НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

До захисту допущено
Завідувач кафедри

_____ Дмитро ЛАНДЕ
(підпис)

“___”_____2025 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи, технології та математичні методи кібербезпеки»

зі спеціальності 125 «Кібербезпека»

на тему: Використання методів машинного навчання при фільтрації фішингових повідомлень

Виконав(ла) здобувач вищої освіти **IV** курсу, групи ФБ-11
(шифр групи)

Маленко Сергій Сергійович
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник: старший викладач кафедри ІБ Рибак О.В.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент: доцент Іван Терещенко
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Здобувач вищої освіти _____
(підпис)

Київ – 2025 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 125 «Кібербезпека»
Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Дмитро ЛАНДЕ
(підпис)
«_____»_____2025 р.

ЗАВДАННЯ
на дипломну роботу здобувачу вищої освіти

Маленко Сергію Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Використання методів машинного навчання
при фільтрації фішингових повідомлень»

Науковий керівник роботи: Рибак Олександр Владиславович, старший викладач кафедри ІБ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «26» травня 2025 р. № 1761-с

2. Термін подання здобувачем дипломної роботи «____» червня 2025 р.

3. Вихідні дані до роботи: Прототип комбінованої системи виявлення фішингових листів з використанням методів машинного навчання

4. Зміст роботи
Вступ
Онсовні поняття фішингу
Існуючі методи захисту від фішингу: огляд та порівняння
Побудова системи виявлення фішингових листів
Висновки

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація для захисту роботи.

6. Дата видачі завдання 24 березня 2025 року

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Формулювання теми дипломної роботи, визначення мети та постановка задач.	24.03.2025	виконано
2	Написання першого розділу: визначення та порівняння основних методів фішингу	24.03.2025 – 01.04.2025	виконано
3	Написання другого розділу: визначення та порівняння сучасних класичних методів виявлення та захисту від фішингу	01.04.2025 – 01.05.2025	виконано
4	Написання третього розділу: створення та тестування комбінованої системи виявлення фішингу	01.05.2025 – 01.06.2025	виконано
5	Підготовка матеріалів до друку та оформлення пояснювальної записки	01.06.2025 – 08.06.2025	виконано
6	Попередній захист дипломної роботи	12.06.2025	виконано
7	Захист дипломної роботи	19.05.2025	

Здобувач вищої освіти _____ Сергій МАЛЕНКО
(підпис) (власне ім'я, ПРІЗВИЩЕ)

Науковий керівник _____ Олександр РИБАК
(підпис) (власне ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Робота складається з 3 розділів, містить 46 ілюстрацій, 18 джерел літератури, обсяг роботи – 59 сторінок.

Метою роботи є розробка і дослідження системи виявлення фішингових листів, що поєднує машинне навчання, аналіз тексту та технічних характеристик листів, інтегруючи класичні методи перевірки як додаткові ознаки.

Об’єктом дослідження є фішингові повідомлення.

Предметом дослідження виступають моделі захисту від фішингу, зокрема комбіновані системи з використанням методів машинного навчання, що допомагають виявляти та протидіяти фішинговим повідомленням.

Методами дослідження є теоретичний аналіз наукової та технічної літератури, експериментальне моделювання, обробка текстових даних, навчання моделей машинного навчання та інтерпретаця результатів

В результаті дослідження було створено комбіновану систему, що поєднала у собі методи аналізу тексту та технічних ознак для виявлення фішингових повідомлень.

Ключові слова: ФІШИНГ, ЕЛЕКТРОННА ПОШТА, ВИЯВЛЕННЯ ФІШИНГУ, МАШИННЕ НАВЧАННЯ.

ABSTRACT

The paper consists of 3 chapters, 46 illustrations, 18 references, and is 59 pages long.

The purpose of the work is to develop and study a phishing email detection system that combines machine learning, text analysis, and email specifications, integrating classical verification methods as additional features.

The object of study is phishing messages.

The subject of the study is phishing protection models, in particular, combined systems using machine learning methods that help to detect and counteract phishing messages.

The research methods are theoretical analysis of scientific and technical literature, experimental modeling, text data processing, training of machine learning models and interpretation of results

As a result of the study, a combined system was created that combines text analysis and technical features to detect phishing messages.

Keywords: PHISHING, EMAIL, PHISHING DETECTION, MACHINE LEARNING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ И ТЕРМІНІВ

Email — електронна пошта; технологія передавання текстових повідомлень та файлів через мережу Інтернет.

NLP (Natural Language Processing) — обробка природної мови; галузь штучного інтелекту, що дозволяє комп'ютерам аналізувати й інтерпретувати людську мову.

IP (Internet Protocol) — протокол міжмережевої взаємодії, який забезпечує адресацію та маршрутизацію пакетів даних.

URL (Uniform Resource Locator) — уніфікований вказівник ресурсу; адреса, яка використовується для доступу до ресурсів в Інтернеті.

SPF (Sender Policy Framework) — механізм автентифікації пошти, що дозволяє виявляти фальсифікацію адреси відправника.

DKIM (DomainKeys Identified Mail) — криптографічний механізм перевірки цілісності та справжності повідомлення електронної пошти.

DMARC (Domain-based Message Authentication, Reporting and Conformance) — політика, яка поєднує SPF та DKIM для підвищення безпеки поштового домену.

DNS (Domain Name System) — система доменних імен; служба, яка перетворює доменні імена на IP-адреси.

ML (Machine Learning) — машинне навчання; підмножина штучного інтелекту, що дозволяє системам навчатися на основі даних без явного програмування.

Feature Engineering — процес створення інформативних змінних (ознак) для покращення якості моделі машинного навчання.

Матриця невідповідності (Confusion Matrix) — таблиця, яка дозволяє оцінити ефективність класифікаційної моделі за кількістю правильних і хибних передбачень.

False Positive / False Negative — типи помилок класифікації: хибнопозитивне (помилково виявлено фішинг) та хибнонегативне (не виявлено справжній фішинг).

HTML (HyperText Markup Language) — мова розмітки, що використовується для створення веб-сторінок та структурування тексту.

TF-IDF (Term Frequency – Inverse Document Frequency) — статистичний метод для визначення важливості терміну в тексті; часто використовується в NLP.

Accuracy / Точність — метрика якості моделі класифікації, що відображає відсоток правильних передбачень від загальної кількості випадків.

ВСТУП

У сучасному світі електронна пошта залишається одним із ключових каналів обміну інформацією — як у приватному, так і в корпоративному секторі. Її зручність, швидкість та універсальність зробили електронну пошту невід'ємною частиною нашого повсякденного життя. Проте разом з розвитком цифрових технологій зростає і кількість кіберзагроз, які намагаються скористатися цією платформою для досягнення злочинних цілей. Однією з найпоширеніших та найбільш небезпечних серед них є фішинг — метод соціальної інженерії, за допомогою якого зловмисники вводять користувачів в оману з метою отримання конфіденційної інформації, фінансових даних або компрометації облікових записів.

Фішингові атаки мають широкий спектр форм і реалізацій: від масових розсилок із загальним змістом до високоспеціалізованих цільових атак (spear-фішинг), спрямованих на окремих осіб або компанії. Згідно з дослідженнями провідних аналітичних центрів, понад 90% кібератак починаються саме з фішингових листів. При цьому методи, які використовуються для фішингу, постійно еволюціонують, пристосовуючись до нових технологій захисту. Зловмисники активно використовують підроблені домени, соціальну інженерію, шкідливі вкладення та посилання, а також автоматизовані генератори фішингових шаблонів.

В умовах постійного зростання масштабів та складності фішингових кампаній традиційні засоби захисту — такі як фільтри спаму, блокування IP-адрес, чорні списки доменів або сигнатурне виявлення — виявляються недостатньо ефективними. У зв’язку з цим зростає інтерес до застосування методів машинного навчання та обробки природної мови (Natural Language Processing, NLP) як більш гнучких і адаптивних рішень для виявлення фішингових атак. Завдяки цим технологіям можливо аналізувати зміст листів, метадані, поведінкові шаблони користувачів, а також автоматично

ідентифікувати підозрілі повідомлення, які могли б пройти повз класичні засоби захисту.

Особливо перспективним є комбінований підхід, коли класичні методи перевірки — такі як аналіз доменів відправника, перевірка URL-адрес через репутаційні сервіси (наприклад, VirusTotal[1]), а також виявлення ключових слів у тексті — використовуються не ізольовано, а як ознаки для моделей машинного навчання. Це дозволяє поєднати експертні знання з адаптивністю алгоритмів і підвищити точність класифікації фішингових повідомлень.

Метою даної дипломної роботи є розробка і дослідження системи виявлення фішингових листів, що поєднує машинне навчання, аналіз тексту та технічних характеристик листів, інтегруючи класичні методи перевірки як додаткові ознаки. У межах роботи буде проведено збір і обробку релевантних датасетів, реалізовано feature engineering із включенням доменних і URL-рейтингів, побудовано декілька моделей класифікації та оцінено їх ефективність за допомогою ключових метрик.

Об'єктом дослідження є фішингові електронні листи як тип соціотехнічних атак у сфері інформаційної безпеки.

Предметом дослідження виступають моделі машинного навчання, що дозволяють виявляти фішингові листи на основі вмісту, структурних характеристик та метаданих повідомлення.

Методи дослідження включають теоретичний аналіз наукової та технічної літератури, експериментальне моделювання, обробку текстових даних, навчання моделей машинного навчання та інтерпретацію результатів

Завданнями роботи є:

1. Здійснити аналіз природи фішингових атак, класифікацію їх основних типів, а також виявити сучасні тенденції розвитку фішингових загроз;
2. Дослідити існуючі методи виявлення фішингових повідомлень та оцінити їх ефективність, зокрема у контексті використання машинного навчання;
3. Обрати релевантний датасет, здійснити його попередню обробку, включно з очищенням, нормалізацією тексту, а також формуванням ознак на основі вмісту листа, заголовків, домену відправника, тощо
4. Побудувати й протестувати кілька моделей машинного навчання для класифікації фішингових листів, порівняти їх ефективність за відповідними метриками;
5. Провести експеримент із застосуванням побудованої моделі на тестовій вибірці та надати аналіз результатів для подальшої інтеграції в системи електронної пошти.

1 Основні поняття фішингу

1.1 Визначення фішингу

Фішинг — це форма кібератаки, яка використовує методи соціальної інженерії для викрадення особистих даних користувачів[2]. Зловмисники маскуються під легітимні організації або осіб, щоб обманом змусити жертву розкрити конфіденційну інформацію. Цей тип шахрайства може здійснюватися через електронні листи, SMS, телефонні дзвінки або фальшиві вебсайти, які імітують відомі сервіси.

Небезпека фішингу полягає не лише в безпосередньому викраденні даних. Такі атаки можуть мати катастрофічні наслідки як для окремих користувачів, так і для цілих організацій. У приватних осіб зловмисники можуть викрасти кошти, використовувати їхні персональні дані для несанкціонованих покупок або навіть для шантажу. У корпоративному чи державному середовищі фішингові атаки часто є початковим етапом складніших загроз — таких як цільові злами, розгортання шкідливого програмного забезпечення чи отримання привілейованого доступу.

Через скомпрометованих користувачів зловмисники отримують можливість обійти зовнішні системи захисту, здійснити атаку зсередини або створити точку входу для подальшого розгортання складніших атак. Унаслідок таких вторгнень організації зазнають значних фінансових втрат, втрати довіри клієнтів, репутаційних ризиків та юридичних наслідків.

1.2 Види фішингу[2], [3]

- **Email-фішинг**: найпоширеніший тип, при якому зловмисники надсилають електронні листи, що імітують повідомлення від відомих компаній або сервісів.
- **Цільовий фішинг (Spear Phishing)**: атака, спрямована на конкретну особу або організацію, з урахуванням зібраної про них інформації.

- **Уейлінг (Whaling)**: фішинг, націлений на високопосадовців або керівництво компаній.
- **Смішинг (Smishing)**: фішинг через SMS-повідомлення, що містять посилання на фальшиві сайти або прохання надати конфіденційну інформацію.
- **Вішинг (Vishing)**: фішинг через телефонні дзвінки, під час яких зловмисники видають себе за представників банків або інших організацій.

1.3 Статистика та ефективність фішингових атак

Через простоту реалізації, масовість та ефективність, фішинг залишається однією з найпоширеніших форм кіберзлочинності у світі. За даними Verizon Data Breach Investigations Report 2025[4], понад 80% успішних кіберзломів використовують фішинг або людський фактор у якійсь мірі. Це свідчить про те, що соціальна інженерія залишається ключовим інструментом зловмисників для проникнення в інформаційні системи.

Через покращення генеративних моделей штучного інтелекту, масштабувати фішинг стало ще простіше, через це, згідно з IBM X-Force Threat Intelligence Index 2025[5], кількість програм-вимагачів, розповсюджених таким чином, виросла на 84% за рік.

Психологічний фактор залишається ключовим — користувачі часто піддаються впливу емоцій, таких як страх втрати доступу, довіра до відомих брендів або терміновість повідомлення. Це робить фішингові атаки ефективними навіть за наявності сучасних технічних засобів захисту.

В останні роки зросла складність фішингових схем: з’явились динамічні фішингові сторінки, персоналізовані повідомлення (spear-фішинг), а також поєднання фішингу з іншими типами атак, що ускладнює виявлення та захист.

Висновки до розділу 1

У роботі було розглянуто основні поняття фішингу як виду кіберзагроз, що базуються на соціальній інженерії та спрямовані на викрадення конфіденційної інформації користувачів.

Пояснено механізми здійснення фішингових атак і їхню різноманітність. Проведено класифікацію основних видів фішингових атак: від масового email-фішингу до цільових форм — spear-фішинг, уейлінг, смішинг і вішинг.

Наведено актуальну статистику щодо поширеності і ефективності фішингових атак на основі звітів провідних аналітичних центрів Verizon та IBM за 2025 рік. Підкреслено, що фішинг залишається одним із найефективніших інструментів початкового доступу у кіберзлочинності.

2 Існуючі методи захисту від фішингу: огляд та порівняння

2.1 Класичні методи захисту

2.1.1 Аутентифікація відправника (SPF, DKIM, DMARC)

Один із базових способів протидії фішинговим листам – перевірка справжності джерела електронної пошти за допомогою стандартів SPF, DKIM та DMARC. Ці технології допомагають виявити спуфінг — підміну адреси відправника, коли лист виглядає нібито надісланим з довіреного джерела. Вони ефективні насамперед проти масових фішингових кампаній, які імітують корпоративні або державні адреси.

- SPF (Sender Policy Framework) — це механізм, який дозволяє домену вказати, з яких IP-адрес дозволено надсилати електронні листи від його імені[6].
- DKIM (DomainKeys Identified Mail) — метод, який додає до листа цифровий підпис, що дозволяє перевірити, чи не було його змінено під час передачі[6].
- DMARC (Domain-based Message Authentication, Reporting and Conformance) — протокол, який поєднує результати перевірки SPF і DKIM та вказує політику обробки листів, що не проходять ці перевірки[6].

Переваги:

- Захищають від підробки адреси відправника.
- Підвищують довіру до легітимних листів.
- Дають змогу отримувачу налаштовувати політики обробки підозрілої пошти.

Недоліки:

- Не працюють, якщо фішингове повідомлення надіслано з легітимного, але зламаного акаунта.
- Не захищають від фішингового вмісту листа — лише перевіряють технічні мета-дані.
- Потребують правильної конфігурації на стороні відправника, яка часто відсутня у малих організаціях.

2.1.2 DNS-фільтрація та перевірка URL-репутації

Багато фішингових атак використовують посилання на шкідливі вебсайти. Тому перевірка репутації вкладених URL — критично важливий напрям.

Сервіси типу Google Safe Browsing, VirusTotal, PhishTank надають API для перевірки URL-адрес у листах.

Переваги:

- Працює незалежно від вмісту листа.
- Може інтегруватись у браузер, мережу або поштовий сервер.
- Оновлюється автоматично з глобальних джерел.

Недоліки:

- Не працює проти нових, обфускованих, або легітимних URL, а також проти редиректів або тимчасових сторінок.
- Залежність від актуальності та досяжності сторонніх баз.

2.1.3 Моніторинг поведінки користувача (UEBA)

UEBA (User and Entity Behavior Analytics) — це підхід до виявлення аномальної активності, який базується на аналізі поведінки користувачів та інших об'єктів IT-систем. UEBA використовує машинне навчання,

статистичні моделі та інші методи аналізу даних для виявлення нетипових дій, які можуть свідчити про компрометацію або зловмисні дії[7].

Ці системи аналізують поведінкові шаблони (час входу, IP-локації, дії в системі, робота з файлами тощо), й у разі виявлення відхилень від «нормальної» поведінки генерують попередження.

Переваги:

- Дозволяє виявити фішинг уже після його здійснення — на етапі експлуатації.
- Особливо корисний для захисту критичних облікових записів.
- Виявляє складні атаки, які обійшли периметрову безпеку.

Недоліки:

- Реагує вже після компрометації.
- Вимагає якісного збору та зберігання логів, а також початкового налаштування та «вивчання» користувачів.

2.1.4 Освітні програми та симуляції фішингу

Оскільки фішингові атаки націлені напряму на експлуатацію людського фактору користувачів, регулярне навчання співробітників і симуляції фішингових атак — важливий елемент захисту.

Переваги:

- Допомагає виявити найвразливіших користувачів.
- Зменшує ймовірність того, що фішинг призведе до втрати даних.
- Створює культуру обережного ставлення до пошти.

Недоліки:

- Не гарантує результату — користувач все одно може припуститись помилки.
- Ефективність знижується з часом після тренінгу.

- Ускладнене вимірювання реального ефекту без аналітики.

2.2 Системи на основі правил і сигнатур

Сигнатурний аналіз — це підхід до виявлення фішингових атак, при якому повідомлення перевіряється на наявність збігів з наперед заданими шаблонами (сигнатурами), які характеризують відомі шкідливі дії[8]. Rule-based (правила) — це набір детермінованих умов (наприклад, «надіслано з підозрілого домену» або «є ключові слова типу “підтвердьте облікові дані”»), які запускають відповідні дії — блокування, маркування тощо[9].

Ці методи лежать в основі багатьох традиційних засобів фільтрації пошти й є одними з найстаріших у сфері захисту від шкідливої електронної комунікації.

Приклади систем

- SpamAssassin — безкоштовна система фільтрації пошти, що використовує набір правил, евристик та сигнатур[10].
- ClamAV — антивірус із відкритим кодом, який підтримує сигнатури для виявлення шкідливих вкладень у поштових повідомленнях[11].
- Cisco Email Security Appliance (ESA) — комерційне рішення, яке поєднує сигнатурний аналіз, URL-фільтрацію та захист від zero-day атак[12].

Переваги:

- Швидкість реагування — сигнатурні системи швидко блокують відомі шаблони атак.
- Простота впровадження — немає потреби в попередньому навчанні чи складних алгоритмах.
- Передбачуваність — адміністратор точно знає, що і чому спрацювало.

Недоліки:

- Неможливість виявлення нових або модифікованих атак — якщо фішингова кампанія змінює шаблон, система її не впізнає.
- Підтримка бази правил — потребує регулярного оновлення сигнатур, інакше ефективність падає.
- Обхід за допомогою обфускації — зловмисники можуть змінювати слова, замінювати символи, щоб обійти правила.
- Вразливість до false positives — суворі правила можуть маркувати легітимні листи як небажані.

2.3 Машинне навчання у виявленні фішингових атак

Машинне навчання (ML) — це підхід, за якого комп'ютерна система навчається розпізнавати закономірності в даних, не маючи заздалегідь заданих правил. У контексті кібербезпеки ML широко використовується для виявлення аномалій, класифікації шкідливих повідомлень, прогнозування поведінки користувачів тощо. Особливо ефективним він є у боротьбі з фішингом, де зловмисники постійно змінюють шаблони повідомлень, що ускладнює застосування традиційних методів.

Наприклад, ML-модель може навчитися розрізняти фішингові та легітимні листи за ознаками стилю мови, структури, частоти слів, наявності підозрілих посилань або незвичних технічних атрибутів.

Формування ознак (Feature Engineering)

Ключовий етап, який безпосередньо впливає на точність моделі. Ознаки можуть бути:

- **Контентні (текстові):**
  - Наявність характерних слів або фраз ("підтвердьте акаунт", "терміново дійте");
  - Кількість слів, довжина речень, пунктуація;

    - Частотні характеристики (TF, TF-IDF);
    - Векторні представлення (Word2Vec, FastText, BERT);
- **Технічні:**
    - Час відправлення;
    - Довжина заголовка;
    - Наявність вкладень;
    - Форматування HTML;
    - Наявність URL-адрес;
- **Репутаційні:**
    - Оцінка домену (наприклад, результат перевірки через VirusTotal);
    - Вік домену;
    - Відповідність SPF/DKIM;

Моделі класифікації

Від вибору моделі машинного навчання, результати класифікації можуть відрізнятися дуже сильно, отже потрібно приділити багато уваги для їх вибору. Популярні моделі, які використовуються для задач виявлення фішингу:

- Логістична регресія — проста інтерпретована модель для лінійної класифікації.
- Random Forest / XGBoost — ансамблі дерев, добре працюють з табличними даними.
- Multinomial Naive Bayes — особливо ефективний для роботи з текстом.
- Нейронні мережі (RNN, CNN, LSTM, BERT) — використовуються для складного аналізу контексту в текстах.
- Гібридні підходи — комбінування ML-моделі з класичними ознаками (наприклад, через Rule+ML pipelines).

Оцінка ефективності

Для вимірювання точності роботи моделей найчастіше використовують такі метрики:

- Accuracy — загальна точність класифікації.
- Precision / Recall / F1-score — особливо важливі для задач із незбалансованими класами (як, наприклад, фішингові листи).
- ROC-AUC — показує здатність моделі відокремити позитивний клас від негативного.

Наприклад, у Google Jigsaw's Phishing Benchmark[13] люди виявляли фішингові листи з точністю близько 68%, тоді як добре навчена модель ML — понад 95%.

Переваги:

- Адаптивність — модель навчається на нових даних і не залежить від фіксованих правил.
- Висока точність — сучасні моделі досягають понад 97–99% точності на тестових вибірках[13].
- Автоматизація — немає потреби вручну оновлювати правила для нових фішингових шаблонів.
- Можливість комбінування — ML легко інтегрується з класичними фільтрами та репутаційними службами.

Недоліки:

- Потреба в якісному датасеті — модель не працюватиме без достатньої кількості маркованих прикладів.
- Схильність до переобучення — при великій кількості ознак модель може втратити здатність до узагальнення.
- Проблеми інтерпретації — особливо у випадку нейронних мереж, важко пояснити причину рішення.

- Швидкість обробки — складні моделі можуть вимагати значних ресурсів, особливо у real-time системах.

2.4 Гібридні та комбіновані підходи

Гібридні системи виявлення фішингу об'єднують класичні методи (фільтрація на основі правил, перевірка SPF/DKIM, чорні списки, репутаційні сервіси) з інтелектуальними алгоритмами (машинне навчання, поведінковий аналіз). Ідея полягає в тому, щоб отримати переваги обох підходів і компенсувати їхні слабкі сторони.

Такі системи зазвичай мають багаторівневу архітектуру, де:

1. Первинна перевірка виконується класичними фільтрами: перевірка домену, наявності шкідливих посилань, сигнатурний аналіз.
2. Другий рівень — застосування моделей ML до тих листів, які не були заблоковані, але виглядають підозріло.
3. Поведінковий моніторинг або XDR/SIEM системи спостерігають за реакцією користувача на потенційно шкідливу пошту.

Типові архітектури гібридного захисту

- Rule-based + ML: класичні правила додають ознаки до вхідних листів (наприклад, прапорець “відсутній SPF”), які потім подаються в ML-модель.
- URL reputation + ML: результат перевірки посилань через VirusTotal або інші сервіси інтегрується у вектор ознак.
- Багатоетапна оцінка: листи проходять кілька етапів фільтрації: сигнатурний аналіз, контентний аналіз, ML-класифікація, поведенкова оцінка.

Переваги:

- Сумісність з існуючою інфраструктурою — класичні фільтри залишаються, але їхні результати можуть слугувати входом до ML-моделі.
- Гнучкість і масштабованість — можлива побудова кастомних конвеєрів фільтрації залежно від потреб організації.
- Покращена точність — комбінування ознак із різних джерел дає багатшу інформацію для прийняття рішення.
- Зменшення false positives — ML-модель може уточнювати результат класичного фільтра й не блокувати легітимні листи помилково.

Недоліки:

- Технічна складність — інтеграція декількох підходів вимагає високої кваліфікації та налагодженої інфраструктури.
- Вимоги до ресурсів — деякі ML-моделі потребують потужного обчислювального середовища або хмарних сервісів.
- Труднощі в оцінці — важко ізолювати, який саме компонент системи спрацював, особливо при помилковій класифікації.
- Залежність від якісних даних — без належного датасету для навчання навіть найкраща архітектура не дасть результату.

2.5 Аналіз ефективності та слабких сторін методів захисту

Жоден із методів захисту від фішингових атак не є універсальним чи достатнім сам по собі. Найкращі результати показують **комбіновані підходи**, які поєднують класичні техніки перевірки з інтелектуальними алгоритмами та людським фактором (навчання персоналу).

Таблиця 2.1 – Переваги та недоліки класичних методів виявлення фішингу

Підхід	**Сильні сторони**	**Слабкі сторони**
Аутентифікація (SPF/DKIM/DMARC)	Ефективний проти підробки адреси відправника	Не працює проти скомпрометованих акаунтів і шкідливого контенту
DNS-фільтрація / URL-репутація	Швидка реакція на відомі загрози	Не виявляє нові або зашифровані URL, залежить від актуальності баз
UEBA / поведінковий аналіз	Виявляє аномалії після компрометації	Не блокує загрозу до активації; потребує логування та ресурсів
Rule-based / сигнатурний аналіз	Швидкість, інтерпретованість, надійність для типових шаблонів	Неможливість виявити нові або персоналізовані атаки
Машинне навчання	Адаптивність, гнучкість, здатність виявляти нові загрози	Високі вимоги до даних, продуктивності, інтерпретованості
Гібридні системи	Висока точність, масштабованість, стійкість до нових атак	Складність інтеграції, потреба в інженерних ресурсах, складність оцінки

Основні проблеми та виклики

1. Хибні висновки

False Positive та False Negative помилки мають прямі наслідки: перші – порушують бізнес-процеси, другі – загрожують безпеці, тому метою будь-якої системи аналізу має бути їх мінімізація.

2. Адаптивність зловмисників

Сучасні фішингові кампанії часто використовують:

- Персоналізовані повідомлення (spear-фішинг),
- Зміни стилістики, уникнення ключових слів,
- Нові URL, скорочувачі посилань, редиректи.

Це ускладнює фільтрацію за фіксованими правилами або репутаційними базами.

3. Людський фактор

Навіть найкраща технічна система може бути зруйнована одним неправильним кліком. За статистикою Proofpoint[14], у 2023 році 71% організацій зазнавали фішингових інцидентів через дії користувачів, незважаючи на наявність технічного захисту. Також, через популяризацію інших каналів фішингу – SMS (smishing), телефонні дзвінки (vishing), месенджери, соцмережі, тощо, ніяка система не може повністю захистити від фішингу по усім можливим каналам, тому компаніям варто інвестувати не лише у технічні інновації, а й у навчання своїх працівників.

Висновки до розділу 2

У цьому розділі було здійснено комплексний аналіз сучасних методів захисту від фішингових атак. Розгляд охопив як класичні підходи — засновані на сигнатурах, правилах, аутентифікації та репутаційних механізмах — так і інтелектуальні системи, побудовані на основі машинного навчання та поведінкового аналізу.

Зокрема, встановлено, що традиційні інструменти накшталт SPF/DKIM/DMARC, rule-based фільтри та репутаційні бази URL-адрес, демонструють високу ефективність у виявленні відомих або масових фішингових кампаній. Водночас вони мають обмеження при протидії новим, персоналізованим або обфускованим атакам.

Системи на основі машинного навчання — особливо при використанні контентного аналізу, технічних атрибутів листа та репутаційних ознак як фіч — демонструють значно вищу адаптивність та здатність виявляти раніше невідомі фішингові зразки. Проте їх ефективність значною мірою залежить від якості навчальних даних, вибору моделей і належної інтеграції в загальну інфраструктуру захисту.

Різні методи захисту мають свої переваги та недоліки, але тільки комбінований підхід, що поєднує технічні засоби (аутентифікацію, фільтрацію, ML), аналітику (UEBA) та освітні ініціативи, дає змогу ефективно протидіяти фішинговим загрозам. При цьому важливо не лише впроваджувати технічні рішення, але й постійно аналізувати їхню ефективність, оновлювати стратегії, враховуючи нові вектори атак і поведінку користувачів. Саме такі підходи є найбільш перспективними в умовах швидко змінюваного ландшафту загроз.

У розділі також було виокремлено ключові виклики: хибні спрацьовування, потреба в адаптивності, а також людський фактор, який залишається одним з основних векторів ризику.

Таким чином, ефективна протидія фішинговим атакам у сучасних умовах потребує не лише надійних технічних рішень, але й гнучкої багаторівневої стратегії, що включає навчання користувачів, автоматизовану аналітику та постійний моніторинг результатів. У наступному розділі буде розглянуто процес побудови практичної системи виявлення фішингових повідомлень на основі комбінованого підходу з використанням машинного навчання та вбудованих репутаційних механізмів.

3 Побудова системи виявлення фішингових листів

3.1 Постановка задачі та загальний підхід

У межах цієї роботи ставиться задача створення інтелектуальної системи виявлення фішингових повідомлень електронної пошти. На відміну від традиційних фільтрів, що працюють на основі фіксованих правил або сигнатур, запропонований підхід базується на алгоритмах машинного навчання та використовує комбінований набір ознак, що включає як текстову інформацію, так і технічні атрибути повідомлення.

Наша система буде класифікувати електронні листи як фішингові або легітимні на основі наступних даних:

- Текст теми та тіла повідомлення,
- Метадані (Ім’я та адреса відправника, список отримувачів),
- Результати зовнішніх перевірок (URL-репутація, автентифікація домену тощо).

Використані дані

Для реалізації завдання використовується набір CEAS_08.csv з відкритого джерела Phishing Email Dataset на Kaggle[15].
Цей датасет був обраний з огляду на наявність таких ключових полів:

- Адреса відправника (Sender),
- Адреса отримувача (Receiver),
- Тема листа (Subject),
- Тіло повідомлення (Body),
- Дата надсилання (Date),
- Мітка класу (Label).

Запропонована модель використовує гібридну логіку виявлення фішингу: класичні ознаки не виконують фільтрацію напряму, але

включаються у модель як фічі. Це дозволяє зберегти інтерпретованість та ефективність класичних методів, водночас використовуючи гнучкість машинного навчання.

Наприклад, система використовуватиме такі ознаки:

- Результати перевірки URL-адрес у тілі листа через VirusTotal API (чи були знайдені в базах як шкідливі).
- Перевірка SPF-запису домену відправника (наявність або відсутність валідної авторизації).
- Наявність підозрілих слів/фраз у темі чи тексті (наприклад: “validate account”, “urgent”, “click here”).
- Різниця між доменом відправника та відомими популярними сервісами (наприклад, gmail.com vs gma1l.com), оскільки фішингові повідомлення часто намагаються їх імітувати.
- Чи збігається домен адреси відправника з посиланнями у тілі повідомлення.

Список ознак не є вичерпним та остаточно правильним, вибір ознак буде обгрунтований при їх використанні.

Такі ознаки будуть обчислюватися автоматично під час попередньої обробки повідомлення та використовуватись як вхід до моделі класифікації.

Обраний стек технологій:

- Мова програмування: Python
- Парсер файлів: eml-parser
- Обробка даних: Pandas
- Обробка тексту: scikit-learn
- Класифікація: XGBoost, RandomForest, Naive Bayes (для порівняння)
- Зовнішні перевірки: VirusTotal API

Такий підхід дозволяє побудувати адаптивну й ефективну систему, яка не лише виявляє фішинг за шаблоном, а й узагальнює поведінкові та технічні закономірності, властиві шкідливим листам.

3.2 Опис датасету

Для реалізації системи виявлення фішингових повідомлень у межах цієї роботи було обрано відкритий набір даних “Phishing Email Dataset”, доступний на платформі Kaggle[15]. Зокрема, використовувався піднабір CEAS_08.csv, який був створений для участі у конференції CEAS (Conference on Email and Anti-Spam). Варто зазначити, що датасет містить переважно повідомлення 2008-го року, що підходить для теоретичного розгляду підходу у рамкаї цієї роботи, проте для практичного використання краще обрати чи створити новіший датасет, оскільки технології фішингу за останні роки дуже сильно адаптувалися та змінилися, та модель натренована на настільки старих даних буде гірше виявляти сучасні листи. З урахуванням сказаного, цей датасет має низку ключових переваг:

- Містить як фішингові, так і легітимні листи.
- Є структурованим та добре анотованим.
- Містить як текстову частину повідомлення, так і метаінформацію (адреси, заголовки тощо), що дозволяє реалізувати комбінований підхід до побудови ознак.

Загальні характеристики:

Таблиця 3.1 – Опис основних метрик датасету

Формат	CSV
Кількість записів	~39,000 листів
Класи	Phishing / Legitimate
Кількість фішингових	≈ 50%
Мова листів	Англійська

Поля датасету:

- Sender — адреса відправника електронного листа.
- Receiver — адреса отримувача.
- Date — дата й час надсилання.
- Subject — тема повідомлення.
- Body — основний текст листа.
- Label — цільова змінна (0 = легітимний лист, 1 = фішинговий).

Відповідність задачі

Цей датасет є особливо зручним для побудови системи, яка поєднує класичні та ML-підходи. Наявність ключових полів дозволяє реалізувати:

- Контентний аналіз теми та тіла листа;
- Оцінку технічних метаданих (SPF, URL, кількість посилань, довжина теми тощо);
- Побудову комплексного класифікатора.

Єдиним обмеженням є те, що датасет не включає вкладення, тому аналіз вмісту файлів за межами тексту не є можливим у цій роботі. Також, оскільки датасет складається лише з інформації про повідомлення, а не самих .eml файлів, в яких міститься набагато більше інформації про листи, дані щодо SPF-захисту будуть братися з сучасних баз даних, а не тих, що існували

на час відправки листа. Знову, це не вплине на результати у рамках поточної задачі, оскільки ми будемо тестувати модель в тих же умовах, але для реального використання варто навчити модель на .eml файлах, оскільки вони містять дані про SPF-захист на момент відправки.

3.3 Попередня обробка даних

Попередня обробка є критично важливим етапом у побудові системи виявлення фішингових повідомлень. У цій роботі обрано практичний і продуктивний підхід, який поєднує класичну обробку тексту методом TF-IDF з додаванням структурних та технічних ознак. Це дозволяє створити збалансовану систему, здатну виявляти як шаблонні фішингові повідомлення, так і складніші, які не містять очевидних ключових слів.

Структура наших вхідних даних вже була описана вище.

3.3.1 Константи та допоміжні функції

Попередньо було створено декілька констант та допоміжних функцій, що надалі будуть використовуватися для спрощення роботи:

VIRUSTOTAL_API_KEY – API-ключ до VirusTotal для аналізу репутації посилань у тексті.

SUSPICIOUS_KEYWORDS – перелік підозрілих слів, що можуть вказувати на фішинг.

COMMON_DOMAINS – перелік 1000 найпопулярніших доменів задля перевірки їх імітації у адресі відправника[16].

```
VIRUSTOTAL_API_KEY = ''
SUSPICIOUS_KEYWORDS = ['verify', 'login', 'click here', 'account', 'urgent', 'password', 'update', 'security']
COMMON_DOMAINS = ['google.com', 'youtube.com', 'facebook.com', 'baidu.com', 'yahoo.com', 'amazon.com', 'wikipedia.org', 'google.co.in',
```

Рисунок 3.1 - Константи для попередньої обробки даних

extract_urls – функція для отримання усіх посилань з тексту.

extract_emails – функція для отримання усіх email-адрес з тексту.

get_domain – функція для виділення домену з будь-якого посилання.

```
def extract_urls(text):
    return re.findall( pattern: r'(https?://[^\s]+)', text)


1 usage
def extract_emails(text):
    return re.findall( pattern: r'[\w\.-]+@[\w\.-]+\.\w+', text)

4 usages
def get_domain(url):
    domain = url.replace('http://', '').replace('https://', '')
    domain = domain.split('/')[0]
    domain = '.'.join(domain.split('.')[-2:])
    return domain
```

Рисунок 3.2 - Допоміжні функції для попередньої обробки даних

domain_distance – функція для розрахування текстової відстані (помилок) між двома доменами.

```
def domain_distance(domain1, domain2):
    return sum(a != b for a, b in zip(domain1, domain2)) + abs(len(domain1) - len(domain2))
```

Рисунок 3.3 - Функція domain_distance

3.3.2 Векторизація тексту

Для перетворення тексту в числову форму використано метод TF-IDF (TfidfVectorizer з sklearn), без попередньої складної обробки:

- Текст не лематизується,
- Стоп-слова не видаляються,
- HTML-теги не видаляються,
- Токенізація — стандартна (на рівні слів).

Цей підхід дозволяє отримати базову, але інформативну векторну репрезентацію кожного повідомлення, проте при створенні справжньої системи виявлення фішингу радиться використовувати усі вищеперераховані підходи для кращього тренування моделі.

Також, задля спрощення обробки та навчання моделі, поля теми та тіла повідомлення були скомбіновані у одне поле – text_combined.

```
data['text_combined'] = data['subject'] + ' ' + data['body']

vectorizer = TfidfVectorizer()
X_tfidf = vectorizer.fit_transform(data['text_combined']).toarray()
```

Рисунок 3.4 - Векторизація тексту

3.3.3 Побудова структурних та технічних ознак

Окрім тексту, було додано шість штучно створених ознак, які формуються на основі вмісту листа та адреси відправника:

num_urls – кількість посилань, виявлених у тексті листа.

Використовує вищезазначену допоміжну функцію extract_urls.

Логіка включення: фішингові повідомлення часто використовують велику кількість посилань та закликів до дії, щоб створити ілюзію терміновості або важливості.

```
urls = extract_urls(text_combined)
num_urls = len(urls)
```

Рисунок 3.5 - Підрахунок кількості посилань у тексті

has_spf – наявність SPF-запису для домена відправника повідомлення (0 або 1).

Використовує dnspython[17] для отримання DNS-записів про домен відправника.

Логіка включення: при відсутності SPF-запису, шахраї можуть використовувати реальний довірений домен для відправки фішингових повідомлень з високим рівнем правдоподібності, при присутності SPF-запису, лише довірені доменом обличчя можуть відправляти повідомлення.

Зауваження: звісно, використання асинхронності у цій функції та при перевірці репутації домену на VirusTotal значно покращило б швидкість попередньої обробки, але для нашого відносно невеликого датасету це не сильно вплине на кінцевий час.

```
def check_spf(url):
    domain = get_domain(url)
    try:
        answers = dns.resolver.resolve( qname: f'{domain}', rdtype: 'TXT')
        for rdata in answers:
            if any(b'v=spf1' in txt for txt in rdata.strings):
                return 1
    except Exception:
        pass
    return 0
```

Рисунок 3.6 - Функція check_spf

suspicious_count – кількість підозрілих слів, типових для фішингу, у тексті повідомлення.

Використовує вищезазначений перелік підозрілих слів SUSPICIOUS_KEYWORDS.

Логіка включення: велика кількість фішингових повідомлень використовують схожі методи для зазивання читача на наступний крок – терміновість (“urgent”, “click here”, “update”, тощо), небезпека (“verify”, “password”, “security”, тощо), мімікрія під легітимне повідомлення (“login”, “update”, “account”, тощо).

```
suspicious_count = sum(1 for kw in SUSPICIOUS_KEYWORDS if kw in text_combined.lower())
```

Рисунок 3.7 - Підрахунок підозрілих слів у тексті

domain_match – вказує чи зустрічається домен відправника серед доменів у тексті повідомлення.

Використовує вищезазначену функцію отримання домену get_domain.

Логіка включення: фішингові повідомлення часто розсилаються масово з сотен або тисяч різних поштових аккаунтів з різними доменами, проте ведуть зазвичай на один й той самий. Також шахраї бояться використовувати справжній домен, щоб він не почав помічатися як шкідливий, чи спам, легітимні ж повідомлення зазвичай відправляються від імені компанії.

```
domain_match = int(any(sender_domain in get_domain(url) for url in urls))
```

Рисунок 3.8 - Перевірка наявності домену відправника у тексті

min_dist – мінімальна текстова відстань до відомих доменів.

Використовує вищезазначені перелік найпопулярніших доменів COMMON_DOMAINS та функцію для розрахування відстані між доменами domain_distance.

Логіка включення: шахрайські повідомлення та домени часто імітують справжні легітимні повідомлення та домени добре знайомих компаній (наприклад, google.com та g00gle.com).

```
min_dist = min([domain_distance(sender_domain, d) for d in COMMON_DOMAINS])
```

Рисунок 3.9 - Відстань до найбільш схожого популярного домену

vt_score_min – найнижчий рейтинг посилання з повідомлення за версією VirusTotal.

Логіка включення: VirusTotal є найбільшим та найпопулярнішим агрегатором інформації про шкідливі домени, ми можемо використати його рейтинг сайту як одну з ключових метрик його легітимності.

Зауваження: через обмеження використання API та те, що переважна більшість доменів з датасету або не існують, або були використані лише у якості прикладу, ця ознака не буде використовуватися при практичному тестуванні моделі.

```python
def get_virustotal_score(url):
    headers = {'x-apikey': VIRUSTOTAL_API_KEY}
    try:
        url_id = base64.urlsafe_b64encode(url.encode()).decode().strip('=')
        response = requests.get( url: f'https://www.virustotal.com/api/v3/urls/{url_id}', headers=headers)
        if response.status_code == 200:
            data = response.json()
            return data.get('data', {}).get('attributes', {}).get('reputation', )
    except Exception:
        pass
    return 0
```

Рисунок 3.10 - Функція get_virustotal_score

```python
vt_scores = [get_virustotal_score(url) for url in urls] if urls else [0]
vt_score_min = min(vt_scores)
```

Рисунок 3.11 - Отримання домену з найгіршою репутацією VirusTotal

Оброблені дані з новими ознаками будемо зберігати за допомогою pickle[18] для подальшого навчання моделі.

Повний скрипт для попередньої обробки датасету:

```
import pandas as pd
import numpy as np
import base64
import pickle
import re
from sklearn.feature_extraction.text import TfidfVectorizer
import dns.resolver
import requests

VIRUSTOTAL_API_KEY = ''
SUSPICIOUS_KEYWORDS = ['verify', 'login', 'click here', 'account', 'urgent', 'password', 'update', 'security']
COMMON_DOMAINS = ['google.com', 'youtube.com', 'facebook.com', 'baidu.com', 'yahoo.com', 'amazon.com', 'wikipedia.org', 'goo


1 usage
def extract_urls(text):
    return re.findall( pattern: r'(https?://[^\s]+)', text)


1 usage
def extract_emails(text):
    return re.findall( pattern: r'[\w\.-]+@[\w\.-]+\.\w+', text)

3 usages
def get_domain(url):
    domain = url.replace('http://', '').replace('https://', '')
    domain = domain.split('/')[0]
    domain = '.'.join(domain.split('.')[-2:])
    return domain


1 usage
def check_spf(url):
    domain = get_domain(url)
    try:
        answers = dns.resolver.resolve( qname: f'{domain}', rdtype: 'TXT')
        for rdata in answers:
            if any(b'v=spf1' in txt for txt in rdata.strings):
                return 1
    except Exception:
        pass
    return 0
```

Рисунок 3.12 - Скрипт для попередньої обробки датасету

```python
def get_virustotal_score(url):
    headers = {'x-apikey': VIRUSTOTAL_API_KEY}
    try:
        url_id = base64.urlsafe_b64encode(url.encode()).decode().strip('=')
        response = requests.get( url: f'https://www.virustotal.com/api/v3/urls/{url_id}', headers=headers)
        if response.status_code == 200:
            data = response.json()
            return data.get('data', {}).get('attributes', {}).get('reputation', )
    except Exception:
        pass
    return 0


1 usage
def domain_distance(domain1, domain2):
    return sum(a != b for a, b in zip(domain1, domain2)) + abs(len(domain1) - len(domain2))


1 usage
def preprocess_data(input_file, output_file):
    data = pd.read_csv(input_file).dropna()

    data['text_combined'] = data['subject'] + ' ' + data['body']

    extra_features = []
    rows_to_drop = []
    for i, row in data.iterrows():
        text_combined = row['text_combined']

        sender_emails = extract_emails(row['sender'])
        if len(sender_emails):
            sender_domain = get_domain(sender_emails[-1].split('@')[1])
        else:
            rows_to_drop.append(i)
            continue

        urls = extract_urls(text_combined)
        num_urls = len(urls)
        vt_scores = [get_virustotal_score(url) for url in urls] if urls else [0]
        vt_score_min = min(vt_scores)
```

Рисунок 3.13 - Скрипт для попередньої обробки датасету

```
        has_spf = check_spf(sender_domain)
        suspicious_count = sum(1 for kw in SUSPICIOUS_KEYWORDS if kw in text_combined.lower())
        domain_match = int(any(sender_domain in get_domain(url) for url in urls))
        min_dist = min([domain_distance(sender_domain, d) for d in COMMON_DOMAINS])

        extra_features.append([
            num_urls,
            vt_score_min,
            has_spf,
            suspicious_count,
            domain_match,
            min_dist
        ])

    data.drop(index=rows_to_drop, inplace=True)
    data.reset_index(drop=True, inplace=True)

    vectorizer = TfidfVectorizer()
    X_tfidf = vectorizer.fit_transform(data['text_combined']).toarray()
    X_extra = np.array(extra_features)
    X_final = np.hstack([X_tfidf, X_extra])

    y = data['label'].values

    with open(output_file, 'wb') as f:
        pickle.dump( obj: (X_final, y, vectorizer), f)

    print(f'Preprocessed data saved to {output_file}')


if __name__ == '__main__':
    preprocess_data( input_file: '../data/CEAS_08.csv',  output_file: '../data/preprocessed_data.pkl')
```

Рисунок 3.14 - Скрипт для попередньої обробки датасету

3.4 Побудова моделі

Після формування повного набору ознак на основі вмісту електронного листа, технічних метаданих та структурних характеристик, наступним кроком є побудова та навчання моделі машинного навчання для класифікації повідомлень як фішингових або легітимних.

3.4.1 Постановка задачі

Метою цього розділу є побудова бінарного класифікатора, який на основі усіх перелічених вище ознак має передбачати чи є лист фішинговим (label = 1), чи ні (label = 0).

3.4.2 Вибір моделей

Для початку тестування було розглянуто декілька поширених моделей машинного навчання. Основними критеріями вибору стали: здатність моделі працювати з великою кількістю ознак (зокрема, з векторами TF-IDF), стійкість до шуму, інтерпретованість та якість класифікації. У результаті було обрано наступні моделі:

- Logistic Regression – найпростіша та найшвидша модель, що досить добре підходить для високовимірних даних, таких як TF-IDF вектори. Основа її перевага – легка інтерпретованість, проте вона дуже чутлива до некоректного масштабування ознак і менш ефективна при наявності складних нелінійних взаємозв’язків між фічами.
- Random Forest – ансамблева модель на основі дерев рішень. Основна перевага – стійкість до надлишкових та неінформативних ознак, а також відсутність потреби у масштабуванні. Також ця модель дуже добре працює саме з комбінованим набором ознак (як наші TF-IDF та структурні ознаки). Основним недоліком є значно більша потреба в ресурсах, ніж у логістичної регресії.
- XGBoost – одна з найпотужніших та найресурсоємніших ансамблевих моделей. Показує дуже високі результати на табличних даних, а також

ефективно справляється з незбалансованими класами (що притаманно нашому датасету) та має дуже гнучку систему налаштування гіперпараметрів. Недоліками її є складність налаштування та набагато більший час навчання.

Заздалегідь можемо припустити, що ансамблеві підходи справляться з задачею краще, оскільки краще працюють з великою кількістю різнорідних ознак та не потребують нормалізації даних.

3.4.3 Тестування моделей

Для моделей будемо використовувати базові налаштування, з подальшою можливістю гіперпараметричного налаштування.

Зауваження: через недостатню потужність мого комп'ютера навчання та тестування моделей проходило на половині обраного датасету (20.000 записів), що може бути недостатньо для повного тренування моделі, проте вже має показати переваги комбінованого підходу.

Підхід до тестування:

Розіб'ємо датасет, де 80% записів будуть відведені для тренування, а 20% для тестування.

```
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.2)
```

Рисунок 3.15 - Розбиття датасету на тренувальні та тестові дані

Натренуємо модель на тренувальному датасеті.

```
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

Рисунок 3.16 - Приклад тренування моделі

Перевіримо точність моделі за допомогою вбудованої функції accuracy_score.

```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Model Accuracy: {accuracy}')
```

Рисунок 3.17 - Оцінка точності моделі

Результати тестування:

Налаштування Logistic Regression з модуля scikit-learn:

```
model = LogisticRegression(max_iter=1000)
```

Рисунок 3.18 - Налаштування Logistic Regression

Точність Logistic Regression – 99.04%:

```
Model Accuracy: 0.9903943377148635
True positives: 2201    True negatives: 1717
False positives: 27    False negatives: 11
```

Рисунок 3.19 - Точність Logistic Regression

Налаштування Randon Forest з модуля scikit-learn:

```
model = RandomForestClassifier()
```

Рисунок 3.20 - Налаштування Random Forest

Точність Random Forest – 99.19%:

```
Model Accuracy: 0.9919110212335692
True positives: 2186    True negatives: 1738
False positives: 16    False negatives: 16
```

Рисунок 3.21 - Точність Random Forest

Налаштування XGBoost з модуля xgboost:

```
model = XGBClassifier(
    n_estimators=100,
    max_depth=5,
    learning_rate=0.1,
    use_label_encoder=False,
    eval_metric='logloss'
)
```

Рисунок 3.22 - Налаштування XGBoost

Точність XGBoost – 99.49%:

```
Model Accuracy: 0.9949443882709807
True positives: 2252    True negatives: 1684
False positives: 12    False negatives: 8
```

Рисунок 3.23 - Точність XGBoost

Можемо побачити, що як хоча різниця у результатах досить невелика, як ми і припускали, XGBoost показав найкращі результати. З іншого боку, Logistic Regression показав результати значно краще, ніж очікувалося, що може бути пов’язано з невеликим розміром датасету, через що ансамблеві моделі не встигли достатньо навчитися.

Також можемо розглянути важливість додаткових ознак для Random Forest (майже аналогічна для інших).

Пояснення вагів:

- Кількості посилань у повідомленні була приділена найбільша вага, можливо через те що велика кількість записів у тренувальному датасеті була створена з використанням SpamAssassin, таким повідомленням зазвичай притаманна велика кількість посилань.
- Оцінка домену відправника за допомогою VirusTotal має нульову вагу, оскільки реальні дані не вносилися через вищезазначені причини. У

реальних умовах можемо очікувати, що цій метриці буде приділена найбільша вага.

- Наявності SPF флагу була приділена відносно мала вага по двум причинам. По-перше, датасет був створений у 2008 році, з того часу більша частина доменів, що реально використовувалися для фішингу або були перерофільовані, або закинуті. По-друге, у датасеті є досить велика кількість синтетично створених повідомлень з використанням реальних легітимних доменів з SPF-флагом як доменів відправника, через що цей флаг не так сильно впливав на результат. Знову, у реальних умов значення цього флагу має бути значно вищим.
- Кількість підозрілих слів має відносно велику вагу через вищезазначені причини: синтетично створені повідомлення та дата створення датасету, у ті часи фішинг, як і сам інтернет ще був досить молодим та використовував найтривіальніші методи з підозрілими словами.
- Наявності домену відправника у повідомленні також була приділена досить висока вага, саме через те що велика кількість повідомлень у датасеті мала різних відправників, проте доволі схожі або навіть однакові повідомлення.
- «Схожості» домену відправника на популярні сервіси також приділена нульова вага, це могло статися по двум причинам: список популярних доменів ми брали сучасний, можливо більша частина цих доменів ще навіть не існувала, або не була такою популярною у часи створення датасету, або знову через синтетично створені листи.

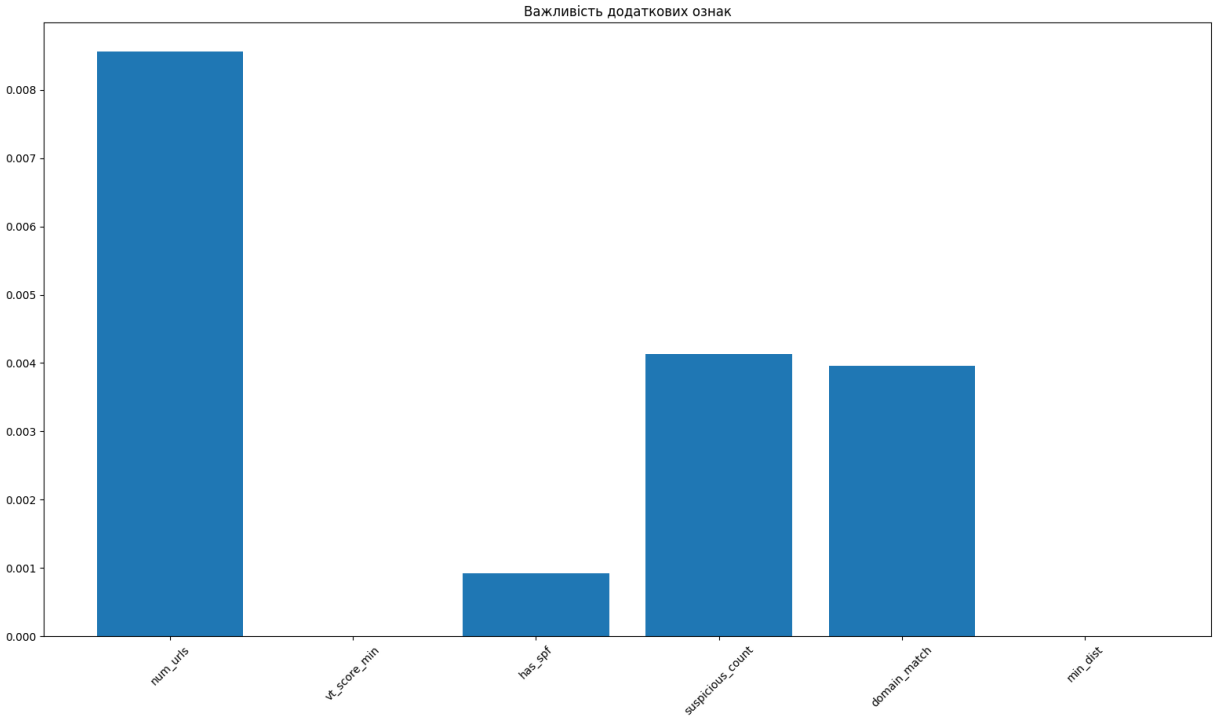

Рисунок 3.24 - Вага додаткових ознак

Повний скрипт для тренування та тестування моделі:

```
import pickle
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix

1 usage
def train_model(data_file, model_file):
    with open(data_file, 'rb') as f:
        X, y, _ = pickle.load(f)

    X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.2)

    model = LogisticRegression()
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    CM = confusion_matrix(y_test, y_pred)
    print(f'Model Accuracy: {accuracy}')
    print(f'True positives: {CM[1][1]}    True negatives: {CM[0][0]}')
    print(f'False positives: {CM[0][1]}    False negatives: {CM[1][0]}')

    with open(model_file, 'wb') as f:
        pickle.dump(model, f)

    print(f'Model saved to {model_file}')

if __name__ == '__main__':
    train_model( data_file: '../data/preprocessed_data_0.pkl',  model_file: '../models/logistic_regression.pkl.pkl')
```

Рисунок 3.25 - Скрипт для тренування та тестування моделі

3.4.4 Тестування класичних підходів

Для тестування класичних підходів виявлення фішингу, оцінимо ознаки, що були передані, та їх індивідуальну передбачувану здатність.

Для початку, проведемо невеликий дослідницький аналіз даних та подивимося як кожна ознака розподілена та де краще провести лінію розподілу між фішингом та легітимними повідомленнями.

Кількість посилань

Можемо побачити, що дана ознака має велику кількість викидів. Також, якщо у повідомленні 0 посилань, воно може бути як легітимним так і фішинговим, проте якщо більше 0 – майже усі такі повідомленя є фішинговими.

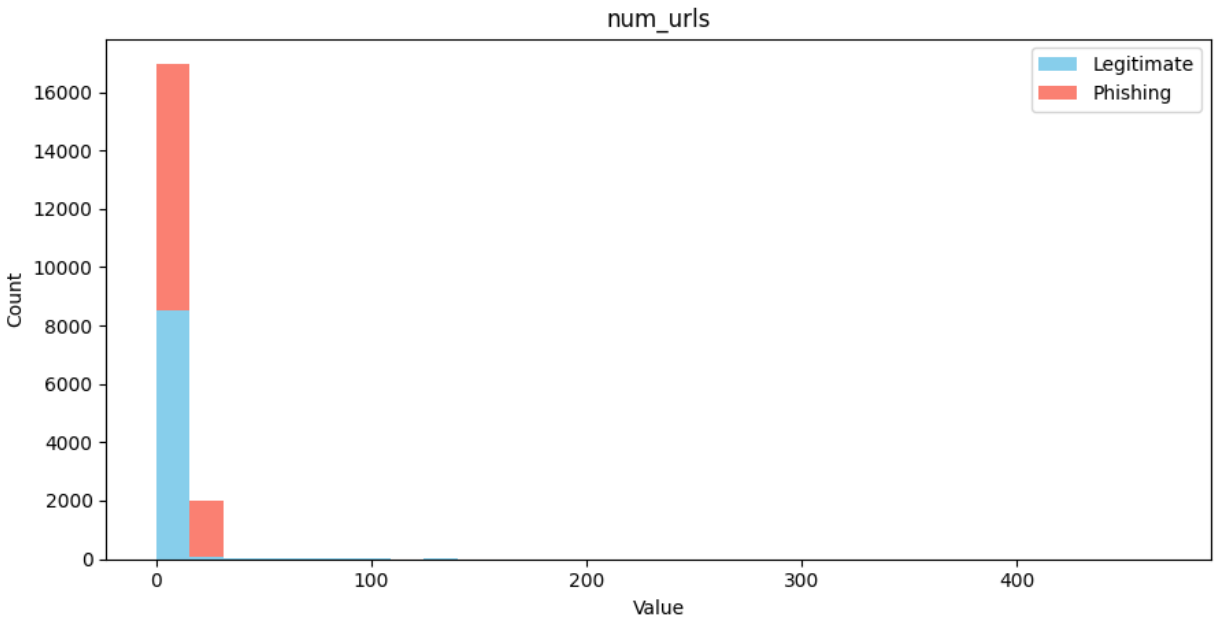

Рисунок 3.26 - Розподіл кількості посилань

Наявність SPF

Наявність SPF запису не є вирішальною ознакою, оскільки приблизно половина усіх повідомлень, що з ним що без нього, є фішинговими. Скоріше за все, це через вищезазначені причини – синтетичні повідомлення та старий датасет.

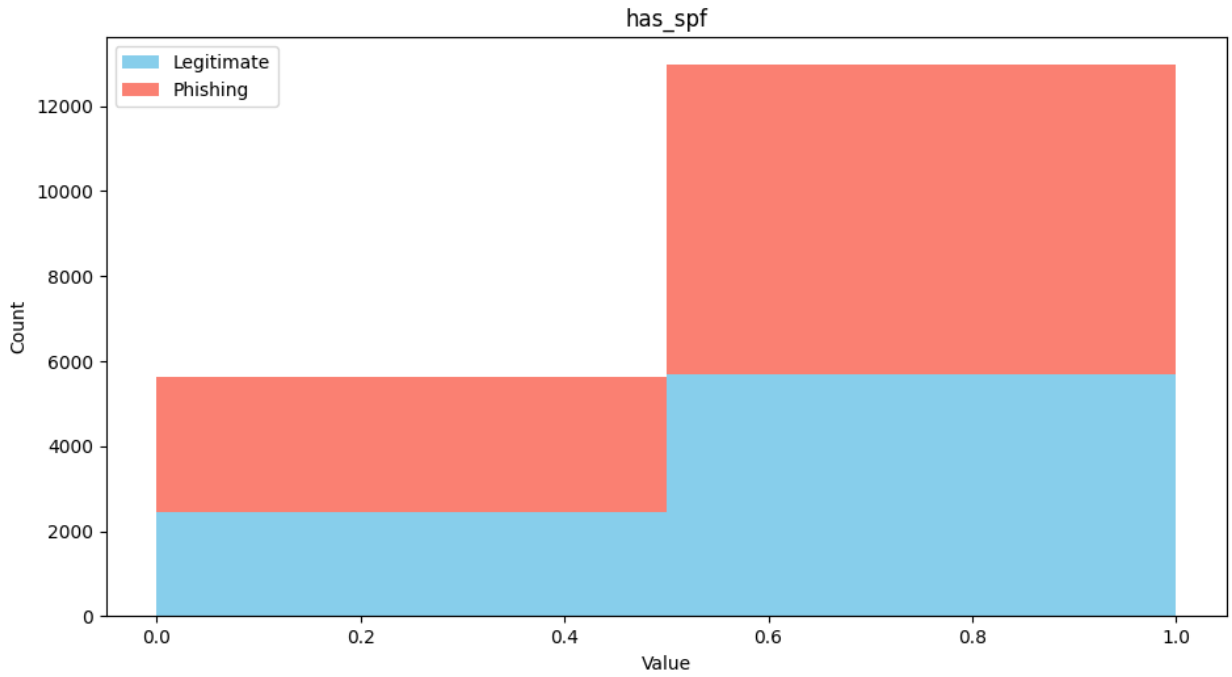

Рисунок 3.27 - Розподіл наявності SPF запису

Кількість підозрілих слів

Хоча ця ознака і є дуже показниковою у наші часи, мабуть, у часи створення датасету такі підходи до фішнгу не були популярні, тому якщо у повідомленні 0 підозрілих слів, з більшою частиною вірогідності воно фішингове, а при збільшенні кількості таких слів, повідомлення все частіше є легітимними.

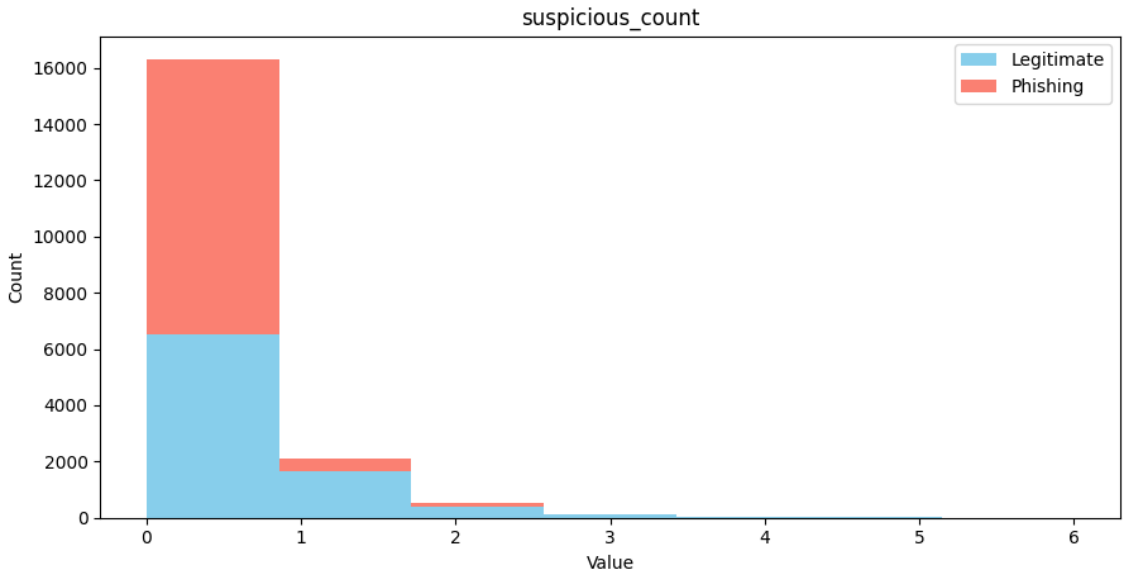

Рисунок 3.28 - Розподіл кількості підозрілих слів

Співпадіння адреси відправника з посиланнями у повідомленні

Можемо побачити, що при неспівпадінні адреси відправника, повідомлення частіше є фішинговим, проте при співпадінні – абсолютно усі повідомлення є легітимними.

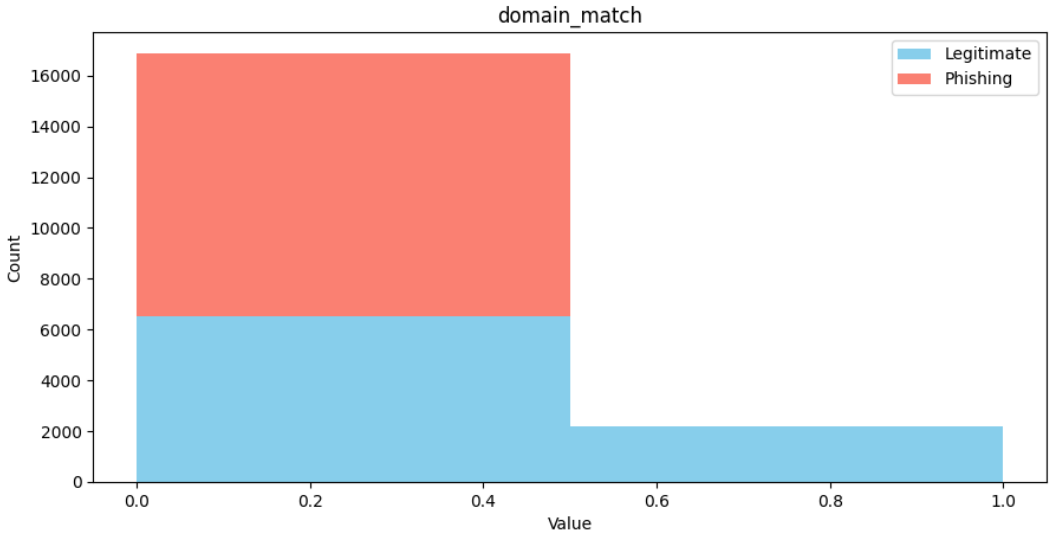

Рисунок 3.29 - Розподіл співпадіння домену

Відстань до популярних доменів

Можемо побачити, що переломний момент настає приблизно на відстані у 5, після якої майже усі повідомлення є фішинговими.

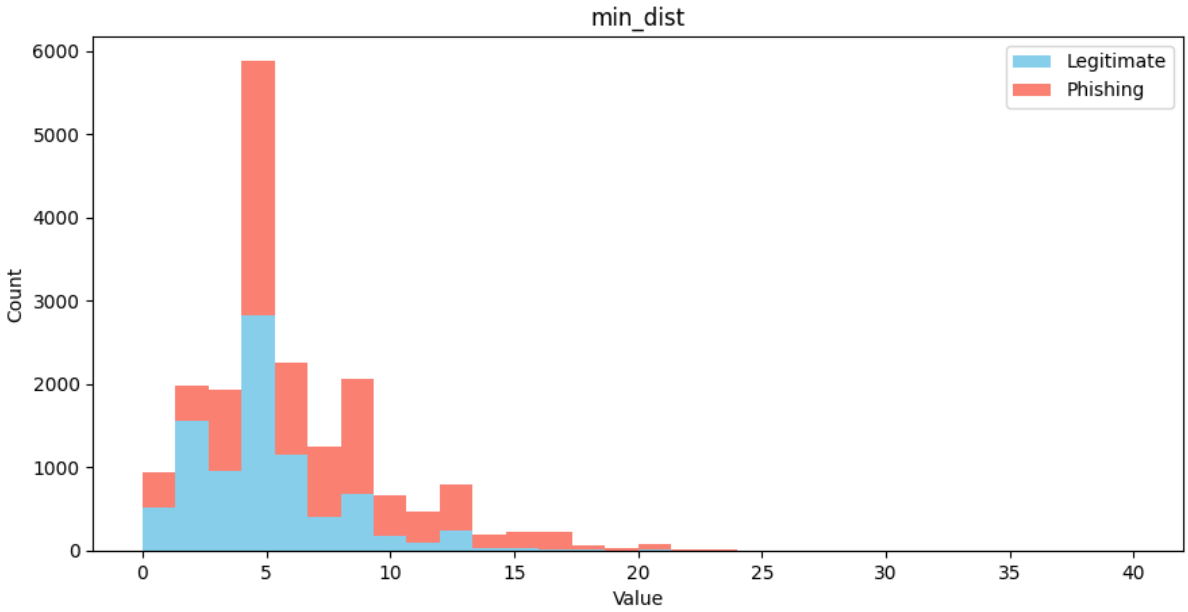

Рисунок 3.30 - Розподіл мінімальної відстані до популярних сервісів

Для кількості посилань у повідомленні встановимо границю у 1, тобто усі повідомлення з посиланнями будуть маркуватися як фішингові.

```
y_pred = np.where(extra_features['num_urls'] < 1, 0, 1)
accuracy = accuracy_score(y_test, y_pred)
CM = confusion_matrix(y_test, y_pred)
print('\n----- Number of urls -----')
print(f'Accuracy: {accuracy}')
print(f'True positives: {CM[1][1]}    True negatives: {CM[0][0]}')
print(f'False positives: {CM[0][1]}    False negatives: {CM[1][0]}')
```

Рисунок 3.31 - Встановлення границі для кількості посилань

Отримали точість у 53.15% з досить рівномірним розподілом у матриці невідповідностей.

```
----- Number of urls -----
Accuracy: 0.5314857502095558
True positives: 7292    True negatives: 2853
False positives: 5863    False negatives: 3080
```

Рисунок 3.32 - Матриця невідповідностей для кількості посилань

Для SPF флагу будемо лише перевіряти його наявність.

```
y_pred = extra_features['has_spf']
accuracy = accuracy_score(y_test, y_pred)
CM = confusion_matrix(y_test, y_pred)
print('\n----- SPF record -----')
print(f'Accuracy: {accuracy}')
print(f'True positives: {CM[1][1]}    True negatives: {CM[0][0]}')
print(f'False positives: {CM[0][1]}    False negatives: {CM[1][0]}')
```

Рисунок 3.33 - Встановлення границі для наявності SPF запису

Це дає нам точність у 52.22% з також досить рівномірним розподілом.

```
----- SPF record -----
Accuracy: 0.5221605196982397
True positives: 7420    True negatives: 2547
False positives: 5785    False negatives: 3336
```

Рисунок 3.34 - Матриця невідповідностей для SPF запису

Для кількості підозрілих слів у повідомленні встановимо границю у 1, тобто усі повідомлення з 0 підозрілих слів будуть маркуватися як фішингові, усі інші – як легітимні.

```
y_pred = np.where(extra_features['suspicious_count'] > 1, 0, 1)
accuracy = accuracy_score(y_test, y_pred)
CM = confusion_matrix(y_test, y_pred)
print('\n----- Number of suspicious keywords -----')
print(f'Accuracy: {accuracy}')
print(f'True positives: {CM[1][1]}    True negatives: {CM[0][0]}')
print(f'False positives: {CM[0][1]}    False negatives: {CM[1][0]}')
```

Рисунок 3.35 - Встановлення границі для підозрілих слів

Це дає нам точність у 56.59% з малою кількістю True та False negatives.

```
----- Number of suspicious keywords -----
Accuracy: 0.5658528918692373
True positives: 10243    True negatives: 558
False positives: 8158    False negatives: 129
```

Рисунок 3.36 - Матриця невідповідності для пудозрілих слів

Для співпадіння адреси відправника з посиланнями, усі повідомлення зі співпадінням, будемо позначати як легітимні, усі інші – як фішингові.

```
y_pred = (~extra_features['domain_match'].astype(bool)).astype(int)
accuracy = accuracy_score(y_test, y_pred)
CM = confusion_matrix(y_test, y_pred)
print('\n----- Domain match -----')
print(f'Accuracy: {accuracy}')
print(f'True positives: {CM[1][1]}    True negatives: {CM[0][0]}')
print(f'False positives: {CM[0][1]}    False negatives: {CM[1][0]}')
```

Рисунок 3.37 - Встановлення границі для відповідності домену відправника

Це дає нам точність у 65.84% майже без False negatives (оскільки майже усі повідомлення зі співпадінням є легітимними)

```
----- Domain match -----
Accuracy: 0.6583717518860017
True positives: 10365    True negatives: 2202
False positives: 6514    False negatives: 7
```

Рисунок 3.38 - Матриця невідповідностей для відповідності домену відправника

Для відстані до популярних доменів, розділимо повідомлення таким чином, що усі повідомлення з менше ніж 5 посиланнями будуть позначатися як легітимні, усі інші – як фішингові.

```
y_pred = np.where(extra_features['min_dist'] < 5, 0, 1)
accuracy = accuracy_score(y_test, y_pred)
CM = confusion_matrix(y_test, y_pred)
print('\n----- Domain distance -----')
print(f'Accuracy: {accuracy}')
print(f'True positives: {CM[1][1]}    True negatives: {CM[0][0]}')
print(f'False positives: {CM[0][1]}    False negatives: {CM[1][0]}')
```

Рисунок 3.39 - Встановлення границі для відстані до популярних доменів

Це також дає досить непогану точність у 60.16% з більш-менш рівномірним розподілом.

```
----- Domain distance -----
Accuracy: 0.6016345347862532
True positives: 6605    True negatives: 4879
False positives: 3837    False negatives: 3767
```

Рисунок 3.40 - Матриця невідповідності для відстані до популярних доменів

Машинне навчання лише на додаткових ознаках

Створимо просту модель машинного навчання на основі RandomForest, де ознаками будуль лише вищеперераховані методи, для їх комбінації.

Для попередньої обробки даних, будемо зберігати лише додаткові ознаки, без тексту.

```
extra_features = pd.DataFrame(extra_features, columns=['num_urls', 'suspicious_count', 'domain_match', 'min_dist'])
extra_features['label'] = data['label'].values

with open(output_file, 'wb') as f:
    pickle.dump( obj: (extra_features.drop(columns=['label']), extra_features['label']), f)

print(f'Preprocessed data saved to {output_file}')
```

Рисунок 3.41 - Попередня обробка лише додаткових ознак

Тренування та тестування моделі будуть аналогічними

```python
1 usage
def train_model(data_file, model_file):
    with open(data_file, 'rb') as f:
        X, y, _ = pickle.load(f)

    X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.2)

    model = RandomForestClassifier()
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    CM = confusion_matrix(y_test, y_pred)
    print(f'Model Accuracy: {accuracy}')
    print(f'True positives: {CM[1][1]}    True negatives: {CM[0][0]}')
    print(f'False positives: {CM[0][1]}    False negatives: {CM[1][0]}')

if __name__ == '__main__':
    train_model( data_file: '../data/preprocessed_extra_features.pkl', model_file: '../models/random_forest_extra_features.pkl')
```

Рисунок 3.42 - Тренування та тестування моделі лише на додаткових ознаках

Досягли точності у 82.48%, що досить непогано, проте містить у 34 рази більше помилок, ніж наша найкраща комбінована модель.

```
Model Accuracy: 0.8247773703509691
True positives: 1788    True negatives: 1361
False positives: 408    False negatives: 261
```

Рисунок 3.43 - Матриця невідповідності для моделі лише на додаткових ознаках

Машинне навчання без додаткових ознак

Початкова обробка даних буде дуже спрощена – лише векторизація тексту та збереження векторизовних даних.

```
1 usage
def preprocess_data(input_file, output_file):
    data = pd.read_csv(input_file).dropna()

    data['text_combined'] = data['subject'] + ' ' + data['body']

    vectorizer = TfidfVectorizer()
    X_tfidf = vectorizer.fit_transform(data['text_combined']).toarray()

    y = data['label'].values

    with open(output_file, 'wb') as f:
        pickle.dump( obj: (X_tfidf, y, vectorizer), f)

    print(f'Preprocessed data saved to {output_file}')


if __name__ == '__main__':
    preprocess_data( input_file: '../data/CEAS_08.csv',  output_file: '../data/preprocessed_data_no_extra_features.pkl')
```

Рисунок 3.44 - Попередня обробка лише тексту

Тренування та тестування без змін

```
def train_model(data_file, model_file):
    with open(data_file, 'rb') as f:
        X, y, _ = pickle.load(f)

    X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.2)

    model = RandomForestClassifier()
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    CM = confusion_matrix(y_test, y_pred)
    print(f'Model Accuracy: {accuracy}')
    print(f'True positives: {CM[1][1]}    True negatives: {CM[0][0]}')
    print(f'False positives: {CM[0][1]}    False negatives: {CM[1][0]}')

if __name__ == '__main__':
    train_model( data_file: '../data/preprocessed_data_no_extra_features.pkl',  model_file: '../models/random_forest_no_extra_features.pkl')
```

Рисунок 3.45 - Тренування та тестування моделі лише на тексті

Точність моделі без додаткових ознак склала 97.62%, що може розглядатися як достатня, проте наша комбінована система все ще помиляється у 4.5 рази менше.

```
Model Accuracy: 0.9762386248736097
True positives: 2193    True negatives: 1669
False positives: 42    False negatives: 52
```

Рисунок 3.46 - Матриця невідповідності для моделі лише на тексті

Висновки до розділу 3

У третьому розділі було реалізовано повний цикл створення системи виявлення фішингових повідомлень, що поєднує класичні ознаки, структурний аналіз та методи машинного навчання. Зокрема:

- Проведено аналіз обраного датасету CEAS_08, виявлено його сильні сторони (наявність тексту і метаданих) та обмеження (застарілість, відсутність вкладень).
- Реалізовано попередню обробку даних, зокрема: об’єднання тексту теми та тіла, векторизація за допомогою TF-IDF, а також побудова структурних ознак (кількість посилань, SPF, підозрілі слова, схожість доменів тощо).
- Порівняно ефективність кількох моделей машинного навчання: Logistic Regression, Random Forest та XGBoost. Найкращу точність (≈99.5%) показала модель XGBoost з комбінованими ознаками.
- Продемонстровано, що навіть прості класичні правила (наприклад, наявність SPF чи кількість посилань) мають певну предиктивну силу, але в ізоляції їх точність не перевищує 66%.
- Доведено ефективність комбінованого підходу: об'єднання текстового аналізу з технічними ознаками дозволяє значно покращити точність та зменшити кількість хибних спрацьовувань у порівнянні з окремими методами.

Таким чином, практичне тестування підтвердило гіпотезу про доцільність використання гібридних систем для виявлення фішингу: класичні методи мають залишатися частиною захисту, але саме алгоритми ML дають змогу адаптуватися до нових атак та шаблонів.

ВИСНОВКИ

У ході виконання дипломної роботи було проведено глибоке дослідження проблеми фішингових атак, їхнього впливу на інформаційну безпеку та сучасних підходів до їх виявлення. Робота поєднала в собі теоретичний аналіз наявних методів протидії з практичною реалізацією системи виявлення фішингових листів, побудованої на базі методів машинного навчання.

Розглянувши існуючі підходи до протидії фішингу, вдалося виявити ключові переваги та недоліки rule-based та сигнатурних методів, які хоч і забезпечують певний базовий рівень захисту, але часто не здатні виявляти нові або цілеспрямовані атаки. Це створило передумови для розробки комбінованого рішення, яке враховує не лише зміст повідомлення, а й низку технічних характеристик — наявність записів SPF, подібність доменів, підозрілі ключові слова, кількість URL-посилань, тощо.

Було побудовано модель, яка ефективно поєднує векторизацію текстових даних (через TF-IDF) з ручним створенням спеціалізованих ознак на основі аналізу структури листа та технічної інформації. На основі відкритого датасету CEAS_08 було реалізовано повний цикл обробки, моделювання та тестування, що дало змогу досягти точності понад 99.5% з використанням XGBoost. Цей результат свідчить про доцільність використання комбінованого підходу, що дозволяє досягти значно кращих показників у порівнянні з традиційними методами.

Таким чином, результати дипломної роботи підтвердили як актуальність обраної теми, так і ефективність обраного підходу до виявлення фішингових листів на основі машинного навчання та інженерії ознак.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. VirusTotal. https://virustotal.com/
2. Hideez. Що означає фішинг? Визначення та основні типи фішингу. https://hideez.com/uk-ua/blogs/news/phishing-explained
3. Microsoft. Що таке фішинг? Захисний комплекс Microsoft. https://www.microsoft.com/uk-ua/security/business/security-101/what-is-phishing
4. Verizon. 2025 Data Breach Investigations Report. https://www.verizon.com/business/resources/T254/reports/2025-dbir-data-breach-investigations-report.pdf
5. IBM. X-Force 2025 Threat Intelligence Index. https://www.ibm.com/downloads/documents/us-en/1227cc9e83cb97ae
6. Cloudflare. What are DMARC, DKIM and SPF? https://www.cloudflare.com/learning/email-security/dmarc-dkim-spf/
7. IBM. What is UEBA (User and Entity Behavior Analytics)? https://www.ibm.com/topics/ueba
8. Fortinet. What is Signature-based Detection? https://docs.fortinet.com/document/fortigate/7.4.2/ngfw-atp-concept-guide/756476/signature-based-detection
9. Palo-Alto Network. Policies to Detect Threats. https://docs.paloaltonetworks.com/saas-security/behavior-threats/policies-to-detect-threats
10. Apache SpamAssassin. https://spamassassin.apache.org/
11. ClamAV. Open Source Antivirus Engine for Detecting Trojans, Viruses, Malware & Other Malicious Threats. https://www.clamav.net/

12. Cisco. Secure Email Threat Defense (formerly ESA). https://www.cisco.com/c/en/us/products/security/email-security/index.html
13. Google. Jigsaw's Phishing Quiz. https://phishingquiz.withgoogle.com/
14. Proofpoint. State of the Phish. https://www.proofpoint.com/sites/default/files/threat-reports/pfpt-us-tr-state-of-the-phish-2024.pdf
15. Kaggle. Phishing Email Dataset. https://www.kaggle.com/datasets/naserabdullahalam/phishing-email-dataset
16. GitHub. Top 1000 domains. https://gist.githubusercontent.com/jgamblin/62fadd8aa321f7f6a482912a6a317ea3/raw/33c6752125188cfdacdeee3f4fd6e01909e50eef/urls.txt
17. Dnspython. https://www.dnspython.org/
18. Pickle. https://docs.python.org/3/library/pickle.html