Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

# КВАЛІФІКАЦІЙНА РОБОТА

Галицького Володимира Володимировича

на здобуття ступеня вищої освіти Бакалавра

Система виявлення шкідливих вебсайтів на основі аналізу URL-адрес

Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітня програма Кібербезпека

Шифр КРБКБ.2102143.21.02.06 ПЗ

Виконав студент 4 курсу група КБ-21-2 Володимир ГАЛИЦЬКИЙ
Керівник д.т.н., професор Михайло КАСЯНЧУК
Нормоконтролер старший викладач Сергій МОСТОВИЙ

До захисту допускаю:
Завідувач кафедри кібербезпеки Юрій КЛЬОЦ

2 06 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Кібербезпеки

Рівень вищої освіти Бакалавр

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека

Освітня програма Кібербезпека

ЗАТВЕРДЖУЮ

Завідувач кафедри кібербезпеки

Юрій КЛЬОЦ

15 лютого 2025 р.

**ЗАВДАННЯ**
**НА КВАЛІФІКАЦІЙНУ РОБОТУ**
Галицького Володимира Володимировича

1 Тема роботи Система виявлення шкідливих вебсайтів на основі аналізу URL-адрес

Керівник роботи д.т.н., професор Михайло Касянчук

Затверджено наказом ректора університету від 07 лютого 2025 № 23

2 Строк подання студентом кваліфікаційної роботи на кафедру 1.06.2025

3 Вихідні дані до роботи Проаналізувати існуючі підходи до виявлення шкідливих вебсайтів. Розглянути особливості побудови URL адрес та їхній вплив на ідентифікацію загроз. Визначити критерії за якими можна класифікувати URL як шкідливі або безпечні. Обрати та реалізувати оптимальну модель машинного навчання для класифікації URL. Розробити та інтегрувати систему в браузерне розширення. Провести тестування для оцінки точності та ефективності системи.

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити) Проаналізувати існуючі підходи до виявлення шкідливих вебсайтів. Розглянути особливості побудови URL адрес та їхній вплив на ідентифікацію загроз. Визначити критерії за якими можна класифікувати URL як шкідливі або безпечні. Обрати та реалізувати оптимальну модель машинного навчання для класифікації URL. Розробити та інтегрувати систему в браузерне розширення. Провести тестування для оцінки точності та ефективності системи.

5 Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Загальна архітектура функціонування системи. Архітектура процесу навчання моделі Random Forest. Архітектура процесу виявлення шкідливих вебсайтів.

6 Консультанти розділів кваліфікаційної роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| | | | |

7 Дата видачі завдання 16 лютого 2025 р.

## КАЛЕНДАРНИЙ ПЛАН

| Назва етапів (розділів) кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|
| Вибір і затвердження теми кваліфікаційної роботи | 07.02.2025 | |
| Ознайомлення з предметною областю | 10.02.2025 | |
| Дослідження існуючих рішень | 26.02.2025 | |
| Постановка задачі | 06.03.2025 | |
| Визначення загальних принципів рішення задачі | 18.03.2025 | |
| Деталізація принципів рішення задачі | 14.04.2025 | |
| Розробка політик експлуатації і безпеки | 24.04.2025 | |
| Оформлення пояснювальної записки згідно вимог | 04.05.2025 | |
| Оформлення графічної частини | 29.05.2025 | |
| Захист КР | 05.06.2025 | |

Студент _____ Володимир ГАЛИЦЬКИЙ

Керівник кваліфікаційної роботи _____ Михайло КАСЯНЧУК

# АНОТАЦІЯ

Тема кваліфікаційної роботи: Система виявлення шкідливих вебсайтів на основі аналізу URL-адрес.

Автор роботи: Галицький Володимир Володимирович.

Керівник роботи: д.т.н., професор Касянчук Михайло Миколайович.

Пояснювальна записка: 62 с., 1 додаток, 18 рисунків, 2 таблиці, 40 джерел.

Графічна частина: 3 плакати.

СИСТЕМА ВИЯВЛЕННЯ ШКІДЛИВИХ ВЕБСАЙТІВ НА ОСНОВІ АНАЛІЗУ URL-АДРЕС.

Кваліфікаційна робота присвячена створенню ефективної системи виявлення шкідливих вебсайтів на основі аналізу URL-адрес. Основна мета полягає у розробці рішення, яке дає змогу виявляти потенційно небезпечні ресурси та попереджати користувачів про можливі загрози. У роботі розглянуто сучасні підходи до аналізу посилань, зокрема чорні списки, евристичні методи та машинне навчання. Проаналізовано наявні системи виявлення шкідливих сайтів, а також їхні сильні та слабкі сторони.

Основна частина присвячена побудові моделі машинного навчання з використанням алгоритму Random Forest. Для цього здійснено попередню обробку даних, обрано ключові ознаки та проведено навчання моделі на наборі URL-адрес. Створене рішення інтегровано у браузерне розширення для Google Chrome, що дає змогу автоматично перевіряти вебсайти в реальному часі. У результаті розроблено систему, яка виявляє шкідливі вебсайти з високою точністю, покращуючи безпеку користувачів під час перегляду. Отримані результати підтверджують ефективність обраного підходу та можливість його практичного застосування.

29.05.2025 _____

# ABSTRACT

Subject of qualification work: Complex system of information protection of the automated workplace of the head of the enterprise.

Author: Volodymyr Volodymyrovych Halitsky KB-21-2.

Head of work: d.t.s., professor Mykhailo Mykolayovych Kasyanchuk.

Explanatory note: 62 p., 1 appendices, 18figures, 2 tables, 40 sources

Graphic part: 3 posters.

A SYSTEM FOR DETECTION OF MALIGNANT WEBSITES BASED ON URL ANALYSIS.

The qualification work is devoted to the creation of an effective system for detecting malicious websites based on URL analysis. The main goal is to develop a solution that allows you to detect potentially dangerous resources and warn users about possible threats. The work considers modern approaches to link analysis, in particular blacklists, heuristic methods and machine learning. The existing systems for detecting malicious sites, as well as their strengths and weaknesses, are analyzed.

The main part is devoted to building a machine learning model using the Random Forest algorithm. For this purpose, data preprocessing was carried out, key features were selected and the model was trained on a set of URL addresses. The created solution is integrated into a browser extension for Google Chrome, which allows you to automatically check websites in real time. As a result, a system was developed that detects malicious websites with high accuracy, improving user safety while browsing. The results obtained confirm the effectiveness of the chosen approach and the possibility of its practical application.

29.05.2025 _____

ЗМІСТ

Вступ ..... 7
1 Аналіз наявних рішень ..... 8
1.1 Шкідливі вебсайти та їх загрози ..... 8
1.2 Особливості URL адрес ..... 17
1.3 Аналіз наявних рішень ..... 20
1.4 Постановка задачі ..... 26
2 Розробка системи виявлення шкідливих вебсайтів ..... 27
2.1 Модель виявлення шкідливих вебсайтів ..... 27
2.2 Реалізація моделі машинного навчання та інтеграція в браузерне розширення ..... 30
2.3 Висновки до розділу ..... 44
3 Впровадження та оцінка достовірності системи ..... 45
3.1 Оцінка достовірності системи ..... 45
3.2 Тестування розширення ..... 48
3.3 Висновки до розділу ..... 56
Висновки ..... 58
Перелік джерел посилання ..... 59
Додаток А. Копія графічної частини ..... 63

# ВСТУП

З розвитком технологій всесвітньої мережі та збільшенням кількості інтернет-сторінок, тема кіберзахисту набирає обертів. Небезпечні вебсайти застосовують різноманітні методи, щоб ввести в оману користувачів, викрасти секретні дані або заразити пристрої шкідливими програмами. Одним із ключових способів боротьби цим загрозам є аналіз URL адрес для ідентифікації потенційно небезпечних ресурсів. У цій кваліфікаційній роботі розглядається проблема шкідливих вебсайтів та їх впливу на безпеку користувачів. Аналізуються методи виявлення шкідливих вебсайтів на основі аналізу URL адрес, включаючи використання евристичних алгоритмів, чорних списків, машинного навчання та інших підходів. Також досліджуються існуючі розширення для браузерів, що забезпечують подібний функціонал, їхні переваги та недоліки. Основна мета цього дослідження є розробка дієвої системи виявлення шкідливих вебсайтів, ґрунтуючись на аналізі URL адрес, у вигляді розширення для браузера Google Chrome. Система повинна бути точною, швидкою та зручною у використанні, забезпечуючи користувачів своєчасним попередженням про потенційні загрози.

Для досягнення поставленої мети у кваліфікаційній роботі розглядається процес створення системи, яка має на меті виявляти шкідливі вебсайтів. Її реалізація передбачає аналіз структури посилань, а також використання алгоритмів машинного навчання з метою їх класифікації. Для оцінки ефективності створеної системи проведено тестування на реальних даних. Усі отримані результати розробки та досліджень подано з усіма подробицями у кваліфікаційній роботі. Зокрема, там викладено архітектуру створеного розширення, алгоритм, що використовується для аналізу URL адрес, результати проведеного тестування та оцінку загальної ефективності розробленої системи.

# 1 АНАЛІЗ НАЯВНИХ РІШЕНЬ

## 1.1 Шкідливі вебсайти та їх загрози

Шкідливі вебсайти є основою загроз в інтернет-просторі. Із збільшенням кількості користувачів інтернету зросла і кількість зловмисних сайтів. Такі вебсайти здатні заподіяти значну шкоду, що впливає не лише на захищеність особистої інформації користувача, але й безпеку його гаджетів та мережевого з'єднання загалом. Однією з головних небезпек, які пов'язані з шкідливими сайтами, є їх здатність запускати різні форми зловмисного програмного забезпечення, яке потрапляє на комп'ютер або мобільний пристрій користувача без його відома. Це можуть бути віруси, трояни, шкідливі скрипти або програми-вимагачі, які інфікують систему. Віруси та інші шкідливі програми здатні спричинити втрату особистих даних, фінансів, а також завдати шкоди функціонуванню пристроїв, що веде до значних витрат часу та ресурсів на усунення наслідків.

Окрім прямих збитків для користувача, зловмисні вебсайти слугують інструментом для здійснення фішингових атак. Фішингом є шахрайство, при якому користувачів обманним шляхом змушують надати свої особисті дані, такі як паролі, номери кредитних карток, дані про банківські рахунки та іншу конфіденційну інформацію. Шкідливі сторінки нерідко маскуються під популярні фінансові установи, онлайн-магазини або соціальні платформи, копіюючи їхній зовнішній вигляд та інтерфейс з метою ввести в оману та здобути довіру користувача. Якщо користувач надає власну інформацію на подібному ресурсі, ці дані опиняються у шахраїв. Вони можуть вдатися до використання цих даних для викрадення коштів або втілення інших злочинних планів. Наприклад, на рисунку 1 представлено фішинговий сайт Binance (https://www.binance.com), який багато користувачів відвідали за посиланням яке розміщували в соціальних мережах, де обіцяли швидку вигоду. Щойно користувачі входять на сайт, зловмисники можуть вкрасти їхні облікові дані [1].

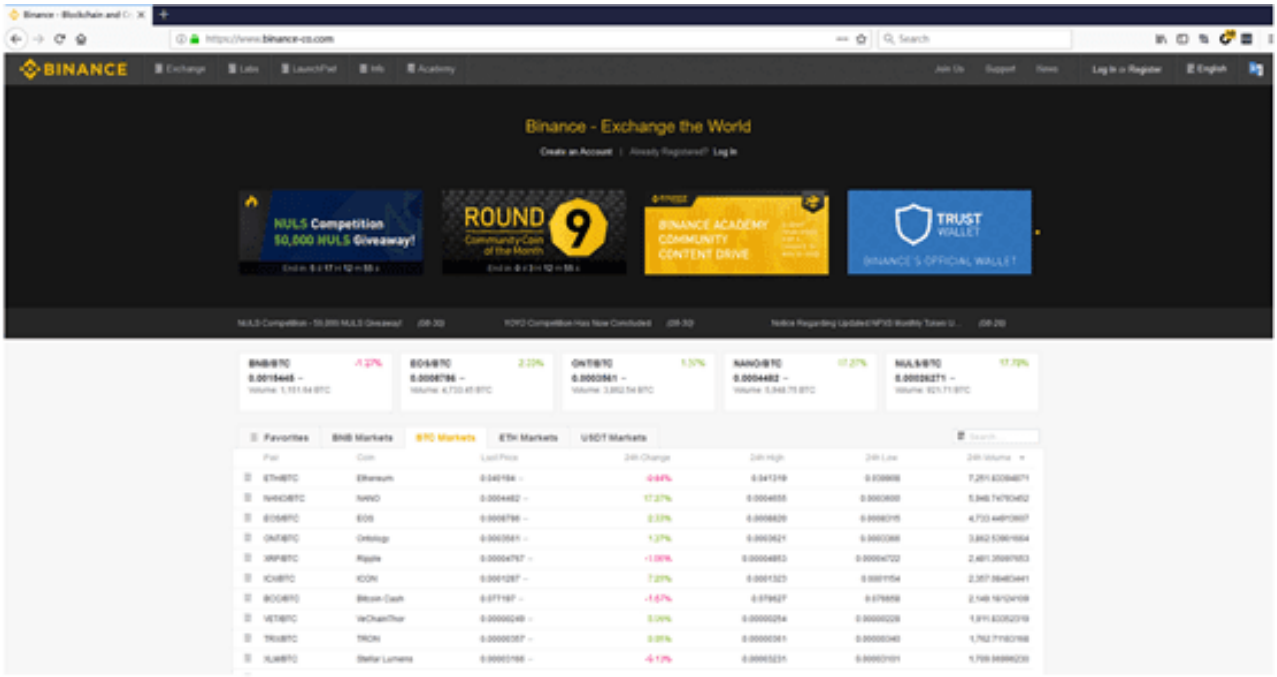



Рисунок 1 - Фішинговий сайт Binance

Атаки шляхом прихованого завантаження (drive-by downloads). У їхньому випадку зловмисні вебсайти самостійно завантажують та інсталюють шкідливе програмне забезпечення на пристрій користувача, не ставлячи його до відома, або ж пересилають cookies на сервер кіберзлочинця. Цей процес може бути дуже швидким і непомітним для користувача, що робить його особливо небезпечним. Найчастіше такі атаки здійснюються через застарілі або вразливі браузери, плагіни або інші компоненти програмного забезпечення, які не були оновлені [2]. Сучасні шкідливі сайти також можуть бути частиною бот-мереж, які використовуються для поширення спаму, здійснення атак типу відмова в обслуговуванні або інших шкідливих дій. Вебсайти можуть без відома користувача інфікувати його пристрій спеціальними скриптами, які перетворюють його комп'ютер на частину такої бот-мережі [3]. Приклад використання бот-мереж наведено на рисунку 2.

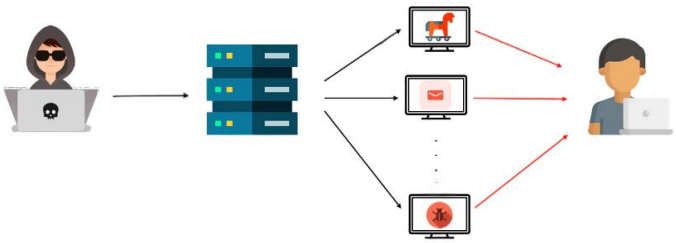


Рисунок 2 – Бот-мережа

Основним джерелом поширення шкідливих сайтів є нелегальні онлайн-ресурси, піратські платформи для скачування контенту, а також вебсайти, які мають вразливості, які дозволяють зловмисникам використовувати їх для поширення зловмисних програм. Також значною загрозою є атаки міжсайтового скриптингу (XSS), де зловмисники намагаються інтегрувати шкідливі скрипти безпосередньо у вебсайти, які переглядають інші користувачі. Такі атаки можуть використовуватися для крадіжки конфіденційної інформації, поширення шкідливого програмного забезпечення або захоплення сесії користувача. Вони виникають у випадках, коли вебсайт неналежним чином обробляє користувацькі дані і дозволяє виконання зловмисного коду іншими користувачами. Атака міжсайтового скриптингу зазвичай передбачає участь трьох основних сторін, серед яких зловмисник, вразливий вебдодаток і користувач. У типовому сценарії ця атака розгортається наступним чином:

- знаходження вразливості у вебсайта, що дозволяє впровадити шкідливі скрипти;

- виконання шкідливого скрипта який потім подається іншим користувачам, які відвідують вебсторінку;

- після запуску скрипта існує ризик викрадення облікових даних користувача, зокрема токенів сесії, або іншої конфіденційної інформації, введеної на скомпрометованій вебсторінці.

Наприклад якщо сайт не фільтрує введені дані, зловмисник може ввести свій скрипт у поле коментаря або пошукового запиту [4]. Ще однією загрозою є Експлойт набори, дані інструменти, які зловмисники застосовують для

виявлення вразливостей у популярних програмах, встановлених на пристрої користувача. Вони дозволяють здійснювати проникнення в систему, інсталюючи шкідливі компоненти без відома користувача. Завдяки таким наборам зловмисники можуть обійти захист і скористатися слабкими місцями програмного забезпечення з метою порушення безпеки [5].

Експлойти, як правило, працюють за конкретним алгоритмом. Спочатку відбувається пошук вразливостей, що реалізується шляхом сканування цільової системи. Метою якої є виявлення відомих недоліків безпеки у поширеному програмному забезпеченні, прикладом якого є веббраузери. Це дозволяє ідентифікувати, які саме системи можуть бути потенційними мішенями для атаки. Коли вразливість знайдено, на комп'ютер жертви доставляється зловмисне програмне забезпечення. Цей процес, здебільшого, протікає непомітно для тих, хто користується інтернетом. Зазвичай, його ініціюють через зламані або ж навмисно зроблені шкідливі вебсайти. Зловмисне програмне забезпечення створюється з метою експлуатувати конкретну, виявлену вразливість. Щоб уникнути виявлення, вони використовують різноманітні методи обфускації. Серед яких приховування коду або його шифрування, що ускладнює процес виявлення системами захисту.

За допомогою цих прийомів вони збільшують шанси на успішне вторгнення в систему. Після доставки та встановлення цього програмного забезпечення може відбуватися серія операцій, включаючи викрадення особистої інформації, шифрування файлів для подальшого вимагання викупу або відкриття задніх дверей для дистанційного доступу зловмисників. Також серйозною загрозою в наш час є зловмисна реклама (Malvertising). Цей вид атак поєднує в собі методи кіберзлочинів з можливостями використання реклами задля розповсюдження зловмисного програмного забезпечення через онлайн-оголошення. На відміну від звичайних способів розповсюдження, зловмисна реклама використовує розгалужену мережу рекламних платформ для впровадження зловмисного коду в легітимну рекламу на сайтах з гарною репутацією. Такий спосіб дає кіберзлочинцям охоплювати значно більшу

аудиторію, яка не очікує на небезпеки, збільшуючи потенційну дію їхніх атак [6].

Також існує загроза SQL-ін’єкції, яка є одним із найпоширеніших типів атак на бази даних. Вона дозволяє зловмиснику впливати на SQL-запити, що виконуються додатком. Така атака виникає коли вебсайт недостатньо перевіряє вхідні дані, що дає змогу зловмиснику вставляти шкідливі команди. Унаслідок цього можливий несанкціонована модифікація або видалення інформації. Атака SQL-ін’єкції відбувається шляхом введення зловмисних команд у поля, які програма використовує у SQL-запитах. Наприклад, замість очікуваного введення даних у форму, зловмисник може вставити зловмисний фрагмент коду, який змінить всю логіку запиту. Якщо вебсайт не застосовує параметризовані запити або не використовує екранування спеціальних символів, база даних може виконати небезпечну команду. Основні наслідки SQL-ін’єкції включають отримання доступу до конфіденційної інформації, зміну вмісту бази даних, створення бекдорів у системі, проведення атак на інші сервери та навіть повний контроль над сервером. Для захисту від подібних атак використовують параметризовані запити, попередньо підготовлені вирази, використання об'єктно-реляційного відображення а також обмеження привілеїв бази даних, щоб звести до мінімуму потенційну шкоду у разі компрометації системи [7].

Ще однією загрозою є HTML (HyperText Markup Language) ін’єкція, яка полягає у вставленні або зміні коду на сторінці з боку зловмисника. Це може призвести до підміни вмісту сторінки, крадіжки даних користувача або запуску шкідливих скриптів. Така атака зазвичай можлива у випадках, коли введені користувачем дані недостатньо фільтруються і безпосередньо відображаються на сторінці. Атака може включати додавання зловмисних посилань, фальшивих форм для збору особистих даних або навіть підміну вмісту сторінки для маніпуляції користувачем. Наприклад, якщо вебсайт дозволяє вводити коментарі без належної перевірки, зловмисник може вставити код, який створить фальшиву форму входу або кнопку, що перенаправляє користувача на шахрайський сайт. Наслідки HTML-ін’єкції можуть варіюватися від незначної

підміни контенту до серйозних загроз безпеці, включаючи крадіжку облікових даних або встановлення шкідливого програмного забезпечення через приховані елементи. Щоб запобігти HTML-ін’єкції, необхідно ретельно перевіряти введені дані, обмежувати використання небезпечних символів та застосовувати політику безпеки контенту, яка блокує виконання сторонніх скриптів у браузері [8].

Ще однією поширеною загрозою є атака через приховане натискання (clickjacking), яка полягає в тому, що зловмисник змушує користувача взаємодіяти з прихованими елементами вебсторінки, сам того не усвідомлюючи. Суть атаки полягає в тому, що легітимна сторінка завантажується у прихованому вікні, над яким розміщується елементи з іншим виглядом. Користувач має намір натиснути на безпечний об'єкт, але фактично запускає шкідливу операцію. Наприклад, здійснює покупку або змінює налаштування безпеки. Clickjacking може використовуватись для викрадення облікових даних, здійснення фінансових транзакцій без згоди користувача чи встановлення шкідливих розширень. Наприклад, користувач може бачити кнопку підтвердження на банківському сайті, але в реальності її натискання активує іншу дію часто це переказ грошей на рахунок шахрая [9].

Також часто використовуються приховані скрипти, відомі як кейлогери, які вбудовуються у вебсторінки і збирають дані з клавіатури користувача, включно з логінами, паролями та іншою конфіденційною інформацією. Зловмисники можуть впроваджувати їх через вразливості на вебсайті, шкідливі розширення або компрометацію сторонніх бібліотек, які використовує вебсайт. Кейлогери функціонують у фоновому режимі, фіксуючи кожне натискання клавіш або навіть роблячи знімки екрану. Наприклад, якщо сайт використовує скрипт від ненадійного постачальника, такий скрипт може таємно передавати введені користувачем дані на сервер зловмисника [10].

Фальшиві капча (CAPTCHA) сервіси є шахрайськими механізмами, що імітують перевірку "Я не робот", але насправді призначені для збору введених користувачем даних. Такі сервіси можуть з'являтися у вигляді підроблених форм на шкідливих вебсайтах або впроваджуватися в легітимні ресурси через

скомпрометовані скрипти. Зловмисники застосовують підроблені CAPTCHA, щоб отримати логіни та паролі, або для тренування ботів на проходження справжніх перевірок. Людина може ввести свої особисті дані у фальшиву форму, не підозрюючи, що цим передає інформацію зловмисникам [11].

Маніпуляції через соціальну інженерію є методом впливу на людей для викрадення конфіденційної інформації або спонукання їх до виконання певних дій. Одним із поширених прикладів такого підходу є шахрайство у формі підробленої технічної підтримки. Зловмисники створюють фальшиві вебсайти, які імітують офіційні джерела якогось популярного бренду або банківських установ. Вони можуть показувати повідомлення про "вірусне зараження" або "блокування акаунту", закликаючи користувача зателефонувати "в службу підтримки". Шахраї можуть переконати користувача встановити програму для дистанційного керування, під приводом усунення його проблеми, щоб здобути контроль над його комп’ютером [12].

Штучний інтелект все частіше ним користуються шахраї для створення реалістичного, але фальшивого контенту, що допомагає в обмані користувачів. Це можуть бути підроблені відгуки, фейкові новини, шахрайські чат-боти або навіть фейкові відео з відомими людьми, які нібито рекламують фінансові афери. Автоматично згенеровані тексти з використанням штучного інтелекту можуть повторювати стиль офіційного листа, вводячи користувачів в оману. Застосування AI-технологій для створення підроблених голосових або відео повідомлень дозволяє видавати себе за керівників компаній, що відкриває нові можливості для шахрайства на підприємствах [13].

Фальшиві оновлення програмного забезпечення, є методом кібератаки коли зловмисники створюють підроблені повідомлення про необхідність оновлення програм або драйверів. Такі оновлення можуть з’являтися у вигляді спливаючих вікон на вебсайтах, рекламних банерів або навіть як підроблені системні повідомлення. Метою цього методу є спонукання користувача до завантаження та встановлення програмного забезпечення, яке може виявитися вірусом, трояном або програмою-вимагачем. Часто шахраї поширюють

фальшиві оновлення через підроблені сторінки, які імітують офіційні сайти популярного програмного забезпечення, такого як текстовий редактор або антивіруси. Після завантаження цього оновлення користувач може непомітно встановити бекдор, що дасть зловмисникам доступ до його системи, або інструмент для викрадення персональних даних [14].

Атака людина посередині (MITM) являє собою різновид кібернетичного нападу, у ході якого зловмисник перехоплює та змінює дані, що передаються між взаємодіючими сторонами, не ставлячи їх до відома про свою присутність. Цей метод використовується для перехоплення таємної інформації, такої як імена користувачів, паролі, номери банківських карток чи особисті повідомлення. Поширеною форм атаки є перехоплення трафіку в громадських мережах. Зловмисник може створити підроблену точку доступу з назвою, схожою на легітимну наприклад, "Free_Coffee_WiFi", і підключаючи користувачів до цієї мережі, відстежувати весь їхній трафік. У разі відсутності захищеного протоколу, зловмисник може легко отримати логіни й паролі користувача з переданого трафіку [15].

Динамічний DNS (Dynamic Domain Name System), дана технологія, яка дає змогу змінювати відповідність домена різним IP-адресам в режимі реального часу. Це корисна функція для легальних цілей, наприклад, для забезпечення віддаленого доступу до домашніх серверів або камер відеоспостереження. Однак цю технологію також використовують для маскування шкідливих сайтів та уникнення блокування. Основний принцип використання DDNS у кіберзлочинності полягає в тому, що замість фіксованої IP-адреси шкідливий вебсайт постійно змінює своє розташування. Це дозволяє атакуючим уникати блокування, оскільки традиційні механізми безпеки часто покладаються на статичні списки. Крім того, DDNS дозволяє хакерам швидко розгортати нові домени після того, як попередні буде виявлено та заблоковано [16].

Скорочення URL, дана технологія дозволяє зменшити довгі вебадреси, створюючи короткі посилання через сервіси на кшталт Bit.ly або TinyURL. Такий підхід широко використовується для зручності в соціальних мережах та

месенджерах, проте він також став популярним серед зловмисників для маскування шкідливих сайтів. Основна небезпека таких посилань у тому, що користувач не може побачити, на який сайт він потрапить після натискання. Це робить їх надзвичайно ефективними знаряддями для фішингу, розповсюдження шкідливих програм, а також різноманітних шахрайських схем. Наприклад, шахраї можуть надсилати коротке посилання, яке виглядає безпечним, але перенаправляє жертву на сторінку, що імітує сайт банку або соціальної мережі для викрадення облікових даних [17].

Квішинг (QR phishing), дана техніка атаки при якій зловмисники використовують підроблені QR-коди для спрямування користувачів на фішингові вебсайти або заражені ресурси. Оскільки їх часто використовують для миттєвого отримання інформації, здійснення платежів або підтвердження особи, користувачі можуть не перевіряти адреси, на які вони ведуть, перш ніж їх відкрити. Їх можуть розміщувати у громадських місцях або відправляти у вигляді зображень через електронну пошту та месенджери. Захист від подібних атак передбачає перевірку посилань перед відкриттям, використання антивірусних рішень із підтримкою сканування QR-кодів та обмеження автоматичного виконання команд, які можуть бути закодовані [18].

Зловживання токенами інтерфейсу прикладного програмування (API) має місце, коли зловмисники здобувають неправомірний доступ до токенів аутентифікації. Ці токени застосовуються для взаємодії між сервісами та користувачами. Витік токенів можливий через розміщення коду у публічних репозиторіях, атаки на сервери, або фішингові кампанії. У випадку, коли токен скомпрометовано, зловмисники можуть отримати доступ до чутливих даних, змінювати конфігурації або видаляти інформацію [19].

Необхідність ефективного захисту від шкідливих вебсайтів стає все більш актуальною. Традиційно для виявлення таких сайтів використовуються бази даних зі списками відомих небезпечних ресурсів, проте цього недостатньо, оскільки кіберзлочинці постійно створюють нові домени і змінюють свої методи обману. Тому важливим напрямом у розробці систем захисту є створення

інтелектуальних механізмів виявлення, які використовують аналіз URL адрес, структури сайтів, а також поведінки користувачів для виявлення потенційно небезпечних ресурсів. Зважаючи на всі ці фактори, важливою є розробка системи виявлення шкідливих вебсайтів, яка забезпечує не лише точне виявлення загроз, а й своєчасну реакцію на них. Така система повинна враховувати різні типи атак, швидко реагувати на зміни в середовищі та інтегруватися з існуючими методами захисту для забезпечення найвищого рівня безпеки для користувачів.

## 1.2 Особливості URL адрес

URL адреса (Uniform Resource Locator) – це універсальний ресурсний ідентифікатор, що дає змогу користувачам та комп'ютерним системам знаходити та звертатися до ресурсів в інтернеті. Вона відіграє ключову роль у переміщені мережею, тому аналіз її структури і характеристик є важливим для забезпечення безпеки в інтернеті. Шкідливі посилання часто використовуються для фішингу, поширення вірусів або здійснення атак на користувачів, і їхнє виявлення є актуальною проблемою сучасної кібербезпеки [20]. Кожна URL адреса має чітко визначену структуру приклад наведено на рисунку 3.


Рисунок 3 – Структура URL адреси

Вона складається з кількох компонентів, які виконують різні функції. Основою будь-якого посилання є протокол доступу, наприклад, http або https. Сайти, які використовують захищене з'єднання (https), забезпечують

шифрування трафіку між сервером і користувачем, що створює додатковий рівень безпеки. Після протоколу слідує доменне ім’я, яке визначає сервер або ресурс, до якого здійснюється доступ. Наприклад, у адресі https://exam17.com, доменом є exam17.com. Домен може бути головним індикатором безпеки сайту, адже саме доменне ім’я може імітувати відомі бренди чи компанії для введення користувачів в оману.

Наступною частиною посилання є шлях, який вказує на конкретний ресурс на сервері. Наприклад, у https://exa538.com/log, шляхом є /log. Він часто використовується для організації доступу до окремих сторінок або функцій сайту. Шлях може бути доповнений параметрами запиту, які передають додаткову інформацію до сервера. Ці параметри розташовані після символу “?” і можуть мати вигляд, наприклад, ?id=647312&name=test294. Параметри дають змогу вебсайту гнучко коригувати наповнення або отримувати потрібну інформацію від відвідувача, але їх також можна використати для пересилання зловмисних даних. Останньою частиною адреси є фрагмент, що позначається символом решітки і використовується для вказівки на певний елемент на сторінці [21].

Шкідливі посилання розпізнаються за способом, яким вони збудовані та з яких даних складаються. Для того, щоб ввести в оману відвідувачів, нерідко застосовують домени, що вдають із себе справжні сайти. Це можливо реалізувати, використовуючи схожі символи. Наприклад, якщо замінити латинську літеру "o" на відповідний кириличний символ в слові "google", то утвориться домен, що зовні не відрізнятиметься від оригінального. Подібним чином шкідливі сайти можуть використовувати піддомени, щоб створювати ілюзію належності до авторитетних ресурсів. Наприклад, адреса secure-login.pay.com.attacker.com може ввести користувача в оману, оскільки справжній домен прихований у кінці [22].

Особливу увагу варто приділяти параметрам запиту, які можуть містити шкідливий код або перенаправляти користувача на небезпечні ресурси. Наприклад, у випадку з адресою http://exam078.com/?redirect=http://malicious-

site.com, користувач, переходячи за цим посиланням, буде автоматично перенаправлений на шкідливий сайт. Це один із поширених методів атак, що використовуються для приховування реального призначення посилання. Ще одним індикатором небезпеки є ключові слова, які можуть з'являтися у шляху або параметрах посилання. Наприклад, слова "login", "secure", "account", "verify" часто використовуються у фішингових атаках для створення ілюзії безпеки. Якщо такі слова зустрічаються у посиланні, пов'язаному з незнайомим доменом, це має викликати підозру. Адреса http://secure-log.examle.com може видатись переконливою, але уважно проаналізувати її будову, стає очевидним, що справжнім доменом є "example.com".

Шкідливі посилання також можуть бути надто довгими, що ускладнює їх перевірку. Довжина і складність адреси зазвичай створюється з метою приховування шкідливого контенту. Наприклад, адреса http://example.com/very/long/path/with/many/parameters?id=95&redirect=http://malicious-site.com приховує шкідливу частину після численних параметрів і шляхів. Такі URL є типовими для спаму або атак через електронну пошту. Кодовані символи в URL адресах, на зразок %20 чи %3F, здатні маскувати справжню будову посилання. Наприклад, у URL http://example.com/%2Fmacius%attack.com, закодовані символи приховують шкідливий контент, що ускладнює його виявлення. Важливим аспектом аналізу URL адрес є перевірка доменної зони. Наприклад, сайти, що використовують зони xyz, top, tk, часто асоціюються зі шкідливими ресурсами, оскільки їх вартість реєстрації є дуже низькою. Також використання доменної зони onion використовується для доступу до сайтів у мережі Tor, яка забезпечує анонімність і приховує місцезнаходження серверів та користувачів.

Даркнет-ресурси, часто асоціюються з незаконною діяльністю зокрема продажем викрадених даних, зброї, наркотиків, підроблених документів і послуг хакерів. Однак даркнет не обов'язково використовується лише в кримінальних цілях. Він також слугує засобом анонімного спілкування, особливо для журналістів, активістів та людей у країнах з жорсткою цензурою. Наприклад,

багато медіакомпаній, таких як The New York Times і BBC, мають таку версії своїх сайтів для безпечного доступу в умовах цензури [23]. SSL використовується для шифрування трафіку між користувачем і сервером, забезпечуючи конфіденційність переданих даних та створюючи візуальні ознаки захисту, які підвищують довіру до сайту. Проте зловмисники використовують SSL для створення шахрайських сайтів, щоб користувачі довіряли їм. Існують безкоштовні SSL-сертифікати, які можна отримати без верифікації домену. Шахраї часто реєструють домени, схожі на назви відомих сайтів, наприклад, із заміною символів, щоб змусити користувачів вводити конфіденційні дані, при цьому використовуючи справжні сертифікати безпеки для створення враження безпеки [24].

Серед найпоширеніших прикладів небезпечного використання посилань є випадки, що стосуються фішингових дій. Наприклад, під час відомої фішингової кампанії користувачі отримували електронні листи з посиланням на фальшивий сайт PayPal, що мав адресу http://secure.paypa1.com. Такі сайти копіюють зовнішній вигляд легальних сервісів, аби спонукати людей розкрити логіни, паролі та приватну інформацію. Ретельний аналіз структури URL адреси є важливим інструментом для виявлення шкідливих сайтів. Маніпуляції з адресами, нестандартні значення або порти часто сигналізують про ймовірну небезпеку ще на початковому етапі перевірки.

## 1.3 Аналіз наявних рішень

Визначення шкідливих вебсайтів є нагальною потребою через невпинне збільшення кібернетичних загроз. Шкідливі вебсайти можуть бути інструментами для крадіжки приватної інформації, організації фішингових кампаній або поширення вірусів серед користувачів. Для протидії цим загрозам розроблено велику кількість інструментів та методів, які використовують різні підходи до аналізу посилань та вмісту вебсайтів. Сучасні підходи до виявлення

шкідливих сайтів можна умовно розділити на кілька категорій. Одним із найчастіше застосовуваних методів є верифікація домену шляхом використання баз даних. Цей підхід полягає в порівнянні введеної адреси з базами вже відомих загрозливих інтернет-ресурсів. Хоча перевірка на основі відомих баз є швидкою, вона не здатна виявити нові загрози, які ще не були зареєстровані в системі.

Зазвичай, ці рішення можуть аналізувати вебсайт на предмет потенційної небезпеки або шкідливого контенту. Антивірусне програмне забезпечення використовує бази даних з переліками потенційно небезпечних вебсайтів, відстежує активність користувача та перевіряє домени в реальному часі. Наприклад, програмні продукти таких компаній, як Norton або Avast, містять функцію блокування доступу до вебсайтів, класифікованих як шкідливі. Проте, такі інструменти втрачають ефективність, якщо нещодавно створений загрозливий ресурс ще не включений у відповідні списки [25].

Браузерні розширення є ще однією популярною категорією інструментів для виявлення шкідливих сайтів. Такі інструменти активно відстежують активність користувача, допомагаючи запобігти доступу до небезпечних сайтів у реальному часі. До прикладу, таке браузерне розширення, як McAfee WebAdvisor здійснює перевірку посилання, що відкривається користувачем, і попереджає про потенційно небезпечний ресурс. Такі інструменти мають перевагу в тому, що вони є доступними та легкими в використанні. Однак їх ефективність залежить від алгоритмів, які використовуються для аналізу, та актуальності баз даних [26]. Іншим шляхом є дослідження вмісту вебсайту. Це передбачає оцінювання текстового та візуального наповнення сторінки, а також розбір структури коду. Наприклад, шкідливі ресурси можуть містити скрипти для прихованого майнінгу криптовалюти, або коди, які націлені на використання вразливостей у веббраузерах. Мінусом цього підходу є значне споживання ресурсів для аналізу сайту, а також тривалість самого процесу перевірки, що важливо у реальному часі [27].

Останніми роками великого поширення набули методи, які базуються на використанні машинного навчання. Ці підходи дозволяють автоматично

виявляти аномалії в посиланнях, використовуючи моделі, натреновані на великих обсягах даних. Наприклад, алгоритми можуть аналізувати довжину посилання, кількість доменів, частоту використання специфічних ключових слів або навіть поведінку користувачів на вебсторінці. Такий підхід є перспективним, оскільки дозволяє виявляти раніше невідомі загрози. Проте його впровадження вимагає значних ресурсів і часу на навчання моделей, а також періодичного оновлення даних для підтримки ефективності [28].

Також існує метод нульової довіри, дана концепція полягає в тому що жодному користувачу не довіряється за замовчуванням, навіть якщо вони знаходяться в межах однієї мережі. Всі запити, незалежно від їхнього походження, повинні бути перевірені перед наданням доступу до ресурсів. Серед переваг це суттєве зниження ризику атак із внутрішньої мережі та покращений контроль над тим, хто і до чого має доступ. До недоліків такого підходу належать технічна складність впровадження та потреба в безперервному адмініструванні й оновленні правил доступу [29].

Іншим ефективним методом є перевірка цифрових сертифікатів, які засвідчують автентичність сайту та шифрування з’єднання між користувачем і сервером. Це дозволяє виявити фальшиві або підроблені сертифікати, що можуть використовуватись для атак. Перевагою є покращення безпеки шляхом перевірки довіри до вебсайтів. Недоліком є те, що не всі сертифікати можуть бути автоматично перевірені, і часом складно розпізнати підроблені сертифікати без глибшого аналізу [30]. Виявлення нещодавно зареєстрованих доменів, даний підхід полягає в аналізі нових доменів для виявлення шкідливих ресурсів. Зловмисники часто використовують нові домени для проведення фішингових атак чи інших шахрайських схем, оскільки їх складно заблокувати за допомогою традиційних систем. Перевагою є своєчасне виявлення підозрілих ресурсів. Проте такий аналіз іноді може спричинити хибні тривоги, бо не всі нещодавно зареєстровані домени є шкідливими [31].

Блокування вебсайтів із перевищеним лімітом редиректів, даний підхід полягає в блокуванні вебсайтів, які здійснюють перевищення кількості

редиректів, оскільки часто це є ознакою зловмисних вебсайтів. Перевагою є простота і швидкість виявлення небезпечних ресурсів, які використовують перенаправлення для приховування справжнього місцезнаходження. Втім, надмірна кількість редиректів не завжди свідчить про небезпеку, оскільки в деяких випадках це необхідна частина функціоналу сайту, що може викликати помилкові блокування [32]. Аналіз відношення IP до місцезнаходження, даний підхід полягає у визначенні географічного місцезнаходження користувача або сервера за його IP-адресою. Цей метод допомагає виявити підозрілі або невідповідні місцезнаходження для певних дій, таких як аномальна активність або доступ до заблокованих ресурсів. Перевагою є можливість блокувати доступ з високих ризиків чи незвичних регіонів. Одним з головних недоліків при використанні цього методу є можливість приховування справжньої IP-адреси, що знижує точність і надійність аналізу [33].

Розпізнавання шкідливих доменів за допомогою кластеризації трафіку даний метод використовує алгоритми групування вебсайтів на основі схожих характеристик трафіку. Це дозволяє виявляти вебсайти, що мають схожі моделі поведінки. Перевагою є здатність автоматично виявляти нові типи шкідливих доменів без потреби в постійній ручній перевірці. Одним із недоліків залишаються труднощі в точному розпізнаванні через складність обробки трафіку у режимі реального часу [34]. Застосування VPN фільтрації для обмеження доступу до небезпечних сайтів, даний метод, який полягає у фільтрації та блокуванні доступу до небезпечних або шкідливих вебсайтів через VPN-сервіси. Перевага цього рішеня дозволяє приховати ідентифікаційні дані та мінімізувати ризики, пов’язані з цілеспрямованими атаками. Однак недоліком є те, що використання VPN може бути обхідним способом для зловмисників, які шукають анонімності [35].

Оцінювання надійності доменного імені з урахуванням його історії ґрунтується на аналізі попередньої активності. Такий підхід охоплює перегляд змін у реєстраторах або власниках, а також перевірку на присутність у списках небезпечних чи сумнівних серверів. Однією з переваг цього методу є здатність

виявляти потенційно шкідливі домени ще до того, як вони починають здійснювати атаки або іншу шкідливу діяльність. Водночас важливо враховувати, що ефективність такого підходу обмежується у випадках із новоствореними доменами, які ще не мають історії, тому системи можуть не ідентифікувати їх як загрозу [36].

Обмеження виконання JavaScript, є одним з найпоширеніших способів підвищення безпеки під час взаємодії з вебсайтами. Сенс цього методу полягає у повному блокуванні або ж у деталізованому контролі запуску скриптів на вебсайті. Це дає можливість мінімізувати ризики, пов’язані з атаками через скрипти, з кейлогерами та з іншими шкідливими техніками, що використовують можливості JavaScript. Цей підхід надзвичайно ефективний при використанні політик безпеки та спеціальних браузерних доповнень, таких як NoScript, які фільтрують небажані сценарії ще до моменту їхнього запуску. До позитивних аспектів цього рішення належить значне зменшення векторів атак, що ґрунтуються на виконанні шкідливого коду, що у свою чергу підвищує загальний рівень захищеності користувача. Водночас існують і певні недоліки, серед яких варто зазначити ймовірність некоректного функціонування окремих сайтів, особливо тих, що активно використовують JavaScript для побудови динамічного контенту, реалізації інтерактивних елементів інтерфейсу або завантаження інформації у фоновому режимі [37].

Використання honeypot-систем є одним із дієвих способів захисту інформаційних систем, що ґрунтується на створенні спеціальних пасток для кіберзлочинців. Пастки створюють вигляд активних систем, приваблюючи зловмисників до фальшивих об’єктів. Метою є збір даних про техніки порушників, що взаємодіють з обманними середовищами., а також інструменти, які використовуються в ході діяльності. Однією з головних переваг цього підходу є можливість виявлення нових типів атак ще до того, як вони завдадуть шкоди справжнім системам, що значно сприяє вдосконаленню засобів захисту. Вивчення методів атак сприяє побудові ефективних моделей захисту. Проте, існують і певні ризики, зокрема ймовірність того, що досвідчені зловмисники

зможуть розпізнати штучну природу системи або навіть використати її як плацдарм для подальшого проникнення у справжню інфраструктуру, якщо рівень ізоляції буде недостатнім [38].

Використання захищених DNS (Domain Name System) сервісів, є ефективним способом захисту конфіденційних даних під час виконання запитів до вебсайтів. Ці технології дозволяють шифрувати запити до серверів доменних імен, що суттєво зменшує ймовірність перехоплення або підміни переданої інформації зловмисниками. Завдяки шифруванню, атаки типу імітації, при яких підробляється відповідь сервера, стають менш ефективними або взагалі неможливими. Крім того, DNS забезпечує захист від стороннього моніторингу, що особливо важливо при підключенні до публічних та ненадійних мереж. До основних переваг такого підходу належить підвищення рівня конфіденційності, зменшення ризику витоку персональних даних, а також захист від втручання третіх сторін у процес резолюції доменних імен. Водночас слід враховувати і певні недоліки. Наприклад, додаткове шифрування DNS-трафіку може трішки впливати на швидкість обробки запитів, що особливо помітно на пристроях із обмеженими ресурсами або при повільному інтернет-з’єднанні. Також можливі ситуації, коли провайдери блокують або обмежують роботу сервісів, що може ускладнювати їх використання без додаткового налаштування або обходу обмежень [39].

Не зважаючи на велику кількість рішень, жоден не є цілком досконалим. Однією з основних проблем є висока кількість помилкових спрацьовувань, коли безпечні сайти позначаються як небезпечні. Через такі часті помилки користувачі можуть сумніватися в надійності систем безпеки. Іншою проблемою є недостатня ефективність багатьох інструментів у реальному часі, що ускладнює їх використання для захисту від швидкоплинних загроз. Крім того, більшість сучасних інструментів мають обмежену інтеграцію з браузерами. Втім, функціональні можливості таких інструментів обмежені, що знижує ефективність їх застосування для повного захисту.

У відповідь на це з'являється потреба впровадити інноваційні методи,

здатні гарантувати міцний захист, застосовуючи передові технології, засновані на машинному навчанні та інтегровані у середовищі браузера. Таким чином, існуючі інструменти здатні забезпечити базовий рівень захисту, але вони потребують подальшого вдосконалення для зниження помилкових спрацьовувань, підвищення ефективності у реальному часі та покращення інтеграції з іншими системами. Це створює можливості для розробки нових підходів, які враховуватимуть сучасні виклики у сфері кібербезпеки.

## 1.4 Постановка задачі

У межах цієї роботи ставиться завдання розробити ефективний та зручний у використанні інструмент для виявлення шкідливих вебсайтів у реальному часі, який працюватиме у вигляді браузерного розширення. Основна мета полягає у створенні системи, яка здійснюватиме перевірку посилання активної вебсторінки за допомогою моделі машинного навчання, заснованої на аналізі специфічних ознак вебсайта. При цьому щоб система працювала автономно, без обов'язкової залежності від зовнішніх серверів. Для досягнення поставленої цілі необхідно виконати наступні завдання:

- зібрати та опрацювати набір даних, що включає як легітимні, так і шкідливі вебсайти;
- сформувати набір ознак які дозволяють класифікувати сайти за поведінковими та структурними характеристиками;
- побудувати і навчити модель машинного навчання для класифікації вебсайтів;
- реалізувати браузерне розширення, яке інтегрує модель у середовище браузера;
- забезпечити UX/UI інтерфейс, який відображатиме результати перевірки у доступній формі.

# 2 РОЗРОБКА СИСТЕМИ ВИЯВЛЕННЯ ШКІДЛИВИХ ВЕБСАЙТІВ

## 2.1 Модель виявлення шкідливих вебсайтів

У сучасному цифровому середовищі питання виявлення шкідливих сайтів набуло особливої актуальності. Шкідливі ресурси щодня еволюціонують, застосовують нові техніки обману та намагаються обійти системи захисту. Саме тому автоматизовані засоби ідентифікації небезпечних вебресурсів мають бути достатньо гнучкими, точними й адаптивними. У межах цієї роботи було використано модель машинного навчання і набір ознак URL адрес, така система дозволяє отримати баланс між швидкістю прийняття рішень і точністю класифікації.

Для класифікації було обрано алгоритм Random Forest. Він є одним із найефективніших та найпоширеніших алгоритмів машинного навчання, що відноситься до класу ансамблевих. Його ключова задумка полягає в тому, щоб сформувати велику кількість дерев рішень, причому кожне з них проходить навчання на власній випадково обраній порції наявного навчального набору даних [40]. Рисунок моделі під час класифікації наведено на рисунку 4.

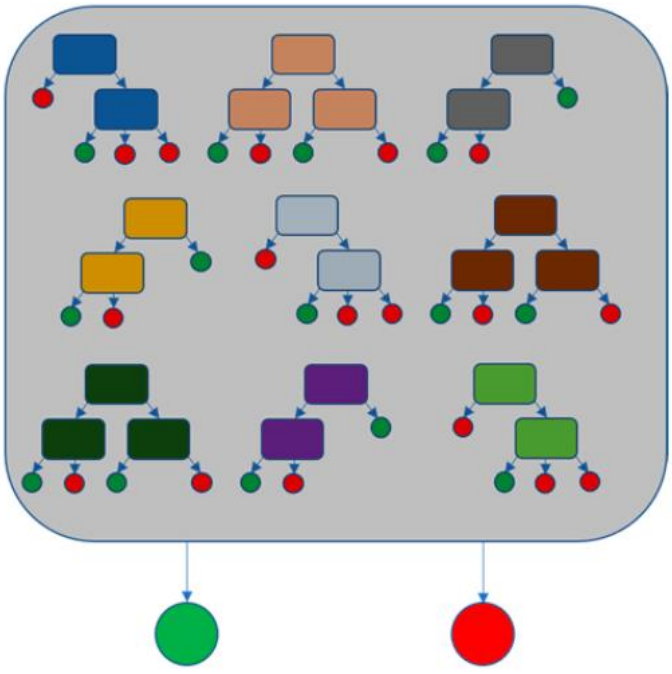


Рисунок 4 – Класифікація дерева рішень

Потім всі ці дерева працюють разом, голосуючи за кінцевий результат. Отже, Random Forest не ухвалює рішення, опираючись на єдине дерево, подібно до звичайного decision tree. Натомість, він використовує колективний підхід, агрегуючи результати багатьох моделей. Дерева у цій моделі формуються із застосуванням техніки bootstrap aggregating. Це означає, що для кожного дерева шляхом випадкового підбору з поверненням обирається підмножина прикладів з навчального набору. Крім цього, на кожному з етапів розбиття вузлів дерева використовується лише частина доступних ознак, а не всі. Це сприяє зменшенню кореляції між деревами, що робить їх стійкішими до перенавчання та підвищує їх узагальнюючу здатність.

У процесі навчання кожне дерево в Random Forest намагається знайти найкращі розділення для своїх вузлів, щоб максимально відокремити приклади різних класів наприклад, шкідливі сайти від легітимних. Дерево вивчає логіку розподілу класів у підмножині даних, яка йому дісталась, але оскільки всі дерева мають різні дані й вибирають різні ознаки, то їх помилки виявляються некорельованими. Ключова перевага полягає у використанні ансамблю незалежних рішень де численні слабкі класифікатори в комбінації формують потужну модель з підвищеною точністю Під час передбачення тобто коли алгоритм отримує новий приклад для класифікації кожне дерево у моделі видає свій варіант відповіді, а фінальний клас визначається більшістю голосів. У випадку класифікації, як це відбувається у даному розширенні, якщо більшість дерев вважають, що сайт є шкідливим, то результатом буде відповідь шкідливий. Якщо більшість голосів за те що вебсайт безпечний, то результатом буде безпечний.

Однією з переваг Random Forest є також інтерпретованість. Після навчання можна оцінити, які ознаки були найбільш важливими для прийняття рішень, що дає змогу не тільки використовувати модель як чорний ящик, але й пояснювати її рішення. Це особливо ціно в контексті безпеки, де важливо розуміти, чому певний сайт був визнаний небезпечним. Ще однією рисою, властивою Random Forest, є його здатність витримувати вплив шуму у даних та незначні варіації

вхідних параметрів. На відміну від окремих дерев прийняття рішень, які часто мають високу чутливість до незначних змін у навчальних даних, Random Forest демонструє високу надійність, стабільність прогнозів і добру узагальнювальну здатність навіть у складних умовах. Окрім статистичної моделі, було реалізовано додатковий набір логічних ознак, які можуть ще до машинного аналізу попередньо виявити очевидні ознаки фішингу. Наприклад, якщо сайт використовує не захищене з'єднання замість захищеного, система одразу може отримує попередження без необхідності повної побудови вектору ознак. Такий підхід особливо ефективний для пришвидшеної фільтрації явно небезпечних ресурсів або для ситуацій, коли обчислювальні ресурси обмежені. Крім того, в окремих випадках дані правила можуть виявити нові атаки, які ще не були представлені у тренувальних даних моделі.

Використання моделі машинного навчання для аналізу виявилось найбільш ефективним рішенням для задачі виявлення шкідливих сайтів. Також набір ознак дозволяє швидко ідентифікувати найбільш очевидні загрози, що не потребують складної обробки. Водночас модель машинного навчання зокрема алгоритм Random Forest забезпечує глибший аналіз сайтів на основі численних ознак, що враховують як структуру URL, так і елементи DOM та поведінку сторінки. Такий комбінований підхід дозволяє досягти високого рівня точності при низькому рівні неправильних спрацювань, що важливо в умовах роботи розширення в реальному часі. Крім того, інтеграція моделі в браузерне середовище без потреб звернення до зовнішніх серверів забезпечує швидкість обробки та захист конфіденційних даних користувача.

Таким чином, застосування методу, що ґрунтується на моделі Random Forest, є цілком обґрунтованим і відповідає основним критеріям, що висуваються до систем для ідентифікації шкідливих вебсайтів. Завдяки своїй здатності працювати з великою кількістю параметрів без істотного зниження продуктивності, дана модель демонструє стабільність результатів у різних умовах. Також важливим аспектом є безпека моделі яка має вбудовані механізми захисту, що дозволяють зменшити ризик неправильних спрацьовувань і загалом

підвищити довіру до системи. У сукупності всі ці характеристики роблять обраний підхід оптимальним для створення інструменту, для захисту від шкідливих вебсайтів.

## 2.2 Реалізація моделі машинного навчання та інтеграція в браузерне розширення

Реалізація системи виявлення шкідливих вебсайтів розпочалася зі створення ефективної та незалежної від серверної частини моделі машинного навчання, яка б могла працювати повністю локально в браузерному середовищі, не потребуючи надсилання даних на зовнішні сервери. Такий підхід забезпечує не лише конфіденційність користувача, а й значно покращує швидкодію системи, дозволяючи здійснювати миттєву перевірку посилань без затримок, пов’язаних із мережею. Було вирішено застосувати алгоритм Random Forest, інтегруючи його з набором характерних ознак, що забезпечать функціональну модель. Такий вибір обумовлений декількома ключовими факторами по-перше, Random Forest відзначається стійкістю до перенавчання, по-друге він має хорошу узагальнюючу здатність, і по-третє його структура ідеально підходить для перетворення у формат, придатний для реалізації в браузері.

Першим кроком у створенні моделі стало формування навчального датасету. Для цього було використано відкритий датасет Phishing Websites, розміщений у репозиторії UCI Machine Learning Repository [https://archive.ics.uci.edu/dataset/327/phishing+websites]. Даний набір містить адреси, які вже класифіковані як шкідливі або легітимні, та охоплює різноманітні характеристики, що дозволяють ідентифікувати потенційно небезпечні вебсайти. Перед початком навчання моделі також було проведено попередню обробку даних, видалено невалідні або пошкоджені записи, усунуто дублікати, а також забезпечено однаковий формат представлення кожної ознаки. Це дозволило забезпечити цілісність і якість навчального набору для

подальшого навчання класифікатора. На основі аналізу цього датасету було обрано та сформовано 17 ключових ознак, які найкраще відображають поведінку шкідливих сайтів. Ці ознаки охоплюють різні аспекти вебресурсу, як технічні так і поведінкові. Перелік ознак для виявлення шкідливих вебсайтів наведено в таблиці 1.

Таблиця 1 - Перелік ознак для виявлення шкідливих вебсайтів

| № | Ознака | Опис |
|---|---|---|
| 1 | 2 | 3 |
| 1 | Наявність IP адреси в URL | Використання IP адреси замість доменного імені викликає підозри. |
| 2 | Довжина URL | Адреси які довші за 75 символів, часто свідчать про спробу приховати справжнє призначення. |
| 3 | Скорочена URL адреса | Використовується для маскування кінцевого призначення посилання. |
| 4 | Символ "@" у URL | Може бути використаний для редиректу, що вводить в оману щодо реального домену. |
| 5 | Подвійний слеш ("//") у шляху | Може застосовуватись для обходу фільтрів або маскування структури адреси. |
| 6 | Дефіс у доменному імені | Наявність дефісів може свідчити про фальшивий домен, схожий на легітимний. |
| 7 | Кількість піддоменів | Якщо більше трьох доменів є можливість спроби імітації авторитетного ресурсу. |
| 8 | Захищене з’єднання (HTTPS) | Відсутність захищеного з’єднання може вказувати на фішинговий сайт. |
| 9 | HTTPS у доменному імені | Включення https у домен може вводити в оману, не забезпечуючи безпеку. |
| 10 | Походження favicon (логотипу сайту) | Якщо логотип сайту завантажується з іншого сайта це вже є підозріло. |

Кінець таблиці 1

| 1 | 2 | 3 |
|---|---|---|
| 11 | Нестандартний порт | Використання портів, відмінних від стандартних, може містити загрозу. |
| 12 | Зовнішні зображення | Велика кількість зображень із зовнішніх джерел може свідчити про небезпечність. |
| 13 | Зовнішні скрипти та CSS | Завантаження скриптів зі сторонніх сайтів може містити шкідливий код. |
| 14 | Анкор-посилання на зовнішні домени | Використовуються для редиректу або крадіжки трафіку. |
| 15 | Наявність iFrame | Приховані фрейми можуть містити шкідливий код або підміняти інтерфейс. |
| 16 | Обробник форм (SFH - Server Form Handler) | Відбувається перевірка, чи відправлення форми адресоване на той же домен, чи на сторонній. |
| 17 | Використання mailto | Вказує на намагання перехопити інформацію, введену користувачем, відправляючи її напряму на електронну скриньку зловмисника. |

Кожна ознака оцінюється за шкалою з трьох рівнів де -1 нормальна поведінка, 0 це нейтральна та 1 підозріла поведінка. Таким чином формується вектор ознак, який подається на вхід моделі машинного навчання. Для побудови моделі було застосовано мову програмування Python, разом з бібліотекою scikit-learn. Остання є однією з найпоширеніших бібліотек для розробки алгоритмів машинного навчання. Вибір саме цієї бібліотеки обумовлений її широким функціоналом, простотою у використанні та можливістю гнучко налаштовувати параметри моделей, зокрема дерев рішень та ансамблевих методів.

Процес реалізації моделі складався з кількох основних етапів де спочатку завантаження попередньо сформованих ознак, оцінка якості моделі за допомогою крос-валідації, остаточне навчання класифікатора, тестування на

відкладеній вибірці та експорт у формат JSON який у подальшому можна використати в браузерному середовищі. На першому етапі здійснюється завантаження даних. Вектор ознак і відповідні мітки класів що зберігаються у форматі npy, та забезпечують швидке читання даних без потреби у додатковому парсингу. Створюється об’єкт моделі RandomForestClassifier та перевіряється за допомогою 10-кратної крос-валідації, що дозволяє оцінити її узагальнюючу здатність на різних підвибірках даних. Середнє значення точності крос-валідації є першим показником ефективності побудованого класифікатора: Після позитивного результату крос-валідації виконується навчання моделі на повному навчальному наборі.

Для оцінки роботи моделі тобто для імітації реального сценарію завантажується тестова вибірка де є окремий набір прикладів, який не використовувався під час навчання. Після цього виконується передбачення класів на основі тестових даних і обчислюється остаточна точність моделі за допомогою основних метрики якості класифікації для машинного навчання в які входить акуратність (accuracy), точність (precision), повнота (Recall) та F1-міра що є усередненим значення між точністю та повнотою.

Останнім етапом стало експортування навченої моделі у формат JSON, який є придатним для завантаження в браузер. Оскільки scikit-learn не передбачає прямого перетворення моделей у JSON-структури, було реалізовано допоміжний модуль, який містить функцію forest_to_json(). Вона рекурсивно обходить усі дерева класифікатора та зберігає їх у вигляді вкладених словників зі структурою умов на кожному вузлі дерева. Це забезпечує можливість побудови інтерпретованої моделі у середовищі JavaScript. Файл classifier.json містить повну структуру лісу з усіма деревами, порогами, номерами ознак та відповідними виходами. Цей файл згодом буде використовуватись у клієнтському розширенні для побудови класифікації, яка працюватиме прямо в браузері, без запитів до серверів. Схема алгоритму навчання моделі наведена рисунку 5.

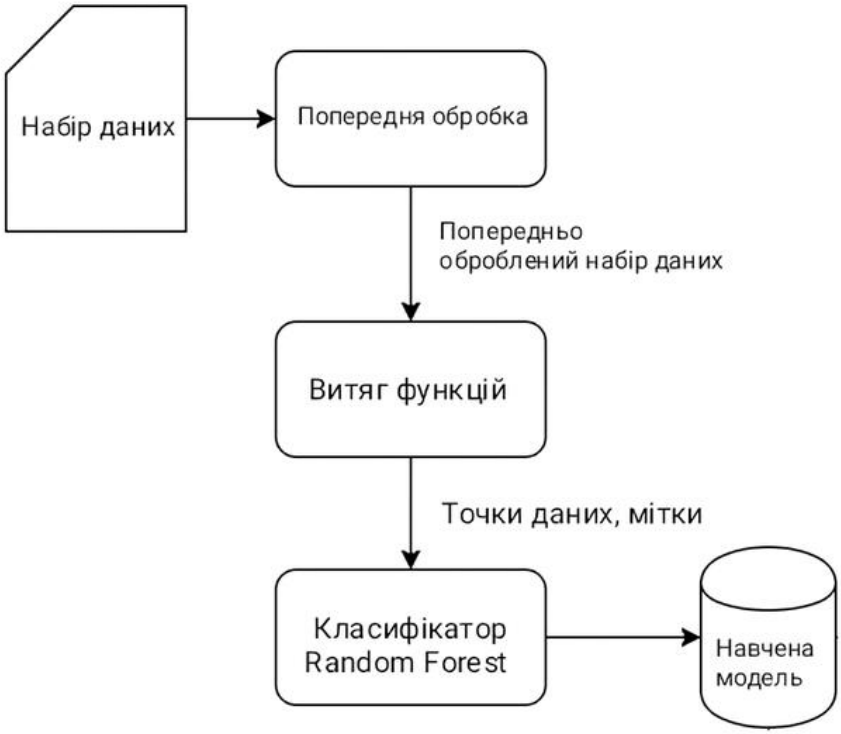



Рисунок 5 - Схема алгоритму навчання моделі

Таким чином, модель повністю відповідає вимогам локальної обробки, високої швидкості роботи та захисту конфіденційних даних користувача. Розробка самого розширення і інтеграція навченої моделі в нього, складається з декількох логічно взаємопов’язаних компонентів, кожен із яких виконує вузькоспеціалізовану функцію у процесі обробки вебсторінки. Ключовими структурними файлами є конфігураційний файл manifest.json, модуль формування ознак, фоновий класифікатор, модель, та користувацький інтерфейс. Ці компоненти організовані відповідно до структури MVC, де features.js та background.js виконують функції моделі, а елементи інтерфейсу вигляду.

Розроблено файла manifest.json що є основним конфігураційним файлом браузерного розширення, він описує всі його ключові характеристики, дозволи, точки входу та ресурси. Цей файл є невід’ємною частиною структури будь-якого розширення й виконує роль своєрідного паспорта, що визначає, як розширення інтегрується у браузер, які ресурси воно може використовувати, та які його компоненти активуються у відповідні моменти. У manifest.json зазначено назву

розширення SafeChecker, його версію та короткий опис. Для цього використано версію маніфесту 2, яка хоч вважається вже застарілою в нових версіях браузера Chrome, однак залишається повністю сумісною з більшістю функцій, які необхідні для даного проєкту. Окрема увага була приділена системі дозволів, які надають доступ до ключових функцій браузера, а саме:

- activeTab, що дозволяє взаємодіяти з активною вкладкою, читати її вміст та впроваджувати скрипти;
- declarativeContent, яка використовується для обробки умов та запуску розширення за визначеними параметрами;
- storage, що забезпечує збереження локальних даних;
- webNavigation, що відслідковує зміни в навігації між сторінками.

Секція background вказує на файли, які будуть виконуватись у фоні, поза межами поточної сторінки. Також був встановлений параметр "persistent" в значені “true” що означає що фонова частина завжди активна, та забезпечує безперервну готовність до обробки повідомлень і подій. Секція browser_action описує взаємодію з інтерфейсом користувача зокрема, який HTML-файл у моєму випадку це plugin_ui.html який відкривається при натисканні на іконку розширення в панелі інструментів.

Секція content_scripts визначає скрипти, які автоматично вбудовуються в кожну завантажену сторінку, що відповідає шаблонам адрес http та https. Секція web_accessible_resources вказує на файли, які можуть бути доступні скриптам із контексту сторінки або іншим компонентам розширення. Тут міститься файл classifier.json, у якому експортовано навчену модель, та файл з тестовими даними для перевірки якості класифікації моделі. Загалом, тут визначено повноцінну та злагоджену структуру браузерного розширення з чітким розділенням відповідальності між компонентами, доступом до необхідних API, організацією інтерфейсу користувача та забезпеченням клієнтської обробки складних моделей машинного навчання. Саме завдяки цьому маніфесту браузер розуміє, як саме має працювати розширення та які дії воно може виконувати.

Розроблено файл features.js, який виконує роль в аналізі вебсторінки. Саме

тут реалізовано модуль збору ознак, що використовуються для виявлення потенційно шкідливих вебсайтів. Цей скрипт автоматично виконується в контексті активної вкладки завдяки налаштуванням у manifest.json, де його зазначено в секції content_scripts. Основне його завдання полягає у побудові вектора ознак, який кодує характеристики сторінки на основі аналізу за списком ознак який був зроблений раніше. Всього формується 17 параметрів, які кодуються числовими значеннями де -1 це ознака яка вказує на безпечну поведінку, 0 якщо її значення нейтральне, та 1 якщо вона є потенційно шкідливою. Результат цієї перевірки додається до підсумкового об'єкта, який після завершення передається у фоновий скрипт. Таким чином, features.js фактично виконує роль сенсора, що формує цифровий знімок поточної сторінки, на основі якого далі виконується класифікація за допомогою моделі машинного навчання. Такий підхід забезпечує локальну обробки без потреби у надсиланні даних на сторонні сервери, високу швидкість роботи, гнучкість для інтеграції нових ознак і масштабованість під різні браузери.

Розроблено скрипт background.js, який відіграє критично важливу роль у роботі розширення, оскільки саме в ньому реалізовано логіку взаємодії з іншими компонентами системи, запуск моделі машинного навчання та формування остаточного рішення щодо безпечності або підозрілості відвідуваного вебсайту. Цей скрипт працює у фоні та постійно активний у контексті браузера, навіть тоді, коли жодна вкладка прямо не взаємодіє з розширенням. Завдяки цьому розширення може обробляти стан події, реагувати на зміни вкладок, а також зберігати тимчасові дані. У самому початку коду оголошуються три зміні, вони використовуються як локальні сховища для проміжних результатів класифікації, зокрема для збереження оцінки безпечності ресурсу, а також для позначення, чи виявлений сайт є шкідливим.

Функція fetchLocal(), яка відповідає за завантаження попередньо реалізованої моделі машинного навчання у форматі classifier.json. Функція виконує асинхронний запит до локального ресурсу розширення, після чого зберігає модель у кеш. Це дозволяє використовувати кешовану версію моделі

між сесіями та уникати повторного завантаження. Функція fetchCLF() є обгорткою для fetchLocal() і забезпечує доступ до моделі. Вона перевіряє, чи модель збережена у кеші і чи не минуло багато часу з останнього моменту збереження. Якщо все в порядку, повертається кешована модель, інакше виконується нове завантаження моделі. Такий механізм дозволяє забезпечити стабільну і швидку роботу моделі без непотрібних повторних запитів.

Основна класифікація виконується у функції classify(). Вона приймає два параметри ідентифікатора активної вкладки і об’єкта, що містить вектор ознак, зібраний за допомогою features.js. Цей об’єкт проходить попередню обробку рахується кількість ознак, які мають значення 1 (означає шкідливу активність), 0 (нейтральну поведінку) та -1 (безпечну поведінку). На основі цього обчислюється відсоток легітимності сайту та зберігається у legitimatePercents. Після цього вектор ознак конвертується у формат, придатний для класифікації, і передається у модель Random Forest, яка викликається через random_forest(clf). Ця функція, реалізована у файлі randomforest.js, що виконує роботу дерева рішень у браузерному середовищі. Модель повертає прогноз який вказує чи сайт класифікується як шкідливий чи ні.

Якщо сайт позначено як шкідливий, то для відповідної вкладки об'єкта isPhish проставляється позитивне значення. Після цього відбувається передача повідомлення використовуючи команду "alert_user". Це повідомлення перехоплює інтерфейс, де відповідно буде показане попередження користувачу через зміну вмісту інтерфейсу розширення. Насамкінець, скрипт слухає зовнішні повідомлення. Отримані відомості зберігаються у змінній, а потім викликається функція класифікації. У сукупності, background.js реалізує зв’язувальну логіку між збором ознак та прийняттям рішень. Він гарантує асинхронну, відірвану від взаємодії з користувачем роботу моделі машинного навчання, дозволяючи системі функціонувати, незважаючи на поточний стан користувача або зв'язку з мережею. Це значно покращує надійність та безпеку розширення.

Розроблено randomforest.js, який є ключовим елементом, що дозволяє виконувати класифікацію на основі алгоритму Random Forest безпосередньо в

середовищі браузера. Його головною функцією є забезпечувати роботу попередньо навчених дерев рішень, які було експортовано. Це дозволяє уникнути використання серверної частини або сторонніх інтерпретаторів, і забезпечити повністю клієнтську реалізацію машинного навчання. Першою частиною скрипта є реалізація функції decision_tree, яка моделює окреме дерево рішень. Кожне дерево побудоване у вигляді вкладених вузлів типу "split" (вузли розгалуження) або листків з кінцевими класами. Функція predictOne реалізує обхід дерева для одного вектору ознак. Вона починає з кореневого вузла та поступово переходить до лівої або правої гілки дерева залежно від того, чи значення певної ознаки менше або більше за заданий поріг та поки не дійде до листка кінцевого рішення дерева. Повертається значення, що вказує на клас наприклад, [0, 1] або [1, 0].

Функція predict в об'єкті decision_tree виконує передбачення одразу для кількох векторів ознак тобто для масиву ознак. Вона використовує predictOne для кожного з них і збирає результати у масив. У скрипті також реалізується структура яка описує ансамбль із багатьох дерев. Алгоритм передбачає, що кожне дерево окремо робить передбачення, а далі результати об’єднуються за принципом більшості голосів. У коді це реалізується через послідовний виклик кожного дерева, яке створюється за допомогою функції decision_tree.

Після цього всі передбачення збираються у загальний масив, де кожен елемент є результат голосування окремого дерева. Для кожного прикладу тобто кожного сайту, який класифікується підраховується, скільки дерев проголосувало за позитивний клас (шкідливий) і скільки за негативний (безпека). На основі більшості визначається підсумковий клас, а також обчислюється кількість голосів це дозволяє оцінити ступінь впевненості моделі у рішенні. Уся реалізація є повністю ізольованою та автономною а сама модель зчитується з файла classifier.json, її структура уніфікована, що дозволяє ефективно обробляти великі ансамблі дерев, а передбачення виконуються швидко завдяки оптимізованому обходу дерева. Уся складність роботи моделі, таким чином, сконцентрована у цьому файлі, що дозволяє легко підтримувати

або оновлювати алгоритм без втручання в інші частини розширення.

Реалізовано користувацький інтерфейс розширення, що забезпечує зручну і зрозумілу взаємодію між системою виявлення шкідливих сайтів та кінцевим користувачем. Основу інтерфейсу становлять три ключові файли plugin_ui.html, plugin_ui.css та plugin_ui.js. Вони разом утворюють єдину фронтенд-структуру, яка вбудовується в браузер Chrome і відображається у вигляді спливаючого вікна, що активується при натисканні на іконку розширення. Основною метою створення цього інтерфейсу було забезпечення інформативного візуального зворотного зв'язку, користувач має миттєво розуміти, чи є вебсайт, який він відвідує, безпечним або підозрілим.

Візуальне подання інформації реалізовано за допомогою логічно впорядкованих компонентів логотипа, що ідентифікує назву системи, округлого кольорового індикатора з відсотковою оцінкою легітимності сторінки, пояснювального тексту з вердиктом моделі та списку активних ознак, які було виявлено під час аналізу. Файл plugin_ui.html відповідає за семантичну розмітку цих елементів, використовуючи сучасні вебтехнології, зокрема CSS-фреймворк bootstrap, для підвищення адаптивності й зручності у відображенні. Водночас файл стилів plugin_ui.css визначає кольорову палітру інтерфейсу, оформлення шрифтів, просторову ієрархію блоків, а також візуальні ефекти для інтерактивних елементів. Завдяки ретельно продуманому дизайну, користувач здатен легко розуміти результати аналізу. Індикатор сам змінює колір відповідно до рівня загрози: зелений колір вказує на безпечну сторінку, жовтий на сумнівну, а червоний про потенційну небезпеку.

Уся логіка взаємодії з відображенням даних розроблена у файлі plugin_ui.js, який відповідає за обробку інформації, що надходить із фонової частини розширення. Після завантаження сторінки скрипт запитує актуальні результати класифікації для активної вкладки, зокрема, оцінку легітимності у відсотках, статус безпеки сайту фішинговий або ні та перелік ознак, які спрацювали. Кожну з ознак скрипт виводить на екран у вигляді підписаного елемента з відповідним кольором фону, який візуально позначає її характер.

Таким чином користувач може бачити не лише загальний результат, а й розуміти, які саме аспекти сайту викликали підозру. Окрім цього, в інтерфейсі передбачено можливість перегляду результатів моделі через окреме посилання, що веде на сторінку з тестуванням яке розраховується в реальному часі. Важливо зазначити, що модуль інтерфейсу не лише відображає інформацію, а й підвищує рівень довіри до системи, оскільки забезпечує прозорість роботи механізму виявлення загроз. Його інтеграція дає змогу органічно поєднати результати складної класифікації із зручною та наочною формою, що робить взаємодію з розширенням приємною і корисною для будь-якого користувача, незалежно від технічного рівня підготовки. Приклад реалізованого інтерфейсу наведено на рисунку 6.

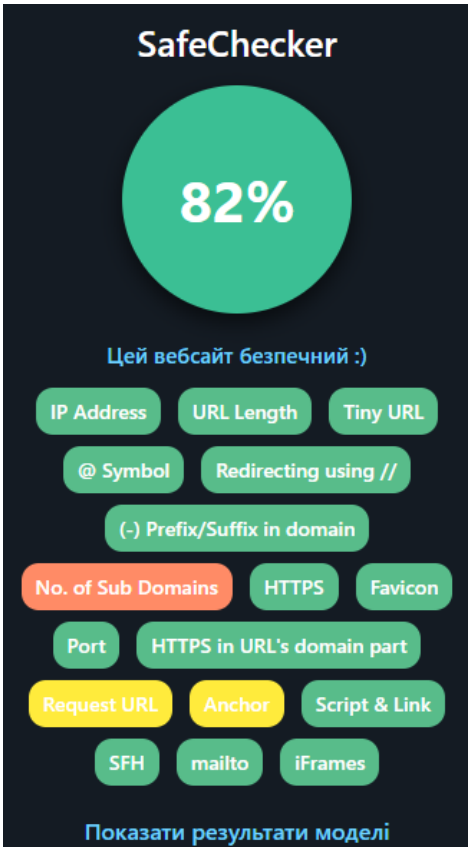



Рисунок 6 - Інтерфейс розширення

Використано бібліотеку jQuery, її роль полягає у спрощенні взаємодії з HTML-елементами, управлінні подіями та виконанні динамічних змін вмісту сторінки без необхідності писати складний або громіздкий код на чистому

JavaScript. Попри те, що основна логіка розширення включаючи збір ознак, класифікацію та обробку повідомлень реалізована з використанням стандартних засобів, jQuery залишається ефективним інструментом для реалізації зручного і швидкого інтерфейсу користувача. У межах розширення бібліотека jQuery використовується у файлі plugin_ui.js, де вона допомагає динамічно оновлювати інтерфейс спливаючого вікна відповідно до результатів аналізу сторінки. За допомогою компактного синтаксису jQuery реалізовано виведення повідомлень, зміна стилів елементів, додавання або приховування блоків, а також обробка подій натискання чи зміни стану елементів керування.

Наприклад, коли класифікатор визначає сторінку як шкідливу, jQuery дозволяє швидко оновити текст у DOM, змінити фон або колір індикатора ризику та показати відповідне попередження користувачу. Крім того, jQuery полегшує асинхронну взаємодію між частинами розширення, дозволяючи зручно реалізовувати механізми отримання й передачі даних, де jQuery допомагає оформити логіку реакції у компактній та зрозумілій формі. Хоча сучасний JavaScript включає багато можливостей, які раніше були унікальними для jQuery наприклад, querySelector та fetch, використання цієї бібліотеки досі є актуальне, коли йдеться про швидку розробку в умовах обмежених ресурсів чи часових рамок.

Таким чином, jQuery у цьому розширенні виконує допоміжну, але важливу роль що дозволяє прискорити розробку, зменшити обсяг коду і забезпечити стабільну взаємодію користувача з інтерфейсом, залишаючи основну логіку машинного навчання та обробки на ванільному скрипті. Уся взаємодія побудована на внутрішній логіці браузерного розширення, без використання мережевих з'єднань, що забезпечує високий рівень продуктивності та захисту особистих даних. Обрана архітектура розширення має низку важливих переваг, які визначають її ефективність і доцільність використання в реальному середовищі. Насамперед варто відзначити її повну локальність: усі обчислення, обробка даних і прийняття рішень відбуваються без звернення до зовнішніх серверів або API. Такий підхід не лише захищає конфіденційність користувача, а

й забезпечує стабільну роботу навіть за відсутності інтернет-з'єднання. Важливо й те, що архітектура побудована з урахуванням принципу масштабованості. Наприклад, модель класифікації, збережена у вигляді окремого файлу, може бути легко оновлена або замінена новою версією без необхідності змінювати основну логіку розширення. Це суттєво спрощує процес підтримки й модернізації програмного продукту.

Ще однією важливою рисою є гнучкість архітектури. Завдяки модульному підходу, нові ознаки або методи аналізу можуть бути додані до відповідних частин коду, зокрема до модуля features.js, без потреби втручання в інші компоненти. Це дозволяє адаптувати розширення до нових викликів типів атак. Не менш вагомою перевагою є висока продуктивність, усі операції від моменту завантаження сторінки до класифікації виконуються за частки секунди, завдяки чому користувач отримує миттєвий зворотний зв’язок. Це особливо актуально в умовах, коли рішення про безпечність ресурсу має бути прийняте негайно, до того як користувач взаємодіє зі шкідливим вмістом.

Також важливо відзначити прозорість та відкритість запропонованого рішення, що є суттєвою перевагою в контексті розробки безпечних, гнучких та довготривало підтримуваних інструментів для кінцевих користувачів. Завдяки використанню стандартних API, які надаються безпосередньо самим браузером, вдається уникнути залежності від пропрієтарного програмного забезпечення, закритих бібліотек чи платформ із обмеженою документацією або сумнівною репутацією. Така архітектура не лише знижує ризики безпеки, а й спрощує аудит, модифікацію та подальший розвиток розширення.

Крім того, обраний підхід істотно полегшує адаптацію розширення до різних браузерів і операційних систем, забезпечуючи високу браузерну сумісність. Це означає, що користувачі, незалежно від середовища Chrome, Firefox або будь-який інший браузер із підтримкою WebExtension API можуть однаково скористатися усіма функціями розробленої системи без необхідності встановлення додаткових компонентів чи виконання спеціальних налаштувань. Така універсальність підвищує доступність рішення для широкого кола

користувачів та зменшує поріг входу для його інтеграції в існуючі середовища. Усе це робить представлене розширення не лише експериментальним прототипом для демонстрації концепції, а й реальним, життєздатним продуктом, готовим до застосування в умовах реального інтернет-простору. Крім того, відкритість коду і зрозуміла структура проєкту створюють підґрунтя для залучення спільноти до його вдосконалення або адаптації під нові типи загроз, що постійно еволюціонують. Візуалізована схема роботи розширення подана на рисунку 7, де ілюструються ключові компоненти системи та логіка взаємодії між ними, зокрема процеси збору ознак, класифікації та подання результатів користувачеві.

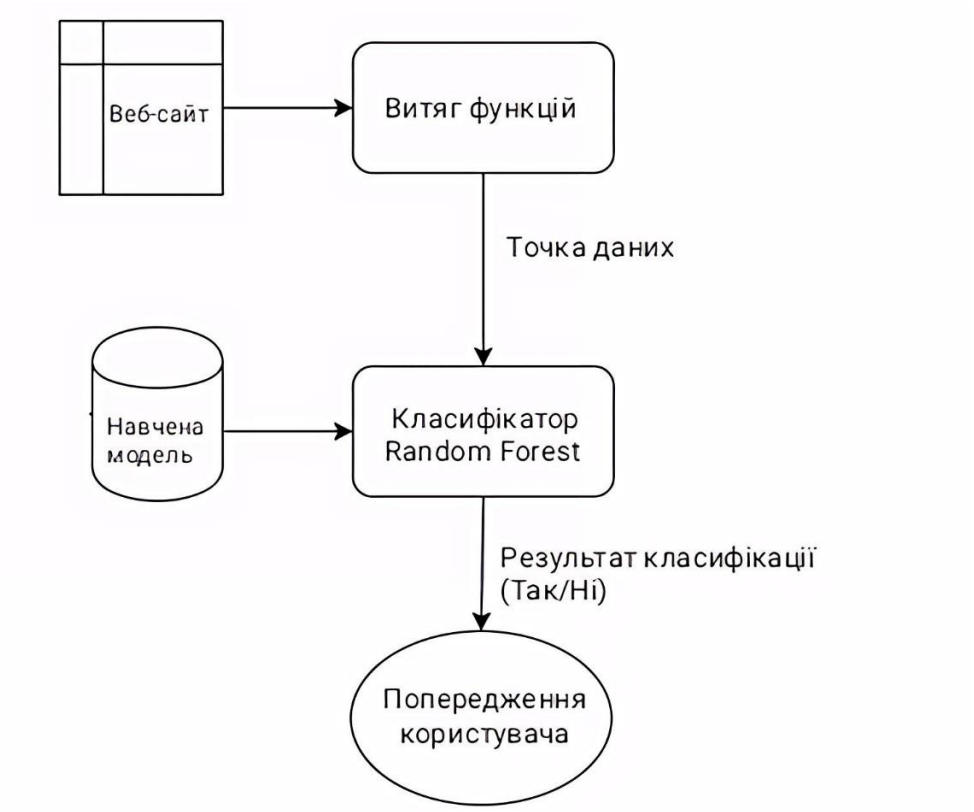



Рисунок 7 – Схема роботи розширення

Отже, розроблена архітектура поєднує в собі ефективність, адаптивність і безпеку, створюючи надійний фундамент для реалізації автоматизованого виявлення шкідливих вебсайтів.

## 2.3 Висновки до розділу

Підсумовуючи, у даному розділі було всебічно представлено процес створення системи для виявлення шкідливих вебсайтів, що базується на сучасних підходах машинного навчання. На етапі реалізації була побудована високоефективна модель класифікації, що оперує вектором із 17 ознак, які визначають специфіку вебресурсу. Ці ознаки були ретельно відібрані на основі найкращих практик у галузі виявлення фішингових і шкідливих сайтів, а їхній перелік та логіка обробки реалізовані в скрипті features.js. Саме цей скрипт відповідає за формування структурованого набору параметрів сторінки для подальшої обробки класифікатором, що працює без звернення до зовнішніх серверів. Такий підхід гарантує автономність, швидкодію та придатність до роботи в режимі реального часу.

Окрему увагу було приділено інтеграції навченої моделі у браузерне середовище. Для цього розроблено інтерфейс користувача, який не лише відображає результати класифікації, а й дозволяє зрозуміти, на основі яких ознак система прийняла те чи інше рішення. Вивід результатів реалізований у вигляді візуального повідомлення, що сигналізує про рівень небезпеки поточного сайту. Подібний механізм не лише підвищує зручність взаємодії користувача із системою, а й сприяє формуванню довіри до автоматизованого інструменту. Це особливо важливо в контексті систем, які прямо впливають на інформаційну безпеку користувача. У результаті була реалізована ефективна система, яка демонструє успішну інтеграцію алгоритмів машинного навчання у прикладне рішення, здатне працювати на практиці як повноцінне браузерне розширення. Отриманий продукт об’єднує технічну ефективність, UX/UI інтерфейсу та високу адаптивність до реальних умов функціонування у середовищі.

# 3 ВПРОВАДЖЕННЯ ТА ОЦІНКА ДОСТОВІРНОСТІ СИСТЕМИ

## 3.1 Оцінка достовірності системи

На даному етапі було здійснено широкомасштабне тестування створеної моделі для виявлення шкідливих вебсайтів. Метою було оцінити її продуктивність та точність. Особливу увагу зосереджено на здатності моделі точно розпізнавати як шкідливі, так і безпечні ресурси. Для цього було самостійно сформовано вибірку з понад 2000 унікальних вебсайтів, яка складалася з шкідливих та безпечних ресурсів. Дані були зібрані з відкритих баз PhishTank, OpenPhish, які щоденно оновлюються та містять перевірені записи про шкідливі домени. Для отримання переліку безпечних вебсайтів використовувався сайт Alexa Top Sites, а також адреси вебсайтів ключових державних та освітніх закладів. Перед тим, як передати дані до моделі, всі посилання підлягали попередній обробці за певними критеріями. Це гарантувало відповідність формату та готувало інформацію для подальшого аналізу. Цей етап включав очищення, нормалізацію, а також приведення адрес до єдиного вигляду, зручного для подальшої автоматизованої обробки. Отже, кожен запит спершу зазнавав стандартної підготовки, що забезпечувала його відповідність технічним параметрам та сумісність з моделлю. Після цього відбувався процес класифікації, який відбувався у режимі реального часу. Для оцінки якості класифікації використовувалися наступні метрики:

- Повнота (Recall) - співвідношення достовірно ідентифікованих шкідливих вебсайтів до загальної кількості дійсно шкідливих. Вона обчислюється за формулою 1.

$$\text{Recall} = \frac{TP}{TP + FN}, \quad (1)$$

де TP - істиннопозитивні; FN - хибнонегативні.

- Точність (Precision) - частка дійсно шкідливих вебсайтів серед усіх, які модель ідентифікувала як шкідливі. Вона обчислюється за формулою 2.

$$\text{Precision} = \frac{TP}{TP + FP}, \quad (2)$$

де TP - істиннопозитивні; FP - хибнопозитивні.

- Акуратність (Accuracy) - визначає загальну частку всіх правильних класифікацій. Вона обчислюється за формулою 3.

$$\text{Accuracy} = \frac{TP+TN}{TP + FP+FN+TN}, \quad (3)$$

де TP - істиннопозитивні; TN - істиннонегативні; FP – хибнопозитивні; FN - хибнонегативні.

- F1-міра (F1-score) – середнє значення між точністю та повнотою. Вона обчислюється за формулою 4.

$$\text{F1-score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}, \quad (4)$$

Після тестування моделі було отримано такі значення:

- TP (істиннопозитивні) - 1750;
- FP (хибнопозитивні) - 128;
- TN (істиннонегативні) - 1370;
- FN (хибнонегативні) - 69.

На основі цих значень було виконано обчислення:

Точність (Precision):

$$\text{Precision} = \frac{1750}{1750 + 128} = 0{,}9318\%, \quad (5)$$

Повнота (Recall):

$$\text{Recall} = \frac{1750}{1750 + 69} = 0{,}9621\%, \quad (6)$$

Акуратність (Accuracy):

$$\text{Accuracy} = \frac{1750 + 1259}{1750 + 128 + 69 + 1370} = 0{,}9406\% , \quad (7)$$

F1-міра (F1-score):

$$\text{F1-score} = 2 * \frac{0{,}9318 * 0{,}9621}{0{,}9318 + 0{,}9621} = 0{,}9467\% , \quad (8)$$

На рисунку 8 представлено порівняння підсумкових значень точності (precision), повноти (recall), акуратності (accuracy) та F1-міри між розробленою моделлю та іншими популярними моделями машинного навчання. Це дозволяє оцінити ефективність запропонованого рішення в контексті існуючих методів і виявити його сильні сторони в умовах реального застосування.

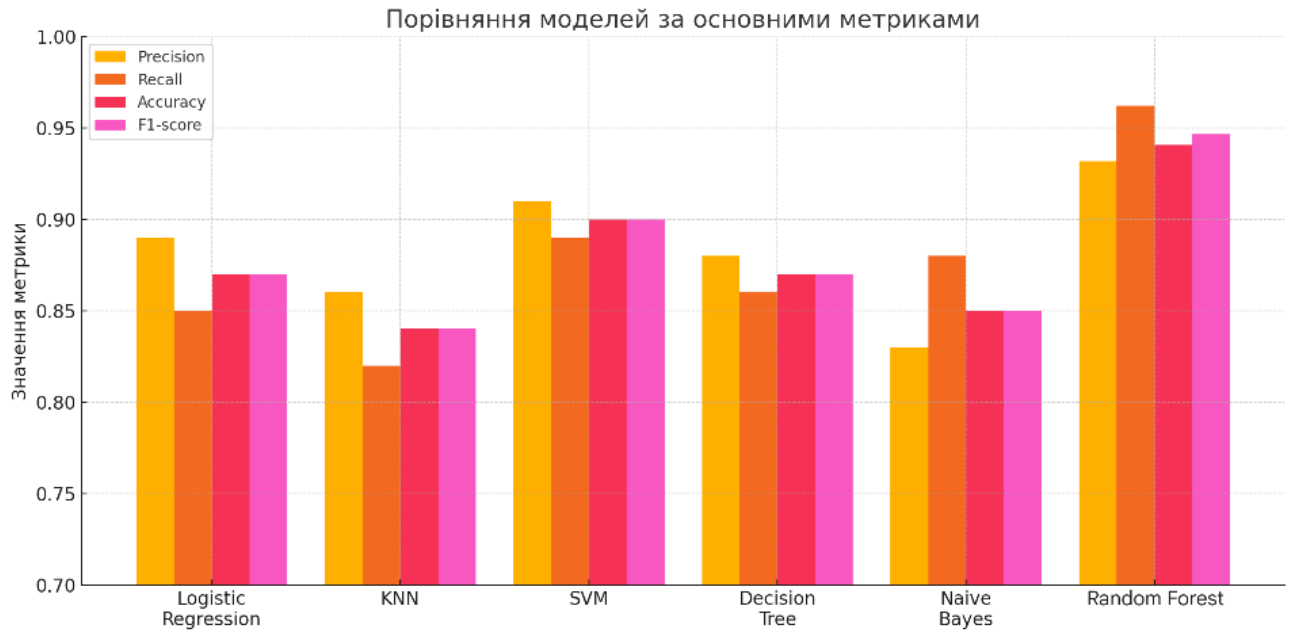



Рисунок 8 – Порівняння моделей за основними метриками

Отримані показники свідчать про те, що розроблена модель має високу здатність виявляти шкідливі вебсайти, при цьому демонструє низьку частоту помилкових спрацьовувань на безпечні сайти. Особливо важливою є висока F1-міра, що вказує на мінімізацію хибнонегативних результатів, коли шкідливий

вебсайт помилково класифікується як безпечний. Саме такі помилки є найнебезпечнішими в контексті реального використання системи, тому їхнє зниження є критично важливим. Під час дослідження помилок виявлено, що невелика частка хибної класифікацій спостерігалась у випадках, коли безпечні вебсайти мали нестандартну структуру або практикували численні перенаправлення, наприклад, новинні агрегатори чи ресурси, що взаємодіють з рекламними мережами. Натомість деякі з потенційно небезпечних вебсайтів, позначених як безпечні, демонстрували незначні відмінності від справжніх ресурсів, що ускладнювало їхнє виявлення без аналізу вмісту.

Ці результати дозволяють зробити висновок, що модель є ефективною у виявленні широкого спектра шкідливих вебсайтів, зокрема таких, які маскуються під легітимні сайти за допомогою соціального інжинірингу або використання технічних атрибутів довіри наприклад, шифрування, сертифікати та схожі доменні імена. Завдяки багатокомпонентному набору ознак, який охоплює як технічні характеристики, модель демонструє універсальність і стійкість до маніпуляцій з боку зловмисників.

Оцінка моделі у відриві від браузерного інтерфейсу дала змогу ізольовано виміряти її точність як математичної системи. Надалі така модель може бути використана не лише в рамках розширення, а й як серверна частина для інтеграції в інші продукти наприклад електронну пошту, корпоративні шлюзи безпеки або антифішингові API. Отже, результати оцінювання засвідчують доцільність використання обраного методу, а також демонструють ефективність застосування класичних ознак для класифікації.

## 3.2 Тестування розширення

У межах цього етапу було перевірено ефективність роботи моделі в умовах реального браузерного середовища. Для тестування сформовано контрольну вибірку з 550 вебсайтів, що включали як надійні офіційні ресурси,

так і потенційно небезпечні сторінки з бази PhishTank. Під час тестування було виявлено, що деякі шкідливі сайти отримували високий відсоток довіри, іноді вищий за безпечні ресурси. Проте класифікація при цьому залишалась правильною модель ідентифікувала їх як потенційно небезпечні. Слід розуміти, що відсоток довіри, який виводиться на інтерфейсі, не є абсолютним показником безпечності ресурсу. Це значення демонструє впевненість моделі у своєму рішенні та доповнюється аналізом додаткових ознак. Фрагмент використаної для тестування вибірки представлено в таблиці 2.

Таблиця 2 - Фрагмент вибірки, використаної для тестування

| № | URL сайта | Тип сайта | Результат |
|---|---|---|---|
| 1 | https://rozetka.com.ua | Безпечний | Безпечний |
| 2 | https://monobank.ua | Безпечний | Безпечний |
| 3 | https://wikipedia.org | Безпечний | Безпечний |
| 4 | https://github.com/resources/articles/ai/machine-learning-in-software-development | Безпечний | Безпечний |
| 5 | https://msn.khmnu.edu.ua/login/index.php | Безпечний | Безпечний |
| 6 | https://pagamento.novobeneficiobolsa.com.br/ | Небезпечний | Небезпечний |
| 7 | https://airdrop-ondo.com/ | Небезпечний | Небезпечний |
| 8 | https://xn--q3ccwc2bxc7erc.com/arawkaan.php | Небезпечний | Небезпечний |
| 9 | https://www.papasqueens.pe/IT-en/ARUBA/ARUBA/ | Небезпечний | Небезпечний |
| 10 | https://stellartokens.fun | Небезпечний | Небезпечний |

Першим сайтом на перевірку став інтернет-магазин розетка, який має багаторічну репутацію. Також має домен верхнього рівня ua та підтримує шифрування, і візуально виглядає легітимно. Результат тестування наведено на рисунку 9.

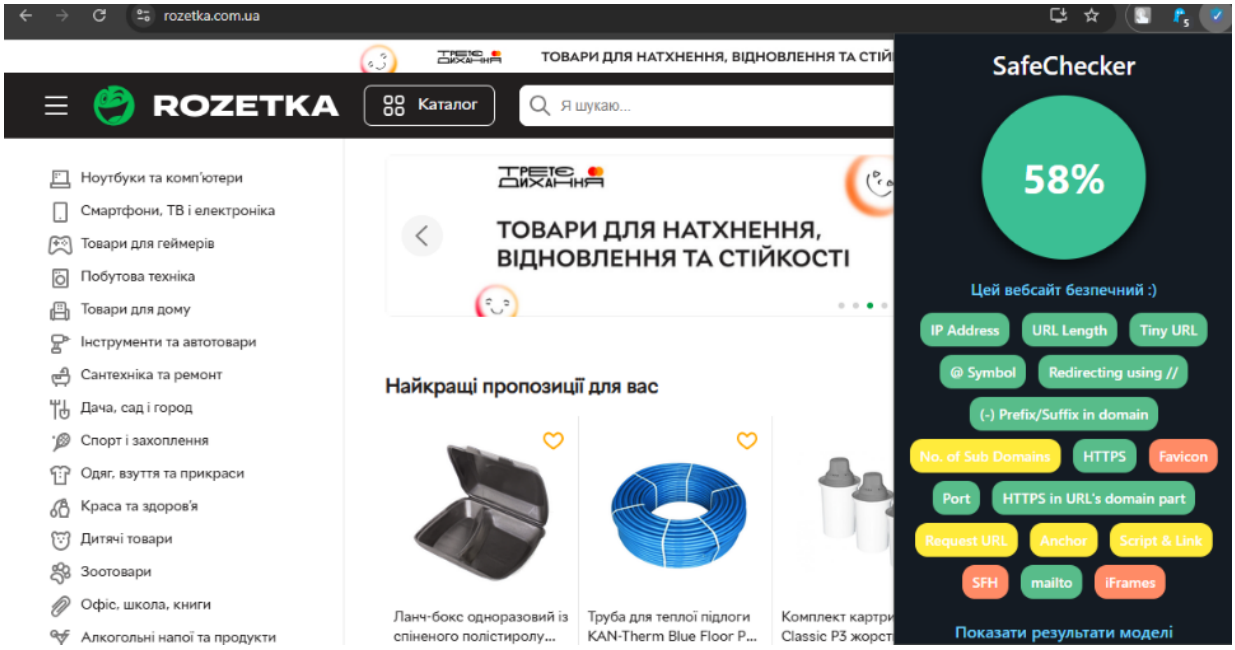



Рисунок 9 - Результат перевірки сайта rozetka

Після перевірки були виявлено лише ознаки фавікон, вбудованих фреймів і обробника форм. Сайт був класифікований як безпечний що є правильним результатом. Далі була перевірка онлайн банкінгу монобанк він є офіційним сайтом сучасного українського банку, що має високу впізнаваність серед користувачів. Результати перевірки наведені на рисунку 10.

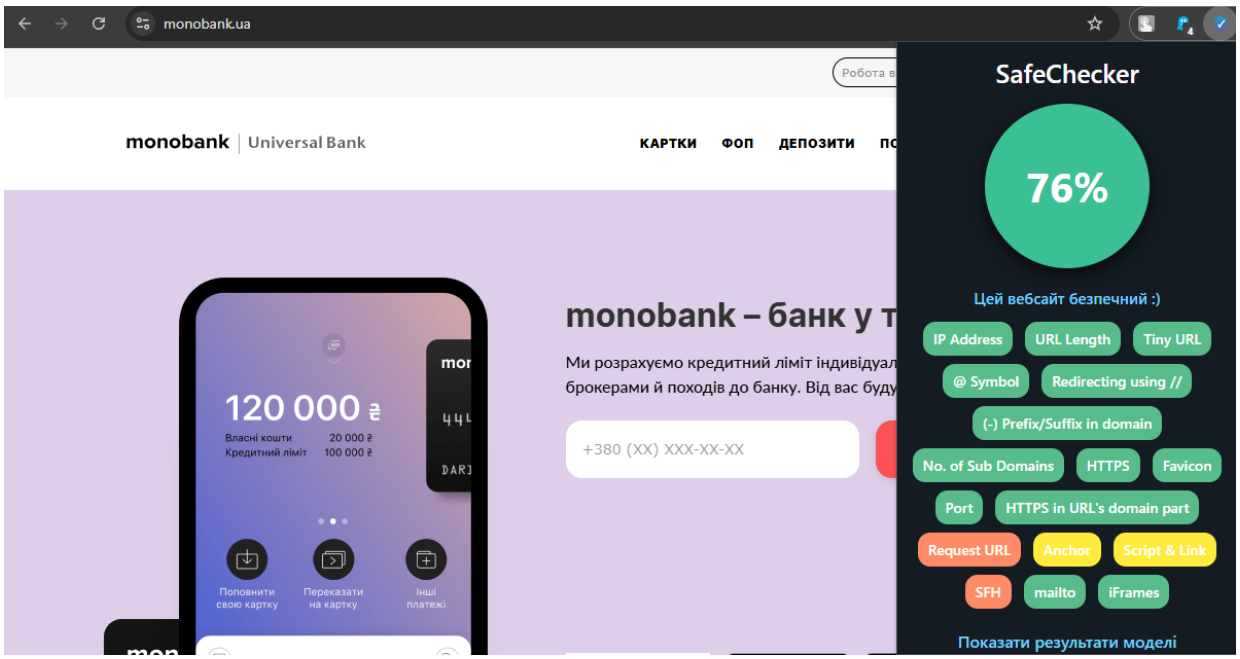



Рисунок 10 - Результат перевірки сайта monobank

Після перевірки було виявлено ознаки перенаправлень і обробника форм.

Сайт був класифікований як безпечний що є правильно. Далі було перевірено сайт глобальної енциклопедії з відкритим доступом, яка відома багатьом користувачам. Сайт має публічний домен та масштабовану структуру. Результати перевірки наведено на рисунку 11.


Рисунок 11 - Результат перевірки сайта wikipedia

Після перевірки було виявлено лише ознаки фавікону. Сайт був класифікований як безпечний що є вірно. Далі було перевірено сайт GitHub він є одним із найвідоміших у світі вебсайтів для спільної розробки програмного забезпечення. Зазначене посилання веде на статтю, що присвячена використанню машинного навчання у процесі розробки програмного забезпечення. Сайт має вбудовану систему авторизації, сертифікацію, і загалом підтримує високу репутацію серед користувачів і пошукових систем. Результати перевірки відображено на рисунку 12.

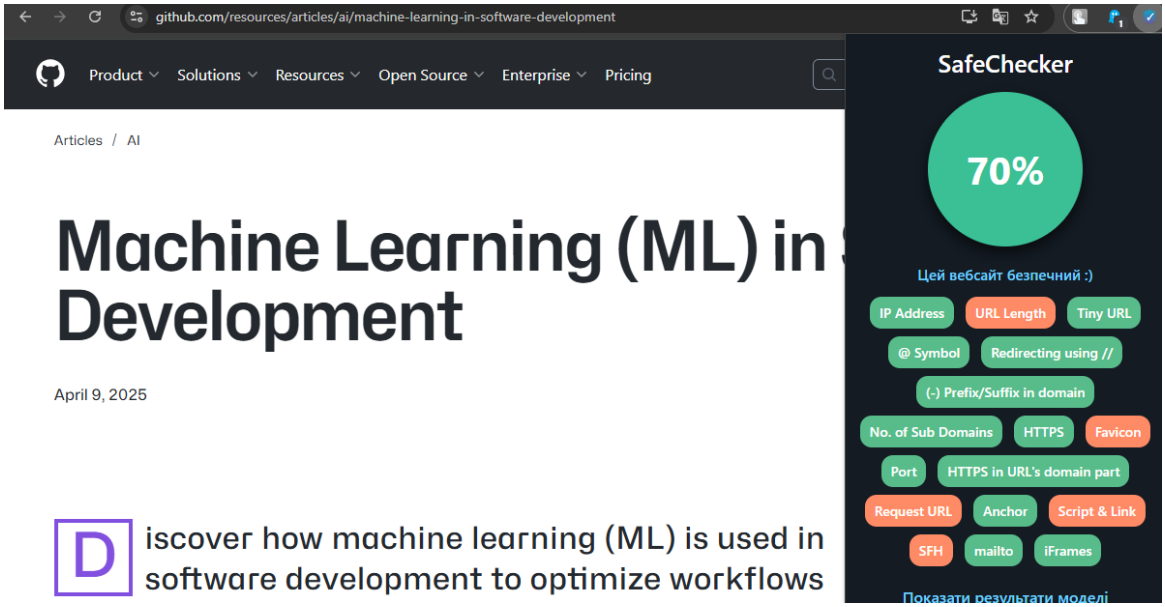



Рисунок 12 - Результат перевірки сайта github

Після перевірки було виявлено ознаки довжини адреси, логотипа, запитів на інші ресурси, використання скриптів та обробника форм. Сайт був класифікований як безпечний що є вірно. Далі було перевірено сайт модульного середовища який функціонує на піддомені державного освітнього закладу з доменом edu.ua. Така доменна зона зазвичай використовується лише для акредитованих навчальних установ, що вже створює початкову довіру до ресурсу. Результати перевірки наведено рисунку 13.

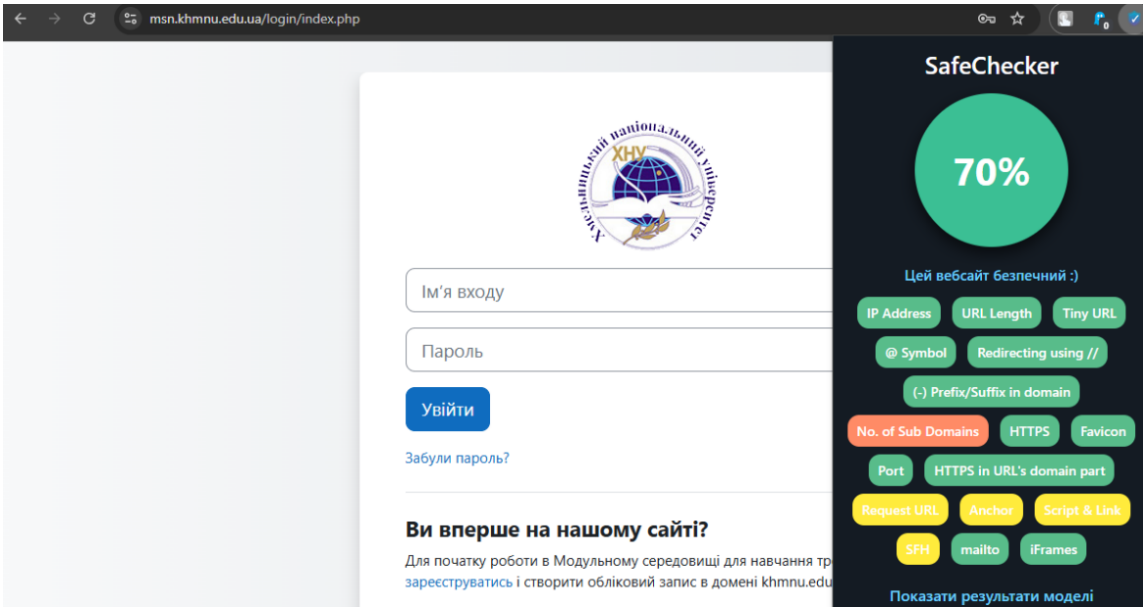



Рисунок 13 - Результат перевірки сайта модульного середовища

Після перевірки було виявлено лише ознаки кількості субдоменів. Сайт був визначений як безпечний що є правильно. Далі був розпочатий етап перевірки на завідомо небезпечних вебсайтах, які були відібрані з публічних баз шкідливих ресурсів, зокрема PhishTank. Перевірка небезпечних сайтів почалась з сайту pagamento сама адреса виглядає як частина соцвиплат Бразилії, однак структура адреси викликає підозру. Домен виглядає неофіційно. Результат наведено на рисунку 14.

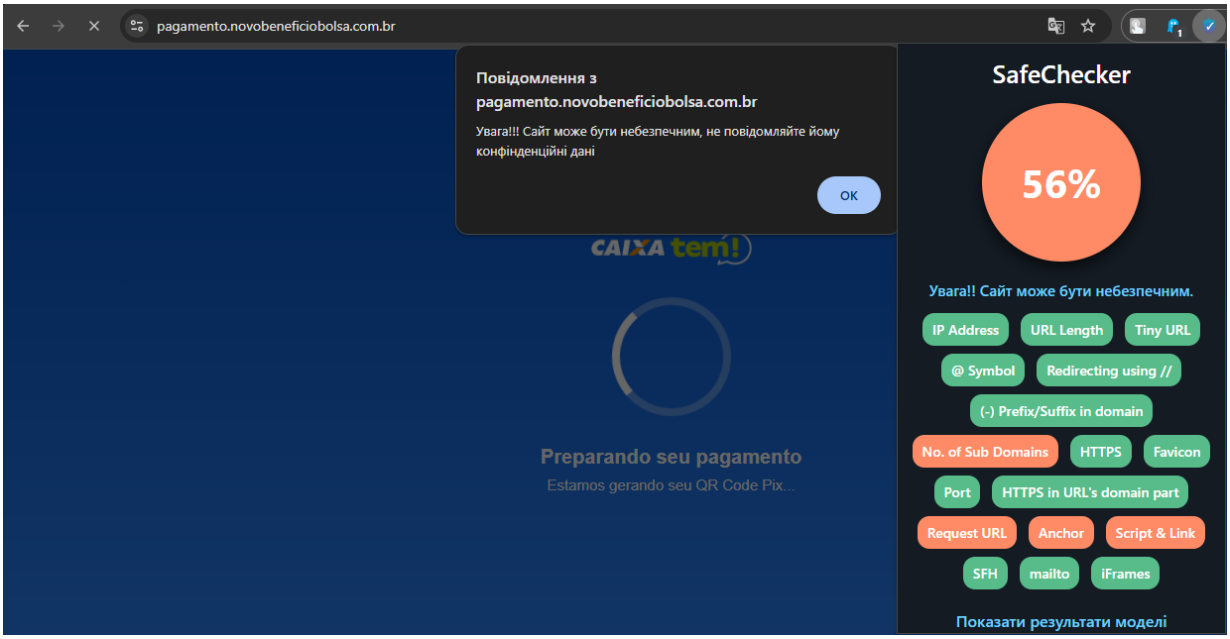



Рисунок 14 - Результат перевірки сайта pagamento

Перевірка виявила ознаки кількості субдоменів, запитів на інші ресурси, анкор посилань та використання скриптів. Сайт врешті був визначений як небезпечний що є вірно. Перевіркою сайта airdrop-ondo було встановлено, що сторінка присвячена нібито роздачі криптовалюти у вигляді так званого "airdrop", що вже само по собі є типовим індикатором шахрайської діяльності. Цей сайт приваблює користувачів обіцянками швидкого прибутку або безкоштовного отримання токенів в обмін на надання даних криптогаманця. Результат перевірки наведено на рисунку 15.

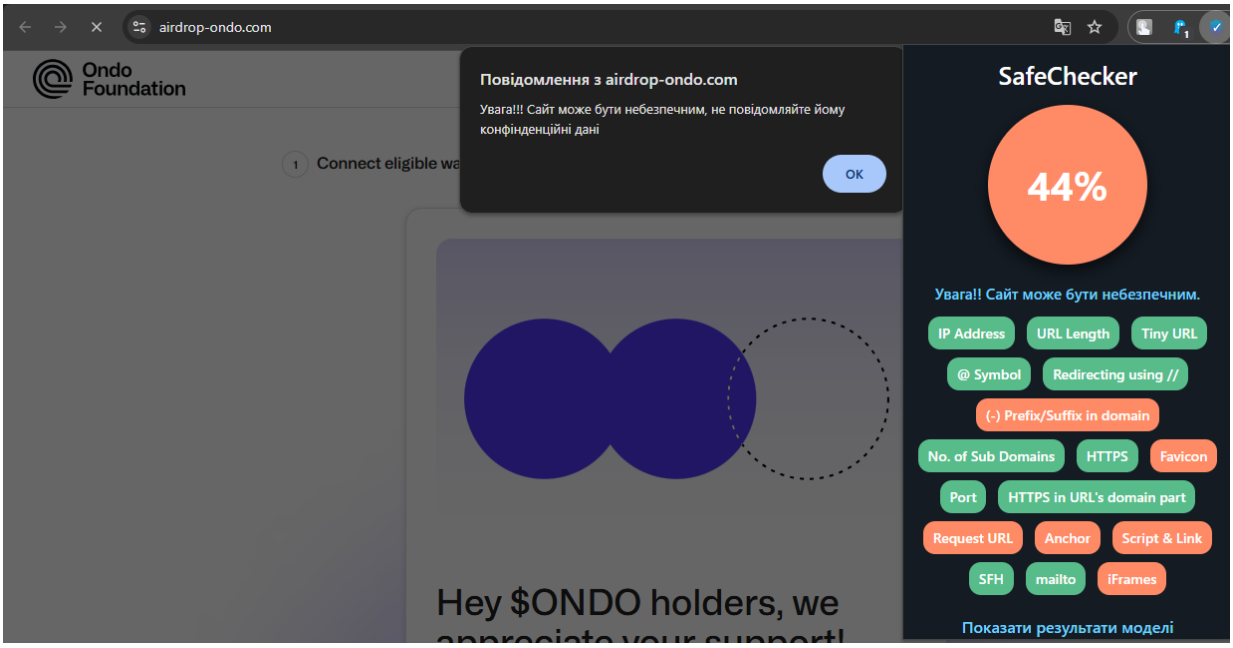



Рисунок 15 - Результат перевірки сайта ondo foundation

Перевірка виявила ознаки префіксів або суфіксів в домені, фавікон, запитів на інші ресурси, анкор посилань, використання скриптів та фреймів і загалом сайт був визначений як небезпечний що є вірно. Під час перевірка сайту з адресою xn--q3ccwc2bxc7erc було помічено, що він поданий у форматі punycode що може часто використовується для маскування фішингових доменів під відомі ресурси. Результат перевірки наведено на рисунку 16.

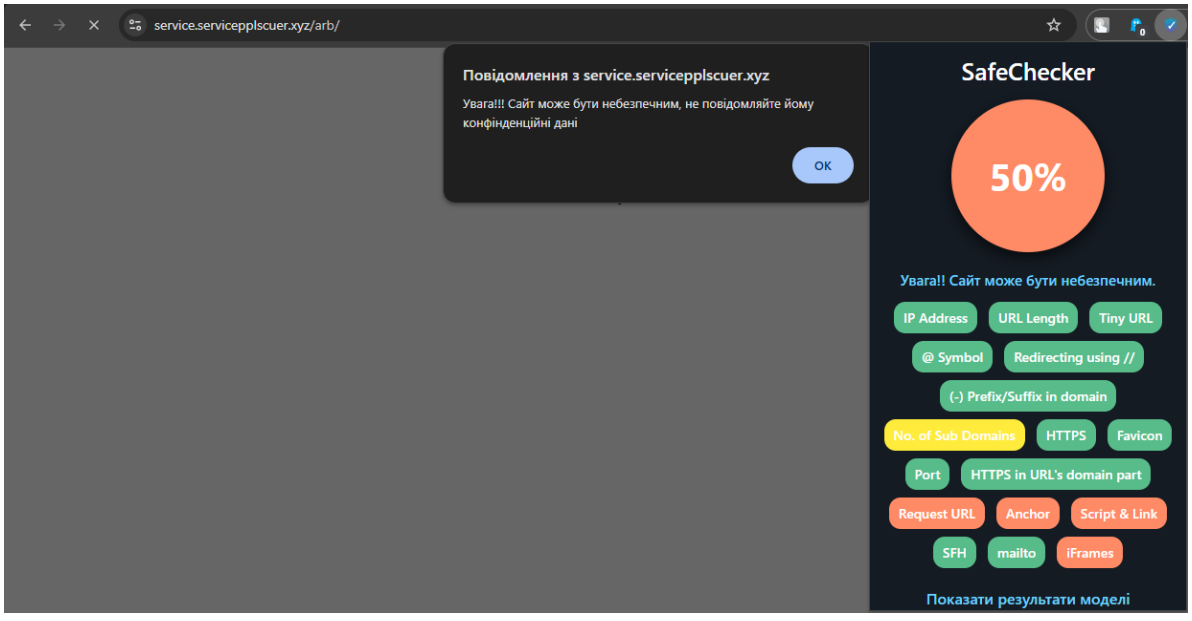



Рисунок 16 - Результат перевірки сайта xn--q3ccwc2bxc7erc

Перевірка виявила ознаки запитів на інші ресурси, анкор посилань, використання скриптів та фреймів. Сайт був визначений як небезпечний що є вірно. Далі було перевірено сайту papasqueens який імітував реальний онлайн магазин, але тут потрібно було вносити передплати за товари що вже може насторожити. Результат тестування наведено на рисунку 17.

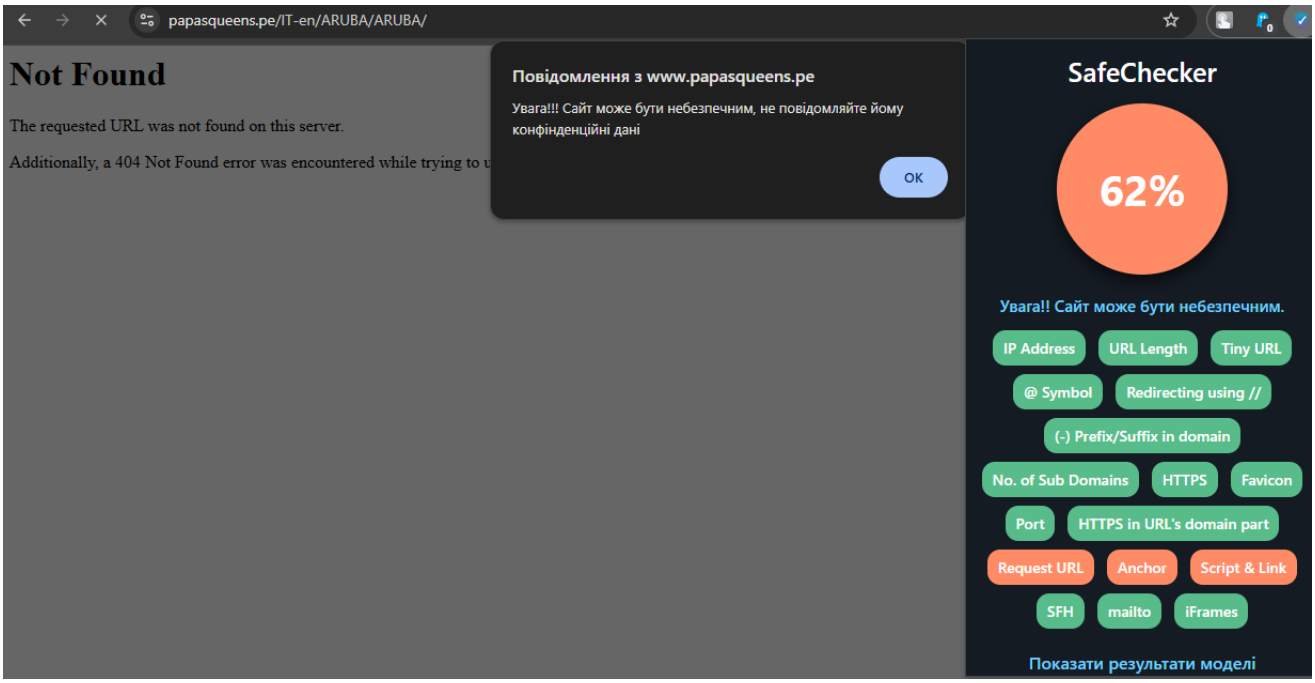



Рисунок 17 - Результат перевірки сайта papasqueens

Хоча сайт під час перевірка вже не працював але розширення змогло виявити ознаки запитів на інші ресурси, анкор посилань та використання скриптів і класифікувати його як небезпечний що є вірно. Далі був сайт stellartokens який має привабливу назву, пов’язану з криптовалютою, що одразу викликає інтерес у користувачів, особливо тих, хто цікавиться блокчейн-проєктами чи інвестуванням. Доменна зона, яку використовує сайт, зазвичай не асоціюється з офіційними або серйозними фінансовими ресурсами, що вже може викликати підозру. Результати тестування наведено на рисунку 18.

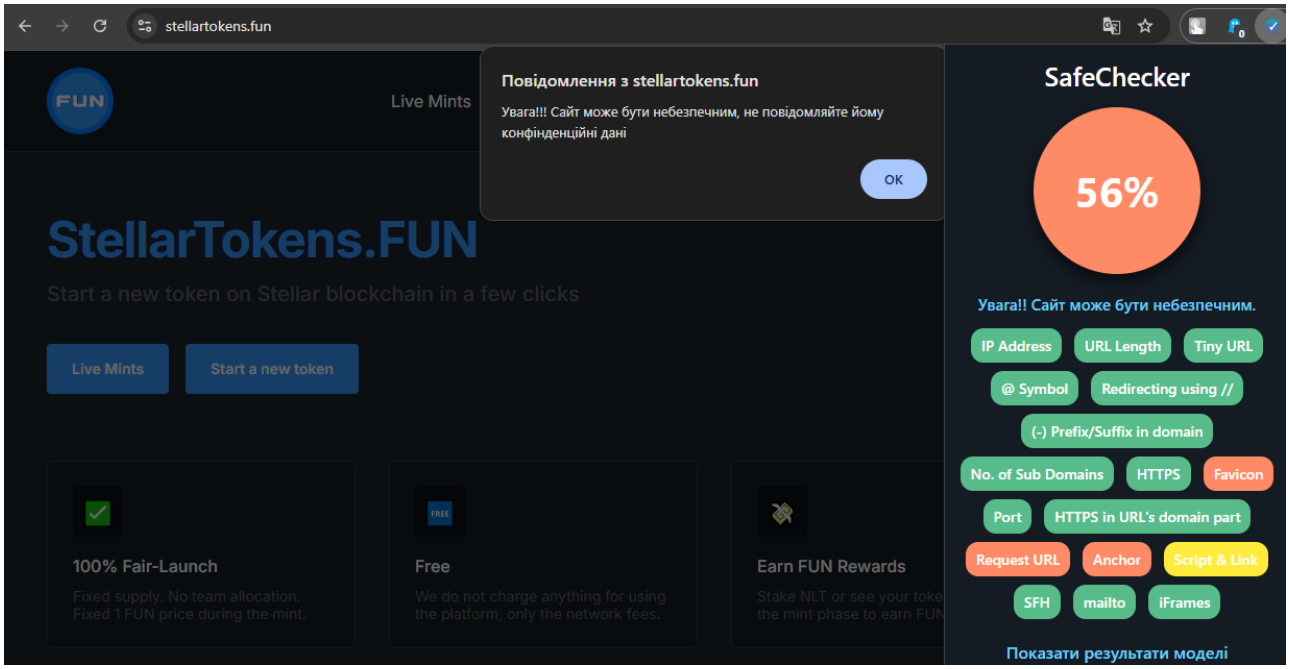



Рисунок 18 - Результат перевірки сайта stellartokens

Перевірка виявила ознаки фавікону, запити на інші ресурси та анкор посилання і класифікувала його як небезпечний що є вірно.

У результаті проведеного тестування стало очевидно, що розроблене браузерне розширення демонструє стабільну та надійну роботу в умовах реального середовища. Усі перевірені безпечні сайти були визначені як безпечні, а шкідливі ресурси як небезпечні. Це підтверджує, що розроблена система може ефективно функціонувати як засіб попередження шкідливих атак і підвищення безпеки користувачів у мережі. Крім того, розширення показало високу швидкість обробки інформації та незначне навантаження на ресурси пристрою. Тестування підтвердило зручність використання інтерфейсу для користувачів з різним рівнем підготовки.

## 3.3 Висновки до розділу

У результаті проведеного тестування та оцінки браузерного розширення для виявлення шкідливих вебсайтів було підтверджено його функціональність, стабільність і відповідність вимогам реального використання. Тестування

охоплювало ресурси з різним ступенем ризику, що дало змогу створити об'єктивне уявлення про функціонування системи в обставинах, наближених до реальних. Алгоритми аналізу ефективно опрацьовували вхідні дані, точно їх інтерпретували та оперативно формували висновки. Результати обробки подавалися у зрозумілому для користувача форматі. Позитивні результати підтверджувалися не лише стабільною роботою інтерфейсу, а й високою швидкодією системи. Завдяки оптимізованій структурі та злагодженій взаємодії між складовими, досягалася узгодженість усіх елементів розширення, зокрема під час опрацювання нестандартних або неоднозначних звернень. Модель демонструвала надійність під час взаємодії з сайтами, що імітують легітимні ресурси або використовують методи соціальної інженерії, що свідчить про її здатність розпізнавати спроби маніпуляцій.

Окремо було протестовано механізм відображення попереджень користувачеві та зручність взаємодії з інтерфейсом. Також було оцінено сумісність з іншими розширеннями, у результаті конфліктів не було виявлено. Це дозволяє говорити що розроблена система має високу взаємодію та універсальність. Оцінка ефективності показала, що дана система є корисним засобом у повсякденному використанні для захисту користувача від загроз які несуть шкідливі вебсайти. Воно не тільки повідомляє про потенційну небезпеку, а й формує більш обережне ставлення до підозрілих вебсайтів. Під час тестування також виявлено потенціал для подальшого розвитку, зокрема удосконалення адаптивності моделі до нових загроз і розширення інтеграції з додатковими джерелами інформації. Отримані результати підтверджують ефективність обраного підходу та демонструють доцільність впровадження розширення в реальні умови експлуатації.

# ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було створено ефективну систему для виявлення шкідливих вебсайтів у режимі реального часу. Вона поєднує в собі збір ознак та класифікацію за допомогою попередньо натренованої моделі машинного навчання. Метою було забезпечити швидкий, зручний інструмент для захисту користувачів під час вебсерфінгу. Важливою умовою була мінімізація навантаження на ресурси браузера, що вдалося досягти за рахунок оптимізованої структури коду. Розробка передбачала собою створення архітектури розширення, визначення вектора характеристик, навчання моделі та організацію взаємодії між складовими. Система пройшла тестування на безпечних та шкідливих сайтах, що дозволило оцінити її точність і стабільність. Було взято до уваги різноманітні варіанти користувацьких дій, зокрема перехід за підозрілими посиланнями та взаємодія з потенційно небезпечним вмістом. Результати показали високу точність виявлення загроз та стійкість до складних сценаріїв.

Таким чином, обране рішення виявилося ефективним у плані швидкодії, точності та інтеграції в браузер. Система здатна виявляти не лише очевидні, а й менш помітні загрози. Також було зосереджено увагу і на зрозумілості інтерфейсу, щоб використання системи було максимально комфортним. Проєкт демонструє успішну інтеграцію машинного навчання в браузерне середовище задля підвищення кібербезпеки. Його можна легко адаптувати до потреб різних категорій користувачів як технічно підготовлених, так і звичайних. Він не потребує з’єднання з сервером, що є перевагою з точки зору продуктивності та конфіденційності. У майбутньому систему можна покращити за рахунок впровадження нових алгоритмів аналізу та вдосконалення механізмів виявлення нових типів загроз. Це свідчить про її актуальність і значний потенціал для подальшого розвитку.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ


1. Приклади фішингових сайтів. Binance. URL: https://www.binance.com/uk-UA/support/faq/detail/360020550592 (дата звернення: 12.02.2025).
2. Визначення атаки drive-by-download. VPN Unlimited. URL: https://www.vpnunlimited.com/ua/help/cybersecurity/drive-by-attack (дата звернення: 12.02.2025).
3. Що таке Ботнет? Як це працює? Визначення та приклади. Gridinsoft. URL: https://gridinsoft.ua/botnet(дата звернення: 12.02.2025).
4. Weamie S. J. Y. Cross-Site Scripting Attacks and Defensive Techniques: A Comprehensive Survey. Independent Researcher, College of Computer Science & Electronic Engineering, Hunan University, Changsha, China, 2022. - 15 с. (дата звернення: 12.02.2025).
5. Визначення експлойт-набору. VPN Unlimited. URL: https://www.vpnunlimited.com/ua/help/cybersecurity/exploit-kit (дата звернення: 13.02.2025).
6. Малвартайзинг. VPN Unlimited. URL: https://www.vpnunlimited.com/ua/help/cybersecurity/malvertising (дата звернення: 13.02.2025).
7. Fadlalla F. F., Elshoush H. T. Input Validation Vulnerabilities in Web Applications: Systematic Review, Classification, and Analysis of the Current State-of-the-Art. Khartoum: University of Khartoum, 2023. 15 с. (дата звернення: 13.02.2025).
8. Що таке ін’єкція HTML. VPN Unlimited. URL: https://www.vpnunlimited.com/ua/help/cybersecurity/html-injection (дата звернення: 13.02.2025).
9. Що таке клікаджакінг. VPN Unlimited. URL: https://www.vpnunlimited.com/ua/help/cybersecurity/clickjacking (дата звернення: 13.02.2025).

10. Ruhani, A. B. B., Zolkipli, M. F. Keylogger: The Unsung Hacking Weapon. J. Appl. Comput. Sci. & Technol., 2023, vol. 6, no. 1. 35-37c. (дата звернення: 13.02.2025).

11. Фальшиві сторінки CAPTCHA атакують користувачів, які шукають піратські комп'ютерні ігри. ProIT. URL: https://proit.ua/falshivi-storinki-captcha-atakuiut-koristuvachiv-iaki-shukaiut-piratski-kompiutierni-ighri/ (дата звернення: 13.03.2025).

12. Фішинг допомагає маніпулювати людьми заради даних. Як захиститися від кіберзагроз. GenTech. URL: https://journal.gen.tech/post/sho-take-socialna-ingeneria (дата звернення: 13.03.2025).

13. Штучний інтелект все частіше використовується для створення фейків. Укрінформ. URL: https://www.ukrinform.ua/rubric-technology/3687330-stucnij-intelekt-vse-castise-vikoristovuetsa-dla-stvorenna-fejkiv-mkip.html (дата звернення: 13.03.2025).

14. Фальшиві оновлення поширюють зловмисне програмне забезпечення. B2B Cyber Security. URL: https://b2b-cyber-security.de/uk/підроблені-оновлення-поширюють-шкідливі-програми/ (дата звернення: 13.03.2025).

15. Атаки типу Man-In-The-Middle: що треба знати кожному. Imena.ua. URL: https://www.imena.ua/blog/man-in-the-middle/ (дата звернення: 13.03.2025).

16. Що таке DDNS (Dynamic Domain Name System)?. Ruby Developers. URL: https://rubydevelopers.org/t/ddns-dynamic-domain-name-system/448 (дата звернення: 13.03.2025).

17. Сервіси скорочення посилань: як вони працюють та як їх використовувати. UABio. URL: https://uabio.me/blog/URL-shorteners (дата звернення: 13.03.2025).

18. Що таке квішинг? Як захиститися від фішингу з QR-кодами. CyberCalm. URL:https://cybercalm.org/novyny/shho-take-quishing/(дата звернення: 13.03.2025).

19. ProgIngContrSystems. Pupenasan. URL:

https://pupenasan.github.io/ProgIngContrSystems/Лекц/6_httpapi.html (дата звернення: 13.03.2025).

20. Що таке URL-адреса сайту? Ukraine.com.ua. URL: https://www.ukraine.com.ua/blog/seo-optimization/chto-takoe-URL-adres-sajta.html (дата звернення: 13.03.2025).

21. Що таке URL-адреса сайту? Wedex. URL: https://wedex.com.ua/blog/shho-take-URL-adresa-sajtu/ (дата звернення: 13.03.2025).

22. Шахрайство з PayPal - популярні схеми обману, як розпізнати?. ESET. URL: https://www.eset.com/ua/about/newsroom/blog/data-protection/naiboleye-populyarnyye-skhemy-moshennichestva-s-paypal-kak-ikh-raspoznat/ (дата звернення: 13.03.2025).

23. Що таке даркнет: переваги та недоліки прихованого сегмента інтернету. Web-Promo. URL: https://web-promo.ua/ua/blog/sho-take-darknet-perevagi-ta-nedoliki-prihovanogo-segmenta-internetu/ (дата звернення: 14.03.2025).

24. Oppliger R. SSL and TLS: Theory and Practice. Third Edition. Norwood: Artech House, 2023. 23-34 с. (дата звернення: 14.03.2025).

25. Три кращі безкоштовні антивірусні програми 2020. Остер. URL: https://oster.com.ua/blog/antivirus (дата звернення: 14.03.2025).

26. Створюємо та публікуємо розширення для Chrome. DOU. URL: https://dou.ua/forums/topic/48605/ (дата звернення: 14.03.2025).

27. Аналіз вмісту сайту: ключові аспекти та методи. WebMate. URL: https://webmate.ua/analiz-contenta-saita (дата звернення: 14.03.2025).

28. Machine Learning, ML. IT.UA. URL: https://www.it.ua/knowledge-base/technology-innovation/machine-learning (дата звернення: 15.03.2025).

29. Zero Trust Architecture (ZTA): A Comprehensive Survey. IEEE Xplore. URL: https://ieeexplore.ieee.org/abstract/document/9773102/metrics#metrics (дата звернення: 15.03.2025).

30. Захищений протокол SSL/TLS – принцип роботи та використання. Hosting.in.ua. URL: https://hosting.in.ua/ua/articles/domeny/zakhyshchenyi-protokol-

ssl-tsl-pryntsyp-roboty-ta-vykorystannia/ (дата звернення: 15.03.2025).

31. Передбачаючи нові домени спаму за допомогою машинного навчання. Unite.AI. URL: https://www.unite.ai/uk/передбачаючи-нові-домени-спаму-за-допомогою-машинного-навчання/ (дата звернення: 15.03.2025).

32. Що таке редирект та як його налаштувати. TheInWeb. URL: https://theinweb.media/chto-takoe-redirecti-i-kak-ih-nastroit/ (дата звернення: 15.03.2025).

33. Блокування сайтів в Україні: як це працює. DOU. URL: https://dou.ua/lenta/articles/blocking-websites/ (дата звернення: 16.03.2025).

34. Web Domain Intelligence: виявляємо та усуваємо шкідливі сайти. KR Labs. URL: https://kr-labs.com.ua/blog/web-domain-intelligence/ (дата звернення: 16.03.2025).

35. Що таке фільтрація пакетів - Терміни та визначення у сфері кібербезпеки. VPN Unlimited. URL: https://www.vpnunlimited.com/ua/help/cybersecurity/packet-filtering (дата звернення: 16.03.2025).

36. Історія домену - історія змін в домені WHOIS. UkrNames. URL: https://www.ukrnames.com/ukr/domain/history/ (дата звернення: 17.03.2025).

37. Як запобігти XSS-атакам. CoreWin. URL: https://corewin.ua/blog/how-to-prevent-xss-attacks (дата звернення: 17.03.2025).

38. Omar A. H. E., Soubra H., Moulla D. K., Abran A. An Innovative Honeypot Architecture for Detecting and Mitigating Hardware Trojans in IoT Devices. 2024. 4-5 с. (дата звернення: 17.03.2025).

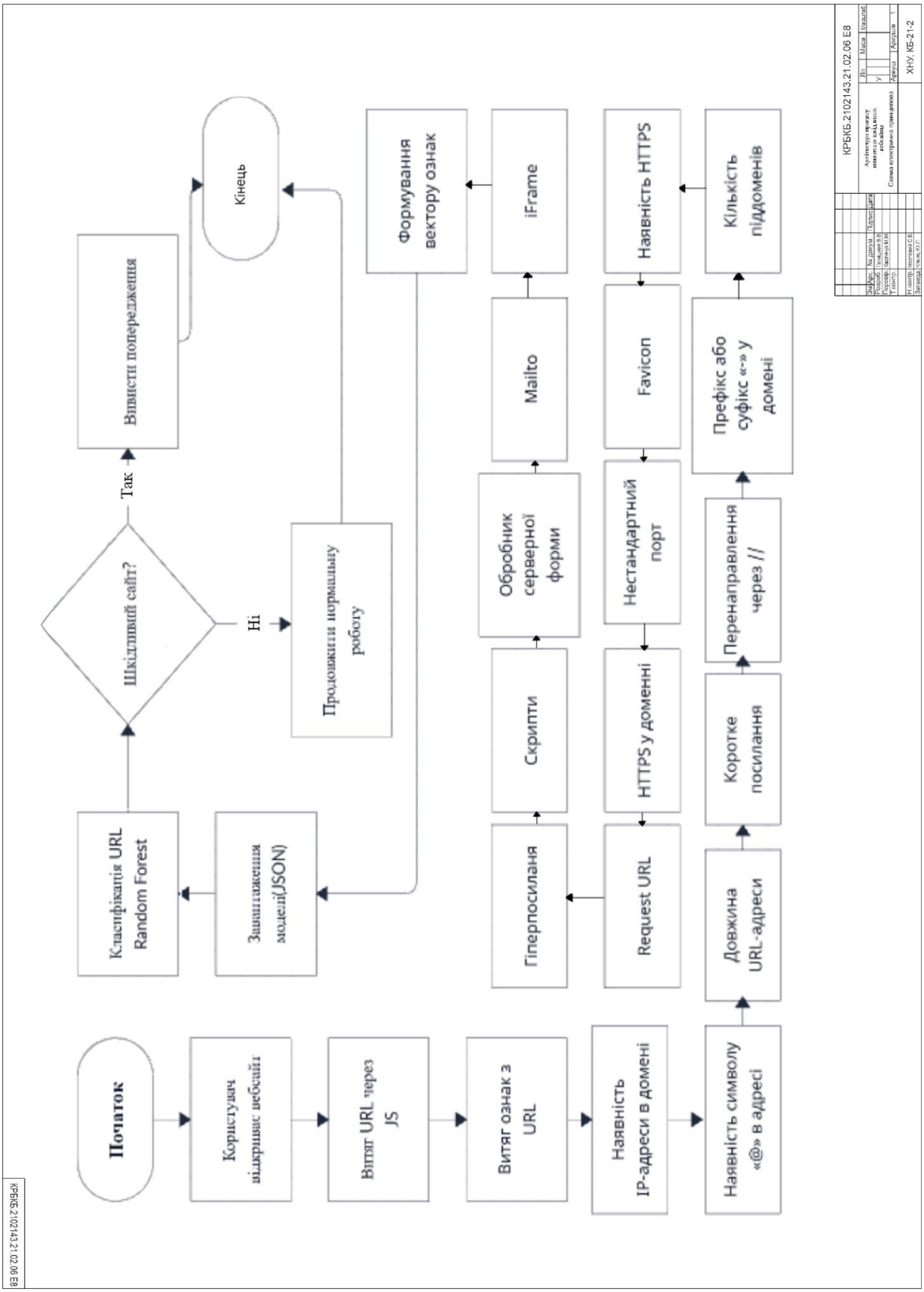
39. Функція шифрування DNS: додатковий рівень захисту чи чергова проблема. ESET. URL: https://www.eset.com/ua/about/newsroom/blog/data-protection/funktsiya-shifrovaniya-dns-dopolnitelnyy-uroven-zashchity-ili-ocherednaya-problema/ (дата звернення: 17.03.2025).

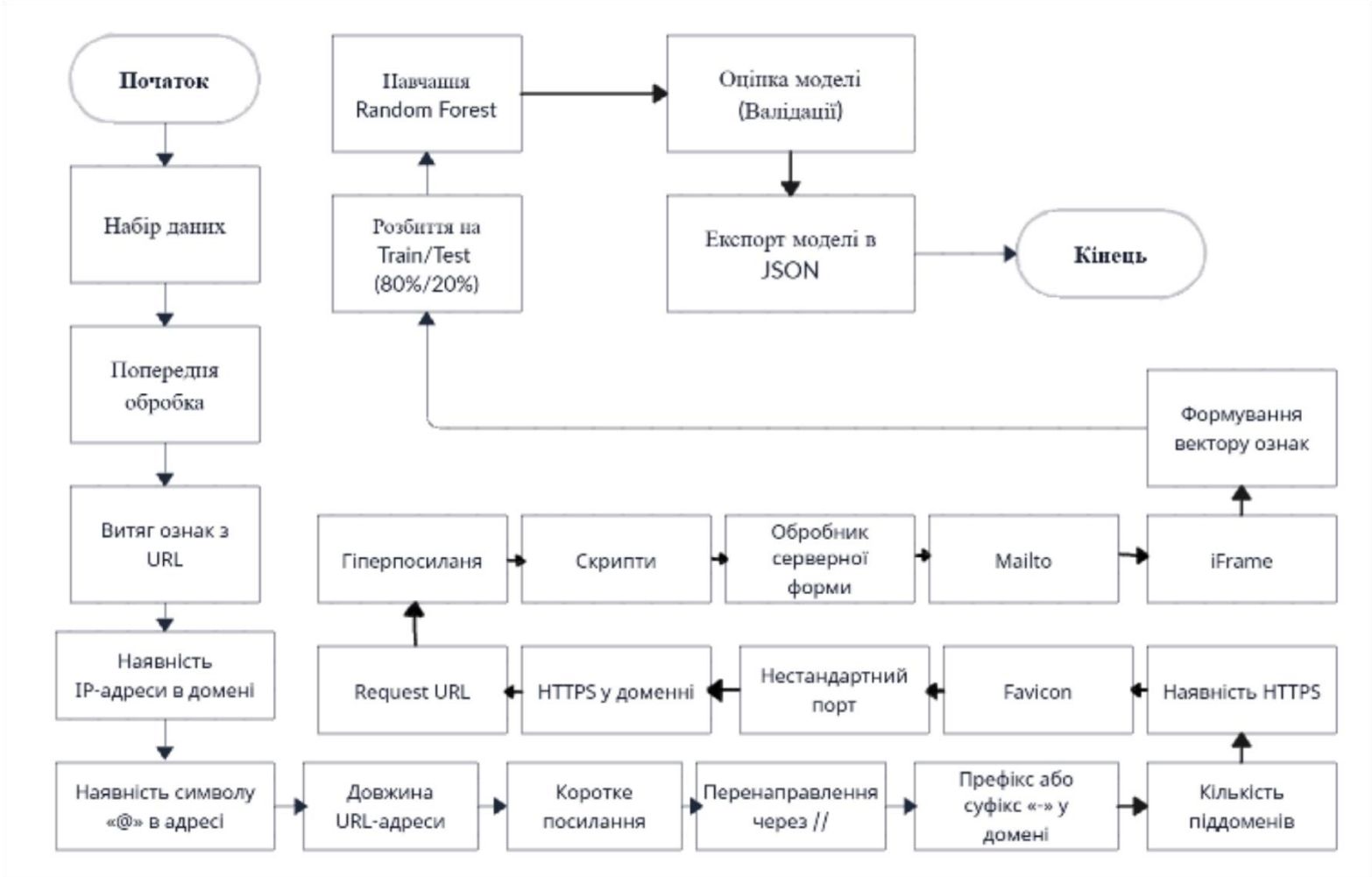
40. What is Random Forest? [Beginner's Guide + Examples]. CareerFoundry. URL:https://careerfoundry.com/en/blog/data-analytics/what-is-random-forest/(дата звернення: 12.04.2025).

ДОДАТОК А

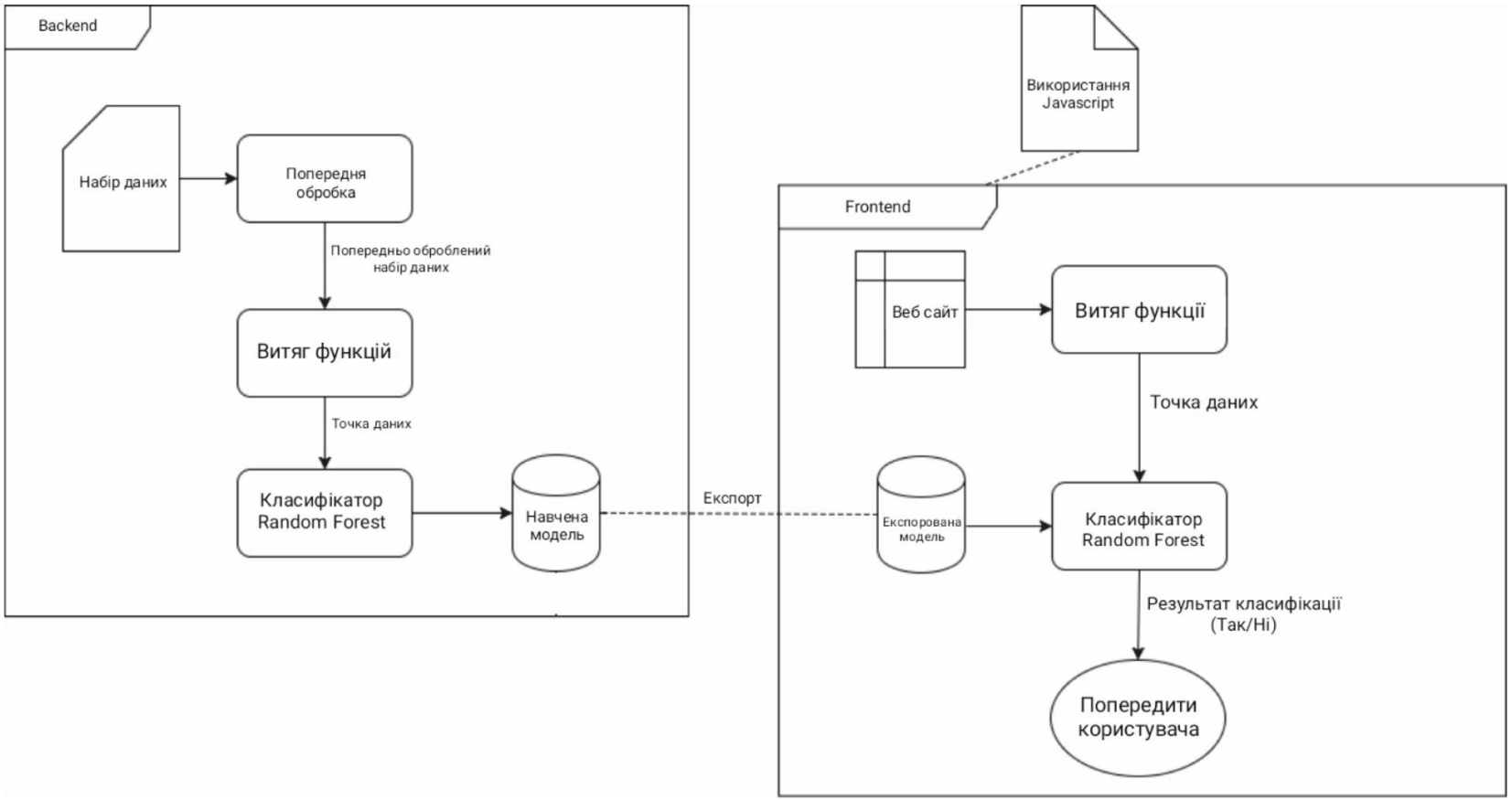
(обов’язковий)

Копія графічної частини






| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | КРБКБ.2102143.21.02.06 Е8 | |
| Зм. | Арк. | № докум. | Підпис | Дата | Архітектура процесу навчання моделі Random Forest | Літ. / Маса / Масштаб |
| Розроб. | | Галецький В.В. | | | Схема електрична принципова | У / / |
| Перевір. | | Касян-ук М.М. | | | | Аркуш / Аркушів 1 |
| Т.контр. | | | | | | |
| Н.контр. | | Мостовий С.В. | | | | ХНУ, КБ-21-2 |
| Затверд. | | Ключа Ю.І. | | | | |




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | КРБКБ.2102143.21.02.06 Е8 | |
| Зм. | Арк. | № докум. | Підпис | Дата | | | |
| Розроб. | | Галицький В.В. | | | Загальна архітектура функціонування системи | Літ. | Маса | Масштаб |
| Перевір. | | Касянчук М.М. | | | | У | | |
| Т. контр. | | | | | Схема електрична принципова | Аркуш | Аркушів 1 | |
| Н. контр. | | Мостовий С.В. | | | | ХНУ, КБ-21-2 | | |
| Затверд. | | Кільов Ю.П. | | | | | | |

Завідувачу кафедри кібербезпеки
к.т.н., доц. Кльоцу Ю.П.
Галицький Володимир Володимирович
ПІБ здобувача вищої освіти

Студента ФІТ, 4 курсу, групи КБ-21-2

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у хмельницькому національному університеті» від 31.08.2023, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (StrikePlagiarism та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

28.05.2025
дата

підпис

# Anti-Plagiarism (UA) v-15.281 Educational

**The maximum coincidence with one document 1.0%**

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 12%**

| ID: 242691<br>Title: Система виявлення шкідливих вебсайтів на основі аналізу URL-адрес<br>Added in a DB: 2025-05-30<br>Authors: Галицький Володимир Володимирович<br>Heads: Касянчук М.М.<br>Consultants:<br>Opponents: | Document | | Sum coincidence on the DB | |
|---|---|---|---|---|
| | Symbols | Lexemes | Symbols | Lexemes |
| | 94936 | 767 | 1550 (2%) | 21 (3%) |

Plagiarism sources

| ID | Description | Plagiarism presence in the document | |
|---|---|---|---|
| | | Symbols | Lexemes |

**Протокол аналізу звіту подібності науковим керівником**

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

**Автор:** Галицький Володимир Володимирович

**Співавтор:**

**Назва:** Система виявлення шкідливих вебсайтів на основі аналізу URL-адрес

**Науковий керівник:**

**Підрозділ:** Кафедра кібербезпеки

**Коефіцієнт подібності 1:**2.2%

**Коефіцієнт подібності 2:**0.4%

**Мікропробіли:** 0

**Заміна букв:** 0

**Інтервали:** 0

**Білі знаки:** 0

**Дата створення звіту:** 2025-05-30 23:58:49.0

**Після аналізу Звіту подібності констатую наступне:**

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

02.06.2025р.

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:
Назва: Система виявлення шкідливих вебсайтів на основі аналізу URL-адрес

Автор: Галицький Володимир Володимирович
Спеціальність: 125 – Кібербезпека
Освітня програма: освітньо-професійна
Науковий керівник: Касянчук Михайло Миколайович, д.т.н, професор
Після аналізу звіту подібності зроблено такий висновок:

| № | Висновок | Позначка про відповідність |
|---|---|---|
| 1 | Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту. | відповідає |
| 2 | Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи | |
| 3 | Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат. | |
| 4 | Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту. | |

Підтвердження:

Оригінальність тексту роботи за результатами перевірки системою StrikePlagiarism складає 97,8%, оригінальність тексту роботи за результатами перевірки системою Anti-Plagiarism v-15.257 складає 99%.

Згідно з правилами чинного Положення «Про систему забезпечення академічної доброчесності у хмельницькому національному університеті» від 31.08.2023, авторська робота, обсяг оригінального тексту у відсотках до загального обсягу матеріалу в якій складає 90-100 %, визнається роботою з високою унікальністю тексту і допускається до захисту.

Керівник роботи _____ М.М. Касянчук

Гарант ОП _____ В.М. Чешун

Завідувач кафедри кібербезпеки _____ Ю.П. Кльоц

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

# РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «бакалавр»

Студент Галицький Володимир Володимирович

Тема Система виявлення шкідливих вебсайтів на основі аналізу URL-адрес

Спеціальність 125 – Кібербезпека

**Обсяг кваліфікаційної роботи освітньо-кваліфікаційного рівня «бакалавр»:**

кількість листів креслень 3; кількість сторінок записки 62.

1. Короткий зміст роботи та прийнятих рішень У кваліфікаційній роботі було розроблено систему виявлення шкідливих вебсайтів на основі аналізу URL-адрес. Система реалізована з використанням алгоритму машинного навчання Random Forest, який дозволяє класифікувати вебсайти на шкідливі та безпечні виключно за структурою та особливостями URL. Це дає змогу оперативно реагувати на нові загрози без необхідності аналізу вмісту сайту. Модель була інтегрована у браузерне розширення, що надає зручний інтерфейс для автоматичного попередження користувача про загрозу при переході на потенційно шкідливий сайт.

2. Висновок про відповідність кваліфікаційної роботи завданню У кваліфікаційній роботі було виконано поставлене завдання як у теоретичній, так і в практичній частині.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі розглянуто сучасні типи шкідливих вебсайтів, їх загрози, а також проведено аналіз структури URL-адрес. У другому розділі описано побудову ознак URL, використання алгоритму Random Forest для навчання моделі, а також інтеграцію цієї моделі в браузерне розширення. У третьому розділі здійснено тестування ефективності, зокрема на реальних даних, проаналізовано точність, повноту, акуратність, F1-міру, і наведено графічні результати.

4. Позитивні сторони роботи Кваліфікаційна робота має практичну спрямованість та високу актуальність. Алгоритм Random Forest забезпечує хорошу інтерпретованість та надійність класифікації.

5. Негативні сторони роботи У реалізації браузерного розширення використано Manifest V2, яка втратила підтримку у нових версіях браузера Google Chrome відповідно до політики Manifest V3, що обмежує перспективність використання розширення в довгостроковому періоді.

6. Оцінка графічного оформлення та пояснювальної записки роботи Графічне оформлення кваліфікаційної роботи відповідає темі роботи та виконане з дотриманням стандартів. В цілому, графічне оформлення є якісним, а пояснювальна записка відповідає нормам оформлення.

7. Відгук про роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки, оскільки тема виявлення шкідливих вебсайтів на основі аналізу URL-адрес є актуальною, а обраний підхід із застосуванням алгоритму Random Forest є виправданим та ефективним. Модель інтегрована у браузерне розширення, що свідчить про практичну цінність розробки. Структура роботи є логічною, кожен розділ розкриває окремий етап проєкту від теоретичних основ до тестування системи. Графічний матеріал добре ілюструє результати і сприяє кращому розумінню прийнятих технічних рішень.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Враховуючи всі позитивні та негативні сторони представленої кваліфікаційної роботи, можна зробити висновок, що вона заслуговує оцінки «добре».

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи)
Підченко Сергій Констянтинович, завідувач кафедри телекомунікацій, медійних та інтелектуальних технологій, доктор технічних наук, професор ХНУ.

«02» 06 2025 (підпис)