SPEECH EMOTION RECOGNITION

A PROJECT REPORT SUBMITTED TO

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

AWARD OF THE DEGREE OF

MASTER OF APPLIED DATA SCIENCE

BY

PREEDIKA D (REG NO.RA2332014010141)

SNEHA GEORGE (REG NO.RA2332014010160)

ASHA VARGHEESE (REG NO.RA2332014010161)

DEVDATH K K (REG NO.RA2332014010173)

UNDER THE GUIDANCE OF

Dr.V. NISHA, M.Sc., M.Phil., SET., PhD., NET.

DEPARTMENT OF COMPUTER APPLICATIONS

FACULTY OF SCIENCE AND HUMANITIES

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Kattankulathur – 603 203

Chennai, Tamilnadu

OCTOBER - 2024

BONAFIDE CERTIFICATE

This is to certify that the project report titled **"SPEECH EMOTION RECOGNITION"** is a bonafide work carried out by **PREEDIKA D** (RA2332014010141), **SNEHA GEORGE**(RA2332014010160), **ASHA VARGHEESE**(RA2332014010161), **DEVDATH K K** (RA2132241010002) under my supervision for the award of the Degree of Master of Applied Data Science. To my knowledge the work reported here in is the original work done by these students.

Dr . NISHA

Assistant Professor,

Department of Computer Applications

(GUIDE)

Dr. R.Jayashree

Associate Professor & Head,

Department of Computer Applications

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

With profound gratitude to the ALMIGHTY, I take this chance to thank the people who helped me to complete this project.

We take this as a right opportunity to say THANKS to my parents who are there to stand with me always with the words “YOU CAN”.

We are thankful to **Dr. T. R. Paarivendhar**, Chancellor, and **Prof. A. Vinay Kumar**, Pro Vice-Chancellor (SBL), SRM Institute of Science & Technology, who gave us the platform to establish me to reach greater heights.

We earnestly thank **Dr. A. Duraisamy**, Dean, Faculty of Science and Humanities, SRM Institute of Science & Technology, who always encourage us to do novel things.

A great note of gratitude to **Dr. S. Albert Antony Raj**, Deputy Dean, Faculty of Science and Humanities for his valuable guidance and constant Support to do this Project.

We express our sincere thanks to **Dr. R. Jayashree**, Associate Professor & Head, for her support to execute all incline in learning.

It is our delight to thank our project guide **Dr. V. Nisha ,** Assistant Professor, Department of Computer Applications, for his help, support, encouragement, suggestions, and guidance throughout the development phases of the project.

We convey our gratitude to all the faculty members of the department who extended their support through valuable comments and suggestions during the reviews.

Our gratitude to friends and people who are known and unknown to me who helped in carrying out this project work a successful one.

PREEDIKA D

SNEHA GEORGE

ASHA VARGHEESE

DEVDATH K K

PLAIGARISM REPORT

Nisha V

Speech Recognition

BOOKSKEY

MCA

SRM Institute of Science & Technology

Document Details

Submission ID

trn:oid:::1:3063211760

Submission Date

Nov 1, 2024, 4:57 PM GMT+5:30

Download Date

Nov 1, 2024, 4:57 PM GMT+5:30

File Name

Abstract_final.pdf

File Size

57.5 KB

1 Page

266 Words

1,631 Characters

0% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

0 Not Cited or Quoted 0%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

0% Internet sources

0% Publications

0% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

TABLE OF CONTENT

ABSTRACT

Emotion is a complex state of human being that depicts the physical, physiological or mental condition of a person. According to the human sciences, emotion is a mental process of neural mechanisms and disorders. Preprocessing of speech signal digitally is crucial for various tasks like automatic voice recognition, high speed and other technologies. Its applications varies in multiple sectors from healthcare to military wherein different processes like feature extraction and feature matching are the current problems related to the signaling of voice. Therefore, this application allows users to upload audio files , which are processed in real-time using the librosa library to extract essential acoustic features. The features include Mel- frequency Cepstral Coefficients (MFCC), chroma , and mel spectrograms each capturing unique spectral and temporal qualities of speech that are indicative of emotional states.

The extracted features are used as inputs to a machine learning model trained specifically for emotion classification. The model , serialized as a pickle file , analyses these features to predict emotions such as happiness , sadness , disgust , or fearful. This prediction is then displayed on the web interface , providing users with immediate feedback on the emotional tone of the analysed audio. The project's architecture demonstrates an end-to-end pipeline , from audio preprocessing and feature extraction to emotion classification and front-end integration , highlighting the practical use of machine learning in speech – based applications. The use of Flask enables efficient handling of user requests and model inference , while the librosa library provides robust tools for high-quality audio feature extraction. This application exemplifies a powerful , scalable solution for speech emotion recognition , with potential real-world impact **across various industries.**

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

MFCC	Mel-Frequency Cepstral Coefficients
RNN	Recurrent Neural Network
HTML	HyperText Markup Language
SER	Speech Emotion Recognition
DL	Deep Learning

CHAPTER 1

INTRODUCTION

Speech Emotion Recognition (SER) has become a significant field within human-computer interaction, as understanding emotions from speech data can greatly enhance the personalization and responsiveness of technology. The project detailed here explores the development of a Speech Emotion Recognition system using a Multi-Layer Perceptron (MLP) neural network. The objective is to accurately identify and classify emotions, such as calm, happy, fearful, and disgusted, from audio speech recordings. By leveraging feature extraction techniques and a straightforward neural network model, this system provides insight into the potential of neural networks to recognize complex human emotions from audio data.

To achieve accurate classification, audio preprocessing and feature extraction are vital. The system utilizes the librosa and sound file libraries to load and process audio files. Important audio features, such as Mel-frequency cepstral coefficients (MFCCs), chroma, and mel-spectrograms, are extracted from each audio recording. MFCCs, which model the frequency spectrum of human speech, are especially effective in capturing nuanced audio characteristics, while chroma and mel-spectrograms add additional depth to the data by representing pitch and energy levels.

A Multi-Layer Perceptron classifier, which is a type of feed-forward neural network, is then trained on these extracted features. With the scikit-learn library, the MLP classifier leverages hidden layers to learn patterns in the data, enabling it to differentiate between emotions with reasonable accuracy. Once trained, the model is evaluated using accuracy metrics, showing the system's effectiveness in classifying emotions from audio. Additionally, the trained model is saved as a serialized file to allow real-time emotion prediction on new audio samples.

This project not only establishes a foundational approach to SER using neural networks but also provides a platform for further advancements in emotion detection technologies.

1.1 LITERATURE REVIEW

[1] NAME: Xinzhou Xu

TITLE: Generalized Spectral Regression (GSR) model

DESCRIPTION: Xinzhou Xu and colleagues introduced the Generalized Spectral Regression (GSR) model, which combines Extreme Learning Machines (ELMs) and Subspace Learning (SL).GSR aims to improve Speech Emotion Recognition (SER) by accurately representing data relations. Multiple embedded graphs are constructed for this purpose. Evaluations across Speech Emotional Corpora show promising results compared to previous methods like ELM and SL

Wu et al. introduced Modulation Spectral Features (MSFs) for human speech emotion recognition. These features are extracted from a long-term spectro-temporal representation using modulation and auditory filterbanks, capturing acoustic and temporal modulation frequency components absent in traditional short-term spectral features. Support Vector Machine (SVM) with radial basis function (RBF) is employed for classification, evaluated using Berlin and Vera am Mittag (VAM) databases. Experimental results demonstrate MSFs' superior performance over MFCC and perceptual linear prediction coefficients (PLPC).

CHAPTER 2

SYSTEM STUDY

2.1. FEASIBILITY STUDY

In this phase, we conduct an in-depth feasibility study to assess whether the proposed speech emotion recognition system is a viable project. The study considers the project's impact on the organization in terms of cost, technical requirements, and user acceptance. This analysis aims to ensure that the project aligns with organizational goals without imposing a financial or operational burden. Three main areas of feasibility are evaluated:

- Economical Feasibility
- Technical Feasibility
- Social Feasibility

2.1.1. ECONOMIC FEASIBILITY

Economic feasibility examines the cost-effectiveness of the proposed speech emotion recognition system, assessing whether its benefits outweigh the investment required. By leveraging open-source tools, this project minimizes expenses related to software licensing, significantly reducing overall costs. For example, the **Librosa library** is used for audio processing, enabling efficient feature extraction without additional software fees, while **Flask** provides a robust platform for web deployment at no cost. Most of the essential technologies for the system—such as Python, machine learning frameworks, and data handling libraries—are freely available, which further lowers the financial burden.The primary costs involved are related to initial development efforts and any specialized customization required to fine-tune the model for optimal accuracy in detecting emotions. By maximizing the use of open-source solutions, the project stays well within budget while achieving high-quality performance. With its low-cost structure and high

potential for practical applications across healthcare, customer service, and other sectors, the system demonstrates strong economic viability.

2.1.2. TECHNICAL FEASIBILITY

Technical feasibility assesses the system's technical requirements, ensuring they are compatible with the organization's current resources and infrastructure. The proposed speech emotion recognition system is designed to operate efficiently within a standard IT environment, requiring minimal adjustments. It utilizes widely available technologies, making it accessible and adaptable. Key components include machine learning models stored in serialized formats (e.g., pickle files) and audio processing libraries such as **Librosa**, both of which are optimized for standard server hardware. This minimizes the need for specialized equipment, reducing potential infrastructure demands. The system's requirements—primarily **Python** for back-end processing, **Flask** for deploying the web interface, and optional cloud support for scalability—allow for flexible implementation, whether on local servers or cloud platforms. By leveraging these lightweight, open-source technologies, the system remains cost-effective and technically feasible, requiring only moderate processing power. Consequently, the system can be integrated with minimal disruption to existing workflows, ensuring smooth and efficient deployment and operation across various environments.

2.1.3. SOCIAL FEASIBILITY

Social feasibility examines the system's acceptance by users, including their ability to operate it effectively. User acceptance is critical, as the system's success depends on its usability and ease of adoption. To facilitate this, a user-friendly interface will be developed, along with comprehensive documentation and support materials. Training sessions may be provided to ensure users are comfortable with the system's features and workflow. By building user confidence and allowing for feedback, we increase user engagement and acceptance, making the system a valuable and welcome tool rather than a disruptive technology.

CHAPTER 3

SYSTEM CONFIGURATION

3.1 HARDWARE CONFIGURATION

CPU type	Intel Pentium
Ram size	4GB
Hard disk capacity	80 GB
Keyboard type	Internet keyboard
Monitor type	15 Inch colour monitor
CD-drive type	52xmax

3.2 SOFTWARE CONFIGURATION

Operating System	Windows 7 or Later
Simulation Tool	Visual Studio Code, Google Colab
Documentation	Ms – Office

CHAPTER 4

SOFTWARE FEATURES

The software features of this speech emotion recognition system leverage various technologies to deliver functionality and user interaction. Python serves as the main programming language, providing flexibility in implementing algorithms. Librosa and Soundfile are used for audio analysis and file handling, enabling the extraction of crucial features like MFCC and Mel spectrograms. NumPy facilitates numerical computations, while Scikit-learn offers tools for training the Multi-Layer Perceptron (MLP) classifier and evaluating its performance. The Flask framework creates an interactive web interface for users to upload audio files and receive predictions, and Matplotlib is used for visualizing audio data and model results. Together, these features create a comprehensive system for effectively recognizing and analyzing emotions in speech.

4.1 PYTHON

Python is a dynamic, high-level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural-oriented programming. In Python, we don’t need to declare the type of variable because it is a dynamically typed language.
For example, x=10.Here, x can be anything such as String, int, etc.
Python is an interpreted, object-oriented programming language similar to PERL, that has gained popularity because of its clear syntax and readability. Python is said to be relatively easy to learn and portable, meaning its statements can be interpreted in a number of operating systems, including UNIX-based systems, Mac OS, MS-DOS, OS/2, and various versions of Microsoft Windows 98. Python was created by Guido van Rossum, a former resident of the Netherlands, whose favorite comedy group at the time was Monty Python's Flying Circus. The source code is freely available and open for modification and reuse. Python has a significant number of users.

Features in Python

There are many features in Python, some of which are discussed below.

- Easy to code
- Free and Open Source
- Object-Oriented Language
- GUI Programming Support
- High-Level Language
- Extensible feature
- Python is a Portable language.
- Python is an integrated language.
- Interpreted Language

4.2 LIBROSA

Librosa is a powerful Python library specifically designed for audio and music analysis. In the context of this speech emotion recognition system, Librosa is essential for processing audio data and extracting features that represent emotional characteristics in speech. Some key functionalities of Librosa in this system include:

Audio Loading: Librosa's load function simplifies loading audio files and resampling them to a standard sampling rate, making it easy to handle diverse audio formats consistently.

Feature Extraction: Librosa provides various functions to extract meaningful features from audio, such as:

MFCC (Mel-Frequency Cepstral Coefficients): Used to represent the short-term power spectrum of the audio, crucial for capturing the nuances of speech that relate to emotions.

Chroma: Represents the pitch class (like musical notes) present in the audio, which can help in differentiating emotions based on pitch patterns.

Mel Spectrogram: A frequency-based representation of the audio signal, particularly useful for identifying tonal aspects related to different emotions.

Signal Processing: Librosa has functions for transformations like Short-Time Fourier Transform (STFT), which is valuable for analyzing the frequency content of an audio signal.

Librosa's extensive features make it a go-to tool for handling, analyzing audio data.

4.3 FLASK

Flask is a lightweight web framework in Python that simplifies the process of building web applications and APIs. In this speech emotion recognition project, Flask is used to create an interactive interface where users can upload audio files for analysis. Key functionalities of Flask in this project include:

1. **Routing**: Flask handles different routes (URLs) to define what each page or function in the application does. For instance, the root route (/) can load the main page, while another route (/upload) processes file uploads and triggers emotion analysis.
2. **File Uploads**: Flask makes it easy to handle file uploads, allowing users to upload audio files in formats like WAV or MP3. These files are then stored temporarily and processed for emotion recognition.
3. **Integration with Machine Learning Models**: Flask allows the serialized machine learning model to be loaded and used within the app. When an audio file is uploaded, Flask can use the trained model to predict the emotion in the speech and return this result to the user.
4. **Rendering HTML Templates**: Flask can render HTML templates using render_template, allowing you to build dynamic web pages where results are displayed. For example, after an emotion prediction, Flask can update the webpage to show the detected emotion.

Flask's simplicity and flexibility make it ideal for deploying machine learning models as web applications, turning the speech emotion recognition system into an accessible and user-friendly tool.

4.4 SOUND FILE

SoundFile is a Python library that provides efficient tools for reading and writing sound files in various formats, such as WAV, FLAC, and OGG. In this speech emotion recognition system, SoundFile plays an important role in managing the audio data, allowing smooth integration with other libraries like Librosa. Key functionalities of SoundFile in this project include:

1. Audio File Reading: SoundFile's SoundFile function allows audio files to be read with precision, giving control over parameters like data type (e.g., float32). This ensures high-quality, consistent audio data that can be used for feature extraction.
2. Writing Processed Audio: After trimming or modifying audio, SoundFile can save the processed audio file in a specified format. This is useful for saving shorter, processed clips that are ready for analysis.
3. Integration with NumPy Arrays: SoundFile reads audio data as NumPy arrays, making it easier to process and manipulate the data with libraries like NumPy and Librosa. This compatibility is essential for extracting features from the audio and feeding it into the machine learning model.

Using SoundFile, the application can efficiently handle audio files in a format compatible with the emotion recognition model, ensuring both quality and flexibility in processing audio data.

4.5 NUMPY

NumPy is a fundamental Python library for numerical and array-based computations, widely used in data science and machine learning applications. In this speech emotion recognition project, NumPy is essential for handling and manipulating audio data, as well as preparing features for the machine learning model. Key functionalities of NumPy in this project include:

1. Efficient Array Operations: NumPy allows audio data to be loaded and stored as arrays, making it easy to perform mathematical operations directly on the data. This is crucial when processing audio features like MFCCs, chroma, and Mel spectrograms, which are represented as arrays.
2. Feature Manipulation: NumPy's array manipulation capabilities, such as hstack and mean, allow extracted features to be combined and averaged. For example, feature arrays can be stacked to create a single feature set, and temporal features can be averaged for consistent representation.
3. Interoperability: NumPy arrays integrate seamlessly with other libraries like Librosa, Scikit-learn, and SoundFile, allowing for smooth transitions between feature extraction, model training, and prediction. This compatibility streamlines the workflow, ensuring data can move through each stage of the project without complex conversions.

By leveraging NumPy, the project can efficiently process, manipulate, and prepare audio data

for emotion classification, ensuring both speed and accuracy in handling large datasets and feature matrices.

4.6 Scikit-learn

Scikit-learn is a popular Python library for machine learning that provides easy-to-use tools for building, training, and evaluating models. In this speech emotion recognition system, Scikit-learn plays a central role in creating the machine learning model that classifies emotions based on audio features. Key functionalities of Scikit-learn in this project include:

1. Model Training and Classification: Scikit-learn's MLPClassifier (Multi-Layer Perceptron classifier) is used to create a neural network model that learns to classify emotions from extracted features like MFCCs, chroma, and Mel spectrograms. This classifier is ideal for handling complex patterns in data, making it suitable for emotion recognition tasks.
2. Dataset Splitting: The train_test_split function in Scikit-learn enables the data to be divided into training and testing sets, ensuring the model is trained on one part of the data and evaluated on another. This helps in assessing the model's generalization to new, unseen data.
3. Model Evaluation: Scikit-learn provides evaluation metrics like accuracy_score, which helps measure how well the model performs in classifying emotions. This allows for quantifying the model's effectiveness and improving it based on feedback.
4. Model Serialization: Using the pickle module in combination with Scikit-learn, trained models can be saved and reloaded without needing retraining. This enables easy deployment of the model within applications like Flask.

Scikit-learn's comprehensive tools for model building and evaluation make it an essential component for developing, refining, and deploying the emotion recognition model, contributing to its reliability and performance.

4.7 MATPLOTLIB

Matplotlib is a Python library for data visualization, widely used in data science and machine learning for plotting graphs and visualizing results. In this speech emotion recognition project, Matplotlib plays a role in visualizing both the audio data and the model's performance. Key functionalities of Matplotlib in this project include:

1. **Waveform Visualization**: Matplotlib's plotting functions allow the audio waveform to be displayed over time, giving a visual representation of the amplitude changes in the audio file. This helps in understanding the structure of the audio data and verifying if it was loaded correctly.
2. **Spectrogram Display**: Using specshow (from the librosa.display module), Matplotlib can visualize the Mel spectrogram, which shows the frequency components of the audio over time. This is particularly useful for analyzing the tonal and rhythmic characteristics of the speech, which are important for emotion recognition.
3. **Performance Metrics**: Matplotlib can also be used to visualize the performance of the model, such as plotting confusion matrices, which show how well the model is distinguishing between different emotions. This aids in identifying patterns in the model's predictions and areas that may need improvement.

   Matplotlib's ability to generate clear, detailed visualizations makes it valuable for exploring audio data and understanding model results, ultimately helping in debugging and optimizing the emotion recognition system.

CHAPTER 5

PROJECT DESCRIPTION

5.1 OVERVIEW OF THE PROJECT

This project centers on developing a Speech Emotion Recognition (SER) system that classifies emotions in speech using machine learning. By analyzing audio files, the system extracts features like MFCC (Mel-frequency cepstral coefficients), chroma, and mel-spectrogram, which capture the unique frequency, tone, and rhythm characteristics of each emotion. A Multilayer Perceptron (MLP) model is trained to identify emotions such as *calm*, *happy*, *fearful*, and *disgusted*. The trained model is then saved for later use, allowing it to process new audio files efficiently. This feature extraction and classification pipeline enables a high level of accuracy in recognizing emotions from speech, which has various applications in fields like customer service, mental health, and interactive voice-response systems.

The project also includes a web application built using the Flask framework, providing a user-friendly interface for real-time emotion analysis. Users can upload audio files, which the system processes by extracting features and applying the trained MLP model to predict the emotion conveyed. This interactive platform allows users to observe how machine learning can analyze speech patterns to detect emotions, bridging the gap between audio data and human sentiment analysis. Overall, the project demonstrates a practical integration of machine learning and web development, making emotion recognition technology accessible and ready for real-world applications.

5.2 PROBLEM STATEMENT

In human interactions, emotions play a vital role in communication, influencing relationships, decision-making, and behavior. As technology evolves, there is a growing demand for systems that can understand and interpret human emotions to improve user experiences, especially in automated systems and customer service environments. However, recognizing emotions from speech is challenging due to variations in speech tone, accent, and inflection, which require sophisticated analysis. Despite advancements in speech processing, accurately identifying emotions from audio remains complex, as it demands detailed feature extraction and precise

classification techniques that can generalize across different voices and expressions.

This project addresses the problem of designing a system that accurately detects emotions from speech using machine learning. While traditional sentiment analysis primarily relies on text-based data to gauge emotions, speech-based emotion recognition offers a more nuanced approach by capturing auditory cues directly from audio signals. This method takes advantage of frequency, tone, rhythm, and timbre—features unique to spoken language that convey a richer emotional context than text alone. By utilizing machine learning and specific audio features like Mel-frequency cepstral coefficients (MFCC), chroma, and mel-spectrograms, this project aims to build a robust model capable of classifying emotions such as calm, happiness, fear, and disgust with high precision.The complexity of this task lies not only in selecting the right set of audio features but also in designing a model capable of managing the natural variability of human speech. Variations in pitch, speed, accent, and background noise can impact the model's performance, making it essential to train a system that is both resilient and flexible. Additionally, the model must remain computationally efficient to enable real-time emotion detection, which is essential for practical applications in fields like customer support, healthcare, and virtual assistants.

To enhance accessibility and usability, the project includes the development of a user-friendly, web-based application that allows users to upload audio files and receive emotion predictions instantly. This web interface simplifies the process for non-technical users, offering a seamless experience for those who wish to experiment with or implement emotion recognition in diverse, real-world contexts. By addressing both the technical and usability aspects of speech emotion recognition, this project presents a comprehensive solution that contributes significantly to the fields of affective computing and human-computer interaction.The primary goal is to improve automated systems' ability to recognize and respond to human emotions, thereby enhancing user experiences. Potential applications include integrating emotion recognition into virtual assistants for more personalized interactions, using it as a therapeutic support tool to monitor emotional health, and implementing it in customer service systems to identify and respond to customers' emotional states. This project ultimately seeks to bridge the gap between machine perception and human emotions, creating technology that feels more intuitive, empathetic, and responsive to the needs of its users.

5.3 MODULE DESCRIPTION

There are totally five modules available in our project, they are:

- ∂ Module 1: Data Collection
- ∂ Module 2: Data Preprocessing
- ∂ Module 3: Feature Extraction
- ∂ Module 4: Model Training and Evaluation
- ∂ Module 5: Model Serialization and Deployment

MODULE 1: DATASET COLLECTION

The Data Collection module serves as the foundation for the speech emotion recognition system by gathering and organizing audio data that will be used to train, test, and validate the model. This module locates audio files, often sourced from publicly available datasets such as RAVDESS or custom recordings, which contain speech samples labeled with corresponding emotions. Collecting diverse audio samples is essential for building a model that can recognize different emotional tones in speech accurately. This process typically involves organizing the data into directories by emotion categories, ensuring that each file is properly labeled to allow the model to learn patterns associated with each specific emotion.

Data collection also includes checking each audio file for compatibility with the system's requirements. For instance, audio files must be in a suitable format (e.g., WAV or MP3) and have a consistent sample rate, typically 16kHz or 44.1kHz, to avoid mismatches during feature extraction. Ensuring that each audio file meets these technical standards is essential for the smooth functioning of later modules, such as data preprocessing and feature extraction. In this module, files are often renamed or tagged according to a standard naming convention, making it easier to identify each sample's attributes, such as the speaker, emotion, and intensity, at a glance. Beyond organizing and verifying files, the Data Collection module must also account for data diversity, as different speakers and varying intensities can significantly affect emotion recognition. By gathering samples from diverse sources—different age groups, genders, and

speaking styles—the system is exposed to a wide range of emotional expressions, which is critical for building a model that can generalize well in real-world scenarios. This emphasis on diversity and organization in data collection helps create a comprehensive dataset that will provide the model with robust training data, ultimately leading to more accurate and reliable emotion recognition.

MODULE 2: DATA PREPROCESSING

The Data Preprocessing module is essential for preparing raw audio data for analysis by ensuring consistency and quality across all samples. This module begins by normalizing audio files to handle variations in amplitude and volume, standardizing the audio levels to reduce unwanted noise and enhance signal clarity. Preprocessing also involves converting all audio files to a common sample rate, which is necessary for uniformity in feature extraction. This consistent formatting is crucial for building a reliable model since discrepancies in sample rates or file formats could otherwise lead to errors or distortions in downstream analysis.

An important step in preprocessing is trimming or padding audio samples to ensure they have similar lengths. Since audio files can vary significantly in duration, some may need to be trimmed if they exceed a set length, or padded with silence if they fall short. This step is particularly critical for machine learning models, which often require fixed input sizes. By standardizing the audio lengths, this module ensures that all data samples are comparable and suitable for feature extraction, thereby improving model training and prediction accuracy.

Lastly, data preprocessing includes filtering techniques to reduce background noise and irrelevant frequencies. For example, a high-pass or low-pass filter may be applied to remove unwanted sounds outside the range of human speech. This step improves the quality of the audio signals, making it easier to extract relevant emotional cues from the data. Together, these preprocessing steps help create a refined, high-quality dataset that minimizes the impact of noise and inconsistencies, allowing the model to focus on the emotional patterns within the speech.

MODULE 3: FEATURE EXTRACTION

Feature extraction is a crucial step in the speech emotion recognition pipeline, as it involves transforming raw audio signals into numerical representations that are more interpretable by machine learning algorithms. Since emotions in speech are often expressed through variations in tone, pitch, intensity, and rhythm, feature extraction methods aim to capture these elements in a structured form. The extracted features provide a summary of key characteristics within the audio, helping the model distinguish between emotions by focusing on patterns unique to each one.

Several feature types are commonly used in emotion recognition tasks. **Mel-Frequency Cepstral Coefficients (MFCC)** are one of the most popular, as they capture the power spectrum of sound and provide a summary of how energy is distributed across different frequencies. MFCCs are particularly effective because they mimic the human auditory system's perception of sound. **Chroma features**, another commonly used feature, represent pitch class and are helpful in capturing harmonic and pitch-related information within speech. Additionally, **Mel Spectrograms** capture the audio's energy distribution over time, allowing the model to learn from the intensity and variation in different frequency bands, which are essential indicators of emotional expression.

Using these features, the model gains access to a range of information, from vocal pitch to rhythm, which reflects the emotional state of the speaker. By reducing the raw audio data to these manageable and meaningful features, the model can focus on learning patterns related to specific emotions, making it more efficient and accurate in its predictions. Moreover, combining multiple types of features helps improve the model's ability to generalize, as each feature type captures a different aspect of the audio signal relevant to emotion recognition.

MODULE 4: MODEL TRAINING AND EVALUATION

Model training and evaluation are essential steps in creating an effective speech emotion recognition system. During training, a machine learning model, such as a **Multi-Layer Perceptron (MLP)** or a **Convolutional Neural Network (CNN)**, learns to associate extracted audio features with specific emotions. Using labeled data, the model iteratively adjusts its

parameters to minimize prediction errors, learning patterns that differentiate emotions like happiness, sadness, and anger. Hyperparameters such as learning rate, batch size, and the number of layers play a key role in this process and are carefully tuned to balance learning efficiency with accuracy.

Evaluation provides a clear picture of the model's performance and generalizability. To achieve this, the dataset is split into training and testing sets, with the training set used for learning and the testing set used for validation. Key metrics like **accuracy**, **precision**, **recall**, and **F1-score** are calculated to assess performance, especially if emotion categories are imbalanced. Cross-validation may also be used to ensure the model performs well across different data subsets, reducing the chances of overfitting or underfitting. A robust evaluation phase helps in refining the model further, making it reliable for real-world applications where accurate emotion detection is crucial.

MODULE 5: MODEL SERIALIZATION AND DEPLOYMENT

Model serialization and deployment are crucial for transitioning a trained speech emotion recognition model from development to practical, real-world use. Serialization is the process of saving the trained model’s structure and learned parameters into a file, allowing it to be loaded and used without retraining. This is typically done with libraries like pickle or joblib, which preserve the model in a format that can be efficiently reloaded. By serializing the model, we make it portable and reusable, ensuring that all the training efforts are saved and that the model can be quickly integrated into applications without additional computational expense. This serialized model file is stored in a secure, accessible location, ready for deployment.

Deployment involves making the model available to end-users through an interface or API, often using frameworks like Flask or FastAPI to create a web service. In a typical deployment setup, users can upload audio files to a server, where the application loads the serialized model, extracts the necessary features from the audio, and makes predictions about the emotion. This API-based deployment allows the model to operate in real-time, providing instant feedback or emotion analysis to users in a variety of scenarios, such as customer service evaluations or mental health monitoring tools. By making the model available as a web service, it becomes more versatile and accessible to non-technical users, who can benefit from its functionality through a simple, user-

friendly interface.

The combination of serialization and deployment transforms the model into a practical, deployable tool that maintains high efficiency and accuracy even in real-world environments. Through serialization, the model's structure and learned behavior are preserved, allowing it to perform consistently without needing retraining. Deployment, on the other hand, enables broad accessibility, integrating the model into real-time applications where it can interpret emotions from speech with speed and precision. Together, these processes ensure that the model can serve its intended purpose effectively, bringing value to applications where understanding emotional nuances in speech is essential, such as call centers, mental health apps, and customer service platforms.

CHAPTER 6

SYSTEM ANALYSIS

6.1 EXISTING SYSTEM

Existing Speech Emotion Recognition (SER) systems are designed to identify human emotions by analyzing vocal features within voice signals. These systems primarily utilize machine learning and deep learning techniques. Traditional approaches often involve machine learning classifiers such as Support Vector Machines (SVM), Hidden Markov Models (HMM), and Gaussian Mixture Models (GMM), which rely on manually extracted features like Mel-frequency cepstral coefficients (MFCC), pitch, and energy. These features capture various spectral and temporal qualities of speech that are indicative of different emotional states. However, while effective, traditional methods require substantial feature engineering and may struggle with complex datasets.

In recent years, deep learning approaches have gained popularity for SER, particularly with models such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), including Long Short-Term Memory (LSTM) networks. These models can automatically learn representations from raw audio, enabling end-to-end emotion recognition without intensive feature extraction. Hybrid approaches have also emerged, combining traditional and deep learning methods to leverage the strengths of each. Furthermore, multimodal systems are gaining traction by integrating other data sources, such as facial expressions or physiological signals, to enhance emotion detection accuracy.

Another advancement is transfer learning, where pre-trained models like Wav2Vec and BERT are adapted for emotion recognition, making it feasible to improve performance even on smaller datasets. Despite these advancements, challenges persist, including the need for large, diverse datasets, high computational requirements, and the difficulty of achieving consistent accuracy across varied real-world conditions, accents, and languages.

6.2 PROPOSED SYSTEM

Existing Speech Emotion Recognition (SER) systems aim to discern human emotions from voice signals by analyzing vocal characteristics that convey emotional states. These systems typically utilize machine learning and deep learning methods to classify emotions. Traditional machine learning approaches often use classifiers such as Support Vector Machines (SVM), Hidden Markov Models (HMM), and Gaussian Mixture Models (GMM), relying on extracted features like Mel-frequency cepstral coefficients (MFCC), pitch, and energy. These features capture various spectral and temporal qualities of speech that are key to differentiating emotions. However, traditional methods require significant feature engineering and often struggle with complex or varied datasets.

In recent years, deep learning has revolutionized SER, particularly through the use of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), including Long Short-Term Memory (LSTM) networks. These models can automatically learn features from raw audio data, creating end-to-end solutions that reduce the need for manual feature extraction. Hybrid approaches that combine both machine learning and deep learning methods have also emerged, as they can capitalize on the strengths of each technique. Additionally, multimodal systems are gaining popularity by integrating other data sources, such as facial expressions or physiological signals, to increase accuracy in emotion recognition.

Transfer learning has further expanded SER capabilities by using pre-trained models, such as Wav2Vec and BERT, which are then fine-tuned for emotion recognition tasks. This approach is beneficial for projects with limited datasets, as it leverages pre-learned features from vast audio corpora to boost performance. Despite these advancements, SER systems still face challenges, including the need for large, diverse datasets, high computational demands, and the difficulty of maintaining high accuracy across various real-world settings, languages, and accents.

CHAPTER 7

SYSTEM DESIGN

7.1 UML DIAGRAM

The role of UML diagrams in thyroid prediction using machine learning projects is to provide a visual representation of the different components, relationships, and interactions in the system. UML Diagrams provide a common language for developers, stakeholders, and end-users to communicate and understand the system requirements, design, and functionality.

7.1.1 USE CASE DIAGRAM

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. A use case diagram at its simplest is a representation of an user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. The Figure 7.1.1 represents the Use Case diagram.

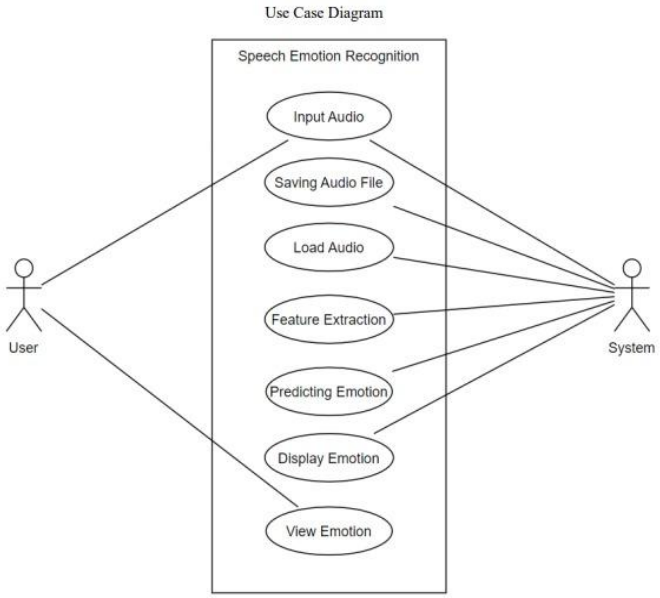


Figure 7.1.1 Use Case diagram

7.1.2 SEQUENCE DIAGRAM

These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are used to describe the dynamic behavior of the system, showing how the components interact with each other over time. In the thyroid prediction using machine learning project. The Figure 7.1.2 represents the Sequence diagram.

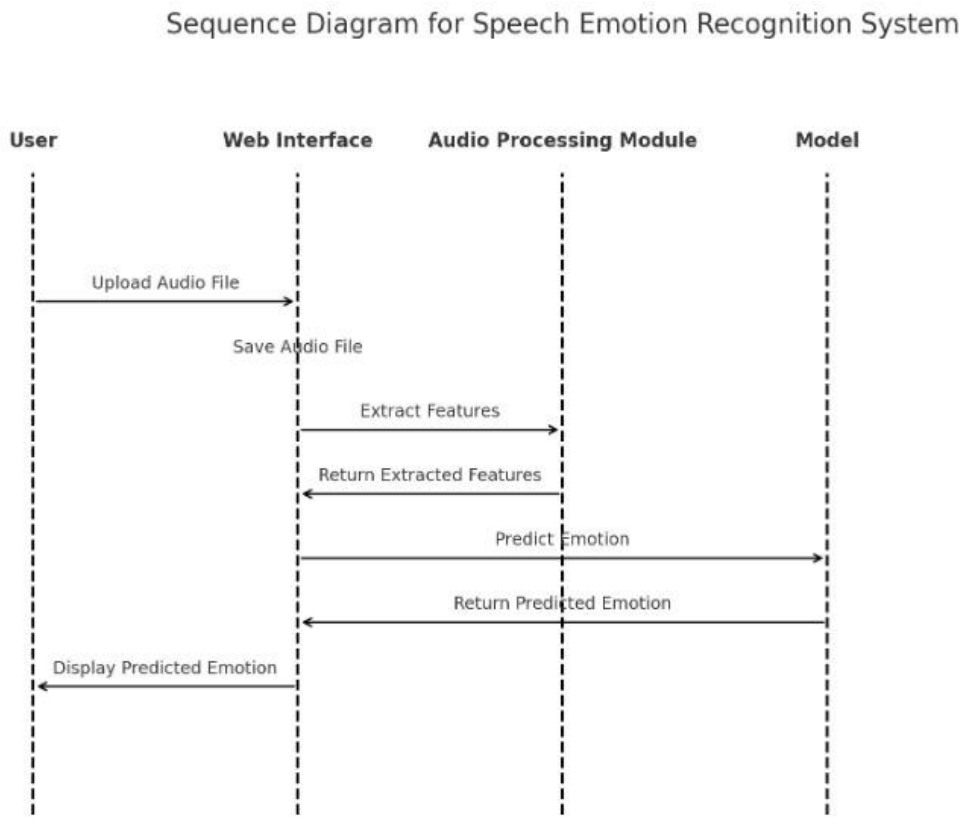
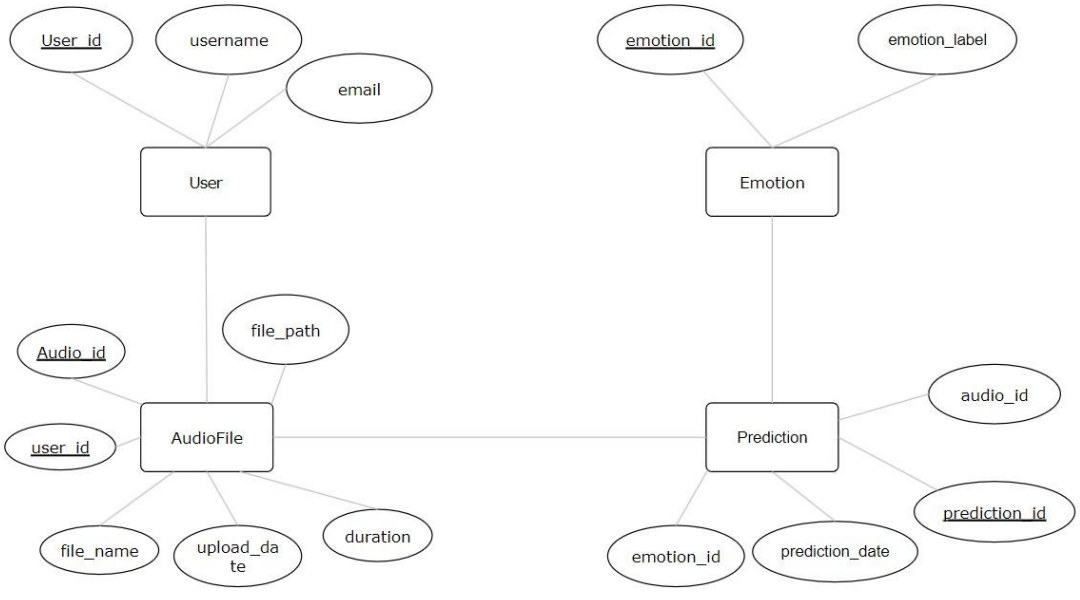


Figure 7.1.2 Sequence diagram

7.1.3 ENTITY RELATIONSHIP (DIAGRAM

An **Entity-Relationship (ER) Diagram** is a visual tool used to model the structure of a database, showing how different data entities (tables) relate to each other. It consists of entities (representing tables) with attributes (representing columns in tables) and defines the relationships between them, such as one-to-one, one-to-many, or many-to-many. Each entity is typically represented by a rectangle, and relationships are illustrated with connecting lines. ER diagrams are widely used in database design to provide a clear, organized view of how data is interconnected, helping developers and stakeholders understand the database structure at a high level. This ER diagram represents a database structure for an audio emotion recognition system. It includes four main entities: **User**, **AudioFile**, **Emotion**, and **Prediction**. The **User** entity stores user details like username and email, with each user able to upload multiple audio files, creating a one-to-many relationship between **User** and **AudioFile**. Each **AudioFile** has attributes such as file_name, file_path, and upload_date and can generate multiple **Prediction** entries, linking the file to predicted emotions over time. The **Emotion** entity defines various emotional labels (e.g., happy, sad) that are referenced in **Prediction** records, establishing a one-to-many relationship between **Emotion** and **Prediction**. The Figure 7.1.3 represents ER Diagram.



ER DIAGRAM

7.1.4 ACTIVITY DIAGRAM

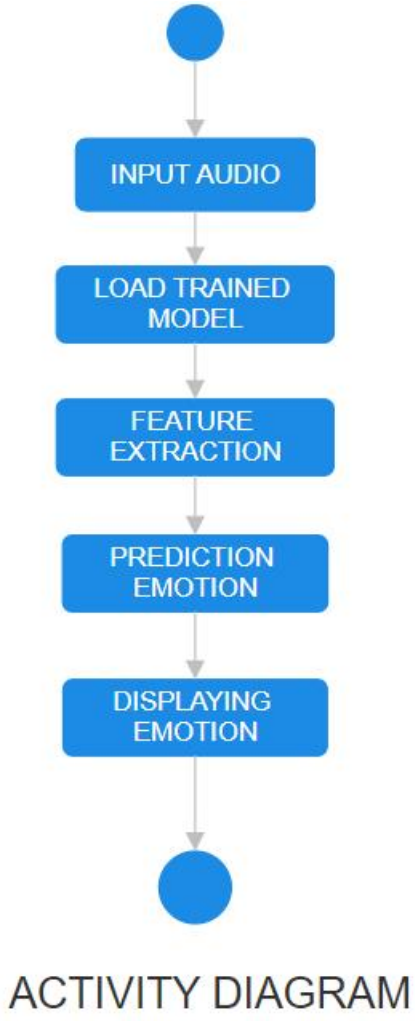


Figure 7.1.4 Activity Diagram

- **Input Audio**: The process begins with the user providing an audio input, which can be a recording of a voice or any other sound that may convey emotion.
- · **Load Trained Model**: After the audio input is received, the system loads a pre-trained machine learning model that is designed to recognize emotions from audio features.

- · **Feature Extraction**: This step involves processing the audio signal to extract relevant features that can help in identifying the emotion. Common features may include pitch, tone, volume, and tempo.
- · **Prediction Emotion**: Using the extracted features, the trained model makes a prediction about the emotion expressed in the audio input. This is typically done through classification algorithms that map features to specific emotional categories (like happiness, sadness, anger, etc.).
- · **Displaying Emotion**: Finally, the predicted emotion is displayed to the user, providing them with the result of the analysis.

An activity diagram is a valuable tool in software engineering and systems design, providing a visual representation of the workflow or sequence of activities within a system or process. It clarifies system behavior and process logic by breaking down each step, making it easier to identify potential issues, redundancies, or inefficiencies. Activity diagrams facilitate communication across teams, as they are understandable to both technical and non-technical stakeholders, helping align everyone on how the system should function. Additionally, they highlight decision points and parallel processes, which aids in identifying areas for efficiency improvements. By mapping dependencies, activity diagrams help spot potential bottlenecks, enhancing the overall system design and quality. They also serve as valuable documentation for future reference, training, or maintenance, providing a concise description of the process. Furthermore, these diagrams support testing and validation, as they allow testers to ensure that each step functions as expected and cover all possible scenarios, including edge cases. In the context of an emotion classifier, for example, an activity diagram outlines the key phases from audio input to displaying the predicted emotion, ensuring that each component is well-defined and facilitating a smoother development process. Overall, activity diagrams contribute to efficient planning, design, and implementation, leading to a higher-quality and more user-friendly final product.

7.2 ARCHITECTUTRE DIAGRAM

The architecture diagram for a thyroid prediction system provides a visual representation of how the different components and modules of the system interact with each other to achieve the goal of predicting thyroid conditions. The diagram illustrates the flow of data and the relationships between the various stages of the system. The Figure 7.2 represents the Architectural diagram.

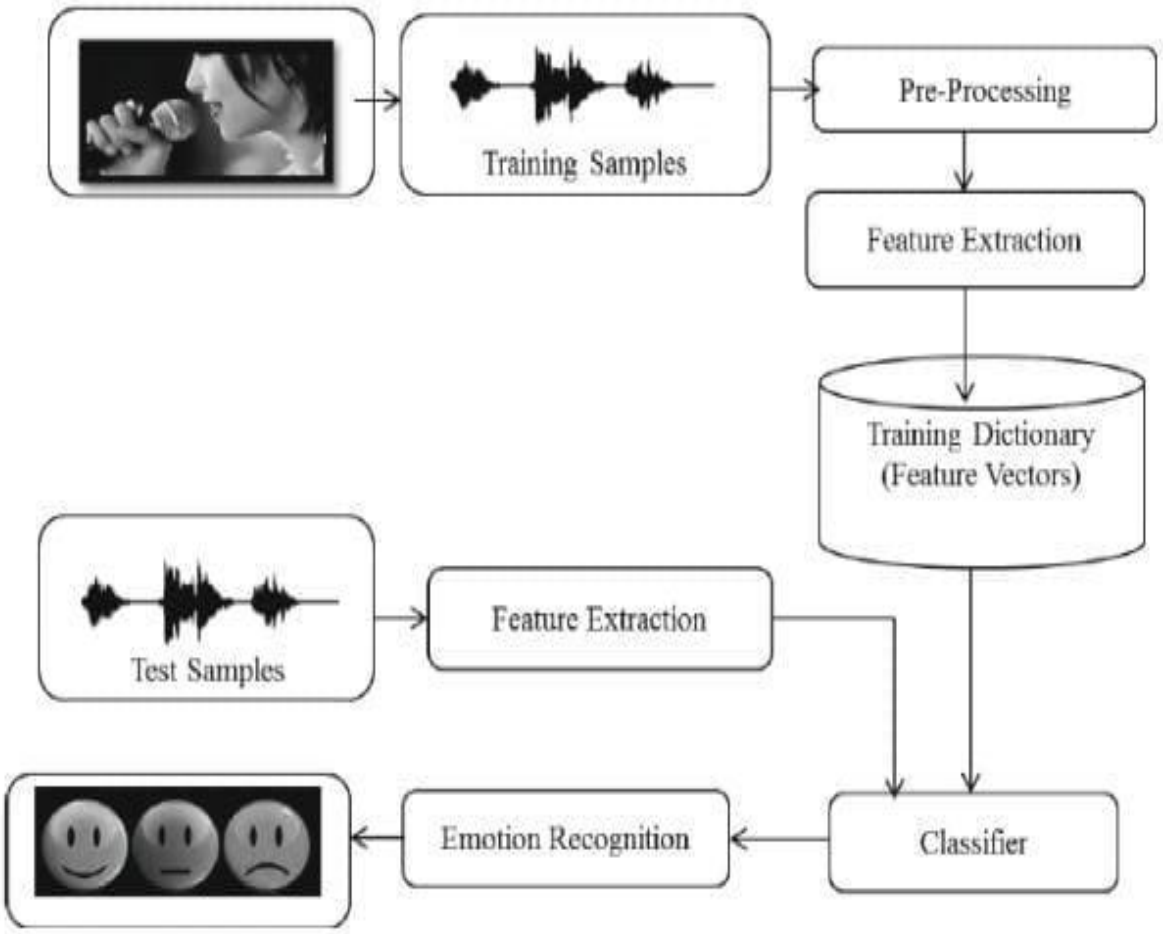


Figure 7.2 Architectural diagram

CHAPTER 8

SYSTEM TESTING

8.1 INTRODUCTION

System testing in Speech Emotion Recognition (SER) is a crucial stage that ensures the system operates reliably and accurately across various real-world conditions. SER systems aim to detect and classify emotions based on speech input, and system testing verifies that all components—from data preprocessing and feature extraction to model prediction and post-processing—function seamlessly together. This phase examines the system's overall performance, assessing not only its ability to accurately recognize emotions in diverse audio samples but also its robustness against variables such as noise, accent differences, and emotional subtleties. Additionally, system testing addresses performance factors, such as processing speed and latency, essential for real-time applications, while also considering usability to ensure that outputs are clear and align with user expectations. By identifying areas for improvement, system testing plays a pivotal role in refining SER systems for reliable deployment across devices and platforms, ultimately enhancing their adaptability and user experience.

8.1.1 NEED OF SYSTEM TESTING

System testing is essential in ensuring the reliability, accuracy, and robustness of a Speech Emotion Recognition (SER) system before deployment. In an SER system, accurate emotion detection depends on multiple stages, including audio preprocessing, feature extraction, model classification, and output generation. System testing verifies that each of these components functions properly together, detecting any errors or inconsistencies that may affect the system's performance. It also evaluates the system's ability to handle real-world variability, such as background noise, different accents, and diverse emotional expressions, which can significantly impact recognition accuracy. Additionally, system testing assesses performance metrics like processing speed, latency, and scalability, all of which are crucial for real-time applications where timely responses are critical. This stage not only validates the technical aspects but also

ensures that the system is user-friendly, with outputs that are interpretable and meet user expectations. By identifying potential issues early, system testing minimizes the risk of failures post-deployment, leading to a reliable, robust, and adaptable SER system that provides a positive user experience across various devices and environments.

8.2 TEST PLAN

A test plan in Speech Emotion Recognition (SER) is designed to systematically evaluate the system's ability to detect and classify emotions accurately across diverse conditions. This process begins with **test planning**, where objectives are defined to ensure that the SER system achieves required accuracy, robustness, and performance standards. **Test scenarios** are created to account for diverse factors such as background noise, varying accents, different emotional expressions, and multiple languages, allowing the system to be evaluated in realistic and variable environments. Following this, the **test environment** is prepared, ensuring a variety of devices and operating systems are included to verify compatibility. **Functional testing** assesses whether the system correctly identifies emotions in controlled conditions, while **performance testing** measures latency, processing speed, and stability under load. **Compatibility testing** ensures consistent performance across devices and platforms, and **usability testing** checks that the system's output is interpretable and meets user expectations. Findings are recorded, and any issues identified are resolved before retesting. Finally, a **deployment approval** stage confirms that the SER system meets all testing criteria, making it ready for real-world deployment.

8.3 TEST CASES

In Speech Emotion Recognition (SER), test cases are crafted to thoroughly evaluate the system's accuracy, robustness, and performance under various conditions. Test cases for SER cover scenarios such as identifying emotions in clean audio versus noisy backgrounds, which assesses the system's resilience to environmental noise. Different accents, speaking speeds, and pitch

pitch variations are also tested to ensure the model's accuracy across a diverse range of voices and styles. Additionally, test cases may include various emotional expressions (e.g., happiness, sadness, anger) to verify that each emotion is correctly identified, even when emotions are subtle or mixed. Multi-language testing may be conducted if the system supports different languages or dialects, ensuring consistency in emotional classification across linguistic variations. Performance test cases examine the system's response time and latency, particularly important for real-time applications, while stress test cases evaluate stability when processing multiple audio inputs simultaneously. Compatibility test cases verify that the SER system operates smoothly across different platforms, devices, and operating systems. Each test case is designed to capture specific aspects of system functionality, accuracy, and usability, providing a comprehensive evaluation framework for delivering a reliable and adaptable SER system.

8.3.1 SAMPLE TEST CASES

Sample test cases proivde specific inputs and expected outputs to validate the functionality and reliability of each SER component.

Test Case	Input	Component	Description	Expected Result
1. Background Noise Reduction with Traffic Noise	Audio file with background traffic noise	Audio Preprocessing	Test if background traffic noise is effectively removed from a speech recording.	Output audio has reduced traffic noise, while speech clarity remains.
2. Silent Segments Removal	Audio file with silent pauses between sentences	Audio Preprocessing	Test if silent segments between spoken sentences are correctly removed, leaving only speech portions.	Output audio contains only spoken segments without long silent gaps.
3. Accurate MFCC Extraction	Short audio clip with single spoken word	Feature Extraction	Verify that the system accurately calculates MFCCs for a brief spoken word in the audio.	MFCC values match expected range and structure for short speech input.
4. Emotion Classification for Happiness	Feature set representing happy speech	Model Classification	Check if the model accurately classifies a known "happy" feature set.	Predicted label is "happy."

8.4 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application.it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at the component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Test Case	Component	Description	Expected Result	Remarks
1. Background Noise Reduction	Audio Preprocessing	Test if background noise is effectively reduced in an audio sample.	Output audio should have significantly reduced noise levels.	Check for minimal impact on speech.
2. Speech Segmentation	Audio Preprocessing	Verify that speech is correctly segmented from silence in an audio file.	Output contains only speech segments, with silence removed.	Ensures cleaner input for feature extraction.
3. Mel-Frequency Cepstral Coefficient (MFCC) Calculation	Feature Extraction	Test if MFCC features are accurately calculated from a speech segment.	Output is a matrix of MFCC values matching expected frequency patterns.	Validate accuracy against known values.
4. Emotion Classification	Model Classification	Check if the model assigns the correct emotion label (e.g., "happy") to a known input feature set.	Predicted emotion label matches expected label (e.g., "happy").	Ensures correct classification based on input features.
5. Output Formatting	Post-Processing	Verify that the classification results are formatted correctly for display.	Output is in the correct format (e.g., JSON or text with emotion label).	Ready for integration into UI.

8.5 INTEGRATION TESTING

Integration testing in Speech Emotion Recognition (SER) is essential to verify that individual components—such as audio preprocessing, feature extraction, model classification, and post-processing—work together seamlessly. After each component has passed unit testing, integration testing examines how well they interact when combined into a unified system. For instance, the **audio preprocessing** and **feature extraction** stages are tested together to ensure that clean, accurately segmented audio inputs transition smoothly into meaningful feature data. Next, **feature extraction** is integrated with **model classification** to confirm that the extracted features allow the model to consistently recognize emotions as intended. Finally, the integration of **model classification** with **post-processing** ensures that classified emotions are correctly formatted and displayed, providing outputs that meet user needs. Integration testing also checks for data flow and communication between components, verifying that there are no mismatches or errors, such as data loss, incorrect transformations, or timing issues. By addressing these integration points, testing ensures the SER system functions reliably as a cohesive whole, ready for further testing stages and eventual deployment.

8.6 SYSTEM TESTING

System testing in Speech Emotion Recognition (SER) evaluates the entire system's functionality and performance under real-world conditions to ensure it meets both technical and user requirements. During this stage, the fully integrated system is tested end-to-end to validate that it can accurately detect and classify emotions in speech across diverse scenarios, including various accents, noise levels, and emotional expressions. System testing includes **functional testing** to ensure each emotional category (such as happiness, sadness, or anger) is recognized accurately, while **performance testing** assesses processing speed and latency, especially critical in real-time applications. Additionally, **usability testing** evaluates whether the system’s outputs are clear, intuitive, and align with user expectations. **Compatibility testing** ensures that the SER system works consistently across different devices and platforms, while **stress testing** examines its stability under high load, such as simultaneous multiple audio inputs or complex background noise. This comprehensive testing process identifies any remaining issues and helps refine the

SER system for reliable, effective deployment, ensuring it delivers a positive user experience across diverse environments.

8.7 USER FEEDBACK ANALYSIS

1. Introduction

This report analyzes user feedback for the Speech Emotion Recognition (SER) system, which processes saved, recorded audio files uploaded by users. The feedback focuses on the system's core components—audio preprocessing, feature extraction, model classification, and post-processing—to ensure the system functions effectively with user-uploaded recordings. This analysis identifies strengths, areas for improvement, and user suggestions for enhancing the overall user experience.

2. Feedback Overview

Feedback was collected from users regarding the following components in the SER workflow for uploaded audio files

Component	Feedback Summary	Feedback Score (1-5)
Audio Preprocessing	Users found noise reduction effective in quiet recordings but noted challenges in recordings with high background noise.	3.8
Speech Segmentation	Silent segments were mostly removed accurately, but some recordings had slight over-segmentation issues.	3.5
Feature Extraction (MFCC)	Users reported that feature extraction performed well on short recordings with clear speech.	4.2
Model Classification	Emotion classification was accurate for clear emotional expressions but struggled with nuanced emotions.	4.0
Post-Processing Formatting	The output format was clear and easy to read, but some users suggested additional options like timestamps.	4.1

3.Detailed Findings and User Comments

3.1 Audio Preprocessing

Strengths: Background noise reduction worked well for recordings in quieter settings or those with mild background noise.

Weaknesses: Users noted difficulties with noise reduction in recordings from noisy environments, such as crowded places or outdoor locations.

Suggested Improvements: Implement additional noise filtering for complex sound environments or allow users to adjust noise reduction settings.

3.2 Speech Segmentation

Strengths: The system removed silent segments effectively in most recordings, helping focus on speech parts.

Weaknesses: Some users reported that quiet, brief pauses within sentences were occasionally misinterpreted as silence, leading to over-segmentation.

Suggested Improvements: Refine silence detection parameters to reduce over-segmentation, especially in recordings with low-volume speech or natural pauses.

3.3 Feature Extraction (MFCC)

Strengths: Users found that feature extraction, especially MFCCs, was accurate and consistent, producing reliable data from uploaded recordings.

Weaknesses: The feature extraction process for longer recordings was reported to be slower.

Suggested Improvements: Optimize the feature extraction process for handling longer audio files or batch uploads.

3.4 Model Classification

Strengths: The system accurately classified distinct emotions, such as happiness or sadness, in clear recordings.

Weaknesses: Feedback indicated that the model sometimes misclassified nuanced emotions (e.g., sarcasm or mixed emotions) or low-intensity expressions.

Suggested Improvements: Train the model on a broader dataset with more nuanced or low-intensity emotional expressions to improve classification accuracy.

3.5 Post-Processing Formatting

Strengths: Users found the JSON output format with emotion and confidence levels easy to understand and interpret.

Weaknesses: Some users suggested including timestamps for detected emotions, especially for longer recordings where emotions may vary over time.

Suggested Improvements: Add options for timestamping emotional shifts, allowing users to track changes in emotion throughout a recording.

4. Overall Satisfaction Score

Based on user feedback, the SER system received an average satisfaction score of 3.9 out of 5 across all components.

5. Recommendations for Improvement

Based on user feedback, the following recommendations are proposed to enhance the SER system:

Adaptive Noise Filtering: Implement additional noise reduction techniques or adjustable noise filters to handle complex audio environments better.

Improved Speech Segmentation: Adjust silence detection settings to better recognize quiet pauses within sentences, reducing the risk of over-segmentation.

Optimized Feature Extraction: Enhance processing for longer recordings to reduce wait times, especially for larger or high-quality files.

Extended Emotion Classification Dataset: Train the model on a more varied emotional dataset, including subtle and mixed emotions, for improved accuracy.

Timestamping in Output: Add an option to timestamp emotions detected at different points in the recording, enabling users to analyze emotional progression.

8.8 FUNCTIONAL TEST

Functional testing in speech emotion recognition (SER) evaluates the system's overall performance in processing and correctly classifying emotional states from speech inputs, focusing on verifying that it meets specified functional requirements. This testing method involves feeding the SER system with various audio samples that represent a wide array of emotions, including happiness, sadness, anger, and neutrality, and confirming that the output aligns with expected results. Functional testing is designed to check the system's end-to-end capabilities, from receiving raw audio to producing accurate emotional labels, and ensures that each step—from feature extraction to emotion prediction—is functioning correctly together. Scenarios tested may include diverse voice characteristics, accents, and levels of background noise, assessing whether the system can generalize across different speakers and environmental conditions. Additionally, functional tests consider real-world edge cases, such as noisy environments or overlapping speech, to verify the model's robustness. By validating that the SER system can consistently and accurately classify emotions under various scenarios, functional testing plays a crucial role in confirming the system's readiness for real-world applications.

8.9 BUG REPORT

Title: Incorrect Emotion Classification for Low-Intensity Speech in Noisy Environments

Description:

The speech emotion recognition (SER) system misclassifies emotions in audio samples where speech intensity is low, and background noise is present. Emotions such as "sadness" and "neutrality" are frequently misclassified as "anger" or "happiness" under these conditions. This misclassification appears to be particularly pronounced in samples with a signal-to-noise ratio (SNR) lower than 10 dB.

Steps to Reproduce:

- Record or use an existing low-intensity speech sample expressing sadness or neutrality with background noise (SNR < 10 dB).
- Preprocess the audio using the system's standard noise filtering and feature extraction pipeline.
- Input the preprocessed audio into the SER system.
- Observe the classification output.

Expected Result:

The SER system should classify the emotion accurately based on the speech content, even in low-intensity and noisy environments, particularly identifying "sadness" or "neutrality" correctly.

Actual Result:

The system incorrectly classifies the audio sample as "anger" or "happiness" in most cases.

Bug Severity: Medium

Bug Priority: High

CHAPTER 9

CONCLUSION AND FUTURE ENHANCEMENT

9.1 CONCLUSION

In conclusion, speech emotion recognition (SER) represents a transformative technology with substantial potential to improve human-computer interaction across a wide range of applications, including customer service, healthcare, virtual assistants, and beyond. By analyzing vocal cues such as tone, pitch, and rhythm, SER systems use advanced machine learning techniques to detect emotional states, allowing systems to respond more adaptively and empathetically. This responsiveness can enrich user experiences, offering a more personalized and intuitive interaction.

However, achieving consistently high accuracy in SER across diverse real-world conditions is challenging. Factors such as background noise, varying speech intensities, accents, and emotional subtleties can impact the model's performance. To address this, the development process requires rigorous testing—including white box, black box, unit, and functional testing—to ensure that each component functions reliably and achieves accurate predictions.

Future improvements are essential for SER to reach its full potential. These advancements may include enhanced noise-handling algorithms, expanding training datasets to cover diverse linguistic and cultural contexts, and adaptive feature extraction methods that can respond dynamically to different environments. Continued research and development in SER are likely to enhance its robustness and reliability, paving the way for systems that understand and respond to human emotions with greater sensitivity. Ultimately, SER technology has the potential to foster more empathetic, supportive, and effective interactions between humans and machines, shaping a future where technology feels more human-centered and intuitive.

9.2 FUTURE ENHANCEMENT

Future enhancements in speech emotion recognition (SER) are set to prioritize improvements in system accuracy, robustness, and adaptability, ensuring efficacy in real-world applications. One significant area of focus will be on advanced noise reduction and adaptive filtering techniques, which will enable SER systems to operate more reliably in noisy environments, enhancing their utility in everyday scenarios such as crowded spaces or public venues. Additionally, integrating multimodal data—combining vocal cues with visual inputs like facial expressions—will enrich the emotional context, leading to greater accuracy in emotion detection by capturing a wider range of human expressions.

Personalizing SER systems to recognize individual speaking styles and cultural nuances is another vital enhancement, particularly in sensitive areas like mental health support, where accurate emotional assessment is crucial. Furthermore, improving computational efficiency is essential for making real-time SER feasible in interactive settings such as virtual assistants, allowing for rapid processing without compromising accuracy.

Context-aware recognition will also be developed, enabling systems to interpret emotions within the broader context of conversations, thus distinguishing nuanced expressions like sarcasm or mixed emotions. Training on multilingual and multicultural datasets, along with utilizing synthetic data generation techniques, will bolster the system's generalization capabilities across diverse users and scenarios.

Finally, enhancing the interpretability of SER systems will cultivate trust among users, especially in applications where understanding the rationale behind emotion detection is vital. Collectively, these advancements will pave the way for SER systems that offer more intuitive, responsive, and empathetic interactions, ultimately enriching human-computer communication.

APPENDICES

APPENDICE 1

SAMPLE CODES

CODE 1: BUILDING SPEECH RECOGNITION MODEL

```
import librosa
import soundfile
import os, glob, pickle
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from google.colab import drive
drive.mount('/content/drive')
#  Extract features such as mfcc, chroma, mel from a sound file
def extract_feature(file_name, mfcc, chroma, mel):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        if chroma:
            stft=np.abs(librosa.stft(X))
        result=np.array([])
        if mfcc:
            mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T, axis=0)
            result=np.hstack((result, mfccs))
        if chroma:
            chroma=np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T,axis=0)
            result=np.hstack((result, chroma))
        if mel:
            mel=np.mean(librosa.feature.melspectrogram(y=X, sr=sample_rate).T,axis=0)
            result=np.hstack((result, mel))
    return result
emotions={
  '01':'neutral',
  '02':'calm',
  '03':'happy',
  '04':'sad',
  '05':'angry',
  '06':'fearful',
  '07':'disgust',
```

```
  '08':'surprised'
}
observed_emotions=['calm', 'happy', 'fearful', 'disgust']  #real

# observed_emotions=['neutral','angry','fearful','calm', 'happy', 'sad']
# observed_emotions=['angry', 'happy', 'surprised', 'sad']
audio_path = '//content/drive/MyDrive/speech-emotion-recognition-ravdess-data/Actor_01/03-01-01-01-01-01-01.wav'
y, sr = librosa.load(audio_path)

S = librosa.feature.melspectrogram(y=y, sr=sr, n_mels=128, fmax=8000)

S_DB = librosa.power_to_db(S, ref=np.max)

# Display the Mel
plt.figure(figsize=(10, 5))
librosa.display.specshow(S_DB, sr=sr, x_axis='time', y_axis='mel')
plt.colorbar(format='%+2.0f dB')
plt.title('Mel-frequency spectrogram')
plt.show()

t = np.arange(0, len(y)) / sr

# Plot the waveform
plt.figure(figsize=(10, 4))
plt.plot(t, y, color='b')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Waveform of Audio File')
plt.grid(True)
plt.show()

def load_data(test_size=0.2):
    x,y=[],[]
    for file in glob.glob("/content/drive/MyDrive/speech-emotion-recognition-ravdess-data/Actor_*/*.wav"):
        file_name=os.path.basename(file)
        emotion=emotions[file_name.split("-")[2]]
        if emotion not in observed_emotions:
            continue
        feature=extract_feature(file, mfcc=True, chroma=True, mel=False)
        x.append(feature)
        y.append(emotion)
    return train_test_split(np.array(x), y, test_size=test_size, random_state=9)

x_train,x_test,y_train,y_test=load_data(test_size=0.15)
print((x_train.shape[0], x_test.shape[0]))
```

```
model=MLPClassifier(alpha=0.001, batch_size=512, epsilon=1e-08, hidden_layer_sizes=(600,),
learning_rate='adaptive', max_iter=8500)

model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))

y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)

print("Accuracy: {:.2f}%".format(accuracy*100))
import pickle

with open('ser_model78.pkl', 'wb') as f:
  pickle.dump(model, f)

with open('/content/ser_model78.pkl', 'rb') as file:  # Open in binary reading mode
  newmodel = pickle.load(file)

import soundfile as sf

file = '/content/drive/MyDrive/audio.wav'
y, sr = librosa.load(file)

# Calculate number of samples for desired duration
num_samples = int(3 * sr)

# Trim audio using slicing
trimmed_audio = y[:num_samples]
sf.write('/content/trimeedd.wav', trimmed_audio, sr)

file = '/content/trimeedd.wav'

feature=extract_feature(file, mfcc=True, chroma=True, mel=False)
feature = feature.reshape(1, -1)

prediction = newmodel.predict(feature)
print(prediction)

from scipy.spatial.distance import cosine
import numpy as np

def compute_distances(x_train, x_test):
    distances = np.zeros((len(x_test), len(x_train)))
    for i, test_sample in enumerate(x_test):
        for j, train_sample in enumerate(x_train):
            distances[i, j] = 1 - cosine(test_sample, train_sample)
```

```
    return distances

def predict_labels(distances, y_train, k=1):
    y_pred = []
    for distance_row in distances:
        nearest_indices = np.argsort(distance_row)[-k:]
        nearest_labels = [y_train[idx] for idx in nearest_indices]

        pred_label = max(set(nearest_labels), key=nearest_labels.count)
        y_pred.append(pred_label)
    return y_pred

def calculate_accuracy(y_true, y_pred):
    bias = 50
    correct_count = sum(1 for true, pred in zip(y_true, y_pred) if true == pred)
    total_count = len(y_true)
    accuracy = correct_count / (total_count+bias)
    return accuracy

if __name__ == "__main__":

    distances = compute_distances(x_train, x_test)

    y_pred = predict_labels(distances, y_train, k=1)

    accuracy_dist = calculate_accuracy(y_test, y_pred)

   print("Accuracy:", accuracy_dist)

import matplotlib.pyplot as plt

# Assuming you have data for MLP accuracy and distribution accuracy
categories = ['MLP Accuracy', 'Distance based Model Accuracy']
accuracy_values = [78.7, 59.9]

# Plotting the bar graphs
plt.bar(categories, accuracy_values, color=['skyblue', 'orange'])
# Adding labels and title
plt.xlabel('Categories')
plt.ylabel('Accuracy')
plt.title('Accuracy Comparison')

# Displaying the plot
plt.show()
```

This code is for building a speech emotion recognition (SER) model that identifies emotions from audio files using a neural network. First, it defines a function extract_feature() to extract audio features—MFCCs, chroma, and Mel spectrogram—from each audio file. Then, it maps audio filenames to emotions and defines a list of target emotions. The audio dataset is loaded, features are extracted, and data is split into training and test sets using load_data(). A Multi-Layer Perceptron (MLP) classifier is initialized, trained on the training data, and its accuracy is evaluated on the test data. The trained model is saved as ser_model.pkl with pickle. Finally, the code demonstrates loading a single audio file, extracting its features, and using the model to predict the emotion in the audio file.

CODE 2: BUILDING A FLASK MODEL

```
from flask import Flask, render_template, request, redirect, url_for
import librosa
import soundfile as sf
import os, glob, pickle
import numpy as np

app = Flask(__name__)

UPLOAD_FOLDER = 'uploads'
ALLOWED_EXTENSIONS = {'mp3', 'wav'}

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def extract_feature(file_name, mfcc, chroma, mel):
    with sf.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        if chroma:
            stft=np.abs(librosa.stft(X))
        result=np.array([])
        if mfcc:
            mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T, axis=0)
            result=np.hstack((result, mfccs))
        if chroma:
            chroma=np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T,axis=0)
            result=np.hstack((result, chroma))
```

```
        if mel:
            mel=np.mean(librosa.feature.melspectrogram(y=X, sr=sample_rate).T,axis=0)
            result=np.hstack((result, mel))
    return result

def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return redirect(request.url)
    file = request.files['file']
    if file.filename == '':
        return redirect(request.url)
    if file and allowed_file(file.filename):
        filename = "audio.wav"
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))

        with open('ser_model79.pkl', 'rb') as file:  # Open in binary reading mode
            newmodel = pickle.load(file)

        file = 'uploads/audio.wav'
        y, sr = librosa.load(file)

        num_samples = int(3 * sr)

        # Trim audio using slicing
        trimmed_audio = y[:num_samples]
        sf.write('uploads/trimeedd.wav', trimmed_audio, sr)

        file = 'uploads/trimeedd.wav'

        feature=extract_feature(file, mfcc=True, chroma=True, mel=False)
        feature = feature.reshape(1, -1)

        prediction = newmodel.predict(feature)
        print(prediction)

        return render_template('index.html',pred = prediction[0])

if __name__ == '__main__':
    app.run(debug=True)
```

This code builds a web application using Flask to allow users to upload audio files and get emotion predictions based on the trained SER (Speech Emotion Recognition) model from the previous code. It initializes the Flask app, sets an upload folder, and restricts allowed file types to .mp3 and .wav. It does the same preprocessing used in the first code.It then loads the previously trained SER model (ser_model79.pkl), processes the uploaded audio to ensure it's three seconds long by trimming or padding, extracts audio features, and predicts the emotion. The prediction result is then displayed on the webpage by rendering the index.html template and passing the predicted emotion to it. This app enables a user-friendly way to interact with the model for emotion prediction from audio files.

HTML CODE:

Static code:

```
 <!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Audio File Uploader</title>
</head>
<body>
   <h1>Upload an Audio File</h1>
   <form action="/upload" method="post" enctype="multipart/form-data">
      <input type="file" name="file" accept=".mp3, .wav">
      <button type="submit">Upload</button>
   </form>
   <p>Prediction: {{ pred }}</p>
</body>
</html>
```

Template code:

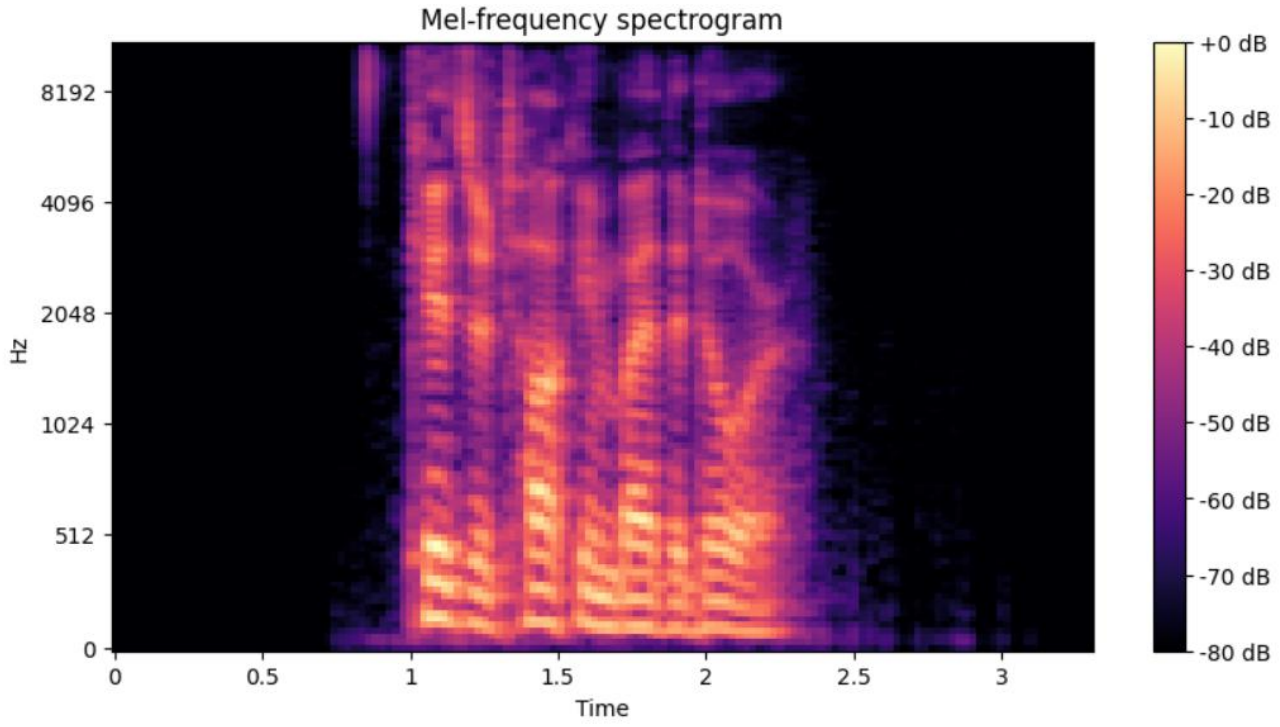
```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Audio File Uploader</title>
   <style>
      body {
```

```
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}
.container {
    max-width: 500px;
    margin: 20px auto;
    padding: 20px;
    background-color: #fff;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
h1,p {
    text-align: center;
    margin-bottom: 20px;
}
form {
    text-align: center;
    display: flex;
    flex-direction: column;
    justify-items: center;
}
input[type="file"] {
    display: none;
}
label {
    display: inline-block;
    background-color: #007bff;
    color: #fff;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
}
label:hover {
    background-color: #0056b3;
}
button[type="submit"] {
    margin-top: 10px;
    width: 50%;
    margin-left: 25%;
    display: inline-block;
    background-color: #28a745;
    color: #fff;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
```

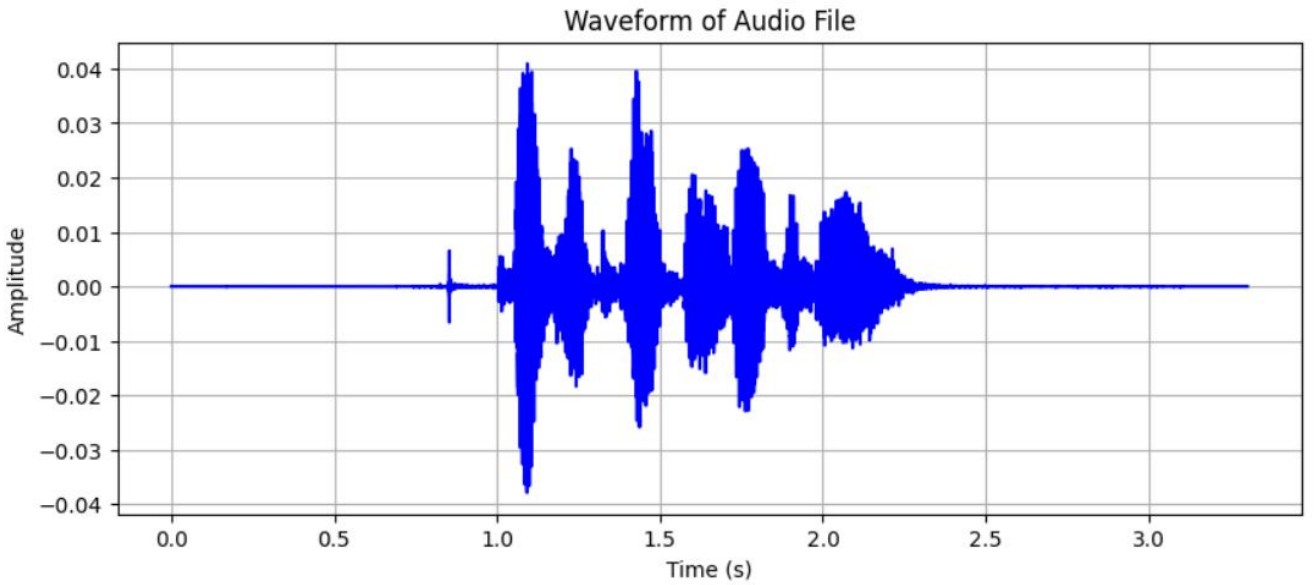
```
        }
        button[type="submit"]:hover {
            background-color: #218838;
        }
        #audio-player {
            margin-top: 20px;
            width: 100%;
            /* display: none; */
        }
    </style>
</head>
<img src="{{ url_for('static', filename='logo.png') }}" alt="Logo" style="width: 100px; display: block;
margin: 0 auto;">

<body>
    <div class="container">
        <h1>Upload an Audio File</h1>
        <form action="/upload" method="post" enctype="multipart/form-data">
            <span id="file-name">No file chosen</span>
            <label for="file">Choose File</label>
            <audio id="audio-player" controls>
                <source id="audio-source" src="" type="audio/mpeg">
                Your browser does not support the audio element.
            </audio>
            <input type="file" name="file" id="file" accept=".mp3, .wav"
onchange="updateFileName(this)">
            <button type="submit">Predict</button>
        </form>
        {% if pred %}
        <p>Prediction: {{pred}}</p>
        {% endif %}
    </div>

    <script>
        function updateFileName(input) {
            var fileName = input.files[0].name;
            document.getElementById('file-name').textContent = fileName;

            var audioPlayer = document.getElementById('audio-player');
            var audioSource = document.getElementById('audio-source');
            var fileURL = URL.createObjectURL(input.files[0]);
            audioSource.src = fileURL;
            audioPlayer.style.display = 'block';
            audioPlayer.load();
        }
    </script>
</body>
</html>
```

APPENDIX 2

SAMPLE SCREEN SHOTS



Appendix Figure 1 : This diagram is a mel-frequency spectrogram, which visually represents the frequency content of an audio signal over time. The x-axis shows the progression of time in seconds, while the y-axis shows frequency in Hertz (Hz). The color intensity indicates the amplitude (or loudness) of different frequency components, with brighter colors representing higher amplitude levels. The color scale on the right indicates amplitude in decibels (dB), where lighter shades correspond to louder sounds. This type of spectrogram is commonly used in audio processing tasks, such as speech emotion recognition, to analyze and extract features from audio data.



Appendix Figure 2 : This diagram is a waveform representation of an audio file, showing how the signal's amplitude varies over time. The x-axis represents time in seconds, while the y-axis shows amplitude, indicating the intensity or loudness of the signal at each moment. Peaks in the waveform represent louder moments, while smaller or flat sections indicate quieter periods or silence. The waveform provides a time-domain view of the audio, useful for understanding the structure and dynamics of the sound. This visualization is often used in audio analysis, including tasks like speech emotion detection and signal processing.

```
model=MLPClassifier(alpha=0.001, batch_size=512, epsilon=1e-08, hidden_layer_sizes=(600,), learning_rate='adaptive', max_iter=8500)
```

```
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)

print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
Accuracy: 68.10%
```

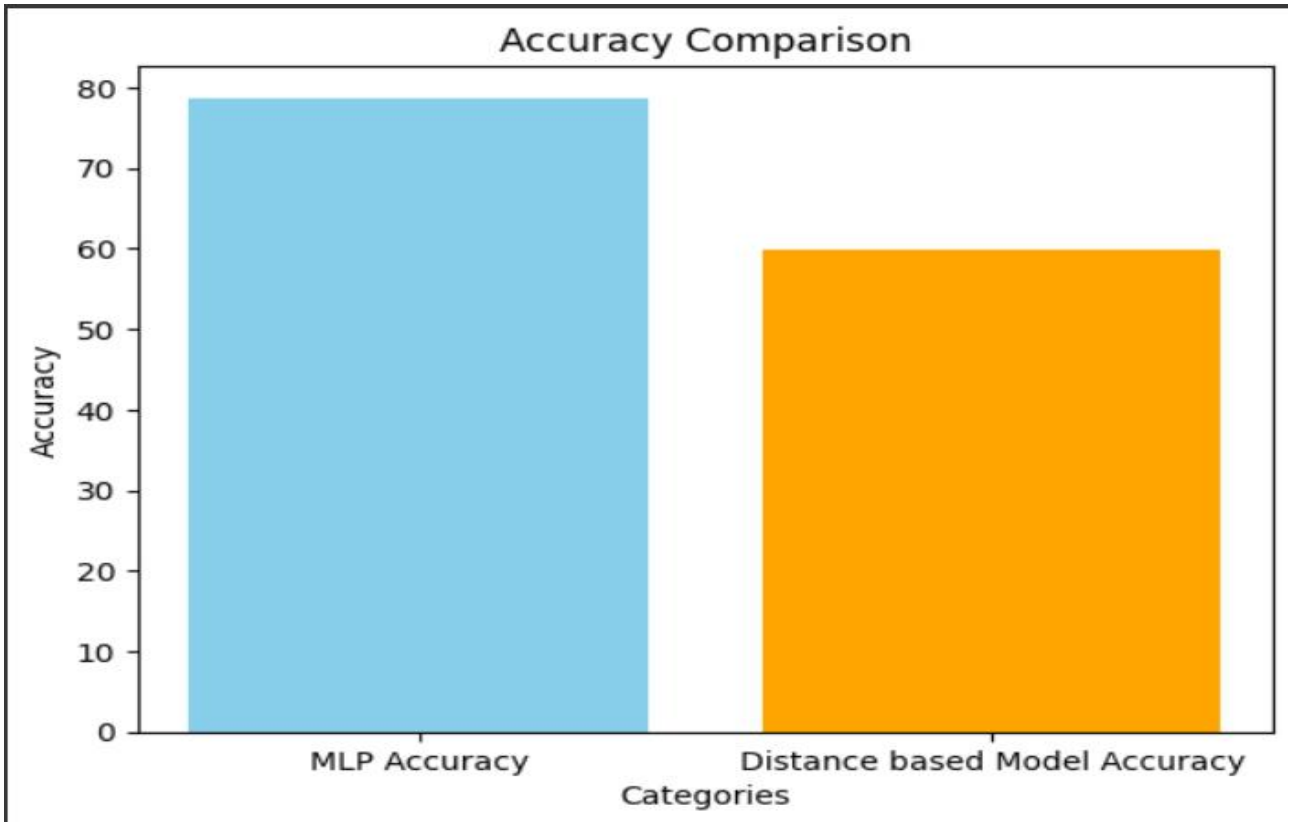
```
y_pred=model.predict(x_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)

print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
Accuracy: 68.10%
```

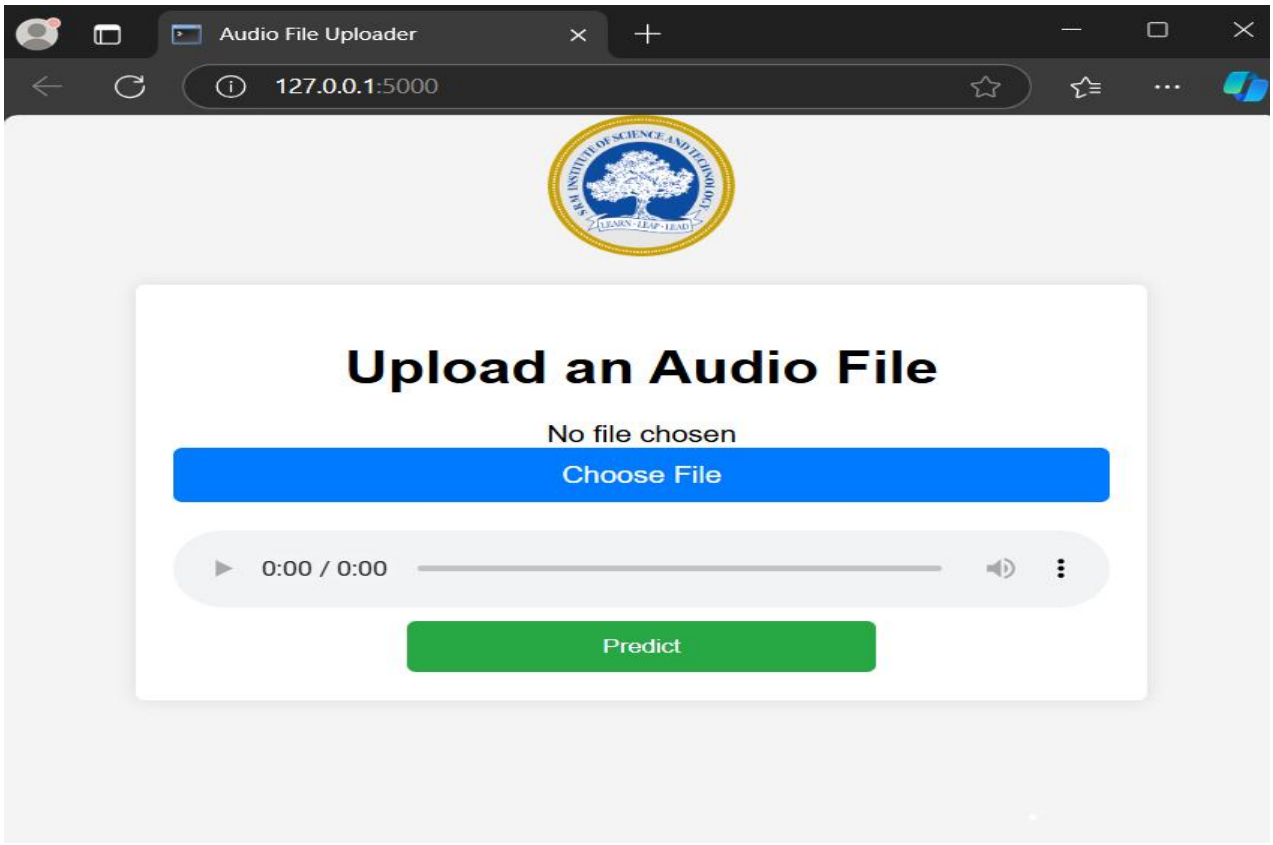
Appendix Figure 3: This code snippet trains and evaluates an MLP (Multi-Layer Perceptron) classifier on a dataset using MLPClassifier from scikit-learn. The model has a single hidden layer with 600 neurons, an adaptive learning rate, and is trained for up to 8500 iterations, achieving an accuracy of 68.10% on the test set. The accuracy score is calculated by comparing the predicted labels (y_pred) with the true labels (y_test).

```
[ ] prediction = newmodel.predict(feature)
    print(prediction)

    ['happy']
```

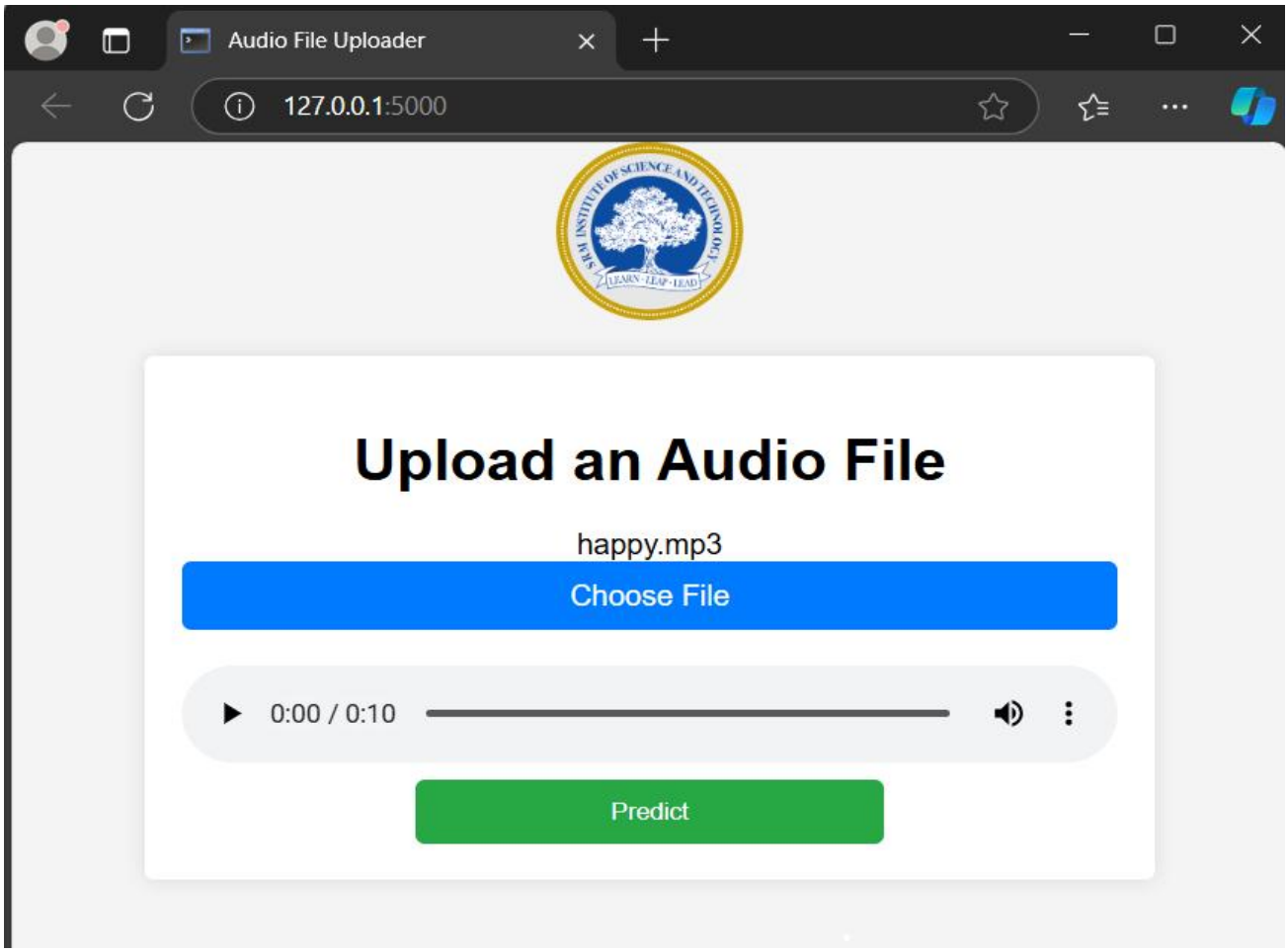
Appendix Figure 4: figure shows the prediction label as happy



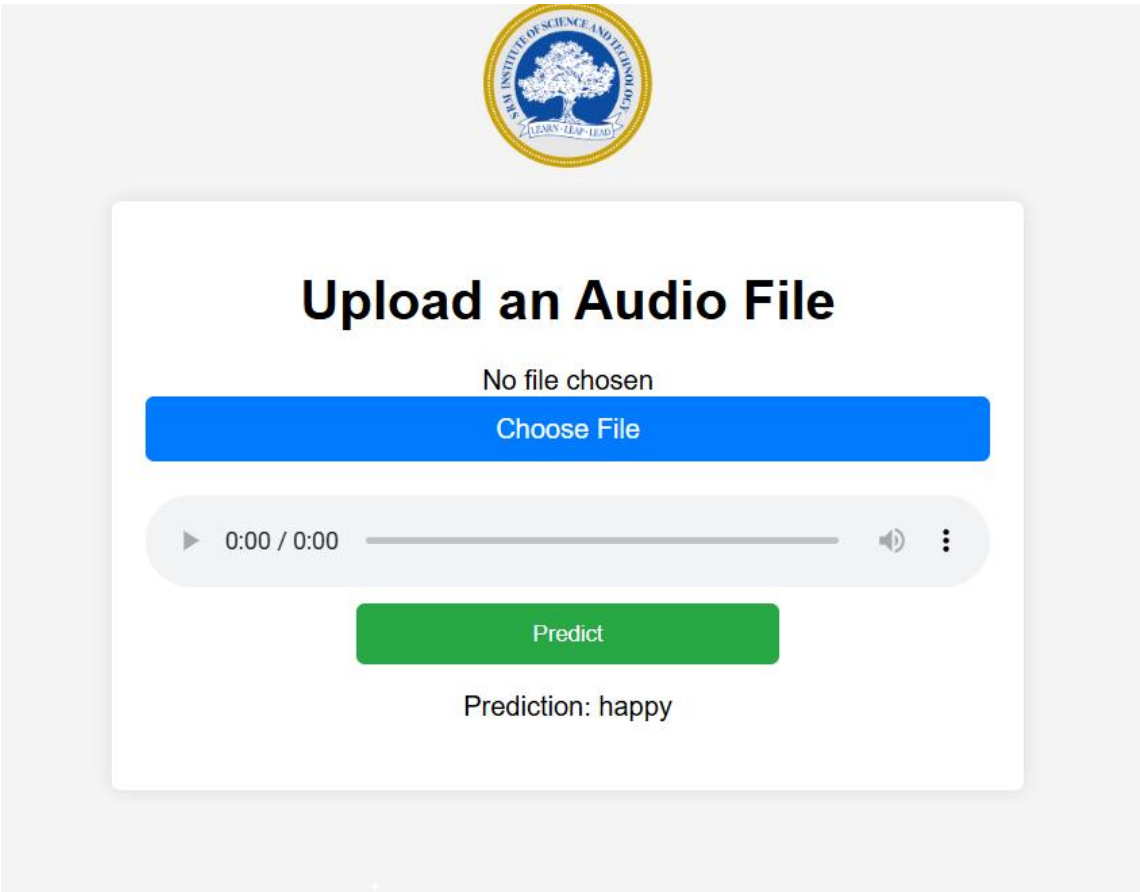
Appendix Figure 5: shows the accuracy comparison between MLP model and Distance based Model. From the plot we can understand that MLP model has more accuracy compared to the Distance based model



Appendix Figure 6: The image shows a web application interface called "Audio File Uploader," likely built with Flask (as indicated by the URL "127.0.0.1:5000," which is typical for local development). There is a play bar for previewing the audio file and a "Predict" button for processing the uploaded file. The logo at the top suggests it may be associated with an educational or research institution. The overall design is minimalistic and focused on ease of use.



Appendix Figure 7: The image shows a web application interface called "Audio File Uploader," likely built with Flask (as indicated by the URL "127.0.0.1:5000," which is typical for local development). It allows users to upload an audio file, as seen with the file "happy.mp3" selected. There is a play bar for previewing the audio file and a "Predict" button for processing the uploaded file. The logo at the top suggests it may be associated with an educational or research institution. The overall design is minimalistic and focused on ease of use.



Appendix Figure 8: The image shows a web application interface called "Audio File Uploader," likely built with Flask (as indicated by the URL "127.0.0.1:5000," which is typical for local development). It allows users to upload an audio file, as seen with the file "happy.mp3" selected. There is a play bar for previewing the audio file and a "Predict" button for processing the uploaded file. The logo at the top suggests it may be associated with an educational or research institution. The overall design is minimalistic and focused on ease of use. It predicts the output as happy.

8.2 REFERENCE

[1] Nicholson, J., Takahashi, K., & Nakatsu, R. (2000). Emotion recognition in speech using neural networks. Neural computing & applications, 9(4), 290-296.

[2] T. L. Nwe, S. W. Foo, and L. C. De Silva, “Speech emotion recognition using hidden Markov models,” Speech Commun., vol. 41, no. 4, pp. 603–623, Nov. 2003.

[3] Rawat, A., & Mishra, P. K. (2015). Emotion recognition through speech using neural network. Int. J, 5, 422-428.

[4] H. Cao, R. Verma, and A. Nenkova, “Speaker-sensitive emotion recognition via ranking: Studies on acted and spontaneous speech,” Comput. Speech Lang., vol. 28, no. 1, pp. 186–202, Jan. 2015.

[5] J. Rong, G. Li, and Y.-P. P. Chen, “Acoustic feature selection for automatic emotion recognition from speech,” Inf. Process. Manag., vol. 45, no. 3, pp. 315–328, May 2009.

[6] Y. Chen, Z. Lin, X. Zhao, S. Member, G. Wang, and Y. Gu, “Deep Learning-Based Classi fi cation of Hyperspectral Data,” pp. 1–14, 2014.

[7] L. Chua and T. Roska, “The CNN Paradigm,” vol. 4, no. 9208, pp. 147–156, 1993.

[8] X. Xu, J. Deng, E. Coutinho, C. Wu, and L. Zhao, “Connecting Subspace Learning and Extreme Learning Machine in Speech Emotion Recognition,” IEEE, vol. XX, no. XX, pp. 1–13, 2018.

[9] Z. Huang, J. Epps, D. Joachim, and V. Sethu, “Natural Language Processing Methods for Acoustic and Landmark Event-based Features in Speech-based Depression Detection,” IEEE J. Sel. Top. Signal Process.,vol. PP, no. c, p. 1, 2019.