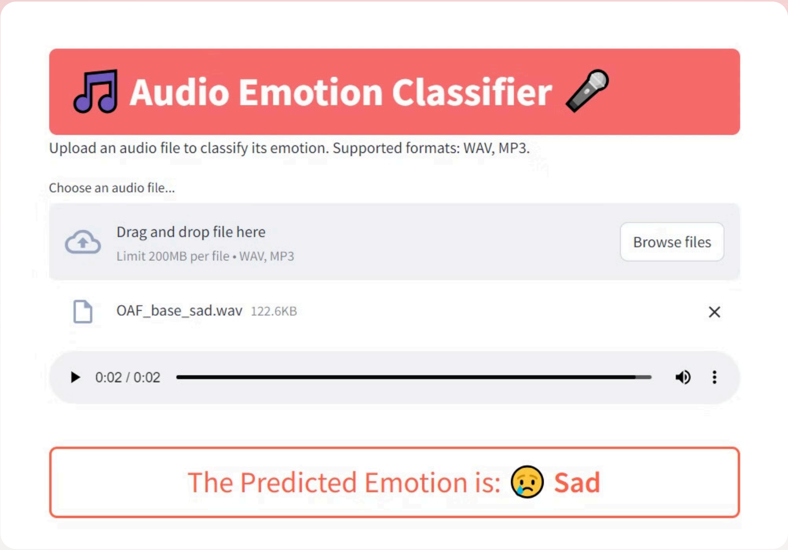
Emotion Recognition From Speech

This project explores the fascinating field of emotion recognition from speech. By analyzing audio data, we aim to develop a model capable of identifying different emotions expressed through spoken words.

by Adil naeem



Life Cycle Of Project

1 Data Collection

The project begins with the collection of audio data. This involves recording individuals speaking specific target words while expressing different emotions. The dataset is carefully curated to ensure a diverse range of voices and emotional expressions.

2 Data Preprocessing

Once collected, the audio data undergoes preprocessing. This step involves cleaning and preparing the data for analysis. It may include tasks like noise reduction, normalization, and segmentation.

3 Data Augmentation

To enhance the model's robustness and generalization ability, data augmentation techniques are applied. These techniques create synthetic data samples by introducing variations to the original recordings, such as adding noise or altering the pitch.

4 Feature Extraction

Feature extraction is a crucial step where relevant information is extracted from the audio data. This involves analyzing the audio signals and extracting features like zero crossing rate, MFCCs, and spectrograms.

5 Model Training

The extracted features are then used to train a machine learning model. The model learns to associate specific features with different emotions, enabling it to predict emotions from new audio data.

6 Model Evaluation

After training, the model's performance is evaluated using a separate test dataset. This helps assess the model's accuracy in recognizing emotions and identify areas for improvement.

7 Save The Model

Once the model has been trained and evaluated, it is saved for future use. This allows for easy deployment and application of the model to new audio data.

8 Model Deployment

The final step involves deploying the trained model. This could involve integrating it into a web application, a mobile app, or other systems where emotion recognition is desired.

About Dataset Content

1 Target Words

The dataset consists of 200 target words spoken by two actresses, each expressing seven different emotions: anger, disgust, fear, happiness, pleasant surprise, sadness, and neutral.

2 Actresses

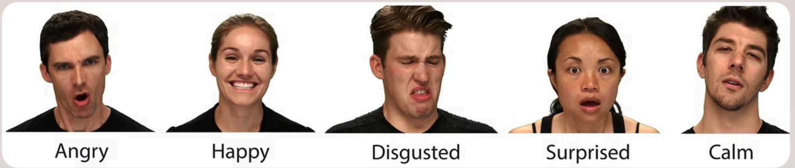
The actresses involved in the recordings are aged 26 and 64 years, providing a range of vocal characteristics and emotional expressions.

3 Audio Files

The dataset contains 2800 audio files in total, with each file representing a specific word spoken with a particular emotion. The audio files are in WAV format.

4 Dataset Organization

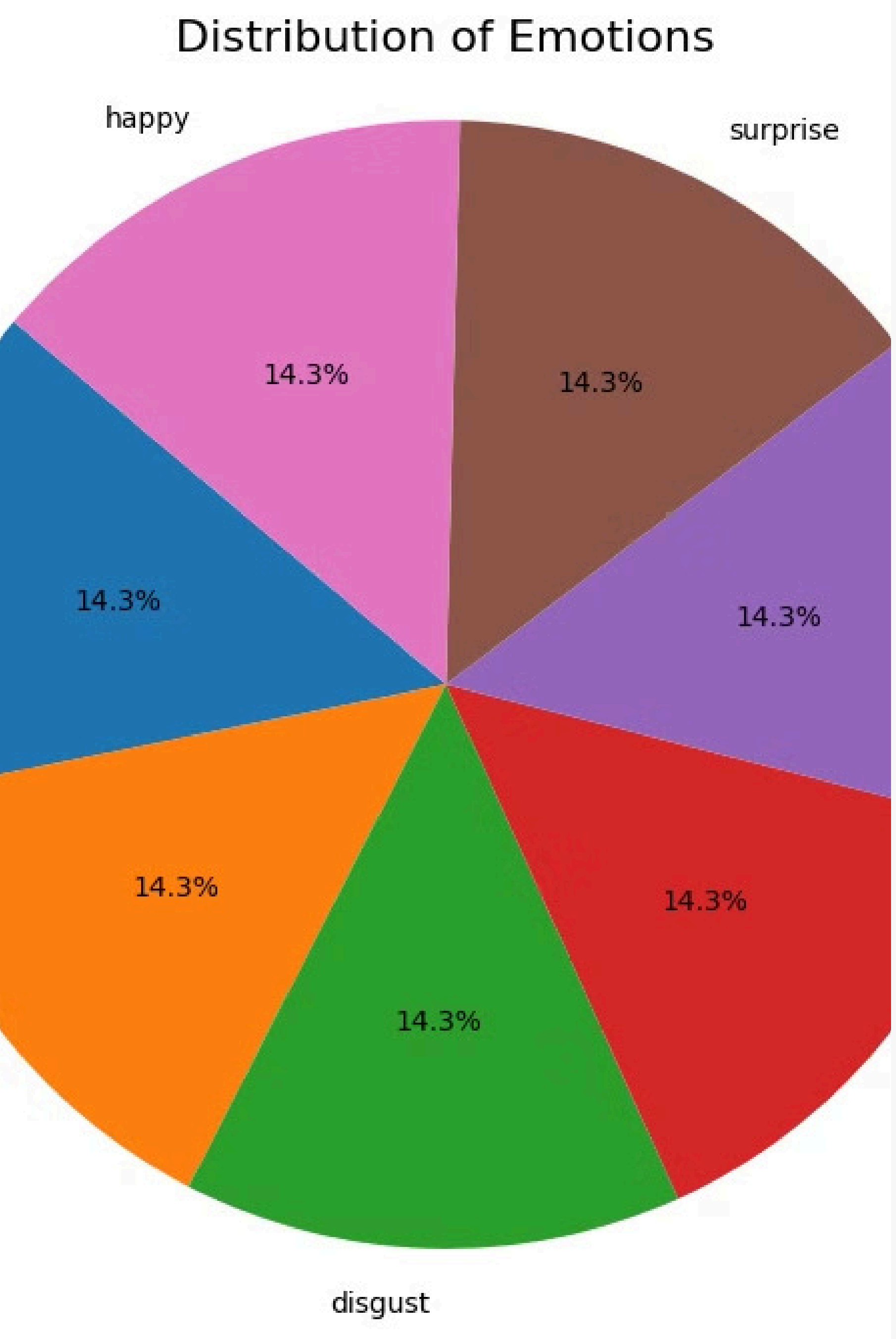
The dataset is organized into folders based on the actresses and their emotions. Each folder contains the audio files for the 200 target words spoken with the corresponding emotion.



Directory Traversal And Dataframe Creation For Emotion Data

This code segment focuses on traversing the dataset directory structure and creating a DataFrame to store information about each audio file. The code iterates through the folders, extracts emotion labels from filenames, and creates a DataFrame containing corresponding file paths and emotions.

```python
# List all directories in the base directory
tess_directory_list = os.listdir(Tess)

# Initialize lists to hold emotions and file paths
file_emotion = []
file_path = []

# Loop through each directory in the dataset
for dir in tess_directory_list:
    # Construct the full path to the current directory
    dir_path = os.path.join(Tess, dir)  # Use os.path.join for proper path construction
    if os.path.isdir(dir_path):  # Check if the path is a directory
        # List all files in the current directory
        directories = os.listdir(dir_path)
        for file in directories:
            # Extract the emotion part from the filename
            part = file.split('.')[0]  # Get the filename without extension
            part = part.split('_')[2]   # Split by underscore and take the emotion part

            # Map part to corresponding emotion
            if part == 'ps':
                file_emotion.append('surprise')  # Special case for 'ps'
            else:
                file_emotion.append(part)  # Append the emotion

            # Construct the full file path and append to the list
            file_path.append(os.path.join(dir_path, file))  # Use os.path.join for the file path

# Create a DataFrame for emotions
emotion_df = pd.DataFrame(file_emotion, columns=['Emotions'])

# Create a DataFrame for file paths
path_df = pd.DataFrame(file_path, columns=['Path'])

# Concatenate the two DataFrames to create a final DataFrame
Tess_df = pd.concat([emotion_df, path_df], axis=1)

# Display the first few rows of the final DataFrame
Tess_df.sample(10)
```



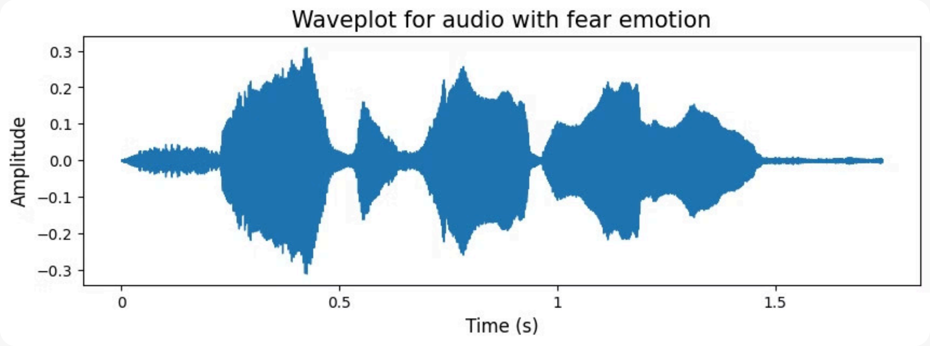
Visualize Emotions

This section involves visualizing the distribution of emotions in the dataset. The code counts the occurrences of each emotion and presents the results in a visual format, such as a bar chart or pie chart.

Waveplots And Spectrograms

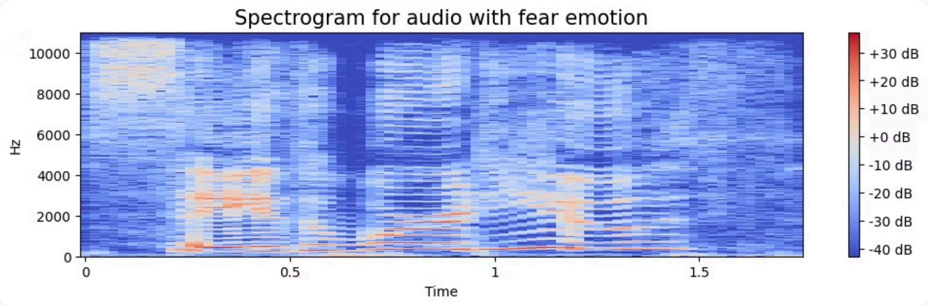
Waveplots

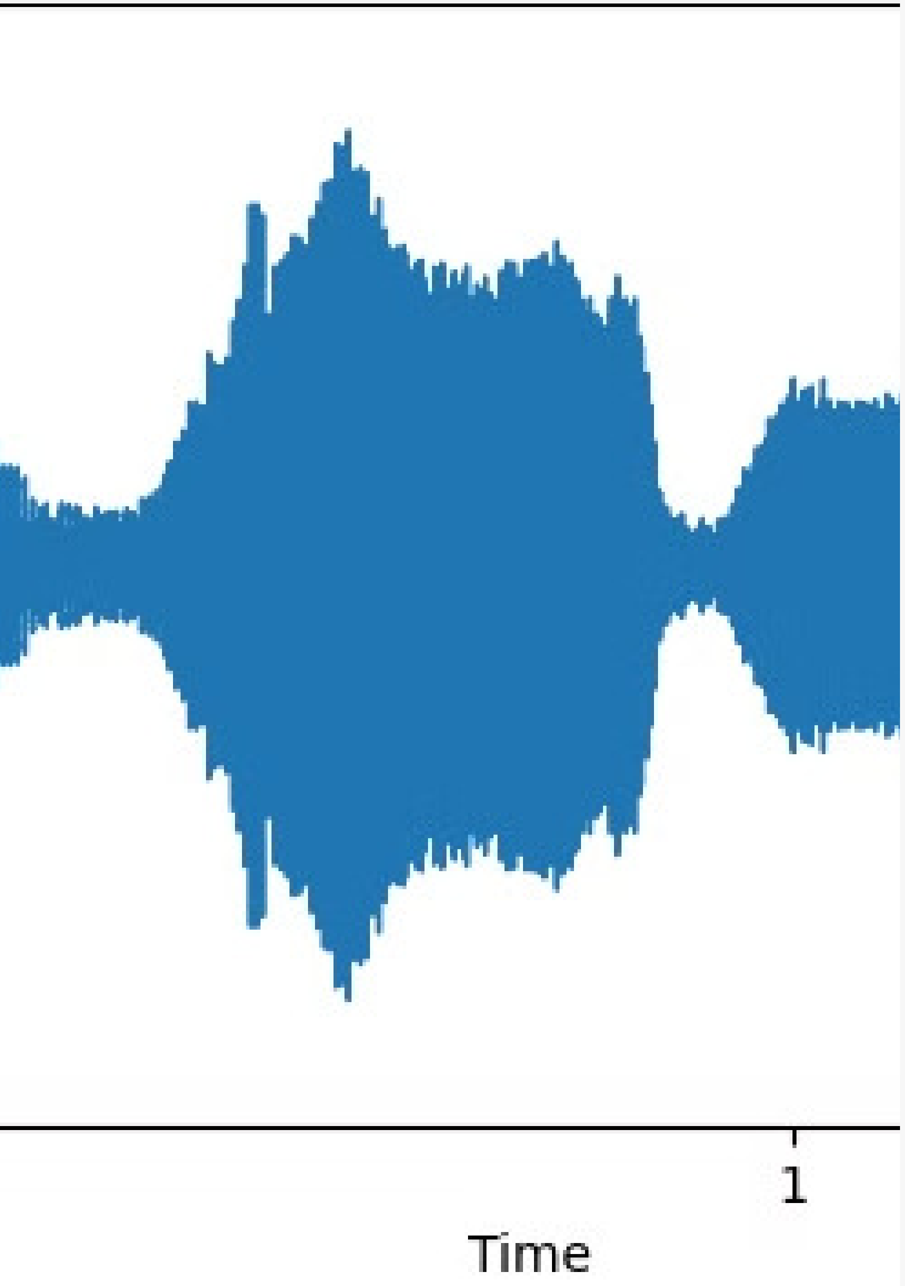
Waveplots provide a visual representation of the loudness of audio signals over time. They are useful for understanding the overall structure and dynamics of audio recordings.



Spectrograms

Spectrograms visually represent the frequency content of audio signals over time. They are valuable for analyzing the spectral characteristics of sounds and identifying different frequencies present in the audio.





Data Augmentation

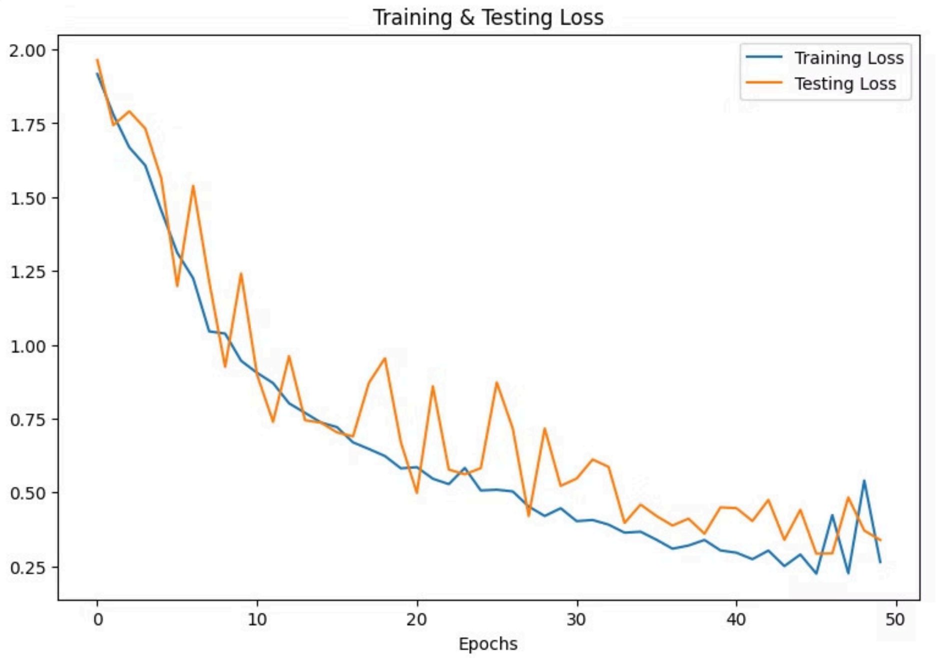
Data augmentation techniques are employed to create synthetic data samples by introducing variations to the original recordings. This helps improve the model's robustness and generalization ability by exposing it to a wider range of audio variations.

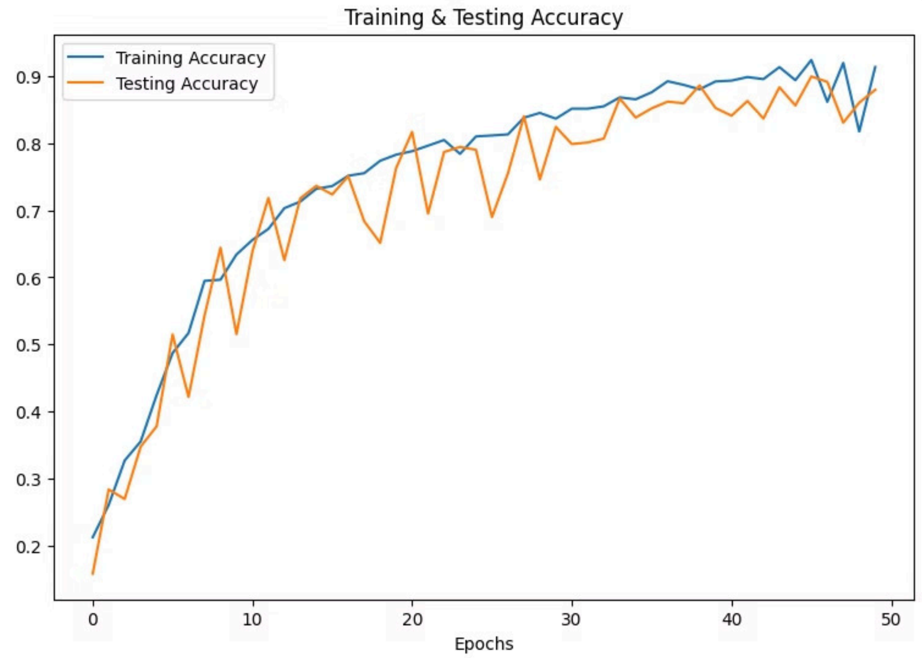
Feature Extraction

Feature extraction involves extracting relevant information from the audio data. This project focuses on extracting five key features: zero crossing rate, chroma STFT, MFCCs, RMS value, and Mel spectrogram. These features capture different aspects of the audio signal and provide valuable information for emotion recognition.

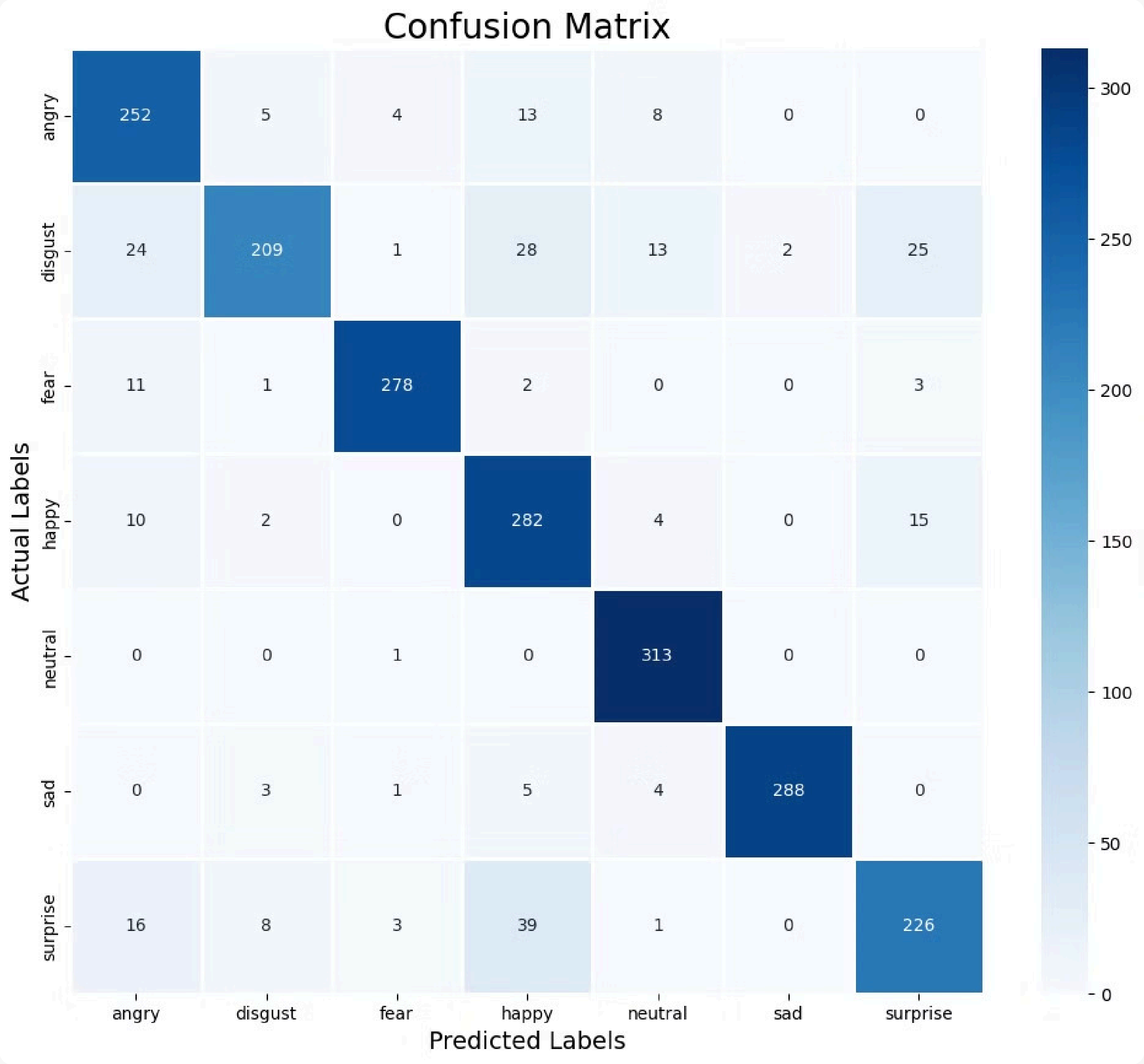
Lstm Model Training

We trained Lstm model and achived 88 % Accuracy.





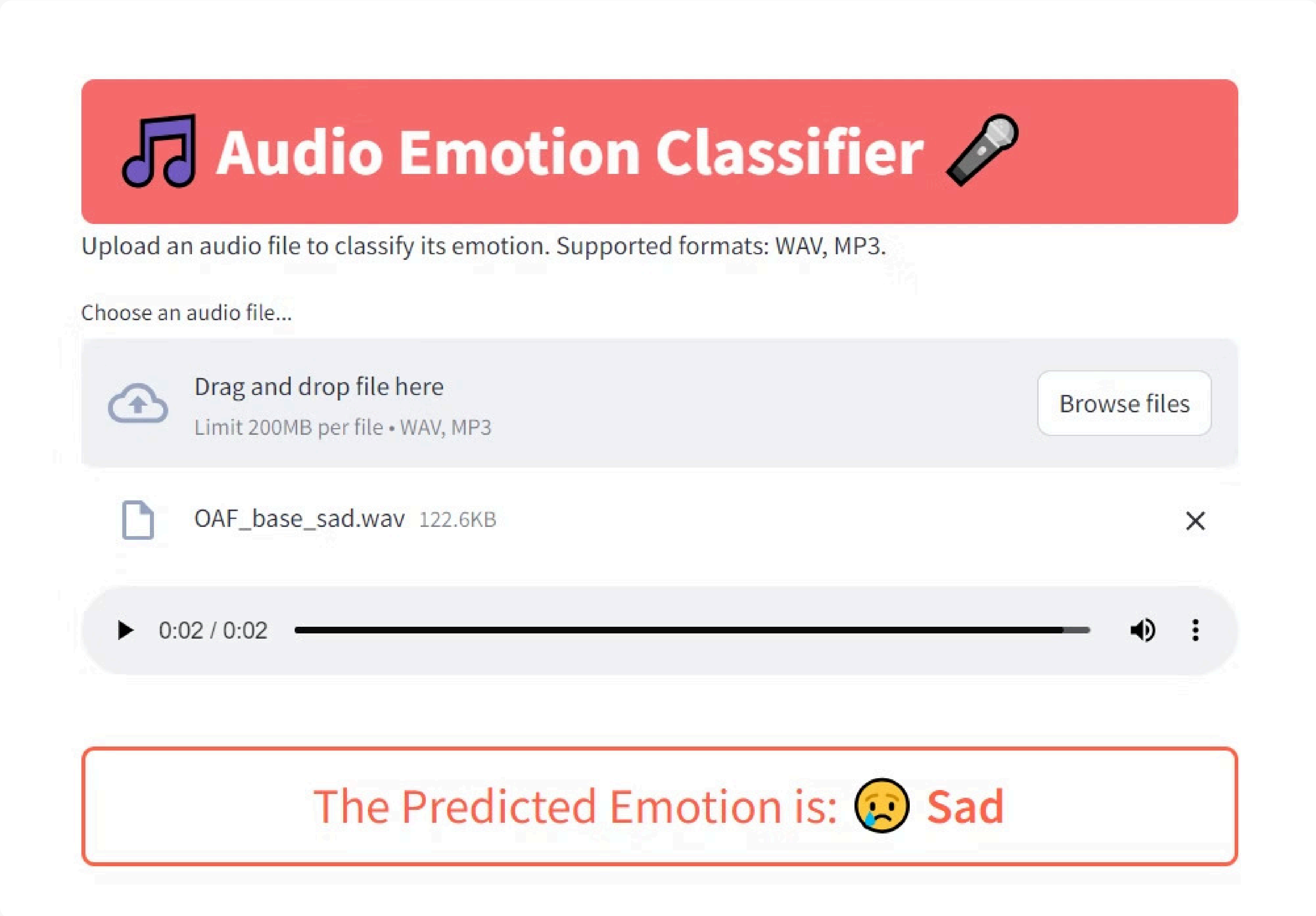
Compute the confusion matrix using the actual and predicted labels



Save the Model For Emotion Recognition from Speech

```
# Save the model
model.save('lstm_model.h5')  # Save as HDF5 file
```

Make a Streamlit Web App For Emotion Recognition Using this model



Final Thoughts

- The LSTM model outperforms the SimpleRNN on this dataset.
- With an accuracy of **88%** on the test data, there's potential for improvement through hyperparameter tuning.
- This experimentation is just the beginning; consider exploring different features for further enhancements.