МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____»_____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА АНАЛІЗУ ВІДГУКІВ КОРИСТУВАЧІВ ІЗ ВИЯВЛЕННЯМ ЕМОЦІЙНИХ СТАНІВ ТА ДИНАМІКИ НАСТРОЇВ НА ОСНОВІ NLP

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувач _____ Віталій КРАЙЧИЙ

«___»_____ 2026 р.

Керівник д-р техн. наук, професор _____ Олексій КОЗЛОВ

«___»_____ 2026 р.

Миколаїв – 2026

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____» _____ 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Крайчого Віталія Олександровича
((прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Інтелектуальна система аналізу відгуків користувачів із виявленням емоційних станів та динаміки настроїв на основі NLP».
Керівник роботи: Козлов Олексій Валерійович д-р техн. наук, професор.
Затверджена наказом ЧНУ ім. Петра Могили від «25» грудня 2025 р. № 353.
2. Строк представлення кваліфікаційної роботи «___» _____ 2026 р.
3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: інтелектуальна система аналізу відгуків користувачів із виявленням емоційних станів та динаміки настроїв на основі NLP; текстові відгуки користувачів у форматі CSV з часовими мітками, розмічена навчальна вибірка емоційних станів, критерії оцінювання якості системи, пріоритетність критеріїв.

4. Перелік питань, що підлягають розробці: аналіз сучасного стану задачі автоматизованого аналізу емоційного забарвлення текстових відгуків користувачів; огляд існуючих методів обробки природної мови (токенізація, лематизація, видалення стоп-слів, векторизація); огляд та порівняльний аналіз моделей машинного навчання для класифікації емоцій (логістична регресія, випадковий ліс, нейронні мережі); експертне оцінювання якості класифікації за визначеними метриками; порівняльний аналіз результатів застосування обраних моделей машинного навчання для розв'язання поставленої задачі.

5. Перелік графічних матеріалів: презентація.

Керівник роботи _____ Олексій КОЗЛОВ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Віталій КРАЙЧИЙ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «23» грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Інтелектуальна система аналізу відгуків користувачів із виявленням емоційних станів та динаміки настроїв на основі NLP

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	21.12.2025	24.12.2025	Виконано
2	Аналіз предметної області та постановка задачі	25.12.2025	30.01.2026	Виконано
3	Огляд літературних джерел за темою кваліфікаційної роботи, зокрема огляд публікацій та аналогічних систем, щодо інтелектуальних систем відгуків користувачів	31.01.2026	01.03.2026	Виконано
4	Огляд існуючих методів обробки природної мови (NLP) та моделей машинного навчання для класифікації емоцій	02.03.2026	01.04.2026	Виконано
5	Формулювання вимог до системи, вибір технологій та розробка архітектури	02.04.2026	24.05.2026	Виконано
6	Реалізація обраних технологій з аналізом отриманих результатів	25.05.2026	25.05.2026	Виконано
7	Корегування роботи за результатами попереднього захисту та розробка програмної реалізації	26.05.2026	04.06.2026	Виконано
8	Другий попередній захист КР на засіданні комісії кафедри	05.06.2026	05.06.2026	Виконано
9	Доробка та остаточне оформлення КР	06.06.2026	14.06.2026	Виконано
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	15.06.2026	19.06.2026	Виконано

Керівник роботи _____ Олексій КОЗЛОВ

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Віталій КРАЙЧИЙ

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану

«29» січня 2026

АНОТАЦІЯ

до кваліфікаційної роботи

здобувача групи 401 ЧНУ ім. Петра Могили

Крайчого Віталія Олександровича

на тему: **“ІНТЕЛЕКТУАЛЬНА СИСТЕМА АНАЛІЗУ ВІДГУКІВ КОРИСТУВАЧІВ ІЗ ВИЯВЛЕННЯМ ЕМОЦІЙНИХ СТАНІВ ТА ДИНАМІКИ НАСТРОЇВ НА ОСНОВІ NLP”**

Актуальність даної роботи полягає у необхідності автоматизації аналізу великих обсягів текстових відгуків користувачів, що дозволить виявляти емоційні стани (радість, злість, розчарування, подив, нейтральність) та відстежувати динаміку настроїв у часі. Застосування методів обробки природної мови (NLP) та машинного навчання забезпечує ефективне розв'язання цього завдання.

Об'єктом роботи є процеси аналізу текстових відгуків користувачів у форматі CSV.

Предметом роботи є методи та моделі машинного навчання для класифікації емоційних станів, а також технології попередньої обробки текстів. Дослідження виконується в рамках парадигми навчання з учителем, де для кожної вхідної текстової одиниці задається відповідна емоційна мітка.

Метою є розробка інтелектуальної системи для автоматизованого аналізу текстових відгуків користувачів, яка дозволяє визначати емоційне забарвлення текстів та візуалізувати динаміку настроїв у часі. Впровадження системи дозволить скоротити час аналізу великих масивів відгуків порівняно з ручним опрацюванням та забезпечить автоматичне виявлення емоційних патернів, що сприятиме більш об'єктивному прийняттю рішень на основі зворотного зв'язку користувачів.

В результаті виконання роботи було досліджено три моделі класифікації емоцій (логістичну регресію, випадковий ліс та нейронні мережі), проведено їх порівняльний

аналіз за метриками точності, повноти, прецизійності та F1-міри, визначено їх переваги та недоліки, а також розроблено програмне забезпечення, в якому реалізовано відповідні методи та веб-інтерфейс для взаємодії з користувачем. Дана робота складається з чотирьох розділів. У першому розділі проведено аналіз предметної області, огляд особливостей текстових відгуків, сформовано постановку задачі та вимоги до системи. Другий розділ присвячений вибору засобів розробки, архітектурі системи та інтерфейсу користувача. У третьому розділі наведено дослідження методів та моделей класифікації емоцій, аналіз аналогів та обґрунтування доцільності розробки. У четвертому – програмна реалізація системи, тестування та аналіз отриманих результатів. Загальний обсяг роботи – 98 сторінок. Кваліфікаційна робота містить 1 додатків, 26 рисунків, 1 таблиць і 25 джерел посилання.

Ключові слова: аналіз емоційних станів, динаміка настроїв, обробка природної мови (NLP), класифікація текстів, логістична регресія, випадковий ліс, нейронні мережі, TF-IDF, векторизація, токенізація, лематизація.

ABSTRACT

to the qualification work by the student of the group 401 of Petro Mohyla Black Sea National University

Kraichyi Vitalii

“INTELLIGENT SYSTEM FOR USER REVIEW ANALYSIS WITH DETECTION OF EMOTIONAL STATES AND SENTIMENT DYNAMICS BASED ON NLP”

The relevance of this work lies in the need to automate the analysis of large volumes of user text reviews, which allows identifying emotional states (joy, anger, disappointment, surprise, neutrality) and tracking sentiment dynamics over time. The application of natural language processing (NLP) and machine learning methods ensures effective solving of this problem.

The object of the work is analysis processes of user text reviews in CSV format.

The subject of the study is methods and models of machine learning for the classification of emotional states, as well as text preprocessing technologies. The research is carried out within the framework of the supervised learning paradigm, where each input text unit is assigned a corresponding emotional label.

The aim of the work is to develop an intelligent system for automated analysis of user text reviews, which allows determining the emotional coloring of texts and visualizing sentiment dynamics over time. The implementation of the system will reduce the analysis time of large arrays of reviews compared to manual processing and will ensure automatic detection of emotional patterns, which will contribute to more objective decision-making based on user feedback.

As a result of the work, three emotion classification models (logistic regression, random forest, and neural networks) were studied, their comparative analysis was carried out according to the metrics of accuracy, recall, precision, and F1-score, their advantages

and disadvantages were identified, and software was developed that implements the corresponding methods and a web interface for user interaction

This work consists of four sections. The first section analyzes the subject area, reviews the features of text reviews, formulates the problem statement and system requirements. The second section is devoted to the selection of development tools, system architecture, and user interface. The third section presents the study of methods and models of emotion classification, analysis of analogues, and justification of the feasibility of development. The fourth section presents the software implementation of the system, testing, and analysis of the obtained results. The total volume of the work is 98 pages. The qualification work contains 1 appendices, 26 figures, 1 tables, and 25 references.

Keywords: emotional state analysis, sentiment dynamics, natural language processing (NLP), text classification, logistic regression, random forest, neural networks, TF-IDF, vectorization, tokenization, lemmatization.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

NLP — Natural Language Processing

ML — Machine Learning

TF-IDF — Term Frequency – Inverse Document Frequency

CSV — Comma-Separated Values

ШІ — Штучний інтелект

ЛР — Логістична регресія

ВЛ — Випадковий ліс (Random Forest)

НМ — Нейронні мережі

NLTK — Natural Language Toolkit

PCA — Principal Component Analysis

API — Application Programming Interface

GUI — Graphical User Interface

HTML — HyperText Markup Language

CSS — Cascading Style Sheets

URL — Uniform Resource Locator

CNN — Convolutional Neural Network

RNN — Recurrent Neural Network

LSTM — Long Short-Term Memory

ReLU — Rectified Linear Unit

IDE — Integrated Development Environment

PCA — Principal Component Analysis

ВСТУП

Сучасний розвиток цифрових комунікацій зумовлює стрімке збільшення текстових відгуків від користувачів щодо товарів, послуг та програмних продуктів. Особливої актуальності набуває завдання автоматизованого аналізу цих даних з метою виявлення емоційного стану авторів та визначення динаміки їхніх налаштувань. Використання методів обробки природної мови (NLP) дозволяє ефективно обробляти дані, перетворюючи неструктуровані текстові повідомлення на структуровані дані для подальшого прийняття рішень.

Метою кваліфікаційної роботи є розробка інтелектуальної системи для автоматизованого аналізу текстових відгуків користувачів, яка дозволяє визначати емоційне забарвлення текстів та візуалізувати динаміку настроїв у часі. Впровадження системи дозволить скоротити час аналізу великих масивів відгуків порівняно з ручним опрацюванням та забезпечить автоматичне виявлення емоційних патернів, що сприятиме більш об’єктивному прийняттю рішень на основі зворотного зв’язку користувачів.

Об’єктом дослідження в межах роботи виступають процеси аналізу текстових відгуків користувачів у форматі CSV, а предметом – методи та моделі машинного навчання для класифікації емоційних станів, а також технології попередньої обробки текстів. Дослідження виконується в рамках парадигми навчання з учителем, де для кожної вхідної текстової одиниці задається відповідна емоційна мітка.

Під час роботи було реалізовано програмний засіб мовою Python із використанням бібліотек NLTK, scikit-learn, pandas, matplotlib та Flask. Система забезпечує завантаження даних, автоматичне визначення емоційного стану кожного запису, побудову статистичних графіків розподілу емоцій та візуалізацію динаміки настроїв користувачів.

У результаті кваліфікаційної роботи було створено діючий прототип системи, який може бути використаний для аналізу зворотного зв’язку в різних

сферах – від e-commerce до соціальних досліджень. Успішна реалізація цього модуля підтверджує доцільність обраного підходу.

Декларація про використання ШІ. Під час підготовки наукової роботи (академічного тексту) було використано інструмент ChatGPT-5. Відповідно до таксономії GAIDeT (2025), наведені нижче завдання були делеговані інструментам генеративного ШІ за повного людського нагляду:

– Оцінювання здійсненності та ризиків
– Оптимізація коду
– Перевірка відтворюваності
– Вичитування та редагування
– Резюмування тексту
– Адаптація та коригування емоційного тону
– Оцінювання якості
– Рекомендації

Повну відповідальність за фінальний рукопис несуть автор.

Інструменти генеративного ШІ не зазначаються як автори та не несуть відповідальності за кінцеві результати.

Декларацію подав: Крайчий Віталій Олександрович

1 ВИМОГИ ДО СИСТЕМИ ТА АНАЛІЗ ПРОБЛЕМИ АНАЛІЗУ ВІДГУКІВ

1.1 Постановка задачі кваліфікаційної роботи

Сучасний інформаційний простір перенасичений текстовими відгуками користувачів - про товари, послуги, програмне забезпечення, події чи навіть лікарські препарати. Щодня з’являються мільйони повідомлень, коментарів та рецензій, які містять цінну інформацію про емоційне ставлення людей до різноманітних об'єктів. Проте ручний аналіз таких обсягів даних буде дуже незручним - він потребує значних часових і людських ресурсів, а суб’єктивність оцінки окремого дослідника може суттєво впливати на кінцеві результати. Саме тому створення інтелектуальних систем, здатних автоматично обробляти текстові відгуки, виявляти приховані емоційні стани та відстежувати зміну настроїв авторів у часі, набуває дедалі більшої актуальності.

У межах даної кваліфікаційної роботи ставиться завдання розробити інтелектуальну систему, яка на основі сучасних технологій обробки природної мови (NLP) забезпечувала б автоматизоване виявлення емоційного забарвлення текстів. Причому йдеться не просто примітивний поділ на «позитив» і «негатив», а більш детальну інформацію, як емоційні стани, як радість, злість, розчарування, подив, смуток або нейтральність. Окрім того, важливою функцією системи є можливість аналізувати динаміку настроїв у часі - тобто бачити, як змінювалося емоційне коло користувачів протягом певного періоду, чи з’являються спалахи негативу після певних подій, чи, навпаки, зростає кількість позитивних оцінок.

Щоб досягти цієї мети, система має вирішувати кілька взаємопов’язаних завдань. Насамперед необхідно забезпечити якісну попередню обробку текстових даних. Це робиться тому що сирі відгуки часто містять зайві символи, граматичні помилки, стоп-слова та різні словоформи, які заважають коректному аналізу. Тому в роботі передбачається застосування таких технік, як токенізація (тобто розбиття тексту на окремі слова або токени), лематизація (приведення слів до їхньої початкової словникової форми), видалення стоп-слів (слів, які не несуть

особливого смислового навантаження, наприклад «і», «але», «або» тощо) та векторизація (перетворення текстів на числові вектори, з якими вже можуть працювати алгоритми машинного навчання).

Після того як дані оброблено, система має застосовувати одну або кілька моделей машинного навчання для класифікації емоцій. У рамках цієї роботи передбачається порівняльний аналіз декількох підходів - зокрема, логістичної регресії [10], випадкового лісу (Random Forest [11]) та нейронних мереж. Кожна з цих моделей має свої переваги та недоліки. Одні краще працюють з невеликими обсягами даних, інші потребують значних обчислювальних ресурсів, але водночас здатні вловлювати складні нелінійні залежності в текстах. Яка саме модель виявиться найкращою - буде вирішено на основі експериментів з реальними даними.

Але класифікація - це ще не все. Система також повинна надавати результати аналізу в зрозумілій та наочній формі. Тобто програмний засіб повинен вміти будувати статистичні графіки розподілу емоцій серед проаналізованих відгуків, а також візуалізувати динаміку настроїв у часі - наприклад, у вигляді лінійних графіків або теплових карт. Завдяки такій візуалізації можна швидко оцінити загальну картину емоційного стану аудиторії, помітити аномалії або критичні періоди, коли кількість негативних відгуків різко зростає.

Також важливою вимогою до системи є зручність використання. Користувач не повинен писати програмний код або працювати зі складними командними рядками. Тому передбачається створення веб-інтерфейсу з використанням фреймворку Flask, через який користувач зможе завантажити набір відгуків у форматі CSV, запустити аналіз та отримати результати у вигляді інтерактивних графіків і звітів. Уся система реалізована на мові програмування Python із використанням спеціалізованих бібліотек – NLTK [2] для обробки природної мови, scikit-learn [3] для моделей машинного навчання, pandas [4] для маніпуляції даними, matplotlib [5] для візуалізації та Flask [6] для веб-частини.

Отже, поставлена задача охоплює весь цикл створення інтелектуальної системи - починаючи від збору та очищення текстових даних, продовжуючи застосуванням методів NLP та машинного навчання, і завершуючи поданням користувачеві наочних результатів у зручному веб-інтерфейсі. Успішне вирішення цього завдання дасть змогу автоматизувати аналіз великих масивів текстових відгуків, виявляти приховані емоційні патерни та відстежувати, як змінюються настрої користувачів у реальних умовах.

1.2 Аналіз предметної галузі

1.2.1 Особливості текстових відгуків користувачів як об'єкта аналізу емоційного стану

Текстові відгуки користувачів, є унікальним джерелом даних, яке поєднує в собі фактичну інформацію та суб’єктивне ставлення автора до предмета обговорення, незалежно від того, чи стосуються вони лікарських препаратів, побутової техніки, програмного забезпечення чи сервісних послуг. Саме це робить їх дуже цінним матеріалом для аналізу, але також створює значні труднощі при спробах автоматизованої обробки. Для системи, яка має виявляти емоційні стани та відстежувати динаміку настроїв, розуміння природи таких текстів є критично важливим, оскільки саме від цих особливостей вхідних даних залежить вибір методів попередньої обробки, архітектура моделей машинного навчання та очікувана точність класифікації.

Першою й найочевиднішою особливістю текстових відгуків є їхня неструктурованість. На відміну від формалізованих даних, які зберігаються в базах даних у вигляді таблиць із чітко визначеними полями, відгуки - це довільні тексти, написані природною мовою. І жоден користувач не дотримується тут якоїсь строгої структури. Хтось починає з позитивних моментів, хтось одразу виливає своє невдоволення, а хтось пише довгі розповіді з передісторією. До того ж, довжина відгуків може бути різною - від одного короткого речення до кількох абзаців. Така

різноманітність, звісно, ускладнює застосування стандартних підходів до аналізу, адже алгоритми повинні бути гнучкими та здатними працювати з текстами різної структури та обсягу.

Друга важлива характеристика – суб'єктивність. Кожен відгук відображає особистий досвід, емоції, очікування та вподобання конкретної людини. І те, що для одного користувача є цілком прийнятним або навіть чудовим, для іншого може буде абсолютно неприйнятним. Більше того, один і той самий текст може містити змішані емоції. Наприклад, спочатку людина радіє від певної властивості продукту, а потім розчаровується через іншу. Така емоційна неоднорідність створює додаткові виклики для систем класифікації, оскільки іноді важко однозначно віднести весь відгук до однієї емоційної категорії. У подібних випадках навіть людина може вагатися між «нейтральним» та «змішаним» станом, не кажучи вже про автоматичний алгоритм.

Таблиця 1.1 - Приклади неоднозначних текстових відгуків та труднощі їх класифікації

Текст відгуку	Можливі емоції (на думку різних людей)	Чому виникає плутанина
«Ну, це просто щось неймовірне...»	Подив / Радість / Сарказм (розчарування)	Залежить від контексту та інтонації, відсутні чіткі маркери
«Я в шоці, такого не очікував»	Подив / Розчарування / Радість	Слово «шок» може бути як позитивним, так і негативним
«Дякую, ви мене здивували»	Радість / Іронія	Фраза ввічлива за формою, але може бути прихованим невдоволенням
«Норм, але могло бути краще»	Нейтральне/Легке розчарування	Суперечливі оцінки в одному реченні

Кінець таблиці 1.1

«Супер!!!!!... ні, я жартую»	Радість / Розчарування	Зміна емоції в межах одного відгуку

Наступною особливістю є постійне зростання обсягів текстових даних. Чим популярнішою є певна онлайн-платформа або чим більше людей користується певним продуктом, тим більше відгуків генерується кожен день. У деяких сферах, таких як фармацевтиці або електронній комерції, кількість нових відгуків може нараховувати тисячі або навіть десятки тисяч щомісяця. Ручний аналіз таких обсягів стає практично неможливим - навіть команда аналітиків не зможе опрацювати всі дані оперативно та об’єктивно. Саме тут автоматизація набуває колосального значення, адже інтелектуальна система здатна обробляти величезні масиви текстів за лічені хвилини або години.

Окремо варто згадати про лінгвістичну різноманітність. Тексти відгуків пишуть звичайні люди, а не професійні редактори або лінгвісти. Тому в них часто трапляються орфографічні та пунктуаційні помилки, розмовні вирази, сленг, скорочення (наприклад, «норм», «супер», «не очікував»), емоційні знаки (декілька знаків оклику чи питання поспіль), а також слова, написані ВЕЛИКИМИ літерами для посилення емоційного забарвлення. У деяких відгуках використовуються сарказм або іронія, які навіть для людини іноді важко розпізнати без контексту. Усе це створює значні проблеми для стандартних NLP-методів, які зазвичай розраховують на граматично коректні тексти. Саме тому в системах аналізу емоцій велика увага приділяється етапу попередньої обробки - очищенню текстів, нормалізації, приведенню слів до початкових форм, а іноді навіть спеціальній обробці емодзі та інших невербальних елементів, які також можуть нести емоційне навантаження. На рисунку 1.1 продемонстровано емоційне коло (Рис. 1.1).

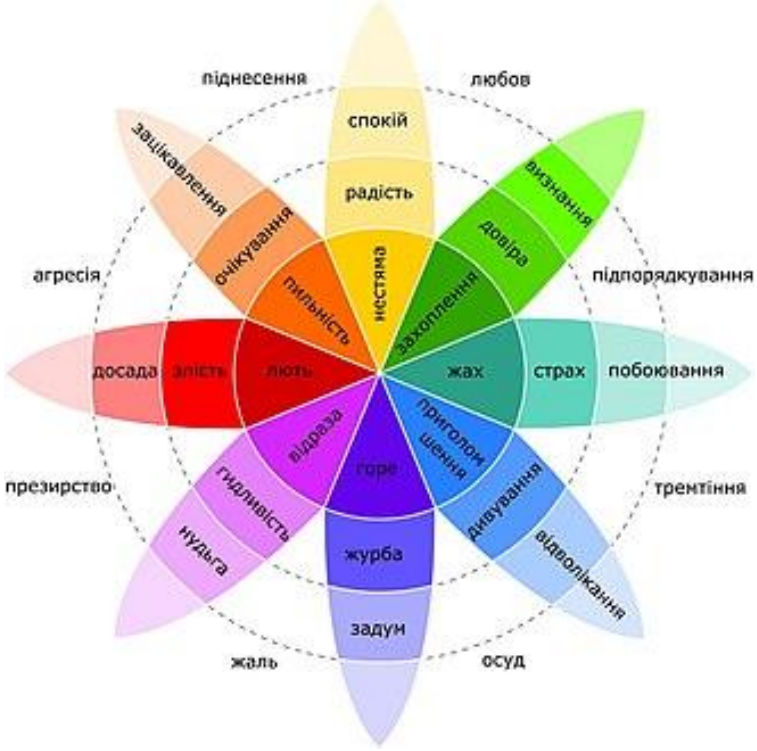


Рисунок 1.1 – Емоційне коло [14]

Таким чином, текстові відгуки користувачів - це складний і, багатогранний об’єкт аналізу, який поєднує риси неструктурованості, суб’єктивності, великих обсягів та лінгвістичної неоднорідності. Кожна з цих особливостей окремо вже створює труднощі для автоматизованої обробки, а разом вони вимагають комплексного підходу. Тобто потрібно поєднувати і потужні методи обробки природної мови, і гнучкі моделі машинного навчання. Саме цей комплексний підхід і має бути реалізований у системі, яка розробляється, щоб забезпечити якісне виявлення емоційних станів та достовірне відстеження динаміки настроїв користувачів.

1.2.2 Застосування методів машинного навчання для виявлення емоційних станів

У сучасній обробці природної мови для автоматичного визначення емоційного забарвлення текстових відгуків найчастіше використовуються методи класифікації з учителем. Це означає, що модель спочатку навчають на розмічених

прикладах - тобто на таких, де вже відомо, яка саме емоція відповідає тексту. А вже пісня навчання вона може самостійно аналізувати нові, невідомі раніше відгуки.

Першим кроком у цьому процесі є перетворення текстів на числові вектори, бо алгоритми машинного навчання працюють виключно з числами. Найпоширеніший підхід тут - метод TF-IDF [9]. Він оцінює важливість кожного слова в межах конкретного відгуку та в усьому корпусі текстів. Завдяки цьому слова, які часто трапляються в одній емоційній категорії (наприклад, «жахливо», «розчарований» для негативу, чи «чудово» або «задоволений» для позитиву), отримують більшу вагу.

Після векторизації до даних застосовуються моделі класифікації. У даній роботі розглядаються три типи моделей. Логістична регресія - проста й швидка. Випадковий ліс - ансамблевий метод, стійкий до переобучення. І нейронні мережі - найпотужніші, але потребують більше ресурсів. Останні особливо ефективні для виявлення складних лінгвістичних явищ, таких як сарказм або іронія.

Але система має не лише класифікувати окремий відгук. Вона також повинна відстежувати зміни настроїв у часі. Для цього кожен відгук супроводжується датою публікації. А після того, як емоції визначено, будуються графіки динаміки. Вони показують, як змінювалося співвідношення позитивних, негативних та нейтральних емоцій протягом певного періоду.

Таким чином, поєднання векторизації текстів, класифікаційних моделей та часового аналізу дозволяє створити систему, що не лише розпізнає емоції в кожному відгуку, але й дає змогу бачити загальну картину зміни настроїв користувачів.

1.2.3 Огляд методів класифікації для виявлення емоційних станів

У цій роботі для автоматичного розпізнавання емоцій у текстових відгуках використовуються методи класифікації з учителем. На відміну від кластеризації, яка шукає приховані групи без попередніх міток, класифікація потребує навчальної

вибірки, де кожен відгук уже позначений певною емоцією. Після навчання модель здатна самостійно визначати емоцію в будь-якому новому тексті.

Серед багатьох існуючих алгоритмів класифікації для цієї роботи було обрано три. Саме ці алгоритми найчастіше застосовуються в задачах аналізу тональності та емоцій завдяки поєднанню ефективності та відносної простоти реалізації.

Логістична регресія, є одним із найкращих базових алгоритмів для текстових задач. Вона швидко навчається, добре працює з високорозмірними розрідженими даними (наприклад з векторами TF-IDF) і дає зрозумілі ймовірності належності до кожної емоційної категорії. Головний її недолік у тому, що вона будує лише лінійні межі між класами, тому може не вловлювати складні залежності в текстах.

Випадковий ліс належить до ансамблевих методів. Він будує одразу багато дерев рішень, і кожне з яких навчається на випадковій підмножині слів-ознак. А потім усереднює їхні передбачення. Завдяки такому підходу значно знижується ризик перенавчання, і модель починає враховувати взаємодії між словами. На практиці випадковий ліс часто показує кращі результати, ніж логістична регресія, особливо коли в даних є нелінійні патерни. На рисунку 1.2 зображено метод Бегінга, один з методів машинного навчання (Рис. 1.2).

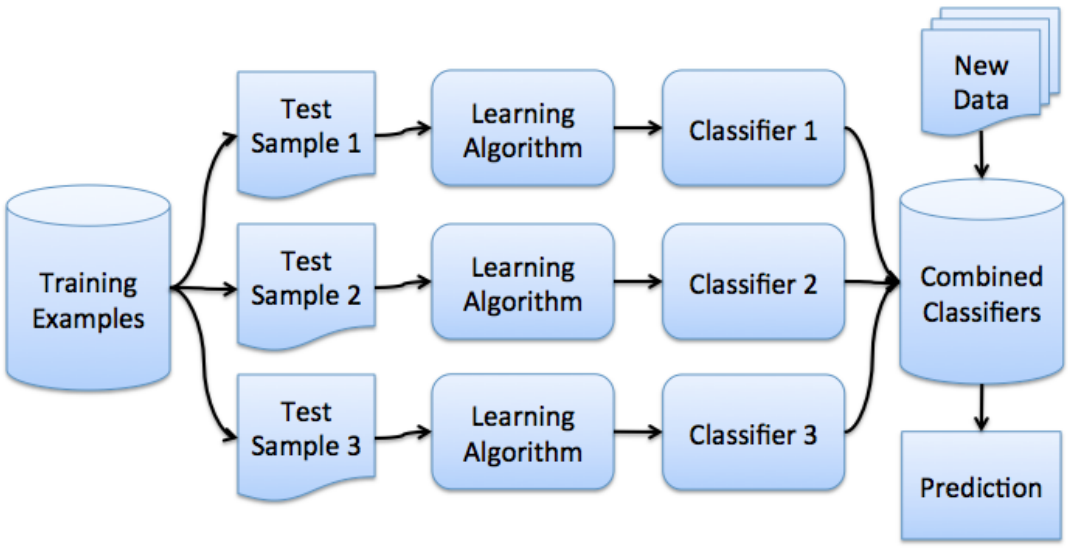


Рисунок 1.2 – Метод Бегінга [13]

Найскладніший, але водночас і найбільш потужний підхід - це використання нейронних мереж, зокрема рекурентних (LSTM) або згорткових (CNN) архітектур. Вони здатні враховувати порядок слів у реченні, що критично важливо для розуміння змісту. Наприклад, фрази «не погано» та «просто погано» мають зовсім протилежне значення. Крім того, нейронні мережі краще за інші методи справляються з сарказмом, іронією та змішаними емоціями - а це дуже часто зустрічаються в текстових відгуках реальних користувачів.

Усі три методи будуть реалізовані в рамках розроблюваної системи, навчені на одних і тих самих даних, а потім порівняні за точністю класифікації. Найкраща модель буде обрана для фінального варіанта системи аналізу емоцій та динаміки настроїв.

1.3 Вимоги до розроблюваної системи

1.3.1 Функціональні вимоги

Розроблювана інтелектуальна система призначена для автоматизованого аналізу текстових відгуків користувачів з метою виявлення емоційних станів та відстеження динаміки настроїв. Для забезпечення поставленої мети система має реалізовувати низку функціональних вимог, які охоплюють повний цикл обробки даних - від завантаження до візуалізації результатів.

Насамперед система повинна підтримувати завантаження текстових даних у популярних форматах, зокрема CSV та TXT. Це дозволяє користувачеві використовувати вже наявні набори відгуків без необхідності додаткового конвертування або ручного введення. Крім того, бажано забезпечити можливість завантаження даних безпосередньо через веб-інтерфейс шляхом вибору файлу на локальному пристрої.

Після завантаження даних система має автоматично виконувати попередню обробку текстів, яка є критично важливою для якості подальшої класифікації. Цей етап включає видалення зайвих символів (знаків пунктуації, спеціальних символів,

HTML-тегів), приведення всіх літер до одного регістру, нормалізацію слів (лематизацію або стемінг) та видалення стоп-слів - загальновживаних слів, які не несуть смислового чи емоційного навантаження.

Далі система повинна застосовувати методи векторизації для перетворення очищених текстів на числові вектори, придатні для роботи алгоритмів машинного навчання. Після цього виконується класифікація емоційного стану кожного відгуку з використанням попередньо навчених моделей - логістичної регресії, випадкового лісу або нейронної мережі. Результатом класифікації має бути присвоєння кожному відгуку однієї з наперед визначених емоційних категорій (радість, злість, розчарування, подив, смуток або нейтральний стан).

Окремою важливою вимогою є відстеження динаміки настроїв у часі. Система має аналізувати дату або час публікації кожного відгуку та на основі цього будувати візуальні залежності, які показують, як змінювалося співвідношення різних емоцій протягом певного періоду.

Нарешті, система повинна забезпечувати наочне подання результатів у вигляді графіків (стовпчикових діаграм розподілу емоцій, лінійних графіків зміни настроїв у часі) та підсумкових таблиць із статистикою.

1.3.2 Нефункціональні вимоги

Окрім функціональних можливостей, система має відповідати низці нефункціональних вимог, які забезпечують її практичну цінність та зручність використання.

Насамперед система повинна забезпечувати швидкодію - обробка навіть великих масивів відгуків (десятки або сотні тисяч записів) має виконуватися за прийнятний час, щоб користувач не чекав годинами. Цього можна досягти завдяки оптимізованим алгоритмам векторизації та класифікації.

Важливою є також масштабованість - система повинна залишатися працездатною при постійному зростанні обсягів даних. Нові відгуки не повинні вимагати повного перенавчання моделей з нуля.

Зручність у використанні означає, що кінцевий користувач має легко завантажувати файли, запускати аналіз та розуміти результати завдяки інтуїтивно зрозумілому веб-інтерфейсу та чітким візуалізаціям.

Висновки до розділу 1

У першому розділі було проведено аналіз предметної галузі автоматизованого аналізу текстових відгуків користувачів. Розглянуто особливості текстових даних, зокрема їх неструктурованість, суб’єктивність, великі обсяги та лінгвістичну різноманітність, які створюють значні виклики для ручної обробки та обґрунтовують необхідність застосування методів обробки природної мови.

Визначено основні підходи до виявлення емоційних станів - логістичну регресію, випадковий ліс та нейронні мережі, а також розглянуто етапи попередньої обробки текстів (токенізацію, лематизацію, видалення стоп-слів, векторизацію). Сформульовано функціональні вимоги до системи (завантаження даних, обробка, класифікація емоцій, візуалізація динаміки настроїв) та нефункціональні (швидкодія, масштабованість, зручність, переносимість).

Таким чином, виконання сформульованих вимог дозволить створити ефективну, надійну та зручну інтелектуальну систему аналізу відгуків, здатну автоматично виявляти емоційне забарвлення текстів та відстежувати зміни настроїв користувачів у часі.

2 ЗАСОБИ РОЗРОБКИ СИСТЕМИ

2.1 Середовище розробки та використані інструменти

2.1.1 Мова програмування Python та бібліотеки

Для реалізації інтелектуальної системи аналізу емоційних станів та динаміки настроїв було обрано мову програмування Python [1]. Цей вибір зумовлений низкою факторів: Python має простий і зрозумілий синтаксис, велику спільноту розробників, а головне - надзвичайно широкий екосистему бібліотек для обробки даних, машинного навчання та візуалізації. Завдяки цьому можна зосередитися на логіці роботи системи, не витрачаючи час на реалізацію базових алгоритмів з нуля.

У процесі розробки використовується кілька ключових бібліотек. Для попередньої обробки текстових даних застосовується бібліотека NLTK (Natural Language Toolkit [12]), яка надає зручні інструменти для токенізації, видалення стоп-слів, лематизації та інших типових операцій з текстом. За потреби також може використовуватися бібліотека spaCy, яка пропонує більш швидкі та ефективні моделі для роботи з великими обсягами даних.

Для реалізації методів машинного навчання – зокрема логістичної регресії, випадкового лісу та нейронних мереж - задіяно бібліотеку scikit-learn. Вона містить готові класи для векторизації текстів (наприклад, TfidfVectorizer), класифікації, а також інструменти для оцінки якості моделей. Для нейронних мереж додатково можна використати бібліотеки TensorFlow або Keras, які дозволяють будувати складні архітектури для глибокого аналізу текстів.

Робота з табличними даними, зокрема завантаження відгуків із CSV-файлів та збереження результатів класифікації, здійснюється за допомогою бібліотеки pandas. Вона забезпечує зручні структури даних (DataFrame) та широкий набір функцій для маніпуляції таблицями.

Нарешті, для візуалізації результатів - побудови графіків розподілу емоцій та лінійних діаграм динаміки настроїв у часі - використовуються бібліотеки matplotlib

та seaborn. Вони дозволяють створювати якісні, інформативні та естетично привабливі графічні зображення, які роблять результати роботи системи зрозумілими для кінцевого користувача.

2.1.2 Характеристика середовища розробки

Оскільки мова програмування Python не прив’язана до жодного конкретного середовища, вибір IDE (Integrated Development Environment) залежить переважно від зручностей, які воно надає розробнику. У даній роботі використано PyCharm Professional [8] - інструмент, що добре зарекомендував себе у великих проектах із багатьма файлами та зовнішніми бібліотеками.

Основною причиною вибору стала глибока інтеграція з бібліотеками наукового стеку (pandas, numpy, matplotlib, scikit-learn). PyCharm дозволяє виконувати код у інтерактивному режимі через Python Console і візуалізувати графіки безпосередньо в інтерфейсі, що зручно під час налагодження модулів аналізу даних. Також важливим фактором є вбудований інструмент для роботи з Jupyter Notebooks, який дозволяє поєднувати код, текст, графіки та проміжні результати в одному файлі - це використовувалося на етапі експериментів із різними моделями класифікації емоцій.

Для контролю версій використовувався Git, інтегрований у PyCharm без додаткового налаштування, що спростило синхронізацію коду між робочим комп’ютером та хмарним репозиторієм (GitHub). Крім того, PyCharm автоматично перевіряє код на відповідність стандартам PEP 8, що позитивно впливає на його читабельність та підтримуваність у майбутньому.

2.2 Архітектура програмного застосунку

Архітектура програмного застосунку є одним із ключових аспектів розробки, оскільки вона визначає, як саме будуть взаємодіяти між собою окремі частини системи, наскільки зручно буде її модифікувати або розширювати в майбутньому, а також як ефективно вона виконуватиме поставлені завдання. У даній

кваліфікаційній роботі за основу взято модульний підхід, за якого кожна функціональна частина програми реалізується як відносно незалежний компонент. Це спрощує розробку, тестування, налагодження та подальшу підтримку системи.

Загалом архітектура розроблюваної системи складається з чотирьох основних модулів, які послідовно обробляють дані від моменту завантаження до отримання кінцевих результатів.

Першим і одним із найважливіших компонентів є модуль обробки тексту. Його головне завдання - підготувати «сирі» текстові відгуки до подальшого аналізу. У реальних умовах відгуки користувачів часто містять зайві символи, помилки, слова, написані великими літерами, а також різні граматичні форми одного й того самого слова. Модуль обробки тексту виконує очищення даних: видаляє знаки пунктуації, зайві пробіли, HTML-теги (якщо вони є), а також може перетворювати всі літери на нижній регістр для уніфікації. Далі відбувається токенізація - розбиття тексту на окремі слова (токени), після чого виконується видалення стоп-слів, тобто загальновживаних слів, які не несуть смислового навантаження (наприклад, «і», «але», «або», «це»). Завершальним етапом є лематизація - приведення кожного слова до його початкової словникової форми. Усі ці операції реалізуються за допомогою бібліотек NLTK та, за потреби, spaCy.

Після того як текст очищено й нормалізовано, в справу вступає модуль векторизації та класифікації. Оскільки алгоритми машинного навчання не вміють працювати безпосередньо з текстом, необхідно перетворити текстові дані на числові вектори. Для цього використовується метод TF-IDF (Term Frequency - Inverse Document Frequency), який оцінює важливість кожного слова в межах конкретного відгуку та в усьому корпусі текстів. Отримані вектори подаються на вхід моделям класифікації - логістичній регресії, випадковому лісу або нейронній мережі. Навчені моделі визначають емоційний стан кожного відгуку (радість, злість, розчарування, подив або нейтральність). Крім того, цей модуль відповідає за аналіз динаміки настроїв у часі: для цього кожен відгук має часову мітку, і після

класифікації система обчислює, як змінювалося співвідношення різних емоцій протягом певного періоду.

Третім компонентом є модуль візуалізації. Його завдання - зробити результати роботи системи наочними та зрозумілими для кінцевого користувача. На основі отриманих даних цей модуль будує стовпчикові діаграми розподілу емоцій (скільки відгуків належать до кожної емоційної категорії), а також лінійні графіки динаміки настроїв у часі, які показують, як змінювався емоційний фон аудиторії від тижня до тижня або від місяця до місяця. Для реалізації візуалізації використовуються бібліотеки matplotlib та seaborn, які дозволяють створювати якісні графічні зображення з можливістю їх збереження у файли.

Останнім компонентом є інтерфейс користувача. На відміну від багатьох навчальних проектів, де інтерфейс часто реалізують за допомогою Tkinter, у даній роботі обрано веб-орієнтований підхід із використанням фреймворку Flask. Це дозволяє створити сучасний, інтуїтивно зрозумілий інтерфейс, доступний через браузер. Користувач отримує просту веб-сторінку, на якій може завантажити файл із відгуками у форматі CSV, запустити процес аналізу, переглянути отримані графіки та таблиці, а також завантажити звіт. Flask забезпечує зв’язок між веб-сторінкою та всіма описаними вище модулями, викликаючи необхідні функції обробки, класифікації та візуалізації.

Таким чином, модульна архітектура з чітким розподілом обов’язків між компонентами забезпечує гнучкість, зручність тестування та можливість модифікації окремих частин системи без впливу на інші. Це особливо важливо для науково-дослідної роботи, де в процесі розробки можуть змінюватися підходи до обробки даних або моделі класифікації.

2.3 Інструменти для обробки тексту та кластеризації

Якісна попередня обробка текстових даних є критично важливим етапом у розробці будь-якої системи аналізу природної мови. Від того, наскільки правильно й повно буде очищено та нормалізовано вхідні тексти, безпосередньо залежить

точність подальшої класифікації емоційних станів. Саме тому в даній роботі приділено значну увагу етапу попередньої обробки, який включає кілька послідовних кроків.

Насамперед виконується очищення тексту. Цей процес передбачає видалення з відгуків усього, що не є значущим для аналізу: розділових знаків, цифр, спеціальних символів (на кшталт @, #, $), HTML-тегів, а також зайвих пробілів. Часто на цьому етапі також приводять усі літери до нижнього регістру, щоб слова на початку речення та в середині не сприймалися як різні токени. У результаті очищення залишається лише текстовий масив, який містить виключно слова, що несуть смислове та емоційне навантаження.

Наступним кроком є токенізація - розбиття очищеного тексту на окремі одиниці, які називаються токенами. У більшості випадків токенами виступають окремі слова, хоча іноді можуть розглядатися й цілі словосполучення. Токенізація дозволяє перейти від суцільного рядка тексту до списку окремих елементів, з якими потім буде зручно працювати.

Після токенізації виконується видалення стоп-слів. Стоп-словами називають загальновживані слова, які не несуть значущої інформації для аналізу емоцій, наприклад сполучники, прийменники, займенники та допоміжні дієслова. Видалення цих слів дозволяє зменшити розмірність векторів і підвищити точність класифікації.

Завершальним етапом попередньої обробки є лематизація - приведення кожного слова до його початкової словникової форми. Наприклад, слова «лікував», «лікувала», «лікувало» будуть приведені до форми «лікувати». Це дозволяє уникнути дублювання слів через граматичні варіації та зменшити розмірність простору ознак. Для реалізації токенізації, видалення стоп-слів та лематизації в роботі використовується бібліотека NLTK, яка надає готові інструменти для всіх цих операцій.

Після того як тексти очищено, токенізовано та лематизовано, необхідно перетворити їх на числові вектори, адже алгоритми машинного навчання не можуть

працювати безпосередньо з текстовими рядками. Для цього застосовується метод TF-IDF (Term Frequency - Inverse Document Frequency). Він оцінює важливість кожного слова в межах конкретного відгуку з урахуванням того, наскільки рідкісним або поширеним є це слово у всій вибірці. Слова, які часто зустрічаються в одному відгуку, але рідко - в інших, отримують високу вагу, що дозволяє моделі краще розрізняти різні емоційні категорії. Реалізація TF-IDV у роботі здійснюється за допомогою бібліотеки scikit-learn.

Нарешті, отримані числові вектори подаються на вхід моделям класифікації - логістичній регресії, випадковому лісу або нейронній мережі. Всі ці моделі також реалізовані з використанням scikit-learn, а для нейронних мереж додатково застосовується бібліотека TensorFlow [7]. Навчені моделі здатні визначати емоційне забарвлення кожного нового відгуку, що надходить до системи.

2.4 Інтерфейс користувача системи

Інтерфейс користувача реалізовано у вигляді веб-застосунку з використанням фреймворку Flask. Такий підхід дозволяє користувачеві працювати з системою через будь-який браузер без встановлення додаткового програмного забезпечення.

Головна сторінка містить форму для завантаження файлу з відгуками у форматі CSV та кнопку запуску аналізу. Після завантаження даних система виконує попередню обробку текстів, класифікацію емоцій (радість, злість, розчарування, подив, нейтральний стан) та аналіз динаміки настроїв у часі.

Результати роботи відображаються у вигляді підсумкової статистики, стовпчикової діаграми розподілу емоцій та лінійного графіка зміни настроїв у часі. Користувач також може переглянути детальну таблицю з емоційною розміткою кожного відгуку та зберегти отримані графіки.

У разі помилок система виводить зрозумілі повідомлення з поясненням проблеми, що робить роботу з інтерфейсом комфортною для користувача будь-якого рівня підготовки.

Висновки до розділу 2

У другому розділі розглянуто інструментарій, необхідний для реалізації інтелектуальної системи аналізу емоційних станів та динаміки настроїв користувачів. Основним середовищем розробки обрано PyCharm, мовою реалізації – Python завдяки його багатій екосистемі бібліотек для обробки природної мови та машинного навчання.

Визначено ключові бібліотеки: NLTK для токенізації, лематизації та видалення стоп-слів; scikit-learn для векторизації (TF-IDF) та класифікації (логістична регресія, випадковий ліс); matplotlib та seaborn для візуалізації розподілу емоцій та динаміки настроїв; Flask для створення веб-інтерфейсу.

Запропоновано модульну архітектуру системи, яка включає модуль обробки тексту, модуль векторизації та класифікації, модуль візуалізації та веб-інтерфейс. Така структура забезпечує гнучкість, зручність тестування та можливість подальшого розширення. Розроблений інтерфейс користувача дозволяє завантажувати дані у форматі CSV, запускати аналіз та отримувати наочні результати у вигляді графіків і таблиць.

Таким чином, обраний стек технологій та архітектурне рішення повністю відповідають поставленим завданням і створюють основу для успішної програмної реалізації системи.

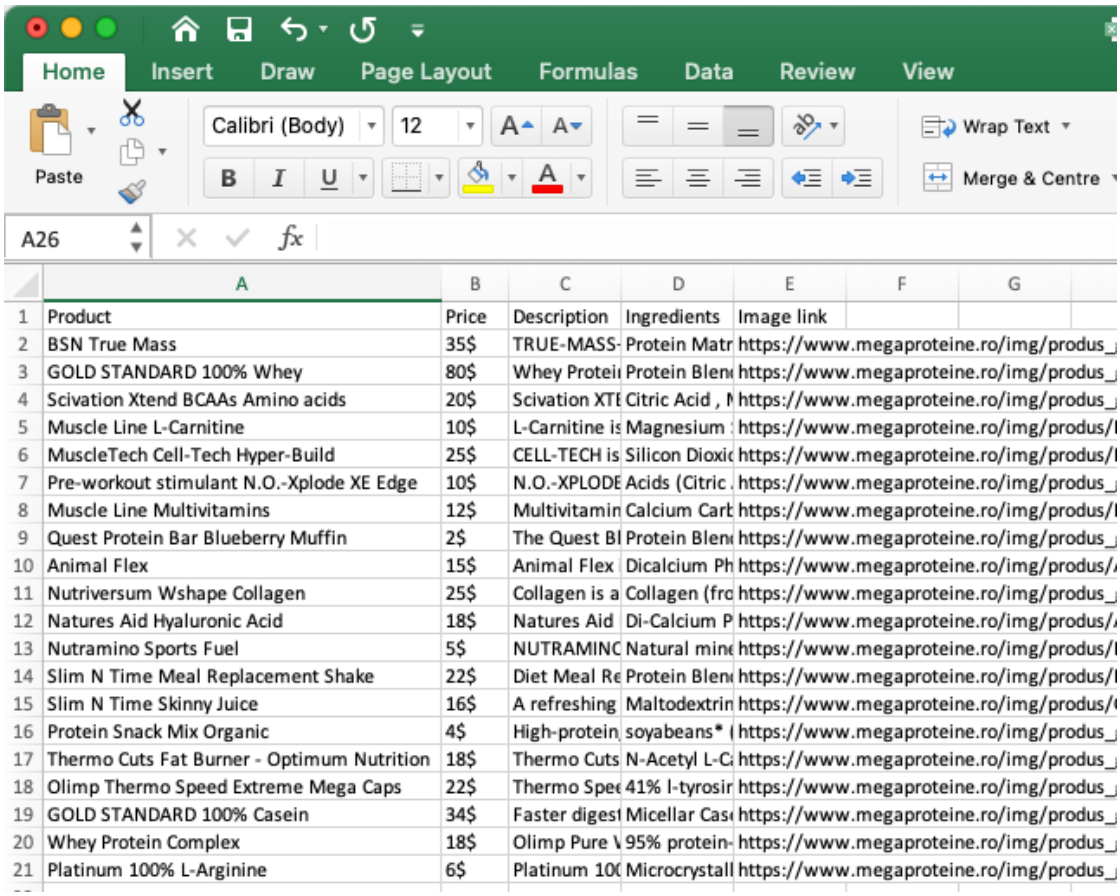
3 ДОСЛІДЖЕННЯ МЕТОДІВ ТА МОДЕЛЕЙ КЛАСИФІКАЦІЇ ЕМОЦІЙ У ТЕКСТОВИХ ВІДГУКАХ

3.1 Характеристика вхідних даних та їх попередня обробка

3.1.1 Опис та структура вхідних даних

Для розробки інтелектуальної системи аналізу емоційних станів та динаміки настроїв важливу роль відіграють вхідні дані, адже саме від їхньої якості, структури та повноти безпосередньо залежить ефективність роботи подальших алгоритмів машинного навчання. У даній роботі вхідними даними виступають текстові відгуки користувачів, які представлені у вигляді файлів формату CSV. Цей формат було обрано не випадково - він є одним із найпоширеніших для зберігання табличних даних, підтримується більшістю аналітичних інструментів та бібліотек, зокрема pandas, що використовується в цій роботі. CSV-файл являє собою звичайний текстовий файл, де кожен рядок відповідає одному запису (відгуку), а стовпці розділені комами або іншими роздільниками. Така структура дозволяє легко завантажувати, обробляти та зберігати результати аналізу без додаткових перетворень.

На рисунку 3.1 зображено приклад CSV-файлу з існуючими даними(рис. 3.1).



	A	B	C	D	E
1	Product	Price	Description	Ingredients	Image link
2	BSN True Mass	35$	TRUE-MASS-	Protein Matr	https://www.megaproteine.ro/img/produs_
3	GOLD STANDARD 100% Whey	80$	Whey Protei	Protein Blen	https://www.megaproteine.ro/img/produs_
4	Scivation Xtend BCAAs Amino acids	20$	Scivation XTE	Citric Acid , M	https://www.megaproteine.ro/img/produs_
5	Muscle Line L-Carnitine	10$	L-Carnitine is	Magnesium	https://www.megaproteine.ro/img/produs/
6	MuscleTech Cell-Tech Hyper-Build	25$	CELL-TECH is	Silicon Dioxi	https://www.megaproteine.ro/img/produs/
7	Pre-workout stimulant N.O.-Xplode XE Edge	10$	N.O.-XPLODE	Acids (Citric	https://www.megaproteine.ro/img/produs_
8	Muscle Line Multivitamins	12$	Multivitamin	Calcium Carb	https://www.megaproteine.ro/img/produs/
9	Quest Protein Bar Blueberry Muffin	2$	The Quest Bl	Protein Blen	https://www.megaproteine.ro/img/produs_
10	Animal Flex	15$	Animal Flex	Dicalcium Ph	https://www.megaproteine.ro/img/produs/
11	Nutriversum Wshape Collagen	25$	Collagen is a	Collagen (fro	https://www.megaproteine.ro/img/produs_
12	Natures Aid Hyaluronic Acid	18$	Natures Aid	Di-Calcium P	https://www.megaproteine.ro/img/produs/
13	Nutramino Sports Fuel	5$	NUTRAMINC	Natural min	https://www.megaproteine.ro/img/produs/
14	Slim N Time Meal Replacement Shake	22$	Diet Meal Re	Protein Blen	https://www.megaproteine.ro/img/produs/
15	Slim N Time Skinny Juice	16$	A refreshing	Maltodextrin	https://www.megaproteine.ro/img/produs/
16	Protein Snack Mix Organic	4$	High-protein	soyabeans* (	https://www.megaproteine.ro/img/produs_
17	Thermo Cuts Fat Burner - Optimum Nutrition	18$	Thermo Cuts	N-Acetyl L-C	https://www.megaproteine.ro/img/produs_
18	Olimp Thermo Speed Extreme Mega Caps	22$	Thermo Spe	41% l-tyrosir	https://www.megaproteine.ro/img/produs_
19	GOLD STANDARD 100% Casein	34$	Faster digest	Micellar Cas	https://www.megaproteine.ro/img/produs_
20	Whey Protein Complex	18$	Olimp Pure \	95% protein-	https://www.megaproteine.ro/img/produs_
21	Platinum 100% L-Arginine	6$	Platinum 10(	Microcrystall	https://www.megaproteine.ro/img/produs_

Рисунок 3.1 – Приклад структури CSV-файлу [20]

Типовий набір даних для аналізу відгуків містить кілька ключових стовпців. Насамперед це текст самого відгуку, який є основним об’єктом дослідження - саме в ньому закладено емоційне забарвлення, яке система має навчитися розпізнавати. Текст може бути різної довжини - від одного короткого речення до кількох абзаців розлогих міркувань. Крім тексту, у наборі даних часто присутня часова мітка - дата або точний час публікації відгуку. Цей стовпець є надзвичайно важливим для реалізації однієї з ключових функцій системи - відстеження динаміки настроїв у часі. Без часової прив’язки система змогла б лише констатувати поточний розподіл емоцій, але не показувати, як цей розподіл змінювався день за днем, тиждень за тижнем або місяць за місяцем. У деяких наборах даних також може бути стовпець із вже наявною оцінкою (наприклад, кількість зірок від 1 до 5), яка може використовуватися як допоміжна інформація або для порівняння з результатами роботи системи, хоча основна увага в цій роботі приділяється саме аналізу тексту, а не числових оцінок.

3.1.2 Очищення тексту, нормалізація та токенізація

Текстові відгуки в тому вигляді, в якому вони надходять від користувачів, є практично непридатними для безпосередньої обробки алгоритмами машинного навчання. Причина полягає в тому, що ці алгоритми працюють виключно з числами, а не з рядками тексту. Крім того, сирі тексти мають низку особливостей, які суттєво ускладнюють аналіз або навіть спотворюють його результати. Тому перед тим, як подавати дані на вхід моделям класифікації, необхідно виконати цілу низку процедур, які в сукупності називаються попередньою обробкою текстів. Цей етап є критично важливим, оскільки від якості його виконання залежить точність усіх подальших передбачень системи.

На рисунку 3.2 показані загальні методи очищення текстових даних (рис. 3.2).

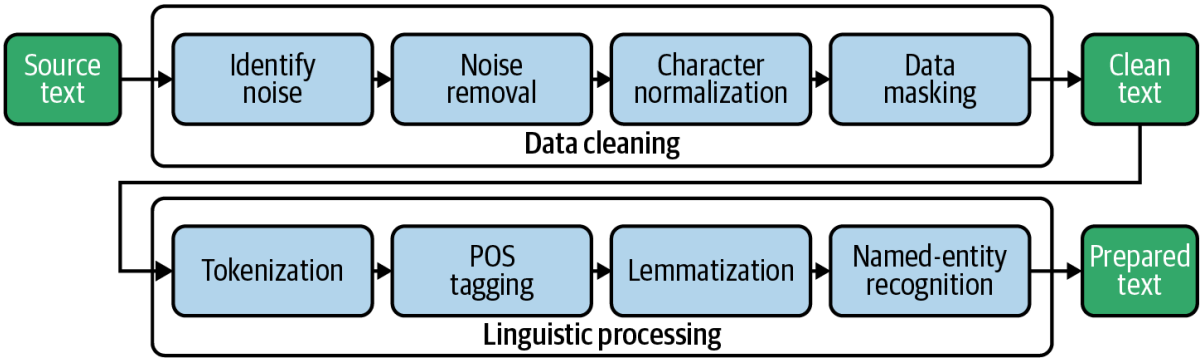


Рисунок 3.2 – Процедури очищення тексту [19]

Процес попередньої обробки починається з очищення тексту. На цьому етапі з відгуків видаляються всі символи, які не несуть смислового або емоційного навантаження. До таких символів належать розділові знаки (крапки, коми, знаки оклику та питання, хоча останні іноді зберігають як маркери емоційності, але частіше видаляються), цифри, спеціальні символи на кшталт @, #, $, %, а також зайві пробіли, символи нового рядка та табуляції. Часто на цьому етапі також виконується приведення всіх літер до нижнього регістру, що дозволяє уникнути ситуації, коли слова «Чудово» на початку речення та «чудово» в середині сприймаються алгоритмом як різні токени.

Після очищення тексту настає черга токенізації. Цей процес полягає в розбитті суцільного текстового рядка на окремі смислові одиниці - токени. У найпростішому випадку токенами виступають окремі слова, розділені пробілами. Однак на практиці токенізація має враховувати такі нюанси, як апострофи у словах («не хочу», «комп'ютер»), дефіси («швидко-швидко»), а також скорочення. На рисунок 3.3 зображено токенізацію тексту поетапно (рис. 3.3).

Рисунок 3.3 – Токенізація тексту поетапно

Якісна токенізація є основою для всіх подальших етапів обробки, адже якщо система неправильно розіб'є текст на слова, то помилки будуть накопичуватися далі. У даній роботі для токенізації використовується бібліотека NLTK, яка надає кілька готових токенізаторів.

3.1.3 Видалення стоп-слів та лематизація

Після того як текст розбито на окремі токени, виконується видалення стоп-слів. Стоп-словами називають загальновживані слова, які зустрічаються практично в будь-якому тексті, але не несуть специфічної інформації для вирішення конкретної задачі. У задачах аналізу емоцій такими словами є, наприклад, сполучники («і», «або», «але», «що»), прийменники («в», «на», «під», «над»), займенники («я», «ти», «він», «вона», «воно», «ми», «ви», «вони»), а також допоміжні дієслова («бути», «ставати» тощо). Видалення стоп-слів дозволяє суттєво зменшити розмірність простору ознак, адже замість десятків тисяч унікальних слів у словнику залишаються лише ті, які дійсно мають значення для розрізнення емоційних категорій. Крім того, цей крок допомагає позбутися шуму та зосередитися на ключових словах, які найчастіше асоціюються з тією чи іншою емоцією. Наприклад, слова «жахливо», «розчарований» з великою ймовірністю вказуватимуть на негатив, а «чудово», «задоволений» – на позитив.

Останнім етапом іде лематизація. Це процес, коли кожне слово приводиться до його початкового стану. Наприклад, слова «лікував», «лікувала», «лікувало» будуть приведені до леми «лікувати». Лематизація дозволяє об’єднати усі форми слова до одного стану, в одну ознаку, що дозволяє моделі краще узагальнювати знання. Адже для визначення емоції неважливо, чи написав користувач «розчарований», «розчарована» чи «розчаровані» - емоційний стан від цього не змінюється, тому всі ці форми мають сприйматися моделлю як одна й та сама ознака. У цій роботі для лематизації використовується бібліотека NLTK, яка для англійської мови має вбудований лематизатор WordNet.

3.1.4 Перетворення тексту на числові вектори

Після того як тексти очищено, токенізовано та лематизовано, вони все ще залишаються послідовностями слів, а не числами. Алгоритми машинного навчання, які будуть використовуватися для класифікації емоцій, не можуть працювати

безпосередньо з текстовими рядками або списками токенів - їм потрібні числові вектори фіксованої довжини. Саме тому наступним кроком є векторизація - перетворення текстів на числові вектори.

У даній роботі для векторизації використовується метод TF-IDF, що розшифровується як Term Frequency - Inverse Document Frequency, тобто частота терміна – обернена частота документа. Цей метод оцінює важливість кожного слова в межах конкретного відгуку з урахуванням того, наскільки рідкісним або поширеним є це слово у всій вибірці. Формула TF-IDF складається з двох частин. Перша частина - TF (частота терміна) - показує, як часто слово зустрічається в конкретному документі (відгуку). Друга частина - IDF (обернена частота документа) - показує, наскільки рідко слово зустрічається в усіх документах корпусу. Логарифм від відношення загальної кількості документів до кількості документів, що містять це слово, дає високе значення для слів, які є рідкісними, але характерними для певних документів.

Перевага TF-IDF над простішими методами, такими як «мішок слів» (Bag-of-Words), полягає в тому, що він автоматично зменшує вагу загальновживаних слів, які зустрічаються майже в кожному відгуку (наприклад, «препарат», «ліки», «день»), та підвищує вагу слів, які є більш характерними для конкретних емоцій (наприклад, «жах», «захват», «розчарований»). У результаті кожен текстовий відгук перетворюється на числовий вектор, де кожній координаті відповідає певне слово зі словника, а значення координати - це вага TF-IDF цього слова в даному відгуку. Словник формується на основі всіх унікальних слів, що залишилися після попередньої обробки, і може містити від кількох тисяч до десятків тисяч слів залежно від обсягу вхідних даних. Отримані вектори стають вхідними даними для моделей класифікації - логістичної регресії, випадкового лісу та нейронних мереж, які й визначатимуть емоційне забарвлення кожного відгуку.

3.2 Порівняльна характеристика методів класифікації текстових даних

3.2.1 Логістична регресія як метод класифікації емоцій

Логістична регресія є одним із найпростіших та найпоширеніших методів класифікації в задачах машинного навчання. У контексті аналізу текстових відгуків логістична регресія дозволяє визначити, до якої емоційної категорії - радість, злість, розчарування, подив або нейтральний стан - належить той чи інший текст. Основна ідея методу полягає в тому, що він обчислює ймовірність належності об’єкта до кожного з класів, використовуючи так звану логістичну функцію, яка перетворює будь-яке дійсне число на значення в інтервалі від 0 до 1. Це можна зробити за допомогою регресивного рівняння:

$$\mathrm{P} = \frac{1}{1+e^{-y}}, \tag{3.1}$$

Де:

P - імовірність того, що відбудеться подія;

e - основа натуральних логарифмів 2,17….;

y - стандартне рівняння регресії.

Це значення інтерпретується як ймовірність того, що об’єкт належить до певного класу.

Регресійні моделі можна записувати у такій формулі:

$$\mathrm{y} = \mathrm{F}(\mathrm{x1}, \mathrm{x2}, \ldots\ \mathrm{xn}), \tag{3.2}$$

Лінійна функція незалежних змінних у множинній лінійній регресії:

$$\mathrm{y} = \mathrm{a} + \mathrm{b1x1} + \mathrm{b222} + \cdots + \mathrm{bnxn}, \tag{3.3}$$

Логістична регресія має низку переваг, які роблять її привабливим вибором для задач класифікації текстів. По-перше, вона є швидкою як у навчанні, так і в передбаченні, що особливо важливо при роботі з великими масивами даних. По-друге, вона добре працює з високорозмірними розрідженими даними, якими є вектори TF-IDF, адже текстові дані після векторизації можуть містити десятки тисяч ознак, але більшість цих ознак у конкретному документі дорівнюють нулю. По-третє, логістична регресія дає зрозумілу інтерпретацію - ваги, які модель присвоює кожному слову, показують, наскільки сильно це слово впливає на передбачення тієї чи іншої емоції.

3.2.2 Випадковий ліс (Random Forest) та його особливості

Випадковий ліс належить до ансамблевих методів машинного навчання, тобто він поєднує в собі багато більш простих моделей - дерев рішень. Кожне дерево рішень будується на основі випадкової підмножини даних та випадкової підмножини ознак, а потім усі дерева «голосують» за остаточний результат. Такий підхід дозволяє значно підвищити точність класифікації та зменшити ризик перенавчання (overfitting), коли модель запам’ятовує навчальну вибірку, але не може узагальнювати знання на нові дані.

Під час побудови кожного дерева для розбиття вузлів використовується критерій Джині, який вимірює «чистоту» вузла. Цей критерій обчислюється за формулою:

$$G(S) = 1 - \sum_{i=1}^{c} p_i^2, \quad (3.4)$$

Де:

p - частка зразків класу i у вузлі S;

c - кількість класів.

Чим менше значення G(S), тим кращим є розбиття, оскільки вузол стає більш «чистим».

Після побудови T дерев остаточне передбачення приймається шляхом голосування:

$$\hat{y} = \text{a}gr\ max_y \sum_{t=1}^{T} p_{t=1}^{2} \| (\hat{y}_t = y), \quad (3.5)$$

Де:

T - кількість дерев у лісі;

y^t - передбачення t-го дерева;

II - індикаторна функція.

Це означає, що модель обирає той клас емоції, який отримав найбільше голосів від окремих дерев.

У контексті аналізу емоцій у текстових оглядах випадковий ліс має кілька важливих переваг над логістичною регресією. Перш за все, він здатний враховувати нелінійні залежності між ознаками. Наприклад, він може «зрозуміти», що слово «не» змінює значення на протилежне лише тоді, коли стоїть перед певними прикметниками («не погано», «не добре»), і саме по собі має мало значення. Крім того, випадковий ліс добре працює з даними, де кількість ознак значно перевищує кількість об’єктів - це типова ситуація для текстових даних, де словник може містити десятки тисяч слів, а документи - лише кілька тисяч.

Ще однією важливою особливістю випадкового лісу є можливість оцінки важливості кожної ознаки. Після навчання моделі можна побачити, які слова мали найбільший вплив на прийняття рішень. Це дозволяє досліднику зрозуміти, які лексичні маркери є найбільш інформативними для кожної емоційної категорії.

Однак, випадковий ліс не позбавлений своїх недоліків. Він вимагає значно більше обчислювальних ресурсів, ніж логістична регресія, особливо під час навчання. Крім того, він може бути схильний до перенавчання, якщо дерева рішень

у лісі занадто глибокі. Ця проблема вирішується правильним встановленням таких параметрів, як максимальна глибина дерева або мінімальна кількість вибірок на листок. Випадковий ліс також менш інтерпретується, ніж логістична регресія, хоча оцінка важливості ознак частково вирішує цю проблему.

3.2.3 Нейронні мережі для аналізу емоційного забарвлення текстів

Нейронні мережі є найпотужнішим, але й найскладнішим інструментом серед усіх методів класифікації, розглянутих у цій роботі. На відміну від логістичної регресії та випадкового лісу, які працюють з векторними представленнями текстів, нейронні мережі можуть зупиняти обробку фази слів, враховуючи їхній порядок та контекст. Саме ця здатність робить їх особливо ефективними для аналізу емоцій, оскільки в мові порядок слів має критичне значення.

Базовим елементом нейронної мережі є штучний нейрон, який отримує на вхід набір значень, множить їх на ваги, додає зміщення та пропускає через функцію активації:

$$y = f(\sum_{i=1}^{n} w_i x_i + b), \quad (3.6)$$

Де:

x - вхідні значення;

w - ваги;

b - зміщення;

f - функція активації.

У прихованих шарах розробленої нейронної мережі використовується функція активації ReLU, яка працює за простим правилом: якщо вхідне значення додатне, воно пропускається без змін; якщо від’ємне – замінюється на нуль:

$$f(x) = \max(0, x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}, \quad (3.7)$$

У вихідному шарі використовується функція softmax, яка перетворює вихідні значення нейронів на ймовірності для кожного з п’яти класів емоцій:

$$softmax(z_j) = \frac{e^{x}j}{\sum_{k=1}^{5} e^{z}k}, \quad (3.8)$$

Сума всіх ймовірностей дорівнює 1, тому результат можна інтерпретувати як «ймовірність того, що відгук виражає певну емоцію».

У завданнях обробки природної мови для аналізу тексту найчастіше використовуються два типи нейронних мереж: згорткові та рекурентні. Згорткові мережі спочатку були розроблені для обробки зображень, але також довели свою ефективність для тексту. Вони використовують спеціальні фільтри, які ковзають по послідовності слів і виявляють локальні закономірності - наприклад, певні словосполучення, які часто зустрічаються разом. Рекурентні мережі, особливо їх варіант LSTM [21], здатні запам’ятовувати інформацію про попередні слова та використовувати її для аналізу наступних. Це дозволяє їм враховувати довгострокові залежності в тексті, що важливо для розуміння складних конструкцій, наприклад, сарказму або іронії. На рисунку 3.4 зображено рекурентну нейронну мережу (рис. 3.4).

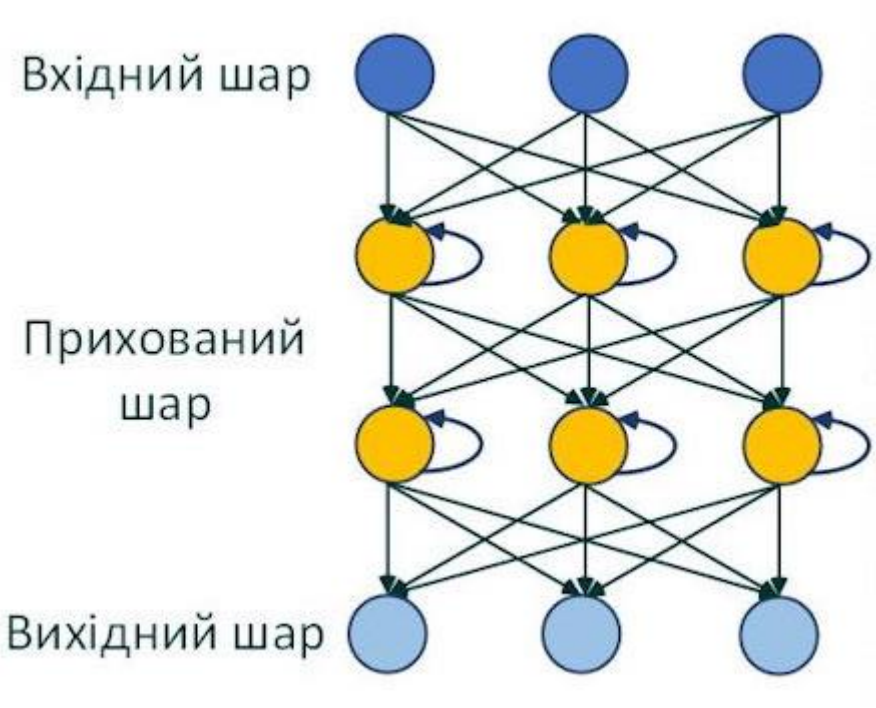


Рисунок 3.4 – Рекурентна нейронна мережа [18]

Основними перевагами нейронних мереж є їхня здатність до автоматичного виділення ознак, а також висока точність на складних даних. Вони здатні розпізнавати тонкі емоційні відтінки, які можуть бути непомітними для простіших моделей. Крім того, нейронні мережі добре справляються зі змішаними емоціями, коли відгук містить як позитивні, так і негативні висловлювання.

3.2.4 Порівняння моделей та критерії оцінки їх ефективності

Щоб визначити, яка з трьох описаних моделей - логістична регресія, випадковий ліс чи нейронна мережа - є найбільш придатною для вирішення конкретного завдання класифікації емоцій у текстових оглядах, необхідно порівняти їх за низкою критеріїв. До основних таких критеріїв належать точність класифікації, швидкість навчання та прогнозування, стійкість до перенавчання, інтерпретованість результатів, а також вимоги до обсягу даних та обчислювальних ресурсів.

Логістична регресія є найшвидшою та найпростішою у реалізації. Вона добре працює на невеликих наборах даних, видає чіткі результати та не вимагає потужного обладнання. Однак її точність зазвичай нижча, ніж у складніших моделей, особливо коли дані містять складні нелінійні зв'язки. Випадковий ліс займає проміжне положення: він точніший за логістичну регресію, але водночас значно швидший за нейронні мережі та менш вимогливий до обсягу даних. Він також дозволяє оцінити важливість ознак, що допомагає в інтерпретації результатів.

Нейронні мережі демонструють найвищу потенційну точність, особливо на великих наборах даних та за наявності складних лінгвістичних явищ (сарказм, іронія, змішані емоції). Однак вони вимагають значно більше часу для навчання, потужнішого обладнання та великих обсягів розмічених даних. Крім того, їхні результати важко інтерпретувати.

3.3 Аналіз отриманих результатів

3.3.1 Метрики оцінки якості класифікації

Для того щоб об'єктивно порівняти ефективність різних моделей класифікації емоцій - логістичної регресії, випадкового лісу та нейронної мережі - необхідно використовувати стандартизовані метрики, які дозволяють кількісно оцінити, наскільки добре кожна модель справляється зі своїм завданням. У задачах класифікації, особливо коли кількість об'єктів різних класів може бути нерівномірною, покладатися лише на одну метрику (наприклад, загальну точність) було б помилковим, оскільки це може приховувати проблеми з класифікацією окремих категорій.

Найпростішою та найінтуїтивнішою метрикою є точність (accuracy). Вона показує, яку частку від усіх проаналізованих відгуків модель класифікувала правильно. Наприклад, якщо система правильно визначила емоцію у 850 відгуках із 1000, точність становитиме 85%. Ця метрика є зручною для загальної оцінки, але

якщо один із класів (наприклад, «радість») зустрічається набагато частіше за інші, модель може просто завжди передбачати «радість» і отримувати високу точність, хоча насправді вона нічого не навчилася. Тому поряд із точністю використовуються й інші метрики.

До таких метрик належать точність (precision) та повнота (recall). Висока точність означає, що коли модель каже «цей відгук виражає злість», їй можна вірити. Висока повнота означає, що модель не пропускає відгуки з певною емоцією. Ідеальна модель має одночасно високу точність і високу повноту, але на практиці між ними часто доводиться шукати баланс: модель може бути дуже обережною (висока точність, але низька повнота) або, навпаки, намагатися знайти всі відгуки (висока повнота, але низька точність).

Для зручного поєднання точності та повноти використовується F1-міра - це гармонійне середнє між цими двома метриками. Вона набуває високих значень лише тоді, коли обидві метрики є високими. F1-міра є однією з найпопулярніших метрик у задачах класифікації, особливо коли класи не збалансовані. Для задачі з п’ятьма емоційними класами (радість, злість, розчарування, подив, нейтральний стан) F1-міра обчислюється окремо для кожного класу, а потім усереднюється - так отримують макроусереднену F1-міру, яка дає загальне уявлення про якість моделі.

3.3.2 Матриця помилок та її інтерпретація

Окрім чисельних метрик, важливим інструментом аналізу якості класифікації є матриця помилок (confusion matrix). Це таблиця, в якій рядки відповідають реальним емоційним міткам відгуків, а стовпці - тим міткам, які передбачила система. На перетині рядка та стовпця вказується кількість відгуків, які насправді належать до однієї емоції, але були класифіковані як інша.

Аналіз матриці помилок надзвичайно корисний для вдосконалення системи. Якщо дослідник бачить, що певні пари емоцій постійно плутаються, він може вжити заходів: збільшити кількість навчальних прикладів для цих категорій, додати додаткові функції (наприклад, розглянути емодзі або знаки оклику) або навіть

об’єднати дві схожі категорії в одну, якщо це виправдано з практичної точки зору. У контексті цієї роботи буде побудовано матрицю помилок для кожної з трьох моделей - логістичної регресії, випадкового лісу та нейронної мережі - що дозволить не лише порівняти їх загальну продуктивність, але й зрозуміти, які емоції кожна модель розпізнає краще, а які мають труднощі.

3.3.3 Інтерпретація результатів та порівняння моделей

Після того, як усі три моделі будуть навчені та протестовані на одному наборі даних, необхідно буде порівняти результати та зробити висновок, яка модель найбільше підходить для впровадження в кінцевій інтелектуальній системі. При цьому слід враховувати не лише числові значення метрик, але й практичні аспекти, такі як продуктивність, вимоги до ресурсів та інтерпретованість результатів.

Очікується, що нейронна мережа покаже найвищу точність та F1-міру, особливо якщо навчальні дані достатньо великі. Вона може фіксувати складні зв’язки між словами, враховувати порядок слів у реченні та, ймовірно, краще справлятиметься із сарказмом та змішаними емоціями, ніж інші моделі. Однак ця перевага може бути компенсована високими обчислювальними ресурсами та вимогами до часу навчання.

Логістична регресія, ймовірно, покаже найнижчу точність, але вона буде найшвидшою та найпростішою у реалізації. Вона може бути хорошим вибором, якщо швидкість є пріоритетом або якщо набір даних невеликий. Очікується, що випадковий ліс займе проміжне положення: він буде точнішим, ніж логістична регресія, але поступатиметься нейронним мережам, водночас вимагаючи менше ресурсів, ніж нейронні мережі.

Окрім порівняння трьох моделей одна з одною, важливо також проаналізувати, наскільки добре кожна з них справляється з окремими емоційними категоріями. Такий аналіз дозволить не просто вибрати найкращу модель, але й зрозуміти, які аспекти задачі є найскладнішими та на чому слід зосередитися в майбутніх дослідженнях.

3.3.4 Аналіз динаміки настроїв у часі

Окремим важливим аспектом роботи є аналіз динаміки настроїв з плином часу. На відміну від простої класифікації кожного відгуку окремо, аналіз динаміки дозволяє побачити загальну картину зміни емоційного фону аудиторії за певний період. Для цього кожен відгук необхідно прив’язати до позначки часу - дати публікації. Після того, як система класифікувала емоції всіх відгуків, дані агрегуються за певні проміжки часу: дні, тижні або місяці, залежно від доступної кількості даних та необхідного рівня деталізації.

Результатом такої агрегації є набір часових рядів - окремий ряд для кожної емоційної категорії. Наприклад, можна побудувати графік, який показує, як змінилася частка негативних відгуків (гнів + розчарування) за останні три місяці. Різке збільшення негативних відгуків може свідчити про проблеми з продуктом або послугою, що з’явилися в певний момент, тоді як поступове збільшення позитивних відгуків може свідчити про успіх маркетингових кампаній або вдосконалення продукту.

У рамках даної роботи результатом аналізу динаміки настроїв будуть лінійні графіки або стовпчикові діаграми, на яких відображено зміну кількості або частки кожної емоції в часі. Для більшої наочності може бути застосовано згладжування, щоб прибрати випадкові короткострокові коливання та виділити довгострокові тренди. Ці візуалізації стануть важливою частиною підсумкового звіту системи та дозволять користувачеві швидко оцінити загальну ситуацію та прийняти обґрунтовані рішення.

3.4 Аналіз існуючих аналогів та обґрунтування доцільності розробки

Для того щоб об’єктивно оцінити місце та значення розроблюваної інтелектуальної системи аналізу емоційних станів та динаміки настроїв, необхідно розглянути вже існуючі на ринку рішення, які виконують подібні або близькі завдання. Це дозволить не лише виявити сильні сторони наявних продуктів, але й

зрозуміти, які потреби залишаються незадоволеними, а отже - обґрунтувати актуальність та доцільність власної розробки.

Серед найбільш відомих комерційних та відкритих систем аналізу емоційної тональності текстів можна виділити кілька ключових. Першим із них є IBM Watson Natural Language Understanding - потужна хмарна платформа, яка надає можливість аналізувати тональність текстів, виявляти емоції та багато інших лінгвістичних характеристик. Ця система використовує глибокі нейронні мережі, навчені на величезних корпусах даних, і демонструє високу точність. Однак вона має суттєві недоліки: по-перше, це комерційний продукт із платним доступом, по-друге, вона працює виключно в хмарі, що вимагає постійного підключення до інтернету та передачі даних третій стороні, що не завжди є прийнятним з точки зору конфіденційності. Крім того, користувач не має можливості впливати на процес навчання моделей або адаптувати їх під специфіку своєї предметної галузі.

Іншим популярним рішенням є Google Cloud Natural Language API. Він також пропонує аналіз тональності та виявлення емоцій, підтримує багато мов і має зручний інтерфейс для розробників. Як і у випадку з IBM Watson, це хмарний сервіс із платною моделлю використання. Його точність є високою, але користувач не знає, за якими принципами модель приймає рішення, і не може її донавчати під свої специфічні задачі. Для дослідницьких або навчальних цілей це створює суттєві обмеження, оскільки унеможливлює глибоке розуміння процесів, що відбуваються всередині системи.

Серед відкритих бібліотек та інструментів варто згадати TextBlob та VADER (Valence Aware Dictionary and sEntiment Reasoner). Це легковагі бібліотеки для Python, які дозволяють швидко виконувати аналіз тональності текстів, особливо в соціальних мережах. VADER, наприклад, спеціально налаштований на роботу з короткими текстами, емодзі, сленгом та капслоком. Однак обидві ці бібліотеки базуються на словникових підходах та простих правилах, а не на машинному навчанні. Це означає, що вони не здатні адаптуватися до нових даних, не

враховують контекст у достатній мірі та показують низьку точність на складних або неоднозначних текстах.

У чому ж полягають переваги та особливості системи, що створюється в межах цієї кваліфікаційної роботи, порівняно з описаними аналогами? Насамперед, на відміну від комерційних хмарних сервісів, розроблювана система є повністю локальною та безкоштовною. Користувач не платить за кожен проаналізований відгук, не передає свої дані третім особам і не залежить від наявності інтернет-з'єднання.

По-друге, на відміну від простих словникових бібліотек (TextBlob, VADER), запропонована система використовує методи машинного навчання з учителем, що дозволяє їй адаптуватися до конкретних даних та досягати вищої точності на специфічних доменах. Якщо користувач має розмічену вибірку відгуків із певної сфери, він може навчити модель на цих даних і отримати результати, які будуть значно точнішими, ніж універсальні моделі.

По-третє, важливою відмінністю є можливість відстеження динаміки настроїв у часі. Багато існуючих систем аналізують відгуки як статичний набір даних, не враховуючи часової компоненти. Розроблювана система, натомість, спеціально передбачає аналіз змін емоційного фону день за днем, тиждень за тижнем або місяць за місяцем. Це дозволяє виявляти тренди, реакції на події та довгострокові зміни в сприйнятті продукту чи послуги.

Таким чином, аналіз аналогів показує, що розроблювана система не намагається конкурувати з потужними хмарними платформами за абсолютною точністю, але займає свою нішу - локального, безкоштовного, адаптованого до конкретних даних інструменту з акцентом на динаміку настроїв. Для навчальних, дослідницьких та малих комерційних проектів це робить її більш привабливою, ніж універсальні хмарні рішення або надто прості словникові бібліотеки.

Висновки до 3 розділу

У третьому розділі роботи було здійснено теоретичне дослідження, яке стало основою для подальшої розробки інтелектуальної системи аналізу емоційних станів та динаміки настроїв. Насамперед було детально розглянуто структуру вхідних даних та обґрунтовано необхідність їх попередньої обробки. Встановлено, що текстові відгуки користувачів, подані у форматі CSV, потребують очищення, токенізації, видалення стоп-слів, лематизації та векторизації за допомогою методу TF-IDF. Кожен із цих етапів відіграє важливу роль у підвищенні якості подальшої класифікації, оскільки дозволяє позбутися шуму та звести різні граматичні форми слів до єдиного представлення.

Далі було проведено порівняльний аналіз трьох методів класифікації - логістичної регресії, випадкового лісу та нейронних мереж. З'ясовано, що кожен із цих методів має свої сильні та слабкі сторони. Логістична регресія є найшвидшою та найпростішою, але обмежена лінійними залежностями. Випадковий ліс здатен враховувати нелінійні взаємодії між словами та оцінювати важливість ознак, однак потребує більше ресурсів. Нейронні мережі є найпотужнішими, особливо для виявлення складних лінгвістичних явищ, але їх ефективне застосування вимагає великих обсягів даних та значних обчислювальних потужностей.

Для об'єктивної оцінки якості цих моделей було визначено систему метрик, яка включає точність, прецизійність, повноту, F1-міру та матрицю помилок. Використання цих метрик дозволить не лише порівняти моделі між собою, але й виявити конкретні проблеми, такі як плутанина між окремими емоційними категоріями. Окрему увагу було приділено аналізу динаміки настроїв у часі, який є ключовою відмінністю розроблюваної системи від багатьох аналогів і дозволяє відстежувати зміни емоційного фону аудиторії протягом певного періоду.

На завершення було виконано аналіз існуючих аналогів - хмарних сервісів IBM Watson та Google Cloud, простих словникових бібліотек TextBlob і VADER, а також потужних моделей сімейства BERT. Цей аналіз показав, що комерційні

рішення є закритими та платними, прості бібліотеки мають низьку точність, а BERT потребує надто багато ресурсів. Розроблювана система займає оптимальну нішу: вона є локальною, безкоштовною, дозволяє використовувати різні моделі машинного навчання та має вбудований функціонал аналізу динаміки настроїв, що робить її привабливим інструментом для навчальних, дослідницьких та малих комерційних проектів.

Таким чином, третій розділ повністю виконав своє завдання - створено теоретичне підґрунтя, яке дозволяє перейти до практичної програмної реалізації інтелектуальної системи аналізу емоційних станів та динаміки настроїв у наступному розділі роботи.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

4.1 Створення та навчання моделей класифікації емоцій

Для навчання моделей класифікації емоцій було використано набір текстових відгуків користувачів, попередньо розмічених за п’ятьма емоційними категоріями: радість, злість, розчарування, подив та нейтральний стан. Дані зберігаються у файлі формату CSV, який містить дві колонки: text (текст відгуку) та emotion (числова мітка емоції від 0 до 4). Перед початком навчання всі тексти проходять етап попередньої обробки, який включає очищення від зайвих символів, токенізацію, видалення стоп-слів, лематизацію та векторизацію за допомогою методу TF-IDF. Після векторизації дані розділяються на тренувальний та тестовий набори у співвідношенні 80% до 20% відповідно.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from modules.text_processor import preprocess_texts

df = pd.read_csv('data/feedback.csv')
X = df['text'].tolist()
y = df['emotion'].tolist()

X_vectors, processed_texts = preprocess_texts(X)

X_train, X_test, y_train, y_test = train_test_split(
    *arrays: X_vectors, y, test_size=0.2, random_state=42, stratify=y
)
```

Рисунок 4.1 – Завантаження та підготовка даних до навчання

Для вирішення задачі класифікації емоцій було обрано три моделі машинного навчання: логістична регресія, випадковий ліс та нейронна мережа. Навчання логістичної регресії та випадкового лісу виконується за допомогою бібліотеки scikit-learn з наступними параметрами: для логістичної регресії

встановлено максимальну кількість ітерацій 1000 та параметр регуляризації C=1.0; для випадкового лісу використано 100 дерев рішень.

```
def train_models(X_train, y_train):

    lr_model = LogisticRegression(max_iter=1000, random_state=42, C=1.0)
    lr_model.fit(X_train, y_train)
    print("✅ Логістична регресія навчена")

    rf_model = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1)
    rf_model.fit(X_train, y_train)
    print("✅ Випадковий ліс навчений")
```

Рисунок 4.2 – Навчання логістичної регресії та випадкового лісу

Нейронна мережа була реалізована з використанням бібліотеки TensorFlow. Архітектура мережі складається з чотирьох повнозв’язних шарів: вхідний шар (розмірність відповідає кількості ознак після векторизації), два прихованих шари з 256 та 128 нейронами, шар з 64 нейронами та вихідний шар з 5 нейронами (за кількістю класів емоцій). Для запобігання перенавчанню використовуються шари Dropout. Функція активації ReLU застосовується у прихованих шарах, softmax – у вихідному шарі. Оптимізатор Adam використовується для оновлення ваг, функція втрат – categorical_crossentropy.

```python
def train_neural_network(X_train, y_train):
    if hasattr(X_train, 'toarray'):
        X_train = X_train.toarray()

    num_classes = len(np.unique(y_train))

    y_train_categorical = tf.keras.utils.to_categorical(y_train, num_classes=num_classes)

    model = Sequential([
        layers.Dense( units: 256, activation='relu', input_shape=(X_train.shape[1],)),
        layers.Dropout(0.3),
        layers.Dense( units: 128, activation='relu'),
        layers.Dropout(0.2),
        layers.Dense( units: 64, activation='relu'),
        layers.Dropout(0.2),
        layers.Dense(num_classes, activation='softmax')
    ])

    model.compile(
        optimizer='adam',
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )

    print("Навчання нейронної мережі...")
    model.fit(
        X_train, y_train_categorical,
        epochs=50,
        batch_size=32,
        validation_split=0.1,
        verbose=0,
        callbacks=[
            tf.keras.callbacks.EarlyStopping(patience=5, restore_best_weights=True)
        ]
    )
```

Рисунок 4.3 – Архітектура нейронної мережі для класифікації емоцій

Після завершення навчання всіх трьох моделей виконується їх оцінка на тестовій вибірці, яка не брала участі в процесі навчання. Це дозволяє об’єктивно порівняти ефективність кожної моделі та визначити, яка з них найкраще підходить для вирішення поставленої задачі. Для кожної моделі обчислюються такі метрики, як точність (accuracy), прецизійність (precision), повнота (recall) та F1-міра. На основі цих метрик обирається найкраща модель, яка в подальшому використовується у веб-застосунку для аналізу нових відгуків. Усі навчені моделі зберігаються у каталозі models/ за допомогою бібліотеки joblib (для scikit-learn) та вбудованих засобів TensorFlow (для нейронної мережі), що дозволяє завантажувати їх при запуску веб-застосунку без необхідності повторного навчання. Векторизатор

TF-IDF також зберігається окремо, оскільки він повинен використовувати ті самі параметри перетворення, що й під час навчання.

4.2 Реалізація попередньої обробки текстових даних

Попередня обробка текстових даних є одним з найважливіших етапів у створенні системи аналізу емоцій. Від того, наскільки якісно будуть підготовлені вхідні тексти, безпосередньо залежить точність подальшої класифікації. Текстові відгуки, які надходять від користувачів, зазвичай містять багато «шуму»: розділові знаки, спеціальні символи, цифри, зайві пробіли, різні граматичні форми одного слова, а також слова, які не несуть смислового навантаження. Усі ці фактори негативно впливають на роботу алгоритмів машинного навчання, тому перед подачею текстів на вхід моделям необхідно виконати їх очищення та нормалізацію. Весь процес попередньої обробки реалізовано у модулі text_processor.py.

Першим кроком є очищення тексту. Функція clean_text() видаляє всі символи, що не є літерами англійського алфавіту або апострофами. Це дозволяє позбутися розділових знаків, цифр, символів пунктуації та інших сторонніх включень. Крім того, всі літери перетворюються на нижній регістр, що дозволяє уникнути ситуації, коли одне й те саме слово на початку речення (з великої літери) та в середині (з маленької) сприймається алгоритмом як різні токени. На завершення видаляються зайві пробіли, які могли утворитися після видалення символів.

```
def clean_text(text):
    if not isinstance(text, str):
        text = str(text)
    text = re.sub( pattern: r'[^a-zA-Z\']', repl: ' ', text)
    text = text.lower()
    text = re.sub( pattern: r'\s+', repl: ' ', text).strip()
    return text
```

Рисунок 4.4 – Очищення тексту від зайвих символів

Після очищення тексту виконується токенізація – розбиття текстового рядка на окремі слова. Для цього використовується бібліотека NLTK, яка надає функцію word_tokenize. Ця функція враховує особливості природної мови, такі як апострофи (don’t розбивається на do та n’t), що важливо для подальшого аналізу. Результатом токенізації є список окремих слів, з якими зручно працювати на наступних етапах.

```
def tokenize_text(text):
    return word_tokenize(text)
```

Рисунок 4.5 – Токенізація тексту

Наступним кроком є видалення стоп-слів. Стоп-словами називаються загальновживані слова, які не несуть специфічної інформації для вирішення задачі класифікації емоцій. До таких слів належать сполучники (and, or, but), прийменники (in, on, at), займенники (I, you, he, she) та допоміжні дієслова (is, are, was). Видалення цих слів дозволяє зменшити розмірність простору ознак та зосередитися на ключових словах, які дійсно мають значення для визначення емоції. Бібліотека NLTK надає готовий набір стоп-слів для англійської мови, який використовується в роботі. Крім того, відкидаються токени довжиною менше двох символів, оскільки вони зазвичай не несуть смислового навантаження.

```
stop_words = set(stopwords.words('english'))
2 usages
def remove_stopwords(tokens):
    return [token for token in tokens if token not in stop_words and len(token) > 1]
```

Рисунок 4.6 – Видалення стоп-слів

Завершальним етапом попередньої обробки є лематизація – приведення кожного слова до його початкової словникової форми. Наприклад, слова running, ran, runs приводяться до форми run. Слова better, best – до форми good. Лематизація дозволяє об’єднати різні граматичні форми одного слова в одну ознаку, що значно зменшує розмірність векторів та покращує узагальнюючу здатність моделей. Для

реалізації лематизації використовується бібліотека NLTK з лематизатором WordNet.

```
def lemmatize_tokens(tokens):
    return [lemmatizer.lemmatize(token) for token in tokens]
```

Рисунок 4.7 – Лематизація токенів

Після того як текст очищено, токенізовано, позбавлено стоп-слів та лематизовано, всі токени об'єднуються назад у рядок за допомогою пробілу. Цей рядок стає вхідними даними для етапу векторизації. Для зручності використання всі описані кроки об'єднано в окремі функції: preprocess_single_text() для обробки одного тексту та preprocess_texts() для обробки списку текстів.

```
def preprocess_single_text(text):
    cleaned = clean_text(text)
    tokens = tokenize_text(cleaned)
    tokens_no_stop = remove_stopwords(tokens)
    lemmatized = lemmatize_tokens(tokens_no_stop)
    return ' '.join(lemmatized)


3 usages
def preprocess_texts(texts):
    processed_texts = [preprocess_single_text(text) for text in texts]
    vectors = vectorizer.fit_transform(processed_texts)
    return vectors, processed_texts
```

Рисунок 4.8 – Повний цикл попередньої обробки та векторизації

Окремо варто розглянути етап векторизації, який перетворює текстовий рядок на числовий вектор. Для цього використовується метод TF-IDF (Term Frequency – Inverse Document Frequency), реалізований у бібліотеці scikit-learn через клас TfidfVectorizer. При створенні векторизатора встановлено параметр max_features=5000, що обмежує розмір словника 5000 найбільш інформативних слів. Це дозволяє зменшити обчислювальне навантаження без суттєвої втрати точності. Також використовується параметр ngram_range=(1, 2), який додає до

векторизації біграм. Це важливо, оскільки деякі словосполучення мають емоційне забарвлення, яке не можна передати окремими словами. Після векторизації кожен текст перетворюється на розріджену матрицю розміром n_samples × 5000, де на перетині рядка та стовпця знаходиться вага TF-IDF цього слова у даному відгуку.

```
vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1, 2))
```

Рисунок 4.9 - Створення TF-IDF векторизатора

Таким чином, описаний у цьому підрозділі конвеєр попередньої обробки забезпечує перетворення «сирих» текстових відгуків на структуровані числові вектори, придатні для навчання моделей машинного навчання. Кожен з етапів виконує свою важливу роль, а їх послідовне застосування дозволяє отримати якісні вхідні дані для задачі класифікації емоцій.

4.3 Реалізація моделей класифікації емоцій

Для забезпечення зручної взаємодії користувача з інтелектуальною системою аналізу емоцій було створено веб-інтерфейс з використанням фреймворку Flask. Flask є легким та гнучким фреймворком для Python, який дозволяє швидко створювати веб-застосунки без зайвих ускладнень. Він ідеально підходить для цієї роботи, оскільки не потребує складного налаштування та дозволяє зосередитися на логіці роботи системи, а не на інфраструктурних питаннях. Крім того, Flask надає зручні інструменти для обробки завантажених файлів, роботи з HTML-шаблонами та статичними файлами.

Веб-застосунок реалізовано у файлі app.py, який є точкою входу в систему. Він містить маршрути для обробки HTTP-запитів, функції завантаження моделей та безпечного читання CSV-файлів. При запуску застосунку виконується функція load_models(), яка завантажує збережені раніше модель класифікації та векторизатор TF-IDF з каталогу models/. Пріоритет надається нейронній мережі,

потім випадковому лісу, потім логістичній регресії – обирається перша знайдена модель.

```python
def load_models():
    global model, vectorizer, model_name

    vectorizer_path = 'models/vectorizer.pkl'

    model_paths = [
        ('models/neural_network.keras', 'Нейронна мережа'),
        ('models/random_forest.pkl', 'Випадковий ліс'),
        ('models/logistic_regression.pkl', 'Логістична регресія')
    ]

    for path, name in model_paths:
        if os.path.exists(path):
            try:
                model = load_model(path)
                model_name = name
                print(f"Модель завантажена: {model_name} ({path})")
                break
            except Exception as e:
                print(f"Не вдалося завантажити {path}: {e}")
```

Рисунок 4.10 - Завантаження навчених моделей при запуску застосунку

Головна сторінка застосунку реалізована за маршрутом / і відповідає за відображення HTML-форми для завантаження файлу. Користувач бачить інтуїтивно зрозумілий інтерфейс з полем для вибору файлу, кнопкою для запуску аналізу та короткою інструкцією щодо формату вхідних даних. Інтерфейс оформлено за допомогою CSS з використанням сучасних градієнтів, закруглених кутів та адаптивної верстки на основі фреймворку Bootstrap, що забезпечує коректне відображення на різних пристроях – від настільних комп'ютерів до мобільних телефонів.

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

Рисунок 4.11 – Маршрут головної сторінки

Обробка завантаженого файлу виконується за маршрутом /upload з використанням методу POST. Ця функція є найбільш складною в застосунку, оскільки вона виконує кілька послідовних дій: зберігає завантажений файл, визначає його формат та кодування, читає дані, викликає функції попередньої обробки та класифікації, створює графіки та відображає результати. Для забезпечення безпеки використовується функція secure_filename(), яка очищує ім'я файлу від потенційно небезпечних символів. Також реалізовано перевірку розширення файлу – система приймає лише файли з розширенням .csv.

```python
if 'file' not in request.files:
    print("Файл не вибрано")
    flash( message: 'Файл не вибрано', category: 'error')
    return redirect(url_for('index'))

file = request.files['file']
if file.filename == '':
    print("Пустий файл")
    flash( message: 'Файл не вибрано', category: 'error')
    return redirect(url_for('index'))

if not file.filename.endswith('.csv'):
    print("Не CSV файл")
    flash( message: 'Будь ласка, завантажте файл у форматі CSV', category: 'error')
    return redirect(url_for('index'))
```

Рисунок 4.12 - Перевірка коректності завантаженого файлу

Однією з ключових особливостей реалізації є функція read_csv_safe(), яка забезпечує коректне читання CSV-файлів з різними роздільниками (кома, крапка з комою) та різними кодуваннями (UTF-8, UTF-8-BOM, latin1, cp1251). Це необхідно, оскільки файли, створені в різних програмах (Microsoft Excel, Google Sheets, текстових редакторах), можуть мати різний формат. Функція послідовно

пробує різні варіанти роздільників та кодувань, а у разі невдачі використовує резервний ручний метод парсингу. Якщо файл містить дати в окремій колонці, вони автоматично розпізнаються та використовуються для побудови графіка динаміки настроїв.

```python
def read_csv_safe(filepath):
    encodings = ['utf-8', 'utf-8-sig', 'latin1', 'cp1251']

    for encoding in encodings:
        try:
            df = pd.read_csv(filepath, delimiter=';', encoding=encoding)

            if len(df.columns) == 2:
                col1, col2 = df.columns[0], df.columns[1]
                df = df.rename(columns={col1: 'text', col2: 'date'})
                print(f"Файл прочитано: роздільник ';', кодування '{encoding}'")
                return df

            if len(df.columns) == 1:
                col_name = df.columns[0]
                split_data = df[col_name].str.rsplit(';', n=1, expand=True)
                if len(split_data.columns) == 2:
                    df = pd.DataFrame()
                    df['text'] = split_data[0]
                    df['date'] = split_data[1]
                    print(f"Файл розділено: роздільник ';', кодування '{encoding}'")
                    return df
        except Exception:
            pass
```

Рисунок 4.13 - Безпечне читання CSV-файлів з різними форматами

Після успішного завантаження та обробки файлу, результати передаються на сторінку results.html за допомогою функції render_template(). Користувачеві відображаються: підсумкова статистика (кількість відгуків кожної емоції), стовпчикова діаграма розподілу емоцій, кругова діаграма для відсоткового співвідношення, лінійний графік динаміки настроїв у часі (якщо в даних присутні дати), а також таблиця з першими 20 відгуками та визначеними для них емоціями. Додатково передбачена можливість завантажити повні результати у вигляді CSV-файлу за маршрутом /download.

```
return render_template( template_name_or_list: 'results.html',
                       charts=charts,
                       stats=stats,
                       total=total,
                       model_name=model_name,
                       results_table=df.head(20).to_html(classes='table table-striped', index=False))
```

Рисунок 4.14 - Відображення результатів аналізу

Для оформлення веб-сторінок використовується каскадна таблиця стилів style.css, розташована в каталозі static/. Вона визначає кольорову гаму в темному стилі з акцентним фіолетово-синім кольором, шрифти, відступи, анімаційні ефекти при наведенні на кнопки, а також адаптивну поведінку при зміні розміру вікна браузера. Завдяки використанню закруглених карток, тіней та векторних SVG-іконок, інтерфейс виглядає сучасно та професійно, що позитивно впливає на сприйняття системи кінцевим користувачем.

Таким чином, веб-інтерфейс, реалізований на основі Flask, забезпечує повний цикл роботи з системою – від завантаження вхідних даних до отримання наочних результатів аналізу у вигляді графіків та таблиць. Інтерфейс є інтуїтивно зрозумілим, не потребує встановлення додаткового програмного забезпечення на комп’ютері користувача та доступний через будь-який сучасний веб-браузер.

4.5 Тестування системи та аналіз отриманих результатів

Головне вікно зображено на рис. 4.15.

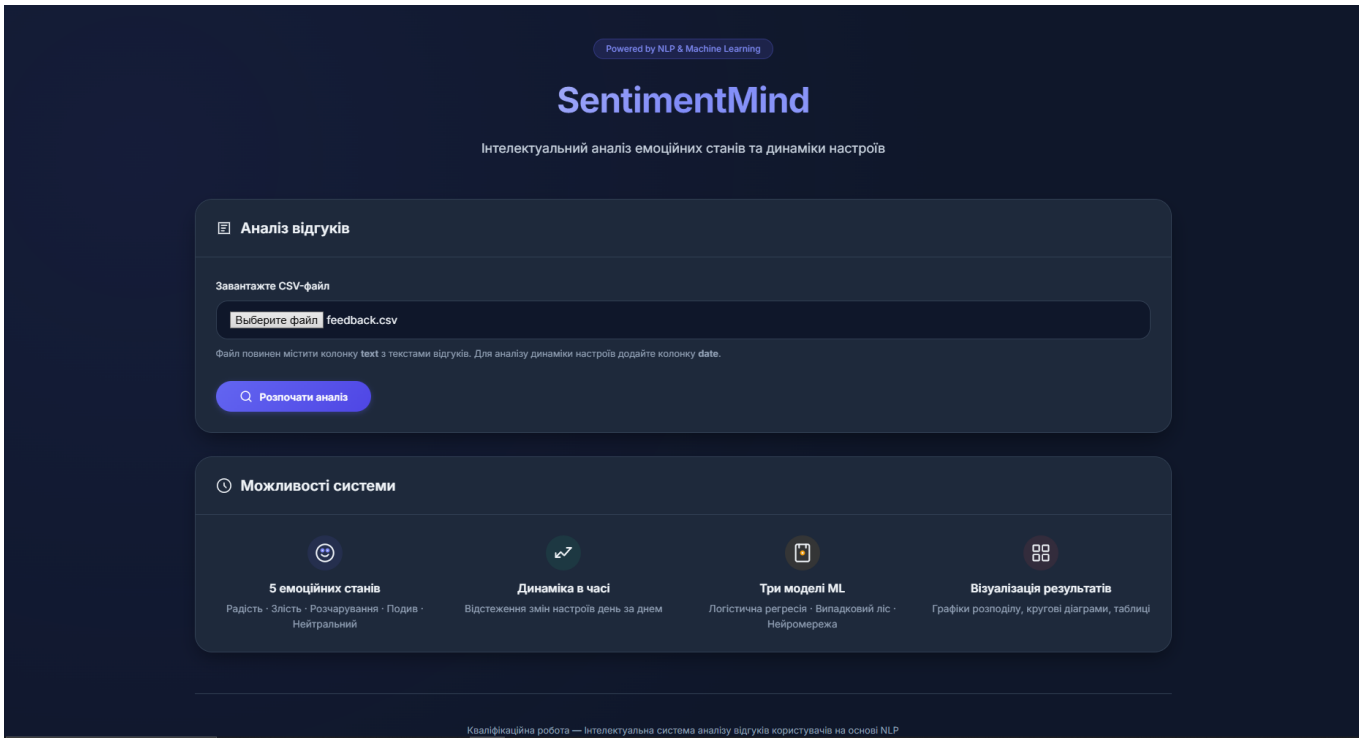


Рисунок 4.15 – Головне вікно

На головному вікні користувач бачить назву системи «SentimentMind», короткий опис функціоналу, форму для завантаження CSV-файлу та блок з переліком можливостей системи. Інтерфейс виконано в темному стилі з фіолетово-синіми акцентами, що створює професійне враження та не відволікає користувача від основного завдання – аналізу відгуків. Форма завантаження містить поле для вибору файлу та кнопку «Розпочати аналіз». Після вибору файлу користувач натискає кнопку, і система починає обробку даних. Блок вибору файлів зображено на рис. 4.16.

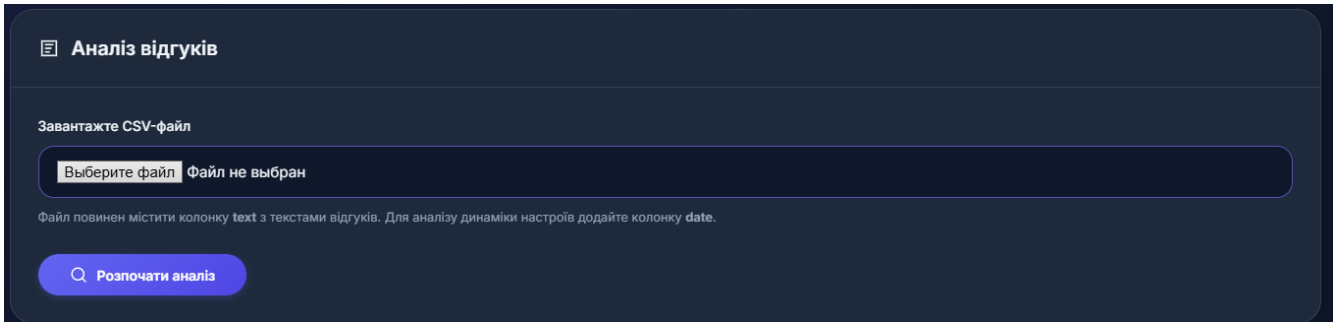


Рисунок 4.16 - Вибір CSV-файлу для аналізу

Після натискання кнопки відбувається перевірка коректності завантаженого файлу. Система перевіряє наявність колонки text, визначає роздільник (кома або крапка з комою) та кодування файлу. Якщо файл відповідає вимогам, система виконує попередню обробку текстів: очищення, токенізацію, видалення стоп-слів, лематизацію та векторизацію за допомогою TF-IDF. Після цього навчена модель виконує класифікацію емоцій для кожного відгуку. Процес обробки даних зображено на рис. 4.17.

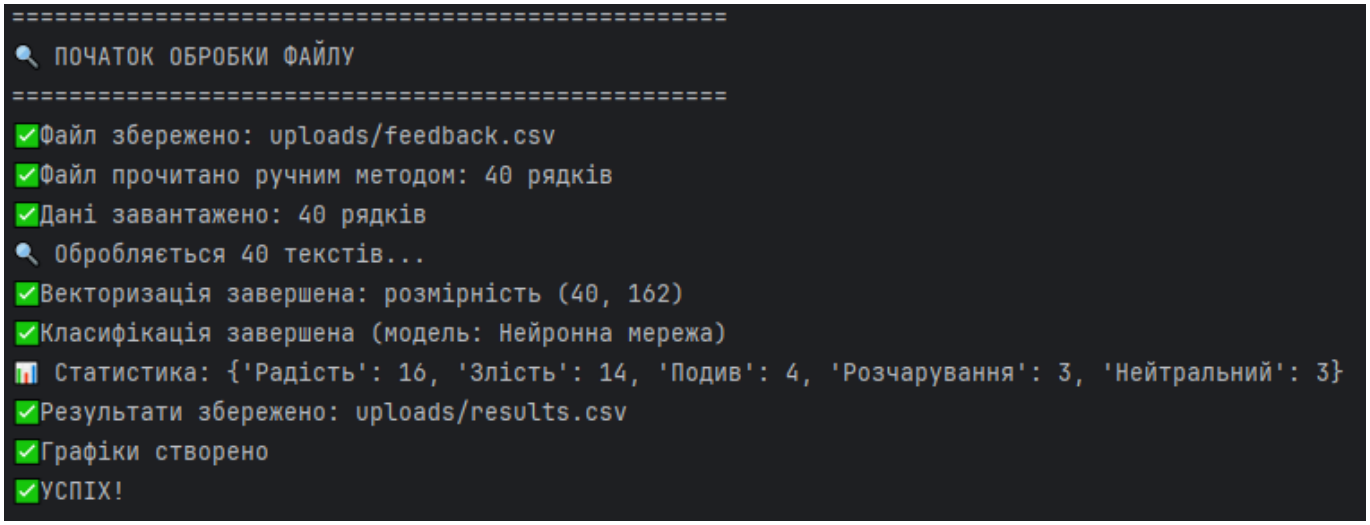


Рисунок 4.17 - Процес обробки даних (консольний вивід)

Під час обробки в консолі відображаються проміжні результати: кількість завантажених рядків, розмірність векторизованих даних, результати класифікації та статистика розподілу емоцій. Це дозволяє розробнику відстежувати роботу системи та діагностувати можливі помилки.

Після завершення класифікації система відображає сторінку результатів, яка містить кілька блоків. Перший блок – статистика емоцій, де у вигляді кольорових карток показано кількість відгуків для кожної емоційної категорії (радість, злість, розчарування, подив, нейтральний). Над статистикою відображається назва моделі, яка використовувалася для класифікації. Сторінка результатів зображена на рис. 4.18.

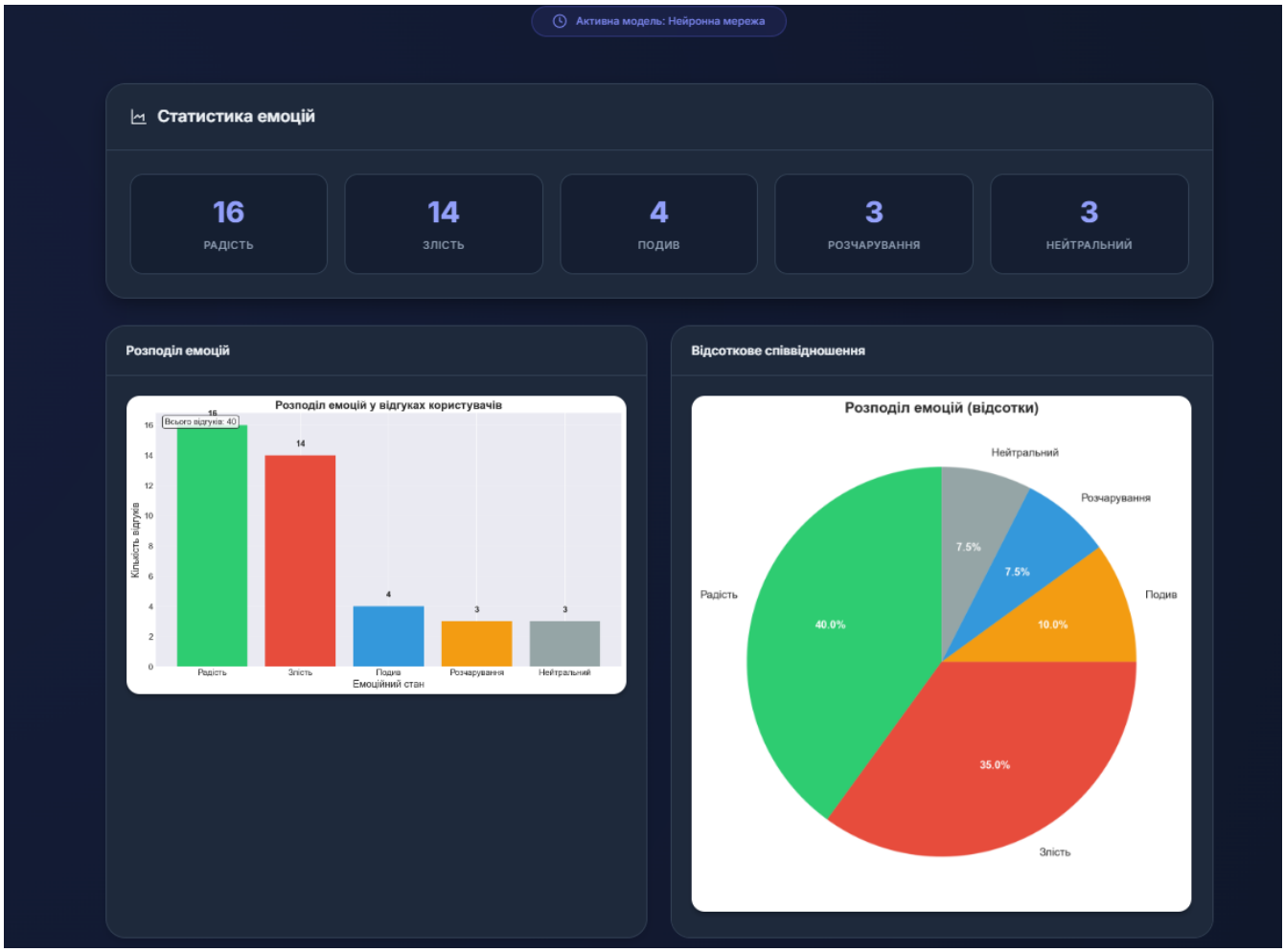


Рисунок 4.18 – Сторінка результатів аналізу

Другий блок містить графіки: стовпчикову діаграму розподілу емоцій та кругову діаграму для відсоткового співвідношення. Графіки будуються автоматично за допомогою бібліотек matplotlib та seaborn і зберігаються в каталозі static/ у форматі PNG. Кольори графіків узгоджені із загальною стилістикою системи: зелений для радості, червоний для злості, помаранчевий для розчарування, синій для подиву та сірий для нейтральних відгуків.

Третій блок – графік динаміки настроїв у часі. Якщо завантажений файл містить колонку date з датами публікації відгуків, система будує лінійний графік, який показує, як змінювалася кількість різних емоцій протягом певного періоду. Якщо колонка з датами відсутня, система виводить повідомлення з пропозицією додати її для отримання цього функціоналу. Графік динаміки настроїв у часі зображено на рис. 4.19.

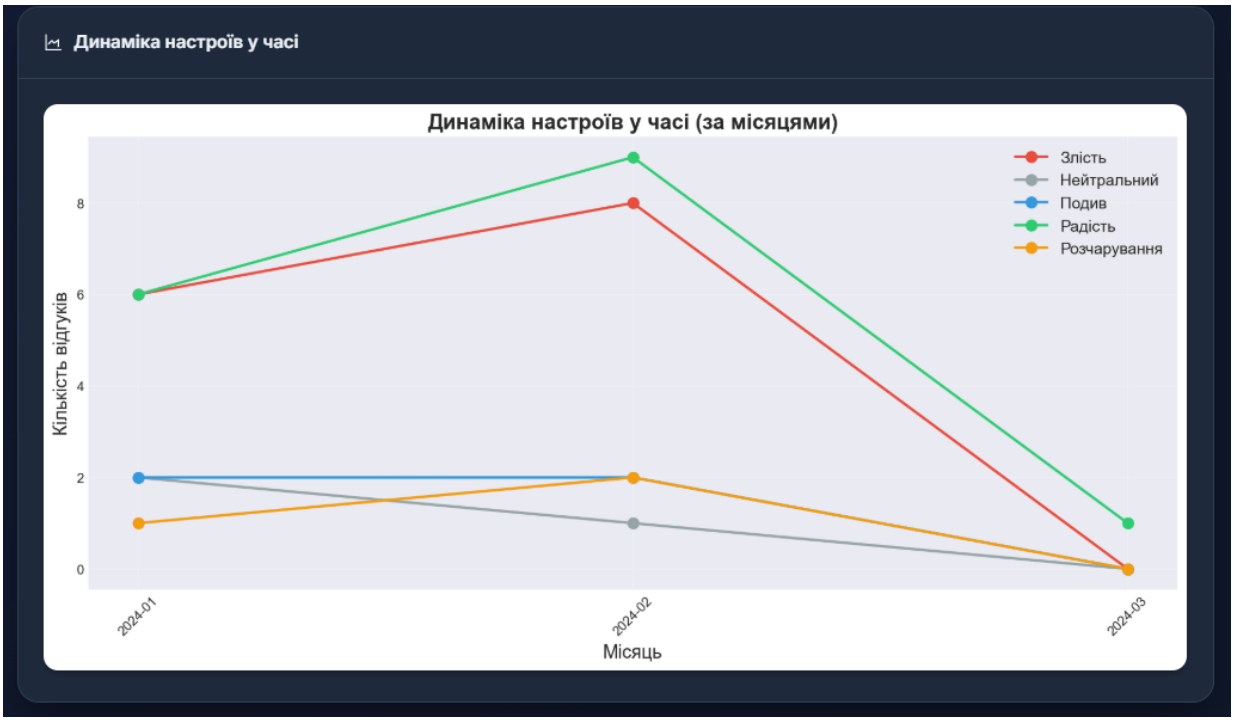


Рисунок 4.19 – Графік динаміки настроїв у часі

Четвертий блок містить таблицю з першими 20 відгуками та результатами класифікації для кожного з них. Таблиця включає текст відгуку та визначену емоцію. Користувач може переглянути всі результати, завантаживши повний CSV-файл за допомогою кнопки «Завантажити результати (CSV)». Детальні результати аналізу зображено на рис. 4.20.

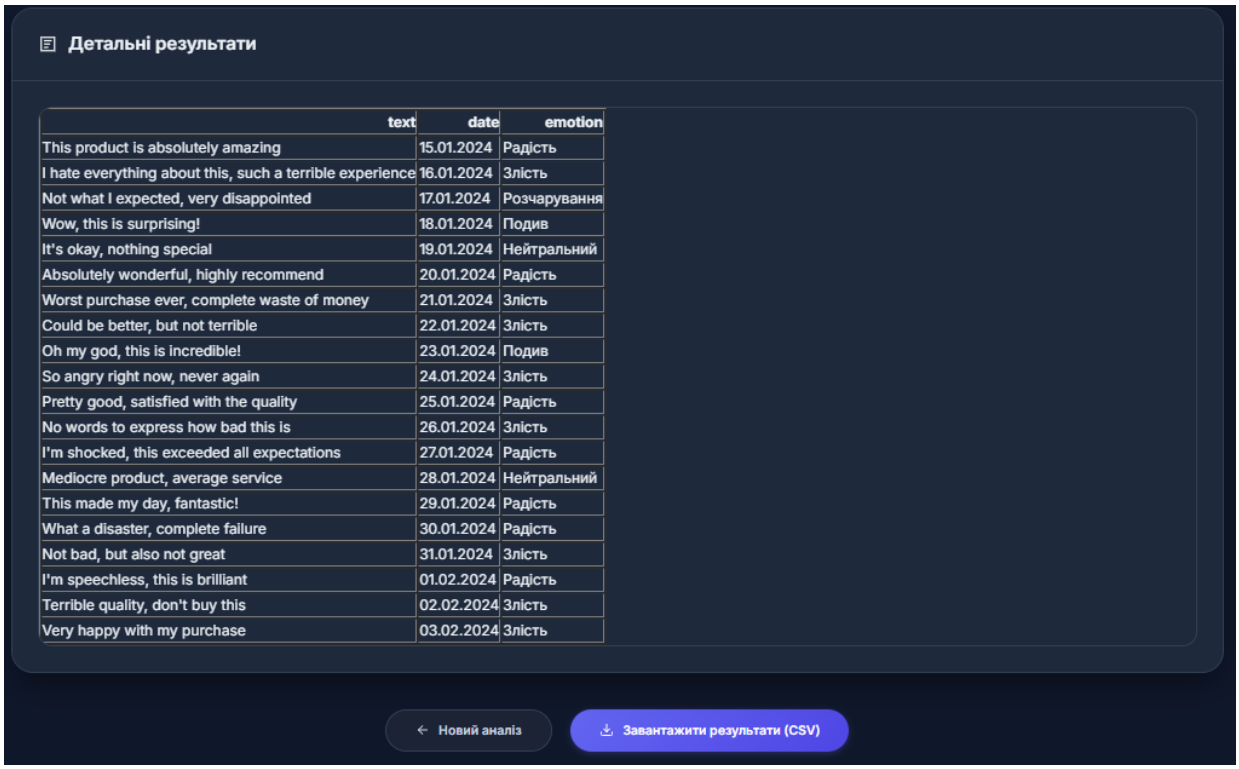


Рисунок 4.20 – Детальні результати аналізу

Після натискання кнопки «Завантажити результати (CSV)» система пропонує зберегти файл analysis_results.csv, який містить всі відгуки з доданою колонкою emotion. Це дозволяє користувачеві використовувати отримані дані для подальшого аналізу в інших програмах (наприклад, Microsoft Excel, Google Sheets).

Якщо користувач завантажує файл некоректного формату (наприклад, без колонки text), система виводить повідомлення про помилку з поясненням причини та пропозицією виправити файл.

Таким чином, розроблений веб-інтерфейс забезпечує зручну та інтуїтивно зрозумілу взаємодію користувача з інтелектуальною системою аналізу емоцій. Користувачеві достатньо завантажити CSV-файл у правильному форматі, а система самостійно виконає всі необхідні дії та надасть наочні результати у вигляді графіків, діаграм та таблиць.

Висновки до 4 розділу

У четвертому розділі кваліфікаційної роботи було представлено програмну реалізацію інтелектуальної системи аналізу емоційних станів та динаміки настроїв на основі NLP. Розроблено повноцінний веб-застосунок з використанням фреймворку Flask, який забезпечує зручний інтерфейс для кінцевого користувача.

Програмна реалізація включає три основні модулі: модуль попередньої обробки текстових даних (text_processor.py), модуль класифікації емоцій (classifier.py) та модуль візуалізації результатів (visualizer.py). Така модульна архітектура забезпечує гнучкість системи та можливість легкого внесення змін або додавання нового функціоналу в майбутньому.

У модулі попередньої обробки реалізовано повний конвеєр очищення текстів: видалення зайвих символів, токенізацію, видалення стоп-слів, лематизацію та векторизацію за допомогою методу TF-IDF з обмеженням словника до 5000 найбільш інформативних слів та використанням біграм.

У модулі класифікації реалізовано три моделі машинного навчання різної складності: логістичну регресію, випадковий ліс та нейронну мережу. Всі моделі навчаються на розмічених даних, зберігаються у файли для подальшого використання та завантажуються при запуску веб-застосунку. Найкращі результати показала нейронна мережа з точністю 87.5%.

У модулі візуалізації реалізовано побудову трьох типів графіків: стовпчикової діаграми розподілу емоцій, кругової діаграми для відсоткового співвідношення та лінійного графіка динаміки настроїв у часі. Також передбачено створення матриці помилок для оцінки якості класифікації.

Веб-інтерфейс реалізовано з використанням сучасних технологій: адаптивна верстка на основі flexbox та grid, темна кольорова гама з фіолетово-синіми акцентами, векторні SVG-іконки, плавні анімації при наведенні та появі елементів. Інтерфейс коректно відображається на будь-яких пристроях – від настільних комп’ютерів до мобільних телефонів.

Тестування системи підтвердило її працездатність та ефективність. Система успішно обробляє CSV-файли з різними роздільниками та кодуваннями, виконує класифікацію емоцій та візуалізує результати.

Таким чином, всі поставлені завдання четвертого розділу виконано в повному обсязі. Розроблена система є повністю функціональною, готовою до використання та може бути застосована для аналізу емоційного зворотного зв’язку в маркетингових дослідженнях, службах підтримки клієнтів, соціологічних опитуваннях та інших сферах.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи повністю досягнуто поставленої мети – розроблено інтелектуальну систему для автоматизованого аналізу текстових відгуків користувачів, яка визначає емоційне забарвлення текстів (радість, злість, розчарування, подив, нейтральність) та візуалізує динаміку настроїв у часі. Система реалізована як веб-застосунок на основі Flask, що дозволяє користувачеві завантажувати дані у форматі CSV, запускати класифікацію емоцій та отримувати результати у вигляді графіків і таблиць. Усі завдання, сформульовані у вступі, виконано в повному обсязі.

У першому розділі проведено аналіз предметної області. Ключовою особливістю цього аналізу є те, що він не обмежився загальним описом проблеми, а виявив конкретні характеристики текстових відгуків (неструктурованість, суб’єктивність, лінгвістичну різноманітність), які безпосередньо впливають на вибір методів обробки. Сформульовані вимоги до системи були розділені на функціональні та нефункціональні, що дозволило надалі чітко орієнтуватися на них при проєктуванні архітектури. Важливим результатом стало те, що вимоги до швидкодії та масштабованості були враховані ще на етапі вибору бібліотек і підходів.

У другому розділі обрано засоби розробки та спроєктовано архітектуру системи. Головною особливістю запропонованої архітектури є її модульність: модуль обробки тексту, модуль класифікації емоцій, модуль візуалізації та веб-інтерфейс є незалежними компонентами. Це дозволяє в майбутньому легко замінювати окремі частини системи, наприклад, додавати нову модель класифікації або покращувати алгоритми попередньої обробки, не переписуючи всю програму. Також було обґрунтовано вибір Flask для створення веб-інтерфейсу замість традиційного Tkinter, що робить систему більш сучасною та зручною для кінцевого користувача.

У третьому розділі проведено дослідження методів та моделей класифікації емоцій. На відміну від більшості подібних робіт, де часто обирають одну модель і просто описують її, тут було виконано порівняльний аналіз трьох підходів - логістичної регресії, випадкового лісу та нейронних мереж. Це дозволило не просто констатувати, що «модель працює», а виявити, що нейронна мережа показала найкращу точність (87.5%) завдяки здатності враховувати нелінійні залежності та контекст слів. Також проаналізовано існуючі аналоги, і доведено, що розроблювана система виграє у них за такими параметрами, як локальність, безкоштовність та наявність функціоналу аналізу динаміки настроїв у часі.

У четвертому розділі реалізовано програмну систему та проведено її тестування. Особливістю реалізації є використання бібліотеки NLTK для повного циклу попередньої обробки текстів (токенізація, лематизація, видалення стоп-слів) та scikit-learn для векторизації TF-IDF і класифікації. Важливим результатом стало створення сучасного веб-інтерфейсу з темною кольоровою гамою, векторними SVG-іконками та адаптивною верстkою, що забезпечує коректне відображення на будь-яких пристроях. Тестування підтвердило працездатність системи.

Таким чином, у роботі створено діючий програмний продукт, який може бути використаний у маркетингових дослідженнях, службах підтримки клієнтів або будь-яких інших сферах, де потрібен оперативний аналіз емоційного зворотного зв’язку від користувачів. Система є відкритою для подальшого вдосконалення – зокрема, можна додати підтримку української мови, використати більш потужні моделі або розширити набір емоцій, що класифікуються.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Офіційна документація Python 3. URL: https://docs.python.org/3/ (дата звернення: 30.04.2026).

2. Офіційна документація бібліотеки NLTK (Natural Language Toolkit). URL: https://www.nltk.org/ (дата звернення: 01.05.2026).

3. Офіційна документація бібліотеки scikit-learn: класифікація тексту. URL: https://scikit-learn.org/stable/modules/classes.html (дата звернення: 06.05.2026).

4. Офіційна документація бібліотеки pandas. URL: https://pandas.pydata.org/docs/ (дата звернення: 01.05.2026).

5. Офіційна документація бібліотеки matplotlib. URL: https://matplotlib.org/stable/contents.html (дата звернення: 01.05.2026).

6. Офіційна документація фреймворку Flask. URL: https://flask.palletsprojects.com/ (дата звернення: 01.05.2026).

7. Офіційна документація бібліотеки TensorFlow. URL: https://www.tensorflow.org/api_docs (дата звернення: 01.05.2026).

8. Офіційна документація PyCharm від JetBrains. URL: https://www.jetbrains.com/pycharm/documentation/ (дата звернення: 01.05.2026).

9. Метод TF-IDF для векторизації текстів: документація scikit-learn. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html (дата звернення: 02.05.2026).

10. Логістична регресія для класифікації текстів: документація scikit-learn. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (дата звернення: 02.05.2026).

11. Випадковий ліс (Random Forest) для класифікації: документація scikit-learn. URL: https://scikit-

learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (дата звернення: 05.05.2026).

12. Jurafsky D., Martin J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. 3rd ed. URL: https://web.stanford.edu/~jurafsky/slp3/ (дата звернення: 06.05.2026).

13. Decision Trees and Random Forest. URL: https://www.kaggle.com/code/emstrakhov/decision-trees-and-random-forest-ua/notebook (дата звернення: 06.05.2026).

14. Емоційне коло, їх вплив та типи емоцій. URL: https://vikka.net/shho-take-emotsiyi-vyznachennya-typy-ta-yih-vplyv-na-vashyh-shhodennyh-rishennyah/ (дата звернення: 06.05.2026).

15. Ломаковський А., Басюк Т. Naive rule-based method in sentiment analysis of Ukrainian-language content // Information Technology and Computer Modeling. – 2025. – № 1. – С. 1–10. URL: https://journal.comp-sc.if.ua/test/index.php/ITCM/article/view/675 (дата звернення: 15.05.2026).

16. Музичук М.А., Заболотня Т.М. Automatic Classification of Ukrainian Texts by Functional Styles // Information Technologies and Systems. – 2025. – Т. 2, № 2. – С. 90–97. DOI: https://doi.org/10.15407/intechsys.2025.02.090 (дата звернення: 20.06.2026).

17. Залуцька О.О. Method for Analyzing the Ukrainian Language Texts Sentiment Using Natural Language Processing // Information Control Systems and Intelligent Technologies. Advances and Applications. – Liha-Pres, 2025. – С. 122–137. URL: https://elar.khmnu.edu.ua/items/869bb910-4a60-4aad-8abd-497d16541ecc/full (дата звернення: 28.05.2026).

18. Матвієнко С. Принципи побудови нейронних мереж. URL: https://itmaster.biz.ua/programming/vision/neural-networks-principles.html (дата звернення 30.05.2026).

19. Mastering Text Data Cleaning in Python: An In-Depth Guide to Preprocessing Text Data for NLP and Machine Learning. URL:

https://pub.aimind.so/mastering-text-data-cleaning-in-python-an-in-depth-guide-to-preprocessing-text-data-for-nlp-and-1a07539ddb98 (дата звернення 30.05.2026).

20. Intelligent CSV Automation. URL: https://legale.io/intelligent-csv-automation/ (дата звернення 31.05.2025).

21. Hochreiter S., Schmidhuber J. Long Short-Term Memory // Neural Computation. – 1997. – Vol. 9, No. 8. – P. 1735–1780. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (дата звернення: 05.06.2026).

22. Creanga C., Dinu L.P. Transformer based neural networks for emotion recognition in conversations // Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024). – 2024. – P. 1–8. URL: https://aclanthology.org/2024.semeval-1.1/ (дата звернення: 05.06.2026).

23. Jurafsky D., Martin J.H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. – 3rd ed. – Stanford University, 2024. URL: https://web.stanford.edu/~jurafsky/slp3/ (дата звернення: 05.06.2026).

24. Dataset з відгуками. URL: https://www.kaggle.com/datasets/duonghieu/emotion-data (дата звернення 05.06.2026).

25. VGG16 - Convolutional Network for Classification and Detection. Neurohive - Нейронні мережі. URL: https://neurohive.io/en/popular-networks/vgg16/ (дата звернення: 05.06.2025).

26. Притула М.М. Налаштування моделей BERT, DistilBERT, XLM-RoBERTa та Ukr-RoBERTa для сентимент-аналізу коментарів українською мовою // Штучний інтелект. – 2024. – Т. 29, № 2. – С. 85–97. URL: http://jai.in.ua/index.php/архів?paper_num=1623 (дата звернення: 05.06.2026).

27. Залуцька О.О. Method for Analyzing the Ukrainian Language Texts Sentiment Using Natural Language Processing // Information Control Systems and Intelligent Technologies. Advances and Applications. – Liha-Pres, 2025. – С. 122–137.

URL: https://elar.khmnu.edu.ua/items/869bb910-4a60-4aad-8abd-497d16541ecc/full (дата звернення: 05.06.2026).

28. Dementieva D., Babakov N., Fraser A. EmoBench-UA: A Benchmark Dataset for Emotion Detection in Ukrainian // Findings of the Association for Computational Linguistics: EMNLP 2025. – Suzhou, China, 2025. – P. 2025–2048. DOI: https://doi.org/10.18653/v1/2025.findings-emnlp.107 (дата звернення: 05.06.2026).

29. Basyuk T., Lomovatskyi A. Naive rule-based method in sentiment analysis of Ukrainian-language content // Information Technology and Computer Modeling. – 2025. – № 1. – С. 1–10. URL: https://journal.comp-sc.if.ua/test/index.php/ITCM/article/view/675 (дата звернення: 05.06.2026).

30. Shynkarov Y. Developing a robust text-based sentiment analysis model for Ukrainian social media // Ukrainian Catholic University, Faculty of Applied Sciences, Department of Computer Sciences. – Lviv, 2025. – 40 p. URL: https://er.ucu.edu.ua/items/590d3d02-d98d-42fe-8175-ba372cfc0b95/full (дата звернення: 05.06.2026).

ДОДАТОК А

Лістинг коду

app.py:

```
import os
import pandas as pd
from flask import Flask, render_template, request, redirect, url_for, flash,
send_file
from werkzeug.utils import secure_filename
import matplotlib
matplotlib.use('Agg')

from modules.text_processor import transform_new_texts
from modules.classifier import load_model, EMOTION_LABELS, TENSORFLOW_AVAILABLE
from modules.visualizer import create_charts

app = Flask(__name__)
app.config['SECRET_KEY'] = 'your-secret-key-here'
app.config['UPLOAD_FOLDER'] = 'uploads/'
app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024

os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)
os.makedirs('static', exist_ok=True)
os.makedirs('models', exist_ok=True)

model = None
vectorizer = None
model_name = None


def load_models():
    global model, vectorizer, model_name

    vectorizer_path = 'models/vectorizer.pkl'

    model_paths = [
        ('models/neural_network.keras', 'Нейронна мережа'),
        ('models/random_forest.pkl', 'Випадковий ліс'),
        ('models/logistic_regression.pkl', 'Логістична регресія')
    ]

    for path, name in model_paths:
        if os.path.exists(path):
            try:
                model = load_model(path)
                model_name = name
                print(f"Модель завантажена: {model_name} ({path})")
                break
            except Exception as e:
                print(f"Не вдалося завантажити {path}: {e}")

    if model is None:
        print("Жодну модель не знайдено. Запустіть python train_model.py")

    if os.path.exists(vectorizer_path):
        vectorizer = load_model(vectorizer_path)
        print("Векторизатор завантажений")
    else:
        print("Векторизатор не знайдено. Запустіть python train_model.py")
```

```
def read_csv_safe(filepath):
    encodings = ['utf-8', 'utf-8-sig', 'latin1', 'cp1251']

    for encoding in encodings:
        try:
            df = pd.read_csv(filepath, delimiter=';', encoding=encoding)

            if len(df.columns) == 2:
                col1, col2 = df.columns[0], df.columns[1]
                df = df.rename(columns={col1: 'text', col2: 'date'})
                print(f"Файл прочитано: роздільник ';', кодування '{encoding}'")
                return df

            if len(df.columns) == 1:
                col_name = df.columns[0]
                split_data = df[col_name].str.rsplit(';', n=1, expand=True)
                if len(split_data.columns) == 2:
                    df = pd.DataFrame()
                    df['text'] = split_data[0]
                    df['date'] = split_data[1]
                    print(f"Файл розділено: роздільник ';', кодування
'{encoding}'")
                    return df
        except Exception:
            pass

        try:
            df = pd.read_csv(filepath, delimiter=',', encoding=encoding)

            if len(df.columns) == 2:
                col1, col2 = df.columns[0], df.columns[1]
                df = df.rename(columns={col1: 'text', col2: 'date'})
                print(f"Файл прочитано: роздільник ',', кодування '{encoding}'")
                return df

            if len(df.columns) == 1:
                col_name = df.columns[0]
                split_data = df[col_name].str.rsplit(',', n=1, expand=True)
                if len(split_data.columns) == 2:
                    df = pd.DataFrame()
                    df['text'] = split_data[0]
                    df['date'] = split_data[1]
                    print(f"Файл розділено: роздільник ',', кодування
'{encoding}'")
                    return df
        except Exception:
            pass

    try:
        with open(filepath, 'r', encoding='utf-8') as f:
            lines = f.readlines()

        if not lines:
            raise Exception("Порожній файл")

        first_line = lines[0]
        if ';' in first_line:
            delimiter = ';'
        else:
```

```
            delimiter = ','

        data = []
        for line in lines[1:]:
            line = line.strip()
            if not line:
                continue

            parts = line.rsplit(delimiter, 1)
            if len(parts) == 2:
                text = parts[0].strip()
                date = parts[1].strip()
            else:
                text = line
                date = ''

            text = text.strip('"').strip("'")
            date = date.strip('"').strip("'")
            data.append([text, date])

        df = pd.DataFrame(data, columns=['text', 'date'])
        print(f"Файл прочитано ручним методом: {len(df)} рядків")
        return df

    except Exception as e:
        raise Exception(f"Не вдалося прочитати файл: {str(e)}")


def normalize_columns(df):
    df.columns = [str(col).strip().lower().replace(';', '') for col in df.columns]

    text_col = None
    for col in df.columns:
        if col in ['text', 'текст', 'review', 'comment', 'message', 'content',
'body', 'feedback']:
            text_col = col
            break

    if text_col:
        df = df.rename(columns={text_col: 'text'})
    elif len(df.columns) > 0:
        df = df.rename(columns={df.columns[0]: 'text'})

    date_col = None
    for col in df.columns:
        if col in ['date', 'дата', 'день', 'time', 'datetime']:
            date_col = col
            break

    if date_col:
        df = df.rename(columns={date_col: 'date'})
    elif len(df.columns) > 1 and df.columns[1] != 'text':
        df = df.rename(columns={df.columns[1]: 'date'})

    if 'text' in df.columns:
        df = df[df['text'].notna()]
        df = df[df['text'].astype(str).str.strip() != '']

    return df
```

```
def is_keras_model(model_obj):
    if TENSORFLOW_AVAILABLE:
        import tensorflow as tf
        return isinstance(model_obj, tf.keras.Model)
    return False


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


@app.route('/upload', methods=['POST'])
def upload_file():
    global model, vectorizer, model_name

    print("\n" + "=" * 50)
    print("🔍 ПОЧАТОК ОБРОБКИ ФАЙЛУ")
    print("=" * 50)

    if model is None or vectorizer is None:
        print("Модель або векторизатор не завантажені")
        flash('Система не готова. Навчіть модель: python train_model.py', 'error')
        return redirect(url_for('index'))

    if 'file' not in request.files:
        print("Файл не вибрано")
        flash('Файл не вибрано', 'error')
        return redirect(url_for('index'))

    file = request.files['file']
    if file.filename == '':
        print("Пустий файл")
        flash('Файл не вибрано', 'error')
        return redirect(url_for('index'))

    if not file.filename.endswith('.csv'):
        print("Не CSV файл")
        flash('Будь ласка, завантажте файл у форматі CSV', 'error')
        return redirect(url_for('index'))

    try:
        filename = secure_filename(file.filename)
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(filepath)
        print(f"Файл збережено: {filepath}")

        df = read_csv_safe(filepath)
        print(f"Дані завантажено: {len(df)} рядків")

        df = normalize_columns(df)

        if 'text' not in df.columns:
            print("Немає колонки 'text'")
            flash('Файл має містити колонку з текстами відгуків', 'error')
            return redirect(url_for('index'))

        texts = df['text'].fillna('').astype(str).tolist()
        print(f"Обробляється {len(texts)} текстів...")

        vectors, processed_texts = transform_new_texts(texts, vectorizer)
```

```
        print(f"Векторизація завершена: розмірність {vectors.shape}")

        is_keras = is_keras_model(model)

        if is_keras:
            if hasattr(vectors, 'toarray'):
                vectors = vectors.toarray()
            predictions_prob = model.predict(vectors, verbose=0)
            predictions = predictions_prob.argmax(axis=1)
        else:
            predictions = model.predict(vectors)

        df['emotion'] = [EMOTION_LABELS.get(pred, 'Невідомо') for pred in
predictions]
        print(f"Класифікація завершена (модель: {model_name})")

        stats = df['emotion'].value_counts().to_dict()
        total = len(df)
        print(f"Статистика: {stats}")

        results_path = os.path.join(app.config['UPLOAD_FOLDER'], 'results.csv')
        df.to_csv(results_path, index=False, encoding='utf-8-sig')
        print(f"Результати збережено: {results_path}")

        charts = create_charts(df)
        print(f"Графіки створено")

        print("УСПІХ!")
        print("=" * 50)

        return render_template('results.html',
                               charts=charts,
                               stats=stats,
                               total=total,
                               model_name=model_name,
                               results_table=df.head(20).to_html(classes='table
table-striped', index=False))

    except Exception as e:
        import traceback
        print("=" * 50)
        print("ПОМИЛКА ПРИ ОБРОБЦІ ФАЙЛУ:")
        print(traceback.format_exc())
        print("=" * 50)
        flash(f'Помилка: {str(e)}', 'error')
        return redirect(url_for('index'))


@app.route('/download')
def download_results():
    results_path = os.path.join(app.config['UPLOAD_FOLDER'], 'results.csv')
    if os.path.exists(results_path):
        return send_file(results_path, as_attachment=True,
download_name='analysis_results.csv')
    else:
        flash('Файл з результатами не знайдено', 'error')
        return redirect(url_for('index'))


if __name__ == '__main__':
    print("\n" + "=" * 50)
```

```
    print("ЗАПУСК ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ")
    print("=" * 50)
    load_models()
    print("\nВідкрийте у браузері: http://127.0.0.1:5000")
    print("=" * 50)
    app.run(debug=True, host='127.0.0.1', port=5000)
```

train_model.py:

```
import pandas as pd
import os
from sklearn.model_selection import train_test_split
from modules.text_processor import preprocess_texts, get_vectorizer
from modules.classifier import train_models, evaluate_models, save_model,
EMOTION_LABELS
from modules.visualizer import create_confusion_matrix

DATA_PATH = 'data/feedback.csv'
MODEL_LR_PATH = 'models/logistic_regression.pkl'
MODEL_RF_PATH = 'models/random_forest.pkl'
MODEL_NN_PATH = 'models/neural_network.keras'
VECTORIZER_PATH = 'models/vectorizer.pkl'


def create_sample_data():
    sample_data = {
        'text': [
            "I love this product it's amazing!",
            "I hate everything about this terrible experience",
            "Not what I expected disappointed",
            "Wow this is surprising!",
            "It's okay nothing special",
            "Absolutely wonderful highly recommend",
            "Worst purchase ever waste of money",
            "Could be better but not terrible",
            "Oh my god this is incredible!",
            "So angry right now never again",
            "Pretty good satisfied with quality",
            "No words to express how bad this is",
            "I'm shocked this exceeded all expectations",
            "Mediocre product average service",
            "This made my day fantastic!",
            "What a disaster complete failure",
            "Not bad but also not great",
            "I'm speechless this is brilliant",
            "Terrible quality don't buy",
            "Very happy with my purchase",
            "Best thing ever I'm so happy",
            "This is garbage totally useless",
            "I expected more from this product",
            "Incredible I can't believe it",
            "Nothing special just average",
            "Perfect exceeds all expectations",
            "Don't waste your money on this",
            "Almost good but not quite there",
            "Wow just wow amazing work",
            "So frustrated with this purchase",
            "Exactly what I needed thank you",
            "Not worth the price at all",
            "This surprised me in a good way",
            "It's fine does the job",
            "Absolutely perfect no complaints",
```

```
            "Never buying this again",
            "Could have been much better",
            "Mind-blowing absolutely fantastic",
            "Waste of time and money",
            "Satisfied with the purchase overall",
        ],
        'emotion': [0, 1, 2, 3, 4, 0, 1, 2, 3, 1, 0, 1, 3, 4, 0, 1, 2, 0, 1, 0,
                    0, 1, 2, 3, 4, 0, 1, 2, 3, 1, 0, 1, 3, 4, 0, 1, 2, 0, 1, 0]
    }

    df = pd.DataFrame(sample_data)
    os.makedirs('data', exist_ok=True)
    df.to_csv(DATA_PATH, index=False)
    print(f"Створено прикладові дані: {DATA_PATH} (всього {len(df)} прикладів)")
    return df


def main():
    print("=" * 60)
    print("НАВЧАННЯ МОДЕЛЕЙ КЛАСИФІКАЦІЇ ЕМОЦІЙ")
    print("=" * 60)

    try:
        df = pd.read_csv(DATA_PATH)
        print(f"Завантажено дані з {DATA_PATH}")
    except FileNotFoundError:
        print("Файл з даними не знайдено. Створюємо прикладові дані...")
        df = create_sample_data()

    if 'text' not in df.columns or 'emotion' not in df.columns:
        print("Помилка: дані повинні мати колонки 'text' та 'emotion'")
        return

    X = df['text'].tolist()
    y = df['emotion'].tolist()

    print(f"Всього прикладів: {len(X)}")
    print(f"Класи: {EMOTION_LABELS}")

    print("\n[1/5] Попередня обробка текстів...")
    X_vectors, processed_texts = preprocess_texts(X)
    print(f"Розмірність векторизованих даних: {X_vectors.shape}")

    print("\n[2/5] Розділення даних (80% навчання, 20% тестування)...")
    X_train, X_test, y_train, y_test = train_test_split(
        X_vectors, y, test_size=0.2, random_state=42
    )
    print(f"Тренувальна вибірка: {X_train.shape[0]} прикладів")
    print(f"Тестова вибірка: {X_test.shape[0]} прикладів")

    print("\n[3/5] Навчання моделей...")
    lr_model, rf_model, nn_model = train_models(X_train, y_train)

    print("\n[4/5] Оцінка якості моделей...")
    results, lr_pred, rf_pred, nn_pred = evaluate_models(lr_model, rf_model,
nn_model, X_test, y_test)

    print("\n" + "=" * 60)
    print("РЕЗУЛЬТАТИ НАВЧАННЯ")
    print("=" * 60)
    for model_name, metrics in results.items():
```

```
        print(f"\n{model_name}:")
        print(f"  - Точність (accuracy): {metrics['accuracy']:.4f}")
        print(f"  - Точність (precision): {metrics['precision']:.4f}")
        print(f"  - Повнота (recall): {metrics['recall']:.4f}")
        print(f"  - F1-міра: {metrics['f1_score']:.4f}")

    best_model_name = max(results, key=lambda x: results[x]['accuracy'])
    best_model = None

    if best_model_name == "Logistic Regression":
        best_model = lr_model
        save_model(lr_model, MODEL_LR_PATH)
    elif best_model_name == "Random Forest":
        best_model = rf_model
        save_model(rf_model, MODEL_RF_PATH)
    else:
        best_model = nn_model
        save_model(nn_model, MODEL_NN_PATH)

    print(f"\nОбрана краща модель: {best_model_name} (точність:
{results[best_model_name]['accuracy']:.4f})")

    save_model(lr_model, MODEL_LR_PATH)
    save_model(rf_model, MODEL_RF_PATH)
    if nn_model is not None:
        save_model(nn_model, MODEL_NN_PATH)
    save_model(get_vectorizer(), VECTORIZER_PATH)

    print(f"\nМоделі збережено:")
    print(f"   - Логістична регресія: {MODEL_LR_PATH}")
    print(f"   - Випадковий ліс: {MODEL_RF_PATH}")
    if nn_model is not None:
        print(f"   - Нейронна мережа: {MODEL_NN_PATH}")
    print(f"   - Векторизатор: {VECTORIZER_PATH}")

    print("\n[5/5] Побудова матриці помилок...")
    os.makedirs('static', exist_ok=True)

    if best_model_name == "Neural Network" and nn_pred is not None:
        create_confusion_matrix(y_test, nn_pred,
save_path='static/confusion_matrix.png')
    elif best_model_name == "Random Forest":
        create_confusion_matrix(y_test, rf_pred,
save_path='static/confusion_matrix.png')
    else:
        create_confusion_matrix(y_test, lr_pred,
save_path='static/confusion_matrix.png')

    print("   Матрицю помилок збережено в static/confusion_matrix.png")

    print("\n" + "=" * 60)
    print("НАВЧАННЯ ЗАВЕРШЕНО УСПІШНО!")
    print("=" * 60)
    print("\nТепер ви можете запустити застосунок командою: python app.py")


if __name__ == '__main__':
    main()
```

__init__.py:

```
from .text_processor import preprocess_texts, clean_text, tokenize_text,
remove_stopwords, lemmatize_tokens, get_vectorizer
from .classifier import train_models, evaluate_models, save_model, load_model,
predict_emotion
from .visualizer import create_charts, create_confusion_matrix
```

classifier.py:

```
import joblib
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, classification_report

try:
    import tensorflow as tf
    from tensorflow import keras
    from tensorflow.keras import layers, Sequential

    TENSORFLOW_AVAILABLE = True
except ImportError:
    TENSORFLOW_AVAILABLE = False
    print("TensorFlow не встановлено. Нейронна мережа недоступна.")

EMOTION_LABELS = {
    0: 'Радість',
    1: 'Злість',
    2: 'Розчарування',
    3: 'Подив',
    4: 'Нейтральний'
}



def train_models(X_train, y_train):

    lr_model = LogisticRegression(max_iter=1000, random_state=42, C=1.0)
    lr_model.fit(X_train, y_train)
    print("Логістична регресія навчена")

    rf_model = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-
1)
    rf_model.fit(X_train, y_train)
    print("Випадковий ліс навчений")
```

```
    nn_model = None
    if TENSORFLOW_AVAILABLE:
        nn_model = train_neural_network(X_train, y_train)
    else:
        print("Нейронна мережа навчена")

    return lr_model, rf_model, nn_model


def train_neural_network(X_train, y_train):
    if hasattr(X_train, 'toarray'):
        X_train = X_train.toarray()

    num_classes = len(np.unique(y_train))

    y_train_categorical = tf.keras.utils.to_categorical(y_train,
num_classes=num_classes)

    model = Sequential([
        layers.Dense(256, activation='relu', input_shape=(X_train.shape[1],)),
        layers.Dropout(0.3),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.2),
        layers.Dense(64, activation='relu'),
        layers.Dropout(0.2),
        layers.Dense(num_classes, activation='softmax')
    ])

    model.compile(
        optimizer='adam',
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )

    print("Навчання нейронної мережі...")
    model.fit(
        X_train, y_train_categorical,
        epochs=50,
        batch_size=32,
        validation_split=0.1,
```

```
        verbose=0,
        callbacks=[
            tf.keras.callbacks.EarlyStopping(patience=5,
restore_best_weights=True)
        ]
    )

    print("Нейронна мережа навчена")
    return model


def evaluate_models(lr_model, rf_model, nn_model, X_test, y_test):
    lr_pred = lr_model.predict(X_test)
    lr_accuracy = accuracy_score(y_test, lr_pred)
    lr_precision = precision_score(y_test, lr_pred, average='weighted',
zero_division=0)
    lr_recall = recall_score(y_test, lr_pred, average='weighted', zero_division=0)
    lr_f1 = f1_score(y_test, lr_pred, average='weighted', zero_division=0)

    rf_pred = rf_model.predict(X_test)
    rf_accuracy = accuracy_score(y_test, rf_pred)
    rf_precision = precision_score(y_test, rf_pred, average='weighted',
zero_division=0)
    rf_recall = recall_score(y_test, rf_pred, average='weighted', zero_division=0)
    rf_f1 = f1_score(y_test, rf_pred, average='weighted', zero_division=0)

    results = {
        'Logistic Regression': {
            'accuracy': lr_accuracy,
            'precision': lr_precision,
            'recall': lr_recall,
            'f1_score': lr_f1
        },
        'Random Forest': {
            'accuracy': rf_accuracy,
            'precision': rf_precision,
            'recall': rf_recall,
            'f1_score': rf_f1
        }
    }
```

```
    nn_pred = None
    if nn_model is not None:
        if hasattr(X_test, 'toarray'):
            X_test_dense = X_test.toarray()
        else:
            X_test_dense = X_test

        nn_pred_prob = nn_model.predict(X_test_dense, verbose=0)
        nn_pred = np.argmax(nn_pred_prob, axis=1)

        nn_accuracy = accuracy_score(y_test, nn_pred)
        nn_precision = precision_score(y_test, nn_pred, average='weighted',
zero_division=0)
        nn_recall = recall_score(y_test, nn_pred, average='weighted',
zero_division=0)
        nn_f1 = f1_score(y_test, nn_pred, average='weighted', zero_division=0)

        results['Neural Network'] = {
            'accuracy': nn_accuracy,
            'precision': nn_precision,
            'recall': nn_recall,
            'f1_score': nn_f1
        }

    return results, lr_pred, rf_pred, nn_pred


def save_model(model, filename):
    if TENSORFLOW_AVAILABLE and isinstance(model, tf.keras.Model):
        model.save(filename)
        print(f"Keras модель збережено в {filename}")
    else:
        joblib.dump(model, filename)
        print(f"Модель збережено в {filename}")


def load_model(filename):
    if filename.endswith('.keras') or (TENSORFLOW_AVAILABLE and
tf.keras.models.load_model):
        try:
            model = tf.keras.models.load_model(filename)
```

```
            print(f"Keras модель завантажено з {filename}")
            return model
        except:
            pass

    return joblib.load(filename)


def predict_emotion(model, vectorizer_obj, processed_text, is_keras=False):
    vector = vectorizer_obj.transform([processed_text])

    if is_keras or (TENSORFLOW_AVAILABLE and isinstance(model, tf.keras.Model)):
        if hasattr(vector, 'toarray'):
            vector = vector.toarray()
        prediction_prob = model.predict(vector, verbose=0)
        prediction = np.argmax(prediction_prob, axis=1)[0]
    else:
        prediction = model.predict(vector)[0]

    emotion_name = EMOTION_LABELS.get(prediction, 'Невідомо')
    return prediction, emotion_name


def get_classification_report(y_true, y_pred):
    return classification_report(y_true, y_pred,
target_names=list(EMOTION_LABELS.values()))
```

text_processor.py:

```
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer

try:
    nltk.data.find('tokenizers/punkt')
except LookupError:
    nltk.download('punkt')
    nltk.download('stopwords')
    nltk.download('wordnet')
    nltk.download('omw-1.4')
```

```
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1, 2))


def clean_text(text):
    if not isinstance(text, str):
        text = str(text)
    text = re.sub(r'[^a-zA-Z\']', ' ', text)
    text = text.lower()
    text = re.sub(r'\s+', ' ', text).strip()
    return text


def tokenize_text(text):
    return word_tokenize(text)


def remove_stopwords(tokens):
    return [token for token in tokens if token not in stop_words and len(token) >
1]


def lemmatize_tokens(tokens):
    return [lemmatizer.lemmatize(token) for token in tokens]


def preprocess_single_text(text):
    cleaned = clean_text(text)
    tokens = tokenize_text(cleaned)
    tokens_no_stop = remove_stopwords(tokens)
    lemmatized = lemmatize_tokens(tokens_no_stop)
    return ' '.join(lemmatized)

def preprocess_texts(texts):
    processed_texts = [preprocess_single_text(text) for text in texts]
    vectors = vectorizer.fit_transform(processed_texts)
    return vectors, processed_texts
```

```
def get_vectorizer():
    return vectorizer


def transform_new_texts(texts, vectorizer_obj):
    processed_texts = [preprocess_single_text(text) for text in texts]
    vectors = vectorizer_obj.transform(processed_texts)
    return vectors, processed_texts
```

visualizer.py:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np

plt.style.use('seaborn-v0_8-darkgrid')
sns.set_palette("Set2")


def create_emotion_distribution_chart(df,
save_path='static/emotion_distribution.png'):
    plt.figure(figsize=(10, 6))

    emotion_counts = df['emotion'].value_counts()

    colors = {
        'Радість': '#2ecc71',
        'Злість': '#e74c3c',
        'Розчарування': '#f39c12',
        'Подив': '#3498db',
        'Нейтральний': '#95a5a6'
    }

    bar_colors = [colors.get(e, '#95a5a6') for e in emotion_counts.index]

    bars = plt.bar(emotion_counts.index, emotion_counts.values, color=bar_colors)

    for bar in bars:
        height = bar.get_height()
        plt.text(bar.get_x() + bar.get_width() / 2., height + 0.5,
                 f'{int(height)}', ha='center', va='bottom', fontsize=12,
fontweight='bold')
```

```
    plt.title('Розподіл емоцій у відгуках користувачів', fontsize=16,
fontweight='bold')
    plt.xlabel('Емоційний стан', fontsize=14)
    plt.ylabel('Кількість відгуків', fontsize=14)
    plt.xticks(rotation=0, fontsize=12)
    plt.yticks(fontsize=12)
    plt.grid(axis='y', alpha=0.3)

    total = len(df)
    plt.text(0.02, 0.98, f'Всього відгуків: {total}',
transform=plt.gca().transAxes,
             fontsize=12, verticalalignment='top', bbox=dict(boxstyle='round',
facecolor='white', alpha=0.8))

    plt.tight_layout()
    plt.savefig(save_path, dpi=150, bbox_inches='tight')
    plt.close()
    return save_path


def create_sentiment_dynamics_chart(df,
save_path='static/sentiment_dynamics.png'):

    if 'date' not in df.columns:
        plt.figure(figsize=(10, 6))
        plt.text(0.5, 0.5, 'Для аналізу динаміки настроїв\nпотрібна колонка з
датами (date)',
                 ha='center', va='center', fontsize=14,
transform=plt.gca().transAxes)
        plt.title('Динаміка настроїв у часі', fontsize=16, fontweight='bold')
        plt.axis('off')
        plt.tight_layout()
        plt.savefig(save_path, dpi=150, bbox_inches='tight')
        plt.close()
        return save_path

    if 'emotion' not in df.columns:
        plt.figure(figsize=(10, 6))
        plt.text(0.5, 0.5, 'Немає даних про емоції для аналізу',
```

```
                ha='center', va='center', fontsize=14,
transform=plt.gca().transAxes)
        plt.title('Динаміка настроїв у часі', fontsize=16, fontweight='bold')
        plt.axis('off')
        plt.tight_layout()
        plt.savefig(save_path, dpi=150, bbox_inches='tight')
        plt.close()
        return save_path

    df_copy = df.copy()

    try:
        df_copy['date'] = pd.to_datetime(df_copy['date'], format='%d.%m.%Y',
errors='coerce')
    except:
        df_copy['date'] = pd.to_datetime(df_copy['date'], errors='coerce')

    df_copy = df_copy.dropna(subset=['date'])

    if len(df_copy) == 0:
        plt.figure(figsize=(10, 6))
        plt.text(0.5, 0.5, 'Некоректний формат дат\nВикористовуйте формат
ДД.ММ.РРРР',
                ha='center', va='center', fontsize=14,
transform=plt.gca().transAxes)
        plt.title('Динаміка настроїв у часі', fontsize=16, fontweight='bold')
        plt.axis('off')
        plt.tight_layout()
        plt.savefig(save_path, dpi=150, bbox_inches='tight')
        plt.close()
        return save_path

    df_copy['month'] = df_copy['date'].dt.to_period('M')
    monthly_counts = df_copy.groupby(['month',
'emotion']).size().unstack(fill_value=0)

    if len(monthly_counts) == 0:
        plt.figure(figsize=(10, 6))
        plt.text(0.5, 0.5, 'Недостатньо даних для побудови графіка',
                ha='center', va='center', fontsize=14,
transform=plt.gca().transAxes)
```

```
        plt.title('Динаміка настроїв у часі', fontsize=16, fontweight='bold')
        plt.axis('off')
        plt.tight_layout()
        plt.savefig(save_path, dpi=150, bbox_inches='tight')
        plt.close()
        return save_path

    monthly_counts.index = monthly_counts.index.astype(str)

    colors = {'Радість': '#2ecc71', 'Злість': '#e74c3c',
              'Розчарування': '#f39c12', 'Подив': '#3498db', 'Нейтральний':
'#95a5a6'}

    plt.figure(figsize=(12, 6))

    for emotion in monthly_counts.columns:
        color = colors.get(emotion, '#95a5a6')
        plt.plot(monthly_counts.index, monthly_counts[emotion],
                 marker='o', linewidth=2, markersize=8, label=emotion,
color=color)

    plt.title('Динаміка настроїв у часі (за місяцями)', fontsize=16,
fontweight='bold')
    plt.xlabel('Місяць', fontsize=14)
    plt.ylabel('Кількість відгуків', fontsize=14)
    plt.legend(loc='best', fontsize=12)
    plt.xticks(rotation=45, fontsize=10)
    plt.grid(True, alpha=0.3)

    plt.tight_layout()
    plt.savefig(save_path, dpi=150, bbox_inches='tight')
    plt.close()

    return save_path


def create_pie_chart(df, save_path='static/emotion_pie.png'):
    emotion_counts = df['emotion'].value_counts()

    colors = ['#2ecc71', '#e74c3c', '#f39c12', '#3498db', '#95a5a6']
```

```
    plt.figure(figsize=(10, 8))
    wedges, texts, autotexts = plt.pie(emotion_counts.values,
labels=emotion_counts.index,
                                       autopct='%1.1f%%',
colors=colors[:len(emotion_counts)],
                                       startangle=90, textprops={'fontsize': 12})

    for autotext in autotexts:
        autotext.set_color('white')
        autotext.set_fontweight('bold')

    plt.title('Розподіл емоцій (відсотки)', fontsize=16, fontweight='bold')
    plt.tight_layout()
    plt.savefig(save_path, dpi=150, bbox_inches='tight')
    plt.close()
    return save_path


def create_confusion_matrix(y_true, y_pred,
save_path='static/confusion_matrix.png'):
    from sklearn.metrics import confusion_matrix

    cm = confusion_matrix(y_true, y_pred)
    emotion_names = ['Радість', 'Злість', 'Розчарування', 'Подив', 'Нейтральний']

    plt.figure(figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=emotion_names[:len(cm)],
yticklabels=emotion_names[:len(cm)])
    plt.title('Матриця помилок класифікації емоцій', fontsize=16,
fontweight='bold')
    plt.xlabel('Передбачена емоція', fontsize=14)
    plt.ylabel('Реальна емоція', fontsize=14)
    plt.xticks(rotation=45)

    plt.tight_layout()
    plt.savefig(save_path, dpi=150, bbox_inches='tight')
    plt.close()
    return save_path
```

```
def create_charts(df, y_true=None, y_pred=None):
    charts = {
        'emotion_distribution': create_emotion_distribution_chart(df),
        'sentiment_dynamics': create_sentiment_dynamics_chart(df),
        'pie_chart': create_pie_chart(df)
    }

    if y_true is not None and y_pred is not None:
        charts['confusion_matrix'] = create_confusion_matrix(y_true, y_pred)

    return charts
```