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

Mr. Abhijeet Nagargoje
Project Guide

Mr. Rohit Puranik
Centre Coordinator

ABSTRACT

The classification of audio files into different genres is a complex task that involves extracting meaningful features from audio data and applying machine learning techniques. In this study, we use the Python library Librosa to preprocess and analyze audio files, extracting features such as waveforms, Fourier transforms, spectrograms, and Mel-Frequency Cepstral Coefficients (MFCCs). These features form the basis for our machine learning models, which include K-Nearest Neighbors (KNN), Random Forest, and Support Vector Machine (SVM), with KNN achieving a notable accuracy of 94%.

Our findings highlight the importance of feature selection and model tuning in genre classification tasks. The KNN model stands out for its effectiveness in capturing audio feature nuances, leading to high accuracy. Additionally, we explore a deep learning approach using a multi-layer perceptron (MLP), which shows competitive results and suggests potential for handling larger and more complex datasets.

The successful implementation of these models demonstrates the potential for automated audio genre classification, with applications in music streaming services, content recommendation systems, and media organization. Our study lays the groundwork for future research, particularly in advancing deep learning techniques and feature engineering to further enhance the accuracy and efficiency of genre classification systems.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to everyone who has contributed to the successful completion of this project. First and foremost, I would like to thank my guide, Mr. Abhijeet Nagargoje , for their invaluable guidance, support, and encouragement throughout the duration of this project. Their expertise and insights have been instrumental in shaping the direction and outcome of this work. I extend my sincere thanks to our respected Centre Co-Ordinator, Mr. Rohit Puranik, for allowing us to use the facilities available and for his unwavering support and dedication to ensuring our success. I would also like to express my gratitude to our faculty members, Dr. Shantanu Pathak, Mrs. Priti Take, and Mrs. Priyanka Bhor for their continuous support and encouragement. I am also grateful to my colleagues and peers, who provided continuous support and constructive feedback, fostering an environment of collaboration and learning. Their contributions have enriched the quality of this project. Lastly, I would like to acknowledge Institute for Advanced Computing & Software Development for providing the necessary resources and a conducive environment that enabled me to carry out this project.

Amitkumar M. Bande (240341225007)
Sushant S. Chougule (240341225012)

Table of Contents

1. INTRODUCTION

The music is an integral part of human culture, and its diversity is reflected in awide range of genres that cater to different tastes and emotions. With the digitalage 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 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 machinelearning model for genre classification.

The classification of audio files into different genres is a fundamental task in the field of music information retrieval, with applications ranging from music recommendation systems to automated media organization. This project focuses on using machine learning techniques to accurately classify audio files into their respective genres. We leverage the Python library Librosa to preprocess audio data and extract relevant features, such as waveforms, Fourier transforms, spectrograms, and Mel-Frequency Cepstral Coefficients (MFCCs). These features serve as input to various machine learning models, including K-Nearest Neighbors (KNN), Random Forest, and Support Vector Machine (SVM), which are trained and evaluated to determine their effectiveness in genre classification.

2. REQUIREMENTS

2.1 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.

2.2 Functional Requirement

2.2.1 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.

2.2.2 Preprocessing Requirement

- **Feature Extraction:** The system should extract relevant features from audio files, such as Mel-Frequency Cepstral Coefficients (MFCCs), Chroma features, and Spectrograms.
- **Feature Extraction:** The system should extract relevant features from audio files, such as Mel-Frequency Cepstral Coefficients (MFCCs), Chroma features, and Spectrograms.
- **Data Augmentation:** The system should support data augmentation techniques (e.g., pitch shifting, time stretching) to increase the robustness of the model.

2.2.3 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.

2.2.4 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.

2.3 Non-Functional Requirements

2.3.1 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.

2.3.2 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.

2.3.3 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.

2.4 Other Requirements

2.4.1 Hardware Specifications

- Machine: Desktop/Laptop
- Operating system: Windows 10 (or above) or Linux 18 LTS (or above) Processors: Intel core i5 or AMD 5 series (minimum)
- Memory: 8 GB RAM or above Hard Disk (SSD): 250 GB or more
- Video Card (optional): Intel Integrated Graphics (suggested – 4 GB graphics card - NVIDIA)
- Network: Ethernet / Wi-Fi with 25 Mbps Speed Connection (UL/DL)

2.4.2 Software Specifications

- Language: Python 3.7 or above (stable build)Platform: Anaconda Latest stable build
- Notebooks: Jupyter and Google Colab
- Libraries: librosa, Pandas, Numpy, Matplotlib, Seaborn, Keras, Tensorflow, Scikit Learn, Mahotas.

3. DATASET

The dataset is taken from Tenser Flow website. This dataset contains audioclips, csv files.

Dataset Overview

GTZAN dataset has gained widespread recognition within the researchcommunity 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.

Dataset Composition:

The GTZAN dataset consists of a collection of a collection of 1000 audio clips,each 30 seconds in duration. These clips are evenly distributed across 10 different music genres, with 100 clips per genre.

The genres included in the datasets are:

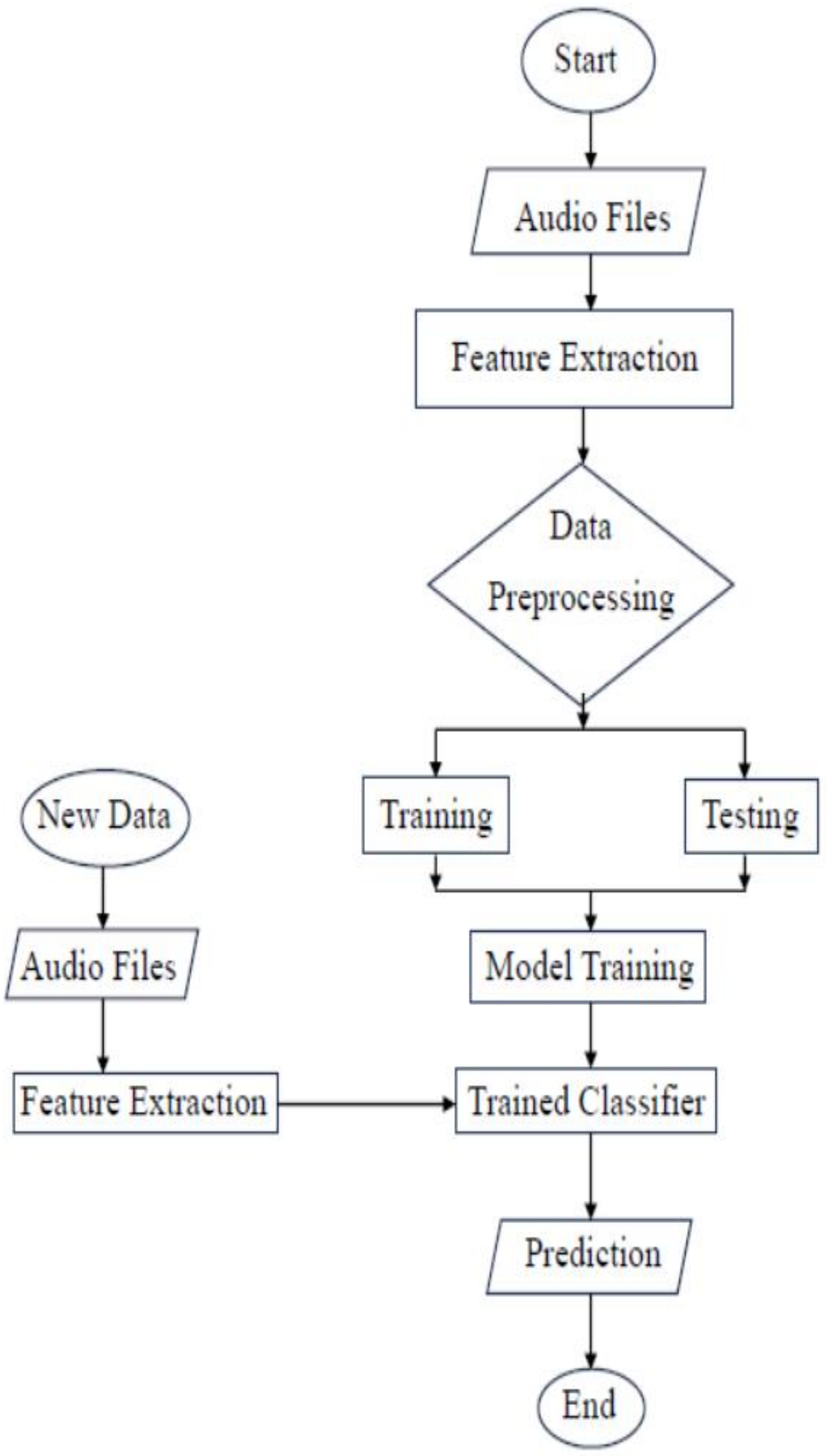
1. Blues
2. Classical
3. Disco
4. Country
5. Hip-hop
6. Jazz
7. Metal
8. Pop
9. Reggae
10. Rock

Audio format:

All audio clips in the dataset are provided in WAV format.

4. DESIGN

- **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.

5. MODULES

5.1 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.

5.2 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.

5.3. 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.

6. IMPLEMENTATION AND VISUALIZATION

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

6.1.1 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 madeby 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 does not have a traditional API in the sense of a RESTful or GraphQL interface. It is 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 is essential for managing and coordinating work on software projects, but it can also be used to track changes in other types of files.

6.1.2 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.

6.1.3 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

6.2 Brief descriptions of Algorithms, Neural Networks that have been implemented

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.

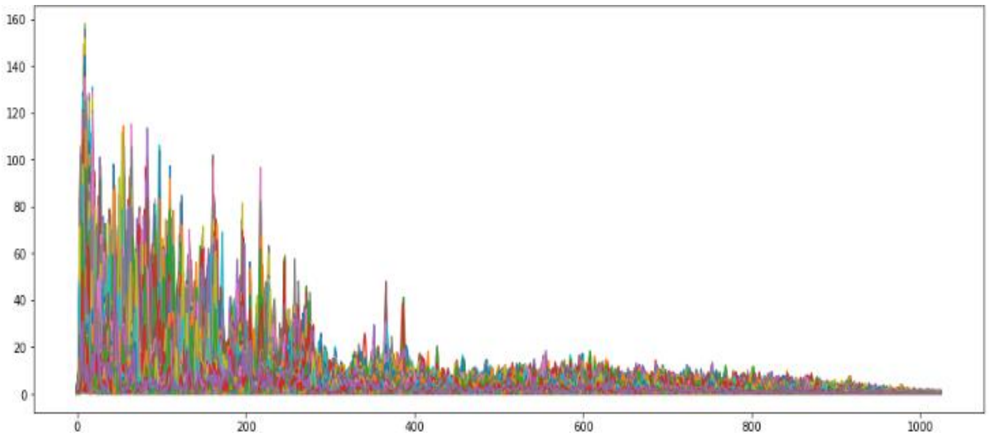
Convolution Neural Network (CNN)

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.

6.3 Visual Implementation

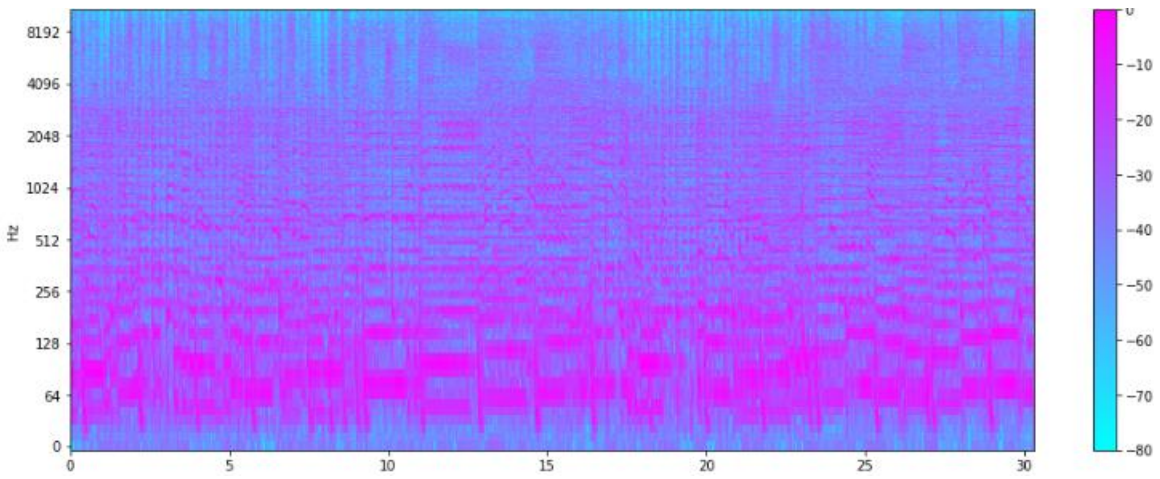
6.3.1 Feature Extraction

6.3.1.1 Fourier Transform



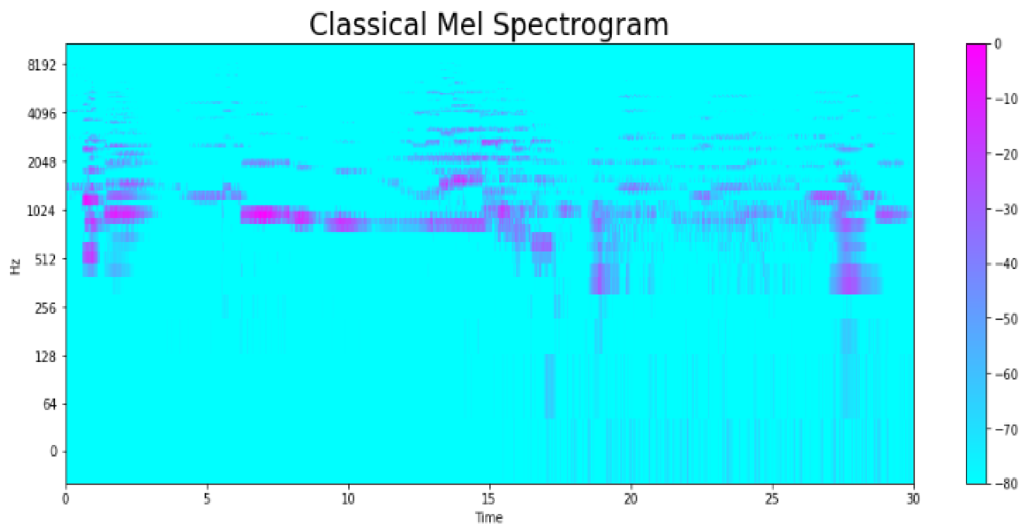
- Function that gets a signal in the time domain as input, and outputs its decomposition into frequencies
- Transform both the y-axis (frequency) to log scale, and the x-axis (amplitude) to Decibels, which is approx. the log scale of amplitudes.

6.3.1.2 Spectrogram



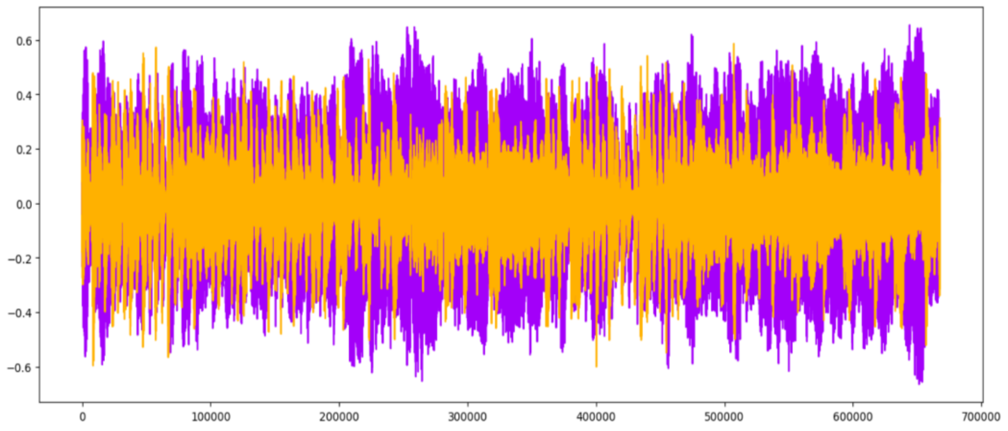
A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time. When applied to an audio signal, spectrograms are sometimes called sonographs, voiceprints, or voicegrams.
Here we convert the frequency axis to a logarithmic one.

6.3.1.3 Mel-Spectrogram



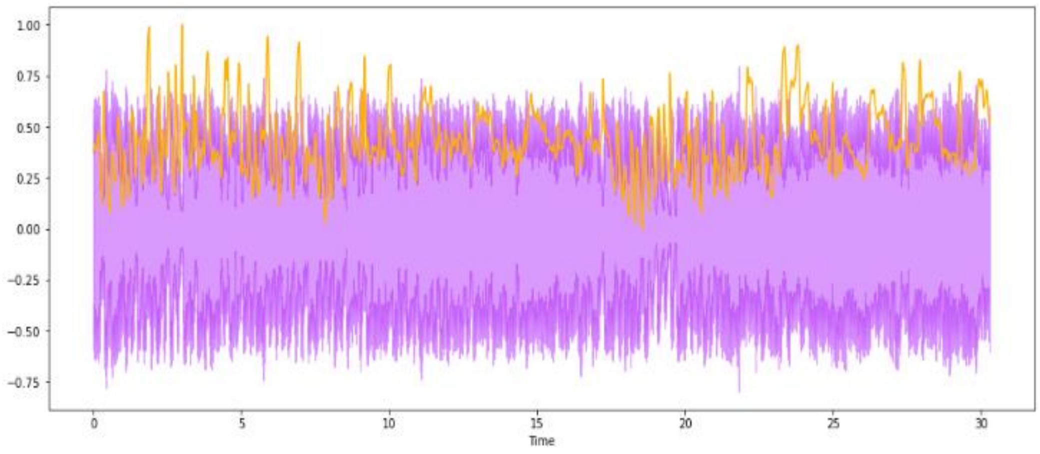
A Mel spectrogram is a type of spectrogram that represents the spectral content of a sound or signal on a scale that is based on the perceived pitch of the different frequencies. The Mel scale is a logarithmic scale that maps frequency to pitch in a way that is more closely related to how the human auditory system perceives pitch.

6.3.1.4 Harmonics and Perceptual



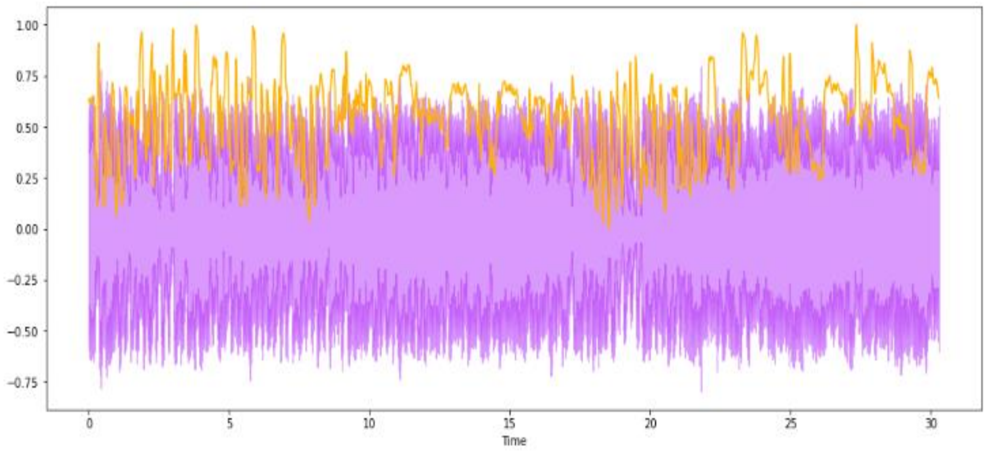
- Harmonics are characteristichs that human years can't distinguish (represents the sound color) .
- Perceptual understanding shock wave represents the sound rhythm and emotion.

6.3.1.5 Spectral Centroid



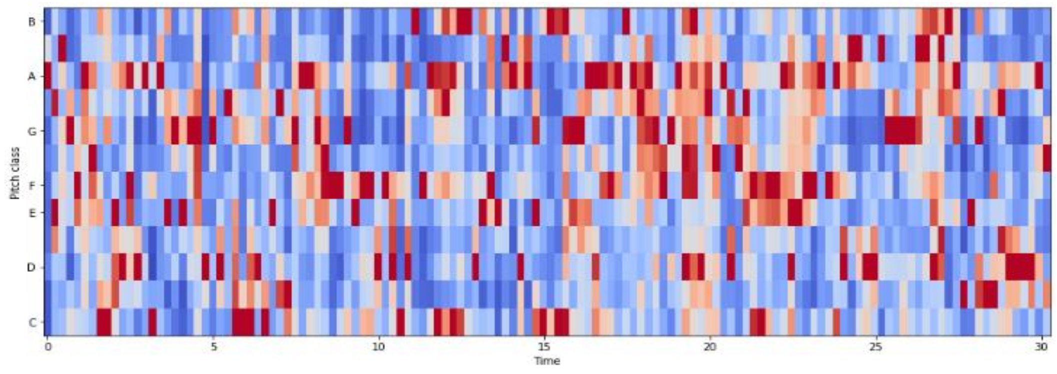
- indicates where the center of mass for a sound is located and is calculated as the weighted mean of the frequencies present in the sound**.**

6.3.1.6 Spectral Rolloff



It is a measure of the shape of the signal. It represents the frequency below which a specified percentage of the total spectral energy, e.g. 85%, lies.

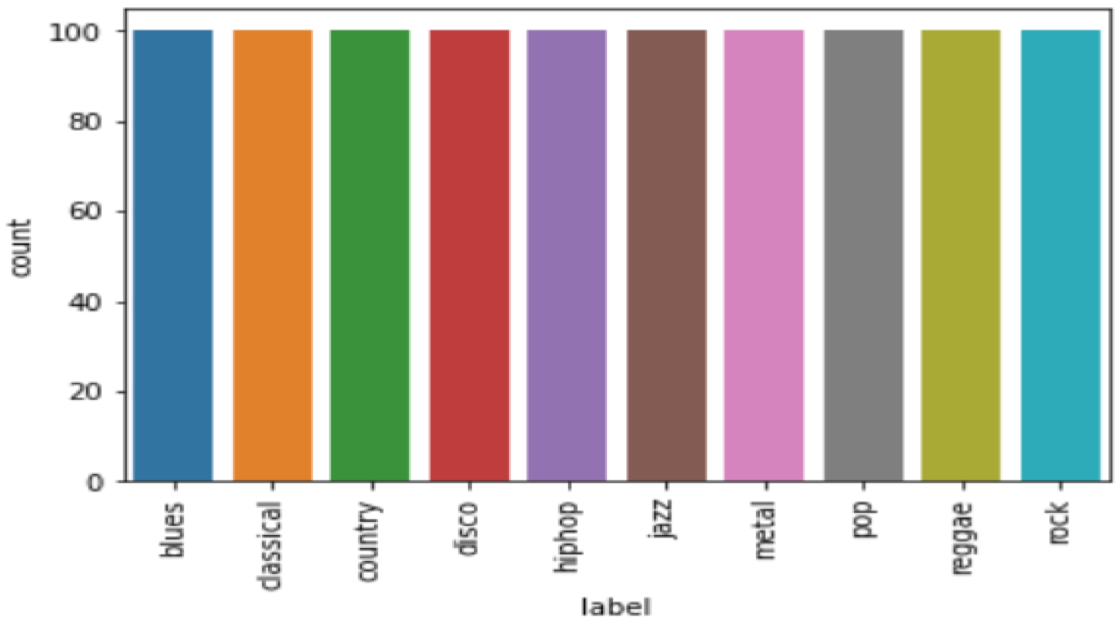
6.3.1.7 Chroma Frequencies



Chroma features are an interesting and powerful representation for music audio in which the entire spectrum is projected onto 12 bins representing the 12 distinct semitones (or chroma) of the musical octave.

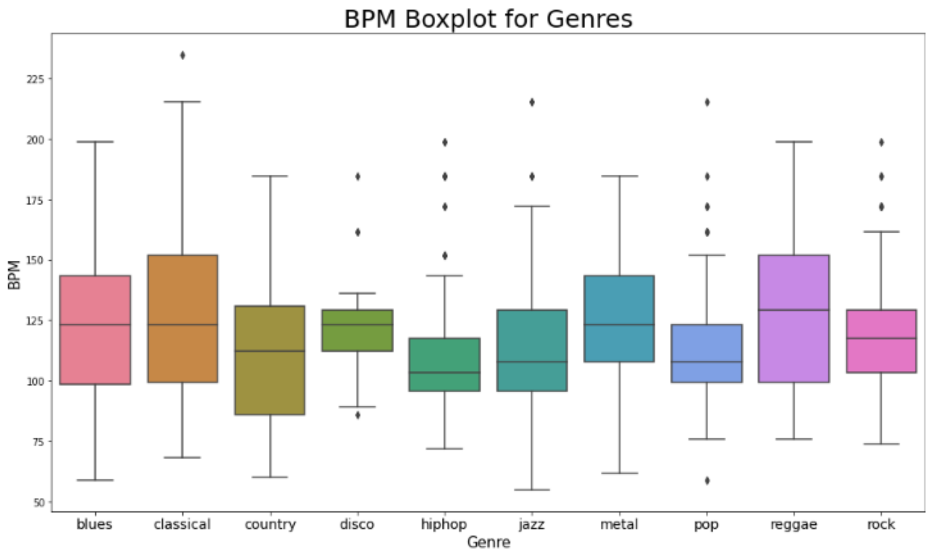
6.3.2 Exploratory Data Analysis (EDA)

6.3.2.1 Bar Chart



- The image presents a bar chart showing the count of different music genres in a dataset.
- The x-axis lists the genres: Blues, Classical, Country, Disco, Hiphop, Jazz, Metal, Pop, Reggae, and Rock.
- The y-axis represents the count, ranging from 0 to 100.
- Each bar corresponds to a genre and its height indicates the number of occurrences of that genre in the dataset.

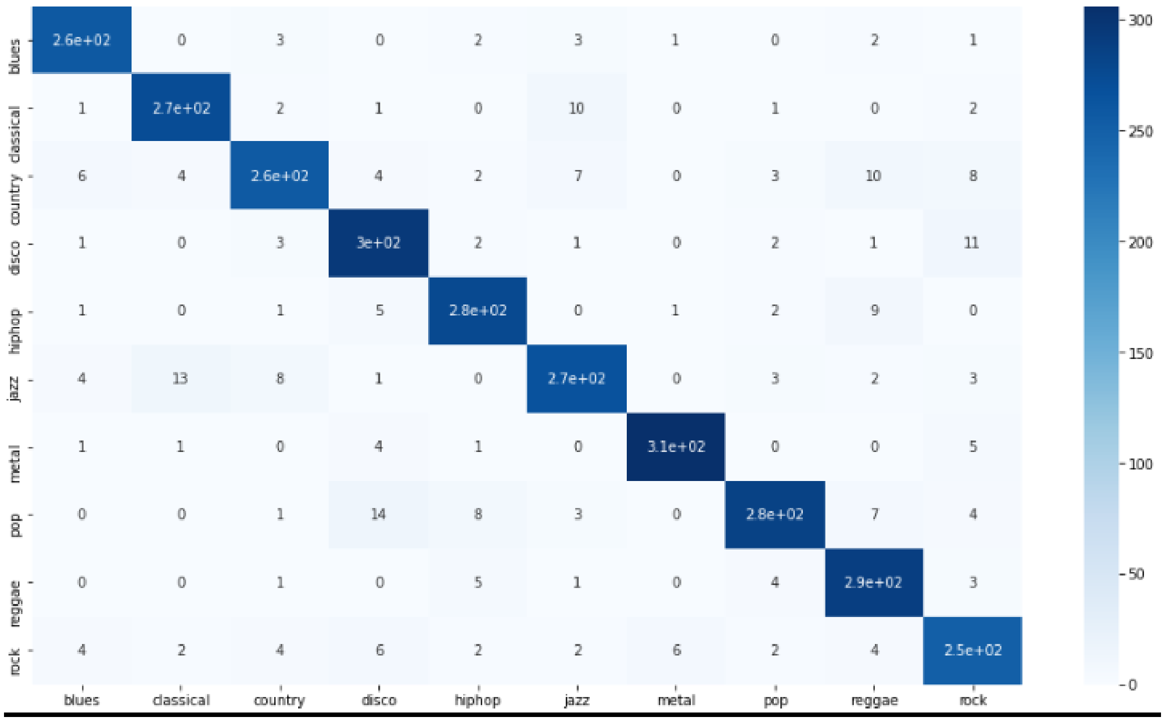
6.3.2.2 Box Plot



- The image presents a boxplot visualization, comparing the distribution of Beats Per Minute (BPM) values across different music genres.
- The x-axis lists the genres: Blues, Classical, Country, Disco, Hiphop, Jazz, Metal, Pop, Reggae, and Rock.
- The y-axis represents the BPM values, ranging from 50 to 225

6.3.3 Accuracy

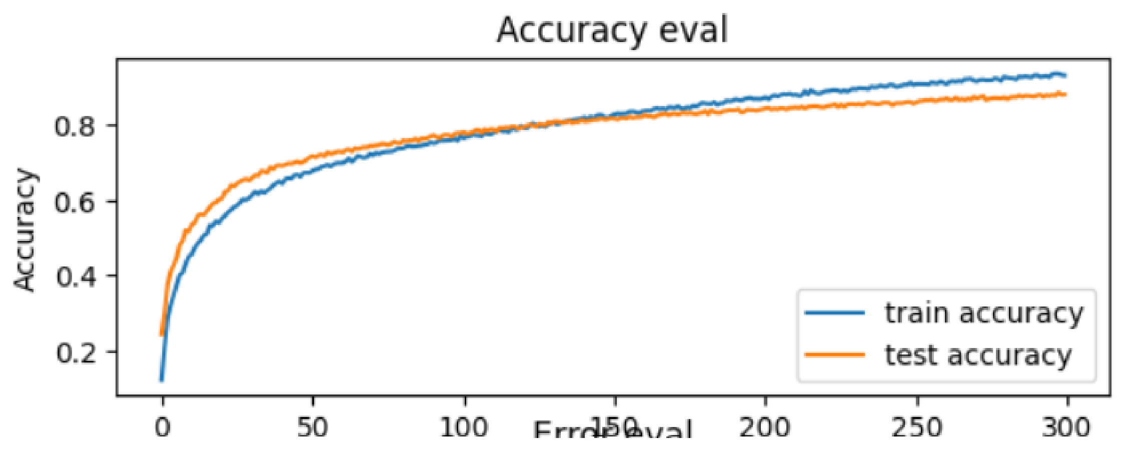
6.3.3.1 Confusion Matrix



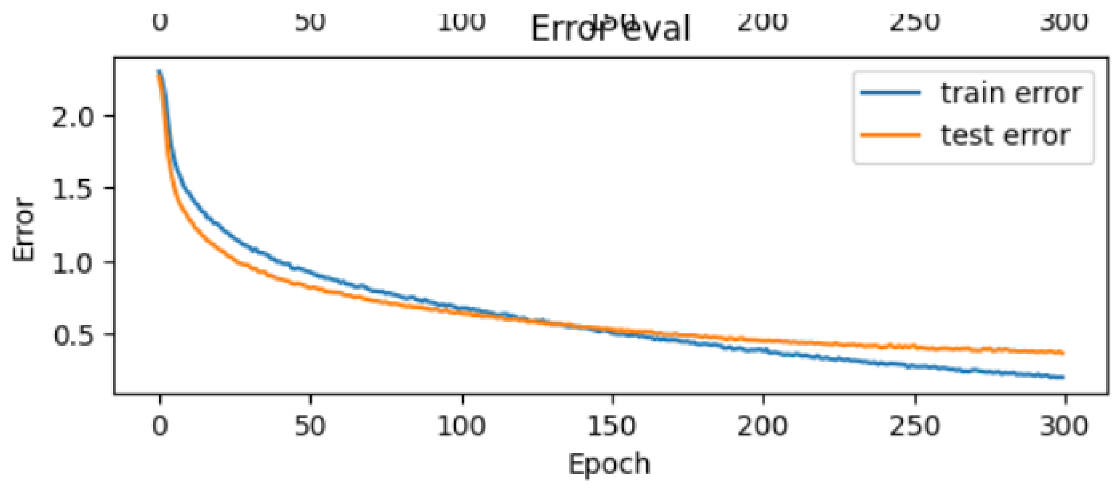
	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	2.6e+02	0	3	0	2	3	1	0	2	1
classical	1	2.7e+02	2	1	0	10	0	1	0	2
country	6	4	2.6e+02	4	2	7	0	3	10	8
disco	1	0	3	3e+02	2	1	0	2	1	11
hiphop	1	0	1	5	2.8e+02	0	1	2	9	0
jazz	4	13	8	1	0	2.7e+02	0	3	2	3
metal	1	1	0	4	1	0	3.1e+02	0	0	5
pop	0	0	1	14	8	3	0	2.8e+02	7	4
reggae	0	0	1	0	5	1	0	4	2.9e+02	3
rock	4	2	4	6	2	2	6	2	4	2.5e+02

- The diagonal elements (top-left to bottom-right) represent correct predictions for each class.
- Off-diagonal elements indicate incorrect predictions, with higher values suggesting more misclassifications between those classes.
- The color intensity likely represents the frequency of occurrences, with darker colors indicating higher counts.

6.3.3.2 Accuracy and Loss



- Accuracy shows the evolution of both training and test accuracy during the training process.
- Training accuracy consistently increases, reaching a plateau around 0.8.
- Test accuracy also increases but at a slower rate, eventually stabilizing below the training accuracy.



- Displays the training and test error throughout the training process.
- Both training and test errors decrease as the model learns.
- The training error drops rapidly initially, then slows down.
- The test error decreases more gradually and levels off at a higher value than the training error.

7. PERFORMANCE

7.1 Training Performance

Train accuracy refers to the percentage of correctly classified instances in the training dataset by a machine learning model. It measures how well the model has learned the patterns and relationships in the training data. High train accuracy indicates that the model is performing well on the data it was trained on, effectively capturing the underlying patterns. However, if the train accuracy is significantly higher than the test or validation accuracy, it might indicate that the model is overfitting, meaning it has learned the training data too well, including noise or irrelevant details, and may not generalize well to new, unseen data.

Algorithms	Accuracy
KNN	80.41%
Random Forest	87.12%
SVM	74.77%
MLP	84.42%

7.2 Hyperparameter Tuning Performance

- Cross-validation performance refers to how well a machine learning model performs when evaluated using cross-validation, The primary benefit of cross-validation is that it provides a more reliable estimate of a model's performance on unseen data, as the model is tested multiple times on different subsets of the data.
- Adam is used for MLP because it tends to perform well on a variety of problems with minimal tuning. It’s efficient in terms of both computation and memory, and generally requires less fine-tuning of hyperparameters like the learning rate compared to other optimization methods.

Algorithms	Accuracy
KNN	94.23%
Random Forest	92.45%
SVM	90.12%
MLP	87.25%

7. CONCLUSION

In conclusion, our project effectively demonstrates the potential of using a combination of Convolutional Neural Networks (CNN) and traditional machine learning algorithms for the task of music genre classification. By leveraging Librosa for feature extraction, we successfully captured essential audio features that represent the characteristics of various genres. The use of CNNs allowed us to automatically learn hierarchical patterns from these features, enhancing the model's ability to distinguish between different genres.

Subsequently, by applying machine learning classifiers such as Random Forest, Support Vector Machine (SVM), k-Nearest Neighbors (KNN), and Multi-Layer Perceptron (MLP), we explored different approaches to further refine the classification process. The diversity of these classifiers provided us with a comprehensive understanding of how different models perform on the extracted features. Our results highlight the effectiveness of this combined approach, offering valuable insights for future work in the field of music genre classification.

Overall, this project underscores the importance of integrating deep learning techniques like CNNs with classical machine learning methods to achieve robust and accurate audio classification. The success of our models on the GTZAN dataset confirms the viability of this approach, paving the way for further advancements in automated music genre identification and similar audio-based applications.

9. 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.

10. APPLICATIONS

a. Music Recommendation Systems

- **Personalized Music Suggestions:** Platforms like Spotify use genre classification to recommend songs that match users' tastes, enhancing engagement.
- **Playlist Creation:** Systems automatically generate playlists by grouping similar genres, helping users find music that suits their mood or activities.

b. Content Creation

- **Efficient Cataloging:** Genre classification helps organize large music libraries, making it easier for producers and users to manage and access content.
- **Search and Retrieval:** Users can quickly find specific genres or types of music in databases, which is useful for DJs and media managers.

c. Music Streaming Services

- **Enhanced User Experience:** Genre classification allows streaming services to offer genre-specific stations and playlists, keeping users engaged.
- **Targeted Advertising:** By knowing users' preferred genres, platforms can deliver more relevant ads.
- **Content Curation:** Platforms ensure a diverse representation of genres, supporting music discovery and promoting new artists.

11. REFFERENCES

- Convolutional Neural Network

  https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

- Machine Learning Algorithms

  https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifie r.html

  https://builtin.com/data-science/random-forest-algorithm

  https://scikit-learn.org/stable/modules/svm.html

  https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/