร้อมการเชื่อมต่อเครือข่ายความเร็ว 1 Gbps เพื่อให้การวัดค่าความหน่วงสะท้อนรูปแบบการทำงานที่สามารถพบได้ในสภาพแวดล้อมของ Proxy Server

5) การประเมินประสิทธิภาพของแบบจำลอง (Model Evaluation)

การประเมินแบบจำลองดำเนินการด้วยตัวชี้วัด Accuracy, Precision, Recall และ F1-score เพื่อเปรียบเทียบความสามารถของโมเดล รวมถึงใช้ค่า ROC-AUC และ PR-AUC เพื่อประเมินประสิทธิภาพภายใต้สภาวะข้อมูลไม่สมดุล สำหรับการอธิบายผลการตัดสินใจของโมเดลได้ใช้เทคนิค SHAP เพื่อวิเคราะห์ความสำคัญของคุณลักษณะ โดยพบว่าฟีเจอร์ที่มีผลต่อการจำแนกฟิชชิงอย่างต่อเนื่อง ได้แก่ has_suspicious_keyword, domain_entropy, url_length, num_subdomains และ path_has_extension นอกจากนี้ยังมีการดำเนินการ Ablation Study เพื่อวิเคราะห์ผลกระทบของการเพิ่มหรือตัดกลุ่มฟีเจอร์ต่าง ๆ ซึ่งสรุปผลไว้ในตารางที่ 4

6) การประยุกต์ใช้งานระบบ (System Deployment)

ผู้วิจัยได้เลือกแบบจำลองที่มีประสิทธิภาพสูงสุดมาบูรณาการเข้ากับ Proxy Server ที่พัฒนาด้วยภาษา Python เพื่อทำหน้าที่ตรวจสอบ URL ที่ผู้ใช้ร้องขอแบบเรียลไทม์ หากตรวจพบว่าเป็นเว็บไซต์ฟิชชิง ระบบจะทำการบล็อกการเข้าถึงด้วยรหัส 403 Forbidden ตามมาตรฐาน HTTP/1.1 (Fielding et al., 2022) หรือเปลี่ยนเส้นทางไปยังหน้า blocked.html ในกรณีการเชื่อมต่อแบบ HTTP ธรรมดา พร้อมบันทึกเหตุการณ์ลงใน Logging System ตามแนวทางของ NIST SP 800-92 (NIST, 2006) และ OWASP Secure Coding Practices (OWASP, 2023) เพื่อใช้ในการวิเคราะห์และปรับปรุงระบบ

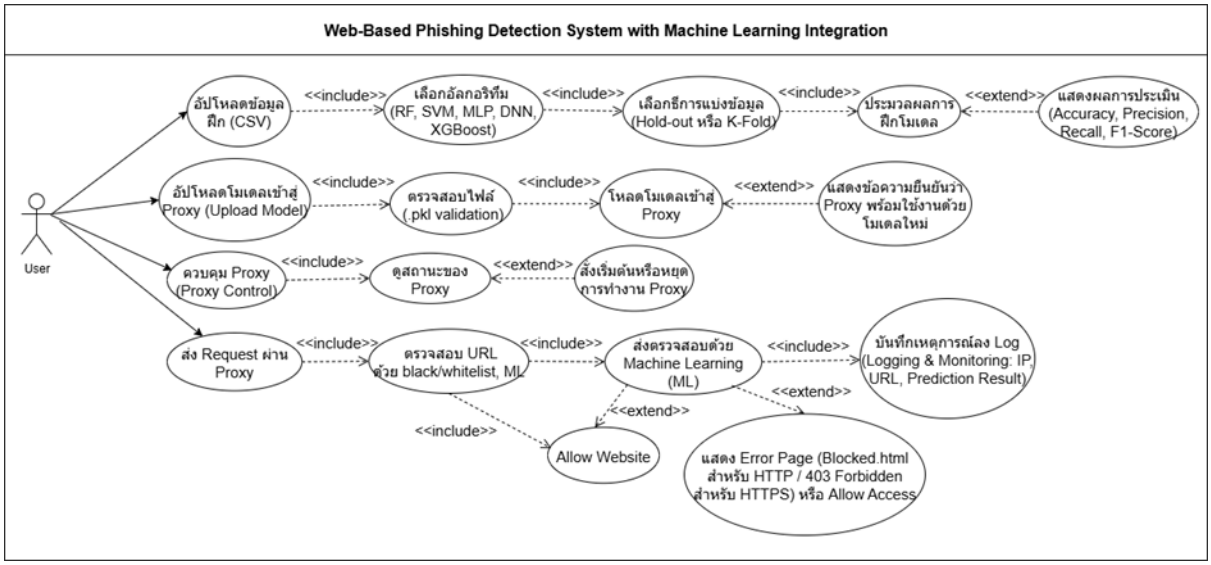
Threat Model and Security Architecture พร็อกซีทำงานในโหมด Explicit Proxy โดยผู้ใช้ตั้งค่าเบราว์เซอร์ให้เชื่อมต่อผ่านพร็อกซีโดยตรง สำหรับคำขอ HTTPS ระบบจะตรวจสอบชื่อโดเมนจาก SNI (CONNECT handshake) โดยไม่ถอดรหัสข้อมูล (non-TLS interception) เพื่อรักษาความเป็นส่วนตัว (Mozilla, 2023) หากพบ URL ที่เข้าข่ายฟิชชิง ระบบจะตอบกลับด้วยรหัส HTTP 403 Forbidden ส่วนคำขอ HTTP ปกติจะถูกเปลี่ยนเส้นทางไปยัง blocked.html การออกแบบดังกล่าวป้องกันปัญหาที่เกี่ยวกับ HSTS และ certificate pinning และยังรักษาความเป็นส่วนตัวของผู้ใช้ในระหว่างการตรวจสอบ

ระบบรองรับการอัปเดต Blacklist และ Whitelist โดยใช้การกรองแบบสองชั้น ซึ่งให้รายการทั้งสองทำหน้าที่เป็นด่านแรกก่อนเข้าสู่กระบวนการอนุมานของโมเดล เพื่อลดภาระการประมวลผลเชิง Machine Learning ลง ในกรณีที่ URL ปรากฏในทั้งสองรายการ ระบบกำหนดให้ Whitelist มีลำดับความสำคัญสูงกว่าเพื่อป้องกันการบล็อกเว็บไซต์ที่เชื่อถือได้โดยไม่ตั้งใจ นอกจากนี้ยังมีการกำหนดค่า TTL (Time to Live) สำหรับแต่ละประเภท เพื่อให้ข้อมูลอัปเดตจากแหล่งข้อมูลภายนอกที่เชื่อถือได้ เช่น OpenPhish และ PhishTank กลไกนี้ช่วยรักษาความถูกต้องของฐานข้อมูล URL ลดโอกาสเกิดการบล็อกผิดพลาด และเสริมความมั่นคงของระบบโดยรวมในระยะยาว

1.1 การออกแบบ Use case ระบบตรวจจับเว็บไซต์ฟิชชิงแบบเรียลไทม์โดยใช้แมชชีนเลิร์นนิง

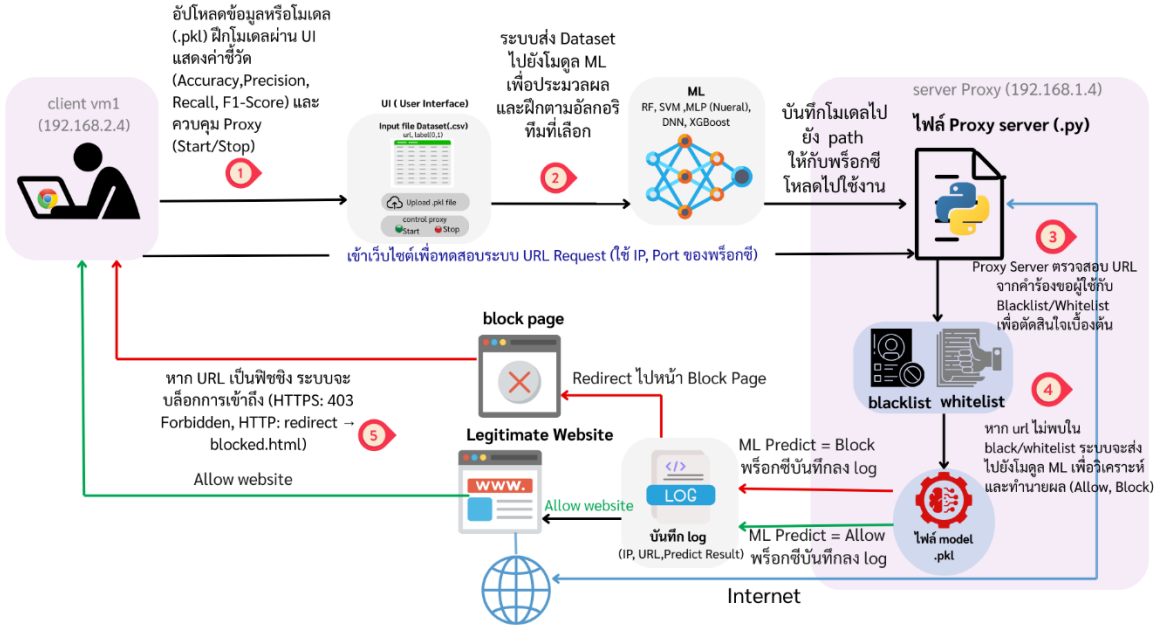
สำหรับในส่วนของ Software ภาพที่ 2 แสดงถึง Use case เพื่อให้เห็นการทำงานของฟังก์ชันหลักและฟังก์ชันย่อยของระบบ ผู้วิจัยได้ออกแบบและนำเสนอ Use Case ทั้ง 4 กรณี ได้แก่ (1) การอัปโหลดข้อมูลฝึก โดยผู้ใช้สามารถอัปโหลดข้อมูลในรูปแบบ CSV เลือกอัลกอริทึมและวิธีการแบ่งข้อมูล (Hold-out หรือ K-Fold) เพื่อสร้างโมเดล Machine Learning และดูผลการประเมินด้วยตัวชี้วัดมาตรฐาน ได้แก่ Accuracy, Precision, Recall และ F1-score ผ่านส่วนติดต่อผู้ใช้ (UI), (2) การอัปโหลดโมเดลเข้าสู่ Proxy โดยผู้ใช้อัปโหลดโมเดลที่ผ่านการฝึกและประเมินแล้ว (ไฟล์ .pkl) ไปยัง Proxy Server เพื่อใช้งานในการตรวจสอบ URL ซึ่งระบบจะตรวจสอบความถูกต้องของไฟล์ เช่น ประเภทและโครงสร้าง ก่อนโหลดโมเดลเข้าสู่ Proxy และแจ้งยืนยันว่าพร้อมทำงานด้วยโมเดลใหม่, (3) การควบคุม Proxy ผู้ใช้สามารถเปิด-ปิดและตรวจสอบสถานะของ

Proxy Server ผ่าน UI เพื่อจัดการการทำงานของระบบตรวจจับได้ตามต้องการ, และ (4) ในขั้นตอนการตรวจสอบและตัดสินใจ URL ระบบ Proxy จะตรวจสอบ URL กับรายการ Whitelist/Blacklist ก่อน และหากไม่พบรายการตรงกัน URL จะถูกส่งต่อให้โมดูล Machine Learning วิเคราะห์แบบ Real-time เพื่อพิจารณาอนุญาตหรือปฏิเสธการเข้าถึง



ภาพที่ 2 แผนภาพกรณีการใช้งานผ่านโปรแกรม

2. ภาพรวมของระบบ



ภาพที่ 3 ภาพรวมของระบบ

ภาพที่ 3 แสดงภาพรวมการทำงานของระบบตรวจจับเว็บไซต์ฟิชชิง โดยมี 5 กระบวนการดังนี้ (1) ผู้ใช้เริ่มต้นจากการอัปโหลดข้อมูล Dataset หรือไฟล์โมเดลสำเร็จรูป (.pkl) ผ่าน UI ซึ่งรองรับการจัดการชุดข้อมูล การฝึก การประเมิน และการนำเข้าโมเดลเข้าสู่ Proxy Server โดยเมื่อการฝึกโมเดลเสร็จสิ้น ระบบจะแสดงผลการประเมิน เช่น Accuracy Precision

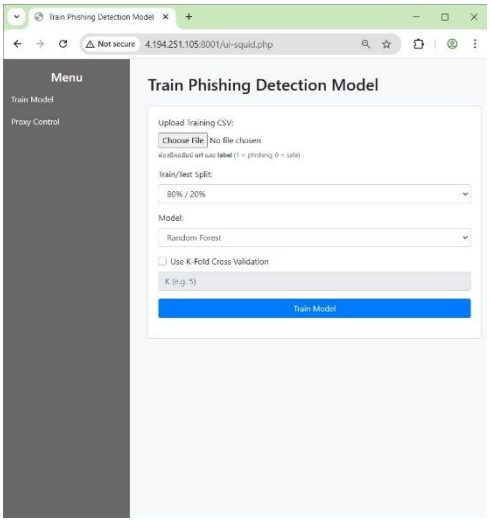
Recall และ F1-score รวมถึงควบคุมสถานะการทำงานของ Proxy Server (เริ่มต้น/หยุดการทำงาน) ได้โดยตรงจาก UI (2) เมื่อผู้ใช้ทำการเข้าถึงเว็บไซต์ ระบบจะส่งคำขอ URL จากเครื่องลูกข่าย (Client VM) ไปยัง Proxy Server เพื่อทำการตรวจสอบ โดยมีการบันทึกข้อมูลพื้นฐาน เช่น IP Address และ Port ของผู้ร้องขอ (National Institute of Standards and Technology [NIST], 2006) (3) Proxy Server จะตรวจสอบ URL ที่ร้องขอเทียบกับฐานข้อมูล Blacklist และ Whitelist เพื่อคัดกรองเบื้องต้น หากพบว่า URL ตรงกับรายการใน Whitelist ระบบจะอนุญาตให้เข้าถึงเว็บไซต์ได้ทันที แต่หากพบใน Blacklist ระบบจะบล็อกการเข้าถึงโดยอัตโนมัติ (4) ในกรณีที่ URL ไม่พบในรายการ Blacklist หรือ Whitelist Proxy Server จะส่ง URL ดังกล่าวต่อไปยังโมดูล Machine Learning (ML) เพื่อทำการทำนายผล โดยโมเดลที่ใช้ประกอบด้วยอัลกอริทึม ได้แก่ Random Forest (RF), Support Vector Machine (SVM), Multilayer Perceptron (MLP), Deep Neural Network (DNN) และ Extreme Gradient Boosting (XGBoost) (Yang et al., 2021; Harinahalli Lokesh & BoreGowda, 2021; Haq et al., 2024) ซึ่งผ่านการฝึกจากชุดข้อมูลฟิชชิงและเว็บไซต์ปกติมาก่อนหน้า (5) ผลการทำนายจากโมเดล Machine Learning จะถูกนำมาใช้ในการตัดสินใจ หาก URL ถูกระบุว่าเป็นฟิชชิง ระบบจะบล็อกการเข้าถึง โดยกรณีที่เป็น HTTP ระบบจะทำการ Redirect ผู้ใช้ไปยังหน้า blocked.html ที่ระบุข้อความแจ้งเตือน ส่วนในกรณีของ HTTPS ระบบจะตอบกลับด้วยรหัสข้อผิดพลาด 403 Forbidden ตามมาตรฐาน HTTP/1.1 (Fielding et al., 2022) แต่หากโมเดลทำนายว่าเป็นเว็บไซต์ปกติ ระบบจะอนุญาตให้เข้าถึงได้ตามปกติ นอกจากนี้ Proxy Server ยังทำการบันทึกเหตุการณ์ทุกครั้งลงในระบบบันทึกเหตุการณ์ (Logging System) ซึ่งเก็บข้อมูลสำคัญ เช่น เวลา, IP Address, URL ที่ร้องขอ และผลการตรวจจับ (Allow/Block) เพื่อใช้วิเคราะห์ย้อนหลังและปรับปรุงระบบในอนาคต ทั้งนี้ แนวทางดังกล่าวสอดคล้องกับแนวคิดของระบบตรวจจับฟิชชิงแบบ Real-time ที่ผสาน Machine Learning เข้ากับกลไก Proxy และ Logging เพื่อเพิ่มประสิทธิภาพในการตรวจจับภัยคุกคาม (Rehman et al., 2025)

3. การพัฒนาโปรแกรมตรวจจับเว็บไซต์ฟิชชิง

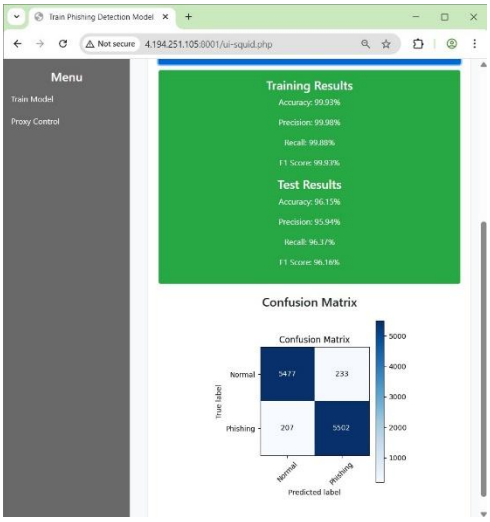
ระบบตรวจจับฟิชชิงนี้มีการพัฒนาส่วนติดต่อผู้ใช้ เพื่อให้ผู้ใช้โต้ตอบกับระบบได้ผ่านเว็บเบราว์เซอร์ ส่วนติดต่อผู้ใช้ของระบบถูกพัฒนาด้วย PHP, HTML, CSS (Bootstrap) และ JavaScript (jQuery) โดยที่ PHP ทำหน้าที่ประมวลผลฝั่งเซิร์ฟเวอร์และเชื่อมต่อกับโมดูลประมวลผลโมเดล, HTML ใช้สร้างโครงสร้างของหน้าเว็บ, CSS (Bootstrap) ช่วยจัดรูปแบบและ หน้าเว็บแสดงผลได้สวยงาม, JavaScript (jQuery) เพิ่มความสามารถในการโต้ตอบและการสื่อสารกับเซิร์ฟเวอร์แบบเรียลไทม์ ทำให้ผู้ใช้อัปโหลดข้อมูล ฝึกโมเดล และควบคุม Proxy ได้สะดวกผ่านเว็บเบราว์เซอร์ แบ่งออกเป็น 2 ส่วนหลัก ดังนี้

หน้าสำหรับการฝึกโมเดล ในหน้าหลักผู้ใช้สามารถอัปโหลดไฟล์ข้อมูลสำหรับการฝึกฝน ที่สามารถเลือกเปอร์เซ็นการเทรน และรูปแบบ Train/Test หรือ K-fold ดังภาพที่ 4(ก) หากเลือกการ Train/Test จะแสดงผลลัพธ์เป็นตัวชี้วัด Accuracy, Precision, Recall, F1-score ทั้งในชุด Training และ Testing และ Confusion ดังภาพที่ 4(ข) หรือเลือก K-Fold Cross Validation โดยระบุค่า K แสดงตัวอย่างผลลัพธ์การทดสอบแบบแบ่งข้อมูลเป็น 5 fold โดยแต่ละ fold จะรายงานค่า Accuracy, Precision, Recall และ F1-score แยกกัน และแสดง Confusion Matrix ดังภาพที่ 4(ค)

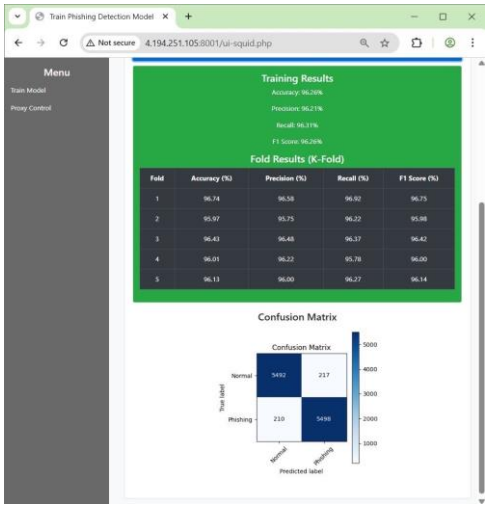
หน้าสำหรับจัดการ Proxy มีเมนู Upload Model (.pkl) สำหรับอัปโหลดโมเดลที่ฝึกแล้วเข้าสู่ระบบ Proxy และก่อนการอัปโหลดผู้ใช้จะต้องกดปุ่ม Stop proxy ก่อนการอัปโหลดโมเดล ดังภาพที่ 4(ง) หากอัปโหลดสำเร็จจะแสดงแจ้งเตือนดังภาพที่ 4(จ) เมื่อผู้ใช้เลือกเทรนโมเดล หรืออัปโหลดโมเดลเสร็จ จะต้องกดปุ่ม Start proxy เพื่อใช้งานโมเดลใหม่ และให้ระบบทำงานในการตรวจจับฟิชชิงแบบอัตโนมัติ ดังภาพที่ 4(ฉ) สำหรับการทดสอบให้เครื่อง Client ตั้งค่า IP และ Port มาที่พร็อกซี เพื่อใช้งานระบบ เมื่อทำการตั้งค่าระบบเสร็จสิ้น ผู้ใช้สามารถทดสอบการทำงานได้ โดยเข้าเว็บไซต์ฟิชชิงที่ใช้โปรโตคอล HTTP ระบบจะแสดงแจ้งเตือนการบล็อก ดังภาพที่ 4(ช) ในการบล็อกโปรโตคอล HTTP จะทำการ redirect ไปยังหน้า blocked.html ทำหน้าที่แจ้งเตือนผู้ใช้ว่าเว็บไซต์ดังกล่าวไม่ปลอดภัยและการเข้าถูกปฏิเสธ และเมื่อทดสอบกับเว็บไซต์ฟิชชิงที่ใช้โปรโตคอล HTTPS ระบบจะแสดงการแจ้งเตือนการบล็อกอีกแบบหนึ่ง ดังภาพที่ 4(ซ)



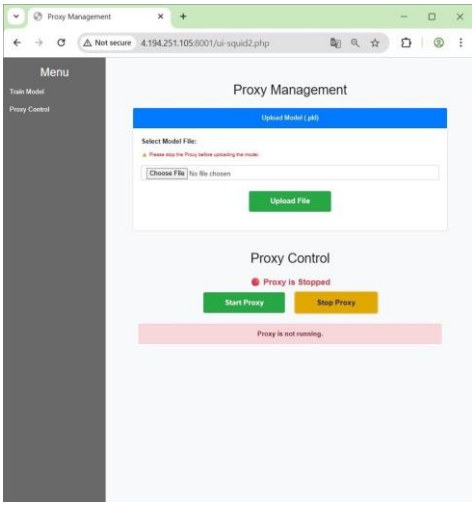
(ก) หน้าหลักโปรแกรม



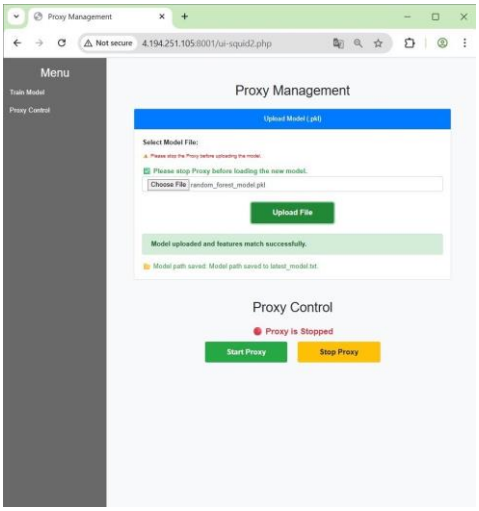
(ข) แสดงผลลัพธ์การเทรนแบบ train/test



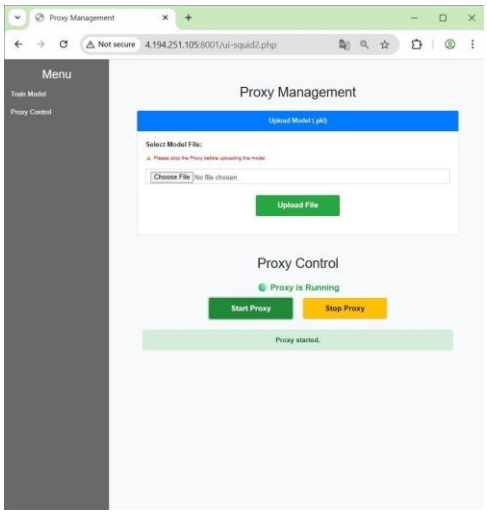
(ค) แสดงผลลัพธ์การเทรนแบบ K-Fold



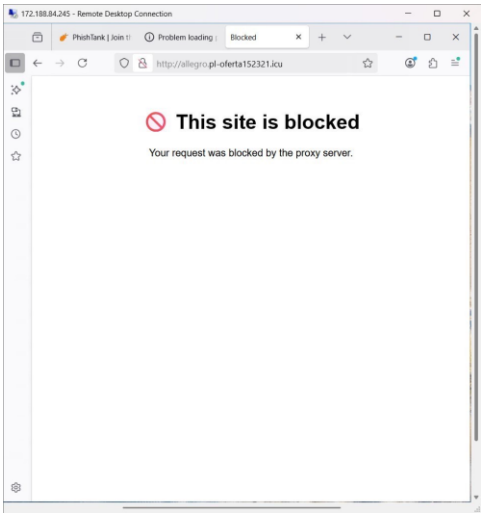
(ง) หน้าแสดงการกดปุ่ม Stop proxy



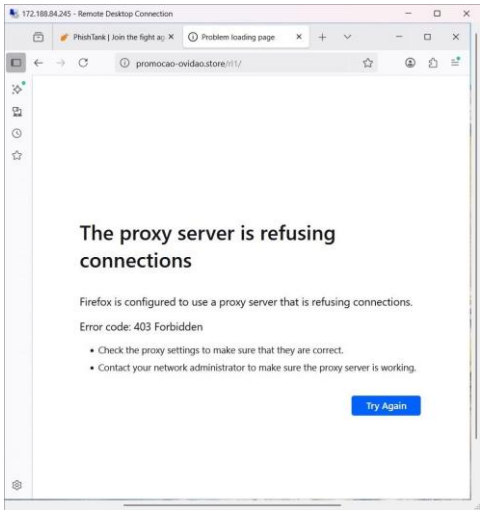
(จ) หน้าแสดงการอัปโหลดโมเดล .pkl



(ฉ) หน้าแสดงการกดปุ่ม Start proxy



(ช) หน้าแสดงการบล็อกโปรโตคอล HTTP



(ซ) หน้าแสดงการบล็อกโปรโตคอล HTTPS

ภาพที่ 4 หน้าจอการใช้งานโปรแกรม (ก) หน้าหลักโปรแกรม (ข) แสดงผลลัพธ์การเทรนแบบ train/test (ค) แสดงผลลัพธ์การเทรนแบบ หรือ K-Fold (ง) หน้าแสดงการกดปุ่ม Stop proxy (จ) หน้าแสดงการอัปโหลดโมเดล .pkl (ฉ) หน้าแสดงการกดปุ่ม Start proxy (ช) หน้าแสดงการบล็อกโปรโตคอล HTTP (ซ) หน้าแสดงการบล็อกโปรโตคอล HTTPS

4. การจัดเก็บฐานข้อมูล

งานวิจัยนี้ใช้ชุดข้อมูล (Dataset) จำนวน 3 ชุดสำหรับการฝึกและประเมินประสิทธิภาพของโมเดล Machine Learning ในการตรวจจับเว็บไซต์ฟิชชิง ได้แก่ (1) ชุดข้อมูลจาก GitHub ที่มีเว็บไซต์ฟิชชิงและปกติอย่างละ 5,715 รายการ รวม 11,430 รายการ (2) ชุดข้อมูลจาก Kaggle ที่ประกอบด้วยเว็บไซต์ฟิชชิง 10,747 รายการ และเว็บไซต์ปกติ 10,750 รายการ รวม 21,497 รายการ (Kaggle, 2024) และ (3) ชุดข้อมูลที่ผู้วิจัยรวบรวมเองจาก PhishTank และ Alexa Top Sites รวม 57,095 รายการ ซึ่งพบว่าให้ประสิทธิภาพสูงเมื่อนำมาใช้ฝึกและประเมินโมเดล (PhishTank, 2024) นอกจากนี้ งานวิจัยยังใช้ Blacklist และ Whitelist จาก GitHub เป็นฐานข้อมูลตั้งต้นในการกรองเว็บไซต์ก่อนผ่านเข้าสู่กระบวนการประมวลผลด้วยโมเดล Machine Learning (Salmi, 2024) เพื่อแก้ไขปัญหา False Positive ได้เพิ่มขั้นตอน canonicalization และการลบข้อมูลซ้ำทั้งแบบ exact และ near-duplicate เพื่อปรับโครงสร้าง URL ให้เป็นมาตรฐานและลดความคลาดเคลื่อนของข้อมูล รวมถึงการตรวจสอบความถูกต้องของรายการใน Blacklist/Whitelist โดยเทียบกับข้อมูลจาก PhishTank เพื่อเพิ่มความน่าเชื่อถือของฐานข้อมูลในขั้นตอนเตรียมข้อมูล (offline preprocessing)

สำหรับกลไก “Soft-Block” เป็นแนวคิดที่ช่วยลด False Positive โดยไม่ทำการบล็อกเว็บไซต์แบบเด็ดขาดเมื่อโมเดลมีความเชื่อมั่นต่ำ แต่เลือกใช้การแสดงหน้าเตือน (warning-page) พร้อมให้ผู้ใช้เป็นผู้ตัดสินใจว่าจะเข้าต่อหรือยกเลิก เช่น กลไกของ Google Safe Browsing ที่ใช้ interstitial warnings และแนวคิด human-in-the-loop ในงานด้าน usable security (Google Safe Browsing, 2021; Cranor, 2008) ผู้วิจัยยังไม่ได้นำ กลไก Soft Block มาใช้ในงานวิจัยแต่วางเป็นแนวทางพัฒนาในอนาคต เพื่อลดโอกาสเกิด False Positive และรองรับสถานการณ์ที่โมเดลมีความไม่แน่นอนสูง

5. เทคนิคที่ใช้พัฒนา

5.1 การเลือกและประมวลผลคุณลักษณะ (Feature Selection and Extraction)

การวิจัยนี้ได้ดำเนินการสกัดคุณลักษณะ (Feature Extraction) จากโครงสร้างของ URL ซึ่งเป็นแนวทางที่ได้รับการยืนยันแล้วว่าสามารถจำแนกเว็บไซต์ฟิชชิงได้อย่างมีประสิทธิภาพ ระบบได้พัฒนาโมดูลการสกัดคุณลักษณะจาก URL โดยตรง เพื่อแปลงข้อมูลในรูปแบบข้อความให้เป็นข้อมูลเชิงตัวเลข (Numeric Representation) สำหรับใช้เป็นอินพุตให้กับโมเดล Machine Learning โดยคุณลักษณะที่สกัดได้จำแนกออกเป็น 3 กลุ่มหลัก ดังนี้

(1) คุณลักษณะเชิงโครงสร้างของ URL (Structural Features) วิเคราะห์องค์ประกอบทั่วไปของ URL เพื่อบ่งชี้ความผิดปกติ เช่น ความยาวของ URL โครงสร้างโดเมน ความซับซ้อนของพาธ และจำนวนเครื่องหมายพิเศษที่อาจสื่อถึงพฤติกรรมหลอกลวง

(2) คุณลักษณะเชิงโดเมนและรูปแบบการตั้งชื่อ (Domain-based Features) ตรวจสอบความผิดปกติของชื่อโดเมน เช่น การใช้ IP Address แทนชื่อโดเมน การใช้โดเมนระดับบน (TLD) ที่ไม่พบบ่อย หรือโดเมนที่มีรูปแบบสุ่มสูงผิดปกติ

(3) คุณลักษณะด้านความปลอดภัยและคำสำคัญ (Security and Keyword Features) ตรวจจับคำหรือรูปแบบที่มักปรากฏในเว็บไซต์ฟิชชิง เช่น “login”, “verify”, “secure” หรือชื่อไฟล์ที่บ่งบอกถึงการหลอกลวง เช่น “account.html”, “secure.aspx”

ตารางที่ 1 คุณลักษณะที่ใช้ในการตรวจจับเว็บไซต์ฟิชชิง

ลำดับ	Feature Name	คำอธิบายภาษาไทย
1	url_length	ความยาวของ URL ทั้งสตริง
2	domain_length	ความยาวของโดเมน (ส่วน netloc)
3	path_length	ความยาวของพาธหลังโดเมน (เช่น /login/account.php)
4	num_subdomains	จำนวนโดเมนย่อย (เช่น a.b.example.com ⟶ 2)
5	num_dots	จำนวนเครื่องหมายจุด (.) ในทั้ง URL
6	num_hyphens	จำนวนเครื่องหมายขีด (-) ในทั้ง URL
7	num_slashes	จำนวนเครื่องหมายทับ (/) ในทั้ง URL
8	contains_at_symbol	การมีสัญลักษณ์ “@” ใน URL
9	contains_double_slash	พบ “//” ซ้ำในพาธ (ไม่รวม http(s)://)
10	path_has_extension	พาธลงท้ายด้วยนามสกุลไฟล์ เช่น .php, .aspx, .jsp
11	first_path_segment_length	ความยาวของพาธส่วนแรกหลัง “/”
12	num_query_params	จำนวนพารามิเตอร์ใน query string
13	num_digits	จำนวนตัวเลขทั้งหมดที่ปรากฏใน URL
14	is_ip_address	ระบุว่าโดเมนเป็นเลข IP (IPv4) หรือไม่ (1/0)
15	uncommon_tld	ใช้โดเมนระดับบน (TLD) ที่ไม่คุ้นเคย เช่น .xyz, .top, .icu
16	domain_entropy	ค่าเอนโทรปีของโดเมน (วัดความสุ่มของตัวอักษรในชื่อโดเมน)
17	has_suspicious_keyword	มีคำที่เข้าข่ายอันตราย เช่น “login”, “verify”, “secure”
18	has_suspicious_filename	มีชื่อไฟล์เสี่ยง เช่น “login.php”, “account.html”
19	has_hexadecimal	มีรหัส escape แบบเปอร์เซ็นต์ เช่น “%2F”, “%20”
20	is_long_url	URL มีความยาวเกินค่ามาตรฐานที่กำหนด (เช่น >75 ตัวอักษร)

คุณลักษณะทั้งหมดจำนวน 20 รายการ ถูกสกัดจากแต่ละ URL และถูกแปลงให้อยู่ในรูปแบบเชิงตัวเลข (Numeric Representation) เพื่อใช้เป็นข้อมูลนำเข้า (Input Features) ของโมเดล Random Forest (RF), Support Vector Machine (SVM), Multilayer Perceptron (MLP), Deep Neural Network (DNN) และ Extreme Gradient Boosting (XGBoost)

นอกจากนี้ งานวิจัยยังได้เพิ่มคุณลักษณะบางรายการที่ไม่ได้ถูกนำเสนอในงานก่อนหน้า เพื่อเพิ่มประสิทธิภาพของการจำแนกเว็บไซต์ฟิชชิง โดยมีตัวอย่างอัลกอริทึมการสร้างคุณลักษณะดังต่อไปนี้ (1) Domain Entropy ใช้ในการวัดระดับความซับซ้อนของชื่อโดเมน โดยคำนวณจากความถี่ของอักขระภายในโดเมน หากค่าความเอนโทรปีสูง แสดงว่าชื่อโดเมนนั้นมีรูปแบบไม่ปกติและอาจเป็นเว็บไซต์ฟิชชิง (2) First Path Segment Length ใช้ตรวจสอบความยาวของพาธส่วนแรกใน URL ซึ่งเว็บไซต์ฟิชชิงมักใช้พาธที่ยาวหรือมีรูปแบบซับซ้อนเพื่อหลอกผู้ใช้ (3) Suspicious Filename ใช้ตรวจจับชื่อไฟล์ที่มักปรากฏในเว็บไซต์ฟิชชิง เช่น login.php, account.html, หรือ secure.aspx ซึ่งมักถูกออกแบบให้คล้ายกับหน้าเข้าสู่ระบบของเว็บไซต์จริง

คุณลักษณะทั้งสามนี้ได้รับการออกแบบเพื่อเพิ่มศักยภาพของระบบในการวิเคราะห์รูปแบบเชิงซ้อนของ URL ทั้งในด้านโครงสร้างและพฤติกรรมการตั้งชื่อ ซึ่งช่วยให้การตรวจจับเว็บไซต์ฟิชชิงมีความแม่นยำและประสิทธิภาพสูงขึ้น โดยแสดงตัวอย่างการสร้างคุณลักษณะได้ดังนี้

```
domain_entropy = calculate_entropy(parsed.netloc)
first_path_segment_length = len(parsed.path.split('/')[1]) if '/' in parsed.path else 0
has_suspicious_filename = int(bool(re.search(r'(login|account)\.(php|html|aspx)', parsed.path)))
```

5.2 การพัฒนาแบบจำลอง Machine Learning

ในการทดลองได้ทดสอบหลายอัลกอริทึม ได้แก่ Random Forest (RF), Support Vector Machine (SVM), Multilayer Perceptron (MLP), Deep Neural Network (DNN) และ Extreme Gradient Boosting (XGBoost) เพื่อเปรียบเทียบประสิทธิภาพของการจำแนกเว็บไซต์ฟิชชิง ผลการทดลองพบว่า RF มีประสิทธิภาพสูงสุด โดยมี Accuracy 96.70% เนื่องจาก RF ใช้หลาย Decision Trees รวมผลการทำนาย ทำให้ลด Overfitting และเพิ่มความถูกต้อง

5.3 วิธีการทดสอบประสิทธิภาพของ Proxy Server (System Evaluation Method)

เพื่อประเมินความสามารถของ Proxy Server ในการรองรับปริมาณคำขอแบบเรียลไทม์ งานวิจัยได้ดำเนินการทดสอบเชิงระบบโดยจำลองสภาวะการใช้งานจริงผ่านการยิงคำขอ HTTP/HTTPS อย่างต่อเนื่องด้วยสคริปต์ Python ที่ใช้สถาปัตยกรรม asyncio บนเครื่องเสมือนสเปก 4 vCPU และ RAM 8 GB พร้อมบันทึกเวลาตอบสนองของคำขอแต่ละรายการและนำมาวิเคราะห์ในรูปของค่าร้อยเปอร์เซ็นไทล์ ได้แก่ P50 (ค่าหน่วงกึ่งกลาง), P95 (ค่าหน่วงของคำขอที่ช้าที่สุด 5%) และ P99 (ค่าหน่วงของคำขอที่ช้าที่สุด 1% สำหรับประเมิน tail latency) แนวทางดังกล่าวเป็นมาตรฐานในการประเมินระบบขนาดใหญ่ เนื่องจากช่วยลดข้อคลาดเคลื่อนที่เกิดจากค่าเฉลี่ยซึ่งอาจบดบังปัญหาความหน่วงปลายหางได้ (Dean & Barroso, 2013) สำหรับอัตราการรองรับคำขอ (Throughput) ใช้สมการ (1)

$$Throughput = \frac{\text{จำนวนคำขอที่ประมวลผลสำเร็จทั้งหมด}}{\text{ระยะเวลาในการทดสอบ (วินาที)}} \quad (1)$$

โดยค่าที่ได้ถูกวัดในหน่วย requests per second (req/sec) ซึ่งเป็นตัวชี้วัดมาตรฐานด้านประสิทธิภาพของระบบเว็บและโครงสร้างพื้นฐาน (NIST, 2018)

ตารางที่ 2 สรุปผลการทดสอบประสิทธิภาพเชิงระบบของ Proxy Server

Metric	ค่าโดยประมาณ
Throughput	1,000–1,500 req/sec
P50	~4–8 ms
P95	6–12 ms
P99	~12–20 ms

จากตารางที่ 2 Proxy Server สามารถรองรับคำขอได้ประมาณ 1,000–1,500 requests/sec ซึ่งสะท้อนศักยภาพของระบบในการประมวลผลคำขอจำนวนมากในระดับใกล้เคียงเรียลไทม์ ค่าความหน่วงที่ได้ ได้แก่ P50 4–8 ms, P95 6–12 ms และ P99 12–20 ms แสดงให้เห็นว่าระบบยังคงมีความหน่วงต่ำแม้ในกรณีคำขอที่ช้าที่สุดเพียง 1% ความแตกต่างระหว่างค่า P95 และ P99 มีเพียงเล็กน้อย ไม่พบคอขวดหรือความผิดปกติด้าน tail False ภายใต้โหลดสูง

5.4 การทำงานร่วมกันระหว่าง Proxy Server และโมเดล ML

ผู้วิจัยได้พัฒนา Proxy Server ด้วยภาษา Python ทำหน้าที่ตรวจสอบ URL ที่ผู้ใช้ร้องขอ โดยจะตรวจสอบกับ Whitelist และ Blacklist ก่อน หากไม่พบข้อมูล ระบบจะส่ง URL ไปยังโมเดล RF เพื่อทำนาย หากเป็นฟิชชิงจะถูกบล็อกด้วยรหัส 403 Forbidden สำหรับ HTTPS หรือ redirect ไปยังหน้า blocked.html สำหรับ HTTP แนวทางนี้สอดคล้องกับงานของ (Rehman et al., 2025) ที่ศึกษาและประเมินประสิทธิภาพของระบบ Proxy ที่ผสานกับ Machine Learning สำหรับการกรองเว็บไซต์และตรวจจับฟิชชิงแบบ Real-time ด้วยชุดคำสั่ง ดังนี้

```
if any(site in url for site in blacklist):
    self.send_response(403)
    self.end_headers()
    self.wfile.write(b"Blocked: Blacklist")
else:
    pred = model.predict([[len(url), url.count('@'), url.count('.')]])[0]
    if pred == 1:  # phishing
        self.send_response(403)
        self.end_headers()
        self.wfile.write(b"Blocked by RF model")
    else:
        self.send_response(200)
        self.end_headers()
        self.wfile.write(b"Access allowed")
```

ระบบ Proxy ที่พัฒนาในงานวิจัยนี้เป็น Explicit Proxy Server ซึ่งผู้ใช้ต้องทำการกำหนดค่า IP Address และ Port ของ Proxy ภายในเว็บเบราว์เซอร์หรือระบบปฏิบัติการก่อนเริ่มต้นใช้งาน โดย Proxy จะทำหน้าที่ตรวจสอบคำขอ URL ของผู้ใช้แบบเรียลไทม์ก่อนส่งต่อไปยังอินเทอร์เน็ต และการประเมินเชิงระบบ

งานวิจัยรายงาน ROC-AUC และ PR-AUC นอกเหนือจาก Accuracy/Precision/Recall/F1-score และแสดง calibration plot เมื่อเหมาะสม นอกจากนี้ประเมินเชิงระบบโดยประมาณ บนเครื่อง 4 cores, 8 GB RAM ระบบรองรับประมาณ 1,000-1,200 requests/sec โดย P95 latency ประมาณ 6-12 ms สำหรับการตรวจสอบแต่ละคำขอ (feature extraction + inference แบบ RF) ซึ่งจัดว่าเป็น near-real-time Reproducibility, Ethics และ License เพื่อความโปร่งใส รายงาน random seed (seed=42), ไลบรารี/เวอร์ชันหลัก, พารามิเตอร์โมเดล, และสคริปต์ preprocessing/canonicalization (แนบภาคผนวกหรือ repository) พร้อมระบุข้อกำหนดการใช้งานชุดข้อมูลต้นทางและแนวทางการปกปิดข้อมูลอ่อนไหว เช่น hashing/anonymization ของ IP

สำหรับการจัดการคำขอที่ใช้โปรโตคอล HTTPS ระบบจะใช้วิธีตรวจสอบชื่อโดเมนจากค่า SNI (Server Name Indication) โดยไม่ถอดรหัสข้อมูลภายใน (ไม่ทำ TLS interception) เพื่อคงไว้ซึ่งความเป็นส่วนตัวของผู้ใช้และหลีกเลี่ยงปัญหาการตรวจจับใบรับรอง (Certificate Pinning) จากเว็บไซต์จริง หาก URL ถูกระบุว่าเป็นเว็บไซต์ฟิชชิง ระบบจะตอบกลับด้วยรหัส 403 Forbidden โดยไม่อนุญาตให้เชื่อมต่อ แต่ในกรณีของ HTTP ระบบจะทำการ Redirect ไปยังหน้า blocked.html เพื่อแจ้งเตือนผู้ใช้ว่าเว็บไซต์ดังกล่าวไม่ปลอดภัย

6. ผลการดำเนินงาน

6.1 การทดสอบความแม่นยำของแบบจำลอง

การทดสอบความแม่นยำของแบบจำลองโดยใช้ 4 ตัวชี้วัดสำคัญ ดังนี้

การทดสอบความแม่นยำของแบบจำลองใช้ตัวชี้วัดทางสถิติที่เป็นมาตรฐานในงานวิจัยด้านการเรียนรู้ของเครื่อง ได้แก่ ความถูกต้อง (Accuracy), ความแม่นยำ (Precision), การเรียกคืน (Recall) และค่าความถ่วงดุล (F-measure หรือ F1-Score) ซึ่งนิยามและสมการพื้นฐานได้รับการยอมรับอย่างแพร่หลายในวรรณกรรมด้านการประเมินแบบจำลอง (Géron, 2022; Powers, 2011; Han et al., 2012) ดังแสดงในสมการ (2)–(5)

$$Acc = \frac{(TP+TN)}{(TP+TN+FP+FN)} \tag{2}$$

$$Precision = \frac{TP}{(TP+FP)} \tag{3}$$

$$Recall = \frac{TP}{(TP+FN)} \tag{4}$$

$$F-mesure = \frac{2\times Precision\times Recall}{Precision+Recall} \tag{5}$$

โดยที่ True Positive (TP) คือ สิ่งที่แบบจำลองทำนายว่า “จริง” และมีค่าเป็น “จริง” False Negative (FN) คือ สิ่งที่แบบจำลองทำนายว่า “จริง” และมีค่าเป็น “ไม่จริง” True Negative (TN) คือ สิ่งที่แบบจำลองทำนายว่า “ไม่จริง” และมีค่าเป็น “ไม่จริง” และ False Positive (FP) คือ สิ่งที่แบบจำลองทำนายว่า “ไม่จริง” และมีค่าเป็น “จริง”

โดยสรุป Random Forest (RF) เป็นอัลกอริทึมที่มีประสิทธิภาพสูงสุด โดยให้ค่า Accuracy 96.70%, Precision 97.21%, Recall 96.16% และ F1-score 96.68% ซึ่งแสดงถึงความสามารถในการจำแนกเว็บไซต์ฟิชชิงได้อย่าง

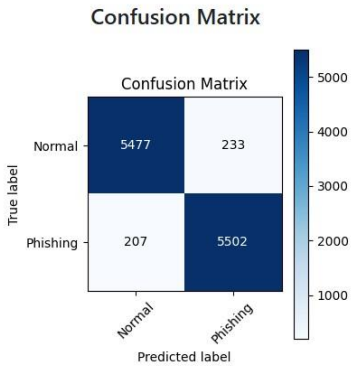
แม่นยำและมีความสมดุลระหว่าง Precision และ Recall มากที่สุด ลดข้อผิดพลาดทั้งแบบ False Positive และ False Negative ได้ดีกว่าอัลกอริทึมอื่น อัลกอริทึมที่มีผลลัพธ์ใกล้เคียงกับ RF ได้แก่ Deep Neural Network (DNN) และ Extreme Gradient Boosting (XGBoost) โดย DNN มีค่า Accuracy 96.41% และ Recall 96.72% สะท้อนถึงความสามารถในการตรวจจับฟิชชิงได้ครบถ้วน ขณะที่ XGBoost มีค่า Accuracy 96.41% และ Precision 96.79% เด่นด้านการลดโอกาสที่เว็บไซต์ปกติจะถูกระบุผิดว่าเป็นฟิชชิง ส่วน Multilayer Perceptron (MLP) มีค่า Accuracy 96.26% และ F1-score 96.26% อยู่ในระดับสูงแต่ยังต่ำกว่า RF เล็กน้อย และสุดท้าย Support Vector Machine (SVM) แสดงผลลัพธ์ต่ำที่สุด โดยมีค่า Accuracy 82.48% และ F1-score 82.60% ซึ่งไม่เหมาะสมสำหรับการใช้งานตรวจจับฟิชชิงเมื่อเทียบกับอัลกอริทึมอื่น ๆ

ตารางที่ 3 ความแม่นยำของประสิทธิภาพแบบจำลอง

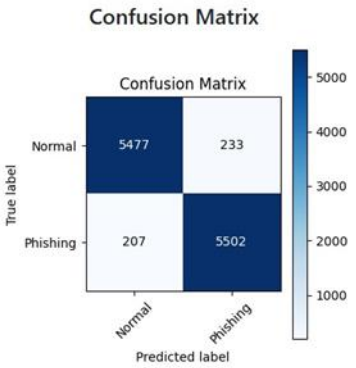
Algorithms	Accuracy (±SD)	Precision (±SD)	Recall (±SD)	F1-score (±SD)
Random Forest (RF)	96.70% ± 0.15	97.21% ± 0.20	96.16% ± 0.18	96.68% ± 0.17
Support Vector Machine (SVM)	82.48% ± 0.30	81.99% ± 0.25	83.24% ± 0.27	82.60% ± 0.28
Multi-Layer Perceptron (MLP)	96.26% ± 0.14	96.21% ± 0.16	96.31% ± 0.19	96.26% ± 0.18
Deep Neural Network (DNN)	96.41% ± 0.12	96.12% ± 0.17	96.72% ± 0.16	96.42% ± 0.15
Extreme Gradient Boosting (XGBoost)	96.41% ± 0.13	96.79% ± 0.18	96.00% ± 0.21	96.39% ± 0.16

หมายเหตุ: แบ่งข้อมูล Train/Test = 80/20 จำนวนข้อมูล Train = 45,676 รายการ, Test = 11,419 รายการ
ใช้ K-Fold Cross Validation = 5 (K = 5)

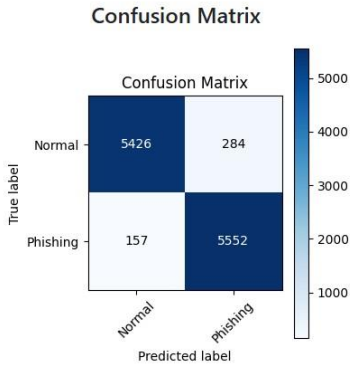
ผลการวิเคราะห์ Confusion Matrix ในภาพที่ 5(ก)–(จ) และค่าประสิทธิภาพเฉลี่ยจากการทดลองแบบ 5-fold สอดคล้องกันอย่างชัดเจน โดย Random Forest (RF) ให้ผลดีที่สุด ทั้งในเชิงตัวเลขและภาพรวมการจำแนก โดยมีค่า Accuracy เฉลี่ย 96.70 ± 0.15% และ F1-score 96.68 ± 0.17% พร้อมค่า True Positive และ True Negative สูงที่สุดในกลุ่ม แสดงถึงความสมดุลระหว่าง Precision และ Recall และสะท้อนความเสถียรของโมเดลจากค่า SD ที่ต่ำ อีกทั้งผลการทดสอบ paired t-test ยังยืนยันว่า RF มีความแตกต่างจากแบบจำลองอื่นอย่างมีนัยสำคัญ ($p < 0.05$) ด้าน Deep Neural Network (DNN) ในภาพที่ 5(ข) ให้ Recall สูงสุด (96.72 ± 0.16%) และมีจำนวนการทำนายคลาสฟิชชิงถูกต้องมากที่สุด จึงเหมาะกับงานที่ต้องการลด false negative ขณะที่ XGBoost ในภาพที่ 5(ค) ให้ Precision สูงที่สุด (96.79 ± 0.18%) และมีค่า False Positive ต่ำ เหมาะกับบริบทที่ต้องลดการแจ้งเตือนผิดพลาด ส่วน MLP ในภาพที่ 5(ง) ให้ผลลัพธ์ใกล้เคียง RF โดยมีค่า F1-score เฉลี่ย 96.26 ± 0.18% แม้ยังพบข้อผิดพลาดเล็กน้อยในคลาสฟิชชิง สำหรับ SVM ในภาพที่ 5(จ) มีประสิทธิภาพต่ำที่สุด (Accuracy 82.48 ± 0.30%, F1-score 82.60 ± 0.28%) และมีอัตราการจำแนกผิดทั้งคลาสปกติและคลาสฟิชชิงสูงกว่าแบบจำลองอื่น โดยสรุปทั้งผล Mean ± SD การวิเคราะห์ Confusion Matrix และผลการทดสอบทางสถิติต่างยืนยันว่า RF เป็นแบบจำลองที่เหมาะสมที่สุด รองลงมาคือ DNN และ XGBoost ซึ่งมีจุดเด่นเฉพาะด้านตามลำดับ



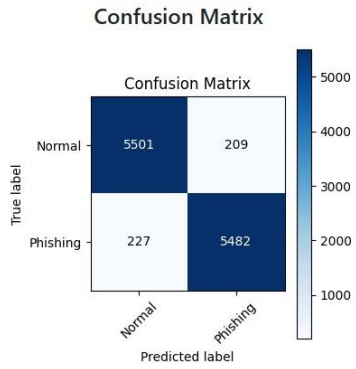
(ก) Confusion Matrix Random Forest



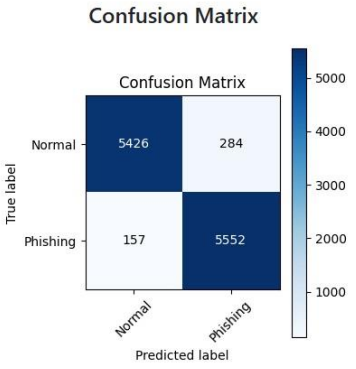
(ข) Confusion Matrix DNN



(ค) Confusion Matrix XGBoost



(ง) Confusion Matrix MLP



(จ) Confusion Matrix SVM

ภาพที่ 5 ภาพ Confusion Matrix (ก) Confusion Matrix Random Forest (ข) Confusion Matrix DNN (ค) Confusion Matrix XGBoost (ง) Confusion Matrix MLP และ (จ) Confusion Matrix SVM

6.2 การศึกษาการตัดทอนคุณลักษณะ (Ablation Study) ของกลุ่มฟีเจอร์และอัลกอริทึม

ได้ทำการศึกษาผลของการตัดทอนกลุ่มฟีเจอร์และเปรียบเทียบประสิทธิภาพของอัลกอริทึมหลัก ภายใต้เงื่อนไขการจูนพารามิเตอร์ที่เป็นธรรม โดยใช้เทคนิค Grid Search, Random Search และ Bayesian Optimization ร่วมกับค่า seed

จำนวน 5 ชุด เพื่อให้ได้ผลเฉลี่ยที่มีความเที่ยงตรง ครอบคลุมการทดสอบ 3 ระดับของชุดฟีเจอร์ ได้แก่ (1) Structural-only ซึ่งใช้เฉพาะฟีเจอร์เชิงโครงสร้างของ URL เช่น ความยาว จำนวนจุด และขีดกลาง (2) +Domain Entropy โดยเพิ่มฟีเจอร์เอนโทรปีของโดเมนเพื่อสะท้อนความสุ่มของชื่อโดเมน และ (3) +Keywords ซึ่งเพิ่มฟีเจอร์คำสำคัญ เช่น “login”, “secure”, และ “account” เพื่อจำแนกพฤติกรรมฟิชชิงที่อาศัยคำหลอกลวงในการโน้มน้าวผู้ใช้

ตารางที่ 4 ผลการทดลอง Ablation Study โดยเปรียบเทียบ ROC-AUC และ PR-AUC ของแต่ละชุดคุณลักษณะ

กลุ่มฟีเจอร์(Features)	Random Forest (RF)	XGBoost (XGB)	Deep Neural Network (DNN)
	ROC-AUC / PR-AUC	ROC-AUC / PR-AUC	ROC-AUC / PR-AUC
Structural-only	0.961 / 0.958	0.958 / 0.955	0.943 / 0.940
+Domain Entropy	0.982 / 0.981	0.985 / 0.983	0.974 / 0.970
+Keywords	0.993 / 0.993	0.992 / 0.992	0.988 / 0.987

การเพิ่มฟีเจอร์เอนโทรปีและคำสำคัญช่วยเพิ่มประสิทธิภาพของโมเดลอย่างชัดเจน โดยกลุ่มฟีเจอร์ +Keywords ให้ค่าทั้ง ROC-AUC และ PR-AUC สูงสุด (≈ 0.99) แสดงถึงความสามารถในการจำแนกเว็บไซต์ฟิชชิงได้อย่างแม่นยำ เมื่อเปรียบเทียบอัลกอริทึม พบว่า Random Forest และ XGBoost ให้ผลลัพธ์ใกล้เคียงและเสถียรกว่า DNN ซึ่งมีความผันผวนเล็กน้อยจากการสุ่มค่าเริ่มต้น

7. สถิติที่ใช้ในการวิจัย

งานวิจัยนี้ใช้สถิติที่สำคัญได้แก่ ค่าเฉลี่ย และส่วนเบี่ยงเบนมาตรฐาน โดยนำผลที่ได้เทียบกับเกณฑ์การประเมิน (ลัดดา, 2562) ปรากฏดังนี้คือ

ค่าเฉลี่ยเท่ากับ 4.51-5.00 หมายความว่า ระดับมากที่สุด
ค่าเฉลี่ยเท่ากับ 3.51-4.50 หมายความว่า ระดับมาก
ค่าเฉลี่ยเท่ากับ 2.51-3.50 หมายความว่า ระดับปานกลาง
ค่าเฉลี่ยเท่ากับ 1.51-2.50 หมายความว่า ระดับน้อย
ค่าเฉลี่ยเท่ากับ 1.01-1.50 หมายความว่า ระดับน้อยที่สุด

ผลการวิจัย และวิจารณ์ผลการวิจัย

จากผลการประเมินทั้ง 4 ด้าน ได้แก่ (1) การทำงานของ UI (2) ความถูกต้องของการจำแนกเว็บไซต์ฟิชชิง (3) ความสามารถในการควบคุม Proxy และ (4) การอนุญาตเว็บไซต์ปกติ/บล็อกเว็บไซต์ฟิชชิง พบว่าระบบสามารถทำงานได้ถูกต้องในช่วง 98–100% โดย UI และการควบคุม Proxy ให้ผลลัพธ์สมบูรณ์ 100% ส่วนการจำแนกเว็บไซต์ฟิชชิงมีความถูกต้องเฉลี่ย 99% แม้บางรอบมีค่าลดลงเหลือ 90% เนื่องจากการสลับป้ายกำกับ ขณะที่การอนุญาตเว็บไซต์ปกติและการบล็อกเว็บไซต์ฟิชชิงมีค่าความถูกต้องเฉลี่ย 98% เมื่อรวมทุกด้าน ระบบมีประสิทธิภาพเฉลี่ยประมาณ 99% สะท้อนถึงความเสถียรและความเหมาะสมต่อการนำไปใช้งานจริงในระดับองค์กร

ผลการทดสอบเชิงระบบตามวิธีที่อธิบายไว้ในหัวข้อ 5.3 พบว่า Proxy Server สามารถรองรับคำขอได้ประมาณ 1,000–1,500 requests/sec บนเครื่องทดสอบที่ใช้ 4 vCPU และ RAM 8 GB โดยมีค่าความหน่วงอยู่ในระดับต่ำ ได้แก่ P50 ประมาณ 4–8 ms, P95 ประมาณ 6–12 ms และ P99 ประมาณ 12–20 ms ผลต่างระหว่างค่า P95 และ P99 มีเพียงเล็กน้อย จึงไม่พบสัญญาณของคอขวดหรือปัญหา tail latency ในคำขอที่ช้าที่สุด 1% ผลลัพธ์ดังกล่าว สรุปในตารางที่ 1 แสดงให้เห็นว่าระบบมีความเสถียรและรองรับการประมวลผลคำขอจำนวนมากได้อย่างเหมาะสมสำหรับการใช้งาน

ตารางที่ 5 ผลการทดสอบประสิทธิภาพระบบตรวจจับเว็บไซต์ฟิชชิงแบบเรียลไทม์โดยใช้แมชชีนเลิร์นนิง

ลำดับ	เกณฑ์การประเมิน	ผลการทดสอบ (ครั้งที่)										สรุป (%)
		1	2	3	4	5	6	7	8	9	10	
1	อัปโหลดข้อมูลผ่าน UI	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	**100%**
2	การจำแนกเว็บไซต์ฟิชชิง	100%	100%	100%	100%	90%	100%	100%	100%	100%	100%	**99%**
3	การควบคุม Proxy ผ่าน UI	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	**100%**
4	การอนุญาตเว็บไซต์ปกติและการบล็อกเว็บไซต์ฟิชชิง	100%	100%	100%	90%	100%	100%	90%	100%	100%	100%	**98%**

งานวิจัยนี้ได้แสดงให้เห็นถึงศักยภาพของการผสาน Machine Learning เข้ากับ Proxy Server เพื่อให้สามารถตรวจจับเว็บไซต์ฟิชชิงแบบเรียลไทม์ นับเป็นก้าวสำคัญจากงานเชิงทฤษฎีสู่การใช้งานจริงในสภาพแวดล้อมองค์กร พร้อมทั้งชี้ให้เห็นข้อจำกัดและทิศทางการพัฒนาระบบเพื่อยกระดับความปลอดภัยและความทนทานในอนาคต และเสนอแนวทางพัฒนาในอนาคต ได้แก่ การตรวจจับ concept drift ตามการเปลี่ยนแปลงของรูปแบบฟิชชิง การปรับปรุงชุดข้อมูลอัตโนมัติ การตรวจสอบ URL ซ้ำกับ threat intelligence ภายนอก และการเสริมระบบประเมินระดับความเชื่อมั่นของโมเดลเพื่อควบคุม False Positive/False Negative นอกจากนี้ ผู้วิจัยมีแผนเพิ่มกลไก “Soft-Block” ซึ่งจะเตือนผู้ใช้เมื่อโมเดลมีความมั่นใจต่ำแทนการบล็อกทันที (Hard-Block) เช่นในเวอร์ชันปัจจุบัน เพื่อให้ระบบเหมาะสมกับการนำไปใช้งานจริง

ผลลัพธ์จากการสอบถามความพึงพอใจของผู้ใช้งานระบบตรวจจับเว็บไซต์ฟิชชิง จำนวน 10 คน แสดงดังตารางที่ 6 พบว่า ผู้ใช้งานมีความพึงพอใจอยู่ในระดับ มากที่สุด โดยเฉพาะด้านความน่าเชื่อถือในการอนุญาตและบล็อกเว็บไซต์ รวมถึงความเสถียรและความต่อเนื่องของระบบ ซึ่งได้รับค่าคะแนนเฉลี่ยสูงสุด สะท้อนถึงความมั่นใจของผู้ใช้ว่าระบบสามารถตรวจจับและจัดการเว็บไซต์ฟิชชิงได้อย่างมีประสิทธิภาพและเชื่อถือได้ นอกจากนี้ ผู้ใช้งานยังมีความพึงพอใจในระดับมากที่สุดในด้าน ความถูกต้องและความรวดเร็วในการจำแนกเว็บไซต์ฟิชชิง การควบคุม Proxy ผ่าน UI ความครบถ้วนของการแสดงผลลัพธ์การประเมินโมเดล และ ความสะดวกและง่ายในการใช้งานระบบ บ่งชี้ว่าระบบมีความพร้อมทั้งในด้านการทำงาน ความถูกต้อง และความเสถียรในการใช้งานจริง อย่างไรก็ตาม พบว่าด้าน ความสามารถของ UI ในการอัปโหลดและฝึกโมเดล ยังอยู่ในระดับ มาก บ่งชี้ว่าผู้ใช้งานบางส่วนเห็นว่ายังมีโอกาสในการพัฒนาให้มีความยืดหยุ่นและรองรับการใช้งานได้สะดวกยิ่งขึ้น

ตารางที่ 6 คะแนนความพึงพอใจของผู้ใช้งาน

หัวข้อประเมิน	$\bar{x}$	S.D.	ระดับความคิดเห็น
1. ความสะดวกและง่ายในการใช้งานของระบบ	4.82	0.39	มากที่สุด
2. ความสามารถของระบบ UI ในการอัปโหลด	4.36	0.67	มาก
3. ความถูกต้องและความรวดเร็วในการจำแนกเว็บไซต์ฟิชชิง	4.64	0.45	มากที่สุด
4. ความน่าเชื่อถือในการอนุญาตเว็บไซต์ปกติและการบล็อกเว็บไซต์ฟิชชิง	4.91	0.30	มากที่สุด
5. การควบคุม Proxy ผ่าน UI	4.64	0.50	มากที่สุด
6. การแสดงผลลัพธ์การประเมินบน UI	4.64	0.50	มากที่สุด
7. ความเสถียรในการทำงานของระบบ	4.91	0.30	มากที่สุด
คะแนนเฉลี่ยโดยรวม	4.75	0.43	มากที่สุด

วิจารณ์ผลการวิจัย

ในหัวข้อนี้ ผู้วิจัยได้เลือกองค์ประกอบหลักที่สำคัญของการพัฒนาระบบตรวจจับเว็บไซต์ฟิชชิงมาอภิปรายผล เพื่อเปรียบเทียบกับงานวิจัยที่เกี่ยวข้อง โดยพิจารณาถึงข้อดี ข้อจำกัด และข้อแตกต่าง ซึ่งแบ่งออกเป็น 2 องค์ประกอบ ได้แก่ (1) การบูรณาการ Proxy Server เข้ากับ Machine Learning และ (2) การทำงานของระบบผ่าน UI

องค์ประกอบแรก คือ การบูรณาการ Proxy Server เข้ากับ Machine Learning งานวิจัยสมัยใหม่ระหว่างปี 2023–2025 ส่วนใหญ่ยังคงประเมินประสิทธิภาพของโมเดลในลักษณะ ออฟไลน์ หรือในสภาพแวดล้อมจำลอง โดยเน้นการใช้ฟีเจอร์จากโครงสร้างของ URL หรือการใช้โมเดลเชิงลึก เช่น งานของ Asiri et al. (2024) ที่นำเสนอ PhishTransformer สำหรับวิเคราะห์ลักษณะ URL, งานของ Xu (2021) ที่ผสาน Deep Learning กับ Transformer และงานของ Haq et al. (2024) ที่ประเมินโครงข่ายประสาทเทียมบนข้อมูลจริง รวมถึงงานของ Rao et al. (2025) ที่พัฒนาโมเดลผสมผสานสำหรับอุปกรณ์โมบาย แม้ผลการทดลองในงานเหล่านี้จะให้ค่าความแม่นยำสูง แต่ยังคงเป็นการทดสอบภายใต้เงื่อนไขควบคุม ไม่ได้ประเมินในระบบจริงที่ต้องรองรับแบบเรียลไทม์

องค์ประกอบที่สอง คือ การทำงานผ่าน UI ซึ่งถูกออกแบบให้ผู้ใช้อัปโหลดชุดข้อมูลหรือโมเดลสำเร็จรูป (.pkl) ทำการฝึก (train) และประเมินผลโมเดลได้ด้วยตนเอง พร้อมทั้งเลือกใช้อัลกอริทึมและดูผลการประเมิน เช่น Accuracy, Precision, Recall และ F1-score ได้โดยไม่จำเป็นต้องมีทักษะด้านการเขียนโปรแกรม งานของ Haq et al. (2024) ได้นำเสนอแนวคิด Explainable AI ด้วยเทคนิค LIME และ SHAP เพื่ออธิบายผลลัพธ์ของโมเดล และงานของ Le et al. (2022) ได้เสนอการตรวจจับฟิชชิงด้วย Graph-based Learning ที่สามารถวิเคราะห์ความสัมพันธ์ระหว่างโดเมนและ URL ได้อย่างมีประสิทธิภาพ งานดังกล่าวยังขาดการนำเสนอระบบในรูปแบบส่วนติดต่อผู้ใช้ที่รองรับการฝึก การประเมิน และการควบคุมการทำงานของโมเดลได้โดยตรง ในขณะที่งานวิจัยนี้ได้พัฒนา UI ให้รองรับทั้งการจัดการโมเดลและการควบคุม Proxy Server ได้ภายในระบบเดียว ผู้ใช้สามารถเปิด-ปิด Proxy และตรวจสอบผลการทำงานได้แบบเรียลไทม์ ซึ่งเป็นข้อแตกต่างที่ช่วยให้ระบบนี้นำไปประยุกต์ใช้งานจริงได้ดีกว่า

สรุปผลการวิจัย

งานวิจัยนี้พัฒนาระบบตรวจจับเว็บไซต์ฟิชชิงโดยผสาน Proxy Server ที่พัฒนาด้วยภาษา Python เข้ากับโมเดลการเรียนรู้ของเครื่อง พร้อมส่วนติดต่อผู้ใช้สำหรับอัปโหลดข้อมูล ฝึกโมเดล และควบคุมการทำงานของ Proxy ได้โดยตรง ผลการทดสอบพบว่าโมเดล Random Forest (RF) ให้ประสิทธิภาพดีที่สุด โดยมี Accuracy 96.70% และ F1-score 96.68% สามารถจำแนกเว็บไซต์ฟิชชิงแบบเรียลไทม์ได้อย่างแม่นยำและมีเสถียรภาพ ระบบบันทึกข้อมูลการใช้งานลงใน Log เพื่อรองรับการวิเคราะห์ย้อนหลัง และจากผลการประเมินความพึงพอใจของผู้ใช้งานพบว่าระบบได้รับคะแนนในระดับ “มากที่สุด” โดยเฉพาะด้านความสะดวกในการใช้งานและความเสถียรของการประมวลผล แสดงให้เห็นว่าสามารถนำไปประยุกต์ใช้ในสภาพแวดล้อมจริงขององค์กรได้

อย่างไรก็ตาม ระบบยังมีข้อจำกัดบางประการ ทั้งในด้านการตรวจจับที่ยังอาศัยโครงสร้างของ URL เป็นหลักและไม่รองรับการวิเคราะห์ทราฟฟิก HTTPS ทำให้ไม่สามารถระบุฟิชชิงที่ใช้การเปลี่ยนเส้นทางได้ครบถ้วน อีกทั้งโมเดลยังไม่ปรับตัวตามรูปแบบการโจมตีที่เปลี่ยนแปลง และระบบต้นแบบใช้การบล็อกแบบทันที ซึ่งเพิ่มโอกาสเกิดผลกระทบจาก False Positive โดยเฉพาะในกรณีที่โมเดลมีความเชื่อมั่นต่ำ ชี้ให้เห็นว่ายังมีความจำเป็นต้องเสริมความยืดหยุ่นของระบบสำหรับการใช้งานจริง

ในอนาคตควรพัฒนากลไก Soft-Block (SB) เพื่อทำหน้าที่เป็นชั้นป้องกันเมื่อโมเดลมีความเชื่อมั่นต่ำ โดยให้ผู้ใช้ยืนยันการเข้าถึงแทนการบล็อกทันทีเช่นในระบบต้นแบบ ซึ่งจะช่วยลดผลกระทบจาก False Positive และรองรับการจัดการกรณี low-confidence predictions ได้ดียิ่งขึ้น นอกจากนี้ ควรเสริมให้ระบบรองรับ TLS Interception เพื่อให้ตรวจสอบพฤติกรรมภายในทราฟฟิก HTTPS ได้อย่างครอบคลุมมากขึ้น

ข้อเสนอแนะ

จากผลการวิจัยพบว่า ระบบตรวจจับเว็บไซต์ฟิชชิงที่พัฒนาขึ้นมีศักยภาพในการนำไปใช้งานจริง เพื่อเพิ่มประสิทธิภาพและความพร้อมในการใช้งานในอนาคต ควรมีการพัฒนาต่อยอดในหลายด้าน อาทิ การเพิ่มฟังก์ชันอัปเดตฐานข้อมูล URL แบบอัตโนมัติเพื่อรองรับการโจมตีรูปแบบใหม่ การปรับปรุงส่วนติดต่อผู้ใช้ (UI) ให้เหมาะสมกับผู้ใช้งานที่มีพื้นฐานทางเทคโนโลยีที่แตกต่างกัน และการเสริมศักยภาพของ Proxy Server ให้รองรับคำร้องขอจำนวนมากได้อย่างมีประสิทธิภาพในสภาพแวดล้อมองค์กรขนาดใหญ่ ซึ่งเป็นข้อเสนอแนะสำหรับการศึกษาครั้งต่อไป

เอกสารอ้างอิง

ลัดดา ภู่เกษม. (2562). *สถิติสำหรับการวิจัยทางการศึกษา*. กรุงเทพมหานคร: สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย.

Alkhalil, Z., Hewage, C., Nawaf, L., & Khan, I. (2021). Phishing attacks: A recent comprehensive study and a new anatomy. Frontiers in Computer Science, 3, Article 563060. https://doi.org/10.3389/fcomp.2021.563060

Anupam, S., & Kar, A. K. (2021).Phishing website detection using support vector machines and nature-inspired optimization algorithms. Telecommunication Systems, 76(1), 17–32. https://doi.org/10.1007/s11235-020-00739-w

Anti-Phishing Working Group. (2023). Phishing Activity Trends Report 2023. Retrieved from https://apwg.org/trendsreports/

Asiri, S., Xiao, Y., & Li, T. (2024).PhishTransformer: A novel approach to detect phishing attacks using URL collection and transformer. Electronics, 13(1), 30. https://doi.org/10.3390/electronics13010030

Chen, Q., Li, X., & Wang, Y. (2022).Phishing website detection based on XGBoost algorithm. Journal of Information Security and Applications, 66, 103149. https://doi.org/10.1016/j.jisa.2022.103149

Cranor, L. F. (2008). A framework for reasoning about the human in the loop. In Proceedings of the 1st Conference on Usability, Psychology, and Security (UPSEC ’08) (pp. 1–15). USENIX Association. Retrieved from https://www.usenix.org/legacy/event/upsec08/tech/full_papers/cranor/cranor.pdf

Dean, J., & Barroso, L. A. (2013). The tail at scale. Communications of the ACM, 56(2), 74–80. https://doi.org/10.1145/2408776.2408794

Fielding, R., Reschke, J., & Berners-Lee, T. (2022). Hypertext Transfer Protocol (HTTP/1.1): Semantics and content. IETF RFC 9110. https://doi.org/10.17487/RFC9110

Géron, A. (2022). Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow (3rd ed.). O'Reilly Media.

Google. (2021). Safe Browsing: Warning pages and protective behavior. Google Security Blog. Retrieved from https://safebrowsing.google.com/

Han, J., Kamber, M., & Pei, J. (2012). Data Mining: Concepts and Techniques (3rd ed.). Morgan Kaufmann.

Haq, Q. E. ul, Faheem, M. H., & Ahmad, I. (2024). Detecting phishing URLs based on a deep learning approach to prevent cyber-attacks. Applied Sciences, 14(22), 10086. https://doi.org/10.3390/app142210086

Harinahalli Lokesh, G., & BoreGowda, G. (2021). Phishing website detection based on effective machine learning approach. Journal of Cyber Security Technology, 5(1), 1–14. https://doi.org/10.1080/23742917.2020.1813396

Kaggle. (2024). Phishing websites dataset for machine learning. Retrieved from https://www.kaggle.com/datasets

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. Proceedings of the International Joint Conference on Artificial Intelligence, 1137–1145.

Le, T. D., Pham, D. T., & Le, H. T. (2022). Detecting phishing websites using graph-based features and machine learning. Computers & Security, 112, 102508. https://doi.org/10.1016/j.cose.2021.102508

Mozilla. (2023). Server Name Indication (SNI) documentation. Retrieved from https://wiki.mozilla.org/Security

Mushtaq, S., Javed, T., & Mohd Su'ud, M. (2024). Ensemble learning-powered URL phishing detection: A performance driven approach. Journal of Informatics and Web Engineering, 3(2), 134–145. https://doi.org/10.33093/jiwe.2024.3.2.10

National Institute of Standards and Technology. (2006). NIST SP 800-92: Guide to Computer Security Log Management. https://doi.org/10.6028/NIST.SP.800-92

Narváez, A., Curipallo, M., Reyes, E., Lara, F., Reyes, E. P., & Barba, M. (2025). Evaluation framework for false positives in open-source WAFs based on OWASP CRS paranoia levels. Engineering Proceedings, 115(1), 1. https://doi.org/10.3390/engproc2025115001

OpenPhish. (2024). Phishing Intelligence Feeds. Retrieved from https://openphish.com

OWASP. (2023). OWASP Secure Coding Practices. Retrieved from https://owasp.org

PhishTank. (2024). Verified phishing data. Retrieved from https://phishtank.com

Powers, D. M. W. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. International Journal of Machine Learning Technology, 2(1), 37–63. Retrieved from https://arxiv.org/abs/2010.16061

Rao, R. S., Kondaiah, C., Pais, A. R., et al. (2025). A hybrid super learner ensemble for phishing detection on mobile devices. Scientific Reports, 15, 16839. https://doi.org/10.1038/s41598-025-02009-8

Raschka, S., & Mirjalili, V. (2020). Python Machine Learning (3rd ed.). Packt Publishing.

Rehman, A. U., Imtiaz, I., Javaid, S., & Muslih, M. (2025). Real-time phishing URL detection using machine learning. Engineering Proceedings, 107, 108. https://doi.org/10.3390/engproc2025107108

Salmi, M. (2024). GitHub phishing blacklist/whitelist repository. Retrieved from https://github.com/salmi/phishing-lists

Xu, P. (2021). A transformer-based model to detect phishing URLs. arXiv preprint arXiv:2109.02138. https://doi.org/10.48550/arXiv.2109.02138

Yang, R., Zheng, K., Wu, B., Wu, C., & Wang, X. (2021). Phishing website detection based on deep convolutional neural network and random forest ensemble learning. Sensors, 21(24), 8281. https://doi.org/10.3390/s21248281