A DEEP LEARNING APPROACH FOR SPEECH EMOTION RECOGNITION USING ML

A Project Report submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR.

in Partial Fulfillment of the Requirements for the Award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND SYSTEMS ENGINEERING

Submitted by

GOPALAM SAI KUMAR	**20121A2917**
GOBBURU JAYA VARDHAN RAJU	**21125A2904**
VEMAREDDY SUSHVITHA	**20121A2956**
GUNDEPOGULA CHAITHANYA	**20212A2919**

Under the Guidance of

Mr. B. Venkata Sivaiah, M. Tech.

Assistant Professor

Department of Computer Science and Systems Engineering

SREE VIDYANIKETHAN ENGINEERING COLLEGE (AUTONOMOUS)

Sree Sainath Nagar, Tirupati – 517 102

(2023 – 2024)

SREE VIDYANIKETHAN ENGINEERING COLLEGE

(AUTONOMOUS)

Sree Sainath Nagar, Tirupati

Department of Computer Science and Systems Engineering

CERTIFICATE

This is to certify that the project report entitled

"**A DEEP LEARNING APPROACH FOR SPEECH EMOTION RECOGNITION USING ML**"

is Bonafide work done by

GOPALAM SAI KUMAR	**20121A2917**
GOBBURU JAYA VARDHAN RAJU	**21125A2904**
VEMAREDDY SUSHVITHA	**20121A2956**
GUNDEPOGULA CHAITHANYA	**20212A2919**

In the Department of Computer Science and Systems Engineering, and submitted to Jawaharlal Nehru Technological University Anantapur, Ananthapuramu in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Systems Engineering during the academic year 2023-2024. This work has been carried out under my supervision. The results of this mini project work have not been submitted to any university for the award of any degree or diploma.

Guide: ***Head:***

Mr. B. Venkata Sivaiah
Assistant Professor
Dept. of CSE (DS)

Dr. K. Ramani
Professor & Head
Dept. of CSSE

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We hereby declare that this project report titled **"A DEEP LEARNING APPROACH FOR SPEECH EMOTION RECOGNITION USING ML"** is a genuine work carried out by us, in **B. Tech *(Computer Science and Systems Engineering)*** degree course of **Jawaharlal Nehru Technological University Anantapur** and has not been submitted to any other course or University for the award of any degree by us.

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signature of the students

1. G Sai Kumar

2. G Jaya vardhan Raju

3. V Sushvitha

4. G Chaithanya

ACKNOWLEDGEMENTS

We are extremely thankful to our beloved Chairman and founder **Dr. M. Mohan Babu,** Padma Shri awardee who took a keen interest and encouraged us in every effort throughoutthis course.

We owe our gratitude to **Dr. B. M. Satish**, Principal, Sree Vidyanikethan Engineering College (Autonomous) for permitting us to use the facilities available to accomplish the project successfully.

We express our heartfelt thanks to **Dr. K. Ramani**, Professor, and Head of the Departmentof Information Technology, for her kind attention and valuable guidance to us throughout this course.

We are thankful to our Project Coordinator **Mr. P. Yogendra Prasad,** Assistant Professor of CSSE for his valuable support and guidance throughout the project work.

We are extremely thankful to our Project Supervisor **Mr. B. Venkata Sivaiah,** Assistant Professor of CSSE took a keen interest and encouraged us in every effort throughout this project.

We also thank all the teaching and non-teaching staff of the CSSE Department for their cooperation.

ABSTRACT

Emotion recognition in audio data is a critical aspect of various fields like psychology, medicine, and human-computer interaction. This project explores the effectiveness of machine learning algorithms for emotion identification using datasets covering a wide range of emotions. Feature extraction techniques, including spectrogram analysis and Mel-frequency cepstral coefficients (MFCCs), are employed to capture emotional cues effectively. Additionally, data augmentation techniques are utilized to enhance dataset diversity and robustness. Long short-term memory (LSTM) networks, decision trees, and convolutional neural networks (CNNs) are investigated as models for emotion categorization. Decision trees offer straightforward classification, LSTMs capture temporal correlations, and CNNs excel at extracting features from audio signals. Notably, CNNs demonstrate superior performance in emotion categorization compared to Decision Trees and LSTMs. This research provides valuable insights into the performance of different machine learning models in audio-based emotion recognition, considering both feature extraction and data augmentation. These findings are crucial for developing reliable emotion detection systems for emotional computing, human-robot interaction, and mental health assessment.

Keywords *— Convolutional Neural Networks (CNN), Emotion Recognition, LSTM Networks, Machine Learning Algorithms.*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

SER	Speech Emotion Recognition
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
DT	Decision Tree
MFCC	Mel-Frequency Cepstral Coefficients

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 Introduction to the topic:

In the dynamic landscape of human-computer interaction and affective computing, Speech Emotion Recognition (SER) emerges as a pivotal frontier, bridging the gap between human expression and technological advancements. The ability to comprehend and decode emotions conveyed through spoken language not only enriches our understanding of human behavior but also fuels innovations in diverse fields such as sentiment analysis, mental health monitoring, and interactive virtual assistants.

This project embarks on a comprehensive exploration of SER, driven by a quest to contribute significant insights and advancements to this domain. Central to our approach is the implementation of robust feature extraction processes designed to capture the intricate emotional nuances inherent in speech. Techniques such as spectrogram analysis, Mel-frequency cepstral coefficients (MFCCs), and feature extraction are leveraged to extract salient features that encapsulate the richness of emotional expression in spoken language.

These extracted features serve as the foundational input to a suite of machine learning models encompassing Convolutional Neural Networks (CNNs), Decision Trees, and Long Short-Term Memory (LSTM) networks. Through a rigorous regimen of model training, evaluation, and hyperparameter tuning, our methodology is meticulously crafted to optimize model performance, thereby enhancing the system's ability to accurately recognize and categorize emotions conveyed through speech.

Furthermore, our project is committed to advancing dataset size and employing sophisticated data preprocessing techniques such as data augmentation. By harnessing the power of larger and more diverse datasets along with advanced preprocessing methodologies, our goal is to develop highly robust and effective SER systems. These systems are not merely tools for emotion recognition but catalysts for fostering more empathetic and intelligent human-computer interactions, ultimately contributing to the evolution of a more emotionally intelligent technological landscape.

1.2 Overview of SER

This provides an insightful overview of the SER process, highlighting its significance in extracting crucial characteristics from spectrograms or Mel-frequency cepstral coefficients (MFCCs) generated by Librosa. These extracted features capture subtle nuances in vocal tone, pitch, and strength. CNNs, renowned for their proficiency in image analysis, prove instrumental in categorizing speech across a broad spectrum of emotional states, including happiness, sadness, and anger. The ultimate goal is to enable machines to comprehend and differentiate various emotions expressed in spoken language, thereby unlocking potential applications for emotionally intelligent technology. This introduction sets the stage for exploring the fusion of Librosa, CNNs, and SER in unraveling the intricate fabric of human emotional expression through speech.

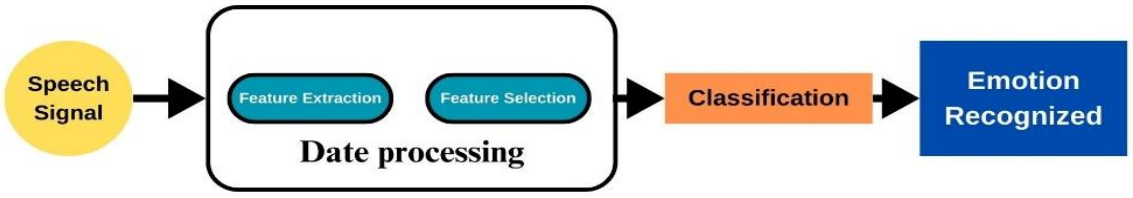

Fig: 1.1 Overview Process of speech emotion recognition

1.3 Problem Statement

The research tackles the difficulty of effectively discerning emotions conveyed through speech; a task known as Speech Emotion Recognition (SER). By harnessing the power of Machine Learning techniques and the Librosa library, the study aims to overcome current obstacles in accurately identifying emotions in spoken language. Its primary goal is to push the boundaries of emotional analysis within speech data, with the ultimate aim of improving applications such as human-computer interaction and healthcare. Through innovative approaches and sophisticated algorithms, the research seeks to enhance our understanding and interpretation of emotional cues in speech, paving the way for more advanced and empathetic technological solutions.

1.4 Motivation

The motivation behind this project stems from the transformative potential of Speech Emotion Recognition (SER) in enhancing human-computer interaction and understanding human emotions. By developing robust SER systems, we aim to improve user experiences, enable early detection of emotional distress, and gain valuable insights into customer sentiments. Leveraging

advancements in machine learning and large datasets, our goal is to contribute to the development of empathetic and intelligent systems that can accurately recognize and respond to emotions conveyed through spoken language.

1.5 Objectives of project work

- **Prediction of Emotion:** Machine learning models can be trained on audio datasets with different emotions to predict emotion in a random given audio sample.
- **Advance Preprocessing:** Develop and implement advanced data preprocessing techniques, including data augmentation, to enhance the diversity and robustness of the SER dataset
- **Improve Accuracy:** Improved accuracy can help in better prediction of the given audio file which can lead to higher understanding in psychology, medicine, and human-computer interaction.

CHAPTER 2

LITERATURE SURVEY

2. LITERATURE SURVEY

[1] Niharika S M, Soumya A: Speech Emotion Recognition using Machine Learning. In: 7th International Conference on Computation System and Information Technology for Sustainable Solutions. (2023)

This paper is majorly focused on utilization of Convolutional Neural Networks (CNNs) has proven effective in capturing crucial auditory properties for discerning emotions from voice data. However, an area that can significantly augment the model's performance lies in the incorporation of diverse preprocessing techniques. Among these, data augmentation stands out as a pivotal approach, enabling the model to learn from varied and robust training data, ultimately enhancing its ability to accurately interpret emotions. Techniques such as pitch shifting, time stretching, and the introduction of background noise can be employed to create augmented datasets that encompass a wide range of acoustic scenarios, thereby improving the model's robustness and generalization capabilities. Furthermore, complementing data augmentation with other preprocessing methods like feature selection can further refine the model's accuracy, interpretability, and resilience to variations in voice data. By synergistically integrating these preprocessing techniques, the CNN-based VER model can be elevated to achieve superior performance in understanding and categorizing emotions expressed through voice.

[2] Majd Saloumi et al: Speech Emotion Recognition using One-Dimensional Convolutional Neural Networks. In: 46th International Conference on Telecommunications and Signal Processing. (2023)

The development of spoken Emotion Recognition (SER) models has witnessed significant advancements, particularly with the utilization of 1D-CNN architectures and Mel-Frequency Cepstral Coefficients (MFCCs) tailored for short spoken recordings. A notable example is a recent SER model trained on the RAVDESS dataset, which achieved an impressive 83% accuracy through the strategic implementation of data augmentation techniques. However, despite the remarkable accuracy achieved by the aforementioned model, there exists untapped potential in expanding the dataset size. A larger and more diverse training dataset can provide the model with a richer and more comprehensive understanding of the nuances in speech patterns associated with different emotions. This expanded dataset can encompass a broader range of speakers, languages, accents, and emotional expressions, further refining the model's accuracy and adaptability across diverse contexts.

[3] K. Vamsi Krishna et al: Speech Emotion Recognition using ML(SVM). In: 6th International Conference on Computing Methodologies and Communication. (2022)

The investigation into Spoken Emotion Recognition (SER) has been a multifaceted endeavor, exploring various machine learning algorithms and audio characteristics to enhance emotion recognition accuracy. In one study, researchers delved into SER using Support Vector Machine (SVM), Multi-layer Perceptron (MLP), and a spectrum of audio features such as MFCC, MEL, and Chroma. However, a significant breakthrough emerged with the adoption of a Convolutional Neural Network (CNN) architecture, which revolutionized SER by focusing on data augmentation techniques and leveraging automated feature learning. The integration of CNNs in SER signifies a paradigm shift, as these architectures excel in learning complex patterns and dependencies directly from raw audio data. Unlike traditional machine learning algorithms that rely heavily on handcrafted features, CNNs autonomously extract hierarchical representations, enabling the model to discern emotions in speech signals with enhanced robustness and accuracy. This approach not only streamlines the feature engineering process but also empowers the model to adapt more effectively to variations in speech patterns and emotional expressions.

[4] Kim, S. H., & Kim, H. Y. "Emotion Recognition System Using Short-Term Monitoring of Physiological Signals." IEEE Transactions on Biomedical Engineering, 60(12), 3374-3383 (2013)

A physiological signal-based emotion recognition system is reported. The system was developed to operate as a user-independent system, based on physiological signal databases obtained from multiple subjects. The input signals were electrocardiogram, skin temperature variation and electrodermal activity, all of which were acquired without much discomfort from the body surface, and can reflect the influence of emotion on the autonomic nervous system. The system consisted of preprocessing, feature extraction and pattern classification stages. Preprocessing and feature extraction methods were devised so that emotion-specific characteristics could be extracted from short-segment signals. Although the features were carefully extracted, their distribution formed a classification problem, with large overlap among clusters and large variance within clusters. A support vector machine was adopted as a pattern classifier to resolve this difficulty. Correct-classification ratios for 50 subjects were 78.4% and 61.8%, for the recognition of three and four categories, respectively.

[5] Koolagudi, S. G., & Rao, K. S. "Emotion Recognition from Speech: A Review." International Journal of Speech Technology, 15(2), 99-117. (2012)

Emotion recognition from speech has emerged as an important research area in the recent past. In this regard, review of existing work on emotional speech processing is useful for carrying out further research. In this paper, the recent literature on speech emotion recognition has been presented considering the issues related to emotional speech corpora, different types of speech features and models used for recognition of emotions from speech. Thirty-two representative speech databases are reviewed in this work from point of view of their language, number of speakers, number of emotions, and purpose of collection. The issues related to emotional speech databases used in emotional speech recognition are also briefly discussed. Literature on different features used in the task of emotion recognition from speech is presented. The importance of choosing different classification models has been discussed along with the review. The important issues to be considered for further emotion recognition research in general and in specific to the Indian context have been highlighted where ever necessary.

CHAPTER 3

METHODOLOGY

3. METHODOLOGY

3.1 Proposed System

Here, we will delve into the technologies and critical information pertaining to understanding the proposed technology and its features for speech emotion recognition. The methodology involves employing pre-processing techniques such as data augmentation and utilizing a large number of datasets to achieve higher accuracy in the emotion recognition process.

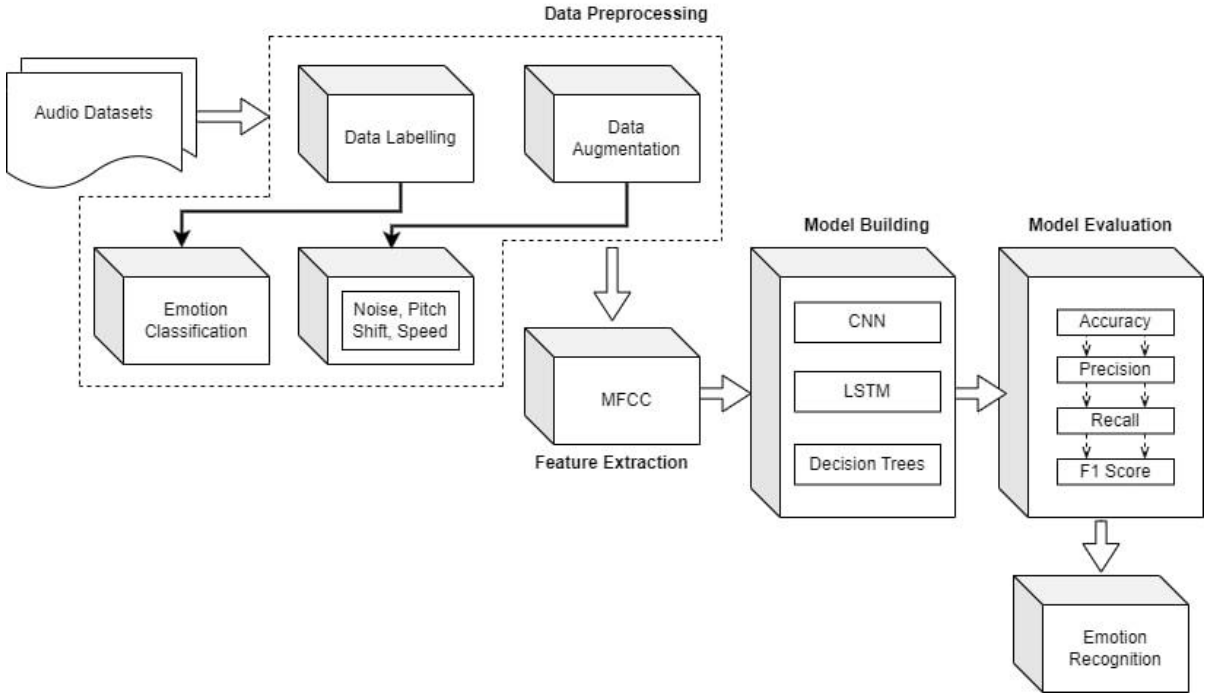

Fig: 3.1 Architecture of Speech Emotion Recognition

The envisioned system strives to propel Speech Emotion Recognition (SER) to new heights by integrating Convolutional Neural Networks (CNN) and the Librosa library, incorporating additional algorithms like Long Short-Term Memory (LSTM) and Decision Trees. While the accuracy of LSTM and Decision Tree algorithms falls below that of CNN, their inclusion adds versatility to the model. Utilizing CNNs on audio spectrograms and Librosa's feature extraction, the architecture integrates convolutional and pooling layers, optimizing the identification of emotional features. Data augmentation and transfer learning enhance model generality. Emphasizing Librosa's, Mel-Frequency Cepstral Coefficients (MFCCs), the system aspires to achieve state-of-the-art results, potentially revolutionizing human-computer interaction and healthcare applications. Despite varied accuracy, the amalgamation of CNN, LSTM, and Decision Tree algorithms showcases the system's adaptability in advancing emotional analysis in

speech, fostering the development of emotionally intelligent technology, and deepening our comprehension of human emotions.

3.2 Implementation

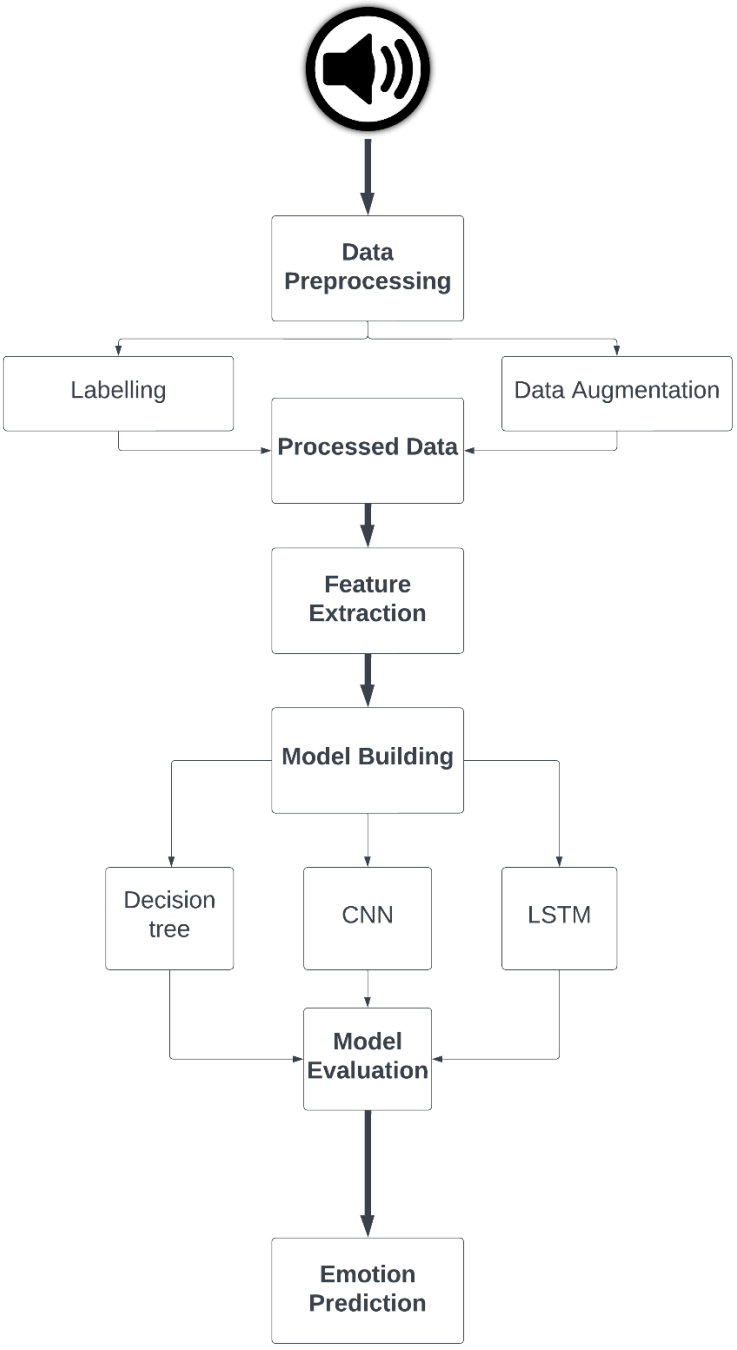

Fig: 3.2. Overview of Implementation

3.2.1 Audio Datasets

The datasets used in this project encompass a diverse range of emotional expressions and speech characteristics crucial for training and evaluating the proposed speech emotion recognition system. The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) provides a comprehensive collection of audiovisual recordings featuring actors portraying various emotions, including calm, happy, sad, angry, fearful, surprised, and neutral. Similarly, the Surrey Audio-Visual Expressed Emotion (SAVEE) dataset offers clear and distinct emotional utterances from male speakers, covering emotions such as anger, disgust, fear, happiness, sadness, surprise, and neutral states. The Crowdsourced Emotional Multimodal Actors Dataset (CREMA-D) and Toronto Emotional Speech Set (TESS) further enrich the dataset pool with audiovisual recordings of actors and speakers displaying a wide array of emotional states, facilitating a thorough analysis of emotional speech recognition across different genders and scenarios. Integrating these datasets provides a robust foundation for training and testing the speech emotion recognition model, ensuring its effectiveness in accurately identifying and categorizing emotions in speech data.

3.2.2 Data Pre-processing

A. Data Labelling

The data labelling process for the speech emotion recognition project involves several systematic steps aimed at organizing and labelling all the datasets for effective analysis and model training. Initially each audio file of the dataset follows a specific name convention that encodes emotional information. Through careful parsing of these filenames, emotions such as 'angry,' 'calm,' 'disgust,' 'fear,' 'happy,' 'neutral,' 'sad,' and 'surprise' are extracted and assigned as labels to the corresponding audio files. This process is crucial as it ensures that each piece of data is accurately categorized based on the underlying emotion conveyed in the speech. The systematic labeling approach employed in the data preprocessing phase lays a solid foundation for subsequent analysis and model training, enabling the speech emotion recognition system to learn and generalize effectively across different emotional expressions.

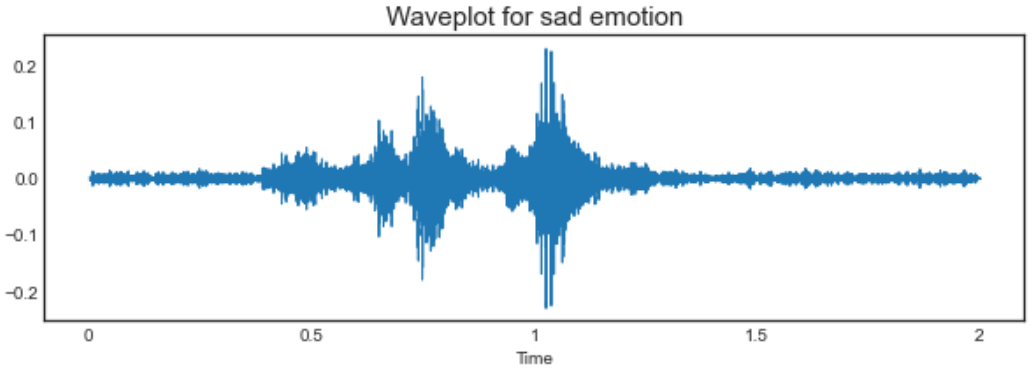

Fig: 3.2.2.A.1 Waveplot for sad emotion of a sample audio

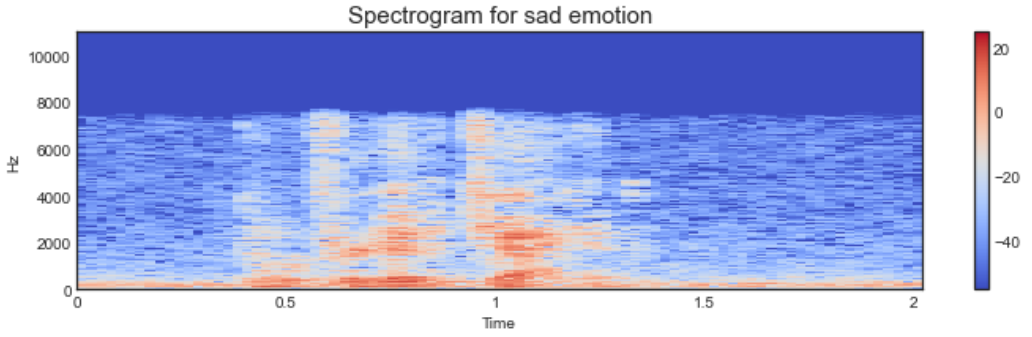

Fig: 3.2.2.A.2. Spectrogram for sad emotion of a sample audio

B. Data Augmentation

Data augmentation is a pivotal technique employed in the speech emotion recognition project to enhance the diversity and robustness of the dataset. This technique involves generating new training samples by applying various transformations to the existing audio data. These transformations may include time stretching, pitch shifting, noise injection, change speed, and other techniques. By augmenting the dataset with these modified versions of the original audio samples, the model becomes more resilient to variations in speech patterns.

One of the primary benefits of data augmentation is its ability to prevent overfitting and improve the generalization capability of the model. By exposing the model to a wider range of synthetic data, it learns to capture and generalize underlying patterns and features more effectively. Moreover, data augmentation helps mitigate the effects of imbalanced datasets by generating additional samples for minority classes, thereby improving the overall performance and accuracy of the emotion recognition system across all emotional categories. Incorporating data augmentation into the training pipeline is crucial for developing a robust and reliable speech

emotion recognition model. It not only enriches the dataset but also enhances the model's ability to accurately recognize and classify emotions in speech data from diverse sources and contexts.

Some of the data augmentation techniques:

- **Noise Injection:** Random noise is introduced into the audio signal to simulate real-world environmental variations and improve model resilience to noise interference.
- **Time Stretch:** Alters the duration of an audio signal while maintaining its pitch and emotional characteristics. This technique uses temporal variations to mimic different speaking rates or durations of emotional expressions.
- **Pitch Shifting:** The pitch of the audio signal is changed, resulting in variations in vocal pitch without changing the emotional content. This technique simulates changes in vocal qualities or speaker identities.
- **Time Shifting:** The audio signal is shifted by a random time interval, resulting in phase shifts and time offsets. This technique represents temporal distortions and variations in the timing of emotional expressions.
- **Speed Modification:** The audio signal's speed is adjusted, either accelerated or decelerated, but its pitch remains unchanged. This technique simulates changes in speech rate or emotional intensity.

3.2.3 Feature Extraction

The feature extraction process in the speech emotion recognition project involves computing Mel-frequency cepstral coefficients (MFCC) from augmented audio data. After applying data augmentation techniques to create varied versions of the audio, MFCC features are computed from these augmented samples. MFCC is a widely accepted method for capturing crucial audio characteristics, particularly in speech processing. These features encode essential information about the spectral content and temporal dynamics of the speech signals, making them highly informative for emotion recognition tasks. The extracted MFCC features contribute to building a diverse and rich dataset essential for training machine learning models in speech emotion recognition. This feature extraction step significantly enhances the dataset's relevance and effectiveness in accurately recognizing and classifying emotions in speech data.

3.2.4 Model Building

The model building process for speech emotion recognition (SER) integrates advanced algorithms like CNNs, LSTMs, and Decision Trees to accurately classify emotional states in speech. By leveraging features such as MFCCs, this approach aims to achieve state-of-the-art results and deepen our understanding of human emotions, with implications for diverse applications.

A. Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed for processing structured data with spatial relationships, such as images and spectrograms. They consist of layers that automatically extract hierarchical features from the input data, starting with low-level features and gradually building up to more abstract representations. CNNs leverage convolutional layers to apply filters across localized regions of the input, capturing patterns and spatial dependencies. This architecture allows CNNs to excel in tasks like image recognition, where features like edges, textures, and shapes are crucial for classification. Additionally, CNNs often incorporate pooling layers to down sample feature maps and reduce computational complexity while preserving important information.

In the context of SER, Convolutional Neural Networks (CNNs) play a crucial role in extracting discriminative features from audio data, such as spectrograms or Mel-frequency cepstral coefficients (MFCCs). CNNs are adept at capturing temporal relationships and patterns in speech signals, including variations in pitch, intensity that convey different emotional states.

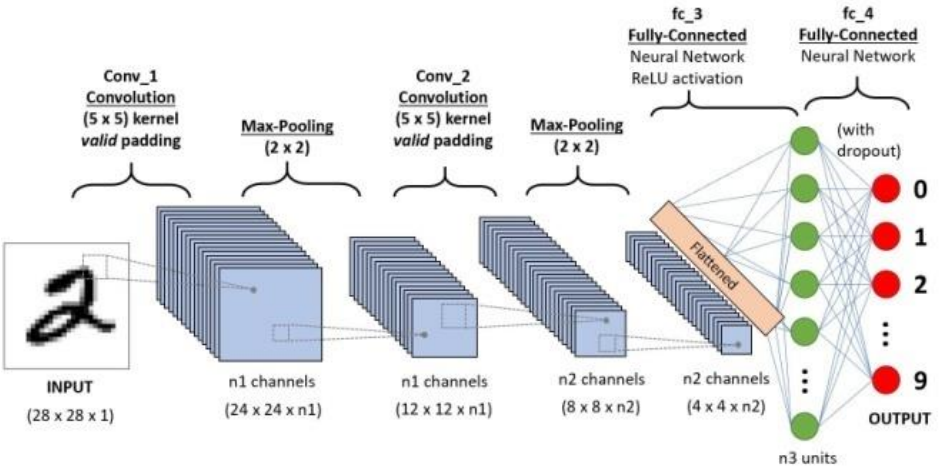

Fig: 3.2.4.1. Architecture of CNN

By analyzing localized regions of spectrograms or MFCCs, CNNs can identify key features associated with specific emotions, such as rising pitch for anger or slower modulation for sadness. This ability to automatically learn and represent emotional cues makes CNNs valuable components in SER models, contributing to their accuracy and robustness in recognizing and classifying emotions conveyed through speech.

B. Long Short-Term Memory

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) designed to process and analyze sequential data with long-term dependencies. Unlike traditional RNNs, LSTMs have specialized memory cells that can retain information over extended time intervals, making them well-suited for tasks involving time-series data like speech. LSTMs excel in capturing temporal patterns and context in sequences, allowing them to understand the nuanced dynamics of emotional expression in speech signals. In SER, LSTMs can effectively model the temporal dependencies of emotional transitions, such as the gradual buildup of intensity or the sustained modulation of pitch associated with specific emotions.

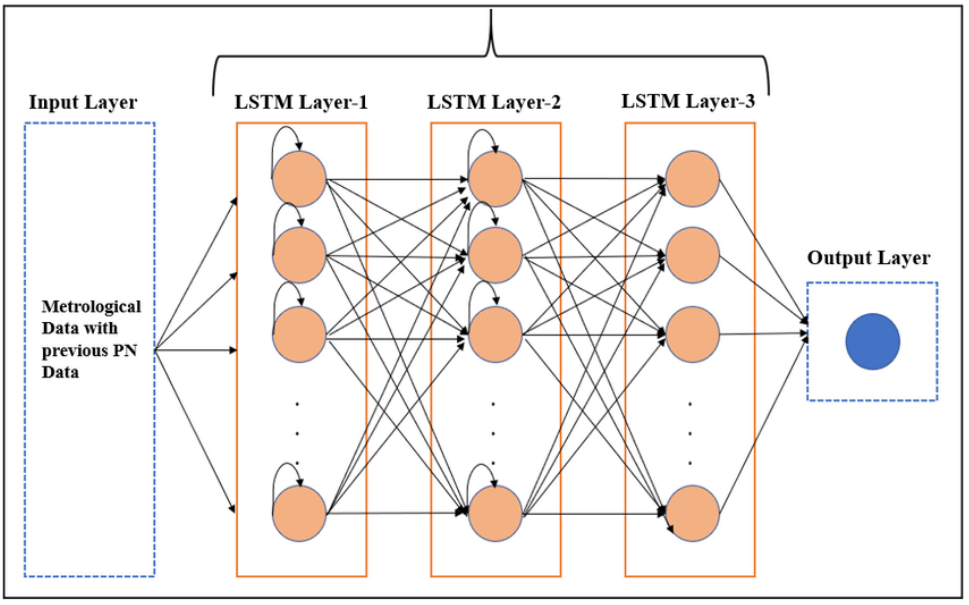

Fig: 3.2.4.2. Architecture of LSTM

In the domain of SER, Long Short-Term Memory (LSTM) networks play a pivotal role in modeling the temporal dynamics and contextual nuances of emotional expression in speech data. LSTMs are particularly adept at capturing long-term dependencies and sequential patterns,

enabling them to discern subtle variations in pitch, rhythm, and intonation that convey different emotional states.

C. Decision Trees

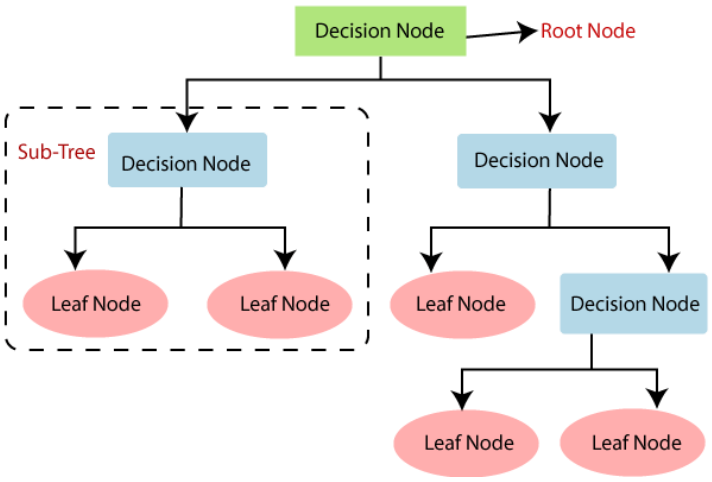

Fig: 3.2.4.3. Architecture of Decision Trees

Decision Trees are non-parametric supervised learning models that partition the input space into regions based on feature values, creating a tree-like structure of decision rules. Each node in the tree represents a decision based on a feature, leading to binary splits that recursively divide the data until reaching leaf nodes corresponding to class labels. Decision Trees are known for their interpretability and ability to handle non-linear relationships and interactions between features. In SER, Decision Trees can be used to identify discriminative features and decision rules that contribute to the classification of different emotional states based on extracted audio features.

Decision Trees can help identify key acoustic features or combinations of features that are indicative of specific emotional states. By partitioning the feature space based on decision thresholds, Decision Trees create intuitive decision boundaries that separate emotions effectively.

3.2.5 Model Evaluation

The evaluation of the developed models for speech emotion recognition involved the use of several key metrics that will later aid in recognizing and classifying emotional states in speech samples.

Accuracy

The capacity of a test to accurately identify weak and strong instances is known as accuracy. We should record the small percentage of true positive and true negative results in thoroughly reviewed instances in order to measure the exactness of a test.

$$Accuracy = \frac{(True\ positives + True\ Negatives)}{Total\ no\ of\ Test\ samples} \quad (1)$$

Precision

Precision quantifies the percentage of correctly classified samples or occurrences among the positives. Consequently, the accuracy may be determined by applying the subsequent formula:

$$Precision = \frac{(True\ positives)}{True\ Positivies + False\ positives} \quad (2)$$

Recall

A ML statistic called recall evaluates a model's ability to identify all relevant samples inside a given class. The ratio of correctly expected positive perceptions to actual positives provides information of a certain class.

$$Recall = \frac{(True\ positives)}{True\ Positivies + False\ Negatives} \quad (3)$$

F1 Score

It is a metric used in ML assessments to gauge a model's accuracy. It combines review scores and model precision. The accuracy measurement calculates the frequency with which a model predicts correctly throughout the whole dataset.

$$F1\ Score = 2X\ \frac{(Precision * Recall)}{(Precision + Recall)} \quad (4)$$

3.2.6 Emotion Prediction

Based on the results, the model with the highest accuracy, which in this case is the CNN model, achieved an accuracy of 0.91. This accuracy is notably higher than the other two models evaluated. Therefore, the CNN model is selected and utilized for emotion prediction of a given sample audio file. Its superior accuracy signifies its effectiveness in accurately recognizing and classifying emotional states in speech data, making it the preferred choice for real-world application and deployment.

CHAPTER 4

RESULTS AND DISCUSSION

4. RESULTS AND DISCUSSION

4.1 Comparison of models

Upon conducting a comprehensive comparative analysis of CNN, LSTM networks, and decision trees for emotion prediction from speech data, several key insights and outcomes have emerged:

Table: 4.1. Performance Evaluation

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1**
CNN	0.9181	0.9345	0.9024	0.9179
LSTM	0.7712	0.8759	0.7290	0.7644
Decision Tree	0.7290	0.7292	0.6810	0.7291

As shown in above Table, The CNN model obtained the maximum accuracy of 91.81%, as well as good precision (93.45%) and recall (90.24%) rates, yielding a strong F1 score of 91.79%. This demonstrates CNN's extraordinary ability to reliably classify emotions from speech data. In contrast, the LSTM model achieved an accuracy of 77.12%, indicating somewhat worse performance than CNN. While LSTM achieved remarkable precision (87.59%), its recall (72.90%) and F1 score (76.44%) were significantly lower, indicating difficulties in understanding some emotional nuances in the dataset. Similarly, the Decision Tree model attained an accuracy of 72.90%, with recall, F1 Score and precision values ranging between 72 and 73%. Although decision trees provide interpretability and simplicity, their performance lags behind CNN and LSTM, demonstrating limits in dealing with the complexity of emotion.

The approach presented in this study achieved an accuracy of 91% in emotion recognition and demonstrated the capability to accurately detect emotions in given speech samples. This performance represents a substantial advancement compared to the existing methods, which relied solely on a smaller number of datasets without utilizing data augmentation techniques and yielded lower accuracies. The CNN model is chosen for its exceptional performance in accurately classifying emotions into the following categories: Angry, Calm, Disgust, Fear, Happy, Neutral, Sad, Surprise

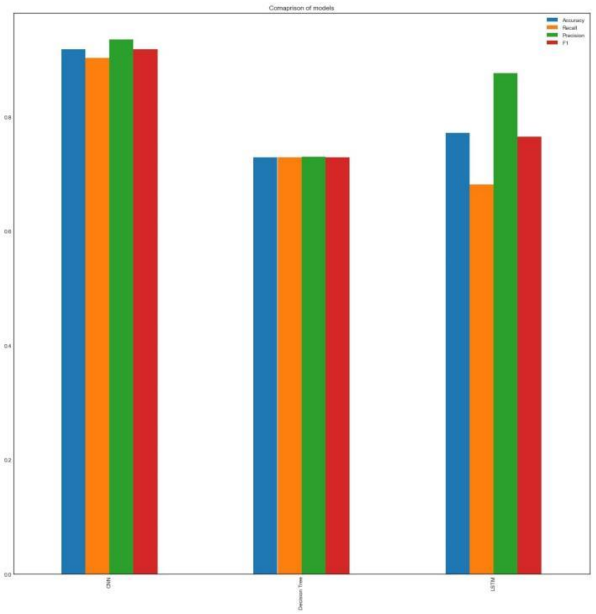

Fig: 4.1. Performance Comparison Graph

In terms of predicting emotions from speech data, the CNN model was used as the primary model. This selection was made based on its demonstrated superiority in accuracy and performance measures when compared to other models like LSTM and decision trees. The CNN model uses convolutional neural networks to analyze and extract information from audio inputs, allowing for accurate emotion classification

4.2 Output

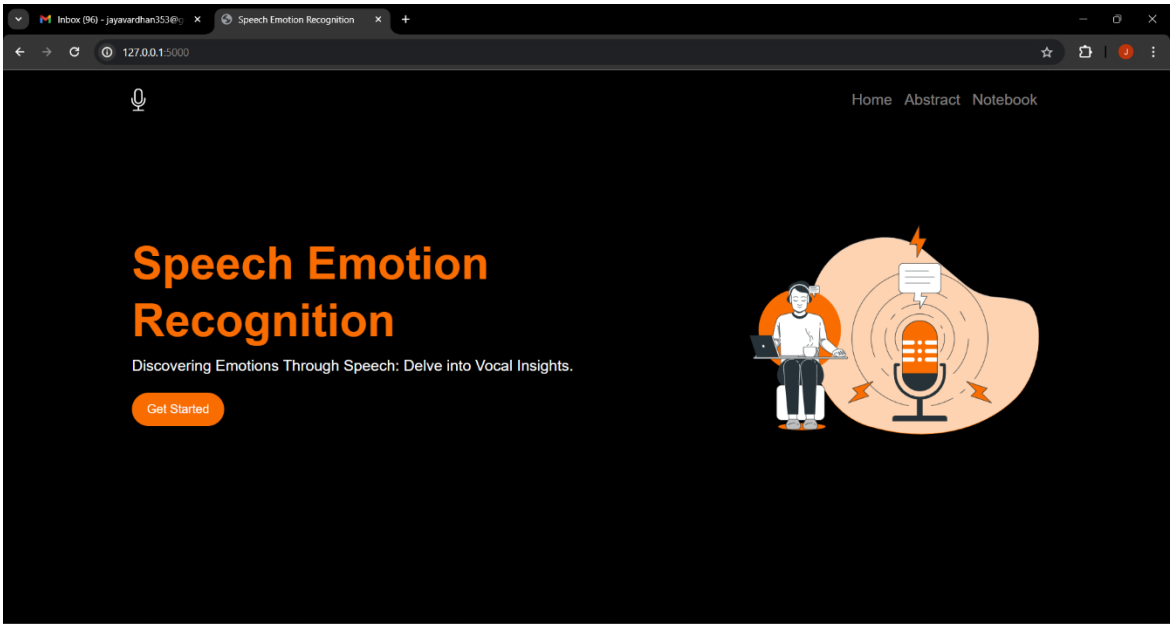

Fig: 4.2.1. Home Page

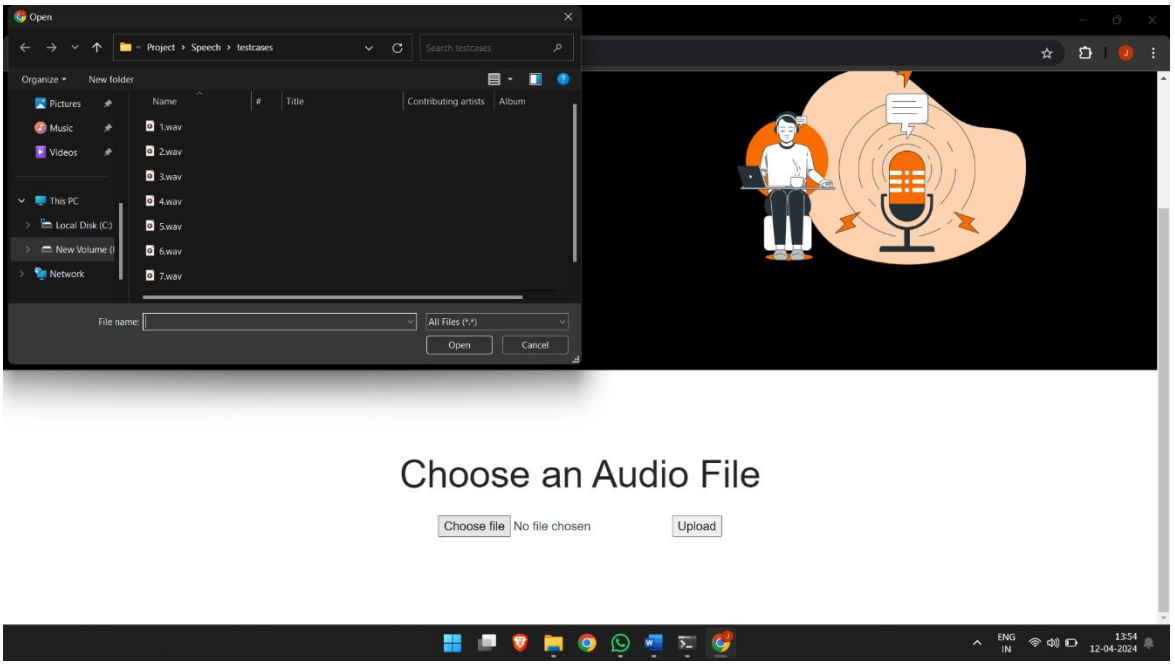

Fig: 4.2.2. Choose an audio file

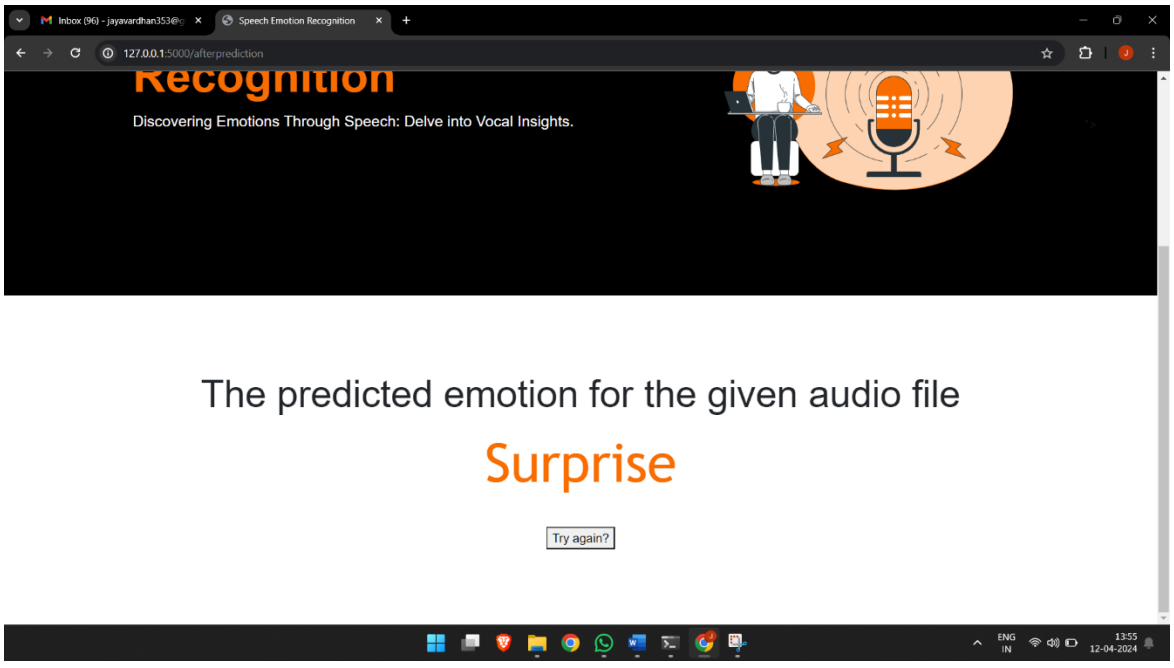

Fig: 4.2.3 Predicted Emotion

CHAPTER 5

CONCLUSION AND FUTURE WORK

5. CONCLUSION AND FUTURE WORK

5.1 Conclusion

The study presented a comprehensive approach to SER using a combination of CNN, LSTM, and decision tree models. Through extensive experimentation and evaluation, several key findings emerged. Firstly, the CNN model outperformed the LSTM and Decision Tree models in emotion classification, with an accuracy of 91.81%. This highlights the effectiveness of CNN architectures in capturing relevant features from audio data for accurate emotion prediction. Secondly, while the LSTM and Decision Tree models showed respectable performance, their accuracies of 77.12% and 72.90%, respectively, were notably lower than those of the CNN model. This suggests that more complex temporal dependencies captured by LSTM may not always translate to better performance in speech emotion recognition tasks, and decision trees may struggle with capturing nuanced patterns in audio data. Furthermore, the study identified specific emotions where each model excelled. For instance, the CNN model demonstrated particularly high accuracy in classifying emotions such as anger, fear, and happiness, indicating its robustness in capturing the spectral features associated with these emotions.

5.2 Future Enhancement

In future work, a promising avenue for enhancement lies in the development of hybrid models that integrate acoustic, contextual, and linguistic features. By combining these diverse modalities of information, hybrid models can potentially achieve more robust and nuanced emotion classification in speech data. Acoustic features such as pitch, intensity, and timbre capture the raw emotional cues present in speech signals, while contextual features consider the surrounding context, such as the speaker's mood or the conversation topic, which can influence the expressed emotions. Additionally, incorporating linguistic features like sentiment analysis or semantic context can provide further context and depth to the emotion recognition process.

REFERENCES

1. Niharika S M, Soumya A: Speech Emotion Recognition using Machine Learning. In: 7th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS). (2023)
2. Majd Saloumi et al: Speech Emotion Recognition using One-Dimensional Convolutional Neural Networks. In: 46th International Conference on Telecommunications and Signal Processing (TSP). (2023)
3. K. Vamsi Krishna et al: Speech Emotion Recognition using ML(SVM). In: 6th International Conference on Computing Methodologies and Communication (ICCMC). (2022)
4. Kim, S. H., & Kim, H. Y. "Emotion Recognition System Using Short-Term Monitoring of Physiological Signals." IEEE Transactions on Biomedical Engineering, 60(12), 3374-3383 (2013)
5. Koolagudi, S. G., & Rao, K. S. "Emotion Recognition from Speech: A Review." International Journal of Speech Technology, 15(2), 99-117 (2012)
6. Kasarapu Ramani, Lakshmi Haritha M. "Deep Learning and its Applications: A Real –World Perspective". In Deep Learning and Edge Computing Solutions for High Performance Computing, Springer. (2021)
7. B. Dinesh, P. Chilukuri, G. P. Sree, K. Venkatesh, M. Delli and K. R. Nandish, "Chat and Voice Bot Implementation for Cardio and ENT Queries Using NLP," 2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA) (2023)
8. A V Sriharsha, Ms K Yochana, “Improving Efficiency of CNN using Octave Convolution”, Journal Article Open Access International Journal of Recent Technology and Engineering (IJRTE), Volume: 8, Issue: 6, Pages: 5412-5418. (2020)
9. Schuller, B., & Batliner, A. "META-INTERSPEECH: A Roadmap Towards the Audio-Visual Processing of Human Emotional Behavior." In Interspeech (2015)
10. Björn Schuller, Anton Batliner et al: T.: Emotion-Oriented Systems: The HUMAINE Handbook, Springer. (2011)
11. Schuller, B., et al. "Recognizing Realistic Emotions and Affect in Speech: State of the Art and Lessons Learnt from the First Challenge." Speech Communication, 53(9-10), 1062-1087. (2011)
12. Lotfian, R., & Cohn, J. F. "Analysis of Head-Pose-Induced Image Variations for Improved Emotion Recognition." In International Conference on Multimodal Interfaces. (2012)
13. Schuller, B., & Steidl, S. "Nonverbal Social Signal Processing in Affective Computing: A Review." Journal on Multimodal User Interfaces, 9(3), 143-159. (2015)

14. Jiang, W., Chen, T., & Sun, M. "Emotion Recognition from Speech: An Overview." Frontiers in Systems Neuroscience, 8, 106. (2014).
15. Zeng, Z., Pantic, M., Roisman, G. I., & Huang, T. S. "A Survey of Affect Recognition Methods: Audio, Visual, and Spontaneous Expressions." IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(1), 39-58. (2009)

APPENDICES

Notebook.py

#Importing libraries

```
import os
import numpy as np # linear algebra
import pandas as pd # data processing
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow.keras.backend as K
# Audio Processing
import librosa
import librosa.display

# to play the audio files
from IPython.display import Audio
plt.style.use('seaborn-white')
from tensorflow.keras.layers import SimpleRNN, Dense, Dropout
## Data Preparation
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split
import pickle
# Modelling
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import load_model
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv1D, MaxPooling1D, Flatten, Dropout, BatchNormalization, AveragePooling1D
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import ModelCheckpoint
```

Importing datasets

```
TESS = "datasets/TESS/"
RAV = "datasets/RAVDESS/"
SAVEE = "datasets/SAVEE/"
CREMA = "datasets/CREMA/"
```

SAVEE

```
# Get the data location for SAVEE
dir_list = os.listdir(SAVEE)
```

```
# parse the filename to get the emotions
emotion=[]
path = []
for i in dir_list:
    if i[-8:-6]=='_a':
        emotion.append('angry')
    elif i[-8:-6]=='_d':
        emotion.append('disgust')
    elif i[-8:-6]=='_f':
        emotion.append('fear')
    elif i[-8:-6]=='_h':
        emotion.append('happy')
    elif i[-8:-6]=='_n':
        emotion.append('neutral')
    elif i[-8:-6]=='sa':
        emotion.append('sad')
    elif i[-8:-6]=='su':
        emotion.append('surprise')
    else:
        emotion.append('unknown')
    path.append(SAVEE + i)

# Now check out the label count distribution
SAVEE_df = pd.DataFrame(emotion, columns = ['labels'])
SAVEE_df = pd.concat([SAVEE_df, pd.DataFrame(path, columns = ['path'])], axis = 1)
print('SAVEE dataset')
SAVEE_df.head()
```

TESS

```
# Get the data location for TESS
path = []
emotion = []
dir_list = os.listdir(TESS)

for i in dir_list:
    fname = os.listdir(TESS + i)
    for f in fname:
        if i == 'OAF_angry' or i == 'YAF_angry':
            emotion.append('angry')
        elif i == 'OAF_disgust' or i == 'YAF_disgust':
            emotion.append('disgust')
        elif i == 'OAF_Fear' or i == 'YAF_fear':
            emotion.append('fear')
        elif i == 'OAF_happy' or i == 'YAF_happy':
            emotion.append('happy')
        elif i == 'OAF_neutral' or i == 'YAF_neutral':
            emotion.append('neutral')
        elif i == 'OAF_Pleasant_surprise' or i == 'YAF_pleasant_surprised':
            emotion.append('surprise')
        elif i == 'OAF_Sad' or i == 'YAF_sad':
```

```
            emotion.append('sad')
        else:
            emotion.append('Unknown')
        path.append(TESS + i + "/" + f)

TESS_df = pd.DataFrame(emotion, columns = ['labels'])
#TESS_df['source'] = 'TESS'
TESS_df = pd.concat([TESS_df,pd.DataFrame(path, columns = ['path'])],axis=1)
print('TESS dataset')
TESS_df.head()
```

RAVDESS

```
# Importing datas from RAVDESS
dir = os.listdir(RAV)

males = []
females = []

for actor in dir:

    files = os.listdir(RAV + actor)

    for file in files:
        part = file.split('.')[0]
        part = part.split("-")

        temp = int(part[6])

        if part[2] == '01':
            emotion = 'neutral'
        elif part[2] == '02':
            emotion = 'calm'
        elif part[2] == '03':
            emotion = 'happy'
        elif part[2] == '04':
            emotion = 'sad'
        elif part[2] == '05':
            emotion = 'angry'
        elif part[2] == '06':
            emotion = 'fear'
        elif part[2] == '07':
            emotion = 'disgust'
        elif part[2] == '08':
            emotion = 'surprise'
        else:
            emotion = 'unknown'

        if temp%2 == 0:
            path = (RAV + actor + '/' + file)
            #emotion = 'female_'+emotion
```

```
            females.append([emotion, path])
        else:
            path = (RAV + actor + '/' + file)
             #emotion = 'male_'+emotion
            males.append([emotion, path])


RavFemales_df = pd.DataFrame(females)
RavFemales_df.columns = ['labels', 'path']

RavMales_df = pd.DataFrame(males)
RavMales_df.columns = ['labels', 'path']

print('RAVDESS datasets')
RavFemales_df.head()

RavMales_df.head()
```

CREMA

```
files = os.listdir(CREMA)

female                                                                                                =
[1002,1003,1004,1006,1007,1008,1009,1010,1012,1013,1018,1020,1021,1024,1025,1028,1029,
1030,1037,1043,1046,1047,1049,

1052,1053,1054,1055,1056,1058,1060,1061,1063,1072,1073,1074,1075,1076,1078,1079,1082,
1084,1089,1091]
males = []
females = []

for file in files:
    part = file.split('_')

    if part[2] == 'SAD':
        emotion = 'sad'
    elif part[2] == 'ANG':
        emotion = 'angry'
    elif part[2] == 'DIS':
        emotion = 'disgust'
    elif part[2] == 'FEA':
        emotion = 'fear'
    elif part[2] == 'HAP':
        emotion = 'happy'
    elif part[2] == 'NEU':
        emotion = 'neutral'
    else:
        emotion = 'unknown'

    if int(part[0]) in female:
```

```
        path = (CREMA + '/' + file)
        #emotion = 'female_'+emotion
        females.append([emotion, path])
    else:
        path = (CREMA + '/' + file)
        #emotion = 'male_'+emotion
        males.append([emotion, path])

CremaFemales_df = pd.DataFrame(females)
CremaFemales_df.columns = ['labels', 'path']

CremaMales_df = pd.DataFrame(males)
CremaMales_df.columns = ['labels', 'path']

print('CREMA datasets')
CremaFemales_df.head()

# Now lets merge all the dataframe
Males = pd.concat([SAVEE_df, RavMales_df, CremaMales_df], axis = 0)
Males.to_csv("males_emotions_df.csv", index = False)

Females = pd.concat([TESS_df, RavFemales_df, CremaFemales_df], axis = 0)
Females.to_csv("females_emotions_df.csv", index = False)
```

DATA VISUALIZATION

```
def create_waveplot(data, sr, e):
    plt.figure(figsize=(10, 3))
    plt.title('Waveplot for {} emotion'.format(e), size=15)
    librosa.display.waveshow(data, sr=sr)
    plt.show()

def create_spectrogram(data, sr, e):
    X = librosa.stft(data)
    Xdb = librosa.amplitude_to_db(abs(X))
    plt.figure(figsize=(12, 3))
    plt.title('Spectrogram for {} emotion'.format(e), size=15)
    librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='hz')
    plt.colorbar()

Crema_df = CremaMales_df

emotion='sad'
path = np.array(Crema_df.path[Crema_df.labels==emotion])[0]
data, sampling_rate = librosa.load(path)
create_waveplot(data, sampling_rate, emotion)
create_spectrogram(data, sampling_rate, emotion)
Audio(path)

emotion='angry'
path = np.array(Crema_df.path[Crema_df.labels==emotion])[0]
```

```
data, sampling_rate = librosa.load(path)
create_waveplot(data, sampling_rate, emotion)
create_spectrogram(data, sampling_rate, emotion)
Audio(path)

emotion='disgust'
path = np.array(Crema_df.path[Crema_df.labels==emotion])[0]
data, sampling_rate = librosa.load(path)
create_waveplot(data, sampling_rate, emotion)
create_spectrogram(data, sampling_rate, emotion)
Audio(path)

emotion='neutral'
path = np.array(Crema_df.path[Crema_df.labels==emotion])[0]
data, sampling_rate = librosa.load(path)
create_waveplot(data, sampling_rate, emotion)
create_spectrogram(data, sampling_rate, emotion)
Audio(path)
```

Data Augmentation

```
def noise(data):
    noise_amp = 0.04*np.random.uniform()*np.amax(data)
    data = data + noise_amp*np.random.normal(size=data.shape[0])
    return data

def stretch(data, rate=0.70):
    return librosa.effects.time_stretch(data, rate)

def shift(data):
    shift_range = int(np.random.uniform(low=-5, high = 5)*1000)
    return np.roll(data, shift_range)

def pitch(data, sampling_rate, pitch_factor=0.8):
    return librosa.effects.pitch_shift(data, sampling_rate, pitch_factor)

def higher_speed(data, speed_factor = 1.25):
    return librosa.effects.time_stretch(data, speed_factor)

def lower_speed(data, speed_factor = 0.75):
    return librosa.effects.time_stretch(data, speed_factor)
```

Feature Extraction

```
#sample_rate = 22050

def extract_features(data):

    result = np.array([])
```

```
    #mfccs = librosa.feature.mfcc(y=data, sr=22050, n_mfcc=42) #42 mfcc so we get frames of ~60 ms
    mfccs = librosa.feature.mfcc(y=data, sr=22050, n_mfcc=58)
    mfccs_processed = np.mean(mfccs.T,axis=0)
    result = np.array(mfccs_processed)

    return result

def get_features(path):
    # duration and offset are used to take care of the no audio in start and the ending of each audio files as seen above.
    data, sample_rate = librosa.load(path, duration=3, offset=0.5, res_type='kaiser_fast')

    #without augmentation
    res1 = extract_features(data)
    result = np.array(res1)

    #noised
    noise_data = noise(data)
    res2 = extract_features(noise_data)
    result = np.vstack((result, res2)) # stacking vertically

    #stretched
    stretch_data = stretch(data)
    res3 = extract_features(stretch_data)
    result = np.vstack((result, res3))

    #shifted
    shift_data = shift(data)
    res4 = extract_features(shift_data)
    result = np.vstack((result, res4))

    #pitched
    pitch_data = pitch(data, sample_rate)
    res5 = extract_features(pitch_data)
    result = np.vstack((result, res5))

    #speed up
    higher_speed_data = higher_speed(data)
    res6 = extract_features(higher_speed_data)
    result = np.vstack((result, res6))

    #speed down
    lower_speed_data = higher_speed(data)
    res7 = extract_features(lower_speed_data)
    result = np.vstack((result, res7))

    return result

female_X, female_Y = [], []
for path, emotion in zip(Females.path, Females.labels):
```

```
    features = get_features(path)
    #adding augmentation, get_features return a multi dimensional array (for each augmentation), so we have to use a loop to fill the df
    for elem in features:
        female_X.append(elem)
        female_Y.append(emotion)

male_X, male_Y = [], []
for path, emotion in zip(Males.path, Males.labels):
    features = get_features(path)
    for elem in features:
        male_X.append(elem)
        male_Y.append(emotion)

print(f'Check shapes:\nFemale features: {len(female_X)}, labels: {len(female_Y)}\nMale features:  {len(male_X)}, labels: {len(male_Y)}')

def setup_dataframe(gender, features, labels):
    df = pd.DataFrame(features)
    df['labels'] = labels
    df.to_csv(f'{gender}_features.csv', index=False)

    print(f'{gender} dataframe')
    df.sample(frac=1).head()

    return df

Females_Features = setup_dataframe('Female', female_X, female_Y)
Males_Features = setup_dataframe('Male', male_X, male_Y)
```

Data Preparation

```
female_X = Females_Features.iloc[: ,:-1].values
female_Y = Females_Features['labels'].values

male_X = Males_Features.iloc[: ,:-1].values
male_Y = Males_Features['labels'].values

# As this is a multiclass classification problem onehotencoding our Y.
encoder = OneHotEncoder()

female_Y = encoder.fit_transform(np.array(female_Y).reshape(-1,1)).toarray()
male_Y = encoder.fit_transform(np.array(male_Y).reshape(-1,1)).toarray()
```

Splitting Data

```
X = np.concatenate((female_X, male_X))
Y = np.concatenate((female_Y, male_Y))

x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state=0, test_size=0.20, shuffle=True)
```

```
x_train.shape, y_train.shape, x_test.shape, y_test.shape

scaler = StandardScaler()

x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

x_train = np.expand_dims(x_train, axis=2)
x_test = np.expand_dims(x_test, axis=2)
x_train.shape, y_train.shape , x_test.shape , y_test.shape

pickle.dump(scaler, open('tk.pkl', 'wb'))
```

Modelling

```
## To check if a system has the necessary GPU resources available to train our large models
print("Num GPUs Available: ", len(tf.config.experimental.list_physical_devices('GPU')))

# Create a MirroredStrategy.
strategy = tf.distribute.MirroredStrategy()
print('Number of devices: {}'.format(strategy.num_replicas_in_sync))

def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_score(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)

    return 2*((precision*recall)/(precision+recall+K.epsilon()))



def specificity_m(y_true, y_pred):
    true_negatives = K.sum(K.round(K.clip((1 - y_true) * (1 - y_pred), 0, 1)))
    possible_negatives = K.sum(K.round(K.clip(1 - y_true, 0, 1)))
    specificity = true_negatives / (possible_negatives + K.epsilon())
    return specificity

def sensitivity_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
```

```
    sensitivity = true_positives / (possible_positives + K.epsilon())
    return sensitivity
```

CNN

```
with strategy.scope():

    def build_model(in_shape):

        model=Sequential()
        model.add(Conv1D(256, kernel_size=6, strides=1, padding='same', activation='relu', input_shape=(in_shape, 1)))
        model.add(AveragePooling1D(pool_size=4, strides = 2, padding = 'same'))

        model.add(Conv1D(128, kernel_size=6, strides=1, padding='same', activation='relu'))
        model.add(AveragePooling1D(pool_size=4, strides = 2, padding = 'same'))

        model.add(Conv1D(128, kernel_size=6, strides=1, padding='same', activation='relu'))
        model.add(AveragePooling1D(pool_size=4, strides = 2, padding = 'same'))
        model.add(Dropout(0.2))

        model.add(Conv1D(64, kernel_size=6, strides=1, padding='same', activation='relu'))
        model.add(MaxPooling1D(pool_size=4, strides = 2, padding = 'same'))

        model.add(Flatten())
        model.add(Dense(units=32, activation='relu'))
        model.add(Dropout(0.3))

        model.add(Dense(units=8, activation='softmax'))
        model.compile(optimizer = 'adam' , loss = 'categorical_crossentropy' , metrics = ['accuracy', f1_score, recall_m, precision_m, specificity_m, sensitivity_m])

        return model

def model_build_summary(mod_dim, tr_features, val_features, val_labels):
    model = build_model(mod_dim)
    model.summary()

    score = model.evaluate(val_features, val_labels, verbose = 1)
    accuracy = 100*score[1]

    return model

rlrp = ReduceLROnPlateau(monitor='loss', factor=0.4, verbose=0, patience=4, min_lr=0.000001)

batch_size = 32
n_epochs = 75

total_model = model_build_summary(x_train.shape[1], x_train, x_test, y_test)
```

```
history = total_model.fit(x_train, y_train, batch_size=batch_size, epochs=n_epochs, validation_data=(x_test, y_test), callbacks=[rlrp])

# Save Model
total_model.save('models/modelV2.h5')

# Extract the metrics from history
train_acc = history.history['accuracy']
train_recall = history.history['recall_m']
train_precision = history.history['precision_m']
train_f1 = history.history['f1_score']
train_sensitivity = history.history['sensitivity_m']
train_specificity = history.history['specificity_m']

val_acc = history.history['val_accuracy']
val_recall = history.history['val_recall_m']
val_precision = history.history['val_precision_m']
val_f1 = history.history['val_f1_score']
val_sensitivity = history.history['val_sensitivity_m']
val_specificity = history.history['val_specificity_m']

# Create a figure and subplot for each metric
fig, axs = plt.subplots(2, 3, figsize=(18, 8))

# Plot accuracy
axs[0, 0].plot(train_acc, label='Train')
axs[0, 0].plot(val_acc, label='Validation')
axs[0, 0].set_title('Accuracy')
axs[0, 0].legend()

# Plot precision
axs[0, 1].plot(train_precision, label='Train')
axs[0, 1].plot(val_precision, label='Validation')
axs[0, 1].set_title('Precision')
axs[0, 1].legend()

# Plot recall
axs[0, 2].plot(train_recall, label='Train')
axs[0, 2].plot(val_recall, label='Validation')
axs[0, 2].set_title('Recall')
axs[0, 2].legend()

# Plot F1 score
axs[1, 0].plot(train_f1, label='Train')
axs[1, 0].plot(val_f1, label='Validation')
axs[1, 0].set_title('F1 Score')
axs[1, 0].legend()

# Plot sensitivity
axs[1, 1].plot(train_sensitivity, label='Train')
axs[1, 1].plot(val_sensitivity, label='Validation')
```

```
axs[1, 1].set_title('Sensitivity')
axs[1, 1].legend()

# Plot specificity
axs[1, 2].plot(train_specificity, label='Train')
axs[1, 2].plot(val_specificity, label='Validation')
axs[1, 2].set_title('Specificity')
axs[1, 2].legend()

# Adjust spacing between subplots
plt.tight_layout()

# Display the plot
plt.show()

a = history.history['accuracy'][-1]
f = history.history['f1_score'][-1]
p = history.history['precision_m'][-1]
r = history.history['recall_m'][-1]


print('Accuracy = ' + str(a * 100))
print('Precision = ' + str(p * 100))
print('F1 Score = ' + str(f * 100))
print('Recall = ' + str(r * 100))

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()


# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

Decision Tree

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state=0, test_size=0.20, shuffle=True)
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=0)
clf.fit(x_train, y_train)
```

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

predictions = clf.predict(x_test)

accuracy = accuracy_score(y_test, predictions)

precision = precision_score(y_test, predictions, average='weighted')

recall = recall_score(y_test, predictions, average='weighted')

f1 = f1_score(y_test, predictions, average='weighted')

print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
```

LSTM

```
def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_score(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)

    return 2*((precision*recall)/(precision+recall+K.epsilon()))



def specificity_m(y_true, y_pred):
    true_negatives = K.sum(K.round(K.clip((1 - y_true) * (1 - y_pred), 0, 1)))
    possible_negatives = K.sum(K.round(K.clip(1 - y_true, 0, 1)))
    specificity = true_negatives / (possible_negatives + K.epsilon())
    return specificity

def sensitivity_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    sensitivity = true_positives / (possible_positives + K.epsilon())
    return sensitivity
```

```
x_train_reshaped = x_train.reshape(x_train.shape[0], x_train.shape[1], 1)
x_test_reshaped = x_test.reshape(x_test.shape[0], x_test.shape[1], 1)

model = Sequential()
model.add(LSTM(units=64, input_shape=(x_train_reshaped.shape[1], x_train_reshaped.shape[2])))
model.add(Dense(8))


model.compile(optimizer = 'adam' , loss = 'mean_squared_error' , metrics = ['accuracy', f1_score, recall_m, precision_m, specificity_m, sensitivity_m])

model.summary()

history = model.fit(x_train_reshaped, y_train, batch_size=batch_size, epochs=n_epochs, validation_data=(x_test_reshaped, y_test), callbacks=[rlrp])

model.save('models/lstm.h5')

# Extract the metrics from history
train_acc = history.history['accuracy']
train_recall = history.history['recall_m']
train_precision = history.history['precision_m']
train_f1 = history.history['f1_score']
train_sensitivity = history.history['sensitivity_m']
train_specificity = history.history['specificity_m']

val_acc = history.history['val_accuracy']
val_recall = history.history['val_recall_m']
val_precision = history.history['val_precision_m']
val_f1 = history.history['val_f1_score']
val_sensitivity = history.history['val_sensitivity_m']
val_specificity = history.history['val_specificity_m']

# Create a figure and subplot for each metric
fig, axs = plt.subplots(2, 3, figsize=(18, 8))

# Plot accuracy
axs[0, 0].plot(train_acc, label='Train')
axs[0, 0].plot(val_acc, label='Validation')
axs[0, 0].set_title('Accuracy')
axs[0, 0].legend()

# Plot precision
axs[0, 1].plot(train_precision, label='Train')
axs[0, 1].plot(val_precision, label='Validation')
axs[0, 1].set_title('Precision')
axs[0, 1].legend()

# Plot recall
```

```
axs[0, 2].plot(train_recall, label='Train')
axs[0, 2].plot(val_recall, label='Validation')
axs[0, 2].set_title('Recall')
axs[0, 2].legend()

# Plot F1 score
axs[1, 0].plot(train_f1, label='Train')
axs[1, 0].plot(val_f1, label='Validation')
axs[1, 0].set_title('F1 Score')
axs[1, 0].legend()

# Plot sensitivity
axs[1, 1].plot(train_sensitivity, label='Train')
axs[1, 1].plot(val_sensitivity, label='Validation')
axs[1, 1].set_title('Sensitivity')
axs[1, 1].legend()

# Plot specificity
axs[1, 2].plot(train_specificity, label='Train')
axs[1, 2].plot(val_specificity, label='Validation')
axs[1, 2].set_title('Specificity')
axs[1, 2].legend()

# Adjust spacing between subplots
plt.tight_layout()

# Display the plot
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()


# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

a1 = history.history['accuracy'][-1]
f11 = history.history['f1_score'][-1]
p1 = history.history['precision_m'][-1]
r1 = history.history['recall_m'][-1]
```

```
print('Accuracy = ' + str(a1 * 100))
print('Precision = ' + str(p1 * 100))
print('F1 Score = ' + str(f11 * 100))
print('Recall = ' + str(r1 * 100))
```

Comparison

```
results ={'Accuracy': [a,accuracy,a1],
 'Recall':[r,recall,r1],
 'Precision': [p,precision,p1],
    'F1' : [f,f1,f11]}
index = ['CNN','Decision Tree','LSTM']

results =pd.DataFrame(results,index=index)
print(results)

fig =results.plot(kind='bar',title='Comaprison of models',figsize =(19,19)).get_figure()
fig.savefig('Final Result.png')

results.plot(subplots=True,kind ='bar',figsize=(4,10))

accuracy_values = results['Accuracy']
accuracy_df = pd.DataFrame({'Algorithms': index, 'Accuracy': accuracy_values})

plt.figure(figsize=(8, 6))
plt.bar(accuracy_df['Algorithms'], accuracy_df['Accuracy'], color='blue')
plt.title('Accuracy of Different Algorithms')
plt.xlabel('Algorithms')
plt.ylabel('Accuracy')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

recall_values = results['Recall']
recall_df = pd.DataFrame({'Algorithms': index, 'Recall': recall_values})

plt.figure(figsize=(8, 6))
plt.bar(recall_df['Algorithms'], recall_df['Recall'], color='orange')
plt.title('Recall of Different Algorithms')
plt.xlabel('Algorithms')
plt.ylabel('Recall')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()


precision_values = results['Precision']
precision_df = pd.DataFrame({'Algorithms': index, 'Precision': precision_values})
```

```
plt.figure(figsize=(8, 6))
plt.bar(precision_df['Algorithms'], precision_df['Precision'], color='green')
plt.title('Precision of Different Algorithms')
plt.xlabel('Algorithms')
plt.ylabel('Precision')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

f1_values = results['F1']
f1_df = pd.DataFrame({'Algorithms': index, 'F1 Score': f1_values})

plt.figure(figsize=(8, 6))
plt.bar(f1_df['Algorithms'], f1_df['F1 Score'], color='red')
plt.title('F1 Score of Different Algorithms')
plt.xlabel('Algorithms')
plt.ylabel('F1 Score')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

App.py

```
import os

import numpy as np

# Keras
import pickle
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
# Flask utils
from flask import Flask, redirect, url_for, request, render_template
import librosa

app = Flask(__name__)

UPLOAD_FOLDER = 'files'

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

UPLOAD_FOLDER1 = 'static/uploads/'

def specificity_m(y_true, y_pred):
    true_negatives = K.sum(K.round(K.clip((1 - y_true) * (1 - y_pred), 0, 1)))
    possible_negatives = K.sum(K.round(K.clip(1 - y_true, 0, 1)))
    specificity = true_negatives / (possible_negatives + K.epsilon())
    return specificity

def sensitivity_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    sensitivity = true_positives / (possible_positives + K.epsilon())
    return sensitivity

def mae(y_true, y_pred):
    return K.mean(K.abs(y_true - y_pred))

def mse(y_true, y_pred):
    return K.mean(K.square(y_true - y_pred))

model_path2 = 'models/modelV2.h5' # load .h5 Model

custom_objects = {
    'f1_score': f1_score,
```

```
    'recall_m': recall_score,
    'precision_m': precision_score,
    'specificity_m': specificity_m,
    'sensitivity_m': sensitivity_m,
    'mae' : mae,
    'mse' : mse
}


model = load_model(model_path2, custom_objects=custom_objects)

model_name = open("tk.pkl","rb")
scaler = pickle.load(model_name)

def extract_features(data):
    result = np.array([])

    mfccs = librosa.feature.mfcc(y=data, sr=22050, n_mfcc=58)
    mfccs_processed = np.mean(mfccs.T,axis=0)
    result = np.array(mfccs_processed)

    return result

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

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

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

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

@app.route('/afterprediction',methods=['GET','POST'])
def afterprediction():
    print("Entered")

    print("Entered here")
    file = request.files['files'] # fet input
    filename = file.filename
    print("@@ Input posted = ", filename)
```

```
    file_path = os.path.join(UPLOAD_FOLDER1, filename)
    file.save(file_path)

    duration = 3
    test_data, _ = librosa.load(file_path, duration=duration, res_type='kaiser_fast')
    test_features = extract_features(test_data)
    test_features = scaler.transform(test_features.reshape(1, -1))  # Scale the features
    test_features = np.expand_dims(test_features, axis=2)  # Add a dimension for CNN input

    # Make predictions using the trained model
    predictions = model.predict(test_features)
    predicted_class = np.argmax(predictions)

    classes = {0:'Angry', 1:'Calm', 2:'Disgust', 3:'Fear', 4:'Happy', 5:'Neutral', 6:'Sad', 7:'Surprise'}

    pred = classes[predicted_class]
    print(pred)

    return render_template('afterprediction.html', pred_output=pred)


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

PROJECT - OUTCOMES

For the research we conducted on Deep Learning Approach for Speech Emotion Recognition, we explored existing systems and identified their drawbacks and limitations. We have authored a research paper detailing my findings and proposed solution, which was submitted to 2024 International Conference on Computational Innovations and Emerging Trends (ICCIET 2K24), organized by Atlantis Press. We're pleased to announce that the conference committee accepted my paper, indicating its significance and contribution to the field. Moreover, we delivered an online presentation of the research paper, showcasing the efforts invested in the research and presentation work. As recognition of this endeavor, we received a presentation certificate, acknowledging the value and impact of the research contribution.

Presentation Certificates

International Conference on Computational Innovations and Emerging Trends (ICCIET-2K24)

organized By

Department of Computer Science Engineering & Allied Branches

Bonam Venkata Chalamayya Engg. College, Odalarevu, AP-INDIA

ATLANTIS PRESS Part of SPRINGER NATURE

Certificate of Presentation

Presented to

G. Sai Kumar

of

UG Scholar, Department of Computer Science and Systems Engineering, Sree Vidyanikethan Engineering College, Tirupati

For presenting a paper titled **"Comparative Analysis of Machine Learning Models for Emotion Classification in Speech Data"** during 4th- 5thApril, 2024.

Dr.BSNMurthy (Co-Convenor) | Dr. P Subba Rao (Convenor) | Dr.Gunamani Jena (Conference Chair) | Dr. Chandra Mouli VSA (Principal-BVCE)

International Conference on Computational Innovations and Emerging Trends (ICCIET-2K24)

organized By

Department of Computer Science Engineering & Allied Branches

Bonam Venkata Chalamayya Engg. College, Odalarevu, AP-INDIA

ATLANTIS PRESS Part of SPRINGER NATURE

Certificate of Presentation

Presented to

G. Jaya vardhan Raju

of

UG Scholar, Department of Computer Science and Systems Engineering, Sree Vidyanikethan Engineering College, Tirupati

For presenting a paper titled **"Comparative Analysis of Machine Learning Models for Emotion Classification in Speech Data"** during 4th- 5thApril, 2024.

Dr.BSNMurthy (Co-Convenor)	Dr. P Subba Rao (Convenor)	Dr.Gunamani Jena (Conference Chair)	Dr. Chandra Mouli VSA (Principal-BVCE)

International Conference on Computational Innovations and Emerging Trends (ICCIET-2K24)

organized By

Department of Computer Science Engineering & Allied Branches

Bonam Venkata Chalamayya Engg. College, Odalarevu, AP-INDIA

ATLANTIS PRESS Part of SPRINGER NATURE

Certificate of Presentation

Presented to

V. Sushvitha

of

UG Scholar, Department of Computer Science and Systems Engineering, Sree Vidyanikethan Engineering College, Tirupati

For presenting a paper titled **"Comparative Analysis of Machine Learning Models for Emotion Classification in Speech Data"** during 4th- 5thApril, 2024.

Dr.BSNMurthy (Co-Convenor)	Dr. P Subba Rao (Convenor)	Dr.Gunamani Jena (Conference Chair)	Dr. Chandra Mouli VSA (Principal-BVCE)

International Conference on Computational Innovations and Emerging Trends (ICCIET-2K24)

organized By

Department of Computer Science Engineering & Allied Branches

Bonam Venkata Chalamayya Engg. College, Odalarevu, AP-INDIA

Certificate of Presentation

Presented to

G. Chaithanya

of

UG Scholar, Department of Computer Science and Systems Engineering, Sree Vidyanikethan Engineering College, Tirupati

For presenting a paper titled **"Comparative Analysis of Machine Learning Models for Emotion Classification in Speech Data"** during 4th- 5thApril, 2024.

Dr.BSNMurthy (Co-Convenor)	Dr. P Subba Rao (Convenor)	Dr.Gunamani Jena (Conference Chair)	Dr. Chandra Mouli VSA (Principal-BVCE)

COLLEGE VISION & MISSION

VISION

To be one of the Nation's premier Engineering Colleges by achieving the highest order of excellence in Teaching and Research.

MISSION

Through multidimensional excellence, we value intellectual curiosity, pursuit of knowledge building and dissemination, academic freedom and integrity to enable the students to realize their potential. We promote technical mastery of Progressive Technologies, understanding their ramifications in the future society and nurture the next generation of skilled professionals to compete in an increasingly complex world, which requires practical and critical understanding of all aspects.

DEPARTMENT VISION & MISSION

VISION

- To become a nationally recognized quality education center in the domain of Computer Science and Systems Engineering through teaching, training, learning, research and consultancy.

MISSION

- The Department offers undergraduate program in Computer Science and Systems Engineering and Post graduate program in Software Engineering to produce high quality information technologists and software engineers by disseminating knowledge through contemporary curriculum, competent faculty and adopting effective teaching-learning methodologies.
- Igniting passion among students for research and innovation by exposing them to real time systems and problems.
- Developing technical and life skills in diverse community of students with modern training methods to solve problems in Software Industry.
- Inculcating values to practice engineering in adherence to code of ethics in multicultural and multi discipline

PROGRAM EDUCATIONAL OBJECTIVES (PEO'S)

After few years of graduation, the graduates of B. Tech (CSBS) will:

1. Enrolled or completed higher education in the core or allied areas of Computer Science and Systems Engineering or management.
2. Successful entrepreneurial or technical career in the core or allied areas of Computer Science and Systems Engineering.
3. Continued to learn and to adapt to the world of constantly evolving technologies in the core or allied areas of Computer Science and Systems Engineering.

PROGRAM SPECIFIC OUTCOMES (PSO'S)

On successful completion of the Program, the graduates of B. Tech (IT) program will be able to:

PSO1 Design and develop database systems, apply data analytics techniques, and use advanced databases for data storage, processing and retrieval.

PSO2 Apply network security techniques and tools for the development of highly secure systems.

PSO3 Analyze, design and develop efficient algorithms and software applications to deploy in secure environment to support contemporary services using programming languages, tools and technologies.

PSO4 Apply concepts of computer vision and artificial intelligent for the development of efficient intelligent systems and applications

PROGRAM OUTCOMES (PO'S)

1. Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems (**Engineering knowledge**).
2. Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences (**Problem analysis**).
3. Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations (**Design/development of solutions**).
4. Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions (**Conduct investigations of complex problems**).
5. Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations (**Modern tool usage**)
6. Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice (**The engineer and society**)
7. Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development (**Environment and sustainability**).
8. Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice (**Ethics**).
9. Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings (**Individual and team work**).
10. Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions (**Communication**).
11. Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to

manage projects and in multidisciplinary environments (**Project management and finance**).

12. Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change (**Life-long learning).**

COURSE OUTCOMES (PO'S)

After successful completion of this course, the students will be able to:

CO1 Create/Design algorithms and software to solve complex Computer Science and Systems Engineering and allied problems using appropriate tools and techniques following relevant standards, codes, policies, regulations and latestdevelopments.

CO2 Consider society, health, safety, environment, sustainability, economics and project management in solving complex Computer Science and Systems Engineering and allied problems.

CO3 Perform individually or in a team besides communicating effectively in written, oral and graphical forms on Computer Science and Systems Engineering based systems or processes.

Mapping of Course Outcomes with POs and PSOs:

Course Outcomes	Program Outcomes												Program Specific Outcomes		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	3	3	3	3	-	-	3	-	-	-	3	3	3	3
CO2	-	-	-	-	-	3	3	-	-	-	3	-	3	3	3
CO3	-	-	-	-	-	-	-	-	3	3	-	-	3	3	3
Average	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Level of correlation of the course	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3