Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«МЕТОД І ЗАСІБ ДЛЯ ВИЯВЛЕННЯ ФІШИНГОВИХ САЙТІВ ІЗ ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ»

Виконав: студент 2 курсу, групи 1БС-24 м
спеціальності 125 Кібербезпека та захист
інформації
_____ Максим ГНАТЮК

Керівник: к. т. н., доцент каф. ЗІ
_____ Володимир ГАРНАГА
« 19 » грудня 2025 р.

Опонент: д.т.н., проф., зав. каф. ПЗ
_____ Олександр РОМАНЮК
« 19 » грудня 2025 р.

Допущено до захисту
В. о. зав. каф. ЗІ д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
« 19 » грудня 2025 р.

Вінниця ВНТУ – 2025 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти II (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 125 Кібербезпека та захист інформації
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
В. о. зав. каф. ЗІ д. т. н., проф.
_____Володимир ЛУЖЕЦЬКИЙ
«24» 09 2025 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Гнатюку Максиму Віталійовичу

1. Тема роботи: «Метод і засіб для виявлення фішингових сайтів із використанням машинного навчання» керівник роботи: Гарнага Володимир Анатолійович, к. т. н., доцент кафедри ЗІ, затверджені наказом ректора ВНТУ від 24 вересня 2025 року № 313.
2. Строк подання студентом роботи 19 грудня 2025 р.
3. Вихідні дані до роботи:
   - дані про актуальні фішингові загрози;
   - датасет фішингових і легітимних вебсайтів;
   - навчена модель Random Forest.
4. Зміст текстової частини: Вступ. 1. Аналіз і постановка задачі. 2. Розробка методу та навчання моделі. 3. Реалізація та дослідження програмного засобу. 4. Економічна частина. Висновки. Список використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу: Візуалізація 50 найважливіших ознак моделі, Аналіз набору даних, Схема архітектури програмного засобу, Візуалізація програмного засобу, Аналіз тестування програмного засобу, Показники економічної ефективності.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	В. ГАРНАГА, к.т.н., доц. каф. ЗІ	25.09.25	06.10.25
2	В. ГАРНАГА, к.т.н., доц. каф. ЗІ	25.09.25	03.11.25
3	В. ГАРНАГА, к.т.н., доц. каф. ЗІ	25.09.25	07.12.25
4	О. ЛЕСЬКО, зав. каф. ЕПВМ, к. е. н., доц	25.09.25	08.12.25

7. Дата видачі завдання 24 вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примі
1	Аналіз завдання. Вступ	24.09.2025 – 26.09.2025	
2	Аналіз інформаційних джерел за напрямком магістерської кваліфікаційної роботи	27.09.2025 – 07.10.2025	
3	Науково-технічне обґрунтування розробки програмного засобу виявлення фішингових сайтів	11.10.2025 – 22.10.2025	
4	Аналіз стандартів, існуючих методів та моделей виявлення фішингових веб-ресурсів	23.10.2025 – 26.10.2025	
5	Аналіз та формування вимог до методу виявлення фішингових сайтів	27.10.2025 – 02.11.2025	
6	Розробка методу виявлення фішингових сайтів на основі методів машинного навчання	03.11.2025 – 10.11.2025	
7	Застосування розробленого методу для аналізу та класифікації фішингових веб-ресурсів	10.11.2025 – 17.11.2025	
8	Розробка розділу економічного обґрунтування доцільності створення програмного засобу	18.11.2025 – 22.11.2025	
9	Оформлення пояснювальної записки магістерської кваліфікаційної роботи	23.11.2025 – 29.11.2025	
10	Попередній захист та доопрацювання магістерської кваліфікаційної роботи	29.11.2025 – 11.12.2025	
11	Перевірка роботи на наявність текстових запозичень	12.12.2025 – 15.12.2025	
12	Представлення магістерської кваліфікаційної роботи до захисту, рецензування	16.12.2025 – 19.12.2025	
13	Захист магістерської кваліфікаційної роботи	19.12.2025 – 23.12.2025	

Студент _____ Максим ГНАТЮ

Керівник роботи _____ Володимир ГАРНАГ

АНОТАЦІЯ

УДК 004.056

Гнатюк М. Метод і засіб для виявлення фішингових сайтів із використанням машинного навчання. Магістерська кваліфікаційна робота зі спеціальності 125 – Кібербезпека та захист інформації, освітня програма – Безпека інформаційних і комунікаційних систем. Вінниця: ВНТУ, 2025. с. 96.

Укр. мовою. Бібліогр.: 38 назв; рис.: 13; табл.: 6.

Магістерська кваліфікаційна робота присвячена розробці методу та програмного засобу для виявлення фішингових веб-сайтів із використанням алгоритмів машинного навчання. У роботі проведено аналіз сучасних фішингових загроз, класифікацію типів атак та огляд існуючих підходів до їх виявлення. Розроблено модель класифікації URL-адрес на основі алгоритму Random Forest, яка продемонструвала високу точність під час тестування. Створено програмний засіб “Phishing Detector” браузерний плагін із локальним модулем машинного навчання, що виконує перевірку сайтів у режимі реального часу та інформує користувача про потенційні ризики. У роботі проведено експериментальні дослідження, здійснено оцінку ефективності моделі та побудовано візуалізації результатів.

Ілюстративна частина містить візуалізацію структури системи, алгоритмів перевірки та результатів моделювання.

В економічному розділі виконано розрахунок витрат на розробку.

Ключові слова: фішинг, машинне навчання, Random Forest, веб-безпека, URL-аналіз, виявлення загроз, пояснювана модель машинного навчання, класифікація веб-ресурсів, аналіз ознак, браузерний плагін.

ABSTRACT

Hnatiuk M. Method and Tool for Detecting Phishing Websites Using Machine Learning. Master's Thesis in the specialty 125 – Cybersecurity and Information Protection, educational program – Information and Communication Systems Security. Vinnytsia: VNTU, 2025. p. 96.

In Ukrainian. Bibliography: 38 titles; figures: 13; tables: 6.

The master's thesis is devoted to the development of a method and software tool for detecting phishing websites using machine learning algorithms. The study includes an analysis of modern phishing threats, a classification of attack types, and an overview of existing detection approaches. A URL classification model based on the Random Forest algorithm has been developed, which demonstrated high accuracy during testing. A software tool "Phishing Detector" was created a browser extension with a local machine-learning module that checks websites in real time and informs users about potential risks. The thesis presents experimental research, evaluation of the model's performance, and visualization of the obtained results.

The illustrative section includes visualizations of the system structure, verification algorithms, and modeling outcomes.

The economic section contains calculations of software development costs and the assessment of implementation feasibility.

Keywords: phishing, machine learning, Random Forest, web security, URL analysis, threat detection, explainable machine learning model, web resource classification, feature analysis, browser extension.

ЗМІСТ

ВСТУП

В умовах розвитку сучасних цифрових технологій та розширення сфери використання мережі Інтернет, питання інформаційної безпеки користувачів залишається надзвичайно актуальним. Одним із найпоширеніших методів несанкціонованого доступу до персональних даних залишається фішинг. Фішинг – це вид інтернет-шахрайства, метою якого є виманювання у користувачів конфіденційної інформації: логінів, паролів, банківських реквізитів та персональних даних [1]. Зловмисники застосовують методи соціальної інженерії, надсилаючи електронні листи або створюючи веб-сторінки, що імітують дизайн легітимних популярних онлайн-сервісів, таких як банківські системи, маркетплейси чи соціальні мережі. У результаті користувачі, не підозрюючи про підробку, самостійно вводять свої дані на фішингових сайтах, надаючи шахраям повний доступ до їхньої персональної інформації.

Традиційні підходи до виявлення фішингових сайтів – це використання чорних списків, сигнатур або ручної модерації, вони мають обмежену ефективність, оскільки не здатні вчасно виявляти нові або модифіковані загрози. Тому актуальною задачею є розробка інтелектуальних систем автоматичного виявлення фішингових ресурсів, здатних аналізувати структуру, поведінку та ознаки сайтів, а також адаптуватися до нових типів атак у динамічному інформаційному середовищі [2].

Сучасні методи машинного навчання відкривають широкі можливості для вирішення цього завдання. Наявність великої кількості відкритих датасетів фішингових і легітимних сайтів дозволяє створювати навчені моделі високої точності, які не залежать від фіксованих шаблонів або баз даних. Такі моделі можна періодично перенавчати на нових вибірках, що забезпечує їхню актуальність і високу точність навіть за умов появи нових схем фішингу, які ще

не внесені до жодних реєстрів. Машинне навчання дозволяє системі аналізувати комплекс характеристик веб-ресурсу: структуру URL, дані домену, метадані сторінки, параметри сертифіката, поведінкові ознаки та самостійно виявляти закономірності, притаманні фішинговим сайтам.

На відміну від шаблонних або сигнатурних методів, розроблена модель не шукає збігів із відомими зразками, а виконує інтелектуальний аналіз параметрів сторінки під час її завантаження, отримує з сайту необхідні дані та формує висновок про рівень його безпечності. Це забезпечує можливість швидкої оцінки нових сайтів у режимі реального часу, без потреби ручного оновлення баз знань. Однією з ключових переваг і наукової новизни запропонованого рішення є те, що розроблений програмний засіб не лише визначає рівень ризику, а й пояснює користувачу, які саме ознаки викликали підозру. Така прозорість дає змогу користувачу самостійно оцінити безпечність веб-ресурсу та прийняти обґрунтоване рішення щодо продовження або припинення роботи із сайтом. Подібна функціональність є рідкісною серед сучасних антифішингових систем, які зазвичай лише блокують доступ до сторінки без пояснення причин. Таким чином, розроблений у межах магістерської кваліфікаційної роботи метод і засіб для виявлення фішингових сайтів із використанням машинного навчання є не лише автоматизованим і адаптивним, але й орієнтованим на пояснюваність рішень і взаємодію з користувачем, що робить його ефективним інструментом для підвищення рівня кібербезпеки у веб-просторі.

Об’єктом дослідження є процес виявлення фішингових вебсайтів у мережі Інтернет.

Предметом дослідження є програмні методи та засоби машинного навчання для автоматичного розпізнавання фішингових ресурсів.

Метою магістерської кваліфікаційної роботи є розробка методу та програмного засобу для виявлення фішингових веб-сайтів на основі алгоритмів

машинного навчання, здатного забезпечити високу точність класифікації та роботу в реальному часі.

Для досягнення поставленої мети необхідно:

- проаналізувати сучасні методи та системи виявлення фішингових сайтів;
- сформувати набір ознак для розпізнавання фішингових веб-ресурсів;
- обрати та дослідити алгоритми машинного навчання для класифікації сайтів;
- розробити модель процесу виявлення фішингових сайтів;
- створити програмний засіб у вигляді браузерного плагіна, який здійснює перевірку сайтів у режимі реального часу;
- провести експериментальні дослідження.

Наукова новизна магістерської роботи полягає у створенні адаптивного методу автоматичного виявлення фішингових сайтів, що використовує аналітичну модель машинного навчання для оцінки ознак веб-ресурсів у реальному часі та механізм пояснення результатів класифікації для користувача. Практична цінність роботи полягає у створенні програмного засобу «Phishing Detector», який може бути використаний для підвищення рівня кіберзахисту користувачів, організацій і корпоративних систем.

Апробація результатів магістерської кваліфікаційної роботи здійснювалася шляхом доповіді на науково-практичній конференції Вінницького національного технічного університету, де були представлені основні результати дослідження та програмний засіб для виявлення фішингових сайтів із використанням машинного навчання [3].

Отримані результати можуть бути застосовані у сфері розробки антивірусних систем, браузерних розширень, систем моніторингу веб-загроз та інформаційного захисту інфраструктурних об’єктів.

1 АНАЛІЗ І ПОСТАНОВКА ЗАДАЧІ

1.1 Актуальність теми та характеристика фішингових атак

У сучасному цифровому світі, де значна частина соціальної, фінансової та професійної активності перемістилася в онлайн-середовище, питання інформаційної безпеки набуває стратегічного значення. Однією з найпоширеніших форм кіберзлочинності, спрямованої на несанкціоноване отримання конфіденційної інформації, є фішинг, це метод який поєднує технічні засоби з елементами соціальної інженерії.

Фішинг (англ. phishing, від fishing – «риболовля») – це вид шахрайства, під час якого зловмисники намагаються обманом отримати від користувачів персональні або фінансові дані: логіни, паролі, номери банківських карт, реквізити доступу до сервісів тощо. Його сутність полягає у створенні підроблених веб-сторінок або повідомлень, які візуально та змістовно імітують справжні ресурси, з метою змусити користувача самостійно розкрити свої дані. Наприклад, шахраї можуть надіслати електронного листа, який виглядає як офіційне повідомлення від банку або служби підтримки, із проханням «підтвердити» облікові дані за посиланням, що веде на фішингову копію сайту [4].

У більшості випадків користувач не підозрює підміну, оскільки дизайн сторінки, логотипи, доменне ім’я або текст повідомлення виглядають правдоподібно. Саме поєднання технічного копіювання й психологічного впливу робить фішинг одним із найнебезпечніших і наймасовіших кіберзагроз сучасності.

За даними звітів Statista та Anti-Phishing Working Group (APWG), лише за 2024 рік кількість фішингових інцидентів у світі перевищила 1,3 мільйона зареєстрованих випадків. Близько 85 % компаній у світі повідомили про хоча б одну спробу фішингової атаки протягом року, а понад 90 % кібератак на

корпоративному рівні починаються саме з фішингових листів. Середній економічний збиток від однієї успішної атаки може сягати від 1000 до 50 000 доларів США залежно від цілей та масштабу інциденту. Для державних установ, банківських структур і підприємств критичної інфраструктури фішинг є першим етапом складніших атак, зокрема зараження систем шкідливим ПЗ (ransomware) або крадіжки облікових записів адміністраторів. Такі тенденції свідчать, що традиційні методи захисту (чорні списки, фільтри спаму, ручна модерація) вже не забезпечують достатнього рівня безпеки зловмисники постійно змінюють тактики та інструменти, створюючи тисячі нових сайтів щодня [5].

Фішингові атаки мають різні форми та канали реалізації. Основні типи включають:

- Email-фішинг, це найпоширеніший метод, що полягає у надсиланні підроблених листів від імені банків, торгових платформ або соціальних мереж;
- такі повідомлення зазвичай містять посилання на фішинговий сайт або вкладення зі шкідливим ПЗ;
- Web-фішинг створення копій легітимних веб-сторінок, де користувачі вводять свої облікові дані, не помічаючи підміни домену або SSL-сертифіката;
- часто використовується підміна символів у домені (наприклад, paypa1.com замість paypal.com);
- Smishing (SMS-фішинг), це надсилання коротких повідомлень із посиланням на підроблений сайт або проханням підтвердити дані;
- Vishing (Voice-фішинг) телефонні дзвінки, у яких шахраї видають себе за представників банку, поліції чи сервісів підтримки.

- клоновий фішинг створення копій легітимних повідомлень, у яких оригінальні посилання замінено на шкідливі;
- фішинг у соціальних мережах, зловмисники створюють підроблені профілі компаній або знайомих користувача, надсилаючи фальшиві запити про допомогу, подарунки чи інвестиції [6].

Основною складовою фішингу є маніпуляція користувачами. Найчастіше застосовуються такі прийоми:

- психологічний тиск і терміновість: повідомлення містять фрази типу “Ваш обліковий запис буде заблоковано”, “Підтвердьте дані протягом 10 хвилин”;
- імітація довіри: використання логотипів і корпоративного стилю відомих компаній (Google, Meta, PayPal) ;
- підміна доменів: створення схожих URL із незначними змінами символів (homograph attack) ;
- обіцянка виграшу або винагороди: повідомлення про “лотереї”, “подарунки”, “знижки”;
- використання технічних посилань і вкладень: вбудовані скрипти або файли, що запускають шкідливий код;
- такі методи ґрунтуються на людському факторі: довірливості, поспіху чи необізнаності користувачів, і тому жодна система безпеки не може повністю усунути ризики без належного рівня автоматизації та штучного інтелекту [7].

1.2 Огляд методів і систем виявлення фішингових сайтів

У сучасних умовах зростання кількості кіберзагроз, пов’язаних із фішингом, ефективне виявлення шкідливих сайтів стало ключовим завданням для систем

інформаційної безпеки. Методи виявлення можна умовно поділити на традиційні (статичні) та інтелектуальні (на основі машинного навчання та штучного інтелекту). Кожен із підходів має свої переваги, недоліки та сфери ефективного застосування. До традиційних методів належать способи, що базуються на статичному аналізі характеристик веб-ресурсу, без урахування поведінкових або контекстних змін.

Основні серед них:

- метод чорних і білих списків (blacklist / whitelist), його суть полягає у перевірці домену або IP-адреси сайту у заздалегідь створених базах фішингових та довірених ресурсів. Прикладом реалізації є сервіси Google Safe Browsing, PhishTank, Netcraft, які автоматично блокують відомі шкідливі сайти. Недоліком цього підходу є неможливість своєчасного виявлення нових або модифікованих фішингових сторінок, які ще не потрапили до бази даних;
- сигнатурний аналіз, передбачає пошук у коді сторінки специфічних фрагментів, шаблонів або ключових ознак (наявність IP-адреси у URL, підозрілих тегів JavaScript, форм без валідації тощо). Метод забезпечує високу швидкість перевірки, проте має низьку адаптивність: достатньо змінити структуру сторінки або шифрувати посилання, і система не зможе виявити загрозу;
- евристичні методи, икористовують набір експертно визначених правил для оцінювання ризику. Наприклад, довжина URL, кількість піддоменів, наявність символів “@” чи “-”, термін існування домену, наявність HTTPS. Евристика дозволяє виявляти аномалії без навчальних даних, але залежить від коректності заданих правил і не враховує нових тенденцій у шахрайських схемах;

– аналіз сертифікатів SSL/TLS, враховує наявність та валідність сертифіката безпеки. Однак через поширення безкоштовних сертифікатів (Let’s Encrypt) цей показник уже не є вирішальним [8].

Отже, традиційні підходи характеризуються простотою реалізації і швидкістю роботи, але не здатні адаптуватися до нових загроз та не забезпечують високої точності виявлення. Інтелектуальні системи на базі машинного навчання (ML) та нейронних мереж (AI) забезпечують динамічний аналіз великої кількості характеристик веб-сторінки, що дає змогу виявляти навіть раніше невідомі фішингові ресурси. Цей підхід полягає у побудові моделі класифікації, яка навчається на історичних даних (dataset) і надалі визначає, чи є новий сайт фішинговим [9]. Порівняльну характеристику традиційних підходів та методів машинного навчання для виявлення фішингових сайтів наведено в таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз підходів

Характеристика	**Традиційні методи**	**Методи машинного навчання**
Приклади	Google Safe Browsing, PhishTank	Random Forest, SVM, MLP AI-моделі
Тип аналізу	Статичний (URL, SSL, чорні списки)	Динамічний, на основі даних
Точність	Обмежена, лише для відомих шаблонів	Висока, здатність виявляти нові атаки
Швидкість	Висока, але залежить від актуальності баз	Середня, залежить від обчислювальної потужності
Адаптивність	Низька, потребує ручного оновлення	Висока, можливість перенавчання
Рівень хибних спрацювань	Високий	Низький (за правильно підібраних ознаках)

Основні алгоритми:

– Decision Tree (дерева рішень) інтерпретований метод, що створює правила класифікації за окремими ознаками (наприклад, «якщо домен молодший за 6 місяців – ймовірно фішинг») ;

- Random Forest ансамблевий метод, який комбінує багато дерев для підвищення точності; відзначається стабільністю та стійкістю до шумів у даних;
- Support Vector Machine (SVM) створює гіперплощину для розділення фішингових і нефішингових зразків; ефективний при великій кількості числових ознак;
- Neural Networks (нейронні мережі) моделюють складні взаємозв'язки між ознаками сторінок. Мережа типу Multilayer Perceptron (MLP) навчається за допомогою алгоритму Backpropagation, коригуючи ваги та підвищуючи точність класифікації [10].

На сьогодні існує низка рішень, які реалізують зазначені методи:

- Google Safe Browsing базується на чорних списках і поведінковому аналізі URL;
- PhishTank це відкрита база для колективного обміну інформацією про фішингові сайти;
- Microsoft SmartScreen, поєднує сигнатурний та евристичний аналіз; AI-PhishNet, CANTINA, PHP-ML Phishing Detector це приклади систем, що використовують машинне навчання або нейронні мережі для класифікації сайтів [11].

1.3 Переваги та актуальність розробленої моделі

Існуючі системи захисту, такі як Google Safe Browsing, PhishTank чи Netcraft, базуються переважно на централізованих базах даних, сигнатурному пошуку або перевірці за чорними списками. Хоча ці підходи ефективні проти вже ідентифікованих фішингових сайтів, вони не забезпечують належного рівня захисту від динамічно змінюваних або новостворених веб-ресурсів, які не

встигають потрапити до баз. Саме тому розробка адаптивної моделі виявлення фішингових сайтів на основі машинного навчання є актуальною і перспективною.

Розроблений метод ґрунтується на використанні алгоритму Random Forest, який є одним із найефективніших ансамблевих підходів до класифікації [12]. Його перевага полягає у здатності працювати з великою кількістю різнорідних ознак (довжина URL, вік домену, наявність HTTPS, кількість гіперпосилань, індексація у пошукових системах тощо) та формувати узагальнену модель, що виявляє приховані закономірності у структурі даних. Завдяки цьому система може з високою точністю розпізнавати фішингові сайти навіть за відсутності явних шаблонів чи попередньо визначених сигнатур. На відміну від статичних методів, що потребують постійного оновлення баз, навчена модель самостійно аналізує нові веб-ресурси, використовуючи закономірності, сформовані під час тренування.

Ще однією перевагою розробленої системи є можливість динамічного перенавчання моделі на нових даних. Це дозволяє підтримувати актуальність алгоритму та підвищувати його точність у міру появи нових видів фішингових атак. Такий підхід гарантує, що навіть при зміні технік шахрайства або появі нових схем маскування система залишатиметься ефективною, не потребуючи кардинального оновлення структури [13].

Важливою відмінністю запропонованої моделі є її локальне функціонування, модель інтегрована у вигляді браузерного плагіна, який виконує перевірку сайтів без необхідності постійного звернення до зовнішніх серверів. Це підвищує швидкість обробки, мінімізує ризики витоку даних та дозволяє використовувати систему навіть у середовищах із обмеженим доступом до Інтернету. Таким чином, користувач отримує захист у реальному часі без додаткових затримок або втручання у приватність.

Окрему увагу приділено інтерактивності та прозорості роботи системи. На відміну від більшості сучасних антифішингових засобів, що лише блокують доступ до підозрілих сторінок, розроблений плагін не просто сповіщає користувача про загрозу, а й показує, які саме ознаки викликали підозру наприклад, занадто короткий термін реєстрації домену, відсутність індексації у пошукових системах або надмірна кількість перенаправлень. Це дозволяє користувачу самостійно оцінити рівень ризику і прийняти обґрунтоване рішення щодо подальших дій.

Поєднання автономності, адаптивності та пояснюваності результатів робить запропоновану модель більш сучасною, ефективною та користувацько орієнтованою у порівнянні з традиційними рішеннями. Таким чином, розроблений метод і засіб виявлення фішингових сайтів не лише підвищує рівень кіберзахисту, а й сприяє формуванню довіри користувачів до технологій штучного інтелекту в системах інформаційної безпеки.

1.4 Аналіз використаного набору даних і характеристика ознак

Для навчання та тестування моделі виявлення фішингових веб-сайтів було використано узагальнений відкритий набір даних Phishing Websites Dataset [14], який широко застосовується у дослідженнях із кібербезпеки. Даний набір є репрезентативним, оскільки охоплює різноманітні приклади як легітимних, так і фішингових веб-ресурсів, що дає змогу якісно оцінювати роботу алгоритмів машинного навчання. Загалом набір містить 11 430 записів, кожен з яких представляє окремий сайт, і 89 ознак, що описують його структуру, поведінку та технічні параметри.

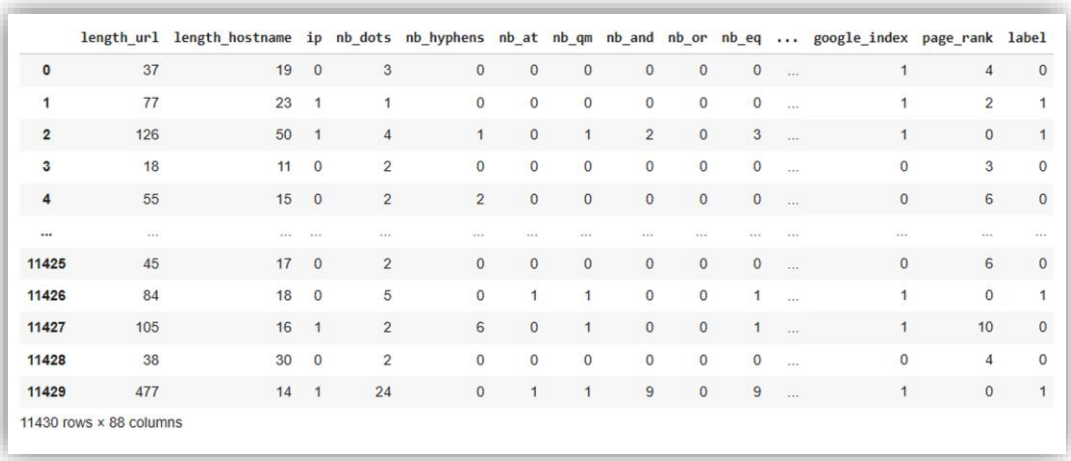
Ознаки у наборі даних охоплюють різні аспекти побудови та функціонування веб-сайтів, що дозволяє моделі виявляти приховані закономірності, характерні для фішингових ресурсів. Кожен запис у наборі даних

описує окремий веб-сайт і складається з числових та логічних полів, що характеризують різні властивості ресурсу. Значення полів можуть відображати як структуру посилання, так і технічні параметри домену, контент сторінки або показники її безпеки. Частина ознак є бінарними (наприклад, наявність HTTPS-протоколу чи DNS-запису), інші числовими (довжина URL, кількість гіперпосилань, рівень трафіку). Останній стовпець таблиці – label є цільовою змінною, що визначає клас об’єкта: значення 0 відповідає легітимному сайту, а 1 фішинговому [15].

Характеристики можна поділити на основні групи:

- структурні характеристики URL (довжина посилання, кількість крапок, дефісів, символів “@”, “&”, “=”, “?” тощо);
- доменні параметри (вік домену, тривалість реєстрації, наявність DNS-запису, використання HTTPS-протоколу, рейтинг PageRank);
- контентні ознаки (кількість гіперпосилань, наявність елементів <iframe>, favicon, форм авторизації, підозрілих JavaScript-скриптів);
- поведінкові індикатори (кількість перенаправлень, співвідношення внутрішніх і зовнішніх посилань, використання подій onmouseover чи onclick);
- параметри безпеки та індексації (чи індексується сайт у Google, наявність SSL-сертифіката, підозрілі доменні зони).

Цільова змінна (label) має два можливих значення це 0 для легітимних сайтів і 1 для фішингових. Для наочного розуміння структури використаного набору даних на рисунку 1.1 наведено фрагмент таблиці з реальними записами датасету, який застосовувався для навчання та тестування моделі.



	length_url	length_hostname	ip	nb_dots	nb_hyphens	nb_at	nb_qm	nb_and	nb_or	nb_eq	...	google_index	page_rank	label
0	37	19	0	3	0	0	0	0	0	0	...	1	4	0
1	77	23	1	1	0	0	0	0	0	0	...	1	2	1
2	126	50	1	4	1	0	1	2	0	3	...	1	0	1
3	18	11	0	2	0	0	0	0	0	0	...	0	3	0
4	55	15	0	2	2	0	0	0	0	0	...	0	6	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
11425	45	17	0	2	0	0	0	0	0	0	...	0	6	0
11426	84	18	0	5	0	1	1	0	0	1	...	1	0	1
11427	105	16	1	2	6	0	1	0	0	1	...	1	10	0
11428	38	30	0	2	0	0	0	0	0	0	...	0	4	0
11429	477	14	1	24	0	1	1	9	0	9	...	1	0	1

11430 rows × 88 columns

Рисунок 1.1 – Фрагмент набору даних Phishing Websites Dataset

Кожен рядок таблиці відповідає окремій веб-сторінці та містить числове представлення її характеристик. Наприклад, ознака length_url відображає загальну довжину URL-адреси, length_hostname: довжину доменного імені, nb_dots та nb_hyphens: кількість крапок і дефісів у посиланні, що є типовими індикаторами підозрілих URL, оскільки фішингові сайти часто використовують складні та перевантажені адреси. Ознаки nb_at, nb_qm, nb_and, nb_eq характеризують наявність спеціальних символів у URL, які можуть використовуватися для маскування справжнього домену або передачі прихованих параметрів. Також у наборі присутні ознаки більш високого рівня, зокрема google_index, що вказує на індексацію сайту пошуковою системою Google, та page_rank, який відображає авторитетність ресурсу в мережі. Останній стовпець label є цільовою змінною: значення 0 відповідає легітимному сайту, а 1 – фішинговому. Саме в такому вигляді дані надходять до моделі машинного навчання: кожен сайт перетворюється на вектор із десятків числових ознак, після чого алгоритм Random Forest аналізує їх сукупність та приймає рішення щодо безпечності ресурсу. Такий підхід дозволяє автоматизувати процес виявлення

фішингових сайтів і виключити суб’єктивний фактор, забезпечуючи стабільну та відтворювану класифікацію веб-ресурсів.

Розподіл класів у наборі даних є абсолютно збалансованим по 5 715 прикладів у кожній групі, що дозволяє уникнути переваги одного класу над іншим і забезпечує стабільність моделі під час навчання. На рисунку 1.2 подано графічне відображення цього балансу, де зеленим кольором позначено легітимні сайти, а червоним фішингові.

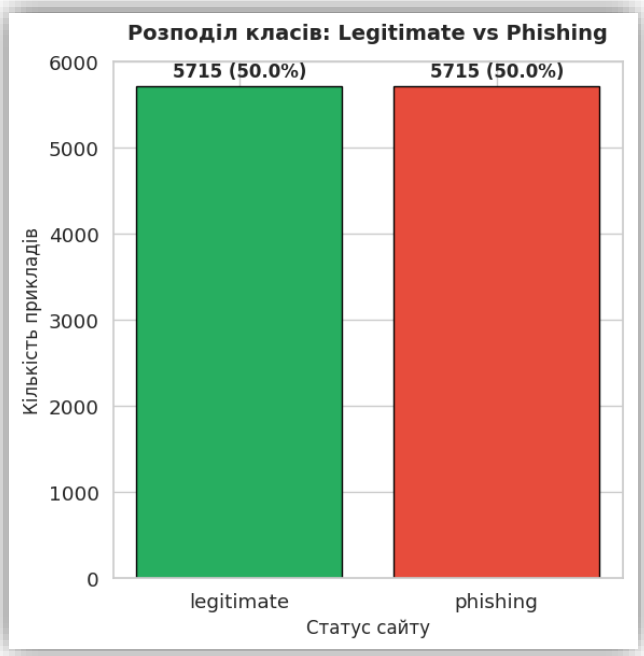


Рисунок 1.2 – Розподіл класів “Legitimate” vs “Phishing”

Під час попереднього аналізу було виконано перевірку даних на наявність пропусків, дублювань і аномальних значень. Усі ознаки виявилися коректними, а пропущені значення відсутні, що дозволило одразу перейти до етапу підготовки даних для моделі. Також було проведено аналіз мультиколінеарності ознак за допомогою кореляційної матриці, результати якої наведено на рисунку 1.3.

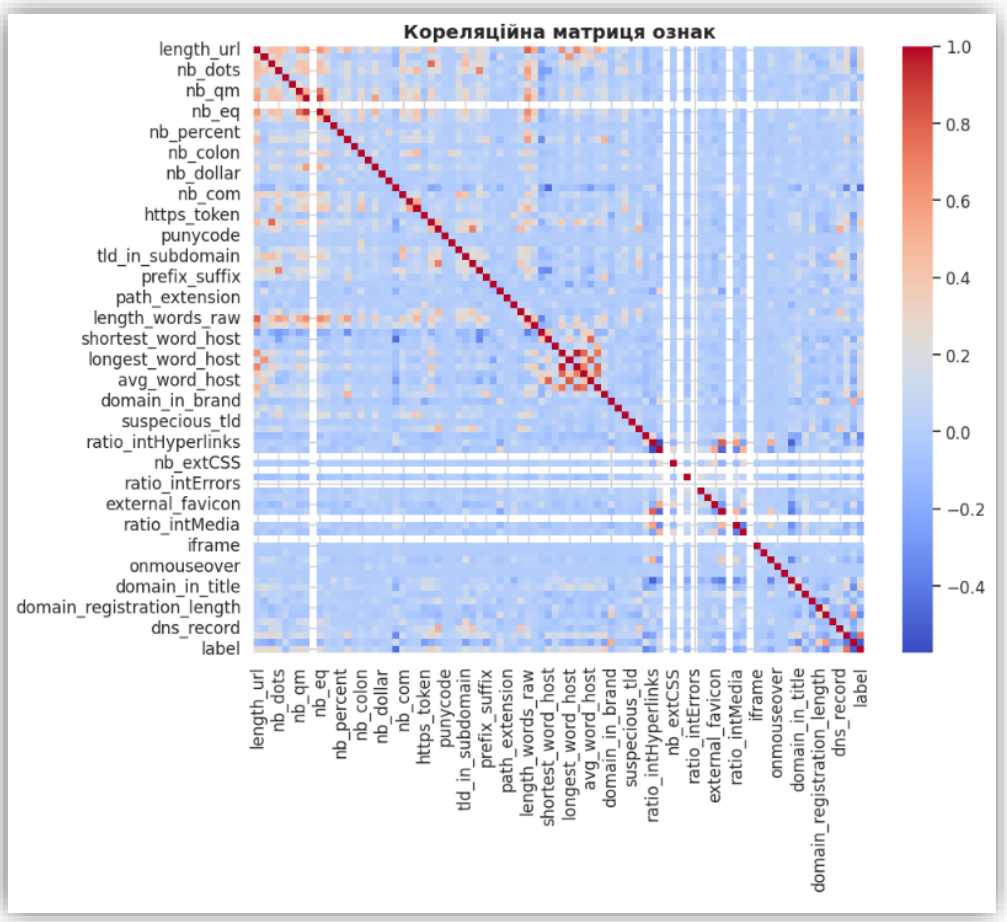


Рисунок 1.3 – Кореляційна матриця ознак

З отриманої матриці видно, що більшість ознак мають низьку або помірну кореляцію між собою (значення коефіцієнта Пірсона менше 0.4), що свідчить про відсутність суттєвої надмірності у даних. Найвищий рівень взаємозв'язку спостерігається між характеристиками, які описують схожі аспекти структури домену, наприклад між domain_registration_length, domain_age та dns_record, або між length_url, nb_dots і nb_hyphens. Такий розподіл є типовим для веб-аналітичних даних і не створює загрози для якості моделі. Водночас, відсутність сильно скорельованих груп ознак дозволяє алгоритмам машинного навчання, зокрема ансамблевим моделям типу Random Forest, ефективно формувати рішення без ризику перенавчання [16].

Вибір саме цього набору даних обґрунтовується кількома чинниками. По-перше, він містить великий обсяг спостережень, що забезпечує статистичну достовірність результатів. По-друге, дані охоплюють як технічні, так і поведінкові аспекти фішингу, що дозволяє побудувати універсальну модель із високим рівнем узагальнення. По-третє, датасет є публічним і добре задокументованим, тому може бути використаний для повторного навчання, тестування або порівняльних досліджень іншими фахівцями. Крім того, збалансованість класів спрощує аналіз ефективності різних алгоритмів і дозволяє зосередитися на якості самої моделі, а не на компенсації перекосів у вибірці.

1.5 Висновки до розділу 1

У першому розділі було проведено комплексний аналіз проблеми фішингових атак як однієї з ключових загроз інформаційній безпеці у сучасному цифровому середовищі. Було здійснено класифікацію основних типів фішингових атак, серед яких найпоширенішими є email-фішинг, web-фішинг, smishing (SMS), vishing (телефонні дзвінки) та соціальний фішинг, що реалізується через соціальні мережі. Описано типові техніки впливу на користувача від психологічного тиску й імітації довіри до створення підроблених доменів і використання фальшивих сертифікатів. У результаті аналізу підтверджено, що головним чинником успішності таких атак залишається людський фактор, а отже, найефективніший захист потребує автоматизації процесу виявлення потенційно шкідливих ресурсів.

Окрему увагу приділено існуючим методам виявлення фішингових сайтів. Показано, що традиційні підходи на основі чорних списків, сигнатурного чи евристичного аналізу мають низьку адаптивність і не здатні вчасно реагувати на появу нових загроз. Водночас інтелектуальні методи на основі машинного навчання демонструють високу точність, гнучкість і можливість самонавчання на

основі великої кількості даних. Алгоритми типу Decision Tree, Support Vector Machine, Neural Networks та Random Forest дозволяють виявляти приховані закономірності між технічними параметрами сайтів і типом ресурсу.

У межах дослідження обґрунтовано доцільність застосування саме ансамблевого алгоритму Random Forest, який забезпечує стабільну роботу з великою кількістю ознак, стійкість до шумів у даних і високу узагальнювальну здатність. На відміну від централізованих, запропонована модель є локальною, тобто не потребує постійного підключення до зовнішніх серверів і може працювати автономно у браузері користувача. Такий підхід підвищує швидкість реагування, зменшує ризик витоку даних і робить систему придатною для застосування у корпоративному або обмеженому середовищі.

Важливою перевагою розробленої моделі є її адаптивність та інформативність. Вона не лише класифікує сайт як безпечний або фішинговий, а й пояснює користувачу причину свого рішення, тобто демонструє, які саме ознаки (наприклад, новий домен, відсутність HTTPS чи низький рейтинг сторінки) викликали підозру. У підрозділі 1.4 детально розглянуто структуру використаного набору даних Phishing Websites Dataset, що містить 11 430 записів і 89 ознак. Проведено аналіз їхнього змісту, типів і статистичних характеристик. Визначено, що дані є збалансованими за класами, не містять пропусків або дублювань і охоплюють широкий спектр технічних, структурних та поведінкових характеристик веб-сайтів.

2 РОЗРОБКА МЕТОДУ ТА НАВЧАННЯ МОДЕЛІ

2.1 Вибір алгоритмів та оцінювання ефективності моделей

Для вирішення задачі класифікації веб-сайтів на фішингові та легітимні було проведено серію експериментів із навчання та порівняння різних алгоритмів машинного навчання. Основна мета цього етапу цевизначити модель, яка забезпечує найвищу точність, стабільність і швидкість роботи під час перевірки сайтів у реальному часі [17].

Перед початком навчання дані було стандартизовано за допомогою методу StandardScaler, що дозволило вирівняти шкали числових ознак. Вибірка була розділена у співвідношенні 80/20, де 80% даних використано для навчання моделі, а 20% для тестування.

Для оцінки якості класифікації використано такі основні показники:

Accuracy (Точність класифікації) [18]:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1.1)$$

де *TP(True Positives)* – кількість правильних позитивних передбачень, TN(True Negatives) – кількість правильних негативних передбачень, FP(False Positives) – кількість помилкових позитивних передбачень, FN(False Negatives) – кількість помилкових негативних передбачень.

Precision (Точність передбачення) [19]:

$$Precision = \frac{TP}{TP+FP} \quad (1.2)$$

Відображає, який відсоток сайтів, позначених як фішингові, справді є шкідливими. Ця метрика важлива, коли помилкове спрацювання може призвести до втрати довіри користувачів.

Recall (Повнота) [20]:

$$Recall = \frac{TP}{TP+FN} \quad (1.3)$$

Визначає, яка частка реальних фішингових сайтів була виявлена системою. Високий показник *Recall* означає, що модель не пропускає шкідливі ресурси.

F1-score (Збалансована метрика) [21]:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (1.4)$$

Цей показник об'єднує *Precision і Recall*, відображаючи баланс між точністю та повнотою. Він особливо корисний у задачах, де обидва типи помилок *(FP і FN)* є критичними.

Для подальшого етапу навчання моделі було проведено порівняльне дослідження різних алгоритмів машинного навчання, щоб визначити, який із них найкраще підходить для класифікації веб-сайтів на фішингові та легітимні в межах обраного набору даних. Такий аналіз є важливим, оскільки кожен алгоритм має власні особливості одні краще працюють із лінійними залежностями, інші ефективніші у випадках, коли між ознаками існують складні нелінійні зв'язки. Крім того, показники якості моделей можуть суттєво змінюватися залежно від характеристик даних, таких як кількість ознак, збалансованість класів чи рівень кореляції між ними. Для цього було обрано п'ять базових алгоритмів класифікації, що добре зарекомендували себе у задачах аналізу веб-контенту та кібербезпеки:

- Random Forest (RF), це ансамблевий метод, який поєднує велику кількість дерев рішень і формує колективне рішення шляхом голосування. Він відзначається високою точністю, стійкістю до шумів і здатністю працювати з різнорідними даними без попереднього масштабування [22];
- Gradient Boosting (GB) це покроковий метод, який послідовно навчає слабкі моделі (дерева рішень), покращуючи результати попередніх. Його перевага

полягає у здатності моделювати складні залежності, проте час навчання зазвичай більший;

- Support Vector Machine (SVM, ядро RBF) це метод, що знаходить оптимальну гіперплощину для розділення класів. Добре підходить для задач із чіткими межами між класами, але потребує обчислювальних ресурсів при великих наборах ознак;
- Logistic Regression (LR) це лінійна модель, яка проста в реалізації та інтерпретації, але має обмежену ефективність при складних, нелінійних закономірностях;
- MLP Neural Network (багатошаровий персептрон) це штучна нейронна мережа, здатна відтворювати складні зв’язки між параметрами сайту. Потребує більше часу на навчання, але забезпечує високу гнучкість.

Додатково було створено ансамблеву модель Voting Ensemble, яка поєднує результати кількох окремих алгоритмів (Random Forest, Gradient Boosting, Logistic Regression) шляхом “м’якого голосування” обчислення середнього значення ймовірностей кожної моделі. Такий підхід дозволяє компенсувати недоліки окремих класифікаторів і підвищити стабільність кінцевого рішення. Порівняння цих алгоритмів дозволило об’єктивно оцінити, який із них найкраще працює на даному наборі даних, забезпечуючи баланс між точністю класифікації, повнотою виявлення фішингових сайтів, стійкістю до помилок та швидкістю передбачення критичними параметрами для системи, що працює в реальному часі.

У таблиці таблиці 2.1 представленно, що всі моделі показали високі результати: середня точність перевищує 93%, що свідчить про якісний вибір ознак і добру збалансованість даних. Проте детальніше порівняння дозволяє виділити найефективнішу модель: Random Forest яка показує найвищу точність (96.1%) та найкращий F1-score (0.9612) серед усіх моделей. Це означає, що

алгоритм найкраще збалансовує між виявленням більшості фішингових сайтів (Recall = 0.9650) та мінімізацією хибних спрацювань (Precision = 0.9575). Його час навчання (9.02 с) є помірним, а швидкість передбачення (0.1 мс/запис) дозволяє використовувати модель у браузерному плагіні без затримок [23].

Таблиця 2.1 – Порівняльна ефективність моделей класифікації фішингових сайтів

Модель **Показник**	**Random Forest**	**MLP Neural Net**	**Gradient Boosting**	**Voting Ensemble**	**SVM (RBF)**	**Logistic Regression**
Accuracy	0.9611	0.9558	0.9536	0.9536	0.9528	0.9366
F1-score	0.9612	0.9559	0.9538	0.9539	0.9526	0.9364
Recall	0.9650	0.9571	0.9580	0.9589	0.9501	0.9335
Precision	0.9575	0.9546	0.9497	0.9489	0.9551	0.9393
Час навчання (с)	9.02	9.29	14.34	9.14	11.37	0.30
Час передбачення (мс/запис)	0.1026	0.0044	0.0035	0.0640	0.1944	0.0004

MLP Neural Net і Gradient Boosting також показали високу точність (≈95%), однак вимагають довшого часу навчання і гірше узагальнюють дані при великій кількості фіч. Voting Ensemble демонструє стабільні результати, але не перевищує показники базових моделей, що свідчить про відсутність суттєвого ефекту від комбінування. SVM (RBF) працює добре, але поступається у швидкості та масштабованості при великому наборі ознак потребує більше часу на передбачення. Logistic Regression є найпростішою і найшвидшою моделлю (0.0004 мс/запис), але її точність (93.6%) значно нижча, тому вона не підходить для високоточного автоматичного виявлення загроз.

Результати підтверджують, що алгоритм Random Forest є найкращим вибором для задачі виявлення фішингових сайтів. Його ключові переваги:

– висока точність класифікації навіть без складної попередньої обробки ознак;
– стійкість до шуму та мультиколінеарності;
– можливість паралельного обчислення (що скорочує час навчання);
– прозора інтерпретація, можна визначити важливість кожної ознаки у процесі прийняття рішення.

Таким чином, подальше навчання, оптимізація параметрів і побудова системи виявлення фішингових сайтів здійснюватимуться на основі моделі Random Forest.

На рисунку 2.1 подано графічне порівняння продуктивності моделей за чотирма основними метриками: Accuracy, F1-score, Recall та Precision. Для кожної моделі візуально відображено середні значення цих показників у вигляді горизонтальних стовпчиків.

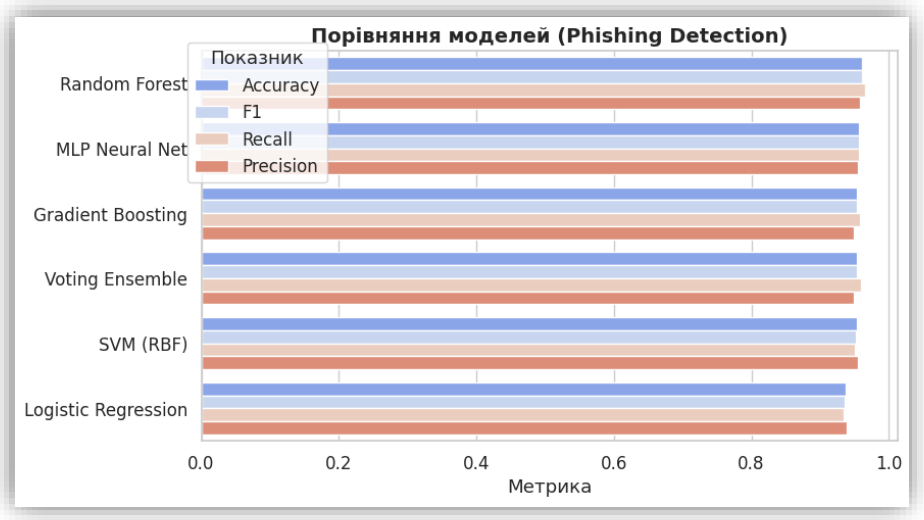


Рисунок 2.1 – Порівняльний аналіз ефективності моделей

З графіка чітко видно, що Random Forest займає провідну позицію за всіма показниками та його стовпчики сягають максимальних значень. Такий візуальний аналіз дає змогу швидко оцінити співвідношення між якістю

класифікації та стабільністю роботи кожної моделі, а також підтверджує доцільність вибору Random Forest як базового алгоритму для подальшої розробки програмного засобу “Phishing Detector”.

Для більш глибокої перевірки ефективності навчання було проведено аналіз моделей за допомогою ROC-кривих (Receiver Operating Characteristic) інструменту, який дозволяє оцінити здатність моделі відрізняти фішингові сайти від легітимних на різних порогах класифікації. На відміну від простої точності, ROC-крива враховує співвідношення між True Positive Rate (чутливістю) та False Positive Rate (1 – специфічністю), що дає змогу оцінити баланс між виявленням загроз і кількістю хибних спрацювань. Під час експерименту для кожної моделі було побудовано ROC-криву та обчислено показник AUC (Area Under Curve) площу під кривою, яка є інтегральним показником якості класифікації. Чим ближче значення AUC до 1, тим краще модель розрізняє два класи: фішингові та безпечні сайти.

На рисунку 2.2 подано графічне порівняння ROC-кривих для п’яти основних моделей машинного навчання. Результати обчислень показали, що Random Forest знову демонструє найвищу ефективність із AUC = 0.993, що практично наближається до ідеального розділення класів. Дуже близький результат показав Gradient Boosting (AUC = 0.990), що підтверджує його високу чутливість до прихованих патернів у даних. Значно нижчі показники спостерігаються у Logistic Regression (AUC = 0.886) та MLP Neural Net (AUC = 0.863), які хоч і правильно класифікують більшість прикладів, проте іноді дають більшу кількість хибних спрацювань. Найгірший результат показав SVM (RBF) AUC = 0.773, що вказує на слабку здатність моделі до узагальнення в умовах складних, багатовимірних даних. Таким чином, аналіз ROC-кривих підтвердив попередні висновки щодо домінування Random Forest, який забезпечує найкращий баланс між чутливістю та специфічністю. Це означає, що модель

може ефективно виявляти фішингові сайти з мінімальною кількістю помилкових спрацьовувань, що є критично важливим для систем реального часу, інтегрованих у браузер або антивірусне середовище.

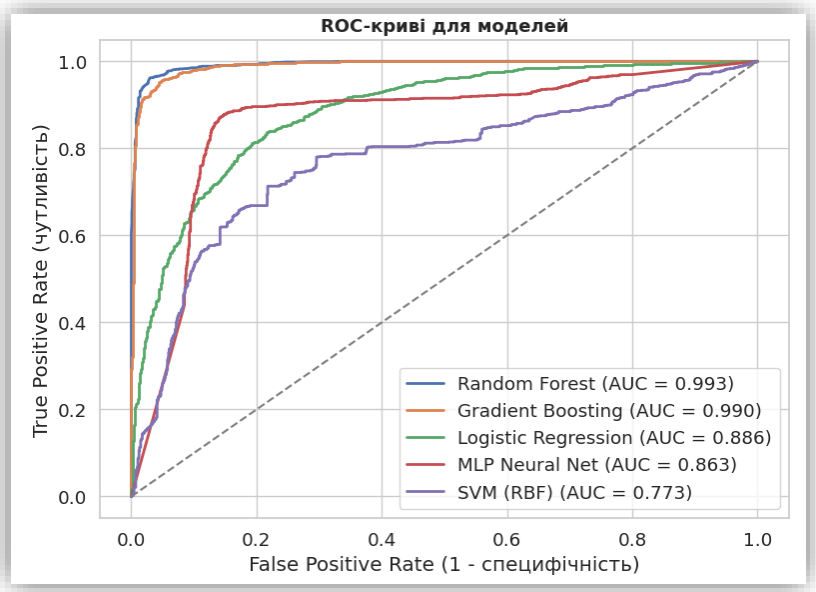


Рисунок 2.2 – ROC-криві для моделей машинного навчання

З аналізу видно, що навіть незначна різниця між показниками Random Forest і Gradient Boosting свідчить про їхню високу узгодженість, проте саме Random Forest краще поєднує точність, швидкість і стабільність роботи. Це ще раз підтверджує його доцільність як основної моделі для побудови інтелектуальної системи виявлення фішингових веб-ресурсів.

2.2 Формування ознак і визначення їх оптимальної кількості

Після того як у попередньому розділі було визначено найефективнішу модель для виявлення фішингових сайтів Random Forest, наступним етапом стало дослідження впливу кількості ознак (features) на якість класифікації. Мета цього аналізу знайти оптимальний баланс між кількістю характеристик, що описують сайт, і точністю передбачення, тобто виявити мінімальний набір параметрів, який

забезпечує найкращий результат при мінімальних обчислювальних витратах. Занадто мала кількість ознак може призвести до втрати важливої інформації, тоді як надлишкова до збільшення часу обробки та ризику перенавчання моделі. Для цього було сформовано кілька піднаборів даних, які включали від 10 до 87 найважливіших ознак. Визначення їхньої важливості здійснювалося за допомогою вбудованого механізму feature importance у моделі Random Forest, який оцінює внесок кожної ознаки у зменшення невизначеності класифікації. Таким чином, на першому етапі ми не лише використовуємо Random Forest як основний алгоритм, але й як інструмент для аналітичного відбору найрелевантніших характеристик.

Ознаки, що використовуються в моделі, можна умовно поділити на три групи: URL-based, доменні та контентні. Перша група охоплює параметри, пов’язані зі структурою веб-адреси: довжиною URL, кількістю крапок, дефісів, символів “@”, “=”, “?” або “&”, наявністю IP-адреси чи підозрілих ключових слів у посиланні. Друга група містить доменні ознаки, які відображають технічні характеристики домену, такі як його вік, тривалість реєстрації, наявність DNS-запису або SSL-сертифіката, використання HTTPS-протоколу, показник індексації в Google та рейтинг PageRank. Третя група: контентні ознаки, які характеризують структуру веб-сторінки, кількість внутрішніх і зовнішніх гіперпосилань, наявність елементів <iframe>, скриптів JavaScript, favicon або перенаправлень. Саме поєднання цих трьох категорій дає змогу комплексно оцінити поведінку сайту й визначити його ймовірну фішингову природу [24].

Процес формування ознак відбувався автоматично, шляхом аналізу структури кожного сайту. Для кожного запису обчислювались як кількісні, так і логічні параметри, наприклад кількість символів, наявність HTTPS, чи сайт індексується в пошукових системах тощо. Це дозволило побудувати єдиний вектор характеристик, який згодом використовувався для навчання моделі. Щоб

визначити оптимальну кількість ознак, було проведено серію експериментів, результати яких візуалізовано на графіках. На рисунку 1.1 продемонтровано залежність Accuracy і F1-score від кількості ознак. Як видно, точність моделі поступово зростає до моменту використання приблизно 50 ознак, після чого стабілізується, а подальше додавання характеристик не дає суттєвого приросту якості. Це свідчить про те, що понад половина ознак має мінімальний вплив на результат, а їх вилучення не погіршує ефективності класифікації.

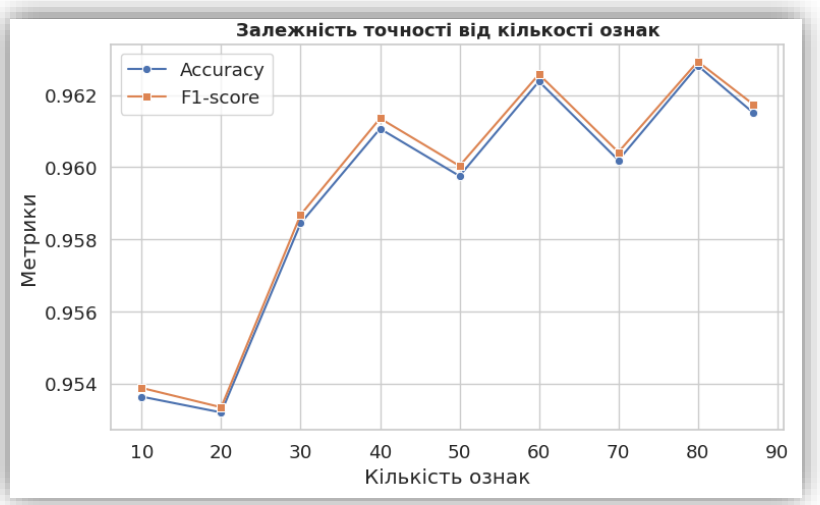


Рисунок 2.3 – Вплив кількості ознак на точність моделі

Другий графік показує поведінку метрик Recall і Precision залежно від кількості характеристик (рис. 2.4). Тут також простежується закономірність: збільшення кількості ознак до 40–60 дозволяє досягти високої чутливості моделі (Recall ≈ 0.968), тобто здатності правильно виявляти фішингові сайти, тоді як точність (Precision ≈ 0.955) залишається стабільною. Це означає, що модель добре розпізнає більшість фішингових сторінок, не збільшуючи кількість хибних позитивних спрацьовувань.

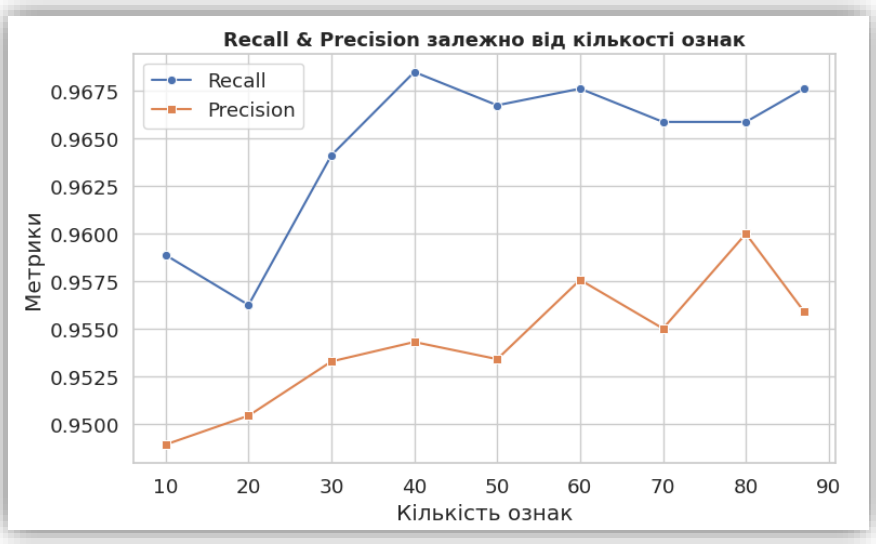


Рисунок 2.4 – Recall і Precision залежно від кількості ознак

Третій графік ілюструє залежність швидкості передбачення (Inference time) від кількості ознак (рис. 2.5). Тут видно, що час передбачення на один запис суттєво зростає при збільшенні кількості характеристик понад 50, що особливо важливо для системи, яка працює в реальному часі. Надмірна кількість параметрів створює додаткове навантаження на модель і може знижувати її продуктивність при інтеграції у браузер або антивірусний плагін.

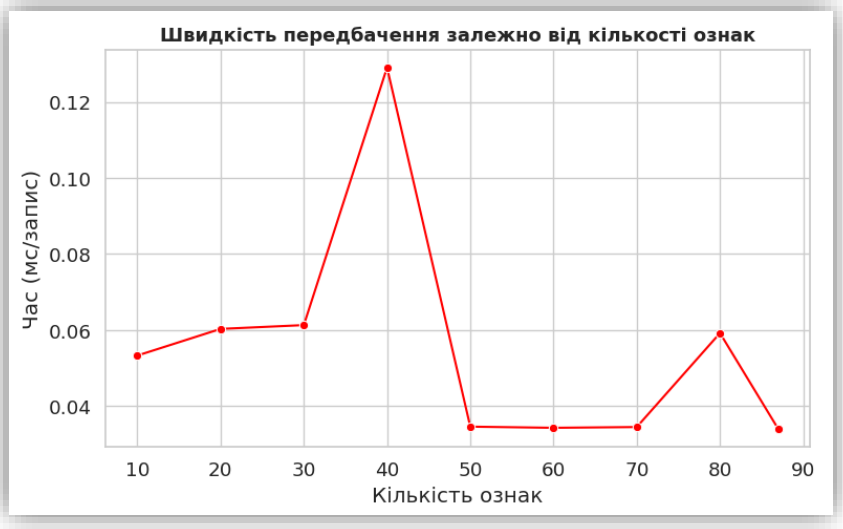


Рисунок 2.5 – Графік залежність швидкості передбачення від кількості ознак

На фінальному (рис. 2.6) наведено узагальнення результатів усіх проведених експериментів із визначення оптимальної кількості ознак для моделі Random Forest. На осі X відображено кількість використаних характеристик, тоді як на осі Y показники якості моделі (Accuracy і F1-score) та швидкість передбачення (інференс, у мілісекундах на один запис). Червона пунктирна лінія позначає точку оптимуму і дорівнює 50 ознак, що виявилися найкращим компромісом між точністю класифікації, стабільністю результатів і продуктивністю системи. Із графіка видно, що зі збільшенням кількості характеристик точність моделі зростає до певної межі (приблизно 50 ознак), після чого спостерігається насичення, додаткові параметри не покращують ефективність, а лише ускладнюють обчислення. Максимальні значення Accuracy ≈ 0.961 і F1-score ≈ 0.962 досягаються саме при використанні 50 найінформативніших ознак. У той же час середній час передбачення становить лише ≈ 0.04 мс на запис, що свідчить про високу швидкодію алгоритму навіть за умови обробки великих масивів даних у реальному часі.

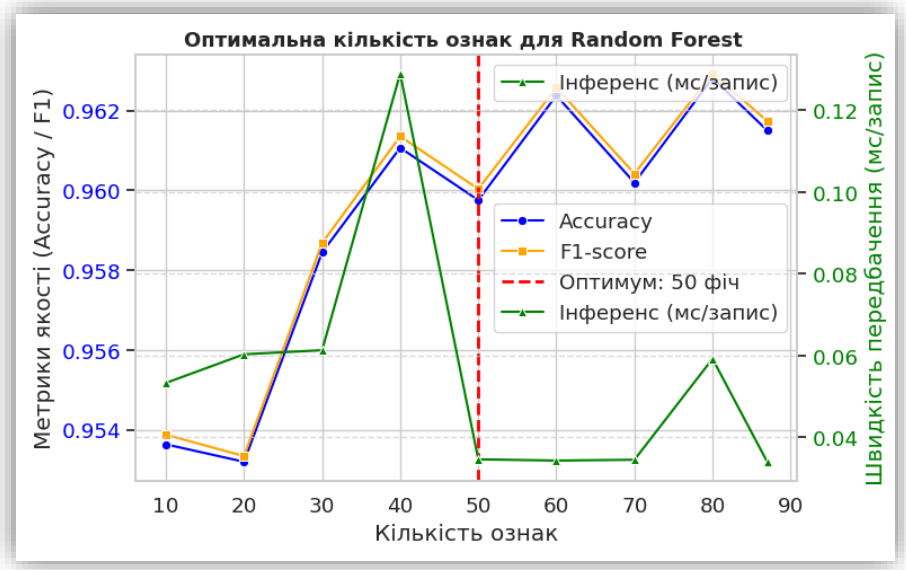


Рисунок 2.6 – Графік залежності показників ефективності моделі Random Forest від кількості ознак

Такий результат особливо важливий у контексті подальшої інтеграції моделі в браузерний плагін “Phishing Detector”. Висока точність гарантує мінімальну кількість хибних спрацьовувань, тоді як мала затримка при передбаченні дозволяє проводити миттєвий аналіз URL-адрес без помітного впливу на швидкість завантаження сторінок. Для підкріплення графічних висновків було сформовано таблицю 2.2, яка узагальнює числові показники точності (Accuracy), узгодженості класифікації (F1-score), чутливості (Recall) і точності позитивних передбачень (Precision) для різних обсягів ознак. Як видно з результатів, приріст якості спостерігається лише до певного моменту, приблизно 50 ознаків. Саме в цій точці досягається оптимальне співвідношення між точністю класифікації (Accuracy = 0.9611, F1 = 0.9620) та швидкістю роботи моделі (≈ 0.04 мс на один запис).

Таблиця 2.2 – Результати тестування моделі Random Forest при різній кількості ознак

Кількість ознак	**Accuracy**	**F1-score**	**Recall**	**Precision**	**Інференс (мс/запис)**
10	0.9539	0.9542	0.9590	0.9490	0.055
30	0.9587	0.9589	0.9642	0.9520	0.063
40	0.9614	0.9617	0.9680	0.9545	0.130
50	0.9611	0.9620	0.9673	0.9552	0.040
70	0.9601	0.9605	0.9661	0.9550	0.037
80	0.9624	0.9628	0.9668	0.9585	0.060

Подальше збільшення кількості ознак (до 70–80) не призводить до помітного покращення результатів, а іноді навіть знижує стабільність метрик через надлишкову інформацію, яка не має вагомого впливу на процес ухвалення рішень. Також можна помітити, що інференс-час змінюється коливально при

зростанні кількості ознак понад 50 модель втрачає частину швидкодії, що підтверджує важливість пошуку компромісу між якістю та ефективністю.

Отже, отримані результати підтверджують висновки, зроблені за графіками (рис. 2.3 – 2.6): оптимальним набором ознак для моделі Random Forest є 50 фіч, що дозволяє досягти максимальної точності без перевантаження системи. Такий обсяг характеристик забезпечує стабільну роботу класифікатора у реальному часі та є найраціональнішим варіантом для подальшої інтеграції моделі у браузерний плагін виявлення фішингових сайтів.

2.3 Відбір ознак для навчання моделі та їх опис

Після попереднього аналізу набору даних і визначення оптимальної кількості ознак для побудови моделі було здійснено детальний відбір найбільш інформативних характеристик, які забезпечують найвищу якість класифікації при мінімальних обчислювальних витратах. Для цього було проведено кількісну оцінку важливості ознак (Feature Importance) із використанням алгоритму Random Forest, який не лише виконує класифікацію, а й дозволяє обчислити внесок кожної ознаки у зниження невизначеності під час побудови дерев рішень.

Модель Random Forest складається з великої кількості дерев рішень, кожне з яких формує власні правила розподілу об’єктів на класи. При кожному розбитті вузла дерева обчислюється, наскільки певна ознака покращує однорідність вибірки (зменшує “невизначеність”). Цей ефект кількісно вимірюється через показник середнього зменшення невизначеності (Mean Decrease in Impurity, MDI).

Математично вагомість ознаки j визначається формулою [25]:

$$Importance(f) = \frac{1}{N}\sum_{t \in T(f)} N(t)\, \Delta\varphi(t), \qquad (2.1)$$

де: $T(f)$ – множина всіх вузлів дерева, у яких ознака f використовувалася для розбиття, $N(t)$ – кількість об’єктів навчальної вибірки, що проходять через вузол t, $\Delta\varphi(t)$ – зміна функції невизначеності (наприклад, критерію Джині або ентропії) після розбиття у вузлі t, N – загальна кількість об’єктів у навчальній вибірці. Таким чином, чим більше певна ознака зменшує невизначеність під час класифікації, тим вищою є її важливість для моделі.

На основі цієї методики було проведено обчислення важливості всіх ознак у наборі даних. Після сортування за спаданням показника *Importance(f)* сформовано ранжований список топ-50 найбільш інформативних характеристик, які мають найбільший вплив на точність класифікації. На рисунку 2.8 наведено графічне відображення важливості топ-50 ознак, отриманих за допомогою моделі Random Forest.

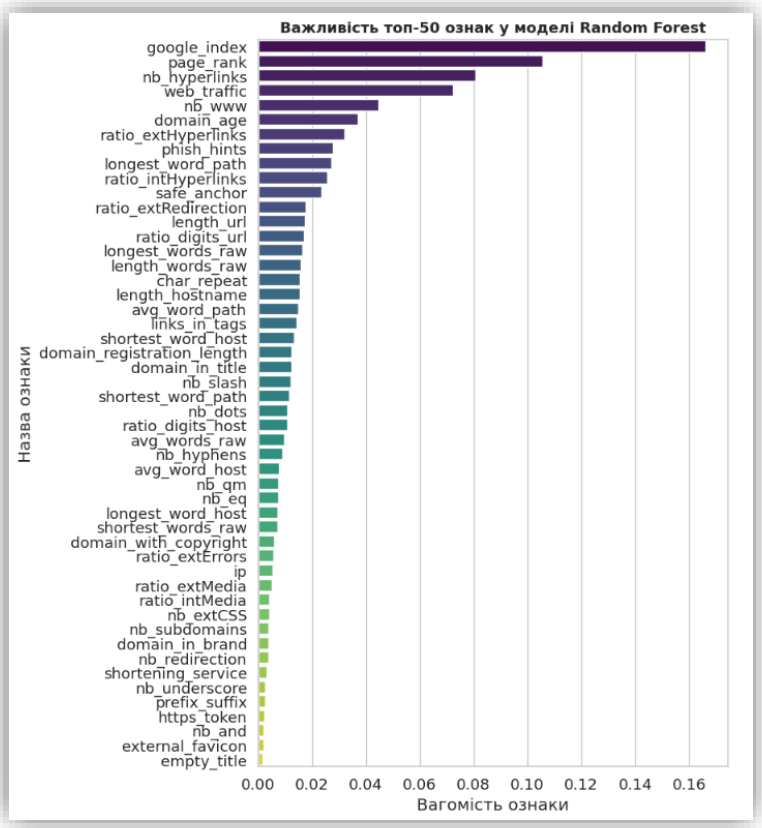


Рисунок 2.7 – Графік важливості топ-50 ознак у моделі Random Forest

Після аналізу важливості ознак та побудови графіка (рис. 2.8) було сформовано остаточний список 50 найбільш інформативних характеристик, які використовуються у подальшому навчанні моделі. Ці ознаки охоплюють як структурні параметри URL, так і доменні та контентні особливості сторінки, створюючи багатовимірний профіль кожного сайту. Загалом, набір відібраних характеристик можна поділити на три логічні групи:

- URL-based ознаки – описують структуру адреси сайту: її довжину, кількість спеціальних символів, глибину шляху, повторюваність елементів тощо. Вони дозволяють виявляти підозрілі шаблони у посиланнях, що часто зустрічаються у фішингових сторінках. Наприклад, length_url, char_repeat, nb_slash, ratio_digits_url, avg_word_path;
- доменні ознаки – характеризують репутацію та технічні параметри домену: вік реєстрації (domain_age), тривалість доменного періоду (domain_registration_length), наявність індексації у Google (google_index), рейтинг сайту (page_rank) або трафік (web_traffic). Ці ознаки найкраще відображають довіру до ресурсу;
- контентні та структурні ознаки – описують HTML-вміст, кількість зовнішніх та внутрішніх гіперпосилань (ratio_extHyperlinks, ratio_intHyperlinks), наявність елементів JavaScript (phish_hints), favicon (external_favicon) або заголовка сторінки (empty_title). Такі фактори допомагають виявити автоматично згенеровані чи неповні сторінки, притаманні фішинговим кампаніям.

З графіка видно, що найбільшу вагомість мають такі параметри, як google_index (0.166), page_rank (0.106), nb_hyperlinks (0.081) та web_traffic (0.072). Вони демонструють, що показники пошукової репутації та активності домену є ключовими маркерами легітимності сайту. У свою чергу, характеристики domain_age, ratio_extHyperlinks, safe_anchor, length_url та

links_in_tags доповнюють загальну оцінку, формуючи другий рівень впливу з вагомістю від 0.03 до 0.01.

Після 30-ї позиції важливість ознак поступово зменшується, але вони залишаються корисними для уточнення рішень моделі в неоднозначних випадках. Наприклад, shortening_service, prefix_suffix або https_token допомагають виявляти спроби маскування URL, що часто використовуються у фішингових посиланнях. Таким чином, сформований набір топ-50 характеристик поєднує як зовнішні показники репутації, так і внутрішні структурні особливості сторінки, що забезпечує комплексний підхід до виявлення фішингових сайтів.

Детальний опис усіх відібраних ознак, їхньої вагомості та змістового значення наведено у Таблиці 2.3 «Топ-50 найважливіших ознак моделі Random Forest», яка відображає внесок кожного параметра у процес ухвалення рішення класифікатором.

Таблиця 2.3 – Топ-50 найважливіших ознак моделі Random Forest

№	Ознака	Важливість	Короткий опис
1	**google_index**	0.166	Показує, чи сторінка індексується Google; фішингові сайти зазвичай не індексуються.
2	**page_rank**	0.106	Рейтинг сторінки у пошукових системах; низький ранг свідчить про недовіру до ресурсу.
3	**nb_hyperlinks**	0.081	Загальна кількість гіперпосилань на сторінці; легітимні сайти мають більше внутрішніх посилань.
4	**web_traffic**	0.072	Рівень відвідуваності сайту; фішингові домени зазвичай мають мінімальний трафік.
5	**nb_www**	0.045	Кількість входжень «www» у домені або посиланнях, надлишок часто вказує на підробку.
6	**domain_age**	0.037	Вік домену у днях або місяцях; молоді домени, типовий признак фішингу.
7	**ratio_extHyperlinks**	0.032	Частка зовнішніх посилань від усіх гіперпосилань; велика частка може вказувати на фішингову сторінку.
8	**phish_hints**	0.028	Ознаки фішингу у вихідному коді (форми, скрипти, ключові слова).

Продовження таблиці 2.3

9	**longest_word_path**	0.027	Довжина найдовшого слова у шляху URL; довгі або беззмістовні шляхи часто є підозрілими.
10	**ratio_intHyperlinks**	0.026	Частка внутрішніх посилань; низьке значення, показник шахрайських сайтів.
11	**safe_anchor**	0.023	Частка безпечних (валідних) анкорів; зменшення свідчить про підроблену структуру.
12	**ratio_extRedirection**	0.018	Кількість зовнішніх перенаправлень; фішингові сторінки часто використовують переадресації.
13	**length_url**	0.017	Загальна довжина URL; занадто довгі посилання типові для фішингу.
14	**ratio_digits_url**	0.017	Частка цифр у посиланні; числові символи часто приховують справжній домен.
15	**longest_words_raw**	0.016	Найдовше слово у тексті сторінки; довгі випадкові токени вказують на автогенерацію.
16	**length_words_raw**	0.016	Середня довжина слів на сторінці; у фішингу часто зустрічаються короткі шаблонні тексти.
17	**char_repeat**	0.015	Повторюваність символів у домені чи шляху; подвійні літери або цифри, ознака імітації бренду.
18	**length_hostname**	0.015	Довжина імені хоста; надмірна довжина характерна для підробок.
19	**avg_word_path**	0.015	Середня довжина слів у шляху URL; допомагає відрізняти згенеровані структури.
20	**links_in_tags**	0.014	Кількість гіперпосилань у тегах; перевищення норми вказує на маскування контенту.
21	**shortest_word_host**	0.013	Найкоротше слово в домені; занадто короткі елементи, спроба приховати справжню адресу.
22	**domain_registration_length**	0.013	Тривалість реєстрації домену; короткі терміни властиві тимчасовим сайтам.
23	**domain_in_title**	0.012	Чи збігається домен із назвою сайту; невідповідність, часта фішингова ознака.
24	**nb_slash**	0.012	Кількість «/» у URL; багато рівнів вкладеності, ознака маскування.
25	**shortest_word_path**	0.011	Найкоротше слово у шляху; дає уявлення про структуру URL.
26	**nb_dots**	0.011	Кількість крапок у домені; велика кількість субдоменів часто використовується у фішингу.
27	**ratio_digits_host**	0.011	Частка чисел у домені; числові домени частіше фішингові.
28	**avg_words_raw**	0.010	Середня кількість слів у тексті сторінки; низьке значення свідчить про шаблонність.
29	**nb_hyphens**	0.009	Кількість дефісів у домені; використовується для створення схожих назв (наприклад, pay-pal.com).

Продовження таблиці 2.3

30	**avg_word_host**	0.008	Середня довжина слів у домені; допоміжна структурна ознака.
31	**nb_qm**	0.008	Кількість знаків «?» у URL; наявність параметрів може вказувати на шкідливі запити.
32	**nb_eq**	0.008	Кількість символів «=» у URL; використовується у фішингових форм-запитах.
33	**longest_word_host**	0.007	Найдовше слово в домені; вказує на аномальну складність імені.
34	**shortest_words_raw**	0.007	Короткі слова в тексті сторінки; багато коротких, ознака автозгенерованого контенту.
35	**domain_with_copyright**	0.006	Чи містить сайт копірайт; відсутність, сигнал недовіри.
36	**ratio_extErrors**	0.006	Частка помилкових зовнішніх посилань; фішингові сайти часто мають “биті” лінки.
37	**ip**	0.005	Наявність IP-адреси замість домену; типова риса шкідливих сторінок.
38	**ratio_extMedia**	0.005	Частка зовнішніх медіа-ресурсів (зображень, відео); надлишок аномальний вміст.
39	**ratio_intMedia**	0.004	Частка внутрішніх медіа; відсутність може свідчити про скопійований контент.
40	**nb_extCSS**	0.004	Кількість зовнішніх CSS-файлів; надлишок можливість прихованих елементів.
41	**nb_subdomains**	0.004	Кількість субдоменів; їх багато, часто використовують для маскування.
42	**domain_in_brand**	0.004	Чи містить домен назву бренду; іноді використовується для підробки (наприклад, paypal-secure.com).
43	**nb_redirection**	0.004	Кількість перенаправлень; надлишок спроба заплутати користувача.
44	**shortening_service**	0.003	Використання скорочувачів посилань (bit.ly тощо); приховує справжню адресу.
45	**nb_underscore**	0.003	Кількість підкреслень «_» у URL; часто використовуються у скам-структурах.
46	**prefix_suffix**	0.002	Наявність префіксів/суфіксів у домені; додають схожість до відомих брендів.
47	**https_token**	0.002	Використання слова “https” у тілі домену типова маскувальна техніка.
48	**nb_and**	0.002	Кількість символів «&»; багато параметрів у запиті підозріло.
49	**external_favicon**	0.002	Іконка сайту розміщена зовні домену; часто копіюється з іншого ресурсу.
50	**empty_title**	0.002	Відсутність заголовка сторінки; типовий дефект фішингових шаблонів.

2.4 Навчання фінальної моделі

Після визначення оптимальної кількості ознак проведено навчання фінальної моделі на відібраному наборі даних. Для цього використано збалансований датасет із рівною кількістю фішингових і легітимних зразків. Перед початком моделювання виконано стандартизацію вхідних ознак за допомогою методу StandardScaler, що забезпечило однакові масштаби даних та стабільну роботу алгоритму. Навчання здійснено на основі 80 % даних, тоді як решта 20 % використовувалася для тестування точності моделі. Як основний алгоритм класифікації застосовано Random Forest із 300 деревами рішень. Такий вибір обґрунтовується високою ефективністю ансамблевого підходу для табличних даних, стійкістю до шуму, відсутністю необхідності у складній нормалізації та можливістю отримання інтерпретованих показників важливості ознак. Кожне дерево у складі ансамблю приймає власне рішення, після чого результати агрегуються шляхом усереднення ймовірностей належності сайту до класу «фішинг». Кінцеве рішення визначається за порогом 0.7 – якщо ймовірність перевищує це значення, сторінка класифікується як потенційно фішингова [26].

Усі розрахунки виконано в середовищі Google Colab, де реалізовано автоматизований конвеєр підготовки даних, навчання, оцінювання моделі та збереження отриманих результатів. Структура системи побудована у вигляді послідовних функціональних модулів, кожен із яких відповідає за окремий етап обробки URL-адреси.

На початку роботи система використовує блок утиліт і кешованих запитів, що забезпечує стабільну взаємодію з мережею. Завдяки функціям cached_html(), cached_whois() і cached_dns() здійснюється попереднє отримання HTML-вмісту сторінки, доменної інформації та DNS-записів. Використання кешування (lru_cache) мінімізує час обробки під час повторних перевірок одного ресурсу.

Наступним етапом є екстракція ознак – обчислення приблизно п’ятдесяти характеристик, які описують як структуру самої адреси, так і поведінку сторінки. Кожна функція, наприклад length_url_flag(), domain_age_flag() чи phish_hints_flag(), повертає числове значення, що відображає наявність або відсутність підозрілої властивості. Отримані значення формують єдиний вектор ознак, який подається на вхід моделі у точно визначеному порядку, збереженому у файлі feature_order.joblib.

Після обчислення ознак дані передаються до модуля масштабування, де застосовується об’єкт StandardScaler, збережений у файлі scaler_top50.joblib. Це гарантує, що нові значення обробляються в тому ж числовому діапазоні, що й під час тренування. Підготовлений вектор подається на вхід класифікатора Random Forest, серіалізованого у файлі final_rf_top50.joblib. Модель складається з 300 дерев рішень і прогнозує ймовірність належності сторінки до класу фішингових.

Результат обчислення містить два основні параметри ймовірність фішингу (prob) та класифікаційний результат (label). Якщо ймовірність перевищує порогове значення 0.7, сторінка маркується як потенційно небезпечна. Додатково формується таблиця з усіма ознаками, де значення «1» позначає наявність певного ризикового фактора (наприклад, відсутність індексації у Google, низький рейтинг сторінки або короткий вік домену).

У процесі навчання створено три ключові файли рисунок 2.8, необхідні для подальшої роботи системи:

- final_rf_top50.joblib – серіалізована модель Random Forest, готова до використання при передбаченні;
- scaler_top50.joblib – збережені параметри масштабування для узгодженої обробки нових даних;
- feature_order.joblib – перелік 50 ознак у правильному порядку для вхідного вектору.

```
os.makedirs('/content/models', exist_ok=True)
joblib.dump(rf_final, '/content/models/final_rf.joblib')
joblib.dump(scaler, '/content/models/final_scaler.joblib')
joblib.dump(FEATURE_ORDER, '/content/models/feature_order.joblib')
```

Рисунок 2.8 – Фрагмент програмного коду для збереження навченої моделі

Після навчання модель досягла високих показників якості: Accuracy ≈ 0.961, F1-score ≈ 0.962, що підтверджує її здатність ефективно відрізняти легітимні ресурси від фішингових. Середній час передбачення становить близько 0.04 мс на один запис, що дає змогу використовувати систему в режимі реального часу без затримок для користувача.

Під час роботи система послідовно проходить три основні етапи: збір ознак, попередню обробку та класифікацію. Після введення або перехоплення браузером веб-адреси, URL потрапляє до модуля екстракції, де виконується перевірка структури посилання, визначення доменного імені та аналіз мережевих показників (WHOIS, DNS, HTML-код). Для оптимізації швидкодії використовується кешування результатів запитів. Після цього дані впорядковуються згідно з feature_order, масштабуються і передаються моделі. Отриманий результат повертається у вигляді прогнозу – безпечний або підозрілий сайт із переліком ознак, що вплинули на рішення.

Послідовність роботи системи Phishing Detector:

1) URL введення, отримання адреси користувачем або браузером.
2) Кешовані запити, функції cached_html(), cached_whois(), cached_dns() отримують дані про сторінку, домен і DNS.
3) Екстракція ознак, обчислення близько 50 характеристик (структурних і поведінкових).
4) Масштабування, нормалізація даних за допомогою StandardScaler.
5) Класифікація Random Forest, модель визначає ймовірність фішингу.

6) Формування результату, функція predict_url() повертає показники prob, label та список спрацьованих ознак.
7) Візуалізація, система виводить результат (“Фішинг” або “Безпечний”).

На рис. 2.9 наведено приклад роботи системи при перевірці фішингового ресурсу https://paypal-login-verification.com. Модель оцінила ймовірність фішингу на рівні 76.67 %, що підтверджується спрацьовуванням кількох ключових ознак: низький вік домену, відсутність індексації в Google і відсутність реального трафіку.

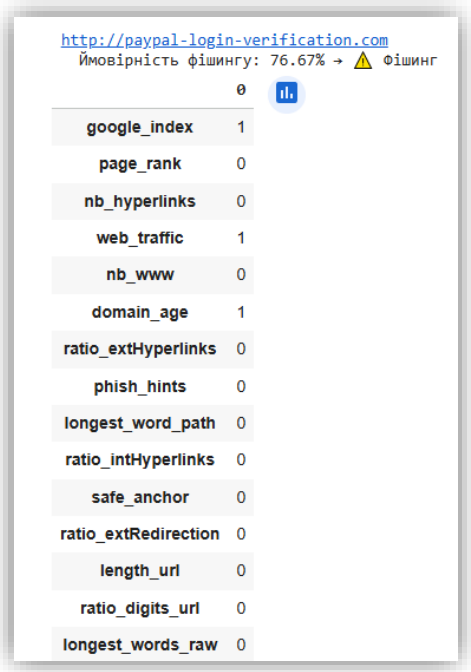


Рисунок 2.9 – Результат перевірки URL-адреси на фішинг

Для перевірки стабільності система протестована на кількох десятках реальних прикладів, серед яких як фішингові домени, так і легітимні ресурси, які не використовували у навчанні таблиця 2.4. У більшості випадків модель демонструвала правильну класифікацію з високою впевненістю, що свідчить про узагальнюючу здатність та ефективність відібраних ознак.

Таблиця 2.4 – Приклади передбачень для тестових сайтів

URL-адреса	Ймовірність фішингу (%)	Рішення	Ключові спрацьовані ознаки
https://www.google.com	1.24	Безпечний	-
https://www.wikipedia.org	3.05	Безпечний	-
http://paypal-login-verification.com	76.67	Фішинг	google_index, domain_age, web_traffic
http://amazon-login-update-payment.xyz	82.41	Фішинг	page_rank, phish_hints, domain_in_title

У підсумку створена система демонструє поєднання високої точності класифікації з практичною придатністю для використання в реальному середовищі. Завдяки оптимізації набору ознак, кешуванню запитів і використанню ансамблевого алгоритму Random Forest досягнуто швидкого та надійного визначення потенційно фішингових ресурсів. Отримана модель є універсальною і вона може бути інтегрована у веб-браузер або застосована як серверний модуль для перевірки URL-адрес у потоковому режимі. Така архітектура забезпечує масштабованість, легке оновлення компонентів і можливість подальшого розширення системи за рахунок нових характеристик або алгоритмів машинного навчання.

2.5 Висновки до розділу 2

У межах другого розділу здійснено повний цикл дослідження, розробки та навчання методу машинного навчання для виявлення фішингових веб-сайтів. Проведена робота охоплює всі ключові етапи побудови системи – від підготовки даних і вибору алгоритму до формування оптимального набору ознак, тренування моделі та оцінки її ефективності.

У результаті порівняння кількох поширених алгоритмів класифікації (Random Forest, Gradient Boosting, SVM, Logistic Regression, MLP Neural Net) встановлено, що найкращі показники точності, стабільності та швидкодії

продемонстрував ансамблевий метод Random Forest. Саме він показав оптимальне поєднання високої точності (≈96%), збалансованого F1-score і низького часу передбачення, що є критично важливим для систем, які працюють у режимі реального часу.

Подальший аналіз був спрямований на відбір найбільш інформативних характеристик, які впливають на результат класифікації. За допомогою механізму Feature Importance у Random Forest визначено 50 найвагоміших ознак, серед яких ключовими стали показники google_index, page_rank, web_traffic, domain_age, ratio_extHyperlinks. Встановлено, що саме ці параметри найкраще відображають довіру до веб-ресурсу та його технічну репутацію. Проведене тестування показало, що використання понад 50 ознак не дає значного приросту точності, проте збільшує обчислювальні витрати, тому така кількість була обрана як оптимальна.

Після завершення етапу відбору ознак виконано навчання фінальної моделі, у ході якого реалізовано автоматизований процес обробки даних, стандартизації, масштабування та серіалізації результатів. Отримані моделі, масштабувальник і порядок ознак збережено у форматах .joblib, що дає змогу легко інтегрувати їх у подальший програмний продукт. Тестування підтвердило, що модель здатна точно розпізнавати навіть складні випадки фішингу, ефективно працює з новими даними та демонструє високу узагальнювальну здатність.

Загалом, у розділі розроблено ефективний метод і навчено модель, яка забезпечує високоточне, швидке та інтерпретоване виявлення фішингових веб-ресурсів. Це створює основу для подальшої реалізації повноцінного інтелектуального засобу “Phishing Detector”, що може функціонувати як самостійний інструмент захисту користувачів у мережі.

3 РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ ПРОГРАМНОГО ЗАСОБУ

3.1 Архітектура системи “Phishing Detector”

Архітектура системи побудована за модульним принципом і включає три основні компоненти: браузерний плагін (frontend), серверну частину Flask API (backend) та модель машинного навчання, яка виконує класифікацію URL-адрес. Такий підхід забезпечує масштабованість, швидкодію та можливість незалежного оновлення компонентів.

Основні складові системи:

Frontend (браузерний плагін) – перехоплює URL, надсилає запит до API, відображає результат.

Backend (Flask API) – отримує адресу, формує набір ознак і передає їх у ML-модель.

ML-модель (Random Forest) – аналізує ознаки та повертає класифікацію “phishing / legitimate”.

Взаємодія компонентів системи відбувається у вигляді послідовного потоку даних (рис. 3.1):

1) Користувач відкриває веб-сторінку або вводить URL-адресу в адресному рядку браузера.
2) Плагін (Frontend) автоматично перехоплює адресу сторінки та надсилає запит до сервера перевірки.
3) Backend (Flask-сервер) приймає запит, виконує попередню обробку, формує набір ознак для URL і передає їх до моделі.
4) ML-модель (Random Forest) виконує класифікацію на основі відібраних 50 характеристик і повертає ймовірність фішингу.

5) Результат обробляється бекендом і передається назад до плагіна, який відображає повідомлення користувачу наприклад: “Фішинговий сайт” або “Безпечний ресурс”.

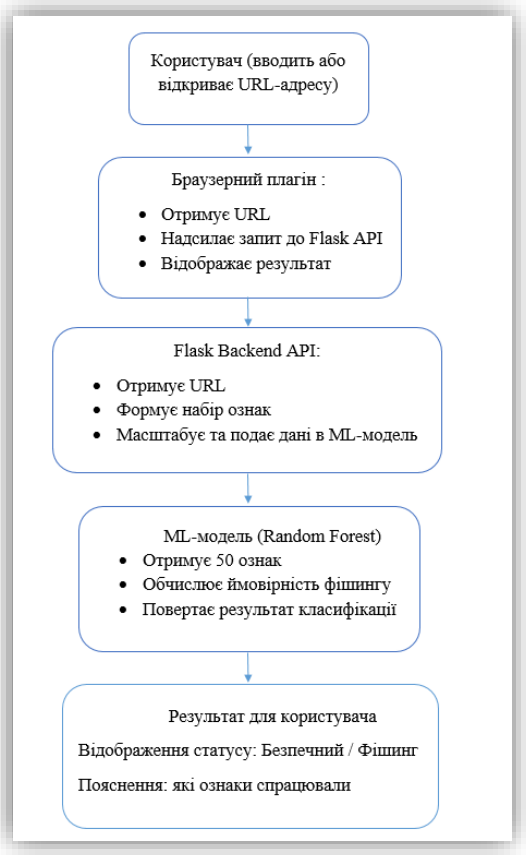


Рисунок 3.1 – Архітектура системи “Phishing Detector”

Архітектура системи “Phishing Detector” побудована за модульним принципом, що забезпечує чіткий поділ функцій між складовими частинами та високу масштабованість. Кожен компонент виконує власне завдання від збору даних у браузері до обробки та класифікації на сервері. Усі елементи взаємодіють між собою через чітко визначений API, що дозволяє підтримувати стабільний обмін інформацією та гнучке оновлення окремих частин без впливу на всю систему.

Фронтендна частина системи реалізована у вигляді браузерного плагіна, створеного за допомогою технологій JavaScript, HTML і CSS. Вона виконує роль інтерфейсу взаємодії користувача із системою та забезпечує візуальне відображення результатів перевірки безпосередньо у браузері. Після встановлення плагін інтегрується у браузер і автоматично перехоплює URL-адресу активної вкладки, коли користувач відкриває нову сторінку. Завдяки цьому відбувається миттєвий аналіз сайту без потреби у додаткових діях користувача. Отримана адреса передається через HTTPS-запит до серверної частини (Flask API) для подальшої обробки. Після отримання результату плагін відображає користувачу інформаційне або попереджувальне повідомлення, залежно від рівня виявленого ризику.

Повідомлення реалізоване у форматі напівпрозорого overlay-вікна або компактного toast-повідомлення у нижньому куті екрана. Візуальна складова оформлена у фірмовій неоновій кольоровій гамі (синьо-фіолетові акценти), що підвищує зручність сприйняття.

Серверна частина системи реалізована на мікрофреймворку Flask (Python), що забезпечує стабільну логіку обробки запитів і взаємодію з моделлю машинного навчання. Backend виконує ключову роль у роботі системи саме тут відбувається отримання URL-адреси, її аналітична обробка, формування набору ознак і передача цих даних у модель для класифікації.

Для підвищення швидкодії реалізовано механізм кешування запитів за допомогою функцій cached_html(), cached_dns() та cached_whois(). Це дозволяє уникнути повторного звернення до зовнішніх джерел у випадку перевірки того самого домену, значно скорочуючи час відповіді.

Основний маршрут API /predict, який приймає вхідний параметр url і повертає структуровану JSON-відповідь. Результат включає оцінену ймовірність

фішингу, текстову класифікацію (“phishing” або “legitimate”) та список основних ознак, що вплинули на рішення моделі.

Формат відповіді має вигляд:

```
{
   "url": "https://example.com",
   "probability": 0.742,
   "label": "phishing",
   "features_triggered":["domain_age", "google_index",
   "web_traffic"]
}
```

Таким чином, сервер виступає проміжною ланкою між браузерним плагіном і моделлю машинного навчання, виконуючи функції аналітичного ядра системи.

3.2 Розробка та інтеграція компонентів системи

Розробка системи “Phishing Detector” передбачала створення взаємопов’язаних компонентів, які об'єднані спільною метою забезпечити автоматичну та швидку перевірку веб-ресурсів на наявність ознак фішингу. Архітектура проєкту організована таким чином, щоб чітко розділити відповідальність між браузерним плагіном, серверною частиною Flask API та моделлю машинного навчання. Завдяки цьому система працює стабільно, легко розширюється і може масштабуватися без потреби у переробці всієї структури.

Загальна структура директорій та файлів проєкту зображена на рисунку 3.2, який демонструє компоненти системи: модулі браузерного плагіна, набір моделей машинного навчання, набір функцій для обчислення ознак та серверну логіку на Python. Папка chrome_extension містить усі елементи, необхідні для роботи плагіна, включаючи файли background.js і content.js. Папка models містить серіалізовані моделі у форматі .joblib, а app_api.py відповідає за серверну інтеграцію.

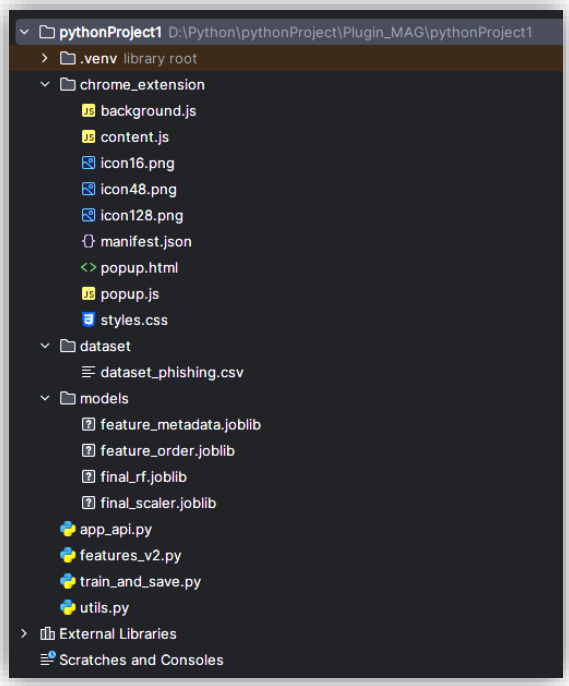


Рисунок 3.2 – Структура проєкту системи “Phishing Detector”

Браузерний плагін виконує роль інтерфейсу взаємодії користувача із системою. Він автоматично перехоплює URL-адресу активної вкладки, формує запит до локального Flask API та отримує відповідь з результатами перевірки. Основними елементами плагіна є файли background.js, content.js та manifest.json. Background.js працює у фоновому режимі та відповідає за комунікацію з API. Content.js обробляє відображення повідомлень безпосередньо на сторінці користувача саме через нього реалізовано попереджувальні overlay-вікна та інформаційні toast-повідомлення. Manifest.json визначає дозволи, підключення сценаріїв та базову конфігурацію розширення. Інтерфейс встановленого розширення у браузері Chrome представлений на рисунку 3.3.

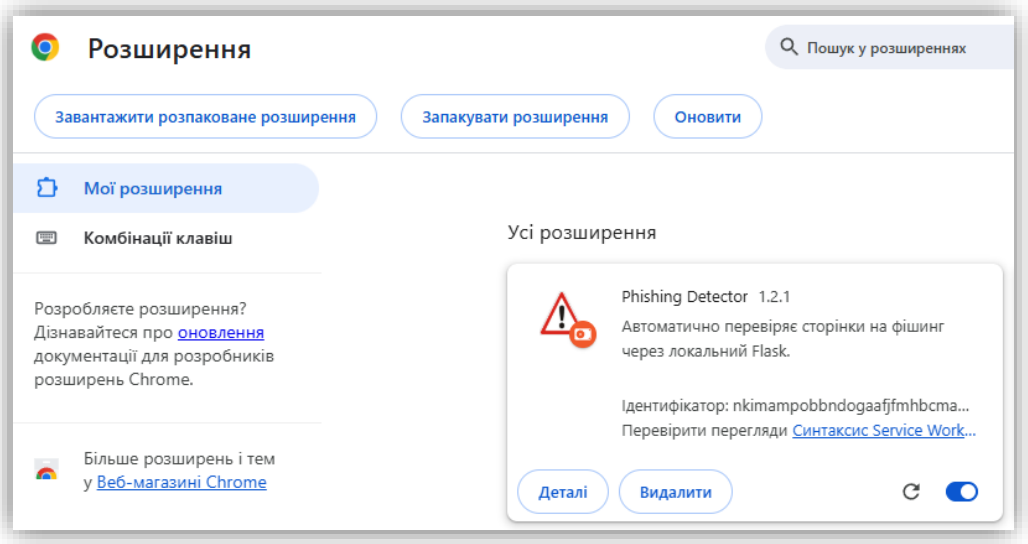


Рисунок 3.3 – Встановлене розширення у браузер Google Chrome

Взаємодія плагіна із backend відбувається через HTTP-запити до єдиного маршруту /predict. На кожному відкритті нового сайту плагін надсилає короткий JSON-пакет з URL-адресою. Flask API отримує цю адресу, виконує її перевірку, формує набір ознак та передає їх у модель машинного навчання. Відповідь API повертається у плагін і містить оцінку ймовірності фішингу, статус («phishing» або «legitimate») та ключові ознаки, які вплинули на рішення. Плагін, у свою чергу, відображає користувачу попередження або повідомлення про безпеку сайту.

Серверна частина розроблена на мікрофреймворку Flask, який забезпечує просту й ефективну логіку роботи API. На етапі запуску сервер завантажує попередньо натреновану модель Random Forest (файл final_rf.joblib), нормалізатор (файл final_scaler.joblib) та список ознак для коректного порядку подачі даних у модель. Після отримання запиту сервер виконує кілька етапів обробки: аналіз структури домену, перевірку віку домену, аналіз контенту HTML-сторінки, оцінку наявності фішингових патернів тощо. Усі розрахунки реалізовані у модулі features_v2.py, що містить понад 50 функцій обробки ознак.

Для підвищення продуктивності система використовує механізми кешування DNS-даних, HTML-коду та WHOIS-запитів. Це дозволяє значно прискорити обробку повторних перевірок одного і того ж домену, зменшуючи навантаження на зовнішні мережеві ресурси. Після формування ознак сервер нормалізує їх, подає до моделі машинного навчання, отримує прогноз і завершує цикл, повертаючи результат плагіну.

Безпека обміну даними забезпечується тим, що вся система працює виключно локально. Плагін взаємодіє із сервером через адресу 127.0.0.1, що гарантує відсутність передачі інформації у зовнішні мережі. Крім того, manifest.json визначає лише мінімально необхідні дозволи для роботи розширення, а сам API приймає лише один параметр URL, який не містить персональних даних користувача.

У результаті інтеграції всіх компонентів створено цілісну систему, яка працює у реальному часі та забезпечує інтуїтивну взаємодію з користувачем. Архітектура побудована таким чином, щоб підтримувати подальше розширення можливостей, зокрема інтеграцію нових моделей, додавання додаткових ознак або створення версії для інших браузерів.

3.3 Інтерфейс користувача та функціональні можливості плагіна

Інтерфейс користувача відіграє важливу роль у системі “Phishing Detector”, оскільки від нього залежить зручність взаємодії та швидкість прийняття рішень користувачем щодо відкритих веб-ресурсів. Плагін працює повністю автоматично, без потреби у ручних діях: щойно користувач відкриває новий сайт або переходить за посиланням, система негайно запускає процес аналізу URL-адреси через локальний Flask API та відображає результат безпосередньо у браузері. Для цього застосовується набір візуальних компонентів overlay-вікна,

toast-повідомлення та інтерактивні кнопки, що дозволяють користувачу швидко реагувати на потенційні ризики. Всі елементи оформлені у єдиному неоновому стилі, який підсилює візуальний акцент на важливих повідомленнях і водночас зберігає сучасний, мінімалістичний вигляд.

Першим етапом взаємодії користувача із системою є екран “Перевірка сайту…”, який з’являється автоматично під час завантаження сторінки. Це overlay-вікно займає центральну частину екрана, затемнюючи фон і повідомляючи, що модель виконує аналіз HTML-коду, доменних характеристик і поведінкових параметрів сайту. Такий підхід забезпечує ефект "активної роботи системи", пояснюючи користувачу, що перевірка триває лише кілька секунд. Приклад завантажувального вікна наведено на рисунку 3.4.

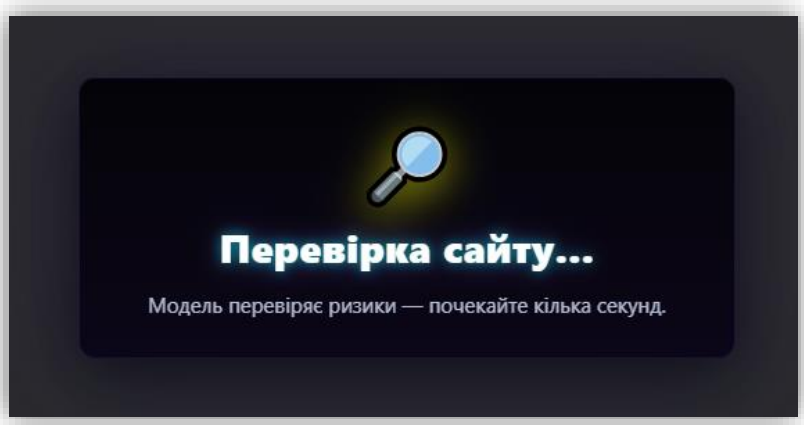


Рисунок 3.4 – Overlay-вікно “Перевірка сайту…” під час аналізу URL

Після завершення перевірки система визначає, чи є сайт безпечним, і відображає відповідне повідомлення. Якщо веб-ресурс не містить ознак фішингу, користувач бачить компактне toast-повідомлення у нижній частині екрана. Такі сповіщення не блокують перегляд сторінки та зникають через дві–три секунди, що робить процес непомітним та інтуїтивним. Це особливо важливо для ресурсів, якими користувач відвідує часто, забезпечуючи ненав’язливий, але

інформативний досвід роботи з плагіном. Приклад такого сповіщення продемонстровано на рисунку 3.5.

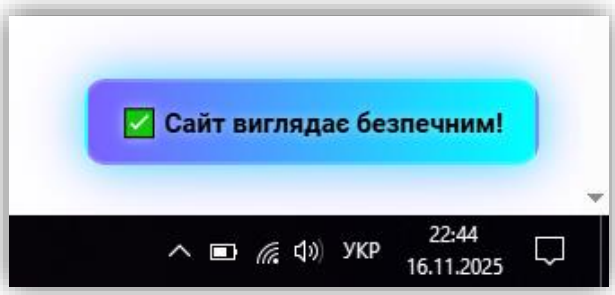


Рисунок 3.5 – Повідомлення про безпечний сайт після перевірки

У разі виявлення фішингових ознак плагін переходить до попереджувального режиму та відображає повноекранне overlay-вікно з яскравим заголовком “Можливий фішинг – будьте обережні”. Центральне повідомлення містить короткий текстовий аналіз ризику та оцінку ймовірності фішингу, виражену у відсотках. Це дозволяє користувачу швидко зрозуміти рівень небезпеки ресурсу. Важливо, що на першому етапі показується лише загальна характеристика без технічних деталей, щоби не перевантажувати інтерфейс та уникнути страху користувача перед складними термінами. Відповідний приклад наведено на рисунку 3.6.

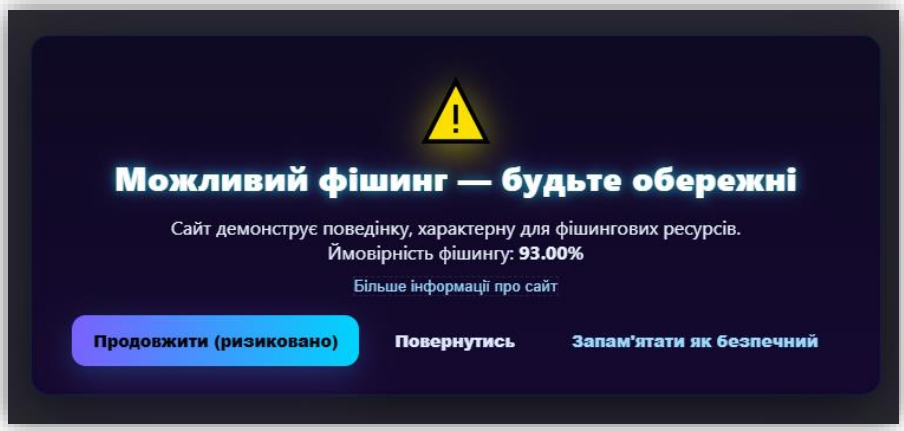


Рисунок 3.6 – Попередження про можливий фішинг (згорнута інформація)

У випадку, якщо користувач бажає отримати більше інформації про причини та ознаки ризику, у центрі інтерфейсу доступна кнопка “Більше інформації про сайт”. Після натискання відкривається додатковий блок зі списком технічних характеристик: значення індексації Google, рівень веб-трафіку, вік домену, кількість зовнішніх посилань, наявність небезпечних скриптів та багато інших параметрів, що вплинули на результат класифікації. Для зручності та наочності кожна ознака має короткий опис доступною мовою. Це дозволяє користувачам різного рівня підготовки від звичайних відвідувачів до кібербезпекових аналітиків зрозуміти, чому сайт позначено як потенційно небезпечний. Розгорнуте попередження наведено на рисунку 3.7.

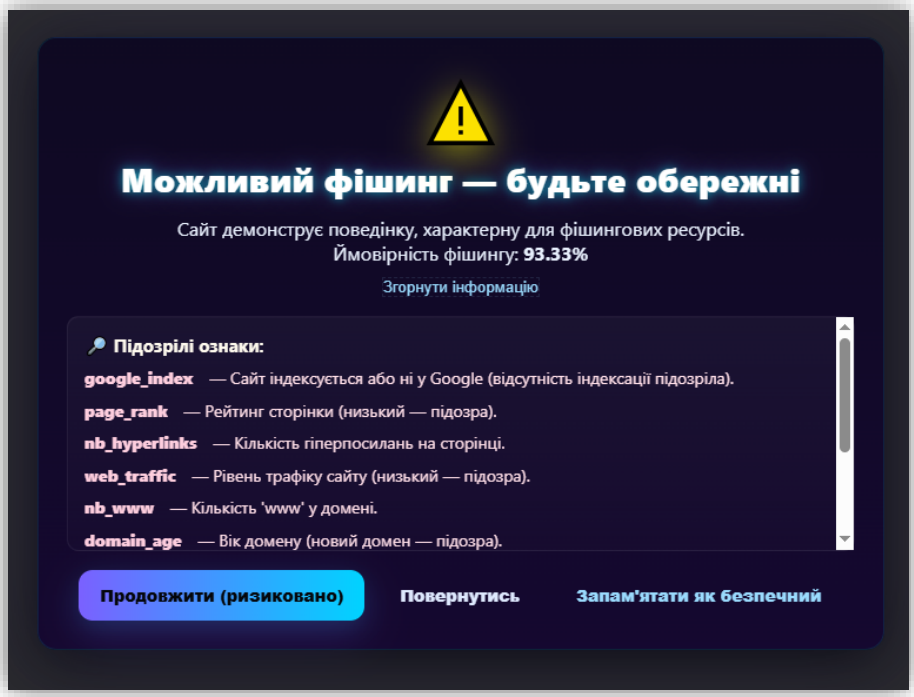


Рисунок 3.7 – Розгорнутий перелік підозрілих ознак сайту

Ще однією важливою функцією плагіна є механізм whitelist – список довірених сайтів, які не потребують подальшої перевірки. Якщо користувач свідомо хоче дозволити відкриття ресурсу, він може натиснути кнопку “Запам’ятати як безпечний”. Після цього плагін автоматично додає домен у

локальний whitelist і наступного разу пропускає перевірку. При успішному додаванні користувач отримує ненав’язливе toast-повідомлення у нижній частині екрана. Приклад наведено на рисунку 3.8.

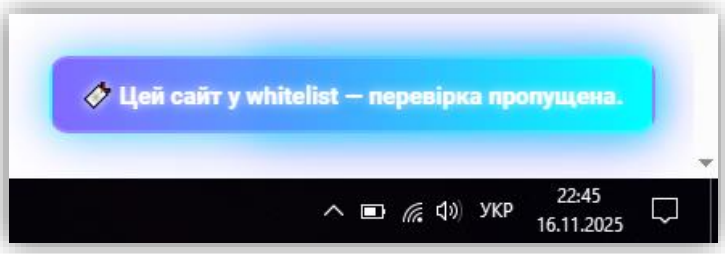


Рисунок 3.8 – Повідомлення про пропуск перевірки для сайтів у whitelist

Завдяки такій структурі інтерфейсу система забезпечує плавний, чіткий та інформативний користувацький досвід. Плагін не лише попереджає про потенційні загрози, а й дає можливість користувачам самостійно контролювати поведінку системи. Чітка візуальна ієрархія, неоновий стиль елементів, інтерактивні кнопки та автоматичний режим роботи роблять “Phishing Detector” зручним, сучасним та ефективним інструментом для захисту від фішингових атак. Він не вимагає спеціальних знань чи окремих дій – достатньо просто відкривати сайти у браузері, і система самостійно здійснює моніторинг безпеки в реальному часі.

3.4 Тестування програмного засобу

Тестування програмного забезпечення “Phishing Detector” проводилося з метою оцінювання його надійності, точності та стійкості під час роботи з реальними веб-ресурсами. Перевірка охоплювала широкий спектр сайтів, включаючи як легітимні високонавантажені ресурси, так і велику кількість фішингових сторінок, зібраних із реальних інцидентів та відкритих баз даних шкідливих доменів. Під час тестування важливо було оцінити не лише

коректність класифікації, але й реакцію інтерфейсу, швидкість відображення попереджень, а також стабільність роботи локального Flask-сервера при частих запитах. Загальний підхід передбачав послідовне відкриття десятків сторінок різного типу, моделювання реальної поведінки користувача та аналіз відповідей системи на кожен із сценаріїв.

Особлива увага приділялася тестуванню на великому наборі фішингових і нефішингових URL-адрес. До вибірки увійшли авторизаційні сторінки банків, псевдо-служби підтримки, підроблені форми входу в соціальні мережі, а також легітимні ресурси таких сервісів, як Google, Microsoft, Amazon та інші. Серед перевірених прикладів були як безпечні сайти (наприклад, https://mail.google.com), так і підозрілі сторінки, зокрема https://autoservis-sindrak.com/wp-login.php, який демонструє типову поведінку фішингових ресурсів. Усі ці адреси дозволили отримати різноманітну картину ризиків, а також перевірити реакцію системи на найпоширеніші ознаки фішингу: від низького вебтрафіку до новостворених доменів, підозрілих шляхів URL та аномальної структури HTML.

Під час тестування на фішингових сайтах плагін послідовно відображав попереджувальне overlay-вікно з детальною інформацією про ймовірність фішингу. Наприклад, під час взаємодії з ресурсом autoservis-sindrak.com система визначила ризик у 93% та чітко повідомила про загрозу. Відповідне повідомлення наведено на рис. 3.7. Така поведінка підтверджує коректну роботу моделі і правильне трактування набору ознак, на основі яких формується висновок.

На легітимних ресурсах система демонструвала стабільно правильні результати. Під час тестування на сервісі Google Mail плагін відобразив ненав'язливе повідомлення “Сайт виглядає безпечним!”, що повністю відповідало очікуваній поведінці, оскільки ресурс має високий рівень захисту,

довірений домен та перевірену інфраструктуру. Приклад такого повідомлення наведено на рис. 3.5.

Тестування точності моделі базувалося на аналізі матриці плутанини, створеної на основі спеціально підготовленої тестової вибірки, що містила збалансовану кількість фішингових та легітимних URL. Отримані результати показали високу ефективність класифікації: модель правильно розпізнала 1094 легітимні сайти та 1103 фішингові, тоді як кількість помилкових класифікацій була мінімальною: 49 хибнопозитивних і 40 хибнонегативних випадків. Сформована матриця плутанини наведена на рисунку 3.9 і свідчить про високу точність роботи, що становить близько 95.7%. Такий показник є конкурентним у порівнянні з багатьма комерційними рішеннями та підтверджує ефективність використання моделі Random Forest у задачах виявлення фішингу.

Рисунок 3.9 – Матриця плутанини моделі Random Forest

Швидкодія системи також була важливою частиною тестування. У реальних умовах час аналізу становив у середньому від 0,8 до 1,6 секунди, що дозволяє відображати результати без помітних затримок при переході між

сторінками. Навіть на важких, великих HTML-сторінках час перевірки рідко перевищував 2 секунди. Це стало можливим завдяки локальному кешуванню DNS, WHOIS та HTML-відповідей, а також попередньому завантаженню моделі в пам’ять Flask-сервера. Протягом тестування не було зафіксовано жодних збоїв або зависань, що підтверджує стабільність системи при інтенсивному використанні.

Для розуміння ефективності плагіна було виконано порівняльний аналіз із традиційними засобами, такими як Google Safe Browsing та антивірусні браузерні фільтри. На відміну від них, “Phishing Detector” не залежить від зовнішніх баз, працює повністю локально та надає користувачу пояснення, чому сайт був визначений як небезпечний. Це робить систему не лише точною, але й прозорою, що є важливою перевагою у сфері кібербезпеки.

Тестування також дозволило визначити перспективи подальшого вдосконалення. Зокрема, розширення набору ознак, інтеграція з глибокими моделями аналізу контенту, удосконалення інтерфейсу та можливість централізованого збору анонімних даних для навчання можуть значно підвищити точність та зручність роботи системи.

У підсумку проведене тестування підтвердило, що програмний засіб працює коректно, стабільно та ефективно виявляє фішингові ресурси в реальних умовах. Висока точність моделі, швидка реакція плагіна та інформативний інтерфейс роблять “Phishing Detector” надійним інструментом для щоденного використання.

3.5 Висновки до розділу 3

У розділі було проведено комплексний аналіз архітектури, механізмів роботи та практичної реалізації системи «Phishing Detector», а також здійснено тестування програмного засобу в умовах, наближених до реальної експлуатації.

Розроблена система продемонструвала функціональну завершеність, високу точність класифікації та стабільну роботу під час обробки великої кількості веб-ресурсів різного типу. Побудована модульна архітектура, що включає браузерний плагін, серверну частину на Flask та інтегровану модель машинного навчання, забезпечила чіткий поділ відповідальності між компонентами та дозволила досягти високої швидкодії та зручності використання.

Фронтендна частина у вигляді браузерного плагіна забезпечила інтуїтивно зрозумілу взаємодію з користувачем і дозволила в реальному часі відображати результати перевірки URL-адрес. Реалізація візуальних елементів, зокрема overlay-вікон, toast-повідомлень та механізму whitelist, зробила роботу системи ненав’язливою та комфортною, водночас забезпечивши повноцінний захист від потенційно небезпечних сайтів. Плагін функціонує автоматично при кожному відкритті сторінки, що перетворює процес моніторингу безпеки на безперервний та прозорий для користувача.

Серверна частина, розроблена на Flask, продемонструвала стабільність та ефективність у формуванні ознак, обробці HTTP-запитів і взаємодії з моделлю машинного навчання. Застосовані механізми кешування дозволили суттєво скоротити час відповіді, що особливо важливо у випадку повторних звернень до того самого домену. Модель Random Forest, що лежить в основі системи, показала високу точність класифікації, яка склала близько 95–96%. Побудована матриця плутанини підтвердила здатність моделі ефективно розрізняти легітимні та фішингові ресурси, мінімізуючи кількість хибнопозитивних і хибнонегативних результатів.

Практичне тестування програмного засобу на великій кількості реальних фішингових та безпечних сайтів підтвердило здатність системи коректно реагувати на різноманітні загрози. Система успішно визначила шкідливі сторінки, зокрема з підробленими формами авторизації, новими або

маловідомими доменами, відсутністю індексації та неприродними URL-шляхами. Водночас вона стабільно розпізнавала легітимні ресурси, демонструючи високий рівень відповідності очікуванням користувача. Швидкість роботи плагіна, що у середньому становила близько однієї секунди, зробила процес перевірки непомітним та не створювала затримок при переході між сторінками.

Проведене порівняння з існуючими рішеннями показало, що запропонована система має суттєві переваги, зокрема локальну роботу без залежності від зовнішніх баз даних, прозоре пояснення причин виявлення загроз, можливість розширення функціоналу та адаптації під різні сценарії використання. Аналіз результатів також дав змогу визначити потенційні напрями вдосконалення, включаючи впровадження більш складних моделей глибинного навчання, розширення набору ознак та створення мобільної або корпоративної версії програмного засобу.

Таким чином, результати дослідження підтвердили, що розроблений засіб виявлення фішингових ресурсів є ефективним інструментом для забезпечення безпеки користувача під час роботи у вебпросторі. Система демонструє високу точність, зручність використання, швидку реакцію та має значний потенціал для подальшого розвитку й інтеграції у більш масштабні системи кіберзахисту.

4 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на практичне використання та подальше впровадження лише за умови, що вона відповідає сучасним вимогам розвитку інформаційних технологій, сприяє науково-технічному прогресу та є економічно доцільною. Тому в рамках виконання магістерської кваліфікаційної роботи важливо провести економічну оцінку результатів дослідження та визначити доцільність реалізації запропонованої розробки в реальних умовах.

Магістерська кваліфікаційна робота на тему «Метод і засіб для виявлення фішингових сайтів із використанням машинного навчання» належить до науково-технічних розробок, орієнтованих на можливе подальше впровадження у вигляді готового програмного продукту. У процесі дослідження можуть бути прийняті рішення щодо комерціалізації отриманих результатів, створення програмного засобу для ринку кіберзахисних сервісів або його інтеграції у вже існуючі системи інформаційної безпеки.

Актуальність створення системи автоматичного виявлення фішингових вебсайтів підтверджується стрімким зростанням кількості кіберінцидентів, що завдають суттєвих фінансових збитків як приватним особам, так і організаціям. Комплексна система на основі методів машинного навчання здатна забезпечити швидше й точніше виявлення фішингових загроз порівняно з традиційними методами. Це створює значну ринкову потребу у таких рішеннях, а отже – перспективу комерціалізації розробки [27].

Для того, щоб надати інвестору або потенційному замовнику об’єктивне економічне бачення проекту, необхідно виконати низку етапів економічного аналізу, а саме:

1) провести комерційний та технологічний аудит науково-технічної розробки, визначивши її конкурентоспроможність, потенціал впровадження та рівень технологічної новизни;

2) розрахувати витрати на створення науково-технічної розробки, включаючи витрати на оплату праці, матеріали, обладнання, програмне забезпечення, енергоресурси та накладні витрати;
3) оцінити економічну ефективність впровадження та комерціалізації, включно з розрахунком терміну окупності, чистого економічного ефекту та доцільності інвестицій.

Виконання цих етапів дає змогу визначити, чи є розробка економічно обґрунтованою, яку потенційну вигоду може отримати інвестор, та чи здатна система забезпечити конкурентні переваги на ринку послуг із кіберзахисту [28].

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою комерційного та технологічного аудиту дослідження за темою «Метод і засіб для виявлення фішингових сайтів із використанням машинного навчання» є визначення рівня науково-технічної новизни, технологічності розробки, а також оцінка її комерційного потенціалу.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах

Продовження таблиці 4.1

Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування

Продовження таблиці 4.1

10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки

Критерії	Експерт (ПІБ, посада)		
	Гарнага В. А.	Долюк Д. П.	Войтович О. П.
	Бали:		
1. Технічна здійсненність концепції	3	3	3
2. Ринкові переваги (наявність аналогів)	2	4	3
3. Ринкові переваги (ціна продукту)	3	4	3
4. Ринкові переваги (технічні властивості)	3	4	4
5. Ринкові переваги (експлуатаційні витрати)	3	4	3
6. Ринкові перспективи (розмір ринку)	3	4	4
7. Ринкові перспективи (конкуренція)	3	4	3

Продовження таблиці 4.2

8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	3	4	4
10. Практична здійсненність (необхідність нових матеріалів)	3	3	4
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів	42	43	44
Середньоарифметична сума балів $СБ_с$	**43**		

За результатами розрахунків, наведених в таблиці 2.5, робиться висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використовують рекомендації, наведені в табл. 4.3.

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Метод і засіб для виявлення фішингових сайтів із використанням машинного навчання» становить 43 бала, що, відповідно до таблиці 4.3, свідчить про високий науково-технічний рівень та високий комерційний потенціал розробки. Отриманий результат підтверджує доцільність подальшої практичної реалізації програмного засобу, можливість його впровадження у сфері інформаційної безпеки, а також перспективність комерціалізації на ринку програмних продуктів з кіберзахисту.

4.2 Розрахунок витрат на проведення науково-дослідної роботи

Проведення науково-дослідної роботи, спрямованої на створення методу та програмного засобу для виявлення фішингових сайтів із використанням методів машинного навчання, потребує визначення та обґрунтування всіх необхідних витрат, пов’язаних із виконанням дослідження. У відповідності до методичних рекомендацій, витрати на виконання науково-дослідної та дослідно-конструкторської роботи групуються за визначеними економічними статтями, що відображають структуру собівартості створення науково-технічної продукції.

При плануванні та калькулюванні витрат на розробку програмного засобу для виявлення фішингових вебресурсів враховуються як прямі, так і непрямі витрати, що забезпечують повний цикл виконання дослідження. До них належать витрати, пов’язані з оплатою праці виконавців науково-дослідної роботи, нарахуваннями на фонд заробітної плати, використанням матеріалів та енергетичних ресурсів, а також витрати на експлуатацію комп’ютерної техніки, необхідної для навчання моделей машинного навчання, тестування та апробації програмного продукту.

Окрему частку витрат становлять витрати, пов’язані з використанням спеціального обладнання та програмного забезпечення, які враховуються шляхом нарахування амортизаційних відрахувань за фактичний період використання.

Загальна структура витрат також включає накладні (загальновиробничі) витрати, які забезпечують організаційне та адміністративне супроводження виконання науково-дослідної роботи. До них належать витрати на утримання приміщень, інфраструктури, обладнання та робочих місць. У сукупності всі зазначені витрати формують повну собівартість науково-дослідної роботи та дозволяють об’єктивно оцінити економічну доцільність розробки програмного

засобу для виявлення фішингових сайтів із використанням методів машинного навчання.

4.3 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим ви конанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці, також будь-які види грошових і матеріальних до плат, які належать до елемента «Витрати на оплату праці».

Проведені розрахунки зведемо до таблиці.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник науково-дослідної роботи (науковий консультант з кібербезпеки)	25 200,00	1 145,45	20	22 909,00
Фахівець з машинного навчання (ML-інженер, розробка моделі класифікації)	24 600,00	1 118,18	12	13 418,16
Розробник програмного забезпечення (інженер-програміст, реалізація детектора)	22 400,00	1 018,18	18	18 327,24

Продовження таблиці 4.4

Технік лабораторії / інженер із тестування (налаштування середовища, тестування моделі)	10 200,00	463,64	10	4 636,40
Всього				59 290,80

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників:

$$З_о = \sum_{i=1}^{k} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (4.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

$М_{ni}$ – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

$Т_р$ – середнє число робочих днів в місяці, $Т_р$=20 днів.

$З_о$ = 25 200,00· 20 / 20 = 25 200,00 грн.

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Метод і засіб для виявлення фішингових сайтів із використанням машинного навчання» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} С_i \cdot t_i, \quad (4.2)$$

де $С_i$ – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду $С_i$ можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (4.3)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо *Мм* = 8000,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б);

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 20 дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 8000,00 \cdot 1,10 \cdot 1,15 / (20 \cdot 8) = 63,25$ грн.
$З_{р1} = 63,25 \cdot 6,00 = 379,50$ грн.

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка обчислювального обладнання для проведення розробки та досліджень системи виявлення фішингових сайтів	6,00	2	1,10	63,25	379,50

Продовження таблиці 4.5

Підготовка робочого місця розробника системи машинного навчання	4,50	3	1,35	77,62	349,31
Інсталяція програмного забезпечення для розробки і тестування засобу виявлення фішингових сайтів	5,00	4	1,50	86,25	431,25
Формування та завантаження навчальних і тестових вибірок URL-адрес	12,00	3	1,35	77,62	931,50
Обробка та валідація логів мережевого трафіку для навчання моделі	18,00	3	1,35	77,62	1 397,25
Запуск серій навчання моделі та збереження контрольних результатів	7,00	5	1,70	97,75	684,25
Налаштування серверного середовища для розгортання детектора фішингових сайтів	10,00	6	2,00	115,00	1150,00

Продовження таблиці 4.5

Проведення тестування працездатності системи на контрольній вибірці	5,00	2	1,10	63,25	316,25
Технічний супровід цифрового експерименту	6,50	4	1,50	86,25	560,62
Всього					6 199,93

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (5 9290,80+6 199,93) · 11 / 100% = 7203,98 грн.

4.4 Відрахування на соціальні заходи

До статті «Відрахування на соціальні заходи» належать відрахування внеску на загальнообов’язкове державне соціальне страхування та для здійснення заходів щодо соціального захисту населення (ЄСВ – єдиний со ціальний внесок). Нарахування на заробітну плату дослідників та робітників розраховується як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (59290,80+6199,93+7203,98)· 22/100 = 15992,84 грн.

4.5 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби й предмети праці, які придбані у сторонніх підприємств, установ і організа цій та витрачені на проведення досліджень за прямим призначенням згідно з нормами їх витрачання, а також витрачені придбані напівфабрикати, що підлягають монтажу або виготовленню й додатковій обробці в цій органі зації, чи дослідні зразки, що виготовляються виробниками за документаці єю наукової організації.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}\,, \qquad (4.6)$$

де H_j – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

K_j – коефіцієнт транспортних витрат, (K_j = *1,1 … 1,15*);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1$ =0,50 · 60,00 · 1,05 – 0,05 · 15,00 = 30,75 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, од.	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір офісний А4 (80 г/м²)	60,00	0,50	0,05	15,00	30,75
Картридж лазерний (тонер)	1 200,00	0,10	0,01	200,00	124,00
USB-накопичувач (16 ГБ)	900,00	0,05	0	0	47,25
Кабель мережевий UTP Cat5e	25,00	2,00	0,10	5,00	52,00
Блокнот робочий	80,00	0,20	0	0	16,80
Канцелярські витратні матеріали (ручки, маркери)	150,00	0,10	0	0	15,75
Всього					286,55

4.6 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Метод і засіб для виявлення фішингових сайтів із використанням машинного навчання», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.7)$$

де $Н_j$ – кількість комплектуючих j-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих *j*-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*).

$К_в = 1 \cdot 2500{,}00 \cdot 1{,}05 = 2625{,}00$ *грн*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.7– Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
SSD NVMe 1 ТБ (для швидкої обробки датасетів та зберігання експериментів)	1	2 500,00	2 625,00
ОЗП DDR4 16 ГБ (для навчання/тестування моделей ML)	1	1 600,00	1 680,00
ДБЖ (UPS) 1500 VA (захист робочої станції під час експериментів)	1	3 200,00	3 360,00
USB-накопичувач 32 ГБ (резервні копії коду/даних)	2	250,00	525,00
Всього			8 190,00

4.7 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення. Втрати на «Спецустаткування» відсутні.

4.8 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для

проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.8)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$B_{прг}$ = *350,00·1·1,05=367,50 грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.8 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Доступ до мережі Internet (високошвидкісний, 1 місяць)	1	350,00	367,50
Хмарне середовище для ML-експериментів (GPU/VM, підписка)	1	1 200,00	1 260,00
Система керування базами даних PostgreSQL / MySQL (хостинг, підписка)	1	500,00	525,00
Інтегроване середовище розробки PyCharm Professional	1	900,00	945,00
Інструменти кібербезпеки та аналізу URL (API-сервіси, підписка)	1	650,00	682,50
Всього			3 780,00

4.9 Амортизація обладнання, програмних засобів та приміщень

До статті «Амортизація обладнання, програмних засобів та приміщень» відносять амортизаційні відрахування по кожному виду обладнання, устаткування та інших приладів і пристроїв, а також програмного забезпе чення для проведення науково-дослідної роботи, за його наявності в дослідній організації або на підприємстві. В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо можуть бути розраховані з використанням прямолінійного методу амортизації за фор мулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.9)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (52000,00 \cdot 1) / (4 \cdot 12) = 1083,33$ *грн*

Таблиця 4.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Робоча станція для машинного навчання (CPU + GPU + 32 GB RAM + NVMe SSD)	52 000,00	*4*	*1*	1 083,33

Продовження таблиці 4.9

Монітор професійний 27″	9 500,00	5	1	158,33
Робоче місце інженера-програміста	7 700,00	5	1	128,33
Робоче місце аналітика з кібербезпеки	8 990,00	5		149,83
Ноутбук для тестування браузерного плагіна	42 000,00	3	1	1 166,67
Операційна система Windows 11 Pro	8 460,00	3	1	235,00
Прикладний пакет Microsoft Office 2019	7 840,00	3	1	217,78
Мережеве обладнання (маршрутизатор / комутатор)	9 000,00	4	1	187,50
Приміщення лабораторії досліджень	650 000,00	30	1	1 805,56
Оргтехніка (принтер, сканер тощо)	12 800,00	5	1	213,33
Всього				5 345,66

4.10 Паливо та енергія для науково-виробничих цілей

До статті «Паливо та енергія для науково-виробничих цілей» нале жать витрати на придбання у сторонніх підприємств, установ і організацій будь-якого палива, що витрачається з технологічною метою на проведення досліджень. Стаття формується у разі виконання енергоємних наукових досліджень за методом прямого внесення витрат і досягає значної питомої ваги у собівартості досліджень.

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (4.10)$$

де W_{yi} – встановлена потужність обладнання на етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,35 \cdot 172,0 \cdot 12,56 \cdot 0,95 / 0,97 = 740,52$ грн.

Таблиця 4.10 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Робоча станція для ML-аналізу (CPU+GPU+32 GB RAM+NVMe SSD)	0,35	172,0	740,52
Ноутбук для тестування браузерного плагіна	0,08	172,0	169,26
Монітор 27″	0,03	172,0	63,47
Маршрутизатор / мережеве обладнання	0,02	172,0	42,32
Робоче місце інженера-програміста (освітлення/периферія)	0,07	172,0	148,10
Робоче місце аналітика з кібербезпеки (освітлення/периферія)	0,07	172,0	148,10
Принтер (періодичний друк)	0,25	4,0	12,30
Оргтехніка (сканер/зарядні)	0,20	6,0	14,76
Всього			1 338,84

4.11 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Метод і засіб для виявлення фішингових сайтів із використанням машинного навчання» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (4.11)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св} = 0\%$.

$В_{св}$ = (59290,80+6199,93) · 0 / 100% = 0,00 грн.

4.12 Витрати на роботи, які виконують сторонні підприємства, установи і організації

До статті «Витрати на роботи, які виконують сторонні підприємс тва, установи і організації» належать витрати на проведення досліджень, що не можуть бути виконані штатними працівниками або наявним облад нанням організації, а виконуються на договірній основі іншими підприємс твами, установами і організаціями незалежно від форм власності та позаштатними працівниками. Витрати за статтею «Витрати на роботи, які виконують сторонні під приємства, установи і організації» розраховуються як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (4.12)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$B_{сп}$ = (59290,80+6199,93) · 30 / 100% = 19647,22 грн.

4.13 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відо браження у зазначених статтях витрат і можуть бути віднесені безпосеред ньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.13)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$I_в$ = (59290,80+6199,93) · 50 / 100% = 32745,37 грн.

4.14 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науковотехнічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.14)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$В_{нзв}$ = (59290,80+6199,93) · 100 / 100% = 65490,73 грн.

Витрати на проведення науково-дослідної роботи на тему «Метод і засіб для виявлення фішингових сайтів із використанням машинного навчання» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.18)$$

$В_{заг}$=59290,80+6199,93+7203,98+15992,84+286,55+8190,00+3780,00+5345,66+1338,84 +0,00+19647,22+32745,37+65490,73 = 225511,92 грн

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.15)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

ЗВ = 225511,92 / 0,9 = 250568,80 грн.

4.15 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Метод і засіб для виявлення фішингових сайтів із використанням машинного навчання» передбачають комерціалізацію протягом 4-х років реалізації на ринку. Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	5000	10000	7000	3000

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 200000 осіб;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 210,00 грн;

$\pm\Delta Ц_o$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 67,81 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою:

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.16)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2025 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо $\rho = 40\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2025 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$$\Delta П_1 = (67,81 \cdot 200000 + 277,81 \cdot 5000) \cdot 0,8333 \cdot 0,4 \cdot 0,9982 = 4974513,71 \text{ грн}$$

Збільшення чистого прибутку 2-го року:

$$\Delta П_2 = (67,81 \cdot 200000 + 277,81 \cdot 15000) \cdot 0,8333 \cdot 0,4 \cdot 0,9982 = 5898843,21 \text{ грн}$$

Збільшення чистого прибутку 3-го року:

$$\Delta П_3 = (67,81 \cdot 200000 + 277,81 \cdot 22000) \cdot 0,8333 \cdot 0,4 \cdot 0,9982 = 6545873,86 \text{ грн}$$

Збільшення чистого прибутку 4-го року:

$$\Delta П_4 = (67,81 \cdot 200000 + 277,81 \cdot 25000) \cdot 0,8333 \cdot 0,4 \cdot 0,9982 = 6823172,71 \text{ грн}$$

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \tag{4.17}$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,12;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП = 4974513,71/(1+0,12)^1 + 5898843,21/(1+0,12)^2 + 6545873,86/(1+0,12)^3 + 6823172,71/(1+0,12)^4 = 18139525,13 грн.$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{4.18}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$ =1,5;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 250568,80 грн.

$PV = k_{інв} \cdot ЗВ = 1,5 \cdot 250568,80 = 375853,20 грн$

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.19}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 18139525,13 грн;

PV – теперішня вартість початкових інвестицій, 375853,20 грн.

$E_{абс} = ПП - PV = 18139525,13 - 375853,20 = 17763671,93 грн$

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.20)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 18139525,13 грн;

PV – теперішня вартість початкових інвестицій, 375853,20 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+17763671,93/375853,20)^{1/4} = 2,64.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (4.21)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d =0,11;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,3.

$\tau_{мін} = 0,11+0,3 = 0,41 < 2,64$ свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою

«Метод і засіб для виявлення фішингових сайтів із використанням машинного навчання» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{4.22}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок} = 1 / 2,64 = 0,38$ р.

$T_{ок} <$ 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

4.16 Висновки до розділу 4

У ході виконання економічного розділу магістерської кваліфікаційної роботи було проведено комплексний розрахунок витрат, пов’язаних із розробкою та впровадженням науково-технічної розробки на тему «Метод і засіб для виявлення фішингових сайтів із використанням машинного навчання».

Визначено структуру та величину основних статей витрат, зокрема витрат на оплату праці дослідників і робітників, додаткової заробітної плати, відрахувань на соціальні заходи, матеріальних витрат, комплектуючих, програмного забезпечення, амортизації обладнання і приміщень, витрат на електроенергію, а також накладних і інших витрат. Загальні витрати на завершення науково-дослідної роботи та оформлення її результатів становлять 250 568,80 грн.

Проведений розрахунок економічної ефективності науково-технічної розробки за умов її можливої комерціалізації потенційним інвестором показав високі показники інвестиційної привабливості. Абсолютний економічний ефект від впровадження та комерціалізації розробки становить 17 763 671,93 грн, що значно перевищує обсяг початкових інвестицій.

Розрахована внутрішня економічна дохідність інвестицій дорівнює 2,64, що суттєво перевищує мінімально допустимий рівень внутрішньої дохідності (0,41). Період окупності інвестицій становить 0,38 року, що свідчить про швидке повернення вкладених коштів.

Отримані результати підтверджують економічну доцільність, конкурентоспроможність та комерційну привабливість розробленого програмного засобу для виявлення фішингових сайтів із використанням машинного навчання. Запропонована науково-технічна розробка може бути рекомендована для впровадження та подальшої комерціалізації на ринку програмних продуктів у сфері кібербезпеки.

ВИСНОВКИ

У магістерській кваліфікаційній роботі було проведено комплексне дослідження проблеми фішингових атак як однієї з ключових загроз сучасного інформаційного простору. У ході виконання роботи встановлено, що традиційні методи протидії фішингу, побудовані на використанні чорних списків, сигнатурному аналізі або статичних евристичних правилах, втрачають ефективність в умовах динамічного розвитку шахрайських технік. Постійне збільшення кількості нових фішингових ресурсів, швидкість їх створення та здатність зловмисників мінімізувати ознаки підробки роблять неможливим забезпечення належного рівня кіберзахисту без застосування інтелектуальних технологій аналізу даних. Це визначило актуальність створення адаптивного методу автоматичного виявлення фішингових сайтів з використанням алгоритмів машинного навчання.

У теоретичній частині здійснено аналіз природи фішингу, його класифікації, типових ознак та психологічних прийомів, що застосовуються для маніпулювання користувачами. Розглянуто існуючі системи боротьби з фішингом, визначено їх сильні та слабкі сторони, а також сформульовано критерії, яким повинна відповідати сучасна інтелектуальна система виявлення загроз: автономність, висока точність, здатність обробляти великі обсяги даних, відсутність залежності від сторонніх баз знань та швидкодія при аналізі сайтів у реальному часі.

На основі аналізу датасету, що містив 11 430 прикладів веб-сайтів із 89 характеристиками, було встановлено, що дані є збалансованими, репрезентативними та придатними для побудови моделі машинного навчання. Проведено глибоку оцінку важливості ознак та взаємної кореляції, визначено найбільш інформативні характеристики, що відображають структурні, доменні,

поведінкові та контентні властивості веб-ресурсів. У результаті експериментів встановлено, що оптимальною є модель, побудована на основі 50 найважливіших ознак, які забезпечують баланс між точністю передбачення та швидкодією.

Порівняльний аналіз алгоритмів машинного навчання (Random Forest, Gradient Boosting, SVM, MLP Neural Network, Logistic Regression, Voting Ensemble) показав, що найкращі результати забезпечує Random Forest. Цей алгоритм досяг точності Accuracy = 0.9611, F1-score ≈ 0.962, Recall ≈ 0.967, що є найвищими показниками серед протестованих моделей. Додаткове підтвердження отримано з аналізу ROC-кривих, де модель Random Forest продемонструвала найвищий показник AUC = 0.993, що свідчить про майже ідеальну здатність розділяти фішингові та легітимні сайти. Такі результати вказують на високу ефективність обраного алгоритму у задачах багатовимірної класифікації, а також його здатність працювати у реальному часі завдяки швидкому передбаченню (≈ 0.04 мс на один запис).

Практичною частиною роботи стала розробка програмного засобу “Phishing Detector” браузерного плагіна, який інтегрує модель машинного навчання і забезпечує аналіз сайтів у момент їх відкриття. Плагін отримує технічні параметри сторінки, формує вектор ознак, здійснює класифікацію і миттєво інформує користувача про рівень безпеки ресурсу. Ключовою перевагою реалізованої системи є її автономність: для роботи не потрібно надсилати дані на сторонні сервери, що повністю виключає ризик витоку приватної інформації користувача. Окрім цього, плагін надає пояснення, які саме фактори стали підставою для класифікації, що робить його прозорим і корисним з освітнього погляду.

У процесі тестування програмний засіб було перевірено на великій кількості реальних фішингових та нефішингових сторінок. Використовувалися різні типи URL, сайти з підозрілими доменами, сторінки зі складною структурою

перенаправлень, легітимні сервіси, поштові платформи та авторизаційні форми. Результати тестування підтвердили стабільність роботи моделі, малу кількість хибних спрацювань та високу швидкість обробки запитів. Прикладом є коректне виявлення фішингових елементів на підроблених формах авторизації та впевнене визначення легітимних ресурсів служб Google, банківських сервісів та авторизованих портальних систем.

Окремо встановлено, що система демонструє стійкість до штучно ускладнених фішингових доменів, включно з доменами-двійниками, URL із прихованими піддоменами, назвами, що імітують популярні бренди, або складними параметрами запиту. Це підтверджує здатність моделі ідентифікувати приховані закономірності, що недоступні для традиційних чорних списків.

Економічна частина роботи засвідчила доцільність розробки такого програмного засобу завдяки низькій собівартості реалізації, високому рівню масштабованості і відсутності витрат на підтримку сторонніх серверів або на оновлення баз даних. Модель може бути інтегрована у корпоративні системи, освітні установи, інструменти веб-безпеки та антивірусні середовища.

Підсумовуючи проведене дослідження, можна стверджувати, що поставлена мета магістерської кваліфікаційної роботи досягнута повністю. Розроблений метод і засіб для виявлення фішингових сайтів на основі машинного навчання продемонстрував високу ефективність, швидкодію, адаптивність та практичну цінність. Створений плагін може слугувати як самостійним інструментом для забезпечення кібербезпеки кінцевих користувачів, так і складовою частиною комплексних систем захисту інформації. Отримані результати відкривають перспективи для подальшого вдосконалення засобу, зокрема через впровадження нейронних мереж глибокого навчання, механізмів онлайн-перенавчання моделі, багатомодального аналізу контенту або розширення функціоналу плагіна до системи активної веб-фільтрації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке фішинг (What is Phishing?). IBM. URL: https://www.ibm.com/topics/phishing (дата звернення: 03.10.2025).
2. ESET. Фішинг. Енциклопедія загроз. URL: https://www.eset.com/ua/support/information/entsyklopediya-zahroz/fishynh/?srsltid=AfmBOoqBqyn_U2uv7mMCEgUJIsZDyovB47430iceZViOksCeRa5cNw3U (дата звернення: 05.10.2025).
3. Гнатюк М. В., Гарнага В. А. Засіб виявлення фішингових сайтів із використанням машинного навчання // Молодь в науці: дослідження, проблеми, перспективи (МН-2026) : матеріали наук.-практ. конф. Вінниця : ВНТУ, 2026. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/26837 (дата звернення: 16.12.2025).
4. Phishing Activity Trends Report. Anti-Phishing Working Group. URL: https://apwg.org/trendsreports/ (дата звернення: 03.10.2025).
5. Anti-Phishing Working Group. Phishing Activity Trends Report, Q4 2024. URL: https://docs.apwg.org/reports/apwg_trends_report_q4_2024.pdf (дата звернення: 20.10.2025).
6. Kymatio. Phishing Trends: AI Phishing, QRishing and Voice Attacks. URL: https://kymatio.com/blog/phishing-trends-ai-phishing-qrishing-and-voice-attacks (дата звернення: 21.10.2025).
7. OWASP Foundation. Unvalidated Redirects and Forwards Cheat Sheet. URL: https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html (дата звернення: 22.10.2025).
8. Alhasnawi B. N., Sadeq A. M., Homod R. Z., Hussain F. F. K., Soběslav V., Bureš V. An extensive examination of cyberattacks, cybersecurity, and energy

management in smart grid, including new advancements and machine learning // Energy Conversion and Management: X. 2025. Vol. 18, 101471. DOI: 10.1016/j.ecmx.2025.101471. URL: https://www.sciencedirect.com/science/article/pii/S2590174525006038 (дата звернення: 22.10.2025).

9. Alasmari S. M., Sakly H., Kraiem N., Algarni A. Phishing detection in IoT: an integrated CNN-LSTM framework with explainable AI and LLM-enhanced analysis // Discover Internet of Things. 2025. Vol. 5, art. 102. DOI: 10.1007/s43926-025-00202-9. URL: https://link.springer.com/article/10.1007/s43926-025-00202-9 (дата звернення: 23.10.2025).
10. IBM. Random Forest: що це таке та як працює. URL: https://www.ibm.com/think/topics/random-forest (дата звернення: 25.10.2025).
11. GeeksforGeeks. Random Forest Algorithm in Machine Learning. URL: https://www.geeksforgeeks.org/machine-learning/random-forest-algorithm-in-machine-learning/ (дата звернення: 24.10.2025).
12. Scikit-learn developers. Ensemble methods – Random Forests. URL: https://scikit-learn.org/stable/modules/ensemble.html#forest (дата звернення: 25.10.2025).
13. Random Forest Algorithm Overview / S. K. Sharma, D. Singh. URL: https://www.researchgate.net/publication/382419308_Random_Forest_Algorithm_Overview (дата звернення: 26.10.2025).
14. Chand E. Phishing Website Detector Dataset. Kaggle. URL: https://www.kaggle.com/datasets/eswarchandt/phishing-website-detector (дата звернення: 29.10.2025).
15. Kaggle. Web Page Phishing Detection Dataset (updated). URL: https://www.kaggle.com/datasets/shashwatwork/web-page-phishing-detection-dataset?resource=download (дата звернення: 03.11.2025).

16. Scribbr. Pearson Correlation Coefficient — What It Is & How to Interpret It. URL: https://www.scribbr.com/statistics/pearson-correlation-coefficient/ (дата звернення: 02.11.2025).
17. Brownlee J. A Gentle Introduction to k-fold Cross-Validation. Machine Learning Mastery. URL: https://machinelearningmastery.com/k-fold-cross-validation/ (дата звернення: 03.11.2025).
18. DigitalOcean. K-Fold Cross-Validation in Python. URL: https://www.digitalocean.com/community/tutorials/k-fold-cross-validation-python (дата звернення: 07.11.2025).
19. Google Developers. ROC Curve and AUC. URL: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc (дата звернення: 31.10.2025).
20. GeeksforGeeks. AUC-ROC Curve in Machine Learning. URL: https://www.geeksforgeeks.org/machine-learning/auc-roc-curve/ (дата звернення: 08.11.2025).
21. Powers D. M. W. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation // Journal of Machine Learning Technologies. 2011. Vol. 2, no. 1. P. 37–63. DOI: 10.48550/arXiv.2010.16061. URL: https://doi.org/10.48550/arXiv.2010.16061 (дата звернення: 04.11.2025).
22. Python Software Foundation. Time – Time access and conversions. URL: https://docs.python.org/3/library/time.html#time.time (дата звернення: 06.11.2025).
23. Murphy K. P. Probabilistic Machine Learning: An Introduction. Cambridge : MIT Press, 2022. Розд.: Log-loss and Cross-Entropy. URL: https://probml.github.io/pml-book/book1.html (дата звернення: 09.11.2025).

24. Open Source Security. PSF deb-Nicholson 2025. URL: https://opensourcesecurity.io/2025/2025-09-psf-deb-nicholson/ (дата звернення: 13.11.2025).
25. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed. New York : Springer, 2009. Розд.: Tree Ensembles and Random Forests. URL: https://www.sas.upenn.edu/~fdiebold/NoHesitations/BookAdvanced.pdf (дата звернення: 14.11.2025).
26. Brownlee J. ROC Curves and Precision-Recall Curves for Classification in Python. Machine Learning Mastery. URL: https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/ (дата звернення: 14.11.2025).
27. Козловський В. О., Лесько О. Й., Кавецький В. В. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. Вінниця : ВНТУ, 2021. 42 с.
28. Кавецький В. В., Козловський В. О., Причепа І. В. Економічне обґрунтування інноваційних рішень : практикум. Вінниця : ВНТУ, 2016. 113 с.

ДОДАТКИ

Додаток А
Протокол перевірки кваліфікаційної роботи

Назва роботи: Метод і засіб для виявлення фішингових сайтів із використанням машинного навчання
Автор роботи: Гнатюк Максим Віталійович
Тип роботи: магістерська кваліфікаційна робота
Підрозділ кафедра захисту інформації ФІТКІ, група 1 БС-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism **0, 44%**

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

√ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

В. о. зав. кафедри ЗІ д. т. н., проф. _____ Володимир ЛУЖЕЦЬКИЙ

Гарант освітньої програми «Безпека інформаційних і комунікаційних систем» к.т.н., доцент _____ Олеся ВОЙТОВИЧ

Особа, відповідальна за перевірку _____ Валентина КАПЛУН

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Володимир ГАРНАГА

Здобувач _____ Максим ГНАТЮК

Додаток Б
Програмна реалізація браузерного плагіна

Файл app_api.py

```
from flask import Flask, request, jsonify
from flask_cors import CORS
import os, joblib, numpy as np
from features_v2 import extract_features_from_url
from utils import is_valid_http_url

BASE_DIR = os.path.dirname(__file__)
MODELS_DIR = os.path.join(BASE_DIR, 'models')

MODEL_FILE = os.path.join(MODELS_DIR, 'final_rf.joblib')
SCALER_FILE = os.path.join(MODELS_DIR, 'final_scaler.joblib')
ORDER_FILE = os.path.join(MODELS_DIR, 'feature_order.joblib')
META_FILE = os.path.join(MODELS_DIR, 'feature_metadata.joblib')

model = joblib.load(MODEL_FILE)
scaler = joblib.load(SCALER_FILE)
feature_order = joblib.load(ORDER_FILE)
meta = joblib.load(META_FILE)
app = Flask(__name__)
CORS(app)
@app.route('/health', methods= ['GET'])
def health():
    return jsonify({'status': 'ok'})
@app.route('/predict', methods= ['POST'])
def predict():
    data = request.get_json(force=True)
    url = data.get('url')
    if not url or not is_valid_http_url(url):
        return jsonify({'error': 'Invalid or missing URL'}), 400
    X = extract_features_from_url(url, feature_order)
    try:
        Xs = scaler.transform(X.fillna(0))
    except Exception as e:
        return jsonify({'error': 'scaler transform failed', 'exc': str(e)}), 500
    proba = float(model.predict_proba(Xs) [0,1])
    pred = int(proba >= 0.85)
    import numpy as np
    imp = model.feature_importances_
    idx = np.argsort(imp) [-10:] [::-1]
    top =  []
    for i in idx:
        feat = feature_order [i]
        top.append({'feature': feat, 'value': float(X.iloc [0, i]), 'importance': float(imp [i])})
    return jsonify({
        'url': url,
        'prob': proba,
        'label': pred,
        'top_features': top,
        'features': X.iloc [0].to_dict()
    })

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Файл features_v2.py

```
import re
from urllib.parse import urlparse
from bs4 import BeautifulSoup
import pandas as pd
```

```
from utils import cached_html, safe_hostname
def google_index_flag(url):
    return int(bool(cached_html(url)))

def page_rank_flag(url):
    u = url or ""
    s = 0
    s += int(len(u) < 60)
    s += int(u.count('/') <= 3)
    return int(s >= 1)

def nb_hyperlinks_flag(url):
    html = cached_html(url)
    if not html:
        return 0
    return html.lower().count('<a ')

def web_traffic_flag(url):
    return int(not bool(cached_html(url)))

def nb_www_flag(url):
    return safe_hostname(url).lower().count('www')

def domain_age_flag(url, threshold_days=180):
    return 1

def ratio_extHyperlinks_flag(url):
    html = cached_html(url)
    if not html:
        return 0
    soup = BeautifulSoup(html, 'html.parser')
    anchors =  [a for a in soup.find_all('a', href=True)]
    if not anchors:
        return 0
    ext =  []
    host = safe_hostname(url)
    for a in anchors:
        h = a.get('href', '')
        net = urlparse(h).netloc
        if net and (net not in host):
            ext.append(a)
    return len(ext) / len(anchors)

def phish_hints_flag(url):
    html = cached_html(url)
    if not html:
        return 0
    text = BeautifulSoup(html, 'html.parser').get_text(" ").lower()
    hints =  ['login', 'verify', 'update', 'password', 'account', 'secure', 'confirm', 'bank']
    return int(any(h in text for h in hints))

def longest_word_path_flag(url):
    p = urlparse(url).path or ""
    words = re.findall(r' [A-Za-z0-9]+', p)
    return max((len(w) for w in words), default=0)

def ratio_intHyperlinks_flag(url):
    html = cached_html(url)
    if not html:
        return 0
    soup = BeautifulSoup(html, 'html.parser')
    anchors =  [a for a in soup.find_all('a', href=True)]
    if not anchors:
        return 0
```

```
    host = safe_hostname(url)
    internal =  []
    for a in anchors:
        h = a.get('href', '')
        net = urlparse(h).netloc
        if net and (net in host):
            internal.append(a)
    return len(internal) / len(anchors)

def safe_anchor_flag(url):
    html = cached_html(url)
    if not html:
        return 0
    soup = BeautifulSoup(html, 'html.parser')
    anchors =  [a for a in soup.find_all('a', href=True)]
    if not anchors:
        return 0
    unsafe =  [a for a in anchors if a ['href'].startswith('javascript:') or a ['href'].startswith('#')]
    return int(len(unsafe) / len(anchors) > 0.5)

def ratio_extRedirection_flag(url):
    html = cached_html(url)
    if not html:
        return 0
    low = html.lower()
    return low.count('http://') + low.count('https://')

def length_url_flag(url):
    return len(url or "")

def ratio_digits_url_flag(url):
    if not url:
        return 0
    digits = sum(c.isdigit() for c in url)
    return digits / max(len(url), 1)

def longest_words_raw_flag(url):
    html = cached_html(url) or ""
    words = re.findall(r' [A-Za-z0-9]+', html)
    return max((len(w) for w in words), default=0)

def length_words_raw_flag(url):
    html = cached_html(url) or ""
    words = re.findall(r' [A-Za-z0-9]+', html)
    return len(words)

def char_repeat_flag(url):
    return int(bool(re.search(r'(.)\1{3,}', url or "")))

def length_hostname_flag(url):
    return len(safe_hostname(url) or "")

def avg_word_path_flag(url):
    p = urlparse(url).path or ""
    words = re.findall(r' [A-Za-z0-9]+', p)
    if not words:
        return 0
    return sum(len(w) for w in words) / len(words)

def links_in_tags_flag(url):
    html = cached_html(url)
    if not html:
        return 0
    soup = BeautifulSoup(html, 'html.parser')
    return len(soup.find_all( ['a', 'link', 'script', 'img']))
```

```
def shortest_word_host_flag(url):
    parts =  [p for p in (safe_hostname(url) or "").split('.') if p]
    if not parts:
        return 0
    return min((len(p) for p in parts))

def domain_registration_length_flag(url):
    return 0

def domain_in_title_flag(url):
    html = cached_html(url)
    if not html:
        return 0
    soup = BeautifulSoup(html, 'html.parser')
    t = soup.find('title')
    dom = (safe_hostname(url) or "").split('.') [0]
    return int(not t or dom.lower() not in t.get_text().lower())

def nb_slash_flag(url):
    return (url or "").count('/')

def shortest_word_path_flag(url):
    p = urlparse(url).path or ""
    words = re.findall(r' [A-Za-z0-9]+', p)
    if not words:
        return 0
    return min((len(w) for w in words))

def nb_dots_flag(url):
    return (safe_hostname(url) or "").count('.')

def ratio_digits_host_flag(url):
    h = safe_hostname(url) or ""
    digits = sum(c.isdigit() for c in h)
    return digits / max(len(h), 1)

def avg_words_raw_flag(url):
    html = cached_html(url) or ""
    words = re.findall(r' [A-Za-z0-9]+', html)
    if not words:
        return 0
    return sum(len(w) for w in words) / len(words)

def nb_hyphens_flag(url):
    return (url or "").count('-')

def avg_word_host_flag(url):
    parts =  [p for p in (safe_hostname(url) or "").split('.') if p]
    if not parts:
        return 0
    return sum(len(p) for p in parts) / len(parts)

def nb_qm_flag(url):
    return int('?' in (url or ""))

def nb_eq_flag(url):
    return int('=' in (url or ""))

def longest_word_host_flag(url):
    parts =  [p for p in (safe_hostname(url) or "").split('.') if p]
    if not parts:
        return 0
    return max((len(p) for p in parts))
```

```
def shortest_words_raw_flag(url):
    html = cached_html(url) or ""
    words = re.findall(r' [A-Za-z0-9]+', html)
    if not words:
        return 0
    return min((len(w) for w in words))

def domain_with_copyright_flag_binary(url):
    html = cached_html(url) or ""
    txt = html.lower()
    d = (safe_hostname(url) or "").split('.') [0]
    return int((('©' in txt or '(c)' in txt) and d in txt))

def ratio_extErrors_flag(url):
    html = cached_html(url) or ""
    return int('404' in html or 'not found' in html.lower())

def ip_flag(url):
    h = safe_hostname(url) or ""
    return int(bool(re.match(r'^(\d{1,3}\.){3}\d{1,3}$', h)))

def ratio_extMedia_flag(url):
    html = cached_html(url) or ""
    return html.lower().count('<img ')

def ratio_intMedia_flag(url):
    html = cached_html(url)
    if not html:
        return 0
    soup = BeautifulSoup(html, 'html.parser')
    imgs =  [i for i in soup.find_all('img', src=True)]
    if not imgs:
        return 0
    host = safe_hostname(url)
    int_imgs =  []
    for i in imgs:
        src = i.get('src', '')
        net = urlparse(src).netloc or ""
        if host and (host in net):
            int_imgs.append(i)
    return len(int_imgs) / len(imgs)

def nb_extCSS_flag(url):
    html = cached_html(url) or ""
    return html.lower().count('.css')

def nb_subdomains_flag(url):
    parts =  [p for p in (safe_hostname(url) or "").split('.') if p]
    return max(0, len(parts) – 2)

def domain_in_brand_flag(url):
    brands =  ['paypal', 'apple', 'amazon', 'google', 'microsoft', 'facebook', 'bank', 'visa']
    dom = (safe_hostname(url) or "").lower()
    return int(any(b in dom and not dom.startswith(b) for b in brands))

def nb_redirection_flag(url):
    html = cached_html(url) or ""
    low = html.lower()
    return int('window.location' in low or 'meta http-equiv="refresh"' in low)

def shortening_service_flag(url):
    shorteners =  ['bit.ly', 'goo.gl', 'tinyurl', 't.co', 'ow.ly', 'is.gd']
    host = safe_hostname(url) or ""
    return int(any(s in host for s in shorteners))
```

```
def nb_underscore_flag(url):
    return (url or "").count('_')

def prefix_suffix_flag(url):
    left = (safe_hostname(url) or "").split('.') [0]
    return int('-' in left)

def https_token_flag(url):
    host = (safe_hostname(url) or "").lower()
    scheme = urlparse(url).scheme
    return int('https' in host and scheme != 'https')

def nb_and_flag(url):
    return int('&' in (url or ""))

def external_favicon_flag(url):
    html = cached_html(url)
    if not html:
        return 0
    soup = BeautifulSoup(html, 'html.parser')
    links = soup.find_all('link', rel=True)
    host = safe_hostname(url)
    for l in links:
        rel = ' '.join(l.get('rel',  [])).lower()
        if 'icon' in rel:
            href = l.get('href', '')
            net = urlparse(href).netloc
            if href and net and (host and net not in host):
                return 1
    return 0

def empty_title_flag(url):
    html = cached_html(url)
    if not html:
        return 1
    t = BeautifulSoup(html, 'html.parser').find('title')
    return int(not t or not t.get_text(strip=True))

FEATURE_NAME_ALIASES = {
    "google_index": "google_index_flag",
    "page_rank": "page_rank_flag",
    "nb_hyperlinks": "nb_hyperlinks_flag",
    "web_traffic": "web_traffic_flag",
    "nb_www": "nb_www_flag",
    "domain_age": "domain_age_flag",
    "ratio_extHyperlinks": "ratio_extHyperlinks_flag",
    "phish_hints": "phish_hints_flag",
    "longest_word_path": "longest_word_path_flag",
    "ratio_intHyperlinks": "ratio_intHyperlinks_flag",
    "safe_anchor": "safe_anchor_flag",
    "ratio_extRedirection": "ratio_extRedirection_flag",
    "length_url": "length_url_flag",
    "ratio_digits_url": "ratio_digits_url_flag",
    "longest_words_raw": "longest_words_raw_flag",
    "length_words_raw": "length_words_raw_flag",
    "char_repeat": "char_repeat_flag",
    "length_hostname": "length_hostname_flag",
    "avg_word_path": "avg_word_path_flag",
    "links_in_tags": "links_in_tags_flag",
    "shortest_word_host": "shortest_word_host_flag",
    "domain_registration_length": "domain_registration_length_flag",
    "domain_in_title": "domain_in_title_flag",
    "nb_slash": "nb_slash_flag",
    "shortest_word_path": "shortest_word_path_flag",
    "nb_dots": "nb_dots_flag",
```

```
    "ratio_digits_host": "ratio_digits_host_flag",
    "avg_words_raw": "avg_words_raw_flag",
    "nb_hyphens": "nb_hyphens_flag",
    "avg_word_host": "avg_word_host_flag",
    "nb_qm": "nb_qm_flag",
    "nb_eq": "nb_eq_flag",
    "longest_word_host": "longest_word_host_flag",
    "shortest_words_raw": "shortest_words_raw_flag",
    "domain_with_copyright": "domain_with_copyright_flag_binary",
    "ratio_extErrors": "ratio_extErrors_flag",
    "ip": "ip_flag",
    "ratio_extMedia": "ratio_extMedia_flag",
    "ratio_intMedia": "ratio_intMedia_flag",
    "nb_extCSS": "nb_extCSS_flag",
    "nb_subdomains": "nb_subdomains_flag",
    "domain_in_brand": "domain_in_brand_flag",
    "nb_redirection": "nb_redirection_flag",
    "shortening_service": "shortening_service_flag",
    "nb_underscore": "nb_underscore_flag",
    "prefix_suffix": "prefix_suffix_flag",
    "https_token": "https_token_flag",
    "nb_and": "nb_and_flag",
    "external_favicon": "external_favicon_flag",
    "empty_title": "empty_title_flag"
}
def resolve_feature_callable(feature_name):
    key = FEATURE_NAME_ALIASES.get(feature_name, feature_name)
    if isinstance(key, str) and key in globals() and callable(globals() [key]):
        return globals() [key]
    if key + '_flag' in globals() and callable(globals() [key + '_flag']):
        return globals() [key + '_flag']
    alt = key.replace('-', '_').lower()
    if alt in globals() and callable(globals() [alt]):
        return globals() [alt]
    if alt + '_flag' in globals() and callable(globals() [alt + '_flag']):
        return globals() [alt + '_flag']
    return None

def extract_features_from_url(url, feature_order):
    row = {}
    for f in feature_order:
        func = resolve_feature_callable(f)
        if func is None:
            row [f] = 0
            continue
        try:
            v = func(url)
            if v is None:
                row [f] = 0
            elif isinstance(v, (float, int)):
                row [f] = v
            else:
                row [f] = int(bool(v))
        except Exception:
            row [f] = 0
    return pd.DataFrame( [row], columns=feature_order)
```

Файл background.js

```
console.log("✅ Background worker запущено (updated)");

const API_URL = "http://127.0.0.1:5000/predict";
const WHITELIST_KEY = "phish_whitelist_v1";

function hostnameFromUrl(url) {
```

```
  try {
    return new URL(url).hostname.replace(/^www\./, "");
  } catch (e) {
    return "";
  }
}

async function isWhitelisted(url) {
  return new Promise((res) => {
    chrome.storage.local.get( [WHITELIST_KEY], (obj) => {
      const list = (obj [WHITELIST_KEY] && Array.isArray(obj [WHITELIST_KEY])) ? obj [WHITELIST_KEY] :  [];
      const host = hostnameFromUrl(url);
      res(list.some(w => host.endsWith(w)));
    });
  });
}

chrome.webNavigation.onBeforeNavigate.addListener(async (details) => {
  try {
    const url = details.url;
    const tabId = details.tabId;
    if (!url || typeof url !== "string") return;
    if (!url.startsWith("http")) return;
    if (!tabId || tabId < 0) return;

    if (await isWhitelisted(url)) {
      console.log("▟ Whitelisted, skip check:", url);
      chrome.tabs.sendMessage(tabId, { type: "phish_whitelist", url });
      return;
    }

    chrome.tabs.sendMessage(tabId, { type: "show_loading_screen" });
    const controller = new AbortController();
    const timeout = setTimeout(() => controller.abort(), 8000);

    let res;
    try {
      res = await fetch(API_URL, {
        method: "POST",
        headers: { "Content-Type": "application/json; charset=utf-8" },
        body: JSON.stringify({ url }),
        signal: controller.signal
      });
    } catch (e) {
      console.warn("✕ Fetch failed or timed out", e);
      chrome.tabs.sendMessage(tabId, { type: "phish_check_error", message: "Backend недоступний  сторінка не перевірялась." });
      return;
    } finally {
      clearTimeout(timeout);
    }

    if (!res || !res.ok) {
      chrome.tabs.sendMessage(tabId, { type: "phish_check_error", message: "Backend повернув помилку або недоступний." });
      return;
    }

    const data = await res.json();
    chrome.tabs.sendMessage(tabId, { type: "phish_check_result", result: data });

  } catch (err) {
    console.error("background onBeforeNavigate error:", err);
  }
});
```

Файл content.js

```
console.log("✅ content.js (updated) активовано для", window.location.href);
const OVERLAY_ID = "phishWarning";
function describeFeature(name) {
   const map = {
      google_index: "Сайт індексується або ні у Google (відсутність індексації підозріла).",
      page_rank: "Рейтинг сторінки (низький – підозра).",
      nb_hyperlinks: "Кількість гіперпосилань на сторінці.",
      web_traffic: "Рівень трафіку сайту (низький – підозра).",
      nb_www: "Кількість 'www' у домені.",
      domain_age: "Вік домену (новий домен – підозра).",
      ratio_extHyperlinks: "Частка зовнішніх посилань (забагато – підозріло).",
      phish_hints: "Текст містить слова: login/verify/password/etc.",
      longest_word_path: "Довжина найдовшого слова у шляху URL.",
      ratio_intHyperlinks: "Співвідношення внутрішніх посилань.",
      safe_anchor: "Багато unsafe-лінків (javascript:, #) – підозріло.",
      ratio_extRedirection: "Кількість зовнішніх редиректів.",
      length_url: "Загальна довжина URL.",
      ratio_digits_url: "Частка цифр у URL.",
      longest_words_raw: "Довжина найдовшого слова у HTML.",
      length_words_raw: "Кількість слів у HTML.",
      char_repeat: "Наявність повторюваних символів.",
      length_hostname: "Довжина hostname.",
      avg_word_path: "Середня довжина слів у шляху.",
      links_in_tags: "Кількість тегів з посиланнями (a, img, script...).",
      shortest_word_host: "Найкоротша частина домену.",
      domain_registration_length: "Тривалість реєстрації домену.",
      domain_in_title: "Чи домен присутній у <title> сторінки.",
      nb_slash: "К-сть '/' у URL.",
      shortest_word_path: "Найкоротше слово у шляху URL.",
      nb_dots: "К-сть '.' у домені.",
      ratio_digits_host: "Частка цифр у домені.",
      avg_words_raw: "Середня довжина слів у HTML.",
      nb_hyphens: "К-сть дефісів у URL.",
      avg_word_host: "Середня довжина частин домену.",
      nb_qm: "Наявність '?' у URL.",
      nb_eq: "Наявність '=' у рядку запиту.",
      longest_word_host: "Найдовше слово у домені.",
      shortest_words_raw: "Найкоротше слово у HTML.",
      domain_with_copyright: "Є © або (c) разом з доменом.",
      ratio_extErrors: "Є 404/Not found помилки на сторінці.",
      ip: "Використовується IP замість домену.",
      ratio_extMedia: "К-сть медіа-елементів.",
      ratio_intMedia: "Частка внутрішніх медіа.",
      nb_extCSS: "К-сть зовнішніх CSS.",
      nb_subdomains: "К-сть піддоменів.",
      domain_in_brand: "У домені згадується бренд (paypal, apple...).",
      nb_redirection: "Наявність JS/meta-редиректів.",
      shortening_service: "Скорочувач в URL (bit.ly, t.co...).",
      nb_underscore: "К-сть '_' у URL.",
      prefix_suffix: "Дефіс у назві хосту (можлива підміна).",
      https_token: "Слово 'https' в середині хосту (імітація).",
      nb_and: "К-сть '&' у запиті.",
      external_favicon: "Favicon завантажується з зовнішнього хосту.",
      empty_title: "Відсутній <title> сторінки."
    };
  return map [name] || "Незвичайна поведінка.";
}

function createLoadingOverlay() {
  if (document.getElementById(OVERLAY_ID)) return;
  const overlay = document.createElement("div");
  overlay.id = OVERLAY_ID;
  overlay.className = "phish-overlay loading";
```

```
  overlay.innerHTML = `
    <div class="phish-box loading-box">
      <div class="icon">🔎</div>
      <h2>Перевірка сайту...</h2>
      <p class="muted">Модель перевіряє ризики – почекайте кілька секунд.</p>
    </div>
  `;
  document.documentElement.appendChild(overlay);
}

function removeOverlay() {
  const ex = document.getElementById(OVERLAY_ID);
  if (ex) ex.remove();
}

function showWarning({ status, prob, features =  [], message, url }) {
  removeOverlay();
  const overlay = document.createElement("div");
  overlay.id = OVERLAY_ID;
  overlay.className = "phish-overlay";
  const header =
    status === "error"
      ? "Система недоступна"
      : "Можливий фішинг –  будьте обережні";
  const probText =
    status === "error"
      ? `<p class="muted">${message || ""}</p>`
      : `<p>Сайт демонструє поведінку, характерну для фішингових ресурсів.<br>Ймовірність фішингу: <b>${prob}%</b></p>`;

  const hasFeatures = Array.isArray(features) && features.length > 0;
  const detailsBlock = hasFeatures
    ? `
      <button id="toggleDetails" class="btn-ghost" style="margin-top:10px;">
        Більше інформації про сайт
      </button>
      <div id="featuresBox" class="features" style="display:none;">
        <h3>🔎 Підозрілі ознаки:</h3>
        <ul>
          ${features
            .map(
              (f) =>
                `<li><span class="feat">${f.feature}</span>  – ${describeFeature(
                  f.feature
                )}</li>`
            )
            .join("")}
        </ul>
      </div>
    `
    : "";
  overlay.innerHTML = `
    <div class="phish-box phishing neon">
      <div class="icon">⚠</div>
      <h2>${header}</h2>
      ${probText}
      ${detailsBlock}
      <div class="buttons">
        <button id="continue" class="btn-primary">Продовжити (ризиковано)</button>
        <button id="back" class="btn-secondary">Повернутись</button>
        <button id="remember" class="btn-ghost">Запам'ятати як безпечний</button>
      </div>
    </div>
  `;
  document.documentElement.appendChild(overlay);
```

```
  overlay.querySelector("#continue").onclick = () => {
    removeOverlay();
  };
  overlay.querySelector("#back").onclick = () => {
    try {
      if (window.history.length > 1) window.history.back();
      else window.close();
    } catch (e) {
      window.close();
    }
  };

  overlay.querySelector("#remember").onclick = async () => {
    const host = new URL(window.location.href).hostname.replace(/^www\./, "");
    chrome.storage.local.get( ["phish_whitelist_v1"], (obj) => {
      const list = Array.isArray(obj ["phish_whitelist_v1"])
        ? obj ["phish_whitelist_v1"]
        : [];
      if (!list.includes(host)) list.push(host);
      chrome.storage.local.set({ phish_whitelist_v1: list }, () => {
        const btn = overlay.querySelector("#remember");
        btn.textContent = "Додано – більше не перевіряти";
        btn.disabled = true;
        btn.style.opacity = "0.8";
      });
    });
  };
  if (hasFeatures) {
    const toggleBtn = overlay.querySelector("#toggleDetails");
    const featuresBox = overlay.querySelector("#featuresBox");

    toggleBtn.onclick = () => {
      const isHidden = featuresBox.style.display === "none";
      featuresBox.style.display = isHidden ? "block" : "none";
      toggleBtn.textContent = isHidden
        ? "Згорнути інформацію"
        : "Більше інформації про сайт";
    };
  }
}
chrome.runtime.onMessage.addListener((msg) => {
  if (!msg || !msg.type) return;

  if (msg.type === "show_loading_screen") {
    createLoadingOverlay();
    return;
  }

  if (msg.type === "phish_whitelist") {
    removeOverlay();
    const toast = document.createElement("div");
    toast.className = "phish-toast neon-toast";
    toast.textContent = " Цей сайт у whitelist – перевірка пропущена.";
    document.body.appendChild(toast);
    setTimeout(() => toast.remove(), 3000);
    return;
  }
  if (msg.type === "phish_check_error") {
    removeOverlay();
    showWarning({ status: "error", message: msg.message || "Помилка перевірки." });
    return;
  }
  if (msg.type === "phish_check_result") {
    removeOverlay();
    const { result } = msg;
```

```
  if (!result) return;
  const prob = typeof result.prob === "number" ? (result.prob * 100).toFixed(2) : "-";
  if (result.label === 1) {
    showWarning({
      status: "phishing",
      prob,
      features: result.top_features ||  [],
      url: result.url
    });
  } else {
    const toast = document.createElement("div");
    toast.className = "phish-toast";
    toast.textContent = `✅ Сайт виглядає безпечним!`;
    document.body.appendChild(toast);
    setTimeout(() => toast.remove(), 2500);
  }
 }
});
```

Файл manifest.json

```
{
  "manifest_version": 3,
  "name": "Phishing Detector",
  "version": "1.2.1",
  "description": "Перевіряє сайти на фішинг перед відкриттям, використовуючи локальну ML-модель.",
  "permissions":  ["tabs", "webRequest","webRequestBlocking", "webNavigation", "notifications", "scripting", "activeTab", "storage"],
  "host_permissions":  ["<all_urls>"],
  "background": {
    "service_worker": "background.js"
  },
  "content_scripts":  [
    {
      "matches":  ["<all_urls>"],
      "js":  ["content.js"],
      "css":  ["styles.css"]
    }
  ],
  "icons": {
    "16": "icon16.png",
    "48": "icon48.png",
    "128": "icon128.png"
  },
  "action": {
    "default_title": "Phishing Detector",
    "default_popup": "popup.html"
  }
}
```

Файл popup.html

```
<!doctype html>
<html>
```

```
<head>
  <meta charset="utf-8" />
  <title>Phishing Detector</title>
  <style>
    body { font-family: "Segoe UI", sans-serif; width: 300px; padding: 12px; background: #0b0620; color: #eaf2ff; }
    h3 { margin:0 0 8px; font-size: 16px; }
    button { margin-top:10px; padding:8px 10px; border-radius:8px; border:none; cursor:pointer; background:linear-gradient(90deg,#7b61ff,#00d4ff); color:#06060a; font-weight:700;}
    .small { font-size:13px; color:#bcd8ff; margin-top:8px; }
  </style>
</head>
<body>
  <h3>Phishing Detector</h3>
  <div class="small">Автоматично перевіряє сайти перед відкриттям.</div>
  <div style="display:flex; gap:8px; margin-top:10px;">
    <button id="showWhitelist" style="background:#222; color:#eaf2ff; padding:6px 8px; font-weight:600;">Whitelist</button>
  </div>
  <script src="popup.js"></script>
</body>
</html>
```

Файл popup.js

```
// popup.js
document.getElementById("test").addEventListener("click", async () => {
  const  [tab] = await chrome.tabs.query({ active: true, currentWindow: true });
  if (!tab?.url) return alert("Не знайдено активної сторінки.");
  fetch("http://127.0.0.1:5000/predict", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ url: tab.url })
  })
    .then(r => {
      if (!r.ok) throw new Error("API error");
      return r.json();
    })
    .then(d => {
      alert(`Ймовірність фішингу: ${(d.prob * 100).toFixed(2)}%`);
    })
    .catch(() => alert("API недоступне"));
});

document.getElementById("showWhitelist").addEventListener("click", () => {
  chrome.storage.local.get( ["phish_whitelist_v1"], (obj) => {
    const list = (obj ["phish_whitelist_v1"] && Array.isArray(obj ["phish_whitelist_v1"])) ? obj ["phish_whitelist_v1"] :  [];
    alert("Whitelist:\n" + (list.length ? list.join("\n") : "(порожній)"));
  });
```

```
        });
```

Файл utils.py

```
import requests
from urllib.parse import urlparse
from functools import lru_cache

try:
    from selenium import webdriver
    from selenium.webdriver.chrome.service import Service
    from webdriver_manager.chrome import ChromeDriverManager
except Exception:
    webdriver = None

USER_AGENT = (
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
    "AppleWebKit/537.36 (KHTML, like Gecko) "
    "Chrome/120.0.0.0 Safari/537.36"
)

def _fetch_html_requests(url: str, timeout: int = 8) -> str:
    try:
        r = requests.get(url, headers={"User-Agent": USER_AGENT}, timeout=timeout)
        if r.status_code == 200 and r.text and len(r.text) > 100:
            return r.text
        return ""
    except Exception:
        return ""

def _fetch_html_selenium(url: str) -> str:
    if webdriver is None:
        return ""
    try:
        options = webdriver.ChromeOptions()
        options.add_argument("--headless=new")
        options.add_argument("--no-sandbox")
        options.add_argument("--disable-dev-shm-usage")
        options.add_argument("--disable-blink-features=AutomationControlled")
        options.add_argument("--disable-gpu")
        options.add_argument("--window-size=1280,800")

        driver = webdriver.Chrome(
            service=Service(ChromeDriverManager().install()),
            options=options
        )
        driver.get(url)
```

```
        html = driver.page_source or ""
        driver.quit()
        return html if len(html) > 100 else ""
    except Exception:
        return ""


@lru_cache(maxsize=None)
def cached_html(url: str) -> str:
    html = _fetch_html_requests(url)
    if html:
        return html
    html = _fetch_html_selenium(url)
    return html or ""


def safe_hostname(url: str) -> str:
    try:
        return urlparse(url).hostname or ""
    except Exception:
        return ""


def is_valid_http_url(url: str) -> bool:
    try:
        p = urlparse(url)
        return p.scheme in ("http", "https") and bool(p.netloc)
    except Exception:
        return False
```

Файл train_and_save.py

```
import os, argparse, joblib
import pandas as pd, numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from features_v2 import extract_features_from_url  # не потрібен на тренуванні, але можна використовувати
from features_v2 import FEATURE_NAME_ALIASES

def main(dataset_path, out_dir):
    os.makedirs(out_dir, exist_ok=True)
    df = pd.read_csv(dataset_path)
    if 'status' in df.columns and 'label' not in df.columns:
        df ['label'] = df ['status'].map({'legitimate':0,'phishing':1})
    if 'label' not in df.columns:
        raise ValueError("Dataset must contain 'label' or 'status' column.")
    X = df.drop(columns= ['label','url'], errors='ignore').select_dtypes(include= [np.number])
    y = df ['label']

    rf_base = RandomForestClassifier(n_estimators=200, random_state=42, n_jobs=-1)
    rf_base.fit(X.fillna(0), y)
    importances = pd.Series(rf_base.feature_importances_, index=X.columns).sort_values(ascending=False)
    top_features = importances.head(50).index.tolist()
    joblib.dump(top_features, os.path.join(out_dir, 'feature_order.joblib'))
    print("Saved feature_order:", len(top_features))

    scaler = StandardScaler().fit(X [top_features].fillna(0))
    Xs = scaler.transform(X [top_features].fillna(0))
```

```
    rf_final = RandomForestClassifier(n_estimators=300, random_state=42, n_jobs=-1)
    rf_final.fit(Xs, y)

    joblib.dump(rf_final, os.path.join(out_dir, 'final_rf.joblib'))
    joblib.dump(scaler, os.path.join(out_dir, 'final_scaler.joblib'))
    types = {}
    for c in top_features:
        uniques = df [c].dropna().unique()
        types [c] = 'binary' if len(uniques) <= 2 else 'numeric'
    meta = {'order': top_features, 'types': types}
    joblib.dump(meta, os.path.join(out_dir, 'feature_metadata.joblib'))

    print("Training complete. Models saved to", out_dir)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument('--dataset', type=str, required=True, help='path to dataset CSV')
    parser.add_argument('--out', type=str, default='models', help='output models dir')
    args = parser.parse_args()
    main(args.dataset, args.out)
```

Файл styles.css

```
.phish-overlay {
  position: fixed !important;
  inset: 0 !important;
  z-index: 2147483647 !important;
  background: rgba(6, 6, 10, 0.86) !important; /* stronger dim */
  display: flex !important;
  align-items: center !important;
  justify-content: center !important;
  font-family: "Segoe UI", Roboto, system-ui, sans-serif !important;
  backdrop-filter: blur(6px) !important;
  -webkit-font-smoothing: antialiased !important;
  -moz-osx-font-smoothing: grayscale !important;
}

html.phish-block-scroll,
body.phish-block-scroll {
  overflow: hidden !important;
}

.phish-toast {
  position: fixed;
  right: 24px;
  bottom: 28px;
  z-index: 2147483650 !important;
  background: linear-gradient(90deg, #7b61ff, #00ffff);
  color: #000;
  font-weight: 700;
  font-size: 15px;
  padding: 12px 18px;
  border-radius: 12px;
  box-shadow: 0 0 20px rgba(0,255,255,0.6), 0 0 40px rgba(123,97,255,0.5);
```

```
  border: 2px solid rgba(255,255,255,0.18);
  text-shadow: 0 0 6px rgba(255,255,255,0.6);
  backdrop-filter: blur(6px);
  animation: toastIn 0.35s ease-out;
  transition: opacity 0.25s ease, transform 0.2s ease;
}
.phish-toast:hover {
  transform: scale(1.04);
  box-shadow: 0 0 28px rgba(0,255,255,0.8), 0 0 56px rgba(123,97,255,0.8);
}
.phish-toast.neon-toast {
  background: linear-gradient(90deg, rgba(123,97,255,0.95), rgba(0,255,255,0.95));
  color: #fff;
  text-shadow: 0 0 6px rgba(255,255,255,0.9);
}

.phish-box {
  width: min(720px, 95%) !important;
  max-width: 1020px !important;
  border-radius: 16px !important;
  padding: 24px !important;
  box-shadow: 0 10px 50px rgba(5,5,12,0.85), 0 0 80px rgba(123,97,255,0.12) !important;
  background: linear-gradient(180deg, rgba(15,10,35,0.98), rgba(22,9,50,0.96)) !important;
  color: #eaf2ff !important;
  text-align: center !important;
  border: 1px solid rgba(123,97,255,0.18) !important;
  transform-origin: center !important;
  animation: pop 0.45s cubic-bezier(.2,.9,.3,1) !important;
  backdrop-filter: blur(8px) !important;
  -webkit-backdrop-filter: blur(8px) !important;
  overflow: visible !important;
}

.phish-box.phishing.neon {
  border: 1px solid rgba(0,255,255,0.10) !important;
  box-shadow: 0 0 48px rgba(123,97,255,0.18) !important, inset 0 1px 0 rgba(255,255,255,0.02) !important;
}

/* loading smaller box */
.loading-box {
  width: 420px !important;
  padding: 26px !important;
  background: linear-gradient(180deg, rgba(0,0,0,0.55), rgba(12,4,30,0.64)) !important;
  border-radius: 12px !important;
}
.phish-box .icon {
  font-size: 52px !important;
```

```
  margin-bottom: 8px !important;
  filter: drop-shadow(0 6px 16px rgba(255,255,0,0.6)) !important;
}

.phish-box h2 {
  font-size: 28px !important;
  font-weight: 800 !important;
  color: #ffffff !important;
  letter-spacing: 0.6px !important;
  margin: 6px 0 12px !important;
  text-shadow:
    0 0 10px rgba(0,255,255,0.6),
    0 0 22px rgba(123,97,255,0.45),
    0 6px 18px rgba(0,0,0,0.45);
}
.phish-box .muted,
.phish-box p.muted {
  font-size: 15px !important;
  line-height: 1.45 !important;
  color: rgba(220,230,255,0.82) !important;
  margin: 6px 0 0 !important;
}
.phish-box .prob {
  margin-top: 8px !important;
  font-weight: 700 !important;
  color: #dfeeff !important;
  font-size: 15px !important;
}


.features {
  background: linear-gradient(180deg, rgba(255,255,255,0.04), rgba(255,255,255,0.01)) !important;
  border: 1px solid rgba(255,255,255,0.06) !important;
  padding: 14px !important;
  margin: 16px auto 12px !important;
  border-radius: 10px !important;
  text-align: left !important;
  color: #ffeefa !important;
  max-height: 200px !important;      overflow-y: auto !important;
  font-size: 14px !important;
  box-shadow: inset 0 1px 0 rgba(255,255,255,0.02) !important;
}
.features h3 { margin: 0 0 8px; font-size: 15px; color: #fffbef; }
.features ul { list-style: none; margin: 0; padding: 0; }
.features li { margin: 8px 0; font-size: 14px; color: #ffdfe8; line-height: 1.4; }
.features .feat { color: #ffcfdf; font-weight: 800; margin-right: 10px; }
.buttons {
```

```
  margin-top: 16px !important;
  display: flex !important;
  gap: 12px !important;
  justify-content: center !important;
  flex-wrap: wrap !important;
}

.buttons button {
  padding: 12px 18px !important;
  border-radius: 12px !important;
  border: none !important;
  cursor: pointer !important;
  font-weight: 800 !important;
  font-size: 14px !important;
  transition: transform .14s ease, box-shadow .14s ease !important;
}

.btn-primary {
  background: linear-gradient(90deg, #7b61ff, #00d4ff) !important;
  color: #04060a !important;
  box-shadow: 0 12px 36px rgba(123,97,255,0.18), 0 6px 18px rgba(0,212,255,0.12) !important;
  border: 1px solid rgba(255,255,255,0.06) !important;
}
.btn-primary:hover { transform: translateY(-3px) !important; box-shadow: 0 18px 48px rgba(123,97,255,0.22) !important; }

.btn-secondary {
  background: transparent !important;
  border: 1px solid rgba(255,255,255,0.06) !important;
  color: #dfeaff !important;
}
.btn-secondary:hover { transform: translateY(-3px) !important; box-shadow: 0 8px 26px rgba(0,0,0,0.45) !important; }

.btn-ghost {
  background: transparent !important;
  color: #9bdcff !important;
  border: 1px dashed rgba(155,220,255,0.12) !important;
}

.footer { margin-top: 12px; font-size: 12px; color: rgba(220,230,255,0.6); }

@keyframes pop {
  from { transform: translateY(12px) scale(0.98); opacity: 0; }
  to { transform: translateY(0) scale(1); opacity: 1; }
}
@keyframes toastIn {
  from { opacity: 0; transform: translateY(8px); }
  to { opacity: 1; transform: translateY(0); }
```

```
}

@media (max-width: 620px) {
  .phish-box { padding: 20px !important; border-radius: 12px !important; width: calc(100% – 28px) !important; }
  .features { max-height: 160px !important; font-size: 13px !important; }
  .phish-box h2 { font-size: 20px !important; }
  .phish-box .icon { font-size: 48px !important; }
}

@media (max-width: 420px) {
  .phish-toast { right: 12px; left: 12px; bottom: 18px; font-size: 14px; padding: 10px 14px; }
}
```

Додаток В
Ілюстраційна частина

МЕТОД І ЗАСІБ ДЛЯВИЯВЛЕННЯ ФІШИНГОВИХ САЙТІВ ІЗ ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

Виконав: студент 2 курсу, групи 1БС-24 м
спеціальності 125 Кібербезпека та захист
інформації
_____ Максим ГНАТЮК
Керівник: к. т. н., доцент каф. ЗІ
_____ Володимир ГАРНАГА
« 19 » 12 2025 р.

ВІЗУАЛІЗАЦІЯ 50 НАЙВАЖЛИВІШИХ ОЗНАК

№	Ознака	Важливість	Короткий опис
1	**google_index**	0.166	Показує, чи сторінка індексується Google; фішингові сайти зазвичай не індексуються.
2	**page_rank**	0.106	Рейтинг сторінки у пошукових системах; низький ранг свідчить про недовіру до ресурсу.
3	**nb_hyperlinks**	0.081	Загальна кількість гіперпосилань на сторінці; легітимні сайти мають більше внутрішніх посилань.
4	**web_traffic**	0.072	Рівень відвідуваності сайту; фішингові домени зазвичай мають мінімальний трафік.
5	**nb_www**	0.045	Кількість входжень «www» у домені або посиланнях, надлишок часто вказує на підробку.
6	**domain_age**	0.037	Вік домену у днях або місяцях; молоді домени, типовий признак фішингу.
7	**ratio_extHyperlinks**	0.032	Частка зовнішніх посилань від усіх гіперпосилань; велика частка може вказувати на фішингову сторінку.
8	**phish_hints**	0.028	Ознаки фішингу у вихідному коді (форми, скрипти, ключові слова).
9	**longest_word_path**	0.027	Довжина найдовшого слова у шляху URL; довгі або беззмістовні шляхи часто є підозрілими.
10	**ratio_intHyperlinks**	0.026	Частка внутрішніх посилань; низьке значення, показник шахрайських сайтів.
11	**safe_anchor**	0.023	Частка безпечних (валідних) анкорів; зменшення свідчить про підроблену структуру.
12	**ratio_extRedirection**	0.018	Кількість зовнішніх перенаправлень; фішингові сторінки часто використовують переадресації.
13	**length_url**	0.017	Загальна довжина URL; занадто довгі посилання типові для фішингу.
14	**ratio_digits_url**	0.017	Частка цифр у посиланні; числові символи часто приховують справжній домен.
15	**longest_words_raw**	0.016	Найдовше слово у тексті сторінки; довгі випадкові токени вказують на автогенерацію.
16	**length_words_raw**	0.016	Середня довжина слів на сторінці; у фішингу часто зустрічаються короткі шаблонні тексти.
17	**char_repeat**	0.015	Повторюваність символів у домені чи шляху; подвійні літери або цифри, ознака імітації бренду.
18	**length_hostname**	0.015	Довжина імені хоста; надмірна довжина характерна для підробок.
19	**avg_word_path**	0.015	Середня довжина слів у шляху URL; допомагає відрізняти згенеровані структури.

ВІЗУАЛІЗАЦІЯ 50 НАЙВАЖЛИВІШИХ ОЗНАК

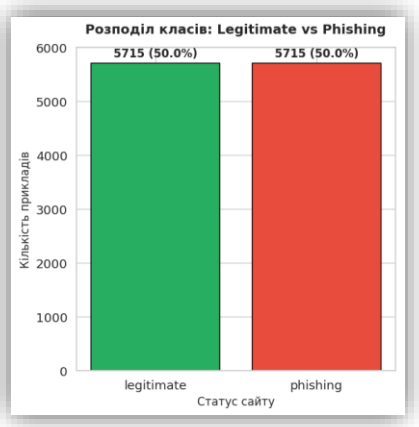
20	**links_in_tags**	0.014	Кількість гіперпосилань у тегах; перевищення норми вказує на маскування контенту.
21	**shortest_word_host**	0.013	Найкоротше слово в домені; занадто короткі елементи, спроба приховати справжню адресу.
22	**domain_registration_length**	0.013	Тривалість реєстрації домену; короткі терміни властиві тимчасовим сайтам.
23	**domain_in_title**	0.012	Чи збігається домен із назвою сайту; невідповідність, часта фішингова ознака.
24	**nb_slash**	0.012	Кількість «/» у URL; багато рівнів вкладеності, ознака маскування.
25	**shortest_word_path**	0.011	Найкоротше слово у шляху; дає уявлення про структуру URL.
26	**nb_dots**	0.011	Кількість крапок у домені; велика кількість субдоменів часто використовується у фішингу.
27	**ratio_digits_host**	0.011	Частка чисел у домені; числові домени частіше фішингові.
28	**avg_words_raw**	0.010	Середня кількість слів у тексті сторінки; низьке значення свідчить про шаблонність.
29	**nb_hyphens**	0.009	Кількість дефісів у домені; використовується для створення схожих назв (наприклад, pay-pal.com).
30	**avg_word_host**	0.008	Середня довжина слів у домені; допоміжна структурна ознака.
31	**nb_qm**	0.008	Кількість знаків «?» у URL; наявність параметрів може вказувати на шкідливі запити.
32	**nb_eq**	0.008	Кількість символів «=» у URL; використовується у фішингових форм-запитах.
33	**longest_word_host**	0.007	Найдовше слово в домені; вказує на аномальну складність імені.
34	**shortest_words_raw**	0.007	Короткі слова в тексті сторінки; багато коротких, ознака автозгенерованого контенту.
35	**domain_with_copyright**	0.006	Чи містить сайт копірайт; відсутність, сигнал недовіри.
36	**ratio_extErrors**	0.006	Частка помилкових зовнішніх посилань; фішингові сайти часто мають “биті” лінки.
37	**ip**	0.005	Наявність IP-адреси замість домену; типова риса шкідливих сторінок.
38	**ratio_extMedia**	0.005	Частка зовнішніх медіа-ресурсів (зображень, відео); надлишок аномальний вміст.
39	**ratio_intMedia**	0.004	Частка внутрішніх медіа; відсутність може свідчити про скопійований контент.
40	**nb_extCSS**	0.004	Кількість зовнішніх CSS-файлів; надлишок можливість прихованих елементів.

ВІЗУАЛІЗАЦІЯ 50 НАЙВАЖЛИВІШИХ ОЗНАК

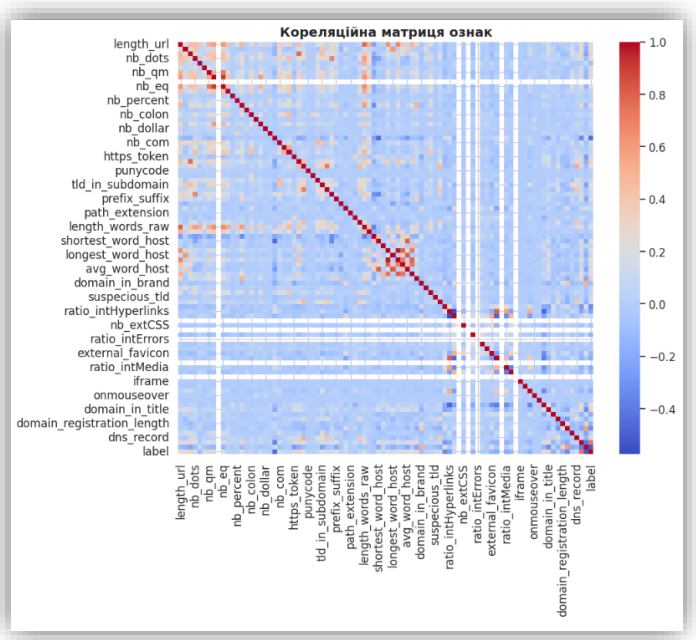
41	**nb_subdomains**	0.004	Кількість субдоменів; їх багато, часто використовують для маскування.
42	**domain_in_brand**	0.004	Чи містить домен назву бренду; іноді використовується для підробки (наприклад, paypal-secure.com).
43	**nb_redirection**	0.004	Кількість перенаправлень; надлишок спроба заплутати користувача.
44	**shortening_service**	0.003	Використання скорочувачів посилань (bit.ly тощо); приховує справжню адресу.
45	**nb_underscore**	0.003	Кількість підкреслень «_» у URL; часто використовуються у скам-структурах.
46	**prefix_suffix**	0.002	Наявність префіксів/суфіксів у домені; додають схожість до відомих брендів.
47	**https_token**	0.002	Використання слова “https” у тілі домену типова маскувальна техніка.
48	**nb_and**	0.002	Кількість символів «&»; багато параметрів у запиті підозріло.
49	**external_favicon**	0.002	Іконка сайту розміщена зовні домену; часто копіюється з іншого ресурсу.
50	**empty_title**	0.002	Відсутність заголовка сторінки; типовий дефект фішингових шаблонів.

АНАЛІЗ НАБОРУ ДАНИХ

Розподіл класів “Legitimate” vs “Phishing”

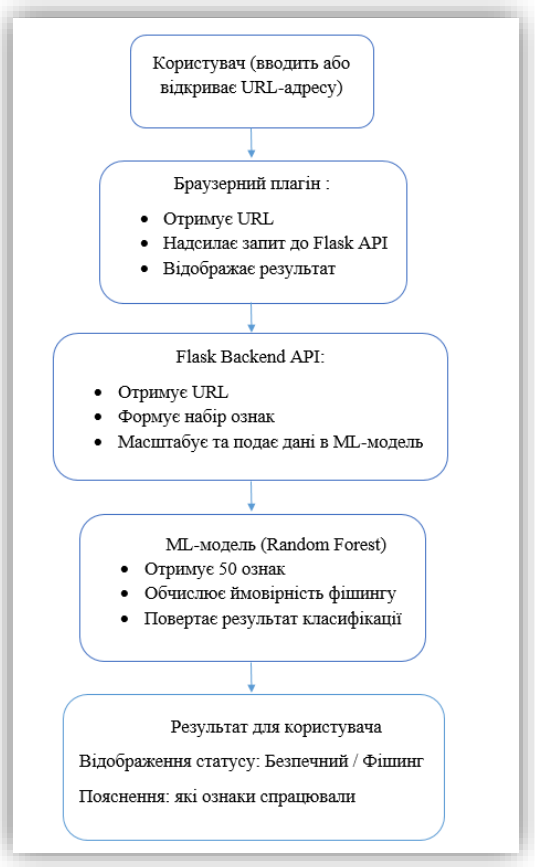


Кореляційна матриця ознак



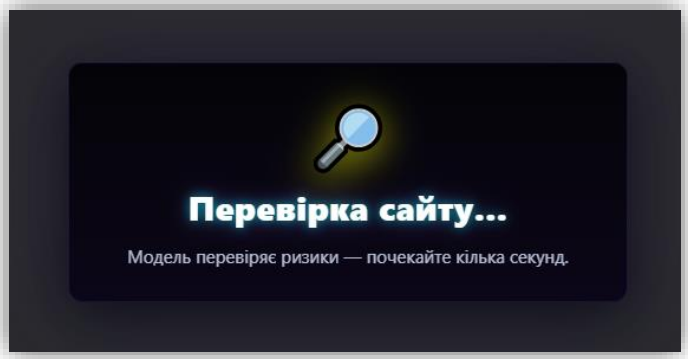
СХУМА АРХІТЕКТУРИ ПРОГРАМНОГО ЗАСОБУ

Архітектура системи “Phishing Detector”

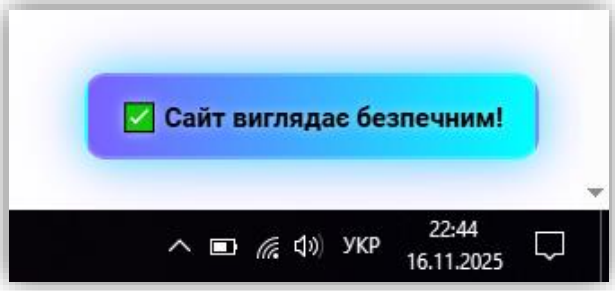


ВІЗУАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСОБУ

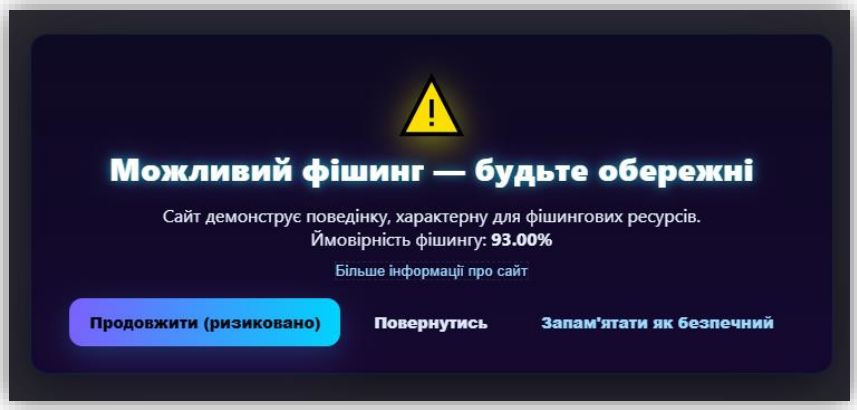
Overlay-вікно “Перевірка сайту…” під час аналізу URL



Повідомлення про безпечний сайт після перевірки

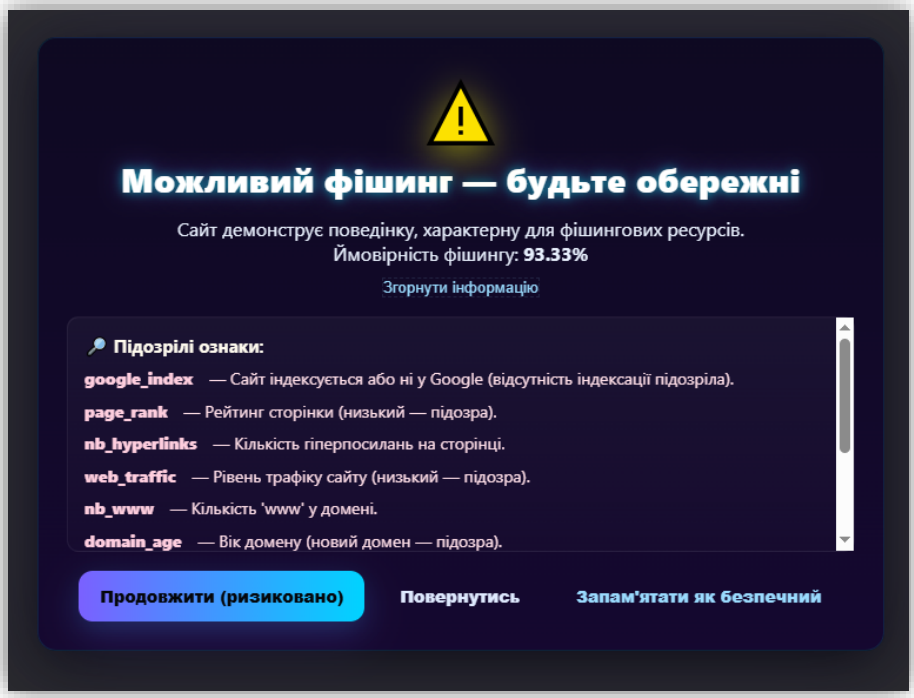


Попередження про можливий фішинг (згорнута інформація)

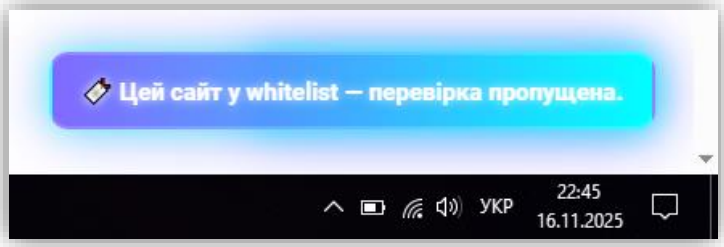


ВІЗУАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСОБУ

Розгорнутий перелік підозрілих ознак сайту



Повідомлення про пропуск перевірки для сайтів у whitelist

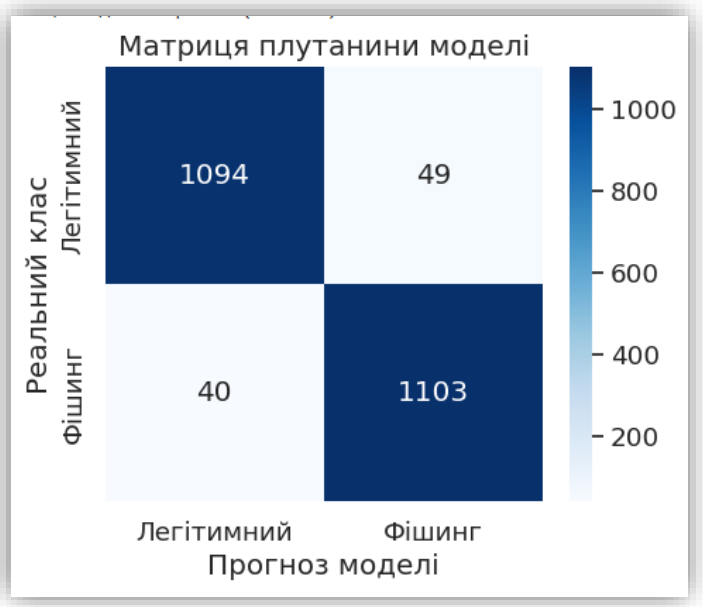


АНАЛІЗ ПРОГРАМНОГО ЗАСОБУ

Приклади передбачень для тестових сайтів

URL-адреса	Ймовірність фішингу (%)	Рішення	Ключові спрацьовані ознаки
https://www.google.com	1.24	Безпечний	-
https://www.wikipedia.org	3.05	Безпечний	-
http://paypal-login-verification.com	76.67	Фішинг	google_index, domain_age, web_traffic
http://amazon-login-update-payment.xyz	82.41	Фішинг	page_rank, phish_hints, domain_in_title

Матриця плутанини моделі Random Forest



ПОКАЗНИКИ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ

Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки.

Критерії	Експерт (ПІБ, посада)		
	Гарнага В. А.	Долюк Д. П.	Войтович О. П.
	Бали:		
1. Технічна здійсненність концепції	3	3	3
2. Ринкові переваги (наявність аналогів)	2	4	3
3. Ринкові переваги (ціна продукту)	3	4	3
4. Ринкові переваги (технічні властивості)	3	4	4
5. Ринкові переваги (експлуатаційні витрати)	3	4	3
6. Ринкові перспективи (розмір ринку)	3	4	4
7. Ринкові перспективи (конкуренція)	3	4	3
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	3	4	4
10. Практична здійсненність (необхідність нових матеріалів)	3	3	4
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів	42	43	44
Середньоарифметична сума балів $СБ_c$	**43**		