TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
ADVANCED COLLEGE OF ENGINEERING AND MANAGEMENT
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

KALANKI, KATHMANDU

A Minor Project Final Defense Report
On
"MUSIC GENRE CLASSIFICATION USING KNN"

[CT654]

Submitted by:

PRAFUL SAPKOTA (30156)
PRASANNA KHADKA (30161)
PRATIK RAJ LAUDARI (30162)

Under Supervision on:

Er. LAXMI PRASAD BHATT

ER. SAMEEP DHAKAL

A Minor Project Final report submitted to the department of Electronics and Computer Engineering in the partial fulfillment of the requirements for degree of Bachelor of Engineering in Computer Engineering

Kathmandu, Nepal

April 30, 2023

ADVANCED COLLEGE OF ENGINEERING AND

MANAGEMENT

DEPARTMENT OF COMPUTER AND ELECTRONICS ENGINEERING

APPROVAL LETTER

The undersigned certify that they have read and recommended to the Institute of

Engineering for acceptance, a project report entitled “MUSIC GENRE CLASSIFICATION USING KNN”

Submitted by:

PRAFUL SAPKOTA (30156)
PRASANNA KHADKA (30161)
PRATIK RAJ LAUDARI (30162)

In partial fulfillment for the degree of Bachelor in Computer Engineering.

.....	.....
.....	.....
Project Supervisor	Project Supervisor

.....
External Examiner

.....

Er. Laxmi Prasad Bhatt

Academic Project coordinator

Department of Computer and Electronics Engineering

Date:

ACKNOWLEDGEMENT

We take this opportunity to express our deepest and sincere gratitude to our Project Coordinator **Er. Laxmi Prasad Bhatt** and **Er. Sameep Dhakal,** Department Of Electronics and Computer Engineering for their insightful advice, motivating suggestions, invaluable guidance, help and support in successful completion of this project and also for his/her constant encouragement and advice throughout our Bachelor's programme.

We express our deep gratitude to **Er. Ajaya Shrestha**, Head of Department of Electronics and Computer Engineering, **Er. Bikash Acharya**, Deputy Head, Department Of Electronics and Computer Engineering for their regular support, cooperation, and coordination. The in-time facilities provided by the department throughout the Bachelors program are also equally acknowledgeable.

We would like to convey our thanks to the teaching and non-teaching staff of the Department of Electronics & Communication and Computer Engineering, ACEM for their invaluable help and support throughout the period of Bachelor's Degree. We are also grateful to all our classmates for their help, encouragement and invaluable suggestions.

Finally, yet more importantly, I would like to express our deep appreciation to my grandparents, parents, siblings for their perpetual support and encouragement throughout the Bachelor's degree period.

PRAFUL SAPKOTA (30156)

PRASANNA KHADKA (30161)

PRATIK RAJ LAUDARI (30162)

ABSTRACT

Machine Learning is an application of Artificial Intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. In this paper, we've put forth an expressive style classification approach using Machine Learning technique. Music plays a really important role in people's lives. Music brings like-minded people together and is the glue that holds communities together. Communities may be recognized by the kind of songs that they compose, or maybe hear. Within the area of Music Information Retrieval (MIR), categorizing musical genres could be a challenging task. The aim of our project and research is to seek out a far better machine learning algorithm than the pre-existing models that predict the genre of songs. Genres may be defined as categorical labels created by humans to spot or characterize the design of music. The concept of automatic expressive style classification has become very talked-about in recent years as a result of the rising of digital show business. This work presents a comprehensive machine learning approach to the matter of automatic style classification using the audio signal. The system is developed employing a K- Nearest Neighbor (KNN) to acknowledge the genres. Here KNN uses proximity to make classifications or predictions about the grouping of an individual data point. We are conducting the experiment on our own data-set.

Keywords: *Genre, Music Information Retrieval (MIR), KNN, Dataset, MFCC*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATION

AI: Artificial Intelligence

KNN: K- Nearest Neighbor

MFCC: Mel Frequency Cepstral Coefficients

SVM: Support Vector Machines

FFT: Fast Fourier Transform

CHAPTER 1

INTRODUCTION

1.1 Background

Music is divided into arbitrary groups known as genres. Applications that deal with musical databases have become more important as a result of the development of the internet and multimedia systems, and the need for Music Information Retrieval (MIR) applications has grown. Since they result from a complex interplay between the general audiences, marketing, historical, and cultural variables, musical genres lack specific definitions and boundaries. Some academics have proposed the definition of a new genre classification system specifically for the purposes of music information retrieval as a result of this observation. Genre hierarchies are now one of the methods used to organize music information on the Web. These hierarchies are mainly developed manually by human experts. A significant part of a comprehensive music information retrieval system for audio signals can be provided by automatic musical genre classification, which has the potential to automate this process.

We use audio data set which contains 450 music pieces each of 30 seconds length. There are 50 pieces from each of the following genres: classical (cl), country(co), disco(d), hip-hop(h), jazz(j), rock(ro), blues(b), reggae(re), pop(p), metal(m). Later for our web-app we have chosen six popular genres namely classical, classical, hip-hop, jazz, metal, pop to get more accuracy. We use pattern recognition algorithms with Mel Frequency Cepstral Coefficients (MFCC) as the feature vectors for classification. We have tried different supervised learning algorithms for our classification problem .Input can be in any audio/video format. The final output will be a label from the 6 genres.

1.2 Motivation

Genre classification is a standard problem in Music Information Retrieval research. Most of the music genre classification techniques employ pattern recognition algorithms. The features are extracted from short-time recording segments. Commonly used classifiers are Support Vector Machines (SVMs), Nearest-Neighbor (NN) classifiers, Gaussian Mixture Models, Linear Discriminant Analysis (LDA), etc. There have been many researches carried out on music genre classification, but none have implemented in a way that the common people without mathematical or programming knowledge can use to find genre of a music. Our attempt is to provide an application that can accept any audio/video file and output a genre label from it.

1.3 Statement of the Problem

Classifying a song to a genre, although an inherently subjective task, comes quite easily to the human ear. Within seconds of hearing a new song one can easily recognize the timbre, distinct instruments, beat, chord progression, lyrics, and genre of the song. For machines on the other hand this is quite a complex and daunting task as the whole "human" experience of listening to a song is transformed into a vector of features about the song. Historically, machines haven't been able to reliably detect many of these musical characteristics that humans recognize in music. We are considering a 10-genre classification problem with the following categories: 'adhunik', 'bhajan', 'bhojpuri', 'classical', 'dohari', 'gazal', 'newa', 'pop', 'rock', 'selo'. We use pattern recognition algorithms with Mel Frequency Cepstral Coefficients (MFCC) as the feature vectors for classification.

1.4 Project objective

The main objectives of music genre classification is to extract the features from the given music and classify them based in the extracted features.

And some feature of the project are.

- To quantify the match of any input music to the listed genres.
- To improve upon the accuracy of genre classification.
- To find the multiple genres to which a music belongs to.

1.5 Significance of the study

- The final Graphical User Interface can be used to identify the genre of any given audio/video file.
- With the help of our classifiers, we can label large data set of untagged music.
- The classifiers can be integrated into any application for Music Information Retrieval.

CHAPTER 2

LITERATURE REVIEW

A music genre classifier is a software program that predicts the genre of a piece of music in audio format. These devices are used for tasks such as automatically Tagging music for distributors such as Spotify and Billboard and determining appropriate background music for events.

2.1 Automatic Musical Genre Classification of Audio Signals

George Tzanetakis has presented a method for automatically classifying audio signals that uses three feature sets to indicate pitch content, rhythmic content, and texture. Through the use of real-world audio collections for training statistical pattern recognition classifiers, the effectiveness and relative relevance of the proposed features are evaluated. There are descriptions of both full file and real-time frame-based classification techniques. A categorization of 61% for ten musical genres is achieved using the suggested feature sets. This outcome was equivalent to results for the classification of musical genres by humans.

A new feature extraction method for music genre classification were proposed by Tao Li, in 2003. They capture the local and global information of music signals simultaneously by computing histograms on their Daubechies wavelet coefficients. Effectiveness of this new feature and of previously studied features are compared using various machine learning classification algorithms, including Support Vector Machines and Linear Discriminant Analysis.

2.2 Audio Music Genre Classification Using Different Classifiers and Feature Selection Methods

An approach by Yusuf Yaslan and Zehra Cateltepe in 2006. They examined performances of different classifiers on different audio feature sets to determine the genre of a given music piece. For each classifier, they also evaluated performances of feature sets obtained by dimensionality reduction methods. They used the freely available MARSYAS software to extract the audio features. Finally, they experimented on increasing classification accuracy by combining different classifiers. Using a set of different classifiers, they obtained a test genre classification accuracy of around 79.6 ± 4.2% on 10 genre set of 1000 music pieces.

There have been no published works that perform large scale genre classification using cross-model methods. Dawen Liang in 2011, proposed a cross-model retrieval framework of model blending which combines features of lyrics and audio. The algorithm scales linearly in the number of songs, and evaluate it on a subset of the MSD (Million Song Dataset) containing 156,289 songs. There is no previous work on genre classification measuring the likelihood of different genre-based HMMs (Hidden Markov Models), or bag-of-words lyric features.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 Project Requirements

3.1.1 Hardware Requirements

- Processor: Intel Core i3 and higher
- Speed:2.1 GHz
- RAM:8GB
- Hard Disk: 40 GB and above.

3.1.2 Software Requirements

- Operating System: Linux/Windows
- Programming Language: Python v3.11.0
- IDE: VS Code

3.1.3 Library Used

- pysimplegui - GUI designer
- numpy - Data analysis and manipulation
- librosa - Mel-frequency cepstral coefficients (for spectrograph creation)
- matplotlib - creating static, animated, and interactive visualizations

3.2 Functional Requirements

The functional requirements describe the core functionality of the model. The functional requirements essentially describe what are expected from a system. They specify the task and functionalities of the system. These are collected from users based on user requirements. These are collected from clients as functional requirements document and developers work for implementing all the features. The project functional requirements are as follows:

1. This model requires 16-bit audio files in .WAV format. Each audio file should be 30seconds long.
2. Datasets must be collected and maintained.
3. User has to provide the file path of music files to be classified and gets the output as predicted music type.

3.3 Non-Functional Requirements

Non-functional requirements are those requirements of the system which are not directly concerned with specific functionality delivered by the system. These are mainly concerned with the functionalities that are not mentioned as the core functionalities of project. They may be related to emergent properties such as reliability, usability etc.

- Ease of use
- Availability
- Reliability
- Maintainability

CHAPTER 4

SYSTEM DESIGN AND ARCHITECTURE

4.1 Block Diagram of overall system

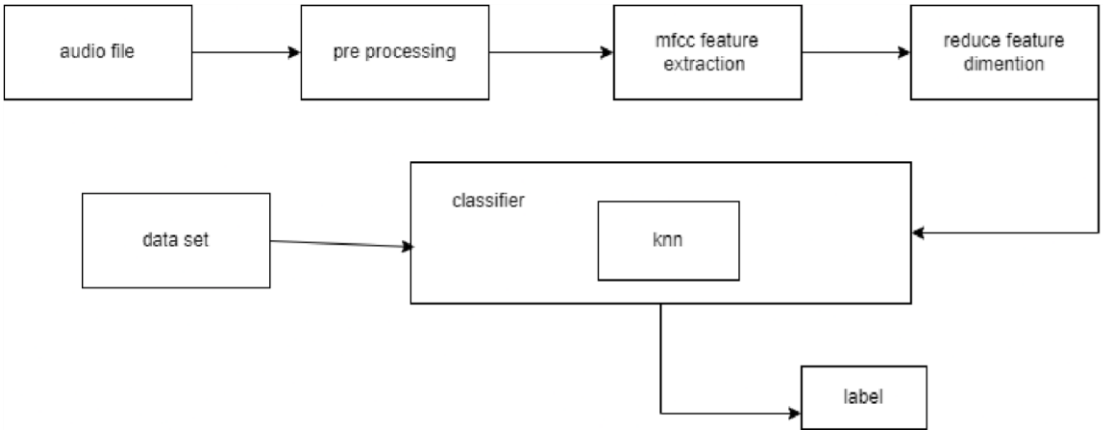

Figure 4.1 Block diagram of overall system

The first step for genre classification is to separate features and segments from the sound records. It includes recognizing the linguistic contents and disposing of noise. Pitch and MFCC features are removed from sound records. These features are utilized to train a KNN classifier. At that point, new sound records that should be arranged go through a similar feature extraction. The trained KNN classifier predicts which one of the 10 genres is the nearest match.

4.2 Flow chart

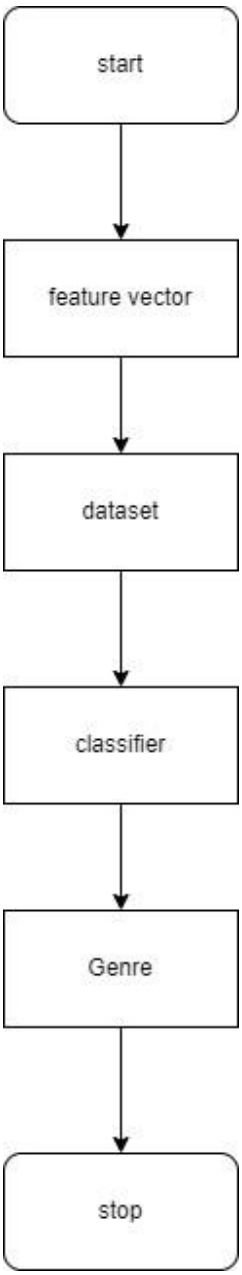

Figure 4.2 Flow chart of classifier

The flow chart diagram 4.1 of feature extraction deals with the initial process of extracting features in which, from the given input audio file feature extraction takes place in the form of feature vectors like MFCC, which is to be used in the next module to classify the music by the classifiers.

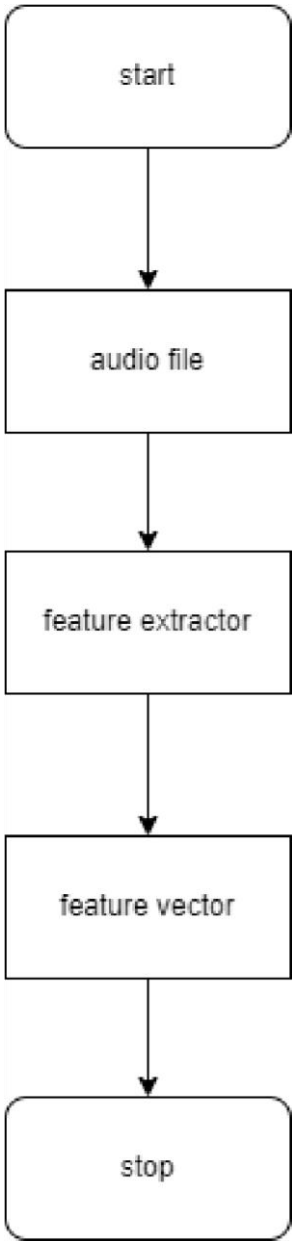

Figure 4.3 Flow Chart of Feature Extraction

The flow chart diagram of fig 4.2 for classifier deals with the training and testing the different types of classifiers by the feature vectors to produce the output to the user as a label. The different types of classifiers that can be used are k-NN, SVM, Logistic regression, etc.

4.3 Use Case Diagram

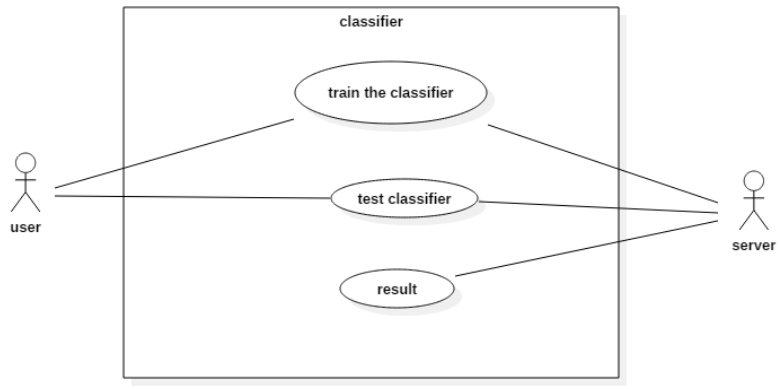

Figure 4.4 Use case diagram of classifier

The figure 4.4 shows the use case diagram for the application. In our use case diagram user uploads the audio file to the server, then the server response with the genre or label to the user.

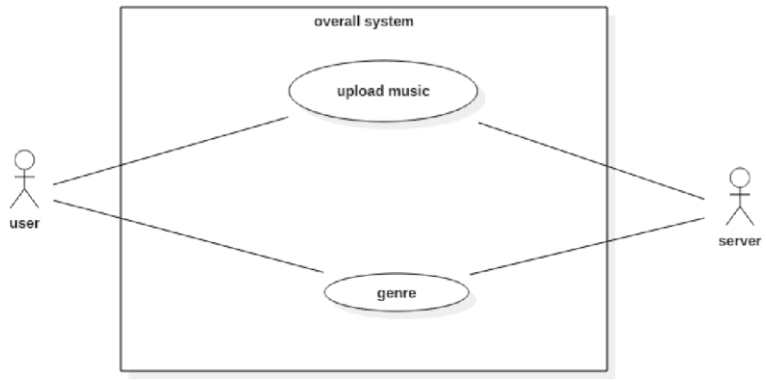

Figure 4.5 Use Case diagram of overall system

The Fig 4.5 shows the use case diagram for the classifier. The above diagram at its simplest is a representation of a user's interaction with the system and depicting the specification of a use case. It deals with training the classifier with the extracted features vectors and combines the classifiers to improve the accuracy of the genre after that testing the classifier for the result and display it.

4.4 Data flow diagram

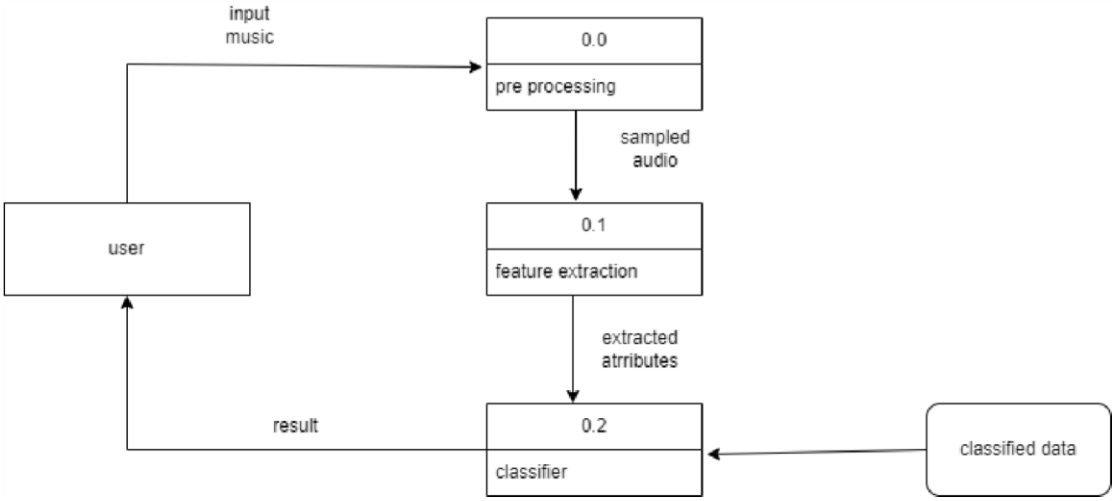

Figure 4.6 Data Flow diagram (level -0) of overall system

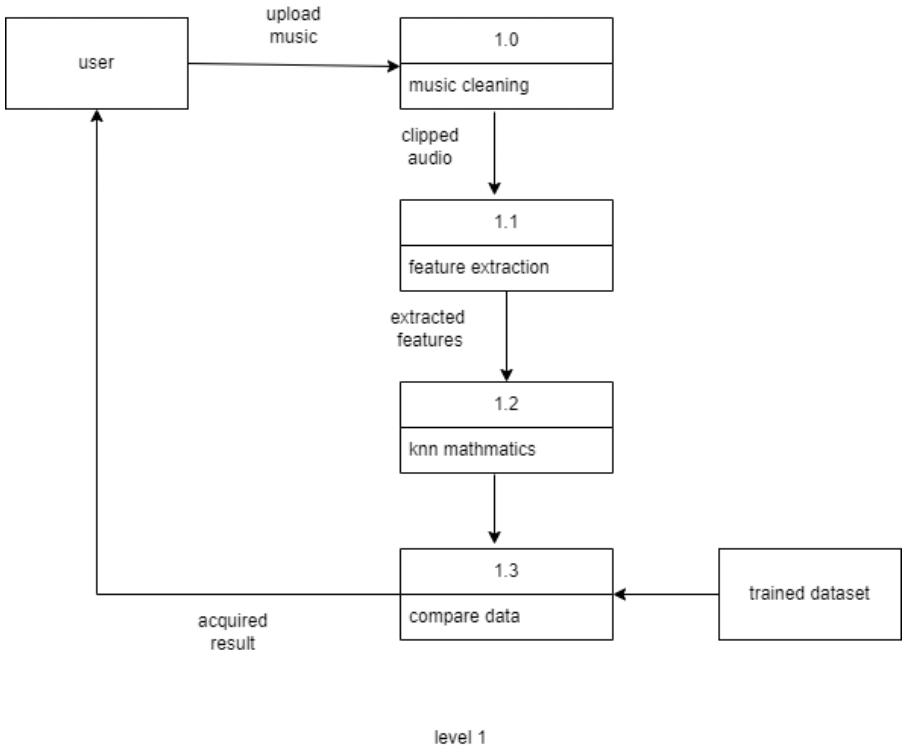

Figure 4.7 Data Flow diagram (level -1) of overall system

CHAPTER 5

METHODOLOGY

The classification of music into genres is a fundamental component of an effective recommendation system. This project's goal is to learn how to work with sound files in Python, calculate sound and audio features from them, apply Machine Learning algorithms to them, and evaluate the results. The major objective is to develop a machine learning model that classify music samples into various genres in a more systematic way. It uses an audio stream as its input and seeks to identify the genre. Making the selection of songs easier and quicker is the goal of automating the music classification. This takes a lot of time and is challenging. Automating music classification can make it easier to locate important information like trends, popular genres, and performers.

5.1 Preprocessing and Features

First, we made the audio data normal for each song to remove the base volume difference where different pieces were recorded, which did not affect their genre. Crude audio is hard to work with as it contains too many data points (22,500 per second) to be computer-generated on multiple sensory networks. It can take a very long time to train, and the data details will make pattern recognition difficult without a large model with prohibition. Therefore, we examined two other integrated data representation methods: Fourier-transform coefficient and Mel-frequency cepstral coefficients (MFCC). Performing Fourier transform involves breaking the audio sample into small segments (~0.1 seconds), and taking a Fast Fourier Transform (FFT) of each segment. The resulting Fourier coefficients vectors were stacked along the time axis to form a 4 time-series matrix of Fourier coefficients, which can be treated like an “image” when training. The FFT was performed using a NumPy function.

Making changes to the MFCC involves the following steps:

1. Take a Fourier version of each audio component
2. Spectacle power map obtained above Mel scale, using triangular windows
3. Take power logarithms for each Mel-frequencies
4. Take the direct cosine-transform of the Mel-log power list

5. MFCCs are the amplitudes of the resulting spectrum

This process is done using the Librosa library (a tool designed for audio processing). We tried the MFCC because according to, it is the industry standard for sound processing, and the author noted that it leads to the development of significant accuracy.

We also tried to pre-process our data by converting crude audio into Mel spectrograms. In doing so, we have experienced a significant increase in performance across all models. Mel-spectrograms are a commonly used method of input sound because they closely represent the way people perceive sound (i.e., at log frequency). To convert crude sound into Mel spectrogram, one must apply Fourier transitions to sound slide windows, usually around 20ms wide. With signal x[n], window w[n], frequency axis ω, and shift m, this is computed as

$$\mathbf{STFT}\{x[n]\}(m,\omega) = \sum_n x[n]w[n-m]e^{-j\omega n}.$$

These are computed more quickly in practice using sliding DFT algorithms. These are then mapped to the Mel scale by transforming the frequencies f by

$$m = 2595\log_{10}(1+\frac{f}{700}).$$

Then we took a discrete cosine effect modification (common in signal processing) to obtain our final Mel spectrogram effect. In our case, we used the Librosa library and chose to use 64 Mel-bins and a 512-sample length window with 50% spacing between windows. Based on previous academic achievements with such a change, we then proceed to measure the log using a formula log (X2). The resulting data can be visualized below:

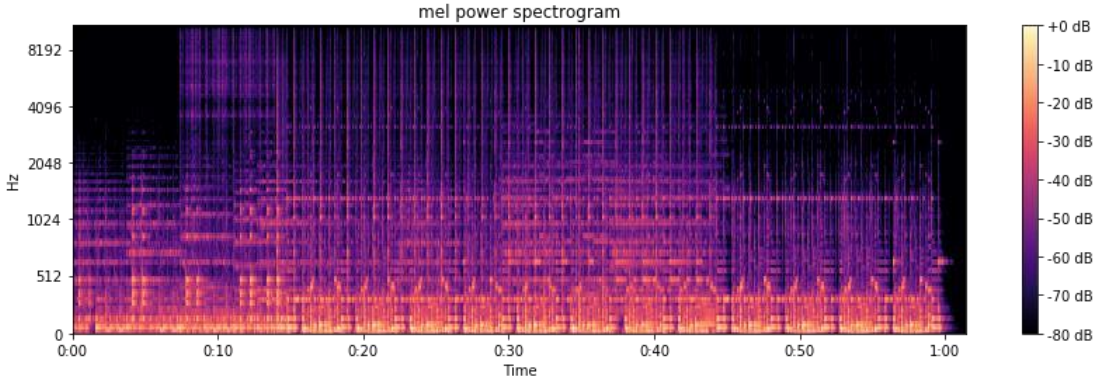

Figure 5.1 Mel Spectrogram

https://Source:source-separation.github.io/tutorial/first_steps/byo_hpss.html(9-122022/04:10)

We decided to use K-Nearest Neighbor (KNN) algorithm for classifying the music genre.

5.2 K-Nearest Neighbor (KNN)

The k-nearest neighbors' algorithm (k-NN) is a non-parametric supervised learning method first developed by Evelyn Fix and Joseph Hodges in 1951, and later expanded by Thomas Cover. It is used for classification and regression. In both cases, the input consists of the k closest training examples in a data set. The KNN algorithm is based on the idea that similar objects live close together. In other words, the same data points are closer. The output depends on whether k-NN is used to split or subtract. K-NN is a type of division where work is measured only locally and all calculations are reversed until the work is evaluated. As this algorithm depends on the distance to be distinguished, if the features represent different body units or come in very different scales to make normal training data can improve its accuracy dramatically [20]. In both divisions and regimes, a practical approach would be to allocate weights to neighbors' contributions, so that nearby neighbors may contribute more to the measure than those far away. For example, a standard measurement scheme consists of giving each neighbor a weight of 1 / d, where d is the distance to the neighbor. Neighbors are taken from a group of items class (k-NN division) or item of property (k-NN reclassification). This can be thought of as algorithm-set training, although no specific training step is required.

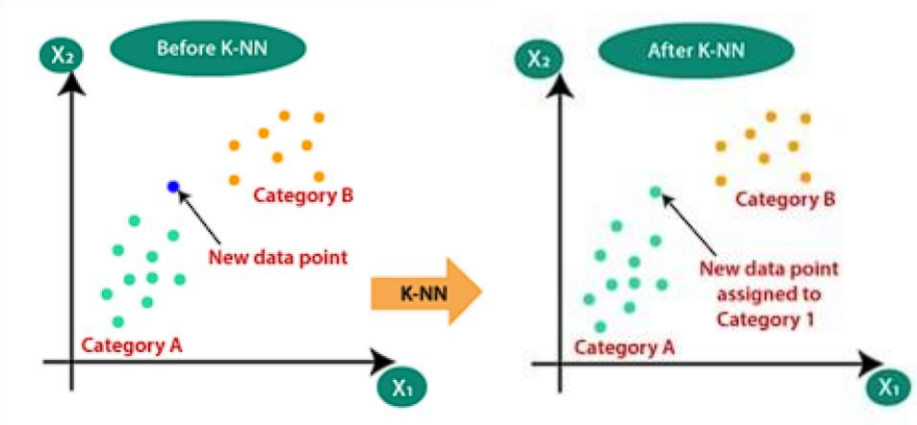

Figure 5.2 KNN Plotting

https://medium.com/geekculture/writing-knn-in-python-from-scratch-67123907165(9-12-2022/04:10)

Finding comparable nearby locations, distance measurements including Euclidean distance, hamming distance, Manhattan distance, and Malikowski distance are used.

The length of a line segment connecting two points in Euclidean space is defined as the Euclidean distance.

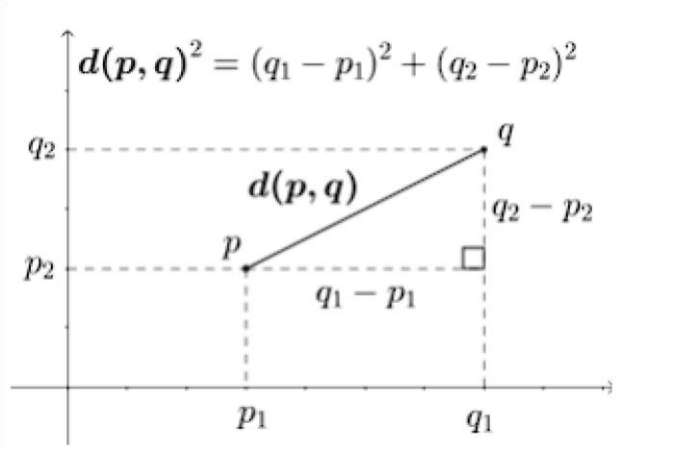

Figure 5.3 Euclidean distance

https://towardsdatascience.com/k-nearest-neighbour-explained-part-2f23d14fcb8c(9-12-2022/04:10)

5.2.1 KNN Algorithm

Examples of vector training are in the multi-dimensional feature area, each with a class label. The training section of the algorithm consists only of keeping feature vectors and class labels for training samples. In the classification phase, k is a user-defined condition, and a vector without a label (question or test point) is separated by providing the most common label between k training samples adjacent to that question point. The most commonly used distance metrics for continuous Euclidean distance fluctuations. For various variations, such as text separating, other metrics can be used, such as scattering metrics (or Hamming range). In the context of genetic microarray data, for example, k-NN is used with compatible coefficients, such as Pearson and Spearman, as metrics. In general, the accuracy of the k-NN phase can be greatly improved if the distance metric is read by special algorithms such as Large Margin Nearest Neighbor or analysis of Neighborhood components. The reversal of the basic “voting for the majority” phase occurs when class distribution is distorted. That is, the examples of the most common class tend to dominate the predictions of the new model, because they are more common among nearby neighbors because of their large numbers. One way to overcome this problem is to measure the distance, taking into account the distance from

the checkpoint to each of the nearest neighbors. The phase (or value, in reversal problems) of each adjacent k point is multiplied by weight in proportion to the distance from that point to the test point. Another way to overcome skew by removing data representation. For example, on a map (SOM), each node represents (center) a set of similar points, regardless of their density in the initial training data. K-NN may be used in SOM.

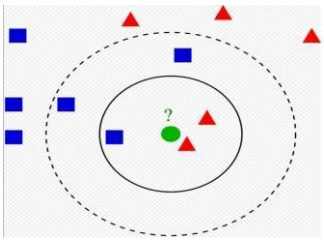

Figure 5.4 KNN Algorithm plot

https://towardsdatascience.com/fast-near-duplicate-image-search-using-locality-sensitive-hashing-d4c16058efcb(9-12-2022/04:10)

CHAPTER 6

RESULT AND ANALYSIS

6.1 Components

This project of music genre classification was divided into 4 major parts they were

- Feature extractor
- Classifier
- Data set
- GUI

6.1.1 Feature extractor

Feature extractor is the part of the program which convert the provided audio file into array file. Then it extract the features by manipulation of the array file. Features which are extracted include:

- Chroma Shift
- RMS
- Spectral Centroid
- Spectral Bandwidth
- Spectral Roll of
- Zero Crossing Rate
- Harmonics
- Tempo
- MFCC
- MFCC 1st Derivative
- MFCC 2nd Derivative

Then the extracted feature's average and variance are calculated. Then the calculated file are save in Comma Separated Variable format.

6.1.2 Classifier

Classifier is another part of this program which first take the values from CSV file then extract same features from newly uploaded audio(test audio). Then calculate the distances between using features as parameter and Euclidean distance formula and send the value of nearest distance to determine the genre of the audio file.

6.1.3 Data Set

We manually created dataset. For which we downloaded in total 400 songs each genre containing 40 songs. The audio file was then cut into 30 seconds sample and labeled its genre. Which feature was extracted using previously mentioned feature extractor.
The genre we are classifying are:

- Aadhunik
- Bhajan
- Bhojpuri
- Classical
- Soft pop
- Ghazal
- Newa
- Tamang
- Rock

6.1.4 GUI

We created simple GUI which can browse file from the computer, send the location of the file to classifier and play or stop the selected audio file.

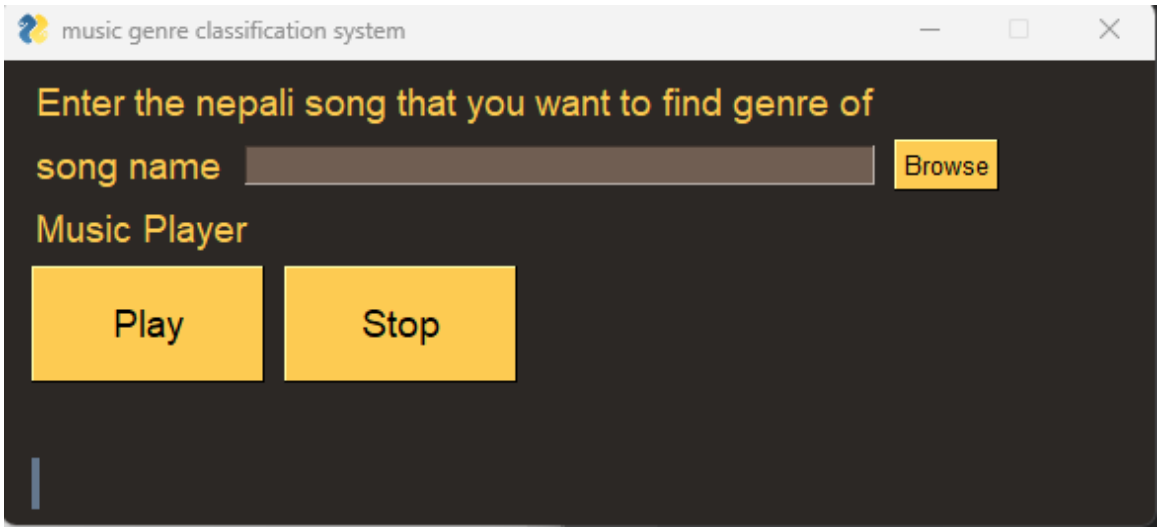

Figure 6.1 GUI on initialization

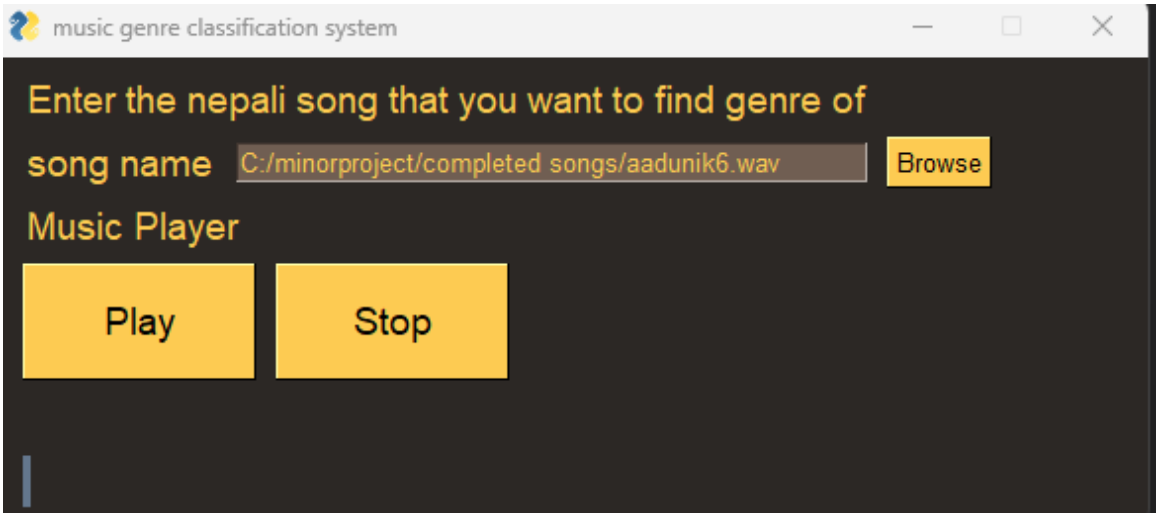

Fig 6.2 GUI after browsing

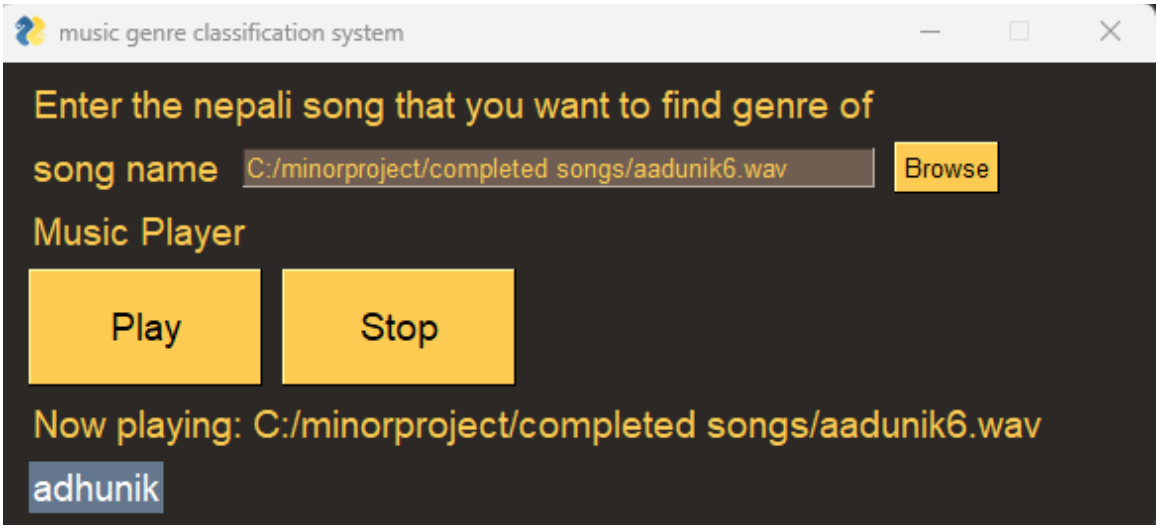

Fig 6.3 GUI after classification

6.2 Analysis

After feature extraction we expected similar songs feature are equivalent to each other. So to show their similarity we plotted a graph. As we are considering 21 parameters for classification putting each parameter as dimension was not possible so we created a 2-D graph having average MFCC and average Tempo on it's y and x axis. So we can see mostly the plotting of same genre are comparatively together.

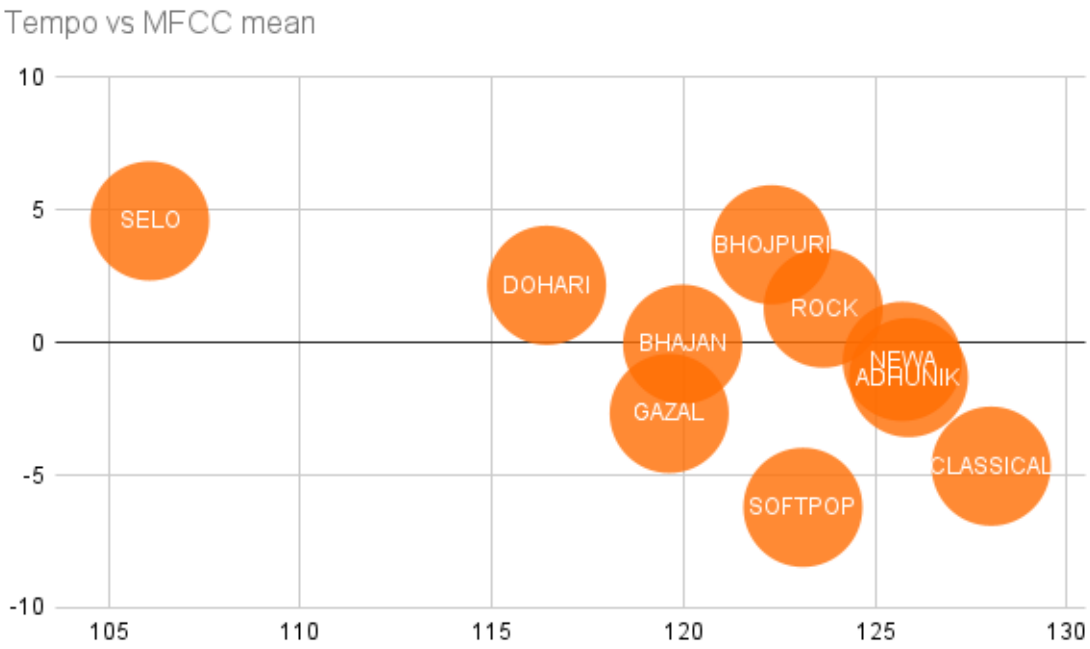

Fig:6.4 Tempo Vs MFCC graph of genre

6.3 Confusion matrix

Training Set											
TARGET / OUTPUT	Adhunik	Bhajan	Bhojpuri	Classical	Dohari	Gazal	Newa	Pop	Rock	Selo	SUM
Adhunik	2 4.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	2 4.00%	0 0.00%	1 2.00%	0 0.00%	0 0.00%	5 40.00% 60.00%
Bhajan	0 0.00%	3 6.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	1 2.00%	1 2.00%	0 0.00%	0 0.00%	5 60.00% 40.00%
Bhojpuri	2 4.00%	0 0.00%	2 4.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	1 2.00%	0 0.00%	0 0.00%	5 40.00% 60.00%
Classical	0 0.00%	0 0.00%	0 0.00%	3 6.00%	1 2.00%	1 2.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	5 60.00% 40.00%
Dohari	0 0.00%	0 0.00%	0 0.00%	0 0.00%	4 8.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	1 2.00%	5 80.00% 20.00%
Gazal	0 0.00%	1 2.00%	0 0.00%	0 0.00%	0 0.00%	3 6.00%	0 0.00%	0 0.00%	0 0.00%	1 2.00%	5 60.00% 40.00%
Newa	0 0.00%	0 0.00%	0 0.00%	0 0.00%	2 4.00%	1 2.00%	2 4.00%	0 0.00%	0 0.00%	0 0.00%	5 40.00% 60.00%
Pop	0 0.00%	0 0.00%	1 2.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	4 8.00%	0 0.00%	0 0.00%	5 80.00% 20.00%
Rock	0 0.00%	0 0.00%	0 0.00%	0 0.00%	1 2.00%	1 2.00%	0 0.00%	0 0.00%	2 4.00%	1 2.00%	5 40.00% 60.00%
Selo	0 0.00%	0 0.00%	0 0.00%	0 0.00%	1 2.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	4 8.00%	5 80.00% 20.00%
SUM	4 50.00% 50.00%	4 75.00% 25.00%	3 66.67% 33.33%	3 100.00% 0.00%	9 44.44% 55.56%	8 37.50% 62.50%	3 66.67% 33.33%	7 57.14% 42.86%	2 100.00% 0.00%	7 57.14% 42.86%	29 / 50 58.00% 42.00%

Fig 6.5 Confusion matrix

6.4 Accuracy

The software was initially evaluated using a sample dataset made up of 50 songs from various genres. The results from the evaluation is given below.

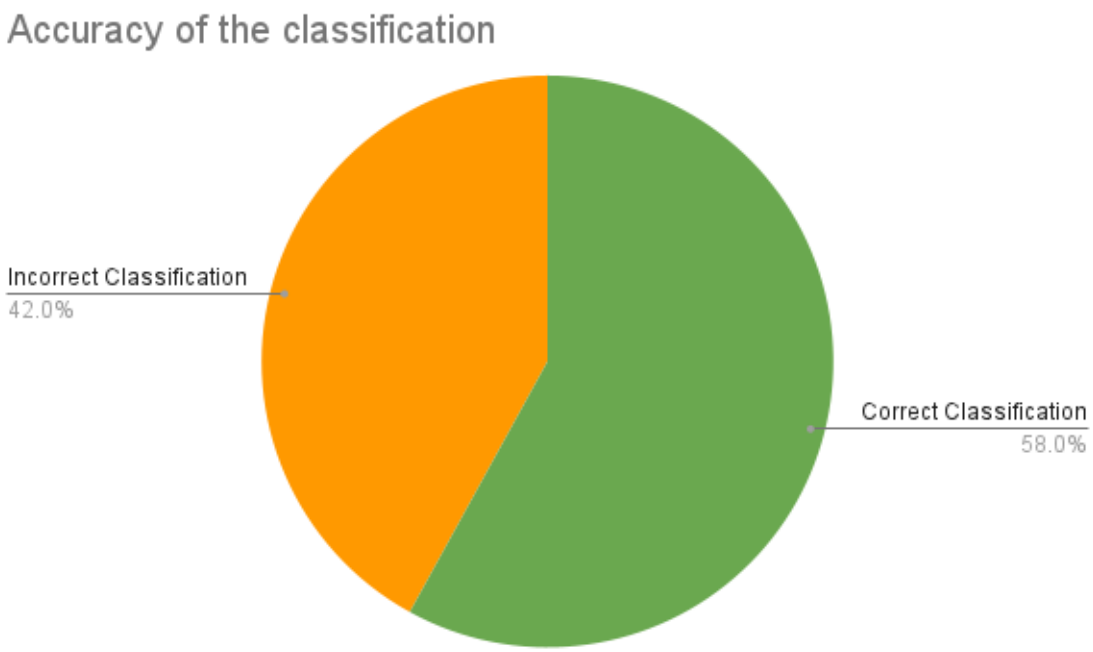

Fig6.6 Overall Accuracy pie chart

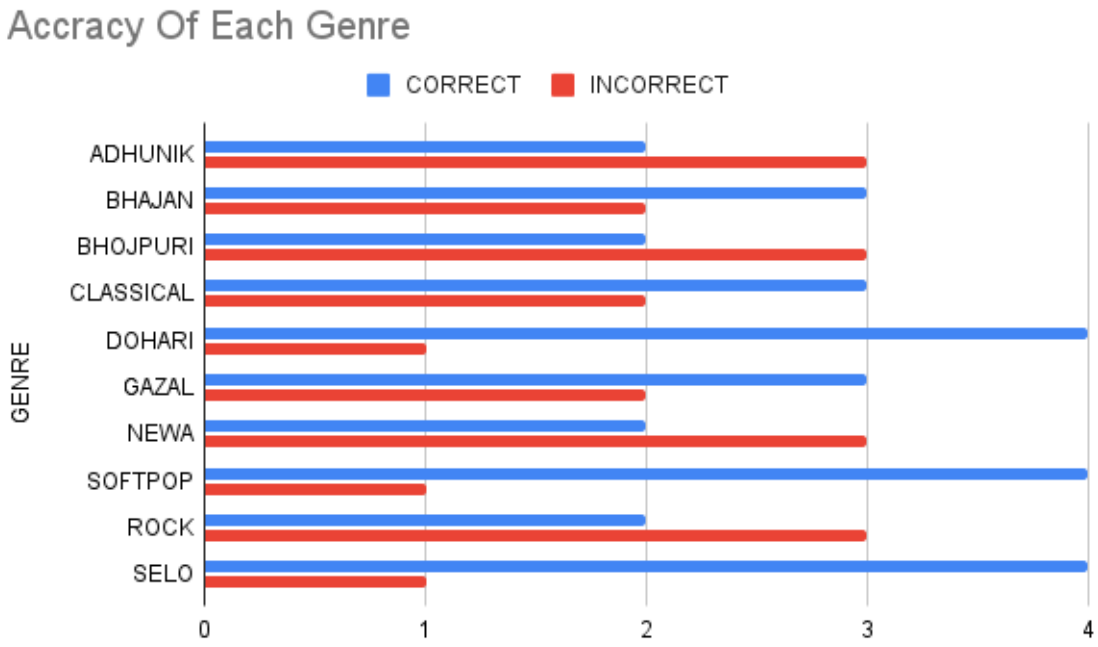

Fig 6.7 Accuracy of each genre

CHAPTER 7

CONCLUSION, LIMITATIONS AND FUTURE ENHANCEMENTS

7.1 Conclusion

The project was more difficult than we anticipated, and there were many challenges to be overcome. The majority of them involved the data and files we required for the project. The major challenges were:

- The dataset was manually created which was a tedious task as it required editing of audio file.
- Frequent changes were brought to different library used which made programming difficult.

Despite obstacles and issues, we finished the assignment within the allotted time. Finally a GUI with options for file viewing, audio interaction, and processing was ultimately produced for the project. The accuracy of the program was found to be about 58%.

7.2 Limitation and Future Enhancement

While talking about the limitation the major limitation of the project are:

- The file format supported by the program is WAV. Due to the library used other files are not supported.
- The number of genre that can be classified is only 10. Any song of other genre will be classified with in these 10 genre which is not ideal.
- The classification does not take account of fusion music i.e. the song made in different genre.
- The GUI of the system is very basic.

The future enhancement that can be done to the project are:

- Improvement in support of other file formats such as mp3, m4a, webm, etc.
- Increase in the number of genre so there is less generalization.
- Determination of multiple genre for same song.
- Improvement in accuracy by increasing the no of songs in dataset.
- Time and space complexity improvement through optimization.

CHAPTER 7

REFERENCES

1. Y. Yaslan and Z. Cataltepe, “Music genre classification using audio features, different classifiers and feature selection methods,” *2006 IEEE 14th Signal Processing and Communications Applications*.
2. B. McFee, C. Raffel, D. Liang, D. Ellis, M. McVicar, E. Battenberg, and O. Nieto, “Librosa: Audio and Music Signal Analysis in python,” *Proceedings of the 14th Python in Science Conference*, 2015.
3. B. McFee, C. Raffel, D. Liang, D. Ellis, M. McVicar, E. Battenberg, and O. Nieto, “Librosa: Audio and Music Signal Analysis in python,” *Proceedings of the 14th Python in Science Conference*, 2015. .
4. M. Li and H. Wang, “Unsupervised deep cross-modal hashing by knowledge distillation for large-scale cross-modal retrieval,” *Proceedings of the 2021 International Conference on Multimedia Retrieval*, 2021. .
5. L. Breiman, “Bagging predictors,” *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.
6. P. Pandey, “Music genre classification with python,” *Medium*, 11-May-2021. [Online]. Available: https://towardsdatascience.com/music-genreclassification-with-python-c714d032f0d8. [Accessed: 06-Feb-2023].
7. L. Breiman, Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.