

**INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT, AKURDI, PUNE**

**Music Audio Files Classification**

**using Machine Learning**

PG-DBDA March 2024

Submitted By:

Group No: 21

Roll No. Name.

243507 Amitkumar M. Bande

243512 Sushant S. Chougule

**Dr. Shantanu Pathak Mr. Rohit Pauranik**

Project Guide Centre Coordinator

## ABSTRACT

Music genre classification is a fundamental problem in the field of audio analysis, with applications ranging from recommendation system to connect organization.

This project focuses on using machine-learning techniques to classify music tracks into distinct genres, leveraging the well-known GTZAN dataset. The dataset contains a diverse collection of audio tracks spanning ten different genres, making it suitable for training and evaluation classification models. Results of the study show that the performance accuracy of these created models varied from 76 % to 80.00 %. KNN classifier being the most.

## ACKNOWLEDGEMENT

I take this occasion to thank God, almighty for blessing us with his grace and taking our endeavor toa successful culmination. I extend my sincere and heartfelt thanks to our esteemed guide, Dr. Shantanu Pathak for providing me with the right guidance and advice at the crucial juncture sand for showing me the right way. I extend my sincere thanks to our respected Centre Co-Ordinator Mr. Rohit Puranik, for allowing us to use the facilities available. I would like to thank the other faculty members also, at this occasion. Last but not the least, I would like to thank my friends and family for the support and encouragement they have given me during the course of our work.

Amitkumar M. Bande (240341225007)

Sushant S. Chougule (240341225012)

IACSD

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Description** | **Page No.** |
| 1 | Introduction |  |
| 2 | Dataset |  |
| 3 | Requirement Analysis |  |
| 4 | Software Requirement and Specification |  |
|  | Design |  |
| 3.2 | Implementation and Coding |  |
| 3.3 | Performance |  |
| 3.4 | Conclusion and Further Work |  |
| 8 |  |  |

## INTRODUCTION

**1.1 Description**

The music is an integral part of human culture, and its diversity is reflected in a wide range of genres that cater to different tastes and emotions. With the digital age enabling massive music collection, there is a growing need for efficient methods to organize and categorize music content. Music genre classification, the task of automatically assigning music tracks to predefined genre categories, plays a pivotal role in achieving this organization. In recent years, the convergence of machine learning and audio analysis has paved the way for accurate and automated genre classification systems.

The primary objective of this project is to explore the realm of music genre classification by leveraging the power of machine learning techniques, focusing on the renowned GTZAN dataset. This dataset has gained widespread recognition within the research community for its comprehensive collection of audio tracks across ten distinct genres. Ranging from blues to hip-hop, the GTZAN dataset provides an ideal platform for training and evaluating machine learning model for genre classification.

Acquired the dataset form Tenser Flow, we tried analyzing this dataset using Convolutional Neural Network (CNN) vs Machine learning algorithm and compared their prediction outcome and accuracy.

### 1.2 Need of the work

The purpose of this document is to provide a detailed specification of the features and functionalities of the " e-Nirvachan " It outlines the requirements, system architecture, and user interactions.

### 1.3 Problem Statement

The system is designed to manage information related to voters, political parties, candidates. It facilitates tasks such as voters registration, user management, casting vote to a candidates of voters choice and political party registration.

### 1.4 Objective of Project

The e-Nirvachan project is designed with a clear set of objectives to enhance the election process. The primary goal is to provide dealerships with a streamlined platform to efficiently cast vote at the comfort and security of home environment making the election process simpler and more sophisticated for citizens.

Key objectives include creating a user-friendly experience for political party and voters, offering real-time voting information availability, ensuring secure user authentication for data protection, and enabling seamless communication between candidate and voters.

The system focuses on maintaining accurate information, enhancing data accuracy and integrity, and adhering to legal requirements related to data privacy and security.

With modern technology and a future-ready approach, the project aims to create a reliable, transparent, and efficient platform that benefits all stakeholders in the election process.

### 2. REQUIREMENT ANALYSIS

2.1 Information gathering

2.2 Project Survey

2.3 System Requirement

2.4 SDLC model diteails

3. Software Requirement and Specification

3.1. Introduction

1. Create a system that uses machine learning and deep learning to classify audio genres into predefined categories.

2. Deploy the classification model into production environment while considering scalability, reliability and maintenance requirements.

3.2 Functional Requirement

1. Input Requirements

* **Audio File Support:** The system must accept various audio file formats (e.g., MP3, WAV, FLAC) as input.
* **Duration Handling:** The system should handle full-length tracks as well as short audio clips.
* **Multiple Audio Channels:** The system must process both mono and stereo audio inputs.

2. Preprocessing Requirement

**Feature Extraction:** The system should extract relevant features from audio files, such as Mel-Frequency Cepstral Coefficients (MFCCs), Chroma features, and Spectrograms.

**Normalization:** The extracted features should be normalized to ensure consistent input to the model.

**Data Augmentation:** The system should support data augmentation techniques (e.g., pitch shifting, time stretching) to increase the robustness of the model.

3. Modeling Requirement

**Model Architecture:** The system should implement appropriate ML/DL architectures for audio classification.

**Training Capability:** The system must be capable of training on labeled datasets with multiple genres.

**Hyperparameter Tuning:** The system must allow for hyperparameter tuning to optimize model performance.

4. Security and Privacy Requirements

 Data **Privacy:** The system must ensure the privacy and security of any user-uploaded audio files, adhering to relevant data protection regulations (e.g., GDPR).

 Access **Control:** The system should have access control mechanisms to restrict the use of sensitive data and features.

3.3 Non Functional Requirements

1. Performance Requirements

 Latency**:** The system should have low latency, with genre classification results delivered within a specific time frame.

 Throughput**:** The system should be capable of processing a high volume of audio files concurrently, maintaining performance under load.

 Scalability**:** The system must scale horizontally or vertically to accommodate growing data volumes and user demand.

2. Reliability Requirements

 Uptime**:** The system must ensure high availability, with minimal downtime.

 Fault **Tolerance:** The system should be resilient to hardware or software failures, ensuring continuous operation without data loss.

 Backup **and Recovery:** The system should have mechanisms for regular backups and quick recovery in case of failures or data corruption.

3. Scalability Requirements

 Elastic **Scaling:** The system should dynamically adjust resources based on workload, scaling up during peak times and scaling down during off-peak times.

 Cloud**-Native Design:** The system should be designed for cloud deployment, leveraging cloud services for scaling and load balancing.

### 3.4 Requirements:

**1. Software requirements:**

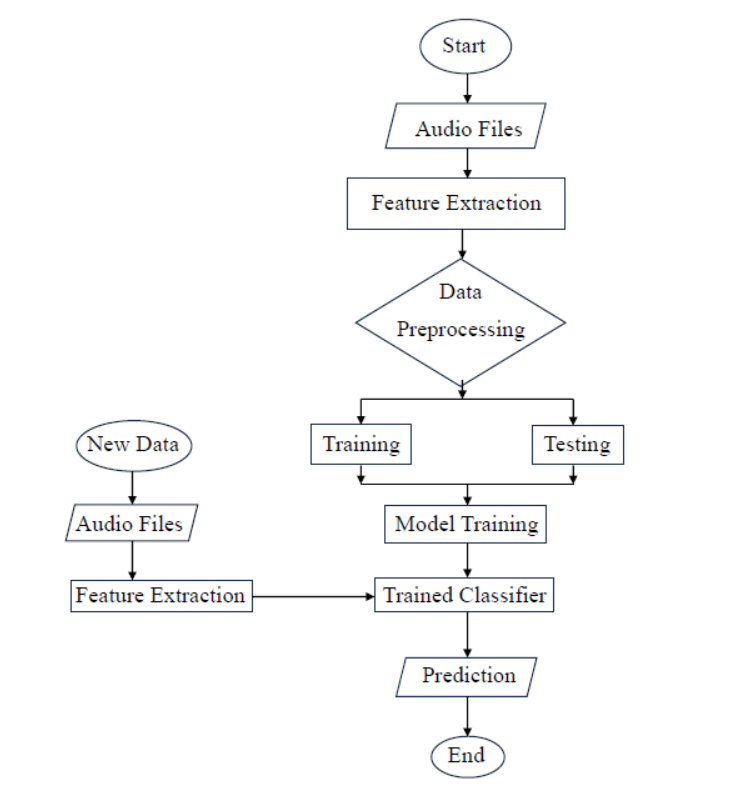
* + - 1. Operating System: Windows 98, Windows XP, Windows7
      2. Language: Python Runtime Environment
      3. Browser: Any of Mozilla, Opera, Chrome etc.
      4. Software Development Kit: Python
      5. Scripting Language Enable: Python Script

**2. Hardware requirements:**

1. 1. System Processor: Core i3/i5
2. 2. Hard Disk: 500 GB

4. Design

4.1 System Architecture



Description:

1. **Audio Files (Left Branch)**: The audio files are inputted into the system, beginning the preprocessing phase.

2. **Feature Extraction** : The system extracts relevant features from the audio files. These features could include Mel-frequency cepstral coefficients (MFCCs), Chroma features, or Spectrograms.

3. **Data Preprocessing** : The extracted features undergo preprocessing, which may include normalization, noise reduction, or other techniques to prepare the data for training or prediction.

4. **Model Training** : The model is trained using the preprocessed and extracted features. The training process results in a trained classifier, which can predict music genres based on new input.

5. **Trained Classifier** : After training, the resulting model (trained classifier) is ready to be used for predicting genres in unseen audio files.

6. **New Data**: The system receives new audio data, which will undergo classification.

7. **Testing** : If the system is in the testing phase, the trained classifier is tested with new audio data to evaluate its performance and accuracy.

8. **Prediction**: The preprocessed features are passed through the trained classifier to predict the music genre of the audio files.

4.2 MODULES:

1. Essential Modules

* **Librosa:** For audio loading, manipulation, and feature extraction.
* **NumPy:** For numerical operations and array manipulation.
* **Pandas:** For data manipulation and analysis.
* **Scikit-learn:** For machine learning algorithms and evaluation metrics.
* **TensorFlow/Keras:** For deep learning model building and training.

2. Feature Extraction

 Mel**-Frequency Cepstral Coefficients (MFCCs):** Commonly used features representing the short-term power spectrum of a sound.

 Chroma **Features:** Represent the 12 pitch classes.

 Spectral **Centroid:** Measures the center of mass of the spectrum.

 Spectral **Rolloff:** Indicates the frequency below which a specified percentage of the signal's energy is located.

 Zero**-Crossing Rate:** Measures the rate at which a signal changes from positive to negative or vice versa.

3. Model

 **Model Selection:** Choose appropriate machine learning or deep learning algorithm.

 **Model Training:** Train the model on the extracted features.

 **Model Evaluation:** Evaluate the model's performance using metrics like accuracy, precision, recall, and F1-score.

 **Model Deployment:** Deploy the trained model for real-time predictions.

5. Implementation and Coding

**5.1 Brief description of technology used, language used, tools used for the development of the project**.

1) Description of technology, tools and languages used –

Visual Studio Code -

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

PYTHON –

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems.

Jupyter Notebook –

Allows you to create and share documents containing live code, equations, visualizations, and narrative text. Supports code execution, data exploration, and visualization in a single document. Can support various programming languages through different kernels. Popular in data science, machine learning, and scientific computing communities.

Streamlit API -

**Streamlit** doesn't have a traditional API in the sense of a RESTful or GraphQL interface. It's primarily a Python library designed for building interactive web applications, particularly for data scientists and machine learning engineers.

Git-

**Git** is a powerful version control system (VCS) used to track changes in computer files and directories over time. It's essential for managing and coordinating work on software projects, but it can also be used to track changes in other types of files.

2) Library used in project-

**Librosa:** For audio loading, manipulation, and feature extraction.

**NumPy:** For numerical operations and array manipulation.

**Pandas:** For data manipulation and analysis.

**Scikit-learn:** For machine learning algorithms and evaluation metrics.

**TensorFlow/Keras:** For deep learning model building and training.

3) Machine Learning Technology:

Music genre classification algorithms, such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Support Vector Machines (SVMs), are commonly used to analyze and classify audio data into specific genres. These algorithms learn patterns and distinguishing features from large datasets of labeled music tracks, enabling them to accurately categorize new, unseen tracks based on their audio characteristics.

Machine learning models used in music genre classification are trained on historical audio data and refined over time to improve their accuracy and generalization. By analyzing attributes such as tempo, rhythm, pitch, and timbre, these models identify patterns that are indicative of specific music genres.

These models can be deployed to classify music in real-time or near real-time, allowing for instant categorization of tracks. Real-time processing is particularly useful in applications such as streaming services, where immediate genre identification can enhance user experience by enabling personalized recommendations or automatic playlist generation

**5.2 Important Algorithms, Neural Networks that have been implemented so far should be mentioned in this chapter.**

**K-Nearest Neighbors (KNN)-**

K-Nearest Neighbors (KNN) is a simple, instance-based learning algorithm used for classification and regression tasks. It operates on the principle that similar data points are likely to have similar labels. When a new data point needs to be classified, KNN identifies the k nearest data points in the feature space and assigns the most common label among those neighbors to the new point. Since KNN doesn’t involve an explicit training phase, it is known as a lazy learner, storing the entire training dataset for use during the prediction phase.

Random Forest-

Random Forest is an ensemble learning algorithm that combines the predictions of multiple decision trees to improve accuracy and reduce the risk of overfitting. It creates a "forest" of decision trees by training each tree on a random subset of the training data and using a random subset of features to split nodes. The final prediction is made by aggregating the predictions from all the trees, typically through majority voting for classification tasks or averaging for regression tasks.

Support Vector Machine (SVM) –

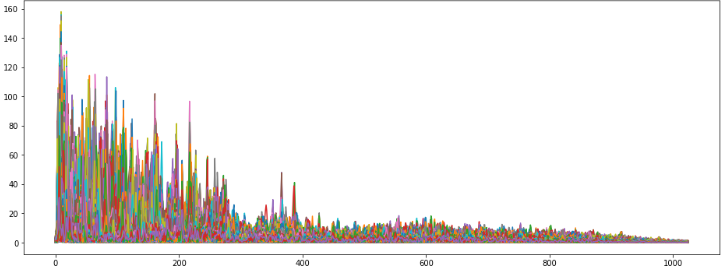
Support Vector Machine is a powerful supervised learning algorithm used for classification and regression tasks. SVM works by finding the optimal hyperplane that best separates the data into different classes, maximizing the margin between the nearest data points of each class. If the data is not linearly separable, SVM can apply a kernel trick to transform the data into a higher-dimensional space where a separating hyperplane can be found. SVM is particularly effective in high-dimensional spaces and with a clear margin of separation between classes.

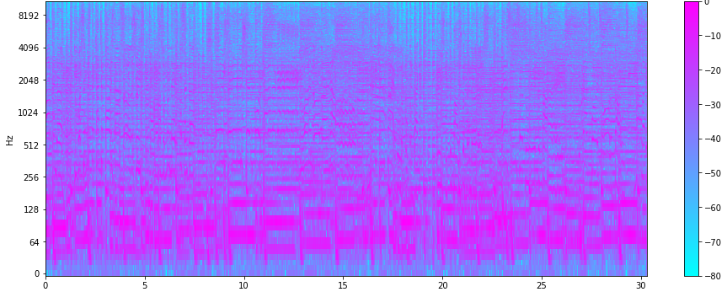
Convolution Neural Network (CNN) –

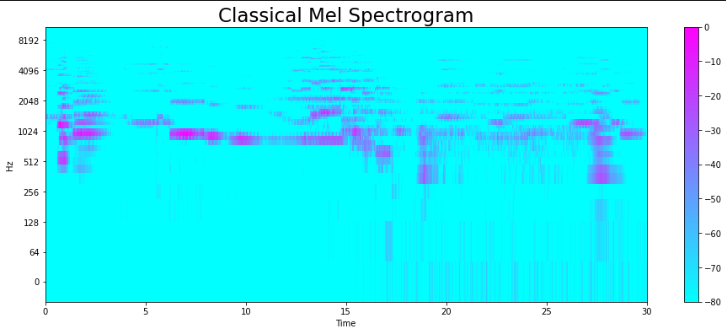
A Convolutional Neural Network is a type of deep learning model specifically designed for processing structured grid data, such as images. CNNs are particularly well-suited for tasks like image recognition, object detection, and image segmentation.

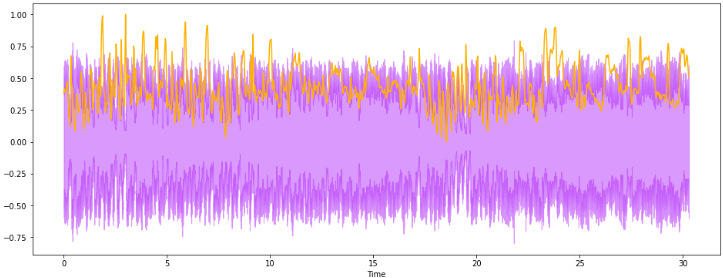
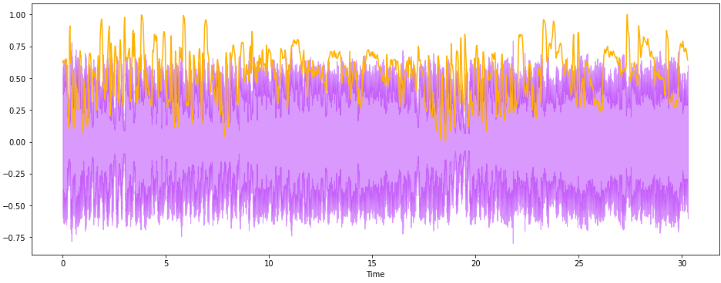
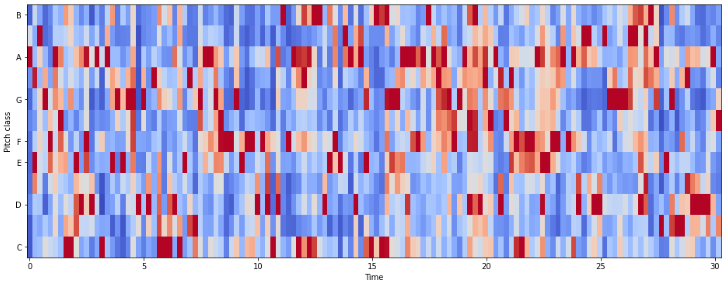
5.3 Few important screen shots should be included with appropriate labels to the screen shots and description.

1) Feature Extraction-

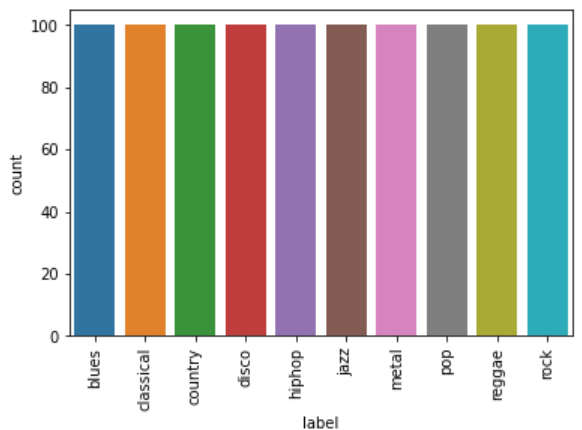


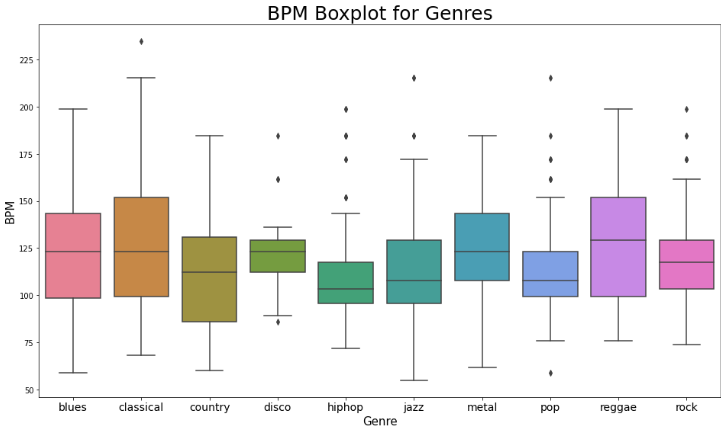




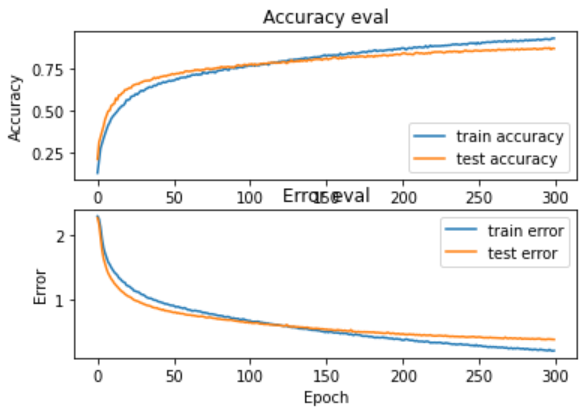
2) Exploratory Data Analysis (EDA)-





**3) Accuracy-**

****

****

**6. Performance**

**8. CONCLUSION AND FUTURE WORK**

**Conclusion-**

The music genre classification project successfully demonstrated the capability of machine learning models, particularly using audio features like spectrograms, to accurately classify audio files into distinct genres. Through feature extraction techniques and model training, the system effectively learned to distinguish between various musical styles, achieving satisfactory accuracy levels. The use of Convolutional Neural Networks (CNNs) proved particularly effective due to their ability to capture intricate patterns in the audio data. This project highlights the potential of combining deep learning with audio signal processing for automated music categorization.

Future work-

**Model Enhancement:** Future work could focus on refining the model architecture, experimenting with more advanced neural network designs such as recurrent neural networks (RNNs) or transformer-based models, which might better capture temporal dependencies in audio data.

**Dataset Expansion:** Expanding the dataset to include more genres and a wider variety of songs within each genre could improve the model's robustness and generalization.

**Real-Time Classification:** Implementing the model for real-time music genre classification, perhaps as a streaming service, could be an interesting extension. This would require optimization techniques to ensure low-latency predictions.

**Cross-Domain Analysis:** Integrating additional data sources such as lyrics, artist information, or cultural context could further enhance classification accuracy by providing more context about the music.