Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

“Інформаційна технологія аналізу та передбачення фішингових сайтів методами машинного навчання”

Виконав: студент 2 курсу, групи 2ІСТ-24м
спеціальності 126 – «Інформаційні системи та технології»
_____ Данило ЛИТВИНЕНКО
Керівник: к.т.н., доц. каф. САІТ
_____ Олексій КОЗАЧКО
«27» ____11____ 2025 р.
Опонент: доц. каф. КН
_____ Володимир ОЗЕРАНСЬКИЙ
«03» ____12____ 2025 р.

Допущено до захисту
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«28» ____11____ 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – другий (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 126 Інформаційні системи та технології
Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«25» 09 2025 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Литвиненку Данилу Олександровичу

1. Тема роботи: "Інформаційна технологія аналізу та передбачення фішингових сайтів методами машинного навчання",
керівник роботи: Олексій КОЗАЧКО, к.т.н., доц. каф. САІТ,
затверджені наказом ВНТУ від «24» 09 2025 року №313
2. Строк подання студентом роботи «28» 11 2025 року
3. Вихідні дані до роботи:
Набір даних фішингових сайтів Kaggle Dataset „Web Page Phishing Dataset" https://www.kaggle.com/datasets/danielfernandon/web-page-phishing-dataset.

4. Зміст текстової частини:

- загальна характеристика об'єкту досліджень;
- основні етапи виконання роботи та огляд набору вхідних даних;
- розроблення інформаційної технології для передбачення фішингових сайтів;
- економічна частина.

5. Перелік ілюстративного матеріалу:
- огляд даних датасету;
- візуалізація даних датасету;
- розвідувальний аналіз даних;
- блок-схема алгоритму інформаційної технології;
- результати.

6. Консультанти розділів МКР

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Олександр ЛЕСЬКО, к. е. н., проф. каф. ЕПВМ	*(дата і підпис)*	*(дата і підпис)*
		05.11.2025	15.11.2025

7. Дата видачі завдання «25» 09 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз сучасного стану моніторингу фішингових сайтів	15.09.2025	25.09.2025	виконано
2	Основні етапи виконання роботи та огляд набору вхідних даних	25.09.2025	05.10.2025	виконано
3	Обробка результатів прогнозування та візуалізація даних у середовищі Kaggle	05.10.2025	25.10.2025	виконано
4	Розробка інформаційної технології	25.10.2025	05.11.2025	виконано
5	Економічна частина	05.11.2025	15.11.2025	виконано
6	Оформлення матеріалів до захисту МКР	15.11.2025	25.11.2025	виконано

Студент Данило ЛИТВИНЕ

Керівник роботи Олексій КОЗАЧКО

АНОТАЦІЯ

Литвиненко Д.О. Інформаційна технологія аналізу та передбачення фішингових сайтів методами машинного навчання. Магістерська кваліфікаційна робота зі спеціальності 126 – інформаційні системи та технології, освітньо-професійна програма – інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2025. 107 с.

На укр. мові. Бібліогр.: 20 назв; рис.: 68; табл.: 11.

У магістерській кваліфікаційній роботі проаналізовано та описано об’єкт дослідження предметної області, а саме проблема фішингового шахрайства в Інтернеті, її основні види та обґрунтовано вибір оптимальних методів. Для розв’язання поставлених задач обрано мову програмування Python та такі методи машинного навчання як Logistic Regression, Decission Tree, Random forest, Gradient boosting trees, XGBoost Model, LightGBM Model, ExtraTrees Model, K-Nearest Neighbors Model (KNN). Для розпізнавання фішингових сайтів у рамках машинного навчання також використовувались нейронні мережі, зокрема багатошаровий перцептрон Multilayer Perceptron MLP.

Ілюстративна частина складається з 7 плакатів із результатами аналізу та передбачення.

У розділі економічної частини розглянуто питання про економічну ефективність розробки та впровадження інформаційної технології передбачення фішингових сайтів. Проведено порівняльний аналіз витрат на створення системи протидії фішингу методами машинного навчання порівняно з потенційними фінансовими втратами для бізнесу та користувачів від успішних фішингових атак.

Ключові слова: інформаційна технологія, фішинг, машинне навчання, нейронні мережі, кібербезпека, передбачення, веб-сайти, аналіз даних.

ABSTRACT

Lytvynenko, D.O. Information technology for analyzing and predicting phishing sites using machine learning methods. Master's thesis in the field of 126 – Information Systems and Technologies, educational and professional program – Information Technologies for Data and Image Analysis. Vinnytsia: VNTU, 2025. 107 p.

In Ukrainian. Bibliography: 20 titles; figures: 68; tables: 11.

The master's thesis analyzes and describes the subject area of the research, namely the problem of phishing fraud on the Internet, its main types, and justifies the choice of optimal methods. To solve the tasks set, the Python programming language and such machine learning methods as Logistic Regression, Decision Tree, Random Forest, Gradient Boosting Trees, XGBoost Model, LightGBM Model, ExtraTrees Model, K-Nearest Neighbors Model (KNN). To recognize phishing sites in the context of machine learning, neural networks were also used, in particular, the Multilayer Perceptron MLP.

The illustrative part consists of 7 posters with the results of analysis and prediction.

The economic section examines the economic efficiency of developing and implementing information technology for predicting phishing sites. A comparative analysis of the costs of creating a phishing countermeasure system using machine learning methods is conducted in comparison with the potential financial losses for businesses and users from successful phishing attacks.

Keywords: information technology, phishing, machine learning, neural networks, cybersecurity, prediction, websites, data analysis.

ЗМІСТ

ВСТУП 4
1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ОБ'ЄКТУ ДОСЛІДЖЕНЬ 7
1.1 Аналіз предметної області 7
1.2 Види та підходи до виявлення фішингових сайтів 14
1.3 Вибір мови програмування та середовища розробки 19
1.4 Опис моделей, які будуть використовуватись для передбачення даних .. 21
1.5 Висновки 31
2 ОСНОВНІ ЕТАПИ ВИКОНАННЯ РОБОТИ ТА ОГЛЯД НАБОРУ ВХІДНИХ ДАНИХ 32
2.1 Характеристика датасету для аналізу фішингових сайтів 32
2.2 Розвідувальний аналіз даних фішингових сайтів 36
2.3 Висновки 42
3 РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ ПЕРЕДБАЧЕННЯ ФІШИНГОВИХ САЙТІВ 43
3.1 Моделювання та передбачення 43
3.2 Вибір оптимальних моделей 90
3.3 Висновки 93
4 ЕКОНОМІЧНА ЧАСТИНА 94
4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки 94
4.2 Розрахунок узагальненого коефіцієнта якості розробки 95
4.3 Розрахунок витрат на проведення науково-дослідної роботи 96
4.3.1 Витрати на оплату праці 97
4.3.2 Відрахування на соціальні заходи 100
4.3.3 Сировина та матеріали 100
4.3.4 Розрахунок витрат на комплектуючі 101
4.3.5 Спецустаткування для наукових (експериментальних) робіт 102
4.3.6 Програмне забезпечення для наукових (експериментальних) робіт 103
4.3.7 Амортизація обладнання, програмних засобів та приміщень 104
4.3.8 Паливо та енергія для науково-виробничих цілей 105

4.3.9 Службові відрядження 106
4.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації 107
4.3.11 Інші витрати 107
4.3.12 Накладні (загальновиробничі) витрати 107
4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором 108
4.5 Висновки 113
ВИСНОВКИ 114
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 115
Додаток А. Технічне завдання 118
Додаток Б. Протокол перевірки кваліфікаційної роботи 121
Додаток В. Лістинг програми 122
Додаток Г. Ілюстративна частина 127

ВСТУП

Актуальність теми. Актуальність дослідження фішингових сайтів зумовлена тим, що ця проблема є однією з найгостріших у сучасному цифровому середовищі, що постійно ускладнюється та зростає. Незважаючи на загальне підвищення рівня обізнаності користувачів про кіберзагрози, зловмисники постійно вдосконалюють методи соціальної інженерії та створюють дедалі переконливіші підроблені веб-ресурси, що суттєво ускладнює їх своєчасне виявлення.

Ключовим фактором, який використовують кіберзлочинці, залишається людський фактор. Недбалість, неуважність або недостатня поінформованість роблять користувачів вразливими до фішингових атак. Навіть найсучасніші системи безпеки не завжди можуть повністю запобігти помилкам користувачів, які стають жертвами шахраїв, розкриваючи конфіденційні дані - паролі, фінансові відомості та особисту інформацію.

Фішинг – це поширена форма інтернет-шахрайства, за якої зловмисники, використовуючи підроблені веб-сайти, електронні листи та повідомлення, обманом змушують жертв розкрити конфіденційну інформацію. З огляду на зростаючу поширеність і складність цих загроз, розробка ефективних методів автоматичного виявлення фішингових сайтів є критично важливою для забезпечення надійного рівня безпеки в Інтернеті.

Саме тому в роботі пропонується застосувати методи машинного навчання для створення інформаційної технології, яка здатна ідентифікувати фішингові ресурси з високою точністю, мінімізуючи ризик, спричинений людським фактором.

Мета і завдання роботи. Метою дослідження є підвищення точності передбачення фішингових веб-сайтів за рахунок використання методів машинного навчання.

Для досягнення поставленої мети необхідно виконати такі задачі:

– Провести аналіз проблеми фішингових атак та сучасних методів їх

виявлення;

– Визначити оптимальні інформаційні технології та моделі машинного навчання для вирішення задачі;

– Розробити інформаційну технологію для аналізу та передбачення фішингових сайтів з використанням методів машинного навчання;

– Оцінити ефективність розробленої технології на основі тестових даних.

Об’єктом дослідження магістерської кваліфікаційної роботи є процес розробки інформаційної технології для передбачення фішингових веб-сайтів.

Предметом дослідження магістерської кваліфікаційної роботи є технологія для аналізу та передбачення фішингових сайтів.

Методи дослідження. У дослідженнях використовувались методи та моделі машинного навчання, включаючи нейронні мережі для передбачення фішингових сайтів. Також здійснено аналіз даних у середовищі Kaggle, використовуючи мову програмування Python.

Новизна одержаних результатів. Дістала подальший розвиток інформаційна технологія аналізу та передбачення фішингових сайтів шляхом застосування таких моделей машинного навчання як XGBoost Model, LightGBM Model, ExtraTrees Model та K-Nearest Neighbors Model (KNN), що дозволило підвищити точність ідентифікації підозрілих сайтів за рахунок оптимізації параметрів вищезгаданих моделей.

Практичне значення роботи сприятиме покращенню рівня кібербезпеки користувачів та організацій у мережі Інтернет. Результати роботи мають цінність для автоматизації процесів виявлення фішингових загроз, можуть бути впроваджені в існуючі системи безпеки

Результати магістерської кваліфікаційної роботи. Результати роботи доповідались на всеукраїнській науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця, 2025-2026 рр.).

Публікації результатів магістерської кваліфікаційної роботи. Опубліковано тези на Всеукраїнській науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця, 2025-2026 рр.) [1].

1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ОБ'ЄКТУ ДОСЛІДЖЕНЬ

1.1 Аналіз предметної області

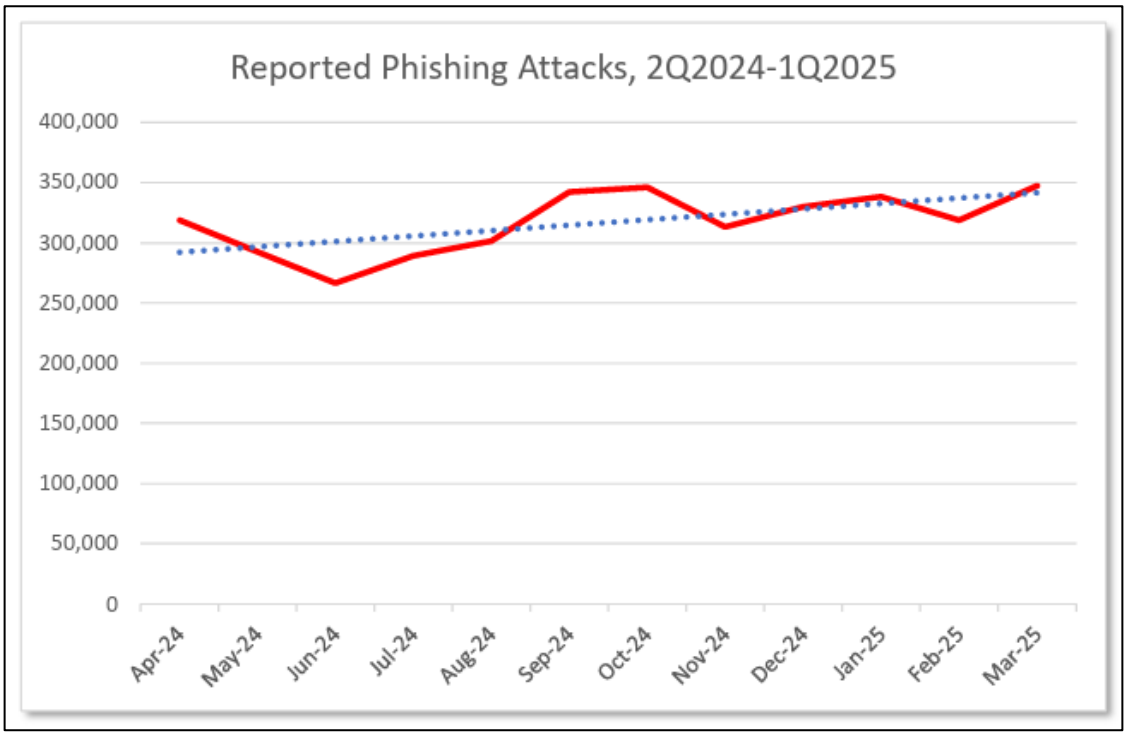
В умовах науково-технічного прогресу та інтенсивного розвитку цифрових сервісів, інформація набуває статусу ключового ресурсу, а її обробка та використання відбувається в процесі комп'ютеризації різних галузей людської діяльності. Однак використання сучасних інформаційних технологій, крім позитивних аспектів, призвело до інтенсивного використання комп'ютерних технологій у корисливих цілях, особливо в кредитній і банківській діяльності, що спричинило появу нового класу злочинів – комп'ютерної злочинності [2]. Інтернет-шахрайство відноситься до будь-якого виду шахрайства, при якому елементи Інтернету (веб-сайти, електронна пошта, чати) використовуються для залучення потенційних жертв або здійснення шахрайських операцій. Дані, накопичені Інтерполом, підтверджують, що World Wide Web став зоною дуже активного зростання злочинності [3]. Одним із найпоширеніших і найнебезпечніших видів кіберзлочинів є фішинг (Phishing) - шахрайська схема, що використовує соціальну інженерію та технічні хитрощі для викрадення персональних даних, таких як паролі, номери банківських карток чи облікові дані. Зловмисники створюють підроблені веб-сайти, які імітують добре відомі та надійні ресурси, максимально копіюючи дизайн та URL-адресу, додаючи лише 1 або 2 символи, що робить різницю майже непомітною для неозброєного ока [4].

Щоб зрозуміти, що таке фішинговий сайт, досить згадати велику кількість спам-листів, в яких обіцяються призи, значні знижки, або пропонується підтвердити схвалення чи відписатися від спаму, як тільки користувач перейде за посиланням і введе свої особисті дані. Така схема здійснюється за допомогою електронних повідомлень, телефонних дзвінків (вішинг), а також онлайн-реклами. Навіть в Україні зафіксована нова форма мобільного фішингу, коли шахраї пропонують впровадити спеціальний набір

USSD-команд, що призводить до зняття грошей з рахунків довірливих абонентів. Особливої актуальності ці схеми набувають в умовах воєнного стану, коли зловмисники використовують емоційну вразливість, створюючи фейкові збори на підтримку ЗСУ чи допомогу постраждалим [5].

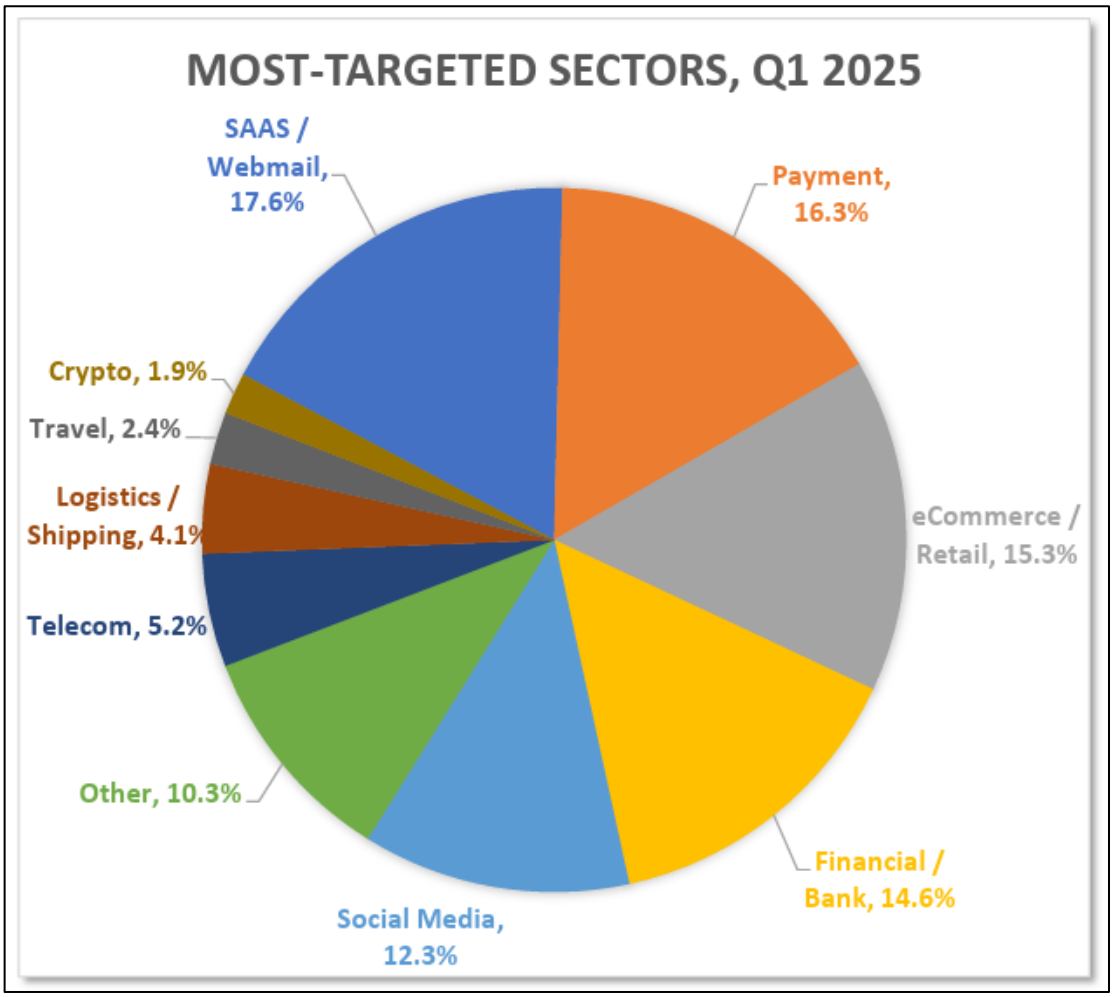
Anti-Phishing Working Group (APWG), спираючись на дані від своїх членів та громадськості, аналізує постійно мінливу природу та техніки кіберзлочинності. APWG відстежує унікальні фішингові сайти (атаки), що є основним показником фішингу в усьому світі, унікальні теми фішингових електронних листів для оцінки різноманітності атак, а також кількість атакованих брендів [6].

У I кварталі 2025 року APWG зафіксувала 1 003 924 фішингові атаки. Це був найбільший квартальний показник з IV кварталу 2023 року (1,07 мільйона) і він демонструє стійке зростання протягом останнього року, піднявшись з 877 536 у Q2 2024 до 989 123 у Q4 2024. У розбивці за місяцями унікальних фішингових веб-сайтів (атак) було виявлено 337 998 у січні, 318 513 у лютому та 347 413 у березні. Кількість атакованих брендів коливалася близько 312–328 щомісяця [7]. На рисунку 1.1 представлено графік фішингових атак 2024-2025 років.



Риснуок 1.1 – Графік фішингових атак

У І кварталі 2025 року сектор SAAS/Webmail (Програмне забезпечення як послуга / Вебпошта) залишався найбільш атакованим, проте його частка знизилася до 17,6% порівняно з 23,3% у Q4 2024. Атаки на онлайн-платіжні компанії та фінансовий банківський сектори зросли, загалом склавши 30,9% від усіх атак. Як зазначають експерти OpSec Security, шахраї продовжують розширювати типи компаній, які вони імітують, включаючи громадські комунальні послуги, паркомати та системи стягнення плати за проїзд. Компанія OpSec Security також відзначила зростання обсягів голосового фішингу та фішинг через SMS. На рисунку 1.2 представлено графік секторів фішингових атак першого кварталу 2025 року.



Риснуок 1.2 – Сектори фішингових атак

Зловмисники активно використовують QR-коди у своїх електронних листах, які, при скануванні мобільним телефоном, перенаправляють користувачів на фішингові веб-сайти або змушують завантажувати шкідливе ПЗ, оскільки ці коди часто не виявляються традиційними фільтрами. За даними Mimecast, за шість місяців Q4 2024 та Q1 2025 було виявлено понад 1,7 мільйона унікальних шкідливих QR-кодів. Для створення QR-кодів люди використовують генератори QR-кодів. Це комерційні онлайн-сервіси. Різні легальні компанії та організації використовують їх для створення QR-кодів для реклами та заходів. Злочинці також використовують ці генератори.

Генератори QR-кодів пропонують різні функції, якими можуть скористатися злочинці:

– Деякі генератори QR-кодів вимагають передплати, інші є

безкоштовними. Безкоштовні сервіси, природно, як правило, виділяють менше ресурсів на запобігання та припинення зловмисного використання.

– Багато генераторів QR-кодів пропонують функцію відстеження – вони дозволяють своїм клієнтам бачити, скільки разів і коли QR-код був відсканований, а також загальне місцезнаходження інтернет-користувачів, які сканують коди. Злочинці використовують відстеження для оптимізації своїх кампаній.

– Деякі генератори QR-кодів дозволяють своїм клієнтам змінювати URL-адресу призначення QR-коду після його генерації. Це зручна функція, яку злочинці використовують, намагаючись обдурити компанії з безпеки та уникнути виявлення.

Злочинці також направляли QR-коди на сервіси скорочення URL-адрес, які потім перенаправляли користувачів на інші URL-адреси призначення. Це інструмент для приховування зловмисного характеру QR-кодів.

Mimecast визначив, які генератори QR-кодів найчастіше використовували злочинці:

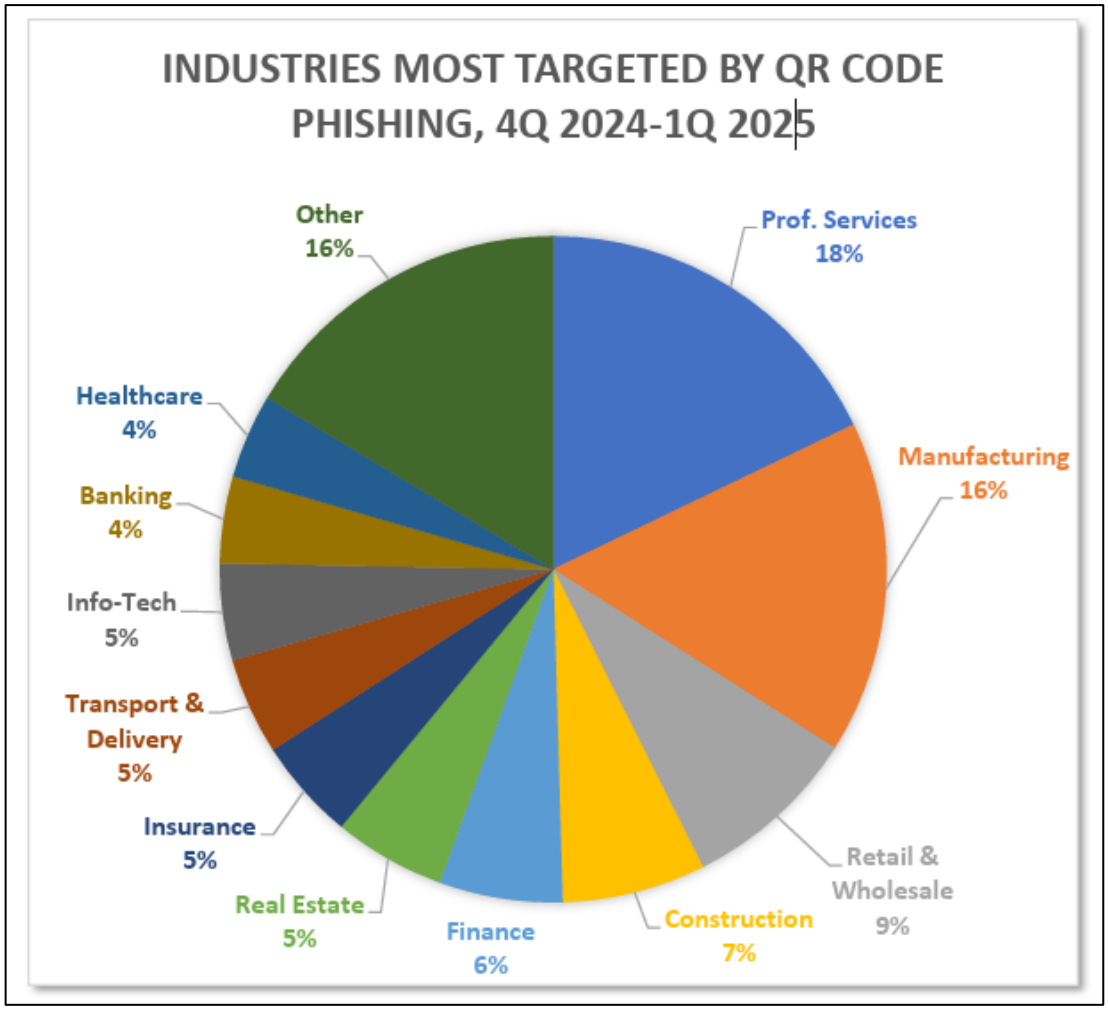
– QRCC[.]IO був найпопулярнішим генератором QR-кодів, за допомогою якого було створено 189 011 унікальних шкідливих QR-кодів. Хоча цей сервіс пропонує легальні бізнес-операції, зловмисники постійно використовували його інфраструктуру для фішингових та шахрайських кампаній.

– QRCO[.]DE використовувався для генерації 177 909 шкідливих QR-кодів. Цей провайдер використовувався для здійснення різних видів атак, включаючи розповсюдження шкідливого програмного забезпечення та фішинг-кампанії.

– ME-QR[.]COM використовувався для генерації 148 004 шкідливих QR-кодів, проте позиціонує себе як сервіс, що дбає про безпеку. Динамічна функція QR-кодів та можливості відстеження цього провайдера є привабливими для злочинців.

На рисунку 1.3 представлено графік галузей, які найбільше піддаються

фішингу з використанням QR-кодів з 4 кварталу 2024 по 1 кварталу 2025 років.

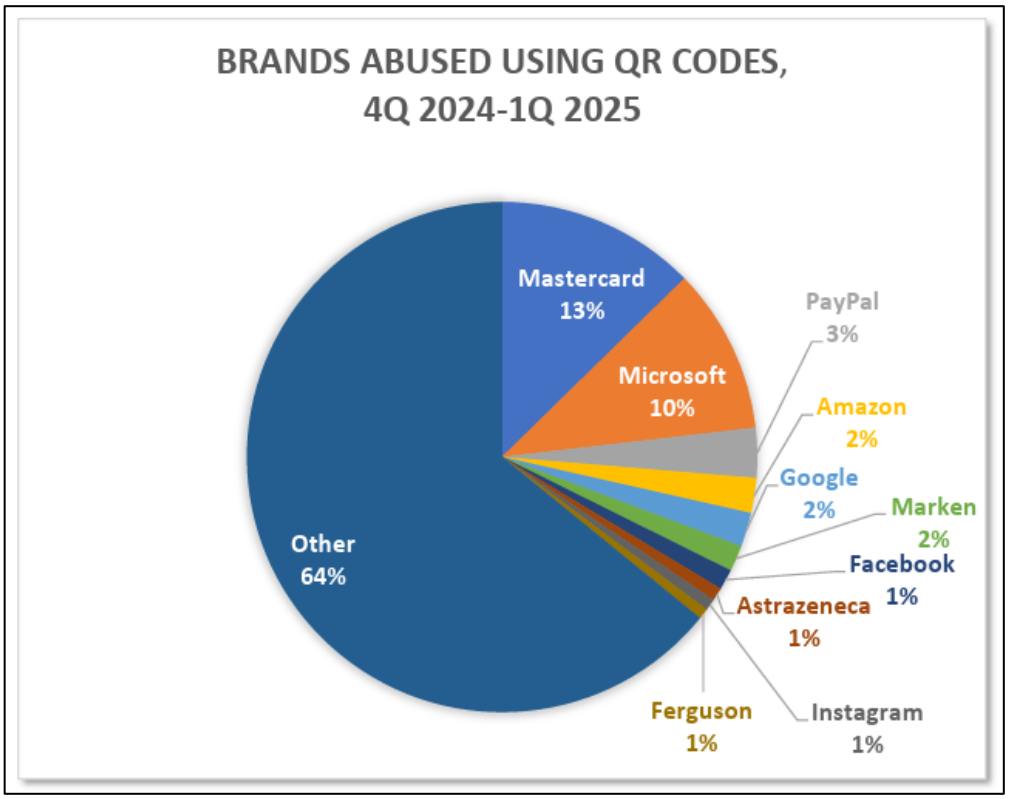


Риснуок 1.3 – Галузі фішингу за допомогою QR-кодів

Найчастіше атакам піддавалася компанія Mastercard – 14 233 QR-коди, а Microsoft – 11 796. Було атаковано велику кількість інших брендів, але з використанням значно меншої кількості QR-кодів, що свідчить про те, що злочинці здійснювали цілеспрямовані атаки, а не випадкові. Атаки на Mastercard підкреслюють бажання зловмисників отримати дані для входу, які вони можуть перетворити на готівку, наприклад, купуючи фізичні товари за допомогою викрадених даних кредитних карток. Платформи обробки платежів зазнавали постійних атак – PayPal було атаковано за допомогою 3546 QR-кодів, а банківські установи зазнали значного обсягу атак. 7 Mimecast також виявив складні регіональні моделі націлювання. Споживачі в Азіатсько-

Тихоокеанському регіоні, як правило, отримували електронні листи, націлені на банки. Споживачі в Європі зазнавали більше атак на платіжні системи та державні служби, а одержувачі електронної пошти в Північній Америці – більше атак на корпоративні хмарні служби та фінансові платформи.

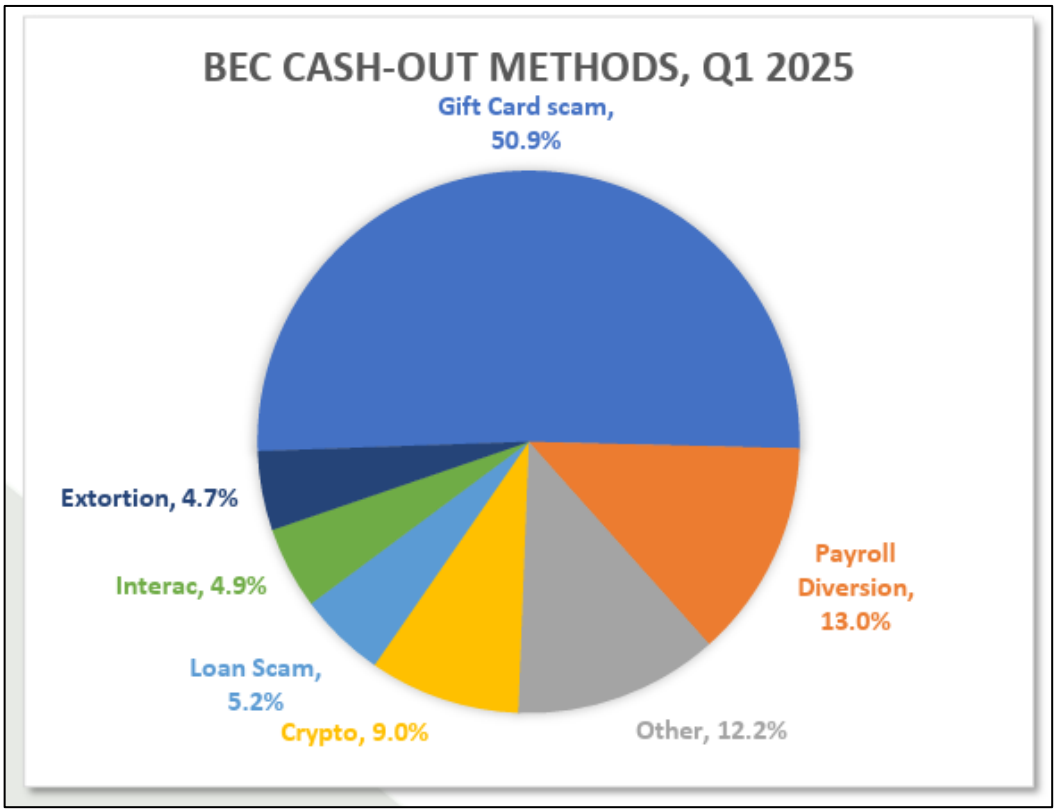
На рисунку 1.4 представлено графік розподілення брендів, які піддавались шахрайству за допомогою QR-кодів.



Риснуок 1.4 – Бренди фішингу за допомогою QR-кодів

Щодо Компрометації корпоративної електронної пошти BEC, за даними Fortra, середня сума, яку запитували в BEC-атаках з вимогою банківського переказу, у Q1 2025 року становила $42 236. При цьому загальна кількість атак зросла на 33%. Найпопулярнішим методом виведення коштів було вимагання подарункових карток 51% атак. 72% BEC-атак було здійснено з використанням безкоштовних вебпоштових доменів, причому Gmail був найпопулярнішим провайдером серед шахраїв 73.5% випадків. Серед реєстраторів доменів, що використовуються для BEC-шахрайства, Cloudflare став найпопулярнішим

28.6%, що свідчить про те, що шахраї відчувають себе захищеними, ховаючись за його сервісами. На рисунку 1.5 продемонстровано графік розподілення суми в банківських переказів у першому кварталі 2025 року.



Риснуок 1.5 – Розподілення суми BEC-атаках з вимогою банківського переказу

1.2 Види та підходи до виявлення фішингових сайтів

Фішинг становить серйозну загрозу кібербезпеці, оскільки зловмисники використовують різноманітні методи соціальної інженерії та технічних маніпуляцій для викрадення конфіденційних даних, таких як паролі, банківські реквізити чи особиста інформація. Ефективне розпізнавання фішингових сайтів вимагає застосування комбінації традиційних і сучасних методів, які враховують як технічні характеристики ресурсу, так і поведінку користувачів. На рисунку 1.6 показано класифікацію фішингових повідомлень.

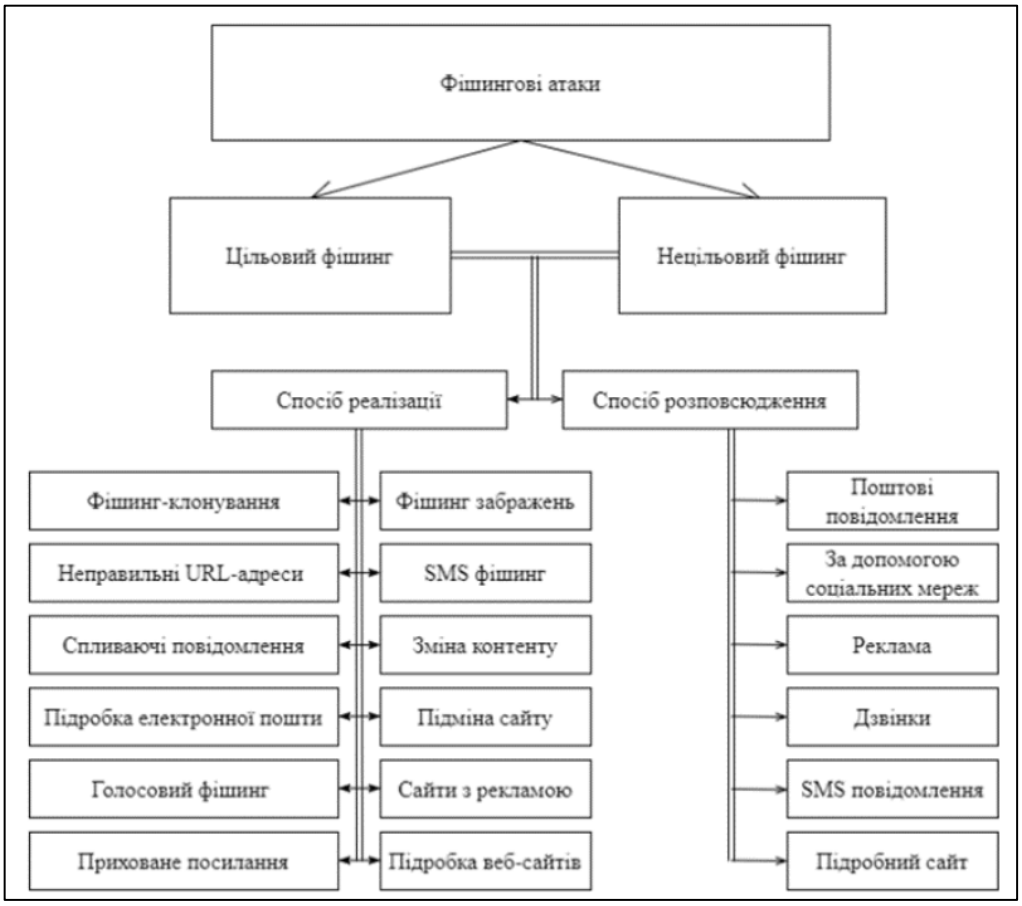


Рисунок 1.6 – Класифікація фішингових атак

Фішингові атаки поділяються за своєю спрямованістю на цільові та нецільовий фішинг [8].

Нецільовий фішинг спрямований на широку аудиторію і часто реалізується через масові розсилки електронних листів, СМС чи повідомлень у соціальних мережах. Він є менш персоналізованим, але охоплює велику кількість потенційних жертв.

Цільовий фішинг є більш складним і небезпечним, оскільки зловмисники попередньо вивчають інформацію про конкретну жертву, наприклад, профілі в соціальних мережах, звички чи контакти. На основі зібраних даних створюються переконливі та індивідуалізовані повідомлення, що значно підвищує ймовірність успіху атаки.

За способом реалізації фішинг класифікується наступним чином:

– Електронний фішинг (Email Phishing) використовує підроблені

електронні листи, що імітують офіційні повідомлення від банків, сервісних центрів чи інтернет-провайдерів. Наприклад, шахраї можуть просити підтвердити дані через фальшивий сайт, посилаючись на "технічний збій";

– Фішинг-клонування (Clone Phishing) передбачає створення копій легітимних, раніше надісланих повідомлень, але із заміною оригінальних посилань чи вкладень на шкідливі;

– Підробка електронної пошти (Business Email Compromise, BEC). Шахраї видають себе за керівників, довірених осіб або постачальників послуг, запитуючи конфіденційні дані або вимагаючи термінового фінансового переказу;

– Використання неправильних URL (Typosquatting/Homograph Attacks). Створення фальшивих сайтів із майже ідентичними адресами, де використовується заміна символів;

– Фішинг через рекламу (Search Engine Phishing). Розміщення привабливих пропозицій або реклами в пошукових системах чи на сайтах, які ведуть на шкідливі ресурси;

– Голосовий фішинг (Vishing) та СМС-фішинг (Smishing). Використання телефонних дзвінків чи текстових повідомлень для обману, де жертву обманом змушують надати інформацію або виконати шкідливі команди;

– Pharming (Фармінг). Це перенаправлення користувачів на фальшиві сайти через маніпуляції з DNS-серверами або локальними файлами hosts на комп'ютері жертви, що є більш технічно складним методом, ніж прямий фішинг;

– Фішинг-зображень та спливаючі повідомлення. Використання мультимедійного контенту чи спливаючих вікон для маскування шкідливого програмного забезпечення або заманювання до введення даних.

Підходи та алгоритми розпізнавання фішингових сайтів. Для виявлення фішингових атак можна виділити два основні підходи: навчання користувачів та використання програмних засобів [9].

Навчання користувачів. Цей підхід спрямований на підвищення обізнаності користувачів щодо природи фішингових атак, що допомагає їм розрізняти шахрайські та легітимні повідомлення. Навчання розглядається як превентивний захід, що сприяє розпізнаванню фішингу, наприклад, через звернення уваги на підозрілі URL чи заклики до термінових дій. Дослідження Університету Карнегі Меллон показало, що багато користувачів оцінюють сайти лише за зовнішнім виглядом, ігноруючи показники безпеки, що підкреслює важливість, але недостатність цього заходу.

Програмні засоби. Програмні рішення покликані усунути прогалини, спричинені помилками чи недостатньою обізнаністю користувачів, шляхом автоматичної класифікації повідомлень та сайтів на фішингові та легітимні. Цей підхід є більш економічним та масштабованим порівняно з навчанням, особливо для великих онлайн-платформ. Ефективність розпізнавання можна підвищити шляхом удосконалення алгоритмів машинного навчання або правил виявлення.

Основні програмні методи розпізнавання:

1. Чорні та білі списки:

– Чорні списки містять відомі фішингові URL-адреси. Приклад – Google Safe Browsing, який захищає понад 5 мільярдів пристроїв, попереджаючи про небезпечні сайти.

– Білі списки включають перелік легітимних адрес.

– Чорні списки мають затримку в оновленні, що дозволяє зловмисникам діяти до внесення сайту в базу. Білі списки також потребують часу для додавання нових легітимних адрес, що знижує їх ефективність проти нових атак.

2. Евристичні методи:

– Ці методи аналізують характеристики веб-сторінки, такі як структура URL, HTML-код, об’єктна модель документа DOM, на наявність підозрілих елементів такі як заплутані URL чи нестандартні скрипти.

– Перевагою є незалежність від попередньо сформованих списків,

що дозволяє виявляти нові атаки, але потребує складних алгоритмів для точного аналізу.

3. Методи візуальної схожості:

– Фішингові сайти часто копіюють дизайн легітимних ресурсів, використовуючи зображення чи Flash. Методи аналізують DOM, CSS чи піксельні характеристики, створюючи "підписи" сайтів для порівняння.

– Дозволяє швидко виявляти точні копії.

– Вимагає значних обчислювальних ресурсів.

4. Методи машинного навчання ML:

– Алгоритми ML є потужним інструментом для класифікації фішингових сайтів завдяки їхній здатності адаптуватися до нових шаблонів атак.

– Класифікатори. Логістична регресія для оцінки ймовірності на основі ознак, таких як довжина URL чи наявність HTTPS, Дерева рішень і Випадковий ліс для аналізу складних комбінацій ознак.

– Ансамблеві методи. Градієнтний бустинг, наприклад, XGBoost, LightGBM для обробки великих наборів даних та досягнення високої точності.

– Нейронні мережі. Згорткові мережі CNN можуть аналізувати візуальні елементи, а рекурентні мережі RNN – послідовності даних, таких як мережеві запити.

– Висока точність і гнучкість.

– Потребують якісних та актуальних наборів даних для навчання.

5. Програмні рішення та розширення браузерів:

– Антивірусне ПЗ наприклад, GoldFish із байєсівською фільтрацією і розширення браузерів SpoofGuard, PwdHash блокують фішингові сайти або попереджають про підозрілі дії.

– Google Chrome із функцією Enhanced Safe Browsing пропонує перевірку в реальному часі та захист від невідомих атак. Однак, такі рішення можуть бути вразливими до нових технік, таких як "байєсівське отруєння", і потребують постійного оновлення та удосконалення [10].

1.3 Вибір мови програмування та середовища розробки

Вибір мови програмування, середовища розробки та алгоритмів машинного навчання є ключовим етапом створення програмного забезпечення, оскільки від цього залежить ефективність, зручність і масштабованість проекту. Python вирізняється своєю універсальністю та широкими можливостями, що робить її оптимальним вибором для багатьох задач, зокрема для розробки моделей машинного навчання. Платформа Kaggle, своєю чергою, забезпечує доступ до якісних даних і інструментів, що ідеально підходить для реалізації таких проектів.

Python - це високорівнева інтерпретована мова програмування з динамічною строгою типізацією та простим синтаксисом, що полегшує її вивчення та використання. Завдяки великій стандартній бібліотеці та активній спільноті, яка розробляє численні зовнішні бібліотеки (наприклад, NumPy, Pandas, Scikit-learn), Python підходить для аналізу даних, веб-розробки та створення систем машинного навчання [11].

Особливості Python:

– Простота синтаксису: Чіткий і зрозумілий код, схожий на природну мову, зменшує кількість помилок і прискорює розробку.

– Ефективні структури даних: Вбудовані списки, словники, множини спрощують обробку даних.

– Об’єктно-орієнтоване програмування: Забезпечує модульність і повторне використання коду.

– Інтерпретація: Відсутність компіляції прискорює тестування та розробку.

– Кросплатформенність: Python працює на всіх популярних платформах, включаючи Windows, Linux, macOS.

Переваги Python:

– Легкість у вивченні завдяки англійському синтаксису та меншій кількості коду порівняно з C++ чи Java.

– Висока продуктивність розробки: Простота дозволяє зосередитися на логіці задачі.

– Інтерпретована природа: Построкове виконання спрощує виявлення помилок.

– Динамічна типізація: Автоматичне визначення типів даних зменшує необхідність їх явного оголошення.

– Безкоштовність і відкритий код: Ліцензія OSI дозволяє вільно модифікувати та поширювати Python.

– Велика бібліотека: Стандартна та зовнішні бібліотеки покривають більшість потреб розробників.

– Портативність: Код легко переноситься між платформами без значних змін.

Недоліки Python:

– Низька швидкість виконання через інтерпретацію та динамічну типізацію.

– Високе споживання пам’яті, що ускладнює оптимізацію ресурсів.

– Обмежене застосування в мобільних розробках через низьку обчислювальну потужність.

– Слабка підтримка баз даних порівняно з іншими технологіями.

– Помилки виконання через динамічну типізацію, яка може змінити тип змінної.

Для реалізації проекту обрано Kaggle - онлайн-платформу для аналізу даних, машинного навчання та змагань у сфері data science. Kaggle забезпечує доступ до інструментів, наборів даних і спільноти, що робить її ефективним середовищем для розробки моделей розпізнавання фішингових сайтів [12].

Переваги Kaggle:

– Доступ до даних: Велика кількість структурованих наборів даних спрощує створення моделей.

– Конкурентне середовище: Змагання дозволяють порівнювати

рішення та вдосконалювати навички.

– Навчальні можливості: Курси, приклади коду та рішення інших учасників сприяють навчанню.

– Спільна робота: Підтримка колаборації для обміну кодом і відгуками.

– Репутація та кар’єра: Успіхи на Kaggle підвищують авторитет у спільноті data science.

– Недоліки Kaggle:

– Висока конкуренція, що може бути викликом для новачків.

– Обмеження обчислювальних ресурсів, які уповільнюють обробку великих моделей.

– Значні часові витрати на досягнення високих результатів.

1.4 Опис моделей, які будуть використовуватись для передбачення даних

Машинне навчання застосовує різноманітні математичні алгоритми для класифікації об'єктів і прийняття рішень. В залежності від конкретної задачі та вхідних даних, різні алгоритми можуть демонструвати різну точність рішень і вимагати різної кількості обчислювальних ресурсів для виконання. Нижче наведено опис одного з найбільш фундаментальних і поширених алгоритмів для бінарної класифікації.

Логістична регресія – це потужний статистичний метод, який використовується для моделювання ймовірності настання події та застосовується для бінарної класифікації [13]. У контексті прогнозування фішингу, вона оцінює ймовірність того, що сайт належить до класу "Фішинг" проти класу "Легітимний", використовуючи незалежні змінні. На відміну від звичайної лінійної регресії, логістична модель не передбачає, що залежна змінна є лінійною функцією незалежних змінних. Натомість, вона використовує лінійну комбінацію ознак, яка стискає результат у діапазон

ймовірностей. Лінійна модель передбачає:

$$y = b_1x_1 + b_2x_2 + \ldots + b_nx_n + uy = b_1x_1 + b_2x_2 + \ldots + b_nx_n + u,$$

де y – залежна змінна,

xi – пояснююча змінна,

bi – коефіцієнт змінної,

u – випадкова похибка.

Умовна ймовірність події розраховується за формулою:

$$P(y \mid x) = b'xP(y \mid x) = b'x.$$

Переваги та Застосування. Однією з головних переваг логістичної регресії є її висока інтерпретованість: коефіцієнти b_i дозволяють зрозуміти вплив кожної ознаки на фінальну ймовірність, що є критично важливим для пояснення рішень у системах безпеки. Також вона є ефективною з точки зору обчислювальних ресурсів для базових моделей, особливо коли кількість вхідних факторів обмежена. У сфері прогнозування фішингу цей метод дозволяє оцінити ступінь ризику, що суттєво підвищує функціональність системи.

Недоліки та Обмеження. Логістична регресія має недоліки, такі як чутливість до аномальних даних (викидів) і мультиколінеарності між незалежними змінними. Це вимагає ретельної попередньої підготовки даних та відбору ознак. Порівняно з більш складними моделями, вона може мати обмежену здатність виявляти складні нелінійні залежності у даних. Крім того, побудова моделі вимагає оптимізації через метод максимальної правдоподібності, який є більш ресурсомістким, ніж, наприклад, метод найменших квадратів. Незважаючи на ці обмеження, логістична регресія залишається наріжним каменем для моделювання ймовірностей і класифікації, і при правильній регуляризації та попередній обробці даних вона забезпечує високу якість прогнозування фішингових загроз.

Метод дерев рішень є одним із найбільш поширених способів вирішення задач класифікації та прогнозування у сфері машинного навчання. Цей метод застосовується для вирішення проблем, де цільова змінна має дискретні або

безперервні значення. Дерево рішень являє собою ієрархічну послідовну структуру, де рішення приймаються на основі відповідей "так" або "ні" на низку послідовних питань про вхідні ознаки. У процесі побудови моделі алгоритм ітеративно визначає ступінь впливу кожного вхідного атрибуту на значення вихідного, шукаючи ознаку, розбиття за якою дасть максимальне зниження невизначеності. Модель росте доти, доки розбиття вузла покращує якість прогнозу, завершуючись у листових вузлах [14].

Метод дерев рішень має низку суттєвих переваг: простота інтерпретації та візуалізації, універсальність, обробка різноманітних типів даних та мінімальні вимоги до попередньої обробки даних. Крім того, дерева рішень відносно стійкі до шуму та викидів у даних, ефективно виявляють важливі змінні та швидко навчаються. Ключовим недоліком є схильність до перенавчання, особливо при побудові глибоких структур. Для боротьби з цим застосовують прунінг або встановлюють обмеження на глибину дерева. Однак, найбільша сила дерев рішень полягає у можливості інтеграції з ансамблевими методами, такими як Випадковий ліс і Градиентний бустинг, що дозволяє значно підвищити точність і надійність моделей, вирішуючи проблему перенавчання. На рисунку 1.7 представлено приклад блок-схеми методу дерев рішень.

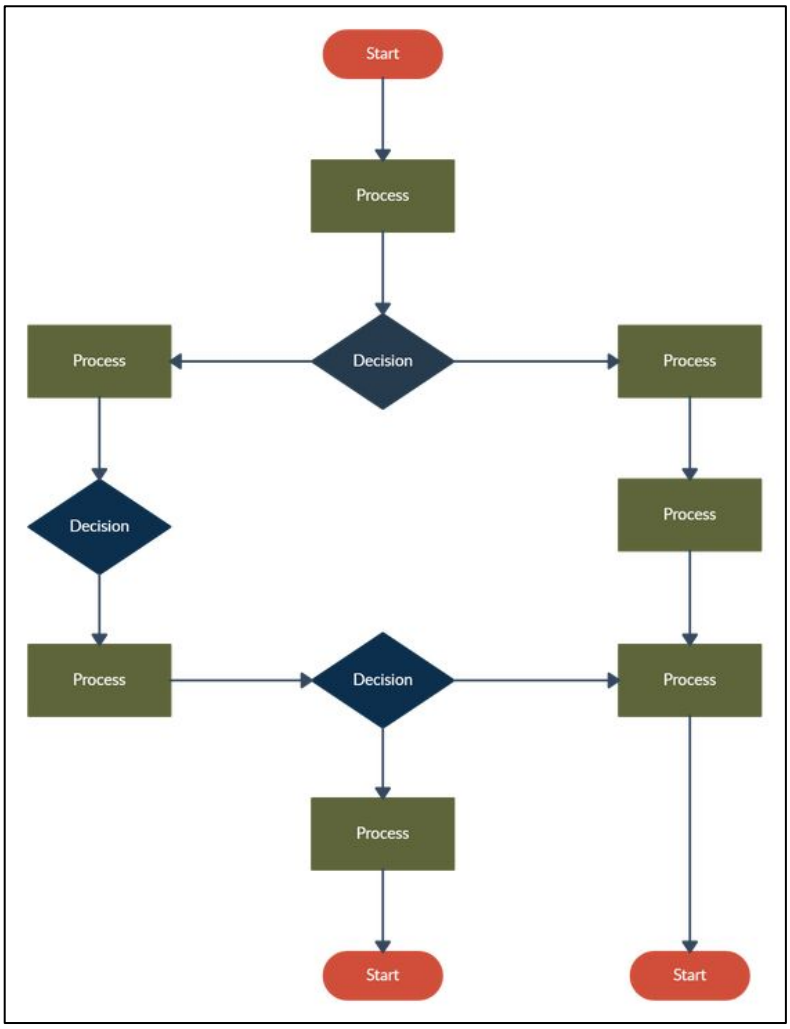


Рисунок 1.7 – Блок-схема методу дерев рішень

Випадковий ліс є потужним ансамблевим методом машинного навчання, який використовується для класифікації та регресії. Його ефективність полягає в тому, що він працює, будуючи велику кількість незалежних дерев рішень під час навчання моделі, а фінальний прогноз формується шляхом голосування або усереднення прогнозів усіх побудованих дерев. Для забезпечення різноманітності кожного дерева і, як наслідок, зниження дисперсії моделі, алгоритм застосовує два рівні випадковості. По-перше, кожне дерево навчається на своїй унікальній випадковій підвибірці даних з вихідного навчального набору. По-друге, на кожному кроці розгалуження дерева випадковим чином обирається лише невелика підмножина ознак, і ознака для оптимального розбиття вибирається тільки з цієї підмножини. Така подвійна випадковість гарантує, що жодне дерево не побачить увесь навчальний набір

та всі ознаки одночасно. Це робить модель високоточною і стійкою до перенавчання, що критично важливо для узагальнення на нових даних, як-от у прогнозуванні фішингу. Метод ефективно працює з великими обсягами даних, потребує мінімального налаштування гіперпараметрів і має природну здатність визначати важливість змінних. Класифікація об'єктів проводиться шляхом голосування: кожне дерево комітету незалежно класифікує об'єкт, і перемагає той клас, за який проголосувало найбільше число дерев. На рисунку 1.8 Представлено приклад блок-схеми методу випадковий ліс.

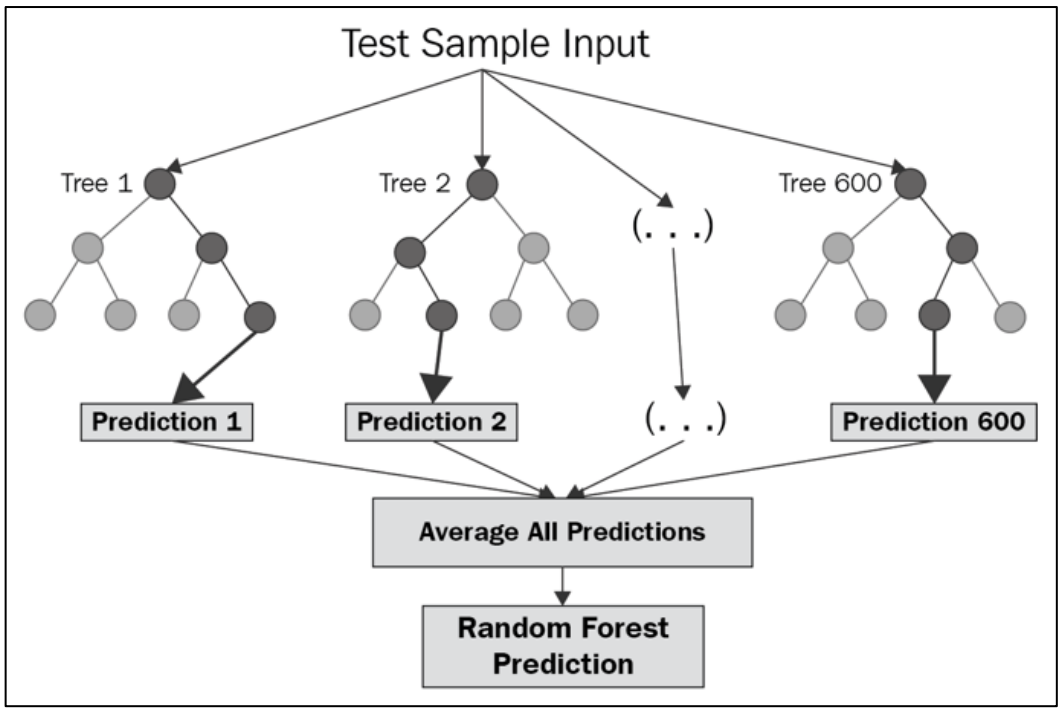


Рисунок 1.8 – Блок-схема методу випадковий ліс

Ансамблеве навчання - це потужний підхід у машинному навчанні, який об'єднує прогнози декількох окремих моделей, відомих як слабкі учні, для досягнення вищої точності та надійності порівняно з будь-якою окремою моделлю. Завдяки поєднанню, цей підхід стає більш гнучким і менш чутливим до випадкових коливань у даних. Найпоширеніші та найефективніші методи ансамблевого навчання - це багінг та бустинг [15].

Багінг - це метод паралельного ансамблювання. Він передбачає незалежне навчання багатьох моделей на різних підмножинах даних, обраних

випадковим чином з повторенням з вихідної навчальної вибірки. Ключова ідея полягає у зменшенні дисперсії моделі та запобіганні перенавчанню. Одним із класичних прикладів застосування багінгу є Випадковий ліс, де створюється серія дерев рішень, кожне з яких навчається на своїй підвибірці, а остаточний прогноз визначається шляхом усереднення результатів усіх дерев або голосуванням.

Бустинг - це метод послідовного ансамблювання. На відміну від багінгу, бустинг навчає моделі послідовно, де кожна наступна модель намагається виправити помилки попередньої. В основі бустингу лежить ідея поступового накопичення слабких моделей, кожна з яких постійно вдосконалюється на основі залишкових помилок попереднього етапу. Це призводить до значного зменшення зміщення моделі.

Градієнтний бустинг є яскравим прикладом цього підходу. Це потужний алгоритм, який об'єднує кілька слабких моделей для створення однієї сильної. У цьому підході кожне нове дерево рішень намагається мінімізувати функцію втрат моделі, рухаючись у напрямку найбільшого градієнта залишкової помилки. Слабкі моделі інтегруються таким чином, що кожна наступна модель враховує залишкові помилки, а фінальна модель комбінує результати всіх етапів, що забезпечує надзвичайно високу точність.

Метод градієнтного бустингу є одним із найбільш продуктивних у машинному навчанні, маючи такі ключові переваги:

– Висока точність. Здатність досягати високої точності прогнозування завдяки послідовному зменшенню помилок та виявленню складних патернів у даних;

– Гнучкість. Дозволяє налаштовувати безліч параметрів та використовувати різні функції втрат, оптимізуючи модель під конкретну задачу;

– Обробка різноманітних типів даних та пропущених значень;

– Висока масштабованість і можливість роботи з великими наборами даних;

– Інтерпретованість. Хоча він складніший за окреме дерево, метод дозволяє оцінювати важливість ознак та розуміти вплив окремих змінних на результати моделі.

Ці переваги роблять градієнтний бустинг потужним і ефективним методом для широкого спектру задач, включаючи складну бінарну класифікацію, як-от прогнозування фішингових сайтів. На рисунку 1.9 представлено приклад блок-схеми методу градієнтного бустингу.

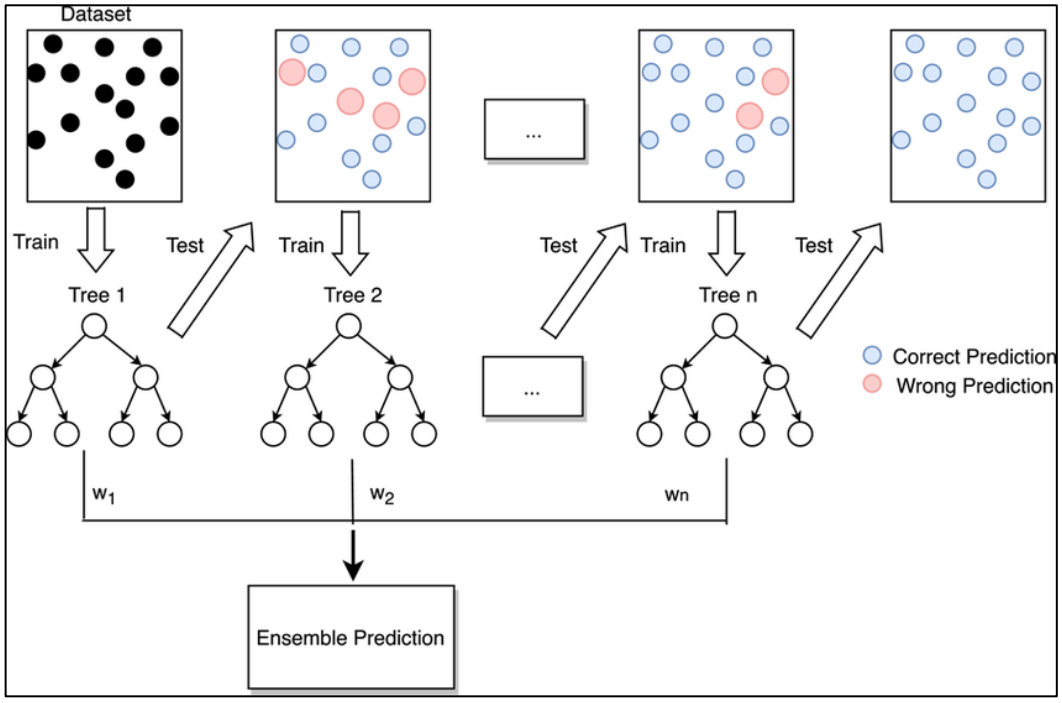


Рисунок 1.9 – Блок-схема методу градієнтний бустинг

XGBoost є масштабованим інструментарієм для навчання на основі узагальненого дерева рішень з градієнтним прискоренням (GBDT). Він є гнучкою і надзвичайно точною версією градієнтного бустингу, розробленою для підвищення продуктивності та швидкості обчислень. На відміну від традиційного GBDT, XGBoost будує дерева паралельно, використовуючи рівневу техніку, та оцінює якість розбиття за допомогою часткових сум [16].

Ключові архітектурні вдосконалення XGBoost включають:

– Регуляризація. Для мінімізації надмірного перенавчання модель штрафує складніші дерева, використовуючи як LASSO (L1), так і Ridge (L2)

регуляризацію;

– Оптимізація апаратного забезпечення. Метод був створений для максимального ефективного використання апаратних ресурсів, включаючи використання кешу для зберігання статистики градієнта та позаядерні обчислення для управління великими наборами даних, які не вміщуються в пам'яті;

– Обрізання. Замість жадібного критерію зупинки розбиття, XGBoost починає обрізання дерев у зворотному напрямку, використовуючи аргумент максимальної глибини, що значно підвищує продуктивність обчислень;

– Зважений квантильний ескіз. Використовується для успішного знаходження найкращих точок розбиття серед зважених наборів даних.

XGBoost набув великої популярності завдяки своїй здатності демонструвати найвищу точність у змаганнях зі структурованих даних, ставши де-факто стандартом для задач регресії, класифікації та ранжування.

LightGBM - це інноваційний фреймворк, заснований на техніці дерев рішень, який є більш швидким та ефективним порівняно з XGBoost, особливо при роботі з великими наборами даних.

Ключові переваги та механізми LightGBM:

– Стратегія розбиття за листками. На відміну від інших методів бустингу, які розбивають дерево за рівнем, LightGBM розбиває дерево на частини на основі найкращого збігу, тобто розширює той листовий вузол, який забезпечує максимальне зниження втрат. Це призводить до побудови значно складніших, асиметричних дерев та вищої точності. Хоча це може збільшити ризик перенавчання, це контролюється параметром максимальної глибини дерева;

– Гістограмний підхід. Для підвищення швидкості та зменшення використання пам'яті LightGBM використовує підхід на основі гістограми, який розбиває неперервні значення ознак на дискретні біни. Це прискорює процес навчання та призводить до нижчого використання пам'яті;

– Сумісність з великими даними. LightGBM здатен працювати з великими наборами даних, вимагаючи при цьому значно менше часу на навчання;

– Налаштування. Висока точність досягається через використання меншої швидкості навчання з великою кількістю ітерацій.

ExtraTrees є варіантом Випадкового Лісу, заснованим на ансамблевому підході багінгу, який використовує додатковий елемент випадковості для подальшого зниження дисперсії та прискорення навчання. На відміну від стандартного Random Forest, який шукає оптимальний поріг розбиття надзвичайна рандомізація ExtraTrees обирає як ознаку, так і поріг розбиття для кожного вузла випадковим чином. Ця додаткова рандомізація робить модель менш схильною до перенавчання та значно прискорює процес навчання, оскільки пропускається трудомісткий етап пошуку оптимального порогу.

Алгоритм k-найближчих сусідів (KNN) - це простий, але потужний метод керованого машинного навчання, який використовується для вирішення задач класифікації та регресії. Його головна функція - оцінити ймовірність того, що нова точка даних належить до тієї чи іншої групи, базуючись на її найближчих сусідах. KNN є непараметричним алгоритмом, оскільки він не робить жодних припущень щодо базового розподілу даних, а також лінивим алгоритмом навчання. Це означає, що ніяке навчання не відбувається під час етапу тренування: модель просто зберігає весь навчальний набір даних. Обчислення та побудова моделі відбуваються лише у момент отримання запиту на прогнозування. Процес прогнозування для нового цільового прикладу складається з наступних етапів: спочатку користувач визначає значення K - бажаної кількості сусідів. Далі обчислюється відстань між цільовим прикладом і кожним іншим прикладом у навчальному наборі даних. Після цього відстані сортуються, і обираються перші K найближчих сусідів. Прогноз щодо цільової точки даних формується на основі міток цих K найближчих сусідів за механізмом голосування. У разі класифікації використовується

режим majority vote - клас, який набрав найбільшу кількість голосів, визначає категорію цільової точки. Якщо ж виконується регресія, береться середнє значення міток K найближчих сусідів. Вибір значення K є критичним: мале K робить модель чутливою до шуму та викидів, а велике K може призвести до недостатнього навчання. Точність класифікації KNN перевіряється за допомогою матриці помилок. На відміну від складних моделей, метод KNN є простим для розуміння та реалізації. Його непараметрична природа дозволяє йому ефективно працювати з даними, які мають нелінійні або складні границі рішень, де точки даних чітко визначені. Реалізація K-Nearest Neighbor має широкий спектр застосувань, включаючи виявлення закономірностей, прогнозування вартості акцій та класифікацію зображень, оскільки він здатний групувати схожі фрагменти даних [17].

Для розпізнавання фішингових сайтів у рамках машинного навчання використовуються нейронні мережі, зокрема багатошаровий перцептрон (Multilayer Perceptron, MLP), який є моделлю з кількома шарами нейронів, що обробляє вхідні дані через приховані шари для виявлення складних нелінійних залежностей. MLP ефективно працює з великими наборами даних, адаптується до різних типів ознак і забезпечує високу точність за умови належного налаштування та достатніх обчислювальних ресурсів [18].

Використано бібліотеки TensorFlow і Scikit-learn для створення та оцінки нейронних мереж. Модель на основі TensorFlow включає три варіанти архітектур:

– Нейронна мережа 1. Плитка архітектура з активацією ReLU та високим рівнем відсіву (Dropout 0.5) для зменшення перенавчання.

– Нейронна мережа 2. Глибша структура з активацією tanh і помірним відсівом (Dropout 0.3) для кращого поширення градієнтів.

– Нейронна мережа 3: Середня за глибиною модель із комбінацією активацій ReLU та ELU з низьким відсівом (Dropout 0.2) для балансу між складністю та узагальненням.

Альтернативно, модель MLP із бібліотеки Scikit-learn налаштована з трьома прихованими шарами (64, 32, 16 нейронів), активацією ReLU, оптимізатором Adam і адаптивною швидкістю навчання. Вона демонструє порівнянну продуктивність завдяки збалансованій архітектурі та автоматичному налаштуванню параметрів.

Ефективність моделей оцінюється через метрики точності, матриці помилок та ROC-криві. Нейронні мережі, зокрема MLP, є потужним інструментом для класифікації фішингових сайтів, оскільки дозволяють обробляти складні патерни в даних, адаптуючись до динамічних змін у фішингових атаках.

1.5 Висновки

У даному розділі здійснено аналіз предметної області та встановлено, що фішинг є домінуючою та найбільш адаптивною кіберзагрозою, що постійно зростає, про що свідчать статистичні дані APWG. Для ефективної протидії фішингу необхідне поєднання навчання користувачів та використання програмних засобів. Найбільш перспективним підходом для точного передбачення фішингових сайтів є методи машинного навчання. В рамках дослідження проаналізовано та застосовано широкий спектр класифікаційних алгоритмів, включаючи моделі такі як: логістична регресія, дерева рішень, випадковий ліс, градієнтний бустинг, ExtraTrees, XGBoost Model, LightGBM Model та K-Nearest Neighbors Model (KNN). Крім того, для виявлення складних нелінійних залежностей досліджено архітектури Нейронних мереж, зокрема Багатошарового перцептрона.

2 ОСНОВНІ ЕТАПИ ВИКОНАННЯ РОБОТИ ТА ОГЛЯД НАБОРУ ВХІДНИХ ДАНИХ

2.1 Характеристика датасету для аналізу фішингових сайтів

У даній роботі використано датасет, що містить дані про фішингові вебсторінки, створений шляхом об’єднання двох наборів даних із однаковими характеристиками. Для уникнення надмірності та зосередження на ключових ознаках було відібрано лише найбільш релевантні характеристики. Об’єднаний набір даних забезпечує комплексне представлення спільних рис вихідних наборів, зберігаючи структурований і цілеспрямований набір ознак [19].

Датасет включає наступні ознаки:

– url_length: загальна кількість символів в URL-адресі, включаючи літери, цифри та спеціальні символи;

– n_dots: кількість крапок (.) в URL;

– n_hyphens: кількість дефісів (-) в URL;

– n_underline: кількість символів підкреслення (_) в URL;

– n_slash: кількість косих рисок (/) в URL;

– n_questionmark: кількість знаків питання (?) в URL;

– n_equal: кількість знаків рівності (=) в URL;

– n_at: кількість символів собачки (@) в URL;

– n_and: кількість амперсандів (&) в URL;

– n_exclamation: кількість знаків оклику (!) в URL;

– n_space: кількість пробілів у URL, які можуть бути закодовані як %20;

– n_tilde: кількість тильд (~) в URL;

– n_comma: кількість ком (,) в URL;

– n_plus: кількість знаків плюс (+) в URL;

– n_asterisk: кількість зірочок (*) в URL;

– n_hashtag: кількість решіток (#) в URL;

– n_dollar: кількість знаків долара ($) в URL;

– n_percent: кількість знаків відсотка (%) в URL;

– n_redirection: кількість HTTP-переспрямувань до кінцевого пункту призначення URL;

– phishing: мітка URL, де 1 позначає фішинговий сайт, а 0 – легітимний, визначена за допомогою алгоритму або моделі для класифікації.

На рисунках 2.1 та 2.2 відображено перші 10 рядків датасету. Це дозволяє отримати початкове уявлення про структуру даних, включаючи значення ознак, таких як довжина URL, кількість крапок, дефісів, інших спеціальних символів, а також мітку phishing. Такий попередній перегляд допомагає оцінити розподіл даних і виявити потенційні особливості або аномалії.

	url_length	n_dots	n_hypens	n_underline	n_slash	n_questionmark	n_equal	n_at	n_and	n_exclamation
0	37	3	0	0	0	0	0	0	0	0
1	77	1	0	0	0	0	0	0	0	0
2	126	4	1	2	0	1	3	0	2	0
3	18	2	0	0	0	0	0	0	0	0
4	55	2	2	0	0	0	0	0	0	0
5	32	3	1	0	0	0	0	0	0	0
6	19	2	0	0	0	0	0	0	0	0
7	81	2	0	0	0	0	0	0	0	0
8	42	2	0	0	0	0	0	0	0	0
9	104	1	10	0	0	0	0	0	0	0

Рисунок 2.1 – Приклад набору даних

n_space	n_tilde	n_comma	n_plus	n_asterisk	n_hastag	n_dollar	n_percent	n_redirection	phishing
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Рисунок 2.2 – Приклад набору даних

На рисунку 2.3 показано загальна інформація про дані датасету.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100077 entries, 0 to 100076
Data columns (total 20 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   url_length       100077 non-null  int64
 1   n_dots           100077 non-null  int64
 2   n_hypens         100077 non-null  int64
 3   n_underline      100077 non-null  int64
 4   n_slash          100077 non-null  int64
 5   n_questionmark   100077 non-null  int64
 6   n_equal          100077 non-null  int64
 7   n_at             100077 non-null  int64
 8   n_and            100077 non-null  int64
 9   n_exclamation    100077 non-null  int64
 10  n_space          100077 non-null  int64
 11  n_tilde          100077 non-null  int64
 12  n_comma          100077 non-null  int64
 13  n_plus           100077 non-null  int64
 14  n_asterisk       100077 non-null  int64
 15  n_hastag         100077 non-null  int64
 16  n_dollar         100077 non-null  int64
 17  n_percent        100077 non-null  int64
 18  n_redirection    100077 non-null  int64
 19  phishing         100077 non-null  int64
dtypes: int64(20)
memory usage: 15.3 MB
```

Рисунок 2.3 – Огляд даних датасету

Вивід показує, що датасет містить 100077 записів і 20 стовпців, усі з типом даних int64. Кожен стовпець не містить пропущених значень, що свідчить про повноту даних. Стовпці включають ознаки, які були описані попередньо. Цей огляд допомагає зрозуміти структуру даних і підтверджує їх готовність до подальшої обробки без необхідності заповнення пропусків.

На рисунках 2.4 та 2.5 показано описову статистику для числових стовпців

	url_length	n_dots	n_hypens	n_underline	n_slash	n_questionmark	n_equal	n_at	n_and	n_exclamation
count	100077.0000	100077.0000	100077.0000	100077.0000	100077.0000	100077.0000	100077.0000	100077.0000	100077.0000	100077.0000
mean	39.1777	2.2244	0.4052	0.1377	1.1354	0.0244	0.2158	0.0221	0.1433	0.0026
std	47.9718	1.2550	1.2855	0.7240	1.8285	0.1678	0.9598	0.2684	0.9137	0.0822
min	4.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25%	18.0000	2.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
50%	24.0000	2.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
75%	44.0000	2.0000	0.0000	0.0000	2.0000	0.0000	0.0000	0.0000	0.0000	0.0000
max	4165.0000	24.0000	43.0000	21.0000	44.0000	9.0000	23.0000	43.0000	26.0000	10.0000

Рисунок 2.4 – Описова статистика

n_space	n_tilde	n_comma	n_plus	n_asterisk	n_hastag	n_dollar	n_percent	n_redirection	phishing
100077.0000	100077.0000	100077.0000	100077.0000	100077.0000	100077.0000	100077.0000	100077.0000	100077.0000	100077.0000
0.0049	0.0036	0.0024	0.0025	0.0041	0.0004	0.0019	0.1093	0.3615	0.3633
0.1446	0.0785	0.0796	0.1044	0.2840	0.0580	0.0974	1.6953	0.7755	0.4810
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-1.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	1.0000
18.0000	5.0000	11.0000	19.0000	60.0000	13.0000	10.0000	174.0000	17.0000	1.0000

Рисунок 2.5 – Описова статистика

Результати, представлені на рисунках 2.4 та 2.5, містять ключові статистичні показники для кожного числового стовпця:

– Count: кількість ненульових значень (100077 для всіх стовпців);

– Mean: середнє значення, що відображає центральну тенденцію даних;

– Std: стандартне відхилення, яке показує розкид значень відносно середнього;

– Min: мінімальне значення в кожному стовпці, що вказує на нижню межу даних;

– 25% (перший квартиль): значення, нижче якого перебуває 25% даних;

– 50% (медіана): значення, що розділяє дані навпіл;

– 75% (третій квартиль): значення, нижче якого перебуває 75% даних;

– Max: максимальне значення, що вказує на верхню межу даних.

Ці показники дозволяють оцінити розподіл і варіативність даних, виявити можливі викиди або аномалії, а також зрозуміти, які ознаки можуть мати найбільший вплив на класифікацію фішингових сайтів. Наприклад, високі значення url_length або n_redirection можуть вказувати на підозрілі URL, що є важливим для подальшого аналізу та навчання моделей.

2.2 Розвідувальний аналіз даних фішингових сайтів

Інформаційна технологія для передбачення фішингових сайтів, реалізована на платформі Kaggle, забезпечує ефективний аналіз і виявлення потенційно небезпечних веб-ресурсів [20]. Вона базується на детальному дослідженні характеристик фішингових сайтів із використанням методів машинного навчання, зокрема нейронних мереж для прогнозування загроз. Аналізуються різні ключові ознаки сайтів, такі як довжина URL-адреси, кількість крапок, дефісів, косих рисок, знаків питання, знаків рівності та інших параметрів, які допомагають відрізнити фішингові сайти від легітимних.

Для аналізу розподілу ознак використано діаграми розмаху boxplot, які порівнюють розподіл двох параметрів між нормальними та фішинговими сайтами.

На рисунку 2.6 зображено розподіл логарифмічних значень довжини URL для нормальних (світло-зелені коробки) та фішингових (коралові коробки) сайтів.

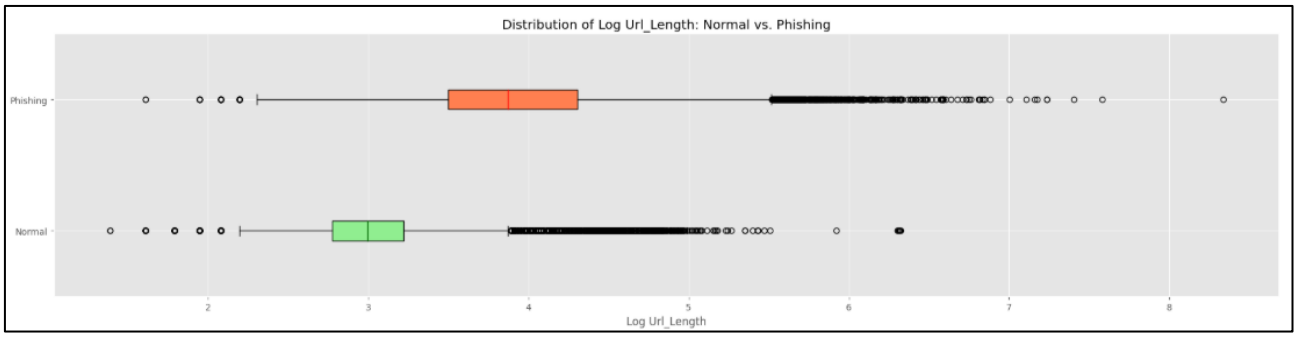


Рисунок 2.6 – Графік розподілу логарифмічних значень довжини URL для нормальних та фішингових сайтів.

Аналогічно, на рисунку 2.7 показано розподіл логарифмічної кількості крапок в URL для обох категорій. Ці графіки дозволяють виявити відмінності в характеристиках, що можуть вказувати на фішингову активність.

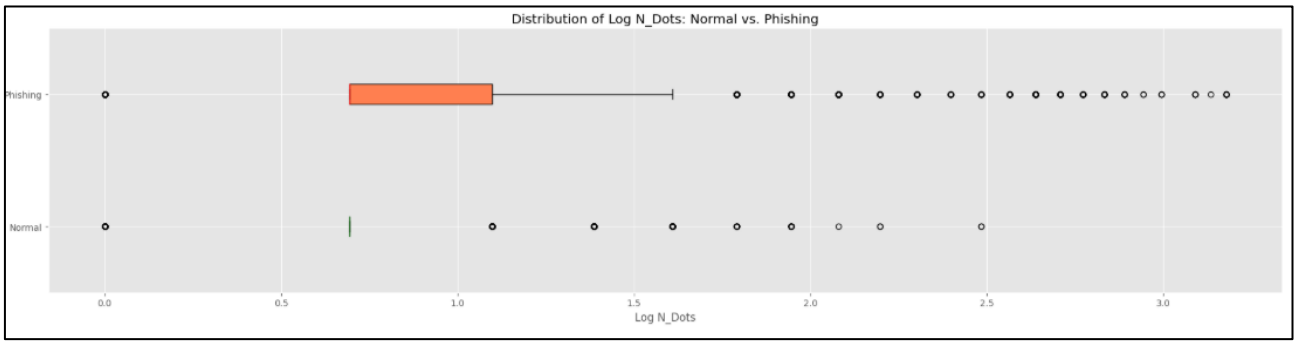


Рисунок 2.7 – Графік розподілу логарифмічних значень кількості крапок у URL для нормальних та фішингових сайтів.

На рисунку 2.8 представлено стовпчикову діаграму, яка порівнює середні значення різних ознак для фішингових і не фішингових сайтів, створену на основі описової статистики. Це дає змогу оцінити, які ознаки мають суттєві відмінності між категоріями, що є важливим для вибору релевантних параметрів для моделей машинного навчання.

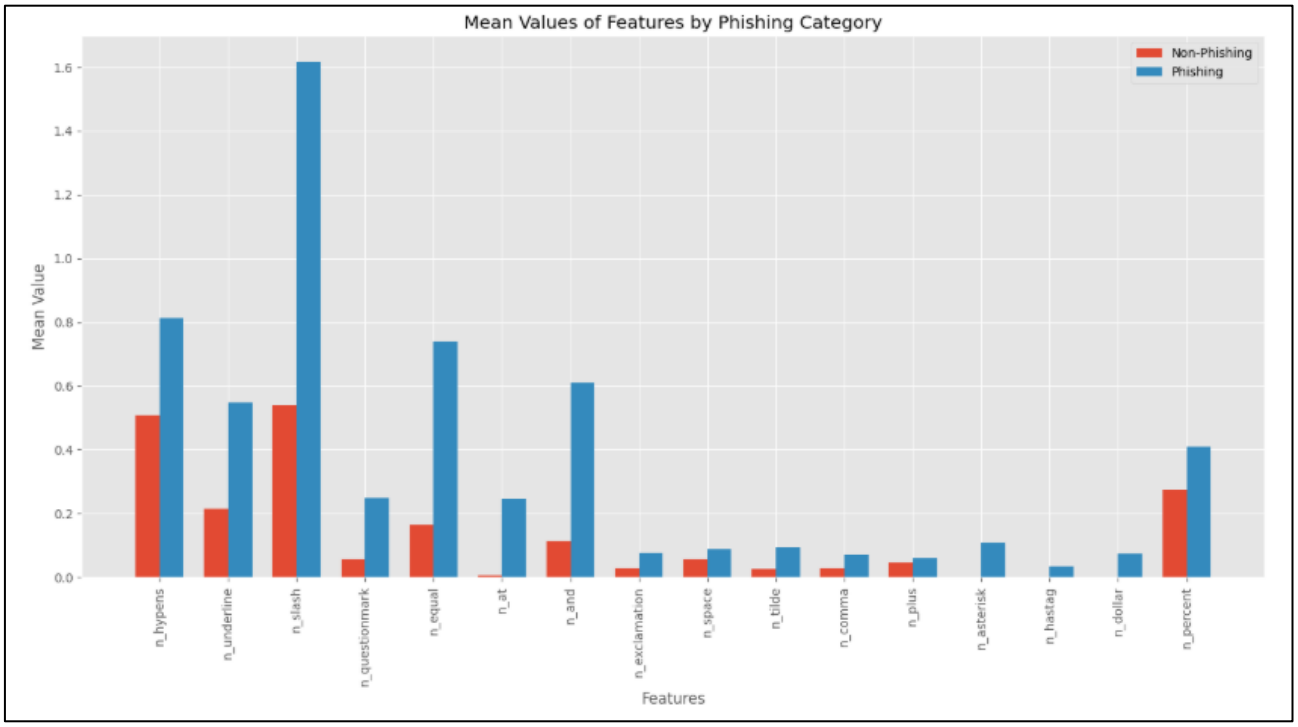


Рисунок 2.8 – Графік стовпчикової діаграми порівняння середніх значень різних ознак для фішингових та не фішингових сайтів

Рисунки 2.9 та 2.10 ілюструють теплову карту кореляційної матриці, де кожна клітинка відображає коефіцієнт кореляції між двома змінними. Теплова карта дозволяє ідентифікувати пари ознак із сильною позитивною або негативною кореляцією, вказуючи на їхній потенційний взаємозв’язок, тоді як низька або нульова кореляція свідчить про відсутність лінійної залежності. Такий аналіз допомагає обрати ознаки, які найбільше впливають на класифікацію.

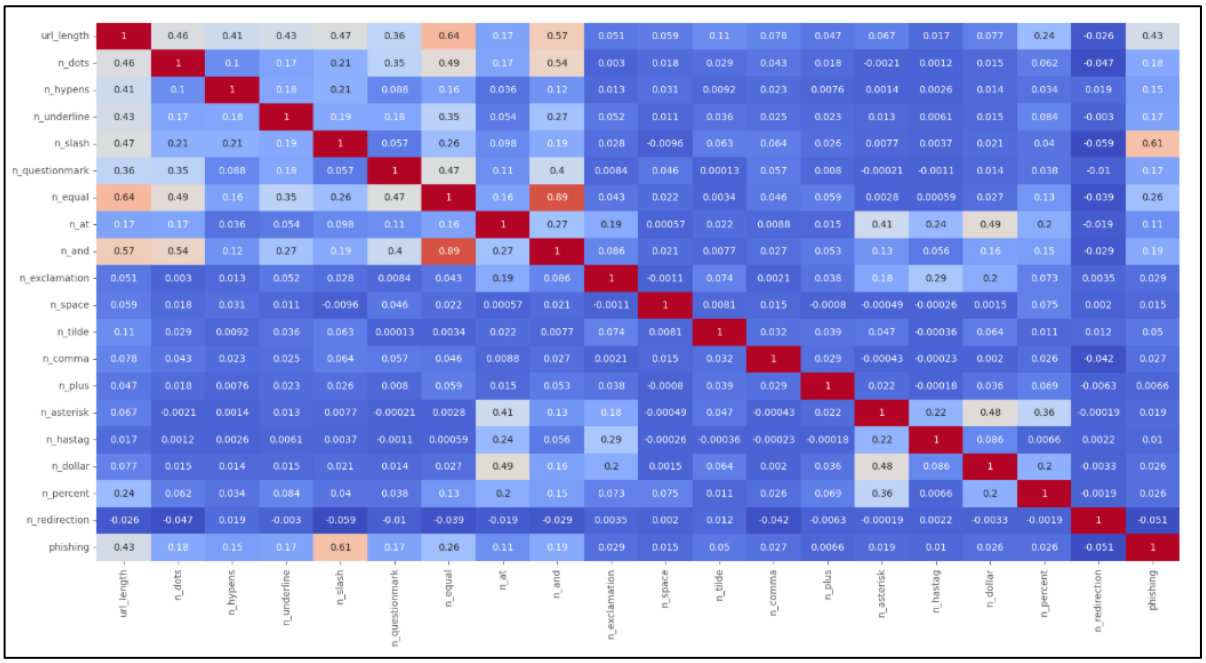


Рисунок 2.9 – Графік теплової карти кореляційної матриці

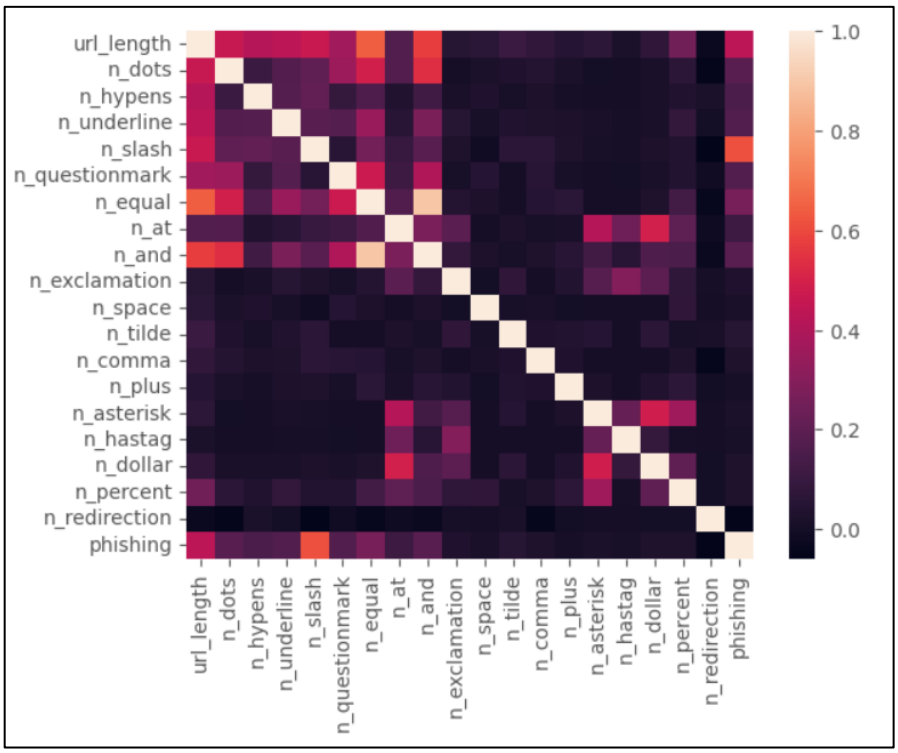


Рисунок 2.10 – Графік теплової карти кореляційної матриці

На основі проведеного аналізу можна зробити такі висновки:

– Фішингові URL часто містять специфічні символи, такі як дефіси, косі риски, знаки питання, знаки рівності чи амперсанди, для маскування під легітимні сайти;

– Ці характеристики є ключовими індикаторами для моделей машинного навчання, що дозволяють ефективно розрізняти шкідливі та безпечні URL-адреси;

– Розуміння закономірностей у розподілі та кореляції ознак відіграє важливу роль у підвищенні точності прогнозування фішингових загроз.

Далі в аналізі ми використовуємо графік кореляції Пірсона для оцінки лінійних залежностей між ознаками набору даних. Формула Пірсона визначається як:

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

де x_i і y_i – значення змінних,

а $\bar{x}$ і $\bar{y}$ – їхні середні значення;

Цей коефіцієнт r показує силу та напрямок лінійного зв’язку, варіюючись від -1 (сильна негативна кореляція) до 1 (сильна позитивна кореляція), з 0, що вказує на відсутність лінійного зв’язку. На рисунку 2.11 ми можемо побачити теплову карту з кольоровою шкалою, де кожна комірка відображає числове значення кореляції між парами ознак, наприклад, високу кореляцію між n_equal і n_questionmark або слабкий зв’язок phishing з n_space, що допомагає ідентифікувати ключові взаємозв’язки для подальшого моделювання.

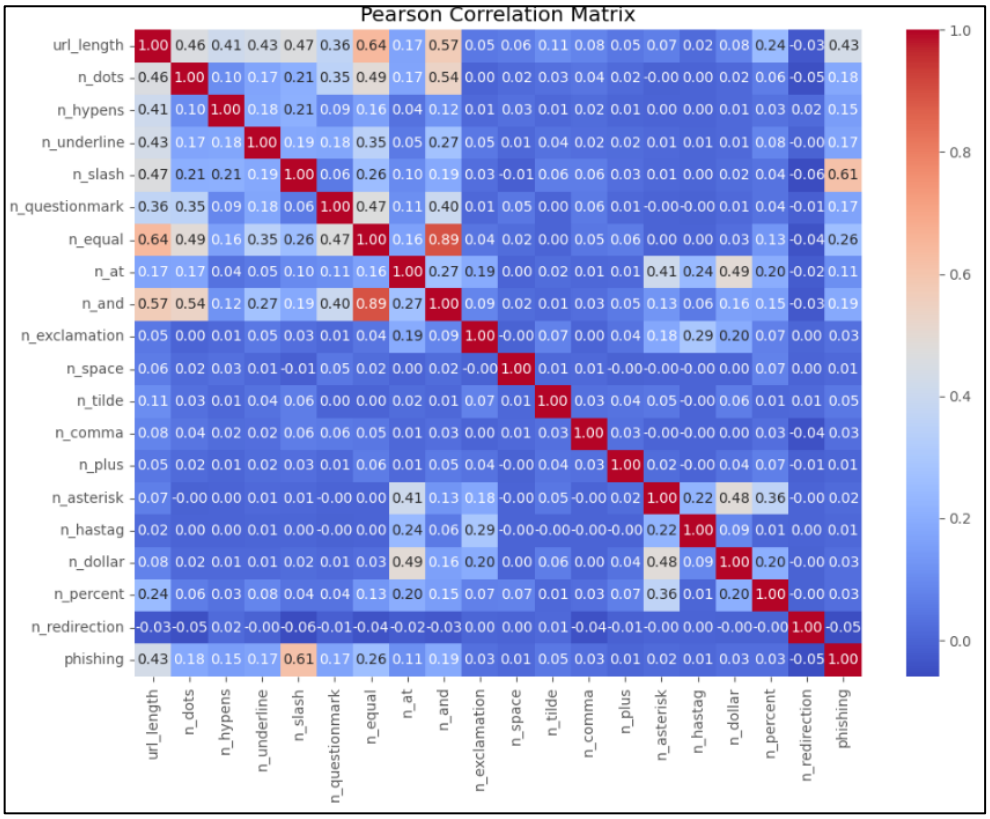


Рисунок 2.11 – Графік теплової карти кореляційної матриці Пірсона

Далі в аналізі ми використовуємо графік кореляції Спірмена для оцінки монотонних залежностей між ознаками набору даних. Формула Спірмена базується на рангах і визначається як:

$$r_s = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$$

де d_i – різниця між рангами пар змінних,

а n – кількість спостережень;

Цей коефіцієнт r_s показує силу та напрямок монотонного зв'язку, варіюючись від -1 (сильна негативна монотонна кореляція) до 1 (сильна позитивна монотонна кореляція), з 0, що вказує на відсутність монотонного зв'язку. На рисунку 2.12 ми можемо побачити теплову карту з кольоровою шкалою, де кожна комірка відображає числове значення кореляції між парами ознак, наприклад, високу кореляцію між n_equal і n_questionmark або помірний зв'язок phishing з n_redirection, що допомагає ідентифікувати ключові монотонні взаємозв'язки для подальшого моделювання.

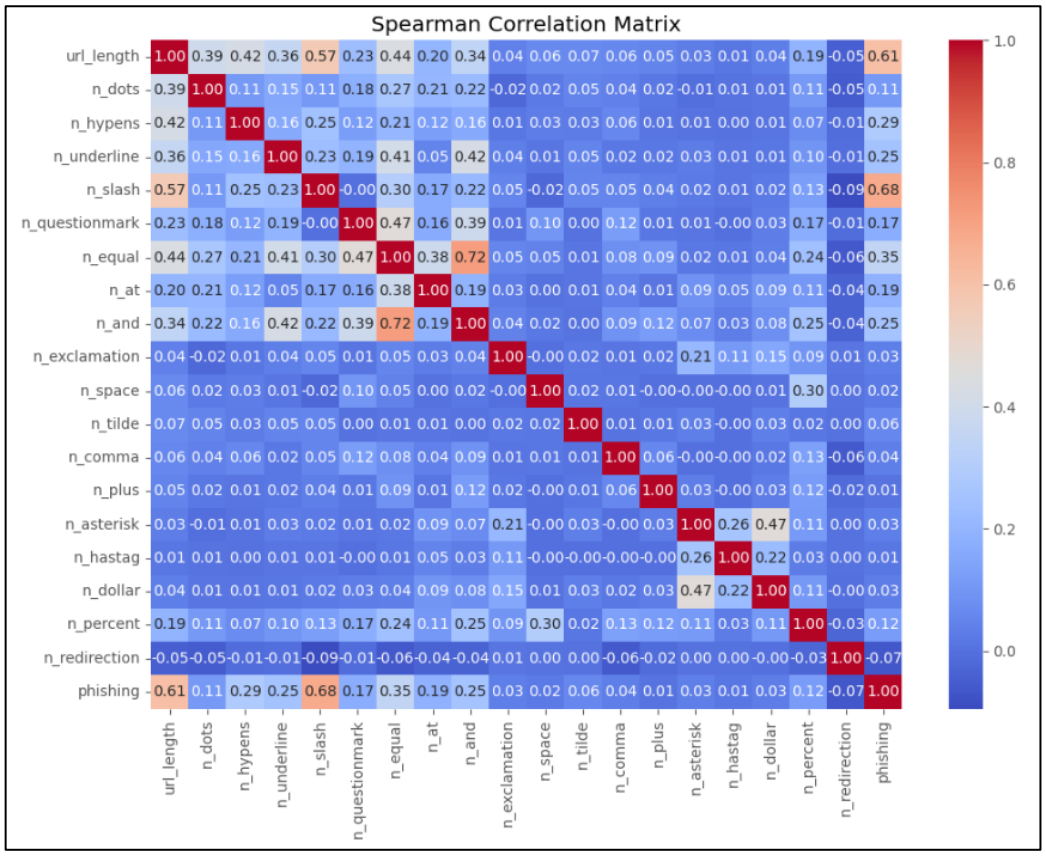


Рисунок 2.12 – Графік теплової карти кореляційної матриці Спірмена

2.3 Висновки

У даному розділі проведено комплексний аналіз датасету та розвідувальний аналіз даних. Для дослідження використано об'єднаний набір даних, що містить 100077 записів і 20 ознак, і підтверджено його готовність до обробки. Розвідувальний аналіз за допомогою описової статистики, boxplot діаграм розмаху та стовпчикових діаграм дозволив оцінити розподіл даних і встановити, що специфічні символи в URL є ключовими індикаторами фішингової активності. Детальний аналіз за коефіцієнтами виявив ключові лінійні та монотонні взаємозв'язки між ознаками, що стало основою для вибору параметрів для подальшого навчання моделей машинного навчання.

3 РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ ПЕРЕДБАЧЕННЯ ФІШИНГОВИХ САЙТІВ

3.1 Моделювання та передбачення

Для початку моделювання використаємо блок-схему алгоритму для візуалізації ключових етапів розробки інформаційної технології передбачення фішингових сайтів, як показано на рисунку 3.1. Алгоритм починається з ініціалізації системи та завантаження необхідних бібліотек для роботи з даними та моделями. Далі відбувається підготовка даних, що включає збір вихідних наборів даних, за якою слідує їх візуалізація для первинного аналізу та виявлення аномалій. Після цього здійснюється ключова обробка даних. Підготовлений набір даних обов'язково підлягає розбиттю на тренувальну та тестову вибірки. Наступні етапи є ітераційними: спочатку відбувається оптимізація моделей (наприклад, шляхом налаштування гіперпараметрів), а потім їх навчання на тренувальній вибірці. Система перевіряє, чи точність влаштовує: якщо результат незадовільний, цикл повертається до оптимізації; якщо точність досягнута, процес завершується і модель готова до використання.

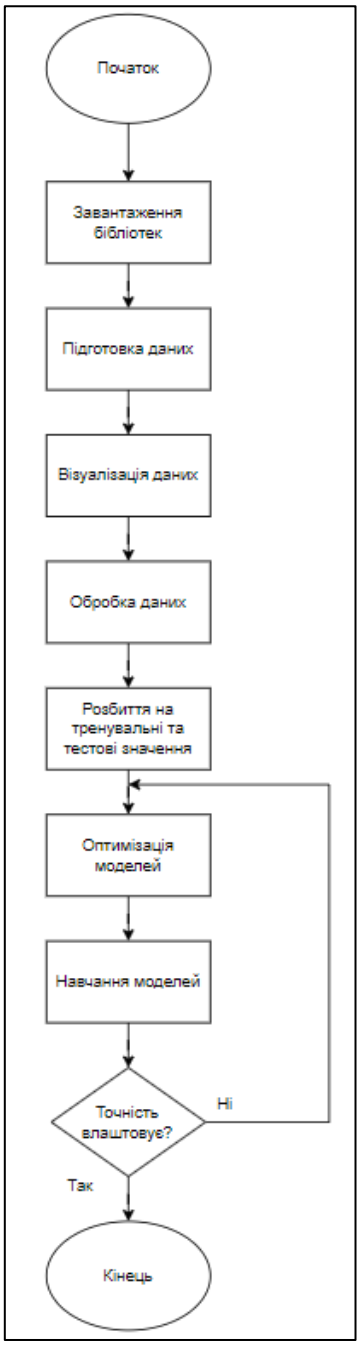


Рисунок 3.1 – Блок-схема алгоритму інформаційної технології

Діаграма сценаріїв використання візуалізує функціональні вимоги до інформаційної технології передбачення фішингових сайтів, визначаючи взаємодію двох основних ролей: користувача та адміністратора. Користувач використовує систему для перевірки URL. Його ключові функції включають введення даних URL для аналізу, валідацію вхідних даних, запуск передбачення та отримання результату передбачення. Він також може запросити результат важливості ознак для отримання пояснення рішення

моделі. Адміністратор відповідає за технічну підтримку та актуалізацію. Його основні функції - моніторинг стану системи та оновлення системи. Процес оновлення включає завдання перенавчання моделі та оновлення залежностей системи, а також управління даними для перенавчання для забезпечення постійної високої якості класифікації. На рисунку 3.2 представлено блок-схему use-case діаграми

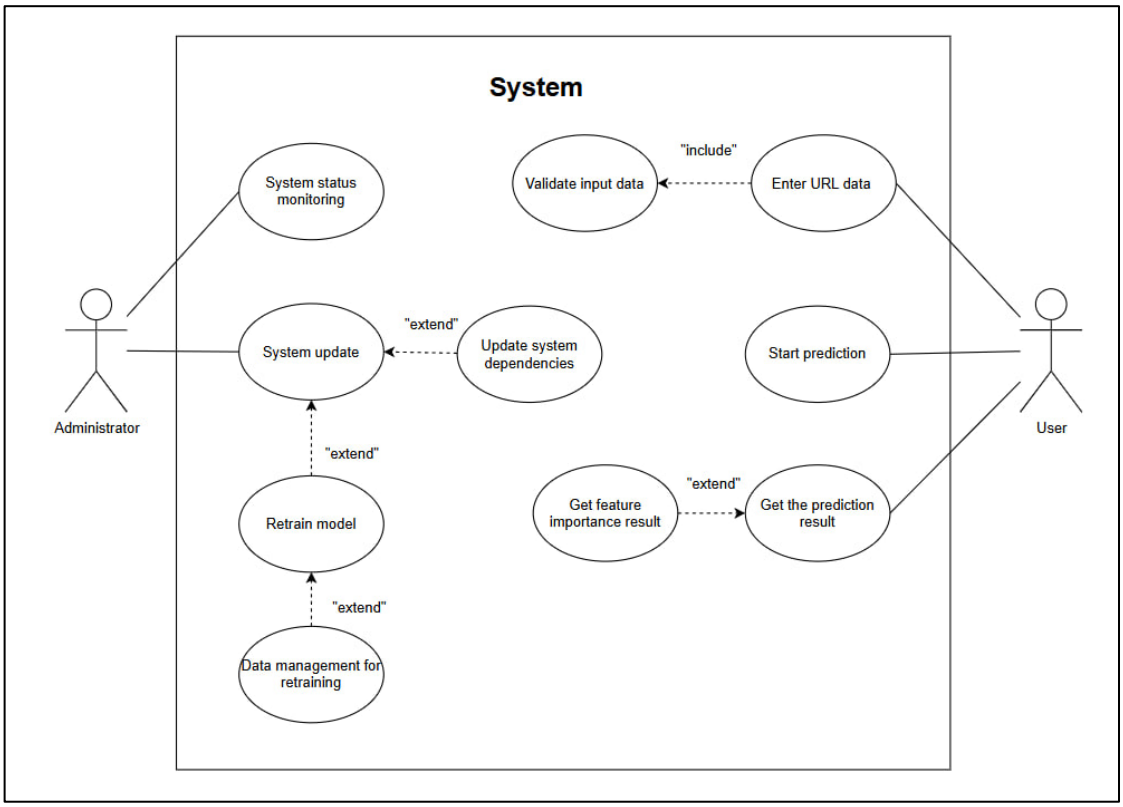


Рисунок 3.2 – Блок-схема Use-case діаграми

Діаграма компонентів представляє фізичну структуру інформаційної технології передбачення фішингових сайтів, відображаючи її основні програмні модулі та залежності між ними. Вона демонструє, що система складається з взаємопов'язаних компонентів. Процес починається з модуля джерел даних, який передає дані до модуля підготовки Даних. Останній залежить від бібліотек Pandas та NumPy і надає оброблені дані модулю навчання моделей. Модуль навчання використовує бібліотеки Scikit-learn та Pandas для навчання алгоритмів. Навчена модель передається до сервісу

прогнозування та модулю оцінки якості. Сервіс прогнозування взаємодіє з аналізом функцій, який, у свою чергу, надає дані для модуля візуалізації, що залежить від Seaborn. Ця архітектура підкреслює високий рівень модульності та чітке розділення відповідальності між компонентами. На рисунку 3.3 представлено блок-схему діаграми компонентів.

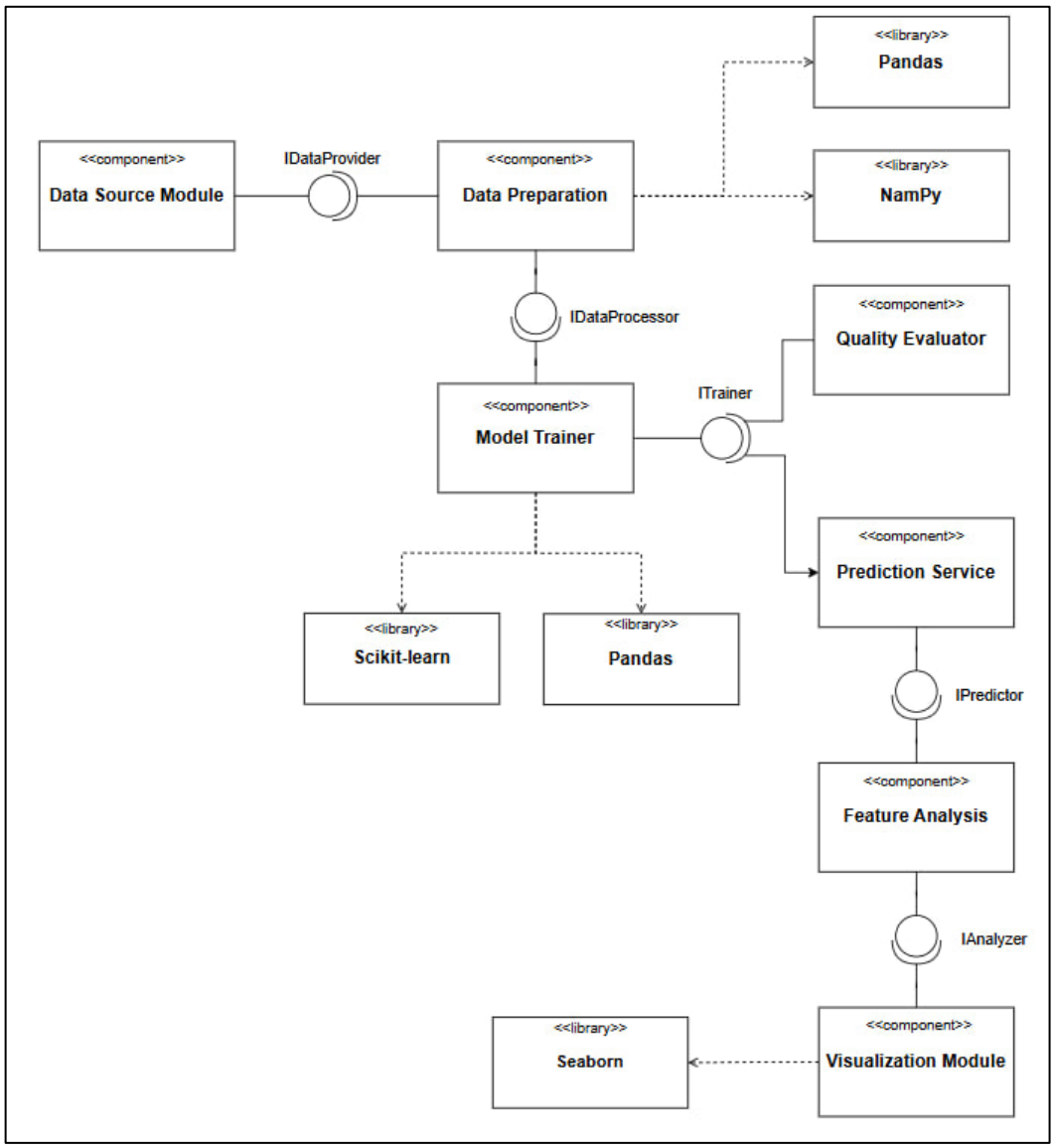


Рисунок 3.3 – Блок-схема діаграми компонентів

Продовжуючи розробку інформаційної технології, для передбачення фішингових сайтів ми використовуємо та досліджуємо ефективність низки потужних алгоритмів машинного навчання, включаючи Логістичну регресію, Decission Tree, Random forest, Gradient boosting trees, XGBoost Model,

LightGBM Model, ExtraTrees Model та K-Nearest Neighbors Model (KNN).

Логістична регресія є лінійною моделлю, яка використовує логістичну функцію для відображення довільного дійсного числа в діапазон від 0 до 1. Це дозволяє інтерпретувати вихідне значення як ймовірність належності об'єкта до певного класу (у нашому випадку - "Фішинг" або "Не-фішинг").

Переваги: Логістична регресія є простою, швидкою в навчанні та має високу інтерпретованість результатів, оскільки коефіцієнти моделі безпосередньо вказують на важливість відповідних ознак. Вона також ефективна для лінійно розділюваних даних.

Недоліки: Модель демонструє низьку ефективність, коли взаємозв'язки між ознаками є нелінійними або складними. У порівнянні з ансамблевими методами, Логістична регресія часто має меншу точність на складних, високорозмірних даних.

На рисунку 3.4 показано лістинг коду для методу логістичної регресії.

```
X_train, X_test, y_train, y_test = train_test_split(data, y, test_size=0.3, random_state=42)
models = {}
models['Logistic Regression'] = LogisticRegression(max_iter=1000)
models['Logistic Regression'].fit(X_train, y_train)

predictions_train = models['Logistic Regression'].predict(X_train)
predictions_test = models['Logistic Regression'].predict(X_test)

print(f"Train Accuracy Score: {accuracy_score(y_train, predictions_train)}")
print(f"Test Accuracy Score: {accuracy_score(y_test, predictions_test)}")
```

Рисунок 3.4 – Лістинг коду для методу логістичної регресії

Результати оцінки продуктивності моделі Логістичної регресії були отримані на незалежній тестовій вибірці та свідчать про її базову здатність до класифікації фішингових сайтів. Згідно з порівняльною таблицею, ключові показники демонструють наступну картину:

- Train Accuracy Score: 0.8582;
- Test Accuracy Score: 0.8566;
- Precision: 0.8717;

- Recall: 0.7157;
- F1-Score: 0.7861.

Матриця плутанини для Логістичної регресії візуально підтверджує ефективність моделі .У полі True Negative (Не-фішинг, правильно передбачено як Не-фішинг) зафіксовано 17810 випадків. У полі True Positive (Фішинг, правильно передбачено як Фішинг) зафіксовано 7909 випадків. Кількість False Negative, де фішинговий сайт був помилково класифікований як легітимний, становить 3141. Це значення є відносно високим і прямо корелює з низьким показником Recall, вказуючи на ризик пропуску реальних загроз. Кількість False Positive, де легітимний сайт був помилково класифікований як фішинговий, становить 1164. На рисунку 3.5 показано графік, який відображає результати класифікації моделі для тестових даних, наведені у вигляді матриці плутанини

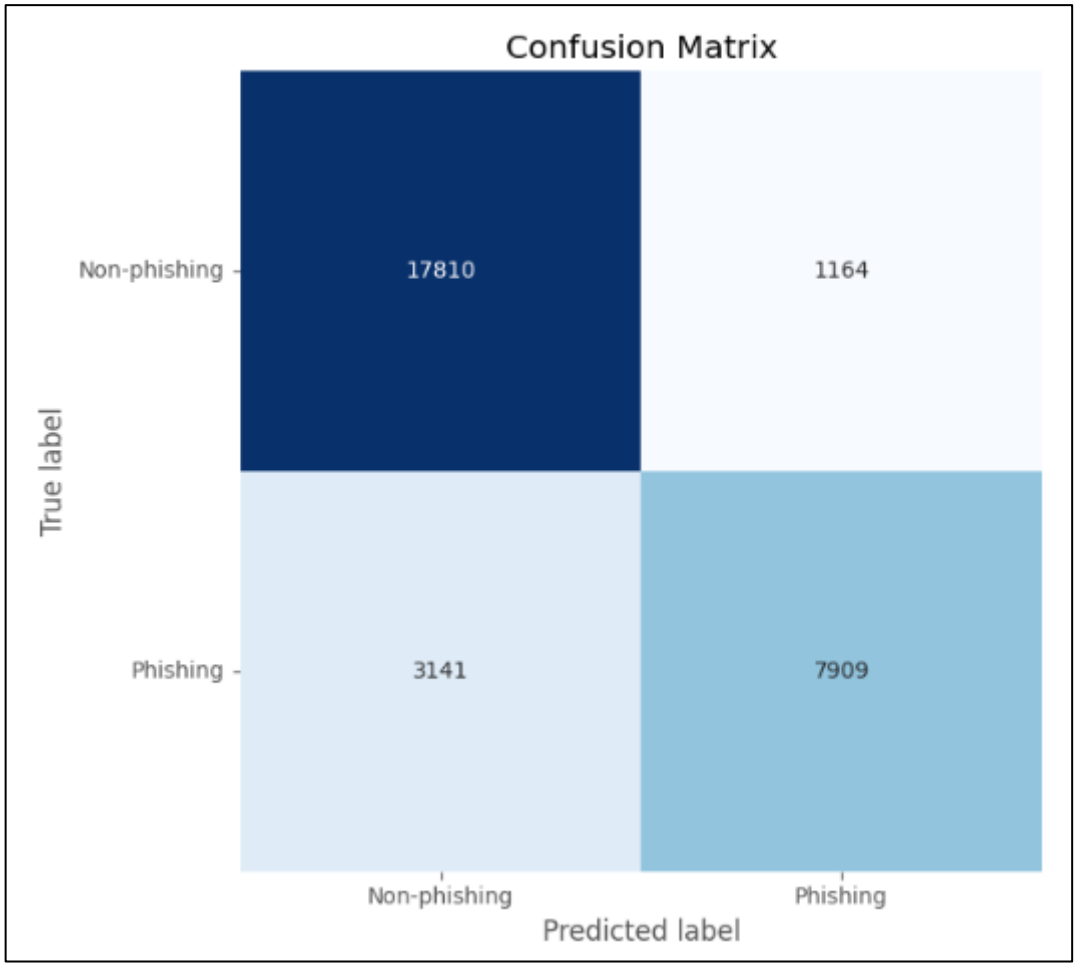


Рисунок 3.5 – Матриця плутанини методу логістичної регресії

Графік ROC-кривої ілюструє якість моделі при різних порогах класифікації. Крива розташована вище діагональної лінії, що вказує на те, що модель має кращу розрізнювальну здатність, ніж випадкове припущення. Площа під кривою AUC становить 0.93. Таке високе значення свідчить про загальну надійність моделі Логістичної регресії, незважаючи на відносно низьку F1-score. На рисунку 3.6 представлено ROC-криву, яка є стандартним графіком для оцінки продуктивності бінарного класифікатора, оскільки вона візуалізує залежність між чутливістю True Positive Rate та специфічністю False Positive Rate при варіації порогу класифікації.

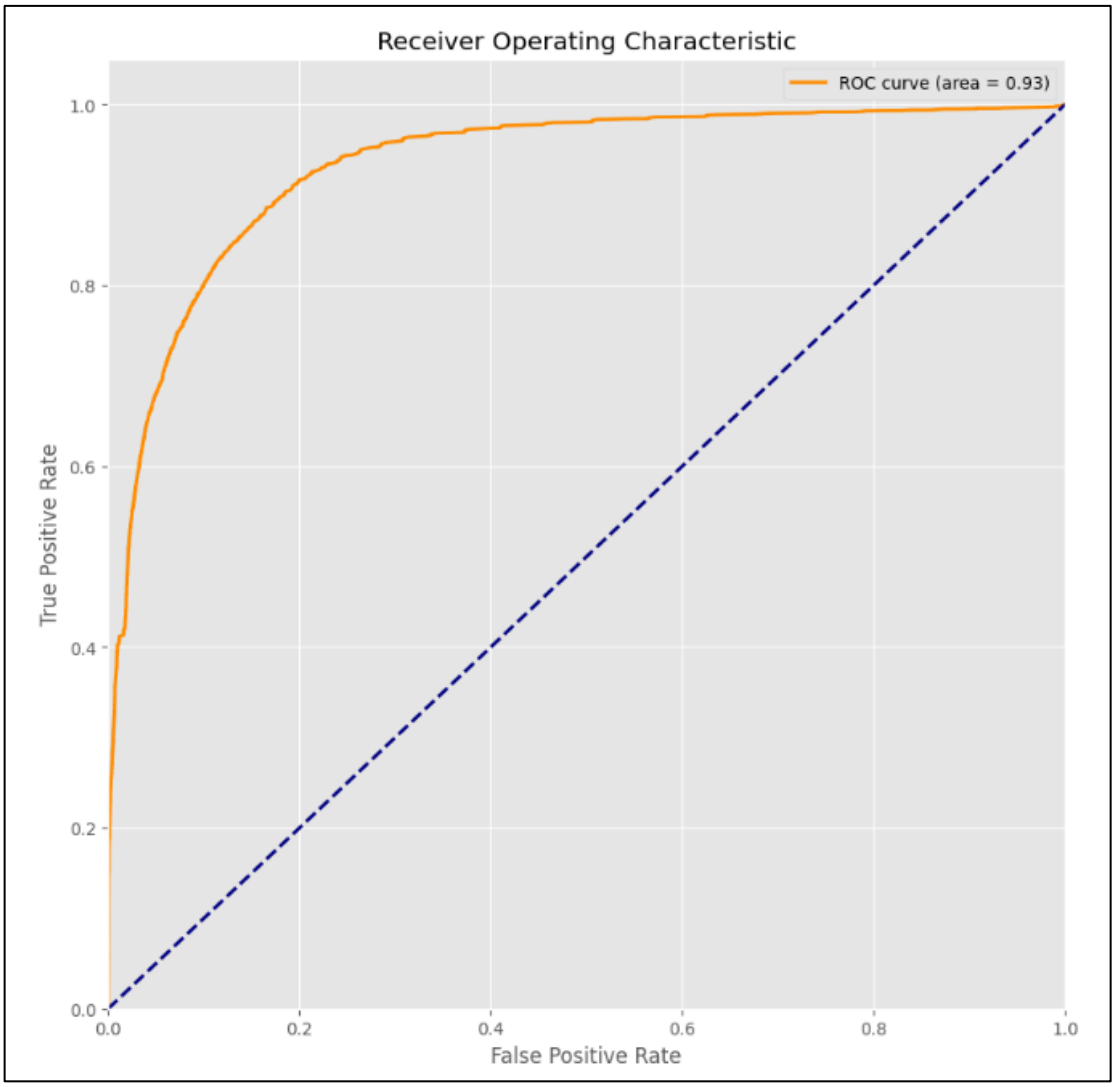


Рисунок 3.6 – Графік ROC кривої методу логістичної регресії

Модель Логістичної регресії, хоч і є найпростішою серед досліджуваних, демонструє прийнятну загальну точність та високу AUC. Однак, низький показник Recall та значна кількість False Negative вказують на те, що ця модель схильна пропускати фішингові сайти. Це є суттєвим недоліком. Тому, незважаючи на інтерпретованість, Логістична регресія не може бути рекомендована як основний алгоритм для фінальної інформаційної технології.

Дерево рішень працює шляхом послідовного розбиття набору даних на менші підмножини на основі значень ознак, формуючи ієрархічну структуру. Кожен внутрішній вузол представляє тест на ознаку, кожна гілка - результат тесту, а кожен кінцевий вузол - клас або значення рішення.

Переваги: Дерева рішень є надзвичайно інтуїтивно зрозумілими та легко інтерпретованими, оскільки вони імітують процес прийняття рішень людиною. Вони не вимагають нормалізації даних і можуть ефективно працювати зі змішаними типами даних.

Недоліки: Основний недолік - схильність до перенавчання, особливо при побудові глибокого дерева, що може призвести до поганого узагальнення на нових даних.

На рисунку 3.7 показано лістинг коду для методу Decision Tree.

```
models['Decision Tree'] = DecisionTreeClassifier(
    ccp_alpha=0.0001, criterion='gini', max_depth=32, min_samples_split=5, random_state=42
)
models['Decision Tree'].fit(X_train, y_train)

predictions_train = models['Decision Tree'].predict(X_train)
predictions_test = models['Decision Tree'].predict(X_test)

print(f"Train Accuracy Score: {accuracy_score(y_train, predictions_train)}")
print(f"Test Accuracy Score: {accuracy_score(y_test, predictions_test)}")
```

Рисунок 3.7 – Лістинг коду для методу Decision Tree

Результати оцінки продуктивності моделі Decision Tree були отримані на незалежній тестовій вибірці та свідчать про її базову здатність до класифікації фішингових сайтів. Згідно з порівняльною таблицею, ключові показники

демонструють наступну картину:

- Train Accuracy Score: 0.8900;
- Test Accuracy Score: 0.8864;
- Precision: 0.8591;
- Recall: 0.8270;
- F1-Score: 0.8427.

Матриця плутанини для моделі Decision Tree візуально підтверджує ефективність моделі. У полі True Negative (Не-фішинг, правильно передбачено як Не-фішинг) зафіксовано 17475 випадків. Це означає, що майже всі легітимні сайти були правильно ідентифіковані. У полі True Positive (Фішинг, правильно передбачено як Фішинг) зафіксовано 9138 випадків. Це є значним покращенням порівняно з Логістичною регресією 7909 випадків. Кількість False Negative, де фішинговий сайт був помилково класифікований як легітимний, становить 1912. Це значення значно нижче, ніж у Логістичної регресії 3141, що прямо корелює з вищим показником Recall. Це свідчить про суттєве зменшення ризику пропуску реальних загроз.

Кількість False Positive, де легітимний сайт був помилково класифікований як фішинговий, становить 1499. Цей показник є дещо вищим, ніж у Логістичної регресії 1164, але є прийнятним компромісом для досягнення вищої повноти. На рисунку 3.8 показано графік, який відображає результати класифікації моделі для тестових даних, наведені у вигляді матриці плутанини.

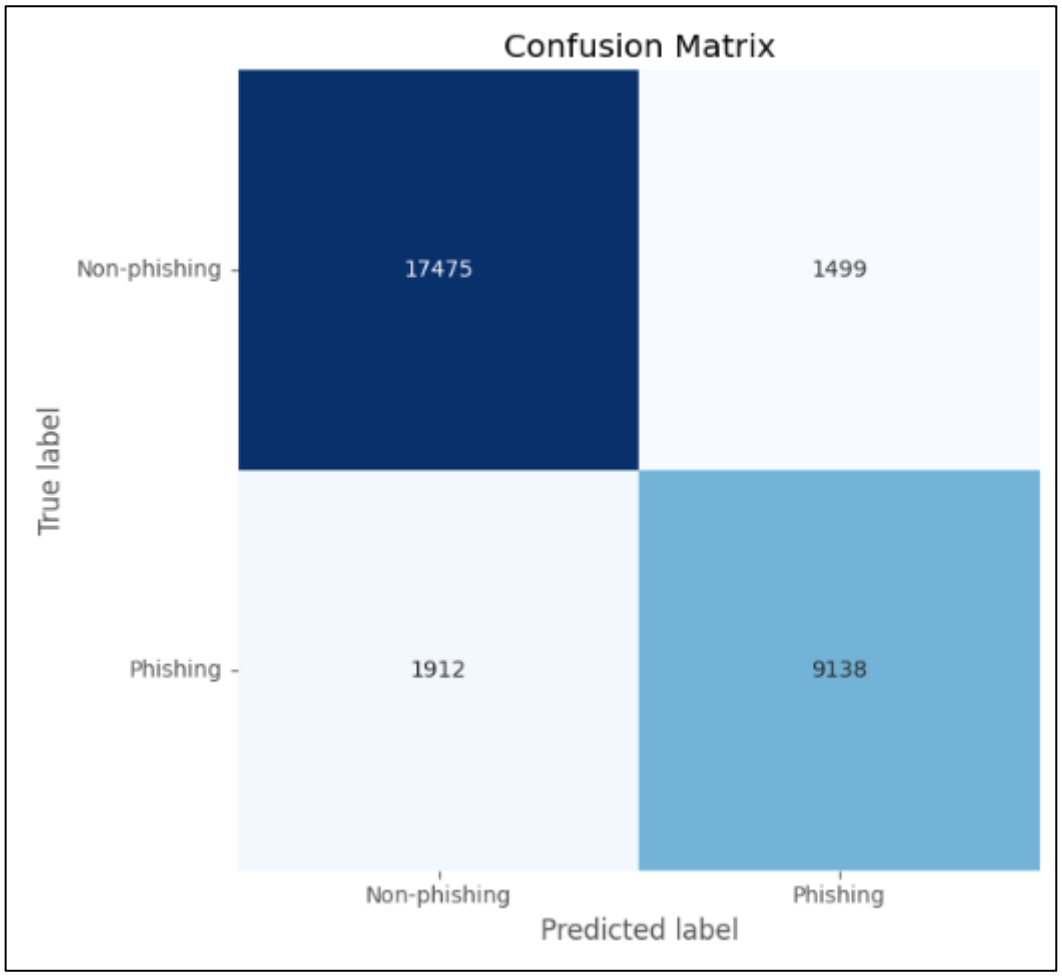


Рисунок 3.8 – Матриця плутанини методу Decision Tree

Графік ROC-кривої для моделі Дерева рішень демонструє відмінну розділювальну здатність. Площа під кривою AUC становить 0.95, що є вищим показником, ніж у Логістичної регресії. Це вказує на те, що модель Decision Tree має високу здатність до розрізнення класів "Фішинг" та "Не-фішинг" при будь-якому порозі класифікації. На рисунку 3.9 представлено ROC-криву, яка є стандартним графіком для оцінки продуктивності бінарного класифікатора, оскільки вона візуалізує залежність між чутливістю True Positive Rate та специфічністю False Positive Rate при варіації порогу класифікації.

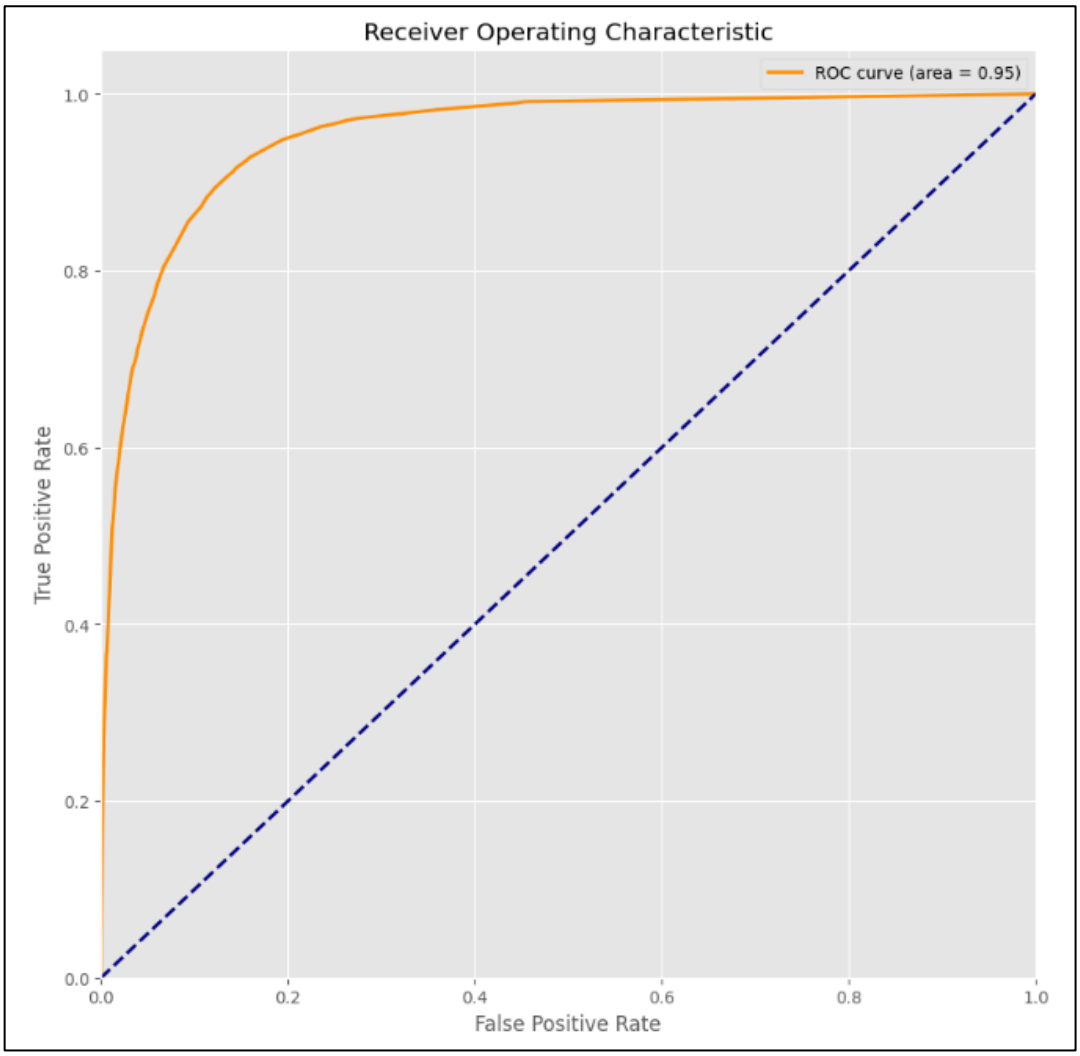


Рисунок 3.9 – Графік ROC кривої методу Decision Tree

Для розуміння принципів прийняття рішень моделлю Decision Tree був проведений аналіз важливості ознак, результати якого наведено на рисунку 3.10. Аналіз демонструє, що не всі ознаки мають однаковий вплив на кінцеве передбачення. Довжина URL виявилася найбільш важливою, її значення становить 0.665. Це підкреслює, що довжина веб-адреси є головним і найбільш надійним індикатором фішингу, оскільки зловмисники часто створюють аномально довгі URL-адреси. Кількість скісних рисок друга за важливістю ознака з показником 0.223. Велика кількість слешів може свідчити про спробу "замаскувати" справжній домен у глибині шляху. Решта ознак, такі як кількість крапок, дефісів та знаків питання, мають значно меншу вагу, хоча вони також беруть участь у формуванні кінцевого рішення.

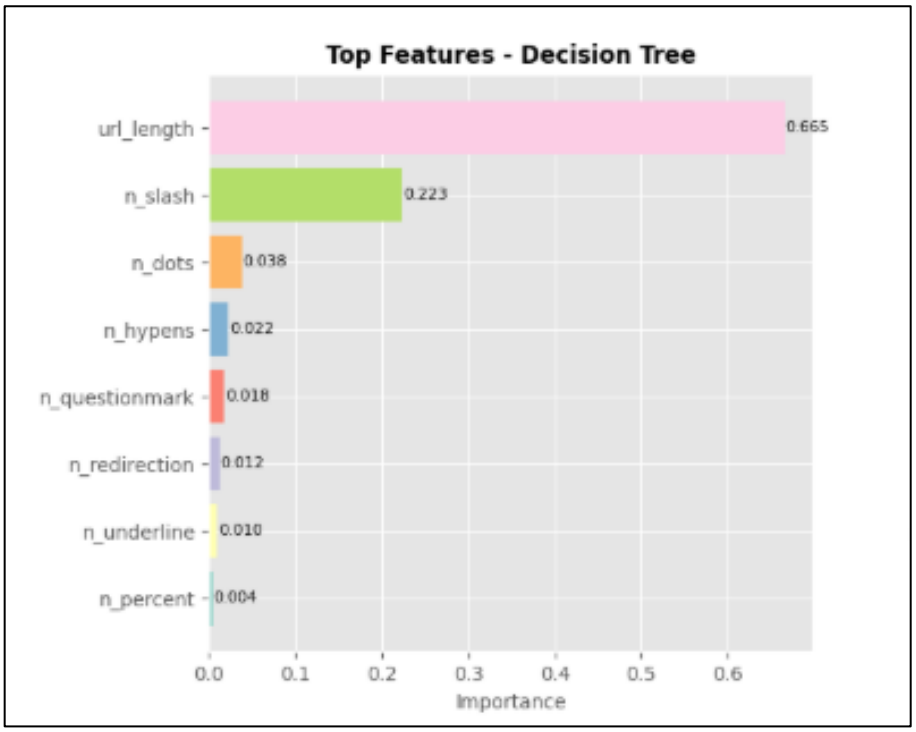


Рисунок 3.10 – Графік важливості ознак методу Decision Tree

Модель Decision Tree демонструє значне покращення ефективності порівняно з Логістичною регресією та зменшення кількості пропущених загроз роблять цю модель набагато якіснішою та придатною для вирішення поставленої задачі. Незважаючи на те, що вона є хорошою моделлю, наступні ансамблеві методи можуть забезпечити ще вищу точність.

Random Forest є ансамблевим методом, який будує велику кількість незалежних Дерев рішень. Кінцеве передбачення класу визначається шляхом голосування більшості між усіма окремими деревами. Це поєднання усереднює результати, що дозволяє значно знизити ризик перенавчання та підвищити точність узагальнення.

Переваги: Висока точність та стабільність результатів, відмінна стійкість до перенавчання. Модель також добре справляється з великими наборами даних і може оцінювати важливість ознак.

Недоліки: Через велику кількість побудованих дерев час навчання та обчислювальні ресурси значно вищі, ніж у одиничного Decision Tree. Модель є менш інтерпретованою порівняно з простим Decision Tree.

На рисунку 3.11 показано лістинг коду для методу Random Forest.

```
models['Random Forest'] = RandomForestClassifier(
    n_estimators=80, max_depth=18, max_features='sqrt', min_samples_split=12, criterion='gini'
)
models['Random Forest'].fit(X_train, y_train)

predictions_train = models['Random Forest'].predict(X_train)
predictions_test = models['Random Forest'].predict(X_test)

print(f"Train Accuracy Score: {accuracy_score(y_train, predictions_train)}")
print(f"Test Accuracy Score: {accuracy_score(y_test, predictions_test)}")
```

Рисунок 3.11 – Лістинг коду для методу Random Forest

Результати оцінки продуктивності моделі Random Forest були отримані на незалежній тестовій вибірці та свідчать про її базову здатність до класифікації фішингових сайтів. Згідно з порівняльною таблицею, ключові показники демонструють наступну картину:

- Train Accuracy Score: 0.9085;
- Test Accuracy Score: 0.8950;
- Precision: 0.8570;
- Recall: 0.8576;
- F1-Score: 0.8573.

Матриця плутанини для моделі Random Forest показує у полі True Negative (Не-фішинг, правильно передбачено як Не-фішинг) зафіксовано 17393 випадки. У полі True Positive (Фішинг, правильно передбачено як Фішинг) зафіксовано 9477 випадків, що є найвищим показником серед перших трьох моделей, підтверджуючи високий Recall .Кількість False Negative, де фішинговий сайт був помилково класифікований як легітимний, становить 1573. Це найнижча кількість серед усіх моделей до цього моменту, що є критично важливим для системи безпеки, оскільки мінімізує пропуск загроз. Кількість False Positive, де легітимний сайт був помилково класифікований як фішинговий, становить 1581. Це прийнятна кількість, яка є компромісом для

забезпечення високої повноти. На рисунку 3.12 показано графік, який відображає результати класифікації моделі для тестових даних, наведені у вигляді матриці плутанини.

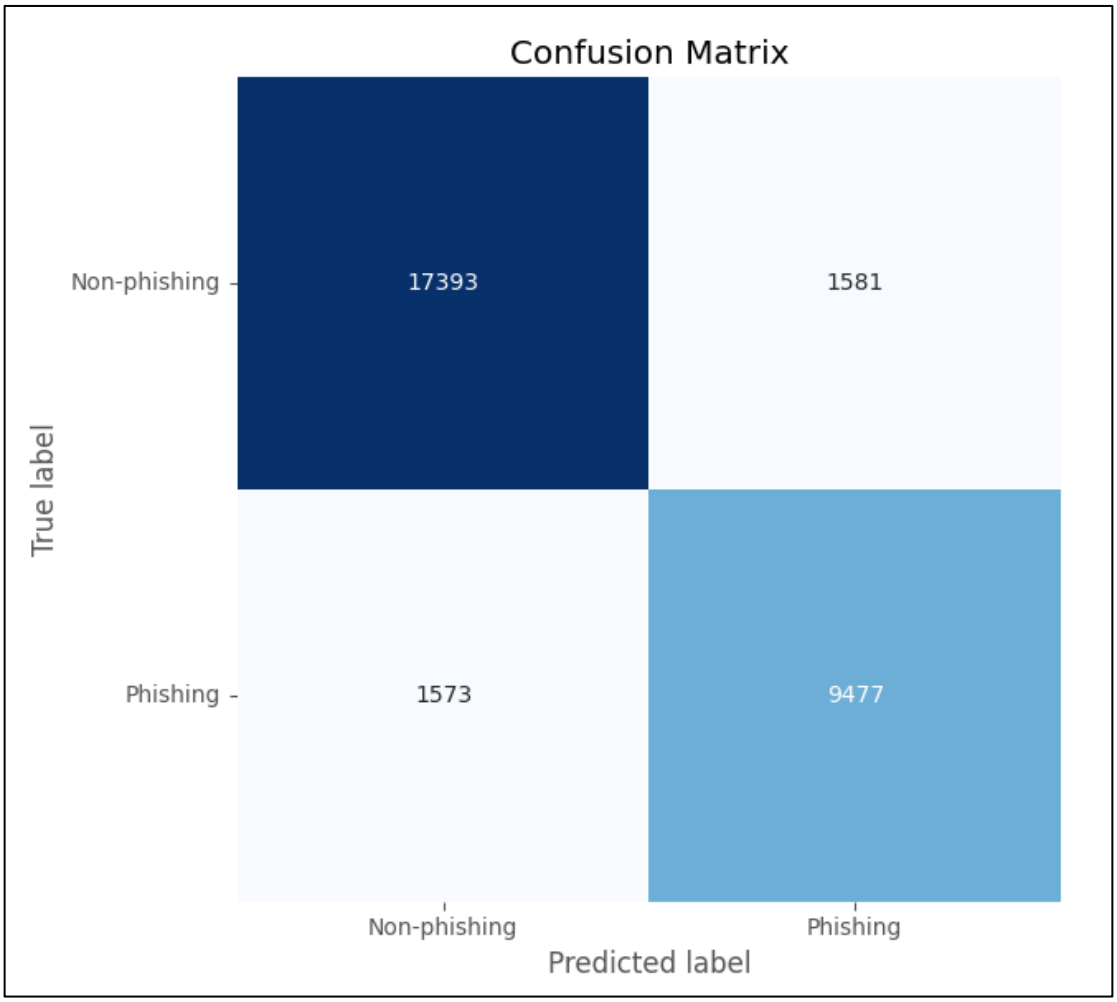


Рисунок 3.12 – Матриця плутанини методу Random Forest

Графік ROC-кривої для Random Forest практично ідеально прилягає до верхнього лівого кута. Площа під кривою AUC становить 0.96, що є найвищим значенням серед усіх розглянутих моделей до цього моменту. Це свідчить про відмінну розрізнювальну здатність та здатність моделі Random Forest розрізняти фішингові та легітимні сайти при будь-якому порозі класифікації. На рисунку 3.13 представлено ROC-криву, яка є стандартним графіком для оцінки продуктивності бінарного класифікатора, оскільки вона візуалізує залежність між чутливістю True Positive Rate та специфічністю False Positive

Rate при варіації порогу класифікації.

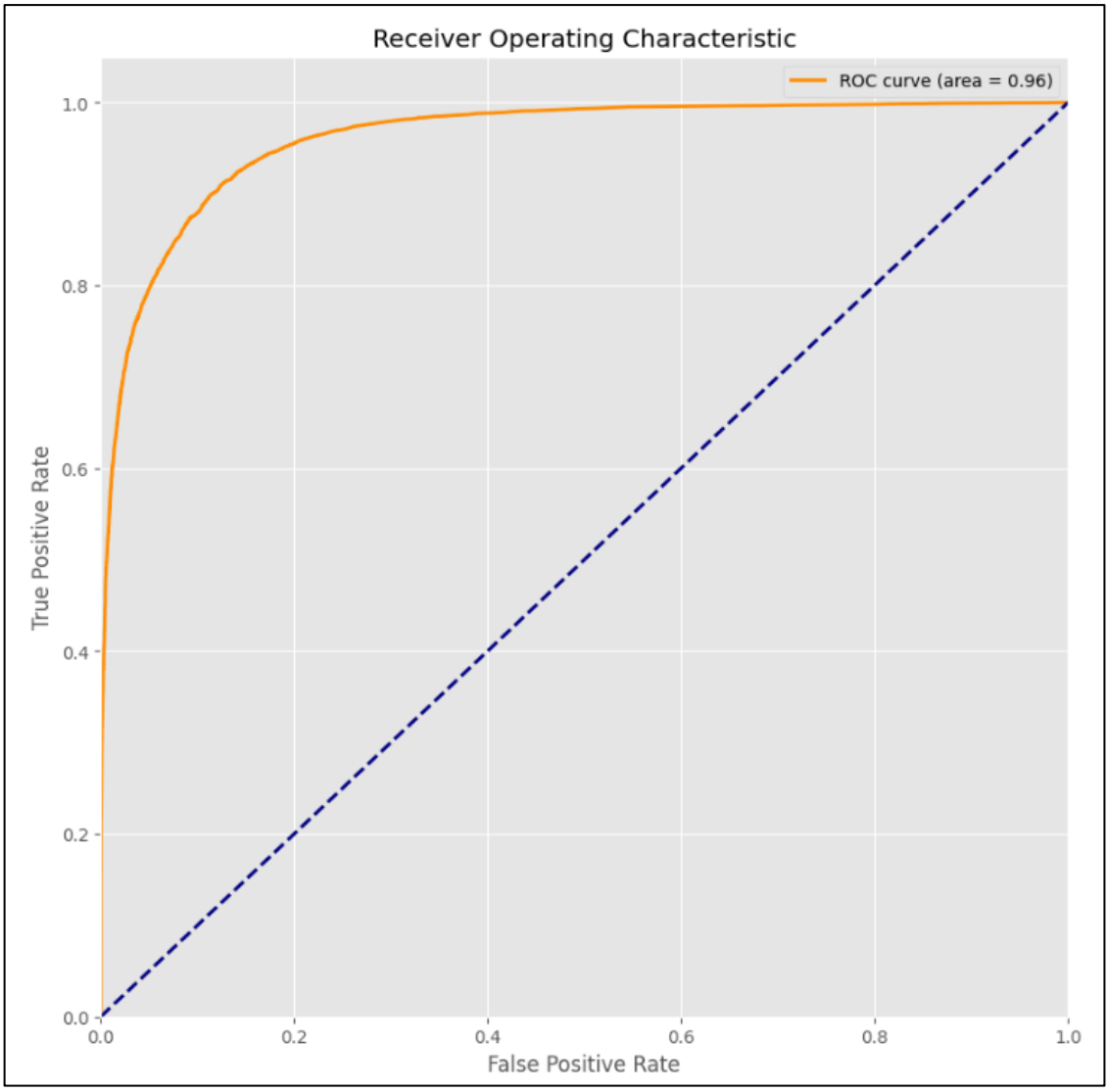


Рисунок 3.13 – Графік ROC кривої методу Random Forest

Аналіз важливості ознак для моделі Random Forest наведено на рисунку 3.14. Довжина URL залишається найважливішою ознакою з вагою 0.386. Кількість скісних рисок також зберігає високу важливість, її значення становить 0.359. Цей аналіз підтверджує, що модель покладається на структурні особливості URL для прийняття рішення, при цьому використовуючи більш збалансований набір індикаторів, ніж просте Decision Tree.

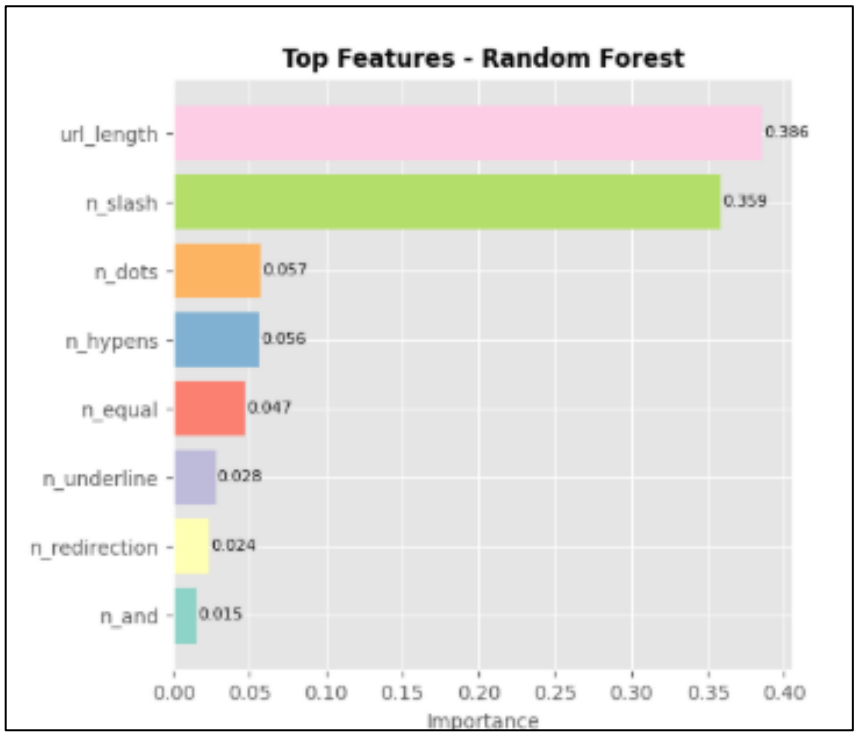


Рисунок 3.14 – Графік важливості ознак методу Random Forest

Gradient Boosting Trees будує моделі послідовно, де кожне наступне Decission Tree навчається на помилках, допущених попередніми деревами. Це дозволяє моделі постійно коригувати свої помилки та з часом фокусуватися на найбільш складних для класифікації випадках.

Переваги: Висока точність і ефективність, часто дає результати, близькі до найкращих серед усіх алгоритмів машинного навчання. Добре працює зі складними нелінійними взаємозв'язками.

Недоліки: Чутливий до перенавчання, якщо параметри налаштовані неправильно. Значно вищі вимоги до часу навчання та обчислювальних ресурсів порівняно з Random Forest.

На рисунку 3.15 показано лістинг коду для методу Gradient Boosting Trees.

```
models['Gradient Boosting'] = GradientBoostingClassifier(
    n_estimators=200, learning_rate=0.01, max_depth=12, subsample=0.8, max_features=0.5, random_s
tate=42
)
models['Gradient Boosting'].fit(X_train, y_train)

predictions_train = models['Gradient Boosting'].predict(X_train)
predictions_test = models['Gradient Boosting'].predict(X_test)

print(f"Train Accuracy Score: {accuracy_score(y_train, predictions_train)}")
print(f"Test Accuracy Score: {accuracy_score(y_test, predictions_test)}")
```

Рисунок 3.15 – Лістинг коду для методу Gradient Boosting Trees

Результати оцінки продуктивності моделі Gradient Boosting Trees були отримані на незалежній тестовій вибірці та свідчать про її базову здатність до класифікації фішингових сайтів. Згідно з порівняльною таблицею, ключові показники демонструють наступну картину:

- Train Accuracy Score: 0.9115;
- Test Accuracy Score: 0.8960;
- Precision: 0.8748;
- Recall: 0.8373;
- F1-Score: 0.8556.

Матриця плутанини для моделі Gradient Boosting показує у полі True Negative (Не-фішинг, правильно передбачено як Не-фішинг) зафіксовано 17650 випадків. У полі True Positive (Фішинг, правильно передбачено як Фішинг) зафіксовано 9252 випадки. Кількість False Negative, де фішинговий сайт був помилково класифікований як легітимний, становить 1798. Хоча це краще, ніж у Decision Tree, це все ж гірше, ніж у Random Forest. Це підтверджує, що Random Forest краще мінімізує пропуск загроз. Кількість False Positive, де легітимний сайт був помилково класифікований як фішинговий, становить 1324. Це найнижча кількість серед ансамблевих методів, що узгоджується з найвищим показником Precision. На рисунку 3.16 показано графік, який відображає результати класифікації моделі для тестових даних, наведені у вигляді матриці плутанини.

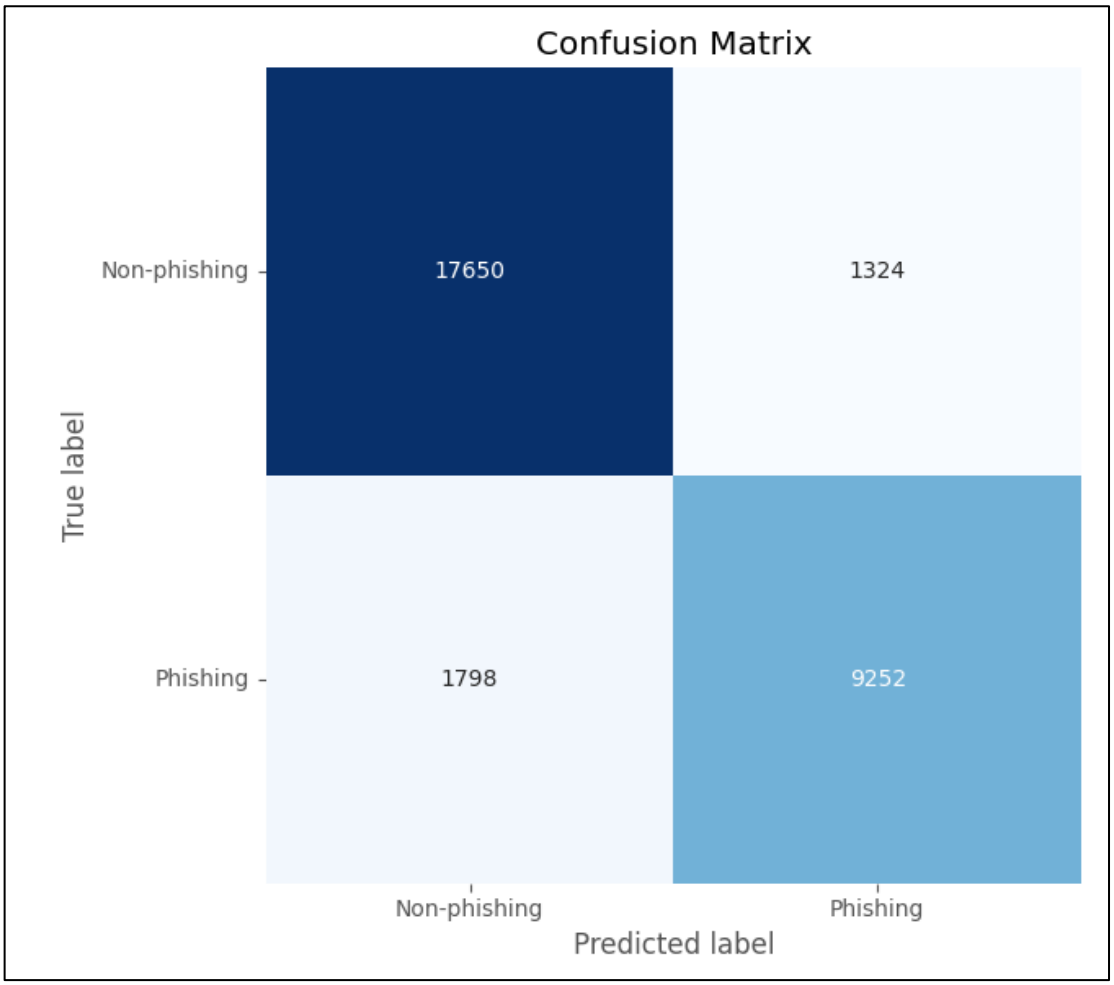


Рисунок 3.16 – Матриця плутанини методу Gradient Boosting Trees

Графік ROC-кривої для Gradient Boosting майже ідентична ROC-кривій Random Forest. Площа під кривою AUC становить 0.96, що підтверджує відмінну загальну розділювальну здатність моделі, рівну найкращому результату, отриманому Random Forest. На рисунку 3.17 представлено ROC-криву, яка є стандартним графіком для оцінки продуктивності бінарного класифікатора, оскільки вона візуалізує залежність між чутливістю True Positive Rate та специфічністю False Positive Rate при варіації порогу класифікації.

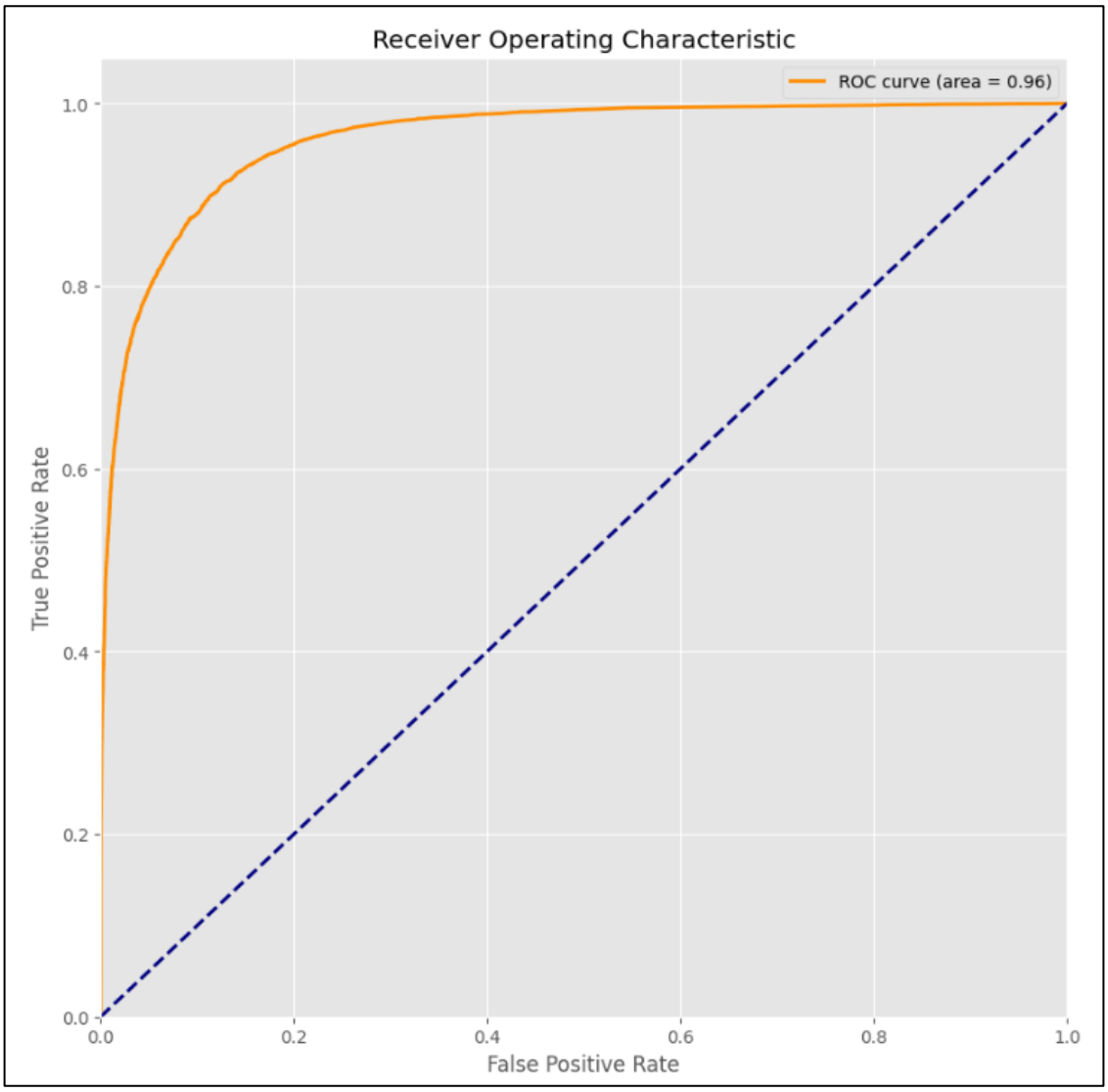


Рисунок 3.17 – Графік ROC кривої методу Gradient Boosting Trees

Аналіз важливості ознак для Gradient Boosting наведено на рисунку 3.18. Довжина URL є найважливішою ознакою з вагою 0.426. Кількість скісних рисок зберігає високу важливість, її значення становить 0.365. Як і в інших ансамблевих методах, модель Gradient Boosting чітко фокусується на структурних аномаліях URL як на основних індикаторах фішингу, з невеликою різницею у важливості між двома лідерами.

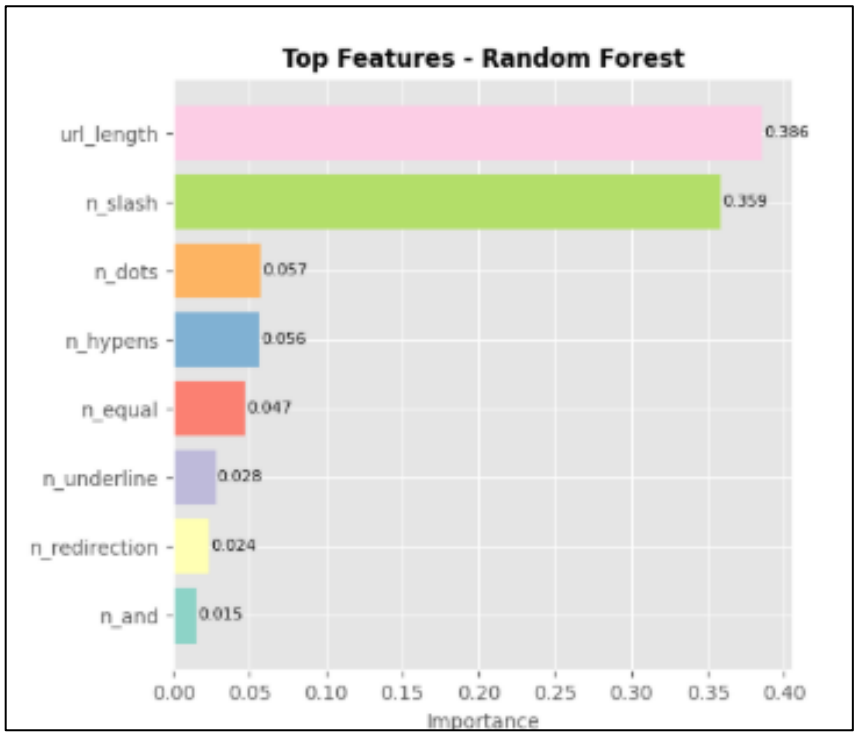


Рисунок 3.18 – Графік важливості ознак методу Gradient Boosting Trees

Модель Gradient Boosting Trees продемонструвала найвищу точність та найвищий Precision серед усіх розглянутих методів, а також високу AUC. Однак, через дещо нижчий Recall та вищу кількість False Negative порівняно з Random Forest, вона не є абсолютно найкращою для критично важливої задачі, де мінімізація пропуску загроз Recall є пріоритетом.

XGBoost є потужною ансамблевою технікою, яка використовує послідовну побудову Decision Tree. Ключова відмінність полягає в оптимізації: XGBoost використовує другий порядок градієнта Гессіан у функції втрат і включає регулярні компоненти для запобігання перенавчанню. Це забезпечує вищу точність і швидкість обчислень порівняно з традиційним Gradient Boosting Trees.

Переваги: Визнаний лідер у багатьох змаганнях з машинного навчання, відомий своєю винятковою точністю, швидкістю виконання та масштабованістю. Має вбудовані механізми для ефективної боротьби з перенавчанням.

Недоліки: Через складність алгоритму є менш інтерпретованою

моделлю, ніж прості дерева. Вимагає ретельного налаштування гіперпараметрів для досягнення оптимальної продуктивності.

На рисунку 3.19 показано лістинг коду для методу XGBoost.

```
models['XGBoost'] = XGBClassifier(
    n_estimators=200,
    max_depth=9,
    learning_rate=0.1,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42,
    eval_metric='logloss'
)

start_time = time.time()
models['XGBoost'].fit(X_train, y_train)

predictions_train = models['XGBoost'].predict(X_train)
predictions_test = models['XGBoost'].predict(X_test)

print(f"Train Accuracy Score: {accuracy_score(y_train, predictions_train):.4f}")
print(f"Test Accuracy Score: {accuracy_score(y_test, predictions_test):.4f}")
```

Рисунок 3.19 – Лістинг коду для методу XGBoost

Результати оцінки продуктивності моделі XGBoost були отримані на незалежній тестовій вибірці та свідчать про її базову здатність до класифікації фішингових сайтів. Згідно з порівняльною таблицею, ключові показники демонструють наступну картину:

- Train Accuracy Score: 0.9100;
- Test Accuracy Score: 0.8961;
- Precision: 0.8679;
- Recall: 0.8466;
- F1-Score: 0.8571.

Матриця плутанини для моделі XGBoost показує у полі True Negative (Не-фішинг, правильно передбачено як Не-фішинг) зафіксовано 17550 випадків. У полі True Positive (Фішинг, правильно передбачено як Фішинг) зафіксовано 9355 випадків. Кількість False Negative, де фішинговий сайт був пропущений, становить 1695. Це є кращим результатом, ніж у Gradient

Boosting Trees, але все ж гіршим, ніж у Random Forest. Кількість False Positive, де легітимний сайт був помилково заблокований, становить 1424. Це значення знаходиться в середині діапазону ансамблевих методів. На рисунку 3.20 показано графік, який відображає результати класифікації моделі для тестових даних, наведені у вигляді матриці плутанини.

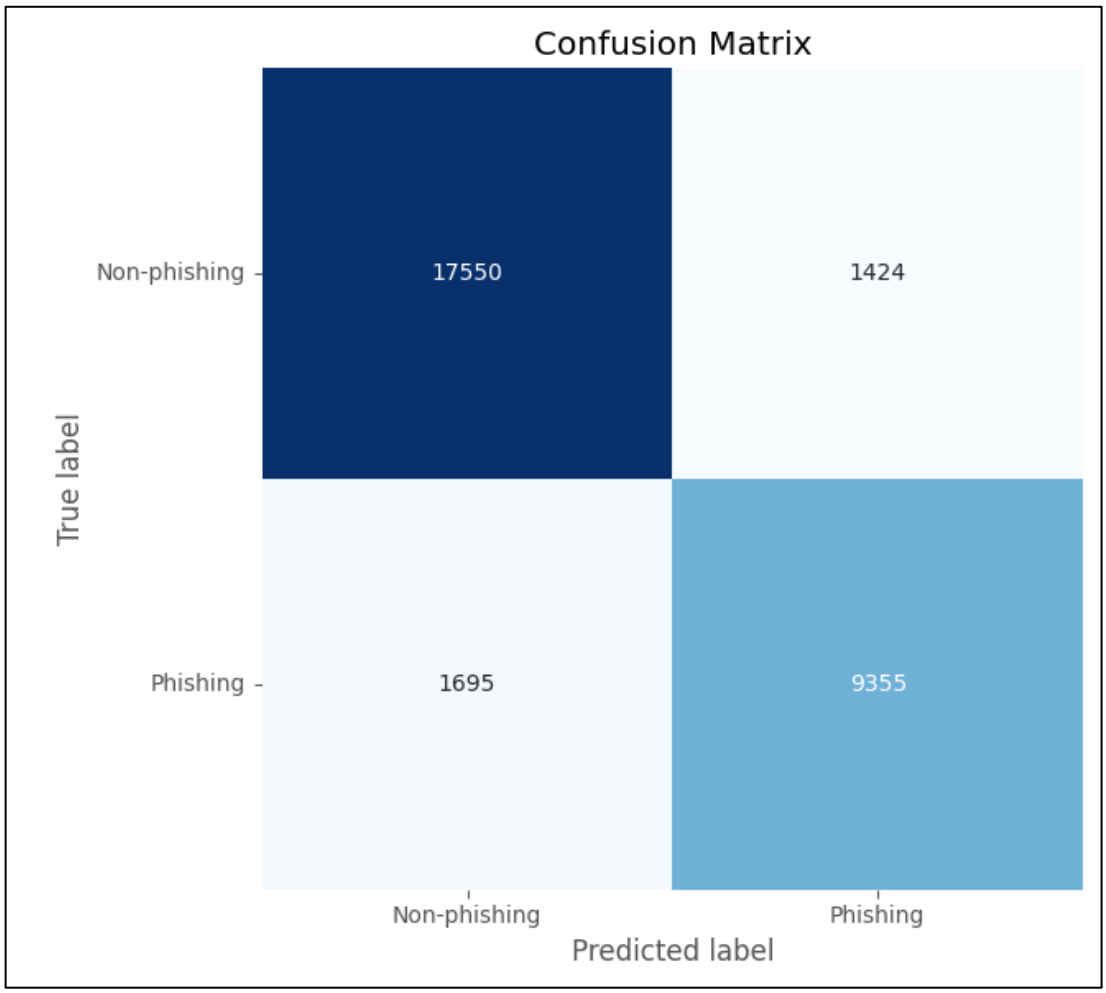


Рисунок 3.20 – Матриця плутанини методу XGBoost

Графік ROC-кривої для XGBoost демонструє відмінну розділювальну здатність. Площа під кривою AUC становить 0.96, що відповідає найкращому результату, отриманому Random Forest та Gradient Boosting, і свідчить про високу загальну якість прогностичної здатності моделі. На рисунку 3.21 представлено ROC-криву, яка є стандартним графіком для оцінки продуктивності бінарного класифікатора, оскільки вона візуалізує залежність

між чутливістю True Positive Rate та специфічністю False Positive Rate при варіації порогу класифікації.

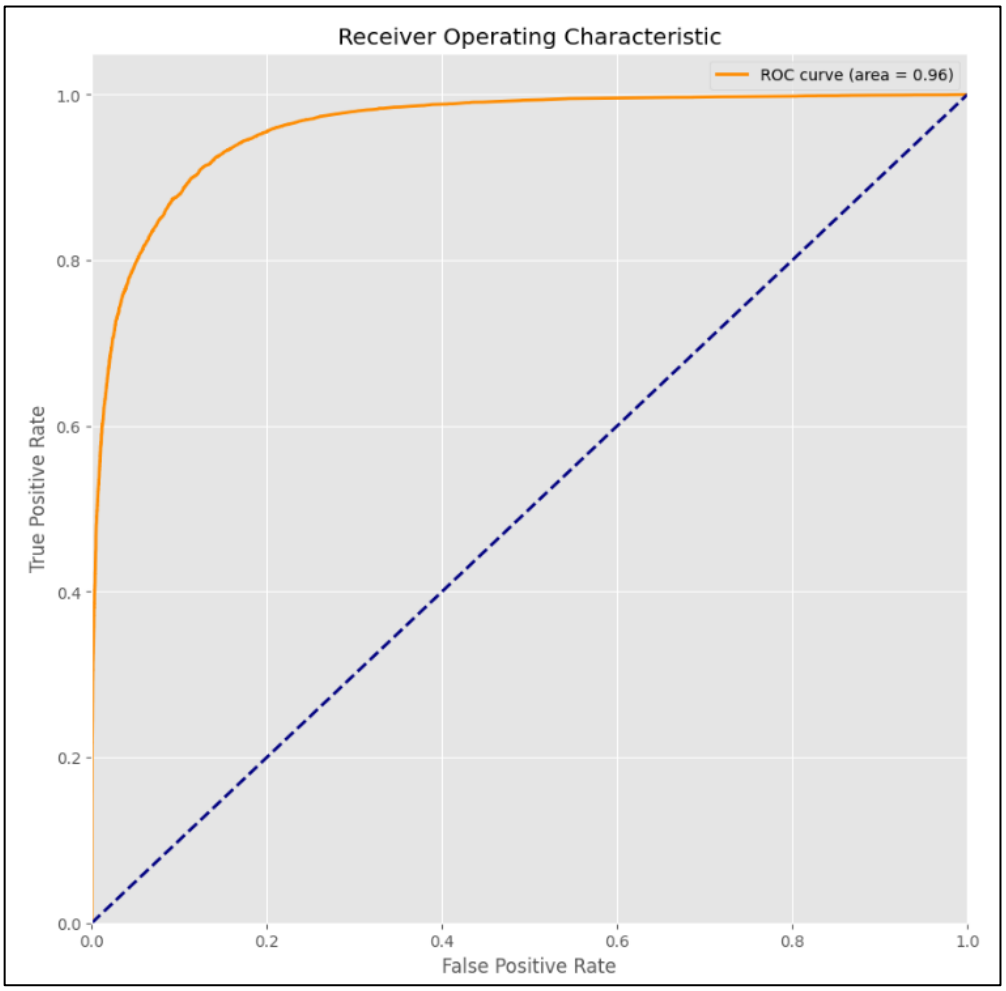


Рисунок 3.21 – Графік ROC кривої методу XGBoost

Аналіз важливості ознак для XGBoost наведено на рисунку 3.22. Кількість скісних рисок виявилася найважливішою ознакою з вагою 0.359. Це свідчить про те, що XGBoost найбільше покладається на цей структурний елемент URL при класифікації. Кількість знаків питання несподівано вийшла на друге місце з вагою 0.141, що вказує на здатність XGBoost ідентифікувати важливість менш очевидних ознак, які можуть бути індикаторами динамічно згенерованих фішингових посилань. Довжина URL має меншу вагу 0.101 порівняно з іншими ансамблевими моделями. Цей аналіз важливістю ознак показує, що XGBoost використовує інший, більш комплексний набір ознак, ніж

Random Forest, для прийняття рішення.

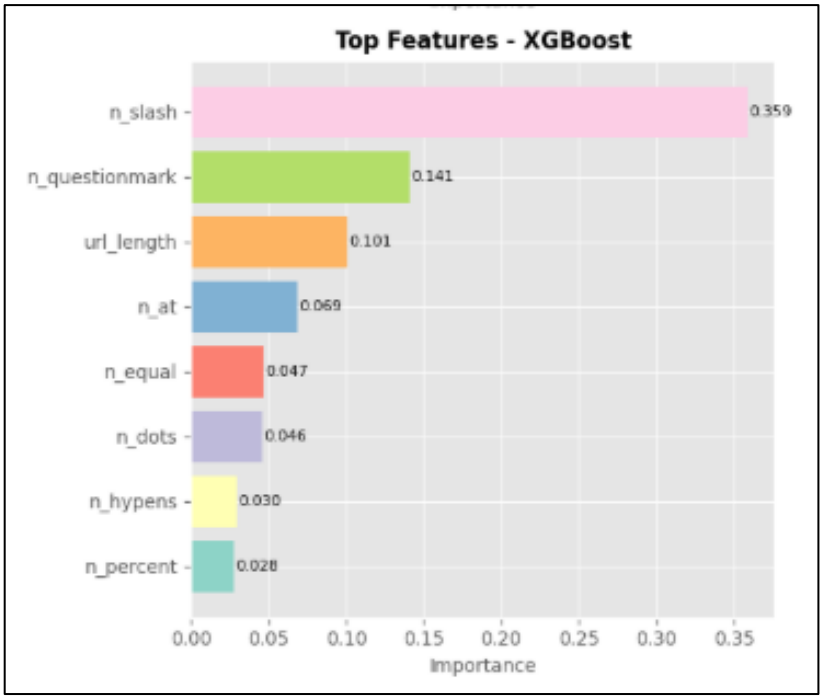


Рисунок 3.22 – Графік важливості ознак методу XGBoost

Модель XGBoost підтвердила свій статус високопродуктивного алгоритму, продемонструвавши найвищу точність та високий F1-Score. За показником AUC вона є однією з найкращих. Незважаючи на це, Random Forest залишається трохи кращим за критичним показником Recall, мінімізуючи пропуск фішингових сайтів. XGBoost є відмінним кандидатом, але його чутливість до n_slash та n_questionmark вказує на його фокус на певних структурних аномаліях.

LightGBM застосовує стратегію листового зростання. Це означає, що він обирає лист, що забезпечує найбільше зменшення втрат, а не росте рівномірно. Крім того, LightGBM використовує техніку ексклюзивного об'єднання ознак для зменшення розмірності та градієнтну вибірку для ефективної роботи з незбалансованими даними.

Переваги: Виняткова швидкість навчання та менше використання пам'яті порівняно з XGBoost та Gradient Boosting. Зберігає високу точність і ефективно справляється з великими масивами даних.

Недоліки: Через стратегію листового зростання більш схильний до перенавчання на малих наборах даних. Вимагає обережного налаштування гіперпараметрів.

На рисунку 3.23 показано лістинг коду для методу LightGBM.

```
models['LightGBM'] = LGBMClassifier(
    n_estimators=500,
    max_depth=9,
    learning_rate=0.1,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42,
    verbose=-1
)

start_time = time.time()
models['LightGBM'].fit(X_train, y_train)

predictions_train = models['LightGBM'].predict(X_train)
predictions_test = models['LightGBM'].predict(X_test)

print(f"Train Accuracy Score: {accuracy_score(y_train, predictions_train):.4f}")
print(f"Test Accuracy Score: {accuracy_score(y_test, predictions_test):.4f}")
```

Рисунок 3.23 – Лістинг коду для методу LightGBM

Результати оцінки продуктивності моделі LightGBM були отримані на незалежній тестовій вибірці та свідчать про її базову здатність до класифікації фішингових сайтів. Згідно з порівняльною таблицею, ключові показники демонструють наступну картину:

- Train Accuracy Score: 0.9065;
- Test Accuracy Score: 0.8965;
- Precision: 0.8652;
- Recall: 0.8514;
- F1-Score: 0.8582.

Матриця плутанини для моделі LightGBM показує у полі True Negative (Не-фішинг, правильно передбачено як Не-фішинг) зафіксовано 17508 випадків. У полі True Positive (Фішинг, правильно передбачено як Фішинг) зафіксовано 9408 випадків.Кількість False Negative, де фішинговий сайт був

пропущений, становить 1642. Цей показник є другим найкращим після Random Forest, підтверджуючи високий Recall. Кількість False Positive, де легітимний сайт був помилково заблокований, становить 1466. На рисунку 3.24 показано графік, який відображає результати класифікації моделі для тестових даних, наведені у вигляді матриці плутанини.

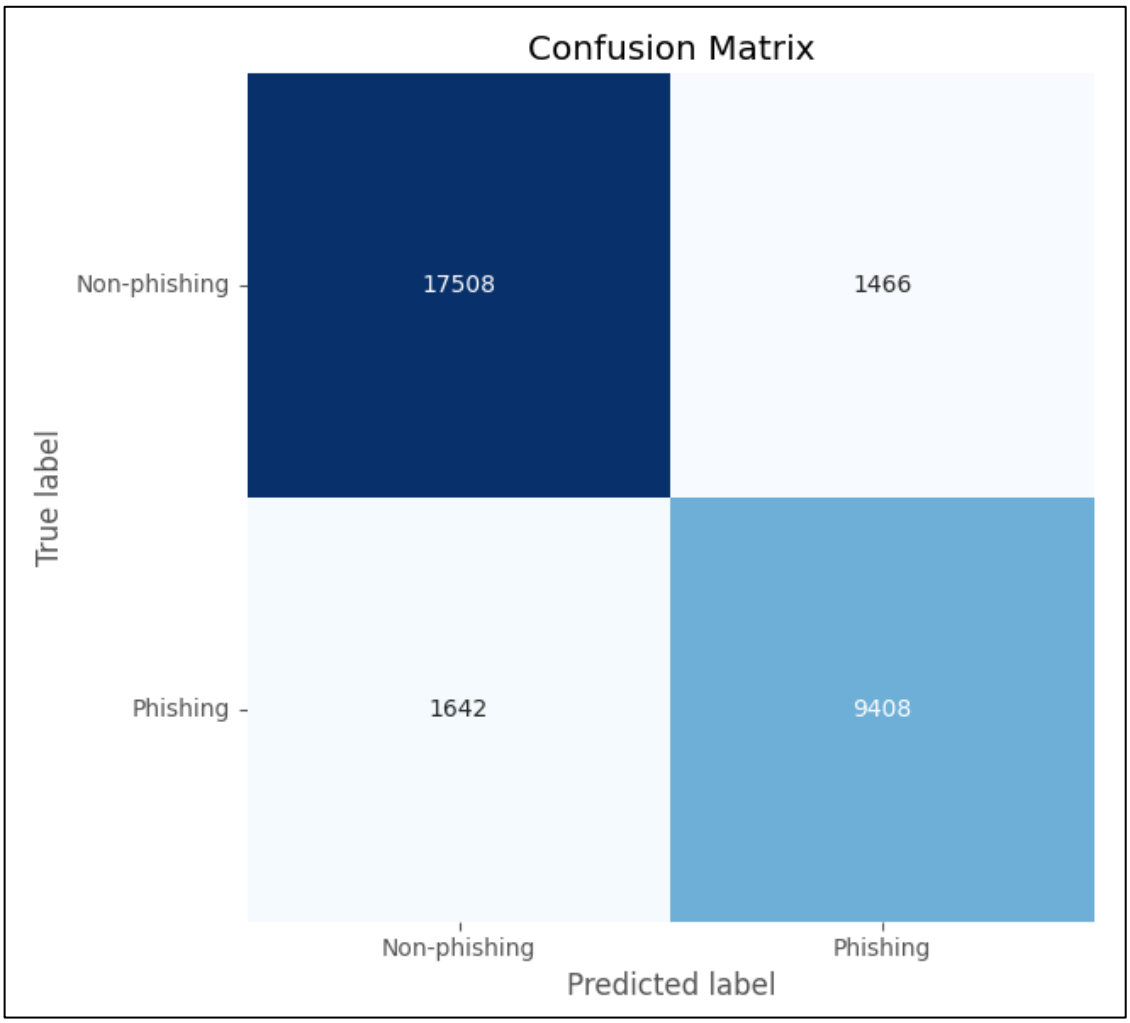


Рисунок 3.24 – Матриця плутанини методу LightGBM

Графік ROC-кривої для LightGBM демонструє відмінну розділювальну здатність. Площа під кривою AUC становить 0.96, що підтверджує, що LightGBM має найвищу загальну прогностичну якість серед усіх проаналізованих моделей, разом із Random Forest та XGBoost. На рисунку 3.25 представлено ROC-криву, яка є стандартним графіком для оцінки продуктивності бінарного класифікатора, оскільки вона візуалізує залежність

між чутливістю True Positive Rate та специфічністю False Positive Rate при варіації порогу класифікації.

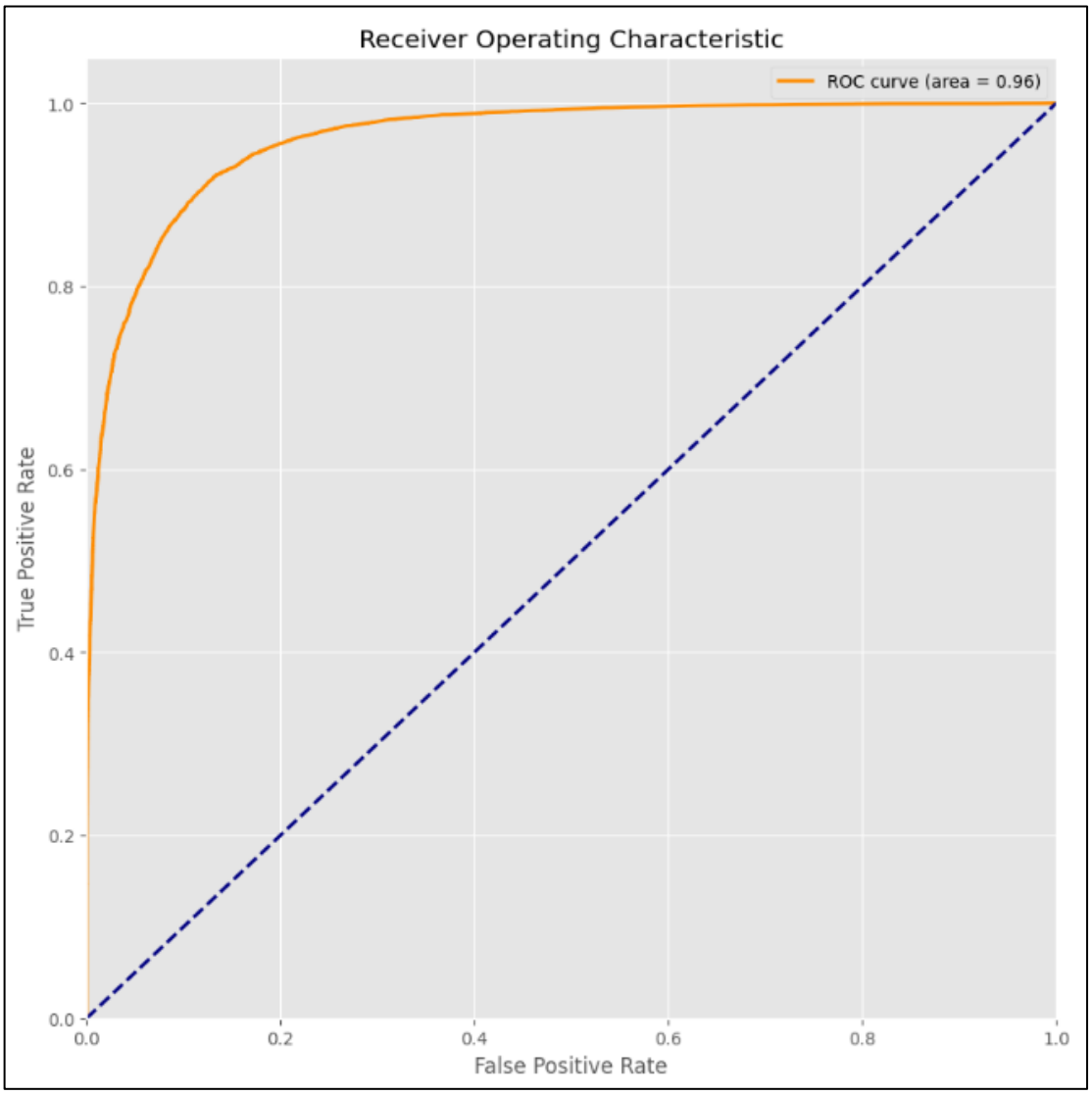


Рисунок 3.25 – Графік ROC кривої методу LightGBM

Аналіз важливості ознак для LightGBM наведено на рисунку 3.26. Важливість тут вимірюється кількістю разів, коли ознака була використана для розбиття у деревах. Довжина URL знову стає найважливішою ознакою з вагою 4637. Кількість дефісів піднялася на друге місце з вагою 1915. Кількість скісних рисок знаходиться на третьому місці з вагою 1860. На відміну від XGBoost, LightGBM більш тісно узгоджується з Random Forest та Gradient Boosting у тому, що найбільше цінує довжину URL. Він демонструє високу

чутливість до дефісів, які часто використовуються у фішингових посиланнях для імітації легітимних доменів.

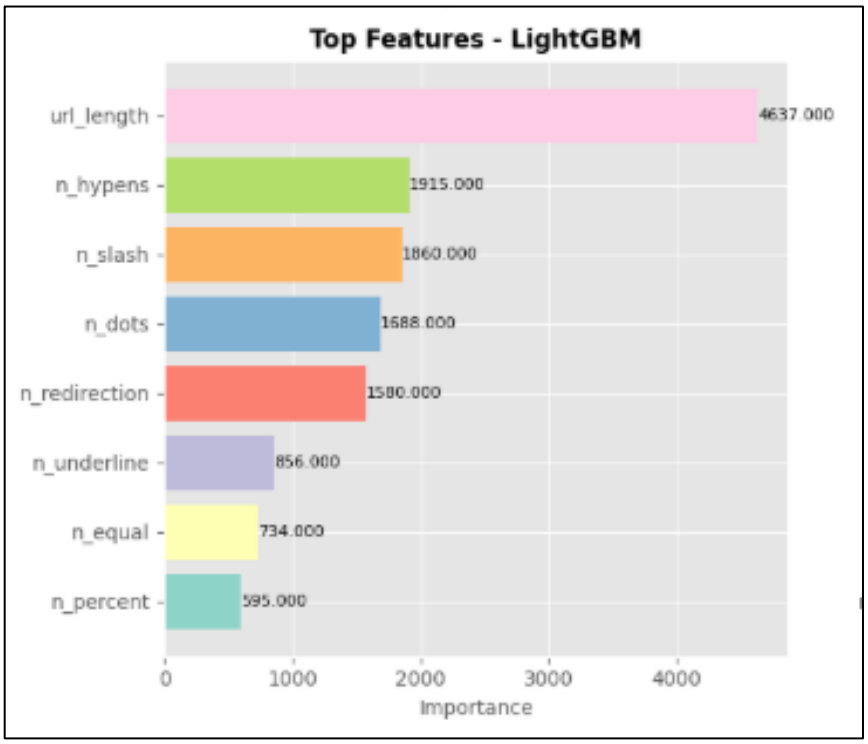


Рисунок 3.26 – Графік важливості ознак методу LightGBM

Модель ExtraTrees, як і Random Forest, будує ансамбль із великої кількості Decission Tree. Ключова відмінність полягає у процесі розщеплення. ExtraTrees використовує весь навчальний набір для кожного дерева. Для кожного вузла розщеплення обирається випадковий поріг для кожної ознаки, замість обчислення оптимального порогу.

Переваги: Висока швидкість навчання порівняно з Random Forest та Gradient Boosting завдяки додатковій рандомізації та спрощенню процесу вибору порогів. Дуже добре справляється зі зниженням дисперсії та боротьбою з перенавчанням.

Недоліки: Іноді може мати трохи нижчу точність у порівнянні з ретельно оптимізованими моделями бустингу. Додаткова випадковість може збільшити зміщення моделі.

На рисунку 3.27 показано лістинг коду для методу ExtraTrees.

```
models['ExtraTrees'] = ExtraTreesClassifier(
    n_estimators=1500,
    max_depth=25,
    random_state=42,
)

start_time = time.time()
models['ExtraTrees'].fit(X_train, y_train)

predictions_train = models['ExtraTrees'].predict(X_train)
predictions_test = models['ExtraTrees'].predict(X_test)

print(f"Train Accuracy Score: {accuracy_score(y_train, predictions_train):.4f}")
print(f"Test Accuracy Score: {accuracy_score(y_test, predictions_test):.4f}")
```

Рисунок 3.27 – Лістинг коду для методу ExtraTrees

Результати оцінки продуктивності моделі ExtraTrees були отримані на незалежній тестовій вибірці та свідчать про її базову здатність до класифікації фішингових сайтів. Згідно з порівняльною таблицею, ключові показники демонструють наступну картину:

- Train Accuracy Score: 0.8942;
- Test Accuracy Score: 0.8827;
- Precision: 0.8779;
- Recall: 0.7914;
- F1-Score: 0.8324.

Матриця плутанини для моделі ExtraTrees показує у полі True Negative (Не-фішинг, правильно передбачено як Не-фішинг) зафіксовано 17758 випадків. У полі True Positive (Фішинг, правильно передбачено як Фішинг) зафіксовано 8745 випадків. Кількість False Negative, де фішинговий сайт був пропущений, становить 2305. Це є найгіршим показником серед усіх ансамблевих моделей, що підтверджує низький Recall. Кількість False Positive, де легітимний сайт був помилково заблокований, становить 1216. Це є найнижчим показником серед усіх моделей, що відповідає високому Precision. На рисунку 3.28 показано графік, який відображає результати класифікації моделі для тестових даних, наведені у вигляді матриці плутанини.

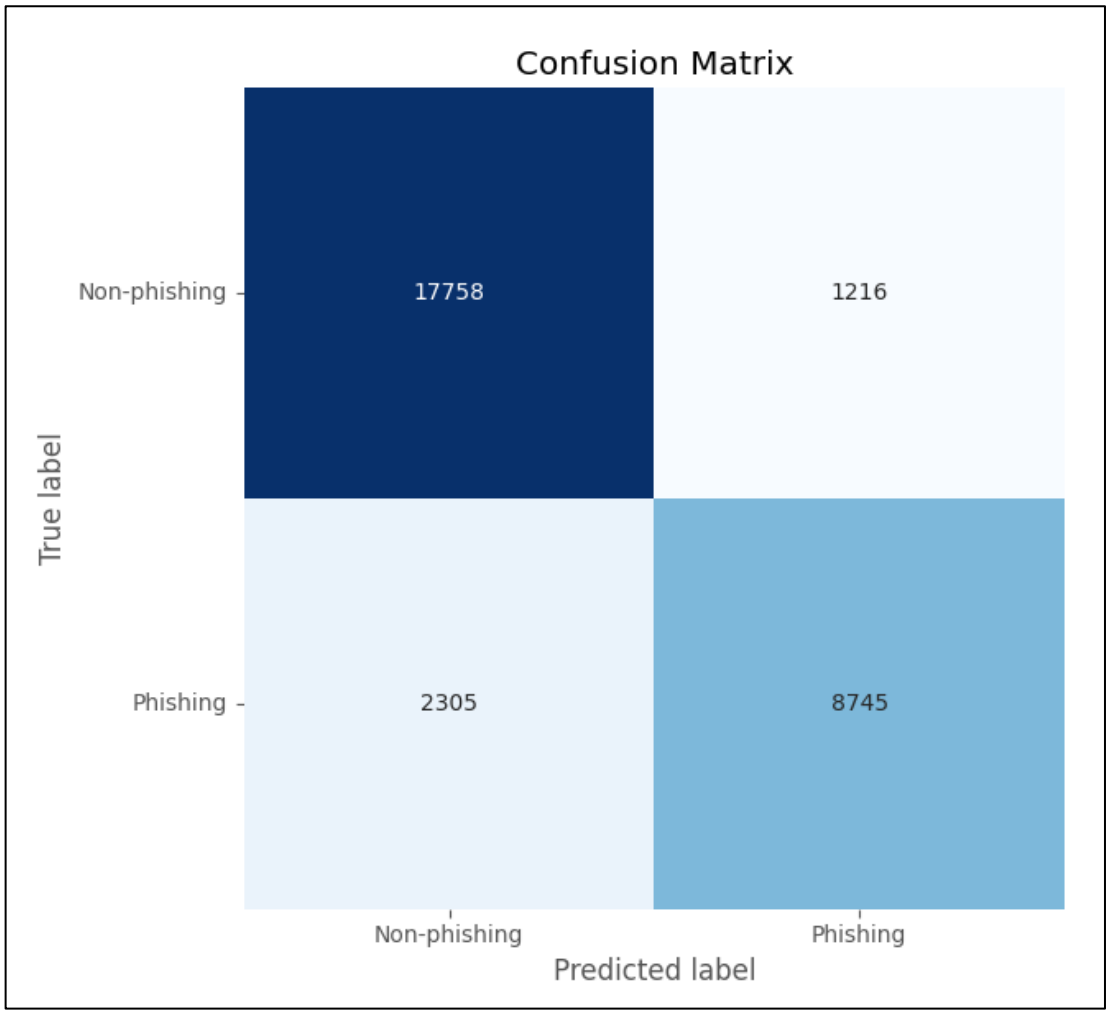


Рисунок 3.28 – Матриця плутанини методу ExtraTrees

Графік ROC-кривої для ExtraTrees демонструє високу розділювальну здатність. Площа під кривою AUC становить 0.95, що трохи нижче, ніж у найкращих ансамблевих моделей, але все ще є дуже високим показником. На рисунку 3.29 представлено ROC-криву, яка є стандартним графіком для оцінки продуктивності бінарного класифікатора, оскільки вона візуалізує залежність між чутливістю True Positive Rate та специфічністю False Positive Rate при варіації порогу класифікації.

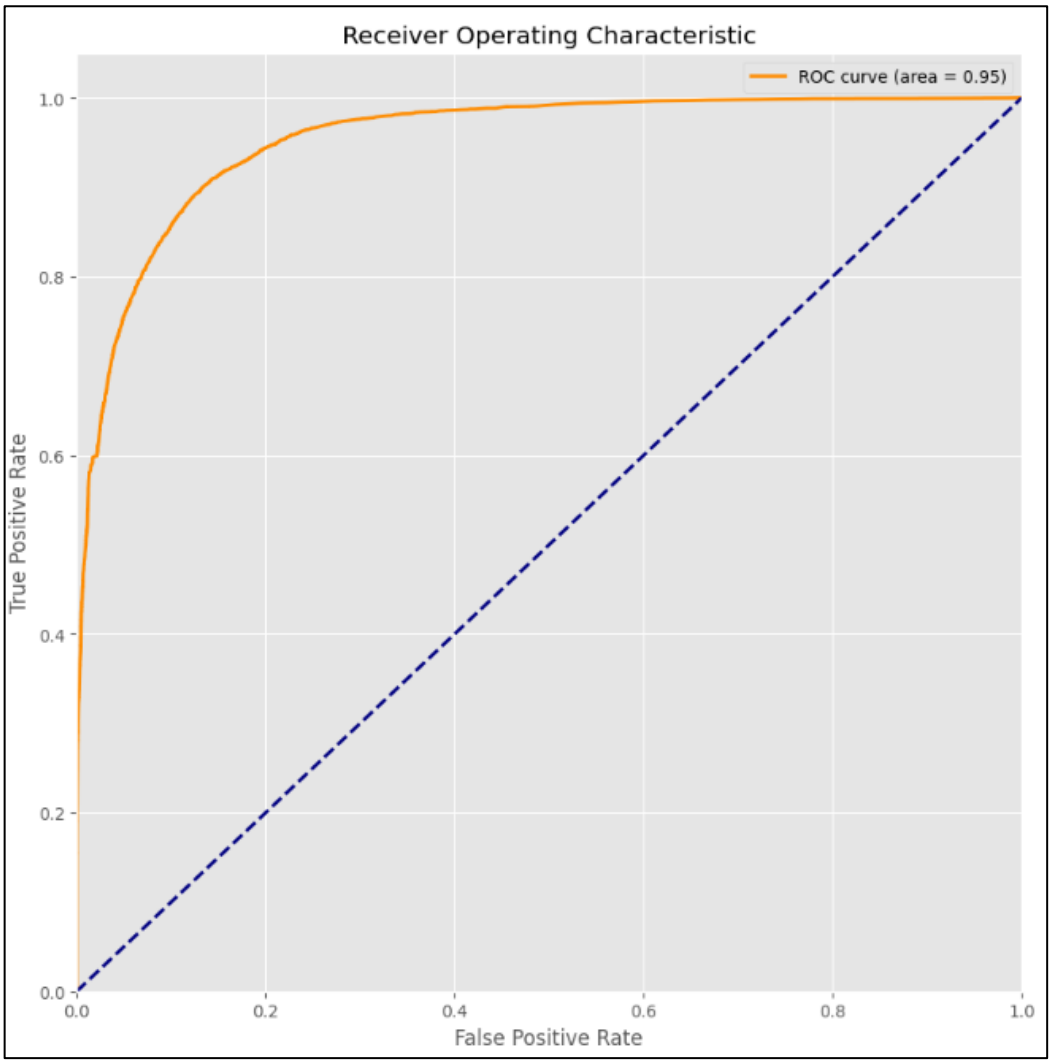


Рисунок 3.29 – Графік ROC кривої методу ExtraTrees

Аналіз важливості ознак для ExtraTrees наведено на рисунку 3.30 .Кількість скісних рисок виявилася найважливішою ознакою з винятково високою вагою 0.527. Це найвищий показник важливості, присвоєний одній ознаці серед усіх ансамблевих методів. Довжина URL значно поступається з вагою 0.152. Цей аналіз показує, що ExtraTrees надмірно залежить від однієї ознаки n_slash, що може бути причиною його відносно низького Recall, оскільки модель може ігнорувати інші важливі, але менш очевидні індикатори.

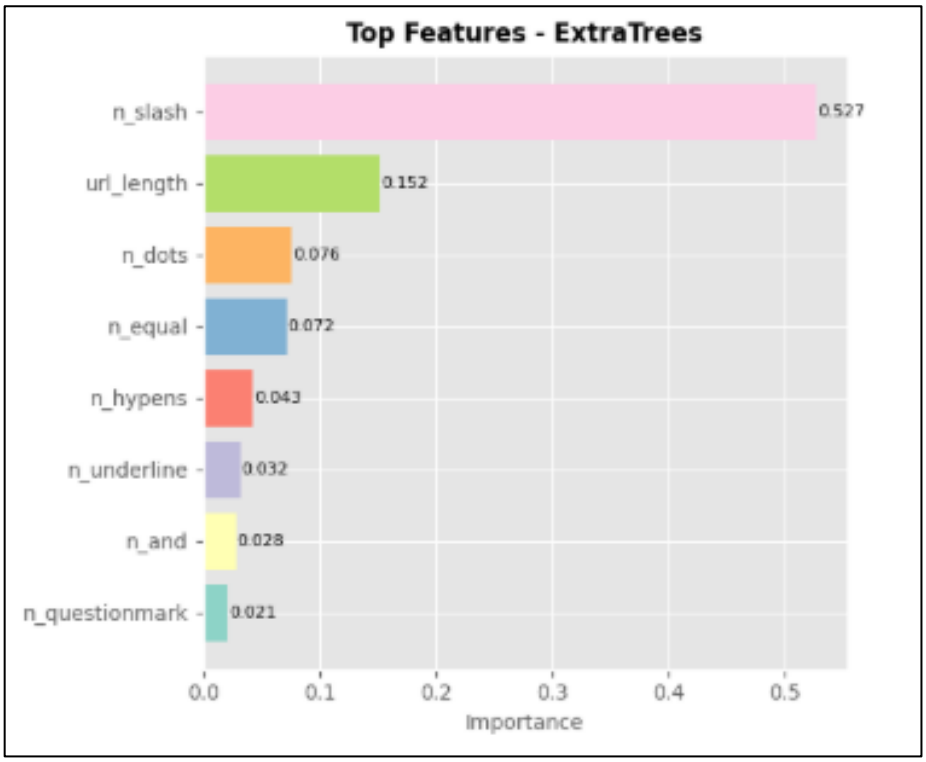


Рисунок 3.30 – Графік важливості ознак методу ExtraTrees

Модель ExtraTrees демонструє виняткову здатність до уникнення помилкових спрацювань, найвищий Precision, але її низький Recall є неприйнятним для систем, де пріоритетом є мінімізація пропуску реальних загроз. Вона є менш якісною для даної задачі, ніж Random Forest, XGBoost та LightGBM, через свій дисбаланс метрик та високу залежність від однієї ознаки.

K-Nearest Neighbors Model (KNN) - це класифікатор, що базується на відстані. Для класифікації нового об'єкта він шукає K найближчих до нього точок у навчальному наборі. Клас нового об'єкта визначається голосуванням більшості серед цих K сусідів.

Переваги: Надзвичайно простий в реалізації та розумінні. Не вимагає етапу навчання як такого. Може працювати з будь-яким розподілом даних.

Недоліки: Чутливий до масштабу ознак. Дуже повільний на великих наборах даних під час передбачення, оскільки вимагає обчислення відстані до кожної точки.

На рисунку 3.31 показано лістинг коду для методу KNN.

```
models['KNN'] = KNeighborsClassifier(
    n_neighbors=14,
    weights='distance',
    algorithm='auto',
    leaf_size=20,
    p=2,
    metric='minkowski'
)

start_time = time.time()
models['KNN'].fit(X_train, y_train)
knn_time = time.time() - start_time

predictions_train = models['KNN'].predict(X_train)
predictions_test = models['KNN'].predict(X_test)

print(f"Train Accuracy Score: {accuracy_score(y_train, predictions_train):.4f}")
print(f"Test Accuracy Score: {accuracy_score(y_test, predictions_test):.4f}")
```

Рисунок 3.31 – Лістинг коду для методу KNN

Результати оцінки продуктивності моделі KNN були отримані на незалежній тестовій вибірці та свідчать про її базову здатність до класифікації фішингових сайтів. Згідно з порівняльною таблицею, ключові показники демонструють наступну картину:

- Train Accuracy Score: 0.9223;
- Test Accuracy Score: 0.8849;
- Precision: 0.8672;
- Recall: 0.114;
- F1-Score: 0.8384.

Матриця плутанини для моделі KNN показує у полі True Negative (Не-фішинг, правильно передбачено як Не-фішинг) зафіксовано 17601 випадків. У полі True Positive (Фішинг, правильно передбачено як Фішинг) зафіксовано 8966 випадків. Кількість False Negative, де фішинговий сайт був пропущений, становить 2084. Це високий показник, гірше, ніж у всіх ансамблевих методів, що підтверджує низьку здатність моделі до виявлення всіх реальних загроз. Кількість False Positive, де легітимний сайт був помилково заблокований, становить 1373. На рисунку 3.32 показано графік, який відображає результати класифікації моделі для тестових даних, наведені у вигляді матриці плутанини.

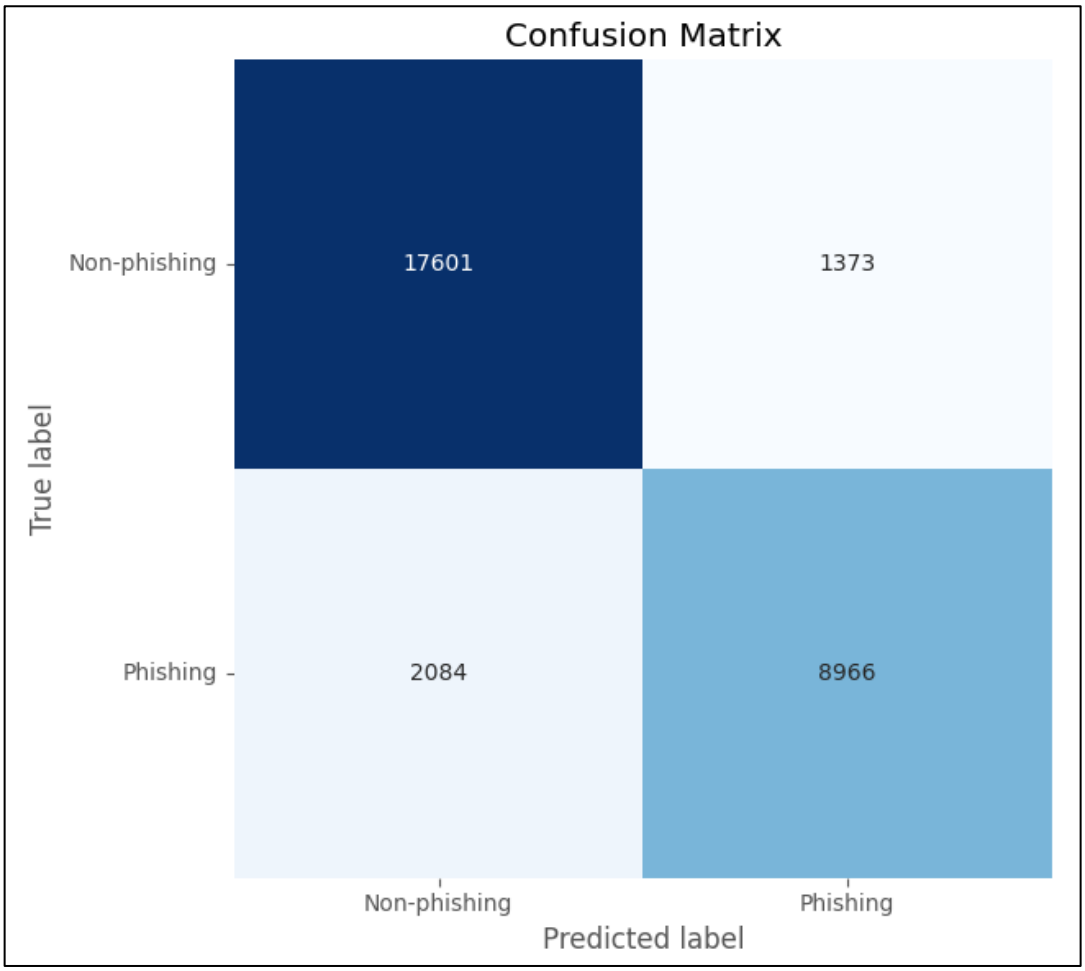


Рисунок 3.32 – Матриця плутанини методу KNN

Графік ROC-кривої для KNN демонструє високу розділювальну здатність, Площа під кривою AUC становить 0.94, що є нижчим показником, ніж у більшості ансамблевих методів. Це свідчить про те, що, незважаючи на загальну ефективність, вона гірше розрізняє класи порівняно з провідними моделями. На рисунку 3.33 представлено ROC-криву, яка є стандартним графіком для оцінки продуктивності бінарного класифікатора, оскільки вона візуалізує залежність між чутливістю True Positive Rate та специфічністю False Positive Rate при варіації порогу класифікації.

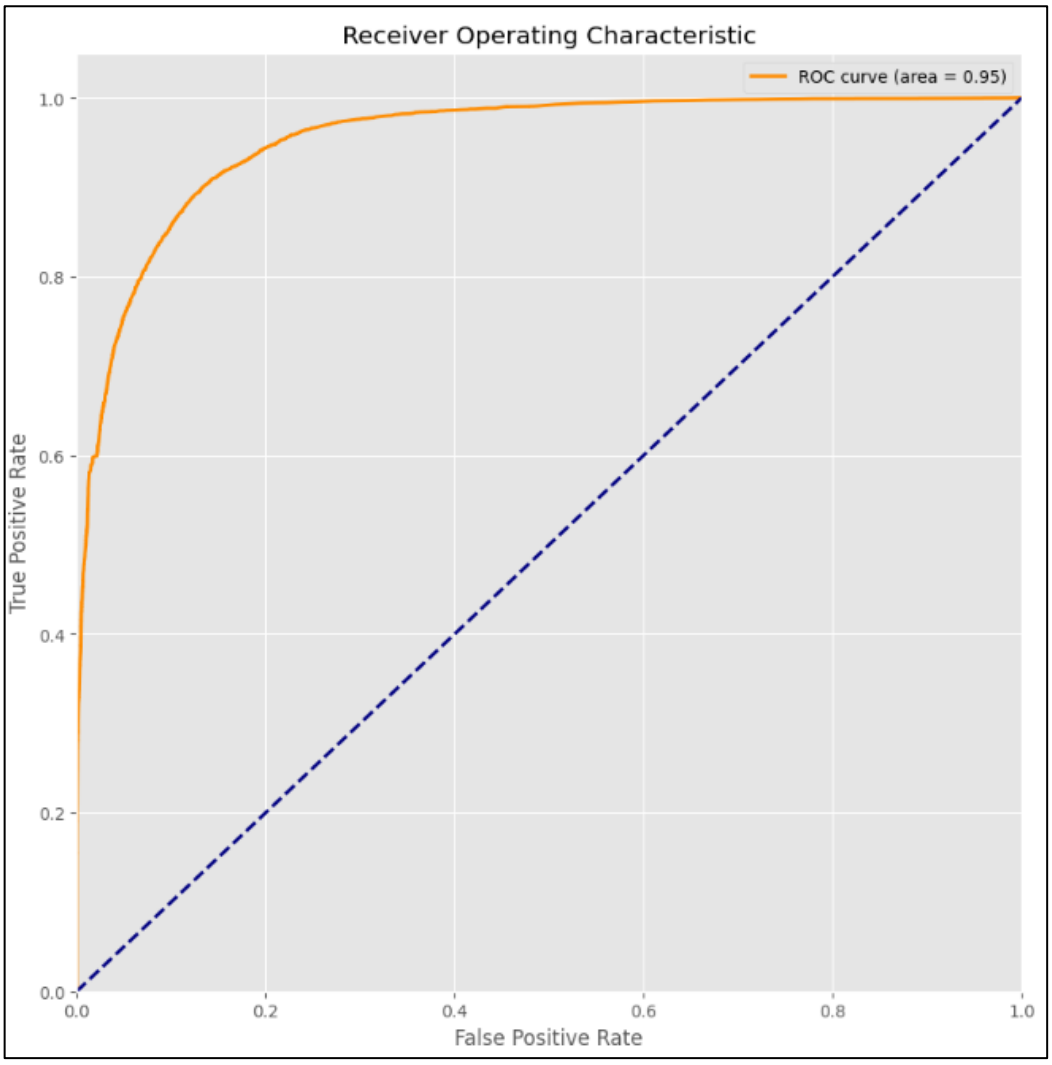


Рисунок 3.33 – Графік ROC кривої методу KNN

Модель K-Nearest Neighbors Model (KNN) є найменш придатною для цієї задачі серед усіх ансамблевих та простих моделей на основі дерев, оскільки вона демонструє найбільше перенавчання. Має низький Recall, що призводить до високої кількості пропущених загроз. Має найменший F1-Score серед ансамблевих моделей, що вказує на неоптимальний баланс метрик.

Для порівняльного аналізу та оцінки ефективності глибокого навчання, були розроблені та протестовані кілька конфігурацій нейронних мереж.

Перша модель нейронної мережі є багатошаровим перцептроном (MLP). Вона складається з вхідного шару, двох прихованих шарів 64 та 32 нейрони з функцією активації ReLU, та вихідного шару з сигмоїдною активацією. Високе значення Dropout 0.5 було обрано для того, щоб змусити мережу навчитися

більш надійним ознакам та уникнути надмірної залежності від будь-якого конкретного нейрона.

Переваги: Простота архітектури та відносно швидке навчання. ReLU допомагає боротися з проблемою зникаючого градієнта.

Недоліки: Високий Dropout може призвести до недооцінки загальної потужності моделі, якщо це не потрібно.

На рисунку 3.34 показано лістинг коду для першої моделі нейронної мережі.

```
# Neural Network 1: Shallow network with ReLU and high dropout
def create_nn1(input_shape):
    model = Sequential([
        Input(shape=(input_shape,)),
        Dense(64, activation='relu'),
        Dropout(0.5),
        Dense(32, activation='relu'),
        Dropout(0.5),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    return model
```

Рисунок 3.34 – Лістинг коду для першої моделі нейронної мережі

Результати оцінки продуктивності першої моделі нейронної мережі були отримані на незалежній тестовій вибірці та свідчать про її базову здатність до класифікації фішингових сайтів. Згідно з порівняльною таблицею, ключові показники демонструють наступну картину:

- Accuracy: 0.8834;
- Val_Accuracy: 0.8836.

Матриця плутанини для моделі Neural Network 1 показує у полі True Negative (Не-фішинг, правильно передбачено як Не-фішинг) зафіксовано 17441 випадок. У полі True Positive (Фішинг, правильно передбачено як Фішинг) зафіксовано 9089 випадків. Кількість False Negative, де фішинговий сайт був пропущений, становить 1961. Це є високим показником порівняно з ансамблевими моделями. Кількість False Positive, де легітимний сайт був

помилково заблокований, становить 1533. На рисунку 3.35 показано графік, який відображає результати класифікації першої моделі нейронної мережі для тестових даних, наведені у вигляді матриці плутанини.

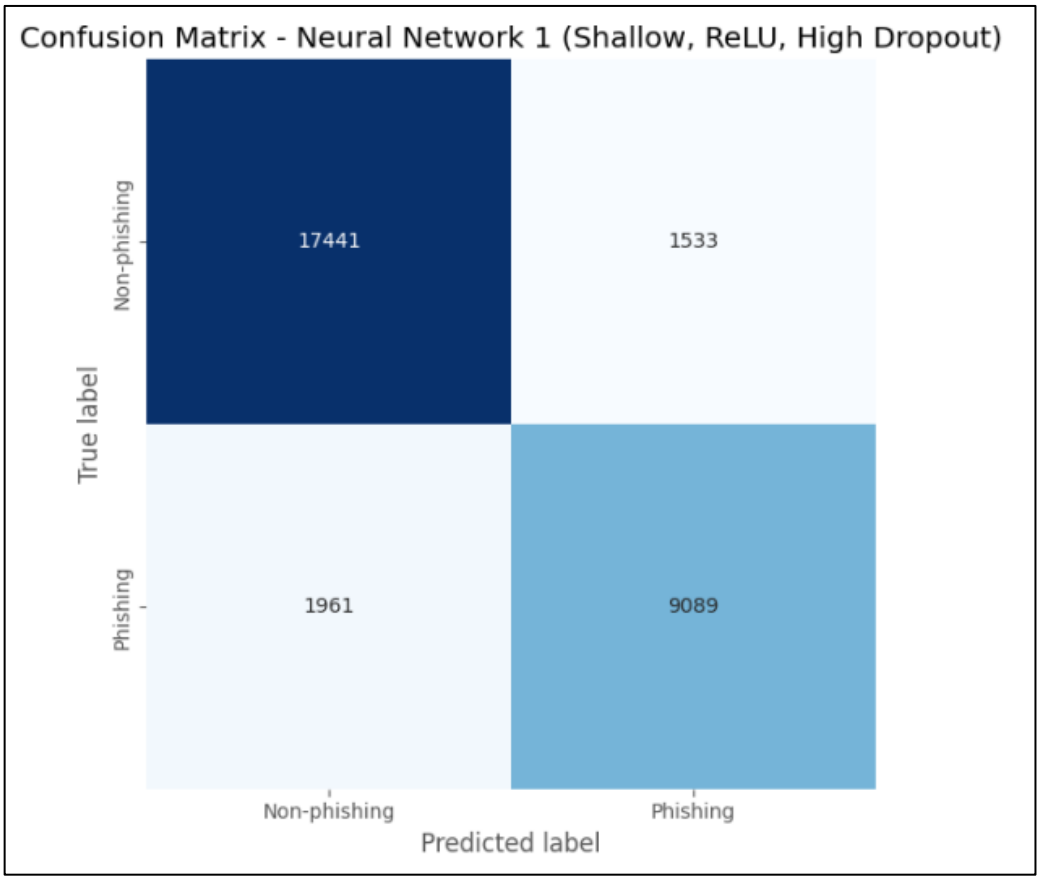


Рисунок 3.35 – Матриця плутанини першої моделі нейронної мережі

Графік ROC-кривої для першої моделі нейронної мережі демонструє площу під кривою AUC 0.95. Це відмінний результат, хоча він трохи нижчий, ніж у найкращих ансамблевих методів. На рисунку 3.36 представлено ROC-криву, яка є стандартним графіком для оцінки продуктивності бінарного класифікатора, оскільки вона візуалізує залежність між чутливістю True Positive Rate та специфічністю False Positive Rate при варіації порогу класифікації.

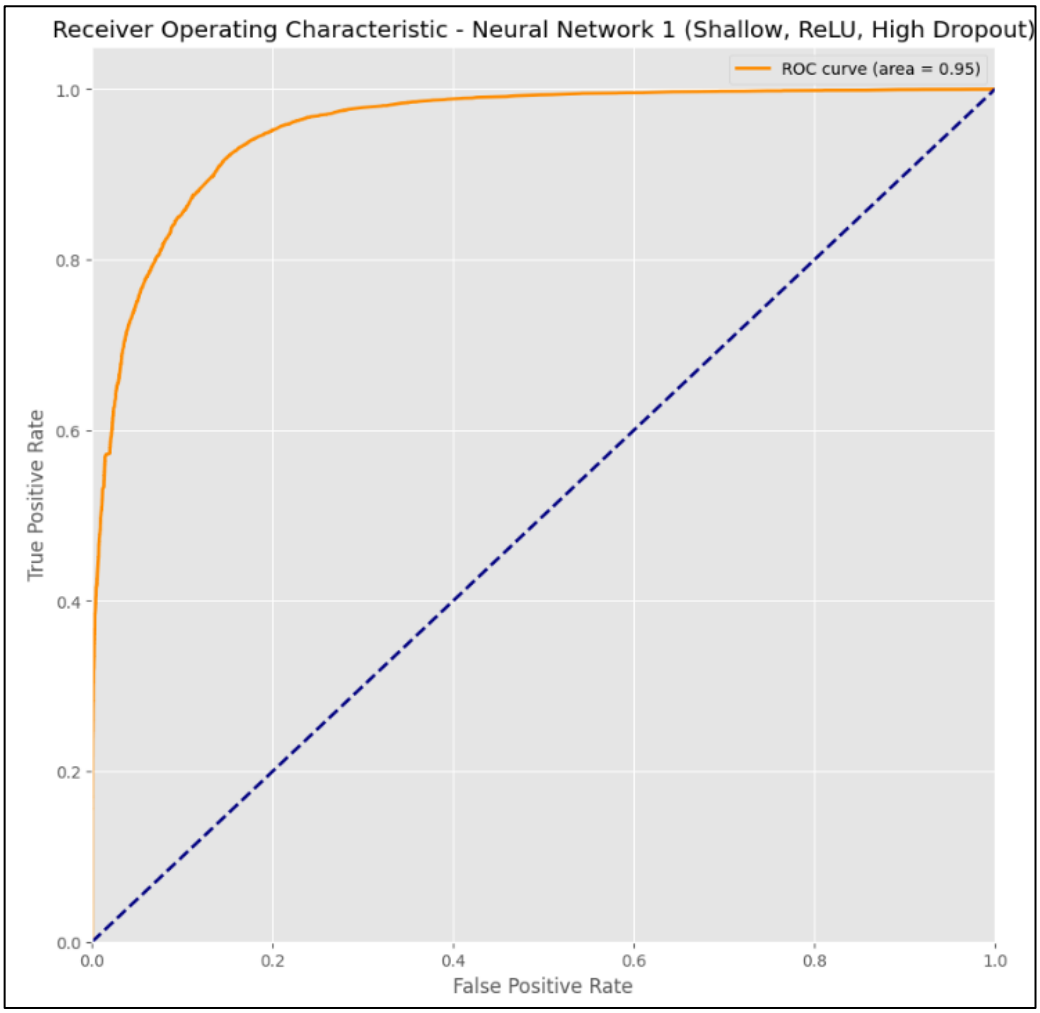


Рисунок 3.36 – Графік ROC кривої першої моделі нейронної мережі

Нейронна мережа 1 є стабільною моделлю завдяки високому Dropout, але її продуктивність поступається найкращим ансамблевим методам LightGBM, Random Forest, XGBoost за загальною точністю та здатністю мінімізувати пропуски загроз. Це вказує на те, що структурні ознаки URL можуть бути ефективніше класифіковані наборами дерев.

Друга модель нейронної мережі є глибшим багатошаровим перцептроном (MLP). Вона має три приховані шари 128, 64 та 32 нейрони. Використання функції активації Tanh може бути ефективним для глибших мереж, дозволяючи швидше сходитися. Рівень Dropout знижено до 0.3, щоб дозволити моделі використовувати більшу частину навчених ваг.

Переваги: Більша кількість шарів може дозволити моделі вивчити більш складні та ієрархічні ознаки. Tanh забезпечує сильніший градієнт порівняно з

сигмоїдою.

Недоліки: Чутливість до ініціалізації ваг. Нижча точність на тестовій вибірці порівняно з NN1 свідчить про те, що дана конфігурація може бути менш ефективною для цього набору ознак.

На рисунку 3.37 показано лістинг коду для другої моделі нейронної мережі.

```
# Neural Network 2: Deeper network with tanh and moderate dropout
def create_nn2(input_shape):
    model = Sequential([
        Input(shape=(input_shape,)),
        Dense(128, activation='tanh'),
        Dropout(0.3),
        Dense(64, activation='tanh'),
        Dropout(0.3),
        Dense(32, activation='tanh'),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    return model
```

Рисунок 3.37 – Лістинг коду для другої моделі нейронної мережі

Результати оцінки продуктивності другої моделі нейронної мережі були отримані на незалежній тестовій вибірці та свідчать про її базову здатність до класифікації фішингових сайтів. Згідно з порівняльною таблицею, ключові показники демонструють наступну картину:

- Accuracy: 0.8737;
- Val_Accuracy: 0.8746.

Матриця плутанини для моделі Neural Network показує у полі True Negative (Не-фішинг, правильно передбачено як Не-фішинг) зафіксовано 17193 випадки. У полі True Positive (Фішинг, правильно передбачено як Фішинг) зафіксовано 9065 випадків. Кількість False Negative, де фішинговий сайт був пропущений, становить 1985. Цей показник є гіршим, ніж у NN1. Кількість False Positive, де легітимний сайт був помилково заблокований, становить 1781. Це найвищий показник серед усіх нейронних мереж. На

рисунку 3.38 показано графік, який відображає результати класифікації другої моделі нейронної мережі для тестових даних, наведені у вигляді матриці плутанини.

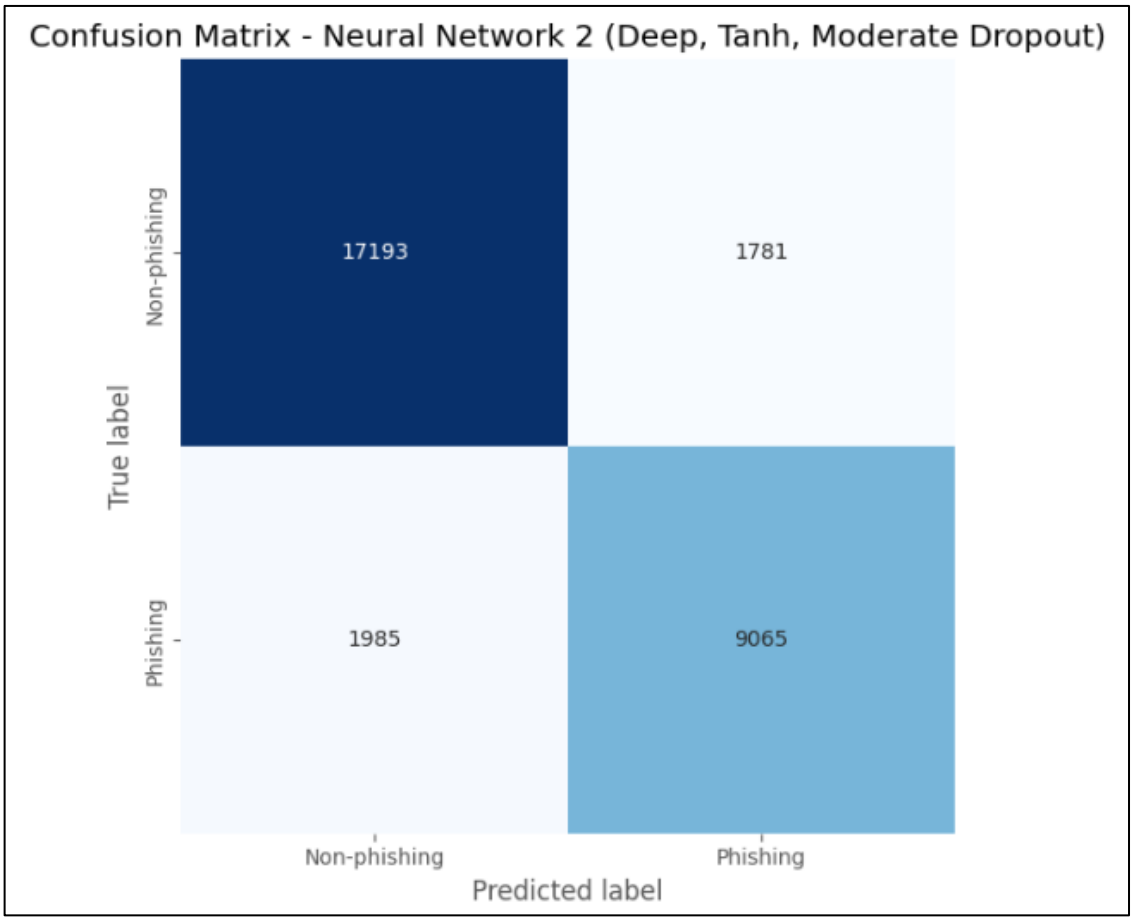


Рисунок 3.38 – Матриця плутанини другої моделі нейронної мережі

Графік ROC-кривої для Neural Network 2 демонструє площу під кривою AUC 0.95, що відповідає результату NN1 та є відмінним показником. На рисунку 3.39 представлено ROC-криву, яка є стандартним графіком для оцінки продуктивності бінарного класифікатора, оскільки вона візуалізує залежність між чутливістю True Positive Rate та специфічністю False Positive Rate при варіації порогу класифікації.

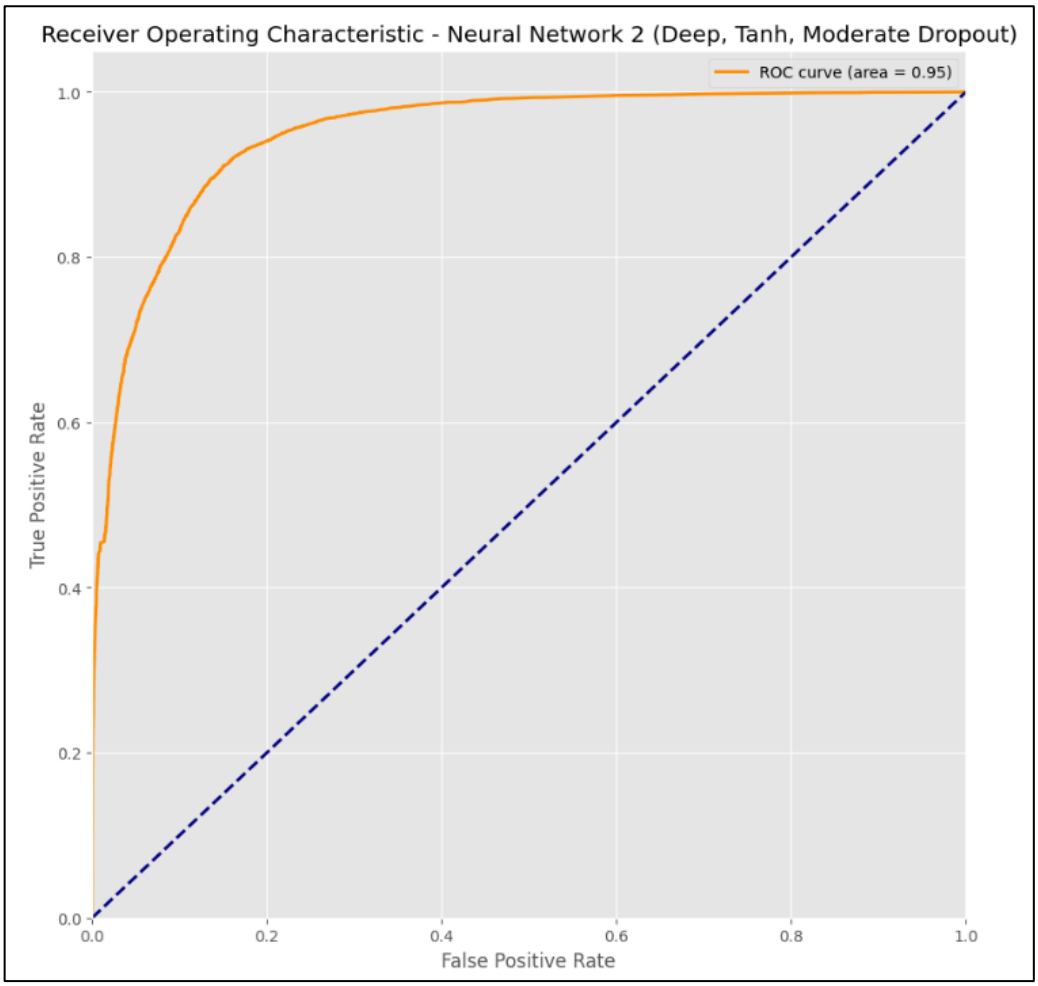


Рисунок 3.39 – Графік ROC кривої другої моделі нейронної мережі

Нейронна мережа 2 є найгіршою моделлю серед усіх нейронних мереж, оскільки має найнижчу точність і найвищий рівень хибнопозитивних спрацювань. Хоча її AUC високий, її продуктивність у реальних умовах є недостатньою порівняно з іншими моделями.

Третя модель нейронної мережі це багатошаровий перцептрон (MLP) з трьома прихованими шарами 64, 32 та 16 нейронів. Ключова особливість - використання змішаних функцій активації: ReLU у першому шарі та ELU (Exponential Linear Unit) у другому. ELU може забезпечити швидше навчання, ніж ReLU. Рівень Dropout знижено до 0.2 у перших двох шарах, щоб дозволити більшій кількості нейронів брати участь у навчанні.

Переваги: Використання ELU може забезпечити більш плавне навчання. Менший Dropout дозволяє мережі вивчати більш складні залежності.

Недоліки: Низький Dropout може підвищити ризик перенавчання, а змішування активацій може ускладнити інтерпретацію.

На рисунку 3.40 показано лістинг коду для третьої моделі нейронної мережі.

```
# Neural Network 3: Medium network with mixed activations and low dropout
def create_nn3(input_shape):
    model = Sequential([
        Input(shape=(input_shape,)),
        Dense(64, activation='relu'),
        Dropout(0.2),
        Dense(32, activation='elu'),
        Dropout(0.2),
        Dense(16, activation='relu'),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    return model
```

Рисунок 3.40 – Лістинг коду для третьої моделі нейронної мережі

Результати оцінки продуктивності третьої моделі нейронної мережі були отримані на незалежній тестовій вибірці та свідчать про її базову здатність до класифікації фішингових сайтів. Згідно з порівняльною таблицею, ключові показники демонструють наступну картину:

- Accuracy: 0.8719;
- Val_Accuracy: 0.8728.

Матриця плутанини для моделі Neural Network 3 показує у полі True Negative (Не-фішинг, правильно передбачено як Не-фішинг) зафіксовано 17360 випадків. У полі True Positive (Фішинг, правильно передбачено як Фішинг) зафіксовано 8846 випадків. Кількість False Negative, де фішинговий сайт був пропущений, становить 2204. Це є другим найгіршим показником серед усіх моделей, включаючи ExtraTrees. Це свідчить про низьку здатність цієї моделі виявляти реальні загрози. Кількість False Positive, де легітимний сайт був помилково заблокований, становить 1614. На рисунку 3.41 показано графік, який відображає результати класифікації третьої моделі нейронної мережі для тестових даних, наведені у вигляді матриці плутанини.

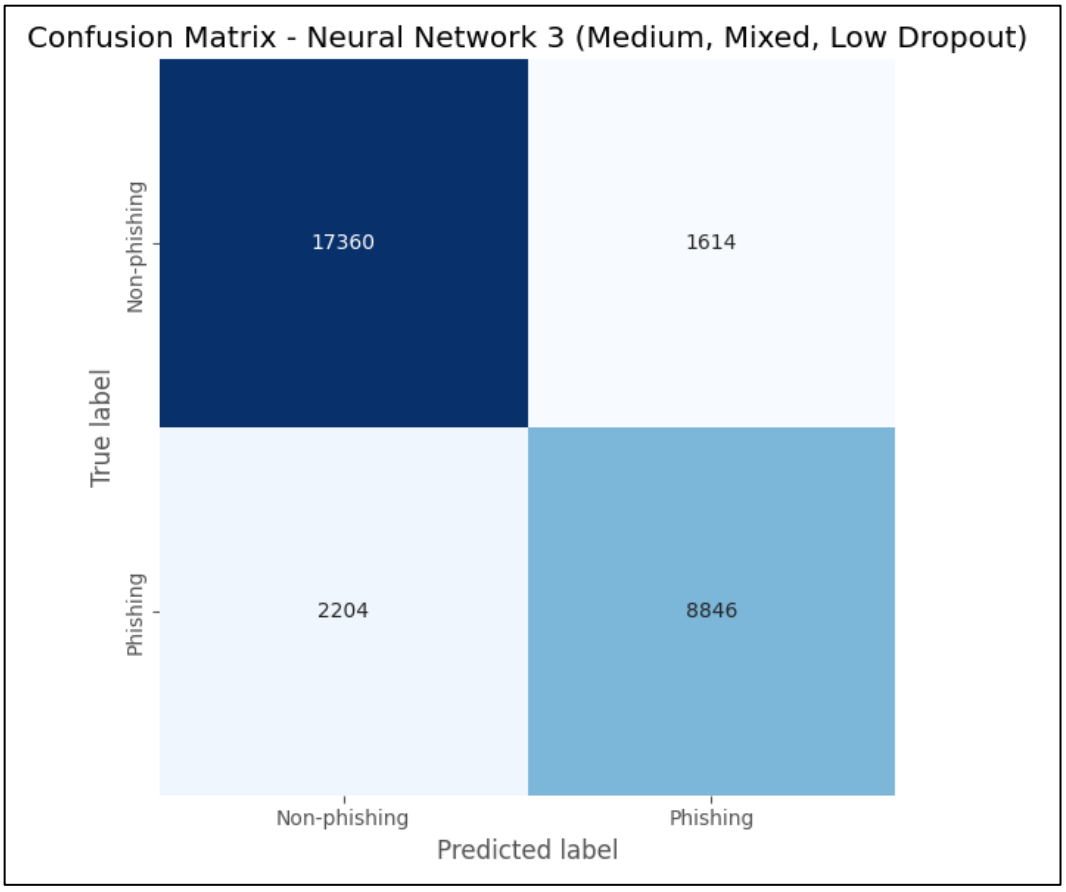


Рисунок 3.41 – Матриця плутанини третьої моделі нейронної мережі

Графік ROC-кривої для Neural Network 3 демонструє площу під кривою AUC 0.95, що відповідає результатам NN1 та NN2. Високий AUC при низькій точності вказує на те, що модель, незважаючи на погану узагальнюючу здатність, може добре ранжувати приклади за їхньою ймовірністю приналежності до класу. На рисунку 3.42 представлено ROC-криву, яка є стандартним графіком для оцінки продуктивності бінарного класифікатора, оскільки вона візуалізує залежність між чутливістю True Positive Rate та специфічністю False Positive Rate при варіації порогу класифікації.

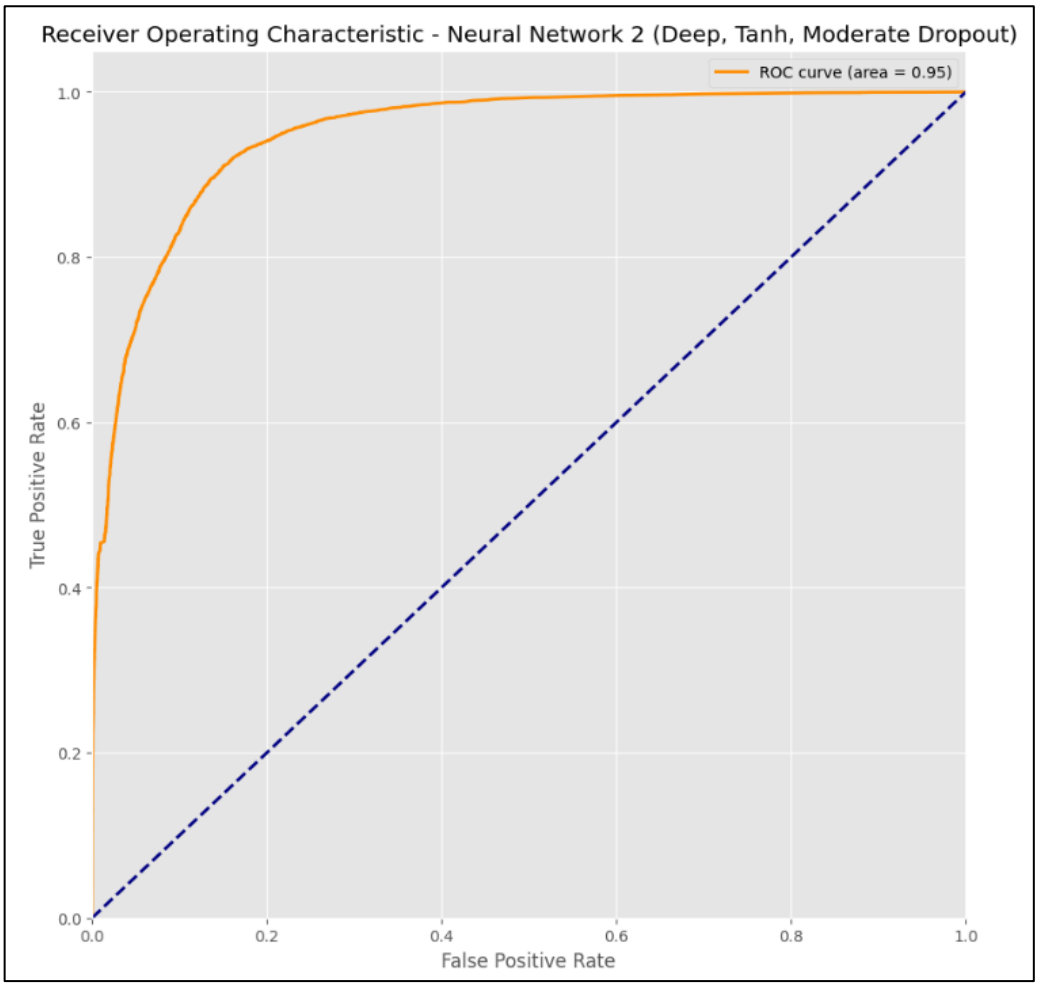


Рисунок 3.42 – Графік ROC кривої третьої моделі нейронної мережі

Нейронна мережа 3 є найменш точною моделлю серед усіх протестованих. Її висока кількість пропущених загроз роблять її непридатною для фінального впровадження.

Модель Scikit-learn MLP використовує три приховані шари 64, 32 та 16 нейронів з функцією активації ReLU. Scikit-learn MLP оптимізований для роботи з Scikit-learn пайплайнами.

Переваги: Висока інтеграція з екосистемою Scikit-learn. Простий в налаштуванні.

Недоліки: Не дозволяє використовувати Dropout, що може підвищити ризик перенавчання.

На рисунку 3.43 показано лістинг коду для MLPClassifier.

```
mlp = MLPClassifier(hidden_layer_sizes=(64, 32, 16), activation='relu', solver='adam',
                    max_iter=100, batch_size=128, random_state=42, learning_rate='adaptive')
mlp.fit(X_train_scaled, y_train)
print("\n=== Scikit-learn MLP ===")
train_acc, test_acc = calculate_metrics(mlp, X_train_scaled, X_test_scaled, y_train, y_test,
                                        model_name="Scikit-learn MLP")
nn_results['Scikit-learn MLP'] = {'train_accuracy': train_acc, 'test_accuracy': test_acc}
```

Рисунок 3.43 – Лістинг коду для Scikit-learn MLP

Результати оцінки продуктивності MLPClassifier були отримані на незалежній тестовій вибірці та свідчать про її базову здатність до класифікації фішингових сайтів. Згідно з порівняльною таблицею, ключові показники демонструють наступну картину:

- Accuracy: 0.9002;
- Val_Accuracy: 0.8935.

Матриця плутанини для моделі Scikit-learn MLP показує у полі True Negative (Не-фішинг, правильно передбачено як Не-фішинг) зафіксовано 17586 випадків. У полі True Positive (Фішинг, правильно передбачено як Фішинг) зафіксовано 9240 випадків. Кількість False Negative, де фішинговий сайт був пропущений, становить 1810. Цей показник є найкращим серед усіх нейронних мереж, але все ще гіршим, ніж у провідних ансамблевих методів. Кількість False Positive, де легітимний сайт був помилково заблокований, становить 1388. На рисунку 3.44 показано графік, який відображає результати класифікації Scikit-learn MLP для тестових даних, наведені у вигляді матриці плутанини.

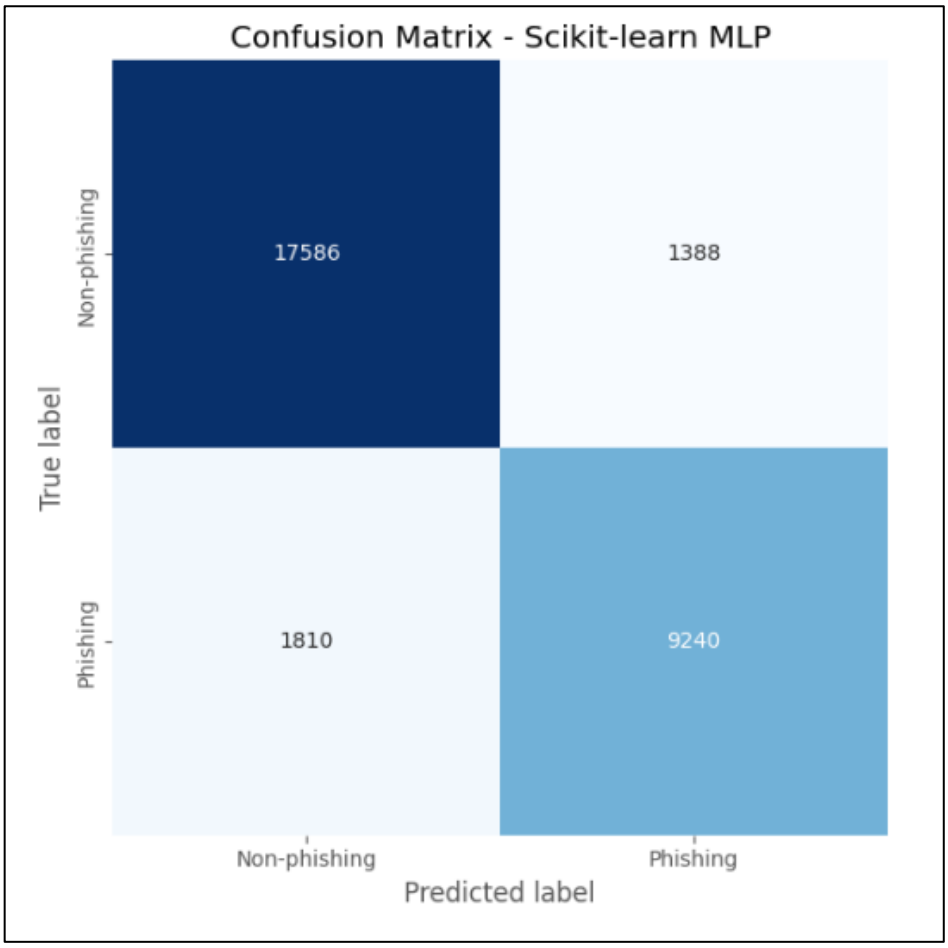


Рисунок 3.44 – Матриця плутанини Scikit-learn MLP

Графік ROC-кривої для Scikit-learn MLP демонструє площу під кривою AUC 0.96. Це найкращий показник серед усіх нейронних мереж і він відповідає результатам провідних ансамблевих моделей. На рисунку 3.45 представлено ROC-криву, яка є стандартним графіком для оцінки продуктивності бінарного класифікатора, оскільки вона візуалізує залежність між чутливістю True Positive Rate та специфічністю False Positive Rate при варіації порогу класифікації.

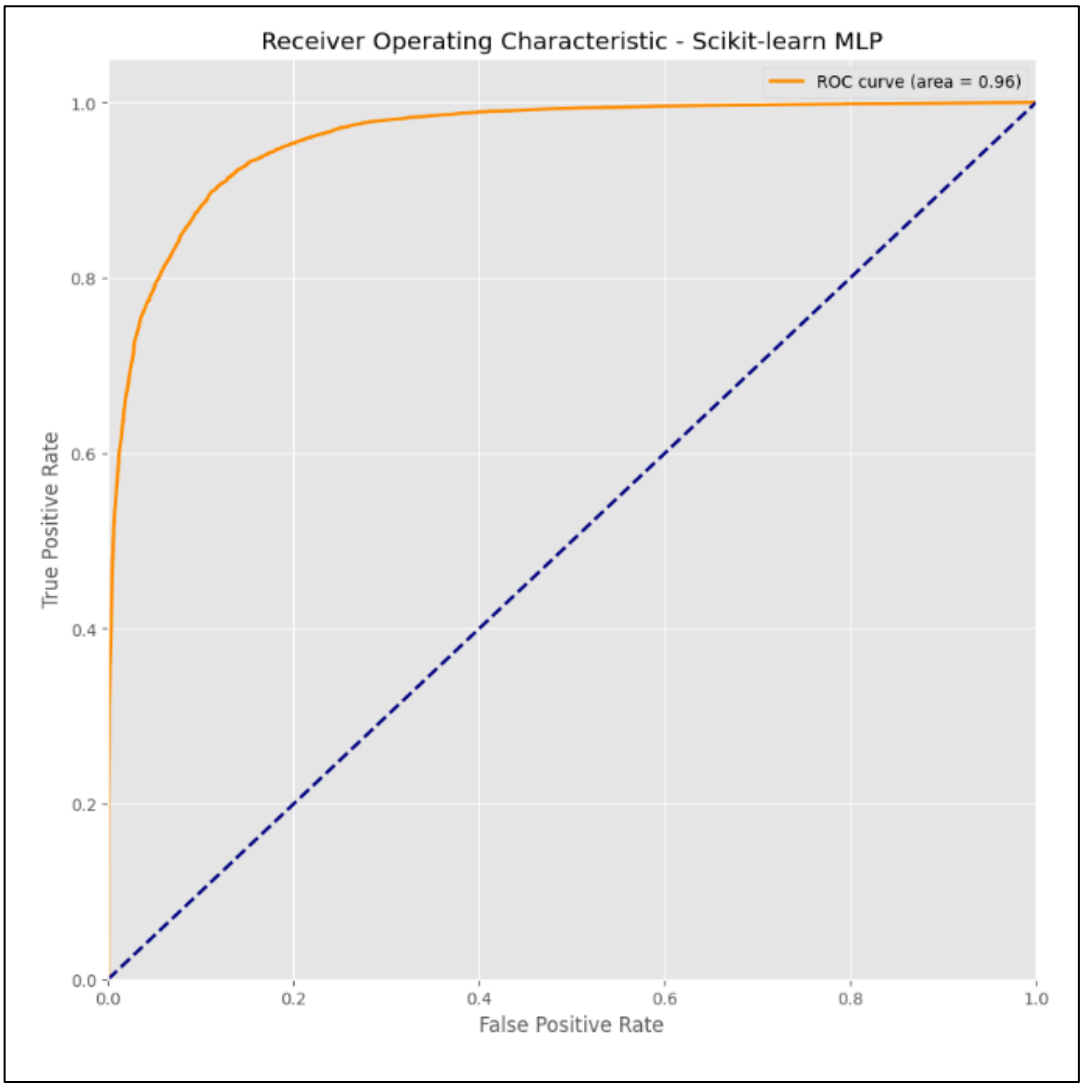


Рисунок 3.45 – Графік ROC кривої Scikit-learn MLP

Аналіз важливості ознак для Scikit-learn MLP наведено на рисунку 3.46. На відміну від ансамблевих методів, нейронна мережа не використовує таку ж пряму оцінку важливості. Однак, було визначено, що найбільш значущими ознаками є довжина URL, найважливіша ознака з вагою близько 0.175. Кількість скісних рисок також дуже важлива з вагою близько 0.175. Кількість крапок значно поступається близько 0.025. Цей розподіл показує, що MLP покладається на багатомірну комбінацію довжини URL та кількості скісних рисок, як і багато інших ефективних моделей.

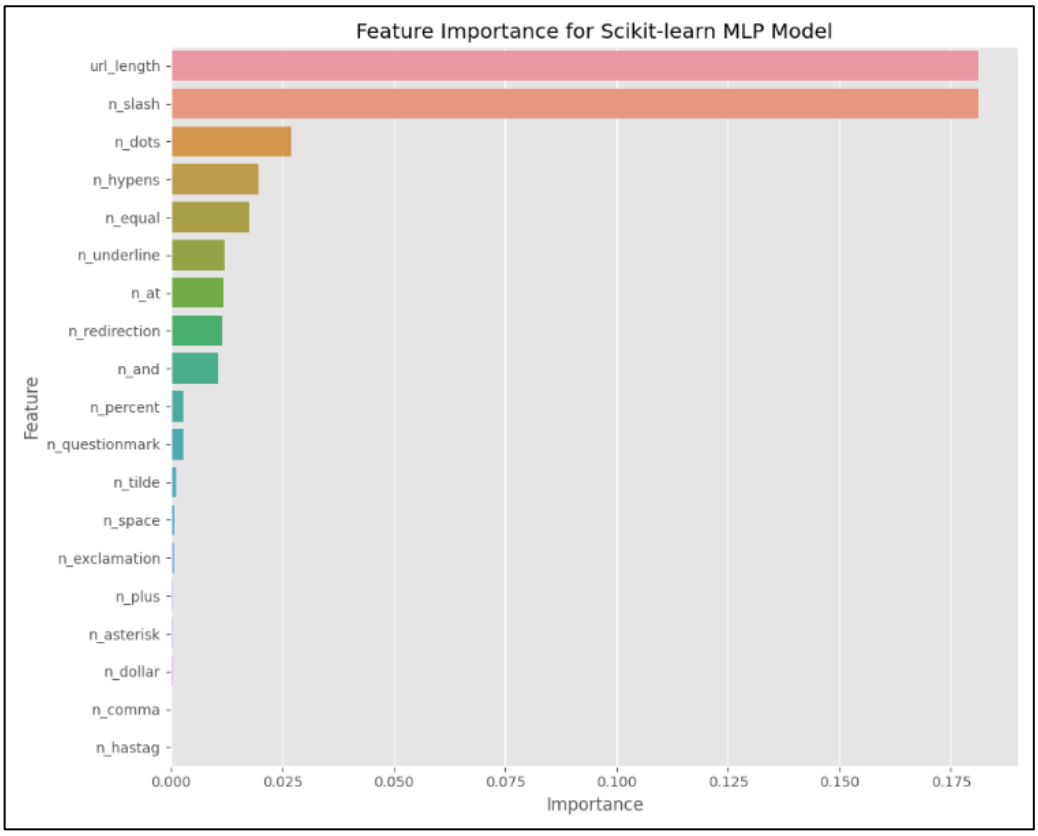


Рисунок 3.46 – Графік важливості ознак Scikit-learn MLP

Модель Scikit-learn MLP є найкращою нейронною мережею для цієї задачі за показниками. Вона майже досягає рівня продуктивності найкращих ансамблевих методів. Однак, незважаючи на високу точність, вона має більшу кількість пропущених загроз порівняно з LightGBM та Random Forest, що робить її менш придатною для критичних завдань кібербезпеки.

3.2 Вибір оптимальних моделей

На рисунку 3.47 представлено графік порівняння методів машинного навчання на тестовій вибірці.

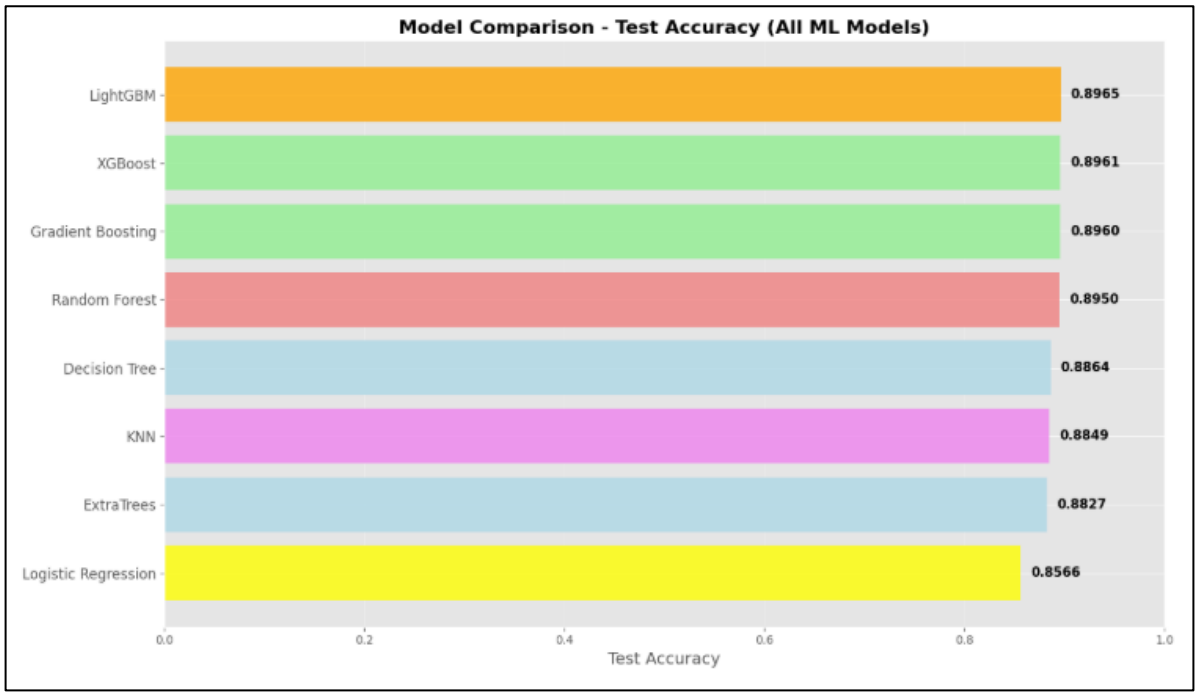


Рисунок 3.47 – Графік порівнянь методів

На рисунку 3.48 представлено графік порівняння методів машинного навчання за метрикою F1-Score.

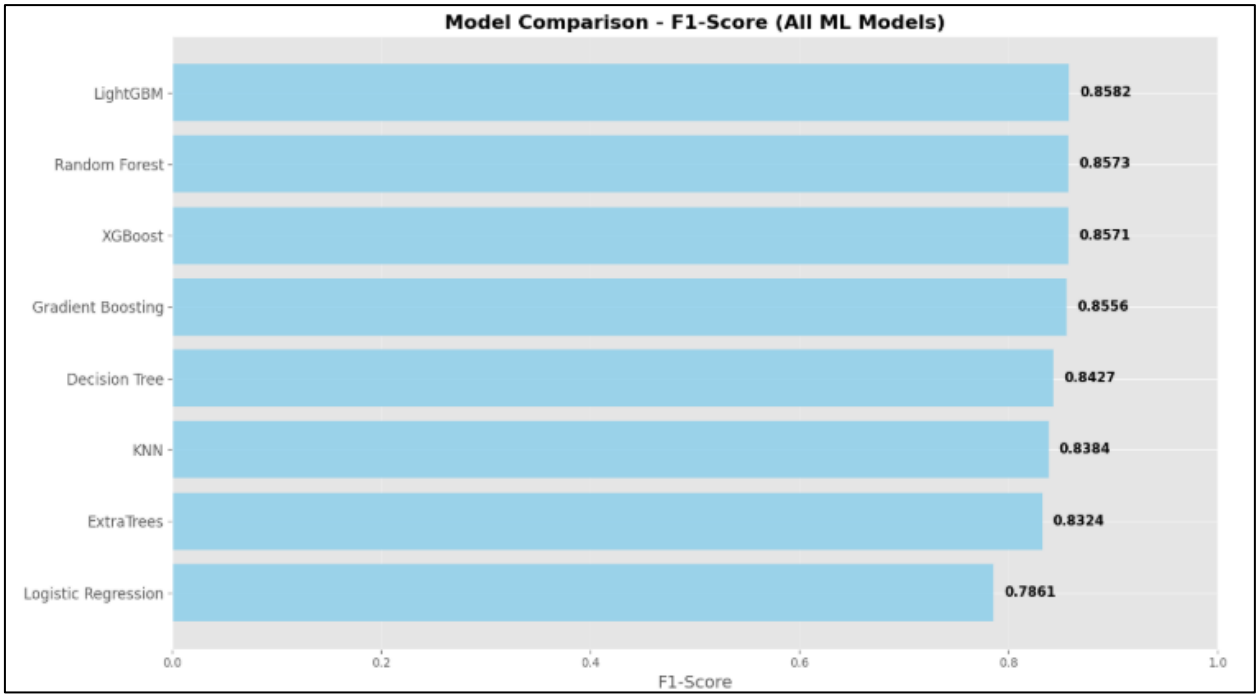


Рисунок 3.48 – Графік порівнянь методів за метрикою F1-Score

У таблиці 3.1 представлено результати всіх методів машинного навчання.

Таблиця 3.1 – Результати передбачення

Назва	Train Accuracy Score	Test Accuracy Score	Precision	Recall	F1_Score
Логістична регресія	0.8582	0.8566	0.8717	0.7157	0.7861
Decision Tree	0.8900	0.8864	0.8591	0.8270	0.8427
Random Forest	0.9085	0.8950	0.8570	0.8576	0.8573
Gradient Boosting	0.9115	0.8960	0.8748	0.8373	0.8556
XGBoost	0.9100	0.8961	0.8679	0.8466	0.8582
LightGBM	0.9065	0.8965	0.8652	0.8514	0.8582
ExtraTrees	0.8942	0.8827	0.8779	0.7914	0.8324
KNN	0.9223	0.8849	0.8672	0.8114	0.8384

Проаналізувавши результати можна сказати, що найкращим методом є LightGBM.

У таблиці 3.2 представлено результати всіх нейронних мереж.

Таблиця 3.2 – Результати передбачення

Назва	Accuracy	Val_Accuracy
Neural Network 1 (Shallow, ReLU, High Dropout)	0.8834	0.8836
Neural Network 2 (Deep, Tanh, Moderate Dropout)	0.8737	0.8746
Neural Network 3 (Medium, Mixed, Low Dropout)	0.8719	0.8728
Scikit-learn MLP	0.9002	0.8935

Проаналізувавши результати можна сказати, що найкращою нейронною мережою є Scikit-learn MLP.

3.3 Висновки

У даному розділі виконали розроблення та комплексне тестування інформаційної технології для передбачення фішингових сайтів, провівши порівняльний аналіз восьми класичних моделей машинного навчання та чотирьох конфігурацій нейронних мереж на основі структурних ознак URL-адрес. Дослідження підтвердило, що найкращою моделлю для впровадження є LightGBM з точністю 0.8965, оскільки вона продемонструвала найвищу точність та найвищий F1-Score 0.8582 серед усіх протестованих алгоритмів, забезпечуючи оптимальний баланс між високою точністю виявлення загроз та мінімізацією хибнопозитивних спрацювань. Серед нейронних мереж найкращі результати показала модель Scikit-learn MLP, досягнувши точності 0.8935, що є найкращим показником серед усіх протестованих конфігурацій глибокого навчання.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного та технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробленого програмного засобу для передбачення фішингових сайтів на основі методів машинного навчання.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями [21].

Таблиця 4.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	3	5	3
2. Ринкові переваги (наявність аналогів)	3	3	4
3. Ринкові переваги (ціна продукту)	3	2	2
4. Ринкові переваги (технічні властивості)	2	2	3
5. Ринкові переваги (експлуатаційні витрати)	3	3	4
6. Ринкові перспективи (розмір ринку)	2	3	2
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	4	5	4
9. Практична здійсненність (наявність фінансів)	2	2	3
10. Практична здійсненність (необхідність нових матеріалів)	4	3	4
11. Практична здійсненність (термін реалізації)	5	4	4

Продовження таблиці 4.1

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
12. Практична здійсненність (розробка документів)	3	3	4
Сума балів	39	38	*40*
Середньоарифметична сума балів $СБ_с$	39		

За результатами розрахунків, наведених в таблиці 4.1, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки.

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія аналізу та передбачення фішингових сайтів методами машинного навчання» становить 39 бали, що, відповідно до [21], свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього)

4.2 Розрахунок узагальненого коефіцієнта якості розробки

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення розрахуємо за формулою [22]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i \quad , \qquad (4.1)$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

α_i – коефіцієнт, який враховує питому вагу *і-го* технічного показника в загальній якості розробки. Коефіцієнт α_i визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

β_i – відносне значення *і-го* технічного показника якості нової розробки.

Результати порівняння зведемо до таблиці 4.2.

Таблиця 4.2 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований продукт	Відношення параметрів нової розробки до аналога	Питома вага показника
1. Кількість використаних моделей машинного навчання	од.	4	8	2	0.2
2. Попередня обробка та очистка даних	од.	1	2	2	0.15
3. Точність передбачення	%	93	96	1.03	0.3
4. Кількість графіків розвідувального аналізу	од.	4	7	1.75	0.2
5. Кількість використаних багатошарових нейронних мереж	од.	1	4	4	0.15

Узагальнений коефіцієнт якості (Вн) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 2 \cdot 0{,}2 + 2 \cdot 0{,}15 + 1{,}03 \cdot 0{,}3 + 1{,}75 \cdot 0{,}2 + 4 \cdot 0{,}15 = 1{,}95.$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,95 рази.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему

«Інформаційна технологія аналізу та передбачення фішингових сайтів методами машинного навчання», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.3.1 Витрати на оплату праці

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [21]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.2)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=20 день.

$$З_о = 24900{,}00 \cdot 10 / 20 = 12450{,}00 \; грн.$$

Проведені розрахунки зведемо до таблиці 4.3.

Таблиця 4.3 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту (проєктний менеджер)	24900,00	1245,00	10	12450,00
Консультант (лікар онколог вищої категорії)	22100,00	1105,00	5	5525,00
Інженер-програміст	21000,00	1050,00	25	26250,00
Інженер-аналітик (системний аналіз)	21600,00	1080,00	10	10800,00
Консультант-аналітик цифрових обчислюваних систем	25300,00	1265,00	5	6325,00

Продовження таблиці 4.3

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Всього				61350,00

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Інформаційна технологія аналізу та передбачення раку легень методами машинного навчання» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \tag{4.3}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \tag{4.4}$$

де M_M – розмір мінімальної місячної заробітної плати, приймемо $Мм$=8000,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення (табл. Б.2, додаток Б) [21];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок;

T_p – середнє число робочих днів в місяці, приблизно T_p = 20 дн;

$t_{зм}$ – тривалість зміни, год.

$$C_1 = 8000{,}00 \cdot 1{,}10 \cdot 1{,}15 / (20 \cdot 8) = 63{,}25 \text{ грн.}$$

$$З_{р1} = 63{,}25 \cdot 8{,}00 = 506{,}00 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці 4.4.

Таблиця 4.4 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Монтаж робочого місця розробника системи прогнозування	11,00	2	1,10	63,25	695,75
Інсталяція програмного забезпечення	6,00	5	1,70	97,75	586,50
Встановлення цифрових обчислювальних систем	3,00	4	1,50	86,25	258,75
Налаштування модулів	7,00	5	1,70	97,75	684,25
Тренування цифрової експериментальної моделі	4,00	4	1,50	86,25	345,00
Формування бази даних прогнозного аналізу	15,00	3	1,35	77,63	1164,45
Інші допоміжні роботи	10,00	3	1,35	77,63	776,30
Монтаж серверного обладнання	4	4	1,50	86,25	345,00
Всього					4856,00

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.5)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 10%.

$З_{дод}$ = (61350,00 + 4856,00) · 10 / 100% = 6620,60 грн.

4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.6)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (61350,00 + 4856,00 + 6620,60) · 22 / 100% = 16021,85 грн.

4.3.3 Сировина та матеріали

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{в\,j}, \quad (4.7)$$

де $Н_j$ – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*);

$В_j$ – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1$ = 3,000 · 170,00 · 1,05 - 0 · 0 = 535,50 грн.

Проведені розрахунки зведемо до таблиці 4.5.

Таблиця 4.5 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 од, грн	Норма витрат, од.	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір канцелярський офісний Maestro Standart (A4-500)	170,00	3,000	0	0	510,00
Начиння канцелярське Buromax	200,00	3,000	0	0	600,00
Органайзер офісний Buromax	244,00	3,000	0	0	732,00
Картридж для принтера Canon CLI-471Bk	1200,00	2,000	0	0	2400,00
Диск оптичний Sony (CD-R)	35,00	4,000	0	0	140,00
Диск оптичний Sony (CD-RW)	37,00	4,000	0	0	148,00
FLASH-пам'ять Kingstar (32 ГБ) Class 10	250,00	1,000	0	0	250,00
FLASH-пам'ять Kingstar (64 ГБ) Class 10 A	320,00	1,000	0	0	320,00
Всього					5100,00

4.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна технологія аналізу та передбачення фішингових сайтів методами машинного навчання», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.8)$$

де $Н_j$ – кількість комплектуючих j-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих *j*-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*).

$К_в = 1 \cdot 3500{,}00 \cdot 1{,}05 = 3675{,}00$ *грн.*

Проведені розрахунки зведемо до таблиці 4.6.

Таблиця 4.6 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Wi-Fi роутер Mikrotik hAP ac3 (RBDAPG-2HND)	1	3500,00	3675,00
Ліцензія на ОС Windows 11 Pro	1	4800,00	5040,00
SSD диск Crucial P5 Plus 1TB NVMe M.2	2	2200,00	4620,00
Оперативна пам'ять Kingston Fury 16GB (2x8GB)	1	2400,00	2520,00
Всього			15855,00

4.3.5 Спецустаткування для наукових (експериментальних) робіт

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \,, \qquad (4.9)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

$К_i$ – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($К_i$ = *1,10…1,12);*

k – кількість найменувань устаткування.

$B_{спец} = 60000{,}00 \cdot 1 \cdot 1{,}05 = 24953{,}25$ *грн.*

Отримані результати зведемо до таблиці 4.7.

Таблиця 4.7 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Ноутбук MSI Katana B12V	1	60000,00	63000,00
Маршрутизатор TP-LINK Archer C20 WiFi5	1	1500,00	1575,00
Всього			64575,00

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.10)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 9230,00 \cdot 1 \cdot 1,05 = 9355,50$ *грн.*

Отримані результати зведемо до таблиці 4.8.

Таблиця 4.8 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Середовище мови програмування Python	1	9230,00	9691,50
Середовище розробки програмного забезпечення PyCharm	1	7510,00	7885,50
Програмне забезпечення розробки Kaggle	1	1199,00	1258,95
Високошвидкісний доступ до мережі Internet (міс)	2	250,00	525,00
Бібліотеки для машинного навчання Scikit-Learn, PyTorch, TensorFlow	1	7650,00	8032,50
Всього			27393,45

4.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.11)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (65000{,}00 \cdot 2) / (3 \cdot 12) = 3611{,}11$ *грн.*

Проведені розрахунки зведемо до таблиці 4.9.

Таблиця 4.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Програмно-аналітичний комплекс ПК Intel Core i7-14700K / RAM 32ГБ / SSD 2ТБ / nVidia GeForce RTX 4070 12ГБ	65000,00	3	2	3611,11
Персональний комп'ютер Ноутбук Dell G15 5530 (N101L5530H1607W)	42000,00	3	2	2333,33
Спеціалізоване робоче місце розробника інформаційної технології	8400,00	5	2	280,00
Пристрій виводу текстової інформації Принтер HP Laser	11299,00	4	2	410,79
Оргтехніка Samsung Office	11399,00	5	2	379,97
ОС Windows 11 Pro	8628,00	3	2	479,33
Прикладний пакет Microsoft Office 2021 Professional Plus	7320,00	3	2	406,67
Мережеве обладнання передачі цифрових даних	6700,00	4	2	279,17
Всього				8240,37

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \qquad (4.12)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; приймемо $Ц_e = 12,56$ грн;

$K_{вті}$ – коефіцієнт, що враховує використання потужності, $K_{вті} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,40 \cdot 240,0 \cdot 12,56 \cdot 0,95 / 0,97 = 1180,80$ *грн.*

Проведені розрахунки зведемо до таблиці 4.10.

Таблиця 4.10 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Програмно-аналітичний комплекс ПК Intel Core i7-14700K / RAM 32ГБ / SSD 2ТБ / nVidia GeForce RTX 4070 12ГБ	0,40	240,0	1180,80
Персональний комп'ютер Ноутбук Dell G15 5530 (N101L5530H1607W)	0,06	240,0	177,12
Спеціалізоване робоче місце розробника інформаційної технології	0,08	240,0	236,16
Пристрій виводу текстової інформації Принтер HP Laser	0,03	4	1,48
Мережеве обладнання передачі цифрових даних	0,07	240,0	206,66
Серверне обладнання на основі ПК ARTLINE Business B71v31 (Intel Core i5-13400F + GTX 1650)	0,40	240,0	1180,80
Всього			2983,02

4.3.9 Службові відрядження

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (4.13)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$В_{св}$ = (61350,00 + 4856,00) · 20 / 100% = 13241,20 грн.

4.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (4.14)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$В_{сп}$ = (61350,00 + 4856,00) · 30 / 100% = 19861,80 грн.

4.3.11 Інші витрати

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_{в} = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.15)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_{в}$ = (61350,00 + 4856,00) · 50 / 100% = 33103,00 грн.

4.3.12 Накладні (загальновиробничі) витрати

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{H_{нзв}}{100\%}, \quad (4.16)$$

де $H_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 110%.

$B_{нзв}$ = (61350,00 + 4856,00) · 110 / 100% = 72826,60 грн.

Витрати на проведення науково-дослідної роботи на тему «Інформаційна технологія аналізу та передбачення фішингових сайтів методами машинного навчання» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + І_в + B_{нзв}. \quad (4.17)$$

$B_{заг}$= 61350,00 + 4856,00 + 6620,60 + 16021,85 + 5100,00 + 15855,00 + 64575,00 + 27393,45 + 8240,37 + 2983,02 + 13241,20 + 19861,80 + 33103,00 + 72826,60 = 352021,89 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.18)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

ЗВ = 352021,8 / 0,9 = 391135,43 грн.

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

Результати дослідження проведені за темою «Інформаційна технологія аналізу та передбачення фішингових сайтів методами машинного навчання» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть

формувати:

ΔN – збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

Отримані результати зведемо до таблиці 4.11.

Таблиця 4.11 – Кількість споживачів

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	400	700	900	500

N– кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 4000 осіб;

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 8000,00 грн;

$\pm\Delta Ц_о$ – зміна вартості послуги від впровадження результатів, приймемо 1900,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [21]:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.19)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2025 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо $\rho = 39\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний

інвестор, у 2025 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (1900{,}00 \cdot 4000 + 9900{,}00 \cdot 400) \cdot 0{,}83 \cdot 0{,}39 \cdot (1\text{-}0{,}18/100\%) =$ 3736791,82 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (1900{,}00 \cdot 4000 + 9900{,}00 \cdot 700) \cdot 0{,}83 \cdot 0{,}39 \cdot (1\text{-}0{,}18/100\%) =$ 4694432,15 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (1900{,}00 \cdot 4000 + 9900{,}00 \cdot 900) \cdot 0{,}83 \cdot 0{,}39 \cdot (1\text{-}0{,}18/100\%) =$ 5333552,78 грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4 = (1900{,}00 \cdot 4000 + 9900{,}00 \cdot 500) \cdot 0{,}83 \cdot 0{,}39 \cdot (1\text{-}0{,}18/100\%) =$ 4056262,40 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.20)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}1$;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 3736791,82/(1+0,1)^1+4694432,15/(1+0,1)^2+5333552,78/(1+0,1)^3+$$
$$+4056262,40\ /(1+0,1)^4=$$
$$3397083,47+3879695,99+4007177,14+2770462,81=14054419,41 \text{ грн.}$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{4.21}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 391135,43 грн.

$PV = k_{інв} \cdot ЗВ$= 2 · 391135,43 = 782270,86 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.22}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 11533167,93 грн;

PV – теперішня вартість початкових інвестицій, 782270,86 грн.

$E_{абс} = ПП - PV$ = 14054419,41 - 782270,86 = 13272148,55 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.23}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 13272148,55 грн;

PV – теперішня вартість початкових інвестицій, 782270,86 грн;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_{в} = \sqrt[Tж]{1+\frac{E_{абс}}{PV}} - 1 = (1+13272148{,}55/782270{,}86)^{1/4} - 1 = 1{,}06.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \tag{4.24}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,35.

$\tau_{мін}$= 0,1+0,25 = 0,35 < 1,06 свідчить про те, що внутрішня економічна дохідність інвестицій $E_{в}$, вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія аналізу та передбачення фішингових сайтів методами машинного навчання» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_{в}}, \tag{4.25}$$

де $E_{в}$ – внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = 1 / 1{,}06 = 0{,}94 \text{ р.}$$

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

4.5 Висновки

Згідно з проведеними дослідженнями, рівень комерційного потенціалу розробки за темою «Інформаційна технологія аналізу та передбачення фішингових сайтів методами машинного навчання» становить 39 бала, що свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки є високим). При оцінюванні за технічними параметрами, згідно з узагальненим коефіцієнтом якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,95 рази. Економічні розрахунки підтвердили високу привабливість проекту: Чиста поточна вартість склала 13272148,55 грн. Крім того, термін окупності проекту є дуже коротким, становлячи 0,94 р., що значно менше за 3-х років. Це свідчить про високу комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок. Отже, можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія аналізу та передбачення фішингових сайтів методами машинного навчання» та її високу інвестиційну привабливість.

ВИСНОВКИ

У першому розділі проаналізовано та описано об’єкт дослідження предметної області, а саме проблема фішингового шахрайства в Інтернеті, її основні види та обґрунтовано вибір оптимальних методів. Для розв’язання поставлених задач обрано мову програмування Python та такі методи машинного навчання як Logistic Regression, Decission Tree, Random forest, Gradient boosting trees, XGBoost Model, LightGBM Model, ExtraTrees Model, K-Nearest Neighbors Model (KNN). Для розпізнавання фішингових сайтів у рамках машинного навчання також використовувались нейронні мережі, зокрема багатошаровий перцептрон Multilayer Perceptron MLP.

У другому розділі вибрано та описано датасет «Web Page Phishing Dataset» з 20 ознаками, а проведений розвідувальний аналіз даних показав, що найбільш поширеними та інформативними ознаками фішингових сайтів є структурні аномалії URL, зокрема, косі риски та дефіси.

У третьому розділі розроблено інформаційну технологію передбачення фішингових сайтів на базі цього датасету з використанням широкого спектру методів машинного навчання, включаючи ансамблеві моделі та нейронні мережі. Дослідження підтвердило, що найкращою моделлю є LightGBM, яка досягла найвищої точності 0.8965 та найвищого F1-Score 0.8582 забезпечуючи оптимальний баланс між виявленням загроз та мінімізацією помилкових спрацювань. Серед нейронних мереж найкращі результати показала модель Scikit-learn MLP точністю 0.8935.

У четвертому розділі проведено оцінку комерційного потенціалу інформаційної технології, що засвідчило його високий рівень (39 балів). Економічні розрахунки підтвердили високу інвестиційну привабливість проекту: чиста поточна вартість склала 13272148,55 грн, а термін окупності є надзвичайно коротким — 0,94 р., що значно менше за 3-х років. Таким чином, розроблена технологія, є ефективним інструментом у боротьбі з фішингом, забезпечуючи надійний захист для користувачів Інтернету.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Литвиненко Д. О. ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ ФІШИНГОВИХ САЙТІВ МЕТОДАМИ МАШИННОГО НАВЧАННЯ. *Всеукраїнська науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця, 2025-2026 рр.)*. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/26556/21953

2. Ayaburi, E. W., & Andoh-Baidoo, F. K. How do technology use patterns influence phishing susceptibility? A two-wave study of the role of reformulated locus of control. *European Journal of Information Systems*. 2023. Vol. 5 (12). P. 1-21. DOI: https://doi.org/10.1080/0960085X.2023.2186275

3. Чекмарьова. І. М. Шахрайство в Інтернеті як один із видів шахрайства. *Аналітично-порівняльне правознавство №2. Розділ 8. Кримінальне право та кримінологія*. 2024. С. 639-643. DOI: https://doi.org/10.24144/2788-6018.2024.02.106

4. Бережна О. Б., П. Р. Васькевич. Тенденції використання нейромереж у розробці веб-сайтів. *Міжнар. наук.-техн. конф. Харків: ТОВ «Друкарня Мадрид»*. 2024. С. 67-68. URL: https://openarchive.nure.ua/handle/document/26691

5. Гарасимчук О., Оліярник Ю., Нестор А., Наконечний Т. ПСИХОЛОГІЧНІ МЕТОДИ ШАХРАЙСТВА В КІБЕРПРОСТОРІ ТА СПОСОБИ ЇМ ПРОТИДІЯТИ. *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»*. 2025. С. 511–529 URL: https://csecurity.kubg.edu.ua/index.php/journal/article/view/990

6. Anti-Phishing Working Group. URL: https://apwg.org/

7. Anti-Phishing Working Group. Phishing Activity Trends Report, 1st Quarter. 2025. P. 1-13. URL: https://docs.apwg.org/reports/apwg_trends_report_q1_2025.pdf

8. Штонда Р. М., Черниш Ю. О., Терещенко Т. П., Терещенко К. В., Цикало Ю. Г., Поліщук С. А. Класифікація та методи виявлення фішингових атак. 2024. С. 1-12. URL: https://csecurity.kubg.edu.ua/index.php/journal/article/view/591/464

9. Дика А. І., Трофименко О. Г., Ковальжі А. С. ІНТЕЛЕКТУАЛЬНЕ ВИЯВЛЕННЯ ФІШИНГОВИХ СТОРІНОК У СЕРЕДОВИЩІ ВЕБЗАСТОСУНКІВ ІЗ ЗАСТОСУВАННЯМ ВІЗУАЛЬНОГО ПОДАННЯ. 2025. С. 1-11. DOI: https://doi.org/10.32782/2663-5941/2025.1.2/43

10. Google Safe Browsing. URL: https://safebrowsing.google.com/

11. Підручник з Python. URL: https://docs.python.org/uk/3.13/tutorial/index.html

12. Kaggle. URL: https://www.kaggle.com

13. Петро Кравець, Володимир Пасічник, Микола проданюк. МАТЕМАТИЧНА МОДЕЛЬ ЛОГІСТИЧНОЇ РЕГРЕСІЇ ДЛЯ БІНАРНОЇ КЛАСИФІКАЦІЇ. *Національний університет «Львівська політехніка».* 2024. С. 290-321. DOI: https://doi.org/10.23939/sisn2024.15.290

14. Меженна І. Д. Дослідження та порівняння методів навчання для прогнозування погодних умов на основі метеорологічних. *М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки.* 2025. С. 35-39. URL: https://openarchive.nure.ua/handle/document/29652

15. Зорін М. М. Бегінг в ансамблях нейронних мереж для адаптації кольорів зображень до специфічних стилістичних, історичних або художніх контекстів. *М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки*. 2025. С. 14-27. URL: https://openarchive.nure.ua/handle/document/32366

16. Янкович С. Є. Розробка вебзастосунку для оптимального обрахунку логістики із урахуванням поточної завантаженості доріг на основі рандомізованих ансамблевих алгоритмів. *М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки*. 2025. С. 27-35. URL: https://openarchive.nure.ua/handle/document/32960

17. Джоші А., Павленко В. І. ПОРІВНЯННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ВИЗНАЧЕННЯ ФЕЙКОВИХ НОВИН. *Інфокомунікаційні та комп'ютерні технології*. 2024. С. 46-55. DOI: https://doi.org/10.36994/2788-5518-2024-01-07-06

18. Довідник по Machine Learning – Multilayer Perceptron. URL: https://itwiki.dev/data-science/ml-reference/ml-glossary/multilayer-perceptron

19. FERNANDO DANIEL. Web Page Phishing Dataset. 2024. URL: https://www.kaggle.com/datasets/danielfernandon/web-page-phishing-dataset/data?select=web-page-phishing.csv

20. Литвиненко Д. О. Web Phishing Analysis. 2024. URL: https://www.kaggle.com/code/linean/web-phishing-analysis

21. Козловський В. О., Лесько О. Й., Кавецький В. В. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. *Вінницький національний технічний університет*. 2021. С. 42. URL: https://epvm.vntu.edu.ua/wp-content/uploads/2021/09/Ekonom_magister_tehn_2021.pdf

22. Кавецький В. В., Козловський В. О., Причепа І. В. Економічне обґрунтування інноваційних рішень: практикум. *Вінницький національний технічний університет*. 2016. С. 113. URL: https://epvm.vntu.edu.ua/wp-content/uploads/2018/01/EOIP_prakt_kavetsky.pdf

Додаток А

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«___» _____ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу
«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ДЛЯ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ ФІШИНГОВИХ САЙТІВ МЕТОДАМИ МАШИННОГО НАВЧАННЯ»
08-34.МКР.004.02.000.ТЗ

Керівник: к.т.н., доц. каф. САІТ
_____ Олексій КОЗАЧКО
«___» _____ 2025 р.

Розробив: студент гр. 2ІСТ-24м
_____ Данило ЛИТВИНЕНКО
«___» _____ 2025 р.

Вінниця 2025

1. Підстава для проведення робіт

Підставою для виконання роботи є наказ № __ по ВНТУ від «__» _____ 2025 р., та індивідуальне завдання на МКР, затверджене протоколом № __ засідання кафедри САІТ від «__» _____ 2025 р.

2. Джерела розробки:

1) Anti-Phishing Working Group. Phishing Activity Trends Report, 1st Quarter. 2025. P. 1-13. URL: https://docs.apwg.org/reports/apwg_trends_report_q1_2025.pdf

2) Ayaburi, E. W., & Andoh-Baidoo, F. K. How do technology use patterns influence phishing susceptibility? A two-wave study of the role of reformulated locus of control. European Journal of Information Systems. 2023. Vol. 5 (12). P. 1-21. DOI: https://doi.org/10.1080/0960085X.2023.2186275

3. Мета і призначення роботи:

Підвищення точності інформаційної технології для виявлення та передбачення фішингових веб-сайтів за допомогою нових методів машинного навчання.

4.Вихідні дані для проведення робіт:

FERNANDO DANIEL. Web Page Phishing Dataset. 2024. URL: https://www.kaggle.com/datasets/danielfernandon/web-page-phishing-dataset/data?select=web-page-phishing.csv5.

5. Методи дослідження:

Використання платформи Kaggle, мови програмування Python та методів машинного навчання та нейронних мереж.

6. Етапи роботи і терміни їх виконання:

1. Аналіз сучасного стану моніторингу фішингових сайтів .. ____ – ____
2. Основні етапи виконання роботи та огляд набору вхідних даних____ – ____
3. Обробка результатів прогнозування та візуалізація даних у середовищі Kaggle ..... ____ – ____
4. Розробка інформаційної технології ..... ____ – ____
5. Економічна частина..... ____ – ____
6. Оформлення матеріалів до захисту МКР. ..... ____ – ____

7. Очікувані результати та порядок реалізації:

Отримання інформаційної технології для підвищення точності передбачення фішингових сайтів.

8. Вимоги до розробленої документації

Пояснювальна записка оформлена у відповідності до вимог «Методичних вказівок до виконання магістерських кваліфікаційних робіт для студентів спеціальності 126 «Інформаційні системи та технології» (освітня програма «Інформаційні технології аналізу даних та зображень»)

9. Порядок приймання роботи

Публічний захист..... «___» _____ 2025 р.
Початок розробки ..... «___» _____ 2025 р.
Граничні терміни виконання МКР..... «___» _____ 2025 р.

Розробив студент групи 2ІСТ-24м _____ Данило ЛИТВИНЕНКО

Додаток Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Інформаційна технологія аналізу та передбачення фішингових сайтів методами машинного навчання»

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра САІТ, ФІІТА, гр. 2ІСТ-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 2,5 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне):

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

□ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

□ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Віталій МОКІН, зав. каф. САІТ _____
(підпис)

Сергій ЖУКОВ, доц. каф. САІТ _____
(підпис)

Особа, відповідальна за перевірку _____ Сергій ЖУКОВ
(підпис)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Олексій КОЗАЧКО, к.т.н., доц. каф. САІТ
(підпис)

Здобувач _____ Данило ЛИТВИНЕНКО
(підпис)

Додаток В

Лістинг програми

```
import numpy as np
import pandas as pd
import seaborn as sns
import xgboost as xgb
from scipy import stats
import statsmodels.api as sm
import matplotlib.pyplot as plt

from sklearn.metrics import roc_curve, auc
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model selection import train test split
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import precision_score, recall_score, f1_score
import time

plt.style.use('ggplot')
pd.set option('display.max rows', None)
pd.set_option('display.float_format', lambda x: '%.4f' % x)
```

```
data = pd.read_csv('/kaggle/input/web-page-phishing-dataset/web-page-phishing.csv')
data.head(10)
```

```
data.info()
```

```
y = data.pop('phishing')
```

```
X train, X test, y train, y test = train test split(data, y, test size=0.3, random state=42)
models = {}
models['Logistic Regression'] = LogisticRegression(max_iter=1000)
models['Logistic Regression'].fit(X train, y train)

predictions_train = models['Logistic Regression'].predict(X_train)
predictions_test = models['Logistic Regression'].predict(X_test)

print(f"Train Accuracy Score: {accuracy score(y train, predictions train)}")
print(f"Test Accuracy Score: {accuracy_score(y_test, predictions_test)}")
```

```
def calculate_metrics(model, X_test, y_test):
    predictions = model.predict(X test)
    print(f"Accuracy Score: {accuracy_score(y_test, predictions)}")
    confusion_matrix_plot(y_test, predictions)
```

```
    area_under_curve(model, X_test, y_test)

def confusion_matrix_plot(y_test, predictions):
    cm = confusion_matrix(y_test, predictions)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", square=True, cbar=False,
                xticklabels=['False', 'True'], yticklabels=['False', 'True'])
    plt.title('Confusion Matrix')
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    tick_marks = np.arange(2) + 0.5
    plt.xticks(tick_marks, ['Non-phishing', 'Phishing'], rotation=0)
    plt.yticks(tick_marks, ['Non-phishing', 'Phishing'], rotation=0)
    plt.tight_layout()
    plt.show()

def area_under_curve(model, X_test, y_test):
    y_pred_proba = model.predict_proba(X_test)[:, 1]
    fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(10, 10))
    plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic')
    plt.legend()
    plt.show()

calculate_metrics(models['Logistic Regression'], X_test, y_test)
```

```
models['Decision Tree'] = DecisionTreeClassifier(
    ccp_alpha=0.0001, criterion='gini', max_depth=32, min_samples_split=5, random_state=42
)
models['Decision Tree'].fit(X_train, y_train)

predictions_train = models['Decision Tree'].predict(X_train)
predictions_test = models['Decision Tree'].predict(X_test)

print(f"Train Accuracy Score: {accuracy_score(y_train, predictions_train)}")
print(f"Test Accuracy Score: {accuracy_score(y_test, predictions_test)}")
```

```
models['Random Forest'] = RandomForestClassifier(
    n_estimators=80, max_depth=18, max_features='sqrt', min_samples_split=12, criterion='gini'
)
models['Random Forest'].fit(X_train, y_train)

predictions_train = models['Random Forest'].predict(X_train)
predictions_test = models['Random Forest'].predict(X_test)

print(f"Train Accuracy Score: {accuracy_score(y_train, predictions_train)}")
print(f"Test Accuracy Score: {accuracy_score(y_test, predictions_test)}")
```

```
models['Gradient Boosting'] = GradientBoostingClassifier(
    n_estimators=200, learning_rate=0.01, max_depth=12, subsample=0.8, max_features=0.5, random_state=42
)
models['Gradient Boosting'].fit(X_train, y_train)

predictions_train = models['Gradient Boosting'].predict(X_train)
predictions_test = models['Gradient Boosting'].predict(X_test)

print(f"Train Accuracy Score: {accuracy_score(y_train, predictions_train)}")
print(f"Test Accuracy Score: {accuracy_score(y_test, predictions_test)}")
```

```
models['XGBoost'] = XGBClassifier(
    n_estimators=200,
    max_depth=9,
    learning_rate=0.1,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42,
    eval_metric='logloss'
)

start_time = time.time()
models['XGBoost'].fit(X_train, y_train)

predictions_train = models['XGBoost'].predict(X_train)
predictions_test = models['XGBoost'].predict(X_test)

print(f"Train Accuracy Score: {accuracy_score(y_train, predictions_train):.4f}")
print(f"Test Accuracy Score: {accuracy_score(y_test, predictions_test):.4f}")
```

```
models['LightGBM'] = LGBMClassifier(
    n_estimators=500,
    max_depth=9,
    learning_rate=0.1,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42,
    verbose=-1
)

start_time = time.time()
models['LightGBM'].fit(X_train, y_train)

predictions_train = models['LightGBM'].predict(X_train)
predictions_test = models['LightGBM'].predict(X_test)
```

```
print(f"Train Accuracy Score: {accuracy_score(y_train, predictions_train):.4f}")
print(f"Test Accuracy Score: {accuracy_score(y_test, predictions_test):.4f}")
```

```
models['ExtraTrees'] = ExtraTreesClassifier(
        n_estimators=1500,
        max_depth=25,
        random_state=42,
)

start_time = time.time()
models['ExtraTrees'].fit(X_train, y_train)

predictions_train = models['ExtraTrees'].predict(X_train)
predictions_test = models['ExtraTrees'].predict(X_test)

print(f"Train Accuracy Score: {accuracy_score(y_train, predictions_train):.4f}")
print(f"Test Accuracy Score: {accuracy_score(y_test, predictions_test):.4f}")
```

```
models['KNN'] = KNeighborsClassifier(
        n_neighbors=14,
        weights='distance',
        algorithm='auto',
        leaf_size=20,
        p=2,
        metric='minkowski'
)

start_time = time.time()
models['KNN'].fit(X_train, y_train)
knn_time = time.time() - start_time

predictions_train = models['KNN'].predict(X_train)
predictions_test = models['KNN'].predict(X_test)

print(f"Train Accuracy Score: {accuracy_score(y_train, predictions_train):.4f}")
print(f"Test Accuracy Score: {accuracy_score(y_test, predictions_test):.4f}")
```

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Input
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, auc
from sklearn.inspection import permutation_importance
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Neural Network 1: Shallow network with ReLU and high dropout
def create_nn1(input_shape):
        model = Sequential([
                Input(shape=(input_shape,)),
```

```
            Dense(64, activation='relu'),
            Dropout(0.5),
            Dense(32, activation='relu'),
            Dropout(0.5),
            Dense(1, activation='sigmoid')
        ])
        model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
        return model
```

```
# Neural Network 2: Deeper network with tanh and moderate dropout
def create_nn2(input_shape):
        model = Sequential([
            Input(shape=(input_shape,)),
            Dense(128, activation='tanh'),
            Dropout(0.3),
            Dense(64, activation='tanh'),
            Dropout(0.3),
            Dense(32, activation='tanh'),
            Dense(1, activation='sigmoid')
        ])
        model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
        return model
```

```
# Neural Network 3: Medium network with mixed activations and low dropout
def create_nn3(input_shape):
        model = Sequential([
            Input(shape=(input_shape,)),
            Dense(64, activation='relu'),
            Dropout(0.2),
            Dense(32, activation='elu'),
            Dropout(0.2),
            Dense(16, activation='relu'),
            Dense(1, activation='sigmoid')
        ])
        model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
        return model
```

Додаток Г

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ ФІШИНГОВИХ САЙТІВ МЕТОДАМИ МАШИННОГО НАВЧАННЯ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2025 р.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100077 entries, 0 to 100076
Data columns (total 20 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   url_length       100077 non-null  int64
 1   n_dots           100077 non-null  int64
 2   n_hypens         100077 non-null  int64
 3   n_underline      100077 non-null  int64
 4   n_slash          100077 non-null  int64
 5   n_questionmark   100077 non-null  int64
 6   n_equal          100077 non-null  int64
 7   n_at             100077 non-null  int64
 8   n_and            100077 non-null  int64
 9   n_exclamation    100077 non-null  int64
 10  n_space          100077 non-null  int64
 11  n_tilde          100077 non-null  int64
 12  n_comma          100077 non-null  int64
 13  n_plus           100077 non-null  int64
 14  n_asterisk       100077 non-null  int64
 15  n_hastag         100077 non-null  int64
 16  n_dollar         100077 non-null  int64
 17  n_percent        100077 non-null  int64
 18  n_redirection    100077 non-null  int64
 19  phishing         100077 non-null  int64
dtypes: int64(20)
memory usage: 15.3 MB
```

Рисунок Г.1 – Огляд даних датасету

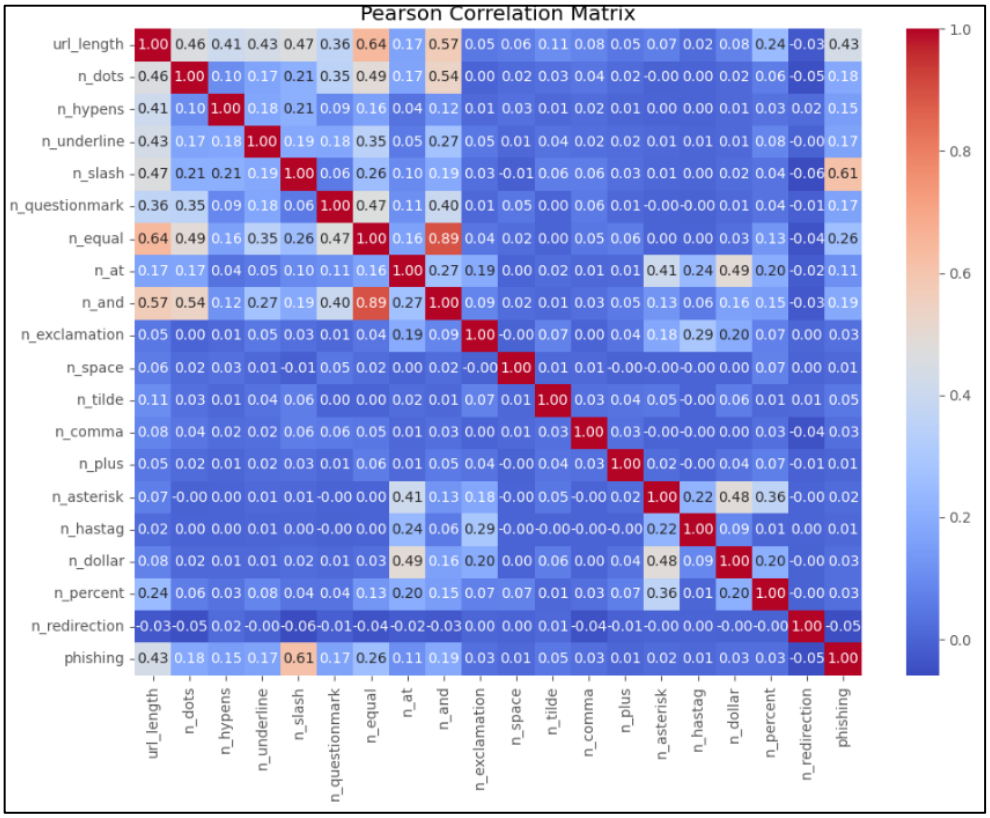


Рисунок Г.2 – Графік теплової карти кореляційної матриці Пірсона

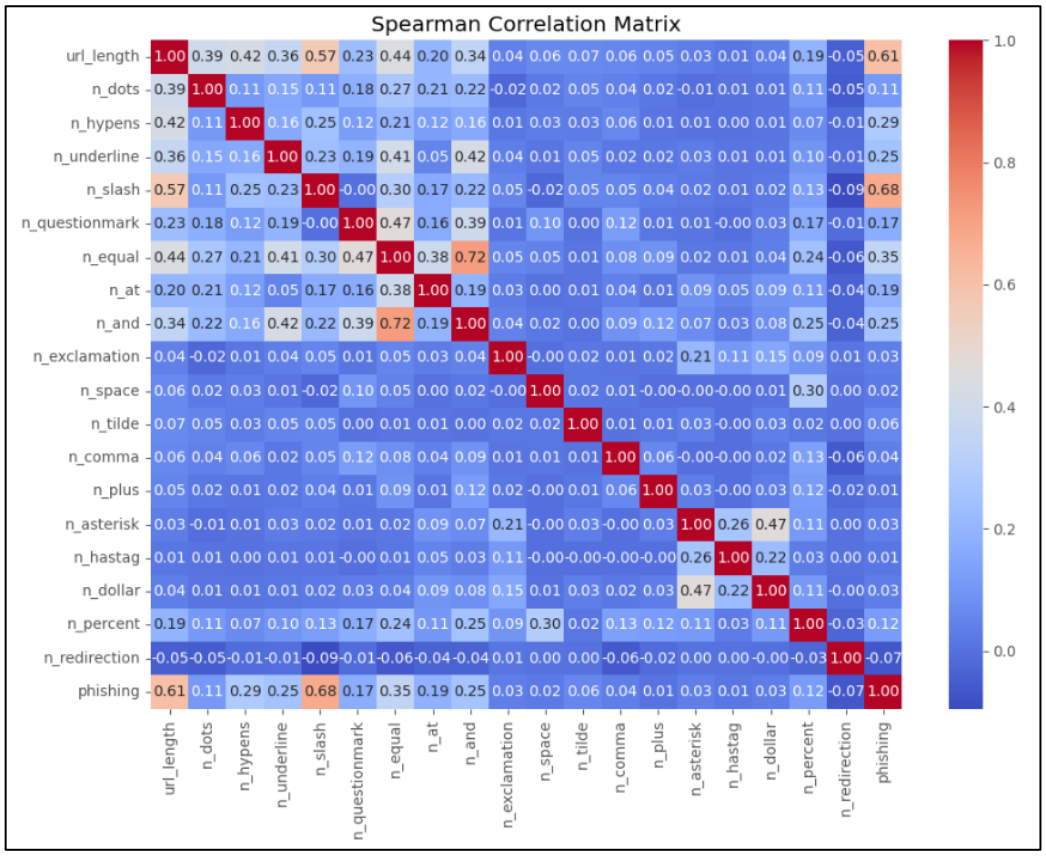


Рисунок Г.3 – Графік теплової карти кореляційної матриці Спірмена

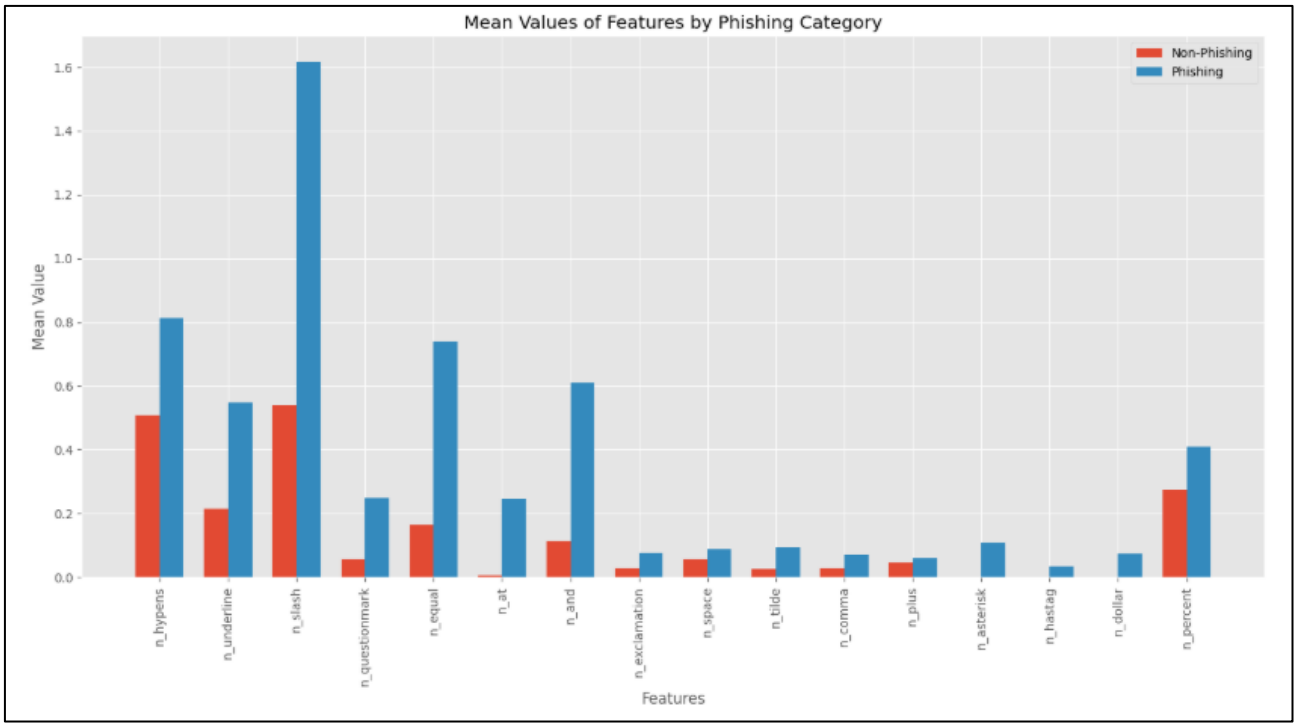


Рисунок Г.4 – Графік стовпчикової діаграми порівняння середніх значень різних ознак для фішингових та не фішингових сайтів

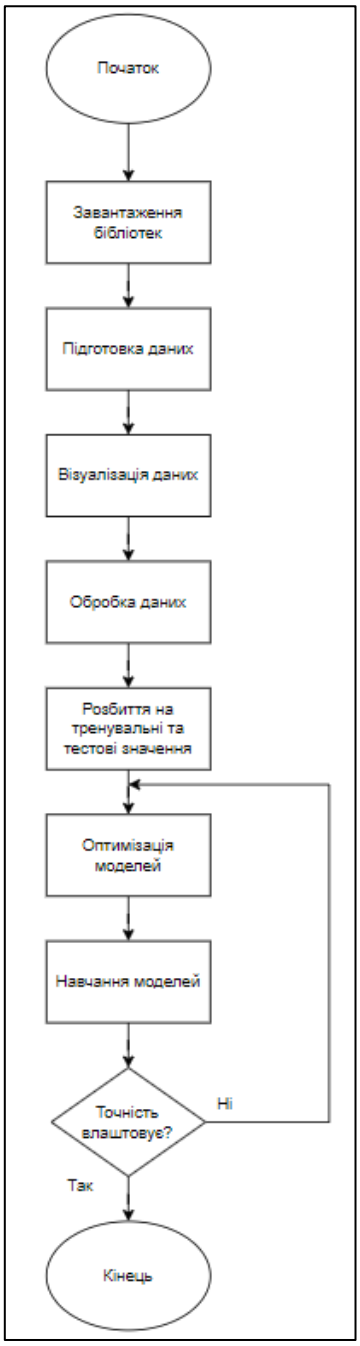


Рисунок Г.5 – Блок-схема алгоритму інформаційної технології

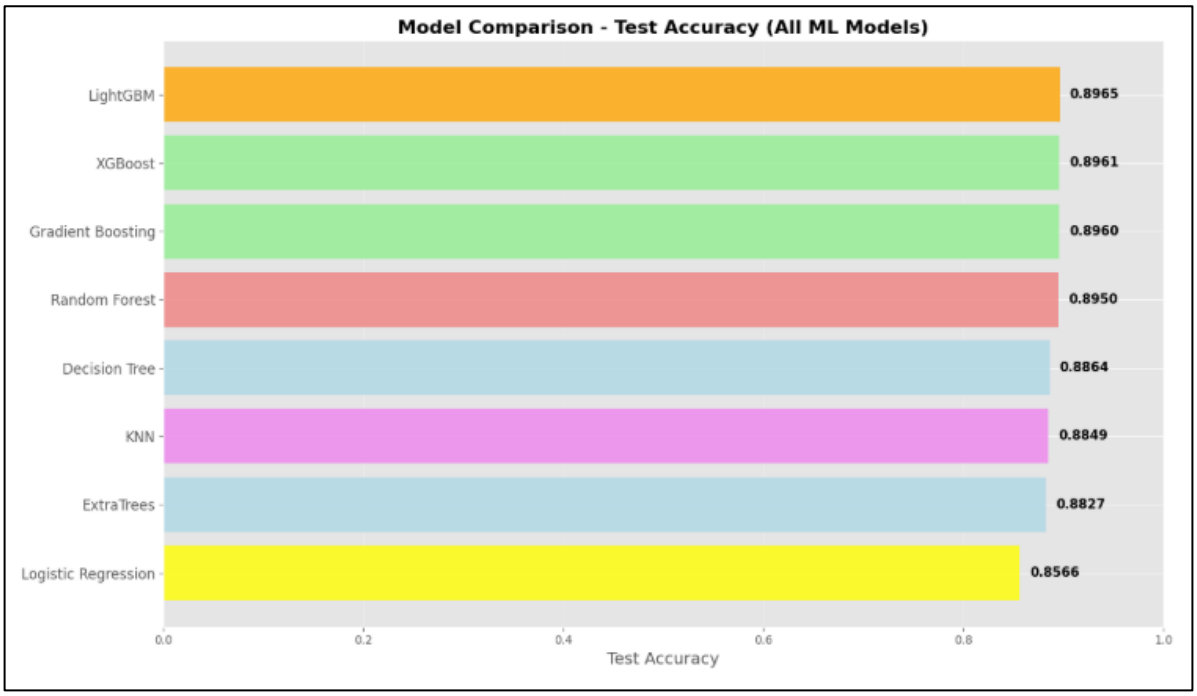


Рисунок Г.6 – Графік порівнянь методів

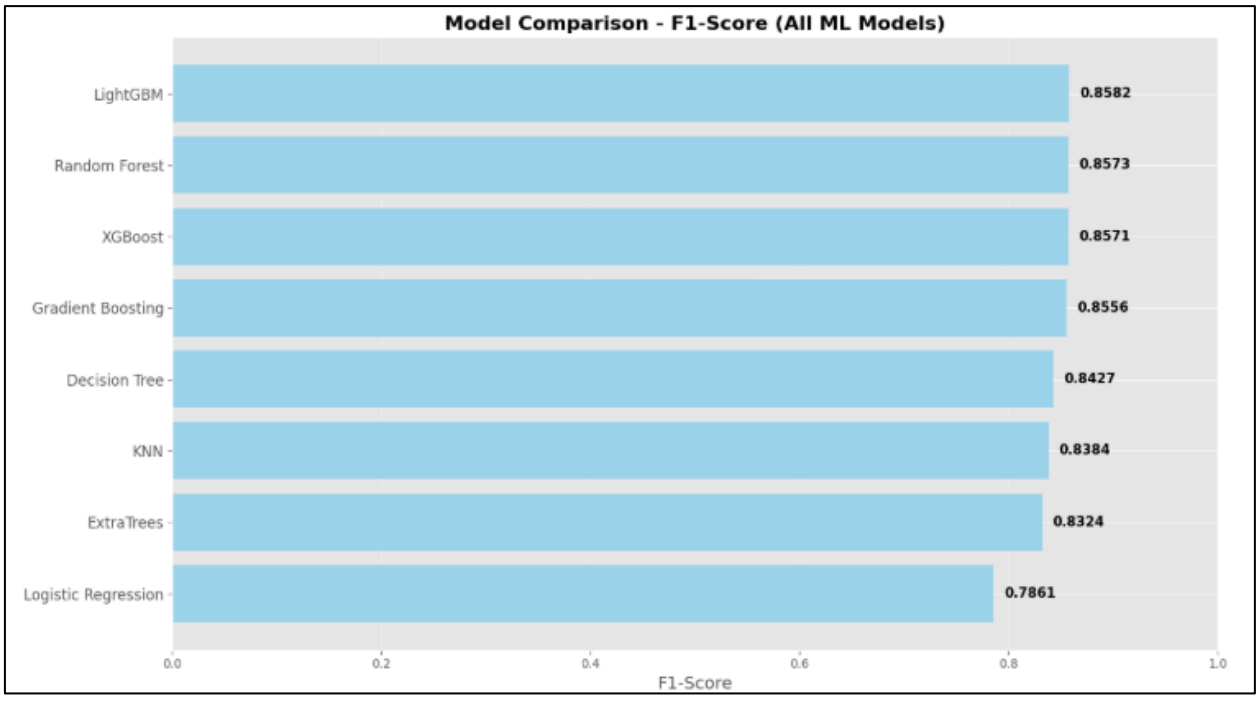


Рисунок Г.7 – Графік порівнянь методів за метрикою F1-Score