**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Західноукраїнський національний університет**
**Факультет комп'ютерних інформаційних технологій**
Кафедра кібербезпеки

**ПРОКОПЕНКО Денис Петрович**

# Алгоритми виявлення спаму на основі машинного навчання / Spam detection algorithms based on machine learning

спеціальність: 125 – Кібербезпека та захист інформації

освітньо-професійна програма – Кібербезпека

Кваліфікаційна робота

Виконав студент групи
КБзм -21

Д.П.Прокопенко

Науковий керівник
д.т.н., професор В.В.Яцків

Кваліфікаційну роботу
допущено до захисту:

«____» _____ 2024 р.

Завідувач кафедри
_____ **В.В.Яцків**

**ТЕРНОПІЛЬ - 2024**

**Факультет комп'ютерних інформаційних технологій**
Кафедра кібербезпеки
Освітній ступінь «магістр»
спеціальність: 125 – Кібербезпека та захист інформації
освітньо-професійна програма – Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ В.В.Яцків
«____» _____ 2023 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

**ПРОКОПЕНКО ДЕНИС ПЕТРОВИЧ**
**(прізвище, ім'я, по батькові)**

**1. Тема кваліфікаційної роботи:**

**Алгоритми виявлення спаму на основі машинного навчання / Spam detection algorithms based on machine learning**

керівник роботи д.т.н., професор В.В. Яцків

затверджені наказом по університету від 12 грудня 2023 року № 753

2. Строк подання студентом закінченої кваліфікаційної роботи 12 грудня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
- спам та його види;
- методи розсилки спаму;
- розглянути алгоритми виявлення спаму на основі МН;
- алгоритм визначення спаму;

5. Перелік графічного матеріалу у роботі:
–види спаму та захист від нього;
– схема роботи спам робота;
– класифікація методів SPAM – фільтрації

6. Консультанти розділів кваліфікаційної роботи

| | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| | | | |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

| № зп/п | Назва етапів кваліфікаційної роботи | Строки виконання етапів кваліфікаційної роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз підходів виявлення спаму | 12.2023 р. –03.2024 р. | |
| 2 | Алгоритми виявлення спаму на основі машинного навчання | 03.2024 р. –06.2024 р. | |
| 3 | Розробка та дослідження алгоритмів антиспаму | 06.2024 р. – 11.2024 р. | |

Студент _____ Прокопенко Д.П.
( підпис )

Керівник роботи _____ д.т.н., професор В.В. Яцків

# АННОТАЦІЯ

Кваліфікаційна робота на тему «Алгоритми виявлення спаму на основі машинного навчання» на здобуття освітнього ступеня «Магістр» зі спеціальності 125 «Кібербезпека та захист інформації» освітньо-професійної програми «Кібербезпека» написана обсягом 104 сторінок і містить 29 ілюстрації, 2 таблиці, 2 додаток та 45 джерела за переліком посилань.

Метою роботи є аналіз методів машинного навчання по виявленню спаму та розробити алгоритм якій буде визначати оптимальність алгоритмів машинного навчання.

Методи досліджень. Для розв’язання поставлених задач у даній кваліфікаційній роботі використано: аналіз підходу виявлення спаму, розглянуті основні методи виявлення спаму та розробка алгоритму для порівнянь різних методів виявлення спаму на основі машинного навчання.

Результати дослідження: розроблено алгоритм для аналізу методів машинного навчання на мові програмування PYTHON. В кінцевому результаті навчались бази даних SVM, RANDOM_FOREST, BAYES.

Показано можливість та переваги тих чи інших методів антиспаму на основі машинного навчання. Результати досліджень показали, що найкраще працює модель SVM, яка найчастіше ідентифікувала спам, найгірше себе показала система, навчена на Bayes.

Результатом роботи стало те, що система дійсно робоча і може навчатись на кастомних CSV, які можна генерувати з власних повідомлень або брати готові вибірки в мережі інтернет.

Ключові слова: спам, антиспам, машинне навчання, python, наївний Байєсівський класифікатор, метод опорних векторів, метод random forest, CSV.

# ANNOTATION

The qualification paper titled "Spam Detection Algorithms Based on Machine Learning", prepared to obtain a Master's degree in specialty 125 "Cybersecurity and Information Protection" under the educational-professional program "Cybersecurity ", is 104 pages long and includes 29 illustrations, 2 tables, 4 appendices, and 45 references.

The goal of the paper is to analyze machine learning methods for spam detection and to develop an algorithm that determines the optimality of machine learning algorithms.

Research methods: To address the objectives of this qualification paper, the following approaches were employed: an analysis of spam detection approaches, a review of primary spam detection methods, and the development of an algorithm for comparing various machine learning-based spam detection methods.

Research results: An algorithm for analyzing machine learning methods was developed using the Python programming language. As a result, datasets were trained using SVM, Random Forest, and Bayes models.

The study demonstrated the feasibility and advantages of various machine learning-based anti-spam methods. Research results indicated that the SVM model performed best, as it most accurately identified spam, whereas the system trained on the Bayes model showed the poorest performance.

The outcome of the work is a functional system capable of learning from custom CSV files, which can be generated from personal messages or sourced from ready-made datasets available online.

Keywords: spam, anti-spam, machine learning, Python, Naive Bayes classifier, support vector machine, random forest method, CSV.

ЗМІСТ

# ВСТУП

**Актуальність роботи**. Боротьба зі спамом є невід’ємною частиною забезпечення кібербезпеки. Спам виступає не лише як дратівливий потік небажаних повідомлень, але і як канал для кіберзагроз, таких як фішинг, розповсюдження шкідливого програмного забезпечення, або здійснення атак соціальної інженерії. За даними останніх досліджень, понад 90% фішингових атак починаються зі спамових повідомлень.

Протягом останніх п'яти років спостерігається стійка тенденція зростання кількості спам-повідомлень, що надсилаються користувачам електронної пошти. Це явище, яке отримало назву "спам", являє собою масове розсилання небажаних комерційних повідомлень, що здійснюється спамерами – особами, які займаються збором електронних адрес з різних джерел, включаючи веб-сайти, чати та інфіковані комп'ютерні системи.

Спам-повідомлення мають негативний вплив на ефективність роботи користувачів електронної пошти, оскільки значно збільшують обсяг нерелевантної інформації, що надходить на їхні поштові скриньки. Це призводить до витрат часу на сортування та видалення спаму, а також до перевантаження систем зберігання даних та мережевих каналів.

Крім того, масштабна розсилка спаму створює значне навантаження на інфраструктуру електронної пошти, включаючи сервери, мережеве обладнання та процесори. Це може призводити до зниження швидкості обробки пошти, втрати даних та інших технічних проблем. За оцінками експертів, спам становить понад 77% всього глобального трафіку електронної пошти, що свідчить про його значний вплив на роботу інтернету.

Негативні наслідки спаму виходять за рамки технічних проблем. Невдоволення користувачів якістю послуг електронної пошти, спричинене спамом, може призвести до втрати клієнтів для провайдерів та інших компаній, що надають послуги, пов'язані з електронною поштою.

Таким чином, проблема спаму є актуальною та потребує подальшого дослідження та розробки ефективних методів боротьби з нею.

Машинне навчання – це розділ штучного інтелекту, який фокусується на розробці алгоритмів, здатних навчатися на даних без явного програмування. Завданням машинного навчання є створення моделей, які можуть самостійно виявляти закономірності в даних, робити прогнози та приймати рішення.

Ключовою особливістю машинного навчання є здатність систем до самонавчання. Алгоритми машинного навчання аналізують великі обсяги даних, виявляють в них патерни та залежності, а потім використовують отримані знання для виконання нових завдань. Таким чином, машини здатні імітувати когнітивні функції людини, такі як узагальнення, класифікація та прогнозування.

Значний прогрес у галузі машинного навчання став можливим завдяки розвитку комп'ютерних технологій. Збільшення обчислювальних потужностей та поява ефективних алгоритмів дозволили створювати складні моделі машинного навчання, які здатні вирішувати широкий спектр завдань.

Прикладами застосування машинного навчання є системи рекомендацій, які аналізують поведінку користувачів для підбору персоналізованих рекомендацій (наприклад, Netflix), системи розпізнавання зображень, які використовуються в безпілотних автомобілях, а також системи прогнозування, які застосовуються в фінансах, медицині та інших галузях.

Машинне навчання є фундаментальним інструментом у сучасній науці про дані. Застосовуючи статистичні методи, алгоритми машинного навчання здатні автоматично виявляти закономірності в даних, здійснювати класифікацію, прогнозування та проводити аналіз текстової інформації. Отримані в результаті моделювання знання можуть бути використані для прийняття обґрунтованих рішень в різних сферах діяльності, від бізнесу до науки, сприяючи оптимізації процесів та підвищенню ефективності.

Сучасні технології в цьому напрямі відкрили нові можливості для розробки адаптивних алгоритмів, які ефективно реагують на динамічні умови

та нові форми спаму. Традиційні антиспам-фільтри, такі як системи, засновані на чорних списках або регулярних виразах, мають істотні обмеження, оскільки не здатні ефективно протидіяти спаму, що постійно змінює структуру чи стиль повідомлень. Натомість сучасні підходи, зокрема глибокі нейронні мережі (DNN) та методи градієнтного бустингу, дозволяють працювати з великими масивами даних, виявляючи складні закономірності в текстах, зображеннях і навіть у метаданих, значно підвищуючи ефективність боротьби зі спамом.

Рекурентні нейронні мережі (RNN) демонструють надзвичайну ефективність при аналізі текстових послідовностей. Їхня унікальна архітектура дає змогу виявляти приховані закономірності навіть у текстах з нестандартним синтаксисом. Це особливо важливо в контексті протидії спам-комунікаціям, де повідомлення постійно трансформуються, щоб уникнути стандартних фільтрів.

Додаткові методи машинного навчання, зокрема Word2Vec та BERT, розширюють можливості інтелектуальних систем. Ці технології забезпечують глибоке розуміння контексту та семантики повідомлень, дозволяючи розпізнавати приховані змісти та маніпулятивні сигнали навіть за умови використання завуальованої лексики.

Ключова перевага цих алгоритмів криється в їхній гнучкості та здатності до постійного вдосконалення. Машинне навчання дозволяє системам швидко реагувати на нові тактики шахраїв. Припустимо, спамери намагаються обійти захисні механізми шляхом додавання незвичайних символів або графічних елементів – інтелектуальні системи миттєво адаптуються, вивчаючи нові патерни та оновлюючи свої фільтри. Завдяки цьому забезпечується висока ефективність захисту та мінімізується ризик проникнення небажаного контенту.

**Мета і завдання дослідження.** Метою дослідження є розробка алгоритму виявлення спаму на основі методів машинного навчання з використанням мови програмування Python, а завдання полягають у аналізі сучасних підходів до виявлення спаму, створенні алгоритму, його реалізації та оцінці ефективності

на тестових даних. Для досягнення цієї мети необхідно вирішити наступні питання:

– дослідити види спаму, класифікації та способи визначення спаму;

– провести аналіз підходів до виявлення спаму;

– дослідити алгоритми виявлення спаму на основі машинного навчання;

– розробити та дослідити алгоритми виявлення спаму.

**Об'єкт дослідження** – процеси збору, аналізу та обміну даними про спам.

**Предмет дослідження –** алгоритми аналізу спам повідомлень.

**Методи досліджень.** Для розв'язання поставлених задач у даній кваліфікаційній роботі використано: методи поширення спаму, методи аналізу та проектування.

**Наукова новизна одержаних результатів.** Розроблено алгоритми аналізу спам повідомлень.

**Практичне значення отриманих результатів.** Результати цього дослідження мають реальне практичне значення, адже створений алгоритм можна інтегрувати в поштові клієнти, корпоративні системи та платформи обміну повідомленнями для підвищення точності фільтрації спаму та зменшення навантаження на IT-ресурси. Завдяки своїй адаптивності алгоритм ефективно сприяє зміцненню кібербезпеки, допомагаючи запобігати фішинговим атакам і поширенню шкідливого програмного забезпечення. Крім того, його гнучкість робить його корисним як у практичному застосуванні для бізнесу, так і як навчальний інструмент для студентів та спеціалістів.

**Публікації та апробація КР**.

1. Прокопенко Д.П. Спам та штучний інтелект. Матеріали XI Міжнародна науково-практична конференція PERSPECTIVES OF CONTEMPORARY SCIENCE: THEORY AND PRACTICE, Львов, 2024. - *С. 365-371.*

2. Прокопенко Д.П. Розробка і дослідження штучний інтелект. Матеріали конференції Development of Education, Science andBusiness: Results 2024., Дніпро, 2024.

3. Прокопенко Д.П. Методи машинного навчання для протидії спаму. Матеріали X Всеукраїнської науково-практичної конференції Інтелектуальні комп’ютерні системи та мережі, Тернопіль, 2024 - С. 137-140.

# 1 АНАЛІЗ ПІДХОДІВ ВИЯВЛЕННЯ СПАМУ

## 1.1 Спам та його види

"Історично термін "спам" був пов'язаний виключно з продуктом харчування. Однак з часом значення цього слова зазнало радикальних змін, перетворившись на загальноприйнятий термін для позначення небажаних масових розсилок. Все почалося у 1937 року, коли компанія Hormel Foods Corporation представила консервовану свинину під маркою Spam. Незважаючи на те, що точне походження абревіатури Spam досі є предметом дискусій, найпоширеніша версія пов'язує її з англійським словосполученням "spiced ham" [1]."основного інгредієнта нових консервів (рисунок 1.1) [2].


Рисунок 1.1 - Виставка консервів Spam у Музеї Spam

Мовна трансформація, що відбулася зі словом "спам", є цікавим прикладом впливу масової культури на мову. Скетч "Монті Пайтон", який пародіював масову рекламу, зробив значний внесок у цей процес. Використання слова "спам" у контексті нав'язливої реклами в гумористичній

замальовці сприяло тому, що цей термін став асоціюватися з небажаними повідомленнями в свідомості широкої аудиторії [1]

Скетч групи "Монті Пайтон" створив потужний культурний рух, асоціюючи слово "спам" з нав'язливою рекламою. Цей код був актуалізований у 1986 році, коли користувач Usenet Дейв Родес, скориставшись відносною безконтрольністю ранніх онлайн-платформ, розпочав масову розсилку рекламних повідомлень. Повторюваність цих повідомлень і їхній небажаний характер для користувачів мережі створили прямий паралель зі скетчем "Монті Пайтон", закріпивши термін "спам" як позначення небажаної пошти. [1]

Спроби компанії Hormel Foods захистити свою торгову марку "Spam" від негативу, пов'язаних з масовими розсилками, виявилися неефективними. Незважаючи на ці спроби, слово "спам" продовжує асоціюватися з небажаними повідомленнями, що свідчить про успішність несанкціонованого маркетингу та його вплив на формування споживчого сприйняття.

Розглянемо основні типи спаму:

1) реклама – деякі компанії, що працюють у легальному бізнесовому секторі, використовують спам як один з інструментів маркетингової комунікації. Привабливість такого методу полягає у відносно низьких витратах та потенційно широкому охопленні аудиторії. [3]. Проте, незважаючи на ці переваги, спам часто спричиняє негативну реакцію споживачів, що може суттєво знизити ефективність рекламних компаній;

2) реклама незаконної продукції – спам часто використовується для просування незаконної продукції та послуг, які не можуть бути рекламовані легальними методами, до прикладу: порнографічні матеріали, контрафактні товари (підробки), лікарські засоби з обмеженим обігом, незаконно отримана конфіденційна інформація, контрафактне програмне забезпечення [3];

3) антиреклама – форма поширення інформації, заборонена чинним законодавством про рекламу [3]. Вона включає негативні відгуки або дії, спрямовані на дискредитацію конкурентів і їхньої продукції. Водночас така інформація може поширюватися через небажані комунікації, наприклад, спам;

4) «нігерійські листи» – це вид фішингового шахрайства, яке полягає в надсиланні електронних листів із проханням допомогти у вирішенні фінансової проблеми, пов'язаної з великою сумою грошей; відправники обіцяють значну винагороду в обмін на допомогу, але зазвичай вимагають від одержувача попередньої оплати різних витрат або надання особистої інформації. Метою таких листів є обманом змусити жертву передати гроші або дані, які шахраї можуть використати для фінансової вигоди [3];

5) фішинг – це вид кіберзлочинності, що передбачає введення користувачів в оману з намаганням отримання їхньої конфіденційної інформації, наприклад паролі, номери кредитних карток чи іншу особисту інформацію. Зловмисники зазвичай маскуються під легітимні установи або відомі сервіси, надсилаючи електронні листи, повідомлення чи створюючи фальшиві вебсайти, які виглядають справжніми; основною метою фішингу є змусити жертву надати свою особисту інформацію, яку потім можна використати для фінансового або особистого зиску, такого як крадіжка особистості або злом облікових записів [3].

Крім спаму, поширеними видами кібератак є DoS та DDoS-атаки, а також соціальна інженерія у вигляді масової розсилки повідомлень від імені інших осіб з метою дискредитації або поширення шкідливого програмного забезпечення. Останнє часто використовується для початкової фази кібератак, спрямованих на зараження комп'ютерних систем. Це листи з приголомшливою історією, що розповідає про те, нібито кожен інтернет-провайдер сплачує певну суму на лікування родині потерпілого за кожне пересилання листа. Головною метою цієї розсилки є збір e-mail адрес, оскільки після багатьох пересилань всім знайомим в тексті таких листів часто можна знайти e-mail адреси всіх, кому вони були переслані раніше [3]. І серед наступних одержувачів може бути навіть той, хто ініціював цей спам.

Як користувачу протидіяти спаму. Найбільш ефективним способом боротьби зі спамом є запобігання розкриттю електронної адреси спамерам.

Хоча це завдання є досить складним, існують певні заходи, які можуть мінімізувати ризики:

1) уникати публікації своєї електронної адреси на відкритих веб-ресурсах;

2) за необхідності, якщо адресу електронної пошти доводиться публікувати, можна використати просте кодування, до прикладу "u_s_e_r_(a)_d_o_m_a_i_n_._n_e_t", бо з метою збору електронних адрес для масових розсилок спамери використовують спеціалізоване програмне забезпечення для сканування веб-ресурсів [4]. Незважаючи на це, навіть прості методи маскування електронних адрес можуть ускладнити автоматизований збір даних. Однак, слід зазначити, що сучасні програми для спам-розсилки здатні розпізнавати деякі види маскування. Таким чином, хоча маскування електронних адрес не є панацеєю, воно може створити додаткові перешкоди для спамерів, не завдаючи при цьому суттєвих незручностей легітимним користувачам [4];

3) більшість публічних веб-ресурсів не відображають електронні адреси зареєстрованих користувачів у відкритому доступі. Натомість вони надають можливість надсилати повідомлення, використовуючи псевдонім (нікнейм). У таких випадках реальна адреса залишається прихованою, оскільки вона автоматично підставляється сервером із профілю користувача та недоступна для інших користувачів [4];

4) електронну адресу можна зобразити у вигляді графічного зображення [4]. Для цього існують онлайн-сервіси, які автоматизують цей процес. Однак варто враховувати, що деякі з цих сервісів можуть збирати введені користувачами адреси та передавати їх третім сторонам. Альтернативним підходом є створення такого зображення самостійно за допомогою графічного редактора або написання адреси вручну з подальшим фотографуванням [4];

5) взаємодія з спам-повідомленнями, така як відповідь на них або перехід за вміщеними в них посиланнями, є підтвердженням активності електронної адреси [4]. Це сигнал для спамерів про те, що дана адреса належить реальній людині і може бути використана для подальших розсилок. Таким чином,

ігнорування спаму є одним з найефективніших способів захисту своєї електронної пошти;

6) механізм завантаження зображень в електронних листах може бути використаний для встановлення так званих "веб-маяків" – невеликих невидимих зображень, які завантажуються на сервер відправника при відкритті листа [5]. Ця інформація дозволяє відстежувати активність користувача та збирати дані про його поведінку в мережі. Для мінімізації ризиків рекомендується відключати автоматичне завантаження зображень у поштовому клієнті та вручну вирішувати, які зображення безпечно завантажувати [5];

7) При виборі адреси електронної пошти з точки зору кібербезпеки доцільно зупинитися на складному, довгому і важкому для вгадування імені. Наприклад, існує менше 12 мільйонів комбінацій імен, що складаються з п’яти латинських букв. Навіть із додаванням цифр і символу підкреслення їх кількість збільшується до менш ніж 70 мільйонів. Спамери можуть використовувати метод масового розсилання на всі можливі імена, відсіюючи ті, для яких сервер повертає відповідь "адресата не існує". Щоб зменшити ризик, ім’я адреси бажано зробити довшим за шість символів. Якщо в ньому відсутні цифри, рекомендується мінімальна довжина в сім символів. Крім того, слід уникати використання слів, що існують у будь-якій мові, включаючи власні назви та українські слова, транслітеровані латиницею. Такі адреси можуть бути вгадані шляхом перебору комбінацій за словниковими базами. Застосування більш складних і нестандартних імен значно ускладнює роботу спамерів;

8) регулярна зміна електронної адреси є одним із можливих заходів підвищення рівня безпеки, оскільки ускладнює відстеження користувача зловмисниками. Однак, такий підхід супроводжується значними незручностями, пов'язаними з необхідністю інформувати про зміну адреси всіх контактних осіб.

У всіх методах приховування електронної адреси є суттєвий недолік: вони створюють незручності не лише для потенційних спамерів, а й для

реальних користувачів, які намагаються встановити контакт. Крім того, у багатьох випадках публікація адреси є необхідністю, наприклад, коли це контактна адреса компанії, що забезпечує зв'язок із клієнтами чи партнерами.

Всі ці види спаму і методи протидії покажемо на рисунку 1.2 [6].

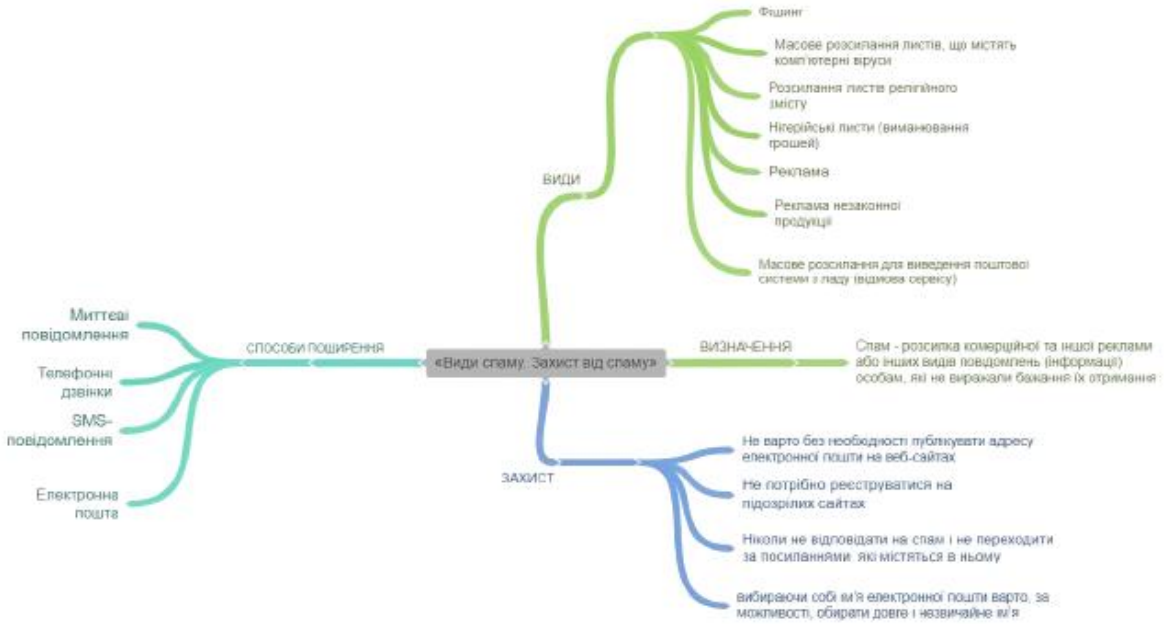



Рисунок 1.2 – Види спаму та захист від нього

## 1.2 Методи розсилки спаму

Один із основних методів є ручний. Подібна розсилка може здійснюватися з реальної адреси або зі спеціально створених для цих цілей скриньок на безкоштовних поштових серверах. Такий спам не може бути масовим, і на перший погляд з ним порівняно просто боротися: вносити адреси та IP-сервери спамерів до чорних списків. Однак якщо спам розсилається, наприклад, з yahoo.com, то внесення IP-адреси yahoo.com до чорного списку блокує прийом пошти з будь-якої поштової скриньки цього сервера. Блокування за поштовими адресами позбавлене цього недоліку, але воно і малоефективне, оскільки спамеру достатньо буде періодично створювати нові скриньки [7].

Розсилка за допомогою спеціальних програм і утиліт відрізняється від попереднього лише ступенем автоматизації – замість відправки листів вручну за адресною книгою спамер використовує спеціалізовану програму, що містить базу даних з адресами для розсилки [7].

Розсилка через некоректно сконфігуровані поштові сервери заснований на пошуку та подальшій експлуатації SMTP-серверів з некоректними налаштуваннями і дозволяє використовувати їх для неавторизованої розсилки спаму. Можна виділити два основні випадки некоректності:

– поштовий сервер приймає і пересилає листи від неавторизованого користувача, тобто є так званим open relay;

– в налаштуваннях поштового сервера встановлено опцію, що передбачає повернення відправнику листів, надісланих на неіснуючу скриньку. Виявивши такий сервер, спамер може підключитися до нього і відправляти листи на заздалегідь відомі йому неіснуючі поштові скриньки, використовуючи адреси зі своєї бази розсилки. Після прийому листа поштовий сервер виявить, що його отримувач некоректний, і поверне повідомлення за адресою відправника, тобто відправить спам отримувачу.

Розсилка із застосуванням веб-інтерфейсу поштових серверів дозволяють відправляти пошту через веб-інтерфейс. Недоліки в механізмах реєстрації та авторизації таких серверів можуть використовуватися спамерами: при виявленні вразливого сервера нескладно написати програму, яка буде автоматично реєструвати поштові скриньки і згодом використовувати їх для розсилки спаму. З метою боротьби з такими методами розробники програмного забезпечення для подібних серверів застосовують складні методи реєстрації, зокрема введення контрольних кодів, що відображаються на формі реєстрації у вигляді картинки, або багатоступеневе підтвердження реєстрації.

Некоректно сконфігурований проксі-сервер для розсилки спаму може використовуватися будь-який Socks-, проксі- або HTTP-проксі-сервер, який підтримує метод CONNECT. Якщо такий проксі-сервер налаштований

некоректно і приймає неавторизовані запити з довільних IP-адрес, то він може бути використаний спамерами.

Троянські проксі – це шкідлива програма, яка перетворює уражений комп’ютер на троянський проксі-сервер. Після активації ця програма передає своїм власникам інформацію про уражений комп’ютер і його IP-адресу, що дозволяє включити уражений комп’ютер у список активних проксі і використовувати його для подальшої розсилки спаму.

Спам-боти – самокеровані програми, які інфікують комп'ютери користувачів та використовують їх ресурси для масової розсилки небажаних повідомлень. Висока ефективність спам-ботів пояснюється їхньою здатністю обходити традиційні засоби захисту, такі як фільтри за IP-адресами [7]. Крім того, спам-боти часто виконують додаткові функції, наприклад, сканують інфіковані комп'ютери на предмет нових електронних адрес для поповнення баз даних спамерів [7].

Механізм роботи спам-бота показаний на рисунку 1.3.

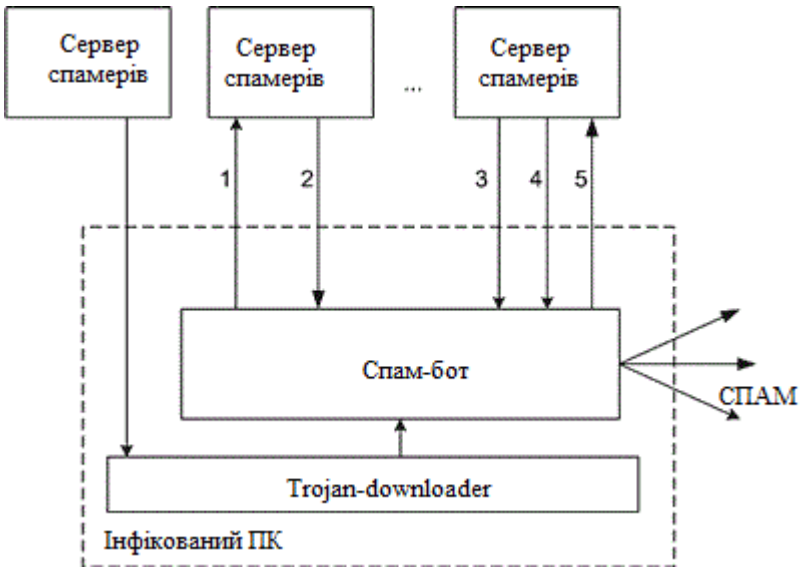



Рисунок 1.3 – Схема роботи спам-бота

Найбільш типовий метод встановлення спам-бота – за допомогою троянського завантажувача. Крім того, для доставки спам-бота можуть застосовуватися поштові та мережеві черв'яки.

Після інфікування комп’ютера спам-бот встановлює зв’язок із сервером, що належить зловмисникам (крок 1), та надсилає звіт про успішну інсталяцію. Адреси таких серверів зазвичай вбудовані у код спам-бота. У відповідь бот отримує конфігураційні дані (крок 2), які вказують, з яких серверів необхідно запитувати інформацію для виконання розсилки спаму [7].

На наступному етапі спам-бот з’єднується із зазначеним сервером (крок 3) і отримує список електронних адрес для розсилки, шаблони повідомлень та інші параметри, необхідні для виконання завдання (крок 4) [7]. Після завершення розсилки бот надсилає зловмисникам звіт, що містить статистику, включаючи список адрес, на які не вдалося доставити повідомлення, із зазначенням причин помилок [7].

Для забезпечення актуальності програмного забезпечення на зараженій машині спам-бот може мати функцію самооновлення або взаємодіяти з Trojan-Downloader, який завантажує оновлення спам-бота на пристрій[7]. Цей механізм підвищує ефективність і стійкість шкідливого програмного забезпечення.

Описаний алгоритм є універсальним, реальні спам-боти можуть мати простішу структуру. Слід зазначити, що, крім розсилки спаму, можлива реалізація ще одного завдання — пошуку на комп'ютері електронних адрес для поповнення спамерських баз.

## 1.3 Сучасні методики боротьби зі спамом та захист від антиспам-фільтрів

Комплексний захист від спаму передбачає реалізацію кількох ключових етапів: аналіз відправника, застосування фільтраційних механізмів та оцінку змісту електронного повідомлення. Технічно ці етапи базуються на двох основних підходах до фільтрації спаму. Перший підхід передбачає аналіз формальних ознак повідомлення, таких як спосіб його відправлення та форматування. Другий – використовує семантичні методи фільтрації, зосереджуючись на аналізі змісту листа для виявлення потенційно небажаних

характеристик. (рис. 1.4). Формальні методи включають фільтрацію за списками (поштових адрес, IP-адрес) та за формальними ознаками листа (наявність багатьох відправників, відсутність одержувача, формат, розмір тощо) [8].

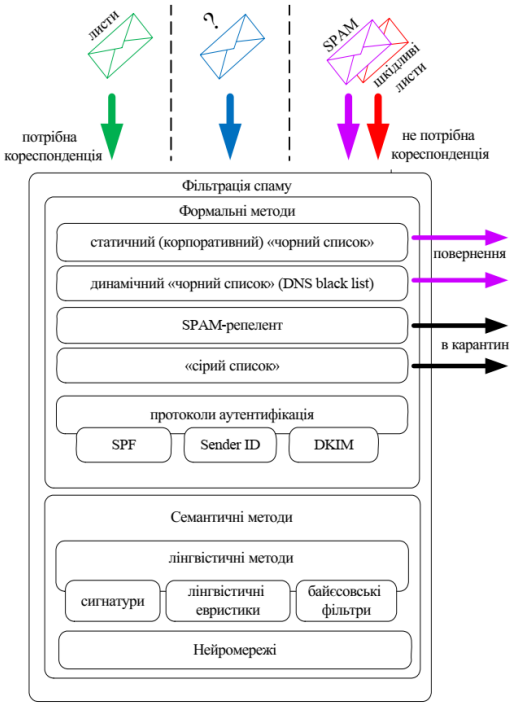



Рисунок 1.4 – Класифікація методів SPAM – фільтрації

Усі сучасні методики боротьби зі спамом можна умовно розділити на певні категорії.

Методи, засновані на аналізі листа — їхнє завдання полягає у вивченні листа з метою його класифікації. Подібний метод вирішує дві задачі: виявляє спам та ідентифікує листи, які гарантовано не є спамом. Відомо кілька найбільш поширених методів аналізу:

1) за формальними ознаками;

2) за вмістом із використанням сигнатурного аналізу. Цей метод базується на пошуку в тексті листа певних сигнатур, описаних в оновлюваній базі даних;

3) за вмістом із застосуванням статистичних методик. Більшість сучасних методів цього класу засновані на теоремі Байєса;

4) за вмістом із використанням SURBL (Spam URL Realtime Block Lists – список блокування спамерських URL). Ідея методу полягає в пошуку

розміщених у тілі листа посилань і їх перевірці за базою SURBL. Цей метод ефективний проти спаму, в якому для обходу фільтрів замість реклами застосовується посилання на сайт із рекламою;

5) детектори масової розсилки – як випливає з назви, їхнім завданням є виявлення розсилки схожого листа великій кількості абонентів;

6) методи, засновані на визнанні відправника листа як спамера – вони спираються на різні чорні списки IP- та поштових адрес. Як і в попередньому випадку, можливе вирішення зворотного завдання – розпізнавання надійних користувачів або поштових серверів, від яких необхідно приймати пошту в будь-якому разі. Перевагою цього методу є те, що для розпізнавання відправника як спамера поштовому серверу не потрібно приймати лист, що суттєво економить трафік. Розпізнавання відправника здійснюється за його IP-адресою;

7) методи, засновані на верифікації зворотної адреси відправника та його домену, – найпростішим методом захисту є звичайний DNS-запит за ім'ям домену відправника листа. Якщо з'ясовується, що домен відправника не існує, то, ймовірно, адреса відправника є підробленою. Проте цей метод малоефективний, оскільки спамери можуть використовувати реальні адреси як адреси відправника, випадково вибрані з бази розсилки. Більш складна технологія передбачає перевірку, чи дозволена відправка листа з вказаною зворотною адресою з даної IP-адреси [8].

Інша методика полягає в тому, що після отримання підозрілого листа антиспам-фільтр може спробувати зв’язатися з відправником листа і запропонувати йому тим чи іншим способом довести, що він не є спамером. Зазвичай перевірка зводиться до відвідування певної сторінки та заповнення на ній веб-форми. Подібну методику, зокрема, практикує фільтр спаму на google.com та ukr.net.

Антиспам – фільтри є важливими інструментами для захисту електронної пошти та інших комунікаційних систем від небажаних повідомлень, але

зловмисленники використовують спроби його обходу. Один із таких способів є модифікації та спотворення тексту

У ранній період становлення спаму масові електронні розсилки здійснювалися без редагування або трансформації початкового тексту.

Це призвело до появи перших фільтрів, причому елементарний з них міг просто розрахувати контрольну суму повідомлення й порівняти його з базою.

Як реакція на посилення фільтрації, спамери створили інноваційні методи автоматизованої зміни тексту, призначені маскувати повідомлення від систем розпізнавання спаму:

– персоналізація повідомлень – це найпростіший метод маскування, що полягає в невеликій модифікації повідомлення з використанням доступних спамеру даних, зокрема e-mail-адреси. Припустимо, є незмінний текст: «Шановний вебмастер! Спеціалісти нашої компанії вивчили ваш сайт і прийшли до висновку, що наша фірма може удосконалити його дизайн…» – нескладно отримати повідомлення, яке буде змінюватися: «Шановний Іванов! Спеціалісти нашої компанії вивчили ваш сайт www.rogaikoputa.com і прийшли до висновку, що наша фірма може удосконалити його дизайн…» Зрозуміло, що ці дані легко можна отримати з поштової адреси rogaikoputa @ukr.net;

– використання пробілів та інших роздільників у словах – наприклад замість «Купуйте супертовар» — «К У П У И ТЕ супер-товар»;

– транслітерація – наприклад виділені напівжирними літерами фрази «Купуйте супертовар» можуть бути замінені ідентичними за начертанням латинськими літерами, при цьому видимої різниці читаючий текст користувач не виявить. Можливі більш помітні на око модифікації, що не впливають на розуміння тексту, — наприклад «][акер», «}{акер» або «4ерний» — у цьому прикладі літери заміняються цифрами або схожими за начертанням наборами символів. Цей прийом ефективний проти байєсовських фільтрів, і для боротьби з ним антиспамова система повинна виконувати нормалізацію тексту та детекцію подібних спотворень;

- перефразування – ефективною технікою маскування спам-контенту, яка передбачає створення численних варіацій повідомлення з однаковим змістом. Механізм роботи цього методу полягає у генеруванні десятків семантично близьких фраз, з яких потім автоматично формується кінцевий лист. Продемонструємо це на прикладі: звороти "Купіть супертовар" та "Придбайте нашу відмінну продукцію" мають практично ідентичний посил, проте для алгоритмів фільтрації, зокрема фільтра Байєса або сигнатурного пошуку, вони виглядають абсолютно по-різному. Додатковим інструментом урізноманітнення тексту слугує застосування словників синонімів, що дозволяє здійснювати довільну заміну слів на контекстуально близькі лексичні еквіваленти;

– додавання чужорідного тексту – цей метод полягає в тому, що окрім реклами, до листа вставляються декілька пропозицій, що не мають ніякого відношення до теми листа, наприклад уривок з будь-якого літературного твору. Цей прийом ускладнює налаштування і погіршує якість роботи фільтра Байєса, але не впливає на якість роботи сигнатурного аналізатора.

Ще одна з спроб обійти системи антиспаму є використання можливостей HTML. Спамери давно взяли HTML на озброєння, оскільки він дозволяє реалізувати кілька простих і ефективних методів маскування. Розглянемо невеликий приклад, підготовлений на основі аналізу основних методів:

*<font color=#FFFFFF>Літаки літаки літаки</font><br>*

*Купуйте товар по супер ціні<br>*

*<!— пролітали пролітали —>*

*Купіть два печева по ціні чотирьох і два получити цілком безкоштовно !<br>*

*<font size=1 color=Gray>низько низько низько...</font>*

*<br>*

*<br><br><br><br><br><br><br><br><br><br><br><br><br>*

*<br><br><br><br>*

*Це невидимий текст 1<br>*

*<table width=”5000" border=”0" cellspacing=”0"*

*cellpadding=”0">*

*<tr><td align=right> Це невидимий текст 2*

*</table>*

Перегляд даного тексту в почтовому клієнті призведе до відображення інформації, який показаний на рисунку 1.5.

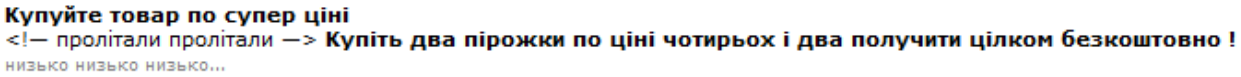



Рисунок 1.5 – HTML-спам

Як легко помітити, перший рядок тексту не видно, оскільки колір символів співпадає з кольором фону. Рядок коментар також не видно, оскільки він є коментарем HTML і ігнорується під час відображення. Нарешті, текст " низько низько низько..." видно, але розмір шрифту та його колір майже не помітні на тлі основних написів. Рядок тексту "Невидимий текст 1" є видимим з точки зору HTML, але непомітним для користувача, оскільки перед ним рядяться маса тегів <br> (примусовий перенос рядка) і він опиняється поза полем зору. Текст "Невидимий текст 2" також бачить HTML, але для користувача, що читає лист, він непомітний, оскільки вирівнюється по правому краю комірки таблиці шириною в 5 тис. пікселів і таким чином також опиняється поза полем зору.

Описані техніки видозміни тексту є нескладними у впровадженні, однак вони суттєво ускладнюють процес аналізу електронного повідомлення за допомогою сигнатурних або статистичних методів. Це пов'язано з тим, що більшість антиспам-фільтрів налаштовані на виявлення найпростіших прийомів приховування інформації: використання невидимого тексту, мініатюрних шрифтів або вбудованих коментарів.

Незважаючи на відносну простоту цих методів, факт застосування текстового маскування може слугувати додатковим індикатором потенційного спам-контенту для фільтраційних систем.

Заміна рекламної інформації URL цей метод полягає в тому, що замість включення реклами в лист, спамери можуть розмістити її на web-сервері й розсилати листи з посиланням на цей сервер. Такі листи можна виявити за допомогою сигнатурних методів (включаючи в базу сигнатур використані спамерами посилання) або за допомогою Spam URI Real-time Block List (SURBL) - система, яка виявляє та блокує спам-посилання у повідомленнях електронної пошти. SURBL використовується для перевірки URL-адрес, які з'являються в електронних листах, з метою виявлення спаму та шахрайства. Ця система обробляє мільйони посилань щодня та додає їх до списку заблокованих, щоб спамери не могли використовувати їх для поширення небажаних повідомлень.

Заміна тексту картинкою прийшла на заміну текстовим повідомленням. Ідея проста: рекламний текст передається у вигляді зображення формату GIF або JPEG, і замість тексту листа в ньому присутнє посилання на зображення. Ця методика зазнала трьох удосконалень:

– використання статичних зображень без додаткового маскування часто виявляється досить ефективним способом обходу байєсівських фільтрів, проте сучасні методи аналізу дозволяють досить легко ідентифікувати такі зображення як потенційний спам. Зокрема, існують дієві підходи протидії: порівняння сигнатур конкретного зображення з наявними базами сигнатур спам-зображень та застосування технологій оптичного розпізнавання символів (OCR). Останні дають змогу автоматично витягувати текстовий вміст з графічних файлів та перетворювати його для подальшого аналізу. Для сигнатурного пошуку мережеві адміністратори успішно використовують інструменти на кшталт Clam AntiVirus, який має відкриті бази даних, що порівняно легко поповнюються власними сигнатурами;

– динамічно генерувані зображення є ефективним засобом обходу систем безпеки, оскільки їх візуальний образ постійно змінюється. Це унеможливлює створення статичних баз даних для виявлення шкідливого контенту. Єдиним способом прочитати інформацію з таких зображень залишається оптичне розпізнавання символів, що може бути трудомістким і не завжди точним;

– динамічно сформоване зображення з анти-OCR-елементами – це візуальні пазли, де кожен фрагмент постійно змінюється. Додавання спеціальних шумів, деформацій та інших перешкод робить неможливим для стандартних алгоритмів точно розпізнати текст. Найпростіші види такого шуму представляють собою хаотично розподілені точки та лінії різної довжини та товщини. Ці елементи створюють візуальні перешкоди, ускладнюючи сегментацію зображення на окремі символи та, як наслідок, їх правильне розпізнавання (рис. 1.6а), у більш складному – застосовуються індивідуальні зміщення кожної букви тексту, випадкові зміни розміру, типу і кольору шрифту, в зображенні присутні складні «шуми», рамки, багатокольорові багатокутники (рис. 1.6 б). Наявність графічного шуму суттєво ускладнює процес оптичного розпізнавання символів, що обмежує можливості сучасних OCR-систем;


Рисунок 1.6 - Динамічно сформоване зображення з анти-OCR-елементами

- динамічно сформоване зображення з анти-OCR-елементами та ефектом 25-го кадру — вони аналогічні попереднім методам, але мають одну новинку: зображення є анімованою GIF-картинкою, що містить кілька кадрів — основний і 1-2 кадри, які відображаються ненадовго і, крім графічних «шумів»,

містять «установчі фрази», наприклад, «BUY!» (рис. 1.7). Очевидно, творці цього спаму вирішили, що короткочасні спалахи кадрів з такими словами повинні впливати на підсвідомість читача. Звісно, ніякого значущого впливу вони на людину не мають, але мерехтливе зображення привертає увагу та ускладнює роботу аналізаторів антиспам-системи.


Рисунок 1.5 – Приклад використання в спамі ефекту 25 -го кадру

# 2 АЛГОРИТМИ ВИЯВЛЕННЯ СПАМУ НА ОСНОВІ МАШИННОГО НАВЧАННЯ

## 2.1 Наївний байєсівський класифікатор

Наївний Байєсівський класифікатор являє собою відносно простий та елегантний інструмент у світі машинного навчання, який ґрунтується на оригінальному статистичному підході. Його принципова особливість полягає в застосуванні теореми Байєса з умовним припущенням про незалежність ознак, що суттєво спрощує математичні обчислення.

Сфера застосування цього методу надзвичайно широка: від класифікації текстових документів до побудови складних фільтрів спаму та розроблення рекомендаційних систем. Перевагами класифікатора є висока швидкість навчання та ефективність роботи з масивними наборами даних.

Попри свою концептуальну "наївність", метод демонструє вражаючу точність у багатьох практичних задачах, особливо коли вхідні дані відповідають базовим статистичним припущенням моделі.

Цей класифікатор набув широкого застосування на початку 1960-х років [9] і досі залишається популярним у багатьох сферах. Завдяки належній попередній обробці даних, він демонструє конкурентоспроможність навіть порівняно з більш сучасними підходами, такими як метод опорних векторів (Support Vector Machine)..

Наївні байєсовські класифікатори — популярний статистичний метод фільтрації електронної пошти [10]. Зазвичай вони використовують набір слів/ознаків для ідентифікації спам-повідомлень, підхід, який часто використовується в класифікації текстів. Наївні байєсовські класифікатори працюють, порівнюючи використання токенів (зазвичай слів, а іноді й інших речей) зі спам-повідомленнями та неспам-повідомленнями, а потім, використовуючи теорему Байєса, обчислюють ймовірність того, чи є повідомлення спамом чи ні.

Певні слова мають певну ймовірність появи у спам-листах і в легітимних листах. Наприклад, більшість користувачів електронної пошти часто стикаються зі словами «Lottery» і «Luck Draw» у спам-листах, але рідко бачать їх в інших листах. Кожне слово в листі вносить свій вклад у ймовірність того, що лист є спамом, або тільки найбільш цікаві слова. Цей вклад називається апостеріорною ймовірністю і обчислюється з використанням теореми Байєса. Потім ймовірність спаму листа обчислюється за всіма словами в листі, і якщо загальне значення перевищує певний поріг (наприклад, 95%), фільтр позначає лист як спам.

Нехай документи розбиті на кілька класів $c_1,...,c_k$, $C$ – загальна безліч класів. Суть її полягає у тому, що ймовірність того, що документ $d$ потрапить в клас $c$, записується як $P(c|d)$:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)},$$

де, $P(d|c)$ – ймовірність зустріти документ $d$ серед всіх документів класу $c$;

$P(c)$ – безумовна ймовірність зустріти документ класу $c$ в корпусі документів;

$P(d)$ – безумовна ймовірність документа наявності $d$ в корпусі документів.

Простіше кажучи, ми ігноруємо той факт, що слова в тексті не існують ізольовано. Вони взаємопов’язані між собою, і ймовірність появи одного слова залежить від контексту, тобто від інших слів у реченні. Наприклад, слово "інтеграл" частіше зустрічається разом зі словом "рівняння", ніж зі словом "бактерія". Ця модель не враховує таких взаємозв’язків, тому її називають наївною [11].

Щоб оцінити умовну ймовірність $P(d|c) = P(t1, t2, ..., tn\ |c)$, де $tk$ – терм з документа $d$, $n$ – загальна кількість термів в документі (включаючи повторення), необхідно ввести припущення про умовну незалежності термів і про незалежність позицій термів.

Уникаючи математичних перетворень та висновків наведем остаточний варіант формули, яку використовують в більшості реалізацій

$$c_m = \arg\max\left[\log P(c) + \sum_{k=1}^{n} \log P(t_k|c)\right].$$

Ця формула має просту інтерпретацію. Шанси класифікувати документ часто зустрічається класом вище, і доданок $\log P(c)$ вносить в загальну суму відповідний внесок [12]. Величини $\log P(t_k|c)$ тим більше, ніж важливіше терм $t$ для ідентифікації класу $c$, і, відповідно, тим вагомішим їх внесок в загальну суму [12].

Основні переваги цього алгоритму:

1) наївний Байєсівський класифікатор базується на простих ймовірнісних моделях, що робить його легким у реалізації та добре працює з великими наборами даних;

2) здатність ефективно працювати з текстовими даними;

3) швидко навчається і здатний обробляти великі обсяги даних, що робить його ідеальним для швидкої класифікації;

4) алгоритм може давати хороші результати навіть при наявності невеликої кількості даних для навчання, на відміну від більш складних моделей.

Але не дивлячись на простоту алгоритму він має і недоліки, а саме:

1) припущення про те, що всі ознаки незалежні одна від одної, що не завжди відповідає дійсності, що може призвести до зниження точності класифікації, якщо ознаки є сильно залежними;

2) через припущення про незалежність, алгоритм не завжди добре працює з складними датасетами, де є взаємодія між ознаками;

3) наївний Байєсівський класифікатор краще працює з категоріальними даними, а для числових даних може вимагати додаткових перетворень або попередньої обробки;

4) алгоритм потребує якісного збору та підготовки даних, зокрема правильного вибору та обробки ознак, хоча це притомано будь-якому алгоритму машинного навчання.

Таким чином, наївний Байєсівський класифікатор є ефективним інструментом для багатьох задач класифікації, особливо коли простота і швидкість є пріоритетними. Однак, його обмеження, пов'язані з припущенням про незалежність ознак, означають, що він не завжди є найкращим вибором для складних або сильно корельованих даних [13].

## 2.2 Метод опорних векторів

У машинному навчанні метод опорних векторів (Support Vector Machine, SVM) є підходом до аналізу даних, який використовується для задач класифікації та регресії. Цей метод належить до моделей із керованим навчанням і базується на алгоритмах, відомих як опорно-векторні машини. Модель SVM представляє зразки у вигляді точок у багатовимірному просторі, де їх розміщують таким чином, щоб зразки з різних категорій були розділені максимально широкою межою, що забезпечує чітке розмежування між класами [14].

Формально, SVM будує гіперплощину (або множину гіперплощин) у просторі високої або навіть нескінченної розмірності з метою розв’язання задач класифікації, регресії та інших. Оптимальне розділення досягається шляхом побудови гіперплощини, що максимізує відстань до найближчих точок навчальної вибірки, що належать до різних класів. [15].

Це пояснюється тим, що більша відстань між класами зазвичай знижує похибку узагальнення класифікатора, підвищуючи його ефективність на нових даних.

Метод опорних векторів (SVM) є потужним інструментом машинного навчання, який широко використовується для задач класифікації, зокрема в обробці природної мови (наприклад, визначення категорій текстів, фільтрація

спаму) та комп'ютерному зорі (розпізнавання образів, класифікація за кольором). SVM також застосовується в системах оптичного розпізнавання символів (OCR) для автоматизації обробки рукописних текстів [16].

Для розуміння принципу роботи SVM необхідно ознайомитися з наступними ключовими поняттями:

- відділяюча гіперплощина - лінійна функція, що розділяє простір ознак на області, кожна з яких відповідає певному класу.
- гіперплощина максимальної межі - оптимальна розділяюча гіперплощина, яка максимізує відстань до найближчих точок даних різних класів.
- м’яка межа-параметр, що дозволяє деяким точкам даних порушувати обмеження, пов’язані з гіперплощиною, для підвищення загальної точності класифікації.
- функція ядра - нелінійна функція, що дозволяє відобразити дані в простір вищої розмірності, де їх можна розділити лінійною гіперплощиною.

Відділяюча гіперплощина є математичною конструкцією, яка слугує для розмежування класів об’єктів з подібними характеристиками. Наприклад, як показано на рисунку 2.1, у тривимірному просторі гіперплощина розділяє світлі кульки від темних, забезпечуючи чітке розмежування між цими двома класами.

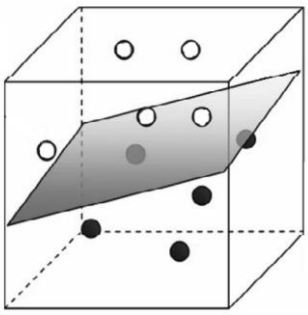


Рисунок 2.1 - Приклад відділяючої площини

Цю процедуру можна математично узагальнити на простори вищої розмірності, ніж три. У багатовимірному просторі, лінія, що розділяє різні класи даних, називається гіперплощиною. Варто зазначити, що розташування такої гіперплощини, отриманої за допомогою алгоритму SVM, не є

однозначним. Існує безліч можливих гіперплощин, які можуть розділити дані. (рисунку 2.2) [17].

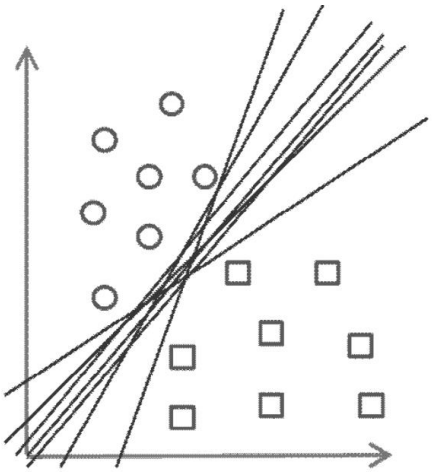


Рисунок 2.2 – Можливі варіанти розташування гіперплощини у двохвимірному просторі

Переваги використання SVM для виявлення спаму:

1) забезпечує високу точність класифікації, особливо коли дані добре підготовлені;

2) завдяки регуляризації, SVM може ефективно уникати перевитрати, навіть у випадках з великою кількістю ознак;

3) SVM добре справляється з високовимірними текстовими даними, такими як електронні листи.

Але системи SVM мають свої недоліки:

1) SVM може вимагати значних обчислювальних ресурсів, особливо для великих наборів даних;
2) вибір правильного ядра та його параметрів може бути складним і вимагає експериментів.

Метод опорних векторів залишається одним з найефективніших алгоритмів для виявлення спаму завдяки своїй здатності точно класифікувати дані навіть у складних просторах.

## 2.3 Штучні нейронні мережі та алгоритм зворотного поширення помилки

Штучні нейронні мережі (ШНМ) є обчислювальними моделями, натхненними біологічними нейронними мережами. ШНМ не є окремим алгоритмом, а радше архітектурою, що забезпечує основу для широкого спектру алгоритмів машинного навчання [19]. Вони дозволяють ефективно обробляти складні багатовимірні дані [20]. Системи навчаються на основі прикладів, де кожен зразок даних має відповідну мітку класу. Цей процес дозволяє їм виявляти закономірності та робити узагальнення, які можуть бути застосовані до нових даних. Наприклад, для розпізнавання зображень зображення кішок, системи навчаються на наборах даних, де кожне зображення має відповідну мітку (кіт/не кіт). Алгоритми машинного навчання виявляють візерунки та характеристики, характерні для зображень кішок, і використовують цю інформацію для класифікації нових, раніше не бачених зображень.

Штучні нейронні мережі - це обчислювальні моделі, які намагаються імітувати роботу людського мозку. Вони складаються з великої кількості взаємопов'язаних вузлів (нейронів), які обробляють інформацію. Кожен нейрон отримує сигнали від інших нейронів, обробляє їх і передає далі [20]. Сила зв'язку між нейронами визначається вагою, яка змінюється під час навчання мережі. Ця структура дозволяє мережам навчатися на прикладах і виконувати складні завдання, такі як розпізнавання образів, обробка мови та прогнозування.

Первісною метою створення ШНМ було моделювання процесів вирішення завдань аналогічно до роботи людського мозку. Однак із часом фокус змістився на вирішення конкретних прикладних завдань, що спричинило відхід від біологічної аналогії [18].

Штучні нейронні мережі знаходять застосування у широкому спектрі завдань, таких як комп'ютерне бачення, розпізнавання мовлення, машинний

переклад, фільтрація контенту в соціальних мережах, ігрові системи, відеоігри та медична діагностика.

Одним із ключових методів навчання ШНМ є алгоритм зворотного поширення помилки. Цей метод дозволяє обчислювати градієнт, необхідний для оптимізації ваг мережі. У процесі алгоритму помилка, обчислена на виході, передається назад через усі шари мережі. Алгоритм зворотного поширення широко використовується для навчання глибоких нейронних мереж, забезпечуючи їх ефективну оптимізацію [20].

Нейронні мережі стали популярним підходом для виявлення спаму завдяки своїй здатності знаходити складні закономірності в даних. Вони забезпечують високу точність класифікації, особливо в задачах, пов’язаних з обробкою тексту.

Розглянемо переваги і недоліки деяких видів нейронних мереж які застосовуються для виявлення спаму:

- рекурентні нейронні мережі (RNN) - підходять для обробки послідовних даних, таких як текст, завдяки своїй архітектурі, яка враховує попередні елементи у послідовності, вони добре захоплюють контекст і послідовність у тексті, що може бути важливим для визначення спаму, але можуть мати проблеми з запам'ятовуванням довгих залежностей у тексті;

- довгострокова короткострокова пам’ять (LSTM) - покращеною версією RNN, яка здатна зберігати інформацію на тривалий час завдяки спеціальним механізмам забування і збереження стану, що дає можливість ефективно обробляти довгі текстові повідомлення і можуть виявляти спам, враховуючи довготривалі залежності у тексті;

- згорткові нейронні мережі (CNN) - зазвичай використовуються для обробки зображень, але можуть бути застосовані до тексту шляхом перетворення тексту на вектори. Вони добре виявляють локальні патерни і структури у тексті, що може бути корисним для розпізнавання ключових слів або фраз спаму.

- трансформери, такі як BERT або GPT, використовують механізми самоуваги для обробки тексту, що дозволяє моделі одночасно враховувати весь контекст тексту, це забезпечує дуже високу точність в задачах обробки природної мови, включаючи виявлення спаму, а також вони здатні вловлювати складні патерни та зв'язки між словами.

В загальному можна зазначити, що нейронні мережі є потужним інструментом для виявлення спаму, здатним забезпечити високу точність і адаптивність до нових викликів у боротьбі зі спамом бо можуть виявляти складні патерни у тексті, що забезпечує високу точність виявлення спаму, а також можуть адаптуватися до нових типів спаму за допомогою додаткового навчання. Проте навчання великих нейронних мереж може бути ресурсомістким і вимагати значних обчислювальних потужностей і для ефективного навчання моделей потребує великих обсягів даних, які можуть бути важкими для збирання та анотації [21].

## 2.4 Дерево прийняття рішення

Дерева прийняття рішень (ДПР) є потужним інструментом у сфері статистики та машинного навчання, що використовується для задач класифікації та регресії. ДПР представляють собою ієрархічну структуру, яка відображає процес прийняття рішень на основі набору правил.

Дерево класифікації є різновидом ДПР, де цільова змінна має дискретний характер (наприклад, "так" або "ні", "клас А", "клас В" тощо). Листя в такому дереві відповідають конкретним класам, а гілки представляють логічні правила, які ведуть до цих класів.

Дерево регресії використовується для прогнозування числових значень цільової змінної (наприклад, ціна, вік, температура) [22]. Структура дерева регресії аналогічна до дерева класифікації, але замість класів у листах містяться числові значення.

Дерева прийняття рішень є ієрархічними моделями, що використовуються для класифікації об’єктів на основі їхніх атрибутів. Кожне дерево представляє собою набір правил, які послідовно застосовуються до вхідних даних, ведучи до кінцевого рішення – класифікації об’єкта до певної категорії.

Вузли дерева відповідають атрибутам об’єктів, а гілки – можливим значенням цих атрибутів. Рухаючись від кореня дерева до листя, алгоритм приймає рішення на основі значень атрибутів, поки не досягне листка, яке вказує на клас, до якого належить об’єкт.

Формально, дерево класифікації має такі властивості:

- вузли - кожен внутрішній вузол асоціюється з атрибутом і задає питання про значення цього атрибута.
- гілки- з’єднують вузли і представляють можливі відповіді на питання, задане у вузлі.
- листя - кінцеві вузли дерева, які містять класи або розподіли ймовірностей

класів.

Процес класифікації:

1. Ініціалізація, яка починається з кореня дерева.
2. Перевірка умови, де перевіряється значення атрибута, що відповідає поточному вузлу.
3. Вибір гілки, де вибирається гілка, яка відповідає значенню атрибута.
4. Перехід до наступного вузла – відбувається перехід до вузла, з яким з’єднана вибрана гілка.
5. Повторення, даний процес повторюється до тих пір, поки не буде досягнуто листа.
6. Класифікація, це клас, пов’язаний з листом, вважається класом вхідного об’єкта.

На відміну від таких методів, як нейронні мережі, які часто сприймаються як "чорні скриньки", результати алгоритмів побудови дерев рішень легко

піддаються інтерпретації. Ця характеристика є цінною не лише для класифікації нових об'єктів, але й для загальної інтерпретації моделі класифікації. Дерева рішень дозволяють зрозуміти логіку, за якою конкретний об'єкт віднесено до певного класу, що робить цей підхід зручним і прозорим для користувачів.

Алгоритм побудови дерева рішень не вимагає від користувача попереднього вибору вхідних атрибутів (незалежних змінних). Усі доступні атрибути можуть бути подані на вхід алгоритму, який автоматично визначає найбільш значущі з них і використовує лише ці атрибути для створення дерева. Це суттєво спрощує роботу користувача порівняно, наприклад, з нейронними мережами, де вибір кількості вхідних змінних значно впливає на час навчання та якість моделі.

Дерева прийняття рішень є потужним інструментом у сфері машинного навчання, що відзначаються рядом суттєвих переваг:

- інтуїтивна зрозумілість - дерева прийняття рішень візуалізуються у вигляді

графів, що нагадують природні дерева, що робить їх легко зрозумілими навіть для тих, хто не має глибоких знань зі статистики та машинного навчання. Кожен шлях від кореня до листка дерева представляє логічне правило прийняття рішення, що дозволяє легко простежити процес класифікації або регресії;

- простота підготовки даних - дерева рішень є відносно невибагливими до

якості даних. Вони не вимагають складних процедур нормалізації або створення додаткових змінних (dummy-перемінних), що значно спрощує процес підготовки даних до моделювання;

- універсальність- дерева рішень можуть працювати з різними типами

даних, включаючи як числові, так і категоріальні. Це дозволяє застосовувати їх до широкого спектру задач;

- автоматичний вибір ознак- алгоритми побудови дерев прийняття рішень

здатні автоматично визначати найважливіші ознаки, що впливають на цільову змінну. Це дозволяє зменшити розмірність даних та підвищити ефективність моделі;

- багатофункціональність дерева прийняття рішень можуть бути використані як для задач класифікації, так і для регресії. Це робить їх універсальним інструментом для вирішення різних завдань прогнозування.

Підсумовуючи переваги, дерева прийняття рішень є цінним інструментом для аналізу даних завдяки своїй простоті, універсальності та здатності автоматично виявляти важливі закономірності в даних.

Дерева прийняття рішень, незважаючи на свої переваги, мають і ряд недоліків, які необхідно враховувати при їх застосуванні:

- схильність до перенавчання, яке полягає в тому, що глибокі дерева, тобто дерева з великою кількістю рівнів, мають тенденцію до перенавчання, тобто надмірно адаптуються до навчальних даних, включаючи випадкові шуми. Це призводить до зниження точності на нових даних.
- нестабільність, при незначних змінах в навчальних даних можуть призвести до суттєвих змін у структурі дерева, що знижує стабільність моделі та ускладнює її інтерпретацію;
- обмеження масштабованості, що виникає при побудові дерев рішень для великих наборів даних або задач з великою кількістю класів може бути обчислювально складним завданням;
- локальний оптимум, що виникає коли алгоритми побудови дерев зазвичай використовують жадібний підхід, що гарантує знаходження локально оптимального рішення, але не обов'язково глобально оптимального;
- незбалансованість класів виникає у випадку, коли один з класів представлений у даних значно частіше за інші, може виникнути проблема дисбалансу класів, що може призвести до упереджених моделей.

Для подолання цих недоліків використовуються різні методи:

- обрізка дерев - зменшення розміру дерева для уникнення перенавчання;
- ансамблі дерев - поєднання декількох дерев для підвищення стійкості та точності моделей (наприклад, випадкові ліси);
- балансування класів - використання різних технік для вирівнювання розподілу класів у навчальних даних.

Дерева прийняття рішень є потужним інструментом для аналізу даних, але їх застосування вимагає обережного підходу та врахування можливих проблем. Застосування різних методів для подолання недоліків дерев дозволяє створювати більш надійні та ефективні моделі. [24].

## 2.5 Метод random forest

У сучасному інформаційному суспільстві, де обсяги даних постійно зростають, аналіз неструктурованих даних стає все більш складним завданням. Щоб ефективно працювати з великими масивами інформації, необхідно проводити первинну обробку даних. Цей процес передбачає структурування, узагальнення та виявлення ключових характеристик даних, що дозволяє отримати цінну інформацію для подальшого аналізу.

Для вирішення цієї задачі широко застосовуються методи класифікації, які забезпечують ефективну обробку даних, спрощуючи їхній подальший аналіз і полегшуючи інтерпретацію результатів [25].

Випадковий ліс (Random Forest) є потужним алгоритмом ансамблевого навчання, запропонованим Лео Брейманом та Адель Катлер. Він базується на ідеї побудови множини дерев рішень, кожне з яких будується на випадковій вибірці навчальних даних. Цей метод поєднує в собі два ключових принципи, а саме:

- бэггінг (bootstrap aggregating): Кожне дерево будується на випадковій вибірці з повторенням вихідних даних. Це дозволяє уникнути перенавчання та збільшує різноманітність дерев;

- випадкові підпростори: Під час побудови кожного дерева розглядається лише випадкова підмножина ознак, що додатково збільшує різноманітність моделей.

Дерева рішень є популярним інструментом для вирішення різноманітних завдань машинного навчання. Як зазначають Хасті та інші дослідники, "навчання дерев рішень найбільш відповідає вимогам використання готових процедур для інтелектуального аналізу даних"[26]. Це обумовлено їхньою інваріантністю до масштабування та інших перетворень значень ознак, стійкістю до включення нерелевантних функцій і можливістю створення моделей, які легко перевірити. Водночас, їхня точність часто залишається обмеженою.

Особливо це стосується дуже глибоких дерев, які схильні до вивчення нерегулярних шаблонів у даних, що призводить до перенавчання. Такі моделі демонструють низький зсув, але високу дисперсію, що погіршує їхню здатність до узагальнення.

Випадкові ліси (random forests) пропонують ефективний підхід до зменшення дисперсії, використовуючи усереднення результатів кількох глибоких дерев рішень, кожне з яких навчене на різних підмножинах одного навчального набору. Хоча це призводить до деякого збільшення зміщення і втрати інтерпретації, загалом метод значно підвищує продуктивність кінцевої моделі.

Random Forest можна порівняти з колективом дерев рішень, які працюють спільно. За рахунок об'єднання зусиль багатьох дерев досягається більш висока точність прогнозування порівняно з окремим деревом рішень. Цей метод аналогічний проведенню k-кратної перехресної перевірки, де модель тренується на різних підмножинах даних, а потім об'єднуються результати. (рис 2.3) [27].

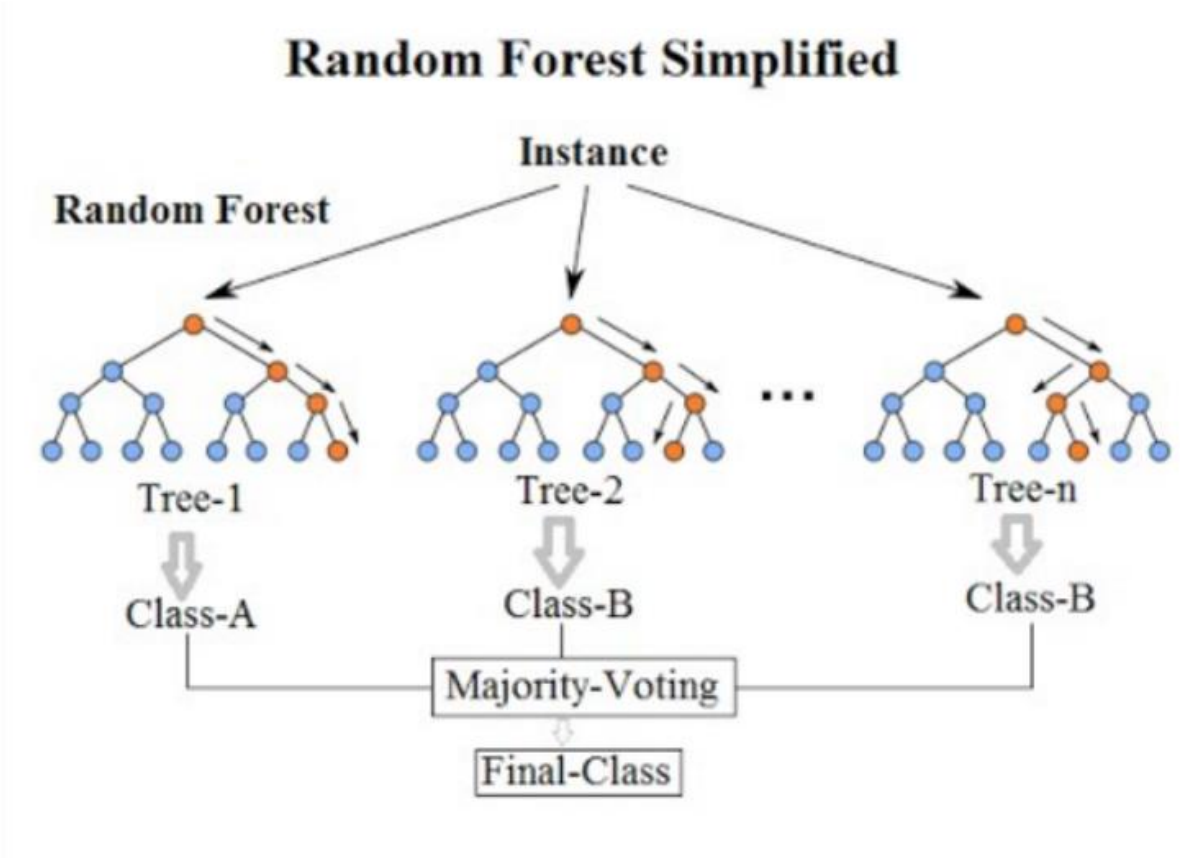



Рисунок 2.3 - Діаграма лісу випадкових рішень

Відмінність методу випадкового лісу від методу дерев у прийнятті рішень наступна:

1) якщо до дерева рішень подати навчальний набір із заданими характеристиками та мітками, алгоритм сформує набір правил, які будуть використовуватися для прогнозування. У випадку методу випадкового лісу вибір характеристик для побудови окремих дерев здійснюється випадковим чином. Після цього результати кількох дерев усереднюються для підвищення точності та стабільності прогнозування;

2) глибокі дерева рішень, тобто дерева з великою кількістю рівнів, мають тенденцію до перенавчання. Це означає, що вони надмірно адаптуються до навчальних даних, включаючи випадкові шуми, і можуть погано узагальнюватися на нові дані. Таке явище часто спостерігається, коли дерево "запам'ятовує" тренувальні дані замість того, щоб вивчати загальні

закономірності.. Однак, це не спрацьовує кожного разу, а також це впливає на час обчислення: чим більша кількість дерев – тим більше часу знадобиться [29].

Для створення випадкового лісу необхідно мати навчальний набір даних, що складається з N спостережень та M ознак. Процес побудови лісу включає в себе наступні кроки:

1. З множини всіх доступних N даних n-разів із повтореннями обираються навчальні підмножини. Кожна підмножина має становити не більше 70% від загальної кількості вибірок. Дані, які не увійшли до навчальної підмножини, використовуються для оцінки похибки дерева, перевіряючи, наскільки правильно воно передбачає класи, до яких належать ці вибірки..

2. У кожному вузлі дерева випадковим чином обирається підмножина ознак розміром (m), яка є значно меншою за загальну кількість ознак M. Рішення для цього вузла базується на вибраних ознаках. Серед них визначається та, яка забезпечує найкраще розщеплення відповідно до поставленої мети, і вона використовується для створення розбиття на цьому вузлі. На наступному вузлі з усієї множини M ознак знову випадковим чином обирається нова підмножина mmm. Така процедура повторюється до побудови кінцевого дерева..

На відміну від беггінгу, де для кожного розгалуження дерева використовується повний набір ознак, випадковий ліс вводить додаткову випадковість. Для кожного вузла дерева випадково вибирається лише підмножина ознак, які розглядаються для розбиття даних. Такий підхід дозволяє уникнути перенавчання моделі та покращує її узагальнюючу здатність.

3. Кожне дерево поступово «зростає» і не обрізається (на відміну від того, як це відбувається у класичному методі дерев прийняття рішень) [30], [31].

При додаванні нових ознак у систему, пошук здійснюється повторно для всіх дерев. Результат обчислюється як середнє значення або середньозважене значення рішень усіх вузлів. У випадку категоріальних змінних кінцевий результат визначається за принципом більшості голосів [30].

На сьогоднішній день існує дуже багато алгоритмів, що реалізують дерева рішень: CART, C4.5, NewId, ITrule, CHAID, CN2 і т.д [32]. Але найбільшого поширення і популярність отримали наступні два:

- CART (Classification and Regression Tree) є фундаментальним алгоритмом машинного навчання, що використовується для побудови бінарних дерев рішень. Цей алгоритм здатний вирішувати як задачі класифікації, так і регресії. Бінарне дерево рішень передбачає, що кожен внутрішній вузол дерева має рівно два нащадки, що відповідають двом можливим результатам тесту на цьому вузлі.;

- C4.5 є однією з найвідоміших і широко використовуваних алгоритмів для побудови дерев рішень. Він є розширенням алгоритму ID3 і вирізняється низкою покращень, які роблять його більш універсальним та ефективним.

Дерева рішень – це універсальний інструмент, який ефективно застосовується як для підтримки прийняття рішень, так і в задачах data mining [33]. Дерева рішень є незамінним інструментом як для бізнес-аналітики, так і для наукових досліджень.

До складу багатьох пакетів, призначених для інтелектуального аналізу даних, вже включені методи побудови дерев рішень. В областях, де висока ціна помилки, вони слугують відмінною підмогою аналітика або керівника.

Випадкові ліси є потужним ансамблевим методом машинного навчання, який демонструє ряд значущих переваг:

1. Висока точність - випадкові ліси зазвичай забезпечують високу точність прогнозування на різних типах даних, що робить їх одним з найбільш популярних методів у практиці data science.
2. Здатність обробляти велику кількість ознак - алгоритм ефективно працює з наборами даних, що містять тисячі ознак, без необхідності попереднього відбору ознак.
3. Нечутливість до підготовки даних: Випадкові ліси не вимагають складних

процедур підготовки даних, таких як нормалізація або стандартизація.

4. Важливість ознак - метод дозволяє оцінити важливість кожної ознаки для прогнозування, що допомагає краще зрозуміти дані та побудувати більш прості моделі.

5. Внутрішня оцінка похибки - випадкові ліси надають вбудований механізм оцінки похибки узагальнення, що дозволяє оцінити якість моделі без необхідності використання додаткових методів перехресної перевірки.

6. Обробка відсутніх даних - алгоритм ефективно справляється з відсутніми даними, зберігаючи високу точність навіть при значній кількості пропусків.

7. Збалансування класів - випадкові ліси можуть бути адаптовані для роботи з незбалансованими наборами даних, де один клас представлений значно частіше за інші.

8. Повторне використання моделей - згенеровані моделі можуть бути збережені та повторно використані для прогнозування на нових даних.

9. Аналіз даних - випадкові ліси можуть бути використані для аналізу даних,

наприклад, для виявлення схожих ознак, виявлення викидів та кластеризації даних.

10.Неконтрольоване навчання – метод може бути застосований для неконтрольованого навчання, такого як кластеризація та виявлення аномалій.

Підсумовуючи переваги**,** випадкові ліси є універсальним інструментом машинного навчання, який може бути застосований для вирішення широкого кола задач. Їх висока точність, простота використання та здатність працювати з різними типами даних роблять їх популярним вибором для багатьох практичних завдань.

Хоча метод випадкових лісів має багато переваг, він також має деякі недоліки, про які варто пам'ятати при його застосуванні:

1. Обчислювальна складність - побудова великої кількості дерев може бути обчислювально дорогим, особливо для великих наборів даних.

2. Складність інтерпретації - інтерпретація окремих дерев може бути складною через їхню нелінійну природу.

3. Перенавчання - незважаючи на те, що випадкові ліси менш схильні до перенавчання, ніж окремі дерева рішень, все ж таки існує ризик перенавчання, особливо при великій кількості дерев.
4. Нечутливість до лінійних залежностей - випадкові ліси можуть бути менш

ефективними для виявлення простих лінійних залежностей між змінними, порівняно з лінійними моделями.

5. Великий розмір моделі - збереження великої кількості дерев може вимагати значних обсягів пам'яті.
6. Чорна скринька - хоча існують методи для інтерпретації випадкових лісів, вони залишаються відносно непрозорими моделями, що ускладнює розуміння причин, що призвели до певного прогнозу.

Таким чином**,** випадкові ліси є потужним інструментом машинного навчання, але їх застосування може бути обмежено обчислювальними ресурсами, складністю інтерпретації та ризиком перенавчання. Вибір випадкових лісів для конкретної задачі залежить від компромісу між точністю, інтерпретованістю та обчислювальною складністю.

## 2.6 Вибір оптимального алгоритму машинного навчання

Для вирішення задачі класифікації спам-повідомлень було обрано три широко використовувані алгоритми машинного навчання: наївний Байєс, метод опорних векторів та випадковий ліс. Вибір цих алгоритмів обумовлений їхньою ефективністю у задачах текстової класифікації та наявністю в науковій літературі численних досліджень, що підтверджують їхню придатність для вирішення подібних задач. Для наївного Байєса буде використана мультиноміальна модель, оскільки вона добре підходить для аналізу текстових даних. Метод опорних векторів буде застосовано з радіальним базисним ядром, яке зазвичай забезпечує високу точність на нелінійно роздільних даних. Для випадкового лісу буде сформовано ансамбль з 100 дерев, використовуючи

критерій інформаційного виграшу для розбиття вузлів. Очікується, що наївний Байєс продемонструє високу швидкість навчання, але може страждати від припущення про умовну незалежність ознак. Метод опорних векторів, як правило, забезпечує високу точність, але може бути чутливим до вибору параметрів ядра. Випадковий ліс, завдяки ансамблевому підходу, очікується, що продемонструє високу стабільність та узагальнюючу здатність

# 3 РОЗРОБКА ТА ДОСЛІДЖЕННЯ АЛГОРИТМІВ

## 3.1 Створення чату для імплементації SPAM CHECKER

Перед створенням системи виявлення спаму, було реалізовано чат на фреймворку Django. Django - фреймворк для мови python, який надає готову архітектуру та інструменти для роботи з базами даних, автентифікації користувачів, керування сесіями, маршрутизації тощо. Оскільки Django вже має багато компонентів, які зазвичай доводиться створювати з нуля, він значно прискорює процес розробки, дозволяючи програмістам зосередитися на бізнес-логіці замість рутинних завдань, таких як налаштування безпеки або робота з формами. Його основний принцип – це "чиста архітектура", спрямована на створення лаконічного та читабельного коду, який легко редагувати та підтримувати.

Для встановлення Django потрібно було мати мову Python. Щоб встановити її, було здійснено перехід за посиланням python.org, що можна побачити на рисунку 3.1 і завантажено останню версію Python.

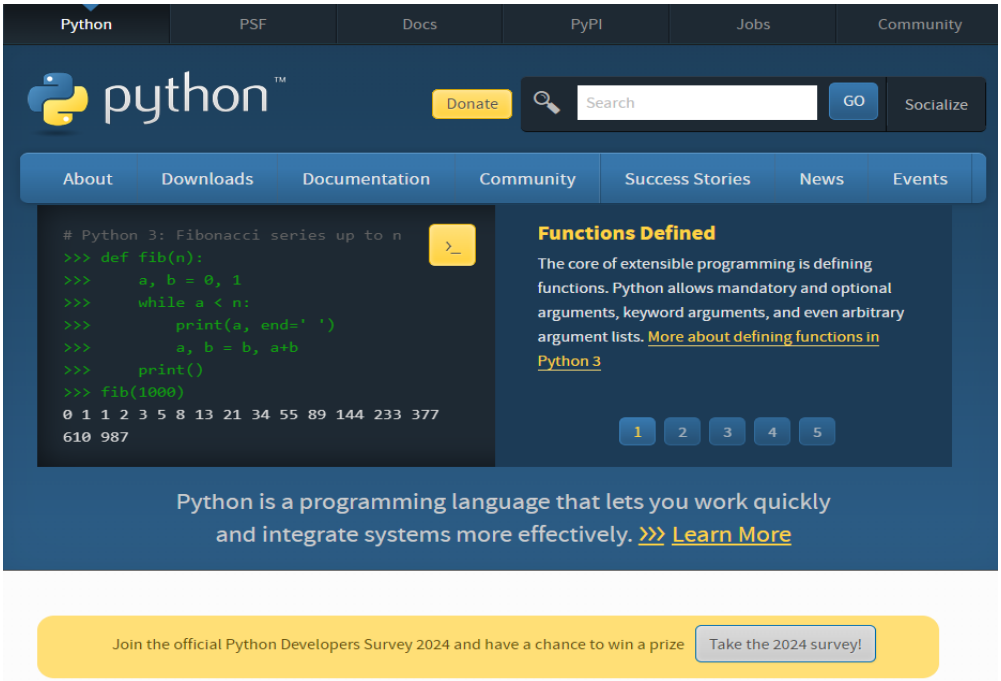



Рисунок 3.1 – Офіційна сторінка сайту Python

Після встановлення python на комп’ютер, через термінал cmd встановлено пакет модулів, які потрібні для нормального функціонування. Перш за все – Django та пакети *django ma channels*

Django, власне відповідає за бек-енд та фронт-енд частину, дозволяючи запустити повноцінний проект - чат, в майбутньому, а channels - відповідає за комплект пакетів, які необхідні для веб-сокетів.

Веб-сокети реалізують повнодуплексний канал зв'язку між клієнтом та сервером, що дозволяє одночасно передавати дані в обох напрямках. На відміну від HTTP, який заснований на парадигмі запит-відповідь, веб-сокети підтримують постійне з'єднання, що забезпечує низьку затримку та високу пропускну здатність.

На рисунку 3.2. зображено різницю між звичайним запитом (POST, GET, UPDATE, DELETE) та запитом у веб-сокетах, де вся інформація надходить по мірі її отримання (без оновлення сторінки/фронтенду).

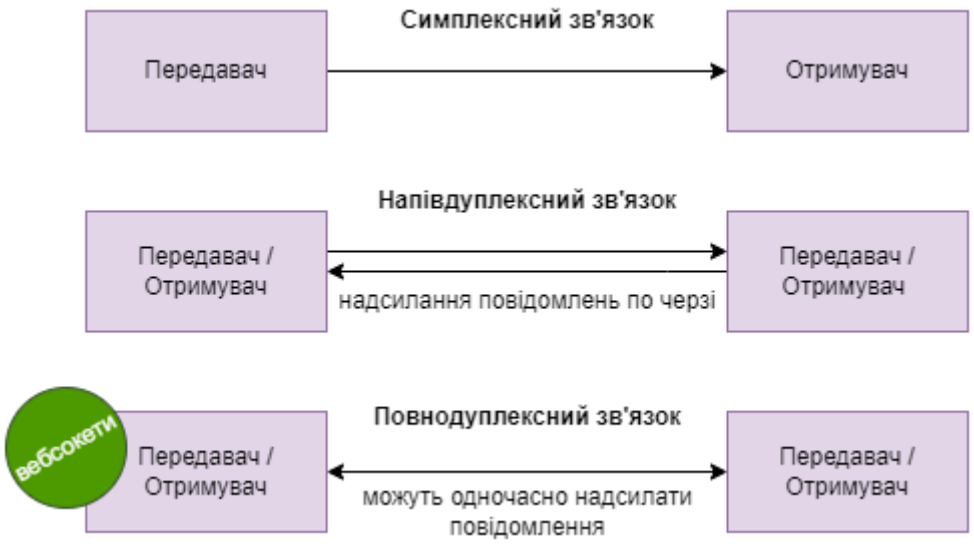



Рисунок 3.2 – Порівняльний рисунок звичайних запитів та веб-сокетів.

У веб-сокетах комунікація відбувається через один, постійно відкритий канал. Це означає, що після встановлення з'єднання, обидві сторони можуть вільно обмінюватися даними, поки не вирішать його закрити.

В роботі було обрано такий тип презентації скрипта, оскільки це найбільш наближено до реального середовища, а також можна поглянути на опрацювання спаму не тільки як для отримувача, а також як і для відправника, де перший отримує декілька повідомлень чи повідомлення, які містять дивний текст, а з другої сторони, де відправник намагається швидко проклікати декілька повідомлень (заспамити, наприклад, однією літерою) або надсилає рекламу, що також є спамом.

Для коректної роботи ASGI і web_socket було зареєстровано аплікацію daphne, що є другою в списку installed_apps. Саме ASGI файл відповідає за веб-сокети в середовищі Django.

ASGI файл - це основний модуль Python-додатку, який використовує асинхронну архітектуру. Він визначає, як ваше додаток відповідає на різні типи запитів (HTTP, WebSocket тощо).Після того, як було встановлено пакети, через команду django-admin tartproject dyploma, було створено проект, а потім прописано python manage.py - startapp chat - для створення аплікації чату, і добавлено її в INSTALLED_APPS (рис.3.3) в основному файлі settings.py

Django.files - стандартні аплікації джанго, які відповідають за обробку адмін-панелі, статичних файлів, сесій користувачів і т.п., а daphne, authenticatnon, channels і chat - кастомні додатки, які окрім того поділяються на ще два типи, де:

Channels і Daphne - аплікації модулів, що дозволяють приєднувати модулі, потрібні для роботи з Django в контексті цього проекту, а chat і authentication - аплікації, які були написані власноруч з 0.

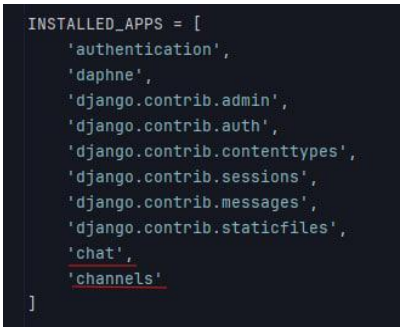



Рисунок 3.3 – Список встановлених додатків

Після створення додатків, було добавлено файл consumers, для роботи з чатами, views.py і models.py (рис.3.4).

```
class ChatRoomConsumer(AsyncWebsocketConsumer):
    Codeium: Refactor | Explain | Docstring | ×
    async def connect(self):
        self.room_name = self.scope['url_route']['kwargs']['room_name']
        self.room_group_name = f'chat_{self.room_name}'

        await self.channel_layer.group_add(
            self.room_group_name,
            self.channel_name
        )
        await self.accept()

    Codeium: Refactor | Explain | Docstring | ×
    async def disconnect(self, code):
        await self.channel_layer.group_discard(
            self.room_group_name,
            self.channel_name
        )

    Codeium: Refactor | Explain | Docstring | ×
    async def receive(self, text_data):
        text_data_json = json.loads(text_data)
        message = text_data_json['message']
```

Рисунок 3.4 – Файл для роботи з чатами

Файл моделі складається з двох класів: ChatRoom і Message.

ChatRoom відповідає за кімнату, до якої можна буде підключитись завдяки consumers і routers, відповідно messages - окремі об’єкти повідомлень, які в майбутньому можна буде аналізувати на вміст СПАМУ.

В моделі ChatRom знаходиться лише одне поле, яке відповідає за його назву. Проте, варто зазначити, що існують фантомні поля, які завжди будуть існувати при любому створенні екземпляру класу - поле ID, унікальний ідентифікатор кожної кімнати.

Нижче представлений код класу моделі.

*from django.db import models*

```
from django.contrib.auth.models import User
class ChatRoom(models.Model):
    name = models.CharField(max_length=100)
    def __str__(self):
        return self.name
```

В той час як в Messages містить в собі ключі, які є дочірніми до користувача і до кімнати, а саме chatroom і sender.

Зміст message включає в себе текст, який буде відправлено користувачем, а timestamp існує для того, щоб відслідковувати кількість часу, затрачену між повідомленнями в разі виникнення спаму.

```
class Message(models.Model):
    chatroom = models.ForeignKey(ChatRoom, on_delete=models.CASCADE)
    sender = models.ForeignKey(User, on_delete=models.CASCADE)
    message = models.TextField()
    timestamp = models.DateTimeField(auto_now_add=True)
    def __str__(self):
        return self.message
```

Після створення моделі, було прописано також і views.py, для можливості взаємодії елементів бекенду та фроненду. Саме в views.py вказуються методи, які будуть перенаправляти з візуальної частини (фронтенду) до бекенду, а в майбутньому до скрипта та бази даних, навченої на нейронних мережах по спаму. Де def index відповідає за те, що виводяться списки кімнат, а room - перехід на веб-сокет ( в плані фронтенду, тільки перехід по сторінкам, логіка прописана в consumers і js).

```
def index(request):
    user = request.user
```

```
    if not user.is_authenticated:
        return redirect('login')
    chat_rooms = get_chatrooms_with_last_message()
        return render(request, 'index.html', context={'chat_rooms': chat_rooms})
def room(request, room_name):
    chatroom, created = ChatRoom.objects.get_or_create(name=room_name)
    chat_messages = Message.objects.filter(chatroom=chatroom)
    context = {
        'room_name': room_name,
        'chat_messages': chat_messages,
        'current_user': request.user.username
    }
    return render(request, 'chat_room.html', context)
```

Після реалізації views.py та моделей, було створно ChatRoomConsumer, який включає в себе методи connect, disconnect, receive.

Connect відповідає за те, щоб при’єднати користувача до сокета, в якому він буде отримувати змінення в базі даних, тобто нові повідомлення і зміни в кімнаті, додавання нових користувачів.

Disconnect відповідає за відключення.

Receive за відправку повідомлень.

На рисунку 3.5. зображено фрагмент коду, що містить в собі room_name, room_group_name і т.п.

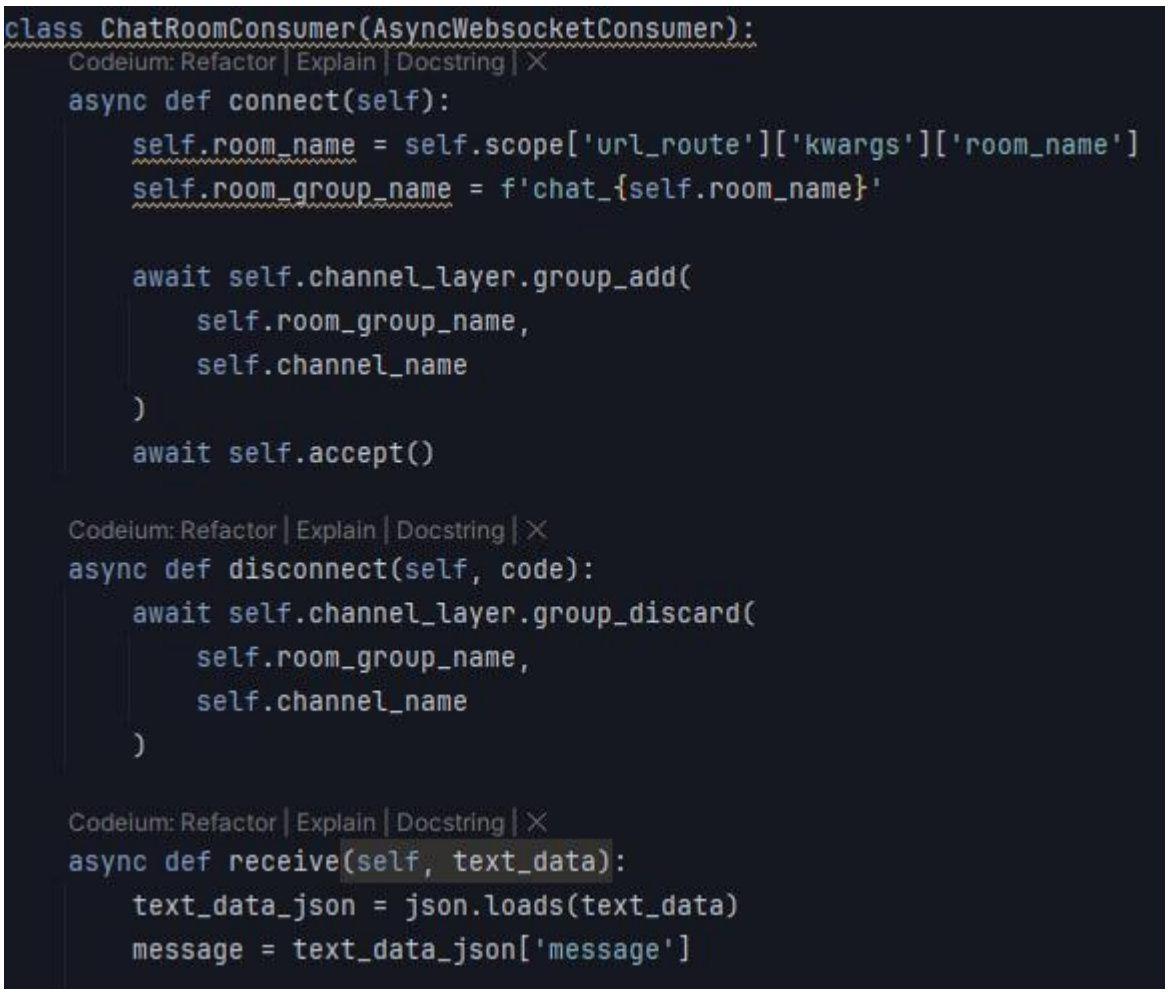



Рисунок 3.5 - Consumers для підєднання до кімнати

Проведено міграції в базу даних, в результаті чого з’явилась sqlite.db за допомогою команди python manage.py makemigrations і python manage.py migrate.

Також було додано роутери в згальні налаштування, щоб Django бачив окрім WSGI → ASGI, а також написанні посилання в web/chat/urls.py для переспрямування людин в кімнату (рис.3.6.), посилання WebSockets в роутерах (рис 3.7), про що було згадано вище в цьому ж розділі, оскільки Django в стандартних налаштуваннях не містить в собі дозволів на підключення вебсокетів.

```python
from channels.auth import AuthMiddlewareStack
from django.core.asgi import get_asgi_application
from channels.routing import ProtocolTypeRouter, URLRouter
from chat import routing

application = ProtocolTypeRouter({
    'http': get_asgi_application(),
    'websocket': AuthMiddlewareStack(URLRouter(routing.websocket_urlpatterns))
})
```

Рисунок 3.6 - Налаштування роутерів в проекті

Посилання на  WebSockets в роутерах приведено на рисунку 3.7.

```python
from django.urls import re_path

from . import consumers

websocket_urlpatterns = [
    re_path(r'ws/chat/(?P<room_name>\w+)/$', consumers.ChatRoomConsumer.as_asgi()),
]
```

Рисунок 3.7 –  Посилання для WebSockets в роутерах

Створено суперкористувача з можливістю заходити на сайт (рис.3.8), в якого є поля username, email, password. Це стандартні Django поля, проте для відтворення середовища, де потрібно проаналізувати спам - цього достатньо.

```
Username: denys
Email address: denys@gmail.com
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

Рисунок 3.8 – Створення користувача

Після створення суперкористувача і проведених всіх міграцій, було запущено сервер (рис. 3.9) з можливістю до нього підключитись (ASGI вказує на те, що все працює коректно).

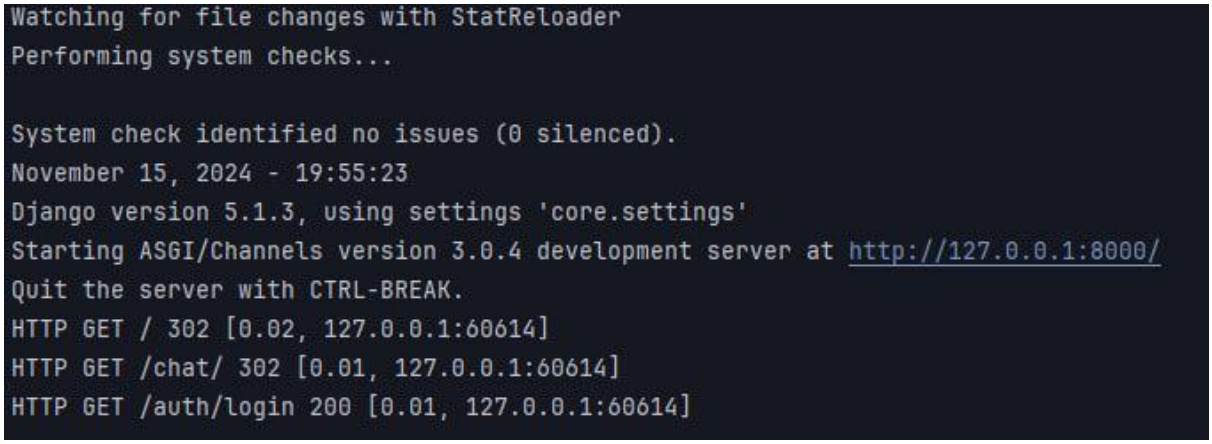



Рисунок 3.9 – Запуск сервера

## 3.2 Навчання моделей для спаму

Після успішного запуску сервера, потрібно добавити модуль для виявлення спаму. Вирішено було зробити його на python.

Модулі, які потрібні:

а. scikit-learn==1.5.1;
б. imbalanced-learn==0.12.3;
в. pandas==2.2.2;
г. nltk==3.9.1;
д. setuptools==74.0.0;
е. pytest==8.3.2;
ж. requests~=2.32.3;
з. imblearn~=0.0;
и. joblib~=1.4.2;
к. xgboost==2.1.1.

Розглянемо більш детально ці модулі:

1) scikit-learn - (також відома як sklearn або scikits.learn) – це потужна екосистема інструментів для машинного навчання в Python. Вона надає широкий спектр алгоритмів, від класичних до сучасних, а також ефективну інфраструктуру для передобробки даних, оцінки моделей та їхнього порівняння. Інтеграція з NumPy та SciPy забезпечує високу продуктивність обчислень;

2) imbalanced-learn – це бібліотека для вирівнювання даних. Користуємося нею, коли в наших даних один тип інформації зустрічається набагато частіше за інший. Наприклад, якби ми хотіли навчити комп'ютер виявляти шахрайські транзакції, але таких транзакцій дуже мало, то imbalanced-learn допомогла б нам "збалансувати" дані і зробити навчання більш ефективним pandas - програмна бібліотека, написана для мови програмування Python для маніпулювання даними та їхнього аналізу. Вона, зокрема, пропонує структури даних та операції для маніпулювання чисельними таблицями та часовими рядами;

3) nltk - інструментів природної мови, або частіше NLTK, – це набір бібліотек і програм для символьної та статистичної обробки природної мови (NLP) для англійської мови, написаних мовою програмування Python. Його розробили Стівен Берд і Едвард Лопер з кафедри комп'ютерних та інформаційних наук університету Пенсільванії [37]. NLTK містить як набори даних, так і графічні матеріали. До пакету входить книга, яка пояснює основні концепції завдань обробки мови, що підтримуються набором інструментів[5], а також прикладами застосування пакету [39];

4) setuptools - це як конструктор для Python-пакетів. Він автоматизує багато рутинних завдань: визначає, які файли і бібліотеки потрібні для вашого проекту, створює готові пакети для встановлення та інтегрується з основними інструментами Python. Це значно спрощує процес розповсюдження вашого коду та його використання іншими розробниками.

5) pytest – це модуль тестування, яка дозволяє користувачам писати тестові коди за допомогою Python мова програмування. Це допомагає вам писати прості та масштабовані тести для баз даних, API або інтерфейсу користувача. PyTest в основному використовується для написання тестів для API. Це допомагає писати тести від простих модульних тестів до складних функціональних тестів;

6) requests – це проста HTTP-бібліотека для Python, створена для людей. Дозволяє надзвичайно легко надсилати HTTP/1.1 запити. Немає необхідності вручну складати URL-адреси або кодувати дані для запитів PUT і POST;

7) imblearn - це бібліотека Python, спеціально розроблена для роботи з незбалансованими наборами даних. Це означає, що коли в наборі даних один клас представлений значно більшою кількістю зразків, ніж інший, imbalanced-learn пропонує методи для вирівнювання цього дисбалансу.

8) joblib – це Python-бібліотека, яка використовується для оптимізації роботи з великими обсягами даних, забезпечуючи ефективну серіалізацію (збереження та завантаження об'єктів) та паралельне виконання обчислень. Вона особливо популярна в машинному навчанні та обробці даних.

9) xgboost – **(**Extreme Gradient Boosting**)** – це високопродуктивна бібліотека для створення моделей машинного навчання на основі методу градієнтного бустингу. Вона особливо популярна серед аналітиків даних і дослідників через свою швидкість, гнучкість і високу точність у прогнозуванні.

Після їх успішного встановлення, було написано модуль, який дозволить навчати моделі на основі файлу csv.

Для створення основного файлу, імпортувались модулі:

import os - це стандартний модуль Python, який забезпечує інтерфейс для взаємодії з операційною системою. Він дозволяє виконувати широкий спектр операцій з файловою системою, середовищем виконання та системними процесами.

imoprt sys - це стандартний модуль Python, який надає доступ до специфічної інформації та функцій, пов’язаних із самим інтерпретатором Python. Він дозволяє взаємодіяти з параметрами запуску програми, середовищем виконання та потоками введення-виведення

from pathib import Path - це зручний інструмент у Python для роботи з файловою системою. Він надає об’єктно-орієнтований спосіб роботи з шляхами до файлів і директорій, що робить код більш читабельним і лаконічним

*from sklearn.model_selection import train_test_split*

*from training.train_models import ModelTrainer*

*from logger_config import init_logging*

*from classifiers.classifier_types import ClassifierType*

Також були імпортовані інші файли  з папочки з скриптами, ттам знаходиться файл **__init__** який вказує на те, що всі файли в папці training - також є модулями.

logger потрібен для відслідковування процесу, оскільки писати UI - часозатратніше та не зовсім коректно, а python дозволить конфігуруватись під лінукс, убунту і т.п.

Даний код демонструє процес навчання моделі машинного навчання у структурованому Python-проєкті.

*import os*

*import sys*

*from pathlib import Path*

*from sklearn.model_selection import train_test_split*

*project_root = Path(__file__).parent.parent*

*sys.path.append(str(project_root))*

*from classifiers.classifier_types import ClassifierType*

*from logger_config import init_logging*

*from training.train_models import ModelTrainer*

*logger = init_logging()*

*def train_model(classifier_type, model_filename, vectoriser_filename, X_train, y_train):*

*logger.info(f'Training {classifier_type}')*

*trainer_ = ModelTrainer(data=None, classifier_type=classifier_type, logger=logger) trainer_.train(X_train, y_train)*

*trainer_.save_model(model_filename, vectoriser_filename)*

Цей код завершує опис функціональності, забезпечуючи тренування кількох моделей на заздалегідь підготовлених і поділених даних.

*if __name__ == '__main__':*

*# Load and preprocess data once*

*data_path = os.path.join(project_root, 'spam_detector_ai', 'data', 'spam.csv') initial_trainer = ModelTrainer(data_path=data_path, logger=logger)*

*processed_data = initial_trainer.preprocess_data_()*

*# Split the data once*

*X__train, _, y__train, _ =*

*train_test_split(processed_data['processed_text'],*

*processed_data['label'],*

*test_size=0.2, random_state=0)*

*# Configurations for each model*

*configurations = [*

*(ClassifierType.SVM, 'svm_model.joblib', 'svm_vectoriser.joblib'),*

*(ClassifierType.NAIVE_BAYES, 'naive_bayes_model.joblib',*

*'naive_bayes_vectoriser.joblib'),*

*(ClassifierType.RANDOM_FOREST, 'random_forest_model.joblib',*

*'random_forest_vectoriser.joblib'),*

*(ClassifierType.XGB, 'xgb_model.joblib', 'xgb_vectoriser.joblib'),*

*(ClassifierType.LOGISTIC_REGRESSION, 'logistic_regression_model.joblib',*

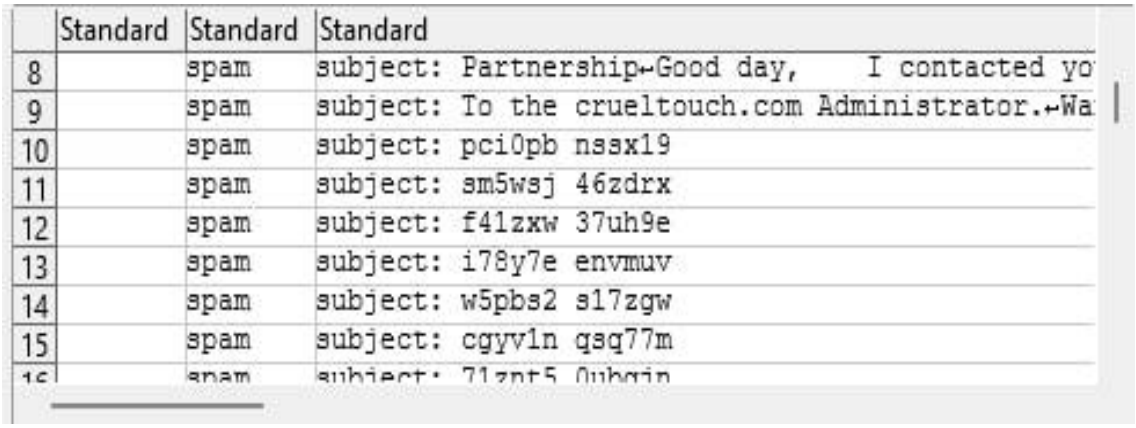
*'logistic_regression_vectoriser.joblib') ]*

*# Train each model with the pre-split data*

*logger.info(f"Train each model with the pre-split data\n") for ct, mf, vf in configurations: train_model(ct, mf, vf, X__train, y__train)*

*((web/spam_detector_ai/trainer.py))*

Проте перед тим було створено вибірку з csv, яка дозволить нейронній мережі навчитись відрізняти спам. В CSV є 3 поля, де 1 - ID, 2 - Тип повідомлення, в нашому випадку - або Спам, або Ham. (гра слів), а третьому - саме повідомлення (рис 3.10).



| | Standard | Standard | Standard |
|---|---|---|---|
| 8 | | spam | subject: Partnership←Good day, I contacted yo |
| 9 | | spam | subject: To the crueltouch.com Administrator.←Wa |
| 10 | | spam | subject: pci0pb nssx19 |
| 11 | | spam | subject: sm5wsj 46zdrx |
| 12 | | spam | subject: f41zxw 37uh9e |
| 13 | | spam | subject: i78y7e envmuv |
| 14 | | spam | subject: w5pbs2 s17zgw |
| 15 | | spam | subject: cgyv1n qsq77m |
| 16 | | spam | subject: 71znt5 0ubgin |

Рисунок 3.10 - Вміст CSV

Деякі з них були згенеровані рандомними символами, для подальшого відслідковування спаму одиночними буквами чи різними символами.

Після чого, запущено процес навчання через trainer.py для створення моделей, як пізніше стануть основою для відслідковування спаму (рис. 3.11), в кінцевому результаті навчались бази даних SVM, RANDOM_FOREST, BAYES.

Алгоритми SVM, RANDOM_FOREST та BAYES були розписані в розділі 2. Нагадаємо коротко про кожний з них

SVM (Support Vector Machine) - це алгоритм, який будує лінію (або гіперплощину в багатовимірному просторі), що найкраще розділяє дані на різні класи. Він добре працює з невеликими та середніми наборами даних та високовимірними проблемами.

Random Forest - це ансамбль алгоритмів, який об'єднує багато дерев рішень. Він дуже точний, стійкий до перенавчання і може працювати з різними типами даних.

BAYES - (особливо Naive Bayes) - це простий і ефективний алгоритм, заснований на теоремі Байєса. Він добре підходить для великих наборів даних і задач класифікації текстів.

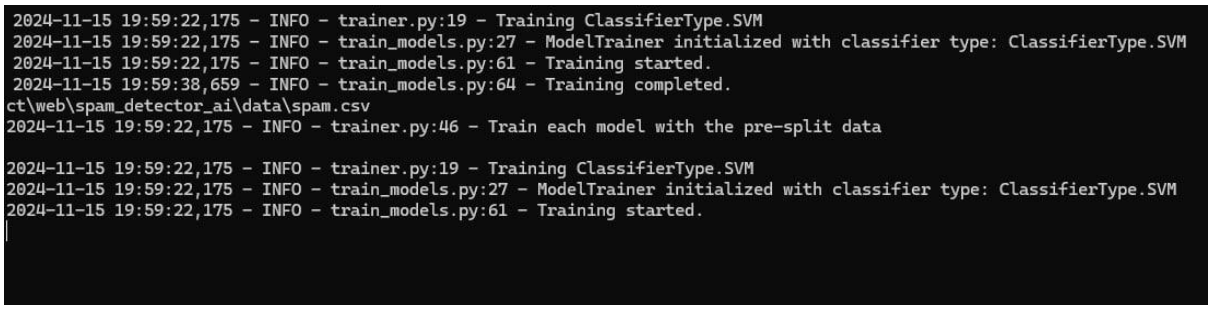



Рисунок 3.11 - Навчання моделі на основі алгоритмів SVM, Random Forest, Bayes (Наївний Байєс).

Особливості і різницю алгоритмів показані в таблиці 3.1.

Таблиця 3.1 – Огляд ключових характеристик алгоритмів машинного навчання

| Характеристика | SVM (Опорні вектори) | Random Forest (Випадковий ліс) | Bayes (Наївний Байєс) |
|---|---|---|---|
| Принцип роботи | Будує гіперплощину, що оптимально розділяє дані. | Ансамбль дерев рішень, кожне з яких будується на випадковій вибірці даних. | Обчислює ймовірність класу на основі теореми Байєса. |
| Переваги | Добре працює з високовимірними даними. | Висока точність, стійкий до перенавчання, обробляє різні типи даних. | Простий, ефективний для великих наборів даних, обробляє пропущені дані. |

Продовження таблиці 3.1

| | | | |
|---|---|---|---|
| Недоліки | Чутливий до вибору ядра та параметрів, складний для інтерпретації. | Може бути обчислювально дорогим для великих наборів даних, складний для інтерпретації окремих дерев. | Припущення про незалежність ознак може бути неточним. |
| Типові застосування | Класифікація, регресія, виявлення аномалій. | Класифікація, регресія, ранжування. | Класифікація текстів, фільтрація спаму, рекомендаційні системи. |
| Стійкість до шуму | Середня | Висока | Середня |
| Інтерпретація | Складна | Складна для окремих дерев, але можна оцінити важливість ознак. | Проста |
| Швидкість роботи | Середня, може бути повільною для великих наборів даних, особливо при використанні нелінійних ядер. | Залежить від кількості дерев, але зазвичай швидка для тренування і прогнозування. | Дуже швидка, особливо для великих наборів даних. |

Після того, як моделі були навчені, проведено текстування (рис.3.12 – 3.14) моделей різних типіс (SVM, RANDOM_FOREST, BAYES).

```
Model: SVM
[[2125   20]
 [  10 1134]]
              precision    recall  f1-score   support

         ham       1.00      0.99      0.99      2145
        spam       0.98      0.99      0.99      1144

    accuracy                           0.99      3289
   macro avg       0.99      0.99      0.99      3289
weighted avg       0.99      0.99      0.99      3289

0.9908786865308604
```

Рисунок 3.12 - Тестування моделі SVM

```
Model: RANDOM_FOREST
[[2129   16]
 [  11 1133]]
              precision    recall  f1-score   support

         ham       0.99      0.99      0.99      2145
        spam       0.99      0.99      0.99      1144

    accuracy                           0.99      3289
   macro avg       0.99      0.99      0.99      3289
weighted avg       0.99      0.99      0.99      3289
```

Рисунок 3.13 - Тестування моделі RANDOW_FOREST

```
Model: NAIVE_BAYES
[[2129   16]
 [ 825  319]]
              precision    recall  f1-score   support

         ham       0.72      0.99      0.84      2145
        spam       0.95      0.28      0.43      1144

    accuracy                           0.74      3289
   macro avg       0.84      0.64      0.63      3289
weighted avg       0.80      0.74      0.69      3289
```

Рисунок 3.14 - Тестування моделі NAÏVE-BAYES

де ham - нормально, spam – спам та  різниця між ступенями нормального і спаму.

В розрізі виглядає наступним чином:

True Negative (TN): 2129 повідомлень було правильно ідентифіковано як не спам.

Помилковий результат (FP): 16 повідомлень було неправильно визначено як спам.

Помилково негативний (FN): 11 спам-повідомлення було неправильно ідентифіковано як спам.

True Positive (TP): 1133 повідомлення було правильно визначено як спам.

Було виявленно, що алгоритм А, показав себе краще ніж Алгоритм Б чи Алгоритм С, оскільки, як видно на рисунках 3.12 - 3.14, найкраще себе показала база даних на основі SVM. І тобто на онові цих формул, повнота виявилась 0.99, вгадано було абсолютно все, результ1.  І зверху якраз є значення TP, TN, FN, FP.

У метриках для оцінки класифікаційних моделей часто використовуються precision (точність), recall (повнота), F1 – score та support.

Розглянемо ці терміни більш детально.

Precision (точність) - є одним з ключових показників ефективності моделей машинного навчання, особливо в задачах класифікації. Вона визначає, яка частка з усіх об'єктів, класифікованих моделлю як позитивні, насправді є позитивними. Іншими словами, точність показує, наскільки модель впевнена у своїх позитивних прогнозах і розраховується по формулі

$$\Pr ecision = \frac{TP}{TP + FP},$$

де, TP (True Positive) - кількість об'єктів, які модель правильно класифікувала як позитивні;

FP (False Positive) - кількість об'єктів, які модель помилково класифікувала як позитивні.

Повнота (Recall) – це ще один важливий показник ефективності моделей машинного навчання, особливо в задачах класифікації. Вона визначає, яка частка всіх дійсно позитивних об'єктів була правильно класифікована моделлю як позитивні. Іншими словами, повнота показує, наскільки добре модель виявляє всі позитивні приклади і визначається по формулі

$$\mathrm{Re}call = \frac{TP}{TP + FN},$$

де, TP (True Positive) - кількість об'єктів, які модель правильно класифікувала як позитивні.

FN (False Negative) - кількість об'єктів, які модель помилково класифікувала як негативні (тобто, не виявила позитивні приклади). Після цього підключено до сайту модуль в consumers.py (рис. 3.15) з виявленням спаму. Який стягує дві бази даних, обрані з навчальних матеріалів, та дозволяє стосовно них пробувати визначати чи є це повідомлення спамом, чи ні.

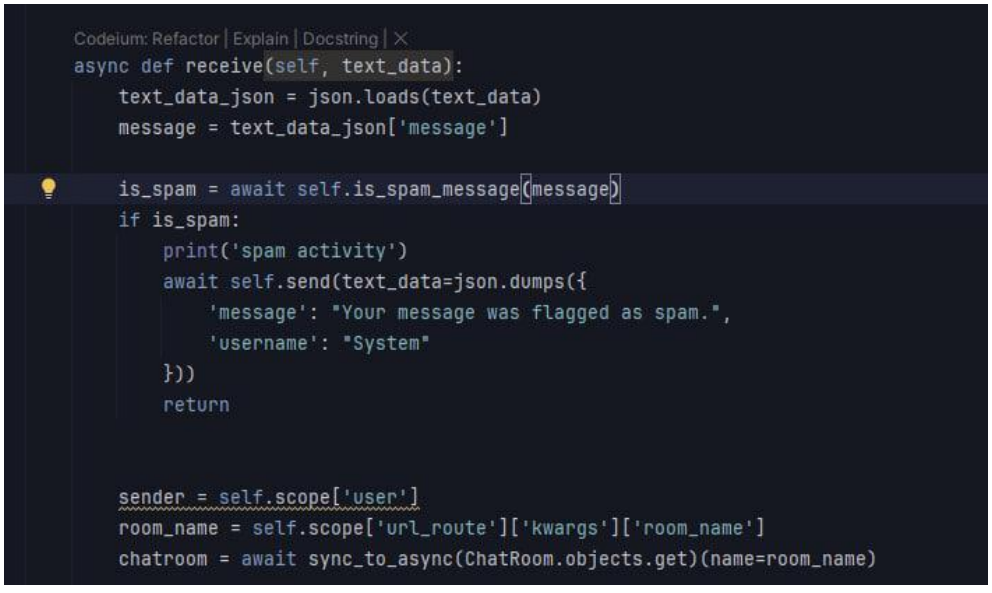



Рисунок 3.15 - Метод в consumers для відслідковування спаму

Потім іде визначення спаму та добавляється асинхроність. Тестування і оцінка показані на рисунку 3.16.

```
    @sync_to_async
    def is_spam_message(self, message):
        """Перевірка, чи є повідомлення спамом."""
        transformed_message = vectorizer.transform([message])
        prediction = spam_model.predict(transformed_message)
        return prediction[0] == 1
```

*sync_to_async - добавилась асинхронність.*

*spam_model.predict - файл і внутрішній метод*

```
spam_model = joblib.load(os.getcwd() + '\\chat\\spam_models\\spam_model.joblib')
vectorizer = joblib.load(os.getcwd() + '\\chat\\spam_models\\vectoriser.joblib')
```

3.16 – Тестування і оцінка

Декоратор @sync_to_async використовується для того, щоб зробити синхронну функцію асинхронною. Це означає, що коли ця функція викликається, вона не блокує виконання інших частин коду, а виконується паралельно. Це дає можливість покращити продуктивність за рахунок того, що ви можете обробляти кілька повідомлень одночасно, що значно підвищить продуктивність, а також дозволяє підтримувати інтерфейс чутливим до дій користувача.

Після проведених операцій, можна перейти до чату. З рисунку 3.9. копіюється адреса сервера, а саме локальний хост, або 127.0.0.1:8000, Переходимо до чату (рис.3.17)

Файл CSS, HTML та JS для цього візуалу знаходиться у додатку А.

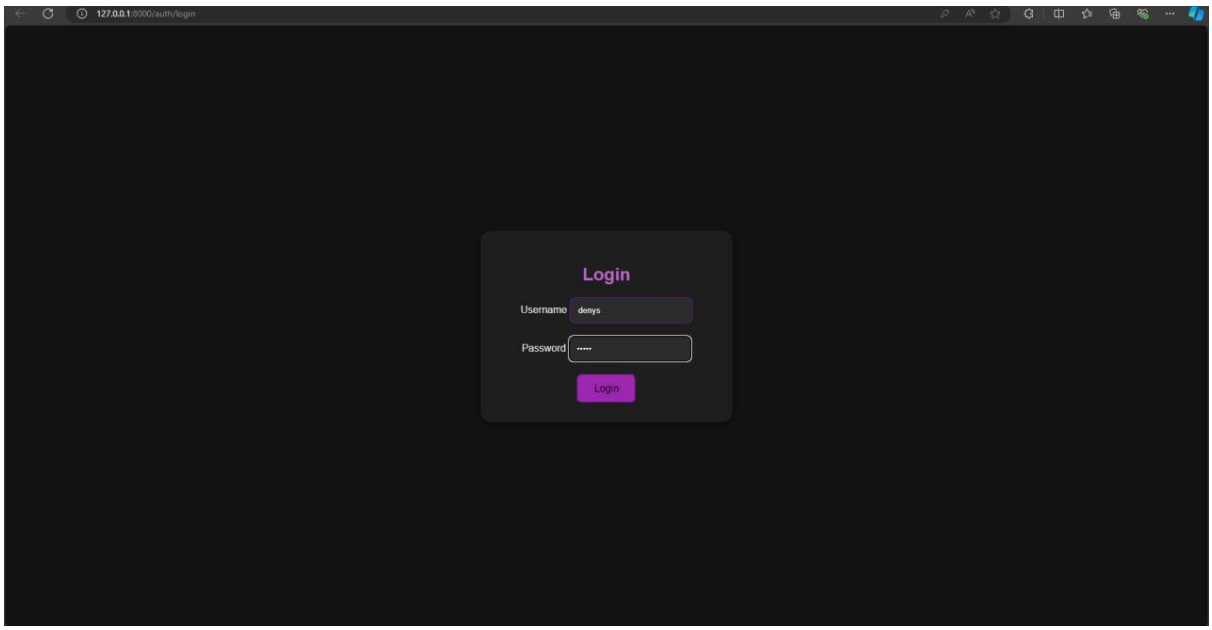



Рисунок 3.17 - Авторизація на сайт з чатом

Перед тим було згенеровано кімнату, створення якої згадувалось вище, для того, щоб мати можливість протестувати скрипт виявлення спаму (рис.3.18).

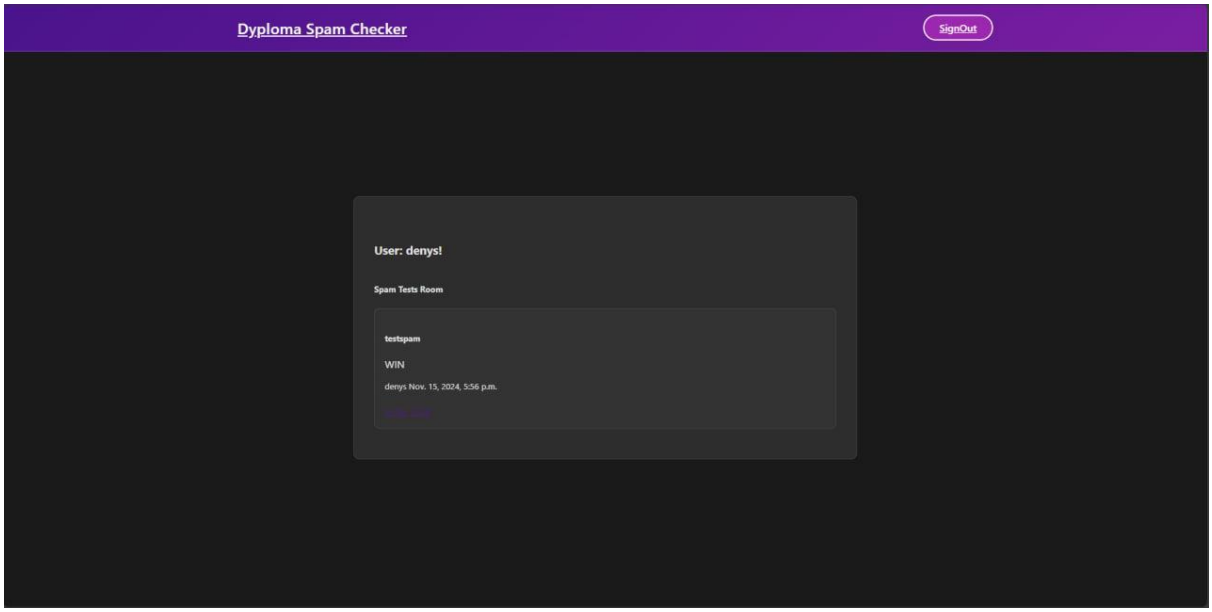



Рисунок 3.18 - Візуал перед кімнатою для спам чекера

Залогінений користувач Denys відправив декілька повідомлень з вмістом f, що мала б відслідкувати система, вона це успішно і виконала, адже відносно

скрипта мав відправитись JSON dump, а також вивестись print зі звичайним текстом: spam, що і відбулось на рисунку 3.19.

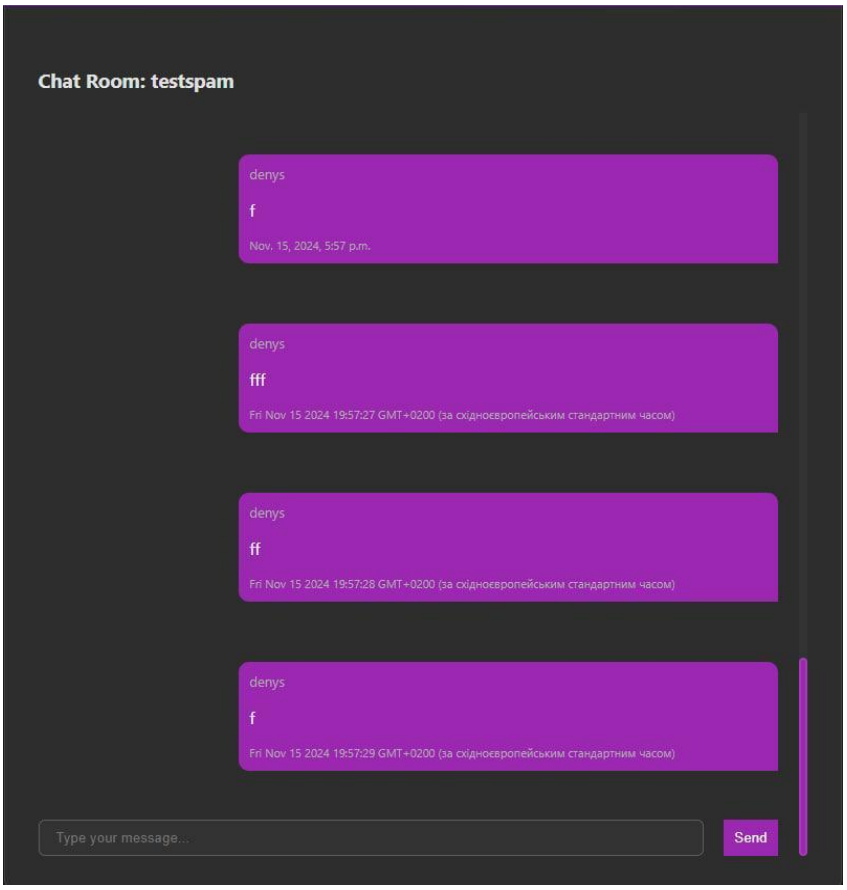



Рисунок 3.19 – Надсилання спаму

Успішно ловиться в консолі і виводить повідомлення на основі баз, які були підгружені (рис. 3.19).

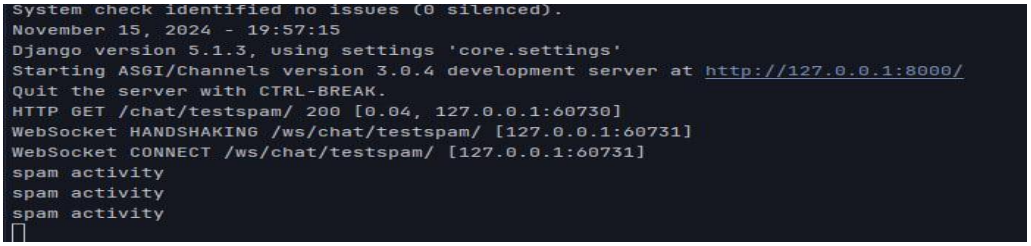



Рисунок 3.19 - Логування спаму в консолі

На скріні також видно, що успішно підключено сокета, відповідно сокет обробив вхід і працює коректно.

Варто зауважити, що найкраще працює модель SVM, яка найчастіше ідентифікувала спам, найгірше себе показала система, навчена на Bayes. Таким чином, можна скласти таблицю 3.2.

Таблиця 3.2– Оцінка спрацювань

| | Тестування моделі SVM | Тестування моделі RANDOW_FOREST | Тестування моделі NAÏVE-BAYES |
|---|---|---|---|
| True Negative (TN): | 2125 | 2129 | 2129 |
| Помилковий результат (FP) | 20 | 16 | 16 |
| Помилково негативний FN): | 10 | 11 | 825 |
| True Positive (TP): | 1134 | 1133 | 319 |
| Похібка | 1,134 | 1,03 | 2,58 |

Результати мого дослідження також підтверджують рисунки 3.12 - 3.14 де скрипт тестування, який перевірив правильність досліджених spam/ham повідомлень вивів такі результати:

Знайдено 2125 спаму, 20 пропущено, 10 пропущено ham, 1134 знайдено. У відношенні до таблиці 3.1. похибка склала всього 1.14%, що є некритичним показником для нейронних мереж.

Результатом роботи стало те, що система дійсно робоча і може навчатись на кастомних CSV, які можна генерувати з власних повідомлень або брати готові вибірки в мережі інтернет.

# ВИСНОВКИ

В кваліфікаційній роботі розв’язано актуальну задачу дослідження ефективності різних алгоритмів машинного навчання для виявлення спаму, що дозволяє внести свій вклад у розвиток методів аналізу текстових даних.

При цьому отримано наступні результати.

1. Проведено аналіз спаму та його види.
2. Розглянуто методи та способи розсилки спаму.
3. Сучасні методи боротьби зі спамом , та способи розсилки спаму.

Наведені методи дуже прості в реалізації, але вони значно ускладнюють аналіз листа за допомогою сигнатур або статистичних методів, оскільки зазвичай антиспам-фільтрами підтримуються прості методики маскування: невидимий текст, текст з малим розміром шрифту, коментарі.

4. Розглянуто алгоритми виявлення спаму на основі машинного навчання. Було розглянуто наївний Байєсівський класифікатор, метод опорних векторів, штучні нейронні мережі, дерева прийняття рішень, метод random forest. Розглянути їх основні переваги та недоліки.

5. Розглянуті алгоритми машинного навчання, SVM, Байєсівський класифікатор та , метод random forest та вибір отпимального алгоритму.

6. Розробка алгоритму для виявлення спаму. Модуль для виявлення спаму було вирішено зробити на мові програмування python. Для цього потрібні були модулі scikit-learn==1.5.1; imbalanced-learn==0.12.3; pandas==2.2.2; nltk==3.9.1; setuptools==74.0.0; pytest==8.3.2; requests~=2.32.3;imblearn~=0.0; joblib~=1.4.2;xgboost==2.1.1.

7. Найкраще працює модель SVM, яка найчастіше ідентифікувала спам, найгірше себе показала система, навчена на Bayes.

8. Результатом роботи стало те, що система дійсно робоча і може навчатись на кастомних CSV, які можна генерувати з власних повідомлень або брати готові вибірки в мережі інтернет.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Чому спам називається спамом: несподівана історія популярного терміна. URL: https://www.grafiati.com/uk/info/dstu-8302-2015/website/ (дата звернення 24.10.2024)

2. Что такое спам и как от него избавиться: пошаговая инструкция. Микола Ладуба Редактор корисних текстів. URL: https://mc.today/tag/email-rassylka/ (дата звернення 24.10.2024)

3. Денисенко І.М. Програмне забезпечення для виявлення спаму на основі машинного навчання: курс. магістр: 121- «Програмна інженерія». Київ, 2020. 34 с.

4. Що таке спам? URL: https://2ip.ua/ua/blog/spam (дата звернення 24.10.2024).

5. Спам, види спаму і боротьба зі спамом. URL: https://best-free-soft.at.ua/publ/spam_vidi_spamu_i_borotba_zi_spamom/1-1-0-33 (дата звернення 24.10.2024).

6. URL:https://coggle.it/diagram/ (дата звернення 24.10.2024).

7. Захист інформації в комп’ютерних системах та мережах : навч. посіб. / С.Г.Семенов, А.О.Подорожняк, О.І.Баленко, С.Ю.Гавриленко – Х.: НТУ «ХПІ», 2014.– 251 с.

8. Kuzma K. Analysis of methods of email filtering from spam / K. Kuzma, V. Zivenko. // Геометричне моделювання та інформаційні технології. – 2017. – №1. – С. 84–89.

9. Maron M. E. Automatic indexing: an experimental inquiry/ maron, m. E.// Journal of the ACM. - 2016. – №8(3). – С. 404–417.

10. Rish I. An empirical study of the naive bayes classifier. / rish i.// in ijcai Workshop on Empirical Methods in AI – 2001.

11. Russell S. Artificial Intelligence: A Modern Approach / S. Russell, P. Norvig. – New Jersey: Prentice Hall, 2003. – 1151 с. – (Prentice hall series in artificial intelligence). – (2).

12. Hastie T. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. / Hastie T, Tibshirani R., Friedman J.. – Verlag: Springer, 2013. – 746 с.

13. Bayesian poisoning // Wikipedia The Free Encyclopedia [Електроний ресурс] // URL: https://en.wikipedia.org/wiki/Bayesian_poisoning — (дата звернення 14.04.2020).

14. Rokach, L., & Maimon, O. (2008). Data mining with decision trees: theory and applications. World Scientific Pub Co Inc.

15. Robert Nisbet. Handbook of Statistical Analysis and Data Mining Applications / Robert Nisbet. – California: Academic Press, 2018. – 792 с. – (Copyright © 2018 Elsevier Inc. All rights reserved). – (Second Edition)

16.Press, William H.; Teukolsky, Saul A.; Vetterling, William T.; Flannery, B. P. (2007). Section 16.5. Support Vector Machines. Numerical Recipes: The Art of Scientific Computing (вид. 3rd). New York: Cambridge University Press. ISBN 978-0-521-88068-8.

17. Training of support vector machine with the use of multivariate normalization/ J. Martínez López[ та ін.]// Applied Soft Computing – 2014 - № 24 – p. 1105-1111

18. Artificial Neural Networks as Models of Neural Information Processing | Frontiers Research Topic. (2018).

19. Encephalos Journal. (2019). Retrieved from URL: www.encephalos.gr (дата звернення 10.11.2024)

20. Build with AI | DeepAI. (2018). DeepAI. Retrieved from URL: https://deepai.org/learn (дата звернення 10.11.2024)

21. Смирнов Л.М. Метод виявлення шкідливого коду з використанням технологій інтелектуального аналізу даних. курс. магістр: 125- «Кіберпезпека». Харків, 2019. 65 с.

22. Rokach, L., & Maimon, O. (2008). Data mining with decision trees: theory and applications. World Scientific Pub Co Inc.

23. Quinlan, J. R. Induction of decision trees// Machine Learning, - 1986 - №1 с. 81– 106.

24. Козін І.В. Методи дерев рішень, класифікації та прогнозування// Запорізький національний університет – 2017 – Підручник – №9.

25. Christopher M. Bishop. Pattern Recognition and Machine Learning Information science and statistics / Christopher M. Bishop. – India: Springer (India) Private Limited, 2013. – 738 с. – (Private Limited).

26. QGIS Development Team. (2023). QGIS Geographic Information System. [Веб-сайт]. https://qgis.org/. (дата звернення 10.11.2024)

27. Пиж О. І. Інтелектуальна система аналізу та класифікації вхідних e-mail повідомлень на основі їх вмісту: курс. магістр: 121- «Інженерія програмного забезпечення». Київ, 2022. 105 с.

28. Christopher M. Bishop. Pattern Recognition and Machine Learning / Christopher M. Bishop. – India: Information Science and Statistics, 2006. – 738 с. – (Springer).

29. Donges, N., 2018. The Random Forest Algorithm. Towards Data Science.

30. Benyamin, D., 2012. A Gentle Introduction to Random Forests, Ensembles, and Performance Metrics in a Commercial System. CitizenNet Blog.

31. Reinstein, I., 2017. Random Forests(r), Explained. KDnuggets.

32. Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques* (3rd ed.). Morgan Kaufmann Publishers.

33. Ian H. Witten. Data Mining Practical Machine Learning Tools and Techniques / Ian H. Witten, Eibe Frank, Mark A. Hall. – Burlington: Morgan Kaufmann Publishers is an imprint of Elsevier, 2011. – 630 с. – (The Morgan Kaufmann series in data management systems). – (Third Edition).

34. Жлуктенко Юлія Пізнавальний потенціал методу «random forest» в емпіричних соціологічних дослідженнях: матеріали XII Міжнародної конференції студентів та молодих науковців м.Київ, 14-15 листопада, 2019р.

35. Leo Breiman and Adele Cutler, 2001. Random Forests. Machine Learning Journal

36 Трухов А.С. Маршрутизація дрону на пересіченій місцевості з використанням методів машинного навчання. курс. магістр: 124- «Системний аналіз». Миколаїв, 2021. 75 с.

37. *Preface. www.nltk.org. Архів оригіналу за 26 січня 2022. Процитовано 15 червня 2016.*

*38.* Bird, Steven; Klein, Ewan; Loper, Edward (2009). *Natural Language Processing with Python*. O'Reilly Media Inc. *ISBN 978-0-596-51649-9.*

39. *Perkins, Jacob (2010). Python Text Processing with NLTK 2.0 Cookbook. Packt Publishing. ISBN 978-1849513609.*

40. Методи машинного навчання при проєктуванні автоматизованих систем керування [Електронний ресурс] : навч. посіб. для аспірантів спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» / Укладач: Т. Г. Баган; КПІ ім. Ігоря Сікорського. Електронні текстові дані (1 файл: 313 кБайт). Київ: КПІ ім. Ігоря Сікорського, 2021. 28 с

41 Chollet, F. Deep Learning with Python / Chollet, F.., 2018. – 500 с. – (Manning Publications)

42. Ма К. Тривимірне глибоке навчання на PYTHON / Ма К., Хегде В., Йольан Л.. – Contrada Rotol: Print2prin, 2023. – 225 с.

41. Омелянко Я. Еволюційні нейромережі мовою Python / Омелянко Я.. – Київ: ДМК Прес, 2022. – 310 с.

42. Richard S. Sutton Reinforcement Learning, second edition: An Introduction (Adaptive Computation and Machine Learning series) / Sutton S. Richard Barto G. Andrew. – 2018. – 552 p.

43. Andriy Yerokhin, Alina Nechyporenko, Andrii Babii, Oleksii Turuta, Ihor Mahdalina. Usage of Phase Space Diagram to Finding Significant Features of Rhinomanometric Signals // Computer Science & Information Technologies (CSIT'2016), 6-10 Sept. 2016, Lviv, Ukraine. – P. 70 – 72. DOI: 10.1109/STCCSIT.2016.7589871.

44. В. В. Залива, А.П. Бондарчук, О. А. Золотухіна Фільтрація спаму в електронної пошти за допомогою машинного навчання . слово науковця. наука, експлуатація, виробництво. Київ, 2019. №6. с. 61-66.

45. Панем Чаранарур1· Суворий джайн2· Г. Срініваса Рао· Дебабрата Самант4· Сандіп Сінгх Сенгар· Чамінда Тушара Хеваге Детектор спаму на основі машинного навчання. SPRINGER NATURE JOURNAL, Інформатика, Сандіп Сінгх Сенгар, 2023. с. 857-870.

Додаток А Файл CSS, HTML та JS візуальної картини

```
{% load static %}

<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">
    <link rel="icon" href="{% static 'chat/favicon.ico' %}">
    <title>Dyploma Spam Checker</title>
    {% block head %}{% endblock %}
    <style>
body {
    background-color: #1a1a1a;
    color: #e0e0e0;
    font-family: 'Segoe UI', system-ui, -apple-system, sans-serif;
    min-height: 100vh;
    margin: 0; /* Remove default body margin */
    display: flex;
    justify-content: center; /* Center content horizontally */
    align-items: center; /* Center content vertically */
}

.navbar {
    background: linear-gradient(135deg, #4a148c, #7b1fa2)
!important;
    width: 100%;
    padding: 1rem;
    position: fixed;
    top: 0;
    left: 0;
    z-index: 1000;
    border-bottom: 2px solid rgba(255, 255, 255, 0.1);
```

```
}

.content-container {
    width: 100%;
    max-width: 800px !important;
    background: #2d2d2d;
    border-radius: 10px;
    padding: 2rem;
    margin-top: 70px; /* Adjust for navbar height */
    border: 1px solid rgba(255, 255, 255, 0.1);
    box-sizing: border-box;
    display: flex;
    flex-direction: column;
    justify-content: center;

}

/* For better control over spacing and alignment */
.navbar-container {
    width: 100%;
    max-width: 1200px;
    margin: 0 auto;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 0 2rem;
}

        .navbar-brand {
            color: #fff !important;
            font-weight: 600;
            font-size: 1.5rem;
            letter-spacing: 0.5px;
        }

        .signout-btn {
```

```
    background: #9c27b0;
    border: 2px solid white;
    color: white;
    padding: 0.5rem 1.5rem;
    border-radius: 25px;
    font-weight: 500;
    transition: all 0.3s ease;
}

.signout-btn:hover {
    background: #ba68c8;
    transform: translateY(-2px);
    box-shadow: 0 4px 12px rgba(156, 39, 176, 0.3);
}



.room-card {
    background: #333;
    border: 1px solid rgba(255, 255, 255, 0.1);
    border-radius: 8px;
    padding: 1rem;
    margin-bottom: 1rem;
    transition: transform 0.2s ease;
}

.room-card:hover {
    transform: translateY(-3px);
    box-shadow: 0 4px 15px rgba(156, 39, 176, 0.2);
    border-color: rgba(255, 255, 255, 0.3);
}

input, textarea, select {
    background-color: #333 !important;
    border: 1px solid rgba(255, 255, 255, 0.2) !important;
    color: #fff !important;
```

```
            border-radius: 6px;
            padding: 0.5rem 1rem;
        }

        input:focus, textarea:focus, select:focus {
            border-color: #9c27b0 !important;
            box-shadow: 0 0 0 2px rgba(156, 39, 176, 0.25)
!important;
            outline: none;
        }

        @keyframes fadeIn {
            from {
                opacity: 0;
                transform: translateY(10px);
            }

            to {
                opacity: 1;
                transform: translateY(0);
            }
        }

        .room-card {
            animation: fadeIn 0.3s ease forwards;
        }
    </style>
</head>

<body>
    <nav class="navbar fixed-top">
        <div class="navbar-container">
            <a class="navbar-brand" href="{% url 'home'
%}">Dyploma Spam Checker</a>
            <a class="signout-btn" href="{% url 'signout'
%}">SignOut</a>
```

```
        </div>
    </nav>

    <div class="content-container mt-5">
        <br>
        {% block content %}
        {% endblock %}
    </div>

    <script      src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/js/bootstrap.bundle.min.js"></script>
</body>

</html>


{% extends 'base.html' %}

{% block head %}

{% endblock %}

{% block content %}
  <h3 class="text-center">Chat Room: {{ room_name }}</h3>

  <div id="chatMessages" class="card mb-3">
    <div class="card-body">
      <div class="row" id="chatBox">
        <!-- Beginning of chat history -->
        <div class="col-2"></div>
        <div class="col-8 text-center">
          <a  class="badge  text-secondary-emphasis  bg-secondary-
subtle gray-900"></a>
        </div>
        <div class="col-2"></div>
        <!-- Beginning of chat messages -->
```

```
        {% for message in chat_messages %}
          <div  class="col-4  d-{%  if  message.sender.username  ==
current_user %}block{% else %}none{% endif %}"></div>
          <div class="col-8">
              <div      class="message      mb-3      {%      if
message.sender.username  ==  current_user  %}self  float-end{%  endif
%}">
                  <div class="message-content">
                    <small       style="font-size:       13px;">{{
message.sender.username }}</small>
                    <p class="mb-1">{{ message.message }}</p>
                    <small       style="font-size:       11px;">{{
message.timestamp }}</small>
                  </div>
              </div>
          </div>
          <div  class="col-4  d-{%  if  message.sender.username  ==
current_user %}none{% else %}block{% endif %}"></div>
        {% endfor %}
      </div>
    </div>
  <br>
    <div class="card-footer">
      <div class="input-group input-group-sm">
        <input  type="text"  class="form-control"  id="messageInput"
placeholder="Type your message..." />
        <div class="input-group-append">
          <button  type="button"  id="sendButton"  class="btn  btn-
primary">
            <i class="fa fa-paper-plane fa-fw"></i>
            <span class="d-none d-sm-inline"> Send</span>
          </button>
        </div>
      </div>
    </div>
  </div>
```

```
  {{ request.user.username|json_script:"user_username" }}
  {{ room_name|json_script:"room-name" }}
  <script>
    // Variables passed through template
    const roomName = JSON.parse(document.getElementById('room-name').textContent);
    const user_username = JSON.parse(document.getElementById('user_username').textContent);

    // Connect WebSocket
    const chatSocket = new WebSocket(`ws://${window.location.host}/ws/chat/${roomName}/`);

    // Receive new messages from Web Socket
    chatSocket.onmessage = function (e) {
      const message = JSON.parse(e.data);
      console.log(message)
      document.getElementById("chatBox").innerHTML += `
        <div class="col-4 d-${message.username == user_username ? 'block' : 'none'}"></div>
        <div class="col-8">
          <div class="message mb-3 ${message.username == user_username ? 'self float-end' : ''}">
            <div class="message-content">
              <small style="font-size: 13px;">${message.username}</small>
              <p class="mb-1">${message.message}</p>
              <small style="font-size: 11px;">${new Date()}</small>
            </div>
          </div>
        </div>
        <div class="col-4 d-${message.username == user_username ? 'none' : 'block'}"></div>
      `;
```

```
    }

    // Send Message to WebSocket
    const sendButton = document.getElementById('sendButton');
    const messageInput = document.getElementById('messageInput');
    sendButton.addEventListener('click', (event) => {
      const message = messageInput.value.trim();
      if (message !== '') {
        // Send the message to the server
        chatSocket.send(JSON.stringify({
            'message': message,
        }));
        messageInput.value = '';
      }
    });
  </script>
    <style>
     body {
    background-color: #1a1a1a;
    color: #e0e0e0;
    font-family: 'Segoe UI', system-ui, -apple-system, sans-serif;
    min-height: 100vh;
    padding-top: 70px;
  }

  .navbar {
    background:    linear-gradient(135deg,    #4a148c,    #7b1fa2)
!important;
    width: 100%;
    padding: 1rem;
    position: fixed;
    top: 0;
    left: 0;
    z-index: 1000;
    border-bottom: 2px solid rgba(255, 255, 255, 0.1);
  }
```

```
.navbar .navbar-container {
  width: 100%;
  max-width: 1200px;
  margin: 0 auto;
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 0 2rem;
}

.navbar-brand {
  color: #fff !important;
  font-weight: 600;
  font-size: 1.5rem;
  letter-spacing: 0.5px;
}

.signout-btn {
  background: #9c27b0;
  border: 2px solid white;
  color: white;
  padding: 0.5rem 1.5rem;
  border-radius: 25px;
  font-weight: 500;
  transition: all 0.3s ease;
}

.signout-btn:hover {
  background: #ba68c8;
  transform: translateY(-2px);
  box-shadow: 0 4px 12px rgba(156, 39, 176, 0.3);
}

.content-container {
  width: 100%;
```

```
  max-width: 800px !important;
  background: #2d2d2d;
  border-radius: 10px;
  padding: 2rem;
  margin-top: 2rem !important;
  border: 1px solid rgba(255, 255, 255, 0.1);
}

.room-card {
  background: #333;
  border: 1px solid rgba(255, 255, 255, 0.1);
  border-radius: 8px;
  padding: 1rem;
  margin-bottom: 1rem;
  transition: transform 0.2s ease;
}

.room-card:hover {
  transform: translateY(-3px);
  box-shadow: 0 4px 15px rgba(156, 39, 176, 0.2);
  border-color: rgba(255, 255, 255, 0.3);
}

/* Styling for chat messages */
.message {
  padding: 10px;
  border-radius: 10px;
  background-color: #333;
  color: #fff;
  margin-bottom: 10px;
  position: relative;
}

/* Alignment for self messages */
.self {
  background-color: #9c27b0;
```

```
    border-radius: 10px 10px 0 10px;
    margin-left: auto;
    max-width: 70%;
  }

  /* Style for other user messages */
  .message-content {
    display: inline-block;
    width: 100%;
  }

  .message-content small {
    color: #b0b0b0;
    font-size: 12px;
  }

  /* Chat box container */
  #chatMessages {
    overflow-y: auto;
      padding-right: 20px;
    max-height: 75vh;
  }

  #chatMessages::-webkit-scrollbar {
  width: 8px;
}

#chatMessages::-webkit-scrollbar-thumb {
  background-color: #9c27b0;
  border-radius: 4px;
  border: 2px solid rgba(255, 255, 255, 0.1);
}

#chatMessages::-webkit-scrollbar-track {
  background-color: #333;
  border-radius: 4px;  }
```

```
#chatBox {
  display: flex;
  flex-direction: column;
  gap: 1rem;
}

/* Adjustments for input field and send button */
  #messageInput {
    background-color: #333;
    color: #fff;
    border: 1px solid rgba(255, 255, 255, 0.3);
    width: 90%;
    padding-right: 10px;
    box-sizing: border-box;  }

#sendButton {
  background-color: #9c27b0;
  border: none;
  color: white;
  padding: 10px;
  cursor: pointer;
  transition: all 0.3s ease;
}

#sendButton:hover {
  background-color: #ba68c8;
  transform: translateY(-2px);
  box-shadow: 0 4px 10px rgba(156, 39, 176, 0.3);
}

.input-group-sm {
  display: flex;
  justify-content: space-between;
}
```

```
    </style>
{% endblock %}









{% load static %}

<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">
    <link rel="icon" href="{% static 'chat/favicon.ico' %}">
    <title>Dyploma Spam Checker</title>
    {% block head %}{% endblock %}
    <style>
body {
    background-color: #1a1a1a;
    color: #e0e0e0;
    font-family: 'Segoe UI', system-ui, -apple-system, sans-serif;
    min-height: 100vh;
    margin: 0; /* Remove default body margin */
    display: flex;
    justify-content: center; /* Center content horizontally */
    align-items: center; /* Center content vertically */
}

.navbar {
    background: linear-gradient(135deg, #4a148c, #7b1fa2)
!important;
```

```
    width: 100%;
    padding: 1rem;
    position: fixed;
    top: 0;
    left: 0;
    z-index: 1000;
    border-bottom: 2px solid rgba(255, 255, 255, 0.1);
}

.content-container {
    width: 100%;
    max-width: 800px !important;
    background: #2d2d2d;
    border-radius: 10px;
    padding: 2rem;
    margin-top: 70px; /* Adjust for navbar height */
    border: 1px solid rgba(255, 255, 255, 0.1);
    box-sizing: border-box;
    display: flex;
    flex-direction: column;
    justify-content: center;

}

/* For better control over spacing and alignment */
.navbar-container {
    width: 100%;
    max-width: 1200px;
    margin: 0 auto;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 0 2rem;
}

        .navbar-brand {
```

```
    color: #fff !important;
    font-weight: 600;
    font-size: 1.5rem;
    letter-spacing: 0.5px;
}

.signout-btn {
    background: #9c27b0;
    border: 2px solid white;
    color: white;
    padding: 0.5rem 1.5rem;
    border-radius: 25px;
    font-weight: 500;
    transition: all 0.3s ease;
}

.signout-btn:hover {
    background: #ba68c8;
    transform: translateY(-2px);
    box-shadow: 0 4px 12px rgba(156, 39, 176, 0.3);
}



.room-card {
    background: #333;
    border: 1px solid rgba(255, 255, 255, 0.1);
    border-radius: 8px;
    padding: 1rem;
    margin-bottom: 1rem;
    transition: transform 0.2s ease;
}

.room-card:hover {
    transform: translateY(-3px);
    box-shadow: 0 4px 15px rgba(156, 39, 176, 0.2);
```

```
            border-color: rgba(255, 255, 255, 0.3);
        }

        input, textarea, select {
            background-color: #333 !important;
            border: 1px solid rgba(255, 255, 255, 0.2) !important;
            color: #fff !important;
            border-radius: 6px;
            padding: 0.5rem 1rem;
        }

        input:focus, textarea:focus, select:focus {
            border-color: #9c27b0 !important;
            box-shadow: 0 0 0 2px rgba(156, 39, 176, 0.25)
!important;
            outline: none;
        }

        @keyframes fadeIn {
            from {
                opacity: 0;
                transform: translateY(10px);
            }

            to {
                opacity: 1;
                transform: translateY(0);
            }
        }

        .room-card {
            animation: fadeIn 0.3s ease forwards;
        }
    </style>
</head>
```

```
<body>
    <nav class="navbar fixed-top">
        <div class="navbar-container">
            <a    class="navbar-brand"    href="{%    url    'home'
%}">Dyploma Spam Checker</a>
            <a    class="signout-btn"    href="{%    url    'signout'
%}">SignOut</a>
        </div>
    </nav>

    <div class="content-container mt-5">
        <br>
        {% block content %}
        {% endblock %}
    </div>

    <script      src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/js/bootstrap.bundle.min.js"></script>
</body>

</html>
```

**Додаток Б** Копії публікацій

SCI-CONF.COM.UA

# PERSPECTIVES OF CONTEMPORARY SCIENCE: THEORY AND PRACTICE

## PROCEEDINGS OF XI INTERNATIONAL SCIENTIFIC AND PRACTICAL CONFERENCE DECEMBER 9-11, 2024

LVIV
2024

# СПАМ ТА ШТУЧНИЙ ІНТЕЛЕКТ

**Прокопенко Денис Петрович**
Доктор Phd по Прикладній механіці
Західноукраїнський національний університет
м. Тернопіль, Україна

**Вступ./ Introductions** спам перетворився з простої масової розсилки на кладну зброю в руках кіберзлочинців. Від простих рекламних оголошень до ішингових атак та спроб викрадення персональних даних – сучасний спам міщує в собі широкий спектр загроз. Щоб успішно протистояти цій проблемі, еобхідно розуміти, як працюють спамери та які методи вони використовують.

**Мета роботи. / Aim.** Ця робота спрямована на всебічне дослідження еномену спаму, його еволюції з часом та сучасних тенденцій. Ми прагнемо розуміти, як спамери використовують нові технології, щоб обійти системи ахисту та зробити свої повідомлення більш ефективними та як штучний нтелект зможе нам допомогти боротися із спамом.

**Матеріали та методи./Materials and methods.** Не можного говорити про пам не розуміючи зо це таке і з чим його треба їсти, хоча може і не потрібно. Спам – це справжнє лихо сучасного інтернету. Він захаращує наші поштові скриньки, відволікає від важливих справ і може стати причиною фінансових трат. Спам – це як спам у банці, тільки цифровий, і він так само неприємний. І нарешті спам – це не просто досада, це реальна загроза. Адже серед купа сміття можуть ховатися фішингові листи, які можуть призвести до втрати грошей або особистих даних. Так що спам – це не просто дрібниця, а серйозна проблема .

Але перейдимо від котлет до сучасного спаму і розглянемо деякі його види.

- реклама - деякі компанії, що займаються легальним бізнесом, рекламують
- реклама незаконної продукції - за допомогою спаму рекламують
- продукцію, про яку не можна повідомити іншими

способами-наприклад, порнографію, контрафактні товари (підробки, конфіскат), лікарські засоби з обмеженнями по обороту, незаконно отриману закриту інформацію (бази даних), контрафактне програмне забезпечення, сюди ж відноситься і реклама самих послуг розсилки спаму ;

- антиреклама - заборонена законодавством про рекламу інформація - наприклад, ганьбити конкурентів і їхню продукцію, - також може поширюватися за допомогою спаму;
- «нігерійські листи» - це вид фішингового шахрайства, яке полягає в надсиланні електронних листів із проханням допомогти у вирішенні фінансової проблеми, пов'язаної з великою сумою грошей; відправники обіцяють значну винагороду в обмін на допомогу, але зазвичай вимагають від одержувача попередньої оплати різних витрат або надання особистої інформації. Метою таких листів є обманом змусити жертву передати гроші або дані, які шахраї можуть використати для фінансової вигоди [3];
- фішинг - це вид кіберзлочинності, що передбачає обман користувачів з метою отримання їхньої конфіденційної інформації, такої як паролі, номери кредитних карток або інші особисті дані. Зловмисники зазвичай маскуються під легітимні установи або відомі сервіси, надсилаючи електронні листи, повідомлення чи створюючи фальшиві вебсайти, які виглядають справжніми; основною метою фішингу є змусити жертву надати свою особисту інформацію, яку потім можна використати для фінансового або особистого зиску, такого як крадіжка особистості або злом облікових записів.

Крім цього є розповсюдженими DoS і DDoS-атаки, масове розсилання від імені іншої особи, для того щоб викликати до нього негативне ставлення чи масове розсилання листів, що містять комп'ютерні віруси (для їх початкового поширення). Розсилка листів з приголомшливою історією, що розповідає про те, нібито кожен інтернет-провайдер виплатить певну суму грошей на лікування родині потерпілого за кожне переслання листа. Головною метою цієї розсилки є збір e-mail адрес, оскільки після багатьох пересилань всім знайомим в тексті таких листів часто можна знайти e-mail адреси всіх, кому вони були

переслані раніше. І серед наступних одержувачів може бути навіть той, хто ініціював цей спам.

Усі сучасні методики боротьби зі спамом можна умовно розділити на певні категорії.

Методи, засновані на аналізі листа — їхнє завдання полягає у вивченні листа з метою його класифікації. Подібний метод вирішує дві задачі: виявляє спам та ідентифікує листи, які гарантовано не є спамом. Відомо кілька найбільш поширених методів аналізу:

- за формальними ознаками;
- за вмістом із використанням сигнатурного аналізу. Цей метод базується на пошуку в тексті листа певних сигнатур, описаних воновлюваній базі даних;
- за вмістом із застосуванням статистичних методик. Більшість сучасних методів цього класу засновані на теоремі Байєса;
- за вмістом із використанням SURBL (Spam URL Realtime Block Lists — список блокування спамерських URL). Ідея методу полягає в пошуку розміщених у тілі листа посилань і їх перевірці за базою SURBL. Цей метод ефективний проти спаму, в якому для обходу фільтрів замість реклами застосовується посилання на сайт із рекламою;
- детектори масової розсилки — як випливає з назви, їхнім завданням є виявлення розсилки схожого листа великій кількості абонентів;
- методи, засновані на визнанні відправника листа як спамера — вони спираються на різні чорні списки IP- та поштових адрес. Як і в попередньому випадку, можливе вирішення зворотного завдання-розпізнавання надійних користувачів або поштових серверів, від яких необхідно приймати пошту в будь-якому разі. Перевагою цього методу є те, що для розпізнавання відправника як спамера поштовому серверу не потрібно приймати лист, що суттєво економить трафік. Розпізнавання відправника здійснюється за його IP-адресою;
- методи, засновані на верифікації зворотної адреси відправника та

його домену, — найпростішим методом захисту є звичайний DNS-запит за ім'ям домену відправника листа. Якщо з'ясовується, що домен відправника не існує, то, ймовірно, адреса відправника є підробленою. Проте цей метод малоефективний, оскільки спамери можуть використовувати реальні адреси як адреси відправника, випадково вибрані з бази розсилки. Більш складна технологія передбачає перевірку, чи дозволена відправка листа з вказаною зворотною адресою з даної IP-адреси .

Інша методика полягає в тому, що після отримання підозрілого листа антиспам-фільтр може спробувати зв’язатися з відправником листа і запропонувати йому тим чи іншим способом довести, що він не є спамером. Зазвичай перевірка зводиться до відвідування певної сторінки та заповнення на ній веб-форми. Подібну методику, зокрема, практикує фільтр спаму на google.com та ukr.net.

Антиспам-фільтри є важливими інструментами для захисту електронної пошти та інших комунікаційних систем від небажаних повідомлень, але зловмисленники використовують спроби його обходу. Один із таких способів є модифікації та спотворення тексту

На початкових етапах розвитку спаму розсилання проводилося без модифікації тексту повідомлення. Це призвело до появи перших фільтрів, причому елементарний з них міг просто розрахувати контрольну суму повідомлення й порівняти його з базою. Відповідною відповіддю спамерів було розроблення наступних методів автоматичної модифікації тексту для ускладнення його ідентифікації як спаму:

- персоналізація повідомлень — це найпростіший метод маскування, що полягає в невеликій модифікації повідомлення з використанням доступних спамеру даних, зокрема e-mail-адреси. Наприклад, замість статичного тексту: «Шановний вебмастер! Спеціалісти нашої компанії вивчили ваш сайт і прийшли до висновку, що наша фірма може удосконалити його дизайн…» - легко можна отримати динамічний: «Шановний Іванов! Спеціалісти нашої компанії вивчили ваш сайт www.rogaikoputa.com і прийшли

до висновку, що наша фірма може удосконалити його дизайн…» Зрозуміло, що ці дані легко можна отримати з поштової адреси rogaikoputa @ukr.net;

- використання пробілів та інших роздільників у словах — наприклад замість «Купуйте супертовар» — «К У П У И ТЕ супер-товар»;

- транслітерація — наприклад виділені напівжирними літерами фрази «Купуйте супертовар» можуть бути замінені ідентичними за начертанням латинськими літерами, при цьому видимої різниці читаючий текст користувач не виявить. Можливі більш помітні на око модифікації, що не впливають на розуміння тексту, — наприклад «][акер», «}{акер» або «4ерний» — у цьому прикладі літери заміняються цифрами або схожими за начертанням наборами символів. Цей прийом ефективний проти байєсовських фільтрів, і для боротьби з ним антиспамова система повинна виконувати нормалізацію тексту та детекцію подібних спотворень;

- перефразування — цей метод полягає в тому, що під час розробки спам-листа можна створити десятки подібних за змістом фраз і потім автоматично зібрати з них лист. Наприклад, фрази "Купіть супертовар" і "Придбайте нашу відмінну продукцію" схожі за змістом, але абсолютно відмінні для фільтра Байєса або сигнатурного пошуку. Крім того, можливе застосування словника синонімів для випадкової заміни слів на близькі за змістом еквіваленти;

- додавання чужорідного тексту — цей метод полягає в тому, що окрім реклами, до листа вставляються декілька пропозицій, що не мають ніякого відношення до теми листа, наприклад уривок з будь-якого літературного твору. Цей прийом ускладнює налаштування і погіршує якість роботи фільтра Байєса, але не впливає на якість роботи сигнатурного аналізатора.

Ще одина з спроб обійти системи антиспаму є використання можливостей HTML. Спамери давно взяли HTML на озброєння, оскільки він дозволяє реалізувати кілька простих і ефективних методів маскування.

**Результати та обговорення./Results and discussion.** В той же час штучний інтелект (ШІ) дозволяє аналізувати великі обсяги даних, виявляти нові

шаблони та тенденції в спамі. Це допомагає розробляти більш ефективні алгоритми для його виявлення та блокування. Незважаючи на значний прогрес, боротьба зі спамом залишається актуальною проблемою. Постійно з'являються нові види спаму, а спамери використовують все більш складні методи. Подальший розвиток ШІ дозволить створювати більш розумні та адаптивні системи захисту від спаму.

**Висновки./Conclusions.** Хочеми ми цього чи ні , але ШІ стане нашим незамінним помічником у боротьбі з небажаною поштою. Він постійно навчається, адаптується до нових хитрощів спамерів та проактивно захищає нашу поштову скриньку. Завдяки здатності аналізувати величезні обсяги даних, штучний інтелект може виявляти навіть найскладніші схеми спаму, які не під силу розпізнати людині. Крім того, він постійно вдосконалюється, зменшуючи кількість помилкових спрацювань і забезпечуючи нам більш точну фільтрацію повідомлень. Таким чином, ми можемо бути впевнені, що наша пошта завжди буде чистою і безпечною, а ми зможемо зосередитися на важливих справах.

ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

# ВСЕУКРАЇНСЬКА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ СТУДЕНТІВ, АСПІРАНТІВ ТА МОЛОДИХ ВЧЕНИХ «ІНТЕЛЕКТУАЛЬНІ КОМП'ЮТЕРНІ СИСТЕМИ ТА МЕРЕЖІ»

**5 ЛИСТОПАДА 2024**

KI.WUNU.EDU.UA/CONFERENCE/

ТЕРНОПІЛЬ

**2024**

Прокопенко Д.П.
магістр студент гр. КБзм -11 ЗУНУ
Науковий керівник завідувач кафедри кібербезпеки, доктор технічних наук, професор
Західноукраїнського національного університету Яцків Василь Васильович

# МЕТОДИ МАШИННОГО НАВЧАННЯ ДЛЯ ПРОТИДІЇ СПАМУ

**Вступ.** спам перетворився з простої масової розсилки на складну зброю в руках кіберзлочинців. Від простих рекламних оголошень до фішингових атак та спроб викрадення персональних даних – сучасний спам вміщує в собі широкий спектр загроз. Щоб успішно протистояти цій проблемі, необхідно розуміти, як працюють спамери та які методи вони використовують.

**Постановка задачи.** Наша задача – розглянути моделі машинного навчання, які зможуть точно класифікувати електронні листи на спам та неспам. Для цього ми будемо використовувати різноманітні методи обробки тексту, такі як аналіз слів, фраз і навіть емоджі. Модель має навчитися виявляти характерні ознаки спаму, наприклад, наявність шаблонних фраз, підозрілих посилань або незвичайних відправників. У результаті ми отримаємо надійний інструмент для боротьби з небажаною поштою

**Основний матеріал.** В сучасному світі ми не обідмемося від того що нам пропанує ЩІ (штучний інтелект). Розглянемо деякі з них.

Наївний Байєсівський класифікатор (Naive Bayes classifier) - це простий ймовірнісний класифікатор, який базується на застосуванні теореми Байєса з "наївним" припущенням про незалежність між ознаками (у вигляді умовної незалежності) для спрощення обчислень. Цей метод широко використовується для класифікації тексту, фільтрації спаму, в рекомендаційних системах та інших задачах машинного навчання. Він є швидким у навчанні та працює добре з великими обсягами даних.

Метод опорних векторів (Support Vector Machine, SVM) забезпечує високу точність класифікації, особливо коли дані добре підготовлені; завдяки регуляризації, SVM може ефективно уникати перевитрати, навіть у випадках з великою кількістю ознак; SVM добре справляється з високовимірними текстовими даними, такими як електронні листи.

Нейронні мережі є потужним інструментом для виявлення спаму, здатним забезпечити високу точність і адаптивність до нових викликів у боротьбі зі спамом бо можуть виявляти складні патерни у тексті, що забезпечує високу точність виявлення спаму, а також можуть адаптуватися до нових типів спаму за допомогою додаткового навчання. Проте навчання великих нейронних мереж може бути ресурсомістким і вимагати значних обчислювальних потужностей і для ефективного навчання моделей потребує великих обсягів даних, які можуть бути важкими для збирання та анотації.

Дерева прийняття рішень (DT) - це один з методів прогнозування, що використовуються в статистиці, інтелектуальному аналізі даних і машинному навчанні. Дерева прийняття рішень є потужним інструментом для аналізу даних, але вони повинні використовуватися з обережністю та часто в поєднанні з іншими методами, такими як ансамблеві методи (наприклад, Random Forest або Gradient Boosting), для покращення стабільності та продуктивності.

Випадковий ліс (random forest) – алгоритм машинного навчання, запропонований Лео Брейманом і Адель Катлер, що полягає у використанні ансамблю дереврішень [1]. Алгоритм поєднує в собі дві основні ідеї: метод баггінга (bagging – від bootstrap aggregation, тобто дерева навчаються по випадковим підвибіркам) і метод випадкових підпросторів. Алгоритм застосовується для задач класифікації, регресії і кластеризації.

Випадкові ліси або ліси випадкових рішень – це метод ансамблевого навчання для класифікації, регресії та інших завдань, який працює шляхом побудови безлічі дерев рішень під час навчання. Для класифікаційних завдань результатом випадкового лісу є клас, обраний більшістю дерев. Для завдань регресії повертається середнє або середнє прогнозування окремих дерев [2]. Для експерименту було застосовано три алгоритми, хоча розроблена система дає

можливість проекспериментувати з любою кількістю алгоритмів. В таблиці 1 ми звели три машинних алгоритми.

Таблиця 1 - Огляд ключових характеристик алгоритмів машинного навчання

| Характеристика | SVM (Опорні вектори) | Random Forest (Випадковий ліс) | Bayes (Наївний Байєс) |
|---|---|---|---|
| Принцип роботи | Будує гіперплощину, що оптимально розділяє дані. | Ансамбль дерев рішень, кожне з яких будується на випадковій вибірці даних. | Обчислює ймовірність класу на основі теореми Байєса. |
| Переваги | Добре працює з високовимірними даними. | Висока точність, стійкий до перенавчання, обробляє різні типи даних. | Простий, ефективний для великих наборів даних, обробляє пропущені дані. |
| Недоліки | Чутливий до вибору ядра та параметрів, складний для інтерпретації. | Може бути обчислювально дорогим для великих наборів даних, складний для інтерпретації окремих дерев. | Припущення про незалежність ознак може бути неточним. |
| Типові застосування | Класифікація, регресія, виявлення аномалій. | Класифікація, регресія, ранжування. | Класифікація текстів, фільтрація спаму, рекомендаційні системи. |
| Стійкість до шуму | Середня | Висока | Середня |
| Інтерпретація | Складна | Складна для окремих дерев, але можна оцінити важливість ознак. | Проста |
| Швидкість роботи | Середня, може бути повільною для великих наборів даних, особливо при використанні нелінійних ядер. | Залежить від кількості дерев, але зазвичай швидка для тренування і прогнозування. | Дуже швидка, особливо для великих наборів даних. |

Після проведених експериментів, за допомогою мови програмування RYTHON ,було встановлено що найкраще працює модель SVM, яка найчастіше ідентифікувала спам, найгірше себе показала система, навчена на Bayes. Таким чином, можна скласти таблицю 2.

Результати мого дослідження також підтверджують, що скрипт тестування, який перевірив правильність досліджених spam/ham повідомлень вивів такі результати:

Знайдено 2125 спаму, 20 пропущено, 10 пропущено ham, 1134 знайдено. У відношенні до таблиці 2. похибка склала всього 1.14%, що є некритичним показником для нейронних мереж.

Результатом роботи стало те, що система дійсно робоча і може навчатись на кастомних CSV, які можна генерувати з власних повідомлень або брати готові вибірки в мережі інтернет.

**Висновок.** Штучний інтелект, зокрема машинне навчання, стає нашим найкращим союзником у боротьбі зі спамом. Завдяки йому ми можемо розробляти все більш досконалі фільтри, які здатні розпізнавати навіть найхитріші спам-повідомлення. Уявіть собі, що комп'ютери зможуть аналізувати мільйони листів щосекунди і виявляти найменші ознаки спаму. Це дозволить нам позбутися цієї надокучливої проблеми і зосередитися на дійсно важливих справах

Таблиця 2 - Оцінка спрацювань

| | Тестування моделі SVM | Тестування моделі RANDOW_FOREST | Тестування моделі NAÏVE-BAYES |
|---|---|---|---|
| True Negative (TN): | 2125 | 2129 | 2129 |
| Помилковий результат (FP) | 20 | 16 | 16 |
| Помилково негативний FN): | 10 | 11 | 825 |
| True Positive (TP): | 1134 | 1133 | 319 |
| Похібка | 1,134 | 1,03 | 2,58 |

**Список літератури**

1. Donges, N., 2018. The Random Forest Algorithm. Towards Data Science.

2 Пиж О. І. Інтелектуальна система аналізу та класифікації вхідних e-mail повідомлень на основі їх вмісту: курс. магістр: 121- «Інженерія програмного забезпечення». Київ, 2022. 105 с.