

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

Кваліфікаційна бакалаврська робота
ЗАСТОСУВАННЯ МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ
ДАНИХ ДЛЯ ВИЯВЛЕННЯ КІБЕРЗАГРОЗ

Виконав:

студент IV курсу, групи ПМК-41с
напряму підготовки (спеціальності)
125 – "Кібербезпека"

(шифр і назва напряму підготовки, спеціальності)



(підпис)

Лесик В.Ю.

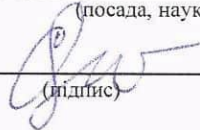
(прізвище та ініціали)



Керівник

Професор, доктор технічних наук

(посада, науковий ступінь та вчене звання)



(підпис)

Моркун Н.В.

(прізвище та ініціали)

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет Прикладної математики та інформатики _____

Кафедра

Кібербезпеки

Спеціальність 125 «Кібербезпека» _____

(шифр і назва)

«ЗАТВЕРДЖУЮ»

Завідувач
кафедри



"31" серпня 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

_____ (прізвище, ім'я, по батькові)

1. Тема роботи Застосування методів інтелектуального аналізу даних для виявлення кіберзагроз

керівник роботи Моркун Наталія Володимирівна, професор, д-р т. наук, професор,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затвердені Вченою радою факультету від " 13" вересня 2022 року № 15

2. Строк подання студентом роботи 13.06.2023р.

3. Вихідні дані до роботи статистичні дані кіберзагроз; інструменти та методи інтелектуального аналізу даних; експериментальні набори даних для практичного застосування інтелектуального аналізу даних

4. Зміст дипломної роботи (перелік питань, які потрібно розробити) Теоретичні основи кібербезпеки та технології інтелектуального аналізу даних, детальний опис практичного застосування ІАД у кібербезпеці, практичне застосування алгоритмів ІАД для розв'язання проблем кібербезпеки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) презентація Microsoft Power point

РЕФЕРАТ

Пояснювальна записка дипломного проекту складається зі вступу, трьох розділів, що містять 15 рисунків, висновків та списку використаних джерел з 53 найменувань. Загальний обсяг роботи становить 77 сторінок.

Об'єкт дослідження: Використання алгоритмів інтелектуального аналізу даних у кібербезпеці.

Метою даної роботи є дослідження кіберзагроз та їх класів. Дослідження методів ІАД та технік протидії шкідливому ПЗ з використанням цих методів. Практичне використання алгоритмів класифікації на реальних наборах даних з метою створення ефективної навченої моделі, яка буде ідентифікувати загрози на подібних наборах даних.

У першому розділі аналізуються типи кіберзагрози та модель OSI для розуміння рівнів мережі, на яких вони діють. Також, описується загальний принцип роботи алгоритмів ІАД.

У другому розділі описуються класи алгоритмів ІАД: класифікація, кластеризація, пошук аномалій тощо. Детальний опис найбільш популярних алгоритмів ІАД для кожного класу. Опис технік протидії шкідливому ПЗ, використовуючи методи ІАД. Опис метрик для оцінювання ефективності алгоритмів класифікації.

У третьому розділі визначається план дослідження обраних наборів даних. Описується проблема та виконується навчання моделей на підготовлених наборах даних. Результат дослідження є визначення найбільш ефективного алгоритму ІАД для класифікації кіберзагроз та наведення порівняльної характеристики методів, що використовуються.

Галузь застосування. Матеріали роботи можуть бути використанні для створення систем кібербезпеки, ідентифікації кіберзагроз чи інших робіт, пов'язаних з наукою про дані.

Ключові слова: ІАД, ОС, КІБЕРБЕЗПЕКА, КІБЕРЗЛОЧИНЕЦЬ, КІБЕРЗАГРОЗА, ПРОТОКОЛ, ВРАЗЛИВІСТЬ, ТЕЗНОЛОГІЯ, SVM, ПК, МЕРЕЖА, СИСТЕМА, КБ, ПЗ, ПРОГРАМА, ФІШИНГ, АТАКА, АЛГОРИТМ, КЛАСИФІКАЦІЯ, КЛАСТЕРИЗАЦІЯ, МЕТРИКИ, ОЗНАКА, ЦІЛЬОВА ЗМІННА, ПЕРЕНАВЧАННЯ, НАВЧАННЯ ПІД НАГЛЯДОМ.

ABSTRACT

The explanatory note of the diploma project consists of an introduction, three sections containing 15 figures, conclusions, and a list of 53 references. The total volume of the work is 77 pages.

Object of research: the use of data mining algorithms in cybersecurity.

The purpose of the diploma project is to study cyber threats and their classes. Research of data mining methods and techniques for countering malicious software using these methods. Practical application of classification algorithms on real datasets to create an effective trained model that will identify threats in similar datasets.

The first section analyzes the types of cyber threats and the OSI model to understand the network levels at which they operate. It also describes the general working principle of data mining algorithms.

The second section describes the classes of data mining algorithms: classification, clustering, anomaly detection, etc. It provides a detailed description of the most popular data mining algorithms for each class. Description of techniques for countering malicious software using data mining methods. Description of metrics for evaluating the performance of classification algorithms.

The third section defines the research plan for the selected datasets. It describes the problem and performs model training on prepared datasets. The research goal is to define the most effective data mining algorithm for classifying cyber threats and providing a comparative analysis of the used methods.

Field of application: The materials of this work can be used for creating cybersecurity systems, identifying cyber threats, or other works related to the field of data science.

Keywords: DATA MINING, OS, CYBERSECURITY, CYBERCRIMINAL, CYBER THREAT, PROTOCOL, VULNERABILITY, TECHNOLOGY, SVM, PC, NETWORK, SYSTEM, CS, SOFTWARE, PROGRAM, PHISHING, ATTACK, ALGORITHM, CLASSIFICATION, CLUSTERING, METRICS, FEATURE, TARGET VARIABLE, OVERFITTING, SUPERVISED LEARNING.

ЗМІСТ

ВСТУП.....	15
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ КІБЕРБЕЗПЕКИ ТА ТЕХНОЛОГІЇ «ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ».....	16
1.1 Важливість кібербезпеки у сучасному світі.....	16
1.2 Загальний опис інтелектуального аналізу даних.....	19
1.3 Опис існуючих кіберзагроз та джерел їх виникнення	24
Висновки до розділу 1	32
РОЗДІЛ 2. ДЕТАЛЬНИЙ ОПИС ПРАКТИЧНОГО ЗАСТОСУВАННЯ ІАД У КІБЕРБЕЗПЕЦІ.....	33
2.1 Загальний опис алгоритмів ІАД	33
2.2 Використання ІАД для виявлення шкідливого ПЗ.....	40
2.3 Алгоритми ІАД для практичного застосування	44
2.4 Метрики для оцінювання алгоритмів ІАД	55
Висновки до розділу 2	59
РОЗДІЛ 3. ПРАКТИЧНЕ ЗАСТОСУВАННЯ АЛГОРИТМІВ ІАД ДЛЯ РОЗВ'ЯЗАННЯ ПРОБЛЕМ КІБЕРБЕЗПЕКИ.....	60
3.1 Опис проблеми та практичне використання алгоритмів ІАД	60
3.2 Огляд та підготовка даних	61
3.3 Моделювання	68
3.4 Застосування метрик до отриманих моделей та порівняння результатів	70
Висновки до розділу 3	80
ВИСНОВКИ.....	81
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	83

ПОНЯТТЯ УМОВНИХ ПОЗНАЧЕНЬ

ІАД	Інтелектуальний аналіз даних
ОС	Операційна система
КБ	Кібербезпека
ПЗ	Програмне забезпечення
ПК	Персональний комп'ютер
OSI	Open Systems Interconnection
SQL	Structured Query Language
API	Application Programming Interface
SVM	Support Vector Machines
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
CART	Classification and Regression Tree
ECLAT	Equivalence Class Transformation
KNN	K-Nearest Neighbors
ANN	All-Nearest Neighbors

ВСТУП

Актуальність. Проблема кібербезпеки є дуже поширеною. Кіберпростір активно розвивається, формуючи нові тенденції загроз та боротьби з ними. Технології штучного інтелекту також стрімко розвиваються. Враховуючи високий потенціал алгоритмів інтелектуального аналізу даних, які є частиною наук про штучний інтелект, вони можуть змінити сучасні системи протидії вторгнень, збільшивши загальний рівень кібербезпеки комп'ютерних мереж.

Метою роботи є дослідження алгоритмів ІАД, типів кіберзагроз та технік боротьби з ними. Використання алгоритмів класифікації для визначення кіберзагроз на кількох наборах даних. Аналіз результатів з метою визначення найбільш ефективних алгоритмів класифікації.

Для вирішення поставленої мети були сформовані наступні завдання:

1. Дослідження популярних типів кіберзагроз та моделі мережі
2. Дослідження методів ІАД
3. Дослідження метрик для визначення ефективності алгоритмів
4. Пошук реальних наборів даних для дослідження
5. Аналізу та підготовки даних до навчання моделей
6. Моделювання та аналізу результатів

Об'єкт дослідження: застосування методів інтелектуального аналізу даних у кібербезпеці

Предмет дослідження: Алгоритми кластеризації та їх використання для виявлення кіберзагроз

РОЗДІЛ 1.

ТЕОРЕТИЧНІ ОСНОВИ КІБЕРБЕЗПЕКИ ТА ТЕХНОЛОГІЇ «ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ»

1.1 Важливість кібербезпеки у сучасному світі

1.1.1 Загальний опис кіберзагроз...

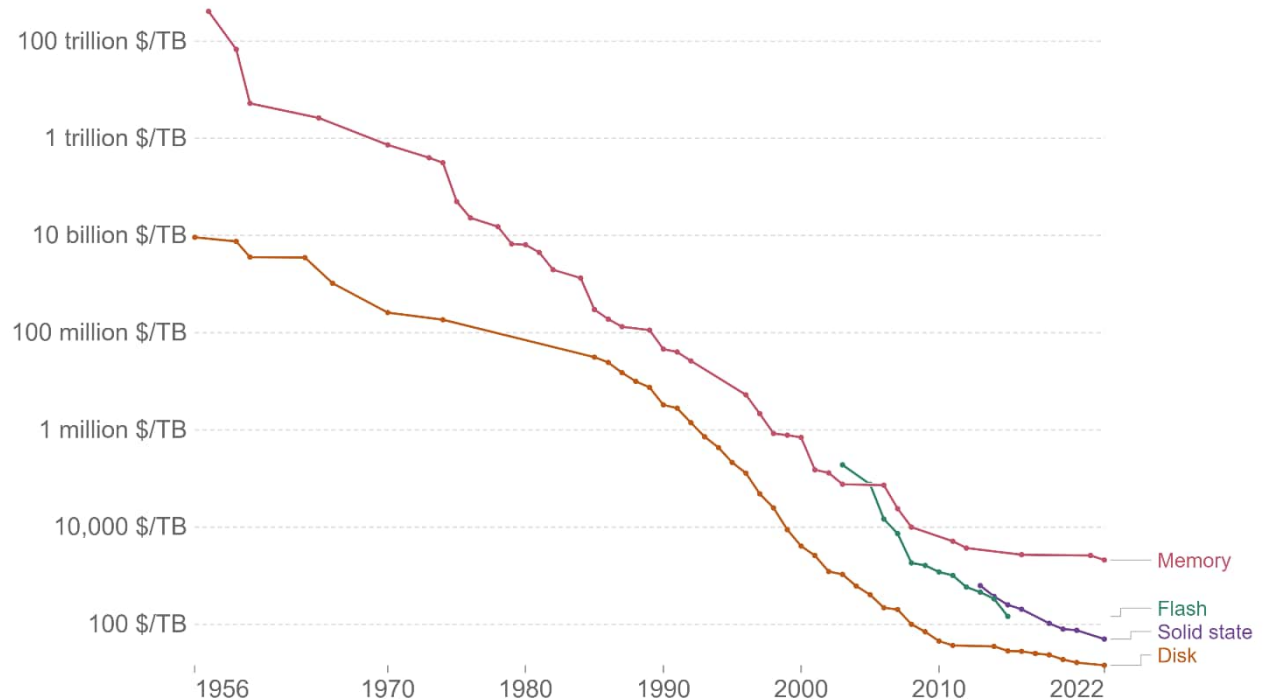
Людський розум завжди прагне вдосконалюватись та переборювати різноманітні виклики. Бажання шукати відповіді змусило людей працювати з інформацією та даними, яких неможливо обробити за короткий час без особливої техніки. Потреба породжує попит, тому у 1941 році Конрад Цузе створив перший електронний обчислювальний пристрій «Z3», який, після низки покращень за відносно малий проміжок часу у 70 років, стане одним з найвидатніших винаходів сьогодення – персональним комп'ютером.

Центральний процесор – це складова ПК, що виконує побітове обчислення. Примітивні математичні операції з нулями та одиницями є, безумовно, ефективними, проте, водночас, складними для інтерпретації та розуміння людиною. Для того, щоб досягнути ефективної роботи ПК і комп'ютерних мереж та зручності користування, було винайдено безліч периферій, протоколів, драйверів, програм та багато інших сутностей. Ці технології дозволили зробити користування ПК інтуїтивним. Відтепер, починаючи з покоління «Зет» та «Альфа», користуватись персональними комп'ютерами, чи подібними пристроями можуть навіть діти віком від 3-ох років.

Враховуючи потреби людей та можливості сучасних ПК, обчислення інформації – це далеко не єдина функція комп'ютера. Окрім неї є ще зберігання та передача інформації. Сучасні носії вже можуть зберігати терабайти даних, будучи доступними для побутового користувача. Для порівняння, ціна 1 терабайта даних для накопичувачів на магнітних дисках у 2023 році становить близько 15 доларів США, коли ціна такого ж об'єму даних у 1985 році становила більше 30 мільйонів доларів США (рисунок 1.1.1).

Historical cost of computer memory and storage

This data is expressed in US dollars per terabyte (TB). It is not adjusted for inflation.



Source: John C. McCallum (2022)

OurWorldInData.org/technological-change • CC BY

Note: For each year, the time series shows the cheapest historical price recorded until that year.

Рисунок 1.1.1: Тенденція цін на терабайт даних

Крім зберігання інформації, комп'ютери також можуть ефективно передавати її іншим комп'ютерам. Для цього необхідно об'єднати комп'ютери, які бажають «спілкуватись», в одну мережу. Комп'ютерні мережі класифікують на: PAN (Personal Area Networks) – персональні мережі, LAN (Local Area Networks) – локальні мережі, CAN (Campus Area Network) – кампусні мережі та WAN (Wide Area Networks) - глобальні мережі. Відрізняються ці мережі, в основному, об'ємом та розміром. Наприклад, локальні мережі можуть налічували лише десятків пристроїв, коли глобальна мережа «Інтернет» налічує вже понад 22 мільярда пристроїв по всьому світу.

Так само, як і в матеріальному світі, цифрова інформація має свою цінність. Кожен байт даних має своє значення, тому інформаційна безпека у кіберпросторі є дуже важливою. Починаючи від архівних сімейних фото та діалогів у месенджерах та закінчуючи базами даних багатомільярдних компаній, ПК та комп'ютерні мережі є пріоритетною ціллю для кіберзлочинців. Протидія

подібним загрозам та досягнення інформаційної безпеки – це головна мета комп'ютерної безпеки (кібербезпеки).

1.1.2 Наявні етапи протидії кіберзагрозам

Враховуючи складність та багаторівневність кіберпростору, існує безліч способів та варіацій кібератак. Злочинці використовують усі доступні їм інструменти: від соціальної інженерії до помилок в реалізації систем безпеки. Не зважаючи на постійний розвиток КБ, нові рішення можуть, як і розв'язати якусь проблему, так і породити нові вразливості. Оскільки необхідно зберігати три головні аспекти ІБ – конфіденційність, цілісність та доступність, проблема КБ є ще більш складною, ніж здається на перший погляд.

Життєвий цикл протидії кіберзагрозам можна поділити на 5 етапів:

- Ідентифікація. Цей етап необхідний, щоб ідентифікувати вразливості, необхідність захисту та особливості системи. Цей крок є клопітким, проте, дуже важливим, адже, навіть, незначні аспекти можуть вплинути на безпеку цілої мережі.
- Захист. Коли аналіз мережі проведено, необхідно усунути всі вразливості на соціальному та технічному рівнях. Для цього використовують криптографію, хешування, систему контролю доступу, системи протидії вторгнень (Intrusion Detection/Prevention System (IDS/IPS)) та багато інших інструментів. Також, користувачі системи повинні діяти за спеціальним протоколом, який мінімізує людський фактор у питанні безпеки.
- Виявлення. Після впровадження системи безпеки у попередньому кроці, необхідно проводити постійний моніторинг моделі захисту. Навіть, якщо попередні етапи були виконані безпомилково, існує небезпека виникнення нових загроз. На цьому етапі, модель захисту вдосконалюється та підтримується в активному стані.
- Відповідь. Якщо система виявилась скомпрометована, необхідно відразу виконати відповідні дії. Необхідно виявити вразливість, зупинити її експансію і згодом повністю її усунути.

- Відновлення. Після усунення порушення, систему потрібно відновити. Необхідно запровадити імунітет проти подібних загроз та наладити її роботу у штатному режимі.

Цей підхід до КБ дозволить спеціалістам побудувати найбільш ефективну та універсальну модель безпеки.

1.2 Загальний опис інтелектуального аналізу даних

1.2.1 Застосування ІАД у кібербезпеці

Враховуючи особливість постійного розвитку кіберпростору, з'являється все більше інструментів, корисних для боротьби з кіберзагрозами. Одним з найбільш перспективних напрямків є штучний інтелект. Ця сфера налічує десятки підгруп, починаючи від робототехніки та закінчуючи алгоритмами планування. Однією з галузей штучного інтелекту, яка б була дуже корисною для кібербезпеки, є інтелектуальний аналіз даних.

Алгоритми ІАД аналізують великі об'єми даних, щоб відшукати закономірності та шаблони зв'язку між ними. Використовуючи алгоритми машинного навчання, статистики на глибинного навчання, ІАД дозволяє встановлювати «неявні» закономірності, тобто ті, яких не видно під час експертного аналізу людиною. Однією з найбільших переваг ІАД є універсальність: вхідними даними можуть бути, як і звичайні тексти чи зображення так і логфайли чи клаптики коду. Така універсальність дозволяє ефективно використовувати алгоритми ІАД у кожному з п'яти етапів життєвого циклу протидії кіберзагрозам.

1.2.2 Визначення ІАД та процес впровадження

Інтелектуальний аналіз даних (Data mining) – це процес сортування та аналізу великих об'єктів даних з метою встановлення неявних закономірностей та зв'язків. ІАД є ключовою методикою у науці про дані та видобуванні знань. Як вже згадувалось раніше, ця методика пошуку закономірностей є дуже корисна

для виявлення, наприклад, зловмисних застосунків, аналізуючи дані результату їхніх дій.

Процес ІАД можна розділити на 4 кроки:

1. Збирання даних. У цьому етапі відбувається організація вибірки даних, які потім будуть оброблятися алгоритмом. Ці дані можуть бути структурованими чи неструктурованими. Втім, головні цілі цього кроку – це не якість даних, а розмір та актуальність вибірки. Розмір необхідний для більш точного аналізу, а правильне розуміння бізнес-проблеми, яку повинен вирішувати ІАД, забезпечує актуальність вибірки.
2. Підготовка даних. Після накопичення даних їх необхідно обробити. Погана база даних може не лише зменшити ефективність ІАД, а і взагалі викривити результати. Щоб цього уникнути, спочатку необхідно очистити базу даних: видалити пусті значення, дублікати та інший шум. Далі можна виділити процес концентрації – видалення входжень, які не є важливими для вибраної бізнес-проблеми. Це дозволить оптимізувати роботу алгоритму. Окрім згаданих процесів є ще й інші, притаманні різноманітним типам даних. Також, варто зазначити, що ІАД може використовуватись для підготовки даних до ІАД. Якщо бізнес-проблема вимагає створення великої кількості моделей, то правильним рішенням може бути створення моделі, яка буде допомагати проводити підготовку баз даних, орієнтуючись на відомі приклади.
3. Створення моделі. Коли дані готові, спеціаліст обирає необхідний алгоритм ІАД та застосовує його, щоб створити навчену модель, яка буде розв'язувати поставлену проблему. Існує чимало алгоритмів ІАД, кожен з яких виконує певне завдання краще за інших. Вибір найбільш ефективного може залежати від природи вибірки чи визначатись експериментальним шляхом.
4. Аналіз та інтерпретація моделі. Після створення, модель тестується на новій вибірці даних, яка відповідає бізнес-проблемі. Готову модель тестують в реальних умовах. Результати цього процесу зберігаються, а також, аналізуються задля покращення роботи моделі. Якщо, після аналізу,

модель відповідає усім вимогам, її вже можна використовувати для нових бізнес-стратегій.

Цей процес дозволяє максимально ефективно створити модель, готову до використання. Поширеним застосуванням моделі ІАД є навчання асоціативних правил, яке, наприклад, використовується для визначення рекомендацій товарів в онлайн магазинах, аналізуючи дані поведінки покупця.

1.2.3 Основні шаблони застосування ІАД

Враховуючи різноманіття алгоритмів ІАД та їх універсальність, закономірним є використання цих алгоритмів у різних сферах. ІАД використовують для покращення користувацького досвіду в додатках, створення самохідних автомобілів, чат-ботів, IoT систем та багато іншого, включаючи кібербезпеку. Втім, природу усіх цих алгоритмів можна розділити на 7 основних класів (технік):

- Навчання асоціативних правил. Завданням цих моделей є пошук закономірностей серед елементів бази даних. На основі цих регулярностей, алгоритм будує «асоціативні правила», які відповідають знайденим закономірностям. Тоді критерії «підтримки» та «впевненості» визначають актуальність певного асоціативного правила. Підтримка аналізує, як часто у вибірці зустрічаються входження, що відповідають правилу, тобто наскільки воно є актуальним, а впевненість підставляє знайдене входження у правило, звіряючи дані з очікуваним результатом. Прикладами використання цієї техніки є аналіз корзини покупця та система рекомендацій.
- Класифікація. Подібно до алгоритму навчання асоціативних правил, алгоритм класифікації шукає закономірності у базі даних. Тоді алгоритм визначає природу елементу та приписує йому ярлик класу. Система ярликів передається алгоритму у вигляді вхідних даних для тренування з уже готовими правильними входженнями. Прикладами роботи таких

алгоритмів є системи ідентифікації спаму, моделі передбачення відтоку клієнтів та системи розпізнавання шкідливого ПЗ.

- Кластеризація. Цей алгоритм також шукає закономірності в наборі даних та намагається розділити їх на групи. Різниця між цим алгоритмом та алгоритмом класифікації є відсутність вхідних даних з вже готовими класами. Натомість, алгоритм сам намагається зрозуміти природу елементів та розділяє дані на групи зі схожим описом. Прикладами є також ідентифікація спаму та сегментація зображень.
- Регресія. Головною метою алгоритму регресії є передбачення. Алгоритм приймає завдання – яку колонку даних необхідно вгадувати. Тоді аналізуються усі інші колонки з метою виробити правила походження. Натренована модель може приймати нові дані, аналізувати їх та передбачати бажане значення. Прикладів застосування алгоритмів передбачення є безліч: починаючи від моделювання середовища, закінчуючи прототипуванням фізико-хімічних властивостей.
- Нейронні мережі. Нейронні мережі є об'єктом дослідження наук машинного навчання та глибокого навчання. Ці алгоритми є подібними до ІАД, однак, мають зовсім іншу структуру. Ця структура подібна до людського мозку. Набір елементів (нейронів) аналізує вхідні дані та формує своє бачення, навчаючись, як людина. Головною перевагою таких алгоритмів є адаптивність. Нейронна мережа може аналізувати навіть необроблені дані чи адаптуватись до зовсім іншого формату даних. Такі системи використовуються для розпізнавання зображень, розпізнавання мовлення чи симуляції людської поведінки у робототехніці.
- Інтелектуальний аналіз тексту. Головна мета цього алгоритму – перетворити необроблені (сирі) дані текстів у структурований результат. Такі алгоритми можуть аналізувати великі об'єму неструктурованого тексту та знаходити закономірності, які є корисними власнику моделі. Поширеними прикладами застосування є аналіз відгуків, аналіз соціальних мереж та класифікація тексту.

- Пошук аномалій. Алгоритм пошуку аномалій є дуже корисним для кібербезпеки, адже дозволяє ідентифікувати входження, які є нетиповими у базі даних. Такі алгоритми використовуються для виявлення шахрайства чи у системах виявлення вторгнень (IDS), аналізуючи зміни у мережевому трафіку.

Окрім згаданих класів є ще також алгоритми аналізу часового ряду (Time Series Analysis), колаборативної фільтрації (Collaborative Filtering), зниження розмірності (Dimensionality reduction) та інші. Виділені 7 класів є найбільш поширеними та корисними з точки зору КБ.

1.2.4 Проблеми та недоліки ІАД

Алгоритми ІАД пропонують велику вибірку можливостей та переваг, дозволяючи систематично визначати природу даних чи передбачати певні значення на основі інших входжень. Такі функції дозволяють автоматизувати речі зі змінною природою та емулювати людську поведінку. На відміну від очевидних переваг, алгоритми ІАД також включають в себе значні виклики і протидії, які варто враховувати, а саме:

- Великі дані. Найбільш очевидною проблемою алгоритмів ІАД є потреба у великій кількості даних. Збір таких даних може бути клопітким, чи навіть незаконним, якщо мова йде про збір даних поведінки користувачів. Крім того, модель може потребувати структуризації цих даних, що також значно ускладнює роботу. Також, варто згадати, що процес навчання, якщо мова йде про бази даних з мільйонами входжень, може витратити багато енергії та часу.
- Точність. Алгоритми ІАД не завжди можуть повертати очікувану точність. Різні алгоритми можуть генерувати різні моделі, і пошук найбільш ефективного може забрати багато зусиль та часу. Проте, навіть, точність моделі понад 95% не дозволяє повністю їй довіряти. Використання таких алгоритмів, як вже згадувалось раніше, потребує постійного нагляду та аналізу.

- Складність та ціна. Сама по собі сутність алгоритмів ІАД не є простою. Їх використання можливе лише кваліфікованими спеціалістами, що зазвичай має свою вагому ціну для компанії, без гарантії отримання задовільних результатів.

Створення ефективних моделей ІАД займає чимало часу та зусиль. Такий крок повин бути обґрунтованим та необхідним для замовника. Однак, перелічені недоліки не зменшують значення та важливості цих алгоритмів, які вносять незамінний вклад у проект та мають безліч переваг.

1.3 Опис існуючих кіберзагроз та джерел їх виникнення

1.3.1 Принципи роботи комп'ютерних мереж. Моделі OSI та TCP/IP

Для того, щоб використати алгоритми ІАД у кібербезпеці, необхідно зрозуміти, які типи загроз існують, та на яких рівнях комп'ютерної мережі вони діють. Серед мережевих стандартів можна виділити модель OSI (Open System Interconnection) та модель TCP/IP (Transmission Control Protocol / Internet Protocol). Структурно та концептуально ці дві моделі є подібними (рисунок 1.3.1.1).

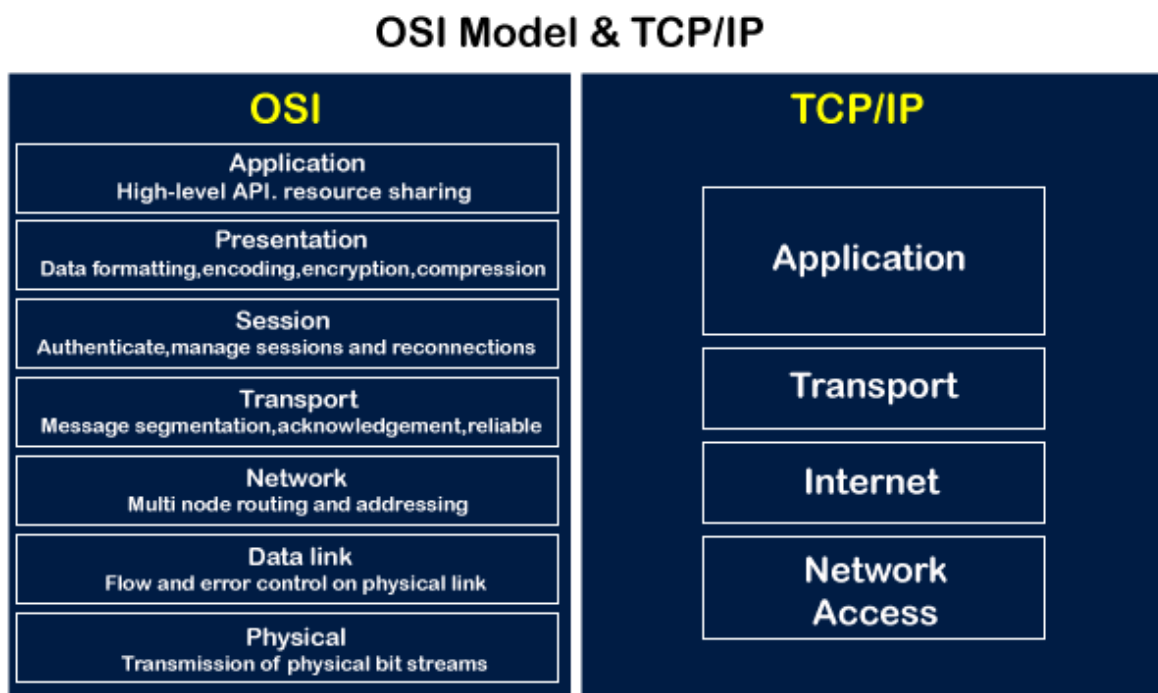


Рисунок 1.3.1.1: Моделі OSI та TCP/IP

Зараз використовується лише модель TCP/IP. Історично вибір розробників впав на модель TCP/IP, хоча ці дві моделі були дуже схожими, TCP/IP виявилась більш зручною для імплементації. Натомість більша кількість рівнів у моделі OSI є більш зручною для опису роботи мережі. Отже, в архітектурі комп'ютерних мереж можна виділити 7 рівнів (шарів):

- Фізичний рівень (Physical layer). Цей рівень є найнижчим в структурі. На цьому рівні відбувається фізичне транспортування сигналу між комп'ютерами, що контактують. Сигнал на цьому рівні є примітивним – лише нулі та одиниці, тому цей рівень вважається найнижчим. Цей рівень відповідальний за фізичне з'єднання, наприклад, оптоволоконні кабелі, конектори та хаби.
- Канальний рівень (Data Link layer). Цей рівень поєднує фізичний та мережевий рівні. Це шар відповідальний за цілісність та відсутність помилок сигналу, який передається на фізичному рівні. Після оброблення, периферія канального шару, а саме комутатори, мости та мережеві адаптери, передає сигнал на мережевий рівень.
- Мережевий рівень (Network layer). Цей рівень відповідальний за функції оброблення пакетів та навігації. Робота з пакетами означає розбивання даних на пакети при надсиланні та обернене збирання їх при отриманні. Іншим завданням є встановлення найкращого шляху на фізичному рівні. Для цього використовується протокол IP (Internet Protocol), щоб максимально ефективно надіслати пакети отримувачу. Усі ці функції виконує маршрутизатор.
- Транспортний рівень (Transport layer). Цей рівень за своїм принципом схожий до канального рівня. Він служить мостом між мережевим та сеансовим рівнями. Його завдання слідкувати за помилками та поділяти надто великі пакети на менші, враховуючи можливості мережі. Якщо дані з сеансового рівня містять помилки, тоді транспортний рівень вимагає повторного надсилання.

- Сеансовий рівень (Session layer). Цей рівень відповідальний за створення каналів комунікації (сесій) між комп'ютерами. Він дозволяє керувати цими сесіями, які необхідні для досягнення синхронної комунікації.
- Рівень представлення (Presentation layer). Завданням цього рівня є підготовка пакетів, отриманих на рівні сеансу, до прикладного рівня. Так само діє на оберненому шляху – оброблює будь-які дані з прикладного рівня для рівня сеансу. Підготовка та оброблення означає кодування/декодування, компресія та конвертація форматів даних.
- Прикладний рівень (Application layer). На цьому рівні відбувається представлення отриманих пакетів. Зазвичай це відбувається у веб-браузері чи у клієнті електронної пошти. Цей рівень є найвищим, отже, користувач сам взаємодіє з даними через аплікації.

Для кожного з цих рівнів було створено чимало протоколів у реалізації TCP/IP. У кожного протоколу своя функція - захисту, форматування, надсилання, навігації чи інша. Перелік протоколів та рівні моделі OSI, на який вони функціонують, зображено на рисунку 1.3.1.2.

Layer	Name	Data Type	Protocols (Examples)
7	Application Layer	PDU	HTTP, SMTP, POP3, DNS, DHCP, Telnet, FTP, SNMP, BGP, RIP, LDP HSRP
6	Presentation Layer	PDU	Data Formats : Text(ASCII, EBCDIC), JPEG, PNG, MPEG, MIDI, TIFF, AVI, MP4
5	Session Layer	PDU	APIs (Common APIs include : TCP/IP Sockets, RPC, NETBIOS), NFS, SQL, SIP
4	Transport Layer	Segment/Datagram	TCP, UDP, OSPF, EIGRP, ICMP, ESP, AH, VRRP, GRE, RSVP, IGMP, PIM
3	Network Layer	Packet	IPv4, IPv6, IPX, Apple Talk, ARP
2	Data Link Layer	Frame	MAC, LLC, VTP, 802.1Q, Frame Relay, PPP, HDLC, CDP, STP, DTP etc.
1	Physical Layer	Bits/Bytes	T1/E1, SONET, SDH, USB, OTN, Ethernet 10BASE-T, 100 Base-T etc.

Рисунок 1.3.1.2: Протоколи мережі та рівні моделі OSI

1.3.2 Класифікація кіберзагроз, та на яких рівнях моделі OSI вони діють

Враховуючи складність архітектури ПК та комп'ютерних мереж існує багато потенційних вразливостей, якими можуть скористатись зловмисники. З теоретичної точки зору існує достатньо інструментів, щоб захиститись від небезпеки. Однак, абстракцій загроз настільки багато, що просто неможливо врахувати їх усіх. Відомі випадки, коли навіть досвідчені професіонали з питань кібербезпеки припускались помилок, результатом який були багатомільярдні збитки для компаній. Серед усього різноманіття небезпек можна виділити 7 найбільш ефективні та поширені категорії:

1. Шкідливе ПЗ (Malware). Головною метою такого ПЗ є проникнення на пристрій жертви (зазвичай через випадкове завантаження в мережі «Інтернет») з ціллю виконання шкідливих дій. Ці програми можуть збирати приватну інформацію з машини жертви, намагатись отримати доступ до акаунту користувача, використовувати ресурси ПК у своїх цілях чи безліч інших шкідливих дій. Серед типів таких загроз можна виділити:

- Віруси (Viruses) – клаптики коду, вбудовані у інше ПЗ. Коли програма виконується, віруси виконують дії, які є корисні зловмисникам. Також здатні повторювати себе задля розповсюдження.
- Комп'ютерний хробаки (Worms) – головною метою цього шкідливого ПЗ є отримання контролю над операційною системою ПК жертви, результатом чого є атака на відмову в обслуговуванні (DoS/DDoS). Також таке ПЗ дуже швидко та легко розповсюджується, як віруси.
- Трояни (Trojans) – шкідливе ПЗ, яке часто замасковане під звичайну програму. Його ціль – отримати доступ до ПК жертви. Навідміну від вірусів та хробаків, трояни не здатні поширюватись самостійно.
- Програма-вимагач (Ransomware). На відміну від троянів, це ПЗ не намагається отримати доступ до ПК, натомість, його ціль – блокування пристрою. Коли така програма потрапляє на ПК та успішно виконується, користувач втрачає доступ до девайсу. Це ПЗ шантажує жертву, вимагаючи кошти, обіцяючи за них відновити доступ до пристрою.

- Шпигунське ПЗ (Spyware). Головна ціль таких засобів – отримання доступу до ПК жертви без їхнього відому та проведення відслідковування усій його дій та файлів. Такі програми можуть отримати доступ до платіжних даних, важливих документів та файлів, чи просто слідкувати за користувачем.
- Рекламне ПЗ (Adware). Головна ціль цього ПЗ – розповсюдження інформації. Також існують реалізації, що діють подібно до шпигунського ПЗ, відслідковуючи історію веб-браузера жертви, щоб просувати відповідну рекламу.

Якщо дивитись на модель OSI, то головним завданням шкідливого ПЗ є прикладний рівень (рівень 7). Також таке ПЗ може атакувати транспортний (рівень 4) та мережевий (рівень 3) рівні задля створення каналів комунікації між жертвою та зловмисником.

2. Соціальна інженерія. На відміну від інших небезпек, ця категорія націлюється на найбільшу вразливість ПК – його користувача. Людська необачність, халатність, недостатня обізнаність, жадність та інші недоліки є одними з найбільш поширених причин успішного проникнення зловмисників у технічно захищені мережі. Серед типів таких загроз можна виділити:

- Фішинг (Phishing). Зловмисники намагаються сконтактувати з жертвою через електронну пошту, маскуючись під лист від довіреного джерела (банку, мобільного оператора чи іншої компанії). Якщо такий лист відкрити він зазвичай вимагатиме введення платіжних чи інших даних.
- Вішинг (Voice Phishing) – те саме, що і фішинг, проте, зловмисники напряму контактують з жертвою через ПЗ зв'язку, чи телефон.
- Цькування (Baiting) - зловмисник заманює користувача у пастку, зазвичай, обіцяючи щось привабливе, наприклад, грошову винагороду.

Враховуючи особливість цієї категорії, неможливо оцінити на якому рівні моделі OSI вона діє. Можна лише виділи прикладний рівень (рівень 7), оскільки більшість таких взаємодій відбуваються через користувацький інтерфейс.

3. Атака на відмову в обслуговуванні ((Distributed) Denial-of-Service Attack (DoS/DDoS)). Таки атаки націлюються на обмежену пропускну здатність мережі та перенавантажують її великою кількістю трафіку. Під таким тиском система не може працювати у штатному режимі.

Головною ціллю такої атаки відносно моделі OSI є транспортний (рівень 4) та мережевий (рівень 3) рівні. Зловмисники перенавантажують мережу трафіком, що унеможлиблює її використання. Найбільш вразливими до DDoS атак є протоколи IP (Internet Protocol) та ICMP (Internet Control Message Protocol).

4. Включення коду (Injection Attacks). Ці атаки направлені на недоліки у кодї веб-додатків. Якщо сервер неправильно обробляє вхідні дані, то можливо створити запит, у якому буде прихований скрипт. Тоді сервер використовує дані з запиту, випадково запускаючи зловмисний код. Серед типів таких загроз можна виділити:

- SQL включення (SQL Injection). Зловмисник націлюється на незахищену базу даних. Вразливий сервер надсилає дані зловмисника до бази даних і виконує будь-які SQL команди, які були введені в запит. Більшість веб-додатків використовують бази даних на основі мови SQL (Structured Query Language), що робить їх вразливими до цієї атаки.
- Міжсайтовий скриптинг (Cross-Site Scripting (XSS)). Подібно до SQL включення, зловмисник вводить рядок тексту, що містить шкідливий JavaScript. Оскільки більшість сайтів використовують цю мову для реалізації логіки, такий код може виконатись всередині сервера, що приведе до жахливих результатів.
- XXE включення (XML External Entity (XXE) Injection). Атака здійснюється за допомогою спеціально створених документів XML. Це включення

відрізняється від інших атак, оскільки використовує вразливі місця в застарілих аналізаторах XML, а не необроблені запити користувача.

Включення коду націлені на прикладний рів (рівень 7), оскільки тут необхідна безпосередня взаємодія з аплікацією. Однак, успішне SQL включення може скомпрометувати на транспортний (рівень 4) та мережевий (рівень 3) рівні.

5. «Людина посередині» (Man-in-the-Middle). Під час цієї атаки зловмисник втручається у комунікаційний канал двох пристроїв, наприклад, користувача та сервера додатку. У такому випадку зловмисник може переглядати запити жертви та навіть викривляти їх. Серед типів таких загроз можна виділити:

- Мережеве прослуховування (Wi-Fi eavesdropping). Зловмисник створює бездротову точку доступу (Access Point) під виглядом реальної установи, наприклад кафе. Правильне її налаштування дозволить злочинцю переглядати усі запити, які роблять під'єднані користувачі.
- DNS спуфінг (DNS spoofing). Domain Name System – це система, відповідальна за доменні імена. Подібно до фішингу, зловмисник створює посилання, домен якого схожий на домен існуючої установи, копіює сторінку веб-додатку та пропонує користувачу увійти. Користувач вводить свій пароль, надаючи його зловмиснику.

Атаки подібного типу зазвичай компрометують канали зв'язку, тобто сеансовий рівень (рівень 5) та рівень представлення (рівень 6). На шостому рівні зловмисник змінює алгоритми шифрування, а на рівні сеансу можуть повністю впливати на утворені сесії комунікацій.

6. Supply Chain Attack. Ця атака є відносно новою. Головною ціллю зловмисників у цьому випадку є недостатньо захищені елементи додатку, такі як: протоколи, архітектура чи сам код. Особливість цієї атаки полягає у тому, що змінюється вхідний код програми, без відому її розробників. Такі додатки

продовжують розповсюджувати свої товари, не знаючи, що вони співпрацюють з зловмисниками.

Такі атаки зазвичай діють на рівнях 3 та 7 моделі OSI. На рівні мережі атаки «Supply chain» можуть компрометувати мережеву інфраструктуру, включаючи маршрутизатори, комутатори або сервери DNS. На прикладному рівні можуть використовувати вразливості в ПЗ.

1.3.3 Потенційне застосування ІАД для вирішення вищезгаданих загроз

Враховуючи універсальність алгоритмів ІАД, методи штучного інтелекту можуть використовуватись для боротьби з усіма категоріями кіберзагроз. Вміння алгоритмів ІАД знаходити закономірності та приховані зв'язки, дозволяє ефективно розпізнавати загрози різних типів. Серед усіх категорій можна виділи шкідливе ПЗ, фішинг та DoS/DDoS атаки, адже проти цих категорій ІАД є найбільш ефективний.

Протидія шкідливому ПЗ. Алгоритми навчання асоціативних правил та пошуку аномалій дозволяють виявляти підозрілу активність таких програм, а алгоритми кластеризації та класифікації дозволяють визначати тип виявленої загрози. Аналізуючи тенденцію останніх років, алгоритми ІАД стають дедалі популярнішими у боротьбі з шкідливим ПЗ (рисунок 1.3.3).

3.1 Malware detection using Data Mining:

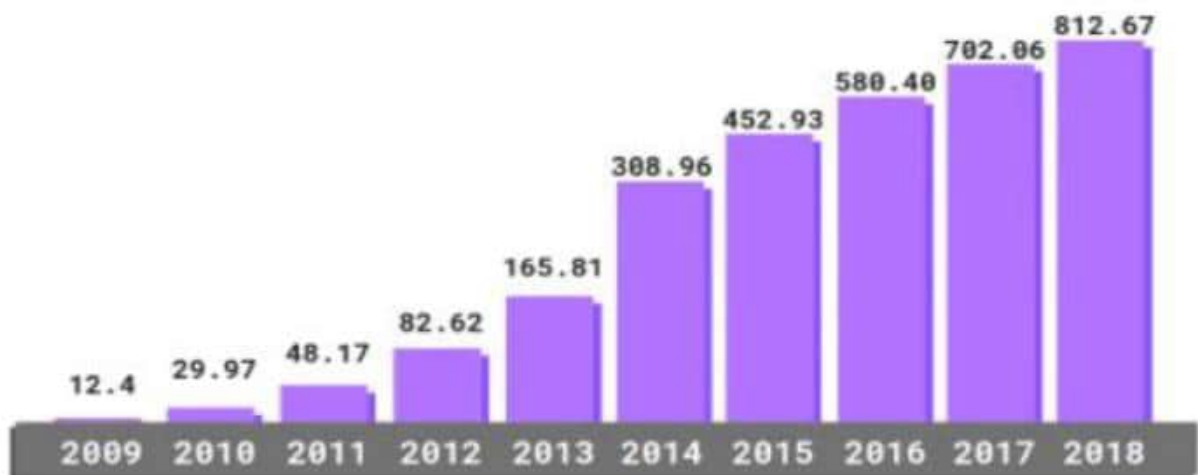


Рисунок 1.3.3: Тенденція використання алгоритмів ІАД для протидії

Протидія фішингу. Алгоритми кластеризації та класифікації дозволяють розширити функціонал анти-спам систем. Шкідливі електронні листи будуть розпізнаватись системою та автоматично видаляться. Також, алгоритми ІАД можуть зробити внесок у сферу аналізу небезпек, шукаючи нові класи та типи подібних загроз.

Протидія DoS/DDoS атакам. Головною вразливістю мереж проти таких атак є проблема визначення шкідливих запитів серед усього мережевого потоку. Такі запити надсилаються з усіх портів та перенавантажують систему. Алгоритми ІАД можуть відрізнити шкідливий трафік від звичайного та аналізувати незвичні патерни та аномалії, які потенційно вказують на втручання кіберзлочинців.

Висновки до розділу 1

Технології штучного інтелекту стрімко розвиваються, підтверджуючи свій потенціал. Алгоритми ІАД мають своє застосування у безлічі сфер, враховуючи сферу кібербезпеки. Тенденції збільшення кількості використання таких підходів підтверджують їх ефективність у протидії кіберзагрозам. Однак важливо не ігнорувати проблеми, пов'язані з кількістю та якістю даних, необхідних для ефективного використання ІАД, інтегруючи методи інтелектуального аналізу даних у стратегії кібербезпеки.

РОЗДІЛ 2.

ДЕТАЛЬНИЙ ОПИС ПРАКТИЧНОГО ЗАСТОСУВАННЯ ІАД У КІБЕРБЕЗПЕЦІ

2.1 Загальний опис алгоритмів ІАД

2.1.1 Алгоритми класу «дерево ухвалення рішень» (Decision Tree)

Група алгоритмів «decision tree» має багато застосувань, тому їх неправильно відносити до однієї з технік ІАД. Дерева ухвалення рішень можуть використовуватись для реалізації підходів кластеризації, регресії та пошуку аномалій.

Ці алгоритми мають ієрархічну структуру дерева, яка складається з кореневого вузла, гілок, внутрішніх вузлів і листових вузлів. Навчання дерева рішень використовує стратегію «розділяй і володарюй», проводячи жадібний пошук, щоб визначити оптимальні точки поділу всередині дерева. Потім цей процес поділу повторюється рекурсивним способом зверху вниз, доки всі або більшість записів не буде класифіковано за певними мітками класу. Чи всі точки даних класифікуються як однорідні набори, значною мірою залежить від складності дерева рішень. Меншим деревам легше отримати чисті вузли листя, тобто точки даних в одному класі. Однак у міру збільшення розміру дерева стає все важче підтримувати цю чистоту, і це зазвичай призводить до того, що в даному піддереві потрапляє занадто мало даних. Коли це відбувається, це називається фрагментацією даних, і це часто може призвести до помилки «перенавчання». Серед алгоритмів ухвалення рішень можна виділи два найбільш поширених:

- Алгоритм C4.5 (J48). C4.5 використовується для створення (навчання) класифікатора у формі дерева рішень із набору даних, які вже були класифіковані. Кожне входження має свої атрибути та параметри. Дерево рішень, створене алгоритмом C4.5, ставить питання про значення атрибута, і залежно від цих значень нові дані класифікуються. Навчальний набір даних позначається класами, що робить C4.5 алгоритмом «навчання з учителем». Дерево рішень завжди легко інтерпретувати та пояснити, що

робить C4.5 швидшим і більш популярним порівняно з іншими алгоритмами ІАД.

- Алгоритм класифікації та регресії дерев (CART). Цей алгоритм ухвалення дерева рішень повертає дерева регресії або класифікації на виході. У CART вузли дерева рішень мають рівно 2 гілки, як і у C4.5. Деревоподібна модель регресії або класифікації будується за допомогою підготовленого навчального набору даних, наданого користувачем. Тому його розглядають як методику навчання під наглядом (з учителем). Наприклад, результатом дерева регресії є безперервне або числове значення, як-от ціна певного товару або тривалість візиту туриста в готель.

2.1.2 Алгоритми класифікації

Алгоритм класифікації – це зазвичай методика навчання під наглядом (з учителем), яка використовується для визначення категорії нових входжень на основі даних навчання.

У класифікації програма вивчає підготовлений набір даних, а потім класифікує нові дані за класами або групами. Наприклад, «так» або «ні», 0 або 1, «спам» або «не спам», «велосипед» або «мотоцикл» тощо. На відміну від регресії, вихідні дані класифікації є категорією, а не значенням. Серед поширених алгоритмів класифікації можна виділити:

- Наївний класифікатор Баєса (Naïve Bayes classifier). Цей алгоритм оснований на теоремі Баєса. Існує багато варіацій цього алгоритму: гауссівський, поліноміальний, комплементарний, категоріальний та варіант Бернуллі. Припущення, яке використовує сімейство алгоритмів, полягає в тому, що кожен параметр даних, які класифікуються, не залежить від усіх інших параметрів, наданих у класі. Для побудови таблиць, алгоритму надається підготовлений (навчальний) набір даних. Тому його можна вважати алгоритмом навчання під наглядом.
- Метод k-найближчих сусідів (K-Nearest Neighbor (KNN)). KNN є одним з найпростіших алгоритмів машинного навчання методики навчання під наглядом. Також цей алгоритм називають «ледачим учнем» (якщо вхідні

дані визначаються однією множиною), оскільки він майже нічого не робить під час процесу навчання, окрім збереження вхідних даних. Ледачі алгоритми починають класифікувати лише тоді, коли на вхід надаються нові немарковані дані. C4.5, SVN і Adaboost, з іншого боку, є «охочими учнями», які починають будувати модель класифікації під час самого навчання.

- Логістична регресія (Logistic regression). Цей тип статистичної моделі часто використовується для класифікації та прогнозу аналітики. Логістична регресія оцінює ймовірність події на основі заданого набору даних незалежних змінних. Оскільки результат є ймовірністю, залежна змінна обмежена в інтервалі від 0 до 1. У логістичній регресії до шансів застосовується логіт-перетворення, тобто ймовірність успіху, поділена на ймовірність невдачі.
- Метод опорних векторів (Support vector machine (SVM)). SVM в основному використовуються для функції класифікації, регресії або ранжирування. Він формується на основі мінімізації структурного ризику та теорії статистичного навчання. Необхідно визначити межі рішення, які зветься гіперплощиною, які задаються рівнянням $y = MX + b$. Це допомагає в оптимальному розподілі класів. Основна робота SVM полягає у визначенні максимізації різниці (відступу) між двома типами. Ця різниця визначається, як кількість простору між двома типами. Так само, як багато інших алгоритмів, SVM є алгоритмом під наглядом.
- Випадковий ліс (Random Forest). Цей алгоритм поєднує результати кількох дерев рішень (Decision Tree) для досягнення єдиного результату. Його простота використання та гнучкість сприяли його поширенню, оскільки він спроможний вирішувати, як проблеми класифікації, так і регресії. Оскільки, дерева ухвалення рішень є зазвичай алгоритмами під наглядом, вони можуть бути схильні до проблем, таких як упередженість і перенавчання. Однак, коли кілька дерев гуртуються в алгоритмі випадкового лісу, вони передбачають точніші результати, особливо коли окремі дерева некорельовані одне з одним.

- Алгоритм AdaBoost. Цей алгоритм належить до групи алгоритмів градієнтного підсилювання. Основний принцип AdaBoost полягає в тому, щоб адаптувати результат «ледачих учнів» (тобто моделі, які лише трохи кращі за випадкові вгадування, такі як невеликі дерева рішень, KNN та Naïve Bayes з обмеженими вхідними даними) на неодноразово модифікованих версіях даних. Усі прогнози потім об'єднуються за допомогою зваженої більшості голосів (або суми) для отримання остаточного прогнозу.

2.1.3 Алгоритми регресії

Алгоритми регресії – це одна з технік ІАД, яка використовується для прогнозування безперервних числових значень на основі зв'язків між вхідними змінними та цільовою змінною (target column). Серед поширених алгоритмів регресії можна виділити:

- Лінійна регресія (Linear Regression). Лінійна регресія — це базовий і широко використовуваний алгоритм, який моделює лінійний зв'язок між вхідними характеристиками та цільовою змінною. Він оцінює коефіцієнти лінійного рівняння, які найкраще відповідають даним. Таку лінію можна задати рівнянням $y = X\beta + \epsilon$, де y – цільова змінна, X – дані, β – коефіцієнти, а ϵ – шум спостереження.
- Поліноміальна регресія (Polynomial regression). Поліноміальна регресія – це більш точна версія лінійної регресії. Якщо однієї площини недостатньо для задовільної точності усереднення, можна використати параболоїд, додавши до рівняння регресії поліноми другого порядку.
- Метод опорних векторів для регресії (Support Vector Regression (SVR)). SVR є розширенням SVM для завдань регресії. Він використовує опорні вектори для оцінки нелінійної функції, яка апроксимує дані, прагнучи мінімізувати помилку передбачення в межах допустимого відхилення.

Також, окрім згаданих алгоритмів для досягнення мети усереднення (регресії) можна використати наступні алгоритми: дерево рішень, випадковий ліс, нейронна мережа та регресія за Баєсом.

2.1.4 Алгоритми кластеризації

На відміну від більшості алгоритмів класифікації, методи кластеризації є алгоритмами без нагляду. Алгоритм кластеризації отримує на вхід багато непідготовлених даних без ярликів, в яких він шукає будь-які подібності, формуючи групи. Ці групи називаються кластерами. Кластер — це група точок даних, схожих одна на одну на основі їх відношення до оточуючих точок даних. Кластеризація використовується для таких речей, як розробка функцій або виявлення шаблонів. Серед поширених алгоритмів кластеризації можна виділити:

- Кластеризація методом *k*-середніх (K-Means Clustering). K-Means є найпоширенішим алгоритмом кластеризації. Цей алгоритм працює, створюючи *k* груп із набору об'єктів на основі подібності між ними. Немає гарантії, що учасники групи будуть абсолютно схожими, але вони будуть більш подібними, ніж учасники, які не є у групі. Відповідно до стандартних реалізацій, *k*-means є алгоритмом навчання без нагляду, оскільки він створює кластери самостійно без будь-якої зовнішньої інформації.
- Ієрархічна кластеризація (Hierarchical clustering). Ієрархічна кластеризація — це загальне сімейство алгоритмів кластеризації, які будують вкладені кластери шляхом їх послідовного злиття або розбиття. Ця ієрархія кластерів представлена у вигляді дерева (або дендрограми). Корінь дерева — це унікальний кластер, який збирає всі зразки, а листя — кластери лише з одним зразком. Є дві стратегії ієрархічної кластеризації: агломеративна (об'єднувальна) — це підхід знизу-вверх (спочатку кожна точка має власний кластер, а далі пари кластерів об'єднуються при підйомі по ієрархії) та розділювальна — це підхід зверху-вниз (спочатку усі точки в одному кластері, який потім рекурсивно розбивається).

- Алгоритм DBSCAN (Density-Based Spatial Clustering of Applications with Noise). Алгоритм DBSCAN розглядає кластери як області високої щільності, розділені областями низької щільності. Завдяки цьому досить загальному підходу, кластери, знайдені DBSCAN, можуть мати будь-яку форму (на відміну від k-means, який припускає, що кластери мають опуклу форму). Центральним компонентом DBSCAN є концепція «основних зразків», тобто зразків, що знаходяться в зонах високої щільності. Таким чином, кластер — це набір основних зразків, кожен з яких знаходиться близько один до одного (вимірюється певною мірою відстані), і набір неосновних зразків, які близькі до основного зразка (але самі по собі не є основними зразками).
- Зсув середнього (Mean Shift). Метою цього алгоритму є виявлення «бульбашок» у згладженій щільності вибірок. Цей алгоритм (заснований на центроїді) працює шляхом оновлення кандидатів на центроїди як середнє значення точок у певному регіоні. Потім ці кандидати фільтруються на етапі пост-обробки, щоб усунути схожі результати, щоб сформувати остаточний набір центроїдів. Положення центроїдів-кандидатів ітеративно коригується за допомогою техніки, що називається підйом на пагорб, яка знаходить локальні максимуми оціненої щільності ймовірності.

2.1.5 Алгоритми навчання асоціативних правил

Навчання асоціативних правил – це тип методу навчання без нагляду, який перевіряє залежність одного елемента даних від іншого і відповідно позначає їх для майбутнього застосування. Він намагається знайти якісь зв'язки чи асоціації між змінними набору даних. Серед поширених алгоритмів навчання асоціативних правил можна виділити:

- Алгоритм Apriori. Apriori працює на основі методу навчання правил асоціації. Коли ці зв'язки встановлені, вони застосовуються до бази даних, що містить велику кількість транзакцій. Цей алгоритм використовується для виявлення закономірностей і зв'язків і, отже,

розглядається як метод навчання без нагляду. Apriori є дуже ефективним, однак, він споживає багато пам'яті, використовує багато дискового простору та займає багато часу.

- Алгоритм ECLAT (Equivalence Class Transformation). Алгоритм ECLAT побудований на основі алгоритму Apriori. Цей алгоритм використовує метод пошуку в глибину для пошуку популярних наборів елементів у базі даних транзакцій. Він виконується швидше, ніж алгоритм Apriori.
- Алгоритм зростання частого зразка (F-P Growth). Алгоритм FP-Growth є альтернативним способом пошуку частих наборів елементів без використання поколінь кандидатів, таким чином покращуючи продуктивність. Для цього він використовує стратегію «розділяй і володарюй». Суть цього методу полягає у використанні спеціальної структури даних під назвою «дерево популярних шаблонів» (FP-tree), яка зберігає інформацію про асоціацію елемента. Дослідження творця алгоритму доводять, що FP-Growth є ефективнішим в плані продуктивності за Apriori та, навіть, ECLAT.

2.1.6 Алгоритми пошуку аномалій

Виявлення аномалій – це процес визначення точок даних, які виходять за межі нормальної поведінки. Наприклад, типові дані зазвичай відповідають певним формам, таким як нормальний розподіл. Переглядаючи набори даних і точки, які не підходять, зацікавлені сторони та команди обробки даних можуть визначити шаблони або точки, які можуть становити загрозу. Компанії також використовують інтелектуальний аналіз даних для виявлення шахрайства та виявлення мережових вторгнень. Серед поширених алгоритмів пошуку аномалій можна виділити:

- Robust covariance. Група алгоритмів Robust covariance базується на тому факті, що аномалії (нетипові входження) призводять до збільшення значень (записів) у Σ (коваріація значень), роблячи розкид даних значно більшим. Отже, $|\Sigma|$ (детермінант) також буде більшим, який теоретично

зменшиться за рахунок видалення нетипових входжень. У статистиці відхилення можна розрахувати за Z -оцінкою.

- **Однокласові SVM.** Однокласові SVM — це алгоритм без нагляду для виявлення аномалій. На відміну від звичайної SVM (що є алгоритмом під надглядом), однокласова SVM не має цільових міток для процесу навчання моделі. Натомість він вивчає межі для звичайних точок даних і ідентифікує дані за межами, як аномалії.

Окрім згаданих алгоритмів, для мети виявлення аномалій також використовують: DBSCAN, випадковий ліс та KNN.

2.2 Використання ІАД для виявлення шкідливого ПЗ

2.2.1 Підходи ІАД до аналізу шкідливого ПЗ

Процес пошуку шкідливого ПЗ називається аналізом. Існує два типи аналізу для виявлення шкідливих програм: статичний аналіз і динамічний аналіз.

Статичний аналіз – це аналіз ПЗ без виконання. Цей підхід досліджує код програми без запуску, з метою виявити шкідливе ПЗ і згодом, можливо, класифікувати його. Однією з проблем статичного аналізу є важкодоступність до початкового коду програми. Тому використання методів статичного аналізу зводиться до аналізу двійкових кодів, що, у свою чергу є дуже складним. При статичному методі двійковий код досліджується і шкідливе ПЗ виявляється саме на його основі. Варто також згадати, що отримання двійкових кодів є відносно клопітким завданням.

На відміну від статичного методу, який покладається на двійкові коди зловмисного ПЗ, динамічний підхід аналізує поведінку програми під час виконання. Цей процес залежить від значення змінних, вхідних даних програми та конфігурації системи. Динамічний аналіз використовується для виявлення нових типів шкідливих програм. Оскільки такий підхід виконує потенційно шкідливий код, задля безпеки передбачене використання віртуального середовища: симуляторів, емуляторів чи віртуальних машин.

Окрім статичного та динамічного типів аналізу, існує також гібридний варіант. Методи гібридного аналізу поєднують переваги статичного та динамічного аналізу. Упаковані шкідливі програми спочатку аналізуються за допомогою динамічного методу, а потім застосовується статична модель для виявлення прихованого коду ПЗ шляхом порівняння її роботи з результатом динамічного аналізу. Приховані файли виявляються за допомогою динамічного аналізатора, тоді як розпаковані файли перевіряються статичною моделлю.

2.2.2 Техніки виявлення шкідливого ПЗ з використанням ІАД

Техніки виявлення шкідливого програмного забезпечення можна узагальнено розподілити на дві категорії: виявлення на основі сигнатур (Signature-based) та виявлення на основі поведінки (Behavior-based). Обидві ці техніки можуть використовувати усі підходи до виявлення шкідливого ПЗ: статичний, динамічний та гібридний.

Виявлення на основі сигнатур. База даних сигнатур зберігає характеристики шкідливого програмного забезпечення попередніх атак. Коли вразливий код виявляється, він перевіряється шляхом вилучення унікальної послідовності байтів як сигнатури шкідливого ПЗ. Якщо вона співпадає з наявною сигнатурою, відповідний файл визнається шкідливим. Техніки видобутку даних, такі як класифікація та регресія, використовуються для категоризації загрози як шкідливого програмного забезпечення з використанням методу навчання під наглядом, що зберігає час і покращує точність передбачення, порівняно з традиційним методом. Цей метод легкий у використанні, надає вичерпну інформацію про шкідливе ПЗ, зручний для пошуку та широко використовується експертами. Однак, такий підхід не в змозі виявити поліморфне шкідливе програмне забезпечення, яке реплікує інформацію в великій базі даних.

Виявлення на основі поведінки. Поведінка програми, швидкість виконання, час реакції, звички перегляду, інформація про куки, типи вкладень, а також статистичні властивості допомагають виявити аномальну поведінку або шкідливий код. У методі виявлення на основі поведінки використовуються

зібрані особливості збірки та методи виклику API за допомогою алгоритмів ІАД. Для аналізу поведінки та виявлення прихованого шкідливого програмного забезпечення можуть бути використані методи без нагляду, такі як кластеризація, метод опорних векторів (SVM), алгоритми найближчих сусідів (KNN, ANN та інші). Цей метод допомагає виявляти поліморфні шкідливі програми, а також виявляти залежності потоку даних у зловмисному ПЗ. Проте, для виявлення складних візуальних шаблонів поведінки потрібно більше часу та обсягу пам'яті.

На рисунку 2.2.2 зображену структуру засобів ІАД для виявлення шкідливого ПЗ.

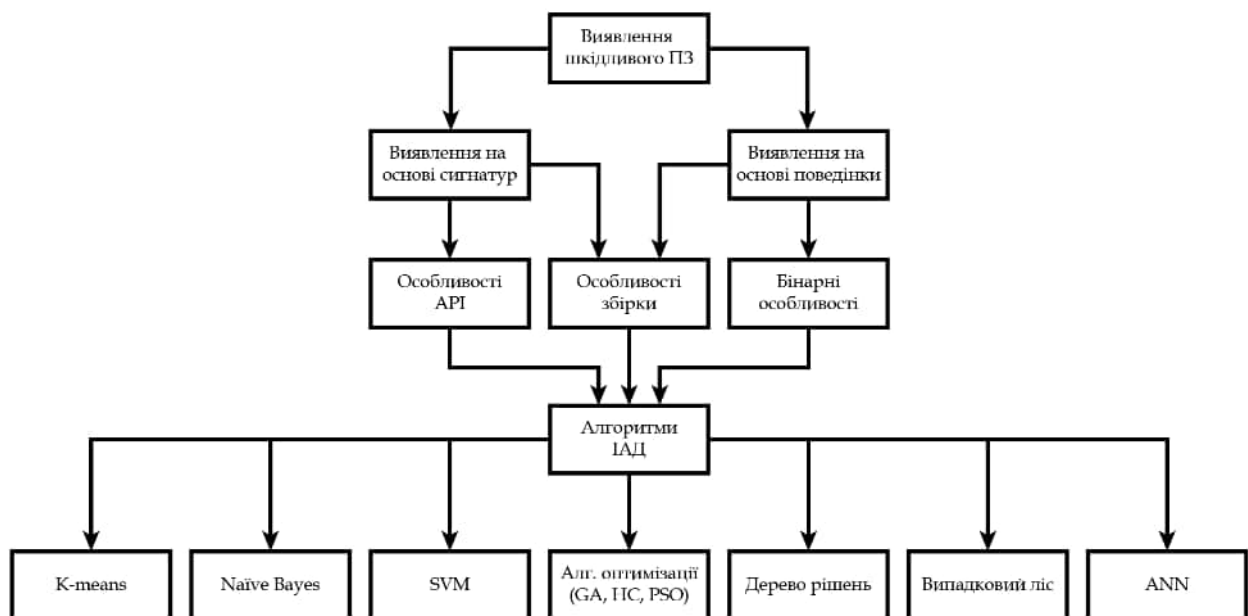


Рисунок 2.2.2: Структура засобів ІАД для виявлення шкідливого ПЗ

2.2.3 Використання алгоритмів ІАД відносно різних завдань та викликів

Згадані алгоритми, підходи та техніки ІАД мають свої застосування для різних проблем, пов'язаних зі шкідливим ПЗ. Нижченаведена таблиця (таблиця 2.2.3) показує різні техніки ІАД, що використовуються для виявлення шкідливого ПЗ згідно їх сигнатурних та поведінкових аспектів. Для вилучення прихованих шаблонів з даних використовуються статичні, динамічні та гібридні техніки аналізу даних з метою покращення точності виявлення шкідливого програмного забезпечення. Для експертів з кібербезпеки найскладнішим та найважливішим

викликом є вибір найкращого алгоритму та техніки ІАД для виявлення прихованих загроз та реалізації системи сповіщень, щоб захистити дані від подальших атак.

Тип шкідливого ПЗ	Алгоритм ІАД	Тип аналізу
Виявлення поліморфічного шкідливого ПЗ	K-means	Динамічний
Виявлення шкідливого ПЗ орієнтованого на Android	SVM, C4.5, Naïve Bayes	Динамічний
Виявлення шкідливого ПЗ через API	Naïve Bayes, SVM, дерево рішень, випадковий ліс	Динамічний
Виявлення шкідливого ПЗ на N-gram	SVM, ANN (усі найближчі сусіди)	Динамічний
Виявлення шкідливого ПЗ, орієнтованого на мобільні сервіси	Naïve Bayes, дерево рішень	Гібридний
Виявлення шкідливого ПЗ на основі послідовних шаблонів	ANN, KNN, SVM	Гібридний
Багатоцільове еволюційне виявлення шкідливого ПЗ	Генетичний алгоритм	Статичний
Виявлення шкідливого ПЗ забезпечення на основі частотних шаблонів	Інтелектуальний аналіз графів (GA)	Статичний
Виявлення шкідливого ПЗ на основі поведінки	Регресія, SVM, C4.5	Динамічний

Таблиця 2.2.3: Використання алгоритмів ІАД відносно різних завдань

Алгоритми ІАД, такі як класифікація, SVM, регресія, дерево прийняття рішень та алгоритми найближчих сусідів (KNN, ANN), можуть бути інтегровані з антивірусною системою для виявлення шкідливого програмного забезпечення до його проникнення в систему, що дозволяє захистити ІТ-інфраструктуру системи від подальших атак. Також, нейронні мережі, генетичні алгоритми, механізм глибокого навчання надають інтелектуальне виявлення шкідливого

програмного забезпечення за допомогою аналізу поведінки та сигнатурної бази даних.

2.3 Алгоритми ІАД для практичного застосування

2.3.1 Вступ до практичного використання алгоритмів ІАД. Опис пакету scikit-learn

Існує безліч можливих реалізацій алгоритмів ІАД. Велика частина великих компаній розробляє свої реалізації. Однак, якщо мова про звичайних користувачів, менші компанії чи компанії, які використовують вже створені реалізації, то є помітна тенденція. Найпопулярнішими мовами програмування для аналізу даних є Python (модулі scikit-learn, TensorFlow, PyTorch, pandas), R (модулі caret, randomForest, C50, arules) та іноді використовується Java (модулі Weka, Apache Mahout, ELKI).

2.3.2 Детальний опис найпопулярніших алгоритмів класифікації для практичного застосування

Далі буде наведено опис алгоритмів класифікації, опираючись на їх реалізацію у Python модулі scikit-learn. Ця бібліотека надає широкий спектр алгоритмів та інструментів для різних завдань видобутку даних, що робить її зручною для вирішення різноманітних проблем.

2.3.2.1 Алгоритм k-найближчих сусідів для класифікації (KNN)

Класифікація на основі найближчих сусідів є видом навчання на основі екземплярів або негенералізуючим навчанням: вона не намагається побудувати загальну внутрішню модель, а просто зберігає екземпляри навчальних даних. Класифікація визначається на основі простого вибору більшості найближчих сусідів кожної точки: цільовій точці присвоюється клас даних, який має найбільшу кількість представників серед найближчих сусідів точки.

Scikit-learn реалізує два різних класифікатори на основі найближчих сусідів: «KNeighborsClassifier» навчається на основі k-найближчих сусідів

кожної цільової точки, де k - це ціле число, задане користувачем. «RadiusNeighborsClassifier» навчається на основі кількості сусідів в межах фіксованого радіусу r кожної точки навчального набору, де r - це десяткове значення, задане користувачем.

Алгоритм KNN можна описати у шість кроків:

- 1) Вибір числа k сусідів
- 2) Розрахунок евклідової відстані (або будь-якої іншої) між k числом сусідів
- 3) Отримання k сусідів, згідно найменших значень відстані
- 4) Серед обраних k сусідів рахується кількість входжень для кожної категорії
- 5) Записати нові входження до категорії, щоб кількість сусідів була найбільшою
- 6) Готова для використання модель

Класифікація з використанням k -найближчих сусідів (k -neighbors) «KNeighborsClassifier» є найбільш поширеною технікою. Вибір оптимального значення k сильно залежить від даних: загалом, більше значення k пригнічує вплив шуму, але робить межі класифікації менш чіткими. Базова класифікація на основі найближчих сусідів використовує рівномірні ваги: значення, призначене цільовій точці, обчислюється на основі простого голосування більшості найближчих сусідів. В деяких випадках краще використовувати вагу сусідів таким чином, щоб ближчі сусіди вносили більший внесок у результат. Це можна зробити за допомогою параметра `weights`. Значення за замовчуванням, `weights = 'uniform'`, надає рівномірні ваги кожному сусіду. `weights = 'distance'` надає ваги, пропорційні оберненому значенню відстані від цільової точки. Також можна вказати власну функцію відстані для обчислення ваги.

2.3.2.2 Алгоритм дерева рішень

Дерева рішень (Decision Tree) є непараметричним методом навчання під наглядом, який використовується для класифікації та регресії. Метою є створення моделі, яка передбачає значення цільової змінної, вивчаючи прості

правила прийняття рішень, отриманих з ознак даних. Дерево можна розглядати, як апроксимацію неперервної функції шляхом кусково-заданої функції.

У таких алгоритмів є наступні переваги:

- Простота розуміння та інтерпретації. Дерева можна візуалізувати.
- Вимагає мало підготовки даних. Інші методи часто вимагають нормалізації даних, створення фіктивних змінних і видалення порожніх значень. Проте, слід зауважити, що цей модуль не підтримує обробку пропущених значень.
- Вартість використання дерева (тобто передбачення даних) логарифмічно залежить від кількості використаних точок даних для навчання дерева.
- Може обробляти як числові, так і категоріальні дані. Проте реалізація `scikit-learn` наразі не підтримує категоріальні змінні. Інші методи зазвичай спеціалізуються на аналізі наборів даних з одним типом змінних.
- Може вирішувати проблеми з багатьма вихідними змінними.
- Використовує модель "білої скриньки". Якщо дана ситуація спостерігається в моделі, пояснення для умови легко пояснюється булевою логікою. У порівнянні з цим, в моделі "чорної скриньки" (наприклад, у штучній нейронній мережі) результати можуть бути складнішими для інтерпретації.
- Можливість перевірки моделі за допомогою статистичних тестів. Це дозволяє враховувати надійність моделі.
- Добре працює навіть якщо їх припущення дещо порушені моделлю, з якої були згенеровані дані.

Серед недоліків можна виділити:

- Навчальні алгоритми рішучих дерев можуть створювати надмірно складні дерева, які погано узагальнюють дані. Це називається перенавчанням. Для уникнення цієї проблеми потрібні механізми, такі як обрізка дерева, встановлення мінімальної кількості зразків у листі або встановлення максимальної глибини дерева.

- Рішучі дерева можуть бути нестійкими, оскільки невеликі зміни в даних можуть призвести до повністю іншого дерева. Цю проблему можна зменшити за допомогою використання рішучих дерев в складі збірки.
- Прогнози рішучих дерев не є гладкими або безперервними, а є кусково-заданими наближеннями. Тому вони погано прогнозують поза межами відомих даних.
- Є концепції, які важко навчити, оскільки рішучі дерева не виражають їх легко, наприклад, проблеми XOR, паритету або мультиплексорів.
- Навчальні алгоритми рішучих дерев створюють незбалансовані дерева, якщо деякі класи домінують на іншими. Тому рекомендується перед навчанням рішучого дерева збалансувати набір даних.

Існує чимало реалізацій алгоритму дерева рішень: ID3, C4.5, C5.0 та CART. Проте, названі алгоритми є покращеними версіями попередників (алгоритми перераховані у хронологічному порядку). Бібліотека scikit-learn використовує оптимізовану версію алгоритму CART (Classification and Regression Trees); проте, наразі реалізація scikit-learn не підтримує категоріальні змінні.

Кроки алгоритму CART (Classification and Regression Trees):

- 1) Вибір кореневого вузла(i_v) (S) на основі коефіцієнту Джині та найбільшого приросту інформації.
- 2) На кожній ітерації алгоритм обчислює коефіцієнт Джині та приріст інформації, враховуючи, що кожен вузол є невикористаним.
- 3) Вибір вузла на основі найнижчого коефіцієнту Джині або найвищого приросту інформації.
- 4) Поділ множини S на підмножини даних.
- 5) Алгоритм продовжує рекурсивно працювати з кожною підмножиною, переконуючись, що атрибути є актуальними, і створює дерево рішень.

Приріст інформації базується на зменшенні ентропії після поділу набору даних за атрибутом. Побудова дерева рішень полягає у пошуку атрибуту, що має найбільший приріст інформації (тобто найбільш однорідні гілки).

$$\text{Приріст}(S, A) \equiv \text{Ентропія}(S) - \sum_{v \in D_A} \frac{|S_v|}{|S|} \text{Ентропія}(S_v)$$

Де $|S_v|$ дорівнює сумі усіх значень вузла, $|S|$ дорівнює сумі значень множини S , $\text{Ентропія}(S_v)$ дорівнює ентропії даної ноди

Коефіцієнт Джині обчислюється шляхом віднімання суми квадратів ймовірностей кожного класу від одиниці. Він сприяє створенню більших партій і є легким у реалізації, в той час як приріст інформації сприяє створенню менших партій з виразними значеннями.

$$\text{Джині} = 1 - \sum_{i=1}^c (p_i)^2$$

Цей коефіцієнт працює з категоріальною цільовою змінною «успіх» або «невдача». Він виконує лише двійкові розбиття.

2.3.2.3 Алгоритм випадковий ліс

Випадковий ліс (Random Forest) є метаоцінювачем, який побудований на основі декількох класифікаторів дерев рішень, які були застосовані до під-вибірок даних. Використовуючи усереднення, випадковий ліс покращує прогностичну точність і контролює перенавчання.

У випадковому лісі кожне дерево у збірці будується з вибірки, отриманої заміщенням (bootstrap sample) з тренувального набору. Крім того, при розбитті кожного вузла під час побудови дерева випадковим чином вибирається краще розбиття з усіх вхідних ознак або з випадкової підмножини розміром змінної *max_features*.

Мета цих двох джерел випадковості полягає у зменшенні варіації оцінювача лісу. Фактично, окремі дерева рішень зазвичай мають високу варіацію і схильні до перенавчання. Випадковість, внесена в ліс, призводить до того, що помилки прогнозування в окремих деревах є в певній мірі незалежними. Шляхом

усереднення цих прогнозів деякі помилки можуть бути знецінені. Випадкові ліси зменшують дисперсію, поєднуючи різноманітні дерева, іноді це відбувається за рахунок невеликого збільшення систематичної помилки. На практиці зменшення дисперсії часто є значним, тому це призводить до отримання загально вдосконаленої моделі. Реалізація scikit-learn комбінує класифікатори шляхом усереднення їх ймовірнісних прогнозів, а не шляхом голосування кожного класифікатора за один клас.

Для кожного дерева прийняття рішень реалізація цього алгоритму у scikit-learn обчислює важливість вузлів, використовуючи коефіцієнт важливості Джині, припускаючи наявність лише двох дочірніх вузлів (бінарне дерево):

$$ni_j = w_j C_j - w_{\text{лівий}(j)} C_{\text{лівий}(j)} - w_{\text{правий}(j)} C_{\text{правий}(j)}$$

Де ni_j – це важливість вузла j , w_j дорівнює зваженому числу вибірок, що досягають вузла j , C_j – це значення нечистоти вузла j , $\text{лівий}(j)$ – дочірній вузол зліва вузла j , $\text{правий}(j)$ – дочірній вузол права вузла j .

Тоді можна розрахувати важливість ознаки:

$$fi_i = \frac{\sum_{j:\text{вузол } j \text{ поділяється на ознаку } i} ni_j}{\sum_{k \in \text{всі вузли}} ni_j}$$

Де fi_i – це важливість ознаки i , а ni_j – важливість вузла J . Після цього можна нормалізувати ці значення важливості до інтервалу від 0 до 1.

Останній крок – це вираховування важливості цілого лісу:

$$RFfi_i = \frac{\sum_{j \in \text{всі дерева}} \text{нормалізоване } fi_{ij}}{T}$$

Де $RFfi_i$ – це важливість усього лісу, $\text{нормалізоване } fi_{ij}$ – це нормалізована важливість ознаки i у дереві j , а T – загальна кількість дерев.

2.3.2.4 Алгоритм логістичної регресії

Логістична регресія – це статистична модель, яка зазвичай використовується для моделювання бінарної залежної змінної з допомогою логістичної функції. Інша назва для логістичної функції - сигмоїдна функція, і вона задається таким чином:

$$F(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

Ця функція допомагає моделі логістичної регресії стиснути значення з інтервалу $(-k, k)$ до $(0, 1)$. Логістична регресія в основному використовується для бінарної класифікації, але її також можна використовувати для багатокласової класифікації.

Причина, чому модель класифікації зветься логістичною регресією, полягає в тому, що, подібно до лінійної регресії, логістична регресія починається з лінійного рівняння. Проте це рівняння складається з логарифмічних шансів, які подаються через сигмоїдну функцію, що стискає вихід лінійного рівняння до ймовірності між 0 та 1. Тоді можливо встановити поріг рішення та використовувати цю ймовірність для проведення завдання класифікації.

Можна почати з припущення, що $p(x)$ є лінійною функцією. Проте, проблема полягає в тому, що p - це ймовірність, яка повинна змінюватися від 0 до 1, тоді як $p(x)$ - це необмежене лінійне рівняння. Щоб вирішити цю проблему, слід вважати, що $\log p(x)$ є лінійною функцією x , і щоб обмежити її в діапазоні $(0, 1)$, ми використовуємо логіт-перетворення. Для цього використовується функція $\log p(x)/(1-p(x))$. Далі необхідно перетворити цю функцію на лінійну:

$$\log \frac{p(x)}{1 - p(x)} = \alpha_0 + \alpha x$$

Після перетворень відносно $p(x)$, це рівняння зводиться до:

$$p(x) = \frac{e^{\alpha_0 + \alpha x}}{1 + e^{\alpha_0 + \alpha x}}$$

Щоб зробити логістичну регресію лінійним класифікатором, можна вибрати певний поріг, наприклад, 0.5. Тепер, якщо $p \geq 0.5$, то у прогнозується, як 1, а якщо $p < 0.5$, тоді $y=0$. Тут 1 і 0 є класами. Оскільки логістична регресія прогнозує ймовірності, слід використовувати метод максимальної правдоподібності для її навчання. Тому для кожної навчальної точки даних x , передбачений клас - це y . Ймовірність y дорівнює p , якщо $y=1$, або $1-p$, якщо $y=0$. Тепер функцію правдоподібності можна записати таким чином:

$$L(\alpha_0, \alpha) = \prod_{i=1}^n p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$

Множення можна перетворити на суму через логарифмування. Тоді, після підставлення у рівняння $p(x)$, функція виглядає наступним чином:

$$l(\alpha_0, \alpha) = \sum_{i=0}^n -\log(1 + e^{\alpha_0 + \alpha x_i}) + \sum_{i=0}^n y_i (\alpha_0 + \alpha x_i)$$

Останній крок - це взяття максимуму з функції правдоподібності, оскільки в логістичній регресії використовується градієнтний підйом (протилежно до градієнтного спуску).

2.3.2.5 Алгоритм AdaBoost

Основний принцип AdaBoost полягає у побудові послідовності слабких моделей (тобто моделей, які незначно краще вгадують, ніж випадкове вгадування, наприклад) на повторно змінених версіях даних. Прогнози з усіх цих моделей потім комбінуються за допомогою зваженої більшості голосів (або суми), щоб отримати кінцевий прогноз. Зміни даних на кожній так званій ітерації підсилення полягають у застосуванні ваг $\omega_1, \omega_2, \dots, \omega_N$ до кожного з навчальних зразків. Спочатку всі ці ваги встановлюються до значення $\omega_i = 1/N$, так що перший крок просто навчає слабку модель на вхідних даних. На кожній наступній ітерації ваги зразків змінюються і навчальний алгоритм повторно застосовується до зважених

даних. На кожному кроці ваги навчальних прикладів, які були неправильно передбачені попередньою моделлю, збільшуються, тоді як ваги неправильно передбачених зменшуються. З кожною наступною ітерацією важко передбачувані приклади отримують все більший вплив. Кожна наступна слабка модель змушена концентруватися на прикладах, які були проігноровані попередніми в послідовності.

Якщо дано набір тренувальних даних з мітками $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ де x_i представляє вхідні ознаки, а y_i – відповідні мітки класу ($y_i \in \{-1, 1\}$). Алгоритм AdaBoost працює за такою схемою:

- 1) Ініціалізування ваги тренувальних прикладів: $\omega_i = 1/n$, де n - загальна кількість тренувальних прикладів.
- 2) Для $t =$ від 1 до T (T - загальна кількість ітерацій або слабких класифікаторів):
 - а) Навчити слабкий класифікатор (наприклад, дерево рішень) на тренувальних даних з вагами ω_i . Слабкий класифікатор намагається знайти найкраще розбиття за обраним критерієм (наприклад, коефіцієнтом Джині або приростом інформації).
 - б) Обчислити помилку слабого класифікатора: $E_t = \sum \omega_i$, якщо слабкий класифікатор неправильно класифікує приклад i , інакше $E_t = 0$.
 - в) Обчислити вагу слабого класифікатора: $\alpha_t = 0.5 * \ln((1 - E_t)/E_t)$. Альфа вимірює внесок слабого класифікатора в кінцевій збірці.
 - г) Оновити ваги тренувальних прикладів: Для правильно класифікованих прикладів: $\omega_i = \omega_i * e^{-E_t}$. Для помилково класифікованих прикладів: $\omega_i = \omega_i * e^{E_t}$.
 - е) Нормалізувати ваги так, щоб вони сумувались до 1: $\omega_i = \omega_i / \sum \omega_i$.
- 3) Отримання кінцевої збірки класифікаторів, комбінуючи слабкі класифікатори з використанням їх ваг. Передбачення вираховується формулою:

$$F_T(x) = \sum_{t=1}^T \alpha_t h_t(x)$$

Де $h_t(x)$ – це прогноз t -го слабкого класифікатора для вхідного x

2.3.2.6 Алгоритм методу опорних векторів (SVM)

Метод опорних векторів (SVM) - це набір методів навчання під наглядом, які використовуються для класифікації, регресії та виявлення відхилень. Переваги методу опорних векторів включають:

- Ефективність у наборах даних високої вимірності.
- Зберігання ефективності навіть у випадках, коли вимірність більша, ніж кількість вибірок.
- Використовує підмножину тренувальних точок у функції прийняття рішення (називаються опорними векторами), тому цей метод ефективний за споживанням пам'яті.
- Універсальність: можна вказати різні ядрові функції для досягнення прийняття рішення. Звичайні ядра вказуються за замовчування, але також можливе вказання власних ядер.

Недоліки методу опорних векторів включають:

- Якщо кількість ознак значно перевищує кількість вибірок, необхідно уникати перенавчання. Правильний вибір ядрових функцій та параметра регуляризації має найбільший вплив.
- Метод опорних векторів не надає прямої оцінки ймовірностей, їх необхідно обчислювати за допомогою складної п'ятикратної перехресної перевірки.

Для практичного використання класифікації використовується версія алгоритму SVM для класифікації – SVC (Support Vector Classification). З математичної точки зору головним завданням SVC є розв'язок головної проблеми:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^n \zeta_i$$

subject to $y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i,$
 $\zeta_i \geq 0, i = 1, \dots, n$

Де дано вектори для тренування $x_i \in \mathbb{R}^p, i = 1, \dots, n$, у двох класах, і вектор $y \in \{1, -1\}^n$. Ціль алгоритму – знайти $\omega \in \mathbb{R}^p$ та $b \in \mathbb{R}$ так, щоб передбачення, що задається $\text{sign}(\omega^T \phi(x) + b)$, справджувалось для більшості вибірок.

2.3.2.7 Алгоритм наївного класифікатора Баєса

Методи наївного Баєса є набором алгоритмів навчання під наглядом, які базуються на застосуванні теореми Баєса з "наївним" припущенням умовної незалежності між кожною парою ознак при заданому значенні змінної класу. Теорема Баєса визначає наступний зв'язок, при заданій змінній класу y та залежному вектору ознак від x_1 до x_n :

$$P(y|x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n|y)}{P(x_1, \dots, x_n)}$$

Використовуючи наївне припущення про умовну незалежність:

$$P(x_i|y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i|y)$$

Для усіх i , це відношення може бути спрощене до

$$P(y|x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^n P(x_i|y)}{P(x_1, \dots, x_n)}$$

Оскільки $P(x_1, \dots, x_n)$ – це константа зі вхідних даних, то можна використати наступне правило класифікації:

$$P(y|x_1, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i|y)$$

Яке можна перетворити у:

$$\hat{y} = \arg \max_y P(y) \prod_{i=1}^n P(x_i|y)$$

Незважаючи на їхні видимо надто спрощені припущення, наївні класифікатори Баєса добре працюють в багатьох реальних ситуаціях, зокрема у класифікації документів та фільтрації спаму. Також, для оцінки необхідних параметрів вони потребують невеликий обсяг тренувальних даних.

Наївні навчальні та класифікаторні алгоритми Баєса можуть бути надзвичайно швидкими порівняно з більш складними методами. Розрив між умовними розподілами ознак класу означає, що кожен розподіл може бути незалежно оцінений як одновимірний розподіл. Це, в свою чергу, допомагає уникнути проблем, пов'язаних з прокляттям вимірності.

Різні наївні класифікатори Баєса відрізняються головним чином припущеннями, які вони роблять щодо розподілу $P(x_i|y)$. Наприклад наївний класифікатор Баєса за Гаусом використовує наступну апроксимацію розподілу ознак:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Де параметри σ_y та μ_y оцінюються максимальною апроксимацією.

2.4 Метрики для оцінювання алгоритмів ІАД

2.4.1 Опис метрик порівняння та оцінки алгоритмів ІАД

Для того, щоб мати змогу оцінити ефективність кожного використаного алгоритму класифікації необхідно використовувати метрики. Найбільш

популярні метрики: точність (Accuracy), позитивна точність (Precision), повнота (Recall), F-міра (F-measure), матриця помилок (Confusion Matrix) та ROC (Receiver Operating Characteristic) крива.

2.4.2 Метрика точності (Accuracy)

Точність (Accuracy) є однією з метрик оцінки моделей класифікації. Неформально, точність представляє собою відношення правильних прогнозів, зроблених моделлю, яка оцінюється. Формально, точність має таке визначення:

$$Accuracy = \frac{\text{Кількість правильних передбачень}}{\text{Загальна кількість передбачень}}$$

Для бінарної класифікації, точність також можна розрахувати за допомогою поняття позитивних і негативних значень наступним чином:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Де TP – правильні позитивні (передбачення) (True Positives), TN – правильні негативні (True Negatives), FP – неправильні позитивні (False Positives), а FN – неправильні негативні (False Negatives).

Точність сама по собі не дає повної оцінки для набору даних з нерівномірним розподілом класів, наприклад, коли існує значна різниця між кількістю позитивних та негативних міток. Результатом є значення в інтервалі від 0 до 1.

2.4.3 Метрика позитивної точності (Precision)

Позитивна точність (Precision) оцінює частину правильності позитивних передбачень:

$$Precision = \frac{TP}{TP + FP}$$

Precision вказує на те, яка частка позитивних екземплярів, визначених моделлю, є дійсно позитивними. Висока точність означає, що модель мало помиляється (визначаючи негативні екземпляри як позитивні). Результатом є значення в інтервалі від 0 до 1.

2.4.4 Метрика повноти (Recall)

Метрика повноти (Recall) використовується для оцінки класифікаційних моделей, зокрема в задачах бінарної класифікації. Вона вимірює пропорцію правильно визначених позитивних екземплярів (True Positives) до загальної кількості позитивних екземплярів (True Positives + False Negatives).

Формально, повнота обчислюється за формулою:

$$Recall = \frac{TP}{TP + FN}$$

Повнота вказує на те, яка частка позитивних екземплярів була правильно визначена моделлю. Висока повнота означає, що модель рідко пропускає позитивні екземпляри, тобто вона вміє "впіймати" більшість позитивних екземплярів. Результатом є значення в інтервалі від 0 до 1.

2.4.5 Метрика F-міра (F-measure)

Метрика F-міра (F-measure/F-score) – це одна з мір точності моделі. Її обчислюють через точність (precision) та повноту (recall) тесту. Міра F_1 є середнім гармонійним точності та повноти. Загальніша міра F_β застосовує додаткові ваги, оцінюючи або влучність, або повноту вище за іншу. Результатом є значення в інтервалі від 0 до 1. Традиційно використовується збалансована міра F_1 , яка розраховується за формулою:

$$F_1 = \frac{2}{Recall^{-1} + Precision^{-1}} = \frac{2TP}{2TP + FN + FP}$$

Метрика F_β , в свою чергу, оцінюється так:

$$F_{\beta} = \frac{(1 + \beta^2)TP}{(1 + \beta^2)TP + \beta^2FN + FP}$$

Де β – це дійсний позитивний коефіцієнт. Значення β вибирається так, щоб повнота (Recall) вважалася β разів важливішою, ніж точність (Precision).

2.4.6 Матриця помилок (Confusion Matrix)

Матриця помилок (Confusion matrix), є також інструментом для оцінки ефективності моделей класифікації. Вона представляється у вигляді таблиці, де кожен ряд відповідає фактичному класу, а кожен стовпець - прогнозованому класу.

Матриця складається з 4 елементів: TP (правильні позитивні), FP (неправильні позитивні), TN (правильні негативні) та FN (неправильні негативні) (див. рис. 2.3.3.5).

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Рисунок 2.3.3.5: Модель матриці помилок

Матриця помилок дозволяє зробити детальний аналіз помилок, які зроблені моделлю, і зрозуміти їх природу.

2.4.7 ROC крива

ROC (Receiver Operating Characteristic) крива є графічним інструментом для візуалізації ефективності класифікатора на бінарній задачі класифікації. Вона відображає залежність між обраною метрикою True Positive Rate (TPR) (також

відома, як повнота або Recall) і False Positive Rate (FPR). FPR вираховується за формулою:

$$FPR = \frac{FP}{FP + TN}$$

ROC-крива побудована шляхом розрахунку TPR та FPR при різних порогових значеннях.

Аналіз ROC надає інструменти для вибору (ймовірно) оптимальних моделей та відкидання менш ефективних незалежно від (і до визначення) контексту витрат або розподілу класів. ROC-крива має діапазон від 0 до 1 і може бути зображена на графіку. Чим ближче крива до верхнього лівого кута графіка, тим краще ефективність класифікатора. Ідеальний класифікатор матиме ROC-криву, яка пройде через точку (0,1), що відповідає 100% TPR і 0% FPR. За допомогою ROC-кривої можна порівнювати ефективність різних класифікаторів і вибирати найкращий для конкретної задачі класифікації.

Висновки до розділу 2

У контексті кібербезпеки, алгоритми ІАД можуть бути використані для виявлення аномалій, ідентифікації вразливостей та прогнозування кібератак. Різні алгоритми класифікації, такі як дерева рішень, випадковий ліс, гаусівський наївний Баєс, метод опорних векторів та інші, можуть бути застосовані для побудови моделей, які класифікують дані та розпізнають вразливості або відхилення.

У процесі оцінки ефективності цих алгоритмів використовуються різні метрики, такі як точність, повнота, F-міра, ROC-крива та інші. Ці метрики дозволяють оцінити якість класифікаційних моделей та їх здатність правильно визначати кіберзагрози.

Практичне застосування алгоритмів ІАД у виявленні кіберзагроз має велике значення та потенціал для захисту інформаційних систем від шкідливих чинників. Вони допомагають виявляти нові типи загроз та адаптуватися до

змінних обставин. Комбінування різних алгоритмів та використання відповідних метрик дозволяє створити ефективні моделі для виявлення кіберзагроз.

РОЗДІЛ 3.

ПРАКТИЧНЕ ЗАСТОСУВАННЯ АЛГОРИТМІВ ІАД ДЛЯ РОЗВ'ЯЗАННЯ ПРОБЛЕМ КІБЕРБЕЗПЕКИ

3.1 Опис проблеми та практичне використання алгоритмів ІАД

3.1.1 Опис проблем

Алгоритми ІАД, як було описано раніше, надають інструменти для покращення ефективності виявлення кіберзагроз. У цій частині буде описано використання алгоритмів класифікації для вирішення проблем кібербезпеки. Дослідження буде проводитись на трьох наборах даних: набір даних для виявлення підозрілих веб-сайтів (фішинг), дані для статичного дослідження ПЗ на ОС Android, опираючись на дозволи, які програма вимагає, та дані для динамічного дослідження ПЗ на ОС Android, опираючись на мережевий трафік програм.

3.1.2 Опис кроків застосування алгоритмів ІАД для вирішення проблем

Застосування алгоритмів ІАД вимагає клопіткої роботи. Оскільки для обраних проблем будуть використовуватись алгоритми навчання під наглядом, вхідні дані мають бути добре підготовленими. Також, вимагається уважність під час навчання та перевірки моделей. Слід звертати увагу на можливий шум у наборі даних та робити відповідні дії, щоб дослідження було максимально ефективним.

Послідовність кроків застосування ІАД було наведено у першому розділі: збір даних, підготовка даних, створення моделей та аналіз моделей. Враховуючи особливість проблем, взятих для розв'язування, можна виділити наступні особливості:

- Збір даних. У цьому випадку збір даних не проводиться. Використовуються вже сформовані набори даних, які були створені для

використання алгоритмами ІАД. Ці набори є опубліковані на веб-сайті <https://www.kaggle.com/>.

- Підготовка даних. Дані, які використовуються можуть мати різноманітні проблеми, які погано сприймаються (або взагалі не сприймаються) моделями. Наприклад: пусті входження, дублікати, шум, надто відокремлені дані, неправильна типізація даних та багато інших. Тому необхідно уважно дослідити вхідні дані та провести редагування набору, де це є необхідним.
- Створення моделей. Коли дані є підготовленими, можна передавати їх моделям. Для цих досліджень було обрано 7 алгоритмів: дерево рішень, випадковій ліс, логістична регресія, AdaBoost, метод опорних векторів для класифікації, алгоритм k-найближчих сусідів та гаусівський наївний класифікатор Баєса.
- Аналіз моделей. Усі моделі були протестовані наступними метриками: точністю (Accuracy), позитивна точністю (Precision), повнотою (Recall), F_1 -мірою (F_1 -score), матрицею помилок та ROC-кривою.

3.2 Огляд та підготовка даних

3.2.1 Опис обраних наборів даних

Дослідження використовує 3 набори даних: дані шкідливих веб-сайтів (фішинг), дані дозволів шкідливих програм на ОС Android та дані мережевого трафіку шкідливих програм на ОС Android.

3.2.1.1 Набір даних ознак шкідливих веб-сайтів (фішингу) (Набір даних №1)

Цей набір даних містить 48 ознак (колонок (не враховуючи колонки id та цільова)), отриманих з 5000 шкідливих (фішинг) веб-сторінок та 5000 легітимних веб-сторінок. Дані формувались з січня по травень 2015 року та з травня по

червень 2017 року. Використовується покращений метод отримання ознак, використовуючи фреймворк автоматизації браузера (Selenium WebDriver), який є більш точним та надійним порівняно з підходом парсингу на основі регулярних виразів. Набір даних має цільову ознаку (колонку) «CLASS_LABEL», яка є булевим значенням типу int64: 0 означає, що веб-сайт є зловмисним, а 1 – веб-сайт є легітимний. Джерелом даних є веб-сайт <https://data.mendeley.com/>.

3.2.1.2 Набір даних з дозволами шкідливого ПЗ (Набір даних №2)

Цей набір даних був створений Крістіаном Уркукі та Андресом Наварро, поширений на платформі <https://www.kaggle.com/>. Дані були отримані шляхом створення бінарного вектора дозволів, який був використаний для кожної проаналізованої програми. Цей набір даних містить 329 ознак (колонок (не враховуючи цільової колонки «android»)) та 398 входжень (проаналізованих програм). Кожна ознака – це дозвіл в ОС Android, де значення 1 означає використовується, а 0 – не використовується. Набір даних має цільову ознаку (колонку) «type», яка є булевим значенням типу int64: 0 означає легітимне ПЗ, а 1 – зловмисне.

3.2.1.3 Набір даних з мережевим трафіком шкідливого ПЗ (Набір даних №3)

Цей набір даних містить інформацію про мережевий трафік ПЗ на ОС Android. Набір був побудований на основі іншого набору «DroidCollector», який був перероблений та сформований у таблицю. Дані поширені Крістіаном Уркукі на платформі <https://www.kaggle.com/>. Цей набір даних містить 15 ознак (колонок (не враховуючи цільової колонки та колонки «name»)) та 7845 входжень (проаналізованих програм). Набір даних має цільову ознаку (колонку) «type», яка є булевим значенням типу object: benign означає легітимне ПЗ, а malicious – зловмисне.

3.2.2 Огляд обраних наборів даних

Для читання даних, які зберігаються у форматі «.csv», можна використати бібліотеку «pandas». Методом «info()» можна отримати інфографіку по наборам даних (див. рис. 3.2.3). Цей метод виводить інформацію про усі колонки (якщо їх не є забагато): кількість входжень, входження, які не є пустими та тип входжень.

За допомогою методу «value_counts()» можна підрахувати кількість входжень кожного типу для вибраної ознаки. Наприклад, для цільової ознаки ці значення відрізняються для кожного набору даних. Набір №1 має 5000 легітимних та 5000 зловмисних входжень, набір №2 має 199 легітимних та 199 зловмисних входжень, а набір №3 має 4704 легітимних та 3141 зловмисних входжень. Дані зображено на графіках (див. рис. 3.2.2):

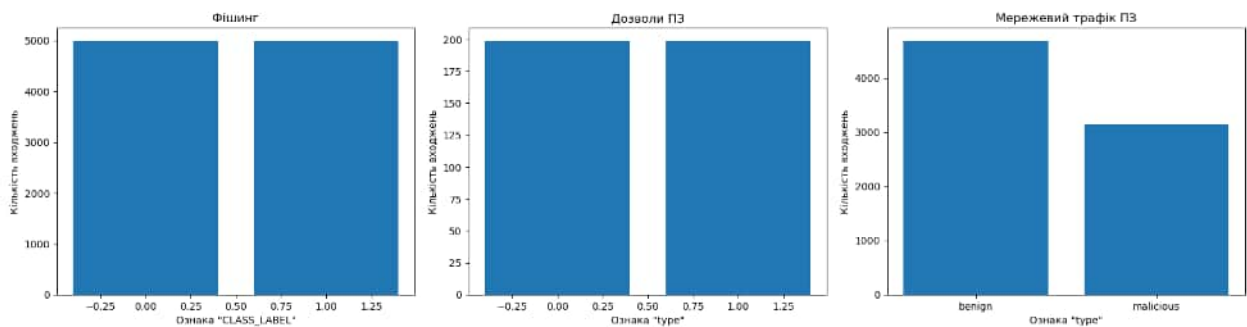


Рисунок 3.2.2: Графіки кількості входжень для цільових ознак для кожного набору даних

Набір даних ознак шкідливих веб-сайтів (фішингу)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   id                                           10000 non-null  int64
1   NumDots                                       10000 non-null  int64
2   SubdomainLevel                             10000 non-null  int64
3   PathLevel                                    10000 non-null  int64
4   UrlLength                                    10000 non-null  int64
5   NumDash                                       10000 non-null  int64
6   NumDashInHostname                          10000 non-null  int64
7   AtSymbol                                      10000 non-null  int64
8   TildeSymbol                                  10000 non-null  int64
9   NumUnderscore                               10000 non-null  int64
10  NumPercent                                   10000 non-null  int64
11  NumQueryComponents                          10000 non-null  int64
12  NumAmpersand                                 10000 non-null  int64
13  NumHash                                       10000 non-null  int64
14  NumNumericChars                             10000 non-null  int64
15  NoHttps                                       10000 non-null  int64
16  RandomString                                 10000 non-null  int64
17  IPAddress                                    10000 non-null  int64
18  DomainInSubdomains                          10000 non-null  int64
19  DomainInPaths                               10000 non-null  int64
20  HttpsInHostname                             10000 non-null  int64
21  HostnameLength                              10000 non-null  int64
22  PathLength                                   10000 non-null  int64
23  QueryLength                                 10000 non-null  int64
24  DoubleSlashInPath                           10000 non-null  int64
25  NumSensitiveWords                           10000 non-null  int64
26  EmbeddedBrandName                           10000 non-null  int64
27  PctExtHyperlinks                            10000 non-null  float64
28  PctExtResourceUrls                          10000 non-null  float64
29  ExtFavicon                                   10000 non-null  int64
30  InsecureForms                               10000 non-null  int64
31  RelativeFormAction                          10000 non-null  int64
32  ExtFormAction                               10000 non-null  int64
33  AbnormalFormAction                          10000 non-null  int64
34  PctNullSelfRedirectHyperlinks               10000 non-null  float64
35  FrequentDomainNameMismatch                  10000 non-null  int64
36  FakeLinkInStatusBar                         10000 non-null  int64
37  RightClickDisabled                          10000 non-null  int64
38  PopUpWindow                                 10000 non-null  int64
39  SubmitInfoToEmail                           10000 non-null  int64
40  IframeOrFrame                              10000 non-null  int64
41  MissingTitle                                10000 non-null  int64
42  ImagesOnlyInForm                            10000 non-null  int64
43  SubdomainLevelRT                            10000 non-null  int64
44  UrlLengthRT                                  10000 non-null  int64
45  PctExtResourceUrlsRT                        10000 non-null  int64
46  AbnormalExtFormActionR                      10000 non-null  int64
47  ExtMetaScriptLinkRT                         10000 non-null  int64
48  PctExtNullSelfRedirectHyperlinksRT         10000 non-null  int64
49  CLASS_LABEL                                 10000 non-null  int64
dtypes: float64(3), int64(47)
memory usage: 3.8 MB
```

Набір даних з дозволами шкідливого ПЗ

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Columns: 331 entries, android to type
dtypes: int64(331)
memory usage: 1.0 MB
```

Набір даних з мережевим трафіком шкідливого ПЗ

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7845 entries, 0 to 7844
Data columns (total 17 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   name                                       7845 non-null  object
1   tcp_packets                               7845 non-null  int64
2   dist_port_tcp                             7845 non-null  int64
3   external_ips                              7845 non-null  int64
4   vulume_bytes                              7845 non-null  int64
5   udp_packets                               7845 non-null  int64
6   tcp_urg_packet                            7845 non-null  int64
7   source_app_packets                        7845 non-null  int64
8   remote_app_packets                        7845 non-null  int64
9   source_app_bytes                          7845 non-null  int64
10  remote_app_bytes                           7845 non-null  int64
11  duracion                                   0 non-null     float64
12  avg_local_pkt_rate                         0 non-null     float64
13  avg_remote_pkt_rate                        0 non-null     float64
14  source_app_packets.1                       7845 non-null  int64
15  dns_query_times                            7845 non-null  int64
16  type                                       7845 non-null  object
dtypes: float64(3), int64(12), object(2)
memory usage: 1.0+ MB
```

Рисунок 3.2.2: Інфографіка наборів даних

3.2.3 Підготовка даних до моделювання

Щоб підготувати дані до етапу моделювання, було виконано наступні дії:

3.2.3.1 Набір даних ознак шкідливих веб-сайтів (фішингу) (Набір даних №1)

- 1) Опираючись на інфографіку (рисунок 3.2.2), усі значення є числовими (типи «int64» та «float64») та не мають пропусків. У плані недоліків та шуму набір даних не потребує додаткового очищення чи редагування.

- 2) Слід видалити колонку «id», оскільки вона є унікальною для кожного входження і не допомагає моделям досягнути мети.
- 3) Більшість ознак мають мало унікальних значень. Переглянути усі унікальні входження можна методом «unique», а їх кількість методом «nunique». Ця інформація дозволяє не проводити очищення відхилень серед набору даних.
- 4) Тепер необхідно визначити, які ознаки не сильно впливають на цільову ознаку «CLASS_LABEL». Для цього можна використати, наприклад, матрицю кореляцій.

Матриця кореляцій - це статистичний інструмент, який використовується для вивчення зв'язків (кореляцій) між змінними. Вона відображає ступінь лінійної залежності між парами змінних. Кожен елемент матриці кореляцій вказує на силу та напрямок зв'язку між двома змінними: значення +1 вказує на досконалу позитивну кореляцію, -1 - на досконалу негативну кореляцію, а значення 0 означає відсутність кореляції. Таким чином, матриця кореляцій дозволяє виявляти взаємозв'язки між змінними та визначати, наскільки сильно одна змінна залежить від іншої.

Таку матрицю можна вивести функцією «heatmap» з бібліотеки «seaborn». Після аналізу матриці кореляцій було видалено 18 ознак з найменшим модулем кореляції, включаючи з ознакою «HttpsInHostname», яка немає кореляції через відсутність варіативності входжень (усі входження цієї ознаки рівні 0) (див. таблицю 3.2.3.1).

Назва ознаки	Модуль кореляції
RelativeFormAction	0.083
DoubleSlashInPath	0.023
HttpsInHostname	-
DomainInSubdomains	0.01
FakeLinkInStatusBar	0.015
RandomString	0.085
EmbeddedBrandName	0.14
AtSymbol	0.017
ImagesOnlyInForm	0.001
NumHash	0.048
AbnormalFormAction	0.15
UrlLengthRT	0.17
PctExtResourceUrlsRT	0.052
PopUpWindow	0.067
RightClickDisabled	0.075
IpAddress	0.13
SubdomainLevelRT	0.076
TildeSymbol	0.096

Таблиця 3.2.3.1: Видалені ознаки, опираючіть на значення кореляції

3.2.3.2 Набір даних з дозволами шкідливого ПЗ (Набір даних №2)

- 1) Усі входження є числовими та непорожніми – очищувати дані непотрібно.
- 2) Необхідно видалити ключову колонку «android», яка не впливає на результат.
- 3) Оскільки набір даних має більше 300-а ознак, побудувати матрицю кореляцій буде проблематично. Проте, через особливість даних, усі входження для усіх ознак є булевими значеннями 0 та 1. Це дозволяє тренувати моделі на такому наборі без потреби додаткового редагування.

3.2.3.3 Набір даних з мережевим трафіком шкідливого ПЗ (Набір даних №3)

- 1) Опираючись на інфографіку (рисунок 3.2.2), більшість значень є числовими (типи «int64» та «float64»). Ознаки «duration», «avg_local_pkt_rate» та «avg_remote_pkt_rate» є повністю пустими, тож їх необхідно видалити.
- 2) Оглянувши опис набору, який можна переглянути методом «describe()», помітно дивну ознаку «tcp_urg_packet». Середнє значення входжень є дуже низьким – 0.015966. Після побудови графіку цієї ознаки стало зрозуміло, що вона містить лише 2 входження, відмінних від 0. Вплив такої ознаки на цільову змінну майже відсутній, тому її слід видалити.
- 3) Оскільки дані мають багато унікальних значень (див. рис. 3.2.3.3), необхідно звернути увагу на розподіл даних у наборі. Після побудови графіків стало помітно, що у наборі багато відхилень. Входження, які сильно відхиляються від групи інших значень також слід видалити, щоб не викривлювати результат.

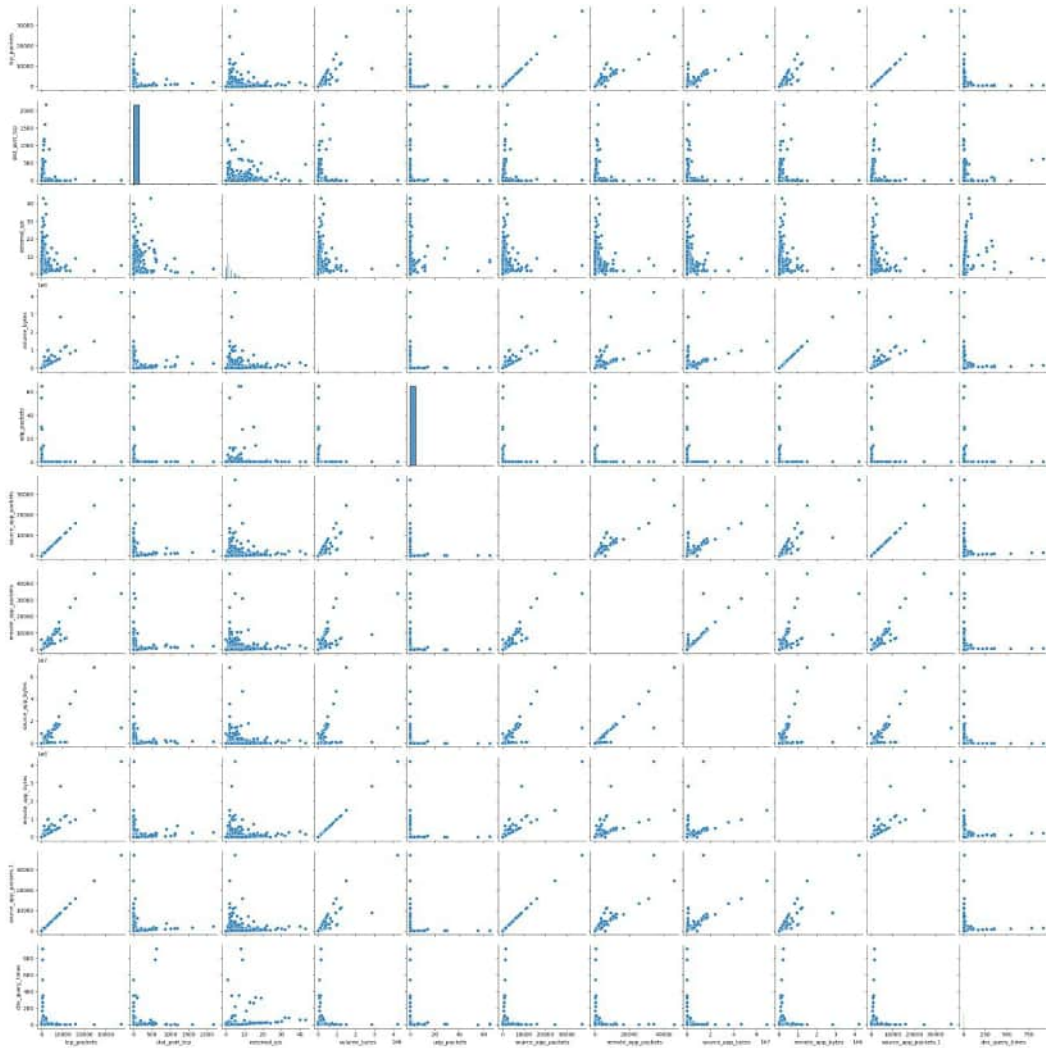


Рисунок 3.2.3.3: Графіки попарних залежностей ознак

- 4) При аналізі даних було виявлену зайву ознаку «source_app_packets.1», яка повністю дублює ознаку «source_app_packets». Тому ознаку «source_app_packets.1» було видалено.

Тепер усі набори даних готові до застосування у моделях.

3.3 Моделювання

3.3.1 Ініціалізація моделей та розподіл даних

Коли усі дані підготовлені, їх необхідно розділити на дві частини: дані для тренування «train» та дані для тестування «test». Головна причина розділення на «train» і «test» полягає в тому, щоб модель навчалась на тренувальних даних і

оцінювалась на тестових даних. Тестові дані служать для оцінки точності і ефективності моделі на нових, раніше невидимих прикладах. Це допомагає виявити, наскільки добре модель відповідає на нові дані та уникнути перенавчання (overfitting), коли модель надмірно пристосовується до тренувальних даних і показує погану прогнозу здатність на нових даних.

Для ефективного розподілу необхідно також визначити співвідношення між набором «train» та «test». Зазвичай типове співвідношення може бути 70-80% даних для тренування і 20-30% даних для тестування. Однак, величина тестового набору може залежати від доступних даних і потреби отримати достатньо впевнені оцінки ефективності моделі. Важливо мати достатньо даних у тренувальному наборі для навчання моделі на змістовних закономірностях і мінімізації статистичного шуму.

Також, ці два набори мають розділятися на групи X та Y . Вхідні ознаки X представляють набір незалежних змінних або функцій, які використовуються для прогнозування або класифікації цільової змінної Y . Вони визначають контекст, у якому модель буде прогнозувати або класифікувати дані. Цільова змінна Y визначає значення, яке модель намагається прогнозувати або класифікувати на основі вхідних ознак X . У задачах регресії це може бути неперервна змінна, наприклад, ціна на нерухомість. У задачах класифікації це може бути дискретна змінна, така як мітка класу, яка вказує, чи є об'єкт "шкідливим" або "легітимним". Розділення на X та Y дозволяє окремо працювати з вхідними ознаками та цільовою змінною, а також подальшу підготовку інформації для навчання та тестування моделі. Вхідні ознаки X використовуються для навчання моделі, тоді як цільова змінна Y використовується для порівняння прогнозів моделі зі справжніми значеннями під час оцінки її ефективності.

У дослідженні розподіл усіх наборів на «train» та «test» відбувався зі співвідношенням 1 до 5 (20% для набору «test»). Це відбувається функцією «train_test_split()» з бібліотеки «sklearn» (scikit-learn).

Коли дані розподілені, відбувається ініціалізація обраних моделей: дерева рішень (DecisionTreeClassifier()), випадкового лісу (RandomForestClassifier()), логістичної регресії (LogisticRegression()), алгоритму AdaBoost

(AdaBoostClassifier()), алгоритму методу опорних векторів для класифікації (SVC()), алгоритму k-найближчих сусідів (KNeighborsClassifier()) та гаусівського наївного класифікатора Баєса (GaussianNB()).

3.3.2 Навчання моделей та передбачення на тестових даних

Після ініціалізації моделей розпочинається процес навчання (підгонки). У цьому випадку більш доречно буде використати визначення підгонки (fitting), оскільки процес навчання чи тренування (training) застосовується для алгоритмів машинного навчання, які є алгоритмами без нагляду і не потребують вхідних даних.

Процес підгонки – це процес тренування моделей на навчальних даних з використанням алгоритмів класифікації. Під час підгонки модель аналізує вхідні ознаки X та цільову змінну Y для виявлення закономірностей та залежностей між ними. Модель оптимізує свої внутрішні параметри, щоб найкращим чином адаптуватись до навчальних даних та мінімізувати помилки прогнозування. Цей процес включає ітерації, де модель покращується з кожною ітерацією, поки не досягне заданої точності або не виконані критерії зупинки.

Після навчання необхідно виконати процес прогнозування. Для прогнозування моделі передаються вхідні ознаки X без цільової змінної Y . Модель використовує вивчені внутрішні параметри та застосовує їх до нових даних (тестового набору) для отримання прогнозованих значень. Прогнози можуть бути числовими (у задачах регресії), дискретними (у задачах класифікації) або ймовірностями належності до різних класів.

3.4 Застосування метрик до отриманих моделей та порівняння результатів

3.4.1 Попередні результати передбачення

Для визначення ефективності отриманих моделей можна застосувати функцію «accuracy_score» з бібліотеки «sklearn», яка порівнює зроблені моделями передбачення на наборі даних Y для тестування з реальними класами, які були

наведені в оригінальному наборі даних. Результати зображені на графіках (рисунки 3.4.1.1-3.4.1.3):

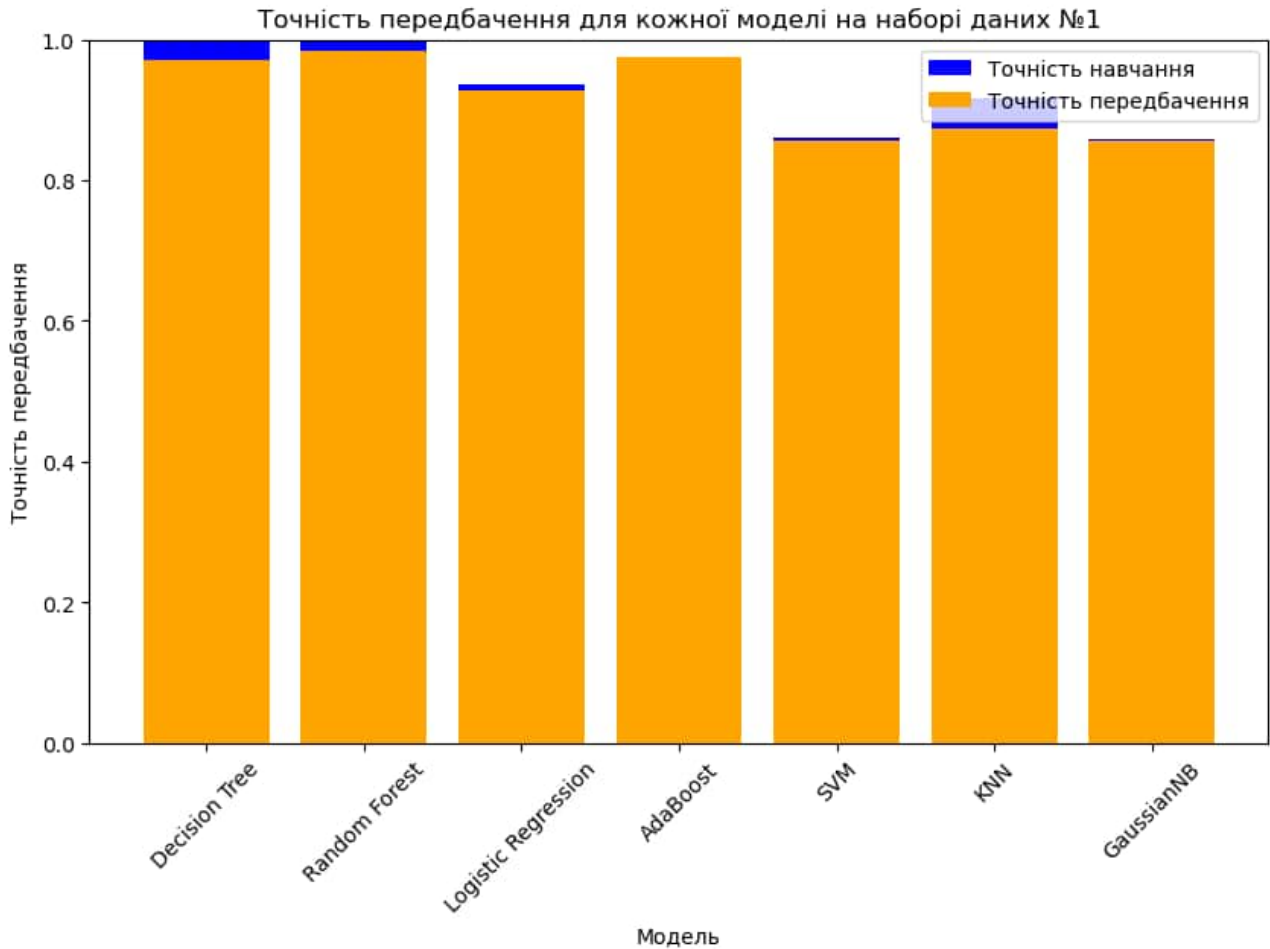


Рисунок 3.4.1.1: Точність передбачення для кожної моделі на наборі даних ознак шкідливих веб-сайтів (фішингу)

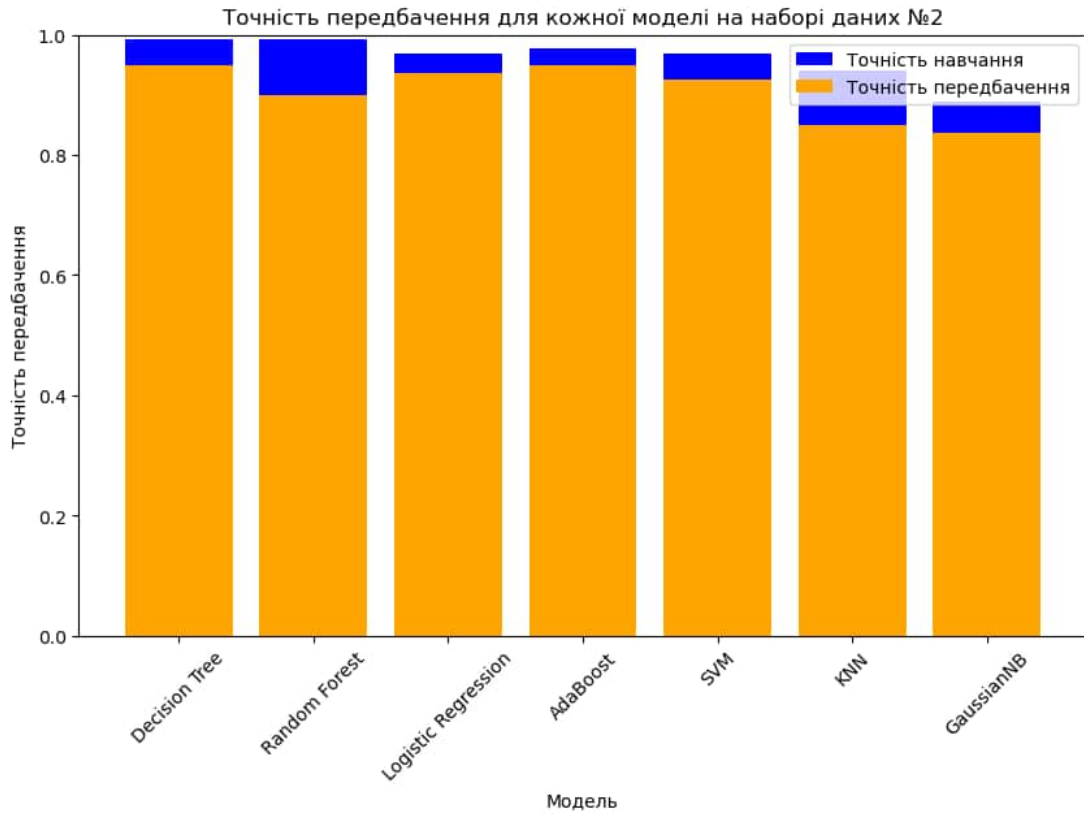


Рисунок 3.4.1.2: Точність передбачення для кожної моделі на наборі даних з дозволами шкідливого ПЗ

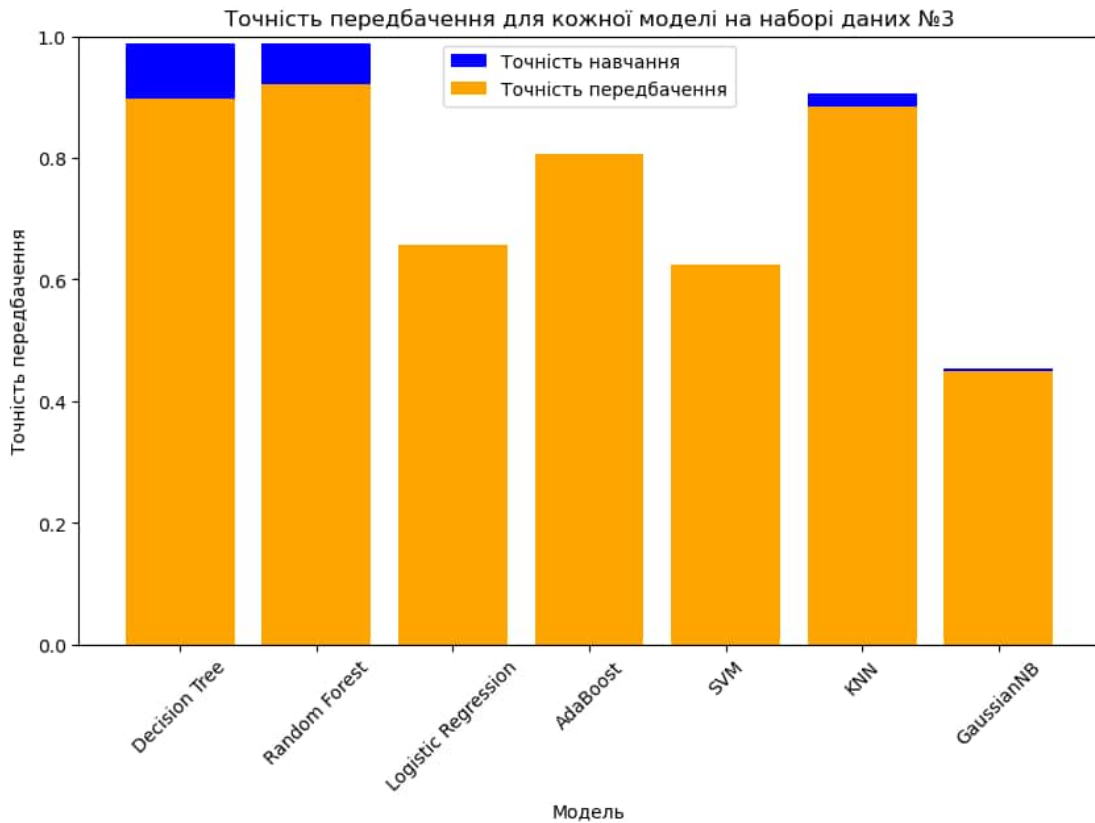


Рисунок 3.4.1.3: Точність передбачення для кожної моделі на наборі даних з мережевим трафіком шкідливого ПЗ

Найбільш ефективними виявились алгоритми дерева рішень та випадкового лісу. Найменш ефективним – наївний класифікатор Баєса. Порівняння отриманих значень точності зображено у таблиці 3.4.1.4:

Назва алгоритму	Значення точності передбачень		
	Набір №1	Набір №2	Набір №3
Дерево рішень (Decision Tree)	0.971	0.95	~ 0.8973
Випадковий ліс (Random Forest)	0.985	0.9	~ 0.9222
Логістична регресія (Logistic Regression)	0.928	0.9375	~ 0.6573
AdaBoost	0.975	0.95	~ 0.806
Метод опорних векторів SVC (SVM)	0.8575	0.925	~ 0.6241
Алгоритм k-найближчих сусідів (KNN)	0.8735	0.85	~ 0.8845
Гусівський наївний класифікатор Баєса (GaussianNB)	0.8565	0.8375	~ 0.4505

Таблиця 3.4.1.4: Порівняння значень коефіцієнта точності передбачень для вибраних моделей

3.4.2 Stacking Classifier, Voting Classifier та ROC-крива

Перед застосуванням метрик слід використати методи ансамблювання алгоритмів, які комбінують прогнози кількох базових моделей для отримання кращих результатів класифікації. Бібліотека «sklearn» надає чимало інструментів для ансамблювання, включаючи Stacking Classifier та Voting Classifier.

Класифікатор накладання (Stacking Classifier) – це метод ансамблювання, де прогнози базових класифікаторів використовуються як вхідні дані для третього рівня моделі, яка називається мета-класифікатором. Класифікатор накладання може навчитись визначати, коли можна довіряти базовим класифікаторам, а коли ні. Накладання дозволяє використовувати сильні сторони кожного окремого оцінювача, використовуючи їх виходи як вхідні дані для кінцевого оцінювача. При використанні цього класифікатора можна вибрати застосування перехресної перевірки на рівні базового моделювання або на рівні кінцевого оцінювача. Використовуючи класифікатор накладання з бібліотеки «scikit-learn», базові моделі підганяються на повному наборі даних X , тоді як

кінцевий оцінювач навчається за допомогою перехресно перевірених прогнозів базових моделей.

Класифікатор голосування (Voting Classifier). Voting Classifier – це метод ансамблювання, в якому кілька базових класифікаторів голосують для визначення кінцевого класу. Кожен базовий класифікатор дає свій прогноз для кожного екземпляра, і результат визначається шляхом голосування більшості або застосування певного правила. У реалізації цього класифікатора у бібліотеці «scikit-learn» можна вибрати між жорстким (hard) та м'яким (soft) типами голосування. Жорстке голосування застосовується до передбачуваних міток класів для голосування за більшість. Воно використовує ідею "більшість вирішує" - тобто рішення приймається на користь того, хто отримав більше половини голосів. М'яке голосування передбачає мітку класу на основі «argmax» сум передбачуваних ймовірностей окремих оцінювачів, що складають ансамбль. М'яке голосування часто рекомендується у випадку ансамблю добре каліброваних/натренованих класифікаторів.

У випадку дослідження ці класифікатори ансамблювання були використані на двох найкращих алгоритмах: дерева рішень та випадкового лісу. Класифікатор голосування використовує налаштування жорсткого голосування. Після отримання результатів передбачення було отримано наступні значення коефіцієнта точності (див. таблицю. 3.4.2.1):

Назва алгоритму ансамблювання	Значення точності передбачень		
	Набір №1	Набір №2	Набір №3
Класифікатор накладання (Stacking Classifier)	0.9835	0.9125	~ 0.9196
Класифікатор голосування (Voting Classifier)	0.98	0.9125	~ 0.9081

Таблиця 3.4.2.1: Порівняння значень коефіцієнта точності передбачень для алгоритмів ансамблювання

Після цього було виведено ROC-криву для кожного з наборів даних (див. рис. 3.4.2.2-3.4.2.4):

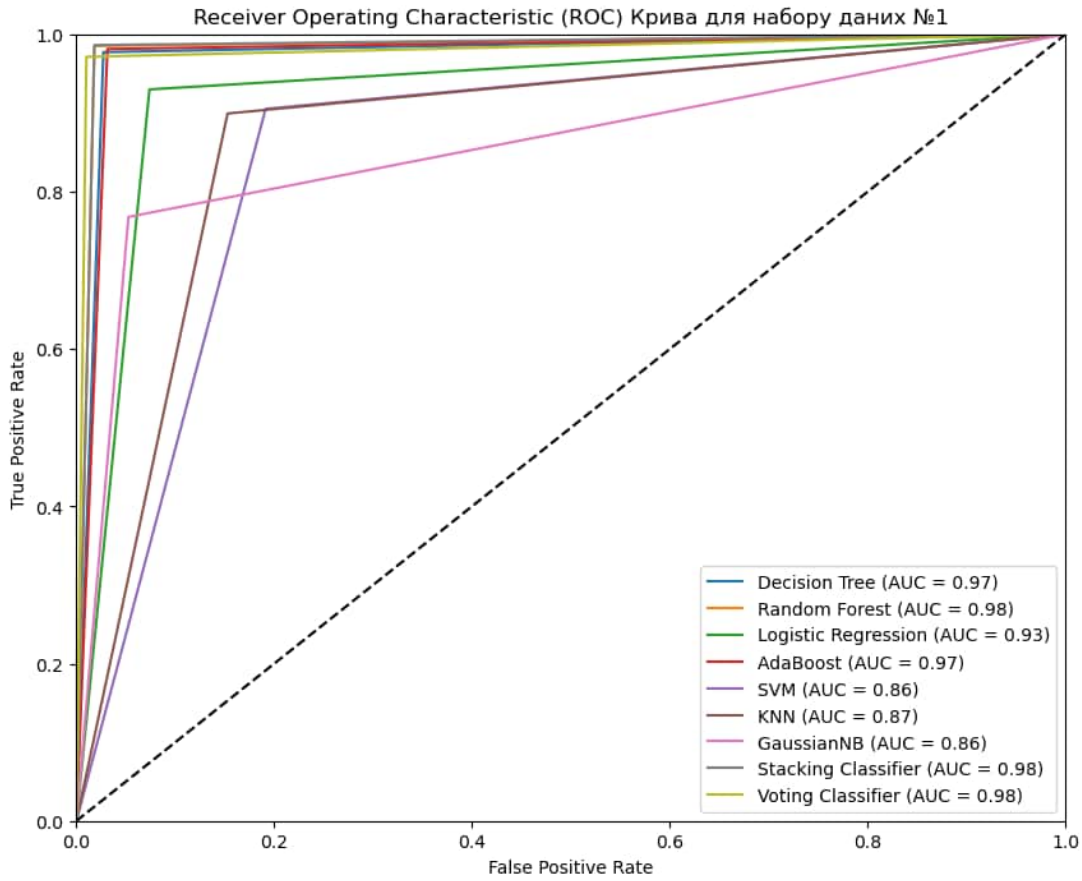


Рисунок 3.4.2.2: ROC-крива для набору даних №1

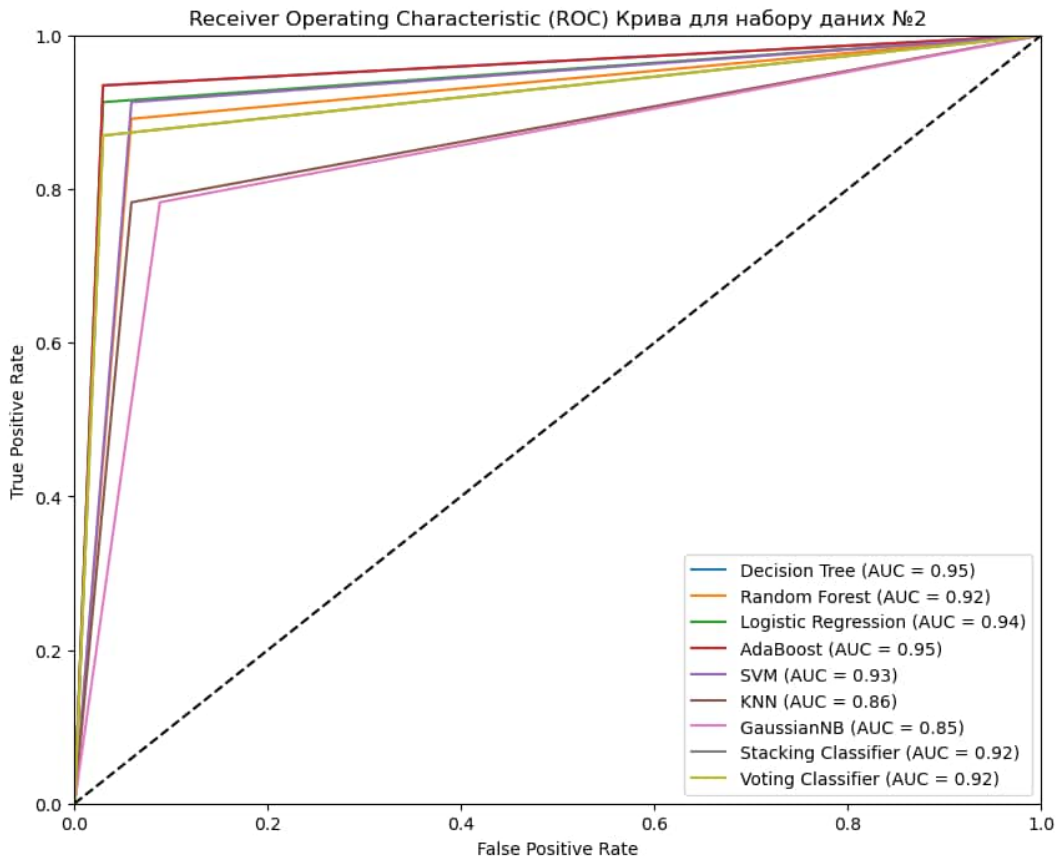


Рисунок 3.4.2.3: ROC-крива для набору даних №2

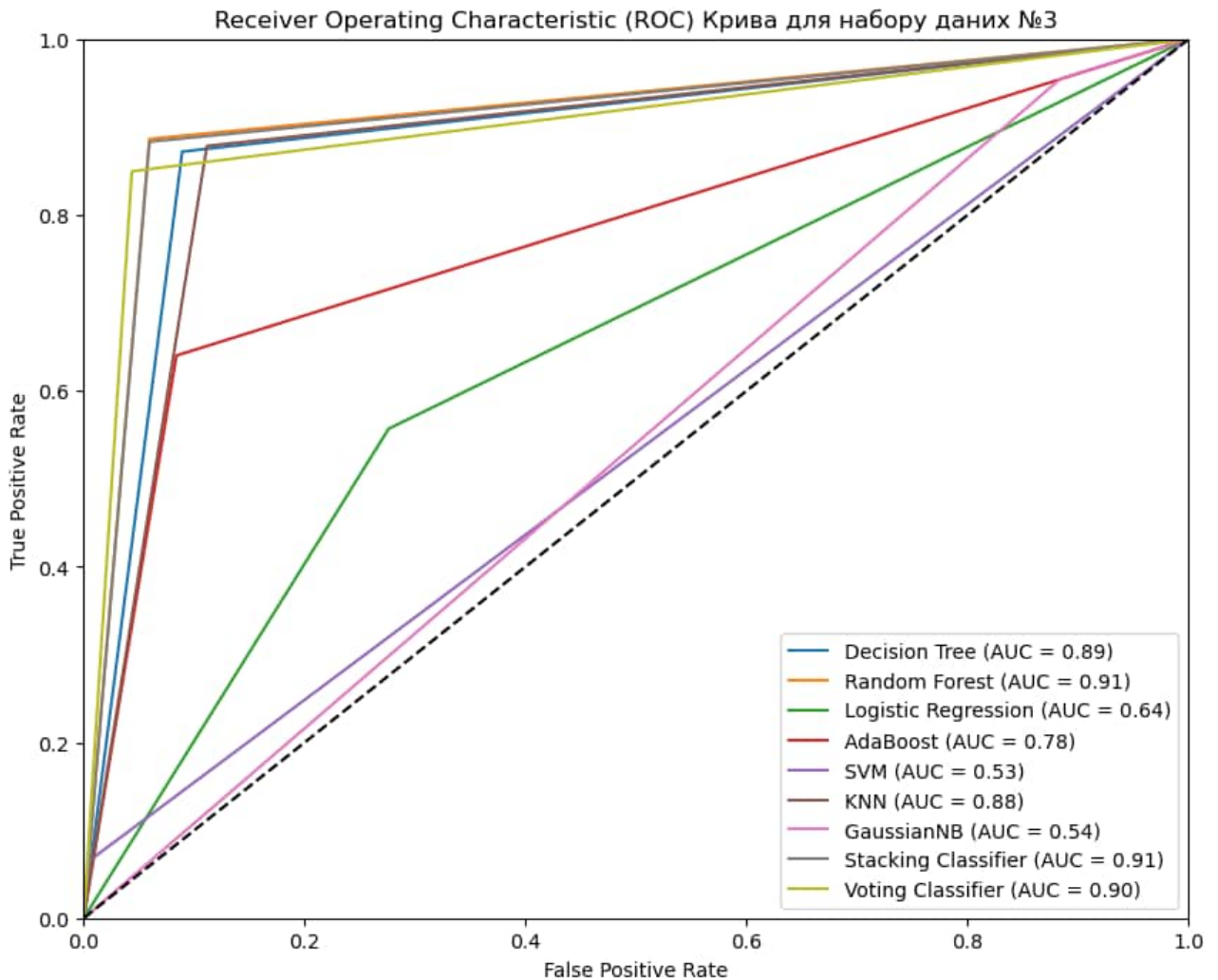


Рисунок 3.4.2.4: ROC-крива для набору даних №3

Ефективність кожного з алгоритмів можна визначити площею під кривою (Area Under the Curve (AUC)): чим більше значення площі – тим ефективнішим є алгоритм класифікації. З результатів кривих помітно, що найбільш ефективними є зазвичай комбіновані алгоритми ансамблювання. Алгоритми SVM та GaussianNB при тренуванні на наборі даних №3 показують лише трохи кращий результат за випадкове вгадування (AUC випадкового вгадування рівний 0.5) – 0.53 та 0.54 відповідно.

3.4.3 Застосування метрик до отриманих моделей

Щоб мати повне уявлення про результат дослідження, до натренованих моделей необхідно також застосувати усі згадані метрики: точність (Accuracy), позитивна

точність (Precision), повнота (Recall) та F_1 -міру (F_1 -score). Результати застосування метрик для моделей зображено на таблицях 3.4.3.1-3.4.3.3:

Результати використання методів ІАД на наборі даних №1					
Назва алгоритму класифікації	Матриця помилок	Точність (Accuracy)	Позитивна точність (Precision)	Повнота (Recall)	F_1 -міра (F_1 -score)
Дерево рішень (Decision Tree)	$\begin{vmatrix} 961 & 27 \\ 23 & 989 \end{vmatrix}$	0.975	~ 0.9734	~ 0.9773	~ 0.9753
Випадковий ліс (Random Forest)	$\begin{vmatrix} 970 & 18 \\ 15 & 997 \end{vmatrix}$	0.9835	~ 0.9823	~ 0.9852	~ 0.9837
Логістична регресія (Logistic Regression)	$\begin{vmatrix} 915 & 73 \\ 71 & 941 \end{vmatrix}$	0.928	~ 0.928	~ 0.9298	~ 0.9289
AdaBoost	$\begin{vmatrix} 957 & 31 \\ 19 & 993 \end{vmatrix}$	0.975	~ 0.9697	~ 0.9812	~ 0.9754
Метод опорних векторів SVC (SVM)	$\begin{vmatrix} 799 & 189 \\ 96 & 916 \end{vmatrix}$	0.8575	~ 0.829	~ 0.9051	~ 0.8653
Алгоритм k-найближчих сусідів (KNN)	$\begin{vmatrix} 837 & 151 \\ 102 & 910 \end{vmatrix}$	0.8735	~ 0.8577	~ 0.8992	~ 0.878
Наївний класифікатор Баєса (GaussianNB)	$\begin{vmatrix} 936 & 52 \\ 235 & 777 \end{vmatrix}$	0.8565	~ 0.9373	~ 0.7678	~ 0.8441
Класифікатор накладання (Stacking Classifier)	$\begin{vmatrix} 970 & 18 \\ 14 & 998 \end{vmatrix}$	0.984	~ 0.9823	~ 0.9862	~ 0.9842
Класифікатор голосування (Voting Classifier)	$\begin{vmatrix} 978 & 10 \\ 29 & 983 \end{vmatrix}$	0.9805	~ 0.9899	~ 0.9713	~ 0.9805

Таблиця 3.4.3.1: Порівняння результатів використання методів ІАД на наборі даних №1, використовуючи метрики

Результати використання методів ІАД на наборі даних №2					
Назва алгоритму класифікації	Матриця помилок	Точність (Accuracy)	Позитивна точність (Precision)	Повнота (Recall)	F_1 -міра (F_1 -score)
Дерево рішень (Decision Tree)	$\begin{vmatrix} 33 & 1 \\ 3 & 43 \end{vmatrix}$	0.95	~ 0.9773	~ 0.9348	~ 0.9556
Випадковий ліс (Random Forest)	$\begin{vmatrix} 32 & 2 \\ 5 & 41 \end{vmatrix}$	0.9125	~ 0.9535	~ 0.8913	~ 0.9213
Логістична регресія (Logistic Regression)	$\begin{vmatrix} 33 & 1 \\ 4 & 42 \end{vmatrix}$	0.9375	~ 0.9767	~ 0.9130	~ 0.9438
AdaBoost	$\begin{vmatrix} 33 & 1 \\ 3 & 43 \end{vmatrix}$	0.95	~ 0.9773	~ 0.9348	~ 0.9556
Метод опорних векторів SVC (SVM)	$\begin{vmatrix} 32 & 2 \\ 4 & 42 \end{vmatrix}$	0.925	~ 0.9545	~ 0.9130	~ 0.9333
Алгоритм k-найближчих сусідів (KNN)	$\begin{vmatrix} 32 & 2 \\ 10 & 36 \end{vmatrix}$	0.85	~ 0.9474	~ 0.7826	~ 0.8571
Наївний класифікатор Баєса (GaussianNB)	$\begin{vmatrix} 31 & 3 \\ 10 & 36 \end{vmatrix}$	0.8375	~ 0.9231	~ 0.7826	~ 0.8471
Класифікатор накладання (Stacking Classifier)	$\begin{vmatrix} 33 & 1 \\ 6 & 40 \end{vmatrix}$	0.9125	~ 0.9756	~ 0.8696	~ 0.9195
Класифікатор голосування (Voting Classifier)	$\begin{vmatrix} 33 & 1 \\ 6 & 40 \end{vmatrix}$	0.9125	~ 0.97560	~ 0.8696	~ 0.9195

Таблиця 3.4.3.2: Порівняння результатів використання методів ІАД на наборі даних №2, використовуючи метрики

Результати використання методів ІАД на наборі даних №3					
Назва алгоритму класифікації	Матриця помилок	Точність (Accuracy)	Позитивна точність (Precision)	Повнота (Recall)	F_1 -міра (F_1 -score)
Дерево рішень (Decision Tree)	$\begin{vmatrix} 858 & 84 \\ 80 & 545 \end{vmatrix}$	~ 0.8953	~ 0.8665	0.872	~ 0.8692
Випадковий ліс (Random Forest)	$\begin{vmatrix} 886 & 56 \\ 71 & 554 \end{vmatrix}$	~ 0.919	~ 0.9082	0.8864	~ 0.8972
Логістична регресія (Logistic Regression)	$\begin{vmatrix} 682 & 260 \\ 277 & 348 \end{vmatrix}$	~ 0.6573	~ 0.5724	0.5568	~ 0.5645
AdaBoost	$\begin{vmatrix} 863 & 79 \\ 225 & 400 \end{vmatrix}$	~ 0.806	~ 0.8351	0.64	~ 0.7246
Метод опорних векторів SVC (SVM)	$\begin{vmatrix} 937 & 5 \\ 584 & 41 \end{vmatrix}$	~ 0.6241	~ 0.8913	0.0656	~ 0.1222
Алгоритм k-найближчих сусідів (KNN)	$\begin{vmatrix} 837 & 105 \\ 76 & 549 \end{vmatrix}$	~ 0.8845	~ 0.8394	0.8784	~ 0.8585
Наївний класифікатор Баєса (GaussianNB)	$\begin{vmatrix} 109 & 833 \\ 28 & 597 \end{vmatrix}$	~ 0.4505	~ 0.4175	0.9552	~ 0.5810
Класифікатор накладання (Stacking Classifier)	$\begin{vmatrix} 886 & 56 \\ 73 & 552 \end{vmatrix}$	~ 0.9177	~ 0.9079	0.8832	~ 0.8954
Класифікатор голосування (Voting Classifier)	$\begin{vmatrix} 901 & 41 \\ 94 & 531 \end{vmatrix}$	~ 0.9138	~ 0.9283	0.8496	~ 0.8872

Таблиця 3.4.3.3: Порівняння результатів використання методів ІАД на наборі даних №3, використовуючи метрики

З результатів застосування метрик можна отримати інформацію про найбільш ефективний алгоритм ІАД для аналізу обраних наборів даних. У середньому, найкращими алгоритмами виявились: дерево рішень, випадковий ліс та ансамблеві класифікатори, побудовані на їх основі.

Висновки до розділу 3

У цьому розділі проведено застосування алгоритмів класифікації на трьох наборах даних з метою визначення їх ефективності. Перед початком моделювання була виконана підготовка даних, що включала очищення, видалення відсутніх значень, зайвих ознак тощо.

Після підготовки даних були використані різні алгоритми класифікації, такі як дерево рішень, випадковий ліс, логістична регресія, AdaBoost, SVM, KNN, GaussianNB, класифікатор накладання та класифікатор голосування. Для кожного алгоритму було проведено навчання моделі, оцінку результатів та аналіз отриманих результатів.

Результати аналізу метрик показали, що моделі мають різну ефективність на різних наборах даних. У середньому найбільш ефективними для розв'язання подібних проблем є алгоритми: дерево рішень, випадковий ліс та комбіновані класифікатори. Найменш ефективними були алгоритми наївного Баєса та методу опорних векторів (SVM).

На підставі результатів можна зробити висновок, що під час цього дослідження було успішно застосовано алгоритми класифікації на трьох наборах даних. Деякі моделі виявилися більш ефективними, забезпечуючи високу точність та повноту класифікації. Враховуючи ці результати, можна рекомендувати використання певних моделей для подальшої роботи з аналізом та класифікацією подібних даних.

ВИСНОВКИ

У цій дипломній роботі було розглянуто та описано основні типи кіберзагроз та технологію «інтелектуального аналізу даних». У першому розділі було представлено загальну інформацію про кібербезпеку, її важливість у сучасному світі та основні загрози, з якими стикаються організації. Було описано поняття "Інтелектуальний аналіз даних" і його роль у кібербезпеці. Також було описано найбільш популярні та небезпечні кіберзагрози та рівні мережі, на яких вони діють.

Другий розділ присвячений детальному опису практичного застосування ІАД в кібербезпеці. Було розглянуто різні методи та алгоритми, використовувані в ІАД, такі як класифікація, кластеризація, виявлення аномалій тощо. Описано, як ці методи можуть бути застосовані для виявлення та класифікації загроз в кіберпросторі, на прикладі виявлення шкідливого ПЗ. Також, було описано способи аналізу ефективності моделей ІАД, включаючи метрики (точність, позитивна точність, повнота тощо) та ROC-криву.

У третьому розділі було проведено практичне застосування алгоритмів ІАД для розв'язання проблем кібербезпеки. Було використано три набори реальних даних, пов'язані з фішингом та шкідливим ПЗ, і проведено аналіз за допомогою різних алгоритмів ІАД. Отримані результати були проаналізовані за допомогою метрик (таких як точність, повнота, специфічність, F_1 -міра тощо) для оцінки ефективності алгоритмів.

Результатом дослідження є успішне застосування алгоритмів класифікації на трьох наборах даних. Деякі моделі виявилися більш ефективними, забезпечуючи високу точність та повноту класифікації. Також, були випадки, коли моделі виявились лише дещо кращими за випадкове вгадування, що свідчить про їх непридатність до використання на певних наборах даних.

Застосування алгоритмів ІАД у кібербезпеці має великий потенціал. Вони дозволяють виявляти, класифікувати та розрізняти загрози, а також прогнозувати майбутні атаки. Результати аналізу показали, що застосування алгоритмів ІАД

може покращити ефективність систем кібербезпеки та допомогти вчасно виявляти та реагувати на потенційні загрози.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. <https://www.techtarget.com/searchbusinessanalytics/definition/data-mining>
2. SlashNEXT - The State of Phishing 2022
3. <https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80>
4. https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BA%D0%BE%D0%BB%D1%96%D0%BD%D0%BD%D1%8F_Z
5. <https://ourworldindata.org/grapher/historical-cost-of-computer-memory-and-storage>
6. <https://www.helpnetsecurity.com/2019/05/23/connected-devices-growth/>
7. https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D0%BA%D0%B0
8. <https://www.cimcor.com/blog/cybersecurity-lifecycle>
9. https://uk.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%BE%D0%B1%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B7%D0%BD%D0%B0%D0%BD%D1%8C
10. <https://www.ibm.com/topics/data-mining>
11. <https://unstop.com/blog/regression-in-data-mining>
12. <https://sisudata.com/blog/what-is-anomaly-detection-in-data-mining>
13. <https://www.aplustopper.com/data-mining-advantages-and-disadvantages/>
14. <https://www.javatpoint.com/osi-vs-tcp-ip>
15. https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%80%D0%B5%D0%B6%D0%B5%D0%B2%D0%B0_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI
16. <https://www.imperva.com/learn/application-security/osi-model/>
17. <https://www.imperva.com/learn/application-security/cyber-security-threats/>
18. CYBER ATTACKS AND ITS DIFFERENT TYPES - International Research Journal of Engineering and Technology (IRJET)

19. https://uk.wikipedia.org/wiki/%D0%A2%D1%80%D0%BE%D1%8F%D0%BD%D1%81%D1%8C%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0
20. https://aws.amazon.com/shield/ddos-attack-protection/?nc1=h_ls
21. Data Mining Approach for Cyber Security - International Journal of Computer Applications Technology and Research
22. <https://www.ibm.com/topics/decision-trees>
23. <https://www.educba.com/data-mining-algorithms/>
24. <https://www.upgrad.com/blog/common-data-mining-algorithms/>
25. <https://www.javatpoint.com/classification-algorithm-in-machine-learning>
26. <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>
27. <https://scikit-learn.org/stable/modules/tree.html>
28. <https://scikit-learn.org/stable/modules/neighbors.html>
29. <https://www.ibm.com/topics/logistic-regression>
30. <https://machinelearningmastery.com/strong-learners-vs-weak-learners-for-ensemble-learning/>
31. <https://scikit-learn.org/stable/modules/ensemble.html#adaboost>
32. <https://www.freecodecamp.org/news/8-clustering-algorithms-in-machine-learning-that-all-data-scientists-should-know/>
33. https://en.wikipedia.org/wiki/Hierarchical_clustering
34. <https://scikit-learn.org/stable/modules/clustering.html#dbscan>
35. <https://www.javatpoint.com/association-rule-learning>
36. <https://towardsdatascience.com/the-eclat-algorithm-8ae3276d2d17>
37. <https://www.javatpoint.com/fp-growth-algorithm-in-data-mining>
38. <https://sisudata.com/blog/what-is-anomaly-detection-in-data-mining>
39. <https://www.baeldung.com/cs/one-class-svm>
40. Malware detection using data mining techniques - International Journal of Advanced Research in Computer and Communication Engineering
41. <https://www.springboard.com/blog/data-science/data-mining/>

42. <https://www.linkedin.com/pulse/decision-tree-cart-algorithms-mathematics-all-behind-algorithm-patel/>
43. <https://towardsdatascience.com/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-and-spark-f2861df67e3>
44. <https://medium.com/analytics-vidhya/the-math-behind-logistic-regression-c2f04ca27bca>
45. <https://developers.google.com/machine-learning/crash-course/classification/accuracy>
46. <https://en.wikipedia.org/wiki/F-score>
47. <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>
48. https://en.wikipedia.org/wiki/Receiver_operating_characteristic
49. Urcuqui, C., & Navarro, A. (2016, April). Machine learning classifiers for android malware analysis. In Communications and Computing (COLCOM), 2016 IEEE Colombian Conference on (pp. 1-6). IEEE.
50. López, C. C. U., Villarreal, J. S. D., Belalcazar, A. F. P., Cadavid, A. N., & Cely, J. G. D. (2018, May). Features to Detect Android Malware. In 2018 IEEE Colombian Conference on Communications and Computing (COLCOM) (pp. 1-6). IEEE.
51. Cao, D., Wang, S., Li, Q., Cheny, Z., Yan, Q., Peng, L., & Yang, B. (2016, August). DroidCollector: A High Performance Framework for High Quality Android Traffic Collection. In Trustcom/BigDataSE/I SPA, 2016 IEEE (pp. 1753-1758). IEEE
52. <https://www.displayr.com/what-is-a-correlation-matrix/>
53. <https://towardsdatascience.com/ensemble-methods-comparing-scikit-learns-voting-classifier-to-the-stacking-classifier-f5ab1ed1a29d>
54. <https://www.kaggle.com/code/vikashchand28/all-model-for-50-new>
55. <https://www.kaggle.com/code/xwolf12/android-malware-analysis>