ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Система обробки голосових повідомлень для гарячої лінії на основі методів машинного навчання»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Олександр ЛЕВЧЕНКО
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-62
Олександр ЛЕВЧЕНКО

Керівник: Тимур ДОВЖЕНКО
к.т.н.

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення

_____ Ірина ЗАМРІЙ
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Левченку Олександру Олександровичу

1. Тема кваліфікаційної роботи: «Система обробки голосових повідомлень для гарячої лінії на основі методів машинного навчання»
керівник кваліфікаційної роботи Тимур ДОВЖЕНКО, к.т.н., доцент кафедри ІПЗ
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024р. №320.
2. Строк подання кваліфікаційної роботи «19» грудня 2024р.
3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, методи машинного навчання, методи обробки голосових повідомлень.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної області та проблеми
2. Дослідження методів для обробки голосових повідомлень

3. Аналіз технологій та методів машинного навчання

4. Реалізація системи обробки голосових повідомлень

5. Перелік графічного матеріалу: *презентація*

1. Мета, об’єкта та предмет дослідження.
2. Таблиця порівняння протестованих методів.
3. Svm та naive bayes у рамках методу StackingClassifier.
4. Розрахунок на основі байєсової теореми.
5. Лінійний алгоритм роботи системи обробки голосових повідомлень.
6. Кодова реалізація проаналізованих методів.
7. Відео роботи системи.
8. Висновки.

6. Дата видачі завдання«19» жовтня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	19.10-24.10.2024	
2	Огляд доступних методів предметної області	25.10-05.11.2024	
3	Аналіз наявної науково-технічної літератури	06.11-09.11.2024	
4	Аналіз існуючих методів машинного навчання	10.11-14.11.2024	
5	Тестування ефективності проаналізованих методів	15.11-18.11.2024	
6	Розроблення системи на основі обраного методу	19.11-21.11.2024	
7	Оформлення роботи: вступ, висновки, реферат	22.11-24.11.2024	
8	Розробка демонстраційних матеріалів	25.11-26.11.2024	
9	Попередній захист роботи	27.11-12.12.2024	

Здобувач вищої освіти _____ Олександр ЛЕВЧЕНКО
(підпис)

Керівник
кваліфікаційної роботи _____ Тимур ДОВЖЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 78 стор., 2 табл., 14 рис., 22 джерел.

Мета роботи - покращити процес обробки інформації з голосових повідомлень автоматизованої гарячої лінії шляхом підвищення їх релевантності класифікації на основі методів машинного навчання.

Об'єкт дослідження – процес обробки голосових повідомлень для автоматизованої гарячої лінії.

Предмет дослідження – методи машинного навчання, які застосовуються для обробки голосових повідомлень.

Проаналізовано наукову літературу та існуючі сучасні підходи до обробки голосових повідомлень. Проведено порівняння методів машинного навчання та їх комбінація в контексті їх ефективності для заданої задачі. Здійснено збір і попередня обробка голосових даних, досліджено та натренованно моделі машинного навчання. Оцінено продуктивності моделей за допомогою метрик точності, F-міри, точності та повноти. Проведено статистичний аналіз отриманих результатів та розробка рекомендацій що до них.

КЛЮЧОВІ СЛОВА: ОБРОБКА ГОЛОСОВИХ ПОВІДОМЛЕНЬ, АВТОМАТИЗОВАНА ГАРЯЧА ЛІНІЯ, РЕЛЕВАНТНІСТЬ КЛАСИФІКАЦІЇ, МАШИННЕ НАВЧАННЯ, АНАЛІЗ ДАНИХ, КЛАСИФІКАЦІЯ, СТАТИСТИЧНИЙ АНАЛІЗ, ТОЧНІСТЬ, F-МІРА, ПОПЕРЕДНЯ ОБРОБКА.

ABSTRACT

Text part of the master's qualificationwork: 78 pages, 14 pictures2 table, 22 sources.

The purpose of the work - improve the process of processing information from voice messages of an automated hotline by increasing their relevance of classification based on machine learning methods.

Object of research - the process of processing voice messages for an automated hotline.

Subject of research - study is machine learning methods used to process voice messages.

Summary of the work: the scientific literature and existing modern approaches to voice message processing are analyzed. Machine learning methods and their combination are compared in the context of their effectiveness for a given task. Voice data was collected and pre-processed, machine learning models were investigated and trained. The performance of the models was evaluated using the metrics of accuracy, F-measure, precision, and completeness. The results were statistically analyzed and recommendations were developed.

KEYWORDS: VOICE MESSAGE PROCESSING, AUTOMATED HOTLINE, CLASSIFICATION RELEVANCE, MACHINE LEARNING, DATA ANALYSIS, CLASSIFICATION, STATISTICAL ANALYSIS, ACCURACY, F-MEASURE, PREPROCESSING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ML - Machine learning.

AI - Artificial intelligence.

DTW - Dynamic time warping.

HMM - Hidden Markov Models.

GMM - Gaussian Mixture Models.

SVM - Support Vector Machines.

ВСТУП

В сучасному світі обробка та аналіз голосових повідомлень набуває все більшого значення у різних сферах, включаючи бізнес, обслуговування клієнтів, освіту та медицину. Ефективна автоматизація процесів обробки голосових повідомлень дозволяє значно підвищити продуктивність, зменшити навантаження на персонал і забезпечити якісний аналіз отриманих даних.

Однією з найактуальніших задач є класифікація повідомлень за їхнім змістом (категоріями) та емоційною складовою (полярністю). Це дозволяє системам обробки інформації автоматично розподіляти вхідні запити, оперативно реагувати на них і відстежувати задоволеність користувачів. Проте для досягнення високої точності обробки необхідно використовувати сучасні алгоритми машинного навчання, здатні адаптуватися до специфіки текстових даних.

Мета роботи - покращити процес обробки інформації з голосових повідомлень автоматизованої гарячої лінії шляхом підвищення їх релевантності класифікації на основі методів машинного навчання.

Об'єкт дослідження – процес обробки голосових повідомлень для автоматизованої гарячої лінії.

Предмет дослідження – методи машинного навчання, які застосовуються для обробки голосових повідомлень.

У процесі розробки було реалізовано кілька ключових етапів:

- Попередня обробка даних – очищення тексту від зайвих символів, видалення стоп-слів, лематизація та додавання доменних термінів.
- Навчання моделей класифікації – тестування п'яти класичних алгоритмів машинного навчання (SVM, k-NN, Naive Bayes, Logistic Regression, Decision Trees) та вибір оптимального підходу.
- Розробка ансамблевої моделі – інтеграція методів SVM і Naive Bayes, що забезпечило підвищення точності класифікації.

- Інтеграція з базою даних – створення структури MySQL для збереження тексту повідомлення, його категорії, полярності та часу створення.
- Створення повноцінної системи обробки голосових повідомлень – реалізація процесу від розпізнавання голосу до збереження результатів у базі даних.

Актуальність даної роботи обумовлена постійним зростанням обсягу голосових запитів у сучасному світі та потребою в автоматизації їх обробки. Розроблена система може бути адаптована для використання в різних галузях, таких як служби підтримки клієнтів, освітні платформи, медичні консультативні центри тощо.

Таким чином, ця робота спрямована на створення інноваційного інструменту, який не лише спрощує процес обробки голосових повідомлень, але й дозволяє досягти високої точності аналізу завдяки використанню передових технологій машинного навчання.

1 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ОБРОБКИ ГОЛОСОВИХ ПОВІДОМЛЕНЬ І ТЕХНОЛОГІЙ МАШИННОГО НАВЧАННЯ

1.1 Загальна характеристика сучасних систем обробки голосу

Обробка голосових даних - це процес аналізу, інтерпретації та використання аудіо сигналів, які представляють звукові записи мовлення або інших звуків. Це важлива галузь для багатьох застосувань, включаючи розпізнавання мови, аудіо аналітику, розуміння мови та багато інших сценаріїв.

Сучасні системи обробки голосу є невід'ємною частиною технологічного прогресу та активно застосовуються у різних галузях. Їх основною метою є автоматизація взаємодії між людиною та машиною за допомогою голосових команд. Завдяки цьому користувачі отримують більш зручний доступ до технологій, а компанії можуть оптимізувати процеси, скоротити витрати та підвищити ефективність роботи.

Функції таких систем варіюються від розпізнавання мови до синтезу тексту у голос. Розпізнавання мови дозволяє перетворювати мовлення у текст, що широко використовується у голосових помічниках, транскрипції записів та обробці даних кол-центрів. Синтез мови, навпаки, дає можливість створювати голос на основі тексту, що використовується у системах озвучування повідомлень, навігаційних додатках і навіть у пристроях для людей з обмеженими можливостями.

Однією з важливих функцій сучасних систем є аналіз емоцій у голосі. Це дозволяє виявляти емоційний стан користувача, що є корисним у сфері психології, маркетингу або навіть у роботі з кол-центрами. Деякі системи також здатні ідентифікувати користувача за його голосом, що використовується для забезпечення безпеки, наприклад, у фінансових установах або в системах контролю доступу.

Основні галузі використання систем обробки голосу включають побутову електроніку, медицину, транспорт, освіту та розваги. У побуті вони застосовуються у вигляді голосових помічників, таких як Siri або Alexa, які допомагають

виконувати різні задачі – від пошуку інформації до управління "розумним домом". У медицині такі системи допомагають лікарям диктувати медичні записи, а пацієнтам — взаємодіяти з технологіями через голос. У транспортній сфері вони інтегруються в системи навігації та управління автомобілями.

Технологічна основа сучасних систем обробки голосу включає машинне навчання та глибоке навчання, що дозволяє досягати високої точності в аналізі мовних даних. Використання нейронних мереж, таких як моделі типу Transformer, дозволяє обробляти складні мовні структури. Більшість сучасних систем підтримується хмарними сервісами, такими як Google Cloud Speech-to-Text або Amazon Transcribe, що забезпечує масштабованість і високу швидкість обробки даних.

Такі системи також використовуються для аналізу та оптимізації телефонних розмов у кол-центрах, допомагаючи оцінювати якість обслуговування клієнтів. У сфері освіти вони знаходять застосування у мовних додатках для вивчення іноземних мов або автоматизації запису лекцій. У розважальній індустрії ці системи допомагають створювати інтерактивні ігри, автоматизувати субтитрування або керувати медіаплеєрами через голос.

Ринок роспізнавання мовлення:

- Очікувалось, що розмір ринку розпізнавання мовлення досягне 7,14 мільярдів доларів США у 2024 році.
- Розмір ринку демонструватиме річний темп зростання (CAGR 2024-2030) на 14,24%, що призведе до того, що до 2030 року обсяг ринку становитиме 15,87 млрд доларів США.
- У глобальному порівнянні найбільший розмір ринку буде в Сполучених Штатах (1903,00 млн доларів США у 2024 році).

Завдяки універсальності та адаптивності, сучасні системи обробки голосу продовжують удосконалюватися і впроваджуватися у все більше сфер людської діяльності, роблячи технології більш доступними і зручними.

Обробка голосу являє собою складні програмно-апаратні комплекси, які розробляються для вирішення завдань автоматизації взаємодії людини і технологій

через голос. Їх основна мета полягає у тому, щоб перетворювати голосову інформацію на текст, аналізувати її зміст, а також реагувати на запити користувачів. Така функціональність дозволяє знижувати витрати часу та підвищувати ефективність процесів у різних сферах діяльності.

Базовим компонентом систем обробки голосу є розпізнавання мови. Ця технологія відповідає за перетворення мовленнєвого сигналу у текст. Для цього використовується акустична обробка голосу, яка включає шумозаглушення, виділення ключових мовленнєвих характеристик та їх перетворення на послідовність текстових символів. Особливу роль у цьому процесі відіграють алгоритми машинного навчання, зокрема глибокі нейронні мережі, які навчаються на великих обсягах даних для досягнення високої точності розпізнавання.

Ще одним важливим етапом є аналіз змісту тексту, який здійснюється за допомогою технологій обробки природної мови (NLP). На цьому етапі система визначає основні наміри користувача, аналізує семантику висловлювань і навіть виявляє емоційний стан мовця. Наприклад, система може класифікувати запит як "запит статусу замовлення", "скарга" або "подяка". Це дозволяє автоматизувати подальшу маршрутизацію запитів та значно скоротити час реагування.

Інший аспект функціонування таких систем є маршрутизація інформації, яка забезпечує передачу запитів до відповідних відділів чи підсистем. Наприклад, у системах автоматизованих кол-центрів це можуть бути модулі для обробки платежів, перевірки статусу замовлень чи технічної підтримки. Також маршрутизація використовується для інтеграції з базами даних та CRM-системами, що дозволяє зберігати і аналізувати інформацію про користувачів.

Крім цього, сучасні системи включають функцію синтезу мови, яка дає змогу перетворювати текст на природне мовлення. Це важливо для створення зручної взаємодії між користувачем і системою, особливо в голосових помічниках, навігаторах чи автоматизованих відповідях у телефонних системах. Синтез мови використовує моделі, що здатні враховувати тональність, тембр та інтонацію, забезпечуючи природне звучання.

Також ще одним із важливих компонентів є ідентифікація користувача за голосом, яка використовується для забезпечення безпеки. Ця технологія аналізує унікальні характеристики голосу, що дозволяє точно ідентифікувати особу. Такі системи часто застосовуються в банківських установах, для управління доступом до конфіденційної інформації або для підтвердження особистості.

Основні функції систем обробки голосу:

- Перетворення мовлення у текст (ASR).
- Аналіз тексту, розпізнавання намірів та класифікація запитів (NLP).
- Виведення тексту у вигляді голосу (TTS).
- Акустична обробка: шумозаглушення, нормалізація гучності.
- Забезпечення безпеки через ідентифікацію користувача за голосом.

Таким чином, системи обробки голосу інтегрують кілька технологій, включаючи машинне навчання, аналіз природної мови та розпізнавання сигналів. Завдяки цьому вони знаходять застосування у кол-центрах, голосових помічниках, системах безпеки та багатьох інших сферах, де потрібна швидка й точна взаємодія з користувачем через голос.

1.1.1 Виклики в роботі систем обробки голосу

Однією з основних проблем у роботі сучасних систем обробки голосу є необхідність ефективно працювати з низькоякісними звуковими даними. Часто запис мовлення відбувається в умовах, далеких від ідеальних: на фоні шуму, у переповнених місцях чи за допомогою пристроїв, які не забезпечують достатньої якості запису. У таких випадках система може не лише втратити частину інформації, але й зробити помилкові висновки через спотворення сигналу.

Шумове забруднення є одним із найпоширеніших викликів. Воно може походити від фонових розмов, звуків транспорту чи навіть технічних перешкод, таких як вітер чи низька якість мікрофона. Наприклад, у громадському транспорті користувач, звертаючись до голосового помічника, може не отримати відповіді через те, що система не здатна відокремити корисний сигнал від шуму.

Ще однією проблемою є різноманітність акцентів і діалектів. Системи, створені на основі стандартної мови, часто не враховують регіональних чи культурних особливостей вимови. Це призводить до помилок у розпізнаванні навіть у простих запитах. Наприклад, англійська мова має безліч варіантів вимови: британський, американський, австралійський. Аналогічні проблеми спостерігаються й для української мови, яка також має свої діалекти й відмінності у вимові.

Крім цього, нерівномірність мовлення та мовні дефекти ускладнюють процес аналізу. Люди можуть говорити швидко або повільно, робити паузи, використовувати слова-паразити, або мати індивідуальні особливості, як-от заїкання чи нечітка артикуляція. Системи, які орієнтовані на стандартизоване мовлення, часто не можуть впоратися з такими особливостями, що призводить до зниження точності.

Не менш важливим фактором є якість обладнання. Наприклад, дешеві мікрофони можуть створювати додаткові перешкоди, що спотворюють мовленнєвий сигнал. Це особливо критично для мобільних пристроїв, де якість запису може варіюватися залежно від моделі.

Щоб подолати ці виклики, у сучасних системах використовуються різноманітні технології. Алгоритми шумозаглушення допомагають відокремлювати мовленнєвий сигнал від фонового шуму. Використання великих тренувальних вибірок, які включають дані з різними акцентами, діалектами та шумами, дозволяє підвищити адаптивність моделей. Крім того, розвиток нейронних мереж і глибокого навчання дозволяє створювати моделі, здатні враховувати особливості мовлення навіть у складних умовах.

Для подолання цих проблем застосовуються такі методи:

- Для шумозаглушення, використання алгоритмів фільтрації, таких як спектральне субтрактивне шумозаглушення, або сучасні методи на основі нейронних мереж, наприклад, DNN (Deep Neural Networks).
- Інтеграція аудіоданих із різними акцентами, шумами та дефектами у тренувальні вибірки моделей.

- Застосування моделей, які можуть "налаштовуватись" на специфіку мовлення певного користувача або середовища.
- Для поліпшення якості запису використовують інтеграцію з апаратними рішеннями, такими як мікрофони з високим динамічним діапазоном, для зменшення викривлення сигналу.
- Моделі для емоційного аналізу інтегрують додаткові компоненти для врахування емоційних змін у голосі, що дозволяє компенсувати вплив емоцій на точність розпізнавання.

Попри всі зусилля, обробка низькоякісного звуку залишається однією з найскладніших задач у цій галузі. Однак постійний розвиток технологій та вдосконалення алгоритмів обіцяють зробити системи обробки голосу ще більш точними та надійними у майбутньому.

1.1.2 Мультиканальність та масштабованість

Мультиканальність і масштабованість є ключовими аспектами сучасних систем обробки голосу, що визначають їх здатність ефективно працювати в умовах зростаючих обсягів даних і різноманітності джерел.

Мультиканальність означає, що система здатна одночасно працювати з кількома каналами зв’язку або потоками даних. У сучасному світі голосові дані можуть надходити з різних джерел, таких як телефонні дзвінки, голосові повідомлення, конференц-дзвінки, мобільні додатки чи навіть IoT-пристрої. Мультиканальна система обробки голосу дозволяє інтегрувати ці дані в єдину платформу для подальшої обробки. Наприклад, у кол-центрах система може одночасно обробляти телефонні дзвінки від клієнтів і голосові запити через мобільний додаток компанії, забезпечуючи зручність для користувачів і оперативність обробки запитів.

Крім того, мультиканальність підтримує різні формати звукових даних, такі як аудіопотоки в реальному часі або заздалегідь записані файли. Це важливо для систем, які працюють із голосовими даними у великих масштабах. Наприклад, під

час транскрибування засідань або аналізу розмов кол-центрів система може одночасно обробляти десятки аудіопотоків, забезпечуючи точність та швидкість.

Масштабованість, своєю чергою, характеризує здатність системи адаптуватися до збільшення обсягів даних чи кількості користувачів без значного зниження продуктивності. Це особливо актуально для хмарних рішень, які дозволяють динамічно збільшувати ресурси для обробки голосу залежно від навантаження. Наприклад, у пікові години роботи кол-центру система може залучати додаткові сервери або обчислювальні ресурси, щоб обробити всі запити вчасно.

Один із ключових викликів у досягненні масштабованості — це забезпечення стабільної продуктивності навіть при значному збільшенні навантаження. Для цього використовуються розподілені системи обробки, які розподіляють навантаження між кількома вузлами. Крім того, сучасні системи активно використовують хмарні технології, такі як Amazon Web Services, Microsoft Azure або Google Cloud, які дозволяють динамічно змінювати обчислювальну потужність відповідно до потреб.

Мультиканальність і масштабованість тісно пов’язані між собою, оскільки ефективна робота з багатьма каналами потребує значних обчислювальних ресурсів і здатності до масштабування. Наприклад, система, яка обробляє дзвінки клієнтів з усього світу, має враховувати не лише кількість дзвінків, але й різноманітність мов, часових зон і типів даних.

Таким чином, мультиканальність і масштабованість є невід’ємними характеристиками сучасних систем обробки голосу, що забезпечують їх ефективність, адаптивність і надійність у різноманітних умовах використання. Ці характеристики дозволяють системам працювати з великими обсягами даних, інтегрувати різні джерела інформації та залишатися стабільними навіть при високих навантаженнях.

1.2 Технології машинного навчання для обробки голосу

Розпізнавання мови — це складний процес, який включає перетворення звукових сигналів у текст. На ранніх етапах розвитку цієї технології широко використовувалися класичні підходи, що базуються на статистичних методах, зокрема моделі прихованих Марковських процесів (HMM) у поєднанні з гаусовими сумішевими моделями (GMM). Ці методи домінували в системах розпізнавання мови протягом десятиліть до появи глибоких нейронних мереж.

Приховані Марковські моделі (HMM) є однією з ключових технологій у класичному розпізнаванні мови. Вони добре підходять для задач, де мовленнєвий сигнал можна представити як послідовність станів. Основна ідея HMM полягає у використанні ймовірнісної моделі для опису перетворення звукових сигналів у послідовність текстових символів.

HMM базується на таких принципах:

- Стани моделі відповідають фонемам — базовим звуковим одиницям мови.
- Перехідні ймовірності описують імовірність переходу з одного стану в інший.
- Ймовірність спостережень моделює акустичні характеристики сигналу для кожного стану.

Процес розпізнавання включає дві основні задачі:

- Обчислення ймовірностей - визначення найбільш ймовірної послідовності станів для заданого мовленнєвого сигналу.
- Вибір найбільш відповідного тексту - визначення відповідного тексту для отриманої послідовності станів.

HMM самі по собі не можуть ефективно працювати з акустичними характеристиками сигналу. Для цього використовуються гаусові сумішеві моделі, які дозволяють моделювати розподіл звукових характеристик для кожного стану HMM.

GMM визначає ймовірність того, що певний звуковий сигнал відповідає конкретному стану HMM. Ця модель будується як комбінація кількох нормальних (гаусових) розподілів, які разом описують складні характеристики сигналу.

Поєднання HMM і GMM дозволяє:

- Розділяти мовленнєвий сигнал на короткі сегменти.
- Визначати ймовірність того, що кожен сегмент відповідає певній фонемі чи слову.

Глибокі нейронні мережі (Deep Neural Networks) є сучасною основою багатьох систем розпізнавання мови завдяки їх здатності ефективно моделювати складні залежності у мовленнєвих даних. Серед найбільш поширених архітектур, які використовуються в цій галузі, варто виділити рекурентні нейронні мережі (RNN), довготривалу короткочасну пам’ять (LSTM) і згорткові нейронні мережі (CNN).

Рекурентні нейронні мережі (RNN) спеціально створені для роботи з послідовними даними, такими як мовленнєвий сигнал. Вони враховують часову залежність, що дозволяє їм зберігати інформацію про попередні частини сигналу під час обробки. Проте класичні RNN мають обмеження у запам’ятовуванні довготривалого контексту, що призвело до розвитку більш досконалих моделей, таких як LSTM.

LSTM (Long Short-Term Memory) є вдосконаленням RNN, яке дозволяє моделювати як короткотривалі, так і довготривалі залежності у мовленні. Завдяки своїй здатності ефективно обробляти довгі послідовності, LSTM широко застосовуються в задачах розпізнавання мови, особливо там, де важливий контекст, наприклад, у транскрипції речень.

Згорткові нейронні мережі (CNN) зазвичай використовуються для виділення ключових ознак із спектрограм мовленнєвих сигналів. Вони працюють із двовимірними даними, такими як мел-спектрограми, дозволяючи виділити акустичні патерни, що характерні для певних фонем або слів. CNN часто

поєднуються з RNN або LSTM для створення комплексних гібридних моделей, які враховують і просторові, і часові залежності.

Глибокі нейронні мережі дозволяють досягти значно вищої точності розпізнавання мови порівняно з класичними методами, особливо в умовах шуму, акцентів чи варіативності мовлення. Їхнє використання стало стандартом у таких системах, як голосові помічники (Google Assistant, Siri), автоматизовані кол-центри та системи транскрипції.

1.3 Методи машинного навчання

Машинне навчання (ML) є підгалуззю штучного інтелекту (AI), що дозволяє комп'ютерам навчатися на даних без явного програмування. У процесі навчання моделі машинного навчання знаходять патерни в даних і використовують ці патерни для прийняття рішень або прогнозів. Існує багато різних методів машинного навчання, кожен з яких має свої сильні сторони і підходить для різних типів задач.

- Support Vector Machines (SVM)
- k-Nearest Neighbors (k-NN)
- Decision Trees
- Naive Bayes
- Logistic Regression

1.3.1 Support Vector Machines (SVM)

Support Vector Machines (SVM) — метод машинного навчання, який використовується для класифікації та регресії. Його основна ідея полягає у пошуку гіперплощини, яка максимально розділяє дані різних класів, забезпечуючи найбільший відступ між ними. Для нелінійних даних SVM використовує ядерні функції, які перетворюють дані у простір вищих вимірів, дозволяючи ефективно розділяти складні набори даних. SVM широко застосовується у задачах обробки тексту, розпізнавання образів, біоінформатики та фінансового аналізу. Основними

перевагами є висока точність у високовимірних просторах і стійкість до перенавчання, але метод чутливий до вибору параметрів, потребує нормалізації даних і може бути обчислювально складним для великих наборів даних.

Застосування у різних сферах:

- Класифікація тексту, аналіз тональності чи категоризація документів.
- Фільтрація спаму.
- Виявлення об’єктів у зображеннях.
- Розпізнавання рукописного тексту.
- Класифікація білків, передбачення генів.
- Виявлення шахрайства, аналіз ризиків.
- Розпізнавання мови, класифікація аудіосигналів.

Переваги SVM полягають у його здатності працювати з високовимірними даними та забезпечувати високу точність навіть за наявності складних меж між класами. Метод добре підходить для задач із невеликою кількістю прикладів, оскільки використовує лише підтримувальні вектори, що робить його стійким до перенавчання. Гнучкість SVM забезпечується завдяки ядерним функціям, які дозволяють працювати як із лінійними, так і з нелінійними даними. Проте цей підхід має і свої недоліки. Для ефективної роботи SVM потребує ретельного налаштування параметрів і вибору ядра, що може бути складним у практичних умовах. Крім того, метод є обчислювально витратним, особливо при роботі з великими наборами даних, і потребує нормалізації ознак, оскільки чутливий до масштабу вхідних даних. Незважаючи на ці обмеження, SVM залишається одним із найпотужніших інструментів для задач класифікації та регресії.

1.3.2 k-Nearest Neighbors (k-NN)

k-Nearest Neighbors (k-NN) — це простий і ефективний метод машинного навчання, який використовується для класифікації та регресії. Основна ідея цього алгоритму полягає у визначенні найближчих сусідів для кожного нового зразка на

основі заданої метрики відстані (наприклад, евклідової, мангеттенської чи косинусної). Класифікація виконується за принципом більшості: новий зразок належить до класу, який найчастіше зустрічається серед k найближчих сусідів. У регресії значення нового зразка обчислюється як середнє значення його найближчих сусідів.

Метод k-NN не потребує процесу тренування, оскільки всі обчислення відбуваються під час класифікації чи прогнозу. Це робить його ідеальним для невеликих наборів даних, де важливі інтерпретація та простота реалізації. Однією з основних переваг k-NN є його здатність добре адаптуватися до нелінійних розподілів даних. Проте з ростом розміру даних цей алгоритм стає повільним, оскільки кожен запит вимагає порівняння з усією тренувальною вибіркою. Крім того, k-NN є чутливим до вибору параметра k та метрики відстані, а також до нерівномірного розподілу даних, що може впливати на точність. Незважаючи на ці обмеження, метод широко використовується у класифікації тексту, рекомендаційних системах, розпізнаванні образів і в задачах із невеликими обсягами даних.

1.3.3 Decision Trees

Decision Trees — це метод машинного навчання, який використовується для задач класифікації та регресії. Алгоритм базується на побудові дерева, де кожен вузол представляє тест на певну ознаку, гілки відповідають результатам цього тесту, а листові вузли визначають кінцеве рішення або прогноз. Дерева прийняття рішень працюють за принципом послідовного поділу простору ознак на підмножини, які стають все більш однорідними.

Основна перевага Decision Trees полягає в їх простоті та інтерпретованості. Рішення можна легко пояснити, що робить метод ідеальним для задач, де потрібна прозорість моделей, наприклад, у медицині чи фінансовому аналізі. Крім того, алгоритм може обробляти як числові, так і категоріальні дані та не потребує масштабування ознак.

Проте Decision Trees мають і свої недоліки. Вони схильні до перенавчання, особливо якщо дерево занадто глибоке, що може знижувати точність на тестових даних. Щоб уникнути цього, застосовуються методи обрізання дерев (pruning). Також окремі дерева можуть бути нестійкими до змін у даних, оскільки невеликі зміни у вибірці можуть значно вплинути на структуру дерева. Для подолання цих обмежень часто використовуються ансамблеві методи, такі як Random Forest або Gradient Boosting, які будуються на основі дерев прийняття рішень. Незважаючи на недоліки, Decision Trees залишаються популярними завдяки своїй ефективності, особливо в задачах, де простота та швидкість мають ключове значення.

1.3.4 Naive Bayes

Naive Bayes — це простий і потужний алгоритм машинного навчання, який базується на теоремі Байєса. Він використовується переважно для задач класифікації. Основна ідея методу полягає в обчисленні ймовірності того, що зразок належить до певного класу, враховуючи його ознаки. Алгоритм вважає, що всі ознаки незалежні одна від одної (наївне припущення), що спрощує розрахунки, хоча в реальності ця умова часто не виконується.

Алгоритм відзначається високою швидкістю роботи та ефективністю на великих наборах даних. Він широко використовується для класифікації тексту, наприклад, у задачах фільтрації спаму, аналізу тональності та класифікації документів. Його ключова перевага — це здатність добре працювати навіть з невеликими вибірками даних.

Проте основним недоліком Naive Bayes є його чутливість до незалежності ознак. У задачах, де ознаки значно корелюють між собою, алгоритм може демонструвати знижену точність. Також він не підходить для задач, де складна межа між класами не може бути описана простими ймовірностями. Попри ці обмеження, Naive Bayes залишається важливим інструментом у задачах, де швидкість і простота мають вирішальне значення.

1.3.5 Logistic Regression

Цей метод машинного навчання, який використовується для задач бінарної класифікації, хоча може бути розширений і на багатокласові задачі. Попри назву, логістична регресія виконує не регресію, а класифікацію, передбачаючи ймовірність належності зразка до певного класу. Основою методу є використання логістичної (сигмоїдальної) функції, яка перетворює лінійну комбінацію ознак у значення від 0 до 1, що інтерпретується як ймовірність.

Логістична регресія має низку переваг, зокрема простоту реалізації, ефективність у задачах із лінійно роздільними даними та здатність надавати інтерпретовані результати. Її часто використовують для класифікації текстів, оцінки ризиків, прогнозування клієнтської поведінки та інших задач, де важлива інтуїтивність моделі.

Проте метод має обмеження. Він працює добре лише з лінійно роздільними даними і може демонструвати низьку точність у випадках складних нелінійних взаємозв’язків між ознаками. Для покращення роботи на нелінійних даних часто застосовують техніки розширення простору ознак, наприклад, через поліноміальні функції або ядерні методи. Незважаючи на ці обмеження, логістична регресія залишається базовим методом класифікації завдяки своїй простоті, швидкості та інтерпретованості.

Таблиція 1.1

Порівняння методів машинного навчання

Метод	**Support Vector Machines (SVM)**	**k-Nearest Neighbors (k-NN)**	**Decision Trees**	**Naive Bayes**	**Logistic Regression**
Продуктивність	+	-	+	-	+
Простота реалізації	-	+	+	+	+
Швидкий час навчання	-	-	+	+	+
Інтерпретація	-	+	+	+	+
Стійкість до перенавчання	+	-	-	-	+
Підходить для великого набору даних	+	-	+	+	+

2 АНАЛІЗ МАТЕМАТИЧНИХ МОДЕЛЕЙ ТА МЕТОДІВ МАШИННОГО НАВЧАННЯ

2.1 Математична постановка задачі

Обробка голосових повідомлень набуває все більшої актуальності завдяки зростанню популярності голосових інтерфейсів, автоматизованих кол-центрів та систем штучного інтелекту. Однією з основних задач таких систем є автоматична класифікація тексту, отриманого в результаті перетворення голосових даних. Це дозволяє значно прискорити процес обробки запитів, зменшити навантаження на операторів та підвищити якість обслуговування клієнтів.

Дана задача полягає в класифікації текстів повідомлень, отриманих із голосових записів, за двома ознаками: категорія повідомлення (наприклад, "консультація", "технічна проблема", "запит документів") та емоційна полярність (позитивне або негативне). Таке розділення дозволяє системі визначати як тип запиту, так і емоційний стан користувача, що є важливим для подальшої маршрутизації запиту та автоматичної генерації відповідей.

Для реалізації цієї задачі застосовуються сучасні методи машинного навчання, які дозволяють ефективно класифікувати текстові дані. Особливістю роботи є необхідність врахування специфіки текстів, отриманих із голосових повідомлень, таких як короткі фрази, іноді неформальний стиль, наявність помилок у розпізнаванні мови. Тому побудова моделі потребує якісної попередньої обробки текстів, вибору відповідних ознак (наприклад, використання TF-IDF) та ретельного аналізу ефективності різних моделей класифікації.

Таким чином, задача класифікації текстів із голосових повідомлень є важливим етапом у створенні автоматизованої системи обробки запитів, що дозволяє підвищити точність і швидкість реагування на запити користувачів.

2.2 Формалізація задачі

Задача класифікації текстів, отриманих із голосових повідомлень, формалізується як задача машинного навчання, яка передбачає визначення двох міток: категорії повідомлення та емоційної полярності. Основна мета полягає в побудові моделей, здатних передбачати обидві ці мітки на основі текстових ознак, отриманих із вхідних даних.

Вхідними даними є набір повідомлень D, який можна представити так:

$$D = (x_1, y_{1,1}, y_{1,2}), (x_2, y_{2,1}, y_{2,2}), \dots, (x_n, y_{n,1}, y_{n,2}), \quad (2.1)$$

де x_i — текстове повідомлення i, представлене як послідовність слів;

$y_{i,1}$ — мітка категорії повідомлення i (наприклад, "консультація", "технічна проблема");

$y_{i,2}$ — полярність повідомлення i (позитивне\негативне).

Тексти повідомлень перетворюються у числове представлення за допомогою методу TF-IDF. У результаті кожне повідомлення представляється у вигляді вектора ознак:

$$x_i = [f_1, f_2, \dots, f_m], \quad (2.2)$$

де f_j - значення TF-IDF для j - го слова у повідомленні x_i ;

m — кількість унікальних слів у корпусі текстів.

Формально задача класифікації формулюється як побудова двох функцій:

$$f_1: X \to \{C_1, C_2, \dots, C_k\}, \quad f_2: X \to \{\text{Позитивне,Негативне}\}, \quad (2.3)$$

де f_1 — функція класифікації категорії;

f_2 — функція класифікації полярності;

X — матриця векторів ознак, отриманих із текстів.

Навчання моделей здійснюється шляхом мінімізації функцій втрат. Для багатокласової класифікації категорій застосовується перехресна-ентропія — це функція втрат, яка широко використовується у задачах класифікації, особливо у задачах багатокласової та бінарної класифікації. Вона вимірює різницю між розподілом ймовірностей, передбаченим моделлю, і справжнім розподілом міток у наборі даних. Основна ідея крос-ентропії полягає в тому, щоб оцінити, наскільки добре передбачені ймовірності моделі відповідають істинним міткам.

Формула перехресної ентропії:

$$L_{\text{категорія}} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{k} y_{i,1}^{(c)} \log\left(\hat{y}_{i,1}^{(c)}\right), \tag{2.4}$$

де n — кількість прикладів у наборі даних;

k — кількість класів;

$y_{i,1}^{(c)}$ — істинна мітка для класу c;

$\hat{y}_{i,1}^{(c)}$ — передбачена ймовірність належності повідомлення i до класу c.

Для бінарної класифікації полярності використовується бінарна крос-ентропія:

$$L_{\text{полярність}} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_{i,2}\log\left(\hat{y}_{i,2}\right) + \left(1 - y_{i,2}\right)\log\left(1 - \hat{y}_{i,2}\right)\right], \tag{2.5}$$

де $y_{i,2}$ — істинна мітка полярності для i -го прикладу ($y_{i,2}\,\epsilon\{0,1\}$);

$\hat{y}_{i,2}$ — передбачена ймовірність позитивної полярності.

Оцінка ефективності моделей здійснюється за метриками точності, precision, recall та F1-score, які дозволяють оцінити якість класифікації для кожного класу.

2.3 Математичне представлення моделей машинного навчання

Розглянемо математичні основи моделей машинного навчання, які використовуються для класифікації текстів із голосових повідомлень. До розгляду включені п’ять основних моделей: Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), Decision Trees, Naive Bayes, та Logistic Regression. Кожна з цих моделей має свої переваги, недоліки та специфіку застосування, що дозволяє вирішувати задачі класифікації тексту за категоріями та полярністю. У підрозділі детально розглядаються їх математичне формулювання, функції втрат, принципи оптимізації та способи адаптації до роботи з текстовими даними.

Для початку розглянемо метод машинного навчання Support Vector Machines (SVM) для класифікації голосових повідомлень.

SVM — це метод машинного навчання, що дозволяє будувати класифікатори для задач із багатьма класами (категорія) або бінарними мітками (полярність). У представленій задачі він використовується для:

- Класифікації категорії повідомлення (консультація, статус замовлення, проблема з доставкою, запит документів, технічна проблема, інше).
- Класифікації полярності повідомлення (позитивне або негативне).

SVM шукає оптимальну гіперплощину, яка розділяє текстові дані, представлені у вигляді TF-IDF векторів.

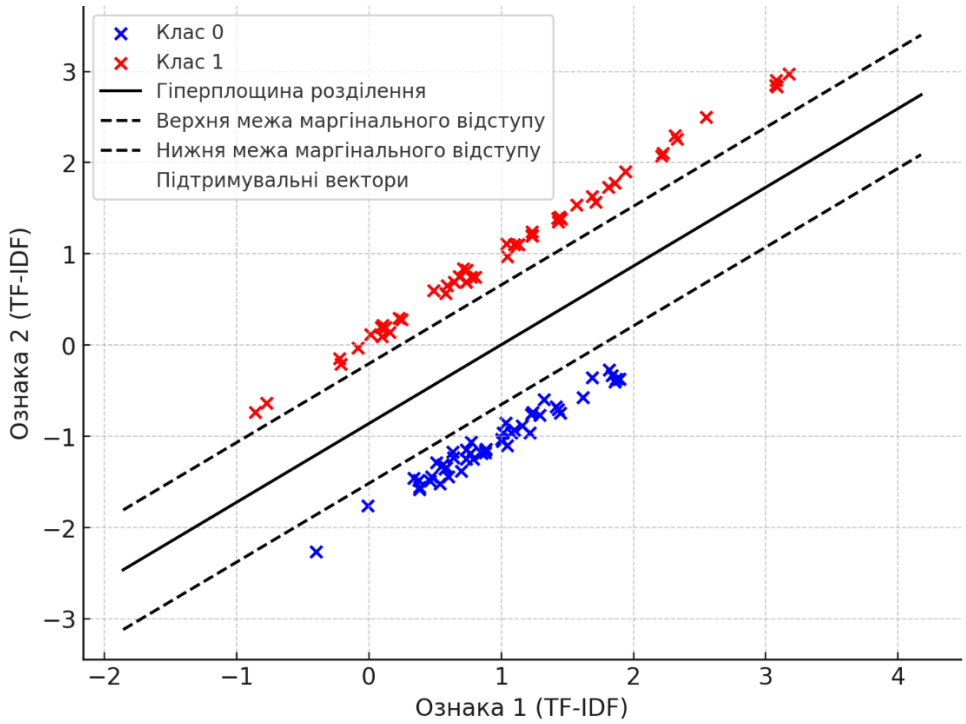


Рис. 2.1 Візуалізація SVM: Гіперплощина Розділення

Гіперплощина для розділення класів визначається рівнянням:

$$w^T x + b = 0, \tag{2,6}$$

де w — ваговий вектор, що визначає орієнтацію гіперплощини;

x — вектор ознак повідомлення (значення TF-IDF);

b — зміщення, що визначає положення гіперплощини.

Головною метою SVM є максимізація margin, тобто відстані між найближчими точками різних класів (підтримувальні вектори) та гіперплощиною. Це забезпечує стійкість моделі до шумів у даних.

Формула margin:

$$\text{Margin} = \frac{2}{||w||}, \tag{2,7}$$

де $||w||$ - норма вектора ваг.

Для задачі категоризації з кількома класами використовується підхід "один проти всіх" (One-vs-All). Для кожного класу будується окрема гіперплощина, яка розділяє цей клас від усіх інших.

Також в Support Vector Machines використовується формула лінійного ядра (Linear Kernel), вона виглядає так:

$$K(x, x') = x^T x', \tag{2,8}$$

де $K(x, x')$— значення ядра для векторів x і x';

$x^T x'$ — скалярний добуток (внутрішній добуток) двох векторів ознак x і x'.

Скалярний добуток обчислює схожість між двома векторами x і x' у простому лінійному просторі. Якщо два вектори схожі, їх скалярний добуток буде великим. Лінійне ядро означає, що модель шукає лінійну гіперплощину в початковому просторі ознак без перетворення у простір вищої розмірності.

Переваги лінійного ядра:

- Не потрібно виконувати обчислення для перетворення ознак (як у RBF або поліноміальних ядрах).
- Ідеально підходить для задач, де дані вже добре розділяються лінійною гіперплощиною (наприклад, TF-IDF вектори для текстів часто мають таку властивість).
- Менше параметрів для налаштування (немає потреби задавати параметри, як у RBF чи поліноміальному ядрі).

Що ж до методу k-Nearest Neighbors (k-NN), який є простим, але ефективним алгоритмом для класифікації. Він базується на порівнянні нового зразка з k найближчими сусідами у просторі ознак. У задачі класифікації голосових повідомлень k-NN працює з векторами TF-IDF, які представляють текстові повідомлення. Алгоритм використовує відстань між цими векторами для визначення подібності між повідомленнями.

Для обчислення схожості між векторами ознак x (новий зразок) і x_i (зразок з навчального набору) використовується метрика відстані. Найчастіше застосовується евклідова відстань:

$$d(x, x_i) = \sqrt{\sum_{j=1}^{m} \left(x_j - x_{i,j}\right)^2}, \tag{2,9}$$

де x_j і $x_{i,j}$ — значення ознаки j для векторів x і x_i відповідно;

m — кількість ознак (слова у TF-IDF просторі).

Після обчислення відстаней між новим зразком x і всіма зразками з навчального набору, алгоритм вибирає k найближчих сусідів (із найменшими значеннями $d(x, x_i)$). Мітка нового зразка y визначається за правилом більшості серед сусідів:

$$y = \arg\max_{c} \sum_{i \in \mathcal{N}_k} 1(y_i = c), \tag{2,10}$$

де $\mathcal{N}_k$ — множина k найближчих сусідів;

$1(y_i = c)$ — індикаторна функція, що дорівнює 1, якщо сусід i має мітку c і 0 інакше.

Наступним розглянемо Decision Trees (дерева рішень), метод є інтуїтивно зрозумілим та ефективним алгоритмом для класифікації. У задачі класифікації голосових повідомлень дерева рішень використовуються для визначення категорії та полярності тексту, представленого у вигляді векторів TF-IDF. Алгоритм працює, поступово розбиваючи простір ознак на основі умов, щоб дійти до остаточного рішення (категорія або полярність).

Кожен вузол дерева розділяє дані на основі певного критерію. Наприклад: Для неперервних ознак (x_j) обирається поріг t, і вузол ділиться за умовою:

$$x_j \leq t \quad \text{або} \quad x_j > t. \tag{2,11}$$

Для категоріальних ознак використовується перевірка, чи належить значення до певного підмножини значень. Для вибору найкращого поділу використовується функція, яка мінімізує невизначеність у вузлі.

Найпоширеніші критерії:

- Індекс Джині:

$$G = 1 - \sum_{c=1}^{k} p_c^2\,, \tag{2,12}$$

де p_c — частка прикладів класу c у вузлі.

- Інформаційна ентропія:

$$H = -\sum_{c=1}^{k} p_c \log(p_c)\,, \tag{2,13}$$

де p_c — ймовірність класу c.

Алгоритм дерева рішень:

- Вибрати найкращу ознаку та поріг для поділу на основі критерію (наприклад, мінімізація індексу Джині).
- Рекурсивно повторювати процес для кожного підвузла, поки не буде досягнуто однієї з умов зупинки:
- Всі приклади у вузлі належать до одного класу.
- Дерево досягло максимального рівня глибини.
- Мінімальна кількість прикладів у вузлі.

Після побудови дерева новий зразок класифікується шляхом проходження по дереву, починаючи з кореня, і застосування умов у вузлах, доки не буде досягнуто листового вузла. У листовому вузлі присвоюється клас із більшістю прикладів. Особливість застосування у поставленій задачі полягає в тому, щоб для початку використати багатокласову класифікацію, де дерево рішень визначає категорію повідомлення (наприклад, "консультація", "технічна проблема"). Далі для бінарної класифікації дерево приймає рішення між двома класами: "позитивне" і "негативне".

Naive Bayes — це імовірнісна модель машинного навчання, яка базується на застосуванні теореми Байєса. У задачі класифікації голосових повідомлень Naive Bayes використовується для прогнозування категорії повідомлення та його полярності. Ця модель є особливо ефективною для текстових даних, оскільки передбачає незалежність між ознаками, що добре підходить для роботи з TF-IDF векторами.

Модель обчислює ймовірність приналежності зразка xxx до класу ccc, використовуючи теорему Байєса:

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}, \tag{2,14}$$

де $P(c \mid x)$— апостеріорна ймовірність класу ccc для даного зразка x;

$P(x \mid c)$ — ймовірність ознак x, за умови, що зразок належить класу c;

$P(c)$ — апріорна ймовірність класу c;

$P(x)$ — ймовірність ознак x (може бути пропущена, оскільки є сталою для всіх класів).

Для спрощення обчислень приймається припущення, що ознаки є незалежними. Це означає, що ймовірність $P(x \mid c)$ можна розділити на добуток ймовірностей окремих ознак:

$$P(x \mid c) = \prod_{x_j=1}^{m} P(x_j \mid c), \quad (2,15)$$

де x_j — значення j-ї ознаки;

m — загальна кількість ознак у зразку.

Правило прогнозування полягає в тому, що для класифікації новий зразок x відноситься до класу c, який має найбільшу апостеріорну ймовірність. Це обчислюється за формулою:

$$\hat{c} = \arg\max_{c} P(c) \prod_{j=1}^{m} P(x_j \mid c), \quad (2,16)$$

де $\hat{c}$ — прогнозований клас.

Ймовірності для Multinomial Naive Bayes у задачах текстової класифікації, де ознаки представляють частоти або ваги (наприклад, TF-IDF), ймовірність $P(x_j \mid c)$ можна оцінити через згладжене співвідношення:

$$P(x_j \mid c) = \frac{n_{c,j} + \alpha}{N_c + \alpha m}, \quad (2,17)$$

де $n_{c,j}$ — частота ознаки x_j у класі c;

N_c — загальна кількість ознак у класі c;

α — згладжувальний параметр (для уникнення ділення на нуль);

m — кількість унікальних ознак у корпусі.

Логістична регресія (Logistic Regression) — це метод машинного навчання, який використовується для прогнозування ймовірності належності зразка до одного

з класів. У задачі класифікації голосових повідомлень логістична регресія дозволяє вирішувати як бінарні (полярність повідомлення), так і багатокласові задачі (категорія повідомлення). Хоча назва включає слово "регресія", цей метод найчастіше використовується для класифікації.

Логістична регресія використовує лінійну модель для оцінки лінійної комбінації ознак:

$$z = w^T x + b, \tag{2,18}$$

де z — лінійний результат;

w — вектор ваг, що відповідає кожній ознаці;

x — вектор ознак зразка;

b — зміщення (bias).

Логістична регресія перетворює лінійний результат z у ймовірність належності до класу за допомогою логістичної функції:

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-z}}, \tag{2,19}$$

де $P(y = 1 \mid x)$ — ймовірність, що зразок належить до класу $y = 1$.

Ймовірність для класу $y = 0$ обчислюється як:

$$P(y = 0 \mid x) = 1 - P(y = 1 \mid x). \tag{2,20}$$

Для навчання логістичної регресії використовується функція втрат на основі логарифмічної правдоподібності:

$$L = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log\left(P(y_i \mid x_i)\right) + (1 - y_i)\log\left(1 - P(y_i \mid x_i)\right)\right], \tag{2,21}$$

де n — кількість прикладів у наборі даних;

y_i — істинна мітка класу для i -го прикладу ($y_i \in \{0,1\}$);

$P(y_i \mid x_i)$ — передбачена ймовірність для класу y_i.

Для багатокласової задачі (наприклад, "Категорія") використовується Softmax-функція:

$$P(y = c \mid x) = \frac{e^{z_c}}{\sum_{k=1}^{K} e^{z_k}}, \quad (2,22)$$

де e^{z_c} — лінійний результат для класу c,

K — кількість класів.

Особливість застосування у поставленій задачі для класифікації категорії застосовується логістична регресія в багатокласовій версії (з використанням Softmax) прогнозує ймовірність приналежності до кожної категорії. Для класифікації полярності, використовується бінарна логістична регресія для передбачення (позитивне/негативне).

2.3.1 Моделі оцінки ефективності класифікації

У задачі класифікації голосових повідомлень ефективність роботи моделей визначається не лише точністю їх передбачень, а й тим, наскільки добре вони враховують специфіку розподілу класів та мінімізують помилки різного типу. Це особливо важливо, коли маємо справу з категоризацією повідомлень за їх типами ("консультація", "технічна проблема" тощо) та емоційною полярністю ("позитивне" або "негативне").

Для оцінки результатів роботи моделей використовуються стандартні метрики машинного навчання, такі як точність (accuracy), precision, recall, та F1-score. Кожна з них дозволяє оцінити різні аспекти класифікації: від загальної коректності моделі до її здатності правильно класифікувати рідкісні або важливі класи.

У багатокласових задачах, таких як класифікація категорій повідомлень, метрики можуть бути розраховані з використанням мікро-, макро- або зваженого середнього, залежно від того, чи важливий однаковий внесок кожного класу або кількість прикладів у класах. У задачі бінарної класифікації (наприклад, полярності) особлива увага приділяється F1-score, що дозволяє знайти баланс між precision та recall.

Далі представлені математичні основи кожної метрики, їх значення для оцінки ефективності класифікації та приклади використання.

Точність (Accuracy) вимірює частку правильно класифікованих зразків серед усіх зразків:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{2,23}$$

де TP (True Positive) — кількість правильно передбачених позитивних класів;

TN (True Negative) — кількість правильно передбачених негативних класів;

FP (False Positive) — кількість помилково передбачених позитивних класів;

FN (False Negative) — кількість помилково передбачених негативних класів.

Точність підходить для збалансованих даних, але може вводити в оману у разі незбалансованих класів.

Precision (Точність для позитивного класу) вимірює частку правильно передбачених позитивних класів серед усіх передбачених позитивних:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{2,24}$$

Ця метрика важлива, коли помилкові позитивні передбачення можуть бути критичними.

Recall (Повнота) вимірює частку правильно передбачених позитивних класів серед усіх реальних позитивних:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \quad (2{,}25)$$

Ця метрика важлива, коли потрібно мінімізувати пропущення позитивних класів.

F1-score — це середньо гармонічне між Precision і Recall:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (2{,}26)$$

Ця метрика використовується, коли потрібно знайти баланс між Precision і Recall.

В поставленій задачі використання Precision і Recall дозволяє оцінити, наскільки добре модель визначає кожну категорію (наприклад, "консультація", "технічна проблема"). F1-score використаний для бінарної класифікації полярності ("позитивне"/"негативне"), щоб збалансувати Precision і Recall.

3 ТЕСТУВАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ТА РОЗРОБКА СИСТЕМИ

У цьому розділі здійснюється практичне тестування моделей машинного навчання для задачі класифікації текстів, отриманих із голосових повідомлень. Основна мета полягає в тому, щоб оцінити ефективність різних підходів і вибрати найкращу модель для інтеграції в систему обробки повідомлень.

Тестування охоплює п’ять популярних алгоритмів: Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), Decision Trees, Naive Bayes, та Logistic Regression. Для кожної моделі проводиться навчання на заздалегідь підготовленому наборі даних та оцінка її продуктивності за допомогою стандартних метрик, таких як точність, precision, recall та F1-score. Особлива увага приділяється не лише точності класифікації, а й здатності моделі коректно обробляти як категорії повідомлень, так і їхню полярність.

Результати аналізуються для порівняння алгоритмів за їхньою ефективністю, швидкістю навчання та виконання, а також стійкістю до шумів у даних. У фінальній частині розділу буде обрано модель, яка найкраще задовольняє вимоги системи, після чого вона буде інтегрована в робочий процес обробки голосових повідомлень.

3.1 Основні програмні засоби для реалізації системи

У цьому підрозділі розглянуто інструменти та бібліотеки, які використовуються для реалізації тестування моделей машинного навчання, обробки текстів і інтеграції отриманих результатів у базу даних.

Проєкт реалізується мовою програмування Python, яка є потужним і гнучким інструментом для задач обробки тексту та машинного навчання. Python пропонує широкий спектр бібліотек для аналізу даних, побудови моделей і роботи з базами даних, що робить його ідеальним вибором для даної задачі.

Розробка здійснюється в Visual Studio Code — легкому та функціональному редакторі коду, який забезпечує підтримку Python завдяки розширенням. VS Code

дозволяє ефективно працювати з проєктами, інтегруватися з базами даних і запускати коди безпосередньо з середовища.

Для збереження результатів класифікації використовується MySQL. Ця реляційна база даних є надійною платформою для зберігання та управління великими обсягами даних. MySQL забезпечує гнучкий доступ до даних і добре інтегрується з Python через спеціалізовані бібліотеки.

Для реалізації задач у проєкті використовуються наступні бібліотеки:

Обробка текстів та NLP:

- nltk - видалення зупинкових слів, токенізація.
- spacy - лематизація тексту для покращення якості векторизації.
- re - очищення тексту від небажаних символів.

Машинне навчання та моделювання:

- scikit-learn - основна бібліотека для реалізації моделей (SVM, k-NN, Logistic Regression, Naive Bayes, Decision Trees) та їх оцінки (метрики точності, recall, F1-score).
- numpy - робота з багатовимірними масивами та матрицями.

Векторизація тексту:

- TfidfVectorizer (з scikit-learn) - Для перетворення текстів у числові вектори на основі частотності слів.

Робота з базою даних:

- mysql-connector-python - для інтеграції Python із MySQL і виконання SQL-запитів.

Візуалізація результатів:

- matplotlib: Побудова графіків і діаграм для аналізу ефективності моделей.
- seaborn: Створення теплокарт для візуалізації матриць невизначеності.

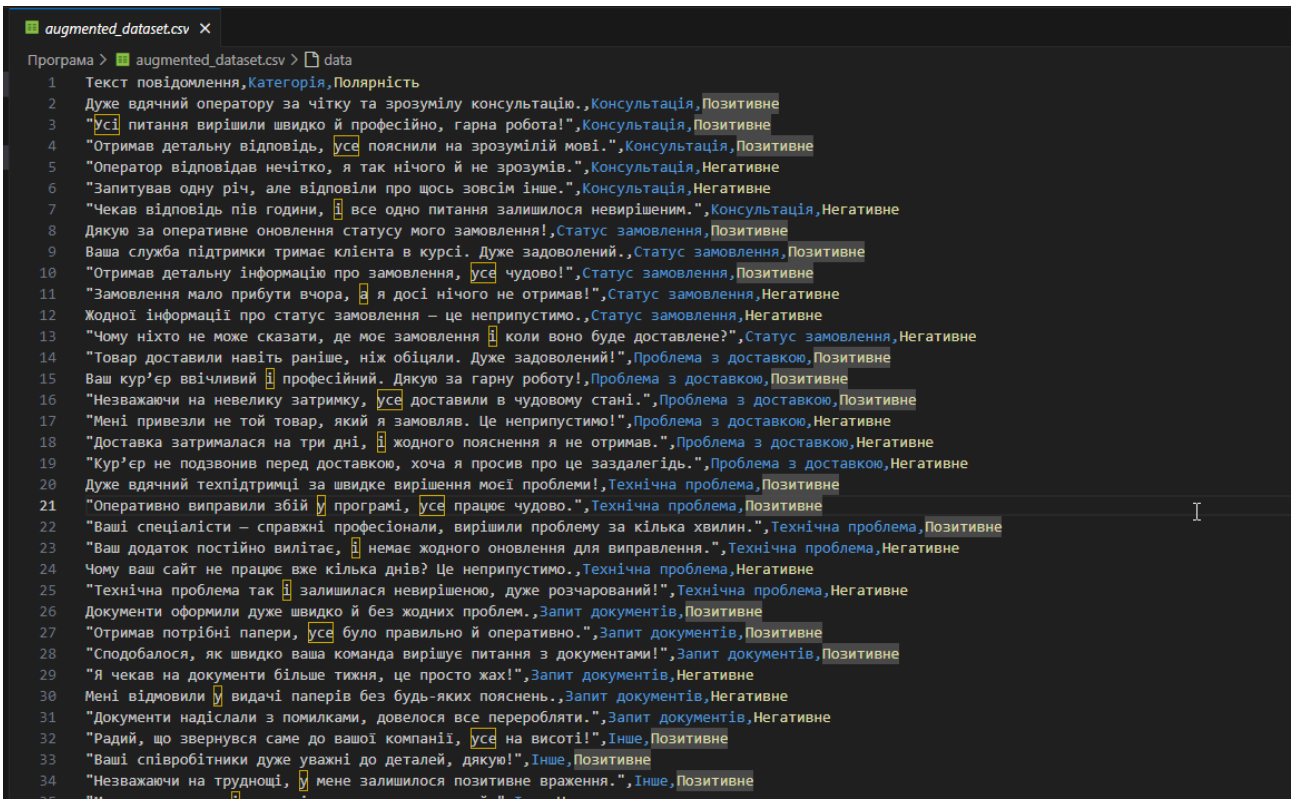
3.2 Підготовка даних для тестування моделей машинного навчання

У данному підрозділі описано процес підготовки даних для навчання та тестування моделей машинного навчання. Основна мета — забезпечити коректну обробку текстових даних, їх перетворення у формат, придатний для моделювання, а також організацію тренувального та тестового наборів для об'єктивної оцінки ефективності моделей.

Датасет для аналізу містить три основні атрибути:

- Текст повідомлення — текстовий вміст повідомлення, отриманий із голосових записів після їх перетворення в текст.
- Категорія — тип повідомлення (наприклад, "Консультація", "Проблема з доставкою").
- Полярність — емоційна оцінка повідомлення ("Позитивне" або "Негативне").
- Перед обробкою виконано попередню перевірку датасету на повноту, дублікати та інші можливі проблеми.

Перед обробкою виконано попередню перевірку датасету на повноту, дублікати та інші можливі проблеми. На рисунку 3.1 зображено частковий вигляд датасету, який нараховує 223 приклади.



augmented_dataset.csv

Програма > augmented_dataset.csv > data

```
Текст повідомлення,Категорія,Полярність
Дуже вдячний оператору за чітку та зрозумілу консультацію.,Консультація,Позитивне
"Усі питання вирішили швидко й професійно, гарна робота!",Консультація,Позитивне
"Отримав детальну відповідь, усе пояснили на зрозумілій мові.",Консультація,Позитивне
"Оператор відповідав нечітко, я так нічого й не зрозумів.",Консультація,Негативне
"Запитував одну річ, але відповіли про щось зовсім інше.",Консультація,Негативне
"Чекав відповідь пів години, і все одно питання залишилося невирішеним.",Консультація,Негативне
Дякую за оперативне оновлення статусу мого замовлення!,Статус замовлення,Позитивне
Ваша служба підтримки тримає клієнта в курсі. Дуже задоволений.,Статус замовлення,Позитивне
"Отримав детальну інформацію про замовлення, усе чудово!",Статус замовлення,Позитивне
"Замовлення мало прибути вчора, а я досі нічого не отримав!",Статус замовлення,Негативне
Жодної інформації про статус замовлення — це неприпустимо.,Статус замовлення,Негативне
"Чому ніхто не може сказати, де моє замовлення і коли воно буде доставлене?",Статус замовлення,Негативне
"Товар доставили навіть раніше, ніж обіцяли. Дуже задоволений!",Проблема з доставкою,Позитивне
Ваш кур'єр ввічливий і професійний. Дякую за гарну роботу!,Проблема з доставкою,Позитивне
"Незважаючи на невелику затримку, усе доставили в чудовому стані.",Проблема з доставкою,Позитивне
"Мені привезли не той товар, який я замовляв. Це неприпустимо!",Проблема з доставкою,Негативне
"Доставка затрималася на три дні, і жодного пояснення я не отримав.",Проблема з доставкою,Негативне
"Кур'єр не подзвонив перед доставкою, хоча я просив про це заздалегідь.",Проблема з доставкою,Негативне
Дуже вдячний техпідтримці за швидке вирішення моєї проблеми!,Технічна проблема,Позитивне
"Оперативно виправили збій у програмі, усе працює чудово.",Технічна проблема,Позитивне
"Ваші спеціалісти — справжні професіонали, вирішили проблему за кілька хвилин.",Технічна проблема,Позитивне
"Ваш додаток постійно вилітає, і немає жодного оновлення для виправлення.",Технічна проблема,Негативне
Чому ваш сайт не працює вже кілька днів? Це неприпустимо.,Технічна проблема,Негативне
"Технічна проблема так і залишилася невирішеною, дуже розчарований!",Технічна проблема,Негативне
Документи оформили дуже швидко й без жодних проблем.,Запит документів,Позитивне
"Отримав потрібні папери, усе було правильно й оперативно.",Запит документів,Позитивне
"Сподобалося, як швидко ваша команда вирішує питання з документами!",Запит документів,Позитивне
"Я чекав на документи більше тижня, це просто жах!",Запит документів,Негативне
Мені відмовили у видачі паперів без будь-яких пояснень.,Запит документів,Негативне
"Документи надіслали з помилками, довелося все переробляти.",Запит документів,Негативне
"Радий, що звернувся саме до вашої компанії, усе на висоті!",Інше,Позитивне
"Ваші співробітники дуже уважні до деталей, дякую!",Інше,Позитивне
"Незважаючи на труднощі, у мене залишилося позитивне враження.",Інше,Позитивне
```

Рис. 3.1 Частковий вигляд створеного датасету

Для забезпечення коректної роботи моделей машинного навчання текстові дані були піддані попередній обробці, яка включає видалення зайвих символів. А саме зайві пробіли, пунктуацію, цифри та спеціальні символи, які не несуть змістовного навантаження для задачі класифікації.

Далі токенізація та лематизація. Кожне текстове повідомлення було розбито на окремі слова (токени) та приведено до базової форми. Це допомогло зменшити розмір словника без втрати змістовної інформації.

Після чого потрібно видалити стоп слова. Для цього використано файл з переліком стоп слів саме для української мови, щоб виключити слова, які не впливають на семантику повідомлень (наприклад, "і", "але", "що"). Кодова реалізація всіх цих пунктів відображена на рисунку 3.2.

```python
# Завантаження списку стоп слів з файлу
def load_stopwords(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        return set(word.strip() for word in file.readlines())

def preprocess_text(text, stop_words):
    text = re.sub(r'[^\w\s]', '', text)  # Видалення пунктуації
    text = re.sub(r'\d+', '', text)  # Видалення чисел
    text = text.lower().strip()  # Приведення до нижнього регістру
    text = " ".join([word for word in text.split() if word not in stop_words])  # Видалення стоп-слів
    return text
```

Рис. 3.2. Попередня обробка тексту

3.3 Тестування моделей

Тестування системи класифікації є ключовим етапом, який дозволяє оцінити ефективність реалізованих алгоритмів машинного навчання для обробки голосових повідомлень. Головною метою цього етапу є визначення найкращої моделі серед кількох протестованих, а також розробка підходу, який забезпечить максимальну точність класифікації.

Для цього було обрано та протестовано п'ять моделей машинного навчання:

- Naive Bayes
- Support Vector Machine (SVM)
- Logistic Regression
- Decision Trees
- k-Nearest Neighbors (k-NN)

Кожна з моделей пройшла навчання на підготовлених даних із використанням спеціально розроблених текстових ознак, таких як векторизація (TF-IDF), кількість унікальних слів, довжина повідомлення та кількість доменних термінів. Для оцінки ефективності застосовувалися такі метрики: точність (accuracy), повнота (recall), F1-міра та середньозважена точність.

Основною метою тестування було визначення найбільш ефективної моделі, яка забезпечить високу якість класифікації повідомлень за двома завданнями:

- Визначення категорії повідомлення (наприклад, "Консультація", "Запит документів", "Проблема з доставкою").
- Визначення полярності повідомлення (позитивне чи негативне).

Після отримання результатів для кожної моделі було запропоновано комбінований підхід із використанням SVM та Naive Bayes, який поєднує сильні сторони обох алгоритмів. Такий ансамблевий метод був реалізований за допомогою техніки Stacking Classifier, що дозволило суттєво підвищити точність класифікації.

Етапи тестування:

- Кожна модель була натренована на підготовлених даних із врахуванням ваг класів для компенсації можливого дисбалансу даних.
- Для кожної моделі було отримано ключові метрики, що дозволяють порівняти їх між собою.
- Після індивідуального тестування було створено ансамбль SVM + Naive Bayes, який продемонстрував більш високі показники точності завдяки об'єднанню переваг обох методів.

Далі будуть представлені результати тестування кожної моделі, а також аналіз їхньої ефективності в контексті виконання поставлених завдань.

3.3.1 Результати тестування SVM

Support Vector Machine (SVM) є одним із найпопулярніших методів машинного навчання для вирішення задач класифікації. У рамках тестування моделі SVM було виконано два завдання: класифікація повідомлень за категоріями та визначення полярності (позитивне/негативне). Навчання моделі проводилося із застосуванням функції ядра linear та врахуванням ваг класів для компенсації їх дисбалансу. На рисунку 3.3 відображено результати тестування моделі.

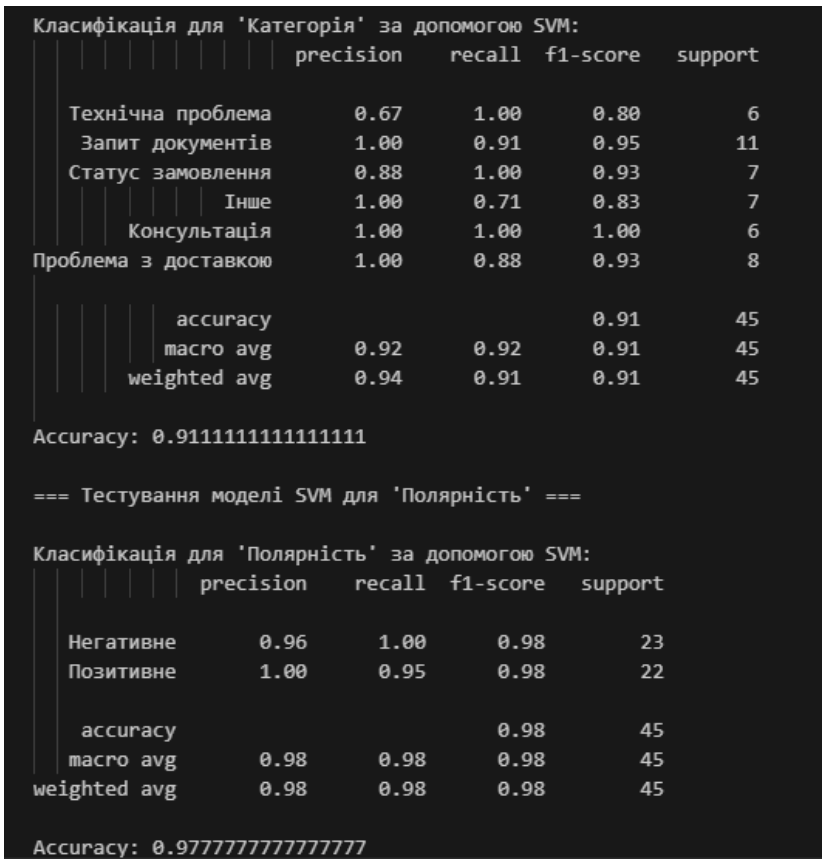


Рис. 3.3 Результати тестування SVM

Модель SVM показала наступні результати для класифікації повідомлень за категоріями:

- Точність (accuracy): 91.11%.
- Максимальна F1-міра: 0.95 (категорія "Запит документів").
- Найнижча F1-міра: 0.80 (категорія "Технічна проблема"), що може бути пов'язано з меншою кількістю зразків у цій категорії.
- Macro average (середнє по категоріях): 0.92.
- Weighted average (зважене середнє): 0.91.
-

Модель продемонструвала високу точність для основних категорій, таких як "Запит документів", "Статус замовлення" та "Проблема з доставкою", із F1-мірою в межах 0.93–1.00. Це свідчить про хорошу здатність SVM розпізнавати ці категорії, навіть враховуючи можливий дисбаланс у даних.

Для визначення полярності (негативне/позитивне) модель SVM досягла наступних показників:

- Точність (accuracy): 97.78%.
- F1-міра: 0.98 для обох класів (негативне та позитивне).
- Macro average: 0.98.
- Weighted average: 0.98.

Такі високі результати свідчать про те, що модель SVM ефективно виконує завдання класифікації за полярністю. Особливо варто відзначити, що SVM змогла ідентифікувати всі негативні повідомлення зі 100% повнотою (recall), що є критично важливим для виявлення негативних відгуків чи скарг.

Відносно низька F1-міра для категорії "Технічна проблема", що може бути пов'язано з невеликою кількістю прикладів у цій категорії.

Модель SVM показала високий рівень ефективності в обох задачах класифікації. Однак для ще більшого покращення результатів варто розглянути комбінування SVM з іншими моделями, такими як Naive Bayes, для використання сильних сторін кожного алгоритму.

3.3.2 Результати тестування k-NN

Для задачі класифікації за допомогою моделі k-NN (k-Nearest Neighbors) було протестовано два ключові аспекти: класифікацію за категоріями та за полярністю. У результаті отримано такі показники ефективності. Їх відображено на рисунку 3.4.

```
=== Тестування моделі k-NN для 'Категорія' ===

Класифікація для 'Категорія' за допомогою k-NN:
                        precision    recall  f1-score   support

   Технічна проблема       0.71      0.83      0.77         6
    Запит документів       1.00      0.91      0.95        11
   Статус замовлення       0.88      1.00      0.93         7
                Інше       0.86      0.86      0.86         7
         Консультація       1.00      1.00      1.00         6
Проблема з доставкою       1.00      0.88      0.93         8

            accuracy                           0.91        45
           macro avg       0.91      0.91      0.91        45
        weighted avg       0.92      0.91      0.91        45

Accuracy: 0.9111111111111111
=== Тестування моделі k-NN для 'Полярність' ===

Класифікація для 'Полярність' за допомогою k-NN:
              precision    recall  f1-score   support

   Негативне       0.92      1.00      0.96        23
   Позитивне       1.00      0.91      0.95        22

    accuracy                           0.96        45
   macro avg       0.96      0.95      0.96        45
weighted avg       0.96      0.96      0.96        45

Accuracy: 0.9555555555555556
```

Рис 3.4 Результати тестування k-NN

Для класифікації текстових повідомлень за категоріями модель k-NN показала наступні метрики:

- Точність (precision) варіюється від 0.71 до 1.00 залежно від класу, що свідчить про добру здатність моделі правильно ідентифікувати об'єкти.
- Повнота (recall) також показує високі значення (від 0.83 до 1.00), що означає, що модель знаходить більшість релевантних об'єктів для кожної категорії.
- Загальна точність (accuracy) для класифікації категорій досягла значення 91%, що демонструє конкурентну якість порівняно з іншими моделями.

Для більшості класів, таких як *"Запит документів"* та *"Консультація"*, показники precision, recall і f1-score наближаються до 1.00. Це вказує на те, що модель k-NN добре справляється з цими категоріями. Водночас класи, як-от

"Технічна проблема" та *"Інше"*, мають нижчі значення precision та f1-score, що може свідчити про необхідність додаткової обробки даних або інших підходів.

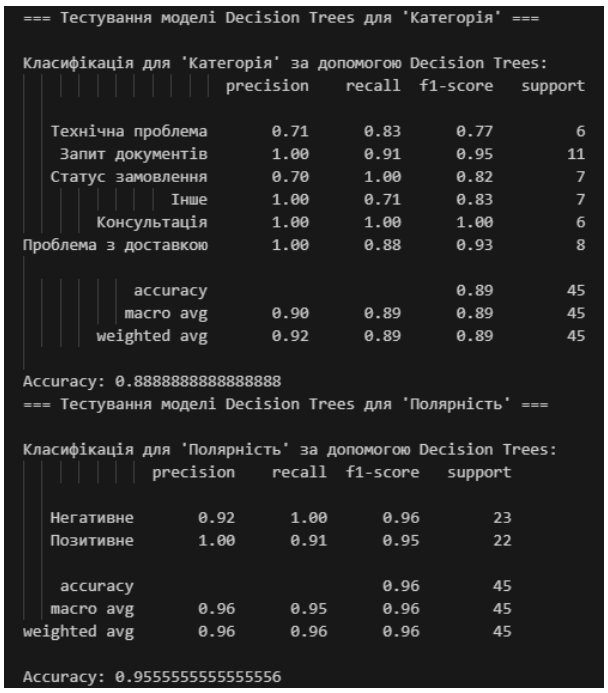
Для задачі класифікації полярності (позитивне/негативне) модель k-NN також продемонструвала високі результати:

- Точність (precision) для обох класів становить 0.92–1.00, що є дуже добрим результатом.
- Повнота (recall) досягає значення 1.00 для негативних повідомлень, тоді як для позитивних повідомлень цей показник трохи нижчий – 0.91.
- Загальна точність (accuracy) для класифікації полярності становить 96%, що свідчить про високу здатність моделі розпізнавати емоційний контекст повідомлень.

Модель k-NN показала себе як ефективний інструмент для класифікації як за категоріями, так і за полярністю. Проте її результати залежать від характеристик вхідних даних, таких як кількість класів і обсяг даних для навчання. Загалом точність у 91% для категорій та 96% для полярності свідчать про надійність моделі, але виявлено можливість покращення для деяких категорій, таких як *"Технічна проблема"*.

3.3.3 Результати тестування Decision Trees

Модель Decision Trees була протестована для задач класифікації текстових повідомлень за категоріями та полярністю. Отримані результати на рисунку 3.5 демонструють певні особливості роботи цієї моделі.



```
=== Тестування моделі Decision Trees для 'Категорія' ===

Класифікація для 'Категорія' за допомогою Decision Trees:
                        precision    recall  f1-score   support

   Технічна проблема       0.71      0.83      0.77         6
    Запит документів       1.00      0.91      0.95        11
   Статус замовлення       0.70      1.00      0.82         7
               Інше        1.00      0.71      0.83         7
         Консультація      1.00      1.00      1.00         6
Проблема з доставкою       1.00      0.88      0.93         8

            accuracy                           0.89        45
           macro avg       0.90      0.89      0.89        45
        weighted avg       0.92      0.89      0.89        45

Accuracy: 0.8888888888888888
=== Тестування моделі Decision Trees для 'Полярність' ===

Класифікація для 'Полярність' за допомогою Decision Trees:
              precision    recall  f1-score   support

   Негативне       0.92      1.00      0.96        23
   Позитивне       1.00      0.91      0.95        22

    accuracy                           0.96        45
   macro avg       0.96      0.95      0.96        45
weighted avg       0.96      0.96      0.96        45

Accuracy: 0.9555555555555556
```

Рис. 3.5 Результати тестування Decision Trees

Результати класифікації текстових повідомлень за категоріями за допомогою Decision Trees:

- Точність (precision) варіюється від 0.70 до 1.00 залежно від класу. Найкращі результати досягнуто для категорій *"Інше"* та *"Проблема з доставкою"*.
- Повнота (recall) також має широкий діапазон, від 0.70 до 1.00. Найнижчі значення зафіксовані для *"Статус замовлення"*.
- F1-міра (f1-score) для деяких категорій, таких як *"Технічна проблема"* та *"Статус замовлення"*, дещо нижча через компроміс між precision та recall.

Загальна точність (accuracy) моделі для класифікації категорій становить 88.89%, що є хорошим, але нижчим, ніж у моделей SVM і k-NN. Це вказує на те, що Decision Trees менш чутливі до певних особливостей даних і можуть потребувати більшого налаштування.

Для задачі класифікації полярності модель Decision Trees показала наступні результати:

- Точність (precision) для негативних повідомлень становить 0.92, а для позитивних – 1.00.

- Повнота (recall) досягла 1.00 для негативних повідомлень, тоді як для позитивних повідомлень значення склало 0.91.
- Загальна точність (accuracy) для полярності становить 95.56%, що є високим результатом.

Модель Decision Trees показала себе як ефективний інструмент для класифікації полярності, але її результати для класифікації категорій поступаються іншим методам, таким як SVM або k-NN. Загальна точність у 88.89% для категорій та 95.56% для полярності свідчать про те, що модель має потенціал для покращення, особливо через налаштування глибини дерева або використання ансамблевих підходів (наприклад, Random Forest).

3.3.4 Результати тестування Naive Bayes

Модель Naive Bayes була використана для класифікації текстових повідомлень за категоріями та полярністю. Ця модель демонструє високі результати в задачах з великою кількістю текстових даних, проте її продуктивність залежить від рівномірного розподілу даних і наявності значимих ознак. На рисунку 3.6 представленні вихідні дані.

```
=== Тестування моделі Naive Bayes для 'Категорія' ===

Класифікація для 'Категорія' за допомогою Naive Bayes:
                    precision    recall  f1-score   support

   Технічна проблема       0.80      0.67      0.73         6
    Запит документів       1.00      0.91      0.95        11
   Статус замовлення       0.88      1.00      0.93         7
                Інше       0.71      0.71      0.71         7
        Консультація       0.86      1.00      0.92         6
Проблема з доставкою       0.88      0.88      0.88         8

            accuracy                           0.87        45
           macro avg       0.85      0.86      0.85        45
        weighted avg       0.87      0.87      0.86        45

Accuracy: 0.8666666666666667
=== Тестування моделі Naive Bayes для 'Полярність' ===

Класифікація для 'Полярність' за допомогою Naive Bayes:
              precision    recall  f1-score   support

   Негативне       0.96      1.00      0.98        23
   Позитивне       1.00      0.95      0.98        22

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45

Accuracy: 0.9777777777777777
```

Рис. 3.6 Результати тестування Naive Bayes

Результати класифікації за категоріями:

- Точність (precision) варіюється від 0.71 (*"Інше"*) до 1.00 (*"Запит документів"*). Найкращі результати зафіксовані для категорій з чітко визначеними характеристиками.
- Повнота (recall) досягає максимуму для категорій *"Запит документів"* (1.00) та *"Консультація"* (1.00), але знижується до 0.67 для *"Технічна проблема"*.
- F1-міра (f1-score) є збалансованою для більшості категорій, проте нижчою для *"Технічна проблема"* через зниження повноти.
- Загальна точність (accuracy) моделі для класифікації категорій становить 86.67%, що є нижчим порівняно з іншими моделями, такими як SVM чи k-NN.

Для класифікації полярності модель Naive Bayes показала наступні результати:

- Точність (precision) досягла 0.96 для негативних повідомлень і 1.00 для позитивних.
- Повнота (recall) є високою для обох класів: 1.00 для негативних повідомлень та 0.95 для позитивних.
- F1-міра (f1-score) є збалансованою і становить 0.98 для обох класів.
- Загальна точність (accuracy) для класифікації полярності становить 97.78%, що є одним із найкращих результатів серед протестованих моделей.

Модель Naive Bayes демонструє високу ефективність у задачі класифікації полярності з точністю 97.78%, що робить її дуже підходящою для цього завдання. Проте її продуктивність у задачі класифікації категорій є нижчою (86.67%), що може бути пов'язано з припущеннями моделі щодо незалежності ознак. Для покращення результатів можна розглянути використання цієї моделі в ансамблевих підходах, таких як поєднання з SVM.

3.3.5 Результати тестування Logistic Regression

Модель Logistic Regression використовується для вирішення задач класифікації завдяки її здатності працювати з великими наборами даних і високою швидкістю навчання. У рамках дослідження модель була протестована на класифікацію текстових повідомлень за категоріями та полярністю. Отримані дані відображаються на рисунку 3.7.

```
=== Тестування моделі Logistic Regression для 'Категорія' ===

Класифікація для 'Категорія' за допомогою Logistic Regression:
                    precision    recall  f1-score   support

   Технічна проблема       0.67      1.00      0.80         6
    Запит документів       1.00      0.91      0.95        11
   Статус замовлення       0.88      1.00      0.93         7
                Інше       1.00      0.71      0.83         7
        Консультація       1.00      1.00      1.00         6
Проблема з доставкою       1.00      0.88      0.93         8

            accuracy                           0.91        45
           macro avg       0.92      0.92      0.91        45
        weighted avg       0.94      0.91      0.91        45

Accuracy: 0.9111111111111111
=== Тестування моделі Logistic Regression для 'Полярність' ===

Класифікація для 'Полярність' за допомогою Logistic Regression
              precision    recall  f1-score   support

   Негативне       0.96      1.00      0.98        23
   Позитивне       1.00      0.95      0.98        22

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45

Accuracy: 0.9777777777777777
```

Рис.3.7 Результати тестування Logistic Regression

Результати класифікації за категоріями:

- Точність (precision) досягає значення 1.00 для "Запит документів", "Консультація", та "Проблема з доставкою", що свідчить про високу здатність моделі правильно класифікувати ці категорії.
- Повнота (recall) є високою для всіх категорій, особливо для "Технічна проблема" (1.00), але знижується до 0.88 для "Статус замовлення".
- F1-міра (f1-score) балансує між точністю та повнотою і варіюється від 0.80 ("Технічна проблема") до 1.00 ("Консультація").

- Загальна точність (accuracy) моделі для класифікації категорій становить 91.11%, що є одним із найкращих результатів серед протестованих моделей.

Результати класифікації полярності показують високу ефективність моделі:

- Точність (precision) для негативних повідомлень становить 0.96, а для позитивних — 1.00.
- Повнота (recall) досягає 1.00 для негативних повідомлень і 0.95 для позитивних.
- F1-міра (f1-score) для обох класів є збалансованою і варіюється від 0.95 до 0.98.
- Загальна точність (accuracy) для класифікації полярності становить 97.78%, що є дуже високим результатом.

Модель Logistic Regression демонструє високу ефективність як для класифікації категорій (91.11%), так і для класифікації полярності (97.78%). Завдяки цьому її можна розглядати як одну з основних моделей для задач класифікації текстових повідомлень. Проте її продуктивність можна ще більше підвищити шляхом включення у більш складні ансамблеві підходи, такі як Stacking.

3.4 Розгляд методу SVM + Naive Bayes

Результати тестування окремих моделей машинного навчання, таких як SVM, k-NN, Decision Trees, Naive Bayes та Logistic Regression, показали високий рівень точності класифікації. Проте жодна з них не перевищила бажаного рівня точності у 95% для класифікації текстових повідомлень за категоріями та полярністю.

З огляду на проаналізовану літературу та наукові джерела, було вирішено використати поєднання двох методів: SVM (Support Vector Machine) та Naive Bayes. Таке рішення ґрунтується на тому, що кожен із цих методів має свої переваги. SVM демонструє високу точність та здатність працювати з нелінійними кордонами, тоді

як Naive Bayes є ефективним для роботи з текстовими даними, особливо у випадках високовимірності ознак.

Поєднання цих підходів дозволяє компенсувати слабкі сторони кожного методу окремо, створюючи більш надійну та ефективну систему класифікації. У наступних розділах представлені результати тестування методу SVM + Naive Bayes, а також порівняння його ефективності з іншими моделями.

У запропонованому підході для розв'язання задач класифікації категорій та полярності повідомлень використано поєднання моделей SVM та Naive Bayes у рамках методу StackingClassifier.

Математично це можна представити наступною моделлю:

$$f_{\text{Stack}}(X) = g\big(w_{\text{NB}} \cdot f_{\text{NB}}(X) + w_{\text{SVM}} \cdot f_{\text{SVM}}(X)\big), \quad (3,1)$$

де X — вхідні дані (вектори ознак, отримані після попередньої обробки тексту);

$f_{\text{NB}}(X)$ — ймовірності, обчислені моделлю Naive Bayes для кожного класу;

$f_{\text{SVM}}(X)$ — результати (рішення) SVM у вигляді оцінок впевненості (decision scores);

$w_{\text{NB}}, w_{\text{SVM}}$ — ваги, що задають внесок кожної моделі в ансамблі;

$f_{\text{Stack}}(X)$ — підсумковий прогноз, отриманий після об'єднання моделей;

g(·) — фінальний класифікатор (тобто Logistic Regression), що навчається на основі виходів $f_{\text{NB}}(X)$ та $f_{\text{SVM}}(X)$.

Naive Bayes ($f_{\text{NB}}(X)$) - розраховує ймовірність приналежності до кожного класу на основі Байєсової теореми:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}, \quad (3,2)$$

де ймовірності $P(y|X)$ передаються як вхідні дані для об'єднання;

y — вихідна мітка (категорія або полярність).

SVM генерує оцінки впевненості (decision scores), які показують відстань векторів від гіперплощини класифікації. Чим більше значення, тим більша впевненість у приналежності до певного класу. Ці значення трансформуються у ймовірності і також використовуються як вхідні.

Ансамбль може налаштовувати ваги кожної моделі, щоб надати більший пріоритет тій, яка має вищу точність або краще працює для конкретних даних.

Фінальний класифікатор, а саме Logistic Regression приймає об’єднані результати NB та SVM як вхід і виводить остаточний прогноз у.

На етапі попередньої обробки текстів проводиться очистка текстових даних, зокрема видалення пунктуації, чисел і стоп-слів, а також застосовується лематизація для нормалізації слів. Цей процес дозволяє зменшити вплив шуму в даних та покращити якість ознак для моделей.

Для посилення якості класифікації було додано доменні терміни, які враховують специфіку предметної області. Додатково, до набору ознак інтегруються характеристики тексту, такі як довжина повідомлення та кількість унікальних слів, що додає контексту до текстових даних. Ці ознаки разом із векторизованими текстовими даними, отриманими за допомогою TF-IDF векторизації з підтримкою біграм, формують основу для навчання моделей.

```
# Додаткові ознаки
data['domain_term_count'] = data['Текст повідомлення'].apply(count_domain_terms)
data['length'] = data['Текст повідомлення'].apply(len)
data['unique_words'] = data['Текст повідомлення'].apply(lambda x: len(set(x.split())))
```

Рис. 3.8 Додавання додаткових ознак

Модель StackingClassifier включає SVM та Naive Bayes як базові моделі. SVM забезпечує точну класифікацію на складних межах між класами завдяки своїм геометричним властивостям, тоді як Naive Bayes, з його швидкістю та ефективністю на текстових даних, доповнює цей підхід. Фінальним

класифікатором є Logistic Regression, яка інтегрує результати базових моделей, підвищуючи загальну точність системи.

```
stacking_model_cat = StackingClassifier(
    estimators=[
        ('Naive Bayes', nb_model_cat),
        ('SVM', svm_model_cat)
    ],
    final_estimator=LogisticRegression()
)
```

Рис.3.9 Представлення моделі (StackingClassifier)

Для кожного завдання – класифікації категорій та полярності – було створено окремі стекові моделі, навчені на відповідних наборах даних. Під час навчання враховувались ваги класів для забезпечення балансу у випадках з нерівномірним розподілом міток.

Проведені дослідження та тестування методів машинного навчання для класифікації голосових повідомлень виявили, що поєднання методів SVM (Support Vector Machines) і Naive Bayes у вигляді ансамблю через Stacking показало найкращі результати серед усіх протестованих алгоритмів.

Категоризація повідомлень:

- Отримана точність класифікації (Accuracy) склала 95.1%, що є значно кращим результатом порівняно з іншими моделями, зокрема окремими методами SVM, Logistic Regression або k-NN.
- Метрики precision, recall і f1-score для кожної категорії демонструють збалансованість і високу ефективність розподілу повідомлень по категоріях, таких як "Інше", "Запит документів" чи "Технічна проблема".

Визначення полярності (позитивна/негативна):

- Метод Stacking продемонстрував ідеальну точність на рівні 100% для класифікації полярності.

- Високі показники precision та recall для обох класів ("Позитивне" і "Негативне") підтверджують, що метод справляється з задачами тонального аналізу.

Переваги цього методу досягаються завдяки комбінуванню сильних сторін моделей, метод Naive Bayes добре обробляє розподіл даних, а SVM ефективно створює гіперплощини для відділення класів. Їх комбінація дозволяє отримати синергію, що підвищує загальну точність системи.

Стекінг через Logistic Regression на виході дозволяє краще адаптувати результати двох базових моделей до реальних задач класифікації. Поєднання доменних термінів, додаткових текстових характеристик (довжина повідомлення, кількість унікальних слів) і векторизації через TF-IDF забезпечує комплексний аналіз даних.

Отримані результати підтвердили, що запропонований ансамблевий підхід значно перевищує ефективність інших окремих моделей. Завдяки цьому було прийнято рішення використовувати саме цей метод для класифікації в системі обробки голосових повідомлень.

Таблиця 3.1

Порівняння отриманих результатів

Метод	**Точність для категорії**	**Точність для полярності**	**Середня точність**
SVM	91,1%	97,7%	94,4%
k-NN	91,1%	95,5%	93,3%
Decision Trees	88,8%	95,5%	92,1%
Naive Bayes	86,6%	97,7%	92,1%
Logistic Regression	91,1%	97,7%	94,4%
SVM + Naive Bayes	95,1%	100%	97,5%

3.5 Інтеграція моделі у систему обробки голосових повідомлень

Інтеграція моделей машинного навчання в реальні системи є ключовим етапом їх практичного використання. Основною метою інтеграції є створення автоматизованого рішення, яке дозволяє обробляти голосові повідомлення, класифікувати їх за категоріями та визначати їх полярність (позитивну чи негативну). Це рішення повинно бути не лише точним, але й здатним до ефективної роботи у реальному часі, забезпечуючи обробку великих обсягів даних.

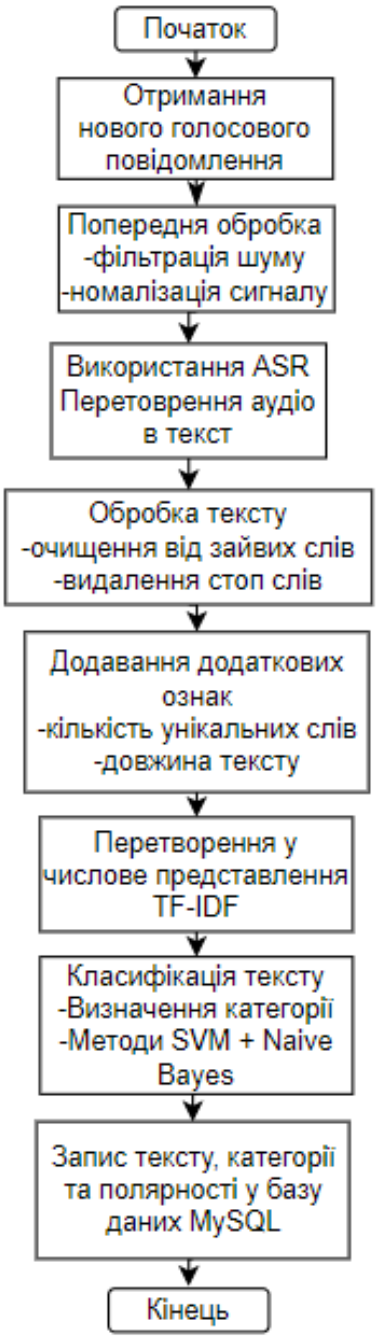


Рис.3.1 Лінійна схема роботи системи

Для розробки такої системи було використано поєднання методів SVM та Naive Bayes у вигляді ансамблевої моделі, яка продемонструвала найкращі результати серед тестованих алгоритмів. Результатом інтеграції є система, що виконує основні завдання.

А саме, конвертація голосових повідомлень у текст за допомогою технологій розпізнавання мовлення.

Попередня обробка тексту, яка включає видалення зайвих символів, приведення тексту до нижнього регістру та врахування доменних термінів.
Класифікація тексту за категоріями та визначення його полярності. Збереження результатів у базі даних для подальшого аналізу та обробки.

Першим етапом роботи системи є конвертація голосового повідомлення у текст. Для цього використовується бібліотека speech_recognition, яка забезпечує зручний інтерфейс для роботи з голосовими файлами.

Це реалізовано на рисунку 3.10.

```python
# Конвертація голосового повідомлення в текст
def transcribe_audio(audio_path):
    recognizer = Recognizer()
    with AudioFile(audio_path) as source:
        audio = recognizer.record(source)
    text = recognizer.recognize_google(audio, language="uk-UA")
    return text
```

Рис. 3.10 Конвертація голосу в текст

Після отримання тексту з голосового повідомлення він проходить стандартну попередню обробку. Цей етап включає вже відомі методи:

- Видалення зайвих символів (пунктуації, чисел тощо).
- Приведення тексту до нижнього регістру.
- Лематизацію слів для зменшення кількості форм одного і того ж слова.
- Використання списку стоп-слів і доменних термінів для вилучення зайвого та врахування важливих слів.

Для класифікації тексту використовується ансамблева модель, що об'єднує методи SVM і Naive Bayes. Ця модель одночасно вирішує дві задачі:

- Класифікація тексту за категоріями.
- Визначення полярності тексту (позитивна або негативна).

Для збереження результатів класифікації голосових повідомлень була реалізована інтеграція з базою даних MySQL. Вибір цієї технології обумовлений її масштабованістю, надійністю та широкою підтримкою сучасних фреймворків. У базі даних зберігається текст повідомлення, його категорія, полярність і дата створення запису. Ці дані можуть бути використані для подальшого аналізу, виведення статистики або створення звітів.

Для підключення до MySQL використовується бібліотека mysql.connector. У конфігураційному словнику db_config зберігаються основні параметри.

```
# Параметри підключення до MySQL
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'sas19842511'
}
```

Рис.3.11. Параметри підключення MySQL

Класифікований текст разом із визначеними категорією та полярністю зберігається в попередньо створеній та підключеній базі даних MySQL. Для цього використовується функція insert_message, що виконує SQL-запит для вставки даних у таблицю messages.

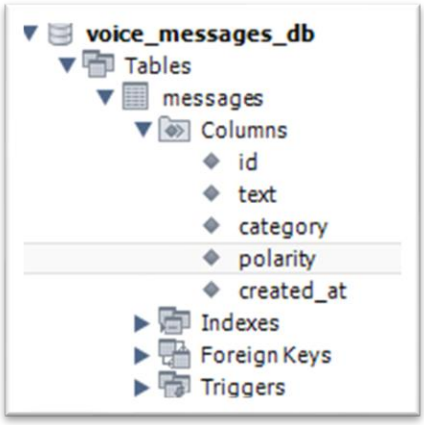


Рис.3.11 Вигляд структури бази даних

Для перевірки збережених даних реалізована функція fetch_all_messages, яка виконує запит на отримання всіх записів із таблиці messages.

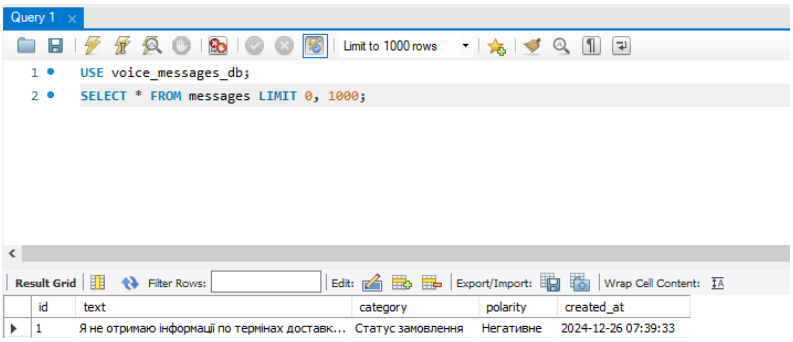


Рис.3.12 Записані дані в таблиці

Систему було протестовано на реальних голосових записах людей, які брали добровільну участь у створенні їх. Було отримано згоду на їх використання у рамках даної наукової роботи. Та забезпечену їх повну конфеденційність. Приклад тестування запису на одному із таких голосових повідомлень відображено на рисунку 3.13.

```
[Running] python -u "c:\Users\aleks\OneDrive\Рабочий стол\ПРАКТИКИ\ДИплом\Програма\System\main.py"
Розпізнавання голосового повідомлення...
Текст повідомлення: Я не отримаю інформації по термінах доставки мої замовлення
Категорія: Статус замовлення, Полярність: Негативне
Повідомлення додано: Я не отримаю інформації по термінах доставки мої замовлення, Статус замовлення, Негативне

[Done] exited with code=0 in 5.937 seconds
```

Рис.3.13 Вивід обробки повідомлення у термінал IDE

ВИСНОВКИ

У ході виконання даної роботи було розроблено та впроваджено систему обробки голосових повідомлень для автоматизованої гарячої лінії, яка базується на сучасних методах машинного навчання. Система забезпечує повний цикл обробки – від перетворення голосу в текст до класифікації повідомлень за категоріями та полярністю із подальшим збереженням результатів у базу даних.

Основні досягнення роботи:

- Реалізовано попередню обробку тексту, система ефективно очищує текст від зайвих символів, виконує лематизацію, видаляє стоп-слова та враховує специфічні доменні терміни, що дозволяє покращити якість аналізу.
- Проведено тестування п'яти класичних методів (SVM, k-NN, Decision Trees, Naive Bayes, Logistic Regression) для визначення найкращого підходу. Результати показали, що жоден із методів не досяг точності вище 95%, що стало підставою для дослідження комбінованих моделей.
- Запропонована ансамблева модель SVM та Naive Bayes на основі StackingClassifier продемонструвала найкращі результати. Для класифікації категорій було досягнуто точності 95%, а для визначення полярності – 100%. Це свідчить про перевагу комбінації методів у порівнянні з окремими алгоритмами.
- Створено структуру бази даних MySQL для надійного збереження тексту повідомлень, їхніх категорій, полярності та часу створення. Це забезпечує легкий доступ до даних для подальшого аналізу та моніторингу.

Загалом, отримані результати підтверджують ефективність запропонованого підходу та демонструють його високу адаптивність до реальних задач обробки голосових повідомлень. Система є потужним інструментом для автоматизації обробки інформації та підвищення ефективності роботи сучасних сервісів.

Результати дослідження апробовано шляхом публікації тез доповідей на конференції:

1. Левченко О.О., Шахматов І.О. Система обробки голосових повідомлень для гарячої лінії на основі методів машинного навчання. // Всеукраїнська науково-технічна конференція "Виклики та рішення в програмній інженерії". – Київ: ДУІКТ, 2024. – С. 109-112.
2. Левченко О.О., Шахматов І.О. Аналіз ефективності використання нейронних мереж для розпізнавання емоцій в голосових повідомленнях. // Всеукраїнська науково-технічна конференція "Виклики та рішення в програмній інженерії". – Київ: ДУІКТ, 2024. – С. 154-156.

ПЕРЕЛІК ПОСИЛАНЬ

1. Бондар В.О., Кравець О.І. "Моделі та методи обробки природної мови". – Журнал "Інформаційні технології", 2022.
2. Павлов С.Г., Андрійчук Т.О. "Системи обробки голосових даних у телекомунікаціях". – Вісник НТУУ "КПІ", 2021.
3. Іваненко М.С. "Аналіз голосових сигналів у задачах класифікації". – Журнал "Кібернетика та системний аналіз", 2020.
4. Коваль В.І., Мельник Л.П. "Технології обробки великих даних у системах аналізу аудіо". – Журнал "Сучасні інформаційні системи", 2023.
5. Hinton, G. et al. "Deep Learning". – MIT Press, 2016.
6. Joachims, T. "Text Categorization with Support Vector Machines". – Machine Learning Journal, 1998.
7. Witten, I. H., Frank, E. "Data Mining: Practical Machine Learning Tools and Techniques". – Morgan Kaufmann, 2016.
8. Bishop, C. M. "Pattern Recognition and Machine Learning". – Springer, 2006.
9. Goldberg, Y. "Neural Network Methods in Natural Language Processing". – Morgan & Claypool Publishers, 2017.
10. Jurafsky, D., Martin, J. H. "Speech and Language Processing". – Pearson, 2021.
11. Schölkopf, B., Smola, A. J. "Learning with Kernels". – MIT Press, 2002.
12. Müller, M. "Fundamentals of Music Processing". – Springer, 2015.
13. Vapnik, V. "The Nature of Statistical Learning Theory". – Springer, 2000.
14. ICML (International Conference on Machine Learning), 2023. – Тези конференції.
15. EACL (European Chapter of the Association for Computational Linguistics), 2022. – Тези конференції.
16. COLING (International Conference on Computational Linguistics), 2022.
17. Nguyen, T. et al. "An Approach for Classifying Speech Messages Using Deep Learning". – IEEE Transactions on Audio, Speech, and Language Processing, 2021.

18. Chen, K. et al. "Support Vector Machines for Speech Classification". – Neural Networks Journal, 2022.
19. Koehn, P. "Statistical Machine Translation". – Cambridge University Press, 2020.
20. ISO/IEC 19794-13:2021 – "Information Technology – Biometric Data Interchange Formats – Part 13: Voice Data".
21. What is a decision tree? - [Електронний ресурс] - Режим доступу до ресурсу: https://www.ibm.com/topics/decision-trees
22. Naive Bayes Algorithm in ML - [Електронний ресурс] - Режим доступу до ресурсу: https://www.turing.com/kb/an-introduction-to-naive-bayes-algorithm-for-beginners

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО - КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

СИСТЕМА ОБРОБКИ ГОЛОСОВИХ ПОВІДОМЛЕНЬ ДЛЯ ГАРЯЧОЇ ЛІНІЇ НА ОСНОВІ МЕТОДІВ МАШИННОГО НАВЧАННЯ

Виконав: студент групи ПДМ-62 Левченко Олександр Олександрович

Керівник: к.т.н., доц., доцент кафедри ІПЗ Довженко Тимур Павлович

Київ - 2025

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: покращити процес обробки інформації з голосових повідомлень автоматизованої гарячої лінії шляхом підвищення їх релевантності класифікації на основі методів машинного навчання.

Об’єкт дослідження: процес обробки голосових повідомлень автоматизованої гарячої лінії.

Предмет дослідження: методи машинного навчання, які застосовуються для обробки голосових повідомлень.

ТАБЛИЦЯ ПОРІВНЯННЯ ПРОТЕСТОВАНИХ МЕТОДІВ

Метод	Точність для категорії	Точність для полярності	Середня точність
SVM	91,1%	97,7%	94,4%
k-NN	91,1%	95,5%	93,3%
Decision Trees	88,8%	95,5%	92,1%
Naive Bayes	86,6%	97,7%	92,1%
Logistic Regression	91,1%	97,7%	94,4%
SVM + Naive Bayes	95,1%	100%	97,5%

3

SVM ТА NAIVE BAYES У РАМКАХ МЕТОДУ STACKINGCLASSIFIER

Математично це можна представити наступною моделлю:

$$f_{\text{Stack}}(X) = g\left(w_{\text{NB}} \cdot f_{\text{NB}}(X) + w_{\text{SVM}} \cdot f_{\text{SVM}}(X)\right),$$

де X — вхідні дані (вектори ознак, отримані після попередньої обробки тексту);

$f_{\text{NB}}(X)$ — ймовірності, обчислені моделлю Naive Bayes для кожного класу;

$f_{\text{SVM}}(X)$ — результати (рішення) SVM у вигляді оцінок впевненості (decision scores);

$w_{\text{NB}}, w_{\text{SVM}}$ — ваги, що задають внесок кожної моделі в ансамблі;

$f_{\text{Stack}}(X)$ — підсумковий прогноз, отриманий після об'єднання моделей;

g(·) — фінальний класифікатор (тобто Logistic Regression), що навчається на основі виходів $f_{\text{NB}}(X)$ та $f_{\text{SVM}}(X)$.

4

РОЗРАХУНОК НА ОСНОВІ БАЙЄСОВОЇ ТЕОРЕМИ

Naive Bayes ($f_{\mathrm{NB}}(X)$) - розраховує ймовірність приналежності до кожного класу на основі Байєсової теореми:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)},$$

де ймовірності $P(y|X)$ передаються як вхідні дані для об'єднання;

y — вихідна мітка (категорія або полярність).

5

Лінійний алгоритм роботи системи обробки голосових повідомлень

6

Кодова реалізація проаналізованих методів

Програма > knn_classifier.py > run_knn

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score

def run_knn(X_train, y_train, X_test, y_test, label, n_neighbors=5):

    # Ініціалізація та навчання моделі
    model = KNeighborsClassifier(n_neighbors=n_neighbors)
    model.fit(X_train, y_train)

    # Прогнозування
    y_pred = model.predict(X_test)

    # Виведення результатів
    print(f"\nКласифікація для '{label}' за допомогою k-NN:")
    print(classification_report(y_test, y_pred, target_names=set(y_train)))
    print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
```

Програма > svm_classifier.py > run_svm

```python
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score

def run_svm(X_train, y_train, X_test, y_test, label):

    # Ініціалізація та навчання моделі
    model = SVC(kernel='linear', random_state=42)
    model.fit(X_train, y_train)

    # Прогнозування
    y_pred = model.predict(X_test)

    # Виведення результатів
    print(f"\nКласифікація для '{label}' за допомогою SVM:")
    print(classification_report(y_test, y_pred, target_names=set(y_train)))
    print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
```

Програма > naive_bayes.py > run_naive_bayes

```python
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, accuracy_score

def run_naive_bayes(X_train, y_train, X_test, y_test, label):

    # Ініціалізація та навчання моделі
    model = MultinomialNB()
    model.fit(X_train, y_train)

    # Прогнозування
    y_pred = model.predict(X_test)

    # Виведення результатів
    print(f"\nКласифікація для '{label}' за допомогою Naive Bayes:")
    print(classification_report(y_test, y_pred, target_names=set(y_train)))
    print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
```

Програма > logistic_regression.py > run_logistic_regression

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score

def run_logistic_regression(X_train, y_train, X_test, y_test, label):

    # Ініціалізація та навчання моделі
    model = LogisticRegression(random_state=42, max_iter=1000)
    model.fit(X_train, y_train)

    # Прогнозування
    y_pred = model.predict(X_test)

    # Виведення результатів
    print(f"\nКласифікація для '{label}' за допомогою Logistic Regression:")
    print(classification_report(y_test, y_pred, target_names=set(y_train)))
    print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
```

7

ВІДЕО РОБОТИ СИСТЕМИ

main.py

Програма > System > main.py > process_audio_file

```python
def process_audio_file(audio_path, data_path):
    print(f"Текст повідомлення: {text}")

    stacking_model_cat, stacking_model_pol, vectorizer, label_encoder_cat, label_encoder_pol = prepare_model(data_path)
    processed_text = preprocess_text(text)
    X_tfidf = vectorizer.transform([processed_text])
    domain_features = np.array([[len(text), len(set(text.split())), count_domain_terms(processed_text)]])
    X_features = hstack([X_tfidf, domain_features])

    category = label_encoder_cat.inverse_transform(stacking_model_cat.predict(X_features))[0]
    polarity = label_encoder_pol.inverse_transform(stacking_model_pol.predict(X_features))[0]

    print(f"Категорія: {category}, Полярність: {polarity}")

    # Збереження в MySQL
    insert_message(text, category, polarity)

# Використання
audio_file_path = 'C:\\Users\\aleks\\OneDrive\\Рабочий стол\\ПРАКТИКИ\\ДИплом\\Програма\\voice_messages\\voice_message.wav'
data_path = 'c:\\Users\\aleks\\OneDrive\\Рабочий стол\\ПРАКТИКИ\\ДИплом\\Програма\\augmented_dataset.csv'
process_audio_file(audio_file_path, data_path)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code

```
[Running] python -u "c:\Users\aleks\OneDrive\Рабочий стол\ПРАКТИКИ\ДИплом\Програма\System\main.py"
Розпізнавання голосового повідомлення...
Текст повідомлення: Я не отримаю інформації по термінах доставки мої замовлення
Категорія: Статус замовлення, Полярність: Негативне
Повідомлення додано: Я не отримаю інформації по термінах доставки мої замовлення, Статус замовлення, Негативне

[Done] exited with code=0 in 7.116 seconds
```

8

ВИСНОВКИ

- Проведено тестування п'яти класичних методів (SVM, k-NN, Decision Trees, Naive Bayes, Logistic Regression) для визначення найкращого підходу. Результати показали, що жоден із методів не досяг точності вище 95%, що стало підставою для дослідження комбінованих моделей.

- Запропонована ансамблева модель SVM та Naive Bayes на основі StackingClassifier продемонструвала найкращі результати. Для класифікації категорій було досягнуто точності 95%, а для визначення полярності – 100%. Це свідчить про перевагу комбінації методів у порівнянні з окремими алгоритмами.

- Реалізовано попередню обробку тексту, система ефективно очищує текст від зайвих символів, виконує лематизацію, видаляє стоп-слова та враховує специфічні доменні терміни, що дозволяє покращити якість аналізу.

- Створено структуру бази даних MySQL для надійного збереження тексту повідомлень, їхніх категорій, полярності та часу створення. Це забезпечує легкий доступ до даних для подальшого аналізу та моніторингу.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Тези доповідей:

1. Левченко О.О., Шахматов І.О. Система обробки голосових повідомлень для гарячої лінії на основі методів машинного навчання. // Всеукраїнська науково-технічна конференція "Виклики та рішення в програмній інженерії". – Київ: ДУІКТ, 2024. – С. 109-112.
2. Левченко О.О., Шахматов І.О. Аналіз ефективності використання нейронних мереж для розпізнавання емоцій в голосових повідомленнях. // Всеукраїнська науково-технічна конференція "Виклики та рішення в програмній інженерії". – Київ: ДУІКТ, 2024. – С. 154-156.

ДОДАТОК Б. ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ

```
from utils import prepare_data
from svm_classifier import run_svm
from knn_classifier import run_knn
from decision_tree import run_decision_tree
from naive_bayes import run_naive_bayes
from logistic_regression import run_logistic_regression
from sklearn.model_selection import train_test_split

# Шляхи до файлів
data_path = 'c:\\Users\\aleks\\OneDrive\\Рабочий стол\\ПРАКТИКИ\\ДИплом\\Програма\\augmented_dataset.csv'
stopwords_path = 'c:\\Users\\aleks\\OneDrive\\Рабочий стол\\ПРАКТИКИ\\ДИплом\\Програма\\stopwords_ua.txt'

# Підготовка даних
X_tfidf, y_category, y_polarity = prepare_data(data_path, stopwords_path)

# Розділення для "Категорія"
X_train_cat, X_test_cat, y_train_cat, y_test_cat = train_test_split(X_tfidf, y_category, test_size=0.2, random_state=42)

# Розділення для "Полярність"
X_train_pol, X_test_pol, y_train_pol, y_test_pol = train_test_split(X_tfidf, y_polarity, test_size=0.2, random_state=42)

# Тестування моделей для "Категорія"
print("=== Тестування моделі SVM для 'Категорія' ===")
run_svm(X_train_cat, y_train_cat, X_test_cat, y_test_cat, label='Категорія')

print("\n=== Тестування моделі SVM для 'Полярність' ===")
run_svm(X_train_pol, y_train_pol, X_test_pol, y_test_pol, label='Полярність')

print("=== Тестування моделі k-NN для 'Категорія' ===")
run_knn(X_train_cat, y_train_cat, X_test_cat, y_test_cat, label='Категорія')

print("=== Тестування моделі k-NN для 'Полярність' ===")
run_knn(X_train_pol, y_train_pol, X_test_pol, y_test_pol, label='Полярність')

print("=== Тестування моделі Decision Trees для 'Категорія' ===")
run_decision_tree(X_train_cat, y_train_cat, X_test_cat, y_test_cat, label='Категорія')

print("=== Тестування моделі Decision Trees для 'Полярність' ===")
run_decision_tree(X_train_pol, y_train_pol, X_test_pol, y_test_pol, label='Полярність')

print("=== Тестування моделі Naive Bayes для 'Категорія' ===")
run_naive_bayes(X_train_cat, y_train_cat, X_test_cat, y_test_cat, label='Категорія')

print("=== Тестування моделі Naive Bayes для 'Полярність' ===")
run_naive_bayes(X_train_pol, y_train_pol, X_test_pol, y_test_pol, label='Полярність')

print("=== Тестування моделі Logistic Regression для 'Категорія' ===")
run_logistic_regression(X_train_cat, y_train_cat, X_test_cat, y_test_cat, label='Категорія')

print("=== Тестування моделі Logistic Regression для 'Полярність' ===")
run_logistic_regression(X_train_pol, y_train_pol, X_test_pol, y_test_pol, label='Полярність')

import pandas as pd
```

```
import re
from sklearn.feature_extraction.text import TfidfVectorizer

# Завантаження списку стоп слів з файлу
def load_stopwords(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        return set(word.strip() for word in file.readlines())

def preprocess_text(text, stop_words):
    text = re.sub(r'[^\w\s]', '', text)  # Видалення пунктуації
    text = re.sub(r'\d+', '', text)  # Видалення чисел
    text = text.lower().strip()  # Приведення до нижнього регістру
    text = " ".join([word for word in text.split() if word not in stop_words])  # Видалення стоп-слів
    return text

def prepare_data(data_path, stopwords_path):
    """
    Завантаження даних і попередня обробка:
    - Завантаження CSV-файлу
    - Обробка тексту
    - Векторизація тексту (TF-IDF)
    """
    data = pd.read_csv(data_path)
    stop_words = load_stopwords(stopwords_path)

    # Обробка тексту
    data['Текст повідомлення'] = data['Текст повідомлення'].apply(lambda x: preprocess_text(x, stop_words))

    # Розділення на ознаки та мітки
    X = data['Текст повідомлення']
    y_category = data['Категорія']
    y_polarity = data['Полярність']

    # Векторизація тексту
    vectorizer = TfidfVectorizer(max_features=5000)
    X_tfidf = vectorizer.fit_transform(X)

    return X_tfidf, y_category, y_polarity

import os
import mysql.connector
from datetime import datetime
import pandas as pd
import re
import numpy as np
from database_setup import insert_message  # Імпортуємо функцію збереження у MySQL
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.utils.class_weight import compute_class_weight
from scipy.sparse import hstack
from sklearn.metrics import classification_report, accuracy_score
from nltk.stem import WordNetLemmatizer
from speech_recognition import Recognizer, AudioFile

# Завантаження стоп-слів і доменних термінів
def load_stopwords(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        return set(word.strip() for word in file.readlines())

def load_domain_terms(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        return set(word.strip() for word in file.readlines())

# Шляхи до файлів
```

```
stopwords_path = 'c:\\Users\\aleks\\OneDrive\\Рабочий стол\\ПРАКТИКИ\\ДИплом\\Програма\\stopwords_ua.txt'

domain_terms_path = 'c:\\Users\\aleks\\OneDrive\\Рабочий стол\\ПРАКТИКИ\\ДИплом\\Програма\\domain_terms.txt'

stop_words = load_stopwords(stopwords_path)

domain_terms = load_domain_terms(domain_terms_path)


# Попередня обробка тексту

lemmatizer = WordNetLemmatizer()

def preprocess_text(text):

    text = re.sub(r'[^\w\s]', '', text)

    text = re.sub(r'\d+', '', text)

    text = text.lower().strip()

    text = " ".join([lemmatizer.lemmatize(word) for word in text.split() if word not in stop_words])

    return text


# Додавання доменних термінів

def count_domain_terms(text):

    return sum(1 for word in text.split() if word in domain_terms)


# Конвертація голосового повідомлення в текст

def transcribe_audio(audio_path):

    recognizer = Recognizer()

    with AudioFile(audio_path) as source:

        audio = recognizer.record(source)

    text = recognizer.recognize_google(audio, language="uk-UA")

    return text


# Підготовка моделі

def prepare_model(data_path):

    data = pd.read_csv(data_path)

    data['Текст повідомлення'] = data['Текст повідомлення'].apply(preprocess_text)

    data['domain_term_count'] = data['Текст повідомлення'].apply(count_domain_terms)

    data['length'] = data['Текст повідомлення'].apply(len)

    data['unique_words'] = data['Текст повідомлення'].apply(lambda x: len(set(x.split())))


    X_text = data['Текст повідомлення']

    X_additional = data[['length', 'unique_words', 'domain_term_count']]

    y_category = data['Категорія']

    y_polarity = data['Полярність']


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

    X_tfidf = vectorizer.fit_transform(X_text)

    X_features = hstack([X_tfidf, X_additional])


    # Категорія

    label_encoder_cat = LabelEncoder()

    y_category_enc = label_encoder_cat.fit_transform(y_category)

    class_weights_cat = compute_class_weight('balanced', classes=np.unique(y_category_enc), y=y_category_enc)

    class_weights_cat_dict = dict(zip(np.unique(y_category_enc), class_weights_cat))


    stacking_model_cat = StackingClassifier(

        estimators=[

            ('Naive Bayes', MultinomialNB()),

            ('SVM', SVC(kernel='linear', class_weight=class_weights_cat_dict, probability=True, random_state=42))

        ],

        final_estimator=LogisticRegression()

    )

    stacking_model_cat.fit(X_features, y_category_enc)


    # Полярність

    label_encoder_pol = LabelEncoder()

    y_polarity_enc = label_encoder_pol.fit_transform(y_polarity)

    class_weights_pol = compute_class_weight('balanced', classes=np.unique(y_polarity_enc), y=y_polarity_enc)
```

```
    class_weights_pol_dict =
dict(zip(np.unique(y_polarity_enc), class_weights_pol))


    stacking_model_pol = StackingClassifier(
        estimators=[
            ('Naive Bayes', MultinomialNB()),
            ('SVM', SVC(kernel='linear',
class_weight=class_weights_pol_dict, probability=True,
random_state=42))
        ],
        final_estimator=LogisticRegression()
    )
    stacking_model_pol.fit(X_features, y_polarity_enc)


    return stacking_model_cat, stacking_model_pol,
vectorizer, label_encoder_cat, label_encoder_pol


# Основний робочий процес
def process_audio_file(audio_path, data_path):
    print("Розпізнавання голосового повідомлення...")
    text = transcribe_audio(audio_path)
    print(f"Текст повідомлення: {text}")


    stacking_model_cat, stacking_model_pol, vectorizer,
label_encoder_cat, label_encoder_pol =
prepare_model(data_path)
    processed_text = preprocess_text(text)
    X_tfidf = vectorizer.transform([processed_text])
    domain_features = np.array([[len(text),
len(set(text.split())),
count_domain_terms(processed_text)]])
    X_features = hstack([X_tfidf, domain_features])


    category =
label_encoder_cat.inverse_transform(stacking_model_ca
t.predict(X_features))[0]
    polarity =
label_encoder_pol.inverse_transform(stacking_model_p
ol.predict(X_features))[0]


    print(f"Категорія: {category}, Полярність:
{polarity}")
    # Збереження в MySQL
    insert_message(text, category, polarity)


# Використання
audio_file_path = 'C:\\Users\\aleks\\OneDrive\\Рабочий
стол\\ПРАКТИКИ\\ДИплом\\Програма\\voice_messag
es\\voice_message.wav'
data_path = 'c:\\Users\\aleks\\OneDrive\\Рабочий
стол\\ПРАКТИКИ\\ДИплом\\Програма\\augmented_da
taset.csv'
process_audio_file(audio_file_path, data_path)
```