Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Програмні засоби розпізнавання голосових повідомлень за допомогою машинного навчання

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІз-21-1
Владислав ДУДНИК
(власне ім'я, прізвище)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник: ас. Анастасія ЛУЦЕНКО
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Дуднику Владиславу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмні засоби розпізнавання голосових повідомлень за допомогою машинного навчання

затверджена наказом по університету від " 5 " травня 2025 р. № 72 Стз

2. Термін подання здобувачем роботи до екзаменаційної комісії 17 червня 2025 р.

3. Вхідні дані до роботи Аудіозаписи, MFCC

4. Перелік питань, що потрібно опрацювати у роботі

1. Системний аналіз предметної області
2. Вибір і обґрунтування методу розв'язування
3. Програмна реалізація
4. Результати обчислювального експерименту
5. Аналіз можливих застосувань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд презентація – 11 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання та аналіз літератури	05.05.2025–18.05.2025	виконано
2	Аналіз існуючих засобів моделювання	19.05.2025–27.05.2025	виконано
3	Вибір алгоритмів	28.05.2025–05.06.2025	виконано
4	Вибір програмних засобів	06.06.2025–12.06.2025	виконано
5	Реалізація та тестуваня	13.06.2025–14.06.2025	виконано
6	Аналіз результатів	14.06.2025–15.06.2025	виконано
7	Оформлення ПЗ	14.06.2025–15.06.2025	виконано

Дата видачі завдання " 5 " травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Анастасія ЛУЦЕНКО
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 60 с., 37 рис., 2 табл., 1 дод., 33 джерел.

РОЗПІЗНАВАННЯ ГОЛОСУ, МАШИННЕ НАВЧАННЯ, MFCC, SVM, GMM, БІОМЕТРИЧНА БЕЗПЕКА.

Метою кваліфікаційної роботи є створення системи розпізнавання голосу з високою точністю ідентифікації мовця та класифікації за статтю.

У ході виконання кваліфікаційної роботи було досліджено сучасні методи машинного навчання, проведено аналіз предметної області та існуючих підходів. Для реалізації було обрано алгоритми MLP, GMM і SVM, а систему створено в Python у середовищі Google Colab. Архітектура обробки включає видалення шуму, вилучення ознак MFCC, навчання класифікаторів і перевірку результатів.

Система протестована як на власних, так і на відкритих наборах даних (Kaggle), що підтвердило її стабільність. Досягнута точність класифікації статі становить до 100% для окремих алгоритмів, а ідентифікація мовця за допомогою GMM виявилася ефективною. Результати демонструють потенціал використання у кібербезпеці, фінансових сервісах та інтелектуальних інтерфейсах. Модель також може бути масштабована для великих даних та інтегрована з IoT-пристроями.

ABSTRACT

Bachelor’s thesis: 60 pages, 37 figures, 2 tables, 1 appendices, 33 sources

VOICE RECOGNITION, MACHINE LEARNING, MFCC, SVM, GMM, BIOMETRIC SECURITY.

The major goal of this thesis is to develop a voice recognition system with high accuracy in speaker identification and gender classification.

The study explored modern machine learning methods, analyzed the domain, and reviewed existing approaches. MLP, GMM, and SVM were selected as the main algorithms, and the system was implemented in Python using Google Colab. The processing architecture includes noise reduction, MFCC feature extraction, classifier training, and result verification.

The system was tested on both custom and public datasets (Kaggle), which confirmed its stability. The results showed up to 100% accuracy in gender classification for some algorithms and effective speaker identification using GMM. These findings highlight the potential of the model in cybersecurity, financial services, and intelligent interfaces. The system can also be scaled for big data and integrated with IoT devices.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ASR – Automatic Speech Recognition

CSV – Comma-Separated Values file format

DCT – Discrete Cosine Transform

DL – Deep Learning

GMM – Gaussian Mixture Model

GUI – Graphical User Interface

MFCC – Mel-Frequency Cepstral Coefficients

ML – Machine Learning

MLP – Multilayer Perceptron

PCM – Pulse-Code Modulation

RBFN – Radial Basis Function Network

STFT – Short-Time Fourier Transform

SVM – Support Vector Machine

VAD – Voice Activity Detection

WAV – Waveform Audio File Format

WER – Word Error Rate

ВСТУП

Розпізнавання голосу – це складний та комплексний процес, який проходить через виконання послідовності компонентів: (1) Попередня обробка мовлення; (2) Вилучення ознак; (3) Класифікація мовлення; та (4) Розпізнавання. Модулі вилучення ознак у діагностиці голосу виражаються в аналізі спектрограм та mel-спектрограм, пов'язаних з отриманням спектральних ознак. Це набори статичних або динамічних кепстральних коефіцієнтів Mel-Frequency (MFCC). Вхідні голосові спектрограми отримуються за допомогою короткочасного перетворення Фур'є (STFT) або дискретного косинусного перетворення (DCT). Згенеровані набори даних ознак служать вхідними параметричними одиницями для різної аналітики вимірів для створення моделей класифікації. Наукові класифікатори базуються на голосових класифікаторах з машинним навчанням (ML) та глибоким навчанням (DL) як методом опорних векторів (SVM), згорткових нейронних мережах (CNN), довгостроковій та короткочасній пам'яті (LSTM), логічній рекурентній одиниці (GRU) та двонаправленій мережі LSTM (Bi-LSTM). Алгоритм та нейронна ефективність аналізуються з використанням гетерогенних наборів даних за такими концептуальними завданнями: (1) Класифікація статі (GC); (2) Верифікація мовця (SV); (3) Ідентифікація мови (LID); (4) Ідентифікація регіонального діалекту (DID); та (5) Класифікація каналів (CC).

Робота зосереджена на покращені системи безпеки за допомогою підходу глибокого навчання для розпізнавання голосу. Крім того, у дослідженні детальніше розглядається використання доступних наборів даних з центральної бази даних. Основна мета цієї статті – застосувати найбільш підходящі методології машинного навчання та глибокого навчання для розпізнавання будь-якої особи за просодичною ознакою мовлення з заданої центральної бази даних.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз методів автоматичного розпізнавання мовлення

Автоматичне розпізнавання мовлення (ASR) – це дуже популярна технологія, яка широко застосовується в реальному житті та бізнес-додатках. Мобільні телефони забезпечують функції перетворення мовлення на текст за допомогою різноманітних програм, голосові помічники направляють вхідні дзвінки, «спілкуючись» з клієнтами, які щодня телефонують до свого банку чи страхової компанії, а водії керують своїми автомобілями за допомогою голосу. Те, що робить її настільки поширеною, – це наш (людський) природний спосіб спілкування – мовлення. Ми вчимося говорити досить рано та практикуємо це щодня, на відміну від спілкування з різними технологіями, яке ми зазвичай робимо через різні інтерфейси користувача. Усі інтерфейси побудовані по-різному, і це ускладнює спілкування з новими технологіями. Нам доводиться вчитися постійно, коли ми отримуємо щось нове в наборі наших гаджетів чи програм. Нещодавні дослідження голосових інтерфейсів дають нам уявлення про те, що в майбутньому проблема величезної різноманітності інтерфейсів користувача буде вирішена [28, 29].

Головна ідея розпізнавання мовлення полягає в перетворенні захоплених аудіосигналів у відповідне текстове представлення (рис. 1.1).

Рисунок 1.1 – Схема розпізнавання мовлення

Система ASR представляє вхідну послідовність звуків (або акустичний вхід) X = {x1, x2, ..., XT} довжини Tas, а результуючу послідовність Y = {y1, y2, ..., YL} (яка може бути символами, фрагментами слів або словами) довжини L.

Моделювання систем ASR – це завдання створення генеративної моделі [30]. Класичний спосіб створення ASR полягає у побудові мовної моделі, моделі вимови та акустичної моделі. Донедавна всі три компоненти були важливими для задачі розпізнавання мовлення (рис. 1.2).

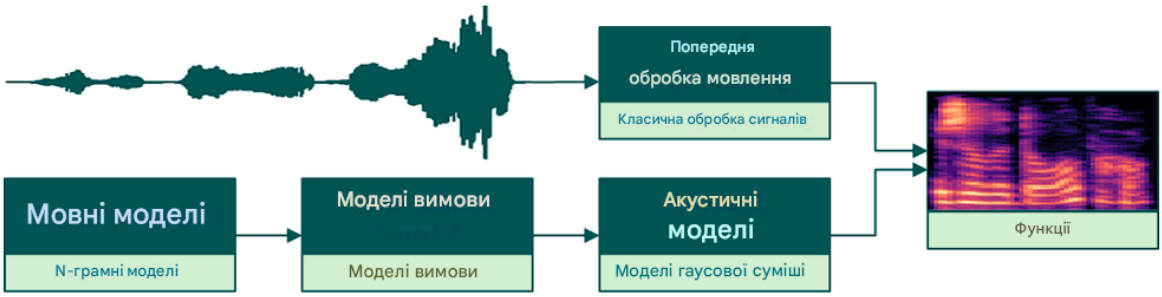


Рисунок 1.2 – Моделювання багатомодульної системи автоматичного розпізнавання мовлення

З мовної моделі ми створюємо послідовність слів. Модель вимови дає результат того, як вимовляється певне слово. Ми побачимо, що воно записується як послідовність фонем, які є основними одиницями звуку (послідовність лексем). Модель вимови перетворює послідовність тексту на послідовність лексем вимови. Після цього вона подається до акустичної моделі, яка створює результат звучання заданої лексеми.

У цьому конвеєрі кожен компонент має свою статистичну модель. Кінцевим результатом спільної роботи всіх моделей є визначення найімовірнішої текстової послідовності Y' = {y1, y2, ..., YL} із заданими даними (аудіохарактеристиками) X = {X1, X2, ..., XT}:

$$Y' = \arg Y \max_Y p(X|Y)p(Y).$$

Одна з цікавих ідей була запропонована командою дослідників з Facebook та Microsoft. У роботі [32] вони використовували стратегію ініціалізації навчання вчителя та учня для перенесення знань зі складної офлайн-моделі наскрізного типу до онлайн-моделі розпізнавання мовлення наскрізного типу. Це допомогло їм усунути потребу в якісному лексиконі чи будь-якому іншому лінгвістичному доповненні. Цю ідею було оцінено на Microsoft Cortana. Це завдання NLP, підключене до персонального асистента. Метод, запропонований авторами, показує 19% покращення WER.

1.2 Програми для розпізнавання голосу

Програмне забезпечення для розпізнавання голосу, також відоме як розпізнавання мовлення або перетворення мовлення на текст, – це технологія, яка перетворює вимовлені слова на команди навігації, введені користувачем дані або письмовий текст. Використовуючи передові алгоритми та машинне навчання, системи розпізнавання голосу аналізують аудіовхідні дані, ідентифікують вимовлені слова та перетворюють їх на команди користувача та транскрипції з надзвичайною точністю. Ця технологія значно розвинулася протягом багатьох років завдяки складним алгоритмам, здатним розпізнавати різні акценти, діалекти та мовленнєві моделі.

Розробка контенту з урахуванням потреб користувачів програмного забезпечення для розпізнавання голосу означає створення цифрового досвіду, де користувачі можуть легко переміщатися та виконувати команди, використовуючи лише свій голос. Ось деякі способи зробити контент сумісним з програмним забезпеченням для розпізнавання голосу:

- Створення інтерактивних елементів та вхідних даних, які можна легко ідентифікувати та прочитати;
- Уникайте загальних текстів посилань, таких як «Натисніть тут» або «Читати далі», особливо коли кілька посилань або кнопок використовують

один і той самий текст;

- Використання заголовків та підписів для чіткої ідентифікації певних розділів або регіонів на сторінці чи в програмі;

- Забезпечення правильного зв'язку між формою та підписами вводу та полем вводу, яке вони описують;

- Обмеження використання складних взаємодій (наприклад, точне введення, введення кількома клавішами, перетягування тощо);

- Грамотна обробка помилок та допомога користувачам у відновленні, пропонуючи альтернативи та пропозиції щодо виправлення будь-яких проблем.

Ознайомтеся з деякими з веб-ресурсів нижче, щоб дізнатися про більше способів зробити цифровий досвід для користувачів програмного забезпечення для розпізнавання голосу якомога інклюзивнішим та приємнішим:

- Google Cloud Speech-to-Text для синтезу природного звучання мовлення та потокової передачі аудіо в реальному часі. (0,016 за 1 хвилину/міс.);

- Amazon Transcribe для автоматичного розпізнавання мовлення (ASR) та послуг транскрипції мовлення в режимі реального часу. (0,024 за 1 хвилину/міс.);

- Інтелектуальні служби розпізнавання Microsoft Custom Recognition Intelligent Services (CRIS) для налаштованого механізму перетворення мовлення на текст та налаштування тексту. ($1/год.);

- API мовлення Microsoft Bing для взаємодії з користувачем у режимі реального часу та розширені алгоритми для обробки розмовної мови. (25 доларів США/1000 транзакцій);

- Шепіт для багатомовності та зручного інтерфейсу для інтеграції з бізнес-додатками. ($0.006/хвилина);

- IBM Watson Speech-to-Text для алгоритмів глибокого навчання на основі штучного інтелекту та налаштовуваного розпізнавання мовлення для

створення кращого контенту. (Доступно за запитом);

- HTK для синтезу мовлення, розпізнавання символів та секвенування ДНК для оптимізації доступності. (Доступно за запитом);

- Deepgram для аналізу настроїв, узагальнення та аналізу настроїв для автоматизації контенту. ($4.50/година);

- Otter.ai для ідентифікації мовця та користувацький словник для інтерпретації нових слів. ($8.33/міс.).

1. Amazon Transcribe

Переваги: Нам не потрібно вручну обробляти аудіофайл, тобто змінювати формат файлу порівняно з конкурентом. Підтримується багато форматів аудіофайлів. Найкраще в Transcribe те, що він може визначити, скільки спікерів є і який спікер що говорив, за допомогою позначки часу. Він також дозволяє додавати словниковий запас. Це найкращий доступний і точний сервіс, який задовольняє наші потреби.

Недоліки: Він не розпізнає числові розряди як вимовлені; він перетворює їх на «один» або «два» замість 1, 2. Використання власного словника є дуже виснажливим завданням.

Microsoft Custom Recognition Intelligent Services (CRIS)

Переваги: CRIS – це інструмент, який допомагає подолати блокування розпізнавання мовлення. Під час міжнародної роботи важливо блокувати фоновий шум. Під час надсилання текстових повідомлень корисно використовувати оптимізацію перетворення мовлення на текст.

Недоліки: Впровадження програмного забезпечення може бути трудомістким і непростим в налаштуванні. Крім того, ціна продукту завищена, що ускладнює обґрунтування рентабельності інвестицій.

2. Microsoft Bing Speech API

Переваги: Це програмне забезпечення дуже просте у використанні, воно значно може спростити роботу. Воно допомагає знайти донорів на новому рівні та залучити офіс.

Недоліки: Переклад може бути дивним, але ви зрозумієте зміст.

3. Whisper

Переваги: Те, що це програмне забезпечення з відкритим вихідним кодом і дуже вигідна ціна при використанні з API OpenAI ($0,006 за хвилину), – це чудово. А Hugging Face також пропонує налаштовані моделі шепоту, такі як шепот JAX. Хоча їх не рекомендується використовувати у виробництві. Це робить їх ідеальними для використання в організаційних чат-ботах тощо.

Недоліки: Головний недолік полягає в тому, що якщо у нас є повна транскрипція, то модель не може повністю транскрибувати її за один раз, оскільки вона розроблена для використання лише 30 секунд аудіофайлу.

4. IBM Watson Speech-to-Text

Переваги: Це одна з найкращих програм для перетворення мовлення на текст, вона добре розпізнає слова. Вона має такі функції, як режим реального часу, користувацькі моделі та визначення ключових слів.

Недоліки: Точність сервісу перетворення мовлення на текст IBM Watson не завжди однакова. Він не зосереджений лише на одній людині, але якщо оратор розпізнає будь-яку мову, він намагається перетворити її на текст, що створює спотворення в текстовому файлі.

5. HTK

Переваги: Простий інструмент для вилучення всіх функцій, моделі навчання фону, детальний посібник користувача та хороша підтримка на форумах.

Недоліки: Продуктивність була низькою, і не можна знайти жодної корисної технічної документації, оскільки вона не була призначена для початківців.

6. Deepgram

Переваги: Він добре працює в тихому середовищі, але якість транскрипції падає, коли є шумові перешкоди. І іноді він неправильно розуміє технічну термінологію, що мені доводиться виправляти за замовчуванням. Однак це не так вже й погано; просто було б добре, якби

служба підтримки клієнтів була трохи швидшою, тому що іноді їм потрібен деякий час, щоб відповісти на складніші запитання.

Недоліки: Обмеження, з яким ми зіткнулися в Deepgram, полягає в веденні щоденника на зустрічах, де є кілька учасників. Це має тенденцію плутати спікерів під час групових обговорень, де беруть участь кілька спікерів і говорять голосами, що перетинаються. Для забезпечення точності транскрипцій та ведення щоденника потрібна ручна перевірка. Ще одна сфера, де ми стикаємося з проблемами, — це нездатність ідентифікувати та автоматично називати спікерів із записів зустрічей.

7. Otter.ai

Переваги: Можна повністю зосередитися на тих, з ким спілкуюся під час дзвінка, не маючи потреби постійно робити нотатки. Розмови можуть стати більш вільними, можна ставити більше запитань і дізнатися набагато більше інформації, Otter робитиме нотатки та записуватиме аудіостенограму.

Недоліки: Іноді точність транскрипції падає через сильні акценти або фоновий шум. Крім того, безкоштовний план має обмежені функції, а резюме зі штучним інтелектом можна було б краще налаштувати.

2 МЕТОДИКА ВИКОНАННЯ

2.1 Вхідні дані

Машинне навчання є підмножиною штучного інтелекту, що передбачає розробку алгоритмів та моделей, що дозволяють комп'ютерам навчатися на основі даних, виявляти закономірності та робити прогнози чи рішення без явного програмування. Це дозволяє машинам покращувати свою продуктивність з часом завдяки досвіду та аналізу даних.

У машинному навчанні є два фундаментальні підходи: навчання з учителем та навчання без учителя.

Кероване навчання передбачає навчання моделі з використанням маркованих даних, де вхідні дані поєднуються з відповідними вихідними мітками. Мета полягає в тому, щоб модель вивчила відповідність між вхідними та вихідними змінними, що дозволить їй робити точні прогнози або класифікації на основі невидимих даних. Модель навчається на наданих прикладах та керується відомими правильними відповідями, що дозволяє їй узагальнювати та робити прогнози на основі нових, немаркованих даних.

З іншого боку, навчання без учителя має справу з немаркованими даними, де модель прагне виявити закономірності, структури або зв'язки в даних без будь-яких попередньо визначених вихідних міток. Мета полягає у виявленні прихованих ідей або груп у даних, часто за допомогою таких методів, як кластеризація або зменшення розмірності. Навчання без учителя дозволяє моделі навчатися самостійно та виявляти притаманні закономірності або структури, які можуть бути не очевидними для спостерігачів-людей.

Існує кілька популярних алгоритмів машинного навчання, кожен з яких має свій унікальний підхід та функціональність. Ось кілька прикладів:

- Алгоритм лінійної регресії використовується для навчання з

учителем та застосовується для моделювання зв'язку між залежною змінною та однією або кількома незалежними змінними. Він підганяє лінійне рівняння до точок даних для здійснення прогнозів;

- Дерева рішень – це універсальні алгоритми, що використовуються як для навчання з учителем, так і для навчання без учителя. Вони створюють деревоподібну модель рішень та їх можливих наслідків. Ставлячи низку питань, дерева рішень класифікують або прогнозують результати на основі вхідних ознак;
- Випадковий ліс – це алгоритм ансамбевого навчання, який поєднує кілька дерев рішень. Він створює ліс дерев рішень і робить прогнози шляхом усереднення результатів окремих дерев. Випадковий ліс відомий своєю стійкістю та здатністю обробляти складні набори даних;
- Метод опорних векторів (SVM) – це алгоритм навчання з учителем, який використовується для задач класифікації та регресії. Він визначає оптимальну гіперплощину, яка розділяє точки даних на окремі класи, одночасно максимізуючи межу між цими класами. Примітно, що SVM демонструє універсальність в обробці як лінійних, так і нелінійних даних, що робить його цінним інструментом у різних областях;

K-середніх – це алгоритм навчання без учителя, який використовується для кластеризації. Він розділяє дані на k кластерів на основі подібності. Він ітеративно призначає точки даних кластерам та оновлює центроїди кластера до досягнення збіжності, прагнучи мінімізувати внутрішньокластерну суму квадратів;

Для реалізації було обрано підхід, який полягає в тому, щоб спочатку отримати зразки голосу, а потім виділити ознаки з аудіозразків. Для вилучення ознак ми використовуватимемо техніку MFCC. Після цього для розпізнавання будуть використані два шари. По-перше, для категоризації осіб з вибірок як чоловіків та жінок будуть використані 9 алгоритмів класифікації, а саме: багатошаровий перцептрон, радіальна базисна функціональна мережа, дерево рішень, випадковий ліс, градієнтне посилення, наївний байєсівський

алгоритм, K-найближчий сусід, логістична регресія та метод опорних векторів, і буде прийнято той алгоритм, який дасть найкращий результат. Після цього буде взято зразок голосу та навчено за допомогою прийнятого алгоритму класифікатора.

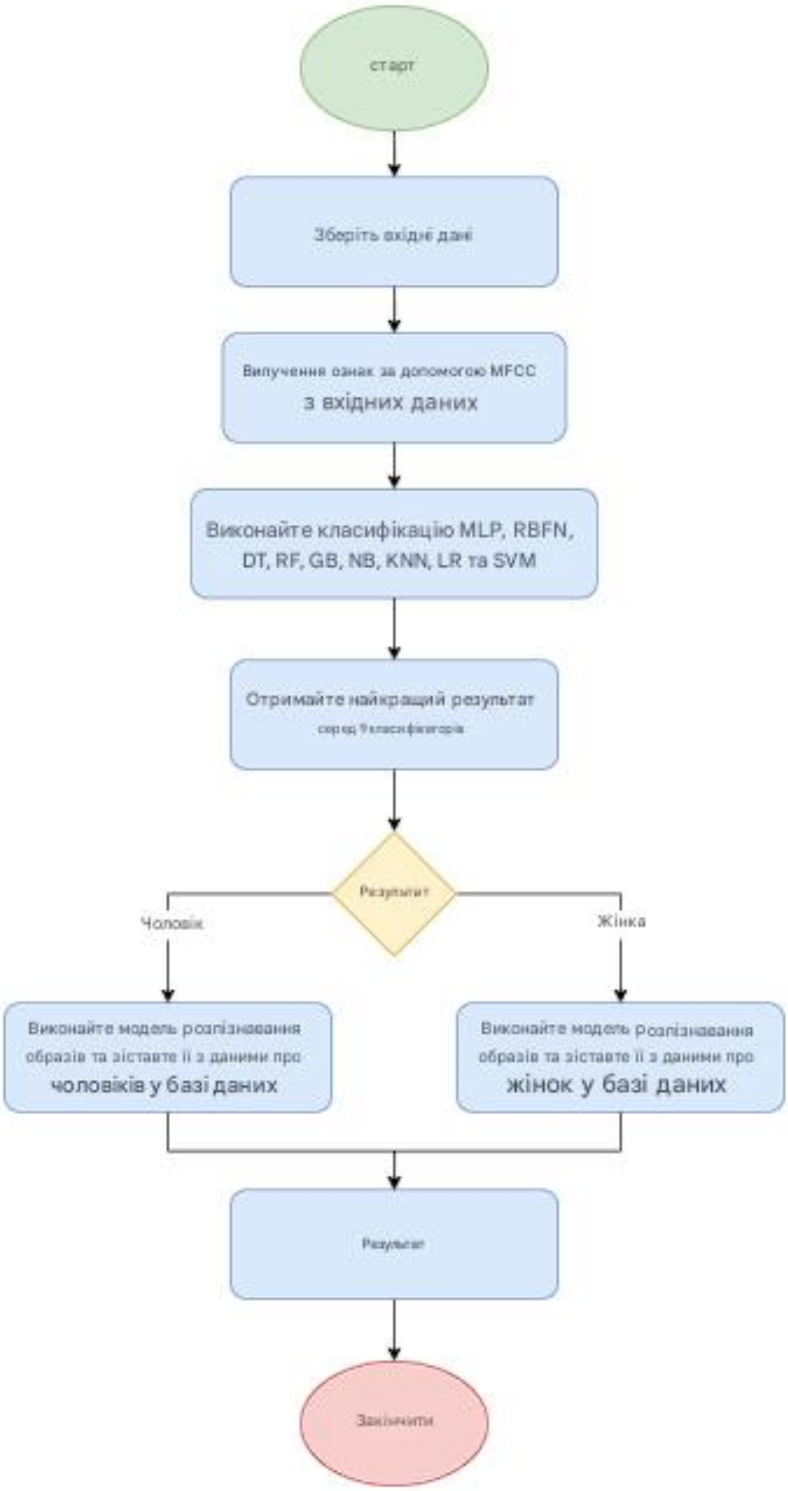


Рисунок 2.1 – Блок-схема запропонованої моделі виявлення голосу

У рамках реалізації системи за допомогою алгоритму класифікатора, алгоритм розпізнавання образів, такий як GMM, буде виконано на другому шарі для виявлення особи, і під час використання моделі розпізнавання образів, якщо результат класифікатора був чоловічим, то в базі даних він буде перевірений лише з чоловічими зразками. Якщо результат був жіночим, то він буде перевірений з жіночими зразками.

Наш запропонований процес розпізнавання голосу, який відповідає за обробку, класифікацію та створення прогнозів, складається з чотирьох основних етапів:

- Попередня обробка вхідних даних: на цьому етапі виконується вилучення ознак. Попередня обробка необхідна, оскільки очищення та трансформація даних – це способи видалення викидів та стандартизації даних, щоб їх можна було легше використовувати для розробки моделі;
- Обробка: Цей етап пов'язаний з класифікацією вхідних даних за гендерними групами за допомогою 9 класифікаторів та побудовою 2 різних фреймів даних;
- Виявлення: На цьому етапі GMM застосовується до двох розділених кадрів даних для виявлення аудіозразків за допомогою витягнутих ознак. Він виконується на всьому наборі даних, що містить як чоловічі, так і жіночі особи, і результати будуть порівняні;
- Прогнози: Цей етап призначений для прогнозування балів на двох розділених фреймах даних.

Вхідні дані вставляються на етап попередньої обробки, а потім використовуються моделями класифікатора для побудови двох фреймів даних; обидва з яких застосовуються до GMM для виявлення вибірок даних. Для досягнення цієї мети попередньо оброблені вхідні дані поділяються на дві групи; одна група використовується для навчання, а інша група - для перевірки точності моделі, після чого здійснюється стратифікація даних.

2.2 Попередня обробка даних

Набір аудіоданих дещо відрізняється від інших наборів даних. Тому що в наборах аудіоданих нам потрібно витягти характеристики із зразків аудіо. Але ми вручну зібрали аудіозразки голосу різних людей через платформи соціальних мереж. Набір даних містить понад 150 зразків голосу Waveform Audio File Format від чоловіків і жінок різного віку.

Рисунок 2.2 – Блок-схема процесу створення набору даних

На рисунку 2.2 показано, як було створено набір даних з використанням функцій, витягнутих із зразків аудіо, зібраних від окремих осіб. Набір даних містить різні функції, такі як спектральна смуга пропускання, спектральний центроїд і, що найважливіше, MFCC. Цей набір даних було зібрано від окремих осіб і в основному не містить даних,

необхідних для виявлення зразків голосу. У результаті його потрібно обробити, перш ніж його можна буде використовувати для створення запропонованої моделі виявлення голосу в цьому дослідженні.

Видалення шуму. Першим кроком, який ми зробили, було зменшення шуму від галасливого аудіо зразки. Шум в аудіо відноситься до залишкового низькорівневого та заважаючого звуку на фоні [20]. Для видалення шуму ми використовували програмне забезпечення Adobe Audition 2021. Видалення шуму з шумного аудіосигналу є необхідним, оскільки він може порушити характеристики аудіозразка. На рисунках 2.3 та 2.4 ми зображено співвідношення шуму та вихідного сигналу до та після видалення шуму.

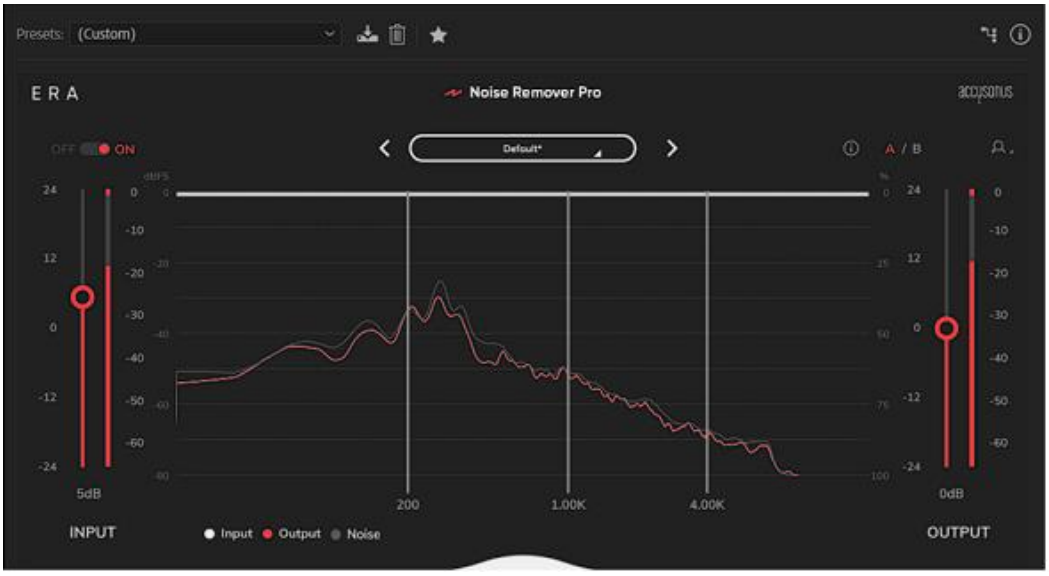


Рисунок 2.3 – Перед видаленням шуму

Вилучення ознак. Для вилучення ознак ми використовували Mel-Frequency Cepstral Coef-фахівців (MFCC). Це найпопулярніший вибір для розпізнавання або ідентифікації мовлення з 1970 року. Мова в основному є згортанням частотної характеристики голосового тракту з голосовим пульсом. Функція формалізації мовлення подається виразом:

$$\log(X(t))=\log(E(t))+\log(H(t)),$$

де log(H(t)) – це частотна характеристика голосового тракту, необхідна для розділення компонентів під час формалізації мовлення. На рисунку 2.5 описано механізм роботи MFCC.

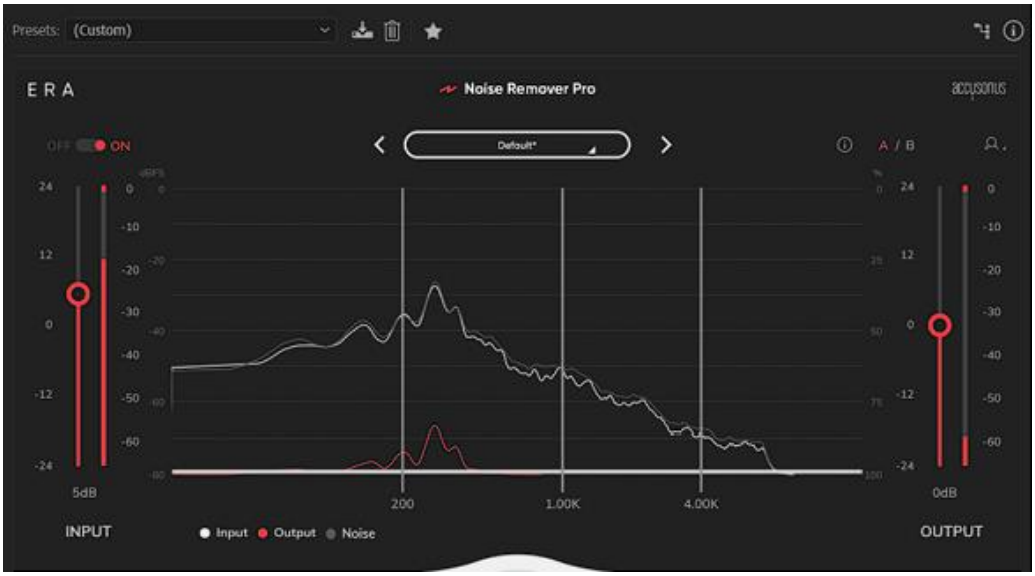


Рисунок 2.4 – Після видалення шуму

```
plt.figure(figsize=(20, 10))
librosa.display.specshow(mfccs,
                         x_axis="time",
                         sr=sr)
plt.colorbar(format="%+2.f")
plt.show()
```

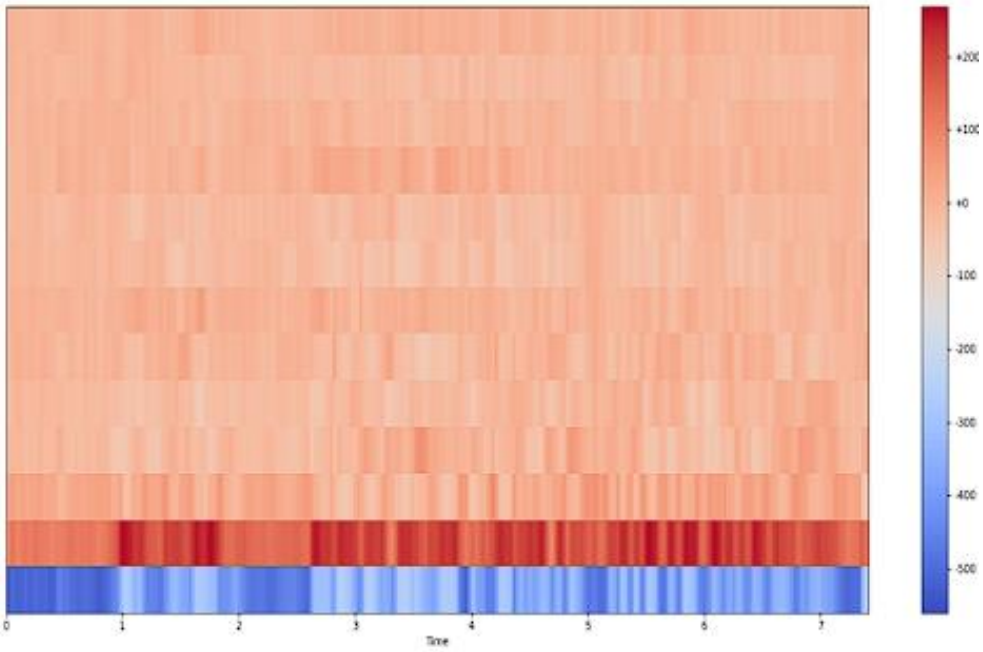


Рисунок 2.5 – Вилучення MFCC з часом

Ми завантажили зразок аудіо з нашого набору даних та витягли ознаки MFCC разом із другим порядком MFCC. Вищі порядки розраховуються, щоб показати, як ознака змінюється з кожним порядком. На рисунках 2.6, 2.7 показано графіки MFCC.

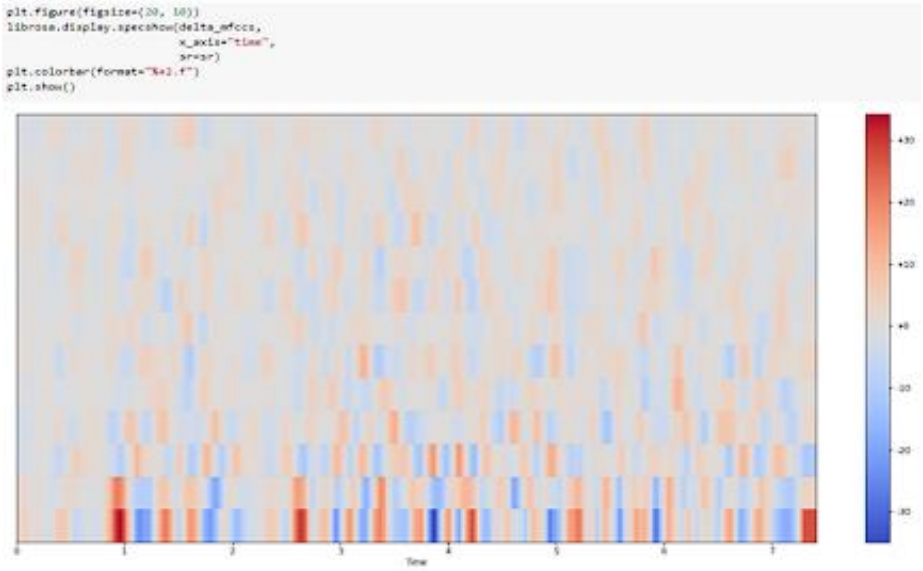


Рисунок 2.6 – Вилучення дельта MFCC з часом

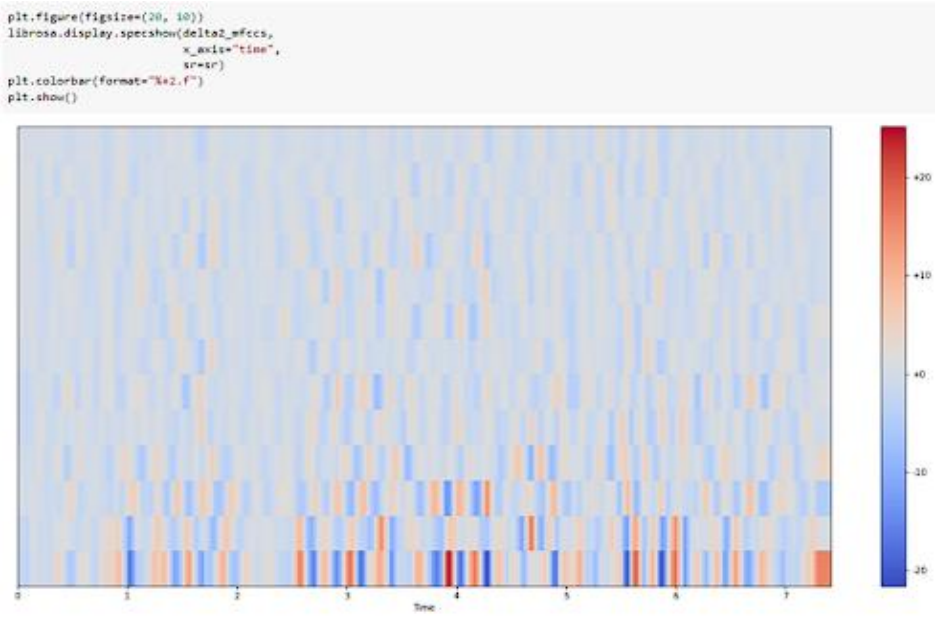


Рисунок 2.7 - Вилучення MFCC delta2 з часом

2.3 Класифікатори та методи розпізнавання образів

2.3.1 MLP

У моделі ми використовували MLP для ідентифікації статі за зразками голосу, оскільки ідентифікація статі за допомогою MLP має дуже високий коефіцієнт розпізнавання [17]. Багатошаровий перцептрон - це штучна нейронна мережа з прямим зв'язком, яка конструює вихідні набори з колекції вхідних даних. Між вхідним і вихідним шарами MLP генерується орієнтований граф. Це поширений алгоритм глибокого навчання, який навчає мережу за допомогою методу навчання з учителем, який називається зворотним поширенням [18]. Для зменшення помилок навчання включає коригування параметрів моделі, або ваг, зміщень, а зворотне поширення коригує ці ваги та зміщення відносно помилки. На рисунку 2.8 зображено процес MLP.

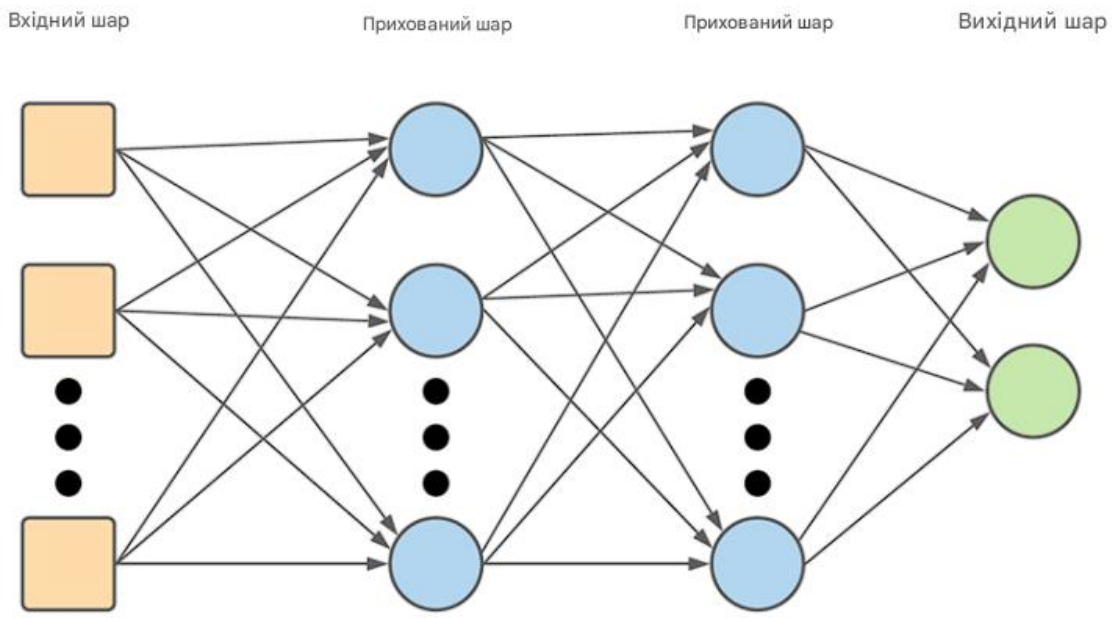


Рисунок 2.8 – Зображення моделі багаторівневого персептрона (MLP).

2.3.2 РБФН

Радіальна базисна функціональна мережа (РБФН) структурно подібна до MLP. RBFN застосовується для ідентифікації мовця шляхом аналізу ознак мовлення, зокрема MFCC. Однією з найбільших переваг RBFN є те, що вона застосовна в більшості вимірів через низькі обмеження на обробку даних. Цей метод зазвичай використовує концепцію евклідового простору Rn. У цьому випадку функція має вигляд:

$$s(x) = \sum_{j=0}^{m} \lambda_j \varphi(\|x - x_j\|), x \in R^n$$

Тут x_j – це точки даних, φ – одновимірна величина, а λ; – параметри скалера [20]. На рисунку 2.9 показано структуру RBFN.

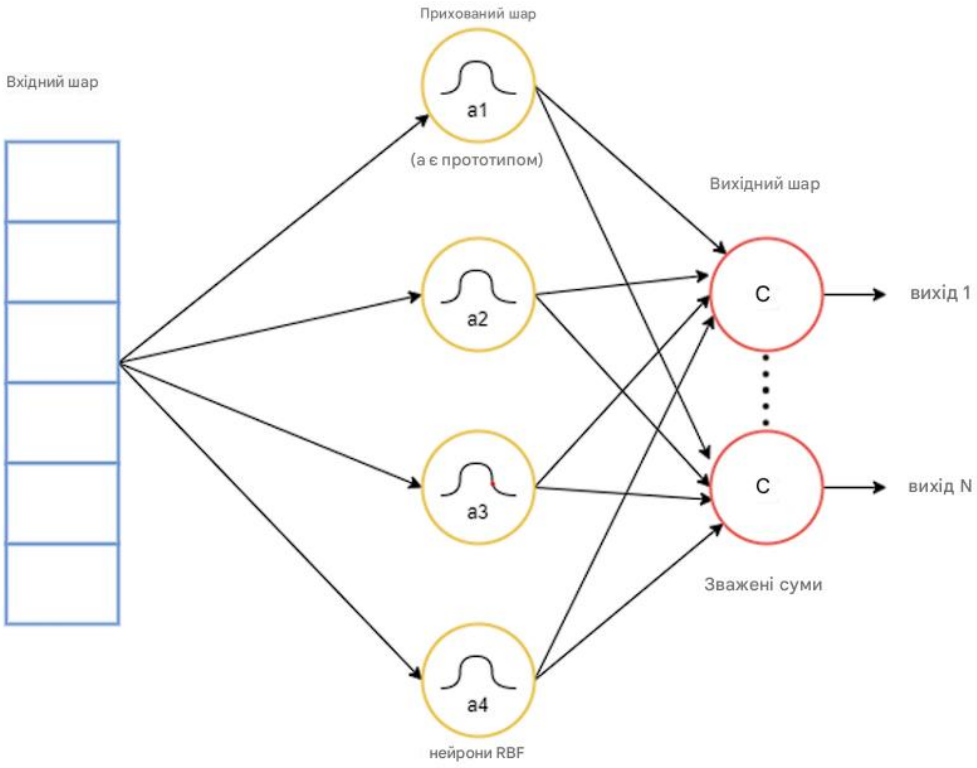


Рисунок 2.9 – Зображення моделі мережі радіальних базових функцій (RBFN)

2.3.3 Дерево рішень

Дерева рішень використовуються як інструмент для прийняття рішень, що використовує концепцію деревоподібної моделі варіантів та їх можливих результатів. Модель алгоритму полягає в тому, що вона повністю встановлена на основі операторів умовного керування.

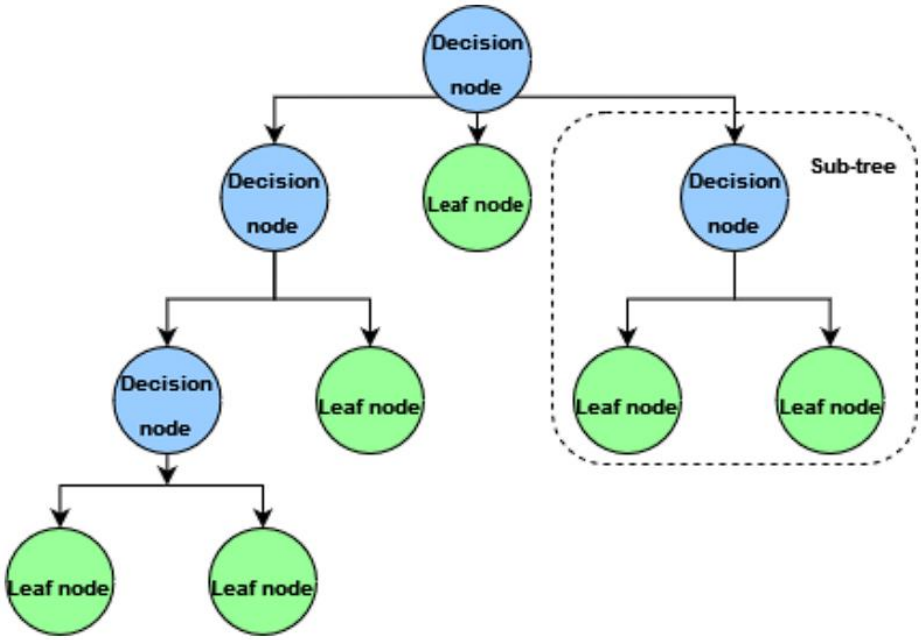


Рисунок 2.10 – Дерево рішень

2.3.4 Класифікатор градієнтного бустингу

Класифікатори градієнтного бустингу – це набір алгоритмів машинного навчання, які інтегрують низку моделей слабкого навчання для створення потужної прогностичної моделі, що використовує стрес рішень. Процеси реалізації градієнтного бустингу такі:

- Спочатку нам потрібно підібрати модель;
- Потім нам потрібно налаштувати параметри моделі та гіперпараметри;
- Далі нам потрібно зробити прогнози;
- Зрештою, на основі цих прогнозів ми можемо інтерпретувати

результати.

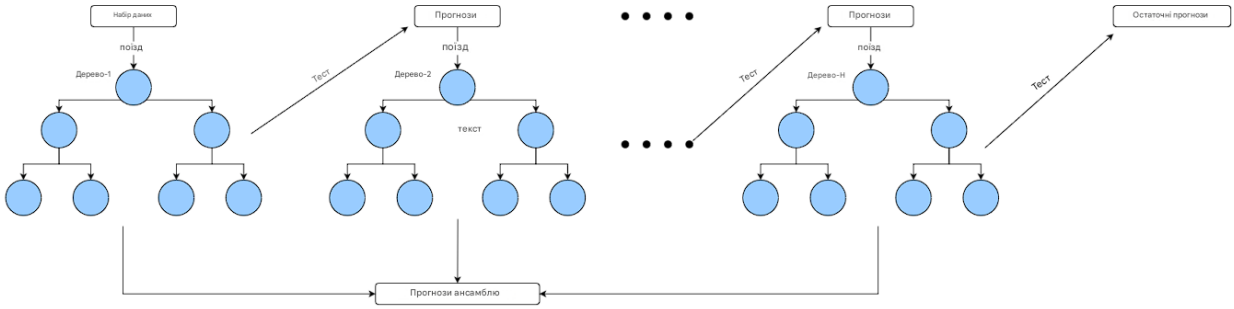


Рисунок 2.11 – Класифікатор посилення градієнта

2.3.5 Випадковий ліс

Випадкові ліси виконують свою роботу з навчання великої кількості дерев рішень. Виходом випадкового лісу є клас, обраний більшістю дерев. Випадкові ліси рішень вирішують проблему надмірного налаштування дерев рішень їх навчального набору. Як результат, у більшості випадків вони дають кращі результати, ніж дерева рішень. Однак вони дають менш точні результати, ніж дерева з градієнтним покращенням.

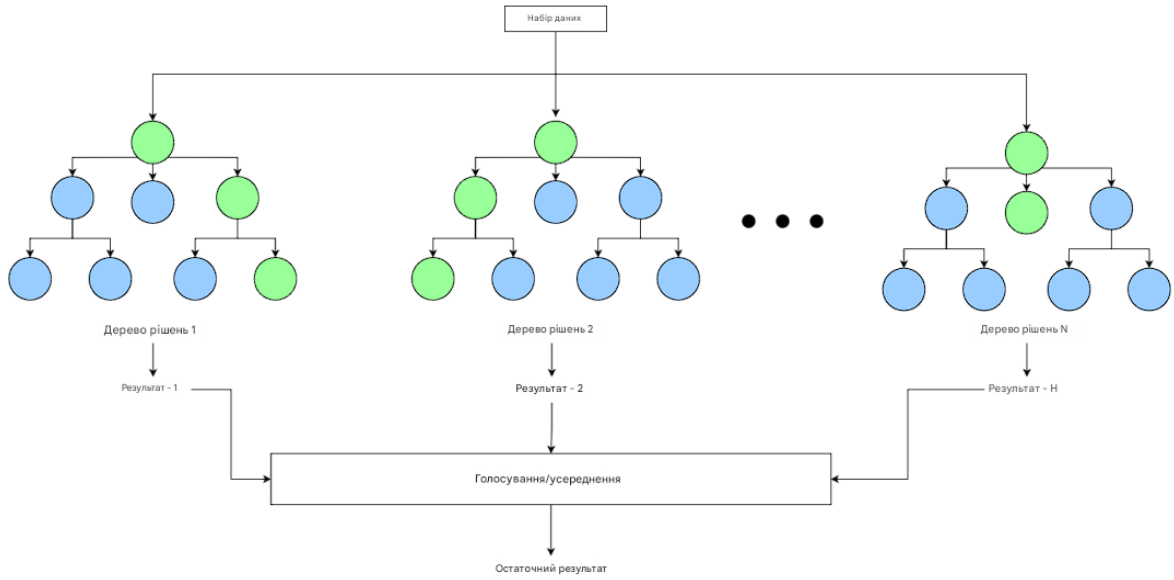


Рисунок 2.12 – Випадковий ліс

2.3.6 Наївний баєсівський закон

Це метод класифікації, заснований на теоремі Байєса та припущенні незалежності предиктора. Простіше кажучи, наївний байєсівський класифікатор припускає, що ймовірність наявності однієї ознаки в класі не залежить від ймовірності наявності будь-якої іншої ознаки.

2.3.7 Логістична регресія

Логістична регресія прогнозує вихідні дані та класифікує їх. Тому результат має бути категоріальним або дискретним значенням. Наприклад, Так чи Ні, 0 чи 1, Правда чи Хибність тощо. Ми підбираємо S-подібну логістичну функцію, яка прогнозуватиме бажаний вихідний результат як два максимальні значення від 0 до 1 у логістичній регресії.

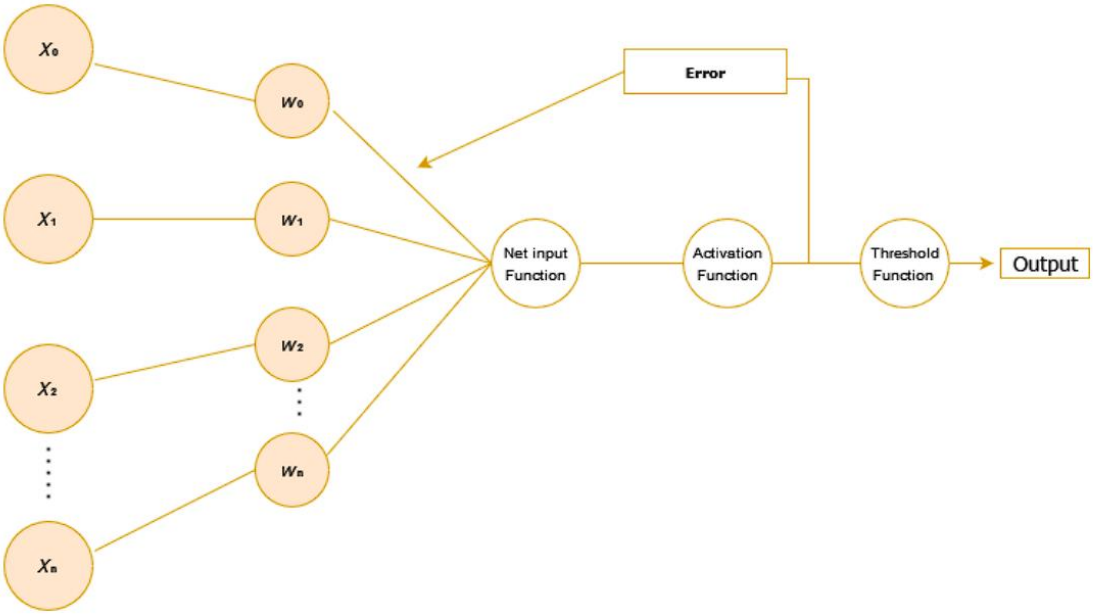


Рисунок 2.13 – Логістичний регресійний класифікатор

2.3.8 Класифікатор методом опорних векторів

SVM – це алгоритм класифікації, який використовується в регресії, класифікації та ідентифікації контурів [26]. Фундаментальна мета SVM полягає в розділенні n-вимірного простору на класи за допомогою лінії або межі рішення, щоб дані можна було легко класифікувати у відповідний клас. Ця межа рішення також називається гіперплощиною [25].

Методи опорних векторів мають перевагу в тому, що вони корисні для високих розмірностей. Крім того, вони корисні після того, як кількість розмірностей перевищує кількість ознак, оскільки межа рішення побудована з підмножини точок навчання.

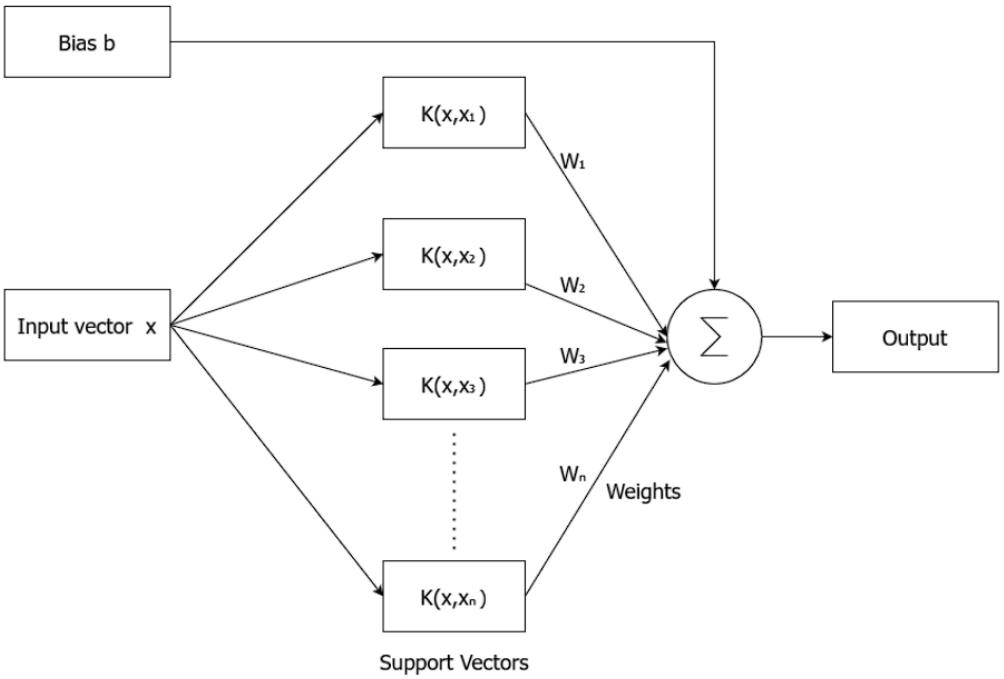


Рисунок 2.14 – Машина опорного вектора (SVM)

Наприклад, метод SVM можна використовувати для категоризації чоловічих та жіночих голосів. Для початку ми повинні навчити нашу модель різноманітним чоловічим та жіночим голосам, щоб наша SVM могла навчитися розрізняти чоловічий та жіночий голоси на основі їхніх характеристик. Таким чином, оскільки опорний вектор встановлює межу

рішення між цими двома наборами даних (чоловічий та жіночий) та вибирає екстремальні випадки, він спостерігатиме екстремальні випадки жінок та чоловіків.

2.3.9 K-Найближчий сусід

У алгоритмі K-найближчих сусідів символ «K» позначає найближчих сусідів нової невідомої змінної, яку необхідно передбачити або класифікувати. Наприклад, іноді ми маємо багато спільних характеристик з нашими найближчими колегами. Як результат, ми схильні формувати дружні стосунки з тими, з ким маємо більше спільних інтересів. Такий самий підхід використовується в алгоритмі KNN.

Його мета — знайти всіх найближчих сусідів нової невідомої точки даних, щоб визначити, до якого класу вона належить. Це метод, заснований на відстані. Щоб правильно класифікувати результати, ми повинні спочатку визначити значення K, яке завжди має бути непарним числом. Оскільки K встановлено на парне, можлива ситуація, коли компоненти обох груп рівні. Бажано вибрати непарне значення для K, оскільки така ситуація рівності між двома класами ніколи не виникне в цьому випадку. Враховуючи, що одна з двох груп все ще буде в більшості, вибирається непарне значення для K. [35] Вплив вибору меншого або більш значущого значення K на модель:

- Збільшення значення K: Коли значення k збільшується, виникає випадок недостатнього налаштування. У цій ситуації модель не може правильно навчатися на навчальних даних;

- Менше значення K: коли значення k менше, розвиваються умови переобладнання. Модель буде фіксувати всі навчальні дані, включаючи шум. За таких обставин ця модель буде погано працювати з тестовими даними.

2.3.10 Гаусова модель суміші

Гаусові моделі суміші – це ефективні моделі кластеризації для класифікації аудіо на основі їхньої здатності розрізняти звуки. Наприклад, для розрізнення різних інструментів у пісні або відділення голосу мовця від фонового шуму, коли він розмовляє зі своїм голосовим асистентом. Гаусова модель суміші – це розподіл ймовірностей.

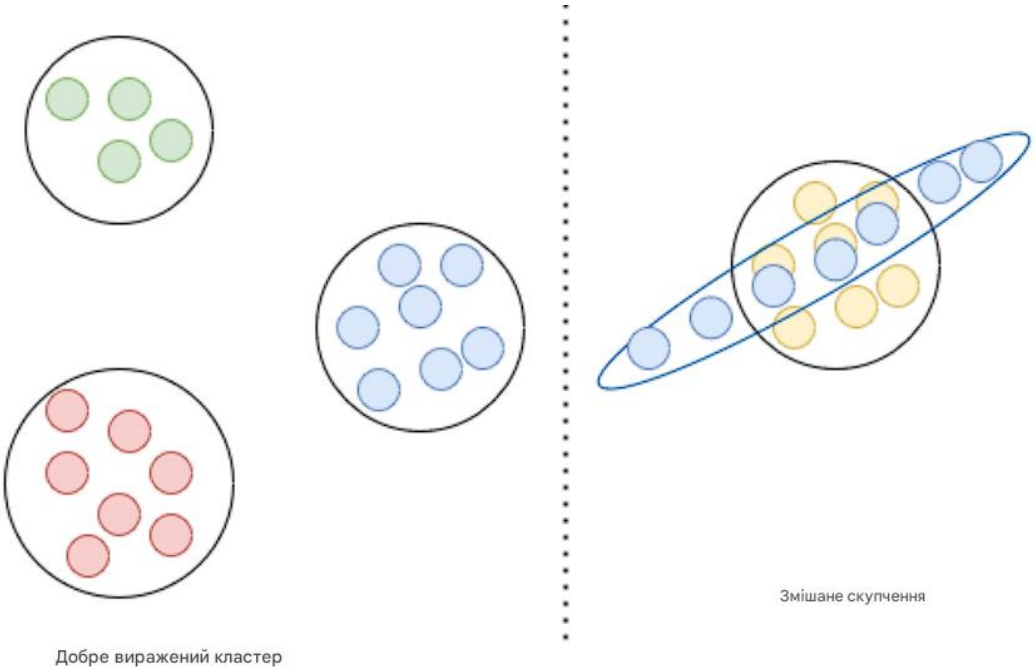


Рисунок 2.15 – Модель суміші Гауса

Однак на рисунку вище перетинаються дві окремі групи кластерів. Таким чином, є три кластери: один із синього, один із жовтого та ще один із синього та жовтого. Отже, GMM розділив набір даних на окремі кластери з перевагою в тому, що немає потреби вказувати зв'язок між точками даних, що містяться в кластерах. Оскільки ця модель здатна класифікувати точки даних у кластери, а потім застосовувати навчання, ці кластери створюються. Обчислюючи логарифмічну ймовірність голосового вектора та порівнюючи її з раніше зібраними даними, модель гаусової суміші може розпізнати це мовлення. Якщо логарифмічна ймовірність голосового вектора відповідає записаним даним, можна ідентифікувати окрему особу [27].

3 РЕАЛІЗАЦІЯ ТА РЕЗУЛЬТАТ

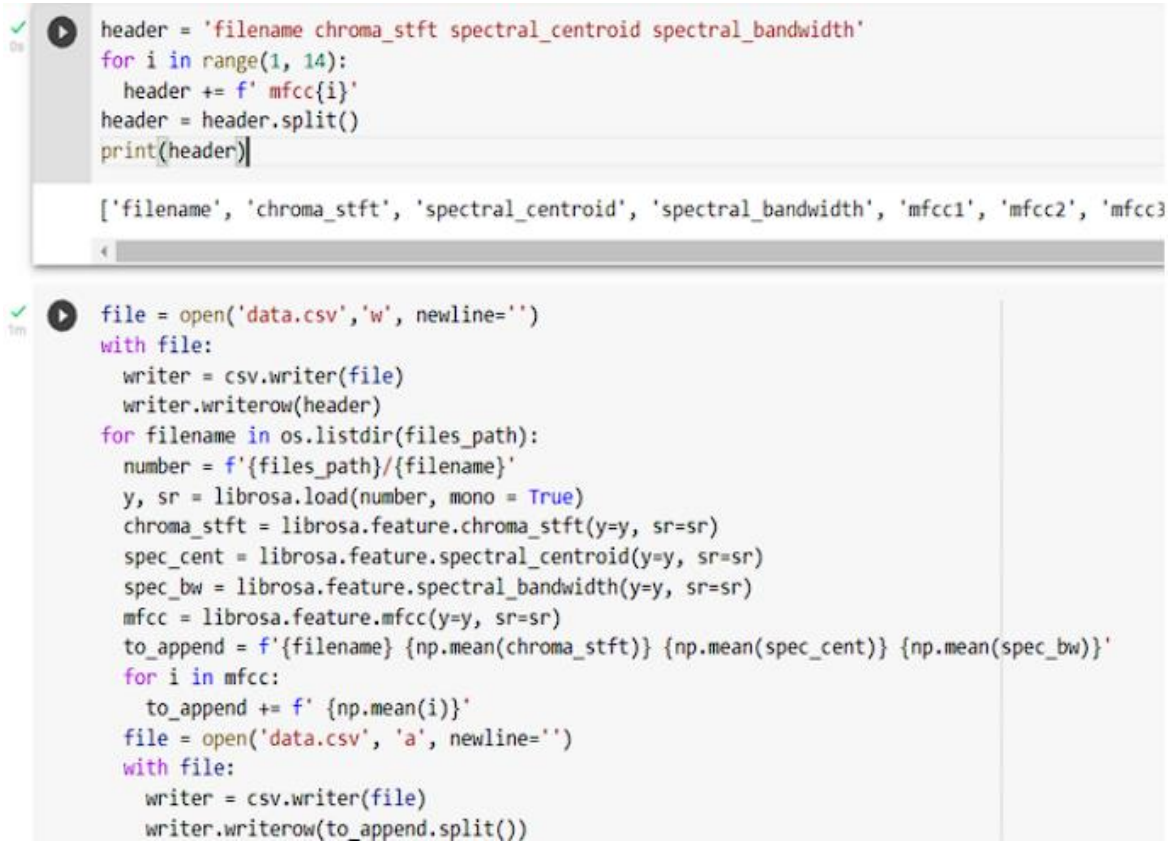
3.1 Реалізація

У цьому розділі описується реалізація запропонованої моделі ідентифікації мовця на основі класифікації за статтю. Перший крок – визначення статі мовця за допомогою алгоритму з найкращим результатом. Алгоритм з найкращим результатом вибирається після реалізації кількох алгоритмів, включаючи MLP, DT, GBM, наївний байєсівський алгоритм, KNN, LR та SVM. Другий крок – застосування алгоритмів зіставлення ознак, включаючи GMM та DTW, для ідентифікації осіб та порівняння між набором даних зі змішаною статтю та набором даних, вже розділеним за статтю, на основі результату найкращого алгоритму на кроці 1. Всі алгоритми були реалізовані та протестовані в Google Colab. По-перше, процес першого кроку містить чотири етапи: створення набору даних, попередня обробка вхідних даних, класифікація та тестування. По-друге, ідентифікація мовця виконується на окремих базах даних для чоловіків та жінок. Спочатку набір даних створюється з вилученими ознаками зразків даних. Після цього вхідні дані попередньо обробляються, щоб зробити набір даних придатним для навчання. Потім алгоритми навчаються, а в кінці тестуються.

3.1.1 Створення набору даних

Спочатку було створено порожній набір даних (CSV-файл) з відповідними заголовками. Потім з кожного аудіосемплу було вилучено та записано в набір даних за допомогою Python у Google Collab 13 (стандартних) ознак MFCC разом з деякими іншими ознаками, такими як спектральний центроїд, спектральна смуга пропускання.

Крім того, ми використовували набір даних Kaggle, що містить раніше отримані ознаки MFCC на основі різних емоцій понад 85 тисяч чоловіків та жінок. Незважаючи на відсутність аудіозаписів, ознаки MFCC вже були отримані, а точніше, 58 на вибірку. Набір даних було відредаговано таким чином, щоб ознаки чоловіків та жінок знаходилися в одній базі даних, за ознаками чоловіків слідували ознаки самки.



```
header = 'filename chroma_stft spectral_centroid spectral_bandwidth'
for i in range(1, 14):
  header += f' mfcc{i}'
header = header.split()
print(header)
```

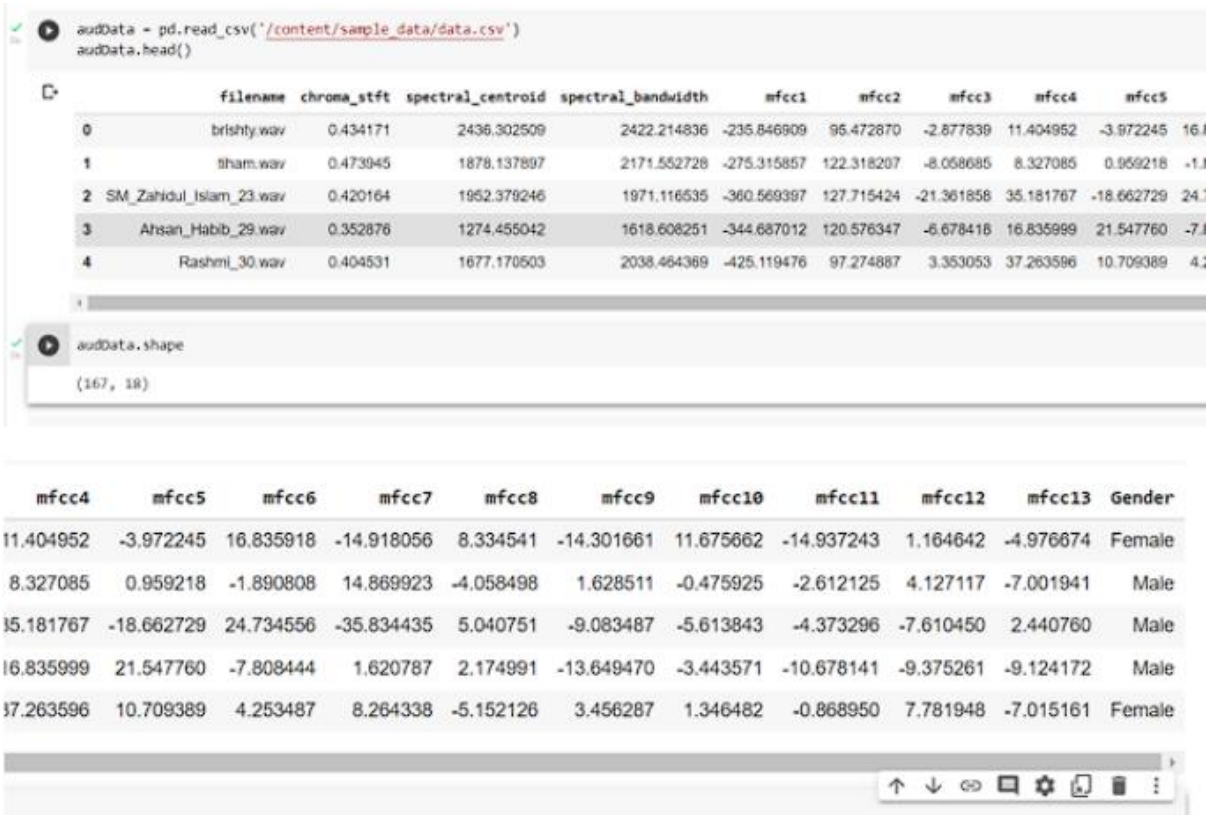
```
['filename', 'chroma_stft', 'spectral_centroid', 'spectral_bandwidth', 'mfcc1', 'mfcc2', 'mfcc3
```

```
file = open('data.csv','w', newline='')
with file:
  writer = csv.writer(file)
  writer.writerow(header)
for filename in os.listdir(files_path):
  number = f'{files_path}/{filename}'
  y, sr = librosa.load(number, mono = True)
  chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
  spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
  spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
  mfcc = librosa.feature.mfcc(y=y, sr=sr)
  to_append = f'{filename} {np.mean(chroma_stft)} {np.mean(spec_cent)} {np.mean(spec_bw)}'
  for i in mfcc:
    to_append += f' {np.mean(i)}'
  file = open('data.csv', 'a', newline='')
  with file:
    writer = csv.writer(file)
    writer.writerow(to_append.split())
```

Рисунок 3.1 – Створення набору даних

3.1.2 Попередня обробка вхідних даних

Після створення першого набору даних з даних, зібраних вручну від окремих осіб, можна завантажити файл CSV. Він буде схожий на дані, показані на рисунку 4.2. Перший стовпець містить назви файлів усіх зразків. 2-й, 3-й та 4-й стовпці містять кольоровість, спектральний центроїд та спектральну смугу пропускання, а решта стовпців показують значення MFCC кожного аудіозразка. З кожного зразка даних було вилучено 13 MFCC.



```
audData = pd.read_csv('/content/sample_data/data.csv')
audData.head()
```

	filename	chroma_stft	spectral_centroid	spectral_bandwidth	mfcc1	mfcc2	mfcc3	mfcc4	mfcc5
0	brishty.wav	0.434171	2436.302509	2422.214836	-235.846909	95.472870	-2.877839	11.404952	-3.972245
1	tiham.wav	0.473945	1878.137897	2171.552728	-275.315857	122.318207	-8.058685	8.327085	0.959218
2	SM_Zahidul_Islam_23.wav	0.420164	1952.379246	1971.116535	-360.569397	127.715424	-21.361858	35.181767	-18.662729
3	Ahsan_Habib_29.wav	0.352876	1274.455042	1618.608251	-344.687012	120.576347	-6.678418	16.835999	21.547760
4	Rashmi_30.wav	0.404531	1677.170503	2038.464369	-425.119476	97.274887	3.353053	37.263596	10.709389

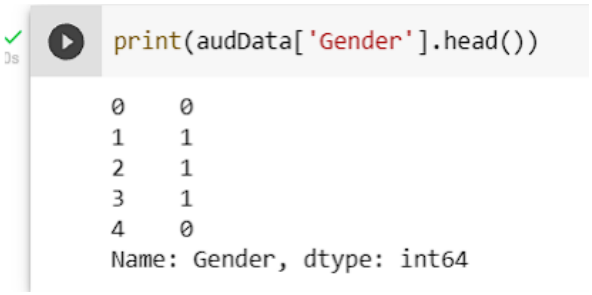
```
audData.shape
```

```
(167, 18)
```

mfcc4	mfcc5	mfcc6	mfcc7	mfcc8	mfcc9	mfcc10	mfcc11	mfcc12	mfcc13	Gender
11.404952	-3.972245	16.835918	-14.918056	8.334541	-14.301661	11.675662	-14.937243	1.164642	-4.976674	Female
8.327085	0.959218	-1.890808	14.869923	-4.058498	1.628511	-0.475925	-2.612125	4.127117	-7.001941	Male
35.181767	-18.662729	24.734556	-35.834435	5.040751	-9.083487	-5.613843	-4.373296	-7.610450	2.440760	Male
16.835999	21.547760	-7.808444	1.620787	2.174991	-13.649470	-3.443571	-10.678141	-9.375261	-9.124172	Male
37.263596	10.709389	4.253487	8.264338	-5.152126	3.456287	1.346482	-0.868950	7.781948	-7.015161	Female

Рисунок 3.2 – Зразок необроблених даних, отриманих із набору даних

Згодом можна побачити, що стовпець «стать» містить значення типу об'єкта. Для підвищення ефективності навчання MLP цей стовпець було закодовано за допомогою кодувальника міток. На рисунку 3.3 показано стовпець «Стать» після кодування, де «жіноча стать» позначено як 0, а «чоловіча стать» – як 1.



```
print(audData['Gender'].head())
```

```
0    0
1    1
2    1
3    1
4    0
Name: Gender, dtype: int64
```

Рисунок 3.3 – Стовпець статі після кодування

Далі, вибірка бази даних ознак MFCC від понад 85 000 чоловіків та жінок була завантажена в Google Colab. Не було потреби починати з нуля,

оскільки дані були зібрані з Kaggle, а потім скориговані, щоб включити нову функцію під назвою «Мітки» для статі. База даних містить 85 134 стовпці та 59 рядків, 58 з яких є ознаками MFCC та однією міткою статі. Базу даних можна побачити на рисунку 3.4.

```
audData = pd.read_csv('/content/sample_data/Book1.csv')
audData.head(85000)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	-280.558350	118.419098	8.925080	29.063490	-17.713118	0.083584	-24.045542	-7.715221	-16.647551	-4.126383	1.334840	-9.725342	-0.689985	-8.838691	3.989187	-3.732647
1	-138.495236	53.219675	19.017487	4.935156	-4.009660	-8.325881	-11.588456	-10.764145	-9.177705	-5.017083	-1.584400	-3.637140	-3.561886	-2.702257	-0.650454	-1.644242
2	-299.933685	118.578461	8.348220	25.962524	-16.300257	-1.788243	-24.209568	-9.280386	-15.846799	-4.361415	2.158371	-9.595265	-1.661132	-9.324622	3.650200	-3.472714
3	-280.343048	118.607193	9.081362	29.117228	-17.677935	0.041437	-24.069937	-7.778329	-16.729486	-4.217350	1.275757	-9.750621	-0.705592	-8.845049	3.975085	-3.773187
4	-292.556380	121.191017	7.017522	22.656330	-19.091734	-3.386593	-26.960073	-7.652481	-17.392126	-0.461076	-2.225410	-9.298290	-2.342303	-7.392892	4.782961	-8.195128
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
84995	-362.886971	81.488898	33.993745	34.702021	6.301329	15.126907	-4.521762	1.532309	-3.787924	-0.497366	3.451696	-7.006967	-0.416498	-5.996360	0.582298	-3.965252
84996	-448.685761	137.188675	-9.039816	71.149712	-25.533913	38.406754	-21.965014	10.740943	-8.306503	-0.151348	9.954624	-13.388996	8.747383	-14.550941	8.153438	-7.419524
84997	-414.068848	142.820358	-12.004197	74.040886	-26.474659	40.529549	-22.532198	11.818319	-8.970530	-0.597110	8.901114	-14.982880	8.361413	-14.919128	8.248922	-9.248197
84998	-442.578980	136.142456	-6.808254	71.033783	-26.833998	41.299500	-23.583801	16.115154	-15.767930	7.922126	0.060528	-10.073532	4.400312	-13.266366	10.407711	-13.913466
84999	-446.745819	138.349686	-10.085860	72.721306	-26.121521	39.278625	-24.700016	10.436246	-9.174704	-0.204039	8.120781	-16.279451	8.204673	-15.248693	8.497868	-9.349370

40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	labels
814257	1.013177	1.351648	2.321742	4.273898	1.879302	3.364791	6.083824	3.990373	3.634491	4.785914	6.822370	6.465144	5.040742	4.462775	4.019160	3.692048	3.415411	male
879782	0.571222	0.144832	1.522707	2.268737	1.940861	2.726537	3.284998	3.285994	3.138365	3.079358	4.527467	5.122600	4.041961	2.732871	2.176067	2.955910	2.600446	male
058828	1.536199	1.563370	3.132132	4.891072	2.284775	3.664436	6.388505	4.848680	4.132059	4.882883	6.965629	6.520880	5.417891	4.888678	3.816338	3.659878	3.295839	male
802715	1.024896	1.402462	2.385694	4.342613	1.930913	3.389000	6.062085	3.935874	3.591076	4.734492	6.781846	6.476598	5.069869	4.473549	4.050447	3.740353	3.456593	male
844623	2.384485	3.765201	5.084002	3.324567	4.281477	7.074114	4.198337	4.277469	4.758847	5.955528	6.064217	4.393935	4.429975	4.463602	4.980473	4.334596	4.759656	male
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
109653	-2.257150	-1.805712	-0.244642	-2.102228	-3.629453	-1.105910	0.867305	0.651796	-1.173131	-0.825280	-0.268169	-0.073765	-0.767470	0.132392	-0.132867	-0.369645	0.519820	female
159285	-1.762966	-2.103024	0.528610	-2.544487	-3.727237	-1.413039	2.287104	0.997904	-1.425704	-0.092397	-0.106547	-0.291907	-0.456864	0.845100	-0.707612	-0.937351	-0.064132	female
715624	-1.500478	-2.434717	0.418023	-2.689296	-3.637624	-1.680580	1.747597	0.844877	-1.608692	-0.756256	-0.105382	0.205703	-1.066054	0.506801	-0.146501	-0.369802	0.281866	female
338427	-2.525414	0.054952	-2.484588	-3.997716	-0.107430	1.950908	0.394009	-3.627412	-0.222841	0.954029	0.082254	-1.991180	-0.894762	-0.646592	0.168910	0.885705	1.421633	female
426824	-1.603953	-2.426892	1.132172	-2.330291	-3.643549	-1.068533	2.323688	0.046560	-2.808129	-1.202936	-0.153376	0.160757	-0.545690	1.332009	-0.042957	-0.704957	0.154987	female

Рисунок 3.4 – Зразок необроблених даних, отриманих із набору даних Kaggle

4	55	56	57	labels
5	4.019160	3.692048	3.415411	1
1	2.176067	2.955910	2.600446	1
8	3.816338	3.659878	3.295839	1
9	4.050447	3.740353	3.456593	1
2	4.980473	4.334596	4.759656	1
...	...	...	...	...
2	-0.132867	-0.369645	0.519820	0
0	-0.707612	-0.937351	-0.064132	0
1	-0.146501	-0.369802	0.281866	0
2	0.168910	0.885705	1.421633	0
9	-0.042957	-0.704957	0.154987	0

Рисунок 3.5 – Стовпець міток після кодування набору даних Kaglle

Аналогічно, як і попередній набір даних, функція labels містить значення типів об'єктів. Отже, цей стовпець було закодовано за допомогою LabelEncoder. На рисунку 3.5 представлено набір даних після кодування стовпця «labels». Після кодування «male» було присвоєно значення 1, а «female» – значення 0.

3.1.3 Класифікація за допомогою 9 класифікаторів

Для навчання алгоритму MLP дані було розділено на навчальні та тестові дані з коефіцієнтом 0,25. Наша мета тут — визначити правильну стать мовця. Тому ознака «Стать» є нашою цільовою міткою. Для кращої точності масштабування було реалізовано за допомогою скалера minmax з бібліотеки sklearn. У розділі результатів будуть порівняні різниці між результатами, отриманими з масштабуванням та без нього. Алгоритм MLP було реалізовано на обох наборах даних. Аналогічно, RBFN, дерево рішень, випадковий ліс, градієнтне бустування, наївний байєсівський алгоритм, KNN, логістична регресія та метод опорних векторів були реалізовані на обох наборах даних.

```
y = audData['Gender']
x = audData.drop(columns='Gender')
x = audData.drop(columns='filename')

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, stratify = y, test_size = 0.25, random_state=1)
print("Training set: x->{} , y->{} \n Testing set: x->{} , y->{}".format(x_train.shape, y_train.shape, x_test.shape, y_test.shape))
```

```
Training set: x->(125, 17) , y->(125,)
 Testing set: x->(42, 17) , y->(42,)
```

Рисунок 3.6 – Розподіл даних на тренувальні та тестові дані

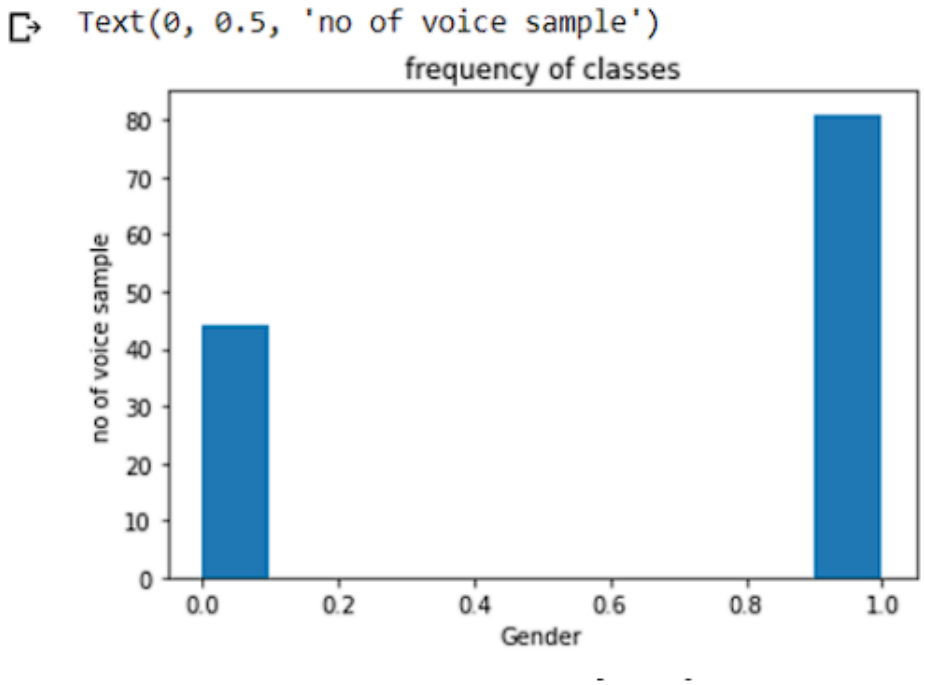


Рисунок 3.7 – Діаграма цільових даних

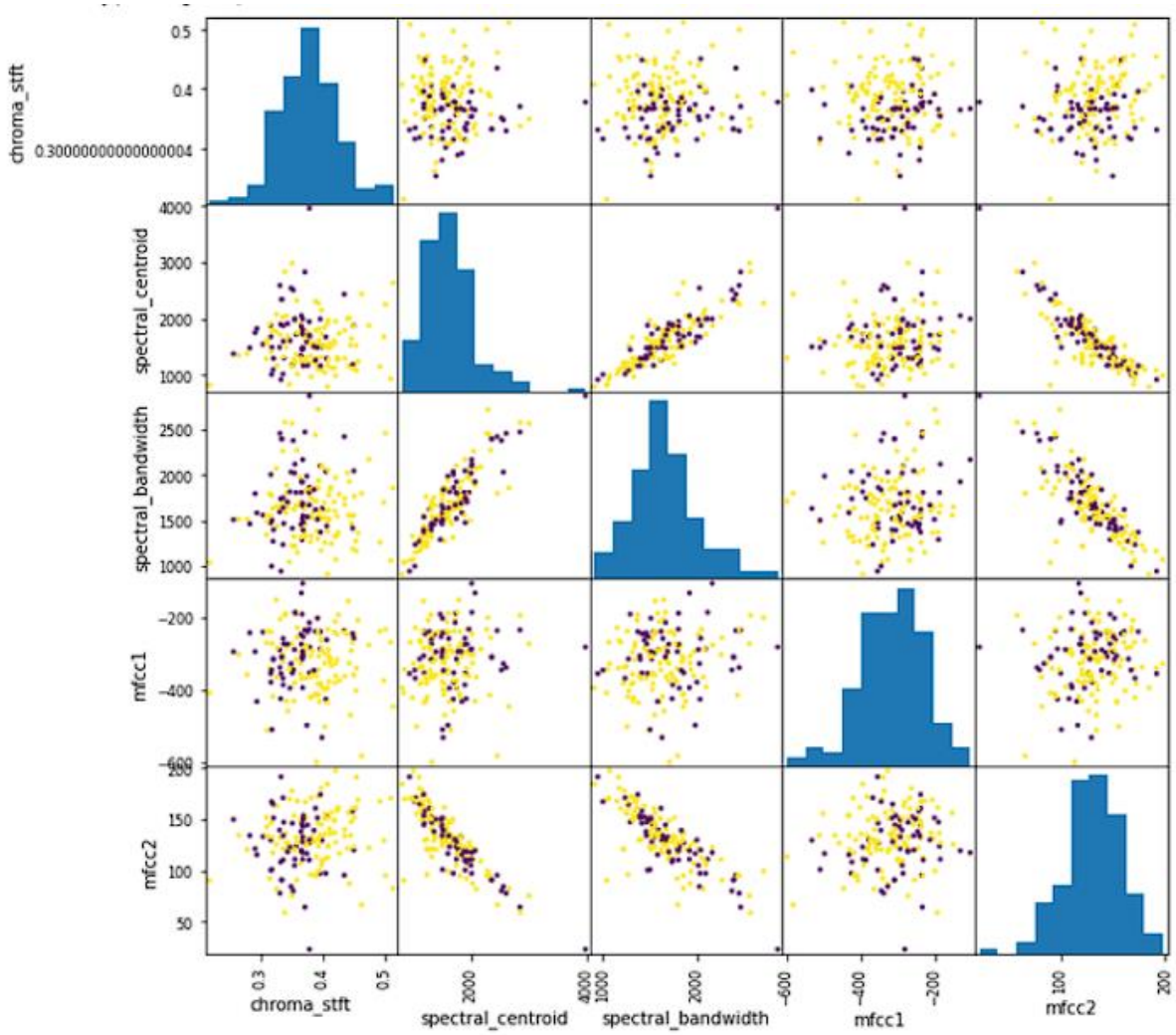


Рисунок 3.8 – Матриця розкиду

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(x_train)
```

```
MinMaxScaler(copy=True, feature_range=(0, 1))
```

```
[18] x_train_scaled = scaler.transform(x_train)
```

Рисунок 3.9 – Масштабування MinMax

```
[37] from sklearn.neural_network import MLPClassifier
```

```
mlp = MLPClassifier(hidden_layer_sizes=(3,2),max_iter=500, activation='logistic')
mlp
```

```
MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
              beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(3, 2), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=500,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

```
mlp.fit(x_train_scaled, y_train)
```

```
MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
              beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(3, 2), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=500,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

Рисунок 3.10 – Навчання MLP

Аналогічно, та сама процедура була виконана для бази даних Kaggle, як видно на рисунках 3.11, 3.12 та 3.13. Класифікатор MLP був навчений аналогічно попередній базі даних.

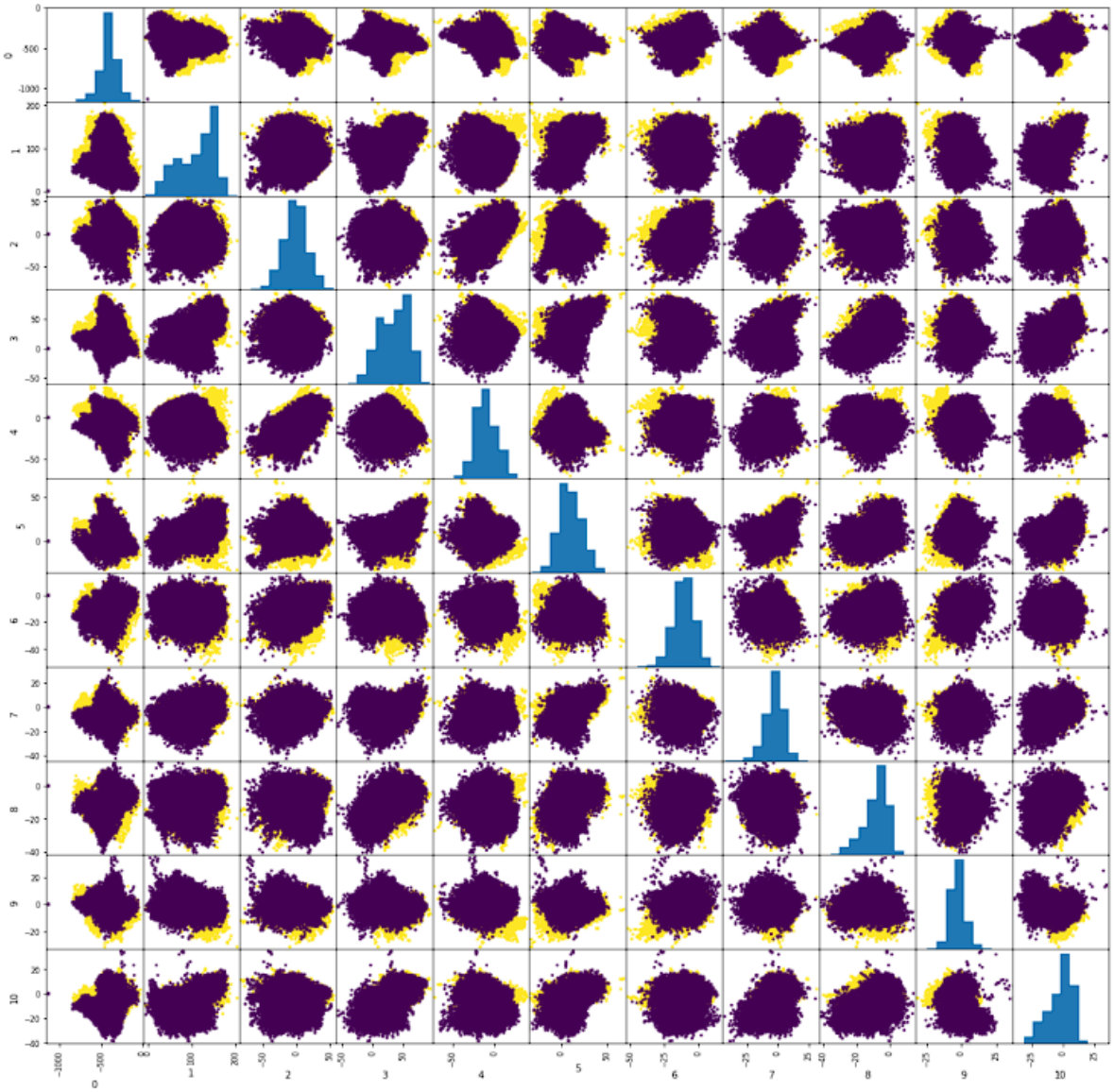


Рисунок 3.11 – Матриця розкиду для набору даних Kaggle

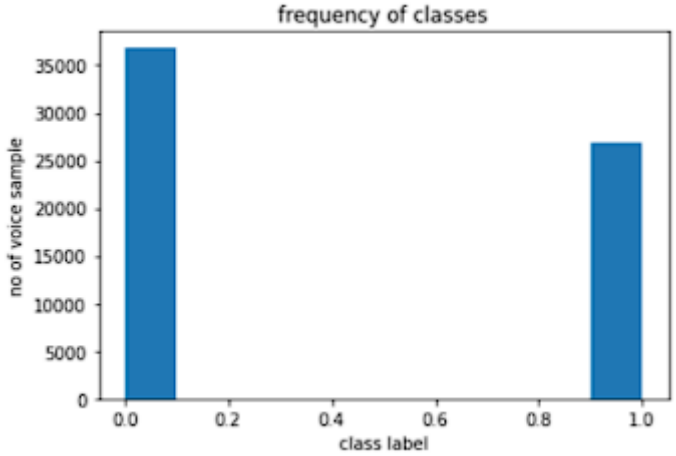


Рисунок 3.12 – Діаграма цільових даних для набору даних Kaggle

```
from sklearn.neural_network import MLPClassifier
```

```
mlp = MLPClassifier(hidden_layer_sizes=(3,2),max_iter=500, activation='relu')
mlp
```

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(3, 2), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=500,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

Рисунок 3.13 – Навчання MLP для набору даних Kaggle

Аналогічно, обидва набори даних були піддані RBFN, дереву рішень, випадковому лісу, градієнтному бустінгу, наївному байєсівському методу, KNN, логістичній регресії та методу опорних векторів.

3.1.4 Ідентифікація мовця за допомогою GM

На основі результату класифікації за статтю, отриманого з найкращого результату кількох класифікаторів, було прийнято вважати, що дані розділені за статтю, а для ідентифікації мовця було використано алгоритм розпізнавання образів GMM. Алгоритм було використано як на об'єднаному наборі даних-1, так і на тому самому наборі даних, розділеному за статтю. На наступних 3 рисунках описано реалізацію GMM.

```python
import glob
import librosa
import numpy as np
import os
import sklearn.mixture
import sys

def load(audio_path):
    y, sr = librosa.load(audio_path)
    y_trim = librosa.effects.remix(y, intervals=librosa.effects.split(y))
    mfcc = librosa.feature.mfcc(y=y_trim, sr=sr)
    return mfcc.T

def fit(frames, test_ratio=0.05, n_components=16):
    index = np.arange(len(frames))
    np.random.shuffle(index)

    train_idx = index[int(len(index) * test_ratio):]
    test_idx = index[:int(len(index) * test_ratio)]

    gmm = sklearn.mixture.GaussianMixture(n_components=n_components)
    gmm.fit(frames[train_idx])

    return gmm, frames[test_idx]

def predict(gmms, test_frame):
    scores = []
    for gmm_name, gmm in gmms.items():
        scores.append((gmm_name, gmm.score(test_frame)))
    return sorted(scores, key=lambda x: x[1], reverse=True)
```

Рисунок 3.14 – Реалізація GMM для Dataset1

```python
def evaluate(gmms, test_frames):
    correct = 0

    for name in test_frames:
        best_name, best_score = predict(gmms, test_frames[name])[0]
        print ('Ground Truth: %s, Predicted: %s, Score: %f' % (name, best_name, best_score))
        if name == best_name:
            correct += 1
    print (correct)
    print ('Overall Accuracy: %f%%' % (float(correct) / len(test_frames)))

if __name__ == '__main__':
    gmms, test_frames = {}, {}

    for filename in glob.glob(os.path.join("sample_data/data", '*.wav')):
        name = os.path.splitext(os.path.basename(filename))[0]
        print ('Processing %s ...' % name)
        gmms[name], test_frames[name] = fit(load(filename))

    evaluate(gmms, test_frames)
```

Рисунок 3.15 – Реалізація GMM для Dataset1

```
if __name__ == '__main__':
    gmms, test_frames = {}, {}

    for filename in glob.glob(os.path.join("sample_data/data", '*.wav')):
        name = os.path.splitext(os.path.basename(filename))[0]
        print ('Processing %s ...' % name)
        gmms[name], test_frames[name] = fit(load(filename))

    evaluate(gmms, test_frames)

    for filename in glob.glob(os.path.join("sample_data/male", '*.wav')):
        name = os.path.splitext(os.path.basename(filename))[0]
        print ('Processing %s ...' % name)
        gmms[name], test_frames[name] = fit(load(filename))

    evaluate(gmms, test_frames)

    for filename in glob.glob(os.path.join("sample_data/female", '*.wav')):
        name = os.path.splitext(os.path.basename(filename))[0]
        print ('Processing %s ...' % name)
        gmms[name], test_frames[name] = fit(load(filename))

    evaluate(gmms, test_frames)

    for filename in glob.glob(os.path.join("sample_data", '*.wav')):
        result = predict(gmms, load(filename))
        print ('%s: %s' % (os.path.basename(filename), ' / '.join(map(lambda x: '%s = %f' % x, result[:5]))))
```

Рисунок 3.16 – Реалізація GMM для Dataset1

3.2 Результати

У цьому розділі наведено результати реалізації алгоритмів класифікатора та ідентифікації. Після запуску моделі за допомогою Colab отримано результати класифікації даних. Перш за все, з результатів, отриманих з набору даних 1, видно, що MLP показав певну задовільність, тоді як RBFN, SVM, логістична регресія показали погані результати, отримавши точність тестування нижче 40 відсотків. KNN також показав погані результати. Можна сказати, що RBFN не підходить для порівняно менших наборів даних. Однак, вражаюча продуктивність була отримана за допомогою дерева рішень, випадкового лісу, градієнтного підсилення та наївного баєсівського класифікатора. Кожен із класифікаторів показав 100-відсоткову точність як на навчальних даних, так і на тестових даних. Набір даних 1 складається зі 167 осіб.

Розглядаючи шляхи підвищення точності, було проведено аналіз попередньої обробки. Для масштабування даних використовувався скалер

Min-Max, але оскільки обидва набори даних містять переважно аудіоелементи, MFCC, витягнуті з аудіосемплів, відмова від масштабування під час попередньої обробки може бути корисною. Після запуску тих самих класифікаторів на немасштабованих даних з набору даних-1, результат був кращим, ніж раніше, як видно з таблиці 4.1. RBFN не зміг дати набагато кращий результат порівняно з результатом, отриманим з масштабованими даними, але логістична регресія подвоїла свою точність, а KNN також покращила свою ефективність. Хоча MLP не показав кращих результатів у цьому випадку, SVM показав приголомшливі 100-відсоткову точність з немасштабованими даними, тоді як у випадку масштабованих даних вона була менше 50%.

Таблиця 3.1 – Аналіз розпізнавання статі та порівняння між 9 класифікаторами в наборі даних

Алгоритми	З масштабуванням (точність)		Без масштабування (точність)	
	Навчання	Тестування	Навчання	Тестування
1	2	3	4	5
Багатошаровий персептрон	0.65	0.64	0.65	0.64
Радіальна базисна функція	0.35	0.36	1.00	0.38
Дерево рішень	1.00	1.00	1.00	1.00
Випадковий ліс	1.00	1.00	1.00	1.00
Посилення градієнта	1.00	1.00	1.00	1.00

Продовження таблиці 3.1

1	2	3	4	5
Наївний Байєс	1.00	1.00	1.00	1.00
K-Найближчий сусід	0.39	0.52	0.81	0.69
Логістична регресія	0.35	0.36	0.82	0.62
Підтримуюча векторна машина	0.35	0.36	1.00	1.00

У випадку набору даних Kaggle для визначення статі було використано 7 класифікаторів, серед яких жоден алгоритм не вразив точністю. Однак після навчання на немасштабованих даних, MLP показав точність понад 90 відсотків, Decision Tree та Logistic Regression перевищили 80 відсотків, тоді як Naive Bayes також покращився. Помітну продуктивність показали Random Forest, Gradient Boosting та KNN, які отримали точність 98 відсотків, 98 відсотків та 96 відсотків на тестових даних відповідно.

Таблиця 3.2 - Аналіз розпізнавання статі та порівняння між 7 класифікаторами в наборі даних Kaggle

Алгоритми	З масштабуванням (точність)		Без масштабування (точність)	
	Навчання	Тестування	Навчання	Тестування
Багатошаровий персептрон	0.63	0.63	0.92	0.91

Дерево рішень	0.52	0.52	0.80	0.80
Випадковий ліс	1.00	0.67	1.00	0.98
Посилення градієнта	1.00	0.67	1.00	0.98
Наївний Байєс	0.42	0.42	0.67	0.67
K-Найближчий сусід	0.60	0.60	0.99	0.96
Логістична регресія	0.58	0.58	0.89	0.89

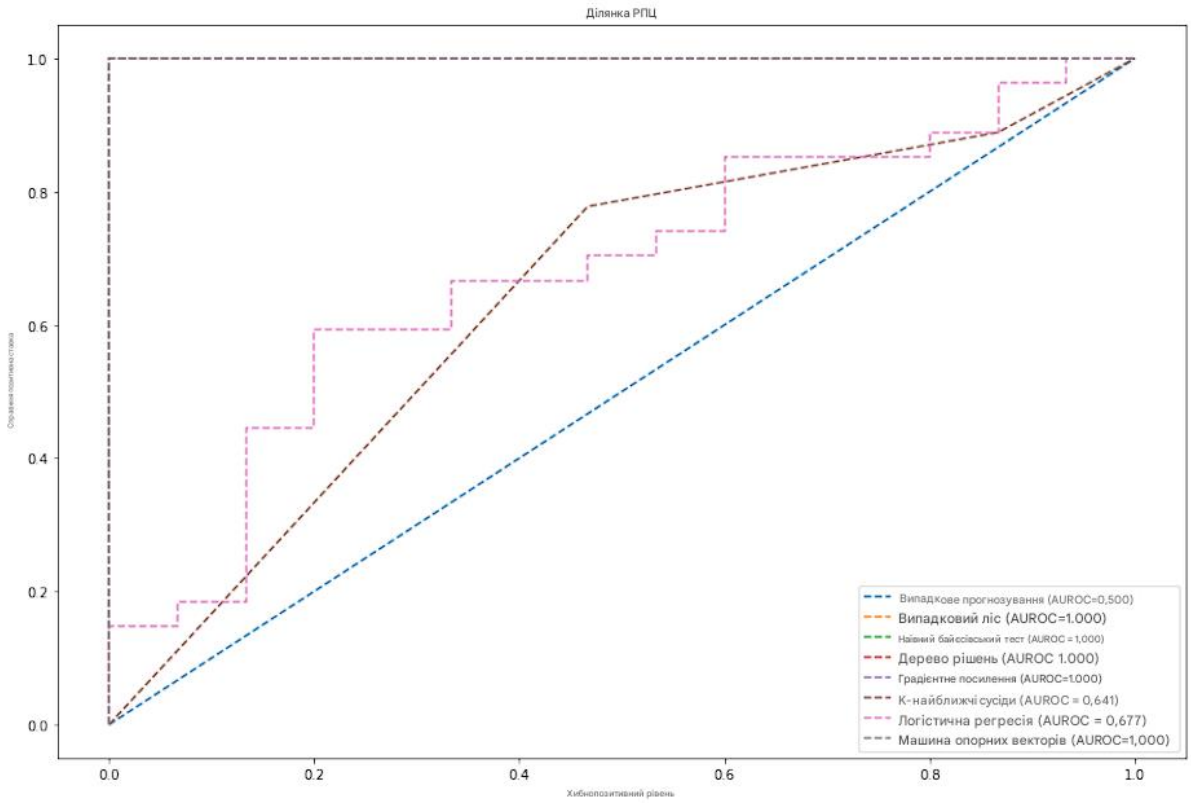


Рисунок 3.17 – Графік ROC для набору даних1 (без масштабування)

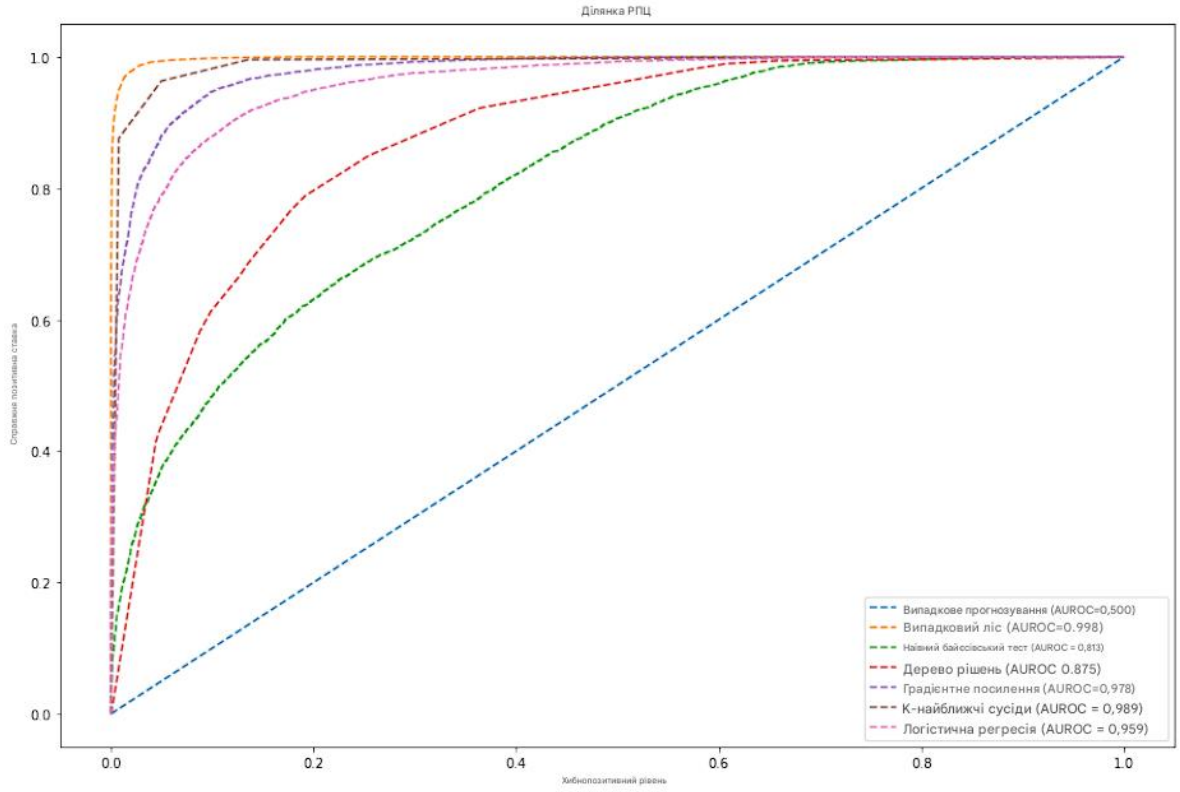


Рисунок 3.18 – Графік ROC для набору даних Kaggle (без масштабування)

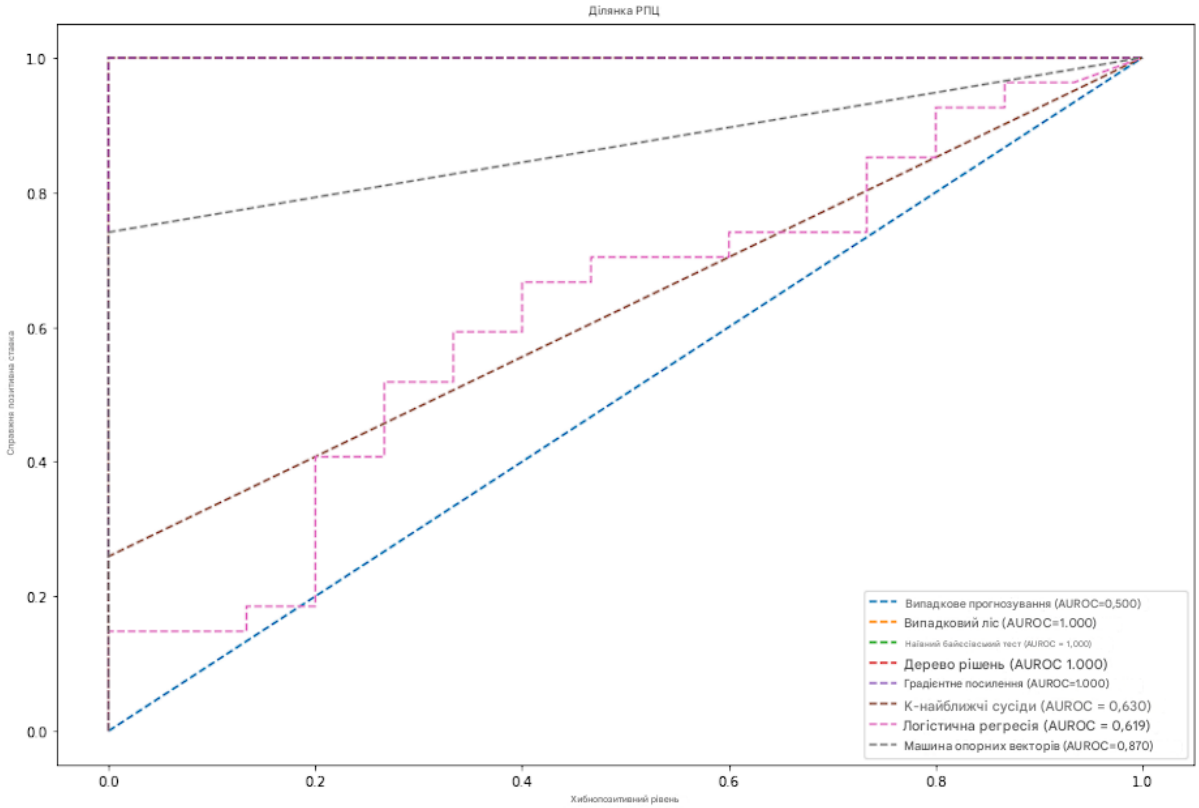


Рисунок 3.19 – Діаграма ROC для набору даних1 (з масштабуванням)

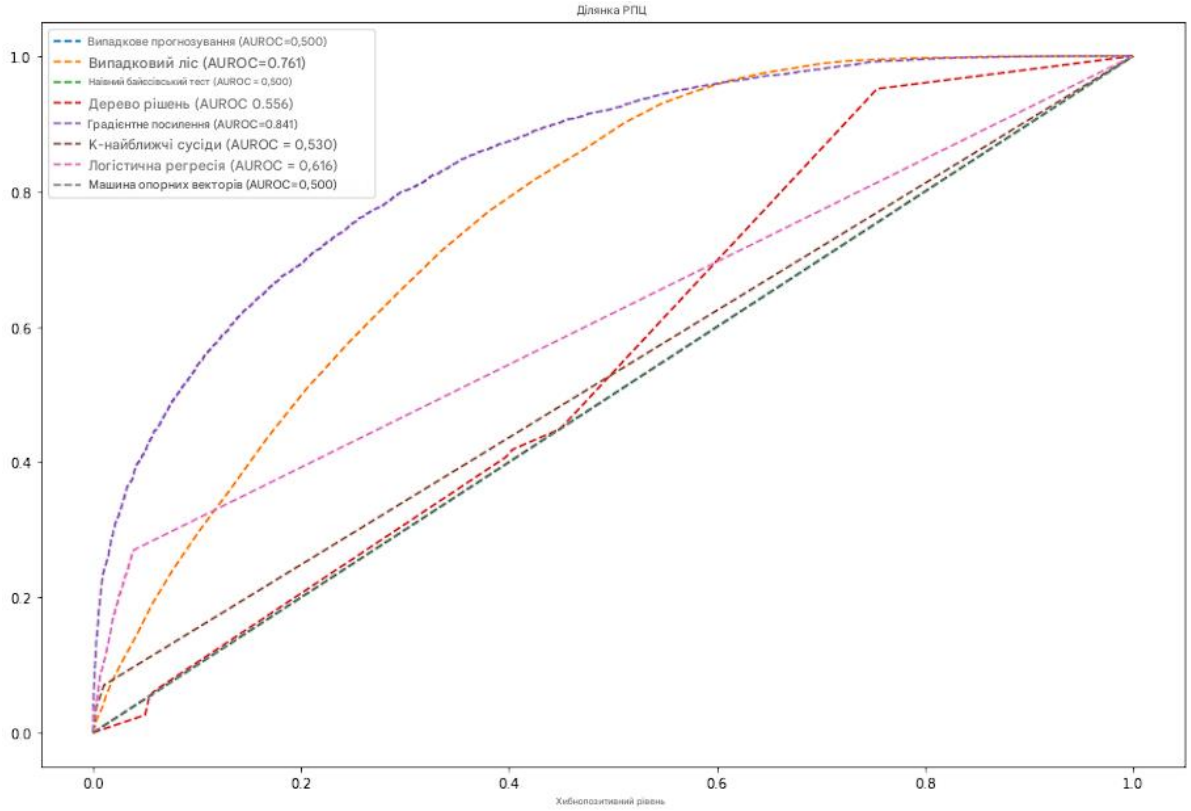


Рисунок 3.20 – Графік ROC для набору даних Kaggle (з масштабуванням)

ВИСНОВКИ

Система ідентифікації особи за допомогою розпізнавання голосу на основі попередньо сформованої бази даних може бути ефективним рішенням для підвищення рівня безпеки. У порівнянні з іншими біометричними технологіями (такими як сканування відбитків пальців чи сітківки ока), розпізнавання голосу має перевагу завдяки зручності збору зразків, особливо в умовах, коли інші біометричні ознаки недоступні.

Досягнута висока точність у визначенні статі мовця дозволяє оптимізувати ідентифікацію особи шляхом обмеження пошуку до відповідного підрозділу бази даних. Під час реалізації було використано декілька класифікаторів, серед яких деякі продемонстрували значну ефективність. Це відкриває можливості для застосування такої системи у масштабних сценаріях, включаючи корпоративні або національні рішення у сфері безпеки.

Подальше вдосконалення можливо шляхом інтеграції більш потужних алгоритмів зіставлення зразків, масштабування системи для обробки великих обсягів даних, а також реалізації гнучкої логіки пошуку в базі даних у разі невизначеності під час класифікації статі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. S. Agrawal, A. Shruti, and C. R. Krishna, “Prosodic feature based text dependent speaker recognition using machine learning algorithms,” International Journal of Engineering Science and Technology, vol. 2, no. 10, pp. 5150–5157, 2010.

2. L. Muda, M. Begam, and I. Elamvazuthi, “Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques,” arXiv preprint arXiv:1003.4083, 2010.

3. J. W. Shin, J.-H. Chang, and N. S. Kim, “Voice activity detection based on statistical models and machine learning approaches,” Computer Speech & Language, vol. 24, no. 3, pp. 515–530, 2010.

4. J. XinXing and S. Xu, “Speech recognition based on efficient dtw algorithm and its dsp implementation,” Procedia Engineering, vol. 29, pp. 832–836, 2012.

5. A. M. Moselhy and A. A. Abdelnaiem, “Lpc and mfcc performance evaluation with artificial neural network for spoken language identification,” International Journal of Signal Processing, Image Processing and Pattern Recognition, vol. 6, no. 3, p. 55, 2013.

6. A. Halageri, A. Bidappa, C. Arjun, M. M. Sarathy, and S. Sultana, “Speech recognition using deep learning,” Int. J. Comput. Sci. Inf. Technol, vol. 6, no. 3, pp. 3206–3209, 2015.

7. J. Padmanabhan and M. J. Johnson Premkumar, “Machine learning in automatic speech recognition: A survey,” IETE Technical Review, vol. 32, no. 4, pp. 240–251, 2015.

8. M. V. K. Kale, P. D. Deshmukh, and M. H. R. Gite, “Voice based biometric system feature extraction using mfcc and lpc technique,” International Journal of Advanced Engineering Research and Science, vol. 3, no. 5, p. 236 709, 2016.

9. B. Soewito, F. L. Gaol, E. Simanjuntak, and F. E. Gunawan, “Smart mobile attendance system using voice recognition and fingerprint on smartphone,” in 2016 International Seminar on Intelligent Technology and Its Applications (ISITIA), IEEE, 2016, pp. 175–180.

10. S. Kinkiri, W. J. Melis, and S. Keates, “Machine learning for voice recognition,” 2017.

11. P. Gupta, S. Goel, and A. Purwar, “A stacked technique for gender recognition through voice,” in 2018 Eleventh International Conference on Contemporary Computing (IC3), IEEE, 2018, pp. 1–3.

12. N. S. Ibrahim and D. A. Ramli, “I-vector extraction for speaker recognition based on dimensionality reduction,” Procedia Computer Science, vol. 126, pp. 1534–1540, 2018.

13. O. Novotnỳ, O. Plchot, P. Matejka, L. Mosner, and O. Glembek, “On the use of x-vectors for robust speaker recognition.,” in Odyssey, 2018, pp. 168–175.

14. D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, “Xvectors: Robust dnn embeddings for speaker recognition,” in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2018, pp. 5329–5333.

15. R. Thiruvengatanadhan, “Speech recognition using svm,” International Research Journal of Engineering and Technology (IRJET), vol. 5, no. 9, pp. 918–921, 2018.

16. L. Verde, G. De Pietro, and G. Sannino, “Voice disorder identification by using machine learning techniques,” IEEE access, vol. 6, pp. 16 246–16 255, 2018.

17. A. F. Fadlilah and E. C. Djamal, “Speaker and speech recognition using hierarchy support vector machine and backpropagation,” in 2019 6th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), IEEE, 2019, pp. 404–409.

18. F. Fang, X. Wang, J. Yamagishi, I. Echizen, M. Todisco, N. Evans, and J.-F. Bonastre, “Speaker anonymization using x-vector and neural waveform

models," arXiv preprint arXiv:1905.13561, 2019.

19. N. H. Tandel, H. B. Prajapati, and V. K. Dabhi, "Voice recognition and voice comparison using machine learning techniques: A survey," in 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), IEEE, 2020, pp. 459–465.

20. Noise, Dec. 2021. URL: https://en.wikipedia.org/wiki/Noise. (дата звернення 2.06.2025).

21. M. Orken, K. Aizat, A. Keylan, O. Dina, Z. Bagashar, and N. Bulbul, "Development of security systems using dnn and i & x-vector classifiers," Mamyrbayev, O., Kydyrbekova, A., Alimhan, K., Oralbekova, D., Zhumazhanov, B., & Nuranbayeva, B.(2021). Development of security systems using DNN and i & x-vector classifiers. Eastern-European Journal of Enterprise Technologies, vol. 4, no. 9, p. 112, 2021.

22. S. E. Tandogan and H. T. Sencar, "Estimating uniqueness of i-vector-based representation of human voice," IEEE Transactions on Information Forensics and Security, vol. 16, pp. 3054–3067, 2021.

23. A brief history of speech recognition. URL: https://sonix.ai/history-of-speech-recognition#:~:text=The%20first%20speech%20recognitio%20n%20 systems,to%2016%20words%20in%20English. (дата звернення 2.06.2025).

24. T. Kathiresan, A. Verma, and V. Dellwo, "Gender bias in voice recognition: An i-vector-based gender-specific automatic speaker recognition study,"

25. Machine learning. URL: https://www.javatpoint.com/machinelearning-support-vector-machine-algorith. (дата звернення 2.06.2025).

26. Scikit. URL: https://scikit-learn.org/stable/modules/svm.html. (дата звернення 2.06.2025).

27. Tahira, Speaker-identification-using-gmm-with-mfcc.URL: https: //www.researchgate.net/publication/274963749_Speaker_Identification_ Using_GMM_with_MFCC. (дата звернення 2.06.2025).

28. T. Tanaka, et al., A joint end-to-end and DNN-HMM hybrid automatic

speech recognition system with transferring sharable knowledge, in: Interspeech, 2019, 2210–2214. doi:10.21437/interspeech.2019-2263.

29. D. Wang, X. Wang, S. Lv, An overview of end-to-end automatic speech recognition, Symmetry, 11(8) (2019) 1–26. doi:10.3390/sym11081018.

30. E. McDermott, A deep generative acoustic model for compositional automatic speech recognition, in: 32nd Conference on Neural Information Processing Systems, Montreal, 2018, pp. 1–17.

31. S. Zhang, et al., Streaming chunk-aware multihead attention for online end-to-end speech recognition, 2020, pp. 1–5. arxiv:2006.01712. To appear.

32. S. Kim, et al., Improved training for online end-to-end speech recognition systems, in: Interspeech 2018, pp. 2913–2917. doi:10.21437/interspeech.2018-2517.

33. Y. Chen, Data techniques for online end-to-end speech recognition, 2020, pp. 1–5. arxiv:2001.09221. To appear.