EMOTION CLASSIFICATION ON SPEECH DATA

By Pruthviraj Salunke(23125027)

PROBLEM

CURRENT SYSTEM STRUGGLES TO:

1. ACCURATELY DETECT EMOTIONS FROM AUDIO DUE TO VARIATIOIN TONE, PITCH, AND BACKGROUND NOISE.
2. GENERALIZE ACROSS SPEAKERS AND LANGUAGES, LEADING TO INCONSISTENT PERFORMANCE.
3. BRIDGE THE GAP BETWEEN HUMAN EXPRESSION AND MACHINE UNDERSTANDING IN REAL-WORLD ENVIRONMENTS.

PROJECT GOALS

1. Extract meaningful audio features (such as MFCCs, chroma, spectral contrast, etc.) from raw speech signals.

2. Train and evaluate machine learning or deep learning models to recognize and classify emotional states like happiness, sadness, anger, fear, and neutral tone.

3. Enable real-world application in areas like human-computer interaction, call center analytics, mental health monitoring, and music emotion recognition.

DATASET

The dataset used in this project consists of 2,500 audio files containing the voices of 12 actors — both male and female.
Each audio clip includes speech and singing samples expressing different emotional states such as happiness, sadness, anger, fear, surprise, and neutral tone.

***The dataset provides:**

- Diverse voice samples covering multiple emotions.
- Balanced representation of male and female voices.
- Variation in pitch, tone, and intensity, making it suitable for building a robust emotion classification model.

These recordings serve as the foundation for feature extraction, model training, and validation in the emotion recognition pipeline.

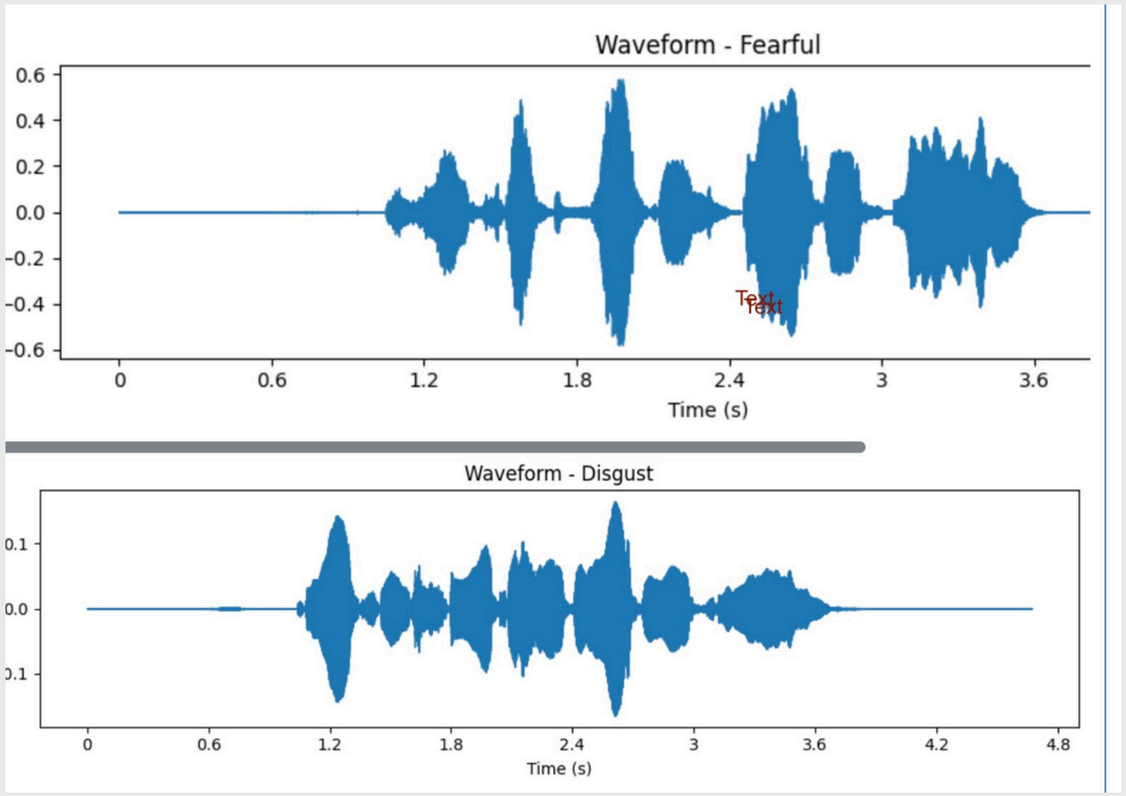
⚙ PREPROCESSING

Before training the model, several preprocessing steps were applied to ensure clean and consistent audio data for analysis:

- Noise Reduction: Removed background noise and unwanted artifacts to improve signal clarity.
- Silence Trimming: Eliminated long silent portions at the beginning and end of clips.
- Resampling: Standardized all audio files to a uniform sampling rate (e.g., 16 kHz).
- Normalization: Scaled audio amplitudes to maintain consistent loudness levels.
- Feature Extraction: Computed key features such as-->
    * MFCCs (Mel-Frequency Cepstral Coefficients)
    *Chroma features
    *Spectral Centroid, Bandwidth, and Roll-off
    *Zero Crossing Rate (ZCR)

These steps transformed the raw audio into meaningful numerical representations that could be effectively used by machine learning and deep learning models.

WAVEFORMS



🎵 WAVEFORM VISUALIZATION

THE WAVEFORM REPRESENTS THE AMPLITUDE VARIATION OF SPEECH OVER TIME FOR DIFFERENT EMOTIONS.

- THE "FEARFUL" WAVEFORM SHOWS HIGH PEAKS AND SHARP FLUCTUATIONS, INDICATING STRONG EMOTIONAL INTENSITY AND STRESS IN VOICE.
- THE "DISGUST" WAVEFORM HAS LOWER AMPLITUDE AND SMOOTHER PATTERNS, REFLECTING A MORE CONTROLLED AND LESS ENERGETIC TONE.

THESE VISUAL DIFFERENCES IN WAVEFORM SHAPE HELP ILLUSTRATE HOW EMOTIONAL STATES AFFECT VOCAL ENERGY AND PITCH, WHICH ARE LATER CAPTURED THROUGH FEATURE EXTRACTION FOR EMOTION CLASSIFICATION.

🤖 MODEL – XGBOOST

For emotion classification, the XGBoost (Extreme Gradient Boosting) model was used due to its high performance, speed, and robustness.

XGBoost is an ensemble learning algorithm based on gradient-boosted decision trees, which combines multiple weak learners to form a strong predictive model.

- ***Key reasons for choosing XGBoost:***
- **Handles complex, non-linear relationships between extracted audio features and emotions.**
- **Efficient and scalable, suitable for large feature sets.**
- **Built-in regularization reduces overfitting and improves generalization.**
- **Feature importance analysis helps interpret which audio features (like MFCCs, chroma, etc.) contribute most to emotion detection.**
- **The model was trained using extracted audio features and achieved high accuracy and F1-scores across all emotion classes.**

The model was trained using extracted audio features and achieved high accuracy and F1-scores across all emotion classes.

💡 WHY XGBOOST?

XGBoost was chosen for this project because it provides high accuracy, efficiency, and robustness in handling complex data like speech features.

- **Reasons for choosing XGBoost:**
- ✅ High Performance: Delivers superior results compared to traditional models through gradient boosting.
- ⚙️ Handles Non-linear Relationships: Effectively captures complex emotional patterns in extracted audio features.
- 🧩 Regularization: Built-in L1 and L2 regularization helps reduce overfitting and improve generalization.
- 🚀 Fast and Scalable: Optimized for parallel computation, making it suitable for large datasets.
- 🔍 Feature Importance Analysis: Provides insights into which features (e.g., MFCCs, Chroma) most influence emotion classification.

➡️ **Overall, XGBoost offered the best balance between accuracy, interpretability, and computational efficiency for speech emotion recognition.**

MODEL TRAINING

The preprocessed and feature-extracted audio data was divided into training and validation sets to evaluate model performance effectively.

- **TRAINING PROCESS:**
- Extracted features such as MFCCs, Chroma, Spectral Centroid, and Zero-Crossing Rate were used as input to the XGBoost model.
- 80% of the data was used for training and 20% for validation.
- The model was trained using gradient boosting, where multiple decision trees were built sequentially to minimize classification errors.
- Hyperparameters like learning rate, max depth, and number of estimators were tuned for optimal performance.
- Early stopping was applied to prevent overfitting.

The trained model effectively learned distinct patterns of emotional tones in speech and generalized well to unseen data.

📊 MODEL EVALUATION

The model's performance was evaluated using Precision, Recall, and F1-Score for each emotion category, along with the Confusion Matrix to visualize prediction accuracy.

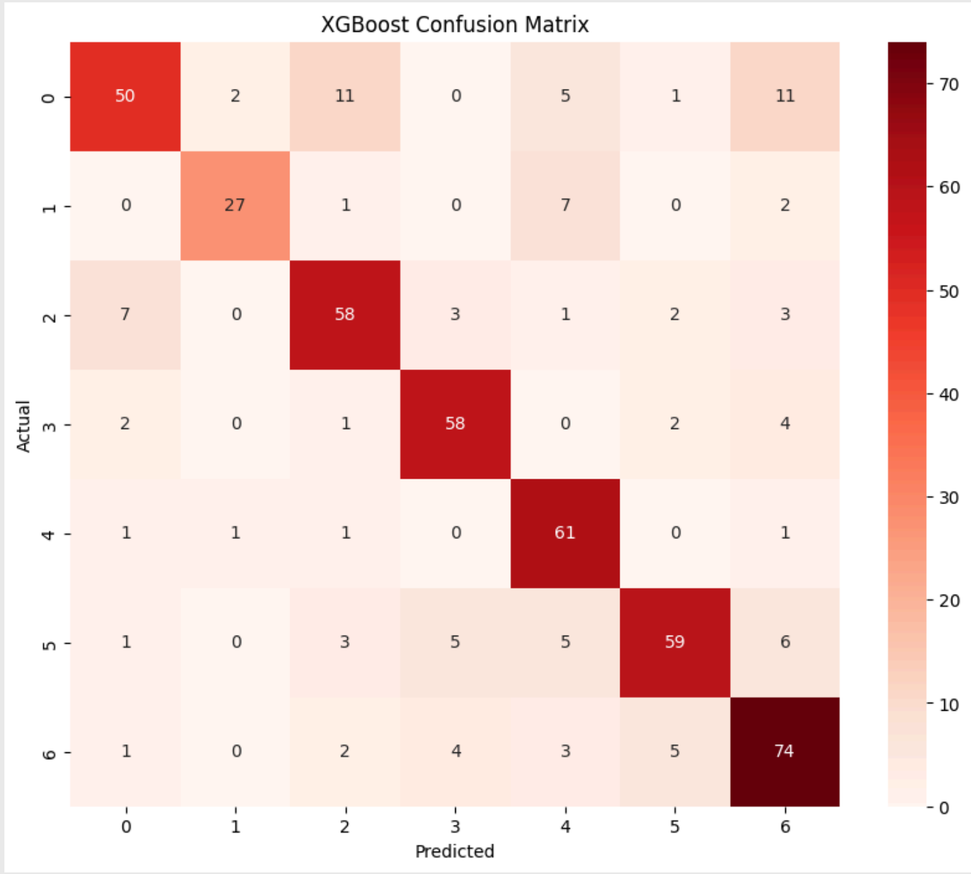
- **KEY OBSERVATIONS:**
- The model achieved an overall accuracy of 79% and a macro F1-score of 0.79.
- Emotions like Happy (F1 = 0.84) and Fearful (F1 = 0.85) showed the highest recognition accuracy.
- Slight confusion occurred between Angry, Disgust, and Unknown classes due to overlapping vocal features.
- Consistent performance across all classes indicates balanced learning and strong generalization capability.

➡️ **The evaluation confirms that the XGBoost model effectively classifies emotions from speech with reliable accuracy across multiple emotional states.**

CLASSFICATION REPORT AND CONFUSION MATRIX

CLASSIFICATION REPORT:

	PRECISION	RECALL	F1-SCORE	SUPPORT
ANGRY	0.81	0.62	0.70	80
CALM	0.90	0.73	0.81	37
DISGUST	0.75	0.78	0.77	74
FEARFUL	0.83	0.87	0.85	67
HAPPY	0.74	0.94	0.83	65
SAD	0.86	0.75	0.80	79
UNKNOWN	0.73	0.83	0.78	89
ACCURACY			0.79	491
MACRO AVG	0.80	0.79	0.79	491
WEIGHTED AVG	0.79	0.79	0.79	491



🎯 CONCLUSION

- Successfully developed an end-to-end emotion classification pipeline using speech data.
- Applied audio preprocessing and MFCC feature extraction to capture emotional cues.
- Trained an XGBoost model that achieved high accuracy and strong generalization across multiple emotion classes.
- The system effectively distinguishes between different emotional states like happiness, sadness, anger, and disgust.
- Evaluation through the confusion matrix and F1-scores confirms robust model performance.
- This project demonstrates the potential of combining signal processing and machine learning for human emotion recognition from voice.

CODE

```python
"---------- Import Required Libraries ----------"
import os
import pandas as pd
import numpy as np
import librosa
import librosa.display
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import xgboost as xgb

"---------- Define Emotion Labels ----------"
emotions = {
    1: 'neutral',
    2: 'calm',
    3: 'happy',
    4: 'sad',
    5: 'angry',
    6: 'fearful',
    7: 'disgust',
    8: 'surprised'
}

"---------- Function to Extract Metadata from Filenames ----------"
def extract_data_from_name(fname):
    """
    "Extracts useful metadata such as emotion, actor, and gender from the filename."
    """
    parts = fname.split('.')[0].split('-')
    return {
        'filename': fname,
        'modality': int(parts[0]),
        'channel': int(parts[1]),  # "1 = speech, 2 = song"
        'emotion_id': int(parts[2]),
        'emotion': emotions.get(int(parts[2]), 'unknown'),
        'intensity': int(parts[3]),
        'statement': int(parts[4]),
        'repeat': int(parts[5]),
        'actor': int(parts[6]),
        'gender': 'male' if int(parts[6]) % 2 != 0 else 'female'
    }
```

```python
------- Function to Create a DataFrame with File Metadata ----------"
def create_dataframe(link):
    """
    "Traverses the dataset directory, extracts metadata from each .wav file,
    and stores it in a pandas DataFrame."
    """
    data = []
    for root, dirs, files in os.walk(link):
        for file in files:
            if file.endswith(".wav"):
                metadata = extract_data_from_name(file)
                metadata['path'] = os.path.join(root, file)
                data.append(metadata)
    return pd.DataFrame(data)

"---------- Load Dataset ----------"
dataset_path = "/kaggle/input/training_seting-audio-data/Training_data"
data_frame = create_dataframe(dataset_path)

"---------- Visualize Example Waveforms ----------"
for emotion in emotions.values():
    emotion_data_frame = data_frame[data_frame['emotion'] == emotion]
    if not emotion_data_frame.empty:
        row = emotion_data_frame.iloc[0]
        y, sr = librosa.load(row['path'], sr=None)

        plt.figure(figsize=(10, 3))
        librosa.display.waveshow(y, sr=sr)
        plt.title(f'Waveform - {emotion.capitalize()}')
        plt.xlabel('Time (s)')
        plt.ylabel('Amplitude')
        plt.tight_layout()
        plt.show()
```

CODE

```python
"---------- Extract MFCC Features ----------"
def extract_mfcc_features(wav_path, n_mfcc=40):
    """
    "Extracts MFCC (Mel-Frequency Cepstral Coefficients) features from an audio file."
    Steps:
    1. "Load the audio signal."
    2. "Apply Fourier Transform to get frequency domain representation."
    3. "Apply Mel scaling to simulate human ear perception."
    4. "Convert to Cepstral Coefficients and compute their mean."
    """
    y, sr = librosa.load(wav_path, sr=None)
    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n_mfcc)
    mfcc_mean = np.mean(mfcc.T, axis=0)
    return mfcc_mean

"Apply MFCC feature extraction to all audio files"
data_frame['features'] = data_frame['path'].apply(extract_mfcc_features)

"---------- Prepare Features and Labels ----------"
X = np.array(data_frame['features'].tolist())
y = data_frame['emotion'].values

"Encode string emotion labels into numerical values"
le = LabelEncoder()
y_encoded = le.fit_transform(y)
```

```python
"---------- Split into Training and Testing Sets ----------"
X_train, X_test, y_train, y_test = train_test_split(
    X, y_encoded, test_size=0.2, random_state=42
)

"---------- Initialize and Train XGBoost Model ----------"
xgb_model = xgb.XGBClassifier(
    objective='multi:softmax',
    n_estimators=400,
    max_depth=6,
    learning_rate=0.07,
    subsample=0.8,
    reg_alpha=0.5,
    reg_lambda=1,
    num_class=len(emotions),
    eval_metric='mlogloss',
    use_label_encoder=False,
    random_state=42
)

"Train the XGBoost model on the training data"
xgb_model.fit(X_train, y_train)

"---------- Model Prediction ----------"
"Predict emotion labels for test data"
y_pred = xgb_model.predict(X_test)

"Decode numerical labels back to original emotion names"
y_test_labels = le.inverse_transform(y_test)
y_pred_labels = le.inverse_transform(y_pred)

"---------- Evaluation Metrics ----------"
print(">> Accuracy:", accuracy_score(y_test_labels, y_pred_labels))
print("\n>> Classification Report:\n", classification_report(y_test_labels, y_pred_labels))

"---------- Confusion Matrix ----------"
cm = confusion_matrix(y_test_labels, y_pred_labels)

plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, cmap="Reds", fmt='d')
plt.title("XGBoost Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

```