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

До захисту допущено
В.о. завідувача кафедри

_____ **Микола ГРАЙВОРОНСЬКИЙ**
(підпис)

«_____»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи, технології та математичні методи кібербезпеки»
спеціальності: 125 «Кібербезпека»

на тему: «Розробка механізму та набору службових прикладних програм для системи виявлення фішингу»

Виконав : здобувач вищої освіти **IV** курсу, групи ФБ-71.
(шифр групи)

Романюк Дмитро Олегович. _____.
(прізвище, ім'я, по батькові) (підпис)

Керівник : доцент кафедри інформаційної безпеки.

Гальчинський Л.Ю.
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім'я, по батькові) (підпис)

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

Київ - 2021 року

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

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

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____Микола ГРАЙВОРОНСЬКИЙ
(підпис)
«_____»_____2021 р.

ЗАВДАННЯ
на дипломну роботу здобувачу вищої освіти

Романюку Дмитру Олеговичу

1.Тема роботи: «Розробка механізму та набору службових прикладних програм для системи виявлення фішингу», науковий керівник:
Гальчинський Леонід Юрійович, доцент кафедри інформаційної безпеки, затверджені наказом по університету
від « » _____ 2021 року №

2.Термін подання студентом роботи: 07 червня 2021 р.

3. Вихідні дані до роботи: попередні дослідження методів фішингу.

4. Зміст роботи: Розробка методів виявлення фішингу, застосування машинного навчання для класифікації електронних листів.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) Розробка методів розпізнавання фішингу в електронних листах - презинтація
6. Дата видачі завдання: 10.02.2021

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Отримання завдання	10.02.2021	
2	Вивчення та аналіз літератури	11.02.2021– 20.03.2021	
3	Розробка програми для виявлення ознак фішингу	21.03.2021 – 10.04.2021	
4	Проходження переддипломної практики	12.04.2021 – 16.05.2021	
5	Написання дипломної роботи	01.04.2021 – 02.06.2021	
6	Отримання допуску до захисту	03.06.2021	
7	Захист дипломної роботи	16.06.2021	

Здобувач вищої освіти _____ Дмитро РОМАНЮК
(підпис) (Власне ім’я, ПРІЗВИЩЕ)

Керівник роботи _____ Леонід ГАЛЬЧИНСЬКИЙ
(підпис) (Власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Дипломна робота має обсяг 58 сторінки, містить 10 рисунків, 1 таблицю, 13 літературних джерел та один додаток.

Об’єкт дослідження: Ознаки які притаманні методам фішингу в електронних листах.

Дана робота містить опис методів які використовують злочинці для фішингової атаки. У ході роботи було проаналізовано ці методи, та представлення правил перевірки, які можуть виявити ці ознаки в електронних листах. Після цього було розроблено антифішингову програму для виявлення таких ознак в електронних листах, та реалізовано машинне навчання для класифікації їх.

Ключові слова: фішинг, ознаки фішингу, машинне навчання, аналіз, класифікація.

ABSTRACT

The work includes 58 pages, 10 figures, 1 table, 13 references and one appendix.

Object of research: description of methods of phishing in e-mails.

This work describes the methods used by criminals for phishing attacks. In the course of the work, these methods were analyzed, and the rules of verification that can detect these features in e-mails was considered. After that, an anti-phishing program was developed to detect phishing features in e-mails, and machine learning was implemented to classify them.

Key words: phishing, features of phishing, machine learning, analysis, classification.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

APWG – Робоча група з протидії фішингу

BEC – business email compromise

SaaS – Software as a service

SVM – Support vector machine

KNN – K-Nearest neighbors

HTML – HyperText Markup Language

URL – Uniform Resource Locator

ВСТУП

Фішинг-атаки – це підроблені повідомлення, які надходять з надійного джерела, але можуть скомпрометувати всі типи джерел даних. Атаки можуть полегшити доступ до ваших облікових записів в інтернеті та особистих даних, отримати дозволи на модифікацію та компрометацію підключених систем, таких як термінали торгових точок та системи обробки замовлень, а в деяких випадках викрадати цілі комп'ютерні мережі, доки не буде виплачено викуп. Іноді хакери задоволені тим, що отримують ваші особисті дані та дані кредитної картки для отримання фінансової вигоди. В інших випадках фішингові електронні листи надсилаються для збору інформації про вхід співробітників або інших деталей для використання у більш зловмисних атаках проти кількох осіб або конкретної компанії.

На сьогоднішній день статистика показує досить великий відсоток користувачів, які не можуть розпізнати фішингові листи. Досить велика кількість організацій зазнали фішингової атаки, тому цей проект спрямований на захист користувачів від данної атаки. Розроблений механізм виявляє притаманні фішинговим листам елементи і попереджає користувачів про небезпеку. Також прикладна програма для збільшення ефективності попередньо проходить машинне навчання на виявленних раніше фішингових листах.

Актуальність роботи випливає з того, що хоча все більше користувачів знайомляться з методами фішингу, це не гарантія не бути ошуканим, тому що, людський фактор відіграє в цьому велику роль.

Метою роботи є розробка прикладної програми з застосуванням машинного навчання для розпізнавання фішингу в електронних листах.

Науковою новизною можна виділити двохетапну процедуру виявлення фішинг-листів.

Практичне застосування: Отримані результати можуть допомогти розпізнати фішингові електронні листи, і проаналізувавши дані, можна вирахувати які методи фішингу застосовуються в парі.

Завдання роботи: Програмна класифікація електронних листів, та перевірка наявності ознак фішингу.

Публікації. Роботу було опубліковано на XIX Всеукрахнській науково-практичні конференції студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики»

1.АНАЛІЗ ЗАГРОЗИ ТА ЇЇ АКТУАЛЬНІСТЬ

1.1 Небезпека фішингу

«Фішинг» - це хакерський термін, створений шахраями, які «виловлюють» конфіденційну інформацію, в основному використовуючи підроблені електронні листи і підроблені веб-сайти, які виступають в якості «приманки», а облікові записи жертви - як мережевий «фішинг». Фішинг - це компонент соціальної інженерії, за допомогою якого людина намагається запросити і вкрасти конфіденційну інформацію у користувача або співробітника, видаючи себе за законну особу, зазвичай у відомих фінансових установ або закладів електронної комерції , оскільки основною метою є для отримання грошей обманним шляхом. Шахраї можуть маніпулювати користувачами замість апаратних / програмних систем, де перешкоди для технологічного злому значно зросли. Фішинг - одне з найпоширеніших видів інтернет-шахрайства. Фішингові атаки приймають дві форми:

1) спроби обдурити жертв, щоб змусити їх 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 - ось кілька популярних випадків, коли організації і їх клієнти фінансово постраждали від фішингових атак [4]. Хоча великий процент фішингових атак націлений на фінанси, це не є правило, шахраї можуть бути зацікавленні іншими приватними даними. Шахраї придумують все більш ефективніші і нові методи, використовують всі сучасні технології, тобто месенджери, трояни, та популярні соціальні мережі. Проектування та виконання фішингової атаки не вимагає від шахраїв великої попередньої підготовки, як інтелектуальної так і технічної. Головним інструментом є обман,

маніпуляції і людські емоції, все це допомагає фішерам змусити жертву розкрити приватну інформацію[1].

Цінна інформація, така як облікові дані для аутентифікації користувача і особиста конфіденційна інформація, може бути отримана шляхом використання вразливостей в межах розуміння 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 і фахівці з безпеки повинні розуміти, як користувачі взаємодіють з цими атаками.

На сьогоднішній день фішинг це велика проблема в кіберпространстві, злочинці можуть полювати на окремих користувачів (spear phishing), так і на цілі підприємства та урядові організації [7]. Хоча обізнаність громадськості про фішингові атаки в цілому зростає, нові і новаторські схеми або «атаки нульового дня», які обходять фішингові фільтри і чорні списки, часто все ж дозволяють обдурити навіть досвідчених користувачів.

Звичайним методом фішингової атаки є спам-розсилка. Спам - це небажана електронна пошта. 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. І навпаки, фішингові атаки призначені для отримання призначеної для користувача інформації, такої як ідентифікатори входу і паролі, щоб зловмисник міг отримати доступ до таких речей, як соціальна мережа і дані банківського рахунку.

Фішинг через електронну пошту все далі і далі розповсюджується. Це один з найпопулярніших методів шахрайства, який використовується для крадіжки персональних даних та фінансових. Особи, які відповідають на фішингові

електронні листи і вводять запитувану фінансову або особисту інформацію в електронні листи, веб-сайти або спливаючі вікна, піддають себе і свої установи ризику [9].

Навіть якщо буде проводена величезна робота для виявлення фішингових листів, не існує набору функцій, які були б визначені як найкращі для виявлення фішингу. Постійно є необхідність продовжувати підвищувати точність методів виявлення [2].

Цей тип атак викликав найбільшу заклопотаність в останньому звіті про злочини в Інтернеті за 2018 рік, опублікованому Центром скарг на злочини в Інтернеті (IC3) Федерального бюро розслідувань США. Статистика, зібрана IC3 ФБР за 2018 рік, показала, що крадіжка, шахрайство і експлуатація через Інтернет залишаються широко поширеними і привели до приголомшуючих фінансових втрат в 2,7 мільярда доларів в 2018 році. Того року IC3 отримав 20 373 скарги на компрометацію корпоративної електронної пошти і злом облікового запису електронної пошти з втратами понад 1,2 мільярда доларів. Робоча група з протидії фішингу (APWG) наголошує, що за останні роки фішингові атаки зросли. Це число поступово зростає із 162 155 в останньому кварталі 2019 року до 165 772 випадків у першому кварталі 2020 року. Фішинг завдав серйозних збитків багатьом організаціям та світовій економіці, у четвертому кварталі 2019 року, виявив член APWG OpSec Security що сайти SaaS та веб-пошти залишаються найчастішими об'єктами фішингових атак. Фішери продовжують збирати дані від цих цілей, працюючи з BEC, і згодом отримують доступ до корпоративних рахунків SaaS.

Фішингові інциденти поширені з вагомих причин. По-перше, цей тип атак зазвичай дешевий у виконанні в порівнянні з іншими атаками, які обходять системи захисту периметра, такі як міжмережеві екрани, IDS/IPS та інші. Зловмисники іноді використовують програмне забезпечення, але в них немає цілі закріпитися в цільовій системі [5]. По-друге, процедура атаки може повторюватися кілька разів і на декількох користувачів системи. Це значно

збільшує ймовірність того, що будь-якої співробітник в організації буде обдурений. По-третє, оскільки електронна пошта часто використовується для відправки запитів, велика частина користувачів схильна довіряти запитам, що відправляється по електронній пошті, і фактично намагається виконати запитані дії. Провівши велику кількість тестів компанія «Cofence» показує наскільки сприйнятливі в цілому. В даному тесті було надіслано 135 мільйонів підроблених фішингових листів, і 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 фішингу може значно варіюватися в залежності від одержувача, ситуації і фішингових контенту (наприклад, повідомлення). 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 2018 року, а в другому кварталі збільшився до 18,4%.

1.2 Етапи проведення атаки

Загалом, фішинг атака виконуються за наступні п’ять кроків:

• Зловмисники створюють підроблені веб-сайти, які виглядають точно так само, як і існуючі, включаючи налаштування веб-сервера, застосування імені DNS-сервера та створення вебсторінок, подібних до веб-сайту призначення.

• Потім фішери надсилають велику кількість підроблених електронних листів використовуючи ілюзію, засновану на зовнішньому вигляді електронного листа (наприклад, структура адреси електронної пошти, заголовок теми і зміст), повинна здаватися більш законною за рахунок використання логотипів установи, термінології і т.д. Для створення автентичності в свідомості жертви. потенційним жертвам від імені законних цільових компаній та організацій, які намагаються переконати користувачів виконати дії, наведені нижче.

• Ввести їх особисту інформацію, таку як ім’я користувача, пароль, номер соціального страхування, номер банківського рахунку тощо за допомогою Форми HTML, яка вбудована в тіло повідомлення або як вкладення. Спонукання до цих дій здійснюється за допомогою фальсифікації, а саме щоб привернути увагу жертв, імовірно в їх інтересах, про те, що проблема існує, наприклад облікові записи клієнтів були викрадені. Електронний лист також може сприйматися як дружнє, наприклад дякуємо вам за співпрацю. Використовуючи зворотний соціальну інженерію, перш ніж проблема буде вирішена, мета відчуває себе в боргу перед зловмисником.

• Користувач спокушається фальшивим попередженням і спокушається клацнути гіперпосилання, яке зазвичай замасковане під текст або зображення, наприклад: «Клацніть тут для перевірки». Тон, разом зі зворотною психологією, використовується, наприклад, для того щоб користувач побоювався, що якщо він вирішить ігнорувати повідомлення, це призведе до автоматичного видалення його облікового запису. За іронією долі, «людська природа» - це не бажати подальших

небажаних наслідків або неприємностей, наприклад поповнення рахунків і т. д.

• Користувачі, яких вийшло переконати, розголошують свою особисту інформацію або через електронну пошту або відвідавши веб-сайт та подавши необхідну інформацію.

• Зловмисники збирають особисту інформацію та здійснюють шахрайства різного роду, такі як переказ грошей з банківського рахунку жертви, надсилання електронної пошти про спам та фішинг використовуючи електронну пошту та акаунти соціальних мереж жертв, викрадення особистих даних та подання заявок для кредитних карток та різних видів позик тощо.

1.3 Техніки фішингу

1) Маніпуляція з посиланнями

Фішинг в основному стосується посилань. Є кілька розумних способів маніпулювати URL-адресою, щоб він виглядав як законний. Один із способів - уявити шкідливі URL-адреси у вигляді гіперпосилань з ім'ям на веб-сайтах. Інший метод - використовувати URL-адреси з помилками, які будуть виглядати як допустимі URL-адреси, наприклад ghoogle.com. Варіант тайппсквоттінга, який набагато важче розпізнати в порівнянні зі згаданими методами маніпулювання посиланнями, називається Спуфінга IDN, при якому зловмисники використовують символ неанглийского мови, який виглядає точно так само, як англійська символ, наприклад, використовуючи кирилицю "с" або "а" замість англійських аналогів.

2) Підробка веб-сайту

У цьому типі атаки фішинг відбувається на законному веб-сайті шляхом маніпулювання кодом JavaScript цільового веб-сайту. Ці типи атак, які також

відомі як XSS, дуже важко виявити, оскільки жертва використовує законний веб-сайт.

3) Приховане перенаправлення

Ця атака націлена на веб-сайти, які використовують протокол OAuth 2.0 і OpenID. Намагаючись надати для токена доступ до легітимного веб-сайту, користувачі передають свій токен шкідливої службі. Однак особливої уваги цей метод не отримав через його низьку значущості

4) Соціальна інженерія

Цей вид фішингу здійснюється за допомогою соціальної взаємодії. Він використовує психологічні прийоми, щоб обдурити користувачів, щоб вони видавали інформацію безпеки. Цей тип атаки відбувається в кілька етапів. Спочатку Фішер досліджує потенційні слабкі місця цілей, необхідних для атаки. Потім Фішер намагається завоювати довіру мети і, нарешті, створити ситуацію, в якій мета розкриває важливу інформацію. Існують деякі методи фішингу з використанням соціальної інженерії, а саме цькування, напугіваніе, предтекст і цільової фішинг.

1.4 Сприйнятливість фішинг-листів

Оцінки компанії «Cofense» показали, що існує значний розкид в уразливості до фішингу, наприклад з рівнем сприйнятливості в одному кварталі більш ніж в чотири рази вище, ніж в іншому. Як заявляє компанія щодо варіації в їх даних: «Відсотки можуть залежати від багатьох факторів, таких як кількість симуляцій, ким і як часто. Як видно з даних про фішинг, час також може бути важливим фактором, оскільки користувачі менш уважні в періоди зайнятості.

На вразливість до фішингу можуть впливати:

1) атрибути повідомлення

2) характер одержувача

3) ситуація, в якій перебуває отримувач.

цей простий список факторів не претендує на те, щоб бути всеосяжною моделлю прогнозування фішингу, а тільки служить способом класифікації змінних, які така модель може захотіти включити. Наведений нижче текст покликаний навести приклади того, як ці три фактори можуть впливати на сприйнятливість до фішингу, з деякими вказівками на дослідження обману і переконання в цілому, а також конкретні приклади з досліджень фішингу.

Інтуїтивно зрозуміло, що зміст і формулювання фішингових повідомлення впливають на ймовірність того, що одержувач виконає запитану дію. Дослідження як обману, так і переконання підтверджують цю думку, а дослідження, присвячені фішингу, підтверджують деякі з цих більш загальних теорій. Однак деякі результати також суперечать загальним теоріям обману і переконання. Приклади підтвердження і спростування встановлених теорій наведені нижче.

Відповідно до одної теорії, люди які мають за мету обманути інших. менш відкриті (наприклад, відповідають менш докладно і, здається, стримуються), більш напружені, менш позитивні / приємні, мають менше звичайних недоліків і незвичайного змісту в своїх оповіданнях і мають менш переконливі розповіді (наприклад,, мати менш захоплюючі і вільні розповіді). Деякі дані свідчать про те, що одержувачі електронної пошти шукають такі сигнали і з більшою ймовірністю будуть обмануті, якщо сигнали відсутні (що може мати місце, якщо у зловмисника було більше часу, щоб спланувати обман). Наприклад, [21] виявив, що одержувачі були більш сприйнятливі до електронних листів з вмістом, яке було знайомим і гумористичним, тобто більш приємним. Другий приклад релевантної теорії, пов'язаної з повідомленням, можна витягти з маркетингових досліджень. Дослідження реклами показали, що використання авторитету є ефективним принципом переконання. Відповідно, в справжніх фішингових листах поле теми зазвичай використовується для позначення авторитетних джерел.

Розумно очікувати, що одних людей легко обдурити, а інших - ні. Теорії і дослідження обману в цілому дійсно демонструють, що існують релевантні змінні, пов'язані з одержувачем, незалежно від того, чи є змінні більш стабільними (наприклад, особистість) або більш гнучкими (наприклад, знання). Такі змінні можуть свідчити про існування особливо вразливих людей, які потребують цілеспрямованих зусилль щодо зниження їх вразливості. Податливі змінні також можуть бути пов'язані з уразливістю до фішингу. Наприклад, лабораторні експерименти з обманом в цілому показали, що хороший настрій робить людей більш легковірними, ніж поганий чи нейтральний настрій [29]. Теорія міжособистісного обману також охоплює податливі змінні. Зокрема, стверджується, що навички одержувачів по виявленню помилкових повідомлень розрізняються. Однак емпіричні докази цього неоднозначні, і навіть якщо є більш кваліфіковані фахівці, неясно, як їх знайти. Наприклад, в загальному дослідженні особистого обману, експерти (наприклад, співробітники правоохоронних органів і аудитори) показали, що вони не більше точні у виявленні брехні, ніж новачки [30]. З іншого боку, лабораторні фішингові експерименти показали позитивну кореляцію між точністю виявлення і самооцінкою спроможності справлятися з фішингом (r = 0,16), обізнаністю про значок замка в браузерах (r = 0,18) і інтелектом (поєднує нові рішення проблем і досвід) (r = 0,27).

Висновки до розділу 1

У цьому розіділі було розглянуто досить важливу загрозу під назвою «фішинг». Як показує статистика досить велика частина користувачів не розрізняє фішинг-листи від легітимних. З кожним роком все більше компаній потрапляють в не приємні ситуації із-за дій зловмисників. Розглянуті етапи проведення даної атаки і її техніки, допомагають зрозуміти як працюють шахраї, і як саме потрібно захищати себе від них. Аналіз атаки і усвідомлення всієї

небезпеки не гарантує падальшого повного захисту від фішингу, велику роль відіграє не тільки дотримування правил кібергігієни, а й «людський фактор». На жаль, попри всі застереження, людина переходить за посиланням, усвідомлено ризикуючи.

2.МЕТОДИ РОЗПІЗНАВАННЯ ФІШИНГУ

2.1 Ознаки фішингу

Більшість фішингових атак відбувається через шахрайство електронною поштою, а отже, зловмисникам потрібен спосіб, яким вони можуть заманити користувачів, створивши фішинговий веб-сайт, схожий на існуючий доброякісний веб-сайт. Зловмисники зазвичай використовують методи обфускації, щоб обдурити користувача при натисканні на фішинг-URL[3]. Замінюючи ім’я хоста IP-адресою або включаючи доменне ім’я цільового бренду в шлях до URL-адреси з помилками орфографії чи без них.

Сторінки фішингу, як правило, мають кілька переспрямувань для перенаправлення з початкової URL-адреси на кінцевий сайт який розміщується зловмисником на будь-якому скомпрометованому сервері. Зловмисники також заражають доброякісні веб-сайти прихованим шкідливим кодом JavaScript, який вбудовує «iframe» з URL-адреса домену зловмисника, а потім видає перенаправлення HTTP 302 для завантаження фішинг веб-сайту.[2]

На сьогоднішній день майже всі отримувані електронні листи базуються на мові розмітки HTML. Теги HTML використовуються фішерами для візуального обману користувачів. Наприклад, гіперпосилання є активними та натискаючими, маскуючи фактичну URL-адресу [3]. 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 кілька методів, таких як приховані елементи, спливаючі вікна і підказки для введення конфіденційної інформації, щоб спонукати користувачів ввести паролі і конфіденційну інформацію [2]. 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.

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 [2]. 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 зламані домени 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.

Функція виявлення фішингу або набір функцій є надійними, якщо або зловмисник не може легко створити фішинговий веб-сайт, для якого функції виглядають як функції нешкідливого веб-сайту, або якщо зловмисник вимагає великих витрат на створення веб-сайту, який з високою ймовірністю не відрізняється від доброякісного сайту[2]. Більшість функцій URL-адрес, таких як кількість крапок в URL-адресі, довжина URL-адреси і т.д., можуть бути змінені зловмисником, щоб вони виглядали як безпечний веб-сайт, і тому ці функції не є надійними окремо. Але в певних випадках, наприклад, коли зловмисник використовує довгі URL-адреси для обману користувачів, ці функції, якщо їх розглядати разом, можуть бути більш надійними, ніж окремі

функції взяті поодинці. Аналогічним чином, з огляду на функції DNS, власник веб-сайту має доступ для зміни поштового сервера, сервера імен і записів PTR для свого веб-сайту, і, отже, ці функції не є надійними. Зловмисники, які розміщують і володіють своїми фішинговими веб-сайтами, можуть змінити ці функції так, щоб вони виглядали як функції безпечного сайту. Таким чином, обговорення надійності різних категорій ознак необхідно, щоб прийти до набору надійних функцій, які можна ефективно використовувати для виявлення фішингових веб-сайтів та електронних листів.

2.2 Класифікація електронних листів

В даній дипломній роботі було реалізовано прикладну програму, яка класифікує вхідну електронну пошту. Кожен вхідний лист проходить перевірку за ознаками фішингу і при виявленні цих ознак сповіщає користувача про небезпеку. Нижче перечислено ознаки які шукаються в електронних листах:

1) HTML листи**:** Теги HTML, такі як: “form” та “script”, використовуються фішерами для візуального обману користувачів. Наприклад, гіперпосилання є активними та натискаючими, маскуючи фактичну URL-адресу [3]. HTML-форми - це один з методів збору інформації від користувачів. Замість того, щоб просити користувачів клацнути посилання, щоб перейти на підроблену веб-сторінку, контрольовану фішерами, користувачам може бути надана форма для заповнення і відправки з їх особистою інформацією. Прикладна програма перевіряє, чи є в електронному листі тег FORM. JavaScript зазвичай використовується в фішингових листах, оскільки він дозволяє обдурити на стороні клієнта, використовуючи скрипти, щоб приховати інформацію або активувати зміни в браузері. Якщо в електронному листі було виявлено тег «Script», ми відзначаємо його як фішинговий.

2) IP-адрес в посиланні: Замість того, щоб купувати та реєструвати доменне ім’я, фішери зазвичай використовують бот-мережі або компрометовані системи для розміщення фішингових веб-сайтів. Використання IP-адреси для обфускації доменного імені - загальна тактика, оскільки забезпечує його простіший та дешевший спосіб розміщення фішингових веб-сайтів. Легітимний веб-сайт зазвичай має доменне ім’я для його ідентифікації. Отже, якщо електронне повідомлення містить посилання, хост якого є IP-адресою (наприклад, http://81.215.214.238/bankofamerica/), ми відзначаємо це як фішинг.[3]

3) Кількість доменів в посиланні: Ми виділяємо та підраховуємо кількість доменів у URL-адресі починаючи з http:// або https://. Два або більше доменних імен використовуються в URL або для обману користувачів, або щоб перенаправити їх на інший домен. Наприклад, URL-адреса «http://www.google.com/url?sa=t&ct=res&cd=3&url=http%3A%2F%2 Fwww.antiphishing.org%2F&ei=-0qHRbWHK4z6oQLTmBM&usg=uIZX_3aJvESkMveh4uItI5DDUzM=&sig2=AVrQFpFvihFnLjpnGHVsx» має два доменних імені, де google.com пересилає користувачів на домен antihphishing.org [3].

4) Вік домену: Фішингові веб-сайти зазвичай мають короткий термін життя, 3-4 дні в середньому. Таким чином, ми можемо використовувати цю ознаку для позначення електронних листів як фішингові виходячи з того, що домен був нещодавно зареєстровано (менше 30 днів) [3].

5) Ключові слова: фішингові електронні листи можуть містити ряд часто використовуваних ключових слів, таких як “suspend”, “verify”, “username” і т.д. Ми використовуємо частоту слів (кількість ключових слів, поділена на загальну кількість слів у електронному листі) як ознаку фішингу.

6) Чорний список: За допомогою сервісу “CheckPhish”, URL-адреса проходить перевірку на перебування в чорному списку адрес.

7) Сервіси скорочення URL: Досить популярними стали сервіси для скорочення URL-адреси, і це стало нагодою для зловмисників. На сьогоднішній день існує досить велика кількість таких сервісів, тому наявність такого посилання в електронному листі позначається як «підозра на фішинг». Наприклад, URL-адреса http://sharif.hud.ac.uk/ може бути скорочена до bit.ly/1sSEGTB.

8) Наявність символу «@» в URL: використання символу «@» в URL-адресі призводить до того, що браузер ігнорує все, що передує символу «@», і справжня адреса часто йде за символом «@».

9) Порт в URL: Наявність порту в URL-адресі не заслуговує довіри. Наприклад посилання: «https://mysite.com:37654» буде маркеровано як підозріле.

10) Кількість крапок в домені: Якщо хостова частина URL-адреси має 5 крапок або більше, чи довжина адреси перевищує 75 символів, чи сама довжина хостової частини перевищує 30 символів це вказує на ознаки присутності фішингу.

2.3 Застосування машинного навчання

Другим методом розпізнаванням фішинг-листів в даній дипломній роботі реалізоване машинне навчання. Машинне навчання це досить потужний інструмент коли стоїть питання передбачення якогось стану. Машинне навчання забезпечує спрощені та ефективні методи для аналізу даних. Це вказує на перспективні результати щодо проблем класифікації в режимі реального часу. Ключовою перевагою машинного навчання є можливість створювати гнучкі моделі для конкретних завдань, таких як виявлення фішингу. Оскільки фішинг - це класифікаційна проблема, машинне навчання добре її вирішує. Моделі машинного навчання можуть адаптуватися, щоб виявити

закономірності шахрайських операцій які допомагають розробити систему ідентифікації на основі навчання.

Як правило, існує два основні методи машинного навчання: навчання з учителем і навчання без учителя. Велика частина машинного навчання, близько сімдесяти відсотків, насправді є навчанням з учителем. Ще десять-двадцять відсотків - це навчання без учителя. Існують і інші методи, такі як напів-контрольоване навчання і навчання з підкріпленням, але вони використовуються не так часто. В нашому випадку було використаний метод контрольованого навчання. Цей метод машинного навчання може бути реалізований тільки тоді, коли ми можемо ідентифікувати вхідні і вихідні дані і позначати дані для роботи. Алгоритм буде отримувати свої вхідні дані разом з відповідними вихідними даними для виявлення помилок. На основі вхідних даних модель буде адаптована. Фактично весь цей процес є розпізнаванням образів, і ці алгоритми завжди покладаються на такі методи, як регресія, класифікація, прогнозування або підвищення градієнта.

Багато задач машинного навчання, такі як аналіз прогнозів, вирішуються за допомогою лінійної регресії. Цей метод є потужним засобом прогнозування результату з високою точністю, якщо дані навчання цінні і не містять занадто багато шуму. Одна з характеристик лінійної регресії - обчислення значення прогнозу на основі незалежних змінних. Іншими словами, ми повинні вибирати цей алгоритм щоразу, коли вам потрібно обчислити зв'язок між рядом змінних.

Розглянемо детальніше використані в даній роботі алгоритми МН:

1) «Support Vector Machines»: Нам потрібен універсальний алгоритм, який може впоратися практично з усім, і в такому випадку гарним вибором буде SVM. SVM може обробляти як регресію, так і класифікацію і навіть виявлення викидів. Метод «Support Vector Machines» популярний, тому що він потужний, але не ресурсомісткий. Звичайно чим більше ми прагнемо до точності, тим повільніше буде йти обробка даних, тому що для цього буде потрібно більше від нашої комп'ютерної системи. Однак

SVM вимагає набагато менше витрат, не впливаючи на точність прогнозу. Крім того, ми можемо використати цей алгоритм, щоб також очистити більшу частину шуму в наборі даних, тим самим зменшивши кількість кроків, які вам потрібно виконати на етапі попередньої обробки. Алгоритм SVM важливий завдяки своїй потужності та універсальності, він застосовуються в самих різних областях. Наприклад, використо вуються для програмного забезпечення розпізнавання осіб, класифікації тексту, розпізнавання почерку, а також в медичній промисловості для класифікації генів. Цей алгоритм має набагато більше застосувань, оскільки він з великою ефективністю розділяє точки даних в наборі даних і дозволяє отримувати досить точні результати, щоб його можна було використовувати в складних додатках.

Плюси:

- Він дійсно добре працює з чітким поділом.
- Він ефективний у просторах великих розмірів.
- Це ефективно в тих випадках, коли кількість вимірювань більше, ніж кількість зразків.
- Він використовує підмножину навчальних точок в функції прийняття рішення (званих векторами підтримки), тому він також ефективний з точки зору пам'яті.

Мінуси:

- Він не працює добре, коли у нас великий набір даних, тому що потрібно більше часу на навчання.
- Він також не дуже добре працює, коли в наборі даних більше шуму, тобто цільові класи перекриваються.
- SVM не дає безпосередньо оцінок ймовірностей, вони розраховуються з використанням дорогої п'ятикратної перехресної перевірки. Він включений в пов'язаний метод SVC бібліотеки Python scikit-learn.

2) «Decision tree». Дерева рішень, як і машини опорних векторів, неймовірно універсальні, тому що вони можуть виконувати як операції регресії, так і класифікації. Це означає, що вони надзвичайно корисні при роботі з великими наборами даних. Крім того, 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, як випливає з назви, дерева рішень використовуються для моделювання рішень, включаючи визначення будь-яких наслідків і системних ресурсів, необхідних для всього процесу. Крім того, всі візуально представлено у вигляді графіка [11]. 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. 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. З урахуванням вищесказаного, давайте обговоримо всі інші переваги, що надаються деревами рішень:

    1. 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.[11]

2. Чисті дані. Незалежно від того, який алгоритм ми використовуєте, 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. [11]

3. 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. [11]

4. 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 вам часто доводиться мати справу з тисячами змінних, а пошук найбільш

важливих з них може бути утруднений без відповідного інструменту[11].

3) “K-Nearest Neighbors” (KNN) – це один з найпростіших алгоритмів, що використовується в машинному навчанні для регресії та класифікації, який є непараметричним і ледачим. У KNN немає потреби в припущенні про базовий розподіл даних. Алгоритм KNN використовує подібність функцій для прогнозування значень нових точок даних, що означає, що новій точці даних буде присвоєно значення на основі того, наскільки точно вона відповідає точкам у навчальному наборі. Подібність між записами можна виміряти різними способами. Після виявлення сусідів можна зробити підсумований прогноз, повернувши найпоширеніший результат або взявши середнє значення. KNN може бути використаний для проблем класифікації або регресії.[1] У випадку класифікації та регресії ми побачили, що вибір правильного K для наших даних здійснюється шляхом випробування декількох K і вибору того, який найкраще працює. Головний недолік KNN - збільшення обсягу даних сповільняє процес та робить його непрактичним вибором в середовищах, де прогнозування потрібно робити швидко. Більше того, існують більш швидкі алгоритми, які можуть дати точніші результати класифікації та регресії. 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.

Ми можемо зрозуміти роботу цього алгоритму за допомогою наступних кроків:

- Для реалізації будь-якого алгоритму нам потрібен набір даних. Таким чином, під час першого кроку KNN ми повинні завантажити дані навчання, а також дані випробувань.

- Далі нам потрібно вибрати значення K, тобто найближчі точки даних. K може бути будь-яким цілим числом.
- Для кожної точки в тестових даних потрібно зробити наступне:
    - Розрахувати відстань між даними випробувань і кожним рядком даних тренування з допомогою будь-якого з методів, а саме: Евклідово, Манхеттенські або Хеммінговскі відстані. Найбільш часто використовуваний метод розрахунку відстані - евклідів.
    - Тепер, грунтуючись на значенні відстані, відсортувати їх в порядку зростання.
    - Далі алгоритм вибере верхні K рядків з відсортованого масиву.
    - Тепер він призначить клас контрольній точці на основі найбільш часто зустрічається класу цих рядків.

4) “Random forest”. Випадковий ліс, як випливає з його назви, містить велику кількість окремих дерев рішень, які діють як група для прийняття рішення про вихід. Кожне дерево в випадковому лісі визначає передбачення класу, і результатом буде самий передбачуваний клас серед дерев рішень. Причина такого дивного результату від Random Forest в тому, що дерева захищають одне одного від індивідуальних помилок. Хоча деякі дерева можуть передбачити неправильну відповідь, багато інших дерева виправлять остаточний прогноз, щоб група дерев могла рухатися в правильному напрямку. Випадкові ліси дозволяють скоротити перенавчання за рахунок об'єднання багатьох слабких учнів, які недостатньо підходять, тому що вони використовують тільки підмножина всіх навчальних вибірок. Випадкові лісу можуть обробляти велику кількість змінних в наборі даних. Також в процесі будівництва лісу вони роблять об'єктивну оцінку помилки узагальнення. Крім того,

вони можуть добре оцінити втрачені дані. Основним недоліком випадкових лісів є відсутність відтворюваності, оскільки процес будівництва лісу носить випадковий характер. Крім того, остаточну модель і наступні результати складно інтерпретувати, оскільки вона включає в себе безліч незалежних дерев рішень [3]. Random Forest через незалежне побудове глибоких дерев вимагає дуже багато ресурсів, а обмеження на глибину зашкодить точності (для вирішення складних завдань потрібно побудувати багато глибоких дерев). Можна помітити, що час навчання дерев зростає приблизно лінійно їх кількості.

В задачі регресії відповіді окремих дерев усереднюються, а в завданні класифікації приймається рішення голосуванням за більшістю. Всі дерева будуються незалежно за наступною схемою:

- Вибирається підвибірка навчальної вибірки розміру samplesize - по ній будується дерево (для кожного дерева - своя підвибірка).
- Для побудови кожного розщеплення в дереві переглядаємо max_features випадкових ознак (для кожного нового розщеплення - свої випадкові ознаки).

Вибираємо найкращі ознаки і розщеплення по ньому (по заздалегідь заданому критерію). Дерево будується, як правило, до вичерпання вибірки (поки в листі не залишаться представники тільки одного класу), але в сучасних реалізаціях є параметри, які обмежують висоту дерева, число об'єктів в листі і число об'єктів в підвибірки, при якому проводиться розщеплення.

Висновок до розділу 2

Встановлено, що функція виявлення фішингу або набір функцій є надійними, якщо або зловмисник не може легко створити фішинговий лист, для якого функції виглядають як функції нешкідливого листа, або якщо зловмисник вимагає великих витрат на створення електронного листа, який з високою

ймовірністю не відрізняється від доброякісного листа. Тому перевірка ознак поодинці не є ефективним рішенням. Аналіз показав, що необхідно застосовувати двохетапну процедуру: на першому етапі проводити відбір, на другому застосовувати методи машинного навчання.

Правила використані в цій дипломній роботі можна вважати «міцними» тому, що в парі вони досить добре виявляють фішинг. Також алгоритми машинного навчання допомагають з гарною точністю класифікувати електронні листи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Постановка задачі та її реалізації

З метою практичного використання результатів отриманих вище проведемо розробку програми, та навчимо програму розпізнавати фішингові електронні листи. Для цього були застосовані разні алгоритми машинного навчання, а для навчання використовувались реальні фішингові листи які пройшли перевірку на ознаки фішингу.

Навчання програми проходить в два етапи:

1) Програмою було сформовано датасет з 2114 записів. Кожен з цих записів це електронний лист, легітимний чи фішинговий, і всі вони пройшли перевірку на наявність ознак фішингу. При кожній знайденій ознаці лист отримував оцінку ”1”, а якщо ознака не була знайдена ”-1”. Результати всіх перевірок були записані і використані як набір вхідних даних для машинного навчання. Для кожного фішинг листа в полі “Result“ записане значення ”1”, відповідно легітимним листам значення ”-1”. Для того, щоб побачити залежність ознак між собою, була проведена кореляція, результат можна побачити на Рисунку 3.1. Розглянувши дану діаграму можна зробити висновок, що є залежність між ознаками притаманними URL-адресу такими як: наявність IP адреси, кількість крапок в домені, наявність порту в посиланні та перенаправлення. Також можна побачити залежність між HTML тегами «script» та «form». Це доводить слова написані в 2 розділі про те, що поодинці ці правила не є «міцною» перевіркою на фішинг, але в купі вони дають гарний результат.

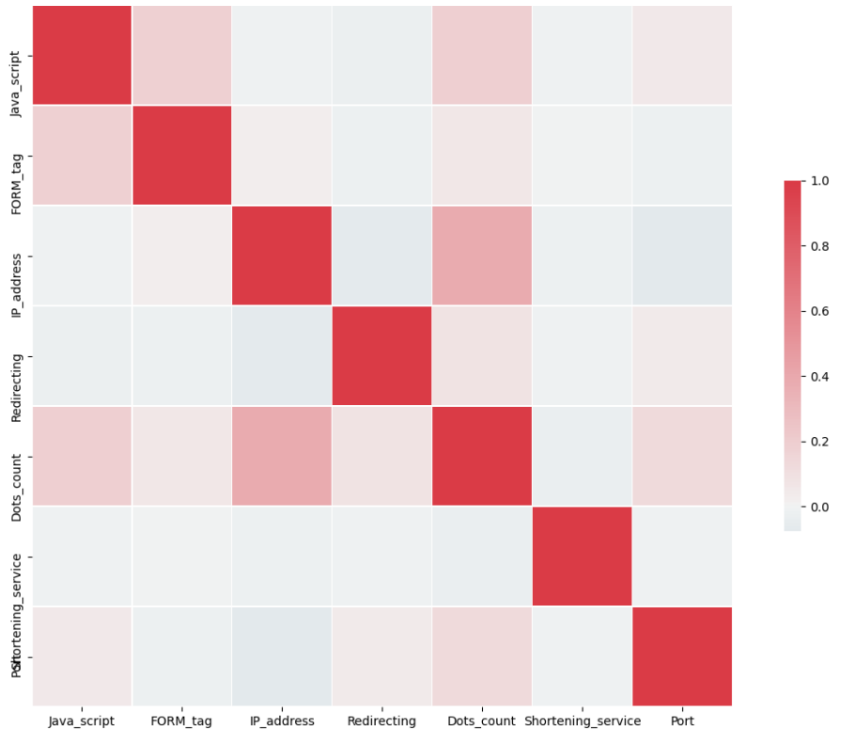


Рисунок 3.1 – Кореляція ознак у вхідних даних

Для прикладу на рисунку 3.2 продемонстровано частина функції оцінювання ознак фішингу в електронних листах, а саме наявність HTML тегів. В функцію яка шукає ознаки надходить декодований електронний лист, зазвичай в розмітці HTML, тому задопомогою бібліотеки «BeautifulSoup» проводиться парсинг електронного листа і далі перевірка на наявність необхідних тегів. Наприклад, якщо було виявленно тег «script», програма виводить про це повідомлення і записує «1» в змінну типу масив для подальшого формування датасета. Таким ж чином перевіряється наявність тегу «form».

```
soup = BeautifulSoup(decoded_message, 'html.parser')
# ----- 8) Presence of JavaScript + -----
script = soup.find("script")
if script:
    print(mail[url_count], '---> ', 'JavaScript in mail, seems like phishing')
    site_is["phishing"] += 1
    url_features.append(1)
else:
    url_features.append(-1)
# ----- 9) Presence of FORM tag + -----
form = soup.find("form")
if form:
    print(mail[url_count], '---> ', 'Form in mail, seems like phishing')
    site_is["phishing"] += 1
    url_features.append(1)
else:
    url_features.append(-1)
```

Рисунок 3.2 – Приклад пошуку ознак фішингу

2) Другим етапом є безпосередньо застосування алгоритмів машинного навчання. В даній роботі було реалізовано чотири алгоритми машинного навчання для класифікації і прогнозування фішингових листів. У 2 розділі були описані їхні плючи та мінуси, а у таблиці 1 можна побачити результати навчання цих алгоритмів. Ми оцінюємо та порівнємо такі параметри цих моделей як: accuracy, precision, recall, F1 score, train time та test time.

Таблиця 3.1 – Результати навчання алогоритмів МН.

	Decision tree	**KNN**	**SVM**	**Random forest**
Train time (s)	0.0027943	0.0071331	0.0394911	0.1457726
Test time (s)	0.0039834	0.0124230	0.0070773	0.0147052
Accuracy	0.7890235	0.6718054	0.7928194	0.7885518
Recall	0.7027436	0.8664247	0.7074864	0.7027436
Precision	0.9366093	0.7300309	0.9377611	0.9354189
F1 score	0.7993062	0.7658823	0.8021308	0.7989316

Як евристика або емпіричне правило, точність (accuracy) може негайно сказати нам, чи правильно тренується модель і як вона може працювати в цілому. Однак ця міра не дає детальної інформації щодо її застосування до проблеми. Проблема використання точності як основної метрики ефективності полягає в тому, що вона погано спрацьовує, коли у вас є серйозний дисбаланс класу [13].

Влучність (precision) - це кількість справжніх позитивів, поділена на кількість справжніх і помилкових позитивів. Іншими словами, це кількість позитивних прогнозів, поділена на загальну кількість передбачених позитивних значень класу. Його також називають позитивною прогнозною цінністю. Точність можна сприймати як міру точності класифікаторів. Низька влучність також може свідчити про велику кількість помилкових спрацьовувань [12].

Recall - це кількість справжніх позитивів, поділена на кількість справжніх позитивів та кількість помилкових негативів. Іншими словами, це кількість позитивних прогнозів, поділена на кількість позитивних значень класу в даних тесту. Його також називають чутливістю або справжнім позитивним коефіцієнтом. «Згадування» можна сприймати як міру повноти класифікаторів. Низький рівень свідчить про багато помилкових заперечень [12].

F1 - загальний показник точності моделі, який поєднує в собі точність і пригадування, дивним чином, що додавання та множення просто змішують два інгредієнти, щоб зробити окрему страву. Тобто, хороший показник F1 означає, що у нас низький рівень хибних спрацьовувань та низький рівень помилкових негативів, тому ми правильно визначаємо реальні загрози і нас не турбують помилкові тривоги. Оцінка F1 вважається ідеальною, коли вона дорівнює 1, тоді як модель – повністю помилково класифікує, міра буде дорівнювати 0 [13].

3.2 Функція пошуку ознак фішингу

Зпершу варто розібрати, що стоїть на вході в функцію пошуку фішингових ознак. HTML тіло електронного повідомлення передається в цю функцію, для пошуку HTML тегів. При формуванні датасету для навчання алгоритмів МН використовувались готова збірка листів в HTML формі, але програма також може в реальному часі перевіряти електронні листи які надійшли на поштовий сервіс «Gmail». Щоб зробити перевірку ознак такого листа потрібно спочатку його декодувати, тому в залежності від завдання, лист спочатку може пройти процес декодування.

Після декодування лист проходить ще 2 функції. Перша – пошук всіх URL-адрес присутніх в ньому, а друга – парсинг «білого» тексту повідомлення. Тому другим аргументом на вході в функцію пошуку ознак фішингу є масив зі всіма посиланнями та самим повідомленням з електронного листа.

Наступним етапом є перевірка посилань на ознаки, пошук ключових слів та HTML тегів. Для перевірки на наявність ключових слів було обрано 8 слів які часто зустрічаються в фішингових листах, а саме: suspend, verify, re-verify, signin, login, confirm, purchase, payment. Електронний лист в якому було виявленно ці ключові слова і посилання з застосуванням сервіса вкорочення URL-адрес, можна рахувати фішинговим. Для пошуку наявності сервісів вкорочення було використано велику кількість відомих доменів цих сервісів.

```
# ----- 12) shortener URLs + -----
try:
    shortener_services = re.match(
        'bit\.ly/goo\.gl/shorte\.st/go2l\.ink/x\.co/ow\.ly/t\.co/tinyurl/tr\.im/is\.gd/cli\.gs/'
        'yfrog\.com/migre\.me/ff\.im/tiny\.cc/url4\.eu/twit\.ac/su\.pr/twurl\.nl/snipurl\.com/'
        'short\.to/BudURL\.com/ping\.fm/post\.ly/Just\.as/bkite\.com/snipr\.com/fic\.kr/loopt\.us/'
        'doiop\.com/short\.ie/kl\.am/wp\.me/rubyurl\.com/om\.ly/to\.ly/bit\.do/t\.co/lnkd\.in/'
        'db\.tt/qr\.ae/adf\.ly/goo\.gl/bitly\.com/cur\.lv/tinyurl\.com/ow\.ly/bit\.ly/ity\.im/'
        'q\.gs/is\.gd/po\.st/bc\.vc/twitthis\.com/u\.to/j\.mp/buzurl\.com/cutt\.us/u\.bb/yourls\.org/'
        'x\.co/prettylinkpro\.com/scrnch\.me/filoops\.info/vzturl\.com/qr\.net/1url\.com/tweez\.me/v\.gd/tr\.im/link\.zip\.net',
        domain[0])
    if shortener_services:
        site_is["suspicious"] += 1
        url_features.append(1)
    else:
        url_features.append(-1)
except:
    url_features.append(-1)
```

Рисунок 3.3 – Фрагмент коду пошуку сервісів вкорочення.

При перевірці на наявність перенаправлення в URL-адресі шукається запит: «?sa=t&ct=res&cd=3&url=https?» що свідчить про перенаправлення на інший домен. Також здійснюється пошук символу «@», адже при наявності такого символу в URL-адресі, все що стоїть зліва – ігнорується.

```
# ----- Number of domains(redirecting) + -----
redirect_in_url = re.findall(r"(?:url\?sa=t&ct=res&cd=3&url=https?)", mail[url_count])
dog_in_url = re.findall(r"@", mail[url_count])
if redirect_in_url or dog_in_url:
    print(mail[url_count], '---> ', 'redirecting in URL, seems like phishing')
    site_is["phishing"] += 1
    result_list.append((mail[url_count], site_is))
    url_features.append(1)
else:
    url_features.append(-1)
```

Рисунок 3.4 – Фрагмент коду пошуку перенаправлення

Фішингові веб-сайти зазвичай мають короткий термін життя, 3-4 дні в середньому. Таким чином, ми можемо використовувати цю ознаку для

позначення електронних листів як фішингові виходячи з того, що домен був нещодавно зареєстровано (менше 30 днів).[3]

```
# ----- 5) Age of domain (whoisxmlapi) + -----
payload = {
    'outputFormat': 'JSON'
}
domain_age = requests.get(f'https://www.whoisxmlapi.com/whoisserver/WhoisService?apiKey={whois_key}&domainName={mail[url_count]}', params=payload)
domain_json = domain_age.json()

age = datetime.datetime.strptime(domain_json["WhoisRecord"]["registryData"]["createdDate"][:10], '%Y-%m-%d')
today_date = datetime.datetime.strptime(str(datetime.date.today()), '%Y-%m-%d')
registred = str(today_date - age).split()
if int(registred[0]) < 31:
    url_features.append(1)
else:
    url_features.append(-1)
```

Рисунок 3.5 – Фрагмент коду застосування сервісу «WHOIS»

3.3 **Реалізація машинного навчання**

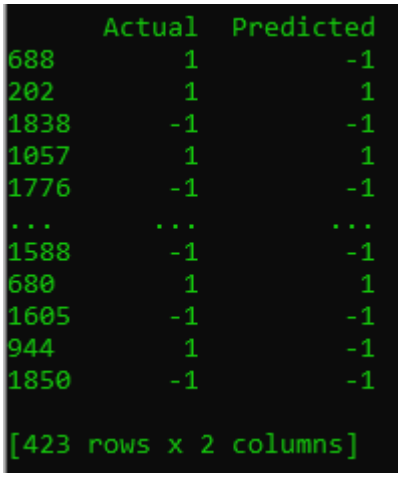
Після того як електронні листи пройшли перевірку і сформували датасет для машинного навчання, було використано 4 алгоритми, а саме: SVM, KNN, Decision tree і Random forest. Всі дані згідно до стандарту були поділені в співвідношені 80:20, де 80% це дані для навчання, а 20% для прогнозування та класифікації.

```
dataset = pd.read_csv('phish_set.csv')

X = dataset.drop('Result', axis=1)
y = dataset['Result']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

Рисунок 3.6 – Розмежування даних для МН

Для застосування алгоритмів машинного навчання була використана бібліотека «sklearn» на мові Python.

Першим буде розглянуто алгоритм «Support vector machine», однак спочатку нам потрібно імпортувати його з бібліотеки Scikit-learn, і після цього все, що нам потрібно зробити зараз, - це побачити, наскільки точним є прогноз (таблиця 3.1). Як видно з цих цифр, машина підтримки векторів спрацювала досить добре, оскільки нам вдалося отримати значення точності 0,79 без застосування будь-яких інших алгоритмів або методів попередньої обробки даних. Ось чому машини з підтримкою векторів використовуються у багатьох технічних галузях, що вимагають точності. Після використання функції прогнозування можна побачити результат на датафреймі з 10 випадкових прогнозованих листів.



	Actual	Predicted
688	1	-1
202	1	1
1838	-1	-1
1057	1	1
1776	-1	-1
...	...	...
1588	-1	-1
680	1	1
1605	-1	-1
944	1	-1
1850	-1	-1

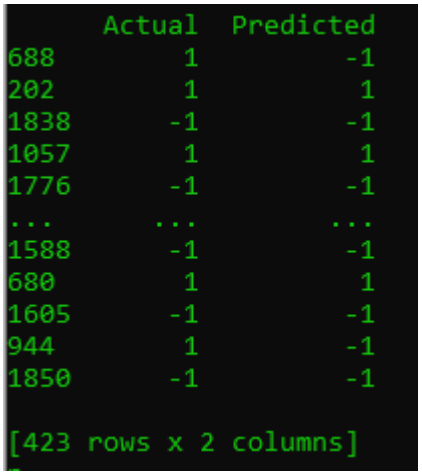
[423 rows x 2 columns]

Рисунок 3.7 – Результат прогнозування SVM

На рисунку 3.7 можна побачити, що з 10 випадків лише 1 був зпрогнозований не вірно.

Наступним розглянемо алгоритм «K-Nearest Neighbors». В таблиці 3.1 результат точності складає 0.67, що не зовсім гарний показник, значення K вказує кількість найближчих сусідів. Ми повинні обчислити відстань між тестовими точками та навченими мітками. Оновлення метрик відстані з кожною ітерацією обчислювально дорого, і тому KNN є лінивим алгоритмом навчання.

Методом перебору було встановлено значення K = 1, при якому це найкращий результат.



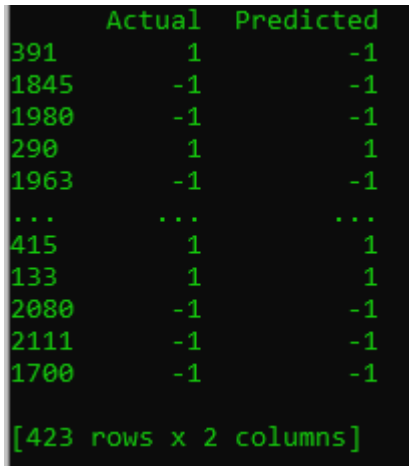
	Actual	Predicted
688	1	-1
202	1	1
1838	-1	-1
1057	1	1
1776	-1	-1
...	...	...
1588	-1	-1
680	1	1
1605	-1	-1
944	1	-1
1850	-1	-1

[423 rows x 2 columns]

Рисунок 3.8 – Результат прогнозування KNN

Як видно на рисунку вище, KNN гірше справився зі своєю задачею в порівняні з SVM, з 10 рішень – 2 помилкових.

Третім буде розглянуто алгоритм «Decision tree», згідно результатам з таблиці 3.1, точність дорівнює 0.78 і виходячи з цього і інших показників, можна вважати що алгоритм досить добре справився зі своєю задачею.



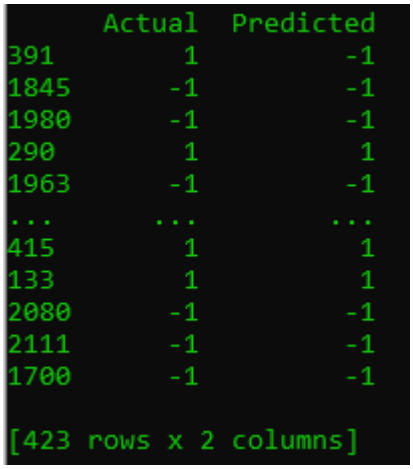
	Actual	Predicted
391	1	-1
1845	-1	-1
1980	-1	-1
290	1	1
1963	-1	-1
...	...	...
415	1	1
133	1	1
2080	-1	-1
2111	-1	-1
1700	-1	-1

[423 rows x 2 columns]

Рисунок 3.9 – Результат прогнозування Decision tree

Як видно з результату, з 10 передбачувань лише 1 було зробено не вірно. Цей алгоритм досить добре справляється з задачею класифікації.

Останнім використаним алгоритмом в даній роботі є «Random forest». Судячи з результатів цей алгоритм справився з задачею на рівні попередньо алгоритму, тобто досить вдало. Як і у алгоритма KNN, Random forest має змінюваний параметр який впливає на точність. Цей параметр визначає кількість дерев які будуть побудовані для класифікації. Але при тестуванні різної кількості дерев, а саме 100-500, не було виявленно змін в показнику точності.



	Actual	Predicted
391	1	-1
1845	-1	-1
1980	-1	-1
290	1	1
1963	-1	-1
...	...	...
415	1	1
133	1	1
2080	-1	-1
2111	-1	-1
1700	-1	-1

[423 rows x 2 columns]

Рисунок 3.10 – Результат прогнозування Random forest

Результат прогнозування також показав 9 з 10 правильних рішень, тому експеремент можна вважати вдалим

Висновок до розділу 3

Розроблена функція пошуку ознак на практиці досить добре знаходить задані характеристики фішингу саме в шахрайських листах. Порівнюючи кількість знайдених ознак в фішингових та легітимних листах це добре видно. Перш ніж формувати датасет, ці правила пошуку ознак було протестовано на підроблених листах, щоб перевірити працездатність.

Застосування машинного навчання описаного в цьому розділі демонструє, що алгоритми досить точно передбачують фішингові листи. Для навчання програми було розподілено всі листи в співвідношені 80:20, що забезпечує такий результат.

ВИСНОВКИ

Протягом цієї роботи були розглянуті основні методи розпізнавання фішингу в цілому та саме в електронних листах. Також була розглянута актуальність проблеми в кіберпросторі і загрозу яку несе дана атака. Після аналізу статистики, можна сказати, що людський фактор має вагомий вплив на результат атаки. Тобто в випадках коли користувач досвідчений і може розпізнати фішинг, є ймовірність, що він все таки виконає дії які потрібні зловмисникам для отримання власної вигоди, саме з цікавості.

Розроблена прикладна програма навчається розпізнавати фішинг за притаманними йому ознаками, щоб в подальшому знизити ризик бути ошуканим злочинцями для недосвідчиних користувачів.

В результаті роботи прикладної програми був проведений пошук ознак фішингу в легітимних та шахрайських листах, що дозволило сформувати датасет і навчати алгоритми машинного навчання класифікувати електронні листи. Три з чотирьох алгоритмів показали досить не погану точність при класифікуванні, і при прогнозуванні.

В результаті чого можна розпізнавати фішинг при надходженні нових листів, що допоможе користувачам не буду ошуканими.

Також отримані результати можуть бути проаналізовані, щоб сформувати перелік ознак які найчастіше використовують зловмисники при формуванні електронних листів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. 1. E.D. Frauenstein and R. von Solms , “An Enterprise Anti-phishing Framework”, pp. 1-9, March 2011.
2. Bhuvana Namasivayam, “Categorization of Phishing Detection Features And Using the Feature Vectors to Classify Phishing Websites”, pp. 1-52, 2017
3. Xun Dong, “Defending Against Phishing Attacks”, pp. 1-218, Sep. 2009
4. Ram Bahadur Basnet, “Detecting Phishing At1tacks: A Comprehensive Approach”, pp. 1-132, Dec. 2011
5. Dhruvalkumar Patel, “Test Utility for Live and Online Testing of an Anti-Phishing Message Security System”, pp. 1-41, 2014
6. Sa’id Abdullah Al-Saaidah, “Detecting Phishing Emails Using Machine Learning Techniques”, pp. 1-72, Jan. 2017
7. Vahid Shahrivari, Mohammad Mahdi Darabi and Mohammad Izadi, “Phishing Detection Using Machine Learning Techniques”, pp. 1-9, Sep. 2020
8. Teodor Sommestad and Henrik Karlzén, “A meta-analysis of field experiments on phishing susceptibility”, pp. 1-14, 2020
9. R. T. Wright, M. L. Jensen, J. B. Thatcher, M. Dinger, and K. Marett, “Research Note —Influence Techniques in Phishing Attacks: An Examination of Vulnerability and Resistance,” Inf. Syst. Res., vol. 25, no. 2, pp. 385–400, Jun. 2014.
10. J. P. Forgas and R. East, “On being happy and gullible: Mood effects on skepticism and the detection of deception,” J. Exp. Soc. Psychol., vol. 44, no. 5, pp. 1362–1367, 2008.
11. Moore R. Python Machine Learning / Richard Moore., 2019. – 48 с.

12. Koo P. S. Accuracy, Precision, Recall or F1? [Електронний ресурс] / Ping Shung Koo. – 2018. – Режим доступу до ресурсу: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9.

13. Nicholson C. Evaluation Metrics for Machine Learning [Електронний ресурс] / Chris Nicholson. – 2019. – Режим доступу до ресурсу: https://wiki.pathmind.com/accuracy-precision-recall-f1.

14. SUNIL R. Understanding Support Vector Machine [Електронний ресурс] / RAY SUNIL. – 2017. – Режим доступу до ресурсу: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

15. Yiu T. Understanding Random Forest [Електронний ресурс] / Tony Yiu. – 2019. – Режим доступу до ресурсу: https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

16. Eijaz A. Building a k-Nearest-Neighbors (k-NN) Model with Scikit-learn [Електронний ресурс] / Allibhai Eijaz. – 2018. – Режим доступу до ресурсу: https://towardsdatascience.com/building-a-k-nearest-neighbors-k-nn-model-with-scikit-learn-51209555453a.

17. Band A. How to find the optimal value of K in KNN? [Електронний ресурс] / Amey Band. – 2020. – Режим доступу до ресурсу: https://towardsdatascience.com/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb.

18. Chaudhary S. Recognition of phishing attacks utilizing anomalies in phishing websites / Sunil Chaudhary., 2012. – 93 с.

19. Hanaa A. Can Phishing Education Enable Users To Recognize Phishing Attacks? / Al-ghamdi Hanaa., 2017. – 86 с.

20. Fincher M. Phishing Dark Waters: The Offensive and Defensive Sides of Malicious Emails / M. Fincher, C. Hadnagy., 2015. – 255 с.

21. Merritt, D. SPEAR PHISHING ATTACK DETECTION / David Merritt., 2011. – 128 c.

22. Blasi M. Techniques for detecting zero day phishing websites / Michael Blasi., 2009. – 90 c.

23. Ludl C. On the Effectiveness of Techniques to Detect Phishing Sites / C. Ludl, S. McAllister., 2007. – 20 c.

24. Lance J. Phishing Exposed / James Lance., 2005. – 399 c.

ДОДАТОК А

```
# import the required libraries
from googleapiclient.discovery import build
from google_auth_oauthlib.flow import InstalledAppFlow
from google.auth.exceptions import RefreshError
from google.auth.transport.requests import Request
import pickle
import os.path
import base64
import re
import time
import csv
import email
import ipaddress
import win32api
from bs4 import BeautifulSoup
from bs4.element import Comment
import requests
import datetime
from api_keys import whois_key, checkphish_key
import mailbox
from rab import getBody

# Define the SCOPES. If modifying it, delete the token.pickle file.
SCOPES = ['https://www.googleapis.com/auth/gmail.readonly']

result_list = []


def black_list(url):
    scan_payload = {
        "apiKey": checkphish_key,
        "urlInfo": {
            "url": url
        },
        "scanType": "full"
    }

    scan = requests.post("https://developers.checkphish.ai/api/neo/scan",
json=scan_payload)
    jobID = scan.json()
```

```
    time.sleep(20)

    checkphish_payload = {
        "apiKey": checkphish_key,
        "jobID": jobID["jobID"]
    }

    blacklist_request =
requests.post("https://developers.checkphish.ai/api/neo/scan/status",
json=checkphish_payload)
    result = blacklist_request.json()
    # print(result)
    return result["disposition"]


def url_from_text(message, decoded_message):
    soup = BeautifulSoup(decoded_message, 'html.parser')
    href_tags = soup.find_all(href=True)
    pattern = re.compile(
        r"https?:\/\/(www\.)?[-a-zA-Z0-9@:%._\+~#=]{1,256}\.[a-zA-Z0-9()]{1,6}\b([-
a-zA-Z0-9()@:%_\+.~#?&//=]*)")
    email_urls = []

    url_from_message = pattern.findall(message)

    if href_tags:
        import validators
        for i in range(len(href_tags)):
            if validators.url(href_tags[i]["href"]) == True:
                email_urls.append(href_tags[i]["href"])

    if url_from_message:
        for i in range(len(url_from_message)):
            email_urls.append(url_from_message[i][0])

    else:
        print('No url in message')

    email_urls.append(message)
    from collections import OrderedDict
    url_classifier(list(OrderedDict.fromkeys(email_urls)), decoded_message)
```

```
def tag_visible(element):
    if element.parent.name in ['style', 'script', 'head', 'title', 'meta', '[document]']:
        return False
    if isinstance(element, Comment):
        return False
    return True


def text_from_html(body):
    try:
        soup = BeautifulSoup(body, 'html.parser')
        texts = soup.findAll(text=True)
        visible_texts = filter(tag_visible, texts)
        return u" ".join(t.strip() for t in visible_texts)
    except:
        pass


def url_classifier(mail, decoded_message):
    data_set = []

    site_is = {
        "phishing": 0,
        "suspicious": 0,
        "legitimate": 0,
        "email": mail[-1]
    }

    for url_count in range(len(mail) - 1):
        url_features = []
        domain = re.findall(r"://([^/]+)/?", mail[url_count])

        soup = BeautifulSoup(decoded_message, 'html.parser')
        # ----- 8) Presence of JavaScript + -----
        script = soup.find("script")
        if script:
            print(mail[url_count], '---> ', 'JavaScript in mail, seems like phishing')
            site_is["phishing"] += 1
            url_features.append(1)
        else:
            url_features.append(-1)
        # ----- 9) Presence of FORM tag + -----
        form = soup.find("form")
```

```
        if form:
            print(mail[url_count], '---> ', 'Form in mail, seems like phishing')
            site_is["phishing"] += 1
            url_features.append(1)
        else:
            url_features.append(-1)

        # ----- 2) Black list of urls (checkphish API); + -----
        blacklist_result = black_list(mail[url_count])
        if blacklist_result == 'phish':
          site_is["phishing"] += 1
          result_list.append((mail[url_count], site_is))
          return
        elif blacklist_result == 'suspicious':
          site_is["suspicious"] += 1

        # ----- 1) IP address in url; + -----
        try:
            ipaddress.ip_address(domain[0])
            print(mail[url_count], '---> ', 'IP address in URL! Suspiciously')
            site_is["suspicious"] += 1
            url_features.append(1)
        except:
            url_features.append(-1)

        # ----- Number of domains(redirecting) + -----
        redirect_in_url = re.findall(r"(?:url\?sa=t&ct=res&cd=3&url=https?)",
mail[url_count])
        dog_in_url = re.findall(r"@", mail[url_count])
        if redirect_in_url or dog_in_url:
            print(mail[url_count], '---> ', 'redirecting in URL, seems like phishing')
            site_is["phishing"] += 1
            result_list.append((mail[url_count], site_is))
            url_features.append(1)
        else:
            url_features.append(-1)

        # ----- 11) Count of dots in url + -----
        count_of_dots = re.findall(r'\.', mail[url_count])
        if len(count_of_dots) > 3:
            print(mail[url_count], '---> ', 'A lot of dots in URL! Suspiciously')
            site_is["suspicious"] += 1
            url_features.append(1)
```

```
        else:
            url_features.append(-1)

        # ----- 10) Keywords + -----
        words = mail[-1].lower().split()
        if ['suspend', 'verify', 're-verify', 'signin', 'login', 'confirm', 'purchase', 'payment']
in words:
            url_features.append(1)
        else:
            url_features.append(-1)

        # ----- 5) Age of domain (whoisxmlapi) + -----
        payload = {
          'outputFormat': 'JSON'
        }
        domain_age =
requests.get(f'https://www.whoisxmlapi.com/whoisserver/WhoisService?apiKey={w
hois_key}&domainName={mail[url_count]}', params=payload)
        domain_json = domain_age.json()

        age =
datetime.datetime.strptime(domain_json["WhoisRecord"]["registryData"]["createdDa
te"][:10], '%Y-%m-%d')
        today_date = datetime.datetime.strptime(str(datetime.date.today()), '%Y-%m-
%d')
        registred = str(today_date - age).split()
        if int(registred[0]) < 31:
          url_features.append(1)
        else:
          url_features.append(-1)
        #

        # ----- 12) shortener URLs + -----
        try:
            shortener_services = re.match(

'bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'

'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'

'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.u
s|'
```

```
'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'

'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'

'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|
'

'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.
me|v\.gd|tr\.im|link\.zip\.net',
                domain[0])
            if shortener_services:
                site_is["suspicious"] += 1
                url_features.append(1)
            else:
                url_features.append(-1)
        except:
            url_features.append(-1)
        # ----- 13) port in url + -----
        try:
            port_in_url = domain[0].split(':')[1]
            if port_in_url:
                print("port in url")
                site_is["suspicious"] += 1
                url_features.append(1)
        except:
            url_features.append(-1)

        data_set.append(url_features)

    result_list.append((mail[:-1], site_is))

    one_arr = []
    if len(data_set) > 1:
        for x in range(len(data_set)):
            one_arr.append(sum(data_set[x][1:]))
        data_set.append(one_arr)
        true_value = max(data_set[-1])
        data_set[data_set[-1].index(true_value)].append(1)
        with open('phish_set1.csv', mode='a') as phish_set:
            csv_w = csv.writer(phish_set, delimiter=",", lineterminator="\n")
            rows = [data_set[data_set[-1].index(true_value)]]
            csv_w.writerows(rows)
```

```
    elif len(data_set) == 1:
        data_set[0].append(1)
        with open('phish_set1.csv', mode='a') as phish_set:
            csv_w = csv.writer(phish_set, delimiter=",", lineterminator="\n")
            rows = [data_set[0]]
            csv_w.writerows(rows)
    else:
        pass


def mail_decoder(mail):
    if mail.get("payload").get("body").get("data"):
        decoded_message =
base64.urlsafe_b64decode(mail.get("payload").get("body").get("data").encode("ASC
II")).decode(
            "utf-8")
        full_msg = text_from_html(decoded_message)
        url_from_text(full_msg, decoded_message)

    elif mail.get("payload").get("parts"):
        html_mail = mail['payload']['parts'][-1]['body']['data']
        decoded_message = base64.urlsafe_b64decode(html_mail).decode("utf-8")
        full_msg = text_from_html(decoded_message)
        url_from_text(full_msg, decoded_message)

    else:
        small_message = mail.get("snippet")
        print(small_message)


def get_html_text(html):
    try:
        content = BeautifulSoup(html, 'lxml').body.get_text(' ', strip=True)
        return content
    except AttributeError:  # message contents empty
        return None


def phishing_mails():
    mails = mailbox.mbox('emails.mbox')
    # 1272 mails
    index = -1
```

```
    for msg in mails:
        content = msg.get_payload()
        try:
            len(content)
        except:
            pass
        if content and len(content) > 10:
            index += 1
            text = text_from_html(content)
            url_from_text(text, content)

        elif content is None or len(content) > 1:
            try:
                index += 1
                body = getBody(msg)
                t = get_html_text(body)
                url_from_text(t, body)
            except:
                pass
        else:
            # get_html_text()
            print('len <<<')

        if index == len(mails):
            break


def getEmails():
    creds = None

    if os.path.exists('token.pickle'):
        with open('token.pickle', 'rb') as token:
            creds = pickle.load(token)

    if not creds or not creds.valid:
        if creds and creds.expired and creds.refresh_token:
            creds.refresh(Request())
        else:
            flow = InstalledAppFlow.from_client_secrets_file('credentials.json', SCOPES)
            creds = flow.run_local_server(port=0)

        with open('token.pickle', 'wb') as token:
```

```
            pickle.dump(creds, token)

    service = build('gmail', 'v1', credentials=creds)

    mail_list = service.users().messages().list(userId='me', labelIds=['INBOX'],
maxResults=400).execute()
    messages = mail_list.get('messages')
    for i in range(len(messages)):
        mail = service.users().messages().get(userId='me', id=messages[i]['id'],
format="full").execute()
        # print(mail)
        mail_decoder(mail)


getEmails()
phishing_mails()

for i in result_list:
        print('\n', i)


for j in range(len(result_list)):
        # print(j, '\n', result_list[j], type(result_list[j]), '\n\n')
        try:
                if result_list[j][1]["phishing"] == 1:
                        win32api.MessageBox(0, f'Phishing URL in email:
{result_list[j][0]} \nMail text:\n{result_list[j][1]["email"]}', 'anti-phisher')

                elif result_list[j][1]["suspicious"] >= 2:
                        win32api.MessageBox(0, f'Suspicious URL in email:
{result_list[j][0]} \n Labeled as phishing ', 'anti-phisher')
        except:
                if result_list[j]["phishing"] == 1:
                        win32api.MessageBox(0,
                                                        f'Phishing URL in email: Mail
text:\n{result_list[j]["email"]}',
                                                        'anti-phisher')
                elif result_list[j]["suspicious"] >= 2:
                        win32api.MessageBox(0, f'Email: {result_list[j]["email"]} \n
Labeled as phishing ',
                                                        'anti-phisher')
```

ДОДАТОК Б

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.model_selection import cross_validate
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm , preprocessing
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns



def mean_score(scoring):
    return {i:j.mean() for i,j in scoring.items()}

dataset = pd.read_csv('phish_set.csv')

X = dataset.drop('Result', axis=1)
y = dataset['Result']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

scoring = {'accuracy': 'accuracy',
           'recall': 'recall',
           'precision': 'precision',
           'f1': 'f1'}
fold_count=10

dtree_clf=DecisionTreeClassifier()
cross_val_scores = cross_validate(dtree_clf, X, y, cv=fold_count, scoring=scoring)
dtree_score = mean_score(cross_val_scores)
dtree_clf.fit(X_train, y_train)
dtr = dtree_clf.predict(X_test)
dtttr = pd.DataFrame({'Actual':y_test, 'Predicted':dtr})
print(dtttr, '\n')
print('dtree_score\n',dtree_score, '\n')


KNeighbors_clf=KNeighborsClassifier(1)
```

```
cross_val_scores = cross_validate(KNeighbors_clf, X, y, cv=fold_count,
scoring=scoring)
KNeighbors_clf_score = mean_score(cross_val_scores)
KNeighbors_clf.fit(X_train, y_train)

y_sec_pred = KNeighbors_clf.predict(X_test)
knn_df = pd.DataFrame({'Actual':y_test, 'Predicted':y_sec_pred})
print(knn_df, '\nn')
print('KNeighbors_clf_score\n', KNeighbors_clf_score, '\n')

rforest_clf=RandomForestClassifier()
cross_val_scores = cross_validate(rforest_clf, X, y, cv=fold_count, scoring=scoring)
rforest_clf_score = mean_score(cross_val_scores)
rforest_clf.fit(X_train, y_train)
frst = rforest_clf.predict(X_test)
frst_df = pd.DataFrame({'Actual': y_test, 'Predicted': frst})
print(frst_df, '\n')
print('rforest_clf_score\n', rforest_clf_score, '\n')

linear_clf = svm.SVC(kernel='linear')
cross_val_scores = cross_validate(linear_clf, X, y, cv=fold_count, scoring=scoring)
linear_svc_clf_score = mean_score(cross_val_scores)
linear_clf.fit(X_train, y_train)
pred_y = linear_clf.predict(X_test)
thrd_df = pd.DataFrame({'Actual':y_test, 'Predicted':pred_y})
print(thrd_df, '\n')
print('linear_svc_clf_score\n', linear_svc_clf_score, '\n')
```