MAR ATHANASIUS COLLEGE OF ENGINEERING
(Affiliated to APJ Abdul Kalam Technological University, TVM)
KOTHAMANGALAM

Department of Computer Applications

Mini Project Report

MUSIC GENRE CLASSIFICATION USING MACHINE LEARNING

Done by

Akhil Rock Babu

Reg No:MAC23MCA-2010

Under the guidance of
Prof. Manu John

MAR ATHANASIUS COLLEGE OF ENGINEERING
(Affiliated to APJ Abdul Kalam Technological University, TVM)
KOTHAMANGALAM

CERTIFICATE

Music Genre Classification Using Machine Learning

Certified that this is the bonafide record of project work done by

Akhil Rock Babu
Reg No: MAC23MCA-2010

during the third semester, in partial fulfilment of requirements for award of the degree

Master of Computer Applications

of

APJ Abdul Kalam Technological University Thiruvananthapuram

Faculty Guide	**Head of the Department**
Prof. Manu John	Prof. Biju Skaria
Project Coordinator	**Internal Examiners**
Prof. Sonia Abraham	

ACKNOWLEDGEMENT

With heartfelt gratitude, I extend my deepest thanks to the Almighty for his unwavering grace and blessings that have made this journey possible. May his guidance continue to illuminate my path in the years ahead.

I am immensely thankful to Prof. Biju Skaria, Head of the Department of Computer Applications, Prof. Manu John, my project guide, and Prof. Sonia Abraham, our dedicated project coordinator, for their invaluable guidance and timely advice, which played a pivotal role in shaping this project. Their guidance, constant supervision, and provision of essential information were instrumental in the successful completion of the mini project.

I extend my profound thanks to all the professors in the department and the entire staff at MACE for their unwavering support and inspiration throughout my academic journey. My sincere appreciation goes to my beloved parents, whose guidance has been a beacon in every step of my path.

I am also grateful to my friends and individuals who generously shared their expertiseand assistance, contributing significantly to the fulfillment of this endeavor.

ABSTRACT

The project aims to create a music genre classification system using machine learning algorithms, specifically k-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Naive Bayes (NB), to accurately predict the genre of unseen audio files using the GTZAN dataset and Python Librosa package.

The music industry leverages genre classification for enhanced user experience, personalized content, and targeted audience targeting in recommendation engines. This system is crucial in organizing digital music libraries, enhancing user experience, and enhancing the effectiveness of music producers and marketers.

Machine learning algorithms have been found to be effective in classifying music genres. Studies have shown that SVM with an RBF kernel is accurate by 87.5%, while KNN outperforms other models with an accuracy of 92.69%, surpassing CNN's 72.40%. KNN's high accuracy and efficiency make it a leading algorithm for this task.

The GTZAN dataset will be used to evaluate three machine learning algorithms for identifying music genres: KNN, SVM, and NB. The KNN algorithm will be evaluated for accuracy, computational efficiency, and robustness. SVM will be assessed for high-dimensional data handling, while Naive Bayes will be considered for its rapid and probabilistic approach. The goal is to find the most accurate algorithm.

The comprehensive dataset includes 10,000 pre-extracted features from 1,000 audio tracks, categorized into 10 music genres. These features, extracted using the Librosa package, are normalized and used to train machine learning models, ensuring a reliable classification system for each genre.

LIST OF TABLES

LIST OF FIGURES

CONTENTS

1. INTRODUCTION

In the rapidly evolving landscape of digital music, the efficient categorization of music genres is pivotal for enhancing user engagement and optimizing content delivery. Music genre classification facilitates personalized recommendations, organizes extensive digital music libraries, and supports targeted marketing strategies. As such, developing a robust and accurate music genre classification system is essential for both users and industry professionals.

This project seeks to address this need by employing advanced machine learning techniques to classify music genres. Specifically, it explores the performance of three prominent algorithms: k-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Naive Bayes (NB). The evaluation will be based on the GTZAN dataset, a well-regarded benchmark in music genre classification research, and the Python Librosa package, a powerful tool for audio analysis.

The GTZAN dataset comprises 10,000 pre-extracted feature records from 1,000 audio tracks, each 30 seconds in duration and segmented into 3-second intervals. This dataset encompasses 10 distinct genres: blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock. Key features of interest include Mel Frequency Cepstral Coefficients (MFCCs), chroma features, zero-crossing rate, and tempo. These features provide a comprehensive representation of the audio data, capturing essential musical characteristics that are critical for genre classification.

Using the Python Librosa package, features from the GTZAN dataset will be meticulously extracted and normalized. This preprocessing step is crucial for ensuring the accuracy and reliability of the subsequent machine learning models. The KNN, SVM, and Naive Bayes algorithms will then be applied to this processed data, each evaluated for its effectiveness in predicting music genres.

Prior studies have demonstrated varying levels of success among these algorithms. SVM with an RBF kernel has achieved an accuracy of approximately 87.5%, showcasing its capability to handle complex data patterns. KNN, however, has consistently delivered superior performance with an accuracy of 92.69%, outperforming other models, including Convolutional Neural Networks (CNNs) which achieved an accuracy of 72.40%. This high accuracy makes KNN a particularly strong candidate for music genre classification. Naive Bayes, with its probabilistic approach, offers rapid classification capabilities, making it an attractive option for certain applications.

The findings of this project will not only provide insights into the relative strengths of these machine learning algorithms but also offer practical recommendations for their application in real-world music classification tasks. Ultimately, the goal is to develop a system that significantly improves the accuracy and efficiency of music genre classification, thereby enhancing user experiences and supporting the broader music industry.

2. SUPPORTING LITERATURE

2.1 LITERATURE REVIEW

Paper 1: Ndou, N., Ajoodha, R., & Jadhav, A. (2021). Music Genre Classification: A review of Deep-Learning and Traditional Machine-Learning Approaches. 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS).

In the evolving landscape of audio classification, music genre identification is becoming crucial for managing large music databases. Ndou, Ajoodha, & Jadhav's paper, "Music Genre Classification: A Review of Deep-Learning and Traditional Machine-Learning Approaches," presented at the 2021 IEEE International IoT, Electronics, and Mechatronics Conference (IEMTRONICS), addresses this need by comparing traditional machine learning models with deep learning methods. The authors explore the performance of these models on features extracted from 3-second and 30-second audio segments from the GTZAN dataset, a benchmark dataset for music genre classification.

The study emphasizes that the key to accurate genre classification lies in selecting the right features. The authors implement the Information Gain Ranking algorithm to determine which features contribute most to the classification task. They evaluate machine learning algorithms such as K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Naive Bayes (NB), alongside deep learning models like Convolutional Neural Networks (CNN).

The results highlight that KNN performed the best among the machine learning models with an accuracy of 92.69% on 3-second segments, surpassing other traditional and deep-learning models. The study concludes that shorter audio segments (3 seconds) are more effective for genre classification than longer ones (30 seconds). By integrating both traditional and deep learning approaches, the authors provide a comprehensive comparison, emphasizing the potential of machine learning in enhancing music genre classification.

Paper 2: Prajwal, Shubham, Naik, & Sugna’s paper, "Music Genre Classification Using Machine Learning. International Research Journal of Modernization in Engineering Technology and Science (IRJMETS)

In the rapidly evolving field of music classification, the demand for efficient genre categorization systems has become critical as digital music platforms grow. Prajwal, Shubham, Naik, & Sugna’s paper, "Music Genre Classification Using Machine Learning," published in the International Research Journal of Modernization in Engineering Technology and Science (IRJMETS), introduces an innovative approach to classify music genres based on various audio features. The authors emphasize the increasing need for automated systems to handle large volumes of music, which manual categorization methods are unable to process effectively.

The authors make use of the GTZAN dataset, a popular dataset containing 1000 audio files, each 30 seconds long, spread across 10 distinct music genres. Key audio features such as Zero-Crossing Rate, Spectral Centroid, Chroma Features, and MFCCs (Mel-Frequency Cepstral Coefficients) were extracted using the Librosa package. These features provide crucial information about the audio signal's frequency and time domain, enabling the machine learning models to accurately differentiate between genres.

The authors evaluate several machine learning models, including K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Random Forests. They further enhance the model performance by applying feature selection techniques like Random Forest Importance and XGBoost, which identify the most critical features for classification. Additionally, hyperparameter tuning using GridSearchCV allowed the models to optimize their performance by finding the best combination of parameters for each algorithm.

The results demonstrate that the SVM model achieved the highest classification accuracy of 76.4%, followed by Random Forest with 69.6%, and KNN at 66.4%. The paper concludes that hyperparameter tuning and feature selection significantly impact the accuracy of music genre classification tasks. The authors also suggest that integrating deep learning models, such as Convolutional Neural Networks (CNNs), could further improve classification accuracy in future work, especially in handling more complex audio datasets.

Paper 3: Setiadi, D. R. I. M., Rahardwika, D. S., Rachmawanto, E. H., Sari, C. A., Irawan, C., Kusumaningrum, D. P., Nuri, N., & Trusthi, S. L. (2020). Comparison of SVM, KNN, and NB Classifier for Genre Music Classification based on Metadata. 2020 International Seminar on Application for Technology of Information and Communication (iSemantic).

In the ever-expanding domain of music streaming, efficient music genre classification is essential for providing personalized recommendations. Setiadi, Rahardwika, Rachmawanto, Sari, Irawan, Kusumaningrum, Nuri, and Trusthi's paper, "Comparison of SVM, KNN, and NB Classifier for Genre Music Classification Based on Metadata," presented at the 2020 International Seminar on Application for Technology of Information and Communication (iSemantic), addresses this challenge by comparing the performance of three machine learning algorithms: Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Naive Bayes (NB).

The authors use the Spotify music dataset, which contains complete metadata for over 228,000 music tracks across 26 genres. Instead of audio feature extraction, the research focuses on classifying genres using metadata features such as acousticness, instrumentalness, popularity, and tempo. The study tests the efficacy of using metadata, which typically requires fewer computational resources than audio-based feature extraction, to accurately classify music genres.

The experiments demonstrate that SVM with an RBF kernel achieves the highest accuracy of 80%, followed by KNN with 77.18%, and Naive Bayes with 76.08%. These results are comparable to those obtained from more complex audio-based feature extraction methods. The findings suggest that metadata can be an effective alternative for genre classification if the data is well-managed. The study highlights the relative computational efficiency of using metadata, which eliminates the need for intensive audio processing, making it faster and more scalable for large music databases.

In conclusion, the paper contributes to the growing body of research on music classification by showing that metadata-based classification can produce competitive results compared to traditional audio-based methods. The comparison of SVM, KNN, and NB offers valuable insights into which algorithms perform best for metadata-driven music classification.

2.1.1. SUMMARY TABLE

Table 2.1 Reference papers summary

TITLE	DATASET	ALGORITHM	ACCURACY
Ndou, N., Ajoodha, R., & Jadhav, A. (2021). Music Genre Classification: A review of Deep-Learning and Traditional Machine-Learning Approaches. 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS).	GTZAN dataset 10,000 Records of 60 features.	K-Nearest Neighbors (KNN) Convolutional Neural Network (CNN) Support Vector Machine (SVM)	**KNN:92.69%** CNN: 72.40% SVM :80.80%
Ghildiyal, A., Singh, K., & Sharma, S. (2020). Music Genre Classification using Machine Learning. 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA).	GTZAN dataset 10,000 Records of 60 features.	Support Vector Machine (SVM) Random Forest (RF) K-Nearest Neighbors (KNN)	**SVM: 87.5%** RF: 69.6% KNN: 66.4%
Setiadi, D. R. I. M., et al . (2020). Comparison of SVM, KNN, and NB Classifier for Genre Music Classification based on Metadata. 2020 International Seminar on Application for Technology of Information and Communication	The study uses metadata features extracted from Spotify music dataset from www.crowdai.org	Support Vector Machine (SVM) K-Nearest Neighbours (KNN) Naive Bayes classifier (NB)	**SVM: 80%** KNN:75.61% NB: 75.05%

2.2. FINDINGS AND PROPOSALS

After a comprehensive review of three key papers in the domain of music genre classification, from Table 2.1 it is evident that each contributes significantly to advancing the field through the use of machine learning models. The exploration of classification algorithms such as K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Naive Bayes (NB) highlights their effectiveness in predicting music genres from various audio features. From the insights gathered, the combination of KNN and SVM emerges as a robust solution for genre classification, offering a balance between accuracy and efficiency.

The first paper by Ndou, Ajoodha, & Jadhav provides a comparative study between deep learning and traditional machine learning approaches using audio features. KNN achieves the highest accuracy of 92.69% on 3-second segments of audio, surpassing other models, including deep learning techniques like CNN. The results suggest that smaller audio segments improve genre classification, and KNN's simplicity makes it highly effective for this task.

The Second paper by Prajwal et al. applies machine learning models such as KNN and SVM on the GTZAN dataset using key audio features extracted with the Librosa package. The paper emphasizes the importance of feature selection and hyperparameter tuning to optimize model performance. SVM achieves a strong accuracy of 76.4%, and the results show that these traditional models, with proper tuning, can still compete with more complex algorithms.

The third paper by Setiadi et al. compares SVM, KNN, and Naive Bayes for genre classification using metadata features. SVM with an RBF kernel achieves the highest accuracy of 80%, outperforming KNN and NB, demonstrating that metadata can be a strong alternative to traditional audio feature extraction methods. The use of metadata like instrumentals and tempo in classification also points to the potential of less computationally demanding approaches.

In conclusion, the combination of KNN and SVM for music genre classification is supported by empirical evidence from the reviewed papers. While KNN excels in handling audio segments and provides high accuracy, SVM brings robustness, especially with optimized kernels. This combination ensures a comprehensive understanding of genre characteristics, aligning with the evolving landscape of music classification systems and promising accurate, efficient, and scalable solutions for real-world applications.

.

3. SYSTEM ANALYSIS

3.1. ANALYSIS OF DATASET

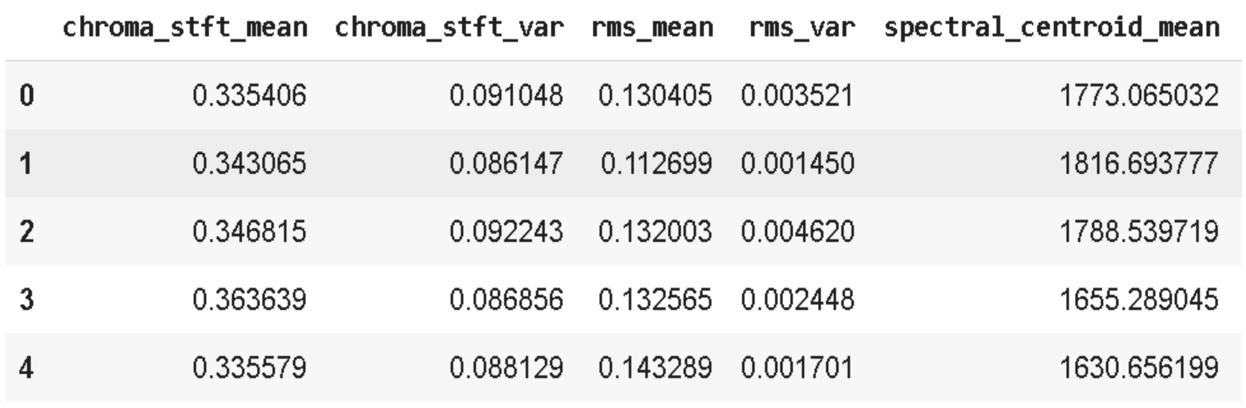
3.1.1. About the Dataset

The GTZAN music genre dataset is a widely used and well-known dataset in the field of music genre classification and audio analysis. It is commonly employed to train and evaluate machine learning models that classify music tracks based on genre.

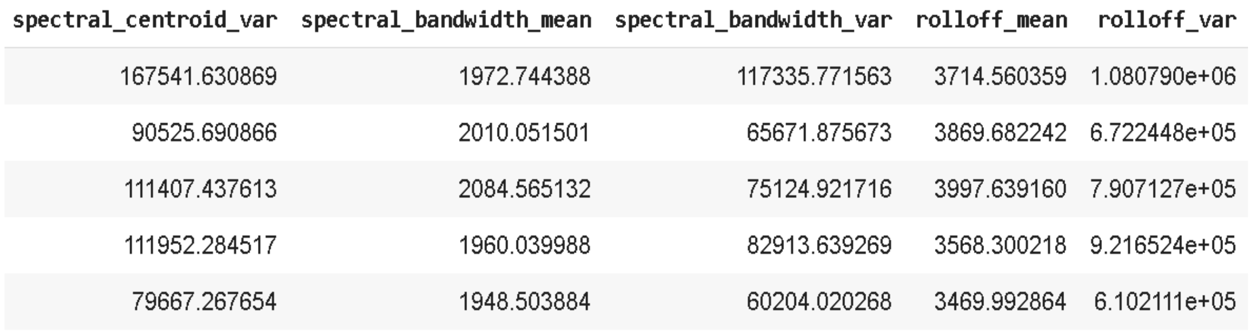
Size: The dataset contains pre-extracted features of1,000 audio tracks, each 30 seconds long, distributed evenly across 10 different genres: Blues, Classical, Country, Disco, Hip Hop, Jazz, Metal, Pop, Reggae, and Rock.

Data Split: The dataset is often split into training and testing sets, with audio features extracted from each track for machine learning classification. It includes pre-extracted features such as MFCCs, chroma, spectral contrast, and other time-frequency features used to predict genres.

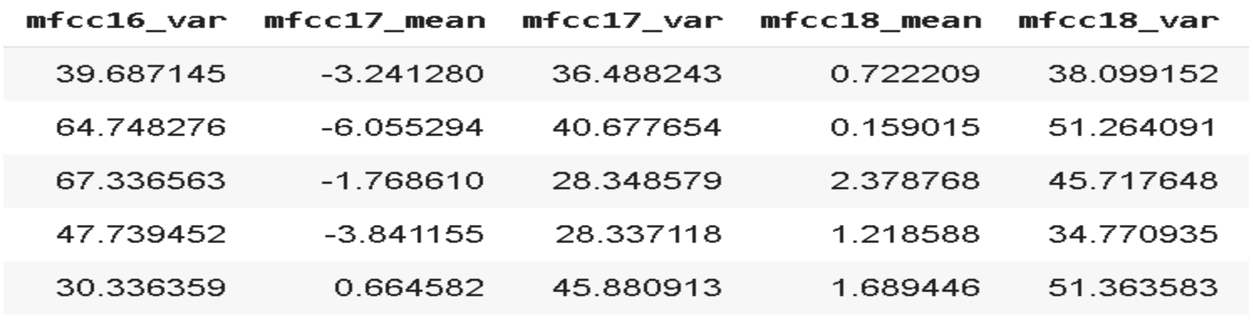
Sparsity: Since the dataset is balanced with equal representation of all genres, it provides a robust framework for developing and testing classification models, with no missing or incomplete data.



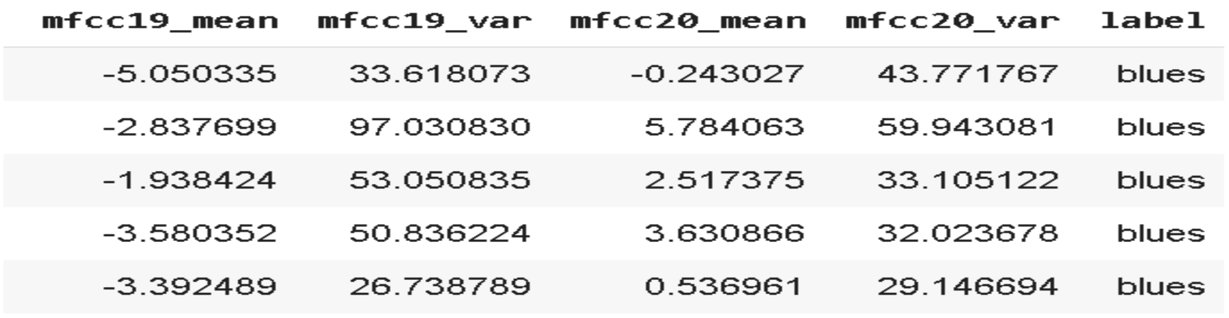
	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean
0	0.335406	0.091048	0.130405	0.003521	1773.065032
1	0.343065	0.086147	0.112699	0.001450	1816.693777
2	0.346815	0.092243	0.132003	0.004620	1788.539719
3	0.363639	0.086856	0.132565	0.002448	1655.289045
4	0.335579	0.088129	0.143289	0.001701	1630.656199



spectral_centroid_var	spectral_bandwidth_mean	spectral_bandwidth_var	rolloff_mean	rolloff_var
167541.630869	1972.744388	117335.771563	3714.560359	1.080790e+06
90525.690866	2010.051501	65671.875673	3869.682242	6.722448e+05
111407.437613	2084.565132	75124.921716	3997.639160	7.907127e+05
111952.284517	1960.039988	82913.639269	3568.300218	9.216524e+05
79667.267654	1948.503884	60204.020268	3469.992864	6.102111e+05



mfcc16_var	mfcc17_mean	mfcc17_var	mfcc18_mean	mfcc18_var
39.687145	-3.241280	36.488243	0.722209	38.099152
64.748276	-6.055294	40.677654	0.159015	51.264091
67.336563	-1.768610	28.348579	2.378768	45.717648
47.739452	-3.841155	28.337118	1.218588	34.770935
30.336359	0.664582	45.880913	1.689446	51.363583



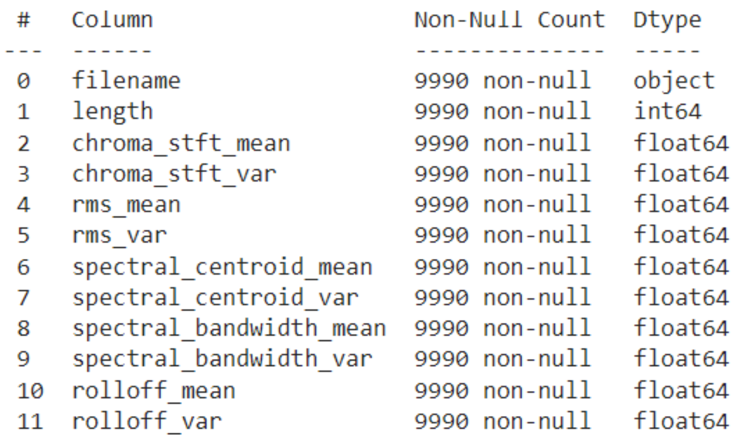
mfcc19_mean	mfcc19_var	mfcc20_mean	mfcc20_var	label
-5.050335	33.618073	-0.243027	43.771767	blues
-2.837699	97.030830	5.784063	59.943081	blues
-1.938424	53.050835	2.517375	33.105122	blues
-3.580352	50.836224	3.630866	32.023678	blues
-3.392489	26.738789	0.536961	29.146694	blues

Figure 3.1 Snapshots of features_30_sec.csv

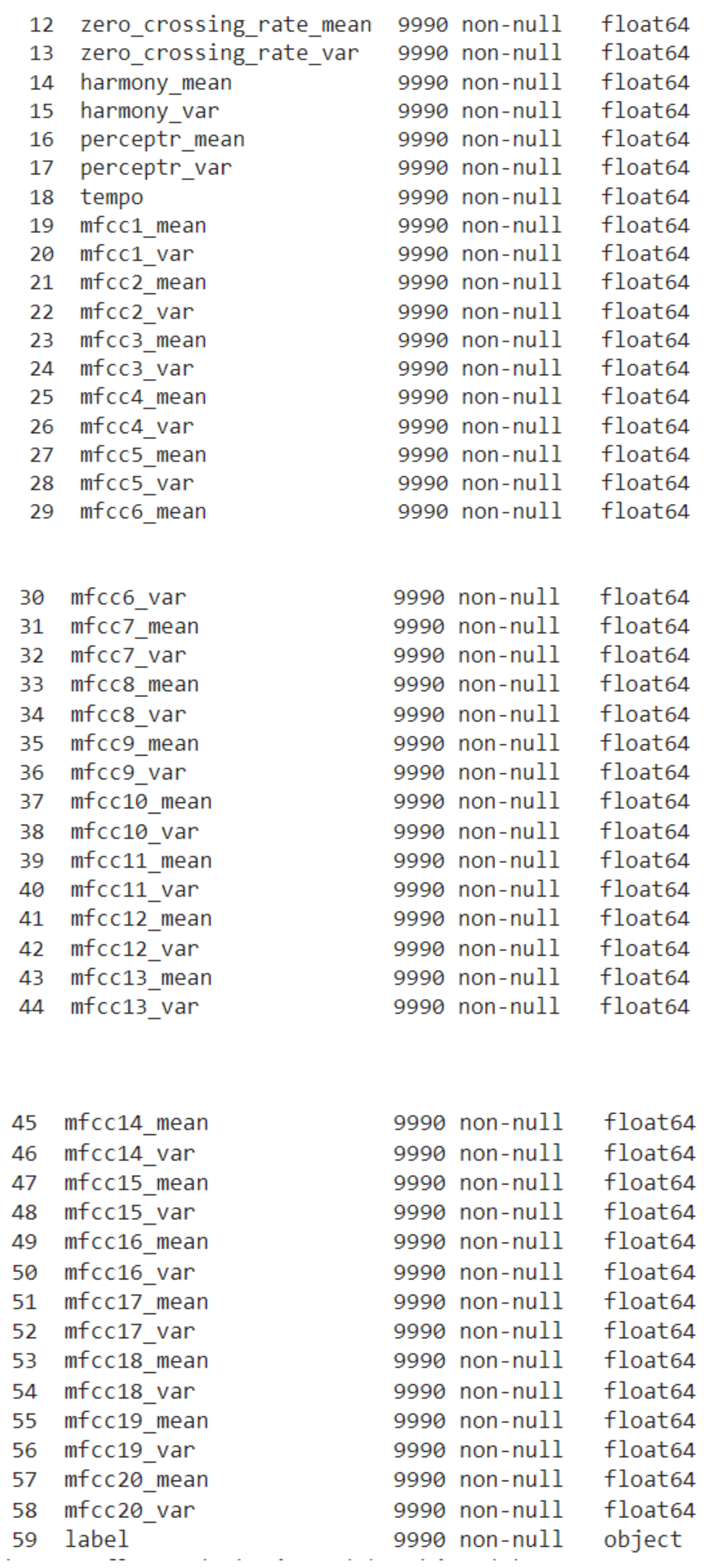
Figure 3.1 is the result produced by listing the dataset files using head() function.

3.1.1. Explore the Dataset

From Figure 3.1, the attributes in this dataset are various audio features such as Mel-frequency cepstral coefficients (MFCCs), chroma features, spectral contrast, and tempo. These features capture the time-frequency characteristics of each audio track. The class label is the music genre, which is predicted by analyzing these extracted features from each 3-second segment of the tracks. The dataset aims to classify music into one of the 10 genres, such as Blues, Classical, Country, Disco, Hip Hop, Jazz, Metal, Pop, Reggae, and Rock. The expected type of values for each feature is as follows:



```
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   filename                9990 non-null   object
 1   length                  9990 non-null   int64
 2   chroma_stft_mean        9990 non-null   float64
 3   chroma_stft_var         9990 non-null   float64
 4   rms_mean                9990 non-null   float64
 5   rms_var                 9990 non-null   float64
 6   spectral_centroid_mean  9990 non-null   float64
 7   spectral_centroid_var   9990 non-null   float64
 8   spectral_bandwidth_mean 9990 non-null   float64
 9   spectral_bandwidth_var  9990 non-null   float64
 10  rolloff_mean            9990 non-null   float64
 11  rolloff_var             9990 non-null   float64
```



```
 12  zero_crossing_rate_mean  9990 non-null   float64
 13  zero_crossing_rate_var   9990 non-null   float64
 14  harmony_mean             9990 non-null   float64
 15  harmony_var              9990 non-null   float64
 16  perceptr_mean            9990 non-null   float64
 17  perceptr_var             9990 non-null   float64
 18  tempo                    9990 non-null   float64
 19  mfcc1_mean               9990 non-null   float64
 20  mfcc1_var                9990 non-null   float64
 21  mfcc2_mean               9990 non-null   float64
 22  mfcc2_var                9990 non-null   float64
 23  mfcc3_mean               9990 non-null   float64
 24  mfcc3_var                9990 non-null   float64
 25  mfcc4_mean               9990 non-null   float64
 26  mfcc4_var                9990 non-null   float64
 27  mfcc5_mean               9990 non-null   float64
 28  mfcc5_var                9990 non-null   float64
 29  mfcc6_mean               9990 non-null   float64
```

```
 30  mfcc6_var                9990 non-null   float64
 31  mfcc7_mean               9990 non-null   float64
 32  mfcc7_var                9990 non-null   float64
 33  mfcc8_mean               9990 non-null   float64
 34  mfcc8_var                9990 non-null   float64
 35  mfcc9_mean               9990 non-null   float64
 36  mfcc9_var                9990 non-null   float64
 37  mfcc10_mean              9990 non-null   float64
 38  mfcc10_var               9990 non-null   float64
 39  mfcc11_mean              9990 non-null   float64
 40  mfcc11_var               9990 non-null   float64
 41  mfcc12_mean              9990 non-null   float64
 42  mfcc12_var               9990 non-null   float64
 43  mfcc13_mean              9990 non-null   float64
 44  mfcc13_var               9990 non-null   float64
```

```
 45  mfcc14_mean              9990 non-null   float64
 46  mfcc14_var               9990 non-null   float64
 47  mfcc15_mean              9990 non-null   float64
 48  mfcc15_var               9990 non-null   float64
 49  mfcc16_mean              9990 non-null   float64
 50  mfcc16_var               9990 non-null   float64
 51  mfcc17_mean              9990 non-null   float64
 52  mfcc17_var               9990 non-null   float64
 53  mfcc18_mean              9990 non-null   float64
 54  mfcc18_var               9990 non-null   float64
 55  mfcc19_mean              9990 non-null   float64
 56  mfcc19_var               9990 non-null   float64
 57  mfcc20_mean              9990 non-null   float64
 58  mfcc20_var               9990 non-null   float64
 59  label                    9990 non-null   object
```

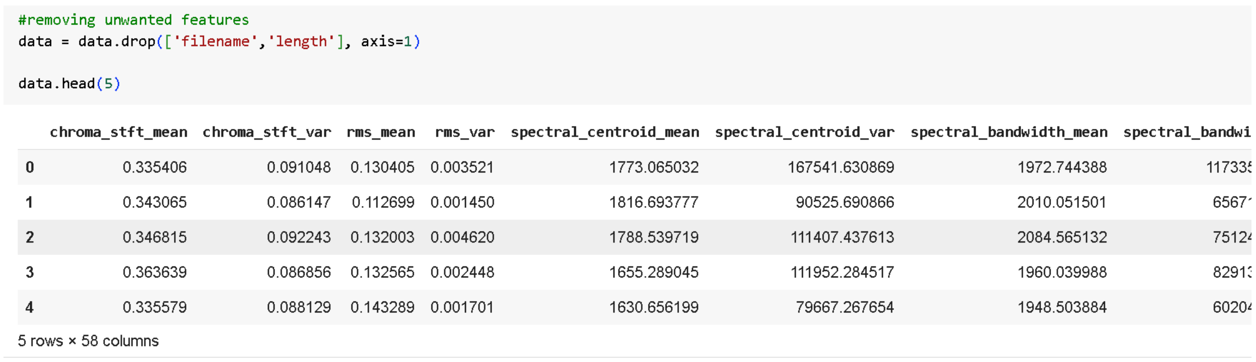
Figure 3.2 Datatypes of each attribute

3.2. DATA PREPROCESSING

3.2.1. Data Cleaning

Data Cleaning is the data pre-processing method we choose. Data cleaning routines attempt to fill in missing values, smooth out noisy data and correct inconsistencies.

Firstly, filename and length are dropped since they are not required for classification.



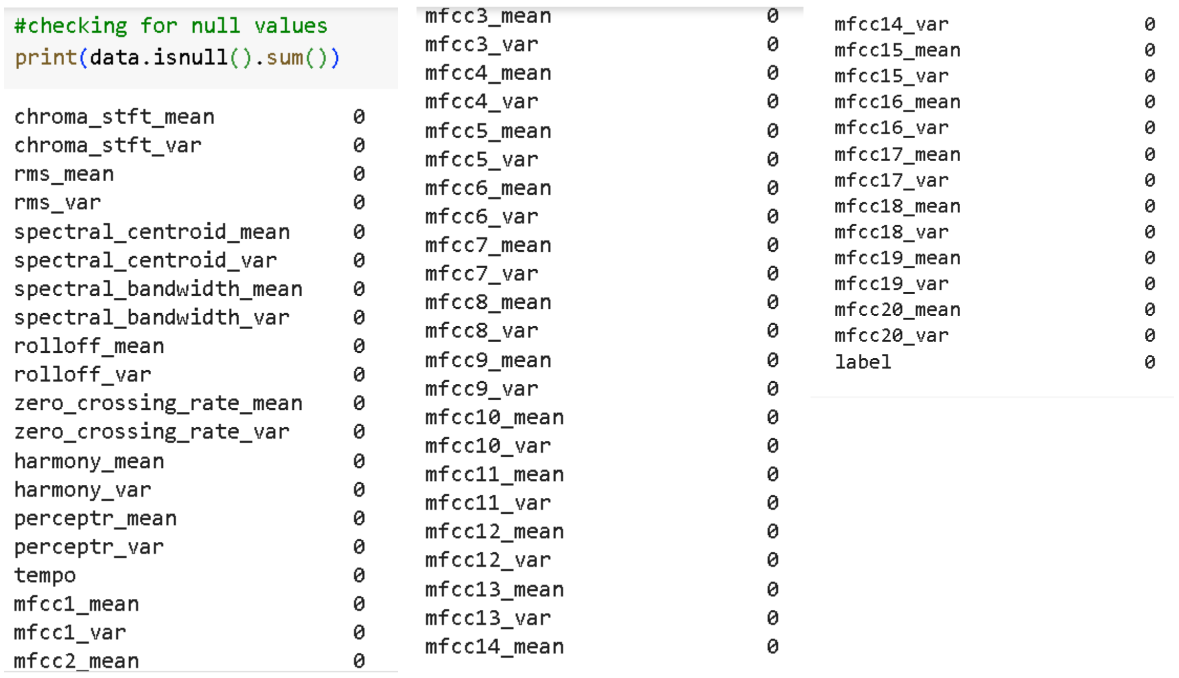
```
#removing unwanted features
data = data.drop(['filename','length'], axis=1)

data.head(5)
```

	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	spectral_bandwi
0	0.335406	0.091048	0.130405	0.003521	1773.065032	167541.630869	1972.744388	11733
1	0.343065	0.086147	0.112699	0.001450	1816.693777	90525.690866	2010.051501	6567
2	0.346815	0.092243	0.132003	0.004620	1788.539719	111407.437613	2084.565132	7512
3	0.363639	0.086856	0.132565	0.002448	1655.289045	111952.284517	1960.039988	8291
4	0.335579	0.088129	0.143289	0.001701	1630.656199	79667.267654	1948.503884	6020

5 rows × 58 columns

Figure 3.3 Dropped attributes filename and length

Handling missing values



```
#checking for null values
print(data.isnull().sum())
```

```
chroma_stft_mean           0
chroma_stft_var            0
rms_mean                   0
rms_var                    0
spectral_centroid_mean     0
spectral_centroid_var      0
spectral_bandwidth_mean    0
spectral_bandwidth_var     0
rolloff_mean               0
rolloff_var                0
zero_crossing_rate_mean    0
zero_crossing_rate_var     0
harmony_mean               0
harmony_var                0
perceptr_mean              0
perceptr_var               0
tempo                      0
mfcc1_mean                 0
mfcc1_var                  0
mfcc2_mean                 0
```

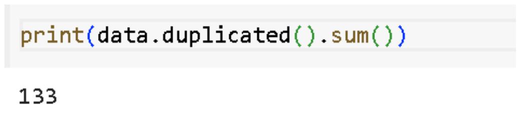
```
mfcc3_mean                 0
mfcc3_var                  0
mfcc4_mean                 0
mfcc4_var                  0
mfcc5_mean                 0
mfcc5_var                  0
mfcc6_mean                 0
mfcc6_var                  0
mfcc7_mean                 0
mfcc7_var                  0
mfcc8_mean                 0
mfcc8_var                  0
mfcc9_mean                 0
mfcc9_var                  0
mfcc10_mean                0
mfcc10_var                 0
mfcc11_mean                0
mfcc11_var                 0
mfcc12_mean                0
mfcc12_var                 0
mfcc13_mean                0
mfcc13_var                 0
mfcc14_mean                0
```

```
mfcc14_var                 0
mfcc15_mean                0
mfcc15_var                 0
mfcc16_mean                0
mfcc16_var                 0
mfcc17_mean                0
mfcc17_var                 0
mfcc18_mean                0
mfcc18_var                 0
mfcc19_mean                0
mfcc19_var                 0
mfcc20_mean                0
mfcc20_var                 0
label                      0
```

Figure 3.4 Handling missing values.

No null values were found.

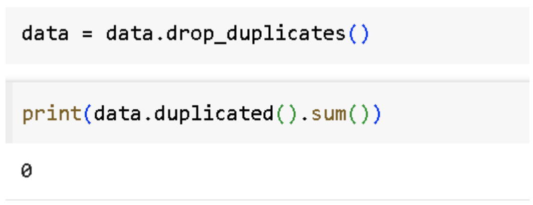
Handling Duplicate Values



```
print(data.duplicated().sum())

133
```

Figure 3.5 Checking for Duplicate Values

Figure 3.5 shows that a total of 133 duplicated records were found and is to be treated.



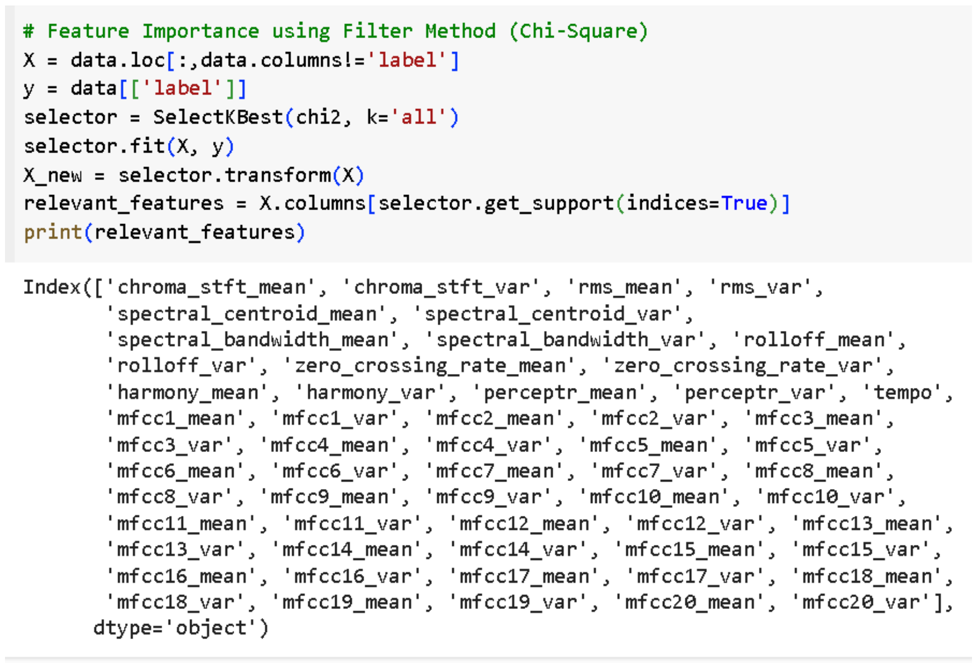
```
data = data.drop_duplicates()

print(data.duplicated().sum())

0
```

Figure 3.6 Handling Duplicate Values.

Figure 3.6 shows that all the duplicated records were dropped from the dataset.

Feature Selection

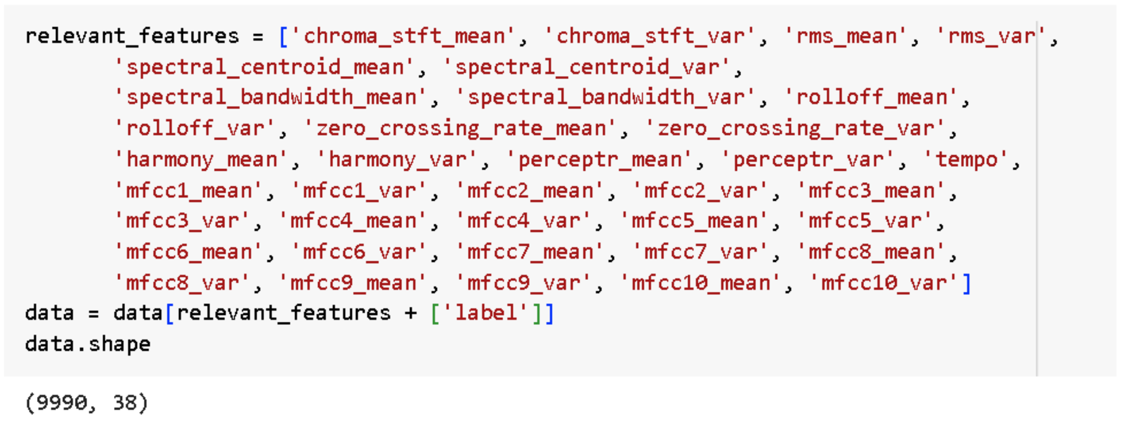
Feature selection is crucial for music genre classification project because it helps in identifying the most relevant audio features, such as MFCCs, chroma, and spectral contrast, which directly impact the accuracy of genre prediction. By selecting only the most important features using methods like the chi-square test, you reduce the noise and irrelevant data, improving both the efficiency and accuracy of the machine learning model. This also reduces computational complexity, ensuring faster model training and better performance on unseen data.



```
# Feature Importance using Filter Method (Chi-Square)
X = data.loc[:,data.columns!='label']
y = data[['label']]
selector = SelectKBest(chi2, k='all')
selector.fit(X, y)
X_new = selector.transform(X)
relevant_features = X.columns[selector.get_support(indices=True)]
print(relevant_features)

Index(['chroma_stft_mean', 'chroma_stft_var', 'rms_mean', 'rms_var',
       'spectral_centroid_mean', 'spectral_centroid_var',
       'spectral_bandwidth_mean', 'spectral_bandwidth_var', 'rolloff_mean',
       'rolloff_var', 'zero_crossing_rate_mean', 'zero_crossing_rate_var',
       'harmony_mean', 'harmony_var', 'perceptr_mean', 'perceptr_var', 'tempo',
       'mfcc1_mean', 'mfcc1_var', 'mfcc2_mean', 'mfcc2_var', 'mfcc3_mean',
       'mfcc3_var', 'mfcc4_mean', 'mfcc4_var', 'mfcc5_mean', 'mfcc5_var',
       'mfcc6_mean', 'mfcc6_var', 'mfcc7_mean', 'mfcc7_var', 'mfcc8_mean',
       'mfcc8_var', 'mfcc9_mean', 'mfcc9_var', 'mfcc10_mean', 'mfcc10_var',
       'mfcc11_mean', 'mfcc11_var', 'mfcc12_mean', 'mfcc12_var', 'mfcc13_mean',
       'mfcc13_var', 'mfcc14_mean', 'mfcc14_var', 'mfcc15_mean', 'mfcc15_var',
       'mfcc16_mean', 'mfcc16_var', 'mfcc17_mean', 'mfcc17_var', 'mfcc18_mean',
       'mfcc18_var', 'mfcc19_mean', 'mfcc19_var', 'mfcc20_mean', 'mfcc20_var'],
      dtype='object')
```

Figure 3.7 Checking Feature Importance using Chi-Square Method

Selecting the 37 most relevant features for training significantly enhances the model's performance, as it allows the system to focus on the key audio characteristics that strongly influence genre classification. By eliminating irrelevant or redundant features, we achieve the best accuracy for the task, improving both the model's efficiency and predictive capability.



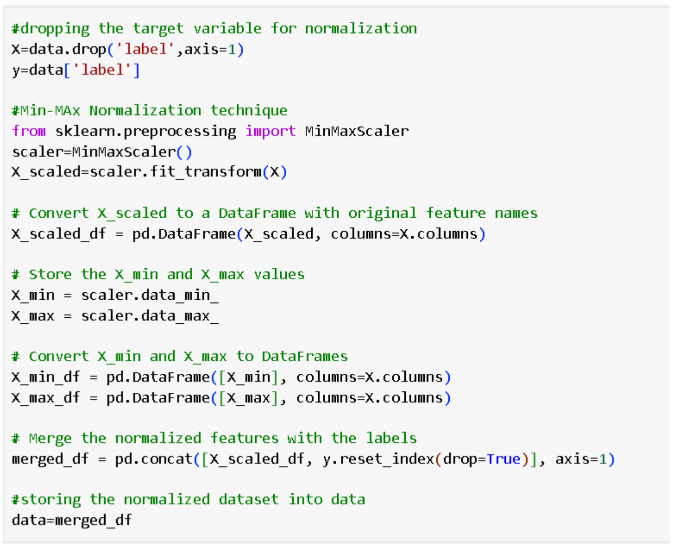
```
relevant_features = ['chroma_stft_mean', 'chroma_stft_var', 'rms_mean', 'rms_var',
       'spectral_centroid_mean', 'spectral_centroid_var',
       'spectral_bandwidth_mean', 'spectral_bandwidth_var', 'rolloff_mean',
       'rolloff_var', 'zero_crossing_rate_mean', 'zero_crossing_rate_var',
       'harmony_mean', 'harmony_var', 'perceptr_mean', 'perceptr_var', 'tempo',
       'mfcc1_mean', 'mfcc1_var', 'mfcc2_mean', 'mfcc2_var', 'mfcc3_mean',
       'mfcc3_var', 'mfcc4_mean', 'mfcc4_var', 'mfcc5_mean', 'mfcc5_var',
       'mfcc6_mean', 'mfcc6_var', 'mfcc7_mean', 'mfcc7_var', 'mfcc8_mean',
       'mfcc8_var', 'mfcc9_mean', 'mfcc9_var', 'mfcc10_mean', 'mfcc10_var']
data = data[relevant_features + ['label']]
data.shape
```

```
(9990, 38)
```

Figure 3.8 Selecting the most important 37 features

Normalizing the Dataset

Normalizing the dataset before training is crucial because it ensures that all features are on the same scale, preventing any single feature from dominating the learning process due to differences in range. In my project, I used Min-Max Scaler to normalize the data, scaling all features between 0 and 1, which helps the machine learning algorithms converge faster and improves overall accuracy by maintaining uniformity across features.



```
#dropping the target variable for normalization
X=data.drop('label',axis=1)
y=data['label']

#Min-MAx Normalization technique
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
X_scaled=scaler.fit_transform(X)

# Convert X_scaled to a DataFrame with original feature names
X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)

# Store the X_min and X_max values
X_min = scaler.data_min_
X_max = scaler.data_max_

# Convert X_min and X_max to DataFrames
X_min_df = pd.DataFrame([X_min], columns=X.columns)
X_max_df = pd.DataFrame([X_max], columns=X.columns)

# Merge the normalized features with the labels
merged_df = pd.concat([X_scaled_df, y.reset_index(drop=True)], axis=1)

#storing the normalized dataset into data
data=merged_df
```

Figure 3.9 Normalizing the Dataset

3.2.2 Analysis of Feature Variables

In this project, the classification model is built using audio features extracted from 3-second segments of the GTZAN dataset. These features are selected using the chi-square method to ensure only the most relevant variables are used to train the model, improving both accuracy and efficiency.

Audio Feature Variables:

1. chroma_stft_mean and chroma_stft_var: These features capture the short-time Fourier transform of the audio signal, which represents how energy is distributed across different frequencies. These are useful for detecting harmonic content in the music, which can help distinguish between genres with distinct harmonic structures, such as classical or rock.

2. rms_mean and rms_var: The root mean square (RMS) energy represents the power of the audio signal. Analyzing the mean and variance of RMS helps understand the overall loudness and dynamics of a track, which are important indicators for genres like jazz or metal.

3. spectral_centroid_mean and spectral_centroid_var: Spectral centroid indicates the "brightness" of a sound by measuring the center of mass of the spectrum. Genres with higher treble content, such as pop or electronic music, typically have higher spectral centroids, while classical or jazz may have lower values.

4. spectral_bandwidth_mean and spectral_bandwidth_var: Spectral bandwidth measures the range of frequencies in a sound. Wider bandwidths are often associated with more energetic genres like rock or metal, while narrower bandwidths may characterize softer, more focused sounds such as classical or folk.

5. rolloff_mean and rolloff_var: Spectral rolloff represents the frequency below which a certain percentage of the total spectral energy is contained. This feature helps in distinguishing genres based on how much of the sound energy lies in the higher frequencies, which is important for differentiating between genres like disco and classical.

6. zero_crossing_rate_mean and zero_crossing_rate_var: This feature measures the rate at which the signal crosses the zero-amplitude level, commonly used for detecting percussive or noisy elements in music. Genres with fast, rhythmic beats, like hip hop or metal, tend to have higher zero-crossing rates.

7. harmony_mean and harmony_var: These features capture the harmonic content of the track, which is crucial for genres that rely heavily on harmonic structures, such as jazz and classical.

8. perceptr_mean and perceptr_var: Perceptual sharpness or flatness measures the flatness of the power spectrum, helping to identify tonal or noise-like characteristics. This can be useful for classifying electronic and ambient music.

9. tempo: Tempo indicates the speed or pace of the music, measured in beats per minute (BPM). It is a critical feature for distinguishing between genres like dance or electronic music, which have faster tempos, and genres like blues or ballads, which are generally slower.

MFCC (Mel-Frequency Cepstral Coefficients) Features:

1. mfcc1_mean to mfcc10_mean and mfcc1_var to mfcc10_var: MFCCs capture the timbral aspects of music by analyzing the energy distribution across the mel scale, a logarithmic scale based on human perception of pitch. These coefficients are key to distinguishing between various music genres, as they represent low-level audio features that capture the texture and tone of the music. Genres like jazz and classical often have distinct MFCC patterns compared to genres like rock or hip hop.

In this project, the chi-square method was used to identify the 38 most relevant features from the pre-extracted audio data. This selection process helps reduce computational complexity and ensures that the model focuses only on the features that contribute the most to classifying music genres, improving both accuracy and efficiency.

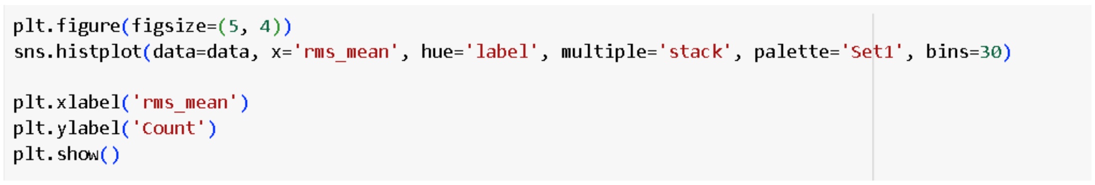
In conclusion, the analysis of these audio features, ranging from frequency-domain characteristics to time-domain features, provides a comprehensive understanding of the music tracks. These selected features ensure that the model is well-equipped to distinguish between genres, allowing for more accurate and efficient music classification.

3.2.3. Analysis of Class Variables

In this project, the primary objective is to classify music tracks into their respective genres based on pre-extracted audio features. The target variable is the music genre, which acts as the class label, and the model is trained to predict one of the 10 distinct genres such as Blues, Classical, Country, Disco, Hip Hop, Jazz, Metal, Pop, Reggae, and Rock. The classification task focuses on analysing various audio features like MFCCs, spectral contrast, and chroma, which help the machine learning models accurately distinguish between different genres. The project aims to enhance accuracy by selecting the most relevant features through the chi-square method, ensuring efficient genre prediction.

3.3. DATA VISUALIZATION

Data visualization is a crucial technique in the analysis of audio features for music genre classification, offering a graphical representation of the dataset through various plots and charts. Its importance lies in transforming complex audio feature data into visually intuitive formats, helping identify trends, patterns, and outliers across different music genres. This visual representation enhances the exploration of features like MFCCs, chroma, and spectral contrast, allowing easy comparison between genres and aiding in the discovery of insights that can improve model accuracy. Different types of visualizations, such as histograms for feature distributions, scatter plots for feature relationships, and heatmaps for correlation analysis, provide a comprehensive understanding of the data. By using tools like Matplotlib and Seaborn, data visualization aids in making data-driven decisions, ultimately enhancing the performance of the music genre classification model.



```
plt.figure(figsize=(5, 4))
sns.histplot(data=data, x='rms_mean', hue='label', multiple='stack', palette='Set1', bins=30)

plt.xlabel('rms_mean')
plt.ylabel('Count')
plt.show()
```

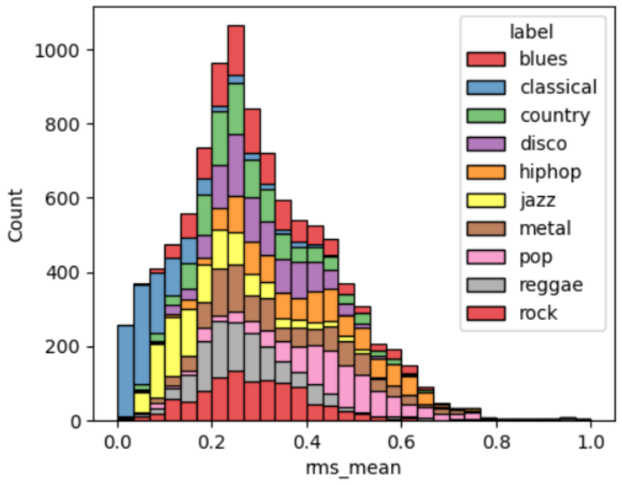
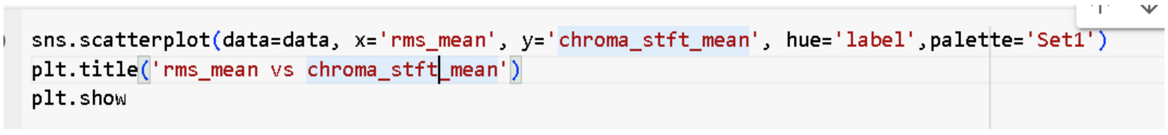


Figure 3.10 Histogram of rms_mean



```
sns.scatterplot(data=data, x='rms_mean', y='chroma_stft_mean', hue='label',palette='Set1')
plt.title('rms_mean vs chroma_stft_mean')
plt.show
```

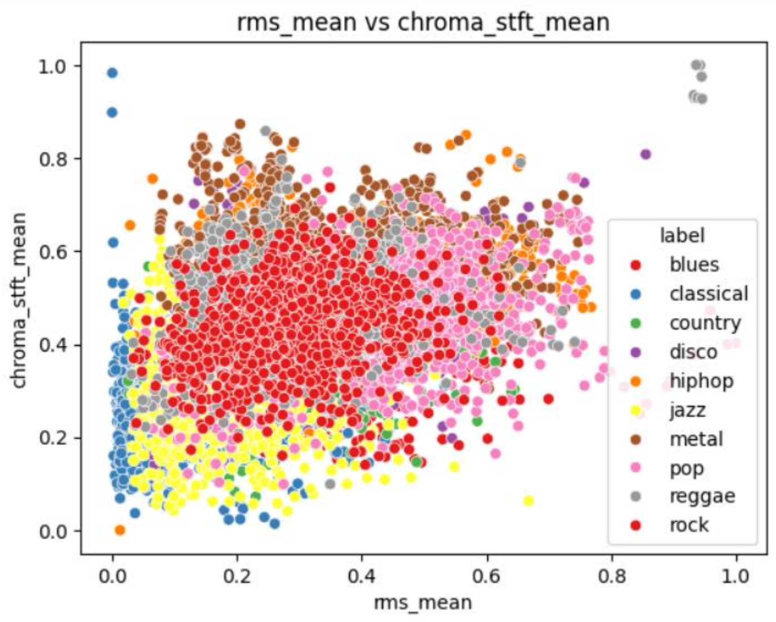
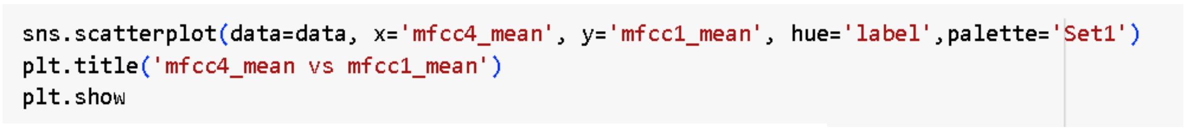


Figure 3.11 Scatter Plot of rms_mean & chroma_stft_mean



```
sns.scatterplot(data=data, x='mfcc4_mean', y='mfcc1_mean', hue='label',palette='Set1')
plt.title('mfcc4_mean vs mfcc1_mean')
plt.show
```

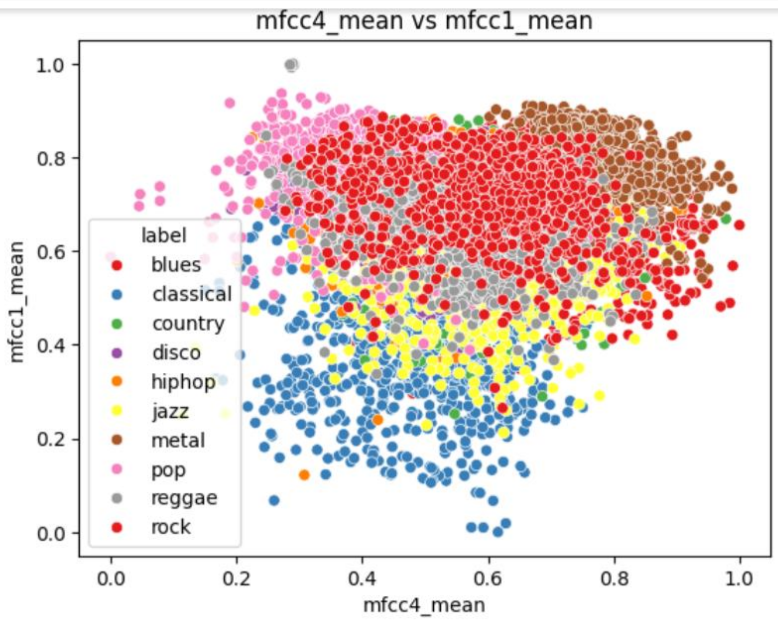


Figure 3.12 Scatter Plot of mfcc1_mean & mfcc4_mean

```
plt.figure(figsize=(5, 3))
sns.countplot(x='label', data=data, palette='Set2')

plt.title('Class Distribution of Music Genres')
plt.xlabel('Genre')
plt.ylabel('Number of Samples')
plt.xticks(rotation=45)
plt.show()
```

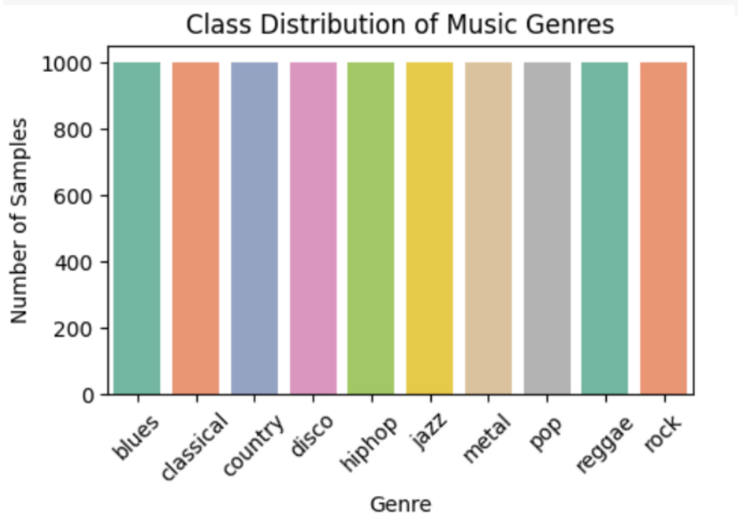


Figure 3.13 Class Balance Graph

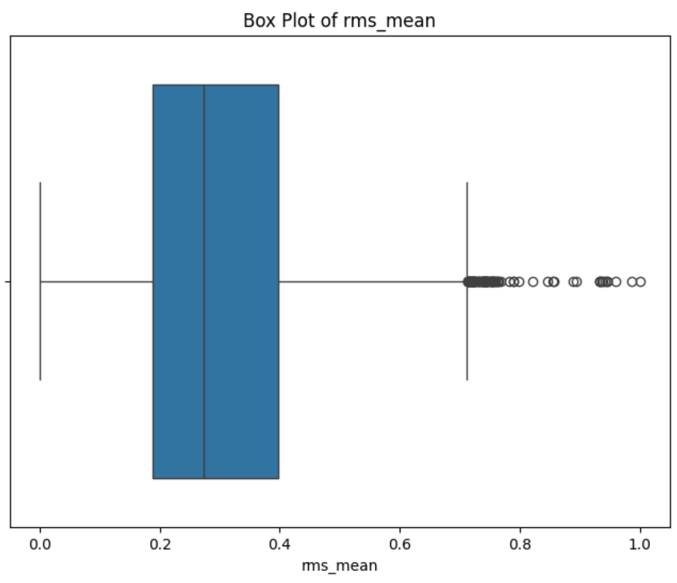


Figure 3.14 Box Plot of rms_mean

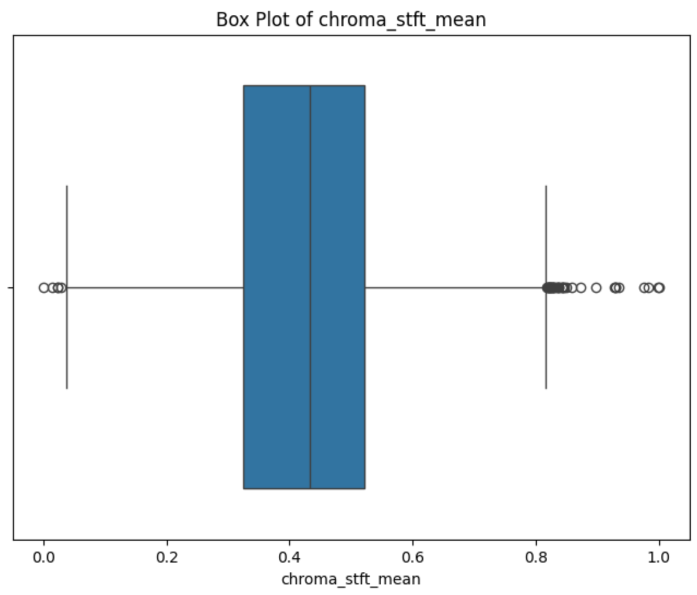


Figure 3.15 Box Plot of chroma_stft_mean

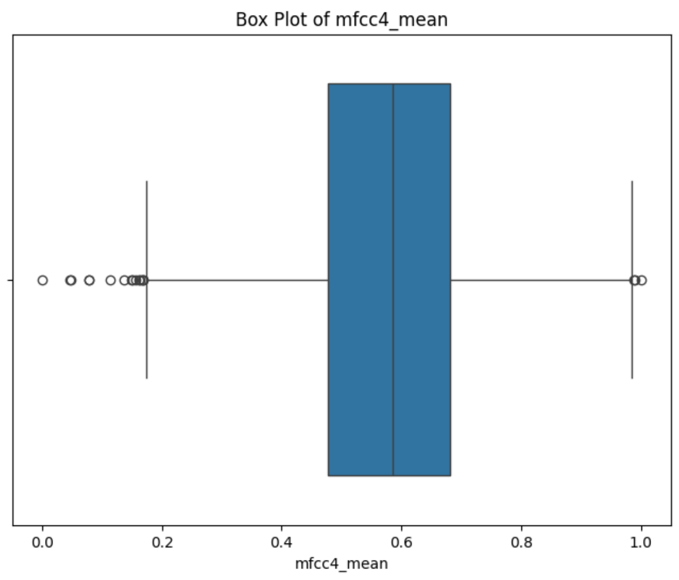


Figure 3.16 Box Plot of mfcc4_mean

3.4 ANALYSIS OF ALGORITHM

K- Nearest Neighbors (KNN)

The algorithm used in this Music Genre Classification project is KNN, with Euclidean Distance as the similarity metric.

The k-Nearest Neighbors (k-NN) algorithm is a simple yet effective machine learning algorithm used for classification tasks, such as predicting the genre of music tracks. It is an instance-based or lazy learning algorithm, meaning it makes predictions based on the similarity between the input audio feature and the features in its training dataset. The k-NN algorithm operates on the principle that similar feature patterns tend to belong to the same music genre. For this project, it looks at the k nearest audio segments in the training dataset to make predictions for new, unseen audio segments.

In the k-NN algorithm for music genre classification, each audio segment is represented as a vector in a multi-dimensional space, where each dimension corresponds to an audio feature such as MFCCs, chroma, or spectral centroid.

- Euclidean distance is used to measure the similarity between these audio feature vectors.
- When classifying new music segments, the k-NN algorithm identifies the k nearest neighbors (audio segments) to the target segment based on their Euclidean distance scores.

Pseudocode of KNN

- Input: training data X (audio features), target variable y (genres), test data X_test, K value
- Output: predicted genre for each audio segment in X_test
- Compute the distance between each observation in X_test and X
- distances = compute_distances(X, X_test)
- Find the K nearest neighbors for each observation in X_test
- nearest_neighbors = find_nearest_neighbors(distances, K)
- Compute the majority class of the nearest neighbors (i.e., the genre) for each observation in X_test
- predictions = compute_majority_class(nearest_neighbors, y)
- Return the predicted genre for each observation in X_test
- return predictions

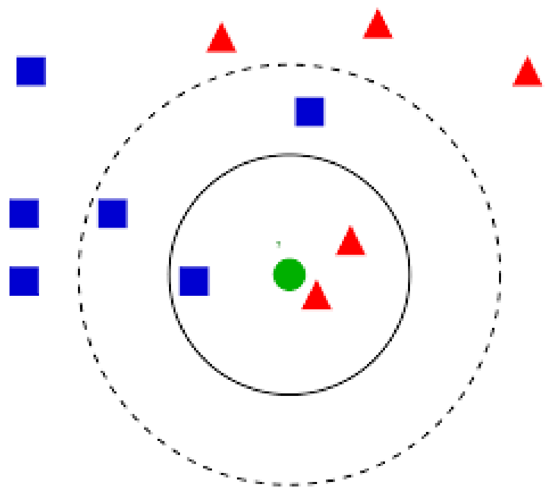


Figure 3.17 Example figure of KNN

Euclidean Distance with KNN:

- Euclidean distance is a mathematical measure used to quantify the similarity (or distance) between two vectors in a multi-dimensional space.
- It calculates the straight-line distance between two vectors and always produces a non-negative value. A value of 0 indicates that the vectors are identical, while larger values indicate greater differences between the vectors.
- The formula to find the Euclidean distance between two vectors is:

$$\mathbf{D(x, y) = \sqrt{\Sigma(x_i - y_i)^2}}$$

Where,

- xi and yi are the individual components (features) of the vectors **x** and **y** (e.g., the audio features).
- The sum of squared differences between corresponding features of the two vectors is computed, followed by taking the square root of the result.

- In the k-NN algorithm for music genre classification, each audio segment is represented as a vector in a multi-dimensional space, where the dimensions correspond to the audio features (e.g., MFCCs, chroma, and spectral centroid).
- Euclidean distance is used to measure the similarity between these audio feature vectors.
- When classifying new audio segments, the k-NN algorithm identifies the k nearest neighbors (audio segments) to the target segment based on their Euclidean distance scores

In the context of your music genre classification project, the KNN algorithm plays a crucial role in predicting the genre of an audio track based on its features. KNN operates on the principle that audio segments with similar feature patterns are likely to belong to the same genre. Here's a breakdown of how KNN works in your project:

Representation of Audio Segments:

Each audio segment is represented as a vector in a multi-dimensional space. The dimensions correspond to various audio features such as MFCCs, chroma features, spectral centroid, and more.

Euclidean Distance Calculation:

To measure the similarity between audio segments, Euclidean distance is employed. It calculates the straight-line distance between the vectors representing the audio segments. A smaller Euclidean distance implies greater similarity in audio characteristics, helping classify the segments into the appropriate genre.

Support Vector Machine (SVM)

Support Vector Machine (SVM) is a powerful classification algorithm designed to find the optimal hyperplane that best separates data points into distinct classes. In the context of this project, SVM will be applied to classify audio tracks into different music genres based on various extracted audio features such as mfcc_mean, chroma_stft_mean, and zero_crossing_rate_mean. SVM is particularly useful for datasets with high dimensionality and can handle both linear and nonlinear relationships by using kernel functions (e.g., radial basis function, polynomial). For music genre classification, SVM's strength lies in its ability to create a clear separation between genres by maximizing the margin, ensuring robust and reliable predictions. Additionally, SVM's effectiveness in minimizing overfitting, especially with noisy or limited data, makes it suitable for dealing with the complexities of audio feature datasets. By leveraging the power of SVM, this project aims to develop a model that accurately classifies audio tracks into their respective genres, offering reliable predictive capabilities for applications such as music recommendation systems and automated music tagging.

Pseudocode of SVM:

```
• Transform the input data using a kernel function (if needed)
• Find the optimal hyperplane that maximizes the margin
• Identify support vectors
• For each new input sample:
    o Calculate the decision function based on the hyperplane
    o Assign the sample to a class based on the sign of the decision function
```

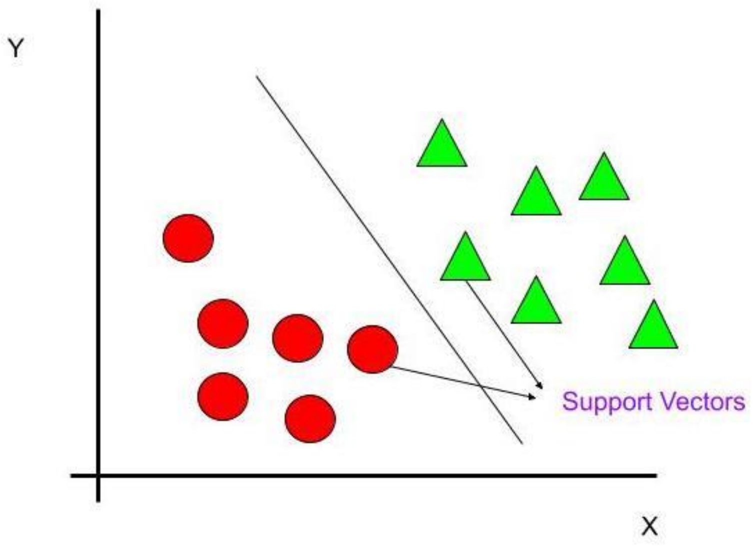


Figure 3.18 Example figure of SVM

Naïve Baye's (NB)

Naive Bayes is a simple, yet highly effective classification algorithm based on Bayes' Theorem, which calculates the probability of a data point belonging to a specific class based on the prior knowledge of the class's frequency in the dataset. In the context of this music genre classification project, Naive Bayes is used to classify songs into different genres based on the extracted audio features such as mfcc_mean, chroma_stft_mean, and spectral_centroid_mean.

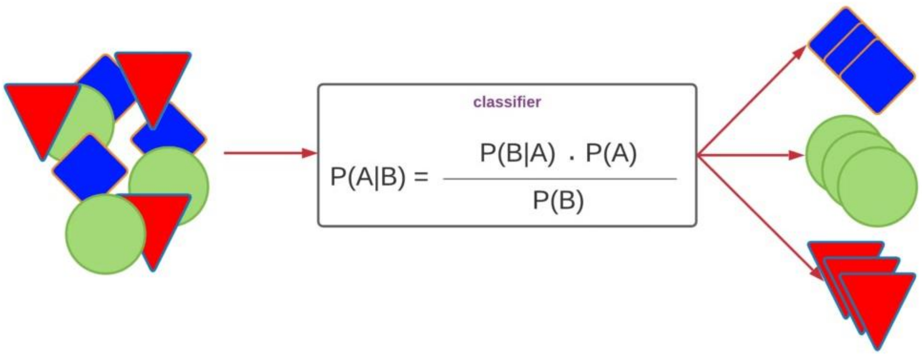


Figure 3.19 Example figure of Naïve Baye's

1. Calculate Prior Probabilities:

   The prior probability P(Ci) for each class Ci is calculated from the training data. This is simply the proportion of instances belonging to each class in the training dataset.

$$P(C_i) = \frac{\text{Number of instances in } C_i}{\text{Total number of instances}}$$

2. Calculate Likelihoods:

   For each feature xj and class Ci, calculate the likelihood P(xj | Ci). The likelihood is the probability of feature xj given class Ci.
   Depending on the type of data (categorical or continuous), different distributions are assumed:

i. For Categorical Data: Use frequency counts to estimate P(xj | Ci).

ii. For Continuous Data: Assume a Gaussian (Normal) distribution and calculate the probability density function.

For a Gaussian distribution, the likelihood is calculated as:

$$P(x_j|C_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_j-\mu)^2}{2\sigma^2}\right)$$

where μ and σ^2 are the mean and variance of the feature x_j in class C_i.

3. Apply Bayes' Theorem:

a. For a given test data point X, calculate the posterior probability for each class using Bayes' Theorem:

$$P(C_i|X) = \frac{P(X|C_i) \cdot P(C_i)}{P(X)}$$

Since $P(X)$ is constant for all classes, it can be ignored for classification:

$$P(C_i|X) \propto P(X|C_i) \cdot P(C_i)$$

Here, $P(X|C_i)$ is the product of the likelihoods of each feature:

$$P(X|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_n|C_i)$$

4. Make Predictions:

a. Assign the class Ci to the test data point X that has the highest posterior probability:

$$\text{Predicted Class} = \arg\max_{C_i} P(C_i|X)$$

Prediction: Once the probabilities are computed for each genre, the algorithm assigns the song to the genre with the highest posterior probability. Naive Bayes is particularly useful when the dataset is large or when real-time classification is needed, as it has low computational complexity.

Pseudocode of Naïve Baye's **:**

```
Input:
    - Training data (X_train, y_train)
    - Test data point (X_test)

Output:
    - Predicted class label for X_test

Steps:
1. For each class C_i in the dataset:
    a. Compute the prior probability P(C_i) from the training data:
       - P(C_i) = (Number of instances in C_i) / (Total number of instances)

    b. For each feature x_j in X_test:
        - Compute the likelihood P(x_j | C_i):
            - For categorical data, use frequency counts.
            - For continuous data, assume Gaussian distribution and calculate using the formula.

    c. Compute the posterior probability P(C_i | X):
       - P(C_i | X) ∝ P(X | C_i) * P(C_i)
       - P(X | C_i) = Product of all likelihoods P(x_j | C_i) for each feature x_j.

2. Assign the class label to X_test that has the highest posterior probability.

End
```

3.4.1 Activity Diagram

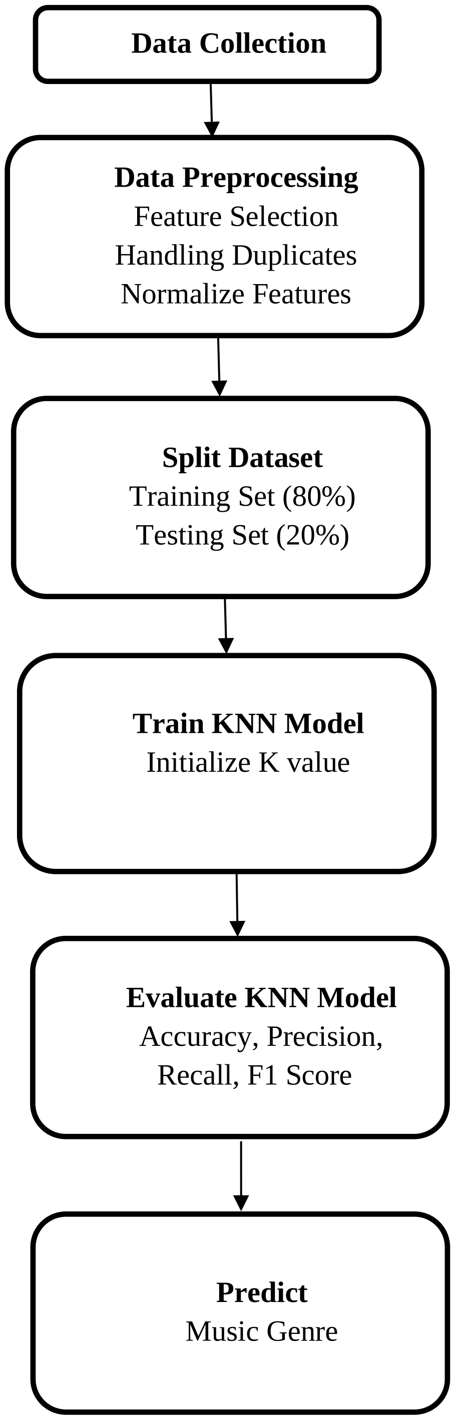


Figure 3.20 Activity Diagram of KNN

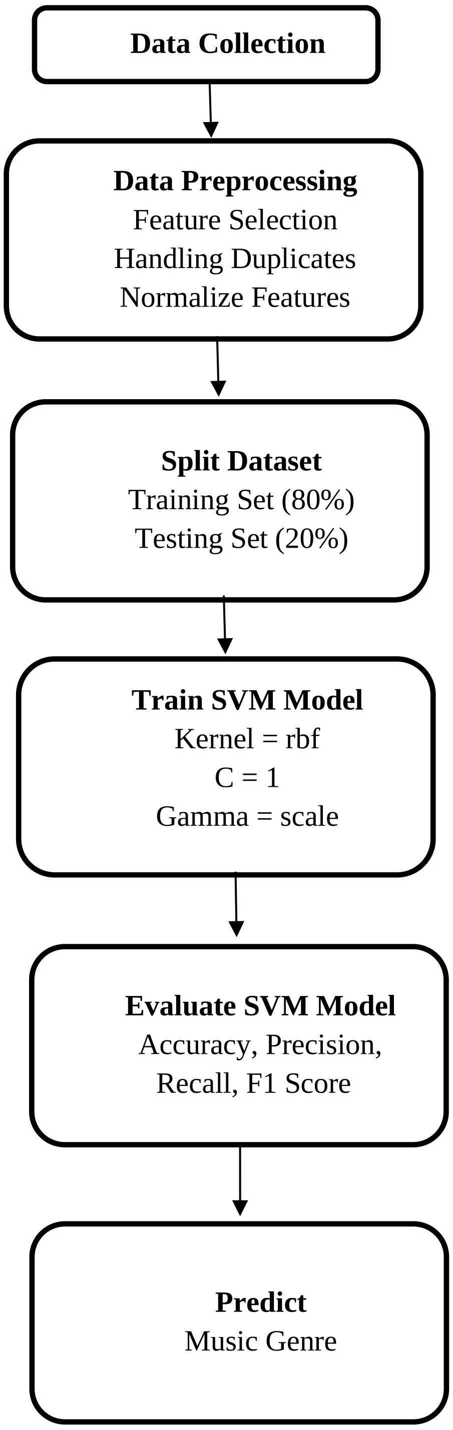


Figure 3.21 Activity Diagram of SVM

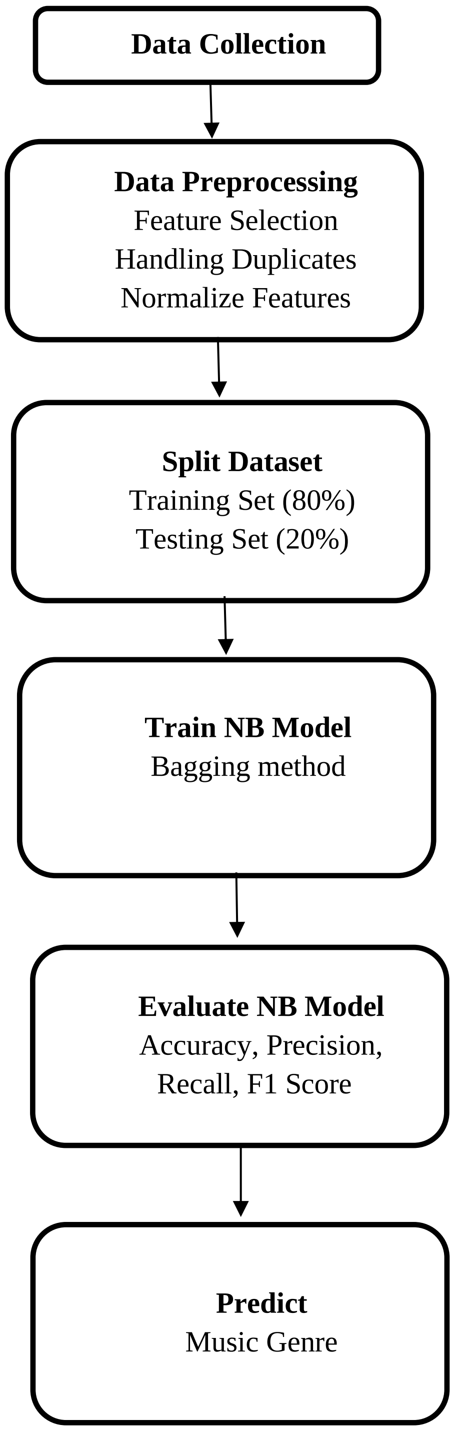


Figure 3.22 Activity Diagram of NB

3.4. PROJECT PLAN

3.4.1. Project Pipeline

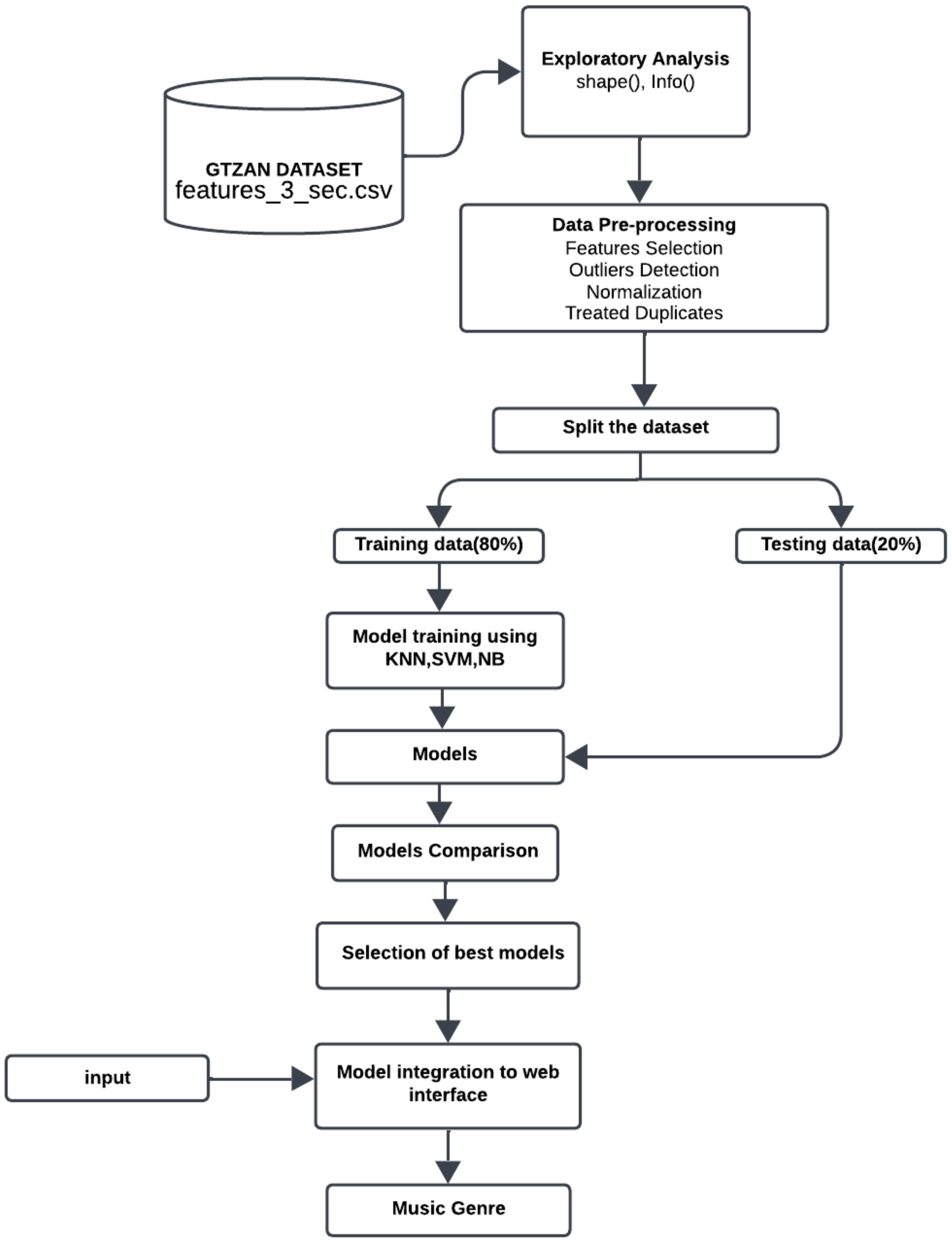


Figure 3.23 Project pipeline

3.5. FEASIBILITY ANALYSIS

A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing system or proposed system, opportunities and threats present inthe natural environment, the resources required to carry through, and ultimately the prospects for success.

Evaluated the feasibility of the system in terms of the following categories:

- Technical Feasibility
- Economic Feasibility
- Operational Feasibility

3.5.1. Technical Feasibility

Technical feasibility assesses whether the technology required for your project is readily available, and if your team has the expertise to implement it effectively.

Data Processing and Analysis:

Feasibility: The project involves extracting, processing and analyzing audio features usinglibraries like Librosa, NumPy, Pandas, and scikit-learn. These tools are widely used and well- supported, ensuring technical feasibility.

Expertise: Familiarity with these tools, as evidenced by their usage in the project, indicatestechnical competency.

Machine Learning Algorithms:

Feasibility: The use of k-Nearest Neighbors (KNN), Support Vector Machine (SVM) and Naïve Baye's (NB) is technically feasible and commonly employed in classification systems.

Expertise: The implementation of these algorithms suggests a technical understanding of machine learning concepts.

Data Visualization:

Feasibility: Matplotlib and Seaborn are used for data visualization, indicating the project'sreliance on established and technically feasible visualization tools.

Expertise: The team's ability to visualize data trends suggests proficiency in using thesetools.

File Handling:

Feasibility: Reading and writing data to CSV files using Pandas is a standard practice,ensuring technical feasibility.

Expertise: File handling is a fundamental skill, and your team's successful utilization of itimplies technical competence.

3.5.2. Economic Feasibility

Economic feasibility assesses whether the project is financially viable, consideringcosts and potential benefits.

Development Costs:

Feasibility: Open-source tools and libraries are used, reducing software acquisition costs. However, consider potential costs related to personnel, training, and hardware.

Benefits: The benefits of a well-functioning classification system, such as increased userengagement and satisfaction, may justify the development costs.

Maintenance Costs:

Feasibility: Open-source tools often have lower maintenance costs. Consider ongoing costsrelated to data updates, algorithm improvements, and support.

Benefits: The long-term benefits, such as user retention and improved recommendations,should outweigh maintenance costs.

3.5.3. Operational Feasibility

Operational feasibility evaluates whether the project aligns with operational requirements and whether it can be smoothly integrated into existing processes.

User Acceptance:

Feasibility: The project aims to enhance the user experience by providing precise Music Genre Classification.

Integration: If the system can seamlessly integrate with existing platforms, it enhances operational feasibility.

Scalability:

Feasibility: Consider scalability issues related to the growth in the number of users andMusic. Implementing collaborative filtering might face challenges with scalability.

Mitigation: Explore strategies like matrix factorization or distributed computing to address scalability concerns.

Ease of Use:

Feasibility: If the system is designed with a user-friendly interface and easy navigation, itenhances operational feasibility.

Training: Assess the ease with which users can understand and interact with the classification system.

The software environment leverages the Python programming language and key libraries for data processing, machine learning, and visualization. Development can be facilitated through Colab Notebooks. The hardware environment requires a standard computing setup with ample RAM and storage, and optional utilization of cloud services forscalability. This combination ensures a robust and flexible system environment for the development, testing, and potential deployment of the classification system.

3.5 SYSTEM ENVIROMENT

3.5.1 Software Environment

The software environment encompasses the tools, frameworks, and platforms utilizedin the development and execution of the recommendation system.

Programming Languages:

Python: The primary programming language employed for its versatility, extensive libraries (NumPy, Pandas, scikit-learn), and strong support in data science and machine learning.

Data Processing and Analysis:

NumPy and Pandas: Utilized for efficient data manipulation, handling, and analysis.

scikit-learn: Employed for implementing machine learning algorithms.

Machine Learning Libraries:

scikit-learn: Used for implementing and training machine learning models.
SciPy: Complements NumPy and provides additional functionality for scientificcomputing.

Data Visualization:

Matplotlib and Seaborn: Chosen for creating insightful visualizations, aiding in theexploration and communication of data patterns.

File Handling:

Pandas: Applied for reading and writing data to CSV files, ensuring seamless datamanagement.

Extracting Audio Features:

Librosa: Applied for extracting features from audio files.

Development Environment:

Google Colab

Colab is a free Jupyter notebook environment that runs entirely in the cloud. We can writeand execute code in Python. Colab supports many machine learning libraries which can be easily loaded in the colab notebook.

Visual Studio Code

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tool a developerneeds for a quick code-build-debug cycle and leaves more complex work flows to fuller featured IDEs, such as Visual Studio IDE

HTML and CSS

Hyper Text Markup Language is used for creating web pages.HTML describes the structureof the web page. Here, the user interface of my project is done using HTML. Cascading Style Sheet is used with HTML to style the web pages.

Flask

Flask is a web framework, it’s a Python module that lets you develop web applications easily. It has a small and easy-to-extend core: it’s a microframework that doesn’t include an ORM (Object Relational Manager) or such features. It does have many cool features likeURL routing, template engine. It is a WSGI web app framework.

GitHub

Git is an open-source version control system that was started by Linus Torvalds. Git is similar to other version control systems Subversion, CVS, and Mercurial to name a few. Version control systems keep these revisions straight, storing the modifications in a centralrepository. This allows developers to easily collaborate, as they can download a new versionof the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute. Git is the preferred version control system of most developers, since it has multiple advantages over the other systems available.It stores file changes more efficiently and ensures file integrity better.

The social networking aspect of GitHub is probably its most powerful feature, allowing projects to grow more than just about any of the other features offered. Project revisions canbe discussed publicly, so a mass of experts can contribute knowledge and collaborate to advance a project forward

3.5.2 Hardware Environment

The hardware environment refers to the physical infrastructure necessary to support the development and deployment of the recommendation system.

Computing Resources:

Processor: 2 GHz, A powerful multi-core processor is recommended to handle the computational demands efficiently. Consider a processor with at least quad- core architecture for optimal performance.

Storage:

Sufficient Disk Space: 512 GB SSDMemory (RAM):
8 GB RAM: Sufficient RAM is crucial for efficiently handling large datasets and performing machine learning tasks.

Internet Connectivity:

High-Speed Internet: A stable and high-speed internet connection is essential foraccessing online resources, libraries, and datasets during development.

4. SYSTEM DESIGN

4.1. MODEL BUILDING

4.1.1. Model Planning

The primary aim of the model planning phase in the Music Genre Classification system is to establish a comprehensive framework that guides subsequent model-building activities. This phase involves defining the scope, selecting appropriate algorithms, and outlining the overall strategy for generating accurate and meaningful Music Genre Classification tailored to specific audio files.

1. Objective:

Build a Music Genre Classification using collaborative filtering techniques toprovide accurate and meaningful classification to users based on the features of audio file.

2. Approach:

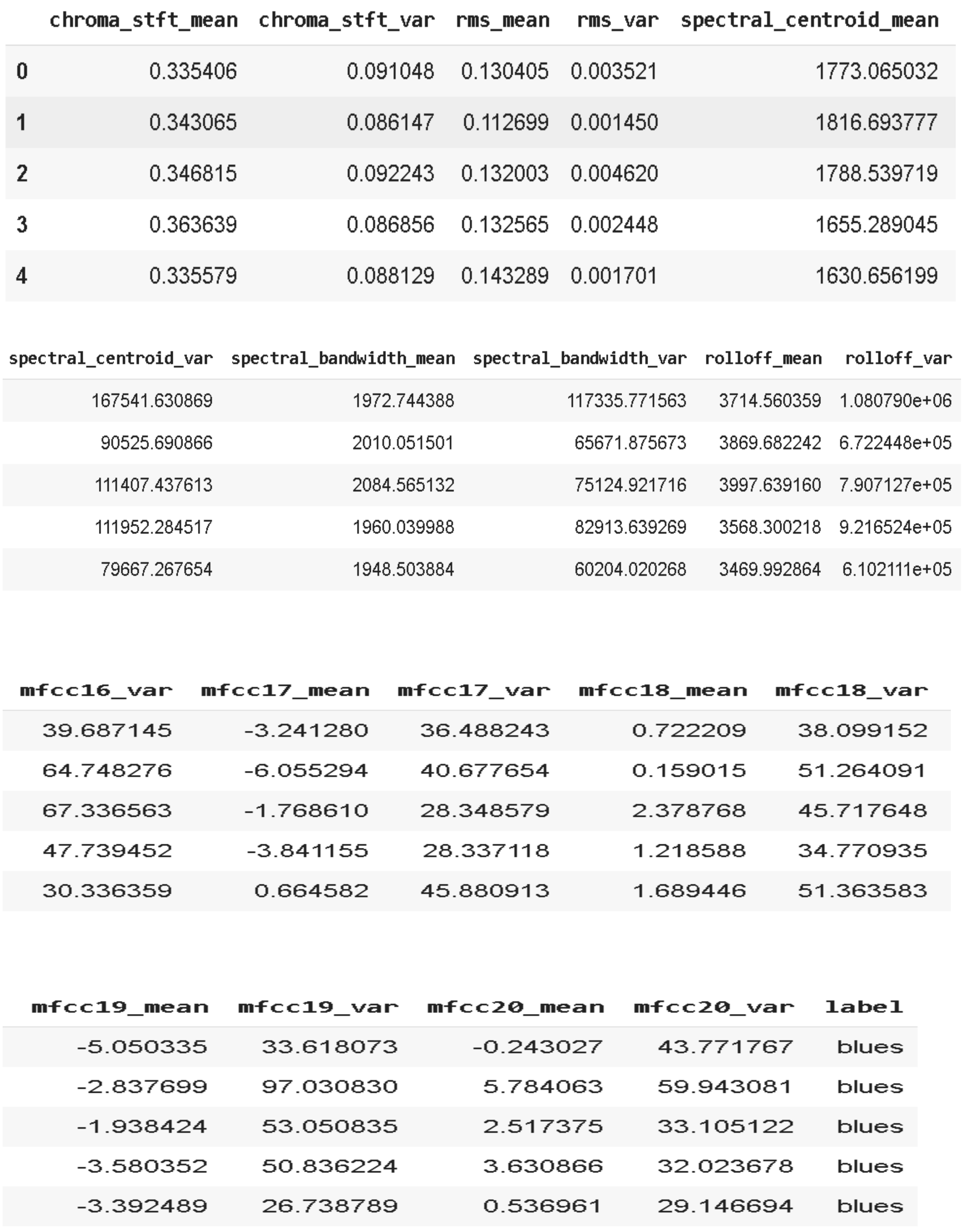
Employees K- Nearest Neighbor (KNN), Support Vector Machine (SVM), and Naïve Baye's (NB) algorithms which algorithms is the best suitable for Music Genre Classification with high accuracy

3. Data Preparation:

Import and preprocess GTZAN features_30_sec.csv Dataset. Handle duplicates and Outliers, Feature Selection and Normalization.

Exploratory Data Analysis (EDA):

Understand the distribution of various audio features. Analyze music genre counts and distribution.



	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean
0	0.335406	0.091048	0.130405	0.003521	1773.065032
1	0.343065	0.086147	0.112699	0.001450	1816.693777
2	0.346815	0.092243	0.132003	0.004620	1788.539719
3	0.363639	0.086856	0.132565	0.002448	1655.289045
4	0.335579	0.088129	0.143289	0.001701	1630.656199

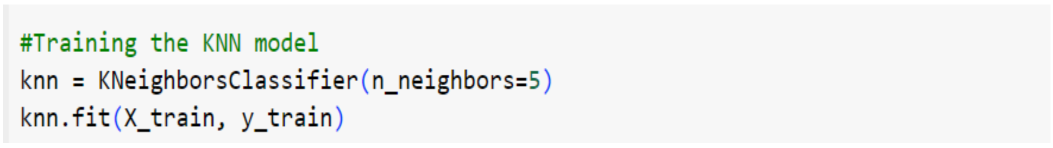
spectral_centroid_var	spectral_bandwidth_mean	spectral_bandwidth_var	rolloff_mean	rolloff_var
167541.630869	1972.744388	117335.771563	3714.560359	1.080790e+06
90525.690866	2010.051501	65671.875673	3869.682242	6.722448e+05
111407.437613	2084.565132	75124.921716	3997.639160	7.907127e+05
111952.284517	1960.039988	82913.639269	3568.300218	9.216524e+05
79667.267654	1948.503884	60204.020268	3469.992864	6.102111e+05

mfcc16_var	mfcc17_mean	mfcc17_var	mfcc18_mean	mfcc18_var
39.687145	-3.241280	36.488243	0.722209	38.099152
64.748276	-6.055294	40.677654	0.159015	51.264091
67.336563	-1.768610	28.348579	2.378768	45.717648
47.739452	-3.841155	28.337118	1.218588	34.770935
30.336359	0.664582	45.880913	1.689446	51.363583

mfcc19_mean	mfcc19_var	mfcc20_mean	mfcc20_var	label
-5.050335	33.618073	-0.243027	43.771767	blues
-2.837699	97.030830	5.784063	59.943081	blues
-1.938424	53.050835	2.517375	33.105122	blues
-3.580352	50.836224	3.630866	32.023678	blues
-3.392489	26.738789	0.536961	29.146694	blues

Figure 4.1 Snapshots of training data

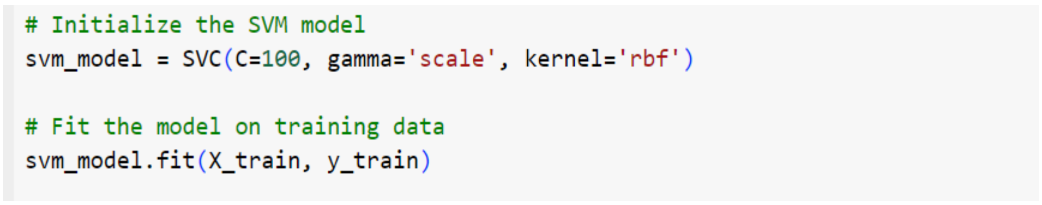
Implemented the K-Nearest Neighbors (KNN) algorithm to provide accurate genre predictions based on user interactions and audio features. After thorough experimentation and evaluation, we selected k=5 for the KNN model due to its high accuracy in classifying music genres.



```
#Training the KNN model
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
```

Figure 4.2 Fitted KNN Model

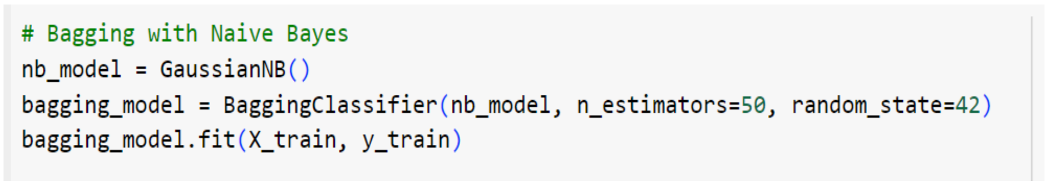
Implemented the Support Vector Machine (SVM) algorithm to achieve accurate genre predictions based on user interactions and extracted audio features. We optimized the model using hyperparameters, specifically the Radial Basis Function (RBF) kernel, C=1, and gamma set to scale. These parameters were determined through the GridSearch method, which systematically explored various combinations to identify the optimal configuration, resulting in high accuracy. The SVM model was trained and validated using cross-validation techniques, ultimately demonstrating exceptional performance in predicting music genres and enhancing the overall music discovery experience for users.



```
# Initialize the SVM model
svm_model = SVC(C=100, gamma='scale', kernel='rbf')

# Fit the model on training data
svm_model.fit(X_train, y_train)
```

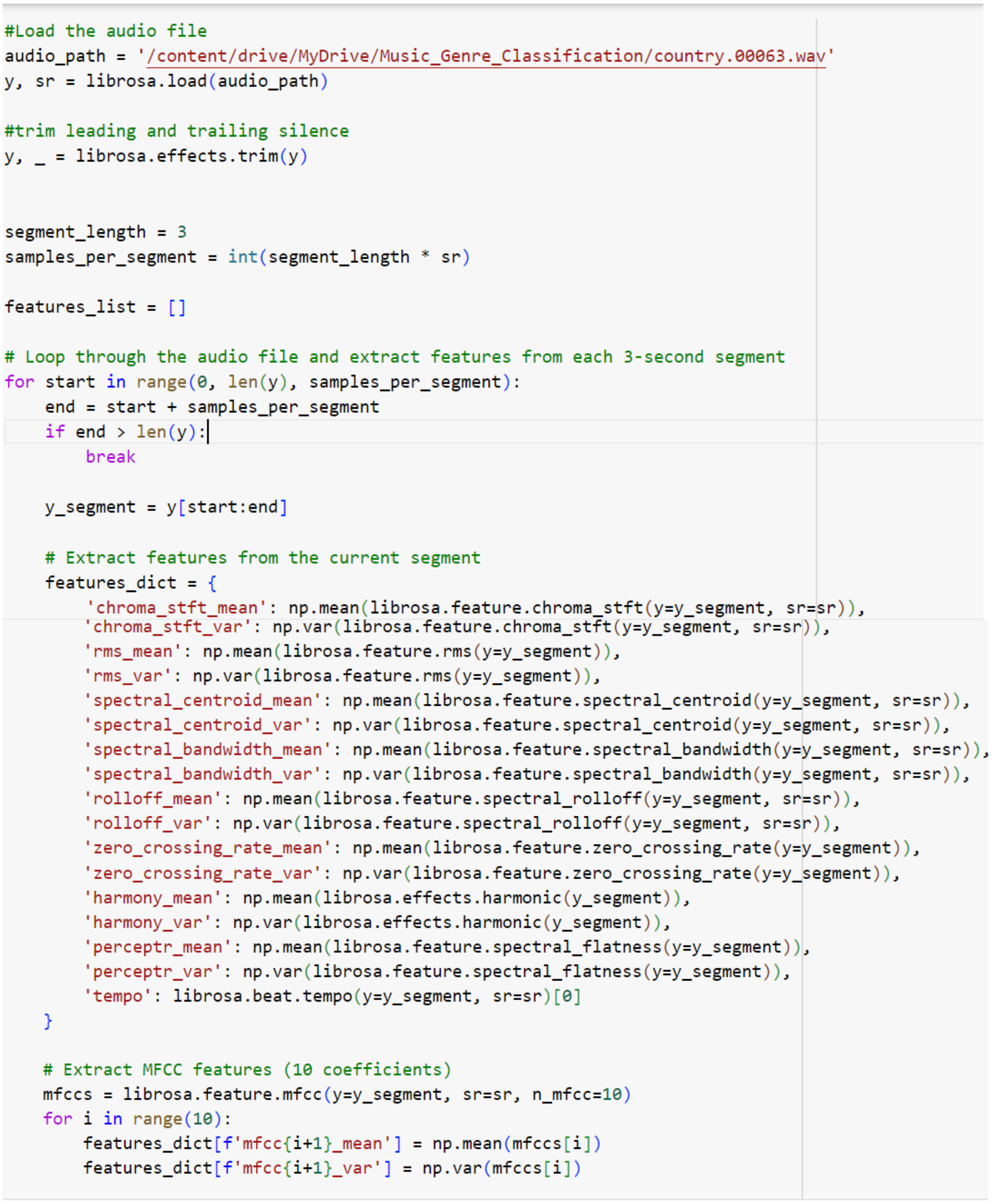
Figure 4.3 Fitted SVM Model

Implemented the Naive Bayes algorithm to provide effective genre predictions based on user interactions and extracted audio features. To enhance the model's accuracy, we employed the Bagging method, which combines multiple Naive Bayes classifiers to reduce variance and improve overall performance. The Bagging technique allowed us to create diverse subsets of the training data, training separate models on each subset and aggregating their predictions for a more robust classification outcome. This ensemble approach significantly improved the accuracy of our Naive Bayes model, demonstrating its effectiveness in predicting music genres and contributing to a more engaging music discovery experience for users.



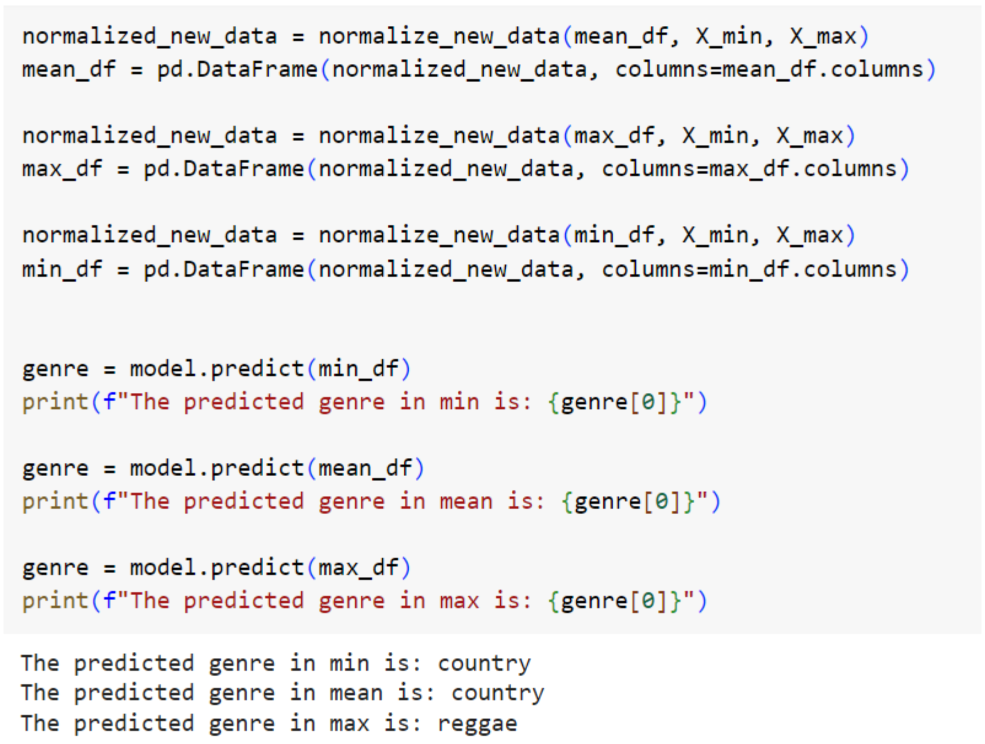
```
# Bagging with Naive Bayes
nb_model = GaussianNB()
bagging_model = BaggingClassifier(nb_model, n_estimators=50, random_state=42)
bagging_model.fit(X_train, y_train)
```

Figure 4.4 Fitted NB Model



```python
#Load the audio file
audio_path = '/content/drive/MyDrive/Music_Genre_Classification/country.00063.wav'
y, sr = librosa.load(audio_path)

#trim leading and trailing silence
y, _ = librosa.effects.trim(y)


segment_length = 3
samples_per_segment = int(segment_length * sr)

features_list = []

# Loop through the audio file and extract features from each 3-second segment
for start in range(0, len(y), samples_per_segment):
    end = start + samples_per_segment
    if end > len(y):
        break

    y_segment = y[start:end]

    # Extract features from the current segment
    features_dict = {
        'chroma_stft_mean': np.mean(librosa.feature.chroma_stft(y=y_segment, sr=sr)),
        'chroma_stft_var': np.var(librosa.feature.chroma_stft(y=y_segment, sr=sr)),
        'rms_mean': np.mean(librosa.feature.rms(y=y_segment)),
        'rms_var': np.var(librosa.feature.rms(y=y_segment)),
        'spectral_centroid_mean': np.mean(librosa.feature.spectral_centroid(y=y_segment, sr=sr)),
        'spectral_centroid_var': np.var(librosa.feature.spectral_centroid(y=y_segment, sr=sr)),
        'spectral_bandwidth_mean': np.mean(librosa.feature.spectral_bandwidth(y=y_segment, sr=sr)),
        'spectral_bandwidth_var': np.var(librosa.feature.spectral_bandwidth(y=y_segment, sr=sr)),
        'rolloff_mean': np.mean(librosa.feature.spectral_rolloff(y=y_segment, sr=sr)),
        'rolloff_var': np.var(librosa.feature.spectral_rolloff(y=y_segment, sr=sr)),
        'zero_crossing_rate_mean': np.mean(librosa.feature.zero_crossing_rate(y=y_segment)),
        'zero_crossing_rate_var': np.var(librosa.feature.zero_crossing_rate(y=y_segment)),
        'harmony_mean': np.mean(librosa.effects.harmonic(y_segment)),
        'harmony_var': np.var(librosa.effects.harmonic(y_segment)),
        'perceptr_mean': np.mean(librosa.feature.spectral_flatness(y=y_segment)),
        'perceptr_var': np.var(librosa.feature.spectral_flatness(y=y_segment)),
        'tempo': librosa.beat.tempo(y=y_segment, sr=sr)[0]
    }

    # Extract MFCC features (10 coefficients)
    mfccs = librosa.feature.mfcc(y=y_segment, sr=sr, n_mfcc=10)
    for i in range(10):
        features_dict[f'mfcc{i+1}_mean'] = np.mean(mfccs[i])
        features_dict[f'mfcc{i+1}_var'] = np.var(mfccs[i])
```

Figure 4.5 Code Snippet for extraction of audio features



```
normalized_new_data = normalize_new_data(mean_df, X_min, X_max)
mean_df = pd.DataFrame(normalized_new_data, columns=mean_df.columns)

normalized_new_data = normalize_new_data(max_df, X_min, X_max)
max_df = pd.DataFrame(normalized_new_data, columns=max_df.columns)

normalized_new_data = normalize_new_data(min_df, X_min, X_max)
min_df = pd.DataFrame(normalized_new_data, columns=min_df.columns)


genre = model.predict(min_df)
print(f"The predicted genre in min is: {genre[0]}")

genre = model.predict(mean_df)
print(f"The predicted genre in mean is: {genre[0]}")

genre = model.predict(max_df)
print(f"The predicted genre in max is: {genre[0]}")
```

```
The predicted genre in min is: country
The predicted genre in mean is: country
The predicted genre in max is: reggae
```

Figure 4.6 Code Snippet for feature based Genre Classification

4.1.2 Training

The training phase involves the construction of the classification model using the Nearest Neighbors algorithm (KNN), Support Vector Machine (SVM), Naïve Bayes (NB). The dataset is utilized without a distinct split into training andtesting sets.

The KNN algorithm is then employed to compute distances or similarities between audio features, providing accurate Classification of Genre.

The SVM algorithm is then employed to create a hyperplane that separates different music genres in the feature space, effectively capturing complex relationships among audio features and enabling accurate classification of genres. By leveraging the Radial Basis Function (RBF) kernel, the SVM model can handle non-linear data, ensuring that the genre predictions are both precise and reliable.

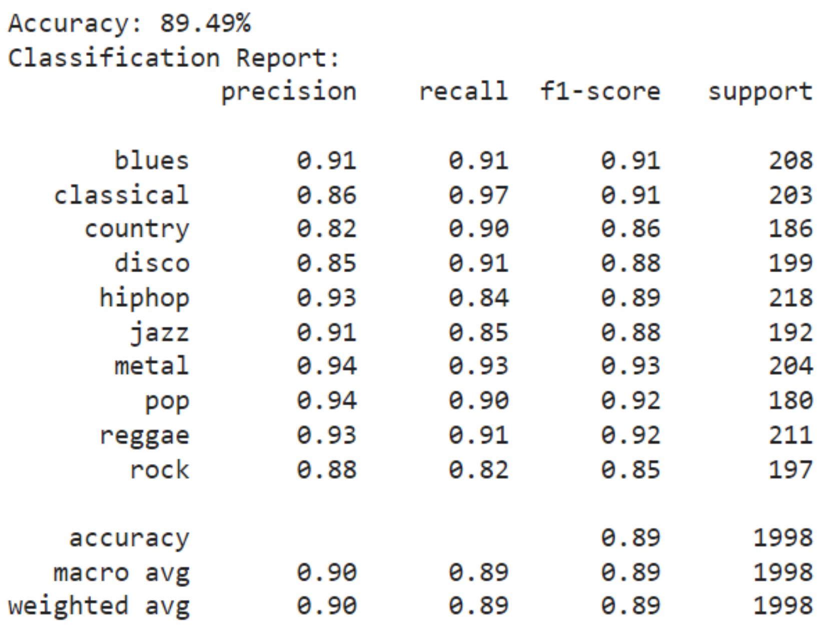
The Naive Bayes algorithm is then utilized to compute the probability distributions of audio features for each music genre, enabling effective classification based on user interactions and extracted data. By applying the principles of Bayes' theorem, the model evaluates the likelihood of a track belonging to a specific genre, considering the independence of features. This probabilistic approach allows the Naive Bayes model to deliver accurate genre predictions efficiently, even in cases with limited training data.

4.1.3 Testing

The testing phase of our music genre classification project evaluates the model's performance and its ability to generalize to unseen audio data. Model evaluation is conducted using a separate test dataset, assessing accuracy, precision, recall, and F1-score, among other relevant metrics. Although the specific code for hyperparameter tuning and detailed error analysis is not explicitly provided, the iterative refinement process involves adjusting the KNN, SVM, and Naive Bayes models based on testing results to enhance their performance. The validation step ensures that each model aligns with the project's objectives, providing meaningful and accurate Genre Classification.

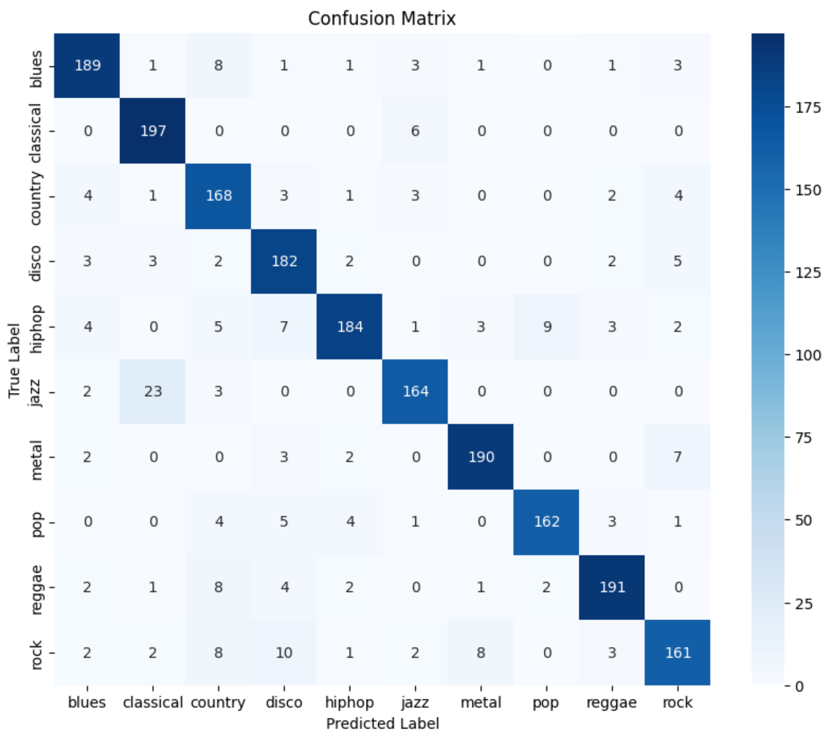
5. RESULTS AND DISCUSSION

KNN Model



```
Accuracy: 89.49%
Classification Report:
              precision    recall  f1-score   support

       blues       0.91      0.91      0.91       208
   classical       0.86      0.97      0.91       203
     country       0.82      0.90      0.86       186
       disco       0.85      0.91      0.88       199
      hiphop       0.93      0.84      0.89       218
        jazz       0.91      0.85      0.88       192
       metal       0.94      0.93      0.93       204
         pop       0.94      0.90      0.92       180
      reggae       0.93      0.91      0.92       211
        rock       0.88      0.82      0.85       197

    accuracy                           0.89      1998
   macro avg       0.90      0.89      0.89      1998
weighted avg       0.90      0.89      0.89      1998
```

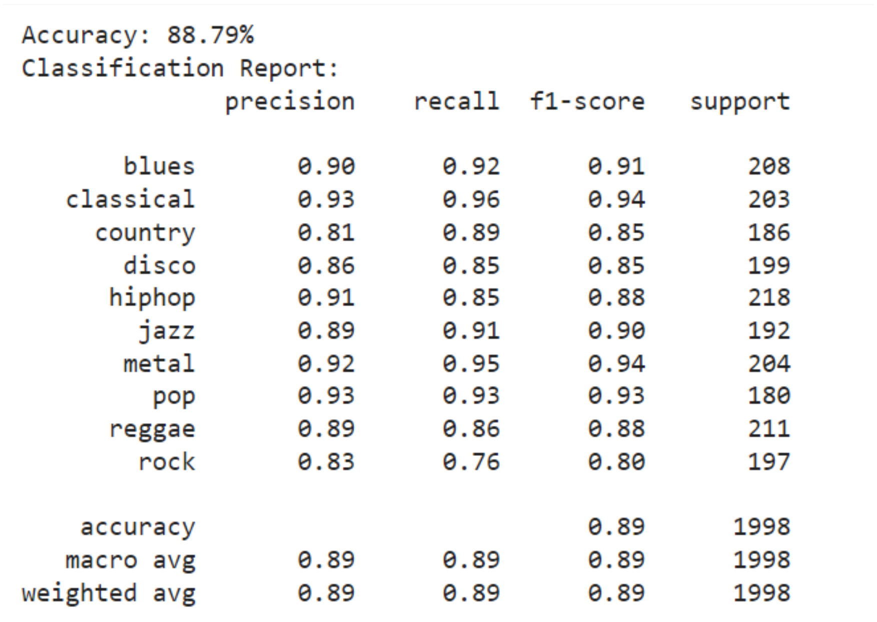
Figure 5.1 Evaluation of KNN Model



	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	189	1	8	1	1	3	1	0	1	3
classical	0	197	0	0	0	6	0	0	0	0
country	4	1	168	3	1	3	0	0	2	4
disco	3	3	2	182	2	0	0	0	2	5
hiphop	4	0	5	7	184	1	3	9	3	2
jazz	2	23	3	0	0	164	0	0	0	0
metal	2	0	0	3	2	0	190	0	0	7
pop	0	0	4	5	4	1	0	162	3	1
reggae	2	1	8	4	2	0	1	2	191	0
rock	2	2	8	10	1	2	8	0	3	161

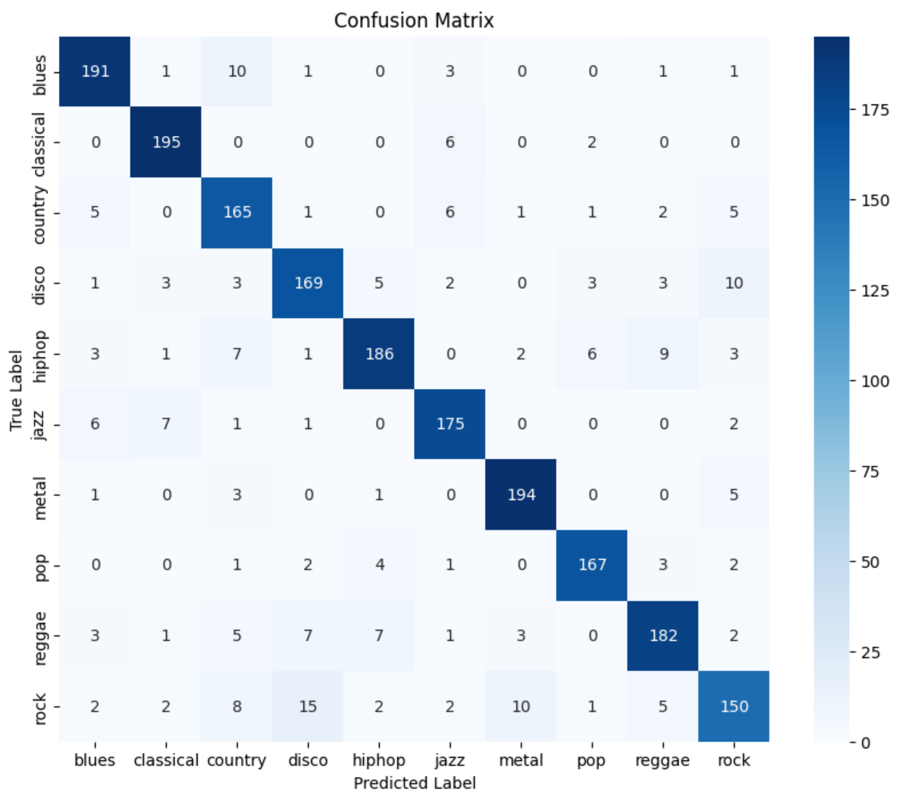
Figure 5.2 Confusion Matrix of KNN Model

SVM Model



```
Accuracy: 88.79%
Classification Report:
              precision    recall  f1-score   support

       blues       0.90      0.92      0.91       208
   classical       0.93      0.96      0.94       203
     country       0.81      0.89      0.85       186
       disco       0.86      0.85      0.85       199
      hiphop       0.91      0.85      0.88       218
        jazz       0.89      0.91      0.90       192
       metal       0.92      0.95      0.94       204
         pop       0.93      0.93      0.93       180
      reggae       0.89      0.86      0.88       211
        rock       0.83      0.76      0.80       197

    accuracy                           0.89      1998
   macro avg       0.89      0.89      0.89      1998
weighted avg       0.89      0.89      0.89      1998
```

Figure 5.3 Evaluation of SVM Model

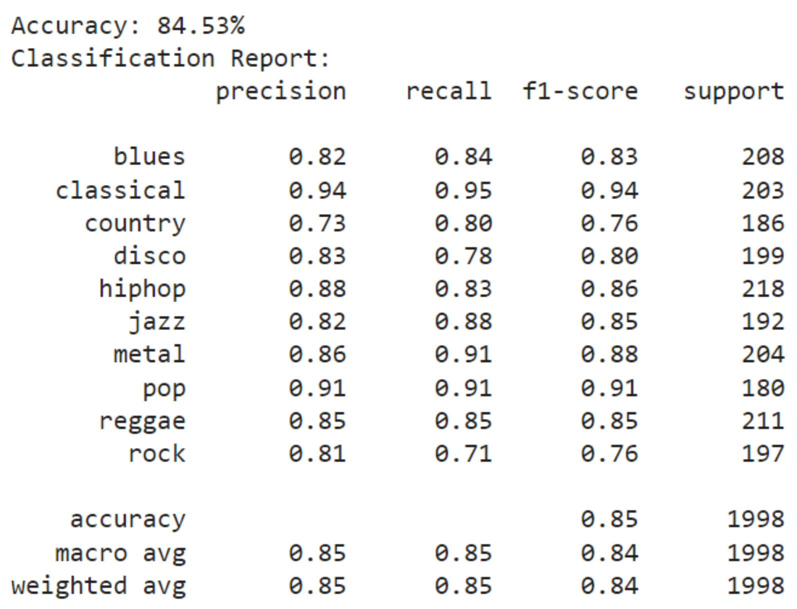


Confusion Matrix

True Label \ Predicted Label	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	191	1	10	1	0	3	0	0	1	1
classical	0	195	0	0	0	6	0	2	0	0
country	5	0	165	1	0	6	1	1	2	5
disco	1	3	3	169	5	2	0	3	3	10
hiphop	3	1	7	1	186	0	2	6	9	3
jazz	6	7	1	1	0	175	0	0	0	2
metal	1	0	3	0	1	0	194	0	0	5
pop	0	0	1	2	4	1	0	167	3	2
reggae	3	1	5	7	7	1	3	0	182	2
rock	2	2	8	15	2	2	10	1	5	150

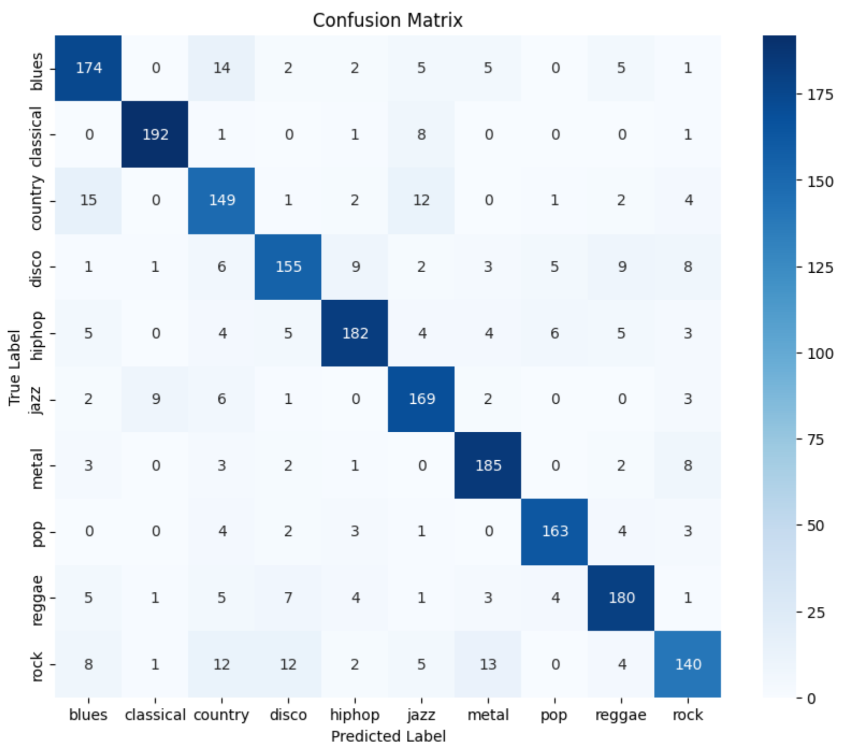
Figure 5.4 Confusion Matrix of SVM Model

Naïve Bayes Model



```
Accuracy: 84.53%
Classification Report:
              precision    recall  f1-score   support

       blues       0.82      0.84      0.83       208
   classical       0.94      0.95      0.94       203
     country       0.73      0.80      0.76       186
       disco       0.83      0.78      0.80       199
      hiphop       0.88      0.83      0.86       218
        jazz       0.82      0.88      0.85       192
       metal       0.86      0.91      0.88       204
         pop       0.91      0.91      0.91       180
      reggae       0.85      0.85      0.85       211
        rock       0.81      0.71      0.76       197

    accuracy                           0.85      1998
   macro avg       0.85      0.85      0.84      1998
weighted avg       0.85      0.85      0.84      1998
```

Figure 5.5 Evaluation of Naïve Bayes Model

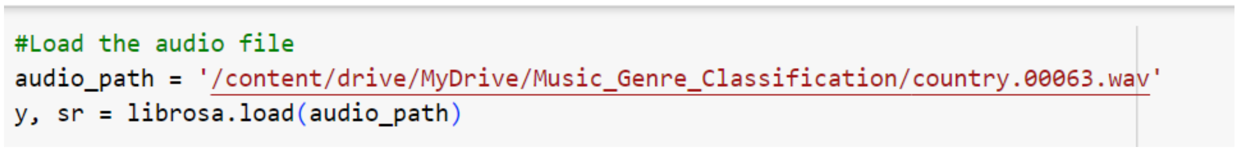


	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	174	0	14	2	2	5	5	0	5	1
classical	0	192	1	0	1	8	0	0	0	1
country	15	0	149	1	2	12	0	1	2	4
disco	1	1	6	155	9	2	3	5	9	8
hiphop	5	0	4	5	182	4	4	6	5	3
jazz	2	9	6	1	0	169	2	0	0	3
metal	3	0	3	2	1	0	185	0	2	8
pop	0	0	4	2	3	1	0	163	4	3
reggae	5	1	5	7	4	1	3	4	180	1
rock	8	1	12	12	2	5	13	0	4	140

Figure 5.6 Confusion Matrix of Naïve Baye’s Model

Compared the performance of three machine learning algorithms: KNN, SVM, and Naive Bayes. KNN and SVM produced nearly similar accuracy, demonstrating their robustness in classifying audio features extracted using the Librosa package. While Naive Bayes also showed reasonable results, it slightly underperformed compared to KNN and SVM. Based on the accuracy metrics, I decided to combine KNN and SVM for the final model deployment. By leveraging the strengths of both algorithms, I aim to enhance the classification accuracy and improve the system's performance for predicting music genres on unseen data.

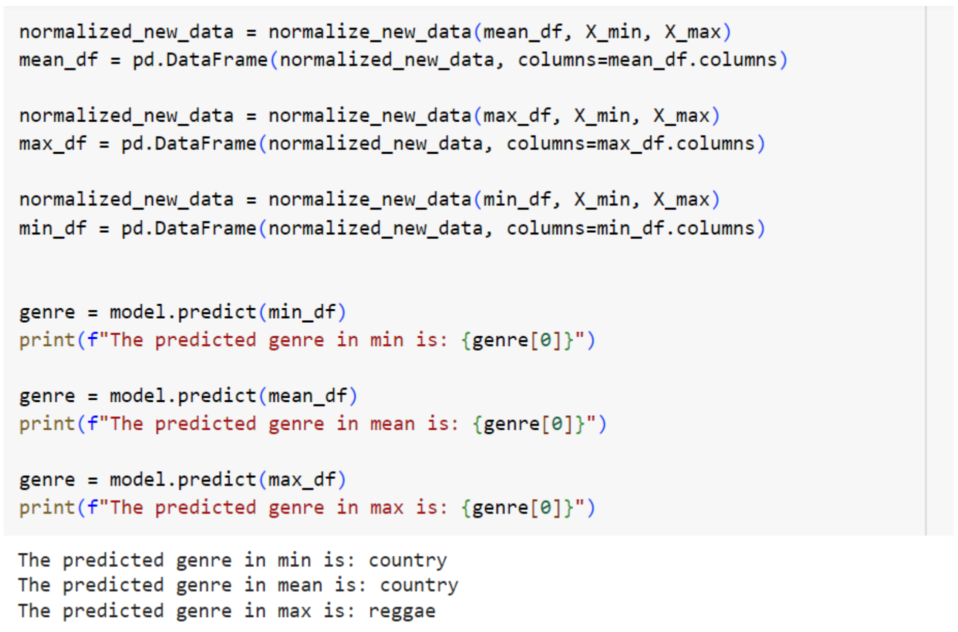
First upload and provide path to the audio file to find the Genre.



```
#Load the audio file
audio_path = '/content/drive/MyDrive/Music_Genre_Classification/country.00063.wav'
y, sr = librosa.load(audio_path)
```

Figure 5.7 Audio file input

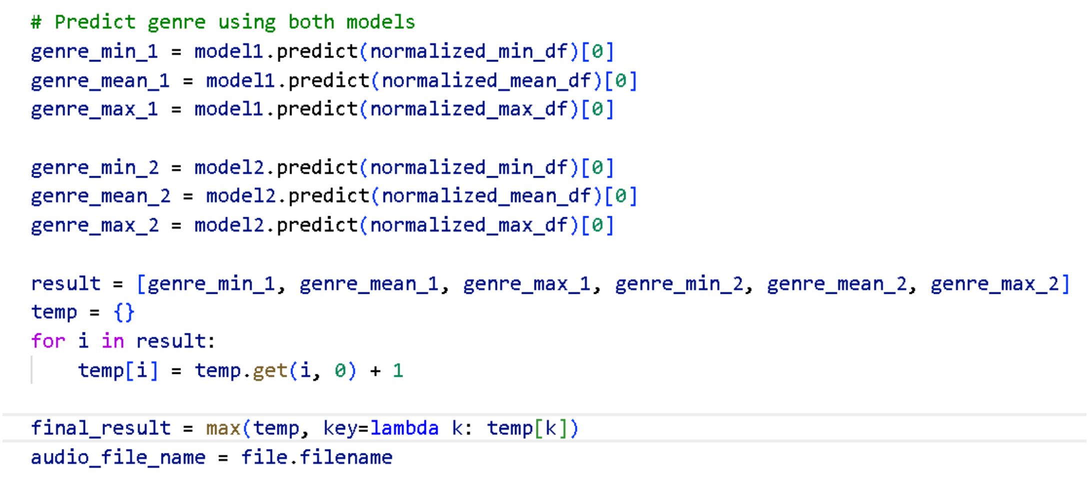
Generates a total of 3 predictions for each unseen audio file by using the minimum, mean, and maximum values of the extracted features. For both the KNN and SVM algorithms, each provides 3 predictions, resulting in a total of 6 predictions. The final class label for the audio file is determined by applying majority voting across these 6 predictions, ensuring a more reliable and robust classification of the music genre.



```
normalized_new_data = normalize_new_data(mean_df, X_min, X_max)
mean_df = pd.DataFrame(normalized_new_data, columns=mean_df.columns)

normalized_new_data = normalize_new_data(max_df, X_min, X_max)
max_df = pd.DataFrame(normalized_new_data, columns=max_df.columns)

normalized_new_data = normalize_new_data(min_df, X_min, X_max)
min_df = pd.DataFrame(normalized_new_data, columns=min_df.columns)


genre = model.predict(min_df)
print(f"The predicted genre in min is: {genre[0]}")

genre = model.predict(mean_df)
print(f"The predicted genre in mean is: {genre[0]}")

genre = model.predict(max_df)
print(f"The predicted genre in max is: {genre[0]}")
```

```
The predicted genre in min is: country
The predicted genre in mean is: country
The predicted genre in max is: reggae
```

Figure 5.8 Code Snippet of KNN making 3 predictions



```
# Predict genre using both models
genre_min_1 = model1.predict(normalized_min_df)[0]
genre_mean_1 = model1.predict(normalized_mean_df)[0]
genre_max_1 = model1.predict(normalized_max_df)[0]

genre_min_2 = model2.predict(normalized_min_df)[0]
genre_mean_2 = model2.predict(normalized_mean_df)[0]
genre_max_2 = model2.predict(normalized_max_df)[0]

result = [genre_min_1, genre_mean_1, genre_max_1, genre_min_2, genre_mean_2, genre_max_2]
temp = {}
for i in result:
    temp[i] = temp.get(i, 0) + 1

final_result = max(temp, key=lambda k: temp[k])
audio_file_name = file.filename
```

Figure 5.9 Code snippet for assigning the final class

6. MODEL DEPLOYMENT

The principal objective of model deployment is to seamlessly integrate the classification models developed during the project into a production environment. This transition empowers end-users to access and leverage the classification system for making informed decisions. Model deployment is a critical phase that bridges the gap between theoretical model development and practical, real-world application. By deployingthe classification models, the project aims to provide a tangible and accessible solution that fulfills the needs and preferences of the target audience.

To facilitate the deployment process, the project utilizes Python Flask framework and Librosa Package (to extract audio features). These tools contribute tothe efficiency and effectiveness of model integration into the production environment.

User Interface (UI):

The recommendation system is integrated into the user interface, creating a cohesive and user-friendly experience. Users can interact with the system through a web interface.

The following figures shows user interface of this application. This interface is simple and easy to understand. This application's user interface is designed for straightforward interaction beginning with a clean introduction page of 'SOLMUSIC'(Music Genre Classification System). The home page allows users to upload audio files and view how the entire system works.

After uploading an audio file, you will be routed to the genre prediction page where a single music genre will be provided as the final output. This output is determined by taking the majority class from a total of 6 predictions: 3 predictions generated by the KNN model and 3 predictions generated by the SVM model, using the minimum, mean, and maximum extracted features from the audio file. A "Back" button is available to return to the upload page, allowing you to upload a new audio file and repeat the process.

Only WAV audio formats are allowed for upload in this system. If a file in another format is uploaded, an error message stating "Only WAV files are allowed!" will appear. Along with the error, users will be provided with an option to convert their audio file to WAV format via a link, making it easy to convert the file and continue with the genre prediction process seamlessly. This ensures that the model only processes compatible audio files for accurate predictions.

Additionally, the result page includes a play/pause button for the uploaded audio file, allowing users to listen to the file while viewing the prediction results.

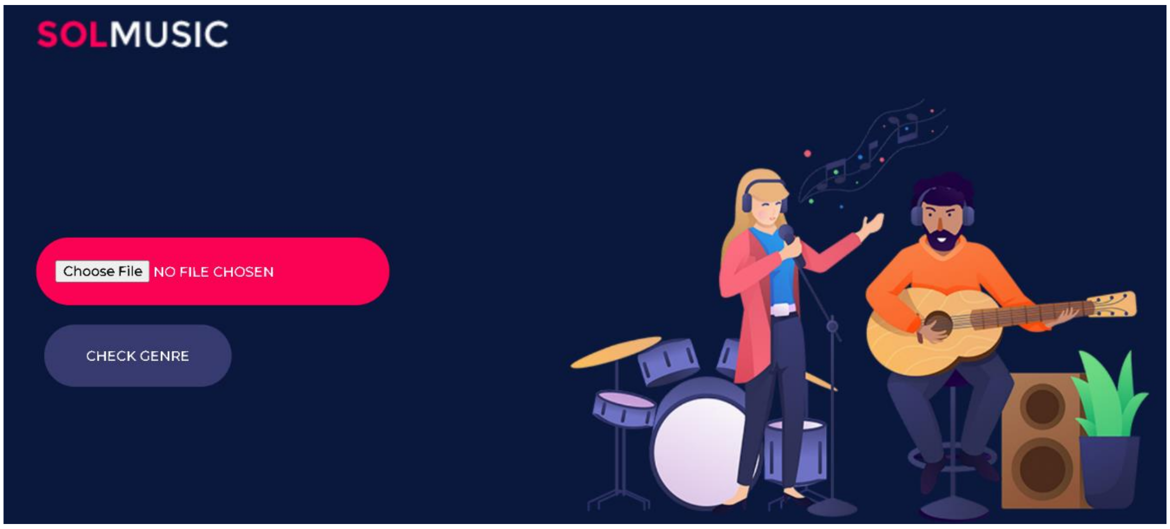
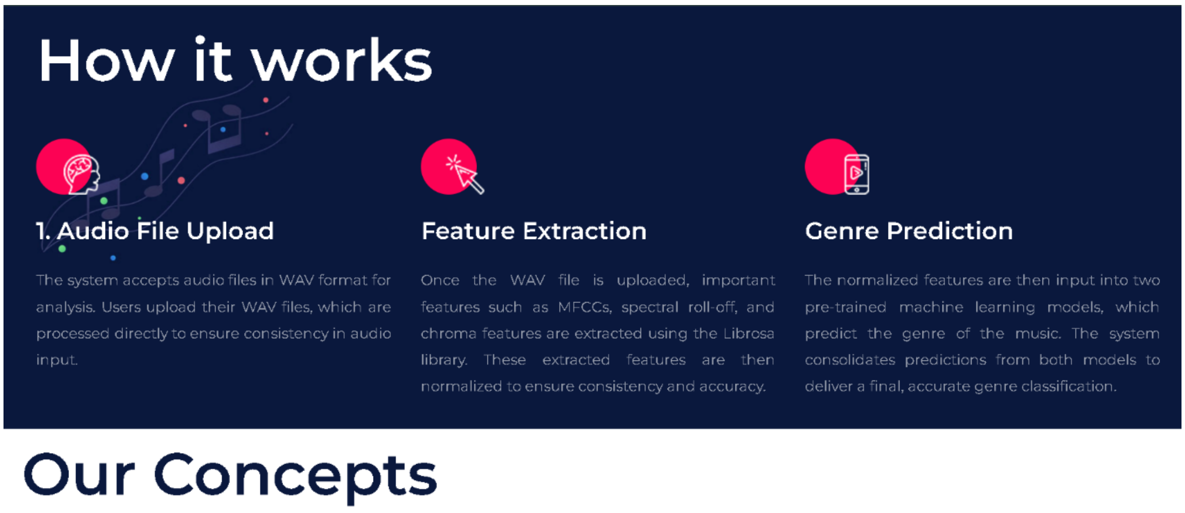


Figure 6.1 Home Page



How it works

1. Audio File Upload

The system accepts audio files in WAV format for analysis. Users upload their WAV files, which are processed directly to ensure consistency in audio input.

Feature Extraction

Once the WAV file is uploaded, important features such as MFCCs, spectral roll-off, and chroma features are extracted using the Librosa library. These extracted features are then normalized to ensure consistency and accuracy.

Genre Prediction

The normalized features are then input into two pre-trained machine learning models, which predict the genre of the music. The system consolidates predictions from both models to deliver a final, accurate genre classification.

Our Concepts

Figure 6.2 UI for working of System

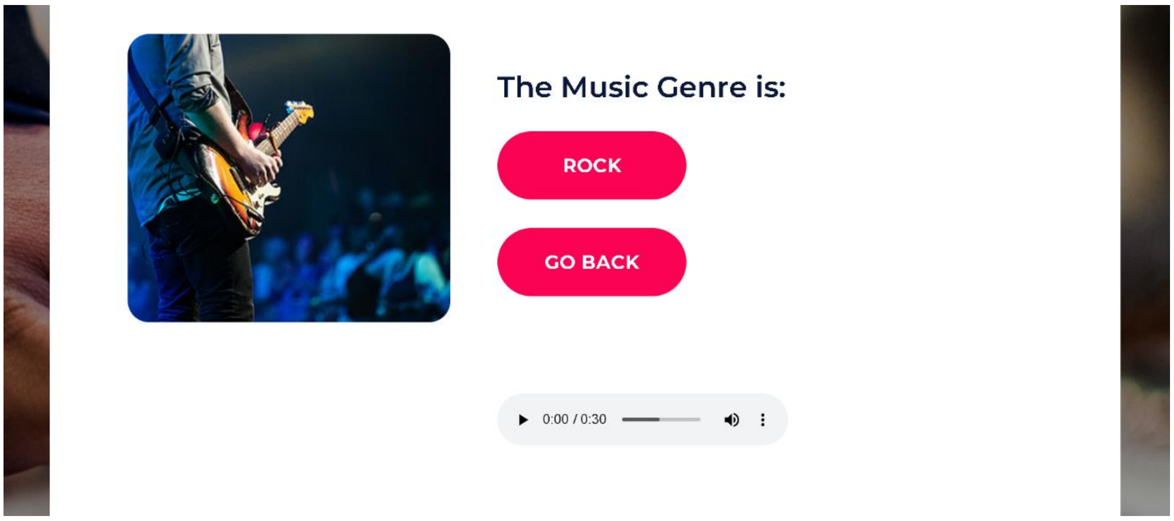
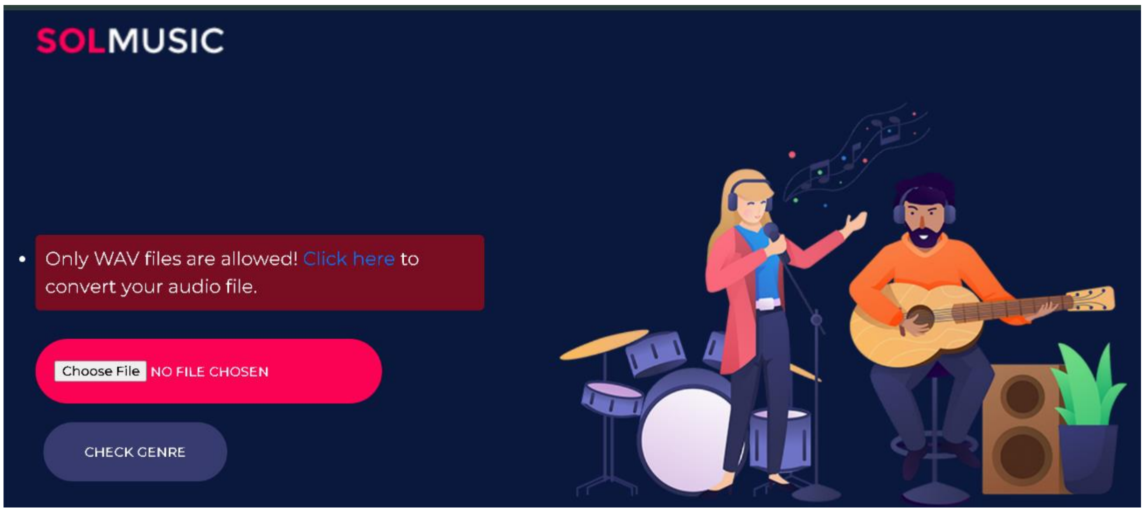


Figure 6.4 Result Page

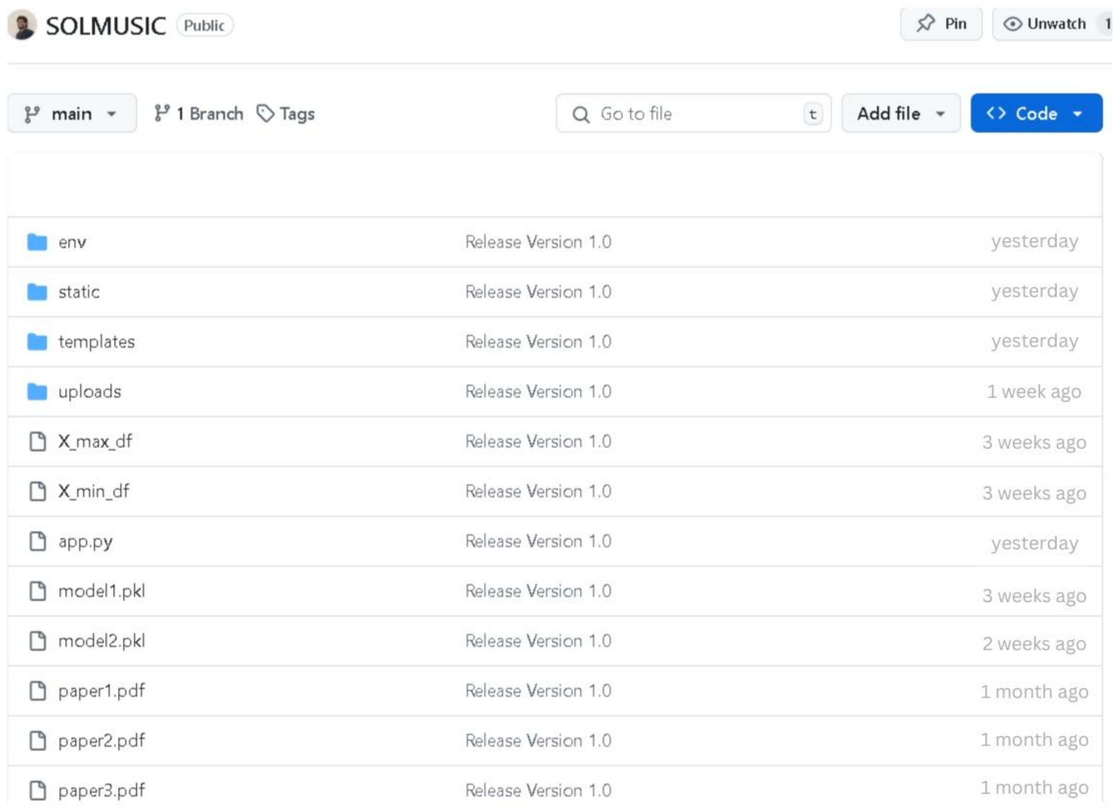


- Only WAV files are allowed! Click here to convert your audio file.

Figure 6.5 Error Message

7. GIT HISTORY

Git Repository SOLMUSIC contains all colab files, py files, html files and three related research papers. It is maintained for systematic way of project presentation and mainly for future reference. The colab files are created separatelyfor threesprint releases during the project.



env	Release Version 1.0	yesterday
static	Release Version 1.0	yesterday
templates	Release Version 1.0	yesterday
uploads	Release Version 1.0	1 week ago
X_max_df	Release Version 1.0	3 weeks ago
X_min_df	Release Version 1.0	3 weeks ago
app.py	Release Version 1.0	yesterday
model1.pkl	Release Version 1.0	3 weeks ago
model2.pkl	Release Version 1.0	2 weeks ago
paper1.pdf	Release Version 1.0	1 month ago
paper2.pdf	Release Version 1.0	1 month ago
paper3.pdf	Release Version 1.0	1 month ago

Figure 7.1 Git History

8. CONCLUSION

In conclusion, the Music Genre Classification project has successfully achieved its primary objective of accurately classifying audio tracks into their respective genres through the development and deployment of robust machine learning models. Leveraging the k-Nearest Neighbors (k-NN) and Support Vector Machine (SVM) algorithms, the system excels in predicting music genres by analyzing complex audio features and identifying distinct patterns within the data. The project's technical contributions, including feature selection using the chi-square method and the combination of KNN and SVM models, ensure high accuracy and efficiency in handling diverse and evolving audio datasets. The user-centric design integrates a seamless web interface, allowing users to upload audio files and receive accurate genre predictions with ease. The successful deployment of production-ready models, real-time genre classification, and frontend integration demonstrates the system's transition from theoretical development to practical application. Security measures, such as file validation and format restrictions, coupled with real-time audio playback features, contribute to the system's reliability and user engagement. Lessons learned from this project will guide future improvements, and plans for enhancing feature selection and model performance ensure continuous refinement. The Music Genre Classification System, grounded in data science principles and audio analysis, is well-positioned to offer personalized and accurate genre predictions, enriching user experience in the dynamic field of music categorization.

9. FUTURE WORK

Advanced Personalization Algorithms:

Exploring more sophisticated personalization algorithms represents a key avenue for future work. Integrating advanced machine learning techniques, such as deep learning, could enable the model to capture intricate patterns in user preferences anditem features, leading to more accurate classifications. Additionally, investigating hybrid models offer a comprehensive solution to enhance classifications precision.

Scalability and Infrastructure Optimization:

As user bases expand, ensuring the scalability of the classification system becomes imperative. Future work may involve optimizing the system's infrastructure, exploring distributed computing frameworks, and leveraging cloud services to handle increasing data volumes and user interactions efficiently, ensuring a seamless and responsive user experience.

Interactive User Feedback Mechanisms:

Implementing an interactive feedback loop empowers users to provide explicit feedback on classifications. Leveraging this feedback, the system can iterativelyadapt and improve its accuracy, fostering a collaborative user-system interaction.

Overlapping audio Segments:

A novel segmentation strategy by splitting audio files into 4-second segments and overlapping each by 2 seconds. This overlapping approach ensures that consecutive segments share contextual audio information, helping the model perceive continuity within the audio. This significantly enhances the classification accuracy by ensuring that the model understands the segments as part of a cohesive audio file rather than isolated snippets.

In summary, the future work for the Music Genre Classification System project encompasses advanced personalization algorithms, scalability and infrastructure optimization, interactive user feedback mechanisms, and innovative audio segmentation strategies. By exploring sophisticated machine learning techniques and hybrid models, the system can achieve more accurate classifications that capture intricate patterns in audio features. Ensuring scalability through optimized infrastructure and distributed computing will accommodate expanding user bases and data volumes, delivering a seamless experience. Implementing interactive feedback loops will enable users to provide explicit input, allowing the system to iteratively enhance its accuracy. Additionally, employing a novel segmentation strategy with overlapping audio segments will improve classification accuracy by maintaining contextual continuity within audio files. Addressing these aspects will facilitate the evolution of the system into a robust, user-centric platform that adapts to dynamic musical preferences and enhances the overall music discovery experience.

10. APPENDIX

10.1. MINIMUM SOFTWARE REQUIREMENTS

To deploy and run Music Genre Classification System, the followingminimum software requirements must be met:

- Python:

  Version 3.6 or above is required to execute the project code and its dependencies.

- NumPy:

  Essential for numerical operations and array manipulations in Python.

- Pandas:

  Necessary for data manipulation, cleaning, and analysis.

- Scikit-learn:

  The scikit-learn library is used for machine learning functionalities, including theimplementation of the k-Nearest Neighbors (k-NN) algorithm.

- Flask:

  A lightweight web framework to serve the application and provide the user interface. Flask is required to manage HTTP requests and deliver predictions via the web interface.

- SciPy:

  Used for scientific and technical computing, particularly for sparse matrix operations.

- Librosa

  Used for extracting audio features from uploaded audio file.

10.2. MINIMUM HARDWARE REQUIREMENTS

The Music Genre Classification System is relatively lightweight, but it does involve some computational processes. The minimum hardware requirements to ensure smooth execution are as follows:

Processor (CPU):

A modern multi-core processor (e.g., Intel Core i5 or equivalent) is recommended to handlethe computation involved in the recommendation algorithms efficiently.

RAM:

A minimum of 8 GB RAM is recommended to manage the dataset and perform machine learning operations smoothly.

Storage:

Adequate storage space for storing datasets, code, and any additional resources. While the system itself doesn't require extensive storage, having sufficient space for datasets and potential model persistence is essential.

Internet Connection:

An internet connection is necessary for downloading datasets, libraries, and dependencies during the setup phase.

11. REFERENCES

1. Ndou N, Ajoodha R, Jadhav A. Music genre classification: A review of deep-learning and traditional machine-learning approaches. In2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS) 2021 Apr 21 (pp. 1-6). IEEE.
2. Prajwal, Shubham, Naik, & Sugna : Music Genre Classification Using Machine Learning. International Research Journal of Modernization in Engineering Technology and Science (IRJMETS) 2021 Jul 07 (pp. 1-16). IEEE.
3. Rahardwika DS, Rachmawanto EH, Sari CA, Irawan C, Kusumaningrum DP, Trusthi SL. Comparison of SVM, KNN, and NB classifier for genre music classification based on metadata. In2020 international seminar on application for technology of information and communication (iSemantic) 2020 Sep 19 (pp. 12-16). IEEE.
4. https://www.kaggle.com/sets/andradaolteanu/gtzan-dataset-music-genre-classification
5. Ghildiyal A, Singh K, Sharma S. Music genre classification using machine learning. In2020 4th international conference on electronics, communication and aerospace technology (ICECA) 2020 Nov 5 (pp. 1368-1372). IEEE.