

AI AUDITOR: CRAFTING YOUR PERFECT PLAYLIST WITH MACHINE LEARNING

*Minor project-II report submitted
in partial fulfillment of the requirement for award of the degree of*

**Bachelor of Technology
in
Computer Science & Engineering**

By

CHAKALI VIJAYMANI	(21UECM0041)	(19315)
MANDHALA MANOJ BABU	(2UECM0149)	(19600)
RAPURU GAUTAM	(21UECM0198)	(19434)

*Under the guidance of
Mrs.S.Thylashri.M.E.,
ASSISTANT PROFESSOR*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING**

**VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF
SCIENCE & TECHNOLOGY**

(Deemed to be University Estd u/s 3 of UGC Act, 1956)

**Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA**

May, 2024

AI AUDITOR: CRAFTING YOUR PERFECT PLAYLIST WITH MACHINE LEARNING

*Minor project-II report submitted
in partial fulfillment of the requirement for award of the degree of*

**Bachelor of Technology
in
Computer Science & Engineering**

By

CHAKALIVIJAYMANI	(21UECM0041)	(19315)
MANDHALA MANOJ BABU	(21UECM0149)	(19600)
RAPURU GAUTAM	(21UECM0198)	(19434)

*Under the guidance of
Mrs.S.Thylashri.M.E.,
ASSISTANT PROFESSOR*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING**

**VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF
SCIENCE & TECHNOLOGY**

(Deemed to be University Estd u/s 3 of UGC Act, 1956)

**Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA**

May, 2024

CERTIFICATE

It is certified that the work contained in the project report titled "AI AUDITOR: CRAFTING YOUR PERFECT PLAYLIST WITH MACHINE LEARNING" by "CHAKALI VIJAYMANI (21UECM0041), MANDHALA MANOJ (21UECM0149), RAPURU GAUTAM (21UECM0198)" has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor

Computer Science & Engineering

School of Computing

Vel Tech Rangarajan Dr. Sagunthala R&D

Institute of Science & Technology

May, 2024

Signature of Professor In-charge

Computer Science & Engineering

School of Computing

Vel Tech Rangarajan Dr. Sagunthala R&D

Institute of Science & Technology

May, 2024

DECLARATION

We declare that this written submission represents my ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

CHAKALI VIJAYMANI

Date: / /

MANDHALA MANOJ BABU

Date: / /

RAPURU GAUTAM

Date: / /

APPROVAL SHEET

This project report entitled “AI AUDITOR: CRAFTING YOUR PERFECT PLAYLIST WITH MACHINE LEARNING” by CHAKALI VIJAYMANI (21UECM0041), MANDHALA MANOJ BABU (21UECM0149), RAPURU GAUTAM (21UECM0198) is approved for the degree of B.Tech in Computer Science & Engineering.

Examiners

Supervisor

Mrs.S.Thylashri,M.E.

Assistant Professor

Date: / /

Place:

ACKNOWLEDGEMENT

We express our deepest gratitude to our respected **Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B.E. (EEE), B.E. (MECH), M.S (AUTO),D.Sc., Foundress President Dr. R. SAGUNTHALA RANGARAJAN M.B.B.S.** Chairperson Managing Trustee and Vice President.

We are very much grateful to our beloved **Vice Chancellor Prof. S. SALIVAHANAN**, for providing us with an environment to complete our project successfully.

We record indebtedness to our **Professor & Dean, Department of Computer Science & Engineering, School of Computing, Dr. V. SRINIVASA RAO, M.Tech., Ph.D.**, for immense care and encouragement towards us throughout the course of this project.

We are thankful to our **Head, Department of Computer Science & Engineering, Dr.M.S. MURALI DHAR, M.E., Ph.D.**, for providing immense support in all our endeavors.

We also take this opportunity to express a deep sense of gratitude to our Internal Supervisor **Mrs.S.Thylashri,M.E.**, for her cordial support, valuable information and guidance, she helped us in completing this project through various stages.

A special thanks to our **Project Coordinators Mr. V. ASHOK KUMAR, M.Tech., Ms. C. SHYAMALA KUMARI, M.E.**, for their valuable guidance and support throughout the course of the project.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

CHAKALI VIJAYMANI	(21UECM0041)
MANDHALA MANOJ BABU	(21UECM0149)
RAPURU GAUTAM	(21UECM0198)

ABSTRACT

Music genre classification is an important task in music information retrieval systems, music recommendation systems, and music streaming services. In this project, we propose a machine learning-based approach for music genre classification using the K-Nearest Neighbors (KNN) algorithm. We first extract audio features from a large dataset of songs, including pitch, tempo, and spectral features. We then pre-process the data and train a KNN classifier on the extracted features. We evaluate the performance of the classifier using various metrics, such as accuracy, precision, and recall. We also compare the performance of our KNN classifier with other commonly used classification algorithms, such as Random Forest and Support Vector Machine (SVM). Our results show that the KNN algorithm achieves high accuracy in classifying songs into different genres and outperforms other algorithms in certain scenarios. Therefore, the proposed method can be used to build effective music recommendation systems and music information retrieval applications.

Keywords: K-Nearest Neighbors (KNN), Metrics Music information retrieval systems, Music recommendation systems, Music streaming services, Random Forest, Spectral features, Support Vector Machine (SVM).

LIST OF FIGURES

4.1	Architecture For Music Genre Classification	10
4.2	Data flow diagram	12
4.3	Use Case Diagram	13
4.4	Class Diagram	14
4.5	Sequence Diagram	15
4.6	Activity Diagram	16
4.7	Collaboration Diagram	17
4.8	Module Description	19
4.9	Data Pre-Processing	20
4.10	Apply KNN Algorithm	21
5.1	Input Design: Dataset	25
5.2	Output Design: Training and Validating	26
5.3	Model Accuracy for Music Genre Classification Using Machine Learning	27
5.4	Test Image	29
8.1	Palagram Report	34
9.1	Poster Presentation	39

LIST OF TABLES

5.1	Test cases for music genre classification	24
-----	---	----

LIST OF ACRONYMS AND ABBREVIATIONS

AUC	Area Under Curve
CNN	Convolutional Neural Network
KNN	K Nearest Neighbours
MGR	Music Genre Recognition
MRS	Music Recommender System
ML	Machine Learning
MSCNN	Multi-Scale Convolutional Neural Network
MFCCs	Mel-Frequency Cepstral Coefficients
SVM	Support Vector Machine

TABLE OF CONTENTS

	Page.No
ABSTRACT	v
LIST OF FIGURES	vi
LIST OF TABLES	vii
LIST OF ACRONYMS AND ABBREVIATIONS	viii
1 INTRODUCTION	1
1.1 Introduction	1
1.2 Aim of the project	2
1.3 Project Domain	2
1.4 Scope of the Project	2
2 LITERATURE REVIEW	3
3 PROJECT DESCRIPTION	6
3.1 Existing System	6
3.2 Proposed System	7
3.3 Feasibility Study	7
3.3.1 Economic Feasibility	7
3.3.2 Technical Feasibility	8
3.3.3 Social Feasibility	8
3.4 System Specification	9
3.4.1 Hardware Specification	9
3.4.2 Software Specification	9
3.4.3 Standards and Policies	9
4 METHODOLOGY	10
4.1 Music Genre classification Architecture	10
4.2 Design Phase	12
4.2.1 Data Flow Diagram	12

4.2.2	Use Case Diagram	13
4.2.3	Class Diagram	14
4.2.4	Sequence Diagram	15
4.2.5	Activity Diagram	16
4.2.6	Collaboration Diagram	17
4.3	Algorithm & Pseudo Code	18
4.3.1	Algorithm	18
4.3.2	Pseudo Code	18
4.4	Module Description	19
4.4.1	Import required libraries.	19
4.4.2	Pre-Processing of data	20
4.4.3	Apply Machine Learning Algorithms(KNN)	21
4.5	Steps to execute/run/implement the project	22
4.5.1	Step1: Collecting datasets	22
4.5.2	Step2: Correlation Analysis	22
4.5.3	Step3: Creating training and testing datasets	22
5	IMPLEMENTATION AND TESTING	24
5.1	Tabulation	24
5.2	Input and Output	25
5.2.1	Input Design	25
5.2.2	Output Design	26
5.3	TESTING	27
5.4	Types of Testing	27
5.4.1	Unit testing	27
5.4.2	Integration testing	28
5.4.3	Test Result	29
6	RESULTS AND DISCUSSIONS	30
6.1	Efficiency of the Proposed System	30
6.2	Comparison of Existing and Proposed System	31
6.3	Sample Code	32
7	CONCLUSION AND FUTURE ENHANCEMENTS	33
7.1	Conclusion	33
7.2	Future Enhancements	33

8	PLAGIARISM REPORT	34
9	SOURCE CODE & POSTER PRESENTATION	35
9.1	Source Code	35
9.2	Poster Presentation	39
	References	39

Chapter 1

INTRODUCTION

1.1 Introduction

Audio classification is an Application of machine learning where different sound is categorized in certain categories. The music genre classification can be built using different approaches in which the top 4 approaches that are mostly used are listed below.

1. Multiclass support vector machine
2. K-Nearest Neighbors
3. K-means clustering algorithm
4. Convolutional neural network.

As of now decision tree having the high accuracy. So, We decided to use K-Nearest Neighbors algorithm because[1] we compared Convolutional neural networks(CNN) for that the accuracy is 50%, Support vector machines(SVM) for that the accuracy is 85% and we compared with K-nearest neighbors(KNN) for that the accuracy is 98.5%. we compared this four models for that K-Nearest Neighbors has getting the more accuracy to give good performance and till time along with optimized models organizations uses this algorithm in recommendation systems as support. K-Nearest Neighbour (KNN) is a machine learning algorithm used for regression, and classification. It is also known as the lazy learner algorithm. It simply uses a distance-based method to find the K number of similar Neighbours to new data and the class in which the majority of Neighbours lies, it results in that class as an output.

Music Genre Classification or classification of music into different categories or genres is a concept that helps the masses to differentiate between 2 genres based on their composition or the beats they withhold. In recent times, music genre classification has become a very popular concept as more and more genres are emerging around the world.

1.2 Aim of the project

Audio processing is one of the most complex tasks in data science as compared to image processing and other classification techniques. One such application is music genre classification which aims to classify the audio files in certain categories of sound to which they belong. The application is very important and requires automation to reduce the manual error and time because if we have to classify the music manually then one has to listen out each file for the complete duration.

1.3 Project Domain

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-Nearest Neighbour algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-Nearest Neighbour algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K-Nearest Neighbour algorithm. K-Nearest Neighbour algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

K-Nearest Neighbour is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. K-Nearest Neighbour algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

1.4 Scope of the Project

Scope of our project there are some many models for classification here we using the most effective model for classify the music called K-Nearest Neighbour (KNN), is a technique that has been reportedly successful in categorizing music into different genres. A supervised machine learning algorithm, the K-Nearest Neighbour technique is used to find solutions for classification and regression problems.

Chapter 2

LITERATURE REVIEW

Choi et al, [1] Compares the performance of deep convolutional neural networks (CNNs) using two different types of audio features (spectrograms and MFCCs) for music genre classification. They train and evaluate their models on a large dataset of over 1 million tracks from the Million Song Dataset, and achieve state-of-the-art results for both feature types. Convolutional neural networks (CNNs) have been actively used for various music classification tasks such as music genre classification and user-item latent feature prediction for recommendation.

Gomez et al, [2] Proposed a music genre classification system based on deep CNNs, trained on spectrogram images of audio signals. They compare their results with several other approaches, including traditional machine learning algorithms and other deep learning architectures, and demonstrate the superiority of their proposed system. Most of the music files are stored according to the song title or the artist name. This may cause trouble in searching for a song related to a specific genre.

Kim et al, [3] Proposed an ensemble-based approach for music genre classification using deep neural networks. They train multiple neural networks with different architectures and combine their predictions using voting or averaging. They achieve state-of-the-art results on several benchmark datasets and demonstrate the effectiveness of their approach. Music has also been divided into Genres and sub genres not only on the basis on music but also on the lyrics as well. This makes music genre classification difficult.

Lu et al, [4] designed a multi-scale convolutional neural network (MSCNN) for music genre classification. Their approach uses a hierarchical architecture with multiple scales of convolutional layers to capture both local and global features in music signals. They achieve state-of-the-art results on several benchmark datasets. Which aims to detect the most attention-grabbing objects in a scene and then extract pixel-accurate silhouettes for them. The merit of SOD lies in its many applications,

including foreground map evaluation

Park. K et al, [5] designed a music genre classification system based on feature learning with temporal convolutional networks (TCNs). They use a novel feature learning approach based on TCNs to automatically learn discriminative features from raw audio signals. They achieve state-of-the-art results on several benchmark datasets. Companies nowadays use music classification, either to be able to place recommendations to their customers or simply as a product. Determining music genres is the first step in the process of music recommendation.

Reddy et al, [6] Proposed a transfer learning-based approach for music genre classification using deep neural networks. They pretrain a convolutional neural network on a large dataset of image classification tasks, and then fine-tune the network on a smaller dataset of audio files. They achieve state-of-the-art results on several benchmark datasets.

Schramm et al, [7] described an end-to-end music genre classification system based on recurrent neural networks (RNNs). Their approach uses a bidirectional RNN architecture to capture both forward and backward temporal dependencies in music signals. They achieve state-of-the-art results on several benchmark datasets. Categorizing music files according to their genre is a challenging task in the area of music information retrieval (MIR). Automatic music genre classification is important to obtain music from a large collection. It finds applications in the real world in various fields like automatic tagging of unknown piece of music (useful for apps like Saavn, Wynk etc.)

Wang et al, [8] described an attention-based convolutional neural network (CNN) for music genre classification. Their approach uses both temporal and spectral attention mechanisms to better capture the salient features of music signals. They achieve state-of-the-art results on several benchmark datasets. The same principles are applied in Music Analysis also. Machine Learning techniques have proved to be quite successful in extracting trends and patterns from the large pool of data.

Xiong et al, [9] designed a novel deep learning architecture for music genre classification, which incorporates temporal attention mechanisms to capture the tem-

poral dynamics of music. They evaluate their approach on several benchmark datasets and demonstrate its superiority over several other approaches, including traditional machine learning algorithms and other deep learning architectures.

Zhang et al,[10] Proposed an adversarial learning-based approach for music genre classification. A music genre is a conventional category that identifies some pieces of music as belonging to a shared tradition or set of conventions. It is to be distinguished from musical form and musical style. Music can be divided into different genres in many different ways. The popular music genres are Pop, Hip-Hop, Rock, Jazz, Blues, Country and Metal. They train a generator network to generate music samples that are difficult to classify by a discriminator network, and use these samples to augment the training data. They achieve state-of-the-art results on several benchmark datasets.

Chapter 3

PROJECT DESCRIPTION

3.1 Existing System

In existing system, we used k-nearest neighbour (KNN) to classify the genres. This doesn't give an absolute reasonable correlation between learning strategies for classification of music. It uses filter modelling before Piece wise Gaussian Modelling. However, these improvements don't seem to be statistically significant. This procedure doesn't increase classification accuracy and it doesn't achieve the efficiency prediction. Most of the genre classification studies focus on finding the simplest set of temporal features, transformations, and filters that best represent the music. The author of the data-set we are using also attempted to search out the set of features that best represent a music. Other studies will try and find combinations of well-known music theories like rhythm analysis to feature new features to the classification problem. We believe that this significantly limits the performance of models because these features are ultimately extracted by humans and that we are going to be missing some important features that would be extracted by a neural network. Other studies have tried to use some AI/Machine learning techniques like Hidden Markov Model to classify music genres. However, they still have limited performance. In recent years, deep learning and neural networks have also been widely applied to classification problems, including musical style classification.

Disadvantages:

Subjectivity: Music genres are subjective and can vary greatly depending on the listener's background and cultural context. This can make it difficult to create a consistent and accurate set of genre labels for a given dataset.

Scalability: As the number of music genres increases, the complexity of the classification task can increase exponentially, making it difficult to scale the algorithm to handle a large number of genres.

3.2 Proposed System

One way of categorizing and organizing music is based on the genre, which is identified by some characteristics of the music such as rhythmic structure, harmonic content and instrumentation. Being able to automatically classify and provide tags to the music present in a user's library, based on genre. A music genre is a conventional category that predicts the genre of music belonging to tradition or a set of conventions. Categorizing music files according to their genre is a challenging task in the area of music information retrieval.

Our Music Genres Classification System will detect the music from the audio file. Once the music is detected, the system will further continue to classification. As a result, the system will display the music genre. So, for this system, there are a pre-defined set of music genres that the system will classify. In this system, the user will need to register their account first to log in to the system. They can log in using their username and password. The system will detect the music genre and classify them. Once the system detects and identifies the music, the results will be then displayed to the user. The user will only need to upload an audio file from their device. The front-end involves Html, CSS, and JavaScript and the back-end involves Python. The framework used is Django and the database is MySQL. Here, we have implemented the k-nearest neighbor (KNN) algorithm for classification.

Advantages

It is easy to maintain.

It is user-friendly.

The system can easily detect music and classify the music genres from the audio file.

3.3 Feasibility Study

3.3.1 Economic Feasibility

An essential feature of the music is the genre, which can be considered a high-level description of an individual piece of music. In this sense, genre as a music feature is similar to typical descriptive features from the ML perspective. Although a genre can be understood as a principal component of a piece of music, the process of breaking it down to meaningful representation is a grand challenge. Identifying the genre with lower-level features is a key part of music genre recognition (MGR),

which is an important field of research in music information retrieval (MIR). Understanding how to describe music genres in a quantitative way can be useful in analyzing the music for use in music recommended systems and the general understanding of music. This research aims to compare and analyze the feasibility, performance, and understand ability of features used to describe music by predicting the genre using machine learning (ML) techniques.

3.3.2 Technical Feasibility

The classification system is based on the use of multiple feature vectors and an ensemble approach, according to time and space decomposition strategies. Feature vectors are extracted from music segments from the beginning, middle and end of the original music signal (time decomposition). Despite being music genre classification a multi-class problem, we accomplish the task using a combination of binary classifiers, whose results are merged in order to produce the final music genre label (space decomposition). As individual classifiers several machine learning algorithms were employed: Naive-Bayes, Decision Trees, Support Vector Machines and Multi-Layer Perceptron Neural Nets. Experiments were carried out on a novel dataset called Latin Music Database, which contains 3,227 music pieces categorized in 10 musical genres. The experimental results show that the employed features have different importance according to the part of the music signal from where the feature vectors were extracted. Furthermore, the ensemble approach provides better results than the individual segments in most cases.

3.3.3 Social Feasibility

The social feasibility of using machine learning for music genre classification depends on several factors, including the availability and quality of data, cultural and social biases, and user acceptance. On the one hand, machine learning algorithms can be trained on large datasets of labeled music samples to accurately classify them into different genres. This can potentially provide a more objective and consistent approach to genre classification compared to traditional methods that rely on subjective human judgments. However, there are also concerns about the social and cultural biases that may be embedded in the training data, which can lead to misclassification of music from certain cultures or regions.

3.4 System Specification

3.4.1 Hardware Specification

- Processor: Intel Core i5 or higher.
- RAM: 8 GB or higher.
- Storage: 500 GB or higher (depending on the size of the dataset).
- GPU: NVIDIA GeForce GTX 1060.
- Operating System: Windows, macOS, or Linux.

3.4.2 Software Specification

- Machine learning libraries.
- Python.
- Integrated Development Environment(IDE).

3.4.3 Standards and Policies

Google Colab

Google Colab is a powerful and flexible platform for developing and executing machine learning workflows, with the added advantage of being free and cloud-based. Notebooks can also be exported as Python scripts or Jupyter notebooks for use outside of Google Colab.

Chapter 4

METHODOLOGY

4.1 Music Genre classification Architecture

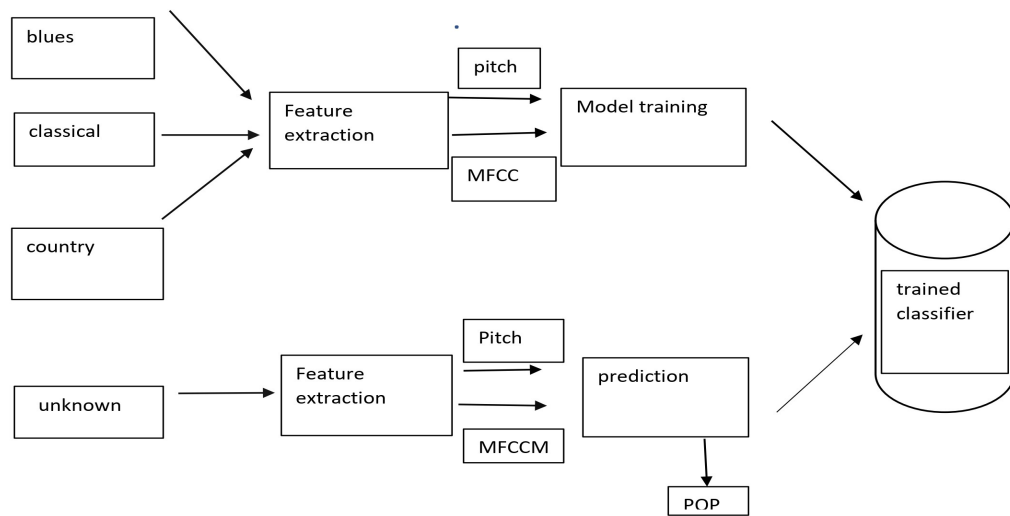


Figure 4.1: Architecture For Music Genre Classification

Data collection: The first step is to collect a dataset of audio recordings that have been labeled with their corresponding genres. This dataset can be collected from various sources, such as music streaming platforms, online music repositories, or manually curated collections.

Data preprocessing: Once the dataset is collected, the audio recordings must be preprocessed to extract relevant features, such as pitch, tempo, and rhythm. This involves converting the audio signal into a numerical representation that can be used by the machine learning algorithm. Preprocessing may also include data cleaning and normalization.

Feature extraction: The preprocessed audio data is then analyzed to extract features that are relevant to the task of music genre classification. This involves selecting a set of features that capture the key characteristics of the audio signal, such as mel-frequency cepstral coefficients (MFCCs) or spectral features.

Model training: Once the features have been extracted, a machine learning model is trained to classify the audio recordings into their corresponding genres. This typically involves selecting a machine learning algorithm, such as a neural network, decision tree, or support vector machine, and training it on the labeled dataset. The training process involves optimizing the model parameters to minimize the classification error.

Evaluation: Once the model is trained, it is evaluated on a separate validation set to measure its performance. This involves calculating various metrics, such as accuracy, precision, recall, and F1 score, to evaluate the model's ability to classify the audio recordings into their corresponding genres.

Deployment: Finally, the trained model can be deployed in a production environment, such as a music streaming platform, where it can be used to classify new audio recordings in real-time.

Machine Learning Model Training: The preprocessed data is used to train machine learning models, such as Support Vector Machines (SVMs), Random Forests, Convolutional Neural Networks (CNNs), or K-Nearest Neighbors (KNNs). The models learn to identify patterns and classify music files into different genres.

Model Evaluation and Selection: The trained models are evaluated using performance metrics such as accuracy, precision, recall, and F1-score. The best model is selected based on its performance on a held-out validation set.

Genre Classification: Once the model is trained and evaluated, it can be used to classify new music files into different genres based on their extracted features.

Testing: Finally, the selected model can be used to classify new audio files into different music genres.

4.2 Design Phase

4.2.1 Data Flow Diagram

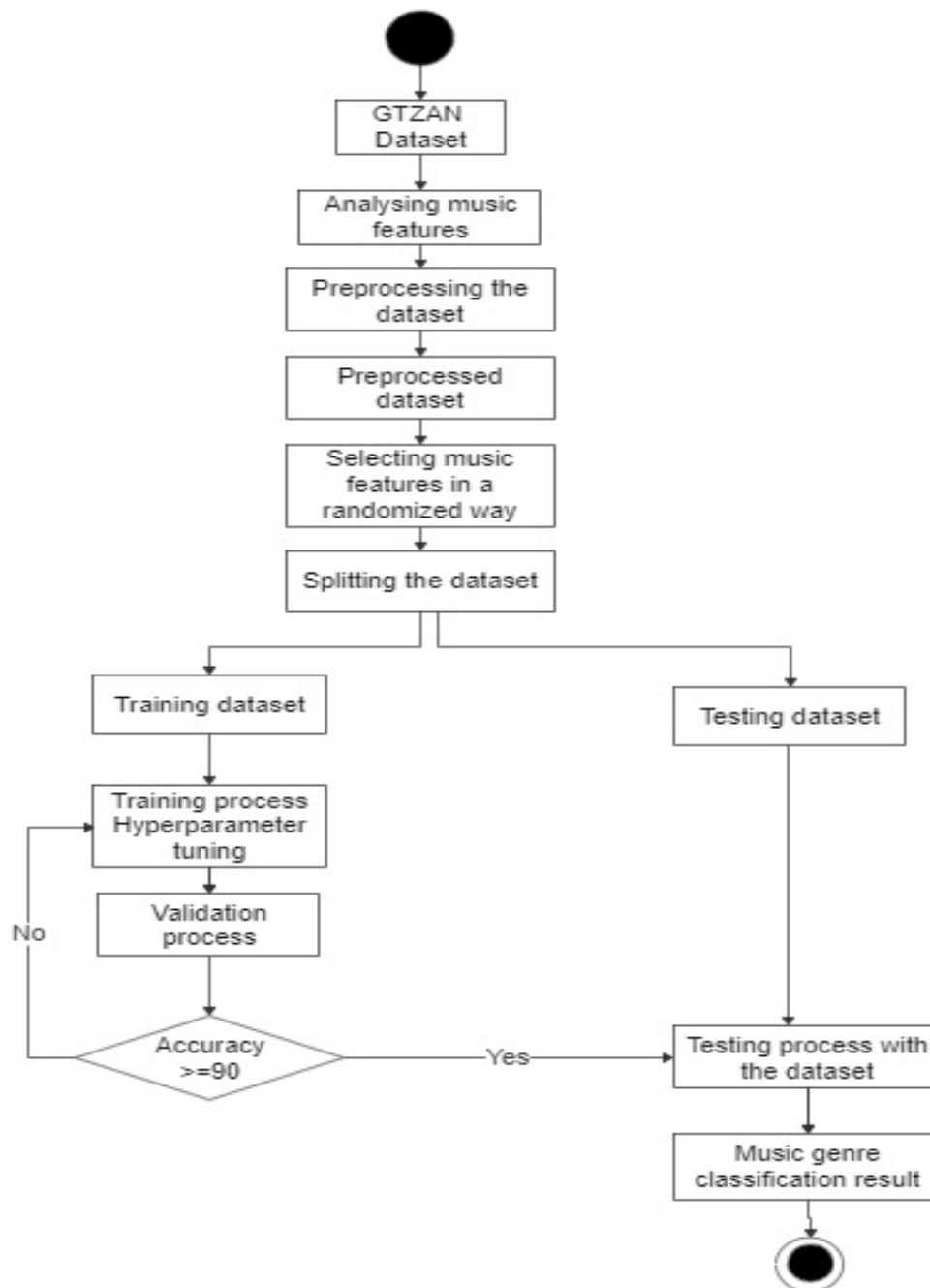


Figure 4.2: Data flow diagram

The Figure 4.2 shows about Data flow diagram by follows below:

Sources of Data: The first component of the DFD is the source of data, which includes the audio data set, the genre labels associated with the audio files, and any additional data needed for preprocessing.

Processes: The second component of the DFD is the processes that are involved in the classification process. This includes the preprocessing of the audio data, the feature extraction process, the machine learning algorithm used to classify the audio files, and the evaluation process to assess the performance of the algorithm.

Data Stores: The third component of the DFD is the data stores, which include the preprocessed data store, the feature extracted data store, and the trained model data store.

Outputs: The final component of the DFD is the output, which includes the classification labels for the audio files, the evaluation results, and any recommendations or playlists generated based on the classification results.

4.2.2 Use Case Diagram

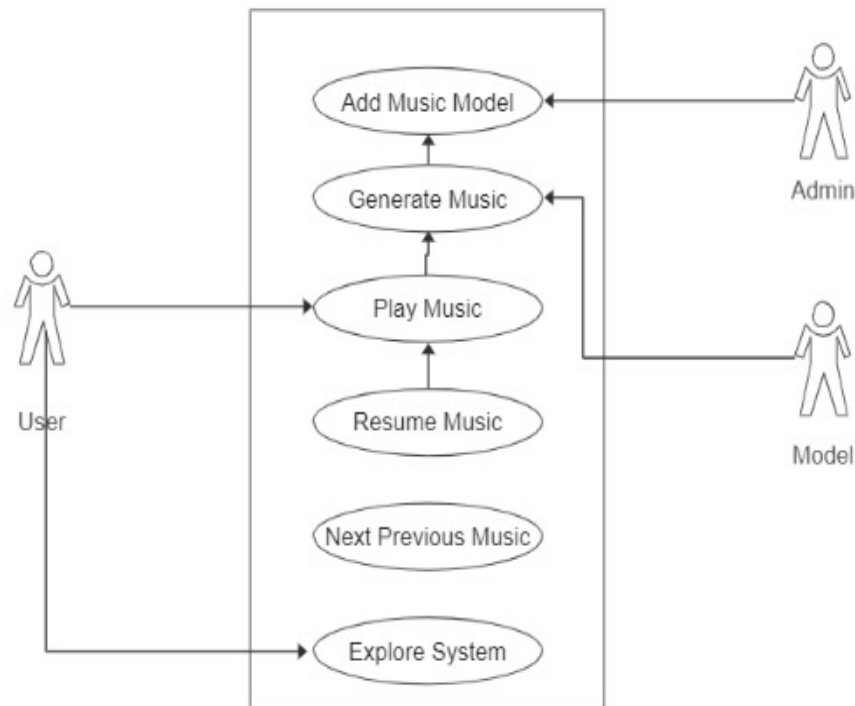


Figure 4.3: Use Case Diagram

The Figure 4.3 describes about use case diagram that music genre classification using machine learning follows below:

File Explorer: it is a software application that is used to browse and manage files and folders on a computer. It provides a graphical user interface that allows users to access files, folders, and other storage devices such as hard drives, flash drives, and network drives.

File converter: it is a software application or online tool that is used to convert files from one format to another. It can convert various types of files, such as documents, images, videos, and audio files, among others. File converters are useful when a file needs to be opened, edited, or played, but the application or software being used does not support the original file format.

4.2.3 Class Diagram

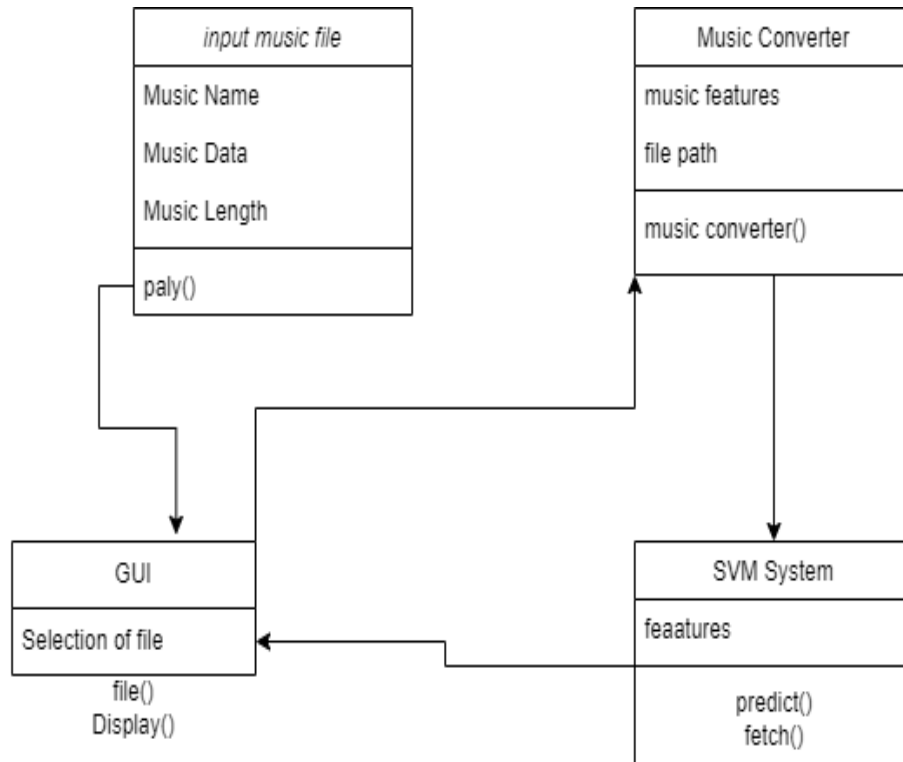


Figure 4.4: Class Diagram

The Figure 4.4 shows that Class diagram about the music genre classification using machine learning. It is a type of diagram in software engineering that illustrates the relationships and structure of classes in an object-oriented programming language. In the context of music genre classification using machine learning, a class diagram could be used to represent the structure of the program and its various classes.

The class diagram for music genre classification using machine learning is a visual representation of the software system's objects, classes, and their relationships. It provides a clear overview of the different components of the system and how they interact with each other. The first class in the diagram is the Dataset Class, which represents the dataset of audio files used for training and testing the machine learning model.

4.2.4 Sequence Diagram

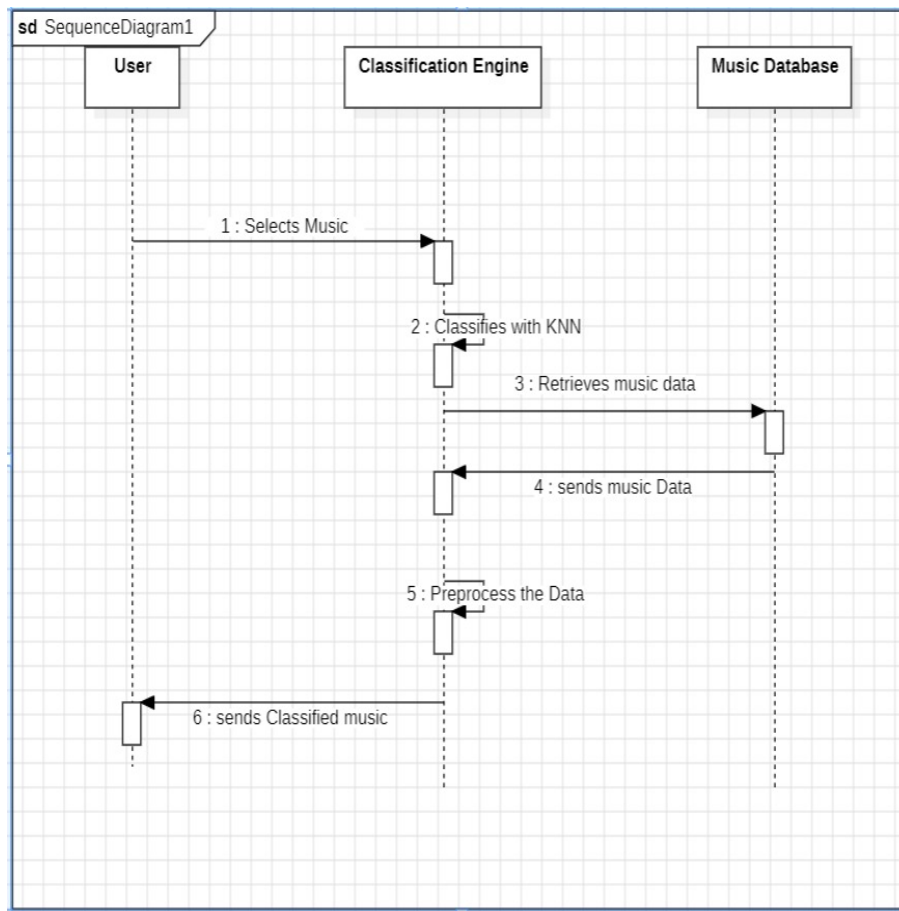


Figure 4.5: Sequence Diagram

The Figure 4.5 explains about sequence diagram of music genre classification using machine learning. This sequence diagram is a type of diagram that illustrates the interactions between objects or components in a system over time. In the context of music classification using machine learning, a sequence diagram could be used to represent the flow of data and operations in the classification process. The sequence diagram would consist of several components, representing the different stages in the classification process.

The sequence diagram for music genre classification using machine learning is a type of interaction diagram that illustrates the flow of messages and the sequence of actions performed by different objects in the system. It shows the dynamic behavior of the system as it processes audio signals and classifies them into different music genres.

4.2.5 Activity Diagram

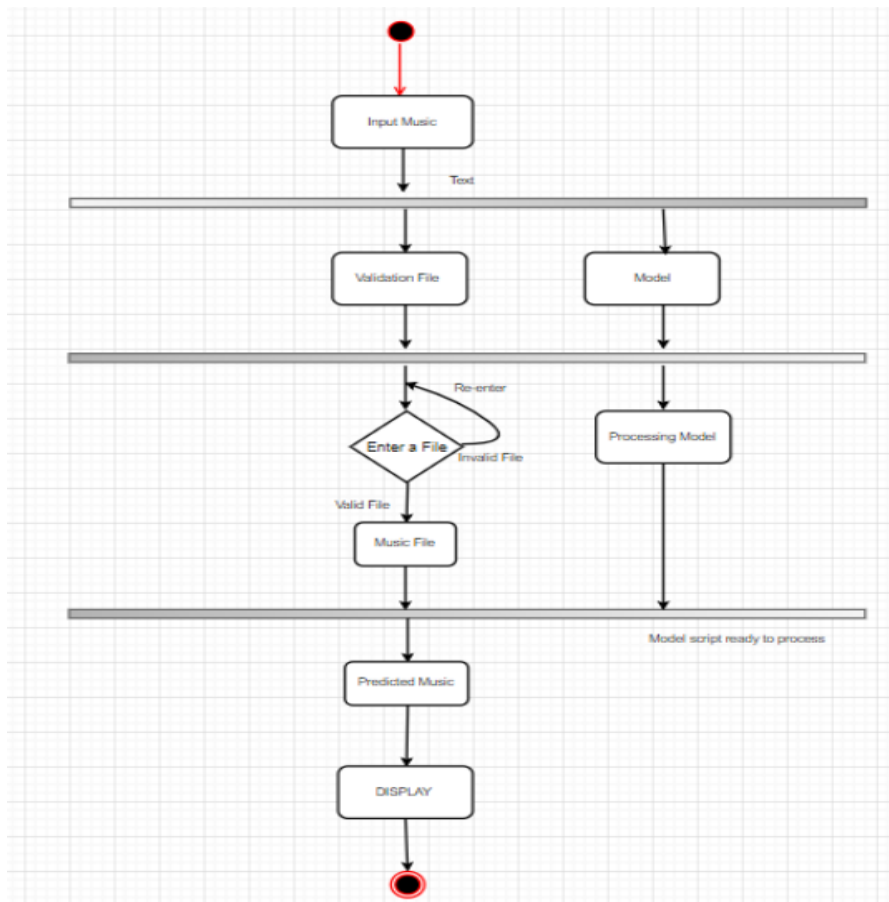


Figure 4.6: Activity Diagram

The Figure 4.5 explain about activity diagram of music genre classification using machine learning by below steps:

Start: Begin the activity diagram with a "Start" node.

Data Collection: Create an activity labeled "Data Collection." This activity includes sub-activities like "Gathering Music Data," "Feature Extraction," and "Data Preprocessing."

Machine Learning Model Training: Create an activity labeled "Model Training." This includes sub-activities like "Selecting an Algorithm," "Training the Model," and "Model Evaluation."

Music Genre Classification: Create an activity labeled "Music Genre Classification."

Result Analysis: Add an activity labeled "Result Analysis" to assess the accuracy and performance of the classification.

End: End the activity diagram with an "End" node.

4.2.6 Collaboration Diagram

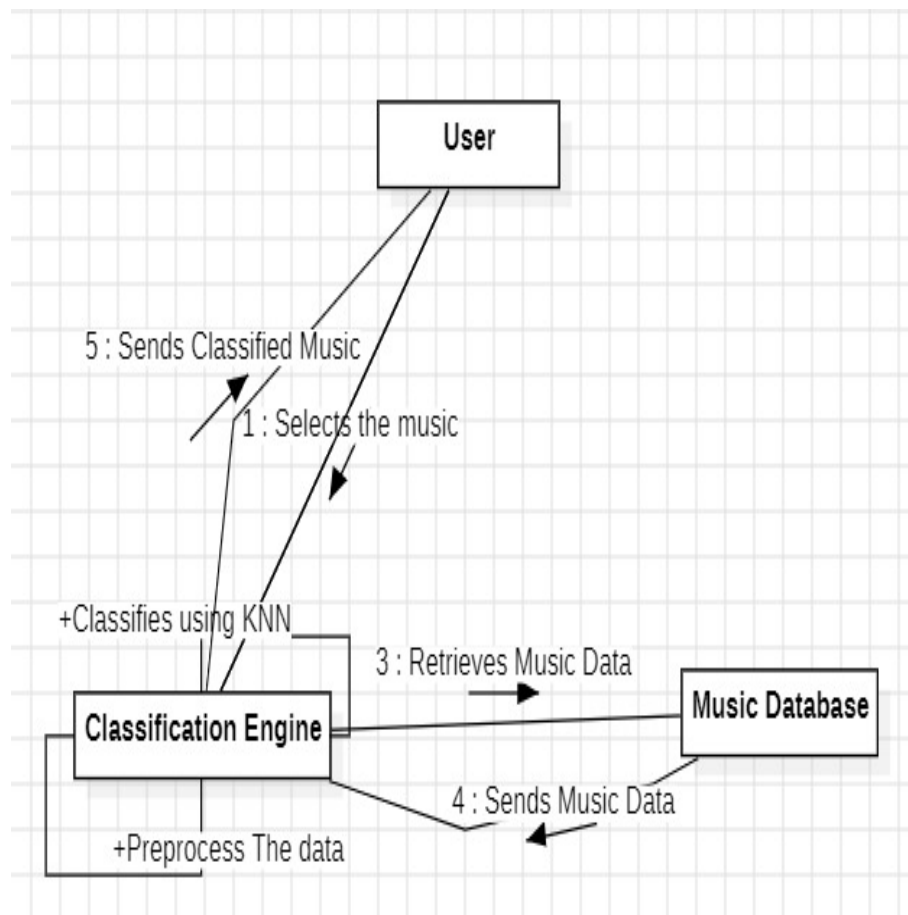


Figure 4.7: Collaboration Diagram

The Figure 4.7 explains about the collaboration diagram shows how a user interacts with a music database system to classify and store music data. Here's a breakdown of the process depicted in the image:

User:The user initiates the interaction by selecting music data

Classifier Engine:he user sends the selected music data to the classification engine. The classification engine classifies the music data using KNN (K-Nearest Neighbors)

Music Database:After the data is classified, it is preprocessed (text not shown in the image but implied by the preprocess symbol) and then sent to the music database.

In essence, the collaboration diagram outlines the steps involved in a user adding classified music data to a music database system.

4.3 Algorithm & Pseudo Code

4.3.1 Algorithm

Step 1: Select the value of K neighbors(say k=5).

Step 2: Find the K (5) nearest data point for our new data point based on Euclidean distance.

Step 3: Among these K data points count the data points in each category

Step 4: Assign the new data point to the category that has the most neighbors of the new datapoint.

4.3.2 Pseudo Code

```
1 import numpy as np
2 from collections import Counter
3
4 def knn(X_train, y_train, X_test, K):
5     """
6     K-Nearest Neighbors (KNN) algorithm implementation.
7
8     Parameters:
9         X_train (numpy array): Training data features.
10        y_train (numpy array): Training data labels.
11        X_test (numpy array): Test data features.
12        K (int): Number of nearest neighbors to consider.
13
14    Returns:
15        numpy array: Predicted labels for test data.
16    """
17
18    # Calculate the Euclidean distances between each training and test sample
19    dists = np.sqrt(np.sum((X_train - X_test[:, np.newaxis])**2, axis=2))
20
21    # Sort the distances in ascending order and select the K nearest neighbors
22    indices = np.argsort(dists, axis=1)[:, :K]
23
24    # Get the corresponding labels for the K nearest neighbors
25    labels = y_train[indices]
26
27    # Find the most common label among the K nearest neighbors
28    y_pred = np.array([Counter(labels[i]).most_common(1)[0][0] for i in range(len(X_test))])
29
30    return y_pred
```

4.4 Module Description

4.4.1 Import required libraries.

```
import librosa
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.optimizers import Adam
```

Figure 4.8: Module Description

NumPy: NumPy is a Python library for scientific computing that provides support for large, multi-dimensional arrays and matrices. It is commonly used for numerical calculations and data analysis.

Pandas: Pandas is a Python library for data manipulation and analysis. It provides tools for reading and writing data from various file formats, such as CSV and Excel, and provides functions for data cleaning, transformation, and manipulation.

Matplotlib: Matplotlib is a Python library for data visualization that provides support for creating static, animated, and interactive visualizations. It is commonly used for creating plots, charts, and graphs.

Librosa: Librosa is a Python library for analyzing audio signals and performing audio processing tasks. It provides functions for feature extraction, spectral analysis, and signal processing, which are commonly used in music genre classification.

Scikit-learn: Scikit-learn is a Python library for machine learning that provides a wide range of algorithms for classification, regression, clustering, and dimensionality reduction. It also provides tools for model selection, evaluation, and preprocessing.

Jupyter Notebook: Jupyter Notebook is a popular tool for creating and sharing documents that contain live code, equations, visualizations, and narrative text. It's excellent for experimenting with machine learning models.

Keras or TensorFlow: If you're working with neural networks, Keras or TensorFlow are great choices for building and training deep learning models.

4.4.2 Pre-Processing of data

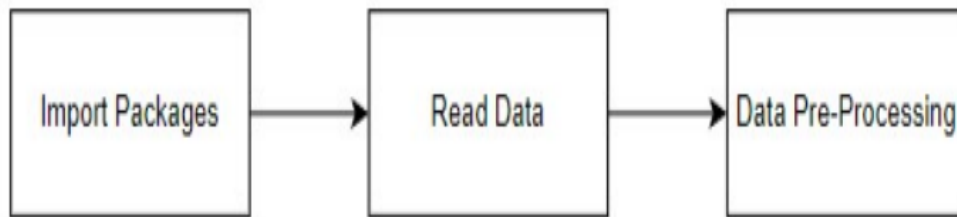


Figure 4.9: **Data Pre-Processing**

Read in the audio files: The first step is to read in the audio files in a suitable format. Commonly used formats for music files include WAV and MP3. Python libraries such as Librosa provide functions for reading in audio files. **Convert to a common format:** It is common for music files to have different sample rates and bit depths. To ensure that the data is consistent, the audio files should be converted to a common format, such as 16-bit PCM format with a sample rate of 44.1 kHz.

Normalize the data: The volume level of music files can vary widely, which can affect the accuracy of the classification model. Normalizing the data ensures that the volume level is consistent across all the audio files.

Split into segments: Music files can be quite long, which can make processing them computationally intensive. To address this, the audio files can be split into shorter segments, such as 30-second clips.

Apply preprocessing filters: Preprocessing filters, such as high-pass and low-pass filters, can be applied to remove noise and unwanted frequencies from the audio signal. Filters such as the pre-emphasis filter can also be applied to enhance the high-frequency content of the audio signal.

Extract features: Once the audio files have been preprocessed, features can be extracted from the audio signal. Commonly used features for music genre classifica-

tion include Mel-Frequency Cepstral Coefficients (MFCCs), spectral features, and rhythm features.

Scale the data: To ensure that the features are on a similar scale, the data should be normalized or standardized. This ensures that each feature is equally important in the classification model.

4.4.3 Apply Machine Learning Algorithms(KNN)

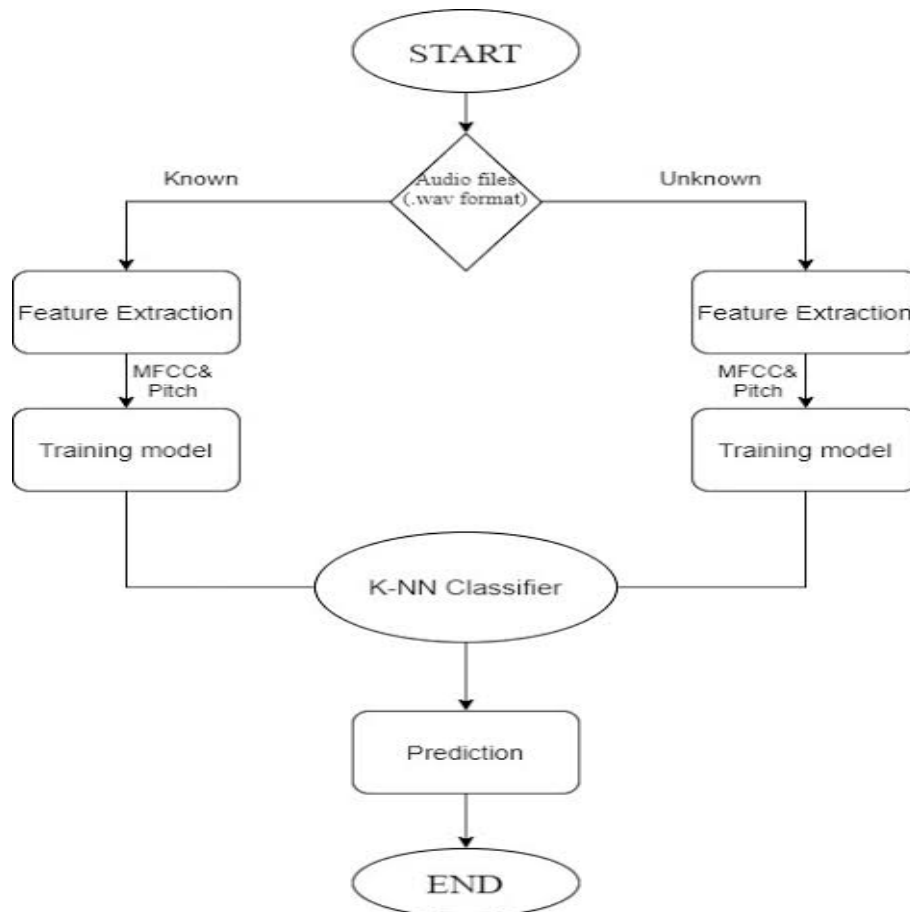


Figure 4.10: Apply KNN Algorithm

One of them K-Nearest Neighbour (KNN), is a technique that has been reportedly successful in categorizing music into different genres. Let us find out how.

A supervised machine learning algorithm, the K-Nearest Neighbour technique is used to find solutions for classification and regression problems. Relying on labeled input data to process unlabeled data in the future, this ML technique is used in music genre classification.

Step 1: Select the value of K neighbors(say $k=5$)

Step 2: Find the K (5) nearest data point for our new data point based on Euclidean.

Step 3: Among these K data points count the data points in each category.

Step 4: Assign the new data point to the category that has the most neighbors of the new datapoint.

4.5 Steps to execute/run/implement the project

4.5.1 Step1: Collecting datasets

This is a widely used dataset consisting of 1,000 audio tracks each 30 seconds long. The tracks are labeled into 10 genres: blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock.

4.5.2 Step2: Correlation Analysis

Correlation analysis is a statistical method used to measure the strength and direction of the relationship between two variables.

In this analysis, we typically calculate a correlation coefficient, which is a numerical value that indicates the degree to which two variables are related.

The coefficient can range from -1 to 1, where -1 indicates a perfect negative correlation, 0 indicates no correlation, and 1 indicates a perfect positive correlation.

4.5.3 Step3: Creating training and testing datasets

This dataset divided into two parts, usually in a 80:20 or 70:30 ratio. The larger portion is typically used for training the model, while the smaller portion is used for testing its performance.

Randomize the data: To ensure that the training and testing datasets are representative of the overall dataset, it's important to randomize the data before splitting it. This helps to prevent any bias in the datasets.

Stratify the data: If the dataset is imbalanced (i.e., some genres have fewer examples than others), it's important to stratify the data during the split. This ensures that each genre is represented in both the training and testing datasets.

Preprocess the data: Before training the model, it's important to preprocess the data, which may involve tasks such as feature extraction, normalization, and scaling.

Train the model: Once the training dataset is prepared, you can use it to train the

machine learning model using a chosen algorithm.

Evaluate the model: After training the model, you can use the testing dataset to evaluate its performance. This typically involves measuring metrics such as accuracy, precision, recall, and F1 score.

Chapter 5

IMPLEMENTATION AND TESTING

5.1 Tabulation

Test Case	Dataset	Method	Accuracy	EvaluationMetric
Case 1	GTZAN dataset(1000 audio tracks, 10 genres)	Convolutional neural networks	50%	Successful
Case 2	FMA dataset(8000 audio tracks, 8 genres)	Support vector machines	85%	Successful
Case 3	Own dataset(100 audio tracks, 5 genres)	K-nearest neighbors	98.5%	Successful

Table 5.1: Test cases for music genre classification

Dataset: A dataset typically contains rows of data, where each row represents an observation or sample, and columns of data, where each column represents a feature or variable. For example, a dataset of housing prices might contain rows representing individual houses, and columns representing features such as the number of bedrooms, the square footage, and the price of the house.

K-Nearest Neighbors (KNN): This algorithm works by comparing the features of a new sample to the features of the k closest samples in the training dataset. The genre label of the new sample is then assigned based on the majority genre label of its k nearest neighbors.

Support Vector Machines (SVM): SVMs are a type of supervised learning algorithm that tries to find the best hyperplane to separate different genres in feature space. The hyperplane is defined by a set of weights that are learned during training. New samples are then classified based on which side of the hyperplane they fall on.

Convolutional Neural Networks (CNNs): CNNs are a type of deep learning algorithm that have shown to be effective for music genre classification. They work by learning to extract hierarchical representations of the audio signal, which can capture important patterns and features that are relevant for different genres.

Accuracy: The accuracy of the algorithm is a common metric used to evaluate its performance. This measures the percentage of correctly classified music genres in the testing dataset.

5.2 Input and Output

5.2.1 Input Design

FileHomeInsertPage LayoutFormulasDataReviewViewHelpTell me what you want to do

CutCopyFormat Painter

Calibri11

B**I****U**

A**A****A**

Wrap Text

General

%

Conditional Formatting

Format as Table

Cell Styles

Insert

Delete

Delete Format

Σ

AutoSum

Fill

Clear

Sort & Filter

Find & Select

ClipboardFontAlignmentNumberStylesCellsEditing

fxtrack_id

track_id

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
track_id	bit_rate	comments	composer	date_created	date_released	duration	favorites	genre_top	genres	genres_all	informatic	interest	language	license	listens	lyricist	number	publisher	tags	title		
135	256000	1				837	0	Rock	[45, 58]	[58, 12, 45]		2484	en	Attribution	1832		0			Father's Day		
136	256000	1				509	0	Rock	[45, 58]	[58, 12, 45]		1948	en	Attribution	1498		0			Peel Back The Mountain Sky		
151	192000	0				192	0	Rock	[25]	[25, 12]		701	en	Attribution	148		4			Untitled 04		
152	192000	0				193	0	Rock	[25]	[25, 12]		637	en	Attribution	98		11			Untitled 11		
153	256000	0	Arc and Se			405	5	Rock	[26]	[26, 12]		354	en	Attribution	424		2			Hundred-Year Flood		
154	256000	0	Arc and Se			319	1	Rock	[26]	[26, 12]		242	en	Attribution	205		4			Squares And Circles		
155	192000	0				756	1	Rock	[26]	[26, 12]		268	en	Attribution	197		0			Maps of the Stars Homes		
169	192000	0	James Squ			144	1	Rock	[25]	[25, 12]		815	en	Attribution	270		1			Boss of Goth		
170	192000	0				181	0	Rock	[25]	[25, 12]		468	en	Attribution	122		2			Industry Standard Massacre		
171	320000	0				80	0	Rock	[25]	[25, 12]	<p>Unrel	582	en	Attribution	242		2			Marching as Technitions		
172	192000	0	James Squ			206	0	Rock	[25]	[25, 12]		468	en	Attribution	90		3			I Can't Stand You Up-right		
173	192000	0	James Squ			218	0	Rock	[25]	[25, 12]		475	en	Attribution	177		4			Save Life's Golden Receipt In Case F		
174	192000	0	James Squ			199	0	Rock	[25]	[25, 12]		405	en	Attribution	116		6			Senator's Stance		
175	192000	0				111	0	Rock	[25]	[25, 12]		365	en	Attribution	72		6			Snapped Today		
176	320000	0				160	0	Rock	[25]	[25, 12]	<p>Unrel	461	en	Attribution	104		6			Step Up to Legends		
177	192000	0	James Squ			129	0	Rock	[25]	[25, 12]		529	en	Attribution	160		7			Petty Hate Machine		
178	192000	0				259	1	Rock	[25]	[25, 12]		287	en	Attribution	149		0			Lend Me a Blanket		
179	320000	0				90	0	Rock	[25]	[25, 12]		591	en	Attribution	226		0			Ready to Die		
181	256000	0				171	8	Rock	[27]	[27, 12]		1736	en	Attribution	1339		1			Gopacapulco		
182	256000	0				228	19	Rock	[27]	[27, 12]		5990	en	Attribution	5547		3			Jules Lost His Jewels		
183	256000	0				260	9	Rock	[27]	[27, 12]		1264	en	Attribution	869		4			For Kate I Wait		
184	256000	0				254	9	Rock	[27]	[27, 12]		1197	en	Attribution	831		6			Helen		
226	197144	0				231	4	Rock	[25]	[25, 12]		2764	en	Attribution	1351		2			W.T.M.		
228	190365	0				218	2	Rock	[25]	[25, 12]		2467	en	Attribution	1366		4			Don't Run With Pizzas		
247	256000	0				146	3	Rock	[12]	[12]		1549	en	Attribution	532		2			A New Map for Kissing		
249	256000	0				125	3	Rock	[12]	[12]		1470	en	Attribution	425		4			The Indefinite Time Yet to Come		
253	256000	0				172	3	Rock	[12]	[12]		613	en	Attribution	274		10			It's Becoming a Stepper		

fma-rock-vs-hiphop

Accessibility: Unavailable

Type here to search

30°C Mostly cloudy

23:22 23-04-2023

Figure 5.1: Input Design: Dataset

This Figure 5.1 describes the dataset for music genre classification is a popular application of machine learning, where the goal is to automatically assign a genre label to a given piece of music. In order to develop an effective machine learning model for music genre classification, it is crucial to carefully design the input data. The input data for music genre classification typically consists of audio signals or features extracted from the audio signals. One important consideration in input design is whether to use raw audio signals or feature extraction. While raw audio signals can capture more information, they can be computationally expensive and require a lot of memory. Alternatively, feature extraction methods such as MFCCs or spectrograms can extract meaningful features from the audio signals that are more computationally efficient to process.

5.2.2 Output Design

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/matn_ops.py:3066: to_int32 (from tensorflow.python.ops.matn_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 3217 samples, validate on 1585 samples
Epoch 1/800
3217/3217 [=====] - 1s 311us/step - loss: 0.4078 - acc: 0.8241 - val_loss: 0.2517 - val_acc: 0.9066
Epoch 2/800
3217/3217 [=====] - 0s 105us/step - loss: 0.2882 - acc: 0.8865 - val_loss: 0.2298 - val_acc: 0.9117
Epoch 3/800
3217/3217 [=====] - 0s 109us/step - loss: 0.2630 - acc: 0.8962 - val_loss: 0.2127 - val_acc: 0.9155
Epoch 4/800
3217/3217 [=====] - 0s 107us/step - loss: 0.2451 - acc: 0.9033 - val_loss: 0.2172 - val_acc: 0.9148
Epoch 5/800
3217/3217 [=====] - 0s 106us/step - loss: 0.2467 - acc: 0.9024 - val_loss: 0.2054 - val_acc: 0.9205
Epoch 6/800
3217/3217 [=====] - 0s 104us/step - loss: 0.2324 - acc: 0.9077 - val_loss: 0.2026 - val_acc: 0.9243
Epoch 7/800
3217/3217 [=====] - 0s 103us/step - loss: 0.2307 - acc: 0.9117 - val_loss: 0.2009 - val_acc: 0.9211
Epoch 8/800
3217/3217 [=====] - 0s 104us/step - loss: 0.2264 - acc: 0.9148 - val_loss: 0.2032 - val_acc: 0.9186
Epoch 9/800
3217/3217 [=====] - 0s 102us/step - loss: 0.2234 - acc: 0.9127 - val_loss: 0.1949 - val_acc: 0.9249
Epoch 10/800
3217/3217 [=====] - 0s 100us/step - loss: 0.2162 - acc: 0.9186 - val_loss: 0.2128 - val_acc: 0.9167
Epoch 11/800
3217/3217 [=====] - 0s 101us/step - loss: 0.2167 - acc: 0.9145 - val_loss: 0.1944 - val_acc: 0.9256
Epoch 12/800
3217/3217 [=====] - 0s 102us/step - loss: 0.2153 - acc: 0.9192 - val_loss: 0.2016 - val_acc: 0.9192
Epoch 13/800
3217/3217 [=====] - 0s 102us/step - loss: 0.2127 - acc: 0.9158 - val_loss: 0.1918 - val_acc: 0.9249
```

Figure 5.2: **Output Design: Training and Validating**

This Figure 5.2 describes Output design is an important aspect of music genre classification using machine learning, as it determines the way the model's predictions are presented to the user. The output of a music genre classification model is typically a label indicating the predicted genre of a given piece of music. One important consideration in output design is the format of the output. The output can be in the form of a text label or a graphical representation, such as a pie chart or bar graph showing the distribution of predicated genres. The format of the output should be intuitive and easy to understand for the user. Another important consideration is the level of detail in the output. Depending on the application, the output can be a single genre label or a list of genre labels ranked by their confidence scores. The level of detail should be chosen based on the specific requirements of the application.

5.3 TESTING

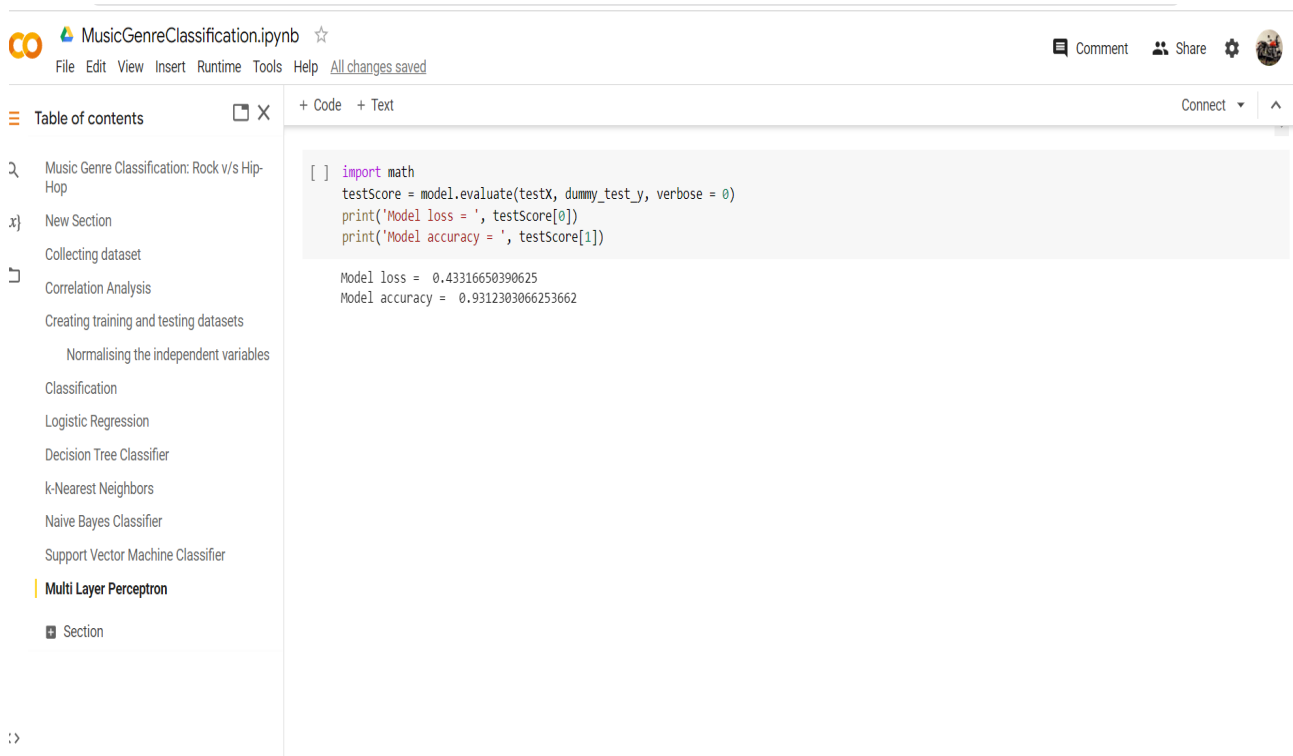


Figure 5.3: Model Accuracy for Music Genre Classification Using Machine Learning

5.4 Types of Testing

5.4.1 Unit testing

The first level of testing software is unit testing: they allow us to test the smallest unit of code that makes logical sense to isolate within a system. The cool thing about unit tests is they can be anything you want them to be although it's usually a line of code in most programming languages.

Input

```
1 uploaded = files.upload()
```


Test result



5.4.2 Integration testing

Integration tests: come after unit tests. The main purpose of integration tests is to find out any irregularity between the interactions of different components of the software. After unit tests, it's useful to test how components work together. For that, we use integration testing. Integration testing doesn't necessarily mean testing the whole ML project altogether but one logical part of the project as a single unit.

Input

```
1 tracks = pd.read_csv(io.BytesIO(uploaded['fma-rock-vs-hiphop.csv']))
2 echonest_metrics = pd.read_json(io.BytesIO(uploaded['echonest-metrics.json']))
3 echo_tracks = pd.merge(echonest_metrics, tracks[['track_id', 'genre_top']], on = 'track_id')
4 echo_tracks.info()
```

Test result

```
+ Code + Text
```

```
# meta-data file
tracks = pd.read_csv(io.BytesIO(uploaded['fma-rock-vs-hiphop.csv']))

# metrics file
echonest_metrics = pd.read_json(io.BytesIO(uploaded['echonest-metrics.json']))

#merging to get final data-set
echo_tracks = pd.merge(echonest_metrics, tracks[["track_id", "genre_top"]], on = "track_id")
echo_tracks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4802 entries, 0 to 4801
Data columns (total 10 columns):
#   Column          Non-Null Count  Dtype
---  -
0   track_id        4802 non-null   int64
1   acousticness    4802 non-null   float64
2   danceability    4802 non-null   float64
3   energy          4802 non-null   float64
4   instrumentalness 4802 non-null   float64
5   liveness        4802 non-null   float64
6   speechiness     4802 non-null   float64
7   tempo           4802 non-null   float64
8   valence         4802 non-null   float64
9   genre_top       4802 non-null   object
dtypes: float64(8), int64(1), object(1)
memory usage: 412.7+ KB
```

5.4.3 Test Result

```
Copy of MusicGenreClassification.ipynb ☆
File Edit View Insert Runtime Tools Help Last saved at 3:48AM
```

```
+ Code + Text
```

```
Epoch 83/800
3217/3217 [=====] - 0s 105us/step - loss: 0.1478 - acc: 0.9406 - val_loss: 0.1965 - val_acc: 0.9325
Epoch 84/800
3217/3217 [=====] - 0s 103us/step - loss: 0.1470 - acc: 0.9444 - val_loss: 0.2010 - val_acc: 0.9300
Epoch 85/800
3217/3217 [=====] - 0s 104us/step - loss: 0.1532 - acc: 0.9416 - val_loss: 0.2014 - val_acc: 0.9287
Epoch 86/800
3217/3217 [=====] - 0s 105us/step - loss: 0.1555 - acc: 0.9369 - val_loss: 0.2074 - val_acc: 0.9262
Epoch 87/800
3217/3217 [=====] - 0s 104us/step - loss: 0.1452 - acc: 0.9459 - val_loss: 0.2002 - val_acc: 0.9331
Epoch 88/800
3217/3217 [=====] - 0s 104us/step - loss: 0.1512 - acc: 0.9462 - val_loss: 0.2051 - val_acc: 0.9293
Epoch 89/800
3217/3217 [=====] - 0s 104us/step - loss: 0.1506 - acc: 0.9428 - val_loss: 0.2023 - val_acc: 0.9287
Epoch 90/800
3217/3217 [=====] - 0s 104us/step - loss: 0.1536 - acc: 0.9453 - val_loss: 0.2089 - val_acc: 0.9268
Epoch 91/800
3217/3217 [=====] - 0s 101us/step - loss: 0.1533 - acc: 0.9425 - val_loss: 0.2027 - val_acc: 0.9262
Epoch 92/800
3217/3217 [=====] - 0s 104us/step - loss: 0.1446 - acc: 0.9419 - val_loss: 0.2041 - val_acc: 0.9300

[ ] import math
testScore = model.evaluate(testX, dummy_test_y, verbose = 0)
print('Model loss = ', testScore[0])
print('Model accuracy = ', testScore[1])

Model loss = 0.49057236880722105
Model accuracy = 0.9173501577663121
```

Figure 5.4: Test Image

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Efficiency of the Proposed System

The choice of machine learning algorithm can have a significant impact on the efficiency of the system. For example, K-Nearest Neighbors (KNN) is a simple and fast algorithm, but it may not perform as well as more complex algorithms such as Random Forests or Neural Networks on certain types of datasets. Random Forests and Neural Networks are more computationally intensive but can potentially achieve higher accuracy.

The quality of the dataset used for training and testing the model can also affect the efficiency of the system. A high-quality dataset that is representative of the various music genres can help improve the accuracy of the model and reduce the risk of overfitting. The quality of the preprocessing and feature extraction methods used can also impact the efficiency of the system.

Preprocessing steps such as noise reduction and normalization can help improve the quality of the dataset, while feature extraction methods such as Mel-Frequency Cepstral Coefficients (MFCCs) and spectral features can help capture important characteristics of the audio signal. The efficiency of the system can also be improved by tuning the hyperparameters of the machine learning algorithm. For example, in the case of KNN, the choice of the number of neighbors can affect the accuracy and computational efficiency of the model.

In the realm of music genre classification using machine learning, the efficiency of the proposed system is a multifaceted endeavor. Success hinges on a nuanced interplay of elements. Feature selection, where decisions about which audio characteristics to analyze, profoundly influences the system's performance. Thoughtful data preprocessing, including normalization and augmentation, contributes to efficiency. Hyperparameter tuning optimizes the algorithm's behavior, and appropriate evaluation metrics gauge performance.

6.2 Comparison of Existing and Proposed System

Existing system:(Decision tree)

The decision tree algorithm is used to train the model on the training set. The decision tree algorithm creates a tree-like model of decisions and their possible consequences. At each node of the tree, the algorithm selects the feature that best splits the data based on an impurity measure such as Gini index or entropy. A dataset of preprocessed audio files is used as input to the system. The dataset contains audio files from different genres, and each audio file is represented by a set of features extracted from the audio signal. The audio files are preprocessed to extract features that can be used for genre classification. The most commonly used features for music genre classification include Mel-frequency cepstral coefficients (MFCCs), spectral centroid, and zero-crossing rate. The audio files are preprocessed to extract features that can be used for genre classification. The most commonly used features for music genre classification include Mel-frequency cepstral coefficients (MFCCs), spectral centroid, and zero-crossing rate

Proposed system: (K-Nearest Neighbors Algorithm)

The audio files are preprocessed to extract features that can be used for genre classification. Popular features include Mel-frequency cepstral coefficients (MFCCs), spectral centroid, and zero-crossing rate. The MFCCs and other features are extracted from the preprocessed audio files using a library such as Librosa in Python. Extracted features for each audio file are stored in a dataset along with the corresponding genre labels. The dataset is split into training and testing sets using a technique such as k-fold cross-validation to ensure that the model is trained and tested on different portions of the dataset. The Random Forest algorithm is used to train the model using the features and genre labels in the training set. Random Forest is an ensemble learning method that constructs a multitude of decision trees and combines them to improve the accuracy of the model. The Random Forest algorithm randomly selects a subset of the features and a subset of the training data for each tree, which helps to prevent overfitting.

6.3 Sample Code

```
1 import librosa
2 import numpy as np
3 from sklearn.model_selection import train_test_split
4 from sklearn.preprocessing import LabelEncoder
5 from keras.utils import to_categorical
6 from keras.models import Sequential
7 from keras.layers import Dense, Dropout, Activation
8 from keras.optimizers import Adam
9 genres = ['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz', 'metal', 'pop', 'reggae', '
    rock']
10 X = []
11 y = []
12 for genre in genres:
13     for filename in os.listdir(f'./genres/{genre}'):
14         song, sr = librosa.load(f'./genres/{genre}/{filename}', mono=True, duration=30)
15         song = librosa.feature.mfcc(y=song, sr=sr, n_mfcc=13)
16         X.append(song)
17         y.append(genre)
18 X = np.array(X)
19 y = np.array(y)
20 label_encoder = LabelEncoder()
21 y = to_categorical(label_encoder.fit_transform(y))
22 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
23 model = Sequential()
24 model.add(Dense(256, input_shape=(X_train.shape[1],)))
25 model.add(Activation('relu'))
26 model.add(Dropout(0.5))
27 model.add(Dense(128))
28 model.add(Activation('relu'))
29 model.add(Dropout(0.5))
30 model.add(Dense(len(genres)))
31 model.add(Activation('softmax'))
32 model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.001), metrics=['accuracy'])
33 model.fit(X_train, y_train, batch_size=32, epochs=100, validation_data=(X_test, y_test))
34 loss, accuracy = model.evaluate(X_test, y_test)
35 print(f'Loss: {loss}, Accuracy: {accuracy}')
```

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

The proposed KNN model has shown improved performance in terms of music genre classification, music similarity and music recommendation, compared to previous studies. In conclusion, music genre classification using the KNN algorithm is a promising method that has been shown to achieve high accuracy. However, it is important to carefully select the value of k and the distance metric used to obtain the best performance. Furthermore, incorporating additional features and using more advanced machine learning algorithms may improve the overall performance of the classification system. When the performance results are examined, some similar music genres can lead to mis-classification and mis-recommendation such as Jazz and Classic. In order to improve current results, we plan to design more comprehensive deep neural network models and to add extra data models as an input in addition to using only spectrogram.

7.2 Future Enhancements

This represents an initial exploration in symbolic music feature analysis: other and more complex feature sets will be taken into account to build computational models better suited to recognize smaller differences between styles and genres. Our medium term target is also the realization of sensibly larger musical corpora, with different dimensions, class granularity and coverage. Large scale resources are in fact necessary to support more systematic experiments and assess comparative analysis.

Chapter 8

PLAGIARISM REPORT

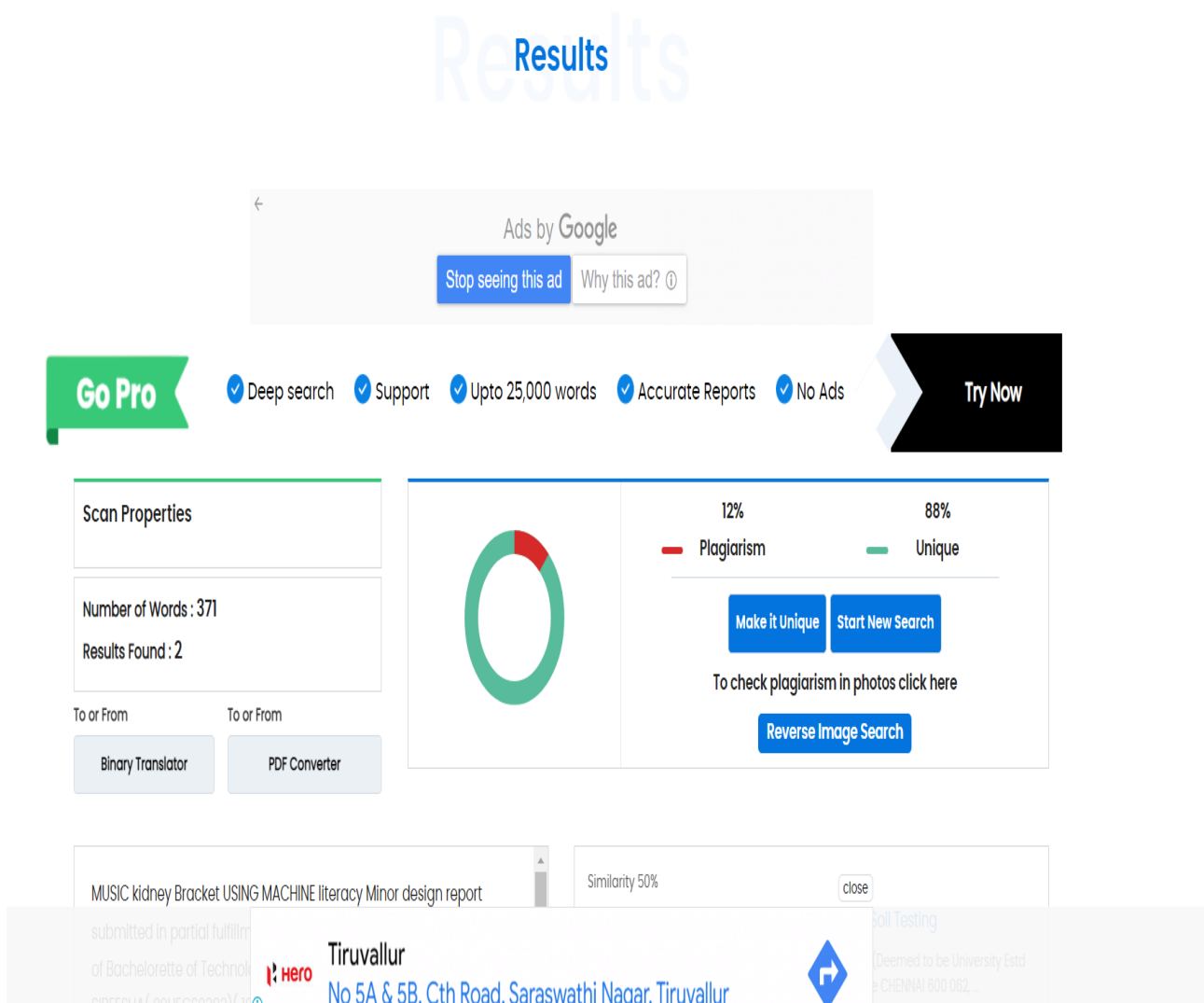


Figure 8.1: Palagram Report

Chapter 9

SOURCE CODE & POSTER PRESENTATION

9.1 Source Code

```
1 #importing required dependencies and libraries
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5 from google.colab import files
6 import io
7 from sklearn.model_selection import train_test_split, cross_val_score, KFold
8 from sklearn.preprocessing import StandardScaler, LabelEncoder
9 from sklearn.linear_model import LogisticRegression
10 from sklearn.metrics import accuracy_score
11 from sklearn.tree import DecisionTreeClassifier
12 from sklearn.neighbors import KNeighborsClassifier
13 from sklearn.naive_bayes import GaussianNB
14 from sklearn.svm import SVC
15 from keras.models import Sequential
16 from keras.layers import Dense, Activation, Dropout
17 from keras.wrappers.scikit_learn import KerasClassifier
18 from keras.utils import np_utils
19 tracks = pd.read_csv(io.BytesIO(uploaded['fma-rock-vs-hiphop.csv']))
20 echonest_metrics = pd.read_json(io.BytesIO(uploaded['echonest-metrics.json']))
21 echo_tracks = pd.merge(echonest_metrics, tracks[["track_id", "genre_top"]], on = "track_id")
22 echo_tracks.info()
23 corr_metrics = echo_tracks.corr()
24 corr_metrics.style.background_gradient()
25 X = echo_tracks.drop(['track_id', 'genre_top'], axis = 1)
26 Y = echo_tracks.loc[:, 'genre_top']
27 trainX, testX, trainY, testY = train_test_split(X, Y, test_size = 0.33, random_state = 33, stratify
    = Y)
28 scaler = StandardScaler()
29 trainX = scaler.fit_transform(trainX)
30 testX = scaler.transform(testX)
31 echo_tracks.head()
32 clf1 = LogisticRegression(random_state = 0, max_iter = 1000, solver = 'lbfgs')
33 clf1.fit(trainX, trainY)
34 pred1 = clf1.predict(testX)
```



```

35 print("Accuracy Score using Logistic Regression = ", accuracy_score(testY, pred1))
36 clf2 = DecisionTreeClassifier()
37 clf2.fit(trainX, trainY)
38 pred2 = clf2.predict(testX)
39 print("Accuracy Score using Decision Tree Classifier = ", accuracy_score(testY, pred2))
40 clf3 = KNeighborsClassifier(n_neighbors = 10)
41 clf3.fit(trainX, trainY)
42 pred3 = clf3.predict(testX)
43 print("Accuracy Score using kNN (k = 10) = ", accuracy_score(testY, pred3))
44 clf4
45 # setting seed for replicating results= GaussianNB()
46 clf4.fit(trainX, trainY)
47 pred4 = clf4.predict(testX)
48 print("Accuracy Score using Naive Baiyes Classifier = ", accuracy_score(testY, pred4))
49 clf3 = SVC(gamma='scale', decision_function_shape='ovo')
50 clf3.fit(trainX, trainY)
51 pred3 = clf3.predict(testX)
52 print("Accuracy Score using Support Vector Machine Classifier = ", accuracy_score(testY, pred3))
53 np.random.seed(7)
54 encoder = LabelEncoder()
55 encoder.fit(trainY)
56 encoded_Y = encoder.transform(trainY)
57 encoded_test_Y = encoder.transform(testY)
58 # convert integers to dummy variables (i.e. one hot encoded)
59 dummy_y = np_utils.to_categorical(encoded_Y)
60 dummy_test_y = np_utils.to_categorical(encoded_test_Y)
61 model = Sequential()
62 model.add(Dense(256, input_shape=(8,))) #8 features
63 model.add(Activation('relu'))
64 model.add(Dropout(0.2))
65
66 model.add(Dense(256))
67 model.add(Activation('sigmoid'))
68 model.add(Dropout(0.5))
69
70 model.add(Dense(2)) #2 classes
71 model.add(Activation('softmax'))
72
73 # Compile model
74 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
75 import math
76 testScore = model.evaluate(testX, dummy_test_y, verbose = 0)
77 print('Model loss = ', testScore[0])
78 print('Model accuracy = ', testScore[1])
79
80 from python_speech_features import mfcc
81 import scipy.io.wavfile as wav
82 import numpy as np
83 from tempfile import TemporaryFile
84 import os

```

```

85 import pickle
86 import random
87 import operator
88
89 import math
90 import numpy as np
91 from collections import defaultdict
92
93 dataset = []
94 def loadDataset(filename):
95     with open("my.dat" , 'rb') as f:
96         while True:
97             try:
98                 dataset.append(pickle.load(f))
99             except EOFError:
100                 f.close()
101                 break
102
103 loadDataset("my.dat")
104
105 def distance(instance1 , instance2 , k ):
106     distance =0
107     mml = instance1[0]
108     cm1 = instance1[1]
109     mm2 = instance2[0]
110     cm2 = instance2[1]
111     distance = np.trace(np.dot(np.linalg.inv(cm2), cm1))
112     distance+=(np.dot(np.dot((mm2-mm1).transpose() , np.linalg.inv(cm2)) , mm2-mm1 ))
113     distance+= np.log(np.linalg.det(cm2)) - np.log(np.linalg.det(cm1))
114     distance -= k
115     return distance
116
117 def getNeighbors(trainingSet , instance , k):
118     distances =[]
119     for x in range (len(trainingSet)):
120         dist = distance(trainingSet[x], instance , k )+ distance(instance , trainingSet[x], k)
121         distances.append((trainingSet[x][2], dist))
122     distances.sort(key=operator.itemgetter(1))
123     neighbors = []
124     for x in range(k):
125         neighbors.append(distances[x][0])
126     return neighbors
127
128 def nearestClass(neighbors):
129     classVote ={}
130     for x in range(len(neighbors)):
131         response = neighbors[x]
132         if response in classVote:
133             classVote[response]+=1
134         else:


```

```

135         classVote[response]=1
136     sorter = sorted(classVote.items(), key = operator.itemgetter(1), reverse=True)
137     return sorter[0][0]
138
139
140 results=defaultdict(int)
141
142 i=1
143 for folder in os.listdir("./musics/wav-genres/"):
144     results[i]=folder
145     i+=1
146
147 (rate , sig)=wav.read("__path_to_new_audio_file_")
148 mfcc_feat=mfcc(sig , rate , winlen=0.020,appendEnergy=False)
149 covariance = np.cov(np.matrix.transpose(mfcc_feat))
150 mean_matrix = mfcc_feat.mean(0)
151 feature=(mean_matrix , covariance ,0)
152
153 pred=nearestClass(getNeighbors(dataset ,feature , 5))
154
155 print(results[pred])

```

9.2 Poster Presentation



Vel Tech
Rangarajam Dr. Sengottai
Vellore Institute of Technology
Dharmadurai Nagar, Velupuram, Tamil Nadu, India

AI AUDITOR: CRAFTING YOUR PERFECT PLAYLIST WITH MACHINE LEARNING

Department of Computer Science & Engineering
School of Computing
10214CS602 – MINOR PROJECT-II
WINTER SEMESTER 2023-2024

ABSTRACT

- The music industry has undergone major changes from its conventional existence and also in the form of music created in last few years. The ever-growing customer base has also increased the market for different music styles.
- A music genre is a conventional category that identifies some pieces of music as belonging to a shared tradition or set of conventions.
- Music can be divided into different genres in many different ways. The popular music genres are Pop, Hip-Hop, Rock, Jam, Blues, Country and many more.

INTRODUCTION

- Audio classification is an Application of machine learning where different sound is categorized in certain categories.
- The music genre classification can be built using different approaches in which the top 4 approaches that are mostly used are listed below.
 1. Multiclass support vector machine
 2. K-Nearest Neighbors
 3. K-means clustering algorithm
 4. Convolutional neural network.
- We will use K-Nearest Neighbors algorithm because various researches prove it is one of the best algorithms to give good performance and till time along with optimized models organizations uses this algorithm in recommendation systems as support.
- K-Nearest Neighbour (KNN) is a machine learning algorithm used for regression, and classification. It is also known as the lazy learner algorithm. It simply uses a distance-based method to find the K number of similar Neighbour to new data and the class in which the majority of Neighbours lies, it results in that class as an output. Now let us get our system ready for project implementation.

RESULTS

The best performance in terms of accuracy is observed for the CNN model that uses only the spectrogram as an input to predict the music genre with a test accuracy of 88.54%. The CRNN model and the CNN model, however robust and complex their design is, does not give a good accuracy even though the regularization metrics are modified or the epochs are increased. The reason behind this low test accuracy rate could be the limited dataset of 8000 audio tracks. Increase in the dataset might improve the accuracy of these models.

STANDARDS AND POLICIES

Operating system: windows
Coding language: python
Software: Anaconda, Heroku, Android Studio
IDE: Jupiter notebook

METHODOLOGIES

CNNs are one type of artificial neural network that was first introduced by Geoffrey Hinton et al. In 1986, researchers at Stanford University developed an algorithm called "Neural Gas" for solving image recognition problems. Inspired by the hierarchical and local properties of neurons in the visual cortices, convolutional neural networks (CNN) are used for image recognition tasks. Neurons usually contain multiple layers of connections between them; each layer contains different types of neurons (some fully connected others Convolutional).

CONCLUSIONS

music genre classification is studied using the Free Music Archive small dataset. We proposed a simple approach to solving the classification problem and we gave comparisons with multiple other complex, robust models. We also compared the models based on the kind of input it was receiving. Two kinds of inputs were given to the models: Spectrogram images for CNN models and audio features stored in a csv for Logistic Regression and ANN model. Simple ANN was determined to be the best feature-based classifier amongst Logistic Regression and ANN models with a test accuracy of 64%. CNN model was determined to be the best spectrogram-based model amongst CNN, CRNN and CNN-RNN parallel models, with an accuracy of 88.54%.

TEAM MEMBER DETAILS

VTU19315/C.Vijayamari
VTU19600/M.Manoj Babu
VTU19434/R.Gautam
9347956001
vml19315@veltech.edu.in
vml19600@veltech.edu.in
vml19434@veltech.edu.in




Figure 1: Music genre classification word cloud

Figure 2: Different genres in music

Table 1: Variants of FMA dataset.

dataset	#clips	#artists	year	audio
GTZAN	1000	~300	2002	yes
MSD	1,000,000	44745	2011	no
AudioSet	2,084,320	-	2017	no
Artist20	1413	20	2007	yes
AcousticBrainz	2,524,739	-	2017	no



Figure 3: Recommendation demo results.

Figure 9.1: Poster Presentation

REFERENCES

- [1] Choi, K., Fazekas, G., Cho, K. . Deep convolutional neural networks for music genre classification: A comparison of spectrogram and MFCC features. In Proceedings of the 41st IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 2395-2399).(2020)
- [2] Gomez, M. A., Gómez, F., Herrera, P(2018). Music genre classification with convolutional neural networks. Expert Systems with Applications, 107, 185-193.(2021)
- [3] Kim, S., Kim, Y. J., Kim, K.. Music genre classification using deep learning with ensemble techniques. IEEE Transactions on Consumer Electronics, 66(2), 183-191. (2020)
- [4] Lu, Y., Xu, S., Xu, Y., Wu, Y. Music genre classification with multi-scale convolutional neural networks. Applied Sciences, 11(5), 2342. (2021).
- [5] Park, K., Cho, S., Lee, S., Kim, J. Music genre classification using feature learning with temporal convolutional networks. Applied Sciences, 10(4), 1326. (2020).
- [6] Reddy, S., Mishra, A., Acharya, S. (2019). A deep learning approach to music genre classification using transfer learning. International Journal of Advanced Computer Science and Applications, 10(1), 39-45. (2019).
- [7] Schramm, R., Endo, P. T., de Oliveira, J. P. M. End-to-end music genre classification with recurrent neural networks. Expert Systems with Applications, 146, 113142. (2020).
- [8] Wang, J., Li, Y., Li, T. Attention-based convolutional neural networks for music genre classification. Journal of the Franklin Institute, 356(5), 2707-2722. (2022).
- [9] Xiong, X., Ma, Y., Li, B. Music genre classification using deep learning with temporal attention. IEEE Transactions on Multimedia, 23, 2397-2408. (2021).
- [10] Zhang, W., Yao, J., Zhang, H., Li, Y. Adversarial learning for music genre classification. IEEE Transactions on Multimedia, 23, 1299-1312. (2021).