КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 – Комп'ютерні науки,

освітньо-наукова програма «Інформаційна аналітика та впливи»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему:

«Розробка системи автоматичної класифікації користувацьких відгуків із використанням методів машинного навчання»

Студентки 2-го курсу групи ІАВ-21

ЛАВРІЙ Софії

(підпис студента)

Науковий керівник:

к.е.н., доцент
(науковий ступінь, учене звання)

Мірошниченко Ігор Вікторович
(прізвище, ім'я, по батькові)

(дата) (підпис)

Попередній захист:

(Висновок: «До захисту в Екзаменаційній комісії»)

Завідувач кафедри технологій управління

(підпис) (прізвище, ініціали) (дата)

Київ-2025

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління
Освітньо-кваліфікаційний рівень Магістр
Спеціальність 122 – Комп'ютерні науки
Освітня програма Інформаційна аналітика та впливи

ЗАТВЕРДЖУЮ
Завідувач кафедри
Професор МОРОЗОВ В. В.
_____
«___» _____ 2025 р.

ЗАВДАННЯ
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студентка Лаврій С.П.

Група ІАВ-21

1. **Тема кваліфікаційної роботи**

Розробка системи автоматичної класифікації користувацьких відгуків із використанням методів машинного навчання

Затверджена наказом по від «___» _____ 2025р. № ____.

2. **Строк подання студентом готової роботи –** «___» _____ 2025р
3. **Цільова установка та вихідні дані до роботи**

Проаналізувати вплив різних факторів на автоматизацію класифікації коментарів користувачів та ефективність застосування методів машинного навчання на даних з обмеженою розміткою. Використати сучасні техніки класифікації текстових даних, зокрема активне навчання, слабоконтрольоване навчання та методи співнавчання, для розробки моделей, які забезпечать точність класифікації коментарів. Вихідні дані включають коментарі користувачів, попередньо визначені категорії

класифікації, частково розмічені дані, а також додаткові атрибути, такі як період проведення опитування та категорія.

4. Зміст роботи

У змісті роботи передбачено аналіз наукових джерел щодо застосування методів машинного навчання для класифікації коментарів користувачів з обмеженою розміткою, підбір та обробку відповідних текстових даних, побудову моделей класифікації на основі різних методів, таких як логістична регресія, випадковий ліс, SVM та XGBoost, а також вдосконалення моделей за допомогою MiniLM та технік самонавчання і співнавчання. Окрім цього, передбачено розробку інтерфейсу користувача для взаємодії з системою, аналіз перспектив розвитку системи, оцінку точності побудованих моделей та формулювання висновків.

5. Перелік графічного матеріалу (слайдів)

1 титульна сторінка, 2 слайди про постановку задачі, 3 слайди з описом даних, 1 слайд про підходи до класифікації тексту з обмеженою розміткою, 1 слайд про підготовку даних, 1 слайд про комбінації реалізованих моделей, 1 слайд з аналізом результатів, 2 слайди про інтерфейс, 1 слайд з перспективами розвитку та 1 слайд висновків.

6. Календарний план виконання роботи:

№ п/п	Назва частин роботи	%	Виконання роботи	
			За планом	Фактично
1	Вибір теми кваліфікаційної магістерської роботи, дослідження актуальності обраної теми, наявності наукових матеріалів з теми.	3	01.10.2024	01.10.2024
2	Протокол кафедри ТУ про затвердження тем дипломних робіт та призначення наукових керівників.	2	27.12.2024	27.12.2024
3	Формування переліку нормативних матеріалів, літератури з проблематики дипломної роботи.	10	15.02.2024	15.02.2024

4	Складання розгорнутого плану виконання та представлення кваліфікаційної роботи.	5	23.02.2025	23.02.2025
5	Ознайомлення наукового керівника з розгорнутим планом кваліфікаційної роботи. Внесення змін.	5	25.02.2025	25.02.2025
6	Підготовка розділу 1 «Постановка задачі та аналіз способів її вирішення».	10	16.03.2025	16.03.2025
7	Підготовка розділу 2 «Методи обробки текстових даних та засоби побудови моделей».	14	31.03.2025	31.03.2025
8	Підготовка розділу 3 «Побудова та оцінка моделей класифікації коментарів».	14	20.04.2025	20.04.2025
9	Підготовка розділу 4 «Інтерфейс користувача та перспективи розвитку».	13	26.04.2025	26.04.2025
10	Оформлення кваліфікаційної роботи. Підготовка аналізу результатів роботи, висновків. Перевірка відповідності початковій меті та задачам роботи.	15	04.05.2025	04.05.2025
11	Передача кваліфікаційної роботи науковому керівникові.	2	05.05.2025	05.05.2025
12	Передача кваліфікаційної роботи рецензенту для рецензування.	2	09.05.2025	09.05.2025
13	Попередній захист кваліфікаційної роботи.	5	12.05.2025	12.05.2025

Дата видачі завдання «____» _____ 2025 р.

Керівник роботи: к.е.н., доцент Мірошниченко І. В.

_____

(підпис)

Завдання прийняла до виконання студентка групи ІАВ-21 Лаврій С.П.

_____

(підпис)

ЗМІСТ

АНОТАЦІЯ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра технологій управління
Спеціальність 122 – Комп'ютерні науки,
освітня програма "Інформаційна аналітика та впливи"

Дипломна робота магістра Лаврій С.П..

Тема роботи – «Розробка системи автоматичної класифікації користувацьких відгуків із використанням методів машинного навчання».

Мета дипломної роботи магістра – збільшення ефективності аналізу зворотного зв'язку користувачів шляхом розробки та впровадження моделі автоматичної класифікації україномовних коментарів в умовах обмеженої доступності розмічених даних.

Об'єкт дослідження – текстові коментарі користувачів, зібрані в межах опитувань щодо якості контенту на платформі.

Предмет дослідження – методи автоматизованої обробки та класифікації текстових коментарів користувачів за тематичними категоріями з використанням інструментів Data Science.

Наукова новизна дослідження полягає у порівнянні підходів напівконтрольованого навчання для багатокласової класифікації текстів.

У роботі проведено огляд сучасних підходів до класифікації текстових даних в умовах обмеженої кількості розмічених прикладів. Здійснено попередню обробку користувацьких коментарів, їх векторизацію за допомогою методів TF-IDF та MiniLM, а також реалізовано побудову базових моделей класифікації (логістична регресія, випадковий ліс, SVM, XGBoost). Проведено експерименти із застосуванням методів

самонавчання та співнавчання для підвищення якості класифікації. Оцінено точність моделей за допомогою метрик precision, recall та F1-міри. Розроблено простий інтерфейс для демонстрації результатів класифікації та сформульовано висновки щодо ефективності різних підходів.

Дипломна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг – 109 сторінок, кількість джерел – 29.

Ключові слова: класифікація тексту, машинне навчання, слабоконтрольоване навчання, самонавчання, співнавчання, MiniLM, TF-IDF, аналіз коментарів, активне навчання, моделі класифікації.

ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ

1. NLP – Natural Language Processing (Обробка природної мови)
2. WSL – Weakly Supervised Learning (Слабоконтрольоване навчання)
3. FSL – Few-Shot Learning (Навчання з малою кількістю прикладів)
4. SSL – Semi-Supervised Learning (Напівконтрольоване навчання)
5. LR – Logistic Regression (Логістична регресія)
6. FR – Forest Regression (Регресія на основі лісу)
7. SVM – Support Vector Machine (Метод опорних векторів)
8. XGBoost – Extreme Gradient Boosting (Розширене підсилення градієнтного бустінгу)
9. NLTK – Natural Language Toolkit (Інструментарій для обробки природної мови)
10. TF-IDF – Term Frequency-Inverse Document Frequency (Частота терміну – обернена частота документа)
11. MiniLM – Miniature Language Model (Мініатюрна мовна модель)
12. TP – True Positive (Істинно позитивне спрацьовування)
13. FP – False Positive (Хибнопозитивне спрацьовування)
14. TN – True Negative (Істинно негативне спрацьовування)
15. FN – False Negative (Хибнонегативне спрацьовування)
16. API – Application Programming Interface (Інтерфейс прикладного програмування)
17. LLM – Large Language Model (Велика мовна модель)
18. SQL – Structured Query Language (Мова структурованих запитів)

ВСТУП

У сучасному цифровому світі компанії щодня стикаються з необхідністю обробки великих обсягів даних, зокрема зворотного зв’язку від користувачів. Зростаюча кількість даних, які надходять із різних каналів, зумовлює потребу у їх структурованому аналізі для покращення якості продукту, користувацького досвіду та бізнес-процесів загалом. Особливої актуальності набуває аналіз неструктурованих текстових даних, таких як відкриті коментарі, які містять цінну інформацію про проблеми, побажання та оцінку роботи компанії з боку користувачів.

У рамках дослідження розглядається приклад компанії, яка здійснює регулярні опитування для моніторингу якості контенту на онлайн-платформі. Учасники опитування залишають коментарі щодо свого досвіду використання платформи: висловлюють задоволення, формулюють зауваження або пропонують ідеї для вдосконалення. Зміст цих відгуків охоплює різні аспекти взаємодії із сервісом: модерацію контенту, функціональність платформи, безпеку, підтримку тощо.

Традиційна ручна обробка таких коментарів є надзвичайно ресурсомісткою та потребує значного часу аналітиків. Крім того, вона схильна до суб’єктивності та нерівномірності оцінювання. Застосування методів Data Science, зокрема методів обробки природної мови (NLP), відкриває нові можливості для автоматизованого аналізу зворотного зв’язку. Це дозволяє виявляти ключові теми, класифікувати коментарі за напрямками (наприклад, модерація, продукт, підтримка) та оперативно реагувати на потреби користувачів.

Завдяки машинному навчанню можна побудувати моделі, які здатні точно та ефективно розподіляти коментарі за змістовними категоріями. Це сприяє глибшому розумінню проблемних зон, покращенню якості сервісу, а також формуванню аналітичних звітів для прийняття обґрунтованих

управлінських рішень. Впровадження таких рішень дозволяє компанії зменшити витрати на ручну працю, підвищити швидкість обробки даних та уникнути людського фактора при аналізі.

Таким чином, використання інструментів Data Science у сфері аналізу зворотного зв’язку користувачів є потужним засобом підвищення якості обслуговування та ефективності бізнес-процесів. У даній кваліфікаційній магістерській роботі буде розглянуто розробку технології класифікації текстових коментарів за допомогою методів обробки природної мови, машинного навчання та статистичного аналізу.

Результатом дослідження стане побудова моделі, яка дозволяє автоматизувати процес обробки користувацьких коментарів та інтегрувати її в практичну діяльність компанії.

Метою даної магістерської кваліфікаційної роботи є збільшення ефективності аналізу зворотного зв’язку користувачів шляхом розробки та впровадження моделі автоматичної класифікації україномовних коментарів в умовах обмеженої доступності розмічених даних.

Об’єктом дослідження є текстові коментарі користувачів, зібрані в межах опитувань щодо якості контенту на платформі.

Предметом дослідження є методи автоматизованої обробки та класифікації текстових коментарів користувачів за тематичними категоріями з використанням інструментів Data Science.

Завдання дослідження:

1. Проаналізувати специфіку даних, що збираються в межах користувацьких опитувань щодо якості контенту.
2. Провести огляд сучасних підходів до автоматичної класифікації текстів та обробки природної мови.
3. Побудувати модель класифікації коментарів за визначеними тематиками з використанням методів машинного навчання.

4. Оцінити якість роботи побудованої моделі за допомогою відповідних метрик.
5. Створити звіт на основі результатів для покращення бізнес процесів.

Наукова новизна дослідження полягає у порівнянні підходів напівконтрольованого навчання для багатокласової класифікації текстів, зокрема застосуванні підходів самонавчання та співнавчання на україномовних даних з реального бізнес-кейсу.

Практична цінність роботи полягає в розробці ефективного підходу до автоматизованої класифікації текстових коментарів користувачів, що дозволяє значно зменшити обсяг ручної праці, прискорити аналіз зворотного зв’язку та забезпечити оперативне реагування на потреби користувачів платформи.

Запропонована модель може бути інтегрована в аналітичні системи компанії та використовуватися для регулярної обробки результатів опитувань, що підвищить якість прийняття управлінських рішень. Крім того, такий підхід сприяє покращенню користувацького досвіду, оскільки дозволяє швидше й точніше виявляти проблемні зони в роботі платформи та реагувати на них.

Отримані результати можуть бути використані не лише в межах досліджуваної компанії, а й адаптовані для інших організацій, які працюють із великими масивами відкритих коментарів і зацікавлені в автоматизації аналізу зворотного зв’язку.

Апробацією результатів даної роботи є публікація тез «Movie recommender system» у «X International Conference Information Technology and Implementation (Satellite) 2023» та тез «Development of Data Science technologies in the field of text classification» у «XI International Conference Information Technology and Implementation (Satellite) 2024».

Публікації:

1. S.P. Lavrii, J.I. Minaieva. Movie recommender system. Information Information Technology and Implementation (Satellite): Conference Proceedings, November 21, 2023, Kyiv, Ukraine / Ministry of Education and Science of Ukraine, Taras Shevchenko National University of Kyiv and [etc]; Vitaliy Snytyuk (Editor). – Kyiv: Publishing House «Caravela», 2023 [1].
2. Ihor Miroshnichenko, Sofia Lavrii. Development of data science technologies in the field of text classification. Information Technology and Implementation (Satellite): Conference Proceedings, November 21, 2024, Kyiv, Ukraine / Ministry of Education and Science of Ukraine, Taras Shevchenko National University of Kyiv and [etc]; Vitaliy Snytyuk (Editor). – Kyiv: Publishing House «Caravela», 2024 [2].

РОЗДІЛ 1
ПОСТАНОВКА ЗАДАЧІ ТА АНАЛІЗ СПОСОБІВ ЇЇ ВИРІШЕННЯ

1.1. Роль користувацьких коментарів та обґрунтування потреби автоматизації

Цифрові платформи активно розвиваються, і ключову роль у цьому відіграє користувацький зворотний зв’язок. Відкриті коментарі, що залишають користувачі в рамках опитувань або на сторінках продуктів, дають цінну інформацію про те, що насправді цікавить або турбує споживачів. Вони є джерелом різноманітних даних, серед яких можуть бути побажання щодо покращення продукту, похвала за хороші функції, а також критика щодо недоліків. Проте обробка таких коментарів за допомогою традиційних методів – ручного аналізу має ряд недоліків. Головним є те, що це надзвичайно трудомісткий процес і потребує значних ресурсів. Крім того, зростання кількості користувачів і збільшення обсягів зворотного зв’язку ускладнюють процеси ручного аналізу. Також ручне опрацювання коментарів є схильним до суб’єктивності, оскільки такі тексти часто охоплюють кілька тем одночасно, що ускладнює однозначне віднесення їх до певної категорії. Усі ці недоліки можуть призвести до затримок у прийнятті важливих рішень, що негативно впливатиме на розвиток бізнесу. Тому автоматизація процесу класифікації коментарів може значно підвищити ефективність, дозволяючи швидше отримувати важливу інформацію. Це дозволить не лише прискорити процес обробки відгуків, а й покращити точність категоризації, зменшивши вплив людського фактора.

Для збору зворотного зв’язку компанія регулярно проводить опитування серед своїх користувачів. Метою цього опитування є отримання безпосередньої інформації про якість контенту, проблеми, з

якими стикаються користувачі, та їхні побажання щодо вдосконалення платформи. Дослідження проводиться щоквартально серед активних користувачів, які взаємодіють з оголошеннями в різних категоріях, і включає низку питань, що дозволяють отримати як кількісні, так і якісні оцінки.

Основні питання охоплюють такі аспекти:

1. Загальна оцінка якості оголошень у вибраній категорії. Користувачі мають вибір між трьома варіантами відповіді: «Так», «Майже так» і «Ні», що дозволяє зрозуміти загальний рівень задоволеності.

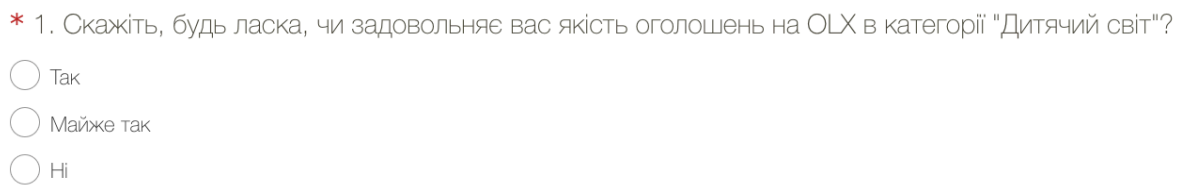


Рисунок 1.1 Приклад питання № 1

2. Суб’єктивне оцінювання частки неякісних оголошень на платформі загалом та в конкретній категорії. Для цього передбачено шкалу від 0 % до 50 %+, що допомагає визначити, наскільки серйозною є проблема в очах користувачів.

Рисунок 1.2 Приклад питання № 2

3. Перелік типів неякісних оголошень, з якими користувач стикався протягом останніх трьох місяців. Відповіді на це питання дають уявлення про найпоширеніші порушення та можуть слугувати основою для модераційних заходів.

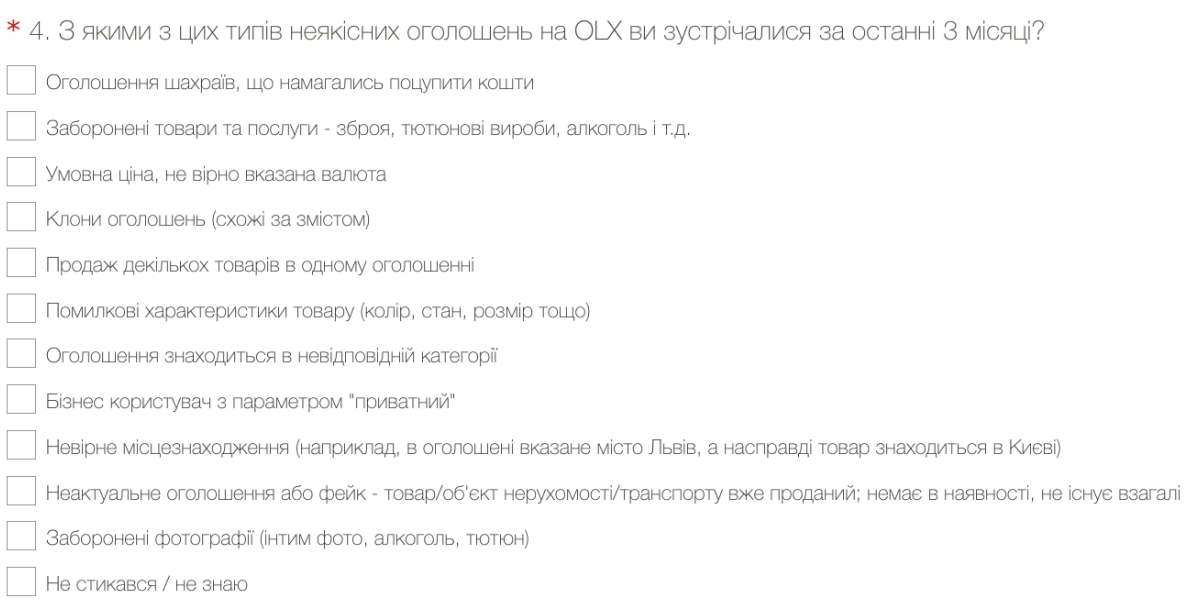


Рисунок 1.3 Приклад питання № 4

4. Відкрите питання для коментарів, у якому користувачі можуть висловити свої зауваження та пропозиції щодо покращення якості оголошень.

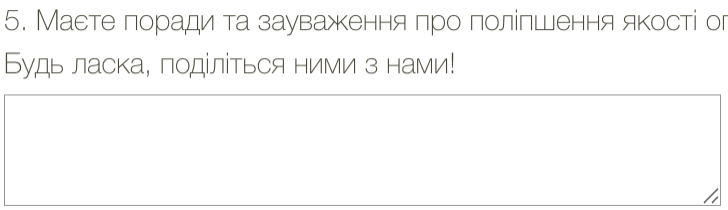


Рисунок 1.4 Питання № 5

На основі відповідей опитування можна зробити кілька важливих висновків. По-перше, результати дозволяють оцінити загальний рівень довіри користувачів до контенту на платформі та визначити категорії, де проблеми з якістю оголошень проявляються найсильніше.

По-друге, вони допомагають виявити типи порушень, з якими користувачі стикаються найчастіше, що, у свою чергу, дає можливість команді модерації ефективніше налаштовувати алгоритми перевірки.

Також отримані дані дозволяють відстежувати динаміку змін у якості контенту, аналізуючи, чи покращується ситуація після впровадження змін у модерації або алгоритмах відбору оголошень.

Окрім цього, користувацькі коментарі містять зворотний зв’язок, який є корисним не лише для модераційної команди, а і для продуктових менеджерів, що займаються вдосконаленням функціоналу платформи.

Зібрані метрики та якісні дані можуть бути використані командами модерації, аналітики, продакт-менеджменту та підтримки користувачів для прийняття рішень щодо покращення якості контенту та підвищення рівня довіри до платформи.

1.2. Сучасні підходи до класифікації даних з обмеженою розміткою

Задача автоматичної класифікації текстів традиційно розв’язується з використанням методів машинного навчання, які працюють із векторизованим представленням тексту. Найбільш поширеними класичними підходами є наївний байєсівський класифікатор, логістична регресія, метод опорних векторів (SVM) та деревоподібні моделі. Ці моделі зазвичай працюють на основі простих способів перетворення тексту в числові вектори, таких як Bag of Words або TF-IDF, і добре підходять для задач, де є достатній обсяг чистих, структурованих даних.

Іншим поширеним підходом є використання нейронних мереж, зокрема рекурентних нейронних мереж (RNN), LSTM чи GRU, які враховують послідовність слів у тексті та можуть краще працювати з короткими, контекстно-залежними висловлюваннями. У складніших випадках використовуються моделі на основі трансформерів, як-от BERT, що дають найвищі результати, однак мають вищі вимоги до обчислювальних ресурсів.

Проте ефективне використання вищезгаданих методів вимагає наявності достатньо великої та збалансованої вибірки вручну розмічених прикладів. У реальних умовах, як-от при роботі з користувацькими коментарями на платформах, створення такої вибірки може бути обмеженим. У таких випадках застосовуються підходи, здатні ефективно працювати із частково або слабо розміченими даними (weakly labeled data).

Залежно від типу обмежень у даних, слабо розмічені дані можна умовно поділити на три основні групи [7]. Перша група – inexact labeled data – описує ситуацію, коли мітки є занадто загальними або недостатньо деталізованими для конкретної задачі. Наприклад, усі коментарі можуть бути поділені лише на дві категорії: «позитивні» й «негативні», тоді як для глибшої аналітики важливо виділяти конкретні підкатегорії, як-от «скарги на модерацію», «проблеми з оплатою», «запити щодо функціоналу» тощо. Така груба розмітка може стати початковим етапом для подальшого уточнення класифікації.

Друга група – inaccurate labeled data – полягає в тому, що мітки, використані для навчання, можуть містити помилки. Наприклад, при автоматичному призначенні міток за ключовими словами або при швидкому ручному маркуванні частина коментарів може бути віднесена до невідповідної категорії через неоднозначність або людський фактор. Навіть при роботі з попередньо класифікованими даними (наприклад, із залученням модераторів) завжди існує ризик помилок через суб’єктивність оцінки або людський фактор.

Остання група – incomplete labeled data – характеризується тим, що лише частина навчального набору має наявні мітки, тоді як більшість прикладів залишаються нерозміченими.

У контексті даної роботи ми стикаємося з проблемою incomplete та inaccurate data. Це типова ситуація для бізнес-сценаріїв, таких як класифікація коментарів на платформах, де ручна модерація потребує

значних ресурсів, а оперативно розмітити великий обсяг нових даних є складним завданням. У результаті, більшість зворотного зв’язку залишається без класифікації, що ускладнює подальший аналіз і прийняття рішень.

1.2.1 Активне навчання

Активне навчання (Active learning) – це підхід у машинному навчанні, який дозволяє моделі самостійно визначати, які приклади варто розмітити, щоб досягти найкращого покращення якості з найменшими витратами. Ідея полягає в тому, що не всі дані є однаково корисними для навчання. Замість того, щоб випадково обирати приклади для розмітки, модель вибирає ті, щодо яких вона найбільш «невпевнена» або які є найбільш «інформативними». Це особливо важливо, коли розмітка даних дорога або трудомістка.

Найчастіше процес реалізується у вигляді циклу. Модель тренується на невеликій кількості розмічених даних. Після цього вона оцінює велику кількість нерозмічених прикладів і вибирає ті, які варто передати експерту для розмітки. Нові мітки додаються до тренувального набору, і модель перенавчається. Процес повторюється, поки не буде досягнута бажана якість або вичерпаний бюджет.

Існує кілька стратегій вибору прикладів:

- Вибір невпевнених прикладів (uncertainty sampling): модель обирає ті приклади, у яких вона найменше впевнена, наприклад, класифікує з ймовірністю, близькою до 0.5 у задачі двійкової класифікації.
- Опитування комітету (query-by-committee): декілька моделей або одна модель із різними гіперпараметрами роблять прогнози на нерозмічених прикладах, і ті приклади, де між ними найбільша розбіжність, передаються на розмітку.

- Вплив на модель (expected model change): вибираються ті приклади, які найбільше змінять параметри моделі після додавання їх до тренувального набору.
- Урізноманітнення (diversity sampling): замість фокусування на невпевненості, модель обирає приклади, які репрезентують різні ділянки простору ознак, щоб забезпечити максимальне покриття.

Такий підхід дозволяє працювати з даними, де невелика кількість розмітки, проте вимагає наявності «людського експерта», здатного оперативно класифікувати відібрані приклади. Важко заздалегідь оцінити, скільки і яких саме прикладів потрібно буде розмітити. Крім того, у деяких випадках «невпевнені» приклади можуть бути просто «шумом» або аномаліями, які не дають користі при навчанні.

Активне навчання може бути більш ефективним у поєднанні з іншими підходами, наприклад, semi-supervised learning або few-shot.

1.2.2 Слабоконтрольоване навчання

Ще одним із підходів, який дозволяє ефективно працювати в умовах обмеженої або непрямої розмітки, є слабоконтрольване навчання (WSL – weakly supervised learning). Це підхід, при якому замість точних, ручних міток застосовуються приблизні або непрямі джерела інформації для навчання моделі.

У рамках підходу weak supervision для створення слабкої розмітки можуть використовуватися різні джерела, які дозволяють автоматизувати або частково автоматизувати процес призначення міток. Одним із найпростіших підходів є формулювання евристичних правил. Це можуть бути прості логічні умови, розроблені аналітиком або експертною командою.

Ще одним джерелом слабкої розмітки є зовнішні ресурси, як-от бази знань, спеціалізовані словники, уже існуючі класифікатори або метадані,

пов’язані з даними. Наприклад, інформація про категорію оголошення, тип пристрою чи час доби, коли було залишено коментар, може слугувати непрямими ознаками належності цього коментаря до певної категорії. Такі джерела допомагають збагачувати датасет додатковими сигналами, навіть якщо сам текст не містить явних маркерів.

Ще один популярний підхід – це навчання на основі відстані (distant supervision), де мітки призначаються на основі відповідності тексту зовнішнім ресурсам. Наприклад, якщо заздалегідь відомо, що певні слова або фрази часто пов’язані з конкретними темами, коментарі, які містять такі вирази, можуть автоматично отримувати відповідну мітку. Цей метод дозволяє масштабувати розмітку на великі обсяги тексту без потреби в повній ручній перевірці.

Окрім цього, застосовується також підхід агрегації думок як моделей, так і людей. Якщо кілька простих моделей незалежно одна від одної дають однаковий результат для певного прикладу, цей результат можна вважати достатньо достовірною слабкою міткою. Аналогічно, коли група аналітиків або модераторів досягає згоди щодо класифікації певного коментаря, така мітка теж може бути використана в якості слабкої.

Одним із популярних фреймворків, який реалізує weak supervision, є Snorkel. У ньому експерт формулює набір так званих labeling functions – правил, які автоматично присвоюють мітки текстам. Далі Snorkel оцінює якість кожної функції, об’єднує результати, і генерує імовірнісну розмітку, яку можна використовувати для навчання моделі [9].

Підхід WSL має низку суттєвих переваг, що робить його привабливим у ситуаціях, коли повноцінна ручна розмітка є занадто ресурсозатратною. По-перше, він дозволяє швидко отримати базову розмітку великого корпусу текстів, що особливо цінно для початкових етапів дослідження або побудови прототипів. Завдяки цьому можна розпочати експерименти без необхідності формування повноцінного

ручного корпусу, що значно скорочує час і ресурси на підготовку даних. Крім того, weak supervision допомагає виявити потенційні слабкі місця в майбутній системі категоризації ще до її повноцінного запуску, що дає змогу скоригувати стратегію до того, як система буде застосована в реальних умовах.

Однак, разом із перевагами, цей підхід має і низку обмежень. Основна проблема полягає в тому, що слабка розмітка часто містить шум – помилкові або непослідовні мітки, які можуть негативно вплинути на якість навчання моделей. Також важливим викликом є створення ефективних евристик або функцій для автоматичної розмітки: це вимагає глибокого розуміння предметної області, і не завжди є можливим за короткий час або без залучення фахівців. Ще одна складність пов’язана з тим, що в багатьох випадках коментарі можуть охоплювати кілька тем одночасно, що ускладнює забезпечення узгодженості міток і може призводити до неоднозначностей у навчанні.

Часто ці підходи використовуються як початковий етап побудови системи класифікації, результати якого надалі уточнюються за допомогою інших технік.

1.2.3 Few-shot класифікація

Few-shot класифікація (few-shot learning / classification, FSL) – це підхід до побудови моделей, які здатні навчатися класифікувати об’єкти, маючи лише кілька прикладів кожного класу. Такий підхід особливо актуальний для реальних умов, де збирання великої кількості міток є складним або неможливим, як-от у класифікації користувацьких коментарів на онлайн-платформах. У таких випадках часто виникає потреба в оперативному додаванні нових категорій, але при цьому є доступними лише поодинокі приклади.

У цьому підході часто використовують терміни n-way та m-shot: n-way означає, що модель має обрати правильний клас серед n можливих; m-shot означає, що для кожного із цих класів доступно лише m розмічених прикладів у тренувальному наборі. Тренувальний набір називають support set. Підмножина прикладів, яку потрібно класифікувати називається query set [8].

Багато методів few-shot classification ґрунтуються на ідеях meta learning – підходу, за якого модель навчається на великій кількості схожих задач для того, щоб у майбутньому ефективно вирішувати нові задачі з обмеженою кількістю прикладів. Іншими словами, meta learning дозволяє системі «навчитися навчатися», щоб швидко адаптуватися до нових класів або категорій із кількома прикладами. Далі розглянемо найпопулярніші підходи реалізації FSL [3]:

1. Прототипні мережі (Prototypical Networks) – цей метод базується на припущенні, що кожен клас можна представити як центр (прототип) у векторному просторі ознак. Для кожного класу обчислюється середній вектор (прототип) на основі доступних прикладів. Нові зразки класифікуються шляхом вимірювання відстані до кожного з прототипів. Перевагою підходу є його простота та ефективність у низькоресурсних умовах. Проте метод може бути чутливим до варіативності даних у класі [10].
2. Мережі відповідності (Matching Networks). У цьому підході використовується механізм уваги для порівняння нових прикладів із тими, що вже відомі. Модель оперує всією навчальною підмножиною і використовує подібність між вектором нового прикладу та векторами прикладів зі support set, щоб визначити найімовірніший клас. Matching Networks вміють краще враховувати локальні контексти та зв’язки між прикладами, що підвищує їхню гнучкість.

3. Мережі відношень (Relation Networks) – це ще один підхід, що спирається на порівняння пар «приклад–клас». Відмінність у тому, що замість евклідової відстані або косинусної подібності модель навчає спеціальну нейронну мережу (relation module), яка визначає «ступінь схожості» між прикладами. Такий модуль дозволяє моделі краще адаптуватися до різних задач і мати глибше розуміння взаємозв'язків.
4. Індуктивні мережі (Induction Networks). Їх ключова особливість – використання індукційного модуля, що будує узагальнене представлення класу на основі набору прикладів [4]. На відміну від простого усереднення в Prototypical Networks, Induction Networks застосовують динамічний фільтр, який враховує як індивідуальні характеристики кожного прикладу в класі, так і їхню взаємодію. Це дозволяє моделі краще адаптуватися до ситуацій, де приклади в межах одного класу значно відрізняються між собою. Особливо це корисно в задачах із текстами, де значення сильно залежить від контексту.

Few-shot класифікація відкриває нові можливості для розв'язання задач машинного навчання в умовах обмежених даних. Його головна перевага полягає в здатності моделі швидко адаптуватися до нових класів на основі кількох прикладів. Це дозволяє значно зменшити залежність від вручну розмічених датасетів і скоротити час, необхідний для запуску нових систем. Крім того, FSL моделі часто розвивають кращу узагальнювальну здатність порівняно з традиційними моделями, адже навчаються використовувати вже наявні знання для розпізнавання нових концептів.

Втім, few-shot підходи мають і свої обмеження. Їх ефективність суттєво залежить від якості представлених прикладів: якщо support set не є репрезентативним або містить шум, модель може легко помилитися. Також

деякі існуючі FSL-архітектур демонструють високу чутливість до змін у формулюваннях або стилі даних, що ускладнює їх застосування в реальних, більш варіативних середовищах. Не всі типи текстових задач легко адаптуються до few-shot формату, зокрема, це стосується випадків, коли межі між класами нечіткі або значно перекриваються, що ускладнює класифікацію навіть для людей. Окрім того, навчання таких моделей часто вимагає попереднього pre-training на великих наборах даних, тобто повна відмова від великих обсягів даних у загальному процесі навчання зазвичай неможлива.

1.2.4 Навчання із частковим наглядом

Навчання із частковим наглядом (semi-supervised learning, SSL) – це підхід, який дозволяє ефективно використовувати поєднання невеликої кількості розмічених прикладів та великої кількості нерозмічених. Основна ідея SSL полягає в тому, щоб дозволити моделі «підказувати самій собі», як класифікувати нові приклади, використовуючи отримані знання з маркованих даних. Розмічені дані задають напрямок, тоді як нерозмічені допомагають уточнити структуру простору ознак. У найтиповішому сценарії спочатку формується базова модель, натренована на обмеженому наборі з мітками. Далі вона передбачає мітки для нерозмічених прикладів, і найнадійніші з них додаються до тренувального набору, поступово вдосконалюючи модель.

Залежно від принципу, за яким відбувається включення нерозмічених даних, виділяють кілька підходів [14].

Перший підхід – самонавчання (self-training). Його суть полягає в тому, що модель спочатку навчається на невеликому наборі розмічених даних, після чого прогнозує мітки для нерозмічених текстів. Найбільш впевнені передбачення додаються до навчального набору як нові розмічені зразки, і модель продовжує навчання. Цей процес повторюється кілька

ітерацій, поступово розширюючи набір розмічених даних. Метод простотий у реалізації та працює добре, коли початковий набір має достатню якість, але може накопичувати помилки, якщо модель робить неправильні передбачення на ранніх етапах. Такий підхід часто використовується в поєднанні з традиційними моделями машинного навчання, такими як SVM або нейромережі.

Другим популярним підходом є співнавчання (co-training). Він передбачає використання двох або більше моделей, кожна з яких навчається на різних «видах ознак», тобто незалежних представленнях одних і тих самих даних. Припускається, що ці представлення доповнюють одне одного. Обидві моделі тренуються на одному невеликому наборі розмічених прикладів. Потім кожна з них робить передбачення на нерозміченому наборі. Найбільш впевнені передбачення однієї моделі використовуються як нові мітки для навчання іншої моделі. Цикл повторюється, поки не буде використано всі або більшість даних.

Також використовуються графові методи (graph-based methods). Це клас підходів, що базується на уявленні всіх даних як розмічених, так і нерозмічених у вигляді графа. У цьому графі вузли відповідають окремим прикладам, наприклад, текстам, реченням чи документам, а ребра відображають міру подібності або зв’язку між ними. Зазвичай вага ребра вказує на ступінь близькості між двома вузлами, яку можна обчислити. Головна ідея полягає в тому, що схожі приклади повинні мати однакові мітки – це називається гладкістю міток (label smoothness assumption). На практиці це реалізується через механізм поширення міток (label propagation), коли моделі дозволяється «переносити» мітки з розмічених вузлів до їхніх найближчих сусідів у графі. У таких підходах можуть використовуватись як класичні алгоритми поширення міток, так і сучасні нейронні архітектури, наприклад, графові нейронні мережі (Graph Neural Networks, GNNs). GNNs не просто поширюють мітки, а й дозволяють

навчити векторні представлення вузлів, які враховують як локальний контекст (сусідів), так і глобальну структуру графа. Проте побудова якісного графа вимагає обчислення подібностей між усіма прикладами, що може бути ресурсомістким при великих обсягах даних; також результат сильно залежить від того, наскільки релевантними є зв’язки між вузлами.

Ще одним підходом є регуляризація узгодженості (consistency regularization). Він заснований на припущенні, що хороша модель повинна бути стійкою до незначних змін у вхідних даних. У практиці напівконтрольованого навчання це правило використовується як регуляризаційний принцип. Під час тренування до вхідного прикладу застосовуються різні аугментації і модель навчається таким чином, щоб її вихід був максимально схожим для оригіналу і змінених версій. У випадку з текстами це можуть бути перефразування, видалення або заміна окремих слів синонімами, інші стилістичні трансформації. Метою є заохочувати модель фокусуватися на суттєвих ознаках, які визначають клас, а не на поверхневих відмінностях. Це дозволяє краще робити узагальнення на нові, раніше не бачені приклади, що особливо важливо у випадках, коли розмічених даних дуже мало. Одним із відомих представників цього підходу є метод UDA (Unsupervised Data Augmentation), який поєднує consistency regularization з потужними аугментаціями тексту і використовує невеликі обсяги розмічених даних разом із великою кількістю нерозмічених. Цей підхід довів свою ефективність у багатьох задачах, адже дозволяє використовувати потенціал нерозмічених прикладів не просто як дані для класифікації, а як інструмент для підвищення стабільності та надійності моделі.

1.3 Аналіз іноземних та вітчизняних науково-інформаційних джерел

У межах вивчення науково-інформаційних джерел, присвячених обробці користувацьких відгуків, варто звернути увагу на вже наявні дослідження, дава що стосуються українськомовного сегменту. Одним із прикладів є стаття «Method for Sentiment Analysis of Ukrainian-Language Reviews in E-Commerce Using RoBERTa Neural Network» [5]. Метою дослідження було створення ефективного методу автоматичного аналізу тональності україномовних відгуків. Було сформовано розмічений датасет, побудовано семантичну модель мови та налаштовано класифікатор на базі архітектури RoBERTa. Отримана модель показала точність у 92 %, що підтверджує дієвість такого підходу.

Незважаючи на те, що в дослідженні акцент зроблено на сентимент-аналізі з бінарною класифікацією, воно є корисним із погляду підходів до обробки україномовного тексту. Автори застосовують модель RoBERTa, яка адаптована до мовних особливостей, зокрема до наявності суржику та розмовних форм.

У роботі «Determining sentiment and important properties of Ukrainian-language user reviews» [11] досліджується аналіз україномовних користувацьких відгуків у сфері готельного бізнесу. Метою дослідження є розробка методу, який дозволяє автоматично визначати емоційну тональність відгуків та виявляти ключові характеристики, що вплинули на таку оцінку. У роботі порівнюються різні підходи до обробки тексту, зокрема fastText, seq2seq, TextCNN, TextRNN і RCNN. Найвищу точність показала модель RCNN з показниками 90 % для одного набору даних та 85 % для іншого.

У зазначених проектах використовувалися повністю розмічені дані, тоді як у нашому випадку доступна лише частина міток, що вимагає

застосування підходів розглянутих у розділі 1.2. Тому далі розглянемо роботи, які орієнтовані на аналіз частково розмічених даних.

У статті «Active Learning Strategies for Efficient Text Classification» [12] досліджується застосування активного навчання для ефективної класифікації текстів із метою покращення точності класифікації при мінімальних витратах на розмітку даних. Експерименти проводилися на реальних наборах даних: на відгуках з Taobao (3 класи) та запитах клієнтів інтернет-магазинів (5 класів). У дослідженні використовувались наївний байєсівський класифікатор, SVM та логістична регресія. Згідно з результатами, застосування стратегії вибору даних для додаткової розмітки за невизначеністю (найближчі до лінії розмежування) дало найкращі показники ефективності. Найбільш ефективним був метод SVM з результатами 87 % точності, 85,6 % F1 та 0.89 AUC.

У дослідженні «Weakly-Supervised Deep Learning for Customer Review Sentiment Classification» [13] автори описують розробку ефективної модель для класифікації настроїв відгуків, використовуючи слабоконтрольоване навчання, а саме доступні рейтинги як слабкі мітки. Пропонується двоетапна архітектура: на першому етапі відбувається навчання моделі WED (Weakly-supervised Deep Embedding) – підхід, який поєднує використання слабко контрольованих міток із потужними глибинними нейронними мережами для навчання ефективних і корисних векторних представлень даних, а на другому етапі додається класифікаційний шар і здійснюється донавчання за допомогою вручну розмічених даних. WED дала найкращі результати (87.7 % accuracy, 87.6 % F1) порівняно з SVM, NB, CNN та іншими моделями.

У роботі «Induction Networks for Few-Shot Text Classification» [4] автори запропонували нову архітектуру для класифікації тексту в умовах few-shot навчання. Основна ідея полягає у використанні індуктивних мереж (Induction Networks), які за допомогою механізму dynamic routing

будують узагальнене представлення кожного класу на основі support-прикладів і потім використовувати його для класифікації нових запитів. Для оцінки ефективності свого підходу автори порівнювали Induction Networks із кількома базовими моделями: Matching Networks, Prototypical Networks, Graph Network, Relation Network. Найкращі результати модель продемонструвала в конфігурації 5-way 10-shot, досягнувши точності 88.5 % та перевищивши точність інших моделей, що свідчить про ефективність даної архітектури для класифікації тексту з обмеженою кількістю міток.

У статті «Analyzing the effectiveness of semi-supervised learning approaches for opinion spam classification» [6] досліджується ефективність напівконтрольованого навчання для задачі виявлення фальшивих відгуків. Автори оцінювали, наскільки напівконтрольовані методи можуть допомогти у випадках, коли є обмежена кількість розмічених даних. Спочатку порівнювали три базові моделі: SVM, Naive Bayes (NB) та Random Forest (RF), які тестували на наборах із 5 %, 10 %, 20 % і 100 % розмічених даних. Після цього кожну модель поєднували із самонавчанням (self-training) та співнавчанням (co-training), проводячи аналогічні експерименти для 5 %, 10 % та 20 % розмічених даних. Жоден метод не перевищив точність 94 % для повністю контрольованого навчання (NB із 100 % даних на TF-IDF біграмах). Проте поєднання самонавчання та NB (на біграмах) на 20 % даних дало 93 % точності. Таким чином, результати дослідження демонструють, що використання нерозмічених даних через self-training дозволяє досягти майже такого ж рівня якості класифікації, як при повністю контрольованому навчанні.

1.4. Постановка задачі дослідження

Проаналізувавши існуючі наукові підходи та результати досліджень у сфері обробки текстових даних, класифікації користувацьких відгуків та

методів навчання із частково розміченими даними, можна сформулювати постановку задачі даного дослідження.

Метою роботи є збільшення ефективності аналізу зворотного зв'язку користувачів шляхом розробки та впровадження моделі автоматичної класифікації україномовних коментарів в умовах обмеженої доступності розмічених даних. Модель повинна працювати за принципом «чорного ящика», де на вхід подається текст коментаря, а на вихід – відповідна категорія.

Для досягнення поставленої мети необхідно реалізувати наступні *завдання*:

1. Отримання та аналіз даних – здійснити завантаження текстових відгуків із внутрішньої системи компанії через API, провести початковий огляд структури даних та визначити доступність розмітки.
2. Попередня обробка тексту – очистити дані, враховуючи особливості україномовного середовища: змішування мов, орфографічні варіації, розмовні форми.
3. Вибір методів машинного навчання – дослідити та обрати методи, що дозволяють працювати із частково розміченими даними: підходи weakly-supervised, semi-supervised, few-shot та active learning.
4. Побудова моделі класифікації – реалізувати модель, що дозволяє ефективно розпізнавати зміст коментарів з урахуванням невеликої кількості міток.
5. Оцінювання якості моделі – провести валідацію отриманих результатів за допомогою відповідних метрик та порівняти з базовими підходами.
6. Формування ключових скарг/пропозицій по кожній категорії – виділити топ біграм по кожній із категорій.

7. Аналітичний звіт – автоматизувати створення періодичних звітів на основі результатів класифікації.

У ході виконання роботи необхідно врахувати кілька важливих факторів, таких як мовна неоднорідність, проблема багатокласовості коментарів та неповна розмітка даних. Мовна неоднорідність проявляється в присутності суржику, русизмів та розмовних конструкцій у текстах, що ускладнює обробку та аналіз текстових даних. Проблема багатокласовості виникає, коли один коментар може стосуватись кількох класів одночасно, що ускладнює правильну класифікацію і створення узагальнюючих моделей. Неповна розмітка даних є ще однією важливою проблемою, оскільки лише частина даних має мітки, що значно обмежує можливості застосування традиційних методів повністю контрольованого навчання.

Висновки

У цьому розділі було детально розглянуто постановку задачі та аналіз різних підходів до її вирішення, зокрема в контексті класифікації користувацьких коментарів. Спочатку було обґрунтовано важливість автоматизації аналізу відгуків: це дозволяє ефективно обробляти великі обсяги даних та виявляти ключові патерни, які можуть допомогти в поліпшенні продуктів або послуг. Також були розглянуті сучасні методи класифікації даних з обмеженою розміткою, зокрема активне навчання, слабоконтрольоване навчання, few-shot класифікація та навчання із частковим наглядом. Ці підходи дозволяють ефективно працювати з великими обсягами даних у ситуаціях, коли є лише частина міток.

Різні дослідження, як іноземні, так і вітчизняні, продемонстрували високу ефективність цих методів у подібних контекстах. За допомогою аналізу літератури було обґрунтовано, що ці підходи мають високу точність класифікації. На основі цього аналізу були сформульовані мета та завдання кваліфікаційної роботи.

РОЗДІЛ 2
МЕТОДИ ОБРОБКИ ТЕКСТОВИХ ДАНИХ І ЗАСОБИ ПОБУДОВИ МОДЕЛЕЙ

2.1 Попередня обробка тексту та векторизація

Перед застосуванням будь-якого методу класифікації або аналізу тексту надзвичайно важливо виконати попередню обробку даних. Попередня обробка дозволяє зменшити варіативність текстових даних, виділити ключову інформацію та забезпечити стабільну роботу алгоритмів машинного навчання. Якісна нормалізація тексту також сприяє кращій узагальнюваності моделі, знижує кількість помилкових співставлень та покращує інтерпретованість результатів. У статті «Data preprocessing and tokenization techniques for technical Ukrainian texts» [15] описано, що попередня обробка тексту може скоротити словниковий запас термінів, що дає можливість значного скорочення обчислювального часу та навантаження.

1. Приведення до нижнього регістру.

Очистка тексту є одним із ключових етапів попередньої обробки, адже сирі текстові дані часто містять багато шуму, який може завадити ефективному навчанню моделі. Перш за все виконується перетворення тексту до нижнього регістру, що дозволяє уникнути дублювання ознак через різні форми одного й того ж слова з великої та малої літери. Наприклад, слова «Продам» і «продам» будуть трактуватись як одне й те саме.

2. Видалення символів.

З тексту видаляються всі небуквені символи: розділові знаки, цифри, лапки, дужки, HTML-теги, спеціальні символи (наприклад, @, #,%, &, *, тощо), а також зайві пробіли. Це робиться для зменшення розмірності

простору ознак і виключення «сміттєвих» елементів, які не мають семантичного навантаження.

Після очищення тексту наступним етапом є токенізація. Вона полягає в розбитті суцільного тексту на менші смислові одиниці – токени. Найчастіше токенами є окремі слова, але залежно від задачі ними також можуть бути речення, складові слова або морфеми. Правильне розбиття тексту на токени дозволяє зберегти максимально релевантну інформацію для подальшої обробки.

Токенізація має низку особливостей при роботі з українськими текстами, особливо якщо вони містять суржик, русизми або розмовні конструкції. У таких випадках стандартні токенізатори, розроблені для англійської чи інших мов, можуть давати некоректні результати, наприклад, об’єднувати слова або ділити їх у неправильних місцях. Тому важливо обирати інструменти, які підтримують українську мову або дозволяють адаптацію під конкретний корпус. Нерідко для цього застосовуються готові NLP-бібліотеки (наприклад, nltk [18], spaCy [19], Stanza [20]).

Після токенізації наступним кроком зазвичай є видалення стоп-слів – найпоширеніших слів мови, які не несуть суттєвого смислового навантаження. До таких слів належать, зокрема, службові частини мови: сполучники, прийменники, займенники, допоміжні дієслова тощо (наприклад, «і», «або», «в», «це», «я», «було»). Їх присутність у корпусі даних може призводити до «зашумлення» моделі, оскільки ці слова трапляються у великій кількості текстів незалежно від змісту.

Видалення стоп-слів допомагає зменшити кількість ознак і покращити якість векторного представлення. Однак варто враховувати, що в окремих задачах (наприклад, при роботі з короткими текстами чи в задачах сентимент-аналізу) повне вилучення таких слів може призвести до втрати важливої інформації. Для української мови існують попередньо

зібрані списки стоп-слів, які можна адаптувати або розширити з урахуванням специфіки дослідження. Видалення стоп-слів зазвичай виконується після токенізації, оскільки саме на рівні окремих токенів найзручніше ідентифікувати непотрібні елементи.

У роботі «»The influence of preprocessing on text classification using a bag-of-words representation» [16] досліджувалась залежність якості моделі класифікації та набору попередньої обробки даних. Найкращі результати показала комбінація CHS (точність класифікації за допомогою SVM досягла 95,7 %), де C – виправлення помилок, H – видалення HTML тегів і S – видалення стоп-слів.

Інколи застосовується також видалення повторюваних символів або слів, що зменшує надмірну варіативність без втрати змісту. На цьому етапі також може виконуватись стандартизація форматування, наприклад, заміна діалектних або неформальних написань на стандартні. Залежно від завдання та специфіки мови, також можливе коригування або видалення слів, які написані з помилками, або мають ознаки суржику. У сукупності всі ці кроки дозволяють підготувати чистий і уніфікований текст, що значно підвищує якість подальшої токенізації, векторизації та класифікації.

Окремим етапом попередньої обробки тексту є нормалізація слів, до якої належать стемінг та лематизація. Обидва методи мають на меті зменшити кількість унікальних форм одного й того ж слова, зводячи їх до спільної базової форми, однак реалізуються по-різному.

Стемінг (stemming) – це процес обрізання закінчень слів для отримання їхнього «стема», тобто основи, яка не обов’язково є повноцінним словом. Стемінг виконується за фіксованими правилами, не враховуючи граматичну коректність, тому часто дає грубі скорочення. Наприклад, слова «працюю», «працював», «праця» можуть бути перетворені на «прац». Хоча це дозволяє значно зменшити розмір словника

і пришвидшити обчислення, такий підхід іноді призводить до втрати точності або до неправильного об’єднання семантично різних слів.

Лематизація (lemmatization) – це більш складний процес, який передбачає приведення слова до його леми – словникової форми, яка відповідає значенню в контексті. Для цього використовуються словники або морфологічний аналізатор, що враховує частину мови та граматичні особливості. Наприклад, слово «бігла» буде зведено до леми «бігти». Лематизація дає точніші результати, ніж стемінг, але потребує більше ресурсів та підтримки для конкретної мови.

Після виконання етапів попередньої обробки тексту наступним кроком є векторизація – процес перетворення текстових даних у числові представлення, які можуть бути використані алгоритмами машинного навчання. Це критично важливо, адже більшість моделей не здатні працювати безпосередньо з текстом, їм необхідно подавати дані у вигляді векторів фіксованої довжини або матриць.

Існує кілька підходів до векторизації тексту, кожен із яких має свої переваги та недоліки. Найбільш поширені серед них:

- Bag-of-Words (BoW): представляє текст як вектор частот слів, не враховуючи їхній порядок або контекст;
- TF-IDF (Term Frequency – Inverse Document Frequency): зважує частоту слів з урахуванням їхньої унікальності для конкретного документа;
- Word embeddings (наприклад, Word2Vec, FastText): використовують попередньо навчені моделі для перетворення слів у щільні вектори з урахуванням семантичної близькості;
- Contextual embeddings (наприклад, BERT, nomic-embed): дозволяють враховувати контекст використання слова, створюючи різні векторні представлення залежно від оточення.

У дослідженні «An Evaluation of Preprocessing Techniques for Text Classification» [17] описано, що TF-IDF векторизація дає кращі результати для класифікації тексту, ніж Bag of words у поєднанні з chi-square (відбір найбільш важливих ознак за допомогою статистичного тесту, який оцінює залежність між словом і класом). Тому для baseline моделей буде використовуватись саме цей вид векторизації.

TF-IDF є одним із найпоширеніших підходів для класичних моделей машинного навчання. Він поєднує локальну важливість слова в конкретному документі з його глобальною рідкістю в усій колекції. Базується на двох ідеях:

1. TF (term frequency) – показує, як часто певне слово зустрічається в конкретному документі. Чим частіше – тим воно важливіше для цього документа.
2. IDF (inverse document frequency) – навпаки, показує, наскільки слово є рідкісним в усіх документах. Чим рідше слово з'являється в колекції текстів, тим воно інформативніше.

Комбінуючи ці два показники, ми отримуємо вагу кожного слова, яка одночасно враховує його значущість у конкретному тексті й те, наскільки воно унікальне в порівнянні з іншими текстами. Наприклад, слово «оголошення» може зустрічатись у кожному коментарі, тому його вага буде низькою. А от слово «заблокували» може з'являтись лише в скаргах, тож воно отримає вищу вагу й допоможе моделі краще відрізнити ці коментарі від інших.

Формально кожен термін *t* в документі *d* отримує вагу за формулою:

$$TF - IDF\,(t,\ d)\ =\ TF(t,\ d)\ *\ IDF\,(t) \qquad (1)$$

Частоту терміна розраховують, як:

$$TF_{t,d} = \frac{f_{t,d}}{\sum\limits_{k} f_{k,d}}, \qquad (2)$$

де$f_{t,d}$– кількість разів, коли термінtз'явля'ться в документіd, а$\sum_k f_{k,d}$– загальна кількість слів у документіd.

Зворотна частота документів розраховується за такою формулою:

$$IDF_t = log(\frac{N}{1+n_t}), \quad (3)$$

де N – загальна кількість документів у корпусі,n_t– кількість документів, що містять термін t, 1 у знаменнику використовується для уникнення ділення на нуль.

Фінальний показник буде високим, якщо слово часто зустрічається в конкретному тексті, але рідко в інших документах. Таке зважування допомагає моделі зосередитись на найбільш інформативних словах.

Проте для підвищення якості аналізу тексту в роботі було також застосовано іншу модель для семантичної векторизації – Paraphrase Multilingual MiniLM-L12-v2, яка входить до бібліотеки sentence-transformers [25]. Вона побудована на базі архітектури BERT та модифікована за принципами Sentence-BERT (SBERT). SBERT трансформує BERT у сіамську архітектуру, здатну ефективно обчислювати векторні представлення речень, які зберігають семантичну схожість [21]. Дана модель була адаптована до компактної та швидкої модифікації MiniLM, що дозволяє досягати високої якості представлення тексту при знижених обчислювальних витратах.

Модель була навчена на парафразних парах із великої кількості мов, що забезпечує її здатність формувати якісні багатомовні семантичні ембеддинги. Вона підтримує понад 50 мов, включаючи українську, що робить її універсальним інструментом для багатомовних задач обробки природної мови. Paraphrase Multilingual MiniLM-L12-v2 формує щільні вектори фіксованої довжини 384 елементи, які зберігають основне семантичне значення тексту [22].

Ця модель є ефективним рішенням для широкого спектра завдань: класифікації, кластеризації, семантичного пошуку, виявлення дублікатів та оцінки схожості між текстами. Вона показує високу точність на багатьох бенчмарках, таких як SEB [24] та MTEB [23], і при цьому залишається компактною та швидкою в роботі. Завдяки гарному балансу між якістю, швидкістю та ресурсною ефективністю, Paraphrase Multilingual MiniLM-L12-v2 є зручним інструментом як для досліджень, так і для практичного використання.

2.2 Методи класифікації текстових даних

2.2.1 Логістична регресія

Логістична регресія є одним із базових статистичних методів класифікації, що активно використовується як у задачах бінарного, так і багатокласового навчання. Основна ідея логістичної регресії полягає в оцінюванні ймовірностей належності прикладу до кожного з класів на основі лінійної комбінації ознак. Вона розраховується для кожного класу k:

$$z_k = \beta_0^{(k)} + \beta_1^{(k)} x_1 + \beta_2^{(k)} x_2 + \ldots + \beta_n^{(k)} x_n, \quad (4)$$

де:

- z_k – лінійне значення для класу k,
- $\beta_j^{(k)}$ – коефіцієнти моделі для класу k,
- x_j – ознака (векторизоване значення тексту),
- n – кількість ознак (розмірність вектора ознак).

Для перетворення цих значень у ймовірності використовується функція softmax, яка нормалізує лінійні комбінації по всіх класах, щоб вони утворили розподіл ймовірностей:

$$P(y = k|X) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}, \quad (5)$$

де $P(y = k|X)$ – ймовірність того, що приклад належить до класу k, K – загальна кількість класів.

Для навчання моделі використовується функція втрат на основі логарифмічної правдоподібності (cross-entropy loss). У випадку багатокласової класифікації вона має вигляд:

$$L = -\sum_{i=1}^{m}\sum_{k=1}^{K} y_{ik} \, logP(y = k|X_i), \quad (6)$$

де m – кількість прикладів у навчальній вибірці; K – кількість класів; y_{ik} – ознака того, чи належить приклад i до класу k: дорівнює 1, якщо так, інакше 0; $P(y = k|X_i)$ – ймовірність, яку модель присвоїла класу k для прикладу i.

Ця функція втрат заохочує модель давати високу ймовірність для правильного класу кожного прикладу. Оптимізація параметрів моделі (коефіцієнтів $\beta_j^{(k)}$) відбувається за допомогою алгоритмів градієнтного спуску, які ітеративно оновлюють ваги таким чином:

$$\beta_j^{(k)} \leftarrow \beta_j^{(k)} + \alpha \frac{\partial L}{\partial \beta_j^{(k)}}, \quad (7)$$

де α – швидкість навчання (learning rate), $\frac{\partial L}{\partial \beta_j^{(k)}}$ – часткова похідна функції втрат за параметром.

Серед ключових переваг логістичної регресії варто відзначити її простоту як у реалізації, так і в інтерпретації. Модель надає зрозумілі коефіцієнти, які можна тлумачити як вплив кожної ознаки на ймовірність належності прикладу до певного класу. Крім того, логістична регресія є досить ефективною з точки зору обчислювальних ресурсів, що дозволяє швидко тренувати модель навіть на великих обсягах даних. Її імовірнісна природа також дає змогу отримувати не лише передбачення, а й рівень

впевненості в них, що є перевагою у завданнях, де важлива інтерпретація невизначеності.

Проте логістична регресія має і низку обмежень. Вона погано працює з нелінійними залежностями між ознаками та класами, якщо не проводити додаткової інженерії ознак або не використовувати поліноміальні розширення. У випадках, коли границі між класами складні або дані не є лінійно роздільними, логістична регресія може поступатися складнішим моделям, таким як дерева рішень або ансамблеві методи.

2.2.2 Випадковий ліс

Випадковий ліс (Random Forest) – це ансамблевий метод машинного навчання, що використовується як для класифікації, так і для регресії. Він базується на концепції побудови великої кількості незалежних дерев рішень та агрегування їхніх передбачень. Для задач класифікації, як у нашому випадку, випадковий ліс приймає рішення шляхом голосування: клас, за який проголосувала більшість дерев, стає фінальним прогнозом.

Основна ідея полягає в зменшенні варіативності (перенавчання окремих дерев) шляхом поєднання кількох слабких моделей (дерев рішень) в одну сильну модель. Побудова моделі відбувається у кілька етапів:

1) З вихідного тренувального набору $D = \{(x_i, y_i)\}_{i=1}^{m}$, де x_i – векторне представлення вхідного рядка (коментаря), а y_i – мітка класу, обирається підмножина даних (бутстреп-вибірка: випадково з поверненням) для кожного дерева:$D_b \subset D, b = 1,\ldots, B$, де B – кількість дерев у лісі.
2) Для кожного дерева T_b при побудові кожного вузла випадковим чином відбирається лише підмножина ознак $m \ll n$, і серед них шукається та, яка найкраще розділяє вибірку. Це створює більшу різноманітність серед дерев.

3) Кожне дерево T_b приймає рішення самостійно. Для нового зразка ххх кожне дерево повертає прогноз $\widehat{y}_b$, після чого ліс агрегує ці прогнози. У класифікації використовується більшість голосів:

$$\widehat{y} = mode(\widehat{y}_1, \widehat{y}_2, \ldots, \widehat{y}_b) \quad (8)$$

У випадку багатокласової класифікації (як у нашому дослідженні), кожне дерево повертає один із можливих класів, а фінальне рішення приймається голосуванням серед них. Метод природно адаптується до випадків, коли кількість класів $K > 2$, і не потребує окремих модифікацій, як у логістичній регресії.

Однією з головних переваг методу є його стійкість до перенавчання завдяки ансамблю слабших моделей. Навіть якщо окреме дерево має високу дисперсію, об'єднання результатів великої кількості дерев згладжує похибки та підвищує узагальнюючу здатність. Також метод нечутливий до мультиколінеарності та масштабування ознак, що є особливо цінним для роботи з текстовими даними, де можуть бути сотні або тисячі корельованих ознак. Крім того, випадковий ліс дозволяє оцінити важливість ознак, що додає інтерпретованості моделі.

Незважаючи на свої переваги, метод має й певні обмеження. По-перше, він часто поступається у швидкості тренування і передбачення простішим моделям, особливо при великій кількості дерев та великій розмірності ознак. По-друге, у задачах, де важлива тонка інтерпретація логіки класифікації, випадковий ліс не є найкращим вибором – окремі дерева можуть бути інтерпретованими, але вся модель загалом є «чорною скринькою». Також метод має обмежену ефективність на надзвичайно розріджених даних.

2.2.3 Метод опорних векторів

Метод опорних векторів (SVM) є одним із класичних підходів до задач класифікації, який демонструє високу ефективність при роботі з високовимірними просторами ознак, зокрема у задачах текстової класифікації. Основна ідея SVM полягає у знаходженні такого гіперплощини, яка максимально розділяє приклади різних класів, забезпечуючи найбільшу можливу відстань між ними. У випадку лінійної класифікації для кожного класу модель будується як:

$$f_k(x) = w_k^T x + x_k, \quad (9)$$

де x – вектор ознак (векторизований текст), w – вектор ваг моделі, b – зсув.

Мета методу знайти таку лінію або гіперплощину, яка розділяє два класи з максимальною відстанню. Відстань вимірюється між гіперплощиною і точками, які найближчі до цієї лінії. Ці точки називаються опорними векторами (support vectors). SVM мінімізує функцію втрат, яка виглядає так:

$$min \frac{1}{2}||w||^2 + C \sum_{i=1}^{m} \xi_i, \quad (10)$$

- $||w||^2$ – це норма вектора ваг (те, що мінімізується, щоб зробити відстань великою);
- C – це параметр, який контролює, наскільки сильно модель уникатиме помилок. Якщо C велике, модель більше прагне до того, щоб не робити помилок;
- ξ_i – це помилки для кожного прикладу. Тобто, якщо точка неправильно класифікована, вона має значення $\xi_i > 0$. В ідеалі $\xi_i = 0$ для всіх точок, тобто жодна точка була класифікована правильно.

Переваги SVM полягають у тому, що він здатен ефективно працювати навіть з високовимірними даними завдяки своїй здатності знаходити оптимальну гіперплощину, яка максимізує відстань між класами. Це дозволяє SVM досягати високої точності, навіть коли дані мають складну структуру. Крім того, SVM є стійким до перенавчання, особливо при наявності великої кількості ознак. Використання ядрових методів дозволяє розв’язувати нелінійні задачі, що робить метод універсальним для різних типів класифікації.

Однак, незважаючи на свої переваги, SVM має й кілька недоліків. Перш за все, цей метод може бути обчислювально дорогим, особливо при великих обсягах даних або великій кількості класів, оскільки кожна модель потребує навчання на великій кількості прикладів. Також вибір ядра та налаштування гіперпараметрів потребують ретельного налаштування, що може ускладнювати процес застосування SVM у реальних задачах. Крім того, SVM може погано працювати, коли дані мають багато шуму або коли класи сильно перекриваються, оскільки він намагається побудувати чітку межу розподілу, що не завжди може бути оптимальним у таких випадках.

2.2.4 XGBoost

XGBoost (Extreme Gradient Boosting) – це метод, заснований на бустингу дерев рішень, де кожне нове дерево моделі намагається виправити помилки попереднього дерева. Основною ідеєю є побудова моделі, що мінімізує функцію втрат шляхом ітераційного додавання нових дерев, які коригують попередні помилки.

XGBoost використовує функцію втрат для оцінки того, наскільки добре модель передбачає результати. Загальна функція втрат комбінує два основні компоненти: функцію втрат (наприклад, логарифмічну втрату для класифікації) та регуляризацію для запобігання перенавчанню.

Загальна функція втрат виглядає так:

$$L(\Theta) = \sum_{i=1}^{n} L(y_i, \widehat{y_i}) + \sum_{k=1}^{K} \Omega(f_k), \quad (11)$$

де $L(y_i, \widehat{y_i})$ – функція втрат для кожного прикладу; $\Omega(f_k)$ – регуляризаційний термін, який штрафує модель за занадто складні дерева; n – кількість прикладів у вибірці; K – кількість дерев у моделі.

XGBoost оптимізує дерево, додаючи його до ансамблю на кожному етапі, враховуючи помилки, допущені попередніми деревами. На кожному етапі моделювання для кожного нового дерева f_k модель прагне мінімізувати функцію втрат, додавши регуляризацію для контролю за складністю дерева:

$$\widehat{y_i} = \sum_{k=1}^{K} f_k(x_i), \quad (12)$$

де $\widehat{y_i}$ – передбачене значення для прикладу i; $f_k(x_i)$ – прогноз k-го дерева для прикладу i.

Модель намагається мінімізувати різницю між передбаченням і реальним значенням за допомогою оновлення ваг дерев. Для цього використовуються методи градієнтного спуску, де оновлення ваги θ_k для кожного дерева виглядає так:

$$\theta_k^{(t+1)} = \theta_k^{(t)} - \eta * \frac{\partial L}{\partial \theta_k}, \quad (13)$$

де $\theta_k^{(t)}$ – вага дерева на кроці t; η – швидкість навчання; $\frac{\partial L}{\partial \theta_k}$ – похідна функції втрат за параметром θ_k.

Регуляризаційний термін допомагає уникнути перенавчання, штрафуючи модель за використання занадто складних дерев. Регуляризація включає два основні компоненти: кількість розгалужень дерева та величину ваг на кожному з розгалужень:

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2, \quad (14)$$

де T – кількість листів у дереві; w_j – вага кожного листа; γ та λ – параметри регуляризації, що контролюють складність дерева.

У процесі навчання XGBoost використовує градієнтний спуск для мінімізації функції втрат, що дозволяє алгоритму «шукати» оптимальні параметри для кожного дерева. Формула для оновлення на кожній ітерації виглядає так:

$$f_k(x) \leftarrow f_k(x) - \eta * \nabla L(y, f_k(x)), \quad (15)$$

де $\nabla L(y, f_k(x))$ – градієнт функції втрат для k-го дерева на етапі навчання.

Алгоритм XGBoost ґрунтується на принципі бустингу, де моделі тренуються по черзі, і кожен новий модель вчиться на тих прикладах, де попередні моделі припустилися помилок. Це дозволяє моделі виявляти складні взаємозв'язки між ознаками та класами, що робить її дуже потужною для класифікації тексту. Окрім цього, XGBoost має кілька ключових характеристик, таких як регуляризація, яка допомагає уникати перенавчання, і здатність працювати з великими обсягами даних завдяки паралельному обчисленню.

Однак, попри свою високу ефективність, XGBoost має деякі недоліки. Оскільки він є складним методом ансамблевого навчання, параметри моделі потребують ретельного налаштування для досягнення оптимальних результатів. Крім того, XGBoost може бути обчислювально важким для великих обсягів даних і великих моделей, що може збільшити час навчання. Також варто зазначити, що XGBoost може бути менш інтерпретованим, ніж простіші методи, такі як логістична регресія, оскільки складність моделі може ускладнити розуміння важливості окремих ознак для передбачень.

2.3 Метрики оцінки якості моделей

Оцінювання якості моделей класифікації є ключовим етапом у процесі побудови машинного навчання. Для цього використовують набір стандартних метрик, що дозволяють кількісно оцінити, наскільки добре модель справляється з передбаченням класів. У цьому розділі розглянемо основні поняття, пов’язані з оцінкою класифікатора, такі як false positive, false negative, true positive, true negative, а також визначимо ключові метрики для оцінки якості моделі класифікації.

- False Positive (FP): кількість прикладів, що належать до певного класу k і були правильно класифіковані як клас k;
- False Negative (FN): кількість прикладів, що належать до інших класів, але були помилково класифіковані як клас k;
- True Positive (TP): кількість прикладів, що належать до класу k, але були помилково класифіковані як інший клас;
- True Negative (TN): кількість прикладів, що не належать до класу k і були правильно класифіковані як не k.

Матриця помилок (Confusion Matrix) є інструментом для візуалізації результатів класифікації. Вона подається у вигляді таблиці, де горизонтальна вісь відповідає фактичним міткам класу, а вертикальна вісь – прогнозованим. Вона включає значення FP, FN, TP і TN.

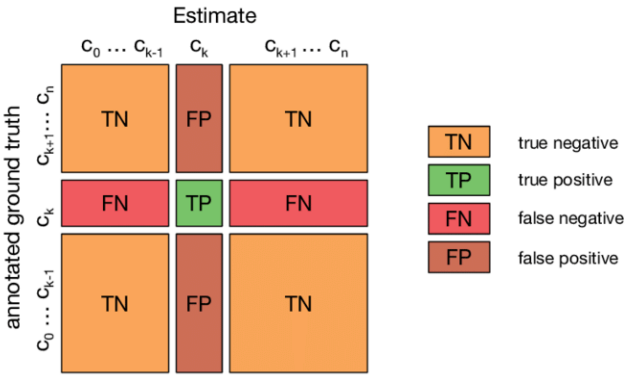


Рисунок 2.1. Матриця похибок для мультикласової класифікації

Точність (Accuracy) є метрикою, яка визначає співвідношення правильно класифікованих прикладів (TP + TN) до загальної кількості прикладів у тестовій вибірці. Ця метрика відображає загальну точність класифікатора. Формула для обчислення точності (Accuracy) виглядає наступним чином:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}, \quad (16)$$

Точність дозволяє оцінити загальну ефективність класифікатора, показуючи, який відсоток прикладів він правильно класифікує з усіх прикладів у тестовій вибірці. Вона широко використовується для оцінки якості моделі класифікації, але її варто поєднувати з іншими метриками, особливо за наявності незбалансованої вибірка даних із нерівномірним розподілом класів.

Влучність (Precision) є метрикою, яка вимірює серед усіх передбачень для певного класу, яка частка справді належить цьому класу. Це важлива метрика в задачах із незбалансованими даними. Формула для обчислення виглядає наступним чином:

$$Precision_k = \frac{TP_k}{TP_k+FP_k}, \quad (17)$$

деTP_k(True Positives) – кількість прикладів, які насправді належать до класуkі були правильно класифіковані.FP_k(False Positives) – кількість прикладів, які насправді належать до інших класів, але були помилково класифіковані як класk.

Чутливість або повнота (Recall) – це метрика, яка показує, наскільки добре модель знаходить усі приклади певного класу. Інакше кажучи: яку частку реальних прикладів класу k модель правильно розпізнала як клас k. Розраховується за формулою:

$$Recall_k = \frac{TP_k}{TP_k+FN_k}, \quad (18)$$

деTP_k(True Positives) – кількість прикладів, які належать до класуkі були правильно класифіковані.FN_k(False Negatives) – кількість прикладів, які насправді належать до класуk, але були класифіковані як інші класи.

У мультикласовій класифікації розрахунок precision та recall відбувається окремо для кожного класу. Щоб отримати загальне уявлення про метрику, можна обчислити його в один із трьох способів:

- Macro average: середнє значення для всіх класів (усі класи рівнозначні).
- Weighted average: середнє precision/recall з урахуванням кількості прикладів у кожному класі.
- Micro average: об’єднує всі TP і FP по всіх класах перед обчисленням.

F1-міра є комбінованою метрикою, яка поєднує precision та recall для оцінки ефективності класифікатора. Вона надає узагальнений показник, який враховує як якість правильних прогнозів, так і здатність класифікатора знаходити всі позитивні приклади. F1 обчислюється на основі гармонічного середнього між точністю і чутливістю, і виражається за наступною формулою:

$$F1_k = 2 * \frac{Precision_k * Recall_k}{Precision_k + Recall_k}, \quad (19)$$

Іноді precision і recall мають компроміс між собою: модель або добре знаходить усі приклади класу (високий recall), але помиляється з іншими (низький precision), або навпаки. F1 дозволяє знайти збалансовану оцінку, коли важливі обидва аспекти. F1-міра особливо корисна коли важливо зменшити і FP, і FN або коли одна з метрик (precision чи recall) занадто низька, і потрібно враховувати обидві.

2.4 Використовувані програмні засоби

2.4.1 Опис мови програмування

Python – мова програмування, яка набула широкої популярності завдяки своїм унікальним перевагам та простоті використання. Однією з головних переваг Python є його читабельний та зрозумілий синтаксис, який дозволяє легко розробляти та розуміти код. Це робить Python відмінним вибором для широкого кола розробників, від початківців до досвідчених фахівців.

Крім того, Python має величезне екосистему бібліотек та фреймворків для різних галузей розробки, включаючи аналіз даних, машинне навчання, обробку природної мови та багато іншого. Наприклад, бібліотеки, такі як scikit-learn, TensorFlow, PyTorch та NLTK, надають потужні засоби для розв’язання складних завдань, таких як класифікація тексту. Саме тому Python є ідеальним вибором для вирішення поставленої в роботі задачі.

Незважаючи на свої переваги, Python має деякі недоліки, зокрема обмежену швидкодію в порівнянні з деякими іншими мовами програмування. Однак у більшості випадків це не становить проблеми. Також, дана мова має обмежені можливості для розробки додатків реального часу.

2.4.2 Бібліотеки для роботи з даними та тренуванням моделей

Pandas – це бібліотека для аналізу даних у Python. Її особливістю є структура даних DataFrame, яка представляє собою таблицю з рядками та колонками. Вона має широкий спектр можливостей маніпуляцій даними, такі як зміна та фільтрація даних, додавання/видалення рядків та стовпців, об’єднання таблиць тощо. Ще одна корисна функція бібліотеки – читання та запис даних: Pandas дозволяє зчитувати дані з різних джерел та форматів, включаючи CSV, Excel, SQL, JSON та інших. Не менш важливою

функцією є аналіз даних. Він надає засоби для обчислення статистичних показників, групування даних та побудови звітів.

NumPy – це бібліотека для наукових обчислень у Python. Вона надає підтримку для роботи з масивами та математичними функціями, дозволяють зберігати та обробляти дані в багатовимірних масивах; підтримує операції лінійної алгебри, такі як матричні операції, обертання матриць, знаходження власних значень та векторів тощо.

Scikit-learn – це бібліотека машинного навчання для Python. Вона надає інструменти для побудови моделей класифікації, регресії, кластеризації. Scikit-learn містить широкий вибір алгоритмів машинного навчання, таких як метод опорних векторів (SVM), логістична регресія, випадковий ліс, нейронні мережі та інші. Бібліотека також пропонує інструменти для попередньої обробки даних, таких як нормалізація, видалення аномалій, кодування категоріальних змінних та створення різних пайплайнів для підготовки даних до моделі. Ще однією важливою функцією є можливість оцінювати точність та ефективність моделей за допомогою різних метрик та методів перехресної перевірки.

Ці бібліотеки утворюють основу для аналізу даних та розробки моделей машинного навчання у Python, і вони часто використовуються в проектах з обробки даних та штучного інтелекту.

2.4.3 Бібліотеки для візуалізації

Matplotlib – це бібліотека для створення статичних, анімованих та інтерактивних візуалізацій у Python. Найпопулярнішим підмодулем є pyplot, який надає високорівневий інтерфейс для створення графіків. Він має велике різноманіття можливостей. Найголовнішою є підтримка різних видів візуалізацій: лінійні графіки, гістограми, діаграми розсіювання, гістограми та інші типи. Крім того, бібліотека включає можливості налаштування осей, маркерів, ліній, легенд, анотацій та багатьох інших

елементів графіків. Також є можливість створення інтерактивних графіків за допомогою бібліотек, таких як mpl_toolkits. Не менш корисною функцією є підтримка збереження графіків у різних форматах, таких як PNG, PDF, SVG.

Plotly – це потужна бібліотека для створення інтерактивних візуалізацій у Python. Вона дозволяє легко будувати графіки, які реагують на дії користувача – масштабування, наведення курсору, фільтрацію тощо. Бібліотека підтримує широкий спектр типів графіків: лінійні графіки, діаграми розсіювання, гістограми, коробчасті діаграми, теплові карти, карти світу, 3D-графіки та багато інших. Основним модулем є plotly.express, який забезпечує простий і швидкий спосіб створення поширених типів графіків. Plotly дозволяє зберігати графіки у форматах HTML (для подальшого вбудовування у веб-сторінки), PNG, SVG та PDF.

Seaborn – це бібліотека для візуалізації даних на основі matplotlib. Вона надає високорівневий інтерфейс для створення привабливих та інформативних статистичних графіків. Основні можливості: підтримка статистичних графіків, таких як графіки розподілу, парні графіки, коробкові діаграми, лінійні графіки, теплові карти тощо; інтеграція з Pandas, що дозволяє працювати безпосередньо з Pandas DataFrame та спрощує процес візуалізації даних; можливість вибору різних стилів та кольорових тем для графіків; зручність для створення комплексних візуалізацій, таких як факторні графіки, графіки регресії, матричні графіки тощо.

WordCloud – це бібліотека для створення хмар слів із текстових даних. Вона допомагає візуалізувати найбільш часто використовувані слова в тексті, де розмір слова відповідає його частоті в тексті. Основні можливості бібліотеки включають генерацію хмар слів, налаштування зовнішнього вигляду, фільтрація слів, що дає можливість виключити деякі слова зі списку частот або використовувати маски для створення хмар слів

у заданих формах, і сумісність з іншими бібліотеками для подальшої візуалізації.

2.4.4 Опис середовища

Для написання коду в роботі використовувалось інтерактивне середовище програмування Google Colab. Це хмарний сервіс, який не потребує встановлення програмного забезпечення на локальний комп'ютер, що робить його зручним і доступним для роботи з Python-кодом. Colab дозволяє поєднувати код, текстові пояснення, візуалізації та результати виконання в єдиному документі, що особливо зручно для експериментів і досліджень.

Серед основних переваг Google Colab безкоштовний доступ до GPU/TPU, що дає можливість прискорити обробку великих обсягів текстових даних або тренування моделей машинного навчання. Також важливою є інтеграція з Google Drive, яка забезпечує зручне зберігання ноутбуків та оброблених даних у хмарі. Крім того, Google Colab дозволяє встановлювати необхідні бібліотеки безпосередньо в середовищі виконання.

2.4.5 API

Google Translate API – це інструмент для автоматизованого перекладу текстів, який надає доступ до сервісу Google Перекладача через програмний інтерфейс. API підтримує понад 100 мов, дозволяє визначати мову вхідного тексту, перекладати окремі слова, речення або великі обсяги текстових даних, і легко інтегрується з Python або іншими мовами за допомогою HTTP-запитів чи відповідних бібліотек.

SurveyMonkey API – це програмний інтерфейс, який дозволяє отримувати доступ до даних опитувань, створених у сервісі SurveyMonkey. API надає можливість автоматизовано завантажувати відповіді респондентів, переглядати структуру опитувань, питання, метадані, а

також створювати, оновлювати або видаляти опитування. Інтерфейс підтримує автентифікацію через OAuth 2.0 і може використовуватись у Python та інших мовах програмування для інтеграції результатів опитувань у аналітичні або звітні системи.

Висновки

Таким чином, проведений огляд методів попередньої обробки тексту та векторизації підкреслює важливість підготовчого етапу для забезпечення якісного представлення текстових даних, необхідного для ефективної роботи алгоритмів класифікації. Розглянуті методи класифікації, від простих і інтерпретованих (логістична регресія) до більш складних ансамблевих підходів (випадковий ліс, XGBoost) та методів, що ефективно працюють у високовимірних просторах (SVM), утворюють набір інструментів, які будуть застосовані для вирішення поставленої задачі класифікації.

Визначені метрики оцінки якості моделей є ключовими для об’єктивного порівняння ефективності обраних алгоритмів та вибору найкращої моделі для конкретного завдання. Використання різноманітних метрик дозволить отримати всебічне розуміння здатності моделі правильно класифікувати текстові дані, враховуючи можливу незбалансованість класів.

Нарешті, опис використаних програмних засобів демонструє вибір потужних та гнучких інструментів, що забезпечують необхідну функціональність для реалізації всіх етапів дослідження – від обробки та аналізу даних до побудови та оцінки моделей машинного навчання.

У наступних розділах буде представлено експериментальну частину роботи, де описані методи будуть застосовані до конкретного набору текстових даних, а їхня ефективність буде оцінена за допомогою обраних метрик.

РОЗДІЛ 3
ПОБУДОВА ТА ОЦІНКА МОДЕЛЕЙ КЛАСИФІКАЦІЇ КОМЕНТАРІВ

3.1 Підготовка та попередній аналіз даних

3.1.1 Збір та опис даних

Перед початком моделювання дані необхідно зібрати, впорядкувати та підготувати до подальшої обробки. У цьому дослідженні джерелом текстової інформації виступають відповіді користувачів із регулярного опитування, яке проводиться на платформі SurveyMonkey. За допомогою SurveyMonkey API було здійснено витяг необхідних відкритих текстових відповідей. Отримані дані були збережені в табличному форматі з урахуванням метаінформації (ідентифікаторів опитувань, дат, типів питань тощо), що дозволило структурувати корпус для подальшого аналізу.

У дослідженні береться не вся інформація з опитувань, а тільки її частина. Основна таблиця містить такі поля:

- l1 – основна товарна категорія на платформі OLX, у якій користувач побачив опитування;
- period – період, у якому проводилось опитування;
- q1 – відповідь на запитання «Чи задоволені ви якістю контенту?» з варіантами: «Так», «Майже так», «Ні»;
- q5 – відкрита текстова відповідь користувача, що містить пропозиції, зауваження або скарги;
- comment_category – вручну присвоєна категорія, до якої належить зміст коментаря (наприклад, модераційні пропозиції, не формат, похвала від клієнта, тощо).

Після первинного огляду було встановлено, що датасет містить 17777 рядків і не має пропущених значень, за винятком колонки comment_category. Такий вигляд мали початкові дані:

	l1	period	q1	q5	comment_category
11231	Тварини	2024-12-01	Так	Все добре	NaN
338	Дитячий світ	2024-01-12	Так	немаю	Похвала від клієнта
7840	Робота	2024-03-01	Ні	Номера в поданих оголошеннях не працюють\n	NaN
10775	Бізнес та послуги	2024-03-01	Майже так	Наберіть компетентних модераторів. На однакові...	NaN
15295	Запчастини для транспорту	2023-06-01	Так	Все добре! Так тримати	NaN
6665	Робота	2024-03-01	Ні	Меньше рекламы а то мешает поиску работы.	NaN
14919	Нерухомість	2023-04-01	Ні	Дайте №-тел. МОРДЕраторів-адміністраторів,які ...	NaN
760	Електроніка	2024-01-12	Так	Заблокувати шахраїв які в назві профілю вказую...	Шахраї
14504	Робота	2023-04-01	Майже так	ни	NaN
14347	Запчастини для транспорту	2023-12-01	Майже так	Будьте честными.	NaN

Рисунок 3.1. 10 випадкових прикладів із початкового набору даних

Оскільки деякі відкриті відповіді на запитання q5 були на російській мові, для уніфікації та спрощення подальшої обробки всі тексти було приведено до однієї мови – української. Для цього використовувався Google Translate API (для мови програмування python використовувалась бібліотека googletrans) [28], який дозволяє автоматично визначити мову тексту та виконати переклад. Це забезпечило одномовність корпусу даних, що є важливим для якісної токенізації, нормалізації та подальшого аналізу. У додатку А наведено приклад коду, що здійснює переклад коментарів. У результаті його виконання було сформовано наступний датасет:

	q5	q5_translated	comment_category	language_origin
13855	Будьте здорові	Будьте здорові	NaN	ua
11582	Дуже довго йде ОЛХ доставка. Відправили 5 груд...	Дуже довго йде ОЛХ доставка. Відправили 5 груд...	NaN	ua
13226	Нет	Ні	NaN	ru
1660	Ні	Ні	Похвала від клієнта	ua
14457	зробіть менше реклами.	зробіть менше реклами.	NaN	ua
17063	Не маеш змоги чомусь поставити відгук якщо про...	Не маеш змоги чомусь поставити відгук якщо про...	NaN	ua
15410	необхідно вам наладити якість відправлення пов...	необхідно вам наладити якість відправлення пов...	NaN	ua
6741	Сообщать, когда обьявление уже снято и не стои...	Повідомити, коли оголошення вже видалено і не ...	NaN	ru
2667	много смс на вайбей или вацап, когда что то пр...	Багато SMS для Vaiber або Vatsap, коли ви прод...	NaN	ru
16471	Ні	Ні	NaN	ua

Рисунок 3.2. 10 випадкових прикладів із датасету після перекладу

3.1.2 Попередній аналіз даних

Для кращого розуміння структури та змісту наявних даних до початку моделювання було проведено їх первинний аналіз із візуалізацією. Такий підхід дозволяє виявити ключові закономірності, розподіли відповідей, особливості по категоріях та потенційні дисбаланси в даних.

Першим кроком у візуалізації стало дослідження ступеня заповненості міток у наборі даних. На круговій діаграмі на рисунку 3.3 зображено розподіл обсягу розмічених та нерозмічених даних. Як видно з графіка, переважна більшість коментарів – 89.6 % залишаються без визначеної категорії, тоді як лише 10.4 % мають відповідні мітки в колонці comment_category. Така ситуація підкреслює актуальність використання підходів до моделювання з обмеженою кількістю розмітки, зокрема напівконтрольованих методів навчання.

Рисунок 3.3 Розподіл розмічених та нерозмічених коментарів

Наступним графіком є візуалізація розподілу мов у вихідних коментарях до їх перекладу на українську мову. Як показує діаграма на рисунку 3.4, більшість коментарів (78.7 %) була написана українською мовою (ua), що свідчить про домінування цієї мови серед користувачів. Водночас, 21.3 % коментарів були написані російською мовою (ru). Цей

графік підтверджує важливість процесу перекладу, здійсненого для уніфікації текстових даних і забезпечення єдиного лінгвістичного контексту для подальшого аналізу.

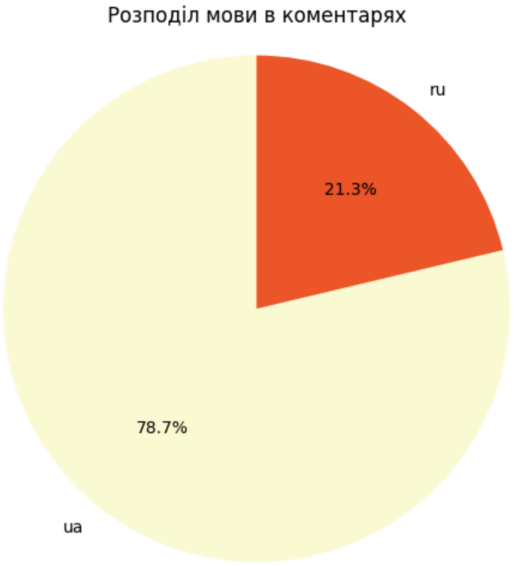


Рисунок 3.4 Розподіл мови в коментарях

Також важливо дослідити розподіл розмічених коментарів за категоріями (класами). Тому ще одним графіком (рисунок 3.5) є стовпчаста діаграма, що відображає розподіл розмічених коментарів за категоріями (класами). Вісь X на графіку показує різні категорії коментарів, а вісь Y демонструє кількість коментарів, що належать до кожної із цих категорій.

З графіку видно, що найбільшу кількість розмічених коментарів становить категорія «Похвала від клієнта», з понад 700 коментарями. За нею йде категорія «Продуктові проблеми/пропозиції», що містить майже 600 коментарів. У категоріях «Не формат» та «Модераційні пропозиції» спостерігається значно менша кількість коментарів, кожна з яких налічує близько 200–250 коментарів. Найменшу кількість коментарів має категорія «Шахраї» з менше ніж 100 коментарями.

Цей графік демонструє значну нерівномірність у розподілі розмічених даних за категоріями, з переважанням похвали та пропозицій щодо продукту. Така нерівномірність є важливим аспектом при навчанні

моделей класифікації, оскільки вона може впливати на точність передбачень для менших категорій.

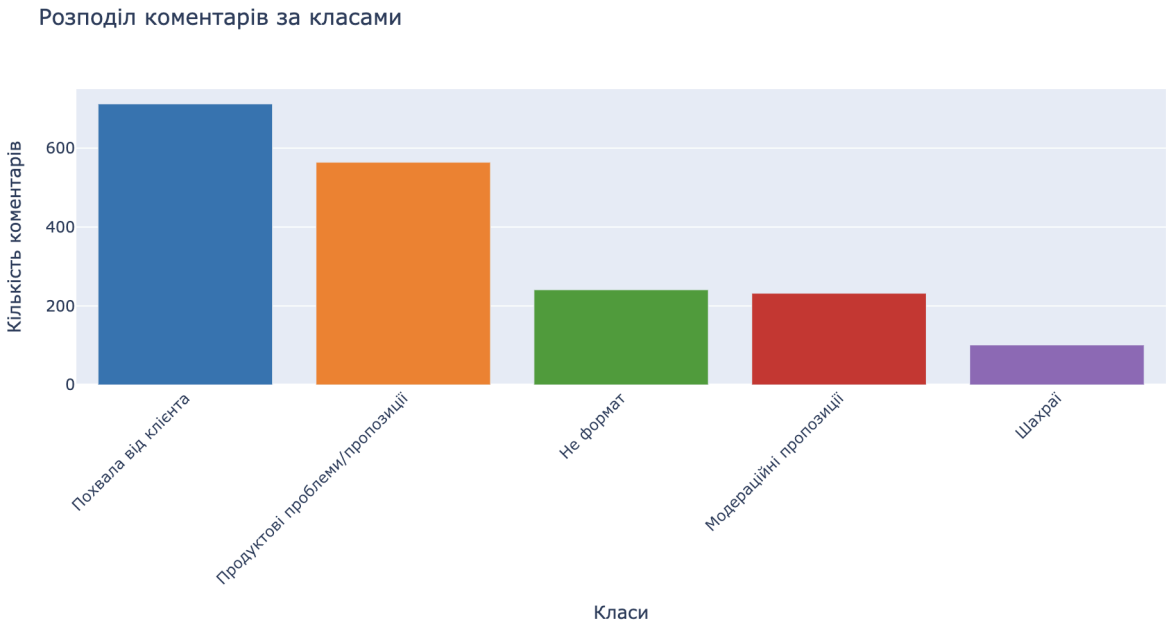


Рисунок 3.5 Розподіл коментарів за класами

Стовпчастий графік на рисунку 3.6 відображає розподіл усіх коментарів (як розмічених, так і нерозмічених) за основними товарними категоріями (всього 11) на платформі OLX, у яких користувачі залишали свої відгуки в опитуванні. Вісь X показує назви категорій, а вісь Y – загальну кількість коментарів, отриманих у кожній із них.

Основні спостереження з графіка показують, що найбільша кількість коментарів пов’язана з категорією «Електроніка». Також значна кількість відгуків надійшла щодо категорій «Хобі, відпочинок і спорт» та «Дім і сад». Кількість коментарів у категоріях «Бізнес та послуги», «Мода і стиль», «Нерухомість» та «Запчастини для транспорту» є приблизно однаковою і дещо меншою, ніж у перших трьох. Найменша кількість коментарів зібрана в категоріях «Дитячий світ», «Транспорт», «Тварини» та «Робота». Це свідчить про різну кількість оголошень або активності в цих категоріях на платформі.

Рисунок 3.6 Розподіл коментарів за категоріями на сайті

Далі проаналізуємо, які користувачі більш схильні залишати коментарі залежно від рівня їх задоволеності якістю контенту. На рисунку 3.7 стовпчастий графік відображає порівняння кількості коментарів для кожного варіанту відповіді на запитання q1: «Так», «Майже так» та «Ні». Вісь X показує варіанти відповідей, а вісь Y відображає кількість користувачів, які обрали кожен із цих варіантів і залишили коментар.

Графік показує, що користувачі, які задоволені якістю контенту є найбільш активними в наданні відгуків. Однак помітна кількість коментарів надійшла також від тих, хто частково задоволений, хоча їх кількість значно менша. Найменша кількість коментарів була залишена незадоволеними користувачами. Це вказує на те, що користувачі з більш позитивним ставленням до контенту більш схильні надавати зворотній зв’язок, тоді як менш задоволені користувачі залишають коментарі рідше, але все ж надають цінну інформацію для аналізу.

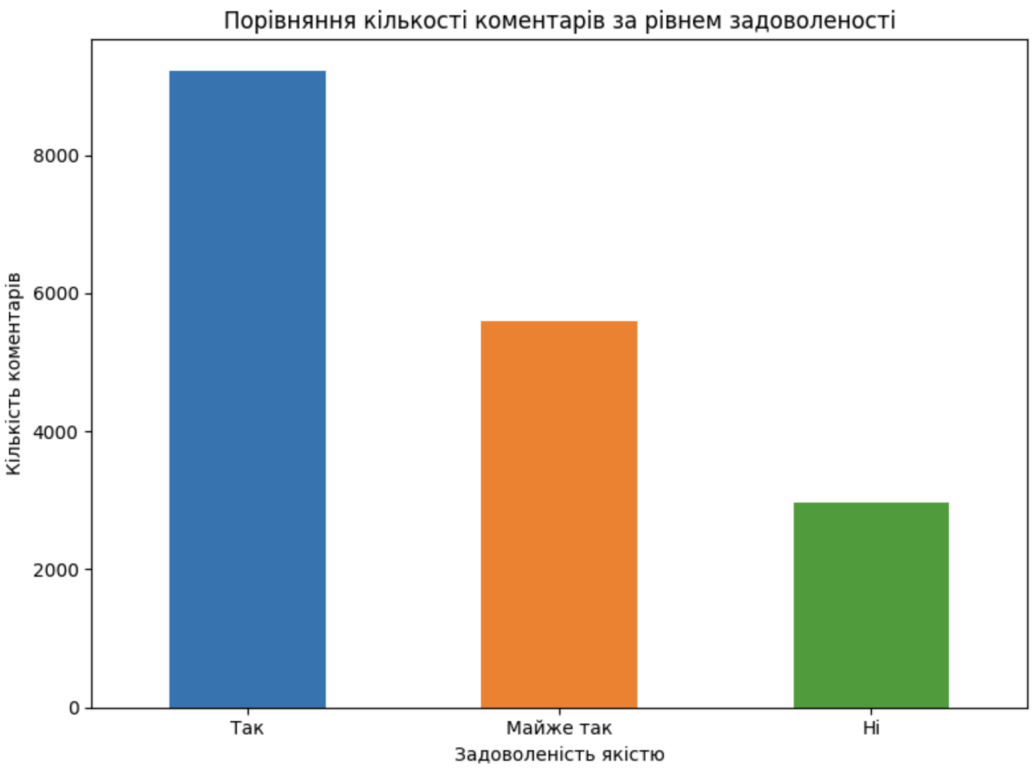


Рисунок 3.7 Порівняння кількості коментарів за рівнем задоволеності

Наступним етапом аналізу є розгляд теплової карти на рисунку 3.8, яка деталізує зв’язок між рівнем задоволеності користувачів та категоріями коментарів. Якщо попередній графік показував загальний розподіл коментарів за відповідями на питання про якість контенту, то тепер ми можемо побачити, які саме типи зворотного зв’язку найчастіше зустрічаються в кожній із груп користувачів.

Карта чітко демонструє, що найбільше позитивних коментарів («Похвала від клієнта») надходить від задоволених користувачів, що є очікуваним. Тим часом частково задоволені й незадоволені користувачі частіше залишають відгуки, пов’язані з продуктовими проблемами або пропозиціями, що свідчить про їхню зацікавленість у покращенні сервісу. Коментарі, пов’язані з модерацією, неформатним вмістом чи шахраями, залишаються менш чисельними у всіх групах, проте вони можуть містити критично важливу інформацію.

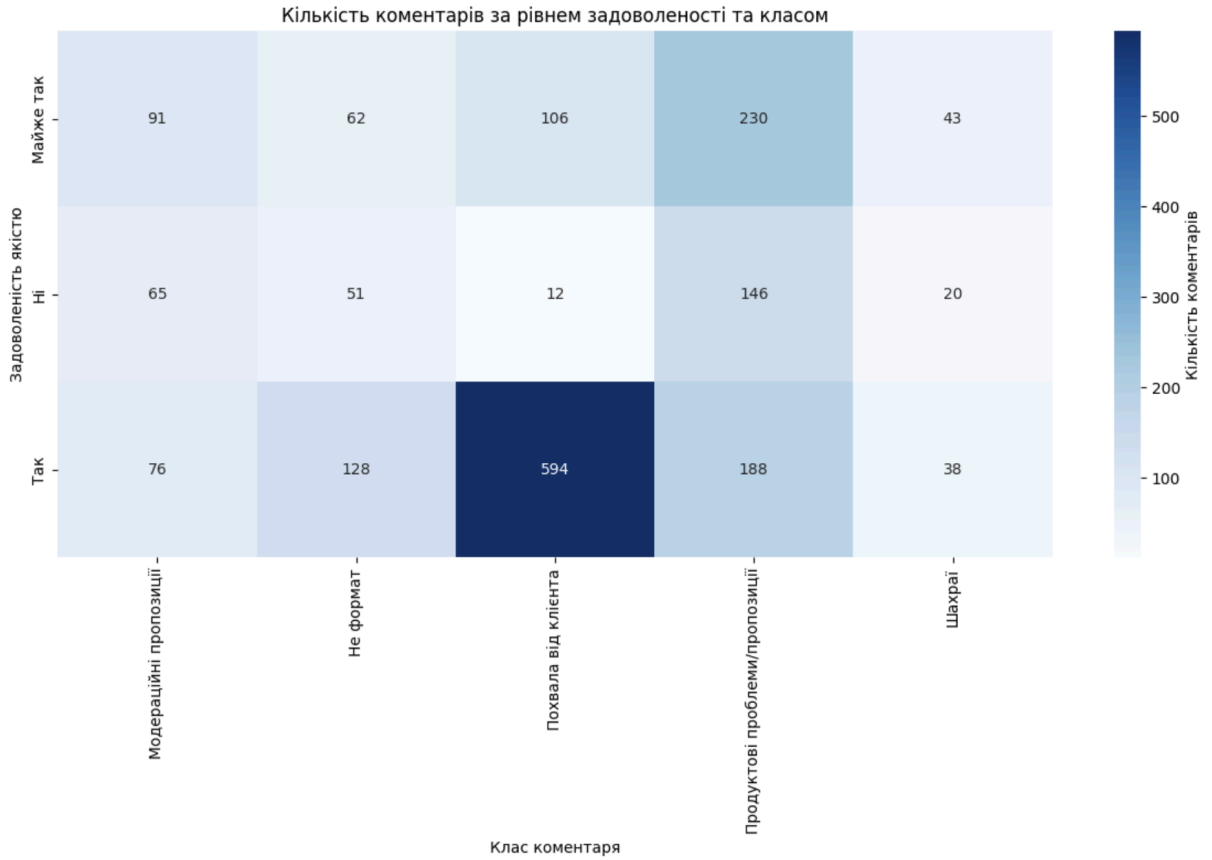


Рисунок 3.8 Кількість коментарів за рівнем задоволеності та класом

Графік на рисунку 3.9 логічно доповнює попередній аналіз, додаючи вимір глибини коментарів залежно від рівня задоволеності користувачів. З нього видно, що чим нижчий рівень задоволеності, тим детальніші відгуки залишають користувачі. Найдовші коментарі в тих, хто не задоволений сервісом, що вказує на потребу докладно викласти причини свого невдоволення. Водночас задоволені користувачі зазвичай обмежуються короткими схвальними фразами. Ця закономірність може бути корисною при подальшому аналізі змісту відгуків: довші тексти частіше містять цінні для вдосконалення інсайти, тоді як коротші слугують загальним індикатором позитивного досвіду.

Рисунок 3.9 Середня довжина коментаря в словах за класом задоволеності

Стовпчаста діаграма на рисунку 3.10 ілюструє, як тематика коментаря впливає на його обсяг. Найдовші, найдетальніші тексти спостерігаються в категоріях «Продуктові проблеми/пропозиції» та «Модераційні пропозиції» – це логічно, адже користувачі прагнуть пояснити суть проблем або навести аргументи щодо змін. У коментарях про «Шахраїв» середня довжина також відносно висока, що, ймовірно, пов’язано з описом ситуацій чи підозрілих дій. Натомість категорії «Похвала від клієнта» й «Не формат» містять найлаконічніші повідомлення: у першому випадку це короткі позитивні оцінки, а в другому – технічні або формальні позначки.

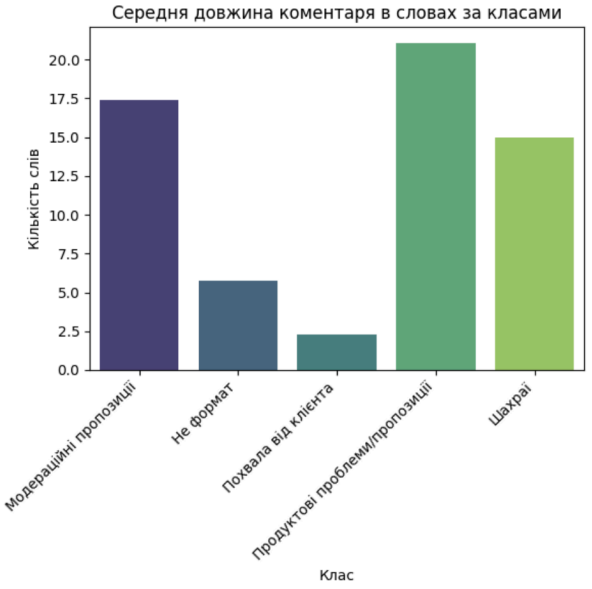


Рисунок 3.10 Середня довжина коментаря в словах за класами

Наступні три горизонтальні стовпчасті діаграми демонструють найпоширеніші пари слів (біграми) у коментарях користувачів для трьох ключових категорій: «Продуктові проблеми/пропозиції» (рисунок 3.11), «Шахраї» (рисунок 3.12) та «Модераційні пропозиції» (рисунок 3.13). Вони допомагають зрозуміти, які теми найчастіше згадуються в кожному типі звернень.

- Продуктові проблеми/пропозиції: Найпопулярніша біграма – «олх доставка», що свідчить про увагу до сервісу доставки. Інші часті згадки стосуються оголошень, телефонних номерів та взаємодії між продавцями й покупцями.
- Шахраї: Основні біграми, як «номер телефон», «заблокувати шахраїв», «низький ціна», вказують на занепокоєння користувачів через шахрайські дії та необхідність їх блокування.
- Модераційні пропозиції: Тут домінують слова на кшталт «різний місто», «якість фото», «перевіряти оголошення». Це говорить про побажання посилити модерацію, покращити контроль за якістю контенту та впровадити фільтри.

Ці діаграми допомагають краще зрозуміти специфіку запитів у межах кожної категорії і можуть слугувати орієнтиром для подальших покращень платформи. Такий аналіз особливо цінний у перспективі – його доцільно повторити після автоматичного розмічування повного масиву коментарів за допомогою обраної моделі. Це дозволить отримати масштабнішу та точнішу картину найбільш поширених тем і проблем, які турбують користувачів.

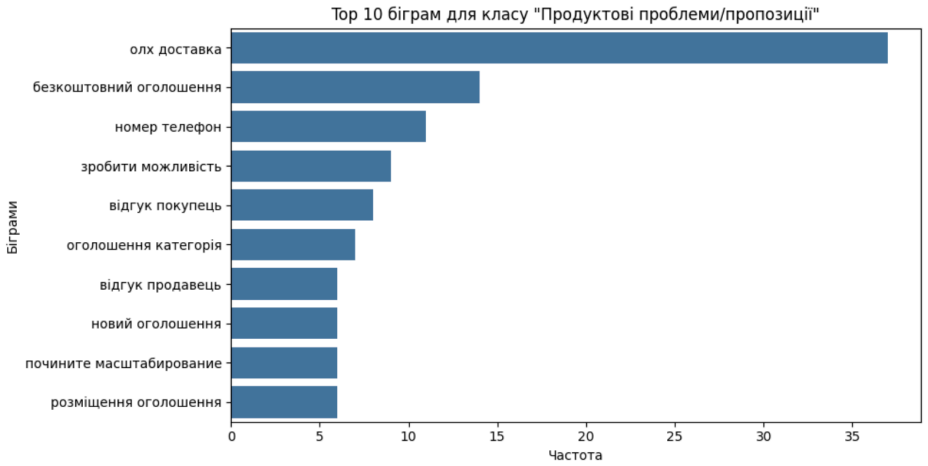


Рисунок 3.11 Топ-10 біграм для класу «Продуктові проблеми/пропозиції»

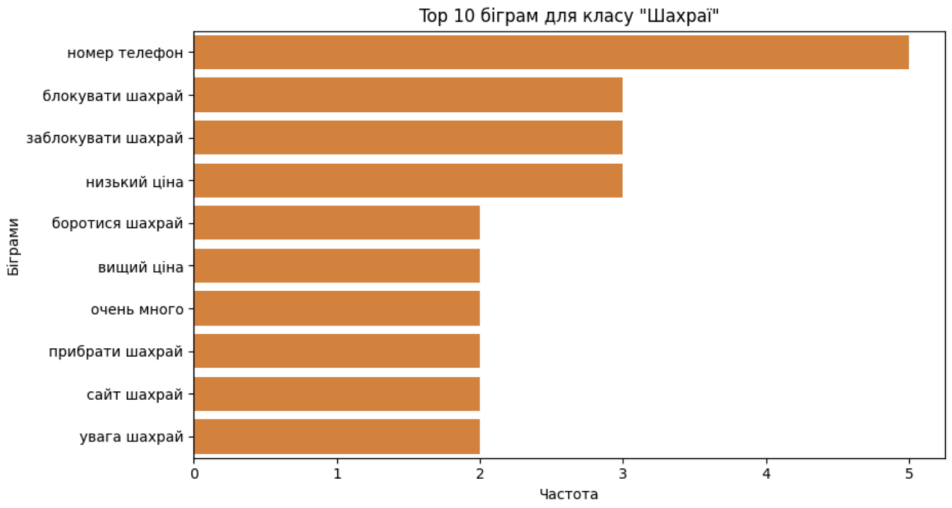


Рисунок 3.12 Топ-10 біграм для класу «Шахраї»

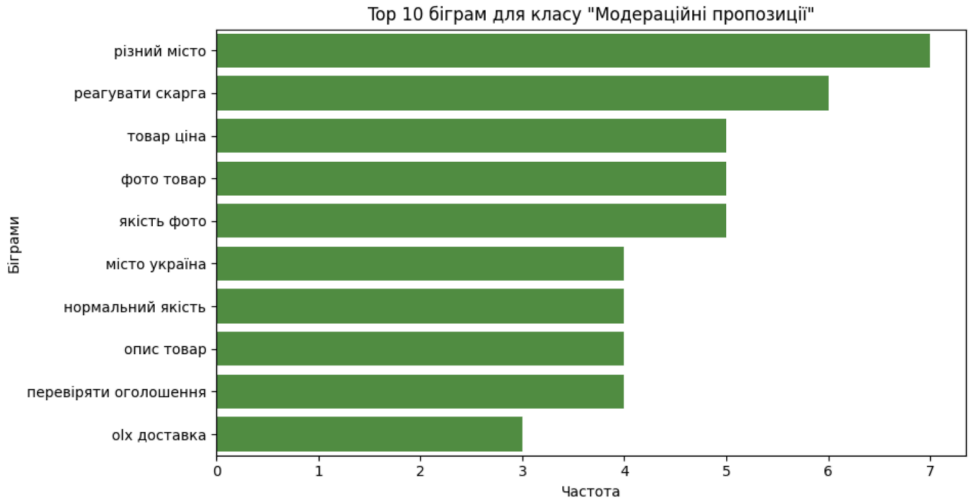


Рисунок 3.13 Топ-10 біграм для класу «Модераційні пропозиції»

Остання візуалізація на рисунку 3.14 ілюструє найчастотніші слова, які зустрічаються в усіх текстових коментарях набору даних після їх перекладу на українську мову. Розмір кожного слова в хмарі пропорційний його частоті в текстах: чим більше слово, тим частіше воно зустрічається.

Найбільші слова в центрі, такі як «оголошення», «продавець», «ціна», «товар», «доставка», «зробити», «писати», «робота», «можливість», «категорія», «фото», «дякую», акцентують увагу на ключових аспектах взаємодії з платформою: від розміщення оголошень і роботи продавців до вартості товарів, функціональності сайту і вдячності за позитивний досвід.

Менш помітні, але все ще значущі слова: «покупець», «відгук», «скарга», «телефон», «пошук», «місто», «шахрай», «фільтр», «неможливо» – допомагають розширити розуміння контексту зворотного зв’язку. Наприклад, згадування шахраїв підтверджує релевантність цієї проблеми, що вже фігурувала в інших візуалізаціях.

Загалом, хмара слів надає зручний візуальний огляд лексики, яка найчастіше використовується у відгуках користувачів. Вона дозволяє виокремити основні напрями зворотного зв’язку – від функціональних і технічних зауважень до позитивних емоцій та критичних тем, як-от безпека й довіра до сервісу.

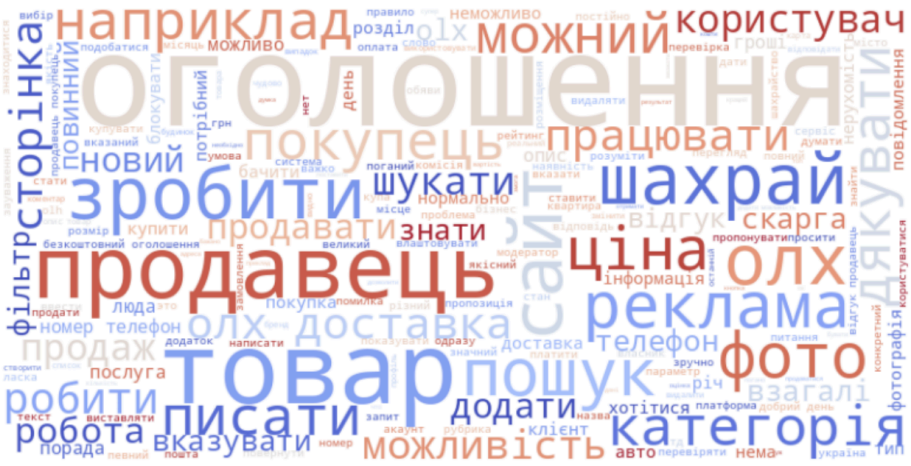


Рисунок 3.14 Хмара слів для всіх коментарів

3.2 Підготовка даних та побудова моделей класифікації

3.2.1 Попередня підготовка тексту

Перш ніж переходити до побудови моделі класифікації, необхідно здійснити базову обробку текстових даних. Коментарі користувачів можуть містити зайві символи, неоднорідні форми слів та інші елементи, які ускладнюють подальший аналіз. Тому на першому етапі текст було приведено до нижнього регістру, очищено від небуквених символів та зайвих пробілів за допомогою функції clean_text, як на рисунку 3.15. Це дозволило стандартизувати коментарі та уникнути шуму, пов'язаного з пунктуацією, великими літерами та неінформативними символами.

```
def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Za-яA-ЯіїєґІЇЄҐ\s]', '', text)
    text = re.sub(r'\n\s+', ' ', text).strip()
    return text
```

Рисунок 3.15 Функція очищення тексту

Для потреб візуального аналізу даних (наприклад, побудови хмар слів або частотних графіків) додатково виконувались видалення стоп-слів та лематизація. Це дозволило зосередитись на смислово значущій лексиці в коментарях. Однак у подальшому моделюванні ці етапи не застосовувались, оскільки вони призводили до погіршення точності класифікації.

Список українських стоп-слів було завантажено з відкритого репозиторію GitHub [29], після чого застосовано функції delete_stop_words і lemmatize_text (рисунок 3.16). Перша видаляла неінформативні слова зі списку, а друга за допомогою мовної моделі spacy (uk_core_news_sm) приводила слова до їхньої базової форми. Це дало змогу виокремити найважливіші лексеми в коментарях.

```
url = 'https://raw.githubusercontent.com/skupriienko/Ukrainian-Stopwords/master/stopwords_ua.txt'
response = requests.get(url)

stopwords_text = response.text
stopwords_uk = set(word.strip() for word in stopwords_text.split('\n') if word.strip())

nlp = spacy.load("uk_core_news_sm")

def delete_stop_words(text):
    cleaned_text = ' '.join([token for token in text.split() if token not in stopwords_uk])
    return cleaned_text

# Функція для лематизації
def lemmatize_text(text):
    doc = nlp(text)
    cleaned_text = ' '.join([token.lemma_ for token in doc])
    return cleaned_text
```

Рисунок 3.16 Функції видалення стоп-слів та лематизації тексту

Після базового очищення тексти були перетворені у числові вектори за допомогою двох різних підходів:

- класична TF-IDF векторизації, реалізованої за допомогою TfidfVectorizer з бібліотеки Scikit-learn;

```
vectorizer = TfidfVectorizer(max_features=5000)
X_train_vect = vectorizer.fit_transform(X_train)
X_test_vect = vectorizer.transform(X_test)
```

Рисунок 3.17 Функція векторизації TF-IDF

- сучасні семантичні ембедінги на основі моделі MiniLM з використанням моделі paraphrase-multilingual-MiniLM-L12-v2 [27] із пакету sentence-transformers [26].

```
from sentence_transformers import SentenceTransformer
mini_lm_model = SentenceTransformer("paraphrase-multilingual-MiniLM-L12-v2")

X_train_minilm = mini_lm_model.encode(X_train.tolist(), show_progress_bar=True)
X_test_minilm = mini_lm_model.encode(X_test.tolist(), show_progress_bar=True)
```

Рисунок 3.18 Функція векторизації paraphrase-multilingual-MiniLM-L12-v2

Усі кроки можна описати діаграмою на рисунку 3.19. Схема демонструє повний процес обробки текстових даних: від базової очистки до підготовки для візуалізацій, включаючи токенізацію, видалення стоп-слів та лематизацію.

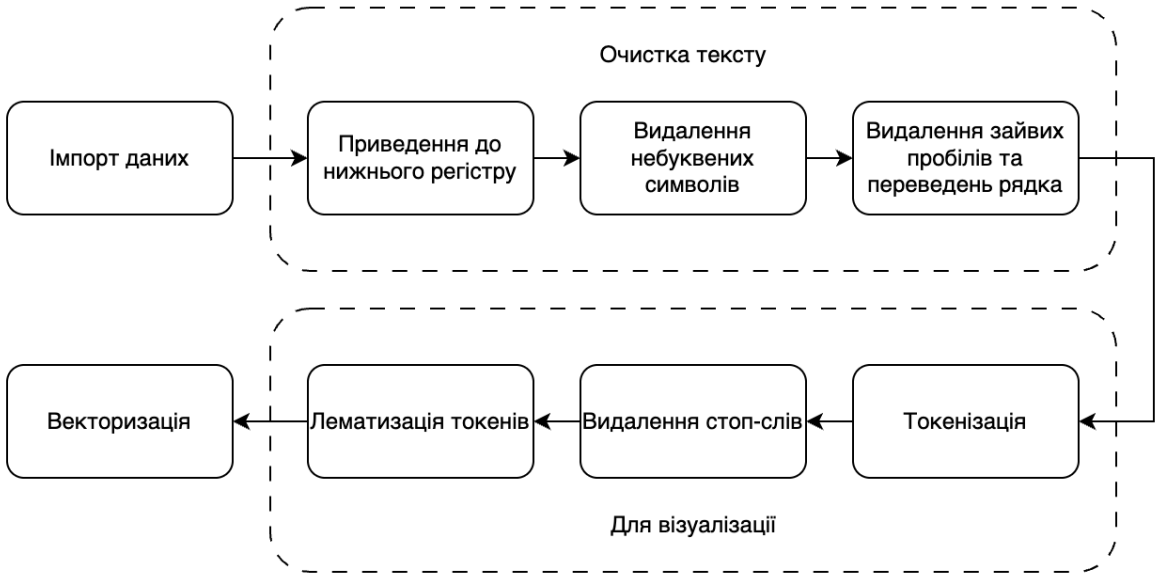


Рисунок 3.19 Схема процесу обробки текстів

3.2.2 Побудова та вдосконалення моделей класифікації

На першому етапі було побудовано базові моделі для задачі класифікації коментарів за допомогою TF-IDF-векторизації, яка переводить текстові дані у числовий формат на основі частоти термінів та їхньої інформативності. Для векторизації було використано TfidfVectorizer зі стандартними параметрами, обмеживши розмір словника до 5000 найбільш інформативних термінів (max_features=5000).

Перед тренуванням дані було розділено на тренувальну та тестову вибірки за допомогою функції train_test_split, де 80% вибірки стало тренувальним набором, а 20% – тестовим. Окрім того, параметри функції були налаштовані таким чином, щоб у тренувальній та тестовій вибірці збереглось співвідношення класів. Такий крок дозволив перевірити узагальнювальну здатність моделей на нових прикладах. Отримані вектори використовувались як вхідні дані для таких моделей:

1. Логістична регресія – проста, але ефективна модель для задач багатокласової класифікації, яка добре працює на розріджених даних і має високу інтерпретованість. Для її реалізації було використано

функцію LogisticRegression з модуля sklearn. Для компенсації можливого дисбалансу класів було використано параметр class_weight='balanced', який автоматично підбирає ваги класів обернено пропорційно до їх частоти у даних. Модель також тренувалась із підвищеним лімітом ітерацій max_iter=1000 для досягнення збіжності. Було встановлено random_state=42 для відтворюваності результатів.

2. Випадковий ліс – ансамблева модель, що об’єднує багато рішень дерев, дозволяє враховувати складні нелінійні зв’язки. Також було використано параметр class_weight='balanced' для роботи з дисбалансом класів, а n_estimators=100 визначає кількість дерев у лісі. random_state=42 задає фіксоване зерно випадковості.
3. Метод опорних векторів (Support Vector Machine) – модель, яка шукає оптимальну гіперплощину для поділу класів, добре працює на невеликих і середніх наборах даних. Для роботи з багатокласовістю було включено параметр probability=True, що дозволяє отримувати ймовірності класів (для подальшого використання в підході з самонавчанням). Також вказано class_weight='balanced'. На рисунку 3.20 показано ініціалізацію трьох вищезгаданих моделей.

```
models = {
    'Logistic Regression': LogisticRegression(class_weight = 'balanced', max_iter=1000, random_state=42),
    'Random Forest': RandomForestClassifier(class_weight = 'balanced', n_estimators=100, random_state=42),
    'SVM': SVC(class_weight='balanced', probability=True, random_state=42)
}
```

Рисунок 3.20 Код ініціалізації моделей

4. XGBoost – потужний градієнтний бустинг, який часто показує відмінні результати при правильному налаштуванні параметрів. Параметр objective='multi:softmax' обирається для багатокласової класифікації, де кожен клас є окремим результатом, і модель навчається призначати об’єкти до одного з кількох можливих класів, повертаючи тільки клас, до якого належить даний об'єкт. Параметр

num_class=len(y.unique()) визначає кількість класів у завданні класифікації, використовуючи кількість унікальних міток у цільовій змінній. Для оцінки якості класифікації застосовується метрика eval_metric='mlogloss', яка обчислює багатокласові кросс-ентропійні втрати, що дозволяє моделі досягати кращих результатів, мінімізуючи значення цієї метрики. На рисунку 3.21 є приклад ініціалізації даної моделі.

```
xgb_model = XGBClassifier(
    objective='multi:softmax',
    num_class=len(y.unique()),
    eval_metric='mlogloss',
    random_state=42
)
```

Рисунок 3.21 Ініціалізація моделі XGBoost

Для методу XGBoost також попередньо проводилось кодування міток класів за допомогою LabelEncoder. Всі класи з текстових значень перейшли в числові: 0, 1, 2 і тд.

Ці моделі тренувались виключно на розміченій частині даних, без використання жодних додаткових текстів з невідомими мітками. Метою цього блоку було створення базової лінії продуктивності, з якою можна буде порівнювати подальші вдосконалені підходи.

Для покращення якості векторизації та покращення якості класифікації замість стандартного TF-IDF векторайзера було застосовано модель paraphrase-multilingual-MiniLM-L12-v2 для отримання більш глибоких та змістовних векторів для кожного тексту. Її було імпортовано за допомогою бібліотеки sentence_transformers. Для векторизації модель на вхід отримує список текстів, а на виході дає масив кодованих значень. Приклад векторизації за допомогою MiniLM наведено на рисунку 3.22.

```
from sentence_transformers import SentenceTransformer
mini_lm_model = SentenceTransformer("paraphrase-multilingual-MiniLM-L12-v2")

X_train_minilm = mini_lm_model.encode(X_train.tolist(), show_progress_bar=True)
X_test_minilm = mini_lm_model.encode(X_test.tolist(), show_progress_bar=True)
```

Рисунок 3.22 Ініціалізація векторайзера та кодування тексту

Для покращення результатів класифікації та використання не тільки розмічених даних, а й нерозмічених текстів, застосовувався підхід самонавчання (SelfTrainingClassifier). Цей підхід полягає в тому, що модель спочатку тренується на розмічених даних, після чого робить прогноз на нерозмічені дані. Потім ці передбачення додаються до тренувальної вибірки як нові "розмічені" дані, що дозволяє моделі покращити свої результати, працюючи з більшою кількістю даних.

Для реалізації підходу самонавчання було використано SelfTrainingClassifier з бібліотеки scikit-learn. Використовувався стандартний параметр threshold=0.75, який визначає поріг ймовірності для віднесення прикладу до певного класу. Якщо ймовірність віднесення до класу перевищує цей поріг, то цей приклад додається до навчальної вибірки для наступної ітерації. Цей метод був застосований до всіх 4 моделей. Реалізація підходу показана на рисунку 3.23.

```
for name, model in models.items():
    print(f"--- {name} ---")
    self_training_model = SelfTrainingClassifier(model)
    self_training_model.fit(Xt_train_vect, yt_train)
    yt_pred = self_training_model.predict(Xt_test_vect)

    mask = yt_test.notna()

    yt_test_clean = yt_test[mask]
    yt_pred_clean = yt_pred[mask]

    print(classification_report(yt_test_clean, yt_pred_clean, zero_division=0))
```

Рисунок 3.23 Реалізація самонавчання

На останньому етапі було застосовано співнавчання для покращення точності моделей. Співнавчання є ще одним підходом напівконтрольованого навчання, який передбачає використання двох моделей, що навчаються одночасно. Кожна модель надає мітки для прикладів, де її впевненість у передбаченні перевищує певний поріг. Ці мітки використовуються для тренування іншої моделі, тим самим покращуючи її навчання. Реалізація функції відбувалась за таким алгоритмом:

1) Клонування моделей.
2) Ініціалізація навчальних вибірок.
3) Основний цикл (кількість ітерацій задана n_iter):
    a) Навчання моделей на їхніх поточних даних.
    b) Прогнозування ймовірностей класів для всіх нерозмічених прикладів.
    c) Обчислення впевненості: для кожного прикладу вибирається максимальна ймовірність.
    d) Вибір найбільш впевнений top_n прикладів для кожної моделі.
    e) Обмін мітками.
    f) Оновлення навчальних вибірок обох моделей.
    g) Видалення використаних прикладів з нерозміченого набору.
4) Фінальне донавчання моделей на розширених наборах даних.
5) Оцінка якості обох моделей на тестовій вибірці.
6) Повернення натренованих моделей.

Для даного підходу було застосовано три пари моделей: логістичної регресії та методу опорних векторів, методу опорних векторів та XGBoost, а також логістичної регресії та XGBoost. Випадковий ліс не був включений до жодної з пар через його слабку продуктивність у попередніх етапах.

На рисунку 3.24 показано приклад для пари моделей логістична регресія та SVM. Кількість ітерацій 100, а кількість прикладів для обміну – 70. Реалізація функції співнавчання наведена в додатку Б.

```
model1 = LogisticRegression(class_weight='balanced', max_iter=1000, random_state=42)
model2 = SVC(class_weight='balanced', probability=True, random_state=42)

lr_svm, smv_lr = co_training(Xl_train_minilm, yl_train, X_unlabeled_minilm,
                             Xl_test_minilm, yl_test, model1, model2, n_iter=100, top_n=70)
```

Рисунок 3.24 Визначення моделей та виклик функції співнавчання

3.3 Оцінка результатів

3.3.1 Результати базових моделей з TF-IDF

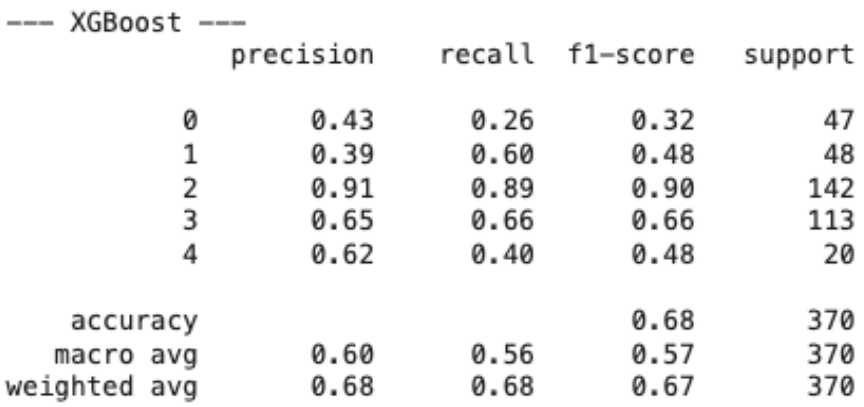
На початковому етапі було здійснено навчання чотирьох базових моделей класифікації: логістичної регресії (Logistic Regression), випадкового лісу (Random Forest), методу опорних векторів (SVM) та градієнтного бустингу XGBoost. Усі моделі були натреновані на розмічених даних із використанням класичної векторизації TF-IDF. Метою цього експерименту було оцінити базову ефективність алгоритмів.

Для кожної моделі було виведено класифікаційний звіт за допомогою функції classification_report, який обчислює набір стандартних метрик якості класифікації: точність (precision), повноту (recall), F1-міру (f1-score) та support (абсолютна кількість тестових прикладів) для кожного класу. Крім того, оцінювалися загальна точність моделі (accuracy), середнє макро (macro avg) та зважене середнє (weighted avg).

Усі розглянуті моделі демонструють нерівномірні результати залежно від класу: найвищі показники precision, recall та f1-score спостерігаються для класу «Похвала від клієнта», що пояснюється як якістю текстів цього класу, так і великою кількістю прикладів у навчальній вибірці. Натомість продуктивність моделей значно знижується для класів «Модераційні пропозиції» та особливо «Шахраї», де найбільше страждає метрика recall, що свідчить про труднощі у виявленні відповідних

прикладів. Це вказує на загальну проблему рідкісних класів: через дисбаланс у представленості категорій моделі не здатні ефективно навчитися їх розпізнавати, що є типовою складністю при класифікації незбалансованих текстових даних.

Логістична регресія забезпечує загалом збалансовану продуктивність з точністю 0.69. Вона найкраще серед усіх ідентифікує приклади з малочисельних класів: «Модераційні пропозиції» (recall = 0.36) та «Шахраї» (recall = 0.50). Random Forest показує трохи нижчу загальну точність (0.68), при цьому для класів, де мало коментарів повнота значно нижча, що знижує загальну F1-міру. SVM досягає найвищої точності серед усіх базових моделей – 0.71. Особливо високі результати демонструє на класі «Продуктові проблеми/пропозиції» (recall = 0.9, precision = 0.55), проте як і в інших моделей, залишаються проблеми з класами «Модераційні пропозиції» та «Шахраї». XGBoost, незважаючи на загальну точність на рівні 0.68, не перевершує інші моделі за жодним із ключових показників. Найкращі результати досягаються знову ж для класу «Похвала від клієнта» (f1-score = 0.90), тоді як продуктивність для рідкісних класів залишається незадовільною. Порівняльна характеристика моделей наведена на рисунках 3.25 та 3.26.



--- XGBoost ---

	precision	recall	f1-score	support
0	0.43	0.26	0.32	47
1	0.39	0.60	0.48	48
2	0.91	0.89	0.90	142
3	0.65	0.66	0.66	113
4	0.62	0.40	0.48	20
accuracy			0.68	370
macro avg	0.60	0.56	0.57	370
weighted avg	0.68	0.68	0.67	370

Рис 3.25 Класифікаційний звіт для XGBoost з використанням TF-IDF

```
--- Logistic Regression ---
                              precision    recall  f1-score   support

        Модераційні пропозиції       0.46      0.36      0.40        47
                    Не формат       0.42      0.62      0.50        48
           Похвала від клієнта       0.98      0.84      0.90       142
Продуктові проблеми/пропозиції       0.64      0.70      0.67       113
                        Шахраї       0.62      0.50      0.56        20

                      accuracy                           0.69       370
                     macro avg       0.62      0.60      0.61       370
                  weighted avg       0.72      0.69      0.70       370

--- Random Forest ---
                              precision    recall  f1-score   support

        Модераційні пропозиції       0.40      0.09      0.14        47
                    Не формат       0.37      0.60      0.46        48
           Похвала від клієнта       0.90      0.89      0.89       142
Продуктові проблеми/пропозиції       0.65      0.78      0.71       113
                        Шахраї       1.00      0.30      0.46        20

                      accuracy                           0.68       370
                     macro avg       0.66      0.53      0.53       370
                  weighted avg       0.70      0.68      0.66       370

--- SVM ---
                              precision    recall  f1-score   support

        Модераційні пропозиції       0.52      0.26      0.34        47
                    Не формат       0.71      0.50      0.59        48
           Похвала від клієнта       0.98      0.84      0.90       142
Продуктові проблеми/пропозиції       0.55      0.90      0.69       113
                        Шахраї       0.86      0.30      0.44        20

                      accuracy                           0.71       370
                     macro avg       0.72      0.56      0.59       370
                  weighted avg       0.75      0.71      0.70       370
```

Рис 3.26 Класифікаційний звіт для моделей з використанням TF-IDF

Отже, використання TF-IDF як методу векторизації дозволяє моделі загалом адекватно розрізняти основні класи, однак недостатньо для якісної класифікації всіх категорій. Погані результати на класах з низькою підтримкою свідчать про потребу в додаткових техніках.

Отримані результати є відправною точкою для подальших експериментів, зокрема використання більш складних методів представлення тексту (семантичні ембединги), технік напівконтрольованого навчання, а також методів роботи з незбалансованими даними, описаних у наступних підрозділах.

3.3.2 Покращення моделей за допомогою MiniLM

Для покращення якості класифікації було замінено векторайзер TF-IDF на paraphrase-multilingual-MiniLM-L12-v2.

Для логістичної регресії значно зросли показники для всіх класів, особливо для проблемних «Модераційні пропозиції» (precision 0.53 vs 0.46 та recall 0.62 vs 0.36) та «Шахраї» (recall 0.70 vs 0.50). Також покращилися результати для «Не формат» та «Продуктові проблеми/пропозиції». Загальні метрики показують ріст точності з 0.69 до 0.76, середнє макро F1-міри з 0.61 до 0.69, а зважене середнє F1-міри з 0.70 до 0.77.

Використання покращеного векторайзера MiniLM принесло певні покращення для методу випадковий ліс: точність зросла з 0.68 до 0.75, середнє макро F1-міри з 0.53 до 0.58, а зважене середнє F1-міри з 0.66 до 0.70. Однак в цілому, випадковий ліс продемонстрував найгірші результати серед усіх моделей. Для класу «Модераційні пропозиції» значно зросла точність (0.80 vs 0.40), проте повнота залишилась незмінною (0.09). Для мітки «Не формат» теж впала повнота (0.35 vs 0.62), але точність зросла (0.71 vs 0.42). Аналогічно для класу«Шахраї» (recall 0.45 vs 0.50, precision 0.90 vs 0.62). Незважаючи на зростання точності, показники повноти залишились незадовільними, що свідчить про труднощі з виявленням рідкісних категорій.

Подібно до логістичної регресії, SVM також демонструє значне покращення для всіх класів, особливо для «Модераційні пропозиції» (precision 0.55 vs 0.52, recall 0.60 vs 0.26, f1-score 0.57 vs 0.34) та «Шахраї» (precision 0.74 vs 0.86, recall 0.70 vs 0.30, f1-score 0.72 vs 0.44). Також помітно зросли показники для «Не формат» та «Продуктові проблеми/пропозиції». Загальні точність зросла з 0.71 до 0.78, середнє макро F1-міри з 0.59 до 0.72, а зважене середнє F1-міри з 0.70 до 0.79.

Для класифікатора XGBoost значно покращилися показники для всіх категорій, особливо для тих, які важко ідентифікувати: «Шахраї» (precision

0.76 vs 0.62, recall 0.65 vs 0.40, f1-score 0.70 vs 0.48), «Модераційні пропозиції» (precision 0.64 vs 0.43, recall 0.34 vs 0.26, f1-score 0.44 vs 0.32).

Загальна точність – accuracy – становить 0.78, що є значним покращенням порівняно з TF-IDF (0.68). Середнє макро для f1-score зросло до 0.69 (було 0.57). Зважене середнє для f1-score становить 0.77 (було 0.67). Детальніше з результатами можна ознайомитись на рисунках 3.27 та 3.28.

```
--- Logistic Regression ---
                              precision    recall  f1-score   support

       Модераційні пропозиції       0.53      0.62      0.57        47
                   Не формат       0.54      0.67      0.60        48
          Похвала від клієнта       0.96      0.92      0.94       142
Продуктові проблеми/пропозиції       0.84      0.68      0.75       113
                       Шахраї       0.52      0.70      0.60        20

                     accuracy                           0.76       370
                    macro avg       0.68      0.72      0.69       370
                 weighted avg       0.79      0.76      0.77       370

--- Random Forest ---
                              precision    recall  f1-score   support

       Модераційні пропозиції       0.80      0.09      0.15        47
                   Не формат       0.74      0.35      0.48        48
          Похвала від клієнта       0.92      0.95      0.93       142
Продуктові проблеми/пропозиції       0.60      0.98      0.74       113
                       Шахраї       0.90      0.45      0.60        20

                     accuracy                           0.75       370
                    macro avg       0.79      0.56      0.58       370
                 weighted avg       0.78      0.75      0.70       370

--- SVM ---
                              precision    recall  f1-score   support

       Модераційні пропозиції       0.55      0.60      0.57        47
                   Не формат       0.58      0.69      0.63        48
          Похвала від клієнта       0.96      0.90      0.93       142
Продуктові проблеми/пропозиції       0.78      0.76      0.77       113
                       Шахраї       0.74      0.70      0.72        20

                     accuracy                           0.78       370
                    macro avg       0.72      0.73      0.72       370
                 weighted avg       0.79      0.78      0.79       370
```

Рисунок 3.27 Класифікаційний звіт для моделей з використанням paraphrase-multilingual-MiniLM-L12-v2

```
--- XGBoost ---
              precision    recall  f1-score   support

           0       0.67      0.34      0.45        47
           1       0.65      0.54      0.59        48
           2       0.93      0.96      0.94       142
           3       0.69      0.87      0.77       113
           4       0.76      0.65      0.70        20

    accuracy                           0.78       370
   macro avg       0.74      0.67      0.69       370
weighted avg       0.78      0.78      0.77       370
```

Рисунок 3.28 Класифікаційний звіт для XGBoost з використанням paraphrase-multilingual-MiniLM-L12-v2

Отже, на етапі використання покращеного векторизатора MiniLM, моделі XGBoost та SVM показують найкращі та дуже схожі результати серед базових моделей, навчених лише на розмічених даних. Перехід до напівконтрольованого навчання може показати, чи вдасться ще більше покращити ці показники.

3.3.3 Застосування техніки самонавчання

Після застосування SelfTrainingClassifier моделі відреагували по різному: випадковий ліс показав погіршення в точності, проте інші моделі мали підвищення в точності, хоч і не значні. Розглянемо детальніше кожну модель.

Загальна точність LR залишилася на рівні 0.76, але спостерігаються покращення для окремих класів. Зокрема, зросли показники F1-міри для «Модераційні пропозиції» (0.62 vs 0.57) та «Не формат» (0.65 vs 0.60). Загальні метрики показали невелике покращення: середнє макро F1-міри зросло з 0.69 до 0.71, а зважене середнє F1-міри залишилося на рівні 0.77.

Результати випадкового лісу після самонавчання погіршилися. Відзначається значне зниження повноти для «Модераційні пропозиції» та «Шахраї», що призвело до низьких F1-міри для цих класів. Для «Не формат» точність зросла, але повнота знизилася. Загальна точність знизилася з 0.75 до 0.69. Це може бути спричинено тим, що модель мала відносно невисокі показники ефективності і коли почала вчитись на своїх “невдалих” прогнозах, то точність стала ще меншою.

Самонавчання дало позитивний ефект на результати SVM, з підвищенням загальної точності з 0.78 до 0.79. Покращились показники F1-міри для «Модераційні пропозиції» (0.66 vs 0.57) та «Не формат» (0.68 vs 0.63). Показники для «Похвала від клієнта» залишились високими, а для «Шахраї» також відбулося незначне зростання F1-міри (0.71 vs 0.72).

Загальні метрики теж демонструють покращення: середнє макро F1-міри зросло з 0.72 до 0.75, а зважене середнє F1-міри залишилось на рівні 0.79.

XGBoost показав покращення результатів, зокрема для «Модераційні пропозиції» (f1-score 0.49 vs 0.44), «Не формат» (f1-score 0.71 vs 0.59) та «Шахраї» (f1-score 0.72 vs 0.70). Також зросла повнота для «Продуктові проблеми/пропозиції». Загальна точність зросла з 0.78 до 0.79, середнє макро F1-міри зросло з 0.69 до 0.72, а зважене середнє F1-міри залишилось на рівні 0.77. XGBoost показав одне з найбільших покращень серед моделей після самонавчання. Детальніше можна ознайомитись зі зміною показників на рисунках 3.29 та 3.30.

Підсумовуючи, більшість моделей продемонстрували покращення результатів після самонавчання, зокрема зростання точності та F1-міри для деяких класів. Після самонавчання SVM та XGBoost досягли найвищої точності (0.79) і зваженого F1-міри (0.79 та 0.78 відповідно). Випадковий ліс продемонстрував погіршення після самонавчання, що, ймовірно, пов'язано з його обробкою даних, позначених автоматично.

```
--- XGBoost ---
              precision    recall  f1-score   support

         0.0       0.79      0.36      0.49        42
         1.0       0.85      0.60      0.71        58
         2.0       0.95      0.91      0.93       125
         3.0       0.65      0.95      0.77       107
         4.0       0.82      0.64      0.72        22

    accuracy                           0.79       354
   macro avg       0.81      0.69      0.72       354
weighted avg       0.82      0.79      0.78       354
```

Рисунок 3.29 Класифікаційний звіт для XGBoost з використанням paraphrase-multilingual-MiniLM-L12-v2 та самонавчанням

```
--- Logistic Regression ---
                              precision    recall  f1-score   support

        Модераційні пропозиції       0.56      0.71      0.62        42
                     Не формат       0.62      0.69      0.65        58
           Похвала від клієнта       0.97      0.88      0.92       125
Продуктові проблеми/пропозиції       0.80      0.68      0.74       107
                        Шахраї       0.52      0.73      0.60        22

                      accuracy                           0.76       354
                     macro avg       0.69      0.74      0.71       354
                  weighted avg       0.79      0.76      0.77       354

--- Random Forest ---
                              precision    recall  f1-score   support

        Модераційні пропозиції       0.00      0.00      0.00        42
                     Не формат       0.93      0.47      0.62        58
           Похвала від клієнта       0.97      0.85      0.91       125
Продуктові проблеми/пропозиції       0.50      0.99      0.67       107
                        Шахраї       1.00      0.23      0.37        22

                      accuracy                           0.69       354
                     macro avg       0.68      0.51      0.51       354
                  weighted avg       0.71      0.69      0.65       354

--- SVM ---
                              precision    recall  f1-score   support

        Модераційні пропозиції       0.65      0.67      0.66        42
                     Не формат       0.78      0.60      0.68        58
           Похвала від клієнта       0.96      0.89      0.92       125
Продуктові проблеми/пропозиції       0.69      0.84      0.76       107
                        Шахраї       0.75      0.68      0.71        22

                      accuracy                           0.79       354
                     macro avg       0.77      0.74      0.75       354
                  weighted avg       0.80      0.79      0.79       354
```

Рисунок 3.30 Класифікаційний звіт для моделей з використанням paraphrase-multilingual-MiniLM-L12-v2 та самонавчанням

3.3.4 Результати співнавчання

Порівняємо результати кожної пари та зробимо загальні висновки щодо ефективності співнавчання.

1) Логістична регресія та SVM (рисунок 3.31):
   - Логістична регресія: Загальна точність 0.77, зважене середнє F1-міри 0.77.
   - SVM: Загальна точність 0.76, зважене середнє F1-міри 0.77.

```
Оцінка для моделі 1:
              precision    recall  f1-score   support

           0       0.61      0.69      0.64        54
           1       0.52      0.58      0.55        43
           2       0.99      0.88      0.93       155
           3       0.75      0.70      0.73        94
           4       0.56      0.79      0.66        24

    accuracy                           0.77       370
   macro avg       0.68      0.73      0.70       370
weighted avg       0.79      0.77      0.77       370

Оцінка для моделі 2:
              precision    recall  f1-score   support

           0       0.57      0.72      0.64        54
           1       0.53      0.60      0.57        43
           2       0.99      0.86      0.92       155
           3       0.76      0.69      0.72        94
           4       0.61      0.79      0.69        24

    accuracy                           0.76       370
   macro avg       0.69      0.73      0.71       370
weighted avg       0.79      0.76      0.77       370
```

Рисунок 3.31 Результати співнавчання: модель 1 – логістична регресія, модель 2 – SVM

Обидві моделі в цій парі показують схожі результати. Логістична регресія показує кращі результати в точності, ніж при самонавчанні, але все ще трохи нижчі за найкращі результати самонавчання для окремих моделей (SVM та XGBoost з точністю 0.79). Проте SVM показала погіршення порівняно з самонавчанням в усіх загальних показниках (accuracy 0.76 vs 0.79, macro avg f10.71 vs 0.75, weighted avg f1 0.77 vs 0.79).

2) Логістична регресія та XGBoost (рисунок 3.32):

- Логістична регресія: Загальна точність 0.76, зважене середнє F1-міри 0.77. Результати майже ідентичні попередній парі.
- XGBoost: Загальна точність 0.78, зважене середнє F1-міри 0.79. XGBoost у цій комбінації показує найкращі результати

досягнутих для XGBoost. Порівняно з самонавчанням ця модель краще прогнозує малочисельні класи.

Ця пара демонструє кращу продуктивність, ніж перша, завдяки сильним показникам XGBoost.

Оцінка для моделі 1:

	precision	recall	f1-score	support
0	0.58	0.72	0.64	54
1	0.50	0.58	0.54	43
2	0.99	0.87	0.92	155
3	0.75	0.70	0.73	94
4	0.64	0.75	0.69	24
accuracy			0.76	370
macro avg	0.69	0.73	0.70	370
weighted avg	0.79	0.76	0.77	370

Оцінка для моделі 2:

	precision	recall	f1-score	support
0	0.57	0.69	0.62	54
1	0.60	0.58	0.59	43
2	0.98	0.88	0.93	155
3	0.73	0.78	0.75	94
4	0.78	0.75	0.77	24
accuracy			0.78	370
macro avg	0.73	0.74	0.73	370
weighted avg	0.80	0.78	0.79	370

Рисунок 3.32 Результати співнавчання: модель 1 – логістична регресія, модель 2 – XGBoost

3) SVM та XGBoost (рисунок 3.33):

- SVM: Загальна точність 0.78, зважене середнє F1-міри 0.77. Результати дещо нижчі за найкращі показники SVM при самонавчанні (точність 0.79, макро-середнє F1 0.75, зважене середнє – 0.79).
- XGBoost: Загальна точність 0.73, зважене середнє F1-міри 0.69. Продуктивність XGBoost у цій парі значно нижча за його найкращі результати при самонавчанні.

Ця пара показує неоднозначні результати, де SVM працює досить добре, а XGBoost – гірше, ніж очікувалося.

Оцінка для моделі 1:

	precision	recall	f1-score	support
0	0.70	0.43	0.53	54
1	0.67	0.47	0.55	43
2	0.97	0.90	0.93	155
3	0.62	0.94	0.75	94
4	0.81	0.71	0.76	24
accuracy			0.78	370
macro avg	0.75	0.69	0.70	370
weighted avg	0.79	0.78	0.77	370

Оцінка для моделі 2:

	precision	recall	f1-score	support
0	0.50	0.09	0.16	54
1	0.67	0.42	0.51	43
2	0.95	0.92	0.93	155
3	0.54	0.96	0.69	94
4	0.88	0.58	0.70	24
accuracy			0.73	370
macro avg	0.71	0.59	0.60	370
weighted avg	0.74	0.73	0.69	370

Рисунок 3.33 Результати співнавчання: модель 1 – SVM, модель 2 – XGBoost

Найуспішнішою виявилась пара логістична регресія та XGBoost: саме тут XGBoost досягає найвищих показників у режимі співнавчання, що порівнянні з результатами найкращих моделей при самонавчанні. Це свідчить про те, що співнавчання може бути не менш ефективним, ніж самонавчання, якщо правильно підібрати пари моделей.

Для того, щоб узагальнити всі моделі було сформовано порівняльну таблицю на рисунку 3.34. Порівняльні таблиці для кожного класу представлені в додатках В, Г, Д, Е, Ж.

Модель	Векторизація	Навчання	Accuracy	Macro Avg F1-score	Weighted Avg F1-score
Логістична регресія	TF-IDF	Лише розмічені дані	69%	61%	70%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	76%	69%	77%
	paraphrase-MiniLM-L12-v2	Самонавчання	76%	71%	77%
	paraphrase-MiniLM-L12-v2	Співнавчання з SVM	**77%**	**70%**	**77%**
	paraphrase-MiniLM-L12-v2	Співнавчання з XGBoost	76%	69%	77%
Випадковий ліс	TF-IDF	Лише розмічені дані	68%	53%	66%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	**75%**	**58%**	**70%**
	paraphrase-MiniLM-L12-v2	Самонавчання	69%	49%	62%
SVM	TF-IDF	Лише розмічені дані	71%	59%	70%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	78%	72%	79%
	paraphrase-MiniLM-L12-v2	Самонавчання	**79%**	**75%**	**79%**
	paraphrase-MiniLM-L12-v2	Співнавчання з LR	76%	71%	77%
	paraphrase-MiniLM-L12-v2	Співнавчання з XGBoost	78%	69%	77%
XGBoost	TF-IDF	Лише розмічені дані	68%	57%	67%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	78%	69%	77%
	paraphrase-MiniLM-L12-v2	Самонавчання	79%	72%	77%
	paraphrase-MiniLM-L12-v2	Співнавчання з LR	**78%**	**73%**	**79%**
	paraphrase-MiniLM-L12-v2	Співнавчання з SVM	73%	64%	69%

Рисунок 3.34 Порівняльна таблиця загальних метрик для різних варіацій моделей

Отже, paraphrase-multilingual-MiniLM-L12-v2 у поєднанні з напівконтрольованим навчанням як у вигляді самонавчання, так і співнавчання дозволив суттєво підвищити якість класифікації. Найкращі результати були досягнуті для моделей SVM та XGBoost, які продемонстрували точність до 0.79. Водночас варто зазначити, що оцінити справжню точність моделі складно, оскільки розмітка виконувалась людьми, що неминуче вводить елемент суб’єктивності. У ряді випадків спостерігається розходження між передбаченням моделі та міткою модератора. Переглядаючи такі приклади, видно, що модель спрогнозувала правильний клас, на відміну від модератора. Це вказує на те, що моделі не лише здатні виявляти шаблони в даних, а іноді й забезпечують більш узгоджену та об’єктивну інтерпретацію контенту, ніж людські оцінки.

Висновки

У цьому розділі було здійснено повний цикл побудови моделей класифікації коментарів користувачів: від збору та попереднього аналізу даних до оцінки результатів різних підходів машинного навчання.

Проведено підготовку текстів, включаючи очищення та векторизацію за допомогою TF-IDF та MiniLM.

Результати показали, що навіть базові моделі на основі TF-IDF (зокрема XGBoost і SVM) забезпечують непогану продуктивність, однак використання MiniLM значно підвищує точність і F1-міру. Подальше застосування технік напівконтрольованого навчання, таких як самонавчання та співнавчання, дозволило покращити результати без додаткової ручної розмітки. Найвищі результати були досягнуті моделями XGBoost та SVM, зокрема після самонавчання або в комбінації з логістичною регресією в межах співнавчання.

Водночас варто відзначити, що оцінка справжньої точності має певні обмеження, пов’язані з людським фактором при розмітці даних. У низці випадків моделі прогнозували категорію, яка здавалася обґрунтованішою, ніж та, яку обрав модератор. Це свідчить про наявність суб’єктивності в оцінці, а отже – про потенціал моделей як додаткового або альтернативного інструменту для підтримки процесу модерації.

РОЗДІЛ 4
ІНТЕРФЕЙС КОРИСТУВАЧА ТА ПЕРСПЕКТИВИ РОЗВИТКУ

4.1 Інтерфейс кінцевого користувача

Інтеграція моделі класифікації відкритих коментарів у внутрішню аналітичну звітність дозволяє суттєво підвищити ефективність обробки даних користувачів і перетворити раніше неструктуровану інформацію на інструмент прийняття рішень. Після розмітки текстів за допомогою моделі дані експортуються у Tableau, де побудовано новий формат дашбордів, орієнтований на візуальний аналіз з автоматичною класифікацією.

У попередній версії блок із відкритими коментарями представляв собою статичну сторінку зі списком текстів і фільтрами за періодом, категорією та питанням опитування. Однак такий підхід не дозволяє виявляти загальні закономірності, тренди чи порівнювати частоту скарг між категоріями. Новий підхід передбачає використання інтерактивних історій, де дані згруповані у логічні блоки з графіками, біграмами, фільтрами та прикладами коментарів.

Інтерфейс дашборда поділено на кілька вкладок, кожна з яких відображає окремий аналітичний зріз:

- Labels stat – статистика за передбаченими мітками,
- Top-20 bigrams – найбільш вживані словосполучення загалом,
- Top-15 bigrams by Labels – ключові біграми в розрізі міток,
- Top-15 bigrams by Categories – біграми за категоріями на сайті,
- Comments – сирі коментарі з автоматичною розміткою.

У верхній частині кожної вкладки розташовані інтерактивні фільтри, які дозволяють обирати період, категорію на сайті, задоволеність респондента якістю контенту та передбачену мітку, що забезпечує користувачам зручну персоналізацію аналітики.

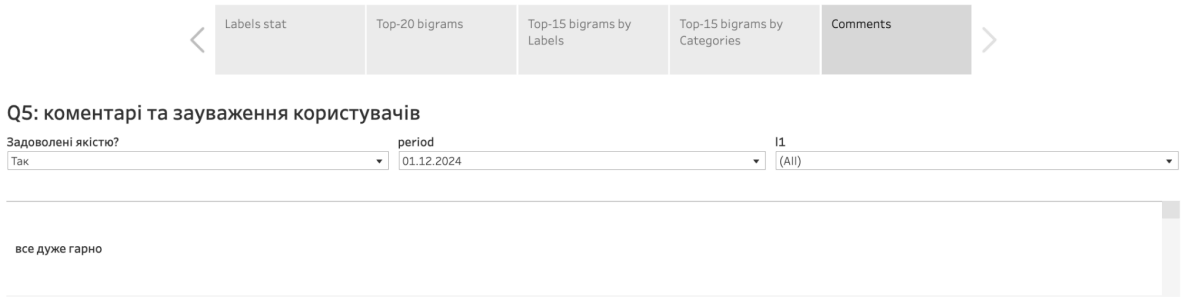


Рисунок 4.1 Приклад інтерфейсу

Перший графік на вкладці зі статистикою представлений у вигляді лінійного графіка, який демонструє зміну частки коментарів із певними передбаченими мітками у часі. По осі X розміщено місяці (починаючи з 2023 року до грудня 2024 року), а по осі Y – відсоток коментарів, що відповідають тій чи іншій мітці. Кожна лінія відповідає окремій мітці, її коливання ілюструють, як змінювалася частка коментарів відповідного типу протягом часу. На рисунку 4.2 видно, що % коментарів з міткою Продуктові проблеми / пропозиції показує динаміку на зменшення, натомість Похвала від клієнта зростає. Це може свідчити про те, що користувачі помічають зміни, які вводяться для покращення їх досвіду користування.

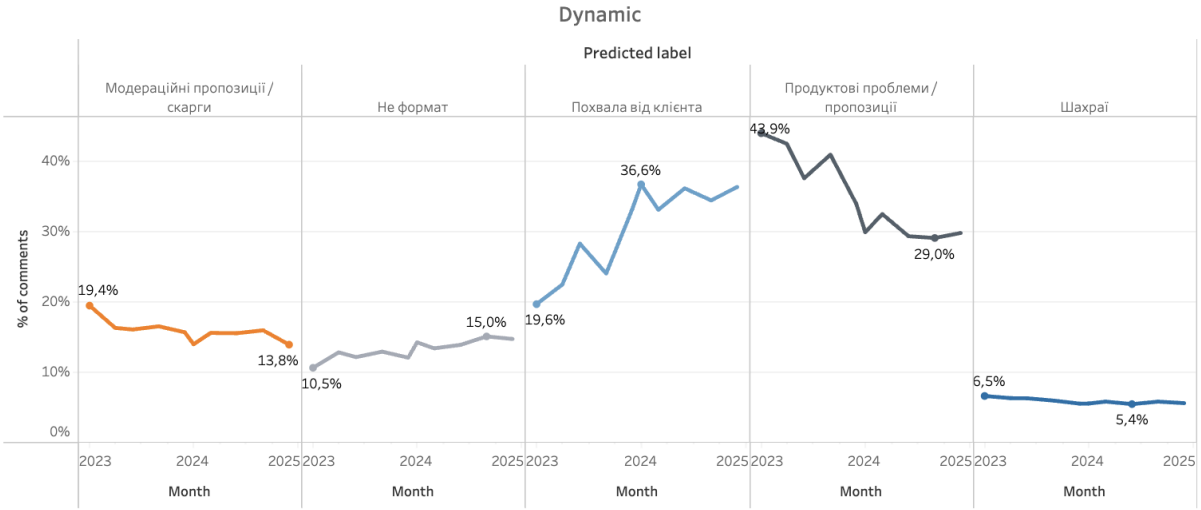


Рисунок 4.2 Динаміка долі коментарів з міткою

Наступний графік складається з двох горизонтальних стовпчастих діаграм, що відображають загальний розподіл міток за обраний період. Ліва діаграма показує абсолютну кількість коментарів з кожною міткою, а права – відсоткову частку кожної мітки від загальної кількості. Обидві діаграми використовують однакову шкалу по осі Y (назви міток), що дозволяє напряму порівнювати абсолютні та відносні значення. Цей блок дає уявлення про те, які типи коментарів є найпоширенішими в цілому та як виглядає їхня структура у відсотковому співвідношенні. В результаті опитування в грудні 2024 року найбільше було коментарів-похвали, на другому місці Продуктові проблеми та пропозиції. Найменше коментарів з міткою Шахраї.

Рисунок 4.3 Розподіл коментарів по міткам в грудні 2024

Останній графік на сторінці (рисунок 4.4) представлений у вигляді двох вертикальних стовпчастих діаграм, розташованих поруч. Вони показують, як передбачені мітки розподіляються між різними категоріями на сайті (наприклад, Бізнес та послуги, Дитячий світ, Дім і сад, тощо). Ліва діаграма відображає абсолютну кількість коментарів у кожній категорії, права – відносну і є нормованою до 100%. Кожен стовпець поділений на кольорові сегменти, відповідні міткам, а числові значення вказують точну кількість. Таким чином видно, які теми є найбільш характерними для кожної категорії незалежно від їхнього розміру. Цей блок дозволяє виявити як популярні теми загалом, так і специфічні патерни у зверненнях користувачів залежно від категорії контенту.

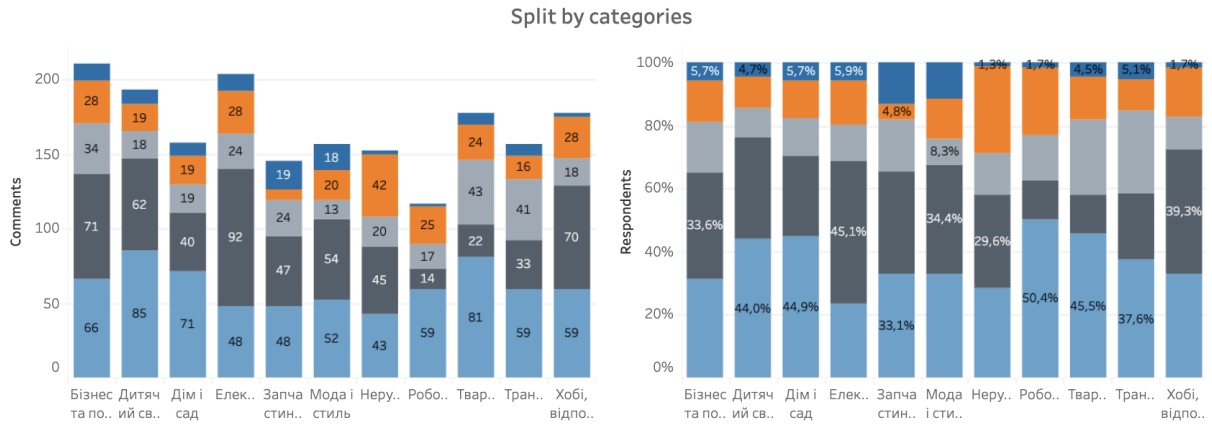


Рисунок 4.4 Розподіл коментарів за категоріями

Блок top-20 bigrams відображає найбільш вживані словосполучення у відкритих коментарях. У центрі розміщено горизонтальну стовпчасту діаграму, де вісь Y показує 20 найпоширеніших біграм (сполучень двох слів), а вісь X – кількість їхніх згадувань у коментарях. Довжина стовпців відповідає частоті вживання, а на їх кінцях зазначено точне числове значення. Наприклад, біграма "номер телефон" зустрічається близько 200 разів, тоді як менш уживані біграми мають частоту близько 40 - 50. У правій верхній частині сторінки розміщені фільтри, як і на попередній сторінці.

Аналіз графіка на рисунку 4.5 свідчить про те, що значна частина коментарів стосується контактної інформації та взаємодії між продавцями і покупцями – це видно з високої частоти таких біграм, як "номер телефон", "відгук продавець", "продавець покупець". Це вказує на важливість комунікаційної складової в оголошеннях та потенційні труднощі або побажання користувачів, пов’язані з процесом зв’язку чи взаємодії.

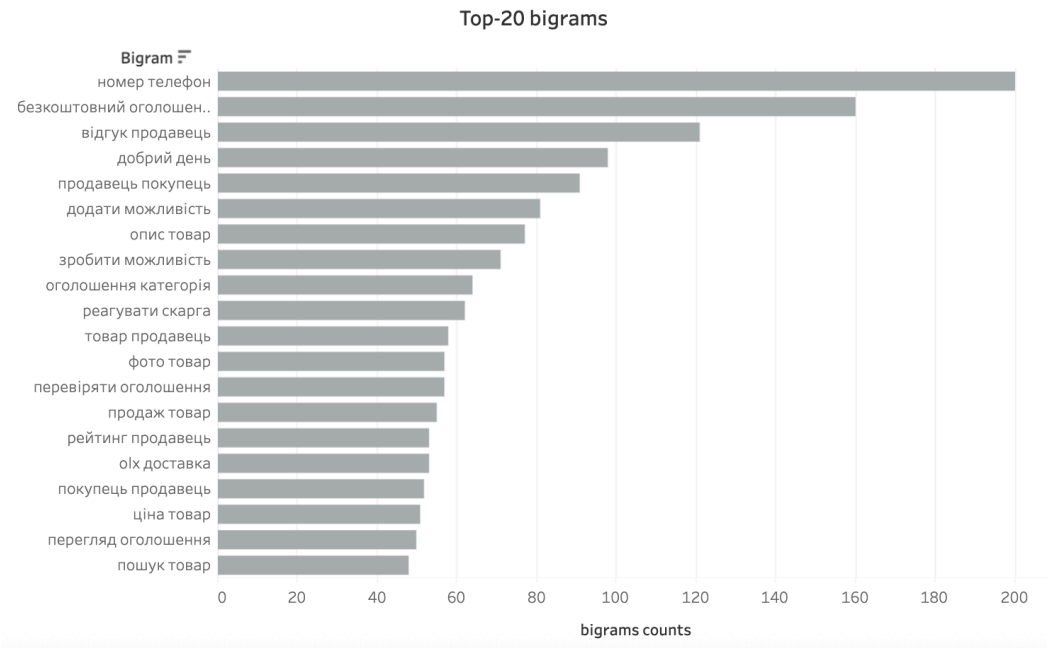


Рисунок 4.5 Топ-20 біграм

Наступний блок Top-15 bigrams by Label дозволяє дослідити найбільш характерні словосполучення в коментарях, які були автоматично класифіковані за передбаченими мітками. Основна частина сторінки представлена у вигляді горизонтальної стовпчастої діаграми, де біграми згруповано відповідно до міток. На рисунку 4.6 відображено дві групи: «Продуктові проблеми / пропозиції» та «Шахраї» – верхня та нижчі частини відповідно.

Для мітки «Продуктові проблеми / пропозиції» найчастіше вживаними є біграми на кшталт: «олх доставка», «безкоштовний оголошення», «відгук продавець», «додати можливість». Вони вказують на основні зони невдоволення або запити користувачів щодо функціональності платформи (доставка, умови розміщення оголошень, технічні можливості).

У мітці «Шахраї» переважають словосполучення, пов’язані з шахрайськими діями: «оголошення шахрай», «блокувати шахрай», «шахрайський оголошення», що свідчить про високу чутливість користувачів до теми безпеки. Часті згадки «олх доставка» і «номер

телефон» у обох мітках підтверджують, що ці елементи є джерелом як технічних проблем, так і потенційних шахрайств.

Порівняльний аналіз груп міток дає змогу виявити лексичні патерни, притаманні різним типам звернень. Наприклад, у «шахрайських» коментарях переважає агресивна лексика та заклики до блокування, тоді як у «продуктових» – конструктивні пропозиції щодо покращення функціоналу.

Завдяки вбудованим фільтрам користувачі можуть деталізувати аналіз: обрати конкретний період, категорію контенту, рівень задоволеності тощо. Це дозволяє вивчати, як змінюється лексика користувачів у залежності від тематики, часу або настрою.

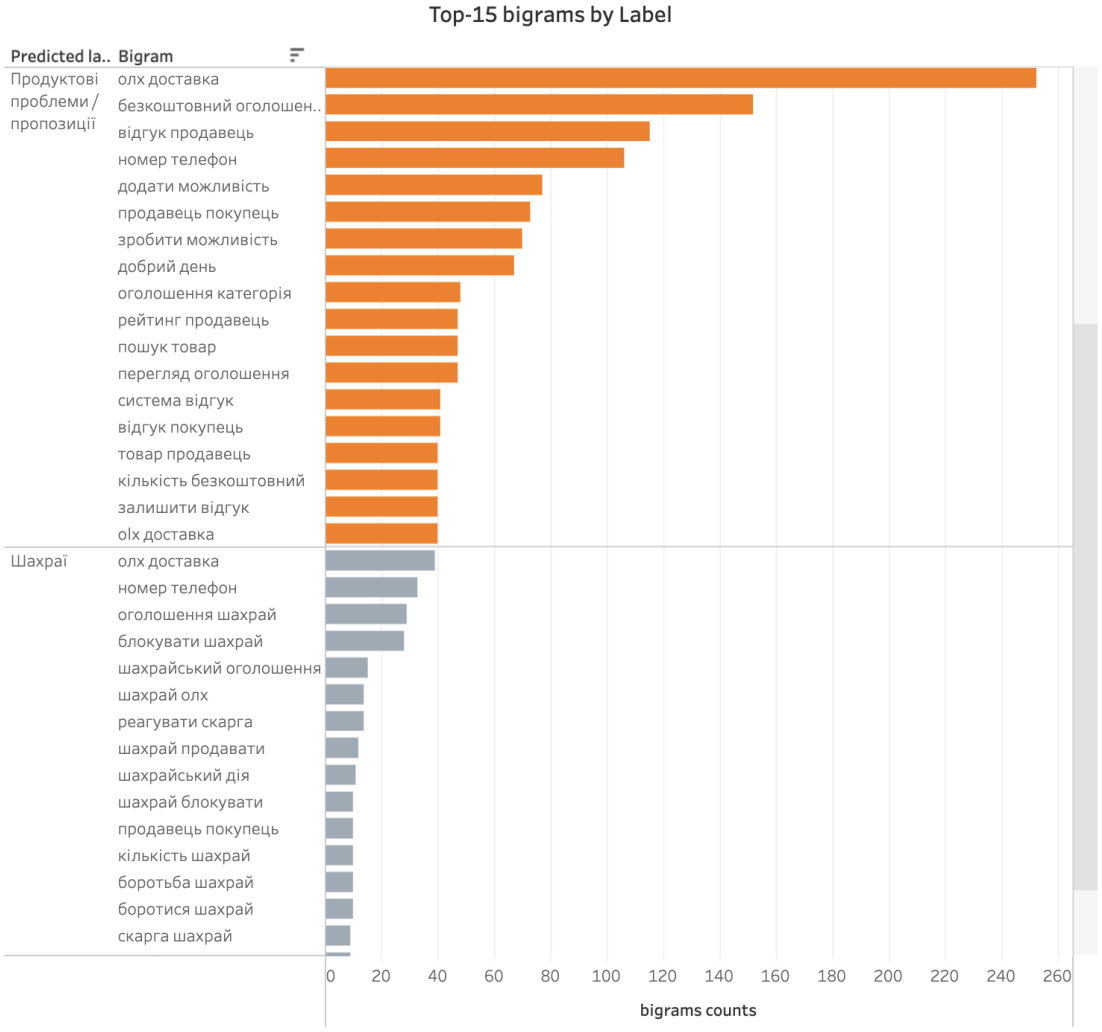


Рисунок 4.6 Топ-15 біграм з поділом на мітки

На рисунку 4.7 показано частину вкладки "Top-15 bigrams by Categories" дашборда. Основна частина візуалізації – це горизонтальні стовпчасті діаграми, де для кожної категорії (наприклад, "Бізнес та послуги", "Робота", "Транспорт") відображено топ-15 найчастіше вживаних біграм у коментарях. Уздовж осі Y розміщено назви категорій і відповідні до них біграми, а вісь X показує кількість згадувань. Стовпці на графіку забарвлені відповідно до передбачених міток (наприклад, синій – "Модераційні пропозиції", помаранчевий – "Продуктові проблеми / пропозиції", тощо), що дозволяє оцінити контекст, у якому згадується кожна біграма. Наприклад, більшість біграм у категорії “Бізнес та послуги” мають помаранчеве або синє забарвлення, що свідчить про концентрацію коментарів на продуктових аспектах (функціональність платформи) та модераційних потребах (наприклад, скарги на контент або користувачів).

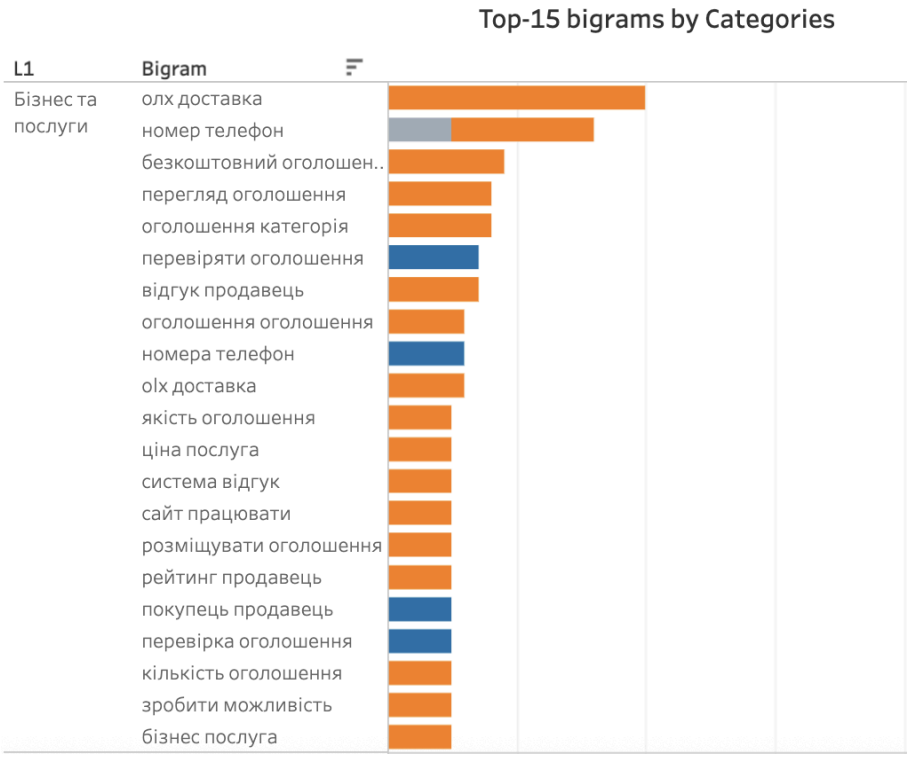


Рисунок 4.7 Топ-15 біграм для категорії Бізнес та послуги

У підсумку, інтеграція автоматичної класифікації коментарів у аналітичну звітність суттєво трансформувала підхід до аналізу зворотного

зв’язку: від ручного перегляду текстів до масштабованого інструменту виявлення трендів, проблем та позитивних змін. Завдяки гнучким фільтрам, візуалізаціям і можливості сегментувати дані за категоріями та мітками, користувачі дашборду можуть оперативно і точно ідентифікувати ключові зони для покращення як продукту, так і процесів. Такий підхід створює надійну основу для подальшої еволюції інструментів аналізу зворотного зв’язку.

4.2 Перспективи розвитку

Одним із найважливіших напрямів розвитку системи класифікації є впровадження сучасних мовних моделей, зокрема архітектур типу Transformer. У базовій реалізації проєкту використовувалися класичні моделі (логістична регресія, випадковий ліс, SVM, XGBoost), які в поєднанні з TF-IDF або MiniLM-векторами показали гарні результати. Проте зростання доступності багатомовних трансформерів, таких як mBERT, XLM-R або компактних моделей (MiniLM, DistilBERT), відкриває можливість підвищити точність класифікації завдяки глибшому розумінню контексту в реченнях. Ці моделі навчені на великих корпусах текстів і здатні враховувати граматичну структуру та значення слів у контексті, що особливо важливо для української мови, де багато значень змінюється залежно від закінчень, порядку слів тощо. Крім того, використання таких моделей може зменшити потребу в ручній інженерії ознак і підвищити адаптивність системи до нових тем і формулювань у відгуках.

На даний момент модель класифікує коментарі у фіксований набір категорій, що задаються вручну. Проте коментарі користувачів часто мають складну структуру й охоплюють кілька аспектів водночас. Наприклад, скарга на модерацію може стосуватись як політики видалення оголошень, так і несвоєчасного реагування. Запровадження ієрархічної класифікації дозволить відображати цю багаторівневу структуру: спочатку

визначатиметься загальний тип проблеми (наприклад, "модерація"), а потім її підтип (наприклад, "відмова в публікації", "видалення без пояснень", "повільна перевірка"). Такий підхід надасть модераторам і аналітикам більш точну інформацію, дозволяючи швидше локалізувати проблемні ділянки сервісу. Ієрархічна структура також дозволить гнучко оновлювати категорії без повного перенавчання всієї моделі – достатньо буде розширити підгілки в дереві класифікації.

Як альтернативна поглибленню ієрархії може бути впровадження методів автоматичного виявлення тем (topic modeling), які дозволяють досліджувати структуру коментарів без попередньої розмітки. Алгоритми, такі як LDA (Latent Dirichlet Allocation) або більш сучасні підходи, як BERTopic, дають змогу згрупувати схожі коментарі за темами та виділити нові напрями, які ще не враховані в існуючій системі категорій. Наприклад, якщо користувачі починають скаржитися на нову функцію, яка щойно з’явилася на платформі, такі моделі зможуть оперативно виявити цю тенденцію ще до того, як модератори вручну її класифікують. Тематичне моделювання також може стати допоміжним інструментом для перегляду та оновлення категорій класифікації, забезпечуючи більшу релевантність до поточних проблем.

Окремим перспективним напрямом є інтеграція великої мовної моделі (LLM), яка б на основі зібраних коментарів у кожній категорії автоматично формувала короткий текстовий звіт чи аналітичний висновок. Така модель отримує як вхідний масив класифікованих повідомлень користувачів, так і назву відповідної категорії (наприклад, "модерація", "платіжна система", "підтримка"), і видає узагальнення основних проблем, які турбують користувачів. Наприклад, у категорії "модерація" модель може згенерувати висновок на кшталт: "Користувачі часто скаржаться на недостатню зрозумілість причин відмови в публікації оголошень і просять детальніші пояснення." Це дозволяє керівництву або продуктовим

командам отримувати не лише структуровані дані, а й змістовні інтерпретації, які легко включити у звіти, презентації чи плани покращень. При цьому використання українськомовної LLM дозволяє досягати високої якості тексту без необхідності ручної обробки.

Доповненням до класифікації може бути визначення емоційного забарвлення коментаря: позитивного, нейтрального або негативного. Це дозволяє не лише розуміти, що саме турбує користувача, але й наскільки гостро він сприймає проблему. Поєднання категоріальної класифікації з аналізом тону дозволяє, наприклад, виявити позитивні відгуки в межах теми "модерація" або, навпаки, негативні реакції на тему "підтримка". Таке розширення значно покращує аналітичну глибину системи та може бути використане для автоматичного пріоритезування звернень, наприклад, негативні коментарі про оплату можуть оброблятися в першу чергу.

Технологія, що використовується для класифікації коментарів з опитувань, може бути поширена й на інші джерела зворотного зв’язку. Наприклад, застосування аналогічних моделей до чатів зі службою підтримки, електронних звернень або повідомлень у соціальних мережах дозволить будувати цілісну картину досвіду користувачів. Це дасть змогу автоматично об’єднувати зворотний зв’язок з різних джерел і виявляти глобальні проблеми або позитивні сторони сервісу, які варто розвивати. З технічної точки зору, це може реалізовуватись через централізовану платформу збору та класифікації комунікацій, інтегровану в поточні BI-інструменти компанії.

Висновки

У даному розділі було представлено реалізацію інтерфейсу користувача для взаємодії з результатами класифікації, а також окреслено напрями подальшого розвитку системи. Застосування сучасних архітектур, зокрема трансформерів, відкриває перспективи для підвищення якості

класифікації завдяки глибшому розумінню мовного контексту. Впровадження ієрархічної класифікації та методів тематичного моделювання дозволяє гнучко адаптувати систему до складної структури користувацьких коментарів і своєчасно виявляти нові проблемні зони. Інтеграція великих мовних моделей для генерації текстових висновків і аналізу тону повідомлень сприяє збагаченню аналітичних можливостей та автоматизації процесу формування інтерпретацій. Розширення сфери застосування класифікаційної моделі на інші джерела зворотного зв’язку сприятиме створенню єдиного аналітичного простору, що забезпечуватиме комплексне розуміння користувацького досвіду. У сукупності ці напрямки формують основу для подальшого вдосконалення та масштабування системи в межах організаційної аналітики.

ВИСНОВКИ

У межах цієї дипломної роботи було розроблено систему автоматичної класифікації користувацьких коментарів з онлайн-опитувань, що дозволяє ефективно структурувати текстові дані для подальшого аналізу. Основною метою дослідження було підвищення якості класифікації за умов обмеженої кількості розмічених даних, що є типовою проблемою для бізнес-даних українською мовою.

У першому розділі було обґрунтовано актуальність задачі класифікації коментарів, здійснено аналіз сучасних методів машинного навчання, зокрема підходів, які дозволяють працювати з обмеженою розміткою: активне навчання, weak supervision, few-shot підходи та навчання із частковим наглядом. Було сформульовано завдання дослідження та визначено його основні етапи.

Другий розділ було присвячено технічним аспектам обробки текстових даних, методам векторизації та побудови моделей. Детально описано класичні алгоритми машинного навчання (логістична регресія, випадковий ліс, SVM, XGBoost) і засоби для роботи з ними, включаючи бібліотеки Python та інструменти для візуалізації й інтеграції через API.

У третьому розділі проаналізовано особливості даних з користувацьких опитувань щодо якості контенту. Встановлено, що коментарі мають нерегулярну структуру, часто містять розмовні вирази, граматичні помилки та змішані мови, що ускладнює автоматичну обробку. Для забезпечення якості вхідних даних виконано очищення текстів та нормалізацію. Також проведено практичну реалізацію та оцінку різних моделей класифікації. Базові моделі на основі TF-IDF продемонстрували хорошу початкову якість. Подальше використання MiniLM дозволило покращити результати за рахунок кращої передачі контексту. Найкращі результати були отримані за допомогою моделі XGBoost у співнавчанні з

логістичною регресією та моделі SVM із застосуванням самонавчання і показали точність 78% та 79% відповідно. Проте порівняно з самонавчанням модель співнавчання ідентифікує більше прикладів для малочисельних класів. Застосування цих підходів значно підвищило ефективність при роботі з обмеженим обсягом розмічених даних. Було визначено оптимальні конфігурації моделей для подальшого впровадження.

У четвертому розділі представлено реалізований інтерфейс для зручного аналізу результатів класифікації у вигляді сукупності графіків, а також окреслено перспективи розвитку системи. Зокрема, розглянуто можливості переходу до трансформерних моделей, запровадження ієрархічної класифікації, тематичного моделювання, аналізу тону повідомлень та генерації підсумкових звітів за допомогою LLM. Також запропоновано розширення системи на інші джерела зворотного зв’язку, що дозволить побудувати більш повну аналітичну картину.

Загалом, результати дипломної роботи підтверджують ефективність використання сучасних підходів до обробки тексту в умовах обмежених ресурсів. Побудована система може бути адаптована до різних бізнес-кейсів, сприяючи автоматизації аналізу відкритих відповідей та прийняттю обґрунтованих управлінських рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. S.P. Lavrii, J.I. Minaieva. Movie recommender system. http://iti.fit.univ.kiev.ua/wp-content/uploads/%D0 %97 %D0 %B1 %D1 %96 %D1 %80 %D0 %BA%D0 %B0_ITI_2023.pdf
2. Ihor Miroshnichenko, Sofia Lavrii. Development of data science technologies in the field of text classification. http://iti.fit.univ.kiev.ua/wp-content/uploads/%D0 %97 %D0 %B1 %D1 %96 %D1 %80 %D0 %BA%D0 %B0–19_12_2024_ITI_2024-%D0 %B5.pdf
3. Amani Aljehani, Syed Hamid Hasan and Usman Ali Khan. Advancing Text Classification: A Systematic Review of Few-Shot Learning Approaches. https://www.researchgate.net/profile/Syed-Hasan-100/publication/387424784_Advancing_Text_Classification_A_Systematic_Review_of_Few-Shot_Learning_Approaches/links/676d3cee894c5520852b62a1/Advancing-Text-Classification-A-Systematic-Review-of-Few-Shot-Learning-Approaches.pdf
4. Ruiying Geng, Binhua Li, Yongbin Li, Xiao-Dan Zhu, Ping Jian, Jian Sun. Induction Networks for Few-Shot Text Classification. https://arxiv.org/pdf/1902.10482v2
5. Zalutska, O., Molchanova, M., Sobko, O., Mazurets, O., Pasichnyk, O., Barmak, O., & Krak, I. (2023). Method for Sentiment Analysis of Ukrainian-Language Reviews in E-Commerce Using RoBERTa Neural Network. In COLINS (1) (pp. 344–356). https://ceur-ws.org/Vol-3387/paper26.pdf
6. Analyzing the effectiveness of semi-supervised learning approaches for opinion spam classification. Alexander Ligthart, Cagatay Catal, Bedir Tekinerdogan. https://www.sciencedirect.com/science/article/pii/S1568494620309625

7. A brief introduction to weakly supervised learning, Zhi-Hua Zhou, National Science Review, Volume 5, Issue 1, January 2018, Pages 44–53, https://doi.org/10.1093/nsr/nwx106 Published: 25 August 2017
8. Rahul Parundekar, Roland Szabo, and Tobias Plötz, No labels are all you need – how to build NLP models using little to no annotated data: https://www.tribe.ai/applied-ai/no-labels-are-all-you-need-how-to-build-nlp-models-using-little-to-no-annotated-data
9. Snorkel: https://www.snorkel.org/get-started/
10. Prototypical Networks for Few-shot Learning. Jake Snell, Kevin Swersky, Richard S. Zemel. https://arxiv.org/abs/1703.05175
11. Babenko, D. (2020). Determining sentiment and important properties of Ukrainian-language user reviews. https://er.ucu.edu.ua/server/api/core/bitstreams/905a577f-791c-4922-bf31–5bca0d5e4fb0/content
12. Abhi Desai. Active Learning Strategies for Efficient Text Classification. https://www.researchgate.net/publication/390608504_Active_Learning_Strategies_for_Efficient_Text_Classification
13. Ziyu Guan, Long Chen, Wei Zhao, Yi Zheng, Shulong Tan, Deng Cai. Weakly-Supervised Deep Learning for Customer Review Sentiment Classification. https://www.ijcai.org/Proceedings/16/Papers/523.pdf
14. Zhu, Xiaojin. Semi-Supervised Learning Literature Survey. https://minds.wisconsin.edu/handle/1793/60444
15. Data preprocessing and tokenization techniques for technical Ukrainian texts. Sergii V. Mashtalir, Oleksandr V. Nikolenko. https://www.researchgate.net/publication/374400742_Data_preprocessing_and_tokenization_techniques_for_technical_Ukrainian_texts
16. The influence of preprocessing on text classification using a bag-of-words representation. Yaakov HaCohen-Kerner, Daniel Miller, Yair Yigal. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0232525

17. An Evaluation of Preprocessing Techniques for Text Classification. Ammar Ismael Kadhim. https://www.researchgate.net/profile/Ammar-Kadhim-4/publication/329339664_An_Evaluation_of_Preprocessing_Techniques_for_Text_Classification/links/5c1b6aa6a6fdccfc705ae648/An-Evaluation-of-Preprocessing-Techniques-for-Text-Classification.pdf
18. NLTK: Natural Language Toolkit. https://www.nltk.org/
19. SciPy. https://scipy.org/
20. Stanza – A Python NLP Package for Many Human Languages. https://stanfordnlp.github.io/stanza/
21. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Nils Reimers, Iryna Gurevych. https://arxiv.org/pdf/1908.10084
22. Paraphrase Multilingual MiniLM L12 V2: Multilingual Sentence Embeddings. https://dataloop.ai/library/model/sentence-transformers_paraphrase-multilingual-minilm-l12-v2/#model-overview
23. MTEB: Massive Text Embedding Benchmark. Niklas Muennighoff, Nouamane Tazi, Loic Magne, Nils Reimers. https://aclanthology.org/2023.eacl-main.148/
24. The Scandinavian Embedding Benchmarks: Comprehensive Assessment of Multilingual and Monolingual Text Embedding. Kenneth Enevoldsen, Márton Kardos, Niklas Muennighoff, Kristoffer Laigaard Nielbo. https://arxiv.org/abs/2406.02396
25. SentenceTransformers Documentation https://sbert.net/index.html
26. Sentence Transformers: https://huggingface.co/sentence-transformers
27. paraphrase-multilingual-MiniLM-L12-v2: https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2
28. Google Translate API: https://pypi.org/project/googletrans/
29. Список стоп-слів: https://raw.githubusercontent.com/skupriienko/Ukrainian-Stopwords/master/stopwords_ua.txt

Додатки

Додаток А

Код для перекладу текстів на українську мову.

```
from googletrans import Translator
import time
from tqdm.notebook import tqdm

# 1. Ініціалізувати перекладач
translator = Translator()

def translate_to_ukrainian(text):
    try:
        if pd.isnull(text) or text.strip() == '':
            return text
        translated = translator.translate(text, dest='uk')
        # time.sleep(0.5)  # невелика затримка, щоб не банили
        return translated.text
    except Exception as e:
        print(f"Translation error: {e}")
        return text
```

Додаток Б

Функція співнавчання.

```python
def co_training(X_labeled, y_labeled, X_unlabeled, X_test, y_test, model1, model2, n_iter=10, top_n=20):
    model1 = clone(model1)
    model2 = clone(model2)

    # Ініціалізація тренувальних даних для кожної моделі
    X1, y1 = X_labeled.copy(), y_labeled.copy()
    X2, y2 = X_labeled.copy(), y_labeled.copy()

    for i in range(n_iter):
        print(f"\nІтерація {i+1}")

        model1.fit(X1, y1)
        model2.fit(X2, y2)

        # Прогнозування на нерозмічених
        probs1 = model1.predict_proba(X_unlabeled)
        probs2 = model2.predict_proba(X_unlabeled)

        # Найвпевненіші приклади
        conf1 = np.max(probs1, axis=1)
        conf2 = np.max(probs2, axis=1)

        top_idx1 = np.argsort(conf1)[-top_n:]
        top_idx2 = np.argsort(conf2)[-top_n:]

        # Передбачення міток
        new_X1 = X_unlabeled[top_idx2]
        new_y1 = model2.predict(new_X1)

        new_X2 = X_unlabeled[top_idx1]
        new_y2 = model1.predict(new_X2)

        # Додавання до тренувального набору
        X1 = np.concatenate([X1, new_X1])
        y1 = np.concatenate([y1, new_y1])

        X2 = np.concatenate([X2, new_X2])
        y2 = np.concatenate([y2, new_y2])

        # Видаляємо використані приклади з unlabeled
        used_indices = np.union1d(top_idx1, top_idx2)
        X_unlabeled = np.delete(X_unlabeled, used_indices, axis=0)

        print(f"Додано {len(top_idx1)} з model1, {len(top_idx2)} з model2. Залишилось: {len(X_unlabeled)}")
        if len(X_unlabeled) < top_n:
            print("Недостатньо прикладів для наступної ітерації")
            break

    # Фінальне навчання та оцінка
    model1.fit(X1, y1)
    model2.fit(X2, y2)

    pred1 = model1.predict(X_test)
    pred2 = model2.predict(X_test)

    print("\nОцінка для моделі 1:")
    print(classification_report(y_test, pred1))

    print("Оцінка для моделі 2:")
    print(classification_report(y_test, pred2))
    return model1, model2
```

Додаток В

Порівняльна таблиця для класу "Модераційні пропозиції":

Модель	Векторизація	Навчання	Precision	Recall	F1-score
Логістична регресія	TF-IDF	Лише розмічені дані	46%	36%	40%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	53%	62%	57%
	paraphrase-MiniLM-L12-v2	Самонавчання	**56%**	**69%**	**62%**
	paraphrase-MiniLM-L12-v2	Співнавчання з SVM	58%	60%	59%
	paraphrase-MiniLM-L12-v2	Співнавчання з XGBoost	58%	57%	57%
Випадковий ліс	TF-IDF	Лише розмічені дані	40%	9%	14%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	**80%**	**9%**	**16%**
	paraphrase-MiniLM-L12-v2	Самонавчання	47%	8%	14%
SVM	TF-IDF	Лише розмічені дані	52%	26%	34%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	55%	60%	57%
	paraphrase-MiniLM-L12-v2	Самонавчання	**60%**	**74%**	**66%**
	paraphrase-MiniLM-L12-v2	Співнавчання з LR	58%	57%	57%
	paraphrase-MiniLM-L12-v2	Співнавчання з XGBoost	61%	55%	58%
XGBoost	TF-IDF	Лише розмічені дані	43%	26%	32%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	64%	34%	44%
	paraphrase-MiniLM-L12-v2	Самонавчання	50%	48%	49%
	paraphrase-MiniLM-L12-v2	Співнавчання з LR	**58%**	**58%**	**58%**
	paraphrase-MiniLM-L12-v2	Співнавчання з SVM	48%	38%	42%

Додаток Г

Порівняльна таблиця для класу "Не формат":

Модель	Векторизація	Навчання	Precision	Recall	F1-score
Логістична регресія	TF-IDF	Лише розмічені дані	42%	62%	50%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	54%	67%	60%
	paraphrase-MiniLM-L12-v2	Самонавчання	**62%**	**71%**	**66%**
	paraphrase-MiniLM-L12-v2	Співнавчання з SVM	52%	58%	55%
	paraphrase-MiniLM-L12-v2	Співнавчання з XGBoost	50%	58%	54%
Випадковий ліс	TF-IDF	Лише розмічені дані	37%	60%	46%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	71%	35%	47%
	paraphrase-MiniLM-L12-v2	Самонавчання	**93%**	**47%**	**62%**
SVM	TF-IDF	Лише розмічені дані	71%	50%	59%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	58%	69%	63%
	paraphrase-MiniLM-L12-v2	Самонавчання	78%	60%	68%
	paraphrase-MiniLM-L12-v2	Співнавчання з LR	60%	58%	59%
	paraphrase-MiniLM-L12-v2	Співнавчання з XGBoost	**67%**	**67%**	**67%**
XGBoost	TF-IDF	Лише розмічені дані	39%	60%	48%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	65%	54%	59%
	paraphrase-MiniLM-L12-v2	Самонавчання	**85%**	**60%**	**71%**
	paraphrase-MiniLM-L12-v2	Співнавчання з LR	67%	47%	55%
	paraphrase-MiniLM-L12-v2	Співнавчання з SVM	67%	42%	51%

Додаток Д

Порівняльна таблиця для класу "Похвала від клієнта":

Модель	Векторизація	Навчання	Precision	Recall	F1-score
Логістична регресія	TF-IDF	Лише розмічені дані	98%	84%	90%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	**96%**	**92%**	**94%**
	paraphrase-MiniLM-L12-v2	Самонавчання	97%	88%	92%
	paraphrase-MiniLM-L12-v2	Співнавчання з SVM	99%	88%	93%
	paraphrase-MiniLM-L12-v2	Співнавчання з XGBoost	99%	87%	92%
Випадковий ліс	TF-IDF	Лише розмічені дані	90%	89%	89%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	**92%**	**95%**	**94%**
	paraphrase-MiniLM-L12-v2	Самонавчання	97%	85%	91%
SVM	TF-IDF	Лише розмічені дані	98%	84%	90%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	**96%**	**90%**	**93%**
	paraphrase-MiniLM-L12-v2	Самонавчання	96%	89%	92%
	paraphrase-MiniLM-L12-v2	Співнавчання з LR	99%	88%	93%
	paraphrase-MiniLM-L12-v2	Співнавчання з XGBoost	98%	88%	93%
XGBoost	TF-IDF	Лише розмічені дані	91%	89%	90%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	**93%**	**96%**	**94%**
	paraphrase-MiniLM-L12-v2	Самонавчання	95%	91%	93%
	paraphrase-MiniLM-L12-v2	Співнавчання з LR	97%	90%	93%
	paraphrase-MiniLM-L12-v2	Співнавчання з SVM	95%	92%	93%

Додаток Е

Порівняльна таблиця для класу "Продуктові проблеми/пропозиції":

Модель	Векторизація	Навчання	Precision	Recall	F1-score
Логістична регресія	TF-IDF	Лише розмічені дані	64%	70%	67%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	**84%**	**68%**	**75%**
	paraphrase-MiniLM-L12-v2	Самонавчання	80%	68%	74%
	paraphrase-MiniLM-L12-v2	Співнавчання з SVM	75%	70%	73%
	paraphrase-MiniLM-L12-v2	Співнавчання з XGBoost	75%	70%	73%
Випадковий ліс	TF-IDF	Лише розмічені дані	**65%**	**78%**	**71%**
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	60%	78%	67%
	paraphrase-MiniLM-L12-v2	Самонавчання	50%	99%	67%
SVM	TF-IDF	Лише розмічені дані	55%	90%	69%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	**78%**	**76%**	**77%**
	paraphrase-MiniLM-L12-v2	Самонавчання	69%	84%	76%
	paraphrase-MiniLM-L12-v2	Співнавчання з LR	75%	70%	73%
	paraphrase-MiniLM-L12-v2	Співнавчання з XGBoost	76%	69%	72%
XGBoost	TF-IDF	Лише розмічені дані	65%	66%	66%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	69%	87%	77%
	paraphrase-MiniLM-L12-v2	Самонавчання	**65%**	**95%**	**77%**
	paraphrase-MiniLM-L12-v2	Співнавчання з LR	62%	94%	75%
	paraphrase-MiniLM-L12-v2	Співнавчання з SVM	54%	96%	69%

Додаток Ж

Порівняльна таблиця для класу "Шахраї":

Модель	Векторизація	Навчання	Precision	Recall	F1-score
Логістична регресія	TF-IDF	Лише розмічені дані	62%	50%	56%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	52%	70%	60%
	paraphrase-MiniLM-L12-v2	Самонавчання	52%	73%	60%
	paraphrase-MiniLM-L12-v2	Співнавчання з SVM	64%	75%	69%
	paraphrase-MiniLM-L12-v2	Співнавчання з XGBoost	**78%**	**75%**	**77%**
Випадковий ліс	TF-IDF	Лише розмічені дані	100%	30%	46%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	**90%**	**45%**	**60%**
	paraphrase-MiniLM-L12-v2	Самонавчання	100%	23%	37%
SVM	TF-IDF	Лише розмічені дані	86%	30%	44%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	74%	70%	72%
	paraphrase-MiniLM-L12-v2	Самонавчання	75%	68%	71%
	paraphrase-MiniLM-L12-v2	Співнавчання з LR	**78%**	**75%**	**77%**
	paraphrase-MiniLM-L12-v2	Співнавчання з XGBoost	88%	58%	70%
XGBoost	TF-IDF	Лише розмічені дані	62%	40%	48%
	paraphrase-MiniLM-L12-v2	Лише розмічені дані	76%	65%	70%
	paraphrase-MiniLM-L12-v2	Самонавчання	82%	64%	72%
	paraphrase-MiniLM-L12-v2	Співнавчання з LR	**81%**	**71%**	**76%**
	paraphrase-MiniLM-L12-v2	Співнавчання з SVM	88%	58%	70%