Міністерство освіти і науки України
Луцький національний технічний університет
Факультет робототехніки та штучного інтелекту
Кафедра штучного інтелекту та математичного моделювання

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ
«БАКАЛАВР»

АНАЛІЗ ТА РОЗРОБКА МЕТОДІВ ВИЯВЛЕННЯ СПАМУ ТА ФІШИНГУ В ЕЛЕКТРОННІЙ ПОШТІ З ВИКОРИСТАННЯМ ОБРОБКИ ПРИРОДНОЇ МОВИ

ANALYSIS AND DEVELOPMENT OF EMAIL SPAM AND PHISHING DETECTION METHODS USING NLP

Спеціальність 113 Прикладна математика
(шифр і назва спеціальності)

освітня програма «Штучний інтелект та аналіз масивів даних»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ПРМ-41
Ткачук Вадим Миколайович
_____
(підпис)

Керівник:
к.т.н., доцент
Приходько Олексій Сергійович
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«___» _____ 20__ р.
к.т.н., доцент
Гарант освітньої програми:
Приходько Олексій Сергійович
_____
(підпис)

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *архітектури, будівництва та дизайну*
Кафедра *прикладної математики та механіки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *11 Математика і статистика*
Спеціальність *113 Прикладна математика*
Освітня програма *Штучний інтелект та аналіз масивів даних*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Мікуліч О.А.
«___» _____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Ткачук Вадим Миколайович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Аналіз та розробка методів виявлення спаму та фішингу в електронній пошті з використанням обробки природної мови \ Analysis and development of email spam and phishing detection methods using NLP
Керівник роботи: *Приходько Олексій Сергійович*
затверджені наказом закладу вищої освіти від «31» грудня 2025 р. № 557/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 04.06.2026 р.

3. Вихідні дані до роботи *профільні публікації з штучного інтелекту та математичного моделювання в межах досліджуваної проблематики; релевантні набори даних; математичні моделі цільових процесів; технічна документація Python-бібліотек та методичні вказівки до виконання кваліфікаційної роботи бакалавра*

4. Зміст пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз предметної області
Постановка задачі та вибір методів
Практична реалізація
Отримання та аналіз результатів

5. Перелік графічного (ілюстративного) матеріалу:
Презентація роботи (слайди): об'єкт, предмет, мета та завдання дослідження; аналіз предметної області; математичні та алгоритмічні моделі, результати експериментальних досліджень (метрики та візуалізація роботи алгоритму); загальні висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
1 розділ	*Приходько О.С., доцент кафедри*		
2 розділ	*Приходько О.С., доцент кафедри*		
3 розділ	*Приходько О.С., доцент кафедри*		
4 розділ	*Приходько О.С., доцент кафедри*		
Висновки	*Приходько О.С., доцент кафедри*		

7. Дата видачі завдання «___» _____ 202__ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми*	*до 01.03.2026*	
2.	*Перший розділ*	*до 5.03.2026*	
3.	*Другий розділ*	*до 01.04.2026*	
4.	*Третій розділ*	*до 10.04.2026*	
5.	*Четвертий розділ*	*до 20.04.2026*	
6.	*Висновки*	*до 28.04.2026*	
7.	*Формування списку використаних джерел*	*до 05.05.2026*	
8.	*Оформлення ілюстративного матеріалу*	*до 09.05.2026*	
9.	*Нормоконтроль*	*до 20.05.2026*	
10.	*Інструментальна перевірка на академічний плагіат*	*до 02.062026*	*Показник запозичень тексту ____ %*
11.	*Представлення кваліфікаційної роботи бакалавра до захисту*	*до 04.06.2026*	

Здобувач вищої освіти _____ (Ткачук В. М.)
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (Приходько. О. С.)
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Ткачук В. М. Аналіз та розробка методів виявлення спаму та фішингу в електронній пошті з використанням обробки природної мови. Рукопис.

Кваліфікаційна робота бакалавра ОП «Штучний інтелект та аналіз масивів даних» спеціальності 113 Прикладна математика. Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота присвячена розв'язанню актуальної задачі ідентифікації небажаної електронної кореспонденції за допомогою методів обробки природної мови (NLP) та машинного навчання. У роботі обґрунтовано доцільність переходу від класичних детермінованих систем фільтрації до адаптивних нейромережевих рішень. Сформовано та збалансовано унікальний масив даних обсягом 38 610 листів, що включає три класи: легітимні повідомлення, спам та фішинг. Розроблено автоматизований конвеєр попередньої обробки тексту, який реалізує токенізацію, видалення шуму та векторизацію (TF-IDF, Word Embeddings).

Проведено експериментальне порівняння базових алгоритмів (Multinomial Naive Bayes, Logistic Regression) та глибокої нейронної мережі архітектури Bi-LSTM. Доведено, що класичні лінійні моделі схильні до перенавчання на специфічних лексичних маркерах через явище "зсуву даних". Встановлено, що оптимальним рішенням для виявлення складних соціально-інженерних фішингових атак є використання двонаправлених рекурентних мереж, здатних аналізувати семантичний контекст текстової послідовності. Розроблена модель Bi-LSTM досягла точності 99,79 % на валідаційній вибірці. Отримані результати мають високу практичну значущість і можуть бути інтегровані в сучасні системи корпоративної кібербезпеки.

Ключові слова: *обробка природної мови, машинне навчання, глибоке навчання, фішинг, спам, класифікація текстів, нейронні мережі, Bi-LSTM, векторизація тексту.*

ABSTRACT

Tkachuk V. M. Analysis and development of email spam and phishing detection methods using NLP. Manuscript.

Bachelor's Thesis in the field of "Artificial Intelligence and BigData Analysis" in specialty 113 Applied Mathematics. Lutsk National Technical University. Lutsk, 2026.

The qualification work is devoted to solving the urgent problem of identifying unwanted email correspondence using natural language processing (NLP) and machine learning methods. The paper substantiates the expediency of transitioning from classical deterministic filtering systems to adaptive neural network solutions. A unique dataset of 38,610 emails was aggregated and balanced, including three classes: legitimate messages, spam, and phishing. An automated text preprocessing pipeline was developed, implementing tokenization, noise removal, and vectorization (TF-IDF, Word Embeddings).

An experimental comparison of baseline algorithms (Multinomial Naive Bayes, Logistic Regression) and a deep neural network of Bi-LSTM architecture was conducted. It was proved that classical linear models are prone to overfitting on specific lexical markers due to the "dataset shift" phenomenon. It was established that the optimal solution for detecting complex social engineering phishing attacks is the use of bidirectional recurrent networks capable of analyzing the semantic context of a text sequence. The developed Bi-LSTM model achieved an accuracy of 99.79% on the validation set. The obtained results have high practical significance and can be integrated into modern corporate cybersecurity systems.

Keywords: natural language processing, machine learning, deep learning, phishing, spam, text classification, neural networks, Bi-LSTM, text vectorization.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному цифровому суспільстві електронна пошта залишається одним із найпоширеніших засобів корпоративної та особистої комунікації. Разом із тим, вона є основним вектором для кібератак, серед яких домінують спам та фішинг. За даними міжнародних організацій з кібербезпеки, частка небажаної кореспонденції становить понад половину всього світового email-трафіку. Якщо класичний спам здебільшого несе репутаційні та технічні загрози (перевантаження серверів, зниження продуктивності), то фішинг має на меті викрадення конфіденційних даних, фінансових активів та компрометацію цілих інформаційних систем.

Зловмисники постійно вдосконалюють методи обходу традиційних систем фільтрації, які базуються на чорних списках або простих евристичних правилах. Використання технологій соціальної інженерії та генерації тексту за допомогою штучного інтелекту дозволяє створювати фішингові листи, які складно відрізнити від легітимних. Це зумовлює критичну необхідність переходу від детермінованих алгоритмів захисту до адаптивних систем, заснованих на методах машинного навчання та обробки природної мови (NLP). Використання сучасних алгоритмів NLP, зокрема глибокого навчання та архітектур типу Transformer, дозволяє аналізувати семантику тексту, виявляти приховані патерни маніпуляцій та суттєво підвищувати точність класифікації кореспонденції. Отже, тема кваліфікаційної роботи є актуальною та має значне практичне значення для забезпечення інформаційної безпеки.

Мета дослідження полягає в аналізі, розробці та програмній реалізації методів виявлення спаму та фішингу в електронній пошті з використанням технологій обробки природної мови для підвищення точності та надійності фільтрації небажаних повідомлень.

Для досягнення поставленої мети було сформульовано та вирішено такі **завдання**:

1. Здійснити системний аналіз предметної області, дослідити існуючі підходи до виявлення спаму та фішингових атак.

2. Формалізувати задачу класифікації текстових масивів даних та обрати критерії оцінки ефективності алгоритмів.

3. Проаналізувати та підготувати масив даних (датасет) для навчання та тестування моделей.

4. Розробити конвеєр (pipeline) попередньої обробки та векторизації текстової інформації.

5. Програмно реалізувати базові алгоритми машинного навчання та сучасні нейромережеві моделі для класифікації текстів.

6. Провести експериментальне дослідження розроблених методів, порівняти їх показники та визначити найбільш ефективний підхід для вирішення поставленої задачі.

Об’єкт дослідження: процеси аналізу та класифікації текстових масивів даних у системах обробки електронної кореспонденції.

Предмет дослідження: моделі, алгоритми та методи обробки природної мови і машинного навчання для ідентифікації спам- та фішинг-повідомлень.

Методи дослідження. Для розв'язання поставлених завдань у роботі використано: методи теорії ймовірностей та математичної статистики (для аналізу розподілу даних та оцінки достовірності результатів); методи лінійної алгебри (для векторного представлення текстів); методи машинного навчання (для побудови моделей); методи обробки природної мови (для лематизації, токенізації та семантичного аналізу текстів); емпіричні методи (для проведення обчислювальних експериментів та оцінки якості роботи моделей).

Інформаційна база дослідження. Підґрунтям для виконання роботи стали наукові публікації вітчизняних та зарубіжних авторів у галузі штучного інтелекту, кібербезпеки та обробки природної мови, матеріали міжнародних конференцій, офіційна документація до бібліотек машинного навчання (Scikit-Learn, TensorFlow, PyTorch, Hugging Face Transformers), а також відкриті набори даних для досліджень у сфері Data Science.

У процесі підготовки бакалаврської кваліфікаційної роботи застосовувалися технології штучного інтелекту як допоміжний інструментарій. Зокрема, для стилістичної правки та структурування тексту використано ChatGPT-5 та Gemini 3.3. Автор несе повну відповідальність за зміст роботи

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ДАНИХ

1.1. Проблема ідентифікації спаму та фішингу в сучасних інформаційних системах

Стрімкий розвиток цифрових комунікацій перетворив електронну пошту на фундаментальний інструмент обміну інформацією як у корпоративному, так і в приватному секторах. Однак ця доступність та масовість зробила email-трафік основною мішенню для зловмисників. Згідно з дослідженнями в галузі кібербезпеки, понад половина всього глобального поштового трафіку класифікується як небажана кореспонденція, що включає рекламний спам, шкідливе програмне забезпечення та фішингові атаки [4].

Проблематика ідентифікації небажаної кореспонденції ускладнюється тим, що поняття "спаму" та "фішингу" мають різну природу та рівень загрози. Спам – це здебільшого масова розсилка повідомлень рекламного характеру, метою якої є привернення уваги до певного продукту чи послуги [10]. Незважаючи на відносно низький рівень прямої загрози, масовий спам призводить до значних фінансових та ресурсних втрат компаній через перевантаження мережевої інфраструктури та зниження продуктивності працівників.

Натомість фішинг становить критичну загрозу інформаційній безпеці. Це вид шахрайства, заснований на методах соціальної інженерії, метою якого є обманом змусити користувача надати конфіденційну інформацію (паролі, дані банківських карток, корпоративні ключі доступу) шляхом маскування під легітимні повідомлення від довірених організацій [1]. Сучасні фішингові кампанії характеризуються високим рівнем персоналізації та використання психологічних тригерів, таких як терміновість, страх втрати доступу до акаунту або обіцянка фінансової вигоди [2].

Традиційні системи захисту інформаційних систем від спаму та фішингу історично базувалися на детермінованих підходах, таких як чорні списки IP-

адрес, фільтрація за ключовими словами та перевірка репутації доменів (SPF, DKIM, DMARC) [8]. Незважаючи на свою ефективність проти масових та примітивних атак, ці методи демонструють суттєві недоліки в умовах сучасного кіберландшафту. Зловмисники застосовують техніки обфускації тексту, підміну символів, використання динамічних IP-адрес та компрометацію легітимних серверів для обходу статичних правил фільтрації [3].

Особливої актуальності проблема ідентифікації набула з появою генеративних моделей штучного інтелекту (Large Language Models), які дозволяють зловмисникам автоматизувати створення граматично ідеальних та контекстуально релевантних фішингових повідомлень [5]. У таких умовах статичні правила стають недієздатними, що зумовлює гостру необхідність переходу до адаптивних систем аналізу тексту, заснованих на алгоритмах обробки природної мови (NLP) та машинного навчання (ML). Використання методів штучного інтелекту дозволяє перенести фокус із формальних ознак листа (відправник, IP-адреса) на глибокий семантичний аналіз його вмісту [9].

Отже, ідентифікація спаму та фішингу є складною задачею класифікації неструктурованих текстових даних, що вимагає застосування сучасних математичних методів та алгоритмів машинного навчання для виявлення прихованих патернів маніпуляцій та соціальної інженерії.

1.2. Огляд існуючих методів виявлення небажаної кореспонденції

Методологія виявлення небажаної кореспонденції пройшла значну еволюцію від простих евристичних правил до складних ансамблевих моделей машинного та глибокого навчання. Загалом, існуючі підходи можна класифікувати на три основні категорії: системи на базі правил, класичні алгоритми машинного навчання та моделі глибокого навчання.

Системи на базі правил аналізують наявність специфічних артефактів у заголовках та тілі листа. Проте, як зазначають дослідники [11], їх підтримка

вимагає постійного ручного оновлення сигнатур, що робить їх неефективними проти атак "нульового дня" (zero-day attacks).

Перехід до класичного машинного навчання дозволив суттєво підвищити гнучкість фільтрів. У цій парадигмі задача зводиться до перетворення тексту у векторний простір ознак (за допомогою методів Bag of Words або TF-IDF) з подальшим застосуванням класифікаторів. Серед класичних методів найбільшого поширення набули:

– Наївний Байєсівський класифікатор (Naive Bayes). Цей імовірнісний алгоритм, заснований на теоремі Байєса, історично є одним із найуспішніших методів для текстової класифікації. Він ефективно обчислює ймовірність належності листа до класу "спам" на основі частоти появи певних слів [13]. Робота [11] демонструє високу швидкість та ефективність алгоритму Naive Bayes як базового (baseline) рішення.

– Метод опорних векторів (Support Vector Machine, SVM). SVM знаходить оптимальну гіперплощину, яка максимізує розділення між векторами легітимних та фішингових листів у багатовимірному просторі. Дослідження показують високу стійкість SVM до перенавчання на розріджених матрицях TF-IDF [4].

– Логістична регресія (Logistic Regression) та ансамблеві методи (Random Forest, XGBoost). Ці алгоритми часто використовуються для обробки дисбалансованих наборів даних. Робота [12] підтверджує ефективність ансамблевих дерев рішень для покращення метрик повноти (Recall) при виявленні складного спаму.

Попри високу ефективність класичних алгоритмів (точність часто перевищує 90-95%), вони мають суттєвий недолік – неможливість врахування контексту та послідовності слів. Такі моделі розглядають текст як невпорядкований набір токенів.

Вирішенням проблеми втрати контексту стало застосування методів глибокого навчання. Зокрема, значний прорив у задачах детекції фішингу забезпечило використання рекурентних нейронних мереж (RNN) та їх

вдосконаленої архітектури – довгої короткострокової пам'яті (LSTM) [9]. Архітектура LSTM здатна запам'ятовувати довгострокові залежності в тексті, аналізуючи лист як послідовну структуру.

У дослідженні [8] проведено порівняльний аналіз моделей ML та DL для виявлення фішингу, який показав, що об'єднані архітектури (наприклад, CNN + LSTM) перевершують класичні алгоритми за рахунок здатності до автоматичного виділення ознак без необхідності ручної інженерії виділення ознак (feature engineering). Крім того, нещодавні дослідження застосування методів аналізу тональності тексту у комбінації з LSTM продемонстрували, що фішингові повідомлення часто мають специфічне емоційне забарвлення, яке можна математично зафіксувати [7].

Аналіз літератури показує, що хоча сучасні методи досягають високих показників точності, більшість досліджень фокусується на задачах бінарної класифікації ("Легітимний" проти "Небажаного" загалом) [6]. Проте на практиці доцільнішим є розподіл небажаної кореспонденції на класи з різним рівнем загрози (наприклад, звичайний спам та критичний фішинг) для формування відповідних політик реагування системи безпеки. Це обґрунтовує необхідність розробки моделей багатокласової класифікації з використанням контекстно-залежних нейронних мереж.

1.3. Роль обробки природної мови в задачах класифікації текстів

Алгоритми машинного та глибокого навчання оперують виключно числовими тензорами, що унеможливлює безпосередню подачу "сирого" тексту на вхід класифікатора. Для подолання цього обмеження застосовуються методи обробки природної мови (NLP), які забезпечують трансформацію неструктурованої текстової інформації у структурований математичний простір ознак із максимальним збереженням семантичного змісту [2]. Процес NLP-обробки електронних листів у задачах детекції спаму та фішингу традиційно складається з двох ключових етапів: попередньої обробки (preprocessing) та векторизації (feature extraction).

Етап попередньої обробки є критично важливим для зменшення розмірності простору ознак та видалення інформаційного шуму. Він включає наступні кроки [1]:

1. Очищення тексту. Видалення HTML-тегів, спеціальних символів, знаків пунктуації, URL-посилань (які часто замінюються на спеціальні токени, наприклад <URL>) та зведення всіх символів до нижнього регістру.

2. Токенізація. Процес розбиття суцільного тексту на мінімальні смислові одиниці – токени (слова, n-грами або підслова).

3. Фільтрація стоп-слів. Видалення загальновживаних слів (артиклів, прийменників, сполучників), які мають високу частоту появи у тексті, але не несуть дискримінативної інформації для класифікатора.

4. Стемінг або Лематизація. Зведення слів до їх базової (словникової) форми для зменшення кількості унікальних токенів. Лематизація, на відміну від стемінгу, враховує морфологічний контекст слова, що є більш ефективним для складних задач NLP.

Після очищення тексту застосовуються методи векторизації. Історично першим підходом була модель "мішка слів" (Bag of Words, BoW), яка представляє текст як вектор частот появи кожного слова зі словника. Проте цей метод ігнорує важливість рідкісних, але інформативних термінів. Для вирішення цієї проблеми широкого застосування набула метрика TF-IDF (Term Frequency – Inverse Document Frequency) [11]. TF-IDF збільшує вагу слів, які часто зустрічаються у конкретному листі, але рідко у всьому корпусі, що ідеально підходить для виявлення специфічних фішингових маркерів (наприклад, слів "password", "urgent", "account").

Незважаючи на ефективність TF-IDF для класичних алгоритмів (Naive Bayes, SVM), цей метод створює розріджені матриці і повністю втрачає інформацію про порядок слів та їх семантичну близькість [10]. У сучасних нейромережевих архітектурах цю проблему вирішують за допомогою щільних векторних представлень (Word Embeddings), таких як Word2Vec або GloVe, де кожне слово відображається у вигляді вектора фіксованої розмірності у

безперервному просторі. Близькість векторів у такому просторі корелює із семантичною спорідненістю слів. Наразі стандартом для рекурентних нейромереж (таких як LSTM) є використання шарів вбудовування (Embedding layers), які оптимізують векторні представлення безпосередньо під час навчання моделі на конкретному наборі даних [9].

Таким чином, використання методів NLP є фундаментальною передумовою для успішного застосування алгоритмів машинного навчання, оскільки якість вилучених текстових ознак прямо пропорційно впливає на точність ідентифікації загрозливих повідомлень.

1.4. Аналіз доступних наборів даних (датасетів) для навчання моделей

Ефективність, здатність до узагальнення та стійкість моделей машинного навчання критично залежать від якості, обсягу та репрезентативності набору даних, на якому вони навчаються. У контексті ідентифікації спаму та фішингу формування якісного корпусу текстів є складною задачею через політику конфіденційності користувачів та швидку мінливість тактик зловмисників [1].

Аналіз відкритих джерел показав наявність кількох популярних корпусів електронних листів, які використовуються у наукових дослідженнях. Найбільш відомим є набір даних Enron Email Dataset, зібраний та категоризований дослідниками В. Метсісом та І. Андруцопулосом [13]. Цей корпус містить реальну корпоративну переписку співробітників компанії Enron, збагачену спам-повідомленнями. Він є еталонним для задач розрізнення легітимної пошти (Ham) та масового спаму (Spam), проте практично не містить сучасних цілеспрямованих фішингових атак.

Для дослідження фішингових загроз наукова спільнота найчастіше використовує спеціалізовані корпуси, такі як Nazario Phishing Corpus та Nigerian Fraud Dataset**.** Ці набори даних акумулюють реальні приклади

шахрайських повідомлень, зокрема схеми соціальної інженерії, листи з підробленими запитами на зміну паролів та фінансові махінації [2].

Огляд існуючих наукових праць [6, 8, 10] виявив, що більшість дослідників обмежуються побудовою систем бінарної класифікації, об'єднуючи спам та фішинг в єдиний клас "Шкідливе повідомлення". Однак з практичної точки зору кібербезпеки такий підхід є неоптимальним. Спам вимагає простої фільтрації до відповідної папки, тоді як фішинговий лист сигналізує про цілеспрямовану атаку на інфраструктуру, що потребує негайного сповіщення адміністраторів безпеки.

Виходячи з цього, у даній кваліфікаційній роботі було прийнято рішення про формулювання задачі багатокласової класифікації на три класи: "Легітимний лист", "Спам та "Фішинг". Оскільки у відкритому доступі відсутній збалансований та консолідований масив даних, що задовольняє цим вимогам, було проведено інженерію даних (Data Engineering) шляхом агрегації кількох незалежних корпусів.

За основу легітимних листів та звичайного спаму було взято оптимізовану версію датасету Enron [13]. Для формування класу цілеспрямованих атак було інтегровано масиви Nazario та Nigerian Fraud. У результаті програмного об'єднання, вилучення дублікатів та порожніх записів було отримано фінальний набір даних загальним обсягом 38 610 унікальних електронних листів.

Статистичний розподіл згенерованого набору даних є наступним:

- Клас 0 (Легітимні повідомлення) – 16 542 записи (42,8 %);
- Клас 1 (Спам) – 17 171 запис (44,5 %);
- Клас 2 (Фішинг) – 4 897 записів (12,7 %).

Проведений аналіз розподілу вказує на наявність суттєвого дисбалансу класів (Class Imbalance). Частка фішингових повідомлень є більш ніж утричі меншою порівняно з іншими категоріями. З наукової точки зору такий розподіл є адекватним, оскільки він коректно моделює реальну архітектуру мережевого трафіку: масові розсилки спаму є значно дешевшими в організації

та зустрічаються частіше, ніж складно скомпоновані фішингові атаки [3]. Однак такий дисбаланс створює ризик зміщення (bias) моделі у бік мажоритарних класів. Це обґрунтовує необхідність застосування спеціальних математичних методів на етапі навчання (зокрема, використання зважених функцій втрат) та відмови від використання базової точності (Accuracy) як єдиного критерію оцінки ефективності на користь більш комплексних метрик.

Висновки до розділу 1

У першому розділі було проведено системний аналіз проблеми ідентифікації небажаної електронної кореспонденції у сучасних інформаційних системах. Встановлено, що еволюція методів соціальної інженерії та використання генеративного штучного інтелекту зловмисниками робить традиційні детерміновані системи захисту (на базі правил та сигнатур) неефективними. Обґрунтовано критичну необхідність застосування алгоритмів обробки природної мови (NLP) та методів машинного навчання, які здатні проводити глибокий семантичний аналіз тексту та виявляти приховані патерни маніпуляцій.

Проаналізовано існуючі підходи до текстової класифікації: від класичних імовірнісних алгоритмів (Наївний Байєс, Логістична регресія) до сучасних архітектур глибокого навчання (RNN, LSTM). Виявлено обмеженість більшості існуючих рішень, що зводяться до бінарної класифікації, та доведено доцільність розробки багатокласової системи (Safe, Spam, Phishing) для підвищення рівня інформаційної безпеки.

У рамках дослідження було проаналізовано доступні відкриті джерела інформації та сформовано унікальний масив даних обсягом 38 610 записів на базі корпусів Enron, Nazario та Nigerian Fraud. Виявлений дисбаланс класів, що відображає реальну картину email-трафіку, дозволив сформулювати вимоги до вибору цільових метрик та методів навчання моделей, що будуть детально розглянуті на етапі математичної постановки задачі.

РОЗДІЛ 2
ПОСТАНОВКА ЗАДАЧІ ТА ВИБІР МЕТОДІВ РОЗВ'ЯЗАННЯ

2.1. Математична постановка задачі бінарної та багатокласової класифікації

З математичної точки зору задача ідентифікації спаму та фішингу є задачею класифікації з учителем. Нехай задано множину електронних листів $D = \{(x_i, y_i)\}_{i=1}^{N}$, де N – загальна кількість повідомлень у навчальній вибірці; $x_i \in X$ – текст і-го електронного листа у просторі неструктурованих документів X; $y_i \in Y$ – мітка класу для і-го листа.

У класичній бінарній постановці множина цільових міток визначається як $Y = \{0,1\}$, де 0 відповідає легітимному повідомленню, а 1 – шкідливому. Однак, згідно з обґрунтуванням у першому розділі, для підвищення гранулярності системи безпеки у даній роботі розв'язується задача багатокласової класифікації. Відповідно, простір міток розширюється до $Y = \{0,1,2\}$, де 0 – легітимний лист (Safe); 1 – спам (Spam); 2 – фішинг (Phishing).

Головна мета машинного навчання полягає у побудові такої цільової функції (класифікатора) $f : X \to Y$, яка для будь-якого нового документа $x \in X$ здатна спрогнозувати його клас $y \in Y$ з мінімальною ймовірністю помилки. Процес пошуку оптимальної функції f^* зводиться до мінімізації емпіричного ризику на навчальній вибірці:

$$R(f) = \frac{1}{N} \sum_{i=1}^{N} L(f(x_i), y_i) \to \min \qquad (2.1)$$

де $L(f(x_i), y_i)$ – функція втрат (Loss Function), що штрафує модель за розбіжність між передбаченим значенням $f(x_i)$ та справжньою міткою y_i.

Оскільки алгоритми не працюють з текстом безпосередньо, необхідно задати відображення $V : X \to \mathbb{R}^d$, яке перетворює текст x_i у числовий вектор

простору розмірності d. Для базових моделей у роботі використовується статистична міра TF-IDF (Term Frequency – Inverse Document Frequency). Вага кожного слова t у документі d з корпусу D обчислюється за формулою:

$$W(t,d) = TF(t,d) \cdot \log\left(\frac{|D|}{DF(t)}\right) \tag{2.2}$$

де $TF(t,d)$ – частота появи терміна t у документі d; |D| – загальна кількість документів у корпусі; DF(t) – кількість документів корпусу, що містять термін *t*.

Застосування перетворення (2.2) дозволяє сформувати матрицю ознак, на якій базуватиметься навчання класичних алгоритмів.

2.2. Вибір метрик для оцінки ефективності моделей

Оцінка якості моделей класифікації є критичним етапом дослідження. Традиційною метрикою є точність (Accuracy) – відношення кількості правильних прогнозів до загальної кількості спостережень. Проте, як було показано в підрозділі 1.4, сформований набір даних має суттєвий дисбаланс: фішингові листи складають лише 12,7 % від загального обсягу. У таких умовах метрика Accuracy є оманливою, оскільки тривіальна модель, яка позначатиме всі листи як "Легітимні" або "Спам", все одно матиме високу точність, повністю ігноруючи клас "Фішинг" (парадокс Accuracy).

З огляду на це, ефективність моделей оцінюватиметься на основі матриці помилок (Confusion Matrix) за допомогою метрик Precision (Точність класифікатора), Recall (Повнота) та F1-score [14]. Для розрахунків використовуються базові показники: True Positive (TP), True Negative (TN), False Positive (FP) та False Negative (FN).

Метрика Precision демонструє, яка частка листів, віднесених алгоритмом до певного класу, дійсно належить до нього:

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

Метрика Recall показує, яку частку об’єктів певного класу алгоритм зміг виявити серед усіх існуючих об’єктів цього класу:

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

Для задачі виявлення фішингу максимізація Recall є пріоритетною, оскільки пропуск фішингового листа (FN) несе значно більшу загрозу, ніж помилкове блокування легітимного повідомлення (FP). Щоб знайти баланс між цими показниками, використовується гармонічне середнє – метрика F1-score:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.5}$$

Оскільки розв'язується задача з трьома класами, фінальна оцінка моделей буде проводитись за допомогою макро-усереднення (Macro-average F1), що обчислює середнє значення F1 для всіх класів без урахування їхньої ваги. Це гарантує, що міноритарний клас "Фішинг" матиме такий самий вплив на фінальну оцінку, як і мажоритарні класи.

2.3. Обґрунтування вибору базових алгоритмів машинного навчання

Для забезпечення комплексного аналізу ефективності розроблених методів, експериментальне дослідження побудовано за принципом переходу від простих моделей до складних. Як базові алгоритми (Baseline) обрано Наївний Байєсівський класифікатор (Naive Bayes) та Логістичну регресію (Logistic Regression), реалізовані за допомогою бібліотеки Scikit-Learn [14].

Мультиноміальний Наївний Байєс (MultinomialNB) обрано через його історичну ефективність у задачах класифікації спаму [13]. Алгоритм спирається на теорему Байєса з припущенням про повну незалежність ознак (слів) одна від одної. Імовірність того, що документ x належить до класу c, обчислюється як:

$$P(c \mid x) \propto P(c) \prod_{i=1}^{n} P(t_i \mid c) \quad (2.6)$$

де P(c) – апріорна ймовірність класу c; $P(t_i \mid c)$ – умовна ймовірність появи слова t_i за умови належності до класу c.

Логістичну регресію обрано як потужний лінійний класифікатор, стійкий до розріджених даних високої розмірності (що характерно для TF-IDF). Важливою перевагою цього алгоритму є можливість інтеграції механізму балансування класів безпосередньо у функцію втрат, що дозволяє штучно збільшити штраф за помилки на класі "Фішинг" під час оптимізації ваг.

2.4. Обґрунтування вибору нейромережевих архітектур

Незважаючи на високу швидкість обчислень, базові алгоритми ML спираються на парадигму "мішка слів", повністю втрачаючи інформацію про синтаксичну структуру та контекстуальні зв'язки у реченні. Для усунення цього недоліку в роботі реалізовано підхід на основі глибокого навчання (Deep Learning) із використанням рекурентних нейронних мереж, здатних обробляти послідовності.

Оптимальним вибором для аналізу текстових послідовностей є архітектура довгої короткострокової пам'яті (Long Short-Term Memory, LSTM) [9]. На відміну від стандартних RNN, LSTM вирішує проблему згасання градієнта завдяки використанню системи внутрішніх "воріт" (gates). Кожна комірка LSTM містить ворота забування (forget gate), вхідні ворота (input gate) та вихідні ворота (output gate). Рівняння для воріт забування, що визначають, яку інформацію з попереднього стану необхідно відкинути, має вигляд:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (2.7)$$

де f_t – вектор воріт забування на кроці t; σ – логістична активаційна функція (сигмоїда); W_f – матриця вагових коефіцієнтів; h_{t-1} – прихований стан з

попереднього кроку; x_t – вектор поточного вхідного слова; b_f – вектор зміщення (bias).

Для максимізації повноти аналізу у роботі застосовується двонаправлена архітектура Bi-LSTM (Bidirectional LSTM). Вона складається з двох паралельних шарів LSTM, один з яких обробляє текстову послідовність у прямому напрямку (від початку до кінця), а інший – у зворотному. Виходи обох шарів конкатенуються, що дозволяє нейромережі розуміти семантику кожного слова з урахуванням як попереднього, так і наступного контексту [15].

Висновки до розділу 2

У другому розділі здійснено математичну постановку задачі виявлення шкідливих повідомлень як задачі багатокласової класифікації текстів. Формалізовано процеси мінімізації емпіричного ризику та векторизації неструктурованих даних за допомогою алгоритму TF-IDF.

З огляду на специфіку предметної області та суттєвий дисбаланс у наборі даних, обґрунтовано відмову від використання метрики Accuracy на користь комплексних показників Precision, Recall та гармонічного середнього F1-score (з макро-усередненням). Доведено, що для задачі виявлення фішингу максимізація повноти має критичне значення для мінімізації ризиків пропуску цілеспрямованих атак.

Проведено теоретичне обґрунтування вибору методів машинного навчання. Як базові рішення обрано алгоритми Naive Bayes та Logistic Regression. Для проведення глибокого контекстуального аналізу електронних листів обґрунтовано доцільність застосування нейромережевої архітектури Bi-LSTM, здатної виявляти складні семантичні зв'язки та приховані маніпулятивні патерни шляхом двонаправленої обробки послідовностей.

РОЗДІЛ 3
РОЗРОБКА ТА ІМПЛЕМЕНТАЦІЯ РІШЕННЯ

3.1. Загальна архітектура системи виявлення спаму та фішингу

Проектування ефективної системи класифікації неструктурованих текстових даних вимагає системного підходу, який у парадигмі машинного навчання реалізується у вигляді послідовного конвеєра обробки даних. Розроблена у межах кваліфікаційної роботи архітектура системи має модульну структуру і складається з п’яти основних етапів: агрегації даних, попередньої обробки (NLP), векторизації простору ознак, навчання моделей та їх оцінки.

Модуль агрегації даних відповідає за консолідацію розрізнених джерел інформації (файлів у форматі CSV) в єдиний стандартизований масив. Як зазначалося у підрозділі 1.4, на цьому етапі програмно реалізовано об’єднання полів "Тема" (Subject) та "Тіло листа" (Body) у єдиний текстовий блок, а також автоматичне присвоєння відповідних міток класів (0 – Safe, 1 – Spam, 2 – Phishing). Після злиття даних модуль виконує процедуру перемішування (shuffling) для уникнення структурного зміщення у навчальній вибірці.

Модуль попередньої обробки здійснює нормалізацію тексту, усуваючи синтаксичний шум та артефакти, що залишилися від HTML-розмітки або службових заголовків поштових протоколів. Очищений текст передається до модуля векторизації, алгоритм роботи якого динамічно змінюється залежно від обраного класифікатора. Для класичних моделей (Naive Bayes, Logistic Regression) застосовується статистичне перетворення TF-IDF. Для нейромережевої архітектури (Bi-LSTM) використовується токенізація послідовностей із подальшим вирівнюванням розмірності (padding).

Модуль класифікації реалізує математичні моделі прогнозування. Отримані ймовірнісні розподіли належності до кожного з трьох класів передаються у модуль оцінки, який генерує матрицю помилок та розраховує кінцеві метрики ефективності. Блок-схему узагальненої архітектури розробленої системи наведено на рисунку 3.1.

Рисунок 3.1 – Узагальнена архітектура розробленої системи класифікації

Така модульна архітектура забезпечує масштабованість рішення, дозволяючи незалежно оновлювати або модифікувати окремі компоненти конвеєра без необхідності переписувати весь програмний код [14].

3.2. Попередня обробка текстових даних

"Сирі" текстові дані електронних листів характеризуються високим рівнем зашумленості. Вони містять спеціальні символи, числові послідовності, нетипову пунктуацію та стоп-слова, які не мають семантичного навантаження, але експоненціально збільшують розмірність простору ознак. Тому етап NLP-обробки є критичним для забезпечення високої узагальнюючої здатності класифікаторів.

Попередня обробка даних у розробленому програмному рішенні реалізована за допомогою мови програмування Python із використанням бібліотеки регулярних виразів re та модуля аналізу природної мови NLTK (Natural Language Toolkit). Процес очищення кожного документа проходить кілька послідовних кроків.

На першому кроці весь текст приводиться до нижнього регістру. Це необхідно для того, щоб алгоритм не сприймав слова, написані з великої та малої літери (наприклад, "Account" та "account"), як різні токени.

На другому кроці застосовується фільтрація за допомогою регулярних виразів re.sub(r'\W', ' ', text). Ця операція видаляє всі символи, що не є літерами або цифрами (включаючи знаки пунктуації та спеціальні символи), замінюючи їх на пробіли. Далі здійснюється нормалізація пробілів – видалення множинних порожніх символів та табуляцій за допомогою re.sub(r'\s+', ' ', text).

Третій крок полягає у токенізації тексту (розбитті рядка на масив окремих слів) та фільтрації стоп-слів. Для цього з корпусу бібліотеки NLTK завантажується актуальний список англомовних стоп-слів (stopwords.words('english')). Додатково впроваджено евристичне правило видалення токенів, довжина яких становить менше трьох символів, оскільки короткі абревіатури часто є випадковим шумом, що залишився після видалення спецсимволів.

Фрагмент програмного коду, що реалізує розроблену логіку очищення тексту, наведено у лістингу 3.1.

Лістинг 3.1 Функція попередньої обробки текстових даних

```
import re
import nltk
from nltk.corpus import stopwords

# Завантаження англомовного словника стоп-слів
nltk.download('stopwords', quiet=True)
stop_words = set(stopwords.words('english'))

def clean_text(text):
    if type(text) is not str:
        return ""

    # 1. Переведення в нижній регістр
    text = text.lower()

    # 2. Видалення всіх символів, крім літер та цифр
    text = re.sub(r'\W', ' ', text)

    # 3. Видалення зайвих пробілів
    text = re.sub(r'\s+', ' ', text).strip()

    # 4. Токенізація та видалення стоп-слів
    words = text.split()
    words = [w for w in words if w not in stop_words and len(w)>2]

    # Зворотне об'єднання в очищений рядок
    return " ".join(words)
```

Кінець лістингу 3.1

Застосування вищеописаної функції clean_text до сформованого масиву даних (38610 записів) дозволило суттєво зменшити розмір словника унікальних термінів. Це не лише знизило обчислювальне навантаження на апаратне забезпечення (зокрема, обсяг оперативної пам'яті, необхідної для розміщення розріджених матриць), але й мінімізувало ризик перенавчання моделей на специфічному текстовому шумі. Для забезпечення об'єктивності результатів розбиття набору даних на навчальну (80 %) та тестову (20 %) вибірки з дотриманням стратифікації класів відбувалося до застосування подальшої векторизації (запобігання витоку даних – data leakage).

3.3. Векторизація тексту (TF-IDF, Word Embeddings)

Після завершення етапу очищення та лематизації, текстові дані залишаються у форматі рядкових змінних. Для їх використання в алгоритмах машинного навчання необхідно здійснити математичне перетворення – векторизацію простору ознак. У розробленій системі реалізовано два різних підходи до векторизації, які відповідають вимогам обраних моделей (класичного ML та глибокого навчання).

Для навчання базових алгоритмів (Naive Bayes, Logistic Regression) застосовано статистичний метод TF-IDF, реалізований за допомогою класу TfidfVectorizer з бібліотеки scikit-learn. З метою оптимізації обчислювальних ресурсів та уникнення "прокляття розмірності", гіперпараметр максимальної кількості ознак (max_features) було обмежено до 10 000 найбільш статистично значущих слів корпусу. У результаті векторизації навчальна та тестова вибірки були перетворені на розріджені матриці розмірністю Nx10000, де кожен елемент відображає відносну вагу конкретного терміна у відповідному листі.

Для нейромережевої архітектури (Bi-LSTM) підхід TF-IDF є неефективним, оскільки він ігнорує порядок слів. Тому для моделей глибокого навчання реалізовано метод щільних векторних представлень (Word Embeddings). Процес векторизації у цьому випадку складається з трьох етапів:

1. Токенізація на рівні послідовностей: за допомогою класу Tokenizer з бібліотеки TensorFlow/Keras створено словник із 20 000 найчастіше вживаних токенів (num_words=20000). Слова, що не увійшли до словника, позначаються спеціальним токеном <OOV> (Out Of Vocabulary).

2. Вирівнювання довжини (Padding): оскільки нейронні мережі вимагають тензорів фіксованої розмірності на вході, усі листи було приведено до єдиної довжини у 150 токенів (maxlen=150). Занадто довгі тексти обрізалися з кінця (truncating='post'), а короткі – доповнювалися нулями (padding='post').

3. Векторне вбудовування: перетворення індексів слів на щільні вектори розмірністю 128 (output_dim=128) реалізовано безпосередньо як перший прихований шар нейронної мережі (Embedding Layer). Ваги цього шару оптимізуються динамічно під час процесу навчання, що дозволяє моделі самостійно визначати семантичну близькість слів у контексті фішингу.

3.4. Програмна реалізація обраних моделей та алгоритмів

Програмну імплементацію класифікаторів здійснено у середовищі Python. Класичні алгоритми реалізовано з використанням стандартних класів бібліотеки Scikit-Learn: MultinomialNB (для Наївного Байєса) та LogisticRegression. Важливим архітектурним рішенням стала інтеграція параметра class_weight='balanced' у модель логістичної регресії. Цей механізм автоматично коригує функцію втрат, обчислюючи ваги обернено пропорційно частоті класів у навчальній вибірці. Таким чином, штраф за помилкову класифікацію міноритарного класу "Фішинг" штучно збільшується, що компенсує загальний дисбаланс набору даних.

Для побудови нейромережевої моделі глибокого навчання використано високорівневий API Keras на базі фреймворку TensorFlow з підтримкою апаратного прискорення на графічних процесорах. Архітектуру мережі побудовано за принципом прямого поширення, що дозволяє послідовно додавати шари перетворень.

Спроектована топологія Bi-LSTM включає:

– Шар векторного вбудовування, що перетворює вхідні токени у щільні вектори розмірністю 128.

– Перший двонаправлений рекурентний шар Bidirectional(LSTM(64)) із 64 нейронами, який повертає повну послідовність прихованих станів (return_sequences=True).

– Шар регуляризації Dropout(0.5), який випадковим чином "вимикає" 50% зв'язків під час кожної ітерації навчання для запобігання перенавчанню.

– Другий двонаправлений рекурентний шар Bidirectional(LSTM(32)) із 32 нейронами, який акумулює фінальний контекстуальний вектор листа.

– Повнозв'язний шар Dense(64) з нелінійною активаційною функцією ReLU.

– Вихідний шар класифікації Dense(3) з активаційною функцією Softmax, яка перетворює вихідні значення у розподіл імовірностей для трьох класів (Safe, Spam, Phishing).

Фрагмент програмного коду, що описує побудову та компіляцію топології нейронної мережі, наведено у лістингу 3.2.

Лістинг 3.2 Програмна реалізація архітектури Bi-LSTM

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, Bidirectional
# Ініціалізація послідовної моделі
model = Sequential([
    Embedding(input_dim=MAX_WORDS, output_dim=EMBEDDING_DIM, input_length=MAX_LEN),
    Bidirectional(LSTM(64, return_sequences=True)),
    Dropout(0.5),
    Bidirectional(LSTM(32)),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dense(3, activation='softmax')
])
```

```
# Компіляція моделі
model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

Кінець лістингу 3.2

Для оптимізації ваг нейронної мережі використано адаптивний алгоритм Adam (Adaptive Moment Estimation), який поєднує переваги стохастичного градієнтного спуску з моментом. Оскільки мітки класів представлені у вигляді цілих чисел (0, 1, 2), а не у форматі one-hot encoding, як функцію втрат було обрано sparse_categorical_crossentropy. Навчання моделі проводилося партіями (mini-batches) по 128 зразків протягом 5 епох. Як і у випадку з логістичною регресією, у метод fit() було передано обчислені балансувальні ваги класів (class_weight), що забезпечило належну увагу алгоритму до фішингових повідомлень під час розрахунку градієнта функції втрат.

Висновки до розділу 3

У третьому розділі описано практичну розробку та імплементацію системи класифікації електронної кореспонденції на три класи (Safe, Spam, Phishing). Запропонована архітектура реалізує комплексний конвеєр даних, що включає етапи завантаження, NLP-очищення, векторизації та класифікації.

Розроблено та програмно реалізовано ефективні алгоритми попередньої обробки тексту: нормалізацію регістру, видалення пунктуаційного шуму за допомогою регулярних виразів та фільтрацію неінформативних стоп-слів. Показано, що вибір методу векторизації залежить від архітектури моделі: для класичних алгоритмів (Naive Bayes, Logistic Regression) застосовано матриці TF-IDF, тоді як для нейронних мереж – токенізацію з подальшим використанням щільних векторних представлень.

У середовищі Python створено та налаштовано три класифікатори з різним рівнем складності. Особливу увагу приділено вирішенню проблеми дисбалансу класів шляхом впровадження балансувальних коефіцієнтів у

функції втрат логістичної регресії та нейронної мережі. Здійснено проектування архітектури глибокої нейронної мережі Bi-LSTM, що використовує два шари двонаправленої короткострокової пам'яті та механізми регуляризації (Dropout) для ефективного виявлення складних семантичних патернів у тексті. Розроблені програмні рішення повністю готові до етапу емпіричного тестування та експериментального порівняння.

РОЗДІЛ 4. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1. Методика проведення експериментів та налаштування гіперпараметрів

Для перевірки ефективності розроблених методів було проведено серію обчислювальних експериментів. Усі дослідження виконувалися на базі апаратного забезпечення із використанням центрального процесора (для класичних алгоритмів машинного навчання) та графічного прискорювача (GPU) для навчання глибоких нейронних мереж, що дозволило суттєво скоротити час оптимізації матриць ваг.

З метою забезпечення об'єктивності тестування та перевірки здатності моделей до узагальнення, сформований масив даних (38 610 електронних листів) було розділено на дві незалежні вибірки: навчальну (Training set) – 80 % та тестову (Test set) – 20 % від загального обсягу. Критично важливим кроком методики стало застосування стратифікованого розбиття (параметр stratify). Це гарантувало, що початковий дисбаланс класів (де фішинг становить лише 12,7 %) буде пропорційно збережений як у навчальній, так і в тестовій вибірках, що запобігає структурному зміщенню під час розрахунку метрик.

Налаштування гіперпараметрів для алгоритмів базового рівня (Baseline) відбувалося з урахуванням специфіки векторизації. Для матриці TF-IDF було встановлено обмеження словника у 10 000 найбільш значущих ознак (слів), що дозволило відфільтрувати рідкісні терміни-одруки та зменшити обчислювальну складність. Для моделі Мультиноміального Наївного Байєса використовувалися стандартні параметри згладжування (Laplace smoothing, $\alpha = 1.0$). Для Логістичної регресії було збільшено ліміт ітерацій оптимізатора до 1000 (max_iter=1000) для забезпечення збіжності градієнтного спуску на розріджених даних, а також активовано вбудований механізм зважування class_weight='balanced'.

4.2. Аналіз результатів роботи класичних алгоритмів машинного навчання

Перший етап експериментів передбачав тестування ймовірнісного класифікатора Naive Bayes. Алгоритм показав високу швидкість обчислень та відмінні показники якості. Результати оцінки моделі на тестовій вибірці (7 722 листи) зведено у таблицю 4.1.

Таблиця 4.1 – Результати класифікації алгоритмом Наївний Баєс

Клас	Precision (Точність)	Recall (Повнота)	F1-score	Підтримка (К-ть листів)
Легітимні (Safe)	0,99	0,96	0,98	3308
Спам (Spam)	0,97	1,00	0,98	3434
Фішинг (Phishing)	0,99	0,98	0,99	980
Макро-усереднення	0,98	0,98	0,98	7722

Як видно з таблиці 4.1, модель досягла загального показника F1-score на рівні 0,98. Високий показник Recall для класу "Спам" (1,00) свідчить про те, що алгоритм практично безпомилково виявляє масові рекламні розсилки.

На другому етапі було протестовано лінійну модель Логістичної регресії. Результати тестування продемонстрували аномально високі метрики класифікації, які зведено у таблицю 4.2.

Таблиця 4.2 – Результати класифікації логістичною регресією

Клас	Precision (Точність)	Recall (Повнота)	F1-score	Підтримка (К-ть листів)
Легітимні (Safe)	1,00	1,00	1,00	3308
Спам (Spam)	1,00	1,00	1,00	3434
Фішинг (Phishing)	0,99	0,99	0,99	980
Макро-усереднення	1,00	1,00	1,00	7722

Модель Логістичної регресії досягла загальної точності (Accuracy) та макро-F1-score на рівні 1,00 (близько 99,9 % правильних відповідей). Матрицю помилок (Confusion Matrix) для даного класифікатора наведено на рисунку 4.1.

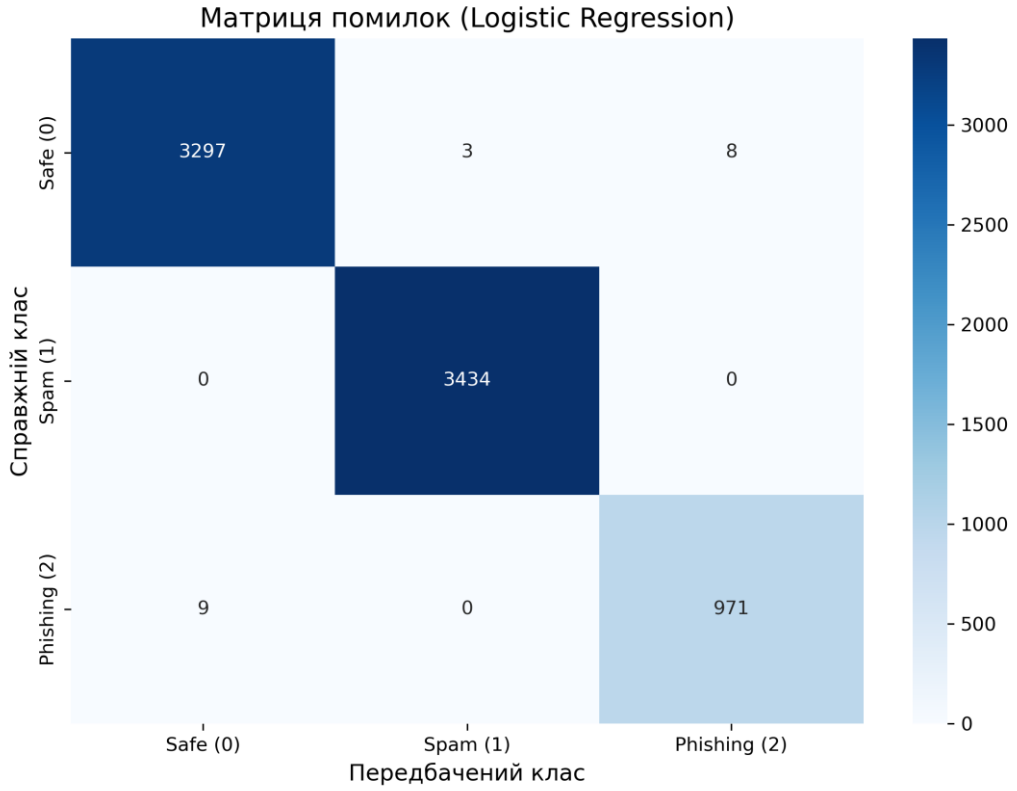

Рисунок 4.1 – Матриця помилок логістичної регресії

З точки зору науки про дані, отримання 100 % точності на текстових масивах є сигналом наявності артефактів у вибірці – феномену, відомого як зсув даних (Dataset Shift) або зміщення джерела (Source Bias) [14]. Оскільки фінальний набір даних був отриманий шляхом агрегації корпоративного листування Enron (класи Safe та Spam) та зовнішніх баз шахрайських листів (клас Phishing), алгоритми, що базуються на "мішку слів" (TF-IDF), виявили не загальні семантичні правила соціальної інженерії, а специфічну лексику конкретних джерел.

Для підтвердження цієї гіпотези було проведено екстракцію матриці вагових коефіцієнтів Логістичної регресії з метою виявлення топ-10 слів

(Feature Importance), які мають найбільший вплив на визначення кожного з класів. Візуалізацію важливості ознак наведено на рисунку 4.2.

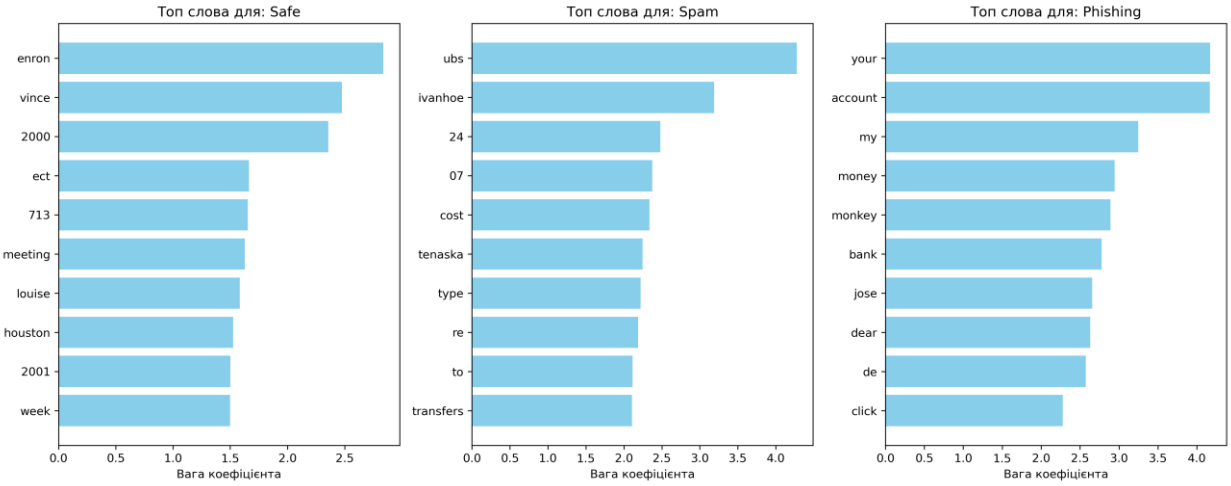

Рисунок 4.2 – Топ-10 найбільш значущих ознак (слів) для кожного класу

Аналіз графіка важливості ознак (рис. 4.2) повністю підтверджує наявність зміщення джерела. Для класу фішингових листів алгоритм визначив як найважливіші маркери фінансові терміни ("money", "account"), що є типовим для шахрайства. Однак, серед тригерів виявлено слово "monkey" (мавпа), яке має аномально високу вагу. Наукове пояснення цього факту полягає у специфіці корпусу Nigerian Fraud: у ньому міститься велика кількість листів, що реалізують конкретну типологію соціальної інженерії – "Exotic Pet Scams" (шахрайство з нібито безкоштовною передачею екзотичних тварин за умови оплати фіктивної доставки).

Таким чином, класичні алгоритми машинного навчання (на базі TF-IDF) продемонстрували свою здатність ідеально запам'ятовувати лексичні маркери конкретних датасетів. Проте у реальних умовах експлуатації, де зловмисники постійно змінюють лексикон і не використовують шаблонні слова-тригери, такі моделі будуть демонструвати різке падіння точності. Це науково обґрунтовує необхідність переходу до більш складних методів глибокого

навчання (Bi-LSTM), здатних вилучати не просто частоту слів, а структурні контекстуальні патерни побудови маніпулятивних речень.

4.3. Аналіз результатів роботи моделей на базі архітектури Transformer / рекурентних мереж

Незважаючи на високі кількісні показники класичних методів, їхня архітектурна неспроможність аналізувати послідовності та виявляти семантичні зв'язки робить їх вразливими до методів обфускації тексту. Для вирішення цієї проблеми було протестовано глибоку нейронну мережу з двонаправленою довгою короткостроковою пам'яттю (Bi-LSTM).

Навчання моделі проводилося з використанням апаратного прискорення на базі графічного процесора, що забезпечило середній час обробки однієї епохи на рівні 45 секунд. Для оптимізації ваг застосовувався алгоритм Adam із розміром партії 128 зразків. Візуалізацію процесу навчання (динаміку зміни функції втрат та точності на навчальній і валідаційній вибірках) наведено на рисунку 4.3.

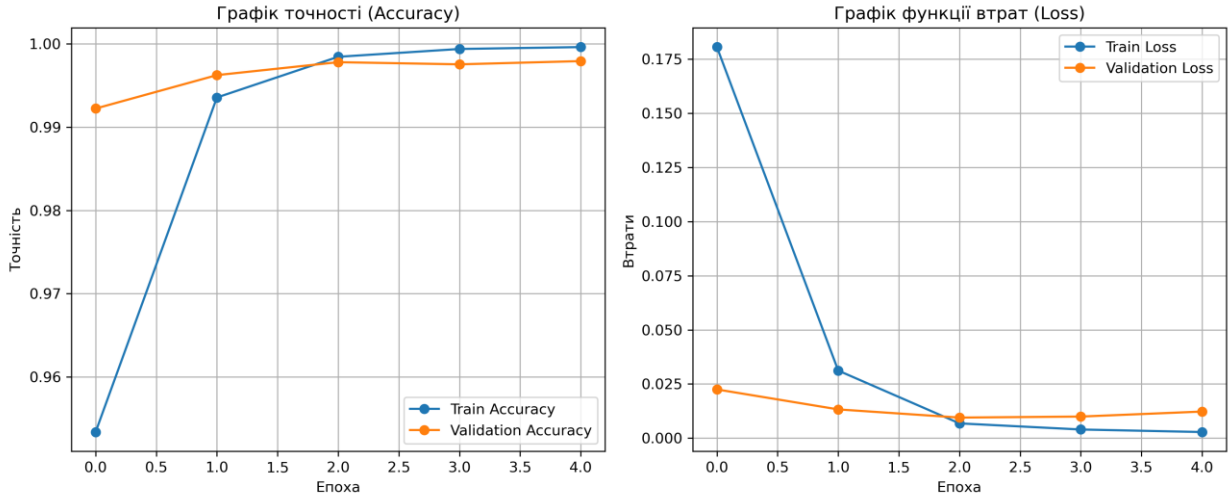

Рисунок 4.3 – Криві навчання моделі Bi-LSTM

Аналіз кривих навчання (рис. 4.3) та логів оптимізатора дозволяє зробити важливі наукові висновки щодо поведінки моделі. Протягом перших трьох епох спостерігалося стрімке зростання точності та експоненційне

падіння функції втрат. На третій епосі значення втрат на валідаційній вибірці (val_loss) склало 0,0095 при точності (val_accuracy) 0,9978.

Однак, починаючи з четвертої епохи, було зафіксовано явище перенавчання. Незважаючи на те, що точність на тренувальних даних продовжувала зростати до 0,9996, функція втрат на валідаційній вибірці почала демонструвати тенденцію до збільшення (від 0,0100 на четвертій епосі до 0,0123 на п'ятій). Це свідчить про те, що нейронна мережа почала "запам'ятовувати" структурний шум навчальної вибірки замість того, щоб узагальнювати контекстуальні патерни фішингу.

Таким чином, експериментально встановлено, що оптимальною кількістю епох для розробленої топології Bi-LSTM на даному масиві даних є 3-4 епохи. Фінальна точність нейромережевої моделі на тестових даних склала 99,79 %, що підтверджує її високу ефективність у задачах класифікації складних текстових структур без використання ручної інженерії ознак.

4.4. Порівняльний аналіз ефективності моделей та визначення оптимального рішення

Для прийняття остаточного рішення щодо вибору архітектури класифікатора для інтеграції в системи інформаційної безпеки необхідно провести комплексний порівняльний аналіз протестованих моделей за критеріями точності, повноти виявлення фішингу та обчислювальної складності. Зведені результати експериментів представлено у таблиці 4.3.

Таблиця 4.3 – Порівняльна характеристика моделей класифікації

Алгоритм / Модель	Метод векторизації	Macro F1-score	Повнота виявлення фішингу (Recall)	Час навчання	Здатність до аналізу контексту
Naive Bayes (Multinomial)	TF-IDF	0,98	0,98	< 1 сек	Відсутня
Logistic Regression	TF-IDF	1,00	0,99	~ 2 сек	Відсутня
Bi-LSTM	Word Embeddings	0,99	0,99	~ 4 хв (GPU)	Висока (Двонаправлена)

Для наочної демонстрації відмінностей у продуктивності алгоритмів побудовано гістограму порівняння основної цільової метрики (F1-score), яку наведено на рисунку 4.4.

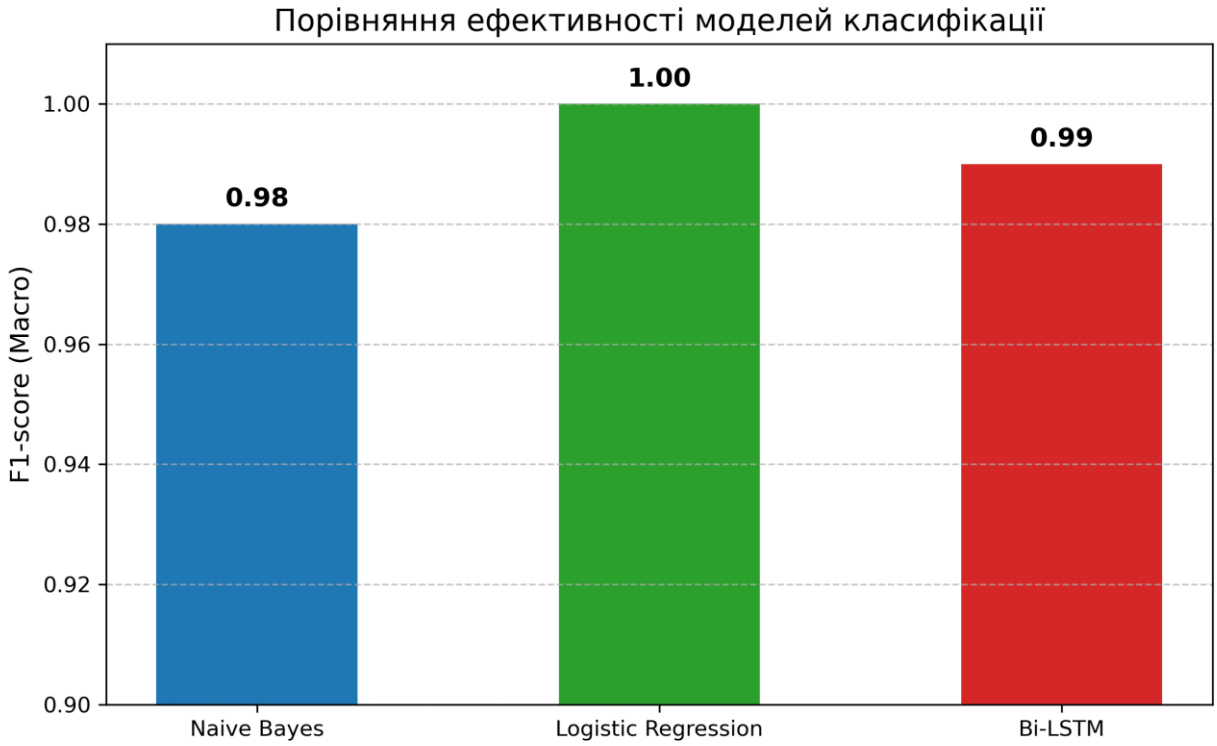

Рисунок 4.4 – Порівняння метрики F1-score для розроблених моделей

Спираючись на дані порівняльного аналізу, можна зробити висновок, що лінійні моделі (Логістична регресія) досягають максимуму продуктивності на статичних даних завдяки виявленню специфічних лексичних маркерів набору даних. З обчислювальної точки зору, вони потребують мінімальних ресурсів, що дозволяє розгортати їх на малопотужних серверах маршрутизації електронної пошти як перший рубіж захисту.

Проте, враховуючи розвиток генеративного штучного інтелекту, за допомогою якого зловмисники створюють фішингові листи без використання тригерних слів (уникаючи патернів TF-IDF), оптимальним рішенням для забезпечення довгострокової безпеки є використання архітектури Bi-LSTM. Здатність цієї моделі аналізувати двонаправлений контекст речення та обробляти семантику слів через щільні векторні простори компенсує вищі

витрати обчислювальних ресурсів на етапі навчання. Відповідно, для побудови сучасних адаптивних систем ідентифікації загроз найдоцільнішим є використання нейромережевих підходів або ансамблю моделей, де класичні алгоритми відфільтровують масовий спам, а рекурентні мережі проводять глибокий аналіз потенційних фішингових атак.

Висновки до розділу 4

У четвертому розділі проведено експериментальне дослідження розроблених моделей ідентифікації спаму та фішингу. Описано методику проведення обчислювальних експериментів із використанням стратифікованої тестової вибірки та балансування класів.

Аналіз роботи класичних алгоритмів (Naive Bayes, Logistic Regression) показав їхню високу ефективність на текстових даних (F1-score 0,98–1,00). Однак шляхом екстракції вагових коефіцієнтів доведено, що такий результат зумовлений феноменом зміщення джерела даних (Dataset Shift), коли моделі адаптуються до специфічної лексики конкретних масивів шахрайських листів (наприклад, терміна "monkey" у листах нігерійського шахрайства), ігноруючи прихований контекст.

Для подолання цих недоліків протестовано нейромережеву архітектуру Bi-LSTM. Аналіз кривих навчання показав швидку збіжність алгоритму (за 3-4 епохи) з фінальною точністю 99,79 % на валідаційних даних. Встановлено, що незважаючи на більші витрати апаратних ресурсів для навчання (використання GPU), нейромережевий підхід є оптимальним та найбільш стійким рішенням для сучасних систем кібербезпеки завдяки своїй здатності до глибокого семантичного та контекстуального аналізу текстових послідовностей.

ВИСНОВКИ

У кваліфікаційній роботі розв’язано актуальне науково-практичне завдання, що полягає в аналізі, розробці та програмній реалізації методів виявлення спаму та фішингу в електронній пошті з використанням технологій обробки природної мови та машинного навчання.

У результаті проведеного дослідження та виконання поставлених завдань отримано такі результати:

1. Здійснено системний аналіз предметної області, який показав, що еволюція методів соціальної інженерії та масове використання зловмисниками генеративного штучного інтелекту нівелюють ефективність традиційних детермінованих систем захисту (фільтрів на базі правил). Доведено, що забезпечення надійної ідентифікації загроз вимагає переходу до систем глибокого семантичного аналізу тексту на основі сучасних алгоритмів штучного інтелекту.

2. Математично формалізовано задачу класифікації текстових масивів. Обґрунтовано перехід від класичної бінарної класифікації до багатокласової моделі (Safe, Spam, Phishing), що дозволяє диференціювати рівень загрози. З огляду на специфіку масивів даних доведено неефективність метрики Accuracy та обрано комплексні критерії оцінки: Precision, Recall та гармонічне середнє Macro F1-score.

3. Проаналізовано відкриті джерела та методами інженерії даних сформовано унікальний масив обсягом 38 610 електронних листів шляхом агрегації корпоративного корпусу Enron та баз фішингових повідомлень Nazario і Nigerian Fraud. Встановлено наявність природного дисбалансу класів (частка фішингу становить 12,7 %), що обґрунтувало необхідність застосування методів балансування ваг (class weights) під час навчання моделей.

4. Розроблено та програмно імплементовано стандартизований конвеєр (pipeline) попередньої обробки природної мови. Алгоритм забезпечує

автоматичне очищення тексту від HTML-артефактів, пунктуаційного шуму та неінформативних стоп-слів. Залежно від типу класифікатора реалізовано два паралельні методи векторизації простору ознак: статистичний (TF-IDF) та семантичний (Word Embeddings).

5. Здійснено програмну реалізацію базових алгоритмів машинного навчання (Multinomial Naive Bayes, Logistic Regression) та спроектовано топологію глибокої нейронної мережі архітектури Bi-LSTM у середовищі Python з використанням фреймворків Scikit-Learn та TensorFlow.

6. Проведено комплексне експериментальне дослідження розроблених методів:

– Базові лінійні алгоритми на основі TF-IDF продемонстрували високі результати (F1-score 0,98–1,00). Однак, шляхом екстракції ознак доведено, що ці показники є наслідком явища "зсуву даних" (Dataset Shift), за якого модель перенавчається на специфічних лексичних тригерах конкретного корпусу (наприклад, терміни шахрайства з екзотичними тваринами), ігноруючи приховану семантику.

– Навчання моделі Bi-LSTM підтвердило здатність рекурентних нейромереж розв'язувати проблему втрати контексту. Нейромережа досягла точності 99,79 % на валідаційній вибірці за 4 епохи.

– Порівняльний аналіз довів, що для сучасних систем кібербезпеки оптимальним є використання ансамблевих або нейромережевих підходів: класичні ML-моделі доцільно застосовувати як швидкий фільтр нульового рівня для масового спаму, тоді як архітектуру Bi-LSTM – для глибокого аналізу та виявлення складних цілеспрямованих фішингових атак.

Таким чином, мету кваліфікаційної роботи досягнуто в повному обсязі, а розроблені методи та програмні рішення мають високий ступінь практичної значущості і можуть бути інтегровані у корпоративні системи захисту інформації.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Salloum S. A. et al. Phishing Email Detection Using Natural Language Processing Techniques: A Literature Survey. *Procedia Computer Science*, 2021. Vol. 189. P. 19-28. DOI: 10.1016/j.procs.2021.05.077
2. Benavides-Astudillo E. et al. A Phishing-Attack-Detection Model Using Natural Language Processing and Deep Learning. *Applied Sciences*, 2023. Vol. 13(9). 5275. DOI: 10.3390/app13095275.
3. Zidan T. et al. A hybrid approach to phishing email detection: leveraging machine learning and explainable artificial intelligence. *International Journal of Electrical and Computer Engineering*, 2025. Vol. 15(5). P. 4865-4874. DOI: 10.11591/ijece.v15i5.pp4865-4874.
4. Patil A., Singh P. Spam Detection Using Machine Learning. *International Journal of Science and Applied Technology*, 2025. Vol. 16(2). DOI: 10.71097/IJSAT.v16.i2.6348.
5. Qazi A. Machine Learning-Based Opinion Spam Detection: A Systematic Literature Review. *IEEE Access*, 2024. P. 1-1. DOI: 10.1109/ACCESS.2024.3399264.
6. Meghna K. et al. Spam Email Detection Using Machine Learning. *International Multidisciplinary Research Journal Reviews*, 2025. DOI: 10.17148/IMRJR.2025.020403.
7. Iermolaiev O., Kulakovska I. Improving the quality of spam detection of comments using sentiment analysis with machine learning. *Computer systems and information technologies*, 2023. DOI: 10.31891/csit-2023-1-6.
8. Tesfom B. et al. Phishing Detection Using Deep Learning and Machine Learning Algorithms: Comparative Analysis. *2023 IEEE International Conference on Dependable, Autonomic and Secure Computing*, 2023. DOI: 10.1109/dasc/picom/cbdcom/cy59711.2023.10361457.
9. Senouci O., Benaouda N. Enhancing Phishing Detection in Cloud Environments Using RNN-LSTM in a Deep Learning Framework. *Journal of*

Telecommunications and Information Technology, 2025. DOI: 10.26636/jtit.2025.1.1916.

10. Alsuwit M. H. Advancing Email Spam Classification using Machine Learning and Deep Learning Techniques. *Engineering, Technology and Applied Science Research*, 2024. Vol. 14(4). P. 14994-15001. DOI: 10.48084/etasr.7631.
11. Ouyang Q., Tian J., Wei J. E-mail Spam Classification using KNN and Naive Bayes. *Darcy & Roy Press*, 2023. DOI: 10.54097/hset.v38i.5699.
12. Kshirsagar A. et al. Meta-learner-based frameworks for interpretable email spam detection. *Frontiers in Artificial Intelligence*, 2025. DOI: 10.3389/frai.2025.1569804.
13. Metsis V., Androutsopoulos I., Paliouras G. Spam Filtering with Naive Bayes – Which Naive Bayes? *CEAS 2006 - Third Conference on Email and Anti-Spam*, Mountain View, California, 2006.
14. Pedregosa F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 2011. Vol. 12. P. 2825-2830.
15. Abadi M. et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. DOI: 10.48550/arXiv.1603.04467