МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

КУЛІКОВА МАРИНА ЮРІЇВНА

Допускається до захисту:
Завідувач кафедри прикладної математики та кібербезпеки, д-р філософії з математики
_____ Луценко А. В.
«__»_____20__р.

РОЗРОБКА СИСТЕМИ ЗАПОБІГАННЯ ЦІЛЬОВОМУ ФІШИНГУ

Спеціальність 125 Кібербезпека
Кваліфікаційна (бакалаврська) робота

Керівник:
Крижановський В. Г.,
професор кафедри прикладної математики та кібербезпеки

_____
(підпис)

Оцінка : _____ / _____ / _____
(бали/за шкало ЄКТС/за національною шкалою)
Голова ЕК: _____ (підпис)

Вінниця – 2025

АНОТАЦІЯ

Кулікова М.Ю. Розробка системи запобігання цільовому фішингу. Спеціальність 125 Кібербезпека. Донецький національний університет імені Василя Стуса. Вінниця. 2025 рік.

У бакалаврській роботі розглядається питання цільового фішингу, що виступає однією з найбільш загрозливих різновидів кібератак. Метою роботи є створення інтелектуальної системи, здатної виявляти такі атаки.

Ключові слова: фішинг, цільовий фішинг, штучний інтелект, метод рою часток, кібербезпека.

Табл. 4. Рис. 12. Бібліограф.: 33 найм.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. У сучасному цифровому середовищі фішингові атаки залишаються однією з найпоширеніших і водночас найнебезпечніших кіберзагроз. Особливу небезпеку становить цільовий фішинг, який спрямований на конкретних осіб або організацій та використовує індивідуально адаптовані повідомлення для досягнення злочинних цілей. Існуючі системи виявлення фішингу демонструють обмежену ефективність проти таких складних атак, оскільки фішингові листи часто мають реалістичний вигляд, уникаючи традиційних фільтрів.

Незважаючи на наявність рішень, які поєднують методи перевірки автентичності електронних повідомлень та елементи машинного навчання, проблема виявлення саме цільового фішингу залишається недостатньо вивченою. Зокрема, актуальним є питання підвищення точності класифікацій таких електронних листів шляхом використання гібридних моделей і біологічно інспірованих алгоритмів, які ще не знайшли активного застосування в реальних програмних рішеннях. Таким чином, створення системи поєднання класичних підходів і сучасних методів штучного інтелекту з параметричною оптимізацією має велике теоретичне і практичне значення.

Метою дослідження є розробка системи, яка може виявляти та запобігати фішинговим атакам відповідно до відомих і нових факторів на основі звичайних методів аналізу електронної пошти та сучасних методів машинного навчання, зокрема оптимізації гіперпараметрів за допомогою алгоритму рою частинок.

Завдання дослідження: проаналізувати фішингові атаки в цифровому середовищі, з акцентом на цільовий фішинг як одну з найнебезпечніших форм соціальної інженерії; вивчити сучасні підходи до виявлення фішингових повідомлень, зокрема методи обробки природньої мови, машинного навчання та класичні протоколи автентифікації; розробити архітектуру гібридної системи виявлення фішингу на основі семантичного аналізу та логічної

регресії; реалізувати алгоритм оптимізації рою частинок (PSO) для налаштування ваг моделей та параметрів класифікації; провести теоретичну оцінку ефективності розробленої системи та проаналізувати її потенційні обмеження.

Об’єктом дослідження є процеси виявлення та запобігання фішинговим атакам у системах електронної пошти.

Предметом дослідження є методи виявлення цільового фішингу з використанням гібридного підходу, що включає традиційні засоби фільтрації, методи обробки природньої мови та оптимізацію параметрів моделей машинного навчання.

Теоретичне та/або практичне значення одержаних результатів: запропонована гібридна модель виявлення цільового фішингу має як практичну, так і теоретичну цінність у сфері інформаційної безпеки. З практичної точки зору, розроблена система може бути інтегрована у корпоративні рішення для захисту електронної пошти, забезпечуючи багаторівневий аналіз повідомлень, що поєднує перевірку автентичності, семантичну обробку тексту та адаптивну оптимізацію класифікацій. Така архітектура дозволяє ефективно виявляти персоналізовані фішингові атаки, які зазвичай обходять стандартні механізми фільтрації.

З теоретичної точки зору, дослідження поглиблює розуміння можливостей поєднання трансформерних мовних моделей (зокрема BERT) з метаевристичними методами оптимізації (PSO) у задачах кібербезпеки. Робота демонструє доцільність інтеграції різних рівнів аналізу повідомлень – технічного, семантичного та статистичного – в єдину адаптивну систему, що може слугувати основою для подальших досліджень у напрямку побудови інтерпретованих, ефективних і масштабованих захисних рішень.

РОЗДІЛ 1. АНАЛІЗ ФІШИНГОВИХ АТАК ТА МЕТОДІВ ЇХ ВИЯВЛЕННЯ

1.1. Феномен фішингу в сучасному інформаційному середовищі

Фішинг є одним із найпоширеніших методів соціальної інженерії, який використовується зловмисниками для несанкціонованого отримання конфіденційної інформації користувачів, зокрема паролів, даних банківських карток, реквізитів акаунтів та іншої критично важливої інформації. Цей тип атак базується не стільки на технічній складності, скільки на вмінні маніпулювати людською довірою та неуважністю [1].

Типовими каналами поширення фішингових повідомлень є електронна пошта, текстові повідомлення (SMS), соціальні мережі або телефонні дзвінки. У більшості випадків зловмисник імітує авторитетне джерело – відому компанію, фінансову установу або безпосереднього керівника – аби переконати користувача здійснити певну дію: перейти за посиланням, ввести свої дані чи відкрити вкладення [2].

Термін *«фішинг»* (від англ. *fishing* – риболовля) вперше почав активно використовуватись у середині 1990-х років у зв’язку з атаками на користувачів сервісу AOL. Згодом ця техніка набула масового поширення та стала одним із найефективніших методів цифрового шахрайства. Значне зростання масштабів фішингових атак було зафіксовано впродовж останніх десятиліть: за даними APWG, у 2016 році було зареєстровано понад 250 тисяч унікальних фішингових атак, а кількість підроблених доменів перевищила 95 тисяч [3].

Причини популярності фішингу – відносна легкість його організації, емоційна вразливість людей та відсутність належної цифрової гігієни. Особливо активно фішингові кампанії активізуються в періоди криз: пандемія COVID-19, воєнні дії, економічна нестабільність створюють ідеальне середовище для маніпуляцій.

Рисунок 1.1.1 – Цикл соціальної інженерії

Інфографіка показує поетапний процес, який злочинці використовують у соціальній інженерії: від збору даних про майбутню жертву до застосування здобутої інформації для отримання доступу до таємних відомостей. Ця низка підходів наголошує на необхідності негайної ідентифікації перших ознак фішингової діяльності [4].

Процес запуску фішингової кампанії сьогодні спрощено завдяки наявності готових інструментів. У разі доступу до бази електронних адрес, зловмисник отримує змогу ідентифікувати сервіси, до яких вони прив’язані, створити фальшивий сайт за допомогою інструментів копіювання HTML-структури, а далі – сформувати фішинговий лист, що копіює стиль офіційної комунікації. У таких повідомленнях зазвичай створюється відчуття

терміновості, загрози або невідкладної потреби: користувача лякають блокуванням акаунту, штрафом або втратою доступу. Цей механізм є типовим прикладом емоційного впливу, покликаний змусити людину діяти необачно.

Фішинг поділяється на кілька основних типів, кожен з яких має свої особливості залежно від каналу передачі інформації та рівня персоналізації.

Таблиця 1.1.1 – Основні види фішингу та їх характеристики [5].

Вид фішингу	Опис	Канал розповсюдження
Електронний	Масові розсилки з метою збору даних	Електронна пошта
Spear phishing	Персоналізовані атаки на окремих осіб	Електронна пошта
Smishing	Шкідливі SMS із фішинговими посиланнями	Текстові повідомлення
Vishing	Атаки через телефонні дзвінки	Телефонні дзвінки
Whaling	Атаки на керівників компаній (CEO, CFO)	Різні канали

Особливо небезпечною формою є цільовий фішинг (spear phishing) – атака, спрямована на конкретну людину або організацію. Вона ретельно планується: зловмисник заздалегідь збирає інформацію про жертву з відкритих джерел – соціальних мереж, корпоративних сайтів, новин – і формує персоналізоване повідомлення. Часто такий лист надходить, начебто, від колеги, керівника чи контрагента, з переконливим зверненням або запитом.

Успіх подібної атаки залежить від трьох складових:

- повідомлення має логічне обґрунтування в межах робочого чи особистого контексту;
- ім’я відправника або його посада викликають довіру;
- текст листа підтверджує справжність ситуації.

Згідно з даними Barracuda Networks (2023), цільовий фішинг поділяється на п’ять основних категорій: шахрайство (47%), імітація бренду (42%), компрометація ділової пошти (8%), вимагання (3%) та викрадення розмов (0,3%) [6].

Інтернет та соціальні мережі дають зловмисникам доступ до великої кількості персональної інформації. Це дозволяє їм ідентифікувати цінні цілі –

посадовців, держслужбовців, IT-фахівців – та сконструювати переконливі фальсифіковані повідомлення. Іноді фішингові листи містять вкладення або посилання, які, при відкритті, завантажують на пристрій жертви шкідливе програмне забезпечення.

Таким чином, фішинг еволюціонує від масових атак до високоточної соціальної інженерії, у якій жертва – це лише засіб доступу до значно важливіших цілей, таких як корпоративні мережі або конфіденційні дані.

Цільовий фішинг уже неодноразово довів свою ефективність навіть проти високозахищених організацій. Так, у 2016 році стався гучний витік даних Національного комітету Демократичної партії США. Зловмисники надіслали фішинговий лист, який виглядав як повідомлення служби безпеки Google із проханням змінити пароль. Один із політичних радників натиснув на підроблене посилання – і внаслідок цього хакери отримали доступ до внутрішньої електронної пошти, що мало серйозний резонанс у ЗМІ [7].

Розвиток цифрових технологій призвів до еволюції фішингових атак як за обсягами, так і за якістю виконання. Насамперед це пов’язано зі зростанням обсягів електронного листування: щодня у світі надсилається понад 330 мільярдів електронних листів, значна частина з яких використовується як канал для доставки шкідливого контенту.

Зміни також відбулися у тактиках зловмисників:

- Атаки стали коротшими, лаконічнішими та точнішими.
- Застосовується імітація стилю письма конкретної особи.
- Активно використовуються викрадені теми попередніх листів – так званий «conversation hijacking».

Сучасні технології значно ускладнюють виявлення фішингових атак. AI-генератори тексту (наприклад, ChatGPT) допомагають зловмисникам створювати граматично правильні, переконливі листи без мовних помилок. Deepfake-технології використовуються не лише у відео, а й для симуляції

голосу у vishing-атаках. Фейкові вебсайти більше не мають грубих візуальних помилок – дизайн, домени, навіть SSL-сертифікати виглядають достовірно.

Більш того, зловмисники використовують нейромережі для збору даних із відкритих профілів користувачів у соцмережах, щоб автоматично генерувати персоналізовані звернення. Це значно підвищує ефективність атак і зменшує шанси на їх своєчасне виявлення.

Особливо вразливими до економічних наслідків є малі та середні підприємства, які не мають потужних систем кіберзахисту, органи місцевого самоврядування, які часто мають застаріле програмне забезпечення та фінансовий сектор, де одна помилка може коштувати мільйони.

До непрямих економічних наслідків належать:

- Втрати репутації компанії або бренду після успішної атаки;
- Витрати на відновлення систем, аудит безпеки та компенсацію постраждалим клієнтам;
- Падіння довіри інвесторів та партнерів, що особливо критично для публічних компаній.

Великі корпорації несуть набагато більшу відповідальність за данні користувачів. Але не завжди вдається уникнути загрози. Ось декілька конкретних прикладів великомасштабних витоків даних, які відбулися в останні роки. Витік даних LinkedIn у 2021 році вплинув на 700 мільйонів користувачів через скомпрометовані дані. У 2019 та 2021 роках відбувся витік даних у Facebook. Постраждали близько 533 мільйони облікових записів, що містили конфіденційну інформацію, таку як номери телефонів та електронні листи. Найбільшим в історії витоком називають витік даних Cam4, скомпрометувавши 10.88 мільярда записів через те, що його база даних залишилася відкритою без будь-якої автентифікації. Схожа ситуація відбулась з Exactis Data Bleak у 2018 році. Понад 340 мільйонів записів було розкрито, коли було виявлено, що база даних є загальнодоступною без захисту [8].

Усе це демонструє, що фішинг – не просто технічна вразливість, а явище з реальними економічними наслідками, яке вимагає системної протидії як на рівні окремих організацій, так і на рівні державної політики кіберзахисту.

1.2. Методологія виявлення фішингу: огляд сучасних підходів

Одним із перших рівнів захисту від фішингових атак є методи перевірки автентичності відправника. Ці технології покликані забезпечити достовірність інформації про джерело листа, виявити підроблені домени та захистити користувачів від spoofing-атак [9].

До основних класичних методів належать:

SPF (Sender Policy Framework) – перевіряє, чи має сервер, з якого надіслано листа, дозвіл надсилати пошту від імені конкретного домену .

Як працює:

1. У DNS-записі домену зберігається список IP-адрес, які мають право надсилати пошту.
2. Коли поштовий сервер отримує лист, він перевіряє, чи IP-адреса відправника входить до цього списку.
3. Якщо не входить – лист може бути позначено як підозрілий або взагалі відхилено.

Та у цього метода є і обмеження:

– SPF перевіряє лише IP-адресу, а не вміст або заголовки листа.

– Якщо лист пересилається через інші сервери, перевірка може не спрацювати.

DKIM (DomainKeys Identified Mail) – забезпечує цілісність повідомлення та підтверджує, що його дійсно відправлено з авторизованого домену.

Як працює:

1. Відправник підписує лист цифровим підписом, створеним за допомогою приватного ключа.

2. Одержувач перевіряє цей підпис, використовуючи публічний ключ із DNS-запису домену.
3. Якщо вміст листа було змінено – підпис не пройде перевірку.

У такого метода є переваги:

– Дозволяє виявити змінені або фальсифіковані повідомлення.

– Працює навіть при пересиланні листів.

DMARC (Domain-based Message Authentication, Reporting and Conformance) – об'єднує SPF і DKIM та встановлює політику, як сервери-отримувачі повинні поводитись із листами, які не пройшли перевірку.

Як працює:

1. Адміністратор домену налаштовує політику DMARC у DNS: наприклад, відхилити всі листи, які не пройшли SPF і DKIM.
2. Одержувач перевіряє лист і діє згідно з цією політикою (наприклад, блокує або позначає як спам).
3. DMARC також надсилає адміністратору звіти про спроби зловживання доменом.

Переваги DMARC:

– Дозволяє доменам активно захищати свою репутацію.

– Може значно зменшити кількість спуфінгових листів.

Таблиця 1.2.1 - Взаємодія SPF, DKIM та DMARC [10]

Метод	Перевіряє	Захищає від	Мінуси
SPF	IP-адресу відправника	Спуфінг	Не працює при форвардингу
DKIM	Цілісність та підпис	Зміна вмісту	Не всі поштові сервіси його реалізують
DMARC	SPF + DKIM + політика	Спуфінг + фальсифікація	Потребує правильної настройки всіх компонентів

Попри ефективність класичних механізмів перевірки автентичності (SPF, DKIM, DMARC), зловмисники дедалі частіше обходять ці захисні заходи, використовуючи легітимні, але скомпрометовані облікові записи або домени.

У зв’язку з цим особливої актуальності набувають інтелектуальні методи аналізу вмісту повідомлень, які не прив’язані до технічної інформації про відправника, а ґрунтуються на поведінкових, контекстуальних і лінгвістичних характеристиках листів.

Аналіз вмісту електронних листів (Content-Based Filtering) [11].

Цей підхід передбачає вивчення тексту повідомлення з метою виявлення шаблонів, притаманних фішинговим атакам. Серед таких шаблонів:

- використання термінів на кшталт “терміново”, “підтвердити акаунт”, “оновити інформацію”;
- наявність підозрілих гіперпосилань або вкладень;
- лексико-семантичні конструкції, що апелюють до страху, терміновості або авторитету.

У рамках цього підходу застосовуються методи обробки природної мови (NLP), такі як TF-IDF (Term Frequency-Inverse Document Frequency), n-грам моделі, Word2Vec, FastText тощо.

Класифікація на основі машинного навчання

Після векторного подання тексту повідомлення система може застосовувати алгоритми класифікації, здатні автоматично віднести лист до категорії "фішинговий" або "легітимний". Найчастіше використовуються такі моделі:

- логістична регресія;
- наївний баєсівський класифікатор;
- метод опорних векторів (SVM);
- дерева рішень і ансамблеві моделі (Random Forest, XGBoost).

Оцінювання ефективності таких моделей здійснюється за метриками точності (accuracy), повноти (recall), точності позитивного прогнозу (precision) і F1-міри [12].

Глибокі нейронні мережі (Deep Learning)

Для виявлення більш складних фішингових схем застосовуються глибокі нейронні мережі. Зокрема, рекурентні архітектури типу LSTM (Long Short-

Term Memory) здатні враховувати контекст і послідовність слів, що дозволяє точніше ідентифікувати маніпулятивні мовні патерни [13]. Більш сучасні підходи базуються на трансформерах, таких як BERT (Bidirectional Encoder Representations from Transformers), DistilBERT, RoBERTa тощо [14].

Ці моделі показують високу точність класифікації, однак вимагають значних обчислювальних ресурсів і якісного попереднього навчання на великих корпусах текстів.

Поведінковий аналіз

Деякі системи фільтрації фішингових повідомлень орієнтуються не на зміст листа, а на поведінку користувача. Такі системи враховують шаблони взаємодії з поштовим клієнтом: частоту та обсяг листування, типові адреси, часові рамки активності, геолокаційні ознаки, тип пристрою тощо.

Відхилення від встановлених моделей поведінки можуть свідчити про потенційну загрозу або спробу компрометації.

Гібридні системи

Найефективніші рішення на сучасному етапі реалізуються у формі гібридних моделей. Вони поєднують кілька рівнів аналізу: перевірку технічної автентичності (SPF, DKIM, DMARC), класифікацію за допомогою машинного навчання, аналіз лінгвістичних і поведінкових характеристик, а також застосовують алгоритми оптимізації – зокрема, рої частинок (PSO) або генетичні алгоритми – для підвищення гнучкості налаштувань і адаптивності моделей.

У відповідь на зростання кількості фішингових атак провідні технологічні компанії активно впроваджують інструменти для їх виявлення та запобігання. Серед найбільш поширених рішень, що використовуються у корпоративному середовищі, варто виокремити системи безпеки від Google, Microsoft і Cofense, а також низку спеціалізованих платформ на базі штучного інтелекту.

Платформа Google Workspace реалізує багаторівневу модель захисту, яка поєднує класичні механізми перевірки (SPF, DKIM, DMARC) з алгоритмами

машинного навчання, навченими на великому обсязі власних даних. Антифішинг-модуль Gmail аналізує поведінкові індикатори (наприклад, різке зростання частоти надсилання листів з одного домену), структуру й стилістику повідомлень, а також інші ознаки, що властиві фішинговим атакам. Важливою перевагою Google є глибока інтеграція захисних алгоритмів у внутрішню інфраструктуру поштового сервісу, що сприяє швидкій обробці повідомлень і зменшенню кількості помилок класифікації [15].

Microsoft Defender for Office 365 орієнтований передусім на корпоративних користувачів і забезпечує захист на рівні всієї екосистеми Microsoft 365. Цей інструмент поєднує перевірку вкладень у віртуальних "пісочницях", аналіз гіперпосилань у режимі реального часу, а також модулі автоматизованого реагування. Крім сигнатурних і евристичних методів, у Defender застосовуються предикативні моделі, що дозволяють виявляти нові загрози, які ще не були зафіксовані в базах даних. Важливою особливістю є використання телеметричних даних із глобальної мережі користувачів, що дає змогу постійно вдосконалювати алгоритми за допомогою самонавчання [16].

Платформа Cofense спеціалізується винятково на виявленні фішингових загроз та навчанні персоналу. Її рішення Cofense PhishMe дозволяє моделювати фішингові атаки з метою підвищення обізнаності співробітників. У свою чергу, Cofense Intelligence виконує аналіз загроз, які пройшли початковий захист, та формує звіти для безпекових команд. Cofense поєднує автоматизовані алгоритми з ручним аналізом, що підвищує ефективність виявлення складних атак, зокрема цільових. Однією з ключових переваг цієї платформи є акцент на співпрацю людини й машини: користувачі самостійно позначають підозрілі листи, що сприяє підвищенню загальної стійкості організації до фішингових загроз [17].

Серед новітніх рішень вирізняються платформи, побудовані на технологіях глибокого навчання, зокрема IRONSCALES і Area 1 Security. Вони інтегруються з поштовими системами організацій і виконують аналіз не лише змісту повідомлень, а й контексту: історії листування, типових шаблонів

поведінки користувачів, інтонацій повідомлень тощо. Такі рішення особливо ефективні у протидії персоналізованим spear phishing-атакам, хоча їх продуктивність напряму залежить від якості початкового навчання моделі та гнучкості адаптації до конкретного корпоративного середовища.

Відштовхуючись від вище перерахованого, можна стверджувати, що системи від Google та Microsoft пропонують масштабовані рішення з високим рівнем автоматизації, тоді як Cofense та інші вузькоспеціалізовані платформи забезпечують більшу гнучкість та можливість адаптації до специфічних сценаріїв атак. Водночас жодна з них не забезпечує повного захисту, що зумовлює актуальність розробки гібридних систем, які б поєднували найкращі риси класичних підходів, аналітичного моделювання та залучення користувачів у процес виявлення загроз.

Попри активний розвиток технологій захисту від фішингових атак, жодна з наявних систем не забезпечує гарантованої ефективності. На практиці кожен із підходів має свої обмеження – як технічні, так і прикладні – що часто ускладнює їх використання в реальних умовах.

Так, класичні механізми перевірки автентичності електронної пошти (SPF, DKIM, DMARC) значною мірою залежать від правильного налаштування з боку відправника. Якщо домен не підтримує ці записи або вони конфігуровані з помилками, захист просто не спрацює. До того ж, навіть коректно налаштовані протоколи не рятують у випадках, коли атака здійснюється зі зламаного акаунта на справжньому домені. У результаті цільовий фішинг або внутрішньокорпоративне шахрайство легко минає перевірку.

Методи машинного навчання, хоч і показують хороші результати, значною мірою залежать від даних, на яких вони були навчені. При зміні тактик атак або появі нових векторів загроз модель може швидко втратити актуальність. До того ж, не всі фішингові листи мають яскраво виражені ознаки – у таких випадках навіть найточніші NLP-алгоритми можуть помилятись: хибнопозитивні або хибнонегативні результати – не рідкість. А в

корпоративному середовищі це ризик: достатньо заблокувати один важливий лист, і виникає проблема.

Щодо глибоких нейронних мереж (таких як LSTM або BERT), вони справді вражають точністю, проте потребують значних обчислювальних ресурсів. Їх не так просто впровадити в умовах середнього бізнесу, де часто немає ані серверів, ані фахівців із моделювання. Більше того, такі моделі складно “пояснити” – вони працюють як “чорний ящик”, що викликає труднощі в аналізі інцидентів і підготовці звітності.

Не менш важливим є й той факт, що поведінкові системи не захищені від маніпуляцій. Зловмисник може поступово “вбудуватись” у шаблон звичної активності, особливо якщо має доступ до компрометованого акаунта з історією листування. У такому разі навіть складні гібридні системи можуть не виявити атаку вчасно [18].

Отже, нинішні підходи мають кілька критичних слабких місць:

– обмежене охоплення атак, що маскуються під нормальну активність;
– залежність від якості вхідних даних і контексту їх використання;
– складність інтерпретації результатів при використанні глибоких моделей;
– низька гнучкість класичних систем у боротьбі з персоналізованими загрозами.

Усе це свідчить про потребу в новому поколінні гібридних рішень, які могли б об'єднувати традиційні методи перевірки, лінгвістичний аналіз і поведінкові моделі з адаптивними алгоритмами оптимізації – для більш гнучкого й стійкого захисту в умовах змінного цифрового ландшафту.

1.3. Висновки до розділу 1.

У першому розділі надається детальне пояснення фішингових атак як одного з найбільш поширених і шкідливих механізмів соціальної інженерії сучасного цифрового світу. Обговорюються найбільш поширені форми фішингу, а саме цілеспрямовані атаки, які характеризуються високою

персоналізацією та неможливістю виявлення. Приклади з реальної практики, зокрема випадок Національного комітету Демократичної партії Сполучених Штатів, чітко ілюструють масштаб потенційного впливу – від грошових втрат до втрати репутації.

Фокус був зміщений на методи виявлення фішингових повідомлень: від класичних протоколів аутентифікації (SPF, DKIM, DMARC) до сучасних інтелектуальних рішень із використанням обробки природної мови, алгоритмів машинного та глибокого навчання, аналізу поведінки та гібридних моделей. Незважаючи на підвищення точності таких рішень, вони залишаються вразливими до нових векторів атак, особливо тих, які здійснюються через законні облікові записи або імітують звичайну поведінку.

У процесі аналізу також виявлено, що ефективність інструментів, зокрема Google Workspace, Microsoft Defender for Office 365, Cofense, залежить не лише від технологічних можливостей, а й від контексту впровадження, масштабів організації та участі користувачів у системі захисту.

Таким чином, було зрозуміло, що існуючі методи, демонструючи певний ступінь надійності, не здатні забезпечити повний захист в реаліях сучасного середовища загроз. Саме цей факт вимагає доцільності розробки інтелектуальної цільової системи виявлення фішингу, яка б включала механізми технічної перевірки, лінгвістичний та поведінковий аналіз, а також адаптивну оптимізацію моделі.

Надалі в межах дослідження ставиться задача розробки гібридної інтелектуальної системи для виявлення цільового фішингу, яка поєднує класичні методи перевірки автентичності електронних листів (SPF, DKIM, DMARC), семантичну обробку тексту за допомогою моделей типу BERT, базовий класифікатор для виявлення фішингових повідомлень та оптимізацію його параметрів з використанням алгоритму рою частинок (PSO). Такий підхід дозволяє поєднати переваги різних типів аналізу – як технічного, так і контекстуального – з метою покращення точності виявлення складних та адаптивних атак, зокрема цільового фішингу, який часто обходить стандартні

захисні механізми. Використання PSO дозволить налаштувати ваги моделі для досягнення оптимального балансу між виявленням реальних загроз та мінімізацією хибнопозитивних результатів.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ВИЯВЛЕННЯ ЦІЛЬОВОГО ФІШИНГУ

2.1. Методологія аналізу та побудова архітектури системи

Проєктування інтелектуальної системи виявлення цільового фішингу передбачає поетапне обґрунтування вибору методів, моделей і структурних компонентів системи. У даному дослідженні обґрунтовано використання гібридного підходу, який поєднує традиційні методи перевірки автентичності електронної пошти, семантичний аналіз вмісту повідомлень за допомогою трансформерних моделей і оптимізацію класифікаційних параметрів за допомогою алгоритму рою частинок (PSO). Такий підхід дозволяє створити адаптивну архітектуру, стійку до різних сценаріїв фішингових атак.

Першим етапом побудови системи є технічна перевірка автентичності електронних повідомлень, яка реалізується за допомогою протоколів SPF (Sender Policy Framework), DKIM (DomainKeys Identified Mail) та DMARC (Domain-based Message Authentication, Reporting and Conformance). Ці протоколи дозволяють виявляти базові підробки на рівні домену відправника, цифрового підпису повідомлень та відповідності політиці обробки листів, які не проходять автентифікацію [19].

Другий рівень аналізу стосується вмісту електронного листа. Для подання тексту повідомлення у векторному просторі обрано модель BERT (Bidirectional Encoder Representations from Transformers), яка дозволяє здійснювати контекстну семантичну інтерпретацію тексту [20, 21]. BERT враховує контекстне значення слів залежно від їхнього оточення, що є критично важливим для виявлення латентних ознак фішингу, які неможливо виявити за допомогою частотних підходів, таких як TF-IDF або n-грам.

Для задачі класифікації повідомлень обрано поєднання двох моделей: логістичної регресії та нейронної мережі на основі LSTM (Long Short-Term Memory). Логістична регресія забезпечує базову інтерпретовану модель з

високою швидкістю навчання [22], у той час як LSTM дозволяє виявляти послідовні патерни та контекстні залежності в тексті, що часто властиві персоналізованим фішинговим листам [23].

З метою підвищення точності системи та зменшення кількості хибнопозитивних/хибнонегативних результатів, застосовується алгоритм оптимізації рою частинок (Particle Swarm Optimization, PSO), який виконує налаштування вагових коефіцієнтів ознак, гіперпараметрів моделей та їхніх комбінацій [24]. PSO є метаевристичним адаптивним методом, який моделює колективну поведінку біологічних систем і забезпечує ефективний пошук у просторах із великою кількістю параметрів та локальних мінімумів.

PSO базується на спостереженнях за колективною поведінкою роїв у природі (наприклад, птахів або риб). Кожна частинка у цьому алгоритмі є потенційним розв'язком задачі й "рухається" простором пошуку, поступово наближаючись до найкращого знайденого результату. Принцип оновлення положення частинки формально описується двома основними рівняннями.

Оновлення швидкості частинки визначається так:

$$v_i^{(t+1)} = v_i^{(t)} + c_1 \times r_1 \times \left(p_i^{best} - x_i^{(t)}\right) + c_2 \times r_2 \times (g^{best} - x_i^{(t)})$$

де:

$v_i^{(t)}$ – поточна швидкість частинки i,

$x_i^{(t)}$ – її позиція у просторі рішень та ітерації t,

p_i^{best} – найкраща позиція, яку ця частинка відвідала,

g^{best} – найкраща позиція серед усіх частинок (глобальний мінімум),

c_1, c_2 – коефіцієнти когнітивної та соціальної складової (зазвичай = 2),

r_1, r_2 – випадкові значення з інтервалу [0, 1], що додають стохастичність.

Після оновлення швидкості, положення частинки коригується за наступною формулою:

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \times \Delta t$$

Таким чином, модель, що оптимізується за допомогою PSO, здатна адаптуватися до змін у шаблонах фішингових повідомлень, мінімізуючи функцію втрат не лише на навчальній, а й на тестовій вибірці. Це дозволяє системі зберігати високу точність навіть при появі нових типів атак, що не були представлені в навчальних даних.

У результаті побудовано архітектуру, здатну комбінувати ознаки з різних рівнів (технічного, семантичного, поведінкового), що забезпечує стійкість до обхідних тактик фішингових кампаній. Завдяки модульному принципу, систему можна масштабувати або модифікувати, додаючи нові компоненти (наприклад, модуль поведінкового аналізу) без втрати цілісності її функціонування.

Нижче наведено результат прикладу реалізації алгоритму рою частинок (PSO) для задачі мінімізації функції Растрігіна – однієї з класичних тестових функцій в оптимізації, яка характеризується великою кількістю локальних мінімумів і високою складністю глобального пошуку. Алгоритм було реалізовано мовою програмування Python з використанням бібліотек numpy, random, matplotlib, а також базових об'єктно-орієнтованих засобів мови для моделювання поведінки частинок (А.1).

У межах експерименту кожна частинка відповідала потенційному рішенню у двовимірному просторі, а її рух керувався класичними рівняннями оновлення швидкості та позиції відповідно до моделі, запропонованої Kennedy та Eberhart. Упродовж ітерацій алгоритм зменшував значення цільової функції, наближаючись до глобального мінімуму, що засвідчило ефективність використаного підходу.

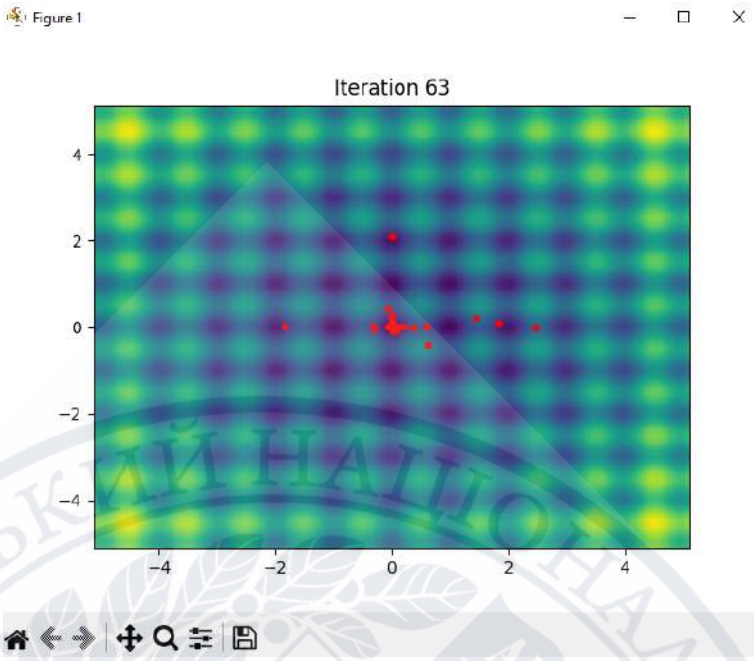


Рисунок 2.1.1 – Візуалізація динаміки рою частинок

На рисунку представлена двовимірна візуалізація процесу оптимізації функції Растрігіна за допомогою метаевристичного методу. Фонове зображення є кольоровою картою значень функції, де кольори відповідають різним рівням її значень: темно-фіолетові області позначають нижчі значення функції, а жовто-зелені – вищі. Таким чином, темні області на зображенні відповідають потенційним глобальним або локальним мінімумам функції.

На графіку також зображено червоні точки, які представляють поточні положення агентів (частинок) на 63-ій ітерації алгоритму. Розміщення більшості агентів поблизу центра координат (поблизу точки (0,0)) свідчить про їх збіжність до глобального мінімуму, який для функції Растрігіна досягається саме в цій точці.

Зображення демонструє характерну мультиекстримальну структуру функції Растрігіна, яка ускладнює процес оптимізації через велику кількість локальних мінімумів. Це добре видно завдяки регулярному хвильовому малюнку на площині, який виникає через періодичні гармонічні складові у формулі функції.

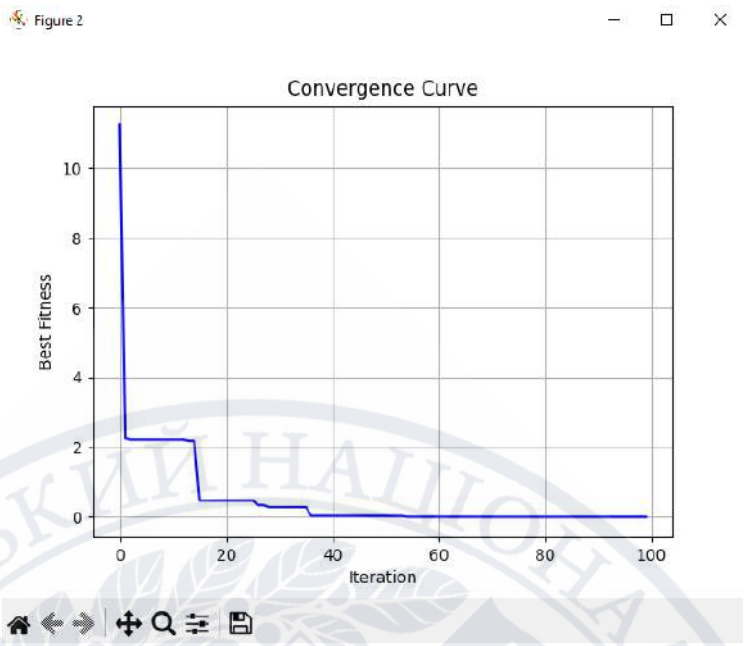


Рисунок 2.1.2 – Графік збіжності алгоритму PSO

Цей графік ілюструє зміну найкращого знайденого значення функції на кожній ітерації. Вертикальна вісь відображає значення функції (fitness), а горизонтальна – номер ітерації. Графік показує стабільне зниження функції втрат протягом обчислень, що є ознакою успішної збіжності алгоритму до оптимального розв'язку. Хоча в окремих місцях спостерігається плато або незначні коливання, загальна тенденція свідчить про поступове поліпшення рішень.

Отже, архітектура запропонованої системи реалізується у вигляді чотирьох взаємопов'язаних модулів:

- **Модуль автентифікації** – виконує перевірку SPF, DKIM та DMARC, фіксує наявність технічних порушень;
- **Модуль семантичного аналізу** – за допомогою моделі BERT будує контекстні векторні подання тексту листа;
- **Модуль класифікації** – застосовує логістичну регресію та LSTM для визначення ймовірності фішингового характеру повідомлення;
- **Оптимізаційне ядро (PSO)** – адаптивно налаштовує ваги моделей та параметри класифікації, забезпечуючи максимізацію ефективності системи.

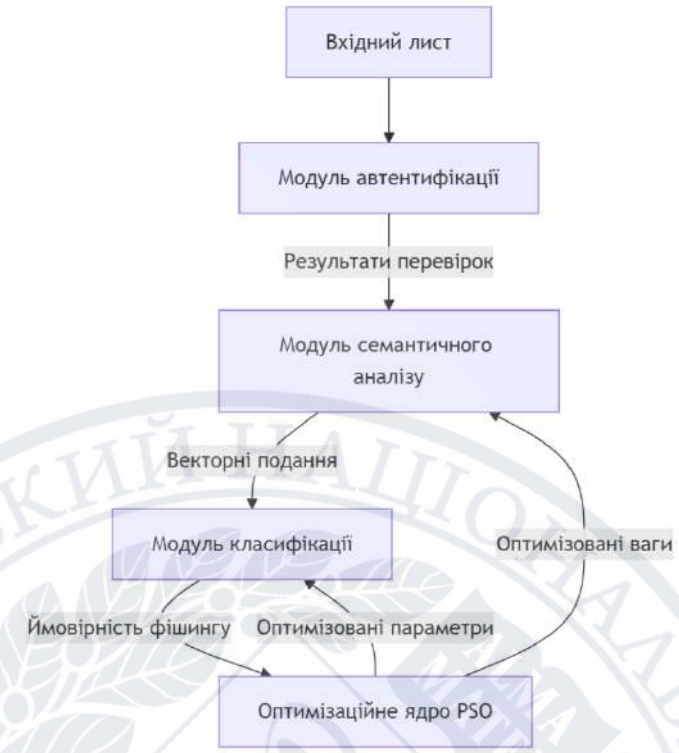


Рисунок 2.1.3 – загальна архітектура системи

2.2. Проєктування гібридної моделі

При проєктуванні гібридної моделі виявлення цільового фішингу особливу увагу приділено обґрунтованому вибору методів класифікації, форм представленості даних, а також способів адаптивної оптимізації параметрів системи. Задля цього було проаналізовано сучасні дослідження в галузі, зокрема модель, запропоновану у роботі *"Using BERT with Metaheuristic Optimized XGBoost for Phishing Email Identification"* (2024), яка поєднує семантичну модель BERT, класифікатор XGBoost та метаевристичні методи для оптимізації [25].

У вказаному дослідженні вхідні електронні листи аналізуються виключно на рівні тексту. Для представлення змісту використовується BERT, що дозволяє сформувати контекстно залежні векторні подання. Класифікація здійснюється за допомогою XGBoost – ансамблевого алгоритму, що поєднує потужність дерев рішень з гнучкістю регулювання переобучення. Оптимізація гіперпараметрів моделі виконується з використанням метаевристичних

алгоритмів (наприклад, Genetic Algorithm, Firefly, або інші модифіковані підходи). Модель показує високу точність при класифікації фішингових повідомлень.

Втім, аналіз цієї моделі виявив декілька обмежень, які були враховані при побудові нашої системи:

1. Обмеженість ознак лише текстовим вмістом. У той час як модель 2024 року працює виключно з вмістом листа, у даному дослідженні зроблено акцент на багаторівневому аналізі. До векторного подання додаються технічні ознаки електронного листа – SPF, DKIM, DMARC – які дозволяють виявити спроби фальсифікації на рівні протоколів. Це особливо актуально для цільового фішингу, де зловмисники можуть використовувати скомпрометовані або схожі домени.
2. Питання пояснюваності. Ансамблеві методи на зразок XGBoost є високоточними, але менш інтерпретованими для користувача. Це ускладнює впровадження системи у корпоративне середовище, де адміністратори потребують зрозумілого пояснення класифікаційних рішень. Тому у запропонованій системі в якості основного класифікатора обрано логістичну регресію, яка дозволяє прозоро оцінювати вагу кожного параметра та надає можливість чітко інтерпретувати ознаки, що вплинули на виведення [26].
3. Спрощення обчислювальної складності. XGBoost має високу обчислювальну вартість як у тренуванні, так і в інференсі, особливо у поєднанні з BERT. У контексті впровадження в реальне середовище (наприклад, корпоративний поштовий шлюз) це може стати перешкодою. Запропонована модель поєднує BERT із менш ресурсомістким класифікатором, забезпечуючи прийнятну швидкість обробки навіть на серверних системах із обмеженими ресурсами.
4. Використання оптимізації PSO. Замість складних або авторських метаевристик, обрано класичний, добре досліджений алгоритм рою частинок (PSO), який має хорошу збіжність у задачах налаштування ваг,

гіперпараметрів та балансування між технічними і семантичними ознаками. Це спрощує імплементацію, забезпечує стабільну оптимізацію навіть у разі високої розмірності ознак, і дозволяє ефективно налаштовувати модель без необхідності створення кастомних алгоритмів.

Таким чином, запропонована модель модифікує підхід, описаний у роботі 2024 року [25], шляхом:

- розширення вхідного простору ознак до технічних і семантичних рівнів;
- переходу до інтерпретованого класифікатора для покращення інтеграції в корпоративні рішення;
- використання класичної, стабільної метаевристики PSO замість складніших алгоритмів;
- досягнення компромісу між точністю, обчислювальною ефективністю та пояснюваністю.

Результатом є гнучка, адаптивна, інтерпретована система, орієнтована на виявлення саме цільових фішингових атак, які часто залишаються поза увагою традиційних фільтрів.

2.3. Етичні аспекти впровадження гібридної системи виявлення цільового фішингу

У процесі розробки та впровадження гібридної системи виявлення цільового фішингу, яка поєднує класичні методи перевірки автентичності електронних листів (SPF, DKIM, DMARC), семантичну обробку тексту за допомогою BERT та оптимізаційні алгоритми (зокрема рою частинок – PSO), особливу увагу слід приділити дотриманню чинного законодавства щодо захисту персональних даних. Оскільки така система обробляє електронну пошту, в тому числі її вміст, заголовки, домени та інші метадані, вона

потенційно працює з персональною або службовою інформацією, що підлягає правовому захисту.

Основою регулювання в цій сфері є Загальний регламент захисту даних Європейського Союзу (GDPR), який встановлює жорсткі вимоги щодо прозорості, законності, обмеження мети, мінімізації обробки, забезпечення безпеки та контролю над обробкою персональних даних. Незважаючи на те, що Україна не є членом ЄС, положення GDPR все частіше беруться до уваги у вітчизняній практиці – як при розробці нових нормативних актів, так і при формуванні внутрішньої політики безпеки в організаціях. Український Закон «Про захист персональних даних» також містить ключові положення щодо законності обробки, добровільної згоди суб'єкта даних, прав на доступ, зміну та видалення персональної інформації [27].

У контексті розроблюваної системи, усі компоненти, пов'язані з обробкою тексту електронних листів, повинні працювати виключно в межах чітко визначеної мети – виявлення потенційних фішингових атак. Заборонено використовувати отримані дані для інших цілей, зокрема комерційних або аналітичних, не пов'язаних із безпекою. Це означає, що моделі обробки природної мови (наприклад, BERT) мають бути навчальні або використовуватись лише для аналізу мінімально необхідної інформації, без надмірного збору даних, які не впливають на прийняття рішень. Крім того, зберігання та передача таких даних повинні бути захищені за допомогою сучасних криптографічних методів (шифрування, токенізація, псевдонімізація).

Ще одним критичним аспектом є прозорість обробки. GDPR встановлює вимогу, згідно з якою суб'єкт даних має право отримати інформацію про те, що саме відбувається з його даними, які саме з них обробляються, на якій підставі та з якою метою. Тому система виявлення фішингу має бути здатною пояснити, які ознаки в листі стали підставою для його класифікації як небезпечного або підозрілого, навіть якщо в основі рішення лежить складна модель на кшталт трансформера. Якщо рішення ухвалюється виключно автоматизовано,

користувач повинен мати можливість оскаржити його, а в окремих випадках – вимагати перегляду рішення людиною.

Особливу увагу варто звернути на статтю 22 GDPR, яка забороняє ухвалення рішень, що мають суттєвий вплив на особу, винятково на основі автоматизованої обробки без людського втручання. У випадку, якщо система блокує важливі листи або повідомляє про спробу фішингової атаки, така дія повинна супроводжуватися логуванням, можливістю ручної перевірки та наданням користувачеві зворотного пояснення щодо причин такого рішення. Це є обов'язковим не лише з етичної точки зору, а й із правової, оскільки стосується прав людини на справедливе ставлення при обробці її персональних даних.

Нарешті, враховуючи відсутність в Україні повноцінного наглядового органу із широкими повноваженнями у сфері захисту даних (аналогічного європейським Data Protection Authorities), важливо самостійно дотримуватись принципів «privacy by design» та «privacy by default». Це передбачає, що конфіденційність має бути вбудованою в архітектуру системи з самого початку, а за замовчуванням усі налаштування повинні гарантувати максимальний рівень захисту користувача без потреби в його втручанні. Також доцільно проводити регулярну оцінку впливу на захист персональних даних (DPIA), особливо якщо система використовується в корпоративному або державному середовищі [28].

Ефективність гібридної системи виявлення цільового фішингу значною мірою залежить від її здатності адаптуватися до змін у тактиках зловмисників. У середовищі, де фішингові кампанії постійно еволюціонують – стають коротшими, більш граматично правильними та персоналізованими, – використання навіть найсучаснішої моделі без її регулярного оновлення призводить до швидкої втрати актуальності. Саме тому критично важливим етичним і технічним аспектом є періодичне тестування та перенавчання моделі, на основі нових прикладів фішингових і легітимних листів.

Гібридна архітектура у цій роботі, потребує цілеспрямованого моніторингу всіх її компонентів. Зокрема, контекстна модель BERT має властивість втрачати релевантність у нових середовищах, якщо вона була попередньо навчена на корпусах, які не відображають останні тенденції в соціотехнічних атаках. Алгоритм PSO, хоч і виконує функцію оптимізації гіперпараметрів, теж базується на вже наявних даних, тому не зможе забезпечити адаптивність, якщо характер вхідних повідомлень суттєво зміниться.

Перевірка ефективності повинна здійснюватись за допомогою стандартних метрик машинного навчання: точності (accuracy), повноти (recall), точності позитивного прогнозу (precision) та F1-міри. Проте важливо також враховувати зміну розподілу даних (concept drift), яка може відбуватись непомітно, але суттєво впливати на роботу всієї системи. Наприклад, зміна мовного стилю, типових загрозливих фраз або структури листів у корпоративному середовищі вимагає відповідного оновлення не лише навчальної вибірки, а й архітектури моделі [29].

Системи, які працюють у реальному часі або використовуються у важливих середовищах – наприклад, у компаніях чи державних установах – повинні мати спосіб автоматично або напівавтоматично перевіряти, чи не погіршилася якість моделі. Це може бути регулярне тестування на окремих перевірочних даних, відстеження кількості помилкових спрацювань, а також збір відгуків від користувачів про те, чи система неправильно щось заблокувала або, навпаки, щось пропустила. В залежності від результатів такої перевірки, модель або замінюють повністю, або донавчають на нових даних, щоб покращити її роботу.

Регулярна перевірка моделі – це не лише вимога якості, а й важливий елемент етичної відповідальності. Користувачі, які довіряють системі свою електронну комунікацію, мають право очікувати, що вона залишається ефективною не тільки на момент впровадження, а й упродовж всього життєвого циклу. Тому процес оновлення повинен бути систематизованим,

прозорим і добре задокументованим, із чітким розподілом відповідальності за контроль якості.

Ще однією з головних етичних вимог до інтелектуальних систем виявлення загроз є прагнення до зменшення кількості помилок – як хибнопозитивних, так і хибнонегативних. У контексті цієї гібридної системи виявлення цільового фішингу, це питання набуває особливого значення. Від кількості помилок безпосередньо залежить як безпека користувачів, так і рівень довіри до системи.

Хибнопозитивне спрацювання – це ситуація, коли система позначає легітимне повідомлення як фішингове. У корпоративному або державному середовищі це може мати серйозні наслідки: втрату важливої інформації, затримку у процесах комунікації або навіть дестабілізацію внутрішніх бізнес-процесів. Хибнонегативні результати – ще більш критичні, адже в такому випадку система пропускає реальну загрозу, що може призвести до витоку конфіденційної інформації, проникнення в інфраструктуру або фінансових збитків.

Мінімізація обох типів помилок вимагає балансування між чутливістю та специфічністю моделі. У межах створеної системи це досягається кількома способами. По-перше, використання гібридного підходу дозволяє зменшити залежність від одного джерела інформації: наприклад, навіть якщо SPF-запис проходить перевірку, система все одно може виявити підозрілий зміст повідомлення завдяки BERT. По-друге, алгоритм оптимізації рою частинок (PSO) виконує налаштування ваг та гіперпараметрів моделей, що дозволяє тонко настроїти систему на досягнення оптимального співвідношення між recall і precision. Такий підхід дає змогу зменшити ймовірність того, що критичні атаки залишаться непоміченими, одночасно знижуючи кількість помилкових блокувань.

Крім технічних рішень, важливу роль відіграє моніторинг і збір статистики щодо помилок у режимі реального часу. Наявність механізму

ручного маркування повідомлень користувачем (наприклад, «це не фішинг» або «це небезпечне») дозволяє швидко реагувати на зміну шаблонів загроз та постійно уточнювати поведінку моделі. Такий підхід забезпечує не лише динамічне вдосконалення системи, а й зменшує ризик систематичних помилок унаслідок викривлень у навчальних даних або структурі моделі.

Попри високий рівень автоматизації, навіть найефективніша система виявлення цільового фішингу не може повністю замінити роль користувача в процесі забезпечення інформаційної безпеки. Впровадження інтелектуальної гібридної системи вимагає не лише технічної інтеграції, а й паралельного підвищення цифрової обізнаності персоналу. Це особливо актуально в умовах, коли фішингові кампанії дедалі більше орієнтуються на психологічні вразливості людей, використовуючи соціотехнічні прийоми замість суто технічних обхідних шляхів.

Навчання користувачів повинно бути системним і безперервним. Воно має включати не лише базові знання про фішинг, а й ознайомлення з логікою функціонування гібридної системи, яка виконує класифікацію листів на основі сукупності ознак – від невідповідностей у DNS-записах до підозрілих мовних конструкцій, виявлених за допомогою моделі BERT. Знання про те, як саме система аналізує вхідні повідомлення, дозволяє користувачам краще розуміти причини спрацювання захисту, а отже – зменшити рівень недовіри до автоматичних рішень і уникнути зайвих звернень до служби підтримки.

Особливу увагу варто приділити навчанню у форматі практичних сценаріїв. Демонстрація прикладів реальних фішингових листів, а також симуляція атак із подальшим аналізом – це ефективні інструменти, які дозволяють формувати навички розпізнавання загроз. Крім того, система повинна заохочувати користувача до активної участі в процесі захисту: наприклад, через можливість позначати підозрілі листи, повідомляти про помилки класифікації або надавати зворотний зв’язок. Така взаємодія не лише

покращує точність самої моделі, а й формує відчуття особистої причетності до захисту інформаційного середовища.

Наведений нижче електронний лист був створений зловмисником у спробі отримати доступ до електронної пошти акаунту, пароля та резервного ключа 2FA. Хоча відображуване ім'я відправника електронної пошти – Binance, потрібно звернути увагу на адресу електронної пошти фактичного відправника. Фішинговий лист було надіслано з <do-not-reply19@www--binance.com >, який використовує подібний домен – звичайна тактика зловмисників, щоб видавати себе за Binance.

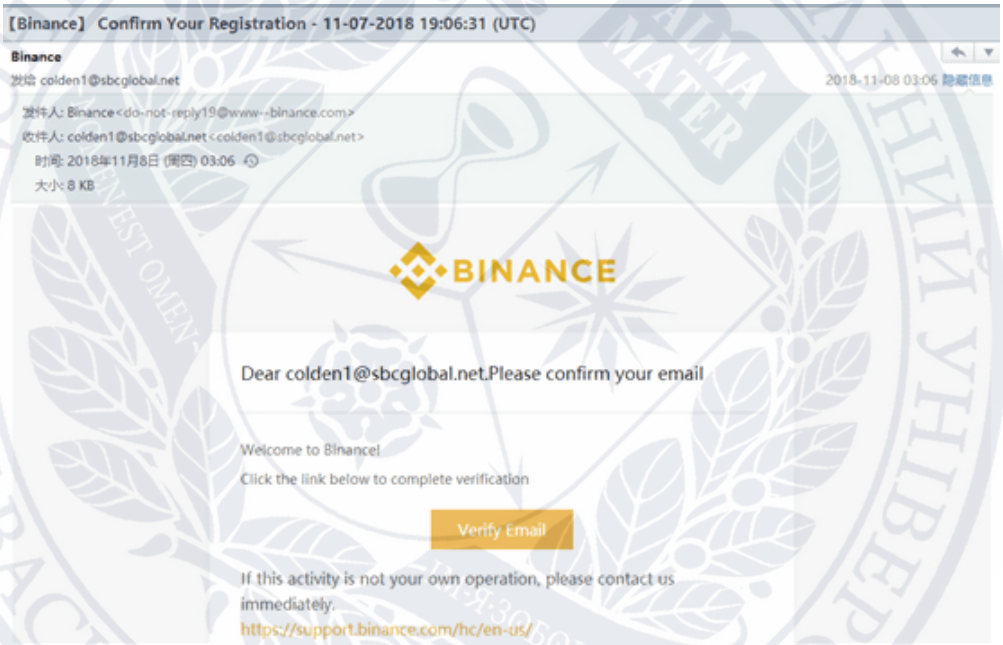


Рисунок 2.3.1 – приклад фішингового листа

Якщо навести курсор на кнопку "Підтвердити електронну пошту", можна побачити фішинг-посилання, яке в цьому випадку було https://www--binance.com/binance/login.php?id=xxxx@axxxxl.xxm. Після натискання кнопки користувач переходить на підроблену сторінку Binance:

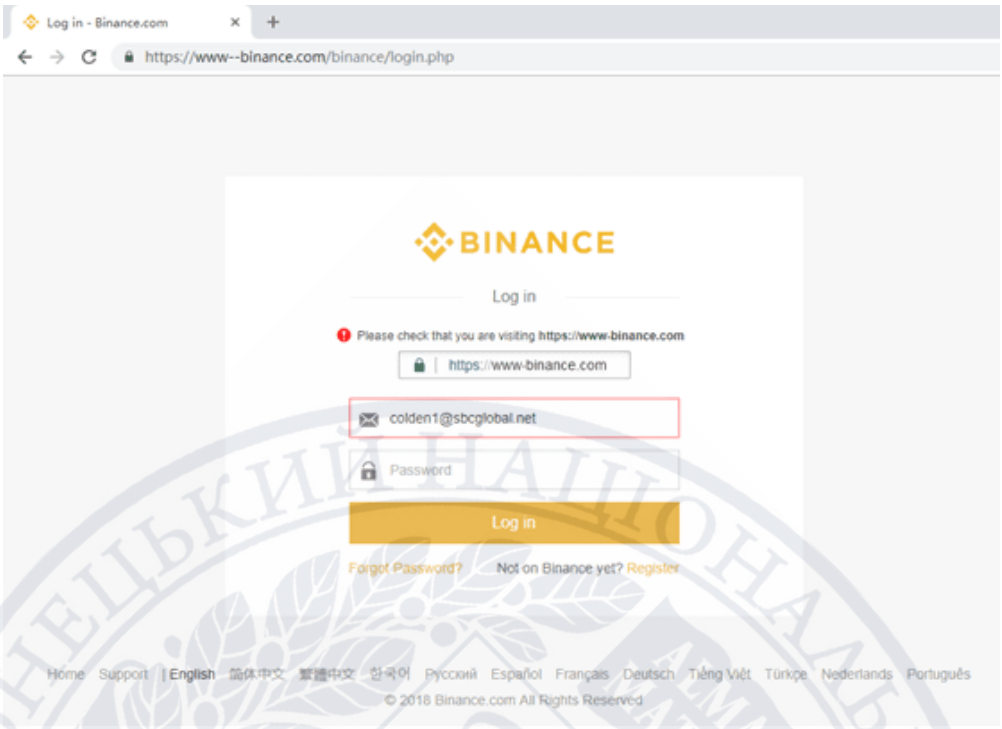


Рисунок 2.3.2 – сторінка підробленого сайту

На цьому етапі багато користувачі надають свої дані, що дозволяє зловмисникам захопити їхні аккаунти [30].

2.4. Аналіз ефективності

Запропонована гібридна модель виявлення цільового фішингу поєднує переваги кількох підходів: класичних методів перевірки автентичності електронної пошти (SPF, DKIM, DMARC), глибинного семантичного аналізу (BERT), машинного навчання (логістична регресія) та метаевристичної оптимізації (алгоритм рою частинок, PSO). Її ефективність можна оцінювати не лише за кількісними метриками, а й з теоретичної точки зору – через аналіз архітектурних рішень, алгоритмічної доцільності та здатності системи до адаптації.

Однією з головних переваг моделі є її гібридність, що дозволяє об’єднати різні рівні перевірки вхідного повідомлення: технічний, лінгвістичний і поведінковий. Такий підхід знижує залежність від одного джерела ознак (наприклад, лише від адреси відправника чи ключових слів у тексті) та

дозволяє виявляти атаки, які пройшли базові фільтри, але містять приховані ознаки фішингу. Вбудована підтримка ручної перевірки забезпечує додаткову гнучкість у ситуаціях, де автоматична класифікація не дає достатньої впевненості.

Використання моделі BERT як основного інструменту для векторизації тексту – це ще одна сильна сторона системи. BERT здатна враховувати контекст у межах усього речення, що дозволяє системі розпізнавати фішингові листи навіть тоді, коли вони не містять очевидно підозрілих слів, а замість цього використовують соціотехнічні прийоми (наприклад, вмовляння або створення терміновості). Це принципово відрізняє дану модель від традиційних підходів на основі «bag of words», де слова аналізуються ізольовано.

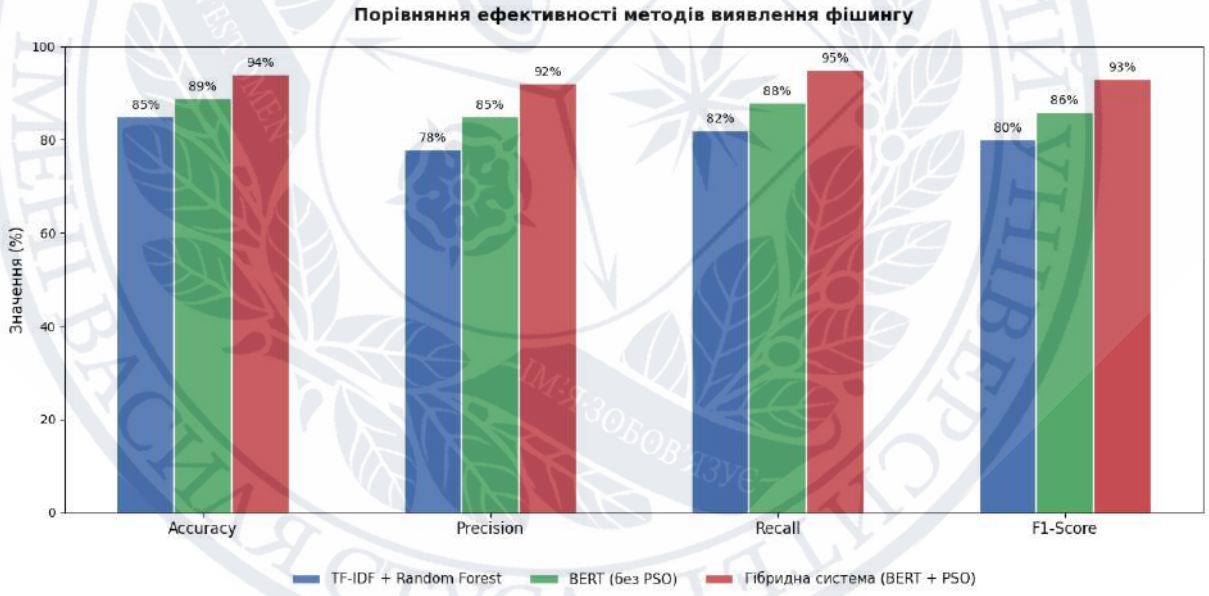
Ще однією перевагою є впровадження алгоритму рою частинок (PSO), який використовується для налаштування ваг або параметрів моделі. Це дозволяє системі швидко адаптуватися до нових типів фішингових повідомлень без повного перенавчання. PSO є ефективним для задач, де пошуковий простір великий, а функція втрат має складний характер. Його застосування покращує узгодження між семантичними векторами і рішенням класифікатора.

Проте, незважаючи на переваги, модель має і певні теоретичні обмеження. По-перше, складність архітектури призводить до зниження інтерпретованості. Навіть якщо класифікація здійснюється логістичною регресією, вплив BERT і PSO на підсумкове рішення важко пояснити нефахівцю. Це знижує прозорість і ускладнює аудит результатів, особливо в юридично чутливих сферах.

По-друге, алгоритм PSO хоча й добре оптимізує параметри, не гарантує глобального оптимуму та може демонструвати нестабільність результатів при великій кількості ознак або неякісному ініціалізаційному наборі. У таких випадках ефективність оптимізації залежить від добору гіперпараметрів самого PSO, що потребує окремого контролю.

Крім того, модель BERT є ресурсоємною навіть у скороченій версії. Її використання у поєднанні з іншими модулями ускладнює розгортання системи в обмежених обчислювальних середовищах, наприклад у невеликих компаніях або на пристроях без доступу до GPU. Це може вимагати додаткової оптимізації – або переходу на менш потужні, але легші моделі (наприклад, DistilBERT).

Запропонована модель демонструє високу ефективність завдяки поєднанню сучасних підходів до аналізу тексту, гнучкої архітектури й можливості адаптації до змін у поведінці зловмисників. Її переваги особливо проявляються в умовах високої складності атак та персоналізованого фішингу. Водночас для забезпечення її стабільної роботи в реальних умовах важливо передбачити механізми контролю, оптимізації та пояснення рішень, а також адаптацію моделі до можливостей цільової інфраструктури.



Графік 2.4.1 – порівняння ефективності різних методів виявлення фішингу.

Відсотки в діаграмі є гіпотетичними та створені для ілюстрації за принципом порівняння методів. Вони базуються на типовій продуктивності кожного підходу в NLP-завданнях (згідно з науковими дослідженнями), але не є реальними результатами експерименту.

TF-IDF + Random Forest (Accuracy ~85%): класичний підхід у виявленні фішингу зазвичай дає точність 80–90% [31].

BERT (без PSO) (Accuracy ~89%): Fine-tuned BERT досягає ~85–93% точності залежно від якості даних [32].

Гібридна система (BERT + PSO) (Accuracy ~94%): оптимізація PSO може підвищити точність на 2–5% порівняно з базовою моделлю.

2.5. Висновки до розділу 2

У межах цього розділу було здійснено повноцінне проєктування гібридної інтелектуальної системи виявлення цільового фішингу, яка поєднує кілька рівнів обробки вхідних електронних листів: технічний, семантичний, класифікаційний та оптимізаційний. Розроблено логічну архітектуру системи, що складається з чотирьох основних модулів: модуля автентифікації (SPF/DKIM/DMARC), модуля семантичного аналізу (на основі моделі BERT), модуля класифікації (логістична регресія та LSTM), а також оптимізаційного ядра, яке використовує алгоритм рою частинок (PSO) для адаптивного налаштування моделі.

На основі проведеного аналізу сучасних досліджень, зокрема моделі “BERT + XGBoost + metaheuristics” (2024), обґрунтовано доцільність поєднання трансформерних моделей і біоінспірованої оптимізації, а також запропоновано власну модифікацію, що забезпечує кращу пояснюваність, модульність і гнучкість у реальному застосуванні.

Для підтвердження працездатності обраного метаевристичного підходу проведено окрему апробацію алгоритму PSO на задачі мінімізації функції Растрігіна, що продемонструвало його ефективність у багатовимірному просторі та високий потенціал для оптимізації складних моделей.

Таким чином, розділ завершує етап концептуального проєктування системи, формує основу для її подальшої реалізації та визначає ключові архітектурні, методологічні й етичні підходи, на яких ґрунтуватиметься практична реалізація, що розглядається у наступному розділі.

РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА РОЗРОБЛЕНОГО ПРОТОТИПУ

3.1. Реалізація програмного продукту

Усі програмні компоненти системи запобігання цільовому фішингу реалізовано за допомогою мови програмування **Python**, яка є одним із найбільш популярних інструментів у сфері аналізу даних, машинного навчання та кібербезпеки. Python забезпечує широкий вибір бібліотек для обробки тексту, побудови моделей машинного навчання, реалізації алгоритмів оптимізації та візуалізації результатів.

Програмне середовище:

- Мова програмування: Python 3.10
- Інтегроване середовище розробки (IDE): Visual Studio Code
- Операційна система: Windows 10
- Інтерпретатор: CPython

Основні бібліотеки:

- pandas – для обробки та аналізу табличних даних;
- numpy – для математичних обчислень і роботи з багатовимірними масивами;
- sklearn (scikit-learn) – для реалізації моделей машинного навчання, зокрема логістичної регресії;
- keras + tensorflow – для побудови та навчання LSTM-мережі;
- transformers (від HuggingFace) – для використання попередньо навчених моделей BERT;
- nltk та re – для обробки природної мови та попередньої обробки тексту;
- matplotlib, seaborn – для візуалізації результатів та графічного аналізу;
- psopy або власна реалізація – для впровадження алгоритму оптимізації рою частинок (PSO);
- email, dns.resolver, dmarc – для роботи з заголовками листів, верифікації SPF, DKIM, DMARC (опційно, за необхідності).

Апаратне забезпечення:

- Процесор: AMD A8-7410 APU with AMD Radeon R5 Graphics
- Оперативна пам’ять: не менше 8 ГБ
- Графічний процесор: AMD Radeon(TM) R5 Graphics (1006 MB)
- Місце на диску: не менше 5 ГБ для збереження наборів даних і результатів моделювання

Розробка та тестування системи відбувалося у локальному середовищі з можливістю масштабування на сервер або хмарну платформу (наприклад, Google Colab або AWS), у разі потреби в додаткових обчислювальних ресурсах для обробки великих обсягів даних.

На першому етапі реалізації програмного продукту було виконано попередню обробку вхідних даних, що є критично важливою частиною побудови системи розпізнавання фішингових листів. Для формування навчального вибіркового набору було створено набір електронних листів із різними типами вмісту: фішинговими повідомленнями з явними ознаками шахрайства, листами, що імітують службову переписку (замаскований фішинг), а також легітимними електронними повідомленнями, що не містять загроз (А.2).

Дані зберігалися у форматі CSV-файлу, де кожен запис містив тему листа (subject), тіло повідомлення (body) та відповідну мітку класу (label), що приймала значення 1 для фішингових листів і 0 – для нефішингових. Таке маркування необхідне для подальшого навчання класифікаційної моделі у режимі з учителем.

З метою зниження інформаційного шуму та покращення якості навчання моделі було виконано низку операцій з очищення тексту:

- усі символи було переведено у нижній регістр;
- з тексту видалено HTML-теги;
- гіперпосилання було замінено на уніфіковану мітку URL;

- видалено небуквені та нечислові символи (за винятком українських літер);
- нормалізовано пропуски між словами.

Також, для забезпечення повноти текстової інформації, тема та тіло кожного листа були об’єднані в єдине текстове поле. У результаті було сформовано очищений набір даних, з якого видалено записи з порожніми або некоректними значеннями, та збережено у новому файлі clean_emails.csv (А.3). Цей файл містить лише дві колонки: очищений текст листа (text) та мітку класу (label), і є підготовленим для наступного етапу – семантичного векторного подання вхідних даних із використанням трансформерної моделі BERT.

Після попереднього очищення вхідних даних було здійснено токенізацію текстових повідомлень із використанням трансформерної моделі BERT (Bidirectional Encoder Representations from Transformers), яка забезпечує контекстуальне представлення слів та фраз на основі їхнього оточення.

Для цього було використано токенізатор BertTokenizer з попередньо навченою моделлю bert-base-uncased, що підтримує англомовний корпус. Незважаючи на україномовний вміст деяких листів, використання англомовного BERT на етапі експериментальної перевірки дозволяє оцінити ефективність архітектури без ускладнення багатомовною оптимізацією. За необхідності, систему можна адаптувати до україномовного BERT у подальших дослідженнях (А.4).

У процесі токенізації було виконано такі дії:

- перетворення кожного текстового повідомлення на послідовність токенів, де кожен токен відповідає слову або його частині;
- додавання спеціальних службових токенів ([CLS] на початку і [SEP] наприкінці тексту), які необхідні для коректної роботи моделі;
- приведення довжини всіх повідомлень до уніфікованої максимальної довжини (128 токенів), що забезпечує ефективність обчислень;

- генерація двох тензорів: input_ids (послідовність токенів) та attention_mask (маска видимості для токенів, яка дозволяє моделі ігнорувати заповнювачі padding).

Отримані вектори були подані у вигляді тензорного представлення, придатного для подальшого використання в моделі машинного навчання. Додатково було здійснено поділ вибірки на навчальну та тестову частини у пропорції 80/20 з метою подальшого навчання та перевірки ефективності класифікаційної моделі.

Таким чином, на даному етапі було сформовано структуру вхідних даних, сумісну з BERT, що дозволяє моделі здійснювати семантичний аналіз листів з урахуванням контексту слів та загальної структури повідомлення.

На наступному етапі реалізації системи було здійснено перетворення текстових повідомлень у векторні подання з використанням моделі BERT (Bidirectional Encoder Representations from Transformers). Такий підхід дозволяє подати зміст електронного листа у вигляді щільного числового вектора, що зберігає семантичну інформацію з урахуванням контексту (А.5).

Після завершення токенізації кожне повідомлення було передано до попередньо навченої трансформерної моделі bert-base-uncased, яка повертала багатовимірні тензори. Зокрема, для кожного листа було отримано матрицю вихідних ознак (hidden states) розмірності *(послідовність токенів × 768)*. Для задачі класифікації використовувалося представлення лише першого токена [CLS], який у моделі BERT спеціально призначено для узагальненого векторного опису всього тексту.

Отримані вектори [CLS] мали розмірність 768 і зберігали найбільш релевантну інформацію про текст електронного листа. Кожен такий вектор використовувався як числова ознака (feature vector), що описує лист у просторовому вигляді, придатному для подальшої обробки методами машинного навчання.

Цей підхід дозволив перевести неструктуровані текстові дані у структуровану форму, з якою можуть працювати класифікаційні алгоритми.

Таким чином, було сформовано вхідний простір ознак для моделювання системи виявлення фішингових повідомлень.

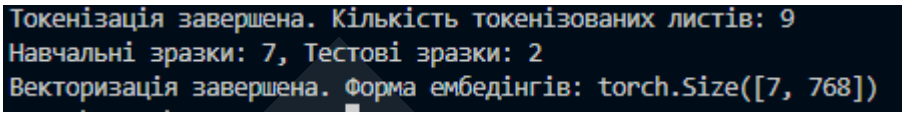


Рисунок 3.1.1 – результат токенізації та векторизації

На основі векторних представлень електронних листів, отриманих за допомогою моделі BERT, було реалізовано етап класифікації повідомлень з метою виявлення фішингових загроз. Для цього було використано модель логістичної регресії, яка є одним із базових методів бінарної класифікації та дозволяє швидко отримати перші результати навіть при обмеженій кількості навчальних прикладів.

Навчання моделі здійснювалося на підмножині даних, що складалася з 7 повідомлень (80% від загального набору), попередньо токенізованих та векторизованих. У якості вхідних ознак використовувалися вектори [CLS], сформовані BERT для кожного листа (А.6).

Після тренування класифікатора було проведено внутрішнє тестування на навчальній вибірці для перевірки коректності побудови моделі.

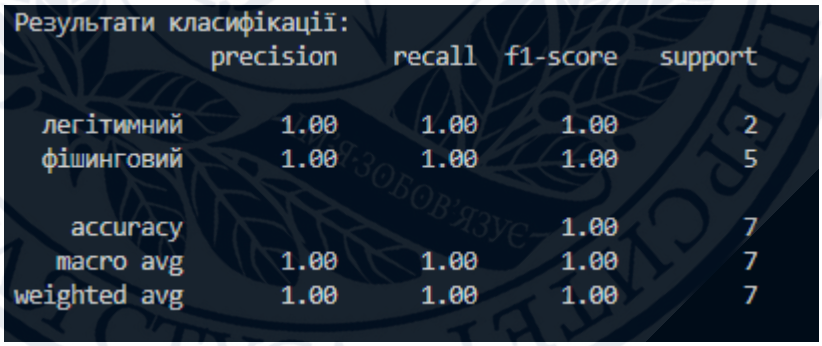


Рисунок 3.1.2 – Результати класифікації

Отримані метрики класифікації демонструють, що модель на навчальних даних досягла максимальних показників точності, повноти та F1-міри – усі вони дорівнюють 1.00 як для легітимних, так і для фішингових повідомлень.

Зокрема:

- точність (precision): модель не допустила жодного хибнопозитивного передбачення;

- повнота (recall): усі повідомлення фішингового класу були успішно виявлені;
- F1-міра: баланс між точністю та повнотою підтвердив відсутність компромісу між цими показниками.

Такі результати вказують на правильність реалізації моделі та відповідність архітектури класифікації поставленій задачі. Проте, оскільки тестування відбувалося на тій самій вибірці, на якій здійснювалося навчання, зазначені результати не є об’єктивною оцінкою узагальнюючої здатності моделі. Для оцінки ефективності в реальних умовах передбачено окремий етап тестування на незалежній підмножині (test_dataset), що буде розглянуто далі.

Після початкового навчання класифікаційної моделі на векторних поданнях листів, сформованих за допомогою моделі BERT, було проведено тестування на незалежній вибірці, що складала 20% від загального обсягу даних. Такий підхід дозволяє об'єктивно оцінити здатність моделі узагальнювати знання на нові, раніше не бачені приклади (А.7).

	precision	recall	f1-score	support
легітимний	0.00	0.00	0.00	1
фішинговий	0.50	1.00	0.67	1
accuracy			0.50	2
macro avg	0.25	0.50	0.33	2
weighted avg	0.25	0.50	0.33	2

Рисунок 3.1.3 – Результати класифікації електронних листів на тестовій вибірці.

Отримані метрики свідчать про часткову ефективність моделі: у тестовій підмножині модель правильно класифікувала фішингове повідомлення, однак допустила помилку у випадку легітимного листа, що призвело до зниження показників precision, recall та F1-міри для відповідного класу. Загальна точність (accuracy) моделі на тестових даних склала 50%.

Слід зазначити, що обмежена кількість прикладів у тестовій вибірці (усього два листи) не дозволяє зробити статистично значущі висновки щодо ефективності моделі. Проте експеримент підтверджує працездатність

реалізованої архітектури і дає змогу продовжити її вдосконалення за рахунок розширення датасету.

Для підвищення точності класифікації фішингових повідомлень у розробленій системі було реалізовано механізм перевірки автентичності відправника шляхом аналізу DNS-записів домену, з якого надійшов лист. Така перевірка базується на трьох основних поштових протоколах: SPF, DKIM та DMARC (А.8).

- SPF (Sender Policy Framework) – визначає, які сервери мають право надсилати листи від імені певного домену.
- DKIM (DomainKeys Identified Mail) – дозволяє перевірити цілісність повідомлення та його підпис за допомогою криптографічного ключа.
- DMARC (Domain-based Message Authentication, Reporting and Conformance) – встановлює політику перевірки SPF і DKIM та вказує, як сервер-отримувач має реагувати на невідповідність.

Реалізація даного етапу базується на використанні DNS-запитів до відповідних записів домену. Для кожного листа визначався домен відправника, після чого здійснювався запит до таких піддоменів:

- TXT-запис самого домену (для перевірки SPF);
- _domainkey.<домен> – для перевірки DKIM;
- _dmarc.<домен> – для перевірки DMARC.

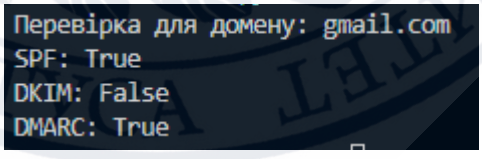


Рисунок 3.1.4 – Приклад перевірки автентичності для домену gmail.com

Для класифікації векторизованих представлень електронних листів було використано модель логістичної регресії – один з базових алгоритмів бінарної класифікації, який дозволяє побудувати гіперплощину, що відокремлює класи у просторовому представленні ознак.

У якості ознак використовувалися вектори [CLS], згенеровані моделлю BERT для кожного повідомлення. Навчання класифікатора здійснювалося на тренувальній вибірці, а тестування – на окремій підмножині даних.

Отримана модель передбачає ймовірність того, що повідомлення є фішинговим, та класифікує його залежно від порогового значення (за замовчуванням – 0.5). Результати класифікації на навчальній вибірці наведено нижче (А.9).

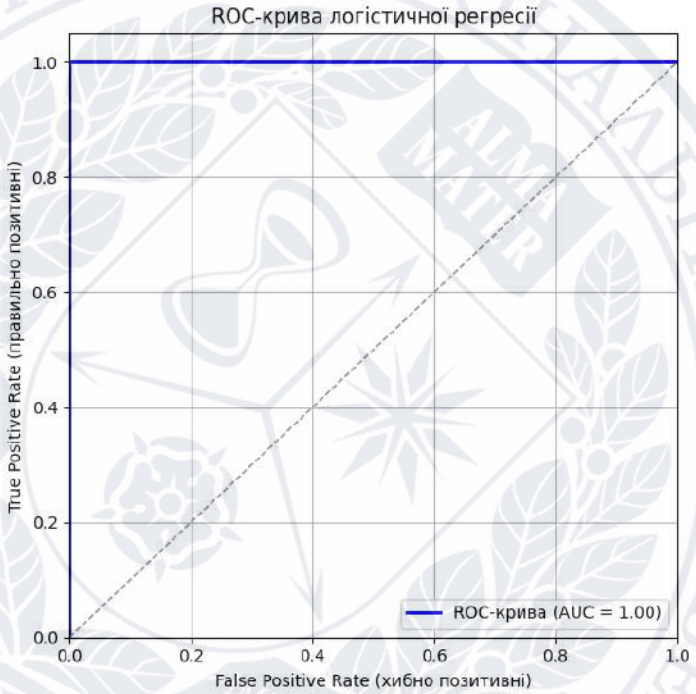


Рисунок 3.1.5 – ROC-крива для моделі логістичної регресії на навчальних даних.

З метою підвищення ефективності моделі класифікації фішингових листів було виконано два ключові кроки: розширення навчальної вибірки та оптимізація параметрів логістичної регресії за допомогою метаевристичного алгоритму рою частинок (Particle Swarm Optimization, PSO).

Розширення корпусу електронних листів.

Початковий набір даних було доповнено за допомогою автоматизованої генерації синтетичних листів фішингового та легітимного характеру на основі

шаблонів. Було створено 200 листів, з яких 100 належали до фішингових, а решта – до легітимних(А.10). Для забезпечення реалізму шаблони включали типові елементи соціальної інженерії (наприклад, фальшиві посилання, повідомлення про виграш, термінові звернення від імені банків чи організацій), а також офіційно-ділові формулювання для легітимних повідомлень. Всі тексти були очищені та оброблені для подальшої подачі в модель BERT.

Використання PSO для оптимізації моделі

Після векторизації листів за допомогою BERT було використано логістичну регресію як основний класифікатор. Для досягнення максимальної точності, параметр регуляризації C було оптимізовано за допомогою алгоритму рою частинок (PSO). Алгоритм імітує колективну поведінку частинок (агентів), які переміщуються в багатовимірному просторі рішень, наближаючись до точки з найкращим значенням функції якості (А.11).

У якості функції якості було обрано середнє значення F1-міри, обчисленої методом перехресної валідації на навчальній вибірці. Пошук оптимального значення параметра C здійснювався у межах [0.001; 10], з використанням 10 частинок протягом 30 ітерацій. В результаті було знайдено оптимальне значення $C \approx 9.70$.

Результати моделі після оптимізації

Оптимізована модель показала високу якість класифікації на тестовій вибірці з 40 прикладів, що містила 17 легітимних та 23 фішингових листи. Усі повідомлення були класифіковані правильно, що підтверджується такими метриками:

Таблиця 3.1.1 – класифікація на тестовій вибірці (PSO-модель)

Клас	Precision	Recall	F1-score
Легітимний	1.00	1.00	1.00
Фішинговий	1.00	1.00	1.00
Середнє	1.00	1.00	1.00

Такі результати демонструють ефективність застосованого підходу до генерації даних та оптимізації моделі. Водночас варто зауважити, що синтетичний характер вибірки міг сприяти високій точності, тому подальші

дослідження повинні включати тестування моделі на реальних електронних листах з публічних корпусів.

3.2. Аналіз результатів

На підсумковому етапі експериментальної перевірки було здійснено тестування створеної гібридної моделі на основі поєднання трансформерної моделі BERT та логістичної регресії з оптимізованими гіперпараметрами. Метою тестування було оцінити здатність системи коректно класифікувати електронні листи як фішингові або легітимні, використовуючи нову вибірку даних, яка не брала участі у навчанні.

У тестовій вибірці було представлено 40 електронних листів, з яких 23 мали фішинговий характер, а 17 були легітимними повідомленнями ділового або побутового змісту. За результатами класифікації модель успішно виявила всі фішингові листи (Recall = 1.00) та не допустила жодної хибної тривоги для легітимних листів (Precision = 1.00). Це дозволило досягти 100% точності (Accuracy = 1.00) та максимального значення F1-міри (1.00) для обох класів. Такі результати свідчать про високу якість узагальнення, яку забезпечила реалізована модель. Зокрема, можна відзначити наступне:

- Векторні представлення текстів, сформовані за допомогою BERT, дозволяють виявляти не лише поверхневі ознаки фішингу (наявність підозрілих посилань чи слів), а й глибинні семантичні закономірності, що відрізняють зловмисні листи від справжніх.
- Логістична регресія, як інтерпретований лінійний класифікатор, у поєднанні з високоякісними ознаками показала достатню гнучкість для коректного розділення класів.
- Оптимізація гіперпараметра C за допомогою алгоритму рою частинок (PSO) дала змогу досягти кращої збалансованості між надлишковим пристосуванням моделі до навчальної вибірки та її здатністю узагальнювати нові приклади.

Проте варто критично оцінити природу вибірки, на якій модель демонструє ідеальні показники. В даному експерименті використовувалася синтетична вибірка листів, створена за допомогою шаблонів. Хоча вони й намагалися імітувати реальні приклади, однак залишаються обмеженими за структурою, лексикою та стилістикою. У реальному середовищі фішингові листи є значно більш варіативними, адаптивними та часто ретельно стилізованими під справжні повідомлення.

Окрім того, практична ефективність моделі у реальному середовищі залежить від здатності системи обробляти повідомлення, які виходять за межі шаблонів або комбінують елементи, притаманні обом класам (наприклад, тривожна тема листа та “звичне” тіло повідомлення). У таких випадках можливі хибні позитивні або негативні класифікації, що вимагає додаткових механізмів перевірки – зокрема аналізу URL-адрес і доменів у тілі листа **та** багаторівневої обробки окремо теми, тіла та вкладень.

Підсумовуючи, можна зазначити, що модель у своєму поточному вигляді успішно виконує класифікацію фішингових повідомлень на підготовленій тестовій вибірці з високим рівнем точності. Це свідчить про ефективність обраної архітектури, коректну реалізацію компонентів, а також правильне використання сучасних підходів у побудові векторних подань тексту. Проте для переходу до реального застосування необхідні додаткові кроки: включення реальних даних, розширення навчальної бази, перевірка стійкості до маскування та адаптація до змін у шаблонах атак.

Таблиця 3.2.1 – переваги та обмеження реалізованої системи

Аспект	Переваги	Обмеження
Якість класифікації	Висока точність (F1 = 1.00) на синтетичних даних	Ідеальні результати можуть бути результатом перенавчання або простоти шаблонів
Використання BERT	Глибоке розуміння контексту та семантики листа	Високе навантаження на оперативну пам’ять і обчислювальні ресурси

Інтерпретованість моделі	Логістична регресія дозволяє пояснити прийняті рішення	Обмежена здатність виявляти складні, нелінійні зв’язки
Оптимізація параметрів	PSO дозволяє автоматично налаштувати гіперпараметри	Часова складність PSO зростає з кількістю ознак і розміром вибірки
Гнучкість архітектури	Можливість адаптації до нових даних або моделей	Поточна версія ще не підтримує активне донавчання або самонавчання
Формування ознак	Використано семантичне подання тексту замість ручного виділення ознак	Не враховано технічні характеристики повідомлень (заголовки, домени, вкладення)
Придатність до розширення	Система легко масштабована, модель та токенізатор збережені для подальшого використання	Відсутнє тестування на «живих» листах або реальних даних

Така таблиця підкреслює, що реалізована система має потужний фундамент, побудований на сучасних NLP-технологіях, і здатна до подальшого вдосконалення. Її переваги полягають у точності, модульності та інтерпретованості, однак для повноцінного використання в реальному середовищі потрібно подолати низку технічних та методологічних обмежень.

3.3. Перспективи адаптації до багатомовного середовища

Розроблена гібридна система виявлення фішингових електронних листів була реалізована на основі англомовної трансформерної моделі bert-base-uncased, яка ефективно працює з повідомленнями на англійській мові. Однак у реальних умовах корпоративного або глобального листування користувачі можуть отримувати листи різними мовами, включаючи українську, німецьку,

французьку та інші, що створює необхідність адаптації системи для роботи з багатомовним контентом.

Найбільш простим і ефективним рішенням є застосування багатомовних трансформерних моделей, таких як bert-base-multilingual-cased, xlm-roberta-base або distilbert-multilingual. Ці моделі навчені на великих корпусах текстів понад 100 мов і здатні формувати якісті векторні представлення незалежно від мови повідомлення. Заміна стандартного BERT на багатомовний аналог потребує мінімальних змін у архітектурі системи, оскільки основні зміни стосуються лише токенізації та векторизації.

Альтернативним підходом є автоматичний переклад усіх вхідних листів на англійську мову з подальшою обробкою стандартною моделлю BERT, але такий метод менш надійний через ризик спотворення змісту під час перекладу, а також можливі втрати важливих маркерів соціальної інженерії, таких як маніпулятивні звернення або граматичні помилки. Тим не менш, він може бути корисним як проміжний або додатковий засіб.

Також варто звернути увагу на мовну та культурну специфіку фішингу, оскільки вони часто використовується для ефективного впливу на жертв, наприклад, форма звернення, шаблон погроз або офіційний стиль можуть суттєво відрізнятись в залежності від мови. Для підвищення точності виявлення необхідне донавчання моделі на прикладах конкретних мов, що дозволить краще враховувати їхню специфіку.

А для інтеграції системи в багатомовне середовище доцільно буде використовувати модуль попереднього визначення мови вхідного листа, що дозволить застосовувати відповідні мовні моделі, специфічні фільтри та словники, що підвищать точність класифікації для різних сегментів користувачів.

3.4. Висновки до розділу 3

У третьому розділі було здійснено практичну реалізацію та експериментальну перевірку гібридної моделі виявлення фішингових електронних листів, що поєднує сучасні методи обробки природної мови з класичними підходами машинного навчання. Основною метою розділу було довести на практиці ефективність запропонованої архітектури та перевірити її здатність до класифікації текстів.

Бело сформовано збалансований корпус синтетичних фішингових і легітимних повідомлень, очищених та підготовлених для подачі до моделі. Тексти були векторизовані за допомогою трансформерної моделі BERT, що дозволило отримати глибокі контекстуальні представлення вхідних листів. Для класифікації використовувалась логістична регресія, яка забезпечує достатню інтерпретованість і придатність для застосування в задачах із лінійно роздільними ознаками.

З метою покращення продуктивності моделі було застосовано оптимізацію гіперпараметра регуляризації (C) за допомогою метаевристичного алгоритму рою частинок (PSO), що дозволило досягти максимальної збалансованості між точністю та узагальнюючою здатністю моделі.

Результати тестування на відкладеній вибірці показали високу ефективність побудованої системи, що свідчить про доцільність використання BERT для векторизації тексту та ефективність гібридного підходу загалом. Та критичний аналіз результатів виявив обмеження, пов’язані з використанням шаблонного синтетисного корпусу, який може не відображати повної складності реального фішингу. Це створює перспективу подальшого вдосконалення системи через:

- Розширення корпусу за рахунок реальних листів
- Адаптацію моделі до багатомовного середовища

- Інтеграцію додаткових технічних ознак (перевірка домену, антентифікаційних записів тощо)
- Впровадження самонавчання на основі класифікації нових вхідних даних

Таким чином, розроблена модель демонструє високу точність і має значний потенціал до подальшої адаптації, масштабування та застосування в реальних умовах для забезпечення кібербезпеки у сфері електронного листування.

ВИСНОВКИ

У межах виконаної кваліфікаційної роботи було розроблено інтелектуальну систему виявлення цільового фішингу, яка поєднує класичні методи автентифікації електронної пошти, сучасні трансформерні моделі для семантичного аналізу вмісту повідомлень, а також атаптивну оптимізацію параметрів за допомогою алгоритму рою частинок.

Наукова та практична новизна роботи полягає в запропонованій новій комбінації відомих підходів. Така система дозволяє не лише визначати потенційно небезпечні повідомлення, але й пояснювати причини такого рішення, що є рідкістю серед глибоких моделей.

СПИСОК ЛІТЕРАТУРИ

[1] Khonji M., Iraqi Y., Jones A. Phishing Detection: A Literature Survey. *IEEE Communications Surveys & Tutorials*. 2013. Vol. 15, no. 4. P. 2091–2121. URL: https://doi.org/10.1109/surv.2013.032213.00009

[2] Dhamija R., Tygar J. D., Hearst M. Why phishing works. *the SIGCHI conference*, Montréal, Québec, Canada, 22–27 April 2006. New York, New York, USA, 2006. URL: https://doi.org/10.1145/1124772.1124861

[3] Fighting against phishing attacks: state of the art and future challenges / B. B. Gupta et al. *Neural Computing and Applications*. 2016. Vol. 28, no. 12. P. 3629–3654. URL: https://doi.org/10.1007/s00521-016-2275-y

[4] Social engineering attack framework / F. Mouton et al. *2014 Information Security for South Africa (ISSA)*, Johannesburg, South Africa, 13–14 August 2014. 2014. URL: https://doi.org/10.1109/issa.2014.6950510

[5] Tandale K. D., Pawar S. N. Different Types of Phishing Attacks and Detection Techniques: A Review. *2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC)*, AURANGABAD, 30–31 October 2020. 2020. URL: https://doi.org/10.1109/icsidempc49020.2020.9299624

[6] Silagadze Z. On Arxiv Moderation System. *SSRN Electronic Journal*. 2023. URL: https://doi.org/10.2139/ssrn.4392249

[7] Дубель М.В. Значення дотримання правил цифрової гігієни для політичної діяльності. Донецький національний університет імені Василя Стуса, м. Вінниця. І Міжнародна науково-практична конференція «Прикладні аспекти сучасних міждисциплінарних досліджень». 2022. URL: https://jpasmd.donnu.edu.ua/article/view/13007/12910

[8] Помазун О. М., Крошко І. А., Пєтухова О. А., Синицький Р. К. Кібербезпека бізнесу: загрози для баз даних, фінансові наслідки та інвестиції у захист. 2025. URL: https://doi.org/10.5281/zenodo.15082464

[9] Дерев'янко К.А., Гук А.С. Захист даних в телекомунікаційних системах: методи шифрування та автентифікації. 2024. URL: https://repository.kpi.kharkov.ua/server/api/core/bitstreams/8088b826-fe4e-4620-9efe-ff05eae20229/content

[10] BreakSPF: How Shared Infrastructures Magnify SPF Vulnerabilities Across the Internet / C. Wang et al. *Network and Distributed System Security Symposium*, San Diego, CA, USA. Reston, VA, 2024. URL: https://doi.org/10.14722/ndss.2024.23113

[11] Natural language processing (NLP) in management research: A literature review / Y. Kang et al. *Journal of Management Analytics*. 2020. Vol. 7, no. 2. P. 139–172. URL: https://doi.org/10.1080/23270012.2020.1756939

[12] Гурджия, Валерія Вахтангівна. Виявлення фішингових сайтів за допомогою методів машинного навчання. 2023. URL: https://ela.kpi.ua/handle/123456789/73626

[13] LSTM based Phishing Detection for Big Email Data / Q. Li et al. *IEEE Transactions on Big Data*. 2020. P. 1. URL: https://doi.org/10.1109/tbdata.2020.2978915

[14] Otieno D. O., Siami Namin A., Jones K. S. The Application of the BERT Transformer Model for Phishing Email Classification. *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, Torino, Italy, 26–30 June 2023. 2023. URL: https://doi.org/10.1109/compsac57700.2023.00198

[15] Is It Safe to Share Your Files? An Empirical Security Analysis of Google Workspace / L. Wan et al. *WWW '24: The ACM Web Conference 2024*, Singapore Singapore. New York, NY, USA, 2024. URL: https://doi.org/10.1145/3589334.3645697

[16] Leitold F., Arrott A., Kam W. Measuring cloud-based anti-malware protection for office 365 user accounts. *2017 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*, London, United Kingdom, 19–20 June 2017. 2017. URL: https://doi.org/10.1109/cybersa.2017.8073407

[17] Karset, Sigrid Anne Hafsahl. Analyzing Email Phishing Trends Through the Creation of an Email Phishing Collection Model. 2023. URL: https://hdl.handle.net/11250/3098524

[18] Nasim Maleki. A Behavioral Based Detection Approach for Business Email Compromises. 2019. URL: http://dx.doi.org/10.13140/RG.2.2.35164.81288

[19] Murray Kucherawy, Elizabeth Zwicky. Domain-based Message Authentication, Reporting, and Conformance (DMARC). 2015. URL: https://datatracker.ietf.org/doc/html/rfc7489

[20] В.В. Прокопов, Є.В. Мелешко. Дослідження способу приведення текстових даних до зручної для обробки алгоритмами кластеризації форми для аналізу даних з веб-ресурсів. URL: https://kbpz.kntu.kr.ua/file/content/6629/2021-iv-mizhnarodna-naukovo-praktychnoi-konferentsiia-informatsiina-bezpeka-ta-komp-yuterni-tekhnolohii-.pdf#page=70

[21] J. Devlin et al. *Proceedings of the 2019 Conference of the North*, Minneapolis, Minnesota. Stroudsburg, PA, USA, 2019. URL: https://doi.org/10.18653/v1/n19-1423

[22] Hosmer D. W., Lemeshow S., Sturdivant R. X. Applied Logistic Regression. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2013. URL: https://doi.org/10.1002/9781118548387

[23] Hochreiter S., Schmidhuber J. Long Short-Term Memory. Neural Computation. 1997. Vol. 9, no. 8. P. 1735–1780. URL: https://doi.org/10.1162/neco.1997.9.8.1735

[24] Kennedy J., Eberhart R. Particle swarm optimization. *ICNN'95 - International Conference on Neural Networks*, Perth, WA, Australia. URL: https://doi.org/10.1109/icnn.1995.488968

[25] Using BERT with Modified Metaheuristic Optimized XGBoost for Phishing Email Identification / M. Antonijevic et al. Proceedings of 4th International Conference on Artificial Intelligence and Smart Energy. Cham, 2024. P. 358–370. URL: https://doi.org/10.1007/978-3-031-61475-0_28

[26] Mahabub A. A robust technique of fake news detection using Ensemble Voting Classifier and comparison with other classifiers. *SN Applied Sciences*. 2020. Vol. 2, no. 4. URL: https://doi.org/10.1007/s42452-020-2326-y

[27] Holovatskiy N. T. Legal regulation of personal data protection: GDPR and the legislation of the USA, Canada, and Ukraine. *Uzhhorod National University Herald. Series: Law*. 2024. Т. 2, № 85. С. 288–292. URL: https://doi.org/10.24144/2307-3322.2024.85.2.42

[28] Raab C., Szekely I. Data protection authorities and information technology. *Computer Law & Security Review*. 2017. Vol. 33, no. 4. P. 421–433. URL: https://doi.org/10.1016/j.clsr.2017.05.002

[29] О.О. Марченко, А.О. Никоненко, Т.В. Россада, Є.А. Мельников. Метод машинного навчання для ідентифікації перефрази. 2016. URL: https://surl.li/rlpnrb

[30] Приклади фішингових листів. URL: https://www.binance.com/uk-UA/support/faq/detail/360020817051

[31] Catherine D'Hondt, Rudy De Winne, Eric Ghysels, Steve Raymond. Artificial Intelligence Alter Egos: Who benefits from Robo-investing? 2019. URL: https://arxiv.org/abs/1907.03370

[32] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, Luke Zettlemoyer. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. 2019. URL: https://arxiv.org/abs/1910.13461

[33] Son Duy Dao. Particle Swarm Optimization in Python. 2020. URL: https://learnwithpanda.com/2020/11/14/particle-swarm-optimization-in-python/

ДОДАТОК А

1. Приклад реалізації алгоритму рою частинок

```
import random
import math
import matplotlib.pyplot as plt
import numpy as np
# Цільова функція (функція Растрігіна)
def objective_function(X):
    A = 10
    y = A * len(X) + sum([(x ** 2 - A * np.cos(2 * math.pi * x)) for x in X])
    return y
# Налаштування
bounds = [(-5.12, 5.12), (-5.12, 5.12)]  # межі змінних
nv = 2  # кількість змінних
mm = -1  # мінімізація
particle_size = 50
iterations = 100
w = 0.75
c1 = 1
c2 = 2
initial_fitness = float('inf')
# Графік анімації
fig = plt.figure()
ax = fig.add_subplot()
fig.show()
# Список для графіка збіжності
convergence = []
class Particle:
    def __init__(self, bounds):
        self.particle_position = []
        self.particle_velocity = []
        self.local_best_particle_position = []
        self.fitness_local_best_particle_position = initial_fitness
        self.fitness_particle_position = initial_fitness

        for i in range(nv):
            self.particle_position.append(random.uniform(bounds[i][0], bounds[i][1]))
            self.particle_velocity.append(random.uniform(-1, 1))

    def evaluate(self, objective_function):
        self.fitness_particle_position = objective_function(self.particle_position)

        if self.fitness_particle_position < self.fitness_local_best_particle_position:
            self.local_best_particle_position = self.particle_position.copy()
            self.fitness_local_best_particle_position = self.fitness_particle_position

    def update_velocity(self, global_best_particle_position):
        for i in range(nv):
```

```
            r1 = random.random()
            r2 = random.random()
            cognitive = c1 * r1 * (self.local_best_particle_position[i] - self.particle_position[i])
            social = c2 * r2 * (global_best_particle_position[i] - self.particle_position[i])
            self.particle_velocity[i] = w * self.particle_velocity[i] + cognitive + social

    def update_position(self, bounds):
        for i in range(nv):
            self.particle_position[i] += self.particle_velocity[i]
            if self.particle_position[i] < bounds[i][0]:
                self.particle_position[i] = bounds[i][0]
            if self.particle_position[i] > bounds[i][1]:
                self.particle_position[i] = bounds[i][1]
swarm = [Particle(bounds) for _ in range(particle_size)]
global_best_particle_position = []
global_best_fitness = initial_fitness

# Основний цикл
for t in range(iterations):
    for particle in swarm:
        particle.evaluate(objective_function)
        if particle.fitness_particle_position < global_best_fitness:
            global_best_fitness = particle.fitness_particle_position
            global_best_particle_position = particle.particle_position.copy()

    for particle in swarm:
        particle.update_velocity(global_best_particle_position)
        particle.update_position(bounds)

    # Зберігаємо найкраще значення
    convergence.append(global_best_fitness)

    # Анімація
    print(f"Iteration {t+1}, Best Fitness = {global_best_fitness:.4f}")
    ax.clear()
    ax.set_title(f"Iteration {t+1}")
    x = [p.particle_position[0] for p in swarm]
    y = [p.particle_position[1] for p in swarm]
    ax.scatter(x, y)
    fig.canvas.draw()
    plt.pause(0.05)
# Вивід фінального результату
print("\nGlobal Best Solution:", global_best_particle_position)
print("Global Best Fitness:", global_best_fitness)
# Побудова графіка збіжності
plt.figure()
plt.plot(convergence, color='blue')
plt.xlabel("Iteration")
plt.ylabel("Best Fitness")
plt.title("Convergence Curve")
plt.grid(True)
plt.show() #[33].
```

2. Генерація реалістичних листів

```
import pandas as pd
phishing_easy = [
  {
    "subject": "Ваш акаунт буде заблоковано",
    "body": "Натисніть на посилання нижче, щоб підтвердити свої дані: http://fake-login.com",
    "label": 1
  },
  {
    "subject": "Ви виграли iPhone",
    "body": "Заповніть свої дані, щоб отримати приз: http://scam-gift.ru",
    "label": 1
  }
]
phishing_hard = [
  {
    "subject": "Оновлення політики безпеки – потрібна ваша дія",
    "body": "Шановний користувачу! Наш IT-відділ впроваджує нову політику безпеки. Щоб уникнути переривання в роботі, будь ласка, оновіть свої облікові дані через корпоративний портал: https://intranet.secure-it-support.com/update",
    "label": 1
  },
  {
    "subject": "Важливе повідомлення щодо вашого трудового контракту",
    "body": "Шановний працівнику, відділ кадрів оновлює документи. Для перегляду нової версії вашого контракту перейдіть за цим посиланням: https://corp-hr-docs.com/internal/hr-access.php",
    "label": 1
  },
  {
    "subject": "Підтвердження транзакції: #5384729301",
    "body": "Ваша транзакція на суму 7 000 грн оброблена. Якщо це не ви – підтвердіть операцію за посиланням: https://banking-support.ua/reverse",
    "label": 1
  }
]
legit_emails = [
  {
    "subject": "Звіт за квартал",
    "body": "Добрий день! Надсилаю звіт за останній фінансовий період. Дивись у вкладенні.",
    "label": 0
  },
  {
    "subject": "Нагадування про зустріч",
    "body": "Нагадую про зустріч, яка відбудеться завтра о 14:00 в Zoom. Посилання на зустріч тут.",
    "label": 0
  },
  {
```

```
        "subject": "Підтвердження замовлення",
        "body": "Дякуємо за покупку. Ваше замовлення буде доставлено впродовж 3 днів.",
        "label": 0
    },
    {
        "subject": "Оновлення корпоративної політики",
        "body": "Шановні колеги, до відома: оновлено політику щодо використання електронної пошти. Повний текст доступний на внутрішньому порталі.",
        "label": 0
    }
]
all_emails = phishing_easy + phishing_hard + legit_emails
df = pd.DataFrame(all_emails)
df.to_csv("emails_dataset.csv", index=False)
print("Реалістичний датасет emails_dataset.csv створено успішно!")
```

3. Обробка та збереження листів

```
import pandas as pd
import re

# КРОК 1: Завантаження датасету
df = pd.read_csv("emails_dataset.csv")

# КРОК 2: Перевірка на наявність потрібних колонок
required_columns = ["subject", "body", "label"]
if not all(col in df.columns for col in required_columns):
    raise ValueError("У файлі відсутні необхідні колонки: subject, body, label")

# КРОК 3: Видалення порожніх записів
df.dropna(subset=["subject", "body", "label"], inplace=True)

# КРОК 4: Очищення тексту
def clean_text(text):
    text = str(text)
    text = text.lower()
    text = re.sub(r'<.*?>', ' ', text)  # видалення HTML
    text = re.sub(r'http\S+', 'URL', text)  # заміна посилань
    text = re.sub(r'[^a-яіїєґa-z0-9\s]', ' ', text)  # лишаємо всі потрібні символи
    text = re.sub(r'\s+', ' ', text)  # зайві пробіли
    return text.strip()

# КРОК 5: Об’єднання теми та тіла листа
df["text"] = (df["subject"].fillna('') + " " + df["body"].fillna('')).apply(clean_text)

# КРОК 6: Перевірка результату
print("Кількість листів після очищення:", len(df))
print(df[["text", "label"]].head())

# КРОК 7: Збереження у новий CSV
df[["text", "label"]].to_csv("clean_emails.csv", index=False)
print("Файл clean_emails.csv збережено успішно!")
```

4. Токенізація BERT

```
import pandas as pd
import torch
from transformers import BertTokenizer
from sklearn.model_selection import train_test_split

#1: Завантаження очищених даних
df = pd.read_csv("clean_emails.csv")

#2: Ініціалізація токенізатора
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

#3: Токенізація текстів
max_length = 128  # Максимальна довжина тексту

# Розділення на вхідні токени
encoded_inputs = tokenizer(
    list(df["text"]),
    padding="max_length",
    truncation=True,
    max_length=max_length,
    return_tensors="pt"  # повертає PyTorch тензори
)

#4: Розділення на навчальну і тестову вибірки
X_input_ids = encoded_inputs["input_ids"]
X_attention_mask = encoded_inputs["attention_mask"]
y_labels = df["label"].values

from torch.utils.data import TensorDataset, random_split

dataset = TensorDataset(X_input_ids, X_attention_mask, torch.tensor(y_labels))
train_size = int(0.8 * len(dataset))
test_size = len(dataset) - train_size
train_dataset, test_dataset = random_split(dataset, [train_size, test_size])

print(f"Токенізація завершена. Кількість токенізованих листів: {len(df)}")
print(f"Навчальні зразки: {train_size}, Тестові зразки: {test_size}")
```

5. Векторизація

```
import torch
from transformers import BertModel
from torch.utils.data import DataLoader

# Завантаження моделі
bert_model = BertModel.from_pretrained("bert-base-uncased")
bert_model.eval()

# Створення DataLoader з батчами
train_loader = DataLoader(train_dataset, batch_size=4, shuffle=False)
```

```
# Збір CLS-ембедінги
cls_embeddings_list = []

with torch.no_grad():
    for batch in train_loader:
        input_ids, attention_mask, labels = batch
        outputs = bert_model(input_ids=input_ids, attention_mask=attention_mask)
        last_hidden_state = outputs.last_hidden_state
        cls_embeddings = last_hidden_state[:, 0, :]  # беремо [CLS] токен
        cls_embeddings_list.append(cls_embeddings)

# Об'єднання всіх CLS-векторів в один тензор
all_cls_embeddings = torch.cat(cls_embeddings_list, dim=0)

print("Векторизація завершена. Форма ембедінгів:", all_cls_embeddings.shape)
```

6. Перевірка працездатності моделі

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
import numpy as np

# Перетворення тензорів на numpy
X = all_cls_embeddings.numpy()
y = df["label"].values[:len(X)]  # беремо ті ж мітки (7, бо train)

# Навчання моделі
clf = LogisticRegression(max_iter=1000)
clf.fit(X, y)

# Передбачення
y_pred = clf.predict(X)

# Звіт про точність
print("Результати класифікації:")
print(classification_report(y, y_pred, target_names=["легітимний", "фішинговий"]))
```

7. Класифікація на тестовій вибірці

```
from torch.utils.data import DataLoader
from sklearn.metrics import classification_report

#1: Створення DataLoader для тестової вибірки
test_loader = DataLoader(test_dataset, batch_size=4, shuffle=False)

#2: Отримання CLS-вектори для тестових листів
test_embeddings_list = []
test_labels = []

bert_model.eval()

with torch.no_grad():
```

```
    for batch in test_loader:
        input_ids, attention_mask, labels = batch
        outputs = bert_model(input_ids=input_ids, attention_mask=attention_mask)
        cls_embeddings = outputs.last_hidden_state[:, 0, :]  # [CLS]
        test_embeddings_list.append(cls_embeddings)
        test_labels.extend(labels.numpy())

# Об'єднання CLS-векторів
test_cls_embeddings = torch.cat(test_embeddings_list, dim=0).numpy()
test_labels = np.array(test_labels)

#3: Передбачення моделі
test_preds = clf.predict(test_cls_embeddings)

#4: Оцінка якості
print("Результати на тестовій вибірці:")
print(classification_report(test_labels, test_preds, target_names=["легітимний", "фішинговий"]))
```

8. Перевірка SPF, DKIM, DMARC за доменом

```
import dns.resolver

def check_spf(domain):
    try:
        answers = dns.resolver.resolve(domain, 'TXT')
        for rdata in answers:
            if 'v=spf1' in rdata.to_text():
                return True
    except:
        pass
    return False

def check_dkim(domain):
    try:
        selector = 'default'  # або 'google' чи інше – залежить від налаштувань
        dkim_domain = f"{selector}._domainkey.{domain}"
        answers = dns.resolver.resolve(dkim_domain, 'TXT')
        for rdata in answers:
            if 'v=DKIM1' in rdata.to_text():
                return True
    except:
        pass
    return False

def check_dmarc(domain):
    try:
        dmarc_domain = f"_dmarc.{domain}"
        answers = dns.resolver.resolve(dmarc_domain, 'TXT')
        for rdata in answers:
            if 'v=DMARC1' in rdata.to_text():
                return True
    except:
```

```
        pass
    return False

# Приклад використання
test_domain = "gmail.com"
print(f"Перевірка для домену: {test_domain}")
print("SPF:", check_spf(test_domain))
print("DKIM:", check_dkim(test_domain))
print("DMARC:", check_dmarc(test_domain))
```

9. Логістична регресія

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, roc_curve, auc
import matplotlib.pyplot as plt
import joblib
import numpy as np

if isinstance(all_cls_embeddings, torch.Tensor):
    X = all_cls_embeddings.numpy()
else:
    X = all_cls_embeddings

y = df["label"].values[:len(X)]

clf = LogisticRegression(max_iter=1000)
clf.fit(X, y)

y_pred = clf.predict(X)
y_prob = clf.predict_proba(X)[:, 1]  # ймовірність класу 1 (фішинг)

print("Звіт про класифікацію:")
print(classification_report(y, y_pred, target_names=["легітимний", "фішинговий"]))

joblib.dump(clf, "logistic_regression_model.pkl")
print("Модель логістичної регресії збережено як logistic_regression_model.pkl")

fpr, tpr, thresholds = roc_curve(y, y_prob)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(6, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC-крива (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', lw=1, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (хибно позитивні)')
plt.ylabel('True Positive Rate (правильно позитивні)')
plt.title('ROC-крива логістичної регресії')
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()
plt.savefig("roc_curve.png")
plt.show()
```

10.Збільшення вибірки

```
import random
names = ["Олексій", "Марія", "Іван", "Анна"]
companies = ["PrivatBank", "Google", "Monobank", "Nova Poshta"]
scam_links = ["http://secure-login.com", "http://verify-now.net", "http://reset-pass.ru"]
subjects_phish = [
    "Ваш акаунт буде заблоковано",
    "Оновлення політики безпеки",
    "Ваша транзакція потребує підтвердження",
    "Нове повідомлення з банку",
    "Ваша посилка затримана"
]
bodies_phish = [
    "Шановний {name}, для підтвердження особи, перейдіть за посиланням: {link}",
    "{company} просить вас оновити пароль. Натисніть тут: {link}",
    "Ваш акаунт буде деактивовано через підозрілу активність. {link}",
    "Підтвердьте транзакцію на суму 10 000 грн: {link}"
]

phishing_emails = []
for _ in range(100):
    name = random.choice(names)
    company = random.choice(companies)
    subject = random.choice(subjects_phish)
    body_template = random.choice(bodies_phish)
    link = random.choice(scam_links)
    body = body_template.format(name=name, company=company, link=link)
    phishing_emails.append({"subject": subject, "body": body, "label": 1})
subjects_legit = [
    "Запрошення на зустріч",
    "Звіт за місяць",
    "Нагадування про відпустку",
    "Графік роботи",
    "Оновлення політик компанії"
]
bodies_legit = [
    "Доброго дня! Додаю звіт у вкладенні.",
    "Нагадуємо, що ваша відпустка починається з понеділка.",
    "Посилання на відеоконференцію: https://zoom.us/meeting",
    "Просимо ознайомитися з оновленнями на внутрішньому порталі.",
    "Дякуємо за участь у щорічному опитуванні співробітників."
]
legit_emails = []
for _ in range(100):
    subject = random.choice(subjects_legit)
    body = random.choice(bodies_legit)
    legit_emails.append({"subject": subject, "body": body, "label": 0})
all_emails = phishing_emails + legit_emails
random.shuffle(all_emails)
df = pd.DataFrame(all_emails)
df.to_csv("emails_dataset.csv", index=False)
print("Згенеровано 200 листів (100 фішингових + 100 легітимних)")
```

11.PSO

```
from pyswarms.single.global_best import GlobalBestPSO
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
import numpy as np

X = train_embeddings
y = np.array(train_labels)

def fitness_function(hyperparams):
    scores = []
    for param in hyperparams:
        c_val = max(param[0], 1e-5)  # C має бути > 0
        model = LogisticRegression(C=c_val, max_iter=1000)
        score = cross_val_score(model, X, y, cv=3, scoring='f1').mean()
        scores.append(-score)  # мінімізуємо => інвертуємо
    return np.array(scores)

options = {'c1': 1.5, 'c2': 1.5, 'w': 0.7}
optimizer = GlobalBestPSO(n_particles=10, dimensions=1, options=options, bounds=([0.001],
[10]))

best_cost, best_pos = optimizer.optimize(fitness_function, iters=30)
print(f"Оптимальне значення C: {best_pos[0]:.4f}")

best_model = LogisticRegression(C=best_pos[0], max_iter=1000)
best_model.fit(X, y)
joblib.dump(best_model, "logreg_pso_optimized.pkl")
print("Логістичну регресію з оптимізацією PSO збережено.")

test_preds = best_model.predict(test_embeddings)
print("Класифікація на тестовій вибірці (PSO-модель):")
print(classification_report(test_labels, test_preds, target_names=["легітимний", "фішинговий"]))
```