FINAL PROJECT AUDIO PREPROCESSING DENGAN METODE SVM
PENGANTAR PEMROSESAN DATA MULTIMEDIA

OLEH:

KELOMPOK 1 KELAS C

Gary Melvin Lie	**2108561023**
Yasinta Anita Kewa Nilan	**2108561034**
I Putu Andi Wiratama Putra	**2108561052**

PROGAM STUDI INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS UDAYANA

BALI

2023

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam era digital saat ini, analisis sentimen atau emosi telah menjadi topik yang menarik dalam pengolahan bahasa alami. Dalam konteks aplikasi dan layanan berbasis suara, memiliki kemampuan untuk mengidentifikasi sentimen atau emosi dari suara orang dapat memberikan banyak manfaat, termasuk meningkatkan pengalaman pengguna, mendukung analisis pasar, dan memungkinkan aplikasi yang lebih cerdas.

Audio Preprocessing adalah proses memanipulasi dan mengolah data audio sebelum dilakukan analisis lebih lanjut. Audio Preprocessing merupakan tahap penting dalam pengolahan sinyal audio, karena membantu meningkatkan kualitas data audio dan mempersiapkannya untuk analisis lebih lanjut seperti pengenalan suara, pemrosesan ucapan, pengenalan musik, dan aplikasi lainnya.

Tujuan dari tugas ini adalah membangun sebuah sistem aplikasi yang mampu mengidentifikasi sentimen atau emosi dari suara orang. Terdapat dua kategori sentimen yang akan diidentifikasi, yaitu sentimen positif (happy) dan sentimen negatif (sad). Dalam sistem ini, tahap preprocessing data audio dilakukan dengan menggunakan metode MFCC (Mel-Frequency Cepstral Coefficients) untuk mengekstrak fitur-fitur penting dari data audio.

MFCC adalah salah satu metode yang umum digunakan dalam pengolahan suara dan pengenalan ucapan. Metode ini berfokus pada penggabungan informasi frekuensi dan spektral suara manusia dengan memodelkan respons pendengaran manusia terhadap berbagai frekuensi. Dengan menghitung koefisien cepstral pada domain frekuensi menggunakan MFCC, kita dapat mewakili karakteristik unik dari suara manusia yang berkaitan dengan sentimen atau emosi.

Dengan membangun sistem aplikasi ini,kita dapat mengidentifikasi sentimen atau emosi dari suara orang dengan cepat dan efisien. Hasil dari proyek ini dapat diterapkan dalam berbagai konteks, seperti pengenalan emosi dalam interaksi manusia-mesin, penilaian layanan pelanggan berbasis suara, atau pemantauan kesehatan mental melalui analisis suara.

1.2 Tujuan

Adapun tujuan dari tugas ini adalah sebagai berikut :

1. Menmbangun system aplikasi yang dapat mengidentifikasi sentiment atau emosi dari suara orang
2. Mengimplementasikan metode MFCC sebagai tahap preprocessing untuk ekstraksi fitur dari data audio.
3. Melatih model pengenalan sentiment atau emosi menggunakan dataset suara yang telah dikumpulkan dan diberi label sentimen atau emosi yang diketahui.
4. Menguji dan mengevaluiasi kinerja system aplikasi yang dibangun dengan menggunakan metrik evaluasi yang relevan, seperti akurasi, presisi, recall, dan F1-Score.

1.3 Manfaat

Adapun manfaat dari tugas ini adalah sebagai berikut :

1. Meningkatkan pemahaman tentang pengenalan sentimen atau emosi dari suara manusia.
2. Memeberikan wawasan yang lebih baik dalam analisis pasar.
3. Membuka potensi pengembangan aplikasi yang lebih lanjut
4. Penerapan praktis dari teknik ekstraksi fitur MFCC pada data audio

BAB II
ISI

2.1 Audio Preprocessing

Audio preprocessing adalah serangkaian langkah atau proses yang dilakukan pada data audio mentah sebelum digunakan untuk analisis atau pengolahan lebih lanjut. Tujuannya adalah untuk mempersiapkan data audio agar lebih siap digunakan dan meningkatkan kualitas serta keakuratan hasil analisis.

Proses audio preprocessing melibatkan berbagai teknik dan metode, termasuk pengurangan kebisingan, normalisasi amplitudo, resampling, pemotongan bagian tidak relevan, dan ekstraksi fitur. Berikut adalah penjelasan singkat tentang langkah-langkah yang umum dilakukan dalam audio preprocessing:

1) Pengurangan kebisingan: Teknik pengurangan kebisingan digunakan untuk menghilangkan atau mengurangi kebisingan yang tidak diinginkan dalam data audio. Metode ini dapat melibatkan penghilangan frekuensi rendah atau tinggi, penggunaan algoritma pengurangan noise seperti Algoritma Wiener atau Spectral Subtraction, atau pengurangan kebisingan adaptif.
2) Normalisasi amplitudo: Normalisasi amplitudo dilakukan untuk memastikan bahwa tingkat kekuatan suara dalam data audio konsisten di seluruh file. Tujuannya adalah untuk menghindari distorsi dan mencapai tingkat volume yang seimbang saat pemutaran atau analisis.
3) Resampling: Resampling melibatkan pengubahan frekuensi sampling audio menjadi tingkat yang diinginkan. Hal ini berguna ketika data audio direkam dengan frekuensi sampling yang berbeda atau perlu diubah agar sesuai dengan persyaratan sistem atau perangkat tertentu.
4) Pemotongan bagian tidak relevan: Bagian awal atau akhir dari data audio yang tidak relevan atau mengandung kebisingan dapat dipotong selama audio preprocessing. Tujuannya adalah untuk fokus pada bagian audio yang penting dan relevan, serta mengurangi beban pemrosesan pada langkah analisis berikutnya.

Ekstraksi fitur: Fitur-fitur khusus dapat diekstraksi dari data audio selama audio preprocessing. Misalnya, mel-frequency cepstral coefficients (MFCCs) adalah fitur yang umum diekstraksi untuk mewakili karakteristik frekuensi dan spektral audio [1].

Fitur-fitur ini berguna dalam pengenalan suara, pemrosesan ucapan, dan aplikasi lainnya.

2.2 MFCC

Mel-Frequency Cepstrum Coefficient (MFCC) adalah metode yang umum digunakan di bidang *speech technology*, di mana rekaman suara manusia diubah menjadi matriks konvolsional, yaitu spektogram atau sinyal audio, lalu diklasifikasikan menurut jenis jenis emosinya [2]. Melalui proses ini, metode MFCC membantu mengurangi dimensi data audio menjadi serangkaian koefisien cepstral yang merepresentasikan fitur-fitur penting dari suara manusia yang berkaitan dengan sentimen atau emosi. Fitur-fitur ini kemudian dapat digunakan sebagai input untuk model pengenalan sentimen atau emosi.

2.3 SVM

Support Vector Machine (SVM) adalah sistem pembelajaran yang dapat menggunakan fungsi linier dalam ruang fitur multidimensi yang dilatih dengan algoritma pembelajaran untuk membuat prediksi dalam kasus klasifikasi [3]. SVM adalah metode yang efektif dalam menangani masalah klasifikasi biner dan dapat diperluas untuk kasus klasifikasi multikelas. SVM telah digunakan secara luas dalam berbagai bidang, termasuk pengenalan pola, pengolahan citra, analisis teks, dan bioinformatika. Algoritma ini dapat memberikan performa yang baik dalam klasifikasi data, terutama ketika ada clear margin antara kelas-kelas yang ingin dipisahkan.

2.4 Source Code

Tahap Preprocessing:

```
[1]  import librosa
     import librosa.display
     import IPython.display as ipd
     import matplotlib.pyplot as plt
     import numpy as np
     import numpy as np
     from sklearn import svm
     from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
     from sklearn.model_selection import train_test_split

[2]  from google.colab import drive
     drive.mount('/content/drive')

     Mounted at /content/drive

[3]  base_path = "/content/drive/MyDrive/KuliahSmst4/PPDM"
```

Code diatas digunakan untuk mendapatkan data audio yang tersimpan pada google drive untuk digunakan sebagai data training.

```
[4]  def get_zero(number):
       len_num = 4
       len_number = len(str(number))
       return "0" * (len_num - len_number)

[5]  total_data_happy = 1061
     name_audio_happy = []
     label_audio_happy = [1 for i in range(total_data_happy)]
     for i in range(total_data_happy):
       temp = get_zero(i+1)
       name_audio_happy.append(f"{base_path}/Happy/happy-{temp}{i+1}.wav")

[6]  total_data_happy = 1088
     name_audio_sad = []
     label_audio_sad = [0 for i in range(total_data_happy)]
     for i in range(total_data_happy):
       temp = get_zero(i+1)
       name_audio_sad.append(f"{base_path}/Sad/sad-{temp}{i+1}.wav")

[7]  feature_vectors = []
     labels = []
```

Kemudian pada code diatas akan dilakukan pemberian label terhadap data audio happy dan sad, dataaudio happy akan diberi label 1, sedangkan untuk data sad akan diberikan label 0.

```
     for audio_file in name_audio_happy:
       audio, sr = librosa.load(audio_file, sr=None)  # Load the audio file
       mfccs = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=13)  # Extract 13 MFCCs
       mfcc_scaled_features = np.mean(mfccs.T, axis=0)
       feature_vectors.append(mfcc_scaled_features)


     for i in label_audio_happy:
       labels.append(i)

[9]  len(feature_vectors)

     1061

[10] for audio_file in name_audio_sad:
       audio, sr = librosa.load(audio_file, sr=None)  # Load the audio file
       mfccs = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=13)  # Extract 13 MFCCs
       mfcc_scaled_features = np.mean(mfccs.T, axis=0)
       feature_vectors.append(mfcc_scaled_features)

     for i in label_audio_sad:
       labels.append(i)

[11] # Convert feature_vectors to a numpy array
     X = feature_vectors
     y = labels

[12] len(X)

     2149
```

Pada gambar diatas, dilakukan ektraksi fitur dengan metode MFCC dengan mengubah data audio menjadi nilai ceptral koefisien. Pada ektraksi fitur ini saya menggunakan nilai mean atau rata-rata untuk mendapatkan nilai ceptral koefisen suatu data audio agar tidak terlalu panjang, dan setelah dirata-ratakan didapatkan 13 data ceptral koefisen di dalam array.

```
for audio_file in name_audio_happy:
  audio, sr = librosa.load(audio_file, sr=None)  # Load the audio file
  mfccs = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=13)  # Extract 13 MFCCs
  mfcc_scaled_features = np.mean(mfccs.T, axis=0)
  feature_vectors.append(mfcc_scaled_features)


for i in label_audio_happy:
  labels.append(i)
```

```
[9] len(feature_vectors)
    feature_vectors
```

```
          1.3145596], dtype=float32),
 array([-3.4795334e+02,  1.0383500e+02,  3.8471085e+01,  2.0702356e+01,
        7.0465622e+00, -4.7007289e+00, -6.7694750e+00, -5.7185326e+00,
       -5.9232259e-01, -2.3744492e-02, -1.5189301e+00, -2.1372988e+00,
       -6.7957063e+00], dtype=float32),
 array([-249.46388  ,  111.82071  ,   13.493727 ,   23.0812   ,
          1.4169915,  -12.227455 ,  -16.003532 ,  -14.911174 ,
         -4.6260157,   -0.9741856,   -2.4813616,   -4.101831 ,
         -6.2930894], dtype=float32),
 array([-289.57727  ,   97.87678  ,   18.322205 ,   22.640194 ,
          1.6008555,   -2.4169912,   -5.553262 ,   -6.521362 ,
          2.8730245,   -8.940381 ,   -4.0098987,   -2.6385083,
         -7.9071198], dtype=float32),
 array([-219.58505  ,   44.43035  ,   -0.39053944,   13.49223  ,
         -5.0420866 ,   -0.293253 ,    0.7944927 ,   -5.7820897 ,
          1.6993301 ,    2.234193 ,  -12.224735  ,   -4.3254423 ,
          1.1354897 ], dtype=float32),
 array([-3.7935071e+02,  1.1487050e+02,  3.5895226e+01,  3.0100014e+01,
        1.0543257e+01,  6.7348784e-01, -5.4704678e-01, -9.0244598e+00,
```

```
[11] for audio_file in name_audio_sad:
       audio, sr = librosa.load(audio_file, sr=None)  # Load the audio file
       mfccs = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=13)  # Extract 13 MFCCs
       mfcc_scaled_features = np.mean(mfccs.T, axis=0)
       feature_vectors.append(mfcc_scaled_features)

     for i in label_audio_sad:
       labels.append(i)
```

```
[12] # Convert feature_vectors to a numpy array
     X = feature_vectors
     y = labels
```

```
[13] len(X)
     X
```

```
          1.3145596], dtype=float32),
 array([-3.4795334e+02,  1.0383500e+02,  3.8471085e+01,  2.0702356e+01,
        7.0465622e+00, -4.7007289e+00, -6.7694750e+00, -5.7185326e+00,
       -5.9232259e-01, -2.3744492e-02, -1.5189301e+00, -2.1372988e+00,
       -6.7957063e+00], dtype=float32),
 array([-249.46388  ,  111.82071  ,   13.493727 ,   23.0812   ,
          1.4169915,  -12.227455 ,  -16.003532 ,  -14.911174 ,
         -4.6260157,   -0.9741856,   -2.4813616,   -4.101831 ,
         -6.2930894], dtype=float32),
 array([-289.57727  ,   97.87678  ,   18.322205 ,   22.640194 ,
          1.6008555,   -2.4169912,   -5.553262 ,   -6.521362 ,
          2.8730245,   -8.940381 ,   -4.0098987,   -2.6385083,
         -7.9071198], dtype=float32),
 array([-219.58505  ,   44.43035  ,   -0.39053944,   13.49223  ,
         -5.0420866 ,   -0.293253 ,    0.7944927 ,   -5.7820897 ,
          1.6993301 ,    2.234193 ,  -12.224735  ,   -4.3254423 ,
          1.1354897 ], dtype=float32),
 array([-3.7935071e+02,  1.1487050e+02,  3.5895226e+01,  3.0100014e+01,
        1.0543257e+01,  6.7348784e-01, -5.4704678e-01, -9.0244598e+00,
       -2.6165061e+00, -3.7561268e-01,  1.7300137e+00, -2.7269611e+00,
```

Tampilan nilai ceptral koefisien setelah dilakukan proses mencari feature extraction dengan metode MFCC.

Tahapan Training :

```
[14] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

[15] # 4. SVM Model Training
     svm_model = svm.SVC(C=0.1, kernel='poly', gamma=0.00001)
     svm_model.fit(X_train, y_train)

     SVC
     SVC(C=0.1, gamma=1e-05, kernel='poly')

     + Code  + Text

[16] # 5. Model Evaluation
     y_pred = svm_model.predict(X_test)
     accuracy = accuracy_score(y_test, y_pred)
     precision = precision_score(y_test, y_pred)
     recall = recall_score(y_test, y_pred)
     f1 = f1_score(y_test, y_pred)
     print("Accuracy:", accuracy)
     print("Precision:", precision)
     print("Recall:", recall)
     print("F1-Score:", f1)

     Accuracy: 0.8534883720930233
     Precision: 0.8686868686868687
     Recall: 0.8229665071770335
     F1-Score: 0.8452088452088452
```

```
# Print the best parameters and the best score
print("\nBest parameters:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)

# Print evaluation metrics
best_classifier = grid_search.best_estimator_
accuracy = best_classifier.score(X_test, y_test)
print("Accuracy:", accuracy)
precision = best_classifier.score(X_test, y_test)
print("Precision:", precision)
recall = best_classifier.score(X_test, y_test)
print("Recall:", recall)
f1 = best_classifier.score(X_test, y_test)
print("F1-Score:", f1)
```

params	mean_test_score	accuracy	precision	recall	f1_score
{'C': 0.1, 'kernel': 'poly'}	0.852810	0.851163	0.851321	0.851163	0.851076
{'C': 0.1, 'kernel': 'rbf'}	0.846422	0.851163	0.851321	0.851163	0.851076
{'C': 1, 'kernel': 'poly'}	0.849322	0.851163	0.851321	0.851163	0.851076
{'C': 1, 'kernel': 'rbf'}	0.854555	0.851163	0.851321	0.851163	0.851076
{'C': 10, 'kernel': 'poly'}	0.859794	0.851163	0.851321	0.851163	0.851076
{'C': 10, 'kernel': 'rbf'}	0.856884	0.851163	0.851321	0.851163	0.851076
{'C': 100, 'kernel': 'poly'}	0.866189	0.851163	0.851321	0.851163	0.851076
{'C': 100, 'kernel': 'rbf'}	0.863865	0.851163	0.851321	0.851163	0.851076

```
Best parameters: {'C': 100, 'kernel': 'poly'}
Best score: 0.8661892331683504
Accuracy: 0.8511627906976744
Precision: 0.8511627906976744
Recall: 0.8511627906976744
F1-Score: 0.8511627906976744
```

Tahapan training yang digunakan dengan metode SVM yaitu dengan menggunakan beberapa kombinasi hyper-parameters (nilai C = 0.1, atau 1, atau 10,atau 100, nilai gamma = 0.0001 atau 0.001, atau 0.1, atau 1, dan fungsi kernel: rbf atau polynomial). Dari data training ini didapatkan yang mempunyai akurasi terbaik yaitu dengan kombinasi hyper-parameters: nilai C = 100, nilai gamma = 0.001, fungsi kernel poly, mendapatkan akurasi = 0.85 , precision = 0.86, recall = 0.82 , F1-Score = 0.84.

Tahapan Testing :

```
[16] import pickle
     # Save the trained SVM model to a .pkl file
     model_path = "svm.pkl"
     with open(model_path, 'wb') as file:
         pickle.dump(svm_model, file)
```

```
import librosa
import numpy as np
import pickle

# Define the path to your saved SVM model (.pkl file)
model_path = "svm.pkl"

# Function to extract MFCC features from audio files
def extract_features(file_path):
    audio, sample_rate = librosa.load(file_path)
    mfcc_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=13)
    mfcc_scaled_features = np.mean(mfcc_features.T, axis=0)
    return mfcc_scaled_features

# Function to predict the emotion of a new audio file
def predict_emotion(file_path, model):
    # Extract features from the new audio file
    features = extract_features(file_path)

    # Reshape the features into a single sample
    features = features.reshape(1, -1)

    # Predict the emotion using the loaded SVM model
    emotion = model.predict(features)[0]

    return emotion

# Path to the new audio file for prediction
new_audio_path = "laughing-man-117725.mp3"

# Load the trained SVM model from the .pkl file
with open(model_path, 'rb') as file:
    trained_model = pickle.load(file)

label_to_emotion = {0: 'sad', 1: 'happy'}

# Predict the emotion of the new audio file using the loaded model
predicted_emotion = predict_emotion(new_audio_path, trained_model)
print("Predicted Emotion:", label_to_emotion[predicted_emotion])
```

```
Predicted Emotion: happy
```

Data input yang digunakan pada tahapan testing ini yaitu data inputan audio baru selain dari data training.

Tahap Visualisasi Deployment Website:

visual.py svm.pkl

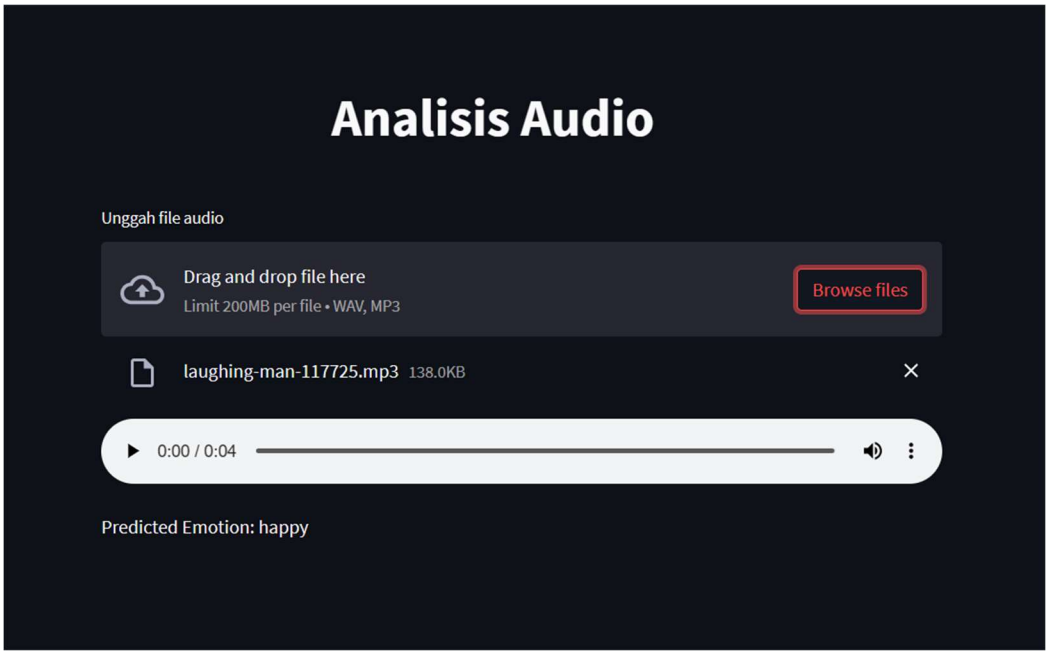
visual.py > ...

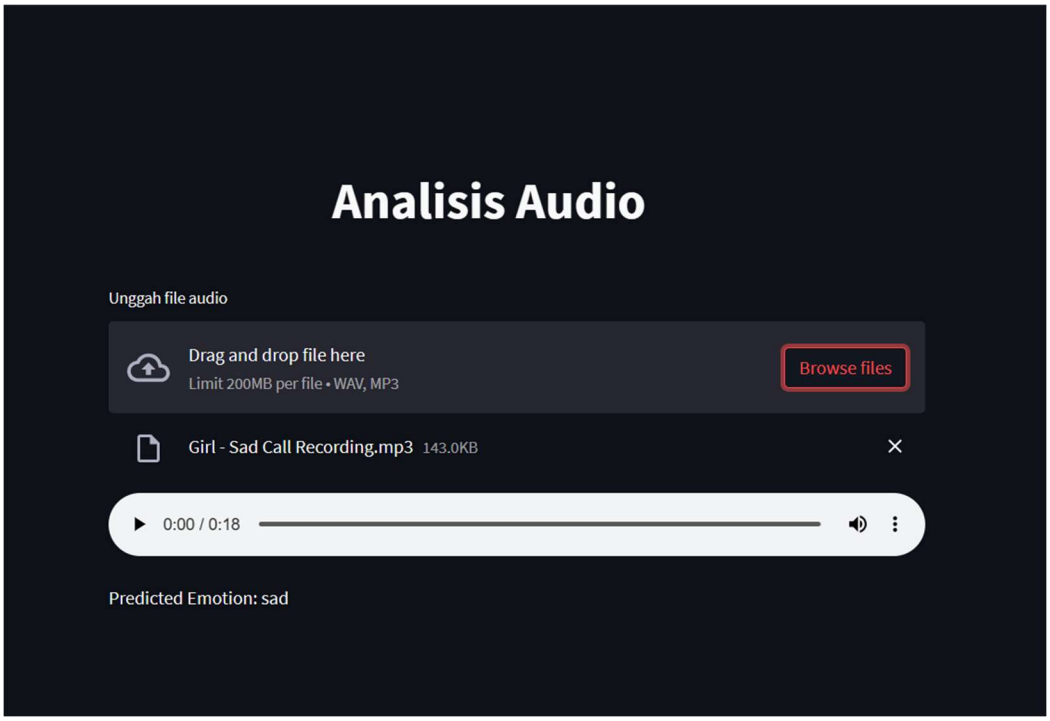
```python
import streamlit as st
import soundfile as sf
import librosa
import numpy as np
import pickle
import sounddevice as sd

st.markdown(
    """
    <style>
    .centered-text {
        display: flex;
        justify-content: center;
        align-items: center;
        height: 20vh
    }
    </style>
    """,
    unsafe_allow_html=True
)
st.markdown('<div class="centered-text"><h1>Analisis Audio</h1></div>', unsafe_allow_html=True)

# File uploaded
uploaded_file = st.file_uploader("Unggah file audio", type=["wav", "mp3"])

# Function to extract MFCC features from audio files
def extract_features(file_path):
    audio, sample_rate = librosa.load(file_path)
    mfcc_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=13)
    mfcc_scaled_features = np.mean(mfcc_features.T, axis=0)
    return mfcc_scaled_features
```

```python
# Function to predict the emotion of a new audio file
def predict_emotion(file_path, model):
    # Extract features from the new audio file
    features = extract_features(file_path)

    # Reshape the features into a single sample
    features = features.reshape(1, -1)

    # Predict the emotion using the loaded SVM model
    emotion = model.predict(features)[0]

    return emotion

if uploaded_file is not None:
    audio, samplerate = sf.read(uploaded_file)
    st.audio(uploaded_file)
    # Define the path to your saved SVM model (.pkl file)
    model_path = "svm.pkl"

    # Path to the new audio file for prediction
    new_audio_path = uploaded_file

    # Load the trained SVM model from the .pkl file
    with open(model_path, 'rb') as file:
        trained_model = pickle.load(file)

    label_to_emotion = {0: 'sad', 1: 'happy'}

    # Predict the emotion of the new audio file using the loaded model
    predicted_emotion = predict_emotion(new_audio_path, trained_model)
    st.write("Predicted Emotion:", label_to_emotion[predicted_emotion])
```

Pada tahap deploy kami menggunakan streamlit untuk memudahkan dalam membuat tampilan website.

2.5 Hasil Implementasi





Pada tahapan deployment kami menggunakan streamlit yaitu framework berbasis python dan bersifat open-source yang dibuat untuk memudahkan dalam membangun aplikasi web. Pada website ini hanya dapat menerima inputan audio suara berupa file, kemudian outputnya berupa hasil sentimen atau identifikasi emosi dari file audio yang di inputkan.

BAB III
PENUTUP

3.1 Kesimpulan

Dalam audio preprocessing, dilakukan pengurangan kebisingan untuk menghilangkan gangguan suara, normalisasi amplitudo untuk menjaga konsistensi volume audio, resampling untuk mengubah frekuensi sampling agar sesuai dengan persyaratan sistem, pemotongan bagian tidak relevan untuk memfokuskan pada bagian audio yang penting, dan ekstraksi fitur seperti Mel-Frequency Cepstral Coefficients (MFCCs) untuk menggambarkan karakteristik audio.

Dalam proses pengembangan sistem aplikasi ini, kami menemukan bahwa penggunaan metode MFCC memberikan representasi fitur yang kuat dari data audio, yang memungkinkan pemodelan yang akurat terhadap sentimen atau emosi yang terkandung dalam suara manusia. Metode MFCC juga membantu dalam mengurangi dimensi data audio, sehingga meningkatkan efisiensi dan kinerja model. Penggunaan algoritma SVM dalam tahap klasifikasi memberikan hasil yang memuaskan. Dengan variasi hyperparameter seperti nilai C, nilai gamma, dan jenis kernel, kami dapat menyesuaikan model SVM untuk mencapai performa yang optimal. Evaluasi menggunakan metrik akurasi, precision, recall, dan F1-Score membuktikan efektivitas model dalam mengenali sentimen atau emosi dari suara orang.

REFERENSI

[1] J. Elektronik and I. Komputer Udayana, “Penerapan Metode MFCC dan Naive Bayes untuk Deteksi Suara Paru-Paru”.

[2] P. Widya Eka Safitri, A. Eka Karyawati, and K. Selatan, “Kombinasi Metode MFCC dan KNN dalam Pengenalan Emosi Manusia Melalui Ucapan,” 2022.

[3] “Deteksi Suara Gitar Dengan Bahan Jenis Senar Berbeda Melalui Ciri Akustik Dengan Mel-Frequency Cepstral Coefficients (MFCC) Dan Support Vector Machine (SVM) Guitar String Detection Through Acoustic Characteristics Using Mel-Frequency Cepstral Coefficients (MFCC) And Support Vector Machine (SVM) Methods.”