НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

КАФЕДРА ІНФОРМАТИКИ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

КУРСОВА РОБОТА

з дисципліни «Аналіз даних в інформаційних системах»

на тему: «Класифікація музичних композицій за жанром на основі їх характеристик»

Студента 2 курсу групи ІП-15

Спеціальності: 121

«Інженерія програмного забезпечення»

Кондрацької Соні Леонідівни

«ПРИЙНЯВ» з оцінкою

_____

доц. Ліхоузова Т.А. / доц. Олійник Ю.О.

_____

Підпис Дата

Київ - 2022 рік

Національний технічний університет України “КПІ ім. Ігоря Сікорського”

Кафедра інформатики та програмної інженерії

Дисципліна Аналіз даних в інформаційно-управляючих системах

Спеціальність 121 "Інженерія програмного забезпечення"

Курс 2 Група ІП-15 Семестр 4

ЗАВДАННЯ

на курсову роботу студента

Кондрацької Соні Леонідівни

1.Тема роботи Класифікація музичних композицій за жанром на основі їх характеристик

2.Строк здачі студентом закінченої роботи 29.05.2023

3. Вхідні дані до роботи методичні вказівки до курсової робота, обрані дані з сайту https://www.kaggle.com/datasets/purumalgi/music-genre-classification?select=train.csv

4.Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

Вступ, постановка задачі, аналіз предметної області, робота з даними, інтелектуальний аналіз, висновки, перелік посилань, додаток А.

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

6.Дата видачі завдання 16.04.2023

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	16.04.2023	
2.	Визначення зовнішніх джерел даних	…	
3.	Пошук та вивчення літератури з питань курсової роботи		
4.	Розробка моделі сховища даних		
4.	Розробка ETL процесів		
5.	Обґрунтування методів інтелектуального аналізу даних		
6.	Застосування та порівняння ефективності методів інтелектуального аналізу даних		
7.	Підготовка пояснювальної записки		
8.	Здача курсової роботи на перевірку		
9.	Захист курсової роботи	02.06.2023	

Студент Кондрацька Соня Леонідівна

(підпис) (прізвище, ім'я, по батькові)

Керівник доц. Ліхоузова Т.А

(підпис) (прізвище, ім'я, по батькові)

Керівник доц. Олійник Ю.О.

(підпис) (прізвище, ім'я, по батькові)

"2" червня 2023 р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 26 сторінок, 19 рисунків, 11 посилань.

Об’єкт дослідження: інтелектуальний аналіз даних.

Предмет дослідження: створення програмного забезпечення, що проводить аналіз даних з подальшим прогнозуванням та графічним відображенням результатів.

Мета роботи: пошук, аналіз та обробка даних, реалізація ПЗ, що використовує отримані дані для подальшого аналізу та прогнозування результату.

Дана курсова робота включає в себе: постановку задачі, аналіз предметної області, роботу з даними, аналіз обраних методів для прогнозування та їх порівняння.

ДАТАСЕТ, ПРОГНОЗУВАННЯ, КЛАСИФІКАЦІЯ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, K-NEAREST NEIGHBORS, DECISION TREE CLASSIFIER, GRADIENT BOOSTING CLASSIFIER, NAIVE BAYES.

Зміст

ВСТУП.....4

1.ПОСТАНОВКА ЗАДАЧІ.....5

2.АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....6

3.РОБОТА З ДАНИМИ.....8
3.1 Опис обраних даних.....8
3.2 Перевірка даних.....8
3.3 Поділ даних.....11

4.ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ.....12
4.1 Обґрунтування вибору методів інтелектуального аналізу даних.....12
4.2 Аналіз отриманих результатів для методу Gradient Boosting Classifier.....14
4.3 Аналіз отриманих результатів для методу Random Forest.....16
4.4 Аналіз отриманих результатів для методу K Neighbors.....17
4.5 Аналіз отриманих результатів для методу Naive Bayes.....19
4.6 Порівняння отриманих результатів методів.....20

ВИСНОВКИ.....22

ПЕРЕЛІК ПОСИЛАНЬ.....23

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ.....25

ВСТУП

У сучасному світі музика відіграє важливу роль у нашому житті, створюючи настрій, впливаючи на емоції та виражаючи різноманітні стилі та жанри. Класифікація музичних композицій за жанром є актуальною задачею, яка може мати широкий спектр застосувань. Від рекомендацій музики для користувачів до аналізу та організації музичних колекцій, правильна класифікація може сприяти поліпшенню музичного досвіду та зручності використання музичних сервісів.

Метою цієї курсової роботи є класифікація музичних композицій за жанром на основі їх характеристик. Для цього дослідження був обраний набір даних, що містить інформацію про різні характеристики музичних композицій, такі як темп, ритм, енергія тощо. За допомогою цих характеристик були створені моделі, для розв’язку поставленої задачі.

У процесі дослідження використовуються різні методи інтелектуального аналізу, такі як Gradient Boosting Classifier, Random Forest Classifier, K Neighbors Classifier та Naive Bayes .

В роботі використовується мова програмування Python та популярні бібліотеки, такі як Pandas, NumPy, scikit-learn і Matplotlib. Ці інструменти допомагають обробляти та аналізувати дані, будувати моделі класифікації та візуалізувати результати.

1.ПОСТАНОВКА ЗАДАЧІ

Виконання даної курсової роботи вимагає виконання декількох задач, а саме: аналізу предметної області; роботи з датасетом: завантаження, дослідження його структури та виправлення наявних помилок; вибір методів для прогнозування та обґрунтування даного вибору; аналіз отриманих результатів кожного з методів та порівняння отриманих результатів ефективності.

Створення застосунку, що поділяє отримані дані на тренувальні та тестові для перевірки декількох методів, у подальшому порівняння ефективності методів.

Для прогнозування буде використано методи Gradient Boosting Classifier, Random Forest Classifier, K Neighbors Classifier та Naive Bayes . Для кожного методу проаналізувати результати та в кінці порівняти їх. Обрати найоптимальніший метод для прогнозування жанру пісні.

Вхідними даними будуть популярність, танцювальність, енергійність, тональність, гучність, режим, мовність, акустичність, інструментальність, жвавість, валентність, темп, тривалість у хв/мс, такт.

2.АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Аналіз музичних композицій за жанром на основі їх характеристик є важливою задачею в області музичного аналізу та музичної індустрії, а також цікавою темою для звичайних людей, які цікавляться музикою. Жанр музики визначає його стиль, настрій, виконавські особливості та аудиторію, до якої він звертається. Точна класифікація музичних композицій за жанром має велике значення для музикознавства, рекомендаційних систем, радіостанцій та потокових платформ, де користувачі можуть шукати музику за жанрами.

Для аналізу необхідні дані з датасету, що містить інформацію про різні характеристики музичних композицій, такі як ритм, темп, тембр, мелодія, гармонія тощо, а також відповідний жанр кожної композиції. Ці дані можуть бути зібрані та збережені у вигляді таблиці або бази даних.

У програмному забезпеченні буде реалізовано такі функціональності:

- Завантаження та дослідження структури датасету: Вивчення ознак та їх значення для класифікації музичних композицій за жанром, перевірка наявності пропущених значень або помилок у даних.
- Використання методів машинного навчання для класифікації: Розробка та навчання моделей класифікації, таких як Gradient Boosting Classifier, Random Forest Classifier, K Neighbors Classifier та Naive Bayes, з використанням навчального датасету.
- Прогнозування жанру музичних композицій: Використання навчених моделей для класифікації нових музичних композицій за їх характеристиками та прогнозування їх жанру.
- Оцінка ефективності методів: Порівняння результатів класифікації різних моделей та визначення найкращого підходу для класифікації музичних композицій за жанром.

- Візуалізація результатів та їх аналіз: Графічне відображення результатів класифікації, аналіз точності моделей.

3.РОБОТА З ДАНИМИ

3.1 Опис обраних даних

Для вирішення поставленної перед нами задачі був обраний датасет «Music Genre Classification», що складається з двох таблиць: “train.csv” з навчальними даними та “submission.csv” з переліком назв жанрів та їх id. Таблиця “train.csv” складається з характеристик 17,996 пісень і з 17 стовпців, а саме: Artist Name, Track Name, Popularity, danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, duration_in min/ms, time_signature, Class.

- Class – вказує на id жанру пісні (від 0 до 10).

3.2 Перевірка даних

Для роботи з даними на мові Python ми використовуємо бібліотеку «pandas».

Для початку ми зчитуємо дані з файлу, видаляємо стовпці Artist Name і Track Name, що не беруть участь у класифікуванні, та виводимо основну інформацію про наш датафрейм (рис. 3.1).

```
#   Column              Non-Null Count  Dtype
---  ------              --------------  -----
0   Popularity          17568 non-null  float64
1   danceability        17996 non-null  float64
2   energy              17996 non-null  float64
3   key                 15982 non-null  float64
4   loudness            17996 non-null  float64
5   mode                17996 non-null  int64
6   speechiness         17996 non-null  float64
7   acousticness        17996 non-null  float64
8   instrumentalness    13619 non-null  float64
9   liveness            17996 non-null  float64
10  valence             17996 non-null  float64
11  tempo               17996 non-null  float64
12  duration_in min/ms  17996 non-null  float64
13  time_signature      17996 non-null  int64
14  Class               17996 non-null  int64
```

Рисунок 3.1 – Загальна інформація про датафрейм

Як можемо побачити, всі дані non-null та типи даних правильно визначилися, що вже дуже добре для нас.

Також переглянемо який вид мають наші дані отражу після завантаження їх в датафрейм. Виведемо перші 5 рядків (рис. 3.2).

	Popularity	danceability	energy	...	duration_in min/ms	time_signature	Class
0	60.0	0.854	0.564	...	234596.0	4	5
1	54.0	0.382	0.814	...	251733.0	4	10
2	35.0	0.434	0.614	...	109667.0	4	6
3	66.0	0.853	0.597	...	173968.0	4	5
4	53.0	0.167	0.975	...	229960.0	4	10

Рисунок 3.2 – Отриманий датафрейм

Наступним кроком перевіримо чи зустрічаються в стовпцях пропущені значення(рис. 3.3).

Popularity	428
danceability	0
energy	0
key	2014
loudness	0
mode	0
speechiness	0
acousticness	0
instrumentalness	4377
liveness	0
valence	0
tempo	0
duration_in min/ms	0
time_signature	0
Class	0

Рисунок 3.3 – Перевірка даних на те, чи зустрічаються числа менші за нуль

Бачимо, що в стовпцях Popularity, key і instrumentalness доволі багато пустих рядків, тому замінимо їх модою їх жанру (стовпець Class) (рис 3.4).

```
Popularity              0
danceability            0
energy                  0
key                     0
loudness                0
mode                    0
speechiness             0
acousticness            0
instrumentalness        0
liveness                0
valence                 0
tempo                   0
duration_in min/ms      0
time_signature          0
Class                   0
```

Рисунок 3.4 – Перевірка на нуль та заміна на середні значення

Останнім кроком перед поділом даних буде побудова матриці кореляцій, щоб перевірити чи є в нашому наборі даних мультиколінеарність (рис 3.5).

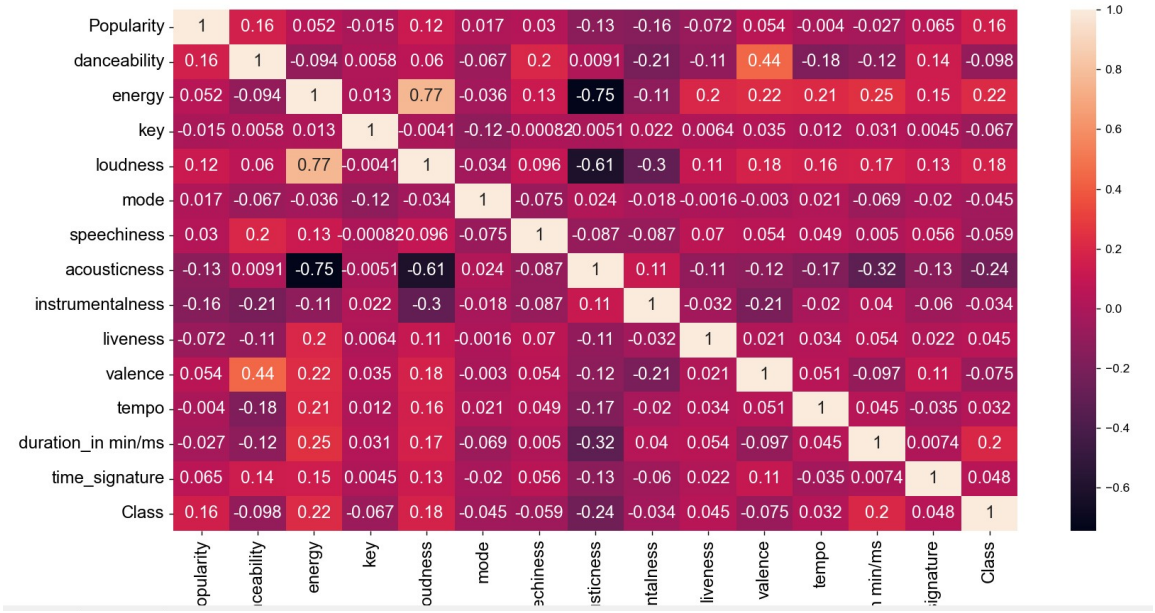

Рисунок 3.5 – Кореляційна матриця

З матриці зрозуміло, що кореляція між факторами знаходиться у нормі.

Чудово, ми отримали датафрейм з яким будемо працювати далі. Тепер можемо подивитись розподіл наших даних по жанрам, попередньо об’єднавши наші дві таблиці “train.csv” і “submission.csv” по стовпцю Class, щоб отримати назви жанрів для відображення на гістограмі (рис 3.6).

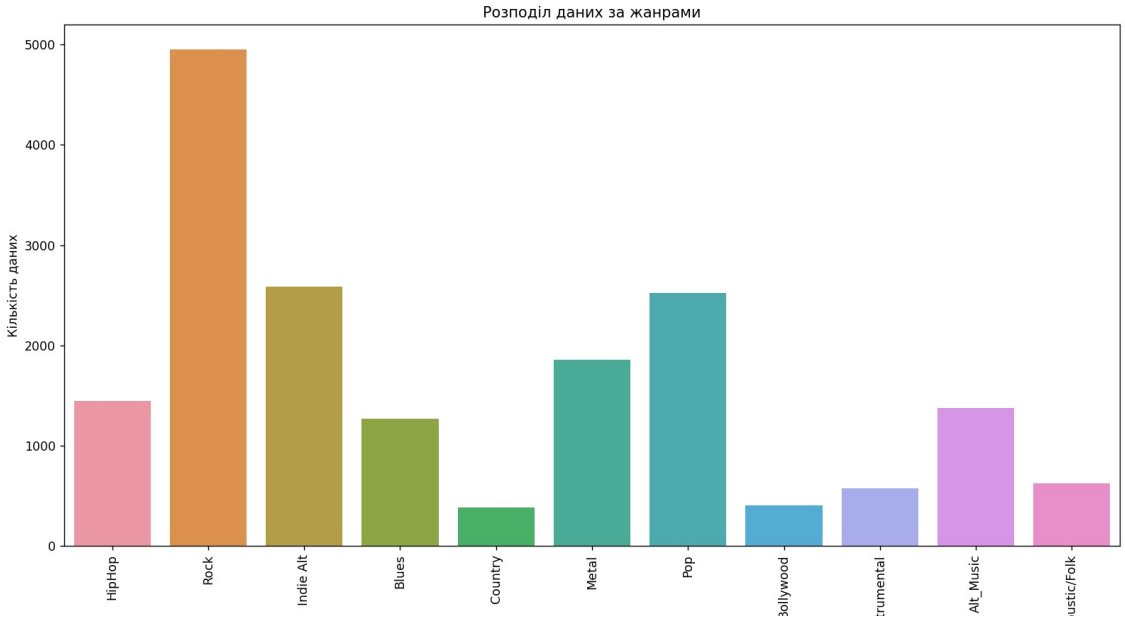

Рисунок 3.6 – Відображення кількості пісень кожного жанру

З графіку видно, що у датафреймі найбільше записів з жанром Rock.

Для усунення небажаних варіацій в даних приведемо їх до спільного масштабу, тобто зробимо стандартизацію і виведемо графіки розподілу для всіх характеристик (рис. 3.7).

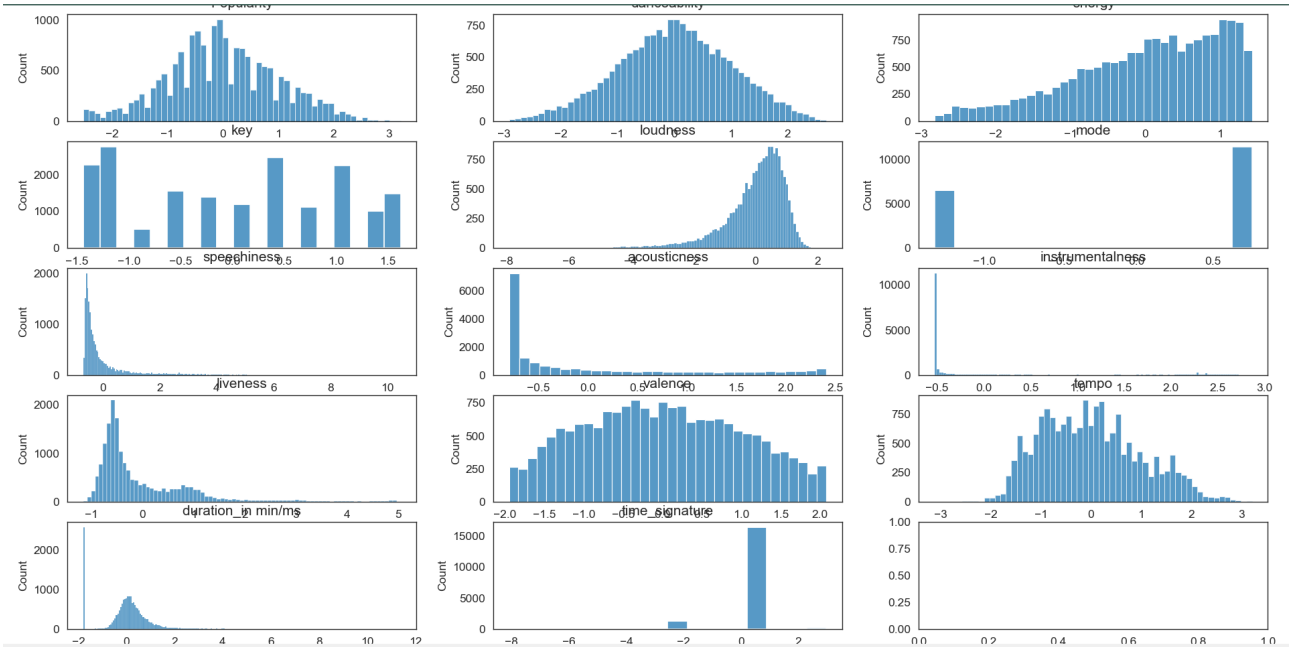

Рисунок 3.7 – Графіки розподілу характеристик

3.3 Поділ даних

Останнім кроком ми ділимо дані на тренувальні та тестові для подальшої роботи з методами класифікації (рис 3.8).

```
# Відокремлення стовпця з мітками класів
labels = train_data['Class']
# Відокремлення ознакових стовпців
features = train_data.drop(['Class'], axis=1)
# Розділення даних на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2)
```

Рисунок 3.8 – Поділ даних на інформацію та результат

Щоб уникнути оверфіту, ми розділили наш набір даних на навчальні та тестові, а саме на 80% тренувальних та 20% даних, на яких буде проводитися тестування. Це дасть нам краще уявлення про те, як наші методи працюють на етапі тестування. Таким чином наші методи тестуються на невидимих даних, для кращого розуміння їх коректності використання для розв’язання поставленої нами задачі.

4.ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

4.1 Обґрунтування вибору методів інтелектуального аналізу даних

Мною було обрано чотири методи для задачі класифікації, а в подальшому і порівняні їх – це методи Gradient Boosting Classifier, Random Forest Classifier, K Neighbors Classifier та Naive Bayes. Нижче наведено обґрунтування цього вибору.

1. **Gradient Boosting Classifier:**

Gradient Boosting Classifier (Градієнтний бустинг) - це ансамбль моделей машинного навчання, який використовує метод градієнтного підсилення для побудови сильної прогностичної моделі. Ансамбль означає поєднання декількох слабких моделей з метою отримання потужної та стійкої прогностичної моделі.

Процес побудови Gradient Boosting Classifier включає послідовне навчання і додавання слабких моделей, кожна з яких намагається скоригувати помилки попередніх моделей. Ансамбль об'єднує прогнози всіх моделей шляхом агрегації або зваженого голосування, що дозволяє отримати кінцевий результат класифікації.

Gradient Boosting Classifier відомий своєю здатністю створювати дуже потужні моделі, здатні до високої точності класифікації, а також метод має гнучкість для різних типів даних. Саме тому цей метод може бути дуже ефективним для розв’язку нашої задачі. Проте, він може бути вразливим до перенавчання, тому необхідно налагодити параметри моделі та обмежити глибину дерев. Також мінусом цього методу є час: модель навчається та прогнозує значення довше ніж інші методи.

2. **Random Forest Classifier:**

Random Forest Classifier (Випадковий ліс) є ще одним ансамблевим методом, який використовує декілька рішень-дерев для класифікації. Кожне дерево в Random Forest Classifier вирішує задачу класифікації шляхом розбиття навчальних даних на основі різних умов.

Процес побудови Random Forest починається з вибору випадкового піднабору даних і побудови дерева рішень для цього піднабору. Дерева формуються, розглядаючи різні атрибути та їх значення для розділення даних на гілки. Потім модель голосує або обчислює середнє значення прогнозів всіх дерев, щоб отримати кінцевий результат класифікації.

Random Forest Classifier показує хорошу ефективність та швидкість, порівняно з іншими алгоритмами класифікації. Він також добре підходить для задач класифікації і регресії. Зважаючи на наявність числових характеристик у ваших даних, використання Random Forest Classifier може бути доцільним для вашої задачі.

3. **K Neighbors Classifier:**

Метод K-Nearest Neighbors цікавий тим, що це алгоритм лінивого навчання і тому не вимагає підготовки перед прогнозуванням в реальному часі. Це робить алгоритм KNN набагато швидшим, ніж інші алгоритми, які потребують навчання.

Одним з завданням для вибору параметрів залишається лише вибір K-найближчих точок – сусідів, звідси метод призначає точку даних до класу, до якого належить більшість K точок даних. Так як музика поділяється на жанри саме по стилістичним та музичним особливостям, було б доцільно шукати схожі точки характеристик, щоб проводити класифікацію.

Мінусом методу KNN є погана робота з даними великої розмірності, оскільки при великій кількості вимірів алгоритму стає важко обчислити відстань у кожному вимірі.

4. **Naive Bayes:**

Naive Bayes (Наївний Баєсівський класифікатор) - це метод класифікації, який базується на застосуванні теореми Баєса з припущенням незалежності між ознаками. Він вважає, що кожна ознака внесе внесок до кінцевого результату класифікації незалежно від інших ознак. Цей метод добре працює в багатьох сферах, зокрема в обробці тексту, аналізі настрою, фільтрації спаму та багатьох інших задачах класифікації.

Процес класифікації за допомогою Naive Bayes включає побудову моделі на основі навчальних даних та використання отриманих ймовірностей для прикладів з тестових даних для прийняття рішення про класифікацію. Наївний Баєс заснований на припущенні, що ознаки мають незалежні внески до класифікації та використовує формулу Баєса для обчислення ймовірностей.

Метод добре працює з великими обсягами даних і може бути ефективним, навіть якщо припущення про незалежність ознак не виконується повністю.

З урахуванням кількості записів (14 000) та характеристик (14) у нашому наборі даних, метод Naive Bayes може бути добрим варіантом для класифікації. Він дозволить швидко навчити модель та здійснювати прогнози для нових прикладів з досить високою ефективністю.

Обрані методи мають потенціал ефективно класифікувати музичні композиції за жанром, забезпечуючи гнучкість, точність та швидкість обробки даних. Для порівняння їх ефективності можна провести експерименти на тренувальному наборі даних та оцінити їхню точність та час виконання.

4.2 Аналіз отриманих результатів для методу Gradient Boosting Classifier

Для аналізу методу підгонимо модель під тренувальні дані та перевіримо точність прогнозування отриманої модельки (рис 4.1).

```
# Створення та навчання класифікатора
clf = GradientBoostingClassifier()
clf.fit(X_train, y_train)
# Передбачення міток класів для тестових даних
y_pred = clf.predict(X_test)
print("Accuracy Gradient Boosting Classifier on train data:", clf.score(X_train, y_train))
print("Accuracy Gradient Boosting Classifier on test data:", clf.score(X_test, y_test))
get_confusion_matrix(y_test, y_pred)
```

```
Accuracy Gradient Boosting Classifier on train data: 0.7018616282300639
Accuracy Gradient Boosting Classifier on test data: 0.6430555555555556
```

Рисунок 4.1 – Результати точності отриманої модельки для Gradient Boosting

З результатів бачимо, що ми отримали достатньо високу точність прогнозування жанру за використання методу GBC і на тестових даних і для тренувальних даних.

Один з методів перевірки продуктивності роботи модельки є матриця невідповідностей. Приклад такої матриця для даного метода зображено на рисунку 4.2.

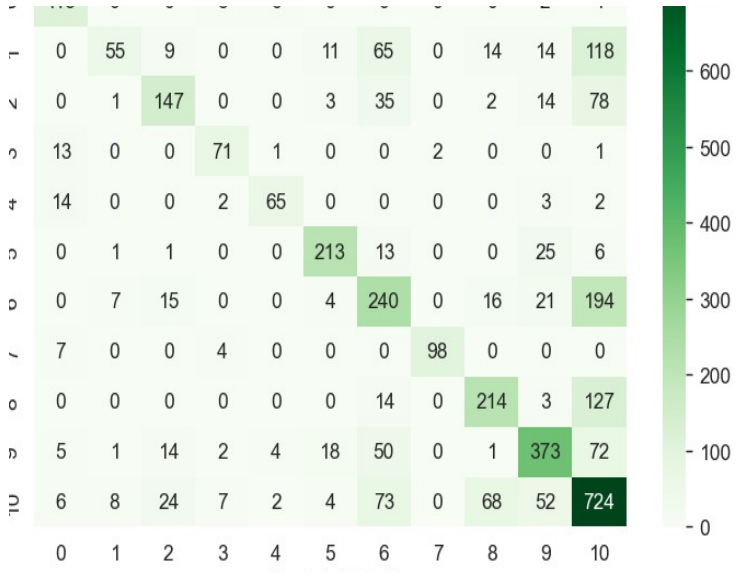

Рисунок 4.2 – Матриця невідповідностей для Gradient Boosting

Дана матриця побудована за допомогою бібліотеки Sklearn. Кожен елемент матриці представляє кількість прикладів, які були класифіковані відповідно до

певної комбінації фактичного та передбаченого класів. Як видно з рисунку 4.6, модель в більшості випадків досить точно визначала жанр до якого належить пісня.

4.3 Аналіз отриманих результатів для методу Random Forest

Для аналізу методу підгонимо модель під тренувальні дані та перевіримо точність прогнозування отриманої модельки (рис 4.3).

```
# Створюємо класифікатор Random Forest та навчаємо його на тренувальних даних
clf = RandomForestClassifier(max_features=50, random_state=42)
clf.fit(X_train, y_train)
# Прогнозуємо класи на тестових даних та обчислюємо точність класифікації
y_pred = clf.predict(X_test)
print("Accuracy Random Forest Classifier on train data:", clf.score(X_train, y_train))
print("Accuracy Random Forest Classifier on test data:", clf.score(X_test, y_test))
get_confusion_matrix(y_test, y_pred)
```

```
Accuracy Random Forest Classifier on train data: 0.9426229508196722
Accuracy Random Forest Classifier on test data: 0.5894444444444444
```

Рисунок 4.3 – Результати точності отриманої модельки для Random Forest

Як бачимо, Random Forest має доволі непогану точність для тестових даних і дуже високу для навчальних. Для нашої вибірки оптимальною кількістю дерев є число 50, при його зменшенні зменшується точність для тестових даних, а при збільшенні особливо не змінюється.

Тепер подивимося на матрицю помилок на рисунку 4.4.

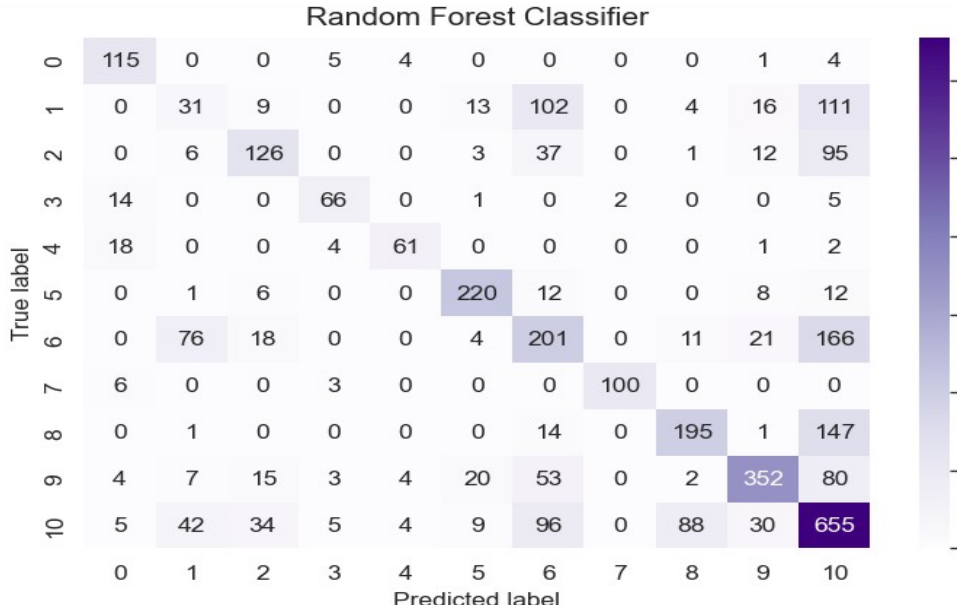

Рисунок 4.4 – Матриця невідповідностей для Random Forest

Як видно з рисунку 4.4, модель в більшості випадків досить точно визначала жанр до якого належить пісня. Хоча для деяких жанрів, зокрема №1 та №6, спостерігається невелика кількість вірних класифікацій, яка менша за кількість помилкових класифікацій.

4.4 Аналіз отриманих результатів для методу K Neighbors

Для початку, щоб аналізувати результати роботи методу KNN, необхідно обрати найкращу з можливих модельок, для цього ми використали модуль, що перебирає всі можливі варіанти параметрів та дає зрозуміти, які параметри найкраще вирішують поставлену задачу (рис. 4.5).

```
#Пошук найкращих параметрів для методу KNN
knn = KNeighborsClassifier()
parameters={'n_neighbors':range(2,25)}
gridsearch=GridSearchCV(knn, parameters,cv=10,verbose=1)
gridsearch.fit(X_train,y_train)
print ("Best parametrs for KNN-", gridsearch.best_estimator_)
```

```
Fitting 10 folds for each of 23 candidates, totalling 230 fits
Best parametrs for KNN- KNeighborsClassifier(n_neighbors=24)
```

Рисунок 4.5 – Пошук найкращих параметрів для методу KNN

Для заданого розміру даних для тренування найкращою кількістю модель з кількістю сусідів, що рівна 16, тому в подальшому будемо використовувати саме її. Після цього підгонимо модель під тренувальні дані та перевіримо точність прогнозування отриманої модельки (рис 4.6).

```
# Створюємо класифікатор і навчаємо його на тренувальних даних
knn = KNeighborsClassifier(n_neighbors=24)
knn.fit(X_train, y_train)
# Перевіряємо точність на тестових даних
y_pred = knn.predict(X_test)
print("Accuracy K Neighbors Classifier on train data:", knn.score(
print("Accuracy K Neighbors Classifier on test data:", knn.score(X_
get_confusion_matrix(y_test, y_pred)
```

```
Accuracy K Neighbors Classifier on train data: 0.5532092247846624
Accuracy K Neighbors Classifier on test data: 0.5083333333333333
```

Рисунок 4.6 – Результати точності отриманої модельки для KNN

З результатів видно, що з нашою вибіркою цей метод справляється середньо, це стосується як і тренувальних так і тестових даних.

Тепер подивимося на матрицю помилок на рисунку 4.7.

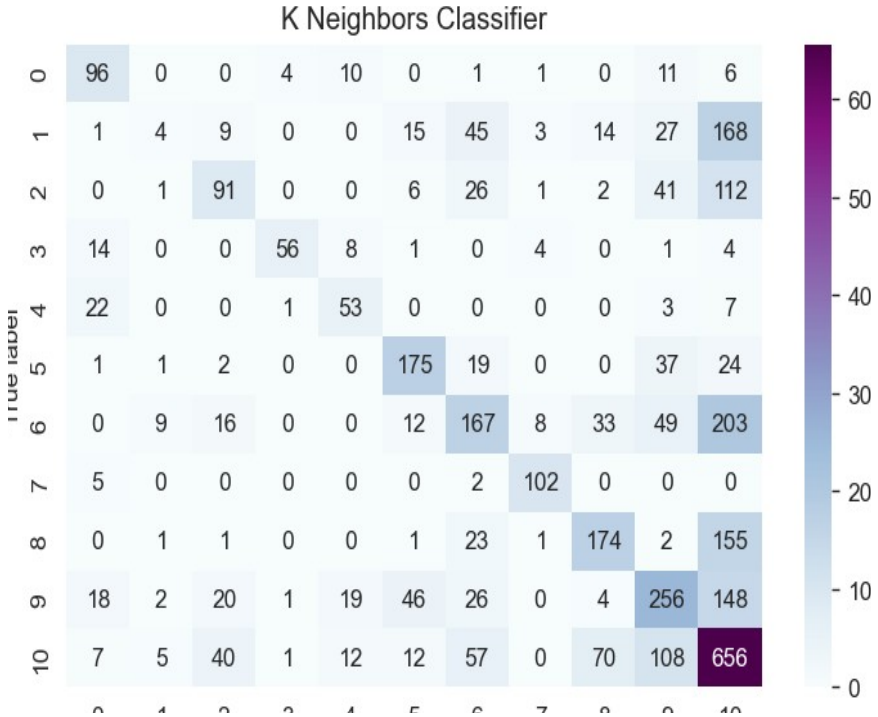

Рисунок 4.7 – Матриця невідповідностей для KNN

Бачимо, що модель допустила багато хибних прогнозів у класі №10 і їх навіть більше ніж правильних. А також погано справилась з прогнозуванням клас №1.

4.5 Аналіз отриманих результатів для методу Naive Bayes

Для аналізу методу підгонимо модель під тренувальні дані та перевіримо точність прогнозування отриманої модельки (рис 4.8).

```
# Створюємо класифікатор і навчаємо його на тренувальних даних
nb = GaussianNB()
nb.fit(X_train, y_train)
# Перевіряємо точність на тестових даних
y_pred = nb.predict(X_test)
print("Accuracy Naive Bayes on train data:", nb.score(X_train, y_train))
print("Accuracy Naive Bayes on test data:", nb.score(X_test, y_test))
get_confusion_matrix(y_test, y_pred)
```

```
Accuracy Naive Bayes on train data: 0.47082522923034176
Accuracy Naive Bayes on test data: 0.455
```

Рисунок 4.8– Результати точності отриманої модельки для Naive Bayes

З результатів видно, що з нашою вибіркою цей метод справляється на рівні нижче середнього, це стосується як і тренувальних так і тестових даних.

Тепер подивимося на матрицю помилок на рисунку 4.9.

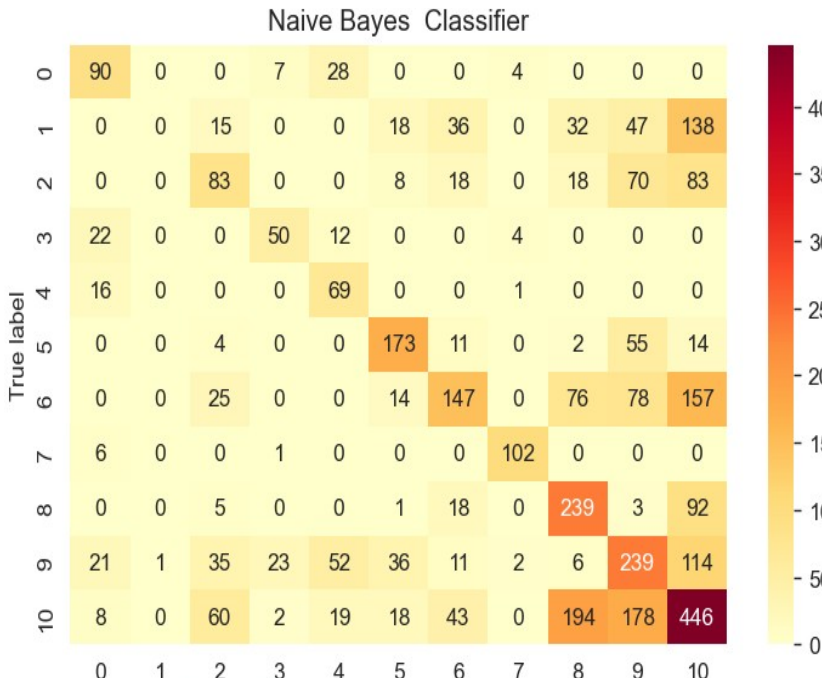

Рисунок 4.9 – Матриця невідповідностей для Naive Bayes

Бачимо, що модель Naive Bayes не дуже справилась з нашим набором, так як є багато хибно визначених прогнозів майже для усіх жанрів. Також можна побачити, що модель не навчилась визначати жанр №1, можливо через маленький розмір даних для нього.

4.6 Порівняння отриманих результатів методів

Проаналізувавши окремо кожен із методів, що були мною використані під час прогнозування захворюваності на підготовлених даних, варто провести порівняння даних методів.

Порівняння точності зображене на рисунку 4.10:

```
Accuracy Random Forest Classifier on train data: 0.9426229508196722
Accuracy Random Forest Classifier on test data: 0.5894444444444444
Accuracy Gradient Boosting Classifier on train data: 0.7018616282300639
Accuracy Gradient Boosting Classifier on test data: 0.6430555555555556
Fitting 10 folds for each of 23 candidates, totalling 230 fits
Best parametrs for KNN- KNeighborsClassifier(n_neighbors=24)
Accuracy K Neighbors Classifier on train data: 0.5532092247846624
Accuracy K Neighbors Classifier on test data: 0.5083333333333333
Accuracy Naive Bayes on train data: 0.47082522923034176
Accuracy Naive Bayes on test data: 0.455
```

Рисунок 4.10 – Порівняння точності методів

Візьмемо для повірняння точності для тестової вибірки, адже суть задачі була саме у прогнозуванні жанру для нових пісень. Отже маємо: для Gradient Boosting Classifier — 64%, для Random Forest Classifier — 59%, для K Neighbors Classifier —50%, і для Naive Bayes Classifier — 45%.

Таким чином, найкращим методом для класифікації музики за жанром є Gradient Boosting Classifier, який показав найвищу точність. Він також продемонстрував гарні результати на матриці помилок (див. рис. 4.11). На другому місці розташувався метод Random Forest Classifier, який відстає від лідера на 5%. Щодо третього місця, K Neighbors Classifier показав себе краще

Naive Bayes і по точності, і по матриці невідповідностей, які можна побачити на рисунку 4.11.

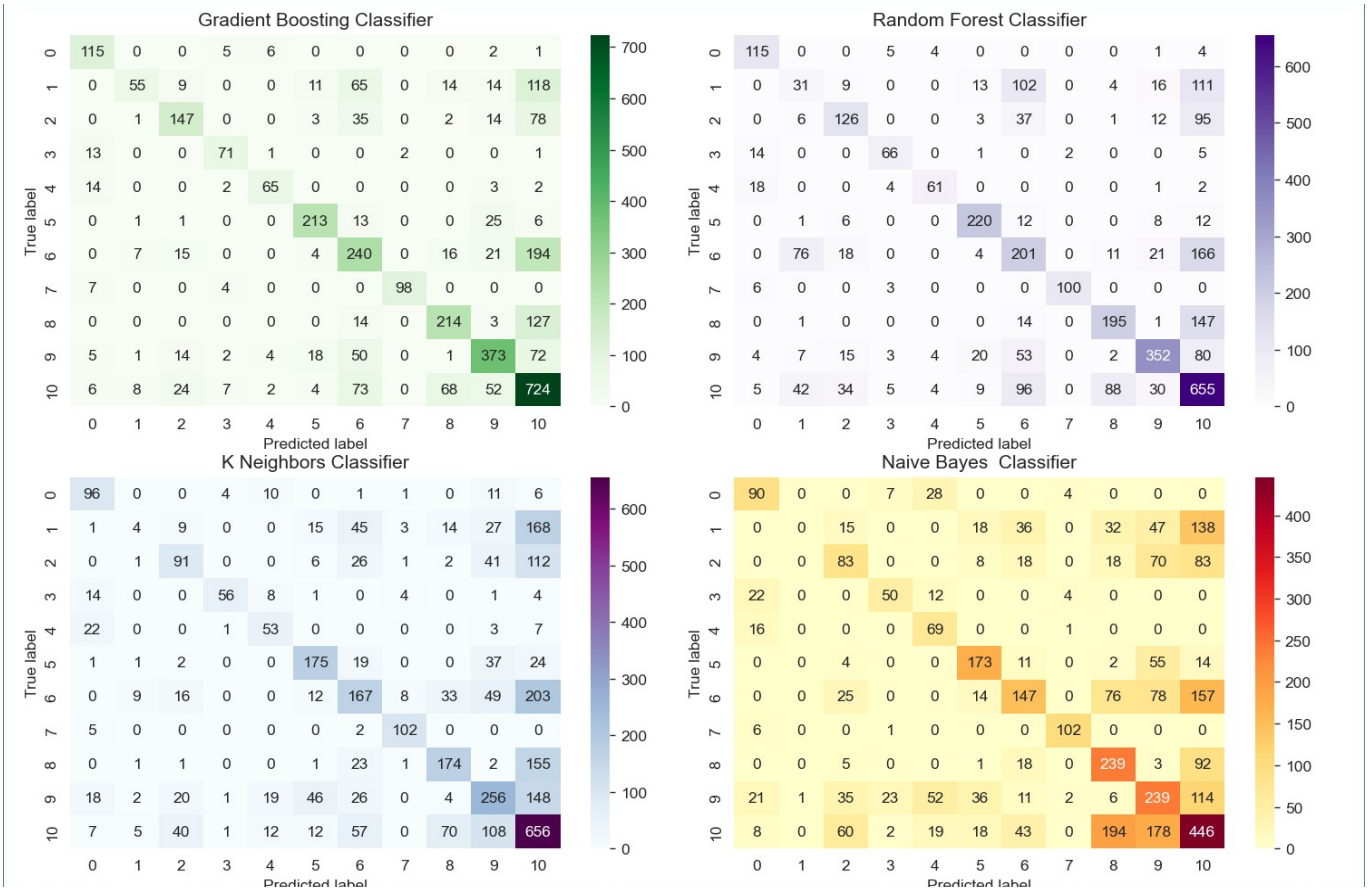

Рисунок 4.11 – Матриця невідповідностей для усіх методів

ВИСНОВКИ

Тема класифікації музики за їхніми характеристиками є надзвичайно захопливою і має широкі практичні застосування. Вона становить велике значення в музичній індустрії, аналізі аудіоданих та задоволенні музичних потреб користувачів. Багато людей пристрасно люблять музику, цікавляться її особливостями та хотіли б знати, який жанр переважає в їхньому плейлисті. Наприклад, такі музичні платформи, як Spotify, не надають користувачам прямої інформації про жанр конкретного треку, але дозволяють переглядати його характеристики, такі як темп, гучність та інші.

Ми реалізували програмне забезпечення, що тренує моделі чотирьох методів - Gradient Boosting Classifier, Random Forest Classifier, K Neighbors Classifier та Naive Bayes Classifier, та під час порівняння результатів допомагає визначити який з них краще обрати.

Нами була виконана робота з датасетом, у ході якої ми дослідили його структуру та виправили наявні помилки.

Також обґрунтовано вибір двох методів для прогнозування результату та проаналізовано результати кожного з методів окремо.

Аналізуючи отримані результати у кожному з пунктів, можна дійти висновків, що найвищу точність серед усіх методів показав Gradient Boosting Classifier, що робить його найкращим вибором для класифікації музики за жанрами. Random Forest Classifier також продемонстрував хорошу точність, але трохи відстає від лідера. Класифікатор K Neighbors та Naive Bayes показали набагато нижчу точність.

З огляду на результати та аналіз матриць помилок, варто використовувати Gradient Boosting Classifier для класифікації музики за жанрами, адже його характеристики є більш коректними для роботи з такими даними.

ПЕРЕЛІК ПОСИЛАНЬ

1. Документація мови програмування Python. [Електронний ресурс] Режим доступу до ресурсу: https://docs.python.org/3/
2. Бібліотека Pandas. [Електронний ресурс] – Режим доступу до ресурсу: https://pandas.pydata.org/docs/
3. Бібліотека Seaborn. [Електронний ресурс] – Режим доступу до ресурсу: https://seaborn.pydata.org/introduction.html
4. Бібліотека Matplotlib. [Електронний ресурс] – Режим доступу до ресурсу: https://matplotlib.org/stable/
5. Бібліотека Sklearn. [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable/user_guide.html
6. Music Genre Classification Based on Deep Learning [Електронний ресурс] / Muhammad Zakarya. – 2022. – Режим доступу до ресурсу: https://www.hindawi.com/journals/misy/2022/2376888/.
7. K-Nearest Neighbors Algorithm in Python and Scikit-Learn [Електронний ресурс] – Режим доступу до ресурсу: https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/.
8. Naive Bayes Algorithm in Python and Scikit-Learn [Електронний ресурс] – Режим доступу до ресурсу: https://stackabuse.com/the-naive-bayes-algorithm-in-python-with-scikit-learn/.
9. Random Forest Algorithm in Python and Scikit-Learn [Електронний ресурс] – Режим доступу до ресурсу: https://stackabuse.com/get-feature-importances-for-random-forests-with-python-and-scikit-learn/.
10. Gradient Boosting Algorithm in Python and Scikit-Learn [Електронний ресурс] – Режим доступу до ресурсу: https://stackabuse.com/gradient-boosting-classifiers-in-python-with-scikit-learn/.

11. Confusion Matrix in Machine Learning. [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/.

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду класифікації музичних композицій за жанром на основі їх характеристик. Методи K-Nearest Neighbors, Gradient Boosting, Naive Bayes, Random Forest

(Найменування програми (документа))

Жорсткий диск

(Вид носія даних)

(Обсяг програми (документа), арк.)

Студентки групи ІП-15 2 курсу

Кондрацької С.Л

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score, confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler



test_data = pd.read_csv("files/archive/test.csv")
test_data = test_data.drop('Artist Name', axis=1)
test_data = test_data.drop('Track Name', axis=1)

def get_confusion_matrix(y_test, y_pred):
    # матриця помилок
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
    plt.xlabel('Predicted label')
    plt.ylabel('True label')
    plt.show()
if __name__ == '__main__':
    train_data = pd.read_csv("files/archive/train.csv")
    genres = pd.read_csv("files/archive/submission.csv")



    train_data = train_data.drop('Artist Name', axis=1)
    train_data = train_data.drop('Track Name', axis=1)




    print(train_data.info())
    print(train_data.describe())
```

```
print(train_data.head())
print(train_data.isnull().sum())
#---------------------- Замінюємо пропущені значення модою за групою
train_data['key'] = train_data.groupby('Class')['key'].transform(lambda x: x.fillna(x.mode()[0]))
train_data['instrumentalness'] = train_data.groupby('Class')['instrumentalness'].transform(lambda x: x.fillna(x.mode()
[0]))
train_data['Popularity'] = train_data.groupby('Class')['Popularity'].transform(lambda x: x.fillna(x.mode()[0]))
# Перевіряємо, чи більше немає пропущених значень
print(train_data.isnull().sum())


# кореляційна матриця
plt.figure(figsize=(16, 10))
sns.heatmap(train_data.corr(), annot=True, annot_kws={"size": 14})
sns.set_style('white')
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()


# Об'єднуємо дані за допомогою id
train_data_g = pd.merge(train_data, genres, on="Class")

# Рахуємо кількість даних для кожного жанру
sns.countplot(data=train_data_g, x='Name')
plt.xticks(rotation=90)
plt.xlabel("Жанр")
plt.ylabel("Кількість даних")
plt.title("Розподіл даних за жанрами")
plt.show()




labels = train_data['Class']
features = train_data.drop(['Class'], axis=1)


#---------------------- графіки розподілу
columns = features.columns
num_rows = (len(columns) - 1) // 3 + 1
num_cols = 3
fig, axes = plt.subplots(num_rows, num_cols, figsize=(15, 5 * num_rows))
```

```
for i, col in enumerate(columns):
    ax_row = i // num_cols
    ax_col = i % num_cols
    # Вибираємо поточний підграфік
    ax = axes[ax_row, ax_col] if num_rows > 1 else axes[ax_col]
    sns.histplot(features[col], ax=ax)
    ax.set_title(col)
plt.tight_layout()
plt.show()



#---------------------- Стандартизуємо дані
scaler = StandardScaler()
scaled_data = scaler.fit_transform(features)
features = pd.DataFrame(scaled_data, columns=features.columns)


columns = features.columns
num_rows = (len(columns) - 1) // 3 + 1
num_cols = 3
fig, axes = plt.subplots(num_rows, num_cols, figsize=(15, 5 * num_rows))
for i, col in enumerate(columns):
    ax_row = i // num_cols
    ax_col = i % num_cols
    # Вибираємо поточний підграфік
    ax = axes[ax_row, ax_col] if num_rows > 1 else axes[ax_col]
    sns.histplot(features[col], ax=ax)
    ax.set_title(col)
plt.tight_layout()
plt.show()




#---------------------- Розділення даних на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2)
```

```
#---------------------- Random Forest ----------------------
clf = RandomForestClassifier(max_features=50, random_state=42)
clf.fit(X_train, y_train)
# Прогнозуємо класи на тестових даних та обчислюємо точність класифікації
y_predR = clf.predict(X_test)
print("Accuracy Random Forest Classifier on train data:", clf.score(X_train, y_train))
print("Accuracy Random Forest Classifier on test data:", clf.score(X_test, y_test))
#get_confusion_matrix(y_test, y_pred)



#---------------------- Gradient Boosting ----------------------
clf = GradientBoostingClassifier()
clf.fit(X_train, y_train)
y_predGR = clf.predict(X_test)
print("Accuracy Gradient Boosting Classifier on train data:", clf.score(X_train, y_train))
print("Accuracy Gradient Boosting Classifier on test data:", clf.score(X_test, y_test))
#get_confusion_matrix(y_test, y_pred)



#---------------------- K Neighbors ----------------------
#Пошук найкращих параметрів для методу KNN
knn = KNeighborsClassifier()
parameters={'n_neighbors':range(2,25)}
gridsearch=GridSearchCV(knn, parameters,cv=10,verbose=1)
gridsearch.fit(X_train,y_train)
print ("Best parametrs for KNN-", gridsearch.best_estimator_)


knn = KNeighborsClassifier(n_neighbors=24)
knn.fit(X_train, y_train)
y_predK = knn.predict(X_test)
print("Accuracy K Neighbors Classifier on train data:", knn.score(X_train, y_train))
print("Accuracy K Neighbors Classifier on test data:", knn.score(X_test, y_test))
#get_confusion_matrix(y_test, y_pred)


#---------------------- Naive Bayes ----------------------
```

```
nb = GaussianNB()
nb.fit(X_train, y_train)
y_predG = nb.predict(X_test)
print("Accuracy Naive Bayes on train data:", nb.score(X_train, y_train))
print("Accuracy Naive Bayes on test data:", nb.score(X_test, y_test))
#get_confusion_matrix(y_test, y_pred)




#---------------------- Побудова матриць помилок
cm1 = confusion_matrix(y_test, y_predGR)
cm2 = confusion_matrix(y_test, y_predR)
cm3 = confusion_matrix(y_test, y_predK)
cm4 = confusion_matrix(y_test, y_predG)


fig, axes = plt.subplots(2, 2, figsize=(12, 10))


sns.heatmap(cm1, annot=True, cmap='Greens', fmt='g', ax=axes[0, 0])
axes[0, 0].set_xlabel('Predicted label')
axes[0, 0].set_ylabel('True label')
axes[0, 0].set_title('Gradient Boosting Classifier')


sns.heatmap(cm2, annot=True, cmap='Purples', fmt='g', ax=axes[0, 1])
axes[0, 1].set_xlabel('Predicted label')
axes[0, 1].set_ylabel('True label')
axes[0, 1].set_title('Random Forest Classifier')


sns.heatmap(cm3, annot=True, cmap='BuPu', fmt='g', ax=axes[1, 0])
axes[1, 0].set_xlabel('Predicted label')
axes[1, 0].set_ylabel('True label')
axes[1, 0].set_title('K Neighbors Classifier')


sns.heatmap(cm4, annot=True, cmap='YlOrRd', fmt='g', ax=axes[1, 1])
axes[1, 1].set_xlabel('Predicted label')
axes[1, 1].set_ylabel('True label')
axes[1, 1].set_title('Naive Bayes  Classifier')
```

```
plt.tight_layout()
plt.show()
```