Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В. о. завідувача кафедри
кібербезпеки та захисту
інформації
_____Іван ПАРХОМЕНКО
«___» червня 2025 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітній ступень бакалавр

освітня програма Кібербезпека
(назва освітньо-професійної програми)

на тему: «Засіб виявлення фішингових листів за допомогою машинного навчання»

Виконавець: студент IV курсу, групи КБ-44 мс

_____ **Богдан ПРИХОДЬКО**
(підпис) (ім'я прізвище)

	Підпис	**Ім'я ПРІЗВИЩЕ**
Керівник		Яніна ШЕСТАК
Нормоконтроль		Іван БІЛОКОНЬ

Київ 2025

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В. о. завідувача кафедри
кібербезпеки
та захисту інформації
_____Іван ПАРХОМЕНКО
«29» листопада 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої-професійної програми)

Студенту **КБ-44 МС** **Приходьку Богдану Сергійовичу**
(група) (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Засіб виявлення фішингових листів за допомогою машинного навчання.

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №6 від 28.11.2024 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Методи виявлення фішингових листів із застосуванням сучасних алгоритмів машинного навчання.

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Необхідно ознайомитися з існуючими підходами до виявлення фішингових електронних листів, проаналізувати переваги застосування методів машинного навчання, здійснити вибір інструментів і реалізувати систему на основі

навчання моделі.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Розробка моделі машинного навчання для виявлення фішингових листів, яка може бути впроваджена в системи електронної пошти з метою підвищення рівня кібербезпеки користувачів.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 29 листопада 2024 року

Завдання видала _____ (підпис) Яніна ШЕСТАК (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Богдан ПРИХОДЬКО (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	29.11.2024 – 22.01.2025	*виконано*
2	Аналіз літератури	29.01.2025 – 11.02.2025	*виконано*
3	Обґрунтування вибору рішення	12.02.2025 – 15.02.2025	*виконано*
4	Обґрунтування вибору методів дослідження	16.02.2025 – 04.03.2025	*виконано*
5	Формування та підготовка датасету	05.03.2025 – 21.03.2025	*виконано*
6	Реалізація моделі виявлення фішингових листів	22.03.2025 – 08.04.2025	*виконано*
7	Навчання та оцінка моделі за ключовими метриками	09.04.2025 – 10.05.2025	*виконано*
8	Оформлення пояснювальної записки	11.05.2025 – 27.05.2025	*виконано*
9	Підготовка до захисту кваліфікаційної роботи	28.05.2025 – 13.06.2025	*виконано*

Завдання видала _____ (підпис) Яніна ШЕСТАК (ім'я, прізвище)

Завдання прийняв _____ Богдан ПРИХОДЬКО

до виконання _____ (підпис) _____ (ім’я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 13 червня 2025 року

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, має 60 сторінки основного тексту, 6 таблиць та 11 рисунки. Список використаних джерел включає 27 найменувань і займає 3 сторінки.

Метою роботи є розробка та впровадження методів машинного навчання для ефективного виявлення фішингових листів на основі аналізу їх текстового вмісту.

Для досягнення зазначеної мети поставлено наступні завдання:

- проаналізувати особливості фішингових атак в електронній пошті;
- ознайомитися з сучасними методами виявлення фішингу на основі машинного навчання;
- сформувати та обробити датасет фішингових і нефішингових листів;
- реалізувати модель BERT для класифікації фішингових листів;
- оцінити ефективність моделі та розробити додаток для її використання.

Об'єктом дослідження є процес виявлення фішингових листів у електронній пошті з використанням технологій машинного навчання.

Предметом дослідження є алгоритми та моделі машинного навчання, що застосовуються для класифікації електронних листів на фішингові та легітимні.

Практичною цінністю отриманих результатів є розроблений засіб виявлення фішингових листів за допомогою машинного навчання, котрий дозволяє:

- виявляти фішинг навіть у складних та нових формах;

- працювати незалежно від платформи;
- захищати користувачів менш популярних поштових систем.

Ключові слова: фішинг, машинне навчання, нейронні мережі, класифікація тексту, LSTM, BERT, виявлення загроз.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

AI	-	штучний інтелект (Artificial Intelligence)
ML	-	машинне навчання (Machine Learning)
NLP	-	обробка природної мови (Natural Language Processing)
BERT	-	Bidirectional Encoder Representations from Transformers
LSTM	-	Long Short-Term Memory
CNN	-	згорткова нейронна мережа (Convolutional Neural Network)
GRU	-	Gated Recurrent Unit
RNN	-	рекурентна нейронна мережа (Recurrent Neural Network)
API	-	програмний інтерфейс прикладного програмування (Application Programming Interface)
F1	-	score – показник збалансованої точності
TP / TN / FP / FN	-	істинно позитивні / істинно негативні / хибнопозитивні / хибнонегативні спрацювання
TF-IDF	-	частота терміна оберненої до частоти документа (Term Frequency-Inverse Document Frequency)
VS Code	-	Visual Studio Code
GPU	-	графічний процесор (Graphics Processing Unit)

ВСТУП

У сучасному цифровому світі електронна пошта залишається доволі актуальним засобом комунікації впрофесійній сфері, та не меншпопулярним середзвичайних користувачів мережі інтернет. Проте ця тенденція зумовлює стрімке зростання обсягів електронних листів, що спричиняє в свою чергу збільшення кількості фішингових атак, які стають більш замаскованими та складнішими для виявлення. Фішинг – це досить небезпечний метод кіберзлочинів, котрий застосовує методи соціальної інженерії, маніпулюючи неуважністю та довірою користувачів для викрадення конфіденційної інформації: логіни, паролі, банківські дані тощо.

Вони здатні маскуватися під легітимні повідомлення від рядових відомих компаній, державних установ чи банків, створюючи правдиве враження. Тому своєчасне виявлення фішингових листів, стає вкрай важливим завданням сьогодні.

Кваліфікаційна робота несе в собі розробку системи виявлення фішингових листів за допомогою методів машинного навчання. В рамках дослідження буде розглянуто теоретичні основи фішингових атак, проведено аналіз сучасних підходів до виявлення, а також розроблено прототип системи для автоматичного аналізу електронних листів із використанням навчання текстової моделі BERT.

Актуальність цієї кваліфікаційної роботи зумовлена стрімким зростанням фішингових атак, а також їх негативним впливом на інформаційну безпеку. Застосування саме методів машинного навчання дає більш потужний та надійний аналіз, у порівнян з традиційними методами фільтрації, які дедалі частіше зазнають невдач через нові форми атак.

Для досягнення цієї мети поставлено такі завдання:

- проаналізувати особливості фішингових атак в електронній пошті;

- ознайомитися з сучасними методами виявлення фішингу на основі машинного навчання;
- сформувати та обробити датасет фішингових і нефішингових листів;
- реалізувати модель BERT для класифікації фішингових листів;
- оцінити ефективність моделі та розробити додаток для її використання.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ЗАСАДИ КЛАСИФІКАЦІЇ ТА РОЗПІЗНАВАННЯ ФІШИНГОВИХ ЛИСТІВ

1.1 Визначення основних понять та класифікація фішингових листів

Фішингові атаки досить поширений вид шахрайства в інтернеті, він спрямований на обман користувачів та викрадення їх конфіденційної інформації. Даний вид атак набув популярність через простоту реалізації та високий рівень успішності. У цьому пункті буде розглянуто основі визначення фішингу, його класифікація, що дозволить краще розуміти даний вид шахрайства та методи які використовують злочинці для досягнення своєї мети.

Фішинг - це практика надсилання шахрайських повідомлень, які виглядають як такі, що надходять із авторитетного джерела, зазвичай через електронну пошту. [1]

Даний вид атак є великою загрозою, бо він використовує людей, а не технологічні вразливості. Зловмисникам не має необхідності проникати в систему шляхом злому чи обходити системи захисту, для цього вони використовують обман людей і можуть змусити їх видати свої гроші, конфіденційну інформацію та іншу важливі інформаційні ресурси.

Атаки проводяться на різні об’єкти: звичайні люди, великі корпорації та урядові установи. Нещодавній звіт компанії Check Point показав, що Microsoft і Google є найбільшими брендами, які піддаються підробці для здійснення фішингових атак. У першому кварталі 2024 року на Microsoft припало 38% усіх спроб фішингових атак на бренди, що робить її головною мішенню, за нею йде Google - 11%. Більшість цих атак, як правило, були пов'язані з начебто легітимними електронними листами, ретельно розробленими, щоб обманом

змусити одержувачів надати свої облікові дані для входу в систему або іншу конфіденційну інформацію. [2]

Розібравши основні поняття слід також звернути увагу на класифікацію фішингових листів. Це дасть змогу провести детальний аналіз специфічних методів та підходів, котрі використовують зловмисники для отримання бажаного результату.

Виділять такі види атак за допомогою фішингу:

1. Масовий фішинг електронної пошти (англ. Bulk email phishing). Під час даного виду атаки, зловмисники без розбору розсилають купу листів якомога більшій кількості людей, в надії на те, що частина з них потрапить до пастки.

Часто створюються електронні листи-підробки, які надходять від легальних компаній: банки, інтернет-магазини або розробники відомих додатків. Видаючи себе за інших, шахраї значно підвищують шанс того, що потенційна жертва є клієнтом даної компанії чи бренду. Чим частіше людина взаємодіє з брендом, тим більша ймовірність, що вона відкриє фішинговий лист, який нібито надійшов від легітимного відправника.

Кіберзлочинці роблять все можливе, щоб фішингові листи виглядали справжніми - це і використання логотипів та брендингу підробленого відправника, підробка адреси електронної пошти, щоб створити враження, ніби повідомлення надійшло з доменного імені підставного відправника. Вони можуть навіть скопіювати справжній електронний лист від підставного відправника та змінити його для зловмисних цілей.

У тексті листів, зазвичай, надходить прохання виконати щось просте, що не викликає застережень але в свою чергу призводить до розголошення конфіденційної інформації або завантаження шкідливого програмного забезпечення. Наприклад, фішингове посилання може виглядати так: «Натисніть тут, щоб оновити свій профіль». Коли жертва натискає на це шкідливе посилання, вона потрапляє на підроблений веб-сайт, який викрадає її облікові дані для входу в систему.

2. Фішинг зі списом (англ. Spear phishing) - це цілеспрямована фішингова атака на конкретну особу.[3] В переважній більшості випадків це людина яка має привілейовані права доступу до конфіденційних даних, які шахрай може використати в свої цілях, наприклад, фінансовий менеджер, який має доступ до переказів грошей з рахунків компанії.

Зловмисник вивчає свою жертву, та будує свій план атаки так, щоб зібрати якомога більше інформації про потенційну жертву, та отримати великий відсоток довіри від неї, він може видавати себе за друга, начальника чи постачальника. В результаті створюються повідомлення, котрі містять персональні дані, щоб не викликати підозр у людини, наприклад фішер може може видати себе за керівника жертви та надіслати електронний лист із текстом: «Знаю, що сьогодні ввечері ви вирушаєте у відпустку, але чи могли б ви оплатити цей рахунок до кінця робочого дня?». Фішинг-атака, спрямована на керівника вищої ланки, заможну особу або іншу цінну ціль, називається «китовий фішинг» або «китобійна атака».

3. Компрометація ділової електронної пошти (англ. Business email compromise, далі - BEC) - це клас фішингових атак, які намагаються викрасти гроші або цінну інформацію - наприклад, комерційну таємницю, дані клієнтів або фінансову інформацію - з бізнесу чи іншої організації.

BEC-атаки можуть мати кілька форм, але дві найбільш поширені це - шахрайство з боку керівника (англ. CEO fraud) та компрометація облікового запису електронної пошти (англ. Email account compromise).

Шахрайство з боку керівника працює наступним чином: злочинець видає себе за керівника вищої ланки, часто шляхом викрадення його електронної пошти, потім надсилає повідомлення працівнику нижчої ланки з інструкцією переказати кошти на шахрайський рахунок, здійснити покупку у нелегітимного продавця або надіслати файли неавторизованій особі.

Другий метод використовує дещо інший підхід: фішер компрометує обліковий запис електронної пошти співробітника нижчої ланки, для прикладу, фінансовий менеджер, менеджер з продажів або інженер з досліджень та

розробок. Маючи доступ до таких даних, зловмисник починає надсилати підроблені рахунки постачальникам, інструкції щодо здійснення фіктивних платежів або запити на доступ до конфіденційної інформації.

1.2 Ключові ідентифікатори фішингових повідомлень

Фішингові атаки стають дедалі складнішими, більш продуманими та різноманітними, тому потрапити в цю пастку звичайному користувачу дедалі легше, одже потрібно вміти розрізняти легітимні листи від зловмисних, для цього слід звернути увагу на ключові ідентифікатори фішингових листів. Ці ознаки включають:

1. Сильні емоції та тактика тиску. Фішинг-шахраї намагаються викликати у жертв відчуття терміновості, щоб вони діяли швидко та без роздумів. Зловмисники часто використовують такі методи, викликаючи у людей сильні емоції: страх, жадібність, зацікавленість. Також вони можуть встановлювати часові обмеження та погрожувати наслідками, для прикладу, тюремним ув’язненням. Нижче наведено приклади типових фраз у даного ідентифікатора:

1.1. "Існує проблема з вашим обліковим записом або фінансовою інформацією. Ви повинні негайно оновити її, щоб уникнути втрати доступу";

1.2. "Ми виявили незаконну діяльність. Сплатіть цей штраф негайно, інакше вас буде заарештовано";

1.3. «Ви виграли безкоштовний подарунок, але ви повинні отримати його прямо зараз»;

1.4. "Цей рахунок прострочений. Ви повинні сплатити його негайно, інакше ми відключимо вас від послуг";

1.5. "У нас є чудова інвестиційна можливість для вас. Покладіть гроші на депозит зараз, і ми гарантуємо неймовірні прибутки".

2. Запити на отримання грошей або конфіденційної інформації**.** Даний ідентифікатор вказує на прагнення зловмисника отримати одну з двох речей: гроші або конфіденційні дані. Неочікувані та несподівані прохання про оплату чи надання особистої інформації можуть вказувати на потенційну фішингову атаку. Для реалізації фішери використовують неправдиві посилання для оновлення платіжної інформації, зміни паролю чи підтвердження облікового запису користувача.

3. Погана орфографія та граматика. Багато зловмисних угрупувань діють на міжнародному рівні і не володіють достатнім рівнем знань з тієї чи іншої мови, тому багато фішингових повідомлень містять граматичні помилки та невідповідності, на котрі слід звернути увагу.

4. Загальні повідомлення. Легальні листи від дійсних відправників, зазвичай, містять конкретні деталі: ім’я клієнта, посилання на конкретне замовлення чи пояснення в чому саме причина звернення, тому нечітке повідомлення на кшталт: «З вашим акаунтом виникла проблема» без жодної додаткової інформації слід детальніше перевіряти.

5. Фальшиві URL-адреси та адреси електронної пошти. Зловмисники можуть використовувати замасковані URL-адреси та адреси електронної пошти, видаючи їх за легітимні. Вони змінюють окремі символи або додають непомітні елементи, що дозволяє підробити адресу таким чином, щоб вона виглядала праводоподібно. Також є поширеним використання субдоменів та сервісів по скороченняю посилань для приховування джерел шкідливого ресурсу.

6. Інші ознаки. Шахраї можуть надсилати файли та вкладення, які адресат не просив і не очікував отримати. Вони можуть використовувати зображення тексту замість власне тексту в повідомленнях і веб-сторінках, щоб уникнути спам-фільтрів.

1.3 Аналіз ефективності фішингових атак: статистичний огляд

Фішингові атаки залишаються одним із найпоширеніших та найефективніших методів кіберзлочинів для викрадення конфіденційної інформації. Вони можуть бути спрямовані як на окремих користувачів, так і на великі організації, використовуючи різноманітні тактики та технології для досягнення свої цілей. У цьому розділі буде розглянуто основні статистичні дані щодо фішингових атак, їх частоту, поширеність, а також ефективність впровадження заходів захисту.

Згідно зі звітами міжнародних дослідницьких організацій, кількість фішингових атак щороку зрозстає, що свідчить про збільшення активності кіберзлочинців у цій сфері. Відомо, що серед усіх кібезагроз фішинг займає значну частку, особливо у секторі електронної пошти. Дослідження показують, що понад 90% успішних кіберзломів розпочинаються саме з фішингового листа, який має на меті змусити користувача відкрити шкідливе посилання або надаи особисту інформацію. За останніми даними компанії IBM, фішинг складає понад 60% усіх кіберзагроз, що робить його основною загрозою для користувачів у мережі інтернет.[3]

Динаміка зростання фішингових атак за останні роки наведена в таблиці 1.1 нижче.

Таблиця 1.1

Динаміка зростання фішингових атак

Показник	**2021**	**2022**	**2023**	**2024**
Загальна кількість фішингових атак (млн)	2	3,1	4,2	5,5
Частка компаній, що зазнали атак типу Business Email Compromise (далі BEC) (%)	45%	52%	59%	64%

продовження таблиці 1.1

Середній фінансовий збиток від однієї BEC-атаки ($)	80 000	110 000	130 000	150 000
Частка фішингових сайтів з HTTPS (%)	65%	72%	78%	80%
Зростання атак з використанням AI (%)	+5%	+10%	+12%	+15%
Частка атак з використанням кількох каналів (%)	25%	30%	35%	40%
Найбільш імітовані бренди	Microsoft (25%)	Microsoft (28%)	Microsoft (30%)	Microsoft (32%)

Середній рівень успішності фішингових атак залежить від методу реалізації та рівня користувачів. Згідно з даними Symantec, цільові атаки на топ-менеджмент та співробітників фінансових установ мають рівень успішності до 30%.[4] У таблиці 1.2 наведено порівняльний аналіз основних типів фішингових атак.

Таблиця 1.2

Типи фішингових атак та їх ефективність

Тип атаки	Середній рівень успішності (%)	Середній збиток (тис. доларів)
Масовий фішинг	3	500
Спірфішниг	30	1200
Смішниг	15	750
Вішинг	10	900

Варто також зазначити, що зловмисники використовують новітні методи маскування та обфускації шкідливих посилань, що ускладнює їх виявлення.

Наприклад, застосування скорочених URL-адрес або підроблених доменів дозволяє маскувати справжнє джерело атаки, роблячи його менш очевидним для користувача. Згідно із статистичними даними, понад 70% фішингових листів містять шкідливі посилання, і лише 30% - вкладення зі шкідливим кодом. У звіті Verizon зазначається, що 43% всіх фішингових листів містять шкідливі вкладення у форматі .zip або .pdf.[5]

Ефективність фішингових атак також залежить від рівня підготовленості користувачів до виявлення таких загроз. Згідно з опитуваннями, проведеними у великих корпораціях, лише 25% працівників можуть правильно визначити нелегітимне повідомлення без попереднього навчання. Це вказує на необхідність впровадження освітніх програм та тренінгів з інформаційної безпеки для зниження ризику успішних атак. За даними дослідження компанії Proofpoint, серед організацій, які впровадили регулярні навчання, рівень успішних фішингових атак знизився на 47%.

Загалом, аналіз ефективності фішингових атак демонструє, що незважаючи на впровадження технічних рішень для їх виявлення, людський фактор залишається основною ланкою, яка підпадає під маніпуляції. Це вимагає комплексного підходу до запобіганню фішингу, включаючи, як технічні засоби захисту, так і освітні заходи для підвищення рівня знань користувачів.

1.4 Нормативно-правова база та відповідальність за фішингові атаки

Враховуючи високий рівень загроз фішнигових атак, котрі вони становлять для користувачів та компаній, питання правового регулювання є досить актуальним. В Україні діє комплекс законодавчих актів, які регулюють покарання за кіберзлочини, зокрема фішинг.

У цьому підрозділі буде викладено основні нормативно-правові акти України, що регулюють питання протидії фішинговим атакам, а також міжнародні стандарти та рекомендації, що сприяють посиленню заходів

кібербезпеки. Окремо буде проаналізовано пложення про відповідальність за вчинення фішингових атак, що передбачені законодавством України.

В Україні протидія фішинговим атакам регулюється законодавчими актами, які охоплюють інформаційну безпеку, захист персональних даних та відповідальність за кібезлочини.

Одним із головних нормативних документів є Закон України «Про захист інформації в інформаційно-комунікаційних системах», даний закон визначає головні принципи захисту інформації. В законі йдеться, що власники інформаційних систем зобов'язані вживати необхідних заходів щодо захисту інформації від несанкціонованого доступу (далі НД), включаючи фішингові атаки. Також, передбачено обов'язок надавати комлексний захист інформації, впроваджувати системи моніторингу та виявлення даного типу загроз. [6]

Закон України «Про електронну комерцію», регулює правові відносини у сфері електронної комерції та встановлює вимоги до ідентифікації сторін у мережі. Це актуально у випадках, коли фішингові атаки спрямовані на викрадення даних банківських карток або облікових записів користувачів. Відповідно до закону, учасники електронної комерції зобов'язані здійснювати відповідну автентифікацію користувачів та захист платіжних даних.[7]

Кримінальний кодекс України (далі ККУ) містить положення, що передбачають відповідальність за фішингові атаки, а саме стаття 190 «Шахрайство». Вона застосовується до осіб, які незаконно заволоділи чужими даними, шляхом обману або зловживанням довіри. [8] Стаття 361 «Несанкціоноване втручання в роботу інформаційних (автоматизованих), електронних комунікаційних, інформаційно-комунікаційних систем, електронних комунікаційних мереж», котра передбачає кримінальну відповідальність за НД до інформаційних ресурсів, у випадках, якщо наслідки фішингової атаки спричинили значні матеріальні збитки або відбувся витік конфіденційної інформації. [8]

Закон України «Про захист персональних даних», який визначає правила обробки, зберігання та передачі інформації між користувачами. Відповідно до

закону, організації, котрі здійснюють збір персональних даних, зобов’язані забезпечувати їх захист від НД, включаючи фішингові атаки. У разі порушення цих норм, компанії можуть бути притягнуті до відповідальності за недбале зберігання даних та сприяння кібезрлочинцям у їх викраденні. [9]

Також регуляція відбувається на рівні нормативних актів, які контролюють діяльність банків та інших фінансових організацій, що часто є об’єктами тих чи інших кібератак, зокрема фішинг. Національний банк України (далі НБУ) визначає положення, що включають в себе забезпечення інформаційної безпеки даних систем, а також вимоги до захисту персональних даних клієнтів та протидії фішингу. Наказ НБУ №217 встановлює вимоги до обробки та зберігання електронних даних, а також заходи для протидії шахрайству. [10]

Окрім вищезгаданих законів, в Україні діє багато підзаконних актів та нормативних документів, котрі ухвалені виконавчими органами влади. Вони визначають конкретні процедури, стандарти та вимоги до забезпечення кібербезпеки у різних сферах економіки та державного сектору.

Ключовим документом у забезпеченні безпеки в інформаційні сфері є Конвенція Ради Європи про кіберзлочинність. Цей міжнародний договір, ратифікований Україною, визначає основні принципи боротьби з кіберзлочинами, включаючи фішинг. Конвенція зобов’язує держави-учасниці криміналізувати дії, спрямовані на НД до інформації, розповсюдження шкідливого програмного забезпечення (далі ШПЗ) та викрадення особистих даних. [11]

Стандарти ISO/IEC 27001 та ISO/IEC 27002 також відіграють важливу роль у забезпеченні кібербезпеки. Вони встановлюють вимоги до системи управління інформаційною безпекою, включаючи політики щодо протидії фішинговим атакам. Зокрема, ці стандарти передбачають впровадження заходів для моніторингу електронних листів, автентифікації користувачів та захисту даних. [12]

Європейський союз (далі ЄС) також активно бореться з фішинговими атаками шляхом прийняття Директиви NIS (Network and Information System Directive). Цей документ зобов’язує держави-члени ЄС вживати заходи для захисту критичної інфраструктури від кіберзагроз. Директива визначає вимоги до операторів критичної інфраструктури щодо виявлення загроз, реагування на інциденти та захисту інформаційних систем. [13]

Крім того, ENISA (European Union Agency for Cybersecurity) надає рекомендації щодо виявлення та запобігання фішинговим атакам. ENISA публікує аналітичні звіти та керівництва для організацій щодо захисту від фішингу, включаючи рекомендації з моніторингу підозрілої активності та навчання персоналу для підвищення обізнаності про кіберзагрози.

Таким чином, міжнародні стандарти та керівництва з кібербезпеки є важливими інструментами в боротьбі з фішинговими атаками. Вони забезпечують єдиний підхід до виявлення та запобігання загрозам і сприяють формуванню глобальної системи кіберзахисту.

Фішингові атаки є серйозним правопорушенням, за яке передбачена відповідальність на декількох рівнях - цивільно-правова, адміністративна та кримінальна. Такий підхід забезпечує комплексний захист від шахрайських дій та дозволяє ефективно реагувати на порушення прав користувачів.

Цивільно-правова відповідальність за фішингові атаки передбачає відшкодування потерпілим завданих збитків. Це може бути компенсація за втрату конфіденційних даних, крадіжку персональних даних або фінансові втрати, спричинені шахрайськими діями. Наприклад, якщо в результаті фішингової атаки було викрадено персональні дані або реквізити банківської картки, жертва може вимагати компенсації через суд. Крім того, цивільне судочинство може бути пов'язане з порушенням авторських прав, якщо шахраї незаконно використовують матеріали, захищені авторським правом, для створення фішингових сайтів або електронних листів.

Адміністративна відповідальність передбачає накладення штрафів або інших покарань за недотримання вимог кібербезпеки. Вона застосовується до

осіб та організацій, які не вживають необхідних заходів для захисту інформації або не повідомляють про кіберінциденти. Наприклад, компанія, яка зазнала витоку даних внаслідок фішингової атаки, може бути оштрафована за незбереження конфіденційної інформації. Ці заходи спрямовані на запобігання кіберзагрозам та заохочення організацій до впровадження систем захисту від фішингових атак.

Кримінальна відповідальність є найсуворішою формою покарання за фішингові атаки. Вона застосовується у випадках, коли фішингові дії завдали значної шкоди, призвели до крадіжки великої кількості персональних даних або порушили роботу критично важливих об'єктів інфраструктури. Залежно від тяжкості злочину, покарання може включати значні штрафи, обмеження волі або тривалі терміни ув'язнення. Наприклад, особи, які організовують масштабні фішингові кампанії, спрямовані на фінансові установи або державні органи, можуть бути притягнуті до відповідальності за шахрайство з використанням електронних засобів зв'язку.

На міжнародному рівні фішингові атаки також вважаються серйозним злочином, який вимагає узгоджених дій між країнами. Міжнародні угоди передбачають механізми співпраці між правоохоронними органами різних країн, обмін інформацією про кіберзлочинців та екстрадицію осіб, відповідальних за фішингові атаки. Така співпраця дозволяє швидко реагувати на фішингові кампанії, які часто здійснюються з території інших країн з використанням анонімних серверів або бот-мереж.

Таким чином, відповідальність за фішингові атаки - це багаторівнева система, яка включає в себе як відшкодування збитків потерпілим, так і покарання зловмисників за їхні шахрайські дії. Комплексний підхід до цього питання забезпечує ефективний захист від фішингових атак, водночас стимулюючи організації до підвищення рівня кібербезпеки та обізнаності щодо загроз у цифровому середовищі.

Висновки за розділом 1

В першому розділі було проведено теоретичний аналіз основ, а саме класифікація та розпізнавання фішингових листів. Розглянуто визначення фішингу як методу шахрайства, який базується на соціальній інженерії та спрямований на викрадення конфіденційної інформації, спираючись на обман користувачів. Відповідно до статистики було виявлено, що фішингові атаки є одним із найпоширеніших кіберзлочинів, оскільки вони не вимагають складних технічних дій та в свою чергу досить високоефективні.

Також, було проведено класифікацію атак такого роду, а саме визначено, що таке масовий фішинг, спірфішинг, компрометацію ділової електронної пошти та інші види атак, які відрізняються рівнем персоналізації, об’єктом цільової атаки та методами маскування шахрайських листів, що дало змогу зрозуміти основні підходи шахраїв до маніпуляції користувачами. Приділено увагу аналізу тактик та методів, які використовуються для успішної атаки, а саме підроблення адрес електронної пошти, маніпуляції з URL-адресами, використання ключових слів на, які слід звертати увагу та застосування соціальної інженерії.

Також, проведено аналіз ефективності фішингових атак, їх динаміку за останні роки, рівень успішності та деякі сценарії реалізації шахрайських схем.

Отже, у першому розділі було систематизовано основні теоретичні аспекти, що стосуються фішингових атак, їх класифікації, методів реалізації та індикаторів розпізнавання. Це створює базу для подальшого аналізу заходів протидії фішинговим атакам та розробки рекомендацій щодо підвищення.

РОЗДІЛ 2
ОГЛЯД ТА АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ВИЯВЛЕННЯ ФІШИНГОВИХ ЛИСТІВ

2.1 Сучасні технологічні рішення для захисту від фішингу

Фішинг є поширеною формою кіберзлочинності, спрямованої на отримання конфіденційної інформації шляхом обману користувачів. Беручи до уваги швидкий розвиток цифрових технологій, даний вид атак стає складнішим. Для ефективної боротьби потрібно впровадження сучасних технологічних рішень, які забезпечують надійний захист.

До основних технологій, які використовують для захисту електронної пошти від фішингових атак можна віднести Sender Policy Framework (далі SPF), Domain Keys Identified Mail (далі DKIM) та Domain-based Message Authentication, Reporting & Conformance (далі DMARC). SPF надає можливість власникам доменів вказувати, які сервери мають право надсилати електронні листи від їхнього імені, що ускладнює використання підроблених адрес для фішингових атак. DKIM даний підхід використовує цифровий підпис (далі ЦП) електронних листів, дозволяючи отримувачам перевірити, чи не було змінено вміст листа під час передачі. DMARC об’єднує в собі два попередні методи захисту, надаючи можливість користувачам встановлювати власні політики оброки електронних листів, які не пройшли перевірку, а також є можливість отримувати звіти про виявлені листи такої категорії. Підсумовуючи, використання вищеописаних технологій значно знижує успіх фішингових атак, які спрямовані саме на підробку електронних листів.

Багатофакторна аутентифікація, також, поширений та надійний інструмент захисту від фішингових атак. Використання даної технології передбачає кілька етапів аутентифікації користувача, таких як пароль,

біометричні дані або одноразовий код. Ці фактори ускладнюють для зловмисників доступ, навіть у випадках, коли ті змогли викрасти дані. Спираючись на дослідження, використання даного методу захисту може значно знизити ризики НД до облікових записів користувачів. [14]

Одним із сучасних методів боротьби з фішингом є використання штучного інтелекту (далі ШІ) та машинного навчання, його використовують для виявлення та блокування підозрілих активностей. Добре навчені моделі можуть гарно аналізувати текст електронних листів, виявляти ознаки фішингу, такі як терміновість, запити надати конфіденційну інформацію або прохання перейти за невідомими посиланнями. Впровадження ШІ дозволяє автоматизувати та значно облегшити виявлення фішингових повідомлень, а також дає можливість зручно фільтрувати нелегітимні повідомлення, що значно підвищує рівень безпеки користувачів. Функціонал ШІ досить великий, його також можна впровадити для аналізу веб-сайтів, виявляючи фішинг за допомогою аналізу стурктури URL, наповненості сторінок та інших важливих характеристик. Це дає змогу блокувати шкідливі сайти ще до того як користувач введе свої дані.

Технології захисту також впроваджують безпосередньо до клієнту пошти, для прикладу системи фільтрації електронної пошти використовують свої підходи до виявлення фішингу, аналізуючи заголовки, тіло листа та додаткові вкладення чи посилання. Постійне вдосконалення з використанням ШІ адаптує дану систему захисту до нових загроз. Крім того, можна використовувати пісочниці, які надають можливість безпечно перевіряти підозрілі вкладення чи посилання у ізольованому середовищі та проводити детальний аналіз для подальшого вдосконалення систем захисту. Це допомагає виявляти ШПЗ або фішинг завчасно, не піддаючи систему небезпеці. [15]

Ще два сучасних методи захисту – системи управління інформаційною безпекою (SIEM) та автоматизовані платформи реагування на інциденти (SOAR). Їх використання дає можливість виявляти фішинг в реальному часі. Принцип роботи полягає в аналізі журналів подій, виявленню аномалій та автоматичному реагуванню на загрози, що в свою чергу створює додатковий

рівень захисту забезпечуючи моніторинг та своєчасне реагування на потенційні загрози.

Незважаючи на вищезгадані системи протидії, людський фактор залишається вагомим слабким місцем в ланці захисту. Тому важливо забезпечувати проведення систематичного навчання користувачів, щодо розпізнавання фішингових атак та правильних дій у випадку їх виявлення. Аналізуючи дослідження, можна зробити висновки, що після проведення такого роду навчань, рівень успішних атак може значно знижуватись.

Таким чином, сучасні технологічні рішення для захисту від фішингу включать комбінацію технічних заходів, таких як SPF, DKIM, DMARC, багатофакторну автентифікацію, використання ШІ та машинного навчання, а також організаційні заходи, такі як навчання користувачів та впровадження систем моніторингу. Комплексний підхід до захисту дозволяє значно знизити ризики, пов’язані з фішинговими атаками, та забезпечити безпеку інформаційних систем.

2.2 Методи машинного навчання у виявленні фішингових листів

Останнім часом виявленню фішингових листів приділяється багато уваги, через їх значний вплив на безпеку користувачів. Тому для вирішення цієї проблеми було розроблено не мало методів по виявленню фішингових листів, починаючи від комунікаційно-орієнтованих методів, таких як протоколи автентифікації, розподіл за білими та чорними списками, до методів фільтрації на основі вмісту. Проте на практиці методи білих та чорних списків не довели своєї достатньої ефективності, тому їх широке використання було знижене з часом. Тим часом, фішингові фільтри на основі вмісту навпаки показали гарні результати та довели свою високу ефективність, тому дослідження зосередились на механізмах виявлення на основі контенту, а також на розвитку

машинного навчання та інтелектуального аналізу даних беручи за основу тіло листа та заголовки.

Беручи до уваги обмежену ефективність вищезгаданих традиційних методів виявлення фішингу, дослідження все частіше зміщують свій фокус на підходи, які використовують машинне навчання. Це дає змогу працювати не тільки з вже відомими шаблонами, а також постійно адаптуватись під нові, раніше незафіксовані загрози. Далі розглянемо ключові методи машинного навчання, що використовуються для виявлення фішингових листів.

Логістична регресія (англ. Logistic regression) є популярним методом машинного навчання, що використовується для задач бінарної класифікації, де вона моделює ймовірність належності результату до одного з двох класів. Даний метод часто використовується в задачах машинного навчання, що включають бінарну класифікацію, де метою є розподіл даних на одну з двох груп на основі набору характеристик. Беручи за основу надані незалежні змінні модель логістичної регресії використовує сигмоїдну або логістичну функцію для прогнозування ймовірності того, що залежна змінна буде дорівнювати 1. Сигмоїдна функція в свою чергу відображає реальні значення вхідних даних у діапазон від 0 до 1, представляючи ймовірність. Алгоритм логістичної регресії використовує оцінку максимальної правдоподібності для оцінки коефіцієнтів незалежних змінних. Ці коефіцієнти потім використовуються для обчислення ймовірності того, що залежна змінна дорівнює 1 на основі вхідних даних. На практиці логістичну регресію можна застосовувати в різних сферах, таких як кредитний скоринг, діагностика захворювань і виявлення шахрайства. Завдяки своїй простоті, інтерпретованості та стійкості, вона є дуже популярним методом. [16]

Деревоподібна модель (англ. *Decision Trees*) зазвичай використовується в машинному навчанні для задач, пов'язаних з регресією та класифікацією. Вона візуально зображує варіанти прийняття рішень на основі обставин та їх результатів. Структура складається з вузлів, гілок і листя. Вузли представляють тести на вхідних даних, гілки - можливі результати, а листя - остаточні рішення

або класифікації. Алгоритм будує дерево шляхом рекурсивного розбиття даних на підмножини на основі значень вхідних ознак. Універсальність дерев рішень може бути корисною як для багатокласової, так і для бінарної категоризації. Метою є розробка моделі, яка прогнозує цільову змінну шляхом послідовного прийняття рішень на основі вхідних ознак. Алгоритм вивчає оптимальні правила прийняття рішень на основі навчальних даних, щоб мінімізувати помилку класифікації та підвищити точність прогнозування. Слід зазначити, що дерева рішень можна інтерпретувати, оскільки вони пропонують чіткий та інтуїтивно зрозумілий спосіб візуалізації процесу прийняття рішень. Це полегшує розуміння факторів, які впливають на остаточне рішення або результат. Крім того, дерева рішень здатні обробляти як безперервні, так і категоріальні вхідні дані, і вони демонструють стійкість до пропущених значень і викидів. Однак дерева рішень можуть бути чутливими до надмірної адаптації, що призводить до потенційних проблем з узагальненням нових, непередбачуваних даних. [17]

Random Forest (англ. *Random Forest*) - це ансамблева техніка навчання, яка складається з дерев рішень, що навчаються незалежно на рандомізованих підмножинах навчальних даних та вхідних ознак. Результати отримують шляхом агрегування виходів дерев, як правило, шляхом усереднення або більшості голосів. Такий підхід зменшує надмірне пристосування і може призвести до покращення точності прогнозування та надійності роботи моделі. Концепція, що лежить в основі Random Forest, полягає у вирішенні проблеми перенавчання та підвищенні точності моделі шляхом об'єднання декількох дерев рішень. Унікальна підмножина вхідних характеристик і навчальних даних використовується для навчання кожного дерева в лісі, тим самим зменшуючи дисперсію моделі та покращуючи її узагальнюючі характеристики. Об'єднуючи результати цих окремих дерев, як правило, за допомогою усереднення або більшості голосів, підхід Random Forest зменшує надмірне пристосування і дає більш надійну і точну модель прогнозування або класифікації. Алгоритм Random Forest працює шляхом вибору випадкової підмножини навчальних

даних і вхідних ознак у кожному вузлі кожного дерева. Потім він будує дерево рішень на основі вибраних даних та ознак. Ця процедура повторюється ітеративно кілька разів, в результаті чого створюється колекція або «ліс» дерев рішень. На етапі прогнозування Random Forest об'єднує результати всіх дерев шляхом усереднення або прийняття рішення більшістю голосів, отримуючи остаточний прогноз або класифікацію. Точність і стійкість моделі покращуються завдяки цій ансамблевій техніці, що використовує колективну здатність до прийняття рішень декількох дерев. [16]

Адаптивне підсилення (англ. *Adaptive Boosting*), також відоме як AdaBoost це популярна техніка ансамблевого навчання, яка об'єднує слабкі класифікатори для створення більш надійного і точного загального класифікатора. AdaBoost ітеративно об'єднує прогнози декількох слабких класифікаторів для створення сильнішого класифікатора, адаптивно підлаштовуючи ваги навчальних вибірок, щоб надати більшого значення неправильно класифікованим зразкам. Такий підхід підвищує продуктивність класифікатора, роблячи його здатним обробляти складні шаблони даних і досягати вищої точності порівняно з окремими слабкими класифікаторами. AdaBoost особливо ефективний при роботі зі складними наборами даних, що містять багато вхідних ознак і класів. [17]

Навчальні дані розбиваються на різні підмножини для кожного слабкого класифікатора, а ваги навчальних вибірок динамічно коригуються під час кожної ітерації, щоб надати більшої важливості зразкам, які були неправильно класифіковані попередніми слабкими класифікаторами. Цей процес створює нову навчальну вибірку, зміщену в бік зразків, які раніше були неправильно класифіковані, що змушує слабкі класифікатори зосередитися на цих зразках і покращити свою роботу. Кінцевим результатом роботи алгоритму є зважена сума прогнозів усіх слабких класифікаторів, причому ваги визначаються точністю кожного слабкого класифікатора. AdaBoost має кілька переваг над іншими ансамблевими методами, такими як випадкові ліси та мішки. Він менш схильний до перенавчання, добре працює з даними високої розмірності і може

обробляти зашумлені та неповні дані. Одним з основних обмежень AdaBoost є його чутливість до пропусків і шуму в навчальних даних. Якщо дані містять багато пропусків або зашумлених вибірок, алгоритм може надмірно пристосуватися до цих вибірок і погано працювати на нових даних. Крім того, AdaBoost може бути дорогим в обчислювальному плані, оскільки вимагає навчання багатьох слабких класифікаторів на багатьох підмножинах навчальних даних.

Екстремальний градієнтний бустинг (англ. *Extreme Gradient Boosting*) - це високоефективний алгоритм машинного навчання, який використовується для задач класифікації. Як метод ансамблевого навчання, він поєднує прогнози від декількох слабких моделей, часто у вигляді дерев рішень, для отримання більш точного та стійкого остаточного прогнозу. Відомий своєю високою ефективністю, швидкістю та здатністю обробляти великі набори даних, XGBoost набув популярності в галузі машинного навчання. Цей процес триває до тих пір, поки не буде досягнута необхідна точність за певну кількість ітерацій. XGBoost також включає кілька методів регуляризації, таких як L1 (Lasso) та L2 (Ridge) регуляризація, щоб запобігти надмірному пристосуванню та покращити продуктивність узагальнення моделі. Ці методи регуляризації карають великі ваги або складні моделі, тим самим сприяючи створенню простіших і стабільніших моделей. Після того, як ансамбль дерев побудовано, прогнози робляться шляхом агрегування прогнозів усіх дерев. Зазвичай XGBoost використовує комбінацію зваженого голосування або усереднення для отримання остаточних прогнозованих ймовірностей класів. Цей аналіз допомагає визначити найбільш релевантні ознаки, які сприяють точному прогнозуванню. Як результат, XGBoost здобув популярність у сфері машинного навчання завдяки своїм унікальним можливостям у вирішенні цих поширених проблем.[18]

Щоб класифікувати набір даних фішингових сайтів, кожен з цих алгоритмів може бути навчений і оцінений за допомогою навчальних і тестових наборів, створених раніше. Для вимірювання продуктивності кожного

алгоритму можна використовувати такі показники, як частота згадувань, точність і достовірність. На основі результатів можна вибрати найбільш точний і ефективний алгоритм для розгортання в реальних сценаріях.

2.3 Використання глибоких нейронних мереж для аналізу тексту

Глибокі нейронні мережі набувають своєї популярності, бо вони є досить потужним інструментом для аналізу тексту, котрі здатні виявляти приховані патерни та складні контексті зв’язки між словами. Також їхньою перевагою є можливість працювати з великими обсягами даних, враховуючи семантичні особливості тексту, що дозволяє не лише розпізнавати примітивні фішингові шаблони, але й детектити нові типи атак, які раніше не зустрічалися. Зокрема, використання глибоких нейронних мереж дозволяє аналізувати структуру повідомлень, їх емоційни тон та наявність нетипових формулювань, що є характерним для фішингових листів.

Порівнюючи з базовими методами машинного навчання, глибокі нейронні мережі мають значну перевагу завдяки своєму функціоналу, а саме можливості аналізувати великі обсяги даних та знаходити складі зв’язки між різними елементами тексту. У той час як традиційні методи використовують прості статистичні показники або попередньо визначені правила, що на практиці не досить ефективно працює з новими підходами зловмисників до фішингових атак. Ще однією перевагою є здатність працювати саме з контекстом завдяки багатошаровій архітектурі та здатності до зберігання інформації у внутрішніх станах, вони можуть враховувати залежності між словами та фразами, що розташовані на різних відстанях одне від одного у тексті. Це дозволяє моделі не тільки розпізнавати ключові слова, але й аналізувати їх контекстуальне значення, визначаючи потенційно небезпечні конструкції. Таким чином, глибокі нейронні мережі забезпечують більш точний та комплексний аналіз тексту,

порівняно з базовими методами машинного навчання. Далі розглянемо існуючі глибокі нейронні мережі для роботи з текстом та їх функціонал.

Long Short-Term Memory (далі LSTM) є покращеною версією рекурентної нейронної мережі (RNN), вони здатні ефективно зберігати довготривалі залежності у послідовних даних, що робить їх особливо придатними для завдань, таких як машинний переклад, розпізнавання мовлення та прогнозування часових рядів.

На відміну від традиційних RNNs, які використовують один прихований стан, що проходить через час, LSTM вводять комірку пам'яті, яка зберігає інформацію протягом тривалих періодів, вирішуючи проблему вивчення довготривалих залежностей.

Архітектура LSTM включає комірку пам'яті, яка контролюється трьома блоками: вхідним блоком, блоком забування та вихідним блоком. Ці блоки вирішують, яку інформацію додавати в комірку пам'яті, видаляти з неї та виводити з неї. [19]

Вхідний блок контролює, яка інформація додається до осередку пам’яті. блок забування визначає, які дані необхідно видалити з пам’яті, тоді як вихідний блок регулює, яка інформація передається на вихід. Завдяки цій структурі LSTM-мережі можуть вибірково зберігати або відкидати дані під час їх проходження через мережу, що дозволяє їм ефективно вивчати довготривалі залежності. Крім того, мережа має прихований стан, що виконує роль короткочасної пам’яті. Ця пам'ять оновлюється за допомогою поточного входу, попереднього прихованого стану і поточного стану комірки пам'яті. [20] Структура LSTM моделі зображена на рисунку 2.1 нижче.

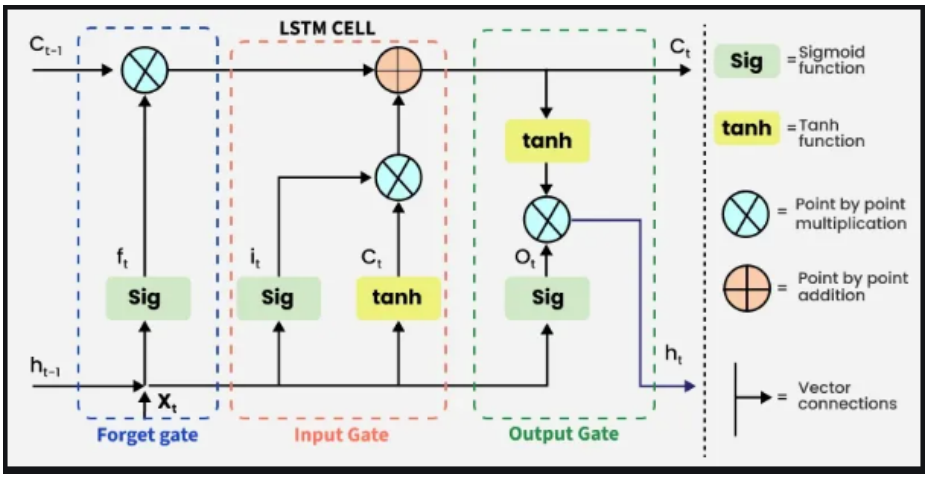


Рисунок 2.1 - Структура LSTM моделі

Непотрібна інформація про стан комірки видаляється за допомогою блоку забування. Два входи xt (вхід у заданий момент часу) та ht-1 (попередній вихід комірки) подаються на блок і множаться на матриці ваг, потім додається зміщення. Результат передається до функції активації, яка генерує двійковий вихід. Якщо вихід для заданого стану комірки дорівнює 0, інформація забувається; якщо вихід дорівнює 1, вона зберігається для подальшого використання.

Додавання корисної інформації до стану комірки виконується за допомогою блоку входу. Спочатку інформація підбирається за допомогою сигмоїдної функції, а значення, які потрібно запам'ятати, фільтруються, як і для блоку забування, використовуючи входи ht-1 та xt. Далі, за допомогою функції tanh створюється вектор, який генерує вихід від -1 до +1, що містить усі можливі значення ht-1 та xt. Нарешті, значення вектора та підібрані значення множаться для отримання корисної інформації.

Блок виходу витягує корисну інформацію з поточного стану комірки, щоб представити його як вихід. Спочатку вектор генерується шляхом застосування функції тангенса до комірки. Потім інформація підганяється за допомогою сигмоподібної функції та фільтрується, щоб зберегти значення, які будуть збережені за допомогою вхідних даних. Нарешті, значення вектора та підібрані

значення множаться та надсилаються як вихідні та вхідні дані до наступної комірки. [19] Переваги та недоліки LSTM моделі наведені в таблиці 2.1 нижче.

Таблиця 2.1

Преваги та недоліки LSTM моделі

Преваги	**Недоліки**
Здатність зберігати довгострокові залежності у тексті	Високі обчислювальні витрати та потреба у великих обсягах пам'яті
Ефективність у роботі з послідовними даними (наприклад, текст)	Можлива проблема перенавчання при обробці великих датасетів
Можливість працювати з нерегулярними послідовностями даних	Час навчання значно збільшується зі зростанням розміру тексту
Стійкість до проблеми зникнення градієнта завдяки механізму пам'яті	Складність у налаштуванні гіперпараметрів (кількість шарів, кількість нейронів)
Здатність зберігати довгострокові залежності у тексті	Високі обчислювальні витрати та потреба у великих обсягах пам'яті

Recurrent neural network (далі RNN) - це багатошарова нейронна мережа, розроблена для обробки послідовних або часових рядів даних з метою створення моделі машинного навчання (ML), яка здатна робити прогнози або висновки на основі послідовних вхідних даних.

Подібно до класичних нейронних мереж, таких як мережі з прямим зв'язком та згорткові нейронні мережі (CNN), рекурентні нейронні мережі також використовують навчальні дані для свого навчання. Відмінність полягає у їхній здатності зберігати «пам'ять», оскільки вони враховують інформацію з попередніх вхідних даних для впливу на поточні вхідні та вихідні значення.

На відміну від традиційних мереж глибокого навчання, які передбачають незалежність вхідних та вихідних даних, рекурентні нейронні мережі формують вихідні дані з урахуванням попередніх елементів у послідовності. Хоча інформація про майбутні події також могла б бути корисною для визначення виходу, односпрямовані рекурентні нейронні мережі не здатні враховувати ці події у своїх прогнозах.

Ще однією характерною особливістю рекурентних нейронних мереж є використання спільних параметрів на кожному шарі мережі. У той час як мережі з прямим зв’язком мають окремі ваги для кожного вузла, рекурентні нейронні мережі використовують однакові ваги на кожному шарі. Водночас, ці ваги все одно коригуються за допомогою методів зворотного поширення та градієнтного спуску, що сприяє ефективному навчанню мережі. [22] Схема рекурентної нейронної мережі зображена на рисунку 2.2 нижче.

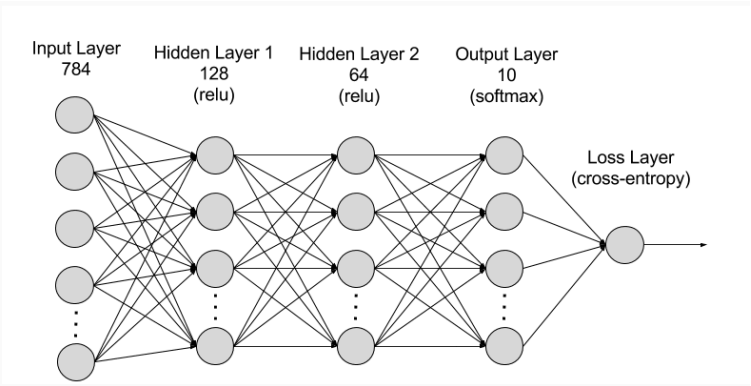


Рисунок 2.2 - Схема рекурентної нейронної мережі RNN

RNN складається з нейронів: вузлів обробки даних, які працюють разом для виконання складних завдань. Нейрони організовані як вхідний, вихідний та прихований шари. Вхідний шар отримує інформацію для обробки, а вихідний шар надає результат. Обробка, аналіз та прогнозування даних відбуваються в прихованому шарі. Вони працюють, передаючи послідовні дані, які вони отримують, до прихованих шарів покроково. Однак вони також мають самоциклічний або рекурентний робочий процес: прихований шар може запам'ятовувати та використовувати попередні вхідні дані для майбутніх прогнозів у компоненті короткочасної пам'яті. Він використовує поточні вхідні дані та збережену пам'ять для прогнозування наступної послідовності. [21] Переваги та недоліки RNN моделі наведені в таблиці 2.2 нижче.

Таблиця 2.2

Преваги та недоліки RNN моделі

Переваги	**Недоліки**
RNN здатні враховувати контекст попередніх даних, що робить їх ефективними для обробки послідовностей, таких як текст, аудіо чи відео.	Під час навчання зворотне поширення градієнтів може призвести до їх швидкого зменшення або збільшення, що ускладнює навчання довгих послідовностей.
Мережа може використовувати інформацію з попередніх кроків, що дозволяє враховувати історію даних.	Навчання RNN є повільним, оскільки кожен крок навчання залежить від попереднього, що потребує послідовної обробки.
Використання однакових ваг на кожному кроці зменшує кількість параметрів у порівнянні з класичними нейронними мережами.	Стандартні односпрямовані RNN не можуть враховувати майбутні елементи послідовності, що може обмежувати точність прогнозів.
RNN можуть обробляти послідовності різної довжини без необхідності жорсткої фіксації розміру входу.	Через складність архітектури та велику кількість параметрів RNN можуть перенавчатися на навчальних даних.
Використовуються у розпізнаванні мови, перекладі тексту, аналізі часових рядів тощо.	Навчання RNN є ресурсномістким через необхідність обробки кожного кроку послідовності окремо.

Bidirectional Encoder Representations from Transformers (далі BERT) - це модель мови глибокого навчання, розроблена для підвищення ефективності завдань обробки природної мови (NLP). Вона відома своєю здатністю враховувати контекст, аналізуючи зв'язки між словами в реченні двонаправлено. [23] Послідовність токенів подається до кодера Transformer. Ці токени спочатку вбудовуються у вектори, а потім обробляються в нейронній мережі. Вихідним результатом є послідовність векторів, кожен з яких відповідає вхідному токену, що забезпечує контекстуалізовані представлення.[24]

Архітектура моделі BERT складається з двох основних компонентів: кодувальника (англ. Encoder) та трансформера (англ. Transformer). Це дозволяє моделі аналізувати текст з обох напрямків, що забезпечує краще розуміння

контексту. Розглянемо детальніше складові цієї архітектури. Кодувальники - це компоненти нейронних мереж, які перетворюють вхідні дані на уявлення, що легше обробляти алгоритмам машинного навчання. Коли кодувальник аналізує вхідний текст, він генерує прихований стан тобто вектор. Приховані стани - це списки значень та внутрішніх параметрів, які надають додатковий контекст. Це упаковане представлення інформації потім передається до трансформера. Трансформер використовує отриману інформацію для виявлення шаблонів або побудови прогнозів, це архітектура глибокого навчання, яка перетворює вхідні дані на інший тип вихідних даних. [23]

BERT також покладається на механізм самоуваги, який фіксує і розуміє зв'язки між словами в реченні. Це завдяки двонаправленим трансформерам в центрі конструкції BERT. Це важливо, тому що часто слово може змінювати своє значення під час розвитку речення. Кожне додане слово розширює загальне значення слова, на якому фокусується алгоритм NLP. Чим більше слів присутні в кожному реченні або фразі, тим більш неоднозначним стає слово, на якому фокусується алгоритм. BERT враховує розширене значення шляхом двонаправленого читання, враховуючи вплив усіх інших слів у реченні на слово у фокусі та усуваючи імпульс зліва направо, який схиляє слова до певного значення в міру просування речення.[25] Приклад роботи моделі BERT зображено на рисунку 2.3 нижче.

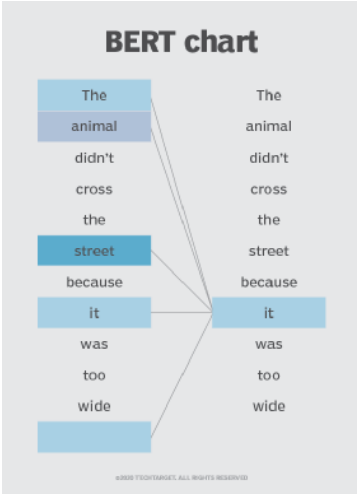


Рисунок 2.3 - Приклад роботи моделі BERT з текстом

Наприклад, на зображенні вище BERT визначає, до якого попереднього слова в реченні відноситься слово «it», а потім, використовуючи механізм самоуваги, зважує варіанти. Слово з найвищим підрахованим балом вважається правильною асоціацією. У цьому прикладі слово «it» стосується «animal», а не «street». Якби ця фраза була пошуковим запитом, результати відображали б це більш тонке і точне розуміння, якого досяг BERT. Переваги та недоліки моделі BERT наведені в таблиці 2.3 нижче.

Таблиця 2.3

Переваги та недоліки моделі BERT

Преваги	**Недоліки**
Можливість аналізувати контекст як зліва направо, так і справа наліво, що дозволяє краще розуміти значення слів у контексті.	Через велику кількість параметрів і складну архітектуру, BERT вимагає потужних апаратних ресурсів для навчання та інференсу.
Використання MLM для передбачення маскованих слів у реченні, що підвищує точність розуміння контексту.	Процес навчання займає багато часу навіть при використанні сучасних GPU або TPU.
Модель може бути донавчена на конкретних задачах, що робить її адаптивною до різних завдань NLP.	BERT містить мільйони параметрів, що ускладнює її використання на пристроях з обмеженими ресурсами.
Модель BERT ефективна для класифікації тексту, заповнення пропусків, аналізу настроїв та інших завдань обробки природної мови.	BERT орієнтований на розуміння контексту, але не на генерацію тексту.
Можливість поєднувати з іншими нейронними мережами для покращення точності прогнозів.	Максимальна довжина послідовності обмежена, зазвичай 512 токенів, що може бути недостатньо для аналізу довгих текстів.

2.4 Порівняльний аналіз алгоритмів класифікації та глибоких нейронних мереж

Беручи до уваги швидкий розвиток можливостей штучного інтелекту та технологій, потрібно також розуміти, що хакерські атаки зловмисників також не залишаються на примітивному рівні, а постійно вдосконалюються. Фішингові

листи стають більш продуманими та завуальованими, використовуючи складні лінгвістичні конструкції та психологічні прийоми для маніпулювання користувачами. Тому вибір ефективного методу класифікації тексту для виявлення нелегітимних листів є досить важливим під час розробки методів захисту.

Традиційні методи машинного навчання, такі як Logistic Regression та Naive Bayes, широко використовуються для класифікації текстів завдяки їх простоті та інтерпретованості. Logistic Regression оперує ймовірнісними оцінками належності текстів до певних класів на основі виділених ознак. Цей метод демонструє прийнятну точність при роботі з простими фішинговими листами, але суттєво обмежений у здатності розпізнавати контекстуальні зв'язки між словами та виявляти приховані маніпулятивні патерни.

Алгоритми на основі дерев рішень, такі як Decision Tree та Random Forest, пропонують більш гнучкий підхід до класифікації, аналізуючи комбінації ознак та їх важливість. Random Forest, як ансамблевий метод, підвищує точність за рахунок усереднення результатів багатьох дерев рішень, що знижує ризик перенавчання. Однак, ці підходи також працюють з попередньо виділеними ознаками та не враховують послідовність слів, що є критичним обмеженням при аналізі складних текстових структур, характерних для сучасних фішингових атак.

Support Vector Machine (SVM) - потужний алгоритм класифікації, який шукає оптимальну гіперплощину для розділення даних різних класів. SVM добре працює з текстами середньої складності, особливо при використанні нелінійних ядер, але його ефективність значно знижується при обробці великих обсягів текстів з комплексними лінгвістичними конструкціями.

Рекурентні нейронні мережі, зокрема LSTM (Long Short-Term Memory), ознаменували значний прорив у обробці текстових даних завдяки здатності аналізувати послідовності та зберігати контекст на довгих відрізках тексту. LSTM моделі ефективно виявляють залежності між словами, враховуючи їх порядок та взаємозв'язки. Архітектура LSTM включає спеціальні механізми

"воріт", які дозволяють контролювати потік інформації, зберігаючи релевантні дані та відкидаючи несуттєві. Це особливо корисно при аналізі фішингових листів, де контекст відіграє вирішальну роль у розумінні справжніх намірів відправника. Однак, попри значні переваги, LSTM має суттєве обмеження – він обробляє текст однонаправлено, зазвичай зліва направо, що не дозволяє повністю врахувати контекст всього повідомлення. У фішингових листах критична інформація або маніпулятивні елементи можуть бути розподілені по всьому тексту, і однонаправлений аналіз може пропустити важливі взаємозв'язки.

На відміну від попередніх підходів, модель BERT представляє революційний підхід до обробки природної мови. Ключова перевага BERT полягає в його здатності аналізувати текст двонаправлено, враховуючи контекст як з попередніх, так і з наступних слів одночасно. Модель використовує механізм самоуваги (self-attention), який дозволяє оцінювати важливість кожного слова відносно всіх інших слів у реченні, незалежно від їх позиції. Архітектура BERT базується на трансформерах, які забезпечують паралельну обробку всього тексту, на відміну від послідовної обробки в LSTM. Це не лише підвищує швидкість навчання та інференсу, але й дозволяє моделі краще розуміти семантичні нюанси та контекстуальні зв'язки. BERT попередньо навчається на величезних корпусах тексту з використанням двох основних задач: передбачення маскованих слів та прогнозування наступного речення, що дозволяє моделі отримати глибоке розуміння мовних структур ще до етапу тонкого налаштування під конкретну задачу.

Вибір BERT як основного алгоритму для аналізу фішингових листів у даному дослідженні обумовлений його здатністю до двонаправленого аналізу тексту, глибокого розуміння контексту та виявлення прихованих взаємозв'язків між різними частинами повідомлення. Ці характеристики є критично важливими для ефективного виявлення сучасних фішингових атак. Хоча BERT потребує більше обчислювальних ресурсів порівняно з традиційними методами, його вища точність та надійність у виявленні фішингу повністю виправдовують

цей недолік, особливо враховуючи потенційні збитки від успішних фішингових атак.

Висновки за розділом 2

У другому розділі було проведено аналіз сучасних технологій захисту від фішингових атак та методів машинного навчання, що використовуються для виявлення шахрайських листів. Зокрема, розглянуто основні технологічні рішення, такі як SPF, DKIM, DMARC, які дозволяють знизити успішність атак, спрямованих на підробку електронних адрес. Також проаналізовано методи багатофакторної автентифікації, які забезпечують додатковий рівень захисту навіть у випадках компрометації облікових записів.

Також було розглянуто застосування алгоритмів машинного навчання: Logistic Regression, Decision Tree, Random Forest та SVM. Виявлено, що традиційні методи демонструють задовільні результати при роботі зі статичними фішинговими шаблонами, але мають суттєві обмеження у випадках складних текстових структур та контекстуальних маніпуляцій. Моделі LSTM довели свою ефективність при обробці текстових послідовностей завдяки здатності зберігати контекст та враховувати взаємозв’язки між словами. Водночас, основним недоліком LSTM є його однонаправленість та високі обчислювальні витрати.

Модель BERT, на відміну від LSTM, реалізує двонаправлений аналіз тексту, що дозволяє краще враховувати контекст та виявляти приховані маніпулятивні патерни. Це робить BERT оптимальним вибором для задачі виявлення фішингових листів, оскільки сучасні шахраї активно використовують техніки маніпуляції контекстом для введення користувачів в оману.

Таким чином, проведений аналіз дозволив визначити основні переваги та недоліки різних підходів до класифікації фішингових листів, що у наступному

розділі стане основою для розробки системи виявлення фішингу із застосуванням моделі BERT.

РОЗДІЛ 3
РОЗРОБКА ТА РЕАЛІЗАЦІЯ ЗАСОБУ ВИЯВЛЕННЯ ФІШИНГОВИХ ЛИСТІВ ЗА ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ

3.1 Вибір інструментів та середовища розробки

Під час реалізації практичної частини кваліфікаційної роботи на тему «Засіб виявлення фішингових листів за допомогою машинного навчання» було прийнято рішення застосовувати мову програмування Python та середовище розробки Visual Studio Code. На боці цього вибору лежить багато об’єктивних причин, що стосуються зручності, функціональності, наявністю всіх необхідних бібліотек та гарною сумісністю.

Python - це проста у вивченні, потужна мова програмування. Вона має ефективні високорівневі структури даних та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис і динамічна типізація Python, а також його інтерпретована природа роблять її ідеальною мовою для написання сценаріїв і швидкої розробки додатків у багатьох сферах на більшості платформ. [26] Також дана мова програмування є досить популярною у сфері машинного навчання, обробки природної мови та роботи з даними. Головними перевагами є велика кількість готових бібліотек, читабельність та зручна структура коду, що дає великі можливості для розробки.

У контексті даної роботи, python має простий інструментарій для реалізації додатку, обробки текстових даних, а саме датасету, проведення токенізації, нормалізації тексту а також для роботи з глибокими нейронними мережами. Більше того, він має зручну інтеграцію з популярними бібліотеками, наприклад:

TensorFlow, PyTorch, Keras, HuggingFace Transformers, котрі було використано під час розробки. Також важливим аргументом у виборі даної мови програмування є наявність інструментів для роботи з даними, зокрема такі бібліотеки як Pandas, Numpy. Вони дозволяють швидко завантажувати, очищати, аналізувати текстову інформацію, що є досить зручним при роботі з великою кількістю даних для формування та редагування датасету.

Під час вибору середовища розробки було проведено дослідження двох кандидатів, а саме Pycharm та Visual Studio Code (далі VS code), та створено порівняльну таблицю 3.1 наведену нижче.

Таблиця 3.1

Порівняльний аналіз середовищ розробки: PyCharm та Visual Studio Code

Характеристика	**PyCharm**	**VS code**
Тип	IDE (повноцінне інтегроване середовище)	Редактор коду (легкий, з можливістю розширення)
Швидкість запуску	Важчий, повільно запускається	Дуже швидкий та легкий
Інтерфейс Jupyter Notebook	Інтегрований (у версії Professional)	Доступний через розширення Jupyter
Інтеграція з Git	Повноцінна	Також хороша, через розширення
Робота з віртуальним середовищем	Автоматичне виявлення	Можна налаштувати через .venv або conda
Ресурси системи	Потребує більше пам’яті	Менш вимогливий до ресурсів
GPU-орієнтована розробка	Можлива, добре інтегрується з TensorFlow/PyTorch	Можлива, зручна через термінал і Jupyter
Ціна	Безкоштовна (Community Edition) / Платна (Pro)	Безкоштовна
Зручність для ML/Data Science	Зручно, особливо у Pro-версії	Дуже зручно, якщо підключено всі необхідні розширення

Враховуючи результати порівняльного аналізу середовищем розробки було обрану VS code. Цей редактор поєднує в собі легкість, швидкодію та

гнучке налаштування. Visual Studio Code має безліч розширень, які полегшують роботу з Python, включно з підсвіткою синтаксису, автодоповненням, запуском скриптів безпосередньо з редактора, а також інтеграцією з Jupyter Notebook. Для інтерактивної обробки даних, перевірки результатів токенізації та аналізу помилок активно використовувалося середовище Jupyter, що дозволяло ефективно тестувати окремі частини коду.

Зручність під час розробки забезпечила можливість працювати з віртуальними середовищами Python безпосередньо з VS Code. Це дозволяє ізолювати залежності проєкту, зменшуючи конфлікти між бібліотеками та забезпечуючи стабільність коду. Для створення та керування віртуальним середовищем використовувалась система venv, що забезпечувала мінімалістичне, але повноцінне середовище для запуску та тестування додатку.

Значну увагу було приділено сумісності інструментів з GPU-прискоренням, адже навчання глибоких нейронних мереж, зокрема BERT, вимагає значних обчислювальних ресурсів. Бібліотеки PyTorch та TensorFlow підтримують апаратне прискорення, а інтеграція з CUDA дозволила суттєво скоротити час навчання моделі. Використання GPU (NVIDIA RTX 3060, 12 GB) дало змогу навчати модель із більшим batch size і швидше здійснювати обчислення порівняно з процесорним режимом.

В результаті комбінація мови Python, середовища Visual Studio Code, інтеграції з Jupyter Notebook та підтримки GPU-прискорення забезпечила оптимальні умови для реалізації усіх етапів практичної частини дипломної роботи — від обробки даних до навчання та тестування моделі машинного навчання. Простота розробки, широка підтримка інструментів та бібліотек, зручність візуалізації результатів та гнучкість у роботі зробили цей вибір найбільш доцільним і ефективним у межах поставленого завдання.

3.2 Реалізація етапів навчання моделі BERT та розробка кінцевого додатку

Першим кроком в реалізації навчання моделі BERT є необхідним забезпечити правильну структуру датасету. Датасет - це сукупність даних, зазвичай організованих у вигляді таблиць, масивів або спеціальних форматів, таких як CSV або JSON, для зручного пошуку та аналізу.[27] Набір даних має містити щонайменше два основних поля, а саме поле «body», яке містить текст з листів, та відповідну мітку приналежності до певного класу «label», яка є показником фішингових листів тобто «1» та легітимних, «0». Також необхідною рекомендацією є забезпечення рівномірний розподіл класів для кращого результату навчання моделі.

Під час формування датасету було проведено аналіз моделі BERT та враховано безліч критеріїв необхідних для якісного навчання, а саме різні приклади нелегітимних листів, як з прямим фішингом так із заувалюваним, повідомлення які не мають логічного змісту, щоб уникнути помилкового виявлення фішингу в них, а також набір чистих повідомлень різного формату. Джерелами для створення датасету були відкриті репозиторії, а також наявні листи з різних поштових доменів. Далі було сформовано дві основні категорії: фішингові листи позначені як клас «1» та легітимні - «0». Перед подачею датасету в модель BERT текстові дані проходять декілька обов’язкових етапі нормалізації: приведення тексту до нижнього регістру, видалення зайвих пробілів, HTML-тегів, спец символів, а також очищення від зайвих технічних вставок. Важливим також було зберегти семантичну цінність тексту, оскільки модель BERT орієнтована саме на контекстуальний аналіз слів у реченні.

Під час виконання робіт з набором даних було використано такі технології:

- Python як основна мова обробки даних;

- бібліотека Pandans для проведення різних маніпуляцій роботи з CSV-файлами;
- NumPy для роботи з векторними представленнями та масивами.

Результатом є сформований збалансований датасет на 19489 рядків даних у форматі CSV під назвою phishing_dataset_bert, який в подальшому буде використано для навчання моделі. Наступним етапом була реалізація процесу навчання моделі BERT, з використанням середовища розробки Visual Studio Code у поєднанні з інтерактивним інтерфейсом Jupyter Notebook, такий підхід забезпечив зручність роботи з великими обсягами текстових даних, можливість поетапного виконання коду та швидкий аналіз результатів. Для ізоляції залежностей та стабільної роботи бібліотек було розгорнуто віртуальне

```
import pandas as pd
df1 = pd.read_csv(r"D:\диплом\datasets\phishing_dataset_bert.csv")

df = pd.concat([df1], ignore_index=True)
```
[1]

середовище Python за допомогою venv. Це дало змогу уникнути конфліктів між версіями бібліотек, забезпечити повторюваність результатів і гнучкість під час встановлення додаткових пакетів. Імопрт необхідних біблиотек зображено на риснуку 3.1 нижче.

Рисунок 3.1 - Імпорт необхідних бібліотек та завантаження вхідного датасету

На початковому етапі реалізації навчання моделі здійснюється імпорт необхідних бібліотек, а саме Pandas та завантаження вхідного датасету для подальшого опрацювання. Дані зчитуються із зовнішнього джерела у табличному форматі, після чого формуються в єдину структуру для забезпечення подальшої обробки. Це дозволяє створити уніфікований масив даних, який надалі буде використано для попередньої обробки, токенізації та навчання моделі. Такий підхід дає змогу легко масштабувати обсяг даних,

об’єднувати кілька джерел та проводити підготовку до навчального процесу.

```
from sklearn.model_selection import train_test_split

train_val_df, test_df = train_test_split(
    df, test_size=0.10, stratify=df["label"], random_state=42
)
train_df, val_df = train_test_split(
    train_val_df, test_size=0.10, stratify=train_val_df["label"], random_state=42
)
```

Поділ датасету зображено на рисунку 3.2 нижче.

Рисунок 3.2 - Поділ датасету на навчальну, валідаційну та тестову вибірки

Наступним етапом є поділ датасету на навчальну, валідаційну та тестові вибірки, що важливо для побудови хорошої моделі. У процесі поділу застосовується метод стратифікованого розділення, що дозволяє зберегти пропорції класів у кожній з вибірок. Спочатку виділяється тестова вибірка 10%, яке не використовується на етапі навчання, а слугує для кінцевої перевірки результатів. Решта даних поділяється на навчальну та валідаційну вибірки у співвідношенні 90/10. Валідаційна використовується для оцінки проміжної якості моделі під час тренування та допомагає уникнути перенавчання. Отже, такий поділ забезпечує коректну організацію даних для ефективного навчання моделі. Токенізація тексту зображена на рисунку 3.3 нижче.

```
from transformers import BertTokenizerFast

tokenizer = BertTokenizerFast.from_pretrained("bert-base-uncased")

def encode_batch(batch):
    tokens = tokenizer(
        batch["body"],
        padding="max_length",
        truncation=True,
        max_length=512
    )
    tokens["labels"] = batch["label"]
    return tokens
```

Рисунок 3.3 - Токенізація тексту

Третім етапом є здійснення токенізації тексту з використанням попередньо навченого токенізатора моделі BERT. Це важлива частина

підготовки даних, яка полягає в перетворенні тексту в числовий формат, зрозумілий для нейронної мережі.

У роботі використано BertTokenizerFast - оптимальний токенізатор з бібліотки Transformers, який забезпечує швидке та ефективне перетворення текстових даних. Також було завантажено модель bert-base-uncased, яка є однією з найпопулярніших версій BERT, не чутливою до регістру.

У створеній функції текстові дані з поля body токенізуються з використанням таких параметрів:

- padding="max_length" — вирівнювання довжини кожного прикладу до фіксованого значення;
- truncation=True — обрізання занадто довгих повідомлень до допустимої довжини;
- max_length=512 — максимальна кількість токенів, яку може обробити BERT.

До структури з токенами додається поле labels, що містить відповідну класову мітку, яка необхідна для навчання моделі. Цей етап дозволяє сформувати вхідний масив даних у форматі, придатному для подачі до моделі BERT, з урахуванням її архітектурних обмежень. Підготовка датасетів зображена на рисунку 3.4 нижче.

```
[4]
from datasets import Dataset

train_ds = Dataset.from_pandas(train_df).map(encode_batch, batched=True)
val_ds   = Dataset.from_pandas(val_df  ).map(encode_batch, batched=True)
test_ds  = Dataset.from_pandas(test_df ).map(encode_batch, batched=True)

for name in ["train_ds", "val_ds", "test_ds"]:
    ds = locals()[name]
    ds = ds.remove_columns(["body", "label", "token_type_ids", "__index_level_0__"])
    ds.set_format("torch", columns=["input_ids", "attention_mask", "labels"])
    locals()[name] = ds
```

Рисунок 3.4 - Підготовка датасетів для подачі в модель у форматі, сумісному з PyTorch

На даному кроці виконується підготовка датасетів для подачі в модель у форматі, сумісному з PyTorch. Перетворення відбувається із використанням

інструментів бібліотеки datasets від HuggingFace, яка дозволяє зручно обробляти великі масиви даних.

Конвертація дата фреймів у формат Dataset:

- використовується метод from_pandas, яки перетворює pandas.DataFrame у формат Dataset;
- до кожного набору застосовується токенізація за допомогою функції encode_batch, параметр batched = True дозволяє обробляти дані блоками, що прискорює процес.

Очищення датасету:

- видаляються непотрібні колонки, зокрема оригінальний текст «body», мітка у старому форматі «label», службові поля, які не потрібні моделі «token_type_ids, __index_level_0__».

Формування під PyTorch:

- метод set_format налаштовує структуру датасету на формат torch.Tensor;
- вказуються ключові поля, які очікує модель, а саме «input_ids», «attention_mask» та «labels».

Цей етап завершує підготовку даних до навчання, приводячи їх у структурований та оптимізований вигляд, сумісний з моделлю BERT і фреймворком PyTorch.

Даний етап присвячений ініціалізації моделі та налаштування параметрів навчання для процесу класифікації електронних листів за допомогою моделі BERT.

Налаштування аргументів навчання:

- output_dir = "bert_phishing"- директорія для збереження проміжних моделей;
- num_train_epochs = 5 – кількість епох навчання;
- per_device_train_batch_size = 16, per_device_eval_batch_size=32 - розмір пакету (batch size) для навчання і валідації;

- evaluation_strategy = "steps", eval_steps=500 – оцінка точності кожні 500 кроків;
- save_strategy = "steps", save_steps=500 - збереження моделі кожні 500 кроків;
- logging_steps = 200 – частота логування проміжних результатів;
- learning_rate = 2e-5 – швидкість навчання (learning rate), підібрана з урахуванням специфіки BERT;
- weight_decay = 0.01 – регуляризація, що допомагає уникнути перенавчання;
- load_best_model_at_end = True – завантаження найкращої моделі після завершення тренування;
- metric_for_best_model = "f1" – використання F1-міри як ключової метрики для оцінювання якості.

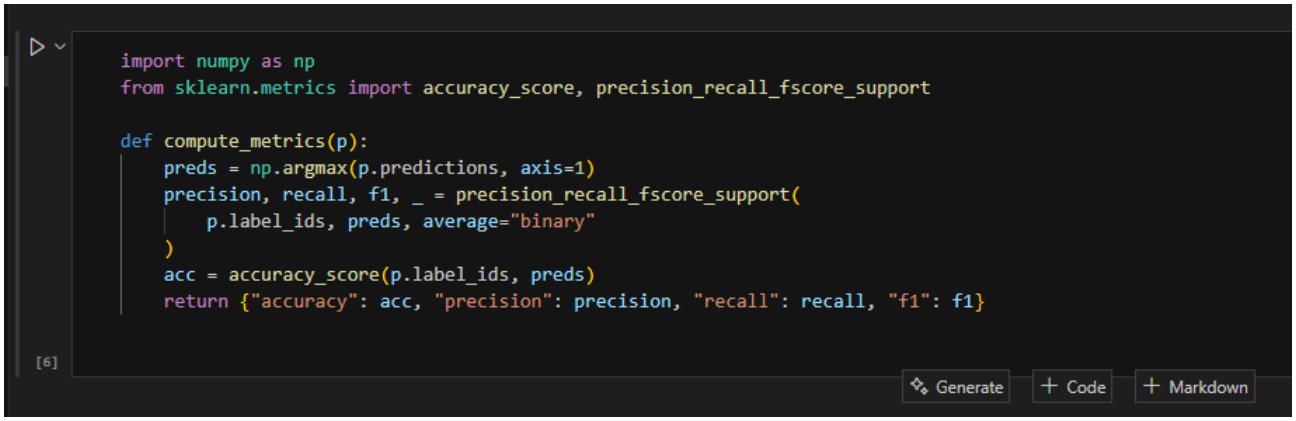
Ініціалізація моеделі зображена на рисунку 3.5 нижче.

```
from transformers import BertForSequenceClassification, TrainingArguments

model = BertForSequenceClassification.from_pretrained(
    "bert-base-uncased",
    num_labels=2
)

training_args = TrainingArguments(
    output_dir="bert_phishing",
    num_train_epochs=5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=32,

    evaluation_strategy="steps",
    save_strategy="steps",
    eval_steps=500,
    save_steps=500,

    logging_steps=200,

    learning_rate=2e-5,
    weight_decay=0.01,

    load_best_model_at_end=True,
    metric_for_best_model="f1"
)
```

Рисунок 3.5 - Ініціалізація моделі та налаштування параметрів навчання

Ці налаштування формують основу для стабільного та керованого процесу навчання, дозволяючи забезпечити ефективне оновлення ваг та контроль за якістю моделі впродовж усього циклу тренування.

Одним із завершальних етапів - це налаштування функції обчислення метрик якості моделі, які використовуються для оцінки її ефективності під час валідації та тестування. Це важливий крок, що дозволяє кількісно оцінити



здатність моделі правильно класифікувати вхідні повідомлення. Налаштування метрик зображено на рисунку 3.6 нижче.

Рисунок 3.6 - Налаштування метрик якості моделі

В процесі реалізації було використано спеціальну функцію compute_metrics, яка обробляє вихідні дані моделі. На початку, на основі ймовірностей класів, обирається найбільш ймовірний варіант класифікації, далі за допомогою функції precision_recall_fscore_support з бібліотеки sklearn.metrics розраховуються ключові метрики, а саме: точність (англ. precision), повнота (англ. recall) та F1-міра, також обраховується загальна точність (англ. accuracy), котра відображає частку правильно класифікованих прикладів у всьому наборі.

Отримані значення повертаються у вигляді словника, що в подальшому дозволяє використовувати їх для автоматичного моніторингу ефективності моделі у процесі навчання. Отже, ця функція забезпечує формалізовану оцінку якості класифікації, що є важливо при побудові надійної системи виявлення

фішингових листів. Процес запуску навчання моеделі зображено на рисунку 3.7 ничже.

```
from transformers import Trainer

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_ds,
    eval_dataset=val_ds,
    compute_metrics=compute_metrics,
    tokenizer=tokenizer
)

trainer.train()
```

Рисунок 3.7 - Процес запуску навчання моделі

На фінальному кроці відбувається процес запуску навчання моделі із використанням інтерфейсу Trainer, який містить бібліотека Transformers.

Основні компоненти конфігурації:

- model = model - передається об’єкт моделі BERT, адаптований до задачі класифікації;
- arg = training_args - передаються параметри тренування, попередньо налаштовані у структурі TrainingArguments;
- train_dataset = train_ds - набір даних, що використовуються для навчання;
- eval_dataset = val_ds - валідаційний набір, який використовується для оцінки якості моделі під час навчання;
- compute_metrics = compute_metrics - функція для обчислення метрик точності, повноти та F1-міри;
- tokenizer = tokenizer - токенізатор, який забезпечує коректне відображення текстових даних у формат, який сприймає модель.

Виклик методу trainer.train() ініціює повноцінний цикл тренування моделі, включаючи обчислення зворотного поширення помилки, оновлення ваг, валідацію на кожному кроці та збереження кращої версії моделі за метрикою F1.

Цей підхід забезпечує структуровану та ефективну організацію навчального процесу, зменшує ймовірність помилок і дозволяє зосередитися на оптимізації та аналізі результатів. Збережння натреновоної моделі зображено на рисунку 3.8 нижче.

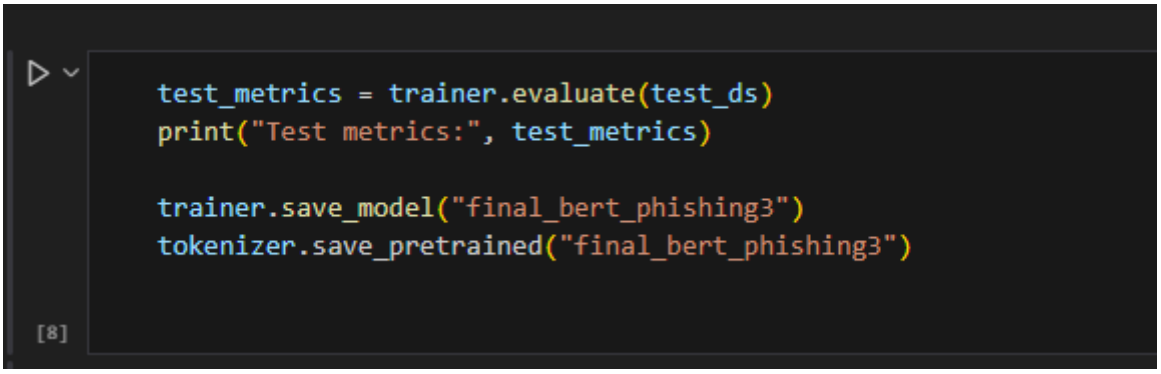


Рисунок 3.8 - Збереження натренованої моделі

На завершальному етапі відбувається оцінка якості навченої моделі на тестовому наборі даних та збереження підсумкової версії моделі і токенізатора.

За допомогою методу evaluate() проводиться оцінювання моделі на тестовій вибірці, яка раніше не використовувалась у процесі навчання або валідації. Це дозволяє об’єктивно перевірити, наскільки добре модель узагальнює знання на нових, невідомих даних.

Метрики, що обчислюються (accuracy, precision, recall, F1), дозволяють оцінити загальну ефективність моделі у виявленні фішингових листів.

Після завершення оцінки модель та токенізатор зберігаються на диск у зазначену директорію. Це дає змогу у подальшому:

- завантажити модель для повторного використання без повторного тренування;
- інтегрувати її в інші системи;
- використати для донавчання або fine-tuning з іншими наборами даних.

Таким чином, завершальний етап забезпечує повторюваність експерименту та створює основу для практичного впровадження моделі в реальні інформаційні системи.

Заключною частиною була реалізація графічного інтерфейсу користувача, що дозволяє інтерактивно взаємодіяти з навченою моделлю виявлення фішингових листів. Для побудови використовувались бібліотеки tkinter та customtkinter, котрі надають як базовий функціонал, так і сучасний стиль оформлення інтерфейсу.

Було реалізовано два основних режими перевірки повідомлень:

1. Автоматичне отримання листів з електронної пошти за допомогою Gmail API. Це дозволяє перевіряти останні отримані листи з облікового запису Gmail, обробляти їхній вміст та передавати в модель для аналізу.

2. Ручне введення тексту у відповідне текстове поле. Ця функція досить корисна для тестування окремих фрагментів тексту або введення листів, що не зберігаються в Gmail.

Також важливим кроком було реалізувати обробку довгих листів. З огляду на обмеження моделі BERT, котра може працювати лише з текстом довжиною до 512 токенів, у додатку реалізовано механізм автоматичного розділення тексту на частини відповідної довжини. Кожен фрагмент аналізується окремо, а результати агрегуються для формування кінцевого рішення щодо легітимності листа. Інтерфейс також передбачає візуальне відображення результатів класифікації, з відповідною позначкою як фішингового так і безпечного листа.

Таким чином, реалізований програмний додаток дозволяє користувачеві легко взаємодіяти з моделлю машинного навчання, застосовувати її у реальних умовах, а також демонструє можливість інтеграції інтелектуальних технологій в існуючі засоби електронного листування.

3.3 Практична цінність та подальші кроки вдосконалення

Розроблений засіб, орієнтований на виявлення фішингових листів за допомогою моделі машинного навчання BERT має важливу практичну цінність у контексті забезпечення кібербезпеки кінцевих користувачів. Враховуючи швидкість розвитку технологій фішингові атаки також набувають інших масштабів та складності, а пошта залишається одним із основних каналів поширення загрози. Хоча окремі великі поштові сервіси, такі як Gmail або Outlook, вже використовують елементи штучного інтелекту для виявлення шкідливих повідомлень, у більшості інших поштових систем захист все ще базується на простих правилах або сигнатурному аналізі. Такий підхід залишає значну частину користувачів незахищеними, особливо у випадках використання менш популярних або корпоративних поштових доменів.

Запропонований додаток дозволяє інтегрувати можливості штучного інтелекту у процес перевірки листів для будь-якого користувача, незалежно від обраної поштової платформи. Він не потребує глибоких технічних знань або складного налаштування, а тому може використовуватись широким колом користувачів. Практична реалізація інтеграції з Gmail API дає змогу здійснювати автоматизований аналіз вхідних повідомлень у реальному часі, виявляти потенційно небезпечні листи та попереджати користувача про можливу загрозу. Додатково реалізована функція введення тексту вручну розширює сферу застосування - модель може бути використана навіть для аналізу повідомлень із зовнішніх джерел, які не пов'язані безпосередньо з електронною поштою.

Окремою перевагою є механізм попередньої обробки тексту та розділення на блоки у разі перевищення порогу в 512 токенів, встановленого для моделі BERT. Це дозволяє зберігати цілісність аналізу навіть при роботі з великими обсягами інформації, що характерно для листів із вкладеннями, довгими інструкціями або внутрішніми службовими повідомленнями. Таким чином, реалізований функціонал є досить гнучким і масштабованим, що дозволяє адаптувати систему під різні сценарії використання.

Слід зазначити, що поточна реалізація додатку є прототипом, який має певні обмеження. Серед основних проблем можна висвітлити обмеженість розміру навчального датасету, обмежені обчислювальні ресурси, які використовувались під час тренування моделі, а також відсутність перевірки листів на основі додаткових критеріїв, таких як структура доменів відправника, наявність шкідливих IP-адрес або аналіз скорочених URL-посилань.

Для вдосконалення системи в майбутньому можна запропонувати кілька ключових напрямків:

1. Розширення та збагачення навчального датасету - залучення більшої кількості якісних даних дозволить підвищити точність класифікації та зменшити кількість хибних результатів.
2. Інтеграція з базами фішингових доменів та IP-адрес - поєднання аналізу тексту з перевіркою технічних параметрів листа дозволить забезпечити більш комплексну оцінку.
3. Інтерфейсна інтеграція - реалізація функціоналу як розширення для браузера або як модуль, вбудований безпосередньо в поштову систему, що дозволить автоматизувати процес без взаємодії користувача.
4. Використання хмарних сервісів - перенесення моделі в хмару дозволить збільшити швидкість обробки та забезпечити доступ до системи з будь-якого пристрою без необхідності локального запуску.

Таким чином, створений додаток вже на початковому етапі продемонстрував свою ефективність та потенціал для масштабування. Його подальший розвиток дозволить забезпечити більш високий рівень захисту користувачів.

Висновки за розділом 3

У третьому розділі було детально описано реалізацію практичної частини кваліфікаційної роботи, яка включала створення, навчання та інтеграцію моделі

машинного навчання для виявлення фішингових електронних листів. Було обґрунтовано вибір основних інструментів та середовища розробки, таких як Python, Visual Studio Code та Jupyter Notebook. Завдяки широкій екосистемі бібліотек Python та можливості GPU-прискорення, реалізація всіх етапів - від попередньої обробки даних до навчання моделі - була здійснена ефективно.

Описано формування та нормалізацію датасету, підготовку даних до подачі в модель BERT, а також архітектурні особливості навчання моделі з урахуванням параметрів, що забезпечують її стабільну роботу. Здійснено інтеграцію натренованої моделі у графічний додаток, який дозволяє користувачу проводити перевірку листів як через Gmail API, так і шляхом ручного введення тексту, з урахуванням обмеження в 512 токенів.

Підсумком розділу є демонстрація практичного застосування розробленого інструменту, що підтверджує можливість успішної реалізації засобу виявлення фішингових листів навіть у середовищі обмежених ресурсів. Запропоновано також шляхи вдосконалення додатку, зокрема розширення функціоналу, збільшення обсягів навчальних даних та можливість хмарної інтеграції, що відкриває перспективи для подальшого розвитку і впровадження системи в реальних умовах.

ВИСНОВКИ

У цій кваліфікаційній роботі було досліджено проблему фішингових атак в електронній пошті та способи їх виявлення за допомогою методів машинного навчання. Актуальність теми обумовлена тим, що фішинг залишається однією з найпоширеніших кіберзагроз у світі та щороку завдає великих збитків як окремим користувачам, так і компаніям.

У теоретичній частині роботи було розглянуто, що таке фішинг, які бувають його види, як виглядають фішингові листи, та які методи використовуються для їх створення. Також було проаналізовано законодавство щодо відповідальності за фішингові атаки, а також статистику подібних інцидентів за останні роки.

В аналітичній частині розглянуто сучасні технологічні підходи до боротьби з фішингом, зокрема методи на основі аналізу вмісту листів та використання штучного інтелекту. Було досліджено моделі машинного навчання, які найкраще підходять для виявлення фішингових повідомлень, такі як LSTM, GRU, CNN та BERT.

У практичній частині було реалізовано систему для автоматичного виявлення фішингових листів. Для цього було обрано мову Python, середовище Visual Studio Code та бібліотеки для роботи з нейронними мережами. Сформовано власний датасет листів, виконано обробку тексту, навчання моделі BERT і її подальнє тестування. Створено зручний графічний інтерфейс, який дозволяє користувачу перевіряти як текст з Gmail, так і вручну введений текст. Також реалізовано автоматичне розбиття довгих листів, які перевищують обмеження моделі.

Розроблена система має практичну цінність, оскільки може допомогти звичайним користувачам, які не мають спеціальних знань у сфері кібербезпеки, розпізнавати потенційно небезпечні листи. На відміну від великих компаній, де

подібні технології вже використовуються, пересічні користувачі не мають доступу до таких рішень, тому створений додаток може бути корисним.

У подальшому систему можна вдосконалити: розширити обсяг даних для навчання, додати аналіз не лише тексту, а й метаданих (адреси, домени, IP), а також реалізувати інтеграцію в поштові клієнти або вебсервіси для зручнішого використання.

Отже, всі цілі та завдання, поставлені на початку роботи, були виконані, а розроблена система підтвердила ефективність застосування машинного навчання для виявлення фішингових листів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cisco. What is phishing? [Електронний ресурс] – Режим доступу: https://www.cisco.com/site/us/en/learn/topics/security/what-is-phishing.html (дата звернення: 12.02.2025).

2. DMARC Report. [Електронний ресурс] – Режим доступу: https://dmarcreport.com/ (дата звернення: 21.03.2025).

3. IBM. What is phishing? [Електронний ресурс] – Режим доступу: https://www.ibm.com/think/topics/phishing (дата звернення: 03.03.2025).

4. Industrial Cyber. [Електронний ресурс] – Режим доступу: https://industrialcyber.co/ (дата звернення: 15.03.2025).

5. Cybeready. [Електронний ресурс] – Режим доступу: https://cybeready.com/ (дата звернення: 10.02.2025).

6. Про інформацію: Закон України від 02.10.1992 № 2657-XII. [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/2657-12#Text (дата звернення: 08.02.2025).

7. Про основні засади забезпечення кібербезпеки України: Закон України від 05.10.2017 № 2163-VIII. [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/675-19#Text (дата звернення: 01.02.2025).

8. Кримінальний кодекс України: Закон України від 05.04.2001 № 2341-III. [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/2341-14#Text (дата звернення: 05.03.2025).

9. Про захист персональних даних: Закон України від 01.06.2010 № 2297-VI. [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/2297-17#Text (дата звернення: 17.02.2025).

10. Концепція забезпечення національної системи кібербезпеки України: Розпорядження КМУ від 15.10.2020 № 1306-р. [Електронний ресурс] – Режим

доступу: https://zakon.rada.gov.ua/laws/show/v0417500-15#Text (дата звернення: 25.02.2025).

11. Convention on Cybercrime. The Budapest Convention. [Електронний ресурс] – Режим доступу: https://www.coe.int/en/web/cybercrime/the-budapest-convention (дата звернення: 14.03.2025).

12. ISO/IEC 27001 – Information security management. [Електронний ресурс] – Режим доступу: https://www.iso.org/standard/27001 (дата звернення: 18.03.2025).

13. Directive on security of network and information systems (NIS Directive). [Електронний ресурс] – Режим доступу: https://digital-strategy.ec.europa.eu/en/library/directive-security-network-and-information-systems-nis-directive (дата звернення: 12.03.2025).

14. Ting, H., & Kamarudin, S. (2023). Analysis of phishing attack trends, impacts and prevention methods: Literature study. [Електронний ресурс] – ScienceDirect – Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S095741742302701X (дата звернення: 02.03.2025).

15. Ali, S. et al. (2023). Analysis of Phishing Attack Trends, Impacts and Prevention Methods. [Електронний ресурс] – ResearchGate – Режим доступу: https://www.researchgate.net/publication/383193964 (дата звернення: 19.02.2025).

16. Bose, A., Applied Machine Learning and AI for Engineers: Solve Business Problems that Can’t Be Solved Algorithmically. – 1st ed. – Independently published, 2023. – 218 p.

17. Khan, A. (2024). A Machine Learning Algorithms for Detecting Phishing Websites: A Comparative Study. [Електронний ресурс] – ResearchGate – Режим доступу: https://www.researchgate.net/publication/384219004 (дата звернення: 04.03.2025).

18. IBM. Introduction to XGBoost. [Електронний ресурс] – Режим доступу: https://www.ibm.com/think/topics/xgboost (дата звернення: 06.03.2025).

19. GeeksforGeeks. Deep Learning: Introduction to LSTM. [Електронний ресурс] – Режим доступу: https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/ (дата звернення: 07.03.2025).

20. NVIDIA Developer. Discover LSTM. [Електронний ресурс] – Режим доступу: https://developer.nvidia.com/discover/lstm (дата звернення: 09.03.2025).

21. IBM. Recurrent Neural Networks. [Електронний ресурс] – Режим доступу: https://www.ibm.com/think/topics/recurrent-neural-networks (дата звернення: 11.03.2025).

22. AWS. What is a Recurrent Neural Network (RNN)? [Електронний ресурс] – Режим доступу: https://aws.amazon.com/what-is/recurrent-neural-network/ (дата звернення: 13.03.2025).

23. Coursera. BERT Model. [Електронний ресурс] – Режим доступу: https://www.coursera.org/articles/bert-model (дата звернення: 20.03.2025).

24. GeeksforGeeks. Explanation of BERT Model (NLP). [Електронний ресурс] – Режим доступу: https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/ (дата звернення: 22.03.2025).

25. TechTarget. BERT Language Model. [Електронний ресурс] – Режим доступу: https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model (дата звернення: 24.03.2025).

26. Python Docs. Official Python Tutorial. [Електронний ресурс] – Режим доступу: https://docs.python.org/3.15/tutorial/index.html (дата звернення: 26.03.2025).

27. IBM. What is a Dataset? [Електронний ресурс] – Режим доступу: https://www.ibm.com/think/topics/dataset (дата звернення: 01.04.2025).