Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

# БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

Розробка системи моніторингу електронної пошти на виявлення фішингових атак за допомогою методів машинного навчання

Виконала: студентка 4 курсу, гр.2КІТС-20б
спеціальності 125– Кібербезпека
Освітня програма – Кібербезпека
інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)

Марчук В.О.
(прізвище та ініціали)

Керівник: д.ф., доц. каф. МБІС

Салієва О.В.
(прізвище та ініціали)

« 6 » червня 2024 р.

Рецензент: к.т.н., доц. каф. ОТ

Городецька О.С.
(прізвище та ініціали)

« 10 » червня 2024 р.

**Допущено до захисту**
Голова секції УБ кафедри МБІС
д.т.н., проф. Яремчук Ю.Є.
« 10 » червня 2024р.

Вінниця ВНТУ – 2024

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти І-й (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

**ЗАТВЕРДЖУЮ**
**Голова секції УБ, кафедра МБІС**

**д.т.н., проф. Яремчук Ю.Є.**
“ 22 ” березня 2024 р.

# З А В Д А Н Н Я
**на бакалаврську дипломну роботу студенту**
Марчук Валерії Олегівні
(прізвище, ім`я, по-батькові)

1. Тема роботи «Розробка системи моніторингу електронної пошти на виявлення фішингових атак за допомогою методів машинного навчання»
Керівник роботи д.ф., ст. викл. каф. МБІС Салієва Ольга Володимирівна
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “ 11 ” березня 2024 року № 80
2. Термін подання студентом роботи 6 червня 2024 р.
3. Вихідні дані до роботи Електронні джерела та наукові статті по темі. Існуюче програмне забезпечення, яке стосується теми бакалаврської дипломної роботи.
4. Зміст текстової частини: Для досягнення мети було поставлено наступні задачі: дослідити актуальність обраної теми; проаналізувати процеси фішингу електронної пошти та визначити життєвий цикл фішингових атак; дослідити підходи та методи розпізнавання фішингових атак в електронній пошті за допомогою методів машинного навчання; здійснити вдосконалення алгоритму роботи системи моніторингу електронної пошти; розробити систему моніторингу електронної пошти на виявлення фішингових атак; здійснити тестування ефективності розробленої системи.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень) Алгоритм роботи модуля експорту та інтеграції розробленої системи, Алгоритм роботи модуля виявлення фішингових ознак розробленої системи, Дерево модуля виявлення фішингових ознак розробленої системи, алгоритм роботи пропонованого підходу до розробки системи.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Розділ I | Салієва О. В. д.ф., доц. каф. МБІС | | |
| Розділ II | Салієва О. В. д.ф., доц. каф. МБІС | | |
| Розділ III | Салієва О. В. д.ф., доц. каф. МБІС | | |

7. Дата видачі завдання 22 березня 2024 р.

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва та зміст етапу | Термін виконання | | Примітка |
|---|---|---|---|---|
| | | початок | закінчення | |
| 1 | Аналіз предметної області | 22.03.2024 | 29.03.2024 | Виконано |
| 2 | Аналіз методів розв'язання задачі | 30.03.2024 | 09.04.2024 | Виконано |
| 3 | Постановка задач розробки програмної реалізації системи | 10.04.2024 | 19.04.2024 | Виконано |
| 4 | Дослідження інформаційних джерел | 20.04.2024 | 09.05.2024 | Виконано |
| 5 | Аналіз варіантів роботи системи | 10.05.2024 | 14.05.2024 | Виконано |
| 6 | Розробка алгоритму роботи | 15.05.2024 | 19.05.2024 | Виконано |
| 7 | Реалізація програми | 20.05.2024 | 23.05.2024 | Виконано |
| 8 | Тестування програми | 24.05.2024 | 26.05.2024 | Виконано |
| 9 | Оформлення матеріалів до захисту БДР | 27.05.2024 | 31.05.2024 | Виконано |

**Студент** _____ Марчук В.О.
(підпис) (прізвище та ініціали)

**Керівник роботи** _____ Салієва О.В.
(підпис) (прізвище та ініціали)

# АНОТАЦІЯ

УДК 004.93

Бакалаврська дипломна робота складається з 95 сторінок формату А4, на яких є 26 рисунків, 2 таблиці, список використаних джерел містить 39 найменувань.

Дана бакалаврська дипломна робота присвячена розробці системи моніторингу електронної пошти, яка використовує передові методи машинного навчання для виявлення фішингових атак. Метою дослідження є розробка ефективної системи, яка забезпечить високий рівень точності виявлення шахрайських повідомлень, мінімізуючи при цьому помилкові результати.

Дипломна робота включає аналіз сучасних систем моніторингу електронної пошти та їх використання для виявлення фішингових атак. Особлива увага приділяється порівнянню ефективності існуючих рішень та визначенню потенційних областей для покращення. Також розглядаються різні підходи і методи виявлення фішингу.

Програма розроблена в рамках цієї роботи, реалізує інноваційний підхід до виявлення фішингових листів, використовуючи поєднання результатів трьох алгоритмів машинного навчання. Він забезпечує автоматичне виявлення підозрілих повідомлень, аналізуючи великі обсяги даних та визначаючи нетипові шаблони поведінки користувачів та структуру листів. У роботі детально описано процес збору та обробки даних, вибір та тренування моделі машинного навчання, а також тестування роботи системи. Представлено результати експериментів, які підтверджують високу ефективність запропонованої системи у порівнянні з традиційними методами виявлення фішингу.

Ключові слова: фішингові атаки, машинне навчання, моніторинг електронної пошти, TensorFlow, imaplib.

## ABSTRACT

UDC 004.93

The bachelor's thesis consists of 95 A4 pages, which have 26 figures, 2 tables, the list of sources used contains 39 items.

This bachelor's thesis is devoted to the development of an e-mail monitoring system that uses advanced machine learning methods to identify and block phishing attacks. The goal of the study is to develop an effective system that will ensure a high level of accuracy in detecting fraudulent messages while minimizing false positive results.

The thesis includes an analysis of modern e-mail monitoring systems and their use to detect phishing attacks. Particular attention is paid to comparing the effectiveness of existing solutions and identifying potential areas for improvement. Different approaches and methods of detecting phishing are also considered.

The application developed as part of this work implements an innovative approach to detecting phishing emails using a combination of three machine learning algorithms. It provides automatic detection of suspicious messages by analyzing large volumes of data and identifying atypical patterns of user behavior and email structure. The work describes in detail the process of data collection and processing, the selection and training of a machine learning model, as well as testing of the system. The results of experiments are presented, which confirm the high efficiency of the proposed system in comparison with traditional methods of detecting phishing.



Keywords: phishing attacks, machine learning, email monitoring, TensorFlow, imaplib.

# ЗМІСТ

# ВСТУП

**Актуальність**. У сучасному світі електронна пошта стала одним з основних каналів комунікації, що робить її мішенню для різноманітних кіберзлочинів, зокрема фішингових атак. Фішинг - це вид шахрайства, який використовує електронну пошту для обману користувачів з метою розголошення конфіденційної інформації, такої як паролі, номери кредитних карток або інші особисті дані.

Фішингові атаки стають дедалі більш витонченими та складними для виявлення, що робить їх серйозною загрозою для користувачів електронної пошти та організацій. Традиційні методи виявлення фішингу, такі як чорні списки та фільтри за ключовими словами, стають менш ефективними, оскільки зловмисники постійно вдосконалюють свої методи.

Використання методів машинного навчання (МН) може допомогти подолати цю проблему. МН - це галузь штучного інтелекту, яка дозволяє комп'ютерам навчатися на даних та робити прогнози без чіткої програми. Застосування МН до виявлення фішингу може допомогти автоматизувати процес аналізу електронної пошти та з високою точністю виявляти підозрілі листи.

**Мета** даної дипломної роботи полягає в розробці системи моніторингу електронної пошти на виявлення фішингових атак за допомогою методів машинного навчання.

Для досягнення цієї мети буде вирішено наступні **задачі**:

– проаналізувати процеси фішингу електронної пошти та визначити життєвий цикл фішингових атак;

– дослідити підходи та методи розпізнавання фішингових атак в електронній пошті за допомогою методів машинного навчання;

– розробити алгоритм роботи системи моніторингу електронної пошти;

– розробити систему моніторингу електронної пошти на виявлення фішингових атак;

– оцінити ефективність розробленої системи.

**Об’єктом дослідження** є методи виявлення фішингових атак в електронній пошті за допомогою методів машинного навчання.

**Предметом дослідження** є розробка системи моніторингу електронної пошти, яка використовує методи машинного навчання для автоматичного виявлення та класифікації фішингових листів.

**Наукова новизна** дослідження полягає в розробці системи моніторингу електронної пошти на виявлення фішингових атак, яка використовує комбінацію трьох методів машинного навчання, що дозволяє підвищити точність та ефективність виявлення фішингових листів.

**Практична цінність** дослідження полягає в тому, що розроблена система моніторингу може бути використана для захисту користувачів електронної пошти та організацій від фішингових атак. Система може бути інтегрована з існуючими системами захисту електронної пошти або використовуватися як самостійний продукт.

**Апробація:** тези доповідей даної тематики представлені на LII Всеукраїнській науково-технічній конференції факультету менеджменту та інформаційної безпеки [1].

# 1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ ВИЯВЛЕННЯ ФІШИНГОВИХ ЛИСТІВ

У першому розділі бакалаврської роботи буде проведено аналіз сучасного стану проблеми фішингу, досліджено існуючі методи виявлення фішингових атак, зокрема традиційних та заснованих на машинному навчанні. Також оцінено ефективність цих методів.

На основі проведеного аналізу буде сформовано висновки та здійснено постановку задач.

## 1.1 Актуальність дослідження фішингових атак

Фішинг – це метод шахрайства в інтернеті, який використовується для “виловлювання” конфіденційної інформації. Шахраї створюють підроблені електронні листи та веб-сайти, які виступають в якості “приманки”, а облікові записи жертви – як мережевий фішинг. Фішинг є частиною соціальної інженерії, де шахраї намагаються вкрасти конфіденційну інформацію, видаючи себе за законну особу, зазвичай представника відомих фінансових установ або електронної комерції. Основна мета - отримати гроші обманним шляхом [1].

Фішингові атаки можуть приймати дві форми [2]:

1. Спроби обдурити жертву, щоб змусити її розкрити свої секрети, видаючи себе за надійних осіб, які справді потребують такої інформації.
2. Спроби отримати секрети шляхом встановлення шкідливих програм на комп’ютери жертв.

PayPal ™, eBay ™, American Online ™, ABSA ™, Standard Bank ™, Google ™, Microsoft - це лише кілька випадків, коли організації та їх клієнти фінансово постраждали від фішингових атак. Хоча більшість фішингових атак націлена на фінанси, шахраї можуть бути зацікавлені і в інших приватних даних. Шахраї постійно винаходять все більш ефективні та нові методи, використовуючи всі сучасні технології, включаючи месенджери, трояни та популярні соціальні мережі [3].

Проєктування та виконання фішингової атаки не вимагає від шахраїв великої попередньої підготовки, як інтелектуальної так і технічної. Головним інструментом є обман, маніпуляції та людські емоції, що допомагає фішерам змусити жертву розкрити приватну інформацію [3].

Цінна інформація, така як облікові дані для аутентифікації користувача та особиста конфіденційна інформація, може бути отримана шляхом використання вразливостей в розумінні користувачем системи, зокрема, нерозуміння призначеного для користувача інтерфейсу. Оскільки бар'єр для використання вразливостей системи з часом значно збільшився, напад на користувачів швидко стало більш дієвою і ефективною альтернативою. Щоб захистити користувачів від фішингових атак, розробники системи та фахівці з безпеки повинні розуміти, як користувачі взаємодіють з цими атаками [4].

На сьогоднішній день фішинг є великою проблемою в кіберпросторі. Злочинці можуть полювати як на окремих користувачів (spear phishing), так і на цілі підприємства та урядові організації. Хоча обізнаність громадськості про фішингові атаки в цілому зростає, нові та новаторські схеми або "атаки нульового дня", які обходять фішингові фільтри та чорні списки, часто все ж дозволяють обдурити навіть досвідчених користувачів [5].

Звичайним методом фішингової атаки є спам-розсилка. Спам - це небажана електронна пошта [6]. Хоча його можна використовувати для поширення фішингового шахрайства, його мета набагато ширше. Спам-повідомлення в основному використовуються для комерційної реклами та інших форм некомерційних цілей. Різниця між звичайним спамом і фішингом полягає в тому, що автори спаму не намагаються вкрасти конфіденційну інформацію у одержувачів. Навпаки, фішингові атаки призначені для отримання призначеної для користувача інформації, такої як ідентифікатори входу та паролі, щоб зловмисник міг отримати доступ до таких речей, як соціальна мережа та дані банківського рахунку [6].

Фішинг через електронну пошту стає все більш поширеним явищем. Це один з найчастіше використовуваних методів шахрайства, метою якого є крадіжка

персональних та фінансових даних. Люди, які відповідають на фішингові електронні листи та вводять запитувану інформацію в електронні листи, веб-сайти або спливаючі вікна, піддають себе та свої організації ризику [7].

Незважаючи на значні зусилля, спрямовані на виявлення фішингових листів, не існує універсального набору критеріїв, які були б визначені як найефективніші для виявлення фішингу. Тому постійно виникає необхідність вдосконалювати методи виявлення [6].

Цей тип атак викликав найбільше стурбованості в останньому звіті про злочини в Інтернеті за 2021 рік, опублікованому Центром скарг на злочини в Інтернеті (IC3) Федерального бюро розслідувань США. За даними статистики, зібраної IC3 ФБР за 2021 рік, крадіжка, шахрайство та експлуатація через Інтернет залишаються широко поширеними і призвели до вражаючих фінансових втрат в 2,7 мільярда доларів в 2021 році [8].

Робоча група з протидії фішингу (APWG) вказує, що за останні роки фішингові атаки зросли. Це число поступово зростає з 162 155 випадків у останньому кварталі 2022 року до 165 772 випадків у першому кварталі 2023 року. Фішинг завдав серйозних збитків багатьом організаціям та світовій економіці [7].

Фішингові інциденти поширені з вагомих причин. По-перше, цей тип атак зазвичай дешевий у реалізації в порівнянні з іншими атаками, які обходять системи захисту периметра, такі як міжмережеві екрани, IDS/IPS та інші. По-друге, процедура атаки може повторюватися кілька разів і на декількох користувачів системи. Це значно збільшує ймовірність того, що будь-який співробітник в організації може бути обманутий. По-третє, оскільки електронна пошта часто використовується для відправки запитів, велика частина користувачів схильна довіряти запитам, що відправляються по електронній пошті, і фактично намагається виконати запитані дії [4].

Провівши велику кількість тестів компанія «Cofence» показує наскільки сприйнятливі показники в цілому. В даному тесті було надіслано 135 мільйонів підроблених фішингових листів, і у 12% випадків користувачі відвідали

потенційно небезпечні веб-сайти або виконали інші підозрілі дії. Справитися з загрозою фішингу складно. Переваги електронної пошти та інших електронних комунікацій значні, і, як було виявлено в ході опитування користувачів комп'ютерів, стратегія відмови від переходу по посиланнях в отриманих електронних листах не може використовуватися без розбору, оскільки такі посилання можуть сприйматися як необхідні для роботи. Однак схильність до фішингу може значно варіюватися в залежності від одержувача, ситуації і фішингового контенту. Наприклад, в деякому дослідженні тільки 0,3% цілей були обмануті електронним листом, що містить шахрайство з кредитною карткою, тоді як 37% були обмануті шахрайством про реєстрацію на курс. Компанія Cofense повідомила, що їй вдалося домогтися успіху тільки в 5,3% випадків фішингу проти співробітників в енергетичній галузі, тоді як в сфері охорони здоров'я показник успіху склав 15,0% . Cofense також повідомив про значні зміни в часі, що вказує на ситуативний фактор. Наприклад, відсоток успіху електронного листа з онлайн-замовленням з вкладенням склав 4,4% в першому кварталі 2021 року, а в другому кварталі збільшився до 18,4% [9].

### 1.2 Поняття фішингу та життєвий цикл фішингових атак

Фішинг сьогодні розглядається як серйозна загроза, що швидко зростає з кожним роком. Це злочинна діяльність, яка поєднує методи соціальної інженерії та технічні прийоми для викрадення конфіденційних даних користувачів, таких як імена користувачів і паролі [10].

Фішингові електронні листи належать до категорії спам-повідомлень. Користувачі отримують електронні листи, нібито надіслані від законної компанії або банку, із проханням перейти за вбудоване посилання. Посилання перенаправить користувача на фальшивий веб-сайт, який запитує конфіденційну інформацію, наприклад, імена користувачів, паролі або номери кредитних карт.

Процес починається з надсилання електронних листів на поштові скриньки цільових осіб із метою змусити їх перейти за посилання. Таким чином, онлайн-фішинг дуже схожий на традиційну риболовлю; у той час як остання

використовує приманку та волосінь для ловлі риби, в онлайн-методі фішер розсилає якомога більше електронних листів, щоб переконати якомога більшу кількість одержувачів "спіймати" приманку та перейти за посиланням). На рисунку 1.1 зображено цикл фішингової атаки [10].

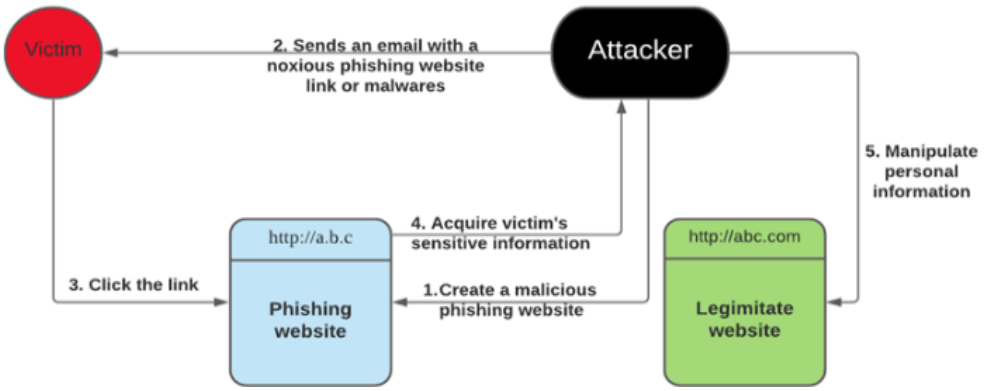



Рисунок 1.1 – Життєвий цикл фішингу [22]

Для досягнення своїх цілей фішери використовують різні технології, які включають [11]:

– маніпуляції з URL-посиланнями. Основний фокус фішингу – це посилання. Існує декілька винахідливих способів маніпулювати URL-адресою, щоб вона виглядала законною. Одним з таких методів є представлення шкідливих URL-адрес у вигляді гіперпосилань з назвою веб-сайту. Інший метод полягає в використанні URL-адрес з помилками, які виглядають як допустимі URL-адреси, наприклад, ghoogle.com. Варіант тайпсквоттінга, який значно важче виявити, ніж вищезгадані методи маніпулювання посиланнями, називається спуфінгом IDN. При цьому зловмисники використовують символи неанглійської мови, які виглядають точно так само, як англійські символи, наприклад, використовуючи кирилицю “с” або “а” замість англійських аналогів;

– підробка веб-сайту – цей тип атаки здійснюється на законному веб-сайті шляхом маніпулювання кодом JavaScript цільового веб-сайту. Ці типи атак, відомі як XSS, дуже важко виявити, оскільки жертва використовує законний веб-сайт;

– приховане перенаправлення – ця атака спрямована на веб-сайти, які використовують протоколи OAuth 2.0 та OpenID. Намагаючись отримати доступ до токена на легітимному веб-сайті, користувачі передають свій токен шкідливій службі. Однак цей метод не отримав особливої уваги через його низьку ефективність;

– соціальна інженерія – цей вид фішингу використовує соціальну взаємодію. Він використовує психологічні прийоми для обману користувачів, змушуючи їх надавати інформацію про безпеку. Цей тип атаки відбувається в кілька етапів. Спочатку фішер досліджує потенційні слабкі місця цілей, необхідні для атаки. Потім фішер намагається завоювати довіру цілі і, нарешті, створює ситуацію, в якій ціль розкриває важливу інформацію. Існують деякі методи фішингу за допомогою соціальної інженерії, такі як цькування, залякування та цільовий фішинг.

Фішингова атака, як правило, проходить через наступні п’ять етапів [12]:

1. Створення підроблених веб-сайтів. Зловмисники створюють веб-сайти, що виглядають як оригінальні, включаючи конфігурацію веб-сервера, використання DNS-імені та створення веб-сторінок, що імітують оригінальний веб-сайт.

2. Відправка підроблених електронних листів. Фішери відправляють велику кількість підроблених електронних листів, використовуючи ілюзію законності, засновану на зовнішньому вигляді електронного листа (структура адреси електронної пошти, заголовок теми, зміст), логотипи установи, термінологія тощо.

3. Запит на введення особистої інформації. Користувачів просять ввести свою особисту інформацію, таку як ім’я користувача, пароль, номер соціального страхування, номер банківського рахунку тощо, за допомогою HTML-форми, вбудованої в текст повідомлення або як вкладення.

4. Спокушання користувача. Користувача спонукають клацнути на гіперпосилання, яке зазвичай маскується під текстом або зображенням, наприклад: «Клацніть тут для перевірки».

5. Збір особистої інформації. Користувачі, які піддалися спокусі, розкривають свою особисту інформацію, відвідуючи веб-сайт та надаючи необхідну інформацію. Зловмисники збирають цю інформацію та використовують її для різних видів шахрайства.

Фішингові атаки, як правило, реалізуються через шахрайські електронні листи, тому зловмисникам необхідний метод, за допомогою якого вони можуть залучити користувачів, створивши фішинговий веб-сайт, що імітує існуючий легітимний веб-сайт. Зловмисники, як правило, використовують методи обфускації, щоб обманути користувача при натисканні на фішинговий URL. Вони можуть замінити ім’я хоста IP-адресою або включити доменне ім’я цільового бренду в шлях до URL-адреси з помилками орфографії або без них [12].

Фішингові сторінки, як правило, мають декілька перенаправлень для перенаправлення з початкової URL-адреси на кінцевий сайт, який розміщується зловмисником на будь-якому скомпрометованому сервері. Зловмисники також інфікують легітимні веб-сайти прихованим шкідливим кодом JavaScript, який вбудовує “iframe” з URL-адресою домену зловмисника, а потім видає перенаправлення HTTP 302 для завантаження фішингового веб-сайту [13].

На сьогоднішній день майже всі отримані електронні листи базуються на мові розмітки HTML. Фішери використовують HTML-теги для візуального обману користувачів. Наприклад, гіперпосилання є активними та натискаються, маскуючи фактичну URL-адресу. Наявність форми входу в систему, наявність полів пароля і наявність ненормального вмісту скриптів може допомогти у виявленні фішингових веб-сторінок та листів.

Більшість фішингових веб-сторінок містять форми з полями для введення даних платіжної картки користувача або пароля. Крім того, використовуються різні методи, такі як приховані елементи, спливаючі вікна і підказки для введення конфіденційної інформації, щоб спонукати користувачів ввести паролі і конфіденційну інформацію.

Аномалії JavaScript, наявність коду оболонки на сторінці і підозрілі елементи управління Active X також можуть вказувати на фішингову сторінку, коли зловмисник використовує будь-яку вразливість на веб-сторінці для впровадження скриптів / коду, які можуть завантажувати конфіденційну інформацію користувача зі сторінки в сервер зловмисника [13].

Також, досить вагомою ознакою при виявленні фішингу є характеристики записів WHOIS і DNS веб-сайту, які дозволяють зрозуміти, як і де розміщуються фішингові веб-сайти. Послуги WHOIS надаються реєстраторами та реєстрами спонсорованих ними доменних імен. Записи WHOIS містять інформацію про власника реєстрації веб-сайту, дати створення і закінчення терміну реєстрації, а також деякі інші деталі [13].

Зазвичай використовуються такі функції WHOIS, як вік домену, тривалість життя домену, відомості про реєстратора та деякі інші, а також функції запису DNS, такі як номер автономної системи, адреса і місце розташування сервера імен, час життя запису DNS і т.д.

Фішери зазвичай націлені на зламі доменів хостингу для запуску своїх атак, так що отримання інформації про користувачів через фішинг веб-сайт є простим. Фішингові сайти створюються на короткий проміжок часу, щоб отримати від них максимум за кілька днів до того, як сайт буде виявлений і заблокований. Фішингові кампанії недовговічні, оскільки фішери не можуть дозволити собі платити за хостинг-домен протягом тривалого періоду [14].

Функції виявлення фішингу або їхній набір вважаються надійними, якщо зловмисник не може без зусиль створити фішинговий веб-сайт, який має такі ж функції, як і безпечний веб-сайт, або якщо для створення веб-сайту, який з великою ймовірністю не відрізняється від легітимного сайту, зловмисник потребує значних витрат.

Більшість функцій URL-адрес, таких як кількість крапок в URL-адресі, довжина URL-адреси та інші, можуть бути змінені зловмисником таким чином, щоб вони виглядали як функції безпечного веб-сайту. Тому ці функції, якщо розглядати їх окремо, не є надійними. Однак у певних випадках, наприклад, коли

зловмисник використовує довгі URL-адреси для обману користувачів, ці функції, якщо їх розглядати в комплексі, можуть бути більш надійними, ніж окремі функції.

Аналогічно, з огляду на функції DNS, власник веб-сайту має можливість змінювати поштовий сервер, сервер імен та записи PTR для свого веб-сайту, і тому ці функції не є надійними. Зловмисники, які самостійно розміщують та володіють своїми фішинговими веб-сайтами, можуть змінити ці функції так, щоб вони виглядали як функції безпечного сайту [14].

Таким чином, обговорення надійності різних категорій ознак є необхідним для визначення набору надійних функцій, які можна ефективно використовувати для виявлення фішингових веб-сайтів та електронних листів.

## 1.3 Методи виявлення фішингових листів за допомогою алгоритмів машинного навчання

Машинне навчання є досить потужним інструментом, коли мова йде про прогнозування певного стану. Воно надає спрощені та ефективні методи для аналізу даних, що вказує на перспективні результати щодо вирішення проблем класифікації в режимі реального часу. Однією з ключових переваг машинного навчання є можливість створювати гнучкі моделі для конкретних завдань, таких як виявлення фішингу. Оскільки фішинг є проблемою класифікації, машинне навчання добре підходить для її вирішення. Моделі машинного навчання можуть адаптуватися, щоб виявити закономірності шахрайських операцій, які допомагають розробити систему ідентифікації на основі навчання [16].

Зазвичай, існують два основні методи машинного навчання: навчання з учителем і навчання без учителя. Більшість машинного навчання, приблизно сімдесят відсотків, насправді є навчанням з учителем. Ще десять-двадцять відсотків – це навчання без учителя. Існують і інші методи, такі як напівконтрольоване навчання і навчання з підкріпленням, але вони використовуються не так часто [15].

Виявлення фішингових електронних листів останнім часом привертає значну увагу через їхній вплив на безпеку користувачів. Тому було розроблено багато методів виявлення фішингових електронних листів, починаючи від методів, орієнтованих на зв'язок, таких як протоколи автентифікації, чорні та білі списки, і закінчуючи методами фільтрації на основі вмісту [12].

Методи чорних і білих списків не довели своєї достатньої ефективності при використанні в різних доменах, і тому вони не є широко поширеними. Тим часом фільтри фішингу на основі вмісту широко використовуються і виявилися дуже ефективними. У зв'язку з цим дослідження були зосереджені на механізмах на основі вмісту та на розробці методів машинного навчання та інтелектуального аналізу даних на основі заголовків і змісту електронних листів [17].

Фахівці розробили широкий спектр фільтрів для прогнозування та запобігання фішинговим електронним листам, а також для боротьби з актуальними загрозами, використовуючи як традиційні методи, такі як захист автентифікації, так і сучасні методи машинного навчання або інтелектуального аналізу даних.

Традиційні методи виявлення поділяються на дві категорії: захист на рівні мережі та захист автентифікації.

Перша категорія захисту на рівні мережі включає фільтри чорних і білих списків, які запобігають фішингу, блокуючи підозрілі IP-адреси або домени від доступу до мережі. Крім того, існують фільтри зіставлення шаблонів і фільтри на основі правил, які покладаються на вручну введені та оновлені фіксовані правила для виявлення [17]. Розглянемо їх детальніше:

– фільтр чорного списку – техніка фільтрації чорного списку, яка забезпечує захист на рівні мережі, класифікуючи отримані електронні листи на основі адреси відправника, IP-адреси або DNS-адреси. Ці дані витягуються із заголовка електронного листа та порівнюються із заздалегідь визначеним списком. Якщо будь-які з цих даних співпадають зі списком, електронний лист буде відхилено. Таким чином, цей метод фільтрує фішингові електронні листи

для забезпечення безпеки на рівні мережі. Інтернет-провайдери (ISP) є відповідальною організацією за надання та реалізацію цього фільтра [17];

– фільтр білого списку – також забезпечує захист на рівні мережі, але на відміну від чорних списків, цей метод порівнює дані електронного листа із заздалегідь визначеним списком, що містить 10 статичних IP-адрес легальних доменів та IP-адрес. У цьому зв'язку, тільки електронні листи з даними, що співпадають зі списком, матимуть доступ до мережі до папки "Вхідні" користувача. Адреси електронної пошти та IP-адреси включаються до білого списку, якщо вони належать законним користувачам або компаніям, які погодилися додати свої адреси до цього списку. Електронні листи з даними, що співпадають із цим списком, будуть класифіковані як легальні лише на основі цього фільтра, тоді як інші електронні листи вважаються фішинговими і їм блокується доступ до мережі, за що цей фільтр також називають класифікатором легальних електронних листів [17];

– фільтр зіставлення шаблонів – техніка зіставлення шаблонів, яка фільтрує електронні листи на основі визначених шаблонів, включаючи слова, текстові рядки та набори символів, згадані у вмісті електронного листа, темі або відправнику. Фільтр шукає в електронному листі ці вказані шаблони, щоб класифікувати електронний лист як фішинговий або легальний. Хоча цей метод забезпечує захист на рівні мережі, він все ж таки дає певні неточні та хибні результати через величезну кількість отриманих електронних листів, які можуть містити заборонені слова або текстові рядки, але не повинні бути заблоковані [17].

Другий напрямок традиційних методів – захист автентифікації. Він пропонує безпеку як на рівні користувача, так і на рівні домену:

– перевірка електронної пошти - є методом автентифікації на рівні користувача, який потребує підтвердження відправника та одержувача. Після того, як відправник прийме повідомлення із запитом на підтвердження, електронний лист буде сертифіковано та класифіковано як легальний для переходу до папки "Вхідні" одержувача. В іншому випадку електронний лист

вважатиметься фішинговим і, таким чином, йому буде заблоковано доступ до папки "Вхідні";

– парольний фільтр – метод, який використовує автентифікацію на рівні користувача для забезпечення захисту. Цей фільтр дозволяє отримувати електронні листи лише за умови, що фільтр зміг виявити визначений пароль у рядку теми, адресі електронної пошти, заголовку або будь-якій іншій частині повідомлення. Якщо фільтр не знаходить пароль або виявляє неправильний пароль, електронний лист відхиляється. Оскільки ці паролі не встановлені за замовчуванням, користувачам цього фільтра вперше потрібно буде домовитися та встановити пароль один з одним, щоб потім їхні листи класифікувалися фільтром як легальні. Цей тип фільтрів має недолік, оскільки деякі легальні електронні листи можуть бути втрачені, якщо пароль не розпізнано, до того ж цей процес потребує часу.

Автоматизовані методи - це методи, що застосовують автоматизовані класифікатори, які базуються на машинному навчанні та інтелектуальному аналізу даних. Ці класифікатори працюють поряд із сервером і фільтрують отримані електронні листи на фішингові або легальні, аналізуючи різні ознаки заголовка та вмісту електронного листа [24].

До автоматизованих методів відносяться:

1. Support Vector Machines (SVM) – цей алгоритм є універсальним рішенням, яке може впоратися з багатьма завданнями. SVM може використовуватися для регресії, класифікації, а також для виявлення викидів. За допомогою цієї техніки дані поділяються на дві категорії за допомогою статистики, квадратних рівнянь та фіксованих правил. Двійкова класифікація даних створюється за допомогою розділяючої гіперплощини, щоб максимізувати простір границі на основі функцій ядра та витягування даних та зберігання їх у векторі, щоб знайти найкраще рішення проблеми та знайти відповідну класифікацію. Ця техніка корисна для знаходження рішень проблем з незнайомою історією, але не може аналізувати великі дані. На рисунку 1.2 опорні вектори зображені на межах, а

розділяюча гіперплощина розташована посередині поля, щоб максимізувати поле розділення [18].

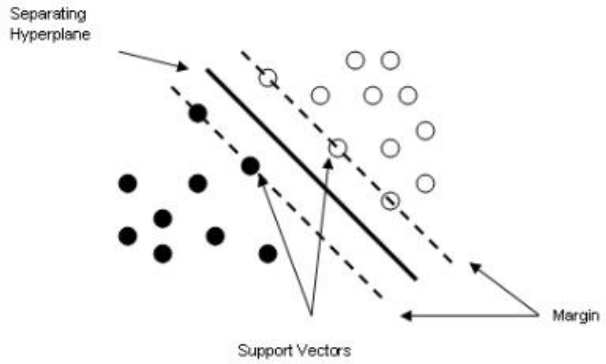



Рисунок 1.2 – Метод Машина опорних векторів [23]

SVM є популярним методом, оскільки він є потужним, але при цьому не вимагає значних обчислювальних ресурсів. Звичайно, чим більше ми прагнемо до точності, тим більше часу потрібно на обробку даних, оскільки це вимагає більше ресурсів від нашої комп'ютерної системи. Однак SVM вимагає значно менше ресурсів, не втрачаючи при цьому точності прогнозу [18].

Крім того, SVM може бути використаний для видалення більшої частини шуму в наборі даних, що зменшує кількість кроків, які потрібно виконати на етапі попередньої обробки.

SVM є важливим завдяки своїй потужності та універсальності. Цей алгоритм використовується в різних областях, наприклад, для програмного забезпечення розпізнавання осіб, класифікації тексту, розпізнавання почерку, а також в медичній промисловості для класифікації генів.

SVM має багато інших застосувань, оскільки він ефективно розділяє точки даних в наборі даних і дозволяє отримувати достатньо точні результати, щоб його можна було використовувати в складних додатках [18].

Переваги SVM [19]:

– добре працює з чітким поділом;

– ефективний у просторах великих розмірів;

– ефективний, коли кількість вимірювань більша, ніж кількість зразків;

– використовує підмножину навчальних точок в функції прийняття рішення (званих векторами підтримки), тому він також ефективний з точки зору пам’яті.

Недоліки SVM [19]:

– не такий ефективний, коли є великий набір даних, оскільки потрібно більше часу на навчання;

– не дуже добре працює, коли в наборі даних багато шуму, тобто цільові класи перекриваються;

– SVM не надає безпосередніх оцінок ймовірностей, вони розраховуються з використанням дорогої п’ятикратної перехресної перевірки. Він включений в пов’язаний метод SVC бібліотеки Python scikit-learn.

2. Дерева рішень – як і машини опорних векторів, дерева рішень є надзвичайно універсальними, оскільки вони можуть виконувати як операції регресії, так і класифікації. Це робить їх надзвичайно корисними при роботі з великими наборами даних (рис. 1.3).

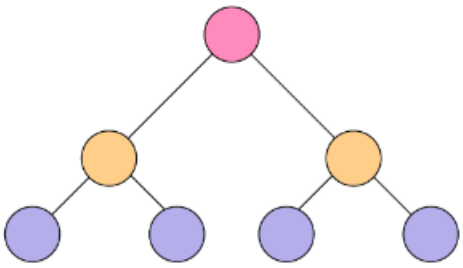


Рисунок 1.3 – Метод Дерева рішень [20]

Дерева рішень формують основу іншого типу алгоритму машинного навчання, відомого як метод ансамблю, зокрема алгоритму, відомого як випадковий ліс. Якщо у нас багато дерев рішень, вони утворюють ліс. Дерева рішень використовуються для моделювання рішень, включаючи визначення будь-яких наслідків і системних ресурсів, необхідних для всього процесу.

Всі ці елементи візуально представлені у вигляді графіка. Іншими словами, в результаті у нас буде блок-схема, яка містить всі виконувані нами тести. Наприклад, ми підкидаємо монету певну кількість разів. Щоразу результати будуть записуватися, і у нас буде кілька гілок дерева, які їх представляють, разом з листям, які представляють мітки класів [20].

Цей алгоритм машинного навчання з учителем вважається одним з найпотужніших, оскільки він пропонує точні результати, не жертвуючи занадто великою ефективністю і не вимагаючи занадто багато ресурсів нашої системи. Крім того, система графіків дозволяє нам легко інтерпретувати дані і, отже, вносити коригування у міру необхідності, замість того, щоб проходити ще один виснажливий процес, який призведе до того ж результату, але з набагато більшим обсягом роботи [20].

Переваги, що надаються деревами рішень [20]:

– зручність для користувача: на відміну від багатьох інших алгоритмів або методів, дерева рішень логічні навіть для тих, у кого немає значного машинного навчання або математичних знань і досвіду. Вам не потрібно бути фахівцем з обробки даних, щоб почати з ними працювати. Вони слідують простій, схожій на програмування структурі, подібної умовним операціями Python. Дерево рішень слідує логіці «якщо це, то зроби те, інакше зроби те». Крім того, результати зрозумілі і не вимагають навіть фундаментальних знань статистики для їх інтерпретації. У цьому перевага вбудованого графіка, який чітко пояснює дані, які ми аналізуємо;

– чисті дані: незалежно від того, який алгоритм ми використовуємо, нам завжди потрібно усувати якомога більше шуму і якомога більше викидів, використовуючи певні методи і прийоми. Однак перевага використання дерев рішень полягає в тому, що на них ці фактори практично не впливають. Викиди і шум будуть лише незначними незручностями, і ми часто зможемо повністю їх ігнорувати і все ж домогтися точного результату;

– універсальність: певні алгоритми машинного навчання призначені для роботи тільки з певними типами даних. Деякі з них можуть застосовуватися тільки до категоріальним даними, в той час як інші спеціально використовуються для числових даних. Часто з цієї причини їх необхідно комбінувати при роботі зі складними наборами даних, що містять обидва типи інформації. Однак дерева рішень можуть обробляти процес навчання для обох типів даних, що значно знижує робоче навантаження [19];

– вивчення даних: вивчення даних – цінний етап, який може зайняти багато часу, тому що вам потрібно визначити найбільш цінні точки даних, а потім встановити з'єднання між ними. Однак дерева рішень краще підходять для дослідницького аналізу, витрачається набагато менше часу на виконання цього кроку. Крім того, є можливість набагато ефективніше створювати нові функції, щоб ще більше підвищити точність прогнозів. Вивчення даних може бути стомлюючим, тому що в реальному світі часто доводиться мати справу з тисячами змінних, а пошук найбільш важливих з них може бути утруднений без відповідного інструменту.

3. K-Nearest Neighbors (KNN): Цей алгоритм є одним з найпростіших методів машинного навчання, який може використовуватися для задач регресії та класифікації. Він є непараметричним, що означає, що він не робить припущень про базовий розподіл даних. KNN використовує подібність між функціями для прогнозування значень нових точок даних. Це означає, що новій точці даних буде присвоєно значення, яке відображає її подібність до точок у навчальному наборі. Подібність між записами можна виміряти різними способами (рис. 1.4).

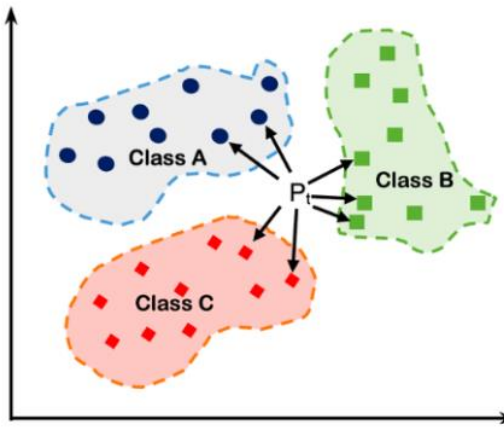



Рисунок 1.4 – Метод найближчих сусідів [21]

Після виявлення сусідів можна зробити підсумований прогноз, повернувши найпоширеніший результат або взявши середнє значення. KNN може бути використаний для проблем класифікації або регресії.

У випадку класифікації та регресії можна побачити, що вибір правильного K для наших даних здійснюється шляхом випробування декількох K і вибору того, який найкраще працює [21].

Головний недолік KNN полягає в тому, що збільшення обсягу даних сповільнює процес і робить його непрактичним вибором в середовищах, де прогнозування потрібно робити швидко.

Однак, якщо є достатньо обчислювальних ресурсів для швидкої обробки даних, які використовуються для прогнозування, KNN все ще може бути корисним для вирішення проблем, які мають рішення, які залежать від ідентифікації подібних об’єктів [21].

4. Random Forest – використовує множину дерев рішень для вирішення задач класифікації. Кожне дерево пропонує свій варіант класифікації, а клас який найчастіше зустрічається стає кінцевим рішенням. Цей метод ефективний, оскільки кожне дерево компенсує помилки інших, що зменшує ризик перенавчання. Випадковий ліс використовує підвибірки даних, що дозволяє зменшити складність моделі та покращити загальну точність [22]. Цей метод також ефективно обробляє велику кількість змінних та може оцінювати важливість кожної з них (рис.1.5).

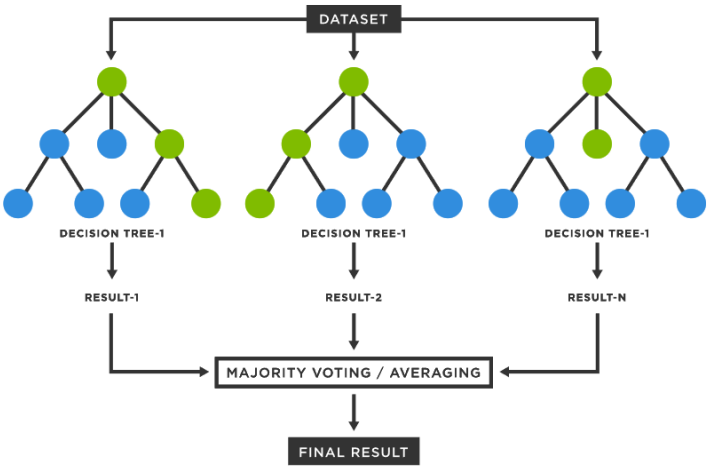



Рисунок 1.5 – Метод Випадковий Ліс [22]

Однак, через випадковість у процесі створення дерев, модель може бути непередбачуваною, і результати важко інтерпретувати. Крім того, велика кількість глибоких дерев може вимагати значних обчислювальних ресурсів, а обмеження глибини може негативно вплинути на точність моделі. Також, час навчання моделі збільшується пропорційно до кількості дерев.

Naive Bayes – це простий алгоритм машинного навчання, який вважає, що всі ознаки в наборі даних є незалежними. Він використовує це припущення для обчислення ймовірності того, що об’єкт належить до певного класу, враховуючи ймовірності попередніх подій (рис. 1.6).

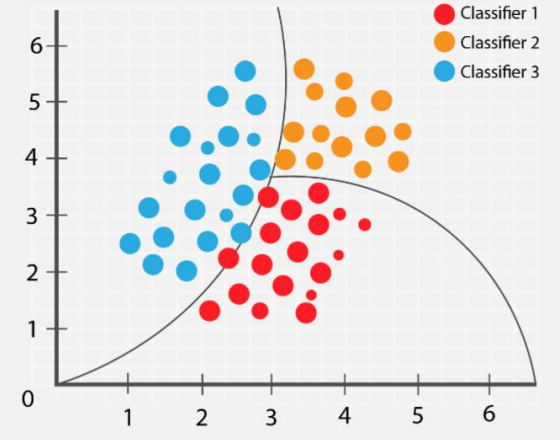



Рисунок 1.6 – Метод Наївний Баєс [23]

Переваги наївного Баєса:

- швидкість і простота прогнозування класу тестового набору даних;
- ефективність у багатокласовому прогнозуванні;
- краще працює з категоріальними ознаками порівняно з числовими;
- висока продуктивність порівняно з іншими простими методами;
- потребує менше тренувальних даних.

Але також має свої недоліки:

- припущення про незалежність ознак може бути обмеженням, оскільки в реальних завданнях цілком незалежні ознаки зустрічаються рідко;
- якщо в тестовому наборі даних є категорія, якої не було в навчальному наборі даних, модель призначить цій категорії нульову ймовірність і не зможе зробити прогноз.

Логістична регресія – це статистичний метод, що використовується для моделювання ймовірності певної події або результату на основі одного чи більше предикторів (незалежних змінних). На відміну від лінійної регресії, яка передбачає безперервні значення, логістична регресія використовується для бінарних або багатокласових змінних. Основна ідея полягає у використанні

логістичної функції (сигмоїдальної функції) для перетворення виходу лінійного рівняння у діапазон значень між 0 та 1, що інтерпретується як ймовірність.

Переваги**:**

– використовує обмежені обчислювальні ресурси;

– легко реалізувати, інтерпретувати та навчати;

– надає міру важливості предиктора (розмір коефіцієнта), а також його значення (позитивне чи негативне);

– має високу точність для простих наборів даних;

– коефіцієнти моделі можна інтерпретувати як індикатори важливості.

Недоліки:

– не рекомендується застосовувати, якщо кількість ознак більша, ніж кількість спостережень, оскільки це може призвести до перенавчання;

– припущення про лінійність між залежною та незалежними змінними - основне обмеження логістичної регресії;

– застосовується лише для прогнозування дискретних функцій. Тобто залежна змінна прив’язана до дискретного набору чисел;

– завдяки логістичній регресії неможливо вирішити нелінійні проблеми.

У таблиці 1.1 продемонстровано переваги та недоліки традиційних і автоматизованих методів виявлення фішингу.

Таблиця 1.1 – Переваги та недоліки традиційних і автоматизованих методів виявлення фішингу

| **Метод виявлення фішингу** | **Переваги** | **Недоліки** |
|---|---|---|
| Традиційні методи | Прості в реалізації | Неточні та неефективні проти складних фішингових атак |
| Автоматизовані методи | Більш точні та ефективні | Потребують великих обсягів тренувальних даних, легко обманюються новими атаками |

Хоча автоматизовані методи виявлення фішингу є більш ефективними, ніж традиційні методи, вони все ж таки мають деякі недоліки. По-перше, ці методи потребують великої кількості тренувальних даних для досягнення високої точності. По-друге, фішери постійно змінюють свої методи, щоб обійти фільтри, що змушує розробників методів виявлення постійно адаптувати свої моделі.

### 1.4 Аналіз існуючих програмних реалізацій

На сьогоднішній день існує велика кількість програмних рішень, як платних, так і безкоштовних, що надають захист від фішингу електронної пошти. Однак, проблемою багатьох існуючих систем моніторингу є не тільки недостатня якість виявлення фішингу, але й блокування легітимних повідомлень.

Однією з таких систем є Barracuda Email Filter. Barracuda Web Filter - це веб-шлюз, який забезпечує захист від веб-загроз, включаючи фішинг, шкідливе програмне забезпечення та небажаний контент [22].

Він використовує різноманітні методи, включаючи аналіз URL-адрес, аналіз HTML-коду та аналіз поведінки користувачів, для виявлення та блокування фішингових листів (рис. 1.7) [23].

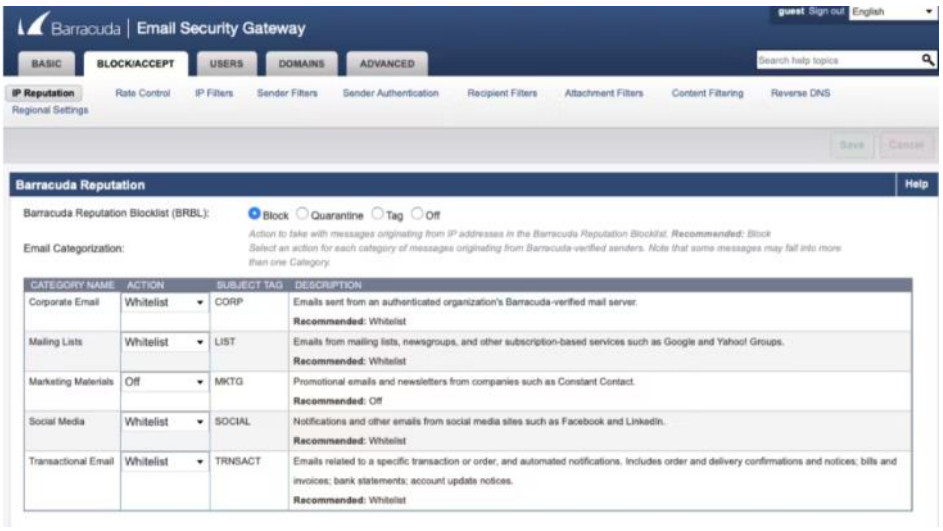



Рисунок 1.7 – Інтерфейс Barracuda Email Filter

Переваги:

- надає комплексний захист від веб-загроз;
- дуже ефективний у виявленні та блокуванні фішингових листів;
- має вбудовані функції для аналізу поведінки користувачів;

– пропонує підтримку 24/7.

Недоліки:

– комерційний продукт, який може бути дорогим для деяких користувачів;
– може бути складним у налаштуванні та обслуговуванні.

Кількісні характеристики ефективності:

– рівень блокування фішингу: 81%;
– рівень хибних спрацьовувань: 0,3%.

Proofpoint Secure Email Gateway – це шлюз електронної пошти, який забезпечує захист від загроз електронної пошти, включаючи фішинг, шкідливе програмне забезпечення та небажаний контент (рис. 1.8).

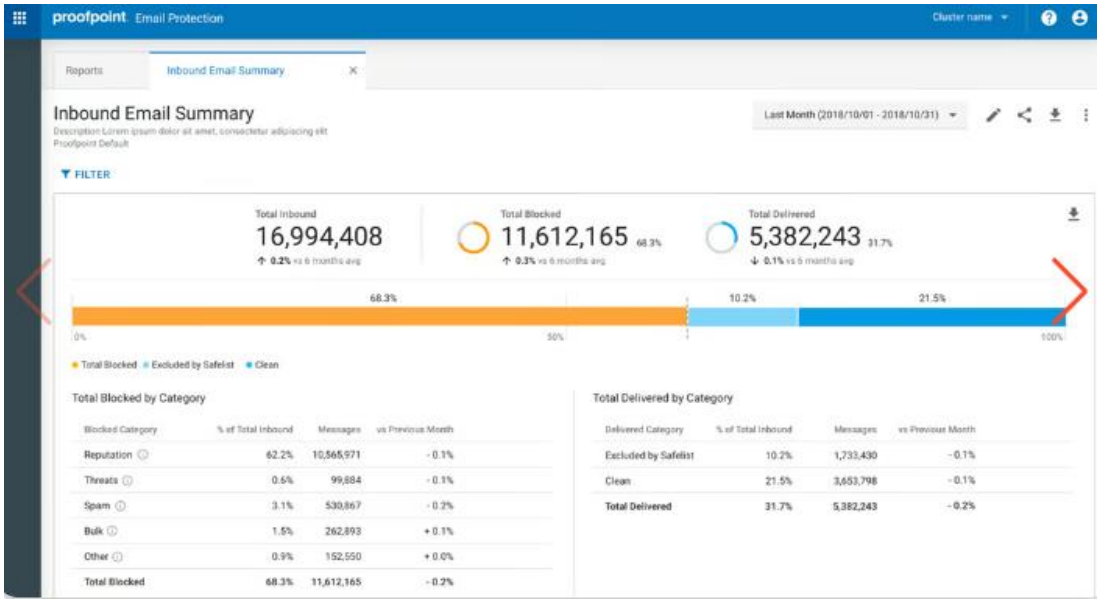



Рисунок 1.8 – Інтерфейс Proofpoint Secure Email Gateway

Він використовує різноманітні методи, включаючи аналіз URL-адрес, аналіз HTML-коду та аналіз поведінки користувачів, для виявлення та блокування фішингових листів [24].

Переваги:

– надає комплексний захист від загроз електронної пошти;
– дуже ефективний у виявленні та блокуванні фішингових листів;
– має вбудовані функції для аналізу поведінки користувачів;
– пропонує підтримку 24/7.

Недоліки:

– комерційний продукт, який може бути дорогим для деяких користувачів;
– може бути складним у налаштуванні та обслуговуванні.

Кількісні характеристики ефективності:

– рівень виявлення фішингу: 79%;
– рівень хибних спрацьовувань: 0,4%.

GFI MailEssentials – це комерційна програма для фільтрації спаму та фішингу, яка пропонує широкий спектр функцій, включаючи чорні списки URL-адрес, аналіз тексту, аналіз зображень, репутацію відправника та багато іншого (рис.1.9).

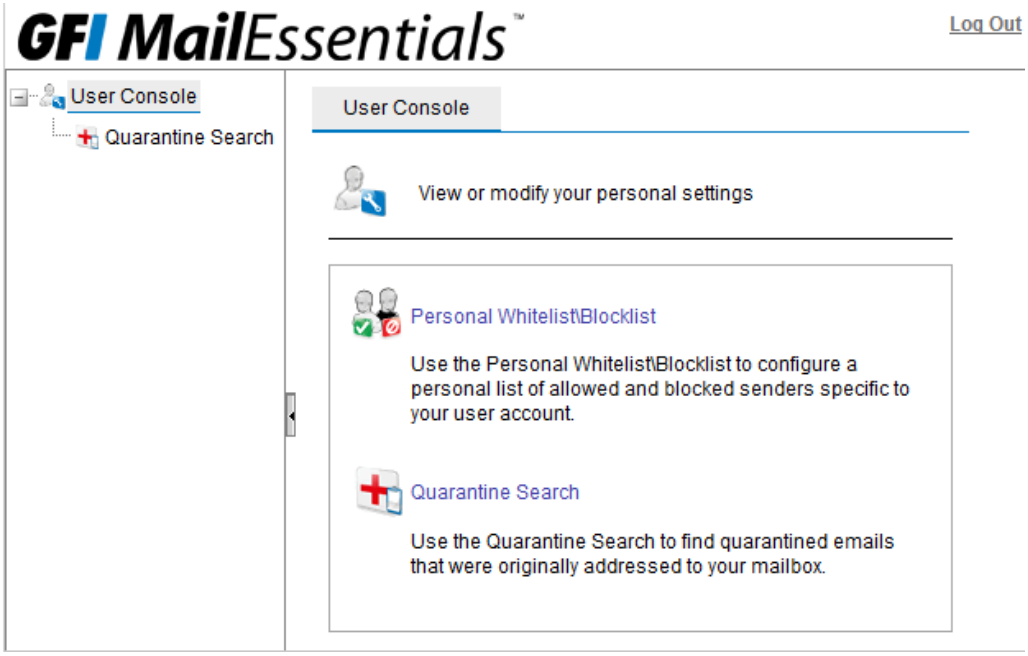



Рисунок 1.9 – Інтерфейс GFI MailEssentials

GFI MailEssentials проста в налаштуванні та має зручний інтерфейс користувача. Програма доступна за різними цінами, що робить її доступною для організацій різного розміру [25].

Переваги:

– проста в налаштуванні та має зручний інтерфейс користувача;
– пропонує широкий спектр функцій, окрім фільтрації спаму та фішингу, таких як антивірусний захист, шифрування електронної пошти та резервне копіювання;

– пропонує цілодобову підтримку від своїх розробників.

Недоліки:

– це платна програма, тому вона не підходить для організацій з обмеженим бюджетом;
– може бути складно розширити за допомогою власних правил та фільтрів.

Кількісні характеристики ефективності:

– рівень виявлення фішингу: 79-80%;
– кількість помилкових спрацьовувань: 0.3-0.5%.

Mimecast Email Security - це хмарна служба, яка забезпечує захист від загроз електронної пошти, включаючи фішинг, шкідливе програмне забезпечення та небажаний контент (рис. 1.10).

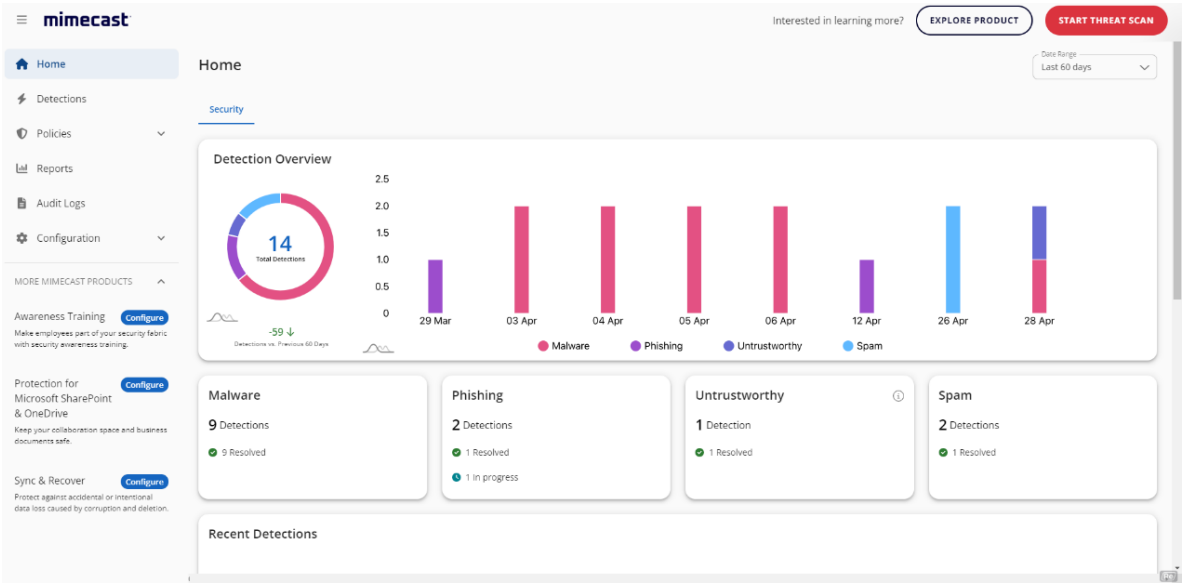



Рисунок 1.10 – Інтерфейс Mimecast Email Security

Вона використовує різноманітні методи, включаючи аналіз URL-адрес, аналіз HTML-коду та аналіз поведінки користувачів, для виявлення та блокування фішингових листів [26].

Переваги:

– надає комплексний захист від загроз електронної пошти;
– дуже ефективна у виявленні та блокуванні фішингових листів;
– має вбудовані функції для аналізу поведінки користувачів;

– пропонує підтримку 24/7;

– має хмарну архітектуру, яка забезпечує масштабованість та гнучкість.

Недоліки:

– комерційний продукт, який може бути дорогим для деяких користувачів;

– залежить від підключення до Інтернету.

Кількісні характеристики ефективності:

– рівень блокування фішингу: 89% ;

– рівень хибних спрацьовувань: 0,5% .

В таблиці 1.2 представлений порівняльний аналіз існуючих систем моніторингу електронної пошти на виявлення фішингових електронних листів.

Таблиця 1.2. Порівняльний аналіз існуючих аналогів системи моніторингу електронної пошти на виявлення фішингових листів

| **Система** | **Переваги** | **Недоліки** | **Кількісні характеристики** | **Ціна** |
|---|---|---|---|---|
| Mimecast Email Security | Безкоштовна та з відкритим кодом, проста в налаштуванні, широке співтовариство | Може мати нижчий рівень виявлення, потребує регулярного оновлення | Рівень виявлення фішингу: 89%, рівень хибних спрацьовувань: 0,5%, | Платна |
| GFI MailEssentials | Простота використання, широкий спектр функцій, підтримка | Платна, складність розширення | Рівень виявлення фішингу: 79-80%, кількість помилкових спрацьовувань: 0.3-0.5% | Платна |
| Barracuda Essentials | Широкий спектр функцій, простота використання, масштабованість | Платна, складність розгортання | Рівень виявлення фішингу: 81%, кількість помилкових спрацьовувань: 0.03% | Платна |

| | | | | |
|---|---|---|---|---|
| Proofpoint Email Protection | Широкий спектр функцій, ефективність виявлення, підтримка | Платна, складність розгортання | Рівень виявлення фішингу: 79%, кількість помилкових спрацьовувань: 0.04% | Платна |

Проаналізовані системи використовують різноманітні методи для виявлення та блокування фішингових атак, включаючи аналіз URL-адрес, аналіз HTML-коду, аналіз поведінки користувачів та інше. Однак, не всі з цих систем можуть надати надійний, швидкий та ефективний захист від фішингу. Більш того, багато з цих систем не забезпечують достатню гнучкість для користувачів, що обмежує їх здатність адаптуватися до нових видів фішингових атак.

Основними недоліками існуючих систем захисту від фішингу є:

- недостатня ефективність: багато систем не здатні ефективно виявляти та блокувати всі види фішингових атак;
- відсутність гнучкості: багато систем не дозволяють користувачам налаштовувати параметри фільтрації для адаптації до нових видів фішингових атак;
- висока вартість: деякі системи є дорогими, що робить їх недоступними для деяких користувачів;
- складність використання: деякі системи можуть бути складними у налаштуванні та обслуговуванні [26].

Враховуючи ці недоліки, було вирішено розробити нову, вдосконалену систему захисту електронної пошти від фішингу. Ця система буде надавати надійний, швидкий та ефективний захист від фішингу, а також буде забезпечувати високу гнучкість для користувачів. Крім того, вона буде доступною та простою у використанні, що зробить її ідеальним рішенням як для окремих користувачів, так і підприємств.

## 1.5 Висновки до розділу 1 та постановка задач

У даному розділі було проведено аналіз сучасного стану проблеми фішингу. Було досліджено існуючі методи виявлення фішингових атак, зокрема традиційні методи та методи, засновані на машинному навчанні.

Було встановлено, що, хоча багато сучасних методів можуть бути досить ефективними в виявленні фішингових атак, жоден з них не є універсальним рішенням. Кожен метод має свої переваги та недоліки, і ефективність кожного залежить від конкретного контексту.

Також було виявлено, що незважаючи на значний прогрес в області виявлення фішингу, фішингові атаки продовжують бути серйозною загрозою. Це свідчить про необхідність подальших досліджень та розробки нових, більш ефективних методів виявлення фішингу.

Таким чином, у процесі проведення загального огляду теоретичного матеріалу, були поставленні наступні задачі:

– розробити алгоритм роботи системи моніторингу електронної пошти;

– здійснити програмну реалізацію системи;

– дослідити коректність та ефективність роботи розробленої програми.

З виконанням усіх поставлених задач, буде досягнута головна мета даної дипломної роботи – розробка системи моніторингу електронної пошти на виявлення фішингових атак за допомогою методів машинного навчання.

# 2 ПРОЄКТУВАННЯ СИСТЕМИ МОНІТОРИНГУ ЕЛЕКТРОННОЇ ПОШТИ НА ВИЯВЛЕННЯ ФІШИНГОВИХ ЛИСТІВ

У сучасному світі, де інформація є одним з найцінніших ресурсів, захист та моніторинг електронної пошти стає все більш актуальним. Електронна пошта є одним з основних каналів комунікації в інтернеті, і вона часто стає мішенню для різноманітних атак, включаючи фішинг, спам та інші види шкідливих дій.

У цьому розділі буде спроєктовано систему моніторингу електронної пошти. Розглянуто різні аспекти цього процесу, включаючи визначення вимог до системи, вибір технологій та методів для її реалізації, а також планування етапів її розробки.

## 2.1 Розробка архітектури системи моніторингу електронної пошти

Архітектура системи моніторингу електронної пошти на виявлення фішингових атак включає в себе кілька модулів (рис. 2.1):

- модуль для експорту та інтеграції електронної пошти;
- модуль для виявлення ознак фішингу;
- модуль для аналізу та класифікації [27].

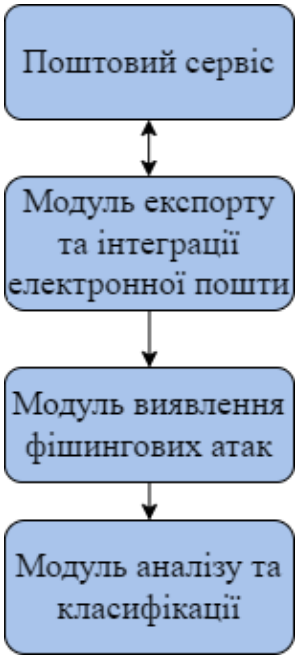



Рисунок 2.1 – Архітектура розробленої системи

Система ініціює свою роботу з модуля, відповідального за експорт та інтеграцію електронної пошти. Цей модуль є складовою програмного забезпечення, яке забезпечує відправлення електронних листів з обраних поштових сервісів, таких як Gmail, Outlook, Yahoo Mail, Пошта.ua та інші. Він також передає ці листи до програми для подальшого аналізу та обробки. Основна мета цього модуля полягає в забезпеченні ефективної інтеграції між програмою та обраними поштовими сервісами. Процес роботи цього модуля представлено на рисунку 2.2.


Рисунок 2.2 – Алгоритм роботи модуля експорту та інтеграції розробленої системи

Опис алгоритму роботи модуля експорту та інтеграції електронної пошти:

Крок 1. Початок інтеграції електронної пошти:

На цьому етапі необхідно перейти у налаштування Gmail акаунта, з якого буде проведено експорт та дозволити використання IMAP, для доступу до листів.

Крок 2. Створення Google App:

Створення власного додатоку Google. Для цього треба перейти у налаштування акаунту, налаштувати двох-факторну автентифікацію задля

дозволу на створення таких додатків. Далі треба створити новий додаток та отримати згенерований для нього пароль для подальших кроків.

Крок 3. Конфігурація бібліотеки:

Імпорт та налаштування бібліотеки. Під час цього кроку також необхідно ввести облікові дані створеного додатку та здійснити вхід за вказаними даними для подальшої роботи із електронною поштою.

Крок 4. Експорт електронних листів:

Експорт листів, які знаходяться у вкладці "Вхідні". Листи експортуються з усім оформленням, яке наявне, тому для належної роботи системи в подальшому, необхідно перейти до наступного кроку.

Крок 5. Оптимізація тексту:

Текст кожного з експортованих повідомлень форматується відповідним чином задля коректної роботи системи при класифікації повідомлень.

Крок 6. Кінець інтеграції електронної пошти:

Інтеграція з електронною поштою успішно створена, а експортовані листи готові до перевірки на виявлення фішингу за допомогою методів машинного навчання. В цілому, модуль експорту та інтеграції дозволяє розширити функціональність програми, надаючи безпосередній доступ до електронних листів та їх обробку в межах програми.

Модуль виявлення фішингових ознак використовується для автоматичного виявлення потенційних ознак фішингу в електронних листах за допомогою визначеного набору правил. Цей модуль базується на наборі критеріїв та правил, які використовуються для виявлення ознак фішингу. Ці критерії та правила розроблені на основі знань про типові методи, характеристики та властивості фішингових листів. Під час роботи модуль використовує ці правила для аналізу тексту, метаданих, посилань та інших елементів листа. Він перевіряє наявність специфічних ключових слів, фраз, форматування або структури, які можуть вказувати на наявність фішингу. [25].

Модуль може бути налаштований для використання різних рівнів чутливості або блокування фішингових листів з різним рівнем підозрілості. Він може

встановлювати мітки або позначати листи як підозрілі для подальшого вивчення адміністратором або користувачем. Правила можуть бути оновлені та доповнювані з часом, враховувати нові тренди та методи фішингу.

Модуль виявлення фішингових ознак не надає безпосереднього визначення, чи є лист фішинговим чи ні. Замість цього, він використовує правила та критерії для оцінки рівня підозрілості листа. Кожне правило або критерій може мати певну вагу або бали, які надаються, якщо відповідна ознака фішингу виявлена. Етапи роботи модуля наведено на рисунку 2.3.


Рисунок 2.3 – Алгоритм роботи модуля виявлення фішингових ознак розробленої системи

Опис алгоритму роботи модуля для виявлення фішингових ознак:

Крок 1. Електронний лист:

Отримання електронний листа, який був переданий з модуля експорту та інтеграції електронної пошти.

Крок 2. Аналіз адреси відправника:

Аналіз адреси відправника листа з метою виявлення можливих ознак фішингу, таких як імітація домену (наприклад, використання адреси, що схожа на легітимну), перевірка доменної відповідності (наприклад, порівняння домену відправника з відомими доменами законних організацій).

Крок 3. Аналіз посилань:

Розбиття кожного посилання, що міститься в листі, з метою виявлення ознак фішингу, таких як підозрілі домени, відсутність захищеного протоколу (наприклад, HTTPS), схожість з відомими фішинговими або шахрайськими сайтами та інше.

Крок 4. Аналіз тексту:

Застосування методів обробки тексту, такі як аналіз ключових слів, машинне навчання, правила та шаблони для виявлення зразків, що схожі на фішингові листи.

Крок 5. Наявність/відсутність фішингових ознак:

Створення бітового масиву, де кожен біт відображає наявність або відсутність конкретної ознаки фішингу.

Таким чином, модуль для виявлення фішингових ознак допомагає визначити ступінь підозрілості листа на основі встановленого набору правил та критеріїв, але остаточне рішення про те, чи є лист фішинговим, залежить від подальшого аналізу та рішення користувача або адміністратора.

Модуль аналізу та класифікації є ключовою частиною системи, яка обробляє дані, отримані від модуля для виявлення фішингових ознак [26].

Його основна задача – аналізувати ступінь підозрілості, отриманий для кожного листа, і класифікувати його на основі визначених критеріїв або правил.

У даному випадку, модуль використовує методи машинного навчання для класифікації листів. Він також може використовувати статистичний аналіз, правила або комбінацію цих підходів. Порогові бали - це задані значення, які

використовуються для встановлення межі між категоріями класифікації. Зазвичай вони використовуються для визначення, коли лист вважається фішинговим або безпечним на основі його рівня підозрілості.

Етапи роботи модуля представлено на рисунку 2.4.


Рисунок 2.4 – Алгоритм роботи модуля аналізу та класифікації розробленої системи

Опис алгоритму модуля аналізу та класифікації:

Крок 1. Наявність/відсутність фішингових ознак:

Модуль спочатку отримує бітовий масив, який відображає наявність або відсутність фішингових ознак. Цей масив є продуктом роботи попереднього модуля.

Крок 2. Ваги фішингових ознак:

Кожній фішинговій ознаці присвоюється вага або значимість. Це залежить від того, наскільки вона релевантна та впливає на визначення статусу листа.

Крок 3. Обчислення бальної оцінки листа:

Ваги фішингових ознак застосовуються до бітового масиву для обчислення бальної оцінки листа.

Крок 4. Аналіз отриманих результатів:

Бальна оцінка листа порівнюється з певним пороговим значенням, яке визначає, чи буде лист класифікований як фішинговий або безпечний.

Крок 5. Статус листа:

Встановлюється статус листа, який може бути позначений як фішинговий або безпечний.

Модуль аналізу і класифікації відіграє ключову роль у забезпеченні безпеки електронної пошти, допомагаючи автоматично визначати і класифікувати листи на основі їх підозрілості. Це дозволяє користувачам ефективно керувати своєю поштою та зменшує ризик отримання фішингових повідомлень [27].

## 2.2 Розробка модуля виявлення фішингових ознак

Модуль виявлення ознак фішингу може бути представлений у вигляді дерева. На рисунку 2.5 нижче наведено приклад модуля виявлення ознак фішингу, представленого у вигляді дерева.

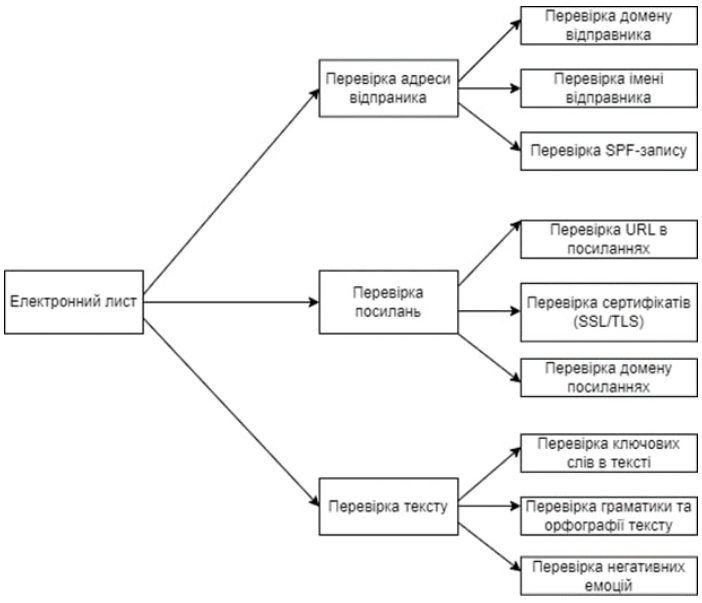



Рисунок 2.5 – Дерево модуля виявлення фішингових ознак розробленої системи

У даному прикладі “Виявлення фішингового листа” є головним вузлом, від якого відходять гілки до різних аспектів аналізу листа. Аспекти аналізу включають “Перевірку адреси відправника”, “Перевірку посилань”, “Перевірку тексту” та “Перевірку вкладень”. Кожен з цих аспектів може мати свої підгілки, які представляють окремі правила або перевірки, наприклад, “Перевірка домену відправника”, “Перевірка URL в посиланнях”, “Перевірка ключових слів в тексті” та “Перевірка типу файлу”. Це синтаксичне дерево надає зрозуміле уявлення про послідовність та ієрархію аспектів аналізу, які використовуються для виявлення фішингового листа. Кожен вузол може представляти окрему перевірку або правило, що використовується для оцінки листа. Аналізуючи таке синтаксичне дерево, можна краще зрозуміти, які аспекти аналізу використовуються для класифікації листа як фішингового або нормального. Тепер розглянемо опис та приклади кожного вузла.

Перевірка адреси відправника виконується наступним чином [28]:

– Перевірка домену відправника: Цей вузол перевіряє домен електронної пошти відправника. Він перевіряє відповідність доменного імені, його реєстрацію, належність до доменів, які часто використовуються в фішингових атаках, а також можливість підробки домену.

– Перевірка імені відправника: Цей вузол перевіряє ім’я відправника, яке вказане в полі “Від” листа. Він аналізує відповідність написання, наявність незвичайних символів або помилок, що можуть вказувати на фішинговий характер листа.

– Перевірка SPF-запису: Цей вузол перевіряє наявність SPF-запису для домену відправника. SPF (Sender Policy Framework) - це механізм, який допомагає перевірити, чи має відправник дозвіл на надсилання повідомлень від імені вказаного домену. Ця перевірка допомагає виявити підроблені адреси відправників.

Перевірка посилань виконується наступним чином:

– Перевірка URL в посиланнях: Цей вузол аналізує URL-адреси, що містяться у листі, і перевіряє їх на відповідність очікуваним шаблонам або

ознакам фішингу, таким як підозрілі домени, некоректний синтаксис або підстановка символів.

– Перевірка сертифікатів (SSL/TLS): Цей вузол перевіряє наявність і відповідність сертифікатів безпеки (SSL/TLS) для посилань, які використовують шифрування HTTPS. Він перевіряє, чи є сертифікати дійсними, чи не мають помилок, чи належать довіреним організаціям. Це допомагає визначити, чи є з'єднання з вебсайтами безпечними і надійними.

– Перевірка домену в посиланнях: Цей вузол аналізує домени, які містяться в посиланнях, з метою перевірки їхньої надійності та безпеки. Він перевіряє відповідність доменного імені, його реєстрацію, належність до небезпечних доменів, а також наявність попереджень або скарг щодо цих доменів.

Перевірка тексту виконується наступним чином:

– Виявлення ключових слів: Ця процедура сканує вміст електронного листа на предмет ключових слів чи виразів, які можуть вказувати на фішинг. Це можуть бути слова, пов'язані з конфіденційною інформацією, запитами на фінансові операції або обманними схемами.

– Перевірка граматики та правопису: Цей етап перевіряє, чи є текст листа граматично правильним та чи не містить орфографічних помилок. Наявність помилок може свідчити про ненадійність або погану якість повідомлення.

– Оцінка емоційного забарвлення: Тут аналізується емоційний тон тексту, зокрема наявність негативних емоцій або агресивного стилю. Це може включати агресивні висловлювання, погрози, спроби викликати тривогу або незаконні вимоги [29].

Модель може бути створена на основі математичних чи символьних відносин, використовуючи вищезазначені правила. Кожному правилу призначається вага, а загальний рейтинг листа визначається за допомогою суми оцінок за всіма правилами. Листи класифікуються як фішингові або безпечні на основі їх загального рейтингу [30].

Правила для аналізу:

1. Якщо домен відправника не відповідає очікуваному або знаходиться в чорному списку, він отримує значення “1” або “True”.

2. Якщо ім’я відправника не співпадає з очікуваним, містить незвичайні символи або помилки, він отримує значення “1” або “True”.

3. Якщо SPF-запис домену відправника не включає поточний сервер відправлення, він отримує значення “1” або “True”.

4. Якщо URL має неправильний синтаксис або символьну підстановку, він отримує значення “1” або “True”.

5. Якщо сертифікати сайтів, на які ведуть посилання, недійсні або містять помилки, вони отримують значення “1” або “True”.

6. Якщо домени сайтів, на які ведуть посилання, не відповідають очікуваним або знаходяться в чорному списку, вони отримують значення “1” або “True”.

7. Якщо текст листа містить ключові слова, пов’язані з шахрайством, особистими даними або фінансовими ризиками, він отримує значення “1” або “True”.

8. Якщо текст листа містить граматичні помилки, орфографічні помилки або незрозумілі фрази, він отримує значення “1” або “True”.

9. Якщо текст листа містить загрозливі, залякуючі або негативні емоції, він отримує значення “1” або “True”.

Процес встановлення ваг для правил та класифікації листів може варіюватися залежно від системи та доступних даних. У розробленій системі ваги правил були визначені за допомогою емпіричного підходу, який включає збір даних, експерименти, спостереження та оцінку результатів. Початкові значення ваг можуть бути встановлені на основі досвіду та інтуїції, з подальшим їх коригуванням в процесі отримання нових даних та зворотного зв’язку [31].

Кожному з критеріїв у системі виявлення фішингу призначено певний ступінь значущості та індекс критичності, які коливаються в діапазоні від 0 до 1. Залежно від індексу, критичність поділяється на три рівні:

– Низька критичність: від 0 до 0.32;

– Середня критичність: від 0.33 до 0.65;

– Висока критичність: від 0.66 до 1.

Перелік критеріїв з вказівкою їхньої критичності:

1. Аналіз домену відправника: середня критичність - 0.5.

2. Оцінка назви відправника: низька критичність - 0.2.

3. Перевірка запису SPF: середня критичність - 0.3.

4. Аналіз URL-адрес у посиланнях: середня критичність - 0.5.

5. Перевірка сертифікатів SSL/ELS: середня критичність - 0.3.

6. Аналіз доменів у посиланнях: середня критичність - 0.4.

7. Виявлення ключових слів у тексті: низька критичність - 0.1.

8. Оцінка граматики та правопису: низька критичність - 0.1.

9. Аналіз емоційного забарвлення тексту: низька критичність - 0.1.

10. Перевірка типу файлів вкладень: висока критичність - 0.7.

11. Оцінка розміру файлів вкладень: низька критичність - 0.2.

Для визначення рівня підозрілості електронного листа використовується наступна формула:

$$\mathrm{A} = \mathrm{P}_1 \times W_1 + \mathrm{P}_2 \times W_2 + \mathrm{P}_3 \times W_3 + \cdots + \mathrm{P}_n \times W_n \qquad (2.1)$$

$$\mathrm{P}_n \rightarrow \{0; 1\},$$

$$W_n \rightarrow \{0, 0.1, \ldots, 1\},$$

$$\mathrm{A}_S = 0 \leq 0.24 \times \mathrm{A}_{max},$$

$$\mathrm{A}_F = \mathrm{A} \geq 0.64 \times \mathrm{A}_{max},$$

де $A$ – оцінка підозрілості листа, $\mathrm{P}_n$ – наявність/відсутність фішингової ознаки, $W_n$ – вага правила, $A_{max}$ – максимальна оцінка підозрілості, $A_S$ – оцінка для безпечного листа, $A_F$ – оцінка для фішингового листа, $A_{max}$ – максимальна оцінка підозрілості, n - кількість правил для аналізу листа.

При таких встановлених вагах $\mathrm{A}_{max} = 4.4$.

У формулі (2.1) оцінка підозрілості A обчислюється шляхом суми добутків наявності (або відсутності) фішингової ознаки P$n$ з вагою правила $W_n$. Кожна характеристика має індивідуальний коефіцієнт, який показує її значущість у

контексті загальної підозрілості листа. Ці коефіцієнти $W_n$ мають значення в межах від 0 до 1.

Після обчислення оцінки підозрілості A, використовується максимальна можлива оцінка підозрілості $A_{max}$ для класифікації листа на дві категорії: безпечний лист $A_S$ та фішинговий лист $A_F$.

Якщо оцінка підозрілості A менше або дорівнює 0.25 помноженому на максимальну оцінку підозрілості Amax, то лист є безпечним $A_S$. І якщо оцінка підозрілості A більше або дорівнює 0.65 помноженому на Amax, то лист вважається фішинговим $A_F$.

Таким чином, зазначена формула дозволяє встановити рівень підозрілості листа шляхом аналізу наявних фішингових ознак та врахування їх ваги. Після цього листи класифікуються відповідно до встановлених порогових значень, що допомагає ідентифікувати безпечні та фішингові повідомлення.

Крім того, у процесі оцінювання підозрілості листа використовуються певні виключення. Ці виключення сприяють більш точному та надійному процесу аналізу, дозволяють уникнути хибних результатів і врахувати специфічних сценарії. Застосування виключень дозволяє адаптувати процес аналізу до конкретних сценаріїв і враховувати специфічні вимоги безпеки. Такий підхід дозволяє системі краще адаптуватись до різних типів листів та забезпечує більш гнучкий підхід до визначення підозрілості [32].

Виключення № 1. Якщо у листі, який аналізується на підозрілість, відсутні посилання, то їх ваги під час розрахунку оцінки підозрілості не враховуються. Це підходить для ситуацій, коли листи не містять активних посилань, або вони не грають ключової ролі в процесі визначення підозрілості. Це виключення допомагає зменшити кількість неправильних результатів та покращити точність системи. Враховуючи, що відсутність посилань не обов'язково означає, що лист є безпечним чи підозрілим, ми зосереджуємося на інших ознаках та їх вагах, що також враховуються при обчисленні оцінки підозрілості. Якщо посилання відсутні у листі, тоді $A_{max} = 2.0$.

Виключення № 2 – ґрунтується на принципі “білого списку” відправників. Це правило дозволяє знизити рівень підозрілості електронного листа, якщо його відправник включений до переліку перевірених та надійних джерел. При отриманні листа система сканує “білий список” на предмет відповідності відправника. У разі збігу відправника з переліком, ризиковість листа може бути автоматично знижена, або лист може бути класифікований як безпечний.

Додавання до “білого списку” може базуватися на історії взаємодій з відправниками, перевірених джерелах кореспонденції, або на основі рекомендацій від довірених організацій. Наприклад, якщо користувач самостійно додасть певний домен чи адресу електронної пошти до “білого списку”, то листи з цих джерел будуть автоматично розглядатися як надійні.

Це виключення дозволяє покращити ефективність системи визначення підозрілості, дозволяючи листам від перевірених відправників обходити зайві перевірки. Таким чином, знижується ймовірність неправильного визначення листів від довірених джерел як шкідливих.

Виключення № 3 – засноване на індивідуальних налаштуваннях користувача, що дозволяє адаптувати процес оцінювання підозрілості до особистих переваг. Користувачі мають можливість визначити специфічні критерії, які будуть враховуватися як виключення під час аналізу листів. Наприклад, можна налаштувати систему таким чином, щоб повідомлення від певних адрес або з конкретними фразами у темі не розцінювалися як підозрілі, незалежно від наявності інших потенційних ознак шахрайства.

Таке виключення надає користувачам контроль над оцінкою підозрілості, гарантуючи, що листи, які відповідають їхнім критеріям, не будуть помилково класифіковані як небезпечні. Використання персональних налаштувань як основи для виключень дозволяє забезпечити більшу гнучкість у процесі оцінки, враховуючи індивідуальні потреби та вподобання кожного користувача. Перед тим як визначити рівень підозрілості, модулі виявлення фішингу та аналізу повинні врахувати ці виключення, які можуть включати перевірку наявності

відправника у списку довірених, аналіз посилань та вкладень, а також інші параметри, задані користувачем [33].

## 2.3 Алгоритм роботи розробленої системи

Першим з основних алгоритмів роботи системи є алгоритм навчання моделі. Він включає в себе велику кількість етапів, які описують процес навчання моделі на тестовому наборі даних для подальшого використання у основному алгоритмі роботи системи для виявлення фішингового вмісту у листах.

Алгоритм процесу навчання моделі на рисунку 2.6.

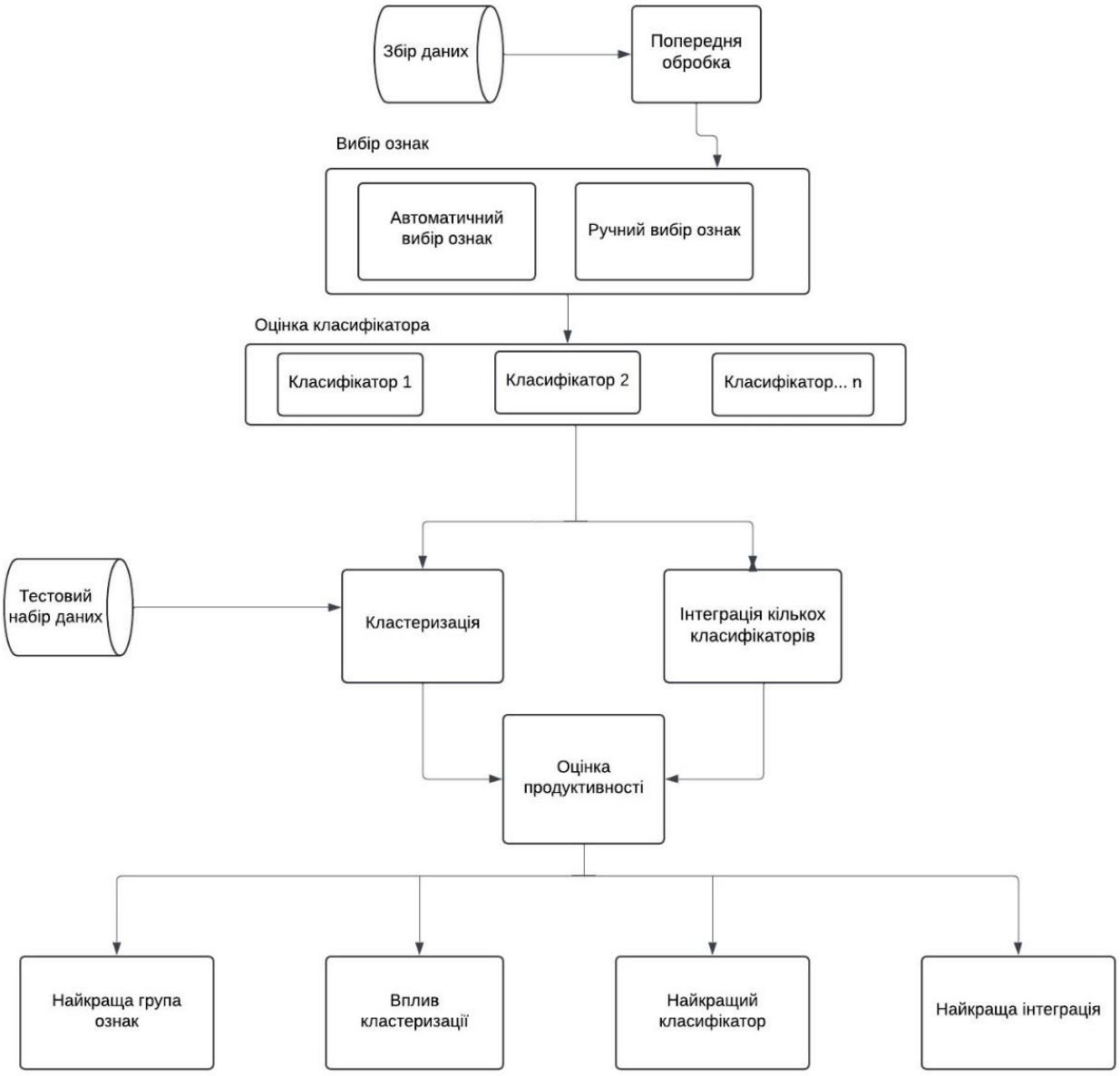



Рисунок 2.6 – Алгоритм навчання моделі розробленої системи

Запропонований підхід до навчання моделі дозволяє використовувати сильні сторони різних методів класифікації, що підвищує загальну точність виявлення шахрайських електронних листів. Він також дозволяє врахувати унікальні особливості різних кластерів, що є корисним для виявлення складних шаблонів в даних. Таким чином, даний алгоритм навчання моделі дозволяє розробити більш ефективну систему виявлення фішингових електронних листів.

Крок 1. Вибір ознак:

Цей етап включає визначення ключових характеристик, які допоможуть виявити шахрайські електронні листи.

Крок 2. Оцінка алгоритмів:

Використовуються три різних алгоритмів класифікації для навчання та тестування моделі на даних, а саме Наївний Байєс, Випадковий Ліс, Логістична регресія. Ці алгоритми вибрано через їх різні стратегії навчання та механізми тестування.

Крок 3. Кластеризація ознак:

Після вибору ознак та набору даних визначаються групи та розміщуються об’єкти у них. Цей процес називається кластеризацією. Він схожий на класифікацію, але відбувається без заздалегідь визначених категорій. Цей процес використовується для групування об’єктів зі схожими характеристиками.

Крок 4. Інтеграція кількох класифікаторів:

Використовується комбінований підхід для побудови мультикласифікатора. Використовуються три алгоритми: Логістична Регресія, Випадковий Ліс та Логістична регресія.

Логістична регресія - це статистичний метод, який використовується для прогнозування ймовірності виникнення певної події шляхом встановлення логістичної функції. У контексті виявлення фішингових листів, логістична регресія аналізує характеристики електронного листа, такі як URL-адреси, ключові слова та структуру тексту, щоб визначити, чи є він фішинговим.

Наївний Байєс – це інший популярний алгоритм машинного навчання, який використовується для класифікації даних на основі принципу незалежності

ознак. Він аналізує електронну пошту, використовуючи різні критерії, такі як наявність певних слів або форматування, щоб визначити, чи є вона фішинговою. Після того, як обидва алгоритми проаналізують електронну пошту, їх результати порівнюються. Якщо обидва алгоритми класифікують листа однаково (фішинговий або легітимний), то ця класифікація вважається остаточною.

Однак, якщо результати відрізняються, то на допомогу приходить третій алгоритм - Random Forest. Random Forest - це ансамблевий метод машинного навчання, який використовує багато дерев рішень для визначення кінцевого рішення. Він аналізує сумнівну електронну пошту та остаточно визначає, чи є вона фішинговою чи легітимною.

Основний алгоритм роботи розробленої системи виявлення фішингових електронних листів включає наступні етапи (рис. 2.7):

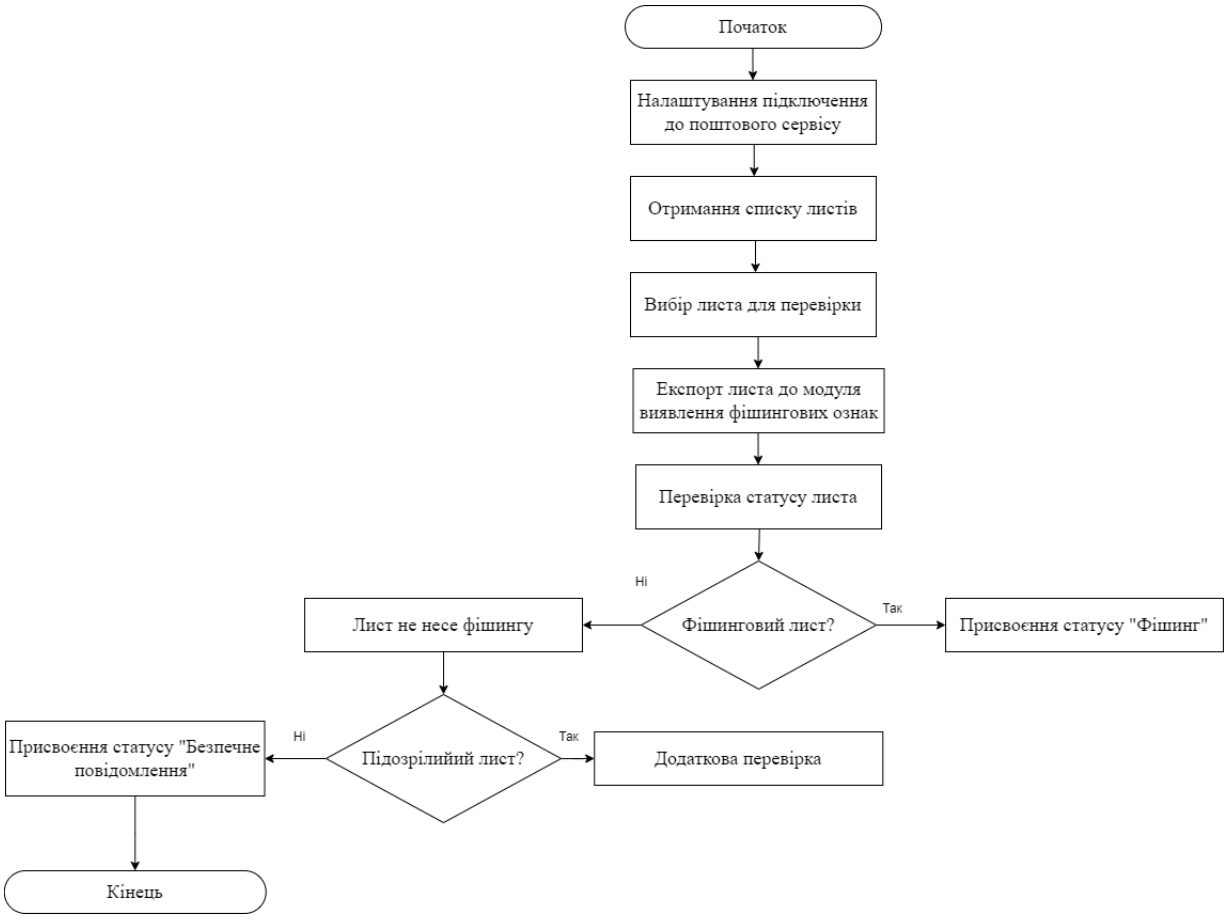



Рисунок 2.7 – Основний алгоритм роботи розробленої системи

Опис алгоритму роботи модуля експорту та інтеграції електронної пошти:

Крок 1. Початок:

Початок роботи із системою моніторингу електронної пошти на виявлення фішингу за допомогою методів машинного навчання.

Крок 2. Налаштування підключення до поштового сервісу:

На цьому кроці проходить процес інтеграції електронної пошти за допомогою створення та налаштування Google App для подальшого експорту електронних листів.

Крок 3. Отримання списку листів:

Після конфігурації підключення до поштового сервісу, відповідний модуль експортує електронні листи, які знаходились у вкладці “Вхідні”. Тексти цих листів будуть відформатовані відповідно до вимог тексту, який розуміє та аналізує навчена модель.

Крок 4. Вибір листа для перевірки:

Користувач обирає лист, який бажає перевірити на наявність фішингу та проводить взаємодію із кнопкою “Перевірити лист”. При взаємодії з цією кнопкою відбувається виклик відповідної кінцевої точки, яка виконує подальші операції.

Крок 5. Експорт листа до модуля виявлення фішингових ознак:

На цьому кроці викликана кінцева точка експортує обраний лист до модуля, який відповідальний за виявлення фішингових ознак.

Крок 6. Перевірка статусу листа:

Модуль виявлення фішингових ознак виконує перевірку листа за допомогою моделі, яка була навчена відповідними методами машинного навчання. На основі отриманих результатів система виконує відповідні дії.

Крок 7. Фішинговий лист?

Модуль виявлення фішингових ознак проводить перевірку листа на вірогідність фішингу.

Крок 7.1. Так

Система виводить користувачеві відповідний статус перевіреного листа, а саме “Фішинг”. Будь-яке слідування інструкціям, описаних в таких листах може призвести до впливу шахраїв.

Крок 7.2. Ні

Система визначає, що лист не несе фішингу і далі проводить перевірку на підозрілість.

Крок 8. Підозрілий лист?

На цьому кроці, відбуваються відповідні дії в залежності від результатів перевірки на підозрілість.

Крок 8.1 Так:

Система направляє обраний лист на повторну перевірку для більш точного прогнозу.

Крок 8.2 Ні:

Система на основі результатів отриманих у модулі виявлення фішингових ознак виводить користувачеві статус повідомлення, а саме “Безпечне повідомлення”.

Крок 9. Кінець:

Основний алгоритм програми виконав поставлену задачу.

Цей алгоритм забезпечує систематичну перевірку електронних листів на наявність фішингових ознак та визначення їх статусу для забезпечення безпеки користувачів.

**Висновок до розділу 2**

В даному розділі було розроблено детальну архітектуру системи для виявлення фішингових електронних листів. Основні елементи цієї архітектури включають модуль експорту та інтеграції електронної пошти, модуль виявлення фішингових ознак та модуль аналізу та класифікації.

Модуль експорту та інтеграції електронної пошти відповідає за збір та обробку вхідних даних, включаючи електронні листи. Модуль виявлення фішингових ознак та модуль аналізу та класифікації використовуються для

аналізу та класифікації цих даних, а також для виявлення потенційних фішингових листів.

Було здійснено удосконалення системи моніторингу електронної пошти на фішингові атаки завдяки комбінованому підходу для побудови мультикласифікатора. Цей підхід дозволяє використовувати сильні сторони різних алгоритмів класифікації та методів кластеризації, що підвищує загальну точність виявлення фішингових електронних листів.

Практичну реалізацію наведеного алгоритму заплановано реалізувати та описати у наступному розділі.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ МОНІТОРИНГУ ЕЛЕКТРОННОЇ ПОШТИ НА ВИЯВЛЕННЯ ФІШИНГОВИХ ЛИСТІВ

У даному розділі буде здійснено вибір середовища розробки програмного додатку та вибір мови програмування для його реалізації. Також буде здійснено розробку систему моніторингу електронної пошти на виявлення фішингових листів за допомогою методів машинного навчання та тестування правильності та ефективності її роботи, після чого розроблену систему буде протестовано.

## 3.1 Вибір мови програмування та середовища розробки

Для даної дипломної роботи середовищем розробки було обрано Visual Studio Code. Цей інструмент є одним з популярних текстових редакторів, який отримав велику увагу серед професіоналів та рекомендується новачкам у програмуванні. Як продукт відомої компанії Microsoft, VS Code є безкоштовним, з відкритим вихідним кодом і сумісним з операційними системами Windows, Linux і macOS.

Цей редактор можна легко встановити і він не займає багато місця на пристрої. Незважаючи на його компактність, ця платформа має багато потужних та сучасних функцій, тому його невеликий розмір не може бути недоліком. За допомогою цього інструменту можна працювати з різноманітними мовами програмування, такими як Java, Python та іншими. Також є можливість додавати необхідні розширення до редактора, наприклад, інструменти для налагодження коду, засоби для розробки в хмарі та веб-розробки [34].

Його інтуїтивно зрозумілий інтерфейс - це ще один плюс. Завдяки чіткій організації редактора на окремі секції, програмісти знаходять його зручним для написання зрозумілого коду та виправлення помилок.

Одна з ключових переваг цього редактора полягає у його універсальності та сучасних функціональних можливостях. Оскільки неможливо, щоб редактор коду надавав рішення для кожної можливої проблеми, він підтримує використання розширень.

Розширення - це функціональність, розроблена сторонніми розробниками, яку програмісти можуть додавати для покращення якості, функціональності та зовнішнього вигляду коду. Також є можливість додавати мови програмування, засоби для налагодження та інші інструменти за допомогою розширень цього редактора [35].

Так як редактор є проєктом з відкритим вихідним кодом, сторонні розширення з інноваційними можливостями постійно додаються до нього та регулярно оновлюються. Тому в Visual Studio Code розробники можуть знайти розширення практично для будь-яких потреб. Дане середовище розробки є незаперечним інструментом для програмістів, оскільки він може працювати з багатьма мовами програмування та має найсучасніші функції.

Вибір мови програмування є важливим кроком у процесі розробки програмного продукту. Кожна мова має свої особливості, сильні сторони та обмеження, які варто враховувати при прийнятті рішення [36].

Python було обрано як мова програмування через наступні переваги:

По-перше, Python є однією з найбільш використовуваних мов програмування в науковому та академічному середовищі. Його широке використання в областях, таких як наукові розрахунки, обробка даних та машинне навчання, підтверджує його потужність та ефективність в цих областях. Відкритий характер Python та наявність багатьох наукових бібліотек, таких як NumPy, Pandas та SciPy, дозволяють легко виконувати складні розрахунки, статистичний аналіз та моделювання.

По-друге, Python має простий та стислий синтаксис, що сприяє зручності розробки програмного коду. Це особливо важливо для дипломної роботи, де важливою є чіткість та читабельність коду. Крім того, наявність великого спектру стандартних бібліотек дозволяє виконувати багато завдань без необхідності написання великої кількості додаткового коду [37].

Третя перевага полягає в наявності численних інструментів для візуалізації даних у Python. Бібліотеки, такі як Matplotlib та Seaborn, надають розширені

можливості для створення графіків, діаграм та інших візуальних елементів, що допомагають ефективно представляти результати досліджень та аналізу даних. Крім того, Python має активну спільноту розробників, яка надає підтримку, документацію та розвиває нові інструменти. Це створює сприятливе середовище для обміну знаннями та співпраці з іншими вченими та дослідниками. [38] Таким чином, вибираючи Python для дипломної роботи, було отримано доступ до потужних наукових інструментів, використовуючи зручний синтаксис та велику кількість бібліотек, що сприяє ефективності та точності досліджень, дозволяє зручно візуалізувати результати та сприяє спільноті розробників для обміну досвідом та підтримкою.

В якості фреймворку машинного навчання у програмній реалізації системи моніторингу електронної пошти на виявлення фішингових атак за допомогою методів машинного навчання буде використано TensorFlow.

TensorFlow, розроблений командою Google Brain, є однією з найпопулярніших бібліотек з відкритим кодом для машинного навчання на Python. Його гнучкість, масштабованість та широкий спектр можливостей роблять його чудовим вибором для різноманітних завдань машинного навчання, від регресії та класифікації до глибокого навчання та обробки природної мови [38].

Переваги TensorFlow [38]:

– Гнучкість**.** Фреймворк пропонує низькорівневе програмне забезпечення, що дає користувачам повний контроль над створенням моделей машинного навчання. Це робить його ідеальним для дослідників та розробників, яким потрібна гнучкість для налаштування алгоритмів під свої конкретні потреби;

– масштабованість**.** TensorFlow може працювати на різних апаратних платформах, від ноутбуків до потужних кластерів GPU. Це робить його придатним для вирішення задач машинного навчання будь-якого масштабу;

– широкий спектр можливостей**.** Фреймворк підтримує широкий спектр моделей машинного навчання, включаючи нейронні мережі, лінійні моделі та

дерева рішень. Він також включає в себе широкий спектр інструментів для обробки даних, візуалізації та оцінки моделей;

– спільнота та ресурси**.** TensorFlow має велику та активну спільноту користувачів та розробників. Це означає, що доступно багато ресурсів для навчання та отримання допомоги, якщо виникнуть проблеми.

### 3.2 Програмна реалізація розробленої системи

Першим кроком для програмної реалізації системи моніторингу електронної пошти на предмет фішингу за допомогою методів машинного навчання є вибір набору даних (dataset). Від якості та релевантності даних залежить успіх моделі, оскільки вони визначають її здатність до навчання та точність прогнозів [39].

Ідеальним місцем для пошуку наборів даних є платформа Kaggle, де представлено безліч високоякісних та різноманітних датасетів. Використання Kaggle дозволяє знайти дані, які максимально відповідають потребам, що значно спрощує процес підготовки та підвищує ефективність розробки моделі машинного навчання [39].

Для запропонованої системи, необхідний набір даних з наступними колонками:

– ID – унікальний ідентифікатор кожного об’єкту;

– текст – власне текст повідомлення для аналізу;

– тип повідомлення – заздалегідь визначений тип (фішинг чи безпечне повідомлення), для того щоб навчити модель.

Після проведення пошуків на платформі, описаним потребам відповідає набір даних https://www.kaggle.com/datasets/subhajournal/phishingemails. Цей набір і використаємо для програмної реалізації системи, а саме для навчання моделі.

Першим кроком програмної реалізації буде навчання моделі з використанням трьох методів машинного навчання: метод логістичної регресії, Random Forest та Naive Bayers. Для кожного з методів буде продемонстровано точність визначення. В подальшому навчена та протестована модель буде

використовуватись для системи моніторингу електронної пошти на предмет фішингу [39].

Для початку завантажуємо набір даних та перевіряємо на наявність порожніх колонок:

```
df = pd.read_csv(r'D:/phishing/content/Phishing_Email.csv')
df.isna().sum()
```

Після перевірки наявно 16 пустих колонок у наборі даних, які необхідно видалити. Також видаляємо дублікати:

```
df.drop(['Unnamed: 0'],axis=1,inplace=True)
df.dropna(inplace=True,axis=0)
df.drop_duplicates(inplace=True)
```

Графічно демонструємо кількість відсортованих повідомлень (рис. 3.1):

```
index = df['Email Type'].value_counts().index
values = df['Email Type'].value_counts().values
plt.bar(index,values,color=['blue','red'])
plt.title("Кількість відсортованих повідомлень")
plt.xlabel("Категорія")
plt.ylabel("Кількість")
plt.show()
```

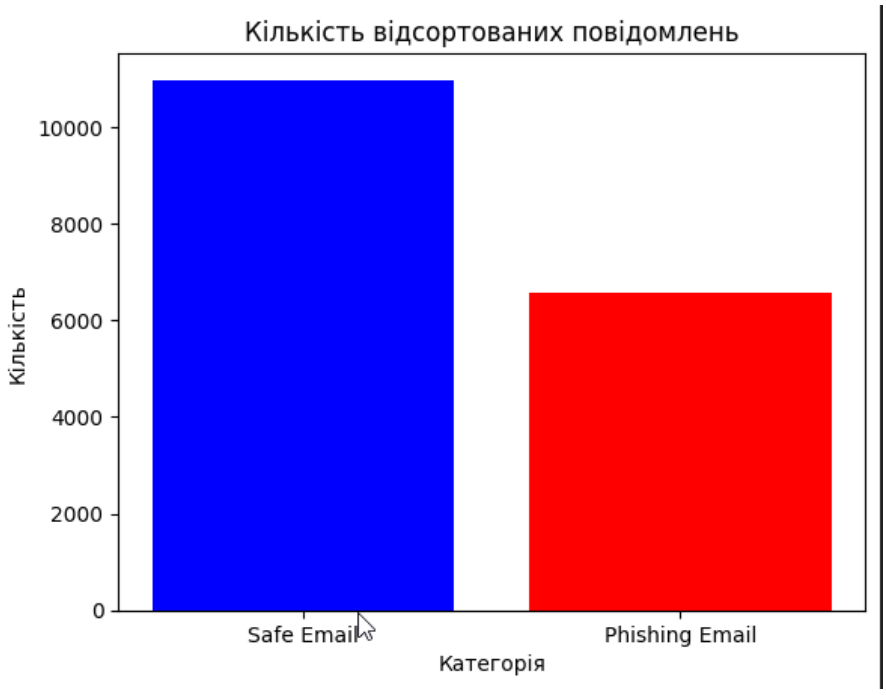



Рисунок 3.1 – Графічне відображення кількості відсортованих повідомлень

Для того щоб комп'ютер розумів категорії повідомлень, трансформуємо їх текстове представлення у числове:

```
lbl = LabelEncoder()
df['Email Type'] = lbl.fit_transform(df['Email Type']) ### 0 позначає фішингове повідомлення, 1 - безпечне
```

Для оброблення інформації нейронною мережею, конвертуємо текст у

нижній реєстр, видаляємо пунктуацію та лишні пробіли:

```
import re
def preprocess_text(text):
   text = re.sub(r'[^\w\s]', '', text)
   text = text.lower()
   text = re.sub(r'\s+', ' ', text).strip()
   return text
df["Email Text"] =df["Email Text"].apply(preprocess_text)
```

Конвертуємо текст у вектор:

```
tf = TfidfVectorizer(stop_words='english',max_features=10000)
feature_x = tf.fit_transform(df['Email Text']).toarray()
y_tf = np.array(df['Email Type'])
```

Розділяємо набір даних випадковим чином на навчальну та тестову частини:

```
X_tr,X_tst,y_tr,y_tst = train_test_split(feature_x,y_tf,test_size=0.2,random_state=0)
```

Використовуємо алгоритм Наївного Баєса та поміщаємо туди навчальну та тестову частини:

```
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()
nb.fit(X_tr,y_tr)
```

Далі визначаємо тип повідомлень на тестовому наборі даних і виводимо результати точності передбачень моделі за відповідним алгоритмом:

```
from sklearn.metrics import accuracy_score,f1_score,classification_report
pred_nav = nb.predict(X_tst)
print(f"точність передбачень використовуючи алгоритм Naive Bayes: {accuracy_score(y_tst,pred_nav)*100:.2f} %")
print(f"f1 результат використовуючи алгоритм Naive Bayes: {f1_score(y_tst,pred_nav)*100:.2f} %")
print("звіт:\n\n",classification_report(y_tst,pred_nav))
```

На рисунку 3.2 показано результати точності моделі за алгоритмом Naive Bayers.

```
точність передбачень використовуючи алгоритм Naive Bayes: 97.52 %
f1 результат використовуючи алгоритм Naive Bayes: 97.99 %
звіт:

              precision    recall  f1-score   support

           0       0.97      0.96      0.97      1351
           1       0.98      0.98      0.98      2157

    accuracy                           0.98      3508
   macro avg       0.97      0.97      0.97      3508
weighted avg       0.98      0.98      0.98      3508
```

Рисунок 3.2 – Результати точності моделі за алгоритмом Naive Bayers

Виходячи з рисунку 3.2, точність алгоритму складає 97,52%, що є високим показником і може свідчити про гарні результати його використання у майбутньому.

Далі переходимо до алгоритму логістичної регресії. Для цього обробляємо набір даних за його допомогою та передбачаємо тип повідомлень у тестовому наборі:

```
lg = LogisticRegression()
lg.fit(X_tr,y_tr)
pred_lg = lg.predict(X_tst)
print("")
print(f"точність передбачень використовуючи алгоритм Logistic Regression:{accuracy_score(y_tst,pred_lg)*100:.2f} %")
print(f"f1 результат використовуючи алгоритм Logistic Regression: {f1_score(y_tst,pred_lg)*100:.2f} %")
print("звіт: \n",classification_report(y_tst,pred_lg))
```

На рисунку 3.3 відображено результати точності моделі за алгоритмом Logistic Regression.

```
точність передбачень використовуючи алгоритм Logistic Regression:97.95 %
f1 результат використовуючи алгоритм Logistic Regression: 98.34 %
звіт:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97      1351
           1       0.98      0.99      0.98      2157

    accuracy                           0.98      3508
   macro avg       0.98      0.98      0.98      3508
weighted avg       0.98      0.98      0.98      3508
```

Рисунок 3.3 – Результати точності моделі за алгоритмом Logistic Regression

Останнім алгоритмом було обрано Випадковий Ліс, проводимо аналогічну операцію із ним та оцінюємо точність:

```
rnf = RandomForestClassifier()
rnf.fit(X_tr,y_tr)
pred_rnf = rnf.predict(X_tst)
print(f"точність передбачень використовуючи алгоритм Random Forest:{accuracy_score(y_tst,pred_rnf)*100:.2f} %")
print(f"f1 результат використовуючи алгоритм Random Forest: {f1_score(y_tst,pred_rnf)*100:.2f} %")
print("звіт: \n",classification_report(y_tst,pred_rnf))
```

На рисунку 3.4 показано результати точності моделі за алгоритмом Випадковий Ліс.

```
точність передбачень використовуючи алгоритм Random Forest:97.63 %
f1 результат використовуючи алгоритм Random Forest: 98.06 %
звіт:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      1351
           1       0.99      0.97      0.98      2157

    accuracy                           0.98      3508
   macro avg       0.97      0.98      0.98      3508
weighted avg       0.98      0.98      0.98      3508
```

Рисунок 3.4 – Результати точності моделі за алгоритмом Random Forest

Після навчання моделі та демонстрації результатів її точності згідно різних алгоритмів, переходимо до розробки самої системи моніторингу.

Створюємо клас, який за допомогою методу check_Phishing(email) буде приймати повідомлення та класифікувати повідомлення.

```
class MyTestClass:
  def check_Phishing(email):
```

Далі обробляємо текст щоб його розуміла нейронна мережа:

```
    email = re.sub(r'[^\w\s]', '', email)
    email = re.sub(r'\s+', ' ', email).strip()
    email = email.lower()
```

Трансформуємо повідомлення у вектор:

```
email_tfidf = tf.transform([email])
```

Визначаємо тип повідомлення за трьома алгоритмами та зберігаємо результати у змінних:

```
prediction_nb = nb.predict(email_tfidf)
prediction_rnf = rnf.predict(email_tfidf)
prediction_lg = lg.predict(email_tfidf)
```

Рахуємо загальну кількість передбачень “фішинг” та “безпечне повідомлення” згідно отриманих результатів, зберігаємо їх у змінні для кожного алгоритму:

```
nb_phishing_count = 1 if prediction_nb[0] == 0 else 0
rnf_phishing_count = 1 if prediction_rnf[0] == 0 else 0
lg_phishing_count = 1 if prediction_lg[0] == 0 else 0

    nb_safe_count = 1 if prediction_nb[0] == 1 else 0
    rnf_safe_count = 1 if prediction_rnf[0] == 1 else 0
    lg_safe_count = 1 if prediction_lg[0] == 1 else 0
```

Далі визначаємо який тип було передбачено більше разів:

```
phishing_count = nb_phishing_count + rnf_phishing_count + lg_phishing_count
    safe_count = nb_safe_count + rnf_safe_count + lg_safe_count
    if phishing_count > safe_count:
      return "Фішинг"
    else:
      return "Безпечне повідомлення"
```

У випадку якщо повідомлення вважається фішингом в більшості передбачень, то повертаємо стрічку “фішинг”, якщо більшість передбачень вважають повідомлення безпечним, то повертаємо стрічку “безпечне повідомлення”.

Завдяки обраному підходу використання моделі, яка надає результати згідно трьох різних високоточних алгоритмів, правильна класифікація повідомлення досягає найбільш можливої точності. Це дає нам змогу використовувати розроблену модель, клас та метод для впровадження у систему моніторингу електронної пошти на предмет фішингу.

Для тестування системи використаємо бібліотеку imaplib, яка дозволяє створити відповідний Google App, який дозволяє отримати повідомлення з вказаної електронної пошти.

Спочатку вказуємо облікові дані створеного Google App:

```
username = "valerymarchuk2103@gmail.com"
app_password = "xiwm vlve pmxw ysbg"
```

Визначаємо функцію, яка буде відповідальна за отримання повідомлень. Для початку створюємо порожній масив даних, в який будуть зберігатись повідомлення:

```
def extract_emails():
  emails = []
```

Далі під’єднуємось до серверу та виконуємо вхід у акаунт:

```
  mail = imaplib.IMAP4_SSL("imap.gmail.com")
  mail.login(username, app_password)
```

Вказуємо які самі повідомлення нам потрібні, у нашому випадку вхідні:

```
  mail.select("inbox")
```

Виконуємо пошук усіх повідомлень:

```
  status, messages = mail.search(None, "ALL")
```

Конвертуємо повідомлення у список з їх ID:

```
  mail_ids = messages[0].split()
```

Далі підтягуємо останні 300 повідомлень (кількість повідомлень можна змінювати в залежності від потреб):

```
for mail_id in mail_ids[-300:]:
    status, msg_data = mail.fetch(mail_id, "(RFC822)")
```

Далі розшифровуємо тему та відправника повідомлення:

```
for response_part in msg_data:
        if isinstance(response_part, tuple):
            msg = email.message_from_bytes(response_part[1])
            subject, encoding = decode_header(msg["Subject"])[0]
            if isinstance(subject, bytes):
            subject = subject.decode(encoding if encoding else "utf-8")
            from_ = msg.get("From")
```

Далі отримуємо вміст повідомлення та обробляємо його у вигляд необхідний для обробки:

```
content_type = msg.get_content_type()
body = msg.get_payload(decode=True).decode()
            if content_type == "text/plain":
                soup = BeautifulSoup(body, "html.parser")
                text_body = soup.get_text()
                text_body = re.sub(r'\s+', ' ', text_body).strip()
                text_body = re.sub(r"\b(\w+)'(\w+)\b", r"\1`\2", text_body)
                emails.append({"Subject": subject, "From": from_, "Body": text_body})
```

Закриваємо з’єднання та виходимо із облікового запису:

```
  mail.close()
  mail.logout()
```

Функція повертає повідомлення:

```
  return emails
```

Для візуального відображення повідомлень у системі створимо та викличемо функцію:

```
def save_emails_to_json(emails, filename):
  if os.path.exists(filename):
    base, ext = os.path.splitext(filename)
    counter = 1
    new_filename = f"{base}{counter}{ext}"
    while os.path.exists(new_filename):
      counter += 1
      new_filename = f"{base}{counter}{ext}"
    filename = new_filename
  else:
    filename = filename

  with open(filename, "w", encoding="utf-8") as file:
    json.dump(emails, file, ensure_ascii=False, indent=4)

save_emails_to_json(filtered_emails, "emails.json")
```

Для фронт-енду розробленої системи використаємо фреймворк Flask, який

дозволяє запустити сервер на Python, створити ендпоінти та під'єднати сторінки з розміткою HTML.

Створюємо наш застосунок:

```
app = Flask(__name__)
CORS(app)
```

Далі створюємо ендпоінт, який буде повертати нам вигляд файлу Index.html, який являє собою головну сторінку системи, яка відображає повідомлення:

```
@app.route('/')
def index():
   with open('emails.json', 'r', encoding='utf-8') as file:
      emails = json.load(file)

   return render_template('index.html', emails=emails)
```

Далі створюємо окремий ендпоінт для перевірки повідомлення на предмет фішингу, який буде викликати інстанцію нашого класу та його метод check_phishing():

```
@app.route('/check_phishing', methods=['POST'])
def check_phishing():
   email_text = request.json['email_text']
   prediction = my_Instance.check_Phishing(email_text)
   return jsonify(prediction=prediction)
```

Таким чином, була розроблена програмна реалізація системи моніторингу електронної пошти на предмет фішингу. Використання моделі, яка поєднує у собі результати використання трьох високоточних алгоритмів машинного навчання, надає змогу класифікувати повідомлення більш якісно та з маленькою вірогідністю допущення помилки.

У ході програної реалізації був розроблений модуль експорту та інтеграції електронної пошти за допомогою бібліотеки imaplib, модуль виявлення фішингових ознак та модуль аналізу та класифікації. Також було розроблено серверну частину системи, відповідний файл розмітки HTML та стилі для нього.

Наступним кроком буде тестування системи.

### 3.3 Тестування розробленої системи

Для тестування системи необхідно запустити модуль, відповідальний за серверну та візуальну частини за допомогою команди python app.py.

Після успішного хостування серверу системи відображається інтерфейс

системи (рис. 3.5).

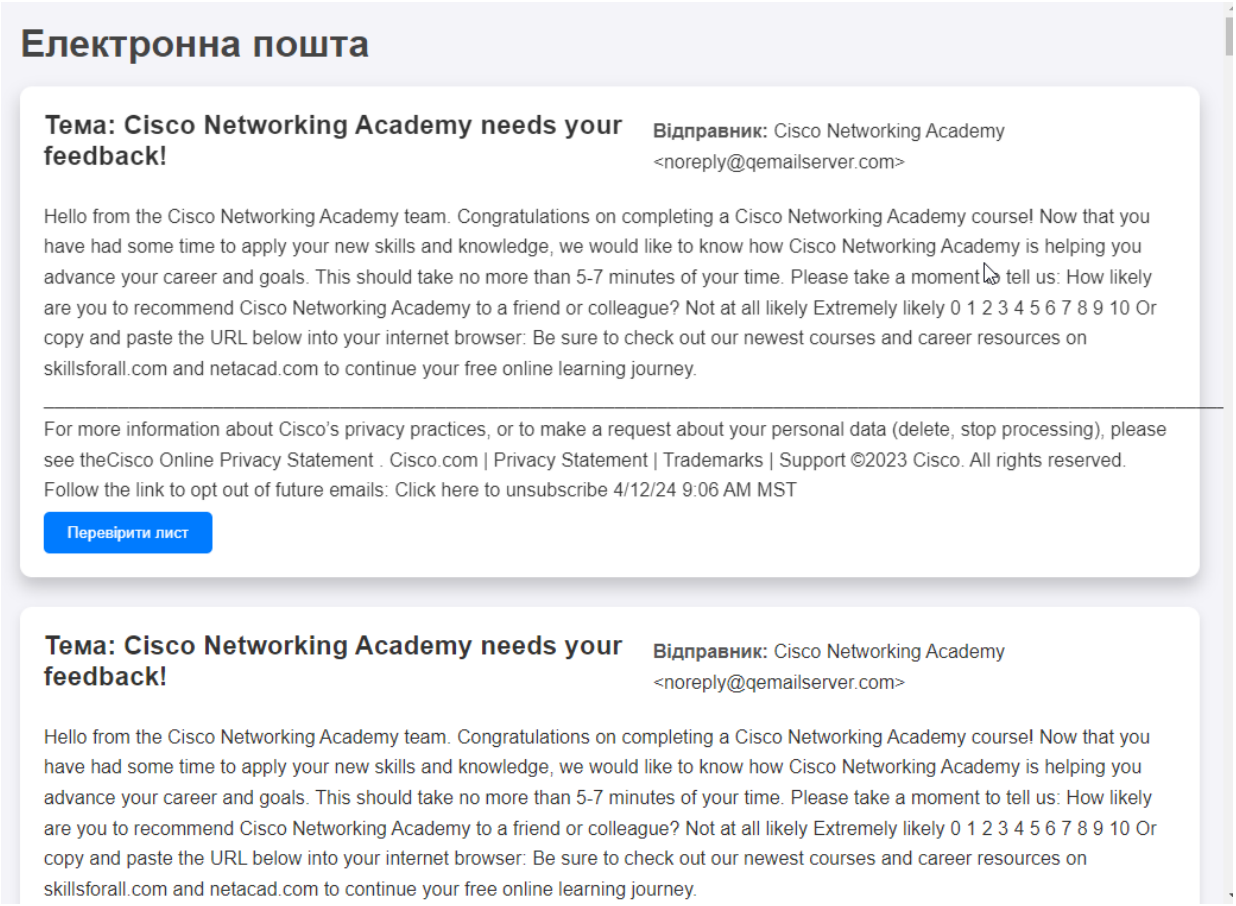



Рисунок 3.5 – Інтерфейс системи

Інтерфейс системи відображає наявні повідомлення електронної пошти, так як їх багато, наявна можливість прогорнути сторінку та переглянути усі. Кожне повідомлення це окремий блок, у якому відображається тема повідомлення, відправник та вміст. Кожне повідомлення можна класифікувати за допомогою відповідної кнопки.

Натиснувши кнопку, повідомлення відправляється у основний модуль системи, який класифікує повідомлення та повертає користувачу результат передбачення.

Для тестування системи, перевіримо кілька повідомлень. Першим повідомлення для перевірки оберемо сповіщення від платформи Discord про зміну пароля облікового запису. Натиснувши кнопку перевірки листа, отримуємо результат “Безпечне повідомлення” (рис. 3.6):

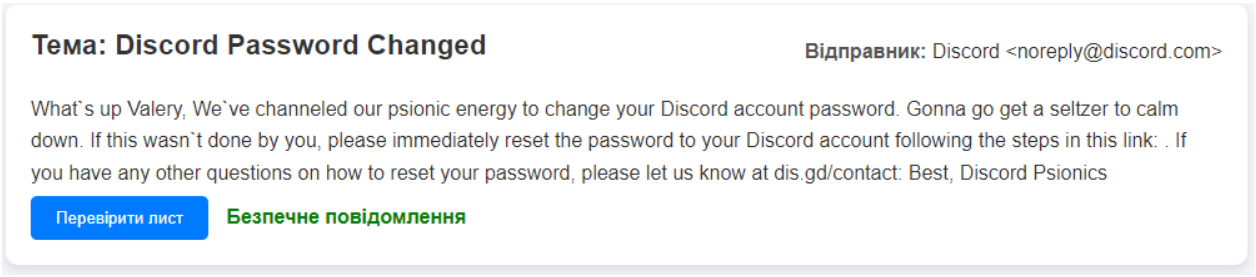



Рисунок 3.6 – Перше перевірене повідомлення

Це повідомлення було класифіковано як безпечне після його обробки основним модулем програми, так як це є офіційне повідомлення від Discord і результати передбачень це продемонстрували. Тут немає жодних підозрілих посилань, орфографічних помилок, відправником є офіційна електронна пошта платформи.

Наступне повідомлення для перевірки (рис. 3.7):

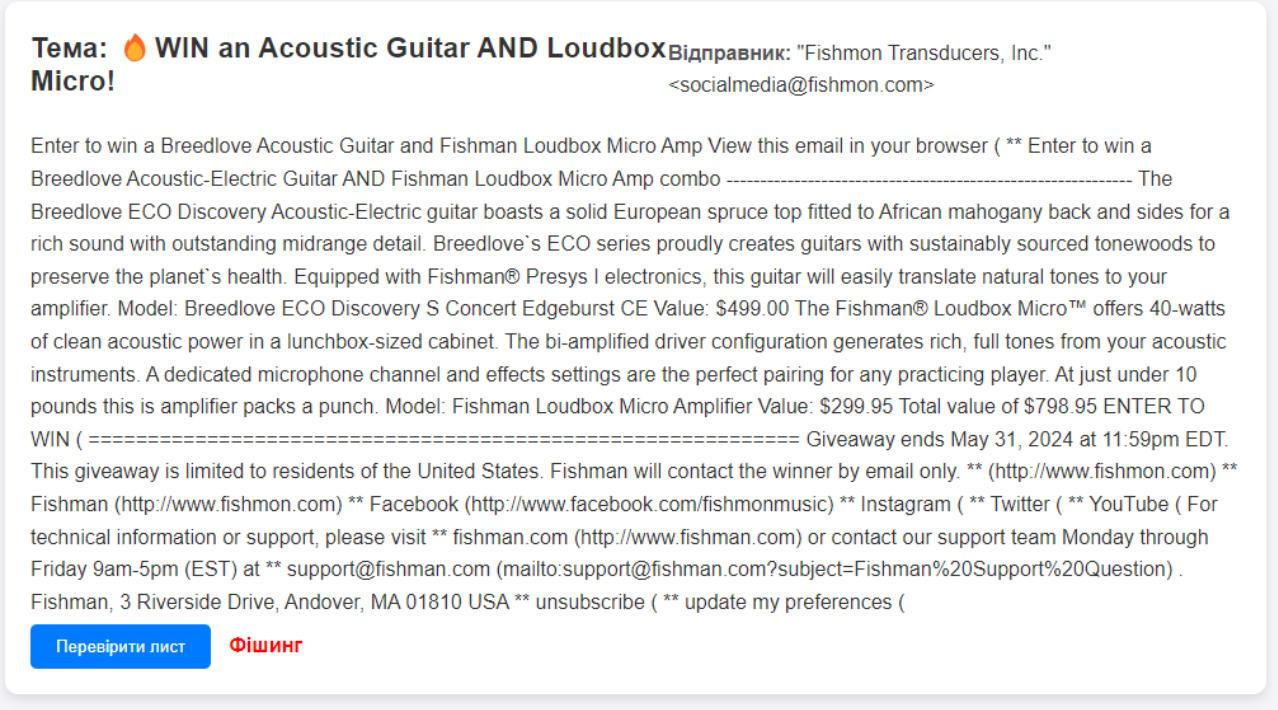



Рисунок 3.7 – Друге перевірене повідомлення

Це повідомлення було класифіковано як фішинг через ряд причин. У темі повідомлення є заклик до виграшу чогось. Відправником являється не перевірена електронна пошта з допущеними орфографічними помилками. Текст повідомлення має багато посилань та закликів до прийняття участі у фейковому розіграші (рис. 3.8).

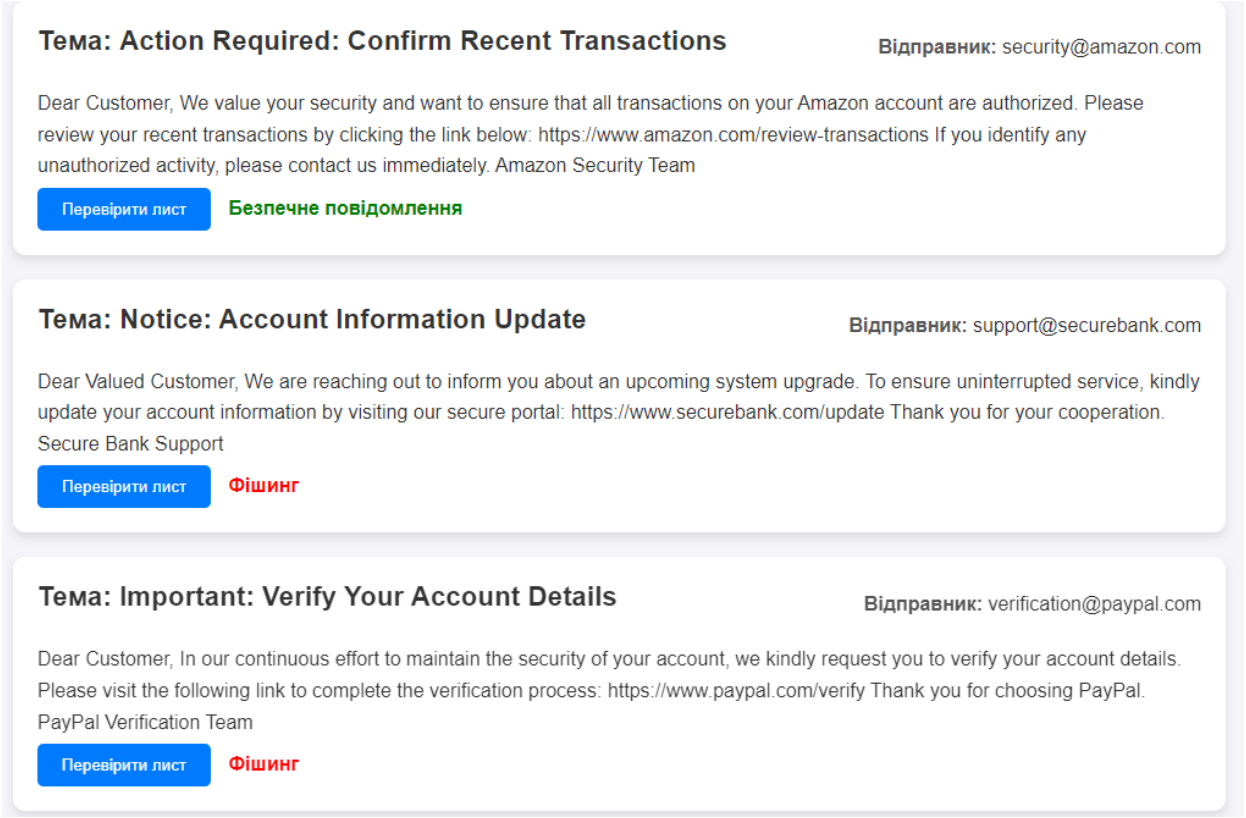



Рисунок 3.8 – Класифіковані повідомлення

Також, система може демонструвати помилкові передбачення, так як надати 100% точності моделі неможливо. Наприклад (рис. 3.9):

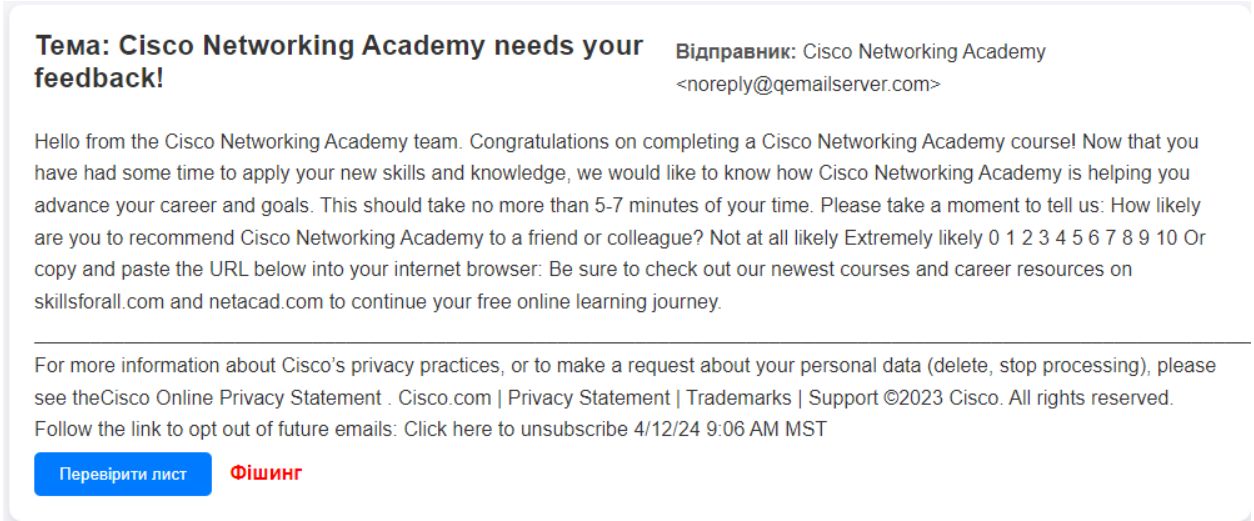



Рисунок 3.9 – Помилкове передбачення

Це повідомлення являє собою прохання надати відгук про курс пройдений на платформі Cisco, відправником є офіційна електронна адреса платформи, повідомлення майже не містить помилок. Але через похибку, система помилково класифікувала повідомлення.

Таким чином, було протестовано систему моніторингу електронної пошти на

предмет фішингу за допомогою методів машинного навчання. Результати тестування демонструють високоточну класифікацію повідомлень завдяки використанню моделі, навченої за допомогою трьох алгоритмів машинного навчання.

### 3.4 Висновок до розділу 3

У цьому розділі було представлено детальний опис процесу розробки системи. Було обрано оптимальну мову програмування та необхідні бібліотеки, які забезпечили ефективність та надійність програмного продукту. Розробка програмного коду та тренування моделі дозволили створити інструмент, здатний точно ідентифікувати фішингові спроби, що є важливим кроком у захисті користувачів від шкідливих атак.

Тестування розробленої системи показало її високу ефективність у виявленні фішингових листів завдяки вибору трьох найкращих методів машинного навчання. Система демонструє значний потенціал для подальшого розвитку та інтеграції з іншими інструментами кібербезпеки, що може забезпечити ще більшу захищеність від фішингових атак у майбутньому.

Завдяки своїй гнучкості та масштабованості, розроблена система може бути адаптована до різних умов експлуатації, що робить її цінним інструментом у боротьбі з кіберзлочинністю.

# ВИСНОВКИ

У рамках цієї дипломної роботи було успішно здійснено програмну реалізацію системи моніторингу електронної пошти, спрямовану на виявлення фішингових атак за допомогою методів машинного навчання. Було проведено детальний аналіз існуючих методів виявлення фішингу, що дозволило виявити, що фішинг залишається серйозною загрозою, незважаючи на значні успіхи в розробці методів його протидії. Також було розглянуто сучасні системи виявлення фішингу, визначено їхні переваги та недоліки.

На основі отриманих даних було розроблено детальну архітектуру системи та алгоритм її роботи. Ключовим елементом системи став комбінований підхід до створення мультикласифікатора, що дозволяє використовувати сильні сторони різних алгоритмів класифікації та методів кластеризації.

Для реалізації системи була обрана мова програмування Python, завдяки її гнучкості та широкій підтримці, що забезпечує легку інтеграцію з різними платформами та сервісами. Розроблений додаток пройшов ретельне тестування, включаючи перевірку точності виявлення фішингових атак.

Отже, запропоновані вдосконалення системи моніторингу виявилися надзвичайно ефективними оскільки поєднання результатів трьох методів машинного навчання дозволило значно підвищити точність виявлення фішингових атак. Система навчається на основі реальних даних та постійно адаптується до змін у тактиках фішерів. Подальше впровадження та оптимізація цієї системи можуть значно посилити кібербезпеку та забезпечити надійний захист від шахрайства у цифровому світі.

Таким чином, розроблена система є важливим кроком у боротьбі з фішингом, пропонуючи ефективний та адаптивний інструмент для виявлення та протидії цій загрозі.

# ПЕРЕЛІК ПОСИЛАНЬ

1. What is Phishing? [Електронний ресурс] / Official Blog // cisco.com – 2023 – Режим доступу: https://www.cisco.com/c/en/us/products/security/email-security/what-is-phishing.html

2. What is Phishing? [Електронний ресурс] / Official Blog // phishing.org – 2024 – Режим доступу: https://www.phishing.org/what-is-phishing

3. What is phishing attacks | Attack techniques & scam examples [Електронний ресурс] / Official Blog // imperva.com – 2023 – Режим доступу: https://www.imperva.com/learn/application-security/phishing-attack-scam/

4. What is a Phishing Attack? [Електронний ресурс] / Official Blog // ibm.com – 2022 – Режим доступу: https://www.ibm.com/topics/phishing

5. Why are phishing attacks still a huge problem? [Електронний ресурс] / Official Blog // ng-it.co.uk – 2022 – Режим доступу: https://ng-it.co.uk/blog-why-are-phishing-attacks-still-a-huge-problem/

6. Spam vs. Phishing [Електронний ресурс] / Original Article // webroot.com – 2024 – Режим доступу: https://www.webroot.com/us/en/resources/tips-articles/spam-vs-phishing

7. Phishing – What's The problem? [Електронний ресурс] / Alive Davies // tailon.net – 2022 – Режим доступу: https://talion.net/blog/phishing-whats-the-problem/

8. Internet Crime Complaint Center (IC3) | 2021 report [Електронний ресурс] 2021 – Режим доступу: https://www.ic3.gov/

9. 2024 Annual State Of Email Security Report [Електронний ресурс] / Cofense Team // cofense.com – 2024 – Режим доступу: https://cofense.com/annualreport/

10. Phising – the danger that lurks on the net [Електронний ресурс] / Original Article // treesolution.com – 2022 – Режим доступу: https://www.treesolution.com/news-english/phishing-the-danger-that-lurks-on-the-net

11. 19 Most Common Types of Phishing Attacks in 2024 [Електронний ресурс] / Kyle Chin // upguard.com – 2024 – Режим доступу: https://www.upguard.com/blog/types-of-phishing-attacks

12. What is Phishing? How Does It work, Prevention, Examples [Електронний ресурс] / Alexander S. Gillis // techtarget.com – 2024 – Режим доступу: https://www.techtarget.com/searchsecurity/definition/phishing

13. Phishing sites. How to recognize a fraudulent site [Електронний ресурс] / Official Blog // maxnet.ua – 2021 – Режим доступу: https://maxnet.ua/en/blog/fishingovye-sayty-kak-raspoznat-moshennicheskiy-sayt/

14. 5 Ways To Spot a Phishing Website [Електронний ресурс] / James MacKay // metacompliance.com – 2024 – Режим доступу: https://www.metacompliance.com/blog/phishing-and-ransomware/5-ways-to-identify-a-phishing-website

15. What is Machine Learning (ML)? [Електронний ресурс] / Official Blog // ibm.com – 2024 – Режим доступу: https://www.ibm.com/topics/machine-learning

16. Improved Phishing Attack Detection With Machine Learning / Sibel Kapan // mdpi.com – 2023 – Режим доступу: https://www.mdpi.com/2076-3417/13/24/13269#:~:text=In%20phishing%20attack%20detection%2C%20machine,classifiers%20directly%20influences%20detection%20performance.

17. Blocking and filtering with the whitelist and blacklist [Електронний ресурс] / Official Blog // ibm.com – 2023 – Режим доступу: https://www.ibm.com/docs/el/capm?topic=tests-blocking-filtering-whitelist-blacklist

18. Support Vector Machine (SVM) Algorithm [Електронний ресурс] / Original Article // geeksforgeeks.org – 2023 – Режим доступу: https://www.geeksforgeeks.org/support-vector-machine-algorithm/

19. Pros and Cons of SVM [Електронний ресурс] / Araharkhan // medium.com – 2023 – Режим доступу: https://medium.com/@azaharkhan2280/pros-and-cons-of-svm-4adaf9b105ce

20. Decision Tree Algorithm in Machine Learning [Електронний ресурс] / Original Article // jatatpoint.com – 2021 – Режим доступу: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

21. K-Nearest Neighbor (KNN) Algorithm [Електронний ресурс] / Original Article // geeksforgeeks.org – 2024 – Режим доступу:

https://www.geeksforgeeks.org/k-nearest-neighbours/

22. Barracuda Email Filter [Електронний ресурс] / Original Article // – 2023 – Режим доступу: https://www.barracuda.com/products/email-protection/email-security-gateway

23. Barracuda Email Protection detects malicious attacks and unauthorized activity [Електронний ресурс] / Industry news // meu.edu.jo – 2021 – Режим доступу: helpnetsecurity.com/2021/12/10/barracuda-email-protection/

24. Proofpoint [Електронний ресурс] / Original Article // – 2024 – Режим доступу: https://www.proofpoint.com/us/products/email-security-and-protection/email-protection

25. Phishing email detection by using machine learning techniques [Електронний ресурс] / Tariku Yabshe Chiksa // meu.edu.jo – 2022 – Режим доступу: https://www.meu.edu.jo/libraryTheses/590422b4d5dd8_1.pdf

26. Detection of E-Mail Phishing Attacks using Machine Learning and Deep Learning Dhruv Ratheen. Suman Man – International Journal of Computer Applications, 2022.

27. Must-know phishing statistics - updated for 2024 [Електронний ресурс] / Egress // egress.com – 2022 – Режим доступу: https://www.egress.com/blog/phishing/phishing-statistics-round-up

28. B. Rieder, Engines of Order: A Mechanology of Algorithmic Techniques. Amsterdam, Netherlands: Amsterdam Univ. Press, 2020.

29. A survey on phishing URL detection using artificial intelligence. / A. Vadariya, N. K. Jadav – Proc. Int. Conf. Recent Trends Mach: 2021, с. 9–20.

30. Heuristicbased strategy for phishing prediction. / C. M. R. D. Silva, E. L. Feitosa, and V. C. Garcia – A survey of URLbased approach. Comput. Secur., vol. 88, Jan. 2020.

31. Email Phishing Detection Using Machine Learning. / Sasirekha C, Nandhini R, Karthiga Mai N L, Bhuvaneshwari R S, Chandra V S. – International Journal of engineering research & technology, 2023.

32. Васильєв О. М. Програмування мовою Python: [підручник] / О. М.

Васильєв. – Україна: ВНТЛ, 2022. – 247 с.

33. Flynn Fisher. Learn Python Programming for Beginners. / Flynn Fisher. – Princeton University Press, 2020. – 1072 p.

34. Sebastian Raschka, Vahid Mirjalili. / Python Machine Learning. / Sebastian Raschka, Vahid Mirjalili. – Packt Publishing Ltd, 2021. – 132 p.

35. Andrew Park. / Python Machine Learning. The Ultimate Guide for Beginners to Master Machine Learning with Practical Applications to Artificial Intelligence and Deep Learning with Python. / Andrew Park. – Lightning source inc, 2021.

36. Building Email Spam Detection Model in Python [Електронний ресурс] / Original Article // dev.to – 2023 – Режим доступу: https://dev.to/shaikhshahid/building-email-spam-detection-model-in-python-2549

37. 5 Most Used Machine Learning Algorithms in Python [Електронний ресурс] / Original Article // mygreatlearning.com – 2024 – Режим доступу: https://www.mygreatlearning.com/blog/most-used-machine-learning-algorithms-in-python/

38. 40 Machine Learning Algorithms with Python [Електронний ресурс] / Original Article // medium.com – 2024 – Режим доступу: https://medium.com/coders-camp/40-machine-learning-algorithms-with-python-3defd764b961

39. Kaggle [Електронний ресурс] / Original Article // kaggle.com – 2024 – Режим доступу: https://www.kaggle.com/

# ДОДАТКИ

**Додаток А. Технічне завдання**

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

**ЗАТВЕРДЖУЮ**
Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор
_____ Юрій ЯРЕМЧУК
" 22 " березня 2024 р.

# ТЕХНІЧНЕ ЗАВДАННЯ

до бакалаврської дипломної роботи на тему:

«Розробка системи моніторингу електронної пошти на виявлення фішингових атак за допомогою методів машинного навчання»

08-72.БДР.014.00.095 ТЗ

Керівник бакалаврської дипломної роботи
д.ф., доц. каф. МБІС:
_____ Салієва О. В.
" 22 " березня

Вінниця – 2024 р.

**1. Найменування та область застосування**

Програмна реалізація системи моніторингу електронної пошти на виявлення фішингових атак за допомогою методів машинного навчання. Область застосування: захист користувачів від фішингових атак на електронну пошту.

**2. Підстава для розробки**

Розробка виконується на основі наказу ректора ВНТУ № 80 від 11.03.2024 р.

**3. Мета та призначення розробки**

3.1 Мета розробки: розробка ефективної системи моніторингу електронної пошти на виявлення фішингових атак за допомогою методів машинного навчання.

3.2 Призначення: розроблена програма аналізує електронні листи на предмет фішингу.

**4. Джерела розробки**

4.1 B. Rieder, Engines of Order: A Mechanology of Algorithmic Techniques. Amsterdam, Netherlands: Amsterdam Univ. Press, 2020.

4.2 A survey on phishing URL detection using artificial intelligence. / A. Vadariya, N. K. Jadav – Proc. Int. Conf. Recent Trends Mach: 2021, с. 9–20.

4.3 Heuristicbased strategy for phishing prediction. / C. M. R. D. Silva, E. L. Feitosa, and V. C. Garcia – A survey of URLbased approach. Comput. Secur., vol. 88, Jan. 2020.

4.4 Email Phishing Detection Using Machine Learning. / Sasirekha C, Nandhini R, Karthiga Mai N L, Bhuvaneshwari R S, Chandra V S. – International Journal of engineering research & technology, 2023.

**5. Вимоги до програми**

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.1.3 Програмний засіб повинен виконувати процес виявлення

фішингових листів.

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.2 Бази даних повинні бути налаштовані на автоматичне створення резервних копій;

5.2.3 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Pentium 1500 МГц і подібні до них;

– оперативна пам’ять – не менше 512 Mb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

**6. Вимоги до програмної документації**

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.4

**7. Вимоги до технічного захисту інформації**

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Неможливість отримання доступу користувачів до інформаційних ресурсів.

**8. Техніко-економічні показники**

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

## 9. Стадії та етапи розробки

| № з/п | Назва етапів бакалаврської дипломної роботи | Початок | Закінчення |
|---|---|---|---|
| 1. | Визначення напрямку бакалаврської роботи, формулювання теми | 21.03.2024 | 23.03.2024 |
| 2. | Аналіз предметної області обраної теми | 24.03.2024 | 22.04.2024 |
| 3. | Розробка алгоритму роботи | 11.05.2024 | 19.05.2024 |
| 4. | Написання бакалаврської роботи на основі розробленої теми | 20.05.2024 | 24.05.2024 |
| 5. | Передзахист бакалаврської дипломної роботи | 24.05.2024 | 25.05.2024 |
| 6. | Виправлення, уточнення, корегування бакалаврської дипломної роботи | 27.05.2024 | 11.06.2024 |
| 7. | Захист бакалаврської дипломної роботи | 12.06.2024 | 17.06.2024 |

## 10. Порядок контролю та прийому

10.1 До приймання бакалаврської дипломної роботи надається:

- ПЗ до бакалаврської дипломної роботи;
- демонстрація результату бакалаврської дипломної роботи;
- презентація;
- відзив керівника роботи;
- відзив рецензента.

Технічне завдання до виконання прийняла _____ Марчук В. О.

# Додаток Б. Лістинг коду

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import re
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.layers import Embedding,GRU,LSTM,Bidirectional,SimpleRNN
from tensorflow.keras.utils import pad_sequences
from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from keras.layers import Dense,Dropout
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score,f1_score,classification_report,ConfusionMatrixDisplay,confusion_matrix
import tensorflow as tf
import warnings

warnings.filterwarnings('ignore')

df = pd.read_csv(r'D:/phishing/content/Phishing_Email.csv')
df.head()

df.isna().sum()

df.drop(['Unnamed: 0'],axis=1,inplace=True)
df.dropna(inplace=True,axis=0)
df.drop_duplicates(inplace=True)

lbl = LabelEncoder()
df['Email Type'] = lbl.fit_transform(df['Email Type']) ### 0 позначає фішингове повідомлення, 1 - безпечне

def preprocess_text(text):
    text = re.sub(r'[^\w\s]', '', text)

    text = text.lower()

    text = re.sub(r'\s+', ' ', text).strip()
    return text

df["Email Text"] =df["Email Text"].apply(preprocess_text)

tf = TfidfVectorizer(stop_words='english',max_features=10000)
feature_x = tf.fit_transform(df['Email Text']).toarray()

y_tf = np.array(df['Email Type'])

X_tr,X_tst,y_tr,y_tst = train_test_split(feature_x,y_tf,test_size=0.2,random_state=0)

nb = MultinomialNB()
nb.fit(X_tr,y_tr)
rnf = RandomForestClassifier()
rnf.fit(X_tr,y_tr)
```

```
lg = LogisticRegression()
lg.fit(X_tr,y_tr)

class MyTestClass:
    def check_Phishing(email):
        email = re.sub(r'[^\w\s]', '', email)
        email = re.sub(r'\s+', ' ', email).strip()
        email = email.lower()
        email_tfidf = tf.transform([email])

        prediction_nb = nb.predict(email_tfidf)
        prediction_rnf = rnf.predict(email_tfidf)
        prediction_lg = lg.predict(email_tfidf)

        nb_phishing_ = 1 if prediction_nb[0] == 0 else 0
        rnf_phishing_count = 1 if prediction_rnf[0] == 0 else 0
        lg_phishing_count = 1 if prediction_lg[0] == 0 else 0

        nb_safe_count = 1 if prediction_nb[0] == 1 else 0
        rnf_safe_count = 1 if prediction_rnf[0] == 1 else 0
        lg_safe_count = 1 if prediction_lg[0] == 1 else 0

        phishing_count = nb_phishing_count + rnf_phishing_count + lg_phishing_count
        safe_count = nb_safe_count + rnf_safe_count + lg_safe_count

        if phishing_count > safe_count:
            return "Фішинг"
        else:
            return "Безпечне повідомлення"
import imaplib
import email
from email.header import decode_header
import re
from bs4 import BeautifulSoup

import json
import os

username = "valerymarchuk2103@gmail.com"
app_password = "xiwm vlve pmxw ysbg"

def extract_emails():
    emails = []

    mail = imaplib.IMAP4_SSL("imap.gmail.com")

    mail.login(username, app_password)

    mail.select("inbox")

    status, messages = mail.search(None, "ALL")

    mail_ids = messages[0].split()

    for mail_id in mail_ids[-300:]:
        status, msg_data = mail.fetch(mail_id, "(RFC822)")

        for response_part in msg_data:
            if isinstance(response_part, tuple):
                msg = email.message_from_bytes(response_part[1])
```

```
            subject, encoding = decode_header(msg["Subject"])[0]
            if isinstance(subject, bytes):
                subject = subject.decode(encoding if encoding else "utf-8")
            from_ = msg.get("From")

            if msg.is_multipart():
                for part in msg.walk():

                    content_type = part.get_content_type()
                    content_disposition = str(part.get("Content-Disposition"))

                    try:
                        body = part.get_payload(decode=True).decode()
                    except:
                        pass

                    if content_type == "text/plain" and "attachment" not in content_disposition:
                        soup = BeautifulSoup(body, "html.parser")
                        text_body = soup.get_text()
                        text_body = re.sub(r'\s+', ' ', text_body).strip()
                        text_body = re.sub(r"\b(\w+)'(\w+)\b", r"\1`\2", text_body)
                        emails.append({"Subject": subject, "From": from_, "Body": text_body})
            else:
                content_type = msg.get_content_type()

                body = msg.get_payload(decode=True).decode()
                if content_type == "text/plain":
                    soup = BeautifulSoup(body, "html.parser")
                    text_body = soup.get_text()
                    text_body = re.sub(r'\s+', ' ', text_body).strip()
                    text_body = re.sub(r"\b(\w+)'(\w+)\b", r"\1`\2", text_body)
                    emails.append({"Subject": subject, "From": from_, "Body": text_body})

    mail.close()
    mail.logout()

    return emails

def filter_cyrillic(emails):
    filtered_emails = []
    pattern = re.compile(r'[\u0400-\u04FF]+')

    for email in emails:
        if not pattern.search(email["Subject"]) and not pattern.search(email["From"]) and not
pattern.search(email["Body"]):
            filtered_emails.append(email)

    return filtered_emails

all_emails = extract_emails()

filtered_emails = filter_cyrillic(all_emails)

def save_emails_to_json(emails, filename):
    if os.path.exists(filename):
        base, ext = os.path.splitext(filename)
        counter = 1
        new_filename = f"{base}{counter}{ext}"
        while os.path.exists(new_filename):
```

```
            counter += 1
            new_filename = f"{base}{counter}{ext}"
        filename = new_filename
    else:
        filename = filename

    with open(filename, "w", encoding="utf-8") as file:
        json.dump(emails, file, ensure_ascii=False, indent=4)

save_emails_to_json(filtered_emails, "emails.json")

from flask import Flask, jsonify, render_template, request
from flask_cors import CORS
from main import MyTestClass

import json

app = Flask(__name__)
CORS(app)

@app.route('/')
def index():
    with open('emails.json', 'r', encoding='utf-8') as file:
        emails = json.load(file)

    return render_template('index.html', emails=emails)

my_Instance = MyTestClass
@app.route('/check_phishing', methods=['POST'])
def check_phishing():
    email_text = request.json['email_text']
    prediction = my_Instance.check_Phishing(email_text)
    return jsonify(prediction=prediction)

if __name__ == '__main__':
    app.run(debug=True)

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Виявлення фішингу</title>
    <link rel="icon" href="{{ url_for('static', filename='icon.png') }}" type="image/png">
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f9;
            margin: 20px;
            color: #333;
        }
        h1 {
            color: #444;
        }
        .email {
            background-color: #fff;
            border-radius: 10px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            padding: 20px;
            margin-bottom: 20px;
```

```
    transition: transform 0.3s, box-shadow 0.3s;
}
.email:hover {
    transform: translateY(-5px);
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
}
.email h3 {
    margin: 0 0 10px;
    font-size: 1.4em;
    color: #333;
}
.email p {
    margin: 5px 0;
    line-height: 1.6;
}
.email .additional {
    margin-top: 15px;
    font-style: italic;
    color: #888;
}
.json-output {
    white-space: pre-wrap;
    background-color: #fff;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    margin-top: 20px;
}
.email-info {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 10px;
}
.email-info strong {
    color: #555;
}
.email-body {
    margin-top: 15px;
}
.check-btn {
    background-color: #007bff;
    color: white;
    border: none;
    padding: 10px 20px;
    cursor: pointer;
    border-radius: 5px;
    transition: background-color 0.3s;
}
.check-btn:hover {
    background-color: #0056b3;
}
.result {
    margin-left: 10px;
    font-weight: bold;
    color: #333;
}
.safe-mail {
    color: green;
}
```

```
      .phishing {
        color: red;
      }
    </style>
    <script>
      function checkPhishing(emailText, resultElement) {
        fetch('/check_phishing', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json'
          },
          body: JSON.stringify({ email_text: emailText })
        })
        .then(response => response.json())
        .then(data => {
          resultElement.textContent = data.prediction;
          if (data.prediction === 'Фішинг') {
            resultElement.classList.add('phishing');
            resultElement.classList.remove('safe-mail');
          } else {
            resultElement.classList.add('safe-mail');
            resultElement.classList.remove('phishing');
          }
        })
        .catch(error => console.error('Error:', error));
      }
    </script>
</head>
<body>
    <h1>Електронна пошта</h1>
    {% for email in emails %}
    <div class="email">
      <div class="email-info">
        <h3>Тема: {{ email.Subject }}</h3>
        <p><strong>Відправник:</strong> {{ email.From }}</p>
      </div>
      <div class="email-body">
        <p>{{ email.Body }}</p>
      </div>
      <div>
        <button class="check-btn" onclick="checkPhishing('{{ email.Body|safe }}',
this.nextElementSibling)">Перевірити лист</button>
        <span class="result"></span>
      </div>
    </div>
    {% endfor %}
</body>
</html>

def fit(self, X, y, sample_weight=None):
      X, y = self._check_X_y(X, y)
      _, n_features = X.shape

      labelbin = LabelBinarizer()
      Y = labelbin.fit_transform(y)
      self.classes_ = labelbin.classes_
      if Y.shape[1] == 1:
        if len(self.classes_) == 2:
          Y = np.concatenate((1 - Y, Y), axis=1)
        else:
```

```
            Y = np.ones_like(Y)

        if sample_weight is not None:
            Y = Y.astype(np.float64, copy=False)
            sample_weight = _check_sample_weight(sample_weight, X)
            sample_weight = np.atleast_2d(sample_weight)
            Y *= sample_weight.T

        class_prior = self.class_prior
        n_classes = Y.shape[1]
        self._init_counters(n_classes, n_features)
        self._count(X, Y)
        alpha = self._check_alpha()
        self._update_feature_log_prob(alpha)
        self._update_class_log_prior(class_prior=class_prior)
        return self

    def _init_counters(self, n_classes, n_features):
        self.class_count_ = np.zeros(n_classes, dtype=np.float64)
        self.feature_count_ = np.zeros((n_classes, n_features), dtype=np.float64)

def predict(self, X):

        check_is_fitted(self)
        X = self._check_X(X)
        jll = self._joint_log_likelihood(X)
        return self.classes_[np.argmax(jll, axis=1)]
        proba = self.predict_proba(X)

        if self.n_outputs_ == 1:
            return self.classes_.take(np.argmax(proba, axis=1), axis=0)

        else:
            n_samples = proba[0].shape[0]
            class_type = self.classes_[0].dtype
            predictions = np.empty((n_samples, self.n_outputs_), dtype=class_type)

            for k in range(self.n_outputs_):
                predictions[:, k] = self.classes_[k].take(
                    np.argmax(proba[k], axis=1), axis=0
                )

            return predictions

    def predict_proba(self, X):
        check_is_fitted(self)
        X = self._validate_X_predict(X)

        n_jobs, _, _ = _partition_estimators(self.n_estimators, self.n_jobs)

        all_proba = [
            np.zeros((X.shape[0], j), dtype=np.float64)
            for j in np.atleast_1d(self.n_classes_)
        ]
        lock = threading.Lock()
        Parallel(n_jobs=n_jobs, verbose=self.verbose, require="sharedmem")(
            delayed(_accumulate_prediction)(e.predict_proba, X, all_proba, lock)
            for e in self.estimators_
        )
```

```
        for proba in all_proba:
            proba /= len(self.estimators_)

        if len(all_proba) == 1:
            return all_proba[0]
        else:
            return all_proba

def predict(self, X):
        xp, _ = get_namespace(X)
        scores = self.decision_function(X)
        if len(scores.shape) == 1:
            indices = xp.astype(scores > 0, indexing_dtype(xp))
        else:
            indices = xp.argmax(scores, axis=1)

        return xp.take(self.classes_, indices, axis=0)

    def _predict_proba_lr(self, X):
        prob = self.decision_function(X)
        expit(prob, out=prob)
        if prob.ndim == 1:
            return np.vstack([1 - prob, prob]).T
        else:
            prob /= prob.sum(axis=1).reshape((prob.shape[0], -1))
            return prob

 def predict(self, X):
        check_is_fitted(self)
        X = self._check_X(X)
        jll = self._joint_log_likelihood(X)
        return self.classes_[np.argmax(jll, axis=1)]

    def predict_log_proba(self, X):
        check_is_fitted(self)
        X = self._check_X(X)
        jll = self._joint_log_likelihood(X)
        log_prob_x = logsumexp(jll, axis=1)
        return jll - np.atleast_2d(log_prob_x).T
```

# Додаток В. Ілюстративний матеріал

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА
на тему:
Розробка системи моніторингу електронної пошти на виявлення фішингових атак за допомогою методів машинного навчання

Виконала студентка групи 2КІТС-20б Марчук Валерія
Науковий керівник: д.ф., доц. каф. МБІС Салієва О.В.

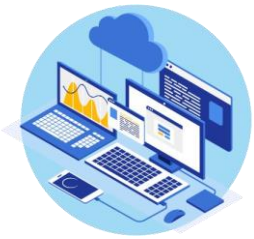
Вінниця ВНТУ – 2024

## Актуальність

Захист особистих даних

Зростання фішингових кількості атак

Підвищення ефективності



# Мета

Метою роботи є розробка ефективної системи моніторингу електронної пошти на виявлення фішингових атак за допомогою методів машинного навчання, яка забезпечить високий рівень точності у виявленні шахрайських повідомлень, мінімізуючи при цьому помилкові результати.

# Об’єкт та предмет дослідження

Об’єктом дослідження є методів виявлення фішингових атак на електронній пошті за допомогою машинного навчання.

Предметом дослідження є розробка системи моніторингу електронної пошти, яка використовує методи машинного навчання для автоматичного виявлення та класифікації фішингових листів.

# Завдання

1) Проаналізувати актуальність тематики, процес фішингу електронної пошти та визначити життєвий цикл фішингових атак;

2) Дослідити підходи та методи розпізнавання фішингових атак в електронній пошті за допомогою методів машинного навчання;

3) Розробити алгоритм роботи системи моніторингу електронної пошти;

4) Здійснити програмну реалізацію системи моніторингу електронної пошти на виявлення фішингових атак;

5) Оцінити ефективність розробленої системи.

# Порівняння з аналогами

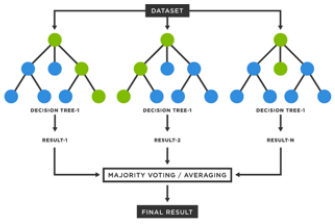
| Система | Переваги | Недоліки | Кількісні характеристики | Ціна |
|---|---|---|---|---|
| **Mimecast Email Security** | Безкоштовна та з відкритим кодом, проста в налаштуванні, широке співтовариство | Може мати нижчий рівень виявлення, потребує регулярного оновлення | Рівень виявлення фішингу: 89%, рівень хибних спрацьовувань: 0,5%, | Платна |
| **GFI MailEssentials** | Простота використання, широкий спектр функцій, підтримка | Платна, складність розширення | Рівень виявлення фішингу: 79-80%, кількість помилкових спрацьовувань: 0.3-0.5% | Платна |
| **Barracuda Essentials** | Широкий спектр функцій, простота використання, масштабованість | Платна, складність розгортання | Рівень виявлення фішингу: 81%, кількість помилкових спрацьовувань: 0.03% | Платна |
| **Proofpoint Email Protection** | Широкий спектр функцій, ефективність виявлення, підтримка | Платна, складність розгортання | Рівень виявлення фішингу: 79%, кількість помилкових спрацьовувань: 0.04% | Платна |

# Новизна розробки

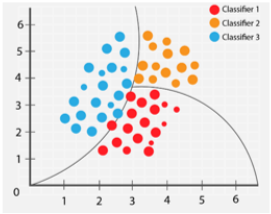
Інноваційність розробленої системи полягає в поєднанні результатів трьох різних методів машинного навчання для досягнення високої точності прогнозування типу повідомлення. Запропонований підхід дозволяє використовувати сильні сторони різних методів класифікації, що підвищує загальну точність виявлення фішингових електронних листів.

# Використані методи класифікації листів

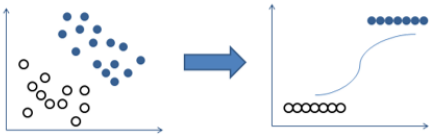
Random Forest - Випадковий ліс



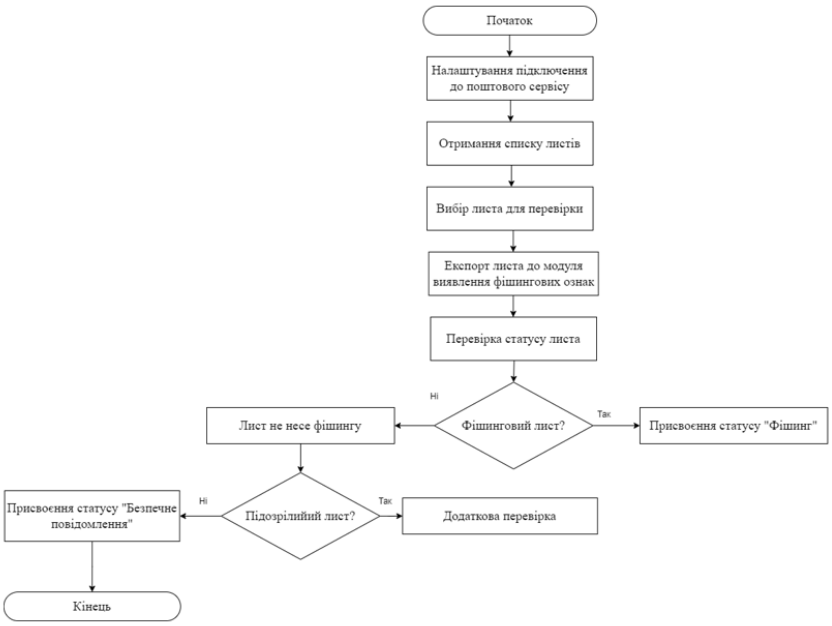

Naive Bayes - Наївний Байєс




Logistic Regression - Логістична регресія



# Алгоритм розробленої системи




# Інструменти розробки

Мова програмування

Фреймворк для машинного навчання

Середовище розробки

# Результати точності передбачень моделі за обраними методами:

```
точність передбачень використовуючи алгоритм Naive Bayes: 97.52 %
f1 результат використовуючи алгоритм Naive Bayes: 97.99 %
звіт:
              precision    recall  f1-score   support

           0       0.97      0.96      0.97      1351
           1       0.98      0.98      0.98      2157

    accuracy                           0.98      3508
   macro avg       0.97      0.97      0.97      3508
weighted avg       0.98      0.98      0.98      3508
```

Наївний Баєс

```
точність передбачень використовуючи алгоритм Random Forest:97.63 %
f1 результат використовуючи алгоритм Random Forest: 98.06 %
звіт:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      1351
           1       0.99      0.97      0.98      2157

    accuracy                           0.98      3508
   macro avg       0.97      0.98      0.98      3508
weighted avg       0.98      0.98      0.98      3508
```

Випадковий Ліс

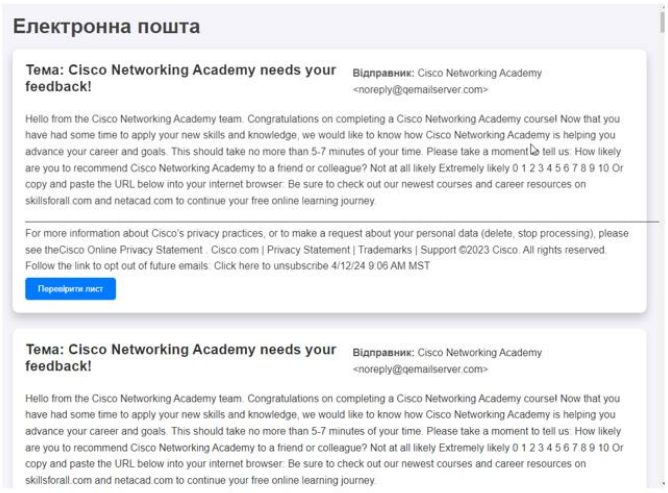
```
точність передбачень використовуючи алгоритм Logistic Regression:97.95 %
f1 результат використовуючи алгоритм Logistic Regression: 98.34 %
звіт:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97      1351
           1       0.98      0.99      0.98      2157

    accuracy                           0.98      3508
   macro avg       0.98      0.98      0.98      3508
weighted avg       0.98      0.98      0.98      3508
```

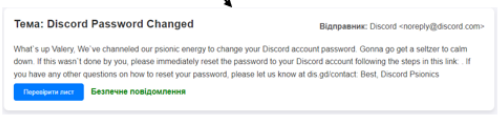
Логістична регресія

# Інтерфейс розробленої системи

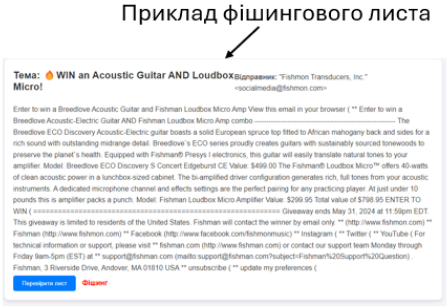


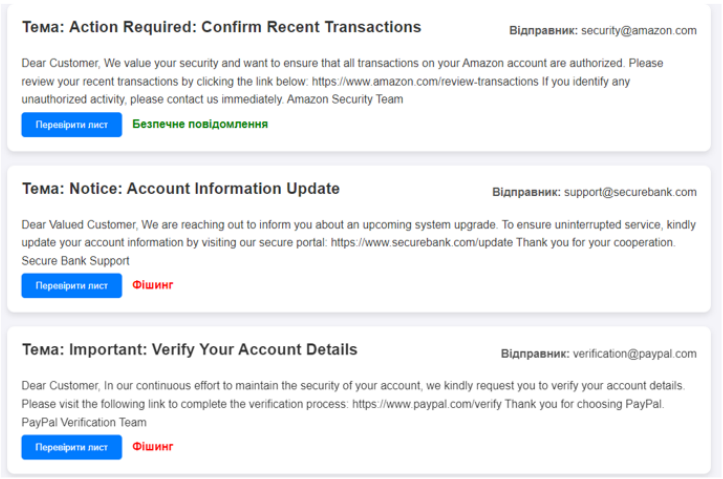
# Тестування результатів роботи

Приклад безпечного листа




Приклад фішингового листа




# Апробація

Тези доповідей даної тематики представлені на LII Всеукраїнській науково-технічній конференції факультету менеджменту та інформаційної безпеки.

# Висновки

У рамках даної дипломної роботи було успішно здійснено програмну реалізацію системи моніторингу електронної пошти, спрямовану на виявлення фішингових атак за допомогою методів машинного навчання.

Було проведено детальний аналіз існуючих методів виявлення фішингу, розглянуто сучасні системи виявлення фішингу, визначено їхні переваги та недоліки.

На основі отриманих даних було розроблено детальну архітектуру системи та алгоритм її роботи. Для реалізації системи була обрана мова програмування Python. Після чого розроблена система пройшла ретельне тестування, включаючи перевірку точності виявлення фішингових атак.

Таким чином, розроблена система є важливим кроком у боротьбі з фішингом, пропонуючи ефективний та адаптивний інструмент для виявлення цій загрозі.

Дякую за увагу!

## Додаток Г. Протокол перевірки на антиплагіат

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ
ЗАПОЗИЧЕНЬ

Назва роботи: Розробка системи моніторингу електронної пошти на виявлення фішингових атак за допомогою методів машинного навчання

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ: Кафедра менеджменту та безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
(кафедра, факультет)

**Показники звіту подібності Unicheck**

Оригінальність 85 % Схожість 15 %

Аналіз звіту подібності (відмітити потрібне):

**1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ (підпис) Коваль Н.П. (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ (підпис) Марчук В.О. (прізвище, ініціали)

Керівник роботи _____ (підпис) Салієва О.В. (прізвище, ініціали)