A COMPARATIVE STUDY OF AUGMENTED FEATURES AND OTHER ENSEMBLE APPROACHES FOR MUSIC GENRE CLASSIFICATION

RAAD SHARIAT
Master of Information and Communication Engineering, Korea University of Technology and Education, 2019

A thesis submitted
in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

A COMPARATIVE STUDY OF AUGMENTED FEATURES AND OTHER ENSEMBLE APPROACHES FOR MUSIC GENRE CLASSIFICATION

RAAD SHARIAT

Date of Defence: April 13, 2023

Dr. John Zhang Thesis Supervisor	Associate Professor	Ph.D.
Dr. Hadi Kharaghani Thesis Co-supervisor	Professor	Ph.D.
Dr. Yllias Chali Thesis Examination Committee Member	Professor	Ph.D.
Dr. Wendy Osborn Thesis Examination Committee Member	Associate Professor	Ph.D.
Dr. John Sheriff Associate Chair, Thesis Examination Committee	Assistant Professor	Ph.D.

Dedication

To my family, who have always supported me and believed in my ability to succeed.

Abstract

Music genre classification is essential in the music streaming industry, with many recommendation systems relying on data mining techniques to accurately classify musical genres. However, classifying music genres is challenging due to the inherent diversity of music, even within a single genre. This diversity can make it difficult for machine learning models to classify music accurately, leading to the development of various techniques to improve the performance of these models. One such technique is ensemble learning, which combines the predictions of multiple models to improve the overall accuracy of the ensemble. In this thesis, we propose a new ensemble method called "*Augmented Features*," which combines the predictions of multiple models by augmenting the input features with additional derived features. This technique can improve the performance of ensemble models by providing additional information to the models, allowing them to capture the music data's complexity better. To evaluate the performance of our ensemble method, we conducted experiments on various music datasets combined with different feature selection techniques. We compared the results to those obtained using the base classifiers and other ensemble methods, including voting, blending, and stacking. Our results showed that the augmented features method repeatedly outperformed the different techniques, particularly on datasets with high dimensionality and complex relationships between features. It is hoped that this work will significantly contribute to ensemble methods and improve the performance of machine learning models in various applications.

Acknowledgments

I extend my sincerest thanks to my academic advisors, Professor John Zhang and Professor Hadi Kharaghani, for their exceptional guidance and support throughout my research project. Their unwavering encouragement and readiness to offer their assistance were a great source of strength and motivation for me.

In addition to my supervisors, I would also like to express my gratitude to my committee members, Professor Wendy Osborn and Professor Yllias Chali, for their insightful feedback and suggestions. Their expertise and knowledge have helped shape my research into what it is today.

I am also deeply grateful to my family - my loving mother, father, and wife - for their unconditional love, support, and encouragement throughout my academic journey. With their guidance and encouragement, I completed my research and achieved this milestone.

Finally, I thank anyone who has provided me with support and encouragement throughout my research journey. Your assistance has meant a lot to me and has contributed significantly to my success. And to anyone reading this thesis, thank you for taking the time to learn about my research. Your support and encouragement are greatly appreciated.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Music is a universal language that brings people together by expressing feelings, thoughts, and experiences. It is integrated into our lives and can express our emotions and memories [1]. The diversity and richness of music are reflected in the multitude of genres that exist, each with its unique style, sound, and cultural significance [2] such as Pop, Jazz, Rock, Blues, and Hip Hop. The task of accurately predicting the genre of a musical piece is challenging due to the subjective nature of music and the diverse range of musical styles.

The problem of music genre prediction is important as it has applications in various fields, such as music recommendation [3] and content-based music classification [4]. With an ever-growing catalog of music and the importance of accurately categorizing it for music streaming services, producers, and songwriters, it is essential to develop models that can accurately predict music genres. However, current methods for music genre prediction have limitations. They can fail to accurately predict the genre of a musical piece due to the constraints of the existing datasets and the need for more robust and scalable machine learning models.

This thesis aims to develop a novel classification approach for music genre prediction that can overcome some of the limitations of current methods and improve their performance. The objectives of this study are to:

- Explore, prepare, and utilize some existing popular music datasets,
- Develop and evaluate an enhanced approach for music genre prediction,
- Compare the performance of the proposed approach with traditional methods to demon-

strate its effectiveness and advantages.

The scope of this thesis will focus on predicting music genres in the Western classical and popular music domains using pre-existing datasets that include various musical features.

1.1 Research Overview

This study contributes to the advancement of the field of music genre prediction and has implications for related areas such as music recommendation and music information retrieval and could be potentially extended to other applications of machine learning. In this research, a novel ensemble learning model called *"Augmented Features"* was introduced, which averages the prediction probabilities of base learners (classifiers) and enhances the features in a dataset to improve the prediction performance. In addition, our research compared and analyzed several classification models and feature selection techniques for music genre categorization. These models embody single-classifier classification methods, including the Multi-layer Perceptron [5], Random Forest [6], XGBoost [7], LightGBM [8], and K-Nearest Neighbors (k-NN) [9] classifiers, as well as ensemble learning methods [10] such as voting, blending, and stacking [11]. Furthermore, we explored various feature selection approaches, including embedded-based and wrapper methods [12] such as Recursive Feature Elimination (RFE), tree-based feature selection, forward, backward, and bidirectional feature selection methods to classify diverse music genres. Finally, the models and feature selection techniques mentioned above were applied to three distinct datasets to evaluate the combination of these feature selection methods and models to discover the one that resulted in the highest classification accuracy. Our results showed that our proposed ensemble learning model outperformed traditional ensemble techniques in several cases.

1.2 Motivation for Using Ensemble Learning

Ensemble learning is a technique used in machine learning that combines the predictions of multiple models to improve overall performance and achieve more accurate and

robust results and has gained popularity in the field of classification due to its ability to produce more accurate predictions than individual models. Ensemble models can capture a broader range of patterns in the data and reduce overfitting by aggregating the predictions of multiple models. Another advantage of ensemble methods is their robustness to model errors. By using ensemble models, the errors made by different models in the ensemble may compensate for each other when the predictions are combined, resulting in more stable and accurate predictions. The structure of an example meta-model-based ensemble learning technique (i.e. stacking) is presented in Figure 1.1, where the predictions of N base models are combined to train a meta-model for final class prediction.

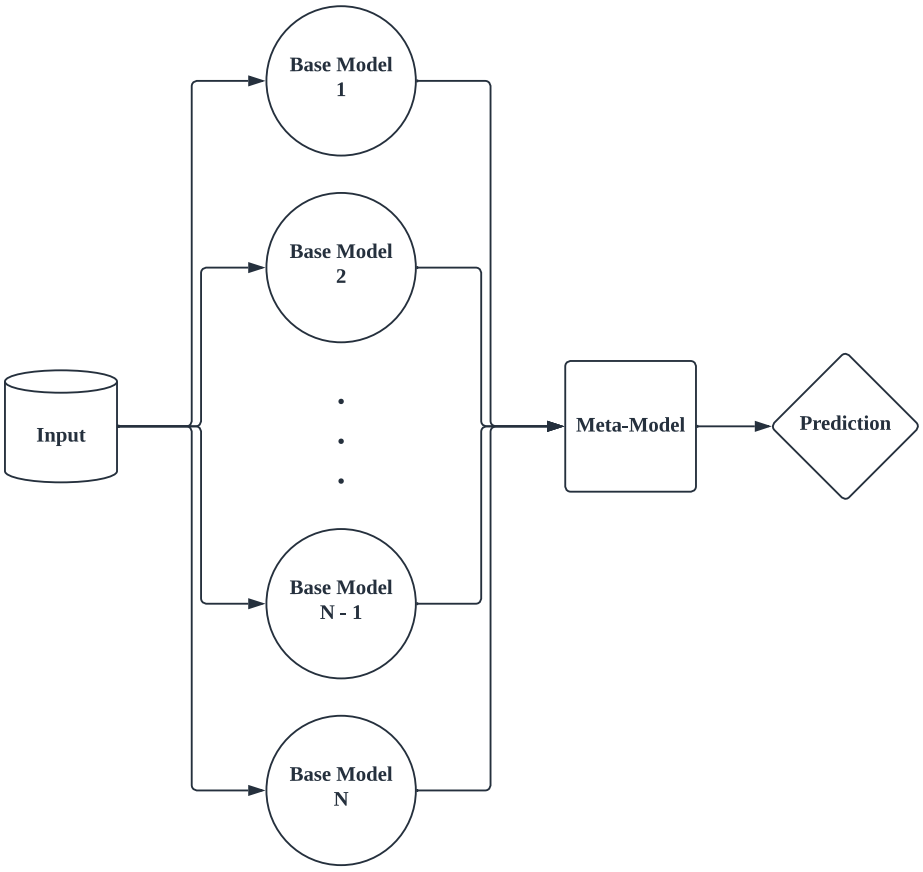

Figure 1.1: Basic design of an ensemble learning model.

In addition to the improved prediction accuracy of ensemble methods, they can be more efficient to train than individual models. Ensemble models can be trained in parallel, significantly reducing the training time. Furthermore, using an ensemble of models makes it possible to capture the underlying structure of the data better and achieve improved generalizability to new unseen data by addressing the issues of overfitting and bias that can arise in individual models.

1.3 Framework of the Research

As outlined in Figure 1.2, three datasets with different sizes and features are utilized in our study to evaluate the performance of the proposed method compared to other commonly used models. First, data preprocessing is conducted to prepare the datasets for our research. We select the dataset's five most frequent genres in data preprocessing. Then, feature conversion is carried out to convert the features into a suitable format for our experiment. Other measures, such as repositioning the target feature, separating the components, encoding the target feature, and scaling the features, are also implemented to ensure the data is in the correct format for classification. After preparing the data, the next step is to select the most relevant features from the dataset. Five feature selection methods are employed in the experiment to accomplish this task. Once the relevant features have been identified, the five different base classifiers used in the experiment are tested. Finally, three traditional ensemble methods, including stacking, voting, and blending, and the proposed ensemble learning model are employed to predict music genres.

Genre prediction is performed using only base classifiers with and without feature selection and ensemble techniques for a fair comparison between the base classifiers and the ensemble methods. Furthermore, the performance of the proposed method and mentioned models are evaluated by cross-validation, a widely used method for assessing the performance of machine learning models [13], allowing a proper comparison between different approaches.

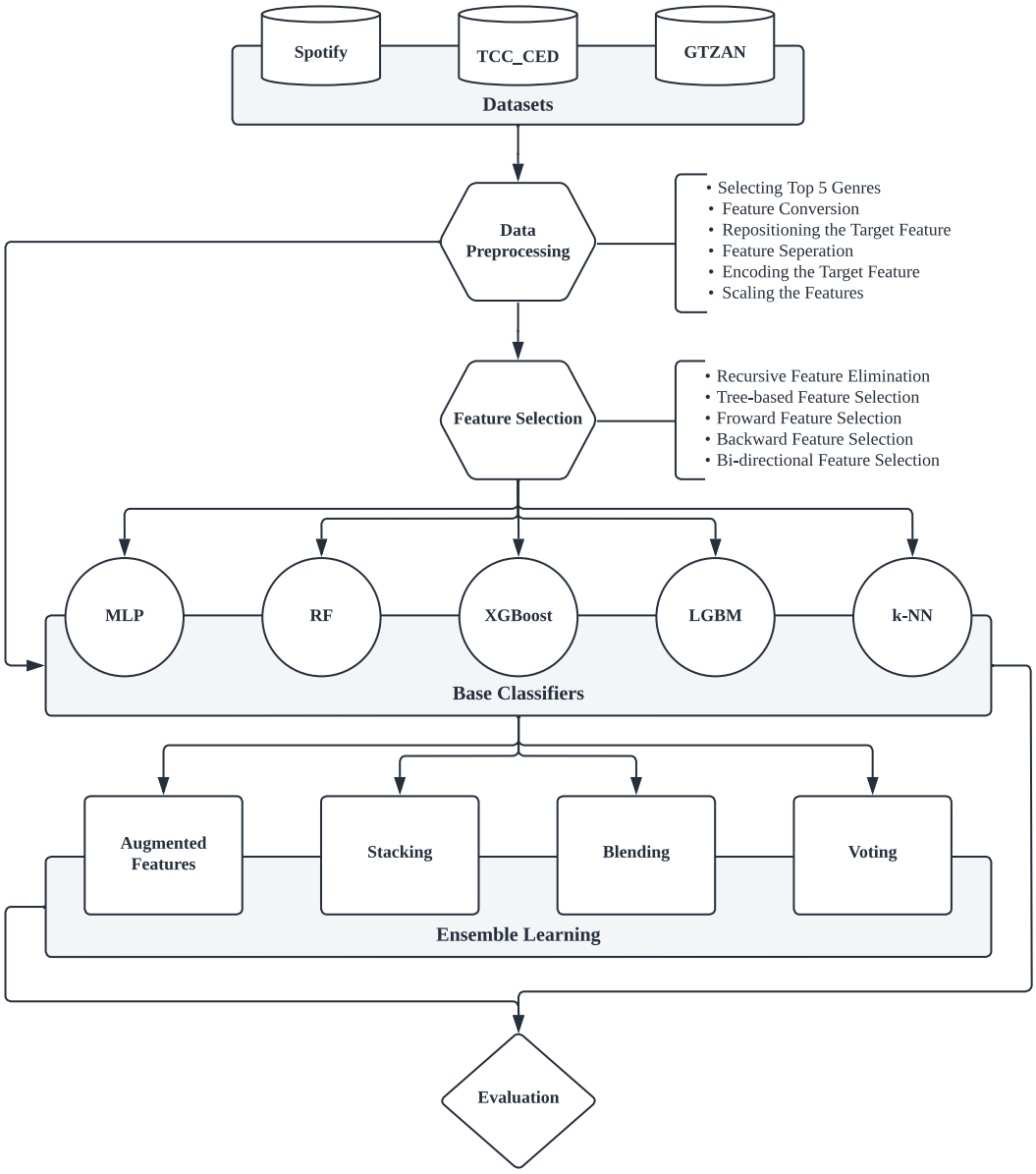

Figure 1.2: Architecture of the proposed framework.

1.4 Thesis Structure

The thesis incorporates six chapters, including this chapter, that thoroughly examine and analyze the topic. Chapter 2 presents a comprehensive overview of the literature on the subject, synthesizing the key concepts and ideas from previous studies. Chapter 3 introduces

the three datasets used in this research (Spotify, TCC_CED, and GTZAN) and the data preprocessing steps employed to prepare them for genre classification in our study. Chapter 4 delves into the specifics of the research methodology proposed, including a detailed explanation of the augmented features ensemble model and algorithms utilized. In Chapter 5, the study's results are analyzed and discussed, focusing on assessing the performance of the proposed augmented features ensemble model and the implications of these findings. Finally, Chapter 6 serves as the conclusion of the thesis, demonstrating an overview of the primary points addressed and potential avenues for our future research.

Chapter 2

Literature Review

Machine learning is an effective tool in many domains, including image recognition [14], natural language processing [15], bioinformatics [14], and other fields. However, in this research, we are particularly interested in exploring machine learning techniques in music genre classification. Music genre classification is a fundamental problem in *"Music Information Retrieval"* and has been a topic of substantial research in the field.

This chapter will introduce music information retrieval and explore the various approaches used for genre classification, including machine learning, deep learning, and ensemble learning models. We will review the literature on these approaches and discuss their strengths and limitations in the context of music genre classification. We will also examine other methods used in music genre classification to provide a comprehensive overview of this topic and its potential future directions.

2.1 Music Information Retrieval

Music Information Retrieval (MIR) is a field that deals with the automatic extraction of information from music, which can include tasks such as automatically identifying the artist and the title of a song, detecting the genre of a piece of music, or even transcribing the music into a symbolic representation such as sheet music. Kassler first introduced MIR in 1966 [16], and nowadays, it is used in various applications, such as music recommendation systems, music classification, and music search engines. In the digital age, music is easily accessible, and the demand for personalized recommendations has increased, making

MIR more critical and widespread. One key aspect and core task of MIR is music genre classification, which involves categorizing music by style or type. However, accurately classifying music by genre can be challenging due to the subjectivity of genre labels and the constantly evolving nature of music. Therefore, researchers have developed more precise systems and models to address this challenge using machine learning [17], deep learning [18], and natural language processing [19] techniques. These approaches have often been effective in identifying patterns and trends in music data and improving music recommendation systems. Music recommendation systems suggest music to users based on their preferences and listening history and use various techniques, including collaborative and content-based filtering, to make personalized recommendations [20, 21, 22]. In addition to genre classification and recommendation, MIR includes tasks such as music transcription, which involves converting audio recordings of music into a written form [23], and music generation [24], which consists in creating new music using computational techniques.

Furthermore, music directly impacts the human brain and can affect cognition and mood [25]. As a result, the effectiveness of MIR systems depends on their ability to provide access to the appropriate genre of music at the right time. For instance, music streaming platforms such as Spotify[1] and SoundCloud[2] use MIR techniques to classify music and offer recommendations to their users.

As discussed, MIR is a crucial music industry aspect with numerous applications. Machine learning approaches have proven effective in improving MIR systems, and music streaming platforms use these systems to classify music and offer personalized recommendations to their users. However, MIR has challenges based on the nature of music [26]. One major challenge is the subjectivity of genre labels, making it difficult to classify music accurately. Another challenge is the sheer volume of music data available, making it difficult for MIR systems to keep up with the constantly evolving music landscape [27]. Despite these challenges, MIR remains a significant area of research and development, with re-

[1] https://open.spotify.com/.
[2] https://soundcloud.com/.

searchers working to improve the accuracy and effectiveness of MIR systems. In the future, MIR will continue to play a significant role in the music industry as technology advances and the demand for personalized recommendations grows. As MIR systems become more sophisticated, they can better meet music listeners' needs and help them discover new and exciting music.

2.2 Music Genre Classification

Music genre classification is a fundamental problem in MIR as it helps to organize and categorize music, making it easier for people to find and enjoy the music they like. However, with the massive amount of music available today, manual classification is becoming increasingly more complex and time-consuming, which is why automating the process by employing machine learning and deep learning techniques has become popular. These techniques have been widely used to analyze music data features for genre classification and have shown to be effective in genre classification by achieving high accuracy and performance. In addition to machine learning and deep learning techniques, researchers have also utilized ensemble learning methods to improve the performance of genre classification. Ensemble learning methods involve combining multiple models to create a more robust and accurate approach. Further research in this area will likely lead to more precise and sophisticated techniques for genre classification, making it more convenient for music enthusiasts to discover the music they enjoy. In this section, we will explore various research and studies on genre classification and evaluate the performance of machine learning, deep learning models, and ensemble learning methods for this task.

2.2.1 Machine Learning Approaches

The literature presented in this section describes various studies conducted on music genre classification using different machine learning algorithms and feature extraction techniques. For example, Li et al. [28] proposed an innovative approach for feature extraction

called Daubechies Wavelet Coefficient Histograms (DWCHs) for music genre classification. In addition, the authors validated the performances of different machine learning algorithms, including Support Vector Machines (SVM) [29] and Linear Discriminant Analysis (LDA) [30]. The first dataset in this study comprised 756 songs with more than five different genres, while the second dataset contained 756 songs with more than ten different genres. The authors utilized ten-fold cross-validation to evaluate the performance of the models. Their results showed the SVM achieved the highest average accuracy when using the DWCHs technique.

The Synat database[3], containing over 50,000 records and 13 musical genres, was used by Aldona R. and Bozena K. [31] to show effective automatic musical genre classification. During the preprocessing stage, the tracks were segmented, and the feature vectors were expanded by parameters relating to the musical instruments that are distinctive for the specified musical genre. Next, the recordings were divided using a semi-supervised instrument separation based on Non-negative Matrix Factorization (NMF) [32]. After that, the data were scaled via normalization to improve model performance. Finally, the model, which created a classifier from labeled and unlabeled data, was trained. The outcomes revealed that the SVM's overall classification accuracy was around 72%. Additionally, the study showed that separating instrumental tracks can be an effective feature representation for music genre classification. The study results demonstrated that the proposed method was a promising approach for music genre classification and could be reliable for this task.

Acoustic elements of music were retrieved using digital signal processing techniques in work presented by Elbir et al. [33]. Subsequently, the authors utilized machine learning algorithms to classify music genres and offer recommendations. SVM fared better in classification accuracy than all the other methods. In this study, employing a deep learning algorithm established no noticeable difference in performance regarding music genre classification.

[3] https://qoe.agh.edu.pl/synat-database/.

In a research conducted by Chillara et al. [34], the Free Music Archive (FMA tiny) dataset [35] was employed to investigate the categorization of genres. The authors offered a simple solution to the classification problem and compared it to several more complex, reliable models. In this research, the image-based classification outperformed the feature-based classification.

The research proposed by Asim and Siddiqui [36] compared the performances of k-Nearest Neighbors (k-NN) and SVM to pick the best classification algorithm for grouping music genres. The authors conducted their research using both Principal Component Analysis (PCA) [37] and without PCA. They suggested that SVM gave better results with 77% accuracy without dimensionality reduction.

Sumanth and Priyadarshini [38] compared linear regression and the Extreme Gradient Boost (XGBoost) algorithm for music genre classification. The authors argued that linear regression was a simple and widely used method for classification, but its performance could have been improved. In contrast, XGBoost, a more robust algorithm based on gradient boosting, could achieve higher accuracy. Results showed that the classification of music based on genre using XGBoost (59.0%) appeared to be more accurate when compared to Linear Regression (53.0%).

2.2.2 Deep Learning Approaches

Prabhakar and Lee [39] presented a new method for music genre classification that utilized deep learning techniques. The authors argued that previous methods for genre classification often rely on a single feature extraction method or machine learning algorithm, which can lead to suboptimal performance. Instead, they proposed a holistic approach that combined multiple feature extraction methods and machine learning algorithms to improve the accuracy of genre classification. Three music datasets were utilized in the trials: GTZAN[4], The International Society for Music Information Retrieval (ISMIR)[5], and Mag-

[4] https://www.tensorflow.org/datasets/catalog/gtzan.
[5] https://ismir2004.ismir.net/.

naTagATune[6]. When the suggested deep learning BAG model was employed, a substantially higher classification accuracy of 93.51% was reached. They showed that combining multiple feature extraction methods, deep learning algorithms, and efficient transfer learning techniques can improve music genre classification performance.

In a study by Singh and Biswas [40], the effectiveness of diverse musical and non-musical characteristics was evaluated using deep learning models of various complexities for music genre classification. The study trained the models on four different datasets and utilized a variety of musical and non-musical characteristics. After training, the results of the training and validation splits were analyzed. Investigating the different attributes helped uncover the datasets' underlying properties and robustness.

Dias et al. [41] employed Convolutional Neural Network (CNN) [42] to categorize music into several genres based on the features extracted from audio recordings from the GTZAN dataset. Additionally, a recommendation algorithm was implemented to suggest similar pieces of music to the users. Similarly, Zhang et al. [43] investigated the effectiveness of using CNNs for music genre classification on the GTZAN dataset. Through experimentation, they found that combining max-pooling and average-pooling and using shortcut connections to skip one or more layers effectively improve classification with CNNs.

The study by Patil and Komati [44] proposed a new system for classifying music genres using a neural network. Their system included three main processes: data preparation, feature extraction, and classification. The authors utilized the spectrogram's feature values as input for their proposed system architecture. They evaluated the performance of the proposed system on the GTZAN dataset and compared it to the existing methods. The study also tested the proposed methodology on Indian music.

Mughal et al. [45] presented another approach for classifying music genres, focusing on music in the Urdu language. The authors created a dataset of Urdu music tracks and used it to train and evaluate different classification models, focusing mainly on CNNs. The

[6] `https://paperswithcode.com/dataset/magnatagatune`.

models' performances were compared in terms of validation accuracy. The model based on CNN achieved the highest level of accuracy of above 90% compared to other techniques.

Multi-layer Independent Recurrent Neural Network (IndRNN) was used to classify music genres in the GTZAN dataset by Wu et al. [46]. The authors used scattering transform to decrease data redundancy. Compared to other cutting-edge models, the IndRNN-based network in this study achieved substantial recognition rates. In addition, it fared well in terms of accuracy and speed when classifying music genres.

Liu. et al. [47] suggested a deep learning-based middle-level feature interaction strategy. They addressed the problem of independent branches in existing multi-feature models, resulting in a need for feature learning for classification. The proposed method showed improvement in classification results compared to state-of-the-art methods. Furthermore, it revealed that the interaction between features could have a gain or inhibitory effect on classification performance.

Pelchat et al. [48] used a CNN to classify musical genres using spectrogram images of brief time segments of songs. The network consisted of six convolutional layers, a fully connected layer, and a softmax function for genre classification. The accuracy in this study was 85%.

Rani et al. [49] introduced a technique that employed a CNN to determine the mood of songs using its genre pitch. The method started by converting the music clip into a spectrogram, then used as an image for CNN to classify the song's mood. The proposed approach was tested on more than 450 music samples and achieved a 90% accuracy in mood classification.

In the study by Cheng et al. [50], a hybrid model combining CNN and Recurrent Neural Network (RNN) [51] was used for music genre classification using the GTZAN dataset. The audio samples were preprocessed using MFCC, similar to human hearing, and used as the feature vector for the model. As a result, the hybrid model achieved an accuracy of 43% with a faster training speed by leveraging the strengths of both CNN and RNN for process-

ing the data. Similarly, Wang and Sohail [52] proposed a deep-learning-based music style classification method, combining CNN and RNN with MFCC coefficient features extracted by discrete Fourier transform. This method had a classification accuracy of 93.3%.

The Bottom-up Broadcast Neural Network (BBNN) is a novel architecture proposed by Liu et al. [53], designed to classify music genres. The BBNN model effectively utilizes both low-level musical features and high-level semantic information. It was tested on datasets like GTZAN, Ballroom[7], and Extended Ballroom [54] and achieved a high classification accuracy above 90%.

2.2.3 Hybrid Approaches

Lau and Ajoodha [55] compared Logistic Regression (LR), Random Forest (RF), Multilayer Perceptron (MLP), k-NN, and SVM with a CNN for categorizing musical genres. The features used in the study were content-based features that were extracted from raw audio signals. The results showed that traditional machine learning models produced similar results to CNN.

Ndou et al. [56] aimed at automatic music genre classification using deep learning and traditional machine-learning models. The work was conducted in three phases, and the results showed that the k-NN model provided the best accuracy at 92.69%, with a relatively low training time. The linear logistic regression and SVM also performed well, attaining accuracy above 80%. However, the CNN implementations provided relatively low accuracy. The authors concluded that traditional machine learning models tend to outperform deep-learning approaches in automatic music genre classification.

Bahuleyan [57] investigated music genre classification using audio set data from Youtube. The author proposed two approaches: the first uses a CNN-based image classifier trained on spectrogram images, and the second uses XGBoost trained on time and frequency domain features. The CNN-based approach performed better, and ensembling the CNN and XGBoost models improved performance further.

[7] https://paperswithcode.com/dataset/ballroom.

2.2.4 Ensemble Learning Approaches

In recent years, several studies have presented ensemble techniques for music genre classification to improve the classification's accuracy. Ensemble techniques combine the predictions of multiple individual classifiers to produce a more accurate final prediction. For example, Nascimento et al. [58] proposed the feature selection process with an ensemble approach. First, binary classifiers were used to complete the multi-classification task. Next, the outputs of the binary classifiers were integrated to produce the final music genre label. The Latin Music Database (LMD) [59] is the dataset utilized for this work, which has 3,227 musical compositions divided into ten musical genres. The classifiers employed were Naive Bayes [60], Decision Trees [61], SVM, and MLP. The ensemble method, as expected, generally outperformed the individual classifiers.

Sanden and Zhang [62] used ensemble techniques for multi-label genre classification. The testing data comprised 430 full-length music pieces from 16 world genres. Three assessment metrics were used: example-based, label-based, and rank-based. The third group drew ranking information from a score vector and conducted assessments accordingly, while the last two groups were based on bipartition vectors. For the ensemble, heterogeneous classifiers were used for training. Multi-label learning was performed using the Mulan [63] open-source library with the default settings. Mel-frequency Cepstral Coefficients (MFCC), Zero-crossing Rate (ZCR), Spectral Centroid, Rolloff, Spectral Flux, and Chroma were the criteria for categorizing content into different genres. Using heterogeneous ensemble approaches improved classification performance for the three datasets utilized in this experiment. However, when separate classifiers were required to be trained and merged, this performance increase came at the sacrifice of intrinsic computational cost.

The research by Chathuranga and Jayaratne [64] demonstrates a method for classifying music into different genres using an ensemble of classifiers. The authors proposed using multiple classifiers, such as k-NN, Decision Tree, RF, and Naive Bayes and combining their predictions to improve classification accuracy. The study found that the ensemble

approach outperformed individual classifiers in accuracy, precision, and recall. The results of both studies support the idea that ensemble techniques can effectively improve music genre classification.

Ricardo et al. [65] suggested a method for classifying music into different genres using a dynamic selection of an ensemble of classifiers. They use multiple classifiers and a selection algorithm to choose the best classifier for each song, depending on its characteristics. The study used the Latin Music Database and found that the dynamic selection ensemble approach outperformed traditional ensemble techniques and individual classifiers. The authors conclude that the dynamic selection of classifiers can effectively improve music genre classification, which is worth considering when developing a music genre classification system.

Leartpantulak and Kitjaidure [66] presented a method for music genre classification using Particle Swarm Optimization (PSO) [67] and the stacking ensemble technique. PSO optimizes a problem by iteratively adjusting a population of candidate solutions called particles. The authors proposed using PSO to optimize the ensemble of classifiers and a stacking ensemble technique to combine predictions of multiple classifiers. The study found that the proposed method performed better than traditional ensemble techniques and individual classifiers in accuracy. The results of this study demonstrate that combining PSO with the stacking ensemble technique can be an effective method for music genre classification, and it can be a promising approach for this task.

Nanni et al. [68] proposed a novel approach for music genre classification by combining deep learning, visual, and acoustic features in an ensemble framework. The authors proposed an ensemble of multiple components and classifiers to improve classification accuracy. The results showed that the proposed ensemble approach performed better than individual classifiers in accuracy and achieved state-of-the-art results. Furthermore, combining visual and acoustic features with deep learning classifiers showed promising results, and the authors suggest this could be a powerful approach for music genre classification.

Furthermore, the study supports the idea that ensemble techniques can effectively improve music genre classification, which is worth considering when developing a music genre classification system.

2.2.5 Other Approaches

Several other approaches have been proposed for music genre classification in recent years. For example, in a study by Meng et al. [69], the authors proposed a multivariate autoregressive feature model to address the difficulty of music genre categorization by using a framework that integrated multiple feature representations, including MFCCs, chroma features, and tempo information. To categorize music data, the authors used the Diagonal Autoregressive (DAR) and Multivariate Autoregressive (MVR) models as prediction models. Although the suggested MAR features outperformed the others, the computational complexity increased.

Another approach was suggested by Martins et al. [70]. In their study, the authors addressed the challenges of selecting a set of features for high genre classification accuracy in music and the need for a representative dataset of regional Brazilian music. They introduced a new set of features to classify genres of music and a new dataset called the Brazilian Music Dataset (BMD). The proposed characteristics were assessed on the BMD and the GTZAN datasets. The results indicated an average accuracy of 79.7% for GTZAN and 86.11% for the BMD.

Salazar [71] proposed a method for music genre classification using complex networks built from Mel-spectrogram features. The author used hierarchical mining at macro and micro levels and expanded feature vectors to include visual, auditory, and topological features. This study also used a logarithmic score to evaluate the performance. The study showed that the proposed method was reliable and effective in music genre classification.

Song et al. [72] suggested a semi-supervised content-based music genre classification technique. The authors dynamically combined several features in this study and used a

regularized least-square framework as a primary classifier. In addition, a distance-based technique that considers only similarity scores across songs as a measure of classification accuracy was used. An expectation-maximization approach was suggested to learn the fusion scores adaptively. The overall best-found accuracy for this study was 84.77%.

Silla and Freitas [73] presented two new top-down approaches for hierarchical classification. Their research evaluated several techniques using a dataset of more than 4,000 songs. An analysis of the experimental data showed that, in most cases, the novel approaches significantly enhance classification accuracy compared to the conventional top-down method. In addition, the proposed strategies benefited from dynamically selecting the best feature representation for each class node instead of using a single fixed feature representation throughout the whole class hierarchy.

Recently, Nguyen et al. [74] proposed a new technique for classifying music genres using the Residual Attention Network (RAN). The authors demonstrate that by incorporating attention mechanisms and stacking attention modules, RAN can be effectively utilized for image processing. To apply this to music genre classification, the method converts audio files into spectral images, which are then classified using RAN. The technique's performance was evaluated on a dataset containing recordings of 10 different musical genres and yielded an accuracy rate of over 70%.

2.2.6 Summary

The studies discussed in this chapter have outlined the effectiveness of various approaches and methods for music genre classification, including machine learning, deep learning, and ensemble learning. While traditional machine learning models have shown good performance, their accuracy can be further improved. Furthermore, deep learning techniques have gained popularity recently, with various features extracted from audio recordings and algorithms employed to enhance genre classification accuracy. Moreover, ensemble techniques have also been widely used, combining the predictions made by multi-

ple classifiers to improve classification accuracy. However, a potential limitation of ensemble techniques is that they do not fully utilize the dataset's potential since they often neglect the importance of the original features from the dataset while training the meta-models, for instance, in the stacking ensemble learning technique where a meta-model is trained on the combination of the predictions of the individual base models, focusing on the outputs of the base models rather than the original features. To address this limitation, we propose the augmented features ensemble technique in this study that extends the dataset's original features by adding additional features based on the prediction probabilities from the base classifiers, potentially leading to better performance and highlighting that it is essential to incorporate both the original features and the predictions made by the base classifiers in ensemble techniques to enhance the accuracy of music genre classification.

Chapter 3

Data Preparation and Exploration

The purpose of this chapter is to describe the details of the datasets used in our study for genre classification. The three datasets selected are Spotify[8], TCC_CED[9], and GTZAN[10], with each carrying unique features and characteristics. Before discussing the data preprocessing steps, we will first provide an overview of the sources and features of the datasets. The Spotify dataset consists of information from the audio signals, including popularity, danceability, and speechiness. The TCC_CED dataset consists of music features in different genres, including Rock, Pop, Hip Hop, and Jazz. It also contains information about the music pieces similar to the Spotify dataset, such as topic, sadness, and acousticness. Finally, the GTZAN dataset, widely used in music genre classification, contains ten distinct music genres and incorporates several features such as Spectral Centroid, Spectral Rolloff, and Mel-Frequency Cepstral Coefficients (MFCCs). We used several criteria to select these datasets, such as availability, variety of genres, and variation in size. After selecting the datasets, we proceeded to the data preprocessing phase, which involved several steps:

- We picked the top genres from each dataset based on their frequency of occurrence to avoid the underrepresented and minority classes.
- We performed feature conversion, which involved converting the features from one format to another, to make them suitable for our analysis. Then, we repositioned the target feature to the beginning of the dataset to make it easier to separate the

[8] https://www.kaggle.com/datasets/zaheenhamidani/ultimate-spotify-tracks-db.

[9] https://data.mendeley.com/datasets/3t9vbwxgr5/1.

[10] https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification?resource=download.

features during the classification process. After that, we conducted feature separation, separating the features into independent and target variables.

- To ensure that the data were suitable for our study, we performed encoding and scaling of the features. The encoding process involves converting categorical target variables into numerical values, while the scaling process involves transforming the values of the features so they have the same scale.

Finally, we used five feature selection methods to select the most relevant features: Recursive Feature Elimination, tree-based, forward, backward, and bidirectional feature selections. These methods helped us identify the most relevant features for genre classification and reduced the data dimensions, which could improve the model's accuracy.

3.1 Datasets, Features, and Genres

Three different datasets were picked to get generalized results. Using multiple datasets for training and testing a classification model can help ensure the model can generalize well to new data rather than perform well on a specific dataset. This can improve the robustness and reliability of the model. Additionally, working with different datasets can help identify potential limitations or biases in the models and provide a more comprehensive evaluation of their performance. We selected a large ($N = 232,725$), a medium ($N = 28,372$), and a small dataset ($N = 1,000$), where N is the number of music pieces in a dataset, for a more comprehensive evaluation of the models and to identify their strengths and drawbacks.

Using a large dataset can help to ensure that the model has enough data to learn from and can help to improve the accuracy of the model by reducing the chance of overfitting. On the other hand, working with a medium-sized dataset can help to balance the trade-off between having enough data to learn from and having a manageable amount of data for testing. In addition, using a small dataset can help identify potential limitations or weaknesses of the model and provide a quick evaluation of the model's performance. It can also help test the model in resource-constrained environments or quickly test new ideas.

3.1.1 Spotify

Spotify includes 232,725 music pieces and 15 features for each piece. The features included in this dataset are presented in Table 3.1. The Spotify dataset originally incorporated information about *"artist_name"*, *"track_name"*, and *"track_id"*. However, these three features were excluded, so the dataset only contains features about the music pieces' audio characteristics. Except for *"mode"* and *"key"*, which are categorical features, all other features in the Spotify dataset are numerical.

Table 3.1: Music features in the Spotify dataset.

genre	popularity	acousticness	danceability	duration_ms	energy
instrumentalness	key	liveness	loudness	mode	speechiness
tempo	time_signature	valence			

There are also 26 unique genres in this dataset, as shown in Table 3.2.

Table 3.2: Music genres in the Spotify dataset.

Movie	R&B	A Capella	Alternative	Country	Dance	Electronic
Anime	Folk	Blues	Opera	Hip Hop	Children's Music	Rap
Indie	Classical	Pop	Reggae	Reggaeton	Jazz	Rock
Ska	Comedy	Soul	Soundtrack	World		

Figure 3.1 displays the correlation among the features in the Spotify dataset, obtained through the Spearman Correlation method [75], revealing a generally low correlation among the features and between the features and the music genre, with the exception of a few cases such as (loudness, acousticness), (energy, acousticness), and (energy, loudness). This may be attributed to the fact that the features in the Spotify dataset are primarily high-level or global features that focus on the sentiment of the music pieces.

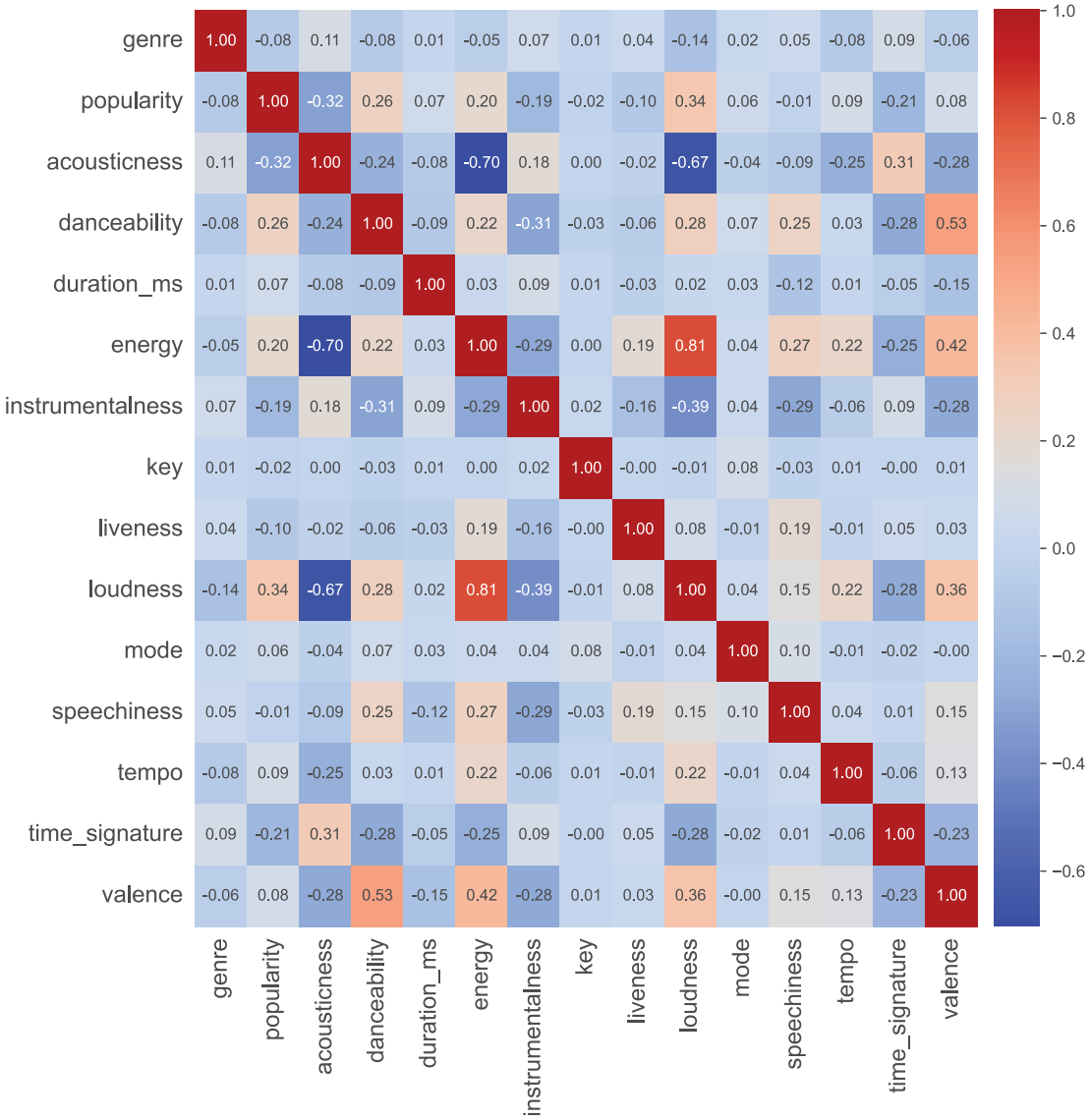

Figure 3.1: Feature correlations within the Spotify dataset.

3.1.2 TCC_CED

The TCC_CED dataset includes 28,372 music pieces and 26 features. Table 3.3 shows the features in this dataset. Similar to the Spotify dataset, this dataset initially included other features such as *"Unnamed: 0"*, *"artist_name"*, *"track_name"*, *"release_date"*, and *"lyrics"*, which were dropped as in case of the Spotify dataset. In the TCC_CED dataset, all features are numerical except for the *"topic"* feature, which is categorical. As with the Spotify dataset, this distinction is crucial for properly processing and analyzing the data.

Table 3.3: Music features in the TCC_CED dataset.

genre	len	dating	violence
night/time	shake the audience	family/gospel	romantic
obscene	music	movement/places	light/visual perceptions
like/girls	sadness	feelings	danceability
acousticness	instrumentalness	valence	energy
world/life	communication	family/spiritual	loudness
topic	age		

The genres in this dataset are given in Table 3.4.

Table 3.4: Music genres in the TCC_CED dataset.

Pop	Country	Blues	Jazz	Reggae	Rock	Hip Hop

Figure 3.2 also shows the correlation among the features in the TCC_CED dataset, indicating a generally low correlation between the features and the music genre. Similar to the Spotify dataset, the TCC_CED dataset incorporates high-level features that mainly capture a song's sentiment and mood.

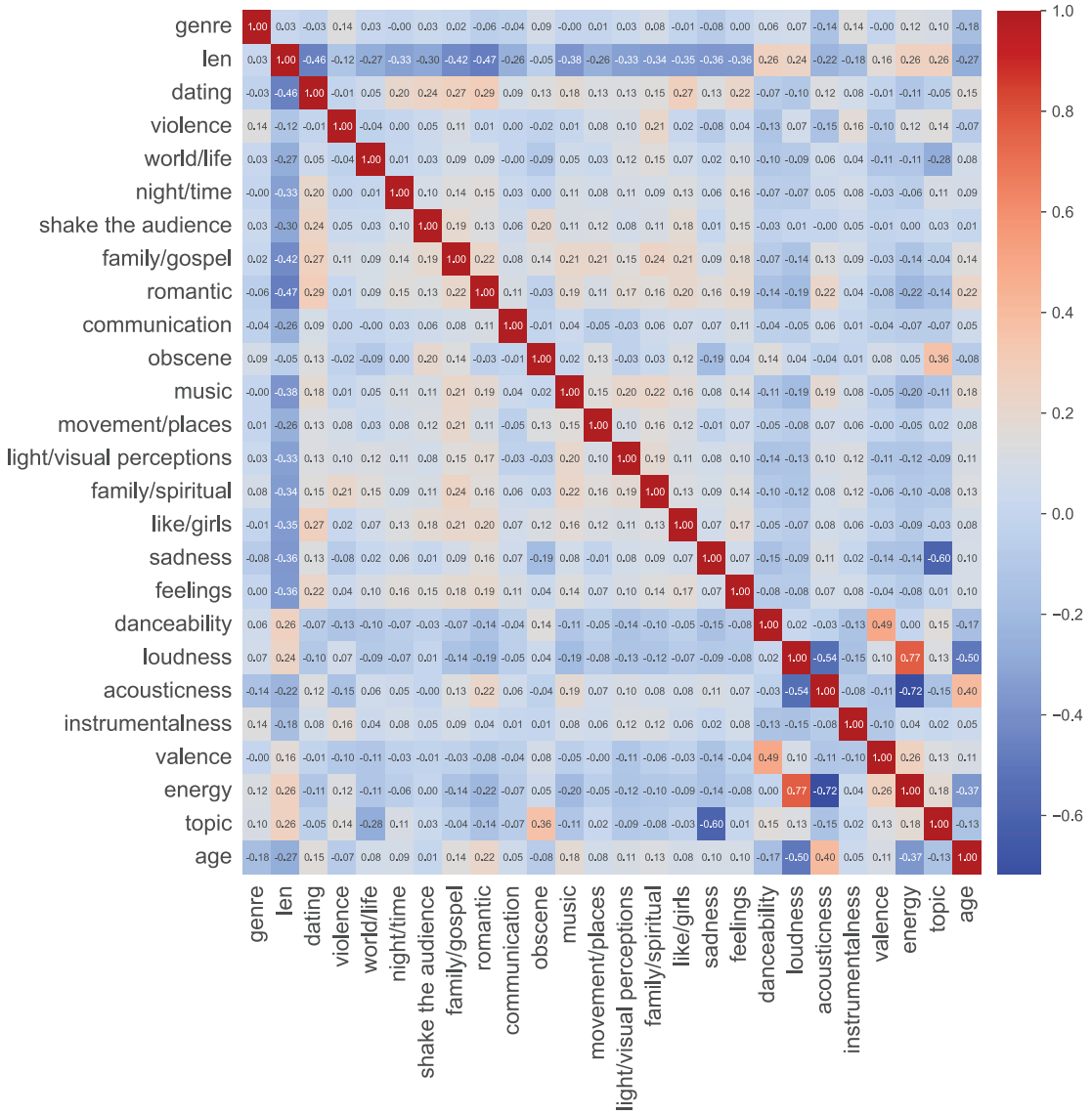

Figure 3.2: Feature correlations within the TCC_CED dataset.

3.1.3 GTZAN

The GTZAN dataset includes information about 1,000 music pieces with ten different genres. This dataset has 27 features overall, shown in Table 3.5. All the features in the GTZAN dataset are numerical, and no features were dropped during preprocessing. The genres represented in the dataset are also indicated in Table 3.6. This dataset serves as a benchmark for various music classification tasks, allowing practitioners to evaluate the performance of different machine learning algorithms on a diverse set of music genres.

Table 3.5: Music features in the GTZAN dataset.

chroma_stft	rmse	rolloff	spectral_centroid	spectral_bandwidth	zero_crossing_rate
mfcc1	mfcc2	mfcc3	mfcc4	mfcc5	mfcc6
mfcc7	mfcc8	mfcc9	mfcc10	mfcc11	mfcc12
mfcc13	mfcc14	mfcc15	mfcc16	mfcc17	mfcc18
mfcc19	mfcc20	genre			

Table 3.6: Music genres in the GTZAN dataset.

Blues	Classical	Country	Disco	Hip Hop	Jazz	Metal	Pop	Reggae	Rock

Figure 3.3 illustrates the correlation among the features in the GTZAN dataset. In contrast to the Spotify and TCC_CED datasets, which include high-level sentiment features, the GTZAN dataset contains acoustic features extracted from music signals that are highly correlated with each other and with the music genres.

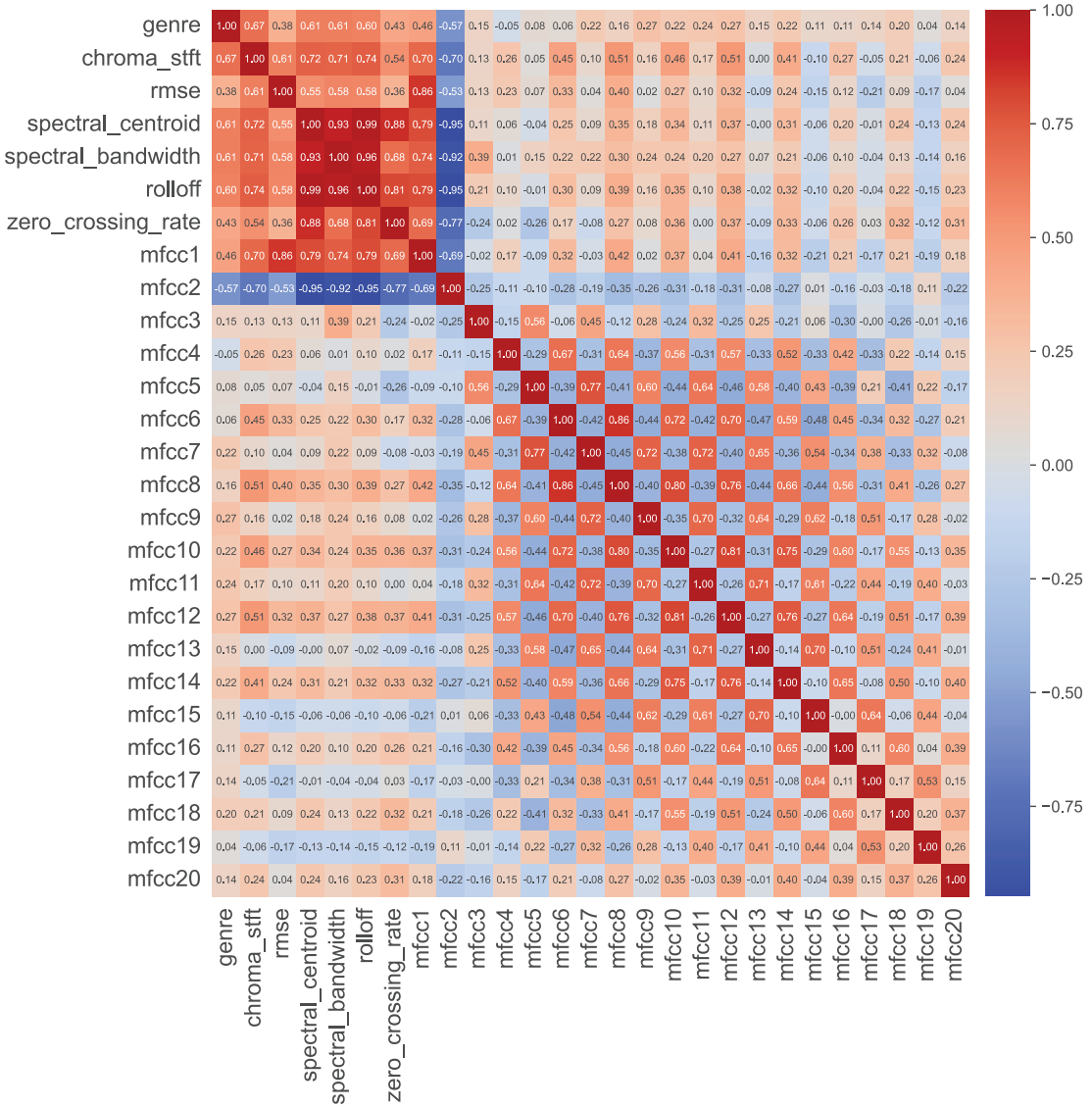

Figure 3.3: Feature correlations within the GTZAN dataset.

As mentioned, one important aspect of the GTZAN dataset is that it contains some standard acoustic features in MIR, providing an essential representation of each track's sound and timbre and allowing researchers to understand the underlying characteristics of different music genres. We will discuss the features of the GTZAN dataset in detail below.

Chroma Short-Time Fourier Transform

Chroma Short-Time Fourier Transform (Chroma STFT) is a signal-processing technique commonly used in music analysis to extract musical features from audio signals. The Chroma STFT converts the audio signal into a chromagram ("a representation of spectral audio information mapped into one octave" [76]), defining the signal in the musical pitch space. This representation highlights the signal's harmonic content [77] and de-emphasizes the differences in timbre and rhythm. Researchers widely use the chromagram in music information retrieval tasks, such as chord recognition, key detection, and melody extraction. The short-time Fourier transform (STFT) $X(t,f)$ [78] is defined as:

$$X(t,f) = \int_{-\infty}^{\infty} x(\tau) w(\tau - t) e^{-2\pi i f \tau} d\tau$$

where $x(\tau)$ is the original audio signal, $w(\tau - t)$ is the window function, t is the center time of the window, f is the frequency, and $e^{-2\pi i f \tau}$ is the complex exponential, which represents a sinusoidal wave of frequency f.

The chroma STFT of the signal is then obtained by mapping the STFT coefficients to a chroma representation, $C(t,k)$, where k represents the musical pitch class. This mapping is typically done by summing the magnitudes of the STFT coefficients within a specific frequency range corresponding to each pitch class. The resulting chroma matrix is then normalized to obtain the final chromagram, which is a 2D matrix with dimensions (T,K):

$$C(t,k) = \frac{\sum_{f \in F_k} |X(t,f)|}{\sum_{f \in F} |X(t,f)|}$$

where F_k is the set of frequencies corresponding to pitch class k, and F is the set of all frequencies.

Specrtal Rolloff

Spectral Rolloff is an acoustic feature used in MIR to describe the shape of the power spectrum of an audio signal, which is the frequency below a certain percentage (usually 85%) of the total spectral energy [79]. Spectral Rolloff is a simple but effective feature that captures some fundamental characteristics of an audio signal, such as its overall tonality and brightness.

$$f_{\text{rolloff}} = \arg\min_{f}(f \mid \frac{\sum_{i=0}^{f} P(i)}{\sum_{i=0}^{f_{max}} P(i)} \geq c)$$

where $P(f)$ is the power spectrum of the audio signal, f_{max} is the highest frequency of interest, f_{rolloff} is the Spectral Rolloff point, and c is the threshold value.

Spectral Centroid

The Spectral Centroid represents the balance between low and high-frequency components and is defined as the weighted mean of the frequency spectrum of a signal [80].

$$C = \frac{\sum_{i=0}^{f_{max}} f_i P(f_i)}{\sum_{i=0}^{f_{max}} P(f_i)}$$

where f_i is the frequency of the i^{th} bin in the power spectrum, a representation of the power of each frequency component of a signal, $P(f_i)$ is the power of the i^{th} frequency interval, and f_{max} is the highest frequency of interest.

Spectral Bandwidth

The Spectral Bandwidth is a measure of the spread of the power spectrum of a sound, which demonstrates the range of frequency components that contribute significantly to the audio. It summarizes the distribution of energy across different frequencies. It is expressed as the difference between the upper and lower frequencies at which a certain fraction of the total power of the spectrum is contained:

$$B = f_{\text{high}} - f_{\text{low}}$$

where f_{high} is the highest frequency and f_{low} is the lowest frequency.

Zero Crossing Rate

The Zero Crossing Rate (ZCR) is a practical feature for characterizing audio signals, especially speech and music signals. ZCR identifies the presence of transient or percussive sounds in audio signals by measuring the rate at which the signal crosses the zero value in the time domain [81]. The ZCR is the number of times the audio signal changes sign per unit or the number of zero crossings in a given time window. Mathematically, the ZCR can be expressed as:

$$\text{ZCR} = \frac{1}{N} \sum_{n=0}^{N-1} [x[n] \cdot x[n+1] < 0]$$

where $x[n]$ is the audio signal at time index n, N is the number of samples in the window, and $[x[n] \cdot x[n+1] < 0]$ is a binary indicator that returns one if the product of two consecutive samples is negative and 0 otherwise.

Mel-Frequency Cepstral Coefficients

Mel-Frequency Cepstral Coefficients (MFCCs) are a way to represent the spectral envelope of a sound signal using a set of coefficients. The process of computing MFCCs involves dividing the audio signal into overlapping frames, windowing each frame, transforming the signal into the frequency domain, converting the power spectrum (representing the power of each frequency component of a signal on a linear scale) to a non-linear scale, taking the logarithm, and finally transforming the result into the cepstral domain [82]. The resulting MFCCs capture the shape of the spectral envelope compactly and efficiently. MFCCs are widely used in MIR and speech processing for various tasks such as music

classification and speech recognition.

$$\text{MFCC}_k = \sum_{n=0}^{N-1} w(n) log(E(n)) cos(\frac{\pi k}{N}(n - \frac{N-1}{2}))$$

where N is the number of frequency bins, $w(n)$ is the triangular window function, $log(E(n))$ is the logarithm of the energy in each frequency bin, and k is the coefficient index ranging from 1 to N.

3.2 Data Preprocessing

Data preprocessing is necessary for machine learning as it helps ensure the data is properly formatted and free from errors and inconsistencies. Furthermore, this process can significantly impact the model's performance and accuracy, as it helps prepare the data for training algorithms. The various steps involved in data preprocessing include cleaning, transforming, and normalizing the data.

Cleaning the data is crucial because it removes, corrects, or updates any missing or incorrect values. This step helps to minimize the potential impact of outliers or irrelevant data on the model's performance. On the other hand, data transformation involves scaling or encoding the data to make it more suitable for training algorithms. It helps to ensure that the data is of uniform size and type, making it easier for the model to learn from. As the name suggests, data normalization transforms the data to conform to a standard normal distribution, ensuring that all features are on a comparable scale [83]. For example, data normalization is essential in algorithms that use distance metrics, such as k-Nearest Neighbors [9], as it helps to eliminate the impact of features with larger magnitudes.

Feature engineering is another technique that can be employed in data preprocessing [84], which involves combining different columns or generating synthetic data. Feature engineering helps to provide the model with more relevant information, improving its performance. This section briefly discusses the techniques we used to preprocess our data for

our study.

3.2.1 Selecting the Top Genres

Music classification is a complex task involving categorizing music pieces into various genres. With the wide variety of music genres, it can become difficult to classify them accurately. To overcome this challenge in our study, we focused on a smaller set of popular and common genres, which are easier to manage and produce robust results. Doing so also helps to avoid the problem of minority classes where a few genres dominate the dataset and make it difficult to classify the other genres accurately. Selecting the top genres based on popularity can make the classification results more interpretable and practical. This is because a smaller set of genres commonly understood across multiple platforms, such as Blues, Country, Electronic, Pop, Metal, Hip Hop, and Disco, can provide better insights and help users better understand the results [85]. Table 3.7 illustrates the top five genres selected for each dataset based on their frequency of occurrence, and their distribution is demonstrated in Figure 3.4. Focusing on these popular genres makes the classification task more manageable, and the results are easier to interpret and more functional.

Table 3.7: Top five music genres in the datasets.

Dataset	**Top Genres**
Spotify	Indie, Pop, Jazz, Comedy, Soundtrack
TCC_CED	Pop, Country, Blues, Jazz, Rock
GTZAN	Blues, Classical, Country, Disco, Hip Hop

After choosing the top five genres in each dataset, the number of music records was reduced to ($N = 47,697$) in Spotify, ($N = 24,970$) in TCC_CED, and ($N = 500$) in GTZAN.

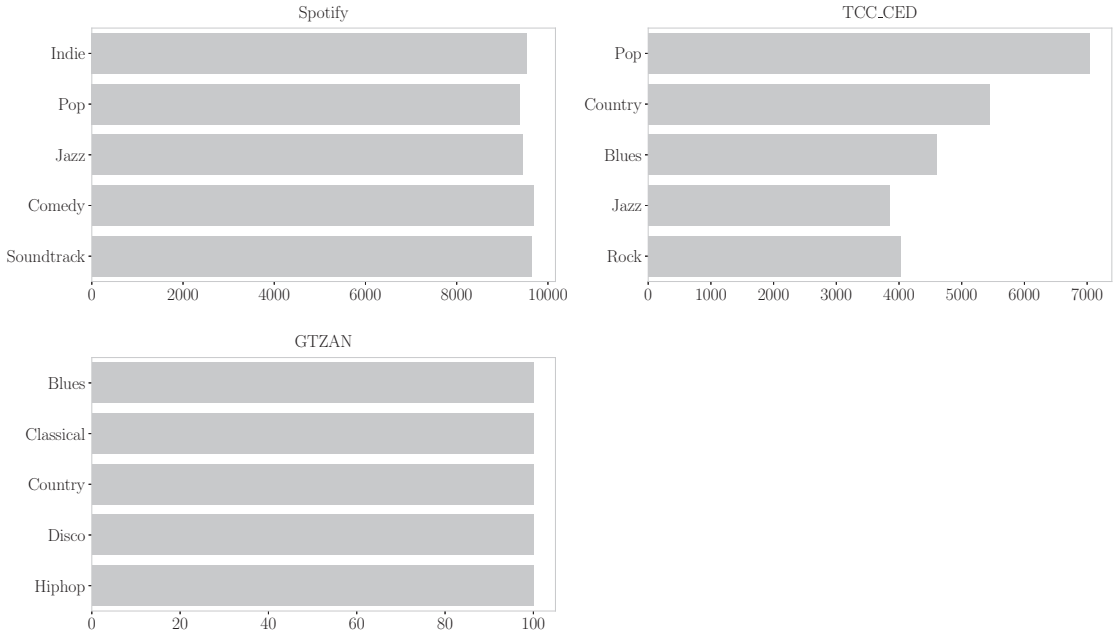

Figure 3.4: Distribution of top five music genres in each dataset.

3.2.2 Feature Conversion

Feature conversion is required because many machine learning algorithms need input data to be numerical to work precisely. By converting non-numeric features to numeric ones, the data can be used to train and make predictions with these algorithms. In this step, we converted all non-numeric values in the independent variables to numeric ones by assigning a unique integer value. Genre information was excluded from this process since it is a label, not a feature, and will be converted in the following steps.

3.2.3 Repositioning the Target Feature

For several reasons, it is essential to reposition the target variable in a dataset. One of the main reasons is that many machine learning libraries expect the target variable to be in the first column of the input data. Another reason is that it improves the readability and understandability of the data. When working with large and complex datasets, it is crucial to have a clear and intuitive structure that allows users to identify and understand the different columns and their meanings quickly. Placing the target variable in the first column instantly

makes it obvious which column represents the outcome the model is trying to predict. In this step, we re-index the data by withdrawing the genre column from the dataset and reinserting it back into the dataset at the first position (index 0).

3.2.4 Feature Separation

Next, we isolated the feature (X) and target variable (y) and treated them as separate objects. Feature separation allows independent variables to be transformed, selected, or extracted and for the target variable to be encoded as necessary for the specific learning algorithms. Additionally, it allows a more convenient preprocessing, evaluation, and adequate interpretability of the results.

3.2.5 Encoding the Target Feature

LabelEncoder[11] class was used to transform the genres into numerical data. This step is necessary because many machine learning algorithms can only operate on numerical target labels. Encoding the genres allows machine learning algorithms to work with categorical data by converting it into numerical data. In our study, we first utilized the *fit()* method of the LabelEncoder class for encoding categorical target labels into numerical labels. This step is necessary to learn the mapping between the original categorical labels and the numerical labels. Next, we applied the computed label encoding to the target data using the *transform()* method from the LabelEncoder class, which replaces each categorical label with its corresponding numerical label, producing the encoded data.

3.2.6 Scaling the Features

Finally, we utilized *StandardScaler*[12] class to scale our data, which is a preprocessing technique that scales the data to fit a standard normal distribution. Scaling the data is necessary because many machine learning algorithms, such as k-Nearest Neighbors, assume

[11] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html`.

[12] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`.

that the input data is standardized and will perform better if this assumption is met. In addition, scaling the data is required because some machine learning algorithms are acute to the scale of the input data. With scaling, these algorithms can avoid the possible large numbers or outliers in the data and may perform better. StandardScaler addresses this issue by transforming the data with a mean of zero and a standard deviation of one. The following equation gives the standardization:

$$x_i' = \frac{x_i - \text{mean}(x)}{\text{std}(x)}$$

where $mean(x)$ is the mean and $std(x)$ is the standard deviation of the feature values, x_i is a feature value, and x_i' is the standardized feature value.

In our research, we first employed the *fit()* method of the StandardScaler class to compute each feature's mean and standard deviation in the input data. By doing so, we calculated the scaling parameters required for transforming the data. Next, we applied the computed mean and standard deviation to standardize the input data using the *transform()* method. As explained in the above frame, this method subtracts the mean value of each feature from the data and then divides it by the standard deviation of that feature, producing the standardized data.

Preprocessing techniques such as feature scaling and encoding can introduce data leakage if applied to the entire dataset before splitting it into the train and test sets because the information from the test set can leak into the training set, leading to overfitting and inaccurate performance estimates. To avoid this issue, the preprocessing transformers are normally fitted on the training set and then used separately to transform the training and testing sets.

3.3 Feature Selection

Repeated, redundant data and unrelated features can reduce a classifier's generalization ability and decrease the classifier's overall accuracy. Additionally, a model's complexity increases when more variables are added. Feature selection [86] is one of the most critical techniques in machine learning. The most relevant and significant features are selected using different methods such as tree-based feature selection [87], Recursive Feature Elimination (RFE) [88], and feature selection using wrapper methods [89]. A combination of feature selection approaches provides a robust and comprehensive approach to evaluating the performance of models with different subsets of features. In our study, we employed RFE, tree-based feature selection, and forward, backward, and bidirectional feature selection techniques, which are all commonly used for selecting important features in machine learning.

3.3.1 Recursive Feature Elimination

RFE is a feature selection technique that aims to select the most informative features of the data [90]. Given an external estimator that assigns weights to features, RFE chooses features by examining smaller sets of features iteratively. The importance of each feature is then calculated using the feature importance characteristics once the estimator has been trained on the initial set of features. Finally, the least important features are removed, given the current list of features. This method is applied iteratively to the pruned set until the required number of features is reached. One key benefit of using RFE is that it can significantly reduce the dimensionality of the data when working with a large number of features. It can also improve model performance and reduce overfitting.

In our study, we employed the Random Forest classifier as the external estimator to evaluate the importance of features and select the top 15 features in each dataset that were most relevant to the genre classification task to strike a balance between model complexity and performance. The features chosen in each dataset after applying RFE are shown in

Table 3.8.

Table 3.8: Selected features in each dataset using RFE.

Dataset	Selected Features
Spotify	popularity, acousticness, danceability, duration_ms, energy instrumentalness, key, liveness, loudness, mode speechiness, tempo, time_signature, valence
TCC_CED	len, violence, world/life, night/time, communication music, movement/places, sadness, danceability, loudness acousticness, instrumentalness, valence, energy, age
GTZAN	chroma_stft, rmse, spectral_centroid, spectral_bandwidth, rolloff zero_crossing_rate, mfcc1, mfcc2, mfcc4, mfcc5 mfcc6, mfcc11, mfcc12, mfcc15, mfcc17

3.3.2 Tree-based Feature Selection

One of the benefits of using tree-based algorithms [87], such as Random Forest and Gradient Boosting [91], is their interpretability. They create a series of simple decision trees that can be easily understood and visualized, making it more convenient to understand how the algorithm makes predictions and which features are most important. The feature selection is achieved through a feature importance attribute in tree-based algorithms. The feature importance attribute assigns a weight or score to each feature in the data based on Gini impurity, representing how much each feature contributes to the algorithm's decision-making process. This information can is used to identify the most important features of the problem for feature selection and dimensionality reduction. Higher scores indicate that the feature is more important or relevant for the outcome. Removing less important features allows the model to be simplified and made more efficient while maintaining high accuracy. This makes tree-based algorithms particularly useful for feature selection, as it is easy to

determine which features are most important for the problem. After applying the tree-based algorithm, each dataset's top 15 highly correlated features are presented in Table 3.9.

Table 3.9: Selected features in each dataset using the tree-based feature selection.

Dataset	**Selected Features**
Spotify	popularity, speechiness, instrumentalness, danceability, liveness valence, acousticness, loudness, energy duration_ms, tempo, key, time_signature, mode
TCC_CED	instrumentalness, acousticness, age, energy, danceability len, loudness, valence, violence movement/places, world/life, dating, sadness communication, obscene
GTZAN	chroma_stft, rmse, mfcc1, spectral_bandwidth, rolloff spectral_centroid, mfcc2, mfcc17, mfcc4, mfcc6 zero_crossing_rate, mfcc11, mfcc5, mfcc8, mfcc3

3.3.3 Froward Feature Selection

Forward feature selection [92] is a greedy search algorithm that starts with an empty set of features and iteratively adds the feature that results in the best performance until no further improvement can be made. The algorithm starts with an empty set of features and evaluates the performance of each remaining feature using a statistical test, typically a t-test [93], or ANOVA [94], with a significance level of ($p - value = 0.05$). Then, the feature that results in the best performance is added to the set of selected features, and the process is repeated until no further improvement can be made. In our study, we assessed the importance of features by fitting a linear regression model in the forward feature selection algorithm. The t-test was utilized to calculate the $p - value$ for each additional feature in

the linear regression model. The null hypothesis for the t-test is that the additional feature does not affect the target variable (i.e., its coefficient in the linear regression model is zero), while the alternative hypothesis is that the additional feature has a non-zero effect on the target variable. If the $p-value$ is below the significance level, the null hypothesis is rejected, and the additional feature is considered significant and added to the list of selected features. Otherwise, the alternative hypothesis is accepted, and the additional feature is not selected. This method is computationally expensive as it requires fitting the model multiple times, once for each feature addition. However, it can benefit datasets with many features, allowing the selection of a subset of the most informative features for the target variable. The selected features by this technique are shown in Table 3.10.

Table 3.10: Selected features in each dataset using forward feature selection.

Dataset	**Selected Features**
Spotify	time_signature, loudness, danceability, energy, instrumentalness popularity, liveness, speechiness, tempo, mode duration_ms, acousticness, key
TCC_CED	instrumentalness, violence, danceability, age, len energy, loudness, valence, acousticness communication, family/spiritual, movement/places, sadness like/girls, dating
GTZAN	chroma_stft, mfcc6, spectral_bandwidth, mfcc18, mfcc4 mfcc12, zero_crossing_rate, spectral_centroid, rolloff

3.3.4 Backward Feature Selection

Backward feature selection is a method that starts with all the features and gradually eliminates the ones that result in a minor decrease in performance. This approach is oppo-

site to the forward feature selection method, where features are incrementally added to the model. In this study, we used a criterion of $p - value = 0.05$ to determine the significance of the features, just like in the forward feature selection method. The final set of features selected using this method is shown in Table 3.11. The backward feature selection method begins with all features, then systematically removes features one by one until no further improvement in performance can be achieved. This approach allows us to determine the most important features contributing to the model's prediction performance.

Table 3.11: Selected features in each dataset using backward feature selection.

Dataset	**Selected Features**
Spotify	popularity, acousticness, danceability, duration_ms, energy instrumentalness, key, liveness, loudness, mode speechiness, tempo, time_signature
TCC_CED	len, dating, world/life, night/time, shake the audience family/gospel, romantic, communication, obscene music, movement/places, light/visual perceptions, like/girls sadness, feelings, danceability, loudness, acousticness instrumentalness, valence, energy, topic, age
GTZAN	chroma_stft, spectral_centroid, rolloff, zero_crossing_rate, mfcc4 mfcc5, mfcc6, mfcc7, mfcc11, mfcc12 mfcc14, mfcc16, mfcc17, mfcc18

3.3.5 Bidirectional Feature Selection

Bidirectional feature selection is a combination of both forward and backward feature selection. It starts by selecting a subset of features using a forward feature selection algorithm, and then it iteratively removes features using a backward feature selection algorithm [95]. This method allows assigning a subset of features that provide satisfactory

performance and are small in size. As a result, it has often been considered a more robust feature selection method than forward and backward feature selection, as it considers both the addition and removal of features. The selected features are shown in Table 3.12:

Table 3.12: Selected features in each dataset using bidirectional feature selection.

Dataset	**Selected Features**
Spotify	time_signature, loudness, danceability, instrumentalness, popularity liveness, speechiness, tempo, mode, acousticness duration_ms, key, energy
TCC_CED	instrumentalness, violence, danceability, age, len energy, loudness, valence, acousticness, communication family/spiritual, movement/places, sadness, like/girls, dating
GTZAN	chroma_stft, mfcc6, mfcc18, mfcc4, mfcc12 zero_crossing_rate, spectral_centroid, rolloff

3.3.6 Summary

In this chapter, we provided an in-depth analysis of the datasets used for genre classification in our study, which included Spotify, TCC_CED, and GTZAN datasets. We also presented the music features and genres within each dataset and provided a brief overview of the acoustic features in the GTZAN dataset.

To ensure that our analysis was thorough, we discussed the data preprocessing steps in detail, which involved selecting top genres to avoid the minority classes, converting categorical input features to the numerical format, repositioning the target feature for better readability, separating the input features and target variable (genre), scaling the input features to follow a standard normal distribution format, and encoding the target feature to convert genres into numerical formats. Furthermore, we discussed the use of five feature selection techniques in this study and identified the most important features selected by

these techniques for each dataset. By following these procedures, we aimed to ensure that our analysis was comprehensive and rigorous, thus enabling a robust evaluation of the results.

Chapter 4

A Novel Ensemble Learning through Augmented Features

In this chapter, we will introduce our proposed ensemble learning technique to increase the prediction accuracy of music genre classification. We will also discuss various machine learning classifiers and ensemble learning models that were utilized. First, we will introduce the base machine learning models applied in the ensemble learning methods, including the Multi-layer Perceptron, Random Forest, XGBoost, LightGBM, and K-Nearest Neighbors classifiers. We will begin by explaining each of these base models' key features and principles, including the algorithms used. These provide a solid foundation for understanding how the ensemble learning methods employed in this study utilized these base models.

Next, we will explore ensemble learning, which combines the predictions of multiple models to produce more accurate results than any individual base learner could achieve. We will examine popular ensemble learning methods such as voting, blending, and stacking and explore how they were applied in our study. To ensure a fair comparison and eliminate any potential variability, we decided to implement the blending and stacking ensembles ourselves rather than relying on built-in libraries, which allow us to run all models under the same environment. Additionally, we will discuss our proposed augmented features model, which benefits from a set of additional features incorporated into the ensemble learning model to improve its prediction performance. Finally, we will cover the performance validation method used to evaluate our study's results, including a detailed explanation of the

metric used to measure each model's performance and the measures employed to determine the best-performing model.

4.1 Classification Techniques Employed

To select the best-performing base classifiers based on our datasets, we employed the Lazypredict[13] library to compare the performance of various classifiers on each dataset and select a combination of the most accurate ones. Lazypredict is a Python [96] library that automates the training process and compares various machine-learning models. It allows for efficient training and evaluates different classifiers on datasets. The selected classification models were Multi-layer Perceptron (MLP), Random Forest, XGBoost, LightGBM, and k-Nearest Neighbors (k-NN) classifiers.

MLP is an effective model that can learn complex, non-linear relationships in the input data, making it well-suited for music genre classification tasks. Random Forest classifier merges multiple decision trees to create a more robust model that benefits music genre classification tasks. In addition, it can help reduce the overfitting problem commonly associated with individual decision trees. XGBoost and LightGBM are gradient-boosting algorithms known for their high accuracy and speed. They can also manage large datasets and categorical variables, which is crucial for music genre classification tasks. Gradient-boosting algorithms build a sequence of prediction models, such as decision trees, where each subsequent model learns from the errors made by the previous model in the sequence. k-NN is a non-parametric method used for classification and regression, which measures the distance between the input and training data points. k-NN can handle high-dimensional and non-linear data, which makes it a practical algorithm for working with music features. For example, music data often has many different features, such as tempo, pitch, timbre, and rhythm, and k-NN can measure the distances between these features to determine which music pieces are similar. It can also capture the input data's underlying structure, which

[13] `https://github.com/shankarpandala/lazypredict`.

helps to determine patterns and relationships within the music data. This section discusses the classifiers used in this study comprehensively.

4.1.1 Multi-layer Perceptron Classifier

MLP [97] is an artificial neural network composed of an input layer, one or more hidden layers, and an output layer. Figure 4.1 demonstrates an example structure of an MLP classifier with a single intermediate layer and five perceptions.

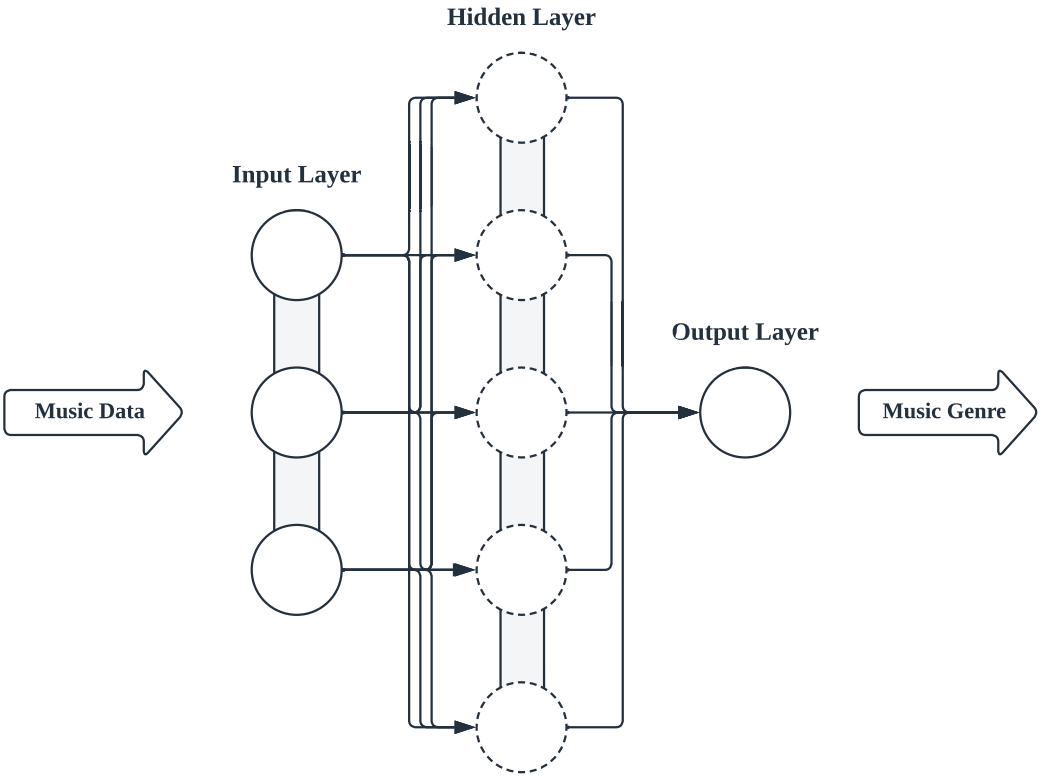

Figure 4.1: Example layout of an MLP classifier.

The input layer receives the input data, which is then processed by the perceptrons in the hidden and output layers [98]. The hidden layers use activation functions to convert the input signal, and the output layer assembles the final result. The weights and biases are adjusted during the training stage using a method such as backpropagation [99] to minimize

the error between the predicted and actual output. The activation functions in the hidden layers of an MLP are typically non-linear functions such as *"sigmoid"*, *"tanh - hyperbolic tangent"*, or *"ReLU - Rectified Linear Unit"*. The purpose of the activation function is to introduce non-linearity into the model, allowing the MLP to learn complex non-linear relationships in the data [100]:

- The sigmoid activation function is a commonly used activation function that maps the input to a value between 0 and 1:

$$f(x) = \frac{1}{1+e^{-x}}$$

- The tanh activation function is similar to the sigmoid function but maps the input to a value between -1 and 1:

$$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

- The ReLU activation function is a non-linear function that maps the input to zero if it is negative and to the input value if it is positive:

$$f(x) = \max(0,x)$$

4.1.2 Random Forest Classifier

Random Forest (RF) is another supervised learning model that integrates multiple decision trees to create a robust model for many tasks [101, 102, 103]. The idea behind Random Forest is to develop many decision trees on different subsets of the training data and then average or vote their predictions to make the final decision [104]. Figure 4.2 exemplifies a Random Forest classifier flow diagram. The use of multiple decision trees helps to reduce the overfitting problem that is typically associated with individual decision trees. It randomly selects a subset of the features and training data to construct each decision tree. This process is repeated numerous times, creating a forest of decision trees. During the

prediction phase, each tree in the forest makes a prediction, and the final output is based on a majority vote or the average of the predictions from different trees.

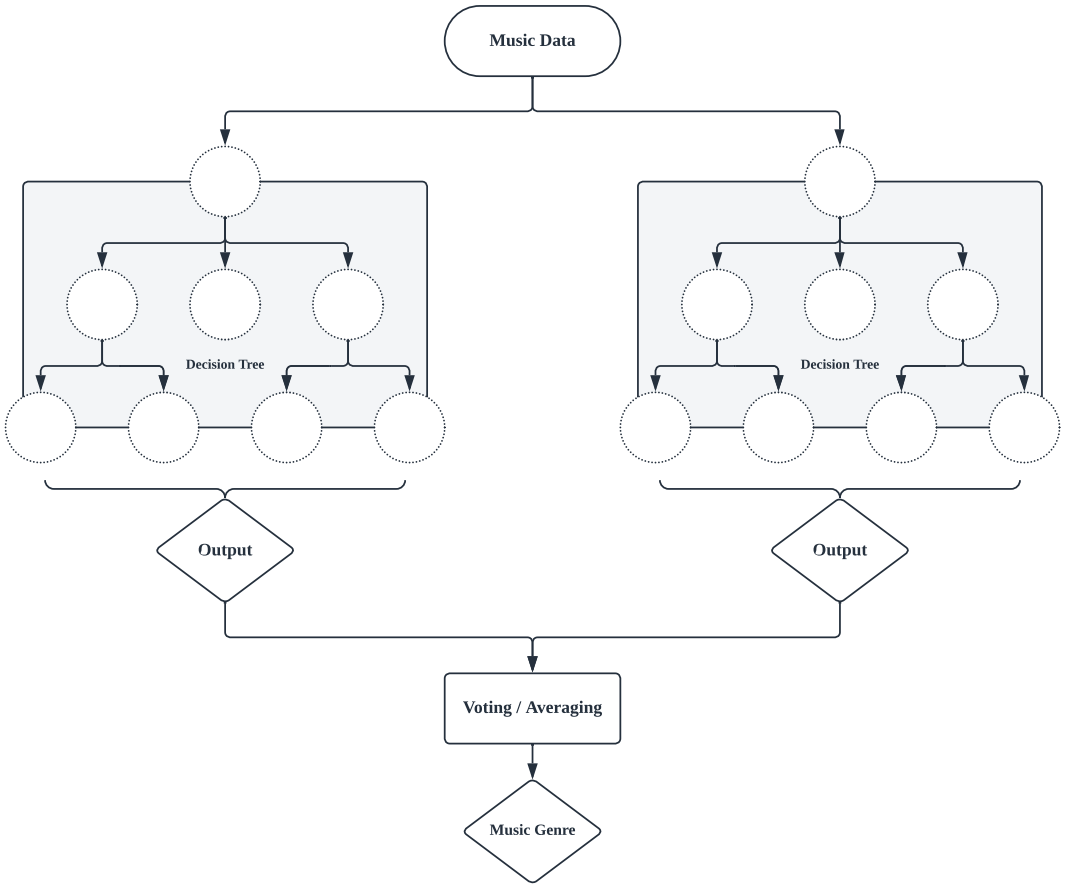

Figure 4.2: Structure of a sample Random Forest classifier.

As discussed in Chapter 3, Random Forest also provides a way to measure the significance of each feature in the dataset. For example, measuring the decrease in *"impurity"* or *"Gini importance"* [105] when a feature is used to split a node can indicate how important a feature is to the model. This can be useful in feature selection and comprehending the underlying relationships in the data. In other words, constructing a decision tree aims to split the data into two or more homogeneous groups based on a specific feature. The quality of the split is measured by impurity, which measures how mixed the labels are in each group; a lower impurity implies a better split. When a feature is used to split a node in a

decision tree, the decrease in impurity is calculated as the difference between the impurity of the parent node and the weighted sum of impurities of the child nodes. The importance of a feature is then calculated by averaging the decrease in impurity over all the trees in the forest that use that feature.

4.1.3 XGBoost Classifier

XGBoost classifier is a distinct implementation of the eXtreme Gradient Boosting algorithm [106] for classification. It is an effective and efficient machine learning model for binary and multi-class classification tasks. In the XGBoost classifier, decision trees are built one at a time and sequentially added to the model. Each tree decreases the error of the previous tree, and then the outputs of the individual trees are combined to make the final prediction. A basic algorithm for an XGBoost for music genre classification is presented below.

```
Create the model with a single decision tree
for t = 1 to T do
    Create a new decision tree
    Fit the tree to the gradient of the loss function of the current model
    Add the new tree to the model
    Update the model's predictions
end for
Apply regularization techniques
Majority vote of the predictions from all the trees
Use the trained model to make predictions on the test dataset
Evaluate the model's performance using different metrics
```

In the algorithm above, the model iterates from 1 to T, where T is the model's maxi-

mum number of decision trees. In each iteration, a new decision tree is created and trained to correct the errors made by the previous trees. In this step, the algorithm computes the gradient of the loss function of the current model for the training samples. The loss function measures how well the current model performs on the training set, and the gradient indicates the direction in which the model needs to be adjusted to reduce the loss. Next, the new tree is added to the model, and the predictions are updated. This process is repeated until the maximum number of trees is reached. After the iteration ends, the algorithm applies regularization techniques to the model to prevent overfitting and improve the model's generalization ability. Finally, the model gets the majority vote on the predictions from all the trees in the model to obtain the final prediction for the input music sample.

XGBoost is highly scalable, making it suitable for large datasets, and it can be parallelized to run on multiple machines. Another benefit of the XGBoost classifier is efficiently managing missing values, data with different features, and categorical variables. It is widely employed in many industries due to its high performance and speed in solving various classification problems, including user identification [107], climate projection [108], financial forecasts [109], and music genre prediction [110].

4.1.4 LightGBM Classifier

LightGBM is a Gradient Boosting Decision Tree (GBDT) algorithm designed to handle large datasets with numerous features more efficiently than XGBoost [111]. It uses two novel techniques [8], "*Gradient-based One-side Sampling*" and "*Exclusive Feature Bundling*":

- In traditional decision tree algorithms, each split is created by examining all the data points in a dataset to find the best-split point. However, the Gradient-based One-side Sampling technique creates a histogram of each feature's values. It uses the histogram to determine the best-split points, significantly reducing the time and memory required to create each tree, especially for large datasets.

- Exclusive feature bundling downsizes the number of features used for each tree by bundling exclusive features. This approach clusters correlated components together and only selects one feature per group for each tree, reducing the model's complexity and improving efficiency.

Being similar in many ways, XGBoost and LightGBM have some key differences. XGBoost uses the traditional greedy algorithm to construct decision trees, which is less efficient and requires more memory than the histogram-based algorithm used in LightGBM, making LightGBM more efficient and faster than XGBoost. Additionally, LightGBM can handle categorical variables more effectively, as it can create direct split points for categorical variables, whereas XGBoost requires them to be encoded as numerical values. LightGBM has been widely used in the music industry for different tasks, such as quantifying the music transition [112], popularity prediction [113], and genre recognition [114]. An instance of a LightGBM algorithm for music genre classification is presented below.

1: Create the model with a single decision tree
2: **for** $t = 1$ to T **do**
3: Create a new decision tree using the LightGBM algorithm
4: Fit the tree to the gradient of the loss function of the current model using gradient-based one-side sampling
5: Add the new tree to the model
6: Update the model's predictions using exclusive feature bundling
7: **end for**
8: Apply regularization techniques
9: Majority vote of the predictions from all the trees
10: Use the trained model to make predictions on the test dataset
11: Evaluate the model's performance using different metrics

4.1.5 K-Nearest Neighbors Classifier

k-NN is a supervised learning model based on the data distribution's local geometry and their relative distance measures [115]. It locates the k nearest data points in the feature space and predicts the class or value based on their majority class or average value, assuming that similar cases have similar class values [116]. The value of k can be chosen based on the problem. For example, a smaller value of k will result in more complex decision boundaries, while a larger value will result in more precise decision boundaries [117]. A decision boundary is a border or surface that separates the different classes or categories in the feature space. The k-NN classifier is easy to understand and implement, does not require a lot of computational resources, and can handle multi-class problems and non-linear decision boundaries. However, it is sensitive to the scale of the data, and it demands a large amount of memory to store the training data [118]. A sample algorithm of a k-NN in genre classification is demonstrated below.

1: Define the number of nearest neighbors (k)
2: Store the feature vectors and corresponding labels in a table or matrix
3: **for** each instance in the test dataset **do**
4: Calculate the distance between the instance and all other instances in the training dataset
5: Select the k-nearest instances from the training dataset based on the calculated distances
6: Determine the most common label among the k-nearest instances
7: Assign this label to the instance in the test dataset
8: **end for**
9: Return the predicted labels for all instances in the test dataset
10: Evaluate the model's performance using different metrics

Now that we have introduced the base models, we will discuss the ensemble learning models used in this study. By creating an ensemble of multiple classifiers, each classifier's weaknesses can be compensated for by the strengths of others, leading to a more robust and precise model for music genre prediction. Three popular ensemble approaches were employed in this research: voting, blending, and stacking models. We discuss these models in more detail below.

4.1.6 Voting Ensemble

A voting ensemble is an ensemble learning approach that trains multiple models independently, then uses their predictions based on a voting [119] strategy to make a final prediction. There are two main types of voting strategies, hard (majority) and soft voting,

1. *Hard voting* takes the majority vote on the base model's predictions:

$$\hat{y}_i = \arg\max_{y_i} \left(\text{count}(B_j(\overrightarrow{x_i}) == y_i) \right)$$

where $\overrightarrow{x_i}$ is the i$^{\text{th}}$ input feature vector, $B_j(\overrightarrow{x_i})$ is the predicted class label by the base model B_j, count() is the function that counts the number of occurrences, $\arg\max_{y_i}$ denotes the class label that has the highest count of votes, and $\hat{y}_i$ is the final predicted class.

2. *Soft voting* assesses the probability of each model's predictions:

$$\hat{y}_i = \arg\max_{y_i} \left(\frac{1}{N} \sum_{j=1}^{N} B_j(y_i|\overrightarrow{x_i}) \right)$$

where $B_j(y_i|\overrightarrow{x_i})$ is the predicted probability given the feature vector $\overrightarrow{x_i}$ and the base model B_j, N is the number of base models, $\arg\max_{y_i}$ denotes the class label that maximizes the sum of the predicted probabilities, and $\hat{y}_i$ is the final predicted class.

The purpose of using a voting ensemble is to enhance the performance of the final model by mixing the strengths of multiple base models. The voting ensemble can perform better than its base classifiers if the individual models are diverse enough and make different errors. For example, if all the base models are variations of the same algorithm with different hyperparameters, their predictions are likely highly correlated, and they probably make similar errors. In that case, the ensemble may not be able to reduce the overall error rate significantly. On the other hand, if base models are diverse enough, i.e., some base models are good at capturing global patterns in the data while others are better at modeling local details, the ensemble can combine these strengths to achieve better overall performance. In our study, we used the *VotingClassifier*[14] library and set the voting strategy to soft voting.

4.1.7 Blending Ensemble

A blending ensemble is another ensemble learning technique that combines multiple models to make a final prediction. It works by training multiple models independently on the same dataset, then using their predictions as input features to train a final meta-model which will make the final prediction. The meta-model is trained on the predictions of the base models rather than the original input features.

For our study, we implemented the blending ensemble method using the Random Forest classifier as the meta-model. The data is first split into the training set, the holdout set, and the testing set. Then the base models are trained on the training set and make predictions on the holdout set. These predictions are then concatenated to train the meta-model. Finally, the base models make predictions on the test set on which the meta-model makes the final predictions. As shown below, we have adopted and implemented the blending ensemble technique in our research.

[14] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html`.

1: Split X and y into training and testing sets
2: Split the training set into two parts: X_{train} and $X_{holdout}$, and their corresponding labels y_{train} and $y_{holdout}$
3: Initialize the base models (MLP, RF, XGBoost, LightGBM, k-NN)
4: **for** each model in base models **do**
5: Train the model on X_{train} and y_{train}
6: Make predictions on $X_{holdout}$
7: **end for**
8: Concatenate the predictions of base models and create X_{meta_train}
9: Initialize the meta-model (RF)
10: Train the meta-model on X_{meta_train} and $y_{holdout}$
11: **for** each model in base models **do**
12: Make predictions on X_{test}
13: **end for**
14: Concatenate the predictions of the base models and create X_{meta_test}
15: Use the meta-model to make final predictions on X_{meta_test}
16: Return the final predictions

4.1.8 Stacking Ensemble

Like blending, the stacking ensemble [11] uses multiple base models and a single meta-model to make the final predictions. The main difference between blending and stacking is how the base models are used to make predictions. In the blending ensemble method, base models are trained on the training set and then used to make predictions on a holdout set. However, in the stacking ensemble method, the base models are trained on different subsets of the training data through cross-validation. In particular, for each base model, cross-validation splits the training data into five folds and trains the model on four of the folds

while using the remaining fold as a validation set. This process is repeated five times using a different fold as the validation set. After each cross-validation iteration, the predicted classes for each instance in the validation set are returned. These predictions are then combined to create a complete set of predictions for the training data. In the next step, the base model predictions are stacked and used as input to the meta-model. This property makes stacking more robust to overfitting than blending. Our adopted stacking ensemble technique, described below, implements this approach.

1: Split X and y into training and test sets
2: Initialize the base models (MLP, RF, XGBoost, LightGBM, k-NN)
3: **for** each model in base models **do**
4: Train the model on X_{train} and y_{train}
5: Make predictions using cross-validation with 5 folds
6: **end for**
7: Concatenate the predictions of base models and create X_{meta_train}
8: Initialize the meta-model (RF)
9: Train the meta-model on X_{meta_train} and y_{train}
10: **for** each model in base models **do**
11: Make predictions on X_{test}
12: **end for**
13: Concatenate the predictions of base models and create X_{meta_test}
14: Use the meta-model to make final predictions on X_{meta_test}
15: Return the final predictions

4.2 Enhancing Classification with Augmented Features

The augmented features ensemble learning model that we propose is designed to improve the accuracy of classification models in comparison to traditional ensemble learning methods. The intuition behind this approach is to use the prediction probabilities generated by base models more effectively. Unlike other ensemble methods, where the meta-model is trained only on predictions of the base models, the augmented features ensemble extends the original input features (Step 2) by incorporating the prediction probabilities obtained from the base learners (Step 1) and makes the final prediction by the meta-model (Step 3), as shown in Figure 4.3.

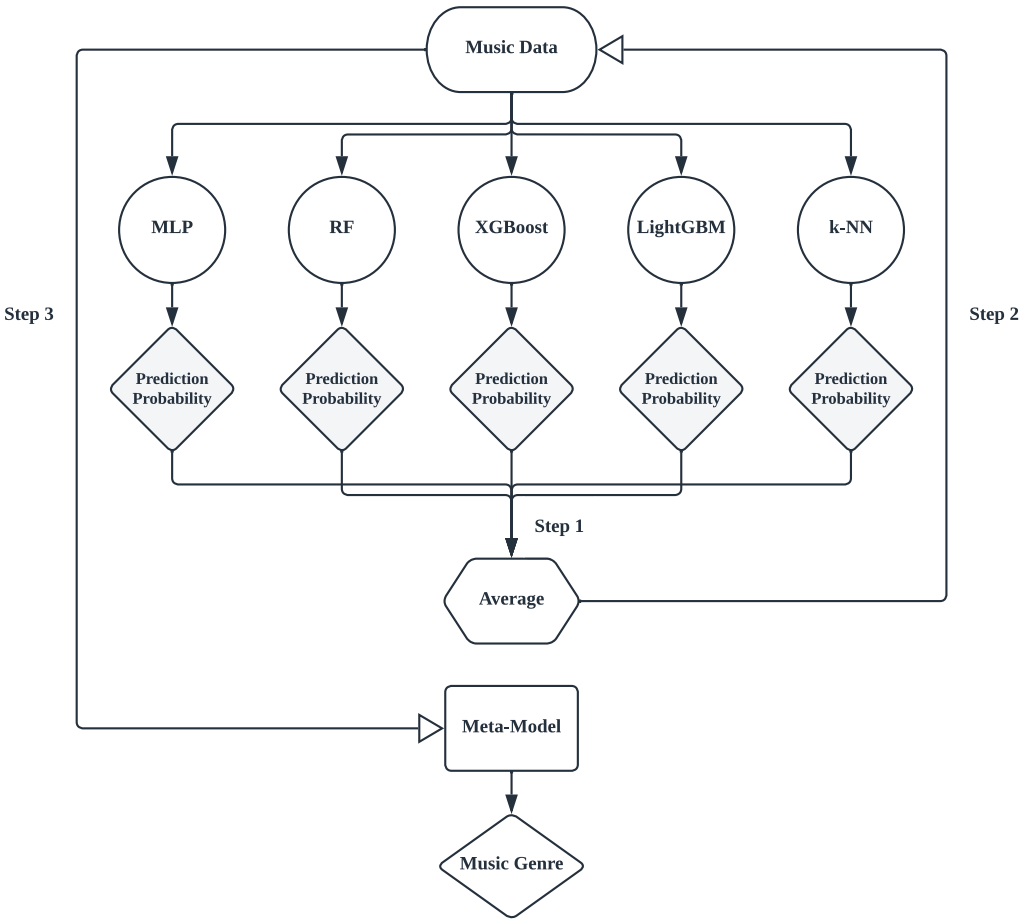

Figure 4.3: Structure of our proposed augmented features ensemble model for genre classification.

Intuitively we believe that this additional information provides a more comprehensive understanding of the data and helps to capture complex relationships and patterns that may not be evident from the original features alone. The augmented features ensemble aims to provide a more sophisticated representation of the input data that the meta-model can leverage to make better predictions. The use of prediction probabilities along with the original features in the augmented features ensemble is expected to produce significant improvements in the performance of classification models, particularly in situations where the input data is complex and the relationships between features are not well understood.

With this intuition in mind, we designed and implemented three approaches for our model. In the first approach, as outlined in section 4.2.1, we utilized an unweighted average to combine the probabilities generated by the base classifiers. This method took the average of the probabilities produced by the individual base classifiers without weighting or adjustment.

The second approach, discussed in section 4.2.2, involved a weighted average. In this method, we assigned different weights to the probabilities generated by the base classifiers, depending on their performance on the training data. The idea was to give more importance to the predictions of base classifiers that had performed better on the training data while giving less weight to those that had performed poorly.

We used a normalized weighted average in our final approach, described in section 4.2.3. This method involved normalizing the weights assigned to the prediction probabilities of the base classifiers. The normalization was done to ensure that all the probabilities generated by the base classifiers had the same scale and could be fairly compared.

After evaluating the results of these three approaches on the three datasets, our findings indicated that the model with an unweighted average outperformed the other two approaches in terms of accuracy. Therefore, for this research, we included only the best-performing approach (unweighted average) results in our findings and conclusions.

4.2.1 Unweighted Average

The proposed augmented features ensemble involves several steps to improve the accuracy of predictions. The unweighted average approach provides a simple yet powerful method for combining prediction probabilities from various base classifiers into a single vector. The general form of the mathematical representation for the unweighted augmented features ensemble is as follows:

$$\hat{y}_i = f_M\left(\text{concat}\left[\overrightarrow{x_i}\ ,\ \frac{1}{N}\sum_{j=1}^{N}\overrightarrow{P_j^i}(y_i|\overrightarrow{x_i},B_j)\right]\right)$$

where $\overrightarrow{x_i}$ is original feature vector for input sample i, $\overrightarrow{P_j^i}(y_i|\overrightarrow{x_i},B_j)$ is the vector of predicted probabilities given the base classifier B_j and feature vector $\overrightarrow{x_i}$, N is the number of base classifiers, $f_M()$ is the meta-model, and $\hat{y}_i$ is the final predicted class for the augmented input sample i by the meta-model.

For each base model B_j, a prediction probability vector $\overrightarrow{P_j^i}(y_i|\overrightarrow{x_i},B_j)$ is computed for the target output y_i. This vector has dimensions of $1 \times$ the number of class labels (genres), representing the probabilities associated with each genre for the given input $\overrightarrow{x_i}$ and the base model B_j. Then, we obtained a sum vector by combining the prediction vectors of each base model. The sum vector is then divided by the number of base models N to obtain the average prediction vector. This average prediction vector is then concatenated with the original feature vector $\overrightarrow{x_i}$, resulting in an additional feature for each target class label, as shown in Figure 4.4. This additional feature provides more information to the meta-model $f_M()$, ultimately making the final prediction $\hat{y}_i$.

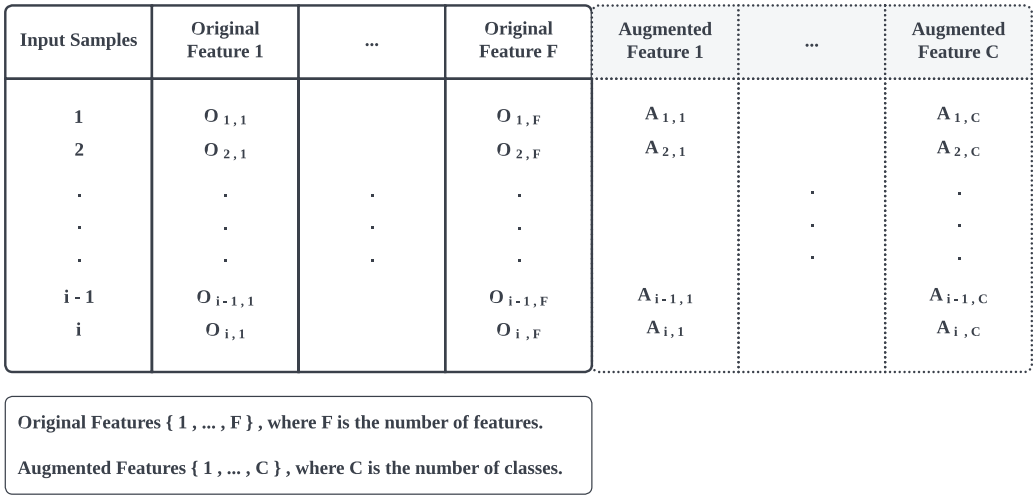

Figure 4.4: Concatenating the augmented features with the original features to create an augmented dataset.

Implementing the Algorithm

The input data is first divided into training and testing sets in the unweighted augmented features ensemble. Then, five base models are initialized. These include MLP, Random Forest, XGBoost, LightGBM, and k-NN. Next, each base model is trained using the training set, and the predictions are made and verified using cross-validation. The prediction probabilities generated by each base model are collected and averaged across all base classifiers. This averaged prediction is then combined with the original features of the training set to form a new feature set called "*X_meta_train*." The meta-model is trained on this new feature set in the next step. The base models are then used to make predictions on the test set. Again, the prediction probabilities generated by each base model are collected and averaged across all base classifiers. These averaged predictions are then combined with the original features of the test set to create "*X_meta_test*." Finally, the meta-model uses this new feature set to make the final predictions:

1: Split X and y into training and test sets
2: Initialize the base models (MLP, RF, XGBoost, LightGBM, k-NN)
3: **for** each model in base models **do**
4: Train the model on X_{train} and y_{train}
5: Make predictions using cross-validation
6: **end for**
7: Average the predictions across all base classifiers
8: Concatenate the averaged probabilities with the original features of the train data and create X_{meta_train}
9: Initialize the meta-model (RF)
10: Train the meta-model on X_{meta_train} and y_{train}
11: **for** each model in base models **do**
12: Make predictions on X_{test}
13: **end for**
14: Average the predictions across all classifiers
15: Concatenate the averaged probabilities with the original features of the test data and create X_{meta_test}
16: Use the meta-model to make final predictions on X_{meta_test}

4.2.2 Weighted Average

This approach considers the base models' performance for making the final prediction. By doing so, we can better balance the strengths and weaknesses of each base model. Once the performance scores of each base model have been computed, the weights can be calculated based on a performance metric. For instance, if we select accuracy as the metric, a base model with a higher accuracy score will be assigned a higher weight. The weights' purpose is to capture each base model's relative importance in the final prediction; Assign-

ing a higher weight will result in a more significant contribution of the base model to the final prediction. Next, the weighted average is computed by considering the weights given to each base model. The formula for the weighted average prediction is shown below:

$$\hat{y}_i = f_M\left(\text{concat}\left[\overrightarrow{x_i}\ ,\ \frac{1}{N}\sum_{j=1}^{N} w_j \overrightarrow{P_j^i}(y_i|\overrightarrow{x_i}, B_j)\right]\right)$$

where $\overrightarrow{x_i}$ is original feature vector for input sample i, $\overrightarrow{P_j^i}(y_i|\overrightarrow{x_i}, B_j)$ is the vector of predicted probabilities given the base classifier B_j and feature vector $\overrightarrow{x_i}$, w_j is the weight assigned to the base model j, N is the number of base classifiers, $f_M()$ is the meta-model, and $\hat{y}_i$ is the final predicted class for the augmented input sample i by the meta-model.

In this approach for each base model B_j, we first calculate the prediction probability $\overrightarrow{P_j^i}(y_i|\overrightarrow{x_i}, B_j)$ for the input feature vector $\overrightarrow{x_i}$ and target output y_i, which is represented as a vector $\overrightarrow{P_j^i}$. Then, we weight each prediction probability vector $\overrightarrow{P_j^i}$ by a weight w_j based on the selected performance metric. In the next step, we compute a weighted sum of the prediction probability vectors $\overrightarrow{P_j^i}$ to acquire a single combined prediction probability vector. Then, the combined prediction vector is concatenated with the original feature vector $\overrightarrow{x_i}$. Finally, the concatenated vector is used in meta-model $f_M()$ to make the final prediction $\hat{y}_i$.

The advantage of this approach is that it can reduce the prediction variance, as it assigns higher weights to the more accurate models and lower weights to the less reliable ones, allowing it to balance each base model's strengths and weaknesses and potentially produce more robust predictions. Furthermore, this approach is flexible and can be adapted to different performance metrics. For example, if the goal is to reduce false negatives, recall can be used as the performance metric instead of accuracy.

4.2.3 Normalized Weighted Average

In the non-normalized weighted average approach, the weight assigned to each base classifier measures the confidence in its predictions. The larger the weight, the more influential the prediction of the base classifier is expected to be in the final result. However, this can lead to an imbalance in the overall prediction if one weight is significantly larger than the others since the dominant weight can overpower the predictions made by other base classifiers and cause the final prediction to be skewed toward the prediction made by the classifier with the highest weight. The equation for this approach is written below:

$$\hat{y}_i = f_M\left(\text{concat}\left[\overrightarrow{x_i},\ \frac{\sum_{j=1}^{N} w_j \overrightarrow{P_j^i}(y_i|\overrightarrow{x_i}, B_j)}{\sum_{j=1}^{N} w_j}\right]\right)$$

where $\overrightarrow{x_i}$ is original feature vector for input sample i, $\overrightarrow{P_j^i}(y_i|\overrightarrow{x_i}, B_j)$ is the vector of predicted probabilities given the base classifier B_j and feature vector $\overrightarrow{x_i}$, w_j is the weight assigned to the base model j, N is the number of base classifiers, $f_M()$ is the meta-model, and $\hat{y}_i$ is the final predicted class for the augmented input sample i by the meta-model.

In normalized weighted augmented features for each base model B_j, we first compute the prediction probability vector $\overrightarrow{P_j^i}(y_i|\overrightarrow{x_i}, B_j)$ for the target output y_i. Then, we weight each prediction vector $\overrightarrow{P_j^i}$ by a weight w_j that reflects the importance of the corresponding base model; The weights w_j are non-negative and sum up to 1. Next, We compute a weighted sum of the prediction vectors $\overrightarrow{P_j^i}$ and normalize it by dividing it by the sum of the weights. This vector is concatenated with the original feature vector $\overrightarrow{x_i}$. Finally, The concatenated vector is passed through a meta-model $f_M()$ that produces the final prediction $\hat{y}_i$.

Normalizing the weights helps to address the issue of imbalanced weights by ensuring that the sum of all weights is equal to one, making it also easier to interpret the relative importance of each base classifier in the final prediction. Furthermore, normalizing helps

prevent any single weight from dominating the final prediction, which is essential in ensemble learning. In ensemble learning, the goal is to create an overall prediction more accurate than any individual base classifier prediction.

4.2.4 Comparing Ensemble Methods with Augmented Features Model

Voting, blending, and stacking, in addition to our model with augmented features, are all ensemble learning methods that aim to improve the performance of a machine learning model by combining the predictions of multiple base models. However, they differ in how they use the primary predictions of base models to make the final predictions.

- Voting is a simple ensemble method that does not use a meta-model. Instead, it aggregates the predictions of multiple individual base classifiers and uses a voting strategy, such as soft or hard voting, to make the final prediction.
- In blending, the base models make predictions on the same dataset, and then the predictions are combined for use in a meta-model. Finally, the meta-model is trained on a holdout set and makes the final predictions.
- In stacking, the base models are trained on different subsets of the dataset, and then the predictions of the base models are used as features for a meta-model. The meta-model is then trained on the entire dataset and makes the final predictions.
- In our augmented features ensemble learning, the base models generate additional features combined with the dataset's original features. These new features are then used to train a meta-model, which makes the final predictions.

While stacking and blending are efficient methods utilizing a meta-model, augmented features ensemble may be considered a more robust and comprehensive approach as it combines original features in the dataset and the average predictions of base models, which leads to a more extensive feature space. Furthermore, it is useful when base models can extract different types of information from the dataset. Another possible advantage is our

model's ability to handle overfitting, a common problem in machine learning models. Overfitting occurs when a model is trained too well on the training data and performs poorly on the unseen data. Combining the prediction probabilities of multiple base models by utilizing cross-validation, the augmented features model may reduce overfitting and improve the prediction performance of the final model, as demonstrated in the next chapter.

4.3 Performance Validation

To assess the performance of the models in this research, we employed cross-validation with five folds. Cross-validation [13] is a technique used to evaluate the performance of machine learning models by splitting the data into multiple subsets called "*folds*." The model is trained on distinct subsets of the data and tested on the remaining subset [120]. In the case of 5-fold cross-validation, As shown in Figure 4.5, we divide the data into five equal parts and train and test the model five times. Each time, one of the five sets is used as the testing set, and the other four are used as the training set. The results of each test are then combined to give a general estimate of the model's performance.

Figure 4.5: 5-fold cross-validation to assess the model's performance.

Cross-validation is a robust method to evaluate model performance as it uses different sets of data to train and test a model. It also helps reduce the potential of overfitting and delivers a more realistic estimate of the model's performance on unseen data. Shuffling the data before splitting it into different folds is also essential to avoid data leakage, ensuring that the data in each fold is a random sample of the whole dataset and that the model is not trained or tested on the same data.

The performance of a genre classification model, a multiclass classification problem, is typically evaluated using metrics such as *accuracy*, *micro/macro-averaged precision*, *micro/macro-averaged recall*, and *micro/macro-averaged F1-score* [121]. These metrics are built upon a confusion matrix [122] and assess the model's performance in predicting a music piece's correct class or genre. First, we will introduce the confusion matrix and its properties, shown in Figure 4.6, and then explain how these metrics are calculated in multiclass classification.

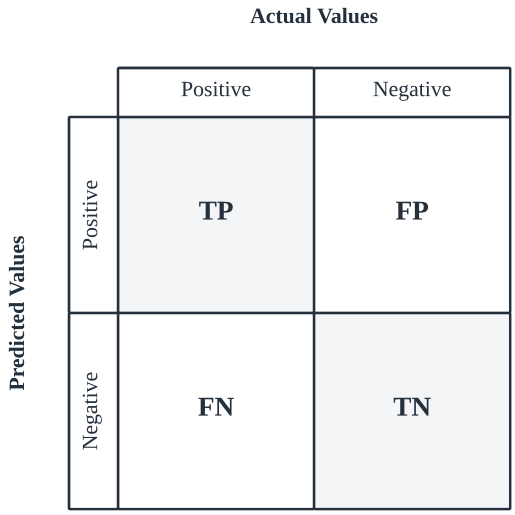

Figure 4.6: The structure of a confusion matrix.

The evaluation metrics mentioned above are based on the number of instances classified correctly or incorrectly. *True Positives (TP)*, *True Negatives (TN)*, *False Positives (FP)*, and *False Negatives (FN)* are four such metrics that are commonly used in multiclass classifica-

tion for this purpose. By understanding these terms, we can comprehend how well a model performs and where it might be making mistakes.

- **TP:** Instances correctly predicted as positive for a given class; the model correctly classifies the instances that belong to a particular class.
- **TN:** Instances correctly predicted as negative for a given class; the model correctly classifies the instances that do not belong to a particular class.
- **FP:** Instances incorrectly predicted as positive for a given class; the instances that do not belong to a particular class but are incorrectly classified as belonging to that class by the model.
- **FN:** Instances incorrectly predicted as negative for a given class; the instances belong to a particular class but are incorrectly classified by the model as not belonging to that class.

Now that we have explained the confusion matrix, we will explain the commonly used micro/macro metrics for evaluating classification models in more detail. Macro and micro metrics are two approaches to combine performance metrics for multiclass classification problems. Micro metrics weight each class based on its frequency in the dataset and give an overall score that reflects the model's performance on all classes. In contrast, macro metrics treat each class equally and provide separate scores for each class. Micro metrics are commonly used when the goal is to optimize the model's overall performance, while macro metrics are used when the goal is to identify the model's performance for each class. Additionally, the micro metrics are sensitive to class imbalance, while macro metrics are less sensitive to this issue.

Accuracy

Accuracy is a global metric that defines the overall percentage of correctly predicted instances across all classes and often is not calculated using macro or micro metrics, as

it measures the overall proportion of correct predictions. It is computed by calculating the ratio of the sum of all true positives and negatives to the sum of all true positives and negatives and false positives and negatives for all classes.

$$\text{accuracy} = \frac{\sum_{i=1}^{C} (TP_i + TN_i)}{\sum_{i=1}^{C} (TP_i + TN_i + FP_i + FN_i)}$$

where C is the number of classes, TP_i is the number of true positives for class i, TN_i is the number of true negatives for class i, FP_i is the number of false positives for class i, and FN_i is the number of false negatives for class i.

Precision

Precision generally represents the model's ability to avoid false positives. Micro-averaged precision is a metric that considers both true and false positives, calculated by dividing the sum of true positives by the sum of all predicted positives for all classes.

$$\text{micro\_precision} = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C} (TP_i + FP_i)}$$

where C is the number of classes, TP_i is the number of true positives for class i, and FP_i is the number of false positives for class i.

Whereas macro-averaged precision is the average of precision scores for all classes, demonstrating the model's performance for each class separately.

$$\text{macro\_precision} = \frac{\sum_{i=1}^{C} \text{precision}_i}{C}$$

where C is the number of classes and precision_i is the precision score for class i.

Recall

Recall represents the model's ability to identify all positive instances. Micro-averaged recall is a metric that considers both true positives and false negatives for all classes. It is calculated by dividing the sum of true positives by the sum of true positives and false negatives for all classes.

$$\text{micro-recall} = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C} (TP_i + FN_i)}$$

where C is the number of classes, TP_i is the number of true positives for class i, and FN_i is the number of false negatives for class i.

On the other hand, macro-averaged recall calculates the average recall score across all classes, providing insight into the model's performance for each class individually.

$$\text{macro-recall} = \frac{\sum_{i=1}^{C} \text{recall}_i}{C}$$

where C is the number of classes and recall_i is the recall score for class i.

F1-score

F1-score provides an overall measure of the model's performance by combining a model's precision and recall scores. Micro-averaged F1-score is the harmonic mean of micro-averaged precision and micro-averaged recall.

$$\text{micro-F1-score} = \frac{2(\text{micro-precision})(\text{micro-recall})}{\text{micro-precision} + \text{micro-recall}}$$

Furthermore, macro-averaged F1-score is a metric that provides a way to evaluate the

model's performance for each class separately by taking the average of the F1-scores for all classes. It offers insight into the model's ability to balance precision and recall across all classes.

$$\text{macro-F1-score} = \frac{\sum_{i=1}^{C} \text{F1-score}_i}{C}$$

where C is the number of classes and F1-score_i is the F1-score for class i.

In our study, we focused on evaluating the performance of different models using the accuracy metric. As discussed, we first split our dataset into five folds using cross-validation. Once the dataset was divided, we trained and tested the models on each fold. Next, we calculated the models' accuracy for each fold, and then, as shown below, we averaged the accuracies across all five folds to obtain the mean accuracy for each model. The mean accuracy measures the overall performance of each model and provides insight into the most effective model for the music genre classification task.

$$\overline{\text{accuracy}} = \frac{1}{N} \sum_{i=1}^{N} \text{accuracy}_i$$

where N is the number of folds ($N = 5$), and accuracy_i is the accuracy score for the fold i.

Chapter 5

Results and Discussion

This chapter presents the results of our proposed ensemble learning approach and also compares it to other ensemble models and classifiers using various feature selection methods. The results of our research are presented in eight sections. In the first section, we present the classification results without using any feature selection method on each dataset; Spotify, TCC_CED, and GTZAN. However, it is also essential to evaluate the performance of the models when utilizing a feature selection technique. This allows us to understand the potential benefits of reducing the number of features used and focusing on the most relevant ones. The subsequent sections exhibit the results after using RFE, tree-based, forward, backward, and bidirectional feature selection techniques, and the best potential approach for each dataset. Finally, in the last section, we discuss our results in detail, highlight our research's most significant findings, and discuss their implications.

5.1 Classification without Using a Feature Selection Technique

First, we compare the performance of the classification models without using a feature selection technique to understand the performance of the models based on all of the available features. The tables in this section include information about the execution time in seconds, the mean accuracy, and the accuracy for five different folds. Table 5.1 presents the classification results for the Spotify dataset without employing a feature selection method. The mean accuracy of the models varied from 78.6% to 86.5%, with the augmented features ensemble model reaching the highest performance among all the models. Meanwhile, the

k-NN model had the poorest performance, with an accuracy of 78.6%.

Table 5.1: Performance results without using a feature selection technique (Spotify).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
Augmented Features	**331.253**	**0.865**	**0.859**	**0.863**	**0.871**	**0.866**	**0.864**
Stacking	319.504	0.863	0.853	0.857	0.877	0.870	0.859
Blending	55.666	0.862	0.857	0.860	0.865	0.864	0.864
MLP	223.665	0.808	0.607	0.855	0.902	0.881	0.798
Voting	315.940	0.806	0.603	0.850	0.896	0.876	0.804
LightGBM	3.686	0.801	0.577	0.858	0.897	0.876	0.799
XGBoost	31.306	0.796	0.575	0.843	0.889	0.871	0.800
RF	43.754	0.790	0.597	0.830	0.873	0.862	0.787
k-NN	13.155	0.786	0.699	0.805	0.817	0.818	0.788

Table 5.2 provides an overview of the results obtained on the TCC_CED dataset. The proposed augmented features ensemble model demonstrated its superior performance again by attaining mean accuracy of 48.3%. On the other hand, the XGBoost model did not perform as expected, with a mean accuracy of only 17.9%.

Table 5.2: Performance results without using a feature selection technique (TCC_CED).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
Augmented Features	**215.721**	**0.483**	**0.468**	**0.499**	**0.503**	**0.461**	**0.482**
Stacking	199.073	0.466	0.462	0.459	0.489	0.460	0.457
Blending	33.796	0.445	0.438	0.440	0.469	0.430	0.445
MLP	124.234	0.389	0.316	0.358	0.412	0.473	0.384
RF	47.028	0.344	0.325	0.243	0.371	0.432	0.351
k-NN	1.145	0.340	0.311	0.315	0.357	0.364	0.351
Voting	194.396	0.240	0.269	0.145	0.199	0.332	0.253
LightGBM	3.105	0.182	0.257	0.109	0.139	0.220	0.186
XGBoost	21.488	0.179	0.253	0.115	0.123	0.211	0.193

Similarly, Table 5.3 presents the results on the GTZAN dataset. Again, the results reveal that the augmented features performed the best, with a mean accuracy of 84%, showing an improvement compared to the second-best performing classifier, the MLP, with a mean accuracy of 81%. Additionally, the augmented features classifier beat the XGBoost classifier by almost 10%, further highlighting its superior performance.

Table 5.3: Performance results without using a feature selection technique (GTZAN).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
Augmented Features	**6.416**	**0.840**	**0.800**	**0.950**	**0.800**	**0.900**	**0.750**
MLP	2.478	0.810	0.790	0.830	0.780	0.820	0.830
Voting	5.162	0.786	0.780	0.820	0.730	0.770	0.830
Stacking	6.072	0.780	0.750	0.850	0.800	0.700	0.800
Blending	1.583	0.780	0.750	0.900	0.750	0.700	0.800
RF	1.067	0.778	0.780	0.820	0.700	0.770	0.820
LightGBM	0.909	0.760	0.730	0.800	0.730	0.770	0.770
k-NN	0.062	0.754	0.730	0.770	0.680	0.800	0.790
XGBoost	0.737	0.742	0.730	0.790	0.680	0.750	0.760

5.2 Recursive Feature Elimination

We applied RFE to select the most important components (previously discussed in Table 3.8) and re-evaluated the performance of the classification models on each dataset. As shown in Table 5.4, the results indicate that similar to not using a feature selection method, augmented features achieved the highest mean accuracy of 86.5% among all the classification models tested on the Spotify dataset, negligibly outperforming stacking ensemble by 0.02%. However, it beat the well-known voting ensemble by 5.9%. The classification accuracy of the different models on the Spotify dataset ranges from 78.6% to 86.5%, with the augmented features model having the best performance and k-NN being the least effective.

Table 5.4: Performance results using RFE (Spotify).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
Augmented Features	**340.587**	**0.865**	**0.859**	**0.863**	**0.871**	**0.866**	**0.864**
Stacking	311.863	0.863	0.853	0.857	0.877	0.87	0.859
Blending	56.682	0.862	0.857	0.86	0.865	0.864	0.864
MLP	228.702	0.808	0.607	0.855	0.902	0.881	0.798
Voting	318.163	0.806	0.603	0.85	0.896	0.876	0.804
LightGBM	4.407	0.801	0.577	0.858	0.897	0.876	0.799
XGBoost	31.632	0.796	0.575	0.843	0.889	0.871	0.8
RF	43.607	0.79	0.597	0.83	0.873	0.862	0.787
k-NN	13.201	0.786	0.699	0.805	0.817	0.818	0.788

As displayed in Table 5.5, the best-performing model was the one that utilized augmented features. This model achieved a mean accuracy of 48%, significantly higher than the other tested models. Specifically, the stacking and blending models had a mean accuracy of 45.2% and 44.6%, lower than the proposed augmented inputs ensemble. Furthermore, the XGBoost model performed poorly and achieved a mean accuracy of 17.5%, a significant 30.5% lower than the proposed augmented inputs ensemble.

Table 5.5: Performance results using RFE (TCC_CED).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
Augmented Features	**193.513**	**0.480**	**0.485**	**0.488**	**0.489**	**0.456**	**0.479**
Stacking	178.590	0.452	0.448	0.440	0.483	0.443	0.445
Blending	31.569	0.446	0.451	0.442	0.459	0.439	0.439
MLP	119.077	0.373	0.306	0.355	0.398	0.475	0.329
k-NN	7.754	0.336	0.298	0.303	0.335	0.373	0.370
RF	32.336	0.328	0.305	0.232	0.348	0.418	0.338
Voting	176.450	0.229	0.270	0.138	0.190	0.315	0.230
LightGBM	2.540	0.176	0.256	0.104	0.136	0.200	0.182
XGBoost	15.072	0.175	0.258	0.104	0.126	0.198	0.188

In Table 5.6, it is clear that the augmented features continue to demonstrate superior performance compared to other classifiers. The mean prediction accuracy for this model was 87%, representing a substantial improvement over the second-best classifier, MLP, which had a prediction accuracy of 79.4%. Furthermore, the augmented features also performed better than the third-best ensemble model, stacking, which had a prediction accuracy of 78%. The k-NN classifier, on the other hand, had the poorest performance of all the models, with a prediction accuracy of only 75.6%.

Table 5.6: Performance results using RFE (GTZAN).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
Augmented Features	**5.619**	**0.870**	**0.850**	**0.950**	**0.900**	**0.900**	**0.750**
MLP	2.479	0.794	0.740	0.870	0.770	0.820	0.770
Stacking	5.444	0.790	0.750	0.950	0.800	0.850	0.600
Voting	4.722	0.788	0.780	0.860	0.710	0.790	0.800
XGBoost	0.674	0.770	0.750	0.830	0.710	0.760	0.800
Blending	1.491	0.770	0.800	0.850	0.750	0.750	0.700
LightGBM	0.784	0.768	0.770	0.840	0.720	0.760	0.750
RF	0.863	0.764	0.780	0.830	0.670	0.750	0.790
k-NN	0.031	0.756	0.720	0.790	0.680	0.790	0.800

5.3 Tree-based Feature Selection

When using the tree-based feature selection (Table 3.9) on the Spotify dataset, as seen in Table 5.7, the stacking ensemble slightly outperformed the augmented inputs ensemble and achieved the highest accuracy of 86.8%, the difference between the two models was only 0.05%, but it was enough to make the stacking ensemble the top model in terms of accuracy. Additionally, the k-NN classifier achieved the lowest classification accuracy of 78.6%, significantly lower than other models' performance.

Table 5.7: Performance results using tree-based feature selection (Spotify).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
Stacking	322.476	0.868	0.860	0.865	0.879	0.872	0.863
Blending	56.009	0.865	0.853	0.865	0.876	0.863	0.866
Augmented Features	**341.767**	**0.863**	**0.853**	**0.862**	**0.873**	**0.867**	**0.860**
MLP	229.170	0.810	0.616	0.853	0.900	0.882	0.796
Voting	318.017	0.808	0.612	0.853	0.895	0.875	0.806
LightGBM	3.842	0.801	0.577	0.858	0.897	0.877	0.799
XGBoost	31.588	0.797	0.575	0.844	0.891	0.869	0.806
RF	43.289	0.791	0.602	0.830	0.876	0.861	0.785
k-NN	13.229	0.786	0.699	0.805	0.817	0.818	0.788

As observed in Table 5.8, the model with the highest mean accuracy on the TCC_CED dataset is the augmented features ensemble, with a score of 47.8%, followed by the blending ensemble with 46.9% mean accuracy. The model with the lowest mean accuracy is XGBoost, with a score of 17.4%. The performance of the models varies significantly, with the top-performing models having accuracy scores over 40%, while the lower-performing models have scores below 30%.

Table 5.8: Performance results using tree-based feature selection (TCC_CED).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
Augmented Features	**194.513**	**0.478**	**0.466**	**0.477**	**0.503**	**0.457**	**0.486**
Blending	31.603	0.469	0.453	0.481	0.484	0.454	0.471
Stacking	175.163	0.449	0.433	0.454	0.466	0.439	0.451
MLP	118.898	0.386	0.319	0.357	0.420	0.489	0.345
k-NN	7.646	0.339	0.305	0.305	0.342	0.379	0.365
RF	31.825	0.317	0.293	0.237	0.324	0.410	0.323
Voting	172.654	0.230	0.270	0.138	0.193	0.308	0.239
LightGBM	2.572	0.179	0.257	0.107	0.137	0.208	0.187
XGBoost	14.962	0.174	0.256	0.109	0.132	0.184	0.190

The best-performing model on the GTZAN dataset (Table 5.9) is the augmented features model with a mean accuracy of 87%. In addition, the proposed model enhanced the mean accuracy of stacking and blending ensembles by 7% and 8%, respectively. On the other hand, the classifiers such as RF, LightGBM, XGBoost, and k-NN performed poorly, with their mean accuracies ranging from 71.2% to 74%.

Table 5.9: Performance results using tree-based feature selection (GTZAN).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
Augmented Features	**5.633**	**0.870**	**0.900**	**0.900**	**0.900**	**0.900**	**0.750**
Stacking	5.490	0.800	0.800	0.850	0.800	0.850	0.700
Blending	1.490	0.790	0.750	0.800	0.700	1.000	0.700
MLP	2.463	0.766	0.750	0.830	0.720	0.750	0.780
Voting	4.737	0.766	0.750	0.820	0.760	0.720	0.780
RF	0.878689	0.740	0.760	0.820	0.630	0.700	0.790
LightGBM	0.815	0.738	0.720	0.760	0.740	0.720	0.750
XGBoost	0.674	0.728	0.720	0.780	0.670	0.720	0.750
k-NN	0.031	0.712	0.690	0.770	0.660	0.670	0.770

5.4 Forward Feature Selection

Table 5.10 presents the results of the genre classification of the Spotify dataset. The best features were selected using the wrapper method forward feature selection as previously discussed in Table 3.10. The augmented inputs ensemble achieved a mean accuracy similar to the stacking and blending ensembles. However, the stacking ensemble slightly outperformed the augmented inputs ensemble by 0.02%. In addition, the augmented inputs ensemble improved the overall classification accuracy by 5.6% compared to the voting ensemble and by 7.6% compared to the k-NN classifier.

Table 5.10: Performance results using forward feature selection (Spotify).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
Stacking	190.239	0.859	0.844	0.850	0.867	0.867	0.866
Augmented Features	**210.976**	**0.857**	**0.845**	**0.846**	**0.871**	**0.863**	**0.857**
Blending	35.285	0.855	0.845	0.844	0.868	0.862	0.855
MLP	95.690	0.803	0.606	0.837	0.895	0.879	0.800
Voting	183.890	0.801	0.601	0.841	0.889	0.872	0.802
LightGBM	3.341	0.793	0.568	0.847	0.889	0.873	0.790
XGBoost	30.977	0.790	0.567	0.840	0.882	0.865	0.798
RF	42.665	0.783	0.588	0.820	0.872	0.858	0.779
k-NN	10.924	0.781	0.687	0.798	0.817	0.817	0.784

On the TCC_CED dataset, as presented in Table 5.11, the augmented inputs ensemble achieved the best overall classification accuracy at 47.6%. This was 1.8%, 2.8%, and a significant 24.9% advancement compared to the blending, stacking, and voting ensembles. However, like before, XGBoost gained the lowest mean accuracy at 17.2%, almost 30% lower than the augmented inputs ensemble.

Table 5.11: Performance results using forward feature selection (TCC_CED).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
Augmented Features	**194.652**	**0.476**	**0.464**	**0.485**	**0.507**	**0.443**	**0.479**
Blending	29.943	0.458	0.451	0.478	0.493	0.415	0.452
Stacking	164.441	0.448	0.447	0.452	0.460	0.436	0.444
MLP	105.140	0.371	0.307	0.330	0.416	0.476	0.329
k-NN	9.412	0.337	0.311	0.295	0.336	0.363	0.379
RF	33.935	0.327	0.311	0.230	0.339	0.412	0.341
Voting	164.627	0.227	0.268	0.133	0.192	0.310	0.232
LightGBM	4.724	0.174	0.253	0.100	0.132	0.203	0.183
XGBoost	17.648	0.172	0.258	0.099	0.127	0.193	0.184

As illustrated in Table 5.12, while the ensemble learning models, excluding blending, achieved a mean classification accuracy of 72% on the GTZAN dataset, it was still a notable drop compared to previous methods (which had an accuracy of around 87%). Furthermore, the LightGBM and k-NN classifiers struggled to acquire an accuracy of 70% even with the help of forward feature selection, highlighting the importance of considering multiple methods and approaches to improve performance.

Table 5.12: Performance results using forward feature selection (GTZAN).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
Augmented Features	**4.378**	**0.720**	**0.650**	**0.800**	**0.750**	**0.750**	**0.650**
Stacking	4.439	0.720	0.700	0.750	0.850	0.750	0.550
Voting	4.148	0.720	0.760	0.720	0.690	0.700	0.730
MLP	1.057	0.716	0.750	0.720	0.670	0.720	0.720
RF	0.909	0.712	0.760	0.740	0.610	0.700	0.750
Blending	1.368	0.710	0.600	0.850	0.700	0.750	0.650
XGBoost	0.744	0.706	0.730	0.700	0.680	0.680	0.740
LightGBM	0.731	0.696	0.730	0.710	0.670	0.670	0.700
k-NN	0.033	0.678	0.730	0.710	0.590	0.650	0.710

5.5 Backward Feature Selection

The results in Table 5.13 showed that the stacking ensemble was the most accurate classification model, surpassing all other methods in terms of mean accuracy. The features selected by the second wrapper method, backward feature selection (Table 3.11), combined with the stacking technique, resulted in a mean accuracy of 85.9%, slightly lower than that of the augmented inputs (0.1%) and blending (0.2%) methods. On the other hand, the voting ensemble demonstrated lower performance with a mean accuracy of 80.1%, making it the least accurate of all the ensemble learning models considered in the study.

Table 5.13: Performance results using backward feature selection (Spotify).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
Stacking	190.924	0.859	0.843	0.848	0.872	0.865	0.868
Augmented Features	**216.332**	**0.858**	**0.849**	**0.854**	**0.869**	**0.863**	**0.858**
Blending	34.019	0.857	0.850	0.845	0.871	0.861	0.856
MLP	98.364	0.803	0.600	0.839	0.898	0.879	0.800
Voting	191.449	0.801	0.603	0.843	0.891	0.869	0.800
LightGBM	4.303	0.794	0.570	0.847	0.889	0.874	0.790
XGBoost	34.611	0.790	0.565	0.838	0.885	0.865	0.795
RF	45.220	0.785	0.596	0.820	0.873	0.857	0.778
k-NN	12.310	0.781	0.687	0.798	0.817	0.817	0.784

Employing the backward feature selection technique on the TCC_CED dataset, Table 5.14, resulted in the augmented features producing the best results with an accuracy of 48%. This marks an increase of 3.2% compared to the accuracy achieved by stacking and blending ensembles and a significant improvement of 24.2% compared to the voting ensemble.

Table 5.14: Performance results using backward feature selection (TCC_CED).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
Augmented Features	**202.789**	**0.480**	**0.465**	**0.503**	**0.494**	**0.458**	**0.480**
Stacking	189.671	0.448	0.439	0.452	0.474	0.423	0.449
Blending	32.129	0.448	0.442	0.444	0.481	0.419	0.453
MLP	115.302	0.383	0.314	0.358	0.423	0.469	0.352
RF	41.835	0.353	0.307	0.277	0.393	0.437	0.352
k-NN	1.269	0.341	0.306	0.318	0.355	0.364	0.360
Voting	179.399	0.238	0.273	0.145	0.208	0.326	0.238
LightGBM	3.456	0.178	0.256	0.105	0.134	0.213	0.182
XGBoost	24.082	0.177	0.257	0.112	0.123	0.203	0.193

Table 5.15 displays the results of the GTZAN dataset, indicating that the base classifiers, such as MLP, RF, k-NN, and LightGBM, performed better than the ensemble learning models, including the augmented features, blending, and stacking. Despite this, the accuracy range of all models is relatively close, with a mean classification accuracy range of between 70% and 74%. This implies that all models are performing similarly, but the base classifiers and the voting ensemble were able to outperform the others in this particular instance.

Table 5.15: Performance results using backward feature selection (GTZAN).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
MLP	2.479	0.748	0.720	0.770	0.760	0.760	0.730
Voting	4.738	0.744	0.730	0.790	0.690	0.730	0.780
RF	0.883	0.730	0.730	0.740	0.650	0.740	0.790
k-NN	0.031	0.718	0.660	0.730	0.670	0.750	0.780
LightGBM	0.736	0.714	0.730	0.740	0.670	0.690	0.740
Augmented Features	**5.514**	**0.710**	**0.600**	**0.800**	**0.800**	**0.800**	**0.550**
Blending	1.474	0.710	0.650	0.750	0.800	0.750	0.600
XGBoost	0.674	0.708	0.720	0.780	0.630	0.710	0.700
Stacking	5.412	0.700	0.550	0.750	0.850	0.750	0.600

5.6 Bidirectional Feature Selection

In the case of the Spotify dataset, the bidirectional feature selection method showed similar results to the traditional forward and backward selection methods. Table 5.16 compares the results of different ensemble models on the Spotify dataset. The proposed augmented features ensemble performed the best, with an accuracy of 85.7% in genre classification. Furthermore, the least accurate classifier was the k-NN model, with an accuracy of only 78.1%, underlining the difference in performance between the augmented features ensemble and the k-NN model, with a gap of 7.6%.

Table 5.16: Performance results using bidirectional feature selection (Spotify).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
Augmented Features	**211.397**	**0.857**	**0.846**	**0.849**	**0.867**	**0.864**	**0.858**
Stacking	184.791	0.856	0.841	0.844	0.868	0.866	0.862
Blending	33.505	0.856	0.844	0.846	0.869	0.862	0.857
MLP	95.027	0.807	0.614	0.837	0.901	0.879	0.800
Voting	181.210	0.803	0.609	0.842	0.889	0.871	0.802
LightGBM	3.372	0.793	0.568	0.847	0.889	0.873	0.790
XGBoost	31.150	0.790	0.569	0.838	0.881	0.867	0.797
RF	42.592	0.783	0.588	0.820	0.872	0.856	0.779
k-NN	10.516	0.781	0.687	0.798	0.817	0.817	0.784

The TCC_CED dataset also produced similar results, with the augmented features ensemble being more precise than other models, successfully classifying the genres 73% of the time. These findings can be seen in Table 5.17. The newly proposed model improved the accuracy of the existing blending, voting, and stacking ensemble methods by a margin of 1%, 1.8%, and 3%, respectively,

Table 5.17: Performance results using bidirectional feature selection (TCC_CED).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
Augmented Features	**193.972**	**0.476**	**0.464**	**0.485**	**0.507**	**0.443**	**0.479**
Blending	31.586	0.458	0.451	0.478	0.493	0.415	0.452
Stacking	177.563	0.448	0.447	0.452	0.460	0.436	0.444
MLP	119.082	0.371	0.307	0.330	0.416	0.476	0.329
k-NN	8.484	0.337	0.311	0.295	0.336	0.363	0.379
RF	31.657	0.327	0.311	0.230	0.339	0.412	0.341
Voting	176.030	0.227	0.268	0.133	0.192	0.310	0.232
LightGBM	2.588	0.174	0.253	0.100	0.132	0.203	0.183
XGBoost	15.183	0.172	0.258	0.099	0.127	0.193	0.184

Utilizing the bidirectional feature selection on the GTZAN dataset (shown in Table 5.18) to choose a subset of features resulted in the augmented features exceeding other classification approaches at 73% and the k-NN model being the least accurate model at 68.2%. Once again, The proposed augmented features ensemble method improved accuracy, averaging a 2% increase compared to other ensemble learning models such as blending, voting, and stacking.

Table 5.18: Performance results using bidirectional feature selection (GTZAN).

Model	Time (s)	Mean Accuracy	First-fold	Second-fold	Third-fold	Forth-fold	Fifth-fold
Augmented Features	**4.236**	**0.730**	**0.750**	**0.750**	**0.700**	**0.750**	**0.700**
Blending	1.302	0.720	0.550	0.800	0.800	0.850	0.600
RF	0.769	0.716	0.730	0.750	0.660	0.710	0.730
MLP	0.988	0.712	0.770	0.700	0.640	0.720	0.730
Voting	3.452	0.712	0.740	0.710	0.700	0.690	0.720
Stacking	4.220	0.700	0.700	0.750	0.650	0.800	0.600
LightGBM	0.721	0.690	0.720	0.670	0.670	0.700	0.690
XGBoost	0.689	0.686	0.710	0.650	0.660	0.680	0.730
k-NN	0.031	0.682	0.730	0.680	0.630	0.640	0.730

Our results demonstrate that using the augmented features ensemble learning model can significantly enhance the accuracy of classification results, regardless of the utilization of feature selection techniques. In particular, employing the augmented features model has significantly impacted classification accuracy among other popular ensemble learning models. In the TCC_CED and GTZAN datasets, the augmented features model significantly improved classification accuracy compared to other ensemble learning models such as stacking, blending, and voting. On average, the proposed model outperformed these models by 2.7%, 2.5%, and 24.7% on the TCC_CED dataset and by 4.2%, 4.4%, and 3.7% on the GTZAN dataset. These results show that using the augmented features ensemble learning model can substantially benefit classification accuracy.

5.7 Choosing the Best Technique

The experiments on the Spotify dataset showed that the highest classification accuracy was achieved using the stacking ensemble with tree-based feature selection, as indicated in Table 5.19. However, a close examination of the results indicates that the augmented inputs and stacking ensembles were the most effective classification models for this dataset, producing almost similar results, with each having the highest accuracy 50% of the time.

Table 5.19: Choosing the best classification approach for the Spotify dataset.

Feature Selection Method	Model	Time (s)	Mean Accuracy
Tree-based Feature Selection	Stacking	322.476	0.868
Without Feature Selection	Augmented Features	331.253	0.865
Recursive Feature Selection	Augmented Features	340.587	0.865
Forward Feature Selection	Stacking	190.239	0.859
Backward Feature Selection	Stacking	190.924	0.859
Bidirectional Feature Selection	Augmented Features	211.397	0.857

The results from the TCC_CED dataset in Table 5.20 show a clear advantage of using the proposed augmented features for classification compared to other models. The findings indicate that the proposed model consistently performs better in all cases, regardless of whether a feature selection method is employed. Furthermore, the results establish that the augmented features ensemble delivers improved classification accuracy compared to other models, providing a robust and reliable performance. When analyzing the impact of feature selection on the results, it can be noticed that using a feature selection technique can result in a slight decrease in the overall mean accuracy.

Table 5.20: Choosing the best classification approach for the TCC_CED dataset.

Feature Selection Method	Model	Time (s)	Mean Accuracy
Without Feature Selection	Augmented Features	215.721	0.483
Recursive Feature Selection	Augmented Features	193.513	0.480
Backward Feature Selection	Augmented Features	202.789	0.480
Tree-based Feature Selection	Augmented Features	194.513	0.478
Forward Feature Selection	Augmented Features	194.652	0.476
Bidirectional Feature Selection	Augmented Features	193.972	0.476

Similarly, the results from the GTZAN dataset in Table 5.21 showed that using augmented features significantly improved the music genre classification accuracy. Specifically, the augmented features ensemble achieved the highest accuracy rate in 5 out of 6 cases, outperforming all other ensemble models. Furthermore, the best performance was achieved by combining the proposed ensemble with RFE and tree-based feature selection, resulting in an accuracy of 87%.

Table 5.21: Choosing the best classification approach for the GTZAN dataset.

Feature Selection Method	Model	Time (s)	Mean Accuracy
Recursive Feature Selection	Augmented Features	5.619	0.870
Tree-based Feature Selection	Augmented Features	5.633	0.870
Without Feature Selection	Augmented Features	6.416	0.840
Backward Feature Selection	MLP	2.479	0.748
Bidirectional Feature Selection	Augmented Features	4.236	0.730
Forward Feature Selection	Augmented Features	4.378	0.720

5.8 Discussion

Machine learning has become an increasingly popular approach in recent years, as it can potentially solve a wide range of classification problems, including music genre classification. In this context, ensemble learning has shown great promise in enhancing the accuracy of classification results. Ensemble learning methods combine the predictions of multiple base models to deliver a more accurate final prediction, and the approach has been used successfully in various applications. However, while researchers have widely used ensemble learning techniques, there are still opportunities to improve its performance.

Our study proposes a novel ensemble model that uses augmented features to address this challenge. The idea behind our proposed ensemble model is to create new sets of features that incorporate information from the base models in addition to the original feature set, which is achieved by computing the predicted probabilities for each class from each base model and then concatenating these probabilities with the original features in the dataset. The resulting augmented dataset can be used then to train a meta-model and produce a more accurate final prediction.

Our research shows that using the augmented features ensemble learning model can significantly enhance the accuracy of classification results, regardless of the utilization of feature selection techniques. The improvement is particularly evident in the TCC_CED and GTZAN datasets, where the augmented features model significantly improved classification accuracy compared to other ensemble learning models and outperformed them by a significant margin, demonstrating the effectiveness of our approach.

One possible reason why the augmented features ensemble technique beats other approaches is that it uses a more sophisticated method to combine the predictions of the base models. Stacking and blending, for example, average the predictions of the base models, while voting selects the class with the highest number of votes. Although these techniques can be effective, they do not include feature augmentation in the training process, which can limit their ability to capture the underlying patterns and variations in the data. By con-

trast, the augmented features ensemble technique creates new feature sets that incorporate information from the base models, allowing the meta-model to learn from the combined strengths of the base models and potentially capture more complex interactions between the features and the target variable, resulting in improved classification accuracy, as demonstrated in our results.

However, it is important to note that there were a few cases where the traditional models slightly outperformed our proposed ensemble model. This could be due to the fact that the feature augmentation did not provide any additional beneficial information beyond the original features. In such cases, the augmented features could add more noise or redundancy to the dataset, making it more difficult for the meta-model to learn the underlying patterns and relationships in the data. Therefore, it is essential to carefully evaluate the quality and usefulness of the augmented features before using them in the ensemble model.

Chapter 6

Conclusion

This chapter provides an overview of our research results and achievements. We aim to present the key insights gained from our study and highlight the contributions made to MIR. Moreover, we will discuss our objectives for future developments and enhancements to the methodology proposed in this study. We seek to continuously improve the proposed model and explore new opportunities to expand its applications in other fields.

6.1 Summary and Contribution of the Research

We proposed a new ensemble learning technique, which aims to enhance the feature representation of a dataset by incorporating the prediction probabilities from its base classifiers. This approach expands the traditional ensemble learning models by adding layers of information to the input data. Our findings indicated that this technique could significantly improve the performance of genre classification tasks. To assess the effectiveness of our proposed approach, we compared it to other popular ensemble methods, such as voting, stacking, and blending on three music datasets of varied sizes. The datasets included the Spotify, TCC_CED, and GTZAN datasets. These datasets were carefully picked to represent a diverse range of music genres and features and were designed to challenge the performance of other classification models.

In the Spotify dataset, our proposed augmented inputs model achieved a similar prediction accuracy with the stacking ensemble of over 85%, outperforming all other models. This result demonstrates the augmented inputs approach's superiority over traditional voting and

blending methods. Furthermore, for the TCC_CED dataset, the augmented inputs model outperformed other ensemble and base models and was selected as the best-performing model with a mean accuracy above 47%. Finally, for the GTZAN dataset, our proposed model surpassed all other models in several instances and achieved the highest accuracy of 87%.

Our results highlight the significance of the augmented inputs ensemble learning technique in machine learning and music information retrieval. The results from the experiments conducted in this research demonstrate our approach's capability to enhance genre classification tasks' performance. The findings of this study provide insights into the potential of this technique to improve the accuracy of music genre classification models significantly. Moreover, this research contributes to the broader field of ensemble learning by introducing a novel perspective on integrating additional information into the input representation of data. The results from this study suggest that the augmented inputs ensemble learning technique could be an effective tool for improving the performance of other machine learning tasks. This research provides a foundation for future work in this area and highlights the importance of incorporating additional information into the input representation of data in machine learning models.

6.2 Future Work

In future research, we plan to design and analyze other approaches for combining the prediction probabilities in our model and verify whether they affect its prediction performance. We also intend to expand the scope of our study by applying the augmented inputs ensemble model to diverse types of datasets beyond just music data. Using different datasets will allow us to verify the model's generalizability and effectiveness in dealing with distinctive features. In addition to exploring different datasets, we plan to investigate the impact of the dynamic selection of base classifiers on the performance of our ensemble model. We intend to experiment with a broader range of traditional machine learning algorithms and

newer deep learning models. By doing so, we can compare the effectiveness of different base classifiers and determine which ones perform best under different circumstances.

Another direction we plan to pursue is to investigate the scalability of our model. As data proliferates, developing models that can handle large datasets efficiently is essential. Therefore, we plan to explore strategies for optimizing the performance and computation time of the augmented inputs ensemble model on large datasets.

Our ultimate goal is to create a robust ensemble model that can adapt to different data types and improve prediction performance. Furthermore, we can comprehensively understand its capabilities and limitations by thoroughly exploring the model's generalizability and effectiveness in dealing with different features and classifiers. This knowledge will be valuable for future research and practical applications of the augmented inputs ensemble model, helping unlock its full potential in various fields.

Bibliography

[1] David Hesmondhalgh. *Why music matters*. John Wiley & Sons, 2013.

[2] John Connell and Chris Gibson. *Sound tracks: Popular music identity and place*. Routledge, 2003.

[3] Markus Schedl, Peter Knees, Brian McFee, Dmitry Bogdanov, and Marius Kaminskas. Music recommender systems. *Recommender systems handbook*, pages 453–492, 2015.

[4] Yu-Lung Lo and Yi-Chang Lin. Content-based music classification. In *2010 3rd International Conference on Computer Science and Information Technology*, volume 2, pages 112–116. IEEE, 2010.

[5] Fionn Murtagh. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5-6):183–197, 1991.

[6] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

[8] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[9] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[10] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[11] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[12] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[13] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. *Encyclopedia of database systems*, 5:532–538, 2009.

[14] Pedro Larranaga, Borja Calvo, Roberto Santana, Concha Bielza, Josu Galdiano, Inaki Inza, José A Lozano, Rubén Armananzas, Guzmán Santafé, Aritz Pérez, et al. Machine learning in bioinformatics. *Briefings in bioinformatics*, 7(1):86–112, 2006.

[15] Fredrik Olsson. A literature survey of active machine learning in the context of natural language processing. ., 2009.

[16] Michael Kassler. Toward musical information retrieval. *Perspectives of New Music*, pages 59–67, 1966.

[17] Carlos N Silla, Alessandro L Koerich, and Celso AA Kaestner. A machine learning approach to automatic music genre classification. *Journal of the Brazilian Computer Society*, 14(3):7–18, 2008.

[18] Rene Josiah M Quinto, Rowel O Atienza, and Nestor Michael C Tiglao. Jazz music sub-genre classification using deep learning. In *TENCON 2017-2017 IEEE Region 10 Conference*, pages 3111–3116. IEEE, 2017.

[19] Raul de Araújo Lima, Rômulo César Costa de Sousa, Hélio Lopes, and Simone Diniz Junqueira Barbosa. Brazilian lyrics-based music genre classification using a blstm network. In *International Conference on Artificial Intelligence and Soft Computing*, pages 525–534. Springer, 2020.

[20] Dmitry Bogdanov, Martín Haro Berois, Ferdinand Fuhrmann, Emilia Gómez Gutiérrez, Herrera Boyer, et al. Content-based music recommendation based on user preference examples. In *Anglade A, Baccigalupo C, Casagrande N, Celma Ò, Lamere P, editors. Workshop on Music Recommendation and Discovery 2010 (WOMRAD 2010); 2010 Sep 26; Barcelona, Spain. Aachen: CEUR Workshop Proceedings; 2010. p. 33-8*. CEUR Workshop Proceedings, 2010.

[21] Kazuyoshi Yoshii, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G Okuno. Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. In *ISMIR*, volume 6, pages 296–301, 2006.

[22] Hugo Siles Del Castillo and Djoerd Hiemstra. Hybrid content-based collaborative-filtering music recommendations. *Master's thesis, University of Twente, The Netherlands*, 2007.

[23] Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, 36(1):20–30, 2018.

[24] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. Deep learning techniques for music generation–a survey. *arXiv preprint arXiv:1709.01620*, 2017.

[25] Katherine E Eskine, Ashanti E Anderson, Madeline Sullivan, and Edward J Golob. Effects of music listening on creative cognition and semantic memory retrieval. *Psychology of Music*, 48(4):513–528, 2020.

[26] Donald Byrd and Tim Crawford. Problems of music information retrieval in the real world. *Information processing & management*, 38(2):249–272, 2002.

[27] Michael A Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008.

[28] Tao Li, Mitsunori Ogihara, and Qi Li. A comparative study on content-based music genre classification. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 282–289, 2003.

[29] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[30] Suresh Balakrishnama and Aravind Ganapathiraju. Linear discriminant analysis-a brief tutorial. *Institute for Signal and information Processing*, 18(1998):1–8, 1998.

[31] Aldona Rosner and Bozena Kostek. Automatic music genre classification based on musical instrument track separation. *Journal of Intelligent Information Systems*, 50(2):363–384, 2018.

[32] Daniel Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13, 2000.

[33] Ahmet Elbir, Hilmi Bilal Çam, Mehmet Emre Iyican, Berkay Öztürk, and Nizamettin Aydin. Music genre classification and recommendation by using machine learning techniques. In *2018 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pages 1–5. IEEE, 2018.

[34] Snigdha Chillara, AS Kavitha, Shwetha A Neginhal, Shreya Haldia, and KS Vidyullatha. Music genre classification using machine learning algorithms: a comparison. *Int Res J Eng Technol*, 6(5):851–858, 2019.

[35] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. Fma: A dataset for music analysis. *arXiv preprint arXiv:1612.01840*, 2016.

[36] Muhammad Asim Ali and Zain Ahmed Siddiqui. Automatic music genres classification using machine learning. *International Journal of Advanced Computer Science and Applications*, 8(8), 2017.

[37] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[38] G. Roop Sumanth and Uma Priyadarshini. Music genre classification using linear regression compared with extreme gradient boost algorithm with improved accuracy. *Journal of Pharmaceutical Negative Results*, pages 1659–1665, 2022.

[39] Sunil Kumar Prabhakar and Seong-Whan Lee. Holistic approaches to music genre classification using efficient transfer and deep learning techniques. *Expert Systems with Applications*, 211:118636, 2023.

[40] Yeshwant Singh and Anupam Biswas. Robustness of musical features on deep learning models for music genre classification. *Expert Systems with Applications*, 199:116879, 2022.

[41] Jessica Dias, Vaishak Pillai, Hrutvik Deshmukh, and Ashok Shah. Music genre classification & recommendation system using cnn. *Available at SSRN 4111849*, 2022.

[42] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[43] Weibin Zhang, Wenkang Lei, Xiangmin Xu, and Xiaofeng Xing. Improved music genre classification with convolutional neural networks. In *Interspeech*, pages 3304–3308, 2016.

[44] Swati A Patil and Thirupathi Rao Komati. Designing of a novel neural network model for classification of music genre. *Ingénierie des Systèmes d'Information*, 27(2), 2022.

[45] Shafaq Fatima Mughal, Shayan Aamir, Samarah Asghar Sahto, and Abdul Samad. Urdu music genre classification using convolution neural networks. In *2022 International Conference on Emerging Trends in Smart Technologies (ICETST)*, pages 1–6. IEEE, 2022.

[46] Wenli Wu, Fang Han, Guangxiao Song, and Zhijie Wang. Music genre classification using independent recurrent neural network. In *2018 Chinese Automation Congress (CAC)*, pages 192–195. IEEE, 2018.

[47] Jinliang Liu, Changhui Wang, and Lijuan Zha. A middle-level learning feature interaction method with deep learning for multi-feature music genre classification. *Electronics*, 10(18):2206, 2021.

[48] Nikki Pelchat and Craig M Gelowitz. Neural network music genre classification. *Canadian Journal of Electrical and Computer Engineering*, 43(3):170–173, 2020.

[49] Shikha Rani, Manoj Kaushik, and Vrinda Yadav. Identifying mood in music using deep learning. In *Artificial Intelligence and Technologies*, pages 571–578. Springer, 2022.

[50] Yu-Huei Cheng, Pang-Ching Chang, Duc-Man Nguyen, and Che-Nan Kuo. Automatic music genre classification based on crnn. *Engineering Letters*, 29(1), 2020.

[51] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5:64–67, 2001.

[52] Wei Wang and Mishal Sohail. Research on music style classification based on deep learning. *Computational and Mathematical Methods in Medicine*, 2022, 2022.

[53] Caifeng Liu, Lin Feng, Guochao Liu, Huibing Wang, and Shenglan Liu. Bottom-up broadcast neural network for music genre classification. *Multimedia Tools and Applications*, 80(5):7313–7331, 2021.

[54] Ugo Marchand and Geoffroy Peeters. The extended ballroom dataset. *HAL open science*, 2016.

[55] Dhevan S Lau and Ritesh Ajoodha. Music genre classification: A comparative study between deep learning and traditional machine learning approaches. In *Proceedings of Sixth International Congress on Information and Communication Technology*, pages 239–247. Springer, 2022.

[56] Ndiatenda Ndou, Ritesh Ajoodha, and Ashwini Jadhav. Music genre classification: A review of deep-learning and traditional machine-learning approaches. In *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pages 1–6. IEEE, 2021.

[57] Hareesh Bahuleyan. Music genre classification using machine learning techniques. *arXiv preprint arXiv:1804.01149*, 2018.

[58] Carlos N Silla Jr, Alessandro L Koerich, and Celso AA Kaestner. Feature selection in automatic music genre classification. In *2008 Tenth IEEE International Symposium on Multimedia*, pages 39–44. IEEE, 2008.

[59] Carlos Nascimento Silla Jr, Alessandro L Koerich, and Celso AA Kaestner. The latin music database. In *ISMIR*, pages 451–456, 2008.

[60] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, number 22 in 3, pages 41–46, 2001.

[61] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[62] Chris Sanden and John Z Zhang. Enhancing multi-label music genre classification through ensemble techniques. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 705–714, 2011.

[63] Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. Mulan: A java library for multi-label learning. *The Journal of Machine Learning Research*, 12:2411–2414, 2011.

[64] Dhanith Chathuranga and Lakshman Jayaratne. Musical genre classification using ensemble of classifiers. In *2012 Fourth International Conference on Computational Intelligence, Modelling and Simulation*, pages 237–242. IEEE, 2012.

[65] Paulo Ricardo Lisboa de Almeida, Eunelson José da Silva Júnior, Tatiana Montes Celinski, Alceu de Souza Britto, Luis Eduardo Soares de Oliveira, and Alessandro Lameiras Koerich. Music genre classification using dynamic selection of ensemble of classifiers. In *2012 IEEE international conference on systems, man, and cybernetics (SMC)*, pages 2700–2705. IEEE, 2012.

[66] Krittika Leartpantulak and Yuttana Kitjaidure. Music genre classification of audio signals using particle swarm optimization and stacking ensemble. In *2019 7th International Electrical Engineering Congress (iEECON)*, pages 1–4. IEEE, 2019.

[67] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.

[68] Loris Nanni, Yandre MG Costa, Rafael L Aguiar, Carlos N Silla Jr, and Sheryl Brahnam. Ensemble of deep learning, visual and acoustic features for music genre classification. *Journal of New Music Research*, 47(4):383–397, 2018.

[69] Anders Meng, Peter Ahrendt, Jan Larsen, and Lars Kai Hansen. Temporal feature integration for music genre classification. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(5):1654–1664, 2007.

[70] Jefferson Martins de Sousa, Eanes Torres Pereira, and Luciana Ribeiro Veloso. A robust music genre classification approach for global and regional music datasets evaluation. In *2016 IEEE International Conference on Digital Signal Processing (DSP)*, pages 109–113. IEEE, 2016.

[71] Andrés Eduardo Coca Salazar. Hierarchical mining with complex networks for music genre classification. *Digital Signal Processing*, 127:103559, 2022.

[72] Yangqiu Song, Changshui Zhang, and Shiming Xiang. Semi-supervised music genre classification. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 2, pages II–729. IEEE, 2007.

[73] Carlos N Silla and Alex A Freitas. Novel top-down approaches for hierarchical classification and their application to automatic music genre classification. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 3499–3504. IEEE, 2009.

[74] Quang H Nguyen, Trang TT Do, Thanh B Chu, Loan V Trinh, Dung H Nguyen, Cuong V Phan, Tuan A Phan, Dung V Doan, Hung N Pham, Binh P Nguyen, et al. Music genre classification using residual attention network. In *2019 International Conference on System Science and Engineering (ICSSE)*, pages 115–119. IEEE, 2019.

[75] Jan Hauke and Tomasz Kossowski. Comparison of values of pearson's and spearman's correlation coefficients on the same sets of data. *Quaestiones geographicae*, 30(2):87–93, 2011.

[76] Hassan Ezzaidi, Mohammed Bahoura, and Glenn Eric Hall. Towards a characterization of musical timbre based on chroma contours. In *Advanced Machine Learning Technologies and Applications: First International Conference, AMLTA 2012, Cairo, Egypt, December 8-10, 2012. Proceedings 1*, pages 162–171. Springer, 2012.

[77] A Shah, M Kattel, A Nepal, and D Shrestha. Chroma feature extraction. *Chroma Feature Extraction using Fourier Transform*, 2019.

[78] Meinard Müller. *Fundamentals of music processing: Audio, analysis, algorithms, applications*, volume 5. Springer, 2015.

[79] Sangeetha Rajesh and NJ Nalini. Musical instrument emotion recognition using deep recurrent neural network. *Procedia Computer Science*, 167:16–25, 2020.

[80] Melissa N Stolar, Margaret Lech, Shannon J Stolar, and Nicholas B Allen. Detection of adolescent depression from speech using optimised spectral roll-off parameters. *Biomedical Journal*, 2:10, 2018.

[81] RG Bachu, S Kopparthi, B Adapa, and BD Barkana. Separation of voiced and unvoiced using zero crossing rate and energy of the speech signal. In *American Society for Engineering Education (ASEE) zone conference proceedings*, pages 1–7. American Society for Engineering Education, 2008.

[82] Beth Logan et al. Mel frequency cepstral coefficients for music modeling. In *Ismir*, volume 270, page 11. Plymouth, MA, 2000.

[83] Salvador García, Sergio Ramírez-Gallego, Julián Luengo, José Manuel Benítez, and Francisco Herrera. Big data preprocessing: methods and prospects. *Big Data Analytics*, 1(1):1–22, 2016.

[84] C Reid Turner, Alfonso Fuggetta, Luigi Lavazza, and Alexander L Wolf. A conceptual basis for feature engineering. *Journal of Systems and Software*, 49(1):3–15, 1999.

[85] Stuart Borthwick and Ron Moy. *Popular music genres: An introduction*. Routledge, 2020.

[86] Noor Azilah Muda, Yun-Huoy Choo, and Norashikin Ahmad. Content-based feature selection for music genre classification. *International Journal of Computer Information Systems and Industrial Management Applications*, 2022.

[87] HongFang Zhou, JiaWei Zhang, YueQing Zhou, XiaoJie Guo, and YiMing Ma. A feature selection algorithm of decision tree based on feature weight. *Expert Systems with Applications*, 164:113842, 2021.

[88] Puneet Misra and Arun Singh Yadav. Improving the classification accuracy using recursive feature elimination with cross-validation. *International Journal on Emerging Technologies*, 11(3):659–665, 2020.

[89] S Visalakshi and V Radha. A literature review of feature selection techniques and applications: Review of feature selection in data mining. In *2014 IEEE International Conference on Computational Intelligence and Computing Research*, pages 1–6. IEEE, 2014.

[90] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46:389–422, 2002.

[91] Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 2013.

[92] Yap Bee Wah, Nurain Ibrahim, Hamzah Abdul Hamid, Shuzlina Abdul-Rahman, and Simon Fong. Feature selection methods: Case of filter and wrapper approaches for maximising classification accuracy. *Pertanika Journal of Science & Technology*, 26(1), 2018.

[93] Tae Kyun Kim. T test as a parametric statistic. *Korean journal of anesthesiology*, 68(6):540–546, 2015.

[94] Lars St, Svante Wold, et al. Analysis of variance (anova). *Chemometrics and intelligent laboratory systems*, 6(4):259–272, 1989.

[95] Alan Jović, Karla Brkić, and Nikola Bogunović. A review of feature selection methods with applications. In *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 1200–1205. Ieee, 2015.

[96] Sebastian Raschka. *Python machine learning*. Packt publishing ltd, 2015.

[97] Hassan Ramchoun, Youssef Ghanou, Mohamed Ettaouil, and Mohammed Amine Janati Idrissi. Multilayer perceptron: Architecture optimization and training. *International Journal of Interactive Multimedia and Artificial Intelligence*, 2016.

[98] Leonardo Noriega. Multilayer perceptron tutorial. *School of Computing. Staffordshire University*, 4:5, 2005.

[99] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[100] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316, 2017.

[101] D Richard Cutler, Thomas C Edwards Jr, Karen H Beard, Adele Cutler, Kyle T Hess, Jacob Gibson, and Joshua J Lawler. Random forests for classification in ecology. *Ecology*, 88(11):2783–2792, 2007.

[102] Pall Oskar Gislason, Jon Atli Benediktsson, and Johannes R Sveinsson. Random forests for land cover classification. *Pattern recognition letters*, 27(4):294–300, 2006.

[103] Xue-Wen Chen and Mei Liu. Prediction of protein–protein interactions using random decision forest framework. *Bioinformatics*, 21(24):4394–4400, 2005.

[104] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25(2):197–227, 2016.

[105] Stefano Nembrini, Inke R König, and Marvin N Wright. The revival of the gini importance? *Bioinformatics*, 34(21):3711–3718, 2018.

[106] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[107] Rongwei Song, Siding Chen, Bailong Deng, and Li Li. extreme gradient boosting for identifying individual users across different digital devices. In *International Conference on Web-Age Information Management*, pages 43–54. Springer, 2016.

[108] Ahmedbahaaaldin Ibrahem Ahmed Osman, Ali Najah Ahmed, Ming Fai Chow, Yuk Feng Huang, and Ahmed El-Shafie. Extreme gradient boosting (xgboost) model to predict the groundwater levels in selangor malaysia. *Ain Shams Engineering Journal*, 12(2):1545–1556, 2021.

[109] Tuong Le, Bay Vo, Hamido Fujita, Ngoc-Thanh Nguyen, and Sung Wook Baik. A fast and accurate approach for bankruptcy forecasting using squared logistics loss with gpu-based extreme gradient boosting. *Information Sciences*, 494:294–310, 2019.

[110] Benjamin Murauer and Günther Specht. Detecting music genre using extreme gradient boosting. In *Companion proceedings of the the web conference 2018*, pages 1923–1927, 2018.

[111] Essam Al Daoud. Comparison between xgboost, lightgbm and catboost using a home credit dataset. *International Journal of Computer and Information Engineering*, 13(1):6–10, 2019.

[112] Qi Gou, Jianrong Ban, and Bingyan Qin. Research on the evolution and revolutionary trend of music artists and genres based on lightgbm. *Frontiers in Economics and Management*, 3(4):440–454, 2022.

[113] Huafeng Zeng, Qiang Yuan, Li Guo, and Shibiao Xu. Song popularity prediction model based on multi-modal feature fusion and lightgbm. In *2022 the 8th International Conference on Communication and Information Processing*, pages 28–32, 2022.

[114] Daniel Boxler. *Machine Learning Techniques Applied to Musical Genre Recognition*. PhD thesis, University of Colorado Colorado Springs, 2020.

[115] Oliver Kramer and Oliver Kramer. K-nearest neighbors. *Dimensionality reduction with unsupervised nearest neighbors*, pages 13–23, 2013.

[116] Oliver Sutton. Introduction to k nearest neighbour classification and condensed nearest neighbour data reduction. *University lectures, University of Leicester*, 1, 2012.

[117] Zhongheng Zhang. Introduction to machine learning: k-nearest neighbors. *Annals of translational medicine*, 4(11), 2016.

[118] Sayali D Jadhav and HP Channe. Comparative study of k-nn, naive bayes and decision tree classification techniques. *International Journal of Science and Research (IJSR)*, 5(1):1842–1845, 2016.

[119] Robi Polikar. Ensemble learning. In *Ensemble machine learning*, pages 1–34. Springer, 2012.

[120] Ross D King, Oghenejokpeme I Orhobor, and Charles C Taylor. Cross-validation is safe to use. *Nature Machine Intelligence*, 3(4):276–276, 2021.

[121] Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*, 2020.

[122] Sofia Visa, Brian Ramsay, Anca L Ralescu, and Esther Van Der Knaap. Confusion matrix-based feature selection. *Maics*, 710(1):120–127, 2011.