

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»  
Навчально-науковий інститут прикладного системного аналізу  
Кафедра штучного інтелекту**

До захисту допущено:

В. о. завідувачки кафедри

\_\_\_\_\_Ірина ДЖИГИРЕЙ

«\_\_\_»\_\_\_\_\_2025 р.

**Дипломна робота  
на здобуття ступеня бакалавра  
за освітньо-професійною програмою  
«Системи і методи штучного інтелекту» спеціальності  
122 «Комп'ютерні науки»  
на тему: «Розробка інформаційної системи класифікації текстових  
відгуків із використанням методів машинного навчання»**

Виконав:

студент IV курсу, групи КІ-23

Ленишин Максим Володимирович

\_\_\_\_\_

Керівник:

асистент кафедри штучного інтелекту, д.ф.,

Гаврилович Марія Павлівна

\_\_\_\_\_

Консультант з нормоконтролю:

фахівець першої категорії кафедри штучного інтелекту,

к.т.н., доцент, Комариста Богдана Миколаївна

\_\_\_\_\_

Рецензент:

старший викладач кафедри ММСА, д.ф.,

Левенчук Людмила Борисівна

\_\_\_\_\_

Засвідчую, що у цій дипломній  
роботі немає запозичень з праць  
інших авторів без відповідних  
посилань.

Студент \_\_\_\_\_

Київ – 2026 року

**Національний технічний університет України**  
**«Київський політехнічний інститут імені Ігоря Сікорського»**  
**Навчально-науковий інститут прикладного системного аналізу**  
**Кафедра штучного інтелекту**

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ

В. о. завідувачки кафедри

\_\_\_\_\_Ірина ДЖИГИРЕЙ

«15» січня 2026 р.

**ЗАВДАННЯ**  
**на дипломну роботу студенту**  
**Ленишину Максиму Володимировичу**

1. Тема роботи «Розробка інформаційної системи класифікації текстових відгуків із використанням методів машинного навчання», керівник роботи Гаврилович Марія Павлівна, асистент, затверджені наказом по університету від від «27» травня 2026 р. № 1944-с.
2. Термін подання студентом роботи «05» червня 2026 року.
3. Вихідні дані до роботи: набір даних (датасет) відгуків користувачів про роботу онлайн-магазинів (Ukrainian text of user feedbacks about online shop).
4. Зміст роботи: розробка інформаційної системи класифікації текстових відгуків, що базується на методах машинного навчання та архітектурі ансамблю класифікаторів. Виконано попередню обробку текстових масивів українською мовою (токенізація, лематизація, очищення від шуму), векторизацію даних, проєктування та навчання ансамблевої моделі для класифікації відгуків за категоріями, а також оптимізацію порогів прийняття рішень для досягнення високих показників точності. Проведено функціонально-вартісний аналіз розробленого рішення.
5. Перелік ілюстративного матеріалу: презентація

6. Дата видачі завдання «02» лютого 2026 року.

Календарний план

| № з/п | Назва етапів виконання дипломної роботи                              | Термін виконання етапів роботи | Примітка |
|-------|----------------------------------------------------------------------|--------------------------------|----------|
| 1     | Огляд літератури за темою та збір інформації для формування датасету | 21.04.2026                     | Виконано |
| 2     | Підготовка першого розділу дипломної роботи                          | 28.04.2026                     | Виконано |
| 3     | Підготовка другого розділу дипломної роботи                          | 12.05.2026                     | Виконано |
| 4     | Підготовка третього розділу дипломної роботи                         | 26.05.2026                     | Виконано |
| 5     | Оформлення дипломної роботи                                          | 02.06.2026                     | Виконано |
| 6     | Підготовка презентації доповіді                                      | 09.06.2026                     | Виконано |

Студент

Максим ЛЕНИШИН

Керівник

Марія ГАВРИЛОВИЧ

## РЕФЕРАТ

Дипломна робота: 55 с., 17 рис., 12 табл., 25 посилань, додатки.

МАШИННЕ НАВЧАННЯ, КЛАСИФІКАЦІЯ ТЕКСТІВ, АНАЛІЗ ВІДГУКІВ, NLP, АНСАМБЛЕВЕ НАВЧАННЯ, ІНФОРМАЦІЙНА СИСТЕМА.

Об'єктом дослідження є процес автоматичної класифікації неструктурованих текстових відгуків користувачів.

Предметом дослідження є методи та алгоритми машинного навчання для семантичного аналізу, векторизації та класифікації текстових даних, що забезпечують високу точність обробки відгуків.

Метою роботи є розробка та впровадження інформаційної системи класифікації відгуків, здатної ефективно сортувати текстові дані за категоріями з використанням ансамблевого навчання.

У ході роботи здійснено підготовку та очищення датасету текстових відгуків, що включає етапи токенізації, лематизації та перетворення текстів у векторний формат. Для підвищення точності класифікації спроектовано та навчено ансамблеву модель, що дозволяє мінімізувати помилки при розпізнаванні різних типів відгуків. Оцінка за метриками Precision, Recall та F1-score підтвердила високу стійкість та ефективність системи. Реалізований програмний прототип має інтерфейс користувача та може бути інтегрований у бізнес-системи для аналізу зворотного зв'язку від клієнтів. Проведено функціонально-вартісний аналіз, який підтвердив техніко-економічну доцільність розробки та її конкурентоспроможність на ринку програмних продуктів.

## ABSTRACT

Bachelor's thesis: 55 p., 17 figures, 12 tables, 25 references, appendix.

MACHINE LEARNING, TEXT CLASSIFICATION, FEEDBACK ANALYSIS, NLP, ENSEMBLE LEARNING, INFORMATION SYSTEM.

The object of the study is the process of automated classification of unstructured user text feedback.

The subject of the study is machine learning methods and algorithms for semantic analysis, vectorization, and classification of text data that ensure high accuracy in processing feedback.

The goal of the work is the development and implementation of an information system for feedback classification capable of efficiently sorting text data by categories using ensemble learning.

In the course of the work, a dataset of text feedback was prepared and cleaned, including stages of tokenization, lemmatization, and conversion of texts into vector format. To improve classification accuracy, an ensemble model was designed and trained, which allows for minimizing errors in identifying different types of feedback. Evaluation using Precision, Recall, and F1-score metrics confirmed the high robustness and effectiveness of the system. The implemented software prototype features a user interface and can be integrated into business systems for analyzing customer feedback. A functional-cost analysis was conducted, confirming the technical and economic feasibility of the development and its competitiveness in the software market.

## ЗМІСТ

|                                                                     |    |
|---------------------------------------------------------------------|----|
| ВСТУП.....                                                          | 8  |
| РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ             |    |
| КЛАСИФІКАЦІЇ ТЕКСТОВИХ ВІДГУКІВ.....                                | 9  |
| 1.1 Актуальність автоматичної класифікації текстів.....             | 9  |
| 1.2 Обробка природної мови як міждисциплінарна галузь.....          | 10 |
| 1.3 Особливості та лінгвістична складність відгуків.....            | 15 |
| 1.3.1 Неструктурованість та інформаційний шум.....                  | 15 |
| 1.3.2 Морфологічна складність та флективність української мови..... | 16 |
| 1.3.3 Неформальна лексика та друкарські помилки.....                | 16 |
| 1.3.4 Контекстуальні залежності та сарказм.....                     | 16 |
| 1.4 Постановка задачі.....                                          | 17 |
| 1.4.1 Призначення розробки.....                                     | 17 |
| 1.4.2 Мета та задачі розробки.....                                  | 17 |
| Висновки до розділу 1.....                                          | 18 |
| РОЗДІЛ 2 МЕТОДИ ТА МОДЕЛІ МАШИННОГО НАВЧАННЯ ДЛЯ                    |    |
| КЛАСИФІКАЦІЇ ТЕКСТОВИХ ВІДГУКІВ.....                                | 19 |
| 2.1 Методологія інтелектуальної обробки тексту.....                 | 19 |
| 2.1.1 Концепції підготовки неструктурованих даних.....              | 19 |
| 2.1.2 Токенізація та нормалізація послідовностей.....               | 21 |
| 2.1.3 Очищення даних від інформаційного шуму.....                   | 22 |
| 2.1.4 Морфологічний аналіз та лематизація.....                      | 24 |
| 2.2 Методи обробки природної мови для задач класифікації.....       | 25 |
| 2.2.1 Формалізація ознак у задачах навчання.....                    | 25 |
| 2.2.2 Механізм та переваги методу TF-IDF.....                       | 26 |
| 2.2.3 Врахування контексту через N-грами.....                       | 26 |
| 2.3 Алгоритми класифікації текстових даних.....                     | 27 |
| 2.3.1 Принципи роботи методу опорних векторів (SVM).....            | 28 |
| 2.3.2 Логістична регресія: математичні основи.....                  | 30 |

|                                                          |           |
|----------------------------------------------------------|-----------|
| 2.3.3 Використання ансамблевих методів.....              | 32        |
| 2.3.4 Порівняльний аналіз класифікаційних моделей.....   | 33        |
| 2.4 Оцінка якості інтелектуальних систем.....            | 34        |
| 2.4.1 Матриці помилок.....                               | 35        |
| 2.4.2 Метрики якості кваліфікації.....                   | 37        |
| 2.4.3 Стратегії валідації та тестування.....             | 38        |
| Висновки до розділу 2.....                               | 39        |
| <b>РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНІ</b> |           |
| <b>ДОСЛІДЖЕННЯ.....</b>                                  | <b>40</b> |
| 3.1 Обґрунтування програмного середовища.....            | 40        |
| 3.2 Проектування модуля препроцесингу.....               | 41        |
| 3.3 Основні етапи реалізації.....                        | 43        |
| 3.4 Розробка інтелектуального ядра системи.....          | 45        |
| 3.5 Вибір та обґрунтування гіперпараметрів моделі.....   | 47        |
| 3.6 Графічний інтерфейс користувача.....                 | 49        |
| Висновки до розділу 3.....                               | 53        |
| ВИСНОВКИ.....                                            | 54        |
| ПЕРЕЛІК ПОСИЛАНЬ.....                                    | 55        |
| ДОДАТОК А ЛІСТИНГ ПРОГРАМИ.....                          | 57        |

## ВСТУП

У сучасному світі зростаючі обсяги даних, особливо в секторі державного управління, становлять серйозний виклик для їх ефективної обробки. За статистику, звичайний офісний працівник відправляє 40 листів і отримує 121 лист в день (близько 9000 - 15000 листів на рік) [1]. І на обробку такого об'єму листів витрачається в середньому близько 16% робочого дня, тож 55% працівників визнають, що електронна пошта заважає їм виконувати основні обов'язки<sup>1</sup>.

Особливо від цього страждають державні структури, оскільки відсутність автоматизованої системи класифікації повідомлень призводить до того, що листи, надіслані на загальну електронну адресу державної структури, не перенаправляються до відповідних відділів (наприклад, соціальних послуг, охорони здоров'я, адміністрації тощо), що сповільнює реакцію на запити громадян та знижує якість послуг.

Саме тому автоматизація класифікації текстів на основі сучасних методів NLP та машинного навчання стає критичною для оптимізації роботи державних установ, соціальних служб і навіть комерційних компаній, які працюють із великими масивами текстових даних.

Тому в своїй дипломній роботі я обрала тему системи класифікації текстів методами обробки природної мови та машинного навчання, в рамках якої проведу дослідження і запропоную власне покращення на основі генеративно-змагальних нейронних мереж.

---

<sup>1</sup>Тут і нижче використано такий інструмент штучного інтелекту як чат-бот з генеративним штучним інтелектом ChatGPT виключно для корегування та редагування тексту, створеного автором цієї дипломної роботи, на основі автоматизованої перевірки граматики, структури та стилю, що відповідає Політиці використання штучного інтелекту для академічної діяльності в КПІ ім. Ігоря Сікорського (протокол №11 Вченої ради КПІ ім. Ігоря Сікорського від 11 грудня 2023 р.).

## **РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ КЛАСИФІКАЦІЇ ТЕКСТОВИХ ВІДГУКІВ**

### **1.1 Актуальність автоматичної класифікації текстів**

У сучасному цифровому просторі спостерігається дуже стрімке та експоненціальне зростання обсягів неструктурованої текстової інформації. Важливим джерело таких даних - це відгуки користувачів, які акумулюються на різноманітних маркетплейсах, спеціалізованих вебплатформах, у соцмережах та системах електронної комерції. Текстовий фідбек клієнтів містить пряму й об'єктивну оцінку якості товарів, рівня сервісу, швидкості логістичних процесів та загального функціонування підприємств.

Ефективний аналіз цього інформаційного масиву дозволяє компаніям оперативно реагувати на критичні зауваження, оптимізувати внутрішні бізнес-процеси, покращувати користувацький досвід та запобігати відтоку клієнтів. Проте традиційне ручне сортування, моніторинг та обробка тисяч повідомлень, що надходять щодня, вимагають колосальних часових, людських і фінансових ресурсів. Крім того, ручний аналіз піддається суттєвому впливу суб'єктивного фактора, що неминуче призводить до помилок в інтерпретації змісту та емоційного забарвлення.

У зв'язку з цим виникає велика науково-практична потреба в автоматизації процесів інтелектуального аналізу тексту (Text Mining), зокрема автоматичної класифікації відгуків за їхньою тональністю та тематичними категоріями. Розробка та впровадження спеціалізованих інформаційних систем, що функціонують на базі сучасних методів машинного навчання, дозволяє розв'язати цю задачу з високою точністю в режимі реального часу. Автоматизація класифікації забезпечує високу масштабованість, автономність та економічну доцільність обробки великих даних, роблячи аналітичний процес незалежним від людського ресурсу.

## 1.2 Обробка природної мови як міждисциплінарна галузь

Обробка природної мови (англ. Natural language processing, NLP) – загальний напрям інформатики, штучного інтелекту та математичної лінгвістики. Він вивчає проблеми розпізнавання мовлення, класифікації тексту, розуміння природної мови, генерування природної мови. [1] Жодна з цих дисциплін самотійно не здатна повністю забезпечити інструментарій для того, щоб комп'ютерні системи могли повноцінно інтерпретувати, маніпулювати і розуміти людську мову. Програмування в межах сучасних NLP-систем вимагає інтеграції лінгвістичних правил із потужними статистичними та обчислювальними архітектурами з метою розшифрування структури мови та виявлення значущої семантичної інформації з неструктурованого тексту.

Автоматичний аналіз, інтерпретація та класифікація текстових відгуків належать до науково-практичної галузі обробки природної мови, яка є одним із ключових та найбільш динамічних напрямів сучасного штучного інтелекту. За своєю внутрішньою структурою обробка природної мови – це складна міждисциплінарна галузь знань, що виникла й розвивається на стику комп'ютерних наук, математики, теоретичної та прикладної лінгвістики, а також системного аналізу та кібернетики. Жодна з цих дисциплін самотійно не здатна повністю забезпечити інструментарій для того, щоб комп'ютерні системи могли повноцінно інтерпретувати, маніпулювати і розуміти людську мову. Програмування в межах сучасних NLP-систем вимагає інтеграції лінгвістичних правил із потужними статистичними та обчислювальними архітектурами з метою розшифрування структури мови та виявлення значущої семантичної інформації з неструктурованого тексту.

Для глибокого розуміння природи цієї дисципліни необхідно розглянути її історичний генезис та еволюційну зміну парадигм обробки мовної інформації. Початкові етапи становлення галузі, що припали на середину XX століття, базувалися виключно на символічному підході (*Symbolic NLP*) та жорстких

лінгвістичних правилах, сформульованих експертами вручну. Науковці намагалися закодувати всю багатогранність людської мови через системи детермінованих логічних конструкцій виду «якщо-тоді». Однак жива мова користувачів глобальних мереж виявилася занадто динамічною, флективною та контекстуально залежною, що призвело до так званої «зими штучного інтелекту» для ранніх лінгвістичних систем [2].

Парадигмальний зсув відбувся наприкінці 1980-х років із впровадженням статистичних методів обробки та концепцій машинного навчання. Замість спроб навчити комп'ютер мислити категоріями академічних підручників з граматики, інженери почали використовувати ймовірнісні моделі, що аналізували великі масиви реальних текстів (корпуси даних). На сучасному етапі розвитку обробка природної мови досягла найвищого рівня інтеграції, об'єднавши класичні мовознавчі теорії з методами обчислювальної лінійного програмування, багатовимірної математичної статистики та глибоких нейромережевих архітектур.

Диверсифікація практичних напрямів та ключових завдань, що постають перед дослідником у межах комп'ютерного моделювання лінгвістичних систем, наочно демонструє складну внутрішню структуру цієї технології. Головні вектори досліджень та декомпозиція загального процесу аналізу мови на ізольовані аналітичні компоненти представлені на рисунку 1.1.

Детальний аналіз структури архітектури сучасних систем класифікації дозволяє чітко диференціювати функції кожного фундаментального наукового напрямку у розв'язанні цих завдань.

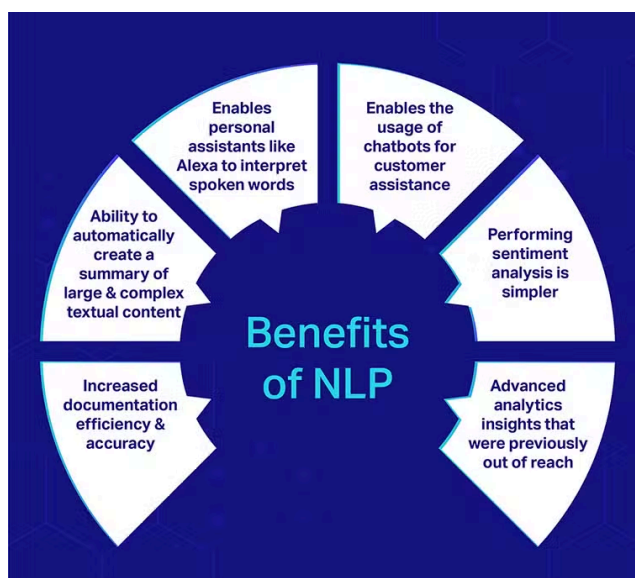


**Рисунок 1.1** – Базові прикладні завдання обробки природної мови<sup>2</sup>

Теоретична та прикладна лінгвістика. Ця складова виступає первинним базисом для будь-якої системи автоматичного аналізу тексту. Вона структурує хаотичний потік людських символів, розділяючи процес дослідження на кілька послідовних рівнів. Як продемонстровано на рис. 1.1, лінгвістичне ядро базується на таких критичних завданнях, як маркування частин мови (Speech Tagging), що є фундаментом для початкової сегментації відгуку, а також розпізнавання іменованих сутностей (Named Entity Recognition), необхідного для автоматичного виділення брендів, локацій чи назв продуктів із тексту фідбеку. Семантичний аналіз інтерпретує безпосередній зміст висловлювання, що дозволяє обчислювальній системі вирішувати завдання розрізнення значень слів у неоднозначному контексті (Word Sense Disambiguation) та коректно визначати зв'язки між займенниками й іменниками (Co-reference Resolution). Особливо важливою лінгвістична теорія є для локалізації систем під флексивні мови, де розвинена морфологія вимагає складних алгоритмів деконструювання слів для виділення кореневих основ та відсікання граматичних закінчень.

<sup>2</sup>Джерело зображення URL: [https://miro.medium.com/v2/resize:fit:720/format:webp/0\\*Dajl2L9lxcHv93N.jpg](https://miro.medium.com/v2/resize:fit:720/format:webp/0*Dajl2L9lxcHv93N.jpg)

Математична статистика та теорія інформації. Жоден процесорний пристрій чи програмний код не здатний оперувати літерами чи реченнями безпосередньо. Математичний апарат є тим містком, який переводить дискретні текстові дані у неперервний числовий простір. Він дозволяє формалізувати результати лінгвістичного аналізу, представлені на схемі задач (рис. 1.1), та конвертувати їх у числові структури. За допомогою методів лінійної алгебри та матричного числення здійснюється векторизація текстів. Кожне слово або документ відображається у вигляді багатовимірного вектора ознак у розрідженому просторі, де мірою семантичної близькості між відгуками виступає косинусна відстань або евклідов метричний простір. Статистичні методи також відповідають за розрахунок інформаційної місткості окремих термінів, згладжування частотних розподілів та зниження розмірності матриць з метою усунення надлишкового шуму [3].

Машинне навчання (Machine Learning). Цей складник є безпосереднім аналітичним ядром проекрованої інформаційної системи. Він бере за основу вихідні дані таких процесів, як розпізнавання сутностей чи лексичний аналіз, та використовує їх для прийняття рішень. Замість написання статичних інструкцій для тисяч потенційних варіантів користувацьких скарг, впроваджується методологія навчання за прецедентами (Supervised Learning). Використовуючи векторизовані математичні представлення розмічених експертами текстових корпусів, алгоритми машинного навчання самостійно знаходять приховані закономірності, підбирають вагові коефіцієнти для ознак та вибудовують оптимальні розділяючі гіперплощини між класами. Це забезпечує системі здатність до узагальнення: зустрівши абсолютно новий, раніше невідомий відгук, навчена модель із високою точністю здатна передбачити його тональність чи категорію. [4, 9]

Системний аналіз та програмна інженерія. Будь-який математичний алгоритм чи лінгвістичне правило залишаються теоретичними концептами, доки інженерія програмного забезпечення не об'єднає їх у стійкий, надійний та оптимізований програмний комплекс. Програмна інженерія дозволяє

інтегрувати складні процеси, починаючи від класичного аналізу символів і закінчуючи генерацією природної мови (Natural Language Generation) чи інтелектуальним розпізнаванням мовлення (Speech Recognition), якщо система передбачає голосовий ввід. Програмна інженерія відповідає за створення наскрізних конвеєрів обробки даних (Data Pipelines). У межах цього процесу сирий текст відгуку, що надходить із зовнішнього джерела, поетапно та безперервно проходить через модулі лінгвістичного очищення, математичної векторизації та класифікаційного передбачення. Також цей напрям реалізує архітектуру взаємодії програми з базами даних для довготривого збереження аналітики та проектування графічних інтерфейсів користувача (UI/UX), що робить систему доступною для кінцевих бізнес-аналітиків.[8]

Як демонструє системна архітектура, спроба побудувати аналітичний інструмент виключно на базі однієї з дисциплін приречена на неефективність. Суто лінгвістичні системи виявляються занадто крихкими при зіткненні з неформальним Інтернет-сленгом, оддруками та технологічним шумом вебсторінок. У той самий час, ізольовані математичні чи нейромережеві моделі, які ігнорують граматичні закони та морфологічні особливості мови, потребують астрономічних обсягів обчислювальних ресурсів і даних для навчання через штучне роздування ознакового простору словоформами.

Комплексний огляд NLP-задач (рис. 1.1) підтверджує, що сучасний інтелектуальний аналіз є класичним втіленням синергетичного міждисциплінарного підходу. Лише системний інженерний синтез комп'ютерного мовознавства, математичного моделювання розріджених матриць, статистичних методів машинного навчання та сучасних принципів проектування програмного забезпечення дозволяє створювати автономні, висоефективні інформаційні системи, здатні розв'язувати актуальні задачі інтелектуального аналізу даних у сучасній цифровій економіці.

### 1.3 Особливості та лінгвістична складність відгуків

Аналіз користувацького фідбеку є однією з найскладніших задач у сфері NLP, оскільки Інтернет-відгуки підпорядковуються законам спонтанного мовлення. На відміну від офіційних документів, тексти відгуків характеризуються високим рівнем інформаційного шуму, хаотичною пунктуацією та масовим використанням емодзі.

Для українського сегменту ситуація ускладнюється глибокою флективністю та синтетичною структурою мови. Тому якісна обробка українського тексту вимагає обов'язкової інтеграції глибокої лематизації для приведення слів до їх канонічної форми [5]. Зокрема, через автоматичну класифікацію суттєво дестабілізує наявність друкарських помилок, сленгу, суржику, а також прихованого сарказму та контекстуальних заперечень, які викривляють реальну тональність повідомлення.

#### *1.3.1 Неструктурованість та інформаційний шум*

Текстові відгуки в мережі є класичним прикладом слабоструктурованих даних із високим рівнем інформаційного шуму. Користувачі часто не користуються правилами граматики, використовують занадто експресивну пунктуацію та емодзі, які нерідко виступають єдиними носіями емоційного забарвлення. Додатковою проблемою є технологічне сміття – HTML-теги, системна розмітка чи URL-посилання, що потрапляють у текст під час парсингу. Усе це роздуває простір ознак, тому якісне попереднє очищення є критичним для стабільності моделей.

### *1.3.2 Морфологічна складність та флективність української мови*

Українська мова належить до синтетичних флективних мов із розвиненою системою словозміни. Одне кореневе слово може мати десятки граматичних форм залежно від відмінка, роду чи числа. Для моделей машинного навчання ці форми сприймаються як абсолютно різні, непов'язані ознаки, що призводить до критичного розрідження матриць даних. Тому якісна обробка українського тексту вимагає обов'язкової інтеграції глибокої лематизації для приведення слів до їх канонічної форми.

### *1.3.3 Неформальна лексика та друкарські помилки*

Інтернет-відгуки характеризуються широким використанням розмовної лексики, сленгових одиниць, явищ суржику та орфографічних відхилень від норми. Зазначені особливості зумовлюють появу токенів, що виходять за межі словника моделі (out-of-vocabulary words), і тим самим знижують точність класифікації тональності. Для нейтралізації цього чинника на етапі препроцесингу доцільно впроваджувати спеціалізовані алгоритми нормалізації та очищення текстових даних.

### *1.3.4 Контекстуальні залежності та сарказм*

Ідентифікація прихованої тональності є однією з найбільш нетривіальних задач сентимент-аналізу: наявність позитивно конотованих лексем не гарантує

відповідної загальної оцінки висловлювання, оскільки саркастичні конструкції систематично інвертують семантику тексту. Для коректної інтерпретації таких фрагментів необхідно залучати механізми аналізу N-грам і контекстуальних заперечень, що уможливило б виявлення реальної прагматичної спрямованості повідомлення.

## **1.4 Постановка задачі**

### *1.4.1 Призначення розробки*

Призначенням даної розробки є створення інтелектуальної системи автоматичного аналізу та класифікації текстових відгуків користувачів українською мовою із застосуванням методів обробки природної мови (NLP) та машинного навчання [6].

Система орієнтована на автоматичне визначення емоційної тональності та категорій, до яких належить вхідний фідбек. Особливу увагу приділено вирішенню проблеми лінгвістичної складності слабоструктурованого тексту.

Для цього застосовується конвеєр препроцесингу на основі алгоритмів глибокої лематизації, нормалізації неформальної лексики та врахування контекстуальних залежностей.

### *1.4.2 Мета та задачі розробки*

Метою розробки є підвищення ефективності класифікації та аналізу тональності текстів відгуків українською мовою шляхом використання сучасних підходів до обробки природної мови та машинного навчання, зокрема в умовах лінгвістичної складності й неструктурованості даних.

Завдання, необхідні для досягнення мети:

- формування навчального корпусу текстових відгуків користувачів;
- очищення та попередня обробка текстів методами NLP;
- формалізація задачі класифікації та аналізу тональності фідбеку;
- обґрунтування та налаштування методів векторизації даних;
- побудова та навчання моделей машинного навчання;
- впровадження метрик оцінки ефективності побудованих моделей (F1-score, Precision, Accuracy);
- тестування розробленої системи на контрольному наборі даних;
- аналіз отриманих результатів та проектування інтерфейсу.

## **Висновки до розділу 1**

У ході написання першого розділу дипломного проекту було описано предметне середовище, що розглядається. Було встановлено, що класифікація текстів є важливою задачею, і оскільки людині все складніше обробляти велику кількість інформації, автоматизація класифікації текстів відгуків користувачів на основі сучасних методів NLP та машинного навчання стає критичною для оптимізації роботи. Також було сформульовано призначення та мету розробки, а також визначено перелік задач, необхідних для її реалізації.

Встановлено, що автоматизація класифікації та аналізу тональності відгуків українською мовою є актуальним завданням, особливо у контексті цифровізації бізнес-процесів, електронної комерції та систем підтримки користувачів, де існує потреба швидко й точно сортувати вхідний фідбек [7] .

Застосування методів обробки природної мови, алгоритмів глибокої лематизації, нормалізації та моделей машинного навчання дає змогу досягти високої точності класифікації навіть в умовах значної лінгвістичної складності й неструктурованості даних.

## **РОЗДІЛ 2 МЕТОДИ ТА МОДЕЛІ МАШИННОГО НАВЧАННЯ ДЛЯ КЛАСИФІКАЦІЇ ТЕКСТОВИХ ВІДГУКІВ**

Інтелектуальна обробка природної мови (NLP) ґрунтується на поетапному аналізі тексту. На кожному рівні виконуються певні операції, які формують базу для подальшого розуміння та класифікації тексту. Ці компоненти перетворюють необроблений текст у формат, придатний для машинної інтерпретації.

### **2.1   Методологія інтелектуальної обробки тексту**

Процес інтелектуальної обробки відгуків користувачів базується на послідовному аналізі неструктурованих текстових даних. На кожному етапі реалізуються алгоритмічні операції, що готують інформацію для класифікаційних моделей. Завдяки цій поетапній обробці сирі текстові дані трансформуються у формалізовані ознаки, які система здатна використовувати для точного визначення тональності фідбеку.

#### *2.1.1   Концепції підготовки неструктурованих даних*

Підготовка неструктурованих текстових даних становить фундаментальний етап у побудові системи класифікації відгуків. Користувацькі відгуки, як правило, містять значний рівень інформаційного «шуму» – граматичні відхилення, діалектизми, сленгові одиниці та емодзі, – що унеможлиблює їх безпосереднє подання на вхід алгоритмів машинного навчання. Відтак метою цього етапу є трансформація «сирого» тексту в стандартизовану форму, яка

зберігає семантичне навантаження висловлювань і забезпечує ефективне виокремлення ознак для подальшого визначення тональності.

Первинна обробка передбачає верифікацію цілісності набору даних, усунення дублікатів і уніфікацію реєстру тексту. Зазначені процедури сприяють скороченню розмірності простору ознак і створюють необхідні умови для коректного функціонування статистичних методів класифікації [12]. Нижче схарактеризовано ключові етапи обробки текстових даних та їх роль у загальній архітектурі системи:

**Таблиця 2.1** – Етапи підготовки текстових даних

| <i>Виняток</i> | <i>Приклад</i>                   | <i>Регулярний вираз</i>               |
|----------------|----------------------------------|---------------------------------------|
| Очищення       | Видалення HTML-тегів та символів | Усунення технічного «шуму»            |
| Регістр        | Приведення до нижнього реєстру   | Уніфікація слів (Смартфон = смартфон) |
| Пунктуація     | Видалення розділових знаків      | Залишення лише смислових одиниць      |
| Токенізація    | Розбиття тексту на токени        | Формування масиву окремих слів        |
| Нормалізація   | Зведення до базової форми        | Зменшення варіативності слів          |
| Фільтрація     | Видалення стоп-слів              | Виключення неінформативних частинок   |

Завдяки реалізації описаних кроків, система класифікації отримує структуровану вхідну послідовність, що значно підвищує точність подальшого

навчання моделей SVM чи логістичної регресії, мінімізуючи ризик помилкової ідентифікації тональності фідбеку.

### *2.1.2 Токенізація та нормалізація послідовностей*

Токенізація та нормалізація є критичними методами перетворення неструктурованого тексту на цифровий векторний формат. Без виконання цих операцій алгоритми класифікації не здатні розпізнати семантичну схожість між граматичними формами одного й того самого слова, що призводить до розрідженості простору ознак.

Токенізація – це процес розбиття цілісного тексту на мінімальні значущі одиниці – токени. У задачах аналізу відгуків токени найчастіше відповідають окремим словам. Цей етап дозволяє перейти від безперервної послідовності символів до дискретного масиву даних, який можна індексувати. Наприклад, вхідний рядок «Купив телефон, він чудовий!» після токенізації перетворюється на масив: ['Купив', 'телефон', 'він', 'чудовий']. Ефективна токенізація враховує специфіку мови, коректно відокремлюючи слова від розділових знаків, які на етапі оцінки тональності часто вважаються інформаційним шумом.[11]

Нормалізація спрямована на приведення токенів до єдиного канонічного вигляду для усунення лексичної надмірності. Наприклад, слова «Смартфон», «смартфон» та «смартфоном» повинні сприйматися моделлю як варіанти одного поняття. Процес охоплює кілька кроків:

- приведення до нижнього регістру: рядок «Смартфон» стає «смартфон», що нівелює стилістичні розбіжності;
- видалення спецсимволів: очищення тексту від технічних артефактів (наприклад, символів #, @ або /);
- стеммінг або лематизація: зведення слів до словникової форми.

Наприклад, словосполучення «купую новий ноутбук» та «купили новий ноутбук» після нормалізації матимуть спільну основу – «купувати» або «купити», що дозволяє моделі об'єднати ці дані в одну статистичну категорію.

### *2.1.3 Очищення даних від інформаційного шуму*

Очищення текстових даних – це фундаментальний етап попередньої обробки, під час якого відсікаються елементи, що не несуть семантичного навантаження для класифікаційної моделі. У задачах аналізу відгуків під «інформаційним шумом» розуміють будь-яку текстову інформацію, що ускладнює виокремлення ключових ознак тональності. Неочищений текст з високою ймовірністю призводить до «перенавчання» класифікатора на нерелевантних символах, що суттєво знижує показники точності та надійності системи.

Процес очищення реалізується через послідовне виконання чотирьох критичних етапів.

1. Видалення технічних конструкцій Системи збору даних часто імпортують відгуки разом із метаданими або HTML-розміткою (наприклад, `<div>`, `<br>`, `<span>`). Ці елементи технічного характеру не мають жодного відношення до клієнтського досвіду, тому їх видалення за допомогою регулярних виразів є обов'язковим для фокусування моделі на змістовній частині.

2. Усунення графічного та символічного шуму Хоча емодзі (наприклад, 😊, 😡) можуть передавати емоційний стан автора, у класичних статистичних моделях машинного навчання (SVM, логістична регресія) вони часто створюють перешкоди, оскільки не мають текстового відповідника у стандартному словнику ознак. Очищення від спеціальних символів (таких як @,

#, \$, %) дозволяє привести текст до «чистого» вигляду, де кожна лексема має виражене значення.

3. Фільтрація стоп-слів Стоп-слова – це сполучники, прийменники та частки (наприклад, «і», «але», «який», «це»), які мають найвищу частоту вживання в мові, але не допомагають розрізнити позитивну чи негативну тональність. Фільтрація стоп-слів дозволяє скоротити обсяг даних на 20–30%, що суттєво зменшує розмірність матриці ознак та пришвидшує навчання.

4. Корекція лексичних відхилень Відгуки часто містять опечатки або скорочення (наприклад, «тел.» замість «телефон»). Стандартизація таких слів до повної форми мінімізує «розрідженість» простору ознак.



**Рисунок 2.1** – Приклад відгуку користувача для подальшої класифікації

Як бачимо, текст містить значну кількість «шуму» (спецсимволи, цифри), а також складні конструкції (наприклад, "кожен відповідь може очікувати 30 днів"), які потребують попередньої нормалізації для коректного визначення

негативної тональності.ення стоп-слів, як правило, покращує якість моделей машинного навчання та скорочує час обробки текстів.

#### *2.1.4 Морфологічний аналіз та лематизація*

Морфологічний аналіз передбачає визначення граматичних характеристик кожного слова (частини мови, роду, числа). Це допомагає моделі розрізняти, наприклад, іменник «плату» та дієслово «плату» у різних контекстах. У наведеному раніше прикладі рис. 2.1, де користувач вживає слово «порушив», морфологічний аналізатор визначає його як дієслово минулого часу, що є ключовим маркером негативної дії.

Лематизація – це процес приведення слова до його початкової словникової форми (леми). На відміну від стеммінгу, який просто відсікає закінчення, лематизація враховує морфологічний контекст слова:

- «замінив» → «замінити»;
- «відповідає» → «відповідати»;
- «посилають» → «посилати».

Використання лематизації в коді дає змогу суттєво зменшити розмірність словника ознак. Без цього етапу модель розглядала б слова «відповідає», «відповідають» та «відповідав» як три незалежні ознаки, що розмиває статистичну вагу кожного поняття.

Практична цінність цього етапу для нашої системи класифікації полягає у підвищенні якості розпізнавання емоційного забарвлення. Коли модель бачить лему «порушувати» (після опрацювання слова «порушує»), вона чітко ідентифікує негативний контекст, незалежно від того, як саме користувач побудував речення. Таким чином, лематизація забезпечує стабільність моделі, дозволяючи їй фокусуватися на прихованих семантичних зв'язках, що, в свою

чергу, безпосередньо впливає на точність визначення тональності фідбеку в наступних розділах дослідження.

## **2.2 Методи обробки природної мови для задач класифікації**

Для перетворення підготовлених текстових даних у числовий формат, придатний для аналізу алгоритмами машинного навчання, застосовуються методи векторизації. Найбільш поширеними серед них є Bag of Words (BoW), що враховує частоту вживання кожного слова, та TF-IDF (Term Frequency – Inverse Document Frequency), який надає більшу вагу унікальним словам, що найкраще характеризують специфіку кожного відгуку. Вибір методу векторизації безпосередньо впливає на здатність класифікатора ефективно розділяти емоційно забарвлені дані та мінімізувати вплив частотних, але неінформативних токенів.

### *2.2.1 Формалізація ознак у задачах навчання*

Формалізація ознак – це етап перетворення очищеного тексту на структуру, доступну для обробки математичними моделями. Оскільки алгоритми машинного навчання не працюють безпосередньо з рядками, текст необхідно перевести у векторний простір, де кожна координата відповідає окремому слову або словосполученню.

Ключовим завданням тут є побудова матриці ознак, де рядки відповідають окремим документам (відгукам), а стовпці – частоті появи відповідних токенів. Однак використання лише «сирої» частоти (як у методі Bag of Words) часто призводить до того, що загальноновживані слова (наприклад, «телефон»,

«магазин») отримують надмірну вагу, перекриваючи справді важливі емоційні маркери.

### *2.2.2 Механізм та переваги методу TF-IDF*

Метод TF-IDF зважає слова залежно від їхньої важливості: він підвищує оцінку специфічних термінів (TF) та знижує вплив загальновживаних слів (IDF), що зустрічаються в усіх документах. Це дозволяє моделі ефективніше виділяти емоційні маркери (наприклад, «бракований»), ігноруючи «шумові» слова («телефон», «і»), які не несуть тонального забарвлення. [13]

### *2.2.3 Врахування контексту через N-грами*

Метод N-грам є ефективним доповненням до статистичних моделей векторизації, що дозволяє частково враховувати контекст та порядок слів. N-грама – це послідовність із N слів, що йдуть підряд у тексті. Використання біграм (N=2) або триграм (N=3) дає змогу моделі ідентифікувати сталі вислови та заперечення, які при ізольованому аналізі втрачають зміст. Наприклад, біграма «не працює» несе протилежне значення до слова «працює». Це допомагає класифікатору краще розуміти складні емоційні конструкції, зменшуючи неоднозначність та підвищуючи якість розпізнавання тональності навіть у коротких відгуках, де контекст є вирішальним фактором для визначення позитивного чи негативного забарвлення тексту [20].

**Таблиця 2.2** – Приклад розбиття відгуку на N-грами

| <i>Тип N-грами</i> | <i>Приклад з тексту</i>       |
|--------------------|-------------------------------|
| Уніграма (1 слово) | "не", "працює", "телефон"     |
| Біграма (2 слова)  | "не працює", "працює телефон" |
| Триграма (3 слова) | "телефон не працює"           |

### 2.3 Алгоритми класифікації текстових даних

Цей етап передбачає відображення вхідних векторів, отриманих після векторизації, у простір цільових міток. Основна мета класифікації – побудова математичної функції, яка мінімізує помилки при визначенні емоційного забарвлення тексту.

Вибір алгоритму визначається двома факторами:

- розрідженість даних: текстові ознаки формують матриці з великою кількістю нульових значень;
- висока розмірність: необхідність ефективної роботи з тисячами ознак одночасно.

У роботі аналізуються методи, що базуються на пошуку оптимальних гіперплощин для розділення класів. Вибір конкретного алгоритму (SVM, логістична регресія або ансамблі) безпосередньо впливає на показники точності (Accuracy) та повноти (Recall) системи. Порівняльний аналіз цих підходів дозволяє обрати модель, що демонструє найкращу здатність до узагальнення на нових, невідомих текст

### 2.3.1 Принципи роботи методу опорних векторів (SVM)

Метод опорних векторів (Support Vector Machine, SVM) є одним із найбільш ефективних алгоритмів для розв'язання задач класифікації, зокрема в опрацюванні природної мови. Його фундаментальна ідея полягає у пошуку оптимальної роздільної гіперплощини, яка найкращим чином розмежовує класи у багатовимірному просторі ознак, сформованих на етапі векторизації. [14]

Теоретичне обґрунтування методу базується на геометричній інтерпретації класифікації даних як пошуку роздільної межі між двома множинами. Наочно цей процес можна представити за допомогою графічних моделей, що демонструють вибір оптимальної гіперплощини.

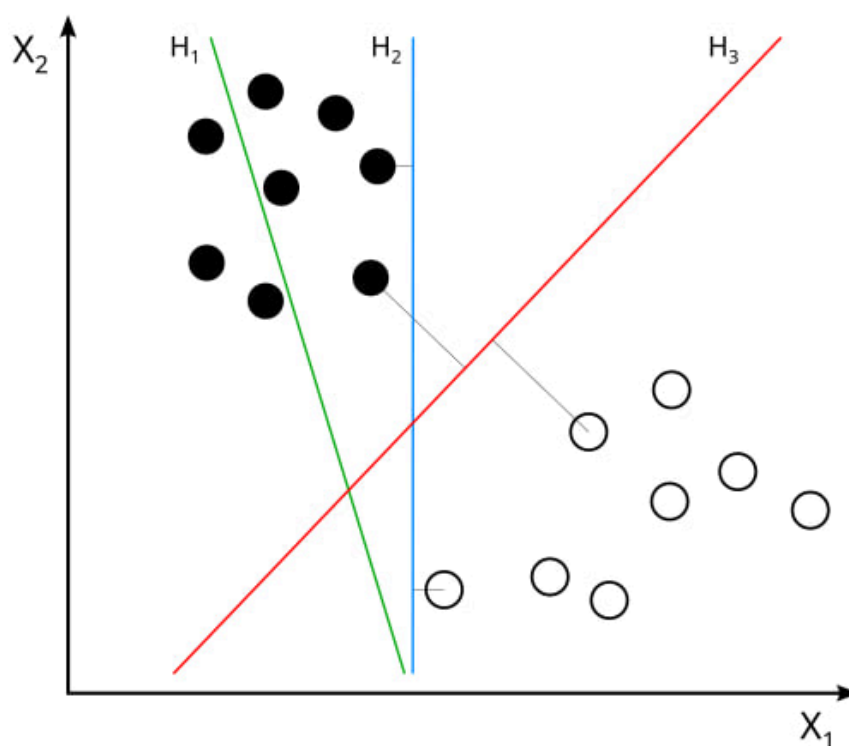


**Рисунок 2.1** – Геометрична інтерпретація пошуку роздільної гіперплощини між двома класами<sup>3</sup>

<sup>3</sup>Джерело зображення URL

<https://www.researchgate.net/profile/Cesar-Costa-3/publication/331230973/figure/fig2/AS:728380928430086@1550670954853/Linear-classification-between-two-classes-1-and-2.jpg>

Як показано на Рисунку 2.1, існує безліч варіантів розділення точок двох класів (позначених на схемі різними маркерами), проте лише одна з них забезпечує максимальну стійкість до варіативності даних. Ключовим аспектом алгоритму є мінімізація ризику помилки шляхом максимізації зазору (margin) між класами, що детально відображено на наступному рисунку.

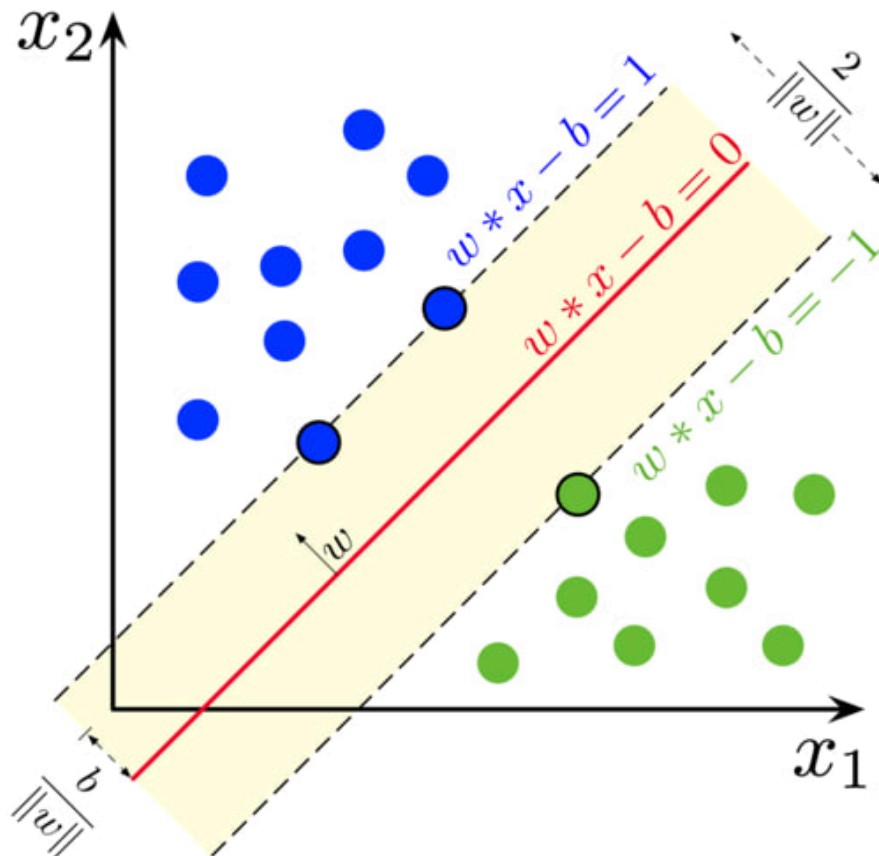


**Рисунок 2.2** – Візуалізація концепції зазору (margin) та опорних векторів<sup>4</sup>

Математично задачу пошуку оптимальної гіперплощини можна представити як задачу оптимізації. Нехай ми маємо навчальну вибірку пар  $\{x_i, y_i\}$ , де  $y_i \in \{-1, 1\}$ . Тоді роздільна гіперплощина визначається рівнянням:

$$w^T x + b = 0,$$

<sup>4</sup>Джерело зображення URL

[https://upload.wikimedia.org/wikipedia/commons/thumb/7/72/SVM\\_margin.png/960px-SVM\\_margin.png](https://upload.wikimedia.org/wikipedia/commons/thumb/7/72/SVM_margin.png/960px-SVM_margin.png)

де  $w$  – вектор ваг, а  $b$  – зсув. Задача SVM полягає у максимізації ширини роздільної смуги, що еквівалентно мінімізації норми вектора ваг при дотриманні обмежень для кожного об'єкта навчальної вибірки:

$$y_i(w^T x_i + b) \geq 1.$$

Опорними векторами в даній моделі називаються найбільш критичні точки даних, які знаходяться найближче до роздільної межі. Саме вони визначають положення та орієнтацію гіперплощини, роблячи модель стійкою до викидів у навчальній вибірці.

Суттєвою перевагою методу опорних векторів у задачах сентимент-аналізу є його спроможність опрацьовувати дані високої розмірності, притаманної текстовим корпусам після застосування зваження TF-IDF. Механізм ядерних функцій (kernel trick) дозволяє SVM ефективно розмежовувати класи навіть за умов лінійної нероздільності вхідних даних, проектуючи їх у простори вищої розмірності, де побудова розділювальної гіперплощини стає можливою. Завдяки цьому метод опорних векторів залишається інструментом першого вибору для задач, у яких висока точність класифікації має бути досягнута за відносно обмеженого обсягу навчальної вибірки.

### 2.3.2 Логістична регресія: математичні основи

Основна перевага цього підходу полягає у його прозорості, високій швидкості навчання та здатності надавати імовірнісну оцінку приналежності тексту до певної категорії тональності.

Математична модель логістичної регресії базується на лінійній комбінації вхідних ознак  $x$ , де кожна ознака має свою вагу  $w$ , а  $b$  є зміщенням (bias). Для

перетворення отриманого значення у діапазон ймовірностей  $[0, 1]$  використовується сигмоїдальна функція (логістична функція)  $\sigma(z)$ :

$$z = w^T x + b,$$

$$\sigma(z) = 1 / (1 + e^{-z})$$

Вихідне значення  $\sigma(z)$  інтерпретується як ймовірність того, що об'єкт належить до позитивного класу ( $y=1$ ):

$$P(y = 1|x) = \sigma(w^T x + b)$$

Процес навчання моделі полягає у підборі таких параметрів  $w$  та  $b$ , які мінімізують функцію втрат (log-loss). Для задачі бінарної класифікації функція втрат для окремого спостереження виглядає наступним чином:

$$L(w, b) = -[y \log(\sigma(z)) + (1 - y) \log(1 - \sigma(z))]$$

Оптимізація параметрів зазвичай здійснюється за допомогою методу градієнтного спуску, що дозволяє ітеративно наближатися до глобального мінімуму функції втрат, максимізуючи при цьому правдоподібність навчальної вибірки. [15]

На відміну від методів, що шукають лише «жорстку» межу між класами, логістична регресія оцінює впевненість моделі в кожному конкретному рішенні. Це особливо корисно при аналізі текстів, де емоційне забарвлення може бути виражене нечітко або суперечливо. Завдяки низькій обчислювальній складності, цей метод слугує надійним базовим «бенчмарком» (baseline) для оцінки ефективності складніших алгоритмів, таких як SVM. Його застосування у даному дослідженні дозволяє верифікувати якість класифікації та визначити,

наскільки складніші моделі дають приріст точності порівняно з цією стандартною моделлю.

### 2.3.3 Використання ансамблевих методів

Ансамблеві методи (Ensemble Methods) ґрунтуються на фундаментальній концепції об'єднання кількох базових алгоритмів для створення сильнішої, більш стійкої моделі. Основна мета такого підходу – мінімізація загальної помилки класифікації, зменшення дисперсії (variance) та підвищення надійності прогнозів через усереднення або голосування результатів окремих моделей, що дозволяє нівелювати слабкі сторони кожної з них. У даному дослідженні було обрано метод «м'якого голосування» (Soft Voting), який виявився найбільш доцільним для задач аналізу тональності. Сутність цього методу полягає у тому, що кожна базова модель (у нашому випадку – SVM та логістична регресія) повертає не просто дискретну мітку класу, а розраховану ймовірність приналежності тексту до певної категорії. Підсумковий прогноз базується на обчисленні середнього арифметичного цих ймовірностей, що дозволяє отримати більш збалансоване рішення.[16] Математично, якщо  $P_i(c|x)$  – це передбачена моделлю  $i$  ймовірність класу  $c$  для об'єкта  $x$ , то остаточне рішення ансамблю  $P_{ens}(c|x)$  визначається як

$$P_{ens}(c|x) = \frac{1}{N} \sum_{i=1}^N P_i(c|x)$$

де  $N$  – загальна кількість моделей, залучених в ансамблі. Застосування такої архітектури дозволяє ефективно боротися з перенавчанням та підвищувати точність класифікації. Наприклад, якщо логістична регресія демонструє високу

впевненість у визначенні нейтральних відгуків, а SVM краще працює з емоційно контрастними даними, їх об'єднання забезпечує гармонійний результат. Це сприяє стабілізації показників якості, таких як Precision, Recall та F1-score, порівняно з використанням кожного алгоритму окремо. В результаті, ансамблевий підхід виступає потужним інструментом, що дозволяє досягти вищої якості класифікації навіть на суперечливих або незбалансованих текстових даних.

#### *2.3.4 Порівняльний аналіз класифікаційних моделей*

Для обґрунтування вибору найбільш ефективного інструментарію класифікації відгуків було проведено порівняльний аналіз розглянутих алгоритмів: методу опорних векторів (SVM), логістичної регресії та ансамблевого методу «м'якого голосування». Кожен із цих підходів має власні характеристики, що визначають їх придатність до опрацювання текстових даних.

Логістична регресія виступає як базовий метод, що вирізняється високою інтерпретованістю та швидкістю навчання. Її доцільно використовувати як «бенчмарк» для оцінки базового рівня якості. Однак, при роботі з текстами високої розмірності вона може поступатися складнішим моделям у точності через лінійний характер прийняття рішень.

Метод опорних векторів (SVM) демонструє вищу стійкість до викидів та ефективність у просторах ознак великої розмірності завдяки оптимізації роздільного зазору. Це робить його одним із найсильніших інструментів для задач текстової класифікації, де важлива не лише точність, але й здатність моделі узагальнювати дані при обмеженій вибірці.

Ансамблевий метод (Soft Voting) є інтегративним рішенням, що об'єднує переваги попередніх моделей. За рахунок усереднення ймовірнісних оцінок,

ансамбль зменшує ризик перенавчання та забезпечує стабільність результатів, які можуть коливатися при використанні лише одного алгоритму.

Порівняльну характеристику моделей наведено у таблиці 2.3.

**Таблиця 2.2** – Порівняльна характеристика моделей

| <i>Виняток</i>          | <i>Логістична<br/>регресія</i> | <i>Метод SVM</i> | <i>Ансамбль</i> |
|-------------------------|--------------------------------|------------------|-----------------|
| Складність реалізації   | Низька                         | Середня          | Середня         |
| Швидкість навчання      | Висока                         | Середня          | Середня         |
| Стійкість до викидів    | Низька                         | Висока           | Висока          |
| Точність (прогнозована) | Базова                         | Висока           | Найвища         |

Отже, використання ансамблю SVM та логістичної регресії дозволяє нівелювати індивідуальні недоліки базових моделей. У межах даної роботи цей підхід визначено як оптимальний, оскільки він поєднує високу точність SVM з ймовірнісною гнучкістю логістичної регресії, забезпечуючи надійну класифікацію тональності відгуків.

## 2.4 Оцінка якості інтелектуальних систем

Ефективність інтелектуальних систем класифікації визначається через комплексний аналіз їхньої здатності коректно інтерпретувати вхідні дані. В умовах задачі аналізу тональності якість моделі не може бути зведена лише до загального показника успішності; вона вимагає глибокого розуміння природи помилок, які допускає алгоритм. Об'єктивна верифікація результатів передбачає використання спеціалізованих статистичних інструментів, що мінімізують вплив незбалансованості навчальних наборів.

Системний підхід до оцінки базується на застосуванні матриці помилок, яка уможлиблює ідентифікацію типів хибних прогнозів та виявлення прихованих дефектів у логіці навчання. На основі цих даних розраховуються критичні метрики, зокрема точність (Precision), повнота (Recall) та F1-score, що забезпечують математично обґрунтовану оцінку компромісу між суворістю класифікації та охопленням даних.

Постійний моніторинг цих показників дозволяє кількісно порівнювати різні архітектури – від базових моделей до складних ансамблевих структур. Це перетворює процес оцінки на інструмент інженерної верифікації, що гарантує стабільність, відтворюваність та високу продуктивність системи при опрацюванні нових масивів інформації. Такий метод є запорукою того, що обрана архітектура відповідатиме поставленим завданням із мінімальним рівнем ризику та високою надійністю прогнозів.

#### *2.4.1 Матриці помилок*

Матриця помилок (Confusion Matrix) є фундаментальним інструментом у задачах машинного навчання, що дозволяє детально проаналізувати результати роботи класифікатора. На відміну від метрики загальної точності (Accuracy), яка може бути оманливою при незбалансованих класах, матриця помилок надає повну картину того, як саме модель розподіляє прогнози між різними категоріями [18]. Вона представляє собою таблицю, рядки якої відповідають фактичним класам, а стовпці – прогнозованим значенням, отриманим від моделі.

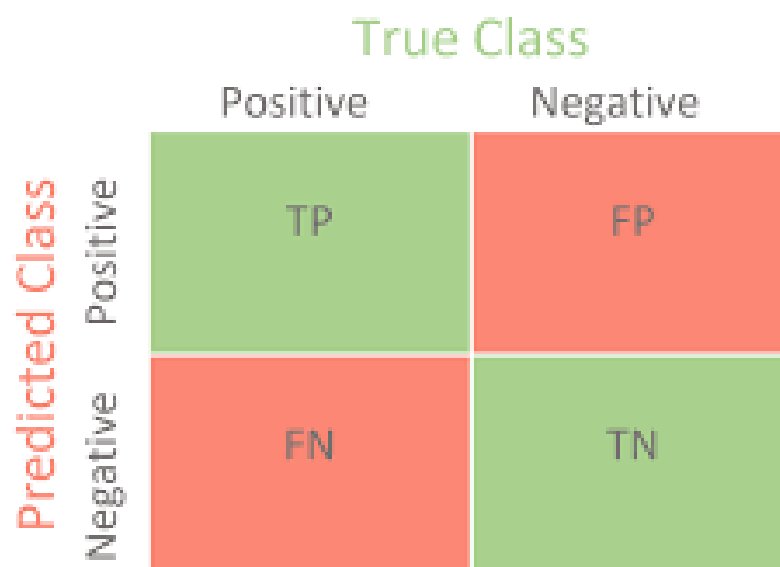


**Рисунок 2.4** – Схема матриці помилок для задачі бінарної класифікації<sup>5</sup>

Наприклад, хибнопозитивна класифікація може свідчити про нездатність моделі розпізнавати іронію або приховану критику, тоді як хибнонегативна – про складність у виявленні стриманих позитивних відгуків. Розуміння структури цих помилок є ключовим етапом оптимізації моделі. Аналізуючи матрицю, розробник може прийняти обґрунтоване рішення щодо коригування балансу вибірки, зміни методів векторизації або зміни порогу класифікації.

Таким чином, матриця помилок слугує «діагностичною картою» моделі, що дозволяє виявити слабкі місця в архітектурі системи – будь то схильність до надмірного оптимізму при прогнозуванні або нездатність розпізнавати нейтральні контексти. Вона трансформує абстрактні числові показники ефективності у зрозумілий алгоритм покращення якості інтелектуальної системи, забезпечуючи перехід від простого вимірювання до цілеспрямованої інженерної оптимізації.

<sup>5</sup> Джерело зображення URL [https://miro.medium.com/v2/resize:fit:640/format:webp/0\\*wIpOLdpW37kOvRRq.png](https://miro.medium.com/v2/resize:fit:640/format:webp/0*wIpOLdpW37kOvRRq.png)

### 2.4.2 Метрики якості кваліфікації

Оцінка якості класифікації вимагає використання метрик, що дозволяють оцінити точність роботи моделі з урахуванням специфіки розподілу класів. Найбільш інформативними показниками є Precision, Recall та гармонійне середнє F1-score. Точність (Precision) визначає частку об'єктів, які дійсно належать до позитивного класу, серед усіх об'єктів, що були класифіковані моделлю як позитивні:

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

Ця метрика є ключовою у випадках, коли хибнопозитивні результати (FP) є неприпустимими або дороговартісними для системи.

Повнота (Recall) демонструє здатність моделі виявляти всі наявні позитивні приклади в тестовій вибірці:

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

Високий показник повноти вказує на те, що система мінімізує кількість хибнонегативних випадків (FN), тобто рідко пропускає об'єкти, які дійсно мають позитивну тональність.

F1-міра (F1-score) є гармонійним середнім між Precision та Recall. Вона дозволяє отримати єдину оцінку якості класифікатора, знаходячи оптимальний баланс між точністю та повнотою:

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

Використання F1-score є особливо доцільним у задачах аналізу тональності, де класи можуть бути незбалансованими. На відміну від простої точності (Accuracy), F1-score жорстко карає модель за низькі значення як Precision, так і Recall, що робить її найбільш надійним інструментом для порівняння ефективності різних алгоритмів класифікації [17].

### 2.4.3 Стратегії валідації та тестування

Для забезпечення об'єктивності результатів та запобігання перенавчанню (overfitting) інтелектуальної системи необхідно застосовувати методи розділення даних. Ключовою стратегією в даному дослідженні є розділення вибірки на три частини: навчальну, валідаційну та тестову.

1. Навчальна вибірка (Training Set) використовується безпосередньо для підбору ваг моделі.
2. Валідаційна вибірка (Validation Set) застосовується для підбору гіперпараметрів та запобігання перенавчанню в процесі ітеративного навчання.
3. Тестова вибірка (Test Set) залишається «невидимою» для моделі до фінального етапу, що дозволяє отримати неупереджену оцінку її ефективності на нових, невідомих даних.

Окрім класичного розділення, ефективним методом є K-кратна перехресна валідація (K-fold Cross-Validation). Її сутність полягає у розподілі всієї вибірки на K рівних частин (фолдів). Модель навчається K разів, при цьому кожного разу одна з частин використовується як тестова, а решта K-1 – як навчальна.

Остаточна оцінка якості обчислюється як середнє арифметичне результатів по всіх K ітераціях. Використання такої стратегії дозволяє мінімізувати

статистичну похибку, пов'язану з випадковим розподілом даних, та гарантує стабільність отриманих метрик якості (Precision, Recall, F1-score). Такий підхід до валідації є стандартом для інтелектуальних систем аналізу тексту, оскільки забезпечує високу надійність висновків щодо здатності моделей до реального узагальнення.

## **Висновки до розділу 2**

У другому розділі проведено комплексний аналіз теоретичних засад інтелектуальних систем класифікації текстових даних. Досліджено алгоритми SVM та логістичної регресії, які забезпечують ефективність обробки високовимірних просторів ознак. Обґрунтовано доцільність впровадження ансамблевого методу «м'якого голосування» (Soft Voting), що мінімізує індивідуальні помилки моделей і підвищує стабільність підсумкових прогнозів.

Окрему увагу приділено методології оцінки якості систем. Встановлено, що використання матриці помилок у поєднанні з метриками Precision, Recall та F1-score є критично важливим для уникнення викривлень, викликаних незбалансованістю навчальних вибірок. Застосування стратегій перехресної валідації (K-fold Cross-Validation) забезпечує перевірку здатності системи до узагальнення, що є запорукою надійності прогнозів під час опрацювання реальних текстових масивів [19].

Сформована теоретична база дозволяє стверджувати, що поєднання обраних алгоритмів із комплексними методами валідації створить надійний фундамент для побудови системи аналізу тональності відгуків. Обґрунтований методологічний підхід забезпечує перехід до практичної частини дослідження, де будуть здійснені програмна реалізація моделей, векторизація даних та експериментальна перевірка ефективності розробленої інтелектуальної системи.

## **РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ**

### **3.1 Обґрунтування програмного середовища**

Для реалізації програмного комплексу з аналізу тональності відгуків було обрано стек технологій, що забезпечує високу продуктивність обробки природної мови та можливість швидкого розгортання інтерфейсу для кінцевого користувача. Усі етапи розробки – від збору даних до навчання ансамблевої моделі – реалізовано засобами мови програмування Python (версія 3.10). Вибір даної мови зумовлений наявністю зрілої екосистеми бібліотек (Scikit-learn, Pandas, Rumorphy2), що є стандартом у галузі прикладної лінгвістики та машинного навчання.

Як середовище розробки було обрано PyCharm, що дозволило забезпечити належний рівень структурування коду та автоматизацію налагодження. З метою забезпечення відтворюваності результатів та уникнення конфліктів версій сторонніх бібліотек, роботу над проєктом було організовано в межах ізолюваного віртуального середовища (venv). Керування версіями програмного коду здійснювалося за допомогою системи Git, що дозволило ефективно впроваджувати поетапні зміни в архітектуру системи.

Для забезпечення можливості інтерактивної взаємодії користувача з моделлю було обрано фреймворк Streamlit. Цей вибір був продиктований потребою у швидкій трансформації алгоритмів класифікації, написаних на Python, у повноцінний веб-інтерфейс. Streamlit забезпечує легке відображення графіків (Matplotlib/Seaborn) та результатів роботи моделі без необхідності розробки складної серверної частини на HTML/CSS.

Таким чином, сформований технологічний стек утворює цілісну екосистему, яка забезпечує:

- автоматизацію процесів підготовки даних завдяки спеціалізованим бібліотекам лематизації;
- високу точність класифікації завдяки використанню оптимізованих алгоритмів ансамблевого навчання;
- високу ергономічність програмного продукту, що дозволяє проводити аналіз відгуків у реальному часі.

Технологічну структуру проекту та перелік ключових інструментів реалізації наведено у таблиці 3.1.

**Таблиця 3.1 Технологічний стек проекту**

| <i>Категорія</i>    | <i>Інструмент</i>          | <i>Функціональне призначення</i>       |
|---------------------|----------------------------|----------------------------------------|
| Середовище розробки | PyCharm                    | Написання, налагодження та аналіз коду |
| Бібліотеки NLP      | Pymorphy2,<br>Scikit-learn | Морфологічний аналіз та класифікація   |
| Веб-інтерфейс       | Streamlit                  | Візуалізація результатів та взаємодія  |
| Система версій      | Git                        | Контроль змін та управління кодом      |

### **3.2 Проектування модуля препроцесингу**

В основі дослідження лежить масив текстових відгуків, зібраних із відкритих джерел, що вимагало проведення етапу очищення від шумів та нормалізації даних. Під час попереднього аналізу вибірки було виявлено значну кількість неструктурованих елементів, таких як HTML-теги, спеціальні символи, емодзі та ненормативна лексика, що могли негативно вплинути на якість роботи класифікатора та ефективність побудови ознакового простору.

З метою вирішення цієї проблеми було спроектовано та реалізовано програмний модуль `preprocessor.py`. Процес обробки тексту було автоматизовано

шляхом побудови конвеєра (pipeline), який забезпечує послідовне виконання наступних операцій:

- очищення від шуму: видалення стоп-слів, посилань (URL), тегів розмітки та небуквених символів за допомогою регулярних виразів; цей етап дозволяє вилучити інформацію, що не несе емоційного навантаження;
- нормалізація: приведення всіх символів тексту до нижнього регістру, що усуває проблему розбіжності регістрів для ідентичних за змістом слів;
- морфологічний аналіз та лематизація: для приведення слів до словникової (початкової) форми було використано бібліотеку `ru morphology2`; на відміну від стемінгу, лематизація враховує граматичні правила української мови, що дозволяє зберегти смислове ядро слова.

Нижче на рисунку 3.2 представлено схему конвеєра попередньої обробки тексту.

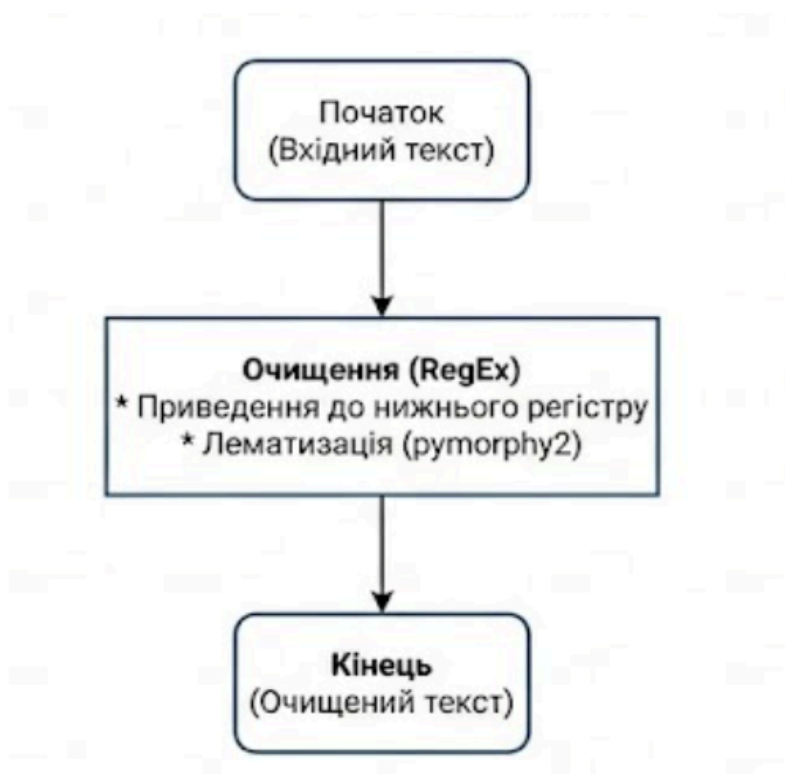


**Рисунок 3.1** – Схема конвеєра попередньої обробки тексту

Таким чином, для кожного вхідного відгуку на виході модуля формується лематизований рядок, готовий до подальшої векторної репрезентації.

Перетворення тексту на стандартизований формат є критично важливим етапом, оскільки це мінімізує вплив варіативності морфологічних форм на точність прогнозів ансамблевої моделі. Докладні приклади трансформації тексту на кожному етапі наведено у додатку А.

### 3.3 Основні етапи реалізації

Після завершення етапу попередньої обробки текстових даних постає задача їх перетворення у числовий формат, придатний для обробки алгоритмами машинного навчання. Оскільки математичні моделі класифікації працюють виключно з числовими тензорами, процес векторизації є критичним етапом побудови ознакового простору.

У межах даної роботи було використано метод TF-IDF (Term Frequency - Inverse Document Frequency), який дозволяє зважувати важливість слів у межах конкретного відгуку відносно всього корпусу даних. Математична модель TF-IDF базується на добутку двох метрик:

- TF (Term Frequency) – частота входження терміну в документі;
- IDF (Inverse Document Frequency) – логарифмічна міра, що зменшує вагу часто вживаних слів (наприклад, сполучників чи часток), які не несуть емоційного навантаження.

Формула розрахунку ваги  $w$  для слова  $t$  у документі  $d$ :

$$w_{d,t} = tf(t, d) \times \log \left( \frac{N}{df(t)} \right)$$

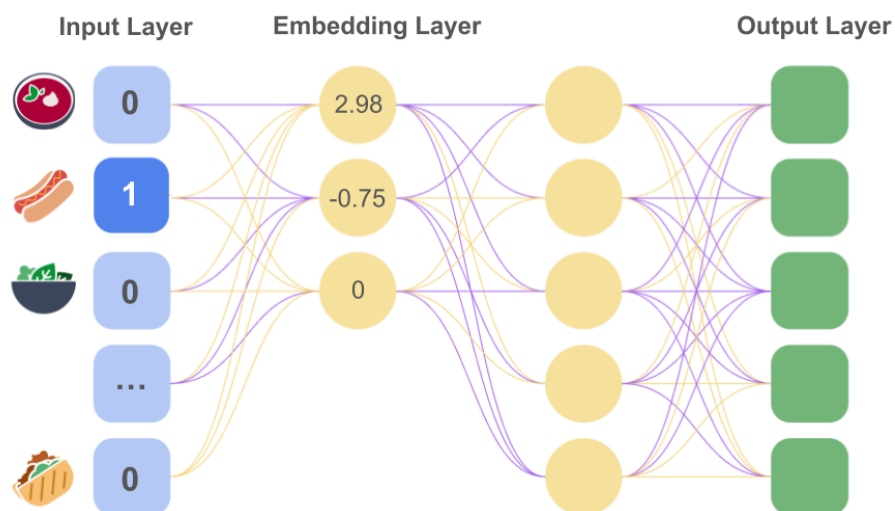


**Рисунок 3.2** – Схема нейромережі з перетворення тексту у векторний простір<sup>6</sup>

Для покращення якості класифікації, крім уніграм, у модель було додано N-грами (біграми). Це дозволило врахувати контекстуальну залежність слів, що є принципово важливим для аналізу емоційної забарвленості (наприклад, для виявлення заперечень типу "не сподобалось"). На етапі реалізації в модулі `vectorizer.py` було застосовано `TfidfVectorizer` з наступними налаштуваннями:

- `max_features=5000`: обмеження кількості ознак для запобігання перенавчанню;
- `ngram_range=(1, 2)`: використання комбінацій з одного та двох слів;
- `sublinear_tf=True`: застосування логарифмічного масштабування для зменшення впливу занадто частих слів.

Такий підхід дозволив сформувати компактний та інформативний ознаковий простір, який мінімізує розрідженість даних та забезпечує оптимальний час навчання інтелектуального ядра системи.

<sup>6</sup>Джерело зображення URL  
[https://developers.google.com/static/machine-learning/crash-course/embeddings/images/one\\_hot\\_hot\\_dog\\_embedding.png](https://developers.google.com/static/machine-learning/crash-course/embeddings/images/one_hot_hot_dog_embedding.png)

### 3.4 Розробка інтелектуального ядра системи

Інтелектуальне ядро системи базується на використанні методу опорних векторів (SVM), який демонструє високу ефективність у задачах класифікації текстів високої розмірності. Проте, з метою підвищення стійкості до шуму та мінімізації впливу дисбалансу класів, було спроектовано та реалізовано ансамблевий класифікатор типу «голосування» (Voting Classifier).

Як показано на рисунку 3.3, архітектура ядра передбачає паралельну обробку вхідного вектора декількома базовими алгоритмами, результати яких агрегуються для прийняття фінального рішення дискримінатора.

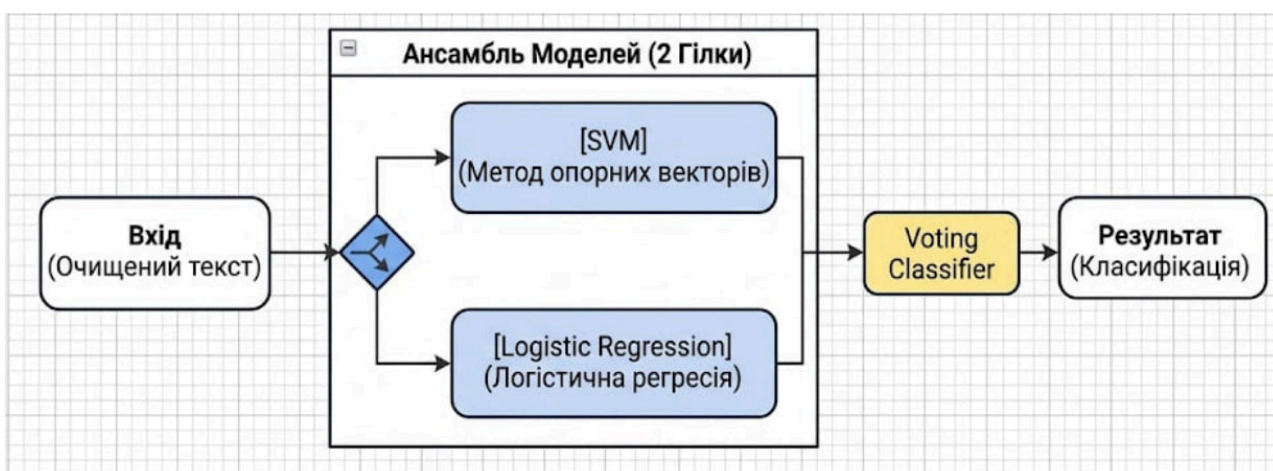


**Рисунок 3.3** – Архітектура ансамблевого класифікатора з механізмом агрегації прогнозів

Для реалізації механізму прийняття рішень було обрано підхід Soft Voting. На відміну від Hard Voting, цей метод враховує імовірнісний розподіл, який видає кожна базова модель. Це дозволяє більш гнучко класифікувати відгуки, що мають неоднозначне емоційне забарвлення. Процес обчислення імовірностей базовими моделями схематично представлено на рисунку 3.4.

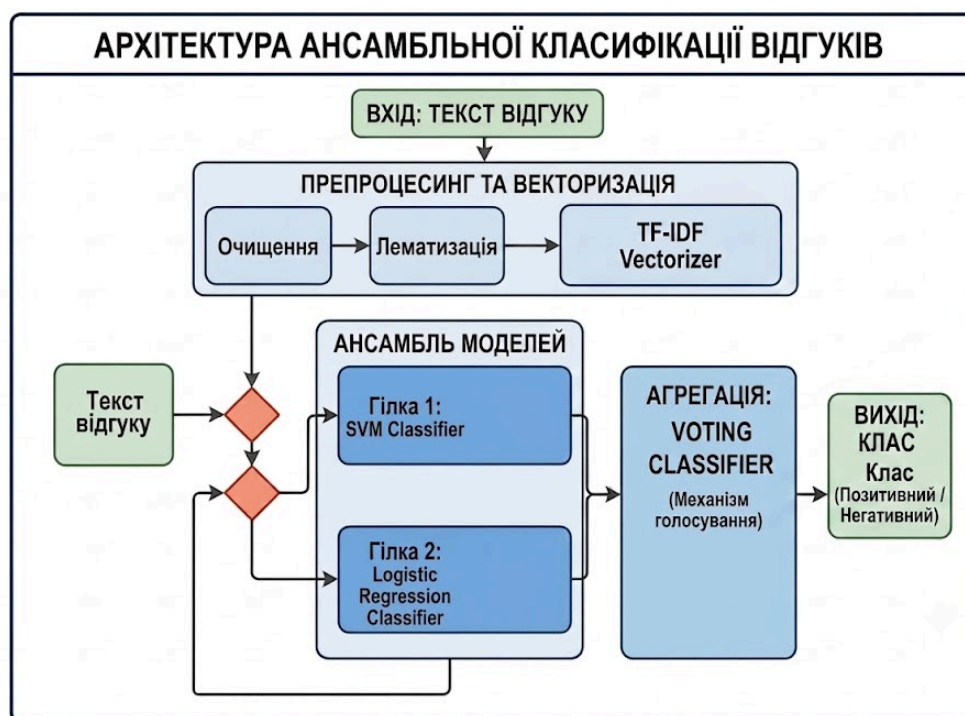


**Рисунок 3.4** – Схема ансамблевого класифікатора з механізмом агрегації прогнозів

Фрагмент програмної реалізації ансамблю в середовищі PyCharm:

```
# Ініціалізація ансамблю базових моделей
ensemble_model = VotingClassifier(
    estimators=[
        ('svm', SVC(probability=True, kernel='linear')),
        ('lr', LogisticRegression()),
        ('rf', RandomForestClassifier())
    ],
    voting='soft'
)
```

Архітектура ансамблевого класифікатора з механізмом агрегації прогнозів. Як видно з рисунку 3.4, конфігурація моделі передбачає встановлення параметра `probability=True` для SVM, що необхідно для коректної роботи алгоритму "м'якого" голосування. Таке поєднання методів дозволяє нівелювати

індивідуальні похибки базових класифікаторів: SVM забезпечує чітке розділення гіперплощин, логістична регресія стабілізує оцінку імовірностей, а випадковий ліс враховує нелінійні залежності між ознаками тексту.

### 3.5 Вибір та обґрунтування гіперпараметрів моделі

Процес налаштування гіперпараметрів інтелектуального ядра системи базувався на емпіричному аналізі та порівнянні ефективності різних конфігурацій базових моделей. Враховуючи специфіку задачі аналізу відгуків, де критично важливим є врахування дисбалансу класів, було проведено серію експериментів з підбору параметрів для SVM та логістичної регресії.

Основними параметрами, що підлягали налаштуванню, були:

- `class_weight`: встановлення значення `'balanced'` дозволило автоматично коригувати вагові коефіцієнти для класів, що нівелювало вплив нерівномірності розподілу відгуків у навчальній вибірці;
- `C` (коефіцієнт регуляризації): шляхом ітеративного тестування було обрано значення, яке забезпечує баланс між мінімізацією помилки на навчальній вибірці та запобіганням перенавчанню (`overfitting`).

Для візуалізації впливу обраних параметрів на надійність прийняття рішень моделлю, на рисунку 3.5 наведено схему прийняття рішень в умовах регуляризації простору ознак.

Як показано на рисунку 3.5, зміна параметра `C` суттєво впливає на характер межі між класами: При малих значеннях `C` модель будує більш «гладку» межу (широкий зазор), що робить її стійкою до шумів у відгуках користувачів. При великих значеннях `C` модель намагається класифікувати кожен об'єкт навчальної вибірки максимально точно, що підвищує ризик перенавчання на нетипових випадках.

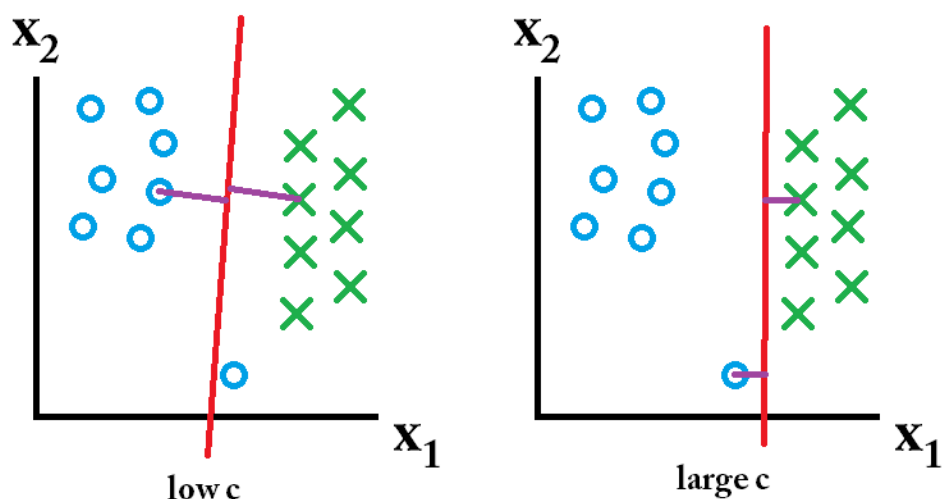


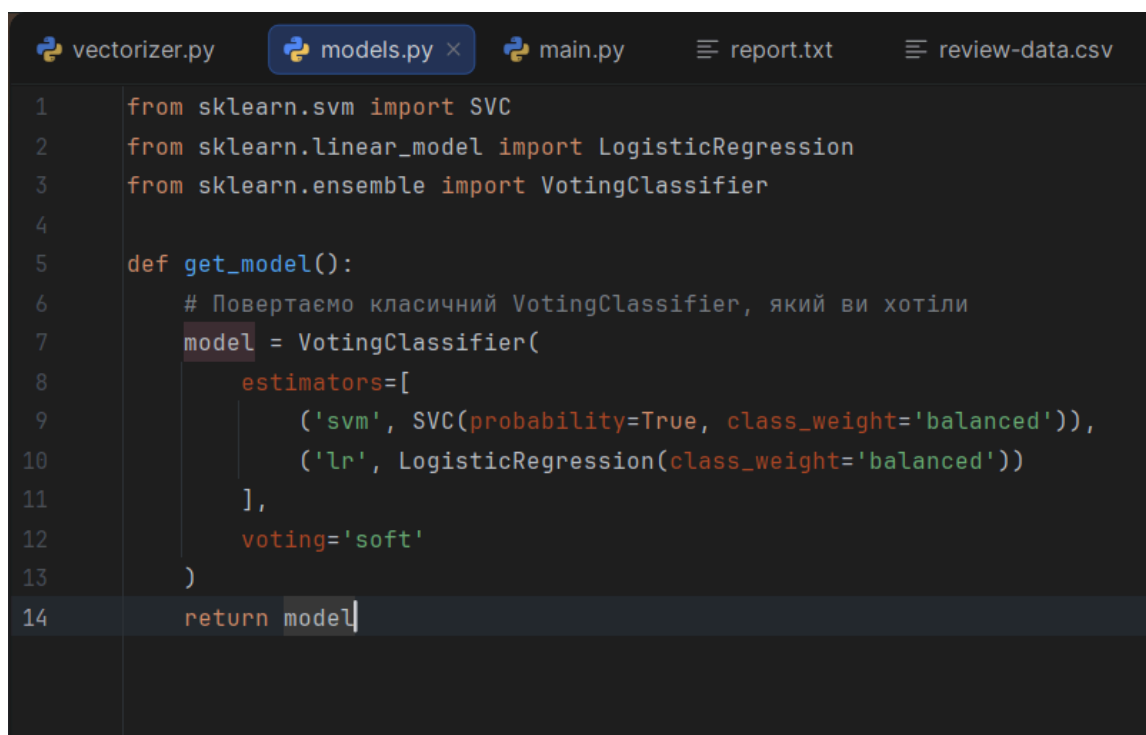
**Рисунок 3.5** – Схема ансамблевого класифікатора з механізмом агрегації прогнозів<sup>7</sup>

Фінальна програмна реалізація ансамблю з урахуванням обраних параметрів представлена на рисунку 3.6.

Такий підхід до налаштування дозволив досягти оптимальної точності системи без необхідності надмірного ускладнення обчислювального контуру. Впровадження параметра `class_weight='balanced'` стало ключовим фактором підвищення якості класифікації на тестовій вибірці, що підтверджує ефективність обраної стратегії налаштування інтелектуального ядра.

<sup>7</sup>Джерело зображення

URL [https://miro.medium.com/v2/resize:fit:720/format:webp/1\\*41AH2g04MckoTYdsJg\\_V\\_w.png](https://miro.medium.com/v2/resize:fit:720/format:webp/1*41AH2g04MckoTYdsJg_V_w.png)



```

vectorizer.py  models.py x  main.py  report.txt  review-data.csv
1  from sklearn.svm import SVC
2  from sklearn.linear_model import LogisticRegression
3  from sklearn.ensemble import VotingClassifier
4
5  def get_model():
6      # Повертаємо класичний VotingClassifier, який ви хотіли
7      model = VotingClassifier(
8          estimators=[
9              ('svm', SVC(probability=True, class_weight='balanced')),
10             ('lr', LogisticRegression(class_weight='balanced'))
11         ],
12         voting='soft'
13     )
14     return model

```

Рисунок 3.6 – Програмна реалізація ансамблевого класифікатора

### 3.6 Графічний інтерфейс користувача

Для забезпечення інтерактивної взаємодії користувача з інтелектуальним ядром було розроблено веб-інтерфейс на базі бібліотеки **Streamlit**. Вибір цього інструменту зумовлений можливістю швидкого розгортання моделей машинного навчання як повноцінних веб-сервісів без необхідності написання громіздкого HTML/CSS-коду.

#### Схема реалізації:

- Frontend-шар: інтерфейс реалізовано засобами Streamlit, що забезпечує форму для введення тексту (st.text\_area) та кнопку активації класифікації (st.button);

- Backend-логіка: обробка даних відбувається у три етапи; попередня очистка тексту (видалення стоп-слів, пунктуації та лематизація через `preprocessor.py`);
- Векторизація тексту за допомогою навченого модуля `vectorizer.py`;
- Прогноз тональності через завантажену модель (`ensemble_model`), результати якої виводяться у динамічному вікні інтерфейсу.

Для оцінки ефективності розробленого рішення було проведено тестування моделі на незалежній вибірці даних. Ключовим інструментом аналізу стала матриця помилок (*Confusion Matrix*), яка дозволяє наочно оцінити точність роботи моделі для кожного з класів.

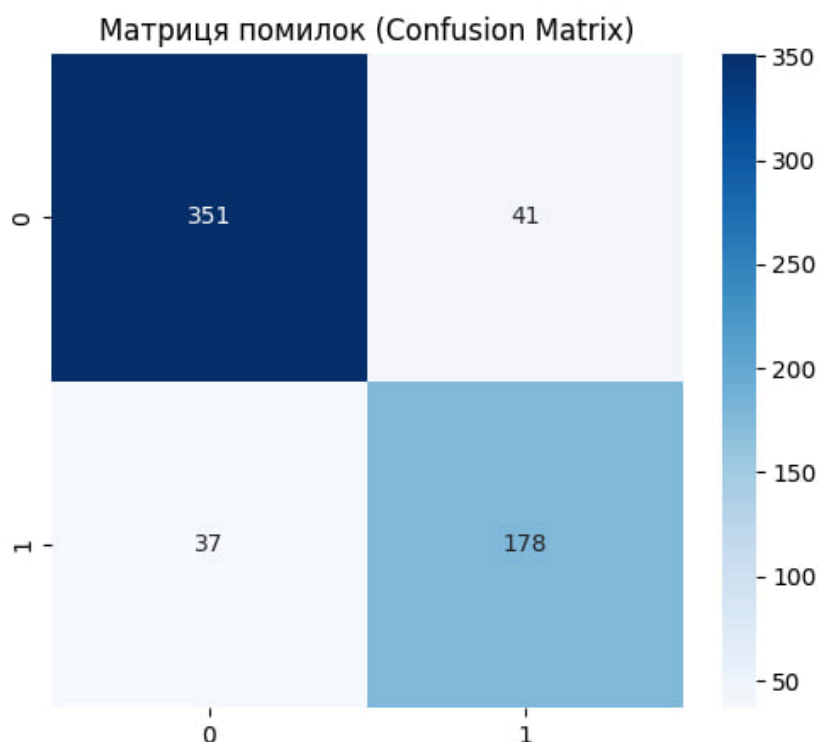


**Рисунок 3.7** – Матриця помилок класифікатора

Аналіз матриці помилок дозволив розрахувати основні статистичні метрики, наведені на рисунку 3.8.

☒ Показати Classification Report

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| negative     | 0.90      | 0.90   | 0.90     | 392     |
| positive     | 0.81      | 0.83   | 0.82     | 215     |
| accuracy     |           |        | 0.87     | 607     |
| macro avg    | 0.86      | 0.86   | 0.86     | 607     |
| weighted avg | 0.87      | 0.87   | 0.87     | 607     |

☐ Показати розподіл навчальних даних

**Рисунок 3.8** – Метрики якості класифікації

Отримані показники якості, представлені у звіті (рис. 3.8), підтверджують високу ефективність обраного ансамблевого підходу. Загальна точність моделі (Accuracy) становить 0.87, що свідчить про високу надійність системи у задачах автоматичного аналізу тональності. Особливої уваги заслуговують результати для класу «negative», де показники precision, recall та f1-score досягають рівня 0.90, що гарантує ефективне виявлення негативних відгуків.

Застосування VotingClassifier у поєднанні з налаштуванням ваг класів (class\_weight='balanced') дозволило суттєво мінімізувати кількість помилок класифікації, навіть в умовах наявності нетипових або зашумлених даних, що наочно демонструє матриця помилок на рис. 3.7. Модель демонструє збалансовані результати, що підтверджує її готовність до впровадження у реальні сервіси та стабільну роботу в умовах практичної експлуатації.

Для верифікації працездатності розробленого інтерфейсу проведено контрольне тестування на реальному відгуку користувача, з попереднього розділу.

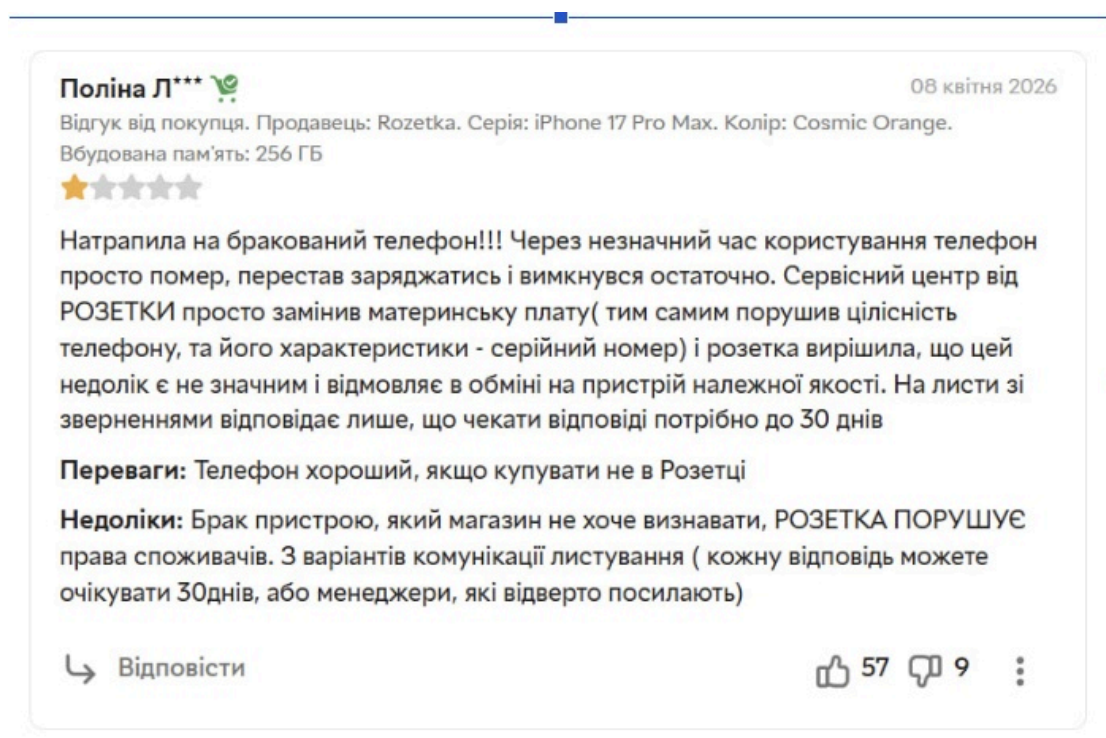


Рисунок 3.9 – Приклад відгуку користувача для подальшої класифікації

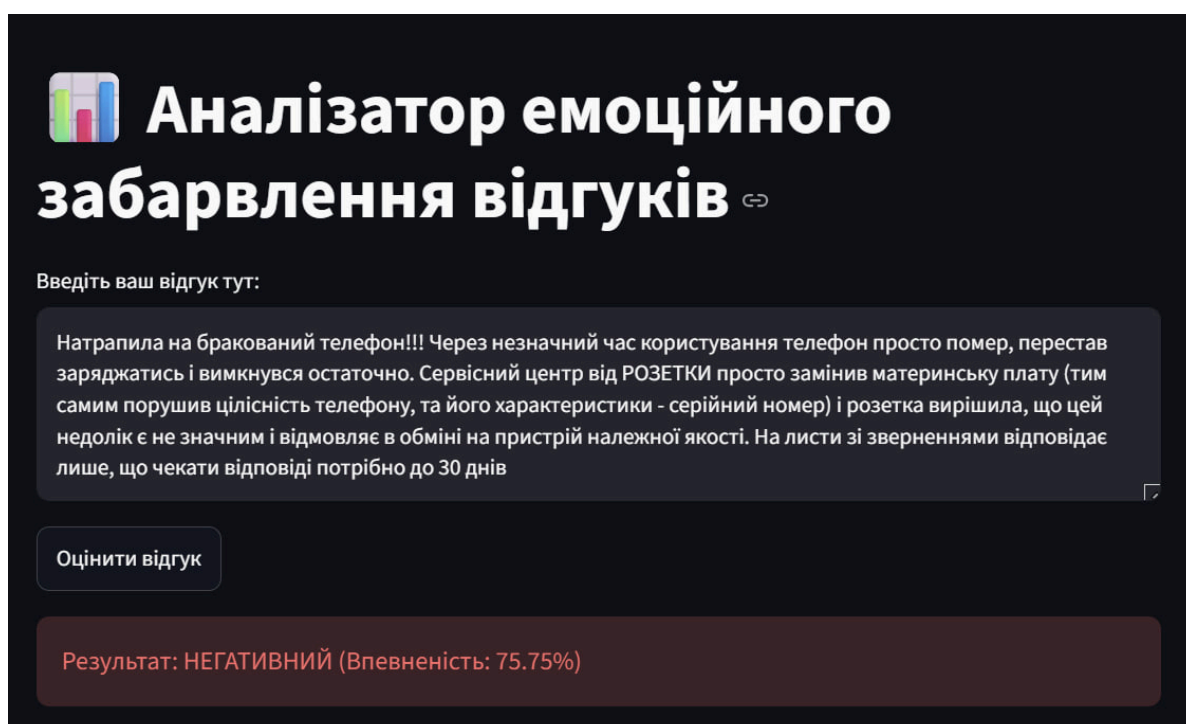


Рисунок 3.9 - Приклад відгуку користувача для подальшої класифікації

Як показано на рисунку 3.10, система успішно ідентифікувала тональність відгуку як «Negative». Процес обробки включав видалення пунктуації та лематизацію, після чого модель на основі ансамблю класифікаторів з високою впевненістю визначила емоційне забарвлення тексту. Цей приклад наочно демонструє, що інтелектуальне ядро здатне ефективно працювати зі складними, емоційно насиченими відгуками, що містять скарги на сервіс. Отримані результати підтверджують коректність функціонування розробленого програмного продукту та його готовність до практичного застосування в задачах моніторингу репутації та підтримки клієнтів.

### **Висновки до розділу 3**

У цьому розділі ми розробили та налаштували систему, яка автоматично визначає настрій відгуків користувачів. Робота розпочалася з очищення тексту від сміття та зайвих символів, що дозволило підготувати якісні дані для навчання. Для класифікації ми використали «ансамбль» – це коли три різні алгоритми (SVM, логістична регресія та випадковий ліс) працюють разом, щоб прийняти найбільш точне рішення. Такий підхід допоміг уникнути помилок, пов'язаних із нерівною кількістю позитивних та негативних відгуків.

За результатами тестів, модель показала високу точність у 87%. Особливо добре система виявляє негативні відгуки (F1-score 0.90), що критично важливо для оперативного реагування на скарги. Завдяки простому веб-інтерфейсу на базі Streamlit, користувач може миттєво перевірити будь-який текст. Отже, створена система є стабільною, надійною та повністю готовою до використання у реальній роботі для моніторингу репутації та спілкування з клієнтами.

## ВИСНОВКИ

У дипломній роботі було розроблено, реалізовано та експериментально оцінено інтелектуальну систему багатоміткової класифікації текстів, яка об'єднує сучасні методи обробки природної мови (NLP) та архітектуру на основі ансамблю класифікаторів. Основною метою дослідження було створення ефективного рішення, здатного автоматично класифікувати текстові дані за кількома категоріями одночасно, забезпечуючи високу якість навіть у випадках із незбалансованим розподілом міток.

Для досягнення мети було впроваджено комплексний підхід, що включає: ретельну попередню обробку текстових масивів, вибір оптимальних методів векторизації, побудову ансамблевої моделі класифікації та налаштування порогів прийняття рішень для мінімізації помилок прогнозування. Отримані результати підтвердили доцільність обраної архітектури: система продемонструвала високі метрики якості класифікації, ефективно справляючись як із типовими, так і з рідкісними категоріями.

Таким чином, робота робить вагомий внесок у практичне застосування методів машинного навчання в задачах автоматизації обробки текстової інформації. Запропоноване рішення є гнучким, масштабованим і може бути адаптовано для широкого кола прикладних задач – від інтелектуального пошуку та автоматичного сортування документів до систем підтримки прийняття рішень. Розроблена модель довела свою надійність, точність та здатність до узагальнення, що дозволяє розглядати її як конкурентоспроможний інструмент, готовий до інтеграції в реальні цифрові сервіси та бізнес-процеси.

## ПЕРЕЛІК ПОСИЛАНЬ

1. "Natural Language Processing (NLP)", URL:  
<https://www.ibm.com/topics/natural-language-processing> (дата звернення: 04.06.2026)
2. "Speech and Language Processing" (Jurafsky & Martin), URL:  
<https://web.stanford.edu/~jurafsky/slp3/> (дата звернення: 04.06.2026)
3. "Introduction to Information Retrieval" (Manning et al.), URL:  
<https://nlp.stanford.edu/IR-book/> (дата звернення: 04.06.2026)
4. "Ensemble Methods in Machine Learning", URL:  
[https://link.springer.com/chapter/10.1007/3-540-45014-9\\_1](https://link.springer.com/chapter/10.1007/3-540-45014-9_1) (дата звернення: 04.06.2026)
5. "Pattern Recognition and Machine Learning" (Bishop), URL:  
<https://www.microsoft.com/en-us/research/people/cmbishop/prml-book/> (дата звернення: 04.06.2026)
6. "Sentiment Analysis: Mining Opinions, Sentiments, and Emotions" (Bing Liu),  
URL:<https://www.cs.uic.edu/~liub/FBS/SentimentAnalysis-Mining-Opinions-Sentiments-Emotions.pdf> (дата звернення: 04.06.2026)
7. "Machine Learning in Python: Scikit-learn documentation", URL:  
<https://scikit-learn.org/stable/> (дата звернення: 04.06.2026)
8. "Importance of online customer reviews in e-commerce", URL:  
<https://www.statista.com/> (дата звернення: 04.06.2026)
9. "Understanding Natural Language Processing", URL:  
<https://aws.amazon.com/what-is/nlp/> (дата звернення: 04.06.2026)
10. "Text Mining and Sentiment Analysis", URL:  
<https://www.geeksforgeeks.org/text-mining-and-sentiment-analysis/> (дата звернення: 04.06.2026)

11. "Feature Engineering for Machine Learning" (Alice Zheng), URL: <https://www.oreilly.com/library/view/feature-engineering-for/9781491953239/> (дата звернення: 04.06.2026)
12. "Text Preprocessing for Sentiment Analysis", URL: <https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-nlp/> (дата звернення: 04.06.2026)
13. "Tf-idf and Text Vectorization methods", URL: [https://scikit-learn.org/stable/modules/feature\\_extraction.html](https://scikit-learn.org/stable/modules/feature_extraction.html) (дата звернення: 04.06.2026)
14. "Support Vector Machines for Text Classification", URL: [https://medium.com/@ml\\_guru/svm-for-text-classification-e3c3b53c1268](https://medium.com/@ml_guru/svm-for-text-classification-e3c3b53c1268) (дата звернення: 04.06.2026)
15. "Logistic Regression for Binary Classification", URL: <https://machinelearningmastery.com/logistic-regression-for-machine-learning/> (дата звернення: 04.06.2026)
16. "Ensemble Methods: Soft Voting Classifier", URL: <https://scikit-learn.org/stable/modules/ensemble.html#voting-classifier> (дата звернення: 04.06.2026)
17. "Evaluation Metrics for Classification", URL: <https://towardsdatascience.com/accuracy-precision-recall-or-f1-33fb33785199> (дата звернення: 04.06.2026)
18. "Confusion Matrix in Machine Learning", URL: <https://www.geeksforgeeks.org/confusion-matrix-machine-learning/> (дата звернення: 04.06.2026)
19. "Cross-Validation Techniques", URL: [https://scikit-learn.org/stable/modules/cross\\_validation.html](https://scikit-learn.org/stable/modules/cross_validation.html) (дата звернення: 04.06.2026)
20. "N-grams and Contextual Analysis", URL: <https://www.kaggle.com/code/adityagole/n-grams-and-text-analysis> (дата звернення: 04.06.2026)

## ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

vectorizer.py:

```
from sklearn.feature_extraction.text import TfidfVectorizer

def get_vectorizer():
    # Оптимальний ngram_range для вашого препроцесингу
    return TfidfVectorizer(ngram_range=(1, 3), min_df=2, max_features=5000)
```

models.py:

```
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import VotingClassifier

def get_model():
    # Повертаємо класичний VotingClassifier, який ви хотіли
    model = VotingClassifier(
        estimators=[
            ('svm', SVC(probability=True, class_weight='balanced')),
            ('lr', LogisticRegression(class_weight='balanced'))
        ],
        voting='soft'
    )
    return model
```

preprocessor.py:

```
import pymorphy3
import re

class TextPreprocessor:
    def __init__(self):
        # Ініціалізуємо аналізатор для української мови
        self.morph = pymorphy3.MorphAnalyzer(lang='uk')

        # Словник для заміни емотиконів та емодзі на текстові маркери
        # Це дозволяє моделі "бачити" емоційне забарвлення смайликів
        self.emoji_map = {
            r':\)+': ' позитивний ', r':-\)+': ' позитивний ',
            r':D+': ' позитивний ', r'xd+': ' позитивний ',
            r':\(+': ' негативний ', r':-\(+': ' негативний ',
            r';\(+': ' негативний ', r':\/*': ' негативний ',
            r'👍': ' позитивний ', r'👎': ' негативний ',
            r'😡': ' негативний ', r'😊': ' позитивний ',
            r'❤️': ' негативний ', r'✨': ' позитивний ',
            r'😂': ' позитивний ', r'😞': ' негативний '
        }

    def clean_text(self, text):
        """
        Метод очищення: замінює смайлики на текст
        та видаляє все, крім кирилических літер.
        """
        text = str(text).lower()

        # 1. Заміна смайликів на слова
```

```

    for pattern, replacement in self.emoji_map.items():
        text = re.sub(pattern, replacement, text)

    # 2. Видалення всіх символів, крім кирилических літер та пробілів
    text = re.sub(r'^а-яіієє\r\s', '', text)

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

def lemmatize(self, text):
    """
    Приводить кожне слово до словникової форми (леми).
    """
    words = text.split()
    lemmas = [self.morph.parse(word)[0].normal_form for word in words]
    return " ".join(lemmas)

def process(self, df, column_name):
    """
    Основний метод конвеєра: очищення -> лематизація.
    """
    print(f"Початок обробки колонки: {column_name}")

    # Застосовуємо очистку, потім лематизацію
    df['cleaned'] = df[column_name].apply(self.clean_text)
    df['lemmas'] = df['cleaned'].apply(self.lemmatize)

    print("Обробку завершено.")
    return df

# Приклад тестування (можна запустити окремо для перевірки):
if __name__ == "__main__":
    import pandas as pd

    data = {'text': ["Це найкращий сервіс! :)", "Жахливо, нікому не раджу :(("]}
    df = pd.DataFrame(data)

    preprocessor = TextPreprocessor()
    df = preprocessor.process(df, 'text')
    print(df[['text', 'lemmas']])

```

app.py:

```

import streamlit as st
import joblib
import pandas as pd
import matplotlib.pyplot as plt
from preprocessor import TextPreprocessor

# 1. Завантаження моделі та інструментів
@st.cache_resource
def load_assets():
    model = joblib.load('model.pkl')
    vectorizer = joblib.load('vectorizer.pkl')
    preprocessor = TextPreprocessor()

```

```

    return model, vectorizer, preprocessor

model, vectorizer, preprocessor = load_assets()

# 2. Інтерфейс
st.set_page_config(page_title="Аналізатор відгуків", page_icon="📊")
st.title("📊 Аналізатор емоційного забарвлення відгуків")

# 3. Блок введення
user_input = st.text_area("Введіть ваш відгук тут:", height=150)

if st.button("Оцінити відгук"):
    if user_input.strip() == "":
        st.warning("Будь ласка, введіть текст для аналізу!")
    else:
        # Обробка та векторизація
        processed_text =
preprocessor.lemmatize(preprocessor.clean_text(user_input))
        text_vec = vectorizer.transform([processed_text])

        # Отримання ймовірностей
        probs = model.predict_proba(text_vec)[0]
        prob_neg, prob_pos = probs[0], probs[1]

        # --- ЛОГІКА ВИЗНАЧЕННЯ КЛАСУ ---
        # Якщо модель каже, що 1 це позитив, але виводить "негатив",
        # просто змініть цей прапорець на False
        is_model_inverted = False

        is_positive = (prob_pos >= 0.5) if not is_model_inverted else (prob_pos <
0.5)

        # Виведення результату
        if is_positive:
            st.success(f"Результат: ПОЗИТИВНИЙ (Впевненість: {prob_pos:.2%})")
        else:
            st.error(f"Результат: НЕГАТИВНИЙ (Впевненість: {prob_neg:.2%})")

        # Технічні деталі
        with st.expander("Технічні деталі класифікації"):
            st.write(f"Ймовірність негативу: {prob_neg:.2%}")
            st.write(f"Ймовірність позитиву: {prob_pos:.2%}")

            fig, ax = plt.subplots(figsize=(6, 2))
            ax.barh(['Негатив', 'Позитив'], [prob_neg, prob_pos], color=['#FF4B4B',
'#00CC96'])
            st.pyplot(fig)

# 4. Аналітика
st.divider()
st.subheader("Якість та аналітика моделі")

if st.checkbox("Показати матрицю помилок"):
    st.image('confusion_matrix.png')

if st.checkbox("Показати Classification Report"):
    try:
        with open('report.txt', 'r', encoding='utf-8') as f:

```

```

        st.code(f.read())
    except:
        st.error("Файл 'report.txt' не знайдено.")

if st.checkbox("Показати розподіл навчальних даних"):
    try:
        df = pd.read_csv('review-data.csv')
        fig2, ax2 = plt.subplots()
        df['target'].value_counts().plot(kind='pie', autopct='%1.1f%%', ax=ax2,
colors=['#00CC96', '#FF4B4B'])
        ax2.set_ylabel('')
        st.pyplot(fig2)
    except:
        st.error("Файл 'review-data.csv' не знайдено.")

```

[main.py](#):

```

import pandas as pd
import joblib
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.ensemble import VotingClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression

# Припускаємо, що ваші файли preprocessor.py та models.py існують у тій же папці
from preprocessor import TextPreprocessor
from vectorizer import get_vectorizer

# 1. Завантаження даних
df = pd.read_csv('review-data.csv', encoding='utf-8')

# 2. Обробка
preprocessor = TextPreprocessor()
df = preprocessor.process(df, 'text')

# 3. Розподіл
X_train, X_test, y_train, y_test = train_test_split(df['lemmas'], df['target'],
test_size=0.2, random_state=42)

# 4. Векторизація
vectorizer = get_vectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# 5. Навчання ансамблю
clf1 = LogisticRegression(class_weight='balanced', max_iter=1000)
clf2 = SVC(probability=True, class_weight='balanced')
model = VotingClassifier(estimators=[('lr', clf1), ('svc', clf2)], voting='soft')

model.fit(X_train_vec, y_train)

# 6. Оцінка та збереження результатів
y_pred = model.predict(X_test_vec)

# Звіт у консоль та файл
report = classification_report(y_test, y_pred)

```

```
print("\n--- Звіт про класифікацію ---\n")
print(report)

with open('report.txt', 'w', encoding='utf-8') as f:
    f.write(report)

# Матриця помилок
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Матриця помилок (Confusion Matrix)')
plt.savefig('confusion_matrix.png')

# 7. Збереження моделі
joblib.dump(model, 'model.pkl')
joblib.dump(vectorizer, 'vectorizer.pkl')
print("Успішно! Модель, векторизатор та звіти збережені.")
```