ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Проєктування вебресурсу по виявленню спаму на основі машинного навчання»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Штучний інтелект
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Нікіта ОСТРЕНСЬКИЙ
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. ШІД-41

Нікіта ОСТРЕНСЬКИЙ

Керівник: Тетяна КИСІЛЬ
Старший викладач

Рецензент:
науковий ступінь,
вчене звання _____
Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту

Ступінь вищої освіти Бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедрою Штучного інтелекту

_____ Ольга ЗІНЧЕНКО
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Остренського Нікіті Петровича
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Розробка комп'ютерного додатку генерації зображень з використанням мереж глибокого навчання

керівник кваліфікаційної роботи Тетяна КИСІЛЬ старший викладач,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» 02.2024р. № 36

2. Строк подання кваліфікаційної роботи «31» травня 2024р.

3. Вихідні дані до кваліфікаційної роботи:
- Мова програмування - Python.
- Середовище розробки вебзастосунку – Visual Studio Code/

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

- Проаналізувати існуючі методи по вирішенню даної задачі.
- Проаналізувати необхідну теорію для розробки програмного продукту.
- Розробити програмний продукт по виявленню спаму.
- Розробити оцінку ефективності програмного продукту

5. Перелік графічного матеріалу: *презентація*
- Мета роботи, об'єкт і предмет дослідження
- Постановка заачі
- Опис використаних методів
- Результати дослідження
- Висновок

6. Дата видачі завдання «27» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	28.02-03.03.24	
2	Аналіз та дослідження існуючих аналогів	04.03-17.03.24	
3	Дослідження програмних об’єктів	18.03-27.03.24	
4	Проєктування веб-застосунку	29.02-10.03.24	
5	Розробка веб-додатку	11.03-25.03.24	
6	Вступ, основна частина, висновки, реферат, демонстраційні матеріали	26.03-13.05.24	
7	Попередній захист роботи	16.05.2024	

Здобувачка вищої освіти _____ Нікіта ОСТРЕНСЬКИЙ
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Тетяна КИСІЛЬ
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 52 стор., 30 рис., 20 джерел.

Мета роботи – розробка вебзастосунка з виявлення спаму на основі аналізу наївного Байеса та порівняння ефективності з іншими алгоритмами. Також проведення аналізу інших можливих рішень з виявленням цієї проблеми..

Об'єкт дослідження – Виявлення спаму за допомогою машинного навчання.

Предмет дослідження – Аспекти виявлення спаму в Інтернеті..

Короткий зміст роботи: У роботі проведено огляд існуючих методів виявлення спаму. Проведено порівняльний аналіз методів по боротьбі зі спамовими повіомленнями на основі машинного навчання.

В ході дослідження було розроблено веб-застосунок для виявлення спаму, що базується на алгоритмі наївного Байеса, порівняння ефективності із іншими методами машинного навчання. Реалізовано веб-інтерфейс з використанням HTML та CSS. Застосунок дозволяє виявляти спам у текстових повідомленнях та класифікувати їх з високою точністю. Він може бути використаний для покращення якості користувачів та забезпечення захисту від небажаних повідомлень у різних сферах бізнесу.

КЛЮЧОВІ СЛОВА: вебзастосунок, розробка, класифікатор

ЗМІСТ

ВСТУП

Можу запевнити, що кожен з нас, хто хоч раз користувався комп'ютером, довелося стикатися з набридливою рекламою, яка називається спамом. Іншими словами, це велика кількість повідомлень, яка приходить користувачу, навіть не даючи згоду на її отримання. Серед них зазвичай є багато реклами, відправлених від різних брендів, також це може бути повідомлення з якоюсь маніпулятивним посиланням, де в умові просять надіслати це іншим користувачам, обіцяючи позитивні або негативні наслідки в разі невиконання.

Щодо негативних наслідків, то такі повідомлення можуть містити не лише комерційний характер, але й також завдати шкоди через шкідливий вміст самого повідомлення. Такі дії з боку зловмисників можуть принести шкоду, як конкретній людині, так і задати збитки бізнесу. Можуть бути різні сценарії вплинути на продуктивність компанії, порушення роботи інфраструктури самої компанії, в разі проникнення вірусів в корпоративну мережу; погіршення репутації компанії та в результаті, це може призвести до втрати клієнтів.

Отже, враховуючи всі негативні наслідки, які завдає спам та постійний приріст таких повідомлень, можу припускати, що баєсовий додаток для фільтрації спаму, стане корисним інструментом задля безпеки користування в Інтернеті.

Саме тому, мета моєї роботи - розробка вебзастосунка з виявлення спаму на основі аналізу наївного Байеса. Даний алгоритм буде отримувати повідомлення від користувача, перевіряючи текст на можливий вміст спаму. Після визначення того, що повідомлення дійсно містить спам, воно буде видалено.

Тому, мета, яка поставлена переді мною, вимагає вирішення таких завдань, як:

- створення вебінтерфейсу додатка з можливістю введення тексту користувачем;
- розробка алгоритму наївного Байеса для аналізу повідомлення користувача з метою виявлення спаму;

- інтеграція алгоритму з вебінтерфейсом додатка;
- тестування роботи алгоритму;

Об’єктом дослідження в моїй роботі є виявлення спаму за допомогою наївного алгоритму класифікатора Байеса. Предметом дослідження являє собою аспекти виявлення спаму в Інтернеті.

З урахуванням мети, завдань, об’єкту та предмету дослідження, можу зазначити, що найбільш доцільними методами дослідження серед теоретичних являється формування гіпотези, стосовно ефективності застосунку та класифікація типів спаму. З емпіричних методів, то краще за все підійде моделювання, що передбачає створення моделі для аналізу текста повідомлень, експерименти з використання реальних вхідних даних задля тестування створеною нами моделі, і також аналіз та порівняння роботи цієї моделі з іншими методами, які використовуються для таких самих цілей.

Теоретичне значення моєї роботи заключається у дослідженні методів машинного навчання у боротьбі зі спам-повідомленнями в Інтернеті, зокрема алгоритму наївного Байеса.

Практичне значення дипломної роботи полягає у створенні вебзастосунка для виявлення та фільтрації спаму з використанням машинного навчання на основі аналізу повідомлень за допомогою алгоритму наївного Байеса. Функціонал мого застосунку можуть використовувати інші компанії в своїх проєктах задля безпеки користувачів в Інтернеті від можливого спам-повідомлення або хакерської атаки.

Наукова новизна передбачає інноваційний підхід по боротьбі зі спамом з використанням машинного навчання. Така робота в спромозі принести ефективне та автоматизоване рішення у сферу боротьби зі спамом.

Якщо підсумувати, то мій проєкт спрямований на розробку вебзастосунка з виявлення спаму з впровадженням байесового алгоритму, для зменшення кількості наслідків, яку він завдає користувачам та бізнесу в Інтернеті.

1 ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ВИЯВЛЕННЯ СПАМУ

1.1 Опис проблему спаму в Інтернеті

У сучасному цифровому світі, проблема спаму стала однією з найважливіших проблем в Інтернеті. Перший випадок, з чого почалася історія виникнення спаму, відбулася ще до 1978 року. Тоді [1] перша комерційна реклама була надіслана 400 користувачам APPANET. В той період, кількість користувачів у мережі складала приблизно 2600 осіб. Тобто, комерційна реклама охопила 15% інтернет-спільноти. Це сталося, через те, що всі користувачі даної мережі були не захищеними. Інформація про них зберігатися у фізичному каталозі та один маркетолог від компанії “Digital Equipment Corp” вирішив скористатися такою можливістю для просування продукції від компанії.

На сьогодні, ця проблема досі залишається. Серед основних типів спаму є електронний спам, месенджер спам, спам у коментарях, браузерний спам, трекбековий спам та негативне SEO.

Електронний спам дуже нам знайомий. Враховуючи статистику [2] (рис. 1.1), 85% серед всіх повідомлень, що приходять до електронної скриньки користувача є спамом. Хоча, враховуючи тенденції, можна відзначити, що за останній час кількість таких повідомлень зменшується, але це все одно великий показник. Спам, який може прийти на почту різний, серед них можуть бути: фішингові атаки, оголошення, листи з дорослим контекстом та шахрайські повідомлення.

Фішингові атаки або підроблені електронні листи використовуються зловмисниками задля того, щоб отримати дані користувача, такі як паролі, дані кредитної картки або інша особиста інформація. Такі повідомлення маскуються під довірливу особу для користувача або компанію. Фішингові повідомлення можуть включати різну різні форми і бути відправленими не тільки на електронну скриньку користувача, а й також в соціальних мережах та месенджери.

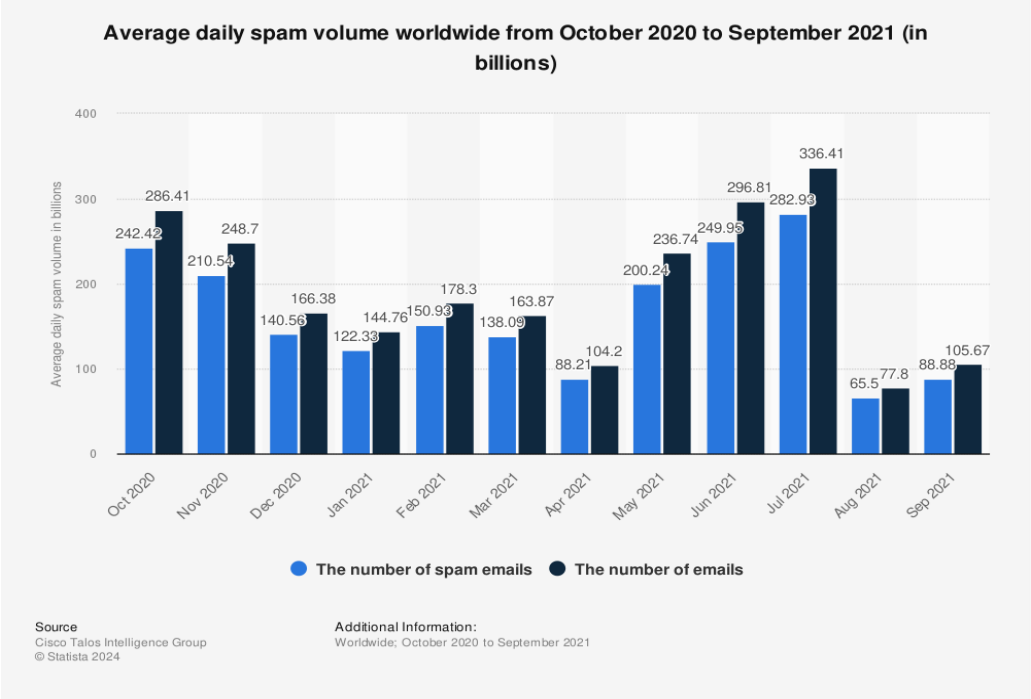


Рис. 1.1 Середній щоденний обсяг спаму в усьому світі з жовтня 2020р. по вересень 2021р.

Непотрібні повідомлення або оголошення не несуть в собі такої шкоди, вони являти собою небажану рекламу. Це може бути як пропозиція якоїсь послуги, так і продукт компанії.

Мета таких повідомлень є привертання уваги користувача для залучення цільової аудиторії та покращення продажу товару. Листи з дорослим контекстом мають популярність через обмежену рекламу і таким чином намагається просувати, використовуючи спам. Зазвичай вони містять такі послуги або товари, які люди не зручно придбати особисто. Шахрайські повідомлення здаються схожими до фішингових, але вони використовують іншу стратегію, намагаючись бути привабливими для отримувача.

Месенджерний спам також можна віднести до найпопулярніших спам-повідомлень. Їх ми можемо побачити, користуючись популярними соціальними мережами, такі як Instagram, Facebook Messenger, LinkedIn, Telegram, WhatsApp і т.д. Шкода від такого типу повідомлень відчутна, бо її важче відфільтрувати. А сам вид спаму є дієвішим ніж електронний, через те, що користувачі більше часу

проводять в соціальних мережах, а отже і з більшою ймовірністю стикаються з такими повідомленнями.

Спам у коментарях полягає у відправці небажаних коментарів, які зовсім не пов’язані зі змістом, чи то вебсайту, форуму або блогу. Ім’я коментатора, який відправляє дане повідомлення не відповідає дійсності. Метою таких дій може бути підвищення трафіку, кращий рейтинг сайту або просування реклами.

Браузерний спам передбачає сповіщеннями на різних сайтах, що є не менш докучливими ніж електронні повідомлення. Це може включати вікна зі вспливаючими повідомленнями, автоматичне перенаправлення на інші вебресурс або відкриття нових вкладок з рекламою.

Трекбек - це сповіщення автора вебконтенту про те, хтось на якомусь іншому ресурсі посилається на його контент. Таким чином створюється зв’язок між двома вебсайтами. Першочергово вони були створенні задля корисливих цілей, але згодом перетворилася на вид заробітку для спамерів і проблемою для вебмайстер, які хочуть бути захищеними від небажаних зв’язків.

Негативна SEO- [3] атака має стратегію, що спрямована на зниження рейтингу сайту. Метою такого виду хакерської атаки є те, що веббраузер підозрює, що власник сайту використовує шкідливі методи оптимізації для покращення свого сайту та підвищення в рейтингу. Зловмисники можуть створювати шкідливі зворотні посилання конкурентним вебсайтам, додавання шкідливого програмного коду. Такі дії можуть призвести до серйозних наслідків. Прикладом може бути втрата трафіку сайту, що в результаті призводить до втрати прибутковості, або погіршення репутації сайту власника.

Окрім того, шахраї, які використовують спам, для своїх цілей, можуть залучати технології з використання штучного інтелекту. Це допомагає збільшити ефективність таких атак та персоналізувати для конкретного користувача. Навіть захищенні ресурси, зазвичай не можуть повністю гарантувати захист користувачеві та відфільтровувати всі відомі нам типи спаму.

1.2 Значення виявлення спаму для користувачів та бізнесу

Після того, як будь-яка з версій спаму потрапить до користувача, він одразу стає мішенню. Такий варіант подій є високоймовірний. Саме тому розв'язування цієї проблеми є вкрай важливою.

Всі такі методи впливають, як на користувача, так і на бізнес компаній в Інтернеті і його не можна недооцінювати. Основним є те, що погіршується продуктивність робочого процесу, відволікання уваги на не потрібні повідомлення і витрачення дорогоцінного часу на перегляд всіх листів, явно знижує ефективність. Окрім цього, після відкриття такого повідомлення працівником компанії або звичайному отримувачу, він дає зрозуміти, що його адреса електронної пошти є активною. Таким чином, зловмисники можуть завантажити шкідливе програмне забезпечення, а користувач відкриє доступ, перейшовши на вірусне посилання.

Не мало важливою частиною є той факт, що електронні спамові повідомлення можуть завдавати шкоди для репутації бренду. Такі дії можуть трапитись за допомогою фішингових або оманливих повідомлень. Такий лист часто містить зловмисні посилання, які вводять в оману користувача, що в результаті призводить до порушення безпеки й отримання зловмисника особистих даних. Клієнт може цього навіть не зрозуміти, бо повідомлення буде відправлено з домену компанії, через що порушує довіру між користувачем та впливає на авторитет. Всі ці дії призводять до значних витрат компаній. Оскільки їм приходиться додатково вирішувати такі проблеми, за допомогою посилення захисту та наймання працівників, щоб вони могли забезпечити оптимальні рішення.

Ще одна сторона, де шкідливі наслідки мають суттєве значення, є той факт, що у 2021 році понад 45% електронного трафіку свого світу припадало на спам. Це створює проблеми для самої інфраструктури Інтернету, що несе за собою перевантаження серверів, зниження продуктивності, а отже підкреслює необхідність задля підсилення заходів для боротьби з даною проблемою.

1.3 Методи виявлення спаму

Завдяки активній боротьбі з цією проблемою, існують різні методи з виявлення спаму. Серед традиційних підходів, які є наразі, можу зазначити спам-фільтри, евристичний аналіз та блеклистинг.

1.3.1 Спам-фільтри

Метод, який використовує спам-фільтри для сортування важливих повідомлень з не важливими і допомагає відсіювати листи, щоб витрати часу користувача. Фільтри бувають серверні, клієнтські та сторонні [4]. Серверні фільтри розміщуються на сервері електронної пошти і перевіряють листи ще до того, як він надходить до отримувача. Такий тип фільтра є ефективним у блокуванні великої кількості спаму. Серед недоліків, це те, що доступ до керування та налаштування такого фільтра є лише в адміністратора, який слідкує за серверами. Клієнтський тип відрізняється тим, що він вже встановлюється на пристрої отримувача напряму. Він може бути зручним, з погляду налаштування під власні потреби користувача. Проте клієнтські аналізують вміст повідомлення на наявність спаму вже після отримання, тому в ефективності вони поступаються серверному типу. Третій тип фільтра аналізує не тільки вміст повідомлення, а й репутацію відправника, щоб визначити, чи являється дане повідомлення спамом. Доступ до таких фільтрів можна отримати через веббраузер. Такий фільтр є більш ефективним ніж клієнтський та серверний, через вміст більшої кількості технологій, що він використовує.

Фільтри містять в собі комбінацію певних дій, які допомагають їм виявити шкідливе повідомлення. Зазвичай вони аналізують вміст, ключові слова, репутацію відправника та сигнатуру. Аналіз вмісту передбачає виявлення характерних ознак спаму за загальною структурою самого листа. Такий варіант передбачає концентруватися на великих символьних чи текстових блоках повідомлення, підозрілим полям, не правильній граматиці або підозрілим полям. Аналіз ключових

слів теж є доволі дієвим способом для ідентифікації таких повідомлень, бо спам за схожою тематикою зазвичай маж комбінацію схожих фраз або слів. Метод з аналізом репутації відправника аналізує IP-адресу, того хто відправив це повідомлення, рейтинг надійності та історію відправника. Збираючи ці дані, фільтр вирішує, чи варто блокувати спамера, якщо він таким являється.

І ще одна комбінація, яку використовує фільтр для пошуку спаму, це аналіз за сигнатурою. Для цього спам-фільтр використовує базу даних сигнатур, яка містить зразки спаму. Фільтр порівнює повідомлення із вже заготовленими шаблонами у базі даних, щоб виявити збіг зі спамом.

1.3.2 Евратичний аналіз

Евратичний аналіз [5] допомагає боротися із такими видами спаму, як фішингові атаки, звичайний спам, листи зі шкідливим програмним забезпеченням, атаки, що націлені на відмову в дієздатності системи та інші види атак. Тому такий метод дозволяє підвищити рівень безпеки електронної пошти і є ефективним у виявленні невідомих загроз. Такий метод працює за правилами, що допомагають визначити незвичну поведінку повідомлень. Він спроможний розпізнавати посилання, що містять підозрілий код, що завдає шкоди користувачу. Також компонент штучного інтелекту, який може допомогти такому аналізу ще краще вираховувати всі моделі поведінки шкідливих повідомлень. В результаті такий метод стає більш точним та ефективним у протидії кіберзагрозі.

1.3.3 Блеклистинговий метод

Блеклистинговий метод передбачає створений список, в який потрапляють IP-адреси або домени, що підозрюються у розсилці спаму. В результаті, вони блокуються і всі подальші повідомлення характеризуються за спам. Такі списки створюються та керують їми компанії безпеки. Методи, за якими працюють такі списки відрізняється від підходів самої компанії. Всі вони формуються за допомогою найкращих методів безпеки та спостережень на виявлення певних загроз. Відправник може дізнатися, що він знаходиться в заблокованому листі, у разі, якщо його повідомлення повертаються або потрапляють у спам отримувач. Якщо повідомлення повертається, то в описі додається діагностична інформація, що вказує на причину, через що, відправник перебуває в такому списку.

Однак технології не стоять на місті й для покращення виявлення спаму, активно використовується машинне навчання, що є вкрай ефективним. Популярними методами є алгоритм наївного байеса, методи опорних векторів (SVM) або нейронні мережі, ансамблеві методи або застосування глибокого навчання.

Розповсюдженим методом серед них є Байесівський класифікатор. Такий підхід використовує статистичні методи, яке працює за принципом ймовірності для визначення, чи є повідомлення спамом, чи ні. Тому дана концепція має назву "наївний", а ще й вона була названа в честь Наївного Байеса, який сформулював теорему ще наприкінці 18 століття. Згодом вона почала використовуватись у машинному навчанні та адаптована у додатках зі штучним інтелектом. Даний алгоритм перевіряє вміст повідомлення на наявність слів, які використовуються у спамових листах. Для використання створюється модель, що навчається. Вона вивчає фрази та слова у тестових листах, які зазначені, як спам спеціально для навчання. Така методика є вкрай надійною та стійкою. [2] Доказом цього є те, що навіть соціальні медіаплатформи, такі як Pixelfed, використовує класифікатор наївного байеса. Компанія стала центром для візуального обміну та спілкування, вирішила захищатися від спаму, використовуючи даний алгоритм. Аналізувавши

підписи до зображень, спам-фільтр з використанням байесового класифікатора визначав, чи містить публікація справжній вміст, чи намагається просувати небажану рекламу та нерелевантну інформацію.

1.3.4 Метод опорних векторів

Метод машинного навчання опорних векторів (SVM) є теж вкрай хорошим варіантом для боротьби зі спамом. Вони добре справляються з багатьма завданнями класифікації.[3] Коли проводиться порівняння між глибоким навчанням та іншими методами, такі як “SVM”, часто виявляється, що різниця не велика. Одна з переваг методу опорних векторів полягає в тому, що вона надзвичайно інформативна і з легкістю визначає, яка конкретна ознака впливає на процес. Цей метод добре здатен працювати доволі ефективно з великою кількістю інформації. Також машина опорних векторів вправно справляється з невідомими даними та новими типами загроз після тренування на основі навчальних даних.

1.3.5 Ансамблевський метод

Ансамблевий підхід [7] своєю чергою використовують декілька моделей, замість однієї, як зазвичай. Такий підхід дозволяє більш точно отримувати результат. Ансамблева концепція поділяється на дві категорії: послідовні та паралельні. У кожної категорії є свої певні види. Послідовні категорії ансамблевого підходу використовують бустрінг та стекінг. Метод бустінгу послідовного ансамблю заснований на концепції навчання слабких моделей, які послідовно виправляють помилки попередніх. Кожна нова модель фокусується на тих моделях, що припустилися помилок. Стекінг прогнозує кілька базових моделей, які використовують вхідні дані. Вони навчаються передбачити кінцевий результат на основі цих прогнозів. Такий метод також комбінує різні типи моделей та використовує їх комбінації для підвищення точності передбачень. До паралельних ансамблей можна віднести багінг або “укладання в мішок” та “випадкові ліси”.

Багінг заснований на побудові основних моделей, кожна з яких навчається індивідуально на випадкових підмножинах даних та методі бустсрапінгу. Це метод вибору зразків з усієї групи за допомогою випадкового відбору з повторенням. Такий підхід дозволяє зробити вибірку випадковою, а потім застосувати базовий алгоритм навчання для отримання зразків.

1.3.6 Метод глибокого навчання

За допомогою глибокого навчання створюються ефективні моделі для боротьби зі спамом [9]. Такі моделі використовуються для виявлення, аналізу та класифікації спаму в будь-яких комунікаційних каналах. Глибоке навчання відрізняється тим, що воно також допомагає прогнозувати ризики, які пов’язані зі спамом. Це відбувається через аналіз шкідливого коду, та передбачення різних типів наслідків, таких як втрата даних або вразливість комп’ютерної системи. Після аналізу тексту, моделі глибокого навчання класифікують повідомлення як спам чи не спам і не пропускають користувачу, в разі негативного результату. Однією з ключових переваг глибокого навчання є те, що моделі можуть працювати в реальному часі. Це дозволяє виявляти шкідливі повідомлення миттєво після їх надходження. Тому це забезпечує швидкий та ефективний захист користувача.

1.3.7 Метод захисту від спуфінгу

Спуфінг полягає у тому, щоб зробити повідомлення виглядом звичайним. Хоча це менше стосується спаму, ніж шахрайства, проблема залишається актуальною. SMTP не автентифікує відправника електронних листів, що дозволяє спамерам надсилати листи з підробленими або існуючими електронними адресами, маскуючись під авторитетну особу. Для вирішення цієї проблеми розроблені методи аутентифікації відправників.

1.3.7.1 DKIM

DomainKey Identified Mail (DKIM) - [4] це криптографічний метод перевірки цілісності електронної пошти та аутентифікації за допомогою інфраструктури відкритого ключа та цифрового підпису. Повідомлення електронної пошти підписуються приватним ключем домену відправника, а домен одержувача перевіряє підпис за допомогою відкритого ключа, доступного через DNS.

1.3.7.2 SPF

Sender Policy Framework (SPF) - обмежує можливість підробки адреси відправника. Використовуючи цей механізм, домен відправника публікує список авторизованих IP-адрес, яким дозволено надсилати електронні листи з цього домену. Одержувач перевіряє DNS-запис відправника, щоб знайти авторизовані IP-адреси, і відхиляє листи з неавторизованих адрес.

1.3.7.3 DMAR

Domain-based Message Authentication, Reporting, and Conformance (DMARC) –[8]об'єднує механізми DKIM та SPF, дозволяючи відправнику публікувати політику аутентифікації електронної пошти та налаштування для перевірки повідомлень, розподілу та звітності про невдалі аутентифікації. У 2015 році лише 26% вхідного трафіку Gmail надходило з доменів з політикою DMARC, а лише 1,1% з мільйона вебсайтів Alexa мали таку політику. Хоча 81% доменів підтримують DKIM і SPF, тільки 11% підтримують лише SPF, а близько 2% - лише DKIM, що обмежує ефективність цих методів.

1.4. Висновок розділу 1

В першому розділі було розглянуто проблему спаму в Інтернеті. Також було наведено приклади, як це може зашкодити. Аналізуючи ситуацію, можна зробити висновку, що це актуальна проблема, яка становить небезпеку користувачам та бізнесу у різних сферах. Методи виявлення спаму містити різноманітні техніки, від традиційних підходів до використання машинного навчання для протидії та захисту користувачів від цих небезпек. Було розглянуто, як ці методи працюють та користь використання кожного підходу.

Виявлення спаму має велике значення для безпечного перебування користувачів та бізнесу в мережі. Розробка вебресурс на основі машинного навчання з найрозповсюдженішим методом наївного Байеса може значно підвищити захист по виявлення спаму. Враховуючи специфіку задачі, основною перевагою є його ефективність роботи з текстом, швидко аналізуючи текст повідомлення.

2. ВИКОРИСТАННЯ АЛГОРИТМУ НАЇВНОГО БАЙЕСА В МАШИННОМУ НАВЧАННІ

2.1 Огляд алгоритму наївного Байеса

Наївний Байеса – [9] це набір алгоритмів навчання, засновані на теоремі Байеса (2.1), що ґрунтуються на припущеннях про незалежність ознак. Це сімейство алгоритмів, де їх об’єднує загальний принцип. Тобто кожна пара ознак, що класифікуються, не залежать одна від одної. Однак, кожна варіація наївного Байєса враховує різні типи даних та умови задачі.

Перш за все, ознайомимось із важливою концепцією в теорії ймовірності та статистиці [3], яка називається теорема Байеса.

$$P\,(S \mid W\,) \;=\; \frac{P\,(\,W\;\mid S\,)\;*\;P\,(\,S\,)}{P\,(\,W\,)} \tag{2.1}$$

Розглядаючи проблему спам-повідомлень з використанням даної теореми, припускаємо, що

– S – подія, що повідомлення є спамом;

– W – подія, що у повідомленні зустрічається слово маркер;

де, $P\,(S \mid W)$ – це буде ймовірність того, що повідомлення є спамом, за умови наявності слова W;

$P\,(W \mid S)$ – ймовірність, що слово W з’являється в спам-повідомленнях;

$P\,(S)$ – ймовірність, що будь-яке повідомлення є спамом (апріорна ймовірність);

$P\,(W)$ – ймовірність того, що слово W з’являється в будь-якому повідомленні;

У машинному навчанні теорема Байеса є основною для багатьох алгоритмів класифікації. Включаючи байесівського класифікатора, який передбачає категорії нових даних на основі попередніх.

Найпростішим і найефективнішим алгоритмом є байесовий класифікатор (рис. 2.1).

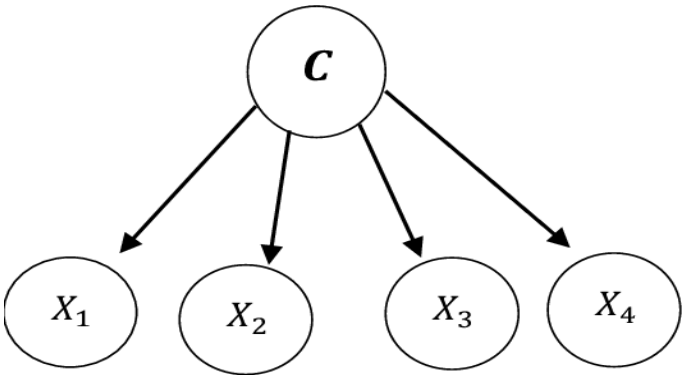


Рис. 2.1 - Найпростіший наївний байесовий класифікатор

Формується такий класифікатор на теоремі Байеса і робить “наївне” припущення про незалежність особливостей [10]. Для заданого набору даних з n особливостей $x_1, x_2, \ldots, x_n$ та класом C, класифікатор наївного Байеса обчислює ймовірність належностей на основі теореми Байеса (рис. 2.2)

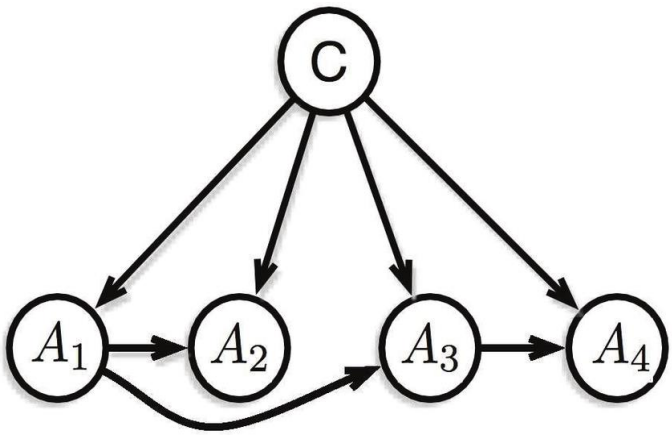


Рис. 2.2 - Аргументований наївний байесівський класифікатор

Аргументований класифікатор наївного Байеса - це варіація традиційного байесівського класифікатора, який включає додатковий шар аргументації в процесі класифікації. Це може бути корисно для підвищення точності моделі.

Такий класифікатор враховує залежності між ознаками. Особливо важливо, коли вони пов’язані та їх відношення має значення для точності класифікації.

2.2 Використання наївного Байеса для класифікації тексту

Основна задача, яка стоїть перед нами, це створити якісну класифікацію тексту. Саме для цього ми будемо використовувати класифікатор наївного Байеса. Він здатний аналізувати текст і автоматично присвоювати йому категорії на основі вводящих даних.

2.2.1 Процес підготовки даних

Важливою складовою для правильного навчання моделі є підбір текстових даних. Вони будуть використовуватись для аналізу та класифікації. Дата-сет повинен відповідати певному направленню для навчання і не може бути загальним для конкретної моделі.

Далі відбувається токенізація тексту. Цей процес полягає у розподіленні тексту у структурований формат. Тобто розподілення тексту на слова. Модель машинного навчання, яку ми тренуємо працює вже із розбитим текстом. Токенізація буває різна і текст може бути розбитий на слова, під слова та символи. Також токенізація допомагає прибрати зайву інформацію. Ми можемо позбутися розділових знаків, спеціальних символів, що значно допомагає зменшити розмір вхідних даних та покращити якість аналізу тексту. А отже, це є важливим етапом перед навчанням або вирішенням конкретних завдань для моделі.

Ще одним важливим аспектом є розподілення даних на навчальну та тестову вибірку. Це дозволяє оцінити, наскільки модель ефективно справляється з невідомими даними та зрозуміти, що вона є навчаною. Навчальна вибірка містить вхідні дані з відповідними позначками, щоб модель могла зорієнтуватись, які дані є позитивні, а які негативні. Тестова вибірка представляє невідомі дані, для того, щоб модель могла сама приймати рішення, базуючись на отриманому досвіді після навчання. Розподіл загальної кількості даних зазвичай відбувається у співвідношенні 80% навчальних даних до 20% тренувальних даних.

2.2.2 Види байесових класифікаторів

Розуміння різних видів наївного байесівського класифікатора, допоможе при виборі методу, що найбільш оптимально підійде для конкретної задачі.

Мультиномінальний наївний Байес є одним із двох класичних варіантів із сімейства Байеса для класифікації тексту. Зазвичай використовується для розподілених даних з лічильними ознаками або текстовими даними.Такий вид передбачає, що кожен об'єкт вектором, де кожний компонент відповідає кількості входжень конкретної ознаки в документі. Даний підхід ефективно працює з текстовими даними, використовуючи частоту слів для передбачення ймовірності того, що текст належить до позитивного чи негативного класу. Мультиномінальна модель виглядає наступним чином:

$$p\ (W \mid C_k)\ =\ \Pi_{i=1}^{n}\ p\ (w_i \mid C_k) \tag{2.2}$$

де, $p\ (W \mid C_k) -$ ймовірність набору ознак W за умови класу C_k;

$\Pi_{i=1}^{n} -$ добуток ймовірностей для всіх ознак від $i = 1$ до n;

Формула використовується для обчислення ймовірності, що послідовність слова W належить до класу C_k . Кожне слово w_i розглядається окремо і для кожно із них обчислюється ймовірність зустрічі у класі C_k. Коли ми вже знайшли значення ймовірності, її можна обчислити у подальшому аналізі у логарифмічному просторі, що робить обчислення простішим та ефективним:

$$\log\ \Pr\ (C_k \mid W)\ \epsilon \log\ \rho\ (C_k)\ +\ \sum_{i=1}^{n} w_i\ *\ log\rho\ (w_i \mid C_k) \tag{2.3}$$

де, $\log\ \Pr\ (C_k \mid W) -$ логарифм апостеріорної ймовірності класу C_k за умови набору ознак W;

$\log \rho\ (C_k) -$ логарифм апріорної ймовірності класу C_k;

Також у випадку, що певний клас і ознака не зустрічалися разом у навчальних даних, модель призначає ймовірність 0. Це може бути проблемою, оскільки при розрахунках ймовірності певного елемента до класу, ймовірності множаться між

собою. Це може ускладнювати правильний прогноз. Розв'язання такої проблеми може бути згладжування Лапласу. Формула виглядає таким чином:

$$\theta_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n} \quad (2.4)$$

де, θ_{yi} – умовна ймовірність ознаки i за умови класу y;

N_{yi} – кількість разів, коли ознака i з'являється в документах;

α – параметр згладжування;

N_y – загальна кількість всіх ознак у документах класу y;

y - загальна кількість можливих ознак в усьому корпусі даних;

Суть формули Лапласа полягає в додаванні "фіктивних" спостережень до кількості спостережень кожної категорії. Це допомагає уникнути ситуації, коли ймовірність виникнення якої-небудь події стає нульовою на основі неповних даних. Тому такий метод може бути корисним у випадку роботи з обмеженими обсягом даних, коли кількість спостережень є малою.

Отже, мультиноміальний підхід є оптимальний для вирішення проблеми зі спам повідомленнями. Він оцінює ймовірність появи певного слова у спамових або звичайних повідомленнях. Його використання може потребувати додаткової обробки та аналізу для досягнення результатів.

Інший вид класифікатора наївного байеса є гауссівський алгоритм. Даний варіант використовується для класифікації даних, де ознаки мають нормальний розподіл. Де ймовірність ознак передбачається гауссівська:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \quad (2.5)$$

де, $\frac{1}{\sqrt{2\pi\sigma_y^2}}$ – це нормувальний коефіцієнт, який забезпечує нормалізацію ймовірності. Залежить від стандартного відхилення y. Таким чином допомагає привести значення до відповідного діапазону.

$exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$ – описує, як швидко ймовірність змінюється з віддаленням ознаки x_i.

Формула гаусівського класифікатора формується на припущеннях, що ознаки у зразках мають гаусівський розподіл. Через це він може бути не таким ефективним у пошуку спам повідомлень. Бо присутність великої кількості слів ускладнює роботу гауссівського методу. Також такий метод буде погано навчатися з обраним дата-сетом, через те, що такий підхід краще працює при збалансованій кількості даних для навчання. У наборі даних для тренування моделі по виявленню спаму, більшість зразків мають звичайні повідомлення.

Також одним із популярних методів є байесівський класифікатор Бернуллі. Даний алгоритм працює за бінарним принципом 0 і 1. Це в свою чергу означає або слово присутнє або ні.

Метод працює за таким правилом:

$$P\,(y \mid x) = P\,(x_i = 1 \mid y)\,x_i + (1 - P\,(x_i = 1 \mid y))\,(1 - x_i) \qquad (2.6)$$

де, $P\,(y \mid x)$ – це ймовірність того, що змінна y приймає певне значееня, враховуючи значення x;

$P\,(x_i = 1 \mid y)$ – ймовірність, що змінна x маж значення 1;

x_i – зміна яка приймає значення 1;

$1 - P\,(x_i = 1 \mid y)$ – ймовірність, що змінна x маж значення 0;

$(1 - x_i)$ – протилежне значення від x_i ;

Такий метод є ефективним і може працювати з наборами даних, не такими великими за обсягом. Тому метод Бернуллі підходить для боротьби зі спам повідомленнями, бо є корисним у завданнях, де ознаки можуть бути представленні або відсутні.

2.3 Аналіз аспектів використання наївного Баєса

Важливим аспектом є ефективна обробка рідкісних та нейтральних слів, які можуть порушувати якість кластеризації документа. Одним із рішень є TF-IDF метод.

(Term Frequency-Inverse Document Frequency є [12] найпоширеніший метод для оцінки важливості термінів у текстових повідомлень. Він зменшує вплив загальних слів, що не несуть суттєвої інформації. Для обчислення використовується формула з двох компонентів: TF - частота терміна (2.7) та IDF - зворотна частота документа (2.8)

$$TF(t,d) = \frac{ft,d}{n_d} \tag{2.7}$$

Term Frequency (TF) - вимірює частоту появи терміна в документі. Тобто вираховує локальну міру важливості терміна для конкретного документа,

де, $f_{t,d}$ – кількість поява терміна t в документі d;

n_d – загальна кількість термінів у документі d;

Inverse Document Frequency (IDF) - переважно оцінює, вагу слово, а саме, наскільки воно є значущим. Головна мета пошуку - знайти записи, які відповідають запиту.

$$idf(t,D) = log\ \frac{N}{n_t} \tag{2.8}$$

де, N – загальна кількість документів;

n_t – кількість документів, що містить термін t;

У нашому випадку, документи - це спам листи, а термінами являються ключові слова. Сама формула IDF працює за принципом: чим більше повідомлень містить термін, тим менше значення, оскільки він вважається менш унікальним і не таким важливим у контексті конкретного повідомлення. Проте, якщо термін зустрічається у маленькій кількості повідомлень, значення його буде високим, а також і важливість у контексті конкретного документа.

Отже, комбінуючи дві формули TF та IDF (2.9), отримуємо важливість слова у контексті повідомлення.

$$TF - IDF(t,d) = TF(t,d) * IDF(t) \tag{2.9}$$

Формула є вкрай ефективною для виявлення спамових повідомлень завдяки здатності визначати важливі терміни у текстових повідомленнях. Це сприяє на

відокремлення спам-листів шляхом оцінювання унікальності слів у контексті всього набору даних.

2.4 Недоліки використання методу наївного Байеса

Використання наївного Байеса має свої переваги, але й також певні недоліки. Саме тому, їх варто враховувати при використанні. Ознайомлення з можливими негативними сценаріями, зменшує ризик отримання не точних результатів.

2.4.1 Припущення незалежних ознак

Одним із основних недоліків класифікатора наївного Байеса у розпізнаванні тексту є припущення про умовну незалежність ознак. Даний алгоритм вважає, що всі ознаки є незалежними одна від одної для кожного класу. Таке припущення рідко відповідає реальним даним, де ознаки можуть бути взаємозалежними.

Наприклад, речення в повідомленнях можуть бути граматично пов’язані між собою. В таких випадках, припущення незалежності не відповідає дійсності. Це може призвести до помилкових оцінок ймовірностей. В результаті, модель може недооцінювати важливість певних комбінацій ознак і це призводить до помиок в класифікації.

2.4.2 Проблема нових слів

Проблема нових слів виникає коли деякі ознаки не з’являлися в навчальному дата-сеті для певної моделі. Це може призвести до некоректної оцінки ймовірностей та до неправильних рішень класифікатора.

При використанні класифікатора наївного Байеса, обчислюється умовна ймовірність кожного класу, який створений для моделі, виходячи з присутності слова B. Формула Байеса (2.1) з умовною ймовірністю класу A та наявності слова B.

Якщо слово B жодного разу не з’являлося в класі A, то P(A | B) = 0. Тобто повідомлення які містять рідкісні слова і не були у навчальному файлі, завжди класифікуються як спам, навіть якщо інші слова у цьому ж повідомленні вказують на протилежне. Тому обсяг файлу з навчальними даними недостатньо для розпізнавання всіх повідомлень з точністю. Рішенням може бути ігнорування слів.

2.4.3 Проблема нейтральних слів

Проблема нейтральних слів полягати в тому, що вони зустрічаються як у спам-повідомленнях, та і в не спам-повідомленнях з приблизно однаковою частотою. Проблема полягає в тому, що ці слова знижують точність та ефективність аналізу байесівського класифікатора. Прикладом нейтральних слів, або стоп-слів можуть бути “the” “is” “are” “and” тощо. Вони не впливають на зміст.

Спосіб розв'язання такої проблеми, може бути видалення цих слів. Але є ще один метод який використовується зменшення впливу загальних, називається він

2.5 Альтернативи наївного Байеса

Хоча алгоритм наївного Байеса є ефективним та широко використовуваним методом для класифікації тексту. Однак для реалізації даної роботи, ми не обмежимось саме цим методом. Існує багато інших методів машинного навчання, які можуть виявитися більш підходящими або ефективними в залежності від конкретних умов та характеристик даних. Для забезпечення повноти аналізу та виявлення найкращого підходу, ми розглянемо й інші алгоритми класифікації тексту.

2.5.1 Метод k - найближчих сусідів (k-NN)

Метод k - найближчих сусідів (k-NN) є одним з найпростіших алгоритмів машинного навчання . Для класифікації об'єкта використовується більшість голосів його найближчих сусідів: об'єкт належить до класу, найбільш поширеного серед цих сусідів. Значення "k" визначає кількість сусідів і зазвичай є невеликим додатним цілим числом. Наприклад, якщо k = 1, об'єкт класифікується за класом найближчого сусіда. У задачах бінарної класифікації рекомендується використовувати непарне значення k, щоб уникнути рівності голосів.

Для визначення сусідів використовується набір об'єктів з відомою класифікацією, який слугує тренувальним набором. Щоб ідентифікувати сусідів, об'єкти представляють у вигляді векторів у багатовимірному просторі ознак. Найчастіше використовується евклідова відстань, але можливе застосування інших мір відстані.

Навчальний етап алгоритму полягає в збереженні векторів ознак та міток класів навчальних зразків. Під час фактичної класифікації новий зразок представляється як вектор у просторі ознак, після чого обчислюються відстані до всіх збережених векторів, і обираються найближчі сусіди.

На (рис. 2.3) показано процес класифікації нового прикладу за допомогою методу k-NN [13]. Ліва частина графіку демонструє новий зразок (позначений як

знак питання), який треба класифікувати, поруч з навчальними зразками двох класів (клас A - сині кола, клас B - зелені кола). Середня частина графіка показує обчислення відстаней від нового зразка до всіх збережених векторів у просторі ознак. Права частина графіку ілюструє вибір найближчих сусідів для k = 3, де новий зразок класифікується на основі найпоширенішого класу серед цих сусідів.

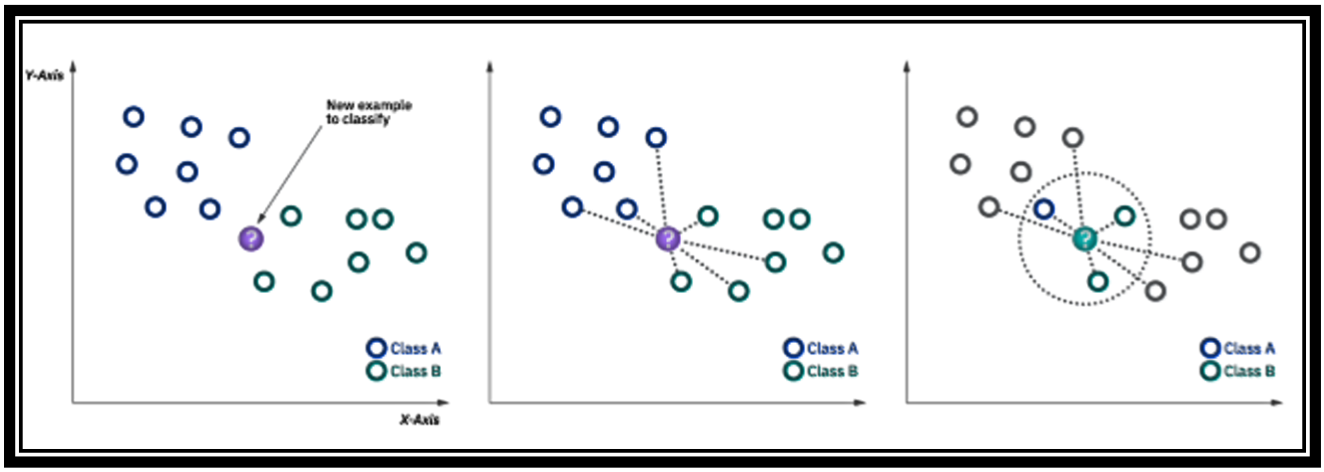


Рис. 2.3 Результати виконання методу KNN найближчих сусідів.

Найчастіше для визначення відстаней використовується евклідова відстань, але можливе застосування інших мір, таких як відстань Манхеттена. Для задач класифікації тексту можуть використовуватися інші метрики, такі як метрика перекриття або відстань Хеммінга.

Один з недоліків методу k-NN полягає в тому, що класи з більшою кількістю прикладів можуть домінувати у передбаченні. Для подолання цієї проблеми можна враховувати відстані кожного з найближчих сусідів до нового вектора, надаючи більшу вагу ближчим сусідам.

Оптимальний вибір значення k залежить від даних. Великі значення k зменшують вплив шуму, але роблять межі між класами менш чіткими. Для вибору найкращого k можна використовувати методи, такі як перехресне підтвердження. При k = 1 алгоритм називається методом найближчого сусіда.

Точність k-NN може погіршуватися через наявність шумних або нерелевантних ознак, а також через різні масштаби ознак. Щоб покращити класифікацію, можна застосовувати методи вибору або масштабування ознак,

наприклад, еволюційні алгоритми або масштабування за допомогою взаємної інформації між даними тренування та класами.

Хоча простий варіант k-NN легко реалізувати, він є обчислювально інтенсивним при великому розмірі навчального набору. Існують оптимізації, які зменшують кількість обчислень відстаней, наприклад, розділення простору об’єктів та обчислення відстаней лише в межах певних сусідніх областей.

Метод k-NN також може бути адаптований для оцінки безперервних змінних, використовуючи зважену відстань до k найближчих сусідів. У цьому випадку обчислюється евклідова відстань, зразки впорядковуються за розрахованими відстанями, вибирається оптимальне значення k на основі перехресної перевірки, і обчислюється середньозважена відстань.

2.5.2 Метод опорних векторів

Метод опорних векторів (SVM) [14] застосовуються для задач класифікації та регресії. Ця техніка машинного навчання з учителем допомагає визначити межу прийняття рішення (МПР) з максимальним зазором, що дозволяє розділити дані на два класи, залишаючи простір для можливої неправильної класифікації[].

Основна ідея SVM полягає в пошуку гіперплощини, яка максимально розділяє дані різних класів у багатовимірному просторі ознак. Це досягається шляхом максимізації відстані між гіперплощиною та найближчими точками кожного класу. На графіках нижче представлено різні варіанти SVM з різними ядрами, які використовуються для класифікації даних на основі, де представлен приклад довжини і ширини чашолистка квітів [15] (рис. 2.4).

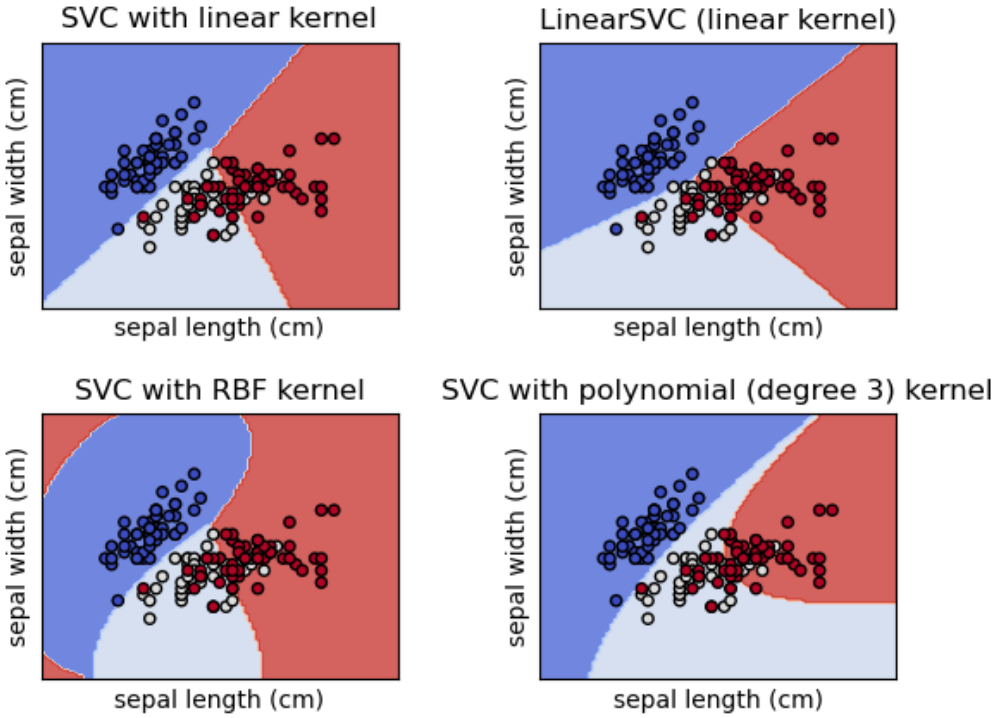


Рис. 2.4 Представлені варіанти SVM з різними ядрами для класифікації

SVM може використовувати різні ядра для створення гіперплощини, що дозволяє адаптувати алгоритм до різних типів даних. Найпоширеніші ядра включають:

– *Лінійне ядро*: Створює лінійну гіперплощину, підходить для лінійно розділених даних.

– *Радіально-базисне ядро (RBF)*: Дозволяє створювати нелінійні межі між класами, підходить для складних розподілів даних.

– *Поліноміальне ядро*: Створює полігональні межі, що дозволяє класифікувати дані зі складною структурою.

SVM визначає оптимальну межу прийняття рішення, яка мінімізує помилки класифікації та максимізує відстань між класами, забезпечуючи високу точність моделі. Завдяки цій здатності, SVM є популярним вибором для багатьох задач класифікації, де потрібна висока точність і надійність.

2.6 Висновок розділу 2

В другому розділі було проведено детальний огляд алгоритму Байеса та теоремі на якій будується навчання моделі. Також було продемонстровано як вона застосовується для розв'язання задач класифікації.

Окремо були розглянуті різні види класифікатора наївного Байеса, такі як Мультиноміальний, гауссівський та Бернуллі. Проведене порівняння вказало на те, що класифікатори Бернуллі та Мультиноміальний є більш ефективними у вирішенні проблеми спаму у повідомленнях.

Також зосередили увагу на альтернативі до наївного Байєсового класифікатора, що здатні вирішувати завдання класифікації. Метод k-найближчих сусідів (k-NN) виявився досить простим, але ефективним у випадках, коли маємо невеликий набір даних та відсутність чіткої структури. Він може бути особливо корисним у випадках, коли генерація моделі стає витратною або складною. З іншого боку, метод опорних векторів проявляє високу ефективність у вирішенні задач класифікації, особливо коли маємо справу з великими наборами даних та необхідністю у високій точності. Він відзначається здатністю підтримувати роботу з великою кількістю ознак та уникати перенавчання. У підсумку, обидва ці методи є важливими інструментами для класифікації та будуть використані у поставленій задачі, для порівняння і перевірки найбільш ефективного методу.

3. РОЗРОБКА ВЕБ-РЕСУРСУ ПО ВИЯВЛЕННЮ СПАМУ

3.1 Розробка моделі для виявлення спаму

У цьому розділі ми детально розглянемо процес рідготовки даних, а також тестування та оцінку її продуктивності. Ці кроки є необхідними для створення надійного інструменту, здатного точно ідентифікувати спам-повідомлення та запобігати їхньому проникненню на веб-ресурс.

3.1.1 Підготовка даних для навчання моделі

Для створення ефективної моделі фільтрації спаму необхідно мати достатньо великий датасет, який містить як спам-повідомлення, так і звичайні листи. Для нашого прикладу будемо використовувати датасет, який містить два типи повідомлень: "ham" (звичайні повідомлення) і "spam" (спам-повідомлення). Набір даних (рис. 3.1) має назву “Email Spam Detection Dataset”, містить приклади спамових і не спамових повідомлень, які трапляються у скриньках електронної пошти. Датасет був взятий з інтернет ресурсу Keggle [16] і містить більше 5500 листів. Другий датасет під назвою “Spam or Not Spam Dataset”,[17] менший за розміром. Набір даних містить загалом 3000 прикладів повідомлень, серед яких 2500 звичайних і 500 спам повідомлень (рис 3.4).

```
ham,"Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...",,,
ham,Ok lar... Joking wif u oni...,,,
spam,Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C'
ham,U dun say so early hor... U c already then say...,,,
ham,"Nah I don't think he goes to usf, he lives around here though",,,
spam,"FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs t
ham,Even my brother is not like to speak with me. They treat me like aids patent.,,,
ham,As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9
spam,WINNER!! As a valued network customer you have been selected to receivea £900 prize reward! To claim call 09061701461. Claim coc
spam,Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Upc
ham,"I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.",,,
spam,"SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply
spam,"URGENT! You have won a 1 week FREE membership in our £100,000 Prize Jackpot! Txt the word: CLAIM to No: 81010 T&C www.dbuk.net
ham,I've been searching for the right words to thank you for this breather. I promise i wont take your help for granted and will fulf
ham,I HAVE A DATE ON SUNDAY WITH WILL!!,,,
spam,"XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message or click here>> http://wap. xxxmobilemoviecl
ham,Oh k...i'm watching here:),,,
ham,Eh u remember how 2 spell his name... Yes i did. He v naughty make until i v wet.,,,
spam,"England v Macedonia - dont miss the goals/team news. Txt ur national team to 87077 eg ENGLAND to 87077 Try:WALES, SCOTLAND 4txt
ham,Is that seriously how you spell his name?,,,
ham,I see the letter B on my car,,,
ham,Anything lor... U decide...,,,
ham,Hello! How's you and how did saturday go? I was just texting to see if you'd decided to do anything tomo. Not that i'm trying to
ham,Pls go ahead with watts. I just wanted to be sure. Do have a great weekend. Abiola,,,
ham,"Did I forget to tell you ? I want you , I need you, I crave you ... But most of all ... I love you my sweet Arabian steed ... Mm
spam,07732584351 - Rodger Burns - MSG = We tried to call you re your reply to our sms for a free nokia mobile + free camcorder. Pleas
ham,WHO ARE YOU SEEING?,,,
ham,Great! I hope you like your man well endowed. I am  &lt;#&gt;  inches...,,,
ham,No calls..messages..missed calls,,,
ham,Didn't you get hep b immunisation in nigeria.,,,
ham,"Fair enough, anything going on?",,,
ham,"Yeah hopefully, if tyler can't do it I could maybe ask around a bit",,,
ham,U don't know how stubborn I am. I didn't even want to go to the hospital. I kept telling Mark I'm not a weak sucker. Hospitals ar
ham,What you thinked about me. First time you saw me in class.,,,
ham,"A gram usually runs like  &lt;#&gt; , a half eighth is smarter though and gets you almost a whole second gram for  &lt;#&gt;",,,
ham,K fyi x has a ride early tomorrow morning but he's crashing at our place tonight,,,
ham,"Wow. I never realized that you were so embarassed by your accomodations. I thought you liked it, since i was doing the best i cc
spam,SMS. ac Sptv: The New Jersey Devils and the Detroit Red Wings play Ice Hockey. Correct or Incorrect? End? Reply END SPTV,,,
ham,Do you know what Mallika Sherawat did yesterday? Find out now @  &lt;URL&gt;,,,
```

Рис. 3.1 Приклад використаних даних для навчання та тренування моделі

Дата-сет містить 87% звичайних прикладів електроних повідомлень і 13% повідомлень зі спамом (рис. 3.2):

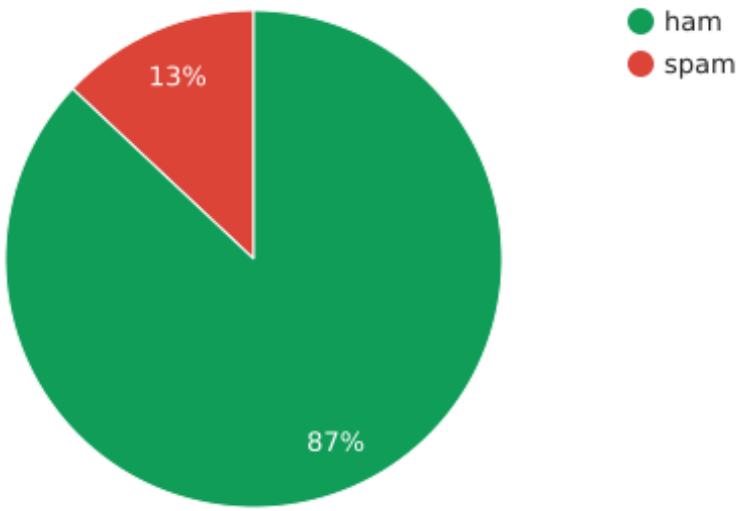


Рис. 3.2Відсоткова кількість спамових та звичайних повідомлень у датасеті “Email Spam Detection Dataset”

Основний файл із даними, що містить датасет, був розподілений на два окремих файли: перший містить 80% всієї інформації із датасету і використовується для тренування моделі, а другий містить решту 20% даних і

використовується для тестування моделі (рис 3.3). Цей розподіл даних забезпечує об'єктивну оцінку продуктивності моделі на нових, раніше не бачених даних.

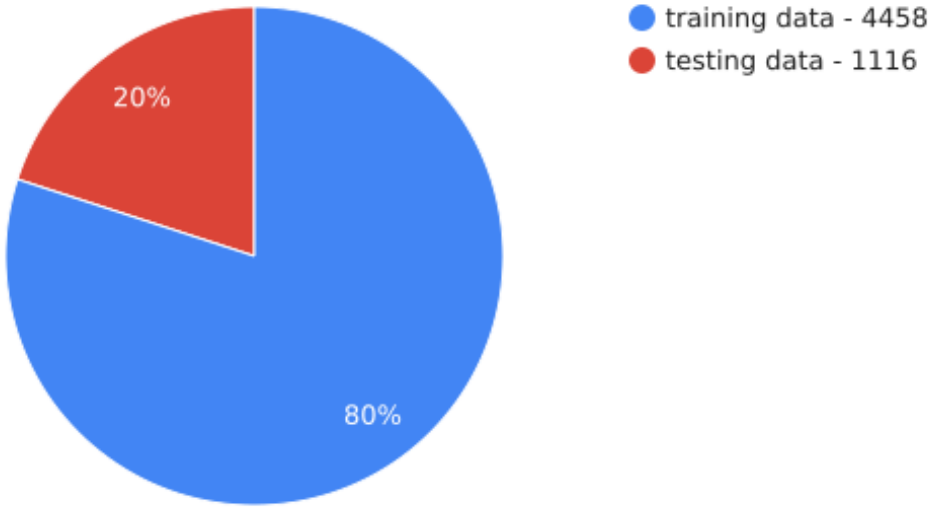


Рис 3.3 – Розподіл даних для використання з моделею.

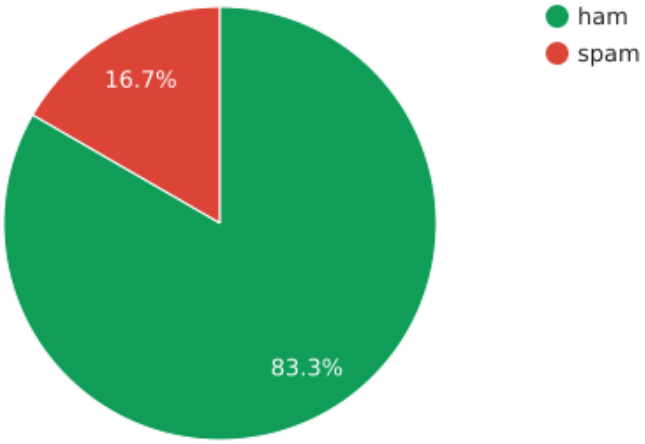


Рис. 3.4 Відсоткова кількість спамових та звичайних повідомлень у датасеті “Spam or Not Spam Dataset”

Найчастіші слова які трапляються у спамових (рис. 3.4) і звичайних (рис. 3.5) електронних листах:

Рис. 3.4 Найчастіші слова, що трапляються у спам-листах.

Рис. 3.5 Слова, що трапляються у звичайних електронних повідомленнях

Таким чином, процес підготовки даних для навчання моделі включає збір даних, їх попередню обробку, розділення на навчальні та тестові вибірки.Це дозволяє створити надійну систему фільтрації спаму на основі машинного навчання.

3.1.2 Тестування моделі та виявлення оцінки ефективності

Виявлення оцінки ефективності є ключовим етапом у процесі розробки та впровадження моделей машинного навчання для класифікації спаму. Тому

важливо розглянути методи тестування моделей машинного навчання, які були використані для оцінки їх ефективності у поставленій задачі. Для кожної моделі було проведено тести з використанням метрики точності та перевірки вводу тексту на вияв спаму.

Метрика точності є однією з найпоширеніших метрик для оцінки класифікаційних моделей. Вона визначається як відсоток правильно класифікованих зразків від загальної кількості зразків. Обчислюється за формулою (3.1):

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{TF}} \quad (3.1)$$

де, TP (True Positive) - кількість правильно класифікованих позитивних зразків;

TN (True Negative) - кількість правильно класифікованих негативних зразків;

FP (False Positive) - кількість хибних класифікованих позитивних зразків;

FN (False Negative) - кількість хибних класифікованих негативних зразків;

Точність є зручною метрикою для збалансованих даних, де кількість зразків кожного класу приблизно однакова.

3.2 Створення веб-ресурсу для використання моделі по виявленню спаму

Проєктування вебзастосунку є ключовим етапом розробки програмного забезпечення, оскільки воно визначає структуру, функціональність та взаємодію між різними компонентами системи.

Для реалізацію вебзастосунку, було обрано фреймворк Flask для мови програмування Python. Даний фреймворк забезпечує просту передачу HTTP-запитів між клієнтом та сервером. На рис. 3.6 представлено базову архітектуру

вебсерверу і клієнта, яка демонструє взаємодію між вебсервером і клієнтом за допомогою HTTP запитів та відповідей.

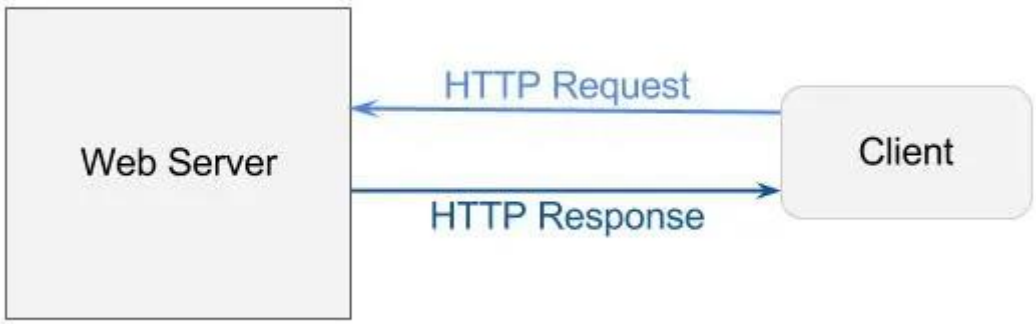


Рис. 3.6 Графічна архітектура вебсервера та клієнта

3.2.2 Впровадження моделі у вебресурс

У вебзастосунку основними двома методами є “train”(рис 3.7) та “predict” (рис. 3.8).

```python
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/train', methods=['POST'])
def train():
    method = request.form['method']
    dataset = request.form['dataset']
    app.logger.debug(f"Received training request: method={method}, dataset={dataset}")
    try:
        if dataset == 'dataset1':
            if method == 'polynomial':
                accuracy = polynomial_model1.train()
            elif method == 'bernoulli':
                accuracy = bernoulli_model1.train()
            elif method == 'knn':
                accuracy = knn_model1.train()
            elif method == 'svm':
                accuracy = svm_model1.train()
            else:
                app.logger.error(f"Invalid method: {method}")
                return jsonify({'error': 'Invalid method'}), 400
        elif dataset == 'dataset2':
            if method == 'polynomial':
                accuracy = polynomial_model2.train()
            elif method == 'bernoulli':
                accuracy = bernoulli_model2.train()
            elif method == 'knn':
                accuracy = knn_model2.train()
            elif method == 'svm':
                accuracy = svm_model2.train()
            else:
                app.logger.error(f"Invalid method: {method}")
                return jsonify({'error': 'Invalid method'}), 400
        else:
            app.logger.error(f"Invalid dataset: {dataset}")
            return jsonify({'error': 'Invalid dataset'}), 400
    except ValueError as e:
        app.logger.error(f"ValueError: {str(e)}")
        return jsonify({'error': str(e)}), 400
    return jsonify({'accuracy': accuracy})
```

Рис. 3.7 Метод у вебзастосунку для тренування моделі

Метод прив’язаний до URL ‘/train’ та одробляє POST – запит, відповідає за навчання моделі машинного навчання на основі обраних користувачем параметрів. Коли користувач відправляє запит на навчання, сервер отримує метод навчання та набір даних, які були обрані користувачем. Метод потім використовує ці параметри для вибору відповідної моделі та набору даних для навчання.

Залежно від обраного набору даних і методу, сервер викликає відповідну функцію навчання для поліноміальної моделі, моделі Бернуллі, K-ближніх сусідів або методу опорних векторів. Наприклад, якщо обраний набір даних dataset1 і метод polynomial, сервер використовує модель polynomial_model1 для навчання.

Після завершення навчання метод повертає точність навченої моделі у форматі JSON.

Якщо користувач обрав некоректний метод або набір даних, метод train повертає повідомлення про помилку, інформуючи про неправильний вибір. Також, метод обробляє можливі помилки, такі як ValueError, щоб забезпечити надійність роботи додатку. Таким чином, метод train забезпечує можливість ефективного навчання моделей машинного навчання на різних наборах даних, дозволяючи користувачам обирати найвідповідніші методи для їхніх потреб.

```python
@app.route('/predict', methods=['POST'])
def predict():
    method = request.form['method']
    dataset = request.form['dataset']
    text = request.form['text']
    app.logger.debug(f"Received prediction request: method={method}, dataset={dataset}, text={text}")
    try:
        if dataset == 'dataset1':
            if method == 'polynomial':
                result = polynomial_model1.predict(text)
            elif method == 'bernoulli':
                result = bernoulli_model1.predict(text)
            elif method == 'knn':
                result = knn_model1.predict(text)
            elif method == 'svm':
                result = svm_model1.predict(text)
            else:
                app.logger.error(f"Invalid method: {method}")
                return jsonify({'error': 'Invalid method'}), 400
        elif dataset == 'dataset2':
            if method == 'polynomial':
                result = polynomial_model2.predict(text)
            elif method == 'bernoulli':
                result = bernoulli_model2.predict(text)
            elif method == 'knn':
                result = knn_model2.predict(text)
            elif method == 'svm':
                result = svm_model2.predict(text)
            else:
                app.logger.error(f"Invalid method: {method}")
                return jsonify({'error': 'Invalid method'}), 400
        else:
            app.logger.error(f"Invalid dataset: {dataset}")
            return jsonify({'error': 'Invalid dataset'}), 400
    except ValueError as e:
        app.logger.error(f"ValueError: {str(e)}")
        return jsonify({'error': str(e)}), 400
    return jsonify({'result': result})
```

Рис 3.8 – Метод для передбачення результату

Метод "predict" призначений для обробки запитів на передбачення результату, заснованих на введеному тексті. Коли користувач відправляє запит на передбачення, сервер отримує обрані параметри методу, набір даних і текст, який

потрібно обробити. Потім метод predict використовує ці параметри, щоб вибрати відповідну модель машинного навчання і передбачити результат.

Таким чином, метод predict дозволяює користувачам вибирати різні моделі і набори даних для аналізу введеного тексту.

3.3 Експерименти та оцінка результату

У цьому розділі представлено експерименти, проведені для оцінки ефективності різних методів класифікації спаму. У своєму застосунку я реалізував методи Бернуллі та поліноміального наївного Байеса, а також інші популярні методами машинного навчання, такими як метод опорних векторів (SVM) та метод k-найближчих сусідів (k-NN). Це допоможе вирішити, який метод є найбільш ефективним у поставленій задачі по виявленню спаму.

Результати першої моделі наївного байесового класифікатора Бернуллі за першим датасетом (рис. 3.9) та другим датасетом (рис. 3.10).

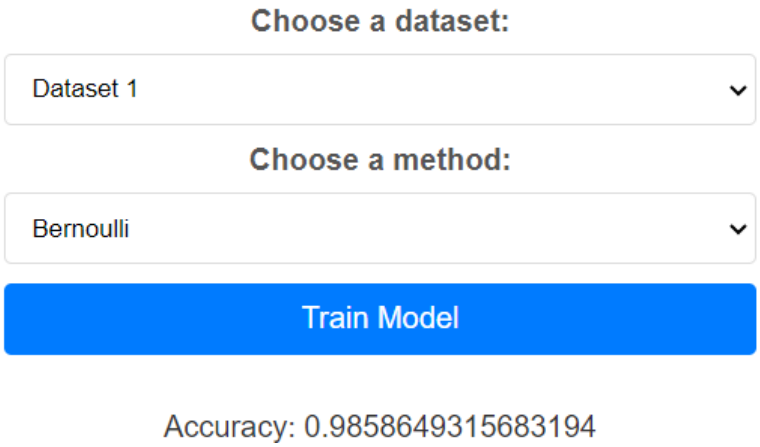


Рис. 3.9 Результат моделі наївного Байеса Бернуллі за першим датасетом

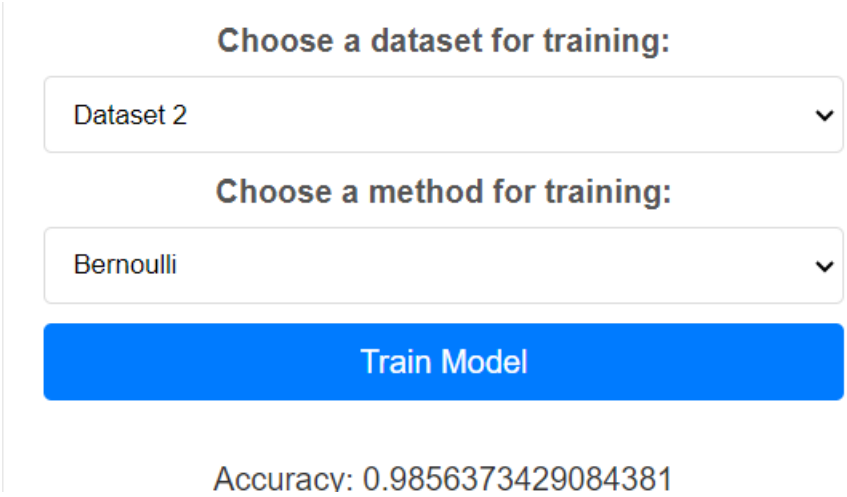


Рис. 3.10 Результат моделі наївного Байеса Бернуллі за другим датасетом

Метод Бернуллі показав доволі високу точність на першому та другому датасеті за результатами метрики точності. Далі порівняємо його із результатами поліноміально байесівського класифікатора:

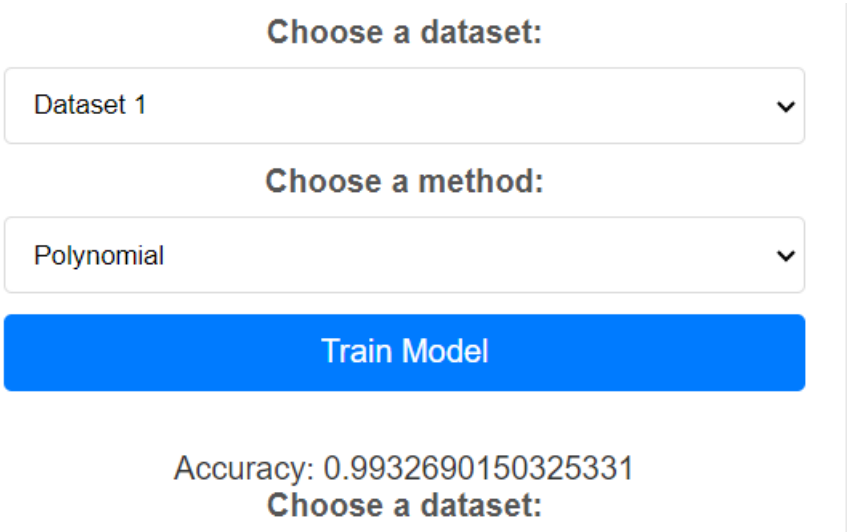


Рис. 3.11 Результат поліноміальної моделі наївного Байеса за першим датасетом

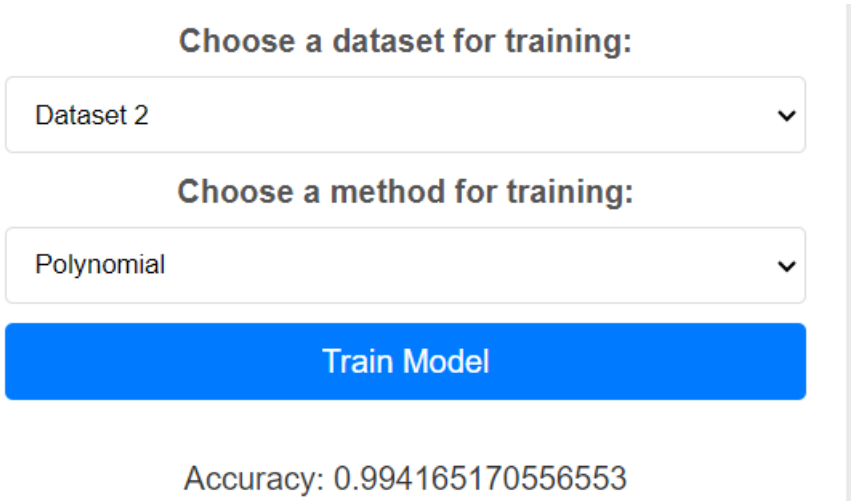


Рис. 3.12 Результат поліноміальної моделі наївного Байеса за другим датасетом

Поліноміальний метод наївного Байеса показав ще кращій показник. Ці результати свідчать про високу ефективність обох моделей, але поліноміальна модель наївного Байеса демонструє трохи кращу точність у порівнянні з моделлю Бернуллі на обох наборах даних.

Далі було проведено тестування інших методів машинного навчання, такі як к-наближених сусідів та метод опорних векторів

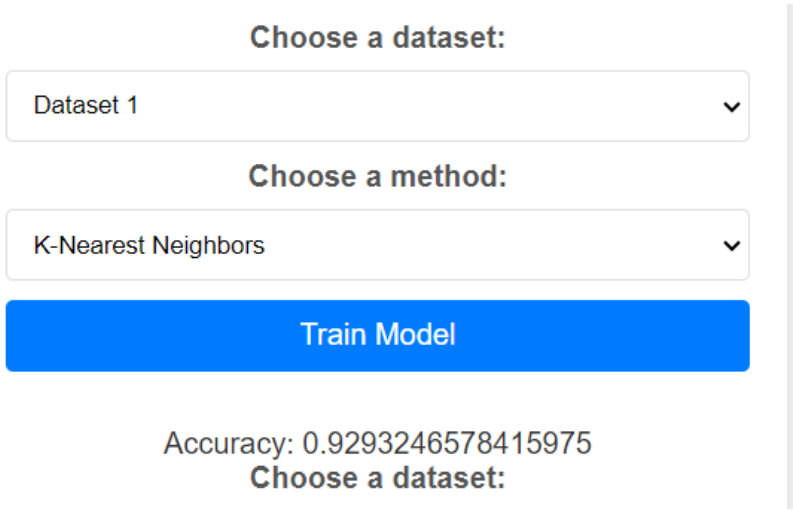


Рис. 3.11 Результат моделі K-ближніх сусідів за першим датасетом

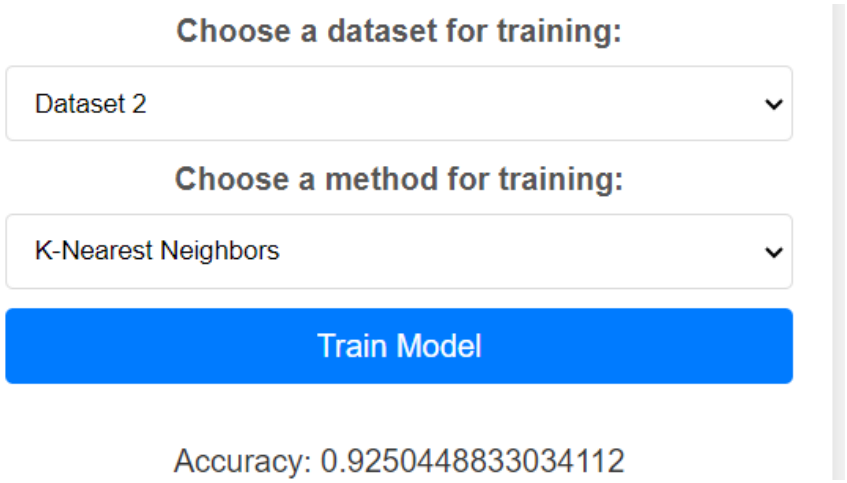


Рис. 3.11 Результат моделі K-ближніх сусідів за другим датасетом

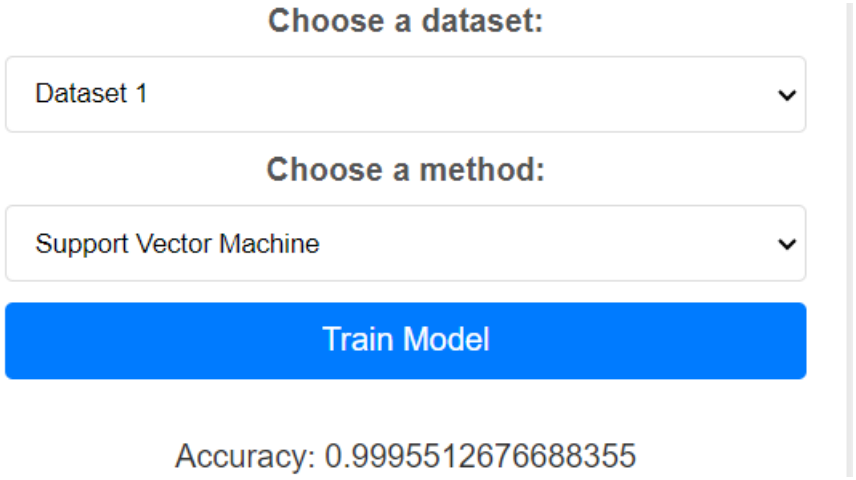


Рис. 3.12 Результат методу опорних векторів за першим датасетом.

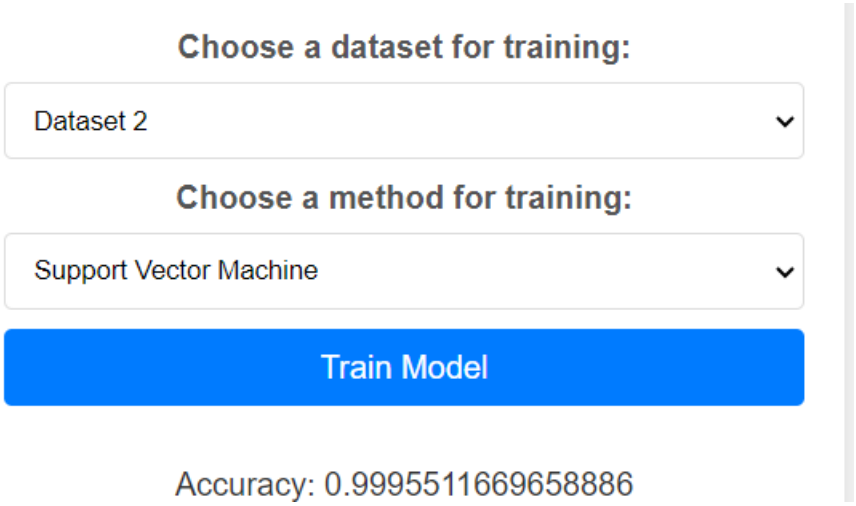


Рис. 3.13 Результат методу опорних векторів за першим датасетом

За результатами видно, що найкращий показник за двома датасетами є у методу опорних векторів. Найнижчий результат продемонстував метод к-наближених сусідів.

Далі збираюсь перевірити, як моделі визначають спамові повідомлення через текстове поле користувача. Для прикладу візьмемо перший датасет, так як він містить більше тернувальних даних. Спочатку повідомлення буде звичайним:

«Hi Maria, I wanted to remind you about our meeting tomorrow at 10:00 AM».

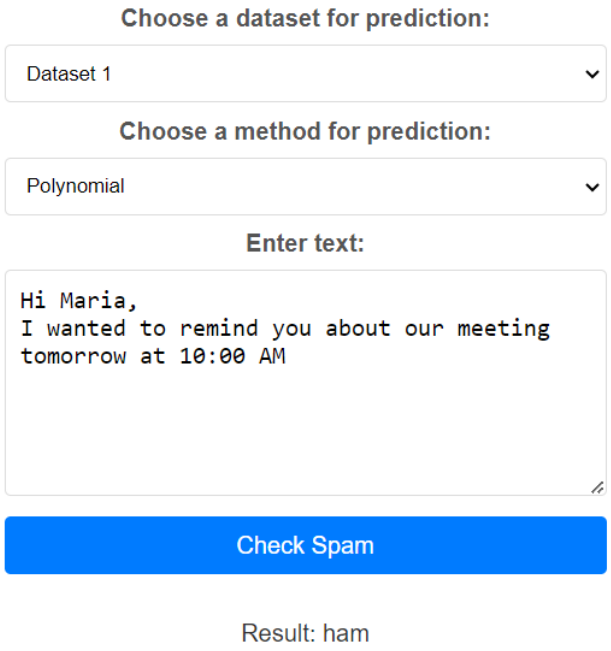


Рис. 3.14 Результат поліноміальної моделі на звичайне повідомлення

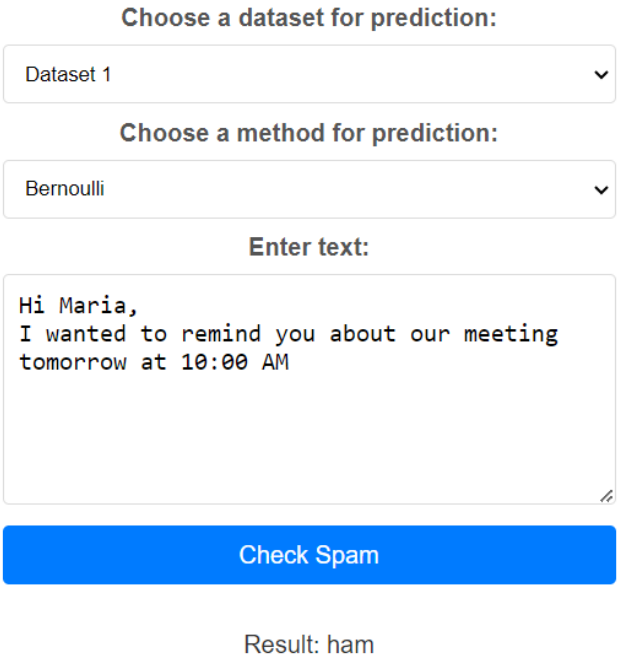


Рис 3.15 Результат наївного Байеса Бернуллі на звичайне повідомлення

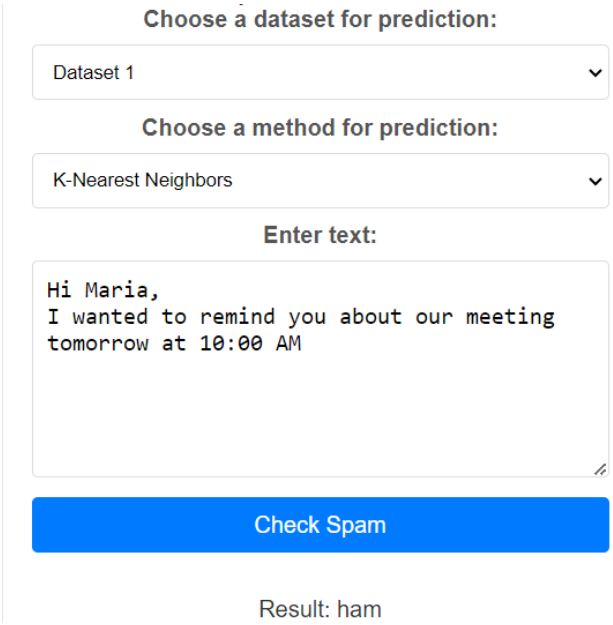


Рис 3.16 Результат K-ближніх сусідів на звичайне повідомлення

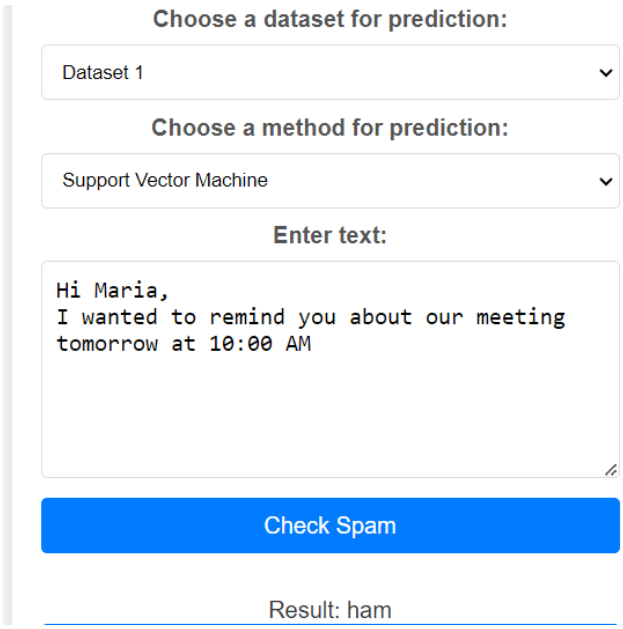


Рис. 3.17 Результат методу опорних векторів на звичайне повідомлення

Тепер перевіримо моделі, використовуючи спамове повідомлення: «Congratulations! You have won $1,000,000! To claim your prize, simply click the link below and fill out the form. Hurry, this offer is only valid until the end of the day!»

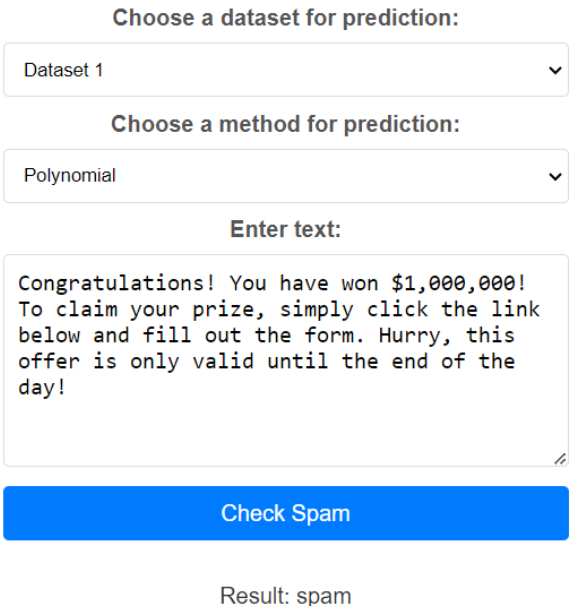


Рис. 3.18 Результат поліноміальної моделі на спамове повідомлення

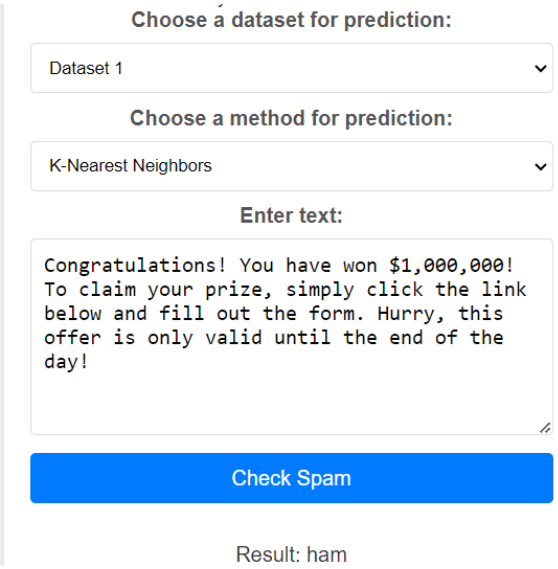


Рис. 3.19 Результат K-ближніх сусідів на спамове повідомлення

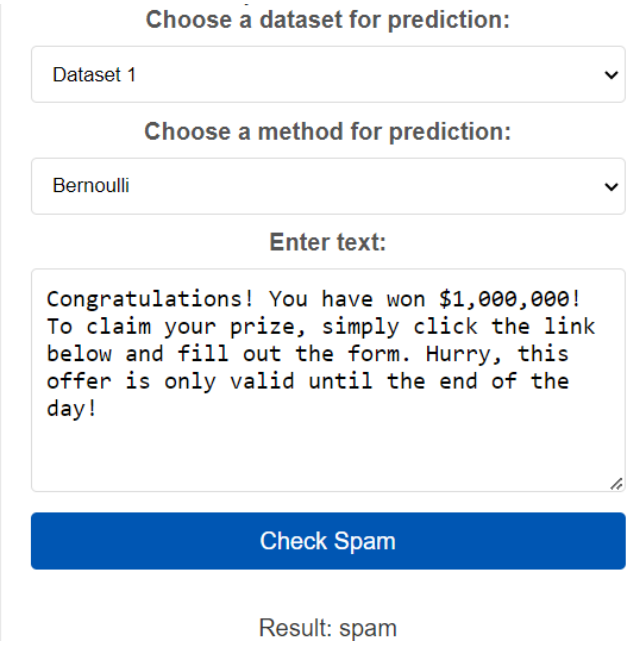


Рис. 3.20 Результат наївного Байеса Бернуллі на спамове повідомлення

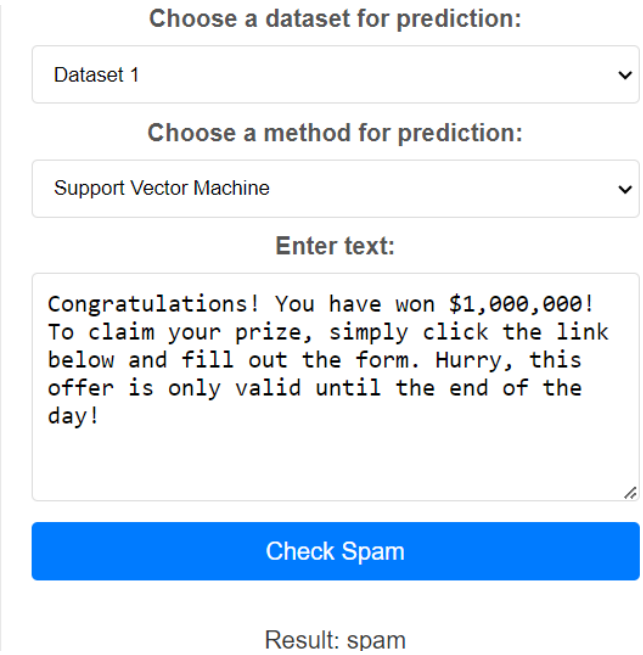


Рис. 3.21 Результат методу опорних векторів на спамове повідомлення

Проведені експерименти з введенням тексту для перевірки спам-фільтра показали цікаві результати. Всі моделі, за винятком методу k-наближених сусідів, успішно класифікували спамові та звичайні повідомлення.

Це може бути зумовлено кількома причинами. По-перше, метод k-NN чутливий до обсягу та якості навчальних даних. Якщо обсяг даних недостатній або

дані мають значну кількість шуму, ефективність цього методу може суттєво знижуватися. По-друге, метод k-NN може бути менш ефективним для завдань, де існує велика варіативність у текстових даних, оскільки він просто порівнює нові дані з вже наявними зразками, не враховуючи більш складні взаємозв'язки між ознаками. Нарешті, можливі проблеми з вибором оптимального значення параметра k, що також може вплинути на результати класифікації. Таким чином, для завдань, де точність виявлення спаму є критично важливою, слід віддавати перевагу іншим методам, які демонструють кращі результати.

3.4 Висновок розділу 3

У цьому розділі було розглянуто процес розробки вебресурсу для виявлення спаму, включаючи розробку та тестування моделей для класифікації спамових повідомлень. Основні етапи включали підготовку даних, реалізацію алгоритмів, тестування моделей та впровадження їх у вебресурс.

Порівнюючи різні моделі за метрикою точності, ми використовували два різних датасети. Результати показали високу ефективність більшості моделей, окрім методу k-наближених сусідів (k-NN). Найвищий результат продемонстрував метод опорних векторів (SVM).

Під час тестування моделей на прикладах спамових та звичайних повідомлень було виявлено, що всі моделі, окрім k-NN, успішно розпізнали спамове повідомлення. Метод k-NN не зміг правильно класифікувати спамове повідомлення і позначив його як "ham".

Таким чином, результати експериментів підтверджують високу ефективність моделей Поліноміального та Бернуллі наївного Байеса, а також методу опорних векторів для задачі виявлення спаму. Впровадження цих моделей у вебресурс дозволяє забезпечити надійне та швидке розпізнавання спамових повідомлень, що дозволить захистить продукт для бізнесу.

ВИСНОВКИ

В даній роботі було розглянуто проблему спаму в Інтернеті та розроблено вебресурс для його виявлення на основі методів машинного навчання. Перший розділ надав огляд існуючих методів виявлення спаму та їх значення для користувачів та бізнесу. Другий розділ детально розглянув алгоритм наївного Байеса, його застосування для класифікації тексту, важливі аспекти та недоліки. Також було розглянуто інші методи машинного навчання по розв'язання цієї проблеми, щоб порівняти їх із байесовим класифікатором Третій розділ описав процес розробки вебресурс для виявлення спаму, включаючи підготовку даних, реалізацію алгоритму, створення вебінтерфейс та експериментальне оцінювання результату.

Результати експериментів показали, що розроблений веб-ресурс здатний ефективно виявляти спам, що дозволяє підвищити якість користувачів та забезпечити захист від небажаних повідомлень. Також, робота виявила переваги та обмеження використання алгоритму наївного Байеса у контексті виявлення спаму.

Отримані результати та розроблений вебресурс можуть бути використані для інтеграції в продукти для бізнесу, допомагаючи уникнути спаму та підвищити якість користувачів. Крім того, ця робота може послужити основою для подальших досліджень у галузі виявлення спаму та застосування методів машинного навчання.

ПЕРЕЛІК ПОСИЛАНЬ

1. What is Spam? A Brief History of Unwanted Email - MagicSpam Business Email Security Blog. *MagicSpam Business Email Security Blog*. URL: https://www.magicspam.com/blog/what-is-spam-a-brief-history-of-unwanted-email/#:~:text=The%20history%20of%20spam%20emails,sent%20by%20Digital%20Equipment%20Corp. (date of access: 15.04.2024).

2. Global average daily spam volume 2021 | Statista. *Statista*. URL: https://www.statista.com/statistics/1270424/daily-spam-volume-global/ (date of access: 15.04.2024).

3. 5 common types of SPAM & how you can protect yourself against them. *SEO Blog | cognitiveSEO Blog on SEO Tactics & Strategies*. URL: https://cognitiveseo.com/blog/18718/5-common-types-spam-can-protect/#3 (date of access: 22.04.2024).

4. Spam filtering: why it's important and how it works. *Five Nines Blog*. URL: https://blog.fivenines.com/spam-filtering-why-its-important-and-how-it-works (date of access: 26.04.2024).

5. Heuristic analysis technology applied to anti-spam filtering solutions. *Altospam*. URL: https://www.altospam.com/en/glossary/heuristic-analysis/ (date of access: 25.04.2024).

6. Domain Message Authentication Reporting & Conformance. *dmarc.org – Domain Message Authentication Reporting & Conformance*. URL: https://dmarc.org/ (date of access: 19.04.2024).

7. Ensemble methods. *Corporate Finance Institute*. URL: https://corporatefinanceinstitute.com/resources/data-science/ensemble-methods/#:~:text=Ensemble%20methods%20are%20techniques%20that,ensemble%20methods%20in%20machine%20learning (date of access: 30.04.2024).

8. DomainKeys Identified Mail (DKIM). *DomainKeys Identified Mail (DKIM)*. URL: https://www.dkim.org/ (date of access: 17.04.2024).

9. Zhang H. The Optimality of Naive Bayes. *Faculty of Computer Science | UNB*. URL: https://www.cs.unb.ca/~hzhang/publications/FLAIRS04ZhangH.pdf (date of access: 17.04.2024).

10. SpamDL: A High Performance Deep Learning Spam Detector Using Stanford Global Vectors and Bidirectional Long Short-Term Memory Neural Networks / R. Ghayoula et al. *https://www.researchgate.net/*. URL: https://www.researchgate.net/publication/363589502_SpamDL_A_High_Performance_Deep_Learning_Spam_Detector_Using_Stanford_Global_Vectors_and_Bidirectional_Long_Short-Term_Memory_Neural_Networks (date of access: 26.04.2024).

11. Autospam and Naive Bayes: The Grandfather of Spam Filters Still Making Waves - Pixelfed blog. *Pixelfed Blog*. URL: https://pixelfed.blog/p/2023/feature/autospam-and-naive-bayes-the-grandfather-of-spam-filters-still-making-waves (date of access: 17.04.2024).

12. Bayes theorem and Bayesian inference. *Department of Computing | Faculty of Engineering | Imperial College London*. URL: https://www.doc.ic.ac.uk/~dfg/ProbabilisticInference/IDAPILecture01.pdf (date of access: 29.03.2024).

13. What is the k-nearest neighbors algorithm? | IBM. *IBM in Deutschland, Österreich und der Schweiz*. URL: https://www.ibm.com/topics/knn#:~:text=What%20is%20the%20KNN%20algorithm,of%20an%20individual%20data%20point (date of access: 23.04.2024).

14. Torabi Z. S., Nabiollahi A., Nadimi-Shahraki M. H. Efficient Support Vector Machines for Spam Detection: A Survey. *https://www.researchgate.net/*. URL: https://www.researchgate.net/publication/316075352_Efficient_Support_Vector_Machines_for_Spam_Detection_A_Survey (date of access: 25.04.2024).

15. 1.4. Support Vector Machines. *scikit-learn*. URL: https://scikit-learn.org/stable/modules/svm.html (date of access: 24.04.2024).

16. Email Spam Detection Dataset (classification). *Kaggle: Your Machine Learning and Data Science Community*.

URL: https://www.kaggle.com/datasets/shantanudhakadd/email-spam-detection-dataset-classification (date of access: 06.05.2024).

17. Spam or Not Spam Dataset. *Kaggle: Your Machine Learning and Data Science Community*. URL: https://www.kaggle.com/datasets/ozlerhakan/spam-or-not-spam-dataset (date of access: 02.05.2024).

18. Markus, A. (2018). "Spam Filtering Techniques: Comparative Analysis." Journal of Machine Learning Research. URL: https://jmlr.org/papers/volume19/markus18a/markus18a.pdf (date of access: 22.04.2024).

19. Кисіль Т.М., Остренський Н.П., ІТ-ІННОВАЦІЇ В ПРОДУКТАХ HEWLETT PACKARD ENTERPRISE ВІДЕО / Науково-практична конференція «Сучасні досягнення компанії HEWLETT PACKARD ENTERPRISE в галузі ІТ та нові можливості їх вивчення і застосування». Збірник тез. – К.: ДУІКТ, 2023р., с. 16-18

20. Кисіль Т.М., Остренський Н.П., Кубар Н.М., СТРАТЕГІЇ КІБЕРСТІЙКОСТІ: УПРАВЛІННЯ РИЗИКАМИ ТА БЕЗПЕРЕРВНІСТЬ БІЗНЕСУ / V Науково-технічна конференція «Сучасний стан та перспективи розвитку IoT» Збірник тез. – К.: ДУІКТ, 2024р., с. 51-53

ДОДАТОК А. ПРОГРАМНИЙ КОД ПРОЄКТОВАНОЇ СИСТЕМИ

Програмний код проєктованої системи

Реалізація моделі Бернуллі

```
class BernoulliModel:
    def __init__(self, train_data, test_data):
        self.train_data = train_data
        self.test_data = test_data
        self.vectorizer = CountVectorizer(binary=True)
        self.model = BernoulliNB()

    def train(self):
        X_train =
self.vectorizer.fit_transform(self.train_data['text'])
        y_train = self.train_data['label']
        if X_train.shape[1] == 0:
            raise ValueError("Empty vocabulary; perhaps the
documents only contain stop words")
        self.model.fit(X_train, y_train)
        accuracy = accuracy_score(y_train,
self.model.predict(X_train))
        return accuracy

    def predict(self, text):
        X_test = self.vectorizer.transform([text])
        prediction = self.model.predict(X_test)
        return 'spam' if prediction[0] == 1 else 'ham'

    def evaluate(self):
        X_test =
self.vectorizer.transform(self.test_data['text'])
        y_test = self.test_data['label']
        y_pred = self.model.predict(X_test)
        cm = confusion_matrix(y_test, y_pred)
        return cm

    def plot_confusion_matrix(self):
```

```
        cm = self.evaluate()
        plt.figure(figsize=(8, 6))
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=['Ham', 'Spam'], yticklabels=['Ham', 'Spam'])
        plt.xlabel('Predicted')
        plt.ylabel('Actual')
        plt.title('Confusion Matrix - Bernoulli Model')
        plt.show()
```

Реалізація поліноміальної моделі

```
class PolynomialModel:
    def __init__(self, train_data, test_data):
        self.train_data = train_data
        self.test_data = test_data
        self.vectorizer = CountVectorizer()
        self.model = MultinomialNB()

    def train(self):
        X_train =
self.vectorizer.fit_transform(self.train_data['text'])
        y_train = self.train_data['label']
        if X_train.shape[1] == 0:
            raise ValueError("Empty vocabulary; perhaps the
documents only contain stop words")
        self.model.fit(X_train, y_train)
        accuracy = accuracy_score(y_train,
self.model.predict(X_train))
        return accuracy

    def predict(self, text):
        X_test = self.vectorizer.transform([text])
        prediction = self.model.predict(X_test)
        return 'spam' if prediction[0] == 1 else 'ham'

    def evaluate(self):
```

```
        X_test =
self.vectorizer.transform(self.test_data['text'])
        y_test = self.test_data['label']
        y_pred = self.model.predict(X_test)
        cm = confusion_matrix(y_test, y_pred)
        return cm

    def plot_confusion_matrix(self):
        cm = self.evaluate()
        plt.figure(figsize=(8, 6))
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=['Ham', 'Spam'], yticklabels=['Ham', 'Spam'])
        plt.xlabel('Predicted')
        plt.ylabel('Actual')
        plt.title('Confusion Matrix - Polynomial Model')
        plt.show()
```

Реалізація моделі к-наближених сусідів

```
class KNNModel:
    def __init__(self, train_data, test_data, n_neighbors=5):
        self.train_data = train_data
        self.test_data = test_data
        self.vectorizer = CountVectorizer()
        self.model =
KNeighborsClassifier(n_neighbors=n_neighbors)

    def train(self):
        X_train =
self.vectorizer.fit_transform(self.train_data['text'])
        y_train = self.train_data['label']
        if X_train.shape[1] == 0:
            raise ValueError("Empty vocabulary; perhaps the
documents only contain stop words")
        self.model.fit(X_train, y_train)
        accuracy = accuracy_score(y_train,
self.model.predict(X_train))
```

```
        return accuracy

    def predict(self, text):
        X_test = self.vectorizer.transform([text])
        prediction = self.model.predict(X_test)
        return 'spam' if prediction[0] == 1 else 'ham'

    def evaluate(self):
        X_test =
self.vectorizer.transform(self.test_data['text'])
        y_test = self.test_data['label']
        y_pred = self.model.predict(X_test)
        cm = confusion_matrix(y_test, y_pred)
        return cm

    def plot_confusion_matrix(self):
        cm = self.evaluate()
        plt.figure(figsize=(8, 6))
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=['Ham', 'Spam'], yticklabels=['Ham', 'Spam'])
        plt.xlabel('Predicted')
        plt.ylabel('Actual')
        plt.title('Confusion Matrix - KNN Model')
        plt.show()
```

Реалізація методу опорних векторів

```
class SVMModel:
    def __init__(self, train_data, test_data):
        self.train_data = train_data
        self.test_data = test_data
        self.vectorizer = CountVectorizer()
        self.model = SVC(kernel='linear', probability=True)

    def train(self):
```

```
        X_train =
self.vectorizer.fit_transform(self.train_data['text'])
        y_train = self.train_data['label']
        if X_train.shape[1] == 0:
            raise ValueError("Empty vocabulary; perhaps the
documents only contain stop words")
        self.model.fit(X_train, y_train)
        accuracy = accuracy_score(y_train,
self.model.predict(X_train))
        return accuracy

    def predict(self, text):
        X_test = self.vectorizer.transform([text])
        prediction = self.model.predict(X_test)
        return 'spam' if prediction[0] == 1 else 'ham'

    def evaluate(self):
        X_test =
self.vectorizer.transform(self.test_data['text'])
        y_test = self.test_data['label']
        y_pred = self.model.predict(X_test)
        cm = confusion_matrix(y_test, y_pred)
        return cm

    def plot_confusion_matrix(self):
        cm = self.evaluate()
        plt.figure(figsize=(8, 6))
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=['Ham', 'Spam'], yticklabels=['Ham', 'Spam'])
        plt.xlabel('Predicted')
        plt.ylabel('Actual')
        plt.title('Confusion Matrix - SVM Model')
        plt.show()
```

Метод розподілення датасетів

```
def split_dataset(input_file, train_file, test_file,
test_size=0.2):
    data = pd.read_csv(input_file, names=['label', 'text'],
header=None)
    train_data, test_data = train_test_split(data,
test_size=test_size, random_state=42)
    train_data.to_csv(train_file, index=False)
    test_data.to_csv(test_file, index=False)

if __name__ == '__main__':
    input_file_1 = 'data/dataset1.csv'
    train_file_1 = 'data/train_data1.csv'
    test_file_1 = 'data/test_data1.csv'
    split_dataset(input_file_1, train_file_1, test_file_1)

    input_file_2 = 'data/dataset2.csv'
    train_file_2 = 'data/train_data2.csv'
    test_file_2 = 'data/test_data2.csv'
    split_dataset(input_file_2, train_file_2, test_file_2)
```

Клас із реалізації контролерів

```
app = Flask(__name__)

logging.basicConfig(level=logging.DEBUG)

train_data1, test_data1 = load_data('data/train_data1.csv',
'data/test_data1.csv')
train_data2, test_data2 = load_data('data/train_data2.csv',
'data/test_data2.csv')

polynomial_model1 = PolynomialModel(train_data1, test_data1)
bernoulli_model1 = BernoulliModel(train_data1, test_data1)
knn_model1 = KNNModel(train_data1, test_data1)
svm_model1 = SVMModel(train_data1, test_data1)

polynomial_model2 = PolynomialModel(train_data2, test_data2)
bernoulli_model2 = BernoulliModel(train_data2, test_data2)
knn_model2 = KNNModel(train_data2, test_data2)
svm_model2 = SVMModel(train_data2, test_data2)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/train', methods=['POST'])
def train():
    method = request.form['method']
    dataset = request.form['dataset']
    app.logger.debug(f"Received training request:
method={method}, dataset={dataset}")
    try:
        if dataset == 'dataset1':
            if method == 'polynomial':
                accuracy = polynomial_model1.train()
            elif method == 'bernoulli':
                accuracy = bernoulli_model1.train()
            elif method == 'knn':
```

```
                accuracy = knn_model1.train()
            elif method == 'svm':
                accuracy = svm_model1.train()
            else:
                app.logger.error(f"Invalid method: {method}")
                return jsonify({'error': 'Invalid method'}), 400
        elif dataset == 'dataset2':
            if method == 'polynomial':
                accuracy = polynomial_model2.train()
            elif method == 'bernoulli':
                accuracy = bernoulli_model2.train()
            elif method == 'knn':
                accuracy = knn_model2.train()
            elif method == 'svm':
                accuracy = svm_model2.train()
            else:
                app.logger.error(f"Invalid method: {method}")
                return jsonify({'error': 'Invalid method'}), 400
        else:
            app.logger.error(f"Invalid dataset: {dataset}")
            return jsonify({'error': 'Invalid dataset'}), 400
    except ValueError as e:
        app.logger.error(f"ValueError: {str(e)}")
        return jsonify({'error': str(e)}), 400
    return jsonify({'accuracy': accuracy})

@app.route('/predict', methods=['POST'])
def predict():
    method = request.form['method']
    dataset = request.form['dataset']
    text = request.form['text']
    app.logger.debug(f"Received prediction request:
method={method}, dataset={dataset}, text={text}")
    try:
        if dataset == 'dataset1':
            if method == 'polynomial':
                result = polynomial_model1.predict(text)
            elif method == 'bernoulli':
```

```
                result = bernoulli_model1.predict(text)
            elif method == 'knn':
                result = knn_model1.predict(text)
            elif method == 'svm':
                result = svm_model1.predict(text)
            else:
                app.logger.error(f"Invalid method: {method}")
                return jsonify({'error': 'Invalid method'}), 400
        elif dataset == 'dataset2':
            if method == 'polynomial':
                result = polynomial_model2.predict(text)
            elif method == 'bernoulli':
                result = bernoulli_model2.predict(text)
            elif method == 'knn':
                result = knn_model2.predict(text)
            elif method == 'svm':
                result = svm_model2.predict(text)
            else:
                app.logger.error(f"Invalid method: {method}")
                return jsonify({'error': 'Invalid method'}), 400
        else:
            app.logger.error(f"Invalid dataset: {dataset}")
            return jsonify({'error': 'Invalid dataset'}), 400
    except ValueError as e:
        app.logger.error(f"ValueError: {str(e)}")
        return jsonify({'error': str(e)}), 400
    return jsonify({'result': result})

@app.route('/confusion_matrix', methods=['POST'])
def confusion_matrix():
    method = request.form['method']
    dataset = request.form['dataset']
    app.logger.debug(f"Received confusion matrix request:
method={method}, dataset={dataset}")
    try:
        if dataset == 'dataset1':
            if method == 'polynomial':
                polynomial_model1.plot_confusion_matrix()
```

```
            elif method == 'bernoulli':
                bernoulli_model1.plot_confusion_matrix()
            elif method == 'knn':
                knn_model1.plot_confusion_matrix()
            elif method == 'svm':
                svm_model1.plot_confusion_matrix()
            else:
                app.logger.error(f"Invalid method: {method}")
                return jsonify({'error': 'Invalid method'}), 400
        elif dataset == 'dataset2':
            if method == 'polynomial':
                polynomial_model2.plot_confusion_matrix()
            elif method == 'bernoulli':
                bernoulli_model2.plot_confusion_matrix()
            elif method == 'knn':
                knn_model2.plot_confusion_matrix()
            elif method == 'svm':
                svm_model2.plot_confusion_matrix()
            else:
                app.logger.error(f"Invalid method: {method}")
                return jsonify({'error': 'Invalid method'}), 400
        else:
            app.logger.error(f"Invalid dataset: {dataset}")
            return jsonify({'error': 'Invalid dataset'}), 400
    except ValueError as e:
        app.logger.error(f"ValueError: {str(e)}")
        return jsonify({'error': str(e)}), 400
    return jsonify({'status': 'success'})

if __name__ == '__main__':
    app.run(debug=True)
```

Реалізація візуальної частини

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Spam Detection App</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    <style>
        textarea {
            width: 100%;
            height: 150px;
            padding: 10px;
            font-size: 16px;
            border: 1px solid #ddd;
            border-radius: 4px;
            box-sizing: border-box;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Spam Detection App</h1>
        <form id="train-form">
            <label for="train-dataset">Choose a dataset for training:</label>
            <select id="train-dataset" name="dataset">
                <option value="dataset1">Dataset 1</option>
                <option value="dataset2">Dataset 2</option>
            </select>
            <label for="train-method">Choose a method for training:</label>
            <select id="train-method" name="method">
                <option value="polynomial">Polynomial</option>
                <option value="bernoulli">Bernoulli</option>
                <option value="knn">K-Nearest Neighbors</option>
                <option value="svm">Support Vector Machine</option>
            </select>
            <button type="submit">Train Model</button>
        </form>
        <div id="accuracy"></div>
        <form id="predict-form">
            <label for="predict-dataset">Choose a dataset for prediction:</label>
            <select id="predict-dataset" name="dataset">
                <option value="dataset1">Dataset 1</option>
                <option value="dataset2">Dataset 2</option>
            </select>
            <label for="predict-method">Choose a method for prediction:</label>
            <select id="predict-method" name="method">
```

```
                <option value="polynomial">Polynomial</option>
                <option value="bernoulli">Bernoulli</option>
                <option value="knn">K-Nearest Neighbors</option>
                <option value="svm">Support Vector Machine</option>
            </select>
            <label for="text">Enter text:</label>
            <textarea id="text" name="text"></textarea>
            <button type="submit">Check Spam</button>
        </form>
        <div id="result"></div>
        <button id="confusion-matrix-btn">Show Confusion Matrix</button>
    </div>
    <script>
        document.getElementById('train-form').addEventListener('submit',
function(e) {
            e.preventDefault();
            const dataset = document.getElementById('train-dataset').value;
            const method = document.getElementById('train-method').value;
            fetch('/train', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/x-www-form-urlencoded',
                },
                body: `dataset=${dataset}&method=${method}`
            })
            .then(response => response.json())
            .then(data => {
                if (data.error) {
                    alert(`Error: ${data.error}`);
                } else {
                    document.getElementById('accuracy').innerText = `Accuracy:
${data.accuracy}`;
                }
            });
        });

        document.getElementById('predict-form').addEventListener('submit',
function(e) {
            e.preventDefault();
            const dataset = document.getElementById('predict-dataset').value;
            const method = document.getElementById('predict-method').value;
            const text = document.getElementById('text').value;
            fetch('/predict', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/x-www-form-urlencoded',
                },
                body: `dataset=${dataset}&method=${method}&text=${text}`
```

```
            })
            .then(response => response.json())
            .then(data => {
                if (data.error) {
                    alert(`Error: ${data.error}`);
                } else {
                    document.getElementById('result').innerText = `Result:
${data.result}`;
                }
            });
        });
document.getElementById('confusion-matrix-btn').addEventListener('click',
function() {
            const dataset = document.getElementById('predict-dataset').value;
            const method = document.getElementById('predict-method').value;
            fetch('/confusion_matrix', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/x-www-form-urlencoded',
                },
                body: `dataset=${dataset}&method=${method}`
            })
            .then(response => response.json())
            .then(data => {
                if (data.error) {
                    alert(`Error: ${data.error}`);
                } else {
                    alert('Confusion Matrix displayed in a new window.');
                }
            });
        });
    </script>
</body>
</html>
```

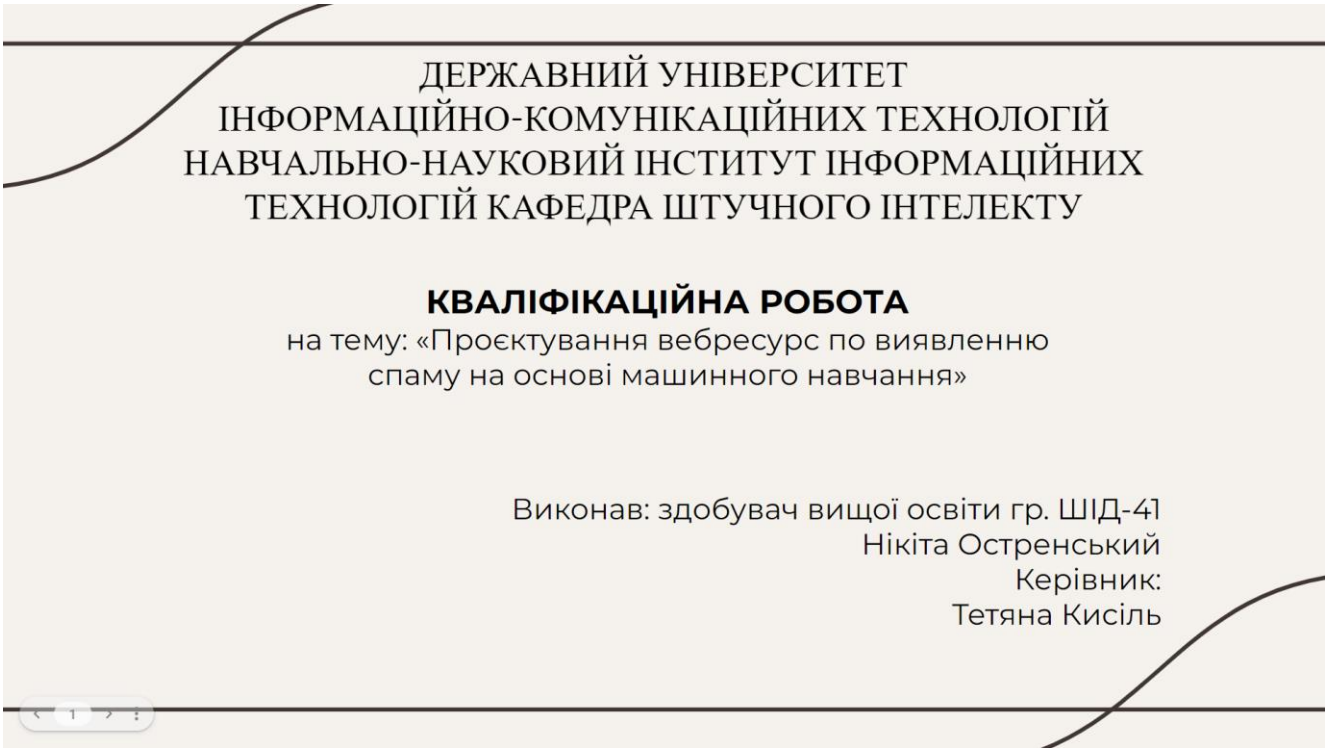
Стилі CSS

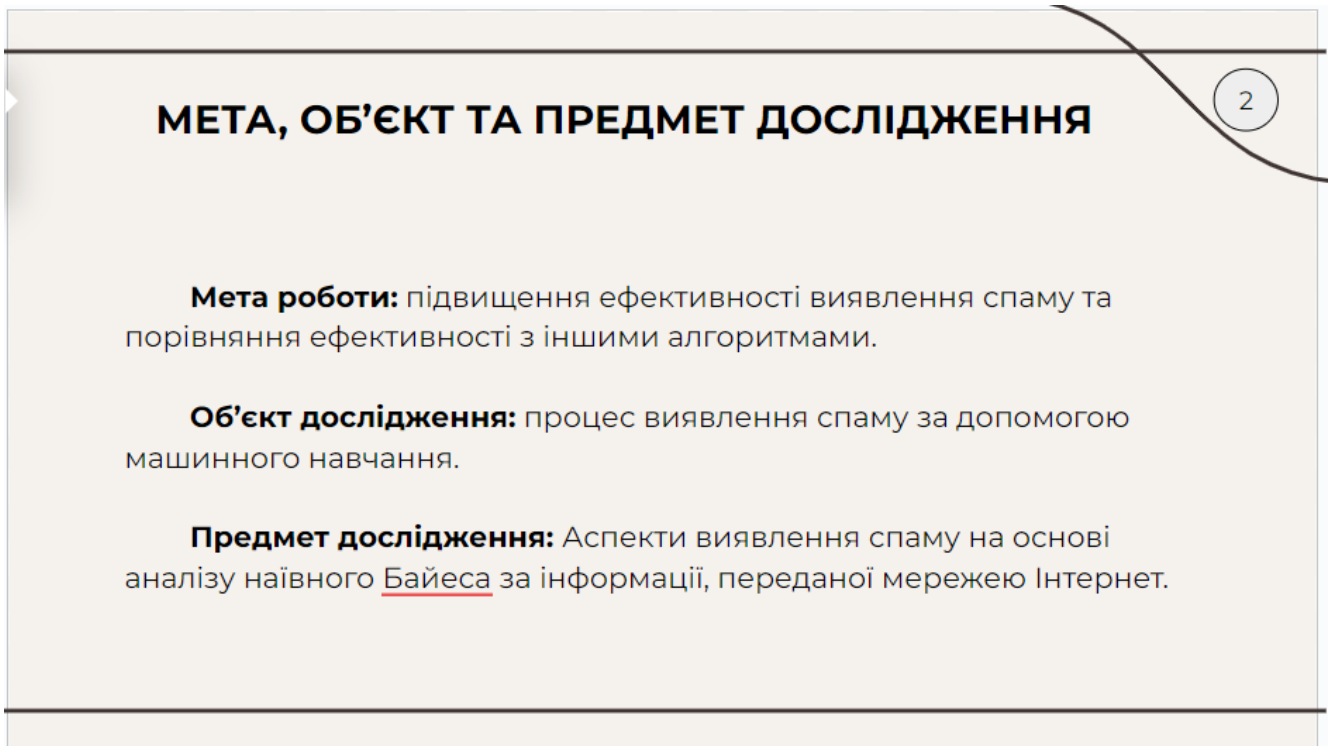
```
body {
    font-family: 'Arial', sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}
```

```
.container {
    background: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    width: 400px;
    text-align: center;
}

h1 {
    font-size: 24px;
    margin-bottom: 20px;
    color: #333;
}

form {
    margin-bottom: 20px;
}

label {
    display: block;
    margin-bottom: 8px;
    font-weight: bold;
    color: #555;
}

select, input[type="text"], textarea, button {
    width: 100%;
    padding: 10px;
    margin-bottom: 10px;
    border: 1px solid #ddd;
    border-radius: 4px;
    box-sizing: border-box;
}

button {
    background-color: #007BFF;
    color: white;
    border: none;
    cursor: pointer;
    font-size: 16px;
}

button:hover {
    background-color: #0056b3;
}

#accuracy, #result {
```

```
    font-size: 16px;
    color: #333;
    margin-top: 10px;
}
```

ДЕМОСТРАЦІЙНІ МАТЕРІАЛИ





3

ПОСТАНОВКА ЗАДАЧІ

1. Створення вебінтерфейсу додатка з можливістю введення тексту користувачем;
2. Розробка методів машинного навчання для аналізу повідомлення користувача з метою виявлення спаму;
3. Інтеграція алгоритму з вебінтерфейсом додатка;
4. Тестування роботи алгоритму;

4

ОПИС ВИКОРИСТАНИХ МЕТОДІВ

Баєсів класифікатор Бернуллі

Баєсів класифікатор Бернуллі є потужним інструментом для виявлення спаму в текстових повідомленнях. Він аналізує кожне слово у повідомленні та визначає, чи є воно спамом на основі його присутності. Цей метод особливо ефективний у випадках, коли важлива лише наявність або відсутність певних ключових слів.

5

ОПИС ВИКОРИСТАНИХ МЕТОДІВ

Поліноміальний наївний Баєсів класифікатор

Цей метод використовується для аналізу текстових даних, зокрема для виявлення спаму в електронних повідомленнях. Він оцінює ймовірність того, що слова зустрічаються у спамових повідомленнях, враховуючи їхні частоти зустрічі. Цей підхід дозволяє класифікувати повідомлення на основі кількості входжень кожного слова та апріорної ймовірності класів.

6

ОПИС ВИКОРИСТАНИХ МЕТОДІВ

Метод опорних векторів

Метод опорних векторів (SVM) є потужним класифікаційним алгоритмом, який працює шляхом побудови гіперплощини або набору гіперплощин у високовимірному просторі, щоб максимально розділити дані на різні класи. SVM ефективно справляється з багатовимірними даними і може застосовуватися для лінійно та нелінійно відокремлюваних задач.

7

ОПИС ВИКОРИСТАНИХ МЕТОДІВ

Метод k-наближених сусідів

Метод k-наближених сусідів (k-NN) є простим і інтуїтивно зрозумілим алгоритмом класифікації, який визначає клас зразка на основі класів його найближчих сусідів у тренувальному наборі даних. При цьому використовується відстань між зразками для визначення найближчих сусідів. Цей метод особливо підходить для задач, де подібні зразки належать до одного класу.

8

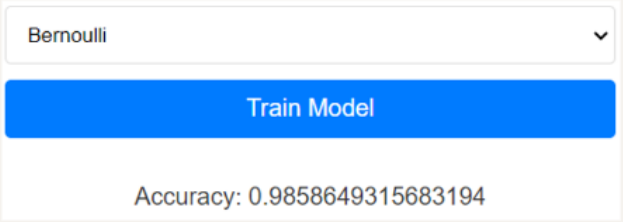
ПІДБІР НАБОРІВ ДАНИХ

Для навчання моделей машинного навчання було обрано дата-сети електронних повідомлень англійською мовою на близько 5500 та 3000 листів.

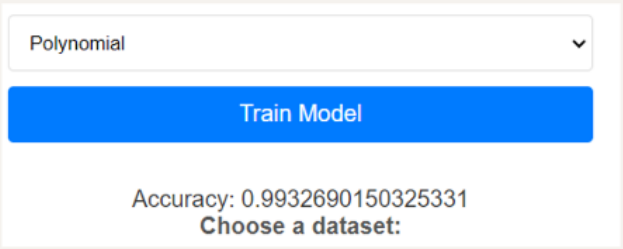
9

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Метод Бернуллі



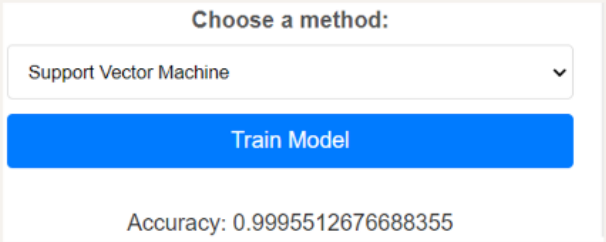
Поліноміальний метод



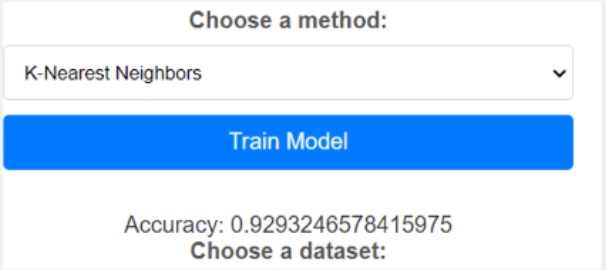
10

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Метод опорних векторів



Метод k-наближених сусідів



11

ВИСНОВОК

У кваліфікаційній роботі було проведено аналіз можливих методів боротьби зі спамом, зокрема електронних повідомлень.

Реалізовано вебзастосунок для виявлення спаму з використанням методів машинного навчання, таких як класифікатор наївного Байєса Бернуллі та поліноміальний класифікатор наївного Байєса. Проведено порівняння Байєсівського алгоритму з методами k-наближених сусідів та методом опорних векторів.

12

Дякую за увагу!

Ч