Міністерство освіти і науки України

«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій

Кафедра кібербезпеки та захисту інформації

ПОЯСНЮВАЛЬНА ЗАПИСКА

дипломної роботи

магістра

(назва освітнього рівня)

галузь знань 12 Інформаційні технології

(шифр і назва галузі знань)

спеціальність 125 «Кібербезпека»

(код і назва спеціальності)

освітня програма «Кібербезпека»

(назва освітньої програми)

на тему: «Метод детектування фішингових сайтів на основі нечіткої кластеризації індикаторів компрометації»

Виконавець: студентка 2 курсу, групи КБм-21

(підпис) **Пяташова Еліна Сергіївна**

(прізвище ім'я по-батькові)

	Прізвище, ініціали	**Підпис**
Науковий керівник	Бучик С.С.	

Рецензент	Самохвалов Ю.Я.	

Нормоконтроль	Даков С.Ю.	

Київ 2022

Міністерство освіти і науки України

«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій

Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувач кафедри
кібербезпеки та захисту інформації

_____Лукова-Чуйко Н.В.

“_____”_____2021 р.

ЗАВДАННЯ
на виконання дипломної роботи

спеціальність _____ 125 «Кібербезпека»
(код і назва напряму підготовки)

Студентці **КБм-21** **Пяташовій Еліні Сергіївни**
(група) (прізвище ім’я по-батькові)

Тема дипломної роботи Метод детектування фішингових сайтів на основі нечіткої кластеризації індикаторів компрометації

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема дипломної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол № 5 від 29.10.2021

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об’єкт досліджень Процес детектування фішингових сайтів за допомогою машинного навчання

Предмет досліджень Методи, засоби і методики виявлення фішингових сайтів

Мета Розробка ефективного методу детектування фішингових сайтів на основі нечіткої кластеризації індикаторів компрометації

Вихідні дані для проведення роботи Метод детектування фішингових сайтів

3. ОЧІКУВАНІ РЕЗУЛЬТАТИ

Наукова новизна Вдосконалення моделі детектування фішингових сайтів на основі нечіткої кластеризації С-середніх індикаторів компрометації за рахунок поєднання найближчих кластерів при умові, що ефективність моделі найбільша, коли кількість кластерів більше двох.

Практична цінність Результати практичної частини можуть бути використані будь-яким вендором.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Робота виконана у повному обсязі відповідно до теми.

5. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
Уточнення постановки задачі	29.10.2022 – 22.01.2022	виконано
Аналіз літератури	23.01.2022 – 11.02.2022	виконано
Обґрунтування вибору рішення	12.02.2022 – 15.02.2022	виконано
Вивчення сучасних методів детектування фішингових веб-сайтів	16.02.2020 – 04.03.2022	виконано
Розробка операційної структури дослідження	05.03.2020 – 10.03.2022	виконано
Вивчення набору даних для проведення дослідження	11.03.2022 – 21.03.2022	виконано
Імплементація методу детектування фішингових вебсайтів на основі нечіткої кластеризації С-середніх	22.03.2020 – 08.04.2022	виконано
Порівняння ефективності методу нечіткої кластеризації С-середніх та широковідомих методів класифікації	09.04.2020 – 09.05.2022	виконано

Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
Оформлення пояснювальної записки	10.05.2022 – 15.05.2022	виконано
Підготовка до захисту дипломної роботи	16.05.20202– 19.05.2022	виконано

6. РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект Зниження збитків через запобігання викраденню даних через фішингові веб-сайти

Соціальний ефект Покращення технологій забезпечення захисту від фішингових атак як особисто так і на підприємствах.

7. ДОДАТКОВІ ВИМОГИ

Завдання видав _____ С.С. Бучик
(підпис) (ініціали, прізвище)

Завдання прийняв до виконання _____ Е.С. Пяташова
(підпис) (ініціали, прізвище)

Дата видачі завдання:
Термін подання дипломної роботи до ЕК

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Метод детектування фішингових сайтів на основі нечіткої кластеризації індикаторів компрометації»: 111 сторінок, 28 рисунків, 1 додаток та 6 таблиць. 109 літературних джерел.

Об’єкт дослідження – процес детектування фішингових сайтів за допомогою машинного навчання.

Мета роботи – розробка ефективного методу детектування фішингових сайтів на основі нечіткої кластеризації індикаторів компрометації.

Методи дослідження – метод некерованого машинного навчання з використаням алгоритму нечіткої кластеризації С-середніх.

У роботі досліджено сучасні загрози та методи протидії фішинговим атакам з використанням фейкових веб-сайтів. Проведено аналіз ринку рішень, використовуючих методи машинного навчання для детектування фішингових сайтів. Запропоновано метод детектування фішингових сайтів на основі нечіткої кластеризації індикаторів компрометаці. Визначена ефективність запропонованого методу.

Була вдосконалена модель детектування фішингових сайтів на основі нечіткої кластеризації С-середніх індикаторів компрометації за рахунок поєднання найближчих кластерів при умові, коли ефективність моделі найбільша, коли кількість кластерів більше двох.

Актуальність теми: фішингові атаки є серйозною загрозою безпеки конфіденційних даних. Традиційне навчання користувачів не здатне в повній мірі протистояти невідомим загрозам. Використання машинного навчання наразі є найефективнішнім механізмом виявлення фішингових сайтів. Тому представлення методу, у якому детектування реалізование на основі нечіткої кластеризації, що забезпечує автоматизовану роботу алгоритму без втручання наглядача, є актуальною темою.

Ключові слова: нечітка кластеризація, детектування фішингових сайтів.

ЗМІСТ

РЕФЕРАТ 5

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ 8

ВСТУП 9

РОЗДІЛ 1 ДЕТЕКТУВАННЯ ФІШИНГОВИХ САЙТІВ ЗА ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ 11

1.1 Фішинг та фішингові веб-сайти як вид кіберзлочину 11

1.1.1 Індикатори компрометації фішингових атак 18

1.1.2 Методи протидії фішинговим атакам 20

1.1.3 Підходи щодо виявлення фішингу 24

1.2 Використання машинного навчання в кібербезпеці 28

1.2.1 Керований та некерований підхід у машинному навчанні 34

1.2.2 Класифікація, кластеризація та регресія в машинному навчанні 41

1.2.3 Нечітка кластеризація як метод некерованного машинного навчання 46

1.2.4 Нечітка кластеризація в кібербезпеці 55

1.3 Сучасний стан детектування фішингових сайтів за допомогою машинного навчання 57

1.3.1 Наявні рішення при детектуванні фішингових сайтів за допомогою машинного навчання 58

1.3.2 Популярні моделі машинного навчання при детектуванні фішингових сайтів 59

1.4 Потенціал розвитку методів детектування фішингових сайтів та постановка задач дослідження 62

Висновки за розділом 1 62

РОЗДІЛ 2 РОЗРОБКА МЕТОДУ ДЕТЕКТУВАННЯ ФІШИНГОВИХ САЙТІВ НА ОСНОВІ НЕЧІТКОЇ КЛАСТЕРИЗАЦІЇ ІНДИКТОРІВ КОМПРОМЕТАЦІЇ 65

2.1 Операційна структура дослідження 65

2.2 Набір даних для дослідження 67

Висновки за розділом 2 74

РОЗДІЛ 3 ІМПЛЕМЕНТАЦІЯ МОДЕЛІ НЕЧІТКОЇ КЛАСТЕРИЗАЦІЇ ДЛЯ ДЕТЕКТУВАННЯ ФІШИНГОВИХ САЙТІВ 76

3.1 Імплементація, тестування та оцінка ефективності моделі нечіткої кластеризації індикаторів компрометації 76

3.2 Порівняння з іншими методами 84

3.2.1 Алгоритм Naïve Bayes 85

3.2.2 Алгоритм Logical Regression 88

3.2.4 Алгоритм Random Forest і Decision tree 90

3.2.5 Алгоритм SVM 91

3.2.6 Результати порівняння 93

Висновки за розділом 3 94

ВИСНОВКИ 96

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 99

ДОДАТОК А 111

ДОДАТОК В 112

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

API - програмний інтерфейс додатку
DNS - система доменних імен
DOM - об’єктна модель документу
FACA - алгоритм швидкої асоціативної класифікації
FCM - нечітка кластеризація С-змінних
FPC - нечіткий коефіцієнт розподілу
HTML - мова гіперструктурної розмітки
IOC - індикатори компрометації
IP - інтернет протокол
MX - почтовий сервер
NS - назва серверу
RSS - збагачене зведення сайту
SSL - рівень захищених сокетів
SVM - метод опорних векторів
TTL - час життя IP-пакетів
URL - універсальний локатор ресурсів

ВСТУП

Актуальність теми магістерської роботи визначається тим, що традиційне навчання користувачів виявляти фішингові сайти не здатне в повній мірі протистояти новим невідомим загрозам і технікам шахраїв. Детектування злочинних веб-сайтів за допомогою програмного забезпечення автоматизує цю діяльність, але сучасні методи детектування фішингових сайтів теж мають свої недоліки. Вони полягають в тому, що потрібно заздалегідь натренувати модель визначати фішингові сайти на основі набору даних з вже класифікованим набором легальних і нелегальних сайтів, не враховуючи те, що різновиди кіберзлочинів еволюціонують кожного дня. Через це потрібно постійно оновлювати тренувальну базу, що не є зручним і ефективним рішенням. Саме тому представлення методу, у якому детектування реалізование на основі нечіткої кластеризації, що забезпечує автоматизовану роботу алгоритму без втручання наглядача, є актуальною темою.

Основною задачею цієї кваліфікаційної роботи є покращення ефективності методу нечіткої кластеризації індикаторів компрометації C-середніх в задачі детектування фішингових сайтів.

Науковою новизною є вдосконалення моделі детектування фішингових сайтів на основі нечіткої кластеризації C-середніх індикаторів компрометації за рахунок поєднання найближчих кластерів при умові, що ефективність моделі найбільша, коли кількість кластерів більше двох.

Об'єктом дослідження в даній роботі є процес детектування фішингових сайтів за допомогою машинного навчання.

Предметом дослідження в даній роботі є методи, засоби і методики детектування фішингових сайтів.

Метою роботи є розробка ефективного методу детектування фішингових сайтів на основі нечіткої кластеризації індикаторів компрометації.

Для досягнення поставленої мети роботи необхідно вирішити наступні *завдання*:

- розглянути сучасні підходи і техніки детектування фішингових сайтів;
- провести аналіз існуючих робіт детектування фішингових сайтів на основі машинного навчання;
- імплементувати метод нечіткої кластеризації С-середніх для детектування фішингових сайтів та визначити його ефективність.

Сформовані в результаті, теоретичні та практичні рекомендації можуть бути використані будь-яким вендором , адже використання ефективного алгоритму нечіткої кластеризації індикаторів компрометації автоматизує процес прийняття рішень щодо детектування фішингових сайтів.

РОЗДІЛ 1
ДЕТЕКТУВАННЯ ФІШИНГОВИХ САЙТІВ ЗА ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ

1.1 Фішинг та фішингові веб-сайти як вид кіберзлочину

Організована злочинність все частіше використовує всесвітню павутину Інтернет для реалізації, а також водночас і приховування своєї діяльності.

Згідно зі звітом консалтингової компанії Accenture «Стан стійкості кібербезпеки» [1], середня кількість кібератак на підприємство в рік зросла з 206 за рік у 2020 році до 270 за рік у 2021 році, тобто у процентному співвідношенні кількість збільшилась на 31%. Національна поліція України в свою чергу теж надала дані, згідно яких кількість організованих груп і злочинних організацій, які використовують інформаційні технології при вчиненні кримінальних правопорушень, за 2020 рік збільшилась на 36 % [2].

Кіберзлочин (комп'ютерний злочин), згідно з законом України "Про основні засади забезпечення кібербезпеки України" [3] визначається як суспільно небезпечне винне діяння в кіберпросторі та або з його використанням, відповідальність за яке передбачена законом України про кримінальну відповідальність та/або яке визнано злочином міжнародними договорами України».

Конвенція Ради Європи про кіберзлочинність [4], яку Верховна Рада України ратифікувала й імплементувала до українського законодавства починаючи з 11.10.2005, запропонувала класифікацію кіберзлочинів, виокремивши наступні чотири основні типи:

- кіберзлочини, пов'язані з комп'ютерами, – підробка та шахрайство, пов'язані з комп'ютерами;

- кіберзлочини проти цілісності, доступності та конфіденційності комп'ютерних систем і даних – втручання в дані, нелегальне перехоплення, незаконний доступ, втручання в систему, зловживання пристроями;
- кіберзлочини, пов'язані з порушенням авторських і суміжних прав;
- кіберзлочини, пов'язані зі змістом, – кіберзлочини, пов'язані з дитячою порнографією;

Одним з найпоширеніших видів кіберзлочинів є фішинг. Наприклад, у 2021 році рівень фішингових атак серед користувачів Інтернету в Бразилії склав 12,39 відсотки. В цій країні користувачі найчастіше ставали мішенню фішингових атак, порівняно зі світом. У досліджуваному році Франція була другою за часткою атакованих користувачів Інтернету. Далі пішла Португалія з понад 11% фішингових атак [5]. Повна статистика наведена на рисунку 1.1.

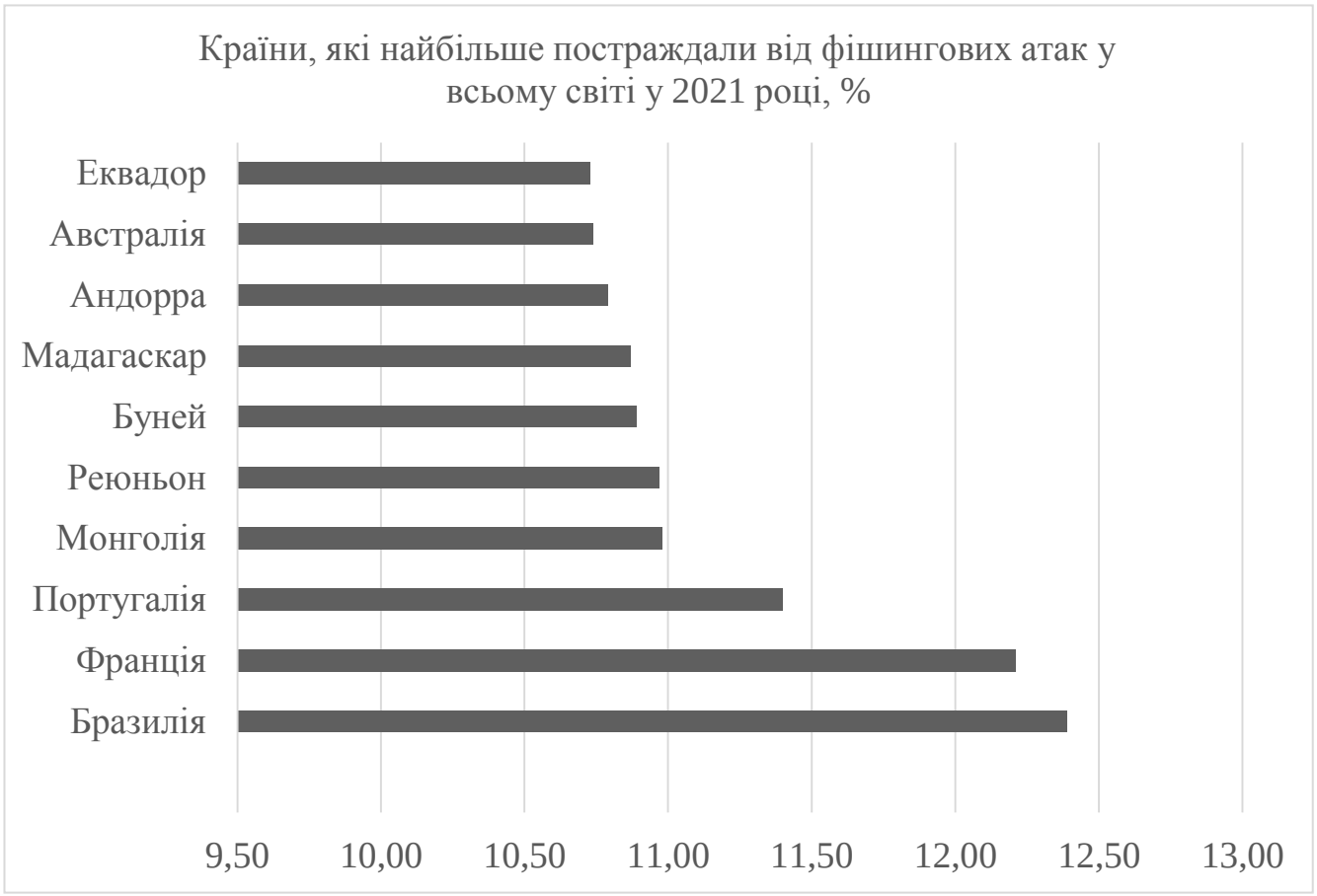

Рисунок 1.1 – Країни, які найбільше постраждали від фішингових атак у всьому світі у 2021 році

Згідно з NIST SP 800-12 [6] фішинг це метод спроби отримання конфіденційних даних, наприклад зловмисник може спробувати отримати номери банківських рахунків за допомогою завідомо шахрайських запитів електронною поштою чи за допомогою веб-сайту-підробки, за допомогою яких зловмисник видає себе за законну ділову чи авторитетну особу, якої можна довіряти. Зловмисники часто використовують фішинг як спосіб отримання дійсних облікових даних коричтувача-жертви або іншої особистої інформації.

Крім того, фішинг – це онлайн-атака, яку зловмисники використовують для шахрайства за допомогою схем соціальної інженерії за допомогою миттєвих повідомлень, електронної пошти або онлайн-реклами, щоб заманити користувачів на фішингові веб-сайти, схожі на законний веб-сайт для отримання конфіденційної інформації про жертву, наприклад як пароль, фінансовий рахунок, ідентифікаційний номер особи та номери фінансових рахунків, які потім можуть бути використані для отримання нелегального прибутку [7].

В цьому контексті визначення фішингу не є фіксованим, але можна цей термін розглядати як факт щодо цілей атаки, який не можна заперечити, змінюються в даному випадку лише методи здійснення. Наприклад, використання веб-сайтів-підробок або електронної пошти є двома методами фішингу, які мають спільні цілі, але між якими є деякі відмінності.

Саме через нефіксоване визначення фішингу корстувачам мережі необхідно розуміти та знати доволі широкий спектр характеристик і класичних, і сучасних, і передових методів та технік фішингових атак, що також включає і визначення сфер, в яких методи і способи боротьби з фішингом просто відсутні, як приклад, від фішингових атак може не бути застрахована навіть критична інфраструктура великої країни. В Україні, наприклад, в грудні 2015 року мережеві адміністратори та IT-персонал компаній, що здійснювали розподіл електроенергії в країні, опинилися мішенню фішингової атаки. Вона включала надсилання небезпечного файлу типу Microsoft Word, який надавав запит на ввімкнення макросів. Одразу після увімкнення макрос інсталював шкідливе програмне забезпечення під назвою BlackEnergy3 в систему, тим самим забезпечивши для зловмисників бекдор. Як

результат така атака стала причиною успішного відключення 30 енергетичних підстанцій, тим самим залишивши 230 000 людей без електроенергії на понад шість годин [8]. Цей приклад насамперед демонструє, наскільки руйнівною і потужною може стати добре спланована і гарно виконана фішинг-атака.

Отож загалом можна виділити 3 складові здійснення фішингових атак [9]:

- середовище – голосове, СМС чи Інтернет;
- вектор атаки – є каналом, через який здійснюється фішингова атака, наприклад, електронний лист чи веб-сайт – підробка, а також залежить від середовища, яке використовує зловмисник;
- технічний підхід - метод, який часто використовується разом із вектором атак, щоб підвищити шанси зловмисника на успіх, наприклад, міжсайтові сценарії або вразливості браузера.

Вищеперераховані складові здійснення фішингових атак є взаємопов’язаними, причому деякі з компонентів можна співвіднести лише для конкретних середовищ, а деякі технічні підходи — лише для певних векторів атак. Найважливішим компонентом є вектори фішингових атак. Як сказано вище, вектори атак залежать від середовища, яким користується зловмисник, а також є каналом, через який здійснюється атака, що можна побачити на рисунку 1.2.

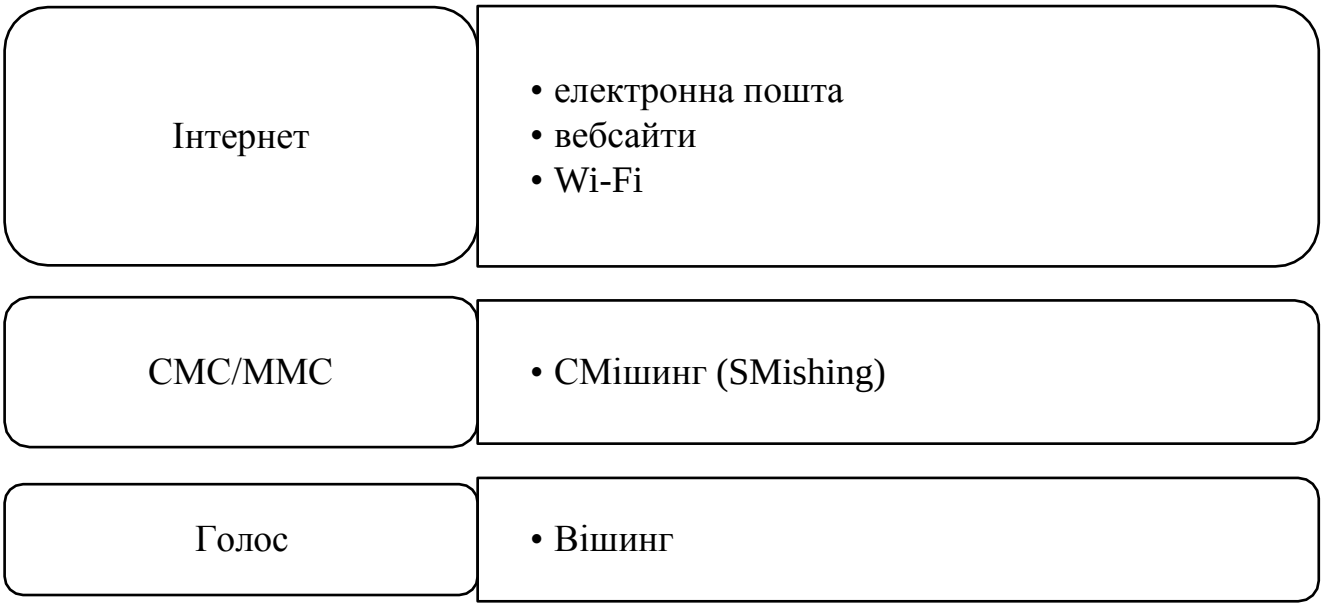

Рисунок 1.2 – Вектори фішингових атак

Інтернет – середовище, яке містить найбільший діапазон векторів, як можна було очікувати. Найбільш типовими і знайомими векторами фішингових атак в цьому середовищі є:

- електронна пошта – вектор атаки, при якому цілі атаки отримують спеціально створені електронні листи. Розповсюджувані зловмисниками, які спонукають жертв виконати дії, які зроблять доступними для зловмисника їх персональні дані;
- веб-сайти – вектор атаки, при якому зловмисники створюють фейкові сайти, щоб видати їх за законні і добре знайомі, а потім використати їх для збору персональних даних жертв, наприклад, зібравши необхідну інформацію при спробі жертви авторизуватися на сайті;
- wi-fi – вектор атаки, при якому зловмисники використовують методи, які перехоплюють трафік у wi-fi мережах з метою отримати або вкрасти, що більш точно, персонаааальну інформацію жертв, яку вони передають, при використанні загальнодоступної точки доступу.

З вищеописаних векторів фішингових атак найпоширенішими є використання електронної пошти та вебсайтів, які йдуть рука об руку, для отримання особистих даних жертв. Наприклад, у 2021 році серед 3500 організацій у всьому світі було проведено опитування [10], при якому три відсотки опитаних співробітників зізнались, що натиснули на шкідливе посилання, коли отримали шкідливий електронний лист, тим самим допомогли зловмисникам при спробі кіберзлочину в сторону організації, в якій працювали, в той же час існує дослідження [11], яке показує, зріст сумарної кількості унікальних фішингових сайтів збільшилась до 571,764 в третьому кварталі 2020 року порівняно з другим кварталом, коли кількість варіювалася у межах 150 тисяч, що доказує поширеність і популярність таких векторов фішингових атак, як веб-сайти та електронні листи. Повна статистика наведена на рисунку 1.3

Рисунок 1.3 – Кількість ідентифікованих фішингових сайтів у період 2013-2021 рр

В той же час, існують дослідження [12], [13], які доводять, що користувачі всесвітньої павутини більше схильні думати, що фішингові атаки здійснюються в основному тільки за допомогою електронної пошти або інших сервісів швидкого обміну повідомленнями онлайн, та як результат, ці користувачі, як правило, набагато менше піклуються про безпеку персональних і конфіденційних даних під час відвідування різноманітних веб-сайтів, що натомість робить їх уразливими до таких векторів фішингових атак, як фішингові веб-сайти.

У такому випадку, коли нічого не підозрючий користувач інтернету переходить за фішинговим посиланням, отриманим, наприклад, за допомогою електронного листа, він спрямовується на фішинговий веб-сайт, що знаходиться під повним контролем шахрая. Сам фішинговий вебсайт буде розроблений і підготований таким чином, щоб він не викликав ніяких сумнівів і виглядав дуже знайомим для жертви. Шахраї-фішери зазвичай візуалаьно підроблюють фірмовий стиль вже існуючого сайту або цільової організації, використовуючи схожі кольори, іконки, логотипи та текстовий контент. В електронному листі, який шахрай відправив жертві, було запропоновано, наприклад, "оновити" свою особисту інформацію на сайті або зробити що завгодно, що можна запропонувати зробити у

письмі, не викликаючи підозру і при цьому спонукаючи ввести свої авторизаційні дані (тобто ім'я користувача та пароль) для доступу до веб-сайту. У випадку, якщо користувач-жертва вводить свої реальні логін та пароль для входу на підроблений веб-сайт, то у подальшому зловмисник, використовуючи інформацію, надану жертвою, може потім видавати себе за неї на справжньому онлайн-сервісі. Це може дозволити зловмиснику, наприклад, перерахувати деяку або всю сумму коштів з рахунків потерпілої жертви фішингової атаки або завдати іншої фінансової або репутаційної шкоди. Необхідно зазначити, що хоча фішинг, як вид кіберзлочину, отримує достатньо широке висвітлення в засобах масової інформації (тим самим збільшучи кількість користувачів всесвітньої павутини, які б могли щось чути про постійне зростання кількості фішингових атак, особливо у період пандемії), дані кібератаки все ще залишаються результативними, так як шахраї-фішери відповідно до сучасних обставин постійно адаптують свої спроби соціальної інженерії. У результаті велика кількість фішингових електронних повідомлень тепер спонукають жертв підтвердити свою персональну інформацію з "цілями підвищення безпеки аккаунту", нібито тому, що цільова організація, яку імітують зловмисники, хотіла б щоб захистити своїх користувачів від загрози фішингу [14].

Загалом виділяють 3 компоненти здійснення фішингових атак за допомогою веб-сайтів [15]:

- приманка – найчастіше буває у вигляді повідомлення електронної пошти, яке, як здається жертвам фішингу, надійшло від дійсної і реальної організації, наприклад, банку або будь-якого іншого постачальника послуг, на яку повідомлення містить посилання
- гачок – це веб-сайт, який імітує сайт законної і дійсної установи, якої жертва фішингової атаки готова розкрити персональну інформацію. Гачок часто приховується шляхом обфускації унікального локатору ресурсів (URL-адреси);
- улов – зібрана зловмисниками-фішерами особиста інформація, отримана від жертви фішингової атаки.

Гачки (фішингові веб-сайти) можна розділити на два поширені типи, а саме: веб-сайти – підробки, які імітують реальні, та вигадані самими зловмисникими веб-сайти [16]. Вебсайтами-підробками вважають фіктивні копії реально існуючих комерційних веб-сайтів. Дуже часто шахраї імітують такі популярні веб-сайти як eBay, PayPal, сайти різних банківських установ, онлайн-магазини або постачальників послуг умовного депонування [17]. Веб-сайти-підробки розроблені з ціллю імітувати сайти легальних організацій і викрасти особисті дані нічого не підозрюючих користувачів інтернету: логіни та паролі облікових записів, персональну інформацію, номери кредитних карток тощо.

З метою зменшити ефективність фішингових атак і навчання користувачів, були засновано різноманітні сервіси і онлайн-сховища, наприклад, такі як PhishTank, які зберігають в своїх базах даних URL-адреси мільйонів перевірених користувачами веб-сайтів-підробок, які зловмисники використовують або використовували для фішингових атак, імітуючи законні онлайн-платформи, яким довіряють користувачі кожну хвилину.

Вигадані веб-сайти в свою чергу намагаються створити у потенційних жертв враження унікальних, дійсних і законних комерційних структур, таких як, наприклад, інвестиційні банки, служби надання послуг умовного депонування, судноплавні компанії чи онлайн-аптеки та магазини, тим самим вводячи користувачів в оману [18]. Метою вигаданих веб-сайтів, на відміну від сайтів-підробок є шахрайство з невідвантаженням та ненаданням послуг; шахрайство з грошима клієнтів, не дотримання своїх обіцянок [19].

Як вебсайти-підробки, так і вигадані веб-сайти в той же час можуть використовуватися з метою поширення шкідливих програм і вірусів [20].

1.1.1 Індикатори компрометації фішингових атак

Індикатори компрометації (IOC) — це криміналістичні артефакти, які залишаються після незаконних кібервторгнень, які ідентифікуються в інформаційних системах організації-жертви на рівні мережі або хоста. IOC надають

організаціям-жертвам цінну інформацію про об'єкти або інформаційні системи, які були атаковані та скомпрометовані. Індикатори компрометації на рівні скомпрометованих та атакованих хостів можуть включати в себе, наприклад, створення значень ключів реєстру. В свою чергу індикатори компрометації на рівні мережі і мережевого трафіку включають, наприклад, універсальний локатор ресурсів або ті елементи мережевих протоколів, які вказують на сервери команд та керування зловмисним програмним забезпеченням. Швидке впровадження та розповсюдження IOC може покращити інформаційну безпеку організацій за рахунок скорочення часу, при якому інформаційні системи та організації стають уразливими до одного і того ж кіберзлочину або виду атаки [21].

В контексті фішингових атак прикладами IOC можуть бути:

- підозрілі файли та програми в отриманих електронних листах;
- домени та IP-адреси, що належать до бот-мереж та/або серверів керування шкідливим програмним забезпеченням;
- виявлення передачі даних через порти, які рідко використовуються;
- сигнатура хеш-файлу або атаки відомого зловмисного програмного забезпечення;
- незвичний розмір відповідей HTML.

Індикатори компрометації, які є пов’язаними з конкретною загрозою, виділяються при процесі аналізу цієї загрози організацією. Наприклад, якщо кіберрозвідка виявила нове програмне забезпечення, яке опинилося шкідливим, у звітах про це забезпечення будуть наведені такі індикатори компрометації, як файли, адреси командних серверів і так далі. В подальшому ці IoC будуть використовуватися для активного пошуку загроз в середині інфраструктури організації. Виявлення IoC в системі означає те, що, імовірно, на цю систему вже ведеться кібератака і необхідно як найскоріше прийняти заходи реагування.

Індикатори компрометації також можна знайти в базах даних антивірусного програмного забезпечення та пасивних засобів моніторингу, доданих туди з ціллю своєчасно та автоматично виявляти та блокувати спроби незаконного проникнення у

систему [22]. Наприклад, у випадку, коли антивірусне програмне забезпечення розпізнає фішинговий веб-сайт та попереджає користовача про знахідку.

1.1.2 Методи протидії фішинговим атакам

Зазвичай проблему фішингу фахівці вирішують за допомогою використання евристичного підходу, який включає в себе навчання користувачів, технологічні вдосконалення та розробку і дотримання відповідних бізнес-процесів, які повинні зменшити ризик становлення жертвою фішингової атаки.

Ключову позицію в процесі розпізнавання фішингових атак займають аналітичні навички та здібності користувача мережі під час використання ним електронних каналів зв'язку. Більша частина технологічних співробітників не є знайомою з прийнятими моделями взаємодії з інформаційними системами, які знаходяться у них в використанні. У результаті шахраям-фішерам стає набагато легше підробити веб-інтерфейс якоїсь знайомої потенційній жертві веб-сторінки з метою її імітування та спонукання користувача ввести свою персональну інформацію, яка натомість опиниться у руках зловмисників. Саме через ймовірність реалізації даного ризику основний акцент в процесі захисту від фішингових атак приділяється навчанню користувачів [15].

Користувачів мережі можна і треба навчати для кращого розуміння ними природи фішингових кібератак, що в свою чергу обов'язково приведе до коректного розпізнавання користувачами фішингових підробок і дійсних веб-сайтів. Цей підхід є протилежним до загальноприйнятої категоризації подібного навчання в організаціях, при якому навчання користувачів мережі розглядається у якості запобіжного заходу. Натомість більш сучасною є теорія, яка констатує, що так як навчання користувачів направлено на виявлення ними фішингових атак в кінці кінців, тому і розглядається як підхід до їх розпізнавання, а не як запобіжний захід [23]. Необхідно підкреслити, що зазвичай фішинг, як кібератака, є націленою на кількох користувачів з однієї організації чи кількох різних організацій, тому обмін

знаннями, який стосується попередження інших користувачів мережі про фішингові атаки стає рівно настільки ж важливим, як і саме по собі розпізнавання атак.

В той же час знання, які були отримані в процесі розуміння природи фішингу, можуть допомогти компаніям точно налаштувати бізнес-процеси організації та усунути лазівки аутентифікації в внутрішніх процедурах. В такому випадку потрібно розробити бізнес-процеси таким чином, щоб відповідні противаги і стримування підтримувалися на місцях, а інформовані відповідні рішення користувачів підкріплювалися підтримкою на рівні самого бізнес-процесу за допомогою множинних перевірок в розподіленому ланцюжку команд, офлайн- та онлайн-перевіркою, тощо.

Різноманітне програмне забезпечення та/або технологічне покращення в процесі захисту від фішингу в свою чергу автоматизує цей процес захисту [15]. Щоб захистити користувачів від фішингових атак, були запропоновані різноманітні методи боротьби з фішингом, засновані на дотриманні різних стратегій, таких як захист на стороні сервера та на стороні клієнта [24]. Частина методів працює з електронною поштою, частина працює з URL-адресами веб-сайтів, а частина з атрибутами веб-сайтів. Існує ще частина, яка зосереджена на тому, щоб допомогти користувачам виявляти та фільтрувати різні типи фішингових атак. Загалом методи протидії фішинговим атакам можна розподілити в чотири наступні категорії [25]:

1. метод фільтрації вмісту;
2. метод чорного списку;
3. метод профілактики на основі симптомів;
4. прив'язка домену.

В основі методу фільтрації змісту є твердження, що контент електронної пошти перевіряється і фільтрується під час надходження до поштової скриньки потенційної жертви за допомогою методів машинного навчання, з використанням таких алгоритмів керованого навчання як машини опорних векторів (SVM) чи байєсівські дерева адитивної регресії.

Метод чорного списку використовує колекцію відомих і визнаних шахрайських веб-сайтів/адрес, опублікованих надійними організаціями, наприклад,

як Google і Microsoft. Метод чорного списку включає в себе як клієнтський, так і серверний компонент. Клієнтський компонент використовується в якості плагіну для браузера або електронної пошти, який є пов’язаним із компонентом сервера, який в свою чергу і у даному випадку являє собою загальнодоступний веб-сайт, однією з функцій якого є надання доступу до списку визнаних фішингових сайтів.

Головною ідеєю методу профілактики на основі симптомів є оцінка вмісту кожної і будь-якої веб-сторінки, яку користувач відвідує, а потім подальшу відправку фішингових сповіщеннь, які містять перелік і опис типів та кількість виявлених симптомів фішингової атаки [26].

Метод прив’язки домену заснований на браузері клієнта, в якому персональна інформація користувача (наприклад, авторизаційні дані) прив’язується до певного домену. При використанні даного методу система попереджає користувача кожен раз, коли він відвідує домен, до якого його облікові дані не були прив'язані [24].

Існує і інша категоризація методів протидії фішинговим атакам, яка містить в собі п’ять категорій, а саме [24]:

1. метод захисту від фішингу на основі атрибутів;
2. метод захисту від фішингу на основі загальних алгоритмів;
3. метод захисту від фішингу на основі ідентифікації;
4. підхід проти фішингу на основі символів;
5. підхід до захисту від фішингу на основі вмісту.

Стратегія методу захисту від фішингу на основі атрибутів полягає в реалізації, як реактивного, так і проактивний захисту від фішингових атак. Така техніка була реалізована в інструменті PhishBouncer.

Метод захисту від фішингу на основі загальних алгоритмів – це підхід для виявлення фішингових заснований на використанні алгоритму «Якщо, то», наприклад: «якщо в електронному листі існує IP-адреса URL-адреси і вона не відповідає визначеному набору правил для білого списку, тоді отримане повідомлення є фішинговим листом»;

Метод захисту від фішингу на основі ідентифікації полягає в дотриманні методології взаємної аутентифікації, коли користувач і онлайн-об’єкт

підтверджують ідентичність один одного під час рукостискання. Цей метод об’єднує частковий обмін обліковими даними та техніку фільтрації клієнтів, щоб запобігти легкому маскуванню фішингових веб-сайтів під законні онлайн-об’єкти. Оскільки виконується взаємна автентифікація, користувачам не потрібно повторно вводити свої облікові дані. Тому паролі ніколи не обмінюються між користувачами та онлайн-організаціями, крім як під час початкового процесу налаштування облікового запису [24].

Метод протидії фішингу на основі символів та підхід захисту від фішингу на основі вмісту буде розглядатися в підходах щодо виявлення фішингових атак.

Однією із перших повинна бути прийнята та реалізована фільтрація електронного вмісту даних і повідомлень, якими обмінюються користувачі в корпоративних мережах. Це також означає, що дані повинні бути зашифровані з метою забезпечення цілісності даних, що розповсюджуються, запобігання отруєнню цих даних та підвищення рівня довіри до власних даних і повідомлень. Разом з цим потрібно також налаштувати антифішингові системи і сервіси, які призначені для фільтрації вхідних повідомлень та надання рекомендацій щодо їх надійності. Використання брендмауерів та фільтрів значно зменшує вірогідність стати жертвою вже «відомих» фішингових шахрайств. Брендмауери та фільтри можуть стати ефективним інструментом, який використовується для зменшення кількості шахрайських повідомлень, які може отримати потенційний користувач-жертва фішингової атаки. Існує також велика кількість антивірусних програм, які надають користувачам повідомлення з попередженнм про підозрілі веб-сайти. Разом з тим антивірусне програмне забезпечення посилює безпеку кінцевої точки і викорінює корисні для фішингової атаки дані з терміналу кінцевого користувача. Сучасні браузери все частіше попереджають своїх користувачів про небезпеку, коли вони вводять особисті дані на веб-сайт, який не захищений SSL.

Недоліками сучасного програмного забезпечення та/або технологічного покращення, яке направлене на протидію та розпізнавання фішинговим атакам, такими як брандмауери та фільтри, що запобігають настанню фішингової атаки шляхом блокування трафіку з підозрілих джерел та підтримки загальнодоступних в

мережі чорних списків, є те, що сучасне середовище для реалізації фішингу і їх способи є більш складними, що призводить до того, що алгоритми брандмауерів та фільтрів, склад їхніх баз даних та чорних списків можуть за цим сучасним середовищем не вспівати [15].

1.1.3 Підходи щодо виявлення фішингу

Садія Афроз і Рейчел Грінштадт у своєму дослідженні 2009 року [27] розділили підходи поточного виявлення фішингових атак на три основні типи:

- підходи, не засновані на вмісті, тобто які не використовують контент сайту з метою його класифікації на легітимний та/чи шахрайський;
- підходи, засновані на вмісті, тобто які використовують контент сайту з метою виявлення фішингу;
- підходи, засновані на візуальної схожості, тобто які використовують візуальну подібність підозрюваного сайту із відомими сайтами з метою виявлення фішингу.

Перший тип підходів, які не використовують вміст сайту з метою його визначення як законний чи зловмисний, включають автоматизовану класифікацію шахрайський вебсайтів, беручи до увагу інформацію про хости та URL-адреси, а також методи внесення підозрілих сайтів в чорний та білий список. У схемах, що класифікують сайти на основі URL-адрес, ці адреси розподіляються на основі як лексичних характеристик адреси, так і властивостей хоста.

Лексичні характеристики URL-адреси описують лексичні особливості підозрілих URL-адрес. Лексичні характеристики – це текстові властивості самої URL-адреси на відміну від контенту сторінки, на яку вона посилається. Як правило лексичні характеристики шкідливих адрес «виглядають інакше» в очах користувачів, які їх бачать. Лексичні характеристики включають довжину імені хоста, довжину всієї URL-адреси, а також кількість крапок в URL-адресі — все це ознаки, що мають реальне значення [28].

Під властивостями хоста URL-адреси маються на увазі [29]:

- властивості IP-адреси, які в тому числі можуть відповідати на питання: “Чи є IP-адреса в чорному списку?»;
- властивості WHOIS, які в тому числі можуть відповідати на питання: «Яка дата реєстрації, оновлення та термін дії домену?», «Хто є реєстратором домену?», «Хто є реєстрантом домену?», « Чи заблоковано запис WHOIS?»;
- властивості доменного імені, які в тому числі можуть відповідати на питання: «Яке значення TTL для записів DNS, пов'язаних з іменем хоста?», «Чи містить ім’я хоста ключові слова «клієнт» чи «сервер»?», «Чи є IP-адреса в імені хоста?»
- географічні властивості, які в тому числі можуть відповідати на питання: «До якого континенту/країни/міста належить IP-адреса?», «Яка швидкість висхідного підключення (широкосмугового, комутованого тощо)?».

На основі цих властивостей будується матриця, яка використовується декількома алгоритмами класифікації та у результатах випробування цих алгоритмів в реальному часі цей підхід має показники успіху в межах 95-99% [15]. У вищезгаданому дослідженні Садії Афроз і Рейчел Гріншадт 2009 року використовувалися як лексичні особливості URL-адреси, так і вміст сайту на разом з аналізом зображень, що в свою чергу мало підвищити продуктивність та зменшити кількість хибнопозитивних випадків.

У підходах, коли використовується чорний список, звіти користувачів або організацій досліджуються з метою виявлення шахрайських веб-сайтів, які зберігаються в базі даних чорних списків. Існує думка, що застосування цього підходу в комерційних панелях інструментів, як Microsoft Edge, CallingID Toolbar, Cloudmark Anti-Fraud Toolbar, EarthLink Toolbar, Netscape Browser 8.1 або GeoTrust TrustWatch Toolbar зіграло свою роль у набутті популярності методу серед інших підходів до протидії фішингу [27]. Недоліком підходу з використання чорних списків є те, що більшість фішингових сайтів нє орієнтовані на постійне використання і часто існують або менше ніж 20 годин [30], або повністю змінюють

URL-адреси, саме тому підхід із внесенням підозрілої URL-адреси у чорний список не може розпізнати більшу частину випадків фішингових атак. Більш того, підхід з чорним списком є загальним і орієнтованим на фонове використання, адже він не зможе виявити атаку, яка є цілеспрямованою на конкретного користувача, а особливо не зможе виявити ті атаки, які спрямовані на нешироко використовувані сайти, але прибуткові сайти, такі як невеликі брокерські контори, корпоративні мережі тощо [27].

З іншого боку існуючі підходи, які орієнтуються на створення подібного білого списку цілеспрямовані на детектування вже широковідомих легальних сайтів [31, 32], але це не означає, що користувач не повинен пам’ятати про перевірку інтерфейсу кожного разу, коли він відвідує будь-який сайт. Деякі існуючі підходи зі створення білого списку використовують перевірку на стороні сервера з метою додання додаткових показників автентифікації за межами SSL до клієнтських браузерів у якості доказу його доброякісної природи, наприклад, цей підхід реалізовують з використанням динамічних обкладинок безпеки [33] чи TrustBar [34].

Якщо розглядати підхід, який заснований на змісті сайту, то в ньому фішингові атаки напряму виявляються шляхом дослідження вмісту сайту. Ознаки, що використовуються в цьому підході можуть, включати в себе [15]:

- поля для вводу паролю;
- орфографічні помилки;
- джерела зображень;
- посилання, вбудовані посилання тощо.

Також у вищезгаданому підході розглядаються ознаки на основі URL-адреси та хосту. Розробки SpoofGuard [35] і CANTINA [36] є двома прикладами підходу на основі контенту. Як приклад можна також привести існуючий антифішинговий фільтр від компанії Google, який виявляє фішингові веб-сайти і зловмисне програмне забезпечення на основі дослідження URL-адреси сторінки та її лексичних ознак, рейтингу сторінки у пошуковій системі, інформації WHOIS та контенту сторінки, який включає в себе HTML, javascript, iframe, зображення тощо [37].

Алгоритм фільтру Google часто перенавчається на нових базах даних з фішинговими сайтами для того, щоб не стати недієздатним і дізнаватися про нові тенденції у способах створення фішингових сайтів. Цей алгоритм є високоточним, але наразі реалізований тільки в автономному режимі, адже для виявлення фішингової атаки потрібно 76 секунд в середньому. Деякі дослідники в своїх працях вивчали підходи засновані на основі нечіткої логіки та використання серії хешів веб-сайтів для ідентифікації фішингових сайтів [38]. Тим не менш, проведені експерименти, що стосуються підходу, заснованому на нечіткому хешуванні, показали, що він хоча і може ідентифікувати наявні атаки, але цю ідентифікацію можна легко уникнути, реструктуризувавши елементи HTML без зміни зовнішнього вигляду сайту [27].

Інструмент GoldPhish також реалізує підхід нечіткого хешування, а як пошукову систему використовує Google. Інструмент GoldPhish дає підвищує рейтинг добре закріплених і знаних веб-сайтів. Вже було помічено, що шахрайські веб-сторінки існують лише протягом невеликого періоду часу, це в результаті дає таким сайтам більш низький рейтинг під час інтернет-запитів, що робить їх основою для антифішингового підходу на основі вмісту [39]. Алгоритм роботи GoldPhish можна описати трьома ключовими кроками [27]:

1. створення знімку поточного веб-сайту користувача у його веб-браузері;
2. перетворення отриманого зображення в текстовий формат, який легко зможе читати комп’ютер, використовуючи оптичне розпізнавання символів.
3. введення тексту, отриманого на другому етапі, у пошукову систему для отримання і аналізу результів та оцінки рейтингу сторінки.

Цей інструмент не призводить до отримання помилкових спрацьовувань і надає захист нульового дня від фішингової атаки нульового дня, що є його основною перевагою. Недоліком інструменту GoldPhish же є час, необхідний для відтворення веб-сторінки. Він є також сприйнятливим до алгоритму Google PageRank і атак пошукових служб Google [39].

Третій тип підходів виявлення фішингових веб-сайтів, що заснований на візуальній подібності, включає оцінку візуальної схожості підозрілого веб-сайту та

інших за допомогою схожості стилю та макету (DOM) [40] і TrustPage [41], який використовує пошук Google і оцінку користувачів для визначення візуально подібної сторінки. В першому випадку візуальні функції, такі як рівень блоку (текст і зображення), схожість макету і загальний стиль (каскадна таблиця стилів) порівнюються з відповідними ознаками легітимного веб-сайту. Кожній функції призначаються ваги відповідно до пріоритету, який грає в розробці законного веб-сайту. Підозрілий веб-сайт класифікується як фішинговий, якщо його візуальна подібність перевищує деяке порогове значення, в іншому випадку підозрілий веб-сайт позначається як легітимний. Недоліком такої техніки є високий час відповіді, адже ця схема використовує і потребує велику достовірну базу даних зображень, в той час як візуальне порівняння підозрілого веб-сайту з елементами бази даних зображень не є швидким процесом [42].

Існують і інші підходи поточного виявлення фішингових атак, але вони не відносяться до виявлення фішингових сайтів. Наприклад, вони включають в себе виявлення фішингових електронних листів, що описано у роботі Яна Фети і його колег «Навчання для виявлення фішингових електронних листів» [43] та, як вже було описано у методах протидії фішингу, навчання користувачів, що описано.у роботі П. Кумарагуру «Система навчання користувачів семантичним атакам» [33].

1.2 Використання машинного навчання в кібербезпеці

Машинне навчання — це дисципліна, що є зосередженою на двох питаннях, які взаємопов’язані між собою. Перше питання стосується можливості побудувати такі комп’ютерні системи, які автоматично покращуються і розвиваються на основі свого ж досвіду. Друг питання стосується існування фундаментальних статистичних і обчислювально-інформаційно-теоретичних законів, що фактично керують усіма системами навчання, включаючи і комп’ютери, і людей, і організації. Поглиблине вивчення машинного навчання є важливим як для вирішення цих двух фундаментальних інженерних та наукових питань, так і для суто практичного

комп’ютерного програмного забезпечення, яке воно створило та застосовувало в багатьох програмах [44]. Простішими словами, машинне навчання — це наука про те, як змусити комп’ютер вчитися на своєму ж досвіді без фактично традиційного програмування, тобто без будь-якого втручання людини. Необхідно відзначити, що мета цієї науки — зробити машину «розумною». Оскільки найрозумніші істоти на Землі — це люди, можна сміливо сказати, що на даному етапі мета науки машинного навчання — зробити машину з таким інтелектом, який буде максимально наближений до інтелекту людини.

Машинне навчання можна спробувати загалом визначити як обчислювальні методи, що використовують досвід для підвищення рівня продуктивності або для створення точних прогнозів. В даному випадку досвід відноситься до вхідної попередньо вивченої інформації, доступної машині, тобто учню, яка зазвичай існує у формі електронних даних, щ обули цілеспрямовано зібрані і надані учню для аналізу. Цими даними можуть бути будь-який набір даних у формі оцифрованих навчальних наборів даних, попередньо класифікований і позначений людьми (кероване навчання), або набір даних з іншими типами інформації, отриманої шляхом взаємодії з навколишнім середовищем. У всіх випадках саме якість і розмір набору даних мають вирішальне значення для успіху передбачень, зроблених машиною-учнем.

Машинне навчання складається і з розробки і реалізації ефективних і точних алгоритмів прогнозування. Як і в інших сферах інформатики, кількома критичними показниками якості цих алгоритмів прогнозування є їхня просторова і часова складність. Потрібно зазначити, що в машинному навчанні користувачам додатково необхідно знати і розуміти поняття складності вибірки, щоб мати змогу оцінити розмір підготованої вибірки, необхідної для алгоритму для вивчення сімейства понять. Загалом, гарантії результативності теоретичного навчання для алгоритму машинного навчання напряму залежать від складності розглянутих класів концепції та розміру навчального набору даних.

Оскільки успіх алгоритму машинного навчання залежить саме від використовуваних даних, глибинне навчання стає невід’ємно пов’язаним зі

статистикою та аналізом даних. Загалом, навчання це методи, що керують даними, що поєднують фундаментальні концепції інформатики з ідеями зі статистики, ймовірності та оптимізації [45].

На сьогоднішній день існує різноманітний вибір алгоритмів машинного навчання, які зазвичай називаються моделлю. Вибір цієї конкретної моделі є окремою справою для різних проблем і задач і визначається він як характеристиками даних, так і типом бажаного результату. Основним значенням при виборі моделі може бути кількість унікальних точок даних на вході. Наприклад, якщо потрібно аналізувати набір даних, що містить більше 10^6 унікальних точок даних, то це означає, що в даному випадку можуть підійти більш екзотичні моделі машинного навчання. З іншого боку якщо потрібно аналізувати набір даних, менший за розміром, то це вказує на те, що в такому випадку стануть надійними і достовірними класичні методи, такі як, наприклад, лінійна регресія або методи дерева рішень. Ці алгоритми поділяють набори даних на різні класи відповідно до визначених користувачем фіксованих правил, і швидше за все будуть працювати краще. Необхідно також звернути увагу на те, що підхід потрібно пристосувати до характеристик вхідних даних, будь то сигнал часового ряду, набір зображень або загальні описові дані [46].

Якщо абстрактно уявити процес машинного навчання, то він обов’язково буде починатися з надання вхідних даних для моделі. Цей набір даних може варіюватися від більш простих і зрозумілих до аналізу даних, таких як текстові дані або числові, до більш складних даних, як, наприклад, зображення чи відеофайл. Цей набір даних потім буде зчитуватися електронно-обчислювальною машиною (комп’ютером) і послідовно оброблятися різноманітними математичними та/або статистичними алгоритмами. Після обробки електронно-обчислювальна машина відображає вхідні дані у вигляді інформації, прийнятною для машинного аналізу, прийняття рішень, ілюстрацій тощо. Повний процес машинного навчання може виглядати дуже просто для людини, але по суті перший і останній етапи — це лише елементи керування машинним навчанням, тоді як другий етап, що відповідає за обробку вхідних даних,

— це саме машинне навчання. Оскільки в машинному навчанні обробка даних означає використання методів статистики і математичних алгоритмів, то в залежності від типу вхідних даних і того, як ці дані в кінцевому підсумку мають виглядати як інформація, обираються найбільш пристосовані і цільові методи та алгоритми обробки даних. Тобто, саме в залежності від мети та завдання обирається, який алгоритм буде обраний і використаний для вирішення конкретно поставлених завдання та мети.

Фаза обробки даних майже у всіх випадках, крім найбільш тривіальних, — це знання або розуміння того, які користувач намагається отримати з ще необроблених алгоритмом машинного навчання даних. Наприклад, якщо перед користувачем стоїть завдання написати рецензію на книгу чи на будь що ще, то він не може цього зробити, прочитавши всього декілька ключових слів чи сторінок у книзі. Для цього користувач повинен прочитати цю книгу від початку до кінця і, зрозумівши повний зміст і суть книги, він може викласти суть інформації в короткому обсязі або огляді. Тобто в даному випадку книгу на двісті і більше сторінок можна легко стиснути до пари сторінок з найнеобхіднішими й основними ідеями цієї книги, що можна потім вважати повною і достатньою інформацією. Отже, мета використання машинного навчання — перетворити дані в інформацію [47].

Існують наступні загальні етапи розробки програми машинного навчання [48]:

1. збір інформації;
2. підготовка вхідних даних;
3. аналіз вхідних даних;
4. тренування алгоритму;
5. тестування алгоритму;
6. використання.

На етапі збору інформації можна зібрати зразки, наприклад, витягнувши цікаві для достлідження дані з веб-сайту, або можна отримати необхідну інформацію з RSS-каналу або API. Також можна налаштувати пристрій, який буде автоматично збирати виміри швидкості вітру та надсилати їх користувачу, або пристрій, який збирає показники рівню глюкози в крові, збирати можна все, що можливо виміряти

або використати з загальнодоступних джерел. Кількість варіантів збору інформації нескінченна.

На етапі підготовки вхідних даних потрібно переконатися, що вони знаходяться в придатному для використання алгоритмом форматі. Формат, який дуже часто використовується в задачах машинного навчання, — це список Python. Перевага такого стандартного формату полягає в тому, що завдяки списку Python можна змішувати і поєднувати алгоритми та джерела даних. Існує велика вірогідність, що користувачу може знадобитися виконати форматування своїх даних для використання їх певним алгоритму. Існують алгоритми, функції яких потребують дані спеціального формату, наприклад, деякі алгоритми мають справу з цільовими змінними та наборами даних у вигляді рядків, а деякі з іншого боку потребують, щоб дані знаходились у форматі цілих чисел.

Аналіз вхідних даних це по суті вивчення і перегляд даних з попереднього кроку. Треба перевірити, що задачі, поставлені перших двох кроках правильно виконані і алгоритм зможе працювати. Необхідно переконатися, що в наборі даних нема купи порожніх значень.

Саме на етапі тренування алгоритму відбувається машинне навчання. На цьому та наступному етапі використовуються основні алгоритми, залежно від обраної моделі. Варто зазначити, що даний етап існує тільки для керованого машинного навчання, у випадку некерованого машинного навчання етап опускається.

Етап тестування алгоритму означає перевірку його працездатності. Саме на цьому етапі використовується інформація, отримана на попередньому кроці. У випадку навчання з наглядом у користувача є деякі завідома відомі значення, які можна використовувати для оцінки роботи моделі. У навчанні без нагляду користувачу скоріш за все доведеться використовувати інші показники для оцінки результатів. У будь-якому випадку, якщо результати є незадовільними, користувач може повернутися до попереднього кроку, вдосконалити алгоритм та спробувати знову його протестувати. Дуже часто причина неефективної роботу моделі може

полягати у неправильному зборі або підготовці даних, тоді користувачу потрібно повернутися до першого кроку.

Після успішного виконання всіх вищеописаних кроків можна сміливо приступати до використання побудованої і протестованої моделі задля задовільнення своїх цілей і поставлених задач, наприклад, коли необхідно класифікувати великий масив даних [48].

Класифікація документів є далеко не єдиним навчальним завданням. Машинне навчання допускає дуже широкий набір практичних застосувань, які включають наступне [45]:

- класифікація тексту або документа;
- обробка природної мови;
- програми для обробки мовлення;
- програми комп'ютерного зору;
- програми обчислювальної біології та інші.

До задач класифікації тексту та документів входять такі проблеми, як призначення теми тексту або документу або автоматичне визначення того, чи є вміст веб-сторінки невідповідним або занадто відвертим; він також включає виявлення спаму.

Більшість завдань у галузі природньої мови, включаючи тегування частини мови, розпізнавання іменованих об’єктів, безконтекстний синтаксичний аналіз або аналіз залежностей, теж розглядаються як проблеми навчання. У цих задачах передбачення допускають певну структурність. Наприклад, тегування частини мови передбачення для речення — це послідовність тегів частини мови, що позначають кожне слово. У безконтекстному розборі передбачення є деревом. Це відомі приклади більших проблем навчання як задачі структурованого прогнозування.

Програми для обробки мовлення включають розпізнавання мовлення, синтез мовлення, перевірку мовця, ідентифікацію мовця, а також підпроблеми, такі як мовне моделювання та акустичне моделювання.

Програми комп’ютерного зору включають розпізнавання об’єктів, ідентифікацію об’єктів, виявлення обличчя, оптичне розпізнавання символів, пошук зображень на основі вмісту або оцінку пози.

Програми обчислювальної біології включають в себе прогнозування функції білка, ідентифікацію ключових сайтів або аналіз генних і білкових мереж.

Багато інших проблем, таких як виявлення шахрайства з кредитними картками, телефонними або страховими компаніями, вторгнення в мережу, навчання гри в ігри, такі як шахи або нарди, керування транспортними засобами, медична діагностика, розробка рекомендаційних систем, пошукових систем або систем вилучення інформації вирішуються за допомогою методів машинного навчання.

Цей список аж ніяк не є вичерпним. Більшість проблем прогнозування, які зустрічаються на практиці, можна відобразити як проблеми навчання, і область практичного застосування машинного навчання постійно розширюється. Алгоритми та методи, які розглядаються в цій книзі, можна використовувати для отримання рішень для всіх цих проблем, хоча ми не будемо детально обговорювати ці програми [45].

Перевага машинного навчання перед людським інтелектом полягає в тому, що машина може знаходити приховані зв’язки, які можуть критично вплинути на результат результату. Тобто інформація машинного навчання може відрізнятися від інформації, яку ми очікуємо отримати. Ця перевага дуже корисна при обробці великих даних, де людині дуже важко знайти прості зв’язки, не кажучи вже про те, що бувають і приховані, але у більшості випадків машинне навчання просто використовується для великих даних, щоб зменшити трудові та матеріальні ресурси [47].

1.2.1 Керований та некерований підхід у машинному навчанні

Методи машинного навчання, як правило, поділяються на контрольовані та неконтрольовані. Кероване машинне навчання починається з попереднього знання бажаного результату у вигляді позначених наборів даних, що дозволяє керувати

процесом навчання, тоді як машинне навчання без нагляду працює безпосередньо з немаркованими даними. За відсутності міток для орієнтації процесу навчання ці мітки повинні бути «відкриті» алгоритмом навчання [49]. На рисунку 1.4 представлений концепт контрольованого машинного навчання [50].

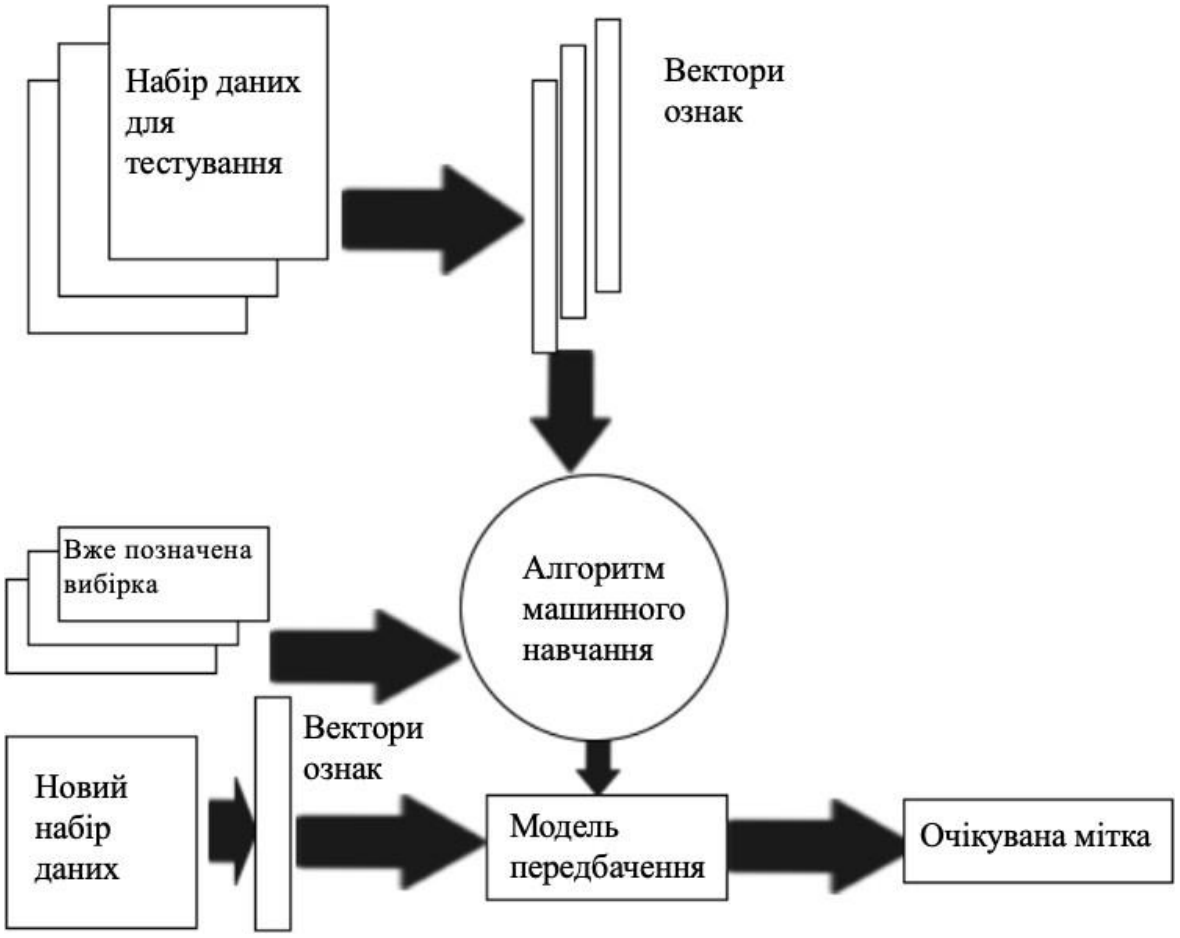

Рисунок 1.4 – Концепт контрольованого машинного навчання

Контрольований сценарій навчання, або навчання з вчителем, характеризується концепцією викладача або супервізора, головне завдання якого — надати агенту точну міру його помилки (безпосередньо порівнянну з вихідними значеннями). У реальних алгоритмах ця функція забезпечується навчальним набором, що складається з пар (вхідний та очікуваний вихід). Виходячи з цієї інформації, агент може коригувати свої параметри, щоб зменшити величину глобальної функції втрат. Після кожної ітерації, якщо алгоритм досить гнучкий і

елементи даних є узгодженими, загальна точність зростає, а різниця між прогнозованим і очікуваним значенням стає близькою до нуля. Звичайно, у сценарії з наглядом метою є навчання системи, яка також повинна працювати зі зразками, яких раніше не було. Отже, необхідно дозволити моделі розвивати здатність узагальнювати і уникати поширеної проблеми, яка називається переобладнанням, яка викликає перенавчання через надмірну здатність.

Кероване навчання передбачає вивчення різниці між набором вхідних змінних X і вихідної змінної Y і застосування цього відображення для прогнозування вихідних даних для ще не введених даних. Кероване навчання є найважливішою методологією машинного навчання, а також має центральне значення в обробці мультимедійних даних [51].

Розглянемо ситуацію, коли дві різні моделі навчаються з однаковими наборами даних (двом більшим рядкам відповідно). Навчальне завдання буде називатись навчанням із наглядом, якщо в цих наборах даних буде присутня якась явна ознака класифікації, наприклад, значення "Так" і "Ні" [52].

Іноді замість прогнозування фактичної категорії краще визначити її розподіл ймовірностей. Наприклад, алгоритм можна навчити розпізнавати рукописну букву алфавіту, тому її вихід є категоричним (англійською мовою буде 26 дозволених символів). З іншого боку, навіть для людей такий процес може призвести до більш ніж одного ймовірного результату, коли візуальне представлення букви недостатньо чітке, щоб належати до однієї категорії. Це означає, що фактичний результат краще описується дискретним розподілом ймовірностей (наприклад, з 26 безперервними значеннями, нормалізованими так, щоб вони завжди сумували до 1).

На рисунку 1.5 наведено приклад класифікації елементів з двома ознаками. Більшість алгоритмів намагаються знайти найкращу розділову гіперплощину (в даному випадку це лінійна задача), накладаючи різні умови. Однак мета завжди одна: зменшення кількості помилкових класифікацій та підвищення шумостійкості. Наприклад, якщо розглянути трикутну точку, яка знаходиться ближче до площини (її координати приблизно [5,1 - 3,0]), то, якщо на величину другої характеристики

вплинув би шум, і тому значення було досить меншим за 3,0, трохи вища гіперплощина могла б неправильно класифікувати точку [53].

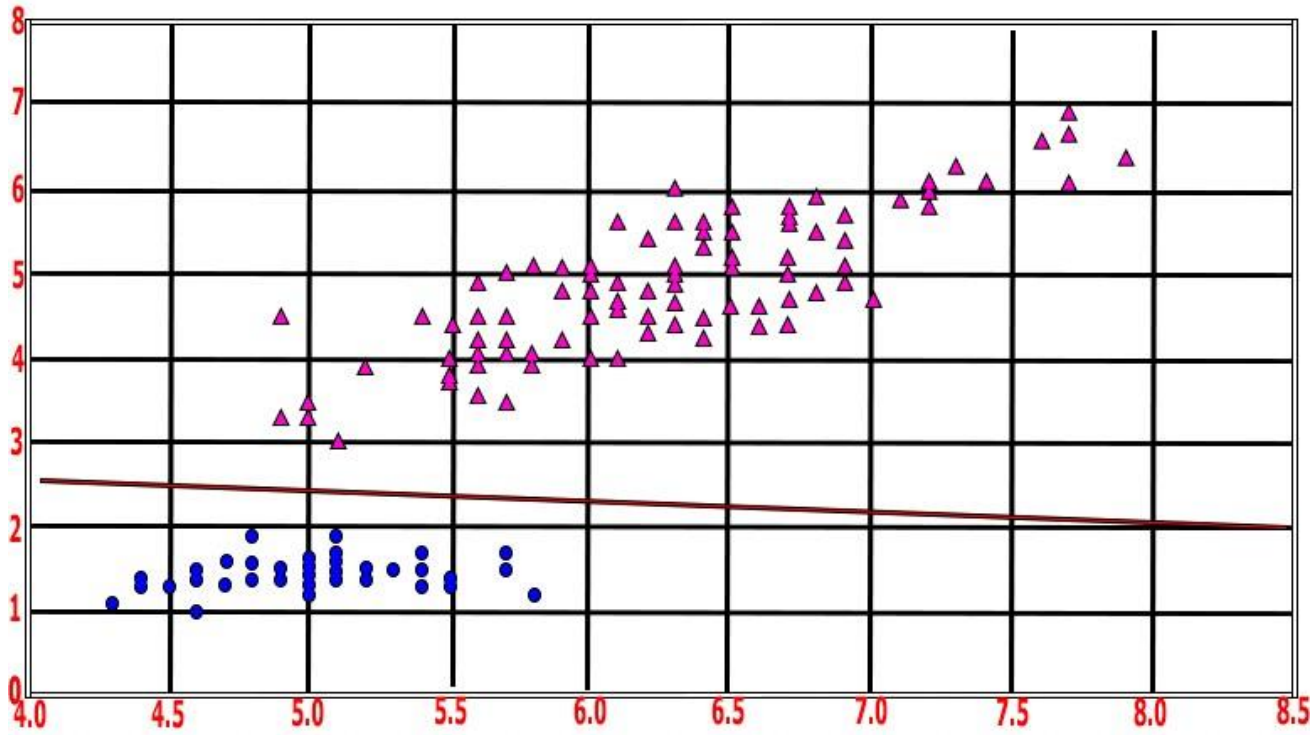

Рисунок 1.5 – Приклад класифікації елементів з двома ознаками

Найбільш широко використовувані та популярні алгоритми навчання з учителем [54]:

- метод опорних векторів;
- лінійна регресія;
- логістична регресія;
- наївний байєсівський класифікатор;
- навчання дерева рішень;
- метод k-найближчих сусідів;
- штучна нейронна мережа;
- вивчення подібності.

Кожні вищезгадані алгоритми мають різні підходи математичних та статистичних математичних методів та формул. Але можна підкреслити загальний шаблон алгоритму, тому що всі ці алгоритми належать до навчання з учителем:

Навчальний набір даних складається з n упорядкованих пар $(x_1, y_1), (x_2, y_2) \ldots (x_n, y_n)$, в яких кожне x_i вимірювання або набір вимірювань однієї точки даних, а y_i – відповідь для цієї точки даних. Наприклад, x_i може бути групою з п'яти вимірювань для пацієнта у лікарні, включаючи ріст, вагу, температуру, рівень цукру в крові та кров'яний тиск. Відповідний y_i може бути класифікацією пацієнта як "здоровий" або "хворий".

Тестові дані в контрольованому навчанні – це ще один набір вимірювань m без відповіді: $(x_{n+1}, x_{n+2}, \ldots, x_{n+m}) \cdot (x_{(n+1)}, x_{(n+2)}, \ldots, x_{(n+m)})$. Як описано вище, мета полягає в тому, щоб зробити обґрунтовані припущення відповіді для тестових даних (наприклад, "здоровий" або "хворий"), роблячи висновки з побудованої шаблонної моделі, який був створений для обробки даних, що навчаються.

Кероване машинне навчання найчастіше здійснюється при [55]:

- автоматичної класифікації зображень;
- автоматичної обробки послідовності (наприклад, музики або мови);
- прогнозного аналізу на основі регресії або категорійної класифікації;
- виявлення спаму;
- виявлення шаблонів;
- обробки природної мови;
- аналізу настроїв.

Навчання без вчителя ґрунтується на відсутності будь-якого керівника, а отже, і на абсолютних показниках похибки. Під час навчання без нагляду машина отримує вхідні дані, але не отримує контрольованих цільових результатів. Може здатися нереальним уявити, чого може дізнатися машина, не отримуючи жодного зворотного зв’язку з оточенням, але у певному сенсі навчання без нагляду можна розглядати як пошук закономірностей у даних, що виходять за межі того, що можна вважати чистим неструктурованим шумом. Двома дуже простими класичними прикладами неконтрольованого навчання є кластеризація та зменшення розмірності [56].

На рисунку 1.6 представлений концепт неконтрольованого машинного навчання [50].

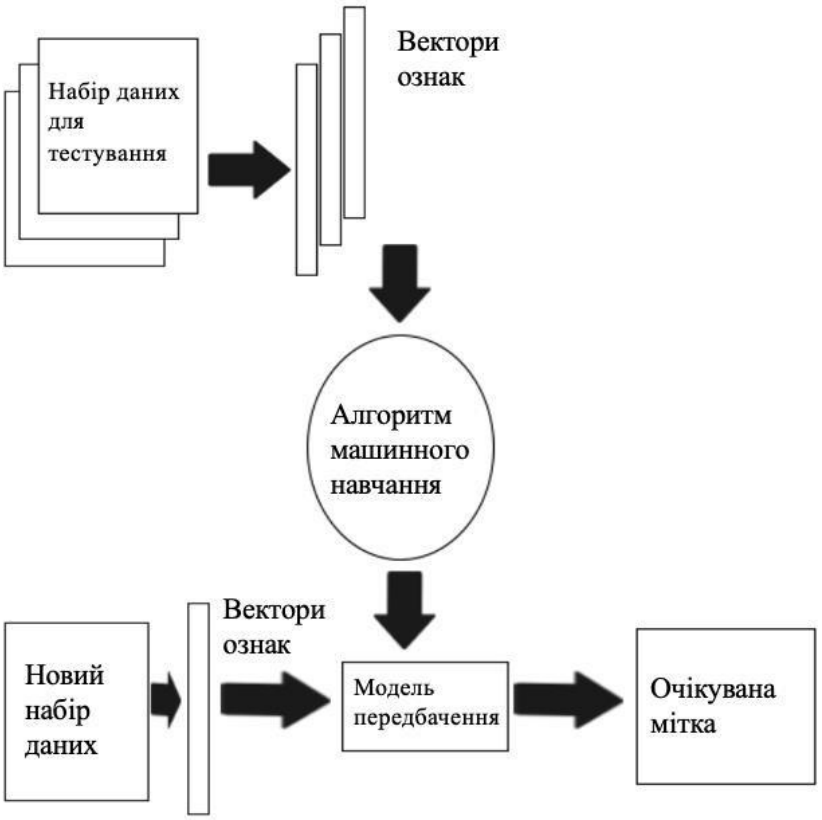

Рисунок 1.6 – Концепція некерованого машинного навчання

Навчання без вчителя є дуже корисним, коли необхідно дізнатися, як набір елементів можна згрупувати відповідно до їхньої схожості (або міри відстані). Наприклад, дивлячись на рисунок 1.5, людина може відразу визначити два набори, не враховуючи кольори чи форми. Фактично, кругові точки (як і трикутні) визначають узгоджений набір; він відокремлений від іншого набагато більше, ніж те, як його точки внутрішньо відокремлені, як море з декількома островами

На рисунку 1.7 кожен еліпс представляє кластер, і всі точки всередині його області можна позначити таким же чином. Існують також граничні точки (наприклад, трикутники, що перекривають область кола), які потребують певного критерію (як правило, компромісної міри відстані), щоб визначити відповідний кластер. Так само, як і для класифікації з неоднозначністю, хороший підхід до кластеризації повинен враховувати наявність викидів і розглядати їх так, щоб

підвищити внутрішню узгодженість (візуально це означає вибрати підрозділ, який максимізує локальну щільність), і поділ між кластерами.

Наприклад, можна надати пріоритет відстані між окремою точкою та центроїдою, або середній відстані між точками, що належать до одного й різних кластерів. На цьому малюнку всі граничні трикутники близькі один до одного, тому найближчим сусідом є інший трикутник. Однак у реальних задачах часто є граничні області, де є часткове перекриття, що означає, що деякі точки мають високий ступінь невизначеності через їх значення характеристик.

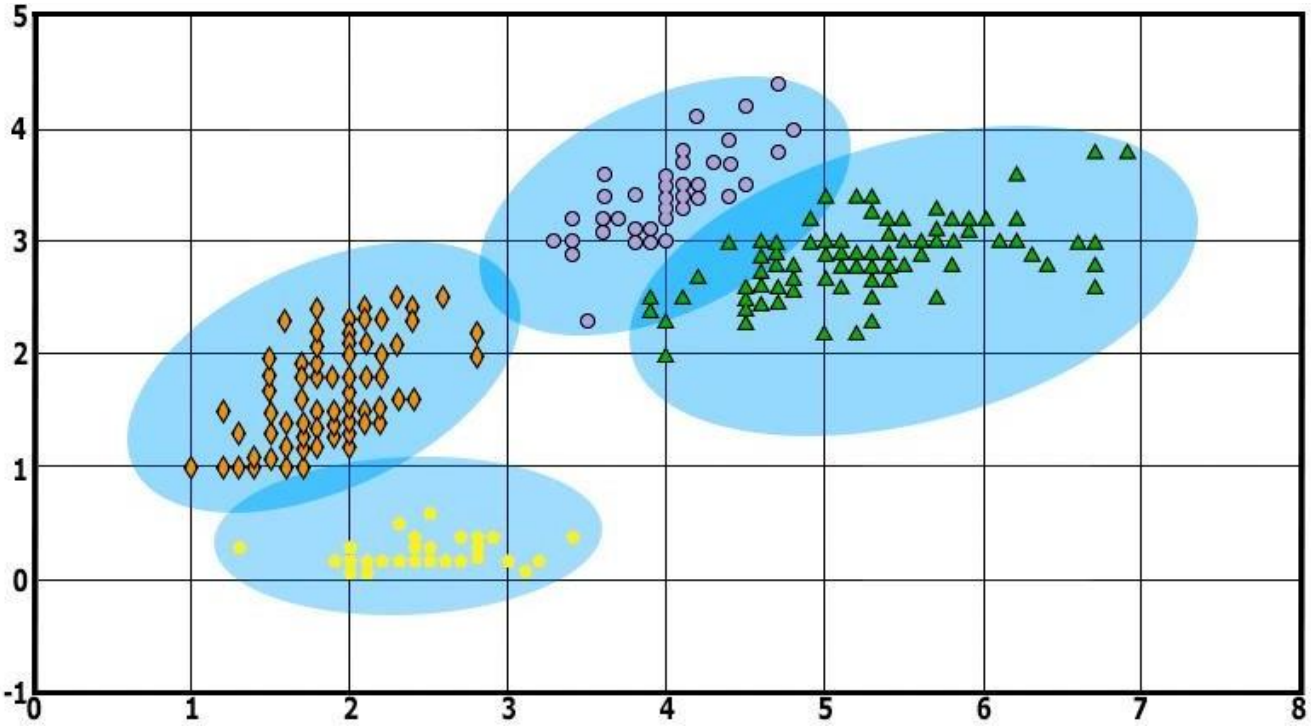

Рисунок 1.7 – Приклад кластеризації елементів за умови, що кількість кластерів рівна 4

Інша інтерпретація може бути виражена за допомогою розподілу ймовірностей. Якщо подивитися на еліпси, то можна побачити, що вони представляють область багатовимірних гауссів, пов’язаних між мінімальною та максимальною дисперсією. Розглядаючи весь домен, точка (наприклад, блакитна зірка) потенційно може належати до всіх скупчень, але ймовірність, задана першим (нижній лівий кут), є найвищою, і тому це визначає членство. Як тільки дисперсія і середнє значення (іншими словами, форма) всіх гауссів стають стабільними, кожна

гранична точка автоматично фіксується одним гауссовим розподілом (крім випадку рівних ймовірностей). Технічно йдеться на увазі, що такий підхід максимізує ймовірність гаусової суміші за певного набору даних. Це дуже важлива концепція статистичного навчання, яка охоплює багато різних застосувань.

Інші важливі методи передбачають використання як позначених, так і немаркованих даних. Тому цей підхід називається напівкерованим і може бути прийнятим, коли необхідно класифікувати велику кількість даних за допомогою кількох повних (позначених) прикладів або коли є потреба накласти деякі обмеження на алгоритм кластеризації (наприклад, призначити деякі елементи до певного кластера або виключаючи інші) [53].

Поширені неконтрольовані програми включають [53]:

• сегментацію об’єктів (наприклад, користувачів, продуктів, фільмів, пісень тощо);

• виявлення подібностей;

• автоматичне маркування.

1.2.2 Класифікація, кластеризація та регресія в машинному навчанні

Машинне навчання також можна класифікувати на основі необхідних результатів:

- класифікація;
- кластеризація;
- регресія.

За допомогою різних підходів машинного навчання, враховуючи деякі конкретні атмосферні вимірювання сьогодні та вчора, користувачі можуть передбачити рівень озону, наприклад, завтра. В той же час, враховуючи значення градації сірого для пікселів оцифрованого зображення рукописної цифри, користувачі можуть передбачити її мітку класу.

Ця відмінність у типі результатів привела до конвенції про іменування для завдань прогнозування: регресія, коли ми прогнозуємо кількісні результати, і

класифікація, коли ми прогнозуємо якісні результати. Ми побачимо, що ці два завдання мають багато спільного, і зокрема обидва можна розглядати як завдання в наближенні функції [57].

Алгоритми регресії зазвичай застосовуються для статистичного аналізу. Регресія допомагає аналізувати модельні відносини між точками даних, а також кількісно визначати силу кореляції між змінними набором даних. Крім того, регресійний аналіз може бути корисним для прогнозування майбутніх значень даних на основі історичних значень. Однак важливо пам'ятати, що регресійний аналіз передбачає, що кореляція пов'язана з причинно-наслідковим зв'язком. Без розуміння контексту навколо даних регресійний аналіз може призвести до неточних прогнозів. Лінійний регресійний аналіз можна розділити на просту лінійну регресю і множинну лінійну регресію [58].

Класифікація в свою чергу є однією з найбільш часто зустрічаються завдань прийняття рішень людської діяльності. Проблема класифікації виникає, коли об’єкт потрібно віднести до попередньо визначеної групи або класу на основі ряду спостережуваних атрибутів, пов’язаних з цим об’єктом. Існує багато промислових проблем, визначених як проблеми класифікації. Наприклад, передбачення фондового ринку, прогноз погоди, передбачення банкрутства, медична діагностика, розпізнавання мовлення, розпізнавання символів тощо [59]. Ці задачі класифікації можна розв’язувати як математично, так і нелінійно.

Варто зазначити, що в машинному навчанні види класифікації робитимуться за типами алгоритмів, які так чи інакше належать до схем класифікації. Найбільш широко використовувані алгоритми навчання [60]:

- лінійна класифікація;
- метод опорних векторів;
- логістична регресія;
- наївний байєсівський класифікатор;
- штучна нейронна мережа;
- дерево прийняття рішень.

Лінійні моделі для класифікації розділяють вхідні вектори на класи за допомогою лінійних (гіперплощинних) меж рішень [62]. Даний алгоритм часто використовується в ситуаціях, коли швидкість класифікації є проблемою, оскільки він вважається найшвидшим класифікатором. Крім того, лінійні класифікатори часто працюють дуже добре, коли кількість вимірів велика, як у класифікації документів, де кожен елемент, як правило, є числом слова в документі [62].

Метод опорних векторів (SVM) – це найновіша методика машинного навчання з наглядом [63]. Моделі опорних векторних машин (SVM) тісно пов’язані з класичними багатошаровими нейронними мережами. SVM обертаються навколо поняття «маржа» — , що є будь-якою стороною гіперплощини, яка розділяє два класи даних. Максимізуючи запас і тим самим створюючи максимально можливу відстань між відокремленнями було доведено, що гіперплощина та екземпляри по обидва боки від неї зменшують верхню межу очікуваної помилки узагальнення [64].

Логістична регресія використовує мультиноміальну логістичну модель регресії з одним оцінювачем. Алгоритм зазвичай стверджує, де існує кордон між класами, а також стверджує, що ймовірності класів залежать від відстані від кордону, у конкретному підході. Ці твердження про ймовірності, які роблять логістичну регресію більше, ніж просто класифікатором. Він дає сильніші, більш детальні прогнози і може бути підігнаний по-різному; але ці сильні прогнози можуть бути помилковими [65].

Наївні байєсівські мережі – це дуже прості мережі, які складаються з орієнтованих ациклічних графів лише з одним батьком (що представляють неспостережуваний вузол) і кількома дочірніми (відповідними спостережуваним вузлам) із сильним припущенням незалежності між дочірніми вузлами в контексту їхнього батька [66]. Класифікатори Байєса зазвичай менш точні, ніж інші більш складні алгоритми навчання, іноді і перевершують ефективність інших схемм навчання, навіть на наборах даних із суттєвою залежністю функцій [67]. Класифікатор Байєса має проблему незалежності атрибутів, яка була вирішена за допомогою усереднених оцінок однієї залежності [68].

Нейронні мережі [69] фактично можуть виконувати декілька завдань регресії та/або класифікації одночасно, хоча зазвичай кожна мережа виконує лише одне. Таким чином, у переважній більшості випадків мережа матиме одну вихідну змінну, хоча у випадку проблем з класифікацією багатьох станів це може відповідати кільком вихідним одиницям (етап постобробки піклується про відображення з вихідні одиниці до вихідних змінних). Штучна нейронна мережа (ANN) залежить від трьох фундаментальних аспектів, функцій введення та активації пристрою, архітектури мережі та ваги кожного вхідного з'єднання. Враховуючи, що перші два аспекти фіксовані, поведінка штучної нейронної мережі визначається поточними значеннями ваг. Вага сітки, яку потрібно тренувати, спочатку встановлюється на випадкові значення, а потім екземпляри тренувального набору повторно відображаються в мережі. Значення для входу екземпляра розміщуються на вхідних одиницях, а вихід мережі порівнюється з бажаним виходом для цього екземпляра. Потім усі вагові коефіцієнти в мережі трохи коригуються в напрямку, який наблизить вихідні значення мережі до значень для бажаного результату. Існує кілька алгоритмів, за допомогою яких можна навчати мережу [70].

Дерева рішень: дерева рішень — це дерева, які класифікують екземпляри шляхом їх сортування на основі значень ознак. Кожен вузол дерева рішень представляє ознаку в екземплярі, який підлягає класифікації, а кожна гілка представляє значення, яке може прийняти вузол. Екземпляри класифікуються, починаючи з кореневого вузла, і сортуються на основі їхніх значень ознак [64]. Більш описові назви для таких моделей дерев — дерева класифікації або дерева регресії [71]. Будь-який вузол можна видалити і призначити йому найпоширеніший клас навчальних екземплярів, які сортуються [64].

Методи машинного навчання під наглядом застосовні в багатьох областях. Ряд прикладних робіт, орієнтованих на машинне навчання (ML), можна знайти в [72], [73].

Як правило, SVM і нейронні мережі, як правило, працюють набагато краще, коли мають справу з багатовимірністю і безперервними функціями. З іншого боку, системи, засновані на логіці, як правило, працюють краще, коли мають справу з

дискретними/категоріальними ознаками. Для моделей нейронних мереж і SVM потрібен великий розмір вибірки, щоб досягти максимальної точності передбачення, тоді як NB може знадобитися відносно невеликий набір даних.

Існує загальна згода, що KNN дуже чутливий до невідповідних ознак: цю характеристику можна пояснити тим, як працює алгоритм. Більше того, наявність невідповідних функцій може зробити навіть навчання нейронної мережі дуже неефективним і непрактичним. Більшість алгоритмів дерева рішень не можуть добре працювати з проблемами, які вимагають діагонального розбиття. Поділ простору екземплярів ортогональний до осі однієї змінної і паралельний усім іншим осям. Таким чином, усі отримані області після розбиття є гіперпрямокутниками. ANN і SVM добре працюють, коли присутня мультиколінеарність і існує нелінійний зв’язок між вхідними та вихідними характеристиками [60].

Кластеризація — це підхід некерованого машинного навчання, який використовуються для класифікації спостережень у наборі даних на кілька груп на основі їх схожості. Алгоритми вимагають від аналітика вказати кількість кластерів, які будуть створені [74]. Кластеризацію можна поділити на м’яку (перекриваючу кластеризацію) і жорстку кластеризацію (або виняткову кластеризацію):

У випадку методів м’якої кластеризації використовуються нечіткі набори, так що кожна точка може належати до двох або більше кластерів з різним ступенем належності. У цьому випадку дані будуть пов’язані з відповідним значенням членства. У багатьох ситуаціях нечітка кластеризація є більш природною, ніж жорстка. Навпаки, у методах жорсткої кластеризації дані групуються ексклюзивним чином, так що якщо певна дата належить до певного кластеру, вона не може включити в інший кластер.

Нечітка кластеризація C-середніх (FCM) є дуже популярною технікою м’якої кластеризації, а алгоритм K-середніх є важливою технікою жорсткої кластеризації [75].

Алгоритм K-середніх відповідає простому та легкому способу класифікації даного набору даних за допомогою певної кількості кластерів (припустимо k кластерів), фіксованих апріорно. Алгоритм K-середніх використовується, коли

мічені дані недоступні [76]. Загальний метод перетворення. грубі практичні правила в дуже точне правило прогнозування. Враховуючи «слабкий» алгоритм навчання, який може послідовно знаходити класифікатори («правила великого пальця») принаймні трохи краще, ніж випадкова, скажімо, точність 55%, з достатньою кількістю даних, алгоритм посилення може доказово побудувати один класифікатор з дуже високою точністю, скажімо , 99% [77].

Важливо зазначити, що з кластеризації, особливо у навчанні без вчителя, алгоритм шукає зв'язок між вхідними даними. Принадність машинного навчання - це пошук прихованих зв'язків між даними, більш відомі як латентні зв'язки. Для кластеризації при пошуку латентних зв'язків, використовується модель прихованих змінних, яка застосовується для вивчення взаємозв'язків між значеннями змінних.

1.2.3 Нечітка кластеризація як метод некерованного машинного навчання

Під прийняттям рішень у нечіткому середовищі розуміють процес прийняття рішень, у якому цілі та/або обмеження, але не обов’язково контрольована система, мають нечіткий характер, адже велика частина прийняття рішень у реальному світі відбувається в середовищі, в якій невідомі точно цілі, обмеження та наслідки можливих дій.

Для кількісної боротьби з неточністю зазвичай використовуються концепції та методи теорії ймовірностей, вважаючи, що неточність — незалежно від її природи — можна ототожнювати з випадковістю. Стверджується, що існує потреба в диференціації між випадковістю та нечіткістю, причому остання є основним джерелом неточності в багатьох процесах прийняття рішень [78].

Неточності у системах прийняття рішень нівелюють за допомогою різноманітних методів і моделей штучного інтелекту. Для того, щоб прийняття рішень у випадку, коли наявна невизначеність умов функціонування системи, була резальтативною, застосовуються методи на основі правил нечіткої логіки. В таких методах використовуються лінгвістичні величини і висловлювання з метою опису стратегій прийняття рішень, а самі вони основуються на нечітких множинах [79].

Прикладом нечітких обмежень є: «Вартість А не повинна бути суттєво вищою за α», де α — задана константа. Аналогічно, прикладом нечіткої цілі є: «х має бути поблизу x_0 де x_0 константа. Слова «поблизу» та «суттєво» є джерелами нечіткості в цих прикладах.

Нечіткі цілі та нечіткі обмеження можна точно визначити як нечіткі множини в просторі альтернатив. Таким чином, нечіткі рішення можна розглядати як перетин заданих цілей і обмежень. Максимізуюче рішення визначається як точка в просторі альтернатив, в якій функція належності нечіткого рішення досягає свого максимального значення.

Під нечіткістю мається на увазі тип неточності, який асоціюється з нечіткими множинами [80], [81], тобто класами, в яких немає різкого переходу від належності до нечленства. Прикладом нечіткої множини, наприклад, може бути клас рожевих об’єктів. Так само класи об’єктів можуть характеризуватися такими загальними прикметниками, як великий, малий, істотний, значний, важливий, серйозний, простий, точний, наближений тощо. На відміну від поняття класу чи множини в математиці, більшість класів у реальному світі не мають чітких меж, які відокремлюють ті об’єкти, які належать до класу, від тих, які не належать. Можна стверджувати, що головна відмінність між людським і машинним інтелектом полягає в здатності людей маніпулювати нечіткими поняттями та реагувати на нечіткі інструкції.

Різниця між випадковістю та нечіткістю полягає в тому, що випадковість пов’язана з невизначеністю щодо належності чи неналежності об’єкта до чіткої множини. Нечіткість, з іншого боку, пов’язана з класами, в яких можуть бути проміжні ступені членства між повним членством і нечленством.

Розуміючи цю різницю, математичні методи боротьби з нечіткістю відрізняються від теорії ймовірностей. Замість міри ймовірності в теорії ймовірностей в теорії нечіткості є більш просте поняття функції належності. Крім того, натомість a + b і ab, де a і b є дійсні числа, є простіші операції Max(a, b) і Min(a, b).

Нечітка мета може полягати в тому, щоб зробити x_3 приблизно рівним 5, починаючи з початкового стану $x_0 = 1$. Тоді задача полягає в тому, щоб знайти послідовність входів U_0, U_1, U_2 яка реалізує задану мету якомога ближче, з урахуванням зазначених обмежень на U_0, U_1, U_2.

Методи теорії нечітких множин дуже зручно використовувати при проектуванні інтерфейсів у людиномашинних системах. На базі нечіткого логічного виведення створюються системи керування, надання знань, підтримки прийняття рішень, параметричної та структурної ідентифікації, розпізнавання об`єктів, оптимізації. Нечітка логіка може знайти собі застосування у діагностиці, побутовій електроніці, різних експертних системах. Нечіткі експертні системи, які використовуються для підтримки прийняття рішень мають широку популярність у медицині, військовій справі та економіці. Саме з допомогою нечітких експертних систем здійснюють бізнес-прогнозування, проводять оцінку ризиків та прибутковості інвестиційних проектів, до того ж досліджують глобальні політичні рішення та моделюють кризові ситуації саме на основі нечіткої логіки [79].

Принципи і саму ідею нечіткої логіки використовують для кластеризації багатовимірних даних, шляхом призначення кожній точці належності до кожного центру кластера від 0 до 100 відсотків. Такий підхід може бути дуже потужним, якщо зрівнювати з традиційною жорсткою програмною кластеризацією, при якій кожній точці призначається точна, чітка мітка. Цей алгоритм працює шляхом призначення членства кожній точці даних, що відповідає кожному центру кластера, на основі відстані між центром кластера та точкою даних. Чим ближче дані до кластерного центру, тим більшою є приналежність цих даних до конкретного кластерного центру. Підсумок чисел приналежності точки даних до кожного кластеру має дорівнювати одиниці.

Так відбувається неконтрольована модель кластеризації, який дозволяє будувати нечіткий поділ даних. Робота алгоритму залежить від параметра U, що відповідає за ступінь нечіткості рішення. Великі значення параметру U розмивають класи, і тоді всі елементи, як наслідок, належать до всіх кластерів. Від параметра U

залежать розв’язки оптимізаційної задачі, що означає, що різний вибір кількості кластерів с зазвичай призведе до різних розділів [82].

Щоб краще розуміти нечітку кластеризацію, треба розуміти концепції жорсткої кластеризації та взагалі розуміти, як працюють їх алгоритми.

Алгоритми кластеризації зазвичай можна згрупувати в 3 категорії: ієрархічні, неієрархічні (плоскі) та змішані. Хоча існують сотні алгоритмів, на практиці використання багатьох із цих алгоритмів було обмеженим через їх складність, ефективність та доступність у статистичному програмному забезпеченні, яке зараз використовується [83]. Вибір хорошого алгоритму для виконання на певному наборі даних залежить від багатьох критеріїв, таких як розмір даних, структура даних і цілі кластерного аналізу [84],[85]. До того ж, як повідомлялося в багатьох дослідженнях, неієрархічні алгоритми розподілу, тобто алгоритми, що належать до сімейства K-середніх, дають хороші результати кластеризації за коротший час порівняно з ієрархічними алгоритмами. на великих наборах даних [86], [87].

Тому з моменту введення МакКуїном у 1967 алгоритму K-середніх, його похідні були найпопулярнішими алгоритмами в дослідницькому аналізі даних і додатках дата майнінгу аж протягом півстоліття.

Алгоритм жорстких C-середніх, або K-середніх, є широко відомим алгоритмом кластеризації, він ділить вхідних набір даних на c (або k) кластери. Параметр с задає кількість кластерів, вони мають бути відомі або попередньо визначені як фіксоване значення перед тим, як модель переходить до кластерного аналізу. В багатьох дослідженнях повідомлялося, що модель K-середніх дає порівняно хороші результати, при умові, коли кластери в наборах даних добре відрізняються або добре розділені.

Алгоритм K-середніх, або жорстких C-середніх, також має низку недоліків, які відносяться до розсіювання кластерів у наборах даних та форми. По-перше, модель може бути невдалою у пошуку кластерів, які перекриваються, також вона не є інваріантною до нелінійних перетворень даних. Через це представлення певного набору даних полярними координатами та з декартовими координатами можуть

видати різні результати кластеризації. В моделі К-середніх в тому числі не вдастся об’єднати нелінійні набори даних і дані з шумом.

Щоб подолати деякі проблеми, з якими стикається алгоритм К-середніх, Джордж Бездек у 1981 ввів нечіткі С-засоби, які засновані на дослідженні 1973 року [88] як розширення алгоритму К-середніх. Однак основний алгоритм С-середніх часто використовується в широкій області застосувань від інженерії до економіки. Алгоритм С-середніх — це м’який алгоритм кластеризації нечітких даних, у якому об’єкт є не тільки членом кластера, але й членом багатьох кластерів із різним ступенем належності. Таким чином, об’єкти, розташовані на кордонах кластерів, не змушені повністю належати до певного кластеру, а можуть бути членами багатьох кластерів із частковим ступенем належності від 0 до 1. Хоча алгоритм С-середніх (FCM) вважається більш ефективним для аналізу нечітких даних, він не має постійної переваги у всіх випадках структур даних [89].

Нехай $\boldsymbol{X} = \{x_1, x_2, \ldots, x_n\}$ – наданий набор даних для аналізу, $\boldsymbol{V} = \{v_1, v_2, \ldots, v_c\}$ — множина центрів кластерів у наборі даних $\boldsymbol{X}$ у p-нескінченному просторі $\mathbb{R}^p$, де n – кількість об’єктів, p – кількість ознак, а c – кількість кластерів.

Кластери описуються об'єктами-членами та їхніми центрами. Зазвичай центроїди використовуються як центри кластерів. Центроїд кожного кластера є точкою, до якої мінімізується сума відстаней від усіх об’єктів у цьому кластері. За допомогою алгоритму розбиття кластерів $\boldsymbol{X}$ розбивається на с кластерів з метою отримання низької внутрішньокластерної та високої неоднорідності між кластерами. Тобто кластер складається з об’єктів, які знаходяться якомога ближче один до одного і якомога далі від об’єктів інших кластерів. Залежно від областей дослідження, набір даних $\boldsymbol{X}$ формується з точками даних, які є представленнями об’єктів, якими можуть бути особи, спостереження, випадки, вимоги, пікселі тощо.

У той час як алгоритми жорсткої кластеризації, такі як К-середніх, призначають кожен об’єкт точно одному кластеру, алгоритми м’якого розділення або нечіткої кластеризації, як-от С-середніх, призначають кожен об’єкт різним кластерам з різним ступенем членства, як зазначено вище. Іншими словами, хоча

членство в кластері дорівнює 0 або 1 у К-середніх, воно коливається від 0 до 1 у FCM. Таким чином, у випадках, коли ми не можемо легко вирішити, що об’єкти належать лише до одного кластеру, особливо якщо набори даних мають шуми або перекриття, алгоритм С-середніх може бути краще ніж К-середніх. З цієї причини очікується, що алгоритм К-середніх може бути хорошим варіантом для ексклюзивної кластеризації, але С-середніх може дати хороші результати для кластерів, що перекриваються.

К-середніх ітеративно обчислює центроїди кластерів для кожної міри відстані, щоб мінімізувати суму по відношенню до вказаної міри. Алгоритм К-середніх спрямований на мінімізацію цільової функції, відому як функція квадрата помилки, наведена в рівнянні 1.1, наступним чином:

$$J_{KM}(X;V) = \sum_{i=1}^{c} \sum_{j=1}^{n} D_{ij}^2 \qquad (1.1)$$

Де D_{ij}^2 – обрана міра відстані, яка зазвичай відповідає евклідовій нормі: $\left\| x_{ij} - v_i \right\|^2,\ 1 \le i \le c,\ 1 \le j \le n_j.$, де n_j представляє кількість точок даних у i^{th} кластері.

Для кластерів c алгоритм К-середніх заснований на ітераційному алгоритмі, що мінімізує суму відстаней від кожного об'єкта на його центроїд скупчення. Об’єкти переміщуються між кластерами, поки сума не може бути зменшена. Алгоритм К-середніх включає наступні кроки [90]:

1. Центроїди кластерів c вибираються з набору даних $\boldsymbol{X}$ випадковим чином
2. Розраховуються відстані між точками даних і центроїдами кластера.
3. Кожна точка даних призначається кластеру, центроїд якого є найближчим до нього.
4. Центроїди кластерів оновлюються за допомогою формули в рівнянні 1.2:

$$v_i = \sum_{i=1}^{n_i} x_{ij} / n_i;\ 1 \le i \le c \qquad (1.2)$$

5. Перераховуються відстані від оновлених центроїдів кластерів.

6. Якщо новому кластеру не призначено жодної точки даних, виконання алгоритму зупиняється, інакше кроки з 3 по 5 повторюються для ймовірного переміщення точок даних між кластерами.

Алгоритм FCM на відміну від K-середніх мінімізує цільову функцію в рівнянні

$$J_{FCM}(X;U,V) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^{m} D_{ijA}^{2} \qquad (1.3)$$

Ця функція відрізняється від класичної КМ використанням зважених квадратів помилок замість використання лише квадратів помилок. У цільовій функції в рівнянні 1.4 $\boldsymbol{U}$ — це нечітка матриця розділу, яка обчислюється з набору даних $\boldsymbol{X}$:

$$U = [u_{ij}] \in M_{FCM} \qquad (1.4)$$

Нечітка кластеризація даних з набору $\boldsymbol{X}$ представлена матрицею членства $\boldsymbol{U}$ у вимірі $c \times n$. Елемент u_{ij} є значенням належності i-го об'єкта до j-го кластера. У цьому випадку формується j-й стовпець матриці $\boldsymbol{U}$ із значеннями належності n об’єктів до j-го кластера. $\boldsymbol{V}$ — вектор-прототип кластерних прототипів (центроїдів):

$$V = [v_1, v_2, \ldots, v_c], v_i \in \mathbb{R}^p \qquad (1.5)$$

D_{ijA}^{2}це відстані між кластерами j^{th}. Вони обчислюються як квадратна норма відстані внутрішнього добутку в рівнянні (1.6):

$$D_{ijA}^{2} = \|x_j - v_i\|_A^2 = (x_j - v_i)^T A (x_j - v_j) \qquad (6)$$

У рівнянні 1.6 $\boldsymbol{A}$ є позитивною та симетричною матрицею норм. Внутрішній добуток з $\boldsymbol{A}$ є мірою відстаней між точками даних і прототипами кластера. Коли $\boldsymbol{A}$ дорівнює $\boldsymbol{I}$, D_{ijA}^{2} виходить у квадраті евклідової норми. У рівнянні 1.3 m є параметром

фазифікатора (або ваговим показником), значення якого вибирається як дійсне число, більше 1 ($m \in [1, \infty)$). У той час як m наближається до 1 кластеризація, як правило, стає чіткою, але коли вона йде до нескінченності, кластеризація стає нечіткою. Значення fuzzifier зазвичай вибирається як 2 у більшості додатків. Цільова функція мінімізується за допомогою обмежень наступним чином (7, 8 і 9):

$$u_{ij} \in [0,1];\ 1 \le i \le c,\ 1 \le j \le n \tag{1.7}$$

$$\sum_{i=1}^{c} u_{ij} = 1;\ 1 \le j \le n \tag{1.8}$$

$$0 < \sum_{j=1}^{n} u_{ij} < n;\ 1 \le i \le c \tag{1.9}$$

FCM є ітераційним процесом, який зупиняється, коли кількість ітерацій досягається максимальної або коли різниця між двома послідовними значеннями цільової функції менша за попередньо визначене значення збіжності (ε). Етапи FCM:

1) Ініціалізація $U^{(0)}$ матриці членства випадковим чином.

2) Розрахунок векторів-прототипів

$$v_j = \frac{\sum_{j=1}^{n} u_{ij}^{m} x_j}{\sum_{j=1}^{n} u_{ij}^{m}};\ 1 \le i \le c \tag{1.10}$$

3) Розрахунок значення членства за допомогою:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(D_{ijA} / D_{kjA} \right)^{\left(\frac{2}{m-1} \right)}};\ 1 \le i \le c,\ 1 \le j \le n \tag{1.11}$$

4) Порівняння $U^{(t+1)}$ з U^{t} , де t - номер ітерації

5) Якщо $\| U^{(t+1)} - U^{t} \| < \varepsilon$ тоді припинення роботи, інакше повернення до кроку 2

На рисунку 1.8 показані кроки перенесення FCM-алгоритму у програму.

Рисунок 1.8 – Алгоритм кластеризації С-середніх

Зазвичай алгоритм FCM складається з кількох кроків виконання. На першому кроці алгоритм випадковим чином вибирає С початкових центрів кластерів із вихідного набору даних. Потім, на наступних кроках, після кількох ітерацій алгоритму, кінцевий результат сходить до фактичного центру кластера. Тому вибір хорошого набору початкових центрів кластерів є дуже важливим для алгоритму С-середніх. Проте випадковим чином вибрати гарний набір початкових центрів кластера важко. Якщо вибрано хороший набір початкових центрів кластерів, алгоритм може зайняти менше ітерацій, щоб знайти фактичні центри кластерів [90]. Як результат різницю між чіткою кластеризацією та нечіткою кластеризацією показано на рисунку 1.9.

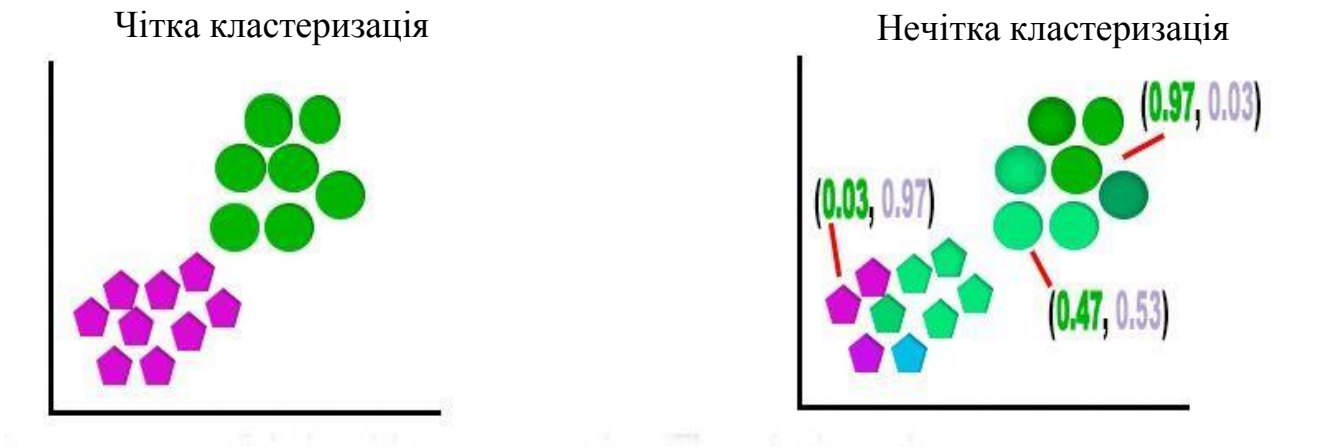

Рисунок 1.9 – Різниця між чіткою та нечіткою кластеризацією

Перевагами даного алгоритму FCM є [91]:

- Найкращий результат для набору даних, що перекривається, і порівняно кращий, ніж алгоритм k-середніх.
- На відміну від алгоритму k-середніх, де точка даних повинна належати виключно одному кластерному центру, тут точці даних призначається членство в кожному центрі кластера, в результаті чого точка даних може належати більш ніж одному кластерному центру.

Недоліками даного алгоритму є [91]:

- Апріорне уточнення кількості кластерів.
- При меншому значенні β ми отримуємо кращий результат, але за рахунок більшої кількості ітерацій.
- Евклідова відстань може неоднаково впливати на основні фактори.

1.2.4 Нечітка кластеризація в кібербезпеці

Безперечно, що використання машинного навчання все більше використовується у сфері забезпечення безпеки інформації, адже системи кібербезпеки відіграють важливу роль у сучасних комп’ютерних системах.

Використання методів нечіткої кластеризації найчастіше зустрічається в системах виявлення вторгнень, які є основним компонентом сучасних систем

кібербезпеки. Завдання системи виявлення вторгнень полягає в тому, щоб виявити аномалії у вхідному трафікі, а потім зупинити їх, перш ніж вони зможуть увійти у внутрішню систему. В останні роки системи виявлення вторгнень були оснащені найновішими алгоритмами машинного навчання для підвищення ефективності прогнозування. Тим не менш, поки що немає бажаного рівня уваги до застосування нечітких методів для розширення можливостей систем виявлення вторгнення. У роботі [92] вивчається використання нечіткої кластеризації в задачі виявлення вторгнень. Було оцінено алгоритм за допомогою найсучаснішого набору даних про вторгнення. Результати дозволяють авторам заявити про потенціал алгоритму нечіткої кластеризації для програм кібербезпеки.

Якщо говорити про нечітку кластеризацію і фішинг, то робота [93] пропонує нову структуру під назвою Phishing Dynamic Evolving Neural Fuzzy Framework (PDENF). Структура розраховує виявляти та передбачати невідомі фішингові електронні листи «нуль днів» із зниженням рівня хибнопозитивних повідомлень електронної пошти та хибнонегативних фішингових листів. Це робиться для підвищення рівня точності та підвищення ефективності класифікації та передбачення значень фішингової електронної пошти в онлайн-режимі, а також для тривалої роботи з об’ємом пам’яті.

Стаття C. Sánchez-Rebollo et al. [94], являє собою перевірку дописів у Twitter для визначення лідерів терористичних мереж та їх послідовників на основі нечіткої кластеризації. У роботі автори розробили методологію для виявлення потенційно небезпечних користувачів, які залишаються частково прихованими, окремо від тих, які найбільш відомі як дуже активні. Представлена методологія складається з визначення набору даних користувачів і кількох показників для визначення місцезнаходження впливових користувачів. Крім того, отримуються інші показники, як-от частота чи настрої їхніх твітів. Ці показники використовуються як вектори для сегментації даних. Для цього типу процедур рекомендується використовувати методи м’яких обчислень, які мають справу з неточністю інформації. У цій проблемі автори використали методи нечіткої кластеризації, щоб вказати тих користувачів, які були схильні до більшої активності в майбутньому і, як наслідок, за якими

вчасно слідкували, щоб перевірити їхню поведінку. Крім того, запропоновані методи аналізу включають неконтрольований алгоритм, тому їх можна застосовувати безперервно, таким чином, цю саму методологію можна використовувати для моніторингу користувачів, позначених як нечіткі.

1.3 Сучасний стан детектування фішингових сайтів за допомогою машинного навчання

Хоча організації повинні навчати співробітників тому, як розпізнавати фішингові електронні листи або посилання, щоб захистити від перерахованих вище типів атак, таке програмне забезпечення, як HTTrack, легко доступне для користувачів, створено, щоб повністю копіювати веб-сайти для задоволення власних цілей. Як наслідок, навіть навчених користувачів все ще можна обманом змусити розкрити конфіденційну чи персональну інформацію, взаємодіючи зі шкідливим веб-сайтом, який, на їхню думку, є законним.

Наведена вище проблема означає, що комп'ютерні рішення для захисту від фішингових атак необхідні разом із навчанням користувачів. Таке рішення дозволить комп'ютеру ідентифікувати шкідливі веб-сайти, щоб запобігти взаємодії користувачів із ними. Один із загальних підходів до розпізнавання незаконних фішингових веб-сайтів ґрунтується на їхніх уніфікованих локаторах ресурсів (URL). URL-адреса — це глобальна адреса документа у всесвітній мережі, і вона служить основним засобом для пошуку документа в Інтернеті. Навіть у випадках, коли вміст веб-сайтів дублюється, URL-адреси все одно можна використовувати, щоб відрізнити справжні сайти від самозванців.

Одним із підходів є використання чорного списку шкідливих URL-адрес, розроблених антивірусними групами. Проблема цього підходу полягає в тому, що чорний список не може бути вичерпним, оскільки нові шкідливі URL-адреси постійно з'являються. Таким чином, потрібні підходи, які можуть автоматично класифікувати нову, раніше незнану URL-адресу як фішинговий сайт, чи як законний. Такі рішення, як правило, є підходами на основі машинного навчання,

коли система може класифікувати нові фішингові сайти за допомогою моделі, розробленої з використанням навчальних наборів відомих атак.

Методи машинного навчання, які ідентифікують фішингові URL-адреси, зазвичай оцінюють URL-адресу на основі певної функції або набору ознак, отриманих з неї. Існують два загальних типи ознак, які можна отримати з URL-адрес, а саме ознаки на основі хосту та лексичні ознаки. Ознаки на основі хоста описують характеристики веб-сайту, наприклад, де він розташований, хто ним керує та коли сайт було встановлено. Крім того, лексичні характеристики описують текстові властивості URL-адреси. Оскільки URL-адреси — це просто текстові рядки, які можна розділити на частини, включаючи протокол, ім’я хоста та шлях, система може оцінити легітимність сайту на основі будь-якої комбінації цих компонентів [95].

1.3.1 Наявні рішення при детектуванні фішингових сайтів за допомогою машинного навчання

Для вирішення задач виявлення фішингових URL-адрес вигадуються багато різноманітних рішень, заснованих на методах машинного навчання. Наприклад, було запропоновано використовувати систему під назвою PILFER для класифікації фішингових URL-адрес. Під час навчання системи автори виокремили набір з десяти ознак шахрфйьских листів, які були спеціально розроблені, щоб підкреслити оманливі методи, які використовуються для обману користувачів. Навчальна вибірка складається з приблизно 860 фішингових електронних листів і 6950 нефішингових листів. Під час дослідження використовувався такий метод машинного навчання як машина опорних векторів (SVM) у якості класифікатору при реалізації. Було навчено та випробувано класифікатор за допомогою 10-кратної перехресної перевірки, а у якості результату автори отримали 92-відсоткову точність [96].

З іншого боку проблему класифікації URL-адрес було розглянуто як проблему бінарної класифікації, В даному випадку була створена система класифікації URL-

адрес, що може обробляти живий канал із позначеними URL-адресами. Система також постійно збирає URL-адреси в режимі реального часу від надійного постачальника послуг електронної пошти. Було навчано систему, яка використостовує як лексичні, так і базовані на хості ознаки, саме на їх основі був навчаний онлайн-класифікатор за допомогою алгоритму зваженої впевненості (CW) [97].

Інша система виявлення фішингових URL-адрес під назвою Multi-label Classifier була побудована на основі асоціативної класифікації. Multi-label Classifier — це алгоритм, заснований на правилах, де кілька правил міток витягуються з набору фішингових даних. Автори використали при навчанні шістнадцять ознак і класифікували URL-адреси на три класи: фішингові, легітимні та підозрілі. [98].

Задля винайдення рішення детектування фішингових сайтів був використаний навіть алгоритм швидкої асоціативної класифікації (FACA) для класифікації фішингових URL-адрес. FACA працює шляхом виявлення всіх частих наборів елементів правил і послідуйочого створення моделі для класифікації. Автори дослідили набір даних, що складається з 11 055 веб-сайтів з двома класами: достовірний і шахрайський. Набір даних містив тридцять ознак [99].

У додаток був розроблений метод ідентифікації сервера командування та керування за допомогою контрольованих точок навчання та ознак, отриманих з інформації WHOIS та DNS. Автори оцінили доменні імена та адреси електронної пошти з WHOIS як вхідні значення для машинного навчання [100, 101].

1.3.2 Популярні моделі машинного навчання при детектуванні фішингових сайтів

Фішингові атаки можна запобігти, виявивши веб-сайти та поінформувавши користувачів про виявлення фішингових веб-сайтів, алгоритми машинного навчання вважаються одними із найпотужніших методів виявлення фішингових атак. Зазвичай використовують наступні методи виявлення:

- дерево рішень та випадковий ліс;
- наївний байєс;
- SVM;
- logical regression;
- knn

Дерево рішень — це структура, подібна до блок-схеми, в якій кожен внутрішній вузол представляє перевірку ознаки (наприклад, підкидання монети випадає орлом чи решкою), кожен вузол листка представляє мітку класу (рішення, прийняте після обчислення всіх функцій) і гілки представляють поєднання ознак, які ведуть до цих міток класів. Шляхи від кореня до листа представляють правила класифікації. Дерева рішень будуються за допомогою алгоритмічного підходу, який визначає способи поділу набору даних на основі різних умов. Це один з найбільш широко використовуваних і практичних методів навчання «з вчителем».

Підвищення інформації визначається як різниця між ентропією до розщеплення та ентропією після розщеплення. Нижче наведено рівняння для розрахунку приросту інформації.

Перевагами дерева рішень є простота у використанні та розумінні, можливість обробляти як категоріальні, так і числові дані, стійкість до викидів, тому вимагає невелику попередню обробку даних.

Недоліками дерева рішень є схильність до переобладнання, вимога певного виміру того, наскільки добре метод працює, необхідність бути обережним з налаштуванням параметрів, можливість створювати упереджені вивчені дерева, якщо деякі класи домінують [102].

Випадковий ліс — це гнучкий, простий у використанні алгоритм машинного навчання, який дає чудовий результат навіть без налаштування гіперпараметрів. Це також один із найбільш використовуваних алгоритмів через його простоту та різноманітність (його можна використовувати як для завдань класифікації, що нам потрібно, так і для задач регресії). «Ліс», який будує алгоритм, являє собою ансамбль дерев рішень, комбінація моделей навчання підвищує загальний результат.

Простіше кажучи: випадковий ліс створює кілька дерев рішень і об’єднує їх разом, щоб отримати більш точне та стабільне передбачення [103].

Naive Bayes – це алгоритм класифікації, який підходить для бінарної та багатокласової класифікації. Наївний Байєс добре працює у випадках категорійних вхідних змінних порівняно з числовими змінними. Це корисно для прогнозування та прогнозування даних на основі історичних результатів.

Наївний байєсівський класифікатор обчислює апостеріорну ймовірність для кожного класу та відносить вибірку до класу з максимальною ймовірністю. Апостеріорна ймовірність для класу може бути розрахована за даними навчального набору [101].

"Support Vector Machine" (SVM) полягає у виборі тієї конкретної гіперплощини, яка знаходиться «прямо посередині» між прикладами двох класів. Математично кажучи, SVM вибирає гіперплощину, яка максимізує мінімальну відстань між цією гіперплощиною та всіма прикладами. Іншими словами, гіперплощина рівновіддалена від прикладів двох найближчих до неї класів. У термінології SVM двократна відстань між гіперплощиною та найближчими до неї точками називається маржею. У результаті SVM також відомий як класифікатор максимальної маржі.[104].

Модель логістичної регресії приймає натуральний логарифм шансів як функцію регресії провісників. У статистиці логістична регресія використовується для прогнозування ймовірності певного класу чи події, наприклад, ймовірність перемоги команди, здоров’я пацієнта тощо. Це можна розширити для моделювання кількох класів подій, наприклад визначення того, чи містить зображення кішку, собаку, лева тощо [105].

1.4 Потенціал розвитку методів детектування фішингових сайтів та постановка задач дослідження

Зазвичай у своїх роботах автори пропонують використання різних моделей машинного навчання для класифікації фішингових атак. Такі підходи складаються з вилучення ознак із сайтів та класифікації веб-сайтів на основі вилучених ознак. Для класифікації цих ознак використовуються такі методи, як опорні векторні машини (SVM), наївні байєсові (NB), випадковий ліс або логістична регресія, але у всіх них є один недолік – їх потрібно постійно навчати на нових вибірках даних, адже методи створення і імітування фішингових сайтів постійно оновлюються чи є тимчасовими, а самі класифікатори зазвичай не є дуже швидкими. Саме ці недоліки усуває машинне навчання без вчителя.

Метою використання неконтрольованого навчання є безпосереднє вилучення структури з набору даних без попереднього навчання. Хоча навчання з наглядом забезпечує набагато більшу точність, навчання без нагляду забезпечує швидкий і надійний підхід до отримання знань із набору даних [106].

Саме тому головними задачами дослідження стали розробка некерованої моделі детектування фішингових сайтів на основі нечіткої кластеризації та знаходження її ефективності, а також порівняння результатів та швидкодії моделі зі звичними алгоритмами детектування шахрайських посилань за допомогою навчання з вчителем, такими як моделі SVM, наївний Байєс, дерева рішень та логістична регресія. Вирішення вищеперерахованих задач допоможе заощадити витрати людиногодин та ресурсів комп’ютера, які вони могли би витратити на навчання контрольованих схем.

Висновки за розділом 1

Сьогодні використання Інтернету дуже поширено, і суспільство в значній мірі покладається на веб-сервіси для різних цілей. Водночас шкідливі веб-сайти

становлять значну загрозу для користувачів Інтернету, а необізнані користувачі можуть стати жертвами шкідливих URL-адрес, які містять небажаний вміст, наприклад фішинг і спам.

Фішинг – це форма крадіжки особистих даних, яка відбувається, коли зловмисний веб-сайт видає себе за легальний, щоб отримати конфіденційну інформацію, таку як паролі, дані облікового запису або номери кредитних карток [107].

Існує два основних методи виявлення фішингових атак: навчання користувачів і класифікація програмного забезпечення. Користувачів можна навчити розпізнавати фішинг і реальні сайти, розуміючи природу фішингових атак. Класифікація програмного забезпечення прагне точніше класифікувати фішингові та автентичні сайти та зменшує людські помилки чи необізнаність.

При класифікації програмного забезпечення з базою даних шкідливих і небезпечних веб-сайтів завдання виявлення фішингових посилань зазвичай виконується за допомогою бінарного класифікатора. Досить застосувати звичайні методи класифікації або розробити систему оцінок, засновану на статистичному розподілі якостей між двома класами посилань та їх експертній оцінці. Цей тип машинного навчання називають навчанням із наглядом, оскільки є дані, які добре позначені.

Однак є ще один метод, який називається неконтрольованим машинним навчанням, який, незважаючи на низьку точність, зараз є найбільш перспективним напрямком наукових досліджень. Такі моделі не вимагають мічених даних, і тому мають великий потенціал для застосування, зокрема у сфері кібербезпеки.

Серед методів навчання без нагляду є нечітка кластеризація, яка являє собою техніку групування багатовимірних об’єктів, яка ґрунтується на представленні результатів окремих спостережень у вигляді точок у відповідному геометричному просторі, а потім на виборі груп у вигляді «скупчень» цих точок.

Таким чином, основна мета аналізу полягає в тому, щоб вибрати в вихідних багатовимірних даних однорідні підмножини такого типу, щоб об’єкти всередині груп були схожими або подібними один до одного, а об’єкти з різних груп, навпаки,

не були схожими. «Подібність» це близькість об'єктів в багатовимірному просторі ознак. В такому випадку задачею стає виділення в цьому багатовимірному просторі природних скупчень об'єктів, які можна вважати однорідними групами [107].

Стаття C. Sánchez-Rebollo et al. [94], являє собою перевірку дописів у Twitter для визначення лідерів терористичних мереж та їх послідовників на основі нечіткої кластеризації. Архітектура великих даних забезпечує категоризацію людей на основі їхньої активності, здатності впливати на інших користувачів і вмісту повідомлень. Автори використовували графіки, щоб дослідити, як повідомлення поширюються по мережі, і методи нечіткої кластеризації, щоб класифікувати людей за профілями, що є доказом того, що навчання без нагляду можна використовувати для виявлення фішингових сайтів.

Як результат у другому розділі пропонується розглянути алгоритм нечіткої кластеризації як спосіб виявлення фішингових сайтів, а в третьому розділі магістерської роботи дослідити ефективність нечіткої кластеризації індикаторів компромісу для виявлення фішингових сайтів. У цьому методі показники компромісу є прикладом ознаки кластеризації. Оскільки показники компромісу є не тільки кількісними, але є й якісні, тому можна застосовувати нечітку кластеризацію.

РОЗДІЛ 2
РОЗРОБКА МЕТОДУ ДЕТЕКТУВАННЯ ФІШИНГОВИХ САЙТІВ НА ОСНОВІ НЕЧІТКОЇ КЛАСТЕРИЗАЦІЇ ІНДИКТОРІВ КОМПРОМЕТАЦІЇ

2.1 Операційна структура дослідження

Перш ніж починати імплементацію методу нечіткої кластеризації індикаторів компрометації, потрібно розробити план дій і структуру проекту. Рисунок 2.1 показує операційну структуру, яка буде дотримуватись у цьому дослідженні.

Дослідження поділено на три етапи, і результат кожної фази є вхідним для наступного етапу, етапи показані в операційній структурі:

- етап 1 заснований на обробці набору даних;
- етап 2 заснований на написанні алгоритму програми та тестуванні програми, вилучення поточних оцінок ефективності;
- етап-3 має на меті оцінити результативність методу нечіткої кластеризації С-середніх за допомогою точності та порівняння її з іншими широко видомими методами детектування фішингових сайтів на основі машинного навчання.

Етап 1, який передбачає обробку набору даних та вилучення ознак потрібно провести на зібраних наборах даних для кращого уточнення їх відповідно до вимог дослідження. Обробка даних повинна включати декілька етапів, наприклад, як вилучення ознак – ідентифікаторів компрометації, нормалізація даних, поділ набору даних і зважування атрибутів. Це дуже необхідно для того, щоб досліджуваний алгоритм міг зрозуміти набір даних і правильно класифікувати їх у еталонні кластери. Результат цього етапу безпосередньо передається до другого етапу дослідження результативності імплементації методу нечіткої кластеризації ідентифікаторів компрометації. .

Етап 2 передбачає безпосередньо написання і імплементацію алгоритму, а також визначення поточних оцінок роботи кластеризатора. У цьому дослідженні необхідно вилучити поточні оцінки роботи кластеризатора для вимірювання ефективності методу, досягнутої за допомогою алгоритму навчання. Для цього використовується тестовий набір, що складається з набору даних з відомими мітками. Далі важливо оцінити кластеризатор шляхом тестування з набором даних, отриманим на етапі 1, використовуючи наступні показники ефективності, як точність та швидкодія.

Етап 3 передбачає порівняння ефективності загальновживаних методів детектування фішингових сайтів з досліджуваним методом нечіткої кластеризації індикаторів компрометації і доведення його результативності.

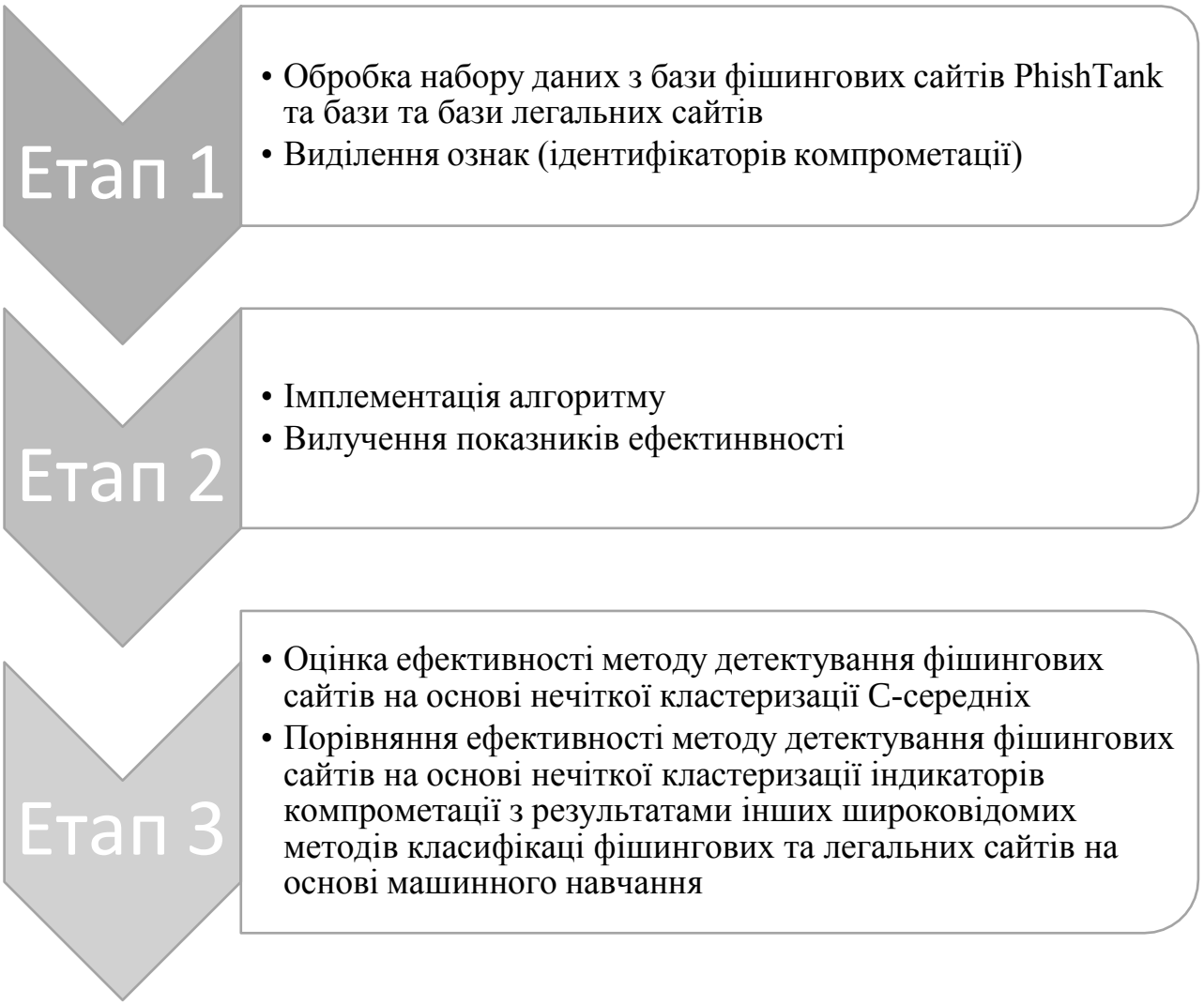

Рисунок 2.1 – Операційна структура дослідження

2.2 Набір даних для дослідження

Набір даних, який буде використовуватися для дослідження, буде завантажено з репозитарію факультета електронної інженерії і комп'ютерних наук словенського університету Марібор з результатом дослідження «Набори даних для детектування фішингових сайтів» [108]. Цей набір даних складається з набору законних, а також фішингових екземплярів веб-сайтів. Кожен веб-сайт представлений набором ознак, які вказують на те, чи є веб-сайт легітимним або ні. Набір даних містить 111 ознак з 88 647 URL-адрес. Набір даних, позначений як фішинговий, був включений з реєстру PhishTank, де сайти перевіряються кількома користувачами. До законних веб-сайтів були включені веб-сайти з загальнодоступних, обраних спільнотою організованих наборів [109], а також із веб-сайтів, які користуються вищим рейтингом Alexa.

Усього набір даних містить 111 атрибутів, за винятком цільового фішингового атрибута, який вказує, чи є конкретний екземпляр законним (значення 0) чи фішинговим (значення 1). Набір даних складається з 88 647 екземплярів з 30 647 екземплярів, позначених як фішинг, і 58 000 екземплярів, позначених як легітимні. Метою цього є імітація реальної ситуації, де є більше легітимних веб-сайтів. Розподіл між класами такого варіанту набору даних представлений на рисунку 2.2

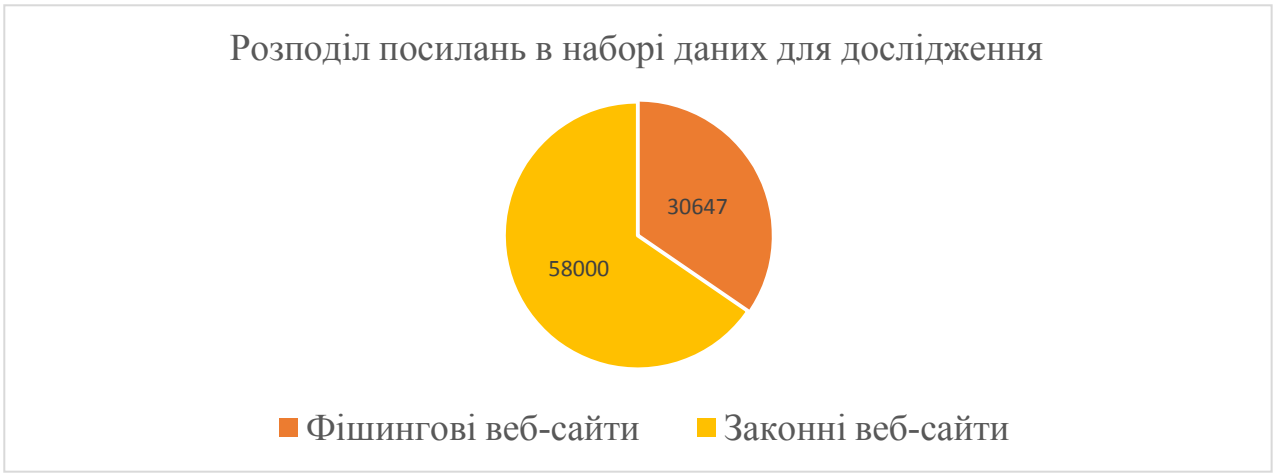

Рисунок 2.2 – Розподіл посилань в наборі даних для дослідження

Атрибути використованого набору даних можна розділити на шість груп, які представлені на рисунку 2.3: загальні властивості рядка URL, властивості домену, властивості каталогу URL-адрес; властивості файлу URL; властивості параметра URL; дані розпізнавання URL-адрес і зовнішніх показників.

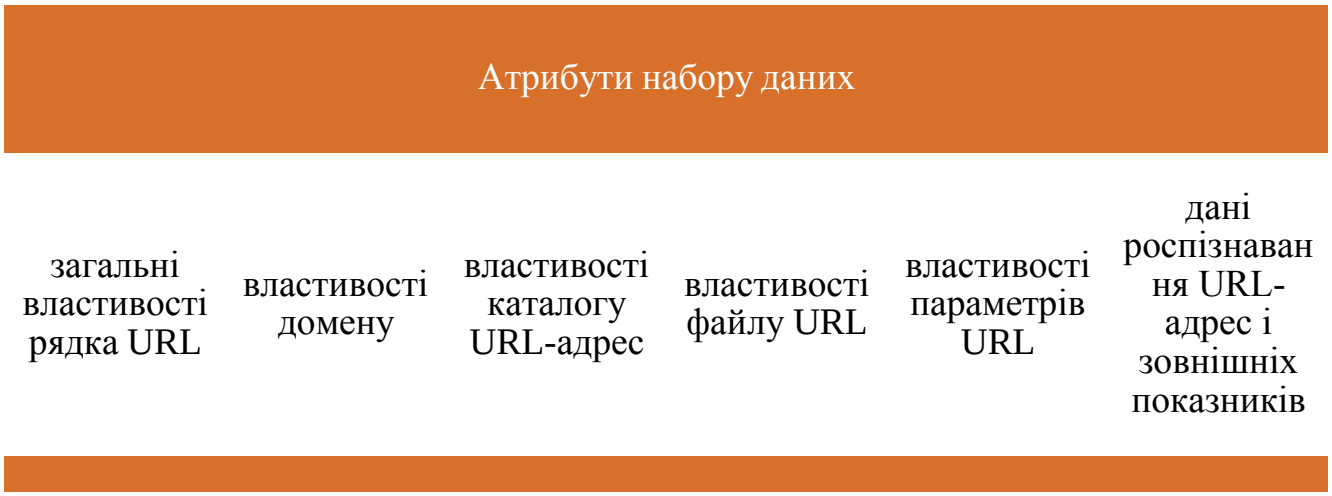

Рисунок 2.3 – Атрибути набору даних

Перша група заснована на значеннях атрибутів усього рядка URL-адреси, тоді як значення наступних чотирьох груп засновані на окремих підрядках, як показано на рисунку 2.4. Атрибути останньої групи засновані на розділенні URL - метрик, а також на зовнішні служби, такі як індекс пошуку Google.

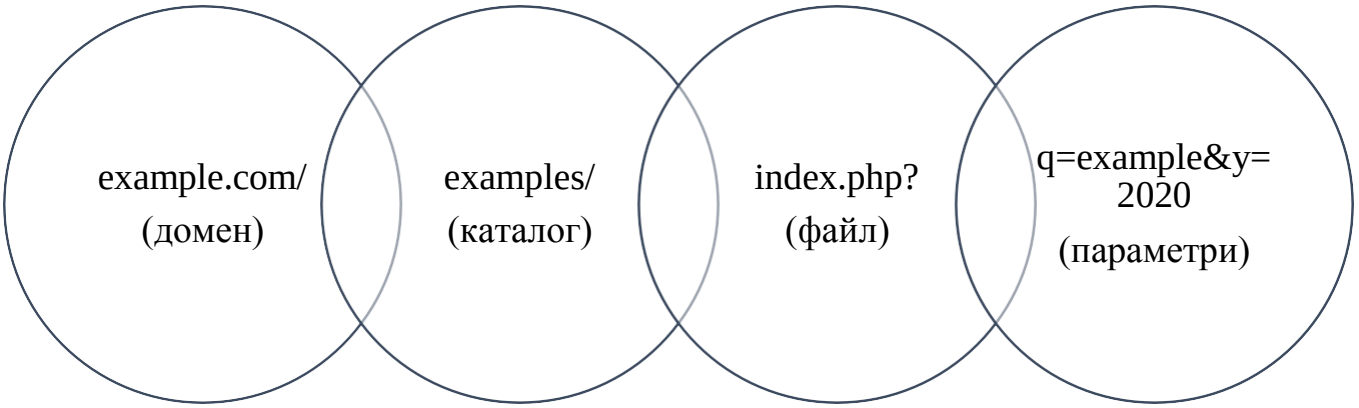

Рисунок 2.4 – Підрядки URL-адреси

Загальні властивості рядка URL представлені в таблиці 2.1;

Таблиця 2.1

Загальні властивості рядка

№	Атрибут	Опис	Формат	Значення
1	qty_dot_url	Кількість ”.”	Числовий	
2	qty_hyphen_url	Кількість ”-”	Числовий	
3	qty_underline_url	Кількість ”_”	Числовий	
4	qty_slash_url	Кількість ”/”	Числовий	
5	qty_questionmark_url	Кількість ”?”	Числовий	
6	qty_equal_url	Кількість ”=”	Числовий	
7	qty_at_url	Кількість ”@”	Числовий	
8	qty_and_url	Кількість ”&”	Числовий	
9	qty_exclamation_url	Кількість ”!”	Числовий	
10	qty_space_url	Кількість ” ”	Числовий	
11	qty_tilde_url	Кількість ”~”	Числовий	
12	qty_comma_url	Кількість ”,”	Числовий	
13	qty_plus_url	Кількість ”+”	Числовий	
14	qty_asterisk_url	Кількість ”*”	Числовий	
15	qty_hashtag_url	Кількість ”#”	Числовий	
16	qty_dollar_url	Кількість ”$”	Числовий	
17	qty_percent_url	Кількість ”%”	Числовий	
18	qty_tld_url	Кількість символів верхнього домену	Числовий	
19	length_url	Кількість символів	Числовий	
20	email_in_url	Примутність електронної адреси	Булеве значення	[0,1]

Властивості домену представлені в таблиці 2.2;

Таблиця 2.2

Властивості домену

№	Атрибут	Опис	Формат	Значення
1	qty_dot_domain	Кількість ”.”	Числовий	
2	qty_hyphen_domain	Кількість ”-”	Числовий	
3	qty_underline_domain	Кількість ”_”	Числовий	
4	qty_slash_domain	Кількість ”/”	Числовий	
5	qty_questionmark_domain	Кількість ”?”	Числовий	
6	qty_equal_domain	Кількість ”=”	Числовий	
7	qty_at_domain	Кількість ”@”	Числовий	
8	qty_and_domain	Кількість ”&”	Числовий	
9	qty_exclamation_domain	Кількість ”!”	Числовий	
10	qty_space_domain	Кількість ” ”	Числовий	
11	qty_tilde_domain	Кількість ”~”	Числовий	
12	qty_comma_domain	Кількість ”,”	Числовий	
13	qty_plus_domain	Кількість ”+”	Числовий	
14	qty_asterisk_domain	Кількість ”*”	Числовий	
15	qty_hashtag_domain	Кількість ”#”	Числовий	
16	qty_dollar_domain	Кількість ”$”	Числовий	
17	qty_percent_domain	Кількість ”%”	Числовий	
18	qty_vowels_domain	Кількість голосних	Числовий	
19	domain_length	Кількість символів домену	Числовий	
20	domain_in_ip	URL-адреса в форматі IP-адреси	Булеве значення	[0,1]
21	server_client_domain	домен «сервер» чи «кліент»	Булеве значення	[0,1]

Властивості каталогу URL-адрес представлені в таблиці 2.3

Таблиця 2.3

Властивості каталогу URL-адрес

№	Атрибут	Опис	Формат	Значення
1	qty_dot_directory	Кількість ”.”	Числовий	
2	qty_hyphen_directory	Кількість ”-”	Числовий	
3	qty_underline_directory	Кількість ”_”	Числовий	
4	qty_slash_directory	Кількість ”/”	Числовий	
5	qty_questionmark_directory	Кількість ”?”	Числовий	
6	qty_equal_directory	Кількість ”=”	Числовий	
7	qty_at_directory	Кількість ”@”	Числовий	
8	qty_and_directory	Кількість ”&”	Числовий	
9	qty_exclamation_directory	Кількість ”!”	Числовий	
10	qty_space_directory	Кількість ” ”	Числовий	
11	qty_tilde_directory	Кількість ”~”	Числовий	
12	qty_comma_directory	Кількість ”,”	Числовий	
13	qty_plus_directory	Кількість ”+”	Числовий	
14	qty_asterisk_directory	Кількість ”*”	Числовий	
15	qty_hashtag_directory	Кількість ”#”	Числовий	
16	qty_dollar_directory	Кількість ”$”	Числовий	
17	qty_percent_directory	Кількість ”%”	Числовий	
18	directory_length	Кількість символів директорії	Числовий	

Властивості файлу URL представлені в таблиці 2.4.

Таблиця 2.4

Властивості файлу URL

№	Атрибут	Опис	Формат	Значення
1	qty_dot_file	Кількість ”.”	Числовий	
2	qty_hyphen_file	Кількість ”-”	Числовий	
3	qty_underline_file	Кількість ”_”	Числовий	
4	qty_slash_file	Кількість ”/”	Числовий	
5	qty_questionmark_file	Кількість ”?”	Числовий	
6	qty_equal_file	Кількість ”=”	Числовий	
7	qty_at_file	Кількість ”@”	Числовий	
8	qty_and_file	Кількість ”&”	Числовий	
9	qty_exclamation_file	Кількість ”!”	Числовий	
10	qty_space_file	Кількість ” ”	Числовий	
11	qty_tilde_file	Кількість ”~”	Числовий	
12	qty_comma_file	Кількість ”,”	Числовий	
13	qty_plus_file	Кількість ”+”	Числовий	
14	qty_asterisk_file	Кількість ”*”	Числовий	
15	qty_hashtag_file	Кількість ”#”	Числовий	
16	qty_dollar_file	Кількість ”$”	Числовий	
17	qty_percent_file	Кількість ”%”	Числовий	
18	file_length	Кількість символів в назві файлу	Числовий	

Властивості параметра URL представлені в таблиці 2.5.

Таблиця 2.5

Властивості параметра URL

№	Атрибут	Опис	Формат	Значення
1	qty_dot_params	Кількість ”.”	Числовий	
2	qty_hyphen_params	Кількість ”-”	Числовий	
3	qty_underline_params	Кількість ”_”	Числовий	
4	qty_slash_params	Кількість ”/”	Числовий	
5	qty_questionmark_params	Кількість ”?”	Числовий	

Продовження табл. 2.5

№	Атрибут	Опис	Формат	Значення
6	qty_equal_params	Кількість ”=”	Числовий	
7	qty_at_params	Кількість ”@”	Числовий	
8	qty_and_params	Кількість ”&”	Числовий	
9	qty_exclamation_params	Кількість ”!”	Числовий	
10	qty_space_params	Кількість ” ”	Числовий	
11	qty_tilde_params	Кількість ”~”	Числовий	
12	qty_comma_params	Кількість ”,”	Числовий	
13	qty_plus_params	Кількість ”+”	Числовий	
14	qty_asterisk_params	Кількість ”*”	Числовий	
15	qty_hashtag_params	Кількість ”#”	Числовий	
16	qty_dollar_params	Кількість ”$”	Числовий	
17	qty_percent_params	Кількість ”%”	Числовий	
18	params_length	Кількість символів в параметрах	Числовий	
19	tld_present_params	Присутність верхнього домену в параметрах	Булеве значення	[0,1]
20	qty_params	Кількість параметрів	Числовий	

Дані розпізнавання URL-адрес і зовнішніх показників представлені в таблиці 2.6.

Таблиця 2.6

Дані розпізнавання URL-адрес і зовнішніх показників

№	Атрибут	Опис	Формат	Значення
1	time_response	Час пошуку домену	Числовий	
2	domain_spf	Домен має SPF	Булеве значення	[0,1]
3	asn_ip	Номер автономної системи	Числовий	

Продовження табл. 2.6

№	Атрибут	Опис	Формат	Значення
4	time_domain_activation	Час активації домену (в днях)	Числовий	
5	time_domain_expiration	Термін дії домену (в днях)	Числовий	
6	qty_ip_resolved	Кількість вирішених IP-адрес	Числовий	
7	qty_nameservers	Кількість вирішених NS	Числовий	
8	qty_mx_servers	Кількість серверів MX	Числовий	
9	ttl_hostname	TTL	Числовий	
10	tls_ssl_certificate	Дійсний TTL сертифікат	Булеве значення	[0,1]
11	qty_redirects	Кількість редиректів	Числовий	
12	url_google_index	Індексованість URL в Google	Булеве значення	[0,1]
13	domain_google_index	Індексованість домену в Google	Булеве значення	[0,1]
14	url_shortened	Скорочений URL	Булеве значення	[0,1]
15	phishing	Фішинговий веб-сайт	Булеве значення	[0,1]

В даному наборі даних атрибути вважаються індикаторами компрометації фішингових сайтів, а сам набір даних планується використовувати як в процесі реалізації алгоритму нечіткої кластеризації, так і для проведення порівняльного аналізу з широковживаними алгоритмами класифікації веб-сайтів.

Висновки за розділом 2

Перш ніж починати імплементацію методу нечіткої кластеризації індикаторів компрометації, потрібно розробити план дій і структуру проекта. Заплановано

поділити дослідження на три етапи, де результат кожної фази є вхідним для наступного етапу:

- етап 1 заснований на обробці наборів даних і виділенні ознак;
- етап 2 засноваий на імплементації алгоритму та тестуванні з визначенням параметрів ефективності
- етап-3 має на меті оцінити результативність методу нечіткої клестеризації С-середніх за допомогою параметрів ефективності та порівняння її з іншими широко видомими методами детектування фішингових сайтів на основі машинного навчання.

Для першого етапу планується використовувати набір даних із 30 647 підтверджених фішингових URL-адрес з веб-сайту Phishtank, 58 000 законних URL-адрес, з яких частина була отримана з веб-сайту Alexa, а інша частина з 27 998 позначених та організованих URL-адрес спільнотою, яка позначила URL-адреси, які вказують на об’єктивно повідомлені новини, і таким чином також є законними.

В даному наборі даних атрибути вважаються індикаторами компрометації фішингових сайтів, а сам набір даних планується використовувати як в процесі реалізації алгоритму нечіткої кластеризації, так і для проведення порівняльного аналізу з широковживаними алгоритмами класифікації веб-сайтів.

РОЗДІЛ 3
ІМПЛЕМЕНТАЦІЯ МОДЕЛІ НЕЧІТКОЇ КЛАСТЕРИЗАЦІЇ ДЛЯ ДЕТЕКТУВАННЯ ФІШИНГОВИХ САЙТІВ

3.1 Імплементація, тестування та оцінка ефективності моделі нечіткої кластеризації індикаторів компрометації

При імплементації моделі нечіткої кластеризації С-середніх для детектування фішингових сайтів був дотриманий наступний алгоритм:

Рисунок 3.1 – Алгоритм побудови моделі нечіткої кластеризації С-середніх

Для початку було імпортовано необхідні для роботи моделі бібліотеки Python. В імпортованому списку є бібліотеки numpy, pandas, matplotlib, skfuzzy, та пакет itertools.

Numpy — це бібліотека, яка додає підтримку великих багатовимірних матриць і масивів, разом з цим і колекцію математичних функцій високого рівня для роботи з цими матрицями і масивами.

Pandas в свою чергу пропонує структури даних та операції для маніпулювання числовими таблицями та часовими рядами.

Matplotlib — це бібліотека для створення графіків та її числового математичного розширення NumPy.

Skfuzzy — колекція алгоритмів для роботи з нечіткою логікою, призначена для використання у стеку SciPy.

Пакет itertools містить три ітератори, які можуть виконувати ітерацію нескінченно.

Код цього імпорту показаний у лістингу 1.

```
import numpy as np, pandas as pd, os
import itertools
import skfuzzy as fuzz
import matplotlib.pyplot as plt
from sklearn.decomposition import TruncatedSVD
from sklearn.metrics import confusion_matrix
import statsmodels.formula.api as smf
from sklearn.preprocessing import StandardScaler, Normalizer
import statsmodels.api as sm
```

Лістинг 1.

За допомогою бібліотеки panda ми можемо прочитати підготовлений набір даних у зручному форматі. В лістингу 2 та на рисунку 3.2 можна побачити код, за допомогою якого зчитується датасет з файлу dataset_small.csv, розміщеного в кореневій директорії, та виводиться у форматі таблиці за допомогою функції shape, результат показаний на рисунку 3.2..

dt = pd.read_csv("dataset_small.csv")

print(dt.shape)

dt.head()

Лістинг 2.

```
dt = pd.read_csv("dataset_small.csv")
```

Python

```
print(dt.shape)
dt.head()
```

Python

```
(58645, 112)
```

	qty_dot_url	qty_hyphen_url	qty_underline_url	qty_slash_url	qty_questionmark_url	qty_equal_url	qty_at_url	qty_and_url	qty_exclamation_url	qty_space_url	...	qty_i
0	2	0	0	0	0	0	0	0	0	0	...	
1	4	0	0	2	0	0	0	0	0	0	...	
2	1	0	0	1	0	0	0	0	0	0	...	
3	2	0	0	3	0	0	0	0	0	0	...	
4	1	1	0	4	0	0	0	0	0	0	...	

5 rows × 112 columns

Рисунок 3.2 – Результат виводу вхідного набору даних у вигляді таблиці

Так як в наборі даних останній 112-ий стовбець заздалегідь позначений як стовпець класу (стовпець, який вказує, яка точка/вхід/URL-адреса є легітимною/фішинговою), була створена змінна x_std, у яку були перенесені всі стовпці датасету окремо від позначеного стовпця класу, тому загалом змінна має 111 стовпців зі 112. За допомогою методу StandardScaler() було переформатовано точки даних у змінній x_std на точки даних від 0 до 1, що показано у лістингу 3.

scaler = StandardScaler()

xstd = scaler.fit_transform(dt.iloc[:,:-1])

Лістинг 3

Розклад за сингулярними значеннями (SVD) – це метод зменшення розмірності для матриць, який зводить матрицю до її компонента для спрощення обчислень.

За допомогою методу TruncatedSVD з використанням алгоритму 'arpack' було перетворено 111 стовпців зі змінної x_std у 2 стовпці для спрощення обчислень.

Створення двох стовпців «component_1» і «component_2» за допомогою pandas. DataFrame показано у лістингу 4.

```
lsa = TruncatedSVD(2, algorithm='arpack')
dtmlsa = lsa.fit_transform(xstd)
dtmlsa = Normalizer(copy=False).fit_transform(dtmlsa)
a = pd.DataFrame(dtm_lsa, columns = ["1_component","2_ component"])
```

Лістинг 4.

За допомогою вищенаведеного лістингу №4 було спрощено дані з матриці 58645 рядків x 111 стовпців до 58645 рядків × 2 стовпців, результат виведено на рисунок 3.3.

	component_1	component_2
0	-0.974461	0.224559
1	0.630044	-0.776560
2	0.576535	-0.817073
3	0.632915	-0.774222
4	0.660093	-0.751184
...	...	...
58640	0.780265	-0.625449
58641	-0.974979	0.222298
58642	0.680411	0.732830
58643	-0.977393	0.211432
58644	0.648256	-0.761423

58645 rows × 2 columns

Рисунок 3.3 – Результат спрощення набору даних до двох стовпців

В лістингу 5 показано призначення стовпця класу до змінної y та побудову 9 графіків, щоб у подальшому побачити, кількість кластерів, які дають найкращий FPC.

FPC розраховується шляхом підгонки даних до алгоритму C-середніх.

```
x=pd.DataFrame(dt.iloc[:,111])
a['targets']=x
plt1, axes1 = plt.subplots(3,3,figsize=(8,8))
```

dataset = np.vstack((a[‘1_component’], a[‘2_component’]))

Лістинг 5.

На рисунку 3.4 показано результат виконаних дій.

Рисунок 3.4 – Основа для побудови графіків

Лістинг 6 показує створення / імплементацію моделі нечіткої кластеризації С-середніх, підгонку даних до неї та вивід вихідного зображення у кореневу папку. Результат виконання показаний на рисунку 3.5.

for ncntrs, axes in enumerate(axes1.reshape(-1),2):

fpc, cntr, u, jm, p = fuzz.cluster.cmeans(dataset, ncntrs, 2, error=0.005, maxiter=1000, init=None)

```
    clstr_membership = np.argmax(u, axis=0)
    for i in range(ncntrs):
        axes.plot(a[1_component][clstr_membership==i],a[2_component
'][clstr_membership == i],'.')
        for j in cntr:
            axes.plot(j[0], j[1], 'rs')
        axes.set_title('Centres = {0}; FPC = {1:2f}'.format(ncntrs, fpc))
        axes.axis('off')
plt1.tight_layout()
plt1.savefig('phishing_detection')
```

Лістинг 6

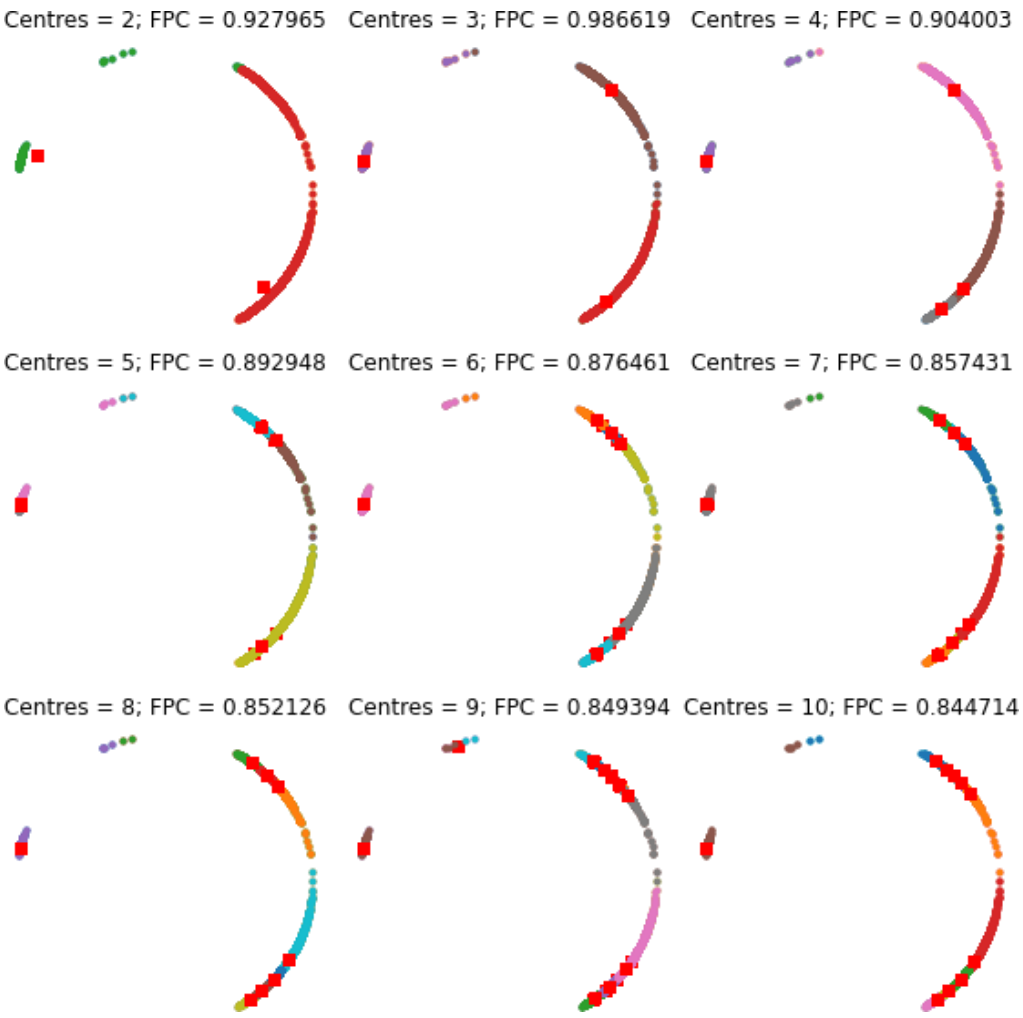

Рисунок 3.5 – Результат нечіткої кластеризації з центрами кластерів від 2 до 11

На рисунку 3.5 другий варіант з трьома кластерами показує найкраще значення FPC. Тобто якщо розглядати 3 центри при використанні нечіткої кластеризації, то це означає, що при використанні нечіткої кластеризації С-середніх набор даних ділиться на 3 кластери, і це є найкращим варіантом для кластеризації в даному випадку з ефективністю 98%.

Для того, щоб перевірити точність методу, було закоментовано код, показаний у лістингу 3, який прибирає з датасету стовбець, в якому позначено, чи є сайт фішинговим, та запущено програму ще раз. Результат цієї дії показаний на рисунку 3.6, і він є ідентичним до попереднього результату.

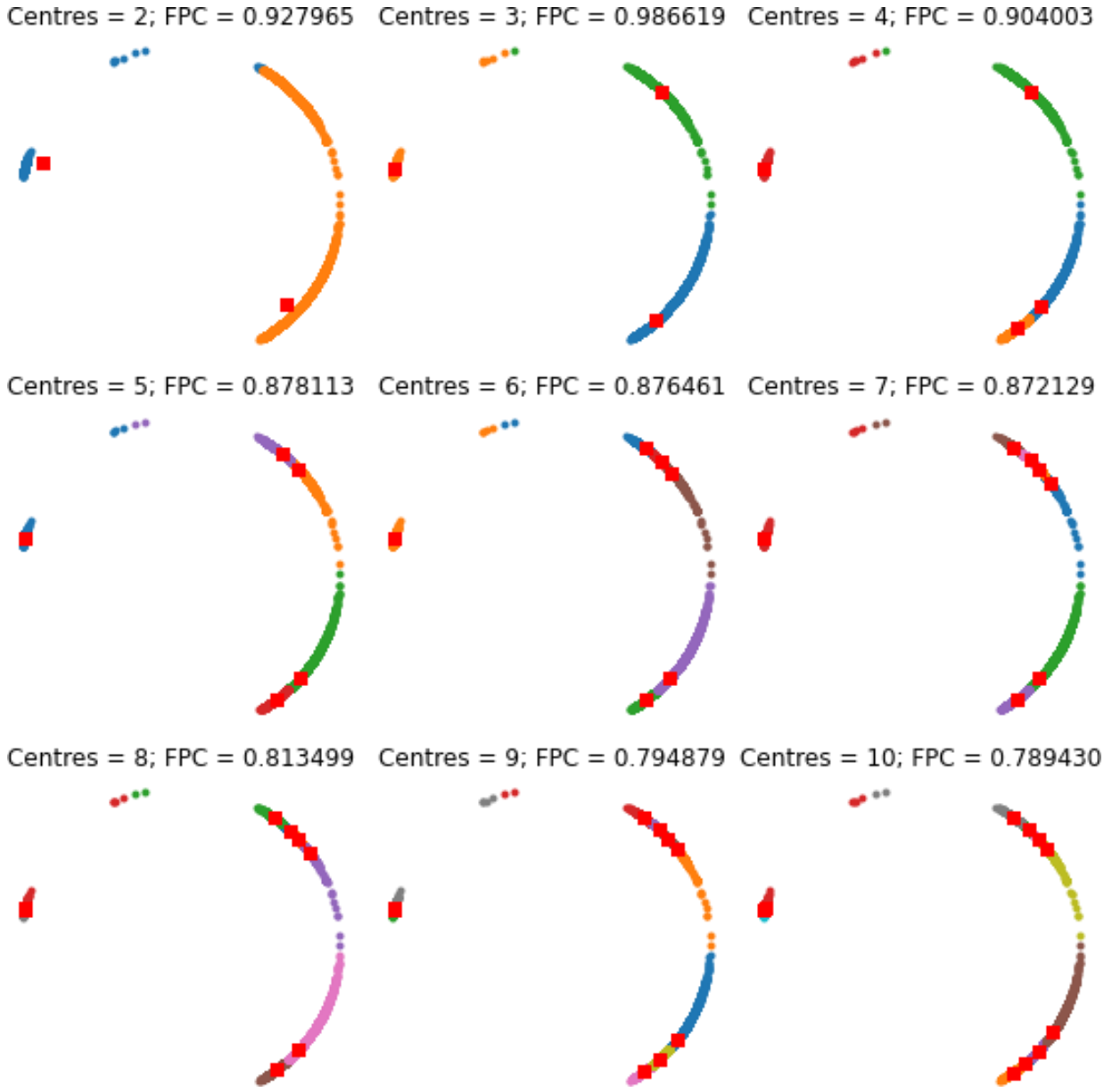

Рисунок 3.6 – Результат перевірки моделі

Тобто дослідження показує, що ефективність методу нечіткої кластеризації індикаторів компромісу при умові, що було задано 3 кластера на вхід – 98%, що показано на рисунку 3.7.

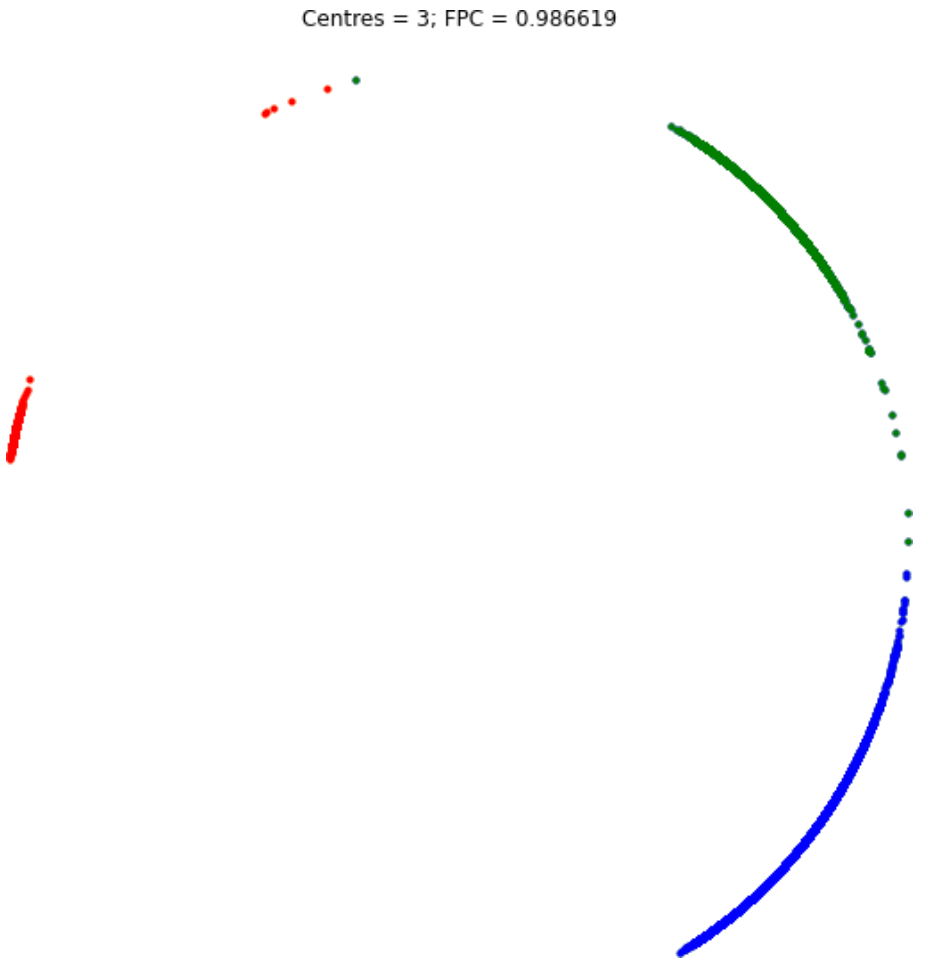

Рисунок 3.7 – Візуалізація нечіткої кластеризації за умовою, що кількість кластерів дорівнює трьом

Так як для детектування фішингових сайтів потрібно визначати тільки сайти фішингові і легальні, потрібно мати 2 кластера, а не 3. Для цього пропонується об’єднувати найближчі кластери до того моменту, поки не стане 2 кластери.

В даному випадку було отримано відображення кластеризації з ефективністю 98%, яке показано на рисунку 3.8, де червоні точки представляють фішингові веб-сайти, а зелені – легальні вебсайти.

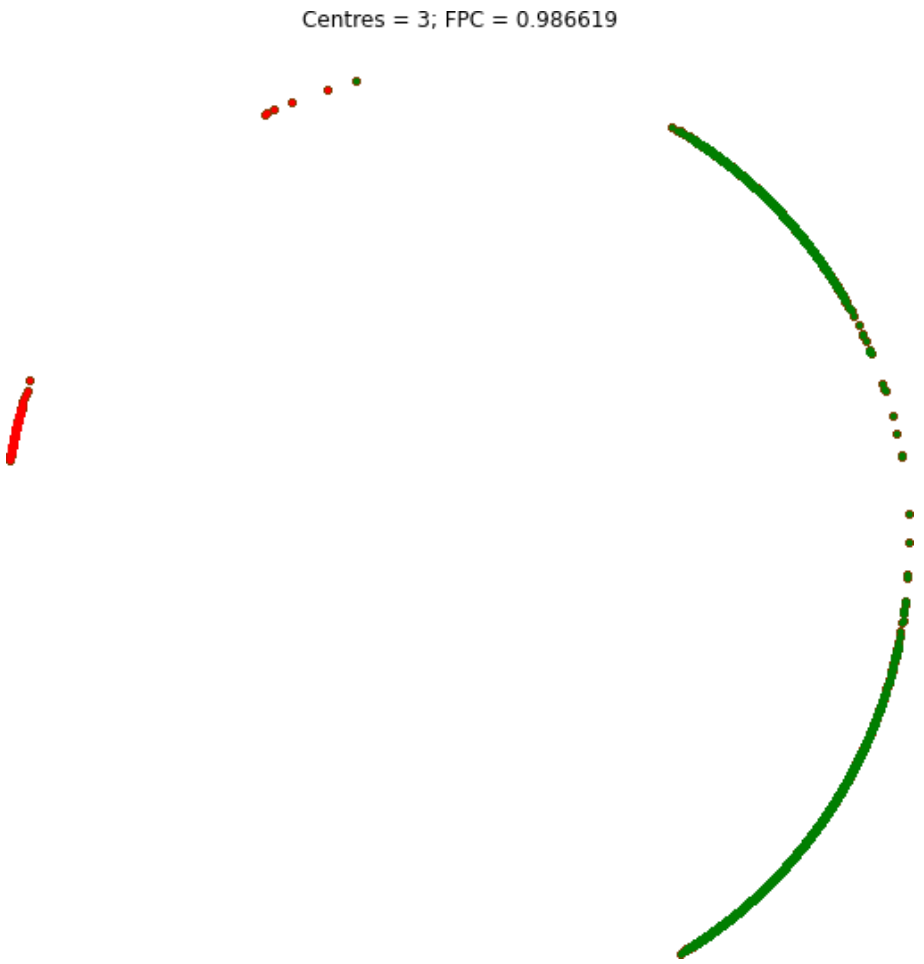

Рисунок 3.8 – Результат запропонованої моделі нечіткої кластеризації індикаторів компрометації

3.2 Порівняння з іншими методами

Для того, щоб порівняти запропонований метод з іншими широковідомими, такими як, побудованими на базі алгоритмів Naïve Bayes, Logical Regression, Random Forest, Decision tree, були імплементовані всі вищеперераховані моделі класифікації і кластеризації та протестовані на тому ж наборі даних.

Зазвичай для моделей керованого навчання при імплементації можна користуватись алгоритмом дій, який показаний на рисунку 3.9.

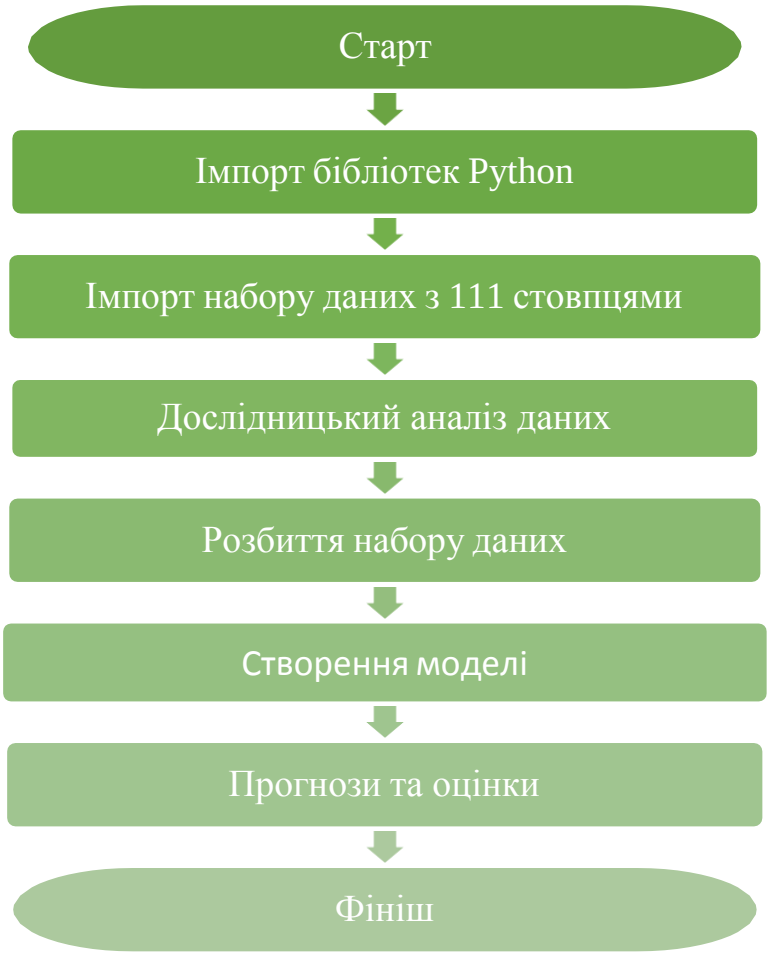

Рисунок 3.9 – Алгоритм побудови моделей керованого навчання

При імплементації алгоритмів Naïve Bayes, Logical Regression, Random Forest, Decision tree був використаний алгоритм, показаний на рисунку 3.9. В самій програмі зазвичай відрізнявся тільки крок зі створенням моделі.

3.2.1 Алгоритм Naïve Bayes

Naive Bayes – це алгоритм класифікації, який підходить для бінарної та багатокласової класифікації. Наївний Байєс добре працює у випадках категорійних вхідних змінних порівняно з числовими змінними. Як наведено на рисунку 3.9, для імплементування алгоритму керованного навчання Naive Bayes потрібно імпортувати бібліотеки Python та набір даних, провести дослідницький аналіз даних та розбити набір даних, створити модель та провести прогнози та оцінки.

Для початку було імпортовано необхідні для роботи алгоритму Naive Bayes бібліотеки Python та зчитано набір даних з виключенням останнього стовбця, який ідентифікує фішингові та легитимні сайти. Код імпорту бібліотек показаний у лістингу 11, а зчитування даних у лістингу 2 та 3 (ідентичні кроки).

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, f1_score
from sklearn.naive_bayes import GaussianNB
```

Лістинг 11

Дослідницький аналіз даних (EDA) проводиться для виявлення нульових значень, що впливає на ефективність моделі. Код EDA показаний у лістингу 12, а візуалізація нульового значення на рисунку 3.10.

```
sns.heatmap(dataset.isnull())
```

Лістинг 12

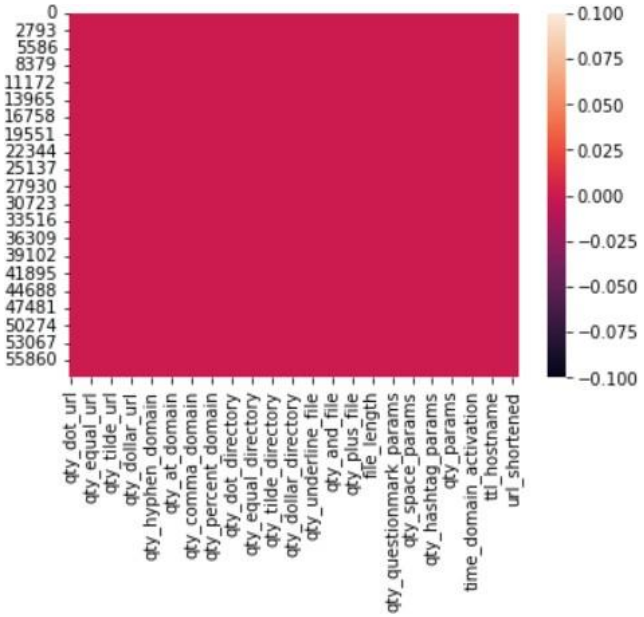

Рисунок 3.10 – Візуалізація дослідницького аналізу даних

Далі було використано метод train_test_split для поділу даних на 2 групи: групу для навчання та групу для тестування моделі. Лістинг цього кроку наведено у лістингу 13.

```
x = dataset.iloc[ : , :-1]
y = dataset.iloc[:, -1:]
XTrain, XTest, YTrain, YTest = train_test_split(x,y,test_size=0.25)
```

Лістинг 13

Нарешті можна створити модель Naïve Bayes, що показано на лістингу 14 та визначити її ефективність за допомогою матриці плутанини та F1 score для визначення точності моделі, що показано на лістингу 15 та рисунку 3.11

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(XTrain, YTrain)
YPred = classifier.predict(XTest)
```

Лістинг 14

```
confusionmatrix = confusion_matrix(YTest, YPred)
f1 = (f1_score(y_pred, y_test))*100
groupnames = ['Negative','FalsePositive', 'FalseNegative','Positive']
groupcounts = ['{0:0.0f}'.format(value) for value in cm.flatten()]
grouppercentages = ['{0:.2%}'.format(value) for value in
confusionmatrix.flatten()/np.sum(confusionmatrix)]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
zip(groupnames,groupcounts,grouppercentages)]
labels = np.asarray(labels).reshape(2,2)
print('Accuracy: {:.2f}%'.format(f1))
print('\nConfusion Matrix:\n {}'.format(cm))
sns.heatmap(cm, annot=labels, fmt='', cmap='Reds')
```

Лістинг 15

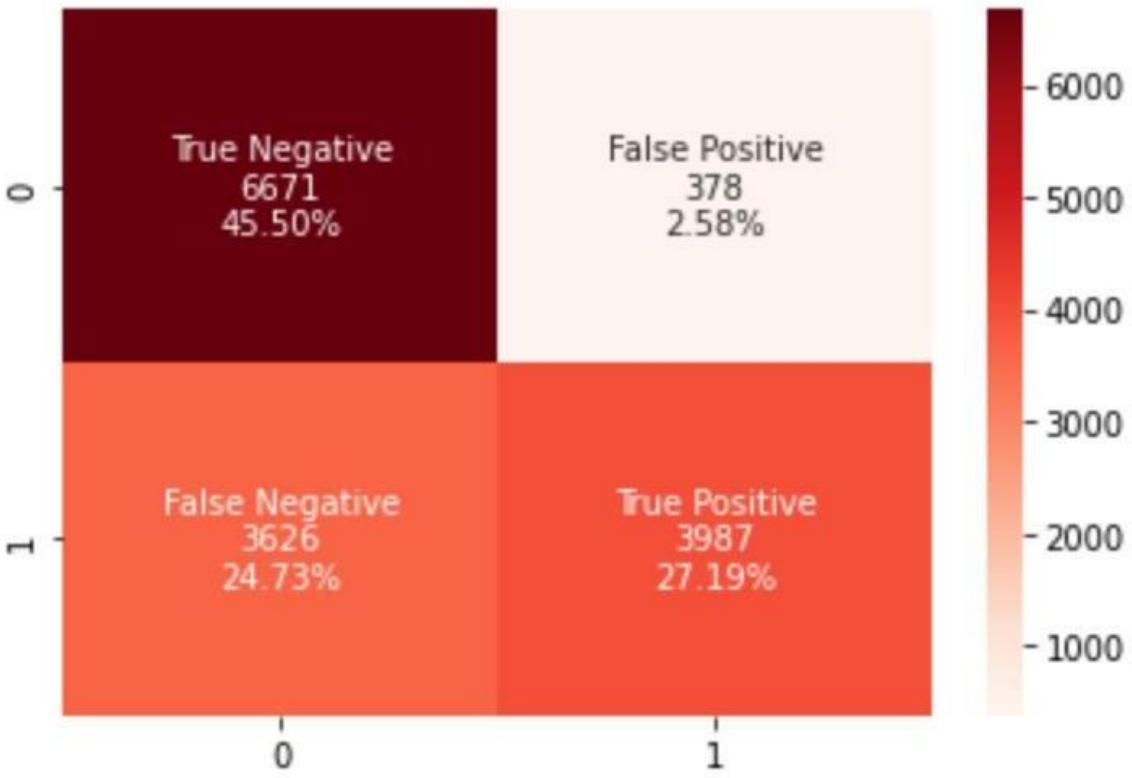

Рисунок 3.11 – Матриця плутанини для моделі Naive Bayes

Точність матриці плутанини обчислюється шляхом додавання 6709+3932 і ділення на загальну кількість або записи. Точність роботи алгоритму Naive Bayes в даному дослідженні = (6709+3932)/14662 = 72,57%.

3.2.2 Алгоритм Logical Regression

У статистиці логістична регресія використовується для прогнозування ймовірності певного класу чи події, наприклад, ймовірність перемоги команди, здоров’я пацієнта тощо. Це можна розширити для моделювання кількох класів подій, наприклад визначення того, чи містить зображення кішка, собака, лев тощо.

Для імплементування алгоритму керованного навчання логістичної регресії потрібно виконати наступні кроки, показані на рисунку 3.7.

Лістинги кроків 2-3, 6 ідентичні і були показані у лістингах 2, 3, 12 та 13 та на рисунках 3.1 і 3.8.

У лістингу 16 показано імпорт бібліотек python.

```
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, f1_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

Лістинг 16

Після виконання кроків 1-4 йде створення моделі, яке показано у лістингу 17.

```
classifier = LogisticRegression(random_state = 0)
classifier.fit(XTrain, YTrain)
YPred = classifier.predict(XTest)
```

Лістинг 17

Ефективність методу визначається матрицею плутанини, вона показана на рисунку 3.12.

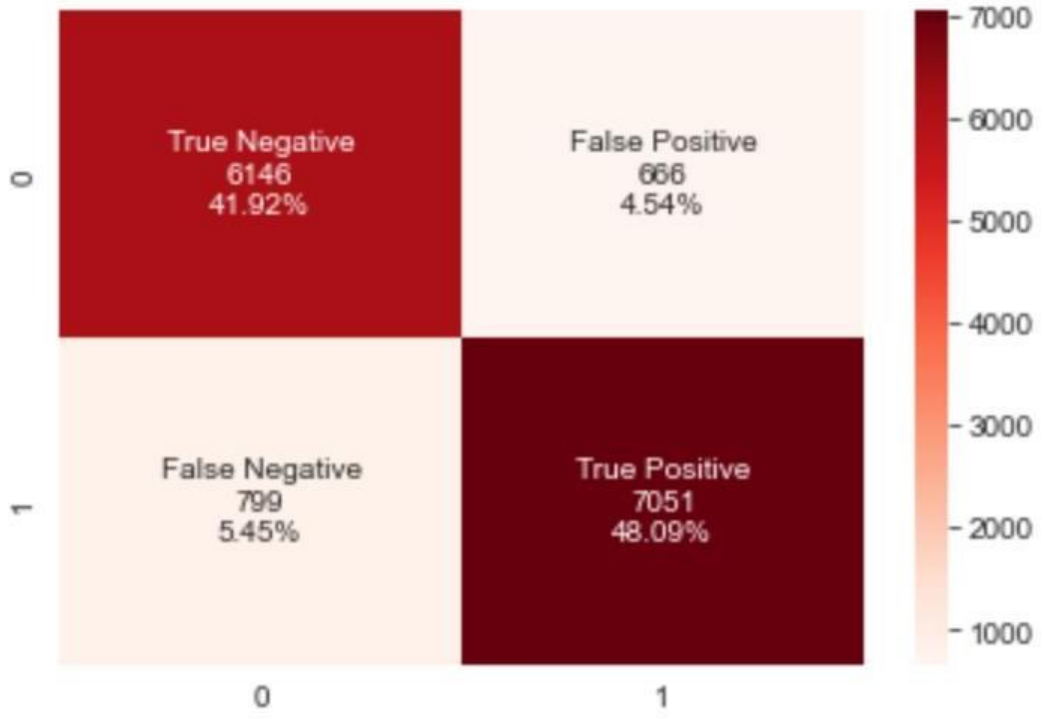

Рисунок 3.12 – Матриця плутанини для моделі логістичної регресії

Точність матриці плутанини обчислюється шляхом додавання 6146+7051 і ділення на загальну кількість чи записів. Точність алгоритму логістичної регресії в даному дослідженні = (6146+7051)/14662 = 90,01%

3.2.4 Алгоритм Random Forest і Decision tree

Дерево рішень є найпотужнішим і популярним інструментом для класифікації та прогнозування. Дерево рішень — це інструмент підтримки прийняття рішень, який використовує деревоподібну модель рішень та їх можливих наслідків, зокрема випадкові результати, витрати ресурсів та корисність.

Випадковий ліс — це контрольований алгоритм машинного навчання, який широко використовується в задачах класифікації та регресії. Він будує дерева рішень на різних вибірках і приймає їх більшість голосів за класифікацію та середнє значення у разі регресії.

Для імплементування алгоритму керованного навчання дерева рішень та випадкового лісу потрібно виконати кроки, показані на рисунку 3.7.

Лістинги кроків 2-3, 6 ідентичні і були показані у лістингах 2, 3, 12 та 13 та на рисунках 3.1 і 3.8.

У лістингу 18 показано імпорт бібліотек python:

```
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, f1_score
from sklearn.tree import DecisionTreeClassifier
```

Лістинг 18

Після виконання кроків 1-4 йде створення моделі, яке показано у лістингу 19.

classifier = DecisionTreeClassifier()

classifier.fit(XTrain, YTrain)

YPred = classifier.predict(XTest)

Лістинг 19

Ефективність методу визначається матрицею плутанини та F1 score, вона показана на рисунку 3.13.

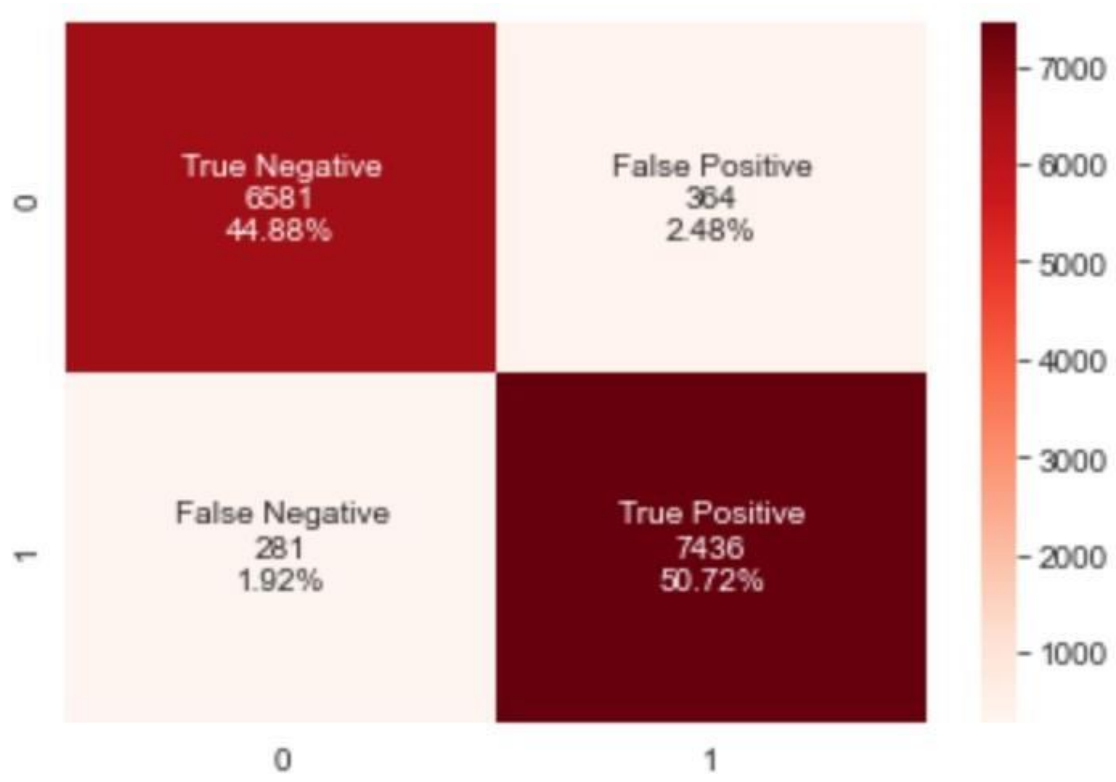

Рисунок 3.13 – Матриця плутанини для моделей Random Forest & Decision tree

Точність матриці плутанини обчислюється шляхом додавання 6581+7436 і ділення на загальну кількість або записи. Точність моделей Random Forest & Decision tree = (6581+7436)/14662 = 95,6%

3.2.5 Алгоритм SVM

"Support Vector Machine" (SVM) — це контрольований алгоритм машинного навчання, який можна використовувати як для класифікації, так і для задач регресії, однак він здебільшого використовується в задачах класифікації. В алгоритмі SVM

зображується кожен елемент даних як точка в n-вимірному просторі (де n — кількість властивостей), при цьому значення кожної характеристики є значенням певної координати. Потім виконується класифікація, знаходячи гіперплощину, яка дуже добре розрізняє два класи.

Для імплементування алгоритму керованного навчання дерева рішень та випадкового лісу потрібно виконати кроки, наведені на рисунку 3.7, а саме імпортувати бібліотеки Python та набір даних, провести дослідницький аналіз даних та розбити набір даних, створити модель та провести прогнози та оцінки.

Лістинги кроків 2-3, 6 ідентичні і були показані у лістингах 2, 3, 12 та 13 та на рисунках 3.1 і 3.8.

У лістингу 20 показано імпорт бібліотек python:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, f1_score
from sklearn.svm import SVC
```

Лістинг 20

Після виконання кроків 1-4 йде створення моделі, яке показано у лістингу 19.

```
from sklearn.svm import SVC
classifier = SVC()
classifier.fit(XTrain, YTrain)
```

Лістинг 21

Ефективність методу визначається матрицею плутанини, вона показана на рисунку 3.14.

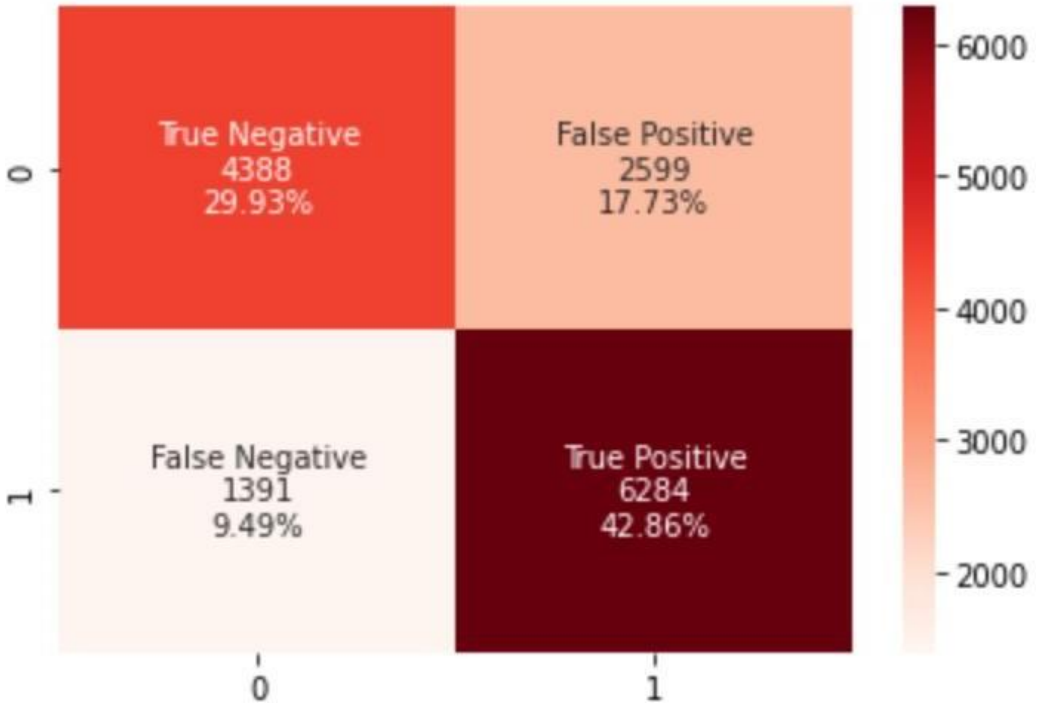

Рисунок 3.14 – Матриця плутанини для моделі SVM

Точність матриці плутанини обчислюється шляхом додавання 4388+6284 і ділення на загальну кількість або записи. Точність алгоритму SVM в даному дослідженні = (4388+6284)/14662 = 72,78%

3.2.6 Результати порівняння

Після імплементування і перевірки ефективності широковикористовуваних алгоритмів машинного навчання, таких як Naïve Bayes, Logistic Regression, Random Forest&Decision Tree, SVM, та запропонованої модифікації методу нечіткої кластеризації С-середніх індикаторів компромісу в задачі детектування фішингових сайтів, було отримано наступні результати:

- алгоритм С-середніх – 98%;
- алгоритм Naïve Bayes – 72,57%;
- алгоритм Logistic Regression – 90,01%;
- алгоритм Random Forest & Decision Tree –95,6% ;
- алгоритм SVM – 72,78%;

Результати також наведені на рисунку 3.15 для більш наглядного порівняння.

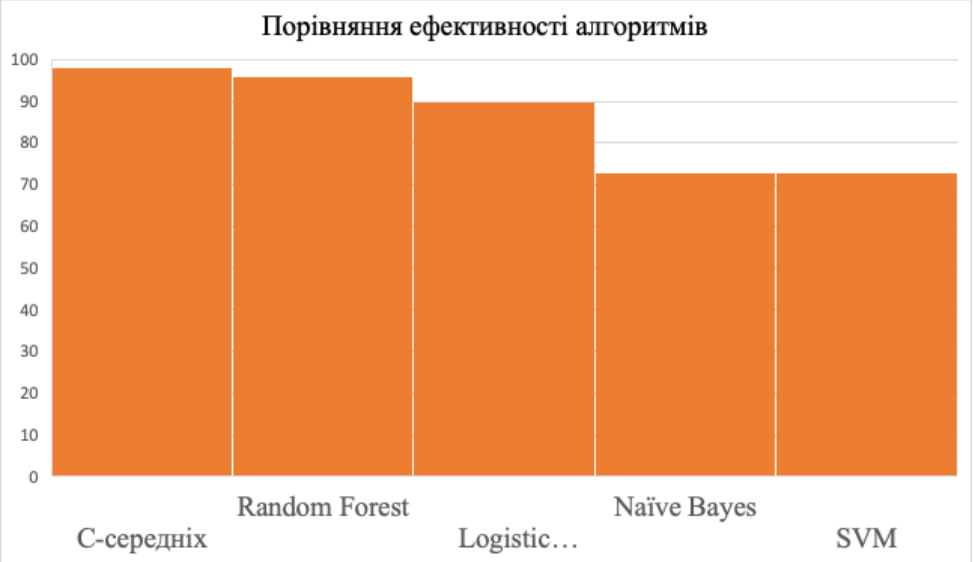

Рисунок 3.15 – Порівняння ефективності моделей класифікації і нечіткої кластеризації

Враховуючи той факт, що тестування алгоритмів нечіткої кластеризації С-середніх, дерева рішень, Naïve Bayes, SVM та логістичної регресії були проведені на одному й тому ж наборі даних, можна вважати результати порівняння достовірними.

Висновки за розділом 3

Було імплементовано і протестовано вдосконалений алгоритм нечіткої кластеризації індикаторів компромісу С-середніх для детектування фішингових сайтів з використанням набору даних, який складається з 88 647 екземплярів, де 30 647 екземплярів позначені як фішинг, а 58 000 інших екземплярів позначені як легітимні. Цей набір даних має 112 ознак, 111 з яких були використані у якості індикаторів компрометації, а 1 останній стовбець, який ідентифікує фішингові сайти, був пропущений. За допомогою спрощення набору даних зі 111 ознак до двох

та побудови 9 графів, які у даному випадку показують результати кластеризації з різною кількість центрів від двох до 10, було імплементовано вдосконалену модель, яка показала результат з ефективністю 98% за умови, що набір даних був поділений на 2 кластери, в порівнянні зі звичайним алгоритмом нечіткої кластеризації індикаторів компрометації, що дає 92% ефективності.

Порівнявши результат ефективності дослідженої моделі нечіткої кластеризації з широковідомими алгоритмами детектування фішингових сайтів на основі машинного навчання з вчителем, було виявлено, що модель детектування фішингових сайтів на основі нечіткої кластеризації індикаторів компрометації, є найефективнішою, адже імплементувавши і протестувавши такі широковідомі методи класифікації, ерева рішень, Naïve Bayes, SVM та логістичної регресії, було виявлено, що вони мають ефективність 95,6%, 72,57%, 72,78% та 90,01% відповідно. Результати порівняння можна вважати достовірними, адже всі вищеперераховані моделі були протестовані на одному і тому ж наборі даних з 88 647 екземплярів.

Отже, досліджений метод детектування фішингових сайтів на основі нечіткої кластеризації індикаторів компрометації є ефективнішим за широкодоступні методи та має результативність 98%.

ВИСНОВКИ

Традиційне навчання користувачів виявляти фішингові сайти не здатне в повній мірі протистояти новим невідомим загрозам і технікам шахраїв. Детектування злочинних веб-сайтів за допомогою програмного забезпечення автоматизує цю діяльність, але сучасні методи детектування таких сайтів теж мають свої недоліки. Вони полягають в тому, що потрібно заздалегідь натренувати модель визначати фішингові сайти на основі набору даних з вже класифікованим набором легальних і нелегальних сайтів, не враховуючи те, що різновиди кіберзлочинів еволюціонують кожного дня. Через це потрібно постійно оновлювати тренувальну базу, що не є зручним і ефективним рішенням. Представлення методу, у якому детектування реалізоване на основі нечіткої кластеризації, що є методом некерованого машинного навчанням і забезпечує автоматизовану роботу алгоритму без втручання наглядача для навчання моделі стало основною ідеєю досліджуваної роботи.

Робота «Виявлення джихадизму в соціальних мережах за допомогою методів великих даних, що підтримуються графіками та нечітким кластеризацією» [94], являє собою перевірку дописів у Twitter для визначення лідерів терористичних мереж та їх послідовників на основі нечіткої кластеризації. Це доказує, що алгоритм нечіткої кластеризації можна імплементувати для детектування фішингових веб-сайтів.

Імплементацію і тестування моделі було проведено в 3 етапи. Спочатку було оброблено набір даних із 30 647 підтверджених фішингових URL-адрес з веб-сайту Phishtank і 58 000 законних URL-адрес. В наборі даних присутні 112 атрибутів, які в даній роботі вважаються індикаторами компрометації фішингових сайтів. Перша частина індикаторів заснована на значеннях атрибутів усього рядка URL-адреси, тоді як значення наступних чотирьох груп засновані на окремих її підрядках: домену, директорії, файлу та параметрів. Атрибути останньої частини засновані на

URL - метриках, а також на метриках зовнішніх служб, таких як індекс пошуку Google.

На основі цього набору даних була створена і протестована вдосконалена модель алгоритму нечіткої кластеризації С-середніх. Для цього було використано 111 із 112 ознак набору даних, де останній, упущений стовбець, відповідає за те, чи є сайт фішинговим або легальним. За допомогою спрощення набору даних зі 111 стовпців, де кожний стовпець відповідає за кожну ознаку, до двох стовпців та побудови 9 графів, які показують результати кластеризації з різною кількістю центрів від двох до 10, було імплементовано модель, яка показала, що за умови, якщо набір даних був поділений на 3 кластери, ефективність алгоритму досягає 98% та 92%, якщо кластерів було 2. За допомогою об'єднання найближчих кластерів, було підвищено ефективніть з 92% до 98% при діленні набору даних на 2 кластери: фішингові і легальні веб-сайти.

Порівнявши результат ефективності дослідженої моделі нечіткої кластеризації з широковідомими алгоритмами детектування фішингових сайтів на основі машинного навчання з вчителем, було виявлено, що запропонована модель детектування фішингових сайтів є найефективнішою, адже побудувавши і протестувавши такі широковідомі методи класифікації, дерева рішень, Naïve Bayes, SVM та логістичної регресії, було виявлено, що вони мають ефективність 95,6%, 72,57%, 72,78% та 90,01% відповідно, а їх візуальне порівняння наведено на рисунку 3.16. Результати порівняння можна вважати достовірними, адже всі вищеперераховані моделі були протестовані на одному і тому ж наборі даних з 88 647 екземплярів.

При проведенні дослідження були вирішені такі *завдання***,** як розгляд сучасних підходів і технік детектування фішингових сайтів, проведення аналізу існуючих робіт детектування фішингових сайтів на основі машинного навчання, імплементування методу нечіткої кластеризації С-середніх з метою детектування фішингових сайтів, визначення його ефективності та порівняння з іншими широковідомими методами детектування фішингових сайтів за допомогою машинного навчання.

Результати практичної частини можуть бути використані будь-яким вендором, адже використання ефективного алгоритму нечіткої кластеризації індикаторів компромісу автоматизує процес прийняття рішень щодо детектування фішингових сайтів, а ефективність методу досягає 98%.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bissell, G. Fox, J. LaSalle, R. Cin, P. How aligning security and the business creates cyber resilience [Electronic resource] / Kelly B., Jacky F., Ryan L., Paolo C. // State of Cybersecurity Resilience 2021 – 2021. – 8 p.

2. Офіційний сайт Національної поліції України [Електронний ресурс]. – Режим доступу: https://www.npu.gov.ua

3. Про основні засади забезпечення кібербезпеки України [Електронний ресурс]: закон України від 05.10.2017 No 2163-VIII-ВР. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2163-19#Text

4. Конвенція про кіберзлочинність [Електронний ресурс]: конвенція Європи від 23.11.2001 No 2163-VIII-ВР. – Режим доступу: https://zakon.rada.gov.ua/laws/show/994-575 - Text

5. Countries most targeted by phishing attacks worldwide in 2021 [Electronic resource]. – Access: https://www.statista.com/statistics/266362/phishing-attacks-country/

6. Nieles, M; Dempsey, K; Pillitteri, V. An Introduction to Information Security [Electronic resource] / Michael N.; Kelley D,; Victoria P. // NIST Special Publication 800-12 Revision 1 – 2017 – 221 p. – Access: https://datatracker.ietf.org/doc/html/rfc4949

7. Iraj Sadegh Amiri, O.A. Akanbi, E. A Machine-Learning Approach to Phishing Detection and Defense [Electronic resource] / Iraj Sadegh Amiri, Akanbi , O.A., E. – Access: https://books.google.it/books/about/A_Machine_Learning_Approach_to_Phishing

8. Zetter, K. Inside the Cunning, Unprecedented Hack of Ukraine's Power Grid [Electronic resource] – Access: https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/

9. Alabdan R. Phishing Attacks Survey: Types, Vectors, and Technical Approaches [Electronic resource] – Access: https://www.mdpi.com/1999-5903/12/10/168/htm

10. How employees interact with malicious emails 2021 [Electronic resource]. – Access: https://www.statista.com/statistics/1253129/user-interaction-malicious-emails/

11. Number of unique phishing sites detected worldwide from 3rd quarter 2013 to 1st Quarter 2021 [Electronic resource]. – Access: https://www.statista.com/statistics/266155/number-of-phishing-domain-names-worldwide/

12. Jakobsson, M. The Human Factor in Phishing. / Jakobsson, M // Priv. Secur. Consum. Inf. – 2007 – p. 7, 1–19. – Acess: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.68.8721&rep=rep1&type=pdf

13. Caputo, D.D.; Pfleeger, S.L.; Freeman, J.D.; Johnson, M.E. Going spear phishing: Exploring embedded training and awareness. / D. Caputo; S.L. Pfleeger; J.D. Freeman; M.E. Johnson // IEEE Secur. Priv. – 2014 – p. 12, 28–38 – Access: https://ieeexplore.ieee.org/document/6585241

14. Hämmerli, B. Detection of Intrusions and Malware, and Vulnerability Assessment [Electronic resource] / B. Hämmerli – Access: https://books.google.it/books

15. Jakobson M., Myers S. Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft / M. Jakobson , S. Myers // Wiley, Hoboken, NJ, USA – 2006 Access: https://www.researchgate.net/profile/Robert-Rittenhouse/publication/296916234_Phishing_Attacks_and_Defenses

16. Abbasi, A., Chen, H. A comparison of fraud cues and classification methods for fake escrow website detection [Electronic resource] /A. Abbasi, H. Chen // Information Technology and Management, 10(2) – 2009 – p. 83-101

17. Abbasi, A., Chen, H. A comparison of tools for detecting fake websites /A. Abbasi, H. Chen // IEEE Computer, 42(10) – 2009 – p. 78-86

18. Abbasi, A., Zahedi, F. M., Kaza, S. Detecting fake medical websites using recursive trust labeling / A. Abbasi, F. M. Zahedi. S. Kaza // ACM Transactions on Information Systems, 30(4) – 2012 – p. 1-36

19. Chua, C. E. H., Wareham, J. Fighting Internet auction fraud: An assessment and proposal [Electronic resource] / C. E. H. Chua, J. Wareham // IEEE Computer, 37(10) – 2004 – p. 31-37 – Access: https://ieeexplore.ieee.org/document/1350724

20. Willis, P. Fake anti-virus software catches 43 million users' credit cards [Electronic resource] / P. Willis // Digital Journal – 2009 – Access: www.digitaljournal.com/article/280746

21. Joint Task Force. Security and Privacy Controls for Information Systems and Organizations [Electronic resource] / J.T. Force // NIST Special Publication 800-53 Revision 5 – 2020 – Access: https://csf.tools/reference/nist-sp-800-53/r5/si/si-4/si-4-24/

22. Indicator of Compromise (IoC) [Electronic resource]. – Access: https://encyclopedia.kaspersky.com/glossary/indicator-of-compromise-ioc/

23. Ільєнко А.В., Кохан Є.Р Методи та засоби виявлення фішингових вебсайтів / А.В. Ільєнко, Є. Р. Кохан // IV Міжнародна науково-практична конференція – 2021 – 37-39 с

24. Gaura V., Madhuresh M,. Anurag J. Anti-Phishing Techniques: A Review / V. Gaura , M. Madhuresh, J. Anurag // International Journal of Engineering Research and Applications (IJERA), 2 – p. 350-355

25. Chuanxiong G., Chen J., Online Detection and Prevention of Phishing Attacks [Electronic resource].г / G. Chuanxiong, J. Chen // ResearchGate – 2006 – Access: https://www.researchgate.net/profile/Chuanxiong-Guo

26. Tout H., Hafner W. Phishpin: An identity-based anti-phishing approach / H. Tout, W. Hafner // Computational Science and Engineering – 2009 – p. 347-352

27. Afroz S., Greenstadt R., Phishzoo: An automated web phishing detection approach based on profiling and fuzzy matching. / S. Afroz, R. Greenstadt // Technical Report DU-CS-09-03, Drexel University – 2009.

28. Ma J., Saul L. K., Savage S., Voelker G. M.. Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs / J. Ma, L. K. Saul, S. Savage, G. M. Voelker // KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining – 2009 – p 1245–1254.

29. Ma J., Saul L. K., Savage S., Voelker G. M.. Identifying Suspicious URLs: An Application of Large-Scale Online Learning / J. Ma, L. K. Saul, S. Savage, G. M. Voelker // ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning – 2009 – p 681–688.

30. Moore T., Clayton R. The Economics of Online Crime / T. Moore, R. Clayton // Journal of Economic Perspectives—Volume 23, Number 3 —2009 – p 3–20

31. Chua C. E. H., Wareham, J. Fighting Internet auction fraud: An assessment and proposal / C. E. H. Chua, J Wareham // . IEEE Computer, 37(10) – p. 31-37.

32. Herzberg, A., Jbara, A. Security and identification indicators for browsers against spoofing and phishing attacks /A. Herzberg, A Jbara // ACM Transactions on Internet Technology, 8(4) – 2008 – p 1-36

33. Kumaraguru, P. A system for educating users about semantic attacks / P Kumaraguru // Carnegie Mellon University.SoCC – 2009

34. Herzberg A., Gbara A. TrustBar: Protecting (even Naive) Web Users from Spoofing and Phishing Attacks / A. Herzberg, A. Gbara // DIMACS Technical Report 2004-23 – 2004 – https://www.researchgate.net/publication/243786849

35. Chou N., Ledesma R., Teraguchi Y., Boneh D., Mitchell J. C. Client-side defense against Web-based identity theft / N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, J. C. Mitchell // In Proceedings of the Network and Distributed System Security Symposium – 2004

36. Zhang Y., Egelman S., Cranor L., Hong J. Phinding phish: Evaluating anti-phishing tools / Y. Zhang, S. Egelman, L. Cranor, J. Hong // In Proceedings of the 14th Annual Network and Distributed System Security Symposium – 2007

37. AntiPhishing Filter [Електронний ресурс] / Режим доступу: https://google.com/ru//pubs/archive/35580.pdf

38. Aburrous M., Alamgir M., Prasad K.. Intelligent Phishing Website Detection System using Fuzzy Techniques / M. Aburrous, M. Alamgir Hossain, K. Prasad Dahal // Conference: Information and Communication Technologies: From Theory to Applications – 2008 – p. 1-6

39. Dunlop M., Shelly D., Groat S. GoldPhish: Using Images for Content-Based Phishing Analysis [Electronic resource] / M. Dunlop, D. Shelly, S. Groat // Internet Monitoring and Protection (ICIMP) – 2010 – p. 123 – 128 – Access: https://www.researchgate.net/publication/224142945_GoldPhish_Using_Images_for_Content-Based_Phishing_Analysis

40. Wenyin L., Huang G., Xiaoyue L., Min Z., Deng X. Detection of Phishing Webpages based on Visual Similarity / L. Wenyin, G. Huang, L. Xiaoyue, Z. Min, X. Deng // 14th International Conference on World Wide Web (WWW): Special Interest Tracks and Posters – 2005

41. Ronda T., Saroiu S.,Wolman A. Itrustpage: a user-assisted anti-phishing tool / T. Ronda, S. Saroiu, A. Wolman // ACM SIGOPS Operating Systems Review, Vol. 42 – 2008 – pp 261-272

42. Rao R.S, Ali S.T., PhishShield: A Desktop Application to Detect Phishing Webpages through Heuristic Approach [Electronic resource] / R.S. Rao, S.T. Ali // Procedia Computer Science Volume 54 – 2015 – pp. 147-156 – Access: https://doi.org/10.1016/j.procs.2015.06.017

43. Norman M. Sadeh, Anthony Tomasic, Ian Fette. Learning to Detect Phishing Emails / Norman M. Sadeh, Anthony Tomasic, Ian Fette // Proceedings of the 16th International Conference on World Wide Web – 2007 – pp. 649-656

44. Jordan M. I., Mitchel T. M. Machine learning: Trends, perspectives, and prospects [Electronic resource] / M. I. Jordan, T. M. Mitchel // Science Vol. 349 – 2015 – pp. 255-260 – Access: https://www.cs.cmu.edu/~tom/pubs/Science-ML-2015.pdf

45. Mohri M., Rostamizadeh A., Talwalkar A., Foundations of Machine Learning, second edition / M. Mohri, A. Rostamizadeh, A. Talwalkar // 2018 – Access: https://books.google.it/

46. Nichols J.A., Chan H.W.H., Baker M.A.B. Machine learning: applications of artificial intelligence to imaging and diagnosis [Electronic resource] / J.A. Nichols, H.W.H. Chan, M.A.B. Baker // Biophysical Reviews – 2019 – pp. 111-118 – Access: https://doi.org/10.1007/s12551-018-0449-9

47. Іваськів І.С. Методи машинного навчання в задачах виявлення нетипової поведінки складної системи / І.С. Іваськів // Магістерська робота – 2017

48. Harrington P. Machine Learning in Action [Electronic resource] / P. Harrington // Manning Publications Co. – 2012 – Access: http://www2.ift.ulaval.ca/~chaib/IFT-4102-7025/public_html/Fichiers/Machine_Learning_in_Action.pdf

49. Greene D., Cunningham P., Mayer R. Unsupervised learning and clustering / D. Greene, P. Cunningham, R. Mayer // Machine Learning Techniques for Multimedia (M. Cord and P. Cunningham, eds.) –2008 – pp. 51-90

50. J. Alzubi1, A. Nayyar, A. Kumar. Machine Learning from Theory to Algorithms: An Overview [Electronic resource] / J. Alzubi1, A. Nayyar, A. Kumar // Second National Conference on Computational Intelligence – 2018 – Access: https://iopscience.iop.org/article/10.1088/1742-6596/1142/1/012012/pdf

51. Cunningham P., Cord M., Delany S.J. Supervise Learning / P. Cunningham, M. Cord, S. J. Delany // Cognitive technologies – 2008 – pp. 24-49 – Access: https://link.springer.com/chapter/10.1007/978-3-540-75171-7_2

52. Liu B. Supervised Learning [Electronic resource] / B. Liu // Data Centrics systems and applications – 2011 – pp. 63-132 – Access: https://link.springer.com/chapter/10.1007/978-3-642-19460-3_3

53. Bonaccorso G. Machine learning algorithms / G. Bonaccorso // Packt Publishing Ltd – 2017 – pp 10-14

54. Niculescu-Mizil A., Caruana R. Predicting good probabilities with supervised learning / A. Niculescu-Mizil, R. Caruana // ICML '05: Proceedings of the 22nd international conference on Machine learning – 2005 – pp. 625-632

55. Liu B. Supervised Learning [Electronic resource] / B. Liu // Data Centrics systems and applications – 2011 – pp. 63-132 – Access: https://link.springer.com/chapter/10.1007/978-3-642-19460-3_3

56. Niculescu-Mizil A., Caruana R. Predicting good probabilities with supervised learning / A. Niculescu-Mizil, R. Caruana // ICML '05: Proceedings of the 22nd international conference on Machine learning – 2005 – pp. 625-632

57. Hastie T., Tibshirani R., Friedman J. H. Overview of Supervised learning / T. Hastie, R. Tibshirani, J. H. Friedman // The elements of statistical learning – 2001 – pp. 9-41

58. Rong S., Bao-wen S. The research of regression model in machine learning field [Electronic resource] / S. Rong, S. Bao-wen // MATEC Web of Conferences 176 – 2018 – Access: https://doi.org/10.1051/matecconf/201817601033

59. Sathya R., Nivas J., Abraham A., Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification / R. Sathya, J. Nivas, A. Abraham // International Journal of Advanced Research in Artificial Intelligence – 2013 – pp. 34-38

60. Osisanwo F.Y., Akinsola J.E.T., Awodele O., Hinmikaiye J. O., Olakanmi O. ,Akinjobi J., Supervised Machine Learning Algorithms: Classification and Comparison / F.Y. Osisanwo, J.E.T. Akinsola, O. Awodele, J. O. Hinmikaiye, O. Olakanmi, J. Akinjobi // International Journal of Computer Trends and Technology (IJCTT) – Volume 48 – 2017 – pp.128-138

61. Elder, J., Introduction to Machine Learning and Pattern Recognition [Electronic resource] / J. Elder // Access: http://www.eecs.yorku.ca/course_archive/2011-12/F/4404-5327/lectures/01%20Introduction.pd

62. Taiwo, O. A., Types of Machine Learning Algorithms, New Advances in Machine Learning / O.A. Taiwo // ISBN: 978-953-307-034-6, InTech, – 2010 – pp 3-31

63. Vapnik V. N. The Nature of Statistical Learning Theory. (2nd ed.) [Electronic resource] / V. N. Vapnik // Springer Verlag – 1995 – pp 1-20 – Access: https://www.andrew.cmu.edu/user/kk3n/simplicity/vapnik2 000.pdf

64. Kotsiantis S. B. Supervised Machine Learning: A Review of Classification Techniques [Electronic resource] / S. B Kotsiantis // Informatica 31– 2007 – pp 249-268 – Access: http://wen.ijs.si/ojs2.4.3/index.php/informatica/article/download/148/140.

65. Newsom I. Data Analysis II: Logistic Regression [Electronic resource] / I. Newsom // 2015 – Access: http://web.pdx.edu/~newsomj/da2/ho_logistic.pdf

66. Rish I., An empirical study of the naive Bayes classifier [Electronic resource] / I. Rish / Access: https://www.cc.gatech.edu/home/isbell/classes/reading/papers/Rish.pdf

67. Domingos P., Pazzani M. On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning [Electronic resource] / P. Domingos, M. Pazzani // Copyright © 1997 Kluwer Academic Publishers – 1997 – pp. 103-130 – Access: http://disi.unitn.it/~p2p/RelatedWork/Matching/domingos9 7optimality.pdf

68. Hormozi H., Hormozi E., Nohooji H. R. The Classification of the Applicable Machine Learning Methods in Robot Manipulators [Electronic resource] / H. Hormozi, E. Hormozi, H. R. Nohooji // . International Journal of Machine Learning and Computing

(IJMLC), Vol. 2, No. 5, – 2012 – pp. 560-563 – Access: http://www.ijmlc.org/papers/189-C00244-001.pdf

69. Bishop C. M. Neural Networks for Pattern Recognition [Electronic resource] / C. M. Bishop // Clarendon Press, Oxford, England – 1995 – Access: http://cs.du.edu/~mitchell/mario_books/Neural_Networks_f or_Pattern_Recognition_-_Christopher_Bishop.pdf

70. Neocleous C., Schizas C. Artificial Neural Network Learning: A Comparative Review [Electronic resource] / C. Neocleous, C. Schizas // Lecture Notes in Computer Science, Volume 2308. Springer, Berlin, Heidelberg – 2002 – pp 300-313 – Access: https://link.springer.com/chapter/10.1007/3-540-46014- 4_27

71. Hastie T., Tibshirani R., Friedman J. H. The elements of statistical learning / T. Hastie, R. Tibshirani, J. H. Friedman // Data mining, inference, and prediction – 2001

72. Setiono R., Loew, W. K. An algorithm for fast extraction of rules from neural networks / R. Setiono, W. K. Loew // Applied Intelligence. 2000

73. Witten, I. H., Frank E. Data Mining: Practical machine learning tools and techniques (2nd ed.) [Electronic resource] / I. H. Witten, E. Frank // ISBN: 0- 12-088407-0, Morgan Kaufmann Publishers, San Francisco, CA, U.S.A. – 2005 – Access: ftp://93.63.40.27/pub/manuela.sbarra/Data Mining Practical Machine Learning Tools and Techniques - WEKA.pdf

74. Kassambara A. Machine learning clustering / A. Kassambara // STHDA – 2017

75. Bora D.J., Gupta A.K. A Comparative study Between Fuzzy Clustering Algorithm and Hard Clustering Algorithm [Electronic resource] / D.J. Bora, A.K. Gupta // International Journal of Computer Trends and Technology (IJCTT) – volume 10 number 2 – 2014 pp. 108-113 – Access: https://arxiv.org/pdf/1404.6059.pdf

76. Alex S., Vishwanathan S.V.N. Introduction to Machine Learning [Electronic resource] / S. Alex, S.V.N. Vishwanathan // Cambridge University Press 2008. ISBN – 2008 – Access: https://www.kth.se/social/upload/53a14887f276540ebc81ae c3/online.pdf Retrieved from website: http://alex.smola.org/drafts/thebook.pdf

77. Schapire R., Machine Learning Algorithms for Classifrication [Electronic resource] / R. Schapire // Access: http://dimacs.rutgers.edu/archive/Workshops/DataMiningTutorial/slides/schapire.pdf

78. Bellman R.E., Zudeh L.A., Decision-making in a fuzzy environment [Electronic resource] / R. E. Bellman, L. A. Zudeh // NGL-05-003-016 by Electronics research – 1970 – Access: https://ntrs.nasa.gov/api/citations/19700017250/downloads/19700017250.pdf

79. Кравець П., Киркало Р., Системи прийняття рішень з нечіткою логікою [Electronic resource] / П. Кравець, Р. Киркало // 2009 – Access: http://vlp.com.ua/files/special/17_0.pdf

80. Zadeh, L. A., Fuzzy Sets / L.A. Zadeh // Information and Control, Vol. 8 – 1965 – pp.338-353 https://ntrs.nasa.gov/api/citations/19700017250/downloads/19700017250.pdf

81. Zadeh, L. A., Toward a Theory of Fuzzy Systems / L.A. Zadeh // ERL Report No. 69-2 – 1969

82. Paunovic M., The Clastering in Analyzing Effect of the Presence of Patient Age, Infection and Systemic Disease of Connective Tissue on the Occurrence of Dehiscence Laparotomy/ M. Paunovic// ISSN 2687-7007 – 2018 – pp. 8-11

83. Cebeci Z., Yildiz F., Comparison of K-Means and Fuzzy C-Means Algorithms on Different Cluster Structures [Electronic resource] / Z. Cebeci, F. Yildiz // Journal of Agricultural Informatics (ISSN 2061-862X) – 2015 – pp.13-23 – Access: http://real.mtak.hu/29902/1/196_1015_1_PB_u.pdf

84. Velmurugan T. Performance Comparison between K-Means and Fuzzy C-Means Algorithms Using Arbitrary Data Points / T. Velmurugan // Wulfenia Journal, vol. 19, no. 8 – 2012 – pp. 234-241

85. Bezdek J.C., Pattern Recognition with Fuzzy Objective Function Algorithms / J.C. Bezdek // Plenum Press, New York., doi: 10.1007/978-1-4757-0450 – 1981

86. Dong W., Ren J.D., Zhang D. Hierarchical K-Means Clustering Algorithm Based on Silhoutte and Entropy / W. Dong, J.D. Ren, D. Zhang // AICI 2011, Part I, LNAI vol. 7002 2011 – pp. 339-347

87. Kaur M., Kaur U., 'Comparison Between K-Mean and Hierarchical Algorithm Using Query Redirection'/ M. Kaur, U. Kaur // Int. J. of Advanced Research in Computer Science and Software Engineering, vol. 3 , no. 7 2013 – pp. 1454-1459

88. Dunn J.C. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters' / J.C. Dunn // Cybernetics, vol.3, no.3 – 1973 – pp. 32-57

89. Wu K.L., Yang M.S., Alternative c-means clustering algorithms / K.L. Wu, M.S. Yang // Pattern Recognition Volume 35, Issue 10 – 2002 – pp. 2267-2278

90. Hungand M.C., Yang D.L., An Efficient Fuzzy C-Means Clustering Algorithm / M.C. Hungand, D.L. Yang // 2001

91. Fuzzy c-means clustering algorithm [Electronic resource] / Access: https://sites.google.com/site/dataclusteringalgorithms/fuzzy-c-means-clustering-algorithm

92. Dang Q.V., Studying the Fuzzy clustering algorithm for intrusion detection on the attacks to the Domain Name System [Electronic resource] / Q.V. Dang // World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4) – 2021 – Access: https://ieeexplore.ieee.org/abstract/document/9514038

93. Almomani A.; Gupta B.B.; Wan T.C.; Altaher A.; Manickam S., Phishing Dynamic Evolving Neural Fuzzy Framework for Online Detection "Zero-day" Phishing Email [Electronic resource] / A. Almomani; B.B. Gupta; T.C. Wan; A. Altaher; S. Manickam // Indian Journal of Science and Technology Vol. 6 – 2013 – pp. 122-126 – Access: https://arxiv.org/pdf/1302.0629.pdf

94. Sánchez-Rebollo C., Puente C., Palacios R., Piriz C., Fuentes H.P., Jarauta J., Detection of Jihadism in Social Networks Using Big Data Techniques Supported by Graphs and Fuzzy Clustering [Electronic resource] / C. Sánchez-Rebollo, C. Puente, R. Palacios, C. Piriz, H.P. Fuentes, J. Jarauta // Advances in Complex Systems and Their Applications to Cybersecurity – 2018

95. Kulkarni A.D., Brown L.L., Phishing Websites Detection using Machine Learning / Kulkarni A.D., L.L. Leonard // (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 10, No. 7 – 2019 – pp. 8-13 – Access: https://scholarworks.uttyler.edu/cgi/viewcontent.cgi?article=1022&context=compsci_fac

96. Sadeh N., Tomasic A., Fette I. Learning to detect phishing emails / N. Sadeh, A. Tomasic, I. Fette // Proceedings ofthe16thinternational conference on world wide web – 2007 – pp. 649-656

97. Ma J., Savag S.S., Voelker G.M., Learning to detect malicious URLs / J. Ma, S. S. Savag, G. M. Voelker // ACM Transactions on Intelligent Systems and technology, vol. 2, no. 9 – 2011 – pp. 30:1-30:24

98. Abdelhamid N., Ayesh A., Thabtah F., Phishing Detection based Associative Classification / N. Abdelhamid, A. Ayesh, F. Thabtah // Data Mining. Expert Systems with Applications (ESWA), vol. 41– 2014 – pp. 729-734

99. Hadi W., Aburrub F., and Alhawari S., A new fast associative classification algorithm for detecting phishing websites / W. Hadi, F. Aburrub, S. Alhawari // Applied Soft Computing vol. 48 – 2016 – pp. 729-734

100. Kuyama M., Kakizaki Y., Sasaki R., Method for detecting a malicious domain by using whois and dns features / M. Kuyama, Y. Kakizaki, R. Sasaki // The Third International Conference on Digital Security and Forensics (DigitalSec2016) – 2016 – p.74

101. Kulkarni A.D., Brown L.L., Phishing Websites Detection using Machine Learning / Kulkarni A.D., L.L. Leonard // (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 10, No. 7 – 2019 – pp. 8-13 – Access: https://scholarworks.uttyler.edu/cgi/viewcontent.cgi?article=1022&context=compsci_fac

102. Дерево рішень в машинному навчанні – [Електронний ресурс] – Режим доступу: https://towardsdatascience.com/decision-tree-in-machine-learning-e380942a4c96

103. Алгоритм випадкового лісу – [Електронний ресурс] – Режим доступу: https://builtin.com/data-science/random-forest-algorithm

104. Gudivada V.N., Rao D.L., Cognitive Computing: Theory and Applications / V.N. Gudivada, D.L. Rao // Handbook of Statistics – 2016

105. LaValley M.P. Logistic Regression [Electronic Resourse] / M.P. LaValley // CirculationVolume 117, Issue 18 – 2008 – pp. 2395-2399 – Access: https://www.ahajournals.org/doi/epub/10.1161/CIRCULATIONAHA.106.682658

106. Sung A.H., Mukkamala S., R. Basnet, Detection of phishing attacks: A Machine Learning Approach [Electronic Resourse] / A.H. Sung, S. Mukkamala, R. Basnet // Soft computing Applications in Indrustry – pp. 373-383 – Access: https://link.springer.com/chapter/10.1007/978-3-540-77465-5_19

107. Basnet R., Mukkamala S., Sung A.H., Detection of Phishing Attacks: A Machine Learning Approach / R. Basnet, S. Mukkamala, A.H. Sung // Soft Computing Applications in Industry – 2008 – pp 373-383.

108. Vrbanci G., Fister I., Podgoreleca V., Datasets for phishing websites detection [Electronic Resourse] / G. Vrbanci, I. Fister, V. Podgoreleca / Data in Brief 33 – 2020 – Access: http://www.iztok-jr-fister.eu/static/publications/288.pdf

109. Tests-lists [Electronic Resourse] / Access: https://github.com/citizenlab/test-lists/tree/master/lists

ДОДАТОК А

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Тези наукових доповідей:

1. Serhii Buchyk, Elina Piatashova. Phishing detection based on unsupervised learning method. VIII International conference «Information Technology and Implementation» (IT&I-2021) – Taras Shevchenko National University of Kyiv, December 1-3, 2021.

2. Бучик С.С., Пяташова Е.С. Ефективність методів класифікації індикаторів компрометації при детектуванні фішингових сайтів – 2022 – подано тези.

ДОДАТОК В

Лістинг алгоритму нечіткої кластеризації С-змінних індикаторів компрометації

```
import numpy as np, pandas as pd, os
import matplotlib.pyplot as plt
import skfuzzy as fuzz
import itertools
from sklearn.preprocessing import StandardScaler, Normalizer
from sklearn.decomposition import TruncatedSVD
from sklearn.metrics import confusion_matrix
import statsmodels.api as sm
import statsmodels.formula.api as smf
dataset = pd.read_csv("dataset_small.csv")
print(dataset.shape)
dataset.head()
scaler = StandardScaler()
x_std = scaler.fit_transform(dataset.iloc[:,:-1])
lsa = TruncatedSVD(2, algorithm='arpack')
dtm_lsa = lsa.fit_transform(x_std)
dtm_lsa = Normalizer(copy=False).fit_transform(dtm_lsa)
a = pd.DataFrame(dtm_lsa, columns = ["component1","component2"])
y=pd.DataFrame(dataset.iloc[:,111])
a['targets']=y
fig1, axes1 = plt.subplots(3,3,figsize=(8,8))
all_data = np.vstack((a['component1'], a['component2']))
fig2, axes2 = plt.subplots(1,1,figsize=(8,8))
fig3, axes3 = plt.subplots(1,1,figsize=(8,8))
maxfpc = 0
colors=['r','g','b','c','m','y','k', 'orange','Brown','ForestGreen']
```

```
for ncntrs, ax in enumerate(axes1.reshape(-1),2):
    cntr, u, jm, p, fpc = fuzz.cluster.cmeans(all_data, ncntrs, 2, error=0.005, maxiter=1000, init=None)
    fpcarray.append(fpc)
    clstr_membership = np.argmax(u, axis=0)
    for k in range(ncenters):
        if fpc > maxfpc:
            maxfpc = fpc
        ax.plot(a['component1'][clstr_membership==k], a['component2'][clstr_membership == j],'.', color=colors[k])
        for pt in cntr:
            ax.plot(pt[0], pt[1], 'rs')
        ax.set_title('Centres = {0}; FPC = {1:2f}'.format(ncntrs, fpc))
        ax.axis('off')
        if fpc == maxfpc:
            neededcentre = (j + 1)
fig1.tight_layout()
fig1.savefig('phishing_detection')
cntr, u, jm, p, fpc = fuzz.cluster.cmeans(all_data, neededcentre, 2, error=0.005, maxiter=1000, init=None)
clstr_membership = np.argmax(u, axis=0)
for label in range(neededcentre):
    axes2.plot(a['component1'][clstr_membership == label], a['component2'][clstr_membership == label],'.', color=colors[label])
axes2.set_title('Centres = {0}; FPC = {1:2f}'.format(neededcentre, fpc))
axes2.axis('off')
fig2.tight_layout()
fig2.savefig('phishing_detection_bestclusters')
```