НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

До захисту допущено
Завідувач кафедри

_____ **Дмитро ЛАНДЕ**
(підпис)

«_____»_____2023 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи, технології та математичні методи кібербезпеки»
спеціальності 125 «Кібербезпека»

на тему: Використання машинного навчання для виявлення спаму

Виконав (-ла): здобувач вищої освіти **IV** курсу, групи ФБ-92

Сьомченко Дмитро Вікторович _____ (підпис)

Керівник к.т.н., доцент Гальчинський Леонід Юрійович _____ (підпис)

Рецензент Інженер Samsung Research and Development Institute Ukraine Мандренко Анна Євгенівна _____ (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Здобувач вищої освіти _____ (підпис)

Київ – 2023 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 125 «Кібербезпека»
Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Дмитро ЛАНДЕ
(підпис)

«___»_____2023 р.

ЗАВДАННЯ
на дипломну роботу здобувачу вищої освіти

Сьомченку Дмитру Вікторовичу

1. Тема роботи: «Використання машинного навчання для виявлення спаму»,

керівник роботи: к.т.н., доцент Гальчинський Леонід Юрійович

затверджені наказом по університету від 26 травня 2023 р. № 2028-с

2. Термін подання здобувачем вищої освіти роботи: 15 червня 2023 р.

3. Вихідні дані до роботи: попередні дослідження використання машинного навчання для виявлення спаму

4. Зміст роботи: аналіз предметної області, вибір та попередня обробка даних, розробка програми мовою Python для класифікації спаму методами машинного навчання, експериментальна перевірка розробленої програми

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація

6. Дата видачі завдання 01 листопада 2022 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Формування теми роботи, визначення завдання та мети	16.01.2023-31.01.2023	виконано
2	Узгодження структури дипломної роботи з керівником	01.02.2023-15.02.2023	виконано
3	Робота над першим розділом; опрацювання літературних джерел	15.02.2023-19.03.2023	виконано
4	Робота над другим розділом; дослідження машинного навчання	20.03.2023-16.04.2023	виконано
5	Проходження переддипломної практики	17.04.2023-21.05.2023	виконано
6	Робота над третім розділом; розробка програми на основі машинного навчання для класифікації спаму	17.04.2023-31.05.2023	виконано
7	Оформлення текстової та графічної частин дипломної роботи	22.05.2023-04.06.2023	виконано
8	Написання загальних висновків	05.06.2023-07.06.2023	виконано
9	Підготовка до передзахисту дипломної роботи; оформлення презентації	07.06.2023-14.06.2023	виконано
10	Передзахист дипломної роботи	15.06.2023	виконано
11	Впровадження рекомендацій; підготовка до захисту дипломної роботи	16.06.2023-21.06.2023	виконано
12	Захист дипломної роботи	22.06.2023	

Здобувач вищої освіти _____ Дмитро СЬОМЧЕНКО
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ Леонід ГАЛЬЧИНСЬКИЙ
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Дипломна робота містить: 66 сторінок, 33 ілюстрації, 10 таблиць, 1 додаток, 43 джерела літератури.

Метою роботи є визначення можливостей використання методів машинного навчання для виявлення спаму в комунікаційних каналах.

Об’єктом дослідження є процес виявлення спаму в електронній пошті та інших комунікаційних каналах.

Предметом дослідження є методи машинного навчання, які можуть бути застосовані для виявлення спаму. Робота включає в себе аналіз існуючих методів та їхньої ефективності.

Методи дослідження можуть включати аналіз літературних джерел, дослідження та реалізацію алгоритмів машинного навчання, проведення експериментів та оцінку результатів.

Робота містить опис процесу використання машинного навчання для виявлення спаму в електронних комунікаціях. Вона розглядає процес виявлення спаму, визначає ключові особливості спаму та описує, які існують методи протидії спаму. Також розроблено програму мовою Python для класифікації електронних та текстових повідомлень.

Ключові слова: спам, машинне навчання, класифікація.

ABSTRACT

Thesis contains: 66 pages, 33 illustrations, 10 tables, 1 appendix, 43 references.

The aim of the work is to study the possibilities of using machine learning methods to detect spam in communication channels.

The object of research is the process of detecting spam in email and other communication channels.

The subject of the study is machine learning methods that can be used to detect spam. The work includes an analysis of existing methods and their effectiveness.

Research methods may include literature analysis, research and implementation of machine learning algorithms, experiments, and evaluation of results.

The paper describes the process of using machine learning to detect spam in electronic communications. It examines the process of spam detection, identifies key features of spam, and describes what methods exist to counteract spam. It also develops a Python program for classifying email and text messages.

Keywords: spam, machine learning, classification.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SPAM — небажані електронні повідомлення

HAM — електронні повідомлення, що не є спамом, антонім до SPAM

NB — наївний Байєс (англ. Naïve Bayes)

SVM — метод опорних векторів (англ. Support Vector Machine)

KNN — k найближчих сусідів (k-Nearest Neighbors)

RF — випадковий ліс (англ. Random Forest)

BERT — двоспрямовані кодувальні представлення з трансформерів (англ. Bidirectional Encoder Representations from Transformers)

GPT — породжувальний попередньо тренований трансформер (англ. Generative Pre-trained Transformer)

NLP — обробка природної мови (англ. Natural Language Processing)

TP — істинно позитивний (англ. True Positive)

TN — істинно негативний (англ. True Negative)

FP — хибно позитивний (англ. False Positive)

FN — хибно негативний (англ. False Negative)

ВСТУП

Актуальність роботи полягає в тому, що проблема спаму є досить поширеною в сучасному світі. Останні роки кількість спам-повідомлень в електронній пошті, месенджерах та інших комунікаційних каналах залишається високою, що призводить до надмірного споживання часу та ресурсів користувачів на їх фільтрацію. Крім того, спам може бути ризиком для інформаційної безпеки, забруднювати бази даних та сповільнювати роботу комунікаційних систем. Застосування методів машинного навчання може допомогти ефективно розв’язати ці проблеми та покращити якість взаємодії користувачів.

Метою роботи є визначення можливостей використання методів машинного навчання для виявлення спаму в комунікаційних каналах.

Об’єктом дослідження є процес виявлення спаму в електронній пошті та інших комунікаційних каналах.

Предметом дослідження є методи машинного навчання, які можуть бути застосовані для виявлення спаму. Робота включає в себе аналіз існуючих методів та їхньої ефективності.

Для досягнення мети роботи були сформовані наступні завдання:

- дослідити поняття спаму та існуючі методи його виявлення, зокрема розглянути машинне навчання для класифікації спаму;
- запропонувати рішення, що використовує обраний набір даних та застосовує моделі машинного навчання для класифікації спаму, та експериментально перевірити його ефективність;
- провести налаштування гіперпараметрів використаних моделей для оптимізації їхньої ефективності та дослідити вплив різних конфігурацій і параметрів на точність класифікації спаму;
- проаналізувати результати роботи застосунку та порівняти ефективність різних моделей машинного навчання, оцінити практичне значення одержаних результатів та надати рекомендації щодо подальших досліджень.

Методи дослідження можуть включати аналіз літературних джерел, дослідження та реалізацію алгоритмів машинного навчання, проведення експериментів з використанням тестових даних та оцінку результатів.

Практичне значення одержаних результатів

Отримані результати можуть мати практичне значення для підприємств та користувачів, які стикаються з проблемою спаму. Автоматичне виявлення спаму може допомогти зменшити кількість небажаних повідомлень, підвищити продуктивність роботи та забезпечити безпеку комунікацій. Крім того, результати дослідження сприяють зростанню обсягу знань в галузі інформаційної безпеки та машинного навчання, надаючи аналіз методів виявлення спаму та їх ефективності, а також можуть бути використані для подальшого вдосконалення алгоритмів виявлення спаму та їхнього застосування в реальних умовах.

Апробація результатів роботи була здійснена шляхом реєстрації авторського права та твір (комп’ютерну програму). Цей процес включає подання комплекту документів до Державного підприємства «Український інститут інтелектуальної власності» та отримання документа, що підтверджує правовий статус програми. Це підтверджує ступінь розробленості та практичного застосування досліджуваної комп’ютерної програми в рамках дипломної роботи.

ОГЛЯД АКТУАЛЬНОЇ ЛІТЕРАТУРИ

Велика кількість небажаної кореспонденції, такої як спам-повідомлення, спонукало багатьох дослідників до використання методів машинного навчання для виявлення та фільтрації спаму. Одне з перших досліджень було проведено Полом Гремом у 2002 році. Грем використовував алгоритм машинного навчання для навчання спам-фільтру, який досяг точності 99% [1].

Далі буде наведено огляд деяких досліджень, які можуть бути корисними для розуміння теми. Він має на меті визначити важливі досягнення, проблеми та тенденції в цій галузі, що забезпечить основу подальшої роботи.

Наукові дослідження на задану тему були знайдені шляхом формування відповідного запиту до пошукової системи Google Scholar [2].

Виявлення спаму значно еволюціонувало протягом багатьох років, адаптуючись до мінливого ландшафту методів розсилання спаму. Ранні підходи до виявлення спаму в основному покладалися на правила, які створювалися вручну. Однак ці статичні методи не могли впоратися з динамічною та адаптивною природою спаму. Як наслідок, дослідники звернули увагу на методи машинного навчання, які могли б автоматично навчатися та адаптуватися до нових шаблонів спаму. Дослідники розробили різні моделі та методи для створення інноваційних систем виявлення та фільтрації спаму [3].

У своєму дослідженні про інтелектуальне виявлення спаму Saleh та ін. [4] розглядають ризики безпеки, пов’язані з електронною поштою, зокрема спамом, і досліджують сферу аналізу спаму. Вони висвітлюють як методи машинного навчання, так і статичні методи, що використовуються для виявлення та фільтрації спаму. Зазначається, що машинне навчання з учителем широко застосовуються для виявлення спаму в електронній пошті завдяки високій точності [5]. Ефективність методів навчання з учителем пояснюється їхньою здатністю навчатися на основі маркованих даних. Крім того, підкреслюється, що більшість досліджень у сфері виявлення спаму, особливо виявлення листів,

значною мірою спираються на класифікацію на основі слів, оскільки вміст електронних листів відіграє вирішальну роль у виявленні спам-повідомлень.

Nosseir та ін. [6] представили метод виявлення спаму на основі символів. Цей підхід використовує класифікатор з декількома нейронними мережами, де кожна нейронна мережа навчається з використанням нормалізованих ваг, отриманих на основі ASCII-значень символів у словах. Однак цей метод стикається з проблемами, коли шахраї намагаються уникнути виявлення, маскуючи слова. Вони можуть спотворювати написання слів або використовувати візуальні трюки, що знижує відсоток правильних виявлень системою, оскільки спам-повідомленням вдається обійти механізми виявлення.

У своєму огляді підходів до фільтрації спаму Bhuiyan та ін. [7] проводять всебічний аналіз різних методів та оцінюють їхню точність на основі різних параметрів. Вони зазначають, що існуючі підходи до фільтрації спаму загалом ефективні у фільтрації спаму, причому деякі з них досягають хороших результатів, а інші прагнуть підвищити точність. Однак, незважаючи на успіхи, в методах фільтрації спаму все ще існують проблеми, над вирішенням яких активно працюють дослідники. Автори звертають увагу на постійні зусилля, спрямовані на розробку механізму фільтрації спаму наступного покоління, здатного обробляти великі обсяги мультимедійних даних та ефективно фільтрувати спам-повідомлення. Вони підкреслюють, що алгоритми наївного Байєса та метод опорних векторів є найбільш поширеними методами фільтрації спаму в електронній пошті, оскільки вони вже довели свою надійність та ефективність у виявленні та класифікації спаму.

Також спам є предметом дослідницьких конференцій, зокрема Text REtrieval Conference (TREC) [8].

1 ОСНОВНІ ПОНЯТТЯ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Спам — це не нова, але актуальна проблема, яка в сучасному світі постійно змінюється та адаптується до нових умов. За даними Kaspersky Lab, за 2022 рік кількість спаму серед усіх повідомлень, відправлених електронною поштою, склала майже 49%, а майже 30% з них мали російське походження [9]. В цілому, згідно Statista, кількість спаму за останні 12 років значно знизилася, проте залишається на досить високому рівні, що зображено на рисунку 1.1 [10].

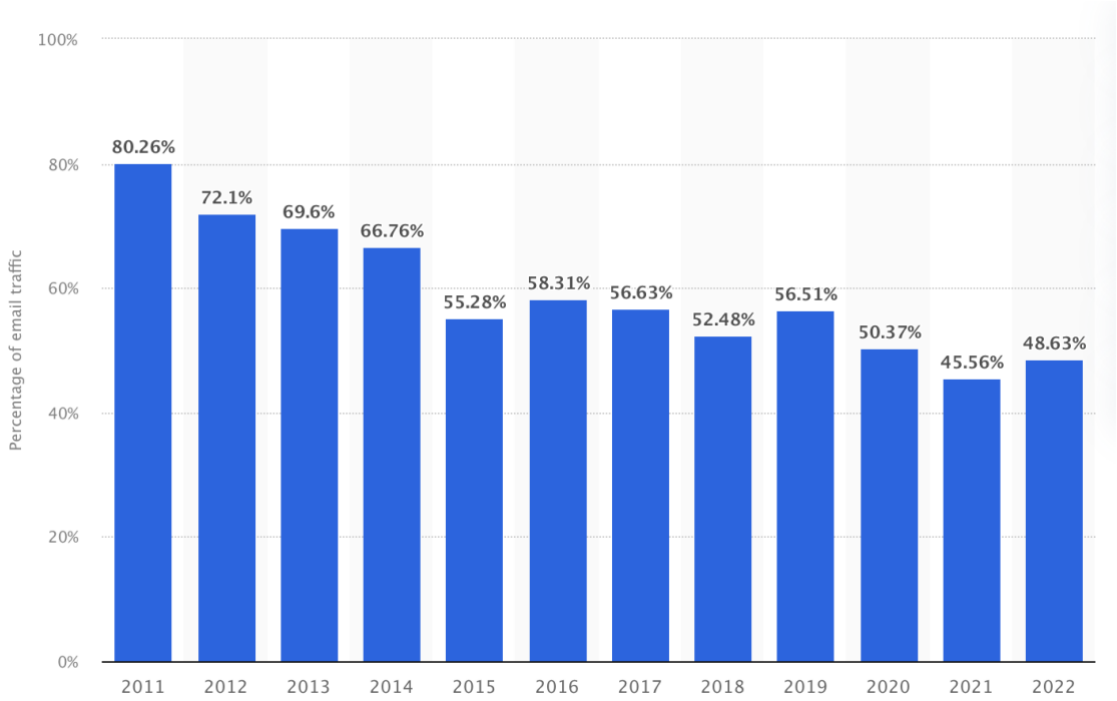


Рисунок 1.1 — Відсоток спаму серед email-трафіку з 2011 по 2022 рік [10]

1.1 Поняття та сутність спаму

Спам — це небажані повідомлення, які зазвичай масово розповсюджуються у великих кількостях з метою обману, реклами або поширення шкідливого контенту. Термін «спам» набув загальної відомості після виходу гумористичного скетчу Монті Пайтона, де це слово повторювалося надмірно часто [11]. У контексті електронної комунікації спам охоплює різні

форми, такі як спам в електронній пошті, месенджерах, соціальних мережах і коментарях.

Спам-повідомлення характеризуються масовим поширенням, повторюваністю та відсутністю явної згоди одержувачів, а також створюють низку викликів і загроз як для окремих осіб, так і для організацій, та навіть всієї інтернет-екосистеми в цілому. Сама суть спаму полягає в його оманливому або деструктивному намірі, з метою реклами, шахрайства і т.д. Він знижує продуктивність, споживає мережеву пропускну здатність і ресурси зберігання даних, а також може призвести до порушень безпеки, фішингових атак і розповсюдження шкідливого програмного забезпечення. Спам часто намагається обдурити одержувачів, маскуючись під легітимну комунікацію, що робить необхідним розробку ефективних механізмів його виявлення.

Спамери часто використовують низку тактик, щоб обійти фільтри і надати повідомленням вигляд справжніх. Ці тактики включають спотворення тексту за допомогою навмисних помилок, вставлення випадкових символів або використання зображень замість вмісту для обходу текстових фільтрів. Спамери також можуть використовувати такі методи, як підміна домену, коли вони фальсифікують особу відправника, щоб виглядати як надійне джерело, або використовують ботнети та зламані системи для розповсюдження спаму з декількох джерел, що ускладнює відстеження його походження.

Суть виявлення спаму полягає в тому, щоб точно відрізнити легітимні повідомлення від спаму. Ефективні механізми виявлення спаму мають на меті зменшити кількість хибно позитивних і хибно негативних виявлень, забезпечуючи баланс між мінімізацією впливу спаму і запобіганням помилковому позначенню легітимних повідомлень як спам.

1.2 Методи боротьби зі спамом

Методи виявлення спаму мають на меті відрізнити спам від легітимних повідомлень, використовуючи різні способи, засновані на їхніх характеристиках

і поведінці. Ці методи можна умовно поділити на дві категорії: статичні та динамічні.

Статичні методи виявлення спаму покладаються на заздалегідь визначені правила та шаблони для ідентифікації спам-повідомлень. Вони працюють на основі специфічних критеріїв і характеристик, пов’язаних зі спамом, що дозволяє їм класифікувати вхідні повідомлення. Хоча статичні методи мають обмеження в роботі з еволюціонучим спамом, вони можуть бути ефективними для виявлення відомих і поширених шаблонів. До статичних методів виявлення відносяться такі прийоми, як фільтрація за ключовими словами, списками.

Метод фільтрації спаму на основі правил зосереджений на аналізі вмісту листа. Підтримуючи заздалегідь визначений список слів або регулярних виразів, пов’язаних зі спамом, процес фільтрації порівнює вміст повідомлення з цим списком і класифікує повідомлення, що містять ключові слова або відповідають патерну регулярного виразу, як спам. Цей підхід передбачає, що присутність певних ключових слів вказує на ймовірність того, що повідомлення є спамом. Однак, такі фільтри іноді можуть давати помилкові спрацьовування. Наприклад, якщо список слів містить такі загальновживані слова, як «дохід», то легітимні листи, такі як робочі звіти, можуть бути помилково позначені як спам. Крім того, спамери можуть навмисно неправильно писати слова, щоб уникнути виявлення, а список необхідно регулярно оновлювати, що може забирати багато часу.

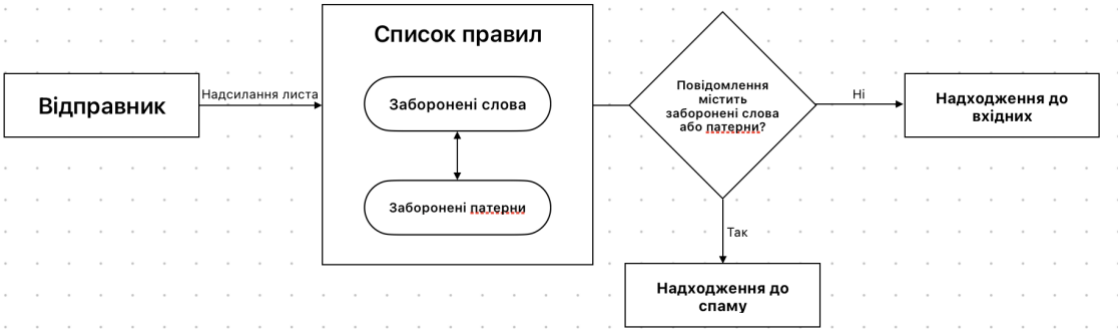


Рисунок 1.2 — Принцип роботи фільтру на основі правил

Фільтрація на основі списків — це ще один із методів. Списки бувають чорні, білі та сірі. Чорний список — найпопулярніший з них. Він містить заздалегідь визначений набір електронних або IP-адрес, з яких раніше надсилався спам. Щоразу, коли надходить новий лист, фільтр порівнює адресу з чорним списком, і якщо вона там присутня, то лист не потрапляє до вхідних повідомлень, а одразу переноситься до відповідної категорії, оскільки вважається спамом. Чорний список може генерувати помилкові спрацьовування, якщо спамер використовує IP-адресу, яку використовували легальні відправники [12]. Існують чорні списки в режимі реального часу (англ. Real-time Blackhole List (RBL)), також відомі як чорні списки системи доменних імен (англ. DNS Blocklist (DNSBL)). Вони мають всеохоплюючий набір даних та обслуговуються третіми сторонами, що робить їх використання більш зручним та не потребує зусиль для оновлення та підтримки.

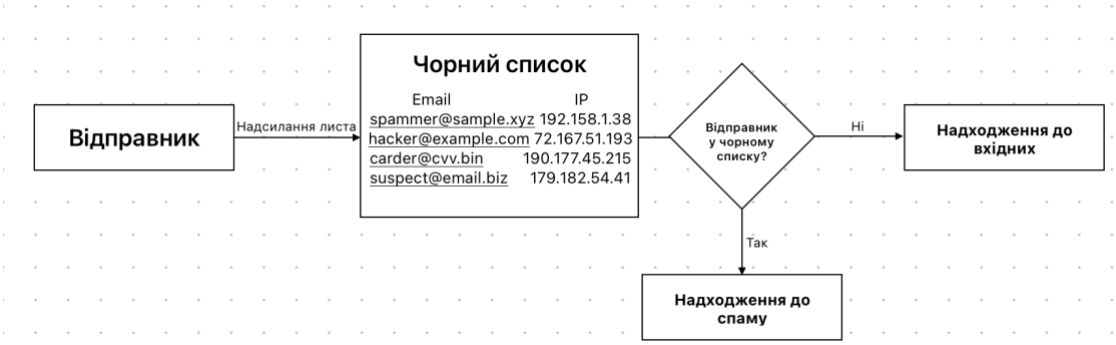


Рисунок 1.3 — Принцип роботи чорного списку

Білий список працює за принципом, протилежним до чорного списку, оскільки він дозволяє отримувати листи лише від певних відправників. Він працює шляхом створення списку довірених користувачів, де вказуються адреси, з яких дозволено отримувати листи. Щоб зменшити ризик блокування легітимних листів, можна використовувати автоматичні білі списки. Вони перевіряють попередньо невідомих відправників листів на причетність до розсилки спаму по базах даних, і якщо вони не були помічені за цим, їх адреса додається до білого списку, а лист надходить до вхідних повідомлень.

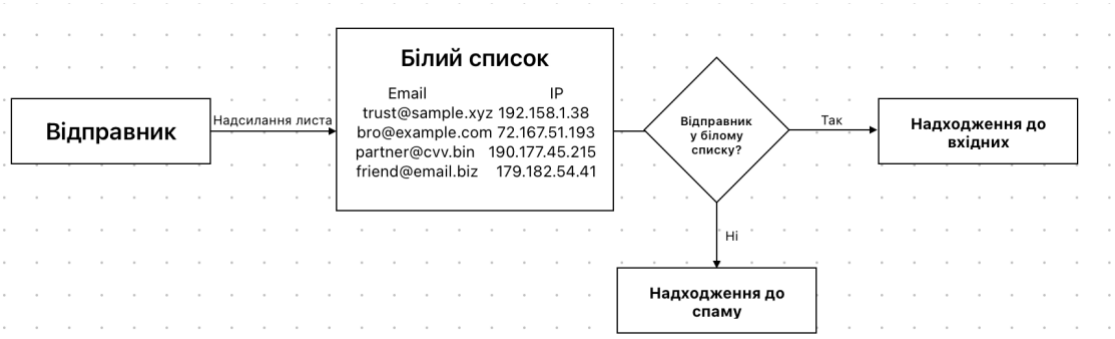


Рисунок 1.4 — Принцип роботи білого списку

Що стосується білих та чорних списків джерел спаму, то проект Spamhaus, започаткований у 1998 році, став професіоналом у цій сфері [13]. Його близьким аналогом є SURBL [14].

Сірі списки — це відносно новий тип списків, який використовує припущення щодо поведінки більшості спамерів, що вони зазвичай надсилають спам однією партією. Принцип роботи полягає в тому, що перше повідомлення від невідомого користувача відхиляється, тому відправник повинен надіслати його вдруге, тоді повідомлення потрапляє до вхідних, а електронна або IP-адреса відправника додається до списку дозволених. Хоча сірі списки вимагають небагато системних ресурсів порівняно з деякими іншими фільтрами спаму, вони можуть спричиняти затримку в доставці пошти, що може бути незручно, коли очікується надходження важливих повідомлень у визначений час.

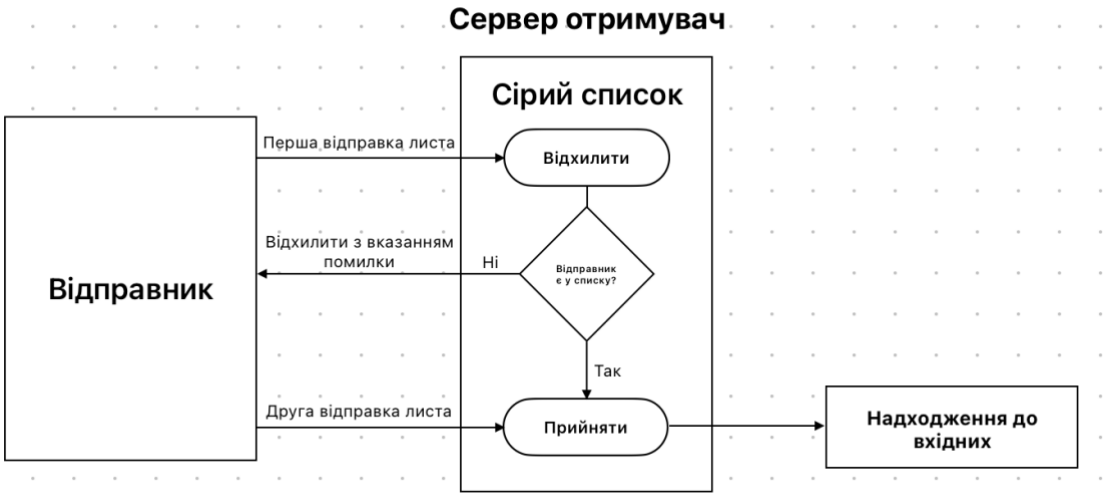


Рисунок 1.5 — Принцип роботи сірого списку

Таблиця 1.1 — Порівняння спам-фільтрів

Фільтр	Актуальність	Переваги	Недоліки
Фільтрація на основі правил	Ефективний для простих і відомих шаблонів спаму	Легко впровадити та оновлювати правила; Низький рівень помилкових спрацьовувань; Націленість на конкретні типи спаму.	Обмежена ефективність проти нових видів спаму; Потребує регулярного оновлення правил.
Чорний список	Коли відомо, що певні відправники або домени є джерелами спаму	Блокує листи з відомих джерел спаму; Може регулярно оновлюватися новими джерелами; Ефективний проти злісних спамерів.	Складно підтримувати список актуальним і повним; Може блокувати легітимні імейли від неправильно внесених відправників.
Чорний список в режимі реального часу	Коли використання зовнішніх баз даних є прийнятним	Використовує бази даних у режимі реального часу для виявлення джерел спаму; Блокує листи з перелічених джерел спаму.	Покладається на зовнішні бази даних, які можуть охоплювати не всі джерела спаму; Може хибно спрацьовувати, якщо легітимні відправники помилково потрапляють до баз даних.
Білий список	Коли слід дозволяти лише листи від відомих надійних відправників	Дозволяє отримувати листи лише від попередньо затверджених відправників; Висока ефективність, якщо список добре підтримується; Низький рівень помилкових спрацьовувань.	Потребує постійного обслуговування для додавання нових надійних відправників; Може блокувати легітимні листи від невідомих або нових відправників; Може забирати багато часу в управлінні для великих організацій.
Сірий список	Коли затримка з незнайомих джерел є прийнятною	Затримує доставку електронної пошти з незнайомих джерел; Ефективно блокує спам від автоматизованих систем розсилання спаму;	Може спричиняти затримки в доставці листів; Менш ефективний проти наполегливих спамерів.

Статичні методи мають переваги з точки зору простоти та обчислювальної ефективності, однак їхня ефективність обмежена статичністю, оскільки вони не можуть адаптуватися до нових методів і варіацій спаму. Спамери постійно вдосконалюють свої стратегії обходу фільтрів, що робить необхідним їх доповнення динамічними підходами, які використовують машинне навчання та

аналіз у реальному часі, що підвищує загальну точність і ефективність систем виявлення спаму.

Динамічні методи виявлення спаму передбачають аналіз поведінки та властивостей повідомлень у режимі реального часу для визначення їхнього класу. На відміну від статичних методів, які покладаються на заздалегідь визначені правила, вони адаптуються до мінливих шаблонів спаму і використовують алгоритми, які постійно навчаються на вхідних повідомленнях. Ці методи використовують методи машинного навчання та аналізу даних для виявлення спаму, враховуючи різні особливості та характеристики повідомлень.

Фільтрація на основі вмісту передбачає аналіз текстового вмісту повідомлень для виявлення шаблонів спаму. Алгоритми машинного навчання тренуються на маркованому наборі даних, де вони вчаться розпізнавати притаманні спаму ознаки. Ці алгоритми виділяють відповідні ознаки зі змісту повідомлення, такі як ключові слова, граматичні структури або лінгвістичні шаблони, і використовують їх для класифікації вхідних повідомлень. Фільтрація на основі вмісту ефективна для виявлення спам-повідомлень, які мають специфічні текстові характеристики, пов’язані зі спамом.

Ще одним динамічним методом є виявлення аномалій. Методи виявлення аномалій спрямовані на виявлення повідомлень, які значно відхиляються від очікуваних шаблонів легітимної комунікації. Будуються моделі, засновані на нормальній поведінці, і їх використовують для виявлення відхилень у вхідних повідомленнях. Наприклад, алгоритм виявлення аномалій може вивчити звичайні шаблони надсилання повідомлень легітимним користувачем і позначити повідомлення, які демонструють незвичну частоту надсилання або нетиповий вміст. Виявляючи незвичну поведінку, вони можуть ефективно ідентифікувати раніше невідомі типи спаму.

Динамічні методи виявлення спаму використовують аналіз у режимі реального часу та адаптивність для ефективної протидії еволюціонуючим

методам спаму. Постійно навчаючись на нових даних і коригуючи свої моделі, динамічні методи можуть виявляти варіації спаму, які не можуть бути зафіксовані статичними методами, тому добре їх доповнюють, адаптуючись до еволюції спам-шаблонів і підвищуючи точність систем виявлення. Однак вони вимагають достатніх обчислювальних ресурсів і можуть бути більш схильні до помилкових спрацьовувань на етапі навчання.

Існують як комерційні, так і продукти з відкритим вихідним кодом для виявлення спаму, які приймають рішення про те, чи є лист спамом, на основі результатів фільтрації, отриманих за допомогою поєднання багатьох фільтрів. Одним із найвідоміших та найбільш ранніх таких рішень є Apache SpamAssassin [15]. Це антиспам-платформа з відкритим вихідним кодом, що надає фільтр для класифікації електронних листів та блокування спаму. Він використовує надійну систему оцінювання та плагіни для інтеграції широкого спектру евристичних і статистичних тестів для аналізу заголовків і основного тексту електронних листів, включаючи аналіз тексту, байєсівську фільтрацію, списки блокування DNS і спільні бази даних для фільтрації [15].

Zerospam — це популярне комерційне програмне забезпечення, широко відоме своїми ефективними можливостями виявлення спаму [16]. Воно використовує різні методи для покращення своїх можливостей фільтрації, включаючи перевірку IP-адрес і доменів, сканування вкладень і URL-адрес, евристичну фільтрацію та байєсівську фільтрацію [17].

Висновки до розділу 1

У цьому розділі було розглянуто поняття та сутність спаму, описано його природу та проблеми, які він створює. Було розглянуто методи виявлення спаму: статичні покладаються на заздалегідь визначені правила та шаблони, тоді як динамічні адаптуються до мінливих спам-шаблонів і використовують методи

машинного навчання. Ці методи є основою для розробки ефективних алгоритмів виявлення спаму. У наступних розділах буде розглянуто застосування методів машинного навчання для розробки надійної і ефективної програми для класифікації спаму.

2 МАШИННЕ НАВЧАННЯ ЯК СПОСІБ ПРОТИДІЇ СПАМУ

Машинне навчання стало потужним підходом до протидії спаму завдяки своїй здатності автоматично навчатися та адаптуватися до нових методів розсилання спаму. Використовуючи великі масиви даних і вдосконалені алгоритми, машинне навчання може ефективно аналізувати і класифікувати вхідні повідомлення, розрізняючи спам і легітимну комунікацію. У цьому розділі буде розглянуто поняття та сутність машинного навчання, різні підходи до нього та його застосування для виявлення спаму.

2.1 Поняття та сутність машинного навчання

Машинне навчання — це підгалузь штучного інтелекту, яка зосереджується на розробці алгоритмів і моделей, здатних автоматично навчатися і вдосконалюватися на основі даних [18]. Воно охоплює низку методів, які дозволяють комп’ютерам виявляти закономірності, робити прогнози та приймати рішення на основі спостережуваних даних. Концепція і суть машинного навчання полягає в його здатності виділяти значущу інформацію і знання з великих масивів даних, дозволяючи системам адаптуватися і покращувати свою продуктивність з часом.

У контексті виявлення спаму алгоритми машинного навчання можна тренувати на маркованих наборах даних, де кожне повідомлення позначене як спам або легітимне. Аналізуючи особливості та характеристики маркованих прикладів, алгоритми вчаться розпізнавати патерни, що вказують на спам, і робити прогнози щодо нових, небачених повідомлень. Чим більш різноманітний і репрезентативний навчальний набір даних, тим краще алгоритм може узагальнювати і точно виявляти спам.

2.1.1 Машинне навчання з учителем

Машинне навчання з учителем передбачає навчання моделей машинного навчання на прикладах з мітками, де вхідні дані (ознаки повідомлень) асоціюються з відповідними вихідними мітками (spam/ham). Моделі навчаються на маркованих прикладах, визначають відповідні шаблони та особливості, а також роблять прогнози щодо нових повідомлень [19].

Процес тренування моделей машинного навчання з учителем зазвичай включає в себе два етапи: навчання і тестування. На етапі навчання алгоритм навчається на маркованих прикладах, налаштовуючи свої внутрішні параметри для мінімізації помилок в майбутньому. На етапі тестування точність алгоритму оцінюється на окремому наборі маркованих прикладів, які не використовувалися під час навчання, що дає змогу оцінити його ефективність у виявленні спаму.

2.1.2 Машинне навчання без учителя

Машинне навчання без учителя — це ще один підхід, з назви якого випливає, що модель не матиме жодних маркованих даних для роботи, а отже, навчання не буде відбуватися. На відміну від навчання з учителем, аналізуються характеристики і закономірності всього набору даних, не покладаючись на марковані приклади. Ці алгоритми спрямовані на виявлення прихованих структур або кластерів у даних, групуючи схожі повідомлення разом [20]. Для виявлення кластерів спам-повідомлень можна використовувати алгоритми кластеризації, такі як k-середні.

Методи машинного навчання без учителя корисні, коли доступність маркованих прикладів обмежена або коли ви маєте справу з новими або невідомими шаблонами спаму. Ці методи можуть виявити викиди або аномалії в наборі даних, які можуть вказувати на раніше невидиму спам-діяльність.

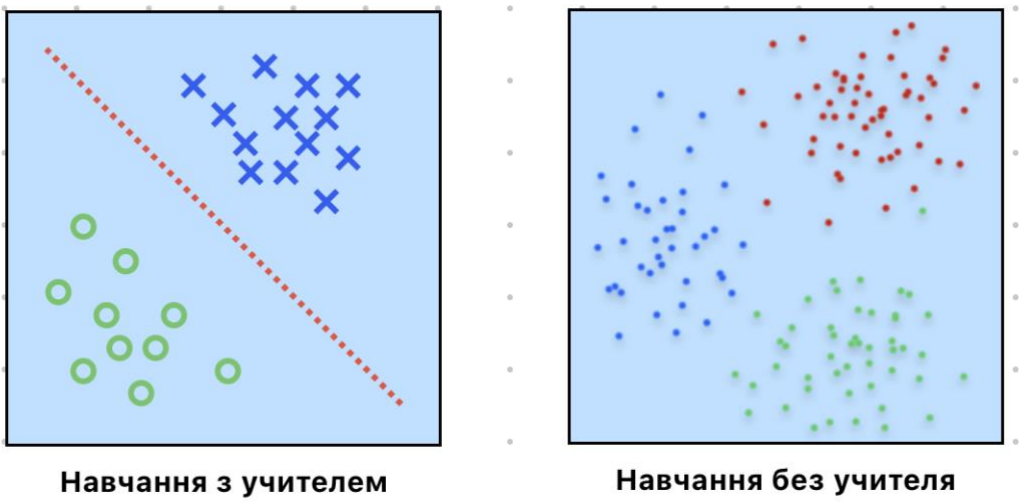


Рисунок 2.1 — Порівняння навчання з учителем і без учителя

2.2 Методи машинного навчання для виявлення спаму

У контексті класифікації спаму для досягнення точних і ефективних результатів застосовуються різні методи машинного навчання. У цьому підрозділі буде розглянуто такі моделі, як наївний Байєс, метод опорних векторів, k-найближчих сусідів та випадковий ліс.

2.2.1 Naïve Bayes

Наївний Байєс — це імовірнісний класифікатор, який передбачає незалежність між ознаками. Він обчислює ймовірність того, що повідомлення є спамом або легітимним, на основі наявності певних ознак. Незважаючи на своє спрощене припущення, наївний Байєс показав багатообіцяючі результати у виявленні спаму завдяки своїй обчислювальній ефективності та простоті реалізації. Наївний Байєс спирається на теорему Байєса [21]:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}, \text{ де}$$

A і *B* — події,

P(A) і *P(B)* — апріорні ймовірності подій A і B,

P(A | B) — умовна ймовірність події A при умові істинності B (апостеріорна вірогідність),

P(B | A) — ймовірність B за умови істинності A.

Існують різні типи байєсових класифікаторів, наприклад: Гаусів, мультиноміальний та Бернуллі. Mccallum та Nigam [22] виявили, що мультиноміальний наївний Байєс краще за інші моделі працює з класифікацією тексту, тому враховуючи поставлене завдання (класифікація текстових повідомлень), у цій роботі використано його.

2.2.2 Support Vector Machine

Метод опорних векторів — це алгоритм машинного навчання з учителем, який має на меті знайти оптимальну гіперплощину (англ. hyperplane), що відокремлює спам-повідомлення від легітимних, максимізуючи відстань між ними. Нові повідомлення класифікуються на основі їхнього положення відносно вивченої площини. SVM може ефективно обробляти дані високої розмірності і здатен вловлювати складні взаємозв'язки між елементами. Він широко використовуються в системах виявлення спаму завдяки своїй надійності та здатності обробляти великі масиви даних.

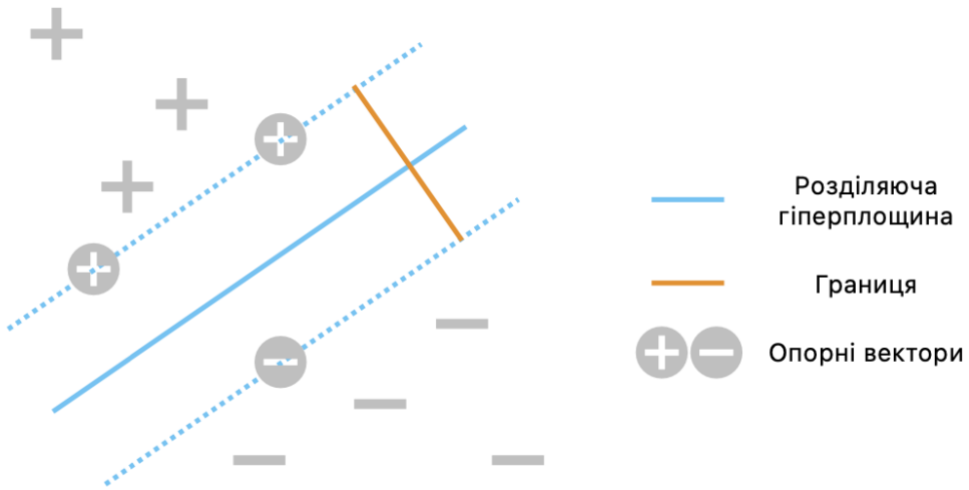


Рисунок 2.2 — Схема роботи SVM

2.2.3 k-Nearest Neighbors

k-найближчих сусідів — це простий, але ефективний алгоритм, який класифікує повідомлення на основі міток класів його найближчих сусідів у просторі ознак. KNN належить до лінивих алгоритмів: це означає, що він намагається лише запам’ятати процес, якому він не навчається самостійно, а також не приймає власних рішень [23]. Він працює на припущенні, що повідомлення зі схожими ознаками, як правило, належать до одного класу. KNN є універсальним і може адаптуватися до різних типів даних, що робить його придатним для завдань виявлення спаму.

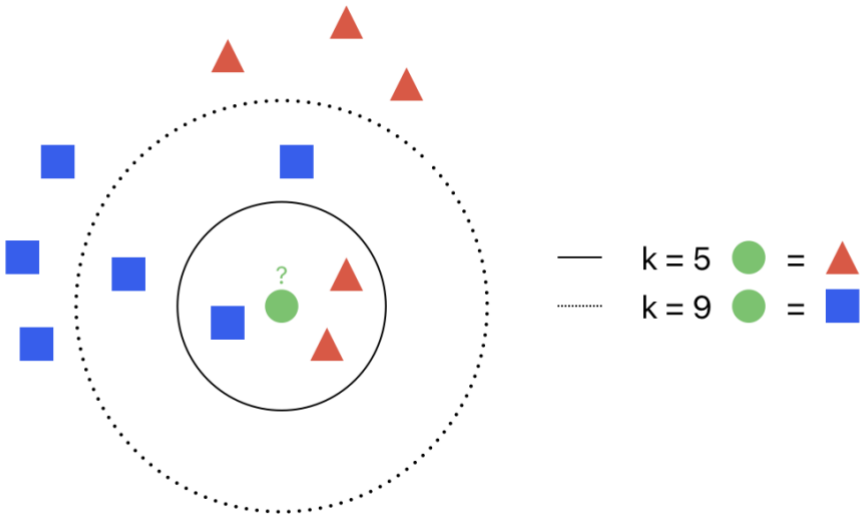


Рисунок 2.3 — Схема роботи KNN

2.2.4 Random Forest

Випадковий ліс — це ансамблевий метод навчання, який поєднує кілька дерев рішень для прогнозування. Кожне дерево в лісі навчається на підмножині даних, а остаточний прогноз визначається шляхом агрегування прогнозів окремих дерев. Random Forest може обробляти багатовимірні дані та фіксувати складні взаємодії, а випадковість, що вводиться при побудові дерев, допомагає

покращити узагальнення. Він був успішно застосований для виявлення спаму завдяки своїй надійності та здатності обробляти великі набори даних.

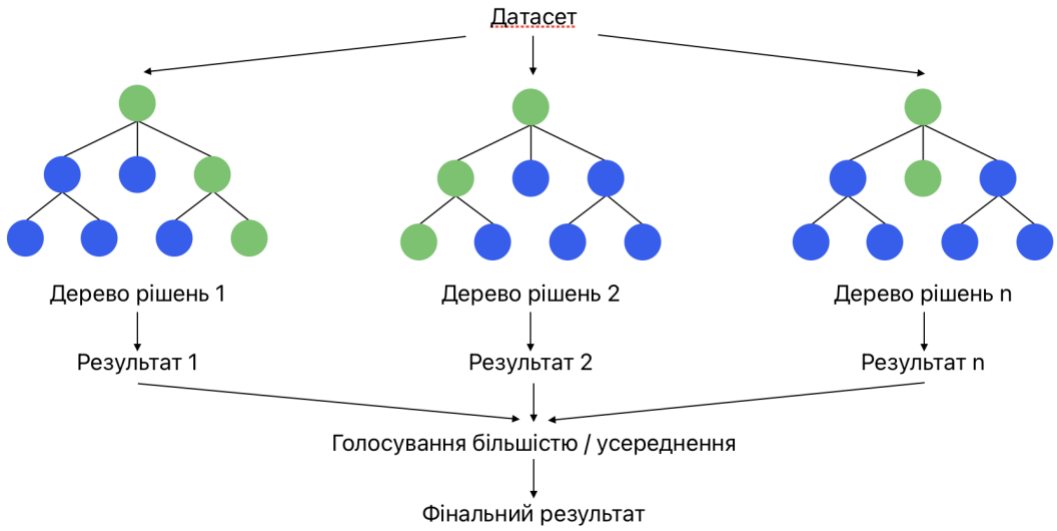


Рисунок 2.4 — Схема роботи Random Forest

2.3 Методи глибинного навчання та мовні моделі для виявлення спаму

Глибинне навчання, підгалузь машинного навчання, що зосереджена на навчанні нейронних мереж (алгоритмів, що намагаються моделювати роботу людського мозку), привернуло значну увагу в останні роки завдяки своїй здатності автоматично вилучати ознаки високого рівня з даних [26]. Моделі глибинного навчання трансформери, наприклад, BERT (англ. Bidirectional Encoder Representations from Transformers) та мовна модель GPT-2 (англ. Generative Pre-trained Transformer 2), продемонстрували вражаючу продуктивність у різних завданнях обробки природної мови.

Трансформер — це архітектура нейронної мережі, основна ідея якої полягає у використанні самоуваги (англ. self-attention) — механізму, що дозволяє диференційовно зважувати важливість кожної частини вхідних даних та враховувати контекст [27]. Найбільше трансформери використовують у NLP

через їх здатність ефективно розуміти й генерувати текст, також вони використовуються у сфері CV (комп'ютерне бачення, англ. Computer Vision).

Трансформер працює у два основні етапи: кодування та декодування. У процесі кодування вхідна послідовність (наприклад, речення) подається на вхід трансформеру. Самоувага використовується для визначення важливості кожного слова у контексті і створення векторних представлень цих слів. В процесі декодування модель послідовно генерує наступні елементи послідовності, враховуючи контекст та попередні згенеровані елементи. Трансформери не обмежуються лише обробкою тексту, а можуть бути використані й для роботи з аудіо та зображеннями.

2.3.1 BERT

BERT — це модель глибинного навчання на основі двонаправленого трансформера, представлена у 2018 році дослідниками із Google [24]. Це попередньо навчена модель, що використовує навчання на великих масивах даних для вивчення контекстного представлення слів або лексем, яка досягла неабиякого успіху в різних задачах обробки природної мови [24]. Моделі BERT навчаються шляхом передбачення пропущених слів у реченні, враховуючи навколишній контекст, що дозволяє їм фіксувати багату семантичну та синтаксичну інформацію.

Після тренування BERT може бути використаний для різних завдань, таких як класифікація тексту, проте він не може генерувати та підказувати текст, оскільки не має декодера у своїй архітектурі [28].

2.3.2 GPT-2

GPT-2 — мовна модель глибинного навчання, розроблена компанією OpenAI [25], яка використовує архітектуру на основі трансформера. Вона була

натренована на великій кількості текстових даних, що дозволяє їй розуміти контекст, граматику та семантику мови, тому вона здатна генерувати зв'язний і контекстно релевантний текст. Хоча GPT-2 показав багатообіцяючі результати, його ресурсоємний характер та основний фокус на генерації зрозумілого та креативного тексту вимагають ретельного розгляду при інтеграції його в системи виявлення спаму. Це можна спробувати зробити, застосувавши додаткове налаштування (англ. fine-tuning), проте цей процес вимагає значних обчислювальних ресурсів та експертизи в галузі машинного навчання.

2.4 Тюнінг гіперпараметрів

Гіперпараметри — це параметри моделі, які визначають її структуру та спосіб навчання, а не вивчаються в процесі тренування. Це можуть бути швидкість навчання, кількість шарів у нейронній мережі, глибина, функції активації та багато інших.

Тюнінг гіперпараметрів — це процес вибору найкращих значень гіперпараметрів моделі машинного навчання для досягнення кращої точності або продуктивності.

GridSearchCV (Grid Search Cross-Validation) — це метод автоматичного підбору гіперпараметрів моделі з бібліотеки scikit-learn [36]. Він працює шляхом перебору всіх можливих комбінацій параметрів із заздалегідь заданого набору значень і допомагає знайти оптимальну комбінацію гіперпараметрів для моделі.

2.5 Метрики оцінки моделей

Аналіз результатів роботи алгоритму виявлення спаму має вирішальне значення для отримання уявлення про його ефективність, розуміння його обмежень і прийняття обґрунтованих рішень щодо подальшого використання. Метрики оцінки моделей машинного навчання — це числові показники, які

використовуються для вимірювання ефективності моделей та оцінки їхньої продуктивності. Вони надають об’єктивну міру того, наскільки добре модель працює на вхідних даних та наскільки правильно вона розв’язує поставлену задачу.

Матриця невідповідностей (англ. Confusion Matrix) — корисний інструмент для аналізу ефективності алгоритму. Вона забезпечує табличне представлення прогнозів алгоритму в порівнянні з реальними мітками. Матриця включає чотири компоненти: істинно позитивні (TP), істинно негативні (TN), хибно позитивні (FP) та хибно негативні (FN). Вона дозволяє більш детально проаналізувати роботу алгоритму, виявити типи помилок, яких припускається алгоритм та визначити потенційні можливості вдосконалення. На основі показників TP, TN, FP та FN розраховуються такі метрики, як точність, влучність, повнота й оцінка F1, що будуть розглянуті далі.

Точність (англ. Accuracy) — це найпростіша метрика класифікації. Вона відображає частку правильних прогнозів (як TP, так і TN) серед загальної кількості розглянутих випадків. Точність є поганою метрикою у випадку дисбалансу даних, оскільки вона не може розрізнити конкретні типи помилок (FP та FN). Формула для розрахунку точності має вигляд:

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

Влучність (англ. Precision) — це частка істинно позитивних прогнозів серед загальної кількості позитивних прогнозів (серед усього, що було передбачено як правильне, влучність підраховує відсоток, який виявився істинно правильним). Невлучна модель може знайти багато позитивних результатів, але вона також помилково визначає багато результатів, які насправді не є позитивними. І навпаки, влучна модель може знаходити не всі позитивні

результати, але ті, які модель класифікує як позитивні, з великою ймовірністю є правильними. Вона розраховується за формулою:

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

Повнота (англ. Recall) — це співвідношення правильних позитивних прогнозів до загальної кількості позитивних прикладів (скільки з того, що істинно позитивне, вдалося визначити моделі). Моделі з високою повнотою добре знаходять усі позитивні випадки в даних, хоча вони також можуть помилково ідентифікувати деякі негативні випадки як позитивні, а моделі з низькою повнотою не здатні знайти всі (або більшу частину) позитивних випадків у даних. Повнота розраховується за формулою:

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

Оцінка F1 (англ. F1 Score) — середнє гармонійне (один з випадків усереднення) значення влучності та повноти. Обидва показники важливі, тому логічно використовувати F1, яка комбінує їх в одну метрику. Вона добре працює на незбалансованих даних і розраховується за формулою:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}$$

Висновки до розділу 2

Методи машинного навчання є ефективним засобом протидії спаму, оскільки воно може ефективно класифікувати вхідні повідомлення на spam і ham. Ці методи використовують великі набори даних і комплексні алгоритми для навчання і вилучення ознак, що вказують на спам. Було розглянуто такі моделі машинного навчання, як наївний Байєс, метод опорних векторів, k-найближчих сусідів та випадковий ліс, а також моделі глибинного навчання BERT і GPT-2.

Було розглянуто поняття тюнінгу гіперпараметрів моделі, а також описані метрики оцінки моделей. Використовуючи можливості машинного та глибинного навчання, можна підвищити точність та ефективність систем виявлення спаму, пом’якшуючи вплив спаму на людей та організації.

3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ МЕТОДІВ ВИЯВЛЕННЯ СПАМУ НА ОСНОВІ МАШИННОГО НАВЧАННЯ

3.1 Збір та обробка даних для навчання моделі

Для розробки ефективної програми для виявлення спаму з використанням машинного навчання необхідно знайти або скласти промаркований набір даних, що містить текстові дані та мітки класу (spam/ham, бажано в рівних пропорціях), а потім ці дані обробити та вилучити з них ознаки. Важливо враховувати якість датасету та його розмір.

3.1.1 Вибір та огляд датасета

Для роботи було обрано 2 датасети: Spam Text Message Classification [29] (далі — датасет SMS) та Email Spam Dataset [30] (далі — датасет Email). Такий вибір датасетів має на меті охопити різні типи спам-повідомлень.

Датасет SMS має два стовпці (“Category”: клас повідомлення; “Message”: текст повідомлення):

	C1	C2
1	Category	Message
2	ham	Go until jurong point, crazy.. Available only i…
3	ham	Ok lar... Joking wif u oni...
4	spam	Free entry in 2 a wkly comp to win FA Cup final…
5	ham	U dun say so early hor... U c already then say.…
6	ham	Nah I don't think he goes to usf, he lives arou…
7	spam	FreeMsg Hey there darling it's been 3 week's no…

Рисунок 3.1 — Загальна структура sms_spam.csv

Датасет Email складається з трьох файлів (компіляція датасетів SpamAssassin [31], LingSpam [32], та EnronSpam[33]), що пізніше будуть об’єднані в один. Кожен датасет містить два основні стовпці (“Body”: текст повідомлення; “Label”: клас повідомлення) та зайві, що будуть видалені:

completeSpamAssassin.csv | enronSpamSubset.csv | lingSpam.csv

	<anonymous>	Body	Label
1	0	⏎Save up to 70% on Life Insurance.⏎Why Spend Mo…	1
2	1	1) Fight The Risk of Cancer!⏎http://www.adclick…	1
3	2	1) Fight The Risk of Cancer!⏎http://www.adclick…	1
4	3	##…	1
5	4	I thought you might like these:⏎1) Slim Down - …	1

Рисунок 3.2 — Загальна структура completeSpamAssassin.csv

completeSpamAssassin.csv | enronSpamSubset.csv | lingSpam.csv

	C1	C2	C3	C4
1	<null>	Unnamed: 0	Body	Label
2	2469	2469	Subject: stock promo mover : cwtd⏎ * * * urgen…	1
3	5063	5063	Subject: are you listed in major search engine…	1
4	12564	12564	Subject: important information thu , 30 jun 20…	1
5	2796	2796	Subject: = ? utf - 8 ? q ? bask your life with…	1

Рисунок 3.3 — Загальна структура enronSpamSubset.csv

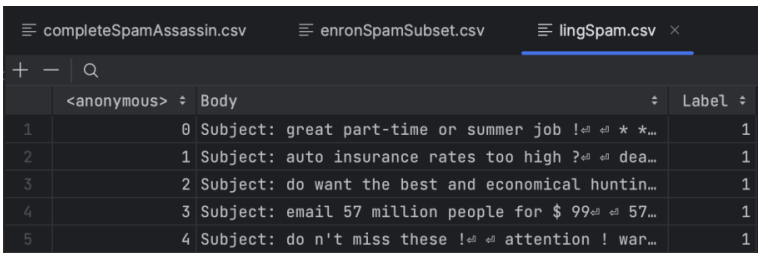
completeSpamAssassin.csv | enronSpamSubset.csv | lingSpam.csv

	<anonymous>	Body	Label
1	0	Subject: great part-time or summer job !⏎ ⏎ * *…	1
2	1	Subject: auto insurance rates too high ?⏎ ⏎ dea…	1
3	2	Subject: do want the best and economical huntin…	1
4	3	Subject: email 57 million people for $ 99⏎ ⏎ 57…	1
5	4	Subject: do n't miss these !⏎ ⏎ attention ! war…	1

Рисунок 3.4 — Загальна структура lingSpam.csv

3.1.2 Попередня обробка даних та вилучення ознак

Попередня обробка має ключове значення для очищення та перетворення вихідних даних у формат, придатний для подальшої роботи. Першим кроком роботи з даними є їх препроцесинг та візуалізація. Спочатку видаляються зайві, перейменовуються основні стовпці, потім з тексту видаляються посилання, слова, що зустрічаються надто часто, та усі неалфавітні символи. Для цього використовується бібліотека для операцій з регулярними виразами re [34]. Далі застосовується стемінг або лематизація (на вибір):



completeSpamAssassin.csv | enronSpamSubset.csv | lingSpam.csv

	<anonymous>	Body	Label
1	0	Subject: great part-time or summer job !⏎ ⏎ * *…	1
2	1	Subject: auto insurance rates too high ?⏎ ⏎ dea…	1
3	2	Subject: do want the best and economical huntin…	1
4	3	Subject: email 57 million people for $ 99⏎ ⏎ 57…	1
5	4	Subject: do n't miss these !⏎ ⏎ attention ! war…	1

Рисунок 3.5 — Вигляд даних до препроцесингу

Стемінг — це метод зведення слів до їх базової форми. В результаті слово скорочується до стема, який не завжди є справжнім словом: наприклад, слова “improve”, “improving”, “improvement”, “improved” будуть зведені до стема “improv”. Стемінг може бути корисним, якщо точність скорочення не є критичною, але потрібна швидкість обробки даних. Для застосування стемінгу використовується PorterStemmer бібліотеки NLTK [35].

	label	text
0	1	great parttim summer job display box credit ap...
1	1	auto insur rate high dear nlpeopl sure agre au...
2	1	want best econom hunt vacat life want best hun...
3	1	email million peopl million email address onli...
4	1	nt miss attent warn adult onli warn adult onli...

Рисунок 3.6 — Результат застосування стемінгу

Лематизація — це інший метод скорочення слів до їх базової форми. В результаті слово скорочується до леми, яка завжди є справжнім словом: наприклад, слова “improve”, “improving”, “improvement”, “improved” будуть зведені до леми “improve”. Лематизація допомагає зменшити кількість унікальних слів у датасеті, що полегшує подальшу обробку та аналіз тексту. Це особливо важливо при використанні методів, які залежать від словника, таких як мішок слів. Для застосування лематизації використовується WordNetLemmatizer бібліотеки NLTK [35].

	label	text
0	1	great parttime summer job display box credit a...
1	1	auto insurance rate high dear nlpeople sure ag...
2	1	want best economical hunting vacation life wan...
3	1	email million people million email address wan...
4	1	nt miss attention warning adult warning adult ...

Рисунок 3.7 — Результат застосування лематизації

Наступним кроком видаляються стоп-слова — слова, що не несуть суттєвої інформації при аналізі тексту, такі як “I”, “the”, “and”, “is”. Вони є досить поширеними, можуть займати багато місця у датасеті та спотворювати результати аналізу. Їх видалення може допомогти зосередитися на більш значущих словах та зробити датасет компактніше. Стоп-слова видаляються з використанням словника “English stopwords” nltk.corpus [35].

Після попередньої обробки дані можна візуалізувати, щоб наглядно побачити, що з себе представляє датасет:

Рисунок 3.8 — Хмара найпопулярніших слів датасету SMS (зліва) та Email (справа)

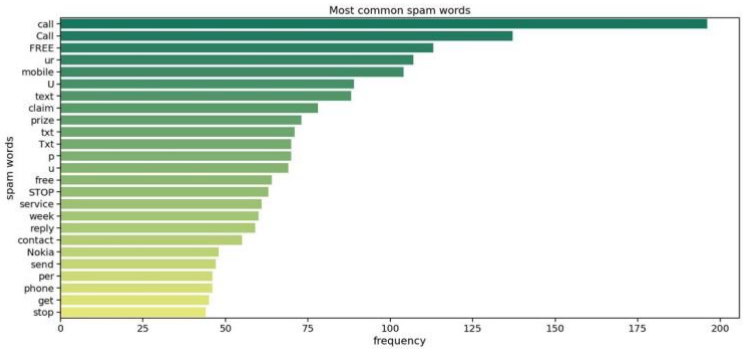


Рисунок 3.9 — гістограма найуживаніх слів у spam’і датасету SMS

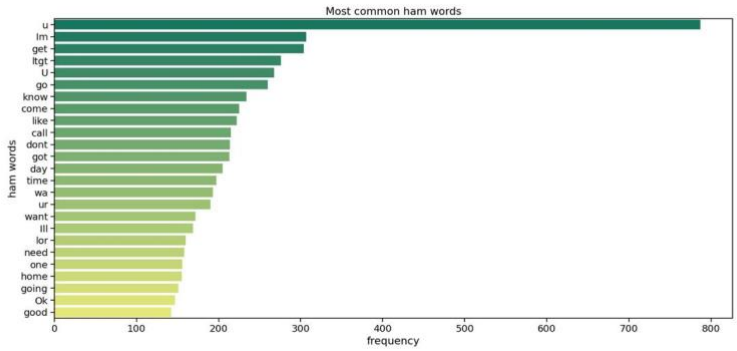


Рисунок 3.10 — гістограма найуживаніх слів у ham’і датасету SMS

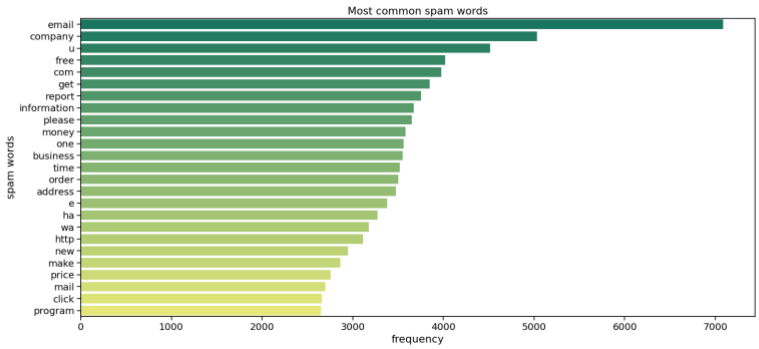


Рисунок 3.11 — гістограма найуживаніх слів у spam'і датасету Email

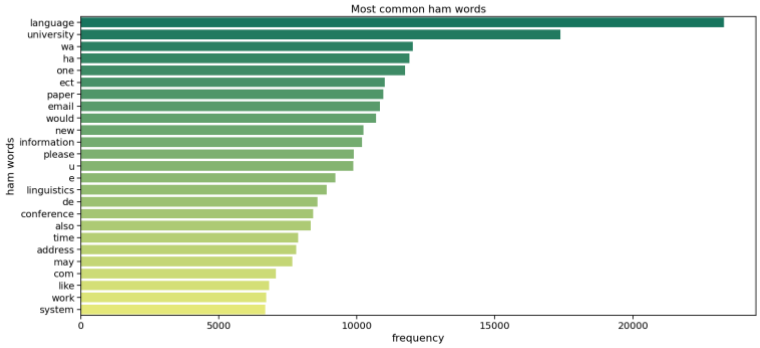


Рисунок 3.12 — гістограма найуживаніх слів у ham'і датасету Email

Як видно з рисунків 3.8-3.13, не всі стоп-слова були видалені. Це трапляється через особливості розбиття речень на слова (токени) при їх обробці. Стоп-слова остаточно буде видалено на етапі векторизації даних.

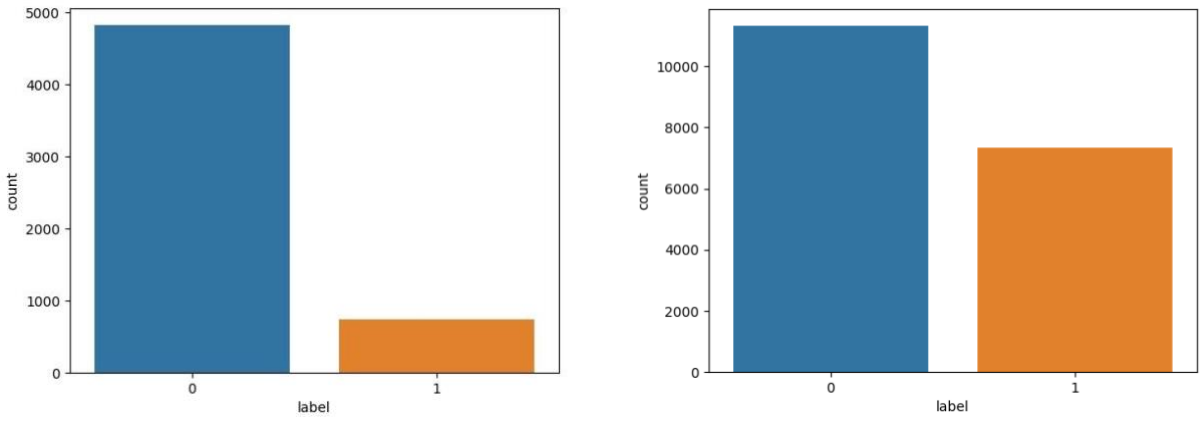


Рисунок 3.13 — розподіл класів у датасеті датасеті SMS (зліва) та Email (справа)

З рисунку 3.13 випливає, що датасет SMS не збалансовано, а отже до нього можна застосувати апсемплінг, продублювавши датафрейм-меншість таку кількість разів, щоб датасет став збалансованим (рис. 3.14).

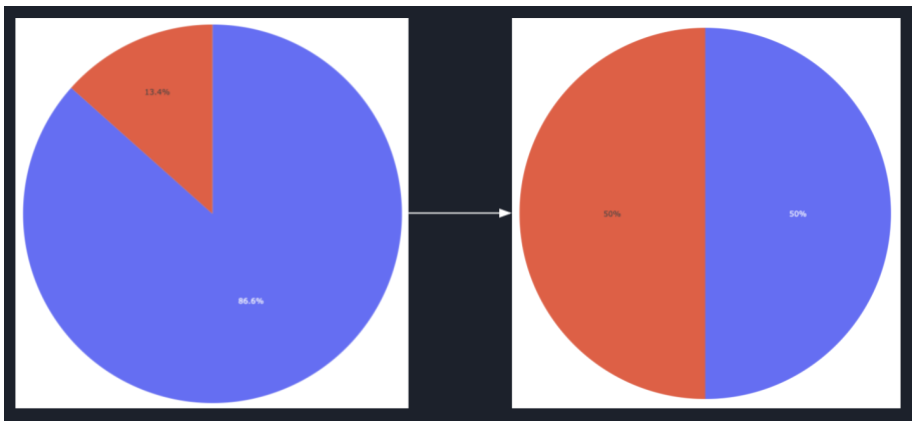


Рисунок 3.14 — розподіл класів датасету SMS до та після апсемплінгу

Щоб перевести зрозумілий для людини текст у зрозумілу для комп'ютера векторну форму, потрібно вилучити ознаки, застосувавши векторизацію.

Мішок слів (англ. Bag of Words) — простий та ефективний метод вилучення ознак з текстових даних. Він передбачає побудову словника всіх унікальних слів, що трапляються у датасеті. Кожен текст представляється у вигляді вектора, де кожна компонента відповідає кількості входжень певного слова зі словника у даному тексті. Значення вектора може бути бінарним (0 — слово не зустрічається, 1 — слово присутнє) або відображати кількість входжень слова. Для застосування векторизації за допомогою мішку слів використовується CountVectorizer бібліотеки scikit-learn [36].

TF-IDF (Term Frequency-Inverse Document Frequency) є іншим популярним методом вилучення ознак з тексту. Він враховує не лише частоту слова у тексті (term frequency), але й його значення у контексті всього датасету (inverse document frequency). TF-IDF обчислюється за формулою, яка залежить від кількості входжень слова у текст, загальної кількості слів у тексті та кількості документів, у яких зустрічається дане слово:

$$w_{x,y} = tf_{x,y} * \log\frac{N}{df_x}, \text{ де}$$

tf — частота *x* в *y*,

df — кількість документів, що містять *x*,

N — загальна кількість документів.

Цей метод дозволяє виділити слова, які є важливими для конкретного тексту, при цьому зменшуючи значення загальних та менш важливих слів. Для застосування векторизації за допомогою TF-IDF використовується TfidfVectorizer бібліотеки scikit-learn [36].

На цьому попередню обробку даних та вилучення ознак можна завершити, і перейти до навчання моделей.

3.2 Розробка застосунку для класифікації спаму

Для імплементації 4-х моделей машинного навчання Naïve Bayes (MultinomialNB()), SVM (SVC()), KNN (KNeighborsClassifier()) та Random Forest (RandomForestClassifier()) були використані модулі naive_bayes, svm, neighbors та ensemble бібліотеки scikit-learn [36] відповідно.

Також було імплементовано BERT. Для його програмної реалізації було використано модуль попередньої обробки тексту [37] та модуль кодування [38], а також застосовано методи бібліотеки TensorFlow [39].

Вибір цих моделей забезпечує різноманітний набір класифікаторів для порівняння їхньої ефективності. Під час роботи програми користувач вирішує, обрати конкретну модель для навчання або всі одразу та чи застосовувати під час навчання тюнінг гіперпараметрів. Для тюнінгу гіперпараметрів у роботі використовується 5-кратна кросвалідація GridSearchCV, а за метрику максимізації взято F1-Score. У таблиці 3.1 наведено перелік гіперпараметрів, що тюнінгуються.

Таблиця 3.1 — Перелік гіперпараметрів, що тюнінгуються (опціонально)

Модель	**Гіперпараметр**	**Опис**	**Значення**
Naïve Bayes	Не має	-	-
SVM	C	Компроміс між точністю класифікації і величиною розділяючої гіперплощини	0.1, 1, 5, 7, 10
	kernel	Тип ядра: визначає функцію, яка використовується для перетворення вхідних даних в більш високу розмірність	linear, rbf, poly, sigmoid
KNN	n_neighbors	Кількість найближчих сусідів, які використовуються для класифікації нового зразка	1, 3, 5, 7, 9
	weights	Вага, яка надається кожному з сусідів при класифікації	uniform, distance
Random Forest	n_estimators	Кількість дерев рішень	10, 20, 50, 100, 200
	max_depth	Максимальна глибина дерев рішень	None, 10, 20, 50, 100

В результаті проведеної роботи мовою програмування Python було написано програму, що виконує завдання з виявлення спаму в текстових повідомлення та електронних листах. Вона складається з наступних етапів:

1. Завантаження та підготовка даних:
    a. Користувач обирає тип даних (SMS або Email).
    b. Завантажуються відповідні дані.
    c. Виконуються необхідні операції, такі як перейменування колонок, видалення зайвих.
2. Попередня обробка даних:
    a. Виконання обробки тексту, такої як видалення посилань, усіх неалфавітних символів.
    b. Вибір типу попередньої обробки (стемінг або лематизація).
    c. Видалення стоп-слів з тексту.
3. Візуалізація даних:
    a. Загальний огляд датасету.
    b. Побудова діаграми розподілу класів (pie chart).

c. Побудова графіку розподілу класів (bar chart).
d. Побудова хмари найбільш поширених слів.
e. Побудова гістограм розподілу найпоширеніших слів кожного класу.

4. Векторизація даних:
   a. Вибір методу векторизації (Bag of Words або TF-IDF).
   b. Застосування відповідного векторизатора для перетворення текстових даних на числові вектори.
5. Розбиття даних на тренувальний набір із відповідними векторами ознак та мітками класів і тестовий набір для оцінки моделі.
6. Навчання моделі:
   a. Вибір алгоритму класифікації.
   b. Тюнінг гіперпараметрів моделі (опціонально).
   c. Навчання моделі на тренувальних даних.
7. Оцінка результатів:
   a. Оцінка ефективності моделі на тестовому наборі.
   b. Виведення метрик оцінки моделі, таких як accuracy, precision, recall та f1 score, побудова матриці невідповідностей.
   c. Графічне порівняння моделей (якщо використовувалися усі одразу).

3.3 Аналіз результатів роботи застосунку та ефективності моделей

3.3.1 Експеримент 1.1

На датасеті SMS, з застосуванням стемінгу та векторизації за допомогою Bag of Words, без апсемплінгу та тюнінгу гіперпараметрів, отримано наступні результати, зображені у таблиці 3.2 та на рисунку 3.15:

Таблиця 3.2 — Порівняння ефективності моделей, Експеримент 1.1

Хар-ка\Модель	**NB**	**SVM**	**KNN**	**RF**
Час тренування, с	0.005	0.397	0.0005	2.62
Accuracy	0.97	0.972	0.928	0.975
Precision	0.853	1.0	1.0	1.0
Recall	0.933	0.792	0.463	0.812
F1-Score	0.891	0.884	0.633	0.896
Матриця невідповідностей	[[942 24] [10 139]]	[[966 0] [31 118]]	[[966 0] [80 69]]	[[966 0] [28 121]]

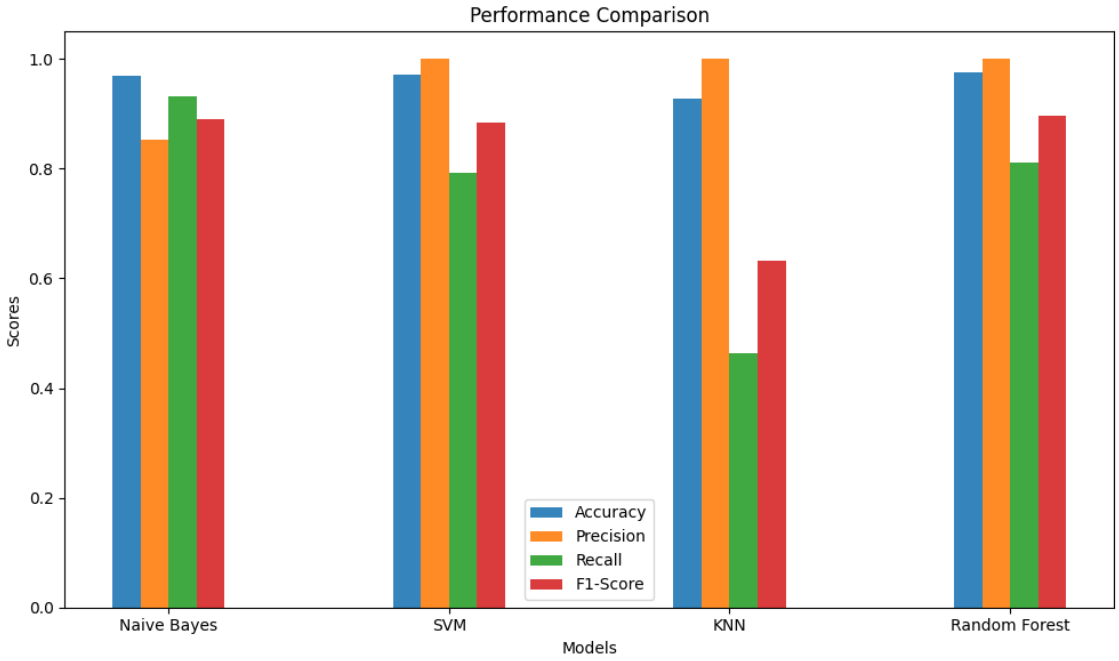


Рисунок 3.15 — Порівняння ефективності моделей, Експеримент 1.1

3.3.2 Експеримент 1.2 (з апсемплінгом)

Провівши апсемплінг, без зміни інших параметрів Експерименту 1.1, було отримано результати, продемонстровані у таблиці 3.3 та на рисунку 3.16:

Таблиця 3.3 — Порівняння ефективності моделей, Експеримент 1.2

Хар-ка\Модель	**NB**	**SVM**	**KNN**	**RF**
Час тренування, с	0.005	0.971	0.0005	3.074
Accuracy	0.975	0.995	0.985	0.999
Precision	0.966	0.996	0.992	0.999
Recall	0.984	0.995	0.977	0.999
F1-Score	0.975	0.995	0.985	0.999
Матриця невідповідностей	[[952 33] [15 930]]	[[981 4] [5 940]]	[[978 7] [22 923]]	[[984 1] [1 944]]

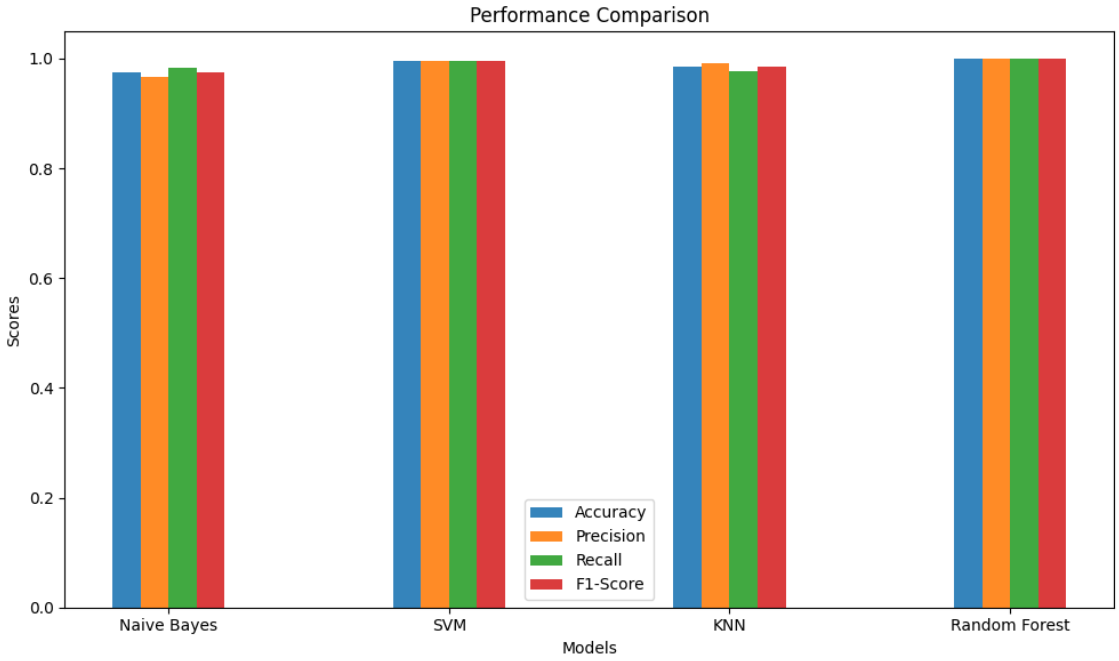


Рисунок 3.16 — Порівняння ефективності моделей, Експеримент 1.2

Як видно з таблиці 3.3 та рисунку 3.16, відбулося перенавчання моделей і задача виявилася надто простою для всіх моделей, всі показники наближені до 1. Спробуємо не робити апсемплінг, а застосувати тюнінг гіперпараметрів.

3.3.3 Експеримент 1.3 (з тюнінгом гіперпараметрів)

Таблиця 3.4 — Порівняння ефективності моделей, Експеримент 1.3

Хар-ка\Модель	**NB**	**SVM**	**KNN**	**RF**
Час тюнінгу, с	-	7.114	0.675	24.749
Час тренування, с	0.004	0.149	0.0005	2.658
Accuracy	0.97	0.984	0.952	0.974
Precision	0.857	1.0	0.962	1.0
Recall	0.926	0.879	0.671	0.805
F1-Score	0.89	0.936	0.791	0.892
Матриця невідповідностей	[[943 23] [11 138]]	[[966 0] [18 131]]	[[962 4] [49 100]]	[[966 0] [29 120]]

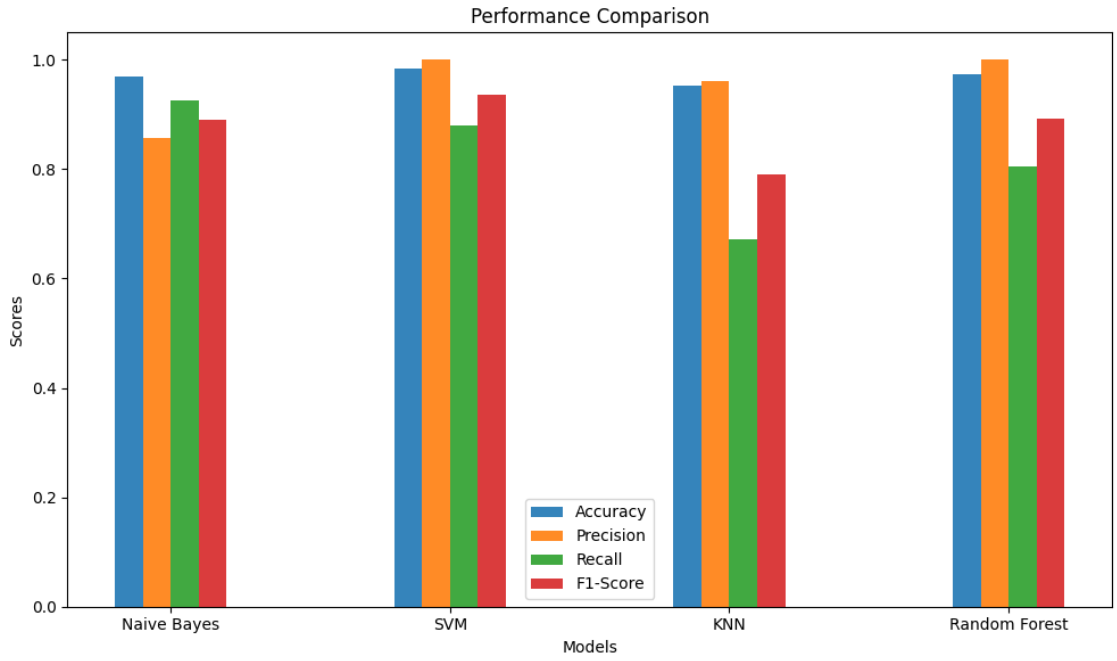


Рисунок 3.17 — Порівняння ефективності моделей, Експеримент 1.3

В порівнянні з Експериментом 1.1, приріст оцінки F1 склав майже 6% для SVM, та майже 25% для KNN, у RF показник змінився несуттєво.

3.3.4 Експеримент 2.1

На датасеті Email, з застосуванням лематизації та векторизації за допомогою Bag of Words, без тюнінгу гіперпараметрів, отримано результати, продемонстровані в таблиці 3.5 та на рисунку 3.18:

Таблиця 3.5 — Порівняння ефективності моделей, Експеримент 2.1

Хар-ка\Модель	**NB**	**SVM**	**KNN**	**RF**
Час тренування, с	0.019	66.367	0.005	35.708
Accuracy	0.928	0.629	0.775	0.964
Precision	0.893	0.988	0.652	0.949
Recall	0.929	0.057	0.913	0.96
F1-Score	0.911	0.107	0.761	0.954
Матриця невідповідностей	[[2103 163] [104 1360]]	[[2265 1] [1381 83]]	[[1554 712] [128 1336]]	[[2191 75] [59 1405]]

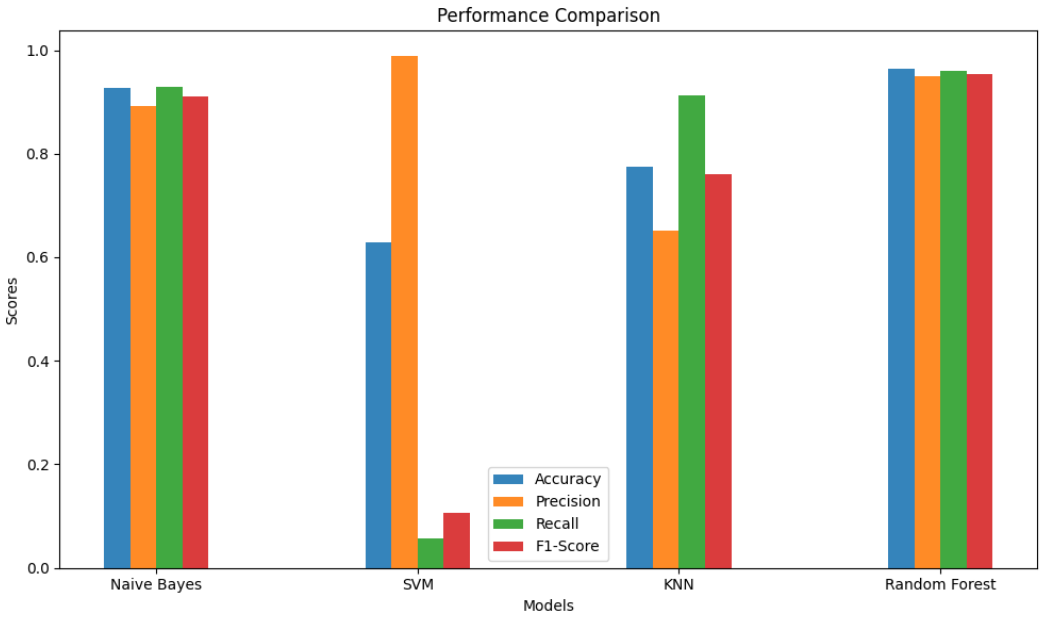


Рисунок 3.18 — Порівняння ефективності моделей, Експеримент 2.1

Як бачимо, з’явилися труднощі з класифікацією, особливо для SVM, і не такі великі для KNN, результати NB та RF досить непогані. Спробуємо змінити метод векторизації.

3.3.5 Експеримент 2.2 (векторизація TF-IDF)

Змінивши лише метод векторизації на TF-IDF, не змінюючи інших параметрів з Експерименту 2.1, отримано результати, зображені в таблиці 3.6 та на рисунку 3.19.

Таблиця 3.6 — Порівняння ефективності моделей, Експеримент 2.2

Хар-ка\Модель	**NB**	**SVM**	**KNN**	**RF**
Час тренування, с	0.02	131.848	0.006	33.913
Accuracy	0.924	0.968	0.774	0.959
Precision	0.994	0.953	0.989	0.947
Recall	0.811	0.966	0.43	0.949
F1-Score	0.894	0.96	0.599	0.948
Матриця невідповідностей	[[2259 7] [276 1188]]	[[2197 69] [50 1414]]	[[2259 7] [835 629]]	[[2189 77] [75 1389]]

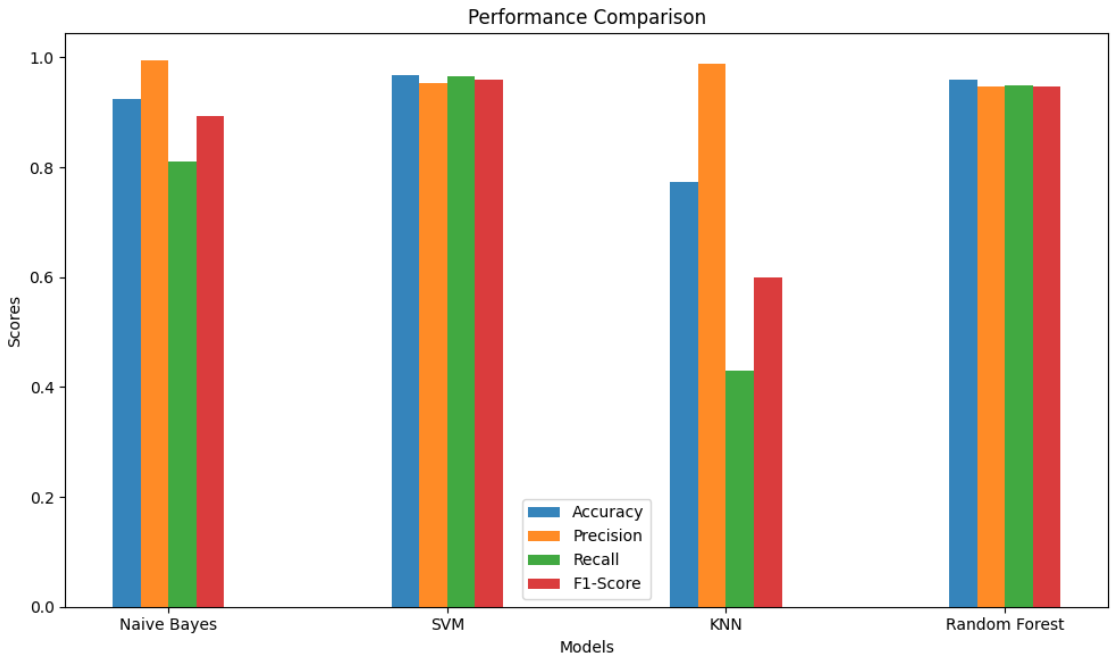


Рисунок 3.19 — Порівняння ефективності моделей, Експеримент 2.2

Дуже значно покращилися результати роботи SVM, його показники наблизилися до 1, на 21% погіршилися результати KNN, NB та RF змінилися незначно.

3.3.6 Експеримент 2.3 (з тюнінгом гіперпараметрів)

Провівши тюнінг гіперпараметрів, не змінюючи інших параметрів, отримано наступні результати:

Таблиця 3.7 — Порівняння ефективності моделей, Експеримент 2.3

Хар-ка\Модель	**NB**	**SVM**	**KNN**	**RF**
Час тюнінгу гіперпараметрів, с	N/A	1891.164	1684.639	466.769
Час тренування, с	0.017	35.208	0.011	71.026
Accuracy	0.924	0.973	0.55	0.961
Precision	0.994	0.95	0.466	0.947
Recall	0.811	0.982	1.0	0.955
F1-Score	0.894	0.966	0.635	0.951
Матриця невідповідностей	[[2259 7] [276 1188]]	[[2191 75] [27 1437]]	[[586 1680] [0 1464]]	[[2187 79] [66 1398]]

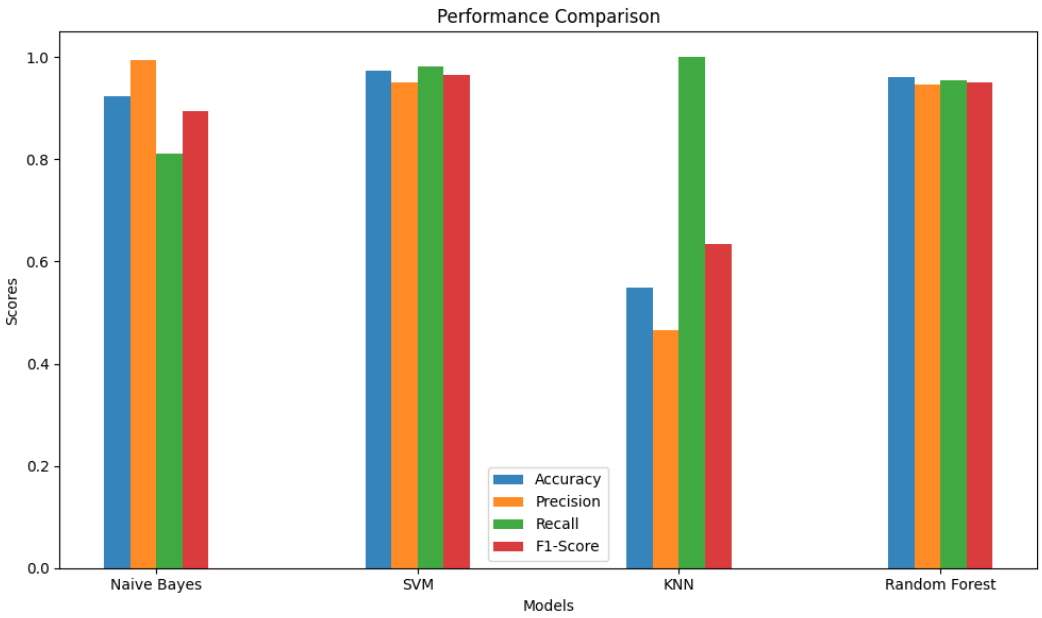


Рисунок 3.20 — Порівняння ефективності моделей, Експеримент 2.3

Показники Precision та Recall у KNN стали протилежними, при цьому показники інших моделей без суттєвих змін.

3.3.7 Експеримент 3

BERT навчався на датасеті Email протягом 15 епох, на MacBook Air 2022 (процесор Apple M2) це зайняло приблизно 550 хв. На рисунку 3.21 зображено резюме моделі, на якому видно, з яких шарів вона складається, та кількість параметрів. Рисунок 3.22 зображає процес навчання моделі.

```
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================================
Inputs (InputLayer)          [(None,)]                 0           []

keras_layer (KerasLayer)     {'input_type_ids':        0           ['Inputs[0][0]']
                             (None, 128),
                              'input_mask': (Non
                             e, 128),
                              'input_word_ids':
                             (None, 128)}

keras_layer_1 (KerasLayer)   {'pooled_output': (       109482241   ['keras_layer[0][0]',
                             None, 768),                             'keras_layer[0][1]',
                              'sequence_output':                     'keras_layer[0][2]']
                              (None, 128, 768),
                              'encoder_outputs':
                              [(None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768)],
                              'default': (None,
                             768)}

Dropout (Dropout)            (None, 768)               0           ['keras_layer_1[0][13]']

Dense (Dense)                (None, 1)                 769         ['Dropout[0][0]']

==================================================================================================
Total params: 109,483,010
Trainable params: 769
Non-trainable params: 109,482,241
```

Рисунок 3.21 — Резюме моделі

```
Epoch 1/15
467/467 [==============================] - 1605s 3s/step - loss: 0.5824 - Accuracy: 0.6923 - precision: 0.6970 - recall: 0.3840
Epoch 2/15
467/467 [==============================] - 1741s 4s/step - loss: 0.4858 - Accuracy: 0.7826 - precision: 0.7872 - recall: 0.6126
Epoch 3/15
467/467 [==============================] - 1706s 4s/step - loss: 0.4421 - Accuracy: 0.8096 - precision: 0.8077 - recall: 0.6767
Epoch 4/15
467/467 [==============================] - 1712s 4s/step - loss: 0.4136 - Accuracy: 0.8201 - precision: 0.8112 - recall: 0.7069
Epoch 5/15
467/467 [==============================] - 1708s 4s/step - loss: 0.3939 - Accuracy: 0.8355 - precision: 0.8222 - recall: 0.7418
Epoch 6/15
467/467 [==============================] - 1562s 3s/step - loss: 0.3800 - Accuracy: 0.8400 - precision: 0.8233 - recall: 0.7549
Epoch 7/15
467/467 [==============================] - 1469s 3s/step - loss: 0.3659 - Accuracy: 0.8497 - precision: 0.8320 - recall: 0.7737
Epoch 8/15
467/467 [==============================] - 8024s 17s/step - loss: 0.3571 - Accuracy: 0.8514 - precision: 0.8299 - recall: 0.7822
Epoch 9/15
467/467 [==============================] - 13076s 28s/step - loss: 0.3488 - Accuracy: 0.8542 - precision: 0.8335 - recall: 0.7860
Epoch 10/15
467/467 [==============================] - 8226s 18s/step - loss: 0.3437 - Accuracy: 0.8555 - precision: 0.8319 - recall: 0.7925
Epoch 11/15
467/467 [==============================] - 1462s 3s/step - loss: 0.3369 - Accuracy: 0.8596 - precision: 0.8353 - recall: 0.8006
Epoch 12/15
467/467 [==============================] - 1508s 3s/step - loss: 0.3307 - Accuracy: 0.8653 - precision: 0.8424 - recall: 0.8087
Epoch 13/15
467/467 [==============================] - 2372s 5s/step - loss: 0.3264 - Accuracy: 0.8651 - precision: 0.8419 - recall: 0.8085
Epoch 14/15
467/467 [==============================] - 1383s 3s/step - loss: 0.3230 - Accuracy: 0.8649 - precision: 0.8400 - recall: 0.8105
Epoch 15/15
467/467 [==============================] - 4403s 9s/step - loss: 0.3181 - Accuracy: 0.8664 - precision: 0.8415 - recall: 0.8133
<keras.engine.functional.Functional object at 0x2c2ae7ed0> Training time: 51958.508 seconds
```

Рисунок 3.22 — Процес навчання моделі

Таблиця 3.8 — Метрики BERT

Метрика	**Кількісна оцінка**
Accuracy	0.868
Precision	0.82
Recall	0.85
F1-Score	0.835

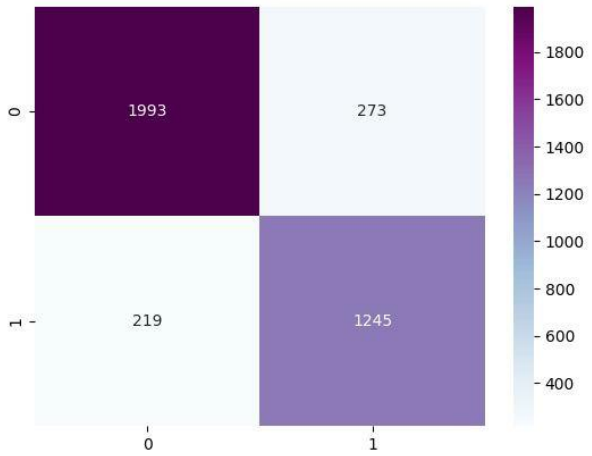


Рисунок 3.23 — Матриця невідповідностей BERT

Після перевірки ефективності роботи моделі було створено 5 легітимних та 5 спам-повідомлень, частково вручну, частково — за допомогою ChatGPT [40]. Усі 10 повідомлень були класифіковані правильно (табл. 3.9, рис. 3.22).

Таблиця 3.9 — Повідомлення для перевірки в реальному часі

Повідомлення	**Істинна мітка**	**Прогнозована мітка**
Hello! I won't be able to attend the tomorrow dinner(	Ham	Ham
Hey friend! I have a prize for you) Go to the site and claim your cash bonus$$$	Spam	Spam
Congratulations! You've won a luxury vacation for two! Claim your prize now by clicking the link below and entering your personal information. Don't miss out on this amazing opportunity!	Spam	Spam
Hi Sarah, just wanted to remind you about our lunch tomorrow at 1 PM. Let's meet at the usual spot. Looking forward to catching up with you!	Ham	Ham
URGENT: Your account has been compromised! Click here to secure your account and prevent further damage.	Spam	Spam
Hey John, I saw the movie you recommended and it was incredible! Thanks for the suggestion.	Ham	Ham
Get rich quick! Join our exclusive investment program and start making thousands of dollars in just a few days. Don't miss out on this lucrative opportunity!	Spam	Spam
Claim your prize! You've been selected as the winner of a brand new luxury car. Click the link below to collect your prize. Hurry, offer ends soon!	Spam	Spam
Hi Mark, just wanted to check if you received the meeting agenda for tomorrow's presentation. Let me know if you have any questions or need any additional information.	Ham	Ham
Hey Lisa, I hope you're doing well. I wanted to invite you to my art exhibition this weekend. It would mean a lot to me if you could make it. Looking forward to seeing you there!	Ham	Ham

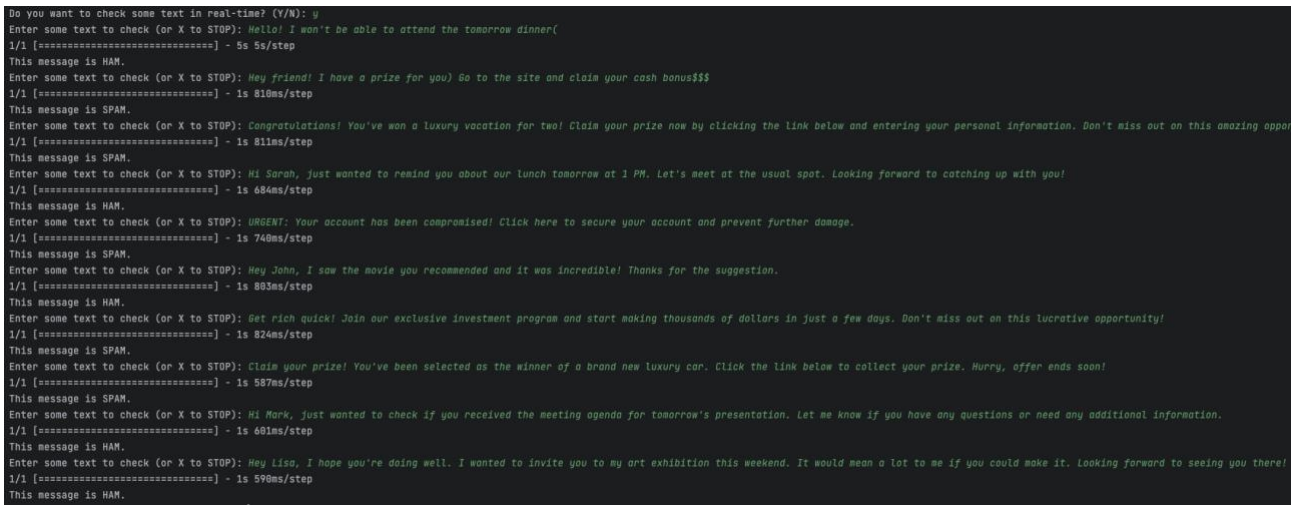


Рисунок 3.24 — Перевірка згенерованих повідомлень у реальному часі

Висновки до розділу 3

Розробка програми з використанням машинного навчання для виявлення спаму передбачає ретельний вибір датасету, попередню обробку даних та вилучення ознак. Алгоритм навчається, налаштовується та оцінюється на маркованих наборах даних, що забезпечує його ефективність у розрізненні спаму та легітимних повідомлень. Експериментальна перевірка роботи алгоритму включає аналіз метрик, порівняння з іншими моделями та висновки щодо його ефективності виявлення спаму.

Naïve Bayes вирізняється простотою та швидкістю, маючи добрі показники при класифікації. SVM є більш складним алгоритмом, точним у бінарній класифікації, але вимагає підбору гіперпараметрів. KNN є простим, але має середню точність та потребує налаштування гіперпараметрів. Random Forest є високоефективною ансамблевою моделлю, яка добре справляється з великими обсягами даних. BERT, хоча потребує багато ресурсів та часу для навчання, виявився дуже ефективним при класифікації природної мови, досягнувши високої точності при класифікації введених повідомлень.

ВИСНОВКИ

Виявлення спаму за допомогою алгоритмів машинного навчання є ефективним методом фільтрації небажаних повідомлень в електронних листах і текстових повідомленнях. Різні моделі машинного навчання застосовуються для виявлення та фільтрації спаму з різним рівнем успіху.

Досліджено поняття спаму, існуючі методи його виявлення (статичні, динамічні), зокрема застосовність до нього машинного навчання (з учителем, без учителя). Також була приділена увага поняттям попередньої обробки даних (стемінг, лематизація) та вилучення ознак (Bag of Words, TF-IDF), що є ключовими етапами у процесі класифікації спаму.

Мовою Python розроблено програмний застосунок, що, використовуючи обраний набір даних, застосовує моделі машинного навчання для класифікації спаму, та експериментально перевірено їх ефективність за кількісними метриками, такими як точність, влучність, повнота та оцінка F1, побудовано матриці невідповідності та порівняльні графіки. Це показало, що при правильному поєднанні попередньої обробки даних і вилучення ознак з різними моделями машинного навчання можна досягти високої точності класифікації.

Проведено тюнінг гіперпараметрів використаних моделей за допомогою 5-кратної кросвалідації GridSearchCV для оптимізації їхньої ефективності та досліджено вплив різних конфігурацій і параметрів, таких як спосіб попередньої обробки даних/векторизації на точність класифікації спаму.

Аналіз результатів роботи застосунку та порівняння ефективності різних моделей машинного навчання дозволили встановити, що деякі моделі можуть досягати кращих результатів у виявленні спаму, ніж інші, і важливо враховувати особливості як даних, так і моделей.

Отримані результати мають практичне значення для підприємств та користувачів, що борються з проблемою спаму. Автоматичне виявлення спаму сприяє зменшенню кількості небажаних повідомлень, підвищенню продуктивності та забезпеченню якості й безпеки комунікацій.

Подальшого вдосконалення методів виявлення спаму можна досягти шляхом розширення набору даних для покращення загальної репрезентативності, а також використовуючи моделі глибинного навчання, наприклад, CNN (Convolutional Neural Network) і RNN (Recurrent Neural Network) у поєднанні з NLP.

На закінчення, методи машинного навчання мають великі перспективи в розробці надійних і ефективних систем виявлення спаму в електронних комунікаціях. Ретельно обираючи та впроваджуючи найбільш підходящі моделі та методи, можна створити ефективні спам-фільтри, які зможуть захистити користувачів від небажаних повідомлень та потенційних загроз.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. A spam filter [Електронний ресурс] / Paul Graham. – 2002. – Режим доступу до ресурсу: http://www.paulgraham.com/spam.html
2. Пошуковий запит до Google Scholar [Електронний ресурс] – Режим доступу до ресурсу: https://scholar.google.com/scholar?start=0&q="spam+detection"+AND+"machine+learning"+AND+("message"+OR+"email"+OR+"sms")+AND+("survey"+OR+"literature+review"+OR+"meta+analysis")&hl=en&as_sdt=0,5&as_ylo=2018
3. Mohamed Amine Ferrag, Leandros Maglaras, Sotiris Moschoyiannis, Helge Janicke, Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study, Journal of Information Security and Applications, Volume 50, 2020, 102419, ISSN 2214-2126, https://doi.org/10.1016/j.jisa.2019.102419
4. Saleh, A.J.; Karim, A.; Shanmugam, B.; Azam, S.; Kannoorpatti, K.; Jonkman, M.; Boer, F.D. An Intelligent Spam Detection Model Based on Artificial Immune System. *Information* 2019, *10*, 209. https://doi.org/10.3390/info10060209
5. Kotsiantis, Sotiris & Zaharakis, I. & Pintelas, P.. (2006). Machine learning: A review of classification and combining techniques. Artificial Intelligence Review. 26. 159-190. 10.1007/s10462-007-9052-3
6. Nosseir, A., Nagati, K., & Taj-Eddin, I. (2013). Intelligent word-based spam filter detection using multi-neural networks. *International Journal of Computer Science Issues (IJCSI)*, *10*(2 Part 1), 17
7. Bhuiyan, Hanif & Ashiquzzaman, Akm & Juthi, Tamanna & Biswas, Suzit & Ara, Jinat. (2018). A Survey of Existing E-Mail Spam Filtering Methods Considering Machine Learning Techniques.
8. Text Retrieval Conference [Електронний ресурс] – Режим доступу до ресурсу: https://trec.nist.gov

9. Spam and phishing in 2022 [Електронний ресурс] // Kaspersky Lab. – 2023. – Режим доступу до ресурсу: https://securelist.com/spam-phishing-scam-report-2022/108692/
10. Global spam volume as percentage of total e-mail traffic from 2011 to 2022 [Електронний ресурс] // Statista. – 2023. – Режим доступу до ресурсу: https://www.statista.com/statistics/420400/spam-email-traffic-share-annual/
11. "Spam" sketch [Електронний ресурс] / Monty Python – Режим доступу до ресурсу: https://youtu.be/_bW4vEo1F4E
12. Liu, Pingchuan & Moh, Teng-Sheng. (2016). Content Based Spam E-mail Filtering. 218-224. 10.1109/CTS.2016.0052
13. Spamhaus [Електронний ресурс] – Режим доступу до ресурсу: https://www.spamhaus.org/
14. SURBL [Електронний ресурс] – Режим доступу до ресурсу: https://surbl.org
15. Apache SpamAssassin [Електронний ресурс] – Режим доступу до ресурсу: https://spamassassin.apache.org/
16. Anti-spam Software [Електронний ресурс] // Capterra – Режим доступу до ресурсу: https://www.capterra.com/anti-spam-software/
17. ZeroSpam [Електронний ресурс] – Режим доступу до ресурсу: https://www.zerospam.ca/
18. Python Machine Learning / Sebastian Raschka., 2015.
19. Sathya, R. & Abraham, Annamma. (2013). Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification. International Journal of Advanced Research in Artificial Intelligence. 2. 10.14569/IJARAI.2013.020206.
20. Qian, Feng & Pathak, Abhinav & Hu, Yu & Mao, Zhuoqing & Xie, Yinglian. (2010). A case for unsupervised-learning-based spam filtering. Performance Evaluation Review. 38. 367-368. 10.1145/1811039.1811090
21. Bayes' theorem [Електронний ресурс] // Wikipedia – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Bayes%27_theorem

22. Mccallum, Andrew & Nigam, Kamal. (2001). A Comparison of Event Models for Naive Bayes Text Classification. Work Learn Text Categ. 752.
23. Okfalisa, I. Gazalba, Mustakim and N. G. I. Reza, "Comparative analysis of k-nearest neighbor and modified k-nearest neighbor algorithm for data classification," *2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, Yogyakarta, Indonesia, 2017, pp. 294-298, doi: 10.1109/ICITISEE.2017.8285514
24. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [Електронний ресурс] / [J. Devlin, M. Chang, L. Kenton та ін.]. – 2018. – Режим доступу до ресурсу: https://arxiv.org/abs/1810.04805
25. OpenAI [Електронний ресурс] – Режим доступу до ресурсу: https://openai.com
26. Deep learning [Електронний ресурс] // Wikipedia – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Deep_learning
27. Transformer (machine learning model) [Електронний ресурс] // Wikipedia – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)
28. BERT (language model) [Електронний ресурс] // Wikipedia – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/BERT_(language_model)
29. Spam Text Message Classification [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/datasets/team-ai/spam-text-message-classification
30. Email Spam Dataset [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/datasets/nitishabharathi/email-spam-dataset
31. SpamAssassin Dataset [Електронний ресурс] // Apache – Режим доступу до ресурсу: https://spamassassin.apache.org/old/publiccorpus/
32. LingSpam Dataset [Електронний ресурс] – Режим доступу до ресурсу: http://www.aueb.gr/users/ion/data/lingspam_public.tar.gz

33. Enron Dataset [Електронний ресурс] – Режим доступу до ресурсу: https://www2.aueb.gr/users/ion/data/enron-spam/
34. Офіційна документація Python [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/
35. Офіційна документація NLTK [Електронний ресурс] – Режим доступу до ресурсу: https://www.nltk.org/index.html
36. Офіційна документація scikit-learn [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable/
37. Модуль попередньої обробки тексту BERT для англійської мови [Електронний ресурс] – Режим доступу до ресурсу: https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3
38. Модуль кодування BERT для англійської мови [Електронний ресурс] – Режим доступу до ресурсу: https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4
39. Офіційна документація TensorFlow [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org
40. ChatGPT [Електронний ресурс] – Режим доступу до ресурсу: https://chat.openai.com/
41. A. Karim, S. Azam, B. Shanmugam, K. Kannoorpatti and M. Alazab, "A Comprehensive Survey for Intelligent Spam Email Detection," in *IEEE Access*, vol. 7, pp. 168261-168295, 2019, doi: 10.1109/ACCESS.2019.2954791.
42. Dada, E. G., Bassi, J. S., Chiroma, H., Abdulhamid, S. M., Adetunmbi, A. O., & Ajibuwa, O. E. (2019). Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon*, *5*(6), e01802. https://doi.org/10.1016/j.heliyon.2019.e01802
43. Naeem Ahmed, Rashid Amin, Hamza Aldabbas, Deepika Koundal, Bader Alouffi, Tariq Shah, "Machine Learning Techniques for Spam Detection in Email and IoT Platforms: Analysis and Research Challenges", *Security and Communication Networks*, vol. 2022, Article ID 1862888, 19 pages, 2022. https://doi.org/10.1155/2022/1862888

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
import re
import time
from collections import Counter

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import plotly.express as px
import seaborn as sns
import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_text as text
from nltk import PorterStemmer, WordNetLemmatizer, word_tokenize
from nltk.corpus import stopwords
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.utils import resample
from wordcloud import WordCloud


def load_data(type_):
    if type_ == '1':
        df_ = pd.read_csv('data/sms_spam.csv')
        df_ = df_.rename(columns={"Category": "label", "Message": "text"})
        df_['label'] = df_['label'].map({'ham': 0, 'spam': 1})
    elif type_ == '2':
        df_1 = pd.read_csv('data/lingSpam.csv')
        df_2 = pd.read_csv('data/enronSpamSubset.csv')
        df_3 = pd.read_csv('data/completeSpamAssassin.csv')
        df_1.drop("Unnamed: 0", inplace=True, axis=1)
        df_2.drop(["Unnamed: 0", "Unnamed: 0.1"], inplace=True, axis=1)
        df_3.drop("Unnamed: 0", inplace=True, axis=1)
        df_ = pd.concat([df_1, df_2, df_3], axis=0)
        df_.dropna(inplace=True)
        df_ = df_.rename(columns={"Label": "label", "Body": "text"})
```

```
        df_ = df_[['label', 'text']]
        df_["text"] = df_["text"].str.lower()
        df_['text'] = df_['text'].apply(lambda x: re.sub('subject|from|date|fw|enron', '', x))
    else:
        print("Wrong type of data. Using SMS as default.")
        df_ = load_data('1')
    return df_


def visualize_data(df_):
    print(df_.shape, df_.head(), sep='\n')  # Розмір датасету та перші рядки

    # Розподіл класів (pie chart)
    class_dis = df_.label.value_counts()
    class_names = class_dis.index
    fig = px.pie(names=class_names,
                 values=class_dis,
                 color=class_names,
                 title='Class distribution of spam messages')
    fig.show()

    # Розподіл класів (bar chart)
    sns.countplot(x=df_["label"], data=df_)
    plt.show()

    # Хмара слів
    wordcloud = WordCloud(background_color='white').generate(df_.text.to_string())
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.tight_layout()
    plt.show()

    # Розподіл довжини повідомлень
    find_most_common_words(df_, 1)
    find_most_common_words(df_, 0)


def find_most_common_words(df_, label_):
    # Пошук найпоширеніших слів
    all_words = []
    for message in df_[df_['label'] == label_]['text'].to_list():
        for word in message.split():
            all_words.append(word)
    df_ = pd.DataFrame(Counter(all_words).most_common(25), columns=['word',
'frequency'])
```

```
    # Візуалізація
    label_ = 'spam' if label_ == 1 else 'ham'
    sns.set_context('notebook', font_scale=1.3)
    plt.figure(figsize=(18, 8))
    sns.barplot(y=df_['word'], x=df_['frequency'], palette='summer')
    plt.title(f"Most common {label_} words")
    plt.xlabel("frequency")
    plt.ylabel(f"{label_} words")
    plt.show()


def preprocess_data(df_):
    preprocessing_method = input("Choose preprocessing type (1 - Stemming, 2 -
Lemmatization, N - No Preprocessing): ")
    if preprocessing_method.lower() != 'n':
        # Видалення посилань та неалфавітних символів
        df_["text"] = df_['text'].apply(lambda x: re.sub(r"http\S+|https\S+", "", x))
        df_['text'] = df_['text'].apply(lambda x: re.sub('[^a-zA-Z\s]|\n', '', x))

    if preprocessing_method == '1':
        stemmer = PorterStemmer()  # Стемінг
        df_['text'] = df_['text'].apply(lambda x: ' '.join([stemmer.stem(word) for word in
word_tokenize(x)]))
    elif preprocessing_method == '2':
        lemmatizer = WordNetLemmatizer()  # Лематизація
        df_['text'] = df_['text'].apply(lambda x: ' '.join([lemmatizer.lemmatize(word) for
word in word_tokenize(x)]))
    else:
        print("No preprocessing is used.")

    # Видалення стоп-слів
    stop_words = set(stopwords.words('english'))
    df_['text'] = df_['text'].apply(
        lambda x: ' '.join([word for word in word_tokenize(x) if word.lower() not in
stop_words]))

    if input("Unnecessary data removed. Visualize the data? (Y/N): ").lower() == 'y':
        visualize_data(df_)
    return df_


def vectorize_data(method):
    if method == '1':
        vectorizer = CountVectorizer(lowercase=True, stop_words='english')  # Bag of
```

```
Words
    elif method == '2':
        vectorizer = TfidfVectorizer(lowercase=True, stop_words='english')  # TF-IDF
    elif method.lower() == 'n':
        print("No vectorization is used.")
        return df['text'], df['label']  # Повернення датафрейму без векторизації
    else:
        print("Invalid choice. Using Bag of Words as default.")
        vectorizer = CountVectorizer(lowercase=True, stop_words='english')

    X_ = vectorizer.fit_transform(df['text'])  # Векторизація тексту
    y_ = df['label']
    return X_, y_


def upsample_data(df_):
    # створення датафреймів більшості та меншості
    df_majority = df_[(df_['label'] == 0)]
    df_minority = df_[(df_['label'] == 1)]

    # апсемплінг меншості
    df_minority_upsampled = resample(df_minority,
                                     replace=True,
                                     n_samples=4825,
                                     random_state=42)

    # об'єднання датафреймів
    df_ = pd.concat([df_majority, df_minority_upsampled])
    return df_


def tune_hyperparameters(model_, param_grid_):
    start_time_ = time.time()
    # Пошук найкращих гіперпараметрів
    grid_search = GridSearchCV(model_, param_grid_, scoring='f1', verbose=1,
n_jobs=-1)
    grid_search.fit(X_train, y_train)
    end_time_ = time.time()
    print(f"Hyperparameter Tuning time for {model_}:", round(end_time_ -
start_time_, 3), "seconds")
    return grid_search.best_estimator_


def rate_performance(y_test_, y_pred_):
    # Оцінка ефективності моделі
```

```
    accuracy_ = round(accuracy_score(y_test_, y_pred_), 3)
    precision_ = round(precision_score(y_test_, y_pred_), 3)
    recall_ = round(recall_score(y_test_, y_pred_), 3)
    f1_ = round(f1_score(y_test_, y_pred_), 3)
    confusion_mat_ = confusion_matrix(y_test_, y_pred_)

    # Виведення метрик та візуалізація
    print("Accuracy:", accuracy_)
    print("Precision:", precision_)
    print("Recall:", recall_)
    print("F1-Score:", f1_)
    sns.heatmap(confusion_mat_, cmap='BuPu', annot=True, fmt='d')
    plt.show()


def implement_all_models(models, X_train_, y_train_, X_test_, y_test_):
    scores = {}
    tune_hyper = input("Perform hyperparameter tuning? (Y/N): ")
    # Перебір моделей
    for model_name, model_ in models.items():
        if tune_hyper.lower() == 'y':
            if model_name == 'Naive Bayes':
                pass
            elif model_name == 'SVM':
                param_grid = {'C': [0.1, 1, 5, 7, 10], 'kernel': ['linear', 'rbf', 'poly',
'sigmoid']}
                model_ = tune_hyperparameters(model_, param_grid)
            elif model_name == 'KNN':
                param_grid = {'n_neighbors': [1, 3, 5, 7, 9], 'weights': ['uniform',
'distance']}
                model_ = tune_hyperparameters(model_, param_grid)
            elif model_name == 'Random Forest':
                param_grid = {'n_estimators': [10, 20, 50, 100, 200], 'max_depth': [None,
10, 20, 50, 100]}
                model_ = tune_hyperparameters(model_, param_grid)
        start_time = time.time()
        # Тренування моделі
        model_.fit(X_train_, y_train_)
        end_time = time.time()
        print(f"Training time for {model_name}:", round(end_time - start_time, 3),
"seconds")
        y_pred_ = model_.predict(X_test_)
        # Оцінка ефективності моделі
        scores[model_name] = {
            'Accuracy': round(accuracy_score(y_test_, y_pred_), 3),
```

```
            'Precision': round(precision_score(y_test_, y_pred_), 3),
            'Recall': round(recall_score(y_test_, y_pred_), 3),
            'F1-Score': round(f1_score(y_test_, y_pred_), 3),
            'Confusion Matrix': confusion_matrix(y_test_, y_pred_)
        }
        # Виведення метрик
        for metric, score in scores[model_name].items():
            print(f"{metric}:", score)
    return scores


def build_comparison_chart(scores):
    # Отримання моделей та метрик
    models = list(scores.keys())
    metrics = list(scores[models[0]].keys())
    num_models = len(models)

    # Налаштування графіка
    bar_width = 0.1
    index = np.arange(num_models)
    fig, ax = plt.subplots(figsize=(10, 6))

    # Почергове додавання метрик до графіка
    for i, metric in enumerate(metrics):
        metric_scores = [scores[model][metric] for model in models]
        ax.bar(index + i * bar_width, metric_scores, bar_width, alpha=0.9, label=metric)

    # Налаштування графіка
    ax.set_xlabel('Models')
    ax.set_ylabel('Scores')
    ax.set_title('Performance Comparison')
    ax.set_xticks(index + bar_width * (len(metrics) - 1) / 2)
    ax.set_xticklabels(models)
    ax.legend()

    plt.tight_layout()
    plt.show()


def compare_all_models():
    models = {
        'Naive Bayes': MultinomialNB(),
        'SVM': SVC(),
        'KNN': KNeighborsClassifier(),
        'Random Forest': RandomForestClassifier()
```

```
    }

    model_scores = implement_all_models(models, X_train, y_train, X_test, y_test)
    scores = {}
    for model_name, score in model_scores.items():
        scores[model_name] = {
            'Accuracy': score['Accuracy'],
            'Precision': score['Precision'],
            'Recall': score['Recall'],
            'F1-Score': score['F1-Score']
        }
    build_comparison_chart(scores)


def realtime_check(model_):
    while True:
        text = []
        text.append(input('Enter some text to check (or X to STOP): '))
        if text == ['x'] or text == ['X']:
            break
        else:
            result = model_.predict(text)
            print(f'This message is {"SPAM" if result > 0.5 else "HAM"}.')


def detect_spam(choice):
    # Вибір моделей та гіперпараметрів
    if choice == '1':
        model = MultinomialNB()
    elif choice == '2':
        model = SVC()
        tune_hyper = input("Perform hyperparameter tuning for SVM? (Y/N): ")
        if tune_hyper.lower() == 'y':
            param_grid = {'C': [0.1, 1, 5, 7, 10], 'kernel': ['linear', 'rbf', 'poly', 'sigmoid']}
            model = tune_hyperparameters(model, param_grid)
    elif choice == '3':
        model = KNeighborsClassifier()
        tune_hyper = input("Perform hyperparameter tuning for KNN? (Y/N): ")
        if tune_hyper.lower() == 'y':
            param_grid = {'n_neighbors': [1, 3, 5, 7, 9], 'weights': ['uniform', 'distance']}
            model = tune_hyperparameters(model, param_grid)
    elif choice == '4':
        model = RandomForestClassifier()
        tune_hyper = input("Perform hyperparameter tuning for Random Forest? (Y/N):
")
```

```
        if tune_hyper.lower() == 'y':
            param_grid = {'n_estimators': [10, 20, 50, 100, 200], 'max_depth': [None, 10,
20, 50, 100]}
            model = tune_hyperparameters(model, param_grid)
    elif choice == '5':
        # Завантаження попередньо навченої моделі
        # Модуль попередньої обробки
        preprocessor =
hub.KerasLayer('https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3')
        # Модуль кодування
        encoder = hub.KerasLayer('https://tfhub.dev/tensorflow/bert_en_uncased_L-
12_H-768_A-12/4')

        # Створення вхідного шару моделі
        text_input = tf.keras.layers.Input(shape=(), dtype=tf.string, name='Inputs')
        # Попередня обробка тексту
        preprocessed_text = preprocessor(text_input)
        # Кодування тексту
        embeddings = encoder(preprocessed_text)
        # Застосування шару Dropout для регуляризації (уникнення перенавчання)
з коефіцієнтом 0.1
        dropout = tf.keras.layers.Dropout(0.1,
name='Dropout')(embeddings['pooled_output'])
        # Створення вихідного шару моделі з одним нейроном та функцією
активації sigmoid
        outputs = tf.keras.layers.Dense(1, activation='sigmoid', name='Dense')(dropout)

        # Створення екземпляра моделі з заданими вхідними та вихідними шарами
        bert = tf.keras.Model(inputs=[text_input], outputs=[outputs])
        # Виведення інформації про модель
        print(bert.summary())

        # Компіляція моделі, визначення функції втрати, оптимізатора та метрик
        bert.compile(optimizer='adam', loss='binary_crossentropy', metrics=['Accuracy',
'Precision', 'Recall'])

        start_time = time.time()
        # Тренування моделі протягом 15 епох
        bert.fit(X_train, y_train, epochs=15)
        end_time = time.time()
        print(f"{bert} Training time:", round(end_time - start_time, 3), "seconds")

        # Оцінка моделі
        y_pred = bert.predict(X_test)
        y_pred = np.where(y_pred > 0.5, 1, 0)
```

```
            rate_performance(y_test, y_pred)

            # Перевірка тексту в реальному часі
            if input("Do you want to check some text in realtime? (Y/N): ").lower() == 'y':
                realtime_check(bert)
            else:
                return 0
        elif choice.lower() == 'x':
            compare_all_models()
            return 0
        else:
            print("Invalid choice. Using Naive Bayes as default.")
            model = MultinomialNB()

        # Тренування моделі з фіксацією часу
        start_time = time.time()
        model.fit(X_train, y_train)
        end_time = time.time()
        print(f"{model} Training time:", round(end_time - start_time, 3), "seconds")

        # Прогнозування для тестових даних
        y_pred = model.predict(X_test)

        # Оцінка результатів та виведення метрик
        rate_performance(y_test, y_pred)


if __name__ == '__main__':
    # Вибір даних для аналізу
    data_type = input("Choose data type (1 - SMS, 2 - emails): ")
    df = load_data(data_type)  # Завантаження даних
    df = preprocess_data(df)  # Попередня обробка даних

    # Апсемплінг даних
    if data_type == '1' and input("Perform upsampling for the data? (Y/N): ").lower()
== 'y':
        df = upsample_data(df)
        if input("Upsampling performed. Visualize the data? (Y/N): ").lower() == 'y':
            visualize_data(df)

    # Векторизація даних
    vectorization_method = input("Choose vectorization method (1 - Bag of Words, 2 -
 TF-IDF, N - No Vectorization): ")
    X, y = vectorize_data(vectorization_method)
```

```
    # Розбиття даних на тренувальні та тестові
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

    # Вибір моделі та запуск аналізу
    model_choice = input("Choose model (1 - NB, 2 - SVM, 3 - KNN, 4 - RF, 5 -
BERT,  X - All at Once): ")
    detect_spam(model_choice)
```