Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут комп'ютерних наук
та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи

перший (бакалаврський)

(рівень вищої освіти)

на тему: **Розроблення інтелектуальної системи для класифікації музичних жанрів**

Виконав: студент 4 курсу, групи КН-41
спеціальності
122 – "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Лемех Т.М.
(прізвище та ініціали)

Керівник Пірко І.Б.
(прізвище та ініціали)

Рецензент Часковський О.Т.
(прізвище та ініціали)

Львів – 2025 р.

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій

Кафедра комп'ютерних наук

Рівень вищої освіти перший (бакалаврський)

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

ЗАТВЕРДЖУЮ
Завідувач кафедри КН

Борецька І.Б.
"10" червня 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Лемех Тарас Михайлович
(прізвище, ім'я, по батькові)

1. Тема роботи **Розроблення інтелектуальної системи для класифікації музичних жанрів**

керівник роботи Пірко І. Б, канд. фіз.-мат. наук, доцент.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від 15.11.2024 р. № С-882

2. Термін подання студентом роботи 10.06.2025 р.

3. Вихідні дані до роботи:

- вивчити предметну область, проаналізувати існуючі моделі класифікації музичних жанрів;
- розглянути і використати алгоритми, які лежать в основі математичної моделі класифікації музичних жанрів;
- спроектувати інтелектуальну систему з допомогою мови програмування Python та відповідних бібліотек машинного навчання та візуалізації результатів.
- представити результати роботи інформаційної системи.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Розділ 1. Стан проблемної області

Розділ 2. Інформаційне та математичне забезпечення

Розділ 3. Програмне та технічне забезпечення

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

додаток А, додаток Б

6. Дата видачі завдання 18 листопада 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Огляд літературних даних	18.11-31.12.2024	виконано
2	Розділ 1. Стан проблемної області	01.01-30.01.2025	виконано
3	Розділ 2. Інформаційне та математичне забезпечення	01.02-30.03.2025	виконано
4	Розділ 3. Програмне та технічне забезпечення	01.04-31.05.2025	виконано
5	Оформлення дипломної роботи	01.06-6.06.2025	виконано
6	Підготовка до захисту дипломної роботи, оформлення презентації	07.06-15.06. 2025	виконано

Студент _____ (підпис) Лемех Т.М. (прізвище та ініціали)

Керівник роботи _____ (підпис) Пірко І.Б. (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота містить 72 сторінки пояснювальної записки, 15 рисунків, 3 таблиці, 16 джерел, 2 додатки.

Дана робота присвячена розробці інтелектуальної системи для класифікації музичних жанрів за допомогою методів машинного навчання. В ній проведено аналіз існуючих підходів до класифікації музики та обрано відповідний набір даних для тренування моделі. Використано сучасні методи обробки звукових сигналів для отримання ознак аудіофайлів. На основі отриманих ознак побудовано модель класифікації музичних жанрів. Проведено оцінку ефективності моделі за допомогою метрик якості класифікації. Розроблену систему можна використовувати для інтеграції в музичні платформи, що дозволить автоматизувати процес розпізнавання музичних жанрів.

Ключові слова: *класифікація музичних жанрів, машинне навчання, Python, Google Colab, Pandas, Librosa*.

ABSTRACT

Diploma paper contains 72 pages of explanatory note, 15 figures, 3 tables, 16 sources, 2 appendix.

This work is devoted to the development of an intelligent system for classifying musical genres using machine learning methods. It analyzes existing approaches to music classification and selects an appropriate data set for training the model. Modern methods of processing sound signals are used to obtain features of audio files. Based on the obtained features, a model for classifying musical genres is built. The effectiveness of the model is assessed using classification quality metrics. The developed system can be used for integration into music platforms, which will allow automating the process of recognizing musical genres.

Keywords: *music genre classification, machine learning, Python, Google Colab, Pandas, Librosa.*

ТЕХНІЧНЕ ЗАВДАННЯ

В дипломній роботі потрібно розробити інтелектуальну систему класифікації музичних жанрів, для цього потрібно вирішити такі завдання.

1. Дослідити існуючі підходи до класифікації музичних жанрів, визначити основні особливості звукових сигналів для різних жанрів.

2. Обрати та завантажити набір даних GTZAN, виконати попередню обробку аудіосигналів.

3. Провести екстракцію ознак із аудіофайлів: перетворити аудіофайли у набір числових ознак, придатний для навчання моделей машинного навчання, візуалізувати основні характеристики даних.

4. Виконати навчання та тестування моделей машинного навчання на підготовлених даних.

5. Оцінити якість класифікації за допомогою метрик, порівняти результати різних моделей.

6. Розробити графічний інтерфейс інтелектуальної системи для завантаження аудіофайлу та його класифікації.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

Beats-per-minute – темп аудіосигналу, виражений в ударах за хвилину;

CNN (Convolutional Neural Network) – згорткова нейронна мережа для обробки аудіоданих;

DistributionShape – показує значення дисперсії, коефіцієнтів асиметрії;

GTZAN (GTZAN genre collection) – набір даних для класифікації музичних жанрів;

GradientBoostingClassifier() – модель класифікації на основі градієнтного бустингу;

k-NN – метод k найближчих сусідів;

Librosa – бібліотека Python для аналізу аудіосигналів;

LinearSVC (Linear Support Vector Classification) – метод машинного навчання, який використовується для задач класифікації музичних композицій;

LR (Logistic Regression) – метод класифікації, який використовується у задачах машинного навчання;

MFCC (Mel-Frequency Cepstral Coefficients) – коефіцієнти мел-частотного кепстрального аналізу, що відображають характеристики звуку;

ML (Machine learning) – машинне навчання;

Pitch-salience – рельєфність висоти для заданого спектра;

RNN (Recurrent Neural Networks) – рекурентні нейронні мережі;

ROC (Receiver operating characteristic) – графічне представлення залежності між чутливістю та специфічністю;

Root Mean Square – середньоквадратичне значення спектра аудіосигналу;

Spectral rolloff – спектральний спад для заданого спектра;

SVM (Support vector machines) – метод опорних векторів;

Streamlit – бібліотека для створення веб-додатків на Python;

ZeroCrossingRate – відображає частоту переходів аудіосигналу через нуль.

ВСТУП

Актуальність дипломної роботи

У сучасному світі об’єм музичного контенту стрімко зростає, що ускладнює його організацію та пошук для користувачів та музичних платформ. Класифікація музичних жанрів допомагає ефективно впорядковувати аудіофайли, що є важливим для стрімінгових сервісів, музичних бібліотек та рекомендаційних систем. Традиційні методи класифікації музики, які базуються на метаданих або ручному маркуванні, є малоефективними та потребують значних людських ресурсів. Використання методів машинного навчання дозволяє досягти високої точності розпізнавання жанрів завдяки аналізу звукових характеристик. Розробка інтелектуальної системи класифікації музичних жанрів сприятиме покращенню персоналізованих рекомендацій для слухачів. Така система може бути інтегрована в сучасні музичні сервіси, що допоможе автоматично визначати жанр нових композицій без втручання людини. Аналіз музичних сигналів відкриває можливості для подальших досліджень у сфері штучного інтелекту та аудіообробки.

Автоматизація процесу класифікації сприяє підвищенню ефективності роботи музичних платформ, радіостанцій та продюсерських центрів. Розвиток інтелектуальних систем для музичної класифікації може використовуватися в музичній терапії, освітніх проектах і дослідженнях музичних уподобань. Створення такої системи є актуальним завданням, що має широке застосування в реальному світі та сприятиме покращенню взаємодії між користувачами та музичним контентом.

Предмет дослідження – методи та алгоритми машинного навчання для класифікації музичних жанрів на основі аналізу аудіосигналів.

Об’єкт дослідження – музичні аудіосигнали та їхні характеристики, що використовуються для класифікації музичних жанрів.

Мета роботи – розробка інтелектуальної системи класифікації музичних жанрів на основі методів машинного навчання для процесу розпізнавання музичних композицій.

Завдання:

1. Дослідити існуючі підходи до класифікації музичних жанрів, визначити основні особливості звукових сигналів для різних жанрів.

2. Обрати та завантажити набір даних GTZAN, виконати попередню обробку аудіосигналів.

3. Провести екстракцію ознак із аудіофайлів: перетворити аудіофайли у набір числових ознак, придатний для навчання моделей машинного навчання, візуалізувати основні характеристики даних.

4. Виконати навчання та тестування моделей машинного навчання на підготовлених даних.

5. Оцінити якість класифікації за допомогою метрик, порівняти результати різних моделей.

6. Розробити графічний інтерфейс інтелектуальної системи для завантаження аудіофайлу та його класифікації.

Практичне значення одержаних результатів

Розроблена система класифікації музичних жанрів може бути інтегрована в музичні платформи та стрімінгові сервіси для автоматичного визначення жанру музичних композицій. Отримані результати можуть використовуватися для покращення алгоритмів персоналізованих рекомендацій у музичних додатках. Автоматизація класифікації музики сприятиме оптимізації роботи радіостанцій та музичних бібліотек, зменшуючи потребу в ручному маркуванні треків. Система може бути корисною для музичних продюсерів і діджеїв, допомагаючи швидко аналізувати та сортувати великі об’єми музичних файлів. Запропоновані методи можуть знайти застосування у музичній терапії, де класифікація жанрів допоможе підбирати відповідну музику для емоційного впливу. Розроблена модель може бути використана у дослідженнях музичних уподобань та автоматизованому аналізі трендів у музичній індустрії.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Підходи до класифікації музичних жанрів

Класифікація музичних жанрів - одне з ключових завдань в обробці музичних даних. Вона має широкий спектр застосувань, включно з рекомендаційними системами, автоматичною організацією музичних бібліотек і аналізом музичних уподобань користувачів. Розвиток технологій машинного навчання і глибокого навчання відкрив нові можливості для автоматизації цього процесу, даючи змогу аналізувати великі обсяги музичних даних із високим ступенем точності.

Перші дослідження в галузі класифікації музичних жанрів почалися в 1990-х роках і були присвячені використанню традиційних методів машинного навчання, таких як машини опорних векторів (SVM), дерева рішень і k-nearest neighbour. Ці методи ґрунтувалися на ручному вилученні ознак, таких як спектральні характеристики, ритмічні патерни та параметри тембру. Однак ефективність таких підходів була обмежена через складність вибору оптимальних ознак та їхню залежність від специфіки музичного жанру.

З розвитком глибокого навчання з'явилися нові методи, що дають змогу автоматично витягувати ознаки з музичних даних. Одним із найпопулярніших підходів є використання згорткових нейронних мереж (CNN), які дуже ефективні під час роботи з аудіоданими; CNN дають змогу аналізувати спектрограми аудіозаписів, що дає змогу виявити складні залежності між частотними та часовими властивостями музики [1].

Крім того, рекурентні нейронні мережі та їхні поліпшені варіанти, такі як довготривала і короткочасна пам'ять, використовуються для аналізу музичних послідовностей, наприклад, для виявлення ритмічних і мелодійних патернів. Ці підходи особливо ефективні для жанрів із чіткою часовою структурою, таких як класична музика та джаз [2].

Перспективним напрямком у класифікації музичних жанрів є використання транзитивного навчання. Цей підхід може використовувати попередньо навчені моделі, розроблені для інших завдань, таких як класифікація зображень або обробка

природної мови [3]. Наприклад, моделі, навчені на великих наборах даних, таких як ImageNet, можуть бути адаптовані до спектрограм музичних записів. Це може значно скоротити час навчання і підвищити точність класифікації.

Для навчання та оцінки моделей класифікації музичних жанрів використовуються різні набори даних. Одним з найвідоміших наборів даних є GTZAN, який класифікує 1000 аудіозаписів за 10 жанрами. Ці набори даних відрізняються за розміром, кількістю жанрів і якістю анотацій, що впливає на вибір методів і підходів до їх обробки.

Для оцінки ефективності моделей класифікації музичних жанрів використовуються стандартні метрики, такі як точність-акуратність, кількість відтворених композицій, точність-акуратність та середнє квадратичне відхилення (F1). Однак через унікальну природу музичних даних важливо враховувати такі фактори, як нерівномірний розподіл жанрів у наборі даних і можливість перетину між жанрами. Тому дослідники часто використовують додаткові метрики, такі як матриці невизначеності, для аналізу помилок класифікації.

Класифікація музичних жанрів є активною сферою досліджень і постійно розвивається завдяки новим методам машинного та глибокого навчання. Сучасні підходи дозволяють досягти високої точності класифікації, але все ще залишаються відкриті питання, такі як покращення обробки різножанрових записів, зменшення залежності від якості анотацій та адаптація моделей до нових музичних стилів. Ці аспекти є перспективними напрямками для подальших досліджень у розробці інтелектуальних систем для класифікації музичних жанрів.

1.2. Розпізнавання і класифікація музичних жанрів

Розпізнавання та класифікація музичних жанрів на основі аудіо даних є важливим завданням, відомим як класифікація музики. Зі стрімким зростанням музичних архівів мета класифікації музики очевидна. Кількість музичних зразків значно зросла, що ускладнює підтримку музичної послідовності вручну. Класифікація музичних жанрів має потенційне застосування в системах музичних рекомендацій [1] та потокових музичних сервісах [2], що призвело до інтенсивних

досліджень у цій галузі. Однак класифікація музики є складним завданням через нечітку природу різних музичних зразків. Отже, було б корисно дослідити класифікацію музики з постійною точністю.

Технологія нейронних мереж може бути використана для визначення жанру музики шляхом аналізу акустичних характеристик звукозаписів. У цьому випадку нейронна мережа навчається на наборі даних звукозаписів різних жанрів, а потім використовується для визначення жанру нового звукозапису. Для цього необхідно виокремити ознаки звукових записів. До них відносяться гучність, ритм [3,4], частота нульового перетину [4] та цепстральний коефіцієнт (MFCC) [5,6,7]. Також вивчалися спектрограми на основі перетворень Фур'є [8,9], вейвлет-перетворень [10] і перетворень з постійною добротністю [11], які містять більше частотної інформації (інформація про час, періодичні удари і ритм) і можуть забезпечити задовільну продуктивність.

Іншою важливою складовою класифікації музичних жанрів є розробка алгоритмів класифікації для обробки акустичних ознак. Класичні алгоритми машинного навчання включають статистичні методи, такі як наївні класифікатори Байєса [12], випадкові ліси [13] та машини опорних векторів. На додаток до традиційних підходів, з розвитком глибинного навчання [14], для класифікації музики стали використовуватися такі методи, як згорткові нейронні мережі (CNN) і рекурентні нейронні мережі (RNN), які є одними з найефективніших підходів для класифікації музичних даних [15]. У той же час, CNN краще реєструють просторові залежності областей ознак, тоді як RNN задовільно справляються з часовими залежностями послідовних даних.

Однак, існуючі методи мають різні обмеження. Виходячи з аналізу музичних сигналів, можна побачити, що музичний жанр є дуже широким поняттям. Музичні твори, що належать до одного жанру, можуть мати різні акустичні характеристики, такі як ритм і біт. На рисунку 1.1 показано, що спектрограми музичних треків, які належать до жанру рок, дуже відрізняються.

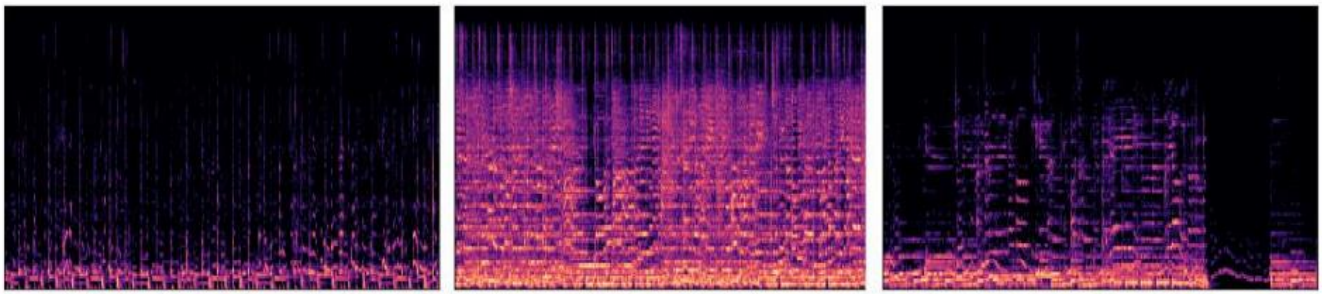


Рисунок 1.1 – Спектрограми аудіозаписів в жанрі рок

Недоліком існуючих методів є те, що вони не можуть добре впоратися з такими різноманітними розподілами даних з великими внутрішньокласовими відмінностями. Щоб правильно класифікувати різні аудіозаписи одного типу, модель повинна охоплювати ще глибшу приховану інформацію. Однак, якщо кількість даних недостатня, це може призвести до помилкових зсувів, що негативно вплине на точність класифікації. Згорткові нейронні мережі використовуються в цій статті для побудови нейромережевої архітектури.

Згорткові нейронні мережі (CNN) - це спеціалізована архітектура штучних нейронних мереж, призначена для ефективного розпізнавання образів. Назва мережевої архітектури походить від операцій згортки, суть яких полягає в тому, що кожен сегмент зображення поелементно множиться на матрицю згортки (ядро), а результати підсумовуються і записуються в ту ж позицію вихідного зображення.

Рекурентні нейронні мережі (RNN) - це тип нейронних мереж, в яких зв'язки між елементами утворюють спрямовану послідовність. Це дає можливість обробляти послідовність подій у часі. Оскільки кількість пісень продовжує зростати, необхідно класифікувати пісні за жанрами, щоб полегшити подальшу обробку пісень, зокрема пошук і рекомендацію пісень. Пропозиція щодо повністю автоматичної системи класифікації музичних жанрів була вперше висунута в роботі [16].

Вони представили три набори ознак, що представляють тональну текстуру, ритмічний зміст і основний тон, і навчили статистичний класифікатор розпізнавання образів, використовуючи реальні аудіоколекції. Представлений ними набір даних GTZAN став еталонним для більшості подальших робіт у цій галузі. З того часу було запропоновано кілька алгоритмів, заснованих на традиційному машинному

навчанні [12], де використовувався багатошаровий класифікатор на основі машин опорних векторів [13], де вектор MFCC, частота кольоровості, спектральне затухання, Для виконання завдань класифікації використовувалися акустичні характеристики, такі як спектральний центр ваги і швидкість нульового перетину. Для виконання завдання класифікації комбінувалися алгоритми SVM і k-NN (k-найближчих сусідів) [14], а для класифікації музичних жанрів досліджувалися наївні методи Байєса, дерева рішень, логістична регресія і випадкові ліси.

З появою алгоритмів глибокого навчання та обчислювальних ресурсів у цій галузі, алгоритми розпізнавання на основі глибоких нейронних мереж стають все більш популярними: В [11] використовується метод взаємодії проміжних навчальних ознак на основі згорткових нейронних мереж, в [12] описується теорія джазової музики з використанням нещодавно запропонованої архітектури глибоких нейронних мереж U-net, в [16] розроблено класифікатор Transformer для аналізу взаємозв'язків між різними аудіокадрами, а в [11] оцінюється продуктивність нової системи класифікації на основі Vision Transformer.

Крім того, успіх глибокого навчання як підходу, що ґрунтується на даних, неможливо відокремити від створення великих наборів даних. Крім того, ефективність системи класифікації залежить від якості та масштабу набору даних, і існує низка безкоштовних наборів даних, а також ряд наборів з відкритим вихідним кодом: GTZAN Genre Collection - широко використовуваний набір даних для класифікації музичних жанрів, що містить попередньо витягнуті ознаки, такі як Mel Frequency Cepstral Coefficients (MFCC) та ознаки насиченості. Free Music Archive - це велика колекція легальної безкоштовної музики, що містить метадані, такі як ім'я виконавця, назва альбому, жанр тощо. Million Song Dataset - це набір аудіо характеристик і метаданих для одного мільйона пісень сучасної популярної музики. MSD включає такі звукові характеристики, як висота тону, тембр і ритм, а також метадані, такі як ім'я виконавця, рік випуску і популярність. Ці набори даних сприяють розвитку спільноти з відкритим вихідним кодом і забезпечують достатню кількість навчальних даних для моделей глибокого навчання на основі даних.

В даний час стає актуальним завдання розпізнавання та пошуку музичних композицій у зв'язку зі збільшенням числа інтернет сервісів з музичною тематикою та зростання їх популярності. Пошук композицій можна розділити на два етапи. До першого етапу належить розпізнавання жанру музичної композиції з використанням моделі багатошарового персептрона. Другим етапом є сам пошук дубліката у межах бази даних композицій знайденого жанру. Загальну схему розв'язання задачі розпізнавання жанрів музичних творів можна умовно поділити на три етапи:

Зростаюча кількість і популярність музичних інтернет-сервісів підвищує важливість завдання розпізнавання і пошуку пісень. Пошук пісень можна розділити на два етапи. Перший етап - розпізнавання жанру пісні за допомогою багатошарової перцептронної моделі. Другий етап - власне пошук дублікатів пісень у базі даних пісень виявленого жанру. Загальну схему розв'язання задачі розпізнавання музичних жанрів можна розділити на три етапи:

- визначення набору класифікаційних ознак творів;
- виділення ознак із файлів аудіозаписів;
- класифікація за допомогою вибраної моделі.

Завдання правильної ідентифікації видів є складним як для людини, так і для комп'ютера. У більшості випадків не існує загальноприйнятої концепції характеристик жанрової класифікації, а також визначення жанрових типів. Хоча класифікації музичних жанрів є суб'єктивними, критерії музичних текстур і ритмічних структур можуть бути використані для визначення жанрів. Для ефективного навчання класифікаторів жанрів потрібні надійні, великомасштабні тестові дані, але загальнодоступних баз даних небагато.

Нейромережеві класифікатори зараз широко використовуються завдяки їх здатності навчатися на обмеженій кількості прикладів та наявності ефективних алгоритмів навчання. Однією з найбільш відомих і добре вивчених нейронних мереж є багатошаровий персептрон. Багатошарова мережа складається з набору вхідних вузлів, що утворюють вхідний шар, одного або декількох прихованих шарів обчислювальних нейронів і вихідного шару. Вхідні сигнали поширюються через мережу в прямому напрямку від шару до шару. Багатошарові персептрони

навчаються під контролем, а контрольоване навчання зазвичай виконується з використанням градієнтних алгоритмів і методів зворотного поширення. У цій статті представлено результати розв'язання задачі розпізнавання музичних жанрів за допомогою багатошарової персептронної моделі на музичній базі даних GTZAN.

1.3. Класифікаційні ознаки музичних жанрів

Основним завданням аналізу аудіосигналу є виокремлення ознак, які характеризують сигнал. У багатьох дослідженнях запропоновано три набори ознак, що представляють тембр, ритм і висоту звуку. Набір тембрових ознак відповідає загальному розпізнаванню мови, тоді як інші два набори характеризують аспекти музики. Тембр визначається як звукова характеристика, яка змушує два звуки однакової висоти та гучності звучати по-різному [3]. Спектральний розподіл сигналу використовується для отримання ознак, що представляють тембр, але деякі з цих ознак обчислюються в часовій області. Багато з цих ознак також можуть бути використані в контексті ідентифікації музичних жанрів. У статті [4] представлено основні низькорівневі ознаки, що використовуються в додатках для розпізнавання жанрів:

- тимчасові ознаки – обчислюються із кадрів звукового сигналу;
- енергетичні ознаки – середньоквадратична енергія кадру сигналу, енергія гармонійної складової спектра сигналу, енергія шумової частини спектра;
- спектральні ознаки – ознаки, що описують форму спектра звукового кадра;
- перцептивні ознаки – ознаки, які стосуються сприйняття, обчислюються з допомогою моделі людського сприйняття звуку (відносна гучність, чіткість).

Перетворення значень ознак, такі як похідні першого та другого порядку, є поширеними способами створення нових ознак або збільшення розмірності вектора ознак. У контексті завдань класифікації значення тембрових ознак часто узагальнюють, застосовуючи статистику нижчого порядку до великого вікна, яке називається вікном текстури. Висотні характеристики - це характеристики, які описують гармонію або мелодію записаної пісні. Гармонію можна визначити як використання і вивчення групи нот або акордів. Мелодія - це послідовність нот

різної висоти, що сприймається як єдине ціле. Гармонію іноді називають горизонтальним аспектом музики, тоді як мелодію - вертикальним. Гармонія і мелодія більш надійно описуються атрибутами нижчого рівня, ніж ноти і акорди [5].

Основна ідея більшості аналізаторів мелодії та гармонії полягає у використанні функцій, які описують розподіл висот у коротких ділянках композиції. Високорівневі характеристики, такі як висота тону, основна частота і послідовність акордів, не використовуються; натомість функція застосовується для обчислення діапазону значень амплітуди положення основних піків, розміру інтервалів між піками та інших статистичних дескрипторів розподілу висоти тону.

Точного визначення ритму не існує. Ритм визнається як часова закономірність. У більш загальному сенсі термін ритм можна використовувати для позначення всіх часових аспектів музики. Ритмічні патерни є важливою характеристикою певних жанрів. Сучасні системи виявлення ритму мають недоліки, а системи автоматичної жанрової класифікації часто використовують низькорівневий підхід [6]. Використовуючи той самий підхід, що і для низькорівневих характеристик висоти тону, ознаки можуть бути вилучені за допомогою функції, яка оцінює періодичність виявленого темпового діапазону (від 40 до 200 ударів на хвилину). Ознаки також можна виділити з усього аудіосигналу, але оскільки багато музичних жанрів мають повторювані музичні структури, багато завдань класифікації використовують невеликі музичні сегменти, які містять достатньо інформації, щоб описати весь твір. Використання невеликих відрізків сигналу може значно зменшити обчислювальні витрати. Часто використовується 30-секундний сегмент через 30 секунд після початку пісні.

1.4. Отримання ознак із аудіозаписів

Метод отримання спектральних характеристик має справу з амплітудним спектром сигналу. Спектр являє собою послідовність комплексних чисел і обчислюється наступним чином. Модуль обробки аудіосигналу зчитує дані аудіофайлу, на вході якого знаходиться ім'я аудіофайлу, а на виході модуля - вихідні аудіодані (серія відліків, взятих через рівні проміжки часу). Стандартна частота

дискретизації хвильового файлу становить 44 100 Гц, а амплітуда сигналу вимірюється 44 100 разів на секунду.

У швидкому перетворенні Фур'є всі відліки є комплексними числами. Амплітудний спектр аудіофайлу обчислюється для серії комплексних відліків. Весь спектр розбивається на окремі вікна (кадри). Значення ознак, отримані для спектра кожного кадру, нормалізуються та усереднюються. Потім зі спектра виділяються ознаки, необхідні для класифікації та виявлення перекриттів. Перелічимо ознаки сигналу, які використовуються як класифікаційні ознаки в цій статті.

ZeroCrossingRate - це характеристика, яка показує частоту, з якою аудіосигнал перетинає нуль, тобто кількість змін знаку між послідовними значеннями сигналу, поділена на загальну кількість значень. Частота перетину нуля є показником зашумленості сигналу. Порогове значення встановлюється для того, щоб запобігти невеликим коливанням близьких до нуля значень через шум.

DistributionShape - це властивість, яка відображає значення коефіцієнтів дисперсії, асиметрії та ексцесу (гостроти піків розподілу) з масиву центральних моментів розподілу. Масив центральних моментів розраховується за допомогою алгоритму, який обчислює центральні моменти від нульового до четвертого порядку для заданого набору значень (в даному випадку спектру аудіосигналу).

Root Mean Square – характеристика середньоквадратичного значення вхідного масиву (спектра аудіосигналу).

Спектральне затухання - це характеристика спектрального затухання в даному спектрі, яка визначається як частота, нижче якої зосереджений певний відсоток загальної енергії спектра. Частота спектрального спаду може бути використана для відокремлення шуму від значущого вмісту.

Сильний пік - це сильний пік, характерний для даного спектра і визначається як відношення максимального значення піку в спектрі до ширини цього піку вище порогового значення, що дорівнює половині максимальної амплітуди піку. Це відношення показує, чи має спектр чіткий максимальний пік.

Виразність висоти тону - це характеристика виразності тону для даного спектру, яка визначається як відношення найвищого значення автокореляції спектру

до незміщеного значення автокореляції. Немузичні звукові ефекти та чисті тони (звуки з обертоновими коливаннями на тій самій частоті) мають значення висотної помітності близьке до нуля, тоді як звуки з кількома обертонами мають вищу висотну помітність.

Гучність - це характеристика гучності аудіосигналу, яка визначається за законом Стівенса як енергія сигналу з точністю до степеня 0,67. Для обчислення цієї характеристики використовується діапазон з 88 200 відліків аудіосигналу. Для кожного з цих інтервалів обчислюється значення гучності, а потім розраховується середнє значення гучності.

Beats-per-minute – характеристика темпу аудіосигналу, виражена в ударах за хвилину. Кожен аудіозапис описується 8 характеристиками, що подаються на вхід нейронної мережі.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Аналіз аудіоданих за допомогою методів машинного навчання

Метою цієї роботи є класифікація музичних жанрів за допомогою згорткових нейронних мереж. Ці нейронні мережі було реалізовано на основі даних, отриманих за допомогою спектрограм та MFCC. Експерименти були проведені на наборі даних GTZAN. Цей набір даних містить 1000 музичних відео, кожне з яких складається з 100 пісень у 10 жанрах (блюз, класика, кантрі, диско, хіп-хоп, джаз, метал, поп, реггі та рок), кожна з яких розділена на 30 секунд, а також таблицю даних, отриманих з 30-секундних сегментів. Частота дискретизації - 22050 Гц, розмір кожного кліпу - 16 біт з моноканалом, закодованим у форматі mp3. Модель навчається на наборі даних 30-секундних пісенних кліпів і розділена на три частини: навчальні дані (70%), валідаційні дані (20%) та тестові дані (10%). Для середовища розробки було обрано платформу Google Colab, яка інтегрує всі необхідні бібліотеки, зокрема NumPy, Pandas, Keras та Tensorflow.

Аудіосигнал - це цифровий аудіофайл у форматі .wav. Звукові хвилі відбираються і оцифровуються через певні проміжки часу, які називаються частотою дискретизації (44,1 кГц, 44100 відліків на секунду). Кожна вибірка - це амплітуда хвилі на певному часовому інтервалі, а розрядність визначає деталізацію вибірки, яку також називають динамічним діапазоном сигналу (16 біт, одна вибірка складається з 65536 значень амплітуди).

В обробці сигналів дискретизація - це перетворення безперервного сигналу в серію дискретних значень. Частота дискретизації - це кількість відліків, зроблених за певний проміжок часу. Висока частота дискретизації призводить до низьких втрат інформації, але високих обчислювальних витрат, тоді як низька частота призводить до високих втрат інформації, але є швидшою і дешевшою.

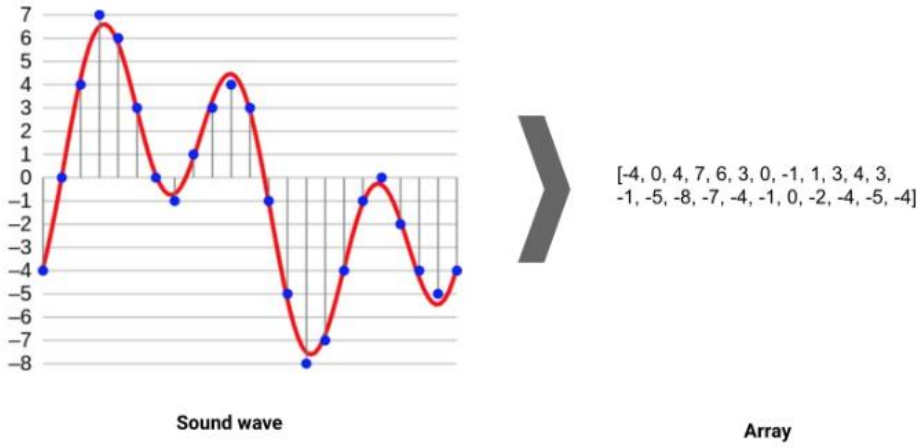


Рисунок 2.1 – Звукова хвиля червоного кольору, представлена в цифровому вигляді синім кольором

Обробка звукових даних за допомогою Python. Звук представляється у вигляді аудіосигналу з такими параметрами, як частота, смуга пропускання і децибел. Типовий звуковий сигнал можна виразити як функцію амплітуди та часу.

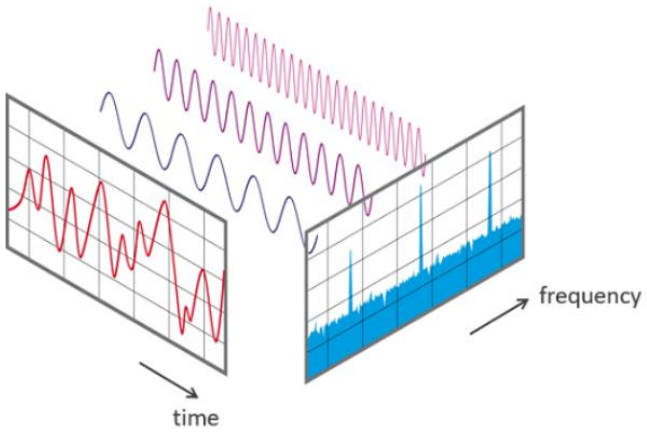


Рисунок 2.2 – Представлення звукового сигналу

Існують пристрої, які допомагають вловлювати ці звуки та представляти їх у форматі, який читається комп’ютером. Прикладами таких форматів є:

- wav (Waveform Audio File);
- mp3 (MPEG-1 Audio Layer 3);
- WMA (Windows Media Audio).

Типовий процес обробки аудіо передбачає виокремлення акустичних характеристик, що мають відношення до поставленого завдання, а потім схему прийняття рішень, що включає виявлення, класифікацію та злиття інформації. Існує кілька бібліотек Python, які роблять це можливим.

Python має кілька бібліотек для обробки звуку, таких як Librosa та PyAudio. Існують також вбудовані модулі для деяких базових аудіо функцій. У більшості випадків для імпорту та відтворення аудіо використовуються дві бібліотеки.

Librosa - модуль Python для аналізу аудіосигналів взагалі, але більш специфічний для музики; використовується для створення систем MIR (Music Information Retrieval); IPython.display.Audio можна використовувати для відтворення аудіо безпосередньо в блокноті Jupyter IPython.display.Audio дозволяє відтворювати аудіо безпосередньо в блокноті Jupyter.

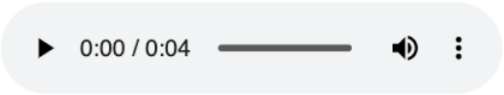


Рисунок 2.3 – Віджет для відтворення звукового файлу в Google Colab

Для візуалізації звуку можна побудувати аудіо масив за допомогою методу librosa.display.waveplot:

```
import matplotlib.pyplot as plt
import librosa.display
plt.figure(figsize=(14, 5))
librosa.display.waveplot(x, sr=sr)
```

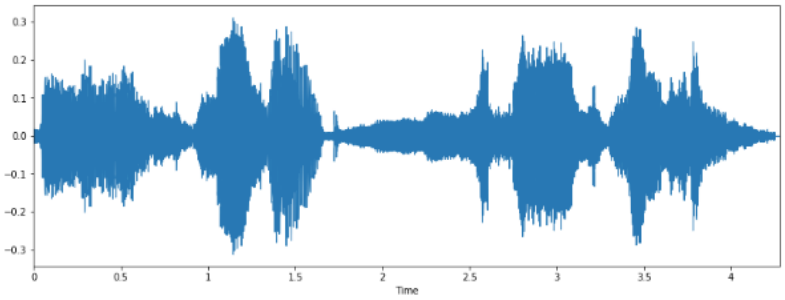


Рисунок 2.4 – Хвильовий графік звукового файлу

Отримали графік огинаючої амплітуди хвилі.

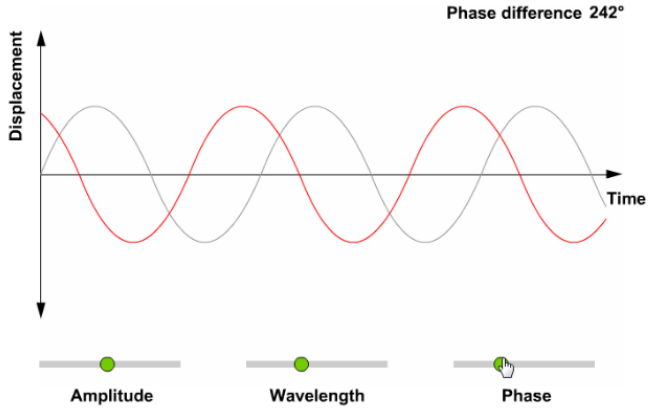


Рисунок 2.5 – Амплітуда, довжина хвилі та фаза звукового сигналу

Спектрограма - це спосіб візуального представлення сили або гучності сигналу на різних частотах у певній формі сигналу в часі. Ви можете не тільки побачити, чи є більше або менше енергії, наприклад, на частотах 2 Гц і 10 Гц, але й побачити, як рівні енергії змінюються з часом. Спектрограма відображається у вигляді теплової карти, тобто зображення, що представляє інтенсивність у вигляді змін кольору і яскравості. Спектрограму можна відобразити за допомогою методу librosa.display.specshow.

```
X = librosa.stft(x)
Xdb = librosa.amplitude_to_db(abs(X))
plt.figure(figsize=(14, 5))
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='hz')
plt.colorbar()
```

Метод stft() виконує короткочасне перетворення Фур'є даних. stft перетворює сигнал так, щоб можна було дослідити амплітуду заданої частоти в заданий момент часу. stft можна використовувати для визначення амплітуди різних частот, що відтворюються в заданий момент часу в аудіосигналі. метод specshow використовується для відображення спектрограми.

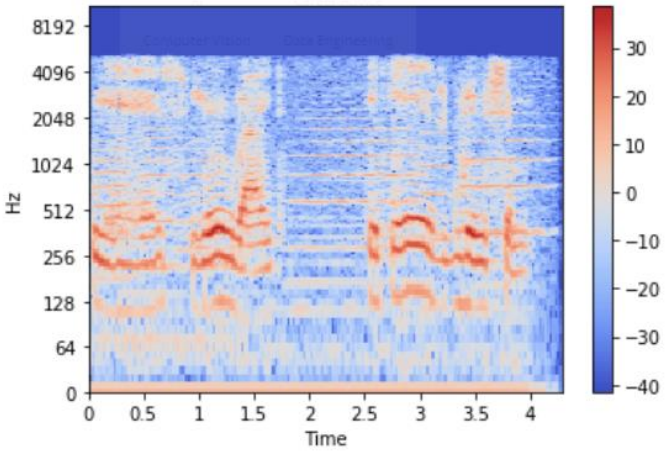


Рисунок 2.6 – Спектрограма звукового сигналу

Вертикальна вісь представляє частоту (0-10 кГц), а горизонтальна вісь - час. Частотну вісь можна перетворити на логарифмічну, оскільки видно, що вся дія відбувається в нижній частині спектра.

Усі аудіосигнали мають багато особливостей. Процес виокремлення ознак для використання в аналізі називається виділенням ознак. Основні частоти, частотні

складові, спектральний центр ваги, спектральний потік, спектральна щільність, спектральне згасання тощо.

Спектральний центроїд вказує на частоті, на якій зосереджена енергія спектру, вказує, де знаходиться центр мас звуку. Це як зважене середнє:

$$f_c = \frac{\sum_k S(k)f(k)}{\sum_k S(k)} \quad (2.1)$$

де *S(k)* - спектральна величина на частоті *k*, *f(k)* - частота на *k*. З допомогою методу librosa.feature.spectral_centroid обчислюють спектральний центроїд для кожного кадру в сигналі.

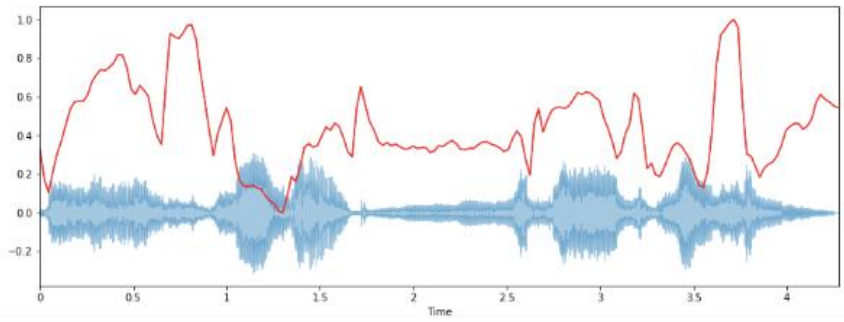


Рисунок 2.7 – Спектральний центроїд для звукового файлу

Спектральне загасання - це показник форми сигналу. Воно являє собою частоту, на якій високі частоти падають до нуля. Щоб отримати це значення, потрібно підрахувати відсоток квадратів, де 85% спектра потужності припадає на низькі частоти; використовуйте метод librosa.feature.spectral_rolloff, щоб обчислити частоту спаду для кожного кадру сигналу:

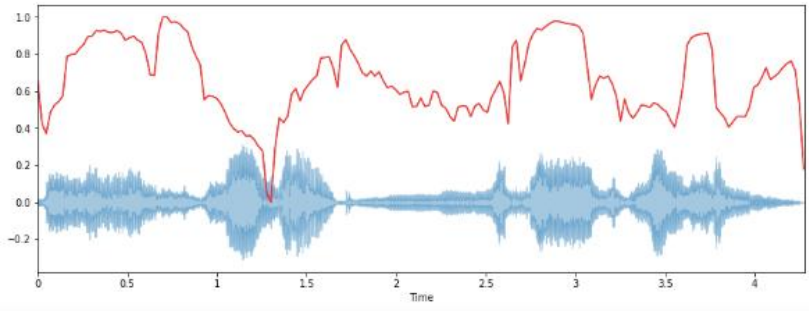


Рисунок 2.8 – Спектральний спад для звукового файлу

Спектральна смуга пропускання визначається як оптична смуга пропускання на половині пікового максимуму і представлена двома вертикальними червоними лініями та λSB на осі довжин хвиль.

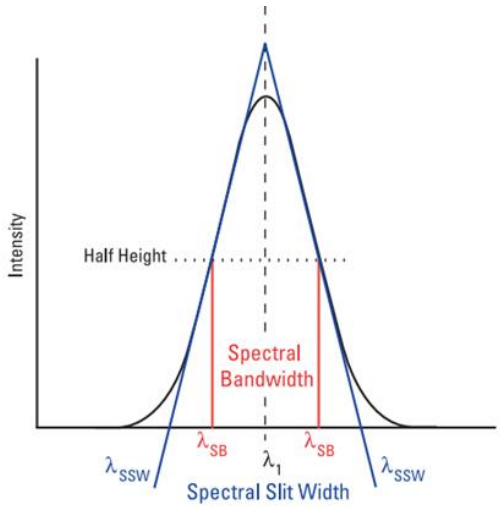


Рисунок 2.9 – Спектральна смуга пропускання

Мел-частотний цепстральний коефіцієнт (MFCC) аудіосигналу - це невеликий набір характеристик (10-20), які описують загальну форму спектральної огинаючої. Це моделює характеристики людського голосу.

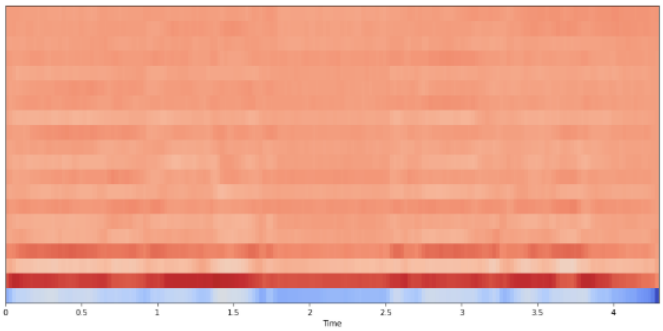


Рисунок 2.10 – Мел-частотні спектральні коефіцієнти звукового сигналу

2.2. Набір даних GTZAN

У цьому дослідженні ми проаналізували набір даних GTZAN - класичний набір даних для класифікації музичних жанрів. Він підходить для навчання базових моделей, але через свої обмеження його потрібно комбінувати з іншими наборами даних або застосовувати доповнення даних. Використання:

- побудова моделей глибокого навчання;
- дослідження музичних характеристик жанрів;
- використання спектрограм для CNN.

GTZAN (GTZAN Genre Collection) - один з найпопулярніших наборів даних для класифікації музичних жанрів, створений Джорджем Тзанетакісом у 2002 році і

використовуваний для тестування моделей машинного навчання в галузі аналізу аудіо.

Структура даних: 10 музичних жанрів по 100 аудіофайлів у кожному жанрі, усього 1000 аудіофайлів. Формат аудіо WAV (22050 Гц, 16-біт, моно). Тривалість кожного файлу 30 секунд.

Жанри в GTZAN:

- blues;
- classical;
- country;
- disco;
- hip-hop;
- jazz;
- metal;
- pop;
- reggae;
- rock.

Цей набір даних широко використовується для розробки моделей машинного навчання, які можуть класифікувати жанри на основі аналізу мовлення.

Основні підходи:

- аналіз частотних характеристик – спектрограми, мел-спектрограми, MFCC (Mel-Frequency Cepstral Coefficients);
- машинне навчання – SVM, KNN, Random Forest;
- глибоке навчання – CNN (Convolutional Neural Networks), RNN (Recurrent Neural Networks), Transformer.

Кожен аудіосигнал було попередньо оброблено і розглянуто різні комбінації спектральних характеристик, отриманих шляхом перетворення часового сигналу в частотну область за допомогою перетворення Фур'є [3-4]. Були визначені наступні детермінанти: 20 частотний коефіцієнт Мела, спектральний центр ваги, частота перетину нуля, частота кольоровості, спектральне згасання, спектральна ширина та

інші. Коефіцієнти зменшувалися при виборі найкращої комбінації класифікаційних ознак.

2.3. Математичне забезпечення інтелектуальної системи класифікації музичних жанрів

Математичне забезпечення системи класифікації музичних жанрів базується на методах машинного навчання. Основні математичні моделі, що використовуються в системі, включають наступні елементи.

Вилучення характеристик аудіофайлу. Для кожного аудіофайлу обчислюються числові характеристики, що описують звукові характеристики пісні, такі як частотні цепстральні коефіцієнти (MFCC), спектральні та хроматичні характеристики. Ці ознаки є основою для подальшої класифікації та навчання моделі.

Алгоритми класифікації. Система використовує математичні моделі машинного навчання для класифікації музичних жанрів.

Оцінка ефективності. Для оцінки ефективності моделі використовуються такі метрики, як точність, прецизійність, F1-міра та матриця плутанини. Крім того, ROC-графіки використовуються для оцінки моделей при різних порогах класифікації.

Оптимізація гіперпараметрів. Математичне забезпечення також включає оптимізаційні процедури, зокрема методи градієнтного спуску для налаштування ваг нейронної мережі та гіперпараметрів моделі. Таким чином, математичне забезпечення інформаційної системи включає традиційні методи машинного навчання, які дозволяють досягти високої точності та ефективності класифікації музичних жанрів.

Математичне забезпечення системи класифікації музичних жанрів базується на декількох базових математичних моделях, які використовуються для отримання ознак, навчання та оцінки ефективності класифікації.

2.3.1. Логістична регресія для класифікації музичних жанрів

Логістична регресія - один з найпростіших і найефективніших методів класифікації, який широко використовується в задачах машинного навчання. Модель використовується для бінарної та багатокласової класифікації і базується на імовірнісному підході. Розглянемо використання логістичної регресії для класифікації музичних файлів за жанрами.

Логістична регресія - це статистичний метод, який використовується для оцінки ймовірності належності об'єкта до певного класу. Основним математичним апаратом моделі є логістична (сигмоїдальна) функція.

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \tag{2.1}$$

де $z=w_0+w_1x_1+w_2x_2+...+w_nx_n$ - лінійна комбінація вхідних ознак.

Для багатокласової класифікації використовується узагальнення логістичної регресії у вигляді Softmax-регресії:

$$P(y = k|x) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}, \tag{2.2}$$

де K - кількість класів, z_k - значення лінійної комбінації вагових коефіцієнтів для класу k.

Класифікація музичних жанрів є складним завданням, оскільки жанри можуть мати схожі або змішані характеристики. Вхідні дані для моделей логістичної регресії зазвичай генеруються на основі акустичних і спектральних характеристик музичних файлів, наприклад:

- MFCC (Mel-Frequency Cepstral Coefficients) – коефіцієнти мел-частотного кепстрального аналізу, що відображають характеристики звуку;
- Chroma features – ознаки, що відображають гармонічні аспекти музики;
- Spectral Contrast – контраст спектра, який допомагає диференціювати різні музичні жанри;
- Tempo – темп композиції, який може бути ключовим фактором при класифікації.

Перед тренуванням моделі необхідно виконати попередню обробку аудіофайлів. Основні етапи:

- завантаження даних – використання набору даних GTZAN;
- екстракція ознак – отримання числових ознак із музичних файлів за допомогою бібліотеки Librosa;
- нормалізація даних – масштабування значень ознак для кращої роботи моделі;
- розділення даних – поділ на навчальну та тестову вибірки (80 % – навчальна, 20 % – тестова).

Модель логістичної регресії можна реалізувати за допомогою бібліотеки Scikit-learn. Основні етапи:

```
import numpy as np
import librosa
import librosa.display
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
data = pd.read_csv("features.csv")
X = data.drop(columns=['genre'])
y = data['genre']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
model = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=500)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

Після того, як модель навчена, оцінюється її якість. Основним показником є точність. Оскільки види можуть мати значну взаємну схожість, необхідні інші методи оцінки, такі як матриці плутанини та оцінки F1.

Логістична регресія - проста та ефективна модель для класифікації музичних жанрів. Вона швидко навчається і легко інтерпретується, але має обмеження, коли має справу зі складними нелінійними зв'язками. Для більш точної класифікації рекомендується використовувати більш складні моделі, такі як глибокі нейронні мережі або ансамблеві методи. Однак логістична регресія може бути хорошим базовим підходом для систем класифікації музики.

2.3.2. Модель LinearSVC для класифікації музичних жанрів

Лінійна класифікація опорних векторів (LinearSVC) - це метод машинного навчання, який використовується для задач класифікації. Це різновид методу опорних векторів (SVM), розроблений для швидшого навчання на великих наборах даних. LinearSVC підходить для задач багатокласової класифікації та ефективний для

LinearSVC базується на концепції пошуку оптимальної гіперплощини, яка розділяє різні класи об'єктів у просторі ознак. Його математична модель описується наступним рівнянням:

$$f(x) = w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n, \quad (2.3)$$

де *w* – вектор вагових коефіцієнтів, *x* – вхідні ознаки. Оптимізація виконується за допомогою функції втрат:

$$L(w, b) = \sum_{i=1}^{N} \max(0, 1 - y_i(w \cdot x_i + b)) + \lambda||w||^2, \quad (2.4)$$

де $\lambda||w||^2$ – регуляризаційний член, який запобігає перенавчанню.

Класифікація музичних жанрів є складним завданням через змішаність характеристик між жанрами. Вхідні ознаки для LinearSVC можуть включати:

- MFCC (Mel-Frequency Cepstral Coefficients) – характеристики спектрального аналізу, що відображають тембральні особливості музики;
- Chroma features – ознаки, що відображають гармонійні аспекти композиції;
- Spectral Contrast – контраст спектра, який допомагає диференціювати жанри;
- Rhythm Features – характеристики ритму та темпу композиції.

Модель LinearSVC можна реалізувати за допомогою бібліотеки Scikit-learn. Основні кроки:

```
import numpy as np
import librosa
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score
data = pd.read_csv("features.csv")
X = data.drop(columns=['genre'])
y = data['genre']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)
    model = LinearSVC(max_iter=5000, dual=False)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy: {accuracy:.2f}")
```

Після того, як модель навчена, оцінюється її точність. Основним показником точності є прецизійність. Крім того, для більш глибокого аналізу якості класифікації можна використовувати оцінку F1 матриці помилок або матриці невизначеності.

LinearSVC є ефективною моделлю для задач класифікації музичних жанрів, оскільки вона швидко навчається і може обробляти великі масиви даних. Інші підходи, такі як глибокі нейронні мережі, можуть знадобитися, коли жанри мають складні нелінійні зв'язки. LinearSVC є базовою моделлю для експериментів і початкового аналізу класифікації аудіо даних.

Виділення ознак аудіофайлів. Важливим кроком у класифікації музичних жанрів є перетворення аудіофайлів у числові ознаки, що характеризують звукові характеристики. Одним з основних методів є вилучення MFCC, який широко використовується для представлення аудіосигналів; формула обчислення MFCC виглядає наступним чином:

$$MFCC = DCT(\log(|S(f)|^2)) \tag{2.5}$$

де *S(f)* - спектр звукового сигналу, отриманий після застосування Фур'є-перетворення, log - логарифмічне перетворення для підвищення контрасту серед низькочастотних компонент, *DCT* - дискретне косинусне перетворення, яке дозволяє отримати компактне представлення сигналу.

Крім того, в систему можуть бути включені інші ознаки, такі як хроматичні ознаки, що описують гармонічні компоненти сигналу:

$$C(t) = \sum_{f \in \text{chromas}} |S(f,t)| \tag{2.6}$$

де *S*(*f*, *t*) - амплітуда частоти *f* в момент часу *t*; *C(t)* - набір частот, що відповідають музичним нотам.

Для оцінки результатів класифікації використовуються стандартні метрики машинного навчання, такі як точність, пригадування, точність, F1-міра та матриця

невизначеності. Точність визначається як відношення кількості правильно класифікованих об'єктів до загальної кількості об'єктів:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.7}$$

де *TP* - кількість справжніх позитивних результатів, *TN* - кількість справжніх негативних результатів, *FP* - кількість хибних позитивних результатів, *FN* - кількість хибних негативних результатів.

F1-міра є середнім гармонійним між точністю та чутливістю:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.8}$$

Для оцінки моделі на різних порогах класифікації використовують ROC (Receiver Operating Characteristic) криву, яка показує залежність між чутливістю (True Positive Rate) та специфічністю (1 - False Positive Rate):

$$\text{True Positive Rate} = \frac{TP}{TP + FN}, \quad \text{False Positive Rate} = \frac{FP}{FP + TN} \tag{2.9}$$

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Інтелектуальна система класифікації музичних жанрів

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

Імпортуйте бібліотеки для аналізу даних та машинного навчання на Python; Pandas - бібліотека для роботи з табличними даними, що використовується для обробки, аналізу та візуалізації даних; Numpy - бібліотека для роботи з числовими масивами, здатна виконувати векторизовані операції та обчислення; Numpy - бібліотека для роботи з даними у форматі графіків та діаграм. Matplotlib - бібліотека візуалізації даних для створення графіків та діаграм.

```
data = pd.read_csv("data/features_30_sec.csv", index_col="filename")
data.head()
```

filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	spectral_bandwidth_var	rolloff
blues.00000.wav	661794	0.350088	0.088757	0.130228	0.002827	1784.165850	129774.064525	2002.449060	85882.761315	3805.8
blues.00001.wav	661794	0.340914	0.094980	0.095948	0.002373	1530.176679	375850.073649	2039.036516	213843.755497	3550.5
blues.00002.wav	661794	0.363637	0.085275	0.175570	0.002746	1552.811865	156467.643368	1747.702312	76254.192257	3042.2
blues.00003.wav	661794	0.404785	0.093999	0.141093	0.006346	1070.106615	184355.942417	1596.412872	166441.494769	2184.7
blues.00004.wav	661794	0.308526	0.087841	0.091529	0.002303	1835.004266	343399.939274	1748.172116	88445.209036	3579.7

Набір даних features_30_sec.csv містить 1000 записів, включає 60 стовпців, більшість з яких є числовими, також є два текстові. Дані повні, відсутніх значень немає.

Структура набору даних.

Ідентифікаційні дані:

- filename - ім'я файлу (наприклад, blues.00000.wav);
- length - довжина аудіофайлу в стиснених одиницях (кількість семплів).

Спектральні характеристики:

- chroma_stft_mean, chroma_stft_var - середнє та дисперсія хроматичної STFT (Short-Time Fourier Transform), що показує енергетичний розподіл по частотах;
- spectral_centroid_mean, spectral_centroid_var - центр мас спектра, оцінює яскравість звуку;
- spectral_bandwidth_mean, spectral_bandwidth_var - ширина спектру;

- rolloff_mean, rolloff_var - точка, у якій енергія спектра зменшується до певного порогу;
- zero_crossing_rate_mean, zero_crossing_rate_var - кількість разів, коли сигнал змінює знак, використовується при аналізі тембру.

Динамічні характеристики;

- rms_mean, rms_var - середньоквадратичне значення амплітуди звуку та його дисперсія;
- harmony_mean, harmony_var - гармонічні характеристики звуку;
- perceptr_mean, perceptr_var - перцептивні характеристики;

Ритмічні характеристики:

- tempo - темп музичного треку, ударів на хвилину.

MFCC (Mel-frequency cepstral coefficients), коефіцієнти, які використовуються для аналізу спектральної структури:

- mfcc1_mean, mfcc1_var, ..., mfcc20_mean, mfcc20_var - 20 середніх значень та їх дисперсії.

Теги жанрів: теги - музичні жанри. Ці ознаки є стандартними для класифікації музичних жанрів та аналізу аудіофайлів у задачах обробки аудіосигналів. Теги в останньому стовпчику містять жанри музичних треків, які є категоріальними ознаками. Всього існує 10 жанрів, кожен з яких містить по 100 треків (всього 1000 записів).

Блюз характеризується розслабленим темпом і м'якими гармоніями. Класичні композиції записуються в академічному стилі і часто використовують оркестрові інструменти. Кантрі містить елементи народної та традиційної американської музики. Для диско характерні танцювальні ритми та синтезовані звуки. Хіп-хоп - ритмічний стиль з чітким перкусійним супроводом. Джаз - складні гармонії, імпровізація та багатий тембр. Метал - агресивний стиль з важким гітарним звучанням. Поп - легка, комерційна музика з простими гармоніями. Реггі - слабкі ритми, домінуючі басові лінії, спокійні темпові ритми. Рок - динамічний стиль з гітарним супроводом.

Жанри відрізняються темпом (tempo), гармоніями (harmony_mean), спектральними ознаками (spectral_centroid_mean, rolloff_mean). Metal та rock мають високий zero_crossing_rate_mean та tempo. Classical має низькі zero_crossing_rate_mean та spectral_centroid_mean. Reggae та hiphop мають низький темп, але відрізняються гармонійним складом. Pop та disco мають високий spectral_rolloff_mean, оскільки збагачені високочастотними звуками.

```
data.isnull().sum().sum()
```

```
0
```

Ця команда використовується для перевірки пропущених значень у наборі даних. Команда підраховує загальну кількість відсутніх значень у даних. Якщо результат дорівнює 0, то в наборі даних немає пропущених значень; якщо результат більше 0, то пропущені значення необхідно обробити і заповнити. Результат 0 означає, що в наборі даних немає пропущених значень і додаткової обробки або заповнення пропущених даних не потрібно.

```
data['label'].unique()
```

Отримати унікальне значення стовпчика тегів, що містить жанр музики. Вивести результат:

```
array(['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz',
       'metal', 'pop', 'reggae', 'rock'], dtype=object)
```

Масив показує, що набір даних містить 10 унікальних музичних жанрів, що відповідає очікуваній структурі набору даних GTZAN.

```
data.describe()
```

	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	spectral_bandwidth_var	r
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000	1000.000000	
mean	662030.846000	0.378682	0.086340	0.130930	0.003051	2201.780898	4.696916e+05	2242.541070	137079.155165	
std	1784.073992	0.081705	0.007735	0.065683	0.003634	715.960600	4.008995e+05	526.316473	96455.666326	
min	660000.000000	0.171939	0.044555	0.005276	0.000004	570.040355	7.911251e+03	898.066208	10787.185064	
25%	661504.000000	0.319562	0.082298	0.086657	0.000942	1627.697311	1.843505e+05	1907.240605	67376.554428	
50%	661794.000000	0.383148	0.086615	0.122443	0.001816	2209.263090	3.384862e+05	2221.392843	111977.548036	
75%	661794.000000	0.435942	0.091256	0.175682	0.003577	2691.294667	6.121479e+05	2578.469836	182371.576801	
max	675808.000000	0.663685	0.108111	0.397973	0.027679	4435.243901	3.036843e+06	3509.646417	694784.811549	

8 rows × 58 columns

Цей метод використовується для розрахунку основних статистичних показників для кожного числового стовпчика і розуміння розподілу даних.

Таблиця 3.1 – Типові метрики набору даних

Метрика	Значення
count	кількість непорожніх значень у стовпці
mean	середнє значення
std	стандартне відхилення (розкид значень)
min	мінімальне значення
25 %	1-й квартиль (нижні 25% значень)
50 %	2-й квартиль (медіана, середнє значення)
75 %	3-й квартиль (верхні 25% значень)
max	максимальне значення

З цього аналізу можна отримати наступну інформацію Діапазон значень - мінімальне та максимальне значення вказують на діапазон. Середнє та медіана - чи є значне відхилення. Середньоквадратичне відхилення std - чи не надто розкидані дані. Викиди - чи є викиди.

Для spectral_centroid_mean середнє значення становить 2201,78, а стандартне відхилення - 715,96, тобто значення розподілені навколо середнього. Для rms_mean (середня енергія сигналу) мінімальне значення дорівнює 0,005276, а максимальне - 0,397973. Медіанне значення близьке до середнього, що свідчить про відсутність сильних викидів у більшості стовпчиків.

Дані добре збалансовані та не містять пропущених значень. У більшості стовпчиків середні та середні значення схожі, і немає сильних викидів. Розкид значень для кожної ознаки є різним, що може вплинути на навчання моделі.

```
plt.scatter(data['length'], data['label'])
```

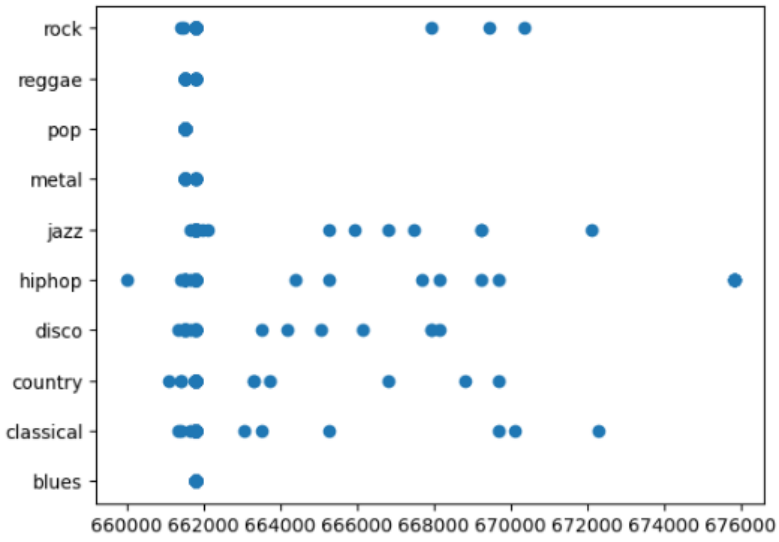


Рисунок 3.1 – Діаграма розсіювання довжини аудіозаписів за музичними жанрами

Створюють діаграму розсіювання зі значеннями у стовпчику Length (довжина аудіофайлу, кількість семплів) на осі X та значеннями у стовпчику Tags (музичний жанр) на осі Y. Цей графік є діаграмою розсіювання і показує залежність між двома змінними: Вісь X - довжина аудіозапису, а вісь Y - музичний жанр, що представляє різні типи музики, такі як рок, реггі, поп, метал, джаз, хіп-хоп, диско, кантрі, класика та блюз.

Кожна точка на графіку відповідає аудіозапису певної довжини, згрупованому за жанрами. Розподіл точок показує, як розподіляється довжина аудіозаписів у межах кожного жанру. У деяких жанрах, таких як джаз і хіп-хоп, розподіл довжини ширший, що може свідчити про більшу різноманітність довжини треків у цих жанрах. В інших жанрах, таких як реггі та метал, точки більш концентровані, що може свідчити про меншу варіативність довжини треків. Графік допомагає проаналізувати, як змінюється довжина звукозаписів у різних жанрах музики, що важливо для розуміння характеристик даних.

```
X, y = data.drop(['length', 'label'], axis=1).to_numpy(), data['label'].to_numpy()
```

Для навчання моделі машинного навчання дані готуються шляхом поділу на вхідні ознаки X та цільові мітки y. Змінна X містить ознаки у вигляді послідовності. Це дані, які використовуються для навчання моделі. Змінна y містить мітки у вигляді масиву. Це цільові значення, які модель намагається передбачити.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42, stratify=y)
```

Функція train_test_split в бібліотеці Scikit-Learn використовується для розділення даних на навчальну та тестову частини. Таке розділення даних дозволяє навчати модель на одній частині даних (X_train, y_train) і тестувати її роботу на іншій частині (X_test, y_test).

Параметр test_size=0.3 вказує, що 30 % даних буде віднесено до тестового набору, решта 70 % - до навчального набору.

```
from sklearn.base import BaseEstimator, TransformerMixin
import numpy as np
```

Імпортують компоненти, необхідні для створення власних трансформаторів властивостей у бібліотеці Scikit-Learn. Ці імпортовані компоненти використовуються для створення власних класів трансформації властивостей, які можна інтегрувати у ваш конвеєр даних.

```
class PolynomialTransformer(BaseEstimator, TransformerMixin):
    def __init__(self, degree=2, include_bias=True):
        self.degree = degree
        self.include_bias = include_bias
    def fit(self, X, y=None):
        return self
    def transform(self, X):
        if not isinstance(X, np.ndarray):
            X = np.array(X)
        powers = [X**i for i in range(1, self.degree + 1)]
        X_transformed = np.hstack(powers)
        if self.include_bias:
            bias = np.ones((X.shape[0], 1))
            X_transformed = np.hstack([bias, X_transformed])
        return X_transformed
```

Цей трансформер можна використовувати для створення поліноміальних ознак з вхідних даних, що може бути корисним для задач регресії або класифікації, де нелінійні залежності між ознаками та мітками є важливими. Визначають клас PolynomialTransformer, який є власним трансформером ознак для створення поліноміальних ознак з даних.

```
pd.DataFrame(PolynomialTransformer(degree=3,
include_bias=False).fit_transform(X_test))
```

	0	1	2	3	4	5	6	7	8	9	...	161	162	163	164
0	0.395407	0.090621	0.116579	0.001520	1777.839586	330193.868576	2173.800933	151560.984240	3821.177641	1.441895e+06	...	-0.286754	5.099827e+04	-339.188835	173339.726021
1	0.250492	0.090480	0.110509	0.003491	1336.153705	79907.295110	1554.249504	37204.779247	2751.820122	4.866422e+05	...	-3.059470	1.517499e+06	0.485068	277531.435494
2	0.435302	0.082635	0.143163	0.001065	2789.486477	373290.877657	2795.819855	173281.566427	6113.356305	1.839723e+06	...	-4.039685	5.079470e+04	18.124845	33004.653373
3	0.322393	0.086365	0.080559	0.001165	1481.568587	283211.662057	1412.025522	55694.215657	2657.352145	7.853093e+05	...	-262.485117	1.767412e+05	-24.131276	302353.102406
4	0.410039	0.088842	0.139203	0.002202	2026.298986	575184.204476	2398.102837	155401.937825	4462.743857	3.048074e+06	...	0.025811	1.001317e+05	-12.820161	65664.406573
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
295	0.370623	0.080463	0.075741	0.000366	3029.382383	215849.559866	2829.048804	64688.064353	6201.978842	1.046685e+06	...	-1.686926	9.243883e+04	-1.206907	516738.665495
296	0.305175	0.080065	0.120560	0.000325	2428.928767	97719.368632	2286.710780	24997.757809	5064.932456	2.318504e+05	...	279.434935	6.340662e+04	-1276.753138	48009.474512
297	0.321170	0.087852	0.055436	0.001231	857.638582	285170.859067	1238.029736	170818.132853	1443.765464	1.566617e+06	...	-93.564809	8.163043e+04	-1.156350	264009.721295
298	0.276245	0.097367	0.101321	0.003951	1309.382408	756062.612347	1883.042903	284143.663735	2763.186256	4.687547e+06	...	-3865.342490	1.621458e+05	-1315.557310	231297.681932
299	0.448050	0.081806	0.206921	0.002887	2722.744188	432474.935900	2569.340181	88645.239090	5471.916925	1.674843e+06	...	-0.752310	7.618457e+04	-0.974074	301883.693732

За допомогою класу PolynomialTransformer з параметрами degree=3 та include_bias=False створюється набір даних з поліноміальними функціями, отриманими з тестового набору даних X_test. Результатом є кадр даних, що містить поліноміальні характеристики з тесту X_test. Кожен стовпець набору даних відповідає одній з поліноміальних функцій. Таким чином, можна побачити, як виглядає нова функція після перетворення: Метод fit_transform застосовується до X_test для створення нової функції. Результат перетворення перетворюється у фрейм даних для зручного маніпулювання даними, що дозволяє відображати та аналізувати нові функції.

Попередня обробка та моделювання

```
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV, StratifiedKFold, cross_val_score
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

Імпортує різні компоненти з бібліотеки Scikit-Learn, які використовуються для створення конвеєрів даних, пошуку гіперпараметрів, оцінки моделей і вибору функцій. Конвеєри використовуються для створення потоків послідовних даних. Конвеєри можуть об'єднувати кілька етапів обробки даних в один об'єкт, GridSearchCV використовується для пошуку найкращих гіперпараметрів моделі за допомогою перехресної перевірки. Він автоматично тренує модель з різними комбінаціями гіперпараметрів і вибирає ту, яка дає найкращі результати.

stratifiedKFold використовується для розбиття даних на складки для перехресної перевірки. Він повертає оцінку для кожного розшарування.

StandardScaler використовується для масштабування ознак до стандартного нормального розподілу. Це допомагає покращити продуктивність багатьох алгоритмів машинного навчання. PolynomialFeatures використовується для створення поліноміальних ознак шляхом масштабування вхідних ознак до заданого степеня.

LogisticRegression - реалізація логістичної регресії для задач бінарної та багатокласової класифікації; SelectKBest використовується для вибору найкращих ознак на основі заданої оцінки. accuracy_score використовується для обчислення точності моделі. confusion_matrix використовується для створення матриці помилок, яка показує, скільки разів кожен клас було правильно або неправильно передбачено. classification_ report використовується для створення матриці помилок, яка показує, скільки разів кожен клас був правильно або неправильно передбачений. report використовується для створення звіту про класифікацію, який містить такі метрики, як точність, повнота та метрики F1 для кожного класу.

Ці компоненти дозволяють створювати конвеєри даних і моделі машинного навчання, автоматично підбирати гіперпараметри та оцінювати продуктивність моделі.

```
scaler = StandardScaler()
poly_features = PolynomialFeatures(degree=2, include_bias=False)
feature_selector = SelectKBest(f_classif, k=500)
```

Три об'єкти створюються для обробки ознак, які можуть бути використані в машинному навчанні. Об'єкт scaler використовується для масштабування ознак. Ознаки стандартизуються таким чином, що середнє значення ознак дорівнює 0, а стандартне відхилення - 1. Об'єкт poly_features піднімає вхідні ознаки в певному порядку для створення поліноміальних ознак. Об'єкт feature_selector використовується для відбору найкращих ознак. Цей метод допомагає зменшити розмірність даних, вибираючи лише ті ознаки, які найбільше пов'язані з цільовою змінною.

```
def make_pipeline(model):
```

```
    return Pipeline([
        ('poly', poly_features),
        ('scaler', scaler),
        ('feat_select', feature_selector),
        ('model', model),
    ])
```

Функція make_pipeline створює конвеєр обробки даних і навчання моделі за допомогою бібліотеки Scikit-Learn. Model - модель машинного навчання, яка використовується в конвеєрі. Конвеєр автоматично передає вихідні дані одного кроку на вхід наступного кроку, таким чином зменшуючи ймовірність помилок. Конвеєри можна легко використовувати в поєднанні з перехресною перевіркою і пошуком за гіперпараметрами, наприклад, GridSearchCV. Ця функція дозволяє створювати конвеєри з різними моделями, що корисно для експериментів з різними алгоритмами машинного навчання.

```
def fit_and_evaluate(clf, X_train=X_train, y_train=y_train, X_test=X_test,
y_test=y_test):
    print(type(clf['model']).__name__)
    clf.fit(X_train, y_train)
    print("Classification report on train set\n",
classification_report(y_true=y_train, y_pred=clf.predict(X_train)))
    print("Classification report on test set\n", classification_report(y_true=y_test,
y_pred=clf.predict(X_test)))
```

Функція fit_and_evaluate призначена для навчання моделі на тренувальному наборі даних та оцінки її роботи як на тренувальному, так і на тестовому наборах. Вона корисна для оцінки якості моделі та порівняння її роботи на навчальних і тестових наборах. clf - конвеєр або модель для навчання та оцінки; X_train, y_train - навчальні ознаки та мітки; X_test, y_test - тестові ознаки та мітки. Використовуйте classification_report для створення звіту про класифікацію для навчального набору даних. Цей звіт включає такі метрики, як точність, повнота і метрики F1 для кожного класу і генерує звіт про класифікацію для навчального набору. Аналогічно, використовують classification_report для створення звіту про класифікацію для тестового набору даних і відображають його в тестовому наборі даних.

```
log_clf = make_pipeline(LogisticRegression(class_weight='balanced', max_iter=10000,
penalty='elasticnet', solver='saga', l1_ratio=1))
fit_and_evaluate(log_clf)
```

Створення та оцінка логістичної регресії в конвеєрі. Для цього за допомогою функції make_pipeline створюється конвеєр, що включає різні етапи попередньої обробки та моделювання. Створюється модель логістичної регресії; функція fit_and_evaluate(log_clf) навчає модель на навчальних даних і оцінює її на тестових даних. Метод model-fit навчає модель на навчальних даних. Після навчання модель оцінюється на тестових даних за допомогою таких метрик, як матриці точності та невизначеності.

3.2. Результати роботи інтелектуальної системи

Представлено звіт про класифікацію classification report для моделі Logistic Regression, яка була натренована на наборі даних для класифікації музичних жанрів.

```
LogisticRegression
Classification report on train set
              precision    recall  f1-score   support

       blues       0.92      0.97      0.94        70
   classical       0.99      1.00      0.99        70
     country       0.97      0.96      0.96        70
       disco       0.94      0.93      0.94        70
      hiphop       0.97      0.97      0.97        70
        jazz       0.99      0.99      0.99        70
       metal       0.99      0.97      0.98        70
         pop       0.94      0.96      0.95        70
      reggae       0.93      0.94      0.94        70
        rock       0.91      0.86      0.88        70

    accuracy                           0.95       700
   macro avg       0.95      0.95      0.95       700
weighted avg       0.95      0.95      0.95       700
```

Перша колонка показує музичні жанри, які модель може розпізнати: блюз, класика, кантрі, диско, хіп-хоп, джаз, метал, поп, реггі та рок. Наступні стовпчики показують метрики оцінки якості для цієї моделі: Точність - показує частку об'єктів, передбачених для даного класу, які дійсно належать до цього класу; Впізнаваність - показує частку всіх реальних об'єктів у класі, які модель правильно розпізнає. F1 Score - середнє гармонійне значення точності та запам'ятовування. Підтримка - кількість зразків для кожного типу (по 70 для кожного класу, всього 700). Загальна точність моделі становить 95%, що свідчить про високий рівень правильного передбачення.

Найкраще класифіковані жанри:

- classical (precision = 0.99, recall = 1.00, f1-score = 0.99);
- jazz (precision = 0.99, recall = 0.99, f1-score = 0.99);
- metal (precision = 0.99, recall = 0.97, f1-score = 0.98).

Найгірше класифікований жанр:

- rock (precision = 0.91, recall = 0.86, f1-score = 0.88) – модель частіше помиляється в розпізнаванні цього жанру.

Модель добре справляється з класифікацією жанрів, особливо для таких стилів, як classical, jazz, metal. Однак є труднощі з жанром rock, де recall найнижчий (0.86), що означає, що модель іноді не розпізнає всі екземпляри цього жанру.

У звіті про класифікацію на тестовому наборі даних наведено метрики оцінки якості моделі на тестовому наборі даних (300 зразків). Загальна точність accuracy 77 % – модель правильно класифікує 77 % тестових зразків.

```
Classification report on test set
              precision    recall  f1-score   support

       blues       0.85      0.73      0.79        30
   classical       0.97      0.97      0.97        30
     country       0.85      0.73      0.79        30
       disco       0.70      0.63      0.67        30
      hiphop       0.71      0.67      0.69        30
        jazz       0.80      0.93      0.86        30
       metal       0.84      0.87      0.85        30
         pop       0.81      0.87      0.84        30
      reggae       0.58      0.70      0.64        30
        rock       0.66      0.63      0.64        30

    accuracy                           0.77       300
   macro avg       0.78      0.77      0.77       300
weighted avg       0.78      0.77      0.77       300
```

Жанри з найкращою класифікацією:

- classical (точність 97 %, повнота 97 %) – майже ідеально розпізнається;
- jazz (точність 80 %, повнота 93 %) – добре розпізнається, хоча бувають помилки;
- metal та pop (F1-score = 85%) – також добре класифікуються.

Жанри, що викликають труднощі:

- reggae (точність 58 %, повнота 70 %) – модель часто плутає цей жанр з іншими;

- rock (точність 66 %, повнота 63 %) – є проблеми з правильним розпізнаванням;
- disco (точність 70 %, повнота 63 %) – модель має труднощі з виявленням усіх випадків цього жанру.

Різниця між тренувальним і тестовим набором: на тренувальному наборі точність 95 %, а на тестовому 77 %. Це свідчить про перенавчання, модель запам'ятала навчальні дані, але гірше узагальнює на нових зразках. Жанри reggae, rock, disco потребують покращення розпізнавання, classical найкраще класифікований жанр.

```
from sklearn.svm import LinearSVC
svc = make_pipeline(
    LinearSVC(dual='auto', C=0.08, max_iter=10000)
)
fit_and_evaluate(svc)
```

Моделі створюються за допомогою лінійних машин опорних векторів SVM та оцінюється їхня продуктивність. Знову ж таки, функція make_pipeline використовується для об'єднання різних етапів попередньої обробки та моделювання в єдиний конвеєр. Функція fit_and_evaluate(svc) навчає модель на навчальних даних і оцінює її на тестових даних. В результаті можна отримати метрику оцінки моделі.

```
LinearSVC
Classification report on train set
              precision    recall  f1-score   support

       blues       1.00      0.99      0.99        70
   classical       0.99      1.00      0.99        70
     country       0.96      0.99      0.97        70
       disco       0.96      0.97      0.96        70
      hiphop       1.00      0.97      0.99        70
        jazz       1.00      0.99      0.99        70
       metal       0.99      0.99      0.99        70
         pop       0.99      0.99      0.99        70
      reggae       1.00      0.99      0.99        70
        rock       0.94      0.96      0.95        70

    accuracy                           0.98       700
   macro avg       0.98      0.98      0.98       700
weighted avg       0.98      0.98      0.98       700
```

Звіт класифікації для LinearSVC на тренувальному наборі показує, як модель Linear Support Vector Classifier працює на тренувальному наборі з 700 зразків.

Загальна точність accuracy 98 % – модель класифікує майже всі тренувальні зразки правильно. Середні метрики: precision 0.98, recall 0.98, F1-score 0.98

Отримали дуже високу точність на тренувальному наборі 98 %. Модель добре розпізнає всі жанри, навіть ті, які складніше класифікувати (rock, disco, country). Майже ідеальна класифікація жанрів classical, jazz, metal, reggae, pop: F1-score = 0.99 або навіть 1.00. Модель чудово розпізнає ці жанри. Rock та country мають трохи нижчі показники (F1-score = 0.95-0.97). Це може означати, що вони мають схожі характеристики з іншими жанрами. Модель добре розпізнає більшість жанрів, але rock та country можуть викликати складнощі.

```
Classification report on test set
              precision    recall  f1-score   support

       blues       0.88      0.77      0.82        30
   classical       0.94      0.97      0.95        30
     country       0.77      0.80      0.79        30
       disco       0.64      0.47      0.54        30
      hiphop       0.79      0.73      0.76        30
        jazz       0.80      0.93      0.86        30
       metal       0.81      0.83      0.82        30
         pop       0.79      0.87      0.83        30
      reggae       0.61      0.73      0.67        30
        rock       0.59      0.53      0.56        30

    accuracy                           0.76       300
   macro avg       0.76      0.76      0.76       300
weighted avg       0.76      0.76      0.76       300
```

Звіт класифікації для LinearSVC на тестовому наборі демонструє, як модель LinearSVC працює на тестових даних (300 зразків, по 30 на кожен жанр). Загальна точність accuracy 76 %. Середні метрики: precision 0.76, recall 0.76, F1-score 0.76.

Точність значно знизилась у порівнянні з тренувальним набором, на тренувальному наборі 98 %, на тестовому наборі 76 %. Це свідчить про перенавчання, модель запам'ятала тренувальні дані, але не узагальнює їх добре на нових зразках.

Жанри з високою точністю: classical 95%, jazz 86 %, metal 82 %, blues 82 %. Ці жанри модель розпізнає добре, що може означати їхні чіткі відмінності від інших. Жанри з низькою точністю: disco 54 %, rock 56 %, reggae 67 %. Disco та rock мають найгірші показники. Це означає, що вони часто плутаються з іншими жанрами.

Низький показник recall для disco 47 %. Модель пропускає багато треків цього жанру, disco часто класифікується неправильно.

LinearSVC на тренувальних даних показав 98 % точності, а на тестових 76 %, що вказує на перенавчання. Жанри classical, jazz і metal модель розпізнає добре, а disco, rock і reggae погано.

```
from sklearn.ensemble import GradientBoostingClassifier
gb_clf = make_pipeline(
    GradientBoostingClassifier(
        learning_rate=0.01,
        n_estimators=100,
        max_depth=3,
        subsample=0.9
    )
)
fit_and_evaluate(gb_clf)
```

Створюють модель градієнтного бустінгу для класифікації; використовуйте make_pipeline для об'єднання декількох етапів попередньої обробки та моделі в один конвеєр; GradientBoostingClassifier() - модель класифікації на основі градієнтного бустінгу. max_depth=3 - максимальна глибина для кожного дерева. subsample=0.9 - частка вибірок, яку кожне дерево використовує для навчання. Значення 0.9 означає, що кожне дерево навчається на 90% випадкових вибірок, взятих з даних. fit_and_evaluate(gb_clf) використовується для навчання моделі на даних та її оцінки на тестових даних.

```
GradientBoostingClassifier
Classification report on train set
              precision    recall  f1-score   support

       blues       0.99      0.96      0.97        70
   classical       1.00      0.99      0.99        70
     country       0.93      0.99      0.96        70
       disco       0.89      0.97      0.93        70
      hiphop       0.94      0.94      0.94        70
        jazz       0.93      1.00      0.97        70
       metal       0.99      0.96      0.97        70
         pop       0.98      0.93      0.96        70
      reggae       0.96      0.91      0.93        70
        rock       0.93      0.89      0.91        70

    accuracy                           0.95       700
   macro avg       0.95      0.95      0.95       700
weighted avg       0.95      0.95      0.95       700
```

Основні результати звіту класифікації моделі GradientBoostingClassifier на тренувальному наборі. Загальна точність accuracy 95 %. Середні метрики: precision 0.95, recall 0.95, F1-score 0.95. Висока точність 95 %, модель добре навчилася розпізнавати жанри на тренувальному наборі.

Класична музика classical класифікується ідеально: precision = 1.00, recall = 0.99. Це свідчить, що класична музика має чіткі особливості, які модель добре розпізнає. Найгірше розпізнається rock (F1-score = 0.91) та reggae (F1-score = 0.93). Recall для rock = 0.89, що означає, що деякі пісні rock класифікуються неправильно. Reggae має recall = 0.91, що означає, що модель трохи плутає цей жанр.

Основні результати звіту класифікації GradientBoostingClassifier на тестовому наборі. Загальна точність accuracy 72 %. Середні метрики: precision 0.73, recall 0.72, F1-score 0.72. Отримали суттєве падіння точності на тестових даних 72 % порівняно з тренувальними 95 %. Модель добре запам’ятала тренувальні дані, але гірше працює на нових прикладах.

Жанри з високою точністю precision: classical 0.88, metal 0.92, jazz 0.76, pop 0.80. Це означає, що модель добре розпізнає чітко визначені жанри. Жанри з найгіршою точністю: blues 0.73 → 0.53 recall, rock 0.57 precision, reggae 0.72 → 0.60 recall. Blues та rock плутаються найбільше. Це може бути через схожість характеристик. Classical має найвищий F1-score 0.90, що вказує на його найкращу розпізнаваність.

```
log_clf.fit(X, y)
svc.fit(X, y)
```

Виконується навчання двох моделей логістичної регресії log_clf та лінійної підтримки векторних машин svc, використовуючи весь набір даних X та y. Метод fit навчає модель логістичної регресії (log_clf) на всіх даних. X - матриця ознак features, що містить вхідні дані (характеристики музичних треків). y - вектор міток labels, що містить класи (музичні жанри). Логістична регресія вчиться знаходити оптимальні коефіцієнти для кожної ознаки в X, щоб передбачити класи в y.

Метод svc.fit(X, y) виконує навчання лінійної підтримки векторної машини svc на тих самих даних. В SVM модель намагається знайти гіперплощину, яка найкраще

розділяє дані на класи. Є дві різні моделі log_clf та svc, кожна з них буде навчатися на тих самих даних. Це дозволяє порівняти їх ефективність після тренування на одному й тому ж наборі даних.

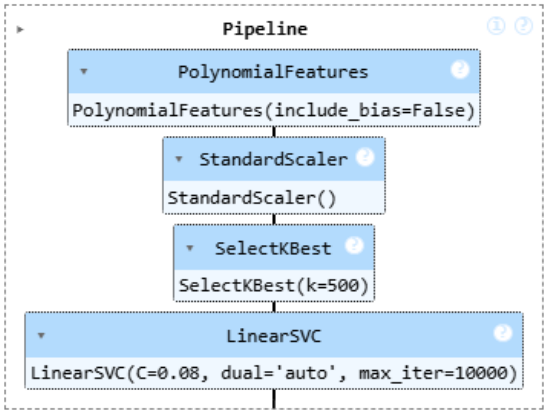


Рисунок 3.2 – Конвеєр обробки даних та побудови моделі машинного навчання

Результатом є конвеєр даних і машинного навчання. Він включає кілька кроків для підготовки даних і побудови моделей машинного навчання.

PolynomialFeatures() використовується для впорядкування всіх ознак і створення нових ознак; StandardScaler() стандартизує ознаки так, щоб їхнє середнє значення дорівнювало 0, а стандартне відхилення - 1. SelectKBest (k=500) використовується для вибору 500 найкращих елементів на основі певних критеріїв. Це допомагає зменшити розмірність даних і покращити продуктивність моделі.LinearSVC () - це останній крок, на якому будується лінійна модель класифікатора опорних векторів. Цей конвеєр є типовим прикладом підготовки даних і побудови моделі для задачі класифікації.

```
import joblib
joblib.dump(log_clf, 'output/log_clf_wl_full.joblib')
joblib.dump(svc, 'output/linSVC_wl_full.joblib')
```

Збереження моделей машинного навчання логістичної регресії та лінійних опорних векторів у файл для подальшого використання. Метод joblib.dump() використовується для збереження моделі логістичної регресії log_clf у файл. Це дозволяє зберегти модель після навчання і використовувати її в майбутньому без необхідності перенавчання. Модель зберігається у файлі output/log_clf_wl_full.joblib; папка output повинна існувати, інакше її потрібно створити.

Той самий процес виконується для лінійної машинної моделі опорних векторів svc. joblib є більш ефективним, ніж стандартний модуль pickle, для зберігання великих об'єктів, таких як моделі машинного навчання. Після збереження моделей у файлі їх можна завантажити за допомогою joblib.load(), щоб зробити більше прогнозів на нових даних без необхідності перенавчання. Потім ці моделі можна використовувати для прогнозування нових даних.

3.3. Розроблення інтерфейсу інтелектуальної системи

```
import streamlit as st
import librosa
import librosa.display
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
from model_inference import predict_genre, predict
import models_performance
```

Імпортують необхідні бібліотеки, створіть графічний інтерфейс та виконайте деяку обробку та моделювання аудіо Streamlit - бібліотека для створення веб-додатків на Python з акцентом на машинне навчання та швидке розгортання аналізу даних. Librosa - бібліотека для обробки аудіо на Python, яка широко використовується для аналізу звуків, музичних жанрів та отримання аудіо характеристик. librosa.display - спеціальний модуль Librosa для відображення результатів аналізу аудіо (спектрограма, mel-спектрограма). Це модуль для візуалізації аудіофайлів у графічному інтерфейсі.

NumPy - бібліотека для роботи з масивами та Matplotlib - бібліотека для побудови графіків; Matplotlib використовується для візуалізації графіків, спектрограм та інших даних. Було імпортовано три функції: predict_genre - функція для прогнозування музичних жанрів, predict - функція для виконання прогнозів у моделях машинного навчання та models_performance - модуль для оцінки продуктивності моделі (метрики точності, матриця невизначеності).

```
@st.cache_data
def load_file(audio_file):
    return librosa.load(uploaded_file, sr=22050)
```

Завантажує аудіофайл і зберігає його у вигляді послідовності чисел. Ці значення можна використовувати для подальшої обробки або прогнозування. Завдяки кешу один і той самий файл можна зчитувати кілька разів без повторної обробки, що прискорює роботу програми.

```
@st.cache_data
def display_signal(y):
    fig, ax = plt.subplots(figsize=(10, 4))
    librosa.display.waveshow(y, sr=sr, ax=ax, color="#1f77b4")
    ax.set_title("Аудіосигнал", fontsize=15)
    ax.set_xlabel("Час (с)")
    ax.set_ylabel("Амплітуда")
    st.pyplot(fig)
```

Візуалізуйте аудіосигнали у вигляді кривих, будуйте графіки за допомогою бібліотек Librosa і Matplotlib, встановлюйте мітки і назви та відображайте їх у Streamlit. Функція кешування дозволяє швидко відображати графіки, навіть якщо один і той самий сигнал відображається багато разів.

У ньому оголошується функція display_signal, яка отримує параметр y - аудіосигнал для візуалізації (масив чисел), де y - результат виклику методу librosa.load(), тобто масив, що містить аудіосигнал.

Метод librosa.display.waveshow використовується для відображення хвильової форми аудіосигналу. y - сам аудіосигнал, sr=sr - частота дискретизації.

```
@st.cache_data
def display_spectogram(y):
    fig, ax = plt.subplots(figsize=(10, 4))
    S = librosa.feature.melspectrogram(y=y, sr=sr)
    librosa.display.specshow(librosa.power_to_db(S, ref=np.max),
                             sr=sr, x_axis='time', y_axis='mel', ax=ax,
cmap='plasma')
    ax.set_title("Мел-спектрограма", fontsize=15)
    st.pyplot(fig)
```

Функція display_spectogram(y) зчитує аудіосигнал і обчислює його mel-спектрограму. mel-спектрограми використовуються для візуалізації частотних складових звуку в часовому ряді і часто застосовуються в задачах класифікації музичних жанрів.

Метод librosa.feature.melspectrogram обчислює мел-спектрограму для аудіосигналу y. y - аудіосигнал, який завантажено раніше за допомогою librosa.load(). sr=sr -частота дискретизації, це кількість вибірок на секунду для аудіосигналу. S - матриця мел-спектрограми, де кожен стовпець містить енергію частотних бінів у певний момент часу.

Метод librosa.display.specshow використовується для візуалізації спектрограми, метод librosa.power_to_db(S, ref=np.max) перетворює амплітуду спектрограми в децибели. Це дозволить зменшити динамічний діапазон сигналу. sr=sr - частота дискретизації.

```
@st.cache_data
def make_prediction(y, chosen_model, prediction_type='deep'):
    use_model = 'log' if chosen_model == 'LogisticRegression' else 'svc'
    y_trimmed, _ = librosa.effects.trim(y)
    start = datetime.now()
    if prediction_type == 'global':
        genre = predict(y_trimmed, sr, use_model=use_model)
        output = (genre, [genre])
    else:
        output = predict_genre(y_trimmed, sr, use_model=use_model)
    st.write(f"Час виконання: {(datetime.now() - start).total_seconds():.0f} секунд")
    return output
```

Функція make_prediction() дозволяє користувачеві завантажити аудіофайл і вибрати модель класифікації (логістична регресія або SVC) і тип прогнозу (загальний або детальний). Прогнозування здійснюється за допомогою попередньо навченої моделі.

choose_model - обрана користувачем модель прогнозування (логістична регресія або SVC) prediction_type - тип прогнозу; якщо обрано модель логістичної регресії, log присвоюється use_model svc_type - тип прогнозу. Якщо вибрано SVC, присвоюється значення svc. Це дозволяє розрізняти моделі, що використовуються для прогнозування, залежно від вибору користувача. Результатом є результат прогнозування, що містить тип або список типів.

```
st.markdown("""
    <style>
    .main-title {
        text-align: center;
```

```
        font-size: 3rem;
        color: #00c4ff;
        font-weight: bold;
        margin-bottom: 0;
    }
    .subtitle {
        text-align: center;
        font-size: 1.5rem;
        color: #dddddd;
        margin-top: 0;
    }
    </style>
    <h1 class="main-title">♪ Класифікація музичних жанрів ♪</h1>
    <h2 class="subtitle">Перетворіть вашу музику у визначений жанр!</h2>
    """, unsafe_allow_html=True)
```

Створюють стилізовані заголовки на сторінці. Заголовки мають бути великими, жирними та синіми, а підзаголовки - маленькими, світло-сірими та вирівняними по центру. Таким чином, користувачі з більшою ймовірністю зосередяться на назві та головному призначенні додатку.

```
chosen_model = st.sidebar.selectbox("Виберіть модель для передбачення",
options=['LogisticRegression', 'LinearSVC'])
prediction_type = st.sidebar.radio(
    "Тип передбачення",
    options=['samples', 'global'],
    index=0,
    format_func=lambda x: "Глобальні передбачення" if x == 'global' else
"Передбачення за зразками",
)
with st.sidebar.expander("Про тип передбачень"):
    st.write(
        "Оберіть _:blue[Передбачення за зразками]_ для детального аналізу, сегмент за
сегментом,"
        "або _:red[Глобальні передбачення]_ для загального аналізу аудіо.\n"
        "Передбачення за сегментами займають менше часу."
    )
```

st.sidebar.selectbox - це елемент інтерфейсу Streamlit, який дозволяє користувачеві вибрати опцію зі списку. Користувач обирає одну з двох моделей для прогнозування: selected_model зберігає обрану користувачем опцію.

st.sidebar.radio - елемент інтерфейсу для вибору одного варіанту з кількох, але з можливістю вибору тільки одного варіанту одночасно. Користувач вибирає між двома типами передбачень:

- samples - передбачення для кожного окремого сегмента аудіофайлу (детальне передбачення).
- global - глобальне передбачення для всього аудіофайлу (загальний аналіз).

st.sidebar.expander - розгортається меню, яке дозволяє показати додаткову інформацію за потреби. Користувач може натискати на заголовок, щоб розгорнути або сховати деталі. В даному випадку текст описує два типи передбачень:

Передбачення за зразками - означає аналіз сегментів аудіофайлу окремо. Це може бути корисно для швидшого виконання і детальнішого аналізу.

Глобальні передбачення - аналіз всього аудіофайлу в цілому, що може займати більше часу, але дає загальний результат.

Сегментне передбачення відбувається швидше, оскільки кожен сегмент обробляється окремо. Глобальне передбачення займає більше часу, але дає загальний результат для всієї аудіодоріжки.

У боковій панелі з'являться:

- selectbox, де користувач вибирає модель для передбачення;
- radio, де вибирається тип передбачення (за зразками або глобальне);
- розгортається меню з поясненням для кожного типу передбачення, що допомагає користувачеві краще зрозуміти, як працюють різні режими аналізу аудіофайлів.

Це дозволяє зробити інтерфейс більш інтуїтивним та зручним для користувачів, допомагаючи їм вибирати відповідні параметри для аналізу музичних жанрів.

```
uploaded_file = st.file_uploader("Завантажте аудіофайл для передбачення (мінімум 30 секунд):", type=["wav", "mp3", "ogg", "flac", "aac", "aiff"])
```

Створюють елемент інтерфейсу для завантаження аудіофайлу за допомогою бібліотеки Streamlit. st.file_uploader - віджет у Streamlit, який дозволяє користувачам завантажувати файли. У цьому випадку елемент дозволяє завантажувати аудіофайли різних форматів.

"Завантажте аудіофайл для передбачення": текст, який з'являється на кнопці завантаження, пояснюючи користувачу, що потрібно завантажити файл для передбачення музичного жанру.

type=["wav", "mp3", "ogg", "flac", "aac", "aiff"] - список дозволених форматів файлів, які можна завантажити. Користувач може вибрати один з таких форматів:

- wav - формат аудіофайлу без стиснення;
- mp3 - популярний формат з стисненням;
- ogg, flac, aac, aiff — інші аудіоформати, що підтримуються.

Якщо користувач спробує завантажити файл в іншому форматі, система не дозволить йому це зробити. Користувач натискає кнопку, щоб завантажити файл. Коли файл завантажено, його вміст зберігається у змінній uploaded_file і може бути використаний для вгадування жанру музики. Це корисно для користувача, який може завантажити аудіофайл, а програма обробить цей файл, щоб визначити тип музики.

```
if uploaded_file is not None:
    try:
        y, sr = load_file(uploaded_file)
        st.audio(uploaded_file)
        st.subheader("🎶 Візуалізація аудіосигналу")
        display_signal(y)
        st.subheader("🎛 Спектрограма")
        display_spectogram(y)
        with st.spinner("Аналіз характеристик аудіо"):
            predicted_genre, predictions = make_prediction(y, chosen_model,
prediction_type)
```

Завантажують аудіофайл, отримують візуалізацію аудіосигнал та спектрограму, а також передбачення жанру музики за допомогою обраної моделі.

Викликається функція load_file, яка завантажує аудіофайл. y - аудіосигнал у вигляді масиву (значення амплітуд). sr - частота дискретизації, яка показує, скільки вимірювань за секунду робиться в аудіофайлі.

З допомогою методу st.audio(uploaded_file) відображають аудіофайл у веб-додатку, дозволяючи користувачеві прослухати завантажений файл. Це дозволяє прослухати музику безпосередньо через інтерфейс Streamlit.

Встановлюють підзаголовок на сторінці: Візуалізація аудіосигналу. Це дає зрозуміти користувачеві, що далі буде показано графічне представлення аудіосигналу. Викликається функція display_signal, яка відповідає за побудову графіку для аудіосигналу. Це дозволяє візуалізувати форму хвилі аудіофайлу (амплітуду сигналу в часі).

Встановлюється ще один підзаголовок: "Спектрограма". Цей підзаголовок сигналізує користувачу, що далі буде показано спектрограму аудіофайлу. Викликається функція display_spectogram, яка генерує та показує мел-спектрограму аудіофайлу. Спектрограма є корисним інструментом для аналізу частотних компонентів сигналу.

Викликається функція make_prediction для передбачення жанру музики. y - аудіосигнал. chosen_model - вибрана модель для передбачення (Logistic Regression або LinearSVC). prediction_type - тип передбачення (глобальне або по сегментах).

Результат функції make_prediction повертає predicted_genre - жанр, який було передбачено, predictions - додаткові передбачення для кожного сегмента (якщо передбачення за зразками).

```
            st.markdown(f"""
            <div style="text-align: center; padding: 20px; border-radius: 10px;
background: linear-gradient(45deg, #1e3c72, #2a5298); color: white; margin: 1rem 0;">
                <h3>🎤 Передбачений жанр:</h3>
                <h1 style="font-size: 3rem;"> {predicted_genre}</h1>
            </div>
            """, unsafe_allow_html=True)
```

Відображають результат передбачення жанру музики у вигляді стилізованого блоку HTML за допомогою Streamlit.

```
        st.info("Передбачувані жанри: *blues, classical, country, disco, hiphop,
jazz, metal, pop, reggae, rock* з точністю 85%.")
        st.info("Результати можуть змінюватися залежно від вибраної моделі. Одна
пісня може включати кілька жанрів.")
        st.markdown(f"Використана модель: **:blue[{chosen_model}]**.")
```

Використовують функції Streamlit для додавання інформаційних повідомлень та форматування тексту на сторінці. Метод st.info("...") використовується для

виведення інформаційних повідомлень. Текст всередині буде відображений в блакитному блоці.

У першому повідомленні користувачеві пропонується список жанрів, які може передбачити модель: блюз, класика, кантрі, диско, хіп-хоп, джаз, метал, поп, реггі та рок; у другому повідомленні вказується, що результати можуть відрізнятися залежно від обраної моделі, і що пісня може належати до кількох жанрів одночасно. З'являється текстове повідомлення з назвою моделі, яку користувач обрав для вгадування жанру.

Код інформує користувача про доступні види, точність моделі, можливість того, що результати можуть відрізнятися в залежності від вибору моделі, і дозволяє користувачеві переглянути обрану модель.

```
st.write(predictions)
st.download_button(
    label="📥 Завантажити результати",
    data=str(predictions),
    file_name="результат_жанру.txt",
)
st.success("Аналіз завершено")
```

Код дозволяє користувачам переглядати прогнози видів, завантажувати їх у вигляді текстового файлу та отримувати підтвердження, що аналіз завершено. Це зручний інтерфейс для взаємодії з результатами, що полегшує подальше використання та зберігання даних.

Цей фрагмент додає виведення результатів, функцію завантаження та повідомлення про завершення аналізу. ST. write(predictions) Відображає передбачення змінної на сторінці. Прогнози містять жанрові передбачення, тому відображається або сам прогноз, або список передбачуваних жанрів для аудіофайлу. Streamlit автоматично форматує різні типи даних (списки, рядки) для зручного перегляду.

st.download_button(...) - створюють кнопку для завантаження результатів аналізу. data=str(predictions) - дані для завантаження. file_name="результат_жанру.txt" - назва файлу, який буде завантажено користувачем (текстовий файл результат_жанру.txt). st.success("Аналіз завершено"): виводиться

повідомлення успіху на сторінку, інформуючи користувача про те, що аналіз завершено. Текст відображається в блоці з зеленим фоном, що позначає успішне виконання операції.

```
    except Exception as e:
        st.error(f"Сталася помилка")
        with st.expander("Деталі помилки"):
            st.markdown(f'\n{e}')
```

Обробляють помилки, надаючи користувачеві повідомлення про помилку та можливість переглянути деталі помилки у зручному вигляді для виявлення причин несправностей під час виконання програми.

```
with st.expander("Переглянути більше деталей про ефективність моделей у передбаченні жанрів"):
    models_performance.run()
```

Додають можливість відображати додаткову інформацію про продуктивність моделі при класифікації музичних жанрів. Користувачі можуть розширити цей блок для відображення більш детальних результатів, що допоможе їм оцінити і вибрати найбільш підходящу модель.

with st.expander("Переглянути більше деталей про ефективність моделей у передбаченні жанрів"): створюють інтерактивний елемент, який дозволяє приховувати або розгортати додаткову інформацію. Заголовок "Переглянути більше деталей про ефективність моделей у передбаченні жанрів" буде відображений у вигляді посилання чи кнопки, яку користувач може натискати для розгортання цього блоку з додатковими деталями.

models_performance.run(): виклик функції run() з модуля models_performance. Вона відповідає за обчислення та відображення інформації про ефективність моделей, таких як точність, звіт про класифікацію та інші метрики, які використовуються для оцінки продуктивності моделей.

Цей блок дає можливість користувачеві побачити деталі, такі як порівняння різних моделей класифікації жанрів (LogisticRegression і LinearSVC) на основі результатів їх тестування.

```
st.markdown("""
    <style>
    .footer {
```

```
        position: fixed;
        bottom: 0;
        text-align: center;
        font-size: 0.8rem;
        color: #bbbbbb;
        width: 100%;
    }
```

Метод st.markdown дозволяє вставляти HTML або Markdown контент у Streamlit додаток. Стилі визначені у тегу <style>, який задає CSS для футера.

Для запуску моделі в командному рядку вводять команду streamlit run music.py. Після цього в броузері за адресою http://localhost:8501/ появиться інтерфейс інтелектуальної системи.

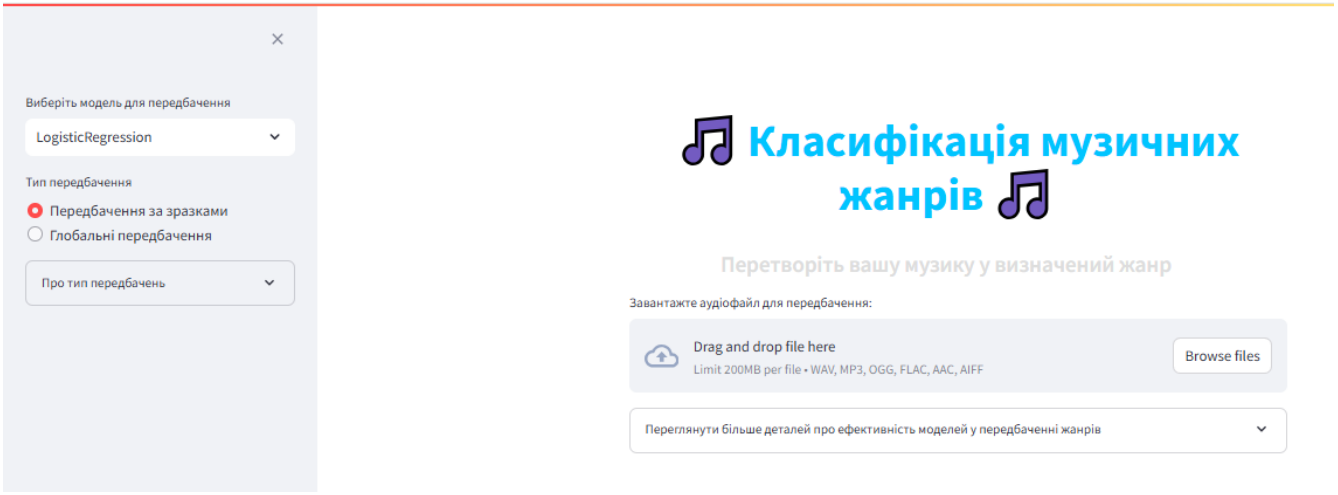


Рисунок 3.3 – Інтерфейс інтелектуальної системи класифікації музичних жанрів

На лівій бічній панелі виберіть модель прогнозування зі спадного списку - LogisticRegression або LinearSVC. За допомогою перемикачів під ним виберіть тип прогнозу: вибірковий або глобальний. Ви можете вибрати вибірковий прогноз для детального аналізу, сегментний прогноз або глобальний прогноз для загального аналізу аудіо. Прогнозування за сегментами займає менше часу.

В центральній частині екрану потрібно завантажити аудіофайл для передбачення музичного жанру. Для цього потрібно натиснути кнопку Browse files і вибрати музичний файл для подальшої класифікації його жанру. Після цього нижче в головному вікні появиться назва файлу, який був завантажений в інтелектуальну систему та віджет для його прослуховування. Нижче від віджету появиться хвильовий графік цього файлу:

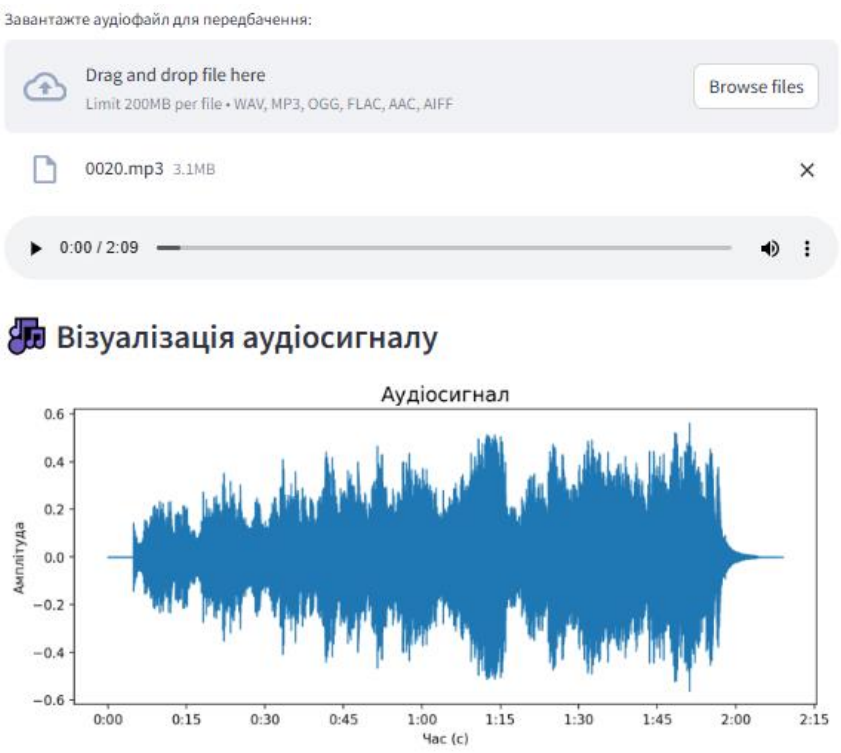


Рисунок 3.4 – Віджет для прослуховування завантаженого музичного файлу та його хвильовий графік

Ще нижче в головному вікні появиться мел-спектрограма досліджуваного музичого файлу, час виконання аналізу (8 секунд) та результат аналізу. Даний музичний файл належатиме до жанру classical:

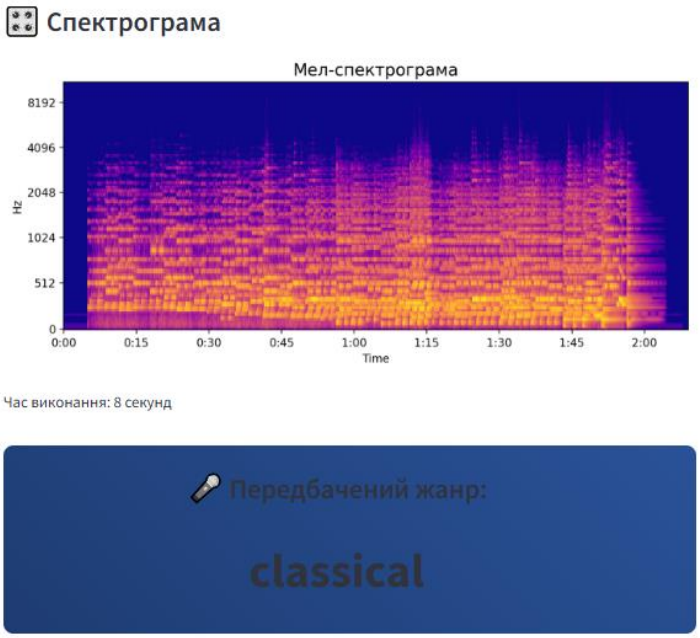


Рисунок 3.5 – Віджет для прослуховування завантаженого музичного файлу та його хвильовий графік

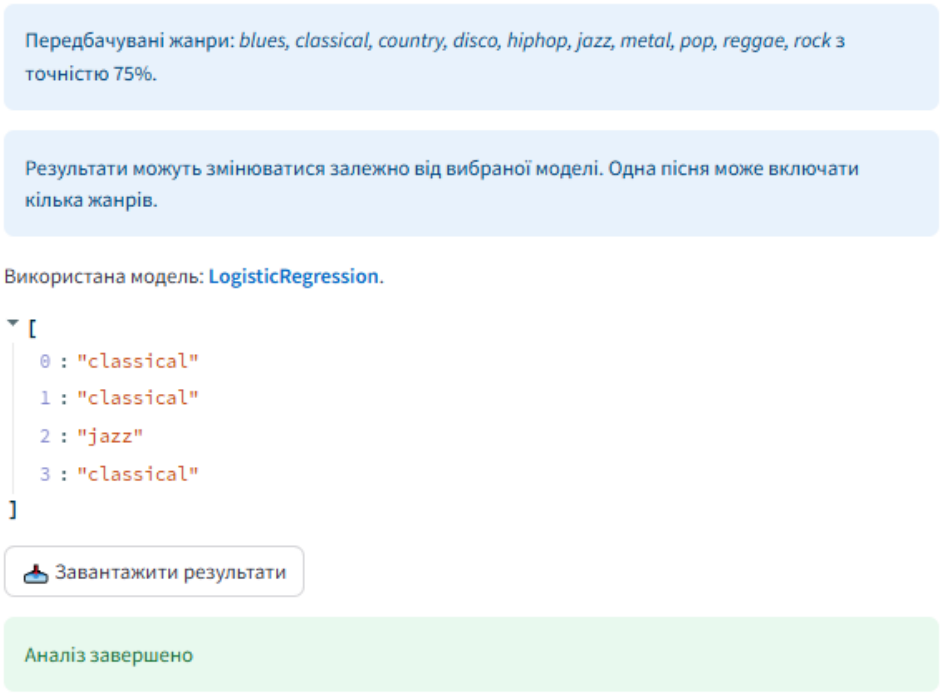
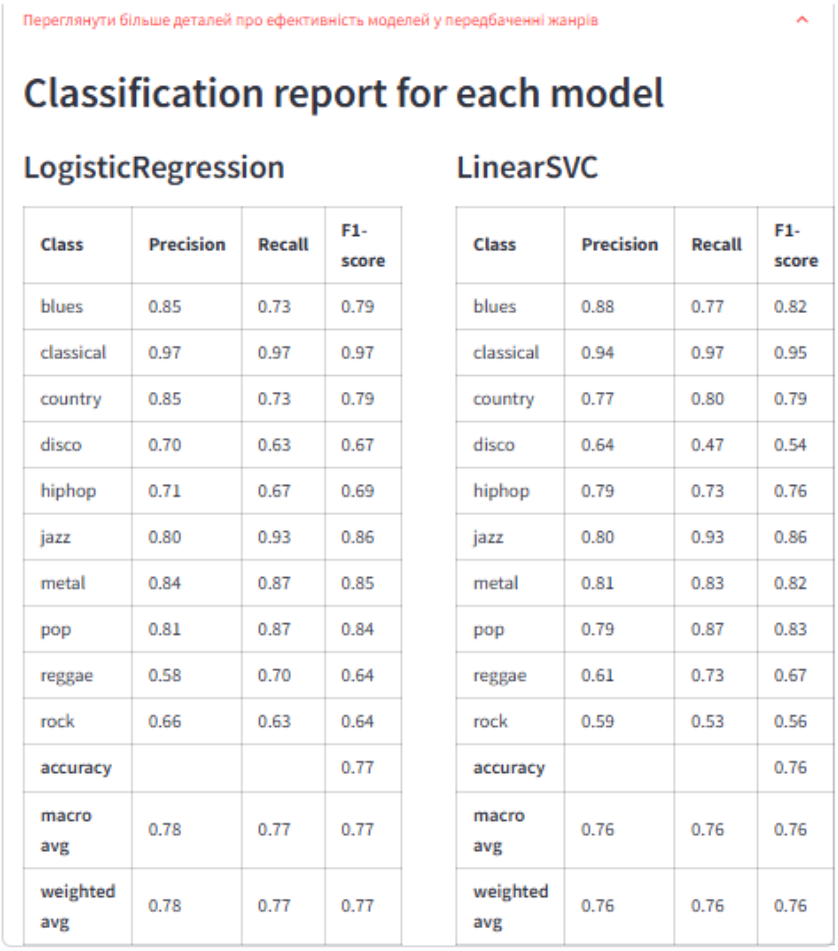


Рисунок 3.6 – Інтерфейс інтелектуальної системи з результатами аналізу музичного файлу

Далі йде інформація про результати аналізу. Модель LogisticRegression, що використовувалася в аналізі, передбачила музичний жанр файлу з точністю до 75%. З'являється повідомлення про те, що аналіз завершено, і ви можете натиснути на кнопку Завантажити результати, щоб зберегти результати аналізу у файлі genre result.txt. У нижній частині головного вікна є випадаючий список для перегляду додаткової інформації про ефективність моделі у визначенні жанру.



Class	Precision	Recall	F1-score
blues	0.85	0.73	0.79
classical	0.97	0.97	0.97
country	0.85	0.73	0.79
disco	0.70	0.63	0.67
hiphop	0.71	0.67	0.69
jazz	0.80	0.93	0.86
metal	0.84	0.87	0.85
pop	0.81	0.87	0.84
reggae	0.58	0.70	0.64
rock	0.66	0.63	0.64
accuracy			0.77
macro avg	0.78	0.77	0.77
weighted avg	0.78	0.77	0.77

Class	Precision	Recall	F1-score
blues	0.88	0.77	0.82
classical	0.94	0.97	0.95
country	0.77	0.80	0.79
disco	0.64	0.47	0.54
hiphop	0.79	0.73	0.76
jazz	0.80	0.93	0.86
metal	0.81	0.83	0.82
pop	0.79	0.87	0.83
reggae	0.61	0.73	0.67
rock	0.59	0.53	0.56
accuracy			0.76
macro avg	0.76	0.76	0.76
weighted avg	0.76	0.76	0.76

Рисунок 3.7 – Інформація про ефективність моделей у передбаченні жанрів

3.4. Характеристики апаратного та програмного забезпечення

Для проектування інформаційної системи використовувався комп’ютер з наступними характеристиками.

Таблиця 3.3 – Характеристики персонального комп’ютера

Процесор	Intel Pentium	Core i5
Материнська плата	Asus	ZY60-24-Premium
ОП	DDR3	4 GB
Відеокарта	Nvidia GeForce	1024 МГц
Жорсткий диск	Kingstone	300 Гб
Монітор	22″	Samsung

В дипломній роботі використано такі програмні продукти.

Таблиця 3.4 – Програмне забезпечення

Операційна система	Windows 10
Середовище розробки	Python, Google Colab
Растровий графічний редактор	Adobe Photoshop CC
Векторний графічний редактор	CorelDraw 2019
Текстовий редактор	Microsoft Word 2016

ВИСНОВКИ

У дипломній роботі розроблено інтелектуальну систему для класифікації музичних жанрів, що використовує сучасні методи машинного навчання. Застосування автоматизованих підходів до класифікації музики значно підвищує точність та ефективність роботи в порівнянні з традиційними методами. Використання набору даних GTZAN дозволило продемонструвати можливості отримання важливих ознак з аудіосигналів, що є важливим етапом у процесі навчання моделей.

Аналіз звукових характеристик, таких як тембр, ритм та гармонія, дав змогу виділити ключові фактори, що впливають на класифікацію різних жанрів. Результати навчання моделей, включаючи логістичну регресію та LinearSVC, показали високу точність, що свідчить про їхню здатність до адаптації на нових даних. Оцінка якості класифікації за допомогою метрик, таких як точність та чутливість, підтвердила ефективність розробленої системи.

Особливу увагу було приділено розробці графічного інтерфейсу, що робить систему зручною для користувачів. Інтерфейс дозволяє безпосередньо завантажувати аудіофайли та отримувати результати класифікації в реальному часі. Це значно розширює можливості системи та її впровадження в практичні музичні сервіси.

Результати даного дослідження можуть бути корисними для музичних платформ, радіостанцій і продюсерських центрів, де автоматизація класифікаційних процесів дозволяє зекономити ресурси та підвищити якість обслуговування. Виконана робота підкреслює важливість інтеграції штучного інтелекту в музиці та відкриває нові горизонти для подальших досліджень у цій галузі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Elbir A., Aydin N. Music genre classification and music recommendation by using deep learning. Electronics Letters. 2020, 56, pp. 627-629.

2. Rajanna A., Aryafar K., Shokoufandeh A., Ptucha R. Deep neural networks: A case study for music genre classification. In proceedings of the 2015 IEEE 14th International conference on machine learning and applications (ICMLA), 2015, pp. 655-660.

3. Tzanetakis G., Cook P. Musical genre classification of audio signals. IEEE Transactions on speech and audio processing. 2002, 10, pp. 293-302.

4. Xu C., Maddage N., Shao X. Musical genre classification using support vector machines. In proceedings of the 2003 IEEE International conference on acoustics, speech, and signal processing, 2003. Volume 5, pp. 429-432.

5. Kour G., Mehan N. Music genre classification using MFCC, SVM and BPNN. International journal of computer applications. 2015, 112, pp. 12-14.

6. Patil N., Nemade M. Music genre classification using MFCC, K-NN and SVM classifier. International journal of computer engineering. 2017, 4, pp. 43-47.

7. Khasgiwala Y., Tailor J. Vision transformer for music genre classification using mel-frequency cepstrum coefficient. In proceedings of the 2021 IEEE 4th International conference on computing, power and communication technologies (GUCON), Kuala Lumpur, 2021, pp. 1-5.

8. Pelchat N., Gelowitz C. Neural network music genre classification. Canadian journal of electrical and computer engineering. 2020, 43, pp. 170-173.

9. Cheng Y., Kuo C. Machine learning for music genre classification using visual mel spectrum. Mathematics, 2022, 10, p. 4427.

10. Jena K., Bhoi S., Mohapatra S., Bakshi S. A hybrid deep learning approach for classification of music genres using wavelet and spectrogram analysis. Neural computing and applications. 2023, pp. 1-26.

11. Zhao H., Zhang C., ZhuB. Self-supervised pretraining with swin transformer for music classification. In proceedings of the ICASSP. IEEE International conference on acoustics, speech and signal processing (ICASSP), Singapore, 2022, pp. 606-610.

12. Silla C., Koerich A., Kaestner C. A machine learning approach to automatic music genre classification. Journal of the brazilian computer society, 2008, 14, pp. 7-18.

13. LeCun Y., Bengio Y., Hinton G. Deep learning. 2015, 521, pp. 436- 444.

14. Bisharad D., Laskar R. Music genre recognition using convolutional recurrent neural network architecture. Expert systems. 2019, 36, pp. 1-13.

15. AbdoliS., Cardinal P., Koerich A. End-to-end environmental sound classification using a 1D convolutional neural network. Expert systems with applications. 2019, 136, pp. 252-263.

16. Wu W., Han F., Song G. Music genre classification using independent recurrent neural network. In proceedings of the 2018 Chinese automation congress (CAC), China, 2018, pp. 192-195.

ДОДАТКИ

ДОДАТОК А

music.ipynb

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


data = pd.read_csv("D:/703/data/features_30_sec.csv", index_col="filename")
data.head()


data.isnull().sum().sum()


data['label'].unique()


data.describe()


plt.scatter(data['length'], data['label'])


X, y = data.drop(['length', 'label'], axis=1).to_numpy(), data['label'].to_numpy()


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42, stratify=y)


from sklearn.base import BaseEstimator, TransformerMixin
import numpy as np

class PolynomialTransformer(BaseEstimator, TransformerMixin):
    def __init__(self, degree=2, include_bias=True):
        self.degree = degree
        self.include_bias = include_bias

    def fit(self, X, y=None):
        return self
```

```
def transform(self, X):
        if not isinstance(X, np.ndarray):
            X = np.array(X)
        powers = [X**i for i in range(1, self.degree + 1)]
        X_transformed = np.hstack(powers)
        if self.include_bias:
            bias = np.ones((X.shape[0], 1))
            X_transformed = np.hstack([bias, X_transformed])
        return X_transformed


pd.DataFrame(PolynomialTransformer(degree=3,
include_bias=False).fit_transform(X_test))


from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV, StratifiedKFold, cross_val_score
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report


scaler = StandardScaler()
poly_features = PolynomialFeatures(degree=2, include_bias=False)
feature_selector = SelectKBest(f_classif, k=500)

def make_pipeline(model):
    return Pipeline([
        ('poly', poly_features),
        ('scaler', scaler),
        ('feat_select', feature_selector),
        ('model', model),
    ])

def fit_and_evaluate(clf, X_train=X_train, y_train=y_train, X_test=X_test,
y_test=y_test):
    print(type(clf['model']).__name__)
    clf.fit(X_train, y_train)
    print("Classification report on train set\n",
    classification_report(y_true=y_train, y_pred=clf.predict(X_train)))
    print("Classification report on test set\n", classification_report(y_true=y_test,
    y_pred=clf.predict(X_test)))
```

```
log_clf = make_pipeline(LogisticRegression(class_weight='balanced', max_iter=10000,
penalty='elasticnet', solver='saga', l1_ratio=1))
fit_and_evaluate(log_clf)


from sklearn.svm import LinearSVC
svc = make_pipeline(
    LinearSVC(dual='auto', C=0.08, max_iter=10000)
)
fit_and_evaluate(svc)


from sklearn.ensemble import GradientBoostingClassifier
gb_clf = make_pipeline(
    GradientBoostingClassifier(
        learning_rate=0.01,
        n_estimators=100,
        max_depth=3,
        subsample=0.9
    )
)
fit_and_evaluate(gb_clf)


log_clf.fit(X, y)
svc.fit(X, y)


import joblib
joblib.dump(log_clf, 'output/log_clf_wl_full.joblib')
joblib.dump(svc, 'output/linSVC_wl_full.joblib')
```

ДОДАТОК Б

music.py

```
import streamlit as st
import librosa
import librosa.display
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
from model_inference import predict_genre, predict
import models_performance

@st.cache_data
def load_file(audio_file):
    return librosa.load(uploaded_file, sr=22050)

@st.cache_data
def display_signal(y):
    fig, ax = plt.subplots(figsize=(10, 4))
    librosa.display.waveshow(y, sr=sr, ax=ax, color="#1f77b4")
    ax.set_title("Аудіосигнал", fontsize=15)
    ax.set_xlabel("Час (с)")
    ax.set_ylabel("Амплітуда")
    st.pyplot(fig)

@st.cache_data
def display_spectogram(y):
    fig, ax = plt.subplots(figsize=(10, 4))
    S = librosa.feature.melspectrogram(y=y, sr=sr)
    librosa.display.specshow(librosa.power_to_db(S, ref=np.max),
                             sr=sr, x_axis='time', y_axis='mel', ax=ax,
cmap='plasma')
    ax.set_title("Мел-спектрограма", fontsize=15)
    st.pyplot(fig)

@st.cache_data
def make_prediction(y, chosen_model, prediction_type='deep'):
    use_model = 'log' if chosen_model == 'LogisticRegression' else 'svc'
    y_trimmed, _ = librosa.effects.trim(y)
    start = datetime.now()
    if prediction_type == 'global':
```

```
        genre = predict(y_trimmed, sr, use_model=use_model)
        output = (genre, [genre])
    else:
        output = predict_genre(y_trimmed, sr, use_model=use_model)
    st.write(f"Час виконання: {(datetime.now() - start).total_seconds():.0f} секунд")
    return output

st.markdown("""
    <style>
    .main-title {
        text-align: center;
        font-size: 3rem;
        color: #00c4ff;
        font-weight: bold;
        margin-bottom: 0;
    }
    .subtitle {
        text-align: center;
        font-size: 1.5rem;
        color: #dddddd;
        margin-top: 0;
    }
    </style>
    <h1 class="main-title">♪ Класифікація музичних жанрів ♪</h1>
    <h2 class="subtitle">Перетворіть вашу музику у визначений жанр</h2>
    """, unsafe_allow_html=True)

chosen_model = st.sidebar.selectbox("Виберіть модель для передбачення",
options=['LogisticRegression', 'LinearSVC'])
prediction_type = st.sidebar.radio(
    "Тип передбачення",
    options=['samples', 'global'],
    index=0,
    format_func=lambda x: "Глобальні передбачення" if x == 'global' else
"Передбачення за зразками",
)
with st.sidebar.expander("Про тип передбачень"):
    st.write(
        "Оберіть _:blue[Передбачення за зразками]_ для детального аналізу, сегмент за
сегментом,"
        "або _:red[Глобальні передбачення]_ для загального аналізу аудіо.\n"
        "Передбачення за сегментами займають менше часу."
    )
```

```
uploaded_file = st.file_uploader("Завантажте аудіофайл для передбачення",
type=["wav", "mp3", "ogg", "flac", "aac", "aiff"])

if uploaded_file is not None:
    try:
        y, sr = load_file(uploaded_file)
        st.audio(uploaded_file)
        st.subheader("🎶 Візуалізація аудіосигналу")
        display_signal(y)
        st.subheader("🎛 Спектрограма")
        display_spectogram(y)
        with st.spinner("Аналіз характеристик аудіо"):
            predicted_genre, predictions = make_prediction(y, chosen_model,
prediction_type)

            st.markdown(f"""
            <div style="text-align: center; padding: 20px; border-radius: 10px;
background: linear-gradient(45deg, #1e3c72, #2a5298); color: white; margin: 1rem 0;">
                <h3>🎤 Передбачений жанр:</h3>
                <h1 style="font-size: 3rem;"> {predicted_genre}</h1>
            </div>
            """, unsafe_allow_html=True)

        st.info("Передбачувані жанри: *blues, classical, country, disco, hiphop,
jazz, metal, pop, reggae, rock з точністю 85%.")
        st.info("Результати можуть змінюватися залежно від вибраної моделі. Одна
пісня може включати кілька жанрів.")
        st.markdown(f"Використана модель: **:blue[{chosen_model}]**.")

        st.write(predictions)
        st.download_button(
            label="📥 Завантажити результати",
            data=str(predictions),
            file_name="результат_жанру.txt",
        )
        st.success("Аналіз завершено")

    except Exception as e:
        st.error(f"Сталася помилка")
        with st.expander("Деталі помилки"):
            st.markdown(f'\n{e}')
```

```
with st.expander("Переглянути більше деталей про ефективність моделей у передбаченні жанрів"):
    models_performance.run()

st.markdown("""
    <style>
    .footer {
        position: fixed;
        bottom: 0;
        text-align: center;
        font-size: 0.8rem;
        color: #bbbbbb;
        width: 100%;
    }
```