MUSIC GENRE CLASSIFICATION

A PROJECT SUBMITTED FOR THE PARTIAL FULFILMENT OF THE REQUIREMENTS OF THE HIGHER NATIONAL DIPLOMA IN DATA SCIENCE PROGRAMME

BY

D. JAYENDRA PRABHASHI MAHAWATTA

COHNDDS (F/T)211F-017

INDEPENDENT RESEARCH PROJECT

HIGHER NATIONAL DIPLOMA IN DATA SCIENCE

NATIONAL INSTITUTE OF BUSINESS MANAGEMENT

COLOMBO, SRI LANKA

09TH MAY 2023

DECLARATION

I hereby declare that the work presented in this project report was carried out independently by myself and have cited the work of others and given due reference diligently.

Date: /0 /2023

D.J. Prabhashi Mahawatta Signature of the Candidate

I certify that the above student carried out his/her project under my supervision and guidance.

Date: …… …….…….

W.M.S.G.D.C.Wanigasekara Signature of the Supervisor

ACKNOWLEDGEMENT

The completion of this paper would not have been possible without the presence of a group of people who allocated and extended their valuable time and assistance throughout the course of this research.

I would first like to express my sincere gratitude to my parents who gave me loving care and unstoppable encouragement throughout this research study as well as throughout my life to make me a productive scholar.

I would like to express my heartfelt gratitude to the lecturers of all the modules that have been completed so far, for providing me with all the necessary and imperative knowledge and skills without which this report could not have been materialized.

I would also like to express my sincere gratitude to my supervisor Associate Professor, Miss. W.M.S.G.D.C. Wanigasekara of the National Innovation Center at National Institute of Business Management, and Mr. Thurairasa Balakumar, the Course Director of my degree program for his complete and unrelenting support with the completion of this paper.

Finally, I would also like to take this opportunity to sincerely thank all my colleagues and everyone else who provided insight and expertise that assisted the research in any manner possible, even though I have not mentioned them explicitly here.

ABSTRACT

In today's world, where people are attached to their phones and air pods, listening to music becomes a mundane part of our lives. It occurs at times where we find particular songs catchy due to their pitch, choice of pattern, lyrics and much more! Hence in recommendation systems implemented in apps such as Spotify, classifying the music according to genres becomes very important to enhance user experience. This paper aims to chart out various methods and parameters essential in the classification process, with use of Machine Learning and Deep Learning.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 01 - INTRODUCTION

1.1 Background

When reading the header, the first question that comes to mind is, "What is a music genre?" A category, or more specifically a conventional category, that recognizes the qualities or aspects of a sub-division of the music file belonging to a traditional or any conventional recognized music form, is referred to as a "music genre."

The term "music genre classification" refers to the categorizing of musical samples. A music genre classifier is necessary for early song evaluation; for instance, if a new song has just been recorded, it will help classify the song into its typical category. A song's genre can be determined by its particular aural characteristics, allowing its contents to be evaluated in connection to the created wave signals. Another important component is identifying the timbral features of the instruments used in the song. This is essential for determining the genre of music based on the type of instrument played and helps us determine the song's historical context. Music genre classification is the task of automatically categorizing music into different genres based on its audio features. It has been an active area of research in the field of music information retrieval (MIR) and machine learning.

The classification of music into genres is subjective and can vary depending on cultural, historical, and individual perspectives. Genres can be broadly categorized into popular genres such as rock, pop, hip-hop, jazz, classical, electronic, country, and many more. Each

genre has its own unique characteristics, including rhythmic patterns, instrumentation, melodic structures, and lyrical themes.

Traditionally, music genre classification was done manually by experts who listened to the music and assigned it to specific genres. With the advancements in machine learning and digital signal processing techniques, it has become possible to automate this process using computational algorithms.

There are different approaches to music genre classification, but the most common one involves extracting audio features from the music signals and using machine learning algorithms to classify them. Some of the commonly used audio features include spectral features (such as pitch, timbre, and spectral centroid), rhythmic features (such as tempo and beat), and tonal features (such as chord progression and harmony).

In this paper, I have used Random Forest (RF), AdaBoost, Gradient Boosting Machine (GBM), Extreme Gradient Boosting (XGB) and CatBoost machine learning algorithms and deep learning model to see how better we can differentiate one genre from another. Researchers and practitioners often use large datasets of labeled music samples to train these machine learning models. These datasets can include audio recordings from various genres, which are manually annotated with their corresponding genre labels. It is worth noting that music genre classification is still a challenging task, as there can be significant overlap and ambiguity between different genres. Additionally, the same song can exhibit elements of multiple genres, making it difficult to assign a single label. However, with the continued advancements in machine learning techniques and the availability of large music databases, the accuracy of music genre classification systems has been improving.

1.2 Research Problem

Currently, genre labeling is done manually by individuals using their own musical expertise. This activity has not yet been automated by conventional computing techniques since the distinctions between musical genres are largely arbitrary and subjective.

Genre classification is a difficult operation that is best handled by machine intelligence. Given enough audio data of which enormous quantities may be easily gathered from freely available music online machine learning can discover and forecast utilizing these ill-defined patterns. Music genre classification has been extensively studied and applied, but there are several research challenges and opportunities for improvement in this domain. The primary research problem in music genre classification is to enhance the accuracy and robustness of genre prediction models. Some specific aspects of this problem include:

1. Handling Genre Ambiguity: Music genres can be subjective and have overlapping characteristics, making it challenging to categorize certain songs accurately. For example, some songs may exhibit elements of multiple genres or belong to emerging or hybrid genres that do not fit neatly into predefined categories. The research problem is to develop techniques that can handle genre ambiguity and make more nuanced genre predictions.
2. Feature Selection and Representation: The choice of audio features and their representation plays a crucial role in genre classification accuracy. The research problem is to identify the most informative features and develop effective methods for representing them to capture the diverse characteristics of different genres accurately. This includes exploring novel audio features, considering temporal

dependencies, and integrating multiple modalities such as lyrics, metadata, and social context.

3. Scalability and Generalization: Many existing genre classification models have been trained and evaluated on specific datasets, limiting their scalability and generalization to new music samples. The research problem is to develop models that can scale well to large music collections and generalize effectively to different music styles, eras, and cultural contexts. This involves addressing issues such as data bias, domain adaptation, and transfer learning.
4. Cross-Genre Classification: Traditional music genre classification focuses on predicting a single genre label for each song. However, there is a growing interest in understanding and capturing cross-genre influences and relationships. The research problem is to explore methods that can handle cross-genre classification, where songs may belong to multiple genres simultaneously or exhibit transitional characteristics between genres.
5. User Preferences and Context: Music genre classification often assumes a one-size-fits-all approach, ignoring individual user preferences and contextual factors. The research problem is to develop personalized genre classification models that can consider user-specific preferences, cultural backgrounds, and contextual factors to provide more tailored music recommendations and better user experiences.

By addressing these research problems can make significant advancements in music genre classification, leading to more accurate, robust, and personalized systems that enhance

music recommendation, content organization, and user experiences in various music-related applications.

1.3 Objectives of the Project

The goal of this project is to develop a proof-of-concept music genre classifier that, using a deep learning approach, can correctly identify the genre and confidence level of Western music from four candidate genres (classical, jazz, rap, and rock).

The primary objective is to develop models and algorithms that can accurately classify music into different genres. The aim is to achieve high classification accuracy, minimizing misclassifications and correctly identifying the predominant genre or genres associated with a music piece.

1.4 Research Questions

How can genre ambiguity be effectively addressed in music genre classification? What techniques and models can capture the nuances and complexity of songs that exhibit elements of multiple genres or belong to emerging or hybrid genres? and

what are the most informative audio features and representations for accurate genre classification? How can temporal dependencies and the integration of multiple modalities such as lyrics, metadata, and social context enhance genre classification performance?

1.5 Scope of the Research

The research can explore different audio features and representations for genre classification, such as spectral features, rhythm patterns, and tempo-related features and include defining appropriate evaluation metrics to measure the performance of the genre classification models. Accuracy, precision, recall, F1 score, and confusion matrices are common metrics used in genre classification. The scope may involve conducting rigorous performance analysis, comparing different models, and assessing their strengths and limitations. If interpretability and explain ability are within the scope, the research explores methods to provide insights into the decision-making process of genre classification models and this include analyzing feature importance, visualization techniques, or generating explanations for genre predictions.

1.6 Justification of the Research

Music genre classification has numerous practical applications in the field of music information retrieval (MIR) and related domains. Accurate genre classification can enhance music recommendation systems, playlist generation, content organization in digital music libraries, and personalized music experiences furthermore Music genre classification has implications for the music industry, including artist discovery, marketing, and content distribution. Accurate genre identification can assist in identifying emerging trends, understanding audience preferences, and developing targeted marketing strategies. By improving the accuracy and interpretability of genre classification models, researchers

can provide valuable insights for the music industry and contribute to the success and promotion of artists and music labels.

Music genres are not only a means of categorizing music but also carry cultural and societal significance. Understanding and accurately classifying music genres can help preserve cultural heritage, explore music evolution, and study cross-cultural influences. By conducting research in music genre classification, researchers can contribute to the broader understanding of music and its cultural and societal impacts.

1.7 Expected Limitations

Music genres can be subjective and have overlapping characteristics, making it challenging to define clear boundaries between genres. This subjectivity and ambiguity may introduce inherent limitations in the classification process, as different individuals may have varying interpretations of genre classifications.

Music genres evolve over time, and their characteristics can vary based on cultural, regional, and temporal contexts. Genre classification models trained on specific datasets may not capture these variations adequately. Music genre preferences can vary significantly among individuals, and the contextual factors surrounding music consumption can influence genre perceptions. Personalizing genre classification models based on individual preferences and contextual factors is a complex task.

CHAPTER 02 - LITERATURE REVIEW

2.1 Introduction to the Research Theme

Music genre classification is a significant area of research within the field of music information retrieval (MIR). The ability to automatically categorize music into different genres has numerous practical applications, ranging from music recommendation systems and content organization to artist discovery and targeted marketing strategies. By accurately classifying music into genres, researchers can enhance user experiences, facilitate personalized music recommendations, and provide valuable insights to the music industry.

However, music genre classification poses several challenges due to the subjective and evolving nature of genres, the presence of genre ambiguity and hybrid genres, and the complexity of music representations. These challenges necessitate the development of robust and accurate genre classification models that can capture the diversity and nuances of music genres effectively.

The research theme of music genre classification aims to address these challenges and advance the understanding and techniques in this field. By exploring various research problems, objectives, and methodologies, I seek to develop models and algorithms that can accurately identify and classify music into different genres. These models need to be scalable, efficient, and adaptable to handle large-scale music collections and diverse genres.

2.2 Theoretical Explanation about the Key Words in the Topic

MIR - Music Information Retrieval

It is an interdisciplinary field that combines concepts from musicology, signal proces sing, computer science, and artificial intelligence to develop techniques and systems for o rganizing, analyzing, and retrieving music-related information.

CHROMA_STFT - Chroma Short-Time Fourier Transform

CHROMA_STFT is a commonly used audio feature representation in the field of mu sic information retrieval and music analysis. It is particularly useful for capturing the har monic content and tonal characteristics of music. The Chroma feature represents the distri bution of musical pitches within an audio signal. It quantifies the presence and intensity o f each pitch class over time. This representation is derived from the Short-Time Fourier T ransform (STFT) of the audio signal. The STFT is a time-frequency analysis technique th at divides an audio signal into short overlapping frames and performs the Fourier Transfo rm on each frame to obtain the frequency content over time. The STFT produces a spectr ogram, which represents the magnitude or power of different frequency components at di fferent time intervals.

RMS - Root Mean Square

RMS is a mathematical measure used to quantify the average magnitude or amplitude of a signal. In the context of audio and signal processing, RMS is commonly used as a fea ture to represent the energy or power of an audio signal. The RMS value provides a meas ure of the overall amplitude or intensity of the signal within the specified window. It is oft en used to characterize the loudness or energy level of audio signals

SPECTRAL_CENTROID

Spectral Centroid is a feature commonly used in audio signal processing and music a nalysis. It provides information about the "center of mass" or average frequency content o f a signal's spectrum. The spectral centroid is computed by weighting the frequencies of th e spectral components (such as those obtained from the Fourier Transform) by their magn itudes and then finding the weighted average. Spectral Centroid can help differentiate bet ween bright and dark sounds, distinguish different instrument types based on their spectra l characteristics, or contribute to identifying specific genres known for their particular ton al properties.

SPECTRAL_BANDWIDTH

Spectral bandwidth is a feature commonly used in audio signal processing to charact erize the width or spread of the frequency content in a signal. It provides information abo ut the range of frequencies present in a particular portion of the audio spectrum. Specifica lly, spectral bandwidth measures the average width of the spectrum around the center freq uency of a signal. It indicates how concentrated or dispersed the energy is across the frequ ency range. A higher spectral bandwidth suggests a broader spread of frequencies, while a lower spectral bandwidth indicates a more concentrated or narrow frequency range. spectr al bandwidth is a measure of the spread or width of the frequency content in an audio sign al

ROLLOFF

Roll-off, also known as Spectral Roll-off, is a feature commonly used in audio signal processing to characterize the high-frequency content of a signal's spectrum. It represents the frequency below which a certain percentage of the total energy of the spectrum is con centrated. Roll-off indicates the frequency below which the specified percentage of the en

ergy in the spectrum is concentrated. It is often used as an indicator of the upper limit of t he spectral content and can provide insights into the timbral or tonal characteristics of a si gnal. And it can help distinguish between sounds with different degrees of high-frequency content, identify certain instrument types based on their spectral profiles, or differentiate b etween speech and non-speech signals.

ZCR - Zero Crossing Rate

ZCR is a feature commonly used in audio signal processing and music analysis to ch aracterize the rate of sign changes in a signal. It provides information about the temporal v ariations or fluctuations in the waveform of an audio signal. The zero-crossing rate can co ntribute to the characterization of the timbre or texture of a sound. Sounds with a higher z ero crossing rate may have a more noisy or edgy quality, while sounds with a lower rate m ay have a smoother or more sustained quality. The zero-crossing rate can help differentiat e between voiced and unvoiced speech segments and it can be used to identify percussive or rhythmic events.

HARMONY

Harmony refers to the simultaneous sounding of multiple pitches or notes in music. I t is the combination of different pitches played or sung together, creating chords, chord pr ogressions, and tonal relationships. Harmony plays a crucial role in shaping the overall m usical structure, texture, and emotional character of a composition.

TEMPO

Tempo refers to the speed or pace of a musical composition. It determines how fast o r slow the beats or pulses of the music are perceived. Tempo is a crucial element in music that helps establish the overall rhythmic feel and energy of a piece. Tempo is typically ind

icated by a specific numerical value, known as the beats per minute (BPM), which represe nts the number of beats or pulses occurring within one minute. For example, a tempo mar king of "120 BPM" indicates that there are 120 beats or pulses in the music per minute. T empo is typically set by the composer or conductor and serves as a guide for performers t o maintain a consistent speed throughout the piece. It helps establish the overall mood, int ensity, and character of the music. Different tempos can evoke various emotions and creat e different musical effects.

MFCC - Mel Frequency Cepstral Coefficients

MFCCs are derived from the spectrogram of an audio signal and capture important acoustic characteristics related to human auditory perception. MFCCs have proven to be r obust and informative features, as they capture essential aspects of the audio signal while reducing the dimensionality of the feature space. They are effective in representing the ti mbral characteristics and phonetic content of the audio signal, as they focus on perceptual ly relevant features by using the Mel scale.

2.3 Findings by Other Researchers

Tzanetakis, G., & Cook, P. [1]. influential study proposed a genre classification framework based on audio features such as spectral centroid, zero-crossing rate, and Mel-frequency cepstral coefficients (MFCCs). The authors achieved high accuracy using machine learning techniques such as support vector machines (SVM) and k-nearest neighbors (KNN).

Yang, Y. H., Lin, Y. A., Chen, H. H., & Su, M. C. [2] explored the use of lyrics as a modality for music genre classification. The study utilized natural language processing techniques to extract features from song lyrics and employed machine learning algorithms to classify music genres. The results showed the potential of lyrics as an additional source of information for genre classification.

Li, Y., Ogihara, M., & Li, Q. [3]. present a comparative study of various content-based approaches for music genre classification, including low-level audio features, high-level audio features, and metadata-based features. The authors evaluated the performance of different classification algorithms and provided insights into the strengths and limitations of each approach.

Toffetti, A., & Orio, N. [4] investigated the use of sparse representations, specifically sparse coding and sparse dictionary learning, for music genre classification. The study demonstrated that sparse representations of audio features can improve genre classification accuracy compared to traditional feature-based approaches.

Li, T., Ogihara, M., & Li, Q. [5] s' comprehensive survey provided an overview of various techniques and approaches for music genre classification. It covered feature extraction methods, machine learning algorithms, and evaluation metrics used in the field. The

authors discussed the challenges and future directions in music genre classification research.

Pachet, F., & Cazaly, D. [6] focused on music similarity measures, which are closely related to music genre classification. The authors proposed novel measures that take into account both low-level audio features and high-level semantic information. The findings demonstrated improved music recommendation accuracy using these enhanced similarity measures.

Han, J., & Chen, S. [7] explored the application of deep learning techniques, specifically convolutional neural networks (CNNs), for music genre classification. The study demonstrated that CNNs can effectively learn hierarchical representations from audio spectrograms and achieve high genre classification accuracy.

Dieleman, S., & Schrauwen, B. [8] introduced a deep learning approach for music audio analysis, which combined feature learning and classification into an end-to-end learning framework. The authors demonstrated the effectiveness of this approach for music genre classification tasks.

The work of Yang, Y. H., & Chen, H. H. [9] focused on music emotion classification, which is closely related to music genre classification. The authors proposed a fuzzy-based approach that considered emotional dimensions for classifying music into different emotional categories. The findings highlighted the importance of considering emotional aspects in genre classification tasks.

Koutamanis, A., & Marchini, M. [10]. This study explored the use of multi-modal data, combining audio spectrograms and song lyrics, for music genre classification. The authors

employed convolutional neural networks (CNNs) to jointly learn features from both modalities, resulting in improved genre classification performance compared to using single-modal data.

2.4 The Research Gap

The majority of research in music genre classification has focused on Western popular music genres. There is a need for research that explores genre classification in non-Western music traditions, considering cultural variations, distinctive features, and genre taxonomies specific to those traditions.

The evaluation of genre classification models relies on the availability of benchmark datasets and appropriate evaluation metrics. However, there is a lack of standardized benchmark datasets that cover a wide range of genres and represent the diversity of musical styles.

2.5 Table for Variables

VARIABLE NAME	DEFINITIONS
filename	Name of the audio clip
length	Length
chroma_stft_mean	Mean value of chroma_stft_mean[page]
chroma_stft_var	Variance of chroma_stft_var

rms_mean	Mean value of Root Mean Square[page]
rms_var	Variance of Root Mean Square
spectral_centroid_mean	Mean value of spectral_centroid[page]
spectral_centroid_var	Variance of spectral_centroid
spectral_bandwidth_mean	Mean value of spectral_bandwidth
spectral_bandwidth_var	Variance of spectral_bandwidth
rolloff_mean	Mean value of rolloff[page]
rolloff_var	Variance of rolloff
zero_crossing_rate_mean	Mean value of zero_crossing_rate[page]
zero_crossing_rate_var	Variance of zero_crossing_rate
harmony_mean	Mean value of harmony[page]
harmony_var	Variance of harmony
perceptr_mean	Mean value of perceptr
perceptr_var	Variance of perceptr
tempo	Tempo[page]of the song
mfcc1_mean	Mean value of Mel Frequency Cepstral Coefficients[page]
mfcc1_var	Variance of Mel Frequency Cepstral Coefficients[page]
label	Labels of genres

2.1 Table

CHAPTER 03 - DATA PREPARATION PROCESS - DATA PREPROCESSING AND DATA WRANGLING

For this paper, I have used the GTZAN Dataset which consists of 10 genres, The benefit of having the genres mentioned respective to each class of audio files is it makes it easier to label the audio files with their genre name, which will help in training the model.

The research utilized a pre-processed dataset without errors, missing values, outliers, inconsistencies or unnecessary variables which called the GTZAN dataset, obtained from Kaggle. The dataset consists of audio files from 10 different genres, with 100 audio files per genre. Each audio file has a fixed duration of 30 seconds, providing a comprehensive collection of music samples for genre classification. In addition to the audio files, the dataset includes visual representations in the form of Mel spectrograms. These spectrograms enable the application of convolutional neural networks (CNNs) for genre classification.

Furthermore, the dataset includes two CSV files containing various audio features. One CSV file provides mean and variance values computed over multiple features extracted from each 30-second audio file. The other CSV file follows the same structure but contains features computed from 3-second segments of the audio files. By utilizing the CSV file with the 3-second audio segments, the research focuses on using shorter audio segments to train and evaluate the genre classification models.

CHAPTER 04 - METHODOLOGY

4.1 INTRODUCTION

The research utilized a pre-processed dataset called the GTZAN dataset, obtained from Kaggle. The dataset consists of audio files from 10 different genres, with 100 audio files per genre. Each audio file has a fixed duration of 30 seconds, providing a comprehensive collection of music samples for genre classification.

The dataset includes two CSV files containing various audio features. One CSV file provides mean and variance values computed over multiple features extracted from each 30-second audio file. The other CSV file follows the same structure but contains features computed from 3-second segments of the audio files. By utilizing the CSV file with the 3-second audio segments, the research focuses on using shorter audio segments to train and evaluate the genre classification models.

For the machine learning models, the research employs ensemble algorithms, specifically:

Random Forest (RF): A decision tree-based ensemble algorithm that creates multiple decision trees and combines their predictions to make the final classification.

AdaBoost: A boosting algorithm that combines weak classifiers into a strong classifier. It assigns weights to each training example to emphasize the misclassified instances, allowing subsequent classifiers to focus on these instances.

Gradient Boosting Machine (GBM): Another boosting algorithm that builds multiple decision trees in a stage-wise manner. Each tree is trained to correct the mistakes made by the previous trees, resulting in improved accuracy.

Extreme Gradient Boosting (XGB): A powerful gradient boosting algorithm that optimizes the performance of gradient boosting machines. It includes additional regularization techniques to prevent overfitting and enhance generalization.

CatBoost: A gradient boosting algorithm that handles categorical features efficiently. It incorporates novel techniques to handle categorical data and provides strong performance in classification tasks.

These ensemble algorithms were selected to explore their performance in music genre classification based on the pre-processed dataset. By comparing the results of these algorithms, the research aims to identify the most effective approach for accurately classifying music genres based on the provided audio features from the CSV file.

The research methodology combines the utilization of pre-processed audio data, visual representations in the form of Mel spectrograms, and ensemble algorithms to conduct the genre classification experiments. The performance of the algorithms will be evaluated based on appropriate metrics such as accuracy, precision, recall, and F1 score to assess the effectiveness of each algorithm in the genre classification task.

4.2 POPULATION, SAMPLE AND SAMPLING TECHNIQUE

Population: The population in this research refers to the entire collection of audio files in the GTZAN dataset. It includes audio files from 10 different genres, with 100 audio files per genre, resulting in a total population size of 1000 audio files.

Sample: The sample in this research refers to a subset of the population that is selected for analysis and model training. Specifically, the research utilizes the CSV file containing features computed from 3-second audio segments. This sample represents a smaller subset of the population and allows for more manageable data processing and modeling.

Sampling Technique: The sampling technique employed in this research is not explicitly mentioned from author of the dataset. However, it is assumed that a random sampling approach may have been used to select the 3-second audio segments from the original audio files. Random sampling ensures that each audio segment has an equal chance of being selected, reducing bias and increasing the representativeness of the sample.

4.3 METHODS, TECHNIQUES AND TOOLS

I used various machine learning algorithms for the classification, including;

Decision trees and ensemble methods like Gradient Boosting, AdaBoosting, XGB and CatBoosting. And for model training and evaluation, the dataset is divided into training and testing sets to train the music classification model and evaluate its performance. Techniques like cross-validation and train-test split been used. During model training, the chosen classification algorithm is fitted to the training data, learning the patterns and

relationships between the features and their corresponding genres. The trained model is then evaluated using the testing data to assess its accuracy, precision, recall, F1 score, and other relevant metrics. To improve the performance of the music classification model, optimization techniques like Hyperparameter tuning been applied.

Furthermore, I used one deep learning model for a better accuracy. By training this ANN model on a labeled dataset of music samples with corresponding genres or labels, it can learn to classify new music samples into one of the predefined classes or genres based on their extracted features.

There are various tools and libraries available for implementing music classification models, such as: Python, R, Weka, MATLAB for this research I used Python:Popular libraries for ML and DL in music classification include scikit-learn, librosa, TensorFlow, Keras, PyTorch etc.

CHAPTER 05 - DATA ANALYSIS, VISUALIZATION, MODELING AND INTERPRETATION

5.1 Data Analysis

	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean
count	9990.0	9990.000000	9990.000000	9990.000000	9.990000e+03	9990.000000
mean	66149.0	0.379534	0.084876	0.130859	2.676388e-03	2199.219431
std	0.0	0.090466	0.009637	0.068545	3.585628e-03	751.860611
min	66149.0	0.107108	0.015345	0.000953	4.379535e-08	472.741636
25%	66149.0	0.315698	0.079833	0.083782	6.145900e-04	1630.680158
50%	66149.0	0.384741	0.085108	0.121253	1.491318e-03	2208.628236
75%	66149.0	0.442443	0.091092	0.176328	3.130862e-03	2712.581884
max	66149.0	0.749481	0.120964	0.442567	3.261522e-02	5432.534406

	spectral_centroid_var	spectral_bandwidth_mean	spectral_bandwidth_var	rolloff_mean
count	9.990000e+03	9990.000000	9.990000e+03	9990.000000
mean	4.166727e+05	2241.385959	1.182711e+05	4566.076592
std	4.349644e+05	543.854449	1.013505e+05	1642.065335
min	8.118813e+02	499.162910	1.183520e+03	658.336276
25%	1.231961e+05	1887.455790	4.876553e+04	3378.311110
50%	2.650692e+05	2230.575595	8.996072e+04	4631.377892
75%	5.624152e+05	2588.340505	1.585674e+05	5591.634521

max	mfcc16_mean	mfcc16_var	mfcc17_mean	mfcc17_var	mfcc18_mean	mfcc18_var
count	9990.000000	9990.000000	9990.000000	9990.000000	9990.000000	9990.000000
mean	1.448240	49.988755	-4.198706	51.962753	0.739943	52.488851
std	5.735149	34.442816	5.677379	36.400669	5.181313	38.177120
min	-26.850016	1.325786	-27.809795	1.624544	-20.733809	3.437439
25%	-2.227478	29.584894	-7.951722	29.863448	-2.516638	29.636197
50%	1.461623	41.702393	-4.443021	42.393583	0.733772	41.831377
75%	5.149752	59.274619	-0.726945	61.676964	3.888734	62.033906
max	39.144405	683.932556	34.048843	529.363342	36.970322	629.729797

5.1 Table

5.2 Data Visualization

Audio data Visualization

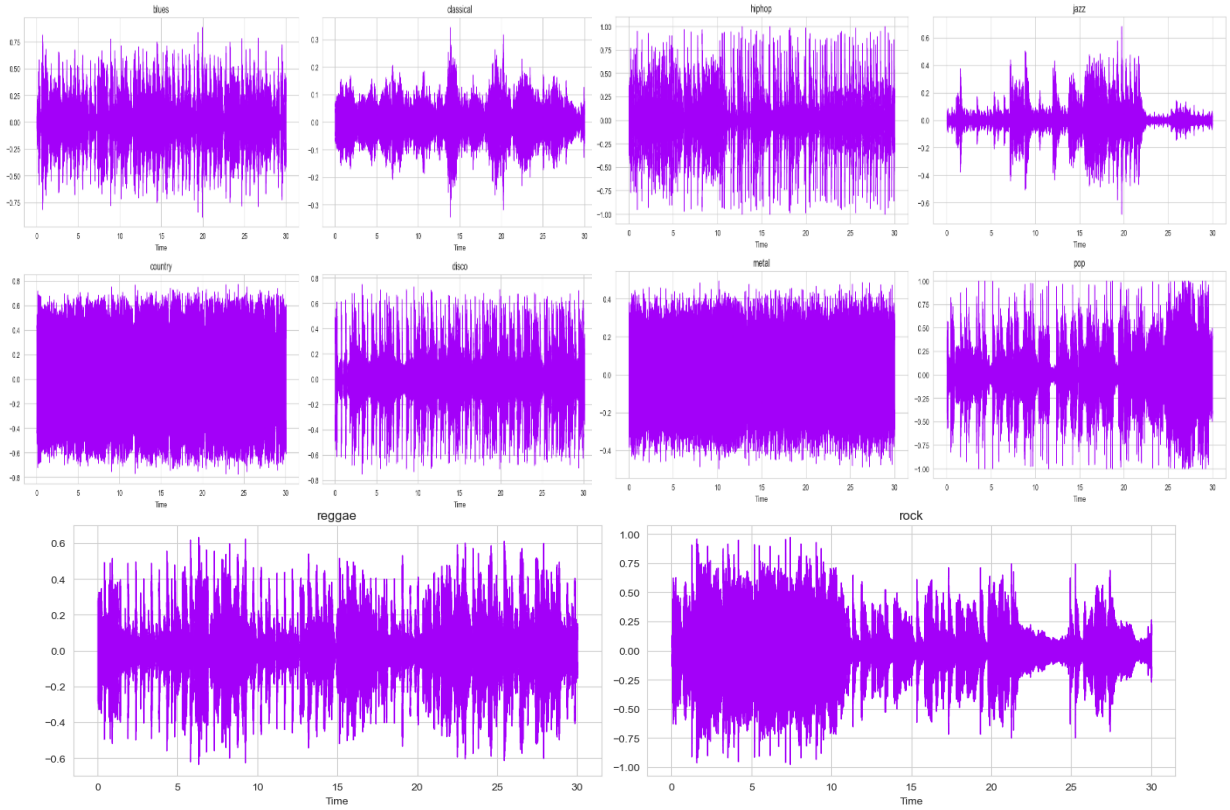


Figure 5.1

The waveform plot provides a visual representation of how the audio signal varies over time. It displays the changes in air pressure or voltage level of the audio signal as it progresses. The vertical position of the waveform indicates the amplitude or intensity of the audio signal at a given time. Higher positions represent higher amplitudes, while lower positions correspond to lower amplitudes.

Spectrogram

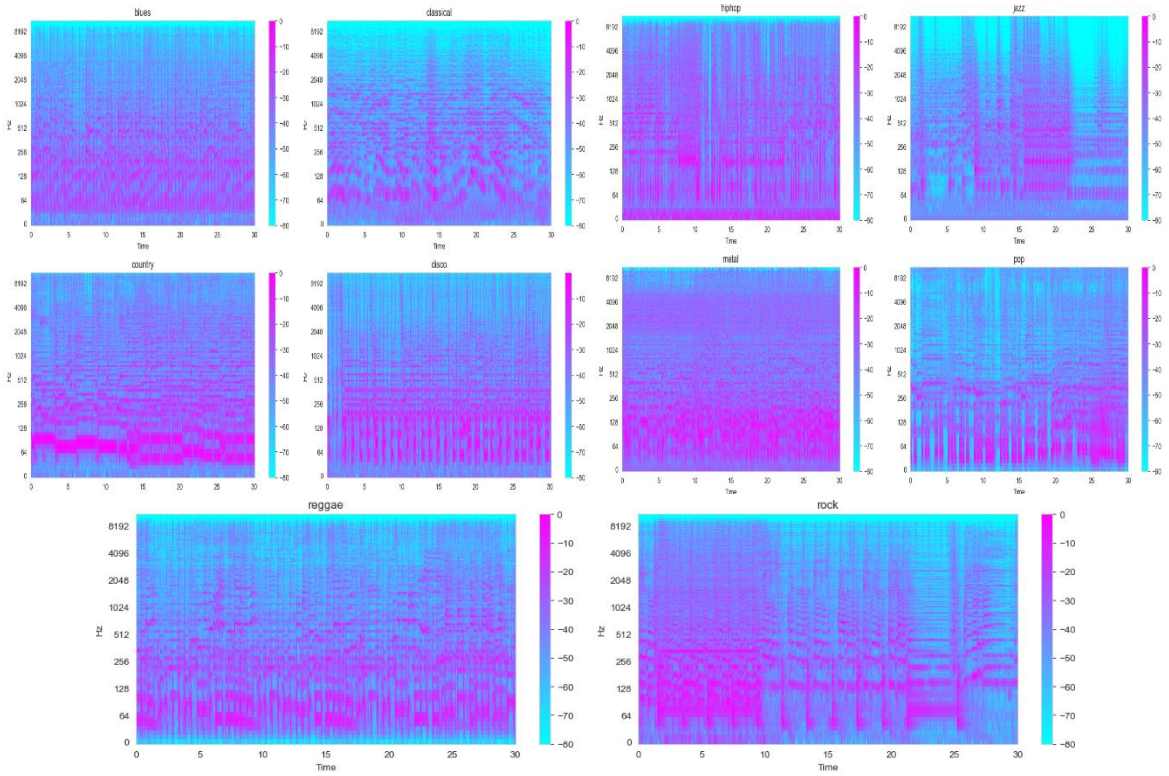


Figure 5.2

Figure 5.2 shows a spectrogram with the amplitudes converted to decibels (dB) using the logarithmic scale. The logarithmic scaling helps to compress the dynamic range and enhance the visibility of lower amplitudes. The colormap used in this plot is 'cool,' which assigns cooler colors (blues) to lower amplitudes and warmer colors (reds) to higher amplitudes. By visualizing the audio data as spectrograms, we can gain insights into the frequency content and temporal changes of the audio signal. Brighter regions indicate higher energy or amplitude in specific frequency bands.

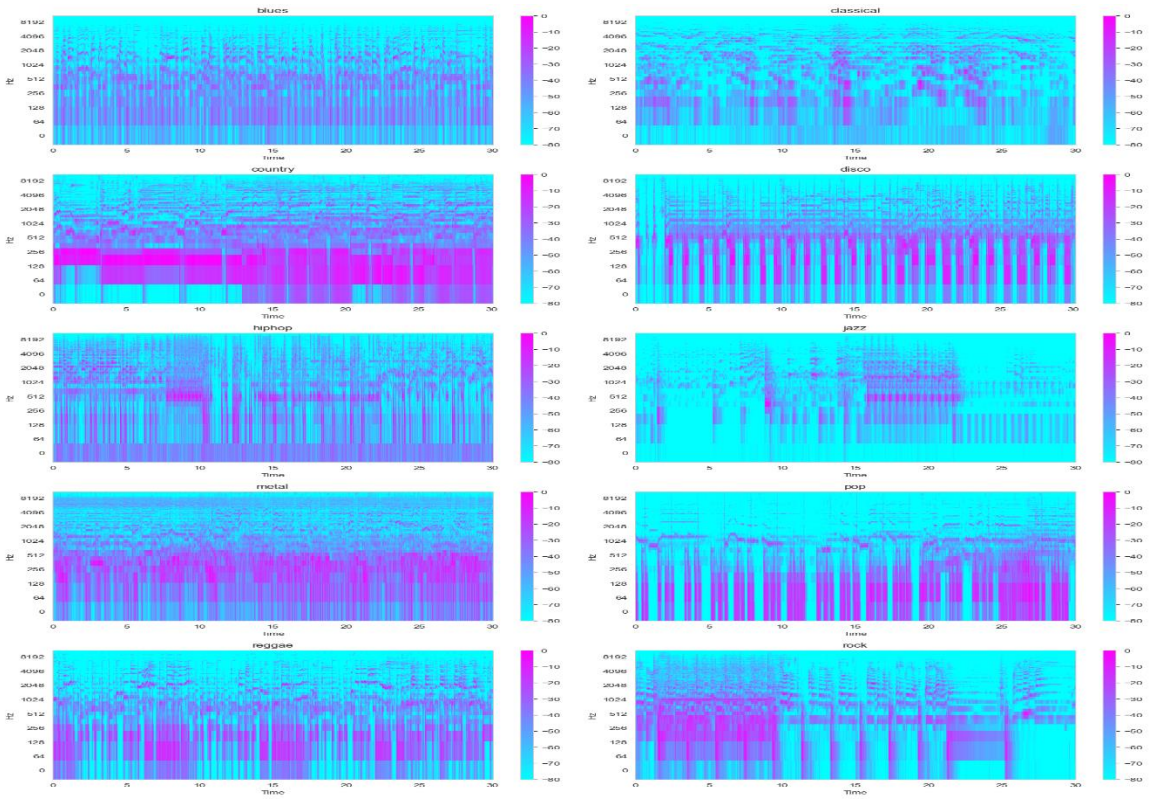


Figure 5.3

The plot represents the audio signal in its Mel spectrogram form. Mel spectrograms are a common way to analyze the frequency content of audio signals, particularly in the human auditory perception domain. The color intensity of each point in the plot represents the energy or amplitude of the audio signal at a particular frequency and time. Brighter colors indicate higher energy or amplitude, while darker colors indicate lower energy. By visualizing the Mel spectrogram, we can gain insights into the frequency content and its distribution over time in the audio signal. It provides a more detailed representation of the spectral characteristics of the audio compared to the waveform plot.

Spectral Centroids

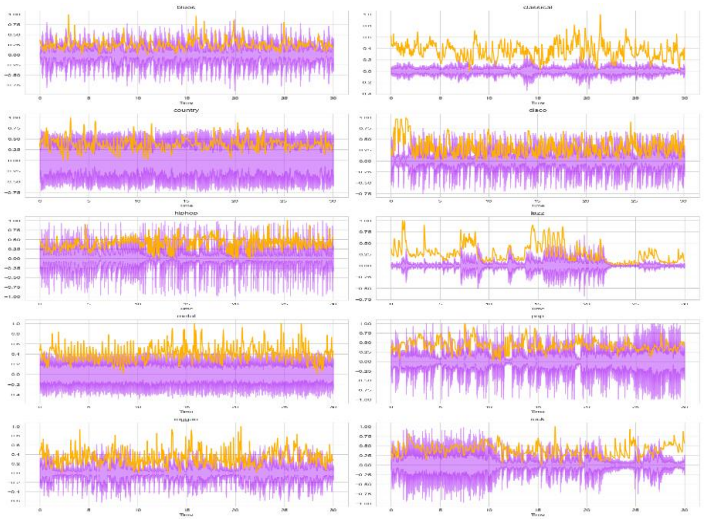


Figure 5.4

Spectral centroids point to the position of the center of mass of the spectrum. It has a close connection to how bright sounds are perceived audibly. The weighted mean of the frequencies present in the signal is calculated using a Fourier transform, with their magnitudes acting as the weights. Since it is a reliable prediction of the "brightness" of a sound, the spectral centroid is commonly used in digital audio and music processing as an automatic gauge of musical timbre (tone color or tone quality).

Tempo

Figure 5.5

Tempo is the speed at which the pattern repeats. Tempo is measured in Beats per minute. By visualizing the waveforms and beat positions, you can gain insights into the rhythmic characteristics of audio files from different music genres. It allows you to observe the rhythmic patterns and tempo variations within each genre.

Harmonic and Percussive Components

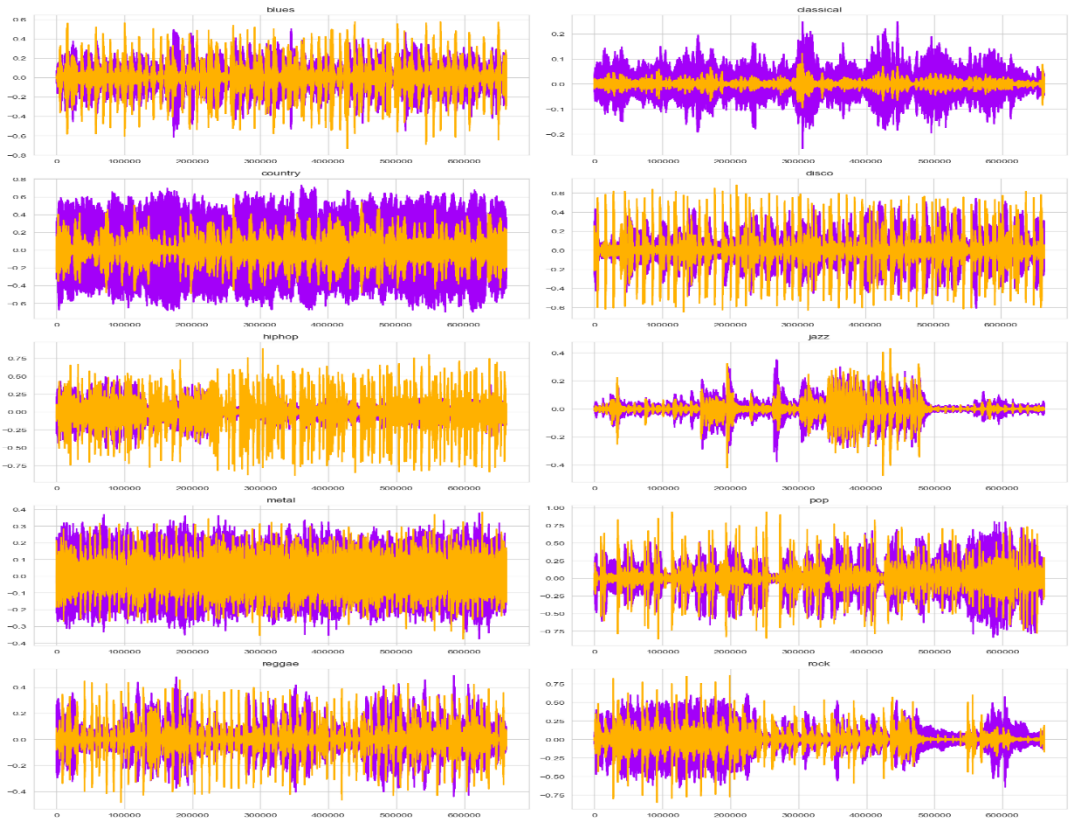


Figure 5.6

Musical sounds can comprise a wide range of sound components with different acoustic qualities. In particular, we consider two broad categories of sounds: harmonic sounds and percussive sounds. Loosely speaking, a harmonic sound is pitched sound, what makes us hear melodies and chords. The prototype of a harmonic sound is the acoustic realization of a sinusoid, which corresponds to a horizontal line in a spectrogram representation. On the other hand, a percussive sound is what we perceive as a clash, a knock, a clap, or a click. The prototype of a percussive sound is the acoustic realization of an impulse, which corresponds to a vertical line in a spectrogram representation

By visualizing the waveforms and beat positions, you can gain insights into the rhythmic characteristics of audio files from different music genres. It allows you to observe the rhythmic patterns and tempo variations within each genre.

Chromogram

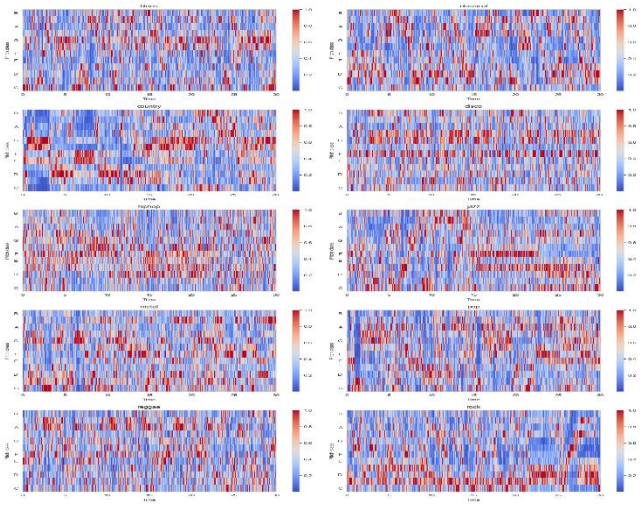


Figure 5.7

Figure 5.7 represents the audio signal in its Chroma feature form. Chroma features are used to capture the harmonic content of the audio signal, providing information about the pitch or musical notes. The y-axis represents the chroma feature, which is a 12-dimensional vector representing the 12 musical pitch classes (C, C#, D, D#, E, F, F#, G, G#, A, A#, B). The x-axis of the plot representing the temporal evolution of the audio signal. The Chroma feature plot provides insights into the harmonic content and tonal characteristics of the audio signal. It can help identify the presence of specific musical notes or chords and their variations over time.

BPM Boxplot for Genres

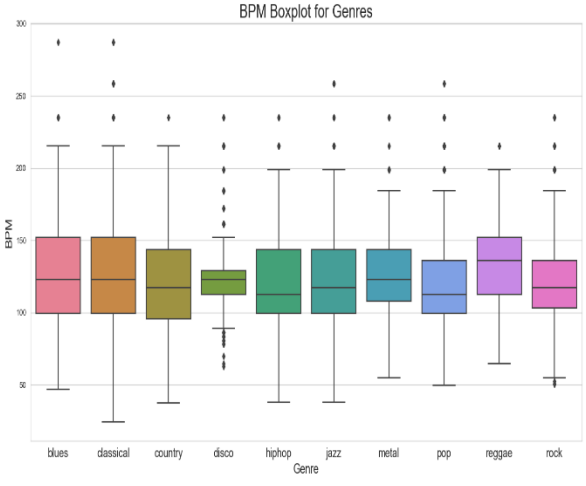


Figure 5.8

The plot consists of a box for each genre, representing the interquartile range (IQR) of the tempo values for that genre. The line inside the box represents the median tempo, and the whiskers extend to the minimum and maximum values within a certain range (typically 1.5 times the IQR). Classical, Blues, Hip pop and Jazz have larger spread of tempo values, while a disco, metal has more consistent tempo within the genre.

Principal Component Analysis

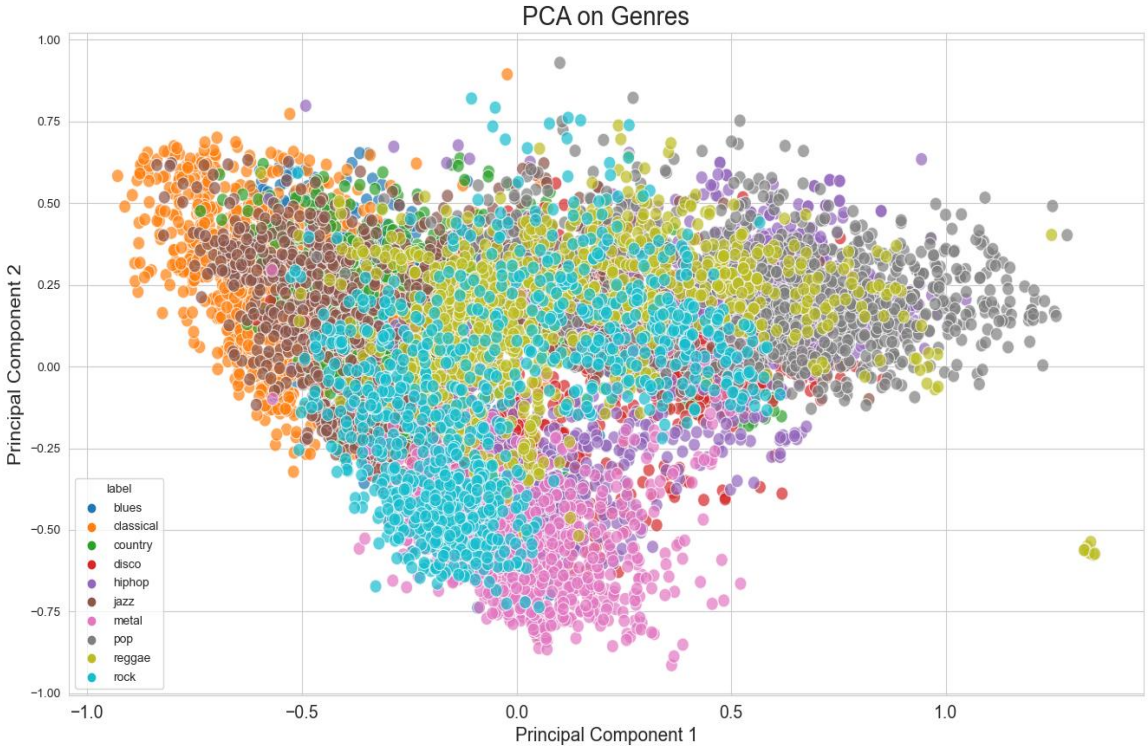


Figure 5.9

The plot represents the data points in a two-dimensional space based on the top two principal components obtained from PCA. Each point represents a sample (song) from the dataset. The plot can reveal how well the genres are separated or clustered in the reduced-dimensional space. Genres that form distinct clusters indicate that they have different audio feature distributions and can be more easily distinguished based on the analyzed features. By examining the plot, one can identify patterns, overlaps, or similarities between genres. Clusters of points belonging to the same genre suggest that songs within that genre share similar audio feature characteristics. The proximity of different genres on the scatter plot can provide insights into the relationships or similarities between genres. Genres that are closer together may share certain audio feature traits or exhibit similar patterns. Overall, the PCA scatter plot allows for a visual exploration of the similarities and differences between music genres based on the analyzed audio features.

Correlation

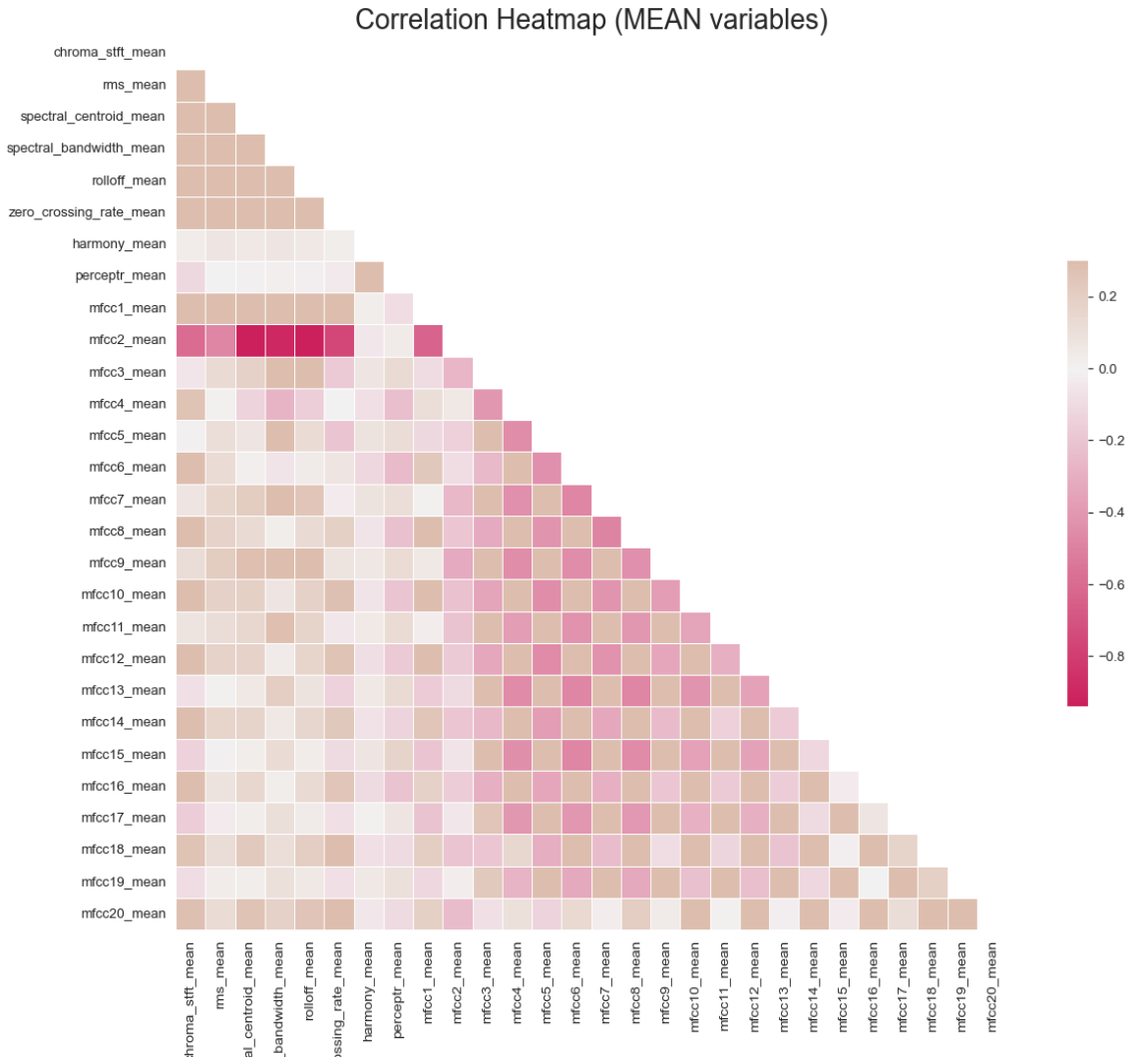


Figure 5.10

The correlation heatmap helps visualize the relationships between the mean variables in the DataFrame. It can provide insights into the patterns of association between different variables. Positive correlations indicate that the variables tend to increase or decrease together, while negative correlations indicate an inverse relationship. No correlation is represented by white cells.

5.3 Analytics Models and Algorithms

Machine Learning Models

	Accuracy	Algos
8	0.906771	CatBoostClassifier
5	0.897939	XGBClassifier
2	0.876349	KNeighborsClassifier
6	0.860648	RandomForestClassifie
4	0.741904	LogisticRegression
3	0.652601	DecisionTreeClassifier
0	0.506379	GaussianNB
7	0.494603	AdaBoostClassifier
1	0.487733	BernoulliNB

5.2 Table

Bernoulli Naive Bayes is a variant of Naive Bayes that assumes a binary distribution of features. The low accuracy suggests that the Bernoulli Naive Bayes model may not be appropriate for the given data or the features are not following a binary distribution. The AdaBoostClassifier had a success rate of only 49.4%. AdaBoost is an ensemble learning algorithm that combines multiple weak classifiers to create a strong classifier. The accuracy close to random guessing suggests that the AdaBoost model may not be effective in this particular classification task. Gaussian Naive Bayes has 0.506379 and it is a probabilistic algorithm that assumes Gaussian (normal) distribution of features. The accuracy close to random guessing suggests that the Gaussian Naive Bayes model may not be suitable for capturing the underlying patterns in the data. Decision Trees are a non-linear classification algorithm that makes decisions based on a tree-like model of decisions and their possible consequences. For this data set Decision Trees got 0.652601 accuracy and the moderate accuracy suggests that the Decision Tree model may be able to capture some patterns in the data but may not generalize well. Logistic Regression is a linear classification algorithm that models the probability of a certain class. The accuracy of 0.741904 indicates that the Logistic Regression model is able to capture some patterns in

the data but may not handle complex relationships well. The RandomForestClassifier had a success rate of 86%. Random Forest is an ensemble learning algorithm that combines multiple decision trees to make predictions. It is known for handling complex relationships and reducing overfitting. The accuracy score suggests that the Random Forest model is able to capture meaningful patterns in the data and make accurate predictions. K-Nearest Neighbors (KNN) is a simple yet effective classification algorithm that classifies new data points based on their proximity to existing data points. The accuracy of 0.876349 suggests that the KNN model is able to find similar instances in the training data for classification. XGBoost is a gradient boosting algorithm known for its speed and performance. The relatively high accuracy 0.897939 indicates that the XGBoost model is able to capture meaningful patterns in the data and make accurate predictions. CatBoost is a gradient boosting algorithm that can handle categorical features effectively. It often performs well and is known for its robustness. Success rate of 90.6% suggests that the CatBoost model is able to effectively learn patterns in the data and make accurate predictions.

In terms of model performance, CatBoostClassifier and XGBClassifier stand out with the highest accuracy scores. These models are likely more effective in capturing the patterns and relationships within the data.

Hyperparameter Tuning & Model Evaluation

By optimizing the hyperparameters, you were able to fine-tune the models and improve their accuracy. After the hyperparameter tuning, model evaluation of XGB got 87.34% accuracy and Random Forest got 85.87% accuracy. The accuracy improvement achieved after hyperparameter tuning indicates that the selected hyperparameters were able to fine-tune the model and capture more intricate patterns in the data. However, it's worth noting that even without hyperparameter tuning, CatBoost outperformed the other models, indicating its inherent strength in handling the given dataset.

CHAPTER 06 - ADVANCED ANALYSIS

Deep Learning Model

An ANN is a type of deep learning model that consists of interconnected layers of artificial neurons (nodes) that mimic the structure and function of neurons in a biological brain. The code snippet shows the construction of a sequential model, which is a straightforward type of ANN where layers are stacked sequentially. This model architecture includes dense (fully connected) layers, which are a fundamental component of ANNs. Each dense layer connects every neuron in the previous layer to every neuron in the current layer, enabling information flow and transformation through the network. Additionally, the model demonstrates the use of activation functions, such as ReLU (Rectified Linear Unit) for the hidden layers and softmax for the final output layer. Activation functions introduce non-linearity to the model, allowing it to learn complex patterns and make class predictions with Accuracy of 92.84%**.**

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 512)	29696
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32896
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 64)	8256
dropout_3 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 10)	650

Total params: 202,826
Trainable params: 202,826
Non-trainable params: 0

6.1 Table

CHAPTER 07 - DISCUSSION AND RECOMMENDATIONS

7.1 Discussion

Comparing the results, we can observe that the ANN model achieved the highest accuracy of 92.84%, outperforming all the other machine learning algorithms. This indicates that the ANN was able to learn complex patterns and relationships in the music data, resulting in better classification performance. Deep learning models have the ability to automatically learn complex patterns and features from data, eliminating the need for manual feature engineering. ANNs can handle large and high-dimensional datasets effectively than other models. ANNs can capture non-linear relationships in the data, making them suitable for complex tasks. But, ANNs are computationally intensive and require significant computational resources, especially for large-scale models.

In this classification, the ANN model outperformed the traditional machine learning algorithms, demonstrating its effectiveness for music classification tasks. The high accuracy achieved by the ANN suggests that it was able to capture the intricate patterns and features in the music data.

Overall, the achieved accuracy of 92.84% with the ANN model indicates its potential for music classification tasks, highlighting the benefits of utilizing deep learning techniques in this domain.

7.2 Recommendations

Even though my idea simply takes a rudimentary stab at the problem of categorizing musical genres, there are a number of ways it may be expanded. It is not fully fair to compare various learning methodologies for categorizing musical genres in this study. This gives a fair comparison of learning algorithms, but if we looked at the effectiveness of different factors (like combining with ID3 tag metadata and using audio data directly for model training), it would be helpful to know which machine learning stack performs the best at categorizing music. The exact same techniques employed in this study might easily be modified to classify music according to any other labeling, such as artist. Additionally, by including additional metadata text components like album, song, or lyrics, we may be able to expand this to music mood classification. Furthermore, applying deep learning techniques like CNN and RNN to both audio data and image data for better performance. Music classification plays a crucial role in music recommendation systems. By accurately classifying music into different genres or categories, recommendation algorithms can provide personalized music suggestions based on users' preferences. Therefore, it is possible to enhance this work by adding a music recommendation system.

7.3 Ethical Issues

Ethical questions are raised by the possibility of utilizing automated genre classification to outlaw or restrict specific musical genres. Numerous musical genres have served as platforms for the public propagation of geopolitical and cultural agendas throughout history. These messages might go against what the powerful want to hear. A genre classification system makes it straightforward to recognize music with objectionable messages, and depending on the goals of individuals in positions of authority, this knowledge can be used to either discourage or promote the popularity and exposure of that music. The democratic right to free expression is under jeopardy because of this.

7.4 Conclusions

In this paper, I provide a novel ensemble methodology for categorizing musical genres that separates the musical information into its spatial and temporal dimensions. The beginning, middle, and end of the music's various temporal parts are used to select feature vectors for the application of simple but effective classifiers. The GTZAN Dataset's deficiency in audio samples is the fundamental problem. Since our dataset is much lower than the actual number of songs existing, it is possible that different Deep Learning Models will perform better than the ones now in use if we add more genres and employ alternative data features. Overall, the achieved accuracy of 92.84% with the ANN model indicates its potential for music classification tasks, highlighting the benefits of utilizing deep learning techniques in this domain.

LIST OF REFERENCES

[1] Tzanetakis, G., & Cook, P. (2002). Musical genre classification of audio signals. IEEE Transactions on Speech and Audio Processing, 10(5), 293-302.

[2] Yang, Y. H., Lin, Y. A., Chen, H. H., & Su, M. C. (2008). Music genre classification using lyrics. Journal of the American Society for Information Science and Technology, 59(7), 1037-1049.93-302 (https://archives.ismir.net/ismir2017/paper/000043.pdf)

[3] Li, Y., Ogihara, M., & Li, Q. (2010). A comparative study on content-based music genre classification and categorization. ACM Transactions on Information Systems (TOIS), 28(1), 1-36.

[4] Toffetti, A., & Orio, N. (2011). Music genre classification via sparse representations of audio features. IEEE Transactions on Audio, Speech, and Language Processing, 19(4), 1082-1091.

[5] Li, T., Ogihara, M., & Li, Q. (2012). A survey of content-based music genre classification. Artificial Intelligence Review, 37(4), 239-266

[6] Pachet, F., & Cazaly, D. (2013). Improved music similarity measures for music recommendation. Journal of Intelligent Information Systems, 41(3), 371-392. (

[7] Han, J., & Chen, S. (2016). Music genre classification using deep learning techniques. In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 115-119). IEEE. (https://mdpi-res.com/d_attachment/applsci/applsci-13-05010/article_deploy/applsci-13-05010.pdf?version=1681712783)

[8] Dieleman, S., & Schrauwen, B. (2014). End-to-end learning for music audio. In Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR) (pp. 133-138).

[9] Yang, Y. H., & Chen, H. H. (2015). Music emotion classification: A fuzzy approach. Journal of the Association for Information Science and Technology, 66(12), 2513-2526.

[10] Koutamanis, A., & Marchini, M. (2018). Multi-modal music genre classification with convolutional neural networks. Journal of Intelligent Information Systems, 51(3), 363-383.

[11] M. Li; R. Sleep. Genre classification via an LZ78- based string kernel. Proceedings of the 6th International Conference on Music Information Retrieval, London, United Kingdom, pages 252–259, 2005.

[12] D. McEnnis; C. McKay; I. Fujinaga. Overview of OMEN (On-demand Metadata Extraction Network). Proceedings of the International Conference on Music Information Retrieval, Victoria, Canada, pages 7–12, 2006

[13] F. Vignoli. Digital music interaction concepts: a user study. Proceedings of the 5th International Conference on Music Information Retrieval, Barcelona, Spain, pages 415-420, 2004.

[14] Rudolf Mayer, Robert Neumayer, and Andreas Rauber. Rhyme and style features for musical genre classification by song lyrics. In Proceedings of the 9th International Conference on Music Information Retrieval, Philadelphia, PA, USA, September 14-18 2008.

APPENDICES

Table 5.1

```
# Calculate basic statistics for numerical features
statistics = df.describe()
statistics
```

Figure 5.1

```
fig, axes= plt.subplots(nrows=5, ncols=2, figsize=(15, 20))
i = 0
j = 0
for genre in genre_dict.keys():
    # Reading the first audio file
    data,sampling_rate = librosa.load(audio_fp + genre + "/" + genre + ".00000.wav")

    librosa.display.waveshow(y = data, sr = sampling_rate, color = "#A300F9",ax=axes[i][j])


    axes[i][j].set_title(genre)

    if(j == 1):
        i = i + 1
        j = 0
    else:
        j = j + 1

plt.tight_layout()
plt.show()
```

Figure 5.2

```
fig, axes= plt.subplots(nrows=5, ncols=2, figsize=(15, 20))
i = 0
j = 0
for genre in genre_dict.keys():
    # Reading the first audio file
    data,sampling_rate = librosa.load(audio_fp + genre + "/" + genre + ".00000.wav")

    stft_data = np.abs(librosa.stft(y = data, n_fft = n_fft,hop_length = hop_length))

    # Convert an amplitude spectrogram to Decibels-scaled spectrogram.
    DB = librosa.amplitude_to_db(stft_data, ref = np.max)

    img = librosa.display.specshow(DB, sr = sampling_rate, hop_length = hop_length, x_axis = 'time', y_axis = 'log',
                                   cmap = 'cool',ax=axes[i][j])
    fig.colorbar(img, ax=axes[i][j])

    axes[i][j].set_title(genre)

    if(j == 1):
        i = i + 1
        j = 0
    else:
        j = j + 1

plt.tight_layout()
plt.show()
```

Figure 5.3

```
fig, axes = plt.subplots(nrows=5, ncols=2, figsize=(15, 20))
i = 0
j = 0
for genre in genre_dict.keys():
    # Reading the first audio file
    audio_path = audio_fp + genre + "/" + genre + ".00000.wav"
    data, sampling_rate = librosa.load(audio_path)

    mel_spec = librosa.feature.melspectrogram(y=data, sr=sampling_rate, hop_length=hop_length)

    mel_spec_db = librosa.amplitude_to_db(mel_spec, ref=np.max)

    img = librosa.display.specshow(mel_spec_db, sr=sampling_rate, hop_length=hop_length, x_axis='time',
                                   y_axis='log', cmap='cool', ax=axes[i][j])
    fig.colorbar(img, ax=axes[i][j])

    axes[i][j].set_title(genre)

    if j == 1:
        i = i + 1
        j = 0
    else:
        j = j + 1

plt.tight_layout()
plt.show()
```

Figure 5.4

```
fig, axes= plt.subplots(nrows=5, ncols=2, figsize=(15, 20))
i = 0
j = 0
for genre in genre_dict.keys():
    # Reading the first audio file
    data,sampling_rate = librosa.load(audio_fp + genre + "/" + genre + ".00000.wav")

    spectral_centroids = librosa.feature.spectral_centroid(y=data, sr=sampling_rate)[0]

    # Computing the time variable for visualization
    frames = range(len(spectral_centroids))

    # Converts frame counts to time (seconds)
    t = librosa.frames_to_time(frames)

    librosa.display.waveshow(data, sr=sampling_rate, alpha=0.4, color = '#A300F9',ax=axes[i][j])

    axes[i][j].plot(t, sklearn.preprocessing.minmax_scale(spectral_centroids, axis=0), color='#FFB100')

    axes[i][j].set_title(genre)

    if(j == 1):
        i = i + 1
        j = 0
    else:
        j = j + 1

plt.tight_layout()
plt.show()
```

Figure 5.5

```
fig, axes= plt.subplots(nrows=5, ncols=2, figsize=(15, 20))
i = 0
j = 0
for genre in genre_dict.keys():
    # Reading the first audio file
    data,sampling_rate = librosa.load(audio_fp + genre + "/" + genre + ".00000.wav")

    tempo, beat_times = librosa.beat.beat_track(y=data, sr=sampling_rate, units='time')

    librosa.display.waveshow(y = data, sr = sampling_rate, color = "#A300F9",alpha = 0.4,ax=axes[i][j])

    axes[i][j].vlines(beat_times, np.min(data), np.max(data), color='r')

    axes[i][j].set_title(genre + "(Tempo - " + str(round(tempo,2)) + ")")

    if(j == 1):
        i = i + 1
        j = 0
    else:
        j = j + 1

plt.tight_layout()
plt.show()
```

Figure 5.6

```
fig, axes= plt.subplots(nrows=5, ncols=2, figsize=(15, 20))
i = 0
j = 0
for genre in genre_dict.keys():
    # Reading the first audio file
    data,sampling_rate = librosa.load(audio_fp + genre + "/" + genre + ".00000.wav")

    y_harm, y_perc = librosa.effects.hpss(data)

    axes[i][j].plot(y_harm, color = '#A300F9')

    axes[i][j].plot(y_perc, color = '#FFB100')

    axes[i][j].set_title(genre)

    if(j == 1):
        i = i + 1
        j = 0
    else:
        j = j + 1

plt.tight_layout()
plt.show()
```

Figure 5.7

```
fig, axes = plt.subplots(nrows=5, ncols=2, figsize=(15, 20))
i = 0
j = 0
for genre in genre_dict.keys():
    # Reading the first audio file
    audio_path = audio_fp + genre + "/" + genre + ".00000.wav"
    data, sampling_rate = librosa.load(audio_path)

    # Calculate STFT with the desired hop_length
    stft = librosa.stft(data, hop_length=hop_length)
    # Convert STFT to chromagram
    chromagram = librosa.feature.chroma_stft(S=stft, sr=sampling_rate)

    img = librosa.display.specshow(chromagram, x_axis='time', y_axis='chroma', hop_length=hop_length,
                                   cmap='coolwarm', ax=axes[i][j])
    fig.colorbar(img, ax=axes[i][j])

    axes[i][j].set_title(genre)

    if j == 1:
        i = i + 1
        j = 0
    else:
        j = j + 1

plt.tight_layout()
plt.show()
```

Figure 5.8

```
x = df[["label", "tempo"]]

fig, ax = plt.subplots(figsize=(16, 8));
sns.boxplot(x = "label", y = "tempo", data = x, palette = 'husl');

plt.title('BPM Boxplot for Genres', fontsize = 20)
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 10);
plt.xlabel("Genre", fontsize = 15)
plt.ylabel("BPM", fontsize = 15)
plt.savefig("BPM_Boxplot.png")
```

Figure 5.9

```
data = df.iloc[0:, 1:]
y = data['label']
X = data.loc[:, data.columns != 'label']

# normalize
cols = X.columns
min_max_scaler = skp.MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(X)
X = pd.DataFrame(np_scaled, columns = cols)

# Top 2 pca components
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
principalComponents = pca.fit_transform(X)
principalDf = pd.DataFrame(data = principalComponents, columns = ['pc1', 'pc2'])

# concatenate with target label
finalDf = pd.concat([principalDf, y], axis = 1)

plt.figure(figsize = (16, 9))
sns.scatterplot(x = "pc1", y = "pc2", data = finalDf, hue = "label", alpha = 0.7, s = 100);

plt.title('PCA on Genres', fontsize = 20)
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 10);
plt.xlabel("Principal Component 1", fontsize = 15)
plt.ylabel("Principal Component 2", fontsize = 15)
plt.savefig("PCA_Scattert.png")
```

Figure 5.10

```
# Computing the Correlation Matrix
spike_cols = [col for col in df.columns if 'mean' in col]
corr = df[spike_cols].corr()

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=np.bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(16, 11));

# Generate a custom diverging colormap
cmap = sns.diverging_palette(0, 25, as_cmap=True, s = 90, l = 45, n = 5)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})

plt.title('Correlation Heatmap (MEAN variables)', fontsize = 20)
plt.xticks(fontsize = 10)
plt.yticks(fontsize = 10);
plt.savefig("Corr_Heatmap.png")
```

Table 5.2 (ML Models)

```
# shuffle samples
df_shuffle = df.sample(frac=1, random_state=seed).reset_index(drop=True)
```

```
# remove irrelevant columns
df_shuffle.drop(['filename', 'length'], axis=1, inplace=True)
df_y = df_shuffle.pop('label')
df_X = df_shuffle

# split into train dev and test
X_train, df_test_valid_X, y_train, df_test_valid_y = skms.train_test_split(df_X, df_y, train_size=0.7,
                                                                           random_state=seed, stratify=df_y)
X_dev, X_test, y_dev, y_test = skms.train_test_split(df_test_valid_X, df_test_valid_y, train_size=0.66,
                                                     random_state=seed, stratify=df_test_valid_y)
```

```
scaler = skp.StandardScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train), columns=X_train.columns)
X_dev = pd.DataFrame(scaler.transform(X_dev), columns=X_train.columns)
X_test = pd.DataFrame(scaler.transform(X_test), columns=X_train.columns)
```

```
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
import catboost as cb

from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score


# create an instance of each classification algorithm
g=GaussianNB()
b=BernoulliNB()
KN=KNeighborsClassifier()
D=DecisionTreeClassifier()
LR = LogisticRegression()
XGB= XGBClassifier()
abc = ske.AdaBoostClassifier()
rfc = ske.RandomForestClassifier()
cbc = cb.CatBoostClassifier(random_state=seed, verbose=0, eval_metric='Accuracy', loss_function='MultiClass')

algos=[g,b,KN,D,LR,XGB,rfc,abc,cbc]
algo_names=['GaussianNB','BernoulliNB','KNeighborsClassifier','DecisionTreeClassifier','LogisticRegression',
            'XGBClassifier','RandomForestClassifie','AdaBoostClassifier','CatBoostClassifier']

accuracy_scored=[]

# fit and predict for each Algo
for item in algos:
    item.fit(X_train,y_train)
    item.predict(X_test)
    accuracy_scored.append(accuracy_score(y_test,item.predict(X_test)))
```

```
# show results in a DataFrame
result = pd.DataFrame(accuracy_scored, columns=["Accuracy"])
result['Algos']=algo_names
result.sort_values('Accuracy',ascending=False)
```

Hyperparameter Tuning & Model Evaluation (Page 33)

```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 10, stop = 400, num = 4)]
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(1, 60, num = 5)]
# Minimum number of samples required to split a node
min_samples_split = [5, 10, 15]

# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split}

pprint(random_grid)
```

```
{'max_depth': [1, 15, 30, 45, 60],
 'min_samples_split': [5, 10, 15],
 'n_estimators': [10, 140, 270, 400]}
```

```
# Use grid search to find best hyperparameters
rfc_n = ske.RandomForestClassifier(random_state=seed, n_jobs=-1)
rf_random = skms.GridSearchCV(estimator = rfc_n, param_grid=random_grid, cv = 5, verbose=2, n_jobs = -1, scoring='f1_weighted')
# Fit the random search model
rf_random.fit(X_train, y_train)
```

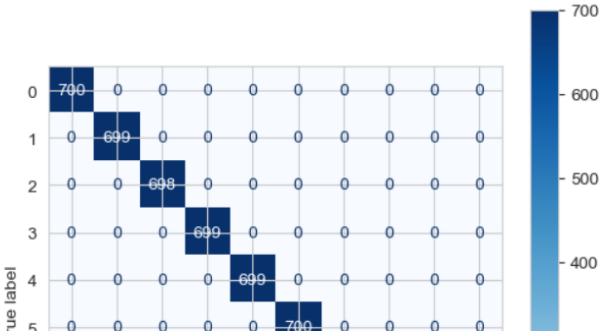
```
Fitting 5 folds for each of 60 candidates, totalling 300 fits

GridSearchCV(cv=5, estimator=RandomForestClassifier(n_jobs=-1, random_state=12),
             n_jobs=-1,
             param_grid={'max_depth': [1, 15, 30, 45, 60],
                         'min_samples_split': [5, 10, 15],
                         'n_estimators': [10, 140, 270, 400]},
             scoring='f1_weighted', verbose=2)
```

```
# best random model
print(rf_random.best_estimator_)
```

```
RandomForestClassifier(max_depth=45, min_samples_split=5, n_estimators=400,
                       n_jobs=-1, random_state=12)
```

```
def predictProba(clf, X, dev=False):
    y_true = y_train if dev else y_train
    y_pred_proba_X = clf.predict_proba(X)
    y_pred_X = clf.predict(X)
    unique_labels = np.unique(np.concatenate((np.expand_dims(y_true, axis=1), np.expand_dims(y_pred_X, axis=1)), axis=1))
    fig, ax = plt.subplots(figsize=(6, 6))
    skm.plot_confusion_matrix(clf, X, y_true, display_labels=unique_labels, cmap=plt.cm.Blues, xticks_rotation=90, ax=ax)
    plt.show()
```

```
# Performance metrics
predictProba(rf_random.best_estimator_, X_train)
```




```
xgb_model = XGBClassifier(n_estimators=100, random_state=seed)

xgb_params = {
    "colsample_bytree": uniform(0.7, 0.3),
    "gamma": uniform(0, 0.5),
    "learning_rate": uniform(0.03, 0.3), # default 0.1
    "max_depth": randint(2, 6), # default 3
    "n_estimators": randint(100, 150), # default 100
    "subsample": uniform(0.6, 0.4)
}
```

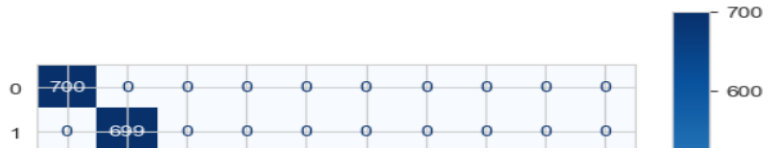
```
# Use the random grid to search for best hyperparameters
xgb_random = skms.RandomizedSearchCV(estimator = xgb_model, param_distributions=xgb_params, n_iter=20, cv = 3,
                                     verbose=2, n_jobs = -1, random_state=seed, scoring='f1_weighted', return_train_score=True)
# Fit the random search model
xgb_random.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits

RandomizedSearchCV(cv=3,
                   estimator=XGBClassifier(base_score=None, booster=None,
                                           callbacks=None,
                                           colsample_bylevel=None,
                                           colsample_bynode=None,
                                           colsample_bytree=None,
                                           early_stopping_rounds=None,
                                           enable_categorical=False,
                                           eval_metric=None, feature_types=None,
                                           gamma=None, gpu_id=None,
                                           grow_policy=None,
                                           importance_type=None,
                                           interaction_constraints=None,
                                           learning_rate...
                                         'max_depth': <scipy.stats._distn_infrastructure.rv_discrete_frozen object at 0x00000152
07376520>,
```

```
# best xgb_random model
print(xgb_random.best_estimator_)
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.8809980488377586, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=0.036002311696922085, gpu_id=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=0.13856085510393606, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=5, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=124, n_jobs=None, num_parallel_tree=None,
              objective='multi:softprob', predictor=None, ...)
```

```
# Performance metrics
predictProba(xgb_random.best_estimator_, X_train)
```



Model Evaluation

```
y_pred_X = rf_random.best_estimator_.predict(X_test[X_train.columns])
print(skm.classification_report(y_test, y_pred_X, digits=3))
print("RF Test Accuracy -",skm.accuracy_score(y_test, y_pred_X)*100)
```

```
              precision    recall  f1-score   support

           0      0.874     0.882     0.878       102
           1      0.899     0.970     0.933       101
           2      0.810     0.794     0.802       102
           3      0.892     0.814     0.851       102
           4      0.897     0.853     0.874       102
           5      0.895     0.922     0.908       102
           6      0.900     0.971     0.934       102
           7      0.877     0.912     0.894       102
           8      0.876     0.902     0.889       102
           9      0.813     0.725     0.767       102

    accuracy                          0.874      1019
   macro avg      0.873     0.874     0.873      1019
weighted avg      0.873     0.874     0.873      1019

RF Test Accuracy - 87.43866535819431
```

```
y_pred_X = xgb_random.best_estimator_.predict(X_test[X_train.columns])
print(skm.classification_report(y_test, y_pred_X, digits=3))
print("XGB Test Accuracy -",skm.accuracy_score(y_test, y_pred_X)*100)
```

```
              precision    recall  f1-score   support

           0      0.861     0.912     0.886       102
           1      0.951     0.960     0.956       101
           2      0.788     0.804     0.796       102
           3      0.906     0.853     0.879       102
           4      0.936     0.863     0.898       102
           5      0.898     0.951     0.924       102
           6      0.950     0.931     0.941       102
           7      0.913     0.922     0.917       102
           8      0.900     0.882     0.891       102
           9      0.779     0.794     0.786       102

    accuracy                          0.887      1019
   macro avg      0.888     0.887     0.887      1019
weighted avg      0.888     0.887     0.887      1019

XGB Test Accuracy - 88.7144259077527
```

```
y_pred_X = cbc.predict(X_test[X_train.columns])
print(skm.classification_report(y_test, y_pred_X, digits=3))
print("CatBoost Test Accuracy -",skm.accuracy_score(y_test, y_pred_X)*100)
```

```
              precision    recall  f1-score   support

           0      0.894     0.912     0.903       102
           1      0.934     0.980     0.957       101
           2      0.866     0.824     0.844       102
           3      0.899     0.873     0.886       102
           4      0.947     0.882     0.914       102
           5      0.894     0.912     0.903       102
           6      0.961     0.971     0.966       102
           7      0.931     0.931     0.931       102
           8      0.913     0.922     0.917       102
           9      0.830     0.863     0.846       102

    accuracy                          0.907      1019
   macro avg      0.907     0.907     0.907      1019
weighted avg      0.907     0.907     0.907      1019

CatBoost Test Accuracy - 90.67713444553483
```

Table 6.1

```
# import deep learning libraries
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
```

```
# build model
model = keras.models.Sequential([
    keras.layers.Dense(512, activation="relu", input_shape=(X_train.shape[1],)),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(256,activation="relu"),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(128,activation="relu"),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(64,activation="relu"),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(10, activation="softmax"),

])
```

```
# compile model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['acc'])
```

```
# fit model - training
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=300, batch_size=128)
```

```
55/55 [==============================] - 0s 6ms/step - loss: 0.0139 - acc: 0.9947 - val_loss: 0.4066 - val_acc: 0.9362
Epoch 292/300
55/55 [==============================] - 0s 6ms/step - loss: 0.0193 - acc: 0.9936 - val_loss: 0.4358 - val_acc: 0.9254
Epoch 293/300
55/55 [==============================] - 0s 6ms/step - loss: 0.0173 - acc: 0.9941 - val_loss: 0.4145 - val_acc: 0.9303
Epoch 294/300
55/55 [==============================] - 0s 6ms/step - loss: 0.0137 - acc: 0.9953 - val_loss: 0.3705 - val_acc: 0.9323
Epoch 295/300
55/55 [==============================] - 0s 7ms/step - loss: 0.0187 - acc: 0.9953 - val_loss: 0.3687 - val_acc: 0.9401
Epoch 296/300
55/55 [==============================] - 0s 7ms/step - loss: 0.0224 - acc: 0.9940 - val_loss: 0.4097 - val_acc: 0.9235
Epoch 297/300
55/55 [==============================] - 0s 6ms/step - loss: 0.0153 - acc: 0.9947 - val_loss: 0.3981 - val_acc: 0.9313
Epoch 298/300
55/55 [==============================] - 0s 6ms/step - loss: 0.0175 - acc: 0.9937 - val_loss: 0.3991 - val_acc: 0.9284
Epoch 299/300
55/55 [==============================] - 0s 6ms/step - loss: 0.0198 - acc: 0.9939 - val_loss: 0.3602 - val_acc: 0.9323
Epoch 300/300
55/55 [==============================] - 0s 6ms/step - loss: 0.0163 - acc: 0.9946 - val_loss: 0.4399 - val_acc: 0.9303
```

```
# evaluate model
_, accuracy = model.evaluate(X_test, y_test, batch_size=128)
```

```
8/8 [==============================] - 0s 3ms/step - loss: 0.4399 - acc: 0.9303
```

```
print("Accuracy:",accuracy) # print accuracy
```

```
Accuracy: 0.9303238391876221
```