Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки та захисту інформації
_____ Іван ПАРХОМЕНКО
«___» червня 2025 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)
спеціальність 125 Кібербезпека
(код і назва спеціальності)
освітній ступень бакалавр
освітня програма Кібербезпека
(назва освітньо-професійної програми)
на тему: «Система аналізу атак соціальної інженерії»

Виконавець: студентка IV курсу, групи КБ-42

_____ **Діана ГУЗОВАТА**
(підпис) (ім'я, прізвище)

	Підпис	**Ім'я, прізвище**
Керівник		Олександр ТОРОШАНКО
Нормоконтроль		Юрій БАБЕНКО

Київ 2025

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«29» листопада 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньо-професійної програми)

Студентці **КБ-42** **Гузоватій Діані Олександрівні**
(група) (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Система аналізу атак соціальної інженерії

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №6 від 28.11.2024 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Методи соціальної інженерії, алгоритми машинного навчання, ознаки фішингових доменів, симетричне шифрування AES

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Необхідно дослідити типи соціоінженерних атак, методи їх виявлення, проаналізувати ефективність класифікаційних алгоритмів, розробити систему прогнозування фішингу за ознаками доменів, реалізувати механізм шифрування службових даних, а також виконати оцінку точності моделі.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Розроблена система виявлення фішингових атак на основі машинного навчання з інтегрованим захистом даних, придатна для використання у кіберзахисних модулях і навчальному процесі.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 29 листопада 2024 року

Завдання видав (підпис) Олександр ТОРОШАНКО (ім'я, прізвище)

Завдання прийняла до виконання (підпис) Діана ГУЗОВАТА (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	29.11.2024 – 22.01.2024	*виконано*
2	Аналіз літератури	29.01.2025 – 11.02.2025	*виконано*
3	Обґрунтування вибору рішення	12.02.2025 – 15.02.2025	*виконано*
4	Формування навчальної вибірки, аналіз ознак, побудова сценаріїв атак	16.02.2025 – 04.03.2025	*виконано*
5	Проєктування архітектури та реалізація програмного модуля системи	05.03.2025 – 21.03.2025	*виконано*
6	Інтеграція алгоритму шифрування AES для захисту службових даних	22.03.2025 – 08.04.2025	*виконано*
7	Проведення тестування системи, аналіз точності та продуктивності моделі	09.04.2025 – 10.05.2025	*виконано*
8	Оформлення пояснювальної записки	11.05.2025 – 27.05.2025	*виконано*
9	Підготовка до захисту кваліфікаційної роботи	28.05.2025 – 07.06.2025	*виконано*

Завдання видав _____ (підпис) Олександр ТОРОШАНКО (ім'я, прізвище)

Завдання прийняла до виконання _____ (підпис) Діана ГУЗОВАТА (ім'я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 13 червня 2025 року

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, додатків, має 83 сторінок основного тексту, 37 рисунків та 9 таблиць. Список використаних джерел містить 30 найменувань і займає 3 сторінки.

Методи дослідження кваліфікаційної роботи:

- системний підхід;
- методи порівняння;
- методи машинного навчання;
- структурно-функціональний аналіз;
- аналіз ознак фішингових доменів;
- емпіричне тестування;
- аналіз метрик якості класифікації.

Об'єктом дослідження є процес виявлення та аналізу атак соціальної інженерії у цифровому інформаційному середовищі.

Предметом дослідження в даній роботі є методи, алгоритми та інструменти інтелектуального аналізу.

У роботі проаналізовано підходи до протидії соціоінженерним атакам та оцінено ефективність класифікаційних алгоритмів для виявлення фішингу. Побудовано модель на основі швидкого дерева рішень, яка демонструє високу точність розпізнавання доменів з ознаками загрози.

Розроблено програмну систему, що автоматично аналізує домени в реальному часі, надає інтерпретовані прогнози й захищає дані користувача за допомогою алгоритму AES. Отримані результати придатні для використання в практичних системах кіберзахисту та навчальному процесі з кібербезпеки.

Ключові слова: соціальна інженерія, фішинг, машинне навчання, доменне ім’я, індекс Жаккара, інтелектуальна система захисту, швидке дерево рішень, AES.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

AES	–	Advanced Encryption Standard
ATT&CK	–	Adversarial Tactics, Techniques, and Common Knowledge
AUC	–	Area Under Curve
BEC	–	Business Email Compromise
DLP	–	Data Loss Prevention
DNS	–	Domain Name System
IAM	–	Identity and Access Management
MFA	–	Multi-Factor Authentication
PoLP	–	Principle of Least Privilege
SMS	–	Short Message Service
SOAR	–	Security Orchestration, Automation and Response
URL	–	Uniform Resource Locator
ВЛ	–	Метод випадкових лісів
ГБ	–	Метод градієнтного бустингу
ШД	–	Метод швидкого дерева

ЗМІСТ

ВСТУП

Актуальність. Однією з найнебезпечніших форм порушення інформаційної безпеки сучасних організацій і приватних осіб залишається вплив, що здійснюється не шляхом технічного злому, а через психологічне маніпулювання – зокрема, шляхом так званих атак соціальної інженерії. Зважаючи на активну цифровізацію суспільства, розширення спектру інформаційно-комунікаційних послуг та динамічний розвиток технологій, спостерігається зростання кількості інцидентів, що пов’язані з шахрайськими методами отримання несанкціонованого доступу до конфіденційної інформації, облікових даних та інфраструктурних ресурсів. Уразливими є не лише окремі користувачі, а й цілі корпоративні системи, державні установи, освітні заклади, медичні організації.

Проблема ускладнюється тим, що подібні атаки часто не залишають технічного сліду, а отже, є важко виявлюваними традиційними засобами кібербезпеки. Успішність соціально-інженерних атак базується на психологічних особливостях сприйняття інформації людиною, зокрема на довірливості, необізнаності або перевантаженості інформаційним потоком. До найпоширеніших форм таких атак належать фішинг, вішинг, смішинг, бейтинг, а також атаки через підроблені соціальні профілі, вебсайти чи електронні листи.

Актуальність вирішення проблеми обумовлена необхідністю формування ефективної, адаптивної та масштабованої системи виявлення потенційних загроз, спричинених людським фактором. Зростання кількості цілеспрямованих фішингових кампаній, що використовують методи соціальної інженерії з високим ступенем персоналізації, зокрема із застосуванням штучного інтелекту, лише підкреслює необхідність пошуку нових підходів до аналізу таких загроз.

У світовій науковій спільноті активно проводяться дослідження в галузі кібербезпеки під керівництвом провідних фахівців. Зокрема, заслуговують на увагу роботи таких вчених, як Кевін Мітнік – один із перших дослідників і популяризаторів тематики соціальної інженерії, який згодом став фахівцем із захисту інформації;

Крістофер Хаднагі – автор численних публікацій про психологічні механізми впливу в кіберсфері, а також дослідників MITRE, які займаються моделюванням соціально-інженерних загроз у рамках проєкту ATT&CK [1]. Автори Соколов В. та Курбанмурадов Д. намагаються посилити дослідження щодо ідентифікації та контролю загроз соціальної інженерії шляхом ілюстрованої класифікації, яка включає типи атак, визначення ступеня можливих збитків для кожного типу відомих атак та заходи протидії загрозам, що призводять до втрати особистої або корпоративної конфіденційної інформації [2].

Попри значний прогрес у вивченні механізмів соціальних атак, існує низка невирішених задач, серед яких: формалізація ознак фішингових впливів у цифровому середовищі, автоматизоване виявлення фейкових електронних ресурсів, аналіз поведінкових характеристик користувачів, що потрапили під вплив, а також розробка систем, здатних в режимі реального часу виявляти та блокувати соціально-інженерні атаки на основі даних з різних джерел.

Враховуючи вищезазначене, виникає потреба у створенні програмно-аналітичних рішень, здатних ідентифікувати ознаки фішингу та інших форм маніпулятивного впливу на основі інтелектуального аналізу даних. Одним із перспективних напрямів є застосування алгоритмів машинного навчання, здатних здійснювати класифікацію доменів за ступенем їх небезпеки на основі комплексу семантичних, структурних та статистичних характеристик. Такий підхід дозволяє вийти за межі статичного списку небезпечних доменів і забезпечити динамічне реагування на нові загрози. Водночас, важливим є не лише виявлення підозрілих ознак, а й формування інтерпретованих рекомендацій для користувача з метою попередження потенційних інцидентів.

Метою даної роботи є розроблення програмної системи, здатної здійснювати автоматизований аналіз загроз соціальної інженерії з акцентом на виявлення фішингових атак на основі інтелектуальної обробки даних.

Досягнення поставленої мети вимагає вирішення таких основних задач:

– провести огляд форм атак соціальної інженерії, класифікувати їх за типами впливу та визначити методи, які найчастіше застосовуються зловмисниками;

– виконати аналіз існуючих підходів до виявлення та запобігання атакам соціальної інженерії;

– спроєктувати архітектуру та реалізувати систему, яка здійснює аналіз потенційно небезпечних доменів;

– провести експериментальне тестування реалізованої системи, оцінити точність, повноту, продуктивність та надійність виявлення фішингових атак.

Об’єкт дослідження: процес виявлення та аналізу атак соціальної інженерії у цифровому інформаційному середовищі.

Предмет дослідження: методи, алгоритми та інструменти інтелектуального аналізу, що забезпечують розпізнавання фішингових атак на основі характеристик доменів та інших ознак соціально-інженерного впливу.

Методи дослідження: системний підхід, методи порівняння, методи машинного навчання, структурно-функціональний аналіз, метод аналізу ознак фішингових доменів, емпіричне тестування, аналіз метрик якості класифікації.

Наукова новизна. У роботі запропоновано вдосконалену методику виявлення фішингових атак на основі класифікаційної моделі, що враховує комплекс ознак доменного імені, включаючи структурні, статистичні та семантичні характеристики. Вперше реалізовано систему, яка інтегрує алгоритми машинного навчання з засобами інтерпретації результатів прогнозування для автоматизованої оцінки ризику соціально-інженерних впливів у реальному часі.

Практичне значення одержаних результатів. Розроблено програмну систему для автоматизованого аналізу фішингових атак на основі характеристик доменних імен з використанням алгоритмів машинного навчання. Система дозволяє здійснювати виявлення потенційно небезпечних доменів у реальному часі, надаючи користувачеві інтерпретовану інформацію про рівень загрози. Отримані результати можуть бути використані для побудови модулів кіберзахисту в організаціях, а також у навчальному процесі при вивченні дисциплін з інформаційної безпеки та аналізу кіберзагроз.

РОЗДІЛ 1.
СУЧАСНИЙ СТАН ПРОБЛЕМИ СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ

1.1 Аналіз сучасних методів атак соціальної інженерії

Атаки соціальної інженерії посідають одне з провідних місць серед загроз інформаційній безпеці у цифрову епоху. Їх особливість полягає у тому, що зловмисник, замість використання складних технічних засобів, експлуатує людський фактор – довіру, неуважність, психологічну вразливість, нестачу знань або досвіду. Успішність таких атак базується на переконливій імітації авторитетних джерел або сценаріїв, які змушують жертву добровільно надати конфіденційні дані, встановити шкідливе програмне забезпечення або виконати інші дії, що порушують політику безпеки.

До найбільш поширених видів атак соціальної інженерії належать фішинг, вішинг, смішинг, бейтінг, скерований фішинг (spear phishing), компрометація ділової переписки (Business Email Compromise, BEC), атаки через соціальні мережі, а також атакуючі дії на фізичному рівні, що включають в себе проникнення до об’єктів або залишення шкідливих носіїв у зоні доступу персоналу [3].

Фішинг є однією з найпоширеніших форм атак соціальної інженерії, що спрямовані на викрадення конфіденційних даних користувачів, таких як логіни, паролі, платіжні реквізити та інші облікові дані. Головною метою фішингових атак є обман жертви шляхом імітації довірених онлайн-ресурсів, таких як банківські сервіси, корпоративні вебсайти, поштові платформи або соціальні мережі [4]. Актуальність цієї загрози значно зросла через використання методів персоналізації атак, при яких зловмисники аналізують інформацію про жертву, щоб зробити свої повідомлення максимально переконливими. Крім того, сучасні фішингові атаки все частіше використовують автоматизовані інструменти, включаючи генерацію підроблених доменів та підміни IP-адрес, що ускладнює їхнє виявлення. У сфері аналізу атак соціальної інженерії фішинг посідає особливе місце, оскільки є одним із

основних методів ініціації зламу інформаційних систем, поширення шкідливого програмного забезпечення та здійснення фінансових махінацій. Для ефективного виявлення фішингових атак використовуються алгоритми машинного навчання, які аналізують характеристики доменних імен, зміст електронних листів, структуру вебсайтів та поведінкові моделі користувачів.

На рис. 1.1 [5] наведена схема реалізації типової фішингової атаки, яка демонструє послідовність дій атакувальника та жертви в межах цього процесу.

Рисунок 1.1 – Схема фішингової атаки

Атакувальник ініціює атаку, надсилаючи жертві електронний лист, що містить фішингове посилання. Цей лист зазвичай імітує офіційне повідомлення від авторитетної організації, наприклад, банку, служби підтримки чи корпоративного відділу безпеки. Жертва, отримавши таке повідомлення, натискає на посилання, яке перенаправляє її на фішинговий вебсайт. Цей вебсайт є точною або майже точною

копією справжнього ресурсу, що вводить жертву в оману та спонукає ввести свої облікові дані. Коли користувач вводить свої логін та пароль, ці дані автоматично передаються атакувальнику. Зловмисник отримує доступ до конфіденційної інформації та може використовувати її для проникнення в особисті акаунти, здійснення фінансових операцій або подальшого розповсюдження атак.

Фішингові атаки залишаються однією з головних загроз кібербезпеці через їхню простоту реалізації та високу ефективність. Вони здатні обходити традиційні системи захисту, використовуючи техніки соціальної інженерії та методи маніпуляції поведінкою користувачів. Успішна протидія таким атакам вимагає комплексного підходу, що включає як технічні методи аналізу фішингових доменів, так і підвищення рівня обізнаності користувачів щодо потенційних загроз.

Цільовий фішинг (spear phishing) є однією з найнебезпечніших форм соціально-інженерних атак, оскільки орієнтований на конкретних осіб або організації, а не на масову аудиторію [6]. Ця атака базується на ретельному зборі інформації про жертву, що дозволяє зловмисникам створювати надзвичайно переконливі повідомлення, які важко відрізнити від справжніх. Основна мета цільового фішингу – змусити користувача відкрити шкідливе вкладення, ввести свої облікові дані на підробленому вебсайті або завантажити програму, що містить шкідливий код. Такі атаки можуть використовуватися для викрадення корпоративних або урядових даних, шпигунства, фінансових махінацій або навіть для ініціації складніших атак на інформаційну інфраструктуру підприємства. Висока персоналізація робить цільовий фішинг ефективним, оскільки жертви менше підозрюють про загрозу.

У контексті аналізу атак соціальної інженерії цей тип загроз є критично важливим, оскільки часто стає першою фазою складних багаторівневих атак, що можуть призвести до масштабних порушень безпеки. Зокрема, успішна смішинг-атака може надати зловмисникам доступ до мобільного банкінгу жертви, корпоративної електронної пошти або двофакторної автентифікації, що значно розширює вектор подальших атак. Наприклад, отримавши контроль над мобільним пристроєм, атакувальник може перехоплювати коди підтвердження з банківських сервісів або навіть здійснювати переказ коштів без відома власника.

На рис. 1.2 [7] наведено схему реалізації типової цільової фішингової атаки, яка демонструє, як зловмисник поступово отримує контроль над системою жертви та використовує її для подальших дій.

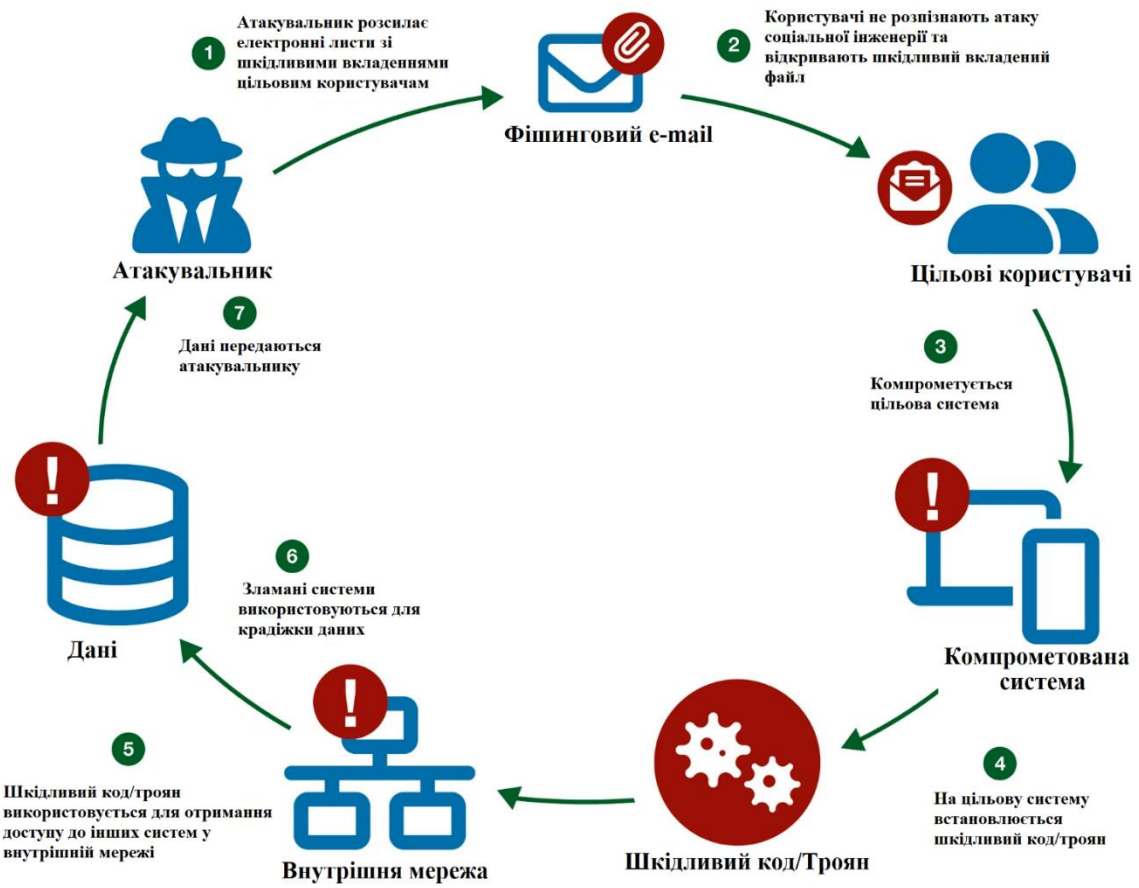

Рисунок 1.2 – Схема цільового фішингу

Атакувальник ініціює атаку, розсилаючи електронні листи із шкідливими вкладеннями або посиланнями на певних користувачів, які є його основною ціллю. Ці повідомлення маскуються під офіційну переписку, що викликає довіру та не викликає підозр у жертви. Користувач відкриває вкладений файл, що містить шкідливий код, або переходить за посиланням, яке запускає процес компрометації його системи. Внаслідок цього встановлюється троян або інше шкідливе програмне забезпечення, що надає атакувальнику можливість отримати контроль над пристроєм жертви. Після успішного зараження зловмисник використовує уражену систему для проникнення у внутрішню мережу організації, отримання доступу до критично важливих даних і їх викрадення. Викрадені дані передаються атакувальнику, який може використовувати

їх для фінансових шахрайств, промислового шпигунства або подальшого розповсюдження атак на інших користувачів.

Цільовий фішинг є складною та небезпечною загрозою, оскільки дозволяє зловмисникам отримати доступ до захищених систем без необхідності ламати складні технічні механізми захисту. Він є особливо ефективним у корпоративному та урядовому середовищі, де компрометація однієї ланки може призвести до масового витоку інформації.

Вішинг (vishing, voice phishing) – це вид соціальної інженерії, в якому зловмисники використовують телефонні дзвінки або голосові повідомлення для отримання конфіденційної інформації [8]. Основна мета цієї атаки – змусити жертву передати облікові дані, фінансову інформацію або виконати небезпечні дії, такі як встановлення шкідливого програмного забезпечення. На відміну від традиційного фішингу, вішинг використовує маніпуляцію в реальному часі, що робить його особливо небезпечним. Сучасні технології, зокрема VoIP-сервіси, дозволяють зловмисникам підміняти номери телефонів, імітуючи дзвінки від банків, державних установ або внутрішніх служб компаній. Це суттєво підвищує рівень довіри жертви до атакуючого.

Даний метод соціальної інженерії є серйозною загрозою, оскільки він часто використовується як перший етап складних комбінованих атак, зокрема у поєднанні з фішингом та смішингом (фішинг через SMS). Наприклад, шахраї можуть спочатку здійснити телефонний дзвінок, видаючи себе за представника банку або технічної підтримки, а потім надіслати SMS з посиланням на підроблений вебсайт для підтвердження особи. У деяких випадках атака може включати голосові повідомлення з автоматизованих систем, які інформують користувача про нібито підозрілу активність на його рахунку, що змушує жертву діяти поспіхом. Крім того, зловмисники часто застосовують методи соціальної маніпуляції, такі як нагнітання паніки або створення відчуття терміновості, щоб жертва не встигла критично оцінити ситуацію. Одним із найбільш небезпечних сценаріїв є використання вішингу для викрадення даних двофакторної автентифікації, що дозволяє зловмисникам отримати

доступ до банківських акаунтів або корпоративних систем без необхідності ламати складні паролі.

На рис. 1.3 [9] представлено схему реалізації типової вішингової атаки, яка демонструє основні етапи взаємодії між зловмисником та жертвою.

Рисунок 1.3 – Схема вішингової атаки

Атакувальник ініціює атаку, телефонуючи жертві через VoIP-сервіс, підміняючи номер на офіційний номер банку або іншої авторитетної установи. Для посилення ефекту він може використовувати записи автоматизованих голосових відповідей або підроблені цифрові ідентифікатори. Після встановлення контакту з жертвою зловмисник надсилає їй SMS із фішинговим посиланням, переконуючи, що це офіційний запит на верифікацію даних або виправлення безпекової проблеми. Жертва читає повідомлення та переходить на підроблений вебсайт, створений атакувальником для збору облікових даних. Ввівши логін та пароль, користувач фактично передає цю інформацію шахраю, який одразу отримує доступ до особистого кабінету жертви. В окремих сценаріях зловмисник може переконати жертву завантажити та виконати шкідливий файл, що відкриває йому віддалений доступ до системи. Після виконання шкідливого коду атакувальник отримує підтвердження успішності атаки та використовує отриману інформацію для подальших дій, таких як фінансові махінації або атаки на корпоративну мережу.

Вішинг є особливо небезпечним видом соціальної інженерії, оскільки використовує прямий психологічний вплив, змушуючи жертву діяти негайно. Поєднання голосового контакту, підроблених повідомлень та технічних маніпуляцій значно ускладнює виявлення атаки традиційними засобами кіберзахисту. Ефективна протидія вішингу вимагає як технологічних рішень, так і підвищення обізнаності користувачів щодо методів маніпуляції, які застосовують зловмисники.

Смішинг (smishing, SMS phishing) – це різновид фішингових атак, у яких зловмисники використовують SMS-повідомлення для обману жертви з метою викрадення конфіденційних даних або встановлення шкідливого програмного забезпечення [10]. На відміну від традиційного фішингу, який здійснюється через електронну пошту, смішинг є більш ефективним, оскільки користувачі частіше довіряють повідомленням на мобільних пристроях. Сучасні смішингові атаки часто маскуються під банківські сповіщення, повідомлення від служб безпеки або технічної підтримки, що створює ілюзію легітимності. Основна мета такої атаки – змусити жертву перейти за шкідливим посиланням або завантажити заражене програмне забезпечення, що призводить до компрометації мобільного пристрою. З огляду на те, що мобільні телефони дедалі частіше використовуються для багатофакторної автентифікації та фінансових операцій, атаки смішингу можуть мати серйозні наслідки для безпеки особистих даних та корпоративних систем.

На рис. 1.4 [11] наведено схему реалізації типової смішингової атаки, яка демонструє основні етапи проникнення зловмисника в систему жертви.

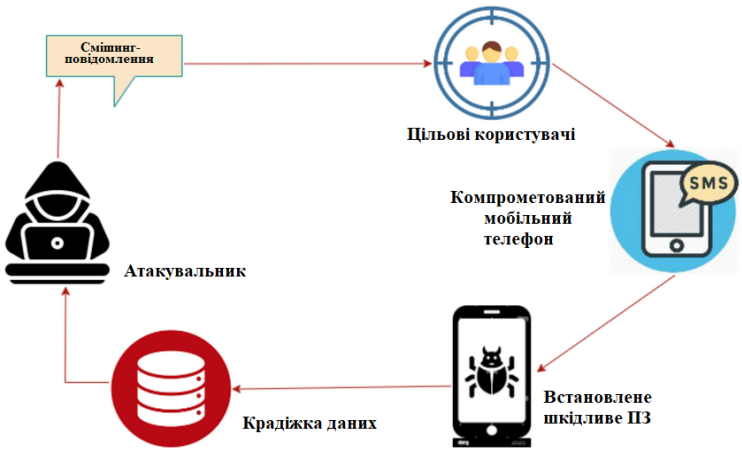

Рисунок 1.4 – Схема смішингової атаки

Атакувальник надсилає SMS-повідомлення цільовим користувачам, у якому міститься заклик до дії, наприклад, термінове оновлення безпеки або підтвердження банківської операції. Жертва, не підозрюючи небезпеки, відкриває повідомлення та переходить за посиланням, яке веде на шкідливий вебсайт або автоматично ініціює завантаження шкідливого файлу. Після встановлення цього файлу мобільний телефон жертви компрометується, надаючи атакувальнику віддалений доступ до системи. Шкідливе програмне забезпечення може використовуватися для збору облікових даних, перехоплення повідомлень, контролю за мобільним пристроєм або навіть поширення атак далі по контактному списку жертви. У підсумку викрадені дані передаються атакувальнику, який може використовувати їх для доступу до фінансових сервісів, соціальних мереж або корпоративних систем жертви.

Смішинг є особливо небезпечним видом соціальної інженерії, оскільки мобільні пристрої менш захищені, ніж стаціонарні комп’ютери, а їхні користувачі часто нехтують базовими заходами безпеки. Висока швидкість обробки SMS-повідомлень сприяє імпульсивному прийняттю рішень, що підвищує успішність атак. Для ефективної боротьби з такими загрозами необхідно впроваджувати технології аналізу підозрілих посилань, блокування SMS зі шкідливими вкладеннями та підвищувати рівень обізнаності користувачів щодо можливих ризиків.

Сучасні методи соціальної інженерії є гнучкими, адаптивними та технологічно розвинутими. Вони дедалі частіше використовують елементи штучного інтелекту, автоматизацію генерації фішингових повідомлень, генерацію шкідливих доменів та персоналізовані психологічні сценарії. Це зумовлює потребу в розробці нових підходів до аналізу та автоматичного виявлення загроз, здатних адаптуватися до нових форм соціально-інженерного впливу.

1.2 Огляд методів протидії атакам соціальної інженерії

Методи соціальної інженерії становлять серйозну загрозу для інформаційної безпеки, оскільки вони орієнтовані на маніпулювання поведінкою людини, а не на технічні вразливості систем. Основна проблема полягає в тому, що навіть

найсучасніші засоби захисту можуть виявитися безсилі перед маніпуляцією користувачем, який добровільно передає свої облікові дані або виконує небезпечні дії. Враховуючи цю особливість, ефективна протидія соціально-інженерним атакам має бути комплексною, поєднуючи технологічні рішення, організаційні заходи та навчальні ініціативи. Наприклад, навіть якщо корпоративна мережа використовує сучасні методи багатофакторної автентифікації та шифрування даних, зловмисник може переконати співробітника надати доступ, видаючи себе за адміністратора безпеки. Особливо небезпечними є цільові атаки на керівників компаній, оскільки компрометація облікового запису топменеджера може відкрити доступ до фінансових операцій або критично важливих систем. Нижче розглянуто сучасні методи запобігання фішинговим атакам, вішингу, смішингу та іншим видам соціальної інженерії, які спрямовані на захист як індивідуальних користувачів, так і корпоративних структур.

Технологічні методи захисту

Технологічні методи захисту від атак соціальної інженерії охоплюють широкий спектр інструментів, спрямованих на виявлення, блокування та нейтралізацію загроз, пов’язаних із маніпулятивними методами впливу на користувачів. Вони базуються на автоматизованому аналізі електронних комунікацій, перевірці автентичності ресурсів, поведінковому моніторингу та контролі доступу. Використання таких рішень дозволяє значно зменшити ризики фішингових атак, вішингу, смішингу та інших форм соціальної інженерії, унеможливлюючи доступ зловмисників до критичних систем і даних [12]. Однією з ключових переваг технологічних рішень є можливість аналізу великих обсягів даних у реальному часі, що забезпечує оперативне реагування на загрози. Такі методи використовуються як окремими користувачами, так і великими організаціями, які прагнуть захистити свою інформаційну інфраструктуру.

Для наочного представлення основних технологічних методів захисту від атак соціальної інженерії у табл. 1.1 наведено їхній опис, принцип роботи, особливості реалізації та типові сфери застосування.

Таблиця 1.1

Технологічні методи захисту від атак соціальної інженерії

Метод захисту	**Принцип роботи**	**Особливості реалізації**	**Сфера застосування**
Антифішингові фільтри	Аналіз заголовків, вмісту листів та доменів відправника	Використання алгоритмів машинного навчання та чорних списків [13]	Захист електронної пошти та веб-браузерів
DNS-фільтрація	Перевірка репутації вебсайтів перед відкриттям	Порівняння доменів із базами шкідливих ресурсів	Корпоративні та домашні мережі
Захист від спуфінгу та підміни номерів	Аналіз вихідних номерів телефонних дзвінків та SMS	Виявлення підміни номерів у мережах операторів [14]	Боротьба з вішингом та смішингом
Аналітика поведінки користувачів	Виявлення аномальної активності в облікових записах	Аналіз поведінкових патернів у системах аутентифікації	Захист корпоративних мереж
Захист браузерів від підроблених сайтів	Вбудовані перевірки SSL-сертифікатів та доменів	Виявлення невідповідностей у структурі сайтів [15]	Запобігання доступу до фішингових сайтів
Системи управління доступом (IAM)	Контроль прав користувачів у інформаційних системах	Автоматичне обмеження доступу до критичних ресурсів	Корпоративні інформаційні системи
Автоматизовані системи реагування (SOAR)	Інтеграція аналітики та засобів безпеки для швидкого реагування	Автоматизація виявлення загроз та запуск заходів реагування [16]	Центри обробки інцидентів (SOC)

Технологічні методи захисту забезпечують багаторівневу безпеку, об’єднуючи превентивні заходи та механізми реагування на загрози. Вони дозволяють значно ускладнити реалізацію атак соціальної інженерії, запобігаючи витоку конфіденційної інформації та компрометації систем. Впровадження таких рішень є критично важливим як для індивідуальних користувачів, так і для організацій, що працюють з

чутливими даними. Разом із навчальними та організаційними заходами технологічні методи формують комплексну систему протидії атакам соціальної інженерії.

Організаційні заходи безпеки

Організаційні заходи безпеки спрямовані на створення внутрішніх політик, процедур та регламентів, які мінімізують ризики соціально-інженерних атак та підвищують рівень захищеності інформаційних систем. Вони використовуються для впровадження чітких правил роботи з конфіденційними даними, контролю доступу до критичних ресурсів та регулювання дій персоналу у разі виявлення загроз [17]. Ці заходи є особливо важливими у корпоративному середовищі, де атаки соціальної інженерії можуть призвести до витоку інформації, фінансових втрат або компрометації систем. Основна мета таких заходів – обмеження можливостей зловмисників маніпулювати співробітниками організації, зменшення ймовірності успішного фішингу, вішингу та інших форм шахрайства. До найпоширеніших організаційних заходів належать розмежування прав доступу, розробка внутрішніх протоколів реагування, використання політик безпечної автентифікації та проведення регулярних аудитів безпеки.

Для структурованого аналізу основних організаційних заходів безпеки у таблиці 1.2 наведено їхні особливості, механізм реалізації та основні сфери застосування.

Таблиця 1.2

Заходи безпеки у протидії атакам соціальної інженерії

Заходи безпеки	**Принцип реалізації**	**Ключові особливості**	**Сфера застосування**
1	2	3	4
Політика мінімальних привілеїв (PoLP)	Надання користувачам доступу лише до необхідних ресурсів	Обмеження ризику компрометації критичних систем	Корпоративні мережі та інформаційні системи
Впровадження багатофакторної автентифікації (MFA)	Використання двох і більше факторів для входу в систему	Зниження ймовірності несанкціонованого доступу [18]	Фінансові сервіси, корпоративні акаунти

продовження табл.1.2

1	2	3	4
Політика зміни та управління паролями	Регулярне оновлення паролів та вимоги до їх складності	Запобігання компрометації через викрадені паролі	Всі користувацькі системи
Контроль доступу до інформаційних ресурсів	Розмежування рівнів доступу залежно від ролі користувача	Запобігання витоку даних та маніпуляції співробітниками	Організації, державні установи
Впровадження Zero Trust Security	Автентифікація кожного запиту перед наданням доступу	Блокування непередбачуваних загроз [19]	Хмарні платформи, критично важливі системи
Внутрішні протоколи реагування на загрози	Регламентація дій у разі фішингових атак та витоків даних	Швидке реагування на інциденти	Корпоративні безпекові служби
Регулярні аудити інформаційної безпеки	Аналіз відповідності політик безпеки сучасним загрозам	Виявлення потенційних слабких місць [20]	Фінансовий сектор, державні структури
Використання Data Loss Prevention (DLP)	Запобігання несанкціонованому поширенню даних	Контроль передачі конфіденційної інформації	Корпоративні інформаційні системи

Організаційні заходи безпеки відіграють важливу роль у формуванні комплексного захисту від соціально-інженерних атак. Вони регулюють поведінку користувачів у межах інформаційної системи, забезпечуючи контроль над доступом, автентифікацією та передачею даних. Такі заходи дозволяють зменшити ймовірність маніпуляцій з боку зловмисників, оскільки користувачі діють у рамках чітко визначених протоколів безпеки. Впровадження організаційних заходів у поєднанні з технологічними методами забезпечує комплексний захист інформаційних ресурсів від атак соціальної інженерії.

Навчання та підвищення обізнаності користувачів

Навчання та підвищення обізнаності користувачів є одним із ключових заходів протидії атакам соціальної інженерії, оскільки більшість таких атак спрямовані на маніпуляцію поведінкою людини. Основна мета цих заходів – формування у

користувачів навичок критичного мислення, вміння ідентифікувати потенційно небезпечні дії та правильно реагувати на підозрілі запити [21]. Використання програм навчання з кібербезпеки дозволяє працівникам організацій та звичайним користувачам усвідомлювати можливі загрози та мінімізувати ризики компрометації інформації. До основних методів навчання належать регулярні тренінги, симуляції фішингових атак, проведення кібербезпекових тестувань та інформування про новітні шахрайські схеми. Особливу роль відіграють інтерактивні платформи, які дозволяють навчатися в умовах, наближених до реальних атак.

Для аналізу основних методів навчання та підвищення обізнаності у табл. 1.3 наведено їхні характеристики, механізм реалізації та сфери застосування.

Таблиця 1.3

Методи навчання та підвищення обізнаності користувачів

Метод навчання	**Сутність**	**Особливості реалізації**	**Сфера застосування**
Інтерактивні тренінги з кібербезпеки	Проведення лекцій та практичних занять з кіберзахисту	Використання симуляційних сценаріїв атак [22]	Корпоративні середовища, навчальні заклади
Симуляції фішингових атак	Проведення контрольованих фішингових атак для оцінки обізнаності	Аналіз помилок користувачів, персональні рекомендації	Комерційні та державні установи
Розсилки інформаційних бюлетенів	Надання актуальної інформації про нові загрози	Оновлення знань працівників через корпоративну пошту	Великі організації, банки, держструктури
Моделювання соціальних атак	Створення сценаріїв атаки для навчання персоналу	Аналіз поведінки співробітників у ситуаціях реальних загроз [23]	Організації, що працюють з конфіденційними даними
Тестування знань персоналу	Проведення періодичних перевірок знань з кібербезпеки	Автоматизовані онлайн-тестування	Будь-які організації, що працюють з даними

Методи протидії атакам соціальної інженерії повинні бути комплексними та охоплювати технологічні рішення, організаційні заходи та навчання користувачів. Технологічні засоби, такі як антифішингові фільтри, багатофакторна автентифікація та системи аналізу аномальної активності, допомагають виявляти та блокувати загрози ще до того, як вони досягають користувачів. Організаційні заходи, включаючи політики контролю доступу, регламентацію роботи з обліковими даними та сценарії реагування на інциденти, підвищують загальний рівень безпеки. Навчання та обізнаність персоналу дозволяють мінімізувати ризики, пов’язані з людським фактором, що робить атаки менш ефективними. Поєднання всіх цих методів є необхідним для побудови надійної системи захисту від загроз соціальної інженерії.

1.3 Порівняльний аналіз методів виявлення та запобігання атакам соціальної інженерії

З метою визначення сильних і слабких сторін різних підходів до протидії соціальній інженерії, доцільним є порівняльний аналіз методів, розглянутих у попередньому підрозділі 1.2. Враховуючи технологічні, організаційні та освітні засоби захисту, доцільно зіставити їх за ключовими характеристиками. Нижче у табл. 1.4 наведено систематизоване порівняння основних методів виявлення та запобігання атакам соціальної інженерії.

Таблиця 1.4

Порівняльний аналіз методів

Метод	Принцип роботи	Рівень автоматизації	Адаптивність до нових загроз	Сфера застосування
1	2	3	4	5
Антифішингові фільтри	Аналіз заголовків, вмісту листів, репутації відправників	Високий	Обмежена, потребує оновлення баз	Електронна пошта, веб-браузери

продовження табл.1.4

1	2	3	4	5
Багатофакторна автентифікація	Додаткове підтвердження особи (код, біометрія)	Середній	Висока	Фінансові сервіси, корпоративні акаунти
DNS-фільтрація	Блокування доступу до фішингових доменів	Високий	Середня	Корпоративні мережі, хмарні сервіси
Навчальні тренінги з кібербезпеки	Освіта користувачів щодо виявлення загроз	Низький	Висока	Організації, держустанови
Симуляції фішингових атак	Тестові атаки для перевірки обізнаності персоналу	Низький	Висока	Корпоративні середовища
Політика мінімальних привілеїв	Обмеження доступу до критичних систем	Середній	Висока	Організації, державні установи
Аналіз поведінки користувачів	Виявлення аномалій у діях користувачів	Високий	Висока	Корпоративні мережі, фінансовий сектор
Автоматизовані системи реагування	Інтеграція аналітики та реагування на інциденти	Високий	Висока	Центри безпеки (SOC)

Згідно з проведеним аналізом, технологічні методи, такі як антифішингові фільтри, DNS-фільтрація та аналіз поведінки користувачів, мають високий рівень автоматизації та дозволяють швидко виявляти загрози у великих інформаційних потоках. Однак вони мають певні обмеження у адаптивності, оскільки нові методи атак можуть обходити існуючі системи детектування. Наприклад, фішингові кампанії можуть використовувати тимчасові домени або зламані акаунти з реальними контактами, що ускладнює автоматизоване блокування загроз.

Організаційні заходи, зокрема політика мінімальних привілеїв та системи багатофакторної автентифікації, є ефективними в довгостроковій перспективі,

оскільки обмежують можливість компрометації систем. Вони менш схильні до застарівання, однак вимагають правильної реалізації та адміністрування. Наприклад, навіть якщо MFA використовується у компанії, зловмисники можуть обійти цей захист, якщо співробітники передають коди підтвердження через соціальні маніпуляції.

Освітні методи, такі як тренінги з кібербезпеки та симуляції атак, є найбільш гнучкими до нових загроз, оскільки навчають користувачів розпізнавати новітні схеми атак. Проте їхній рівень автоматизації низький, і вони не гарантують абсолютного захисту без поєднання з технологічними заходами. Наприклад, регулярне проведення тестових атак на співробітників дозволяє виявити вразливі місця в організації та своєчасно коригувати навчальні програми.

Жоден метод виявлення та запобігання атакам соціальної інженерії не є універсальним, тому ефективний захист можливий лише за умови поєднання технологічних, організаційних та освітніх підходів. Автоматизовані рішення дозволяють швидко виявляти підозрілі активності та блокувати загрози, проте вони можуть виявитися неефективними у разі використання нових, складних методів атак. Організаційні заходи створюють стабільну систему контролю доступу та автентифікації, але потребують правильної імплементації та дотримання безпекових політик. Навчальні ініціативи, незважаючи на низький рівень автоматизації, мають високу адаптивність до нових методів атак, оскільки користувачі отримують навички розпізнавання загроз у реальному середовищі. Особливу перспективу в напрямі автоматизованого захисту становлять методи машинного навчання, які здатні виявляти приховані закономірності в поведінці зловмисників, адаптуватися до нових атак і забезпечувати динамічне вдосконалення систем виявлення. Поєднання всіх цих методів забезпечує багаторівневий підхід до захисту від соціально-інженерних атак, що є критично важливим для сучасних інформаційних систем.

1.4 Постановка задачі дослідження

Виявлення фішингових атак є актуальною задачею в сфері кібербезпеки, що потребує інтелектуальних рішень, здатних аналізувати характерні ознаки доменів і виявляти потенційно шкідливі ресурси. Сучасні фішингові кампанії швидко змінюють свої техніки, і тому система виявлення повинна бути адаптивною, здатною аналізувати як структурні, так і статистичні характеристики доменних імен. Для цього необхідно визначити вимоги до системи, сформулювати загальну задачу класифікації та забезпечити можливість її перевірки на практичних даних.

У ході попереднього аналізу було сформульовано низку функціональних та аналітичних вимог до майбутньої системи. Зокрема, вона повинна бути здатна:

– обробляти вхідні дані з набору доменних імен, що містять як легітимні, так і потенційно фішингові приклади;

– здійснювати попередню обробку вхідних ознак, включаючи текстові характеристики, числові метрики (наприклад, часткові збіги, індекси подібності), позиційні та семантичні індикатори;

– на основі наявного набору навчальних прикладів формувати класифікаційну модель, здатну ідентифікувати нові випадки фішингової активності;

– виконувати оцінювання точності моделі за допомогою відповідних метрик (точність, повнота, F1-міра, AUC) і порівнювати результати з еталонними значеннями;

– забезпечувати інтерпретоване подання результатів користувачу з виведенням детальної інформації про ймовірність фішингу та рекомендаціями щодо дій.

У структурі системи необхідно передбачити функціональні модулі для:

– завантаження даних та запуску навчання моделі;

– тестування моделей на нових прикладах;

– формування текстових звітів з результатами класифікації;

– зберігання та вибору моделей для подальшого використання.

Важливо, що вхідні ознаки повинні мати відповідне змістовне навантаження, наприклад: рейтинг домену (як показник авторитетності), кількість залишкових компонентів, результати структурної декомпозиції, значення коефіцієнтів подібності Жаккара між різними частинами доменного імені. Такий підхід дозволяє побудувати гнучку та інформативну модель, яка може виявляти як прості, так і приховані фішингові схеми.

Таким чином, дослідження має на меті вирішити задачу побудови аналітичної системи, яка здатна автоматизовано здійснювати виявлення доменів з ознаками фішингових атак на основі багатовимірного представлення їхніх ознак. Постановка цієї задачі охоплює розробку узагальненої схеми класифікації, підбір інформативних характеристик, формування моделі навчання та валідації результатів, а також побудову зручного механізму взаємодії з кінцевим користувачем.

Висновки до розділу 1

У рамках даного розділу було проведено всебічне дослідження сучасного стану проблеми атак соціальної інженерії з акцентом на їх різновиди, методи реалізації та засоби протидії. Проаналізовано ключові сценарії атак, зокрема фішинг, цільовий фішинг, вішинг і смішинг, із візуалізацією кожного з механізмів у вигляді відповідних схем, що дозволило виявити загальні риси, відмінності та ключові вектори впливу на користувача.

Окрему увагу приділено огляду сучасних методів протидії зазначеним загрозам. Було досліджено технологічні засоби виявлення атак, організаційні заходи (керування доступом, політики MFA, аудит безпеки) та методи підвищення обізнаності користувачів. Здійснено порівняльний аналіз цих підходів за критеріями автоматизації, адаптивності до нових загроз і сфер застосування, що дозволило обґрунтувати доцільність їх комбінованого використання для досягнення максимальної ефективності захисту.

На основі проведеного аналізу сформульовано задачу дослідження, яка полягає у побудові програмної системи для виявлення фішингових атак на основі ознак

доменних імен, із подальшою класифікацією шкідливих ресурсів та представленням результатів користувачу в інтерпретованому вигляді. Отримані результати створюють ґрунт для переходу до наступного етапу дослідження – аналізу методів виявлення фішингових атак, вибору найбільш релевантного підходу та обґрунтування ознак, які будуть використані в рамках проєктування системи.

РОЗДІЛ 2.
АНАЛІЗ МЕТОДІВ ВИЯВЛЕННЯ АТАК СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ

2.1 Обґрунтування вибору набору даних для аналізу фішингових атак

Одним із ключових аспектів ефективної побудови системи виявлення фішингових атак, що є складовою ширшої проблематики аналізу загроз соціальної інженерії, є використання якісного та репрезентативного набору даних для навчання і тестування моделі. Адекватно підібрана вибірка дозволяє не лише підвищити точність класифікації, а й забезпечити здатність моделі до узагальнення результатів на нові, невідомі приклади.

З огляду на зазначене, в межах цього дослідження було обрано датасет Phishing Website Detection Dataset, розміщений на платформі Kaggle [24]. Вибір саме цього набору обумовлений кількома причинами. По-перше, датасет охоплює широкий спектр атрибутів, що дозволяють здійснювати комплексну оцінку URL-адрес з точки зору ознак, притаманних фішинговим сайтам. По-друге, структура даних добре підходить для реалізації бінарної класифікації – визначення належності ресурсу до категорії легітимного або фішингового.

Кожен запис у датасеті відповідає конкретному веб-домену та описується низкою числових і логічних характеристик, що формуються на основі морфологічного, семантичного й контекстного аналізу URL. Окремі ознаки, такі як співвідношення видалених символів, коефіцієнти схожості Джаккара, кількість змін у структурі домену, дозволяють моделі виявляти неочевидні шаблони, притаманні фішинговим ресурсам. Наявність чітко позначеної цільової змінної label, яка приймає значення 1 для фішингових і 0 для легітимних сайтів, робить набір безпосередньо придатним для формування моделі класифікації. Крім того, збалансованість даних і включення великої кількості прикладів з різною структурою забезпечує стійкість і надійність результатів, що досягаються при навчанні.

У табл. 2.1 подано перелік основних атрибутів вибраного набору даних, їх короткий опис, тип і приклад значення.

Таблиця 2.1

Атрибути набору даних для виявлення фішингових атак

Атрибут	**Опис**	**Тип**	**Приклад значення**
domain	Повна URL-адреса вебресурсу, який аналізується	Катего-ріальний	www.sec.gov / edgar.shtml
ranking	Глобальний рейтинг популярності домену	Числовий	6396
mld_res	Ознака наявності основного домену (main-level domain, MLD) у базі відомих доменів	Бінарний	1.0 (так), 0.0 (ні)
mld.ps_res	Ознака наявності піддомену другого рівня (public suffix) у базі реєстраційних доменів	Бінарний	1.0 (так), 0.0 (ні)
card_rem	Кількість операцій редукції, застосованих до домену під час обробки	Ціле	2
ratio_Rrem	Співвідношення кількості символів, видалених з основного імені ресурсу (R)	Дійсний	16.0
ratio_Arem	Співвідношення символів, видалених з альтернативного імені ресурсу (A)	Дійсний	15.0
jaccard_RR	Коефіцієнт Джаккара між ресурсом R до та після перетворення (redundancy-based)	Дійсний	0.0375
jaccard_RA	Коефіцієнт Джаккара між основним і альтернативним доменом	Дійсний	0.025316
jaccard_AR	Коефіцієнт Джаккара між альтернативним і основним доменом (дзеркальне порівняння)	Дійсний	0.016667
jaccard_AA	Коефіцієнт Джаккара для альтернативного домену до та після редукції	Дійсний	0
jaccard_ARrd	Коефіцієнт Джаккара між альтернативною і редукованою версією основного домену	Дійсний	0.509434
jaccard_ARrem	Показник схожості	Дійсний	0.55
label	Цільова змінна: клас ресурсу (1 – фішинговий сайт, 0 легітимний сайт)	Бінарний	0

У рамках дослідження, що спрямоване на побудову системи аналізу атак соціальної інженерії, зокрема фішингових доменів, важливо не лише відібрати релевантні атрибути для моделі, а й дослідити взаємозв’язки між ними. Такий аналіз дозволяє виявити залежності між окремими характеристиками, знизити надлишковість даних і забезпечити ефективність навчання.

Аналіз теплової карти кореляційної матриці (рис. 2.1) дозволяє глибше зрозуміти характер взаємозв’язків між числовими ознаками, що використовуються у побудові системи виявлення фішингових атак.

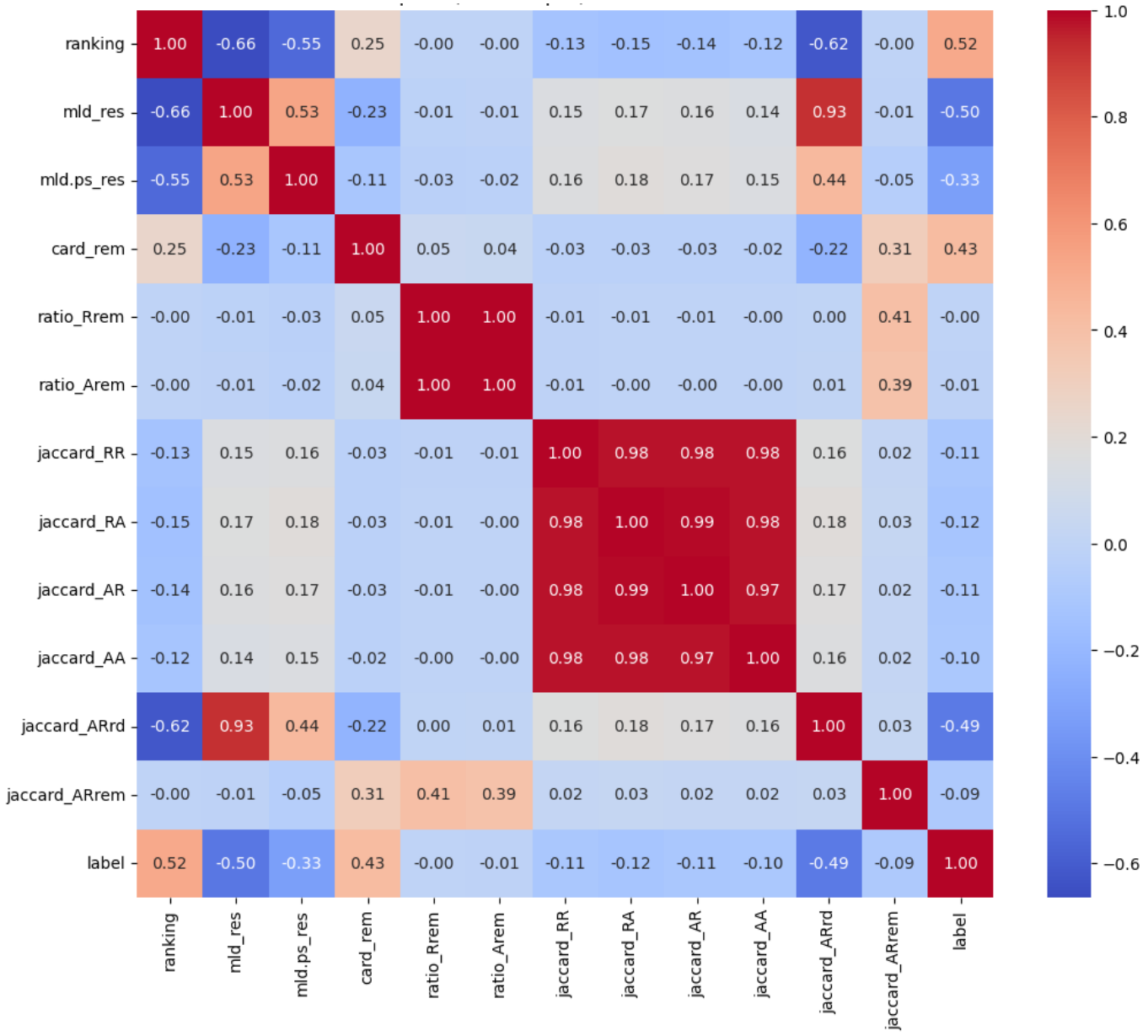

Рисунок 2.1 – Теплова карта кореляції між ознаками

Виявлено, що між багатьма метриками на основі коефіцієнта Джаккара (jaccard_RR, jaccard_RA, jaccard_AR, jaccard_AA) спостерігається майже ідеальна лінійна залежність із коефіцієнтами кореляції, що перевищують 0.97. Це свідчить про високий рівень надлишковості між цими змінними та потенційну доцільність їх об’єднання або виключення окремих з них при побудові моделі, з метою зниження колінеарності.

Ознака ranking має помітний негативний зв’язок із цільовою змінною label, що означає: чим нижче рейтинг ресурсу, тим вищою є ймовірність, що домен є фішинговим. Аналогічну тенденцію виявлено для ознаки mld_res, яка демонструє зворотну залежність із класом – ресурси, основні домени яких зареєстровані у відомих легітимних базах, рідше є фішинговими. У той же час такі характеристики, як card_rem і mld.ps_res, демонструють слабку позитивну кореляцію з класом label, що може свідчити про їхню допоміжну інформативність у контексті комплексного аналізу.

Більшість інших ознак, зокрема ratio_Rrem, ratio_Arem і частина jaccard_* метрик, мають близьке до нуля кореляційне значення з цільовою змінною, що вказує на відсутність сильного лінійного впливу на класову приналежність. Проте, з огляду на складну природу фішингових шаблонів, це не виключає їхнього потенційного впливу у нелінійних моделях, що здатні враховувати складні багатовимірні взаємозалежності. Отже, результати кореляційного аналізу свідчать про наявність як інформативних, так і надлишкових ознак, що є підставою для застосування методів відбору характеристик перед побудовою моделі класифікації.

На рис. 2.2 наведено розподіли чотирьох найбільш інформативних ознак – ranking, mld_res, ratio_Rrem та jaccard_ARrem, – що демонструють характер змінності значень у межах кожного з класів. Помітно, що ознака ranking для більшості фішингових сайтів має значення в діапазоні близькому до максимально можливого (тобто рейтинги або відсутні, або вкрай низькі за репутаційними оцінками), тоді як легітимні ресурси здебільшого концентруються в нижньому діапазоні рейтингу. Це узгоджується з аналітикою попереднього кореляційного аналізу і підтверджує високу релевантність цієї ознаки для класифікації.

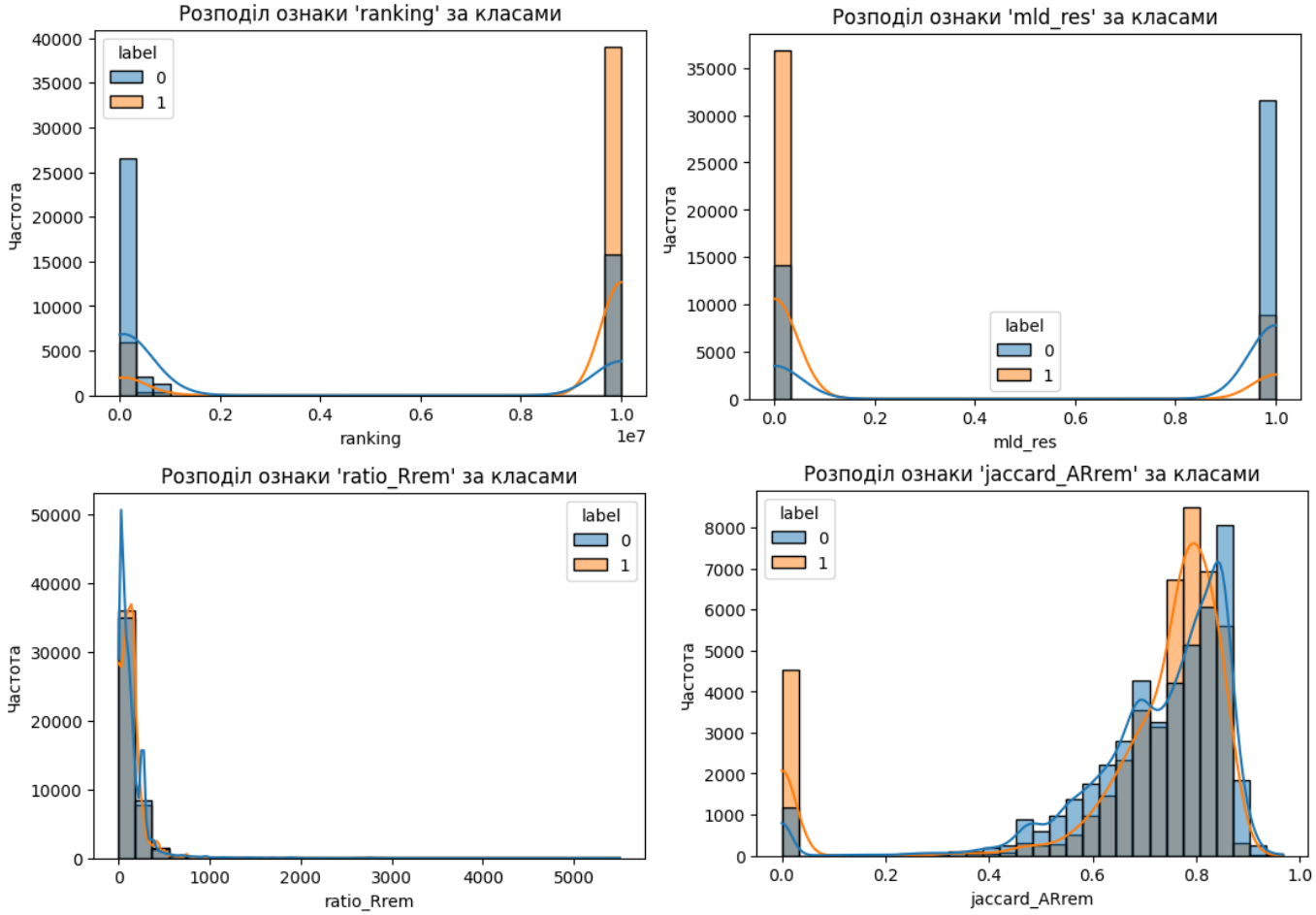

Рисунок 2.2 – Розподіл найбільш інформативних ознак

Ознака mld_res також демонструє значне розділення: легітимні сайти найчастіше мають позначку наявності основного домену в базі (значення 1), тоді як у фішингових ресурсів ця ознака часто відсутня (значення 0), що логічно, враховуючи їхню тенденцію до використання слабо відомих або підроблених доменів.

У випадку з ratio_Rrem розподіл менш чітко розмежований, однак можна спостерігати, що фішингові домени частіше мають вищі значення цієї ознаки, що відображає більшу кількість маніпуляцій із доменними назвами під час їх генерації. Водночас, для легітимних сайтів спостерігається помітне згущення значень у нижньому діапазоні. Щодо jaccard_ARrem, розподіл у обох класах є більш розтягнутим, однак фішингові приклади демонструють тенденцію до зсуву в бік вищих значень коефіцієнта Джаккара. Це вказує на структурну подібність між альтернативними формами доменів та їх редукованими варіантами, що може бути

наслідком генерації схожих доменних структур із ціллю введення користувача в оману.

На наступному етапі аналізу було здійснено просторову візуалізацію об’єктів навчальної вибірки у тривимірному просторі ключових ознак, що дозволяє виявити приховані структури та взаємозв’язки, які не завжди очевидні у площинному представленні. Для цього було обрано три найбільш інформативні характеристики: ranking, ratio_Rrem та jaccard_ARrem, кожна з яких вже продемонструвала релевантність до цільової змінної на попередніх етапах дослідження. Візуалізація наведена на рис. 2.3.

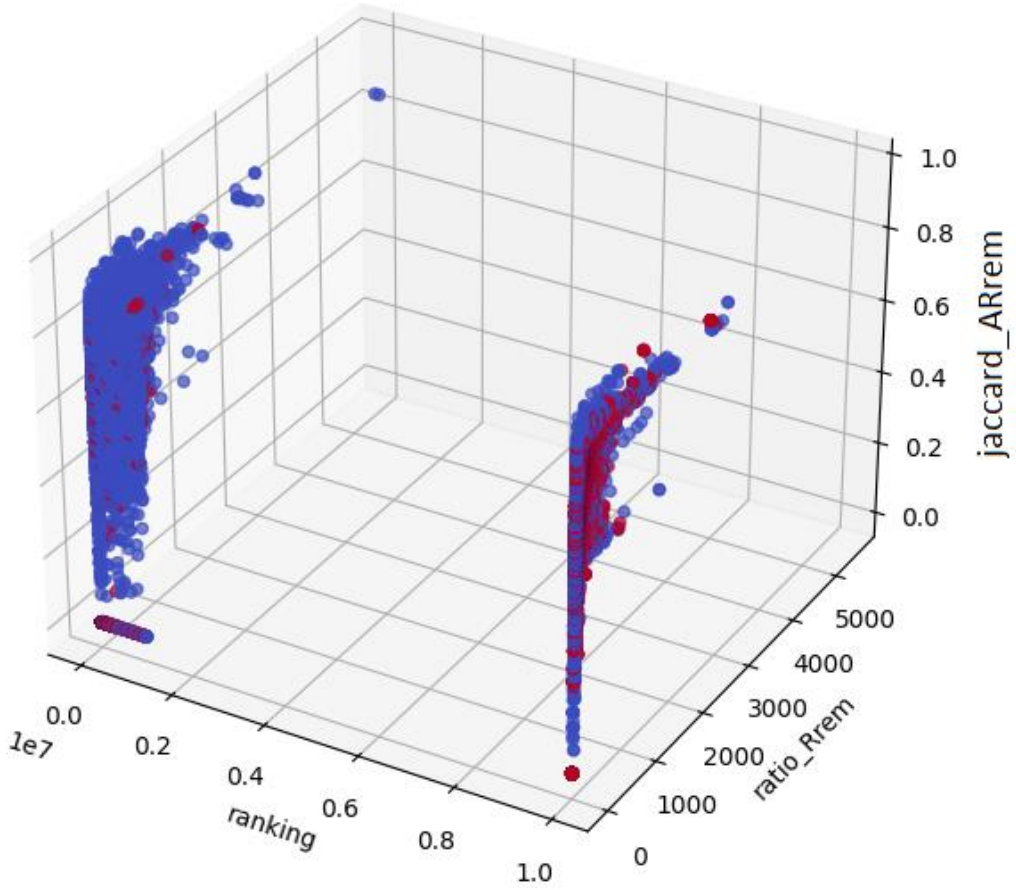

Рисунок 2.3 – 3D-розподіл за ознаками

У результаті побудови 3D-графіка спостерігається певне просторове згущення об’єктів, які належать до різних класів, у межах окремих діапазонів значень. Значна частина легітимних доменів концентрується в області з низькими значеннями ranking (тобто високою репутацією), малим ratio_Rrem (мінімальні модифікації домену) та

високим jaccard_ARrem (низька схожість між редукованими і альтернативними варіантами домену). Натомість фішингові приклади здебільшого мають або дуже високі значення ranking (або відсутній рейтинг), або демонструють підвищену активність у зміні структури імені домену, що відображено у вищих значеннях ratio_Rrem.

Особливу увагу привертає той факт, що для фішингових доменів показник jaccard_ARrem зазвичай також зміщений у бік більших значень, що вказує на вищу схожість між різними варіаціями домену – це може бути прямим наслідком генерації фішингових адрес на основі вже існуючих легітимних імен. Згрупованість об'єктів у тривимірному просторі дає підстави стверджувати, що обрана комбінація ознак має достатній потенціал для побудови гіперплощини розділення між класами за допомогою класифікаційної моделі, зокрема тих, що здатні враховувати нелінійні залежності (наприклад, Random Forest або Support Vector Machines).

З метою поглиблення аналітичного розуміння структурної природи фішингових та легітимних доменів у просторі обраних ознак, доцільним є візуальне представлення парної залежності між ключовими предикторами з одночасним урахуванням класової приналежності об’єктів. Такий підхід дозволяє не лише оцінити тенденції розподілу кожної ознаки окремо, а й виявити потенційні взаємозв’язки між парами змінних у контексті класифікаційної задачі.

На рис. 2.4 представлено матрицю парних графіків для трьох найбільш інформативних ознак – ranking, ratio_Rrem та jaccard_ARrem, – розділену за класами цільової змінної label. На головній діагоналі відображено оцінку щільності розподілу кожної змінної окремо, тоді як на перетинах – діаграми розсіювання між відповідними парами.

Аналіз отриманих результатів дозволяє зафіксувати чітке розділення класів на рівні окремих ознак. Для ознаки ranking клас 1 (фішингові домени) сконцентрований у межах високих значень, що ще раз підтверджує відсутність рейтингу або низьку репутацію фішингових сайтів у глобальних індексах. У свою чергу, клас 0 (легітимні сайти) демонструє концентрацію в нижньому діапазоні ranking, що є ознакою популярності та достовірності таких ресурсів.

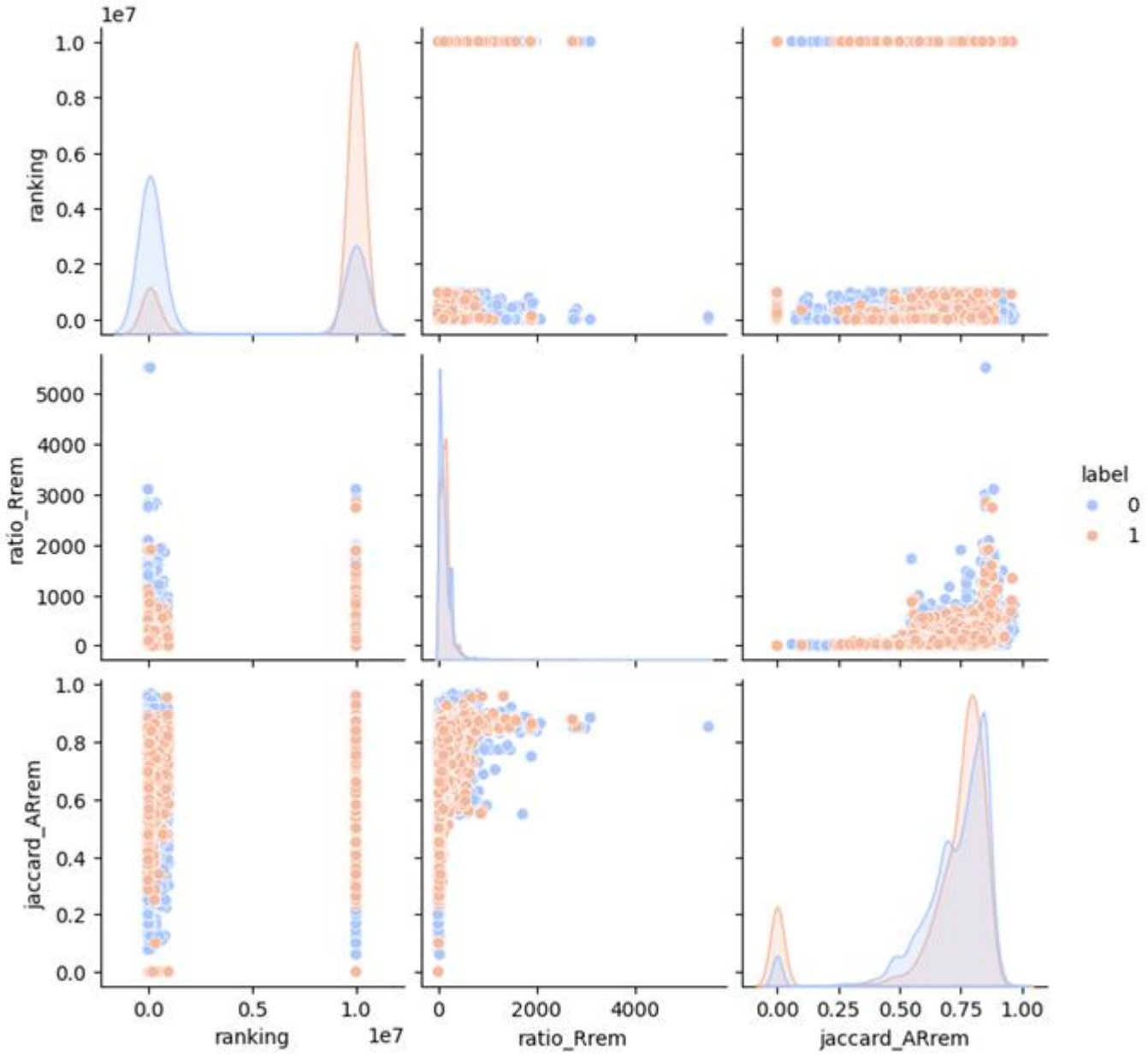

Рисунок 2.4 – Парна кореляція ключових змінних з розділенням за класами

Розподіл ratio_Rrem дещо менш контрастний, однак фішингові сайти загалом демонструють підвищені значення цієї ознаки, що свідчить про вищу інтенсивність модифікацій у доменних структурах. Ознака jaccard_ARrem, у свою чергу, демонструє сильнішу сегментацію: для класу 1 значення зазвичай вищі, тоді як для легітимних ресурсів – ближчі до середнього або нижчого діапазону, що вказує на зменшену схожість між різними версіями домену в останньому випадку.

Візуалізація пар ranking – ratio_Rrem, ranking – jaccard_ARrem та ratio_Rrem – jaccard_ARrem також підтверджує наявність певних групових зосереджень, зокрема

для фішингових ресурсів, що групуються в зонах із високим ranking та водночас з вираженими змінами у структурі URL. Наявність таких структурних патернів вказує на потенційну здатність класифікаційної моделі виявляти фішингові домени навіть у випадку відсутності повної лінійної кореляції, завдяки нелінійним зв’язкам між атрибутами.

Для оцінки внутрішньої структури даних та потенційної наявності кластерів без використання маркерів цільової змінної було здійснено кластеризацію об’єктів за допомогою методу K-середніх (KMeans) з наступним зменшенням розмірності простору до двох головних компонент за допомогою методу головних компонент (PCA). Візуалізація результатів кластеризації подана на рис. 2.5.

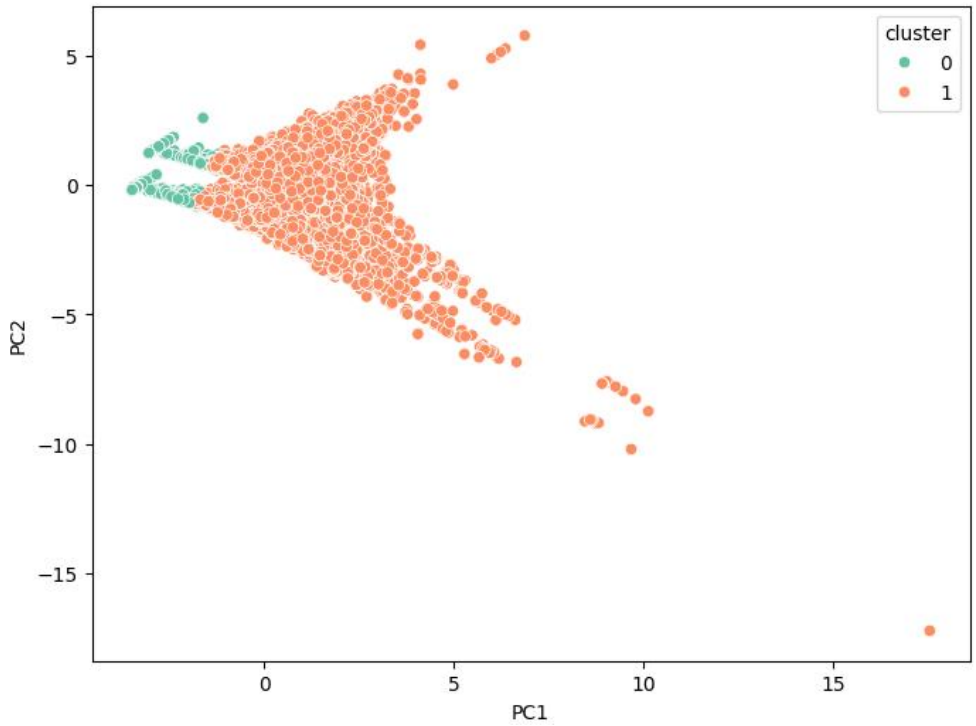

Рисунок 2.5 – Кластеризація KMeans у 2D-просторі

На рисунку зображено дві групи об’єктів, сформовані без урахування цільової змінної, тобто винятково на основі просторових взаємовідносин між ознаками. Візуально спостерігається домінування одного з кластерів (кластер 1), у якому зосереджена більшість об'єктів, а також наявність меншої групи (кластер 0), яка

утворює щільніше згущення ближче до початку координат у просторі головних компонент.

Такий розподіл дозволяє припустити, що дані мають часткову внутрішню структуру, яка, ймовірно, узгоджується із реальним поділом на фішингові та легітимні домени. Зокрема, можна інтерпретувати кластер з меншою кількістю об’єктів як такий, що потенційно відповідає легітимним сайтам, які формують відносно однорідну групу із менш варіативними ознаками, тоді як більший кластер охоплює широкий спектр фішингових варіантів з різноманітними модифікаціями у структурі доменів.

Слід зауважити, що хоча метод KMeans не враховує інформацію про мітки класів, він демонструє здатність виділяти певну внутрішню кластерну структуру у просторі ознак, що підтверджує релевантність вибраних предикторів і підкреслює наявність патернів, притаманних доменам різних типів. У сукупності з результатами попередніх візуалізацій це свідчить про високу потенційну роздільну здатність системи при використанні методів навчання з учителем, які здатні ефективно скористатися виявленими закономірностями.

Для кількісної оцінки відповідності отриманих кластерів реальному розподілу класів було побудовано матрицю відповідності, яка відображає перетин між істинними мітками класів і призначеними алгоритмом кластеризації групами. Візуалізація цієї матриці подана на рис. 2.6.

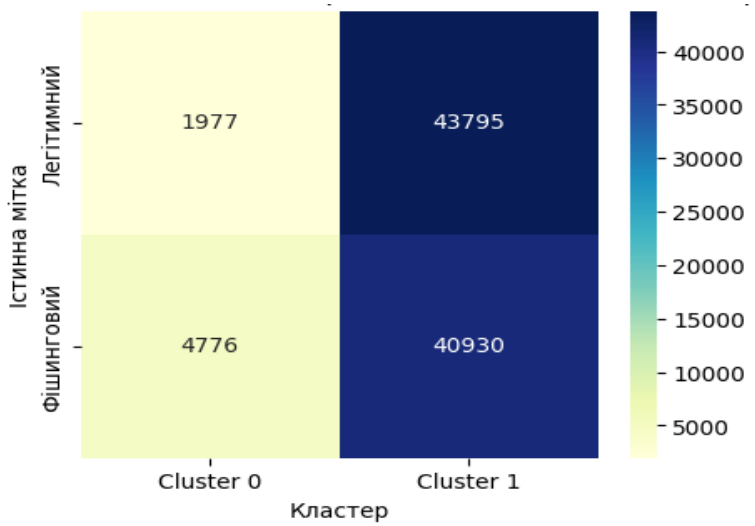

Рисунок 2.6 – Матриця відповідності між справжніми мітками та кластерами

Як видно з рисунку 2.6, кластер 1 є переважним для обох категорій, проте найбільшу кількість об’єктів він об’єднує саме з легітимного класу – 43 795 прикладів, у той час як 4 776 фішингових сайтів також потрапили до цього кластеру. Водночас кластер 0 сформований значно менш чисельно і містить 1 977 легітимних доменів та 40 930 фішингових. Такий розподіл свідчить про певну асиметричність у структурі кластерів, що є типовим у випадках незбалансованих даних або при наявності сильно зміщених ознак.

Незважаючи на те, що метод кластеризації не мав доступу до цільової змінної label, виявлена відповідність між окремими кластерами та реальними класами є доволі показовою. Зокрема, кластер 0 більш точно відповідає фішинговим доменам, тоді як кластер 1 охоплює більшість легітимних сайтів, проте з частковим перетином. Цей факт підтверджує, що обраний вектор ознак дійсно несе в собі корисну інформацію, яка дозволяє виділяти структури, притаманні фішинговим атакам, навіть без нагляду з боку моделі.

Таким чином, результати кластеризації, підкріплені побудованою матрицею відповідності, демонструють здатність системи виявляти закономірності в ознаках фішингових та легітимних доменів, а також слугують якісною вхідною перевіркою перед побудовою моделі класифікації на основі навчання з учителем.

2.2 Методи виявлення вразливостей

На етапі формування системи аналізу атак соціальної інженерії постало завдання обрати найбільш придатні методи виявлення вразливостей, які дозволили б ефективно виявляти потенційно небезпечні об'єкти серед великої кількості вхідних даних. Ураховуючи специфіку таких атак – різноманітність прийомів, постійну зміну тактик і прагнення зловмисників маскувати свої дії під легітимну активність, – до вибору методів потрібно було підійти особливо уважно.

Основна ідея полягала в тому, щоб використати підходи, які добре працюють не тільки на «очевидних» випадках, а й здатні знаходити складні або приховані залежності між ознаками. Враховуючи ці вимоги, для подальшого розгляду було

обрано три методи, кожен з яких має свої переваги та може по-різному проявити себе залежно від характеру вхідних даних.

Метод швидкого дерева (ШД, FastTree) є ефективним алгоритмом побудови моделей бінарної або багатокласової класифікації, який базується на принципах побудови ансамблів дерев рішень [25]. Основною ідеєю методу є поступове формування серії дерев рішень, де кожне нове дерево створюється таким чином, щоб коригувати помилки попередніх дерев, наближаючи модель до мінімізації загальної помилки прогнозування. ШД належить до категорії градієнтного бустингу над деревами рішень, але відрізняється оптимізованою реалізацією, що забезпечує високу швидкість навчання навіть на великих наборах даних.

ШД активно використовується у прикладних задачах машинного навчання, зокрема у системах рекомендацій, прогнозуванні поведінки користувачів, оцінюванні кредитних ризиків, виявленні аномалій та кібербезпеці [26]. Його особливо доцільно застосовувати там, де потрібен компроміс між швидкістю побудови моделі та точністю класифікації.

Для задач аналізу атак соціальної інженерії метод ШД є перспективним завдяки здатності працювати з великими обсягами даних та виявляти складні нелінійні залежності між ознаками, що можуть бути характерними для фішингових доменів або інших проявів шахрайської активності. За рахунок використання ансамблю дерев метод ШД може ефективно інтегрувати в собі різнорідні характеристики об'єктів, такі як рейтинги, показники подібності та ступінь модифікації структур доменних імен.

Переваги методу ШД:

– висока швидкість навчання навіть на великих вибірках завдяки оптимізованому алгоритму побудови дерев;

– здатність моделювати складні нелінійні залежності між ознаками без необхідності їх попередньої обробки;

– відносна стійкість до перенавчання за умови правильного налаштування параметрів.

Недоліки методу ШД:

– зниження інтерпретованості моделі через використання великої кількості дерев та їх комбінацій;

– потреба у ретельному налаштуванні гіперпараметрів для досягнення високої якості класифікації;

– зниження точності на дуже малих вибірках через недостатню кількість інформації для навчання ансамблю.

Отже, метод ШД демонструє високу практичну цінність для завдань, де потрібне швидке та надійне виявлення складних закономірностей у великих масивах даних. Завдяки своїм характеристикам він може слугувати ефективною основою для побудови систем аналізу загроз, зокрема у сфері виявлення атак соціальної інженерії.

Метод випадкових лісів (ВЛ, Random Forest) є ансамблевим методом машинного навчання, який базується на поєднанні великої кількості незалежних дерев рішень для покращення загальної точності класифікації або регресії [27]. Ідея методу полягає у створенні набору дерев, кожне з яких навчається на випадковій підмножині даних та ознак, після чого результати окремих дерев комбінуються шляхом голосування (у задачах класифікації) або усереднення (у задачах регресії). Такий підхід дозволяє суттєво знизити ризик перенавчання, який характерний для окремих дерев рішень, та забезпечити високу узагальнювальну здатність моделі.

ВЛ активно використовуються у найрізноманітніших галузях, включно з біомедичними дослідженнями, фінансовим аналізом, маркетингом, системами рекомендацій, виявленням шахрайства та кібербезпекою [28]. Вони особливо ефективні у випадках, коли дані мають складну внутрішню структуру, містять значну кількість нерелевантних або корельованих ознак, а також коли потрібно працювати з великими обсягами навчальної вибірки.

У задачах аналізу атак соціальної інженерії метод випадкових лісів вважається обґрунтованим вибором завдяки здатності моделі виявляти неочевидні відмінності між легітимними та фішинговими ресурсами на основі складних комбінацій структурних і поведінкових ознак. Використання випадкового підбору ознак для кожного дерева сприяє зменшенню впливу надмірно домінуючих або потенційно

хибних інформативних характеристик, що підвищує стійкість моделі до помилкових рішень.

Переваги методу ВЛ:

– висока точність класифікації завдяки узагальненню результатів великої кількості моделей;

– стійкість до перенавчання та ефективна робота з шумними або частково неповними даними;

– наявність вбудованого механізму оцінки важливості ознак, що сприяє подальшому аналізу результатів.

Недоліки методу ВЛ:

– складність інтерпретації кінцевого результату через велику кількість дерев і глибину ансамблю;

– відносно висока обчислювальна складність, особливо на етапі навчання при великому обсязі даних;

– потреба у точному налаштуванні кількості дерев і глибини дерев для досягнення оптимального балансу між точністю та продуктивністю.

Отже, метод ВЛ добре підходить для задач, де очікується складна взаємодія між великою кількістю ознак, але при цьому існує потреба у високій стійкості до помилок і здатності працювати з неоднорідними даними. Завдяки своїй гнучкості він може ефективно застосовуватись у системах, що потребують надійного виявлення загроз у динамічному середовищі.

Метод градієнтного бустингу (ГБ, Gradient Boosting) належить до класу ансамблевих методів машинного навчання, які поєднують велику кількість слабких моделей (зазвичай дерев рішень) у сильну, високоточну модель класифікації або регресії [29]. Сутність методу полягає у поступовому побудуванні послідовності моделей, кожна з яких навчається на помилках попередніх, з метою покращення загального прогнозу. Основним підходом до навчання в ГБ є мінімізація функції втрат за допомогою градієнтного спуску у просторі функцій. Таким чином, кожна нова модель спрямована на зменшення похибки всієї композиції.

ГБ є одним із найбільш потужних і гнучких методів, що використовуються в прикладному машинному навчанні. Його застосовують у сферах кредитного скорингу, медичної діагностики, систем рекомендацій, аналізу відтоку клієнтів, прогнозування попиту, а також у задачах виявлення аномалій та кіберзагроз [30]. Завдяки здатності працювати з даними складної структури, метод демонструє високу якість класифікації навіть у присутності нерелевантних або сильно корельованих ознак.

У системах виявлення атак соціальної інженерії ГБ доцільно застосовувати тоді, коли важливо враховувати складні, часто нелінійні взаємозв'язки між ознаками, а також потребу в підвищеній чутливості до малопомітних шаблонів у поведінці зловмисників. Наприклад, поєднання кількох ознак – рейтингу домену, показників схожості з відомими ресурсами, частоти використання певних ключових слів, а також рівня вкладеності URL – може мати різну вагу в залежності від їх комбінації. Саме такі тонкі залежності успішно виявляються за допомогою ГБ, що забезпечує глибший рівень аналізу, ніж традиційні моделі.

Завдяки модульній природі цього методу можна легко контролювати ступінь складності моделі, підбираючи кількість базових моделей, глибину дерев, швидкість навчання (learning rate) та інші параметри. Це дозволяє досягати оптимального співвідношення між точністю та продуктивністю. Крім того, сучасні реалізації бустингу, такі як XGBoost, LightGBM або CatBoost, забезпечують високу швидкість обчислень і масштабованість, що дає змогу використовувати їх у режимі обробки великих потоків даних у реальному часі.

Переваги методу ГБ:

– висока точність прогнозування навіть у задачах із зашумленими або складними даними;

– здатність моделювати складні нелінійні залежності між ознаками без необхідності додаткового перетворення даних;

– можливість тонкого налаштування моделі через гнучку систему гіперпараметрів (кількість дерев, глибина, швидкість навчання тощо).

Недоліки методу ГБ:

– висока обчислювальна складність і потреба в значному обсязі ресурсів при роботі з великими вибірками;

– підвищений ризик переобучення у разі неправильно підібраних параметрів моделі;

– обмежена інтерпретованість результатів, що ускладнює пояснення прийнятих рішень у чутливих сферах.

Отже, ГБ доцільно розглядати як інструмент для побудови високоточної моделі в ситуаціях, коли пріоритетом є максимальна якість виявлення складних закономірностей, навіть за умови підвищених вимог до обчислювальної потужності.

Після детального аналізу кожного з розглянутих методів класифікації доцільно здійснити їх порівняння за рядом ключових характеристик, що мають практичне значення при побудові системи виявлення атак соціальної інженерії. Табл. 2.2 ілюструє порівняння трьох методів – ШД, ВЛ та ГБ за найважливішими характеристиками, які враховувались під час вибору алгоритмічної основи для подальшої реалізації.

Таблиця 2.2

Порівняння методів класифікації

Характеристика	**ШД**	**ВЛ**	**ГБ**
Швидкість навчання	Висока	Середня	Низька
Швидкість класифікації	Висока	Середня	Низька
Простота налаштування	Висока	Середня	Низька
Потреба в обчислювальних ресурсах	Низька	Середня	Висока
Стійкість до перенавчання	Висока	Висока	Середня
Інтерпретованість результатів	Висока	Низька	Низька
Робота з великими обсягами даних	Ефективна	Обмежена	Ресурсоємна
Точність класифікації	Середня	Висока	Висока

Вибір методу ШД для аналізу атак соціальної інженерії зумовлений поєднанням високої швидкості навчання та класифікації, простоти налаштування й низьких вимог до обчислювальних ресурсів, що особливо важливо для систем, які працюють у

реальному часі або з великими обсягами даних. У порівнянні з більш складними методами, такими як випадковий ліс чи градієнтний бустинг, ШД забезпечує достатній рівень точності за збереження високої продуктивності. Висока інтерпретованість моделі полегшує аналіз прийнятих рішень і дозволяє використовувати її не лише як інструмент виявлення, а й як компонент аналітичного пояснення ризиків. Такий баланс між швидкодією, надійністю та інтерпретованістю робить метод швидкого дерева оптимальним вибором для побудови системи виявлення загроз соціального типу.

2.3 Розробка сценаріїв атак соціальної інженерії на основі зібраної інформації

Однією з ключових задач системи аналізу соціальних інженерних атак є не лише виявлення вже наявних фішингових або шахрайських доменів, але й розуміння типових механізмів побудови таких атак з урахуванням логіки поведінки зловмисника. Це дозволяє підвищити якість як виявлення, так і превентивної оцінки потенційних загроз. Формалізація дій атакувальної сторони через побудову сценаріїв є ефективним інструментом для опису векторів атак, їх послідовності та цільової аудиторії.

У контексті соціальної інженерії сценарій атаки передбачає логічно структуровану послідовність дій зловмисника, спрямованих на введення в оману користувача з метою отримання конфіденційної інформації або виконання деструктивної дії. Сценарій визначається рядом характеристик: обраним каналом комунікації, типом маніпуляції, ступенем персоналізації, технічними ознаками підробки (зокрема використанням фішингових доменів), а також контекстуальною релевантністю до поведінкових шаблонів жертви.

Для систематизації зібраної інформації та подальшого тестування моделі класифікації було розроблено типові сценарії атак на основі аналізу URL-ознак, які входять до складу використаного датасету, а також узагальнення загальновідомих методик побудови соціоінженерних кампаній. У табл. 2.3 наведено перелік таких

сценаріїв з відповідною класифікацією за параметрами, що використовуються при моделюванні ризику.

Таблиця 2.3

Типові сценарії атак соціальної інженерії на основі структури фішингових доменів

№	Тип атаки	Приклад домену	Основна ознака ураження	Канал поширення	Цільова дія
1	Масове фішинг-звернення	www.paypai.com/login	Схожість із легітимним брендом	Email	Отримання облікових даних
2	Цільова spear-phishing атака	intranet-microsoft-support.com	Персоналізований запит з IT-відділу	Email, месенджери	Захоплення корпоративного облікового запису
3	Приманка для кліку (clickbait)	secure-bonus-amazon.win	Обіцянка виграшу/знижки	Реклама, SMS	Перехід на сайт із вірусним контентом
4	Підміна техпідтримки	appleid-verification.help	Наявність у домені слів «verify», «support»	Вебформи	Введення користувачем паролю
5	Злочин через документообіг	invoice-check-service.info	Симуляція терміновості	Email з вкладенням	Запуск шкідливого коду через PDF
6	Імітація держсайту	gov-uk-payments.net	Наявність офіційних атрибутів	Email, соціальні мережі	Отримання банківських даних

Підготовка таких сценаріїв дозволяє не лише краще сформулювати навчальні приклади для моделі, а й верифікувати її поведінку у випадках, що наближені до реальних умов функціонування. Як видно з таблиці, атаки варіюються як за ступенем складності, так і за каналом доставки. Це свідчить про гнучкість технік соціальної інженерії, що ускладнює їх виявлення на основі обмеженої кількості атрибутів. Однак структурний аналіз доменних імен, зокрема через ознаки подібності (jaccard-

метрики), рейтинг домену (ranking) та редукцію символів (ratio_Rrem), дозволяє виділити спільні закономірності, які характерні для кожного з векторів атак.

Наприклад, сценарії типу spear-phishing та імітації держресурсів зазвичай використовують домени, що містять ключові слова, проте мають низький рейтинг, відсутність у відомих реєстрах (mld_res=0) та високий ступінь модифікації символів. Це дозволяє системі, побудованій на основі моделі машинного навчання, із високою ймовірністю виявити фішинговий характер таких ресурсів навіть за відсутності прямих ознак (наприклад, чорного списку). У свою чергу, масові фішингові кампанії більш одноманітні та легше піддаються детектуванню за рахунок повторюваних шаблонів у доменних іменах.

Висновки до розділу 2

У рамках даного розділу здійснено комплексний аналіз ключових етапів побудови системи виявлення атак соціальної інженерії. Обґрунтовано вибір набору даних Phishing Website Detection Dataset, який містить структуровані ознаки для оцінки ризику фішингової активності на основі доменних імен. Проведено попередню статистичну обробку та візуалізацію даних: зокрема теплову карту кореляцій, графіки розподілу, 3D-представлення об'єктів і парний аналіз змінних за класами. Кластеризація методом KMeans та аналіз відповідності між кластерами і мітками підтвердили наявність структурної сегментації між легітимними та фішинговими доменами.

Оцінено ефективність сучасних методів класифікації: швидкого дерева, випадкового лісу й градієнтного бустингу. Обґрунтовано доцільність використання методу швидкого дерева з огляду на його адаптивність, інтерпретованість, швидкодію та низькі вимоги до ресурсів. Проведено моделювання сценаріїв соціальних інженерних атак із урахуванням маніпулятивних елементів у доменах, що дозволяє використовувати їх для тестування стійкості моделі. Отримані результати формують підґрунтя для подальшого етапу – проєктування й реалізації програмної системи виявлення соціальних загроз.

РОЗДІЛ 3.
ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ ДЛЯ АНАЛІЗУ АТАК СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ

3.1 Архітектура системи аналізу виявлення атак

Побудова системи аналізу атак соціальної інженерії вимагає продуманої архітектури, яка забезпечує стійкість до навантаження, масштабованість, гнучкість у розширенні функціональності та чітке логічне розділення відповідальностей між компонентами. З огляду на необхідність обробки великої кількості вхідних даних, виконання складних обчислень при прогнозуванні й одночасне забезпечення зручної взаємодії з користувачем, архітектура системи повинна відповідати як функціональним, так і нефункціональним вимогам.

Для досягнення зазначених цілей було прийнято рішення про реалізацію системи за принципами трьохрівневої архітектури, яка довела свою ефективність у багатьох інформаційних системах із подібними вимогами. Такий підхід дозволяє логічно відокремити рівень представлення (інтерфейс користувача), рівень бізнес-логіки (механізми обробки, прогнозування та аналізу даних) і рівень доступу до даних (робота з базою даних та зовнішніми джерелами). Кожен рівень може бути реалізований незалежно, що спрощує тестування, обслуговування та подальшу модернізацію системи.

Вибір трьохрівневої архітектури також виправданий тим, що вона дозволяє розподіляти навантаження між клієнтським додатком, серверними обчисленнями та системою зберігання даних, що особливо актуально для задач, пов’язаних із виявленням атак на основі статистичних і машинних моделей. Завдяки чіткому поділу на рівні, розроблена система зберігає можливість масштабування обчислювального модуля без потреби зміни користувацького інтерфейсу або структури даних. Така архітектура створює технологічну основу для подальшого впровадження сервісної

логіки, розширення функціональності та забезпечення безперервної обробки потоків даних у режимі наближеному до реального часу.

На початку було проведено реалізацію структури рівня доступу до даних, що відображає основні об’єкти предметної області. Усі сутності реалізовані у вигляді класів із відповідними властивостями та методами, а взаємозв’язки між ними організовані з урахуванням логіки поділу на функціональні області – облік користувачів, робота з результатами прогнозування, логування подій та управління збереженими моделями машинного навчання. Дана структура дозволяє ефективно відокремити бізнес-логіку від фізичного рівня доступу до даних, забезпечуючи модульність і зрозумілість архітектурного рішення.

На рис. 3.1 представлено діаграму класів рівня даних, яка демонструє основні компоненти та зв’язки між ними.

Рисунок 3.1 – Діаграма класів рівня даних системи аналізу атак

Діаграма класів рівня даних складається з наступних елементів:

– класу UsersProvider, який відповідає за керування обліковими записами користувачів. Він містить методи для додавання, оновлення, видалення користувачів, а також для авторизації, пошуку за логіном, паролем або ідентифікатором. Такий функціонал забезпечує контроль доступу до системи та управління користувацькими правами;

– класу LogsProvider, який реалізує функціональність для ведення журналу подій системи. Він включає методи запису та отримання логів, пов’язаних із діями користувачів і системними подіями, що дозволяє здійснювати аудит активності;

– класу ModelsProvider, призначеного для роботи з об’єктами машинного навчання. Цей клас забезпечує збереження, оновлення та вибірку навчальних моделей, а також асоційованої інформації про дату створення, опис і тип моделі;

– класу ModelInput, що є контейнером для ознак, які надходять на вхід класифікатору. Він включає всі числові характеристики домену: ranking, mld_res, ratio_Rrem, jaccard_RA, jaccard_ARrem та інші, що використовуються для формування прогнозу ризику фішингу;

– класу ModelOutput, який містить результат прогнозування – передбачену мітку (PredictedLabel) та ймовірність належності об’єкта до класу (Score). Це дозволяє формувати зрозумілий зворотній зв’язок для інтерфейсу користувача або зовнішніх систем;

– класу Users, що описує структуру користувача із такими полями, як ім’я, прізвище, логін, пароль, роль і додатковий опис. Клас є частиною базової моделі доступу до системи;

– класу Logs, який містить властивості для опису подій, таких як час дії, її опис, повідомлення, а також ідентифікатор користувача, що її ініціював;

– класу Models, який описує інформацію про збережену модель: назву, дату створення, повідомлення, тип та ідентифікатор. Це дозволяє вести облік і контроль навчальних експериментів.

Структура дозволяє централізовано керувати всіма компонентами, які беруть участь у процесі обробки даних, навчання моделей, логування та авторизації. Чітке

поділення відповідальності між класами забезпечує зручність супроводу коду, повторне використання компонентів та підвищену надійність роботи всієї системи.

З метою забезпечення зручної взаємодії користувача із системою аналізу атак соціальної інженерії було реалізовано повноцінний рівень графічного інтерфейсу, який охоплює всі ключові функції: авторизацію, введення даних, запуск прогнозу, керування користувачами та перегляд журналів подій.

На рис. 3.2 наведено діаграму класів рівня користувацького інтерфейсу, що відображає структуру взаємозв'язків між формами системи, їхні основні методи та логіку обробки подій.

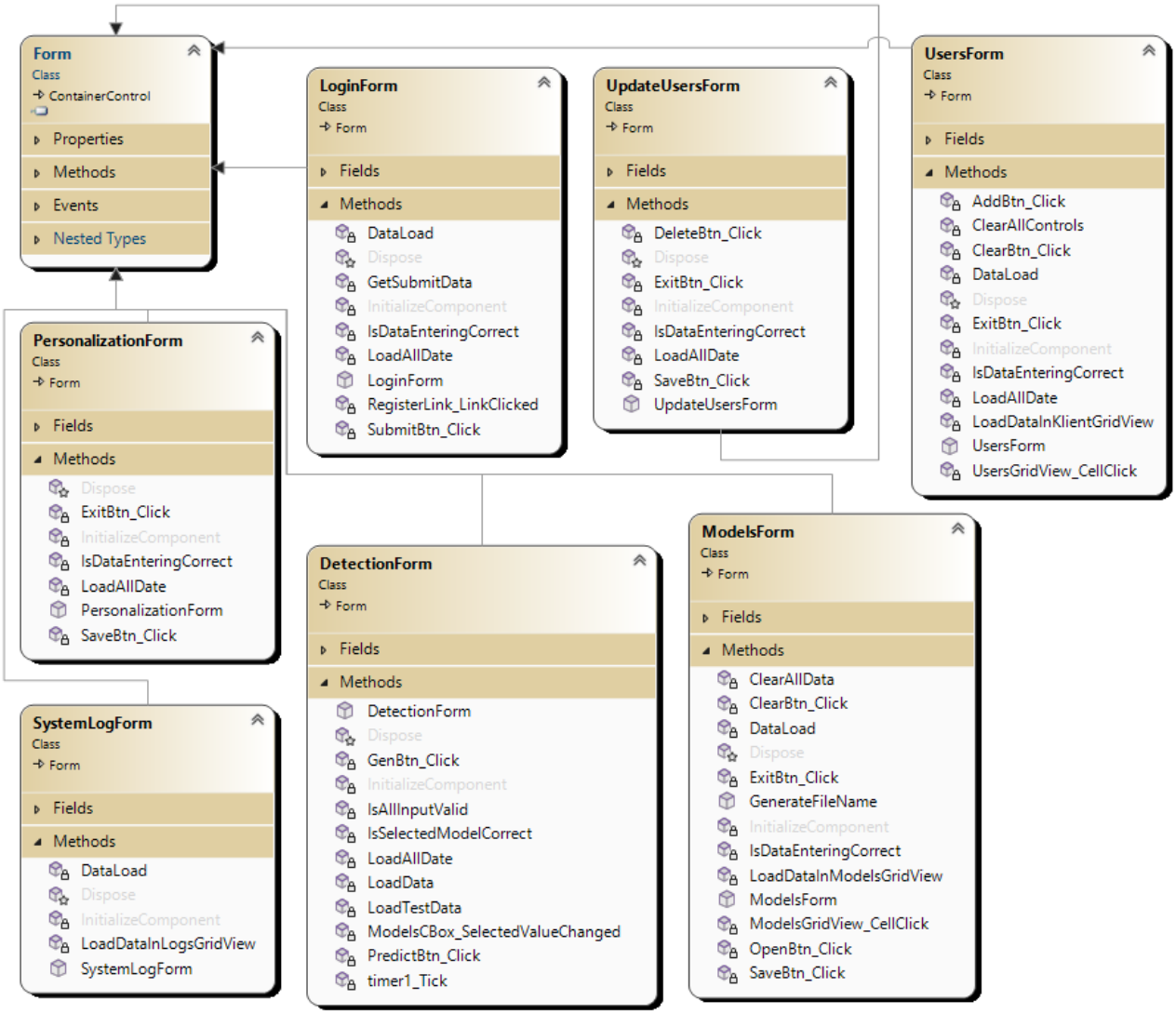

Рисунок 3.2 – Діаграма класів рівня користувацького інтерфейсу

Діаграма класів рівня користувацького інтерфейсу включає такі форми:

– клас LoginForm забезпечує функціональність авторизації користувачів. Він містить методи перевірки коректності введених даних (IsDataEnteringCorrect), обробки події натискання кнопки входу (LoginForm) та реєстрації нового користувача. Реалізовані також методи ініціалізації, завантаження даних і обробки події натискання посилання;

– клас UsersForm відповідає за перегляд і управління переліком користувачів системи. Серед методів: додавання нового запису (AddBtn_Click), очищення таблиці (ClearBtn_Click), завантаження даних у таблицю та обробка події вибору елементів. Цей клас пов’язаний із формами створення та оновлення облікових записів;

– клас UpdateUsersForm реалізує механізми редагування існуючих користувачів. Він включає методи для оновлення інформації, обробки подій кнопок, перевірки даних та збереження змін;

– клас ModelsForm забезпечує управління збереженими моделями машинного навчання. Реалізовано функціонал для перегляду списку моделей, генерації імен файлів, очищення полів та збереження результатів. Події взаємодії з таблицею (ModelsGridView_CellClick) дозволяють відобразити деталі моделі та її параметри;

– клас DetectionForm є ключовим інтерфейсом запуску прогнозу. Він включає перевірку введених даних, активацію обчислень (GenBtn_Click), завантаження вхідних значень і обробку результату класифікації. Реалізовано також таймер для оновлення стану інтерфейсу;

– клас SystemLogForm надає змогу переглядати журнал системних подій. Основні методи забезпечують завантаження даних (DataLoad) та оновлення таблиці логів (LoadDataLogsGridView), що дозволяє здійснювати аудит активності користувачів;

– клас PersonalizationForm відповідає за керування персональними налаштуваннями користувача. Він дозволяє оновити особисту інформацію, зберігати зміни та виходити з профілю. Додаткові методи забезпечують коректне заповнення форми й валідацію введених значень.

Рівень користувацького інтерфейсу реалізовано як набір спеціалізованих форм із чітким функціональним призначенням. Такий підхід забезпечує зручну, логічно

організовану взаємодію з користувачем, дозволяє гнучко масштабувати окремі підмодулі системи без впливу на інші компоненти, а також підтримує належний рівень інтерактивності, необхідний для систем моніторингу та аналізу безпекових інцидентів.

Ключову роль у функціонуванні системи аналізу атак соціальної інженерії відіграє рівень бізнес-логіки, який відповідає за реалізацію прикладної поведінки, перевірку даних, підтримку шифрування, обробку помилок та стандартизацію елементів інтерфейсу. Саме цей рівень забезпечує взаємодію між вхідними даними, користувацьким інтерфейсом та логікою обробки інформації без безпосереднього звернення до бази даних. Класи, що реалізовані на цьому рівні, виконують допоміжні, але критично важливі функції, пов’язані з валідацією, безпекою та стандартизацією системної поведінки.

На рис. 3.3 зображено діаграму класів рівня бізнес-логіки системи, яка відображає основні службові компоненти, що забезпечують внутрішню цілісність та коректність функціонування системи.

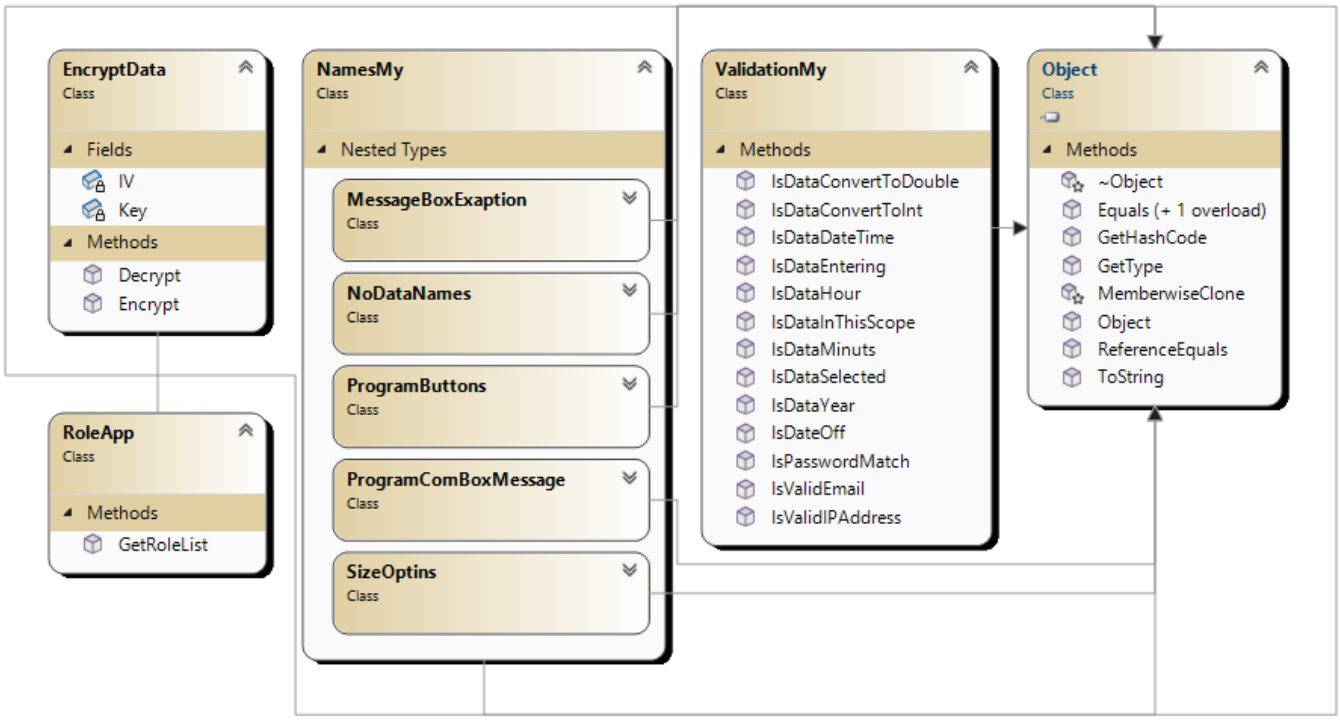

Рисунок 3.3 – Діаграма класів рівня бізнес-логіки системи аналізу атак

Діаграма класів цього рівня складається із:

– клас ValidationMy містить набір методів для перевірки коректності введених даних. Зокрема, реалізовано перевірку формату дати (IsDateTime, IsDataYear, IsDateOff), часових параметрів (IsDataHour, IsDataMinuts), відповідність електронної пошти (IsValidEmail) та IP-адреси (IsValidIPAddress). Також підтримуються методи перевірки пароля (IsPasswordMatch) і типів даних (IsDataConvertToDouble, IsDataConvertToInt). Цей клас є центральним у забезпеченні стабільності вводу та захисту від некоректних або потенційно шкідливих даних.

– клас EncryptData відповідає за шифрування та дешифрування інформації. Містить ключ (Key) і вектор ініціалізації (IV) для реалізації симетричного шифрування, а також методи Encrypt і Decrypt. Його функціональність може застосовуватись для зберігання паролів, токенів або інших чутливих елементів системи.

– клас RoleApp реалізує функцію отримання переліку доступних ролей у системі через метод GetRoleList. Це забезпечує гнучке керування рівнями доступу користувачів до функціональних частин застосунку.

– клас NamesMy містить вкладені допоміжні класи, які слугують для централізованого зберігання системних назв, повідомлень, текстів кнопок, назв елементів форм, розмірів тощо.

Отже, рівень бізнес-логіки системи структуровано таким чином, щоб ізолювати загальновживані операції перевірки, шифрування та обслуговування інтерфейсу в окремі класи. Це забезпечує чистоту архітектури, підвищує модульність проєкту та сприяє повторному використанню коду. Завдяки цьому досягається розділення відповідальностей, що є важливою умовою для надійного, масштабованого і безпечного програмного забезпечення.

3.2 Реалізація механізмів виявлення фішингових атак

Розроблення механізмів виявлення фішингових атак передбачає послідовну інтеграцію інструментів машинного навчання із засобами обробки вхідних даних,

формування векторів ознак і реалізацію взаємодії користувача із системою. Одним із перших кроків у процесі запуску моделі є завантаження тренувального набору даних, що містить інформацію про домени, їхні характеристики та мітки класифікації. Для цього було реалізовано функціональність відкриття файлу з вибіркою у форматі .csv, яка дозволяє інтерактивно вказати шлях до відповідного джерела даних. Механізм завантаження ініціюється під час взаємодії користувача з кнопкою відкриття, після чого система зчитує шлях до обраного файлу й відображає його у графічному інтерфейсі. Реалізація цього функціоналу наведена у фрагменті коду на рис. 3.4.

```
private void OpenBtn_Click(object sender, EventArgs e) {
  // Створення діалогового вікна для відкриття файлу
  OpenFileDialog openFileDialog = new OpenFileDialog {
    Filter = "Text files (*.csv)|*.csv|All files (*.*)|*.*",
    FilterIndex = 2,
    RestoreDirectory = true
  };
  if (openFileDialog.ShowDialog() == DialogResult.OK) {
    _Path = openFileDialog.FileName;
    FileNameTBox.Text = openFileDialog.FileName;
```

Рисунок 3.4 – Фрагмент методу для відкриття тренувальних даних

У цьому коді реалізовано обробник події натискання на кнопку, відповідальну за відкриття діалогового вікна вибору файлу. За допомогою класу OpenFileDialog користувачеві пропонується обрати файл з розширенням .csv, після чого, у разі підтвердження вибору, шлях до цього файлу зберігається у змінну _Path. Одночасно обраний шлях виводиться в текстове поле FileNameTBox, що дозволяє користувачу переконатися у правильності завантаженого джерела.

Укоді на рис. 3.5 відбувається створення екземпляра об’єкта MLContext, який є центральною точкою для всіх операцій, пов’язаних із машинним навчанням у ML.NET. Він ініціалізується зі сталим параметром seed, що дорівнює 1. Встановлення початкового значення генератора випадкових чисел забезпечує відтворюваність результатів при навчанні та тестуванні моделі. Це особливо важливо під час досліджень або налагодження, коли необхідно отримувати однакові результати при кожному запуску. Об’єкт mlContext виступає в ролі базового середовища, через яке

здійснюється доступ до всіх функціональних підсистем ML.NET, таких як завантаження даних, побудова моделей, трансформації ознак і оцінювання.

```
// Ініціалізація MLContext
mlContext = new MLContext(seed: 1);
```

Рисунок 3.5 – Створення екземпляра об’єкта MLContext

На рис. 3.6 наведено реалізацію завантаження даних із CSV-файлу до внутрішньої структури ML.NET для подальшої обробки. Метод LoadFromTextFile ініціалізує об'єкт dataView, який містить дані, зчитані з файлу, шлях до якого задається змінною _Path.

```
dataView = mlContext.Data.LoadFromTextFile<ModelInput>(
    path: _Path, hasHeader: true, separatorChar: ',',
    allowQuoting: true);
var enumerator =
  mlContext.Data.CreateEnumerable<ModelInput>(dataView,
  reuseRowObject: false);
int count = 0;
foreach (var row in enumerator) {
  count++;
}
```

Рисунок 3.6 – Завантаження даних із CSV-файлу до структури ML.NET

Передбачено, що файл має заголовки колонок, використовується кома як роздільник, а також дозволено обробку значень у лапках. Після завантаження виконується спроба пройтися по всіх рядках вибірки з використанням методу CreateEnumerable, що перетворює IDataView у звичайну колекцію об’єктів типу ModelInput. Це дозволяє проітерувати всі рядки й переконатися, що дані коректно зчитуються, не викликаючи винятків. Додатково лічильник count підраховує кількість оброблених записів, що може бути корисним для діагностики або виведення статистики про обсяг навчального набору.

У фрагменті коду на рис. 3.7 реалізується підготовка даних до навчання моделі машинного навчання та створення конвеєра.

```
var split = mlContext.Data.TrainTestSplit(dataView, testFraction: 0.2);
var dataProcessPipeline = mlContext.Transforms.Text.FeaturizeText(
  outputColumnName: "DomainFeaturized",
  inputColumnName: nameof(ModelInput.Domain))
.Append(mlContext.Transforms.Concatenate(
  "Features", "DomainFeaturized",
  nameof(ModelInput.Ranking), nameof(ModelInput.Mld_res),
  nameof(ModelInput.Mld_ps_res), nameof(ModelInput.Card_rem),
  nameof(ModelInput.Ratio_Rrem), nameof(ModelInput.Ratio_Arem),
  nameof(ModelInput.Jaccard_RR), nameof(ModelInput.Jaccard_RA),
  nameof(ModelInput.Jaccard_AR), nameof(ModelInput.Jaccard_AA),
  nameof(ModelInput.Jaccard_ARrd), nameof(ModelInput.Jaccard_ARrem)));
```

Рисунок 3.7 – Підготовка даних до навчання моделі та створення конвеєра

На початку виконано розділення зчитаного набору dataView на тренувальну та тестову вибірки за допомогою методу TrainTestSplit, де 20% даних резервуються для тестування. Це необхідно для подальшої валідації моделі на незалежних прикладах, що не використовувались у навчанні. Далі формується конвеєр попередньої обробки даних, який складається з двох ключових етапів. На першому етапі відбувається перетворення текстової змінної Domain у числовий вектор ознак за допомогою методу FeaturizeText. Цей крок дозволяє моделі працювати з доменними іменами як із векторизованими об’єктами, що відображають їхню структуру та семантику. Другий етап полягає в об'єднанні цієї векторизованої ознаки з іншими числовими полями, зокрема Ranking, Mld_res, Mld_ps_res, коефіцієнтами схожості Жаккара та показниками структурних редукцій. Усі ці стовпці поєднуються в єдиний вектор Features, який виступає входом для класифікатора. Створений конвеєр формує повноцінну матрицю ознак, що буде передана в модель для навчання.

На рис. 3.8 наведено код, у якому здійснюється вибір алгоритму машинного навчання, який буде використано для класифікації, та його включення до загального конвеєра обробки даних. Як модель обрано метод швидкого дерева, що добре підходить для задач бінарної класифікації. Алгоритм конфігурується із зазначенням назви цільової змінної (Label) та назви вектору ознак (Features), сформованого на попередньому етапі. Після цього обраний тренувальний алгоритм додається до конвеєра попередньої обробки шляхом послідовного з'єднання з вже підготовленим

блоком трансформацій. У результаті формується повноцінний тренувальний конвеєр, який поєднує етапи векторизації, об’єднання ознак і навчання моделі. Така побудова дозволяє автоматизовано обробити дані та одразу передати їх у класифікатор у єдиному процесі.

```
var trainer = mlContext.BinaryClassification.Trainers
    .FastTree(labelColumnName: "Label", featureColumnName: "Features");
var trainingPipeline = dataProcessPipeline.Append(trainer);
```

Рисунок 3.8 – Вибір алгоритму машинного навчання

У фрагменті коду на рис. 3.9 реалізується процес навчання моделі на основі раніше сформованого конвеєра trainingPipeline та тренувальної вибірки split.TrainSet. Перед початком обчислень у текстове поле RaportTBox виводиться повідомлення про запуск процедури навчання, після чого фіксується час початку за допомогою об'єкта DateTime.Now. Метод Fit запускає повний цикл навчання моделі, в якому спочатку здійснюється перетворення вхідних даних згідно з конвеєром, а потім виконується оптимізація ваг обраного алгоритму класифікації. Після завершення обчислень фіксується тривалість процесу навчання як різниця між кінцевим і початковим часом. Отримане значення в секундах виводиться до графічного інтерфейсу з повідомленням про завершення побудови моделі. Такий підхід дозволяє користувачеві контролювати перебіг процесу навчання та оцінити його швидкодію.

```
RaportTBox.Text += ("Навчання моделі...\r\n");
var startTime = DateTime.Now;
trainedModel = trainingPipeline.Fit(split.TrainSet);
var duration = DateTime.Now - startTime;
RaportTBox.Text +=($"Навчання завершено за" +
  $" {duration.TotalSeconds:F2} с.\r\n\r\n");
```

Рисунок 3.9 – Процес навчання моделі

На рис. 3.10 реалізовано код оцінювання якості побудованої моделі на незалежній тестовій вибірці.

```
var metrics = mlContext.BinaryClassification.Evaluate(
    data: testSetTransform,
    labelColumnName: "Label",
    scoreColumnName: "Score");
```

Рисунок 3.10 – Оцінювання якості побудованої моделі

Метод Evaluate з простору BinaryClassification викликається для аналізу результатів прогнозування, які були отримані шляхом трансформації тестових даних через навчений конвеєр. Аргументом data передається набір testSetTransform, що містить передбачення для кожного прикладу, а параметри labelColumnName і scoreColumnName вказують відповідно на колонку з фактичними мітками класів і колонку з розрахованими значеннями ймовірності належності об’єкта до позитивного класу. У результаті виконання функції формується об’єкт metrics, який містить усі основні показники якості моделі: точність, повноту, F1-міру, AUC та інші. Отримані значення дозволяють кількісно оцінити ефективність класифікації та визначити, наскільки модель здатна виявляти фішингові об’єкти без перенавчання або втрати узагальнювальної здатності.

У фрагменті коду на рис. 3.11 реалізовано виведення результатів оцінювання моделі у вигляді узагальнених та покласових метрик, які формуються у вигляді структурованого текстового звіту. Спочатку у текстове поле RaportTBox додається заголовок, що позначає початок блоку з глобальними метриками, які відображають загальну якість класифікації по всій тестовій вибірці без поділу на класи. До таких показників належать точність (Accuracy), збалансована метрика F1, площа під ROC-кривою (AUC), а також середнє значення точності (Precision) і повноти (Recall) для позитивного класу.

```
RaportTBox.Text +=("\n===== ГЛОБАЛЬНІ МЕТРИКИ (усі класи разом) =====\r\n");
RaportTBox.Text +=($"Accuracy:  {metrics.Accuracy:F4}\r\n");
RaportTBox.Text += ($"F1 Score:  {metrics.F1Score:F4}\r\n");
RaportTBox.Text+=($"AUC:       {metrics.AreaUnderRocCurve:F4}\r\n");
RaportTBox.Text+= ($"Precision (Overall): {metrics.PositivePrecision:F4}\r\n");
RaportTBox.Text +=($"Recall    (Overall): {metrics.PositiveRecall:F4} \r\n\r\n");

RaportTBox.Text += ("\n===== МЕТРИКИ ДЛЯ КОЖНОГО КЛАСУ =====\r\n");
RaportTBox.Text += ($"Фішинг (Label=true) -> TP={tp}, FP={fp}, FN={fn}, TN={tn}\r\n");
RaportTBox.Text += ($"  Precision(fish) = {fishPrecision:F4}\r\n");
RaportTBox.Text += ($"  Recall(fish)    = {fishRecall:F4}\r\n");
RaportTBox.Text += ($"  F1(fish)        = {fishF1:F4}\r\n\r\n");

RaportTBox.Text += ($"Не фішинг  (Label=false) -> TN={tn}, FP={fp}, FN={fn}, TP={tp}\r\n");
RaportTBox.Text += ($"  Precision(noFish) = {noFishPrecision:F4}\r\n");
RaportTBox.Text += ($"  Recall(noFish)    = {noFishRecall:F4}\r\n");
RaportTBox.Text += ($"  F1(noFish)        = {noFishF1:F4}\r\n\r\n");
```

Рисунок 3.11 – Виведення результатів оцінювання моделі

Після цього формується розділ з розгорнутими метриками окремо для кожного класу. Для випадків, коли об’єкт класифіковано як фішинг (Label=true), відображається кількість істинно позитивних (TP), хибнопозитивних (FP), хибнонегативних (FN) і істинно негативних (TN) прикладів. Далі для цього класу розраховуються й виводяться його специфічні значення точності, повноти та F1-міри. Аналогічно обробляється і негативний клас, який позначає безпечні (нефішингові) домени. Метрики для нього також деталізовано з урахуванням кількості коректно або некоректно класифікованих прикладів.

На рис. 3.12 реалізовано логіку збереження навченої моделі машинного навчання після її побудови. Подія активується при натисканні користувачем кнопки «Зберегти», і насамперед перевіряється коректність введених даних за допомогою методу IsDataEnteringCorrect. Якщо перевірка проходить успішно, система формує унікальне ім’я файлу за допомогою функції GenerateFileName, до якого додається шлях для збереження у підкаталозі \teach\ у межах директорії, де запущено застосунок. Для цього шлях до поточної збірки визначається динамічно через Location.

```
private void SaveBtn_Click(object sender, EventArgs e) {
  if (IsDataEnteringCorrect()) {
    //Зберігання моделі
    string pathName = @"\teach\" + GenerateFileName() + ".zip";
    string localProj =
      System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location);
    _ModelsProvider.InsertModels(ModelsNamesTBox.Text, pathName);
    mlContext.Model.Save(trainedModel, dataView.Schema, localProj + pathName);
    ClearAllData();
    _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId,
      "Було навчено модель " +
      ModelsNamesTBox.Text, DateTime.Now);
    MessageBox.Show("Дані успішно збережено!");
  }
}
```

Рисунок 3.12 – Подія для зберігання навченої моделі

Інформація про модель, зокрема її назва та шлях збереження, передається об’єкту _ModelsProvider, який відповідає за реєстрацію моделі в базі даних. Безпосереднє збереження моделі на диск виконується за допомогою методу Save класу mlContext.Model, у якому вказується сам навчений об’єкт, схема вхідних даних і повний шлях до файлу. Після завершення збереження виконується очищення введених даних на формі через виклик ClearAllData, а також запис у журнал подій через _LogsProvider із фіксацією дій поточного користувача, дати й короткого опису дії. Завершується процес виведенням системного повідомлення, що підтверджує успішне завершення операції.

У методі на рис. 3.13 реалізується процедура завантаження збереженої моделі машинного навчання з диску для подальшого використання в системі.

```
private void LoadModel(string FilePath) {
  string localProj = Application.StartupPath + FilePath;
  var mlContext = new MLContext();
  // Визначення DataViewSchema для навченої моделі
  DataViewSchema modelSchema;
  // Завнтаження тренованої моделі
  model = context.Model.Load(localProj, out modelSchema);
}
```

Рисунок 3.13 – Метод для завантаження даних моделі

Метод LoadModel приймає як аргумент шлях до файлу моделі відносно каталогу запуску застосунку. Повний шлях до файлу формується шляхом об'єднання директорії старту програми (Application.StartupPath) з переданим відносним шляхом. Далі створюється новий екземпляр MLContext, що використовується як середовище для роботи з моделями та обробки даних. Перед завантаженням моделі оголошується змінна modelSchema типу DataViewSchema, яка необхідна для зберігання структури вхідних даних моделі. Завантаження відбувається за допомогою методу Load, що належить до простору Model, де зчитується збережений файл і повертається десеріалізований об’єкт моделі. Паралельно у вихідний параметр out modelSchema записується структура даних, яка буде використана для побудови предиктора. Такий механізм дозволяє повторно використовувати раніше натреновані моделі без необхідності повторного навчання, що суттєво прискорює роботу застосунку при запуску або інтеграції в уже працююче середовище.

У методі на рис. 3.14 реалізовано перемикач режиму моніторингу, який активується при натисканні кнопки з ідентифікатором GenBtn. Спершу перевіряється, чи обрана модель є коректною для запуску, використовуючи метод IsSelectedModelCorrect. Якщо перевірка успішна, здійснюється перемикання стану таймера timer1, що виконує періодичне оновлення даних або запуск прогнозування у фоновому режимі.

```
private void GenBtn_Click(object sender, EventArgs e) {
    if (IsSelectedModelCorrect()) {
        if (timer1.Enabled) {
            timer1.Enabled = false;
            GenBtn.Text = "Моніторити";
        } else {
            timer1.Enabled = true;
            GenBtn.Text = "Зупинити";
        }
    }
}
```

Рисунок 3.14 – Метод обробки запуску та зупинки моніторингу мережі

У разі, якщо таймер уже активний, його робота припиняється і кнопка отримує текстове позначення «Моніторити», що сигналізує про готовність до повторного запуску. Якщо ж таймер був неактивним, він запускається, а текст на кнопці змінюється на «Зупинити», що вказує на активний режим моніторингу. Такий підхід дозволяє реалізувати зручний інтерфейс для керування потоком аналізу в реальному часі, даючи змогу користувачеві запускати та зупиняти процес одним кліком без необхідності змінювати додаткові параметри.

Фрагмент коду на рис. 3.15 відповідає за формування текстового звіту, в якому відображаються всі вхідні параметри, що передаються до моделі для аналізу домену. Для зручності і компактності виведення використовується об'єкт типу StringBuilder, що дозволяє ефективно формувати багаторядковий текст без створення зайвих проміжних об’єктів. У звіт послідовно додаються значення всіх ключових характеристик мережевого ресурсу, зокрема доменне ім’я, глобальний рейтинг, результати попередньої обробки рівня MLD, кількість залишкових компонентів у структурі адреси, а також числові показники, що характеризують ступінь подібності домену до відомих шаблонів – індекси Жаккара для різних парних співвідношень.

```
var report = new StringBuilder();
report.AppendLine("🎯 Дані мережі:");
report.AppendLine($"Домен: {input.Domain}");
report.AppendLine($"Рейтинг: {input.Ranking}");
report.AppendLine($"MLD результат: {input.Mld_res}");
report.AppendLine($"MLD.ps результат: {input.Mld_ps_res}");
report.AppendLine($"Кількість залишкових компонентів: {input.Card_rem}");
report.AppendLine($"Співвідношення Rrem: {input.Ratio_Rrem}");
report.AppendLine($"Співвідношення Arem: {input.Ratio_Arem}");
report.AppendLine($"Індекс Жаккара (RR): {input.Jaccard_RR}");
report.AppendLine($"Індекс Жаккара (RA): {input.Jaccard_RA}");
report.AppendLine($"Індекс Жаккара (AR): {input.Jaccard_AR}");
report.AppendLine($"Індекс Жаккара (AA): {input.Jaccard_AA}");
report.AppendLine($"Індекс Жаккара (ARrd): {input.Jaccard_ARrd}");
report.AppendLine($"Індекс Жаккара (ARrem): {input.Jaccard_ARrem}");
```

Рисунок 3.15 – Формування звіту вхідних параметр

У фрагменті коду на рис. 3.16 реалізовано логіку прогнозування фішингового ризику на основі вхідної інформації. Об'єкт input, який містить усі необхідні ознаки домену, поміщається в масив, що дозволяє конвертувати його в IDataView через метод LoadFromEnumerable. Це необхідно для того, щоб забезпечити сумісність формату даних із навченою моделлю. Після цього на отриманому представленні викликається метод Transform, що застосовує попередньо збережену модель до вхідного прикладу й генерує прогноз.

```
var inputArray = new[] { input };
var testView = context.Data.LoadFromEnumerable(inputArray);
var predictions = model.Transform(testView);
var result =
  context.Data.CreateEnumerable<ModelOutput>(predictions,
  reuseRowObject: false).Single();

report.AppendLine("\n🔎 Результат прогнозування:");
report.AppendLine($"Ймовірність фішингової атаки:" +
  $" {Math.Max(0, Math.Min(1, result.Score)) * 100:F2}%");
report.AppendLine($"Результат: {(result.PredictedLabel ?
  "⚠ Фішинговий домен" : "✅ Легітимний домен")}");
ReportTBox.Text = report.ToString();
```

Рисунок 3.16 – Прогнозування фішингової атаки

Результати обчислення витягуються у вигляді об'єкта типу ModelOutput за допомогою методу CreateEnumerable, з якого вибирається єдиний елемент, оскільки аналізується лише один домен. У сформований раніше звіт додається блок, що містить інформацію про ймовірність фішингової активності – вона обмежується значеннями в межах від 0 до 1 для уникнення помилкових виводів – і текстова інтерпретація результату на основі передбаченого класу. Залежно від значення булевого поля PredictedLabel, користувачу виводиться повідомлення про потенційну небезпеку або про належність ресурсу до легітимних. У підсумку весь накопичений текст звіту виводиться до елементу інтерфейсу ReportTBox, який відображає результат класифікації у зручному й зрозумілому форматі.

3.3 Інтеграція механізмів захисту

У межах розробки системи аналізу фішингових атак особливу увагу було приділено забезпеченню конфіденційності та цілісності внутрішніх даних. Це особливо важливо у випадках, коли система зберігає або обробляє чутливу інформацію, зокрема ідентифікатори користувачів, назви моделей, службові мітки або інші об'єкти, що можуть бути потенційно використані вразливими сторонами. Одним із надійних інструментів забезпечення криптографічного захисту даних є симетричний алгоритм шифрування AES (Advanced Encryption Standard). Даний алгоритм був обраний завдяки своїй швидкодії, стійкості до криптоаналізу та підтримці апаратного прискорення.

Алгоритм AES використовує фіксовану довжину блоку (128 біт) і підтримує ключі довжиною 128, 192 або 256 біт. У симетричному режимі шифрування той самий ключ застосовується як для шифрування, так і для розшифрування, що робить його особливо придатним для локального зберігання або тимчасової обробки даних. У рамках цієї системи AES використовується для шифрування службових рядків, зокрема в момент збереження конфіденційної інформації або взаємодії з користувацькими модулями.

Нижче на рис. 3.17 наведено фрагмент, у якому реалізовано метод шифрування текстового рядка у середовищі .NET.

```
public string Encrypt(string originalString) {
  // Перевірка, чи вхідний рядок не є null або порожнім
  if (String.IsNullOrEmpty(originalString)) {
    throw new ArgumentNullException("The string which needs to be encrypted can not be null.");
  }
  // Використання AES шифрування з використанням CryptoServiceProvider
  using (AesCryptoServiceProvider aes = new AesCryptoServiceProvider()) {
    // Встановлення ключа та IV для шифрування
    aes.Key = Key;
    aes.IV = IV;
```

Рисунок 3.17 – Фрагмент методу шифрування текстового рядка

На початку виконується перевірка вхідного значення на відсутність або порожнечу, і в разі порушення викидається виняток із поясненням. Для

криптографічної обробки використовується об'єкт AesCryptoServiceProvider, що реалізує стандартний алгоритм AES. У межах поточного виклику встановлюються два ключові параметри – шифрувальний ключ (Key) та вектор ініціалізації (IV), які необхідні для забезпечення детермінованості результату і уникнення повторень у шифрованому тексті. Подальші дії включають створення криптографічного потоку, запис шифрованих даних у пам’ять та повернення їх у вигляді рядка, однак ці операції продовжуються поза межами наведеного фрагмента. Така реалізація дозволяє легко інтегрувати механізм шифрування в будь-яку частину системи, де необхідна захищена обробка текстової інформації.

На рис. 3.18 реалізовано основні дії, пов’язані з фактичним процесом шифрування текстових даних. Для початку створюється об’єкт типу MemoryStream, який слугує тимчасовим буфером для зберігання результату шифрування в оперативній пам’яті без залучення файлової системи. Далі на основі цього потоку ініціалізується об’єкт CryptoStream, який виконує роль проміжного рівня між потоком і алгоритмом шифрування. Для шифрування використовується метод CreateEncryptor, який приймає заздалегідь визначений ключ і вектор ініціалізації, що гарантує коректну криптографічну трансформацію.

```
MemoryStream memoryStream = new MemoryStream();
// Створення потоку шифрування
CryptoStream cryptoStream = new CryptoStream(memoryStream,
    aes.CreateEncryptor(aes.Key, aes.IV), CryptoStreamMode.Write);
// Запис вхідного рядка в потік шифрування
StreamWriter writer = new StreamWriter(cryptoStream);
writer.Write(originalString);
writer.Flush();
```

Рисунок 3.18 – Процес шифрування текстових даних

Дані записуються в цей потік за допомогою стандартного текстового записувача StreamWriter. Переданий рядок originalString записується в криптографічний потік у зашифрованому вигляді, після чого виконується явне очищення буфера методом Flush, що гарантує збереження всього обсягу зашифрованих даних у пам’яті.

У завершальній частині процедури шифрування виконується фіналізація обробки даних у криптографічному потоці. Метод FlushFinalBlock закриває потік шифрування, забезпечуючи правильне доповнення блоку згідно з вимогами алгоритму AES, та гарантує, що жодна частина зашифрованих даних не залишиться в буфері. Після цього викликається Flush для текстового записувача, аби остаточно переконатися в тому, що весь вміст повністю передано до пам’яті.

```
cryptoStream.FlushFinalBlock();
writer.Flush();
// Повернення зашифрованих даних у форматі Base64
return Convert.ToBase64String(memoryStream.ToArray());
```

Рисунок 3.19 – Завершальна частина процедури шифрування

Заключним кроком є перетворення зашифрованого вмісту, який зберігається у MemoryStream, у масив байтів, а потім – у зручний для передачі або збереження рядковий формат Base64. Це кодування дозволяє безпечно представити двійкові дані у вигляді звичайного тексту, що зручно для роботи з файлами, базами даних або мережевими інтерфейсами. Повернення результату у вигляді рядка Base64 забезпечує універсальність і сумісність із більшістю текстових середовищ.

У фрагменті коду на рис. 3.20 реалізується початок методу для розшифрування зашифрованого рядка. Метод приймає вхідний параметр у форматі Base64, який був отриманий у результаті попереднього шифрування.

```
public string Decrypt(string cryptedString) {
  // Перевірка, чи вхідний рядок для розшифрування не є null або порожнім
  if (String.IsNullOrEmpty(cryptedString)) {
    throw new ArgumentNullException("The " +
      "string which needs to be decrypted can not be null.");
  }
```

Рисунок 3.20 – Початок методу для розшифрування зашифрованого рядка

На першому кроці виконується перевірка, чи передане значення дійсно містить дані, тобто не є порожнім або нульовим. Якщо рядок не проходить цю перевірку,

генерується виняток типу ArgumentNullException із поясненням, що розшифровування неможливе через відсутність вхідного значення. Такий захист дозволяє уникнути помилок на ранньому етапі виконання й гарантує, що подальша обробка буде здійснюватися лише над коректно переданим параметром.

Рис. 3.21 відображає код у якомуосновна частина процесу розшифрування зашифрованого рядка. Спочатку вхідний рядок, представлений у форматі Base64, декодується у масив байтів за допомогою методу Convert.FromBase64String, що дозволяє отримати його оригінальне бінарне подання. Отримані байти містять зашифровані дані, готові для подальшої криптографічної обробки.

```
// Конвертація рядка з Base64 у масив байтів
var fullCipher = Convert.FromBase64String(cryptedString);
// Використання AES шифрування з використанням CryptoServiceProvider
using (AesCryptoServiceProvider aes = new AesCryptoServiceProvider()) {
  // Встановлення ключа та IV для розшифрування
  aes.Key = Key;
  aes.IV = IV;
```

Рисунок 3.21 – Частина процесу розшифрування зашифрованого рядка

Для виконання розшифрування використовується клас AesCryptoServiceProvider, який надає реалізацію симетричного алгоритму AES. Усередині блоку using створюється екземпляр об’єкта AES, у який передаються попередньо збережені ключ і вектор ініціалізації. Встановлення цих параметрів є обов’язковим, оскільки вони мають точно відповідати тим, що використовувалися під час шифрування. Без відповідності ключів дешифрування не зможе відтворити вихідний текст, тому на цьому етапі забезпечується узгодженість обох процесів.

У фрагменті коду на рис. 3.22 реалізує повну процедуру розшифрування бінарних даних, які раніше були зашифровані алгоритмом AES. Для початку створюється об’єкт MemoryStream, у який завантажується масив байтів, що містить зашифрований вміст. Цей потік служить джерелом для подальшої обробки.

```
MemoryStream memoryStream = new MemoryStream(fullCipher);
// Створення потоку розшифрування
CryptoStream cryptoStream = new CryptoStream(memoryStream,
    aes.CreateDecryptor(aes.Key, aes.IV), CryptoStreamMode.Read);
// Читання розшифрованих даних
StreamReader reader = new StreamReader(cryptoStream);
return reader.ReadToEnd(); // Повернення розшифрованого рядка
```

Рисунок 3.22 – Процедура розшифрування даних

Також ініціалізується об’єкт CryptoStream, що виконує роль проміжного декриптора. Він пов’язаний із потоком у пам’яті й налаштований на читання вхідних байтів за допомогою методу CreateDecryptor, у який передаються актуальні значення ключа та вектора ініціалізації. Цей об’єкт забезпечує поетапне дешифрування даних під час їх зчитування. Для зчитування розшифрованого тексту використовується StreamReader, який читає з криптографічного потоку вже декодовані символи. Весь вміст зчитується до кінця через ReadToEnd, а результатом роботи методу є повернення розшифрованого текстового рядка, готового до використання в інтерфейсі або для подальшої логіки програми. Така структура забезпечує безпечну, пряму і повністю прозору для розробника реалізацію симетричного дешифрування.

3.4 Тестування системи та аналіз продуктивності

Після завершення етапу розробки було проведено комплексне тестування системи виявлення фішингових атак для перевірки її функціональності, точності та стабільності роботи в умовах реального використання. Основною метою цього етапу є оцінка якості побудованої моделі, перевірка її здатності обробляти великі обсяги вхідних даних, а також вимірювання продуктивності окремих компонентів системи у процесі обчислень.

На рис. 3.23 наведено результат одного з навчальних прогонів, під час якого було завантажено та успішно прочитано 91 682 рядки з навчальної вибірки. Повний процес побудови моделі завершився за 41,39 секунди, що свідчить про високу швидкодію реалізованої архітектури навіть при значному обсязі вхідних даних.

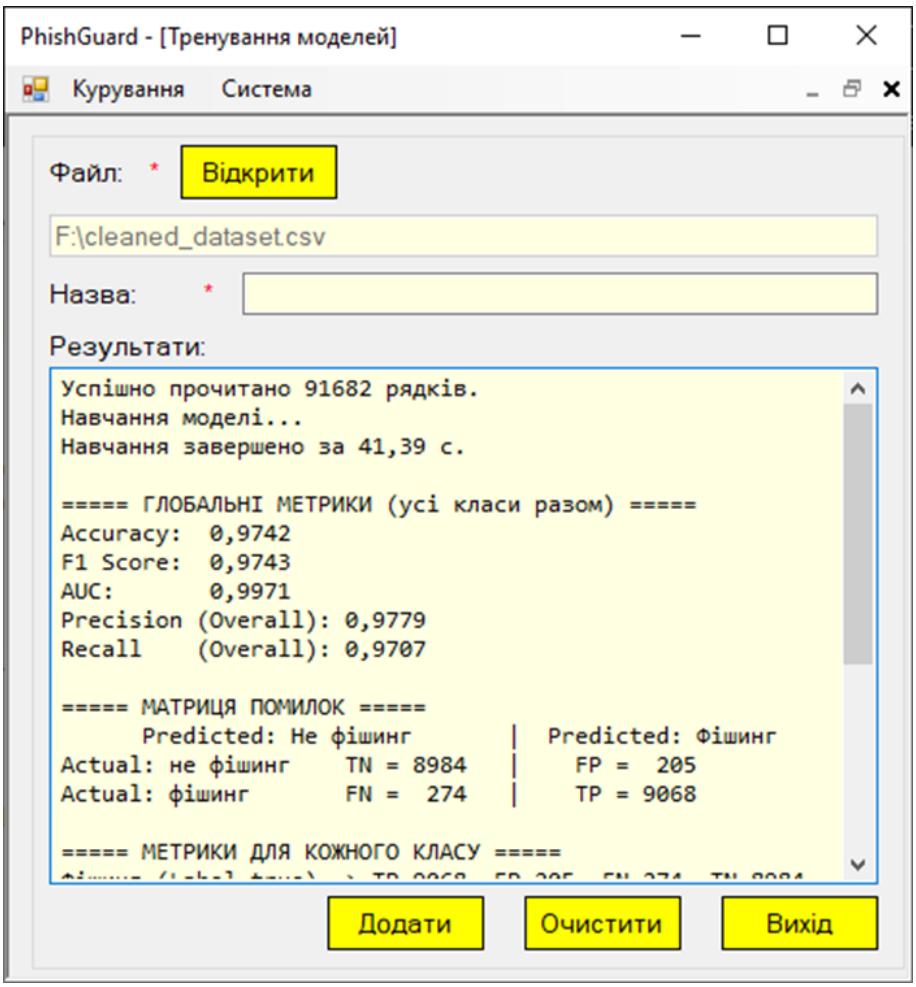

Рисунок 3.23 – Результат тренування моделі машинного навчання

Отримані глобальні метрики підтверджують високу якість моделі: точність класифікації склала 97,42%, значення F1-міри – 97,43%, а площа під ROC-кривою досягла 0,9971, що свідчить про майже ідеальну здатність розрізняти класи. Загальна точність позитивних прогнозів становила 97,79%, при цьому модель змогла виявити 97,07% усіх фішингових доменів, що є дуже високим показником для задачі бінарної класифікації. Такі результати підтверджують ефективність побудованої системи як з точки зору точності, так і з погляду продуктивності.

На рисунку 3.24 представлено розподіл основних метрик класифікації за кожним із двох класів – фішингові та нефішингові домени. Для кожного класу наведено значення точності, повноти та F1-міри, що дозволяє оцінити не лише загальну ефективність моделі, але й її поведінку при роботі з різними типами об'єктів.

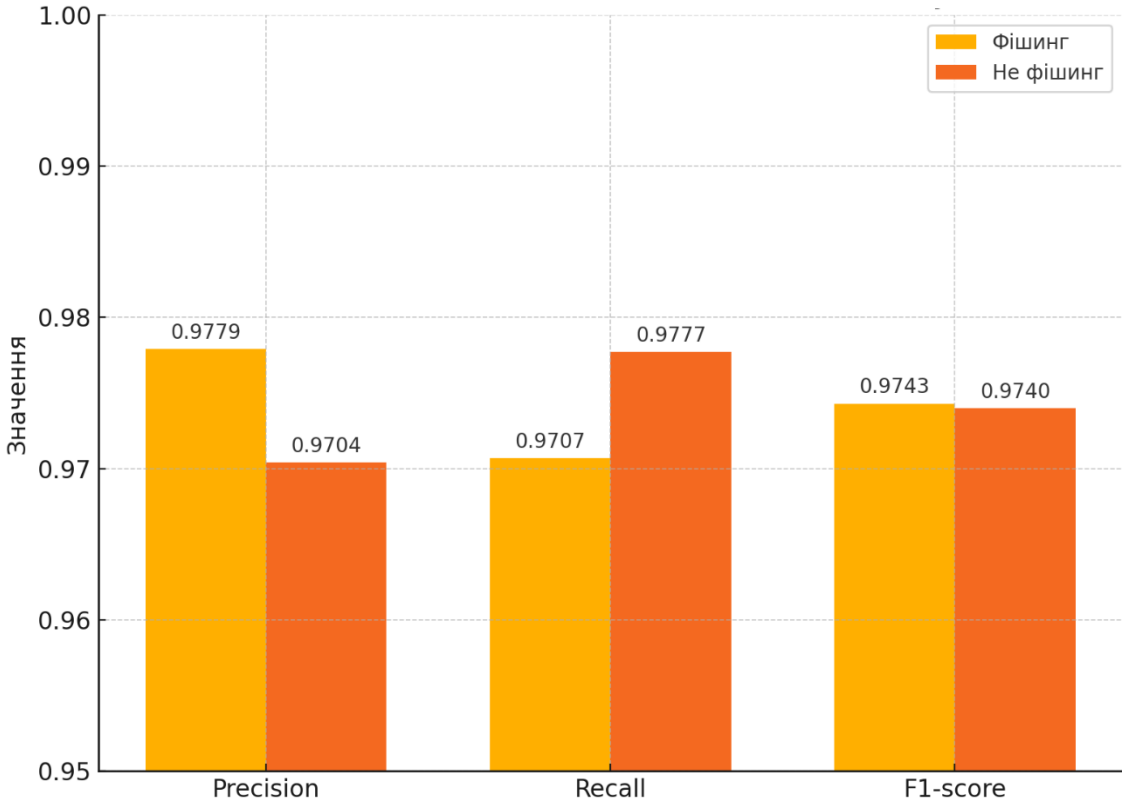

Рисунок 3.24 – Метрики моделі для кожного класу

Для класу «фішинг» спостерігається дещо вищий рівень точності – 0.9779, що означає здатність моделі з високою вірогідністю ідентифікувати дійсно шкідливі домени серед усіх, які вона позначила як фішингові. Значення повноти в межах 0.9707 свідчить про те, що модель коректно виявила більшість фішингових зразків, наявних у тестовому наборі. У свою чергу, клас «не фішинг» продемонстрував трохи нижчу точність – 0.9704, але при цьому має кращу повноту – 0.9777, що означає високу здатність моделі не пропускати легітимні домени.

Загальні значення F1-міри для обох класів залишаються збалансованими (0.9743 для фішингу та 0.9740 для нефішингових об'єктів), що вказує на гармонійне співвідношення між точністю та повнотою. Такий розподіл метрик підтверджує стабільність моделі в обох напрямках класифікації та її практичну придатність для використання в реальних умовах, де однаково важливими є як виявлення загроз, так і мінімізація хибних спрацювань.

Рис. 3.25 висвітлює матрицю помилок, яка відображає результати роботи моделі у задачі бінарної класифікації фішингових доменів. Цей графічний елемент

дозволяє проаналізувати співвідношення між реальними класами та тими, які були передбачені алгоритмом, і дає змогу виявити типові помилки моделі, зокрема хибнопозитивні й хибнонегативні спрацювання.

Рисунок 3.25 – Матриця помилок для задачі виявлення фішингу

У верхньому лівому квадранті розміщено кількість випадків, коли модель правильно класифікувала легітимні домени – таких було 8984. У нижньому правому квадранті відображено кількість коректно виявлених фішингових доменів, що становить 9068. Це означає, що модель точно спрацювала для більшості зразків обох класів.

Натомість у верхньому правому полі зафіксовано 205 помилкових випадків, коли легітимні домени були помилково класифіковані як фішингові. У свою чергу, у нижньому лівому полі відображено 274 зразки фішингових сайтів, які система не розпізнала як загрозу. Незважаючи на наявність деякої кількості хибних класифікацій, їх частка є незначною у порівнянні з кількістю правильних рішень, що підтверджує загальну стабільність і надійність моделі. Отримане співвідношення правильних і помилкових класифікацій є типовим для високоточних моделей, і

демонструє здатність системи ефективно відрізняти фішингові домени від справжніх при мінімальному рівні ризику неправильного спрацювання.

Після побудови та збереження навченої моделі було проведено її перевірку в ручному режимі з метою переконатися в правильності обробки індивідуальних прикладів та здатності системи адекватно реагувати на вхідні дані, задані користувачем.

На рис. 3.26 представлено приклад тестування доменного імені, яке не містить ознак фішингу.

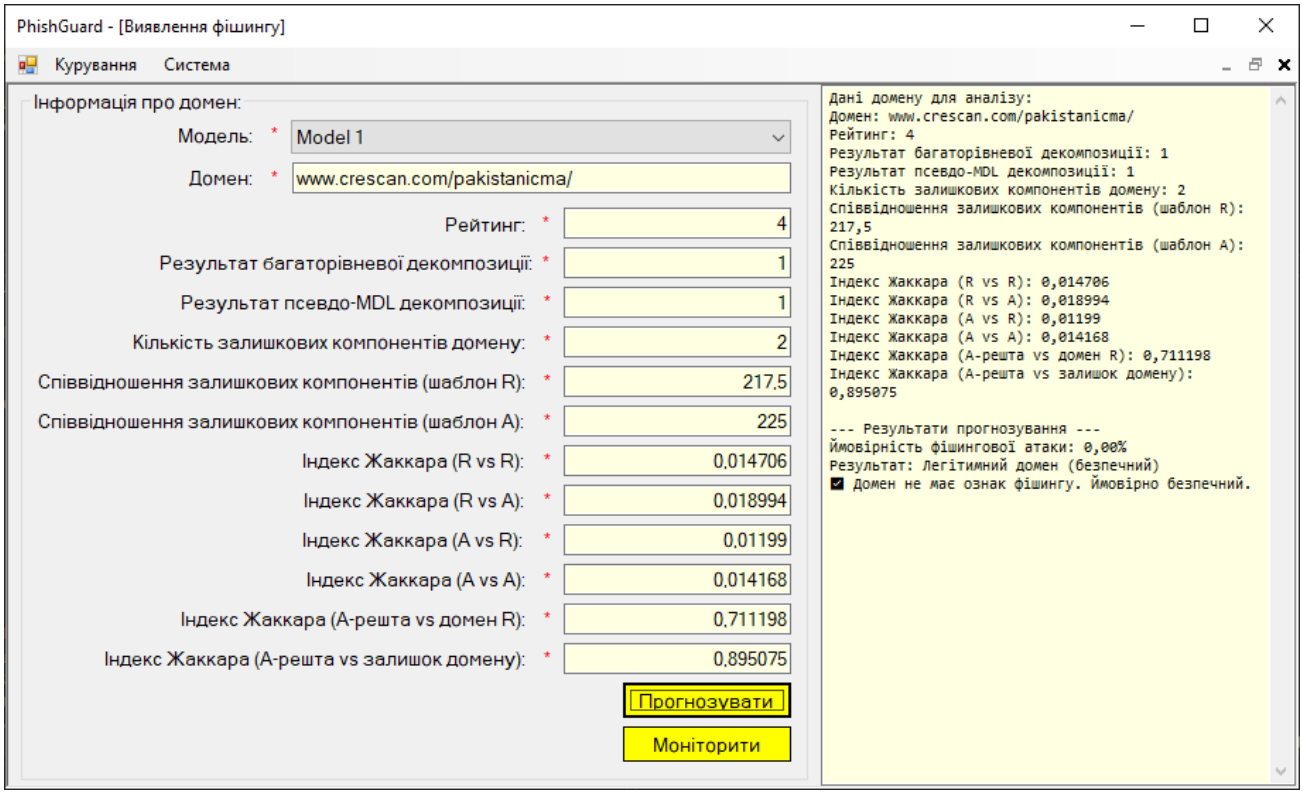

Рисунок 3.26 – Приклад домену без фішингу

Модель оцінила ймовірність належності домену до фішингового класу як 0%, що свідчить про високу впевненість у безпечності ресурсу. Це підтверджується і структурними показниками: більшість коефіцієнтів подібності між шаблоном і аналізованим доменом є дуже низькими, а показники подібності до легітимного класу навпаки – високі. Система надала остаточне текстове пояснення, де зазначено, що виявлених ознак фішингу не зафіксовано і домен класифіковано як безпечний, що підтверджує її правильне функціонування на практиці.

У процесі подальшого функціонального тестування система також була перевірена на здатність виявляти фішингові домени з характерними ознаками маніпуляції. На рис. 3.27 наведено приклад аналізу ресурсу, що за структурними характеристиками викликає підозру на зловмисну активність. Домен має вкрай високий рейтинг – 10 мільйонів, що вказує на його низьку довіру або відсутність у відомих реєстрах. Усі індекси Жаккара дорівнюють нулю, що свідчить про повну відсутність схожості із шаблонами легітимних доменів, натомість коефіцієнт подібності до потенційного фішингу становить 0.6 – значення, характерне для штучно модифікованих назв.

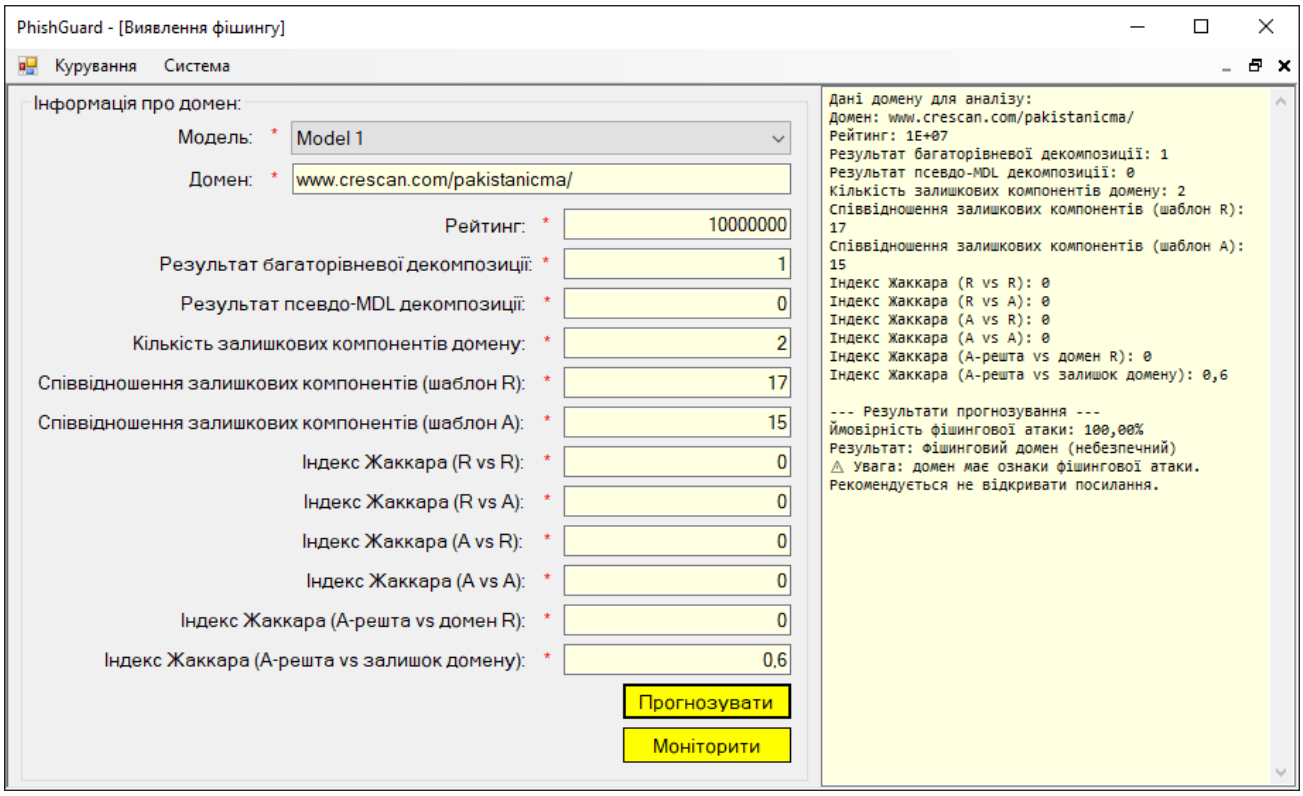

Рисунок 3.27 – Приклад виявлення фішингу

Система класифікувала домен як фішинговий із імовірністю 100%, що супроводжувалося чітким попередженням: «Результат: шкідливий домен (небезпечний)» та рекомендацією не відкривати вказане посилання. Результат свідчить про ефективну роботу моделі в умовах, коли домен має критично високий рівень ризику, зокрема через непрозору структуру і відсутність збігів із легітимними

шаблонами. Це підтверджує здатність системи не лише виявляти класичні шаблони атак, а й фіксувати менш очевидні, але потенційно небезпечні вектори.

3.5 Оцінка результатів проведених експериментів

Після завершення етапу розробки та навчання моделі класифікації було проведено експериментальне оцінювання її здатності до виявлення фішингових атак. З цією метою було здійснено тестування системи на вибірці доменів із різноманітною структурною складністю та різними характеристиками декомпозиції. Для кожного прикладу були зафіксовані результати прогнозування, а також значення ключових структурних параметрів – зокрема кількість залишкових компонентів, співвідношення залишкових елементів та обчислені індекси подібності Жаккара.

Для побудови емпіричної бази було обрано низку доменів, які охоплюють як фішингові, так і легітимні приклади. Основним критерієм для включення у таблицю стали варіативні значення індексу ARrem – одного з найбільш інформативних при класифікації. Окрім того, у табл. 3.1 враховані співвідношення залишкових компонентів та кількість елементів у структурі домену.

Таблиця 3.1

Результати аналізу фішингових та легітимних доменів

№	Ймовірність фішингу, %	Результат	ARrem	Компонентів	Rrem	Arem
1	82,51	Фішинговий	0,782016	2	160,5	166,5
2	0,00	Легітимний	0,895075	2	217,5	225,0
3	95,97	Фішинговий	0,858289	3	115,0	116,7
4	86,04	Фішинговий	0,758288	10	144,0	147,7
5	67,42	Фішинговий	0,810631	9	210,3	213,6
6	0,00	Легітимний	0,5625	2	29,5	20,5

Аналіз виявив декілька важливих закономірностей. По-перше, для більшості фішингових доменів значення індексу ARrem перевищує 0,75, що свідчить про велику схожість між залишковими компонентами після структурування шаблону A та основним тілом домену. Це свідчить про наявність у таких URL типових для фішингових атак патернів – надмірної вкладеності, штучної глибини структури та повторюваних елементів.

По-друге, наявна кореляція між кількістю залишкових компонентів і вірогідністю фішингової атаки. Приклади з 9–10 компонентами демонструють високу ймовірність шахрайської природи. Цей показник прямо вказує на складність URL, яка часто зумовлює приховування справжнього призначення ресурсу.

Серед наведених прикладів наявні і такі, що не підтверджують прямолінійної залежності. Зокрема, другий рядок таблиці демонструє, що навіть при високому значенні ARrem = 0,895 та великих співвідношеннях Rrem/Arem (217,5 / 225) модель впевнено класифікувала домен як легітимний. Це свідчить про важливість сукупного аналізу множини параметрів, а не опори на один-єдиний показник.

Ще одним цікавим аспектом є поведінка моделі щодо структур з малою кількістю компонентів. У шостому прикладі значення ARrem дорівнює 0,5625 при кількості компонентів 2 та дуже низьких співвідношеннях залишкових елементів. Система коректно визначила його як легітимний домен. Це підтверджує, що обмежена структурна складність часто супроводжується відсутністю ознак фішингу.

Таблиця 3.2

Показники подібності Жаккара для тестових прикладів

№	Ймовірність фішингу, %	Результат	RR	RA	AR	AA	ARrd	ARrem
1	82,51	Фішинговий	0	0	0	0	0	0,7820
2	0,00	Легітимний	0,0147	0,0190	0,0120	0,0142	–	0,8951
3	95,97	Фішинговий	0	0	0	0	0	0,8583
4	86,04	Фішинговий	0	0	0	0	0	0,7583
5	67,42	Фішинговий	0	0	0	0	0	0,8106
6	0,00	Легітимний	0	0	0	0	0,3684	0,5625

У більшості фішингових прикладів (рядки 1, 3–5) значення всіх чотирьох базових індексів подібності – RR, RA, AR, AA – дорівнюють нулю. Це вказує на повну відсутність перетину між компонентами шаблону та реальними елементами домену, що свідчить про високий рівень аномальності структури. Така ситуація є типовою для фішингових доменів, які імітують структуру авторитетних ресурсів, використовуючи хибну сегментацію, надлишкові вкладеність та замасковані підрядки.

Натомість легітимні приклади (рядки 2 і 6) мають ненульові значення принаймні частини індексів. У другому прикладі значення RA, AR, AA та RR варіюються в межах 0,01–0,02, що вказує на мінімальний, але наявний ступінь перетину між шаблонами та реальним доменом. Незважаючи на їхню невисоку абсолютну величину, вони демонструють, що структура URL хоча б частково відповідає очікуваній нормі, що й підтверджується класифікацією як легітимного.

Особливої уваги заслуговує приклад із шостого рядка, де основні індекси залишаються нульовими, але ARrd набуває значення 0,3684. Це свідчить про специфічну локальну відповідність між залишковими компонентами шаблону A та частиною домену, що дозволило моделі ідентифікувати домен як безпечний. Індекс ARrd, таким чином, виконує функцію компенсаторного механізму за відсутності глобальної структурної схожості.

Загалом, індекси типу RR, RA, AR та AA забезпечують загальну кількісну оцінку перетину компонентів, тоді як ARrem та ARrd відіграють більш спеціалізовану роль, дозволяючи ідентифікувати локальні відповідності або залишкові фрагменти, характерні для фішингової структури. У цьому контексті значення ARrem > 0,75 у поєднанні з нульовими RR, RA, AR, AA надійно сигналізує про ймовірну шахрайську активність.

Висновки до розділу 3

У рамках даного розділу було здійснено повне проєктування та реалізацію інформаційної системи для аналізу атак соціальної інженерії з фокусом на виявлення фішингових доменів. Система побудована на основі трьохрівневої архітектури, що

забезпечує логічне розділення відповідальностей між рівнями доступу до даних, бізнес-логіки та користувацького інтерфейсу. Для кожного рівня було розроблено відповідні класи, а їх структура відображена на відповідних діаграмах, що гарантує модульність, масштабованість і підтримуваність системи.

Було реалізовано механізми побудови класифікаційної моделі на основі ознак доменів, зокрема використано перетворення вхідних даних, формування векторів ознак та інтеграцію моделі ШД, яка показала найкраще співвідношення точності та швидкодії. У компонент виявлення фішингових атак інтегровано механізм ручного тестування з відображенням ймовірності ризику, а також реалізовано модулі захисту на основі алгоритму AES, що забезпечують шифрування та розшифрування конфіденційних даних системи.

Результати тестування підтвердили високу якість моделі: точність класифікації становить 97,42%, F1-міра – 97,43%, площа під ROC-кривою – 0,9971. Матриця помилок засвідчила переважання правильно класифікованих доменів, з незначною кількістю хибнопозитивних і хибнонегативних рішень. Окрім того, було проведено аналіз класифікаційних метрик окремо для фішингового та нефішингового класів, де обидва класи продемонстрували збалансовану ефективність. Візуалізовані приклади прогнозування показали коректність роботи системи як у випадках безпечних ресурсів, так і при виявленні небезпечних доменів.

Сформовані таблиці з експериментальними даними дозволили деталізувати вплив окремих ознак на результат класифікації та обґрунтувати надійність моделі. Отримані результати створюють передумови для переходу до наступного етапу – розгортання та повноцінного впровадження системи в рамках середовища безпеки з метою автоматизованого моніторингу загроз, пов’язаних із атаками соціальної інженерії.

ВИСНОВКИ

У результаті проведеного дослідження було реалізовано повнофункціональну інтелектуальну систему аналізу фішингових атак, здатну автоматично виявляти потенційно небезпечні домени на основі структурних ознак. Основною розв’язаною інженерно-технічною задачею стало поєднання методів машинного навчання з доменним аналізом і засобами криптографічного захисту в єдиній архітектурі.

Аналіз сучасного стану проблеми соціальної інженерії дозволив класифікувати типи атак та оцінити ефективність методів протидії. Було обґрунтовано доцільність використання машинного навчання як інструмента виявлення фішингу, що не залежить від фіксованих списків і здатен виявляти нові загрози.

У рамках аналітичного етапу було проаналізовано датасет Phishing Website Detection Dataset, здійснено попередню обробку даних, виявлено найбільш інформативні ознаки, побудовано теплові карти кореляцій та виконано кластеризацію, що дозволило підтвердити релевантність ознак для класифікації фішингових атак.

Зіставлення алгоритмів машинного навчання (швидке дерево, випадковий ліс, градієнтний бустинг) засвідчило перевагу методу швидкого дерева, який було обрано для реалізації системи завдяки його швидкодії, стабільній продуктивності та низьким вимогам до обчислювальних ресурсів.

Реалізована трьохрівнева архітектура системи забезпечила чітке розділення між рівнями логіки, даних і інтерфейсу. Розроблено та описано відповідні класи для кожного рівня, що забезпечує гнучкість і масштабованість системи, а також полегшує підтримку й модифікацію.

Механізм виявлення фішингу включає повний цикл: завантаження даних, тренування моделі, збереження, завантаження та прогнозування. Інтерфейс системи дозволяє як автоматичне, так і ручне тестування, а також виведення інтерпретованого звіту.

Для підвищення безпеки додано механізми шифрування й розшифрування конфіденційних даних користувачів і моделей за допомогою алгоритму AES, що гарантує збереження цілісності та конфіденційності даних.

Результати тестування свідчать про високу якість моделі: Accuracy = 0,9742, F1 Score = 0,9743, AUC = 0,9971. Матриця помилок показала низький рівень хибнокласифікацій: TP = 9068, TN = 8984, FP = 205, FN = 274, а метрики для обох класів вказують на збалансовану точність і повноту.

Проведені експерименти із реальними прикладами підтвердили практичну ефективність системи: у всіх протестованих випадках модель правильно класифікувала домени як фішингові або безпечні, що підтверджено структурними показниками (ARrem, індекси Жаккара, кількість компонентів тощо).

Отримані результати є достатньо надійними для практичного впровадження системи в інформаційні середовища, які потребують виявлення загроз соціальної інженерії. Рекомендовано розгорнути систему у вигляді модуля безпеки з можливістю інтеграції з браузером або поштовим сервером, а також розширити функціональність за рахунок виявлення інших типів атак на основі поведенкових патернів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hadnagy, C., & Schulman, S. (2021). Human hacking: Win friends, influence people, and leave them better off for having met you. HarperCollins.

2. Соколов, В. Ю., & Курбанмурадов, Д. М. (2018). Методика протидії соціальному інжинірингу на об’єктах інформаційної діяльності. Науково-технічний журнал" Кібербезпека: освіта, наука, техніка", (1), 6-16.

3. Gupta, S., Pritwani, M., Shrivastava, A., Moharir, M., & AR, A. K. (2024, August). A Comprehensive Analysis of Social Engineering Attacks: From Phishing to Prevention-Tools, Techniques and Strategies. In 2024 Second International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI) (pp. 1-8). IEEE.

4. Carroll, F., Adejobi, J. A., & Montasari, R. (2022). How good are we at detecting a phishing attack? Investigating the evolving phishing attack email and why it continues to successfully deceive society. SN Computer science, 3(2), 170.

5. Naqvi, B., Perova, K., Farooq, A., Makhdoom, I., Oyedeji, S., & Porras, J. (2023). Mitigation strategies against the phishing attacks: A systematic literature review. Computers & Security, 132, 103387.

6. Samad, D., & Gani, G. A. (2020). Analyzing and predicting spear-phishing using machine learning methods. Multidiszciplináris tudományok, 10(4), 262-273.

7. Parmar, B. (2012). Protecting against spear-phishing. Computer Fraud & Security, 2012(1), 8-11.

8. On the Feasibility of Fully AI-automated Vishing Attacks. URL: https://arxiv.org/pdf/2409.13793 (дата звернення 21.03.2025).

9. What is a Vishing Attack? Definition, Examples & Prevention. URL: https://www.wallarm.com/what/vishing-attack (дата звернення 21.03.2025).

10. Akande, O. N., Gbenle, O., Abikoye, O. C., Jimoh, R. G., Akande, H. B., Balogun, A. O., & Fatokun, A. (2023). SMSPROTECT: An automatic smishing detection mobile application. ICT Express, 9(2), 168-176.

11. Schematic Sequence of Smishing Attack. URL: https://www.researchgate.net/figure/Schematic-Sequence-of-Smishing-Attack_fig3_388423758 (дата звернення 21.03.2025).

12. Mughaid, A., AlZu'bi, S., Hnaif, A., Taamneh, S., Alnajjar, A., & Elsoud, E. A. (2022). An intelligent cyber security phishing detection system using deep learning techniques. Cluster Computing, 25(6), 3819-3828.

13. Gangavarapu, T., Jaidhar, C. D., & Chanduka, B. (2020). Applicability of machine learning in spam and phishing email filtering: review and approaches. Artificial Intelligence Review, 53(7), 5019-5081.

14. Nautsch, A., Wang, X., Evans, N., Kinnunen, T. H., Vestman, V., Todisco, M., ... & Lee, K. A. (2021). ASVspoof 2019: spoofing countermeasures for the detection of synthesized, converted and replayed speech. IEEE Transactions on Biometrics, Behavior, and Identity Science, 3(2), 252-265.

15. Chy, M. K. H., & Buadi, O. N. (2024). A Machine Learning Driven Website Platform and Browser Extension for Real-time Scoring and Fraud Detection for Website Legitimacy Verification and Consumer Protection. arXiv preprint arXiv:2411.00368.

16. Mir, A. W., & Ramachandran, R. K. (2021). Implementation of security orchestration, automation and response (SOAR) in smart grid-based SCADA systems. In Sixth International Conference on Intelligent Computing and Applications: Proceedings of ICICA 2020 (pp. 157-169). Springer Singapore.

17. Li, H., Yoo, S., & Kettinger, W. J. (2021). The roles of IT strategies and security investments in reducing organizational security breaches. Journal of Management Information Systems, 38(1), 222-245.

18. Hossain M., Raza M. Exploring The Effectiveness Of Multifactor Authentication In Preventing Unauthorized Access To Online Banking Systems. Multidisciplinary Science Journal. 2023. Vol. 1, No. 1, pp. 8-12.

19. Lund, B. D., Lee, T. H., Wang, Z., Wang, T., & Mannuru, N. R. (2024). Zero Trust Cybersecurity: Procedures and Considerations in Context. Encyclopedia, 4(4), 1520-1533.

20. Ilori, O., Nwosu, N. T., & Naiho, H. N. N. (2024). Third-party vendor risks in IT security: A comprehensive audit review and mitigation strategies. World Journal of Advanced Research and Reviews, 22(3), 213-224.

21. Galinec, D., & Luić, L. (2020). Design of conceptual model for raising awareness of digital threats. WSEAS transactions on environment and development, 16, 493-504.

22. Scherb, C., Heitz, L. B., Grimberg, F., Grieder, H., & Maurer, M. (2023). A cyber attack simulation for teaching cybersecurity. EPiC Series in Computing, 93, 129-140.

23. Wang, Z., Zhu, H., & Sun, L. (2021). Social engineering in cybersecurity: Effect mechanisms, human vulnerabilities and attack methods. Ieee Access, 9, 11895-11910.

24. Phishing Website Detection Dataset. URL: https://www.kaggle.com/datasets/hasibur013/url-data-for-phishing-website-detection (дата звернення 21.03.2025).

25. Piñeiro, C., Abuín, J. M., & Pichel, J. C. (2020). Very Fast Tree: speeding up the estimation of phylogenies for large alignments through parallelization and vectorization strategies. Bioinformatics, 36(17), 4658-4659.

26. Kramer, A. M., Thornlow, B., Ye, C., De Maio, N., McBroome, J., Hinrichs, A. S., ... & Corbett-Detig, R. (2023). Online phylogenetics with matOptimize produces equivalent trees and is dramatically more efficient for large SARS-CoV-2 phylogenies than de novo and maximum-likelihood implementations. Systematic Biology, 72(5), 1039-1051.

27. Schonlau, M., & Zou, R. Y. (2020). The random forest algorithm for statistical learning. The Stata Journal, 20(1), 3-29.

28. Khammas, B. M. (2020). Ransomware detection using random forest technique. ICT Express, 6(4), 325-331.

29. Sahin, E. K. (2022). Comparative analysis of gradient boosting algorithms for landslide susceptibility mapping. Geocarto International, 37(9), 2441-2465.

30. Huang, M., Zhang, X. S., Bhatti, U. A., Wu, Y., Zhang, Y., & Ghadi, Y. Y. (2024). An interpretable approach using hybrid graph networks and explainable AI for intelligent diagnosis recommendations in chronic disease care. Biomedical Signal Processing and Control, 91, 105913

ДОДАТОК А

Лістинг 1. Код класу «ModelsForm»

```
using SocialEnginDetectionApp.AppCode;
using SocialEnginDetectionApp.Forms.Systems;
using SocialEnginDetectionApp.Providers;
using Microsoft.ML;
using Microsoft.ML.Data;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Globalization;
using System.IO;
using System.Diagnostics;

namespace SocialEnginDetectionApp.Forms.SysMS {
  public partial class ModelsForm : Form {
    private MLContext mlContext;
    ITransformer trainedModel;
    private IDataView dataView;
    private string _Path = "";

    private int _selectedRowIndex = 0;
    private ValidationMy _Validation = new ValidationMy();
    private ModelsProvider _ModelsProvider = new ModelsProvider();
    private List<Models> _ModelsList = new List<Models>();
    private LogsProvider _LogsProvider = new LogsProvider();
    private bool _IsModelTrain = false;
    private Random _rand = new Random();

    public ModelsForm() {
      InitializeComponent();
      DataLoad();
    }

    private void OpenBtn_Click(object sender, EventArgs e) {
      OpenFileDialog openFileDialog = new OpenFileDialog {
        Filter = "Text files (*.csv)|*.csv|All files (*.*)|*.*",
        FilterIndex = 2,
        RestoreDirectory = true
      };

      if (openFileDialog.ShowDialog() == DialogResult.OK) {
```

```
        _Path = openFileDialog.FileName;
        FileNameTBox.Text = openFileDialog.FileName;


        mlContext = new MLContext(seed: 1);


        dataView = mlContext.Data.LoadFromTextFile<ModelInput>(
            path: _Path,
            hasHeader: true,
            separatorChar: ',',
            allowQuoting: true);

        var enumerator = mlContext.Data.CreateEnumerable<ModelInput>(dataView, reuseRowObject:
false);
        int count = 0;

        foreach (var row in enumerator) {
          count++;
        }

        RaportTBox.Text = ($"Успішно прочитано {count} рядків.\r\n");
        Application.DoEvents();


        var split = mlContext.Data.TrainTestSplit(dataView, testFraction: 0.2);


        var dataProcessPipeline = mlContext.Transforms.Text.FeaturizeText(
                                  outputColumnName: "DomainFeaturized",
                                  inputColumnName: nameof(ModelInput.Domain))
                                 .Append(mlContext.Transforms.Concatenate(
                                  "Features",
                                  "DomainFeaturized",
                                  nameof(ModelInput.Ranking),
                                  nameof(ModelInput.Mld_res),
                                  nameof(ModelInput.Mld_ps_res),
                                  nameof(ModelInput.Card_rem),
                                  nameof(ModelInput.Ratio_Rrem),
                                  nameof(ModelInput.Ratio_Arem),
                                  nameof(ModelInput.Jaccard_RR),
                                  nameof(ModelInput.Jaccard_RA),
                                  nameof(ModelInput.Jaccard_AR),
                                  nameof(ModelInput.Jaccard_AA),
                                  nameof(ModelInput.Jaccard_ARrd),
                                  nameof(ModelInput.Jaccard_ARrem)));

        var trainer = mlContext.BinaryClassification.Trainers
            .FastTree(labelColumnName: "Label", featureColumnName: "Features");
        var trainingPipeline = dataProcessPipeline.Append(trainer);
```

```
RaportTBox.Text += ("Навчання моделі...\r\n");
var startTime = DateTime.Now;
trainedModel = trainingPipeline.Fit(split.TrainSet);
var duration = DateTime.Now - startTime;
RaportTBox.Text +=($"Навчання завершено за {duration.TotalSeconds:F2} с.\r\n\r\n");

var testSetTransform = trainedModel.Transform(split.TestSet);

var metrics = mlContext.BinaryClassification.Evaluate(
   data: testSetTransform,
   labelColumnName: "Label",
   scoreColumnName: "Score");



RaportTBox.Text +=("\n===== ГЛОБАЛЬНІ МЕТРИКИ (усі класи разом) =====\r\n");
RaportTBox.Text +=($"Accuracy:  {metrics.Accuracy:F4}\r\n");
RaportTBox.Text += ($"F1 Score:  {metrics.F1Score:F4}\r\n");
RaportTBox.Text+=($"AUC:       {metrics.AreaUnderRocCurve:F4}\r\n");
RaportTBox.Text+= ($"Precision (Overall): {metrics.PositivePrecision:F4}\r\n");
RaportTBox.Text +=($"Recall    (Overall): {metrics.PositiveRecall:F4} \r\n\r\n");


bool[] predicted = testSetTransform.GetColumn<bool>("PredictedLabel").ToArray();
bool[] actual = testSetTransform.GetColumn<bool>("Label").ToArray();

int tp = 0, fp = 0, tn = 0, fn = 0;
for (int i = 0; i < actual.Length; i++) {
  if (actual[i] == true && predicted[i] == true) tp++;
  else if (actual[i] == false && predicted[i] == false) tn++;
  else if (actual[i] == false && predicted[i] == true) fp++;
  else if (actual[i] == true && predicted[i] == false) fn++;
}

RaportTBox.Text += ("\n===== МАТРИЦЯ ПОМИЛОК =====\r\n");
RaportTBox.Text += ("      Predicted: Не фішинг       |  Predicted: Фішинг\r\n");
RaportTBox.Text += ($"Actual: не фішинг    TN = {tn,4}   |    FP = {fp,4}\r\n");
RaportTBox.Text += ($"Actual: фішинг       FN = {fn,4}   |    TP = {tp,4}\r\n\r\n");


float fishPrecision = (tp + fp) == 0 ? 0 : (float)tp / (tp + fp);
float fishRecall = (tp + fn) == 0 ? 0 : (float)tp / (tp + fn);
float fishF1 = (fishPrecision + fishRecall) == 0
   ? 0
   : 2f * (fishPrecision * fishRecall) / (fishPrecision + fishRecall);

float noFishPrecision = (tn + fn) == 0 ? 0 : (float)tn / (tn + fn);
float noFishRecall = (tn + fp) == 0 ? 0 : (float)tn / (tn + fp);
float noFishF1 = (noFishPrecision + noFishRecall) == 0
   ? 0
   : 2f * (noFishPrecision * noFishRecall) / (noFishPrecision + noFishRecall);
```

```
      RaportTBox.Text += ("\n===== МЕТРИКИ ДЛЯ КОЖНОГО КЛАСУ =====\r\n");
      RaportTBox.Text += ($"Фішинг (Label=true) -> TP={tp}, FP={fp}, FN={fn}, TN={tn}\r\n");
      RaportTBox.Text += ($"  Precision(fish) = {fishPrecision:F4}\r\n");
      RaportTBox.Text += ($"  Recall(fish)    = {fishRecall:F4}\r\n");
      RaportTBox.Text += ($"  F1(fish)        = {fishF1:F4}\r\n\r\n");

      RaportTBox.Text += ($"Не фішинг  (Label=false) -> TN={tn}, FP={fp}, FN={fn}, TP={tp}\r\n");
      RaportTBox.Text += ($"  Precision(noFish) = {noFishPrecision:F4}\r\n");
      RaportTBox.Text += ($"  Recall(noFish)    = {noFishRecall:F4}\r\n");
      RaportTBox.Text += ($"  F1(noFish)        = {noFishF1:F4}\r\n\r\n");

      RaportTBox.SelectionStart = RaportTBox.Text.Length;
      RaportTBox.ScrollToCaret();

      _IsModelTrain = true;
    }
  }



  private void ModelsGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
    if (e.ColumnIndex == 5 && ModelsGridView[0, e.RowIndex].Value.ToString() !=
_ModelsList[0].Message) {
      if (MessageBox.Show("Ви дійсно хочете видалити цю модель?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
        _ModelsProvider.DeleteModelsByModelsId(Convert.ToInt32(ModelsGridView[0,
e.RowIndex].Value.ToString()));
        DataLoad();
      }
    }
  }

  private void SaveBtn_Click(object sender, EventArgs e) {
    if (IsDataEnteringCorrect()) {
      string pathName = @"\teach\" + GenerateFileName() + ".zip";
      string localProj =

System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location);
      _ModelsProvider.InsertModels(ModelsNamesTBox.Text, pathName);
      mlContext.Model.Save(trainedModel, dataView.Schema, localProj + pathName);
      ClearAllData();
      _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId,
        "Було навчено модель " +
        ModelsNamesTBox.Text, DateTime.Now);
      MessageBox.Show("Дані успішно збережено!");
    }
  }


  private void ClearBtn_Click(object sender, EventArgs e) {
    ClearAllData();
```

```
}

private void ExitBtn_Click(object sender, EventArgs e) {
  this.Close();
}


public string GenerateFileName() {
  DateTime now = DateTime.Now;
  string fileName = string.Format("{0}_{1}_{2}_{3}_{4}_{5}",
      now.Year, now.Month, now.Day, now.Hour, now.Minute, now.Second);

  return fileName;
}

private void ClearAllData() {
  _IsModelTrain = false;
  ModelsNamesTBox.Text = String.Empty;
  RaportTBox.Text = String.Empty;
  DataLoad();
}
private bool IsDataEnteringCorrect() {
  bool isCorrect = true;
  if (!_IsModelTrain) {
    MessageBox.Show("Неможливо зберегти дані. \r\nЩе не навчено модель!", "Увага!");
    isCorrect = false;
  }
  if (_Validation.IsDataEntering(ModelsNamesTBox.Text)) {
    ModelsNamesValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    ModelsNamesValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  return isCorrect;
}

private void DataLoad() {
  int firstRowIndex = 0;
  if (ModelsGridView.FirstDisplayedScrollingRowIndex > 0) {
    firstRowIndex = ModelsGridView.FirstDisplayedScrollingRowIndex;
  }
  try {
    _ModelsList = _ModelsProvider.GetAllModels();
    LoadDataInModelsGridView(_ModelsList);
    if (_selectedRowIndex == ModelsGridView.Rows.Count) {
      _selectedRowIndex = ModelsGridView.Rows.Count - 1;
    }
    if (_selectedRowIndex >= 0) {
      ModelsGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
      ModelsGridView.Rows[_selectedRowIndex].Selected = true;
    }
  } catch (Exception ex) {
```

```
      MessageBox.Show(ex.ToString());
     }
    }
  }
}
```

Лістинг 2. Код класу «DetectionForm»

```
using Microsoft.ML;
using SocialEnginDetectionApp.AppCode;
using SocialEnginDetectionApp.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SocialEnginDetectionApp.Forms.Controls {
  public partial class DetectionForm : Form {
    private ValidationMy _Validation = new ValidationMy();
    private Models _SelectedModels = new Models();
    private List<Models> _ModelsList = new List<Models>();
    private MLContext context = new MLContext();
    private ModelsProvider _ModelsProvider = new ModelsProvider();
    private List<ModelInput> testDataList = new List<ModelInput>();
    private Random rnd = new Random();

    private ITransformer model;
    private bool _IsScenariosLoad = false;

    public DetectionForm() {
      InitializeComponent();
      LoadAllDate();
    }

    private void LoadAllDate() {
      _ModelsList = _ModelsProvider.GetAllModels();
      ModelsCBox.DataSource = _ModelsList;
      ModelsCBox.ValueMember = "ModelsId";
      ModelsCBox.DisplayMember = "ModelsName";
      _IsScenariosLoad = true;
      ModelsCBox_SelectedValueChanged(ModelsCBox, EventArgs.Empty);
      LoadTestData();
    }

    private void ModelsCBox_SelectedValueChanged(object sender, EventArgs e) {
      if (_IsScenariosLoad) {
        _SelectedModels = _ModelsProvider.SelectedModelsByModelsId(
```

```
      Convert.ToInt32(ModelsCBox.SelectedValue));
    LoadModel(_SelectedModels.ModelsFileModel);
  }
}

private void LoadModel(string FilePath) {
  string localProj = Application.StartupPath + FilePath;
  var mlContext = new MLContext();
  DataViewSchema modelSchema;
  model = context.Model.Load(localProj, out modelSchema);
}

private void LoadTestData() {
  string path = Path.Combine(Environment.CurrentDirectory, "test_data.csv");

  var mlContext = new MLContext();
  var dataView = mlContext.Data.LoadFromTextFile<ModelInput>(
      path,
      hasHeader: true,
      separatorChar: ',');

  testDataList = mlContext.Data
      .CreateEnumerable<ModelInput>(dataView, reuseRowObject: false)
      .ToList();
}


private void timer1_Tick(object sender, EventArgs e) {
  if (testDataList.Count == 0) {
    ReportTBox.Text = "❗ Дані не завантажено або список порожній.";
    timer1.Enabled = false;
    GenBtn.Text = "Моніторити";
    return;
  }

  var input = testDataList[rnd.Next(testDataList.Count)];

  var report = new StringBuilder();
  report.AppendLine("🎯 Випадковий приклад з тестового набору:");
  report.AppendLine($"Домен: {input.Domain}");
  report.AppendLine($"Рейтинг: {input.Ranking}");
  report.AppendLine($"MLD результат: {input.Mld_res}");
  report.AppendLine($"MLD.ps результат: {input.Mld_ps_res}");
  report.AppendLine($"Кількість залишкових компонентів: {input.Card_rem}");
  report.AppendLine($"Співвідношення Rrem: {input.Ratio_Rrem}");
  report.AppendLine($"Співвідношення Arem: {input.Ratio_Arem}");
  report.AppendLine($"Індекс Жаккара (RR): {input.Jaccard_RR}");
  report.AppendLine($"Індекс Жаккара (RA): {input.Jaccard_RA}");
  report.AppendLine($"Індекс Жаккара (AR): {input.Jaccard_AR}");
  report.AppendLine($"Індекс Жаккара (AA): {input.Jaccard_AA}");
  report.AppendLine($"Індекс Жаккара (ARrd): {input.Jaccard_ARrd}");
  report.AppendLine($"Індекс Жаккара (ARrem): {input.Jaccard_ARrem}");
```

```
    var inputArray = new[] { input };
    var testView = context.Data.LoadFromEnumerable(inputArray);
    var predictions = model.Transform(testView);
    var result = context.Data.CreateEnumerable<ModelOutput>(predictions, reuseRowObject:
false).Single();

    report.AppendLine("\n🔎 Результат прогнозування:");
    report.AppendLine($"Ймовірність фішингової атаки: {Math.Max(0, Math.Min(1, result.Score)) *
100:F2}%");
    report.AppendLine($"Результат: {(result.PredictedLabel ? "⚠ Фішинговий домен" : "✅
Легітимний домен")}");

    ReportTBox.Text = report.ToString();
  }


  private void PredictBtn_Click(object sender, EventArgs e) {
    if (IsAllInputValid())
    {
      var input = new ModelInput {
        Domain = DomainTBox.Text,
        Ranking = (float)Convert.ToDouble(RankingTBox.Text),
        Mld_res = (float)Convert.ToDouble(MldResTBox.Text),
        Mld_ps_res = (float)Convert.ToDouble(MldPsResTBox.Text),
        Card_rem = (float)Convert.ToDouble(CardRemTBox.Text),
        Ratio_Rrem = (float)Convert.ToDouble(RatioRRemTBox.Text),
        Ratio_Arem = (float)Convert.ToDouble(RatioARemTBox.Text),
        Jaccard_RR = (float)Convert.ToDouble(JaccardRRTBox.Text),
        Jaccard_RA = (float)Convert.ToDouble(JaccardRATBox.Text),
        Jaccard_AR = (float)Convert.ToDouble(JaccardARTBox.Text),
        Jaccard_AA = (float)Convert.ToDouble(JaccardAATBox.Text),
        Jaccard_ARrd = (float)Convert.ToDouble(JaccardARrdTBox.Text),
        Jaccard_ARrem = (float)Convert.ToDouble(JaccardARremTBox.Text)
      };

      var inputData = new[] { input };
      var testData = context.Data.LoadFromEnumerable(inputData);
      var predictions = model.Transform(testData);

      var predictionResult = context.Data
         .CreateEnumerable<ModelOutput>(predictions, reuseRowObject: false)
         .Single();

      var report = new StringBuilder();
      report.AppendLine("Дані домену для аналізу:");
      report.AppendLine($"Домен: {input.Domain}");
      report.AppendLine($"Рейтинг: {input.Ranking}");
      report.AppendLine($"Результат багаторівневої декомпозиції: {input.Mld_res}");
      report.AppendLine($"Результат псевдо-MDL декомпозиції: {input.Mld_ps_res}");
      report.AppendLine($"Кількість залишкових компонентів домену: {input.Card_rem}");
```

```
        report.AppendLine($"Співвідношення залишкових компонентів (шаблон R):
{input.Ratio_Rrem}");
        report.AppendLine($"Співвідношення залишкових компонентів (шаблон A):
{input.Ratio_Arem}");
        report.AppendLine($"Індекс Жаккара (R vs R): {input.Jaccard_RR}");
        report.AppendLine($"Індекс Жаккара (R vs A): {input.Jaccard_RA}");
        report.AppendLine($"Індекс Жаккара (A vs R): {input.Jaccard_AR}");
        report.AppendLine($"Індекс Жаккара (A vs A): {input.Jaccard_AA}");
        report.AppendLine($"Індекс Жаккара (A-решта vs домен R): {input.Jaccard_ARrd}");
        report.AppendLine($"Індекс Жаккара (A-решта vs залишок домену): {input.Jaccard_ARrem}");

        report.AppendLine("\r\n--- Результати прогнозування ---");
        report.AppendLine($"Ймовірність фішингової атаки: {Math.Max(0, Math.Min(1,
predictionResult.Score)) * 100:F2}%");
        report.AppendLine($"Результат: {(predictionResult.PredictedLabel ? "Фішинговий домен
(небезпечний)" : "Легітимний домен (безпечний)")}");

        if (predictionResult.PredictedLabel) {
          report.AppendLine("⚠ Увага: домен має ознаки фішингової атаки. Рекомендується не
відкривати посилання.");
        } else {
          report.AppendLine("✅ Домен не має ознак фішингу. Ймовірно безпечний.");
        }
        ReportTBox.Text = report.ToString();
      }
    }

    private void GenBtn_Click(object sender, EventArgs e) {
      if (IsSelectedModelCorrect()) {
        if (testDataList.Count == 0) {
          LoadTestData();
        }

        if (timer1.Enabled) {
          timer1.Enabled = false;
          GenBtn.Text = "Моніторити";
        } else {
          timer1.Enabled = true;
          GenBtn.Text = "Зупинити";
        }
      }
    }


    private bool IsSelectedModelCorrect() {
      bool isCorrect = true;
      if (Convert.ToInt32(ModelsCBox.SelectedValue) > 0) {
        ScenariosValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        ScenariosValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
```

```
  return isCorrect;
}


private bool IsAllInputValid() {
  bool isCorrect = true;
  if (_Validation.IsDataEntering(DomainTBox.Text)) {
    DomainValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    DomainValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_Validation.IsDataConvertToDouble(RankingTBox.Text)) {
    RankingValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    RankingValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_Validation.IsDataConvertToDouble(MldResTBox.Text)) {
    MldResValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    MldResValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_Validation.IsDataConvertToDouble(MldPsResTBox.Text)) {
    MldPsResValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    MldPsResValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_Validation.IsDataConvertToDouble(CardRemTBox.Text)) {
    CardRemValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    CardRemValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_Validation.IsDataConvertToDouble(RatioRRemTBox.Text)) {
    RatioRRemValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    RatioRRemValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_Validation.IsDataConvertToDouble(RatioARemTBox.Text)) {
    RatioARemValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    RatioARemValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_Validation.IsDataConvertToDouble(JaccardRRTBox.Text)) {
    JaccardRRValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    JaccardRRValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
```

```
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToDouble(JaccardRATBox.Text)) {
        JaccardRATValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        JaccardRATValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToDouble(JaccardARTBox.Text)) {
        JaccardARValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        JaccardARValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToDouble(JaccardAATBox.Text)) {
        JaccardAAValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        JaccardAAValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToDouble(JaccardARrdTBox.Text)) {
        JaccardARrdValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        JaccardARrdValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToDouble(JaccardARremTBox.Text)) {
        JaccardARremValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        JaccardARremValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }


  }
}
```