Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Порівняльний аналіз алгоритмів машинного навчання
для виявлення фішингових вебресурсів
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-2
Максим ЛЯХОВ
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник ас. Ірина МАЛЄЄВА
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Лариса ЧАЛА
(підпис) (власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Ляхову Максиму Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Порівняльний аналіз алгоритмів машинного навчання для виявлення фішингових вебресурсів

затверджена наказом університету від 19 травня 20 25 р. № 378Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 14 липня 20 26 р.

3. Вихідні дані до роботи наукові статті з галузі машинного навчання та інформаційної безпеки, присвячені методам виявлення фішингових вебресурсів, документація бібліотек Python для аналізу даних і побудови моделей машинного навчання, а також набір даних Book-My-Show для задачі класифікації фішингових та легітимних сайтів.

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі
2) Теоретичні дослідження
3) Експериментальні дослідження

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Визначення проблеми	20.05.2025	виконано
3	Аналіз теоретичної основи	21.05.2025	виконано
4	Підготовка до проведення експериментів	29.05.2025	виконано
5	Експериментальні дослідження	01.06.2025	виконано
6	Оформлення пояснювальної записки	10.06.2025	виконано
7	Захист	14.07.2026	

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Ірина Малєєва
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 94 с., 35 рис., 1 табл., 1 дод., 20 джерел.

ВИЯВЛЕННЯ ФІШИНГУ, ГРАДІЄНТНИЙ БУСТИНГ, КЛАСИФІКАЦІЯ URL, МАШИННЕ НАВЧАННЯ, ФІШИНГОВИЙ ВЕБРЕСУРС, K-NEAREST NEIGHBORS, LOGISTIC REGRESSION, ROC CURVE, XGBOOST.

Об’єктом дослідження є процес автоматизованого виявлення фішингових вебресурсів на основі аналізу характеристик URL-адрес із використанням методів машинного навчання.

Предметом дослідження є ефективність застосування різних алгоритмів машинного навчання для класифікації вебресурсів та визначення їх належності до фішингових або легітимних сайтів.

Метою роботи є проведення порівняльного аналізу алгоритмів машинного навчання для задачі виявлення фішингових вебресурсів та визначення моделі, що забезпечує найвищу точність класифікації.

Методами дослідження є методи аналізу даних, методи машинного навчання для класифікації (Logistic Regression, K-Nearest Neighbors, XGBoost), а також методи оцінювання ефективності моделей, зокрема матриця помилок, ROC-крива та крос-валідація.

У результаті виконання роботи проведено експериментальне дослідження ефективності алгоритмів машинного навчання для класифікації вебресурсів.

Результати роботи можуть бути використані під час розробки систем автоматичного виявлення фішингових вебресурсів та засобів підвищення інформаційної безпеки вебсервісів.

ABSTRACT

Bachelor’s thesis contains: 94 pp., 35 fig., 1 tabl., 1 ann., 20 references.

GRADIENT BOOSTING, K-NEAREST NEIGHBORS, LOGISTIC REGRESSION, MACHINE LEARNING, PHISHING DETECTION, PHISHING WEBSITE, ROC CURVE, URL CLASSIFICATION, XGBOOST.

The object of the research is the process of automated detection of phishing web resources based on the analysis of URL characteristics using machine learning methods.

The subject of the research is the effectiveness of applying different machine learning algorithms for the classification of web resources and determining their belonging to phishing or legitimate websites.

The aim of the work is to conduct a comparative analysis of machine learning algorithms for the task of phishing website detection and to determine the model that provides the highest classification accuracy.

The research methods include data analysis methods, machine learning classification algorithms (Logistic Regression, K-Nearest Neighbors, XGBoost), as well as model evaluation methods such as the confusion matrix, ROC curve, and cross-validation.

As a result of this work, an experimental study was conducted to evaluate the effectiveness of machine learning algorithms for classifying web resources.

The results of the work can be used in the development of automated phishing website detection systems and tools for improving the information security of web services.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 9
Вступ 10
1 Аналіз предметної галузі та постановка задачі 12
1.1 Фішинг як сучасна кіберзагроза 12
1.1.1 Поняття фішингу та його основні ознаки 13
1.1.2 Типи фішингових атак та способи їх реалізації 14
1.1.3 Наслідки фішингових атак для користувачів та організацій 16
1.2 Особливості фішингових вебресурсів 17
1.2.1 Характерні ознаки фішингових URL-адрес 18
1.2.2 Структурні та поведінкові характеристики фішингових сайтів 20
1.2.3 Методи маскування фішингових вебресурсів 21
1.3 Методи виявлення фішингових вебресурсів 21
1.3.1 Традиційні підходи до виявлення фішингових сайтів 23
1.3.2 Використання чорних списків та евристичних методів 23
1.3.3 Обмеження класичних підходів 24
1.4 Використання машинного навчання для виявлення фішингових вебресурсів 24
1.4.1 Основні принципи машинного навчання 26
1.4.2 Типи задач машинного навчання у задачах кібербезпеки 27
1.4.3 Класифікація як підхід до виявлення фішингових URL 29
1.5 Постановка задачі 30
2 Теоретичні дослідження 32
2.1 Основи машинного навчання у задачах класифікації 32
2.1.1 Поняття задачі класифікації у машинному навчанні 33
2.1.2 Бінарна класифікація у задачах кібербезпеки 35
2.1.3 Формування ознак для класифікації вебресурсів 36
2.2 Підготовка даних для навчання моделей 37
2.2.1 Попередня обробка даних 38

2.2.2 Відбір та аналіз ознак 39
2.2.3 Розподіл даних на навчальну та тестову вибірки 41
2.3 Алгоритм Logistic Regression 42
2.3.1 Математична модель логістичної регресії 43
2.3.2 Функція втрат та оптимізація параметрів 44
2.3.3 Особливості застосування алгоритму для класифікації 45
2.4 Алгоритм K-Nearest Neighbors 46
2.4.1 Принцип роботи методу найближчих сусідів 48
2.4.2 Метрики відстані у задачах класифікації 49
2.4.3 Переваги та недоліки алгоритму 50
2.5 Алгоритм XGBoost 51
2.5.1 Основні принципи градієнтного бустингу 52
2.5.2 Архітектура алгоритму XGBoost 54
2.5.3 Особливості використання XGBoost у задачах класифікації 55
2.6 Методи оцінювання ефективності моделей машинного навчання 56
2.6.1 Матриця помилок класифікації 56
2.6.2 Показники точності класифікації 57
2.6.3 ROC-крива та AUC 57
3 Експериментальні дослідження 59
3.1 План експериментів 59
3.1.1 Постановка експериментальної задачі 60
3.1.2 Загальна схема експериментального дослідження 61
3.1.3 Обрані алгоритми машинного навчання для порівняння 62
3.2 Опис використаних наборів даних 63
3.2.1 Загальна характеристика набору даних 64
3.2.2 Структура та ознаки набору даних 65
3.2.3 Попередня обробка даних 68
3.3 Реалізація експериментального середовища 71
3.3.1 Використані програмні бібліотеки та інструменти 71
3.3.2 Розподіл даних на навчальну та тестову вибірки 73

3.3.3 Побудова моделей машинного навчання 73
3.4 Результати дослідження 76
3.4.1 Результати класифікації для моделі Logistic Regression 76
3.4.2 Результати класифікації для моделі K-Nearest Neighbors 79
3.4.3 Результати класифікації для моделі XGBoost 81
3.4.4 Порівняльний аналіз результатів моделей 84
Висновки 90
Перелік джерел посилання 92
Додаток А Відомість кваліфікаційної роботи 94

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Accuracy – метрика оцінювання якості моделі, що визначає частку правильно класифікованих об’єктів серед усіх прогнозів;

AUC – Area Under Curve – показник якості класифікації, що визначає площу під ROC-кривою та характеризує здатність моделі розрізняти класи;

GridSearchCV – метод підбору гіперпараметрів моделі машинного навчання шляхом перебору всіх можливих комбінацій параметрів із використанням крос-валідації;

HTTP – Hypertext Transfer Protocol – протокол передачі гіпертексту;

K-Fold – метод крос-валідації, при якому набір даних поділяється на k частин для багаторазового навчання та тестування моделі;

KNN – K-Nearest Neighbors – алгоритм класифікації, що визначає клас об’єкта на основі найближчих сусідів у просторі ознак;

Logistic Regression – алгоритм машинного навчання для задач бінарної класифікації, що оцінює ймовірність належності об’єкта до певного класу;

Machine Learning – напрям штучного інтелекту, що досліджує методи створення алгоритмів, здатних навчатися на основі даних;

ROC-крива – графічний метод оцінювання ефективності класифікаційної моделі, що відображає залежність між часткою істинно позитивних та хибно позитивних результатів;

StratifiedKFold – модифікований метод крос-валідації, що зберігає пропорції класів у кожній підвибірці набору даних;

URL – Uniform Resource Locator – уніфікований покажчик ресурсу, що використовується для ідентифікації та доступу до вебресурсів у мережі Інтернет;

XGBoost – алгоритм ансамблевого машинного навчання, заснований на градієнтному бустингу дерев рішень, що забезпечує високу точність класифікації.

ВСТУП

У сучасному інформаційному суспільстві мережа Інтернет стала невід'ємною частиною повсякденного життя людини. Інтернет використовується для здійснення фінансових операцій, доступу до електронних сервісів, дистанційної роботи, електронного навчання, комунікації та зберігання персональних даних. Разом із розвитком цифрових технологій стрімко зростає і кількість кіберзагроз, які становлять значну небезпеку як для окремих користувачів, так і для організацій та державних структур. Однією з найбільш поширених та небезпечних форм кіберзлочинності є фішинг.

Фішинг являє собою різновид шахрайства у мережі Інтернет, який спрямований на отримання конфіденційної інформації користувачів шляхом введення їх в оману. Зловмисники створюють підроблені вебресурси, що імітують офіційні сайти банків, платіжних систем, соціальних мереж або інших популярних онлайн-сервісів. Такі вебсторінки візуально майже не відрізняються від оригінальних ресурсів, що ускладнює їх розпізнавання звичайними користувачами. Після переходу на подібні сторінки користувач може ввести логін, пароль, банківські реквізити або іншу персональну інформацію, яка потрапляє до зловмисників. У результаті цього можливі фінансові втрати, витік персональних даних або несанкціонований доступ до облікових записів.

З кожним роком кількість фішингових атак зростає, а методи їх реалізації стають дедалі складнішими. Сучасні фішингові вебресурси часто використовують складні механізми маскування, схожі доменні імена, використання HTTPS-протоколу, а також автоматизовані інструменти створення підроблених вебсторінок. У зв'язку з цим традиційні методи виявлення фішингових сайтів, які базуються на чорних списках або простих евристичних правилах, не завжди забезпечують достатній рівень ефективності.

У сучасних умовах одним із перспективних напрямів розв’язання проблеми виявлення фішингових вебресурсів є використання методів машинного навчання. Машинне навчання є важливою складовою штучного інтелекту і передбачає створення моделей, здатних автоматично навчатися на основі даних та виявляти приховані закономірності. На відміну від традиційних підходів, алгоритми машинного навчання дозволяють аналізувати велику кількість параметрів вебресурсів і на основі отриманих характеристик визначати, чи є вебадреса потенційно фішинговою.

Для задачі виявлення фішингових вебресурсів можуть використовуватися різноманітні характеристики вебадрес та вебсторінок. До таких характеристик належать структура URL-адреси, довжина доменного імені, наявність спеціальних символів, використання IP-адреси замість доменного імені, особливості HTML-коду сторінки, наявність зовнішніх посилань, а також інші параметри, які можуть свідчити про потенційно небезпечний характер ресурсу. Аналіз таких ознак за допомогою алгоритмів машинного навчання дозволяє автоматизувати процес класифікації вебресурсів і підвищити ефективність систем кібербезпеки.

У сучасних дослідженнях застосовується значна кількість алгоритмів машинного навчання для розв’язання задач класифікації вебресурсів. Серед найбільш поширених підходів можна виділити логістичну регресію, алгоритм k найближчих сусідів, дерева рішень, методи ансамблевого навчання та алгоритми градієнтного бустингу. Кожен із цих методів має власні особливості, переваги та обмеження. Ефективність їх застосування значною мірою залежить від структури набору даних, кількості ознак та способів попередньої обробки інформації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Фішинг як сучасна кіберзагроза

Фішинг (рисунок 1.1) є одним із найбільш поширених видів кіберзлочинності у сучасному цифровому середовищі. Сутність цього виду атак полягає у введенні користувачів в оману з метою отримання конфіденційної інформації, зокрема логінів, паролів, банківських реквізитів або інших персональних даних. Як правило, зловмисники створюють підроблені вебресурси [1] або надсилають повідомлення, що імітують офіційні сервіси, банки чи інші довірені організації, спонукаючи користувачів добровільно передати свої дані.

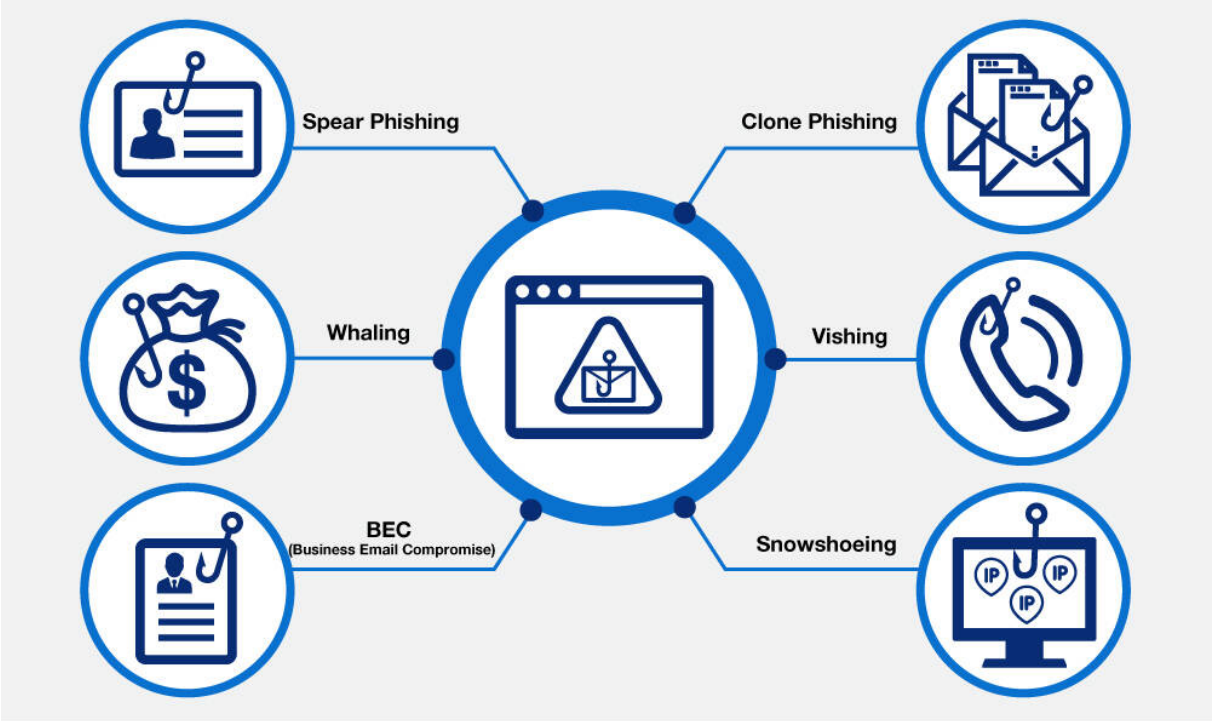

Рисунок 1.1 – Основні типи фішингових атак у сучасному кіберпросторі

Особливістю фішингових атак є використання методів соціальної інженерії, які дозволяють маніпулювати поведінкою користувачів.

Атакуючі можуть використовувати підроблені електронні листи, повідомлення у соціальних мережах, SMS або навіть телефонні дзвінки. У більшості випадків такі повідомлення створюють відчуття терміновості або небезпеки, змушуючи користувача негайно виконати певну дію, наприклад перейти за посиланням або підтвердити свої облікові дані.

Сучасні фішингові атаки можуть мати різні форми та відрізнятися способом реалізації. Серед найпоширеніших різновидів виділяють spear phishing, clone phishing, vishing, whaling та інші методи, що орієнтовані як на окремих користувачів, так і на організації. Кожен з цих типів атак має власні особливості та механізми впливу на жертву (рисунок 1.1), що ускладнює їх своєчасне виявлення та підвищує актуальність розробки ефективних методів протидії.

1.1.1 Поняття фішингу та його основні ознаки

Фішинг є різновидом кіберзлочинної діяльності [1], що базується на використанні методів соціальної інженерії для отримання конфіденційної інформації користувачів. Зловмисники імітують повідомлення або вебресурси відомих організацій, таких як банки, платіжні системи або популярні онлайн-сервіси, з метою змусити користувача добровільно надати свої персональні дані. До таких даних можуть належати логіни, паролі, номери банківських карток, коди підтвердження тощо.

Механізм реалізації (рисунок 1.2) фішингової атаки зазвичай складається з кількох послідовних етапів. Спочатку користувач отримує підроблене повідомлення, яке зовні виглядає як офіційне повідомлення від відомого сервісу. Після цього жертва переходить за надісланим посиланням або встановлює підроблений застосунок, який може вимагати введення персональних даних або надання доступу до певної інформації. У результаті отримані дані передаються зловмисникам, які використовують їх для несанкціонованого доступу до облікових записів або фінансових ресурсів.

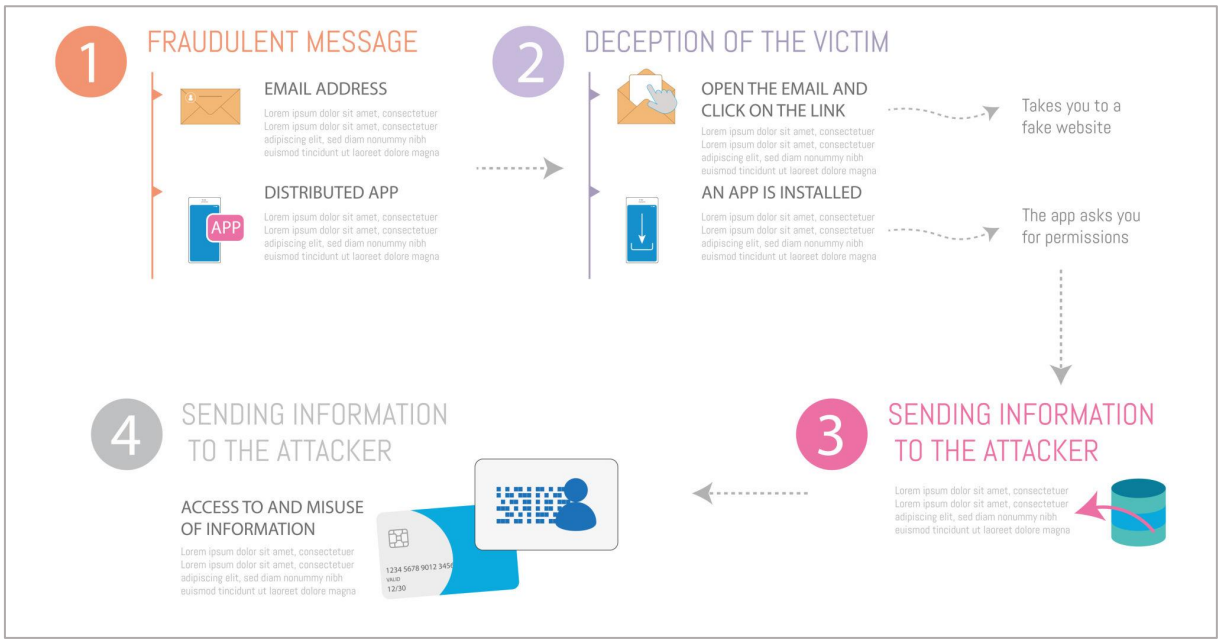

Рисунок 1.2 – Загальна схема реалізації фішингової атаки

Основними ознаками фішингових повідомлень є підозріла адреса відправника, наявність граматичних помилок, використання загрозливих або термінових формулювань, а також прохання підтвердити або оновити особисту інформацію. Крім того, фішингові повідомлення часто містять посилання на підроблені вебсторінки, що імітують дизайн легітимних сервісів. Саме ці характерні ознаки дозволяють виявляти потенційно небезпечні повідомлення та запобігати компрометації персональних даних користувачів.

1.1.2 Типи фішингових атак та способи їх реалізації

Фішингові атаки можуть реалізовуватися у різних формах залежно від способу взаємодії з потенційною жертвою та використаних каналів комунікації. Основною метою таких атак є отримання конфіденційної інформації або несанкціонований доступ до інформаційних систем. Зловмисники можуть використовувати електронну пошту, мобільні повідомлення, телефонні дзвінки, підроблені вебресурси або спеціально

створені шкідливі застосунки для введення користувачів в оману та отримання необхідних даних.

Одним із найбільш поширених (рисунок 1.3) різновидів є spear phishing – цільова фішингова атака, спрямована на конкретну особу або організацію. У цьому випадку зловмисники попередньо збирають інформацію про потенційну жертву, щоб зробити повідомлення максимально переконливим.

Іншим різновидом є whaling [2], що орієнтований на керівників організацій або осіб, які мають доступ до критично важливої інформації. Також широко застосовуються smishing та vishing, які використовують SMS-повідомлення або телефонні дзвінки для введення користувачів в оману.

Рисунок 1.3 – Основні типи фішингових атак та способи їх реалізації

Крім того, існують інші методи реалізації фішингових атак, зокрема baiting та watering hole attacks. У першому випадку зловмисники пропонують користувачам привабливий ресурс або файл, що містить шкідливе програмне забезпечення.

У другому випадку атакуючі компрометують популярні вебресурси, які часто відвідує цільова аудиторія, та використовують їх для поширення шкідливого коду.

1.1.3 Наслідки фішингових атак для користувачів та організацій

Фішингові атаки можуть призводити до значних негативних наслідків як для окремих користувачів, так і для організацій (рисунок 1.4). Основною метою таких атак є отримання конфіденційної інформації, що надалі використовується для фінансового шахрайства, несанкціонованого доступу до облікових записів або поширення шкідливого програмного забезпечення. У результаті успішної атаки користувач може втратити доступ до власних акаунтів, зазнати фінансових збитків або стати жертвою подальших кіберзлочинних дій.

Рисунок 1.4 – Наслідки кіберзлочинних атак для організацій

Для організацій наслідки фішингових атак можуть бути ще більш серйозними. Компрометація облікових записів співробітників може призвести до витоку конфіденційних даних, порушення роботи інформаційних систем та значних фінансових втрат. Крім того, такі

інциденти можуть негативно впливати на репутацію компанії, знижувати рівень довіри клієнтів та партнерів, а також спричиняти юридичні наслідки у випадку порушення вимог щодо захисту персональних даних.

Статистичні дані свідчать про постійне зростання масштабів кіберзагроз та збитків, пов’язаних із фішинговими атаками. Зокрема, спостерігається значне збільшення кількості повідомлень про інциденти кібербезпеки, обсягів викраденої інформації та фінансових втрат організацій.

1.2 Особливості фішингових вебресурсів

Фішингові вебресурси є спеціально створеними сайтами, що імітують легітимні інтернет-ресурси з метою введення користувачів в оману та отримання конфіденційної інформації. Такі сайти можуть зовні практично не відрізнятися від справжніх вебсторінок відомих сервісів, банків або онлайн-платформ. Зловмисники копіюють дизайн, структуру сторінки, логотипи та інші елементи інтерфейсу, щоб користувач не зміг відразу помітити підробку та довірився ресурсу.

Однією з характерних особливостей фішингових вебресурсів є використання підроблених або модифікованих URL-адрес. Зловмисники можуть застосовувати різні прийоми маскування, зокрема використання піддоменів, схожих доменних імен, спеціальних символів або прихованих посилань. Наприклад, адреса може містити назву відомого сервісу у піддомені або використовувати так званий typosquatting, коли домен відрізняється від оригінального лише кількома символами (рисунок 1.5). Крім того, фішингові вебресурси часто використовують методи приховування справжнього посилання, наприклад маскування URL у тексті повідомлення або використання скорочених посилань.

Рисунок 1.5 – Приклади характерних ознак фішингових вебпосилань та методів їх маскування

У деяких випадках атакуючі комбінують декілька методів одночасно, щоб ускладнити виявлення підробленого ресурсу. Саме ці особливості структури вебадрес та інтерфейсу фішингових сайтів є важливими ознаками, що можуть бути використані у подальшому для автоматичного виявлення таких ресурсів.

1.2.1 Характерні ознаки фішингових URL-адрес

URL-адреса [2] є одним із ключових елементів, що дозволяє ідентифікувати вебресурс у мережі Інтернет. Вона містить інформацію про протокол доступу, доменне ім’я, піддомен та інші структурні елементи, які визначають шлях до конкретної вебсторінки.

Структура типової URL-адреси включає протокол передачі даних, домен та піддомен, що разом формують унікальний ідентифікатор ресурсу в мережі (рисунок 1.6).

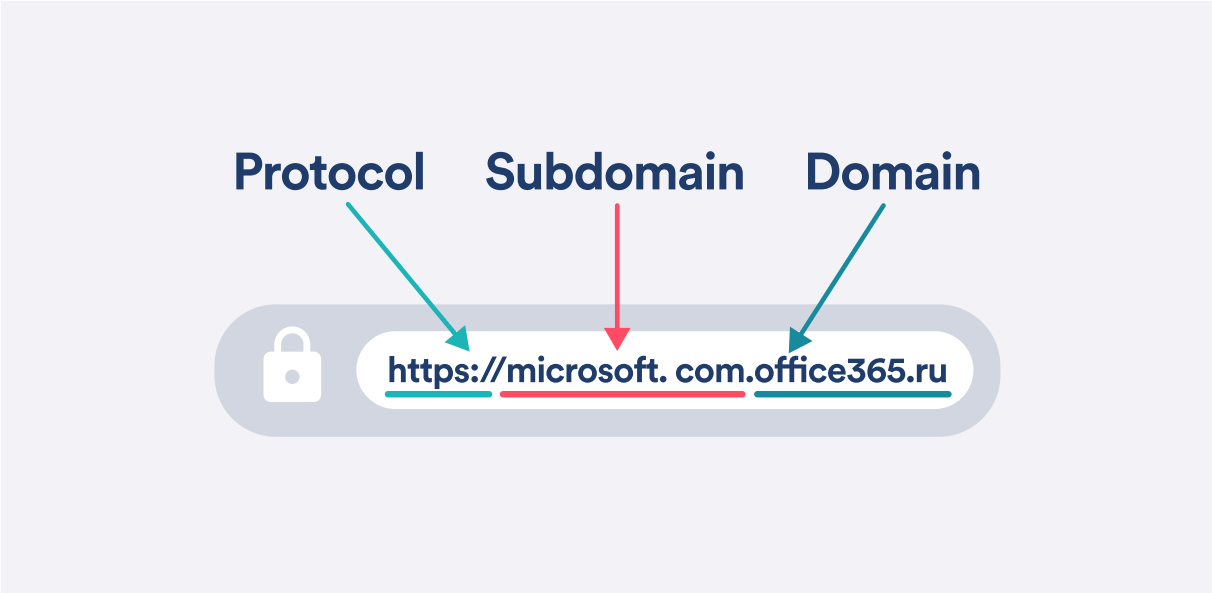

Рисунок 1.6 – Основні компоненти структури URL-адреси вебресурсу

Фішингові вебресурси часто використовують модифіковані або навмисно ускладнені URL-адреси для введення користувачів в оману. Зловмисники можуть застосовувати різні прийоми маскування, наприклад додавання піддоменів (рисунок 1.7), що містять назву відомого сервісу, або використання доменних імен, схожих на легітимні.

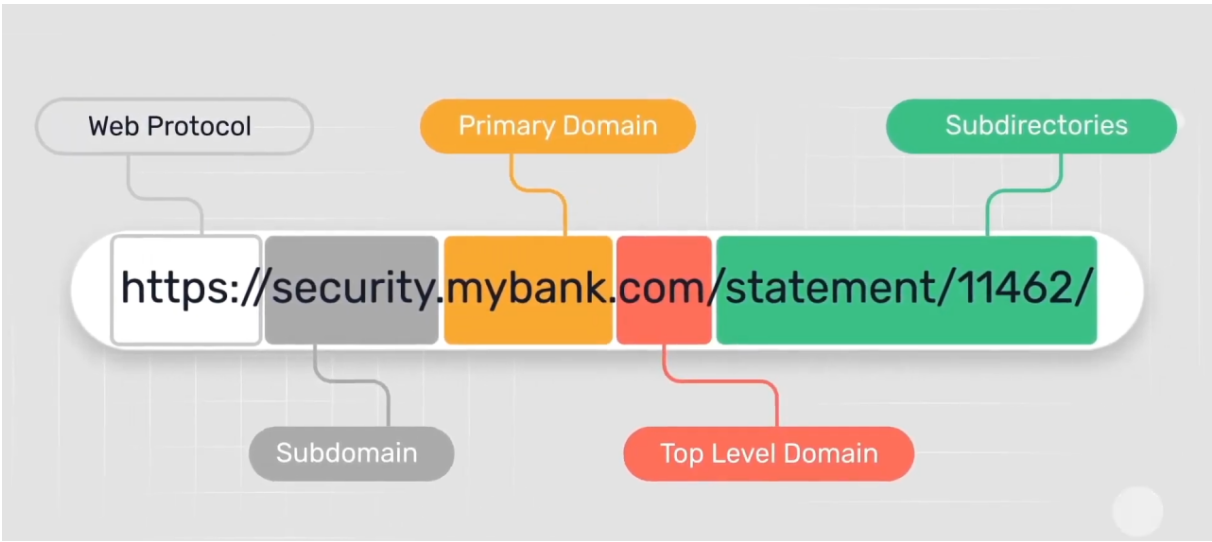

Рисунок 1.7 – Вебадрес з виділенням її ключових елементів

Наприклад, у фішинговій адресі справжній домен може бути розміщений у середині рядка, тоді як фактичний домен сайту знаходиться наприкінці URL, що ускладнює його правильне розпізнавання користувачем. Крім того, фішингові URL-адреси можуть містити зайві символи, довгі послідовності підкаталогів або використання доменів верхнього рівня, які відрізняються від оригінальних. Інколи застосовується так званий typosquatting, коли домен відрізняється від легітимного лише однією або кількома літерами.

1.2.2 Структурні та поведінкові характеристики фішингових сайтів

Фішингові вебресурси часто мають характерні структурні особливості, які дозволяють відрізнити їх від легітимних сайтів. Як правило, такі сторінки копіюють зовнішній вигляд офіційних вебресурсів відомих організацій, включаючи дизайн інтерфейсу, логотипи, кольорову гаму та розташування елементів навігації. Основною метою цього є створення у користувача відчуття довіри до вебсторінки. При цьому на фішингових сайтах часто присутні форми введення конфіденційної інформації, наприклад логінів, паролів або реквізитів банківських карток, які передаються безпосередньо зловмисникам.

Окрім структурних особливостей, фішингові вебресурси характеризуються певними поведінковими ознаками. До них належать автоматичні перенаправлення користувачів на інші сторінки, використання скриптів для збору введених даних, а також спроби приховати справжню адресу сайту. У деяких випадках такі вебресурси можуть обмежувати функціональність сторінки або відображати лише окремі елементи інтерфейсу, необхідні для введення персональної інформації. Виявлення подібних структурних та поведінкових характеристик є важливим етапом аналізу вебресурсів.

1.2.3 Методи маскування фішингових вебресурсів

Зловмисники активно застосовують різноманітні методи маскування фішингових вебресурсів для того, щоб ускладнити їх виявлення та підвищити ймовірність успішного введення користувачів в оману. Одним із поширених способів є використання доменних імен, які візуально схожі на адреси легітимних сайтів. Такий підхід відомий як typosquatting і полягає у створенні доменів, що відрізняються від оригінального лише однією або кількома літерами. Крім того, зловмисники можуть використовувати піддомени, які містять назви відомих сервісів або компаній, щоб створити у користувача помилкове враження про справжність ресурсу.

Іншим поширеним методом є приховування справжньої адреси вебресурсу за допомогою різних технічних прийомів. До них належать використання скорочених URL-адрес, маскування посилань у тексті повідомлень, застосування редиректів або спеціальних символів у вебадресах. Також можуть використовуватися так звані homograph-атаки, при яких у доменному імені застосовуються символи, зовні схожі на літери латинського алфавіту. Використання таких методів дозволяє фішинговим вебресурсам залишатися непоміченими протягом певного часу та значно ускладнює їх автоматичне виявлення.

1.3 Методи виявлення фішингових вебресурсів

Виявлення фішингових вебресурсів є важливим завданням у сфері кібербезпеки, оскільки своєчасне визначення таких сайтів дозволяє запобігти витоку конфіденційної інформації та фінансовим втратам користувачів. Існує кілька підходів до вирішення цієї задачі, які відрізняються за принципами аналізу та рівнем автоматизації. Загалом методи виявлення фішингових ресурсів можна поділити на підходи, що базуються на підвищенні обізнаності користувачів, а також на програмні

методи аналізу вебресурсів, які застосовують різні алгоритми обробки даних (рисунок 1.8).

Програмні методи виявлення фішингових сайтів можуть включати використання списків відомих шкідливих ресурсів, евристичних правил або алгоритмів машинного навчання. Методи на основі чорних та білих списків дозволяють перевіряти вебадреси шляхом порівняння їх із базами даних відомих фішингових сайтів. Евристичні підходи аналізують різні характеристики вебресурсу, зокрема структуру URL-адреси, вміст сторінки, використання скриптів або подібність до легітимних сайтів.

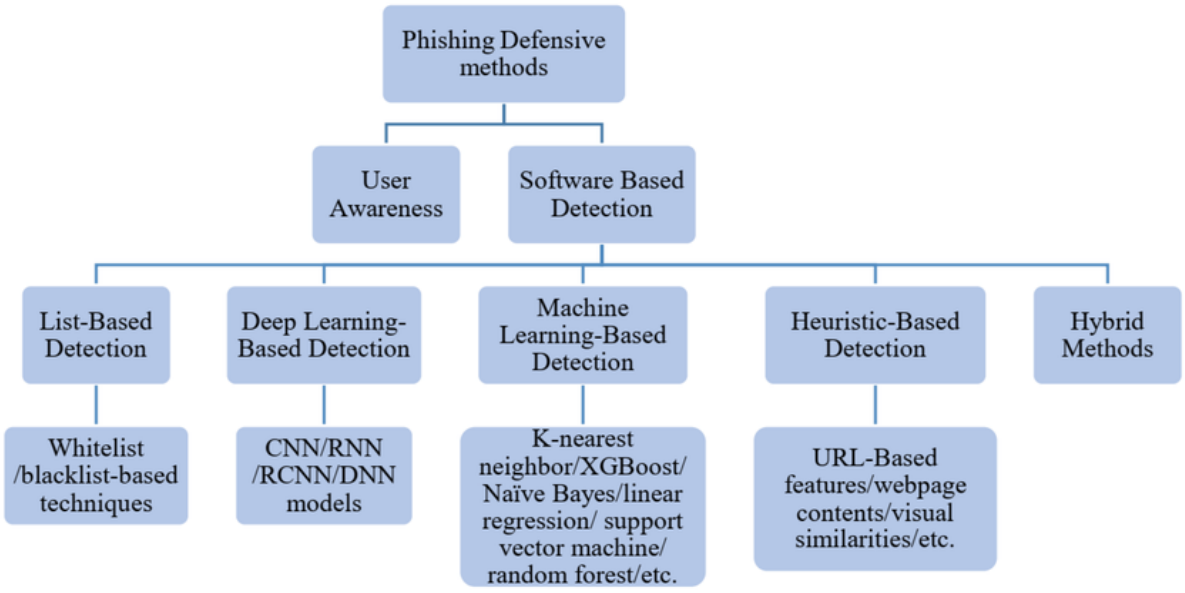

Рисунок 1.8 – Класифікація основних методів виявлення фішингових вебресурсів

У сучасних системах кіберзахисту все частіше застосовуються методи машинного навчання та глибокого навчання, які дозволяють автоматично виявляти складні закономірності у даних. Такі алгоритми здатні аналізувати велику кількість характеристик вебресурсів і визначати ймовірність того, що сайт є фішинговим.

1.3.1 Традиційні підходи до виявлення фішингових сайтів

Традиційні підходи до виявлення фішингових вебресурсів базуються на аналізі відомих ознак шкідливих сайтів та використанні наперед визначених правил перевірки. Такі методи ґрунтуються на аналізі структури URL-адреси, доменного імені, вмісту вебсторінки або інших технічних характеристик ресурсу. Зокрема, система може перевіряти наявність підозрілих елементів у вебадресі, використання незвичних доменів або відсутність захищеного протоколу передачі даних.

Ще одним поширеним підходом є використання сигнатурного аналізу, який передбачає порівняння вебресурсу з базою відомих фішингових сайтів або характерних шаблонів атак. Якщо система виявляє відповідність між перевірюваним ресурсом та збереженими сигнатурами, такий сайт може бути класифікований як небезпечний. Незважаючи на відносну простоту реалізації, традиційні підходи довгий час використовувалися у різних системах захисту інформації та залишаються базовим рівнем фільтрації підозрілих вебресурсів.

1.3.2 Використання чорних списків та евристичних методів

Одним із найбільш поширених методів виявлення фішингових вебресурсів є використання чорних списків, які містять бази даних відомих шкідливих або підозрілих вебадрес. Під час звернення користувача до вебресурсу система перевіряє URL-адресу за допомогою таких баз даних. Якщо адреса вже була раніше ідентифікована як фішингова, доступ до неї може бути заблокований або користувач отримує відповідне попередження про можливу небезпеку.

Окрім чорних списків, широко застосовуються евристичні методи аналізу вебресурсів. Ці підходи базуються на визначенні характерних ознак фішингових сайтів, таких як надмірна довжина URL-адреси, використання

піддоменів, схожість доменного імені з популярними сервісами або наявність підозрілих елементів у коді сторінки. Евристичний аналіз дозволяє виявляти потенційно небезпечні ресурси навіть у тих випадках, коли вони ще не були внесені до бази відомих фішингових сайтів.

1.3.3 Обмеження класичних підходів

Незважаючи на широке застосування традиційних методів виявлення фішингових вебресурсів, вони мають ряд суттєвих обмежень. Зокрема, методи, що базуються на використанні чорних списків, ефективні лише для виявлення вже відомих фішингових сайтів. Нові або нещодавно створені ресурси можуть залишатися поза межами таких баз даних протягом певного часу, що дозволяє зловмисникам успішно використовувати їх для проведення атак.

Крім того, евристичні методи часто залежать від заздалегідь визначених правил, які не завжди здатні врахувати складні або нові схеми фішингових атак. Це може призводити до появи як хибнопозитивних, так і хибнонегативних результатів під час перевірки вебресурсів. У зв’язку з цим виникає необхідність використання більш сучасних підходів, зокрема алгоритмів машинного навчання, які здатні аналізувати велику кількість характеристик вебресурсів та автоматично визначати закономірності, притаманні фішинговим сайтам.

1.4 Використання машинного навчання для виявлення фішингових вебресурсів

Методи машинного навчання [3] відіграють важливу роль у сучасних системах виявлення фішингових вебресурсів. На відміну від традиційних підходів, що базуються на статичних правилах або базах даних відомих шкідливих сайтів, алгоритми машинного навчання здатні автоматично

аналізувати великі обсяги даних та виявляти складні закономірності, характерні для фішингових ресурсів. Це дозволяє підвищити ефективність виявлення нових та раніше невідомих фішингових вебсайтів.

Загальний процес (рисунок 1.9) застосування машинного навчання для виявлення фішингових сайтів включає декілька етапів. Спочатку здійснюється збір даних та формування набору характеристик вебресурсу, таких як структура URL-адреси, особливості гіперпосилань та текстовий вміст сторінки.

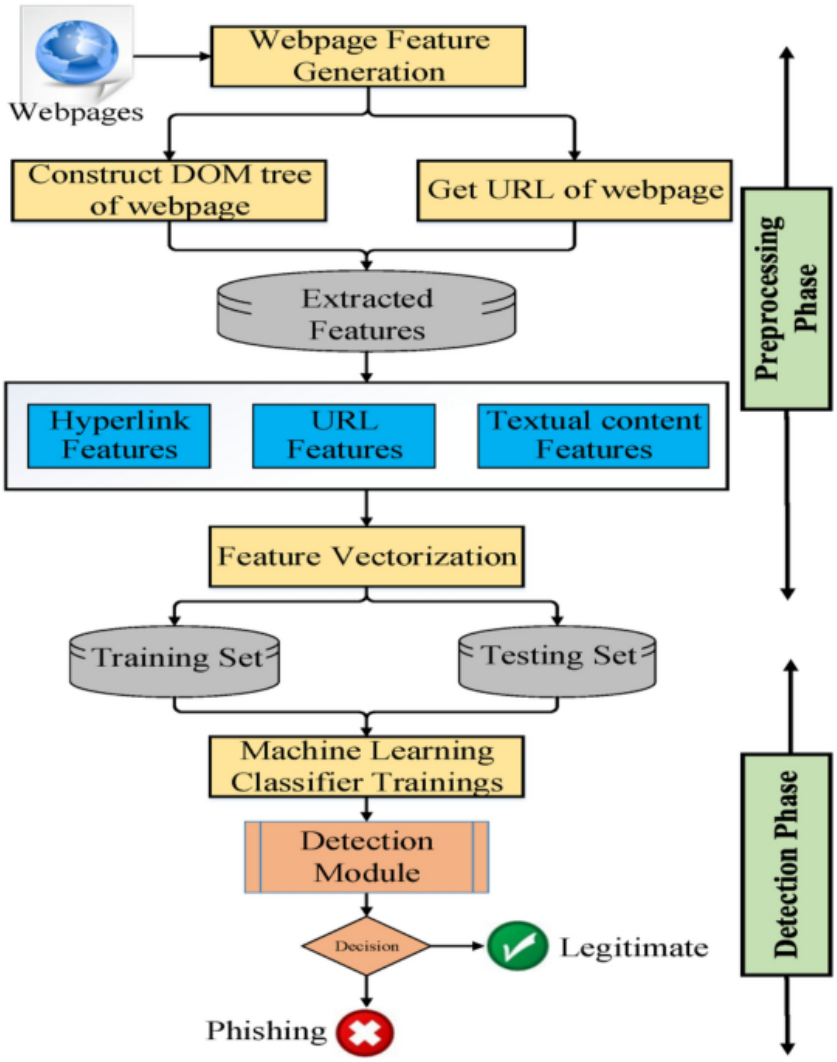

Рисунок 1.9 – Загальна схема використання машинного навчання для виявлення фішингових вебресурсів

Після цього виконується перетворення отриманих характеристик у числовий формат, що дозволяє використовувати їх для навчання моделей класифікації. Після формування навчального набору даних модель машинного навчання проходить етап навчання, у процесі якого вона вчиться відрізняти легітимні вебресурси від фішингових. Для цього використовуються різні алгоритми класифікації, здатні аналізувати значну кількість характеристик вебсторінок. У результаті навчена модель може автоматично визначати тип вебресурсу та класифікувати його як безпечний або фішинговий, що дозволяє значно підвищити ефективність систем захисту від кіберзагроз.

1.4.1 Основні принципи машинного навчання

Машинне навчання є одним із ключових напрямів сучасного розвитку штучного інтелекту, що дозволяє створювати системи, здатні автоматично виявляти закономірності у великих обсягах даних. Основною ідеєю машинного навчання є побудова моделей, які можуть навчатися на основі попереднього досвіду та використовувати отримані знання для прогнозування або класифікації нових даних. У задачах кібербезпеки такі моделі застосовуються для аналізу характеристик вебресурсів та визначення ймовірності того, що сайт є фішинговим.

Процес машинного навчання (рисунок 1.10) зазвичай складається з кількох послідовних етапів. Спочатку здійснюється збір та підготовка даних, які використовуються для навчання моделі. На цьому етапі дані очищуються, структуруються та перетворюються у формат, придатний для подальшого аналізу. Далі відбувається вибір алгоритму машинного навчання та навчання моделі, під час якого вона аналізує навчальні приклади та формує внутрішні закономірності між характеристиками даних.

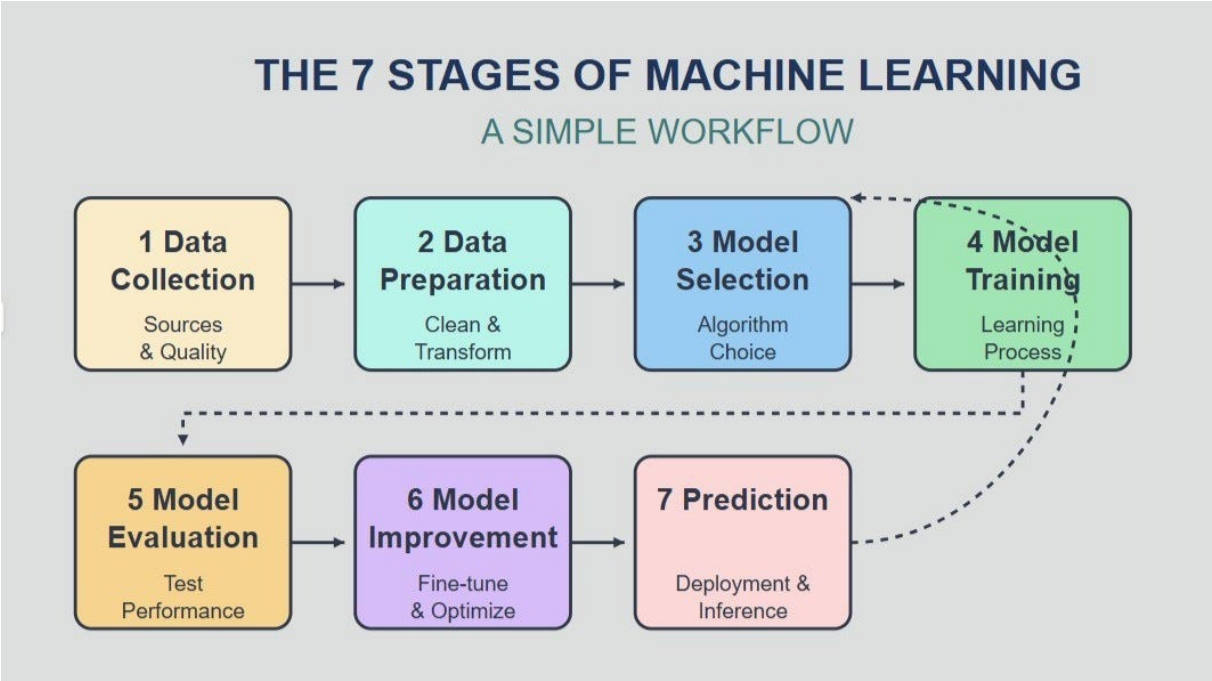

Рисунок 1.10 – Основні етапи процесу машинного навчання

Після завершення навчання модель проходить етап оцінювання та оптимізації. На цьому етапі перевіряється точність роботи алгоритму на тестових даних, що дозволяє оцінити його здатність коректно класифікувати нові приклади. У разі необхідності параметри моделі можуть бути скориговані для підвищення її ефективності. Завершальним етапом є використання навченої моделі для прогнозування або класифікації нових даних, що дозволяє застосовувати машинне навчання у практичних системах виявлення фішингових вебресурсів.

1.4.2 Типи задач машинного навчання у задачах кібербезпеки

У задачах кібербезпеки машинне навчання використовується для автоматичного аналізу великих обсягів даних та виявлення потенційних загроз. Одним із важливих аспектів застосування машинного навчання є класифікація типів задач, які вирішуються за допомогою відповідних алгоритмів (рисунок 1.11).

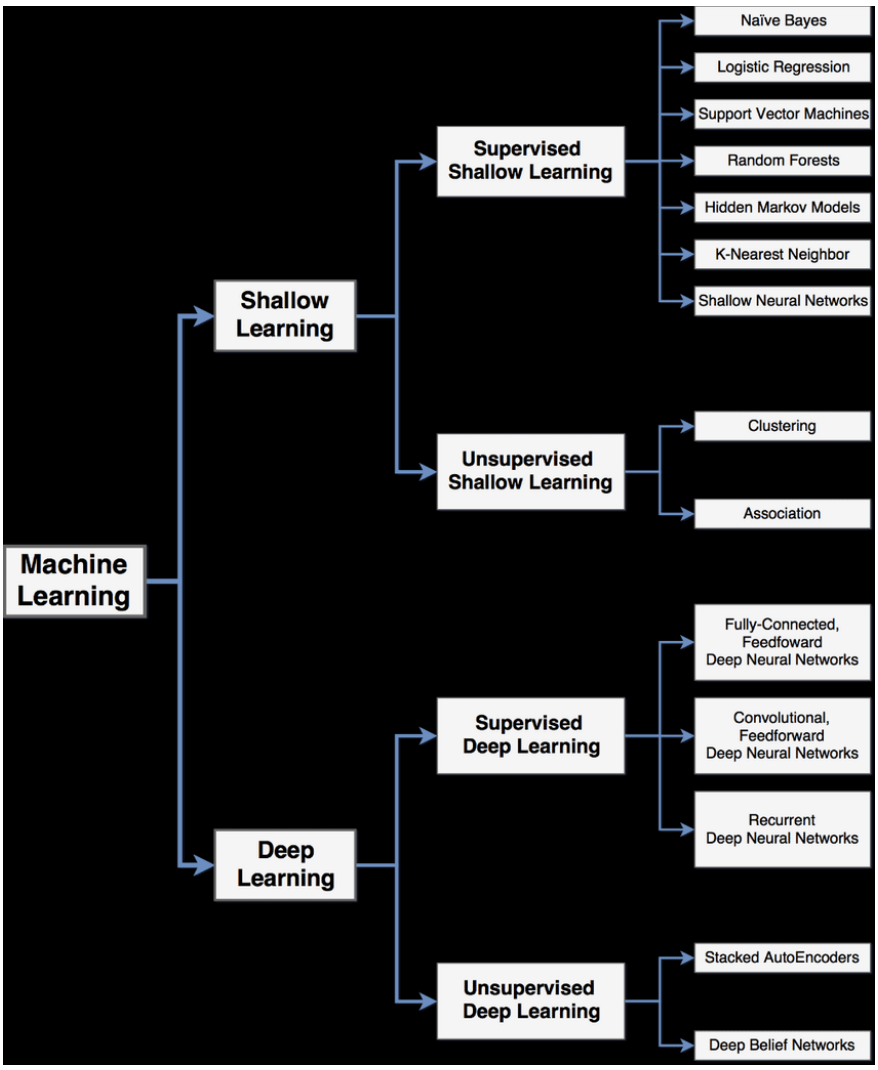

Рисунок 1.11 – Класифікація основних типів задач та алгоритмів машинного навчання

Основними напрямами машинного навчання є навчання з учителем, навчання без учителя та глибоке навчання, кожен з яких має власні особливості та сфери застосування.

Навчання з учителем передбачає використання заздалегідь розмічених даних, де кожному прикладу відповідає відомий результат. У цьому випадку алгоритм навчається встановлювати залежність між вхідними характеристиками та правильним класом. До найбільш поширених алгоритмів цієї групи належать логістична регресія, метод опорних векторів, наївний байєсівський класифікатор, випадковий ліс та

метод k-найближчих сусідів. Саме ці алгоритми часто застосовуються для задач класифікації вебресурсів на легітимні та фішингові.

Навчання без учителя використовується у випадках, коли дані не мають попереднього маркування. У таких задачах алгоритми намагаються самостійно виявити структуру або приховані закономірності у даних, наприклад шляхом кластеризації або пошуку асоціативних залежностей. Окремим напрямом розвитку машинного навчання є глибоке навчання, яке базується на використанні багатошарових нейронних мереж і дозволяє аналізувати складні типи даних. У сфері кібербезпеки ці методи можуть застосовуватися для виявлення складних або раніше невідомих типів атак.

1.4.3 Класифікація як підхід до виявлення фішингових URL

Класифікація [4] є одним із основних підходів, що використовується у задачах виявлення фішингових вебресурсів. Суть цього підходу полягає у віднесенні вебадрес або вебсторінок до певного класу на основі аналізу їх характеристик. У більшості випадків задача зводиться до бінарної класифікації, де система повинна визначити, чи є URL-адреса легітимною або фішинговою. Для цього використовуються спеціальні алгоритми машинного навчання, здатні аналізувати набір ознак вебресурсу та приймати відповідне рішення.

Процес класифікації (рисунок 1.12) зазвичай включає декілька етапів. Спочатку формується набір характеристик вебресурсу, до яких можуть належати параметри URL-адреси, особливості структури сторінки або характеристики гіперпосилань. Далі виконується відбір найбільш значущих ознак та навчання моделі класифікації на основі підготовленого набору даних. Після завершення етапу навчання модель може використовуватися для аналізу нових, раніше невідомих вебресурсів.

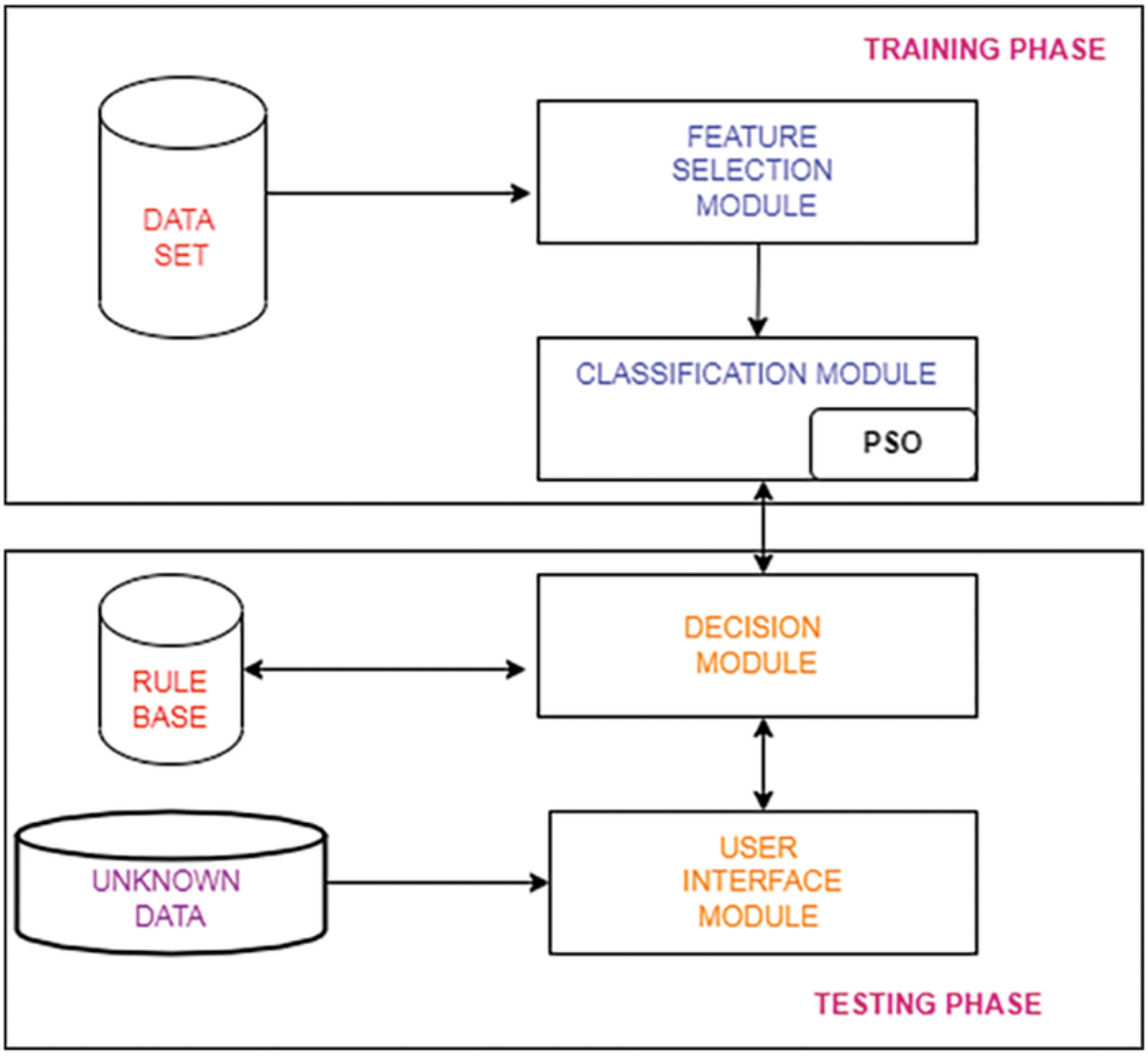

Рисунок 1.12 – Схема класифікаційного підходу до виявлення фішингових вебресурсів

Система отримує дані про URL або вебсторінку, обробляє їх за допомогою навченої моделі та визначає відповідний клас ресурсу.

1.5 Постановка задачі

Метою даної роботи є проведення порівняльного аналізу алгоритмів машинного навчання для виявлення фішингових вебресурсів на основі характеристик URL-адрес та інших ознак вебсторінок. Для цього необхідно дослідити можливість використання алгоритмів класифікації для

автоматичного визначення того, чи належить певний вебресурс до фішингових або легітимних.

У межах роботи планується використати набір даних, що містить характеристики вебадрес та вебсторінок, які можуть бути використані як ознаки для побудови моделей машинного навчання. На основі цих даних необхідно буде виконати попередню обробку та підготовку ознак, сформувати навчальну та тестову вибірки, а також реалізувати навчання моделей класифікації. Для дослідження обрано алгоритми Logistic Regression, K-Nearest Neighbors та XGBoost, які будуть використані для побудови моделей виявлення фішингових URL.

У процесі дослідження необхідно виконати навчання зазначених моделей, провести їх тестування на відповідному наборі даних та оцінити ефективність роботи кожного алгоритму за допомогою показників якості класифікації. За результатами експерименту буде проведено порівняльний аналіз отриманих результатів та визначено алгоритм, який демонструє найкращу ефективність для задачі виявлення фішингових вебресурсів.

2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

2.1 Основи машинного навчання у задачах класифікації

Машинне навчання [5] є одним із ключових напрямів розвитку сучасних інформаційних технологій, що дозволяє автоматично виявляти закономірності у великих обсягах даних та приймати рішення на основі отриманих результатів. У задачах аналізу вебресурсів машинне навчання використовується для визначення того, чи належить певний ресурс до фішингових або легітимних. Загальний процес використання машинного навчання включає декілька послідовних етапів, серед яких збір даних, виділення ознак, навчання моделі, оцінювання її ефективності та використання моделі для прогнозування результатів (рисунок 2.1).

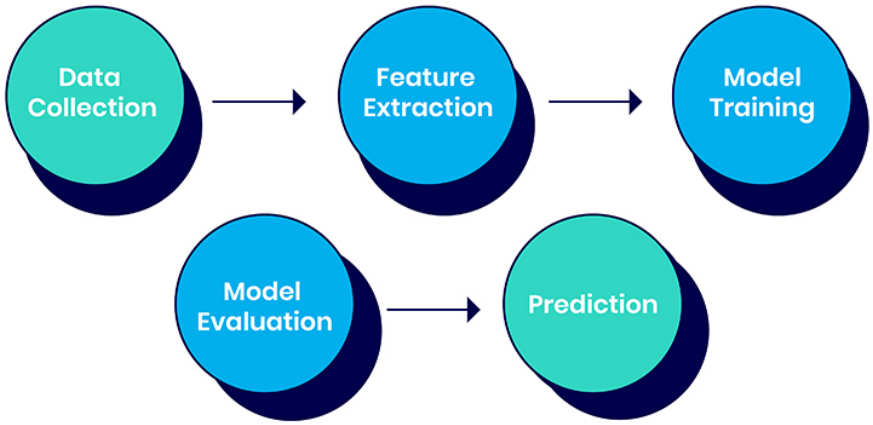

Рисунок 2.1 – Основні етапи процесу машинного навчання

Однією з найбільш поширених задач машинного навчання є задача класифікації. У межах цієї задачі алгоритм отримує набір вхідних характеристик об’єкта та визначає, до якого класу він належить. У задачах кібербезпеки класифікація часто використовується для аналізу вебресурсів, електронних повідомлень або мережевого трафіку з метою виявлення потенційно небезпечних об’єктів. Для цього алгоритми машинного

навчання навчаються на основі попередньо підготовлених наборів даних, які містять приклади об’єктів із відомими класами.

Залежно від кількості можливих результатів класифікації розрізняють декілька типів задач класифікації (рисунок 2.2). Найбільш поширеною є бінарна класифікація, у межах якої об’єкти відносяться до одного з двох можливих класів. Також існують задачі багатокласової класифікації, де об’єкти можуть належати до декількох різних категорій, та багатоміткової класифікації, у якій одному об’єкту може відповідати одразу декілька класів.

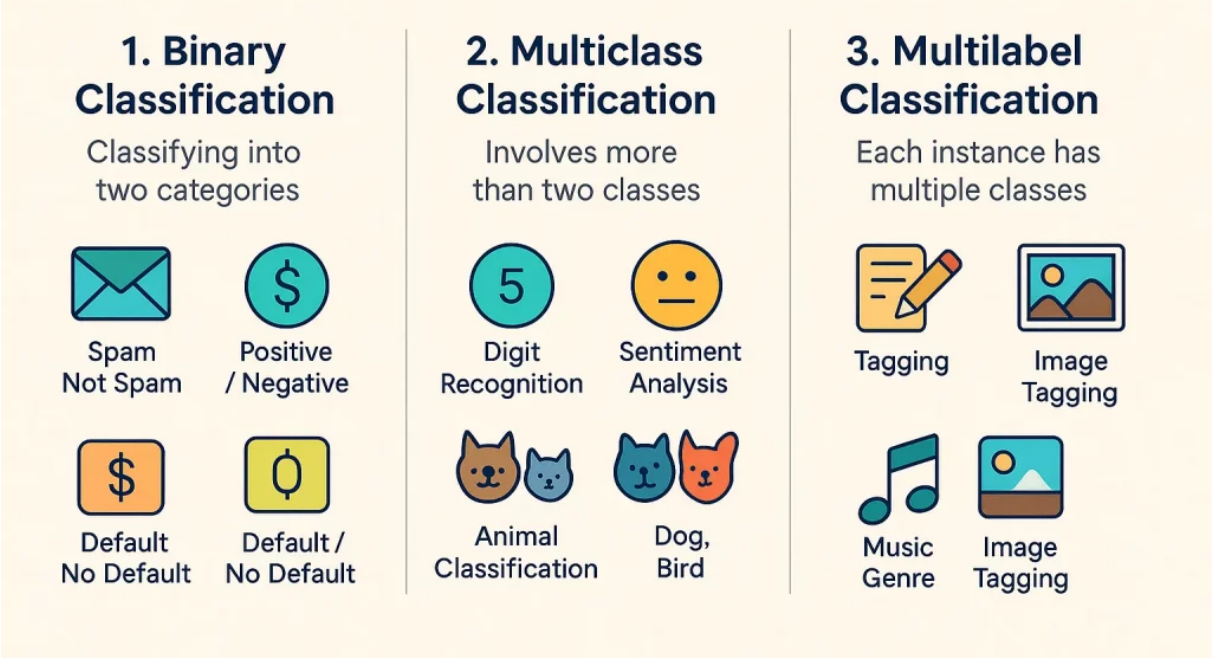

Рисунок 2.2 – Типи задач класифікації у машинному навчанні

2.1.1 Поняття задачі класифікації у машинному навчанні

Задача класифікації [6] є однією з основних задач машинного навчання, що полягає у визначенні належності об’єкта до певного класу на основі його характеристик. У межах цієї задачі алгоритм отримує набір вхідних параметрів, які описують досліджуваний об’єкт, та на основі

попередньо отриманого досвіду визначає категорію, до якої він належить. Такий підхід широко застосовується у різних сферах, зокрема у системах рекомендацій, аналізі текстів, обробці зображень та задачах кібербезпеки.

Процес класифікації (рисунок 2.3) передбачає використання навчального набору даних, що містить приклади об’єктів із відомими класами. Під час навчання модель аналізує зв’язок між характеристиками об’єктів та відповідними класами, формуючи математичну залежність між вхідними даними та результатом. Після завершення навчання модель може використовуватися для класифікації нових об’єктів, для яких клас заздалегідь невідомий.

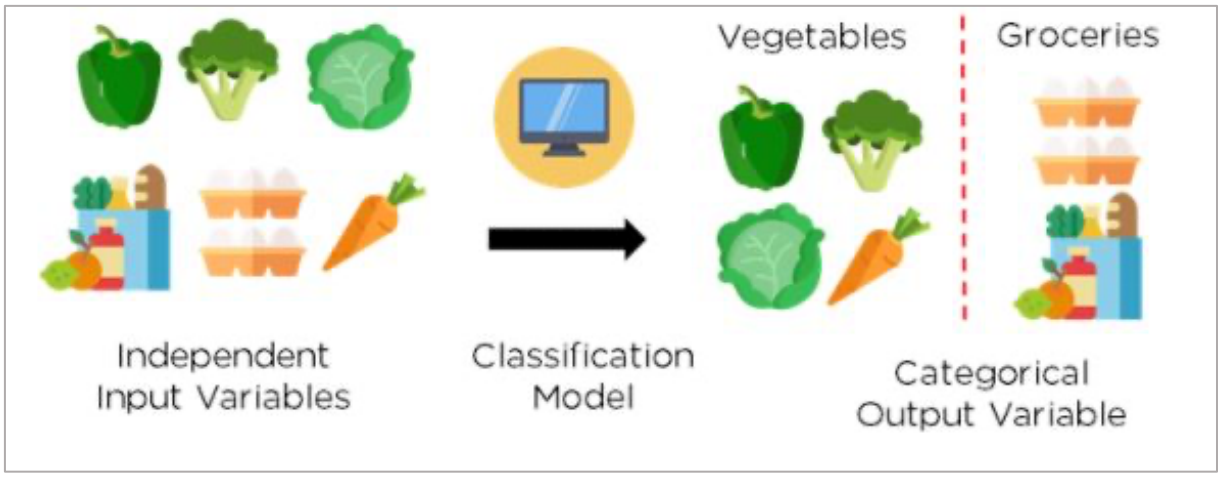

Рисунок 2.3 – Загальна схема роботи моделі класифікації у машинному навчанні

У задачах виявлення фішингових вебресурсів класифікація використовується для визначення того, чи є вебадреса або вебсторінка безпечною або фішинговою. Для цього алгоритм аналізує різні характеристики ресурсу, зокрема параметри URL-адреси, структуру сторінки або інші ознаки. На основі цих характеристик модель приймає рішення щодо належності вебресурсу до одного з визначених класів.

2.1.2 Бінарна класифікація у задачах кібербезпеки

Бінарна класифікація [7] є одним із найпоширеніших підходів у задачах машинного навчання, коли об’єкти розподіляються лише між двома можливими класами. У такій постановці задачі алгоритм аналізує набір характеристик об’єкта та визначає, до якого з двох класів він належить. У сфері кібербезпеки бінарна класифікація широко застосовується для виявлення шкідливих об’єктів, наприклад для визначення того, чи є електронне повідомлення спамом, чи належить файл до шкідливого програмного забезпечення, або чи є вебресурс фішинговим.

У задачі виявлення фішингових вебресурсів бінарна класифікація передбачає віднесення вебадрес або вебсторінок до одного з двох класів: легітимних або фішингових. На основі цих ознак алгоритм будує розділяючу межу між двома класами об’єктів, що дозволяє визначати належність нових вебресурсів до відповідної категорії (рисунок 2.4).

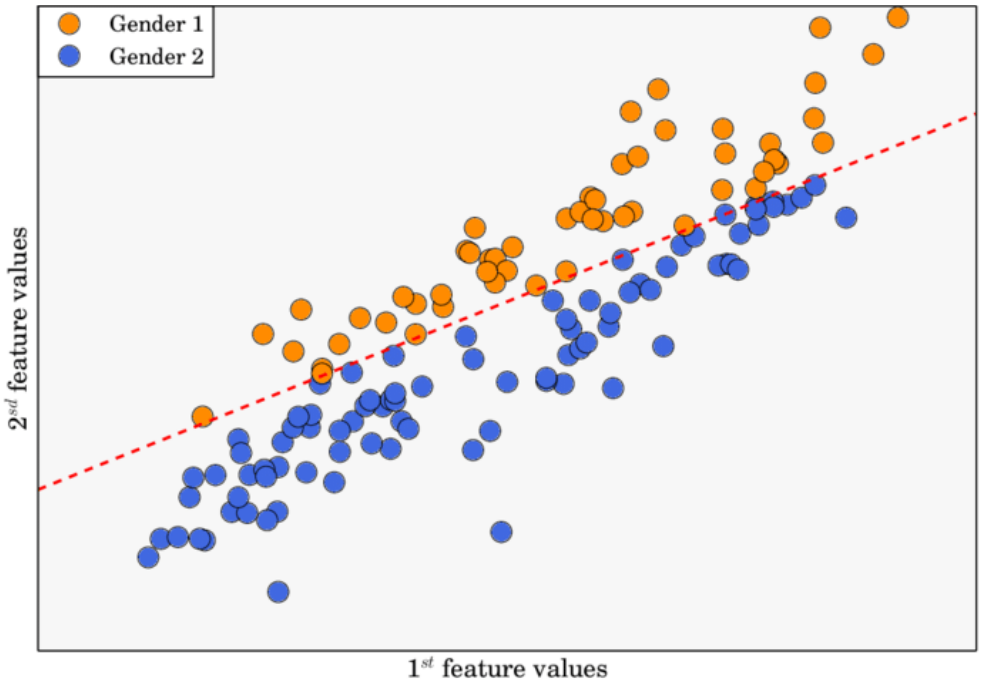

Рисунок 2.4 – Приклад розділення об’єктів у задачі бінарної класифікації

Графічно процес бінарної класифікації можна представити у вигляді розподілу об’єктів у просторі ознак, де кожен клас утворює окрему групу даних. У цьому випадку алгоритм машинного навчання визначає оптимальну межу, яка максимально відокремлює об’єкти різних класів. Такий підхід дозволяє автоматизувати процес аналізу вебресурсів та підвищити ефективність виявлення потенційно небезпечних сайтів.

2.1.3 Формування ознак для класифікації вебресурсів

Формування ознак є одним із ключових етапів побудови моделей машинного навчання для задач класифікації вебресурсів. На цьому етапі необхідно визначити характеристики, які можуть бути використані алгоритмом для розрізнення фішингових та легітимних сайтів. Ознаки повинні відображати властивості вебресурсу, що дозволяють виявляти характерні відмінності між різними типами сайтів. Якість і релевантність сформованих ознак безпосередньо впливають на точність і ефективність роботи моделі класифікації.

У задачах виявлення фішингових вебресурсів ознаки можуть формуватися на основі різних джерел даних. Найбільш поширеним підходом є використання характеристик URL-адреси, таких як довжина доменного імені, кількість піддоменів, наявність спеціальних символів або використання IP-адреси замість доменного імені. Такі параметри дозволяють виявляти підозрілі особливості вебадрес, які часто використовуються зловмисниками для маскування фішингових сайтів.

Окрім характеристик URL-адреси, важливу роль відіграють ознаки, що пов’язані зі структурою та вмістом вебсторінки. До них можуть належати особливості HTML-коду, кількість зовнішніх посилань, наявність форм введення персональних даних або використання скриптів для збору інформації. Аналіз таких характеристик дозволяє виявляти поведінкові та

структурні особливості вебресурсів, які можуть свідчити про їх фішингову природу.

Після визначення набору характеристик ознаки перетворюються у числовий формат, який може бути використаний алгоритмами машинного навчання. Цей процес часто включає нормалізацію даних, кодування категоріальних значень та відбір найбільш інформативних ознак. Отриманий набір характеристик формує основу для навчання моделей класифікації, що дозволяє алгоритмам машинного навчання ефективно аналізувати вебресурси та визначати їх належність до відповідного класу.

2.2 Підготовка даних для навчання моделей

Підготовка даних [8] є одним із найважливіших етапів побудови моделей машинного навчання, оскільки саме якість і структура вхідних даних визначають ефективність роботи алгоритмів. У задачах класифікації вебресурсів необхідно сформувати структурований набір даних, що містить характеристики вебадрес або вебсторінок, які можуть використовуватися як ознаки для подальшого аналізу. Такі набори даних зазвичай містять значну кількість параметрів, що описують різні аспекти вебресурсу, включаючи структуру URL-адреси, властивості гіперпосилань, текстовий вміст сторінки та інші характеристики.

Перед використанням алгоритмів машинного навчання ці дані повинні бути підготовлені та приведені до формату, придатного для подальшої обробки. Сирі дані можуть містити помилки, пропущені значення, дублікати або нерелевантні параметри, що здатні негативно впливати на результати навчання моделей. Тому важливим завданням є проведення попереднього аналізу та очищення набору даних, що дозволяє підвищити його якість та забезпечити коректність подальших етапів обробки.

Підготовка даних також передбачає визначення набору ознак, які будуть використовуватися для навчання алгоритмів класифікації. У задачах виявлення фішингових вебресурсів ознаки можуть описувати як технічні характеристики вебадреси, так і поведінкові властивості вебсторінок. Вибір відповідних ознак є важливим фактором, оскільки надлишкова кількість характеристик може призводити до ускладнення моделі та збільшення часу її навчання.

Ще одним важливим етапом підготовки даних є розподіл набору даних на окремі підмножини для навчання та тестування моделей. Такий підхід дозволяє перевірити здатність алгоритмів узагальнювати отримані знання та правильно класифікувати нові дані. Завдяки цьому можна об’єктивно оцінити ефективність роботи моделей машинного навчання та визначити їхню придатність для використання у задачах виявлення фішингових вебресурсів.

Таким чином, підготовка даних включає комплекс процедур, спрямованих на очищення, аналіз та структурування інформації перед використанням алгоритмів машинного навчання. Якісно підготовлений набір даних дозволяє підвищити точність класифікації, зменшити кількість помилок під час навчання моделі та забезпечити більш стабільні результати під час її застосування.

2.2.1 Попередня обробка даних

Попередня обробка даних [9] є важливим етапом підготовки інформації перед використанням алгоритмів машинного навчання. На цьому етапі виконується аналіз початкового набору даних з метою виявлення можливих помилок, пропущених значень, дубльованих записів або некоректних параметрів. Наявність таких проблем у даних може негативно впливати на процес навчання моделей та призводити до зниження точності класифікації. Тому очищення та перевірка даних є необхідною

умовою для отримання достовірних результатів під час побудови моделей машинного навчання.

Одним із основних завдань попередньої обробки є усунення пропущених або некоректних значень у наборі даних. У деяких випадках такі значення можуть бути замінені середніми або медіанними значеннями відповідних параметрів, а в інших випадках записи з некоректними даними можуть бути видалені. Також на цьому етапі проводиться перевірка даних на наявність дублювань, які можуть спотворювати статистичні характеристики набору даних та впливати на результати навчання моделей.

Крім очищення даних, важливим елементом попередньої обробки є перетворення даних у формат, придатний для використання алгоритмами машинного навчання. Більшість алгоритмів працює з числовими значеннями, тому категоріальні або текстові параметри повинні бути закодовані у числовій формі. Для цього можуть використовуватися різні методи кодування, зокрема перетворення категорій у числові індекси або використання спеціальних схем кодування.

Ще одним важливим етапом попередньої обробки є масштабування або нормалізація числових ознак. У наборах даних різні параметри можуть мати різні діапазони значень, що може впливати на роботу деяких алгоритмів машинного навчання. Нормалізація дозволяє привести всі ознаки до одного масштабу, що сприяє більш стабільному та ефективному процесу навчання моделей. Таким чином, попередня обробка даних забезпечує підготовку якісного набору даних, який може бути використаний для подальшого аналізу та побудови моделей класифікації.

2.2.2 Відбір та аналіз ознак

Відбір ознак є важливим етапом підготовки даних у задачах машинного навчання, оскільки ефективність роботи моделі значною мірою залежить від набору характеристик, які використовуються для навчання

алгоритму. У задачах класифікації вебресурсів кількість потенційних ознак може бути досить великою, однак не всі вони мають однаковий вплив на результат класифікації. Наявність надлишкових або нерелевантних характеристик може призводити до ускладнення моделі, збільшення часу навчання та зниження точності прогнозування. Тому важливим завданням є визначення найбільш інформативних ознак, які забезпечують ефективне розділення об’єктів різних класів.

Аналіз ознак передбачає дослідження їхніх статистичних характеристик та взаємозв’язків між різними параметрами набору даних. Одним із поширених методів такого аналізу є дослідження кореляцій між ознаками, що дозволяє виявити параметри з високим рівнем взаємозалежності. Наявність сильно корельованих характеристик може свідчити про надлишковість інформації, тому частина таких ознак може бути виключена з подальшого аналізу без втрати важливої інформації для моделі.

Крім аналізу кореляції, у процесі відбору ознак можуть використовуватися різні статистичні та алгоритмічні методи, що дозволяють оцінити значущість окремих параметрів для задачі класифікації. До таких методів належать фільтраційні підходи, які базуються на статистичних характеристиках даних, а також вбудовані методи, що використовуються безпосередньо під час навчання моделей машинного навчання. Використання цих методів дозволяє сформувати оптимальний набір ознак, що забезпечує баланс між точністю моделі та складністю обчислень.

Результатом процесу відбору та аналізу ознак є формування компактного та інформативного набору характеристик, який використовується для навчання моделей машинного навчання. Такий підхід дозволяє підвищити ефективність алгоритмів класифікації, зменшити ризик перенавчання моделі та покращити її.

2.2.3 Розподіл даних на навчальну та тестову вибірки

Після завершення етапів попередньої обробки та відбору ознак підготовлений набір даних необхідно розподілити на окремі підмножини для навчання та оцінювання моделі машинного навчання. Такий підхід дозволяє перевірити здатність алгоритму не лише запам’ятовувати навчальні приклади, але й правильно працювати з новими даними. Розподіл даних є необхідним для об’єктивної оцінки ефективності моделі та запобігання перенавчанню.

Навчальна вибірка використовується для безпосереднього навчання алгоритму машинного навчання. На цьому етапі модель аналізує залежності між ознаками об’єктів та їхніми класами, формуючи внутрішню математичну модель, яка дозволяє виконувати класифікацію нових об’єктів. Чим більший та різноманітніший набір даних використовується для навчання, тим вищою може бути здатність моделі до узагальнення отриманих закономірностей.

Тестова вибірка використовується для перевірки ефективності навченої моделі. Вона містить дані, які не використовувалися під час процесу навчання, що дозволяє оцінити точність класифікації на нових прикладах. Такий підхід забезпечує об’єктивну перевірку якості моделі та дозволяє визначити, наскільки добре вона може працювати у реальних умовах.

У більшості задач машинного навчання набір даних розподіляється таким чином, що приблизно 70–80 % записів використовуються для навчання моделі, а решта 20–30 % призначається для тестування. Такий підхід дозволяє забезпечити достатній обсяг даних для навчання алгоритму та одночасно отримати достовірну оцінку його ефективності. У деяких випадках для більш точної оцінки якості моделі можуть також використовуватися додаткові методи перевірки.

2.3 Алгоритм Logistic Regression

Логістична регресія (рисунок 2.5) є одним із базових алгоритмів машинного навчання [10], який широко використовується для задач класифікації. Незважаючи на свою назву, цей метод не використовується для задач регресії у класичному розумінні, а застосовується для прогнозування належності об’єкта до певного класу. У задачах кібербезпеки, зокрема під час виявлення фішингових вебресурсів, логістична регресія може використовуватися для визначення ймовірності того, що певний вебресурс є легітимним або фішинговим.

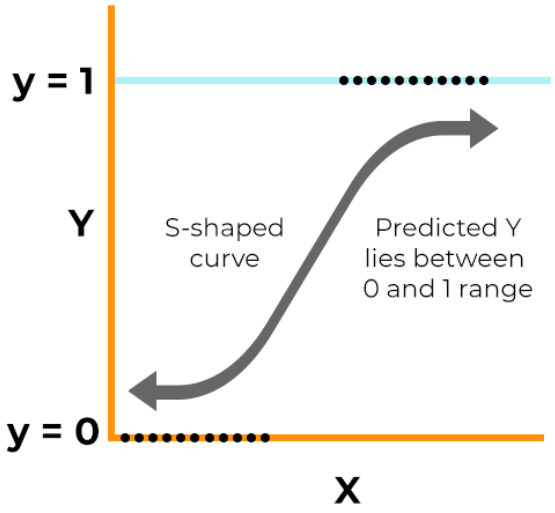

Рисунок 2.5 – Графічна інтерпретація логістичної функції у моделі Logistic Regression

Основою алгоритму є використання логістичної функції, яка перетворює лінійну комбінацію вхідних ознак у значення ймовірності у діапазоні від 0 до 1. Така функція має S-подібну форму, що дозволяє моделі плавно переходити від одного класу до іншого залежно від значень ознак.

Під час навчання моделі алгоритм визначає оптимальні значення вагових коефіцієнтів для кожної ознаки. Ці коефіцієнти відображають ступінь впливу відповідної характеристики на результат класифікації. У

процесі навчання модель мінімізує функцію помилки, що дозволяє підібрати такі параметри, за яких прогнозовані значення максимально відповідають реальним класам об’єктів у навчальній вибірці.

Логістична регресія має ряд переваг, серед яких відносна простота реалізації, висока швидкість навчання та зрозуміла інтерпретація результатів. Завдяки цьому алгоритм часто використовується як базова модель у задачах класифікації. У контексті виявлення фішингових вебресурсів логістична регресія дозволяє ефективно оцінювати ймовірність належності вебадреси до фішингового класу на основі сформованого набору ознак.

2.3.1 Математична модель логістичної регресії

Математична модель логістичної регресії базується на побудові функції, яка дозволяє оцінити ймовірність належності об’єкта до певного класу. На відміну від лінійної регресії, де результатом є безперервне числове значення, логістична регресія використовує спеціальну функцію, що перетворює лінійну комбінацію ознак у значення ймовірності. Це дозволяє застосовувати алгоритм для задач бінарної класифікації, де об’єкти можуть належати до одного з двох можливих класів.

Основою математичної моделі є лінійна комбінація вхідних ознак. Для кожного об’єкта формується набір характеристик, які множаться на відповідні вагові коефіцієнти, після чого результати підсумовуються. Отримане значення відображає вплив усіх ознак на результат класифікації. Таким чином модель формує певну лінійну функцію, що використовується як вхід для подальшого перетворення.

Для перетворення лінійної функції у ймовірність використовується логістична або сигмоїдна функція. Вона має S-подібну форму та переводить будь-яке числове значення у діапазон від 0 до 1. Отримане значення інтерпретується як ймовірність того, що об’єкт належить до позитивного

класу. Якщо це значення перевищує певний поріг, об’єкт відноситься до відповідного класу, інакше – до альтернативного.

Завдяки такій математичній моделі логістична регресія дозволяє ефективно працювати з великою кількістю ознак та визначати їхній вплив на результат класифікації. Це робить алгоритм зручним інструментом для задач кібербезпеки, зокрема для виявлення фішингових вебресурсів, де необхідно оцінювати сукупний вплив різних характеристик URL-адрес та структури вебсайтів.

2.3.2 Функція втрат та оптимізація параметрів

Для ефективної роботи моделі логістичної регресії необхідно визначити такі значення параметрів, які забезпечують найкращу відповідність між прогнозованими результатами та реальними даними. Для цього використовується функція втрат, яка дозволяє кількісно оцінити похибку моделі. Основною метою процесу навчання є мінімізація цієї функції, що забезпечує підвищення точності класифікації.

У задачах логістичної регресії зазвичай використовується логарифмічна функція втрат, яка також відома як логістична або крос-ентропійна функція. Вона оцінює різницю між реальними значеннями класів та ймовірностями, які прогнозує модель. Така функція дозволяє значно сильніше штрафувати модель за великі помилки, що сприяє більш точному налаштуванню параметрів.

Процес оптимізації параметрів полягає у пошуку таких значень вагових коефіцієнтів, при яких значення функції втрат буде мінімальним. Для цього застосовуються числові методи оптимізації, зокрема градієнтний спуск. Цей метод поступово змінює значення параметрів моделі у напрямку зменшення похибки, використовуючи інформацію про градієнт функції втрат.

У результаті багаторазового повторення процесу оптимізації модель поступово наближається до оптимального набору параметрів. Це дозволяє підвищити точність прогнозування та забезпечити більш надійну класифікацію об’єктів. У задачах виявлення фішингових вебресурсів така процедура оптимізації дозволяє моделі ефективно використовувати сформований набір ознак для визначення ймовірності того, що вебресурс є фішинговим.

2.3.3 Особливості застосування алгоритму для класифікації

Логістична регресія [11] є одним із найбільш поширених алгоритмів для розв’язання задач бінарної класифікації. На відміну від класичних методів регресії (рисунок 2.6), де прогнозується безперервне значення, у логістичній регресії результатом є ймовірність належності об’єкта до певного класу. Після обчислення цієї ймовірності модель використовує заданий поріг для прийняття остаточного рішення щодо класифікації об’єкта.

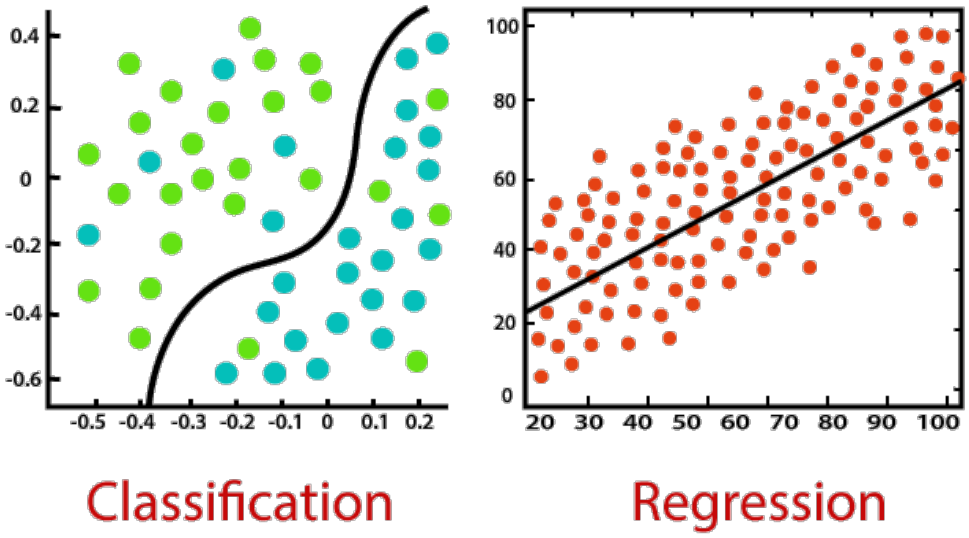

Рисунок 2.6 – Порівняння задач класифікації та регресії у машинному навчанні

Однією з особливостей використання логістичної регресії є здатність будувати лінійну межу прийняття рішень між класами. У багатовимірному просторі ознак така межа являє собою гіперплощину, яка розділяє об’єкти різних класів. Це дозволяє моделі визначати, до якої категорії належить новий об’єкт на основі значень його характеристик. Ще однією важливою особливістю логістичної регресії є можливість інтерпретації впливу окремих ознак на результат класифікації.

Кожен параметр моделі має відповідний ваговий коефіцієнт, який показує, наскільки сильно певна ознака впливає на ймовірність належності об’єкта до певного класу. Завдяки цьому дослідники можуть аналізувати, які саме характеристики вебресурсів мають найбільше значення під час виявлення фішингових сайтів. Логістична регресія також відзначається високою швидкістю навчання та відносно невеликою обчислювальною складністю. У задачах виявлення фішингових вебресурсів цей алгоритм часто використовується як базова модель для порівняння з більш складними методами машинного навчання.

2.4 Алгоритм K-Nearest Neighbors

Алгоритм K-Nearest Neighbors [12] є одним із простих і водночас ефективних методів машинного навчання, що застосовується для задач класифікації та регресії. Основна ідея цього алгоритму полягає у визначенні класу нового об’єкта на основі аналізу найближчих до нього об’єктів із навчальної вибірки. У випадку задач класифікації новий об’єкт отримує той клас, який найчастіше зустрічається серед його найближчих сусідів.

Принцип роботи алгоритму (рисунок 2.7) полягає у вимірюванні відстані між новим об’єктом та всіма іншими об’єктами навчальної вибірки. Після обчислення цих відстаней визначається певна кількість найближчих об’єктів, що позначається параметром k. Далі виконується аналіз класів цих

сусідів, і новий об’єкт класифікується відповідно до найбільш поширеного класу серед них.

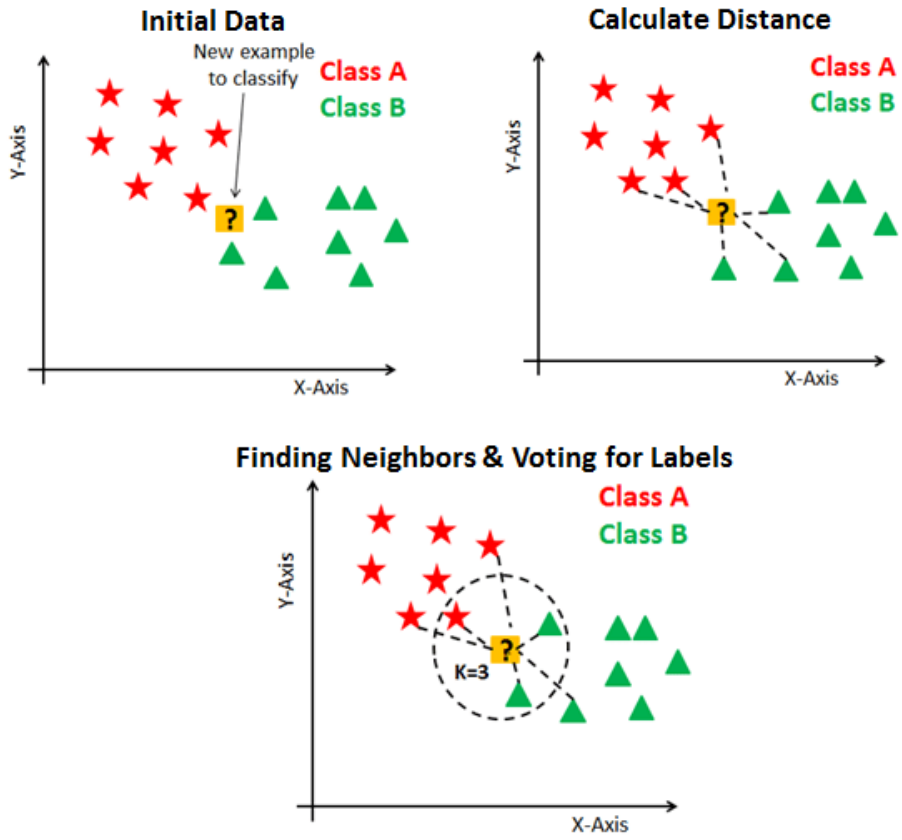

Рисунок 2.7 – Принцип роботи алгоритму K-Nearest Neighbors

Однією з важливих особливостей алгоритму KNN є відсутність явного етапу навчання моделі. Алгоритм не будує математичну модель під час тренування, а використовує всі дані навчальної вибірки безпосередньо під час класифікації нового об’єкта. Такий підхід дозволяє гнучко реагувати на структуру даних, проте може призводити до збільшення обчислювальних витрат при роботі з великими наборами даних.

У задачах виявлення фішингових вебресурсів алгоритм KNN може використовуватися для класифікації вебадрес на основі їхніх характеристик. Якщо новий вебресурс має подібні ознаки до тих, що характерні для фішингових сайтів у навчальній вибірці, алгоритм віднесе його до відповідного класу. Завдяки своїй простоті та зрозумілості KNN часто

використовується як базовий алгоритм для порівняння ефективності різних методів машинного навчання.

2.4.1 Принцип роботи методу найближчих сусідів

Метод найближчих сусідів базується на припущенні [13], що об’єкти, які мають схожі характеристики, зазвичай належать до одного класу. У цьому підході новий об’єкт класифікується шляхом порівняння його параметрів з параметрами об’єктів, що вже присутні у навчальній вибірці.

Основна ідея полягає у визначенні найближчих до нього точок у просторі ознак та використанні інформації про їхні класи для прийняття рішення.

Для реалізації цього підходу (рисунок 2.8) кожен об’єкт представляється у вигляді точки у багатовимірному просторі ознак. Відстань між об’єктами визначається за допомогою певної метрики, найчастіше використовується евклідова відстань. Після обчислення відстаней визначається k найближчих сусідів, які розташовані найближче до нового об’єкта.

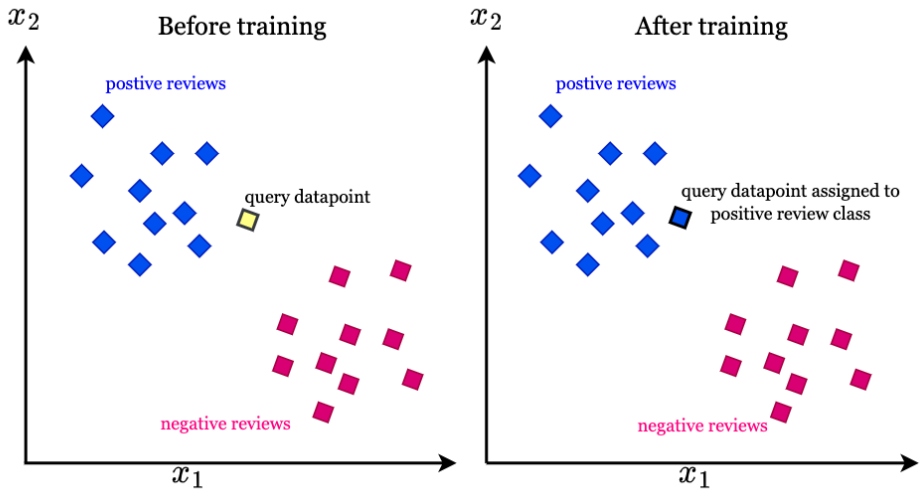

Рисунок 2.8 – Приклад класифікації нового об’єкта за допомогою методу найближчих сусідів у просторі ознак

Після визначення найближчих сусідів виконується процедура голосування, під час якої аналізується, до яких класів належать ці об’єкти. Новий об’єкт отримує той клас, який зустрічається найчастіше серед обраних сусідів. Такий підхід дозволяє враховувати локальну структуру даних та забезпечує достатньо високу точність класифікації у випадках, коли об’єкти одного класу утворюють компактні групи у просторі ознак.

Метод найближчих сусідів не потребує побудови складної математичної моделі, що робить його простим для реалізації та інтерпретації. Водночас ефективність цього методу значною мірою залежить від правильного вибору параметра k та метрики відстані. У задачах виявлення фішингових вебресурсів метод найближчих сусідів може використовуватися для класифікації вебадрес на основі їхніх характеристик, визначаючи схожість нових вебресурсів із відомими фішинговими або легітимними сайтами.

2.4.2 Метрики відстані у задачах класифікації

У алгоритмі K-Nearest Neighbors визначення схожості між об’єктами здійснюється за допомогою спеціальних функцій, які називаються метриками відстані. Ці метрики дозволяють оцінити ступінь близькості між об’єктами у просторі ознак. Від правильного вибору метрики залежить ефективність роботи алгоритму, оскільки саме вона визначає, які об’єкти будуть вважатися найближчими сусідами для нового елемента.

Найбільш поширеною метрикою є евклідова відстань [14], яка обчислює геометричну відстань між двома точками у багатовимірному просторі. Вона використовується у випадках, коли ознаки мають числовий характер і знаходяться у порівнянних масштабах. Евклідова відстань дозволяє оцінити реальну геометричну близькість між об’єктами, що робить її зручною для багатьох задач класифікації.

Окрім евклідової відстані, у задачах машинного навчання можуть застосовуватися й інші метрики. Наприклад, мангеттенська відстань визначає відстань між точками як суму абсолютних різниць їх координат. Також може використовуватися косинусна міра подібності, яка оцінює схожість між об’єктами на основі кута між їхніми векторами ознак. Вибір конкретної метрики залежить від структури даних та характеру використовуваних ознак.

У задачах виявлення фішингових вебресурсів метрики відстані дозволяють визначати ступінь подібності між характеристиками різних вебадрес або вебсторінок. Якщо новий вебресурс має ознаки, подібні до характеристик відомих фішингових сайтів, алгоритм може класифікувати його як підозрілий.

Таким чином, правильний вибір метрики відстані є важливим фактором, який впливає на точність та надійність роботи алгоритму класифікації.

2.4.3 Переваги та недоліки алгоритму

Алгоритм K-Nearest Neighbors має ряд переваг, які роблять його популярним у задачах машинного навчання. Однією з основних переваг є простота реалізації та зрозумілий принцип роботи. Алгоритм не потребує складного етапу навчання, оскільки всі обчислення виконуються безпосередньо під час класифікації нового об’єкта. Це дозволяє швидко застосовувати його для аналізу різних наборів даних.

Ще однією важливою перевагою є гнучкість алгоритму та можливість застосування до різних типів задач. KNN може використовуватися як для класифікації, так і для задач регресії. Крім того, алгоритм добре працює у випадках, коли структура даних є складною та не може бути легко описана за допомогою простої математичної моделі.

Разом з тим алгоритм має і певні недоліки. Одним із основних є висока обчислювальна складність під час класифікації, особливо у випадку великих наборів даних. Оскільки алгоритм порівнює новий об’єкт з усіма об’єктами навчальної вибірки, час виконання може значно зростати при збільшенні обсягу даних. Це може ускладнювати використання алгоритму у системах, що працюють у режимі реального часу.

Ще одним недоліком є чутливість алгоритму до вибору параметра k та до наявності нерелевантних ознак у наборі даних. Неправильний вибір кількості сусідів може призводити до зниження точності класифікації.

2.5 Алгоритм XGBoost

Алгоритм XGBoost [15] є одним із сучасних і високоефективних методів машинного навчання, що широко використовується для задач класифікації та регресії. Він належить до класу ансамблевих алгоритмів, які поєднують результати роботи кількох моделей для підвищення точності прогнозування. Основна ідея алгоритму полягає у побудові послідовності дерев рішень, кожне з яких коригує помилки попередніх моделей.

У XGBoost (рисунок 2.9) використовується метод градієнтного бустингу, що передбачає поетапне навчання нових моделей на основі залишкових помилок попередніх. Під час кожної ітерації алгоритм будує нове дерево рішень, яке намагається мінімізувати похибку прогнозування. Таким чином кожна наступна модель покращує результати попередніх, що дозволяє поступово підвищувати точність класифікації.

Однією з важливих особливостей XGBoost є використання ефективних методів оптимізації та регуляризації. Регуляризація дозволяє зменшити ризик перенавчання моделі, що є важливою проблемою у складних задачах машинного навчання. Крім того, алгоритм підтримує паралельні обчислення, що значно прискорює процес навчання при роботі з великими наборами даних.

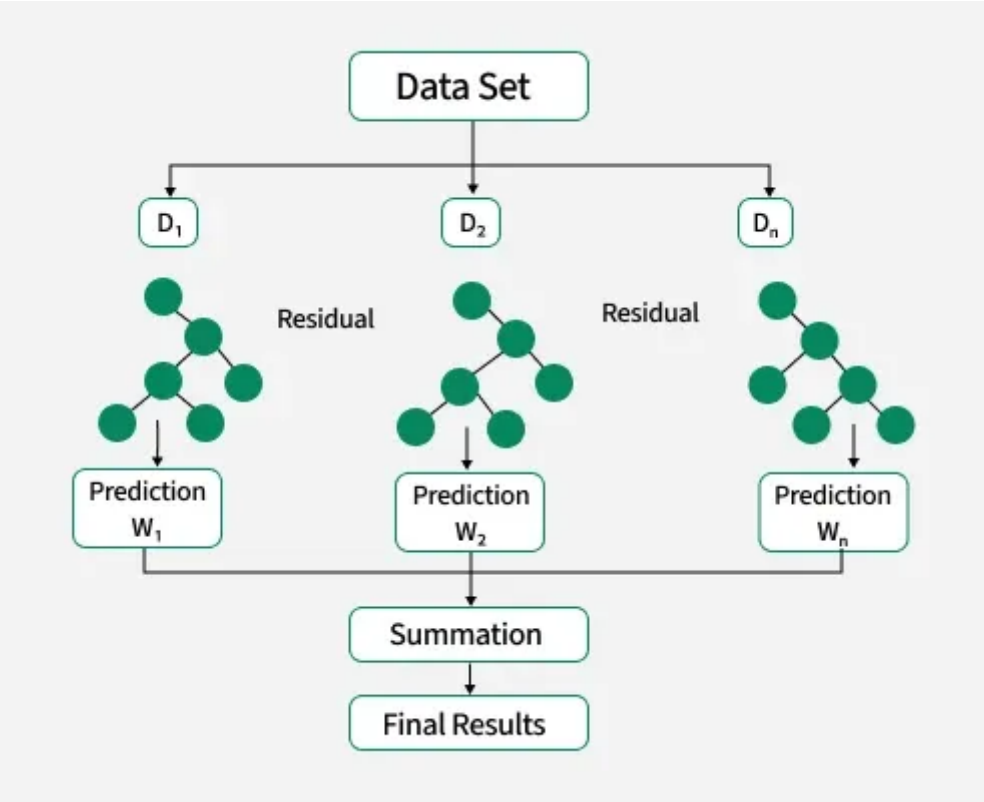

Рисунок 2.9 – Схематичне представлення роботи алгоритму XGBoost та формування ансамблю дерев рішень

У задачах виявлення фішингових вебресурсів алгоритм XGBoost дозволяє ефективно аналізувати велику кількість ознак, які описують властивості вебадрес або структуру вебсайтів. Завдяки поєднанню кількох моделей у єдиний ансамбль алгоритм здатний виявляти складні залежності між ознаками та забезпечувати високу точність класифікації. Це робить XGBoost одним із найбільш перспективних методів для побудови систем автоматичного виявлення фішингових вебресурсів.

2.5.1 Основні принципи градієнтного бустингу

Градієнтний бустинг (рисунок 2.10) є одним із ефективних підходів ансамблевого машинного навчання, який використовується для побудови високоточних моделей класифікації та регресії [16]. Основна ідея цього

методу полягає у послідовному створенні кількох моделей, кожна з яких покращує результати попередніх. Замість побудови однієї складної моделі алгоритм формує ансамбль простіших моделей, які разом забезпечують більш точний результат прогнозування.

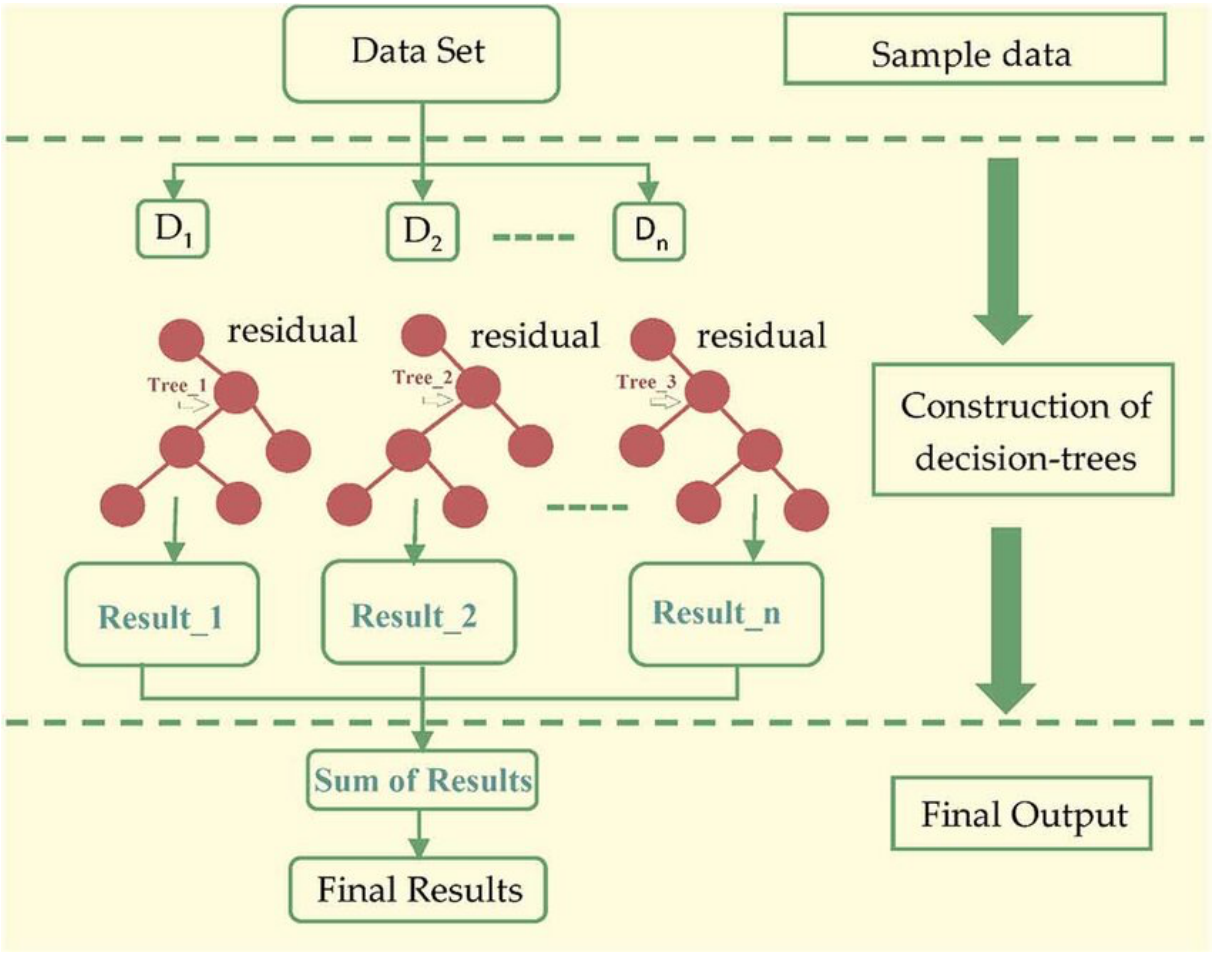

Рисунок 2.10 – Схема роботи алгоритму градієнтного бустингу з послідовним навчанням дерев рішень на основі залишкових помилок

У процесі роботи алгоритм будує послідовність дерев рішень [17], де кожне нове дерево навчається на помилках попередніх моделей. Іншими словами, після побудови першої моделі визначається різниця між прогнозованими та реальними значеннями, яка називається залишком або помилкою. Наступна модель навчається таким чином, щоб зменшити ці помилки.

Ключовою особливістю градієнтного бустингу є використання методу градієнтної оптимізації для мінімізації функції втрат. На кожному етапі алгоритм визначає напрямок, у якому необхідно змінити параметри моделі для зменшення похибки прогнозування. Таким чином кожна нова модель додається до ансамблю, щоб зменшити загальну помилку системи.

Завдяки такому підходу градієнтний бустинг дозволяє ефективно моделювати складні залежності між ознаками та результатом класифікації. У задачах виявлення фішингових вебресурсів цей метод дозволяє аналізувати велику кількість характеристик вебсайтів та визначати їхній вплив на результат класифікації.

2.5.2 Архітектура алгоритму XGBoost

Архітектура алгоритму XGBoost [18] базується на використанні ансамблю дерев рішень, які послідовно навчаються для покращення результатів прогнозування.

Кожне дерево будується таким чином, щоб мінімізувати помилки, допущені попередніми моделями. На відміну від класичних алгоритмів бустингу, XGBoost використовує оптимізовану структуру побудови дерев, що дозволяє значно підвищити ефективність навчання та швидкість обробки даних.

Однією з ключових особливостей архітектури XGBoost є використання функції втрат із додатковим регуляризаційним компонентом. Регуляризація дозволяє контролювати складність моделі та зменшувати ризик перенавчання.

У моделі враховується не лише точність прогнозування, але й структура побудованих дерев, що дозволяє отримувати більш стабільні результати під час класифікації нових даних.

Ще однією важливою складовою архітектури XGBoost є використання ефективних алгоритмів побудови дерев рішень. Під час

навчання алгоритм аналізує різні можливі варіанти поділу даних та обирає ті, що забезпечують найбільше зменшення функції втрат. Для підвищення швидкості роботи застосовуються оптимізовані методи обчислення, що дозволяють ефективно працювати навіть із великими наборами даних.

Крім того, XGBoost підтримує паралельні обчислення та ефективне використання апаратних ресурсів. Завдяки цьому алгоритм може обробляти значні обсяги інформації та виконувати навчання моделей значно швидше порівняно з багатьма іншими методами машинного навчання. Така архітектура робить XGBoost одним із найбільш продуктивних алгоритмів для задач аналізу даних та класифікації.

2.5.3 Особливості використання XGBoost у задачах класифікації

Алгоритм XGBoost широко використовується у задачах класифікації завдяки високій точності прогнозування та здатності працювати з великими наборами ознак. У задачах класифікації модель навчається визначати залежності між характеристиками об’єктів та їхніми класами. У результаті алгоритм може прогнозувати належність нових об’єктів до певної категорії на основі їхніх параметрів.

Однією з важливих особливостей використання XGBoost є його здатність працювати з великою кількістю ознак різного типу. Алгоритм ефективно обробляє як числові, так і категоріальні характеристики, що дозволяє застосовувати його у різноманітних задачах аналізу даних. Крім того, модель автоматично визначає найбільш важливі ознаки, що впливають на результат класифікації.

У задачах кібербезпеки XGBoost активно використовується для виявлення шкідливих або підозрілих вебресурсів. Аналізуючи характеристики URL-адрес, структуру вебсторінок та інші параметри, алгоритм здатний визначати закономірності, характерні для фішингових

сайтів. Це дозволяє автоматизувати процес виявлення загроз та підвищити ефективність систем захисту інформації.

2.6 Методи оцінювання ефективності моделей машинного навчання

Оцінювання ефективності моделей машинного навчання [19] є важливим етапом дослідження, оскільки дозволяє визначити якість роботи алгоритмів класифікації. Для цього використовуються спеціальні статистичні метрики та методи аналізу результатів прогнозування. Вони дозволяють порівнювати різні моделі між собою та визначати алгоритм, який забезпечує найкращу точність класифікації у задачі виявлення фішингових вебресурсів.

2.6.1 Матриця помилок класифікації

Матриця помилок класифікації є одним із основних інструментів оцінювання результатів роботи моделей машинного навчання. Вона представляє собою таблицю, у якій відображаються правильні та неправильні результати класифікації, отримані моделлю. Матриця містить чотири основні типи результатів: правильні позитивні, правильні негативні, хибні позитивні та хибні негативні передбачення.

Аналіз матриці помилок дозволяє зрозуміти, які саме помилки допускає модель та наскільки ефективно вона розрізняє об’єкти різних класів. У задачах виявлення фішингових вебресурсів це дозволяє оцінити, наскільки добре алгоритм розпізнає фішингові сайти та наскільки часто він помилково класифікує легітимні вебресурси як підозрілі.

Крім того, матриця помилок дозволяє виявити дисбаланс у результатах класифікації, який може виникати у випадках, коли одна категорія об’єктів представлена у наборі даних значно більше, ніж інша. У таких ситуаціях модель може демонструвати високу загальну точність, але

при цьому погано розпізнавати об’єкти менш представленого класу. Саме тому детальний аналіз елементів матриці помилок є важливим етапом оцінювання ефективності моделей машинного навчання.

2.6.2 Показники точності класифікації

На основі матриці помилок обчислюються різні показники ефективності класифікації, які дозволяють кількісно оцінити якість роботи моделі. До найпоширеніших метрик належать точність, повнота, точність позитивного прогнозу та F-міра. Кожна з цих метрик характеризує різні аспекти ефективності алгоритму та дозволяє оцінити його здатність правильно класифікувати об’єкти різних класів.

Використання кількох показників дозволяє отримати більш повну оцінку якості моделі. Наприклад, точність показує частку правильних прогнозів серед усіх результатів, тоді як повнота характеризує здатність моделі виявляти всі об’єкти певного класу. У задачах кібербезпеки важливо враховувати всі ці показники, оскільки навіть невелика кількість помилок може призводити до значних ризиків.

Особливе значення має F-міра, яка є гармонійним середнім значенням точності та повноти. Цей показник дозволяє отримати узагальнену оцінку ефективності моделі у випадках, коли необхідно одночасно враховувати обидва параметри. У задачах виявлення фішингових вебресурсів використання таких комплексних показників дозволяє більш об’єктивно оцінити ефективність алгоритмів машинного навчання та визначити найбільш придатну модель для практичного застосування.

2.6.3 ROC-крива та AUC

ROC-крива є графічним методом оцінювання ефективності моделей класифікації. Вона відображає співвідношення між часткою правильно

визначених позитивних результатів та часткою хибних позитивних спрацьовувань при різних значеннях порогу класифікації.

Аналіз ROC-кривої дозволяє оцінити здатність моделі розрізняти об’єкти різних класів. Для кількісної оцінки ROC-кривої використовується показник AUC, який визначає площу під кривою. Значення AUC знаходиться у наступному діапазоні від 0 до 1, де більші значення свідчать про кращу здатність моделі до класифікації. У задачах виявлення фішингових вебресурсів цей показник дозволяє об’єктивно порівнювати ефективність різних алгоритмів машинного навчання.

3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 План експериментів

Експериментальна частина роботи спрямована на дослідження ефективності алгоритмів машинного навчання для задачі виявлення фішингових вебресурсів. Основною метою проведення експерименту є порівняльний аналіз декількох моделей класифікації, що дозволить визначити алгоритм, який забезпечує найкращі результати під час автоматичного розпізнавання потенційно небезпечних вебпосилань. У межах дослідження розглядається задача бінарної класифікації, у якій кожна URL-адреса має бути віднесена до одного з двох класів – легітимний вебресурс або фішинговий вебресурс. Такий підхід дозволяє оцінити можливості алгоритмів машинного навчання щодо виявлення загроз інформаційній безпеці у мережі Інтернет.

План експериментального дослідження передбачає послідовне виконання декількох етапів. На першому етапі планується провести аналіз та підготовку набору даних, що використовується для навчання моделей. Це включає дослідження структури даних, аналіз розподілу ознак, перевірку наявності пропущених значень та виявлення можливих залежностей між характеристиками. Далі передбачається виконати попередню обробку даних, що може включати усунення надлишкових ознак, зменшення кореляції між параметрами та підготовку даних до подальшого використання у процесі навчання моделей машинного навчання.

На наступному етапі експерименту планується здійснити побудову та навчання декількох моделей машинного навчання, які будуть використані для розв’язання задачі класифікації вебресурсів. Після навчання моделей буде виконано їх тестування на окремій вибірці даних, що дозволить об’єктивно оцінити якість роботи алгоритмів. Для оцінювання ефективності моделей передбачається використання стандартних метрик класифікації,

зокрема точності, матриці помилок та ROC-кривої. Отримані результати планується використати для проведення порівняльного аналізу моделей та визначення алгоритму, який демонструє найкращу ефективність у задачі виявлення фішингових вебресурсів.

3.1.1 Постановка експериментальної задачі

Однією з актуальних задач сучасної кібербезпеки є своєчасне виявлення фішингових вебресурсів, які використовуються зловмисниками для отримання конфіденційної інформації користувачів. Такі вебсайти часто маскуються під легітимні ресурси та можуть поширювати шкідливе програмне забезпечення або вводити користувачів в оману з метою викрадення персональних даних. У зв’язку з цим виникає необхідність розробки автоматизованих методів аналізу вебпосилань, здатних визначати потенційно небезпечні ресурси на основі їх характеристик.

Експериментальна задача дослідження полягає у побудові та дослідженні моделей машинного навчання, які здатні виконувати класифікацію URL-адрес за ознакою належності до фішингових або легітимних вебресурсів. Для розв’язання цієї задачі використовується набір даних, що містить опис різних характеристик вебпосилань. Кожен запис у наборі даних представлений набором ознак, які відображають технічні або структурні властивості URL-адреси, а також міткою класу, що визначає тип вебресурсу. У межах експерименту передбачається використання цих ознак як вхідних параметрів для алгоритмів машинного навчання.

У процесі дослідження планується реалізувати та дослідити декілька моделей машинного навчання, що належать до різних підходів до класифікації даних. Основною метою експерименту є оцінювання якості роботи цих моделей під час розв’язання задачі виявлення фішингових вебресурсів. Для цього передбачається провести навчання моделей на підготовленому наборі даних та виконати тестування їх ефективності на

окремій вибірці. Отримані результати дозволять здійснити порівняльний аналіз моделей та визначити алгоритм, який забезпечує найкращу якість класифікації вебресурсів.

3.1.2 Загальна схема експериментального дослідження

Експериментальне дослідження у межах даної роботи передбачає послідовне виконання декількох етапів, спрямованих на побудову та оцінювання моделей машинного навчання для задачі виявлення фішингових вебресурсів. Загальна схема експерименту охоплює підготовку та аналіз набору даних, попередню обробку інформації, навчання моделей машинного навчання та подальше оцінювання їх ефективності. Такий підхід дозволяє систематизувати процес проведення дослідження та забезпечити об’єктивність отриманих результатів.

На першому етапі експериментального дослідження передбачається виконання аналізу набору даних, що використовується для навчання моделей. Цей етап включає дослідження структури даних, визначення кількості записів та ознак, а також перевірку набору даних на наявність пропущених або дубльованих значень. Додатково планується провести аналіз розподілу значень ознак та дослідити можливі взаємозв’язки між ними. Для цього використовуються методи розвідувального аналізу даних, зокрема побудова гістограм розподілу та теплових карт кореляції між характеристиками набору даних.

Після проведення попереднього аналізу виконується етап підготовки даних до навчання моделей машинного навчання. На цьому етапі передбачається усунення надлишкових або сильно корельованих ознак, формування матриці вхідних параметрів та цільової змінної, а також поділ набору даних на навчальну та тестову вибірки. Далі здійснюється побудова моделей класифікації, їх навчання на навчальній вибірці та перевірка якості роботи на тестових даних. Завершальним етапом експерименту є

оцінювання ефективності моделей за допомогою відповідних метрик та проведення їх порівняльного аналізу, що дозволяє визначити найбільш ефективний алгоритм для задачі виявлення фішингових вебресурсів.

3.1.3 Обрані алгоритми машинного навчання для порівняння

Для проведення експериментального дослідження необхідно обрати декілька алгоритмів машинного навчання, які дозволяють виконувати класифікацію даних та можуть бути застосовані для задачі виявлення фішингових вебресурсів. У межах цієї роботи для порівняльного аналізу обрано три алгоритми машинного навчання, що належать до різних підходів до побудови моделей класифікації: Logistic Regression, K-Nearest Neighbors та XGBoost. Використання алгоритмів різної природи дозволяє дослідити їх ефективність у задачі класифікації вебпосилань та визначити підхід, який демонструє найкращі результати.

Алгоритм Logistic Regression належить до лінійних моделей машинного навчання та широко використовується для задач бінарної класифікації. Основна ідея цього методу полягає у визначенні ймовірності належності об’єкта до певного класу на основі лінійної комбінації вхідних ознак. Незважаючи на відносну простоту моделі, Logistic Regression часто використовується як базовий алгоритм для порівняння результатів інших методів машинного навчання, оскільки дозволяє отримати інтерпретовані результати та швидко навчатися на великих наборах даних.

Другим алгоритмом, що використовується у дослідженні, є метод K-Nearest Neighbors. Цей алгоритм належить до методів навчання на основі близькості об’єктів і базується на припущенні, що подібні об’єкти мають подібні характеристики. Під час класифікації новий об’єкт відноситься до того класу, який є найбільш поширеним серед найближчих до нього об’єктів у просторі ознак.

Третім алгоритмом є XGBoost – один із сучасних методів ансамблевого навчання, який базується на принципі градієнтного бустингу над деревами рішень. Цей алгоритм здатний ефективно моделювати складні нелінійні залежності у даних та часто демонструє високі результати у задачах класифікації. Порівняння результатів роботи зазначених алгоритмів дозволить визначити модель, яка забезпечує найкращу ефективність у задачі виявлення фішингових вебресурсів.

3.2 Опис використаних наборів даних

Для проведення експериментального дослідження використовується набір даних Book-My-Show (рисунок 3.1), що містить інформацію про вебпосилання та їх характеристики, які бути використані для визначення належності ресурсу до фішингових або легітимних вебсайтів [20]. Даний набір даних включає приблизно 11 тисяч записів, кожен з яких відповідає окремій URL-адресі. Кожен запис описується набором ознак, що відображають технічні або структурні властивості вебпосилання, зокрема характеристики доменного імені, параметри безпеки, використання спеціальних символів у URL-адресі та інші особливості.

Рисунок 3.1 – Структура набору даних

Набір даних має табличну структуру, у якій кожен рядок відповідає окремому вебресурсу, а кожен стовпець містить значення певної ознаки. Більшість ознак представлені у вигляді числових значень, що відображають наявність або відсутність певної характеристики вебпосилання. Зокрема, ознаки можуть набувати значень −1, 0 або 1, які використовуються для кодування різних типів властивостей URL-адреси. Такий спосіб представлення дозволяє ефективно використовувати ці дані як вхідні параметри для алгоритмів машинного навчання.

3.2.1 Загальна характеристика набору даних

Структура набору даних (рисунок 3.2) включає 32 стовпці, серед яких міститься 31 ознака представлена числовими значеннями, а один стовпець містить текстову інформацію. Така структура дозволяє використовувати більшість характеристик без додаткового перетворення під час навчання моделей машинного навчання. Наявність значної кількості ознак, які описують різні аспекти структури вебпосилань, створює можливість для побудови ефективних моделей класифікації, здатних визначати потенційно небезпечні вебресурси.

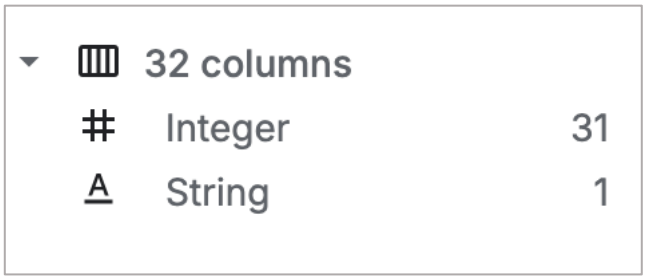

Рисунок 3.2 – Характеристика типів даних у наборі даних

Кожен запис у наборі даних представлений сукупністю характеристик, що відображають різні параметри URL-адреси та пов’язаних

з нею елементів вебсторінки. До таких характеристик належать, зокрема, довжина URL-адреси, наявність символу «@», використання сервісів скорочення посилань, кількість піддоменів, наявність перенаправлень, використання SSL-сертифікатів та інші ознаки, що можуть використовуватися для аналізу поведінки вебресурсу. Значення ознак кодуються числовими значеннями, що дозволяє використовувати їх як вхідні параметри для алгоритмів машинного навчання.

3.2.2 Структура та ознаки набору даних

Для дослідження структури набору даних та первинного аналізу його характеристик використовується програмний код, наведений у лістингу 3.1. На цьому етапі здійснюється видалення службового стовпця, що не використовується у подальшому аналізі, а також визначається кількість унікальних значень для кожної ознаки. Крім того, виконується перевірка набору даних на наявність пропущених значень та аналіз структури таблиці, що дозволяє визначити типи даних для кожного стовпця та переконатися у коректності їх представлення.

Лістинг 3.1 – Програмний код, що реалізує аналіз структури набору даних та дослідження розподілу ознак

```
df_data = df_data.drop('index',1)
df_data.nunique()
#check for NULL value in the dataset
df_data.isnull().sum().sum()
df_data.info()
df1 = df_data.T
print(df1.duplicated().sum())  # there is no duplicate column values
df_data.hist(bins=50, figsize=(20,15))
plt.show()
pd.value_counts(df_data['Result']).plot.bar()
```

Додатково у межах попереднього аналізу виконується перевірка набору даних на наявність дублювань ознак. Після цього проводиться побудова гістограм розподілу значень для ознак набору даних, що дозволяє дослідити характер розподілу параметрів та виявити можливі особливості даних. Також аналізується розподіл значень цільової змінної Result, що дає змогу оцінити співвідношення між класами легітимних та фішингових вебресурсів. Подальший аналіз структури (рисунок 3.3) набору даних дозволяє визначити кількість можливих значень для кожної ознаки.

```
having_IPhaving_IP_Address     2
URLURL_Length                  3
Shortining_Service             2
having_At_Symbol               2
double_slash_redirecting       2
Prefix_Suffix                  2
having_Sub_Domain              3
SSLfinal_State                 3
Domain_registeration_length    2
Favicon                        2
port                           2
HTTPS_token                    2
Request_URL                    2
URL_of_Anchor                  3
Links_in_tags                  3
SFH                            3
Submitting_to_email            2
Abnormal_URL                   2
Redirect                       2
on_mouseover                   2
RightClick                     2
popUpWidnow                    2
Iframe                         2
age_of_domain                  2
DNSRecord                      2
web_traffic                    3
Page_Rank                      2
Google_Index                   2
Links_pointing_to_page         3
Statistical_report             2
Result                         2
dtype: int64
```

Рисунок 3.3 – Кількість унікальних значень для ознак набору даних

Результати такого аналізу демонструють, що більшість характеристик набору даних мають обмежену кількість можливих значень, як правило два або три. Це пояснюється тим, що значення ознак використовуються для кодування певних властивостей вебпосилань, наприклад наявності або відсутності певної характеристики. Це значно спрощує процес підготовки даних до навчання моделей та зменшує необхідність у складних процедурах попереднього перетворення ознак.

Для більш детального дослідження структури набору даних також проводиться аналіз розподілу значень для кожної ознаки (рисунок 3.4). Для цього будуються гістограми, які відображають частоту появи різних значень параметрів у наборі даних. Такий підхід дозволяє візуально оцінити характер розподілу ознак та визначити можливі дисбаланси у даних.

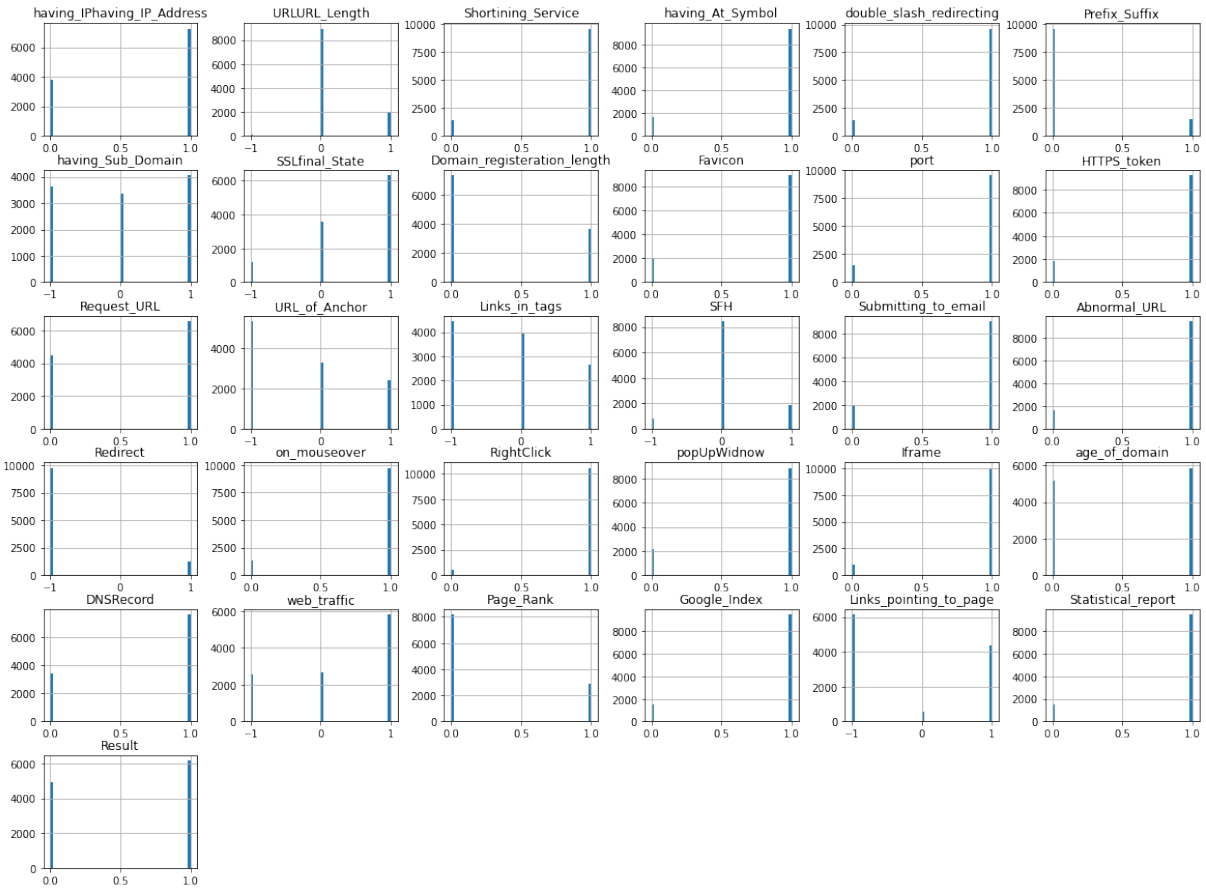

Рисунок 3.4 – Гістограми розподілу значень ознак набору даних

Окрему увагу приділено аналізу розподілу значень цільової змінної Result (рисунок 3.5), яка визначає клас вебресурсу. Для цього будується стовпчикова діаграма, що демонструє співвідношення між різними класами у наборі даних.

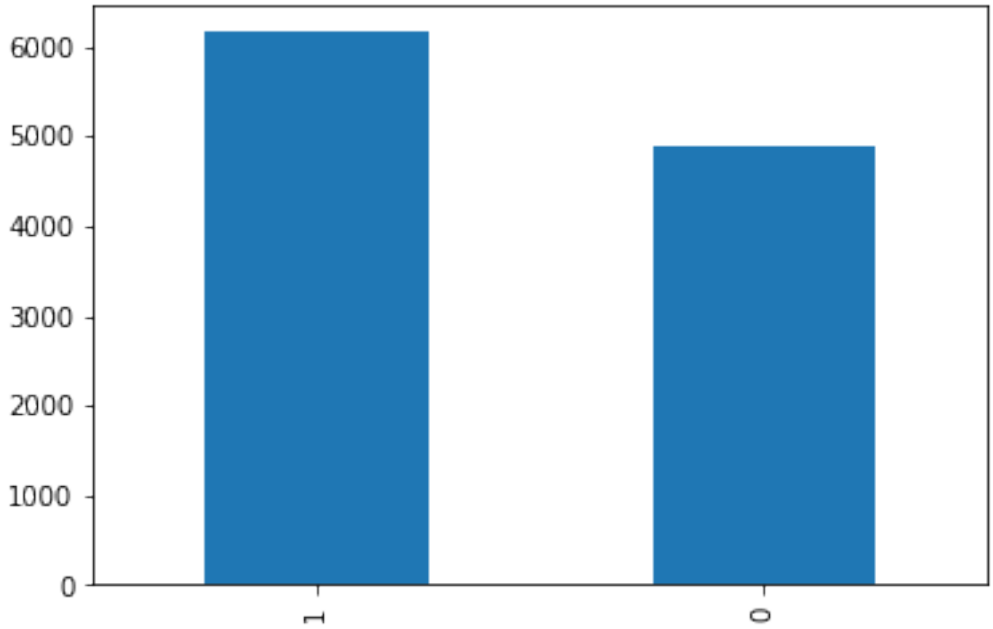

Рисунок 3.5 – Розподіл класів цільової змінної Result у наборі даних

Такий аналіз дозволяє оцінити баланс між легітимними та фішинговими вебресурсами, що є важливим фактором для подальшого навчання моделей машинного навчання.

3.2.3 Попередня обробка даних

Важливим етапом підготовки даних перед застосуванням алгоритмів машинного навчання є попередня обробка набору даних. Основною метою цього етапу є підвищення якості вхідних даних, зменшення надлишковості ознак та усунення можливих залежностей між параметрами, що можуть негативно впливати на результати навчання моделей.

Для виявлення залежностей між ознаками набору даних було виконано аналіз кореляції між параметрами (рисунок 3.6). З цією метою було побудовано кореляційну матрицю, яка відображає ступінь взаємозв’язку між усіма характеристиками набору даних. Візуалізація кореляційної матриці у вигляді теплової карти дозволяє наочно визначити пари ознак, між якими спостерігається сильна залежність. Значення кореляції, близькі до 1 або −1, свідчать про сильний взаємозв’язок між відповідними характеристиками, тоді як значення, близькі до нуля, вказують на відсутність суттєвої залежності між ознаками. Для реалізації аналізу кореляції та подальшого відбору ознак використовується програмний код, наведений у лістингу 3.2.

Лістинг 3.2 – Програмний код, що реалізує аналіз кореляції ознак та видалення сильно корельованих параметрів

```
    #correlation map
    f,ax=plt.subplots(figsize=(18,18))
    sns.heatmap(df_data.corr(),annot=True,      linewidths=.5,
fmt='.1f',ax=ax)
    cor_matrix = df_data.corr().abs()
    upper_tri=cor_matrix.where(np.triu(np.ones(cor_matrix.shap
e),k=1).astype(np.bool))
    # threshold greater than 0.75
    to_drop = [column for column in upper_tri.columns if
any(upper_tri[column] > 0.75)]
    print(to_drop)
    #drop the columns which are highly correlated
    df_data.drop(to_drop,axis=1,inplace=True)
```

У межах цього коду формується матриця кореляцій (рисунок 3.6) між ознаками набору даних, після чого здійснюється пошук пар параметрів, для яких значення коефіцієнта кореляції перевищує заданий поріг. У даному дослідженні порогове значення кореляції встановлено на рівні 0,75. Це

означає, що у випадку, якщо між двома ознаками спостерігається сильний взаємозв'язок, одна з них вважається надлишковою та може бути вилучена з набору даних. Після визначення списку сильно корельованих характеристик виконується видалення відповідних стовпців з набору даних. Такий підхід дозволяє зменшити кількість ознак, уникнути дублювання інформації та спростити подальший процес навчання моделей.

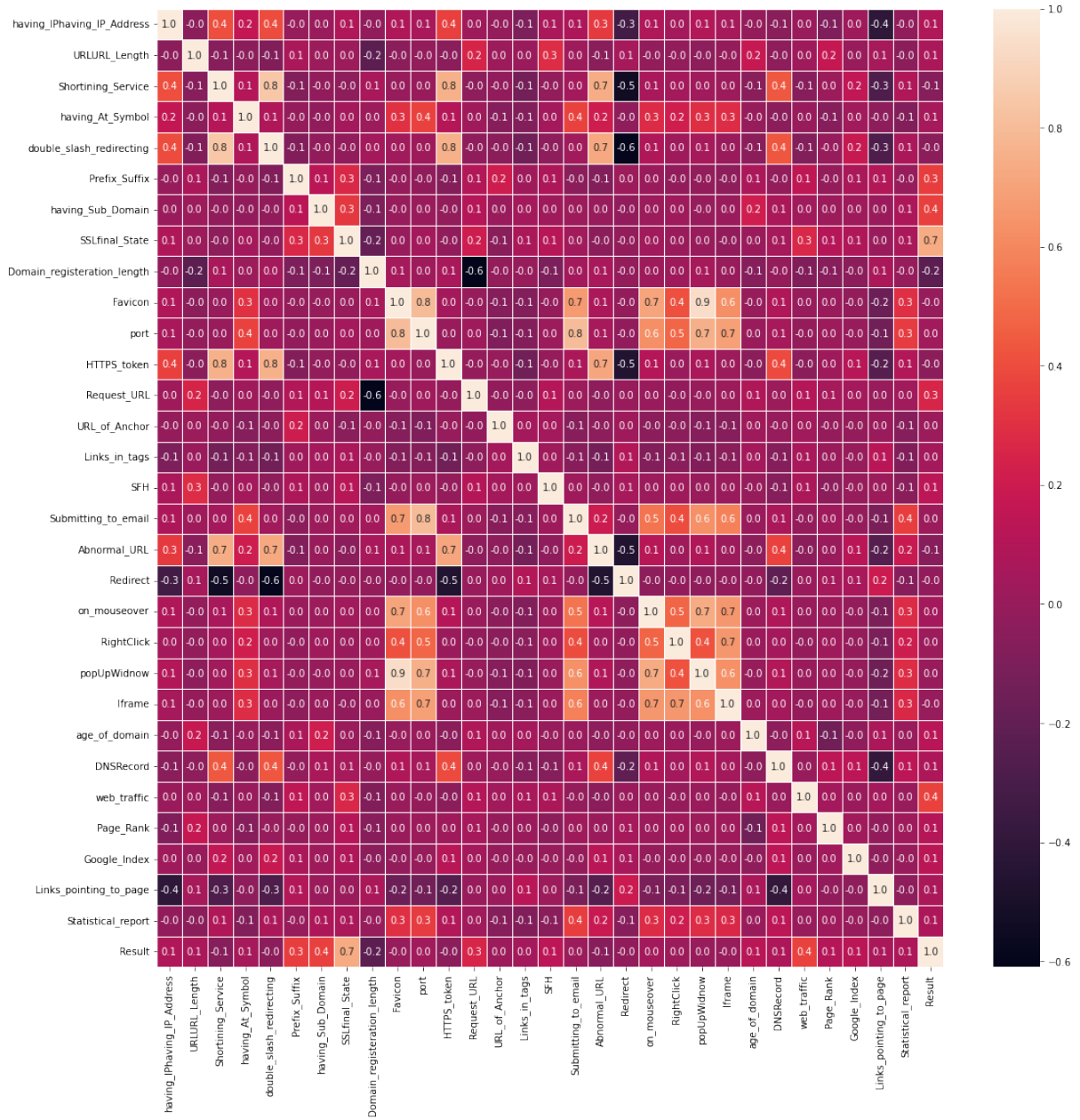

Рисунок 3.6 – Теплова карта кореляції між ознаками набору даних

3.3 Реалізація експериментального середовища

Для проведення експериментального дослідження було сформовано програмне середовище, яке забезпечує можливість виконання повного циклу роботи з даними – від їх попередньої обробки до побудови та оцінювання моделей машинного навчання. Реалізація експериментальної частини передбачає використання сучасних інструментів аналізу даних та бібліотек машинного навчання, що дозволяють ефективно працювати з табличними наборами даних, виконувати статистичний аналіз ознак, а також будувати моделі класифікації. Обране програмне середовище забезпечує зручні засоби для обробки даних, візуалізації результатів та проведення експериментів із різними алгоритмами машинного навчання.

У межах реалізації експериментального середовища було організовано послідовність виконання основних етапів дослідження, що включає підготовку набору даних, навчання моделей, тестування їх ефективності та подальший аналіз отриманих результатів. Такий підхід дозволяє систематизувати процес проведення експериментів та забезпечити відтворюваність отриманих результатів. Крім того, використання єдиного програмного середовища для реалізації всіх етапів експерименту дозволяє спростити процес порівняння результатів різних моделей та підвищити ефективність проведення дослідження.

3.3.1 Використані програмні бібліотеки та інструменти

Для реалізації експериментального дослідження було використано мову програмування Python та низку спеціалізованих бібліотек, призначених для аналізу даних, візуалізації результатів і побудови моделей машинного навчання. Використання такого програмного стеку дозволяє ефективно працювати з табличними наборами даних, виконувати статистичний аналіз характеристик та реалізовувати алгоритми

класифікації. Основні бібліотеки, що використовуються у процесі реалізації експериментального середовища, наведені у лістингу 3.3. Для роботи з даними застосовуються бібліотеки NumPy та Pandas, які забезпечують зручні інструменти для обробки числових масивів і табличних структур.

Лістинг 3.3 – Програмний код імпорту бібліотек для реалізації експериментального дослідження

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier,cv
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score, LeaveOneOut
from sklearn.model_selection import GridSearchCV, StratifiedKFold
```

Візуалізація результатів аналізу даних здійснюється за допомогою бібліотек Matplotlib та Seaborn, що дозволяють будувати графіки, гістограми та теплові карти. Для реалізації алгоритмів машинного навчання використовується бібліотека Scikit-learn,. Додатково у дослідженні використовується бібліотека XGBoost, що реалізує алгоритм градієнтного бустингу.

3.3.2 Розподіл даних на навчальну та тестову вибірки

Одним із важливих етапів підготовки даних для навчання моделей машинного навчання є розподіл набору даних на навчальну та тестову вибірки. Такий підхід дозволяє забезпечити об’єктивну оцінку ефективності побудованих моделей, оскільки навчання алгоритмів відбувається на одній частині даних, а перевірка якості прогнозування здійснюється на іншій, раніше невикористаній частині набору. Це дозволяє оцінити здатність моделі узагальнювати отримані знання та коректно працювати з новими даними.

Для реалізації розподілу набору даних використовується функція train_test_split з бібліотеки Scikit-learn. У межах дослідження використовується стандартне співвідношення, за якого 70 % даних використовується для навчання моделей, а 30 % – для їх подальшого тестування. Такий розподіл дозволяє забезпечити достатній обсяг даних для навчання моделей і водночас зберегти репрезентативну тестову вибірку для оцінювання їх ефективності. Програмна реалізація процедури розподілу набору даних наведена у лістингу 3.4.

Лістинг 3.4 – Програмний код, що реалізує розподіл набору даних на навчальну та тестову вибірки

```
    # split train - test to 70-30
    train_X,test_X,train_Y,test_Y                                    =
train_test_split(X,Y,test_size=0.3, random_state=9)
```

3.3.3 Побудова моделей машинного навчання

Наступним етапом експериментального дослідження є побудова моделей машинного навчання, які використовуються для розв’язання задачі класифікації вебресурсів. Для цього застосовуються алгоритми, що

належать до різних підходів машинного навчання, що дозволяє провести порівняльний аналіз їх ефективності. У межах дослідження реалізується універсальна функція побудови моделей, яка забезпечує можливість навчання різних алгоритмів класифікації на основі одного програмного інтерфейсу. Такий підхід спрощує процес реалізації експериментів та забезпечує однакові умови для оцінювання ефективності кожної моделі.

Реалізація процедури побудови моделей машинного навчання наведена у лістингу 3.5. У межах цієї функції передбачено вибір відповідного алгоритму класифікації залежно від переданого параметра. Після вибору моделі виконується її навчання на навчальній вибірці, сформованій на попередньому етапі дослідження. Далі здійснюється прогнозування результатів для тестової вибірки, що дозволяє оцінити якість роботи побудованої моделі. Отримані результати використовуються для розрахунку показників точності класифікації, а також для побудови матриці помилок, яка наочно демонструє правильність класифікації об’єктів різних класів.

Лістинг 3.5 – Програмний код, що реалізує побудову моделей машинного навчання та оцінювання їх ефективності

```
    #model build for different binary classification and show
confusion matrix
    def   build_model(model_name,train_X,   train_Y,   test_X,
test_Y):
        if model_name == 'LogisticRegression':
            model=LogisticRegression()
        elif model_name =='KNeighborsClassifier':
            model = KNeighborsClassifier(n_neighbors=4)
        elif model_name == 'XGBClassifier':
            model                                              =
XGBClassifier(objective='binary:logistic',eval_metric='auc')
        else:
            print('not a valid model name')
```

Продовження лістингу 3.5

```
        model=model.fit(train_X,train_Y)

        pred_prob=model.predict_proba(test_X)

        fpr, tpr, thresh = roc_curve(test_Y, pred_prob[:,1],
pos_label=1)

        model_predict= model.predict(test_X)
        acc=accuracy_score(model_predict,test_Y)
        print("Accuracy: ",acc)

        # Classification report
        print("Classification Report: ")
        print(classification_report(model_predict,test_Y))
        #print("Confusion Matrix for", model_name)
        con =  confusion_matrix(model_predict,test_Y)
        sns.heatmap(con,annot=True, fmt ='.2f')
        plt.suptitle('Confusion  Matrix  for  '+model_name,
x=0.44, y=1.0, ha='center', fontsize=25)
        plt.xlabel('Predict Values', fontsize =25)
        plt.ylabel('Test Values', fontsize =25)
        plt.show()
        return model, acc, fpr, tpr, thresh
```

Крім того, у процесі побудови моделей визначаються значення ймовірностей належності об’єктів до певного класу, що використовується для подальшого аналізу ефективності алгоритмів. На основі цих значень обчислюються параметри, необхідні для побудови ROC-кривої, яка дозволяє оцінити здатність моделі відрізняти об’єкти різних класів. Таким чином, реалізована функція дозволяє виконати повний цикл навчання та первинного оцінювання моделей машинного навчання, що є необхідним етапом проведення експериментального дослідження.

3.4 Результати дослідження

У цьому підрозділі представлено результати експериментального дослідження, спрямованого на оцінювання ефективності алгоритмів машинного навчання у задачі виявлення фішингових вебресурсів. Основною метою проведення експериментів є визначення здатності різних моделей класифікації коректно розпізнавати вебпосилання на основі їх характеристик. Для цього здійснюється аналіз результатів роботи кожного алгоритму після навчання на підготовленому наборі даних та тестування на окремій вибірці.

Оцінювання ефективності моделей здійснюється із застосуванням стандартних метрик класифікації, які дозволяють визначити якість прогнозування та здатність алгоритмів розрізняти вебресурси різних класів. Серед основних показників, що використовуються у дослідженні, можна виділити точність класифікації, матрицю помилок та ROC-криву.

У наступних підрозділах буде детально розглянуто результати роботи кожного алгоритму машинного навчання окремо. Зокрема, буде проаналізовано результати класифікації для моделей Logistic Regression, K-Nearest Neighbors та XGBoost, а також проведено їх порівняльний аналіз. Отримані результати дозволять визначити алгоритм, який демонструє найкращу ефективність у задачі автоматичного виявлення фішингових вебресурсів.

3.4.1 Результати класифікації для моделі Logistic Regression

Першим алгоритмом, який було використано у межах експериментального дослідження, є модель Logistic Regression. Цей алгоритм широко застосовується для задач бінарної класифікації та використовується як базова модель для порівняння результатів інших методів машинного навчання. Основна ідея алгоритму полягає у визначенні

ймовірності належності об’єкта до певного класу на основі лінійної комбінації вхідних ознак. У задачі виявлення фішингових вебресурсів модель Logistic Regression використовується для прогнозування належності URL-адреси до класу легітимних або фішингових вебсайтів.

Програмна реалізація навчання моделі Logistic Regression наведена у лістингу 3.6. У межах цього коду використовується раніше реалізована універсальна функція побудови моделей машинного навчання, яка дозволяє виконати навчання моделі на навчальній вибірці та оцінити її ефективність на тестових даних. Після завершення процесу навчання виконується прогнозування класів для тестової вибірки, що дозволяє визначити точність класифікації та інші показники якості моделі.

Лістинг 3.6 – Програмний код, що реалізує навчання та тестування моделі Logistic Regression

```
#Model 1 - LogisticRegression
lg_model,acc1, fpr1, tpr1, thresh1 = build_model('LogisticRegression',train_X, train_Y, test_X, test_Y.values.ravel())
```

Результати роботи моделі (рисунок 3.7) представлені у вигляді звіту класифікації, який містить основні метрики оцінювання якості моделі, зокрема precision, recall та F1-score.

Як видно з результатів, модель Logistic Regression демонструє точність класифікації приблизно 0.91, що свідчить про досить високу здатність алгоритму правильно визначати належність вебресурсів до відповідних класів.

З метою більш детального аналізу результатів роботи моделі було побудовано матрицю помилок, яка дозволяє оцінити кількість правильних та помилкових класифікацій для кожного класу.

Accuracy: 0.9092553512209828

Classification Report:

	precision	recall	f1-score	support
0	0.90	0.90	0.90	1477
1	0.92	0.92	0.92	1840
accuracy			0.91	3317
macro avg	0.91	0.91	0.91	3317
weighted avg	0.91	0.91	0.91	3317

Рисунок 3.7 – Звіт класифікації для моделі Logistic Regression

Матриця помилок (рисунок 3.8) відображає співвідношення між фактичними значеннями класів та результатами прогнозування моделі.

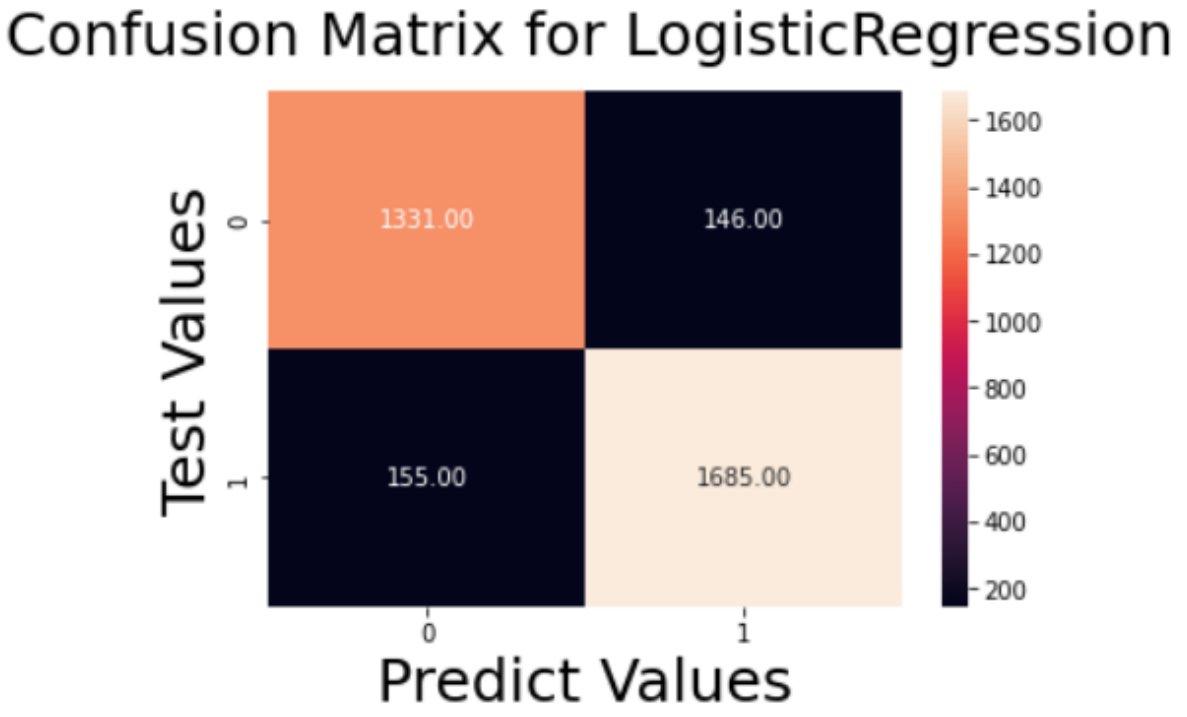

Рисунок 3.8 – Матриця помилок для моделі Logistic Regression

Аналіз отриманих результатів показує, що модель у більшості випадків правильно визначає належність вебресурсів до відповідних класів. Разом з тим, певна частина об’єктів була класифікована неправильно, що є типовим для лінійних моделей у задачах, де залежності між ознаками можуть мати складніший характер. Незважаючи на це, отримані результати свідчать про достатню ефективність моделі Logistic Regression для задачі класифікації вебресурсів.

3.4.2 Результати класифікації для моделі K-Nearest Neighbors

Другим алгоритмом, який було використано у межах експериментального дослідження, є метод K-Nearest Neighbors. Цей алгоритм належить до методів класифікації, що базуються на аналізі відстаней між об’єктами у просторі ознак.

Основна ідея методу полягає у тому, що новий об’єкт відноситься до того класу, який є найбільш поширеним серед найближчих до нього об’єктів навчальної вибірки. У задачі виявлення фішингових вебресурсів цей алгоритм використовує характеристики URL-адрес для визначення подібності між різними вебпосиланнями та прогнозування їх належності до відповідного класу.

Програмна реалізація навчання моделі K-Nearest Neighbors наведена у лістингу 3.7.

Лістинг 3.7 – Програмний код, що реалізує навчання та тестування моделі K-Nearest Neighbors

```
    # Model 2 - KNeighborsClassifier
    knn_model,acc2,      fpr2,      tpr2,      thresh2      =
build_model('KNeighborsClassifier',train_X,  train_Y,  test_X,
test_Y.values.ravel())
```

У межах цього коду виконується виклик раніше реалізованої функції побудови моделей машинного навчання, яка забезпечує навчання алгоритму на навчальній вибірці та подальше тестування його ефективності на тестових даних. Після завершення процесу навчання модель виконує прогнозування класів для тестової вибірки, що дозволяє визначити основні показники якості класифікації.

Результати роботи моделі K-Nearest Neighbors представлені у вигляді звіту класифікації (рисунок 3.9), який містить значення основних метрик оцінювання якості класифікації. Як видно з отриманих результатів, модель демонструє точність класифікації на рівні приблизно 0.94, що перевищує результат, отриманий для моделі Logistic Regression. Значення метрик precision, recall та F1-score для обох класів також свідчать про досить високу ефективність алгоритму під час визначення належності вебресурсів до класів фішингових або легітимних сайтів.

```
Accuracy:  0.9439252336448598
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.92      0.94      1554
           1       0.93      0.97      0.95      1763

    accuracy                           0.94      3317
   macro avg       0.95      0.94      0.94      3317
weighted avg       0.94      0.94      0.94      3317
```

Рисунок 3.9 – Звіт класифікації для моделі K-Nearest Neighbors

Для більш детального аналізу роботи моделі було побудовано матрицю помилок (рисунок 3.10), яка відображає співвідношення між фактичними значеннями класів та результатами прогнозування алгоритму.

Такий підхід дозволяє визначити кількість правильно класифікованих об’єктів, а також випадки помилкової класифікації.

Аналіз матриці помилок показує, що модель досить ефективно визначає належність вебресурсів до відповідних класів, демонструючи велику кількість правильних класифікацій. При цьому кількість помилкових прогнозів залишається відносно невеликою, що свідчить про високу ефективність алгоритму K-Nearest Neighbors у задачі виявлення фішингових вебресурсів.

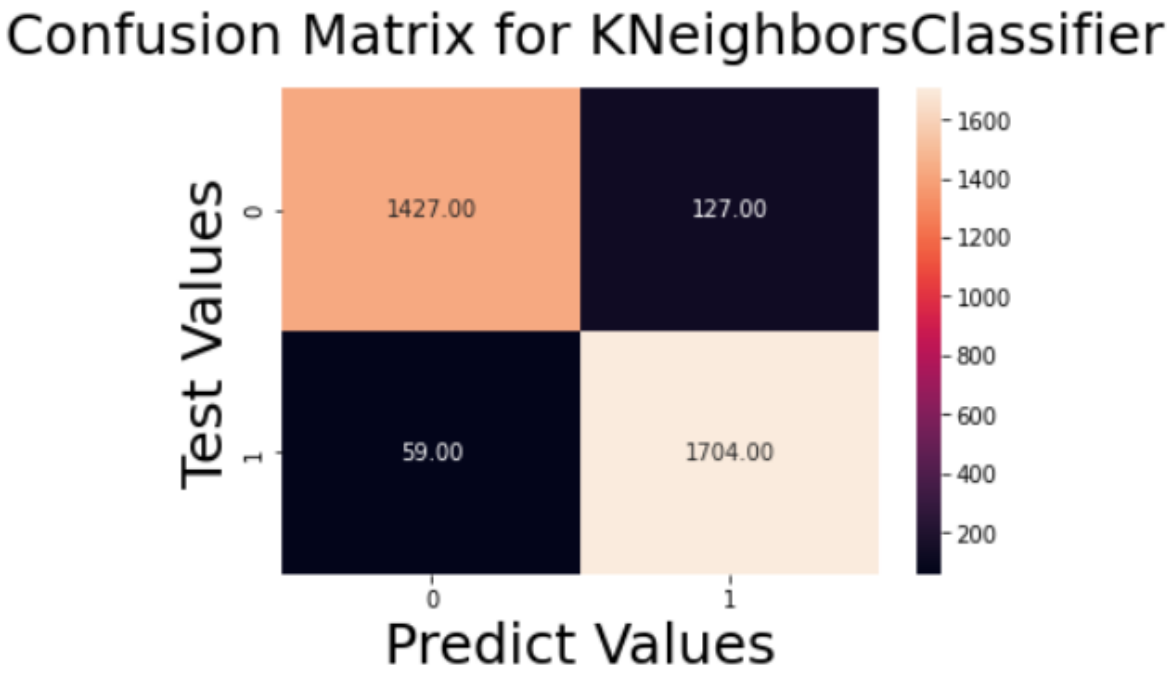

Рисунок 3.10 – Матриця помилок для моделі K-Nearest Neighbors

3.4.3 Результати класифікації для моделі XGBoost

Третім алгоритмом, що використовується у межах експериментального дослідження, є модель XGBoost. Цей алгоритм належить до ансамблевих методів машинного навчання та базується на принципі градієнтного бустингу над деревами рішень. Основна ідея методу

полягає у послідовному побудуванні набору слабких моделей, кожна з яких коригує помилки попередніх моделей. Завдяки такому підходу XGBoost здатний ефективно моделювати складні залежності у даних та демонструвати високі результати у задачах класифікації. У задачі виявлення фішингових вебресурсів цей алгоритм використовує набір характеристик URL-адрес для побудови моделі, що дозволяє прогнозувати належність вебпосилань до відповідного класу.

Програмна реалізація навчання та тестування моделі XGBoost наведена у лістингу 3.8. У межах цього коду використовується раніше розроблена функція побудови моделей машинного навчання, яка забезпечує виконання повного циклу роботи з алгоритмом. Спочатку виконується навчання моделі на навчальній вибірці, після чого здійснюється прогнозування результатів для тестових даних. На основі отриманих прогнозів розраховуються основні показники якості класифікації, що дозволяє оцінити ефективність алгоритму.

Лістинг 3.8 – Програмний код, що реалізує навчання та тестування моделі XGBoost

```
    # Model 3 - XGBClassifier
    xgb_model,     acc3,     fpr3,     tpr3,     thresh3     =
build_model('XGBClassifier',train_X,      train_Y,      test_X,
test_Y.values.ravel())
```

Результати класифікації для моделі XGBoost представлені у вигляді звіту класифікації (рисунок 3.11), який містить значення основних метрик оцінювання якості моделі. Як видно з отриманих результатів, модель демонструє точність класифікації приблизно 0.97, що є найвищим показником серед усіх досліджених алгоритмів. Значення показників precision, recall та F1-score для обох класів також свідчать про високий рівень ефективності моделі під час класифікації вебресурсів.

Accuracy: 0.9662345492915285

Classification Report:

	precision	recall	f1-score	support
0	0.96	0.96	0.96	1482
1	0.97	0.97	0.97	1835
accuracy			0.97	3317
macro avg	0.97	0.97	0.97	3317
weighted avg	0.97	0.97	0.97	3317

Рисунок 3.11 – Звіт класифікації для моделі XGBoost

Для більш детального аналізу результатів роботи алгоритму було побудовано матрицю помилок (рисунок 3.12).

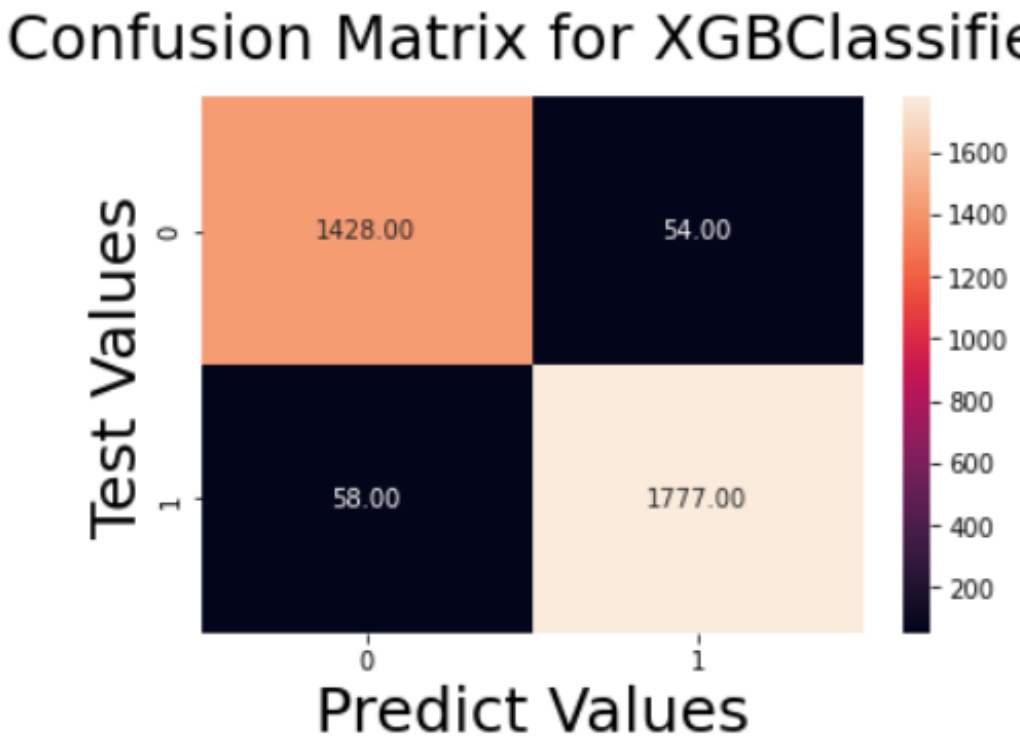

Рисунок 3.12 – Матриця помилок для моделі XGBoost

Матриця помилок дозволяє оцінити, наскільки ефективно модель розпізнає вебресурси різних класів та визначити кількість випадків помилкової класифікації. Аналіз отриманих результатів показує, що модель XGBoost демонструє найкращу ефективність серед досліджених алгоритмів. Кількість правильно класифікованих вебресурсів є найбільшою, тоді як кількість помилкових прогнозів є мінімальною. Це свідчить про високу здатність алгоритму виявляти фішингові вебресурси та підтверджує доцільність використання ансамблевих методів машинного навчання у задачах кібербезпеки.

3.4.4 Порівняльний аналіз результатів моделей

Після отримання результатів класифікації для кожної з досліджених моделей було проведено їх порівняльний аналіз з метою визначення алгоритму, який демонструє найкращу ефективність у задачі виявлення фішингових вебресурсів. Для цього використовуються метрики класифікації, отримані на попередніх етапах дослідження, а також результати графічного аналізу якості моделей. Порівняння результатів дозволяє оцінити здатність алгоритмів коректно розпізнавати об’єкти різних класів та визначити їх загальну ефективність під час роботи з набором даних.

Одним із інструментів такого аналізу є ROC-крива (Receiver Operating Characteristic), яка дозволяє оцінити здатність моделі відрізняти об’єкти різних класів. ROC-крива будується на основі співвідношення між показниками True Positive Rate та False Positive Rate для різних порогових значень класифікації. Чим ближче крива до верхнього лівого кута координатної площини, тим вищою є якість моделі. Побудова ROC-кривих для всіх досліджених моделей дозволяє виконати їх наочне порівняння та визначити алгоритм, який демонструє найкращу здатність до класифікації вебресурсів.

Програмна реалізація побудови ROC-кривих для моделей Logistic Regression, K-Nearest Neighbors та XGBoost наведена у лістингу 3.9. У межах цього коду обчислюються значення показників True Positive Rate та False Positive Rate для кожної моделі, після чого будується графічне представлення відповідних ROC-кривих. Такий підхід дозволяє виконати візуальне порівняння ефективності моделей та визначити алгоритм, який забезпечує найкращі результати класифікації.

Лістинг 3.9 – Програмний код, що реалізує побудову ROC-кривих для порівняння моделей машинного навчання

```
    # roc curve for tpr = fpr
    random_probs = [0 for i in range(len(test_Y))]
    p_fpr, p_tpr, _ = roc_curve(test_Y, random_probs,
pos_label=1)
    plt.style.use('seaborn')

    # plot roc curves
    plt.plot(fpr1, tpr1, linestyle='--',color='orange',
label='Logistic Regression')
    plt.plot(fpr2, tpr2, linestyle='--',color='green',
label='KNN')
    plt.plot(fpr3, tpr3, linestyle='--',color='red',
label='XGBClassifier')
    plt.plot(p_fpr, p_tpr, linestyle='--', color='blue')
    # title
    plt.title('ROC curve')
    # x label
    plt.xlabel('False Positive Rate')
    # y label
    plt.ylabel('True Positive rate')
    plt.legend(loc='best')
    plt.savefig('ROC',dpi=300)
    plt.show()
```

Результати побудови ROC-кривих (рисунок 3.13) для досліджених моделей наведено нижче. Аналіз отриманого графіка показує, що всі моделі демонструють досить високі результати класифікації, оскільки їх ROC-криві знаходяться значно вище за діагональну лінію випадкового прогнозування.

Разом з тим, крива моделі XGBoost розташована найближче до верхньої межі графіка, що свідчить про її найкращу здатність правильно класифікувати вебресурси серед усіх розглянутих алгоритмів. Модель K-Nearest Neighbors також демонструє високі результати, тоді як Logistic Regression має дещо нижчі показники ефективності порівняно з іншими алгоритмами.

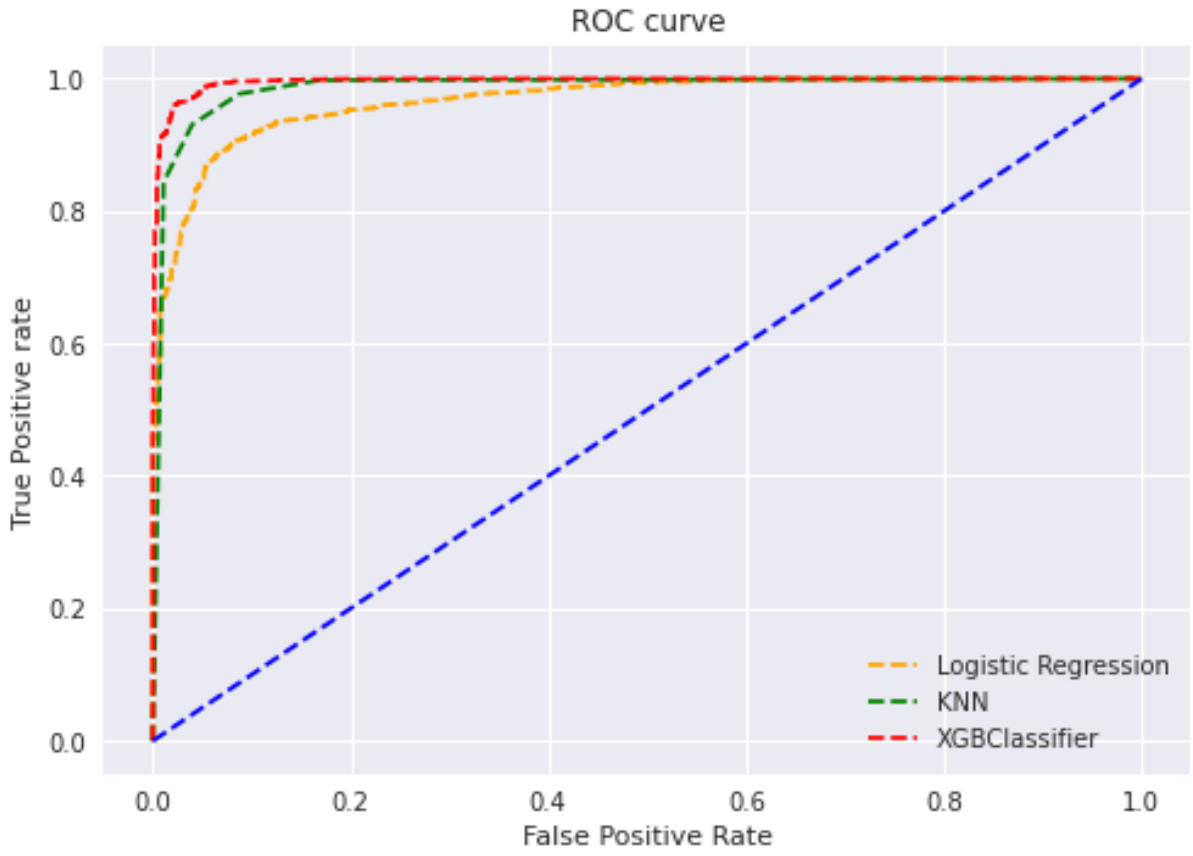

Рисунок 3.13 – ROC-криві для моделей Logistic Regression, K-Nearest Neighbors та XGBoost

Для підвищення достовірності отриманих результатів та перевірки стабільності роботи моделей у дослідженні додатково застосовуються методи крос-валідації. Використання крос-валідації дозволяє оцінити здатність моделей узагальнювати результати на різних підмножинах даних та зменшити ризик перенавчання. У межах даного дослідження застосовуються методи K-Fold та StratifiedKFold, які забезпечують багаторазове розбиття навчального набору даних на декілька підвибірок. Під час кожної ітерації одна частина даних використовується для тестування моделі, тоді як інші підвибірки застосовуються для її навчання.

Крім перевірки стабільності результатів, важливим етапом є підбір оптимальних гіперпараметрів моделей машинного навчання. Для цього використовується метод GridSearchCV, який дозволяє автоматично перебирати різні комбінації параметрів алгоритмів та визначати їх оптимальні значення на основі результатів крос-валідації. У межах дослідження для моделі Logistic Regression підбираються параметри solver, C та penalty, що визначають особливості оптимізації моделі та рівень регуляризації. Для алгоритму K-Nearest Neighbors підбір гіперпараметрів здійснюється для параметрів кількості сусідів, метрики відстані та вагових коефіцієнтів. Реалізацію процедур крос-валідації та підбору гіперпараметрів наведено у лістингу 3.10.

Лістинг 3.10 – Програмний код, що реалізує крос-валідацію та підбор гіперпараметрів моделей машинного навчання

```
    # Create the parameter grid based on the results of random
search
    param_grid = {
        'solver':['liblinear','newton-cg'],
        'C': [0.01,0.1,1,10,100],
        'penalty': ["l1","l2"]
    }
    # Instantiate the grid search model
```

Продовження лістингу 3.10

```
    grid_search = GridSearchCV(estimator = LogisticRegression()
, param_grid = param_grid,
    cv = StratifiedKFold(4), n_jobs = -1, verbose = 1, scoring
= 'accuracy' )

    grid_search.fit(train_X,train_Y.values.ravel())
    grid_params = {
        'n_neighbors':[3,4,11,19],
        'weights':['uniform','distance'],
        'metric':['euclidean','manhattan']
    }
    gs= GridSearchCV(
    KNeighborsClassifier(),
    grid_params,
    verbose=1,
    cv=3,
    n_jobs=-1
    )
    gs_results = gs.fit(train_X,train_Y.values.ravel())
    xgb_cv                                                    =
XGBClassifier(n_estimators=100,objective='binary:logistic',eva
l_metric='auc')
    scores      =      cross_val_score(xgb_cv,      train_X,
train_Y.values.ravel(), cv=10, scoring = "accuracy")
    print("Scores:", scores)
    print("Mean:", scores.mean())
    print("Standard Deviation:", scores.std())
    results=pd.DataFrame({'Model':['LogisticRegression','KNN',
'XGBoost'],'Accuracy Score':[acc1,acc2,acc3]})
    result_df=results.sort_values(by='Accuracy         Score',
ascending=False)
    result_df=result_df.set_index('Model')
    result_df
```

Після виконання експериментів та отримання результатів класифікації для кожної моделі було сформовано узагальнену таблицю порівняння ефективності алгоритмів. У таблиці 3.1 наведено значення точності класифікації для моделей Logistic Regression, K-Nearest Neighbors та XGBoost. Порівняння отриманих результатів дозволяє визначити алгоритм, який демонструє найкращі показники під час розв'язання задачі виявлення фішингових вебресурсів.

Таблиця 3.1 – Порівняння точності моделей машинного навчання

Модель	Точність
XGBoost (XGB)	0.966235
K-Nearest Neighbors (KNeighbors)	0.943925
Logistic Regression	0.909255

Аналіз результатів показує, що найвищу точність класифікації демонструє модель XGBoost. Це пояснюється використанням ансамблевого підходу, який дозволяє ефективно поєднувати результати великої кількості дерев рішень та краще враховувати складні залежності між ознаками набору даних. Модель K-Nearest Neighbors також демонструє досить високий рівень точності, однак поступається XGBoost. У свою чергу модель Logistic Regression показує дещо нижчі результати, що пояснюється її лінійною п

ВИСНОВКИ

У межах виконання кваліфікаційної роботи було проведено дослідження методів машинного навчання для задачі автоматичного виявлення фішингових вебресурсів. У роботі було виконано аналіз предметної галузі, досліджено сучасні підходи до виявлення фішингових вебсайтів, а також розглянуто можливості застосування алгоритмів машинного навчання для розв’язання даної задачі.

У теоретичній частині роботи було проаналізовано основні підходи до класифікації вебресурсів, зокрема методи машинного навчання, що використовуються для виявлення фішингових посилань. Було розглянуто принципи роботи алгоритмів Logistic Regression, K-Nearest Neighbors та XGBoost, а також досліджено методи оцінювання ефективності моделей машинного навчання. Проведений аналіз дозволив визначити найбільш доцільні алгоритми для подальшого експериментального дослідження та сформувати план проведення експериментів.

У межах експериментальної частини роботи було підготовлено набір даних, що містить характеристики URL-адрес, які можуть використовуватися для визначення належності вебресурсів до фішингових або легітимних сайтів. Було виконано попередній аналіз структури набору даних, досліджено розподіл ознак та виконано попередню обробку даних, зокрема аналіз кореляції між параметрами та видалення сильно корельованих характеристик. Такий підхід дозволив зменшити надлишковість даних та підвищити ефективність подальшого навчання моделей машинного навчання.

Під час проведення експериментального дослідження було реалізовано програмне середовище для побудови та тестування моделей машинного навчання. Було виконано розподіл набору даних на навчальну та тестову вибірки, реалізовано процедури навчання моделей та проведено оцінювання їх ефективності за допомогою відповідних метрик класифікації.

У дослідженні було застосовано три алгоритми машинного навчання: Logistic Regression, K-Nearest Neighbors та XGBoost.

За результатами експериментального дослідження було отримано кількісні показники ефективності кожної моделі. Модель Logistic Regression продемонструвала точність класифікації близько 0,91, що свідчить про достатню ефективність базових лінійних моделей у задачі виявлення фішингових вебресурсів. Алгоритм K-Nearest Neighbors показав вищу точність класифікації, яка становила приблизно 0,94. Найкращі результати було отримано при використанні моделі XGBoost, точність якої склала приблизно 0,97. Отримані результати свідчать про високу ефективність ансамблевих методів машинного навчання у задачах класифікації вебресурсів.

Крім оцінювання точності класифікації, у роботі було проведено додатковий аналіз ефективності моделей за допомогою матриці помилок та ROC-кривих. Проведений аналіз дозволив підтвердити, що модель XGBoost демонструє найкращу здатність до розпізнавання фішингових вебресурсів та має найвищі показники якості класифікації серед досліджених алгоритмів. Для перевірки стабільності отриманих результатів також було застосовано методи крос-валідації, зокрема K-Fold та StratifiedKFold, а також виконано підбір оптимальних гіперпараметрів моделей за допомогою методу GridSearchCV.

Отримані результати підтверджують, що використання алгоритмів машинного навчання є ефективним підходом для автоматичного виявлення фішингових вебресурсів. Застосування ансамблевих методів, зокрема XGBoost, дозволяє досягти високої точності класифікації та підвищити ефективність систем виявлення кіберзагроз. Подальші дослідження у цьому напрямку можуть бути спрямовані на розширення набору використовуваних ознак, що описують характеристики вебресурсів, а також на використання більш складних моделей машинного навчання, зокрема методів глибокого навчання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Comparative analysis of machine learning algorithms for phishing website detection / Emmanuel E. J. *World Journal of Advanced Research and Reviews.* 2025. Vol. 27, no. 3. P. 889–896. URL: https://doi.org/10.30574/wjarr.2025.27.3.3216 (date of access: 02.02.2026).

2. Chiew K. L., Yong K. S. C., Tan C. L. A survey of phishing attacks: their types, vectors and technical. *Expert Systems with Applications*. 2018. Vol. 106. P. 1–20. URL: https://doi.org/10.1016/j.eswa.2018.03.050 (date of access: 03.02.2026).

3. Jayasurya S. R. Phishing detection using machine learning techniques. *International Journal on Science and Technology.* 2025. Vol. 16, no. 2. P. 1–16. URL: https://www.ijsat.org/papers/2025/2/3400.pdf (date of access: 04.02.2026).

4. Aggarwal C. C. Data mining. Cham: Springer International Publishing, 2015. URL: https://doi.org/10.1007/978-3-319-14142-8 (date of access: 09.04.2026).

5. Aggarwal C. C. Outlier analysis. New York: Springer, 2017. 466 p.

6. Bishop C. M. Pattern recognition and machine learning. New York: Springer, 2006. 738 p.

7. Breiman L. Random forests. *Machine Learning*. 2001. Vol. 45, no. 1. P. 5–32.

8. Chen T., Guestrin C. XGBoost: a scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016. P. 785–794. URL: https://doi.org/10.1145/2939672.2939785 (date of access: 05.02.2026).

9. Cortes C., Vapnik V. Support-vector networks. *Machine Learning*. 1995. Vol. 20, no. 3. P. 273–297.

10. Géron A. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. Sebastopol: O'Reilly Media, 2019. 819 p.

11. Han J., Kamber M., Pei J. Data mining: concepts and techniques. Boston: Morgan Kaufmann, 2011. 744 p.

12. McKinney W. Data structures for statistical computing in Python. *Proceedings of the 9th Python in Science Conference*. 2010. P. 56–61. URL: https://doi.org/10.25080/Majora-92bf1922-00a (date of access: 07.02.2026).

13. Scikit-learn: machine learning in Python / F. Pedregosa et al. *JMLR*. 2011. Vol. 12. P. 2825–2830. URL: https://jmlr.org/papers/v12/pedregosa11a.html (date of access: 08.02.2026).

14. Python software foundation. *Python documentation.* URL: https://docs.python.org (date of access: 09.02.2026).

15. Meenu, Godara S. Phishing detection using machine learning techniques. *International Journal of Engineering and Advanced Technology*. 2019. Vol. 9, no. 2. P. 3820–3829. URL: https://doi.org/10.35940/ijeat.B4095.129219 (date of access: 09.05.2026).

16. Sahoo D., Liu C., Hoi S. C. H. Malicious URL detection using machine learning: a survey. 2017. URL: https://doi.org/10.48550/arXiv.1701.07179 (date of access: 09.05.2026).

17. Scikit-learn developers. *Scikit-learn documentation.* URL: https://scikit-learn.org/stable (date of access: 11.02.2026).

18. Seaborn statistical data visualization library. *Seaborn Documentaion.* URL: https://seaborn.pydata.org (date of access: 12.02.2026).

19. A scalable URL-based phishing websites detection system using nachine learning techniques / M. N. Deepthi et al. *Human-Centric Smart Computing*. Singapore, 2024. P. 167–180. URL: https://doi.org/10.1007/978-981-99-7711-6_14 (date of access: 09.05.2026).

20. McKinney W. Python for data analysis. O'Reilly Media, Incorporated, 2022.