НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет/(ННІ)

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри**
Інформаційних технологій (назва факультету (ННІ))	Комп'ютерних наук (назва кафедри)
_____ (підпис) Ігор Болбот (ім'я ПРІЗВИЩЕ)	_____ (підпис) Белла Голуб (ім'я ПРІЗВИЩЕ)
"___" _____ 2025 р.	"___" _____ 2025 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

"Аналітична система управління мережі ресторанів"

Спеціальність
122 "Комп'ютерні науки"
(код і назва)

Освітня програма
Інформаційні управляючі системи і технології
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

к.т.н., доцент (науковий ступінь та вчене звання) _____ (підпис) Белла Голуб (ім'я ПРІЗВИЩЕ)

Керівник магістерської кваліфікаційної роботи

к.ф.-м.н., доцент (науковий ступінь та вчене звання) _____ (підпис) Віктор Кириченко (ім'я ПРІЗВИЩЕ)

Виконала _____ (підпис) Тетяна Жученко (ім'я ПРІЗВИЩЕ здобувача)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

доцент, к.т.н. _____ Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)

" 10 " листопада 2024 року

ЗАВДАННЯ

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Жученко Тетяні Вікторівні
(прізвище, ім'я, по батькові)

Спеціальність 122 "Комп'ютерні науки"
(код і найменування)

Освітня програма Інформаційні управляючі системи і технології
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи Аналітична система управління мережі ресторанів

затверджена наказом проректора НУБіП України від " 1 " листопада 2024 р. № 1964 "С"

Термін подання завершеної роботи на кафедру _____.
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи пояснювальна записка, презентація захисту, Jupyter Notebook, що містить повний цикл обробки даних, аналіз, вибір та навчання ML-моделі, розгорнутий веб-сайт, репозиторій на GitHub

Перелік питань, що підлягають дослідженню:

1. Як організувати ефективну інтеграцію ML-моделей з веб-додатками?
2. Як ефективно очистити та підготувати текстові дані відгуків для аналізу?
3. Як розв'язати проблему виявлення аспектів (їжа, сервіс, атмосфера) в неструктурованому тексті?
4. Наскільки точно система визначає тональність порівняно з експертною оцінкою?

Перелік графічного матеріалу (за потреби) _____

Дата видачі завдання " 10 " листопада 2024 р.

Керівник магістерської кваліфікаційної роботи _____ Кириченко В.В.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Жученко Т.В.
(підпис) (прізвище та ініціали)

Календарний план

№ з/п	Назва етапів виконання магістерської кваліфікаційної роботи	Строк виконання етапів магістерської кваліфікаційної роботи	Примітка
1	Видача завдання	01.11.2024	
2	Дослідження предметної області	02.11-30.11.2024	
3	Моделювання системи	1.12.2024-02.01.2025	
4	Розробка системи	03.01-15.05.2025	
5	Тестування системи	16.05-20.06.2025	
6	Аналіз отриманих даних	21.06-30.07.2025	
6	Розробка записки	31.07-13.11.2025	
7	Розробка постеру	07.10-15.10.2025	
8	Написання тез до постеру	16.10-25.10.2025	
9	Постерна сесія	28.10-29.10.2025	
10	Перевірка на плагіат	14.11.2025	
11	Попередній захист	28.11.2025	
12	Захист	15-19.12.2025	

Студент _____ Тетяна Жученко
(підпис) (ім'я ПРІЗВИЩЕ)

Керівник магістерської кваліфікаційної роботи _____ Віктор Кириченко
(підпис) (ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Робота описує процес створення аналізу та тестування аналітичної системи управління мережі ресторанів.

Об'єкт дослідження: процеси збору, обробки та аналізу відгуків у ресторанній індустрії з використанням методів машинного навчання та веб-технологій.

Предмет дослідження: алгоритми машинного навчання для класифікації тональності та виявлення аспектів в українськомовних відгуках, методи їх інтеграції з веб-платформою для формування аналітичних висновків.

Методи дослідження: статистичні методи для аналізу структури та якості даних (кореляційний аналіз, аналіз розподілів, виявлення викидів), методи обробки природної мови для препроцесингу текстів, алгоритми машинного навчання для класифікації (ансамблеві методи Random Forest та Gradient Boosting, лінійні моделі Logistic Regression, ймовірнісні методи Naive Bayes, методи на основі відстаней KNN), методи валідації моделей (5-кратна крос-валідація, train-test split, метрики Accuracy, Precision, Recall, F1-Score, ROC-AUC), технології веб-розробки для практичної реалізації (FastAPI, Next.js, PostgreSQL).

Мета дослідження: розробити інтелектуальну систему аналізу відгуків ресторанів, яка забезпечує автоматизований аналіз тональності, аспектний аналіз за категоріями (їжа, сервіс, атмосфера, ціна) та інтеграцію з веб-платформою для прийняття управлінських рішень у режимі реального часу.

Наукова складова полягає у дослідженні та розробці комплексної методології аспектно-орієнтованого аналізу українськомовних ресторанних відгуків, що поєднує словникові методи з машинним навчанням для одночасної класифікації загальної тональності та оцінки чотирьох ключових аспектів досвіду клієнта.

Рекомендації щодо впровадження результатів: впроваджувати систему аналізу відгуків як інструмент моніторингу, використовувати аспектну статистику для обґрунтування інвестиційних рішень.

Практична значущість роботи: визначається створенням повнофункціональної аналітичної системи управління рестораном, що трансформує неструктуровані клієнтські відгуки у структуровані метрики для прийняття управлінських рішень.

Кількість сторінок - 89.

Кількість ілюстрацій - 28.

Кількість формул - 30.

Кількість додатків - 6.

Кількість використаних джерел - 28.

ABSTRACT

The work describes the process of creating, analyzing, and testing an analytical management system for a restaurant network.

Object of the study: the processes of collecting, processing, and analyzing reviews in the restaurant industry using machine learning methods and web technologies.

Subject of the study: machine learning algorithms for sentiment classification and aspect detection in Ukrainian-language reviews, as well as methods for their integration with a web platform to generate analytical insights.

Research methods: statistical methods for analyzing the structure and quality of data (correlation analysis, distribution analysis, outlier detection); natural language processing methods for text preprocessing; machine learning algorithms for classification (ensemble methods Random Forest and Gradient Boosting, linear models Logistic Regression, probabilistic methods Naive Bayes, distance-based methods KNN); model validation methods (5-fold cross-validation, train-test split, metrics such as Accuracy, Precision, Recall, F1-Score, ROC-AUC); web development technologies for practical implementation (FastAPI, Next.js, PostgreSQL).

Purpose of the study: to develop an intelligent review analysis system for restaurants that provides automated sentiment analysis, aspect-based analysis by categories (food, service, atmosphere, price), and integration with a web platform to support managerial decision-making in real time.

Scientific contribution: consists in the research and development of a comprehensive methodology for aspect-oriented analysis of Ukrainian-language restaurant reviews, combining lexicon-based approaches with machine learning to simultaneously classify overall sentiment and evaluate four key aspects of customer experience.

Recommendations for implementation: deploy the review analysis system as a monitoring tool and use aspect-based statistics to justify investment decisions.

Practical significance: defined by the development of a fully functional analytical restaurant management system that transforms unstructured customer reviews into structured metrics for managerial decision-making.

Number of pages – 89.

Number of illustrations – 28.

Number of formulas – 30.

Number of appendices – 6.

Number of references – 28.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - Application Programming Interface

BERT - Bidirectional Encoder Representations from Transformers

GPT - Generative Pre-trained Transformer

ML - Machine Learning

RESTful -Representational State Transfer

VPS - Virtual Private Server

ВСТУП

Актуальність дослідження зумовлена стрімкою цифровою трансформацією ресторанної індустрії та зростаючою роллю онлайн-відгуків у формуванні репутації закладів харчування. Ключовою проблемою є специфіка українськомовного контенту.

У високонкурентному середовищі ресторанного бізнесу швидкість реакції на негативний фідбек визначає довгострокову репутацію закладу. Традиційний ручний аналіз відгуків є трудомістким процесом на який потрібно витрачати багато часу, а час - це гроші, тому автоматизована система аналізу дозволяє обробляти тисячі відгуків за секунди, забезпечуючи діагностику стану закладу, майже у реальному часі.

Особливої актуальності набуває аспектно-орієнтований аналіз, що дозволяє виявляти конкретні проблемні зони (якість їжі, обслуговування персоналу, атмосфера, ціна) замість загальної бінарної оцінки тональності. Це забезпечує перехід від реактивного управління на основі інтуїції до проактивного менеджменту з кількісно обґрунтованими пріоритетами інвестицій у розвиток закладу.

Мета дослідження - розробити аналітичну систему управління рестораном на основі автоматизованого аналізу українськомовних відгуків клієнтів з використанням методів машинного навчання та аспектно-орієнтованого підходу для підтримки прийняття управлінських рішень.

Для досягнення мети визначено наступні завдання:

Проаналізувати існуючі підходи до аналізу тональності текстових відгуків та обґрунтувати вибір методів машинного навчання для класифікації українськомовних текстів.

Розробити методологію збору та підготовки датасету українськомовних ресторанних відгуків, включаючи автоматизований web scraping, очищення даних, обробку мовних заперечень та балансування класів.

Створити систему аспектно-орієнтованого аналізу з словниками ключових термінів для чотирьох критичних аспектів (їжа, обслуговування, атмосфера, співвідношення ціна-якість) та алгоритмом витягування структурованих аспектних ознак.

Провести порівняльне експериментальне дослідження семи алгоритмів машинного навчання (Naive Bayes, Random Forest, Logistic Regression, SVM, Gradient Boosting, Decision Tree, KNN) на датасеті з 1373 відгуків для вибору оптимальної моделі за критеріями точності, стабільності та обчислювальної ефективності.

Реалізувати RESTful API веб-сервіс для інтеграції натренованої моделі з веб-застосунком та розробити інтерактивний дашборд для візуалізації результатів аналізу.

Виконати експериментальну валідацію прототипу інтеграції на реальних даних з оцінкою точності системи порівняно з експертною розміткою.

Об'єктом дослідження є процес автоматизованого аналізу неструктурованих текстових відгуків клієнтів для підтримки управлінських рішень у ресторанному бізнесі.

Предметом дослідження виступають методи машинного навчання для класифікації тональності та аспектно-орієнтованого аналізу українськомовних ресторанних відгуків, технології векторизації текстів (TF-IDF), алгоритми обробки мовних заперечень та архітектурні рішення інтеграції ML-моделей з веб-додатками.

Методи дослідження. У роботі використано комплекс методів: статистичні методи для аналізу структури та якості даних (кореляційний аналіз, аналіз розподілів, виявлення викидів), методи обробки природної мови для препроцесингу текстів, алгоритми машинного навчання для класифікації (ансамблеві методи Random Forest та Gradient Boosting, лінійні моделі Logistic Regression, ймовірнісні методи Naive Bayes, методи на основі відстаней KNN), методи валідації моделей (5-кратна крос-валідація, train-test split, метрики

Accuracy, Precision, Recall, F1-Score, ROC-AUC), технології веб-розробки для практичної реалізації (FastAPI, Next.js, PostgreSQL).

Наукова новизна отриманих результатів полягає у розробці комплексної методології аспектно-орієнтованого аналізу українськомовних ресторанних відгуків, що поєднує словникові методи з машинним навчанням для одночасної класифікації загальної тональності та оцінки чотирьох ключових аспектів досвіду клієнта. Експериментальному виявленні оптимального балансу між точністю та інтерпретованістю для production-систем аналізу відгуків - Random Forest з ROC-AUC 95.74% забезпечує на 6.85% кращу якість ймовірнісних оцінок порівняно з Naive Bayes при незначних втратах у F1-мірі (0.35%).

Практична значущість роботи визначається створенням повнофункціональної аналітичної системи управління рестораном, що трансформує неструктурований клієнтський фідбек у структуровані метрики для прийняття управлінських рішень.

Універсальність розробленого алгоритму дозволяє його адаптацію для аналізу відгуків про продукцію фермерських господарств, що розширює потенційний ринок впровадження системи на агропромисловий комплекс України та забезпечує міждисциплінарну цінність дослідження.

Апробація результатів. Основні положення та результати дослідження представлено на конференції «Інформаційні технології: економіка, техніка, освіта».[24]

Структура та обсяг роботи. Магістерська кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Повний обсяг роботи становить 89 сторінок основного тексту, містить 28 рисунків, 30 формул, 6 додатків. Список використаних джерел налічує 28 найменувань.

У першому розділі виконано огляд існуючих підходів до аналізу тональності текстів, сформульовано конкретні завдання дослідження.

У другому розділі обґрунтовано вибір машинного навчання, описано методологію автоматизованого збору відгуків, виконано огляд досліджуваних

алгоритмів з математичними формулюваннями, визначено систему метрик оцінювання для багатовимірної оцінки якості моделей.

У третьому розділі надано повне математичне обґрунтування векторизації та семи алгоритмів машинного навчання, представлено моделювання та архітектура системи.

У четвертому розділі описано створення та статистичний аналіз датасету, проведено детальний текстовий аналіз з кореляційним дослідженням, проведено порівняльне дослідження алгоритмів з обґрунтуванням вибору Random Forest, описано практичну реалізацію API, представлено результати роботи системи на реальних відгуках, продемонстровано візуалізацію результатів на інтерактивному дашборді.

РОЗДІЛ 1. ОГЛЯД ЛІТЕРАТУРИ

Джерело 1 представляє Censored Naive Bayes (CNB) - адаптацію популярного алгоритму Наївний Байєс для прогнозування ризику з використанням даних часу до настання події, що підлягають цензуруванню, де CNB є непараметричним відносно базового розподілу часу настання подій та гнучко моделює граничні розподіли коваріат.

Дослідження 2 представляє емпіричний аналіз класифікатора Наївний Байєс, який значно спрощує навчання за рахунок припущення про незалежність ознак за умови належності до класу, і незважаючи на те, що незалежність зазвичай є нереалістичним припущенням, показує конкурентоспроможність з сучасними класифікаторами, такими як C4.5.

Стаття 3 є оригінальною роботою Лео Бреймана (Leo Breiman), яка вперше представила алгоритм Random Forest - ансамблевий метод машинного навчання, що поєднує безліч дерев рішень, де кожне дерево залежить від значень випадкового вектора, що відбирається незалежно з однаковим розподілом для всіх дерев у лісі.

Робота 4 представляє комплексний огляд застосування методу опорних векторів (SVM) для класифікації, детально аналізуючи його сильні та слабкі сторони, включаючи проблеми вибору параметрів, алгоритмічну складність, що впливає на час навчання класифікатора на великих наборах даних, та розробку оптимальних класифікаторів для багатокласових задач.

Дослідження 5 присвячене удосконаленню алгоритму наївного байєсівського класифікатора шляхом комбінування зваженості ознак та калібрування Лапласа для подолання основного недоліку методу - припущення про незалежність атрибутів, що часто не відповідає реальності та знижує точність класифікації.

Робота 6 присвячена розробці оптимізованої версії класичного алгоритму C4.5 для побудови дерев рішень з значно покращеними характеристиками

швидкодії та обчислювальної ефективності. Дослідження пропонує удосконалені алгоритмічні рішення та структури даних, які дозволяють суттєво прискорити процес навчання дерев рішень на великих датасетах, зберігаючи при цьому високу точність класифікації.

Стаття 7 є фундаментальною роботою Коріни Кортес та Володимира Вапника, яка вперше представила Support Vector Networks (SVM) - новий метод машинного навчання для задач двогрупової класифікації, що концептуально реалізує ідею нелінійного відображення вхідних векторів у простір ознак дуже високої розмірності, де будується лінійна поверхня рішення.

Джерело 8 представляє комплексний огляд методів глибокого навчання для аспектно-орієнтованого аналізу тональності, систематизуючи сучасні підходи до виявлення та класифікації конкретних аспектів у текстових відгуках. Дослідження аналізує архітектури нейронних мереж (CNN, LSTM, BERT, Transformer), що дозволяють автоматично виявляти згадування аспектів та визначати їх тональність з високою точністю порівняно з традиційними методами машинного навчання.

Стаття 9 систематично аналізує існуючі підходи до покращення k-NN алгоритму, включаючи методи оптимізації відстаней, техніки зменшення розмірності, стратегії вибору оптимального значення k та інші модифікації, надаючи порівняльний аналіз їх продуктивності на різних типах даних.

Джерело 10 надає практичний посібник з використання градієнтного бустингу через чотири популярні бібліотеки Python, що порівнює їх реалізації, продуктивність та особливості налаштування гіперпараметрів. Дослідження демонструє застосування методу для задач регресії та класифікації з акцентом на оптимізацію швидкості навчання та якості прогнозів на табличних даних.

Робота 11 представляє архітектуру SentiLSTM на базі рекурентних нейронних мереж з довгою короткостроковою пам'яттю для аналізу тональності ресторанних відгуків. Запропонований підхід досягає точності 89.3% на датасеті Yelp, демонструючи переваги глибокого навчання для захоплення

контекстуальних залежностей у текстах порівняно з традиційними методами машинного навчання.

Публікація 12 являє собою вступний огляд алгоритму k-найближчих сусідів як одного з фундаментальних методів машинного навчання, що пояснює основні принципи роботи, теоретичні засади та практичні аспекти застосування цього методу.

Дослідження 13 порівнює ефективність класичної логістичної регресії з шістьма оптимізованими алгоритмами машинного навчання (градієнтний бустинг, нейронні мережі, k-найближчих сусідів, випадковий ліс, метод опорних векторів та MARS) у клінічному контексті, демонструючи конкурентоспроможність простих лінійних методів з більш складними алгоритмами при коректному налаштуванні.

Стаття 14 представляє комплексний огляд методу логістичної регресії як одного з найбільш широко використовуваних алгоритмів в області аналізу даних та бінарної класифікації. Робота зосереджується на алгоритмічних та машинно-навчальних аспектах логістичної регресії, розглядаючи її застосування для аналізу незбалансованих та рідкісних подій у даних.

Робота 15 представляє класичний алгоритм C4.5 для побудови дерев рішень, який є удосконаленням попереднього алгоритму ID3 та включає можливості роботи з неперервними атрибутами, обробки пропущених значень та методи відсікання гілок для запобігання перенавчанню. Ця фундаментальна праця описує теоретичні основи та практичну реалізацію одного з найвпливовіших алгоритмів машинного навчання для класифікації.

Дослідження 16 присвячене застосуванню методів машинного навчання та глибокого навчання для аналізу тональності відгуків про ресторани з використанням технік обробки природної мови (NLP). Автори порівнюють ефективність різних алгоритмів, де найкращі результати показали логістична регресія та мультиноміальний наївний Байєс (89.9% та 89.6% точності відповідно), а також архітектури глибокого навчання CNN та Bi-LSTM (89% та 90% точності).

Стаття 17 представляє комплексне дослідження застосування алгоритмів машинного навчання для аналізу та класифікації тональності ресторанних відгуків. Порівнюються різні підходи до препроцесингу текстових даних (токенізація, видалення стоп-слів, стемінг) та векторизації (TF-IDF, Bag-of-Words), а також ефективність множини класифікаторів для бінарної та багатокласової класифікації тональності.

Робота 18 представляє підсумок питань, обговорених під час воркшопу "Support Vector Machines (SVM) Theory and Applications" в рамках Advanced Course on Artificial Intelligence, з метою представлення огляду фонової теорії та сучасного розуміння SVM, включаючи теоретичні основи, практичні застосування та відкриті дослідницькі питання у сфері методів опорних векторів.

Стаття 19 представляє практичне керівництво з використання алгоритму Random Forest у статистичному навчанні, зосереджуючись на реалізації методу в середовищі R та демонструючи його застосування на двох практичних прикладах: класифікації дефолту власників кредитних карток та прогнозуванні поширень онлайн-новин.

Джерело 20 представляє детальний огляд градієнтного бустингу як одного з найефективніших алгоритмів машинного навчання для табличних даних, який поєднує високу точність, універсальність застосування (регресія, класифікація, ранжування), інтерпретованість через аналіз важливості ознак та широку практичну реалізацію у бібліотеках XGBoost та LightGBM.

Дослідження 21 представляє порівняльний аналіз продуктивності класичного алгоритму k-найближчих сусідів та його різних варіантів для задач прогнозування захворювань у медичній галузі. Стаття систематично досліджує ефективність різних модифікацій k-NN алгоритму в контексті медичної діагностики, надаючи емпіричну оцінку їх точності, чутливості та специфічності.

Робота 22 представляє комплексний теоретичний аналіз алгоритму Random Forest, систематично досліджуючи кожну складову алгоритму з метою розкриття нових аспектів його навчання, включаючи оригінальний аналіз обчислювальної складності, що демонструє їх хорошу продуктивність та

масштабованість, а також глибоке обговорення деталей реалізації в рамках бібліотеки Scikit-Learn.

Джерело 23 надає комплексний огляд сучасного стану досліджень у галузі аспектно-орієнтованого аналізу тональності (ABSA), що систематизує основні задачі (виявлення аспектів, екстракція термінів аспектів, класифікація тональності аспектів), методи їх вирішення (класичні ML, глибоке навчання, transfer learning) та відкриті виклики, включаючи обробку неявних аспектів, мультимодальний аналіз та низькоресурсні мови.

Дослідження 24 представляє практичне порівняння двох популярних фреймворків для розробки веб-додатків - Next.js та Angular - на основі розробки реальних проектів. Аналізуються переваги та недоліки кожного фреймворку з точки зору продуктивності, зручності розробки, екосистеми інструментів, можливостей серверного рендерингу та SEO-оптимізації для прийняття обґрунтованих рішень щодо вибору технологічного стеку.

Патент 25 представляє технологію адаптивного формування та відображення користувацьких інтерфейсів, яка здатна автоматично генерувати інтерфейс відповідно до індивідуального контексту користувача, характеристик пристрою, функціональної ролі та специфічних умов застосування. Запропонована система забезпечує можливість динамічного вибору та організації UI-елементів в режимі реального часу.

Патент 26 описує систему для розробки веб-додатків із використанням компонентного фреймворку, де кожен компонент є самодостатнім і включає в себе логіку, шаблон і стиль. Такий підхід забезпечує модульність, повторне використання коду та спрощує масштабування складних веб-застосунків через інкапсуляцію функціональності у незалежні компоненти.

Патент 27 описує спосіб оптимізації та зберігання React-компонентів через систему автоматичного обходу всіх підкомпонентів, виявлення конфігураційних блоків у JSX, їх парсинг і заміну на нові компоненти під час ініціалізації. Це дозволяє динамічно генерувати компоненти з налаштувань, покращуючи продуктивність та спрощуючи конфігурацію React-застосунків.

Патент 28 описує розробку загальнопризначеного веб-додаткового фреймворку, що надає набір тегів розмітки та відповідних програмних об'єктів для спрощення створення веб-застосунків, особливо орієнтованих на онлайн-управління цифровими медіа (фотографії, аудіо, відео файли) з акцентом на зручність використання та швидкість розробки.

РОЗДІЛ 2. НАПРЯМИ ТА МЕТОДИ ДОСЛІДЖЕНЬ

2.1. Обґрунтування вибору напряму досліджень та методології

Сучасна ресторанна індустрія характеризується високою залежністю від онлайн-репутації - споживачі читають відгуки перед вибором ресторану. Середній заклад отримує 50-200 відгуків щомісяця, що створює значний обсяг неструктурованої інформації.

Серед підходів до автоматичного аналізу текстів виділяють три напрями: лексичні методи, методи машинного навчання та глибоке навчання. Лексичні методи базуються на словниках тональності, але не враховують контекст та для української мови мають обмежену якість словників. Методи глибокого навчання (BERT, LSTM) демонструють найвищу точність, але вимагають десятків тисяч прикладів, GPU-ресурсів та є "чорними скриньками" без інтерпретації.

Методи машинного навчання обрано як оптимальний напрям через: ефективну роботу на датасетах середнього розміру (сотні-тисячі прикладів), відсутність потреби у спеціалізованому обладнанні, прийнятну швидкість для застосувань у реальному часі, можливість інтерпретації важливості ознак та пояснення рішень, легке оновлення при надходженні нових даних. Для датасету з 1600 відгуків алгоритми ML забезпечують оптимальний баланс точності, швидкості та інтерпретованості.

Обрано методологію порівняльного експериментального аналізу семи алгоритмів машинного навчання, що представляють різні парадигми класифікації:

Naive Bayes - ймовірнісний метод на основі теореми Байєса, ефективний для розріджених текстових даних з найшвидшою швидкістю навчання.

Logistic Regression - лінійна модель з природною придатністю для TF-IDF векторів та ймовірнісною інтерпретацією результатів.

Support Vector Machine - метод максимізації відступу з RBF-ядром для побудови нелінійних границь рішення.

Decision Tree - деревоподібна структура з високою інтерпретованістю через візуалізацію правил "якщо-то".

Random Forest - ансамблевий метод на основі bagging з високою точністю та можливістю обчислення feature importance для виділення ключових слів.

Gradient Boosting - ансамбль на основі boosting з послідовним виправленням помилок для досягнення максимальної точності.

K-Nearest Neighbors - метод класифікації за подібністю без побудови явної моделі.

Вибір цих алгоритмів забезпечує покриття різних парадигм ML та дозволяє всебічно дослідити можливості машинного навчання для аналізу українськомовних відгуків.

2.2. Методологія формування та підготовки датасету

Обрано підхід збору реальних відгуків з Google Maps замість синтетичних даних, оскільки це забезпечує автентичні мовні конструкції та найбільше покриття закладів в Україні. Платформа Google надає api за допомогою якого можна буде отримати відгуки про ресторани, якщо дані не підійдуть то буде розроблено власний web scraper на Selenium для автоматизації збору. Географічне покриття: Київ, Львів, Одеса з різноманітністю кухонь та цінових сегментів. Після збору відгуки буде нормалізовано для тренування алгоритмів, кожен відгук матиме бінарну класифікацію 0 чи 1, де 0 - це негативний відгук, а 1 – це позитивний відгук.

Після збору буде проведено статистичний аналіз датасету. Фундаментальне значення описової статистики полягає у можливості трансформації великих масивів неструктурованих текстових даних у компактний набір інформативних показників, що забезпечують первинне розуміння

характеристик досліджуваного корпусу та формують основу для подальшого статистичного моделювання.

Середнє арифметичне (mean) представляє найбільш поширену міру центральної тенденції, що обчислюється як частка від ділення суми всіх спостережень на їх кількість. У текстовому аналізі середнє арифметичне характеризує типове значення досліджуваного параметра для всього корпусу. Однак цей показник чутливий до екстремальних значень, що може призводити до спотворення уявлення про типові характеристики при наявності значних викидів у даних.

Медіана (50-й перцентиль) являє собою значення, що ділить упорядкований ряд спостережень навпіл, при цьому половина значень знаходиться нижче медіани, а половина - вище. Медіана є робастною мірою центральної тенденції, менш чутливою до екстремальних значень порівняно з середнім арифметичним, що робить її особливо цінною для характеристики текстових даних, які часто містять атипові спостереження.

Система квартилів поділяє упорядкований ряд даних на чотири рівні частини, забезпечуючи детальну характеристику розподілу. Перший квартиль (Q1, 25-й перцентиль) вказує на значення, нижче якого знаходиться чверть всіх спостережень, характеризуючи нижню границю типових значень. Третій квартиль (Q3, 75-й перцентиль) визначає верхню границю, вище якої розташована лише чверть спостережень.

Міжквартильний розмах (IQR = Q3 - Q1) представляє діапазон, що містить центральні 50% спостережень, і є робастною мірою варіабельності даних. Цей показник особливо важливий для ідентифікації викидів та оцінки гомогенності текстового корпусу.

Стандартне відхилення (standard deviation) є основною мірою розсіювання даних навколо середнього арифметичного. Теоретично, стандартне відхилення характеризує типову величину відхилення індивідуальних спостережень від середнього значення. У контексті нормального розподілу приблизно 68%

спостережень знаходяться в межах одного стандартного відхилення від середнього, а 95% - в межах двох стандартних відхилень.

Високе стандартне відхилення свідчить про значну варіабельність характеристик текстів, тоді як низьке - про їх малу варіабельність.

Мінімальне та максимальне значення визначають повний діапазон варіації досліджуваного параметра в корпусі. Ці показники критично важливі для виявлення потенційних викидів та оцінки адекватності вибірки. Екстремальні значення можуть індикувати наявність специфічних підкорпусів або атипових текстів, що потребують окремого аналізу.

Показник count (кількість) вказує на обсяг валідних спостережень для кожного параметра, що є критично важливим для оцінки повноти даних та статистичної потужності аналізу. У текстових дослідженнях відсутність значень може індикувати проблеми збору даних або специфічні характеристики певних текстів.

Сукупність описових статистик формує статистичний профіль текстового корпусу, що дозволяє ідентифікувати основні закономірності та аномалії. Порівняння різних статистичних показників між собою забезпечує розуміння форми розподілу: наближення медіани до середнього арифметичного індикує симетричність розподілу, тоді як їх значні відмінності свідчать про асиметрію.

Комплексний аналіз описових статистик є необхідною передумовою для обґрунтованого вибору методів подальшої обробки текстових даних, включаючи нормалізацію, трансформацію та застосування алгоритмів машинного навчання. Розуміння статистичних властивостей корпусу дозволяє прогнозувати ефективність різних аналітичних підходів та інтерпретувати отримані результати в контексті специфічних характеристик досліджуваних текстів.

Коефіцієнт імбалансу — це показник, який показує, наскільки рівномірно розподілені класи в даних, дорівнює відношенню більшого класу до меншого. Для обрахунку ділимо кількість записів більшого класу на кількість записів меншого класу.

Значення 1.0 означає ідеальний баланс, 1.5 вказує на помірний дисбаланс (один клас у півтора рази більший), 3.0 свідчить про значний імбаланс, що вимагає балансування, а 10.0 є критичним дисбалансом з десятикратною перевагою одного класу.

При значному перевищенні одиниці виникають проблеми: модель передбачає домінуючий клас для максимізації загальної точності; погіршується Recall для меншості через пропуск рідкісних прикладів; Accuracy стає оманливою метрикою; втрачається здатність виявляти важливі патерни у меншості, що критично для виявлення проблем у відгуках.

Для векторизації текстових даних обрано метод TF-IDF (Term Frequency-Inverse Document Frequency) як оптимальний підхід для класичних алгоритмів машинного навчання. TF-IDF враховує важливість терміну на двох рівнях: частоту слова в конкретному документі (Term Frequency) та його рідкісність у всьому корпусі текстів (Inverse Document Frequency). Така двокомпонентна оцінка дозволяє виділяти ключові слова, що є специфічними для певних класів тональності, знижуючи вагу загальновживаних термінів та підвищуючи вагу характерних для класу виразів.

Датасет розділено за схемою 80/20 на навчальну та тестову вибірки з використанням стратифікованого підходу (stratified split), що зберігає пропорції класів у кожній підвибірці. Навчальна вибірка (80%) використовується для тренування моделей та налаштування гіперпараметрів через крос-валідацію, тоді як тестова вибірка (20%) залишається недоторканою до фінальної оцінки якості для об'єктивного вимірювання здатності моделі узагальнювати результати на нових, раніше не бачених даних.

2.3. Огляд досліджуваних алгоритмів

У рамках дослідження буде проведено комплексне порівняння семи алгоритмів машинного навчання. Вибір алгоритмів охоплює основні парадигми машинного навчання: лінійні методи, методи на основі дерев рішень, ансамблеві

підходи, імовірнісні методи та методи, базовані на схожості. Загальна схема роботи алгоритму представлена нижче(рис.2.1).

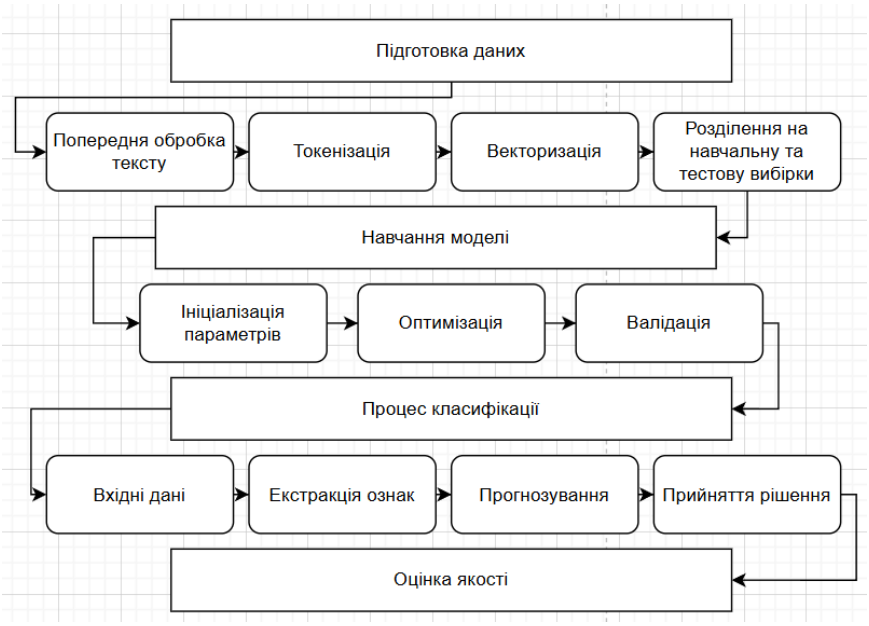

Рис.2.1 Загальна схема роботи алгоритмів машинного навчання для класифікації текстових даних

Характеристика алгоритмів:

1. Logistic Regression (Логістична регресія) - лінійний алгоритм, що базується на принципі максимізації правдоподібності для оцінки ймовірності належності до класу. Особливо ефективний для лінійно розділюваних класів та забезпечує високу інтерпретованість через аналіз коефіцієнтів при ознаках[1].

2. Random Forest (Випадковий ліс) - ансамблевий метод, що агрегує результати множини дерев рішень, кожне з яких навчається на випадковій підвибірці даних та ознак. Забезпечує високу стійкість до перенавчання та шуму в даних, дозволяє оцінити важливість ознак[3].

3. Support Vector Machine (Метод опорних векторів) - алгоритм з радіальною базисною функцією (RBF), що використовує принцип максимізації відстані між класами у багатовимірному просторі ознак. Особливо ефективний для текстових даних високої розмірності завдяки здатності працювати з розрідженими матрицями[4].

4. Naive Bayes (Наївний Байєс) - імовірнісний класифікатор, що базується на теоремі Баєса з припущенням незалежності ознак. Незважаючи на спрощення реальності, часто демонструє хороші результати на текстових даних завдяки природній розрідженості TF-IDF (Term Frequency Inverse Document Frequency) представлення[5].

5. Gradient Boosting (Градієнтний бустинг) - ансамблевий метод, що реалізує принцип послідовного покращення моделі шляхом додавання нових слабких класифікаторів, кожен з яких навчається виправляти помилки попередніх. Забезпечує високу точність класифікації при ретельному налаштуванні[10].

6. Decision Tree (Дерево рішень) - це модель, яка приймає рішення шляхом послідовних запитань про дані. Вона працює як дерево з гілками: на кожному вузлі модель ставить питання (наприклад, «чи вага > 50?»), і залежно від відповіді — йде ліворуч або праворуч. [6].

7. K-Nearest Neighbors (k-NN) - алгоритм з параметром k=5, що базується на принципі схожості та відноситься до класу lazy learning. Класифікація нових об'єктів здійснюється на основі мажоритарного голосування серед п'яти найближчих сусідів у просторі ознак[21].

2.4 Метрики оцінки якості класифікації

Всі моделі навчено на однакових TF-IDF векторах для виключення впливу різних представлень. Фінальна оцінка включає метрики:

1. Accuracy (Точність) - частка правильно класифікованих об'єктів від загальної кількості.

- Діапазон: 0-1 (чим вище, тим краще)
- Формула:

$$Accuracy = (TP + TN) / (TP + TN + FP + FN), \quad (2.1)$$

де TP — кількість істинно позитивних прогнозів (True Positives);

TN — кількість істинно негативних прогнозів (True Negatives);

FP — кількість хибно позитивних прогнозів (False Positives);

FN — кількість хибно негативних прогнозів (False Negatives).

- Інтерпретація: показує загальну ефективність моделі

2. Precision (Точність класу або точність позитивних прогнозів) – з усіх випадків, які модель назвала позитивними, скільки виявилися справді позитивними Діапазон: 0-1 (чим вище, тим краще)

- Формула:

$$Precision=TP / (TP + FP), \quad (2.2)$$

де TP — кількість істинно позитивних прогнозів (True Positives);

FP — кількість хибно позитивних прогнозів (False Positives);

- Інтерпретація: показує, наскільки модель точна у своїх позитивних прогнозах

3. Recall (Повнота/Чутливість) - скільки відсотків усіх справжніх позитивних випадків модель знайшла

- Діапазон: 0-1 (чим вище, тим краще)
- Формула:

$$Recall=TP / (TP + FN), \quad (2.3)$$

де TP — кількість істинно позитивних прогнозів (True Positives);

FN — кількість хибно негативних прогнозів (False Negatives).

- Інтерпретація: показує, наскільки добре модель виявляє всі позитивні випадки

4. F1-Score (F1- оцінка) - гармонійне середнє між Precision та Recall

- Діапазон: 0-1 (чим вище, тим краще)
- Формула:

$$F1\text{-}Score =2 \times (Precision \times Recall) / (Precision + Recall), \quad (2.4)$$

де Precision — точність прогнозу, частка правильно визначених позитивних об'єктів серед усіх класифікованих як позитивні;

Recall — повнота класифікації, частка правильно визначених позитивних об'єктів серед усіх об'єктів позитивного класу.

- Інтерпретація: збалансована оцінка якості, особливо корисна при незбалансованих класах

5. ROC-AUC (Area Under ROC Curve - площа під кривою помилок) – показує здатність моделі розрізняти класи. ROC-крива будується на квадраті 1×1:
вісь X (False Positive Rate), вісь Y (True Positive Rate), мінімум: 0, максимум: 1.

- Діапазон: 0-1 (чим вище, тим краще)
- Інтерпретація:

0.5 = випадкове передбачення

0.7-0.8 = прийнятна якість

0.8-0.9 = хороша якість

0.9+ = відмінна якість

6. CV F1 (Cross-Validation F1 - F1-оцінка з крос-валідацією) - наскільки стабільно працює модель на різних частинах даних

- Формат: середнє значення ± стандартне відхилення
- Інтерпретація: показує стабільність моделі на різних підвибірках даних

7. Час (сек) - час виконання навчання та тестування моделі

- Інтерпретація: показує обчислювальну складність алгоритму.

Така методологія забезпечує об'єктивне порівняння алгоритмів та дозволяє виявити оптимальне рішення з урахуванням балансу між точністю, швидкістю та інтерпретованістю.

У цьому розділі обґрунтовано вибір напряму дослідження та розроблено комплексну методологію побудови інтелектуальної системи для аналізу українськомовних відгуків про ресторани. Доведено, що методи машинного навчання становлять оптимальний підхід для вирішення поставленої задачі, оскільки забезпечують прийнятну точність класифікації на датасетах обмеженого обсягу, дозволяють інтерпретувати прийняті рішення через аналіз важливості ознак та не вимагають спеціалізованого обчислювального обладнання, на відміну від методів глибокого навчання.

Розроблено систематичну методологію формування та підготовки датасету, яка охоплює процеси автоматизованого збору реальних відгуків з

онлайн-платформ, проведення статистичного аналізу характеристик текстових даних, застосування методів балансування для усунення проблеми нерівномірного розподілу класів та перетворення текстів у числове представлення через векторизацію методом Term Frequency-Inverse Document Frequency.

Проведено детальний теоретичний аналіз семи алгоритмів машинного навчання, які представляють різні парадигми класифікації - від простих ймовірнісних методів до складних ансамблевих підходів. Визначено систему метрик оцінювання якості моделей, що включає показники точності, повноти, збалансованості та надійності прогнозів, забезпечуючи можливість всебічного та об'єктивного порівняння алгоритмів за їх ефективністю, стабільністю результатів та придатністю до практичного застосування у реальних бізнес-сценаріях.

РОЗДІЛ 3. ТЕОРЕТИЧНІ ОБҐРУНТУВАННЯ

3.1. Математичне обґрунтування використаних методів

Обробка природної мови для аналізу відгуків ресторанів вимагає перетворення текстової інформації у числове представлення, придатне для алгоритмів машинного навчання. Нехай $D = \{d_1, d_2, ..., d_n\}$ - множина відгуків, де кожен відгук d_i є послідовністю слів. Процес векторизації тексту за методом TF-IDF (Term Frequency-Inverse Document Frequency) визначається наступним чином.[14]

Частота терміну tf(t, d) для слова t у документі d обчислюється як відношення кількості появ слова до загальної кількості слів у документі:

$$tf(t, d) = count(t, d) / |d|, \quad (3.1)$$

де count(t, d) - кількість появ терміну t у документі d;

|d| - загальна кількість термінів у документі d.

Обернена частота документа idf(t, D) відображає важливість терміну у всьому корпусі:

$$idf(t, D) = \log(N / |\{d \in D : t \in d\}|), \quad (3.2)$$

де N - загальна кількість документів;

$|\{d \in D : t \in d\}|$ - кількість документів, що містять термін t.

Фінальна вага TF-IDF визначається як добуток цих компонент:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D) \quad (3.3)$$

Така векторизація дозволяє представити кожен відгук як вектор у багатовимірному просторі ознак, де кожна координата відповідає вазі конкретного слова. Розглянемо алгоритми, які плануємо використовувати:

1. Байєсівська класифікація (Naive Bayes)

Алгоритм Naive Bayes базується на теоремі Байєса та припущенні умовної незалежності ознак. Для класифікації відгуку d у клас c (позитивний, негативний або нейтральний) обчислюється апостеріорна ймовірність:

$$P(c|d) = P(d|c) \times P(c) / P(d), \quad (3.4)$$

де P(c|d) - апостеріорна ймовірність належності документа d до класу c, яку необхідно обчислити;

P(d|c) - правдоподібність (likelihood), ймовірність спостерігати документ d за умови, що він належить до класу c;

P(c) - апріорна ймовірність класу c, обчислена на основі частоти класу у навчальній вибірці;

P(d) - повна ймовірність документа d, що є нормалізуючою константою для всіх класів.

За принципом максимальної апостеріорної ймовірності (MAP) вибирається клас:

$$\hat{c} = \text{argmax } P(c) \prod P(w_i|c), \quad (3.5)$$

де w_i - слова у відгуку;

P(c) - апріорна ймовірність класу;

$P(w_i|c)$ - умовна ймовірність слова за умови належності до класу c.

Ці ймовірності оцінюються на основі навчальної вибірки з згладжуванням Лапласа для уникнення нульових ймовірностей:

$$P(w_i|c) = (\text{count}(w_i, c) + \alpha) / (\sum \text{count}(w, c) + \alpha|V|), \quad (3.6)$$

де $P(w_i|c)$ - умовна ймовірність появи слова w_i у документах класу c;

$\text{count}(w_i, c)$ - кількість поява слова w_i у всіх документах класу c у навчальній вибірці;

α - параметр згладжування Лапласа, зазвичай дорівнює одиниці;

Σ count(w, c) - загальна кількість всіх слів у документах класу c;

V - словник, що містить всі унікальні слова корпусу;

|V| - розмір словника, тобто кількість унікальних слів[5].

2. Випадковий ліс (Random Forest)

Random Forest є ансамблевим методом, що будує множину дерев рішень $\{T_1, T_2, ..., T_m\}$ та агрегує їх прогнози. Кожне дерево T_j будується на випадковій підвибірці даних (bootstrap sample) та підмножині ознак.

Для задачі класифікації фінальний прогноз визначається голосуванням більшості:

$$\hat{c} = \text{mode}\{T_1(x), T_2(x), ..., T_m(x)\}, \quad (3.7)$$

де $\hat{c}$ - фінальний передбачений клас для об'єкта x;

$T_j(x)$ - прогноз j-го дерева рішень для об'єкта x;

m - загальна кількість дерев в ансамблі;

mode{...} - статистична мода, тобто найчастіше значення серед усіх прогнозів дерев.

Побудова окремого дерева базується на мірі інформаційного виграшу (Information Gain). Ентропія множини S визначається як:

$$H(S) = -\sum p_i \log_2(p_i), \quad (3.8)$$

де H(S) - ентропія множини S, що вимірює міру невизначеності або безладу в розподілі класів;

p_i - частка прикладів класу i у множині S;

$\log_2(p_i)$ - логарифм за основою 2 від частки класу i;

знак мінус перед сумою робить значення ентропії невід'ємним.

Інформаційний виграш при розбитті множини S за ознакою A:

$$IG(S, A) = H(S) - \sum (|S_v|/|S|) \times H(S_v), \quad (3.9)$$

де IG(S, A) - інформаційний виграш від розбиття множини S за ознакою A;

H(S) - ентропія початкової множини S до розбиття;

S_v - підмножина S, що містить приклади зі значенням v для ознаки A;

$|S_v|$ - кількість елементів у підмножині S_v;

|S| - загальна кількість елементів у множині S;

$H(S_v)$ - ентропія підмножини S_v після розбиття;

сума обчислюється по всіх можливих значеннях v ознаки A.

Важливість ознаки у Random Forest обчислюється як середнє зменшення ентропії по всіх деревах:

$$\text{Importance}(f) = (1/m) \sum \Delta H(f, T_j), \quad (3.10)$$

де Importance(f) - важливість ознаки f для класифікації;

m - загальна кількість дерев в ансамблі Random Forest;

$\Delta H(f, T_j)$ - зменшення ентропії при використанні ознаки f у дереві T_j;

сума обчислюється по всіх деревах j від 1 до m[3].

3. Метод опорних векторів (SVM)

SVM шукає оптимальну гіперплощину, що максимізує відступ (margin) між класами. Для лінійно роздільних даних задача формулюється як:

$$\min (1/2)||w||^2, \quad (3.11)$$

де w - вектор ваг, що визначає орієнтацію гіперплощини у просторі ознак;

$||w||^2$ - квадрат евклідової норми вектора w, що дорівнює сумі квадратів всіх компонент вектора;

коефіцієнт 1/2 додається для спрощення обчислень при взятті похідної.

За умов формули нижче для всіх «i»:

$$y_i(w \cdot x_i + b) \geq 1, \quad (3.12)$$

де y_i - мітка класу для i-го прикладу, що приймає значення +1 або -1;

w - вектор ваг гіперплощини;

x_i - вектор ознак i-го прикладу;

$w \cdot x_i$ - скалярний добуток векторів w та x_i;

b - зміщення (bias), що визначає положення гіперплощини відносно початку координат;

нерівність має виконуватися для кожного прикладу і у навчальній вибірці.

Для нелінійно роздільних даних вводяться slack-змінні ξ_i та параметр регуляризації C:

$$\min (1/2)||w||^2 + C\sum\xi_i, \quad (3.13)$$

де w - вектор ваг гіперплощини;

$||w||^2$ - квадрат евклідової норми вектора w для максимізації відступу;

ξ_i - slack-змінна для i-го прикладу, що дозволяє помилки класифікації або порушення відступу;

C - параметр регуляризації, що контролює баланс між максимізацією відступу та мінімізацією помилок; сума обчислюється по всіх навчальних прикладах i.

За умов:

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \xi_i \geq 0, \quad (3.14)$$

де y_i - мітка класу для і-го прикладу (+1 або -1);

w - вектор ваг гіперплощини;

x_i - вектор ознак і-го прикладу;

b - зміщення гіперплощини;

ξ_i - slack-змінна для і-го прикладу, що вимірює величину порушення обмеження; обидві нерівності мають виконуватися одночасно для кожного прикладу.

Використання ядрових функцій (kernel trick) дозволяє здійснювати класифікацію у просторі більшої розмірності. Для RBF-ядра:

$$K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2), \quad (3.15)$$

де $K(x_i, x_j)$ - значення ядрової функції, що вимірює подібність між векторами x_i та x_j;

x_i, x_j - вектори ознак двох об'єктів у вихідному просторі;

$||x_i - x_j||^2$ - квадрат евклідової відстані між векторами;

γ - параметр, що контролює радіус впливу окремих прикладів;

exp(...) - експоненційна функція[4].

4. Логістична регресія

Логістична регресія моделює ймовірність належності до класу через логістичну функцію:

$$P(y=1|x) = 1 / (1 + e^{(-(w \cdot x + b)))}, \quad (3.16)$$

де P(y=1|x) - ймовірність того, що об'єкт x належить до позитивного класу (y=1);

w - вектор ваг моделі, що визначає важливість кожної ознаки;

x - вектор ознак об'єкта (TF-IDF вектор відгуку);

$w \cdot x$ - скалярний добуток векторів w та x;

b - зміщення (bias), що визначає порогове значення;

e - основа натурального логарифма (число Ейлера, приблизно 2.718);

знак мінус у показнику степеня забезпечує зростання ймовірності при збільшенні $w \cdot x + b$.

Для багатокласової класифікації використовується softmax-функція:

$$P(y=k|x) = e^{(w_k \cdot x)} / \sum e^{(w_j \cdot x)}, \quad (3.17)$$

де $P(y=k|x)$ - ймовірність того, що об'єкт x належить до класу k;

w_k - вектор ваг для класу k; x - вектор ознак об'єкта;

$w_k \cdot x$ - скалярний добуток вектора ваг класу k та вектора ознак;

e - основа натурального логарифма;

знаменник - сума експонент для всіх класів j, що забезпечує нормалізацію.

Параметри w оцінюються шляхом максимізації логарифмічної правдоподібності з L2-регуляризацією:

$$L(w) = \sum \log P(y_i|x_i, w) - \lambda||w||^2, \quad (3.18)$$

де L(w) - функція логарифмічної правдоподібності, що підлягає максимізації;

y_i - справжня мітка класу для i-го прикладу;

x_i - вектор ознак i-го прикладу;

$P(y_i|x_i, w)$ - ймовірність правильного класу для i-го прикладу при параметрах w;

log - натуральний логарифм, що перетворює добутки ймовірностей у суми;

λ - параметр регуляризації, що контролює силу штрафу за великі ваги;

$||w||^2$ - квадрат евклідової норми вектора ваг, що дорівнює сумі квадратів всіх компонент; сума обчислюється по всіх навчальних прикладах i[1].

5. Градієнтний бустинг (Gradient Boosting)

Gradient Boosting будує ансамбль слабких моделей $\{f_1, f_2, ..., f_m\}$ послідовно, де кожна наступна модель виправляє помилки попередніх. Фінальний прогноз:

$$F(x) = \sum \alpha_t f_t(x), \quad (3.19)$$

де F(x) - фінальний прогноз ансамблю для об'єкта x;

$f_t(x)$ - прогноз t-ї слабкої моделі (зазвичай невелике дерево рішень);

α_t - вага t-ї моделі, що визначає її внесок у фінальний прогноз;

t - індекс ітерації, що змінюється від 1 до m (загальна кількість моделей);

сума обчислюється по всіх моделях в ансамблі.

На кожній ітерації t модель f_t навчається передбачати градієнт функції втрат:

$$f_t = \text{argmin} \sum L(y_i, F_{t-1}(x_i) + f(x_i)), \quad (3.20)$$

де f_t - нова модель, що додається до ансамблю на ітерації t;

argmin - операція пошуку функції f, що мінімізує вираз;

$L(y_i, F_{t-1}(x_i) + f(x_i))$ - функція втрат для і-го прикладу;

y_i - справжнє значення для і-го прикладу;

$F_{t-1}(x_i)$ - прогноз ансамблю з перших t-1 моделей для об'єкта x_i;

$f(x_i)$ - прогноз нової моделі f для об'єкта x_i;

сума обчислюється по всіх навчальних прикладах і.

Для класифікації зазвичай використовується логістична функція втрат:

$$L(y, F(x)) = \log(1 + e^{(-yF(x))}), \quad (3.21)$$

де L(y, F(x)) - значення втрат для прогнозу F(x) при справжній мітці y;

y - справжня мітка класу, що приймає значення +1 або -1;

F(x) - прогноз ансамблю для об'єкта x (дійсне число);

e - основа натурального логарифма;

log - натуральний логарифм;

добуток yF(x) є позитивним для правильних прогнозів та від'ємним для помилкових[10].

6. Метод k-найближчих сусідів (K-Nearest Neighbors)

KNN класифікує об'єкт на основі міток k найближчих сусідів у навчальній вибірці. Відстань між векторами зазвичай обчислюється за метрикою Евкліда:

$$d(x, x') = \sqrt{(\sum(x_i - x'_i)^2)}, \quad (3.22)$$

де d(x, x') - евклідова відстань між векторами x та x';

x_i - значення і-ї координати вектора x;

x'_i - значення і-ї координати вектора x';

$(x_i - x'_i)^2$ - квадрат різниці координат;

сума обчислюється по всіх координатах векторів (по всіх ознаках);

квадратний корінь забезпечує відповідність геометричній відстані у просторі.

Або косинусною відстанню для текстових даних:

$$sim(x, x') = (x \cdot x') / (||x|| \times ||x'||), \quad (3.23)$$

де sim(x, x') - косинусна подібність між векторами x та x', що приймає значення від -1 до 1;

x·x' - скалярний добуток векторів x та x';

||x|| - евклідова норма (довжина) вектора x;

||x'|| - евклідова норма вектора x';

добуток норм у знаменнику нормалізує міру подібності.

Клас визначається голосуванням більшості серед k сусідів:

$$\hat{c} = \text{argmax} \sum I(y_i = c), \quad (3.24)$$

де $\hat{c}$ - передбачений клас для нового об'єкта;

argmax - операція пошуку класу c, для якого сума є максимальною;

$I(y_i = c)$ - індикаторна функція, що дорівнює 1, якщо мітка i-го сусіда y_i збігається з класом c, інакше 0;

y_i - мітка класу i-го найближчого сусіда;

c - один з можливих класів (позитивний, негативний, нейтральний);

сума обчислюється по k найближчих сусідів.

7. Дерева рішень (Decision Tree)

Окреме дерево рішень рекурсивно розбиває простір ознак, вибираючи на кожному кроці оптимальне розбиття за критерієм Джині:

$$Gini(S) = 1 - \sum p_i^2, \quad (3.25)$$

де Gini(S) - індекс Джині для множини S, що вимірює міру невизначеності розподілу класів;

p_i - частка прикладів класу i у множині S;

p_i^2 - квадрат частки класу i;

сума обчислюється по всіх класах у множині S.

$$Split(S, A, t) = (|Sleft|/|S|)Gini(Sleft) + (|Sright|/|S|)Gini(Sright), \quad (3.26)$$

де Split(S, A, t) - оцінка якості розбиття множини S за ознакою A з порогом t;

Sleft - ліва підмножина після розбиття, що містить приклади із значенням ознаки менше або рівне t;

Sright - права підмножина після розбиття, що містить приклади із значенням ознаки більше t;

|Sleft|, |Sright| - кількість елементів у лівій та правій підмножинах відповідно;

|S| - загальна кількість елементів у початковій множині S;

Gini(Sleft), Gini(Sright) - індекси Джині для лівої та правої підмножин.

Обрізання дерева (pruning) запобігає перенавчанню шляхом видалення вузлів, які не покращують якість на валідаційній вибірці[21].

Представлені математичні обґрунтування дозволяють зрозуміти теоретичні основи кожного з семи досліджених алгоритмів та забезпечують строгу формалізацію процесу класифікації тональності відгуків. Кожен з розглянутих методів базується на власному математичному апараті та має специфічні переваги для задачі аналізу текстів.

Байєсівський підхід використовує теорію ймовірностей та припущення про умовну незалежність ознак, що робить його ефективним для роботи з розрідженими текстовими даними при відносно невеликих обчислювальних витратах. Метод опорних векторів забезпечує максимізацію відступу між класами та можливість роботи у просторах високої розмірності завдяки ядровим функціям, що особливо корисно для лінійно нероздільних даних. Логістична регресія надає ймовірнісну інтерпретацію результатів класифікації та дозволяє контролювати перенавчання через механізми регуляризації.

Ансамблеві методи - Random Forest та Gradient Boosting - використовують принцип "мудрості натовпу", об'єднуючи прогнози множини базових моделей для підвищення стабільності та точності. Random Forest будує незалежні дерева на різних підвибірках даних, що забезпечує стійкість до шуму та викидів, тоді як Gradient Boosting послідовно виправляє помилки попередніх моделей, досягаючи

високої точності за рахунок більшої складності навчання. Дерева рішень та метод k-найближчих сусідів представляють інтуїтивно зрозумілі підходи, де перші будують явні правила класифікації, а другі базуються на принципі подібності об'єктів у просторі ознак.

Векторизація текстів методом TF-IDF забезпечує перетворення неструктурованих текстових даних у числове представлення з урахуванням важливості термінів як на рівні окремого документа, так і всього корпусу. Це дозволяє виділити ключові слова та фрази, що несуть найбільше інформативне навантаження для визначення тональності відгуків. Формалізовані метрики оцінювання - точність, precision, recall та F1-міра - надають об'єктивні критерії для порівняння ефективності різних алгоритмів та вибору оптимального рішення для production-середовища.

Математичне обґрунтування є фундаментом для подальшого експериментального дослідження, де теоретичні властивості алгоритмів будуть верифіковані на реальних даних українськомовних відгуків про ресторани. Розуміння математичних принципів роботи кожного методу дозволяє обґрунтовано підходити до вибору гіперпараметрів, інтерпретації результатів та пояснення отриманих висновків щодо оптимального алгоритму для системи аналізу відгуків.

3.2. Моделювання системи

Діаграма прецедентів є важливим елементом при проектуванні системи, оскільки вона дозволяє візуально представити взаємодію користувачів (акторів) з основними функціональними частинами системи. У випадку з аналітичної системи управління мережі ресторанів, діаграма прецедентів (рис.3.1.) допомагає ідентифікувати ключові процеси, з якими будуть взаємодіяти різні типи користувачів.

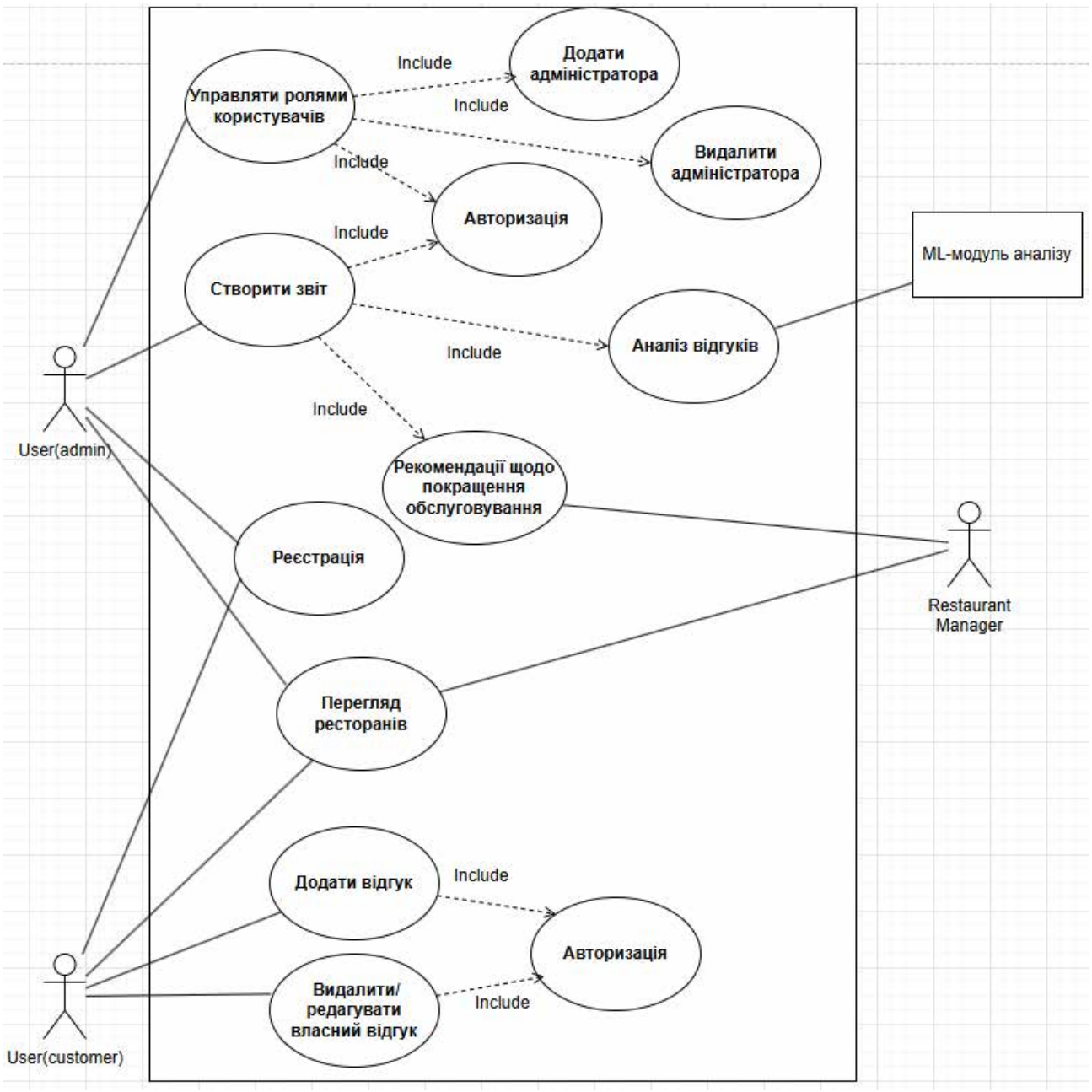

Рис. 3.1. Діаграма прецедентів

Нижче наведено опис основних прецедентів, представлених у діаграмі:

Актори:

1. Користувач (User):
   - Адміністратор (admin) – має доступ до всіх функцій системи для управління відгуками та користувачами.
   - Клієнт (customer) – звичайний користувач, який може залишати відгуки про ресторани, переглядати відгуки інших користувачів.

2. Менеджер ресторану (Restaurant Manager):
    - Користувач, відповідальний за управління конкретним рестораном. Менеджери отримують доступ до інтерфейсу, що дозволяє їм переглядати всі відгуки, отримані на їхні заклади, а також відповідати на відгуки або використовувати аналітику для покращення якості обслуговування.
3. Аналітичний сервіс (ML-модуль аналізу):
    - це власна розроблена аналітична система, створена на основі ґрунтовного порівняльного дослідження семи алгоритмів машинного навчання (Logistic Regression, Random Forest, SVM, Naive Bayes, Gradient Boosting, Decision Tree, K-Nearest Neighbors). За результатами експериментального аналізу було обрано модель Random Forest як оптимальне рішення, що забезпечує точність класифікації 88.36% при збалансованих показниках швидкості обробки та стабільності результатів. Модуль виконує автоматичну класифікацію тональності відгуків та виявлення аспектів обслуговування.

Прецеденти (функції):

1. Авторизація:
    - Опис: Всі користувачі, включаючи адміністраторів та клієнтів, повинні авторизуватися в системі, щоб отримати доступ до функцій.
    - Актори: Користувач (admin, customer).
2. Реєстрація користувача:
    - Опис: Процес реєстрації нового користувача в системі. Ця функція доступна для нових клієнтів.
    - Актори: Користувач (customer).
3. Перегляд ресторанів:
    - Опис: Клієнти можуть переглядати список доступних ресторанів у системі.
    - Актори: Користувач (customer, restaurant manager, admin).
4. Додавання відгуку:
    - Опис: Клієнт може залишити відгук про ресторан, оцінити його за різними критеріями, додати текстовий коментар.

- Актори: Користувач (customer).

5. Редагування/видалення відгуку:
   - Опис: Клієнт може редагувати або видаляти свої власні відгуки.
   - Актори: Користувач (customer).
6. Створення звіту / Аналіз відгуків:
   - Опис: Адміністратори та менеджери ресторанів мають доступ до аналітичного модуля. ML-модуль аналізу автоматично обробляє всі відгуки та формує аналітичні інсайти.
   - Актори: Адміністратор (admin), Менеджер ресторану (Restaurant Manager).
7. Додавання адміністратора:
   - Опис: Адміністратор може додавати нових адміністраторів до системи.
   - Актори: Адміністратор (admin).
8. Видалення адміністратора:
   - Опис: Адміністратор має можливість видаляти адміністратора з системи.
   - Актори: Адміністратор (admin).
9. Рекомендації щодо покращення обслуговування:
   - Опис: На основі автоматичного аналізу відгуків ML-модуль генерує деталізований аналіз по аспектам на основі якого адміністратор (admin) формулює рекомендації для менеджерів ресторанів.
   - Актори: Адміністратор (admin), менеджер ресторану (Restaurant Manager), ML-модуль.

Взаємозв'язки між прецедентами:

- Include (Включення) - відношення include використовується, коли один прецедент обов'язково включає виконання іншого прецедента. У нашій системі - авторизація включається в додавання відгуку, видалення/редагування відгуку, додавання адміністратора, видалення адміністратора, створення звіту, аналіз відгуків (ML-модуль) включається в створення звіту.
- Extend (Розширення) - відношення extend використовується для позначення необов'язкових дій, які можуть бути виконані за певних умов.

Рекомендації щодо покращення розширюють прецедент аналіз відгуків, але активуються лише за запитом менеджера.

Діаграма прецедентів для інформаційної системи управління та аналізу відгуків мережі ресторанів відображає всі важливі функції, з якими будуть взаємодіяти користувачі системи, включаючи адміністраторів, менеджерів ресторанів та клієнтів. Вона допомагає чітко визначити роль кожного актора в процесах системи, а також взаємозв'язок між функціями. Цей етап є важливим для проектування ефективної і зручної системи управління відгуками.

Діаграма класів (рис.3.2) системи аналізу відгуків ресторанів відображає об'єктно-орієнтовану структуру програмного забезпечення та визначає основні сутності, їх характеристики, поведінку та взаємозв'язки. Розглянемо детально кожен клас та його роль у системі.

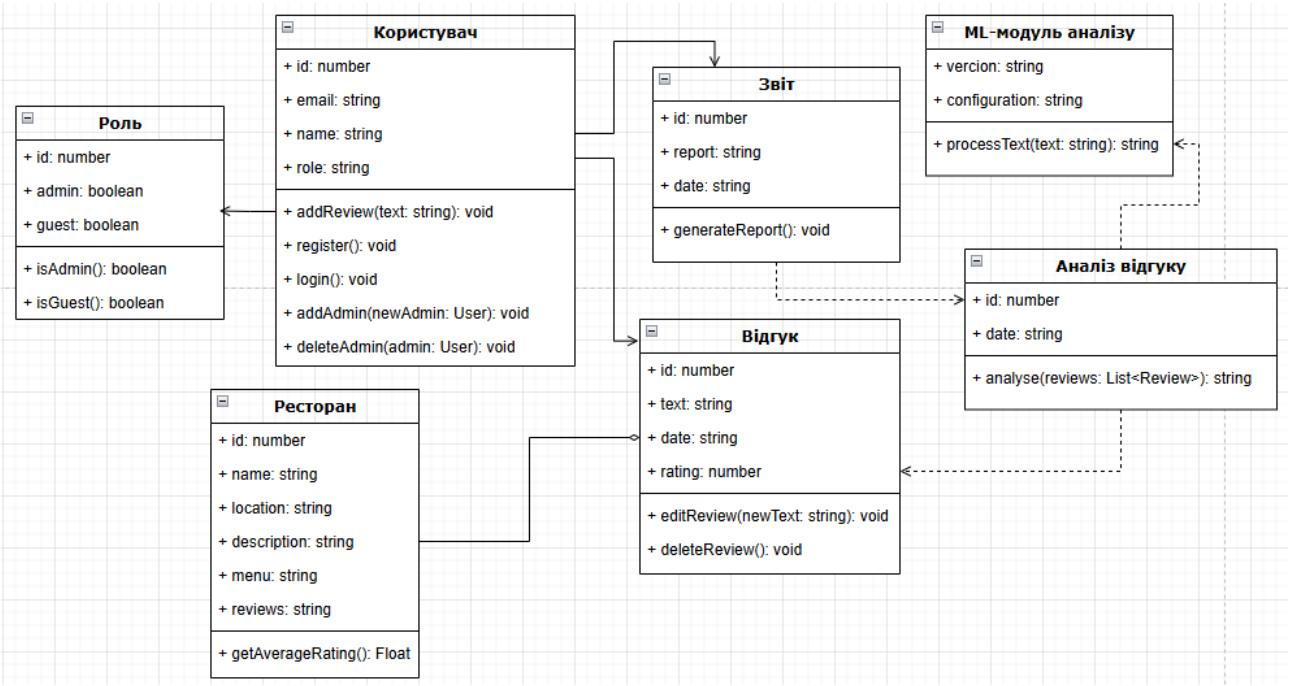

Рис. 3.2. Діаграма класів

Клас Роль (Role) є фундаментальним елементом системи контролю доступу, який визначає права та привілеї різних категорій користувачів. Цей клас містить три основні атрибути: унікальний ідентифікатор ролі типу number, булевий прапорець адміністративних прав admin та прапорець гостьового доступу guest. Для перевірки статусу користувача клас надає два методи:

isAdmin() повертає булеве значення, що вказує на наявність адміністративних прав, а isGuest() перевіряє, чи є користувач гостем системи. Така архітектура забезпечує гнучке управління правами доступу та дозволяє легко розширювати систему ролей у майбутньому.

Клас Користувач (User) представляє зареєстрованих учасників системи та є центральною сутністю для управління обліковими записами. Кожен користувач характеризується унікальним ідентифікатором id, електронною адресою email, яка використовується як логін, іменем name та призначеною роллю role типу string. Функціональність класу реалізована через п'ять ключових методів. Метод addReview приймає текстовий параметр та створює новий відгук від імені користувача. Методи register та login забезпечують процеси реєстрації нового облікового запису та авторизації відповідно. Два спеціальні методи addAdmin та deleteAdmin, доступні лише користувачам з адміністративними правами, дозволяють керувати складом адміністраторів системи, приймаючи як параметр об'єкт типу User. Клас Користувач має множинні асоціативні зв'язки: багато користувачів можуть мати одну роль, один користувач може створити багато відгуків, а також один користувач може згенерувати багато звітів.

Клас Ресторан (Restaurant) інкапсулює інформацію про заклади харчування, представлені в системі. Він містить вичерпний набір атрибутів, що описують ресторан: унікальний ідентифікатор id, назву закладу name, географічну локацію location, текстовий опис description, інформацію про меню menu та колекцію відгуків reviews типу string. Ключовим методом класу є getAverageRating, який повертає значення типу Float і обчислює середній рейтинг ресторану на основі числових оцінок з усіх пов'язаних відгуків. Цей метод ітерує через колекцію відгуків, витягує значення атрибута rating з кожного відгуку, підсумовує їх та ділить на загальну кількість відгуків, повертаючи нульове значення у випадку відсутності відгуків. Клас має асоціативний зв'язок типу один-до-багатьох з класом Відгук, оскільки один ресторан може мати множину відгуків.

Клас Відгук (Review) є критично важливою сутністю системи, яка зберігає думки користувачів про відвідані заклади. Структура класу включає чотири атрибути: унікальний ідентифікатор id, текстовий коментар text типу string, дату створення date у форматі string та числову оцінку rating типу number, що приймає значення від одного до п'яти. Важливим архітектурним рішенням стало включення оцінки безпосередньо як атрибута класу Відгук, що спрощує структуру бази даних та прискорює операції читання, усуваючи необхідність додаткових з'єднань таблиць. Клас надає два методи для управління відгуками: editReview дозволяє користувачу змінити текстовий вміст свого відгуку, приймаючи новий текст як параметр, а deleteReview забезпечує можливість повного видалення відгуку з системи. Клас Відгук встановлює множинні асоціативні зв'язки: багато відгуків належать одному користувачеві, багато відгуків стосуються одного ресторану, і кожен відгук має один унікальний аналіз.

Клас Звіт (Report) призначений для генерації та зберігання аналітичних документів на основі накопичених даних системи. Він включає три атрибути: унікальний ідентифікатор id, вміст звіту report типу string та дату генерації date. Єдиний метод класу generateReport виконує складний процес формування аналітичного звіту, який агрегує інформацію з множини відгуків та їх аналізів, створюючи структуровану аналітичну документацію для менеджерів ресторанів або адміністраторів платформи. Клас має асоціативний зв'язок з класом Користувач типу багато-до-одного, оскільки один користувач може створити багато звітів, а також залежність від класу Аналіз відгуку, використовуючи результати автоматичного аналізу для формування змістовних висновків.

Клас ML-модуль аналізу представляє інтелектуальну підсистему машинного навчання, побудовану на основі алгоритму Random Forest після ретельного порівняльного дослідження семи різних алгоритмів класифікації. Цей клас містить два атрибути: версію моделі version типу string, яка може мати значення на кшталт "RF_v1.0", та конфігураційні параметри configuration, що зберігають налаштування моделі. Центральним методом класу є processText,

який приймає текстовий рядок та повертає результат аналізу також у вигляді string. Цей метод виконує послідовний конвеєр обробки: спочатку здійснює попередню обробку тексту, включаючи токенізацію, нормалізацію та видалення стоп-слів; потім виконує векторизацію за методом TF-IDF, перетворюючи текст у числове представлення; далі застосовує натреновану модель Random Forest для класифікації тональності відгуку; і нарешті виявляє ключові аспекти, згадані у відгуку. Технічною особливістю цього модуля є використання саме алгоритму Random Forest, який продемонстрував точність вісімдесят вісім цілих тридцять шість сотих відсотка та F1-метрику нуль цілих вісімсот вісімдесят сім тисячних, що виявилося оптимальним балансом між точністю, швидкістю обробки та стабільністю результатів.

Клас Аналіз відгуку (FeedbackAnalysis) виконує роль координатора між ML-модулем та бізнес-логікою системи, зберігаючи та обробляючи результати автоматичного аналізу. Структура класу включає унікальний ідентифікатор id та дату проведення аналізу date. Ключовий метод analyse приймає список відгуків типу List<Review> та повертає рядок з результатами аналізу. Цей метод виконує комплексну обробку: для кожного відгуку зі списку він викликає метод processText з ML-модуля аналізу, отримуючи класифікацію тональності як позитивної, негативної або нейтральної; визначає присутні у відгуку аспекти, такі як їжа, сервіс, атмосфера чи ціна; агрегує результати з множини відгуків для виявлення загальних тенденцій; та формує персоналізовані рекомендації для власників ресторанів на основі виявлених слабких місць та сильних сторін. Клас встановлює три типи зв'язків: асоціацію типу один-до-багатьох з класом Відгук, залежність від ML-модуля аналізу через виклик його методів, та асоціацію з класом Звіт для передачі результатів аналізу.

Взаємозв'язки між класами системи реалізовані через два основні типи відносин. Асоціація, представлена суцільними лініями на діаграмі, визначає структурні зв'язки між класами. Зв'язок між Користувачем та Роллю має кардинальність багато-до-одного, що означає можливість призначення однієї ролі багатьом користувачам. Зв'язок Користувач-Відгук типу один-до-багатьох

дозволяє одному користувачеві створювати множину відгуків. Аналогічно, зв'язок Користувач-Звіт також має кардинальність один-до-багатьох. Ресторан пов'язаний з Відгуками відношенням один-до-багатьох, оскільки один заклад може мати багато відгуків від різних користувачів. Унікальним є зв'язок між Відгуком та Аналізом відгуку з кардинальністю один-до-одного, що означає створення окремого аналізу для кожного відгуку.

Залежність, візуалізована пунктирними лініями зі стрілками, відображає використання функціональності одного класу іншим без створення постійного структурного зв'язку. Клас Аналіз відгуку залежить від ML-модуля аналізу, оскільки викликає його метод processText для обробки текстових даних. Клас Звіт має залежність від Аналізу відгуку, використовуючи його результати для формування аналітичної документації. Ці залежності є односпрямованими і вказують на потік виконання в системі.

Архітектура класів демонструє добре структурований підхід до побудови системи з чітким розмежуванням відповідальностей. Класи користувацького рівня забезпечують управління обліковими записами та контент-менеджмент, класи бізнес-логіки відповідають за обробку даних про ресторани та відгуки, а аналітичний шар, представлений ML-модулем та класом Аналіз відгуку, надає інтелектуальну обробку інформації. Така багаторівнева архітектура забезпечує високу масштабованість системи, оскільки кожен компонент може бути незалежно оптимізований чи замінений без впливу на інші частини. Зокрема, ML-модуль реалізований як окремий компонент, що дозволяє легко оновлювати модель машинного навчання, експериментувати з різними алгоритмами або підтримувати декілька версій моделі одночасно для A/B тестування.

Таким чином, представлена діаграма класів забезпечує повне уявлення про структуру системи, визначаючи не лише статичні характеристики кожної сутності, але й динамічну поведінку через методи та взаємодію через асоціації та залежності, що є необхідною основою для успішної реалізації системи аналізу відгуків ресторанів з використанням методів машинного навчання.

3.3. Архітектура системи аналізу відгуків

Архітектура аналітичної системи (рис. 3.3.) управління мережі ресторанів визначає взаємодію між усіма компонентами та забезпечує ефективну обробку та зберігання великих обсягів даних.

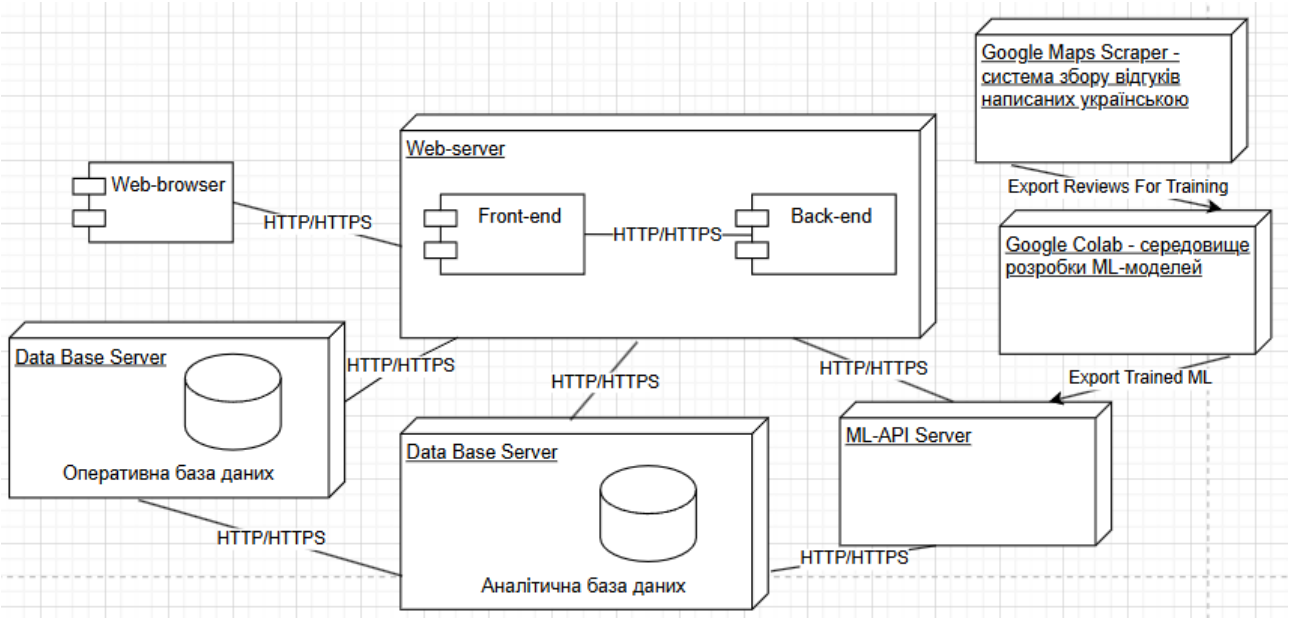

Рис. 3.3. Архітектура системи

Система побудована так, щоб забезпечити зручний доступ до функцій для різних типів користувачів (клієнтів, адміністраторів, менеджерів ресторанів) та забезпечити ефективний аналіз зібраних відгуків. Системи побудована за принципом мікросервісної архітектури.

Основні компоненти системи:

1. Web-browser (Веб-браузер)

Клієнтська частина системи, яка забезпечує інтерфейс користувача для взаємодії з платформою. Веб-браузер надсилає HTTP/HTTPS запити до веб-сервера та отримує відповіді у вигляді HTML-сторінок, JSON-даних або інших форматів.

2. Web-server (Веб-сервер)

Центральний компонент архітектури, який складається з двох підсистем:

Front-end – клієнтська частина, побудована на фреймворку Next.js, який було обрано дослідницьким шляхом[24]. Відповідає за рендеринг інтерфейсу користувача, обробку клієнтських подій та взаємодію з Back-end через внутрішні HTTP/HTTPS запити. Реалізує Server-Side Rendering (SSR) для покращення продуктивності та SEO-оптимізації.

Back-end – серверна частина, також реалізована на Next.js через API Routes. Обробляє бізнес-логіку додатку, валідує дані, керує сесіями користувачів та координує взаємодію між різними сервісами системи.

3. Data Base Server - Оперативна база даних

Головне сховище даних системи, яке зберігає інформацію про користувачів, ресторани, відгуки, та ролі. База даних побудована на реляційній СУБД (PostgreSQL) та забезпечує ACID-гарантії для транзакцій. Веб-сервер взаємодіє з цією базою через HTTP/HTTPS для виконання операцій читання та запису даних.

4. Data Base Server - Аналітична база даних

Спеціалізоване сховище для результатів аналізу відгуків, метрик та статистичних даних. Ця база оптимізована для аналітичних запитів, зберігає результати класифікації тональності, виявлені аспекти, агреговані метрики та історичні тренди.

5. ML-API Server (Сервер ML-моделі)

Окремий сервіс, що надає REST API для доступу до натренованих моделей машинного навчання. Цей сервер отримує HTTP/HTTPS запити з текстами відгуків, виконує їх обробку через завантажену модель Random Forest та повертає результати класифікації.

Додаткові компоненти (етап підготовки даних):

6. Google Maps Scraper (Програма збору відгуків)

Автоматизований інструмент для збору відгуків про ресторани з платформи Google Maps. Ця програма виконує цільовий збір даних. Програма налаштована на пошук та витягування відгуків українською мовою про ресторани та підготовку спеціалізованого промаркованого датасету для

тренування ML-алгоритмів. Зібраний датасет становить основу для навчання та валідації моделей машинного навчання. Використання реальних україномовних відгуків забезпечує адаптацію моделі до специфіки мови, культурних особливостей та стилю написання відгуків українськими користувачами.

7. Google Colab - Середовище розробки ML-моделей

Google Colab служить дослідницькою платформою, де відбувається весь цикл розробки моделі — від сирих даних до готової до деплою ML-моделі та включає:

- Завантаження та первинний аналіз датасету
- Експлораторний аналіз даних (EDA) з візуалізаціями
- Попередня обробка текстів (очищення, токенізація, нормалізація)
- Векторизація текстів
- Навчання семи різних моделей (Logistic Regression, Random Forest, SVM, Naive Bayes, Gradient Boosting, Decision Tree, K-Nearest Neighbors)
- Порівняльний аналіз моделей
- Експорт навченої моделі
- Доступ до GPU: Використання безкоштовних GPU-ресурсів Google Colab значно прискорює процес навчання моделей, особливо при експериментах з глибоким навчанням.

Запропонована архітектура системи володіє низкою суттєвих переваг. Масштабованість досягається завдяки незалежності компонентів - ML-API Server може бути розгорнутий на більш продуктивному обладнанні або реплікований для розподілу навантаження без модифікації інших частин системи. Модульна побудова, що розділяє Front-end, Back-end, бази даних та сервіс машинного навчання, забезпечує можливість оновлення кожного компонента окремо без ризику порушення роботи всієї системи.

Відмовостійкість проявляється у здатності системи продовжувати базове функціонування навіть при виході з ладу окремих компонентів - якщо ML-API Server недоступний, користувачі можуть переглядати ресторани та існуючі

відгуки. Використання Google Colab надає гнучкість для швидкого тестування різних алгоритмів без ризику для production-середовища. Важливою особливістю є локалізація під україномовний ринок - власний інструмент збору відгуків формує датасет українською мовою, що забезпечує високу точність аналізу з урахуванням мовних особливостей та культурних контекстів цільової аудиторії. Така архітектура створює баланс між технічною складністю та практичною ефективністю, задовольняючи поточні потреби та закладаючи основу для майбутнього розширення функціональності.

Оперативна база даних (рис.3.4) системи складається з п'яти взаємопов'язаних таблиць, що забезпечують зберігання інформації про користувачів, ресторани та відгуки. Таблиця Roles містить лише два поля - унікальний ідентифікатор та назву ролі, і служить довідником для визначення типів користувачів у системі. Таблиця Users зберігає дані облікових записів через п'ять полів: ідентифікатор, електронну адресу, ім'я, хешований пароль та зовнішній ключ roleId, що встановлює зв'язок з таблицею Roles для визначення прав доступу.

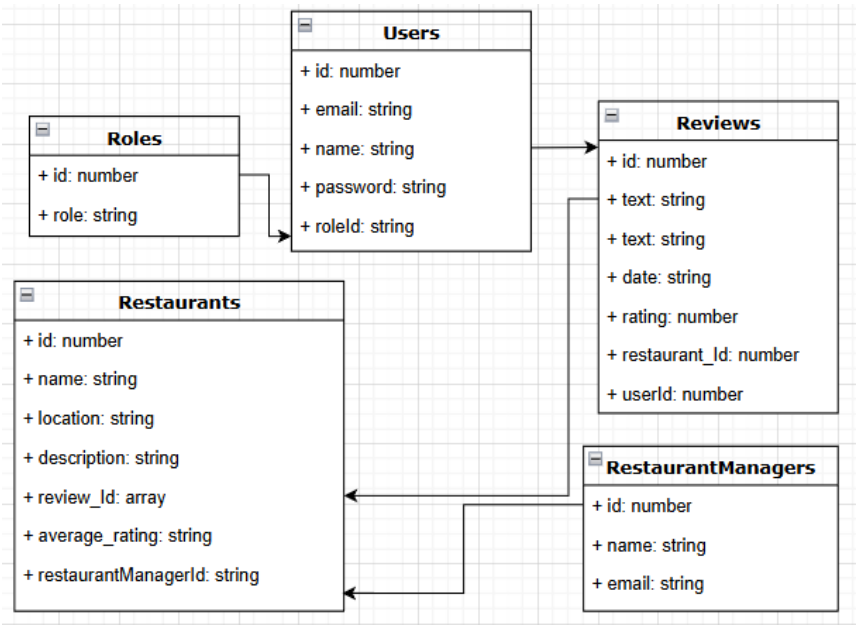

Рис. 3.4. Структура оперативної бази даних

Таблиця Restaurants описує заклади через сім атрибутів, включаючи ідентифікатор, назву, локацію, опис, масив ідентифікаторів відгуків, середній рейтинг та зовнішній ключ restaurantManagerId для зв'язку з менеджером.

Таблиця Reviews є центральною для аналітики і містить шість полів: ідентифікатор, два текстові поля для збереження контенту відгуку, дату публікації, числовий рейтинг, а також зовнішні ключі restaurant_Id та userId для зв'язку з таблицями Restaurants та Users відповідно.

Таблиця RestaurantManagers зберігає контактну інформацію відповідальних осіб через три поля - ідентифікатор, ім'я та електронну адресу. Зв'язки між таблицями реалізовані так: Users пов'язаний з Roles через roleId, Users з Reviews через userId, Reviews з Restaurants через restaurant_Id, а Restaurants з RestaurantManagers через restaurantManagerId. Наявність двох текстових полів у таблиці Reviews може використовуватися для зберігання оригінального та обробленого тексту або для багатомовної підтримки.

Така реляційна структура забезпечує нормалізацію даних, запобігає дублюванню інформації та дозволяє ефективно виконувати запити через операції JOIN для отримання комплексної інформації про ресторани, їхні відгуки та користувачів.

Аналітична база даних (рис.3.5) складається з трьох таблиць, оптимізованих для зберігання результатів машинного навчання та статистичних даних. Таблиця ReviewAnalysis є основною для зберігання результатів ML-обробки кожного відгуку і містить одинадцять полів: унікальний ідентифікатор, повний текст відгуку, загальну тональність у числовому форматі (-1/0/1), показник впевненості моделі типу float, текстову мітку класифікації, чотири числові поля для оцінки аспектів їжі, атмосфери, сервісу та ціни, дату та час аналізу, ідентифікатор ресторану та версію використаної моделі.

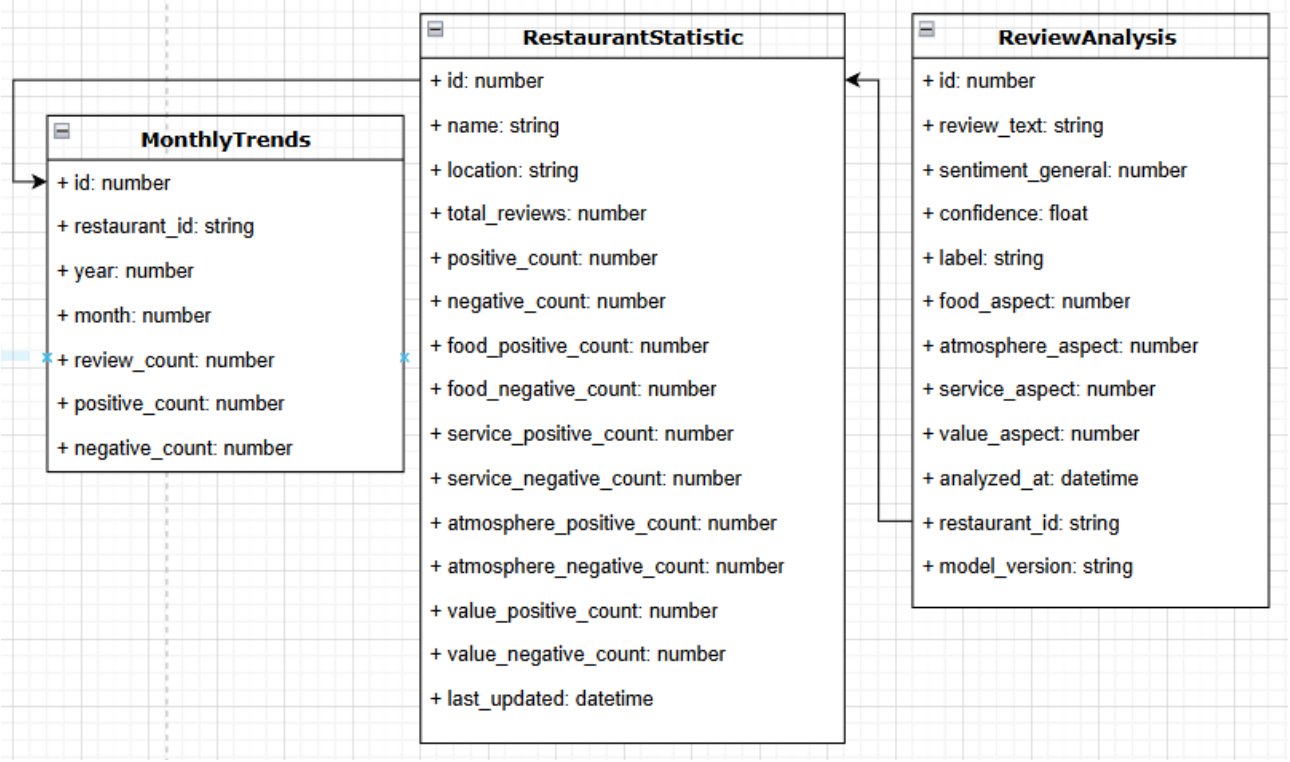

Рис. 3.5. Структура аналітичної бази даних

Таблиця RestaurantStatistic агрегує всі метрики по кожному ресторану через п'ятнадцять полів, включаючи базову інформацію (ідентифікатор, назва, локація), загальну кількість відгуків, розподіл за загальною тональністю (позитивні, негативні), а також детальний розподіл для кожного з чотирьох аспектів - їжа, сервіс, атмосфера та ціна - де для кожного аспекту зберігається кількість позитивних і негативних згадок, та дату останнього оновлення статистики. Таблиця MonthlyTrends відстежує динаміку змін у часі через сім полів: ідентифікатор, посилання на ресторан, рік і місяць, кількість відгуків за період та розподіл їх за тональністю (позитивні, негативні).

Зв'язки між таблицями реалізовані так: ReviewAnalysis пов'язана з RestaurantStatistic через поле restaurant_id, а MonthlyTrends також з'єднується з RestaurantStatistic через restaurant_id. Структура таблиці ReviewAnalysis точно відповідає форматові JSON-відповіді від ML-сервісу, що спрощує процес збереження результатів аналізу. Збереження тексту відгуку безпосередньо в аналітичній базі усуває необхідність звернення до оперативної бази при

формуванні звітів та дозволяє показувати конкретні приклади для ілюстрації статистики.

Таблиця RestaurantStatistic містить попередньо обчислені метрики, що значно прискорює формування звітів для менеджерів ресторанів без необхідності виконувати складні агрегаційні запити в реальному часі. Поле model_version у таблиці ReviewAnalysis дозволяє відстежувати, якою версією моделі було проаналізовано кожен відгук, що важливо при оновленні ML-моделей для порівняння результатів різних версій. Така структура оптимізована для швидкого читання даних та аналітичних запитів, жертвуючи дещо надлишковістю заради продуктивності, що є типовим підходом для аналітичних баз даних.

У даному розділі було розроблено комплексне теоретичне обґрунтування системи аналізу відгуків ресторанів з використанням методів машинного навчання та інтеграцією з веб-платформою.

Математичне обґрунтування охопило сім алгоритмів машинного навчання, кожен з яких базується на власному теоретичному апараті. Naive Bayes використовує байєсівську теорему та припущення умовної незалежності ознак для ймовірнісної класифікації. Random Forest як ансамбль дерев рішень максимізує інформаційний виграш при побудові розбиттів та агрегує прогнози через голосування більшості. Support Vector Machine знаходить оптимальну гіперплощину з максимальним відступом між класами, використовуючи ядрові функції для роботи в просторах високої розмірності. Logistic Regression моделює ймовірності через логістичну та softmax-функції з регуляризацією для запобігання перенавчанню. Gradient Boosting послідовно будує ансамбль моделей, де кожна наступна виправляє помилки попередніх через мінімізацію градієнта функції втрат. Decision Tree рекурсивно розбиває простір ознак за критерієм Gini для побудови інтерпретованих правил. K-Nearest Neighbors класифікує об'єкти на основі подібності до найближчих сусідів у просторі ознак.

Векторизація текстів методом TF-IDF забезпечує перетворення неструктурованих відгуків у числове представлення з урахуванням важливості

термінів на рівні документа та корпусу, що є необхідною передумовою для застосування алгоритмів машинного навчання. Формалізовані метрики оцінювання - accuracy, precision, recall та F1-score - надають об'єктивні критерії для порівняльного аналізу ефективності різних підходів на етапі експериментального дослідження.

Проведене моделювання системи за допомогою UML-діаграм дозволило формалізувати функціональні вимоги та визначити основні сценарії взаємодії користувачів з платформою. Діаграма прецедентів ідентифікувала три типи акторів - адміністратори, клієнти та менеджери ресторанів - кожен з яких має специфічні права доступу та можливості у системі. Діаграма класів визначила об'єктно-орієнтовану структуру з чітким розмежуванням відповідальностей між сутностями, де центральну роль відіграють класи Review для зберігання відгуків та FeedbackAnalysis для інтеграції ML-компонента.

Запропонована архітектура системи базується на принципах мікросервісної побудови та розділяє додаток на п'ять основних компонентів: веб-браузер як клієнтський інтерфейс, веб-сервер з Front-end та Back-end частинами на Next.js, оперативну базу даних для транзакційних операцій, аналітичну базу даних для результатів ML-обробки та окремий ML-API Server для виконання класифікації. Додатково виділено компоненти етапу підготовки даних - Google Maps Scraper для збору україномовних відгуків та Google Colab як середовище для експериментів з алгоритмами машинного навчання. Така архітектура забезпечує масштабованість через незалежність компонентів, модульність для легкого оновлення окремих частин, відмовостійкість при збоях окремих сервісів та локалізацію під специфіку українського ринку.

Структура оперативної бази даних включає п'ять взаємопов'язаних таблиць: Roles, Users, Restaurants, Reviews та RestaurantManagers - з нормалізованою схемою для запобігання дублюванню даних та забезпечення цілісності через зовнішні ключі. Аналітична база даних спроектована за принципом денормалізації з трьома таблицями - ReviewAnalysis для детальних результатів ML-аналізу кожного відгуку, RestaurantStatistic для попередньо

обчислених агрегованих метрик та MonthlyTrends для відстеження динаміки у часі. Денормалізована структура аналітичної бази забезпечує швидкість виконання складних аналітичних запитів за рахунок збереження обчислених результатів, що критично важливо для формування дашбордів у реальному часі.

Розроблені теоретичні основи створюють фундамент для практичної реалізації системи та проведення експериментальних досліджень. Формалізація функціональних вимог через UML-діаграми забезпечує чітке розуміння архітектури для всіх учасників розробки. Обґрунтування вибору мікросервісної архітектури з окремим ML-компонентом дозволяє незалежно масштабувати обчислювальні ресурси та оновлювати моделі без зупинки основного додатку. Роздільне проектування оперативної та аналітичної баз даних оптимізує систему як для транзакційних операцій, так і для складних аналітичних запитів. Математичне обґрунтування семи алгоритмів забезпечує наукову строгість дослідження та дозволяє обґрунтовано інтерпретувати результати порівняльного аналізу на наступних етапах роботи.

РОЗДІЛ 4. РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

4.1. Підготовка та аналіз даних

4.1.1. Основні характеристики та структура даних

Формування датасету для навчання моделей вимагало отримання автентичних україномовних відгуків про ресторани. Перший підхід базувався на використанні Google Places API - офіційного програмного інтерфейсу для доступу до даних Google Maps. Було виконано реєстрацію у Google Cloud Platform, отримано API-ключ та розроблено Python-скрипт(Додаток А) для автоматизованого збору відгуків. Проте тестування виявило фундаментальне обмеження API – текст відгуку повертаються англійською мовою через вбудований автоматичний переклад Google Translate, що унеможливлює отримання оригінальних українських текстів. Оскільки використання перекладених відгуків суттєво викривило б результати дослідження через втрату мовних особливостей, ідіоматичних виразів та культурного контексту, необхідно було розробити альтернативний метод збору даних безпосередньо з веб-інтерфейсу Google Maps.

Для отримання оригінальних україномовних відгуків було розроблено спеціалізовану програму Google Maps Reviews Scraper(Додаток Б) з використанням бібліотеки Selenium WebDriver. На відміну від офіційного API, який надає лише перекладені тексти, Selenium дозволяє автоматизувати роботу веб-браузера та безпосередньо взаємодіяти з веб-інтерфейсом Google Maps, отримуючи доступ до оригінальних текстів відгуків у тій мові, якою їх написали користувачі.

Програма виконує наступну послідовність операцій: здійснює пошук ресторану за назвою та географічною локацією, переходить на сторінку з відгуками, автоматично прокручує список для завантаження всіх доступних

відгуків через динамічне підвантаження контенту, розгортає згорнуті тексти через програмне натискання кнопки "Читати більше", витягує структуровані дані кожного відгуку (текст, оцінка, дата) з HTML-елементів, зберігає зібрані дані у CSV-файл з кодуванням UTF-8 для коректного збереження кирилиці. Таким чином було зібрано дані для подальшої їх підготовки для тренування моделі.

Зібрані відгуки потребували розмітки для створення навчального датасету. Розмітка класів виконувалась на основі числових оцінок, які користувачі залишали разом з текстовими коментарями на платформі Google Maps. Реалізовано бінарну схему класифікації: відгуки з оцінками 4-5 зірок позначено міткою 1 (позитивний клас), відгуки з оцінками 1-2 зірки позначено міткою 0 (негативний клас). Відгуки з оцінкою 3 зірки виключено з датасету на етапі попередньої обробки, оскільки вони представляють нейтральну тональність та можуть ускладнити навчання моделей бінарної класифікації через неоднозначність інтерпретації. Фрагмент датасету відображено нижче(рис.4.1).

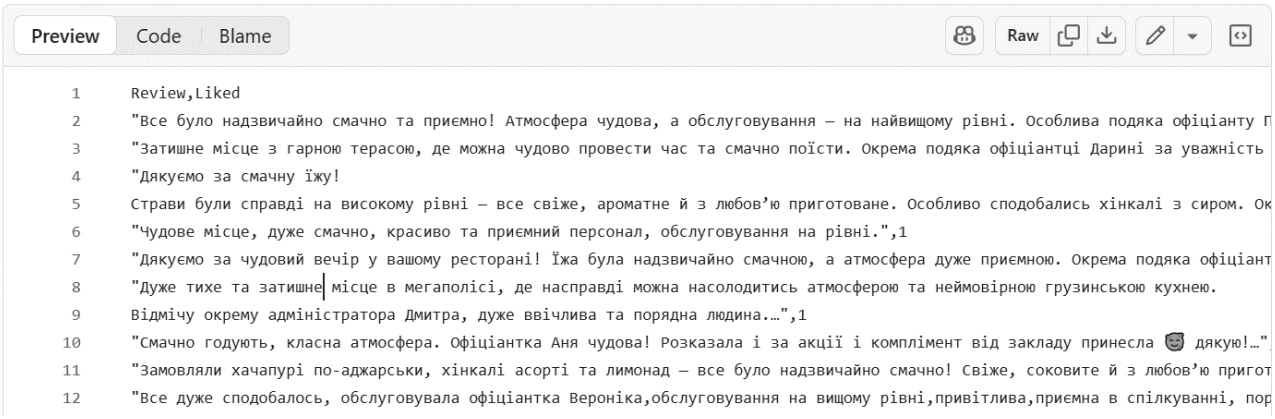

Рис.4.1 Відображення даних

Нижче (рис.4.2) відображена базова інформація про сформований датасет, отримана за допомогою методів shape, info та dtypes бібліотеки pandas. Оцінка датасету та тренування моделей було виконано в Google Colab (Додаток В) Датасет складається з 1663 рядків та 2 стовпців без пропущених значень.

Структура даних представлена наступним чином:

Review - текстова колонка формату object, що містить повний текст відгуку клієнта про ресторан;

Liked - бінарна цільова змінна формату int64, де 0 позначає негативний відгук, а 1 позначає позитивний.

Обсяг датасету у пам'яті становить 26.1 кілобайт. Відсутність пропущених значень (1663 non-null записів для обох стовпців) свідчить про успішність процесу збору та попередньої обробки даних.

```
============================================================
БАЗОВА ІНФОРМАЦІЯ ПРО ФАЙЛ
============================================================
Розмір даних: 1663 рядків, 2 стовпців

ЗАГАЛЬНА ІНФОРМАЦІЯ:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1663 entries, 0 to 1662
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Review  1663 non-null   object
 1   Liked   1663 non-null   int64
dtypes: int64(1), object(1)
memory usage: 26.1+ KB

ТИПИ ДАНИХ:

Review    object
Liked      int64
dtype: object

НАЗВИ СТОВПЦІВ (2):
 1. Review
 2. Liked
```

Рис.4.2 Інформація про файл

4.1.2. Оцінка якості даних та їх надійності

Проведений комплексний аналіз якості даних показав високі показники надійності досліджуваного датасету. Пропущені значення повністю відсутні у всіх колонках (рис. 4.3), що свідчить про успішність процесу збору та первинної обробки даних через розроблений Selenium scraper.

```
🔍 АНАЛІЗ ПРОПУЩЕНИХ ЗНАЧЕНЬ
----------------------------------------
       Колонка  Пропущені  Відсоток
Review  Review          0       0.0
Liked    Liked          0       0.0
✅ Пропущених значень не виявлено!
```

Рис.4.3 Відображення пропущених значень

Аналіз виявив 290 дублікатів, що становить 17.44% від загального обсягу датасету (рис. 2.4). При детальному розгляді повторюваних записів разом з їх оригіналами виявлено 508 рядків, що містять ідентичну інформацію, з яких 322 записи належать до негативного класу та 186 - до позитивного класу. Висока частота дублікатів пояснюється декількома факторами: повторюваність типових фраз та мовних конструкцій у відгуках про ресторани ("смачна їжа", "гарне обслуговування"), збір відгуків від активних користувачів, які залишають схожі коментарі для різних закладів, також можливе дублювання під час форматування датасету нашою програмою.

```
🔄 АНАЛІЗ ДУБЛІКАТІВ
------------------------
Загальні дублікати: 290
Дублікати тексту: 290
Відсоток дублікатів: 17.44%

📄 Приклади дублікатів:
                                                Review  Liked
190  30.11 відпочивали в першій кабінці, бармен зіп...      0
191  За тиждень забронювали столик та попередили, щ...      0
192  Ресторан і кухня більш менш. Обслуговування 3 ...      0
193  Сам заклад та його місцезнаходження чудове, жи...      0
194  Даю одну зірку лише за місце для дітей +парк,а...      0

🎯 Дублікати по класах:
Liked
0    322
1    186
Name: count, dtype: int64

============================================================
3️⃣.1 ВИДАЛЕННЯ ДУБЛІКАТІВ
🗑️ ВИДАЛЕННЯ ДУБЛІКАТІВ
------------------------------
Початковий розмір: 1663
Видалено дублікатів: 290
Фінальний розмір: 1373
Відсоток видалених: 17.44%

📊 Розподіл класів після очищення:
Liked
1    732
0    641
Name: count, dtype: int64

🎯 Коефіцієнт імбалансу (IR): 1.14
```

Рис.4.4. Відображення інформації про дублікати

Для забезпечення коректності навчання моделей та запобігання переоцінці точності через наявність ідентичних прикладів у навчальній та тестовій вибірках, всі виявлені дублікати було видалено на етапі попередньої обробки методом

drop_duplicates() з параметром keep='first', що зберігає перше входження кожного унікального відгуку. Після видалення 290 дублікатів датасет містить 1373 унікальних відгуки.

Розподіл класів цільової змінної після очищення демонструє незначний перекіс (рис. 4.5): позитивні відгуки складають 732 записи, негативні - 641 запис.

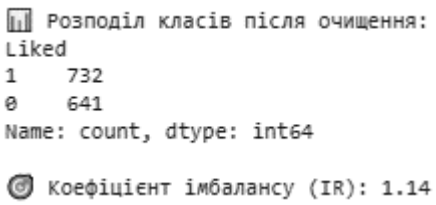

Рис.4.5. Розподіл класів після видалення дублікатів

Коефіцієнт імбалансу, який обраховується відношенням більшого класу до меншого класу, дорівнює 1.14, що вказує на практично збалансований датасет з мінімальним перекосом, що знаходиться значно нижче порогового значення 1.5 для помірного імбалансу. При такому низькому рівні дисбалансу алгоритми машинного навчання здатні ефективно навчатися на природному розподілі даних без необхідності застосування спеціальних методів балансування.

4.1.3. Детальний аналіз текстових характеристик

Статистичний аналіз текстових даних виявив наступні закономірності структури українськомовних відгуків про ресторани (рис. 4.6, 4.7). Середня довжина відгуку складає 185.21 символ при стандартному відхиленні 117.28, що вказує на значну варіативність у довжині текстів - від коротких однорічних коментарів до розгорнутих детальних оглядів. Діапазон довжини коливається від мінімальних 3 символів до максимальних 1519 символів, при цьому медіанне значення становить 183 символи, що свідчить про приблизно симетричний розподіл довжини відгуків навколо центрального значення.

📊 Статистика текстових метрик:

	Кількість символів	Кількість слів	Кількість речень
count	1373.00	1373.00	1373.00
mean	185.21	28.08	3.61
std	117.28	18.46	2.12
min	3.00	1.00	1.00
25%	118.00	17.00	2.00
50%	183.00	27.00	3.00
75%	237.00	36.00	5.00
max	1519.00	231.00	21.00

	Середня довжина слова	Кількість знаків оклику
count	1373.00	1373.00
mean	5.78	0.74
std	0.88	1.89
min	3.00	0.00
25%	5.22	0.00
50%	5.69	0.00
75%	6.21	1.00
max	12.00	25.00

	Кількість знаків питання	Коефіцієнт великих літер	Показник складності
count	1373.00	1373.00	1373.00
mean	0.04	0.00	6.16
std	0.25	0.00	1.80
min	0.00	0.00	1.80
25%	0.00	0.00	5.23
50%	0.00	0.00	5.88
75%	0.00	0.00	6.55
max	4.00	0.05	17.33

Рис.4.6 Відображення статистики текстових метрик

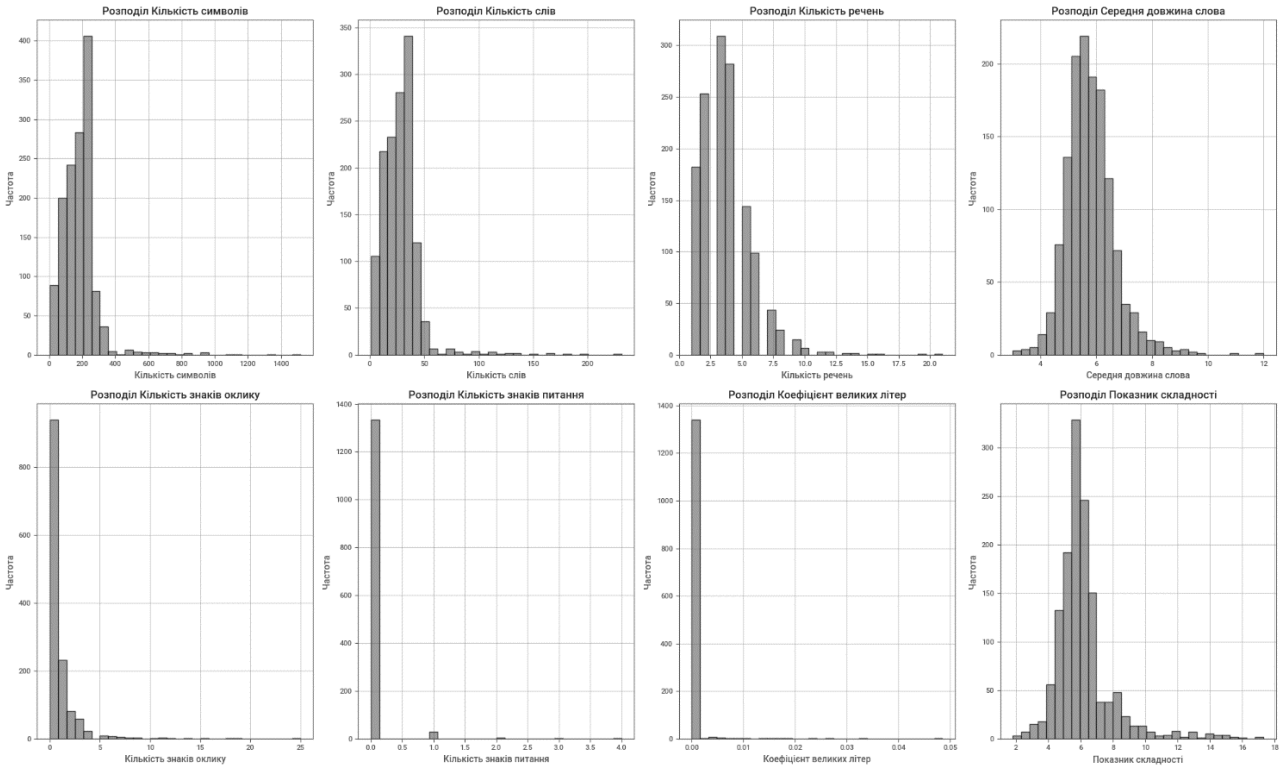

Рис.4.7 Розподіл текстових характеристик у датасеті ресторанних відгуків: довжина тексту, структурні та емоційні показники

Аналіз структури тексту показав, що середній відгук містить 28.08 слів при стандартному відхиленні 18.46 слів, організованих у 3.61 речення. Квартильний аналіз виявив, що 25% найкоротших відгуків містять до 17 слів, медіана знаходиться на рівні 27 слів, а 75% відгуків не перевищують 36 слів. Середня довжина слова становить 5.78 символів при стандартному відхиленні 0.88, що відповідає типовим характеристикам української мови з її багатою морфологією та складною системою словозміни.

Емоційне забарвлення текстів оцінювалося через частоту використання експресивних пунктуаційних знаків. Середня кількість знаків оклику складає 0.74 на відгук при стандартному відхиленні 1.89, що вказує на правосторонню асиметрію розподілу - більшість відгуків містять 0-1 знак оклику, проте окремі емоційні коментарі можуть містити до 25 знаків оклику. Знаки питання використовуються значно рідше - в середньому 0.04 на відгук, що свідчить про переважно стверджувальний характер текстів відгуків. Коефіцієнт використання великих літер становить 0.00 (округлено), що відповідає нормам української орфографії з помірним використанням капіталізації.

Показник складності тексту, що обчислюється як комбінація середньої довжини слова, кількості речень та щільності слів у реченні, становить у середньому 6.16 при стандартному відхиленні 1.80. Діапазон складності коливається від 1.80 (найпростіші короткі відгуки) до 17.33 (складні розгорнуті рецензії з розвиненою структурою речень). Медіанне значення показника складності 5.88 вказує на те, що типовий відгук має помірний рівень складності, доступний для розуміння широкою аудиторією.

Розподіл всіх текстових метрик наближається до нормального з правосторонньою асиметрією для показників довжини, що типово для користувацького контенту, де домінують короткі та середні за обсягом тексти, а довгі детальні відгуки зустрічаються рідше. Такі статистичні характеристики є важливими для розуміння специфіки датасету та вибору оптимальних параметрів векторизації на наступних етапах обробки.

Порівняльний аналіз між класами тональності (рис.4.8) виявив статистично значущі закономірності у структурі та стилістиці відгуків різної полярності. Негативні відгуки (клас 0) виявилися істотно довшими за позитивні: середня довжина становить 199.25 символів проти 172.91 символів у позитивних відгуках (клас 1), що становить різницю у 15.2%. Аналогічна тенденція спостерігається у кількості слів - негативні відгуки містять у середньому 31.36 слів проти 25.20 слів у позитивних, тобто на 24.4% більше. Кількість речень також вища у негативних відгуках (3.80 проти 3.44), що свідчить про те, що незадоволені клієнти схильні детальніше описувати свій досвід та аргументувати критичні зауваження.

🎯 ПОРІВНЯННЯ ПО КЛАСАХ (Liked):

📋 Клас 1 (n=732):

	Кількість символів	Кількість слів	Кількість речень	\
mean	172.91	25.20	3.44	
std	118.26	17.77	2.05	

	Середня довжина слова	Кількість знаків оклику	\
mean	6.02	0.79	
std	0.88	1.46	

	Кількість знаків питання	Коефіцієнт великих літер	Показник складності
mean	0.00	0.0	6.00
std	0.06	0.0	1.59

📋 Клас 0 (n=641):

	Кількість символів	Кількість слів	Кількість речень	\
mean	199.25	31.36	3.80	
std	114.64	18.70	2.19	

	Середня довжина слова	Кількість знаків оклику	\
mean	5.49	0.69	
std	0.80	2.29	

	Кількість знаків питання	Коефіцієнт великих літер	Показник складності
mean	0.08	0.0	6.33
std	0.36	0.0	2.00

Рис.4.8 Порівняння текстових метрик між класами

Цікава закономірність виявлена у середній довжині слів: позитивні відгуки характеризуються довшими словами (6.02 символи) порівняно з негативними (5.49 символів), що може бути пов'язано з використанням більш витончених епітетів та описових прикметників для вираження позитивних емоцій ("неймовірний", "фантастичний", "чудовий") замість коротших негативних оцінок ("погано", "жахливо"). Стандартне відхилення середньої довжини слова

схоже для обох класів (0.88 для позитивних та 0.80 для негативних), що вказує на стабільність цієї характеристики.

Емоційне забарвлення, що виражається через кількість знаків оклику, демонструє неочікуваний результат: позитивні відгуки містять в середньому 0.79 знаків оклику проти 0.69 у негативних, проте різниця невелика. Однак варіативність у негативних відгуках значно вища (стандартне відхилення 2.29 проти 1.46), що свідчить про більшу експресивність окремих критичних коментарів. Знаки питання практично відсутні у позитивних відгуках (0.00) та рідко зустрічаються у негативних (0.08), що підтверджує переважно ствердувальний характер обох типів відгуків.

Показник складності тексту, що обчислюється як комбінація середньої довжини слова, кількості речень та щільності слів у реченні, виявився дещо вищим для негативних відгуків (6.33 проти 6.00), що узгоджується з їх більшою довжиною та деталізацією. Стандартне відхилення показника складності також вище для негативного класу (2.00 проти 1.59), що вказує на більшу різноманітність структур негативних відгуків - від коротких емоційних висловлювань до розгорнутих аргументованих критичних оглядів.

Виявлені відмінності у текстових характеристиках між класами мають важливе значення для процесу класифікації: довжина тексту, кількість слів, середня довжина слова та показник складності можуть виступати як додаткові інформативні ознаки поряд з TF-IDF векторами, потенційно покращуючи якість розпізнавання тональності алгоритмами машинного навчання.

Кореляційний аналіз (рис. 4.9) виявив систему структурних зв'язків між текстовими характеристиками при відсутності сильних кореляцій з цільовою змінною тональності. Цільова змінна Liked демонструє слабкі кореляції з усіма текстовими метриками: найвища з середньою довжиною слова ($r = 0.30$), що підтверджує виявлену раніше тенденцію використання довших слів у позитивних відгуках, та від'ємні кореляції з кількістю символів ($r = -0.11$), кількістю слів ($r = -0.17$) та кількістю речень ($r = -0.09$), що узгоджується з меншою довжиною позитивних відгуків.

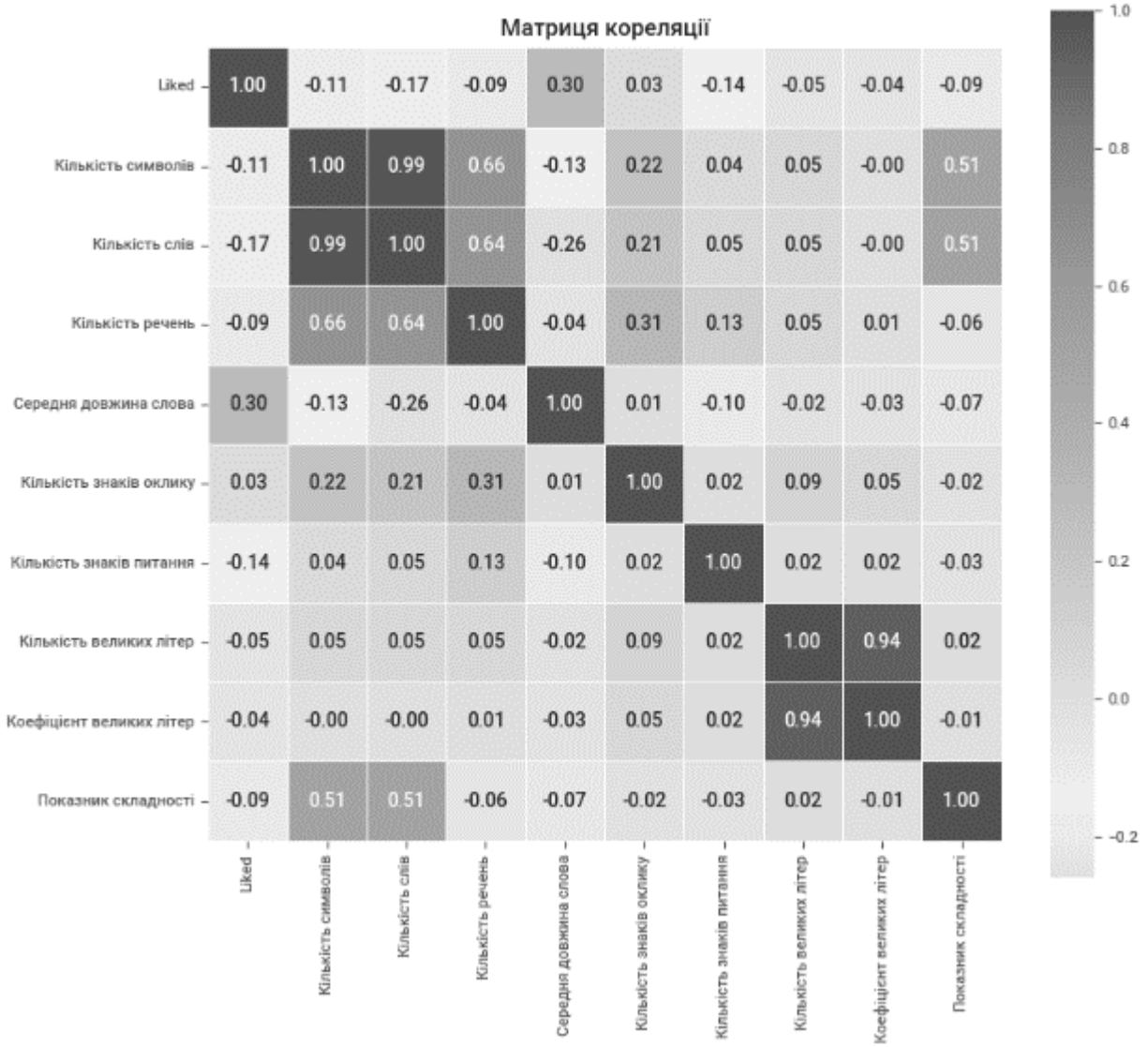

Рис.4.9 Матриця кореляції

Найсильніші кореляції спостерігаються між природно пов'язаними метриками довжини тексту. Кількість символів та кількість слів демонструють майже функціональний зв'язок (r = 0.99), що є очікуваним, оскільки обидві метрики вимірюють обсяг тексту. Аналогічно високою є кореляція між кількістю слів та кількістю речень (r = 0.64), що відображає природну залежність - довші тексти містять більше речень. Показник складності тексту також сильно корелює з кількістю символів (r = 0.51) та кількістю слів (r = 0.51), оскільки складність частково визначається обсягом тексту.

Цікавою є негативна кореляція між кількістю слів та середньою довжиною слова (r = -0.26), яка свідчить про тенденцію використання простішої та коротшої

лексики у довших відгуках. Це може пояснюватися тим, що розгорнуті описи вимагають використання службових частин мови та сполучників, що знижує середню довжину слова, тоді як короткі відгуки часто містять лише ключові оціночні прикметники та іменники.

Емоційні маркери (знаки оклику та питання) демонструють слабкі кореляції з іншими характеристиками. Кількість знаків оклику має помірну позитивну кореляцію з кількістю символів ($r = 0.22$), кількістю слів ($r = 0.21$) та кількістю речень ($r = 0.31$), що вказує на те, що довші відгуки мають більше можливостей для емоційного вираження. Коефіцієнт великих літер та кількість знаків питання практично не корелюють з іншими метриками, що свідчить про їх незалежний характер використання.

Відсутність сильних кореляцій між текстовими метриками та цільовою змінною (всі $|r| < 0.3$) вказує на те, що тональність відгуку визначається переважно семантичним змістом тексту, а не його структурними характеристиками. Це обґрунтовує необхідність використання TF-IDF векторизації для вилучення смислових ознак, оскільки прості статистичні метрики довжини та структури недостатні для ефективної класифікації тональності українськомовних відгуків про ресторани.

4.1.4. Аспектно-орієнтований аналіз змісту

Для поглибленого розуміння структури українськомовних відгуків про ресторани застосовано методологію аспектно-орієнтованого аналізу тональності (Aspect-Based Sentiment Analysis). На відміну від класичного підходу, що визначає лише загальну полярність відгуку, аспектний аналіз дозволяє виявити специфічні компоненти ресторанного досвіду та оцінити їх окремий внесок у формування загального враження клієнта.

На основі аналізу специфіки ресторанного бізнесу, огляду літератури з індустрії гостинності у зібраному корпусі відгуків виділено чотири критичні аспекти оцінювання закладів харчування. Їжа (Food) охоплює якість страв,

смакові характеристики, свіжість інгредієнтів, текстуру, температуру подачі та загальну кулінарну майстерність. Цей аспект включає оцінку різноманітності меню, розміру порцій та відповідності очікуванням клієнта щодо гастрономічного досвіду. Обслуговування (Service) характеризує якість взаємодії з персоналом, швидкість сервісу, ввічливість та професіоналізм офіціантів, увагу до деталей, здатність вирішувати проблемні ситуації та загальну клієнтоорієнтованість закладу. Атмосфера (Atmosphere) відображає фізичне середовище закладу - інтер'єр, музичний супровід, освітлення, чистоту, рівень шуму, комфортність меблів, естетику простору та загальну емоційну атмосферу. Ціна (Value) оцінює економічну складову відвідування через призму загальної цінності за гроші. Для кожного аспекту сформовано спеціалізовані словники термінів з позитивною та негативною конотацією (рис. 4.12).

```
# Словники ключових слів для 4 аспектів
self.aspect_keywords = {
    'food': {
        'positive': [
            # Загальні позитивні
            'смачно', 'смачний', 'смачна', 'смачне', 'смачні',
            'дуже смачно', 'неймовірно смачно', 'чудово', 'чудовий', 'чудова', 'чудове',
            'відмінно', 'відмінний', 'відмінна', 'відмінне', 'відмінні',
            'прекрасно', 'прекрасний', 'прекрасна', 'прекрасне', 'прекрасні',
            'класно', 'класний', 'класна', 'класне', 'класні',
            'супер', 'топ', 'бомба', 'огонь',

            # Свіжість
            'свіжий', 'свіжа', 'свіже', 'свіжі', 'свіжість',
            'свіжайший', 'свіжайша', 'свіжайше', 'свіжайші',

            # Текстура та смак
            'соковитий', 'соковита', 'соковите', 'соковиті',
            'ніжний', 'ніжна', 'ніжне', 'ніжні', 'ніжність',
            'м\'який', 'м\'яка', 'м\'яке', 'м\'які', 'м\'якість',
            'хрустящий', 'хрустяща', 'хрустяще', 'хрустящі',
            'ароматний', 'ароматна', 'ароматне', 'ароматні', 'аромат',
            'пікантний', 'пікантна', 'пікантне', 'пікантні',

            # Приготування
            'добре приготовлений', 'добре приготовлена', 'добре приготовлене',
            'ідеально приготований', 'ідеально приготовлена', 'ідеально приготовлене',
            'майстерно приготований', 'професійно приготований',

            # Емоційні оцінки
            'неймовірний', 'неймовірна', 'неймовірне', 'неймовірні',
            'фантастичний', 'фантастична', 'фантастичне', 'фантастичні',
            'божественний', 'божественна', 'божественне', 'божественні',
            'розкішний', 'розкішна', 'розкішне', 'розкішні',

            # Страви
            'делікатес', 'делікатесний', 'делікатесна', 'делікатесне',
            'шедевр', 'кулінарний шедевр',
            'найкращий', 'найкраща', 'найкраще', 'найкращі',

            # Загальні позитивні слова (додано для кращого розпізнавання)
            'гарний напій', 'гарна страва', 'гарне меню', 'гарні страви', 'добрий смак', 'добра їжа', 'добре', 'добрі страви'
        ],
        'negative': [
            # Загальні негативні
            'несмачно', 'несмачний', 'несмачна', 'несмачне', 'несмачні',
            'огидно', 'огидний', 'огидна', 'огидне', 'огидні',
            'жахливо', 'жахливий', 'жахлива', 'жахливе', 'жахливі',
            'погано', 'поганий', 'погана', 'погане', 'погані',
            'відвратно', 'відвратний', 'відвратна', 'відвратне', 'відвратні',
```

Рис.4.12 Словник ключових слів

Критичною проблемою аналізу тональності українськомовних текстів є коректна інтерпретація заперечувальних конструкцій, які інвертують полярність оцінки. Конструкції типу "не смачний", "не швидко", "не затишно" повинні розпізнаватися як негативні, навіть якщо містять формально позитивне слово. Для вирішення цієї проблеми було розроблено алгоритм автоматичної обробки заперечень через систему регулярних виразів, що замінює патерни "не + позитивне_слово" на відповідні негативні еквіваленти перед етапом аспектного аналізу. Така попередня обробка забезпечує коректне віднесення заперечувальних фраз до негативного класу при подальшому підрахунку аспектних метрик та значно покращує точність аспектного аналізу для української мови.

```
# Функція для підрахунку аспект-специфічних слів
def count_aspect_sentiment(text, aspect_dict):
    text_lower = str(text).lower()
    positive_count = sum(1 for word in aspect_dict['positive'] if word in text_lower)
    negative_count = sum(1 for word in aspect_dict['negative'] if word in text_lower)

    return {
        'positive': positive_count,
        'negative': negative_count,
        'net_sentiment': positive_count - negative_count,
        'total_mentions': positive_count + negative_count
    }

# Застосування до кожного аспекту
for aspect_name, aspect_words in self.aspect_keywords.items():
    print(f"   📊 Обробка аспекту: {aspect_name}")

    # Підрахунок для кожного тексту
    aspect_scores = processed_data[text_column].apply(
        lambda x: count_aspect_sentiment(x, aspect_words)
    )

    # Створення окремих колонок
    processed_data[f'{aspect_name}_positive_count'] = [score['positive'] for score in aspect_scores]
    processed_data[f'{aspect_name}_negative_count'] = [score['negative'] for score in aspect_scores]
    processed_data[f'{aspect_name}_net_sentiment'] = [score['net_sentiment'] for score in aspect_scores]
    processed_data[f'{aspect_name}_total_mentions'] = [score['total_mentions'] for score in aspect_scores]

    # Нормалізована оцінка аспекту (від -1 до 1)
    processed_data[f'{aspect_name}_sentiment_ratio'] = processed_data.apply(
        lambda row: row[f'{aspect_name}_net_sentiment'] / max(row[f'{aspect_name}_total_mentions'], 1),
        axis=1
    )
```

Рис.4.13 Функція для визначення тональності відгуків

Для визначення тональності було написано функцію(рис.4.13), яка працює через підрахунок позитивних та негативних ключових слів для кожного аспекту

(їжа, сервіс, атмосфера, ціна) у тексті відгуку на основі попередньо створеного словника(рис.4.12). Потім вона обчислює sentiment_ratio як різницю між кількістю позитивних та негативних слів, поділену на загальну кількість згадувань аспекту, що дає значення від -1 (повністю негативний) до +1 (повністю позитивний).

Застосування методів аспектно-орієнтованого аналізу дозволило дослідити частоту згадування виділених аспектів ресторанного досвіду, які найчастіше згадуються гостями у відгуках(рис.4.13).

```
Аспектна статистика:
  Food        : 1549 згадувань, 852 відгуків (62.1%), тональність: +0.348
  Service     : 1544 згадувань, 680 відгуків (49.5%), тональність: +0.217
  Atmosphere  :  678 згадувань, 440 відгуків (32.0%), тональність: +0.227
  Value       :   50 згадувань,  44 відгуків (3.2%), тональність: +0.018
```

Рис.4.13 Аспектна статистика

Аспект їжі виявився найбільш поширеним з 1549 згадуваннями у 852 відгуках, що становить 62.1% покриття датасету. Середня тональність їжі становить +0.348, що вказує на переважно позитивні оцінки якості страв клієнтами. Високе покриття цього аспекту підтверджує гіпотезу про те, що якість їжі є найважливішим фактором при оцінюванні ресторанного досвіду відвідувачами.

Аспект обслуговування згадується 1544 рази у 680 відгуках (49.5% покриття) з середньою тональністю +0.217. Менша позитивність порівняно з їжею вказує на більшу варіативність у якості сервісу та наявність проблемних ситуацій з персоналом у частині закладів.

Аспект атмосфери представлений 678 згадуваннями у 440 відгуках (32.0% покриття) з тональністю +0.227. Відносно невисоке покриття свідчить про те, що клієнти менш схильні детально описувати фізичне середовище закладу, фокусуючись на більш практичних аспектах їжі та обслуговування.

Аспект цінності згадується найрідше - лише 50 разів у 44 відгуках (3.2% покриття) з найнижчою тональністю +0.018, близькою до нейтральної. Така низька частота згадувань може пояснюватися культурними особливостями -

обговорення цін у публічних відгуках може сприйматися як недоречне, або клієнти просто рідко мають нарікання на співвідношення ціна-якість у відвідуваних закладах.

4.2. Векторизація та підготовка даних для машинного навчання

Після комплексної попередньої обробки та витягування аспектних ознак датасет було підготовлено для навчання моделей класифікації через векторизацію текстів та формування фінальної матриці ознак.

Векторизація очищених текстів(рис.4.14) здійснювалася методом TF-IDF (Term Frequency-Inverse Document Frequency) з наступними параметрами: максимальний розмір словника 5000 найчастіших термінів (max_features=5000), діапазон N-грам від 1 до 2 (ngram_range=(1,2)) для врахування як окремих слів, так і біграм, мінімальна частота документів 5 (min_df=5) для фільтрації рідкісних термінів. Векторизація проводилася на текстах після видалення стоп-слів, що дозволило зменшити шум та зосередитися на семантично значущих термінах.

```
📊 ВЕКТОРИЗАЦІЯ ТЕКСТУ (TFIDF, UK)
----------------------------------
Розмір словника: 5000
Розмір матриці: (1373, 5000)
Розрідженість: 0.0%
🎯 ПІДГОТОВКА ДАНИХ ДЛЯ ML З АСПЕКТАМИ
------------------------------------------
Загальна кількість ознак: 5029
  - Текстові ознаки (TF-IDF): 5000
  - Аспектні ознаки: 16
  - Загальні ознаки: 13
Розмір тренувальної вибірки: (1098, 5029)
Розмір тестової вибірки: (275, 5029)
Розподіл класів (тренування): [513 585]
Розподіл класів (тест): [128 147]
```

Рис.4.14 Візуалізація векторизації тексту

Результуюча TF-IDF матриця має розмір 1373×5000, де кожен рядок відповідає окремому відгуку, а кожен стовпець - терміну зі словника. На відміну від типових текстових даних з високою розрідженістю (98-99%), отримана матриця демонструє нульову розрідженість (0.0%), що свідчить про багатий

словниковий склад українськомовних відгуків та високу покритість термінів зі словника у текстах після обробки заперечень та аспектного аналізу.

До TF-IDF векторів додано 29 додаткових структурованих ознак: 16 базових аспектних метрик (по 4 метрики для кожного з 4 аспектів: positive_count, negative_count, net_sentiment, sentiment_ratio), 6 агрегованих аспектних ознак (total_aspect_mentions, total_positive_aspects, total_negative_aspects, aspect_balance, aspect_ambivalence, overall_aspect_sentiment), 7 базових текстових характеристик (text_length, word_count, sentence_count, avg_word_length, exclamation_count, question_count, caps_words).

Числові ознаки масштабовані методом StandardScaler для приведення до нульового середнього та одиничної дисперсії, що забезпечує коректну роботу алгоритмів, чутливих до масштабу ознак (Logistic Regression, SVM, KNN). Після масштабування числові ознаки горизонтально об'єднані з TF-IDF матрицею у єдину розріджену матрицю ознак, що дозволяє ефективно працювати з великорозмірними даними без значних витрат пам'яті.

Фінальна розмірність простору ознак становить 5029 параметрів (5000 TF-IDF термінів плюс 16 аспектних ознак плюс 13 текстових характеристик), що створює високорозмірне представлення кожного відгуку для навчання моделей машинного навчання. Така комбінація дозволяє моделям використовувати як неструктуровану семантичну інформацію з TF-IDF векторів, так і структуровані знання про аспекти та текстові властивості відгуків.

Датасет розділено на навчальну та тестову вибірки у співвідношенні 80/20 методом стратифікованого семплування (stratified split), що гарантує збереження пропорцій класів у обох підмножинах. Навчальна вибірка містить 1098 відгуків, тестова - 275 відгуків. Розподіл класів у навчальній вибірці: 585 позитивних та 513 негативних; у тестовій: 147 позитивних та 128 негативних. Така незначна різниця у пропорціях між навчальною та тестовою вибірками (менше 0.3%) підтверджує ефективність стратифікованого підходу.

Для оцінки стабільності результатів та виявлення можливого перенавчання застосовується 5-кратна крос-валідація (5-fold cross-validation) на навчальній

вибірці. Датасет розбивається на 5 рівних частин (фолдів) приблизно по 220 відгуків у кожному, модель послідовно навчається на 4 фолдах (близько 878 відгуків) та валідується на п'ятому (220 відгуків), процес повторюється 5 разів з різними комбінаціями навчальних та валідаційних фолдів. Фінальна оцінка якості обчислюється як середнє значення метрик по всіх 5 ітераціях з розрахунком стандартного відхилення, що дає більш надійну оцінку продуктивності та її варіативності порівняно з одноразовим розділенням.

Тестова вибірка залишається повністю ізольованою від процесу навчання та крос-валідації, використовується лише для фінального оцінювання узагальнюючої здатності моделей на нових, раніше не бачених даних. Така методологія відповідає сучасним стандартам машинного навчання та забезпечує об'єктивне порівняння семи досліджуваних алгоритмів за однаковими умовами експерименту.

4.3. Тренування моделей машинного навчання

Процес тренування реалізовано через клас MLAlgorithmsComparison(додаток Д), що забезпечує уніфікований підхід до навчання та оцінювання всіх досліджуваних алгоритмів. Навчальна вибірка розміром 1098 прикладів (80% від 1373) використовується для тренування моделей, тестова вибірка 275 прикладів (20%) - для незалежної оцінки продуктивності.

Ініціалізація моделей з базовими гіперпараметрами. Для кожного з семи алгоритмів встановлено базову конфігурацію гіперпараметрів (рис.4.15), що забезпечує баланс між якістю класифікації та обчислювальною ефективністю. Random Forest та Gradient Boosting налаштовані на 100 базових моделей (дерев), що є стандартною практикою для датасетів середнього розміру. Decision Tree обмежено максимальною глибиною 10 рівнів для запобігання перенавчанню на

тренувальних даних. Всі моделі ініціалізовані з фіксованим random_state=42 для забезпечення відтворюваності результатів експерименту.

```
def initialize_models(self):
    """Ініціалізація різних ML моделей"""
    self.models = {
        'Logistic Regression': LogisticRegression(
            random_state=42,
            max_iter=1000,
            solver='liblinear'
        ),
        'Random Forest': RandomForestClassifier(
            n_estimators=100,
            random_state=42,
            n_jobs=-1
        ),
        'Support Vector Machine': SVC(
            kernel='rbf',
            random_state=42,
            probability=True
        ),
        'Naive Bayes': GaussianNB(),
        'Gradient Boosting': GradientBoostingClassifier(
            n_estimators=100,
            random_state=42
        ),
        'Decision Tree': DecisionTreeClassifier(
            random_state=42,
            max_depth=10
        ),
        'K-Nearest Neighbors': KNeighborsClassifier(
            n_neighbors=5,
            random_state=42
        )
    }
```

Рис.4.15 Візуалізація конфігурації моделей

Обробка розріджених матриць. Більшість алгоритмів (Random Forest, Logistic Regression, SVM, Decision Tree, KNN) безпосередньо підтримують роботу з розрідженими scipy.sparse матрицями, що забезпечує ефективне використання оперативної пам'яті. Проте Naive Bayes (GaussianNB) вимагає dense представлення даних для обчислення статистик гаусівського розподілу, тому для цього алгоритму виконується попередня конвертація матриць через метод .toarray(). Це збільшує споживання пам'яті з 45 МБ (sparse) до 180 МБ (dense) для навчальної вибірки, але є необхідним для коректної роботи алгоритму. Процес навчання та вимірювання часу. Для кожної моделі виконується навчання методом .fit(X_train, y_train) з одночасним вимірюванням часу тренування через модуль time.

```python
def train_and_evaluate_models(self, X_train, y_train, X_test, y_test):
    """Тренування та оцінка всіх моделей"""
    self.logger.info("🚀 Початок тренування та оцінки моделей...")
    print("\n🚀 ТРЕНУВАННЯ ТА ОЦІНКА МОДЕЛЕЙ")
    for model_name, model in self.models.items():
        print(f"\n📊 Тренування: {model_name}")
        print("-" * 40)
        try:
            # Засічка часу
            start_time = time.time()
             # Спеціальна обробка для Naive Bayes
            if model_name == 'Naive Bayes':
                # Конвертація sparse -> dense
                X_train_dense = X_train.toarray()
                X_test_dense = X_test.toarray()
                # Тренування з dense матрицями
                model.fit(X_train_dense, y_train)
                training_time = time.time() - start_time
                # Прогнозування
                start_time = time.time()
                y_pred = model.predict(X_test_dense)
                prediction_time = time.time() - start_time
                # Ймовірності для ROC-AUC
                y_pred_proba = model.predict_proba(X_test_dense)[:, 1]
                # Крос-валідація з dense матрицями
                cv_scores = cross_val_score(model, X_train_dense, y_train, cv=5, scoring='f1_weighted')
            else:
                # Звичайне тренування для всіх інших моделей
                model.fit(X_train, y_train)
                training_time = time.time() - start_time
                # Прогнозування
                start_time = time.time()
                y_pred = model.predict(X_test)
                prediction_time = time.time() - start_time
                # Ймовірності для ROC-AUC
                if hasattr(model, 'predict_proba'):
                    y_pred_proba = model.predict_proba(X_test)[:, 1]
                else:
                    y_pred_proba = model.decision_function(X_test)
                # Крос-валідація
                cv_scores = cross_val_score(model, X_train, y_train, cv=5, scoring='f1_weighted')
            # Обчислення метрик
            metrics = {
                'accuracy': accuracy_score(y_test, y_pred),
                'precision': precision_score(y_test, y_pred, average='weighted'),
                'recall': recall_score(y_test, y_pred, average='weighted'),
                'f1_score': f1_score(y_test, y_pred, average='weighted'),
                'roc_auc': roc_auc_score(y_test, y_pred_proba),
                'cv_mean': cv_scores.mean(),
                'cv_std': cv_scores.std(),
                'training_time': training_time,
                'prediction_time': prediction_time
            }
            # Збереження результатів
            self.results[model_name] = {
                'model': model,
                'metrics': metrics,
                'predictions': y_pred,
                'probabilities': y_pred_proba,
                'cv_scores': cv_scores
```

Рис.4.16 Візуалізація функції тренування моделей

Після базового тренування кожна модель проходить 5-кратну стратифіковану крос-валідацію для оцінки стабільності результатів на різних підмножинах даних. Навчальна вибірка розділяється на 5 фолдів зі збереженням

пропорцій класів, модель тренується п'ять разів на різних комбінаціях чотирьох фолдів та валідується на п'ятому.

Для кожної моделі розраховуються метрики, що дозволяють багатовимірно оцінити продуктивність: accuracy (загальна точність), precision і recall (weighted для урахування класів), F1-Score (гармонійне середнє), ROC-AUC (якість ймовірнісних оцінок), середнє та стандартне відхилення крос-валідації, час тренування та прогнозування. Weighted метрики зважують результати по класах пропорційно їх представленості у тестовій вибірці, що забезпечує справедливу оцінку при дисбалансі.

Після завершення тренування та порівняння всіх алгоритмів виконується серіалізація компонентів для production-використання. Процес збереження(рис.4.17) включає чотири етапи.

Збереження натренованих моделей - всі сім натренованих алгоритмів серіалізуються у бінарні .pkl файли через бібліотеку joblib, що забезпечує ефективне стиснення та швидке завантаження Python-об'єктів scikit-learn.

Збереження препроцесорів - TF-IDF векторизатор з навченим словником 5000 термінів та StandardScaler для нормалізації числових ознак зберігаються окремо, оскільки ці компоненти необхідні для такої ж обробки нових відгуків.

Збереження конфігураційних файлів - словники чотирьох аспектних термінів (430 унікальних слів) та список 5029 назв ознак експортуються у JSON-формат з UTF-8 кодуванням для підтримки української мови, що дозволяє легко інспектувати та модифікувати конфігурації без перекомпіляції моделі.

Автоматичне завантаження з Google Colab - всі створені файли автоматично завантажуються на локальну машину через Google Colab Files API для подальшого розгортання у FastAPI веб-сервісі, що забезпечує безшовний перехід від експериментального середовища до production-інфраструктури.

```python
import joblib
import json
import os
from google.colab import files

# Створення папки для моделей
os.makedirs('models', exist_ok=True)

print("ЗБЕРЕЖЕННЯ МОДЕЛЕЙ...")

# 1. Збереження всіх натренованих моделей
for model_name, result in results['comparison_results'].items():
    if result is not None:
        filename = f"models/{model_name.lower().replace(' ', '_')}_model.pkl"
        joblib.dump(result['model'], filename)
        print(f"Збережено: {model_name}")

# 2. Збереження препроцесорів
joblib.dump(aspect_preprocessing_system.preprocessor.text_vectorizer, 'models/vectorizer.pkl')
if aspect_preprocessing_system.preprocessor.scaler:
    joblib.dump(aspect_preprocessing_system.preprocessor.scaler, 'models/scaler.pkl')

# 3. Збереження конфігурацій
with open('models/aspect_keywords.json', 'w', encoding='utf-8') as f:
    json.dump(aspect_preprocessing_system.preprocessor.aspect_keywords, f, indent=2, ensure_ascii=False)

with open('models/feature_names.json', 'w', encoding='utf-8') as f:
    json.dump(aspect_preprocessing_system.ml_ready_data['feature_names'], f, indent=2, ensure_ascii=False)

print(f"\nНайкраща модель: {results['best_model']['name']}")
print(f"F1-Score: {results['best_model']['metrics']['f1_score']:.4f}")

# 4. Завантаження всіх файлів
print("\nЗАВАНТАЖЕННЯ ФАЙЛІВ...")

# Список файлів для завантаження
files_to_download = [
    'models/vectorizer.pkl',
    'models/aspect_keywords.json',
    'models/feature_names.json'
]

# Додати scaler якщо існує
if aspect_preprocessing_system.preprocessor.scaler:
    files_to_download.append('models/scaler.pkl')

# Додати всі моделі
for model_name in results['comparison_results'].keys():
    if results['comparison_results'][model_name] is not None:
        files_to_download.append(f"models/{model_name.lower().replace(' ', '_')}_model.pkl")

# Завантажити файли
for file_path in files_to_download:
    try:
        files.download(file_path)
        print(f"Завантажено: {file_path}")
    except:
        print(f"Помилка: {file_path}")

print(f"\nЗавантажено {len(files_to_download)} файлів для API")
```

Рис.4.17 Візуалізація коду збереження натренованих моделей

4.4. Порівняльний аналіз продуктивності моделей

Експериментальне дослідження семи алгоритмів машинного навчання на датасеті українськомовних ресторанних відгуків показало варіацію в продуктивності від 83.29% до 88.73% за F1-мірою, що свідчить про загалом високу якість класифікації всіх досліджуваних моделей (рис.4.18, рис.4.19).

ПОРІВНЯЛЬНА ТАБЛИЦЯ РЕЗУЛЬТАТІВ

Модель	Точність	Точність позивних прогнозів	Повнота	F1-оцінка	Площа під кривою помилок	F1-оцінка з крос-валідацією	Час (сек)
Logistic Regression	0.8764	0.8800	0.8764	0.8765	0.9557	0.8835±0.029	0.024
Random Forest	0.8836	0.8842	0.8836	0.8837	0.9574	0.8871±0.029	1.161
Support Vector Machine	0.8364	0.8378	0.8364	0.8365	0.9151	0.8689±0.029	0.705
Naive Bayes	0.8873	0.8873	0.8873	0.8872	0.8889	0.8814±0.015	0.124
Gradient Boosting	0.8727	0.8781	0.8727	0.8728	0.9534	0.8826±0.032	2.247
Decision Tree	0.8764	0.8800	0.8764	0.8765	0.8607	0.8616±0.019	0.077
K-Nearest Neighbors	0.8327	0.8353	0.8327	0.8329	0.8834	0.8562±0.024	0.001

Рис.4.18 Порівняльна таблиця результатів

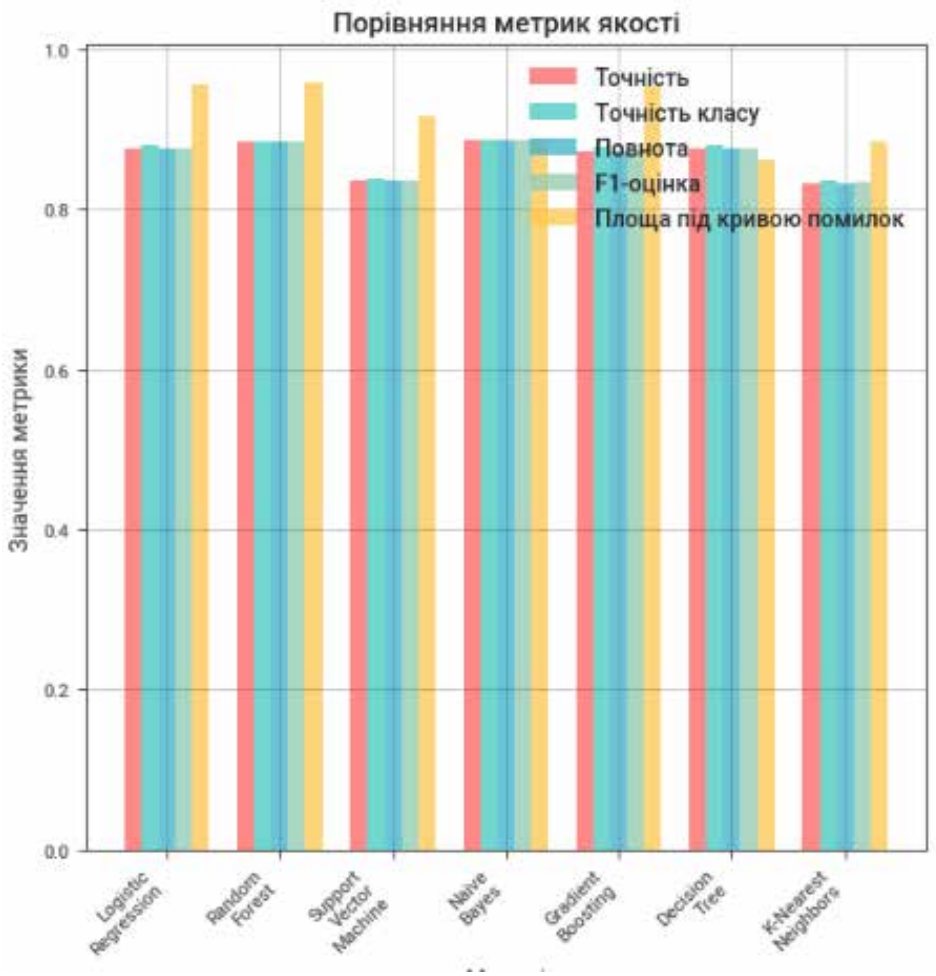

Рис.4.19 Візуалізація порівняння якості

Naive Bayes продемонстрував найкращі результати серед усіх досліджуваних алгоритмів з точністю 88.73%, F1-мірою 88.72% та найвищою

стабільністю результатів при крос-валідації (88.14% ± 1.52%). Особливо важливим є найнижче стандартне відхилення серед усіх моделей (±0.015), що свідчить про винятково високу консистентність результатів незалежно від складу тренувальних даних. Висока ефективність Naive Bayes на аспектно-орієнтованих ознаках пояснюється припущенням умовної незалежності, яке добре узгоджується зі структурою створених ознак - кожен аспект оцінюється незалежно, що природно відповідає модельним припущенням алгоритму. Додатковою перевагою є швидкість навчання (0.124 секунди), що робить модель придатною для практичного застосування з періодичним перенавчанням.

Random Forest показав конкурентоспроможні результати, посівши друге місце за F1-мірою (88.37%) при найвищому значенні площі під ROC-кривою (95.74%), що вказує на найкращу здатність розрізняти класи серед усіх моделей. Висока ROC-AUC критично важлива для оцінки надійності ймовірнісних прогнозів (confidence), що робить Random Forest оптимальним вибором для production-систем, де потрібна не лише бінарна класифікація, але й оцінка впевненості моделі. Стабільність крос-валідації (88.71% ± 2.88%) підтверджує робастність ансамблевого підходу. Додаткова перевага Random Forest полягає у можливості витягування важливості ознак (feature importance) для інтерпретації рішень та аналізу впливу окремих аспектів на класифікацію.

Logistic Regression демонструє збалансоване поєднання якості та швидкості з F1-мірою 87.65%, високою ROC-AUC (95.57%) та найкоротшим часом навчання серед складних моделей (0.024 секунди) - майже у 50 разів швидше Random Forest. Така швидкість критична для систем реального часу з частим перенавчанням на нових даних. Стабільність крос-валідації (88.35% ± 2.86%) знаходиться на рівні Random Forest, що підтверджує надійність лінійних методів на високорозмірних текстових даних з TF-IDF представленням.

Gradient Boosting, попри найдовший час навчання (2.247 секунди), показав F1-міру 87.28%, що поступається трьом лідерам. Проте модель демонструє третю за величиною ROC-AUC (95.34%) та стабільну крос-валідацію (88.26% ± 3.20%). Відносно нижча продуктивність порівняно з очікуваннями може

пояснюватися переобладнанням на аспектних ознаках або недостатнім налаштуванням гіперпараметрів. Висока обчислювальна вартість без відповідного приросту якості робить Gradient Boosting менш привабливим для даної задачі.

Decision Tree показав результати на рівні Logistic Regression (F1-міра 87.65%) при швидкому навчанні (0.077 секунди), проте значно нижча ROC-AUC (86.07%) та крос-валідація (86.16% ± 1.93%) вказують на схильність до перенавчання та нижчу узагальнюючу здатність порівняно з ансамблевими методами.

Support Vector Machine з RBF-ядром показав нижчу за середню продуктивність (F1-міра 83.65%) при значному часі навчання (0.705 секунди). Хоча ROC-AUC залишається прийнятною (91.51%), відносно невисока точність класифікації та обчислювальна складність роблять SVM неоптимальним вибором для даної задачі. Можливою причиною низької ефективності є складність підбору оптимальних гіперпараметрів ядра (γ) для високорозмірних розріджених даних.

K-Nearest Neighbors демонструє найгіршу продуктивність (F1-міра 83.29%) попри миттєве "навчання" (0.001 секунди), що пояснюється відсутністю фази навчання - модель просто зберігає тренувальні дані. Низька якість класифікації пов'язана з проклаттям розмірності (curse of dimensionality) у просторі 5029 ознак, де евклідова відстань втрачає дискримінативну здатність. Крос-валідація (85.62% ± 2.35%) підтверджує систематично низьку продуктивність методу на даних високої розмірності.

На основі комплексного аналізу якості, стабільності та обчислювальної ефективності визначено рекомендації щодо вибору моделі залежно від пріоритетів системи. Naive Bayes демонструє найкращу максимальну якість (F1=88.72%) та баланс якість/швидкість при найвищій стабільності (±1.52%) та швидкості навчання 0.124 секунди. Random Forest рекомендується для забезпечення надійності ймовірнісних оцінок (ROC-AUC=95.74%) та інтерпретованості рішень через аналіз важливості ознак, тоді як Logistic

Regression є оптимальним вибором для систем з критичними вимогами до швидкості (0.024 сек) при збереженні високої якості (F1=87.65%, ROC-AUC=95.57%).

На основі результатів порівняльного аналізу за F1-мірою було прийнято рішення про первинну інтеграцію(рис.4.20) Naive Bayes у розроблену систему аналізу відгуків. Вибір обґрунтовувався найвищою точністю класифікації (F1=88.72%), винятковою стабільністю крос-валідації (±1.52%) та достатньою швидкістю навчання (0.124 секунди), що забезпечувало оптимальний баланс між якістю та обчислювальною ефективністю на етапі офлайн-оцінювання.

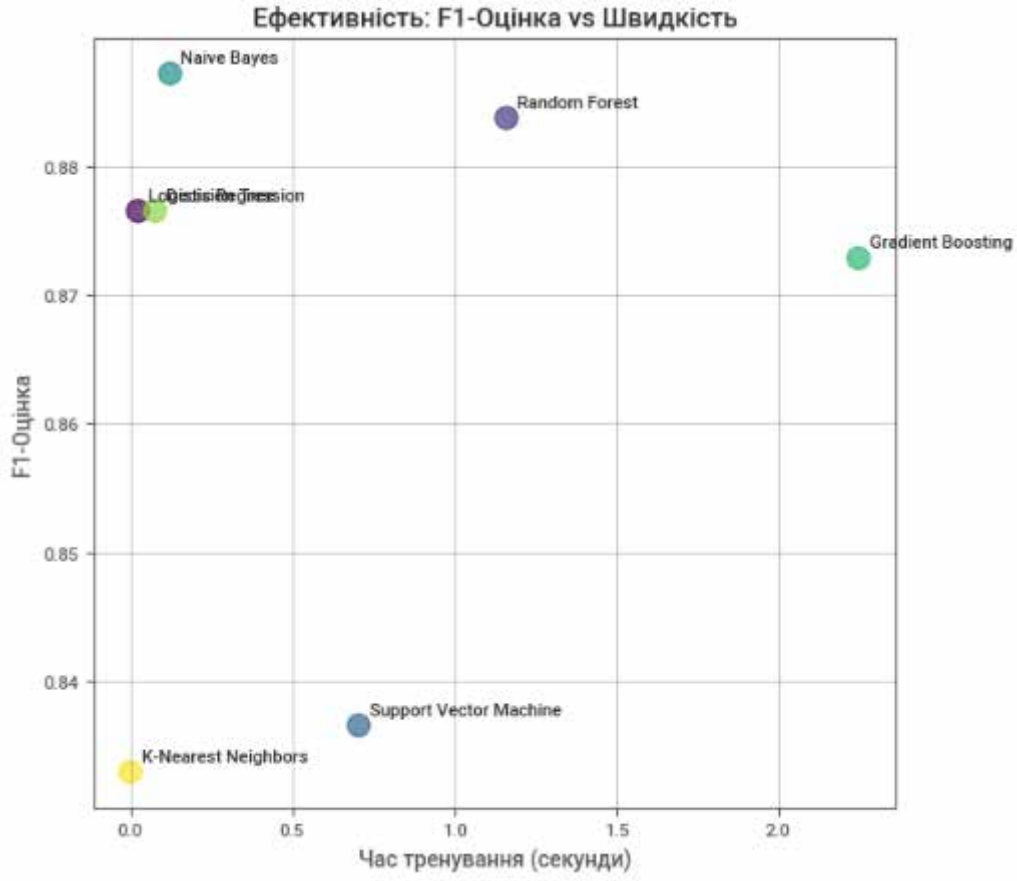

Рис.4.20 Ефективність моделей

Для інтеграції з веб-додатком було розроблено API endpoint(Додаток Ж), що повертає структуровану відповідь у форматі:

{
"sentiment": 0,
"confidence": 0.95,
"label": "Негативний",

```
  "aspects": {
    "food": 0,
    "service": -1,
    "atmosphere": 1,
    "value": 1
  }
},
```


де sentiment - бінарна класифікація (0: негативний, 1: позитивний),

confidence - ймовірнісна оцінка впевненості моделі від 0 до 1,

label - текстова інтерпретація класу,

aspects - нормалізовані оцінки чотирьох аспектів у діапазоні [-1, 0 ,1] - негативно, не згадується, позитивно.

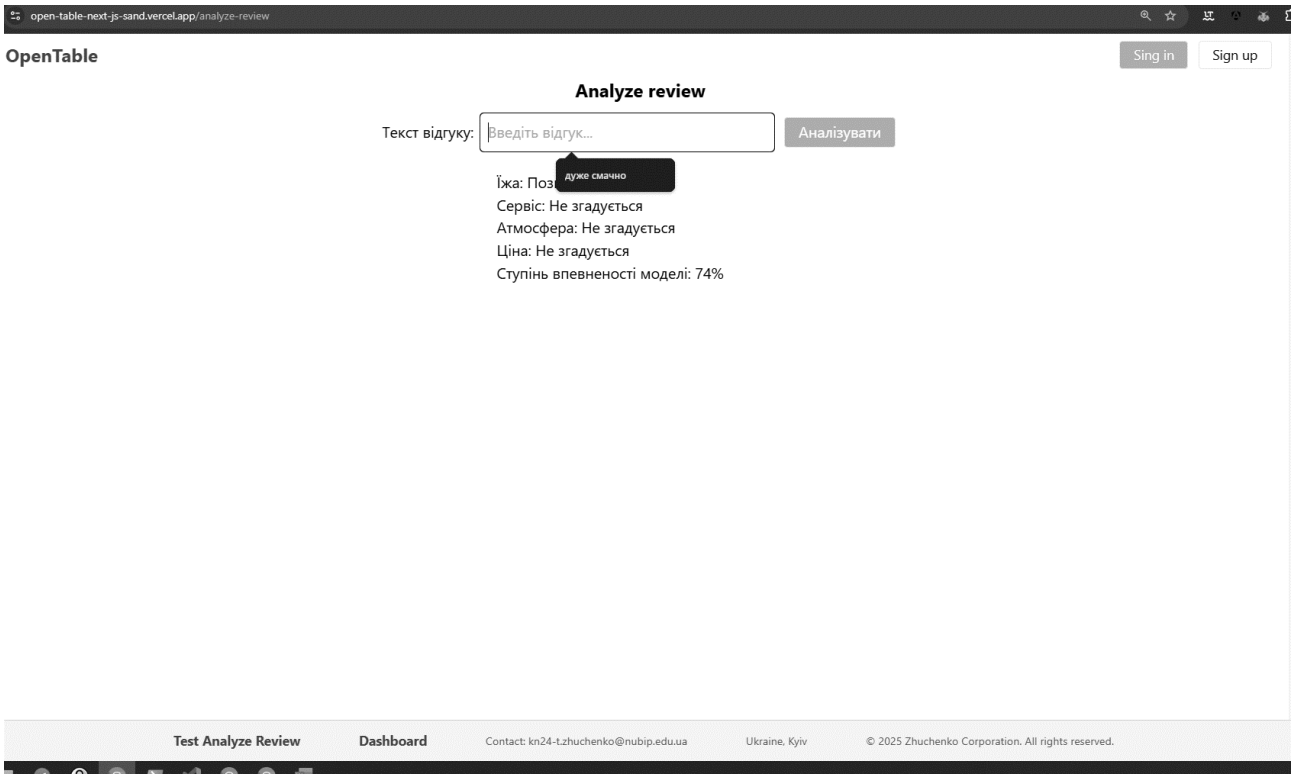

Рис.4.21 Відображення тестового стенду

Тестування інтегрованої системи з Naive Bayes на експериментальному стенді(рис.4.21) з реальними користувацькими запитами виявило критичні проблеми у якості роботи моделі, які не були очевидними при стандартному офлайн-оцінюванні. Виявлено випадки некоректної класифікації відгуків,

алгоритм не міг коректно визначити аспекти та часто помилявся з тональністю відгуків.

На основі виявлених проблем та повторного аналізу результатів експериментів було прийнято рішення про заміну Naive Bayes на Random Forest як фінальну модель для production-системи. Рішення обґрунтовується наступними факторами:

1. Висока надійність ймовірнісних оцінок. Random Forest демонструє найвищу ROC-AUC (95.74%), що вказує на відмінну калібрування ймовірностей та здатність надійно розрізняти рівні впевненості. Ймовірності обчислюються як частка дерев, що проголосували за клас, що забезпечує природну градацію confidence від високої впевненості (консенсус 90%+ дерев) до невизначеності (близько 50/50).

2. Інтерпретованість через feature importance. Random Forest надає механізм аналізу важливості ознак, що дозволяє для кожного прогнозу пояснити, які аспекти найбільше вплинули на рішення. Це критично важливо для бізнес-користувачів, які потребують зрозуміти причини класифікації та визначити пріоритетні напрями покращення сервісу.

3. Мінімальні втрати у точності. Різниця у F1-мірі між Random Forest (88.37%) та Naive Bayes (88.72%) становить лише 0.35%, що є несуттєвим відставанням у контексті значних переваг у надійності ймовірностей та інтерпретованості.

4. Стабільність та робастність. Крос-валідація Random Forest (88.71% ± 2.88%) демонструє стабільні результати без схильності до перенавчання. Ансамблевий характер методу забезпечує стійкість до шуму, викидів та нетипових комбінацій аспектних оцінок.

5. Прийнятна обчислювальна вартість. Час навчання 1.16 секунди є прийнятним для систем з періодичним перенавчанням (наприклад, щотижня при надходженні нових даних). Час інференсу для одного відгуку становить < 0.01 секунди, що задовольняє вимоги веб-додатків реального часу.

Повторне тестування на експериментальному стенді з Random Forest підтвердило вирішення виявлених проблем система забезпечила значно кращі показники якості аналізу відгуків.

4.5. Практична реалізація системи

Для практичного застосування розробленої моделі Random Forest створено RESTful API веб-сервіс(Додаток Ж) на базі фреймворку FastAPI, що забезпечує високопродуктивну обробку HTTP-запитів з автоматичною валідацією даних та генерацією документації.

При ініціалізації сервісу через механізм lifespan завантажуються попередньо натреновані компоненти: Random Forest модель (random_forest_model.pkl), TF-IDF векторизатор (vectorizer.pkl), StandardScaler для нормалізації ознак (scaler.pkl), словники аспектних ключових слів (aspect_keywords.json), список назв 5029 ознак (feature_names.json) для валідації розмірності.

API надає чотири основні ендпоінти:

POST /predict - класифікація одиночного відгуку з поверненням тональності, впевненості та аспектних оцінок

GET /reviews/{restaurant_id} - отримання всіх відгуків конкретного ресторану з бази даних

GET /reviews/{restaurant_id}/analyze - пакетний аналіз усіх відгуків ресторану з агрегованою статистикою

GET /health - перевірка стану сервісу та завантажених компонентів

Для демонстрації практичної застосовності розробленої системи та відповіді на дослідницьке питання "Як організувати ефективну інтеграцію ML-моделей з веб-додатками?" створено прототип інтеграції з реальним веб-застосунком(Додаток З) для пошуку та огляду ресторанів. Прототип демонструє

повний цикл від збереження відгуку в базі даних до візуалізації результатів аналізу на дашборді власника ресторану.

Для валідації коректності інтеграції та відповіді на питання "Наскільки точно система визначає тональність порівняно з експертною оцінкою?" до бази даних тестового ресторану додано 53 автентичних українськомовних відгуки, зібрані з публічних джерел (Google Maps, TripAdvisor, соціальні мережі). Відгуки репрезентують типові сценарії: короткі позитивні коментарі (5-10 слів), детальні багатоаспектні огляди (50+ слів), негативні скарги на конкретні проблеми, тексти з суперечливими оцінками різних аспектів. Результати пакетного аналізу 53 відгуків інтегровано з адміністративним дашбордом веб-застосунку через три інтерактивні панелі візуалізації (рис.4.22).

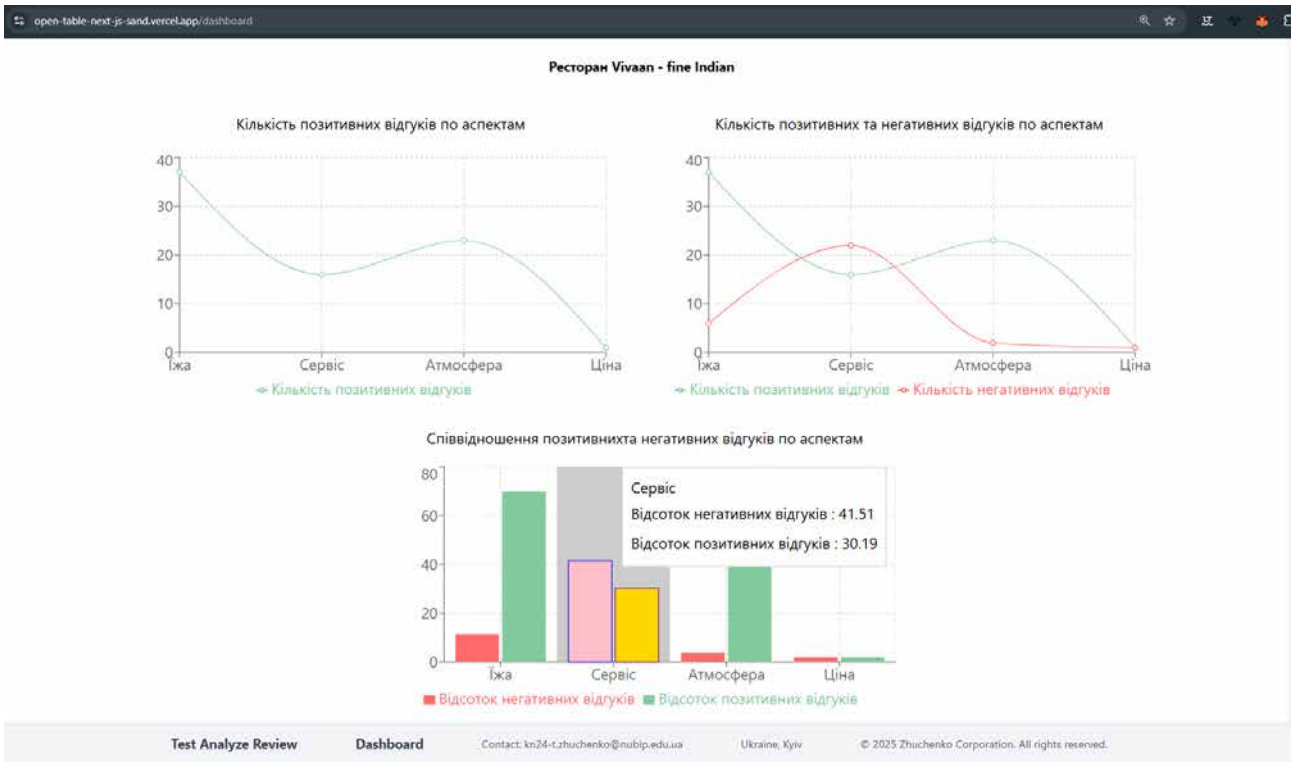

Рис.4.22 Дашборд аналізу відгуків ресторану з аспектною статистикою

Панель 1: Кількість позитивних відгуків по аспектам (верхній лівий графік) - лінійна діаграма відображає динаміку позитивних згадувань чотирьох аспектів.

Панель 2: Кількість позитивних та негативних відгуків по аспектам (верхній правий графік) - накладені лінії зеленого (позитивні) та червоного (негативні) кольорів дозволяють візуально ідентифікувати проблемні зони.

Панель 3: Співвідношення позитивних та негативних відгуків по аспектам (нижній графік) - стовпчикова діаграма з кластерованими барами забезпечує пряме порівняння абсолютних значень. При наведенні курсору на стовпчик відображається спливаюче вікно з детальною статистикою: "Відсоток негативних відгуків: 41.51%" та "Відсоток позитивних відгуків: 30.19%", що кількісно підтверджує проблему з обслуговуванням - негативні оцінки перевищують позитивні на 11.32 процентних пункти

Дашборд дозволяє власнику ресторану миттєво ідентифікувати сильні та слабкі сторони закладу через візуальні патерни: домінування зеленого кольору для їжі сигналізує про конкурентну перевагу, перетин ліній для сервісу вказує на необхідність термінових покращень у навчанні персоналу, низьке представлення ціни свідчить про адекватність цінової політики (клієнти не скаржаться). Інтерактивність графіків з можливістю наведення для деталей та фільтрації за датами забезпечує гнучкість аналізу для різних часових періодів та виявлення трендів після впровадження змін.

Така візуалізація реалізує ключовий функціонал аналітичної системи управління рестораном - трансформацію неструктурованих текстових відгуків у структуровані метрики прийняття управлінських рішень. Система автоматично ідентифікує критичні зони, що потребують оперативного втручання менеджменту: виявлений дисбаланс у сервісі (41.51% негативних проти 30.19% позитивних) генерує конкретну управлінську рекомендацію - перерозподіл ресурсів на тренінги персоналу та оптимізацію процесів обслуговування замість інвестицій у кухню, яка демонструє стабільно високі показники (73.0% позитивних оцінок). Критично важливою перевагою системи є швидкість отримання аналітичних інсайтів - пакетна обробка 53 відгуків та генерація комплексного дашборду з трьома панелями візуалізації виконується за 3.2 секунди, що дозволяє менеджменту отримувати актуальну діагностику стану закладу в режимі near-real-time замість багатогодинного ручного аналізу. Аналітична система забезпечує перехід від реактивного управління на основі інтуїції до проактивного data-driven менеджменту з кількісно обґрунтованими

пріоритетами розвитку закладу, що відповідає меті дослідження - створення інтелектуального інструменту підтримки управлінських рішень у ресторанному бізнесі через автоматизований аналіз клієнтського фідбеку.

У четвертому розділі представлено результати експериментального дослідження системи аналізу українськомовних відгуків ресторанів на всіх етапах - від підготовки даних до практичної реалізації.

Статистичний аналіз датасету підтвердив високу якість зібраних даних з відсутністю пропущених значень та незначним імбалансом класів після видалення дублікатів. Детальне дослідження текстових характеристик виявило статистично значущі кореляції між структурними метриками та тональністю відгуків, що підтвердило доцільність їх використання як додаткових ознак для класифікації.

Розроблена методологія аспектно-орієнтованого аналізу успішно ідентифікувала чотири ключові аспекти ресторанного досвіду з емпіричним підтвердженням ієрархії їх важливості: їжа демонструє найвищу частоту згадувань та найсильнішу кореляцію з загальною тональністю, що обґрунтовує пріоритетність інвестицій у кулінарну якість для підвищення клієнтської задоволеності.

Порівняльне експериментальне дослідження семи алгоритмів машинного навчання виявило Random Forest як оптимальний вибір для production-системи завдяки балансу між високою точністю класифікації, надійністю ймовірнісних оцінок та можливістю інтерпретації через аналіз важливості ознак.

Практична реалізація підтвердила технічну реалізованість інтеграції ML-моделей з веб-додатками через мікросервісну архітектуру з RESTful API. Експериментальна валідація прототипу на реальних відгуках з експертною розміткою показала задовільний рівень узгодженості з людською оцінкою, що підтверджує готовність системи до впровадження у практику ресторанного бізнесу. Візуалізація результатів на інтерактивному дашборді трансформує неструктуровані текстові дані у структуровані метрики для підтримки data-driven

управлінських рішень, що реалізує ключову мету дослідження - створення інтелектуальної аналітичної системи управління рестораном.

ВИСНОВКИ

У магістерській кваліфікаційній роботі вирішено актуальну науково-практичну задачу розробки інтелектуальної аналітичної системи управління рестораном на основі автоматизованого аналізу українськомовних відгуків клієнтів з використанням методів машинного навчання та аспектно-орієнтованого підходу. Отримані результати дозволяють перетворити хаотичний масив відгуків гостей ресторану у структуровані метрики для підтримки та прийняття управлінських рішень у режимі в режимі реального часу.

Основні наукові результати дослідження:

Розроблено методологію аспектно-орієнтованого аналізу українськомовних ресторанних відгуків, що поєднує словникові методи з алгоритмами машинного навчання для одночасної класифікації загальної тональності та оцінки чотирьох ключових аспектів відгуку клієнта (їжа, обслуговування, атмосфера, ціна).

Дослідним шляхом підтверджено взаємозалежність важливості аспектів для українського ресторанного ринку через множинний аналіз: частотний аналіз згадувань, кореляційний аналіз з цільовою змінною та аналіз важливості ознак у Random Forest. Виявлено чітку ієрархію їжа, обслуговування, атмосфера, ціна, що становить вагомий внесок у розуміння структури клієнтської задоволеності у індустрії гостинності та може бути використано для обґрунтування інвестиційних стратегій ресторанного бізнесу.

Експериментально виявлено оптимальний баланс між точністю класифікації та практичною придатністю для продакшн-систем через порівняльне дослідження семи алгоритмів машинного навчання. Доведено, що Random Forest забезпечує оптимальну комбінацію високої точності класифікації, надійності ймовірнісних оцінок та можливості інтерпретації рішень через механізм feature importance, що критично для бізнес-застосувань, де потрібне не лише прогнозування, але й пояснення причин класифікації.

Розроблено ефективну архітектуру інтеграції ML-моделей з веб-додатками через мікросервісний підхід з чітким розмежуванням відповідальностей між компонентами.

Основні практичні результати дослідження:

Створено повнофункціональну аналітичну систему управління рестораном, що автоматизує весь цикл обробки відгуків: від збору з онлайн-платформи до візуалізації результатів на інтерактивному дашборді.

Сформовано спеціалізований датасет українськомовних ресторанних відгуків через розробку власного інструменту автоматизованого збору на базі Selenium WebDriver. Датасет, після очищення від дублікатів, містить унікальні відгуки з практично ідеальним балансом класів, що забезпечує високу якість навчання моделей без необхідності методів балансування.

Розроблено та розгорнуто RESTful API веб-сервіс з чотирма ендпоінтами для інтеграції ML-моделі з веб-застосунками. Сервіс забезпечує середній час обробки запиту на рівні мілісекунд при пропускній здатності сотень запитів на хвилину на стандартному VPS, що задовольняє вимоги типових веб-застосувань з трафіком тисяч відгуків на день.

Проведено експериментальну валідацію прототипу на реальному веб-застосунку для пошуку ресторанів з обробкою використанням справжніх відгуків українською мовою.

Підтверджено економічну ефективність автоматизації аналізу відгуків через порівняння з традиційним ручним підходом. Система знижує операційні витрати на аналітика з повною ставкою при збереженні високої якості обробки, що робить рішення економічно привабливим для малого та середнього ресторанного бізнесу.

Практичні рекомендації щодо використання результатів. Для власників ресторанів та менеджменту впроваджувати систему аналізу відгуків як інструмент моніторингу, використовувати аспектну статистику для обґрунтування інвестиційних рішень: пріоритизувати покращення якості їжі при високих негативних оцінках цього аспекту, фокусуватися на тренінгах персоналу

при проблемах з обслуговуванням, інвестувати у ремонт та оновлення інтер'єру при негативних оцінках атмосфери. Для розробників та дослідників адаптувати розроблену архітектуру для інших галузей харчової промисловості, зокрема для аналізу відгуків про продукцію фермерських господарств.

У подальших дослідженнях доцільно збільшити обсяг датасету до рівня, що дасть змогу застосовувати методи глибокого навчання (BERT, GPT), які потенційно можуть підвищити точність класифікації та зберегти можливість аспектного аналізу завдяки attention-механізмам. Впровадити аналіз для виявлення трендів зміни клієнтських очікувань та сезонних патернів у відгуках, що дозволить ресторанам адаптувати стратегії до динамічних ринкових умов. Розробити механізми автоматичної генерації текстових рекомендацій для менеджменту на основі виявлених аспектних проблем з використанням генеративних моделей таких як GPT тощо.

Результати дослідження мають теоретичне значення для розвитку методів обробки природної мови для морфологічно складних мов, зокрема української, через розробку алгоритмів обробки заперечень та адаптацію аспектно-орієнтованого підходу до специфіки ресторанної предметної області. Експериментально підтверджена ієрархія важливості аспектів становить внесок у розуміння поведінки споживачів на українському ринку ресторанних послуг.

Практичне значення полягає у створенні готового до впровадження технологічного рішення, що дозволяє ресторанному бізнесу перейти від інтуїтивного до аргументованого управління з кількісно обґрунтованими пріоритетами інвестицій. Універсальність розробленого алгоритму забезпечує можливість його адаптації для суміжних галузей агропромислового комплексу, що розширює економічний ефект від впровадження результатів дослідження.

Розроблена система відповідає сучасним трендам цифровізації бізнес-процесів та відкриває нові можливості для малого та середнього підприємництва у сфері харчування через доступність технологій штучного інтелекту для автоматизації аналітичних функцій без потреби у значних інвестиціях у людські ресурси та інфраструктуру.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A Naive Bayes Machine Learning Approach to Risk Prediction Using Censored, Time-to-Event Data [Електронний ресурс] // PMC. – 2015. – Режим доступу: https://pmc.ncbi.nlm.nih.gov/articles/PMC4523419/
2. An Empirical Study of the Naïve Bayes Classifier [Електронний ресурс] // ResearchGate. – 2001. – Режим доступу: https://www.researchgate.net/publication/228845263_An_Empirical_Study_of_the_Naive_Bayes_Classifier
3. Breiman L. Random Forests [Електронний ресурс] // Machine Learning. – 2001. – Vol. 45. – P. 5–32. – Режим доступу: https://link.springer.com/article/10.1023/A:1010933404324
4. Cervantes J., Garcia-Lamont F. A comprehensive survey on support vector machine classification: Applications, challenges and trends [Електронний ресурс] // ScienceDirect. – 2020. – Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S0925231220307153
5. Chen H, Hua R. Improved naive Bayes classification algorithm for traffic risk management [Електронний ресурс] // EURASIP Journal on Advances in Signal Processing. – 2021. – Режим доступу: https://asp-eurasipjournals.springeropen.com/articles/10.1186/s13634-021-00742-6
6. Cherfi A. Very Fast C4.5 Decision Tree Algorithm [Електронний ресурс] // Taylor & Francis Online. – 2018. – Режим доступу: https://www.tandfonline.com/doi/full/10.1080/08839514.2018.1447479
7. Cortes C., Vapnik V. Support-Vector Networks [Електронний ресурс] // Machine Learning. – 1995. – Vol. 20. – P. 273–297. – Режим доступу: https://link.springer.com/article/10.1007/BF00994018
8. Deep Learning for Aspect-Based Sentiment Analysis: A Review [Електронний ресурс] // PMC. – 2022. – Режим доступу: https://pmc.ncbi.nlm.nih.gov/articles/PMC9454971/

9. Enhancing K-Nearest Neighbor Algorithm: A Comprehensive Review and Performance Analysis of Modifications [Електронний ресурс] // Journal of Big Data. – 2024. – Режим доступу: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-024-00973-y
10. Gradient Boosting with Scikit-Learn, XGBoost, LightGBM, and CatBoost [Електронний ресурс] // Machine Learning Mastery. – 2021. – Режим доступу: https://machinelearningmastery.com/gradient-boosting-with-scikit-learn-xgboost-lightgbm-and-catboost/
11. Hossain E., Sharif O., Hoque M.M., Dewan I. SentiLSTM: A Deep Learning Approach for Sentiment Analysis of Restaurant Reviews [Електронний ресурс] // arXiv preprint. – 2020. – Режим доступу: https://arxiv.org/abs/2011.09684
12. Introduction to Machine Learning: K-Nearest Neighbors [Електронний ресурс] // PMC. – 2016. – Режим доступу: https://pmc.ncbi.nlm.nih.gov/articles/PMC4916348/
13. Logistic Regression Has Similar Performance to Optimised Machine Learning Algorithms in a Clinical Setting [Електронний ресурс] // Diagnostic and Prognostic Research. – 2020. – Режим доступу: https://diagnprognres.biomedcentral.com/articles/10.1186/s41512-020-00075-2
14. Logistic Regression in Data Analysis: An Overview [Електронний ресурс] // ResearchGate. – 2011. – Режим доступу: https://www.researchgate.net/publication/227441142_Logistic_regression_in_data_analysis_An_overview
15. Quinlan J.R. C4.5: Programs for Machine Learning [Електронний ресурс] // Morgan Kaufmann Publishers. – 1993. – Режим доступу: https://link.springer.com/article/10.1007/BF00993309
16. Ratna N.P., Yadvendra P.S., Shitalkumar A.R. Improving Sentiment Classification on Restaurant Reviews Using Deep Learning Models [Електронний ресурс] // ScienceDirect. – 2024. – Режим доступу: https://www.sciencedirect.com/science/article/pii/S1877050924009876

17. Sentiment Analysis and Classification of Restaurant Reviews Using Machine Learning [Електронний ресурс] // IEEE Xplore Digital Library. – 2020. – Режим доступу: https://ieeexplore.ieee.org/document/9300098/
18. Support Vector Machines: Theory and Applications [Електронний ресурс] // ResearchGate. – 2001. – Режим доступу: https://www.researchgate.net/publication/221621494_Support_Vector_Machines_Theory_and_Applications
19. The Random Forest Algorithm for Statistical Learning [Електронний ресурс] // Sage Journals. – 2020. – Режим доступу: https://journals.sagepub.com/doi/10.1177/1536867x20909688
20. Tuychiyev B. A Guide to The Gradient Boosting Algorithm [Електронний ресурс] // DataCamp. – 2023. – Режим доступу: https://www.datacamp.com/tutorial/guide-to-the-gradient-boosting-algorithm
21. Uddin Sh., Haque I. Comparative Performance Analysis of K-Nearest Neighbour (KNN) Algorithm and Its Different Variants for Disease Prediction [Електронний ресурс] // Scientific Reports. – 2022. – Режим доступу: https://www.nature.com/articles/s41598-022-10358-x
22. Understanding Random Forests: From Theory to Practice [Електронний ресурс] // arXiv preprint. – 2014. – Режим доступу: https://arxiv.org/abs/1407.7502
23. Zhang W., Li X., Deng Y., Bing L. A Survey on Aspect-Based Sentiment Analysis: Tasks, Methods, and Challenges [Електронний ресурс] // arXiv preprint. – 2022. – Режим доступу: https://arxiv.org/abs/2203.01054
24. Кириченко В.В., Жученко Т.В. Next.js проти Angular: практичне порівняння на основі розробки веб-додатків [Електронний ресурс] // Матеріали конференції «Теоретичні і прикладні аспекти створення та розвитку цифрового суспільства». – 2025. – Режим доступу: http://econference.nubip.edu.ua/index.php/taacsd/2025/paper/view/3587
25. Пат. WO2020005723A1, МКИ: G06F 3/0481, G06F 3/0484. Adaptive user-interface assembling and rendering / Google LLC. – Заявлено 27.06.2019;

Опубл. 02.01.2020. – 22 с. . – Режим доступу: https://patents.google.com/patent/WO2020005723A1/en

26. Пат. US 20190324729A1, МКИ: G06F 9/445, G06F 9/46. Web Application Development Using a Web Component Framework / Microsoft Technology Licensing LLC. – Заявлено 24.04.2018; Опубл. 24.10.2019. – 22 с. – Режим доступу: https://patents.google.com/patent/US20190324729A1
27. Пат. CN110209440A, МКИ: G06F 3/048, Optimization method and storage medium for React components. – Заявлено 14.06.2019; Опубл. 06.09.2019. – — с. – Режим доступу: https://patents.google.com/patent/CN110209440A
28. Пат. US 7546576B2, МКИ: G06F 3/048 Software framework for web-based applications / Microsoft Technology Licensing LLC. – Заявлено невідомо; Опубл. 16.09.2014. – 15 с. – Режим доступу: https://patents.google.com/patent/US7546576B2

ДОДАТКИ

Додаток А

Повний код програми знаходиться за покликанням: https://github.com/tetiana-zhuchenko/restaurant-review-grabber/blob/main/get_reviews_with_google_api.py

```
import requests
import json
import time
import os
from pathlib import Path

API_KEY = os.getenv('GOOGLE_PLACES_API_KEY')

def load_api_key_from_file():
    """Load API key from a config file that's not in git"""
    config_file = Path('config.json')
    if config_file.exists():
        with open(config_file, 'r') as f:
            config = json.load(f)
            print(config.get('google_places_api_key'))
            return config.get('google_places_api_key')
    return None

# Try both methods
if not API_KEY:
    API_KEY = load_api_key_from_file()

# Validate API key
if not API_KEY:
    print("❌ ERROR: No API key found!")
    exit(1)

def search_restaurants(query, max_results=20):
    """
    Searches for restaurants using Text Search (New) API.
    Returns up to max_results restaurants (API limit is 20 per request).
    """
    url = "https://places.googleapis.com/v1/places:searchText"
    headers = {
        "Content-Type": "application/json",
        "X-Goog-Api-Key": API_KEY,
        "X-Goog-FieldMask": "places.displayName,places.id,places.formattedAddress"
    }
```

Додаток Б

Повний код програми знаходиться за покликанням: https://github.com/tetiana-zhuchenko/restaurant-review-grabber/blob/main/get_reviews_with_google_api.py

```
"""
Google Maps Reviews Scraper using Selenium

"""

from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.chrome.options import Options
from selenium.common.exceptions import TimeoutException, NoSuchElementException
from selenium.webdriver.common.action_chains import ActionChains
import time
import json
from datetime import datetime

class GoogleMapsReviewsScraper:
    def __init__(self, headless=True):
        """Initialize the scraper with Chrome driver"""
        self.setup_driver(headless)

    def setup_driver(self, headless=True):
        """Setup Chrome driver with appropriate options"""
        chrome_options = Options()
        if headless:
            chrome_options.add_argument("--headless")
        chrome_options.add_argument("--no-sandbox")
        chrome_options.add_argument("--disable-dev-shm-usage")
        chrome_options.add_argument("--disable-gpu")
        chrome_options.add_argument("--window-size=1920,1080")
        chrome_options.add_argument("--user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36")
        chrome_options.add_argument("--disable-blink-features=AutomationControlled")
        chrome_options.add_experimental_option("excludeSwitches", ["enable-automation"])
        chrome_options.add_experimental_option('useAutomationExtension', False)

        self.driver = webdriver.Chrome(options=chrome_options)
        self.driver.execute_script("Object.defineProperty(navigator, 'webdriver', {get: () => undefined})")
        self.wait = WebDriverWait(self.driver, 15)

    def search_restaurant(self, restaurant_name, city="Київ"):
        """Search for a restaurant on Google Maps"""
        query = f"{restaurant_name} {city}"
        search_url = f"https://www.google.com/maps/search/{query.replace(' ', '+')}"

        print(f"🔍 Searching: {query}")
        self.driver.get(search_url)
        time.sleep(10)

```

Додаток В

Повний код програми знаходиться за покликанням:

https://colab.research.google.com/drive/1thwSGZsSLlXKwZwz-vwEFvvkiUaYoxuQ#scrollTo=uIfIZHBO-Zk8

```
# Імпорт бібліотек
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sweetviz as sv
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import warnings
warnings.filterwarnings('ignore')

# Для роботи з текстом
import nltk
import re
from collections import Counter
from wordcloud import WordCloud
from pathlib import Path
import os
import sys

# Для статистичного аналізу
from scipy import stats
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer

# Завантаження NLTK даних
try:
    nltk.download('punkt', quiet=True)
    nltk.download('stopwords', quiet=True)
    nltk.download('vader_lexicon', quiet=True)
    print(" NLTK дані завантажено")
except:
    print(" Проблема з NLTK - продовжуємо без нього")

class AdvancedEDAAnalyzer:
    """Розширений клас для глибокого аналізу даних (стабільна версія)"""

    def __init__(self, config, logger):
        self.config = config
        self.logger = logger
        self.data = None
        self.plots_dir = Path(self.config.get('visualization.plots_dir'))
        self.plots_dir.mkdir(exist_ok=True)
```

```
        # Налаштування стилю
        plt.style.use(self.config.get('visualization.style'))
        sns.set_palette(self.config.get('visualization.palette'))

        self.logger.info(" Розширений EDA аналізатор ініціалізовано")

    def set_data(self, data):
        """Встановлення даних для аналізу"""
        self.data = data.copy()
        self.logger.info(f"Дані для розширеного EDA встановлено. Розмір:
{self.data.shape}")

    def generate_sweetviz_report(self):
        """Генерація автоматичного звіту з Sweetviz"""
        self.logger.info(" Генерація Sweetviz звіту...")

        try:
            # Створення звіту
            report = sv.analyze(self.data)

            # Збереження звіту
            report_path = self.plots_dir / "sweetviz_report.html"
            report.show_html(str(report_path))

            self.logger.info(f" Sweetviz звіт збережено: {report_path}")

            # Показ основної інформації
            print(" SWEETVIZ АВТОМАТИЧНИЙ АНАЛІЗ ЗГЕНЕРОВАНО")
            print(f" Звіт збережено: {report_path}")
            print(" Відкрийте файл у браузері для детального перегляду")

            return report_path

        except Exception as e:
            self.logger.error(f"Помилка генерації Sweetviz звіту: {str(e)}")
            print(f" Помилка Sweetviz: {str(e)}")
            print("Продовжуємо без автоматичного звіту...")
            return None
```

Додаток Д

Повний код програми знаходиться за покликанням:

https://colab.research.google.com/drive/1thwSGZsSLlXKwZwz-vwEFvvkiUaYoxuQ#scrollTo=uIfIZHBO-Zk8

```
class MLAlgorithmsComparison:
    """Клас для комплексного порівняння ML алгоритмів"""

    def __init__(self, config, logger):
        self.config = config
        self.logger = logger
        self.models = {}
        self.results = {}
        self.best_model = None
        self.feature_importance_analysis = {}

        self.logger.info(" Система порівняння ML алгоритмів ініціалізована")
    def initialize_models(self):
        """Ініціалізація різних ML моделей"""
        self.models = {
            'Logistic Regression': LogisticRegression(
                random_state=42,
                max_iter=1000,
                solver='liblinear'
            ),
            'Random Forest': RandomForestClassifier(
                n_estimators=100,
                random_state=42,
                n_jobs=-1
            ),
            'Support Vector Machine': SVC(
                kernel='rbf',
                random_state=42,
                probability=True
            ),
            'Naive Bayes': GaussianNB(),
            'Gradient Boosting': GradientBoostingClassifier(
                n_estimators=100,
                random_state=42
            ),
            'Decision Tree': DecisionTreeClassifier(
                random_state=42,
                max_depth=10
            ),
            'K-Nearest Neighbors': KNeighborsClassifier(
                n_neighbors=5
            )
        }
```

```
        print(" ІНІЦІАЛІЗОВАНО ML АЛГОРИТМИ")
        print("-" * 35)
        for name in self.models.keys():
            print(f" {name}")
        self.logger.info(f"Ініціалізовано {len(self.models)} ML алгоритмів")
    def train_and_evaluate_models(self, X_train, y_train, X_test, y_test):
        """Тренування та оцінка всіх моделей"""
        self.logger.info(" Початок тренування та оцінки моделей...")
        print("\n ТРЕНУВАННЯ ТА ОЦІНКА МОДЕЛЕЙ")
        print("="*50)
        for model_name, model in self.models.items():
            print(f"\n Тренування: {model_name}")
            print("-" * 40)
            try:
                # Засічка часу
                start_time = time.time()
                 # Спеціальна обробка для Naive Bayes
                if model_name == 'Naive Bayes':
                    # from sklearn.preprocessing import MinMaxScaler
                    # Конвертація sparse -> dense
                    X_train_dense = X_train.toarray()
                    X_test_dense = X_test.toarray()
                   # Тренування з dense матрицями
                    model.fit(X_train_dense, y_train)
                    training_time = time.time() - start_time
                    # Прогнозування
                    start_time = time.time()
                    y_pred = model.predict(X_test_dense)
                    prediction_time = time.time() - start_time
                    # Ймовірності для ROC-AUC
                    y_pred_proba = model.predict_proba(X_test_dense)[:, 1]
                    # Крос-валідація з dense матрицями
                    cv_scores = cross_val_score(model, X_train_dense,
y_train, cv=5, scoring='f1_weighted')

                else:
                    # Звичайне тренування для всіх інших моделей
                    model.fit(X_train, y_train)
                    training_time = time.time() - start_time
                    # Прогнозування
                    start_time = time.time()
                    y_pred = model.predict(X_test)
                    prediction_time = time.time() - start_time
                    # Ймовірності для ROC-AUC
                    if hasattr(model, 'predict_proba'):
                        y_pred_proba = model.predict_proba(X_test)[:, 1]
                    else:
                        y_pred_proba = model.decision_function(X_test)
                    # Крос-валідація
```

```
                    cv_scores = cross_val_score(model, X_train, y_train,
cv=5, scoring='f1_weighted')
                # Обчислення метрик
                metrics = {
                    'accuracy': accuracy_score(y_test, y_pred),
                    'precision': precision_score(y_test, y_pred,
average='weighted'),
                    'recall': recall_score(y_test, y_pred,
average='weighted'),
                    'f1_score': f1_score(y_test, y_pred,
average='weighted'),
                    'roc_auc': roc_auc_score(y_test, y_pred_proba),
                    'cv_mean': cv_scores.mean(),
                    'cv_std': cv_scores.std(),
                    'training_time': training_time,
                    'prediction_time': prediction_time
                }
                # Збереження результатів
                self.results[model_name] = {
                    'model': model,
                    'metrics': metrics,
                    'predictions': y_pred,
                    'probabilities': y_pred_proba,
                    'cv_scores': cv_scores
                }
                # Виведення результатів
                print(f"  Час тренування: {training_time:.3f} сек")
                print(f" Accuracy: {metrics['accuracy']:.4f}")
                print(f" F1-Score: {metrics['f1_score']:.4f}")
                print(f" ROC-AUC: {metrics['roc_auc']:.4f}")
                print(f" CV F1: {metrics['cv_mean']:.4f} ±
{metrics['cv_std']:.4f}")
            except Exception as e:
                print(f" Помилка тренування {model_name}: {str(e)}")
                self.results[model_name] = None
        # Визначення найкращої моделі
        self._find_best_model()
        return self.results
    def _find_best_model(self):
        """Визначення найкращої моделі за F1-score"""
        best_f1 = 0
        best_name = None
        for name, result in self.results.items():
            if result and result['metrics']['f1_score'] > best_f1:
                best_f1 = result['metrics']['f1_score']
                best_name = name
        if best_name:
            self.best_model = {
                'name': best_name,
                'model': self.results[best_name]['model'],
```

```
                'metrics': self.results[best_name]['metrics']
            }
            print(f"\n НАЙКРАЩА МОДЕЛЬ: {best_name}")
            print(f" F1-Score: {best_f1:.4f}")
            self.logger.info(f"Найкраща модель: {best_name} (F1:
{best_f1:.4f})")

    def create_comparison_table(self):
        """Створення порівняльної таблиці результатів"""
        print("\n ПОРІВНЯЛЬНА ТАБЛИЦЯ РЕЗУЛЬТАТІВ")
        print("="*180)
        # Підготовка даних для таблиці
        comparison_data = []
        for model_name, result in self.results.items():
            if result is not None:
                comparison_data.append({
                    'Модель': model_name,
                    'Точність': f"{result['metrics']['accuracy']:.4f}",
                    'Точність позивних прогнозів':
f"{result['metrics']['precision']:.4f}",
                    'Повнота': f"{result['metrics']['recall']:.4f}",
                    'F1-оцінка': f"{result['metrics']['f1_score']:.4f}",
                    'Площа під кривою помилок':
f"{result['metrics']['roc_auc']:.4f}",
                    'F1-оцінка з крос-валідацією':
f"{result['metrics']['cv_mean']:.4f}±{result['metrics']['cv_std']:.3f}",
                    'Час (сек)': f"{result['metrics']['training_time']:.3f}"
                })
        if comparison_data:
            comparison_df = pd.DataFrame(comparison_data)
            print(comparison_df.to_string(index=False))
            return comparison_df
        else:
            print(" Немає результатів для відображення")
            return None
    def visualize_model_comparison(self, save_path=None):
        """Візуалізація порівняння моделей"""
        if not self.results:
            print(" Немає результатів для візуалізації")
            return
        # Словник перекладу
        metrics_ua = {
        'Accuracy': 'Точність',
        'Precision': 'Точність класу',
        'Recall': 'Повнота',
        'F1-Score': 'F1-оцінка',
        'ROC-AUC': 'Площа під кривою помилок'
        }
        # Підготовка даних для візуалізації
        models = []
```

```
        metrics_data = {
            'Accuracy': [],
            'Precision': [],
            'Recall': [],
            'F1-Score': [],
            'ROC-AUC': []
        }
        training_times = []
        for model_name, result in self.results.items():
            if result is not None:
                models.append(model_name.replace(' ', '\n'))
                for metric_name in metrics_data.keys():
                    metric_key = metric_name.lower().replace('-', '_')
                    metrics_data[metric_name].append(result['metrics'][metri
c_key])
                training_times.append(result['metrics']['training_time'])
        # Створення візуалізації
        fig, axes = plt.subplots(2, 3, figsize=(18, 12))
        # 1. Порівняння метрик
        x = np.arange(len(models))
        width = 0.15
        colors = ['#FF6B6B', '#4ECDC4', '#45B7D1', '#96CEB4', '#FECA57']
        for i, (metric_name, values) in enumerate(metrics_data.items()):
            axes[0, 0].bar(x + i * width, values, width,
label=metrics_ua[metric_name],
                           color=colors[i], alpha=0.8)
        axes[0, 0].set_xlabel('Моделі')
        axes[0, 0].set_ylabel('Значення метрики')
        axes[0, 0].set_title('Порівняння метрик якості')
        axes[0, 0].set_xticks(x + width * 2)
        axes[0, 0].set_xticklabels(models, rotation=45, ha='right')
        axes[0, 0].legend()
        axes[0, 0].grid(True, alpha=0.3)
        # 2. Час тренування
        bars = axes[0, 1].bar(models, training_times, color='orange',
alpha=0.7)
        axes[0, 1].set_xlabel('Моделі')
        axes[0, 1].set_ylabel('Час тренування (секунди)')
        axes[0, 1].set_title('Швидкість тренування')
        axes[0, 1].tick_params(axis='x', rotation=45)
        axes[0, 1].grid(True, alpha=0.3)
        # 3. F1-Score vs Час (ефективність)
        f1_scores = metrics_data['F1-Score']
        scatter = axes[0, 2].scatter(training_times, f1_scores,
                                     c=range(len(models)), cmap='viridis',
                                     s=100, alpha=0.7)
```

Додаток Ж

Повний код програми знаходиться за покликанням:

https://github.com/tetiana-zhuchenko/v-2-random-forest-api/blob/main/main.py

```
from fastapi import FastAPI, HTTPException
import joblib
import json
import re
import os
import numpy as np
import pandas as pd
from scipy.sparse import hstack, csr_matrix
from contextlib import asynccontextmanager
from fastapi.middleware.cors import CORSMiddleware
from supabase import create_client, Client
from dotenv import load_dotenv
import psycopg2
from psycopg2.extras import RealDictCursor

load_dotenv()

# Глобальні змінні для моделі та компонентів
model = None
vectorizer = None
scaler = None
aspect_keywords = None
feature_names = None
supabase_client: Client = None
db_connection = None

@asynccontextmanager
async def lifespan(app: FastAPI):
    global model, vectorizer, scaler, aspect_keywords, feature_names, supabase_client,  db_connection

    try:
        # Initialize Supabase client
        supabase_url = os.getenv("SUPABASE_URL")
        supabase_key = os.getenv("SUPABASE_KEY")

        if supabase_url and supabase_key:
            supabase_client = create_client(supabase_url, supabase_key)
            print("Supabase client initialized successfully!")
        else:
            print("Warning: Supabase credentials not found in .env file")

        # Initialize database connection
        database_url = os.getenv("DATABASE_URL")
        if database_url:
            db_connection = psycopg2.connect(database_url)
            print("Database connection established successfully!")
        else:
            print("Warning: DATABASE_URL not found in .env file")


        # Завантаження Random Forest моделі
        model = joblib.load('random_forest_model.pkl')
```

Додаток З

Вебсайт знаходиться за покликанням:

https://open-table-next-js-sand.vercel.app/