```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

```python
import os
import librosa
import librosa.display
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.utils import to_categorical
```

```python
data_path = "/content/drive/MyDrive/TESS Toronto emotional speech set
data"
emotions = os.listdir(data_path)
```

```python
emotions
```

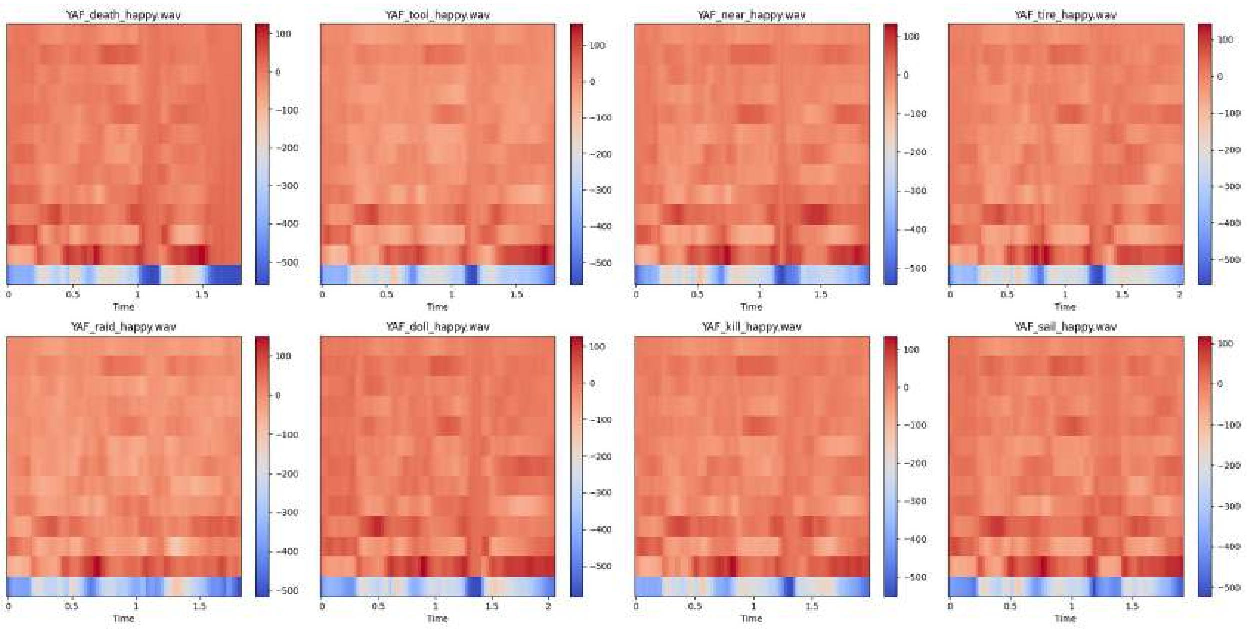
```
['TESS Toronto emotional speech set data',
 'YAF_sad',
 'YAF_neutral',
 'YAF_angry',
 'OAF_Pleasant_surprise',
 'YAF_pleasant_surprised',
 'YAF_happy',
 'YAF_disgust',
 'YAF_fear',
 'OAF_Sad',
 'OAF_disgust',
 'OAF_Fear',
 'OAF_angry',
 'OAF_happy',
 'OAF_neutral']
```

```python
from pathlib import Path

def get_audio_files(data_path, emotion):
    path = Path(data_path) / emotion
    return list(path.glob("*.wav"))


audio_files = get_audio_files(data_path, "YAF_angry")
print(f"Found {len(audio_files)} angry audio files")
```

```
Found 200 angry audio files
```

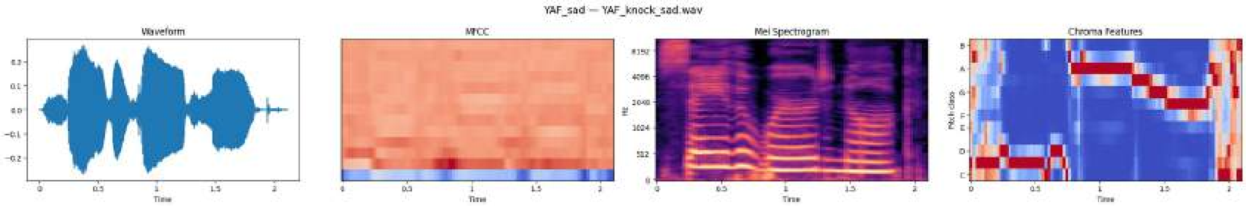
For Specific Column

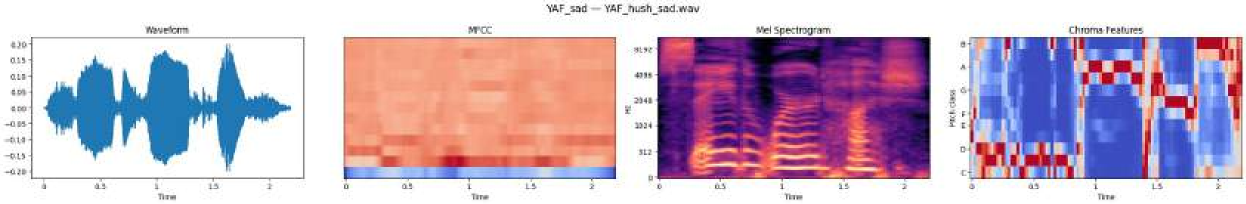
```python
def plot_mfcc_grid(emotion_folder, num_samples=8):
    files = os.listdir(os.path.join(data_path, emotion_folder))
[:num_samples]
    num_cols = 4
    num_rows = int(np.ceil(len(files) / num_cols))

    fig, axs = plt.subplots(num_rows, num_cols, figsize=(20, 5 *
num_rows))

    for idx, file in enumerate(files):
        file_path = os.path.join(data_path, emotion_folder, file)
        y, sr = librosa.load(file_path)
        mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)

        row, col = divmod(idx, num_cols)
        ax = axs[row][col] if num_rows > 1 else axs[col]
        img = librosa.display.specshow(mfcc, x_axis='time', ax=ax)
        ax.set_title(file)
        fig.colorbar(img, ax=ax)

    # Hide empty subplots
    for idx in range(len(files), num_rows * num_cols):
        row, col = divmod(idx, num_cols)
        ax = axs[row][col] if num_rows > 1 else axs[col]
        ax.axis('off')

    plt.tight_layout()
    plt.show()

# Example: plot MFCCs of 'YAF_happy'
plot_mfcc_grid("YAF_happy", num_samples=8)
```

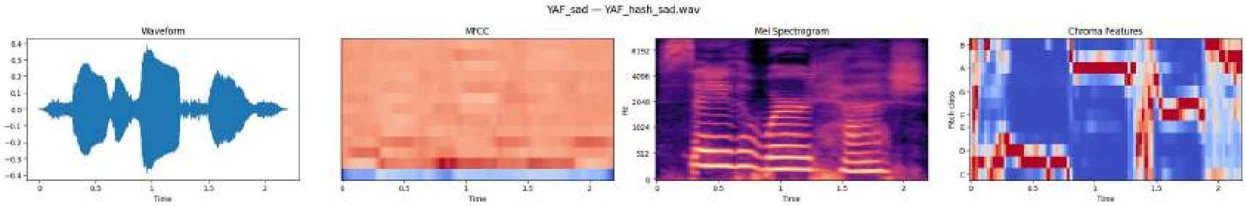


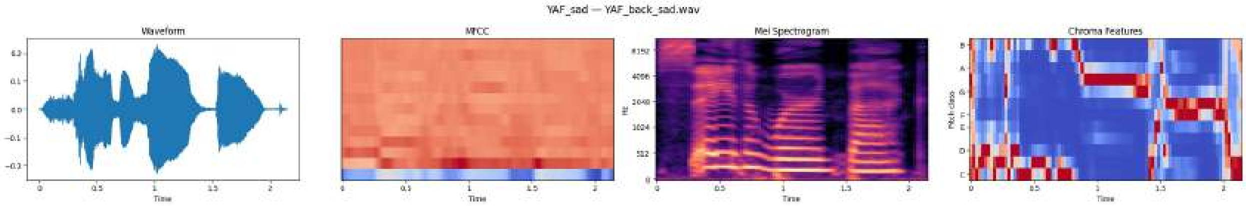
```
def plot_all_features_all_emotions(data_path, emotion_folders,
samples_per_emotion=4):
    for emotion_folder in emotion_folders:
        folder_path = os.path.join(data_path, emotion_folder)
        files = os.listdir(folder_path)[:samples_per_emotion]

        print(f" Emotion: {emotion_folder} – Showing {len(files)}
samples")

        for file in files:
            file_path = os.path.join(folder_path, file)
            y, sr = librosa.load(file_path, duration=3)

            fig, axs = plt.subplots(1, 4, figsize=(24, 4))
            fig.suptitle(f"{emotion_folder} – {file}", fontsize=14)

            # 1. Waveform
            librosa.display.waveshow(y, sr=sr, ax=axs[0])
            axs[0].set_title("Waveform")

            # 2. MFCC
            mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
            librosa.display.specshow(mfcc, x_axis='time', ax=axs[1])
            axs[1].set_title("MFCC")

            # 3. Log-Mel Spectrogram
            mel = librosa.feature.melspectrogram(y=y, sr=sr)
            mel_db = librosa.power_to_db(mel, ref=np.max)
            librosa.display.specshow(mel_db, x_axis='time',
y_axis='mel', sr=sr, ax=axs[2])
            axs[2].set_title("Mel Spectrogram")
            # 4. Chroma
            chroma = librosa.feature.chroma_stft(y=y, sr=sr)
            librosa.display.specshow(chroma, x_axis='time', 
y_axis='chroma', cmap='coolwarm', ax=axs[3])
            axs[3].set_title("Chroma Features")

            plt.tight_layout()
            plt.show()

# □ List all folders (excluding the master folder)
emotion_folders = [f for f in os.listdir(data_path) if not 
f.startswith('TESS') and os.path.isdir(os.path.join(data_path, f))]

# □ Run the full visualization
plot_all_features_all_emotions(data_path, emotion_folders, 
samples_per_emotion=4)
```

```
□ Emotion: YAF_sad — Showing 4 samples
```

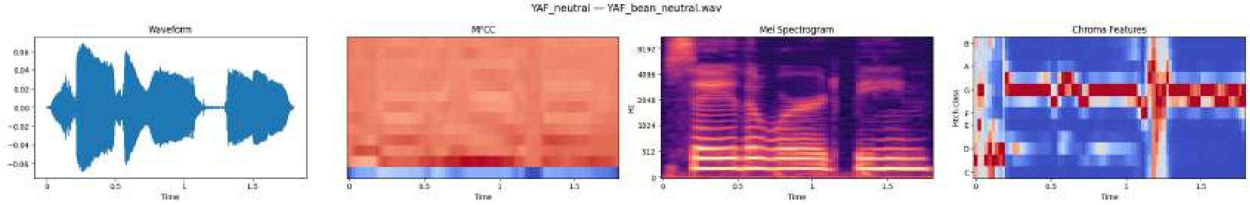


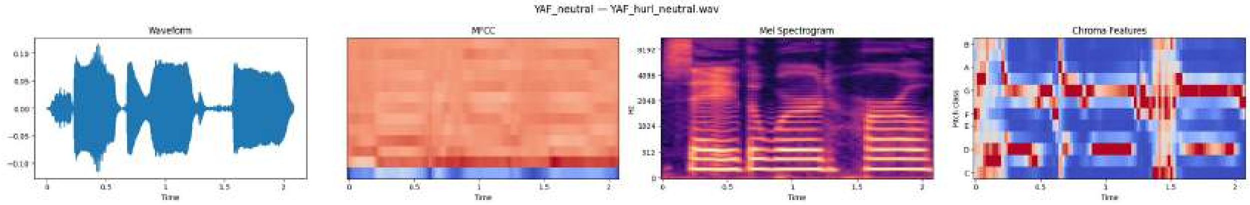


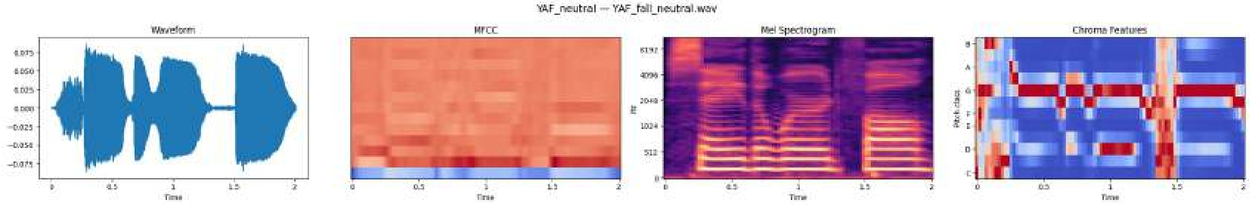


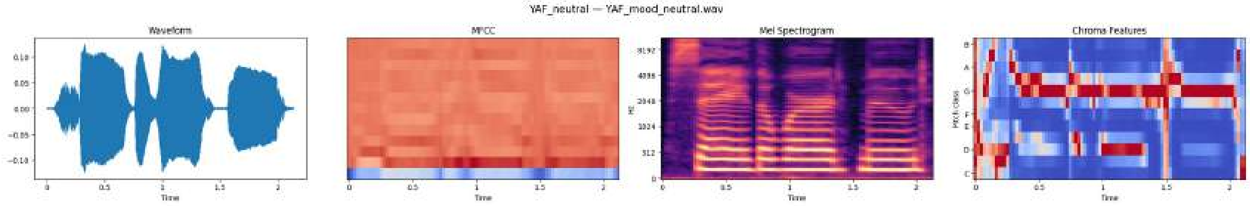


```
□ Emotion: YAF_neutral — Showing 4 samples
```

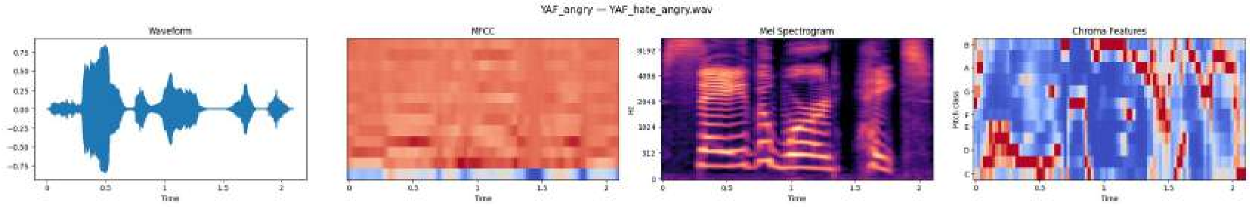


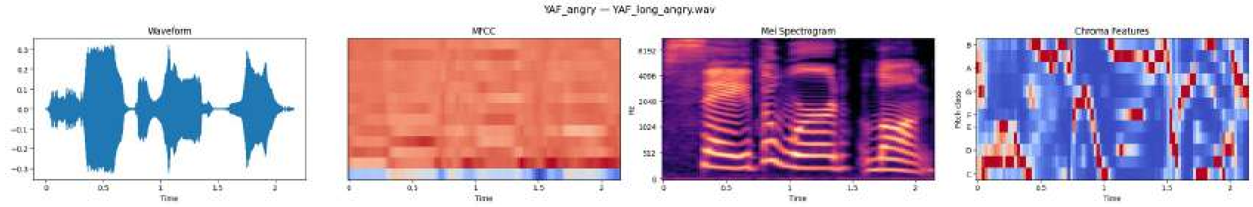


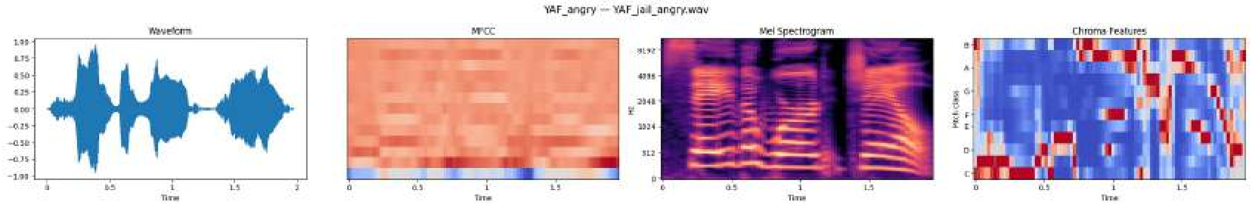


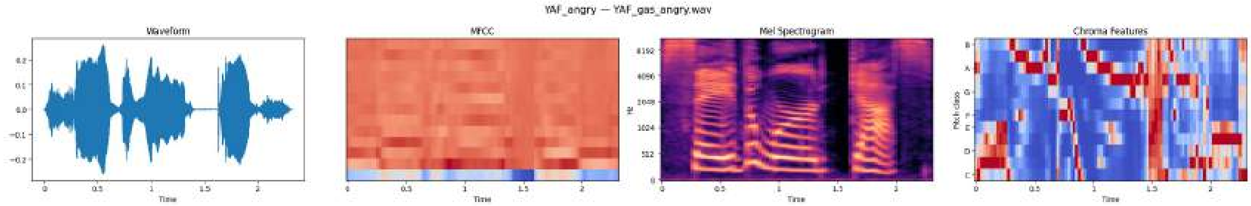


□ Emotion: YAF_angry — Showing 4 samples

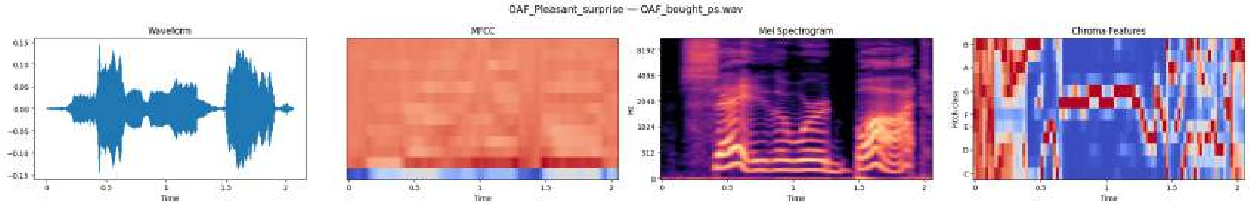


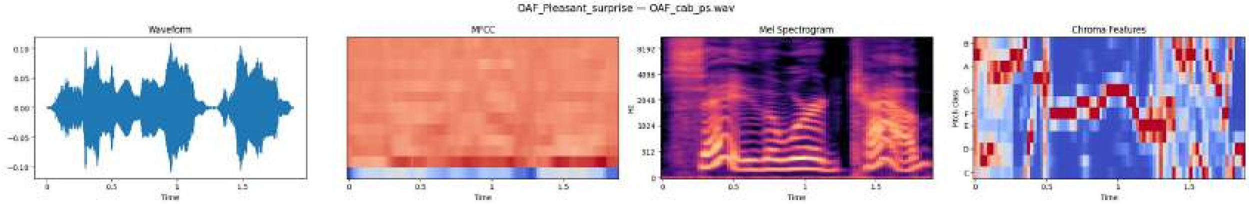


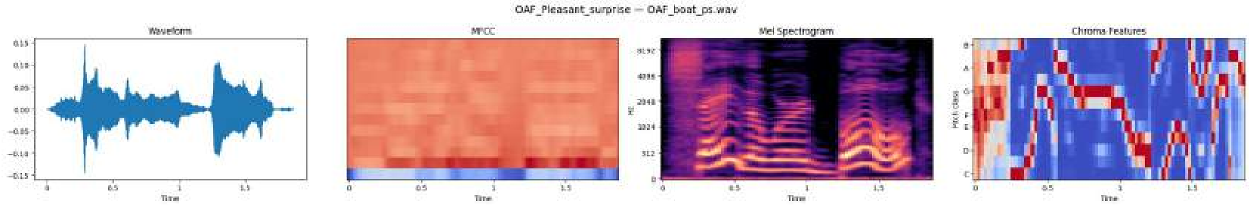


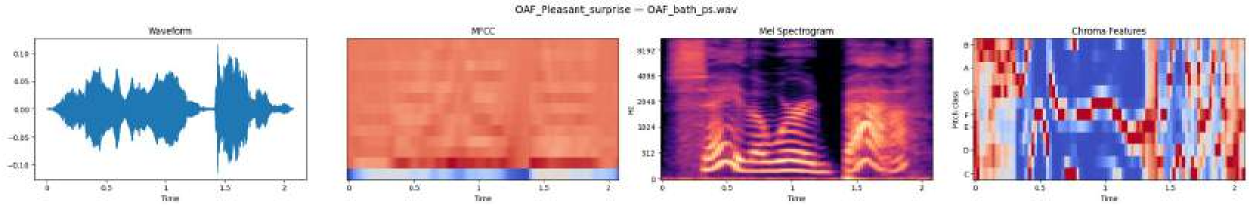


□ Emotion: OAF_Pleasant_surprise — Showing 4 samples

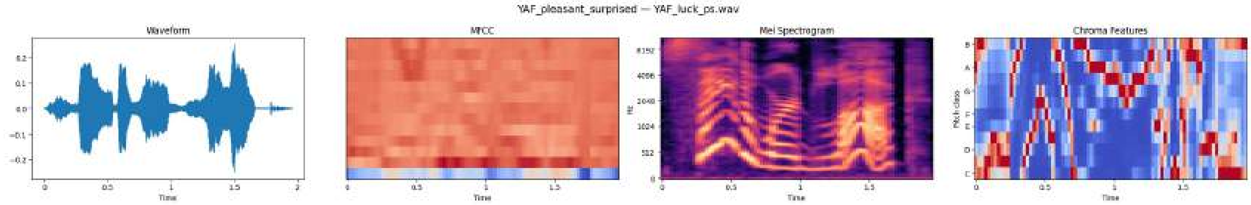




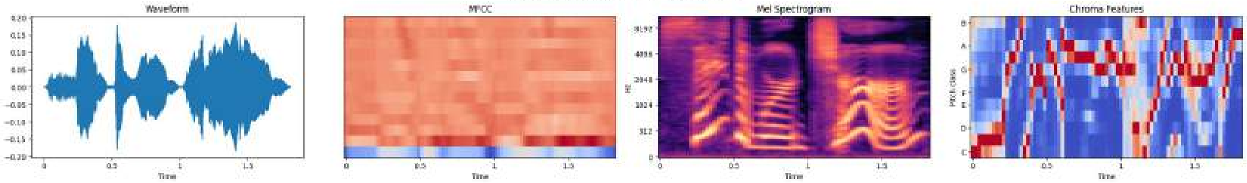




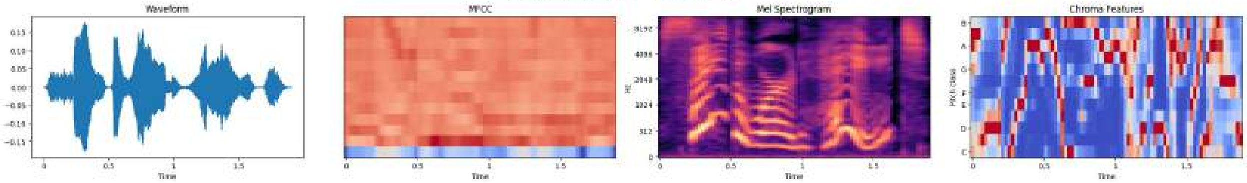
□ Emotion: YAF_pleasant_surprised — Showing 4 samples



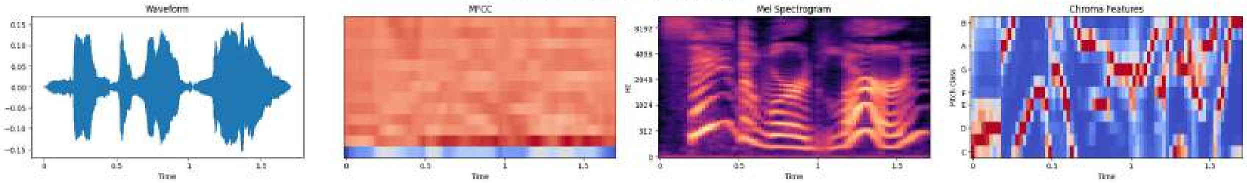
YAF_pleasant_surprised — YAF_germ_ps.wav



YAF_pleasant_surprised — YAF_hate_ps.wav

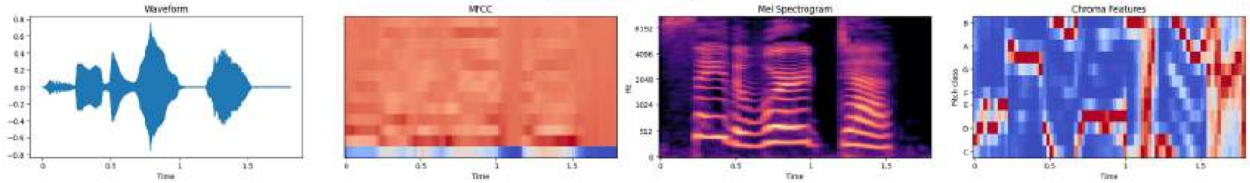


YAF_pleasant_surprised — YAF_learn_ps.wav

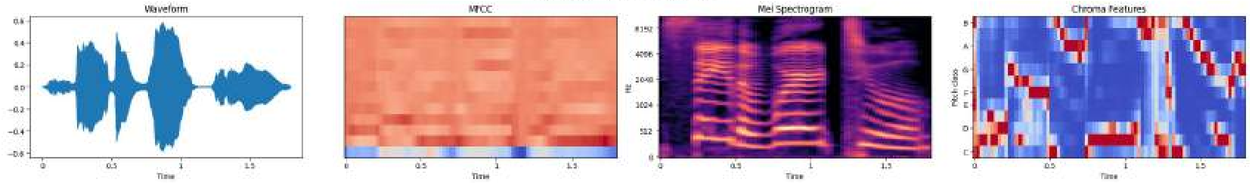


```
□ Emotion: YAF_happy – Showing 4 samples
```

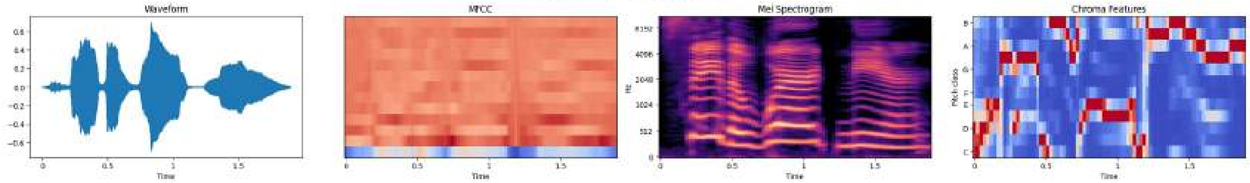
YAF_happy — YAF_death_happy.wav



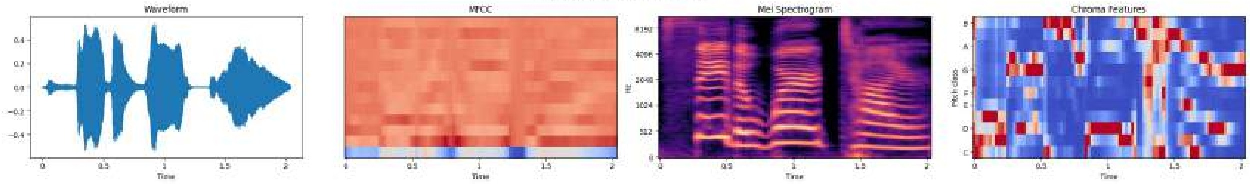
YAF_happy — YAF_tool_happy.wav



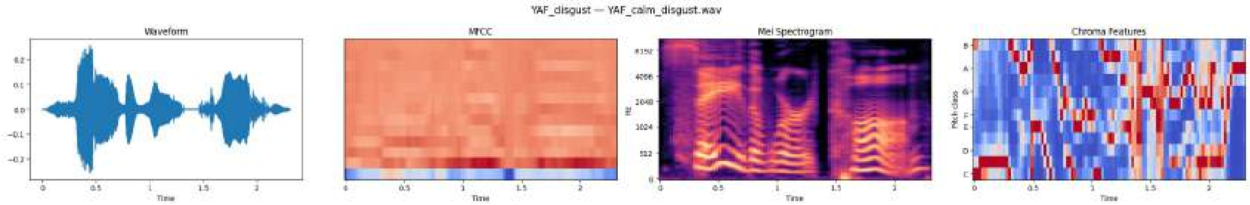
YAF_happy — YAF_near_happy.wav



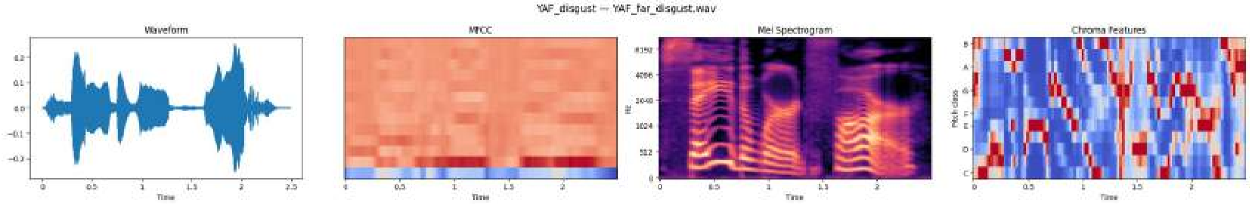
YAF_happy — YAF_tire_happy.wav



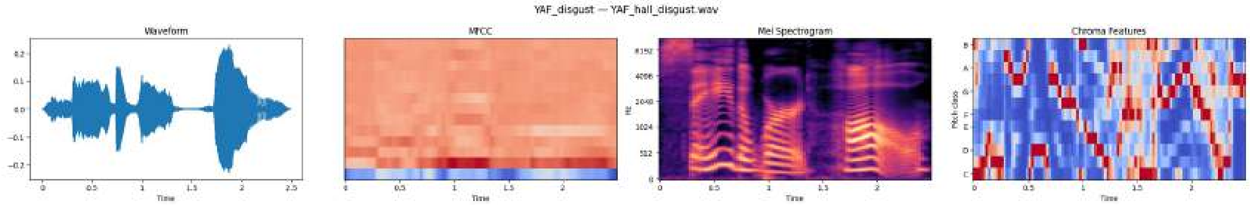
Emotion: YAF_disgust — Showing 4 samples


YAF_disgust — YAF_calm_disgust.wav

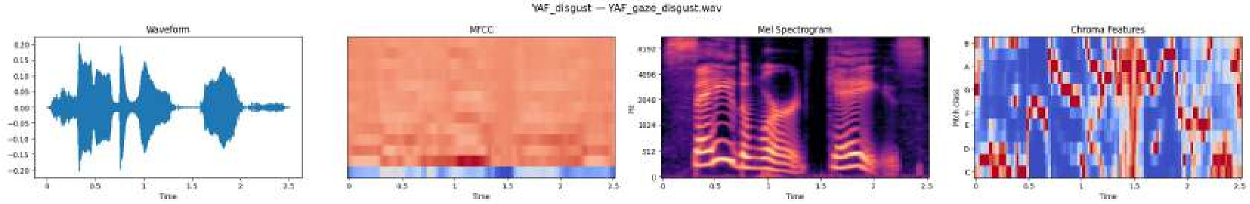
Waveform MFCC Mel Spectrogram Chroma Features


YAF_disgust — YAF_far_disgust.wav

Waveform MFCC Mel Spectrogram Chroma Features

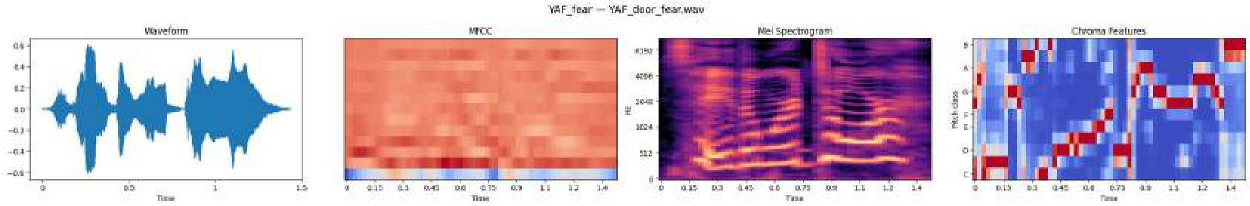

YAF_disgust — YAF_hall_disgust.wav

Waveform MFCC Mel Spectrogram Chroma Features

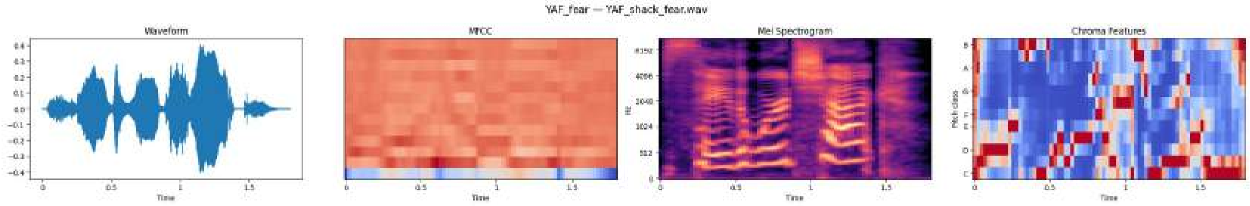

YAF_disgust — YAF_gaze_disgust.wav

Waveform MFCC Mel Spectrogram Chroma Features

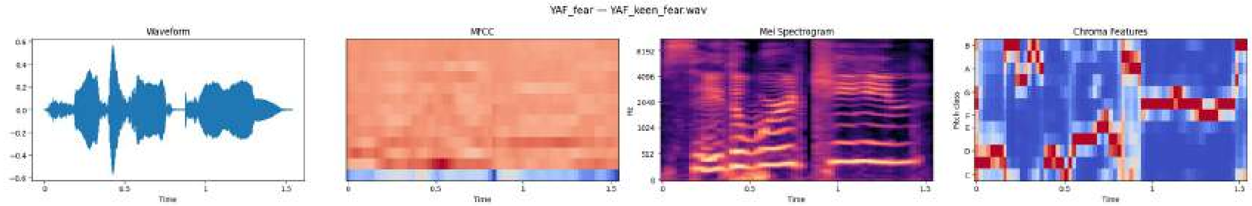
Emotion: YAF_fear — Showing 4 samples

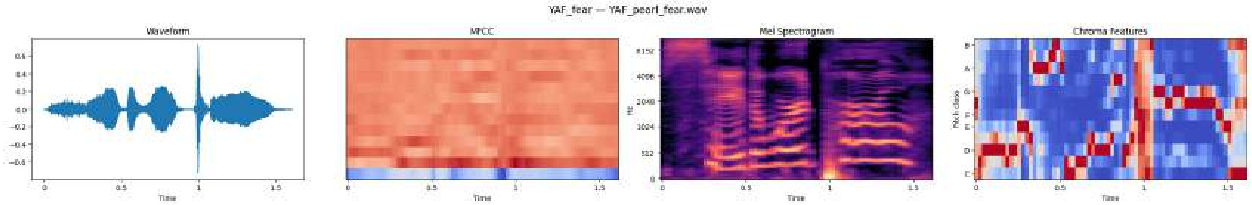

YAF_fear — YAF_door_fear.wav

Waveform MFCC Mel Spectrogram Chroma Features

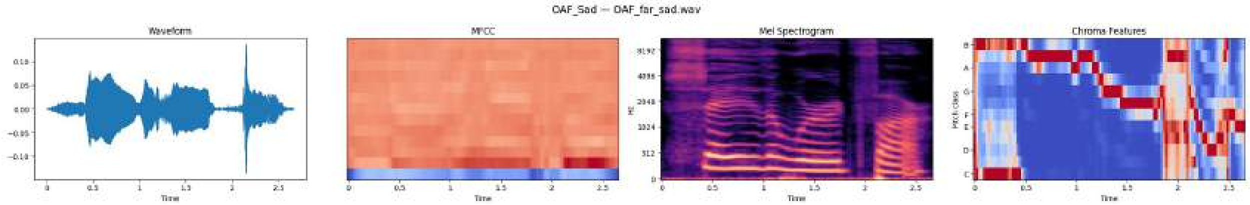

YAF_fear — YAF_shack_fear.wav

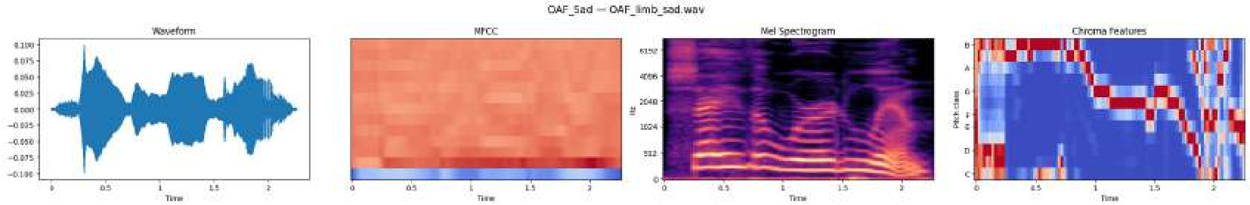
Waveform MFCC Mel Spectrogram Chroma Features

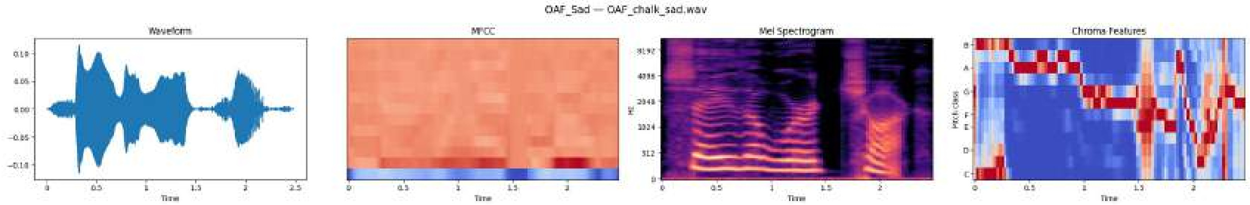


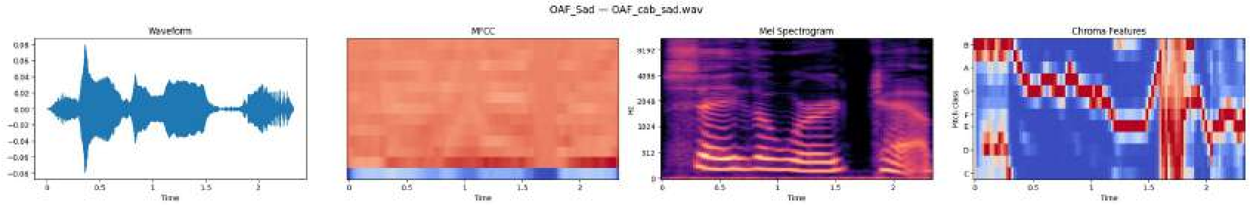


```
 Emotion: OAF_Sad — Showing 4 samples
```



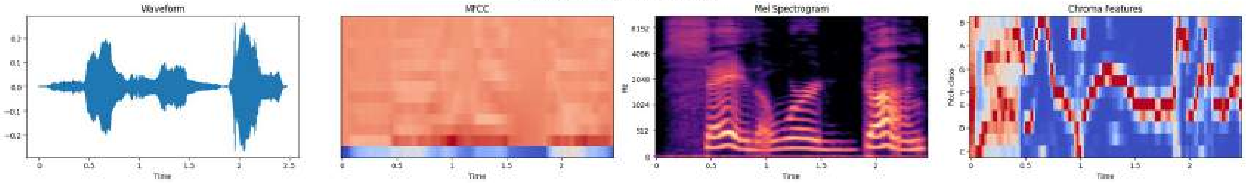




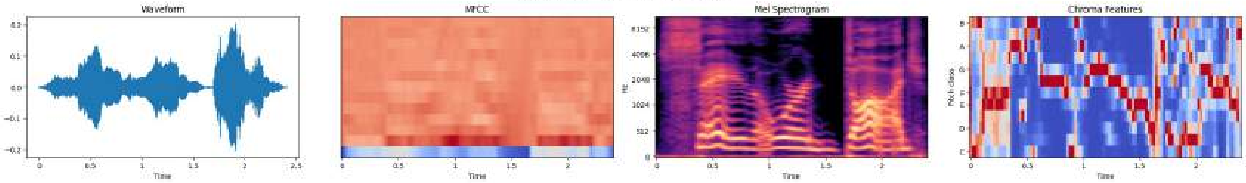


```
 Emotion: OAF_disgust — Showing 4 samples
```

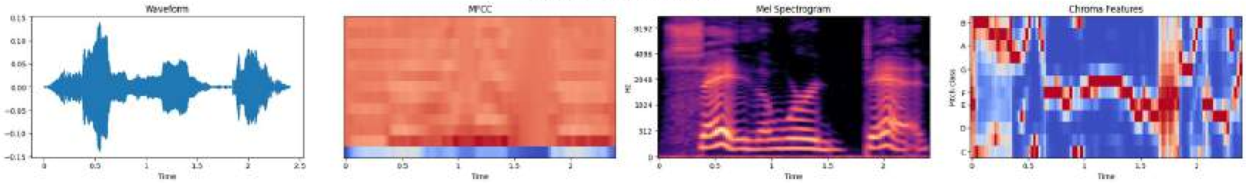
OAF_disgust — OAF_gun_disgust.wav



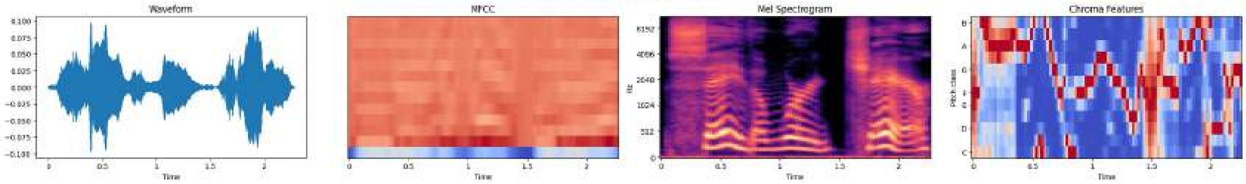
OAF_disgust — OAF_dodge_disgust.wav



OAF_disgust — OAF_dead_disgust.wav

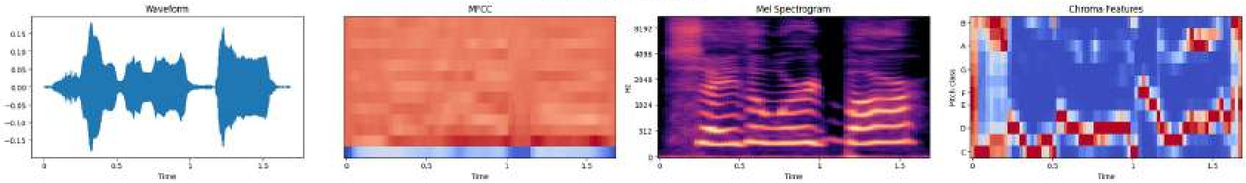


OAF_disgust — OAF_chair_disgust.wav

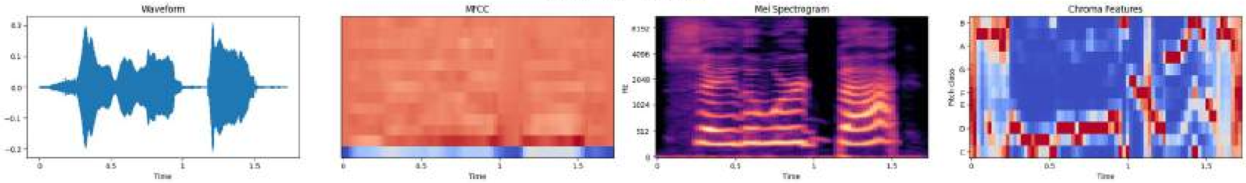


```
□ Emotion: OAF_Fear — Showing 4 samples
```

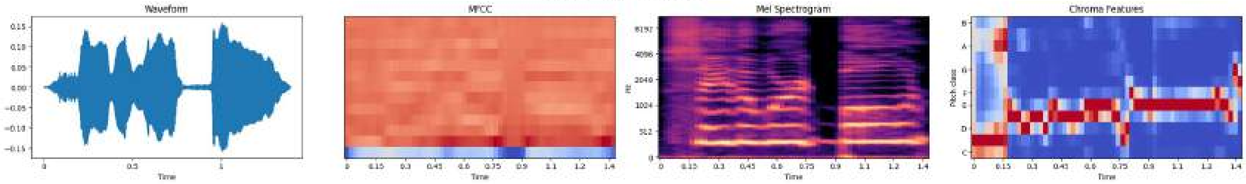
OAF_Fear — OAF_bar_fear.wav

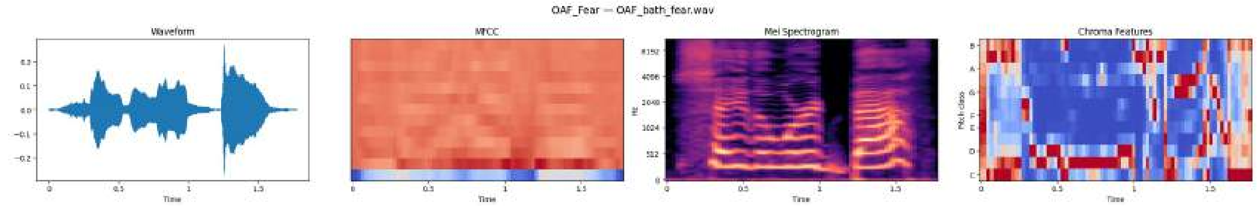


OAF_Fear — OAF_back_fear.wav

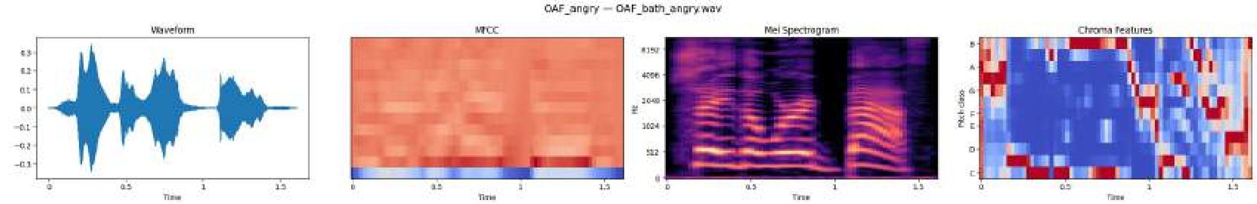


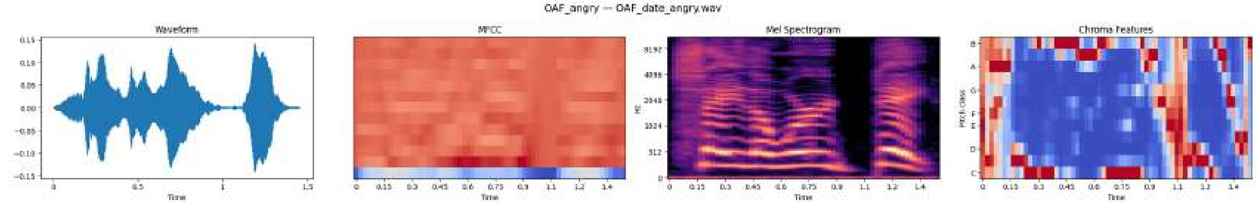
OAF_Fear — OAF_bone_fear.wav

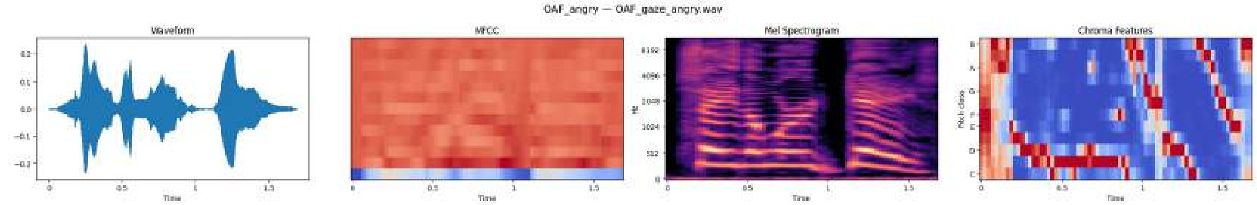


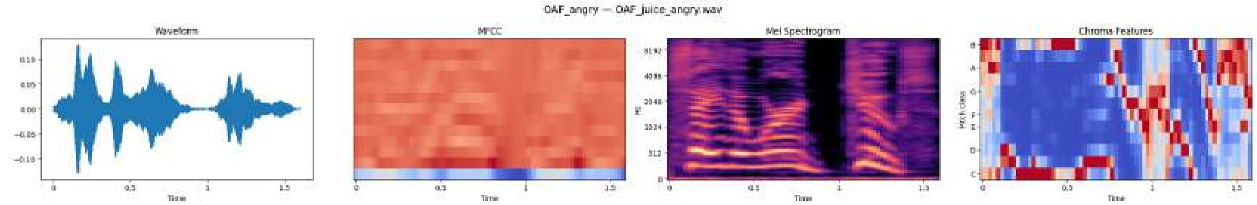


```
□ Emotion: OAF_angry — Showing 4 samples
```

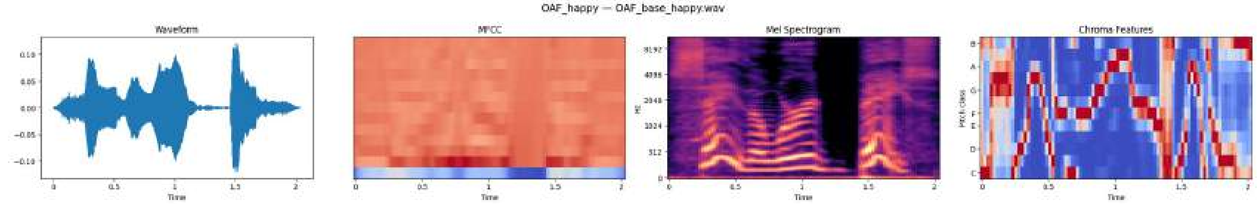




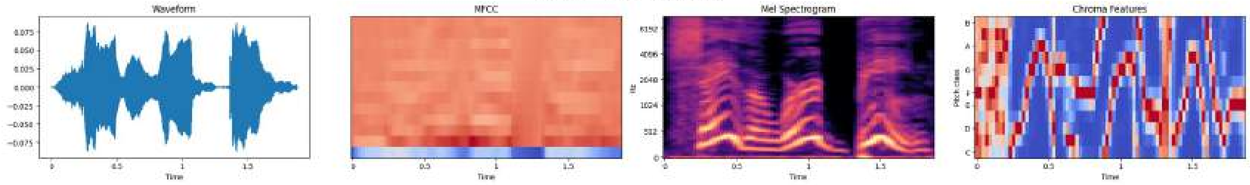




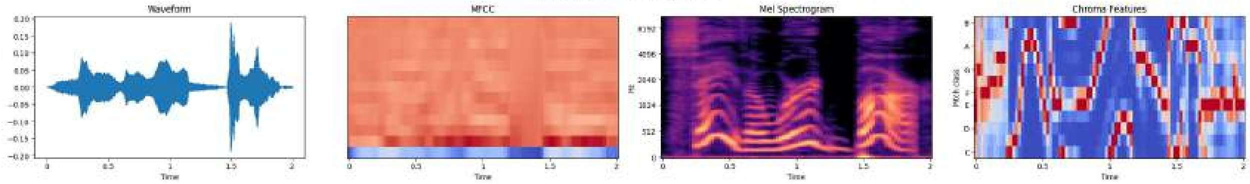
```
□ Emotion: OAF_happy — Showing 4 samples
```



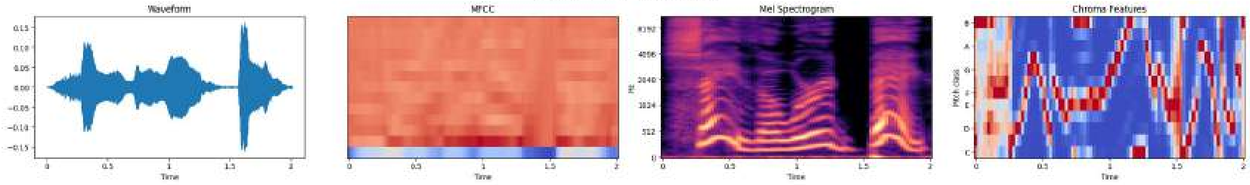
OAF_happy — OAF_bean_happy.wav



OAF_happy — OAF_bar_happy.wav

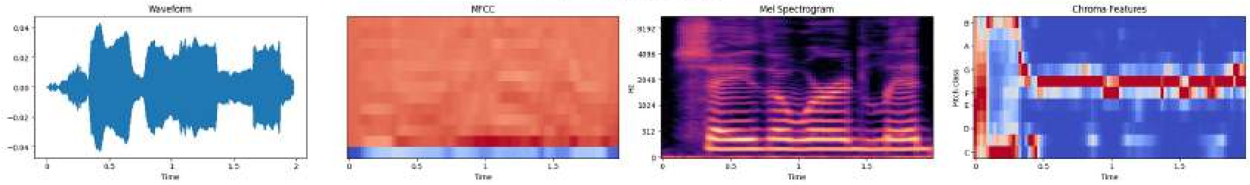


OAF_happy — OAF_beg_happy.wav

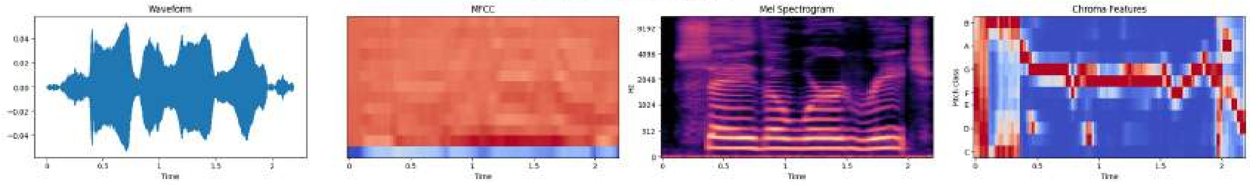


□ Emotion: OAF_neutral — Showing 4 samples

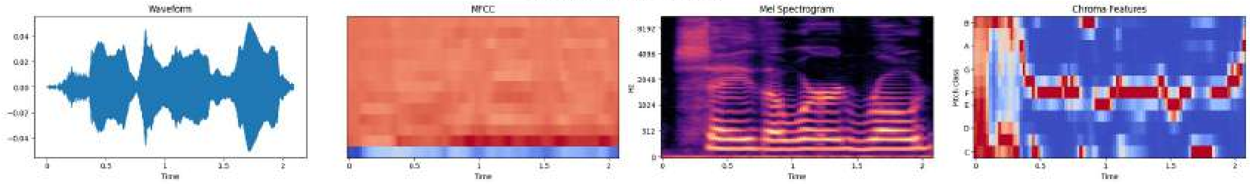
OAF_neutral — OAF_lid_neutral.wav



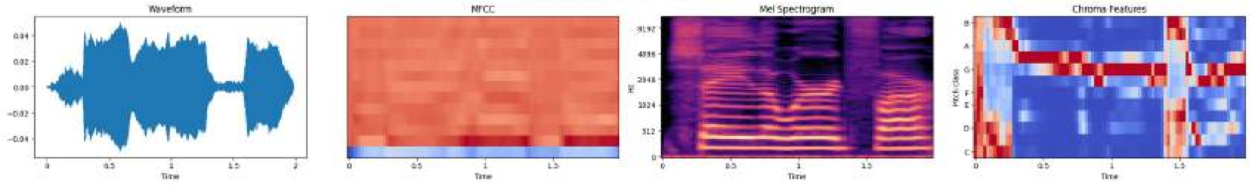
OAF_neutral — OAF_reach_neutral.wav



OAF_neutral — OAF_road_neutral.wav



OAF_neutral — OAF_far_neutral.wav



```python
def extract_features(file_path):
    y, sr = librosa.load(file_path, duration=3, offset=0.5)
    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40)
    mfcc_scaled = np.mean(mfcc.T, axis=0)
    return mfcc_scaled

features = []
labels = []

for folder in emotion_folders:
    emotion_label = folder.split('_')[-1].lower()
    folder_path = os.path.join(data_path, folder)

    for file in os.listdir(folder_path):
        file_path = os.path.join(folder_path, file)
        try:
            mfcc = extract_features(file_path)
            features.append(mfcc)
            labels.append(emotion_label)
        except Exception as e:
            print(f"Error: {file_path} – {e}")

from sklearn.preprocessing import LabelEncoder

X = np.array(features)
y = LabelEncoder().fit_transform(labels)

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42, stratify=y)

print("Train size:", len(X_train))
print("Test size:", len(X_test))
```

```
Train size: 2240
Test size: 560
```

Extract MFCC features and labels for all data

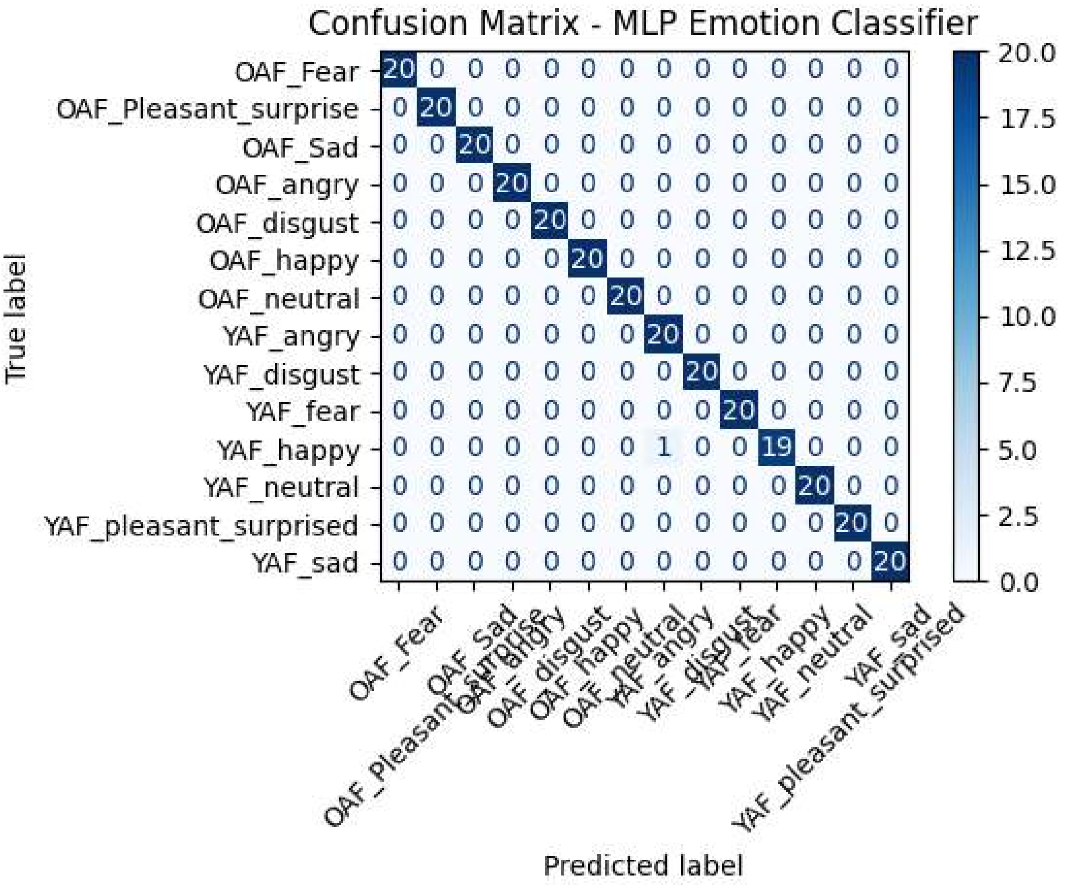
```python
import numpy as np
import os
import librosa
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

def extract_features_labels(data_path, emotion_folders,
max_files_per_emotion=100):
    X, y = [], []

    for emotion in emotion_folders:
        folder_path = os.path.join(data_path, emotion)
        files = os.listdir(folder_path)[:max_files_per_emotion]

        for file in files:
            file_path = os.path.join(folder_path, file)
            y_audio, sr = librosa.load(file_path, duration=3)
            mfcc = librosa.feature.mfcc(y=y_audio, sr=sr, n_mfcc=40)
            mfcc_scaled = np.mean(mfcc.T, axis=0)  # Take mean over
time axis

            X.append(mfcc_scaled)
            y.append(emotion)

    return np.array(X), np.array(y)

# Prepare data
emotion_folders = [f for f in os.listdir(data_path) if not
f.startswith('TESS') and os.path.isdir(os.path.join(data_path, f))]
X, y = extract_features_labels(data_path, emotion_folders,
max_files_per_emotion=100)

# Encode labels
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Split train-test
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded,
test_size=0.2, random_state=42, stratify=y_encoded)
```

Train a simple MLP classifier

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, confusion_matrix,
ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Initialize MLP
mlp = MLPClassifier(hidden_layer_sizes=(128, 64), max_iter=300,
random_state=42)

# Train
mlp.fit(X_train, y_train)

# Predict
y_pred = mlp.predict(X_test)

# Report
print(classification_report(y_test, y_pred, target_names=le.classes_))
```

```
                        precision    recall  f1-score   support

              OAF_Fear       1.00      1.00      1.00        20
 OAF_Pleasant_surprise       1.00      1.00      1.00        20
               OAF_Sad       1.00      1.00      1.00        20
             OAF_angry       1.00      1.00      1.00        20
           OAF_disgust       1.00      1.00      1.00        20
             OAF_happy       1.00      1.00      1.00        20
           OAF_neutral       1.00      1.00      1.00        20
             YAF_angry       0.95      1.00      0.98        20
           YAF_disgust       1.00      1.00      1.00        20
              YAF_fear       1.00      1.00      1.00        20
             YAF_happy       1.00      0.95      0.97        20
           YAF_neutral       1.00      1.00      1.00        20
YAF_pleasant_surprised       1.00      1.00      1.00        20
               YAF_sad       1.00      1.00      1.00        20

              accuracy                           1.00       280
             macro avg       1.00      1.00      1.00       280
          weighted avg       1.00      1.00      1.00       280
```

```
# Plot confusion matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=le.classes_)
disp.plot(xticks_rotation=45, cmap=plt.cm.Blues)
plt.title("Confusion Matrix - MLP Emotion Classifier")
plt.tight_layout()
plt.show()
```



```
def predict_emotion(file_path, model, label_encoder):
    y_audio, sr = librosa.load(file_path, duration=3)
    mfcc = librosa.feature.mfcc(y=y_audio, sr=sr, n_mfcc=40)
    mfcc_scaled = np.mean(mfcc.T, axis=0).reshape(1, -1)
    pred = model.predict(mfcc_scaled)
    emotion = label_encoder.inverse_transform(pred)
    return emotion[0]

new_audio_path = "/content/OAF_boat_happy.wav"
predicted_emotion = predict_emotion(new_audio_path, mlp, le)
print(f"Predicted emotion: {predicted_emotion}")

Predicted emotion: OAF_happy

new_audio_path = "/content/a.wav"
predicted_emotion = predict_emotion(new_audio_path, mlp, le)
print(f"Predicted emotion: {predicted_emotion}")

Predicted emotion: YAF_fear
```