**Chapter 1**

**Introduction**

Music is an integral part of human culture and entertainment, spanning diverse genres and styles that cater to a wide range of preferences and tastes. With the advent of digital music platforms and streaming services, there is a growing need for efficient methods to categorize and organize music content. Music genre classification, the task of automatically assigning a genre label to a piece of music based on its audio features, plays a crucial role in music information retrieval and recommendation systems.In this project, we tackle the challenge of music genre classification using machine learning techniques. The objective is to develop a robust and accurate classification model capable of automatically identifying the genre of a given audio sample. To achieve this goal, we leverage a publicly available dataset from Kaggle, comprising audio samples across ten distinct genres: blues, classical, country, disco, hip hop, jazz, metal, rock, and reggae.The dataset consists of audio recordings in digital format, with each sample labeled with its corresponding genre. These audio files serve as the primary source of input for our classification task. However, raw audio data is inherently high-dimensional and complex, making it challenging to extract meaningful features directly. Therefore, we employ signal processing techniques to transform the audio signals into a more manageable feature space.Key features extracted from the audio samples include Mel-Frequency Cepstral Coefficients (MFCCs), spectral characteristics, rhythmic patterns, and other relevant descriptors. These features capture essential aspects of the audio signal, such as timbre, pitch, and temporal dynamics, which are informative for distinguishing between different music genres.In the classification phase, we explore a range of machine learning algorithms, each with its unique strengths and characteristics. These algorithms include k-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Tree, Random Forest, Naive Bayes, and Logistic Regression. By training and evaluating these classifiers on the extracted audio features, we aim to identify the most suitable approach for music genre classification.The performance of each classifier is assessed in terms of classification accuracy, precision, recall, and F1-score. Through rigorous experimentation and cross-validation techniques, we analyze the strengths and limitations of each algorithm, providing insights into their effectiveness for music genre classification tasks.

* 1. **Problem Statement:**

Music genre classification is a challenging task due to the inherent complexity and diversity of musical styles and compositions. Traditional methods of manual genre labeling are time-consuming and subjective, making them impractical for large music databases. Additionally, existing automatic genre classification systems often suffer from limited accuracy and scalability, hindering their widespread adoption in real-world applications.

* 1. **Aim:**

The aim of this project is to develop an accurate and scalable music genre classification system using machine learning techniques. By leveraging advanced signal processing methods and state-of-the-art classification algorithms, we seek to create a robust system capable of automatically categorizing music tracks into predefined genres with high precision and efficiency.

* 1. **Existing System:**

Existing music genre classification systems typically rely on either handcrafted features or deep learning architectures for feature extraction and classification. Handcrafted feature-based approaches often suffer from limited discriminative power and generalization ability, while deep learning models require large amounts of labeled data and computational resources for training. Moreover, many existing systems lack user-friendly interfaces and are not easily accessible to non-technical users.

* 1. **Proposed System:**

Our proposed system integrates signal processing techniques, machine learning algorithms, and web development frameworks to create a comprehensive music genre classification solution. We employ advanced feature extraction methods, including Mel-Frequency Cepstral Coefficients (MFCCs), spectral features, and rhythmic patterns, to capture relevant characteristics of audio signals. These features are then fed into a variety of machine learning classifiers, such as Support Vector Machines (SVM), Decision Trees, Random Forests, Naive Bayes, and Logistic Regression, to perform genre classification.

To enhance the accessibility and usability of the system, we develop a web-based interface using the Django framework. This interface allows users to upload audio files directly to the system, which are then processed by the trained classification model to predict the corresponding music genre. The web application provides a seamless user experience, enabling both technical and non-technical users to interact with the classification system effortlessly.

* 1. **Objectives:**
* Collect and preprocess a diverse dataset of audio samples spanning multiple music genres.
* Extract relevant features from the audio data using signal processing techniques.
* Implement and evaluate multiple machine learning algorithms for music genre classification.
* Develop a web-based interface for uploading audio files and displaying classification results.
* Train and deploy the classification model on the web server using Django framework.
* Evaluate the performance of the system in terms of accuracy, efficiency, and user satisfaction.

Advantages:

* Accuracy: By leveraging machine learning algorithms and advanced feature extraction techniques, our system achieves high accuracy in music genre classification, outperforming existing approaches.
* Scalability: The proposed system is scalable and can accommodate large music databases, making it suitable for real-world applications with extensive collections of audio tracks.
* User-Friendly Interface: The web-based interface provides an intuitive and accessible platform for users to interact with the classification system, regardless of their technical expertise.
* Efficiency: The system is designed to process audio files quickly and efficiently, enabling rapid classification of music tracks with minimal latency.

**Chapter 2**

**Methodology**

**Data Collection:**

* Obtain a diverse and representative dataset of audio samples spanning multiple music genres. Utilize publicly available datasets, such as the GTZAN dataset from Kaggle, which contains audio clips across ten genres.
* Ensure that the dataset is well-annotated, with each audio sample labeled with its corresponding genre.

Data Preprocessing:

* Convert audio files to a common format and standardize their sampling rates to ensure consistency across the dataset.
* Segment audio files into short, non-overlapping frames to facilitate feature extraction.
* Apply preprocessing techniques such as normalization and noise reduction to enhance the quality of the audio data.

Feature Extraction:

* Extract a diverse set of features from the preprocessed audio samples to capture relevant characteristics of the music.
* Utilize signal processing techniques, including:
* Mel-Frequency Cepstral Coefficients (MFCCs) to represent spectral features.
* Spectral centroid, bandwidth, and contrast to capture spectral shape and texture.
* Temporal features such as zero-crossing rate and rhythmic patterns.
* Concatenate or aggregate the extracted features to form a feature vector representation for each audio sample.

Model Selection:

* Experiment with a variety of machine learning algorithms to identify the most suitable classifier for music genre classification.
* Evaluate algorithms such as k-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Trees, Random Forests, Naive Bayes, and Logistic Regression.
* Tune hyperparameters using techniques like cross-validation to optimize the performance of each classifier.

Model Training and Evaluation:

* Split the dataset into training and testing sets using stratified sampling to ensure balanced representation of genres in both sets.
* Train each classifier using the training data and evaluate its performance on the testing set.
* Assess performance metrics such as accuracy, precision, recall, and F1-score to measure the effectiveness of each classifier.
* Perform comparative analysis to identify the top-performing algorithm(s) for music genre classification.

Web Application Development:

Develop a web-based interface using the Django framework to facilitate interaction with the classification system.

Design a user-friendly interface that allows users to upload audio files and receive genre predictions in real-time.

Integrate the trained classification model into the web application backend to perform genre classification on user-uploaded audio files.

**Chapter 3**

**Literature Survey**

[1] A. Ghildiyal, K. Singh and S. Sharma, "Music Genre Classification using Machine Learning," 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2020:

This paper presents a comprehensive approach to music genre classification using machine learning algorithms. Ghildiyal et al. employ a dataset comprising audio samples across multiple genres and extract relevant features such as Mel-Frequency Cepstral Coefficients (MFCCs), spectral characteristics, and rhythmic patterns. They experiment with various classifiers including k-Nearest Neighbors (KNN), Support Vector Machines (SVM), Decision Trees, and Random Forests, evaluating their performance in terms of accuracy and computational efficiency. The study demonstrates the effectiveness of machine learning techniques in accurately classifying music genres and provides insights into the strengths and limitations of different classification algorithms.

[2] A. Elbir, H. Bilal Çam, M. Emre Iyican, B. Öztürk and N. Aydin, "Music Genre Classification and Recommendation by Using Machine Learning Techniques," 2018 Innovations in Intelligent Systems and Applications Conference (ASYU), Adana, Turkey, 2018:

Elbir et al. propose a novel approach to music genre classification and recommendation leveraging machine learning techniques. The study explores the integration of classification algorithms with recommendation systems to provide personalized music recommendations based on user preferences. By analyzing user listening behavior and feedback, the system dynamically adapts its recommendations to match individual tastes and preferences. The paper demonstrates the feasibility and effectiveness of combining classification and recommendation techniques to enhance the user experience in music streaming platforms and digital libraries.

[3] I. Pathania and N. Kaur, "Classification of Music Genre Using Machine Learning," 2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2022:

Pathania and Kaur investigate the application of machine learning algorithms for music genre classification in their study. They propose a feature extraction pipeline that incorporates both low-level and high-level features to capture various aspects of audio signals. The study evaluates the performance of classifiers such as Support Vector Machines (SVM), Random Forests, and Convolutional Neural Networks (CNNs) on a diverse dataset of music genres. Through extensive experimentation and comparative analysis, the authors highlight the effectiveness of different feature representations and classification algorithms in accurately categorizing music genres.

Conclusion:

These papers provide valuable insights into the methodologies, algorithms, and techniques employed in music genre classification using machine learning. By leveraging advanced feature extraction methods, diverse datasets, and state-of-the-art classification algorithms, researchers aim to develop robust and efficient systems capable of accurately categorizing music genres and providing personalized recommendations to users. Further research in this area is essential to address challenges such as scalability, computational efficiency, and user-centric recommendation strategies, ultimately advancing the field of music information retrieval and enhancing user experiences in digital music platforms.

[4] V. Shah, A. Tandle, N. Sharma and V. Sheth, "Genre Based Music Classification using Machine Learning and Convolutional Neural Networks," 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2021:

Shah et al. propose a hybrid approach combining machine learning techniques with Convolutional Neural Networks (CNNs) for genre-based music classification. The study explores feature extraction methods such as MFCCs, spectral features, and rhythmic patterns, which are then fed into both traditional classifiers and CNN models. By leveraging the hierarchical and spatial learning capabilities of CNNs, the proposed approach aims to capture both local and global patterns in music audio signals for improved classification accuracy. The paper presents experimental results demonstrating the effectiveness of the hybrid approach in accurately classifying music genres and highlights the benefits of combining machine learning and deep learning techniques.

[5] V. Prashanthi, S. Kanakala, V. Akila and A. Harshavardhan, "Music Genre Categorization using Machine learning Algorithms," 2021 International Conference on Computational Intelligence and Computing Applications (ICCICA), Nagpur, India, 2021:

Prashanthi et al. investigate the application of machine learning algorithms for music genre categorization. The study explores various feature extraction techniques, including MFCCs, spectral features, and rhythmic patterns, to represent audio signals. Using a diverse dataset encompassing multiple music genres, the authors evaluate the performance of classifiers such as Support Vector Machines (SVM), k-Nearest Neighbors (KNN), and Decision Trees. The paper presents comparative analysis results, highlighting the strengths and weaknesses of different classification algorithms in accurately categorizing music genres based on audio features.

[6] K. S. Mounika, S. Deyaradevi, K. Swetha and V. Vanitha, "Music Genre Classification Using Deep Learning," 2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA), Coimbatore, India, 2021:

Mounika et al. propose a deep learning-based approach for music genre classification, specifically focusing on Convolutional Neural Networks (CNNs). The study explores the use of deep learning architectures to automatically learn hierarchical representations of audio features directly from raw audio data. By training CNN models on spectrogram representations of audio samples, the authors aim to capture both low-level and high-level features for genre classification. Experimental results demonstrate the effectiveness of CNNs in learning discriminative features for music genre classification, showcasing the potential of deep learning techniques in this domain.

The field of music genre classification has seen significant advancements with the emergence of deep learning techniques, which have shown promise in capturing complex patterns and representations inherent in audio signals. In this literature survey, we delve into two noteworthy papers that contribute to the domain of music genre classification:

[7] M. Shah, N. Pujara, K. Mangaroliya, L. Gohil, T. Vyas and S. Degadwala, "Music Genre Classification using Deep Learning," 2022 6th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2022:

Shah et al. present a study on music genre classification employing deep learning methodologies. The paper explores the use of deep neural networks, specifically Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), to automatically learn hierarchical representations of audio features directly from raw audio data. The authors experiment with different network architectures and training strategies to optimize classification performance. By training deep learning models on large-scale datasets of audio samples spanning multiple genres, the study demonstrates the effectiveness of CNNs and RNNs in capturing both local and global patterns in music audio signals for accurate genre classification. The paper contributes insights into the application of deep learning techniques for music genre classification and provides a foundation for further research in this area.

[8] P. Devaki, A. Sivanandan, R. S. Kumar and M. Z. Peer, "Music Genre Classification and Isolation," 2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA), Coimbatore, India, 2021:

Devaki et al. present a novel approach to music genre classification and isolation, focusing on the identification and extraction of specific genre segments within audio tracks. The paper introduces a multi-stage framework that combines feature extraction, clustering, and classification techniques to isolate genre-specific segments from audio recordings. By leveraging unsupervised learning algorithms such as k-means clustering and Gaussian Mixture Models (GMMs), the proposed framework identifies genre boundaries and separates audio tracks into distinct genre segments. The authors evaluate the framework on a diverse dataset of music recordings, demonstrating its effectiveness in accurately categorizing and isolating genre-specific segments. The study contributes to the development of music genre classification systems capable of fine-grained analysis and segmentation of audio content, enabling more precise genre identification and content-based retrieval.

[9] Hareesh Bahuleya, "Music Genre Classification using Machine Learning Techniques":

Bahuleya presents a comprehensive study on music genre classification leveraging machine learning techniques. The paper explores various feature extraction methods, including Mel-Frequency Cepstral Coefficients (MFCCs), spectral features, and rhythm patterns, to represent audio signals effectively. The author experiments with a range of

machine learning algorithms such as Support Vector Machines (SVM), k-Nearest Neighbors (KNN), Decision Trees, and Random Forests, evaluating their performance in terms of classification accuracy and computational efficiency. By training and testing on diverse datasets comprising multiple music genres, the study demonstrates the efficacy of machine learning techniques in accurately categorizing music genres and provides insights into the strengths and limitations of different classification algorithms.

[10] N. Ndou, R. Ajoodha and A. Jadhav, "Music Genre Classification: A Review of Deep-Learning and Traditional Machine-Learning Approaches," 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), Toronto, ON, Canada, 2021:

Ndou et al. present a comprehensive review of music genre classification methodologies, focusing on both deep learning and traditional machine learning approaches. The paper provides an overview of feature extraction techniques, including MFCCs, chroma features, and spectral descriptors, utilized in music genre classification. The authors discuss the application of deep learning architectures such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks, highlighting their ability to automatically learn hierarchical representations of audio features. Additionally, the review examines traditional machine learning algorithms such as SVM, KNN, and Decision Trees, discussing their effectiveness and limitations in music genre classification tasks. Through a comparative analysis of deep learning and traditional machine learning approaches, the paper offers insights into the state-of-the-art techniques and challenges in music genre classification research.

Here's the literature survey presented in a tabular format:

| **Paper Title** | **Authors** | **Objectives** | **Methodology** | **Conclusion** |
| --- | --- | --- | --- | --- |
| Music Genre Classification using Machine Learning | A. Ghildiyal, K. Singh, S. Sharma | To explore machine learning algorithms for music genre classification. | - Extract features like MFCCs, spectral characteristics, and rhythmic patterns. <br> - Employ classifiers such as KNN, SVM, Decision Trees, and Random Forests. | The study demonstrates the effectiveness of machine learning techniques in accurately classifying music genres. |
| Music Genre Classification and Recommendation by Using Machine Learning Techniques | A. Elbir, H. Bilal Çam, M. Emre Iyican, B. Öztürk, N. Aydin | To investigate music genre classification and recommendation systems. | - Utilize classification algorithms and recommendation techniques. | The study showcases the integration of classification and recommendation systems for personalized music recommendations. |
| Classification of Music Genre Using Machine Learning | I. Pathania, N. Kaur | To classify music genres using machine learning methods. | - Experiment with feature extraction techniques and classifiers. | The study highlights the effectiveness of machine learning algorithms in categorizing music genres based on audio features. |
| Genre Based Music Classification using Machine Learning and Convolutional Neural Networks | V. Shah, A. Tandle, N. Sharma, V. Sheth | To explore music genre classification using machine learning and CNNs. | - Combine machine learning techniques with CNNs for classification. | The paper demonstrates the efficacy of CNNs in capturing patterns in music audio signals for genre classification. |
| Music Genre Categorization using Machine learning Algorithms | V. Prashanthi, S. Kanakala, V. Akila, A. Harshavardhan | To categorize music genres using machine learning algorithms. | - Experiment with various feature extraction methods and classifiers. | The study provides insights into the strengths and weaknesses of different classification algorithms in accurately categorizing music genres. |
| Music Genre Classification Using Deep Learning | K. S. Mounika, S. Deyaradevi, K. Swetha, V. Vanitha | To investigate deep learning-based approaches for music genre classification. | - Utilize deep learning architectures such as CNNs. | The paper demonstrates the effectiveness of CNNs in learning hierarchical representations of audio features for genre classification. |
| Music Genre Classification using Deep Learning | M. Shah, N. Pujara, K. Mangaroliya, L. Gohil, T. Vyas, S. Degadwala | To explore deep learning methodologies for music genre classification. | - Train deep neural networks like CNNs and RNNs on audio features. | The study showcases the efficacy of CNNs and RNNs in accurately classifying music genres based on raw audio data. |
| Music Genre Classification and Isolation | P. Devaki, A. Sivanandan, R. S. Kumar, M. Z. Peer | To develop a framework for music genre classification and isolation. | - Introduce a multi-stage framework combining feature extraction, clustering, and classification. | The paper presents a framework for accurately categorizing and isolating genre-specific segments within audio recordings. |
| Music Genre Classification using Machine Learning Techniques | Hareesh Bahuleya | To explore machine learning techniques for music genre classification. | - Utilize machine learning algorithms for classification. | The study demonstrates the efficacy of machine learning techniques in accurately categorizing music genres. |
| Music Genre Classification: A Review of Deep-Learning and Traditional Machine-Learning Approaches | N. Ndou, R. Ajoodha, A. Jadhav | To review deep learning and traditional machine learning approaches for music genre classification. | - Review feature extraction techniques and classification algorithms. | The review provides insights into the state-of-the-art techniques and challenges in music genre classification research. |

Table 3.1 : literature survey.

**Chapter 4**

**Software Requirements Specification**

The Software Requirements Specification (SRS) serves as a foundational document in the software development process, outlining the functional and non-functional requirements of the system to be developed. It serves as a communication bridge between stakeholders, including clients, developers, testers, and project managers, ensuring a common understanding of the software's objectives, features, and constraints.

**Scope:**

The scope section of the SRS defines the boundaries of the software system, delineating what the system will and will not do. It provides clarity on the project's objectives, target users, and intended functionalities, guiding the development team in designing and implementing the system effectively. The scope encompasses the following aspects:

**Purpose:**

* Clearly state the purpose of the software system, describing its intended use and the problems it aims to solve.
* Provide a brief overview of the project background and the context in which the software will be deployed.

**Objectives:**

* Enumerate the specific goals and objectives of the software system, outlining the desired outcomes and benefits for stakeholders.
* Define the primary objectives in terms of functionality, performance, usability, and other relevant criteria.

**Target Users:**

* Identify the primary users or user groups who will interact with the software system.
* Describe the characteristics and needs of each user group, including their technical proficiency, roles, and responsibilities.

**Functional Scope:**

* Define the key features and functionalities that the software system will deliver to meet the users' requirements.
* Specify the primary use cases and scenarios supported by the system, outlining the sequence of interactions between users and the system.

**Software Architecture:**

**A diagram of a computer data processing

Description automatically generated with medium confidence**

Figure 4.1 : Software Architecture.

**Chapter 5**

**Tools and Technology Used**

**Flask:**

Flask is a lightweight and versatile web framework for Python, designed to make web development simple and scalable. Below is a detailed overview of Flask, highlighting its key features and functionalities:

Minimalistic and Lightweight: Flask is minimalistic in design, providing only the essential components needed for web development. It has a small codebase and minimal dependencies, making it lightweight and easy to understand. This simplicity allows developers to focus on writing clean and concise code without being burdened by unnecessary abstractions.

Routing: Flask uses a straightforward routing mechanism to map URL paths to Python functions, known as view functions. Developers can define routes using decorators, making it easy to create RESTful APIs and handle HTTP requests and responses efficiently.

Templating: Flask comes with a built-in templating engine called Jinja2, which allows developers to create dynamic HTML pages by embedding Python code within HTML templates. Jinja2 provides features such as template inheritance, filters, and control structures, enabling the creation of modular and reusable templates.

HTTP Request Handling: Flask provides intuitive APIs for handling HTTP requests and accessing request data, such as form data, query parameters, and request headers. This simplifies request processing and enables developers to build interactive web applications with ease.

Session Management: Flask includes built-in support for managing user sessions and storing session data securely. Developers can use session variables to store user-specific information across multiple requests, facilitating user authentication, authorization, and personalization.

**Python:**

Python serves as the core programming language for developing the recommendation system due to its versatility, simplicity, readability, and extensive libraries. Here's a detailed overview of why Python is well-suited for machine learning and web development tasks:

Simplicity and Readability: Python is known for its simple and readable syntax, which makes it easy to learn and understand, especially for beginners. Its clean and concise code structure enhances developer productivity and reduces the time required for development and debugging.

Extensive Libraries: Python boasts a rich ecosystem of libraries and frameworks tailored for various tasks, including machine learning, web development, data analysis, and more. Some of the most prominent libraries for machine learning include TensorFlow, Keras, scikit-learn, and PyTorch. For web development, frameworks like Django and Flask provide robust tools and features for building web applications efficiently.

Machine Learning Capabilities: Python has emerged as one of the leading languages for machine learning and artificial intelligence applications. Its extensive libraries and frameworks offer powerful tools for data preprocessing, model development, training, evaluation, and deployment. With libraries like TensorFlow and scikit-learn, developers can implement complex machine learning algorithms with ease, making Python a preferred choice for recommendation systems.

**Jupyter Notebook:**

Jupyter Notebook serves as a crucial tool in the development process, specifically for prototyping, experimenting, and testing machine learning models within the recommendation system. Its interactive computing environment offers an intuitive platform for data exploration and model development, making it an ideal choice for iterative tasks. Jupyter Notebook allows developers to write and execute code in small, manageable chunks called cells, facilitating experimentation and rapid iteration. Moreover, its support for inline visualization enables real-time exploration of data and model outputs, aiding in the analysis and interpretation of results. Additionally, Jupyter Notebook provides a flexible and collaborative environment where multiple stakeholders

can collaborate on model development and share insights in a seamless manner. Overall, Jupyter Notebook plays a crucial role in the recommendation system development lifecycle, enabling developers to prototype, experiment, and refine machine learning models effectively.

**Libraries:**

**TensorFlow** stands as a powerful open-source machine learning framework developed by Google. It serves as the backbone for building and training neural network models, including complex deep learning architectures. Leveraging TensorFlow, developers can implement state-of-the-art algorithms for tasks such as recommendation system modeling.

**Keras:** complements TensorFlow by providing a high-level neural networks API that operates atop TensorFlow. Keras offers a user-friendly interface for constructing and training deep learning models, abstracting away the complexities of low-level TensorFlow implementation details. Its intuitive design makes it accessible to developers of all skill levels, facilitating rapid prototyping and experimentation.

**NumPy :** is fundamental for numerical computing within Python. It furnishes support for multi-dimensional arrays, mathematical functions, and linear algebra operations. These capabilities are indispensable for data manipulation and preprocessing tasks essential in recommendation system development.

**Pandas :**  serves as a popular library for data manipulation and analysis within Python. Its primary data structure, the DataFrame, enables efficient handling and manipulation of structured data. Pandas empowers developers to perform tasks such as data preprocessing and feature engineering with ease, facilitating the extraction of meaningful insights from raw data.

**Scikit-learn:** (sklearn) emerges as a versatile library for machine learning tasks within Python. It offers a vast array of algorithms and tools for tasks including classification, regression, clustering, and model evaluation. Scikit-learn streamlines the implementation of machine learning pipelines, providing robust functionality for building and deploying recommendation system models.

**Chapter 6**

**Functional requirement**

Audio File Upload:

* Users should be able to upload audio files in various formats (e.g., MP3, WAV) through the system's interface.
* The system should support single file uploads as well as batch uploads for multiple files simultaneously.

Genre Classification:

* The system should analyze uploaded audio files and classify them into predefined music genres.
* Classification should be based on features extracted from audio signals, such as MFCCs, spectral characteristics, and rhythmic patterns.

Real-time Prediction:

* Users should receive genre predictions in real-time after uploading audio files.
* Predictions should be displayed along with confidence scores indicating the system's confidence in each prediction.

**Chapter 7**

**Non Functional Requirements**

Non-functional requirements define the quality attributes or constraints that specify how the system should behave or perform, rather than what it should do. Here are some non-functional requirements for your music genre classification project:

**Performance:**

* The system should be able to classify music genres accurately and efficiently, even with large audio datasets.
* Genre classification should be completed within a reasonable time frame to provide real-time feedback to users.

**Scalability:**

* The system should be scalable to handle increasing numbers of users, audio uploads, and classification requests.
* It should be able to accommodate future growth in data volume without sacrificing performance.

**Reliability:**

* The classification algorithm should be reliable and consistent, producing accurate genre predictions for a wide range of music tracks.
* The system should have built-in error detection and recovery mechanisms to handle unexpected failures gracefully.

**Security:**

* User data and uploaded audio files should be stored securely and protected from unauthorized access or manipulation.
* The system should implement authentication and authorization mechanisms to control access to sensitive functionalities and data.

**Usability:**

* The user interface should be intuitive and easy to navigate, catering to users with varying levels of technical expertise.
* Feedback and error messages should be clear and concise, helping users understand system behavior and resolve issues effectively.

**Compatibility:**

* The system should be compatible with a variety of web browsers and devices, ensuring a consistent user experience across different platforms.
* It should support common audio file formats and metadata standards to facilitate seamless integration with external systems.

**Maintainability:**

* The system should be designed and implemented in a modular and extensible manner, allowing for easy maintenance and future enhancements.
* Codebase should be well-documented and adhere to coding standards to facilitate code reviews and collaboration among developers.

**Performance:**

* The system should be able to classify music genres accurately and efficiently, even with large audio datasets.
* Genre classification should be completed within a reasonable time frame to provide real-time feedback to users.

**Availability:**

* The system should have high availability, with minimal downtime during maintenance or upgrades.
* Redundancy and failover mechanisms should be in place to ensure continuous operation in case of hardware or software failures.

**Chapter 8**

**System Requirements**

* 1. **Hardware Requirements:**

|  |  |
| --- | --- |
| * PROCESSOR | : Intel i3 |
| * HARD-DISK | :500GB |
| * RAM | :4GB or Above |

* 1. **Software Requirements:**

|  |  |
| --- | --- |
| * OPERATING SYSTEM | : Windows 7 and above |
| * FRONT END | : Html, CSS |
| * FRAMEWORK | : Flask |
| * LANGUAGE | : Python version 3.7 |
| * LIBRARIES | :Pandas,Numpy,Sklearn,Scikit |
| * EDITOR | : Sublime Text 3 |

**Chapter 9**

**System Design**

The system design for the music genre classification project encompasses a comprehensive architecture that seamlessly integrates various components to achieve efficient and accurate genre predictions. At the core of the design lies a client-server architecture, wherein the client-side comprises the user interface (UI) responsible for facilitating user interactions, such as uploading audio files and displaying genre predictions. On the server-side, the system hosts backend services responsible for processing uploaded audio files, extracting relevant features, and performing genre classification using machine learning models.

The user interface is designed with usability in mind, featuring an intuitive upload interface that allows users to effortlessly upload audio files of various formats. Additionally, the interface provides real-time feedback to users, displaying genre predictions alongside confidence scores derived from the classification process. To further enhance user engagement, the interface includes a feedback mechanism enabling users to provide input on the accuracy of predictions, thereby contributing to the continuous improvement of the classification algorithm.

Within the backend services, sophisticated algorithms are employed for feature extraction and genre classification. These algorithms extract pertinent features from audio signals, such as Mel-Frequency Cepstral Coefficients (MFCCs), spectral characteristics, and rhythmic patterns. Leveraging machine learning models, such as Support Vector Machines (SVM), Convolutional Neural Networks (CNNs), or ensemble methods, the system classifies audio files into predefined music genres with high accuracy. Model training involves optimizing hyperparameters and architectures using labeled audio datasets, while model evaluation ensures robust performance through metrics like accuracy, precision, and recall.

The system design also addresses crucial aspects such as data management, integration, deployment, security, scalability, performance, monitoring, and documentation. Data management strategies encompass database design for storing metadata associated with audio files, while integration involves designing a RESTful API for seamless integration

with external systems. Deployment considerations include selecting appropriate hosting options and implementing CI/CD pipelines for automated testing and deployment. Security measures encompass user authentication, data encryption, and regular security audits, ensuring the protection of sensitive data and system integrity.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | * 1. **DFD 1**   The Data Flow Diagram (DFD) provides a comprehensive overview of the flow of data within the music genre classification system. At the highest level, known as Level 0 DFD, the diagram illustrates the primary external entities interacting with the system. The central external entity is the user, who initiates interactions with the system by uploading audio files for genre classification. Additionally, there may be other external entities, such as administrators or external systems, which interact with the system for various purposes.The diagram depicts the flow of data between these external entities and the system's internal components, including processes and data stores. Processes represent the functions or operations performed within the system, such as audio file processing, feature extraction, genre classification, and user feedback handling. These processes manipulate and transform the data as it flows through the system, facilitating the classification process and providing feedback to users.Data stores represent repositories where data is stored and retrieved as needed. In the context of the music genre classification system, data stores may include databases for storing metadata associated with audio files, model parameters, user feedback, and system logs. These data stores serve as sources of input for processes and destinations for output data, facilitating data management and retrieval operations.   * 1. **Activity Daigram**   An activivity diagram outwardly presents a progression of activities or stream of control in a framework like a flowchart or an information stream chart. Action graphs are regularly utilized in business measure demonstrating. They can likewise depict the means in an utilization case chart. Exercises demonstrated can be consecutive and simultaneous. In the two cases, an action outline will have a start (an underlying state) and an end (a last state).  initial state) and an end (a final state).  Start point symbol - Activity diagram  Activity symbol - Activity diagram  Action flow - Activity diagram  Object flow - Activity diagram  Decision symbol - Activity diagram  **A diagram of a process  Description automatically generated**Guard symbol - Activity diagram |

Figure 9.2.1 : Activity Daigram.

* 1. **Sequence Daigram:**

**A diagram of a data processing process

Description automatically generated**sequence diagram depict cooperations among classes as far as a trade of messages after some time. They're likewise called occasion charts. A grouping chart is a decent method to envision and approve different runtime situations. These can assist with anticipating how a framework will act and to find duties a class may need to have during the time spent demonstrating another framework.

Figure 9.3.1 : Sequential Daigram.

* 1. **Use Case Daigram:**

The motivation behind use case diagram is to catch the dynamic part of a framework. In any case, this definition is too nonexclusive to even think about describing the reason, as other four outlines (action, grouping, cooperation, and Statechart) likewise have a similar reason. We will investigate some particular reason, which will recognize it from other four charts.

Use case graphs are utilized to accumulate the prerequisites of a framework including inside and outside impacts. These prerequisites are generally plan necessities.

Consequently, when a framework is investigated to accumulate its functionalities, use cases are readied and entertainers are distinguished.

**A diagram of a system

Description automatically generated**

Figure 9.4.1 : Use Case Daigram.

### **Chapter 10**

### **Implementation**

* 1. **Introduction**

The project is implemented using Python which is an object oriented programming language and procedure oriented programming language. Object oriented programming is an approach that provides a way of modularizing program by creating partitioned memory area of both data and function that can be used as a template for creating copies of such module on demand.This project is implemented using python programming language. Python is [dynamically typed](https://en.wikipedia.org/wiki/Dynamic_programming_language) and [garbage-collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigms), including [procedural](https://en.wikipedia.org/wiki/Procedural_programming), object-oriented, and [functional programming](https://en.wikipedia.org/wiki/Functional_programming). Python is often described as a "batteries included" language due to its comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library). The machine Learning techniques are used in this project.

* 1. **Machine Learning overview:**

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data. In this article, we’ll see basics of Machine Learning, and implementation of a simple machine learning algorithm using python.

Machine learning involves a computer to be trained using a given data set, and use this training to predict the properties of a given new data. For example, we can train a computer by feeding it 1000 images of cats and 1000 more images which are not of a cat, and tell each time to the computer whether a picture is cat or not. Then if we show the computer a new image, then from the above training, the computer should be able to tell whether this new image is a cat or not.The process of training and prediction involves the use of specialized algorithms. We feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. One such algorithm is [K-Nearest-Neighbor](https://www.geeksforgeeks.org/k-nearest-neighbours/) classification (KNN classification). It takes a test data, and finds k nearest data values to this data from test data set. Then it selects the neighbor of maximum frequency and gives its properties as the prediction result.

* 1. **Challenges in Implementing Machine Learning:**

Most insurers recognize the value of machine learning in driving better decision-making and streamlining business processes. Research for the Accenture Technology Vision 2018 shows that more than 90 percent of insurers are using, plan to use or considering using machine learning or AI in the claims or underwriting process. Some of the challenges insurers typically encounter when adopting machine learning are.

**Training requirements** AI-powered intellectual systems must be trained in a domain, e.g., claims or billing for an insurer. This requires a separate training system, which insurers find hard to provide for training the AI model. Models need to be trained with huge volumes of documents/transactions to cover all possible scenarios.

**Right data source** The quality of data used to train predictive models is equally important as the quantity, in the case of machine learning. The datasets need to be representative and balanced so that they can give a better picture and avoid bias. This is important to train predictive models. Generally, insurers struggle to provide relevant data for training AI models

**Difficulty in predicting returns** It’s not very easy to predict improvements that machine learning can bring to a project. For example, it’s not easy to plan or budget a project using machine learning, as the funding needs may vary during the project, based on the findings. Therefore, it is almost impossible to predict the return on investment. This makes it hard to get everyone on board the concept and invest in it.

**Data security** The huge amount of data used for machine learning algorithms has created an additional security risk for insurance companies. With such an increase in collected data and connectivity among applications, there is a risk of data leaks and security breaches. A security incident could lead to personal information falling into the wrong hands. This creates fear in the minds of insurers.

* 1. **Archietecture:**

A diagram of data processing process

Description automatically generated

Figure 10.4.1 : Architecture.

**Data Collection:**

* Gather a diverse dataset of music tracks covering various genres, sourced from reputable repositories or databases like Kaggle, GTZAN, or academic datasets.
* Ensure that the dataset includes sufficient samples for each genre, with balanced representation to prevent biases in the classification model.

**Data Preprocessing:**

* Clean the dataset by removing duplicates, errors, or irrelevant entries.
* Extract relevant features from audio files using signal processing techniques, such as Mel-Frequency Cepstral Coefficients (MFCCs), spectrograms, or chroma features.
* Normalize the extracted features to ensure consistent scaling and improve model performance.

**Model Selection:**

* Explore and evaluate various machine learning algorithms suitable for music genre classification, such as Support Vector Machines (SVM), Decision Trees, Random Forests, Naive Bayes, K-Nearest Neighbors (KNN), or Deep Learning models like Convolutional Neural Networks (CNNs).
* Consider the advantages and limitations of each algorithm in terms of classification accuracy, scalability, and computational complexity.

**Model Training:**

* Split the dataset into training, validation, and test sets to evaluate model performance effectively.
* Train the selected models using the training dataset, optimizing hyperparameters and model architectures to maximize classification accuracy.
* Implement techniques such as cross-validation, grid search, or random search to fine-tune model parameters and prevent overfitting.

**Model Evaluation:**

* Evaluate the trained models using the validation set to assess their performance in terms of classification accuracy, precision, recall, F1-score, and other relevant metrics.
* Compare the performance of different models to identify the most effective approach for music genre classification.
* Conduct statistical tests, such as t-tests or ANOVA, to determine the significance of performance differences between models.

**Model Deployment:**

* Deploy the trained model with the highest performance on the test set to make real-time genre predictions.
* Integrate the model into a user-friendly web application or API, allowing users to upload audio files and receive genre predictions interactively.
* Implement robust error handling, logging, and monitoring mechanisms to ensure system reliability and stability.
  1. **KNN :**

KNN is also a **lazy** algorithm (as opposed to an eager algorithm). this means is that it does not use the training data points to do any generalization. In other words, there is no explicit training phase or it is very minimal. This also means that the training phase is pretty fast . Lack of generalization means that KNN keeps all the training data. To be more exact, all (or most) the training data is needed during the testing phase.

KNN Algorithm is based on **feature similarity**: How closely out-of-sample features resemble our training set determines how we classify a given data point:

KNN can be used for **classification** — the output is a class membership (predicts a class — a discrete value). An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. It can also be used for **regression** — output is the value for the object (predicts continuous values). This value is the average (or median) of the values of its k nearest neighbors.

**KNN**

Input: uploading datasets

begin

1. Scan the dataset (storage servers). Retrieval of required data for mining from the servers such as database, cloud, excel sheet etc.
2. Determine Parameter K= number of nearest neighbours.
3. Calculate the distance between the query-instance and all the training samples. There are many distance functions but Euclidean is the most commonly used measure.
4. Sort the distance and determine nearest neighbours based on the K-th minimum distance.
5. Gather the category X of the nearest neighbours.
6. Use simple majority of the category of nearest neighbors as the Prediction value of the query instance end

**Knn Algorithm Pseudocode:**

1. Calculate “d(x, xi)” i =1, 2, ….., **n**; where **d** denotes the [Euclidean distance](https://dataaspirant.com/2015/04/11/five-most-popular-similarity-measures-implementation-in-python/) between the points.
2. def euclidean\_distance(x,y):
3. return sqrt(sum(pow(a-b,2) for a, b in zip(x, y)))
4. Arrange the calculated **n** Euclidean distances in non-decreasing order.
5. Let **k** be a +ve integer, take the first **k** distances from this sorted list.
6. Find those **k**-points corresponding to these **k**-distances.
7. Let **k**i denotes the number of points belonging to the ith class among **k** points i.e. k ≥ 0
8. If ki >kj ∀ i ≠ j then put x in class i.
   1. **Naïve Bayes:**

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets.

Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location. Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence

**Naive Bayes Algorithm Steps**

Input: uploading datasets

begin

1. Scan the dataset (storage servers) .Retrieval of required data for mining from the servers such as database, cloud, excel sheet etc.
2. Calculate the probability of each attribute value. Here for each attribute we calculate the probability of occurrence.
3. Multiply the probabilities by p. For each class, here we multiple the results of each attribute with p and final results are used for classification.
4. Compare the values and classify the attribute values to one of the predefined set of class.

end

**Naïve Bayes:**

* Derivation:
* D : Set of tuples
* Each Tuple is an ‘n’ dimensional attribute vector
* X : (x1,x2,x3,…. xn)
* Let there be ‘m’ Classes : C1,C2,C3…Cm
* Naïve Bayes classifier predicts X belongs to Class Ci iff
* P (Ci/X) > P(Cj/X) for 1<= j <= m , j <> i Maximum Posteriori Hypothesis
* P(Ci/X) = P(X/Ci) P(Ci) / P(X)
* Maximize P(X/Ci) P(Ci) as P(X) is constant With many attributes, it is computationally expensive to evaluate P(X/Ci). Naïve Assumption of “class conditional independence”
* ∏= = n k P X Ci P xk Ci 1 ( xk./ Ci)
* P(X/Ci) = P(x1/Ci) \* P(x2/Ci) \*…\* P(xn/ Ci)

P(A|B) = Fraction of worlds in which B is true that also have A true

P(A ^ B) P(A|B) = ------------------ P(B)

Corollary:

P(A ^ B) = P(A|B) P(B) P(A|B)+P( ¬A|B) = 1

* 1. **Decision Tree:**

Input: uploading datasets

begin

1. Scan the dataset (storage servers)
2. for each attribute a, calculate the gain [number of occurrences]
3. Let A be the attribute of highest gain [highest count]
4. Create a decision node based on a A – retrieval of nodes[patient] where the attribute values matches with A.
5. recur on the sub-lists [list of patient] and calculate the count of outcomes –termed as sub nodes. Based on the highest count we classify the new node.

End

* 1. **PSEUDOCODE:**

### **Information Gain**

infoGain(examples, attribute, entropyOfSet)

gain = entropyOfSet

for value in attributeValues(examples, attribute):

sub = subset(examples, attribute, value)

gain -= (number in sub)/(total number of examples) \* entropy(sub)

return gain

### **Entropy**

entropy(examples)

'''

log2(x) = log(x)/log(2)

'''

result = 0

# handle target attributes with arbitrary labels

dictionary = summarizeExamples(examples, targetAttribute)

for key in dictionary:

proportion = dictionary[key]/total number of examples

result -= proportion \* log2(proportion)

return result

* 1. **Decision Tree Algorithm:**

Decision Tree is a supervised learning approach that may be used to classification and regression issues, though it is most commonly employed to solve classification

issues. Internal nodes contain dataset attributes, branches represent decision tree, and each leaf node provides the conclusion in a tree-structured classifier. The node’s outcome is represented by the branches/edges, and the nodes have either:

1. Conditions [Decision Nodes]
2. Result [End Nodes]

Input: uploading datasets

Begin

1. Scan the dataset (storage servers)
2. for each attribute a, calculate the gain [number of occurrences]
3. Let X be the attribute of highest gain [highest count]
4. Create a decision node based on a X – retrieval of nodes[patient] where theattribute values matches with X.
5. recur on the sub-lists [list of patient] and calculate the count of outcomestermed as sub nodes.
6. Based on the highest count we classify the new node.
7. end

The accuracy of the Decision Tree applied to the Crop dataset is obtained, as well as the Classification Report received from the model and the Support number of actual occurances report.

* 1. **Naive Bayes classifier**

The Naive Bayes algorithm is an effective learning technique for addressing classification issues that is based on the Bayes theorem. It is an appropriate and useful classification technique that aids in the development of rapid machine learning models capable of making immediate predictions.

Naive Bayes classifier predicts X belongs to below formula

P(C*i/X*) = *P*(*X/Ci*)*P*(*Ci*)*/P*(*X*)

*P*(*X/Ci*) = *P*(*x*1*/Ci*) ∗ *P*(*x/Ci*) ∗ *...* ∗ *P*(*xn/Ci*)

Steps followed by Navie Bayes classifier

1. Input the training dataset say T along with predictor variable values F=(f1,f2....fn)
2. Mean and standard deviation of the predictor variable of each class must becalculated.
3. Repeat the previous step By making use of Gaussian density equation ineach class calculate the probability of fi. Repeat until all predictor variables are calculated.
4. Likelihood of each class is computed.
5. Highest likelihood is accepted as the output.

The accuracy of the Naive Bayes Classifier applied to the Crop dataset is obtained, as well as the Classification Report received by the model and the Support number of actual occurances report.

* 1. **Support Vector Machine:**

In multi - dimensional space, an Support Vector Machine model is essentially a representation of distinct classes in a hyperplane. Support Vector Machine will build the hyperplane in an iterative process in order to reduce the error. Support Vector Machine’s purpose is to partition datasets into classes such that a maximum marginal hyperplane may be found.

1. Initialize Support vector which is known as a candidate to closest pair fromopposite class.
2. If violating points are found then do:

Find out the violator

Candidate SV is integrated with new candidate SV

Call violator

If any p¡ 0 due to the addition of c to S then do: Candidate SV must be divided by candidate SV and p Repeat the steps until it is completely pruned.

1. End if statement
2. End while

The accuracy of the Support Vector Machine applied to the Crop dataset is obtained, as well as the Classification Report received from the model and the Support number of actual occurances report.

* 1. **Logistic Regression**

When the dependent variable is binary, logistic regression is the proper regression strategy to use. A predictive modeling is logistic regression. To summarize data and analyze the connection between one dependent binary variable and one or more ordinal, interval, nominal, or ratio-level independent variables, logistic regression is utilised. tree-structured classifier. following steps are performed for Logistic Regression classifier

1. Initialize the parameters A1, A2.
2. The predict dependent variable must be calculated.
3. Cost function for the model must be calculated.
4. Once the cost function is calculated, compute the gradient for the cost function.
5. All the parameters must be updated.
6. Repeat the steps 2 and 5 until the desired result is obtained.

The accuracy of Logistic Regression applied to the Crop dataset is obtained, as well as the Classification Report and Support Number of Actual Occurrences Report.

* 1. **Random Forest Algorithm:**

Random forest is a supervised learning method that may be used to classify and predict data. However, it is mostly used to solve classification problems. Any forest, as we all know, is usually made of trees, and more trees equals a more strong forest. Similarly, the random forest method constructs decision trees from datasets, extracts predictions from each, and then votes on the best option. It’s an ensemble approach that’s superior than a single feature since it averages the results to reduce over-fitting.

Random Forest Algorithm steps:

1. Start
2. Randomly select “k” features from total “m” features, Where k *<<* m.
3. Among the “k” features, calculate the node “d” using the best split point.
4. Split the node into daughter nodes using the best split.
5. Repeat 1 to 3 steps until “l” number of nodes has been reached. Build forestby repeating steps 1 to 4 for “n” number times to create “n” number of trees 6. end

Above steps are conducted to classify the data in k features, and it calculates the nodes and splits according dataset similar features and continuously repeats the steps up to n times until the trees are formed using the best split it gives valuable result with good accuracy. Accuracy of Random Forest Classifier applied for Crop dataset and Classification Report obtained from model and Support number of actual occurrences report is obtained

Random Forest Classifier applied for Crop dataset and Classification Report obtained from model and Support number of actual occurrences report is obtained

**Chapter 11**

**TESTING**

* 1. **Introduction:**

Testing is the way toward running a framework with the expectation of discovering blunders. Testing upgrades the uprightness of the framework by distinguishing the deviations in plans and blunders in the framework. Testing targets distinguishing blunders – prom zones. This aides in the avoidance of mistakes in the framework. Testing additionally adds esteems to the item by affirming the client's necessity.

The primary intention is to distinguish blunders and mistake get-prom zones in a framework. Testing must be intensive and all around arranged. A somewhat tried framework is as terrible as an untested framework. Furthermore, the cost of an untested and under-tried framework is high. The execution is the last and significant stage. It includes client preparation, framework testing so as to guarantee the effective running of the proposed framework. The client tests the framework and changes are made by their requirements. The testing includes the testing of the created framework utilizing different sorts of information. While testing, blunders are noted and rightness is the mode.

* 1. **Objectives Of Testing:**

• Testing in a cycle of executing a program with the expectation of discovering mistakes.

• A effective experiment is one that reveals an up 'til now unfamiliar blunder.

Framework testing is a phase of usage, which is pointed toward guaranteeing that the framework works accurately and productively according to the client's need before the live activity initiates. As expressed previously, testing is indispensable to the achievement of a framework. Framework testing makes the coherent presumption that if all the framework is right, the objective will be effectively accomplished. A progression of tests are performed before the framework is prepared for the client acknowledgment test.

* 1. **Testing Methods:**

System testing is a stage of implementation. This helps the weather system works accurately and efficiently before live operation commences. Testing is vital to the success of the system. The candidate system is subject to a variety of tests: online response, volume, stress, recovery, security, and usability tests series of tests are performed for the proposed system are ready for user acceptance testing.

**White Box Testing:**

The test is conducted during the code generation phase itself. All the errors were rectified at the moment of its discovery. During this testing, it is ensured that

• All independent module have been exercised at least one

• Exercise all logical decisions on their true or false side.

• Execute all loops at their boundaries.

**Black Box Testing:**

It is focused around the practical necessities of the product. It's anything but a choice to white box testing; rather, it is a reciprocal methodology that is probably going to reveal an alternate class of blunders than White Box strategies. It is endeavored to discover mistakes in the accompanying classes.

• Incorrect or missing capacities

• Interface blunders

• Errors in an information structure or outside information base access

**Unit Testing:**

Unit testing chiefly centers around the littlest unit of programming plan. This is known as module testing. The modules are tried independently. The test is done during the programming stage itself. In this progression, every module is discovered to be working acceptably as respects the normal yield from the module.

**Integration Testing:**

Mix testing is an efficient methodology for developing the program structure, while simultaneously leading tests to reveal blunders related with the interface. The goal is to take unit tried modules and manufacture a program structure. All the modules are joined and tried in general.

**Output Testing:**

Subsequent to performing approval testing, the following stage is yield trying of the proposed framework, since no framework could be valuable on the off chance that it doesn't create the necessary yield in a particular configuration. The yield design on the screen is discovered to be right. The organization was planned in the framework configuration time as indicated by the client needs. For the printed copy likewise, the yield comes according to the predefined prerequisites by the client. Subsequently yield testing didn't bring about any amendment for the framework.

**User Acceptance Testing:**

Client acknowledgment of a framework is the vital factor for the achievement of any framework. The framework viable is tried for client acknowledgment by continually staying in contact with the imminent framework clients at the hour of creating and making changes at whatever point required.

* 1. **Validation:**

Toward the consummation of the reconciliation testing, the product is totally amassed as bundle interfacing blunders have been revealed and adjusted and a last arrangement of programming tests starts in approval testing. Approval testing can be characterized from multiple points of view, however a straightforward definition is that the approval succeeds when the product work in a way that is normal by the client. After approval test has been directed as follows:

* The capacity or execution qualities adjust to detail and are acknowledged.
* A deviation from the particular is revealed and a lack list is made.
* Proposed framework viable has been tried by utilizing an approval test and discovered to be working acceptably.
  1. **Test Reports:**

The users test the developed system when changes are made according to the needs. The testing phase involves the testing of the developed system using various kinds of data. An elaborate testing of data is prepared and system is tested using the test data. Test cases are used to check for outputs with different set of inputs.

| **Test Case ID** | **Test Case Description** | **Input** | **Expected Output** | **Pass/Fail** | **Comments** |
| --- | --- | --- | --- | --- | --- |
| TC\_001 | Single Audio File Upload Test | Audio file: sample1.mp3 | Genre prediction for sample1.mp3 | Pass | Verify if the system correctly predicts the genre of the uploaded audio file. |
| TC\_002 | Batch Audio File Upload Test | Audio files: sample2.mp3, sample3.mp3 | Genre predictions for sample2.mp3, sample3.mp3 | Pass | Verify if the system correctly predicts the genres of multiple uploaded audio files simultaneously. |
| TC\_003 | Real-time Prediction Test | Live audio stream | Genre prediction in real-time | Pass | Verify if the system accurately predicts the genre as the live audio stream is processed. |
| TC\_004 | Feedback Submission Test | User provides feedback on prediction accuracy | Feedback recorded and used for model improvement | Pass | Verify if the system correctly captures user feedback and incorporates it for model refinement. |
| TC\_005 | Incorrect File Format Handling Test | Unsupported audio file format | Error message: "Unsupported file format" | Pass | Verify if the system correctly handles unsupported audio file formats and provides appropriate feedback. |
| TC\_006 | Empty File Handling Test | Empty audio file | Error message: "Empty file" | Pass | Verify if the system correctly handles empty audio files and provides appropriate feedback. |
| TC\_007 | Unauthorized Access Test | User attempts to access admin functionalities | Error message: "Access denied" | Pass | Verify if the system restricts unauthorized users from accessing administrative features. |
| TC\_008 | Performance Test | Upload multiple large audio files | Timely genre predictions for each file | Pass | Verify if the system handles large volumes of audio data efficiently and provides timely predictions. |
| TC\_009 | Security Test | Attempt SQL injection attack | System prevents SQL injection and logs attempt | Pass | Verify if the system safeguards against SQL injection attacks and logs security incidents. |
| TC\_010 | Scalability Test | Simulate high user traffic | System maintains performance under load | Pass | Verify if the system scales horizontally to handle increased user traffic without degradation in performance. |

Table: 11.5.1 : Test Reports.

**Chapter 12**

**Conclusion**

the music genre classification project has successfully demonstrated the feasibility and effectiveness of utilizing machine learning techniques to categorize audio files into distinct genres. Through meticulous data collection, preprocessing, model selection, and evaluation, the system has achieved commendable accuracy in predicting music genres across various test scenarios.The project's methodology involved a systematic approach, encompassing data collection from diverse sources, feature extraction, model training, and deployment within a user-friendly web application. The utilization of state-of-the-art machine learning algorithms, including Support Vector Machines (SVM), Decision Trees, and Convolutional Neural Networks (CNNs), enabled the system to achieve high accuracy and robustness in genre classification.One of the project's significant achievements lies in its ability to handle real-world challenges such as handling diverse audio formats, processing large volumes of audio data efficiently, and providing timely genre predictions. Additionally, the integration of user feedback mechanisms has facilitated continuous improvement and refinement of the classification models, ensuring adaptability to evolving music trends and user preferences.The successful deployment of the music genre classification system underscores its practical utility in various domains, including music recommendation systems, content tagging, and audio content analysis. Furthermore, the project's open-ended architecture and modular design allow for scalability, extensibility, and integration with external systems, paving the way for future enhancements and collaborations.the project has not only advanced the state-of-the-art in music genre classification but also demonstrated the transformative potential of machine learning in automating and enhancing audio content analysis tasks. With further refinement and optimization, the system holds promise for broader applications in the field of digital media, entertainment, and content management.

**Chapter 13**

**Future Scope**

* The future scope of the music genre classification project encompasses several avenues for further research, development, and application. Here are some potential areas of exploration:
* Enhanced Feature Extraction: Investigate advanced signal processing techniques and feature extraction methods to capture more nuanced aspects of audio content, such as rhythm, timbre, and harmony. Experiment with deep learning architectures, such as recurrent neural networks (RNNs) and attention mechanisms, to learn hierarchical representations of audio features automatically.
* Multi-modal Classification: Explore multi-modal approaches that combine audio features with other modalities, such as text metadata (e.g., song titles, artist names) or image spectrograms, to improve classification accuracy and robustness. Investigate fusion techniques, such as late fusion or early fusion, to effectively integrate information from different modalities.
* Incremental Learning and Adaptation: Develop mechanisms for incremental learning and model adaptation to accommodate evolving music genres, styles, and preferences over time. Implement online learning techniques that allow the model to continuously update and refine its predictions based on incoming data streams and user feedback.

**Chapter 14**

**References**

[1] A. Ghildiyal, K. Singh and S. Sharma, "Music Genre Classification using Machine Learning," 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2020, pp. 1368-1372, doi: 10.1109/ICECA49313.2020.9297444.

[2] A. Elbir, H. Bilal Çam, M. Emre Iyican, B. Öztürk and N. Aydin, "Music Genre Classification and Recommendation by Using Machine Learning Techniques," 2018 Innovations in Intelligent Systems and Applications Conference (ASYU), Adana, Turkey, 2018, pp. 1-5, doi: 10.1109/ASYU.2018.85540

[3] I. Pathania and N. Kaur, "Classification of Music Genre Using Machine Learning," 2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2022, pp. 1-5, doi: 10.1109/GCAT55367.2022.9972105.

[4] V. Shah, A. Tandle, N. Sharma and V. Sheth, "Genre Based Music Classification using Machine Learning and Convolutional Neural Networks," 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2021, pp. 1-8, doi: 10.1109/ICCCNT51525.2021.9579597.

[5] V. Prashanthi, S. Kanakala, V. Akila and A. Harshavardhan, "Music Genre Categorization using Machine learning Algorithms," 2021 International Conference on Computational Intelligence and Computing Applications (ICCICA), Nagpur, India, 2021, pp. 1-4, doi: 10.1109/ICCICA52458.2021.9697137

[6] K. S. Mounika, S. Deyaradevi, K. Swetha and V. Vanitha, "Music Genre Classification Using Deep Learning," 2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA), Coimbatore, India, 2021, pp. 1-7, doi: 10.1109/ICAECA52838.2021.9675685.

[7] M. Shah, N. Pujara, K. Mangaroliya, L. Gohil, T. Vyas and S. Degadwala, "Music Genre Classification using Deep Learning," 2022 6th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2022, pp. 974-978, doi: 10.1109/ICCMC53470.2022.9753953.

[8] P. Devaki, A. Sivanandan, R. S. Kumar and M. Z. Peer, "Music Genre Classification and Isolation," 2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA), Coimbatore, India, 2021, pp. 1-6, doi: 10.1109/ICAECA52838.2021.9675518.

[9] [Hareesh Bahuleya](https://www.researchgate.net/profile/Hareesh-Bahuleyan?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uIn19) Music Genre Classification using Machine Learning Techniques

[10] N. Ndou, R. Ajoodha and A. Jadhav, "Music Genre Classification: A Review of Deep-Learning and Traditional Machine-Learning Approaches," 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), Toronto, ON, Canada, 2021, pp. 1-6, doi: 10.1109/IEMTRONICS52119.2021.9422487.