



كلية الهندسة المعلوماتية

تخصص أمن المعلومات والشبكات

Enhancing phishing detection model using Machine Learning

إعداد

لانا حاتول

نغم عطية

إشراف

د. وسيم الجندي

م. آية الأسود

2025

الملخص

يركز هذا البحث على تطوير نظام فعال لكشف هجمات التصيد الاحتيالي اعتماداً على تحليل روابط الويب (**URLs**) باستخدام تقنيات تعلم الآلة. يهدف النظام المقترن إلى تجاوز قيود الأساليب التقليدية، مثل القوائم السوداء، من خلال الاعتماد على التحليل البنوي والنصي للرابط نفسه دون الحاجة إلى فحص محتوى الصفحة أو الاعتماد على مصادر خارجية.

يعتمد النظام على استخراج نوعين من التمثيلات من الرابط: التمثيل النصي الخام باستخدام **TF-IDF** على مستوى المحارف (**character n-grams**)، والتمثيل الجدولي الذي يتضمن مجموعة من الخصائص البنوية والإحصائية المستخرجة من بنية الرابط. يتم دمج هذين التمثيلين ضمن متجر ميزات هجين (**Hybrid Feature Vector**) بهدف الاستفادة من المعلومات النصية الدقيقة والخصائص البنوية الصريحة في آن واحد.

تم تدريب نموذج **Logistic Regression** على التمثيل الهجين، مع معالجة مشكلة عدم توازن الفئات باستخدام آلية إعادة وزن الفئات (**class weighting**)، لما توفره من استقرار أفضل مقارنةً بأساليب التوليد الاصطناعي للبيانات. كما تم ضبط عتبة القرار (**Decision Threshold**) بناءً على نتائج مجموعة التحقق لتحقيق توازن أفضل بين الدقة والاسترجاع، بما يتناسب مع طبيعة أنظمة الحماية الأمنية.

لتقييم أداء النظام، تم استخدام مجموعة من المقاييس القياسية، بما في ذلك **Precision** و**Accuracy** و**Recall** و**F1-Score** و**ROC-AUC**. إضافةً إلى ذلك، تم توظيف إطار **SHAP** لتفسير قرارات النموذج وتحليل مساهمة الخصائص المختلفة في عملية التصنيف، مما يعزز من شفافية النظام وقابليته للفهم.

وأخيراً، تم دمج النموذج النهائي ضمن واجهة تفاعلية مبنية باستخدام **Streamlit**، تتيح للمستخدم اختبار الروابط بشكل مباشر والحصول على نتائج فورية، مما يجعل النظام المقترن قابلاً للتطبيق العملي في بيئات الإنترنت الحقيقية.

Abstract

This research focuses on developing an effective phishing detection system based on URL analysis using machine learning techniques. The proposed system aims to overcome the limitations of traditional approaches, such as blacklists, by relying solely on the lexical and structural characteristics of URLs without requiring webpage content analysis or external reputation services.

The system extracts two complementary representations from each URL: a raw textual representation using character-level TF-IDF n-grams, and a set of handcrafted tabular features capturing structural and statistical properties of the URL. These representations are combined into a hybrid feature vector to leverage both fine-grained textual patterns and explicit structural indicators of phishing behavior.

A Logistic Regression classifier is trained on the hybrid representation, with class imbalance handled through class weighting to improve sensitivity toward the minority class. Decision threshold tuning is applied based on validation data to achieve a better balance between precision and recall, reflecting the asymmetric cost of misclassification in security-oriented systems.

The performance of the proposed system is evaluated using standard metrics, including Accuracy, Precision, Recall, F1-Score, and ROC-AUC. Furthermore, the SHAP framework is employed to interpret model predictions and analyze feature contributions, enhancing the transparency and explainability of the classification process.

Finally, the trained model is integrated into an interactive interface developed using Streamlit, allowing real-time URL testing and demonstrating the practical applicability of the proposed phishing detection system in real-world scenarios.

فهرس المحتويات Table of Contents

الفصل الأول

7	1.1 مقدمة
7	1.2 مشكلة البحث
8	1.3 أهداف البحث
8	1.4 أهمية البحث
9	1.5 منهجية البحث
10	1.6 الدراسات المشابهة
22	1.7 التحديات
24	1.8 التطبيقات

الفصل الثاني

26	2.1 مفهوم التصيد الاحتيالي
26	2.2 الكشف عن التصيد الاحتيالي بالاعتماد على روابط URL
27	2.3 مبررات التركيز على تحليل روابط URL
27	2.4 الطرق التقليدية لكشف التصيد وعيوبها
28	2.5 دوافع الانتقال إلى تعلم الآلة
28	2.6 الشفافية وقابلية التفسير

الفصل الثالث

30	3.1 مقدمة الفصل
30	3.2 المقارنة بين بيانات العمل المتاحة
31	3.3 تبرير اختيار Visual Studio Code (VS Code)
32	3.4 توصيف عملية التنصيب

3.5 إعداد بيئة التنفيذ البرمجية 33
3.6 تصميم وتنفيذ خط المعالجة البرمجية 35
3.7 تنفيذ التمثيلات المعتمدة للبيانات 35
3.8 حفظ النماذج وإعادة استخدامها 36
3.9 المكتبات المستخدمة 37

الفصل الرابع

4.1 مقدمة الفصل 40
4.2 نظرة عامة على الحل المقترن 40
4.3 بنية النظام المقترن وخطط العمل 41
4.4 جمع البيانات ومصادرها 43
4.5 المعالجة المسبقة للبيانات 44
4.6 تمثيل البيانات 45
4.7 الخوارزميات المستخدمة ومنهجية اختيار النموذج النهائي 46
4.8 مقاييس التقييم المعتمدة في النظام المقترن 52
4.9 اختيار العتبة 53
4.10 التعامل مع عدم توازن الفئات وتسرب البيانات 54
4.11 واجهة المستخدم التفاعلية 55

الفصل الخامس

5.1 مقدمة الفصل 58
5.2 خلاصة النتائج التجريبية 58
5.3 الآفاق المستقبلية للنظام المقترن 59

فهرس الجداول

الجدول (1) - مقارنة بين الدراسات السابقة 21
الجدول (2) - مقارنة بيانات العمل المتاحة 31
الجدول (3) - مصادر مجموعات البيانات المستخدمة 42
الجدول (4) - مقارنة شاملة لنتائج التدريب 47
الجدول (5) - مقاييس تقييم الخوارزميات 54

فهرس الأشكال

الشكل 1 - النظام المقترن لكشف عن التصيد الاحتيالي 9
الشكل 2 - مخطط عمل النظام 42
الشكل 3 - مخطط النموذج النهائي 51
الشكل 4 - واجهة المشروع 56

الفصل الأول : المقدمة

1.1 مقدمة

أصبحت الهجمات الإلكترونية في السنوات الأخيرة أكثر تعقيداً وتنوعاً، ويُعد التصيد الاحتيالي (Phishing) من أخطر هذه الهجمات وأكثرها انتشاراً، نظراً لاعتماده على الهندسة الاجتماعية لخداع المستخدمين ودفعهم إلى الكشف عن معلومات حساسة مثل بيانات تسجيل الدخول أو المعلومات المالية. لا تقتصر خطورة هجمات التصيد على الخسائر المادية فقط، بل تمتد لتشمل تهديد الخصوصية، وسرقة الهوية، واختراق الأنظمة المؤسسية، مما يجعلها من القضايا الأمنية ذات الأولوية العالية في مجال أمن المعلومات.

تعتمد هجمات التصيد الاحتيالي بشكل متزايد على استغلال عناوين URL المصممة بعناية لتبدو مشابهة للروابط الشرعية، مما يصعب على المستخدم العادي التمييز بينها وبين الروابط الآمنة. ومع التطور المستمر في أساليب المهاجمين، أصبحت الحلول التقليدية مثل القوائم السوداء (Blacklists) غير كافية لمواجهة هذا النوع من التهديدات، نظراً لعدم قدرتها على اكتشاف الروابط الجديدة أو المتغيرة بسرعة.

في هذا السياق، برزت تقنيات التعلم الآلي كحل فعال لمشكلة كشف التصيد الاحتيالي، لما تمتلكه من قدرة على تعلم الأنماط الخفية من البيانات، والتكيف مع التهديدات المستجدة دون الحاجة إلى تحديثات يدوية مستمرة [3].

1.2 مشكلة البحث

على الرغم من التقدم الكبير في مجال كشف التصيد الاحتيالي باستخدام التعلم الآلي، إلا أن هذه المشكلة لا تزال تواجه عدة تحديات جوهيرية. من أبرز هذه التحديات الاعتماد المفرط في بعض الدراسات على نوع واحد من تمثيل البيانات، سواء كان تمثيلاً نصياً خاماً لعناوين URL أو خصائص مستخرجة يدوياً، مما قد يحدّ من قدرة النموذج على التقاط جميع أنماط التصيد المحتملة [4].

إضافةً إلى ذلك، تعاني العديد من النماذج المقترحة من ضعف قابلية التعميم عند تطبيقها على بيانات جديدة أو بيانات واقعية، نتيجة عدم معالجة مشكلات مثل عدم توازن الفئات أو تسرب البيانات أثناء التدريب

والتقدير [5]. كما أن بعض الحلول المقترحة ترتكز على تحسين الدقة العامة للنموذج دون الاهتمام الكافي بمقاييس أكثر أهمية في هذا المجال، مثل الاسترجاع، والذي يعدّ حاسماً في تقليل مخاطر تصنيف روابط التصيد على أنها روابط شرعية [3].

بناءً على ما سبق، تتمحور مشكلة هذا البحث حول كيفية بناء نظام كشف تصيد احتيالي يعتمد على عناوين URL، يتمتع بدقة عالية، وقابلية تعليم جيدة، وقدرة فعالة على اكتشاف الروابط الضارة باستخدام منهجية مدرosaة في تمثيل البيانات و اختيار النماذج.

1.3 أهداف البحث

يهدف هذا البحث إلى تحقيق مجموعة من الأهداف الرئيسية، من أبرزها:

- بناء نظام ذكي لكشف روابط التصيد الاحتيالي اعتماداً على تقنيات التعلم الآلي .
- دراسة تأثير أساليب تمثيل البيانات المختلفة لعناوين URL على أداء نماذج التصنيف .
- مقارنة أداء مجموعة من خوارزميات التعلم الآلي باستخدام تمثيلات بيانات متعددة [2].
- اقتراح نموذج هجين يجمع بين التمثيل الخام لعناوين URL والخصائص المستخرجة [4].
- معالجة التحديات الشائعة في هذا المجال، مثل عدم توازن الفئات وتسرب البيانات .
- تقييم أداء النموذج المقترن باستخدام مقاييس تقييم متعددة تعكس الأداء الحقيقي للنظام [3].

1.4 أهمية البحث

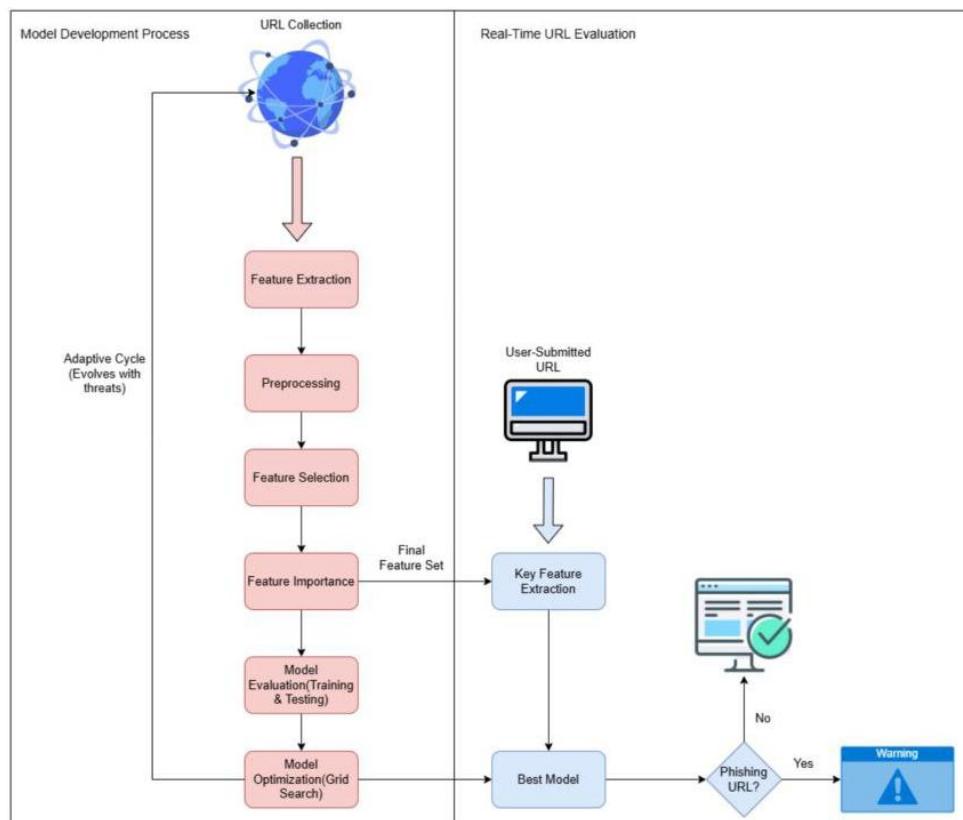
تبغ أهمية هذا البحث من كونه يقدم دراسة تطبيقية متكاملة لمعالجة مشكلة كشف التصيد الاحتيالي باستخدام عناوين URL فقط، دون الحاجة إلى محتوى الصفحة أو معلومات خارجية إضافية، مما يجعله مناسباً للتطبيق في الأنظمة الواقعية ذات القيود الزمنية أو التقنية .

كما يساهم البحث في توضيح أثر الدمج بين أكثر من أسلوب لتمثيل البيانات، ويوفر مقارنة منهجية بين النماذج المختلفة، مما قد يساعد الباحثين والمطوروين على اختيار الحلول الأنسب عند تصميم أنظمة كشف تصيّد فعالة [4].

1.5 منهجية البحث

يعتمد هذا البحث على منهجية تجريبية، تبدأ بجمع بيانات عناوين URL من مصادر موثوقة، ثم تطبيق مجموعة من خطوات المعالجة المسبقة، يليها تمثيل البيانات باستخدام أساليب مختلفة، وتدريب عدد من نماذج التعلم الآلي على هذه التمثيلات [3][5].

بعد ذلك، يتم تقييم أداء النماذج باستخدام مقاييس تقييم متعددة، وتحليل النتائج لاختيار النموذج الأكثر كفاءة واستقراراً. أخيراً، يتم اقتراح نموذج نهائي يعتمد على تمثيل هجين، مع تبرير اختياره بناءً على نتائج التجارب [4].



لشكل 2 النظام المقترن للكشف عن التصيّد الاحتيالي

1.6 الدراسات المشابهة

- دراسة [Nayak et al \(2025\)](#) - الدراسة الأساسية:-

“تحسين الكشف عن التصيّد باستخدام التعلم الآلي مع اختيار الميزات ونماذج التعلم العميق” تهدف هذه الدراسة إلى تطوير نموذج ذكي قادر على تعزيز دقة أنظمة الكشف التقليدية من خلال الدمج بين خوارزميات التعلم الآلي (Machine Learning) ونماذج التعلم العميق (Deep Learning Models)، مع تطبيق آلية متقدمة لاختيار الميزات الأكثر تأثيراً في عملية التصنيف.

اعتمد الباحثون في هذه الدراسة على مجموعة بيانات حديثة (Mendely 2020) تضم 58,645 رابطاً إلكترونياً (URLs) جمعت من مصادر متعددة تحتوي على مواقع تصيّد وأخرى شرعية. وتم استخلاص 111 ميزة (Features) من هذه الروابط، تتضمن خصائص هيكلية مثل طول الرابط وعدد الرموز الخاصة، وخصائص أمنية مثل وجود بروتوكول HTTPS أو شهادات الأمان، بالإضافة إلى سمات تحليلية مرتبطة بخوادم DNS وسجلات SSL.

ولتحسين أداء النماذج وتتجنب زيادة التكلفة الحسابية، استخدم الباحثون خوارزمية اختيار الميزات وترتيب الميزات حسب أهميتها وتأثيرها في عملية التنبؤ.

بعد تطبيق هذه الآلية، تم تقليل عدد الميزات من 111 إلى 20 ثم إلى 14 فقط، ومع ذلك احتفظ النموذج بمستوى دقة عالٍ جداً بلغ 94.46% باستخدام الشبكة العصبية الأمامية (Feed Forward Neural Network – FNN).

كما اختبر الباحثون عدة نماذج أخرى مثل: الشبكة العصبية العميقه (Deep Neural Network – DNN)، النموذج المدمج (Wide & Deep Model)، نموذج TabNet، وهو نموذج يعتمد على بنية عصبية تسمح بتفسير القرار (Explainable AI).

وأظهرت نتائج المقارنة أن جميع النماذج المحسنة قدّمت أداءً مرتفعاً، حيث حقق النموذج الأفضل دقة بلغت 95.53% عند استخدام 20 ميزة فقط، مع الحفاظ على سرعة استجابة مناسبة للتطبيق في الزمن الحقيقي (Real-Time Detection).

ولضمان موثوقية النموذج وعدم انحيازه لمجموعة البيانات الأصلية، أجرى الباحثون اختبار تعميم

(Generalization Test) على مجموعة بيانات جديدة من منصة Mendeley 2021، وقد

حقق النظام أداءً جيداً بنسبة دقة بلغت $\approx 80\%$ ، مما يُظهر قدرته على التكيف مع بيانات مختلفة

عن تلك التي تم التدريب عليها.

كما قدم الباحثون في هذه الورقة مقياساً جديداً لتقييم الأداء أطلقوا عليه اسم "مؤشر مقاومة

التصيّد" (Anti-Phishing Score)، وهو مقياس شامل يأخذ بعين الاعتبار الإيجابيات الكاذبة

Execution والسلبيات الكاذبة False Negatives (False Positives)

Time)، الأمر الذي يجعله أكثر دقة من مؤشرات التقييم التقليدية المعتمدة فقط على الدقة

.(Accuracy)

من الناحية البحثية، شكلت هذه الدراسة نقطة انطلاق أساسية لهذا المشروع، لأنها تُبرز أهمية

استخدام اختيار الميزات والتحسين الشبكي للمعاملات (Hyperparameter Grid Search)

لرفع أداء النماذج دون زيادة التعقيد الحسابي، وهو ما يُعد المحور الرئيس الذي يقوم عليه البحث

الحالي.

• دراسة Zaimi et al. (2024) – Springer

قدم الباحثون في دراستهم الموسومة "A Deep Learning Mechanism to Detect

Phishing URLs using the Permutation Importance Method and SMOTE-

"نمواذجاً متقدماً يعتمد على تقنيات التعلم العميق لتحليل روابط التصيّد باستخدام

.Mendeley 2020 بيانات

اعتمدوا على 111 ميزة تركيبية وسلوكية من أصل 58,645 رابطاً، وقاموا بانتقاء أكثر الميزات

تأثيراً من خلال Permutation Importance SMOTE-Tomek قبل تطبيق تقنيات الموازنة

.لمعالجة اختلال البيانات.

أظهرت النتائج تفوق أداء النموذج العميق المقترن، محققاً دقة بلغت 99.18% و F1-Score = 0.991، مما يعكس أهمية دمج اختيار الميزات الإحصائي مع خوارزميات الشبكات العصبية في رفع كفاءة الاكتشاف.

• دراسة Farea et al. (2025) – Elsevier

في دراستهم "FSFS: A Novel Statistical Approach for Fair and Trustworthy" في دراستهم "Phishing Detection Mendeley 2020" بنسختها الصغيرة (58,645 رابطاً)، مطبقين منهجاً إحصائياً متقدماً لتحسين التوازن في البيانات وتجنب تحيز النماذج.

ركزوا على العلاقة بين المتغيرات البنائية (structural features) والمؤشرات الأمنية للروابط باستخدام تحليل الانحدار المتعدد وتقنيات Feature Scaling المعيارية. حقق النموذج دقة تصنيف 98.7%，مشيراً إلى أن استخدام السمات الإحصائية المعيارية يمكن أن يوفر أداءً منافساً لطرائق التعلم العميق مع تفسير أوضح لنتائج التصنيف.

• دراسة Alasmari (2025) – Springer

قام الباحث Alasmari في دراسته بعنوان "Phishing Detection in IoT: An Integrated CNN-LSTM Framework Convolutional LSTM Networks" بتحليل بيانات Mendeley 2020 مستفيداً من Neural Networks و ميزاتها الترتكيبية الفنية.

هدفت الدراسة إلى بناء نظام قادر على كشف التصيد في بيئات إنترنت الأشياء (IoT) عبر تحليل الروابط المضمنة في واجهات الأجهزة الذكية.

أظهرت النتائج أن النموذج المدمج CNN-LSTM تفوق على النماذج الفردية، محققاً دقة كافية بلغت 99.3% مع Recall = 99.1%， مما يدل على فعالية الدمج بين تحليل الأنماط المكانية والزمانية في الروابط الإلكترونية.

قدم الباحثون في دراستهم "Machine Learning-Based Phishing Detection" نموذجاً إحصائياً يعتمد على خوارزميات Naïve Bayes و Random Forest و System Mendeley 2020 لتصنيف الروابط ضمن بيانات Decision Tree. اعتمدوا على مجموعة الميزات الـ 111 كاملة دون إسقاط أو تقليل، بهدف تقييم أداء الخوارزميات الكلاسيكية عند تطبيقها على البيانات الموحدة. أظهرت النتائج أن نموذج Random Forest حق أداءً تفوق به على البقية بدقة 98.4%. وأشارت الدراسة جدوى استخدام خوارزميات التعلم الآلي التقليدية كأساس يمكن تطويره لاحقاً بدمجها مع تقنيات التحسين العميق.

- الدراسات المعتمدة على الروابط (URL-Based Phishing Detection)

تعتمد هذه الفئة من الدراسات على تحليل خصائص روابط المواقع (URLs) لاكتشاف محاولات التصيد. يتم استخراج السمات المميزة من الرابط مثل طول العنوان، وعدد النقاط، واستخدام الرموز الخاصة، وطول اسم النطاق، ووجود البروتوكول الآمن (HTTPS) وغيرها من الميزات الإحصائية واللغوية.

تطبق خوارزميات التعلم الآلي (Machine Learning) على هذه السمات بعد معالجتها، لتصنيف الرابط إلى "تصيدي" أو "شرعى".

تُستخدم خوارزميات مثل Support Vector و Random Forest و Decision Tree و Machine (SVM) في هذا النوع من الدراسات، بينما تعتمد الدراسات الحديثة على خوارزميات أكثر تطوراً مثل Convolutional Neural Networks (CNN) و Deep Neural Networks لاستخراج أنماط أكثر تعقيداً من روابط المواقع.

• دراسة (2025) .Rao et al

قدم Rao وزملاؤه في بحثهم بعنوان “Phish-Jam: A Hybrid Super Learner” نموذجاً متطوراً يدمج بين Ensemble for Phishing Detection on Mobile Devices

خوارزميات تعلم الآلة والتعلم العميق ضمن إطار Super Learner Ensemble. تميزت هذه الدراسة بتركيزها على بيئة الهاتف المحمولة، حيث صمم النظام ليعتمد فقط على الروابط (URLs) دون تحليل محتوى الصفحات، وذلك لتقليل الحمل الحسابي وזמן التنفيذ.

تم استخراج مجموعة من الميزات التركيبية (Structural Features) مثل طول الرابط وعدد النقاط والرموز الخاصة، إلى جانب تمثيلات نصية (Transformer Embeddings) تُسهم في التقاط الأنماط الخفية في أسماء النطاقات.

أظهر النظام أداءً مرتفعاً جداً، إذ بلغت الدقة 98.93% مع F1-Score = 99.07%， مما يؤكد كفاءته في العمل في الزمن الحقيقي على أجهزة محدودة الموارد.

• دراسة (2025) .Alsaidi et al

قدم الباحثون في ورقتهم “HawkPhish-DNN” إطاراً يعتمد على خوارزمية تحسين الصقر الهراري (Harris Hawk Optimization – HHO) لرفع أداء الشبكات العصبية العميقة (Deep Neural Network – DNN).

استُخدمت مجموعة بيانات معيارية عامة تتضمن روابط شرعية وأخرى خبيثة، وتم تحديد ميزات هيكيلية مثل طول النطاق ومستوى التشفيير.

تميزت هذه الدراسة بمحاولة الموازنة بين الدقة وتقليل معدل الإيجابيات الكاذبة (False Positives).

أظهرت النتائج أن النموذج المقترن حق دقة 99.6% مع FPR = 0.2%， وهو من أعلى المعدلات المسجلة في أبحاث 2025.

• دراسة .Barik et al : (2025)

في دراستهم “EGSO-CNN”， طور Barik وزملاؤه نموذجاً قائماً على الشبكات العصبية الالتفافية (Convolutional Neural Networks – CNN) مدمجة مع خوارزمية التحسين الجماعي المعززة (Enhanced Group Search Optimization – EGSO). سعت الورقة إلى إنشاء مجموعة بيانات جديدة أكثر توازناً وشمولاً، بعد ملاحظة قسم المجموعات التقليدية.

اعتمد النظام على مزيج من الميزات العددية (Numerical Features) وعمليات التطبيع .VAE واستخلاص الأنماط العميق عبر شبكات StandardScaler) حق النموذج أداءً متميزاً بدقة بلغت 99.44% و 99.32% F1-Score = ، مما يبرهن فعالية الدمج بين التحسين التطورى والتعلم العميق.

• دراسة .Maturure et al : (2024)

تناولت ورقة "Hybrid Machine Learning Model for Phishing Detection" إمكانية رفع الدقة من خلال النماذج الهجينية (Hybrid Models) التي تجمع بين خوارزميات مختلفة مثل .Support Vector Machine (SVM)، XGBoost، Random Forest اعتمد الباحثون على مجموعة بيانات أكاديمية مفتوحة، وتم التركيز على دور اختيار الميزات Ensemble في تحسين نتائج النماذج التجميعية (Manual Feature Selection) اليدوي (Models).

ورغم أن الدراسة لم تُصرّح بنسبة دقة محددة، إلا أنها أثبتت أن النماذج الهجينية تتفوق عادة على النماذج الفردية في الثبات والقدرة على التعميم.

• دراسة (2025) Alheyasat

في دراسة "Web Phishing Detection and Awareness Utilizing Hybrid Machine Learning Algorithms" قدم Alheyasat نموذجاً ثنائياً للطبقات يجمع بين التعلم المراقب وغير المراقب، إذ استخدم خوارزمية التجميع K-Means لتصنيف الروابط أولياً، ثم استخدم Random Forest وXGBoost لإجراء التصنيف النهائي. تميزت الدراسة بدمج عنصر التوعية الأمنية (User Awareness) عبر تطوير إضافة متصفح (Browser Extension) تُحدِّر المستخدم فور فتح الرابط المشبوه. بلغت الدقة 99.7% مع تقليل عدد الميزات من 87 إلى 34 دون فقدان الأداء، ما يُظهر كفاءة النموذج في التطبيق العملي.

• دراسة (2025) Kavya & Sumathi

قدمت الورقة "Staying Ahead of Phishers: A Review of Recent Advances in Phishing Detection" مراجعة شاملة لمختلف الاتجاهات التقنية في المجال، بدءاً من الطرق التقليدية المعتمدة على القوائم السوداء وصولاً إلى الأساليب الحديثة مثل نماذج التعلم العميق (Deep Learning Models) والشبكات التوليدية التنافسية (Generative Adversarial Networks – GANs). خلصت الدراسة إلى أنَّ الجمع بين التعلم التجميعي (Ensemble Learning) والتحسين المستمر للنماذج ضد الهجمات الخصومية (Adversarial Robustness) يمثل التوجّه المستقبلي الأهم في أبحاث التصيد.

• دراسة (2024) .Taha et al :

تناولت ورقة "Comparative Machine Learning Algorithms for Detecting Phishing Websites Decision Tree (DT) و Random Forest (RF) و XGBoost" مقارنة أداء مجموعة من الخوارزميات الإحصائية مثل

اعتمدت الدراسة على مجموعة بيانات تتضمن روابط تصيد حقيقية تم جمعها من الإنترنت، وخلصت النتائج إلى تفوق Decision Tree بنسبة 96.89% تلاه Random Forest بنسبة 94.57%

وقد أكدت الورقة أن خوارزميات التجميع (RF, XGBoost) تحقق دقة واستقراراً أعلى من النماذج الفردية التقليدية.

- الدراسات المعتمدة على البريد الإلكتروني (Email-Based Phishing Detection)

تركز هذه الفئة من الدراسات على تحليل الرسائل الإلكترونية بهدف التمييز بين الرسائل الخبيثة والمشروعة، من خلال فحص العنوان (Header) والنص الداخلي (Body) والروابط المضمنة. وتعتمد هذه الأبحاث غالباً على تقنيات التعلم الآلي (Machine Learning) ومعالجة اللغة الطبيعية (Natural Language Processing – NLP) لتحليل أنماط النصوص وتحديد العبارات الدالة على محاولات التصيد.

• دراسة (2025) .Zhang et al

A Combined Feature Selection Approach for Malicious “
في ورقتهم البحثية ”Email Detection Based on a Comprehensive Email Dataset (EPVME)

قام الباحثون ببناء مجموعة بيانات ضخمة تُعرف باسم EPVME Dataset، تتضمن 660,985 رسالة بريد إلكتروني، منها 49,136 رسالة خبيثة (Phishing Emails). استخدمت الدراسة نهجاً مركباً لاختيار الميزات يجمع بين التحليل الإحصائي وخوارزميات الانتقاء التقائي (Automatic Feature Selection) لتحديد الخصائص الأكثر تأثيراً من بين 79 ميزة (Features) تعطي عناصر رأس الرسالة مثل عنوان المرسل ومجال النطاق، إضافة إلى خصائص نصية من جسم الرسالة كالكلمات المفتاحية وروابط الإحالة.

تم اختبار عدة نماذج تصنيف باستخدام تقنيات Machine Learning التقليدية مثل Random Forest و SVM و XGBoost.

حقق النموذج المقترن دقة 99.968% مع FPR = 0.099%， وهي من أعلى النتائج المسجلة في هذا المجال، ما يؤكد فاعلية الدمج بين تحليل البنية النصية والهندسة الخصائصية في تعزيز أداء الكشف.

• دراسة (2024) .Fares et al

"Machine Learning Approach for Email Phishing Detection" تناولت الورقة التي أعدّها Fares وزملاؤه تطبيق خوارزميات التعلم الآلي على بيانات بريد إلكتروني تم جمعها من بيئات أكاديمية ومؤسساتية.

اعتمد الباحثون على مجموعة من الميزات النصية مثل طول الرسالة، وعدد الروابط، وطبيعة الكلمات المفتاحية المرتبطة بمحاولات التصيد (مثلاً "verify", "account", "click").

تم استخدام خوارزميات Support Vector Machine (SVM) و XGBoost لاستخدام المقارنة بين أدائها.

أظهرت النتائج أن خوارزمية **SVM** حققت دقة $\approx 97.6\%$ متفوقةً على **XGBoost** (Feature Selection) ($\approx 96.6\%$)، كما بيّنت الدراسة أهمية استخدام تقنيات اختيار الميزات (Feature Selection) لتقليل التكرار وتحسين سرعة التدريب دون التأثير على الأداء.

من الناحية التحليلية، توضح هذه الدراسات أن الكشف عبر البريد الإلكتروني يعتمد بدرجة كبيرة على تحليل اللغة والسياق النصي، بينما الكشف عبر الروابط (URL-Based) يعتمد أكثر على الخصائص التركيبية والتقنية للرابط نفسه.

ولذلك فإن الدراسات القائمة على البريد الإلكتروني رغم دقتها العالية، إلا أنها أقل ملاءمة للأنظمة الزمنية الفورية (Real-Time Systems) نظراً للحاجة إلى معالجة نصوص معقدة وبيانات أكبر حجماً.

الجدول (1) - مقارنة بين الدراسات السابقة

رقم	الدراسة (السنة)	نوع البيانات (Data Type)	مصدر البيانات	عدد/نوع الميزات	النماذج أو المنهجيات المستخدمه	أفضل دقة مُصرّح بها
1	Nayak et al., 2025	URL-Based	Mendeley 2020	111 → 20/14 مختارة	FNN, DNN, Wide & Deep, TabNet + + اختيار ميزات Grid Search	95.53%
2	Zaimi et al., 2024	URL-Based	Mendeley 2020	111 → مختارة Permutati on	Deep NN + SMOTE- Tomek	99.18%
3	Farea et al., 2025	URL-Based	Mendeley 2020	111	Statistical FSFS Model	98.7%
4	Alasmari, 2025	URL-Based (IoT)	Mendeley 2020	111	CNN + LSTM Hybrid	99.3%
5	AI-Qasmi et al., 2024	URL-Based	Mendeley 2020	111	RF / DT / NB (RF)	98.4%
6	Nayak et al., 2025	URL-Based	Mendeley 2020 + Grid Search	111 → 20 / 14	FNN, Wide & Deep, TabNet	95.53%
7	Rao et al., 2025	URL-Based (Mobile)	روابط مميزة يدوياً + تمثيلات	ميزات تركيبية + تصيّة مضمنة	Super Learner Ensemble + Deep Learning	98.93%

8	Alsaidi et al., 2025	URL-Based	مجموعات معيارية عامة	مميزات هيكلية: طول، اعتلاج	DNN + Harris Hawk Optimization (HHO)	99.6%
9	Barik et al., 2025 – EGSO– CNN	URL-Based	مجموعة جديدة منشأة من روابط تصيد وشرعية	VAE + StandardS caler	CNN + EGSO Optimization	99.44%
10	Mature et al., 2024 – Hybrid ML	URL-Based	مجموعات أكاديمية مفتوحة	مميزات مختارة يدوياً بعد التحليل	RF, XGBoost, SVM (هجين/جمعي)	لم يُصرّح بنسبة الدقة النهائية
11	Alheyasat, 2025 – LWPD	URL-Based	مجموعة متوازنة من الروابط	87 → 34 بعد اختيار الميزات	K-means + RF + XGBoost + إضافة متصفح	99.7%
12	Kavya & Sumathi, 2025 – Review	Mixed (URL/Email)	مراجعة شاملة لأبحاث كشف التصيد	—	تحليل منهجي (ML, DL, GANs, Graph-based)	— ورقة مراجعة
13	Taha et al., 2024 – Comparative ML	URL-Based	بيانات موقع ويب تصيد	مميزات تقليدية من الروابط	RF, DT, XGBoost	96.89%
14	Zhang et al., 2025 – EPVME	Email– Based	EPVME Dataset + قواعد بيانات مفتوحة	79 رأس/نص	+ اختيار ميزات مركبة ML	99.968%
15	Fares et al., 2024 – Email ML	Email– Based	بيانات بريد أكاديمية	مميزات نصية وهيكيلية	SVM, XGBoost, RF	0.976 0.966

1.7 التحديات:

تواجه أنظمة كشف التصيّد الحديثة مجموعة من التحديات التقنية والمنهجية، وذلك بسبب التطور المستمر في أساليب المهاجمين وقدرتهم على تقليد السلوكيات الشرعية لموقع الويب والتطبيقات الإلكترونية.

تمثل أبرز هذه التحديات في قدرة صفحات التصيّد على توليد روابط URL ديناميكية تتغير باستمرار لتجاوز أنظمة الترشيح التقليدية، إضافة إلى استخدام تقنيات إخفاء العناوين (Obfuscation) وإعادة التوجيه المتعددة (Redirection Chains) مما يجعل عملية الكشف أكثر تعقيداً.

من جانب آخر، أدى الاعتماد الواسع على التقنيات الذكية في التسويق الرقمي والتفاعل الاجتماعي إلى زيادة صعوبة التمييز بين الروابط الحقيقة والمزيفة، خصوصاً مع استخدام الذكاء الاصطناعي في توليد محتوى واقعي ومقنع. كما تشكل محدودية مجموعات البيانات المحدثة وضعف توازنها بين الفئات الحقيقية والمُخادعة تحدياً جوهرياً أمام تدريب النماذج الذكية بدقة عالية.

تطلب مواجهة هذه التحديات تكاملاً بين تحليل بنية الرابط (Structural Analysis) والسلوك الزمني للمستخدم (Behavioral Profiling)، إلى جانب تطوير خوارزميات هجينية قادرة على التعلم المستمر من الأنماط الجديدة في الهجمات.

ومع تطور بيئة الويب واستخدامها على الهواتف الذكية والأجهزة المحمولة، أصبحت الحاجة إلى أنظمة كشف تتميز بالسرعة وخففة الأداء من أولويات البحث العلمي في هذا المجال.

1.7.1 الاحتيال والتصيد الاحتيالي:

يُعد التصيد الاحتيالي من أكثر التحديات خطورة في مجال أمن المعلومات، إذ يستغل المهاجمون الثقة المتبادلة بين المستخدم والمنصة لإقناع الضحية بإدخال بياناته الحساسة.

تستخدم المواقع المزيفة في ذلك تقنيات التصميم المتطابق ومحركات توليد الصفحات الآلية لجعل صفحة الاحتيال شبه مطابقة للموقع الأصلي، مما يزيد من احتمالية نجاح الهجوم.

يتضح هنا أن النماذج الذكية تحتاج إلى تحليلٍ أعمق للسلوك العام للرابط بدلاً من الاكتفاء بالتحليل السطحي لاسم النطاق أو امتداده.

1.7.2 إخفاء الهوية وسرقة البيانات:

يستخدم المهاجمون تقنيات إخفاء الهوية لتجاوز الأنظمة الأمنية، حيث تنشأ روابط تعتمد على بروتوكولات تشفير مثل HTTPS مزيفة أو شهادات رقمية غير موثوقة.

كما يتم غالباً تضمين السكريبتات الخبيثة ضمن الصفحات الشرعية لاستخراج كلمات المرور والمعلومات الشخصية بشكلٍ خفي، وهو ما يجعل كشفها عبر التحليل التقليدي أكثر صعوبة.

1.7.3 تجاوز الفلاتر الذكية:

تُظهر الأبحاث أن العديد من أنظمة الترشيح التقليدية، وحتى بعض الأنظمة الذكية، يمكن التحايل عليها عبر التوليد المستمر لعناوين URL فريدة لكل مستخدم.

كما يستخدم المهاجمون تقنيات التمويه البصري (Visual Cloaking) لتغيير مظهر الصفحة دون التأثير على كود المصدر، مما يجعل أدوات الكشف القائمة على السمات البصرية أقل فعالية.

1.8 التطبيقات:

تتنوع تطبيقات أنظمة كشف التصيّد في الوقت الراهن بين الأدوات المستقلة والأنظمة المدمجة ضمن متصفحات الويب والمنصات الاجتماعية.

يهدف هذا التنوع إلى حماية المستخدمين من الوصول إلى صفحات مزيفة أو إدخال بياناتهم في موقع احتيالية.

وقد طورت حديثاً تطبيقات متقدمة تعتمد على تقنيات التعلم العميق (Deep Learning) ونماذج اللغة الكبيرة (LLMs) لتحليل محتوى الرابط، شكله، وسلوك المستخدم عند التفاعل معه.

من بين أبرز التطبيقات العملية نجد إضافات المتصفحات (Browser Extensions) التي تقوم بفحص الرابط بشكل فوري قبل تحميل الصفحة، وتطبيقات الهاتف الذكي التي تستخدم خوارزميات هجينة تجمع بين تحليل الرابط والسلوك البشري لتوفير حماية آنية.

ورغم التطور التقني الكبير، ما زالت بعض التطبيقات تعاني من قصور في الدقة الزمنية (Real-Time Detection) أو تواجه مشكلة في التعامل مع الروابط المختصرة (Shortened URLs) التي تُستخدم بكثرة في شبكات التواصل الاجتماعي.

كما يلاحظ أن أنظمة الكشف المدمجة في متاجر التطبيقات الرسمية مثل Google Play Protect أو Apple Security Gateway لا تزال بحاجة إلى تطوير إضافي لتعطية كافة أنماط التصيّد المستحدثة.

تسعى الأبحاث الحديثة لتطوير تطبيقات تعتمد على التعلم المستمر (Continuous Learning) لتحديث النماذج تلقائياً مع ظهور روابط جديدة، بحيث تزداد فعالية النظام بمرور الوقت دون الحاجة لإعادة التدريب الكامل.

ومن المتوقع أن تتجه الدراسات المستقبلية إلى دمج أنظمة الكشف في البنية التحتية للمتصفحات وواجهات البريد الإلكتروني، بما يضمن حماية المستخدمين على مستوى النظام لا التطبيق فقط.

الفصل الثاني: الإطار النظري

2.1 مفهوم التصيد الاحتيالي (Phishing Concept)

يُعد التصيد الاحتيالي أحد أخطر أشكال الجرائم السيبرانية المعاصرة، ويعتمد بشكل أساسي على استغلال العامل البشري من خلال الخداع والهندسة الاجتماعية بهدف الحصول على معلومات حساسة، مثل بيانات تسجيل الدخول أو المعلومات المالية [1].

تطور هذا النوع من الهجمات بشكل ملحوظ مع تطور تقنيات الويب، حيث لم يعد يقتصر على رسائل بريد إلكتروني بدائية، بل أصبح يعتمد على روابط مصممة بعناية تحاكي الروابط الشرعية بدقة عالية [2].

في السنوات الأخيرة، لوحظ أن عناوين URL أصبحت عنصراً محورياً في معظم هجمات التصيد، إذ تُستخدم كوسيل مباشر لتوجيه الضحية إلى صفحات مزيفة تحاكي الواقع الأصلي. هذا التحول جعل تحليل الرابط نفسه نقطة انطلاق مهمة للكشف المبكر عن الهجمات الاحتيالية قبل وقوع الضرر الفعلي [3].

2.2 الكشف عن التصيد الاحتيالي بالاعتماد على روابط URL (URL-Based Phishing Detection)

يركز الكشف المعتمد على روابط URL على تحليل البنية النصية والتركيبية للرابط دون الحاجة إلى تحميل محتوى الصفحة أو تنفيذ أي شيفرة برمجية مرتبطة بها [4].

يعتمد هذا النهج على ملاحظة أن الروابط الاحتيالية غالباً ما تتضمن أنماطاً غير طبيعية، مثل الطول المفرط للرابط، أو الاستخدام الكثيف للأرقام والرموز الخاصة، أو التلاعب في أسماء النطاقات الفرعية، وهي سمات يمكن استغلالها لبناء أنظمة كشف فعالة.

يتميز هذا الأسلوب بكونه مناسباً للتطبيق في الأنظمة ذات الزمن الحقيقي، نظراً لانخفاض التكلفة الحسابية لتحليل الرابط مقارنة بتحليل المحتوى الكامل للصفحة [7].

2.3 مبررات التركيز على تحليل روابط URL

يُعد الاعتماد على تحليل روابط URL خياراً استراتيجياً في هذا البحث لعدة أسباب تقنية ومنهجية.

ينتج هذا النهج الكشف المبكر عن الهجمات الاختيالية قبل تحميل الصفحة، مما يحد من فرص تنفيذ البرمجيات الخبيثة.

كما يتميز تحليل الروابط بالسرعة والكفاءة، إذ لا يتطلب موارد حاسوبية عالية، مما يجعله مناسباً للتطبيق العملي في البيئات الواقعية.

إضافةً إلى ذلك، فإن انتشار بروتوكول HTTPS جعل تحليل محتوى الصفحات أكثر تعقيداً، في حين يبقى الرابط نفسه متاحاً للتحليل في جميع الحالات تقريباً [10].

بناءً على ذلك، تم اعتماد الكشف المعتمد على URL كأساس منهجي في هذا المشروع.

2.4 الطرق التقليدية لكشف التصيد وعيوبها

2.4.1 القوائم السوداء (Blacklists)

تعتمد القوائم السوداء على قواعد بيانات تحتوي على روابط تم الإبلاغ عنها مسبقاً على أنها روابط تصيد.

ورغم بساطتها وسرعة استخدامها، إلا أنها تعاني من قصور واضح في التعامل مع الهجمات الجديدة (Zero-Day Attacks)، حيث لا يتم اكتشاف الرابط إلا بعد وقوع الضرر [12].

2.4.2 الأنظمة المعتمدة على القواعد (Heuristic-Based Systems)

تعتمد هذه الأنظمة على مجموعة من القواعد اليدوية المستخلصة من خبرات بشرية، مثل طول الرابط أو وجود كلمات معينة.

غير أن هذه القواعد غالباً ما تكون جامدة وغير قادرة على التكيف مع تطور أساليب المهاجمين، مما يؤدي إلى ارتفاع معدلات الخطأ [14].

2.5 دوافع الانتقال إلى تعلم الآلة (Machine Learning Adoption)

بسبب القيود الجوهرية للطرق التقليدية، أصبح الانتقال إلى تقنيات تعلم الآلة ضرورة حتمية في مجال كشف التصيّد الاحتيالي.

تتميز نماذج تعلم الآلة بقدرتها على التعميم (Generalization)، حيث تتعلم الأنماط العامة للروابط الاحتيالية بدلاً من حفظ روابط محددة.

كما تمتلك هذه النماذج القدرة على معالجة كميات كبيرة من البيانات واكتشاف علاقات معقدة يصعب على القواعد اليدوية تمثيلها [17].

إضافةً إلى ذلك، يمكن إعادة تدريب النماذج بشكل دوري لمواكبة تطور تقنيات الهجوم، مما يمنح النظام مرونة واستقرارية أعلى.

2.6 الشفافية وقابلية التفسير (Explainability – SHAP)

تُعد قابلية تفسير قرارات نماذج تعلم الآلة عنصراً أساسياً في الأنظمة الأمنية، حيث لا يكفي أن يكون النموذج دقيقاً دون القدرة على تفسير سبب اتخاذه لقرار معين [21].

في هذا السياق، تُستخدم تقنيات مثل SHAP لتوضيح مساهمة كل خاصية في القرار النهائي للنموذج، مما يعزز ثقة المستخدمين بالنظام ويساعد على تحليل سلوك النموذج [22].

الفصل الثالث : تجهيز بيئة العمل

3.1 مقدمة الفصل

يُعد إعداد بيئة العمل البرمجية مرحلة محورية في تنفيذ أنظمة تعلم الآلة، إذ تشكل الأساس الذي تُبنى عليه جميع التجارب اللاحقة من تدريب وتقدير ونشر النماذج. في سياق أنظمة كشف التصيّد الاحتيالي، تزداد أهمية هذه المرحلة نظراً لحساسية البيانات المستخدمة، وتعقيد خطوط المعالجة، وتعدد التمثيلات المعتمدة على كل من النص الخام والخصائص الجدولية.

يهدف هذا الفصل إلى تقديم وصف تفصيلي للبيئة البرمجية المعتمدة في تنفيذ النظام المقترن، مع توضيح أسباب اختيار أدوات التطوير، وأالية إعداد بيئة التنفيذ، وتصميم خط المعالجة البرمجية (Pipeline)، بالإضافة إلى شرح كيفية حفظ النماذج وتشغيلها ضمن بيئة تطبيقية. يضمن هذا التوصيف العملي قابلية إعادة الإنتاج، ويمكن أي باحث أو مطور من فهم كيفية تنفيذ النظام خطوة بخطوة.

3.2 المقارنة بين بيئات العمل المتاحة:

تم إجراء دراسة تقنية للمفاضلة بين عدة بيئات تطوير (IDEs) ومنصات برمجية، لضمان اختيار الأداة التي توفر أقصى درجات التحكم في النماذج:

الجدول (2) - مقارنة بينات العمل المتاحة

معايير المقارنة	VS Code	PyCharm	MATLAB	Google Colab
خفة الأداء	خفيف جداً، لا يستهلك موارد الجهاز بكثافة أثناء المعالجة.	ثقيل جداً ويحتاج مواصفات قوية للعمل بكفاءة دون تعليق.	مرتفع الاستهلاك جداً ويحتاج مواصفات قوية للعمل بكفاءة	يعتمد على المتصفح والإنترنت، وأي انقطاع يؤدي لفقدان التقدم في التدريب
تعدد المهام	يسمح بتشغيل الأكواد وبناء واجهة Streamlit في وقت واحد.	معقد في الانتقال بين التدريب وبناء واجهة المستخدم.	يركز على المحاكاة الرياضية وليس على نشر التطبيقات الأمنية.	مخصص للخلايا البرمجية فقط، لا يمكن استخدامه لبناء تطبيق نهائي.

مكتباته محدودة بإصدارات معينة تفرضها google، وصعب التخصيص.	محدود جداً في مكتبات الذكاء الاصطناعي Open (Source). مفتوحة المصدر (.Source)	إعداد البيانات الافتراضية فيه قد يتسبب بتناسب الإصدارات.	توافقية مطلقة مع و PyTorch و TensorFlow في بيئه SHAP	ادارة المكتبات
محدود في عرض بعض الرسوم التفاعلية المتقدمة لمكتبة SHAP.	لا يدعم مكتبة SHAP بشكل مباشر، مما يصعب عملية التفسيرية.	يواجه أحياناً مشاكل في عرض المخططات الرسومية الثقيلة.	يدعم عرض مخططات SHAP التفاعلية بدقة عالية وسرعة.	تفسير النتائج (SHAP)

3.3 تبرير اختيار Visual Studio Code (VS Code)

تم اختيار **Visual Studio Code** كبيئة تطوير رئيسية لتنفيذ المشروع نظراً لكونه بيئة خفيفة، مرنّة، وقابلة للتخصيص، وتدعى بشكل مباشر مشاريع تعلم الآلة المعقدة متعددة المراحل. يتطلب النظام المقترن تنفيذ عمليات متنوعة تشمل تجهيز البيانات، بناء التمثيلات، تدريب نماذج متعددة، تقييم الأداء، وربط النموذج بواجهة استخدام تفاعلية، وهي مهام يصعب إدارتها بكفاءة عند استخدام أدوات محدودة الإمكانيات.

يوفر **Visual Studio Code** تكاملاً متقدماً مع لغة **Python**، حيث يتيح اختيار مفسر اللغة وربطه مباشرة بالبيئة الافتراضية الخاصة بالمشروع، مما يقلل من أخطاء التوافق بين الإصدارات ويمنع تشغيل الشيفرة باستخدام إعدادات غير صحيحة. كما يوفر طرفية مدمجة تسمح بتنفيذ أوامر التنصيب والتشغيل دون الحاجة لاستخدام أدوات خارجية، وهو ما يسهل تتبع مراحل التنفيذ وتسويقه [1][2].

ومن أبرز ميزات هذه البيئة :

1. المرونة الفائقة (**Versatility**) : القدرة على دمج ملفات (**Jupyter.ipynb**) لاختبار النماذج، وملفات (**.py**) التقليدية لبناء النظام النهائي في واجهة واحدة.

2. نظام الملحقات (Extensions Ecosystem) لـ Python و Microsoft: الاعتماد على ملحقات Extensions Ecosystem، مما قلل الأخطاء البشرية عند التعامل مع Pylance، مما وفر ميزة الذكاء الاصطناعي في كتابة code، مما يقلل الأخطاء البشرية.

3. التحكم بالموارد: يسمح VS Code بمراقبة استهلاك الجهاز أثناء تدريب نموذج TabNet، مما مكن من تحسين code ليعمل بأقصى سرعة ممكنة.

4. بيئة التطوير الموحدة: القدرة على إدارة ملفات البيانات، النماذج المدرية، وواجهة المستخدم في مكان واحد ومنظمه بشكل احترافي.

3.4 توصيف عملية التنصيب:

تمت عملية تهيئة النظام وفق تسلسل تقني مدروس لضمان انسيابية العمل بين العتاد والبرمجيات:

3.4.1 إعداد بيئة البرمجة:

- تثبيت المترجم:

تم تنصيب لغة Python (3.10) لضمان الاستقرار التام مع مكتبات التعلم العميق المستخدمة.

- البيئة الافتراضية (Virtual Environment):

تم إنشاء بيئة مستقلة باستخدام أداة venv لعزل كافة الاعتمادات البرمجية وضمان نقاء بيئة التجارب.

- تهيئة الملحقات:

تم تفعيل إضافات Python و Jupyter داخل المحرر لتمكين التنفيذ المتسلسل للأكواد.

3.4.2 تنصيب الأدوات البرمجية:

1. محرر الأكواد Visual Studio Code :

تم تنصيبه كأداة تطوير أساسية (IDE) لإدارة ملفات المشروع وتعديل الأكواد، مع تفعيل ميزة "التحكم في النسخ" لضمان تتبع التغييرات البرمجية.

2. مدير الحزم (Pip):

تم استخدام أداة Pip كأداة رسمية لتنصيب وإدارة كافة الحزم البرمجية، حيث ضمنت هذه الأداة جلب النسخ المستقرة والمتواقة مع نظام التشغيل.

3. ملحقات بایثون (Python Extensions):

تم تثبيت حزمة ملحقات Microsoft VS Code داخل Microsoft VS Code، والتي وفرت أدوات التصحيح اللحظي .(IntelliSense) والمساعدة الذكية في كتابة الكود (Debugging)

4. خادم الاستضافة المحلي (Local Hosting Tool):

تم الاعتماد على أدوات تشغيل الويب المدمجة لتشغيل واجهة Streamlit، مما سمح بإنشاء بيئة عرض محلية (Local Server) لمحاكاة واجهة التنبؤ النهائية.

3.5 إعداد بيئة التنفيذ البرمجية

لضمان استقرار النظام ومنع أي تأثير خارجي على نتائج التجارب، تم اعتماد بيئة تنفيذ مستقلة قائمة على إنشاء بيئة افتراضية (Virtual Environment). يسمح هذا الأسلوب بعزل مكتبات المشروع عن مكتبات النظام، ويمنع تعارض الإصدارات بين المكتبات المختلفة، وهو من الممارسات القياسية الموصى بها في المشاريع البحثية المعتمدة على Python [1][3].

3.5.1 إنشاء وتفعيل البيئة الافتراضية

تم إنشاء بيئة افتراضية خاصة بالمشروع باستخدام أدوات Python القياسية، ثم تفعيلها وربطها مباشرةً ببيئة التطوير. بعد التفعيل، أصبحت جميع عمليات التنفيذ وتنصيب المكتبات مرتبطة بهذه البيئة فقط، مما يضمن توحيد ظروف التشغيل عبر جميع مراحل المشروع.

```
python -m venv venv
```

```
venv\Scripts\activate
```

يساهم هذا الإجراء في تقليل الأخطاء الناتجة عن اختلاف إعدادات النظام بين الأجهزة المختلفة، ويعزّز من قابلية إعادة تنفيذ التجارب.

3.5.2 تثبيت الاعتمادات البرمجية وتوثيقها

بعد تفعيل البيئة الافتراضية، تم تثبيت جميع المكتبات البرمجية المطلوبة باستخدام مدير الحزم pip. كما تم توثيق هذه المكتبات وإصداراتها ضمن ملف مخصص للاعتمادات، مما يسمح بإعادة بناء نفس البيئة البرمجية على أي جهاز آخر بنفس الإعدادات، وهو عنصر أساسي لضمان قابلية إعادة الإنتاج [3].

تعليمات تنفيذ نموذجية:

```
pip install -r requirements.txt
```

تساعد هذه الخطوة على ضمان أن جميع التجارب اللاحقة تعتمد على نفس الإصدارات البرمجية، مما يمنع اختلاف النتائج بسبب تغيير المكتبات.

3.6 تصميم وتنفيذ خط المعالجة البرمجية (Processing Pipeline)

تم تصميم النظام وفق خط معالجة برمجي متكم (Pipeline) يربط جميع مراحل التنفيذ بشكل تسلسلي ومنضبط، بدءاً من إدخال الرابط الخام وانتهاءً بإنتاج قرار التصنيف النهائي. يهدف هذا التصميم إلى تقليل التدخل اليدوي، وضمان اتساق المعالجة، وتسهيل تتبع الأخطاء وتحليل الأداء.

يشمل خط المعالجة المراحل التالية:

1. تحميل البيانات وتنظيفها لضمان سلامتها
2. تقسيم البيانات إلى مجموعات تدريب، تحقق، واختبار
3. استخراج التمثيلات النصية والخصائص الجدولية
4. دمج التمثيلات ضمن متوجه ميزات موحد
5. تمرير الميزات إلى نموذج التصنيف وإنتاج القرار النهائي

يساهم استخدام بنية Pipeline في ضمان أن جميع عمليات التحويل والمعالجة المسبقة تُدرَب على بيانات التدريب فقط، ثم تُطبَّق بنفس الإعدادات على بيانات التحقق والاختبار، مما يمنع تسرب المعلومات (Data Leakage) ويحافظ على نزاهة التقييم التجاري [5].

3.7 تنفيذ التمثيلات المعتمدة للبيانات

3.7.1 التمثيل النصي للرابط (Raw URL Representation)

تم التعامل مع عنوان الرابط كنص خام دون تحليل محتوى الصفحة، ثم تحويله إلى تمثيل عددي باستخدام تقنية TF-IDF على مستوى المحارف (Character n-grams). يتيح هذا الأسلوب التقاط الأنماط التركيبية الدقيقة داخل الروابط، مثل التكرار غير الطبيعي للمقاطع النصية أو البني المستخدمة في محاولات الخداع، وهو مناسب بشكل خاص لمعالجة النصوص القصيرة مثل عناوين URL [6].

3.7.2 التمثيل الجدولي (Tabular Features)

بالتوازي مع التمثيل النصي، تم استخراج مجموعة من الخصائص الجدولية التي تصف البنية الإحصائية والهيكلية للرابط. تشمل هذه الخصائص مؤشرات مثل طول الرابط، عدد الرموز الخاصة، وعدد المقاطع، وهي خصائص أثبتت فعاليتها في التمييز بين الروابط الشرعية والاحتياطية في العديد من الدراسات السابقة.

3.7.3 التمثيل الهجين (Hybrid Representation)

تم دمج التمثيل النصي والتمثيل الجدولي ضمن متجه ميزات موحد، بما يسمح للنموذج بالاستفادة من المعلومات النصية الدقيقة والخصائص البنوية الصريحة في آن واحد. يدعم هذا النهج مفهوم "المنظورين المكملين"، حيث يلتقط TF-IDF أنماطاً نصية خفية، بينما توفر الخصائص الجدولية إشارات بنوية مباشرة عن سلوك الرابط. أظهر هذا الأسلوب قدرة أعلى على التعلم والاستقرار مقارنةً بالاعتماد على تمثيل واحد فقط [7].

3.8 حفظ النماذج وإعادة استخدامها (Model Persistence)

بعد انتهاء التدريب، تم حفظ النماذج النهائية وأدوات التحويل المرتبطة بها باستخدام مكتبة `joblib`. يتيح هذا الإجراء إعادة استخدام النموذج مباشرةً في مرحلة التشغيل دون الحاجة لإعادة التدريب، مما يحول النظام من نموذج تجريبي إلى نظام عملي قابل للاستخدام وإعادة النشر [8].

3.9 المكتبات المستخدمة

اعتمد تطبيق النظام المقترن على مجموعة من المكتبات البرمجية القياسية والمعتمدة على نطاق واسع في أبحاث التعلم الآلي وكشف التصيّد، حيث تم اختيار كل مكتبة بناءً على دورها الوظيفي وأهميتها في خط المعالجة البرمجية للنظام.

:NumPy

استُخدمت لتنفيذ العمليات العددية ومعالجة المصفوفات والمتغيرات، خاصةً عند التعامل مع الخصائص الجدولية وتحويلها إلى صيغ عددية قبل إدخالها إلى النماذج [9].

:Pandas

استُخدمت لإدارة مجموعات البيانات، قراءة الملفات، وتنفيذ عمليات التنظيف والمعالجة المساعدة مثل إزالة القيم المفقودة والتكرار، لما توفره من مرونة في التعامل مع البيانات الجدولية [9].

:Scikit-learn

- شكلت هذه المكتبة العمود الفقري للنظام، حيث استُخدمت لتنفيذ:
- تقسيم البيانات إلى مجموعات تدريب وتحقق واختبار
- استخراج الخصائص النصية باستخدام TF-IDF
- بناء خطوط المعالجة (Pipelines)
- تدريب نماذج التصنيف
- تقييم الأداء باستخدام مقاييس متعددة مثل ROC-AUC و F1-Score و Recall و Precision

كما دعمت المكتبة آليات إعادة وزن الفئات ومعالجة عدم توازن البيانات [5][6][12].

:Imbalanced-learn

استُخدمت التجربة تقنيات معالجة عدم توازن الفئات، مثل SMOTE-Tomek، بهدف مقارنة أدائها مع أسلوب إعادة وزن الفئات و اختيار الاستراتيجية الأنسب للنظام [11].

:joblib

استُخدمت لحفظ النماذج المدربة وأدوات التحويل المرتبطة بها بصيغة فعالة، مما أتاح إعادة استخدام النموذج دون الحاجة لإعادة التدريب في كل مرة، وتحويل النظام إلى نموذج قابل التشغيل العملي [8].

:Streamlit

استُخدمت لبناء واجهة تفاعلية بسيطة تسمح بادخال الروابط واختبارها بشكل مباشر، حيث يتم تمرير الرابط عبر نفس خط المعالجة المستخدم أثناء التدريب ثم عرض نتيجة التصنيف للمستخدم بشكل فوري [9].

:Seaborn و Matplotlib

استُخدمت هذه المكتبات لتمثيل النتائج بصرياً، مثل منحنيات ROC وتحليل سلوك النموذج، مما ساعد في تفسير الأداء ودعم عملية اتخاذ القرار عند اختيار النموذج النهائي [8].

:SciPy

استُخدمت لدعم بعض العمليات الإحصائية والتحليلية التي ساعدت في تقييم النتائج والتحقق من استقرار النماذج [9].

يُضمن الاعتماد على هذه المجموعة من المكتبات بناء نظام متكامل يعتمد على أدوات موثوقة ومدعومة علمياً، كما يسهل مقارنة نتائج هذا المشروع مع الدراسات السابقة في مجال كشف التصيّد الاحتيالي.

الفصل الرابع : بنية النظام ومخطط العمل

4.1 مقدمة الفصل

يهدف هذا الفصل إلى عرض الجانب التطبيقي من مشروع كشف التصيّد الاحتيالي المعتمد على تقنيات التعلم الآلي، حيث يتم فيه توضيح الخطوات العملية التي تم اتباعها لبناء النظام المقترن، بدءاً من التعامل مع البيانات، مروراً بتمثيل عناوين URL ، وانتهاءً بتدريب نماذج التصنيف المستخدمة.

يركز هذا الفصل على شرح المنهجية العملية لتنفيذ المشروع وتوضيح تسلسل العمل ومكونات النظام المختلفة، دون التطرق إلى تقييم الأداء أو تحليل النتائج، والتي سيتم تناولها في فصل مستقل لاحقاً.

ويهدف هذا العرض إلى توفير صورة واضحة عن آلية عمل أنظمة كشف التصيّد الاحتيالي المعتمدة على التعلم الآلي في البيئات الواقعية [4] [6].

4.2 نظرة عامة على الحل المقترن

يعتمد الحل المقترن في هذا المشروع على بناء نظام ذكي قادر على تصنیف عناوين URL إلى روابط شرعية وروابط تصيّد احتيالي باستخدام تقنيات التعلم الآلي، من خلال تحليل خصائص الرابط المدخل واستخراج تمثيلات متعددة له [5]

تم تصميم الحل المقترن ليعتمد على أكثر من أسلوب في تمثيل البيانات بهدف دراسة تأثير كل تمثيل على أداء نماذج التصنيف. يشمل ذلك تمثيل الروابط باستخدام الخصائص المستخرجة (Tabular Features) ، والتمثيل الخام لعناوين URL (Raw URL Representation) ، إضافةً إلى التمثيل الهجين الذي يجمع بين الطريقتين.

4.3 بنية النظام المقترن ومخطط العمل

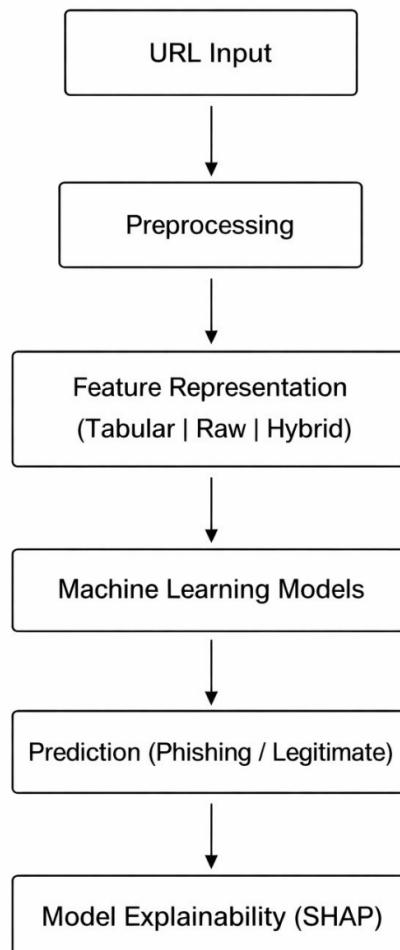
تم تصميم النظام المقترن وفق مخطط عمل متسلسل (Pipeline) يوضح المراحل الأساسية التي يمرّ بها عنوان URL منذ لحظة إدخاله إلى النظام وحتى اتخاذ القرار النهائي بشأن تصنيفه. يُعدّ اعتماد البنية المتسلسلة من الأساليب الشائعة في أنظمة التعلم الآلي، حيث يساهم في تنظيم مراحل المعالجة، وضمان الفصل الواضح بين كل مرحلة وأخرى، مما يسهل عمليات التطوير، والتحليل، وإعادة الاستخدام لاحقاً [6]. تبدأ عملية العمل بإدخال عنوان URL إلى النظام، حيث يتم التعامل معه في المرحلة الأولى كبيانات خام. بعد ذلك، يخضع الرابط لمرحلة المعالجة المسبقة (Preprocessing)، والتي تهدف إلى تنظيف البيانات وتجهيزها بالشكل المناسب قبل الانتقال إلى مرحلة التمثيل، وهو ما يُعد خطوة أساسية لضمان جودة المدخلات وتحسين أداء النماذج المستخدمة.

في مرحلة تمثيل البيانات، يتم إنشاء تمثيلات مختلفة لعنوان URL ، تشمل التمثيل المعتمد على الخصائص المستخرجة (Tabular Features) ، والتمثيل الخام لعناوين (Raw URL Representation) ، إضافةً إلى التمثيل الهجين (Hybrid Representation) الذي يجمع بين الطريقتين. يتيح هذا التنوع في التمثيل دراسة تأثير كل أسلوب على أداء نماذج التصنيف، ويُستخدم على نطاق واسع في الأبحاث الحديثة المتعلقة بكشف التصيد الاحتيالي.

بعد تجهيز تمثيلات البيانات، يتم تمريرها إلى مرحلة تدريب النماذج، حيث تُستخدم عدة خوارزميات تعلم الآلي لتعلم الأنماط المميزة لكل من الروابط الشرعية وروابط التصيد الاحتيالي. يتم تدريب كل نموذج بشكل مستقل على نوع التمثيل المعتمد، مع الحفاظ على نفس آلية التقسيم بين بيانات التدريب والتحقق والاختبار، وذلك لضمان عدالة المقارنة بين النماذج المختلفة وعزل تأثير كل من التمثيل والخوارزمية بشكل منفصل [11] [6].

في المرحلة الأخيرة، يستخدم النظام النماذج المدرّبة لتصنيف الروابط الجديدة غير المعروفة، وإصدار القرار النهائي حول طبيعة الرابط، سواء كان رابطاً شرعياً أو رابطاً تصيد احتيالي. كما يتيح تصميم النظام إمكانية تحليل قرارات النماذج وتفسيرها باستخدام أدوات التفسير المناسبة، وهو ما يُعدّ عنصراً مهماً في أنظمة الأمن السيبراني لزيادة الشفافية والثقة في قرارات النماذج المعتمدة [10] [4] .

يساعد هذا المخطط في توضيح آلية عمل النظام المقترن بشكل شامل، ويشكل أساساً منطقياً للخطوات العملية التي سيتم تفصيلها في الأقسام لاحقاً في هذا الفصل



لشكل 2 - مخطط عمل النظام

4.4 جمع البيانات ومصادرها

يعتمد هذا المشروع على مجموعتين رئيسيتين من البيانات تحتويان على عناوين URL مصنفة إلى روابط تصيد احتيالي وروابط شرعية. تم اختيار هذه البيانات من مصادر عامة وموثوقة تُستخدم على نطاق واسع في الدراسات الأكاديمية المتعلقة بكشف التصيد الاحتيالي، وذلك لضمان واقعية البيانات وإمكانية مقارنة النتائج مع الأبحاث السابقة [6] [2].

تم الحصول على روابط التصيد الاحتيالي من منصة PhishTank ، وهي قاعدة بيانات عامة متخصصة في تجميع روابط التصيد التي يتم الإبلاغ عنها والتحقق منها من قبل المجتمع وخبراء الأمن السيبراني .¹⁰ تُستخدم هذه المنصة بشكل واسع في أبحاث كشف التصيد نظراً لتحديثها المستمر واعتمادها على آلية تحقق جماعية.

أما الروابط الشرعية، فقد تم الحصول عليها من قائمة Tranco ، وهي قائمة عالمية تم تطويرها لتوفير تمثيل موثوق للموقع الشرعية الأكثر انتشاراً على شبكة الإنترنت، وتُستخدم بشكل شائع في الأبحاث الأمنية الحديثة.

بعد جمع البيانات من كلا المصادرين، تم دمجها ضمن مجموعة بيانات واحدة، مع إزالة التكرارات والتحقق من صحة التصنيفات المرتبطة بكل رابط، وذلك لضمان جودة البيانات قبل الانتقال إلى مرحلة المعالجة المسبقة واستخراج الخصائص.

الجدول (3) : مصادر مجموعات البيانات المستخدمة

Data Source	URL Type	Data Acquisition Method	Dataset Size
PhishTank	Phishing URLs	Downloaded from the official PhishTank website	32240 URLs
Tranco	Legitimate URLs	Downloaded from the official Tranco website	1000000 URLs

4.5 المعالجة المسبقة للبيانات

تُعد المعالجة المسبقة للبيانات (Data Preprocessing) خطوة أساسية قبل البدء بتدريب نماذج التعلم الآلي، إذ تهدف إلى تحسين جودة البيانات وضمان جاهزيتها لعملية التعلم، لما لذلك من تأثير مباشر على دقة النماذج واستقرارها على التعميم . في هذا المشروع، تم تطبيق مجموعة من خطوات المعالجة المسبقة على عناوين URL المجمعة من مصادرها المختلفة، بهدف تقليل الضجيج، ومنع التكرار، وتحسين موثوقية النتائج.

في المرحلة الأولى، تم فحص البيانات للتحقق من سلامتها، والتأكد من خلوها من القيم الفارغة أو غير الصالحة، وهي خطوة ضرورية لضمان عدم إدخال بيانات مشوّهة قد تؤثر سلباً على عملية التدريب.

كما تم توحيد صيغة عناوين URL لضمان اتساق البيانات، حيث أزيلت أي مسافات أو رموز غير ضرورية قد تؤدي إلى اختلافات شكلية غير مؤثرة من الناحية الدلالية، لكنها قد تربك عملية المعالجة أو استخراج الخصائص لاحقاً [5].

بعد ذلك، تم إزالة عناوين URL المكررة ضمن مجموعة البيانات، وذلك لتجنب تأثير التكرار على عملية التدريب وتقييم الأداء. إذ قد يؤدي وجود روابط مكررة إلى تحيز النموذج ورفع قيم الأداء بشكل غير واقعي، دون تحسين حقيقي في القدرة على التعميم . كما تم التأكد من عدم وجود روابط متطابقة بين مجموعات التدريب والاختبار، بهدف منع تسرب البيانات(Data Leakage)، والذي يُعد من الأخطاء الشائعة في أنظمة التعلم الآلي وقد يؤدي إلى استنتاجات مضللة حول أداء النماذج.

في المرحلة الأخيرة من المعالجة المسبقة، تم تجهيز البيانات بالشكل المناسب لمرحلة تمثيل الخصائص، سواءً للتمثيل المعتمد على الخصائص المستخرجة أو للتمثيل الخام لعناوين URL. شمل ذلك الحفاظ على عناوين URL كنصوص خام في بعض الحالات، وتحويلها إلى صيغ رقمية قابلة للمعالجة في حالات أخرى، بما يتوافق مع متطلبات كل أسلوب تمثيل مستخدم في هذا المشروع.

تساهم هذه الخطوات في ضمان جودة البيانات واستقرار عملية التدريب، وتشكل أساساً ضرورياً للحصول على نتائج دقيقة وموثوقة في المراحل اللاحقة من النظام المقترن [15]

4.6 تمثيل البيانات

تُعد طريقة تمثيل البيانات من العوامل الأساسية التي تؤثر بشكل مباشر على أداء نماذج التعلم الآلي، إذ تعتمد قدرة النموذج على اكتشاف الأنماط على نوع المعلومات المقدمة له. لذلك، تم اعتماد أكثر من أسلوب لتمثيل عناوين URL بهدف دراسة تأثير كل أسلوب على أداء نماذج كشف التصيد الاحتيالي وتحقيق مقارنة عادلة بين الطرق المختلفة

4.6.1 تمثيل عناوين URL باستخدام الخصائص المستخرجة (Tabular Features)

يعتمد هذا الأسلوب على استخراج مجموعة من الخصائص اليدوية من عناوين URL ، مثل طول الرابط، وعدد الأحرف الخاصة، وعدد الأرقام، وعدد المقاطع، وجود رموز تُستخدم غالباً في روابط التصيد. يُعد هذا النوع من التمثيل من أكثر الأساليب استخداماً في الدراسات السابقة، نظراً لبساطته وقابليته للتفسير وسهولة دمجه مع نماذج التعلم الآلي التقليدية [3] .

4.6.2 التمثيل الخام لعناوين URL (Raw URL Representation)

في هذا الأسلوب، يتم التعامل مع عنوان URL كنص خام دون استخراج خصائص يدوية، حيث يتم تحويله إلى تمثيل رقمي باستخدام تقنيات مثل TF-IDF المعتمد على. char-level n-grams يتميز هذا التمثيل بقدرته على التقاط أنماط خفية داخل بنية الرابط، إلا أنه قد يكون أقل قابلية للتفسير مقارنةً بالتمثيل الجدولي، ويطلب نماذج قادرة على التعامل مع بيانات ذات أبعاد عالية [9] .

4.6.3 التمثيل الهجين (Hybrid Representation)

يجمع التمثيل الهجين بين الخصائص المستخرجة والتمثيل الخام لعناوين URL ، بهدف الاستفادة من مزايا الطريقتين معاً. يتم دمج الخصائص العددية مع التمثيل النصي ضمن متوجه خصائص واحد، ثم تقديمها إلى نموذج التصنيف.

يهدف هذا النهج إلى تعزيز قدرة النموذج على اكتشاف الأنماط الصريحة والضمنية في آنٍ واحد، وقد أثبت فعاليته في العديد من الدراسات الحديثة المتعلقة بكشف التصيد الاحتيالي [9] [13].

4.7 الخوارزميات المستخدمة ومنهجية اختيار النموذج النهائي

تم في هذا المشروع اتباع منهجية تجريبية لاختيار النموذج النهائي لكشف روابط التصيّد الاحتيالي، حيث لم يتم اعتماد أي خوارزمية بشكل مباشر، وإنما تم اختبار مجموعة واسعة من خوارزميات التعلم الآلي باستخدام تمثيلات بيانات مختلفة، ثم تحليل نتائجها بشكل منهجي قبل الوصول إلى القرار النهائي.

تهدف هذه المنهجية إلى ضمان اختيار نموذج يتمتع بأداء عالٍ، واستقرار جيد، وقابلية للعميم في البيانات الواقعية، وهو ما تؤكده العديد من الدراسات الحديثة في مجال كشف التصيّد الاحتيالي.

4.7.1 الخوارزميات التي تم اختبارها تجريبياً

في المرحلة الأولى، تم اختبار عدد من خوارزميات التعلم الآلي الشائعة في مجال تصنيف النصوص وكشف التصيّد الاحتيالي، وذلك بهدف دراسة سلوك كل خوارزمية مع أنواع مختلفة من تمثيل البيانات. شملت الخوارزميات التي تم اختبارها ما يلى:

Logistic Regression ▪

خوارزمية خطية تُستخدم على نطاق واسع في مسائل التصنيف الثنائي، وتتميز بالاستقرار وسهولة التفسير، كما أنها فعالة مع البيانات عالية الأبعاد مثل تمثيل TF-IDF [3][7].

Support Vector Machine (SVM) ▪

تُعد من الخوارزميات القوية في مسائل التصنيف، خاصة عند التعامل مع بيانات ذات أبعاد كبيرة، حيث تعتمد على إيجاد حد فاصل أمثل بين الفئات [6].

SGD Classifier ▪

يعتمد على خوارزميات الانحدار التدرجّي، ويتميز بسرعته في التدريب، مما يجعله مناسباً للتجارب الأولية على مجموعات بيانات كبيرة الحجم [7].

Naive Bayes ▪

خوارزمية احتمالية بسيطة تعتمد على فرضية الاستقلال الشرطي بين الخصائص، وتُستخدم غالباً كنموذج مرجعي في تصنيف النصوص [3].

Random Forest ▪

نموذج تعلم تجاري يعتمد على مجموعة من أشجار القرار، ويتميز بقدرته على التعامل مع البيانات الجدولية وتقليل مشكلة الإفراط في التعلم [4].

Extra Trees (Extremely Randomized Trees) ▪

يشبه Random Forest، لكنه يعتمد على عشوائية أكبر في بناء الأشجار، مما قد يحسن التعميم في بعض الحالات [4].

XGBoost ▪

خوارزمية تعزيز تدرجّي متقدمة، أثبتت كفاءة عالية في العديد من مسائل التعلم الآلي، خاصة عند التعامل مع الخصائص الجدولية [5].

تم استخدام هذه الخوارزميات ضمن إطار تجاري واحد، مع الحفاظ على نفس آلية التقسيم ومقاييس التقييم، وذلك لعزل تأثير الخوارزمية نفسها عن باقي العوامل.

4.7.2 نتائج التجارب الأولية لجميع الخوارزميات

بعد تدريب جميع الخوارزميات المذكورة سابقاً باستخدام تمثيلات البيانات المختلفة، تم تحليل نتائجها باستخدام مجموعة موحدة من مقاييس التقييم تشمل:

الدقة (Accuracy)، الدقة الإيجابية (Precision)، الاسترجاع (Recall)، معامل F1، ومنحى ROC-AUC

الجدول (4) - مقارنة شاملة للنتائج التدريب

Representation	Feature Set / Configuration	Algorithm	F1	ROC-AUC	Accuracy	Time Cost
Tabular	Handcrafted URL features	Logistic Regression	0.8820	0.9857	0.9974	Low
Tabular	Handcrafted URL features	SVM	0.9779	0.9971	0.9994	Medium
Tabular	Handcrafted URL features	Linear SVM	0.9811	0.9364	0.9977	Medium
Tabular	Handcrafted URL features	KNN	0.5798	0.8115	0.9928	High
Tabular	Handcrafted URL features	Decision Tree	0.8995	0.9811	0.9992	Low

Tabular	Handcrafted URL features	Random Forest	0.9731	0.9964	0.9993	Medium
Tabular	Handcrafted URL features	Extra Trees	0.9794	0.9962	0.9995	Medium
Tabular	Handcrafted URL features	Gradient Boosting	0.9522	0.9832	0.9988	Medium
Tabular	Handcrafted URL features + SMOTE-Tomek	MLP (DNN)	≈0.997	≈0.999	≈0.997	Low
Tabular	Handcrafted URL features	Wide & Deep	≈0.997	≈0.999	≈0.997	Medium
Tabular	Handcrafted URL features	TabNet	≈0.997	≈0.998	≈0.997	High
Raw URL	TF-IDF (3–5 char n-grams)	Logistic Regression	≈0.99	≈0.998	≈0.99	Low
Raw URL	TF-IDF (3–5)	SGD Classifier	≈0.99	≈0.998	≈0.99	Low
Raw URL	TF-IDF (3–5)	SVM	≈0.995	≈0.999	≈0.995	Medium
Raw URL	Hashing (3–5)	SGD	≈0.992	≈0.999	≈0.992	Low
Raw URL	TF-IDF	Naive Bayes	≈0.98	≈0.997	≈0.98	Low
Raw URL	TF-IDF	XGBoost	≈0.965	≈0.993	≈0.965	Medium

أظهرت النتائج أن أداء الخوارزميات يختلف بشكل واضح تبعاً لنوع تمثيل البيانات المستخدم. فقد حفقت الخوارزميات المعتمدة على الخصائص الجدولية أداءً مقبولاً من حيث الدقة العامة، إلا أن قدرتها على اكتشاف بعض أنماط التصيّد المعقدة كانت محدودة [2][4].

في المقابل، أظهرت الخوارزميات المعتمدة على التمثيل الخام لعناوين URL باستخدام TF-IDF قدرة أعلى على التقاط الأنماط الحرافية الدقيقة، مما انعكس إيجابياً على مقياس الاسترجاع، ولكن على حساب زيادة أبعاد البيانات وعدم استقرار بعض النماذج [1][3].

تشير هذه النتائج إلى أن الاعتماد على خوارزمية واحدة أو تمثيل واحد للبيانات لا يوفر حلًّا متوازناً لمشكلة كشف التصيّد الاحتيالي، وهو ما دفع إلى الانتقال لمرحلة ترشيح النماذج الأفضل [2].

4.7.3 الخوارزميات المرشحة للنموذج النهائي

بناءً على نتائج التجارب الأولية، تم ترشيح ثلاثة خوارزميات أظهرت أداءً متوازناً واستقراراً أعلى مقارنةً بباقي النماذج، وهي:

- مع الخصائص الجدولية Extra Trees
- مع التمثيل الخام لعناوين URL (TF-IDF) مع التصيّد Logistic Regression
- (Raw + Tabular) مع التمثيل الهجين Logistic Regression

تم اختيار هذه النماذج بناءً على قدرتها على تحقيق توازن مقبول بين الدقة والاسترجاع، إضافةً إلى استقرار نتائجها عبر مجموعات البيانات المختلفة، مقارنةً بالنماذج الأخرى التي أظهرت تذبذباً أو حساسية عالية لأبعاد البيانات.

4.7.4 النموذج النهائي المعتمد (Hybrid Model)

بعد إجراء المقارنة النهائية بين النماذج المرشحة، تم اعتماد النموذج الهجين كنموذج نهائي لهذا المشروع. يعتمد هذا النموذج على دمج نوعين من البيانات ضمن تمثيل واحد على شكل جدول حيث يمثل كل صف رابطاً واحداً، بينما تتوزع الأعمدة بين:

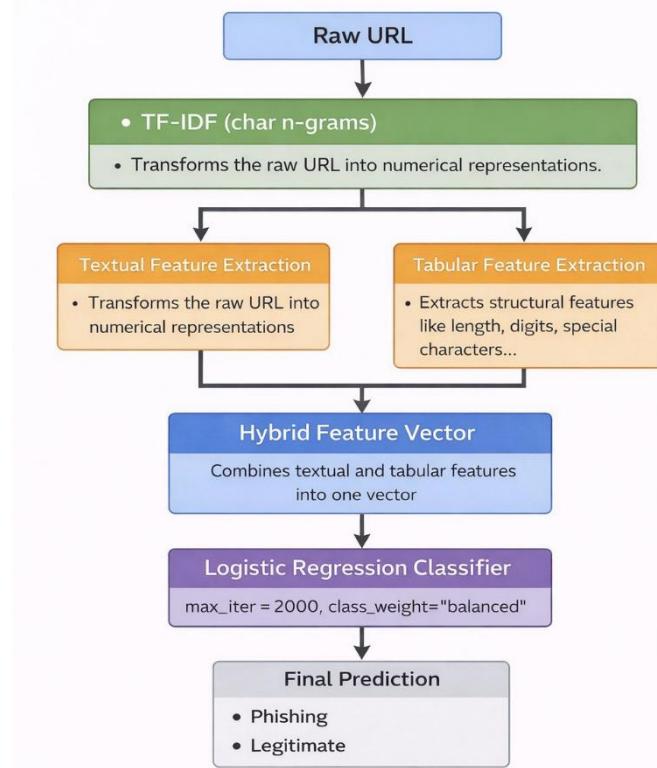
- ✓ خصائص عدديّة مستخرجة تصف بنية الرابط (Tabular Features)
- ✓ تمثيل نصي لعناوين URL باستخدام TF-IDF على مستوى الأحرف

يتم تمرير هذا التمثيل الهجين إلى نموذج Logistic Regression مع تفعيل خيار

معالجة مشكلة عدم توازن الفئات داخل البيانات [3][7].

يتيح هذا النموذج الاستفادة من الأنماط الصريرة التي توفرها الخصائص الجدولية، إلى جانب الأنماط الضمنية التي يتم التقاطها من التمثيل النصي، مما يوفر رؤية أشمل لسلوك روابط التصيد الاحتيالي.

كما يتميز النموذج النهائي ببساطته النسبية وقابليته للتفسير، وهو ما يجعله مناسباً للتطبيقات الواقعية.



شكل 3 – مخطط النموذج النهائي

4.8 مقاييس التقييم المعتمدة في النظام المقترن

نظراً لأن مشكلة كشف روابط التصييد الاحتيالي تُعد من مسائل التصنيف الحساسة أمنياً، لم يكن الاعتماد على مقاييس واحد لتقييم أداء النماذج كافياً. لذلك، تم اعتماد مجموعة متكاملة من مقاييس التقييم الإحصائية بهدف تقديم تحليل شامل ومتوازن لأداء النماذج المستخدمة، مع الأخذ بعين الاعتبار اختلاف تكاليف الأخطاء بين الفئات، خاصةً في ظل عدم توازن البيانات.

أولاً، تم استخدام مقاييس الدقة العامة (Accuracy) لقياس النسبة الإجمالية للتصنيفات الصحيحة مقارنة بعدد العينات الكلي. وعلى الرغم من شيوع هذا المقياس، إلا أنه لا يُعد كافياً بمفرده في مسائل كشف التصييد، إذ قد يعطي انطباعاً مضللاً عن أداء النموذج في حال كانت البيانات غير متوازنة.

لذلك، تم التركيز بشكل خاص على مقاييس الدقة الإيجابية (Precision)، الذي يعبر عن نسبة الروابط المصنفة على أنها تصييد والتي كانت فعلاً روابط تصييد حقيقة. يُعد هذا المقياس مهمًا لتقليل الإنذارات الخاطئة (False Positives)، والتي قد تؤدي إلى حجب روابط شرعية والتأثير سلباً على تجربة المستخدم.

في المقابل، تم اعتماد مقاييس الاسترجاع (Recall) لقياس قدرة النموذج على اكتشاف أكبر عدد ممكن من روابط التصييد الفعلية. يُعتبر هذا المقياس من أهم المؤشرات في الأنظمة الأمنية، نظراً لأن فشل النموذج في اكتشاف رابط تصييد (False Negative) قد يؤدي إلى تعرض المستخدم لهجمات خطيرة.

ولتحقيق توازن بين الدقة الإيجابية والاسترجاع، تم استخدام مقاييس F1-Score، وهو المتوسط التوافقي بين Precision و Recall. يوفر هذا المقياس تقييماً أكثر عدالة لأداء النموذج، خاصة عند مقارنة نماذج متعددة تختلف في طريقة تعاملها مع الأخطاء.

إضافةً إلى ذلك، تم اعتماد مقاييس المساحة تحت منحنى ROC (ROC-AUC) لقياس قدرة النموذج على التمييز بين الروابط الشرعية وروابط التصييد عبر مختلف قيم عتبة القرار. يتميز هذا المقياس بكونه غير متاثر مباشرةً بعدم توازن الفئات، مما يجعله مؤسراً موثقاً للمقارنة العامة بين النماذج المختلفة.

إن اعتماد هذه المجموعة من المقاييس مجتمعة يوفر تقييماً شاملًا ودقيقاً لأداء النظام المقترن، ويضمن أن اختيار النموذج النهائي لم يتم بناءً على مقياس واحد فقط، بل على تحليل متوازن يأخذ بعين الاعتبار الجوانب الأمنية والتطبيقية للنظام.

الجدول (5) - مقاييس تقييم الخوارزميات

المقياس	الاسم بالإنكليزية	التعريف	سبب الاعتماد في هذا المشروع
الدقة	Accuracy	نسبة التنبؤات الصحيحة من إجمالي العينات	مؤشر عام على أداء النموذج، لكنه غير كاف وحده في حالة عدم توافق البيانات
الدقة الإيجابية	Precision	نسبة الروابط المصنفة كتصيد وكانت فعلياً تصيداً	لتقليل الإنذارات الكاذبة وتحسين موثوقية النظام
الاسترجاع	Recall	نسبة روابط التصيد المكتشفة فعلياً من إجمالي روابط التصيد	مقياس حاسم لتقليل مخاطر عدم اكتشاف الروابط الضارة
F1 معامل	F1-Score	المتوسط التوافقي بين Precision, Recall	مقياس متوازن لمشاكل عدم توافق الفئات
المساحة تحت منحنى ROC	ROC-AUC	قدرة النموذج على التمييز بين الفئتين عبر مختلف قيم العتبة	لتقييم قوة النموذج بشكل مستقل عن العتبة المختارة

4.9 اختيار العتبة (Decision Threshold Selection)

في مسائل التصنيف الثنائي، لا يقتصر أداء النموذج على اختيار الخوارزمية أو تمثيل البيانات فقط، وإنما يتأثر بشكل مباشر بقيمة العتبة (Decision Threshold) المستخدمة لتحويل احتمالات الإخراج إلى قرارات تصنيف نهائية. تعتمد العديد من الأنظمة بشكل افتراضي على عتبة مقدارها 0.5، إلا أن هذا الاختيار قد لا يكون مناسباً في جميع الحالات، خاصة في مسائل كشف التصيد الاحتيالي التي تتسم بعدم توافق الفئات.

في هذا المشروع، تم التعامل مع مخرجات النموذج على شكل احتمالات، وتم اختيار قيمة العتبة بناءً على مجموعة التحقق (Validation Set) بدلاً من اعتماد قيمة ثابتة. تم ذلك من خلال تقييم أداء النموذج عند قيم مختلفة للعتبة، واختيار القيمة التي حفّلت بأفضل توازن بين الدقة والاسترجاع وفقاً لمعامل F1.

يسمح هذا الأسلوب بتقليل عدد حالات الخطأ الحرجية، لا سيما حالات تصنيف روابط التصيّد على أنها روابط شرعية، والتي تعد أكثر خطورة في التطبيقات الواقعية.

يساهم اعتماد عتبة مُضبوطة ديناميكياً في تحسين استقرار أداء النموذج عند الانتقال إلى بيانات جديدة، ويعكس فهماً أعمق لطبيعة المشكلة ومتطلباتها العملية.

4.10 التعامل مع عدم توازن الفئات وتسرب البيانات

تُعد مشكلة عدم توازن الفئات (Class Imbalance) من التحديات الأساسية في مسائل كشف التصيّد الاحتيالي، حيث لا تكون أعداد الروابط الشرعية وروابط التصيّد متساوية ضمن مجموعات البيانات المستخدمة. قد يؤدي هذا الخلط إلى تحيز النموذج نحو الفئة الأكثر تمثيلاً، مما ينبع عنه أداء مرتفع ظاهرياً من حيث الدقة، مع ضعف في اكتشاف الفئة الأقل تمثيلاً.

في هذا المشروع، تم التعامل مع مشكلة عدم توازن الفئات من خلال تجربة أكثر من أسلوب، بهدف اختيار الطريقة الأكثر ملاءمة لطبيعة البيانات والنمذج المستخدمة. شملت الأساليب التي تم اختبارها ما يلي:

SMOTE-Tomek، وهو أسلوب يجمع بين توليد عينات اصطناعية للفئة الأقل تمثيلاً (SMOTE) وإزالة العينات المتداخلة أو الضجيجية بين الفئات (Tomek Links)

موازنة الفئات باستخدام الأوزان (Class Weight Balancing)، من خلال إعطاء وزن أكبر للأخطاء المرتكبة على الفئة الأقل تمثيلاً أثناء عملية التدريب.

أظهرت نتائج التجارب أن استخدام SMOTE-Tomek لم يؤد إلى تحسّن ملموس في أداء النمذج، بل في بعض الحالات تسبّب بالانخفاض الاستقرار وزيادة في حالات الإفراط في التعلم. يمكن تفسير ذلك بطبيعة

بيانات عنوانين URL، حيث إن توليد عينات اصطناعية قد لا يعكس بالضرورة البنية الحقيقية للروابط الضارة، مما يؤدي إلى إدخال أنماط غير واقعية إلى البيانات.

في المقابل، أظهر أسلوب موازنة الفئات باستخدام الأوزان (class_weight = balanced) أداءً أفضل وأكثر استقراراً، حيث ساهم في تحسين قدرة النماذج على اكتشاف روابط التصيد الاحتيالي دون التأثير السلبي الكبير على الدقة العامة. يعود ذلك إلى أن هذا الأسلوب يسمح للنموذج بالتعلم من البيانات الأصلية دون تعديل توزيعها، مع زيادة حساسية النموذج للأخطاء الحرجة المتعلقة بالفئة الأقل تمثيلاً.

بناءً على هذه النتائج، تم اعتماد أسلوب Class Weight Balancing كحل نهائي لمعالجة مشكلة عدم توازن الفئات في هذا المشروع، نظراً لتوافقه مع طبيعة البيانات النصية، وتحقيقه توازنًا أفضل بين الأداء والاستقرار.

إضافةً إلى ذلك، تم اتخاذ إجراءات صارمة لمنع تسرب البيانات (Data Leakage)، حيث تم تنفيذ جميع خطوات المعالجة المسبقة واستخراج الخصائص بعد تقسيم البيانات إلى مجموعات تدريب وتحقق واختبار، كما تم التأكد من عدم وجود أي تكرار لعناوين URL بين هذه المجموعات. يساهم ذلك في ضمان واقعية النتائج وقابليتها للعميم على بيانات جديدة.

4.11 واجهة المستخدم التفاعلية (Streamlit Interface)

تم تطوير واجهة مستخدم تفاعلية للنظام المقترن باستخدام إطار العمل Streamlit، بهدف توفير وسيلة سهلة ومرنة لاختبار نموذج كشف التصيد الاحتيالي في بيئة شبه واقعية. تمكن هذه الواجهة المستخدم من التفاعل المباشر مع النموذج المدرب دون الحاجة إلى التعامل مع تفاصيل التنفيذ البرمجي، مما يعزز قابلية الاستخدام ويفرب النظام من التطبيق العملي.

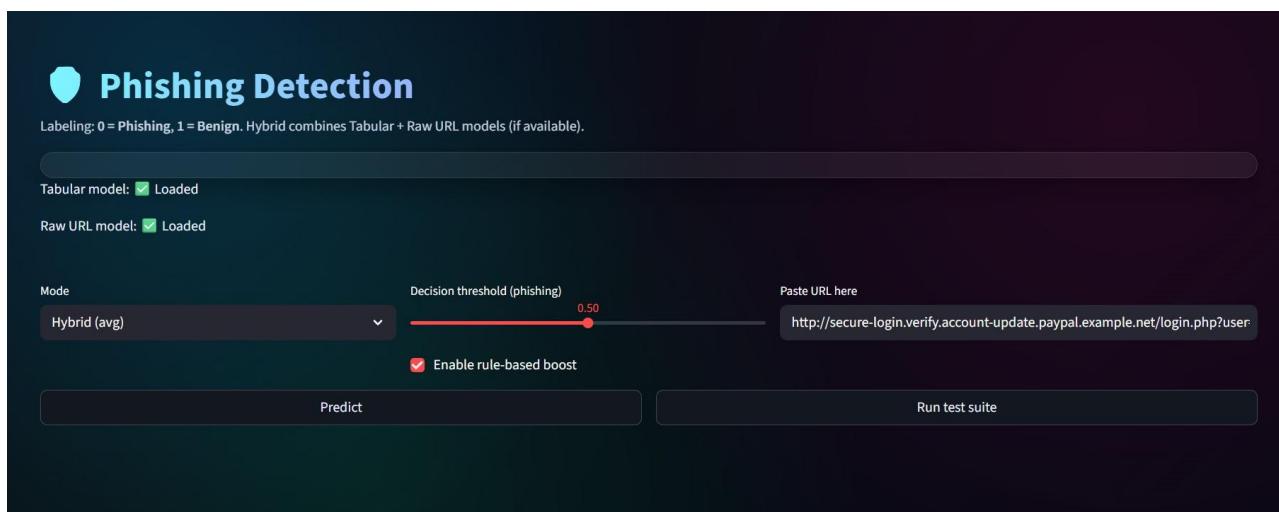
تتيح الواجهة للمستخدم إدخال عنوان URL بشكل مباشر، ليتم تمريره تلقائياً عبر خط معالجة النظام بدءاً من مرحلة استخراج الخصائص النصية باستخدام TF-IDF على مستوى الأحرف، (Pipeline)

مروراً باستخراج الخصائص الجدولية المرتبطة ببنية الرابط، ثم دمج هذه الخصائص ضمن مجّه تمثيل هجين يستخدم كمدخل للنموذج النهائي. بعد ذلك، يعرض النظام نتيجة التصنيف بشكل فوري، مبيناً ما إذا كان الرابط شرعاً أو رابط تصيّد احتيالي.

إضافةً إلى نتيجة التصنيف، تم تصميم الواجهة لعرض معلومات داعمة تساعد في تفسير قرار النموذج، مثل درجة الثقة أو الاحتمالية المرتبطة بالنتيجة، مما يمنح المستخدم تصوّراً أوضحاً عن مدى خطورة الرابط المدخل. ويُعد هذا الجانب مهمًا بشكل خاص في الأنظمة الأمنية، حيث لا يقتصر دور النظام على إعطاء قرار ثانٍ فقط، بل يمتد إلى دعم المستخدم في اتخاذ قرار واعٍ.

تتميز واجهة Streamlit بخفة الأداء وسرعة الاستجابة، إذ تعمل في الزمن الحقيقي (Real-Time) دون الحاجة إلى موارد حسابية مرتفعة، مما يجعلها مناسبة للتجارب التفاعلية والعروض التوضيحية. كما أن بساطة التصميم وسهولة الاستخدام تسهمان في جعل النظام قابلاً للاستخدام من قبل فئات مختلفة من المستخدمين، سواء كانوا باحثين، مطوريين، أو حتى مستخدمين غير تقنيين.

بشكل عام، تشكّل واجهة المستخدم المبنية باستخدام Streamlit عنصراً مكملاً مهماً للنظام المقترن، حيث تُبرز الجانب التطبيقي للمشروع، وتؤكّد إمكانية دمج النموذج الهجين ضمن تطبيقات عملية قابلة للاستخدام في سيناريوهات حقيقة لكشف التصيّد الاحتيالي.



لشكل 4 - واجهة المشروع

الفصل الخامس : النتائج والآفاق المستقبلية

5.1 مقدمة الفصل

يهدف هذا الفصل إلى عرض خلاصة النتائج التي تم التوصل إليها في هذا البحث، مع تسلیط الضوء على مساهمة النظام المقترن في مجال كشف التصيّد الاحتيالي اعتماداً على تحليل عناوين URL وتقنيات التعلم الآلي. كما يستعرض هذا الفصل مجموعة من الآفاق المستقبلية التي يمكن تطوير النظام باتجاهها، سواء على المستوى التقني أو التطبيقي، بما يعزّز من فعاليته ويزيد من قابليته للاستخدام في بيئات واقعية.

5.2 خلاصة النتائج التجريبية

أظهرت النتائج التجريبية أن النظام المقترن القائم على التمثيل الهجين لعناوين URL يحقق أداءً متقدماً مقارنة بالاعتماد على تمثيل واحد فقط. إذ ساهم دمج التمثيل النصي (TF-IDF) مع الخصائص الجدولية في تحسين قدرة النموذج على التمييز بين الروابط الشرعية والاحتيالية، خاصة في الحالات الحدودية التي يصعب تصنيفها باستخدام نهج واحد.

كما أظهرت تجارب معالجة عدم توازن الفئات أن اعتماد إعادة وزن الفئات (class weighting) قدم أداءً أكثر استقراراً من تقنيات إعادة التوليد الاصطناعي، لا سيما عند التعامل مع تمثيلات عالية الأبعاد. إضافةً إلى ذلك، ساهم ضبط عتبة القرار في تحسين قدرة النظام على تقليل الأخطاء الحرجية من نوع **False Negative**، وهو عنصر بالغ الأهمية في الأنظمة الأمنية.

تؤكد هذه النتائج أن النظام المقترن يتمتع بقدرة جيدة على التعلم، ويشكل أساساً مناسباً لبناء أنظمة كشف تصيّد قابلة للتطبيق العملي.

5.3 الآفاق المستقبلية للنظام المقترن

على الرغم من النتائج المشجعة التي حققها النظام المقترن، لا يزال هناك مجال واسع لتطويره وتوسيعه ليواكب متطلبات الاستخدام الواقعي والتطور المستمر في أساليب التصيد الاحتيالي. يمكن تلخيص أبرز الآفاق المستقبلية كما يلي:

5.3.1 تطوير واجهة ويب معتمدة على HTML (HTML-based Web Application)

من أبرز اتجاهات التطوير المستقبلية تحويل النظام الحالي إلى تطبيق ويب متكامل يعتمد على واجهات HTML و CSS و JavaScript، بحيث يمكن للمستخدم التفاعل مع النظام مباشرة عبر المتصفح دون الحاجة لتشغيله محلياً.

يتيح هذا التوجه:

- تحسين سهولة الاستخدام والوصول للنظام.
- دمج النظام ضمن بوابات أمنية أو منصات توعوية.
- دعم إدخال الروابط وتحليلها في الزمن الحقيقي (Real-time).
- يمكن ربط الواجهة الخلفية (Backend) بالنموذج المدرب عبر واجهات برمجية (APIs)، مما يسمح بتوسيع النظام ليخدم عدداً كبيراً من المستخدمين في آن واحد.

5.3.2 تطوير إضافة متصفح (Browser Extension)

يُعد تطوير إضافة لمتصفح مثل Firefox Extension أو Chrome Extension من أهم التطبيقات العملية المستقبلية للنظام المقترن. تتيح هذه الإضافة فحص الروابط تلقائياً أثناء تصفح المستخدم للويب، سواء عند النقر على رابط أو عند تحميل صفحة جديدة.

يسمح هذا التوجه بما يلي:

- الكشف الاستباقي عن الروابط الاحتيالية قبل فتحها.
- تحذير المستخدم بشكل فوري دون الحاجة لأي تدخل يدوي.
- تقليل خطر الوقوع ضحية لهجمات التصيد في الحياة اليومية.
- يمثل هذا السيناريو خطوة مهمة نحو دمج النموذج في بيئه استخدام حقيقية، وليس فقط نظام تجريبي.

5.3.3 دمج تحليل محتوى الصفحة (HTML Content Analysis)

في النسخة الحالية، يعتمد النظام على تحليل عنوان URL فقط، وهو ما يمنه سرعة عالية وكفاءة جيدة. ومع ذلك، يمكن في المستقبل توسيع النظام ليشمل تحليل محتوى الصفحة نفسها بعد تحميلها، مثل:

- تحليل بنية HTML.
- فحص النماذج (Forms) وحقول إدخال البيانات.
- تحليل الروابط الداخلية والخارجية.

يسمح هذا الدمج ببناء نظام كشف متعدد المستويات (Multi-layer Detection) يجمع بين سرعة تحليل الرابط ودقة تحليل المحتوى.

5.3.4 التعلم المستمر وتحديث النموذج (Continuous Learning)

من الآفاق المستقبلية المهمة اعتماد آلية تعلم مستمر Online ، بحيث يتم تحديث النموذج دورياً باستخدام روابط جديدة يتم اكتشافها بمرور الزمن. يساهم هذا النهج في:

- مواكبة أساليب التصيد الحديثة.
- تقليل تدهور الأداء مع تغير أنماط الهجوم.
- تحسين قدرة النموذج على التكيف مع البيانات الجديدة.

5.3.5 توسيع مصادر البيانات وتحسين التعميم

يمكن تعزيز أداء النظام مستقبلاً من خلال:

- دمج مصادر بيانات إضافية غير مقتصرة على مصدر واحد.
- دعم لغات وأنماط روابط مختلفة.

• توسيع مجموعة الخصائص المستخرجة لتشمل خصائص زمنية أو سياقية.

يساعد هذا التوسيع في تحسين قدرة النموذج على التعميم وتقليل تحيزه لمجموعة بيانات محددة.

المراجع:

1. Ma et al., KDD 2009 – Malicious URL detection
2. Verma & Gupta, ARES 2015 – Lexical URL preprocessing
3. Marchal et al., IEEE TrustCom 2014 – Dataset cleaning & bias
4. Sahingoz et al., Expert Systems with Applications 2019
5. Le et al., Computers & Security (Elsevier) 2018
6. Aljofey et al., Applied Sciences 2024 – Hybrid phishing systems
7. Kohavi, R., "A Study of Cross-Validation and Bootstrap," IJCAI, 1995
8. Han, J., Kamber, M., Pei, J., Data Mining: Concepts and Techniques, Morgan Kaufmann
9. Kapoor & Narayanan, "Leakage and the Reproducibility Crisis," ICML Workshop, 2022
10. Ma et al., KDD 2009 – URL-based malicious detection
11. Verma & Gupta, ARES 2015 – Lexical features in phishing
12. Marchal et al., IEEE TrustCom 2014 – Phishing detection systems
13. Sahingoz et al., Expert Systems with Applications 2019
14. Le et al., Computers & Security (Elsevier) 2018
15. Aljofey et al., Applied Sciences 2024 – Hybrid phishing models
16. Scikit-learn documentation – ML pipelines & interpretability
17. Chen & Guestrin, KDD 2016 – XGBoost
18. Jain & Gupta, Security and Communication Networks 2016 <
19. Ma, J., et al., "Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs," KDD, 2009.
20. Garera, S., et al., "A Framework for Detection and Measurement of Phishing Attacks," WORM, 2007.

21. Le, H., et al., “URL-based phishing detection using machine learning,” *Computers & Security*, Elsevier, 2018.
22. Marchal, S., et al., “Know Your Phish: Novel Techniques for Detecting Phishing Sites,” *IEEE TrustCom*, 2014.
23. Verma, R., et al., “Detecting Phishing Websites Using Lexical Features,” *ARES*, 2015.
24. Sahingoz, O.K., et al., “Machine Learning Based Phishing Detection from URLs,” *Expert Systems with Applications*, 2019.
25. Tranco: A Research–Oriented Top Sites Ranking, <https://tranco-list.eu>
26. PhishTank, <https://www.phishtank.com>
27. Aljofey, A., et al., “An Effective Phishing Detection Model Based on Hybrid Features,” *Applied Sciences*, 2024.
28. Scikit-learn Developers, “Machine Learning in Python,” *JMLR*, 2011.
29. Chen, T., Guestrin, C., “XGBoost: A Scalable Tree Boosting System,” *KDD*, 2016.
30. Abutair, H., Belghith, A., “Using case–based reasoning for phishing detection,” *Procedia Computer Science*, 2017
31. Jain, A., Gupta, B., “Phishing Detection: Analysis of Visual Similarity,” *Security and Communication Networks*, 2016.
32. Fawcett, T., An Introduction to ROC Analysis, *Pattern Recognition Letters*, 2006.
33. Saito, T., & Rehmsmeier, M., The Precision–Recall Plot Is More Informative than the ROC Plot, *PLOS ONE*, 2015.
34. Powers, D. M. W., Evaluation: From Precision, Recall and F-Measure to ROC, *Journal of Machine Learning Technologies*, 2011.
35. Sahingoz et al., Machine Learning Based Phishing Detection from URLs, *Expert Systems with Applications*, 2019.
36. Le et al., URL-based Phishing Detection Using Machine Learning, *Computers & Security*, 2018.