

**SISTEM IDENTIFIKASI SENTIMEN POSITIF & NEGATIF PADA
AUDIO MENGGUNAKAN *SUPPORT VECTOR MACHINE*
MATA KULIAH PENGANTAR PEMROSESAN DATA MULTIMEDIA**



DISUSUN OLEH :

ANAK AGUNG MADE KRISNA ASTRAWAN	2108561021/A
CHARLES ALEXANDER RIRIMASSE	2108561043/A
I GUSTI AGUNG NGURAH DIPUTRA WIRAGUNA	2108561075/A
NI PUTU SRI AGNITA SAMYAMI WIRAPUTRI	2108561111/A

**PROGRAM STUDI INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS UDAYANA**

JIMBARAN

2023

DAFTAR ISI

DAFTAR ISI.....	1
DAFTAR GAMBAR.....	2
BAB 1 PENDAHULUAN	3
1.1 Latar Belakang	3
1.2 Rumusan Masalah	4
1.3 Tujuan	5
1.4 Manfaat	5
BAB 2 PERANCANGAN & IMPLEMENTASI.....	6
2.1 Landasan Teoritis	6
2.1.1 Data Audio	6
2.1.2 <i>Preprocessing</i> Data Audio.....	7
2.1.3 <i>Training</i> Data Audio.....	8
2.1.4 <i>Testing</i> Data Audio	9
2.1.5 <i>Support Vector Machine</i>	9
2.1.6 <i>Mel Frequency Cepstral Coefficients</i>	10
2.1 Manual Aplikasi	12
2.1.1 Fitur Sistem.....	13
2.1.2 Antarmuka	14
2.3 Implementasi <i>Coding</i>	17
2.3.1 <i>File test_model.ipynb</i>	17
2.3.2 <i>File svm.py</i>	26
2.3.3 <i>File preprocessing.py</i>	28
2.3.4 <i>File main.py</i>	30
2.3.5 <i>File tab_one.py</i>	31
2.3.6 <i>File tab_two.py</i>	34
BAB 3 PENUTUP.....	39
3.2 Kesimpulan	39
3.3 Saran.....	39
DAFTAR PUSTAKA	41

DAFTAR GAMBAR

Gambar 2.1 - Antarmuka Sistem (Halaman Utama)	15
Gambar 2.2 - Antarmuka Sistem (Hasil Analisis Audio).....	16
Gambar 2.3 - Halaman Dokumentasi	16
Gambar 2.4 - Antarmuka hasil <i>file</i> tab_one.py	33
Gambar 2.5 - Antarmuka hasil <i>file</i> tab_two.py	38

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Dalam era digital yang semakin berkembang, data audio menjadi salah satu sumber informasi yang berharga. Identifikasi sentimen pada data audio menjadi topik yang menarik karena mampu mengungkap perasaan dan opini pengguna terkait suatu topik tertentu. Dalam beberapa konteks, seperti analisis layanan pelanggan, penilaian produk, dan pengawasan media sosial, penting untuk mengidentifikasi apakah sentimen yang terkandung dalam data audio adalah positif atau negatif. Salah satu metode yang dapat digunakan untuk melakukan identifikasi sentimen pada data audio adalah *Support Vector Machine* (SVM).

Metode SVM adalah salah satu algoritma pembelajaran mesin yang populer dan efektif dalam melakukan klasifikasi. SVM bekerja dengan membangun *hiperplane* yang memaksimalkan jarak antara dua kelas yang berbeda dalam ruang fitur. Dalam konteks identifikasi sentimen, SVM dapat digunakan untuk mempelajari pola-pola dari data audio yang mengindikasikan sentimen positif atau negatif [1]

Penggunaan metode SVM dalam identifikasi sentimen pada data audio memiliki beberapa keuntungan. Pertama, SVM mampu mengatasi masalah data yang tidak linier dengan menggunakan fungsi kernel. Hal ini memungkinkan SVM untuk menemukan pola yang kompleks dalam data audio. Kedua, SVM merupakan metode pembelajaran yang relatif cepat dan efisien, terutama ketika dihadapkan pada jumlah data yang besar [2].

Untuk menerapkan metode SVM dalam identifikasi sentimen pada data audio, beberapa tahapan umum yang harus dilakukan antara lain: ekstraksi fitur audio, pengolahan fitur, pemilihan fitur yang relevan, pelabelan data dengan sentimen positif atau negatif, dan pelatihan serta evaluasi model SVM [2].

Ekstraksi fitur pada data audio dapat dilakukan dengan berbagai metode, dan salah satu metode yang umum digunakan adalah *Mel-frequency cepstral coefficients* (MFCC). MFCC adalah representasi numerik yang menggambarkan karakteristik spektral dari sinyal audio. Proses ekstraksi MFCC melibatkan beberapa tahap, seperti penerapan fungsi *windowing* pada sinyal audio, menghitung transformasi Fourier, dan menerapkan filter bank *mel-frequency*. Hasil dari ekstraksi MFCC adalah serangkaian koefisien *cepstral* yang merepresentasikan distribusi energi frekuensi dalam sinyal audio [3].

Tujuan dari pembuatan model sistem yang menggunakan SVM dan MFCC untuk menganalisis sentimen pada data audio dengan pembagian data training sebesar 80% dan data testing sebesar 20% adalah untuk mengembangkan alat yang efektif dalam mengenali sentimen positif dan negatif dalam data audio. Dengan memanfaatkan metode SVM sebagai algoritma klasifikasi dan ekstraksi fitur MFCC sebagai representasi numerik dari sinyal audio, model ini diharapkan dapat mempelajari pola-pola akustik yang mengindikasikan sentimen tertentu. Melalui pembagian data training dan data testing, model dapat dilatih dengan menggunakan data training yang luas untuk memahami variasi sentimen yang ada, dan kemudian diuji dengan menggunakan data testing yang belum pernah dilihat sebelumnya untuk mengukur kinerja dan akurasi model. Dengan demikian, tujuan utama adalah menghasilkan model yang mampu mengidentifikasi sentimen pada data audio secara akurat dan dapat digunakan untuk aplikasi praktis, seperti analisis umpan balik pelanggan, pemantauan media sosial, dan penilaian produk.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, dapat tuliskan perumusan masalah yang akan dibahas, yaitu:

- a. Bagaimana menerapkan metode SVM dan MFCC agar dapat membuat model yang dapat menganalisis sentimen pada data audio?
- b. Bagaimana meningkatkan akurasi pada saat mengidentifikasikan sentimen?

1.3 Tujuan

Adapun tujuan yang ingin dicapai dari penelitian ini adalah memanfaatkan metode SVM sebagai algoritma klasifikasi dan ekstraksi fitur MFCC sebagai representasi numerik dari sinyal audio untuk menghasilkan model yang mampu mengidentifikasi sentimen positif dan negatif pada data audio secara akurat.

1.4 Manfaat

Adapun manfaat yang ingin dicapai dari penelitian ini adalah:

1. Bagi Penyusun
 - a. Sebagai media penerapan dan pengimplementasian hasil ilmu yang diperoleh dari perkuliahan.
 - b. Sebagai media untuk mengukur capaian pembelajaran selama kuliah.
2. Bagi Masyarakat
 - a. Sebagai referensi untuk membuat model yang berkaitan dengan data audio menggunakan metode SVM dan MFCC.
 - b. Menambah pengetahuan mengenai pemrosesan data audio.

BAB 2

PERANCANGAN & IMPLEMENTASI

2.1 Landasan Teoritis

Untuk memperdalam pemahaman mengenai sistem dan model yang digunakan, terdapat beberapa konsep yang digunakan dalam proyek ini, yaitu:

2.1.1 Data Audio

Data audio adalah representasi digital dari gelombang suara. Suara adalah getaran mekanis yang merambat melalui medium, seperti udara atau air. Data audio merekam gelombang suara tersebut dalam bentuk sinyal listrik yang dapat ditangkap, direkam, dan diproses oleh perangkat elektronik. Format audio adalah cara penyimpanan dan representasi data audio. Beberapa format audio umum yang digunakan termasuk WAV (Waveform Audio *File* Format), MP3 (MPEG Audio Layer-3), dan FLAC (Free Lossless Audio Codec). Setiap format audio memiliki karakteristik dan kompresi yang berbeda untuk mengoptimalkan ukuran *file* dan kualitas suara [4].

Data audio memiliki beberapa sifat yang perlu dipahami, antara lain:

Gambar 1. Amplitudo

Amplitudo menggambarkan kekuatan atau tingkat volume suara. Amplitudo ini direpresentasikan oleh besaran angka pada data audio.

Gambar 2. Frekuensi

Frekuensi mengacu pada jumlah siklus gelombang suara yang terjadi dalam satu detik. Frekuensi ini terkait dengan nada suara yang didengar.

Gambar 3. Durasi

Durasi adalah lamanya waktu yang dibutuhkan untuk merekam atau memainkan data audio. Durasi berkaitan dengan panjang atau pendeknya suara yang direkam atau diputar.

Gambar 4. Spektrum

Spektrum audio menggambarkan sebaran energi frekuensi dalam suara. Spektrum ini dapat dianalisis untuk mengekstraksi fitur-fitur yang relevan dalam pemrosesan audio.

2.1.2 *Preprocessing* Data Audio

Preprocessing adalah tahap persiapan data sebelum dilakukan pelatihan dan pengujian model. Tujuan dari *preprocessing* adalah untuk membersihkan, memformat, dan menyesuaikan data agar sesuai dengan kebutuhan model klasifikasi [5].

Tahapan *preprocessing* dapat meliputi:

a. Pengumpulan Data

Data yang akan digunakan dalam model klasifikasi dikumpulkan dari sumber yang relevan dan sesuai dengan tujuan analisis.

b. Pembersihan Data

Data yang terkumpul kemungkinan memiliki *noise*, *missing values*, atau *outlier* yang perlu diidentifikasi dan dihapus atau diimputasi.

c. Transformasi Data

Data dapat mengalami transformasi untuk meningkatkan kualitas atau mengubah skala nilai dalam rentang yang diinginkan, seperti normalisasi atau standarisasi

d. Ekstraksi Fitur

Fitur-fitur yang relevan dan penting untuk model klasifikasi diekstraksi dari data. Hal ini melibatkan pemilihan fitur yang tepat dan mengubah data menjadi representasi numerik yang

sesuai, seperti menggunakan metode seperti MFCC dalam pemrosesan data audio.

2.1.3 Training Data Audio

Training adalah tahap di mana model klasifikasi diperoleh dari data training. Dalam tahap ini, model belajar dari data training untuk mengidentifikasi pola dan relasi antara fitur dengan label klasifikasi yang ada.

Beberapa konsep penting dalam tahap training adalah:

1. Algoritma Klasifikasi

Algoritma klasifikasi, seperti SVM (*Support Vector Machine*), *Decision Tree*, atau *Neural Network*, dipilih dan diterapkan untuk melatih model. Setiap algoritma memiliki prinsip dan karakteristik yang berbeda.

2. Parameter Tuning

Dalam algoritma klasifikasi khususnya SVM terdapat beberapa parameter yang digunakan dalam model klasifikasi. Beberapa parameter dalam SVM meliputi jenis kernel, nilai C, nilai gamma, dan nilai dfs. Parameter tuning dilakukan untuk mencari nilai parameter yang optimal sehingga diperoleh nilai pengujian sistem yang baik.

3. Pembelajaran Supervisi

Proses pembelajaran dilakukan dengan menggunakan contoh-contoh yang dilengkapi dengan label klasifikasi yang sudah diketahui. Model diperbarui secara iteratif melalui pengoptimalan parameter agar dapat meminimalkan kesalahan dalam klasifikasi.

4. Validasi Silang

Untuk mencegah *overfitting* dan mengukur kinerja model secara objektif, teknik validasi silang (*cross-validation*) digunakan untuk membagi data training menjadi subset yang digunakan untuk pelatihan dan pengujian iteratif [6].

2.1.4 Testing Data Audio

Testing adalah tahap di mana model yang telah dilatih dievaluasi menggunakan data testing yang belum pernah dilihat sebelumnya. Tujuan dari tahap ini adalah untuk mengukur kinerja dan akurasi model klasifikasi.

Beberapa aspek penting dalam tahap testing meliputi:

1. Pengukuran Kinerja

Metrik evaluasi, seperti akurasi, presisi, recall, F1-score, atau kurva ROC (Receiver Operating Characteristic), digunakan untuk mengukur kinerja model dalam mengklasifikasikan data testing.

2. Generalisasi

Tingkat generalisasi model, yaitu kemampuannya untuk memprediksi dengan akurat pada data baru yang belum pernah dilihat sebelumnya, dievaluasi melalui hasil pengujian pada data testing.

3. Analisis hasil

Hasil pengujian dan evaluasi model dapat dianalisis untuk memahami kekuatan dan kelemahan model serta memperbaiki atau mengoptimalkan model jika diperlukan [7].

2.1.5 Support Vector Machine

Metode SVM adalah salah satu algoritma pembelajaran mesin yang populer dan efektif dalam melakukan klasifikasi. SVM bekerja dengan membangun *hiperplane* yang memaksimalkan jarak antara dua kelas yang berbeda dalam ruang fitur. Dalam konteks identifikasi sentimen, SVM dapat digunakan untuk mempelajari pola-pola dari data audio yang mengindikasikan sentimen positif atau negatif [1].

Penggunaan metode SVM dalam identifikasi sentimen pada data audio memiliki beberapa keuntungan. Pertama, SVM mampu mengatasi masalah data yang tidak linier dengan menggunakan fungsi kernel. Hal

ini memungkinkan SVM untuk menemukan pola yang kompleks dalam data audio. Kedua, SVM merupakan metode pembelajaran yang relatif cepat dan efisien, terutama ketika dihadapkan pada jumlah data yang besar [2].

2.1.6 *Mel Frequency Cepstral Coefficients*

MFCC adalah metode ekstraksi fitur yang umum digunakan dalam pemrosesan audio, terutama untuk pengenalan suara dan analisis klasifikasi. Metode ini didasarkan pada pengamatan bahwa manusia memiliki persepsi frekuensi yang tidak linear, di mana perbedaan frekuensi rendah lebih mudah dibedakan daripada perbedaan frekuensi tinggi.

Langkah-langkah utama dalam ekstraksi MFCC adalah sebagai berikut:

1. Pre-emphasis

Langkah ini melibatkan penerapan filter pre-emphasis pada sinyal audio untuk meningkatkan energi frekuensi tinggi. Filter pre-emphasis memberikan penekanan yang lebih tinggi pada komponen frekuensi tinggi dalam sinyal audio.

2. Framing

Sinyal audio dibagi menjadi beberapa frame waktu yang saling tumpang tindih. Setiap frame biasanya memiliki durasi sekitar 20-40 milidetik dan memiliki fungsi jendela (seperti jendela Hamming) untuk mengurangi efek spektral pada batas frame.

3. Transformasi Fourier

Transformasi *Fourier Dalam Frame* (FFT) diterapkan pada setiap *frame* untuk mengubah domain waktu menjadi domain frekuensi. Ini memberikan representasi spektral dari setiap *frame* audio.

4. Mel Filterbank

Mel *Filterbank* adalah serangkaian filter yang diterapkan pada hasil FFT untuk mendapatkan estimasi energi spektral dalam rentang frekuensi Mel. Rentang frekuensi Mel adalah skala frekuensi yang menyesuaikan persepsi manusia terhadap frekuensi suara.

5. *Logarithmic Scaling*

Setelah filter Mel, energi spektral logaritmik dari setiap filter dihitung untuk mengkompresi informasi spektral dan menyesuaikan dengan persepsi pendengaran manusia.

6. Transformasi *Cepstral*

Transformasi *cepstral* dilakukan pada spektrum logaritmik dengan menggunakan transformasi cepstral melalui transformasi *Fourier* invers. Transformasi ini menghasilkan koefisien cepstral yang menggambarkan karakteristik frekuensi dan amplitudo sinyal audio.

7. Pengurangan Dimensi

Untuk mengurangi dimensi data, beberapa koefisien cepstral yang memiliki kontribusi rendah dihilangkan [8].

2.1 Manual Aplikasi

Aplikasi yang telah dibuat menggunakan Streamlit *Website* terdiri dari Aplikasi ini dirancang untuk melakukan analisis sentimen pada audio yang diunggah oleh pengguna. Berikut adalah panduan penggunaan aplikasi ini:

1. Pengguna membuka link aplikasi yang telah diberikan. Setelah halaman web terbuka, pengguna akan melihat tampilan antarmuka aplikasi.
2. Di halaman utama, terdapat penjelasan singkat tentang aplikasi dan petunjuk penggunaan. Pengguna dapat melihat informasi ini sebagai panduan awal sebelum memulai analisis audio.
3. Di halaman analisis audio, pengguna akan menemukan formulir untuk mengunggah *file* audio yang ingin dianalisis. Pengguna dapat menggunakan tombol "*Upload File*" untuk memilih *file* audio dari perangkat mereka.
4. Setelah pengguna berhasil mengunggah *file* audio, aplikasi akan memproses audio tersebut menggunakan model yang telah dilatih sebelumnya. Proses ini melibatkan ekstraksi fitur-fitur audio, normalisasi, dan prediksi sentimen menggunakan metode *Support Vector Machine* (SVM).
5. Setelah proses analisis selesai, hasil analisis akan ditampilkan kepada pengguna. Hal ini dapat berupa teks yang menunjukkan sentimen prediksi audio, misalnya "Sentimen audio: Bahagia" atau "Sentimen audio: Sedih".
6. Selain itu, pengguna juga dapat melihat dokumentasi tentang proses analisis yang dilakukan pada audio. Dokumentasi ini menjelaskan tahapan preprocessing audio, ekstraksi fitur-fitur audio seperti waveform, RMSe, ZCR, dan MFCC, serta tahapan penggunaan metode SVM. Dokumentasi ini memberikan pemahaman lebih mendalam tentang proses analisis audio yang dilakukan oleh aplikasi.

7. Pengguna juga dapat beralih ke tab kedua dengan mengklik tab 'Documentation'. Di mana pengguna dapat melihat dokumentasi lebih lanjut tentang tahapan preprocessing dan penggunaan SVM.
8. Di halaman 'Documentation', pengguna dapat melihat penjelasan rinci tentang fungsi-fungsi yang digunakan dalam tahapan preprocessing, seperti menampilkan waveform, RMSe, ZCR, dan MFCC. Selain itu, pengguna juga dapat melihat penjelasan tentang tahapan penggunaan SVM, termasuk ekstraksi fitur audio, normalisasi fitur, dan prediksi sentimen.

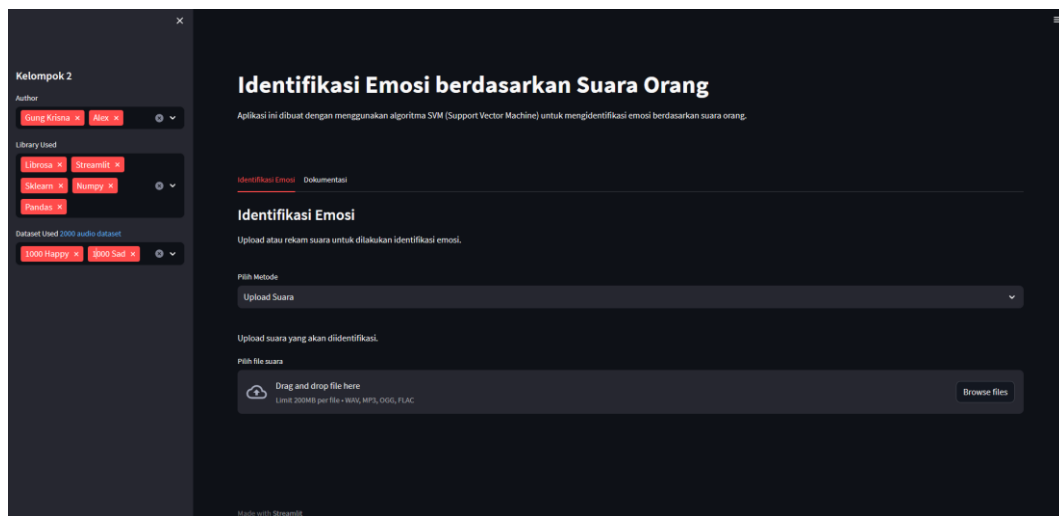
Dengan mengikuti langkah-langkah di atas, pengguna dapat menggunakan aplikasi Streamlit *Website* untuk menganalisis sentimen pada audio yang diunggah. Aplikasi ini memberikan kemudahan bagi pengguna untuk mengunggah *file* audio, melihat hasil analisis sentimen, dan mendapatkan pemahaman lebih dalam tentang proses analisis yang dilakukan pada audio melalui dokumentasi yang tersedia.

2.1.1 Fitur Sistem

Sistem yang telah dikembangkan dapat diakses melalui tautan berikut: <https://audio-identification-svm.streamlit.app/>. Sistem yang dibuat adalah sebuah sistem berbasis *Website* yang menggunakan Streamlit sebagai *framework* untuk mengimplementasikan analisis sentimen pada data audio. Pengguna dapat mengunggah *file* audio yang ingin dianalisis melalui antarmuka *Website* yang disediakan. Setelah *file* audio diunggah, sistem akan menggunakan metode ekstraksi fitur seperti MFCC (*Mel Frequency Cepstral Coefficients*) untuk mengubah audio menjadi representasi numerik yang relevan. Kemudian, fitur-fitur tersebut akan digunakan sebagai input untuk model klasifikasi SVM (*Support Vector Machine*) yang telah dilatih sebelumnya. Model akan memberikan prediksi sentimen audio apakah sedih (*sad*) atau

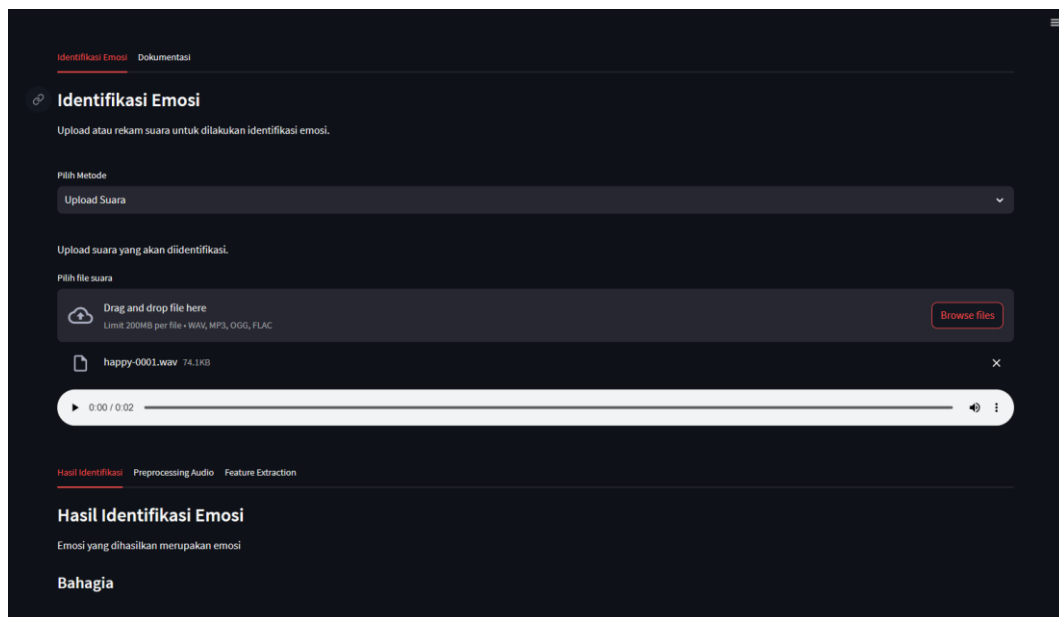
senang (*happy*). Hasil prediksi sentimen akan ditampilkan kepada pengguna melalui antarmuka *Website*.

2.1.2 Antarmuka



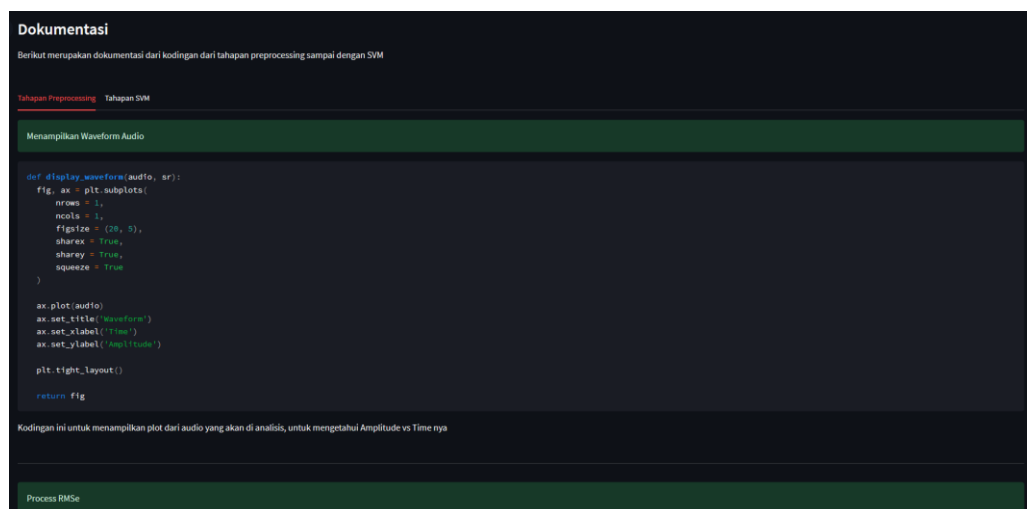
Gambar 2.1 - Antarmuka Sistem (Halaman Utama)

Sistem ini memanfaatkan keunggulan Streamlit sebagai *framework* pengembangan web yang sederhana dan intuitif. Streamlit menyediakan komponen dan fungsi yang mudah digunakan untuk membangun antarmuka pengguna interaktif dengan cepat. Dengan demikian, pengguna dapat dengan mudah mengunggah *file* audio dan menerima hasil analisis sentimen dengan cepat melalui tampilan *Website* yang responsif seperti pada gambar berikut.



Gambar 2.2 - Antarmuka Sistem (Hasil Analisis Audio)

Dalam konteks aplikasi yang menggunakan Streamlit untuk menganalisis sentimen pada data audio, dokumentasi dapat disajikan dengan mengintegrasikan teks, kode, dan output dalam satu antarmuka web yang konsisten. Pada tahap *preprocessing*, dokumentasi dapat menjelaskan langkah-langkah yang terlibat dalam persiapan data sebelum dilakukan pelatihan dan pengujian model.



Gambar 2.3 - Halaman Dokumentasi

2.3 Implementasi Coding

Implementasi kodingan untuk menganalisis sentimen positif dan negatif dari sebuah audio menggunakan Python dengan *library* numpy, pandas, librosa, matplotlib, os, dan path dapat dilakukan dengan langkah-langkah berikut. Pertama, mengimpor *library* yang diperlukan, seperti numpy, pandas, librosa, matplotlib, os, dan path. Kemudian, menggunakan *library* librosa untuk memuat audio dan melakukan ekstraksi fitur, seperti MFCC, dari audio tersebut. Fitur-fitur audio yang diekstraksi dapat disimpan dalam format yang sesuai, seperti data frame menggunakan *library* pandas. Setelah itu, fitur-fitur tersebut dapat divisualisasikan menggunakan *library* matplotlib untuk mendapatkan wawasan visual tentang karakteristik audio. Selanjutnya, dengan menggunakan metode klasifikasi yang sesuai, seperti SVM (*Support Vector Machine*), memungkinkan untuk melatih model untuk memprediksi sentimen audio berdasarkan fitur-fitur yang diekstraksi sebelumnya. Model yang dilatih dapat diuji pada data testing untuk mengukur kinerjanya. Berikut *source code* dari sistem.

2.3.1 File test_model.ipynb

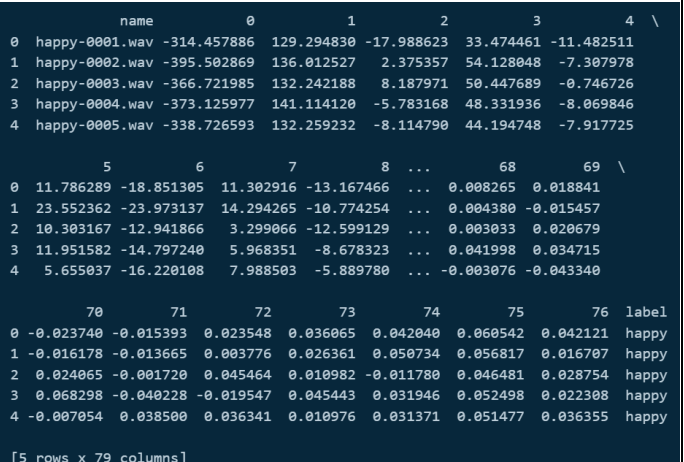
File 'test_model.ipynb' merupakan bagian dari implementasi Sistem Identifikasi Sentimen Positif dan Negatif pada Audio menggunakan metode *Support Vector Machine* (SVM). Pada file ini, dilakukan proses pelatihan model dengan menggunakan dataset yang terdiri dari 2000 sampel audio. Dataset ini terdiri dari 1000 audio dengan sentimen positif (senang) dan 1000 audio dengan sentimen negatif (sedih).

Proses pelatihan model dimulai dengan ekstraksi fitur audio menggunakan teknik *Mel-frequency Cepstral Coefficients* (MFCC). MFCC digunakan untuk mengubah audio menjadi representasi numerik yang dapat digunakan oleh model. Selanjutnya, dilakukan parameter tuning untuk mengoptimalkan kinerja model. Hal ini melibatkan pemilihan parameter SVM yang optimal, seperti jenis kernel, C (*cost*), dan gamma. Setelah melatih model dengan data pelatihan, file ini juga mencakup evaluasi kinerja model menggunakan *confusion matrix* yang

digunakan untuk menghitung jumlah prediksi yang benar dan salah dari model terhadap data uji. Dengan memeriksa *confusion matrix*, dapat dievaluasi sejauh mana model dapat membedakan sentimen positif dan negatif pada audio.

SOURCE CODE : test_model.ipynb

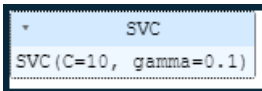
No	Source code	Penjelasan
1 2 3 4 5 6	<pre>import numpy as np import pandas as pd import librosa import matplotlib.pyplot as plt import os from pathlib import Path</pre>	Memasukkan <i>library</i> numpy, pandas, librosa, matplotlib, os dan path
1 2 3 4 5 6 7 8	<pre>path = Path().parent path_dir = path / "dataset" happy_dir = path_dir / "happy" sad_dir = path_dir / "sad" audio_happy = os.listdir(happy_dir) audio_sad = os.listdir(sad_dir)</pre>	Membuat path untuk mengambil data audio sad dan happy menggunakan <i>library</i> os
1 2 3 4 5 6 7	<pre>def rms(audio_path, frame=2048, hop=512): audio, sr = librosa.load(audio_path) rms = librosa.feature.rms(y=audio, frame_length=frame, hop_length=hop)[0] return rms</pre>	Mendeklarasikan sebuah fungsi untuk preprocessing, melakukan ekstraksi fitur dengan root mean square agar data lebih stabil
1 2 3 4 5 6	<pre>def zcr(audio_path, frame=2048, hop=512): audio, sr = librosa.load(audio_path) zcr = librosa.feature.zero_crossing_rate(y=audio, frame_length=frame, hop_length=hop)[0] return zcr</pre>	Mendeklarasikan sebuah fungsi untuk preprocessing, melakukan ekstraksi fitur dengan zero crossing rate agar data lebih stabil
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	<pre>def mfcc(audio_path, frame=2048, hop=512, mfcc_num=25): audio, sr = librosa.load(audio_path) mfcc_spectrum = librosa.feature.mfcc(y=audio, sr=sr, n_fft=frame, hop_length=hop, n_mfcc=mfcc_num) delt1 = librosa.feature.delta(mfcc_spectrum, order=1) delt2 = librosa.feature.delta(mfcc_spectrum, order=2) mfcc_feature = np.concatenate((np.mean(mfcc_spectrum, axis=1), np.mean(delt1, axis=1), np.mean(delt2, axis=1))) return mfcc_feature</pre>	Mendeklarasikan sebuah fungsi yang merepresentasikan mfcc untuk melakukan ekstraksi fitur dengan mel frequency cepstral coefficients, menggabungkan hasil preprocessing rms dan zcr sehingga mendapatkan matrix 1 x jumlah matrix yang diinginkan

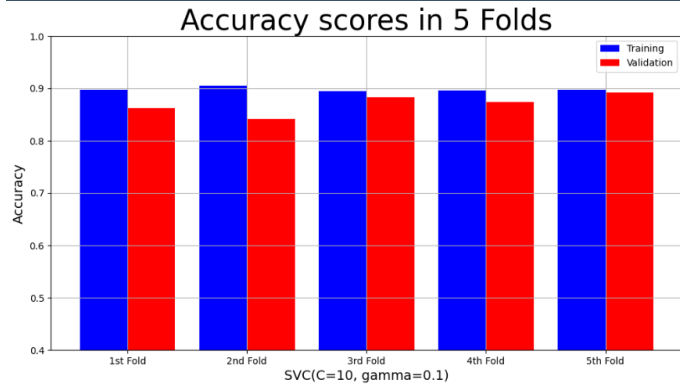
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 24 25 26 27 28	<pre> def feature_extraction(dir, audio_file, label = None, frame=2048, hop=512, mfcc_num=25): audio_mfcc, audio_zcr, audio_rmse, audio_label = [],[],[],[] for audio in audio_file: audio_path = dir / audio mfcc_score = mfcc(audio_path, frame=frame, hop=hop, mfcc_num=mfcc_num) zcr_score = np.mean(zcr(audio_path, frame=frame, hop=hop)) rmse_score = np.mean(rms(audio_path, frame=frame, hop=hop)) audio_mfcc.append(mfcc_score) audio_zcr.append(zcr_score) audio_rmse.append(rmse_score) audio_label.append(label) feature = np.column_stack((audio_mfcc, audio_zcr, audio_rmse)) df = pd.DataFrame(feature) df.insert(loc=0, column="name", value=audio_file) df["label"] = audio_label return df </pre>	Mendeklarasikan fungsi untuk melakukan ekstraksi audio dan optimalisasi data dengan menggabungkan hasil preprocessing dan mfcc sehingga menghasilkan matrix feature
1 2 3	<pre> df_happy = feature_extraction(happy_dir, audio_happy, "happy") print(df_happy.head()) </pre> <p>OUTPUT :</p>  <p>[5 rows x 79 columns]</p>	ekstraksi feature dari dataset happy
1 2	<pre> df_sad = feature_extraction(sad_dir, audio_sad, "sad") print(df_sad.head()) </pre> <p>OUTPUT :</p>	ekstraksi feature dari dataset sad

	<pre> name 0 1 2 3 4 \ 0 sad-0001.wav -449.421448 147.722198 -9.568417 51.008976 -2.845285 1 sad-0002.wav -394.924377 111.003624 26.793255 36.478493 1.580411 2 sad-0003.wav -410.265503 120.917496 16.606251 58.323441 -13.436650 3 sad-0004.wav -457.037567 144.196213 16.820171 76.620232 -15.036819 4 sad-0005.wav -432.509460 148.646072 10.928779 63.711994 -9.459482 5 6 7 8 ... 68 69 \ 0 30.098749 -21.938206 16.471960 -9.312575 ... 0.005756 0.006339 1 13.909043 -0.715320 1.693681 -9.720222 ... 0.005456 0.003881 2 26.781693 -15.498982 12.669683 -6.486483 ... 0.018226 0.021288 3 39.247749 -21.616156 13.772663 -12.134876 ... -0.025775 -0.039017 4 28.708157 -16.011524 10.906192 -9.422981 ... -0.012544 0.001674 70 71 72 73 74 75 76 label 0 0.034731 0.004039 -0.023850 0.004825 0.025454 0.045294 0.008083 sad 1 -0.016117 0.045184 0.016784 0.007699 0.015479 0.050308 0.025227 sad 2 0.004438 0.023711 -0.012422 -0.013668 0.005533 0.081349 0.011498 sad 3 -0.028277 -0.001234 -0.018512 0.047282 -0.000458 0.043415 0.008758 sad 4 -0.050659 -0.060595 0.048996 -0.017408 0.034912 0.033458 0.012231 sad [5 rows x 79 columns]</pre>	
1 2 3	<pre> df = pd.concat([df_happy, df_sad]) df.reset_index(drop=True, inplace=True) df OUTPUT :</pre> <pre> name 0 1 2 3 4 5 6 7 8 ... 68 0 happy-0001.wav -314.457886 129.294830 -17.988623 33.474461 -11.482511 11.786289 -18.851305 11.302916 -13.167466 ... 0.008265 1 happy-0002.wav -395.502869 136.012527 2.375357 54.128048 -7.307978 23.552362 -23.973137 14.294265 -10.774254 ... 0.004380 2 happy-0003.wav -366.721985 132.242188 8.187971 50.447689 -0.746726 10.303167 -12.941866 3.299066 -12.599129 ... 0.003033 3 happy-0004.wav -373.125977 141.114120 -5.783168 48.331936 -8.069846 11.951582 -14.797240 5.968351 -8.678323 ... 0.041998 4 happy-0005.wav -338.726593 132.259232 -8.114790 44.194748 -7.917725 5.655037 -16.220108 7.988503 -5.889780 ... -0.003076 ... 2169 sad-1084.wav -417.924408 139.764908 12.273929 52.263084 -17.304499 27.863864 -14.208331 12.585478 -16.029867 ... 0.020251 2170 sad-1085.wav -446.779114 138.702423 27.430937 46.717842 -11.167727 24.803402 -5.096362 12.582688 -7.809765 ... -0.010041 2171 sad-1086.wav -441.363373 139.999771 16.188068 66.505608 -11.208609 26.212502 -17.580864 12.032382 -12.642307 ... -0.032602 2172 sad-1087.wav -408.045013 138.958069 10.271755 55.551472 -16.234924 26.500011 -16.514378 11.222992 -13.605694 ... -0.008891 2173 sad-1088.wav -397.558472 136.763809 12.830576 58.045250 -17.807928 29.873152 -14.585120 2.985977 -12.319318 ... -0.071918 2174 rows x 79 columns</pre>	menggabungkan feature dari dataset happy dan sad dan memberikan label untuk happy dan sad ekspresi
1 2 3 4 5 6 7 8 9 10 11 12	<pre> from sklearn.preprocessing import LabelEncoder label_encoder = LabelEncoder() encoded_y = label_encoder.fit_transform(df["label"]) label = dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.classes_))) print("Mapping of Label Encoded Classes", label, sep="\n") print("Label Encoded Target Variable", encoded_y, sep="\n") OUTPUT :</pre> <pre> Mapping of Label Encoded Classes {'happy': 0, 'sad': 1} Label Encoded Target Variable [0 0 0 ... 1 1 1]</pre>	fungsi untuk melakukan label encoding yaitu merubah label menjadi angka 0 untuk happy dan 1 untuk sad agar dapat dilakukan training model dengan baik
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	<pre> from sklearn.model_selection import cross_validate def cross_validation(model, _X, _y, cv=5): _scoring = ['accuracy', 'precision', 'recall', 'f1'] results = cross_validate(estimator=model, X=_X, y=_y, cv=_cv, scoring=_scoring, return_train_score=True) return {"Training Accuracy scores": results['train_accuracy'], "Mean Training Accuracy": results['train_accuracy'].mean()*100, "Training Precision scores": results['train_precision'],</pre>	fungsi untuk melakukan k-fold cross validation dengan menggunakan model yang telah dibuat dan menghasilkan nilai akurasi, presisi, recall, dan f1 hal ini dilakukan agar model yang dibuat konsisten dan tidak overfitting

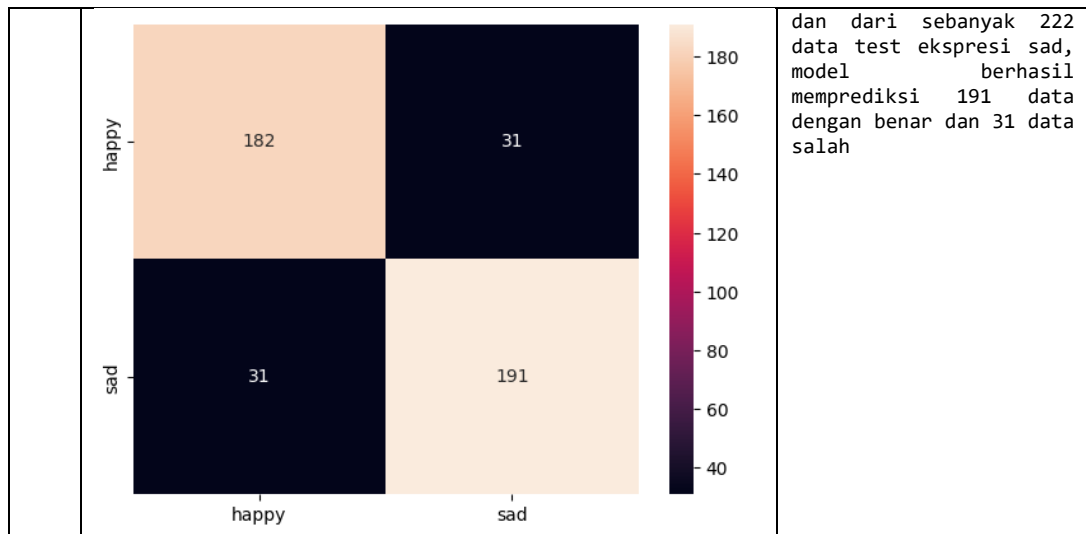
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41	<pre> "Mean Training Precision": results['train_precision'].mean(), "Training Recall scores": results['train_recall'], "Mean Training Recall": results['train_recall'].mean(), "Training F1 scores": results['train_f1'], "Mean Training F1 Score": results['train_f1'].mean(), "Validation Accuracy scores": results['test_accuracy'], "Mean Validation Accuracy": results['test_accuracy'].mean()*100, "Validation Precision scores": results['test_precision'], "Mean Validation Precision": results['test_precision'].mean(), "Validation Recall scores": results['test_recall'], "Mean Validation Recall": results['test_recall'].mean(), "Validation F1 scores": results['test_f1'], "Mean Validation F1 Score": results['test_f1'].mean() } </pre>	
1 2 3 4 5 6 7 8 9 10 11 12	<pre> from sklearn.svm import SVC from sklearn.metrics import accuracy_score def train_svm(X_train, X_test, y_train, y_test, kernel="poly", C=1.0, gamma=None, dfs=None): svm = SVC(kernel=kernel, C=C, gamma=gamma, decision_function_shape=dfs) svm.fit(X_train, y_train) y = svm.predict(X_test) return accuracy_score(y_test, y) </pre>	<p>Fungsi untuk training data set audio ke dalam model SVM yang sudah dibangun kemudian melakukan prediksi berdasarkan data testing. Lalu, mengetahui akurasi dari model dengan fungsi accuracy score</p>
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	<pre> from itertools import product def svm_data(X_train, X_test, y_train, y_test): df = [] gamma = [0.1, 1] dfs = ['ovo', 'ovr'] kernel = ['poly', 'rbf'] c_values = [1, 10] for d, x, y, g in product(dfs, c_values, kernel, gamma): svm = train_svm(X_train, X_test, y_train, y_test, y, x, gamma=g, dfs=d) df.append({ 'Kernel': y, 'C': x, 'Gamma': g, 'Decision Function Shape': d, 'Accuracy (%)': svm.round(2) * 100, }) return pd.DataFrame(df) </pre>	<p>Program untuk melakukan parameter tuning. Parameter tuning meliputi perubahan nilai C (1, 10), nilai gamma (0.1, 1), kernel (polynomial, rbf), dfs ("ovo", "ovr"). Parameter tuning dilakukan untuk menemukan nilai parameter yang optimal untuk model klasifikasi.</p>

	<table><tr><th>Kernel</th><th>C</th><th>Gamma</th><th>Decision Function Shape</th><th>Accuracy (%)</th></tr><tr><td>2</td><td>rbf</td><td>1</td><td>0.1</td><td>ovo</td><td>86.0</td></tr><tr><td>4</td><td>poly</td><td>10</td><td>0.1</td><td>ovo</td><td>86.0</td></tr><tr><td>6</td><td>rbf</td><td>10</td><td>0.1</td><td>ovo</td><td>86.0</td></tr><tr><td>7</td><td>rbf</td><td>10</td><td>1.0</td><td>ovo</td><td>86.0</td></tr><tr><td>10</td><td>rbf</td><td>1</td><td>0.1</td><td>ovr</td><td>86.0</td></tr><tr><td>12</td><td>poly</td><td>10</td><td>0.1</td><td>ovr</td><td>86.0</td></tr><tr><td>14</td><td>rbf</td><td>10</td><td>0.1</td><td>ovr</td><td>86.0</td></tr><tr><td>15</td><td>rbf</td><td>10</td><td>1.0</td><td>ovr</td><td>86.0</td></tr><tr><td>0</td><td>poly</td><td>1</td><td>0.1</td><td>ovo</td><td>85.0</td></tr><tr><td>3</td><td>rbf</td><td>1</td><td>1.0</td><td>ovo</td><td>85.0</td></tr><tr><td>8</td><td>poly</td><td>1</td><td>0.1</td><td>ovr</td><td>85.0</td></tr><tr><td>11</td><td>rbf</td><td>1</td><td>1.0</td><td>ovr</td><td>85.0</td></tr><tr><td>1</td><td>poly</td><td>1</td><td>1.0</td><td>ovo</td><td>81.0</td></tr><tr><td>9</td><td>poly</td><td>1</td><td>1.0</td><td>ovr</td><td>81.0</td></tr><tr><td>5</td><td>poly</td><td>10</td><td>1.0</td><td>ovo</td><td>80.0</td></tr><tr><td>13</td><td>poly</td><td>10</td><td>1.0</td><td>ovr</td><td>80.0</td></tr></table>	Kernel	C	Gamma	Decision Function Shape	Accuracy (%)	2	rbf	1	0.1	ovo	86.0	4	poly	10	0.1	ovo	86.0	6	rbf	10	0.1	ovo	86.0	7	rbf	10	1.0	ovo	86.0	10	rbf	1	0.1	ovr	86.0	12	poly	10	0.1	ovr	86.0	14	rbf	10	0.1	ovr	86.0	15	rbf	10	1.0	ovr	86.0	0	poly	1	0.1	ovo	85.0	3	rbf	1	1.0	ovo	85.0	8	poly	1	0.1	ovr	85.0	11	rbf	1	1.0	ovr	85.0	1	poly	1	1.0	ovo	81.0	9	poly	1	1.0	ovr	81.0	5	poly	10	1.0	ovo	80.0	13	poly	10	1.0	ovr	80.0	Hasil dari fungsi program parameter tuning. Nilai akurasi parameter tertinggi diperoleh senilai 86% dan terkecil 81%.																																																																															
Kernel	C	Gamma	Decision Function Shape	Accuracy (%)																																																																																																																																																																																		
2	rbf	1	0.1	ovo	86.0																																																																																																																																																																																	
4	poly	10	0.1	ovo	86.0																																																																																																																																																																																	
6	rbf	10	0.1	ovo	86.0																																																																																																																																																																																	
7	rbf	10	1.0	ovo	86.0																																																																																																																																																																																	
10	rbf	1	0.1	ovr	86.0																																																																																																																																																																																	
12	poly	10	0.1	ovr	86.0																																																																																																																																																																																	
14	rbf	10	0.1	ovr	86.0																																																																																																																																																																																	
15	rbf	10	1.0	ovr	86.0																																																																																																																																																																																	
0	poly	1	0.1	ovo	85.0																																																																																																																																																																																	
3	rbf	1	1.0	ovo	85.0																																																																																																																																																																																	
8	poly	1	0.1	ovr	85.0																																																																																																																																																																																	
11	rbf	1	1.0	ovr	85.0																																																																																																																																																																																	
1	poly	1	1.0	ovo	81.0																																																																																																																																																																																	
9	poly	1	1.0	ovr	81.0																																																																																																																																																																																	
5	poly	10	1.0	ovo	80.0																																																																																																																																																																																	
13	poly	10	1.0	ovr	80.0																																																																																																																																																																																	
1 2 3 4 5 6 7 8 9 10	<pre>from sklearn.preprocessing import MinMaxScaler from sklearn.model_selection import train_test_split X = df.drop(["name", "label"], axis=1) scaler = MinMaxScaler() X = scaler.fit_transform(X) y = encoded_y X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)</pre>	pembuatan data training dan testing dengan menggunakan fungsi MinMaxScaler untuk melakukan normalisasi data dan train_test_split untuk membagi data menjadi data training dan testing rasio data training dan testing adalah 80:20																																																																																																																																																																																				
1 2	<pre>temp_x = pd.DataFrame(X) temp_x</pre> <p>OUTPUT :</p> <table><tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>...</th><th>67</th><th>68</th><th>69</th></tr><tr><td>0</td><td>0.861386</td><td>0.773441</td><td>0.319268</td><td>0.401871</td><td>0.483258</td><td>0.476217</td><td>0.426332</td><td>0.636291</td><td>0.601741</td><td>0.471916</td><td>...</td><td>0.370065</td><td>0.397031</td><td>0.633576</td></tr><tr><td>1</td><td>0.775929</td><td>0.813626</td><td>0.582723</td><td>0.649824</td><td>0.564598</td><td>0.639785</td><td>0.297822</td><td>0.712349</td><td>0.674125</td><td>0.475777</td><td>...</td><td>0.432850</td><td>0.388118</td><td>0.545738</td></tr><tr><td>2</td><td>0.806276</td><td>0.791072</td><td>0.632214</td><td>0.605640</td><td>0.692443</td><td>0.455599</td><td>0.574603</td><td>0.432785</td><td>0.618931</td><td>0.567910</td><td>...</td><td>0.469115</td><td>0.385027</td><td>0.638285</td></tr><tr><td>3</td><td>0.799524</td><td>0.844144</td><td>0.465186</td><td>0.580240</td><td>0.549753</td><td>0.478515</td><td>0.528050</td><td>0.500654</td><td>0.737518</td><td>0.789288</td><td>...</td><td>0.597929</td><td>0.474423</td><td>0.674231</td></tr><tr><td>4</td><td>0.835796</td><td>0.791174</td><td>0.437311</td><td>0.530571</td><td>0.552717</td><td>0.390982</td><td>0.492350</td><td>0.552019</td><td>0.821859</td><td>0.333096</td><td>...</td><td>0.678438</td><td>0.371010</td><td>0.474328</td></tr><tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td></tr><tr><td>2169</td><td>0.752286</td><td>0.836073</td><td>0.681062</td><td>0.627434</td><td>0.369817</td><td>0.699709</td><td>0.542826</td><td>0.668902</td><td>0.515165</td><td>0.368414</td><td>...</td><td>0.416524</td><td>0.424531</td><td>0.548723</td></tr><tr><td>2170</td><td>0.721861</td><td>0.829717</td><td>0.862266</td><td>0.560862</td><td>0.489391</td><td>0.657177</td><td>0.771450</td><td>0.668831</td><td>0.763788</td><td>0.436935</td><td>...</td><td>0.360472</td><td>0.355031</td><td>0.512256</td></tr><tr><td>2171</td><td>0.727571</td><td>0.837478</td><td>0.727856</td><td>0.798420</td><td>0.488594</td><td>0.676766</td><td>0.458208</td><td>0.654839</td><td>0.617625</td><td>0.552971</td><td>...</td><td>0.548865</td><td>0.303269</td><td>0.573535</td></tr><tr><td>2172</td><td>0.762704</td><td>0.831246</td><td>0.657126</td><td>0.666912</td><td>0.390657</td><td>0.680763</td><td>0.484967</td><td>0.634259</td><td>0.588486</td><td>0.742957</td><td>...</td><td>0.390709</td><td>0.357669</td><td>0.551344</td></tr><tr><td>2173</td><td>0.773761</td><td>0.818120</td><td>0.687717</td><td>0.696851</td><td>0.360008</td><td>0.727655</td><td>0.533373</td><td>0.424824</td><td>0.627394</td><td>0.628028</td><td>...</td><td>0.469974</td><td>0.213067</td><td>0.544434</td></tr></table> <p>2174 rows x 77 columns</p>		0	1	2	3	4	5	6	7	8	9	...	67	68	69	0	0.861386	0.773441	0.319268	0.401871	0.483258	0.476217	0.426332	0.636291	0.601741	0.471916	...	0.370065	0.397031	0.633576	1	0.775929	0.813626	0.582723	0.649824	0.564598	0.639785	0.297822	0.712349	0.674125	0.475777	...	0.432850	0.388118	0.545738	2	0.806276	0.791072	0.632214	0.605640	0.692443	0.455599	0.574603	0.432785	0.618931	0.567910	...	0.469115	0.385027	0.638285	3	0.799524	0.844144	0.465186	0.580240	0.549753	0.478515	0.528050	0.500654	0.737518	0.789288	...	0.597929	0.474423	0.674231	4	0.835796	0.791174	0.437311	0.530571	0.552717	0.390982	0.492350	0.552019	0.821859	0.333096	...	0.678438	0.371010	0.474328	2169	0.752286	0.836073	0.681062	0.627434	0.369817	0.699709	0.542826	0.668902	0.515165	0.368414	...	0.416524	0.424531	0.548723	2170	0.721861	0.829717	0.862266	0.560862	0.489391	0.657177	0.771450	0.668831	0.763788	0.436935	...	0.360472	0.355031	0.512256	2171	0.727571	0.837478	0.727856	0.798420	0.488594	0.676766	0.458208	0.654839	0.617625	0.552971	...	0.548865	0.303269	0.573535	2172	0.762704	0.831246	0.657126	0.666912	0.390657	0.680763	0.484967	0.634259	0.588486	0.742957	...	0.390709	0.357669	0.551344	2173	0.773761	0.818120	0.687717	0.696851	0.360008	0.727655	0.533373	0.424824	0.627394	0.628028	...	0.469974	0.213067	0.544434	hasil data training X modal yang sudah dilakukan scaling dengan MinMaxScaler agar data yang dihasilkan tidak terlalu besar dan hasil data training y model yang sudah dilakukan label encoding
	0	1	2	3	4	5	6	7	8	9	...	67	68	69																																																																																																																																																																								
0	0.861386	0.773441	0.319268	0.401871	0.483258	0.476217	0.426332	0.636291	0.601741	0.471916	...	0.370065	0.397031	0.633576																																																																																																																																																																								
1	0.775929	0.813626	0.582723	0.649824	0.564598	0.639785	0.297822	0.712349	0.674125	0.475777	...	0.432850	0.388118	0.545738																																																																																																																																																																								
2	0.806276	0.791072	0.632214	0.605640	0.692443	0.455599	0.574603	0.432785	0.618931	0.567910	...	0.469115	0.385027	0.638285																																																																																																																																																																								
3	0.799524	0.844144	0.465186	0.580240	0.549753	0.478515	0.528050	0.500654	0.737518	0.789288	...	0.597929	0.474423	0.674231																																																																																																																																																																								
4	0.835796	0.791174	0.437311	0.530571	0.552717	0.390982	0.492350	0.552019	0.821859	0.333096	...	0.678438	0.371010	0.474328																																																																																																																																																																								
...																																																																																																																																																																								
2169	0.752286	0.836073	0.681062	0.627434	0.369817	0.699709	0.542826	0.668902	0.515165	0.368414	...	0.416524	0.424531	0.548723																																																																																																																																																																								
2170	0.721861	0.829717	0.862266	0.560862	0.489391	0.657177	0.771450	0.668831	0.763788	0.436935	...	0.360472	0.355031	0.512256																																																																																																																																																																								
2171	0.727571	0.837478	0.727856	0.798420	0.488594	0.676766	0.458208	0.654839	0.617625	0.552971	...	0.548865	0.303269	0.573535																																																																																																																																																																								
2172	0.762704	0.831246	0.657126	0.666912	0.390657	0.680763	0.484967	0.634259	0.588486	0.742957	...	0.390709	0.357669	0.551344																																																																																																																																																																								
2173	0.773761	0.818120	0.687717	0.696851	0.360008	0.727655	0.533373	0.424824	0.627394	0.628028	...	0.469974	0.213067	0.544434																																																																																																																																																																								
1 2 3	<pre>svm_score = svm_data(X_train,X_test, y_train, y_test) svm_score = svm_score.sort_values(by=['Accuracy (%)'], ascending=False) OUTPUT :</pre>	dilakukan uji score untuk memilih model yang terbaik model yang terbaik adalah model yang memiliki nilai akurasi paling tinggi dan akan digunakan untuk melakukan training model																																																																																																																																																																																				

1	<pre>svm_score.head(5)</pre> <p>OUTPUT :</p> <table><thead><tr><th></th><th>Kernel</th><th>C</th><th>Gamma</th><th>Decision Function Shape</th><th>Accuracy (%)</th></tr></thead><tbody><tr><td>95</td><td>rbf</td><td>10.0</td><td>scale</td><td>ovr</td><td>0.86</td></tr><tr><td>92</td><td>rbf</td><td>10.0</td><td>0.1</td><td>ovr</td><td>0.86</td></tr><tr><td>58</td><td>rbf</td><td>1000.0</td><td>auto</td><td>ovo</td><td>0.86</td></tr><tr><td>55</td><td>rbf</td><td>1000.0</td><td>0.001</td><td>ovo</td><td>0.86</td></tr><tr><td>104</td><td>rbf</td><td>100.0</td><td>0.1</td><td>ovr</td><td>0.86</td></tr></tbody></table>		Kernel	C	Gamma	Decision Function Shape	Accuracy (%)	95	rbf	10.0	scale	ovr	0.86	92	rbf	10.0	0.1	ovr	0.86	58	rbf	1000.0	auto	ovo	0.86	55	rbf	1000.0	0.001	ovo	0.86	104	rbf	100.0	0.1	ovr	0.86	dapat dilihat pada hasil uji score bahwa model dengan kernel rbf mendominasi nilai akurasi tertinggi di sini menggunakan kernel rbf, C=10, gamma=0.1, dfs=ovr dengan akurasi sebesar 0.86
	Kernel	C	Gamma	Decision Function Shape	Accuracy (%)																																	
95	rbf	10.0	scale	ovr	0.86																																	
92	rbf	10.0	0.1	ovr	0.86																																	
58	rbf	1000.0	auto	ovo	0.86																																	
55	rbf	1000.0	0.001	ovo	0.86																																	
104	rbf	100.0	0.1	ovr	0.86																																	
1 2 3	<pre>svm = SVC(kernel="rbf", C=10, gamma=0.1, decision_function_shape="ovr") svm.fit(X_train, y_train)</pre> <p>OUTPUT:</p> 	memilih model yang terbaik untuk dilakukan training																																				
1 2 3 4 5 6 7 8 9	<pre>result = cross_validation(svm, X, y, 5) for k, v in result.items(): if isinstance(v, np.ndarray): for I, val in enumerate(v): print(f'{k} {i+1}: {val}') else: print(k,v) print(k,v) print("-"*50)</pre> <p>OUTPUT :</p> <pre>Training Accuracy scores 1: 0.8976423231742381 Training Accuracy scores 2: 0.9045428407130535 Training Accuracy scores 3: 0.894767107533065 Training Accuracy scores 4: 0.8953421506612996 Training Accuracy scores 5: 0.8971264367816092 Mean Training Accuracy 89.7884171726532 ----- Training Precision scores 1: 0.8827433628318584 Training Precision scores 2: 0.8911111111111111 Training Precision scores 3: 0.8736383442265795 Training Precision scores 4: 0.8837988826815643 Training Precision scores 5: 0.8895152198421646 Mean Training Precision 0.8841613841386555 ----- Training Recall scores 1: 0.9172413793103448 Training Recall scores 2: 0.9218390804597701 Training Recall scores 3: 0.9228998849252014 Training Recall scores 4: 0.9102416570771001 Training Recall scores 5: 0.906896551724138 Mean Training Recall 0.9158237106993109 ----- Training F1 scores 1: 0.8996617812852311 Training F1 scores 2: 0.9062146892655367 Training F1 scores 3: 0.8975937325125909 Training F1 scores 4: 0.896825396825397 ... Validation F1 scores 4: 0.8769574944071588 Validation F1 scores 5: 0.8939051918735892 Mean Validation F1 Score 0.8733433714541089 ----- Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...</pre>	berikut adalah hasil training model yang dilakukan dengan pemilihan model terbaik yang sudah dilakukan sebelumnya																																				
1 2 3 4 5 6 7 8 9 10 11	<pre>import matplotlib.pyplot as plt def plot_result(x_label, y_label, plot_title, train_data, val_data): plt.figure(figsize=(12,6)) labels = ["1st Fold", "2nd Fold", "3rd Fold", "4th Fold", "5th Fold"] X_axis = np.arange(len(labels)) ax = plt.gca() plt.ylim(0.40000, 1) plt.bar(X_axis-0.2, train_data, 0.4, color='blue', label='Training')</pre>	fungsi untuk menampilkan plot hasil perbandingan training dan validation dari model																																				

12 13 14 15 16 17 18 19 20	<pre> plt.bar(X_axis+0.2, val_data, 0.4, color='red', label='Validation') plt.title(plot_title, fontsize=30) plt.xticks(X_axis, labels) plt.xlabel(x_label, fontsize=14) plt.ylabel(y_label, fontsize=14) plt.legend() plt.grid(True) plt.show() </pre>																			
1 2 3	<pre> plot_result(svm, "Accuracy", "Accuracy scores in 5 Folds", result["Training Accuracy scores"], result["Validation Accuracy scores"]) OUTPUT : </pre>  <p>The bar chart displays accuracy scores for 5 folds. The y-axis represents Accuracy from 0.4 to 1.0. The x-axis shows 1st Fold, 2nd Fold, 3rd Fold, 4th Fold, and 5th Fold. For each fold, there are two bars: a blue bar for Training and a red bar for Validation. The Training accuracy is consistently higher than the Validation accuracy across all folds.</p> <table border="1"> <thead> <tr> <th>Fold</th> <th>Training Accuracy</th> <th>Validation Accuracy</th> </tr> </thead> <tbody> <tr> <td>1st Fold</td> <td>0.90</td> <td>0.86</td> </tr> <tr> <td>2nd Fold</td> <td>0.90</td> <td>0.84</td> </tr> <tr> <td>3rd Fold</td> <td>0.89</td> <td>0.88</td> </tr> <tr> <td>4th Fold</td> <td>0.89</td> <td>0.87</td> </tr> <tr> <td>5th Fold</td> <td>0.89</td> <td>0.89</td> </tr> </tbody> </table> <p>SVC(C=10, gamma=0.1)</p>	Fold	Training Accuracy	Validation Accuracy	1st Fold	0.90	0.86	2nd Fold	0.90	0.84	3rd Fold	0.89	0.88	4th Fold	0.89	0.87	5th Fold	0.89	0.89	menampilkan grafik akurasi dalam 5 folds
Fold	Training Accuracy	Validation Accuracy																		
1st Fold	0.90	0.86																		
2nd Fold	0.90	0.84																		
3rd Fold	0.89	0.88																		
4th Fold	0.89	0.87																		
5th Fold	0.89	0.89																		
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	<pre> def prediction(model, df): temp = df X = temp.drop(["name", "label"], axis=1) X = scaler.transform(X) print(X.shape) y = np.array(temp['label'].apply(lambda x: 1 if x == "sad" else 0)) y_pred = model.predict(X) print(y_pred) for i in range(len(y)): label = "happy" if y_pred[i] == 0 else "sad" print(f"Prediction {temp['name'][i]} : {label}") </pre>	fungsi untuk memprediksi data baru dengan menggunakan model yang sudah dilakukan training dan scaler yang digunakan untuk melakukan scaling data																		
1	<pre> prediction(svm, df) OUTPUT : </pre>	menampilkan prediksi data set audio yang dimasukkan oleh user																		

	<pre>(2174, 77) [0 0 0 ... 1 1 1] Prediction happy-0001.wav : happy Prediction happy-0002.wav : happy Prediction happy-0003.wav : happy Prediction happy-0004.wav : happy Prediction happy-0005.wav : happy Prediction happy-0006.wav : happy Prediction happy-0007.wav : happy Prediction happy-0008.wav : happy Prediction happy-0009.wav : happy Prediction happy-0010.wav : happy Prediction happy-0011.wav : happy Prediction happy-0012.wav : happy Prediction happy-0013.wav : happy Prediction happy-0014.wav : happy Prediction happy-0015.wav : happy Prediction happy-0016.wav : happy Prediction happy-0017.wav : sad Prediction happy-0018.wav : happy Prediction happy-0019.wav : happy Prediction happy-0020.wav : happy Prediction happy-0021.wav : happy Prediction happy-0022.wav : sad Prediction happy-0023.wav : happy ... Prediction sad-1085.wav : sad Prediction sad-1086.wav : sad Prediction sad-1087.wav : sad Prediction sad-1088.wav : sad</pre>																															
1 2 3 4 5 6 7 8 9 10 11 12	<pre>from sklearn.metrics import classification_report, confusion_matrix import seaborn as sns pred = svm.predict(X_test) report = classification_report(y_test, pred, output_dict=True) report = pd.DataFrame(report).transpose() report = report.rename(index={'0': 'happy', '1': 'sad'}) report OUTPUT:</pre> <table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>happy</td><td>0.854460</td><td>0.854460</td><td>0.854460</td><td>213.000000</td></tr><tr><td>sad</td><td>0.860360</td><td>0.860360</td><td>0.860360</td><td>222.000000</td></tr><tr><td>accuracy</td><td>0.857471</td><td>0.857471</td><td>0.857471</td><td>0.857471</td></tr><tr><td>macro avg</td><td>0.857410</td><td>0.857410</td><td>0.857410</td><td>435.000000</td></tr><tr><td>weighted avg</td><td>0.857471</td><td>0.857471</td><td>0.857471</td><td>435.000000</td></tr></tbody></table>		precision	recall	f1-score	support	happy	0.854460	0.854460	0.854460	213.000000	sad	0.860360	0.860360	0.860360	222.000000	accuracy	0.857471	0.857471	0.857471	0.857471	macro avg	0.857410	0.857410	0.857410	435.000000	weighted avg	0.857471	0.857471	0.857471	435.000000	
	precision	recall	f1-score	support																												
happy	0.854460	0.854460	0.854460	213.000000																												
sad	0.860360	0.860360	0.860360	222.000000																												
accuracy	0.857471	0.857471	0.857471	0.857471																												
macro avg	0.857410	0.857410	0.857410	435.000000																												
weighted avg	0.857471	0.857471	0.857471	435.000000																												
1 2 3 4 5	<pre>cfmatrix = confusion_matrix(y_test, pred) cfmatrix = pd.DataFrame(cfmatrix, index=['happy', 'sad'], columns=['happy', 'sad']) sns.heatmap(cfmatrix, annot=True, fmt='g') plt.show() OUTPUT:</pre>	<p>dari hasil confusion matrix dilihat bahwa dari total 435 data test yang ada sebanyak 213 data test ekspresi happy, model berhasil memprediksi 182 data dengan benar dan 31 data salah.</p>																														



File 'test_model.ipynb' merupakan bagian utama dalam implementasi sistem identifikasi sentimen positif dan negatif pada audio. Melalui proses pelatihan model, ekstraksi fitur MFCC, parameter tuning, dan evaluasi menggunakan *confusion matrix*, file ini membantu membangun sistem yang dapat mengenali dan membedakan sentimen dalam audio dengan akurasi yang tinggi.

2.3.2 File svm.py

File 'svm.py' merupakan file yang berisi implementasi fungsi-fungsi untuk ekstraksi fitur audio, normalisasi fitur, dan prediksi sentimen menggunakan model *Support Vector Machine* (SVM).

SOURCE CODE : svm.py

No	Source code	Penjelasan
1	import librosa	Pada baris 5 terdapat fungsi yang menggunakan library 'librosa' untuk menghitung zero-crossing rate dalam audio. Fungsi ini mengambil parameter audio (sinyal audio), frame (panjang frame), dan hop (langkah).
2	import pickle	
3	import numpy as np	
4		
5	def zcr(audio, frame=2048, hop=512):	Pada baris 11 terdapat Fungsi yang juga menggunakan library 'librosa' untuk menghitung nilai Root Mean Square (RMS) dari audio. RMS adalah ukuran dari tingkat amplitudo
6	zcr = librosa.feature.zero_crossing_rate(y=audio,	
7	frame_length=frame, hop_length=hop)[0]	
8		
9	return zcr	
10		
11	def rms(audio, frame=2048, hop=512):	
12	rms = librosa.feature.rms(y=audio,	
13	frame_length=frame, hop_length=hop)[0]	
14		
15	return rms	
16		
17	def mfcc(audio, sr, frame=2048, hop=512, mfcc_num=25):	

<pre> 18 mfcc_spectrum = librosa.feature.mfcc(y=audio, sr=sr, 19 n_fft=frame, hop_length=hop, n_mfcc=mfcc_num) 20 21 delt1 = librosa.feature.delta(mfcc_spectrum, order=1) 22 delt2 = librosa.feature.delta(mfcc_spectrum, order=2) 23 24 mfcc_feature = np.concatenate((np.mean(mfcc_spectrum, 25 axis=1), np.mean(delt1, axis=1), np.mean(delt2, axis=1))) 26 27 return mfcc_feature 28 29 def extract_feature(audio, sr, frame=2048, hop=512, 30 mfcc_num=25): 31 audio_mfcc, audio_zcr, audio_rmse = [],[],[] 32 33 mfcc_score = mfcc(audio, sr, frame=frame, hop=hop, 34 mfcc_num=mfcc_num) 35 zcr_score = np.mean(zcr(audio, frame=frame, hop=hop)) 36 rmse_score = np.mean(rms(audio, frame=frame, 37 hop=hop)) 38 39 audio_mfcc.append(mfcc_score) 40 audio_zcr.append(zcr_score) 41 audio_rmse.append(rmse_score) 42 43 feature = np.column_stack((audio_mfcc, audio_zcr, 44 audio_rmse)) 45 46 return feature 47 48 def prediction_data(model, df): 49 y_pred = model.predict(df) 50 51 labels = { 52 0: "Bahagia", 53 1: "Sedih" 54 } 55 pred = labels[y_pred[0]] 56 57 return pred 58 59 def normalize_feature(feature): 60 scaler = 61 pickle.load(open("scaler_svm_2000data_new_1.pkl", "rb")) 62 feature = scaler.transform(feature) 63 64 return feature 65 66 def prediction(audio, sr): 67 model = 68 pickle.load(open("svm_2000data_new_1_fix.pkl", "rb")) 69 70 df = extract_feature(audio, sr) 71 x = normalize_feature(df) 72 pred = prediction_data(model, x) 73 74 return pred 75 76 77 78 79 </pre>	<p>rata-rata dalam audio. Fungsi ini mengambil parameter audio, frame, dan hop.</p> <p>Pada baris 17 terdapat Fungsi mfcc yang juga menggunakan library 'librosa' untuk menghitung Mel-frequency Cepstral Coefficients (MFCC) dari audio. MFCC adalah representasi numerik dari sinyal audio yang merepresentasikan karakteristik frekuensi dan spektralnya. Fungsi ini mengambil parameter audio (sinyal audio), sr (tingkat sampel), frame, hop, dan mfcc_num (jumlah koefisien MFCC yang dihasilkan).</p> <p>Pada baris 29 terdapat fungsi yang digunakan untuk mengekstraksi fitur-fitur audio dari suatu audio dengan menggunakan fungsi-fungsi sebelumnya. Fitur yang diekstraksi meliputi MFCC, zero-crossing rate, dan RMS. Output dari fungsi ini adalah vektor fitur</p> <p>Pada baris 48 terdapat fungsi untuk melakukan prediksi sentimen (bahagia atau sedih) berdasarkan model SVM yang telah dilatih sebelumnya. Fungsi ini akan mengembalikan label prediksi yang sesuai dengan sentimen yang diprediksi.</p> <p>Pada baris 66 terdapat fungsi untuk memprediksi sentimen pada suatu audio menggunakan model SVM. Fungsi ini akan memuat model SVM yang telah dilatih sebelumnya dan menggunakan fungsi-fungsi lainnya untuk melakukan ekstraksi fitur, normalisasi, dan prediksi sentimen.</p>
--	---

Dengan menggunakan *file* 'svm.py', dapat dilakukan prediksi sentimen (bahagia atau sedih) berdasarkan audio yang diberikan. *File* ini menyediakan fungsi-fungsi yang memungkinkan ekstraksi fitur audio, normalisasi fitur, dan prediksi sentimen yang akurat menggunakan metode SVM.

2.3.3 File preprocessing.py

File 'preprocessing.py' berisi implementasi fungsi-fungsi yang digunakan untuk melakukan pra-pemrosesan (*preprocessing*) pada audio sebelum menjalankan algoritma *Support Vector Machine* (SVM) pada sistem identifikasi sentimen positif dan negatif.

SOURCE CODE : preprocessing.py

No	Source code	Penjelasan
1	import matplotlib.pyplot as plt	Setiap fungsi menggunakan <i>library</i>
2	import librosa	matplotlib untuk membuat visualisasi grafik,
3	import numpy as np	serta menggunakan <i>library</i> librosa untuk melakukan ekstraksi
4		fitur audio seperti RMS, ZCR, dan MFCC.
5	def display_waveform(audio, sr):	
6	fig, ax = plt.subplots(Pada baris 5 terdapat fungsi untuk menampilkan representasi gelombang
7	nrows = 1,	suara (waveform) dari audio. Gelombang suara direpresentasikan
8	ncols = 1,	sebagai grafik amplitudo terhadap waktu.
9	figsize = (20, 5),	
10	sharex = True,	
11	sharey = True,	
12	squeeze = True	
13)	
14		
15	ax.plot(audio)	
16	ax.set_title('Waveform')	
17	ax.set_xlabel('Time')	
18	ax.set_ylabel('Amplitude')	
19		
20	plt.tight_layout()	
21		
22	return fig	
23		
24	def display_rms(audio, frame=2048, hop=512):	Pada baris 24 terdapat fungsi untuk menampilkan nilai Root Mean Square (RMS) dari audio. RMS merupakan ukuran dari
25	rms = librosa.feature.rms(y=audio,	kekuatan sinyal audio pada setiap frame waktu tertentu.
26	frame_length=frame, hop_length=hop)[0]	
27		
28	fig, ax = plt.subplots(Pada baris 46 terdapat fungsi untuk menampilkan nilai Zero Crossing Rate (ZCR) dari audio. ZCR mengindikasikan jumlah perubahan tanda (dari positif ke negatif atau
29	nrows = 1,	
30	ncols = 1,	
31	figsize = (20, 5),	
32	sharex = True,	
33	sharey = True,	
34	squeeze = True	
35)	
36		
37	ax.plot(rms)	
38	ax.set_title('RMS')	
39	ax.set_xlabel('Time')	
40	ax.set_ylabel('Amplitude')	
41		
42	plt.tight_layout()	

43		sebaliknya) dalam audio
44	return fig	pada setiap frame waktu.
45		
46	def display_zcr(audio, frame=2048, hop=512):	
47	zcr = librosa.feature.zero_crossing_rate(y=audio,	
48	frame_length=frame, hop_length=hop)[0]	
49		
50	fig, ax = plt.subplots(Pada baris 68 terdapat
51	nrows = 1,	Fungsi untuk menampilkan
52	ncols = 1,	Mel-frequency Cepstral
53	figsize = (20, 5),	Coefficients (MFCC) dari
54	sharex = True,	audio. MFCC adalah
55	sharey = True,	representasi frekuensi
56	squeeze = True	audio dalam domain
57)	cepstral yang banyak
58		digunakan dalam
59	ax.plot(zcr)	pengenalan suara. Fungsi
60	ax.set_title('ZCR')	ini juga menampilkan
61	ax.set_xlabel('Time')	nilai Delta 1 dan Delta
62	ax.set_ylabel('Amplitude')	2, yang merupakan
63		turunan pertama dan
64	plt.tight_layout()	kedua dari MFCC. Pada
65		fungsi ini, audio input
66	return fig	dan tingkat sampling
67		audio (sr) diberikan
68	def display_mfcc(audio, sr, frame=2048, hop=512,	sebagai argumen. Selain
69	mfcc_num=25):	itu, terdapat beberapa
70	mfcc_spectrum = librosa.feature.mfcc(y=audio, sr=sr,	argumen opsional,
71	n_fft=frame, hop_length=hop, n_mfcc=mfcc_num)	seperti frame yang
72		menentukan panjang frame
73	delt1 = librosa.feature.delta(mfcc_spectrum,	dalam pemrosesan audio,
74	order=1)	hop yang menentukan
75	delt2 = librosa.feature.delta(mfcc_spectrum,	jumlah sampel yang
76	order=2)	dilewati antara dua
77		frame, dan mfcc_num yang
78	mfcc_feature =	menentukan jumlah
79	np.concatenate((np.mean(mfcc_spectrum, axis=1),	koefisien MFCC yang
80	np.mean(delt1, axis=1), np.mean(delt2, axis=1)))	dihasilkan. Selain itu,
81		fungsi ini juga
82	fig, ax = plt.subplots(menghitung turunan
83	nrows = 3,	pertama dan kedua dari
84	ncols = 1,	spektrum MFCC, yang
85	figsize = (10, 15),	disebut Delta 1 dan Delta
86	sharex = True,	2. Delta 1 merupakan
87	sharey = True,	turunan pertama dari
88	squeeze = True	MFCC, sementara Delta 2
89)	merupakan turunan kedua.
90		Turunan ini memberikan
91	ax[0].plot(delt1)	informasi tentang
92	ax[0].set_title('MFCC')	perubahan cepstral antar
93	ax[0].set_xlabel('Time')	frame dan dapat membantu
94	ax[0].set_ylabel('Amplitude')	dalam menangkap
95		informasi dinamis dari
96	ax[1].plot(delt2)	audio. Grafik yang
97	ax[1].set_title('MFCC')	dihasilkan oleh fungsi
98	ax[1].set_xlabel('Time')	ini terdiri dari tiga
99	ax[1].set_ylabel('Amplitude')	bagian. Bagian pertama
100		menampilkan Delta 1,
101	ax[2].plot(mfcc_spectrum)	bagian kedua menampilkan
102	ax[2].set_title('MFCC')	Delta 2, dan bagian
103	ax[2].set_xlabel('Time')	ketiga menampilkan
104	ax[2].set_ylabel('Amplitude')	spektrum MFCC itu
105		sendiri. Setiap grafik
106	plt.tight_layout()	menunjukkan perubahan
107		amplitudo terhadap
108	return fig	waktu.

Dengan menggunakan *file* 'preprocessing.py' ini, audio dapat diolah sebelum diteruskan ke model SVM untuk proses klasifikasi sentimen positif dan negatif. Pra-pemrosesan audio ini membantu dalam pemahaman lebih lanjut tentang karakteristik audio sebelum penggunaan SVM, sehingga dapat meningkatkan performa sistem identifikasi sentimen.

2.3.4 File main.py

File 'main.py' merupakan *file* utama dalam implementasi aplikasi Identifikasi Emosi berdasarkan Suara Orang. Pada *file* ini, digunakan framework Streamlit untuk membangun antarmuka pengguna. Aplikasi ini menggunakan algoritma SVM (*Support Vector Machine*) untuk mengidentifikasi emosi berdasarkan suara orang.

SOURCE CODE : main.py

No	Source code	Penjelasan
1	import streamlit as st	Pada baris 5 dilakukan konfigurasi tampilan aplikasi menggunakan fungsi set_page_config dari Streamlit. Tampilan aplikasi diberi judul "Audio Processing SVM Algorithm", ikon menggunakan emoji ":musical_note:", dan tata letak (layout) aplikasi disetel sebagai "wide".
2	from tab_one import content_tab_one	
3	from tab_two import content_tab_two	
4		
5	st.set_page_config(
6	page_title="Audio Processing SVM Algorithm",	Pada baris 13 dan 14 ditampilkan judul aplikasi "Identifikasi Emosi berdasarkan Suara Orang" dan deskripsi singkat mengenai aplikasi tersebut. Paragraf berikutnya memberikan jarak (spacing) menggunakan tag markdown untuk estetika tampilan.
7	page_icon=":musical_note:",	
8	layout="wide",	
9)	
10		
11		Pada baris 21 diatur sidebar (sisi sebelah kiri), untuk menampilkan informasi mengenai kelompok yang membuat aplikasi. Terdapat pilihan untuk memilih penulis (author), library yang digunakan dalam pembuatan aplikasi, dan dataset
12		
13	st.title('Identifikasi Emosi berdasarkan Suara Orang')	
14	st.write('Aplikasi ini dibuat dengan menggunakan algoritma SVM (<i>Support Vector Machine</i>) untuk mengidentifikasi emosi berdasarkan suara orang.')	
15	st.markdown(' ', unsafe_allow_html=True)	
16		
17		
18		
19		
20		
21	with st.sidebar:	
22	st.header("Kelompok 2")	
23	author = st.multiselect(
24	'Author',	
25	['Gung Krisna', 'Alex'],	
26	['Gung Krisna', 'Alex'])	
27		
28	library = st.multiselect(
29	'Library Used',	
30	['Librosa', 'Streamlit', 'Sklearn', 'Numpy',	
31	'Pandas'],	
32	['Librosa', 'Streamlit', 'Sklearn', 'Numpy',	
33	'Pandas'])	
34		
35	dataset = st.multiselect(
36	'Dataset Used :blue[2000 audio dataset]',	
37	['1000 Happy', '1000 Sad'],	
38	['1000 Happy', '1000 Sad'])	

39)	yang digunakan (dalam hal ini, dataset terdiri dari 2000 audio dengan 1000 audio senang dan 1000 audio sedih).
40		
41		
42		
43	tab1, tab2 = st.tabs(['Identifikasi	
44	Emosi', 'Dokumentasi'])	
45		
46	with tab1:	Pada baris 43 dibuat dua tab dengan judul
47	content_tab_one('Identifikasi Emosi')	"Identifikasi Emosi" dan
48		"Dokumentasi"
49	with tab2:	menggunakan fungsi
50	content_tab_two('Dokumentasi')	st.tabs. Tab pertama
51		berisi konten dari file
		'tab_one.py' yang
		menjelaskan tentang
		identifikasi emosi. Tab
		kedua berisi konten dari
		file 'tab_two.py' yang
		berfungsi sebagai
		dokumentasi aplikasi.

Secara keseluruhan, *file* 'main.py' merupakan *file* utama yang mengatur tampilan aplikasi dan menghubungkan dengan konten-konten terkait. Dengan menggunakan Streamlit, aplikasi ini memberikan pengguna kemudahan untuk mengidentifikasi emosi berdasarkan suara orang melalui algoritma SVM.

2.3.5 File tab_one.py

File 'tab_one.py' adalah sebuah *file* Python yang digunakan untuk mengimplementasikan bagian frontend dari sistem identifikasi emosi pada suara menggunakan metode *Support Vector Machine* (SVM). *File* ini menggunakan *library* Streamlit untuk membangun antarmuka pengguna yang interaktif.

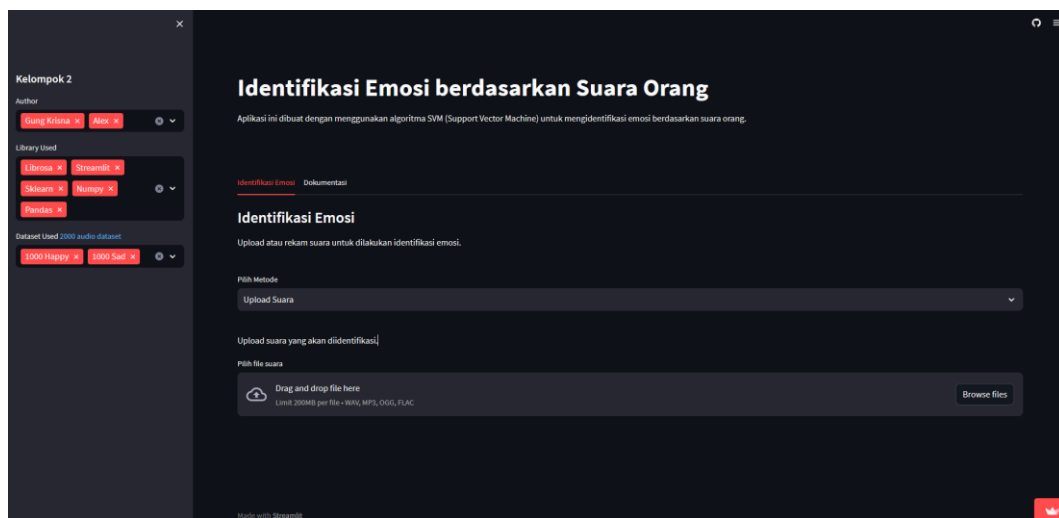
SOURCE CODE : tab_one.py

No	Source code	Penjelasan
1	import streamlit as st	
2	from svm import extract_feature, prediction,	
3	normalize_feature	
4	from preprocessing import display_waveform, display_rms,	
5	display_zcr, display_mfcc	
6	import librosa	
7	import pandas as pd	
8		
9	def tab_hasil(audio, sr):	Pada baris 9 terdapat
10	tab_hasil, tab_preprocessing, tab_feature =	Fungsi 'tab_hasil'
11	st.tabs(['Hasil Identifikasi', 'Preprocessing Audio',	digunakan untuk
12	'Feature Extraction'])	menampilkan hasil
13		identifikasi emosi. Pada
14	with tab_hasil:	bagian ini, prediksi
15	st.subheader("Hasil Identifikasi Emosi")	emosi dilakukan dengan
16	pred = prediction(audio, sr)	memanggil fungsi
17		'prediction' yang
		menggunakan model SVM
		yang telah dilatih

18	st.write("Emosi yang dihasilkan merupakan emosi")	sebelumnya. Hasil
19	st.write(f"<h4>{pred}</h4>", unsafe_allow_html=True)	prediksi emosi
20		ditampilkan pada
21	with tab_preprocessing:	antarmuka pengguna.
22	st.subheader("Preprocessing Audio")	
23	st.markdown(' ', unsafe_allow_html=True)	Pada baris 21 terdapat
24		Fungsi
25	st.write('Waveform')	'tab_preprocessing'
26	waveform = display_waveform(audio, sr)	digunakan untuk
27	st.pyplot(waveform)	menampilkan visualisasi
28		dari proses
29	st.write('---')	preprocessing audio.
30		Pada bagian ini,
31	st.write('Root Mean Square Energy (RMSE)')	terdapat beberapa
32	disp_rms = display_rms(audio)	visualisasi yang
33	st.pyplot(disp_rms)	ditampilkan, seperti
34		waveform, root mean
35	st.write('---')	square energy (RMSE),
36		zero crossing rate
37	st.write('Zero Crossing Rate (ZCR)')	(ZCR), dan Mel-frequency
38	disp_zcr = display_zcr(audio)	Cepstral Coefficients
39	st.pyplot(disp_zcr)	(MFCC). Visualisasi ini
40		membantu pengguna
41	st.write('---')	memahami karakteristik
42		audio yang digunakan
43	st.write('Display Mel Spectrogram(MFCC)')	dalam identifikasi
44	disp_mfcc = display_mfcc(audio, sr)	emosi.
45	st.pyplot(disp_mfcc)	
46		Pada baris 47 terdapat
47	with tab_feature:	Fungsi 'tab_feature'
48	st.subheader("Feature Extraction")	digunakan untuk
49	st.markdown(' ', unsafe_allow_html=True)	menampilkan hasil
50		ekstraksi fitur audio.
51	st.write('Feature Extraction')	Pada bagian ini, fitur
52	feature = extract_feature(audio, sr)	audio diekstraksi
53	feature = pd.DataFrame(feature)	menggunakan fungsi
54	st.dataframe(feature)	'extract_feature' dan
55		ditampilkan dalam bentuk
56	st.write('Hasil Normalize Feature Extraction')	dataframe. Selanjutnya,
57	normalize = normalize_feature(feature)	fitur tersebut
58	st.dataframe(normalize)	dinormalisasi
59		menggunakan fungsi
60	st.write("Jumlah fitur yang dihasilkan")	'normalize_feature' dan
61	st.subheader(feature.shape)	ditampilkan dalam
62		dataframe terpisah.
63	def upload_file(uploaded_file):	Jumlah fitur yang
64	st.audio(uploaded_file, format='audio/ogg')	dihasilkan juga
65	st.markdown(' ', unsafe_allow_html=True)	ditampilkan pada
66		antarmuka pengguna.
67	audio, sr = librosa.load(uploaded_file)	
68		Pada baris 63 terdapat
69	tab_hasil(audio, sr)	fungsi 'upload_file'
70		digunakan untuk
71		mengunggah file audio
72	def content_tab_one(title):	yang akan digunakan
73	st.subheader(title)	untuk identifikasi
74		emosi. File audio
75	st.write('Upload atau rekam suara untuk dilakukan	diunggah menggunakan
76	identifikasi emosi.')	komponen 'file_uploader'
77	st.markdown(' ', unsafe_allow_html=True)	dari Streamlit.
78		
79	select = st.selectbox('Pilih Metode', ['Upload Suara'])	
80	st.markdown(' ', unsafe_allow_html=True)	
81		
82	if select == 'Upload Suara':	
83	st.write('Upload suara yang akan diidentifikasi.')	
84		
85	audio_types = ["wav", "mp3", "ogg", "flac"]	

86	<code>uploaded_file = st.file_uploader("Pilih file suara",</code>	
87	<code>type=audio_types)</code>	
88		
89	<code>if uploaded_file is not None:</code>	
90	<code>upload_file(uploaded_file)</code>	

File 'tab_one.py' merupakan komponen penting dalam implementasi antarmuka pengguna sistem identifikasi emosi pada suara. Dengan menggunakan fungsi-fungsi yang ada, pengguna dapat mengunggah *file* suara, melihat visualisasi preprocessing audio, melihat hasil ekstraksi fitur audio, dan mendapatkan prediksi emosi dari *file* suara yang diunggah.



Gambar 2.4 - Antarmuka hasil *file* tab_one.py

2.3.6 File tab_two.py

File 'tab_two.py' adalah sebuah file yang berisi implementasi dari tahapan preprocessing dan penggunaan *Support Vector Machine* (SVM) pada sistem identifikasi sentimen positif dan negatif pada audio. Pada file ini, digunakan library Streamlit untuk membuat antarmuka pengguna yang interaktif.

SOURCE CODE : tab_two.py

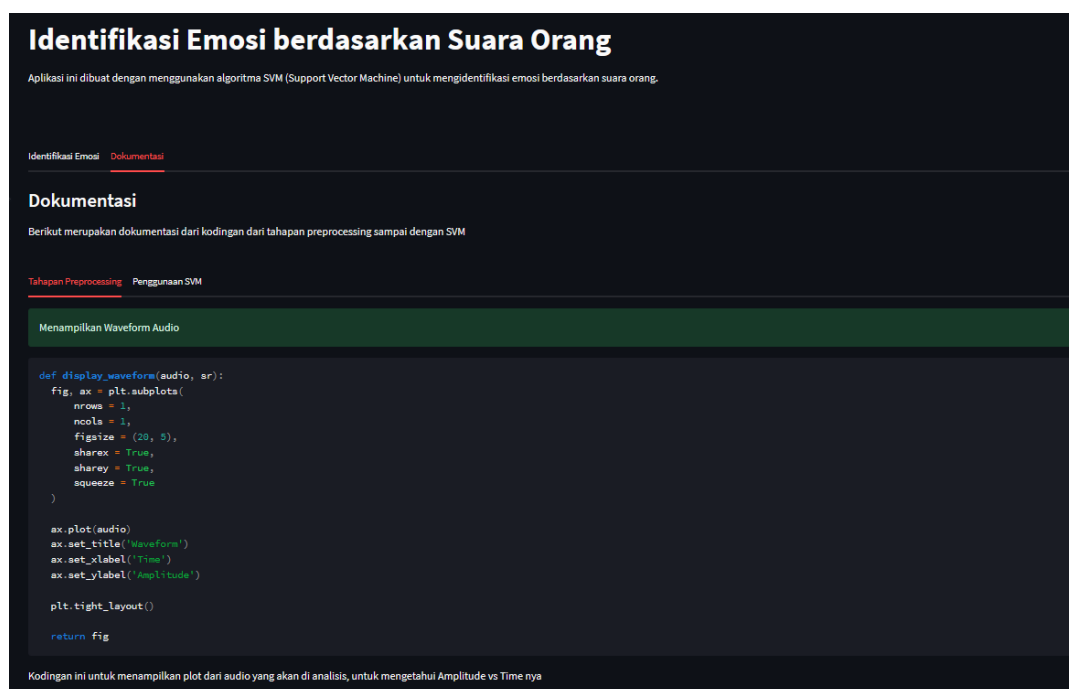
No	Source code	Penjelasan
1	import streamlit as st	Pada baris 3 terdapat fungsi 'preprocessing()' yang bertujuan untuk melakukan tahapan preprocessing pada data audio sebelum proses penggunaan SVM. Tahapan ini melibatkan beberapa proses dan tampilan plot audio yang berguna untuk pemahaman dan analisis data.
2		
3	def preprocessing():	
4	st.success("Menampilkan Waveform Audio")	
5	waveform = '''	
6	def display_waveform(audio, sr):	
7	fig, ax = plt.subplots(
8	nrows = 1,	
9	ncols = 1,	
10	figsize = (20, 5),	
11	sharex = True,	
12	sharey = True,	
13	squeeze = True	
14)	
15		
16	ax.plot(audio)	Pertama, terdapat bagian untuk menampilkan waveform audio dengan fungsi 'display_waveform(audio, sr)'. Fungsi ini menerima audio dan sampling rate (sr) sebagai parameter, dan menghasilkan plot Amplitudo vs. Waktu dari audio tersebut. Tujuan dari plot ini adalah untuk memvisualisasikan bentuk gelombang audio dan melihat variasi amplitudo pada rentang waktu tertentu.
17	ax.set_title('Waveform')	
18	ax.set_xlabel('Time')	
19	ax.set_ylabel('Amplitude')	
20		
21	plt.tight_layout()	
22		
23	... return fig	
24		
25		
26	st.code(waveform, language="python")	
27	st.write("Kodingan ini untuk menampilkan plot dari	
28	audio yang akan di analisis, untuk mengetahui Amplitude	
29	vs Time nya")	
30	st.markdown("---")	
31		
32	st.success("Process RMSe")	Selanjutnya, terdapat bagian untuk menampilkan plot Root Mean Square Energy (RMSe) audio dengan fungsi 'display_rms(audio, frame, hop)'. Fungsi ini menerima audio, frame, dan hop sebagai parameter. RMSe adalah metode untuk mengukur energi rata-rata dalam frame waktu tertentu.
33	disp_rms = '''	
34	def display_rms(audio, frame=2048, hop=512):	
35	rms = librosa.feature.rms(y=audio,	
36	frame_length=frame, hop_length=hop)[0]	
37		
38	fig, ax = plt.subplots(
39	nrows = 1,	
40	ncols = 1,	
41	figsize = (20, 5),	
42	sharex = True,	
43	sharey = True,	
44	squeeze = True	
45)	
46		

47	<code>ax.plot(rms)</code>	Plot RMSe menunjukkan
48	<code>ax.set_title('RMS')</code>	perubahan energi audio
49	<code>ax.set_xlabel('Time')</code>	seiring waktu, yang
50	<code>ax.set_ylabel('Amplitude')</code>	dapat membantu dalam
51		analisis sentimen audio.
52	<code>plt.tight_layout()</code>	
53		Kemudian, terdapat
54	<code>return fig</code>	bagian untuk menampilkan
55	<code>...</code>	plot Zero Crossing Rate
56	<code>st.code disp_rms, language='python')</code>	(ZCR) audio dengan
57	<code>st.write("Kodingan ini untuk mencari RMSe audio dan</code>	fungsi
58	<code>menampilkan plot audio sesudah dilakukan proses RMSe")</code>	<code>'display_zcr(audio,</code>
59	<code>st.markdown("---")</code>	<code>frame, hop)'</code> . Fungsi ini
60		menerima audio, frame,
61	<code>st.success("Process Zero Crossing Rate")</code>	dan hop sebagai
62	<code>disp_zcr = '''</code>	parameter. ZCR adalah
63	<code>def display_zcr(audio, frame=2048, hop=512):</code>	metode untuk menghitung
64	<code>zcr = librosa.feature.zero_crossing_rate(y=audio,</code>	jumlah perubahan tanda
65	<code>frame_length=frame, hop_length=hop)[0]</code>	(crossing zero) dalam
66		frame waktu tertentu.
67	<code>fig, ax = plt.subplots(</code>	Plot ZCR menunjukkan
68	<code>nrows = 1,</code>	perubahan frekuensi dan
69	<code>ncols = 1,</code>	sifat gelombang audio,
70	<code>figsize = (20, 5),</code>	yang dapat memberikan
71	<code>sharex = True,</code>	informasi tambahan
72	<code>sharey = True,</code>	tentang sentimen audio.
73	<code>squeeze = True</code>	
74	<code>)</code>	
75		Terakhir, terdapat
76	<code>ax.plot(zcr)</code>	bagian untuk menampilkan
77	<code>ax.set_title('ZCR')</code>	plot Mel-frequency
78	<code>ax.set_xlabel('Time')</code>	Cepstral Coefficients
79	<code>ax.set_ylabel('Amplitude')</code>	(MFCC) dengan fungsi
80		<code>'display_mfcc(audio, sr,</code>
81	<code>plt.tight_layout()</code>	<code>frame, hop, mfcc_num)'</code> .
82		Fungsi ini menerima
83	<code>return fig</code>	audio, sampling rate
84	<code>...</code>	(sr), frame, hop, dan
85	<code>st.code disp_zcr, language='python')</code>	jumlah MFCC (mfcc_num)
86	<code>st.write("Kodingan ini untuk mencari ZCR audio dan</code>	sebagai parameter. MFCC
87	<code>menampilkan plot audio sesudah dilakukan proses ZCR")</code>	adalah representasi
88	<code>st.markdown("---")</code>	numerik yang
89		menggambarkan fitur
90	<code>st.success("Menampilkan Hasil MFCC yang sudah</code>	frekuensi dan bentuk
91	<code>dilakukan")</code>	gelombang audio. Plot
92	<code>disp_mfcc = '''</code>	MFCC, delta MFCC
93	<code>def display_mfcc(audio, sr, frame=2048, hop=512,</code>	pertama, dan delta MFCC
94	<code>mfcc_num=25):</code>	kedua menunjukkan
95	<code>mfcc_spectrum = librosa.feature.mfcc(y=audio,</code>	perubahan dalam spektrum
96	<code>sr=sr, n_fft=frame, hop_length=hop, n_mfcc=mfcc_num)</code>	frekuensi audio seiring
97		waktu, yang sangat
98	<code>delt1 = librosa.feature.delta(mfcc_spectrum,</code>	relevan dalam analisis
99	<code>order=1)</code>	sentimen audio.
100	<code>delt2 = librosa.feature.delta(mfcc_spectrum,</code>	
101	<code>order=2)</code>	Seluruh fungsi-fungsi
102		tersebut digunakan untuk
103	<code>mfcc_feature =</code>	menampilkan plot audio
104	<code>np.concatenate((np.mean(mfcc_spectrum, axis=1),</code>	dan menggambarkan
105	<code>np.mean(delt1, axis=1), np.mean(delt2, axis=1)))</code>	karakteristik audio yang
106		relevan dengan sentimen.
107	<code>fig, ax = plt.subplots(</code>	Hal ini membantu
108	<code>nrows = 3,</code>	pemahaman dan analisis
109	<code>ncols = 1,</code>	lebih lanjut terhadap
110	<code>figsize = (10, 15),</code>	data audio sebelum
111	<code>sharex = True,</code>	dilakukan proses
112	<code>sharey = True,</code>	prediksi sentimen
113	<code>squeeze = True</code>	menggunakan SVM.
114	<code>)</code>	

115		Pada baris 140 terdapat fungsi 'svm()' yang berisi tahapan penggunaan metode SVM pada sistem identifikasi sentimen audio. Fungsi ini melibatkan beberapa proses penting.
116	ax[0].plot(delt1)	
117	ax[0].set_title('MFCC')	
118	ax[0].set_xlabel('Time')	
119	ax[0].set_ylabel('Amplitude')	
120		
121	ax[1].plot(delt2)	
122	ax[1].set_title('MFCC')	
123	ax[1].set_xlabel('Time')	
124	ax[1].set_ylabel('Amplitude')	
125		
126	ax[2].plot(mfcc_spectrum)	
127	ax[2].set_title('MFCC')	
128	ax[2].set_xlabel('Time')	
129	ax[2].set_ylabel('Amplitude')	
130		
131	plt.tight_layout()	
132		
133	return fig	
134	...	
135	st.code(dispmfcc, language='python')	
136	st.write("Kodingan ini untuk menampilkan plot sesudah dilakukan proses MFCC")	
137		
138		
139	def svm():	Pertama, terdapat fungsi 'extract_feature(audio, sr, frame, hop, mfcc_num)' yang digunakan untuk mengekstraksi fitur-fitur audio yang relevan. Fitur-fitur ini mencakup MFCC, ZCR, dan RMSE. Melalui proses ekstraksi fitur ini, informasi penting dalam audio diekstraksi dan siap untuk digunakan dalam proses prediksi.
140	st.success("Ekstraksi Fitur Audio")	
141	extract = ''	Selanjutnya, terdapat fungsi 'normalize_feature(feature)' yang digunakan untuk melakukan normalisasi pada fitur-fitur yang telah diekstraksi sebelumnya. Normalisasi ini bertujuan untuk memastikan bahwa fitur-fitur memiliki skala yang seragam dan tidak dominan satu sama lain. Dalam implementasi ini, digunakan scaler yang telah di-load dari file pickle untuk melakukan normalisasi.
142	def extract_feature(audio, sr, frame=2048, hop=512, mfcc_num=25):	
143	audio_mfcc, audio_zcr, audio_rmse = [],[],[]	
144		
145		
146	mfcc_score = mfcc(audio, sr, frame=frame, hop=hop, mfcc_num=mfcc_num)	
147	zcr_score = np.mean(zcr(audio, frame=frame, hop=hop))	
148	rmse_score = np.mean(rms(audio, frame=frame, hop=hop))	
149		
150		
151		
152		
153	audio_mfcc.append(mfcc_score)	
154	audio_zcr.append(zcr_score)	
155	audio_rmse.append(rmse_score)	
156		
157	feature = np.column_stack((audio_mfcc, audio_zcr, audio_rmse))	
158		
159	return feature	
160	...	
161	st.code(extract, language="python")	Kemudian, terdapat fungsi 'prediction_data(model, df)' yang digunakan untuk melakukan prediksi label berdasarkan model SVM yang telah dilatih sebelumnya. Fungsi ini mengembalikan hasil prediksi dalam bentuk teks yang lebih bermakna, yaitu "Bahagia" atau "Sedih". Untuk melakukannya, fungsi ini menggunakan dictionary 'labels' yang menghubungkan nilai prediksi (0 atau 1) dengan label yang sesuai.
162	st.write("Fungsi ini untuk melakukan ekstraksi fitur dari audio yang akan di prediksi, fitur yang sudah di extract ini nantinya akan diambil mfcc nya, mfcc yang digunakan yaitu sebanyak 25")	
163	st.write("Hasil yang akan didapatkan yaitu feature mfcc, feature zcr, dan juga feature rmse. Yang diana masing-masing sebanyak 25 matriks, dan akan di gabungkan menjadi 75 matriks feature")	
164	st.markdown("---")	
165		
166		
167	st.success("Normalisasi Feature")	
168	normalize = ''	
169	def normalize_feature(feature):	
170	scaler =	
171	pickle.load(open("scaler_svm_2000data_new_1.pkl", "rb"))	
172	feature = scaler.transform(feature)	
173		
174	return feature	
175	...	
176	st.code(normalize, language="python")	Terakhir, terdapat fungsi
177		
178		
179		
180		
181		
182		

183	st.write("Fungsi ini untuk melakukan normalisasi fitur	'prediction(audio, sr)'
184	yang sudah di extract. Bertujuan agar prediksi data	yang merupakan fungsi
185	akurat, dan fitur tidak bernilai besar, yang dapat	utama untuk melakukan
186	mempengaruhi hasil prediksi")	prediksi sentimen pada
187	st.write("Normalisasi menggunakan scaler yang sudah	audio yang diberikan.
188	dibuat pada saat proses training data model")	Fungsi ini memuat model
189	st.markdown('---')	SVM yang telah dilatih
190		sebelumnya,
191	st.success("Prediction Label Decode")	mengeksktraksi fitur-
192	label = ''	fitur audio menggunakan
193	def prediction_data(model, df):	fungsi
194	y_pred = model.predict(df)	'extract_feature()',
195		melakukan normalisasi
196	labels = {	fitur menggunakan fungsi
197	0: "Bahagia",	'normalize_feature()',
198	1: "Sedih"	dan mengembalikan hasil
199	}	prediksi menggunakan
200	pred = labels[y_pred[0]]	fungsi
201		'prediction_data()'. Model SVM dan scaler yang
202	return pred	digunakan dalam proses
203	'''	prediksi di-load dari
204	st.code(label, language="python")	file pickle yang telah
205	st.write("Fungsi ini untuk melakukan decode label,	di-export sebelumnya.
206	yang dimana hasil prediksi yang bernilai 0 atau 1 akan	
207	dilakukan decode untuk menampilkan hasil emosi, apakah	Fungsi-fungsi dalam
208	sedih atau bahagia. sesuai dengan prediksi dari model	'svm()' bekerja secara
209	yang sudah dibuat")	bersama-sama untuk
210	st.markdown('---')	mengimplementasikan
211		metode SVM dalam sistem
212	st.success("Fungsi Prediksi")	identifikasi sentimen
213	prediction = ''	audio. Tahapan ini
214	def prediction(audio, sr):	melibatkan ekstraksi
215	model =	fitur audio, normalisasi
216	pickle.load(open("svm_2000data_new_1_fix.pkl", "rb"))	fitur, prediksi label,
217		dan penggunaan model SVM
218	df = extract_feature(audio, sr)	yang telah dilatih
219	x = normalize_feature(df)	sebelumnya.
220	pred = prediction_data(model, x)	
221		
222	return pred	
223	'''	
224	st.code(prediction, language='python')	
225	st.write("Fungsi ini untuk melakukan prediksi dari	
226	audio yang akan dilakukan prediksi. Dengan memanggil	
227	fungsi-fungsi yang sudah dibuat tadinya. yaitu ekstraksi	
228	fitur audio, dan dilakukan normalisasi lalu dilakukan	
229	label decode dan hasil prediksi akan ditampillkan.")	
230	st.write("Menggunakan model yang sudah di training	
231	sebelumnya. Yang sudah diexport dan dipanggil menggunakan	
232	library pickle")	
233		
234	def content_tab_two(title):	
235	st.subheader(title)	
236	st.write("Berikut merupakan dokumentasi dari kodingan	
237	dari tahapan preprocessing sampai dengan SVM")	
238	st.markdown(" ", unsafe_allow_html=True)	
239		
240	tab_preprocessing, tab_svm = st.tabs(["Tahapan	
241	Preprocessing", "Penggunaan SVM"])	
242		
243	with tab_preprocessing:	
244	preprocessing()	
245		
246	with tab_svm:	
247	svm()	
248		

Seluruh fungsi dan penjelasan tersebut digabungkan dalam fungsi 'content_tab_two()' yang mengatur tampilan tab pada aplikasi Streamlit. Fungsi ini menerima parameter 'title' untuk menampilkan judul, dan kemudian menampilkan dokumentasi dari tahapan preprocessing dan penggunaan SVM. Dalam tahapan preprocessing, ditampilkan hasil waveform audio, RMSe, ZCR, dan MFCC. Sedangkan dalam penggunaan SVM, ditampilkan fungsi-fungsi terkait seperti ekstraksi fitur audio, normalisasi fitur, decode label, dan fungsi prediksi.



Gambar 2.5 - Antarmuka hasil *file* tab_two.py

Dengan demikian, implementasi kodingan ini memungkinkan untuk menampilkan sebuah *Website* yang memiliki input sebuah audio yang dapat menganalisis sentimen sedih (sad) dan senang (happy) dari audio menggunakan fitur-fitur yang diekstraksi dan model klasifikasi yang telah dilatih.

BAB 3 PENUTUP

3.2 Kesimpulan

Hasil pengujian model SVM yang dibangun melalui data audio berlabel *sad* dan *happy* diperoleh nilai akurasi tertinggi sebesar 0.86 (86%). Nilai tersebut diperoleh dari model SVM kernel RBF dengan nilai $C = 10$, $\gamma = 0.1$, dan $dfs = ovr$. Dengan diperolehnya nilai akurasi tersebut dapat disimpulkan bahwa SVM (*Support Vector Machine*) memiliki tingkat akurasi yang baik dalam mengklasifikasikan dua kelas data. Namun dengan catatan bahwa model SVM sudah ditentukan parameternya seperti nilai C dan γ .

Sistem yang dibuat dapat diakses pada link : <https://audio-identification-svm.streamlit.app/>. Sistem yang dibuat adalah sebuah sistem berbasis *Website* yang menggunakan Streamlit sebagai *framework* untuk mengimplementasikan analisis sentimen pada data audio. Pengguna dapat mengunggah *file* audio yang ingin dianalisis melalui antarmuka *Website* yang disediakan. Kemudian, fitur-fitur tersebut akan digunakan sebagai input untuk model klasifikasi SVM (*Support Vector Machine*) yang telah dilatih sebelumnya. Model akan memberikan prediksi sentimen audio apakah sedih (*sad*) atau senang (*happy*). Hasil prediksi sentimen akan ditampilkan kepada pengguna melalui antarmuka *Website*.

3.3 Saran

Berdasarkan kesimpulan dan hasil implementasi SVM (*Support Vector Machine*), terdapat beberapa hal yang dapat dijadikan acuan untuk evaluasi dan mengembangkan hasil akhir baik dari laporan ini maupun sebagai pedoman untuk implementasi *machine learning* kedepannya. Beberapa saran yang dapat dilakukan pada implementasi kedepannya antara lain :

1. Menambah variasi data audio yang diuji coba baik dari jumlah *file* maupun jumlah kelas data yang diuji (dalam kasus berupa *mood sad* dan *happy*). Hal ini berguna untuk menguji performa model SVM yang dibangun dalam melakukan klasifikasi data dengan dimensi yang tinggi.

2. Menguji coba nilai parameter lain pada SVM tiap *kernel* untuk menguji performa model SVM yang dibangun. Hal ini dapat menjadi acuan bagi implementasi SVM berikutnya terkait nilai parameter berapa yang optimal untuk digunakan.

DAFTAR PUSTAKA

- [1] T. Joachim, *Text categorization with Support Vector Machines: Learning with many relevant features*. 1998.
- [2] C.-W. Hsu, C.-C. Chang, dan C.-J. Lin, "A Practical Guide to Support Vector Classification," 2003. [Daring]. Tersedia pada: <http://www.csie.ntu.edu.tw/~cjlin>
- [3] J. Huang, A. Ariyaeinia, dan B. Krose, *Speech emotion recognition using hidden Markov models*. International Speech Communication Association, 2001. doi: 10.21437/eurospeech.2001-627.
- [4] M. M. Goodwin dan J. C. Bierman, "Digital Audio Fundamentals," in *Introduction to Digital Audio Coding and Standards*. 2003.
- [5] S. S. Kumar, *Data Preprocessing Techniques for Data Mining*. 2014.
- [6] T. Hastie, R. Tibshirani, dan J. Friedman, "The Elements of Statistical Learning: Data Mining, Inference, and Prediction," 2009.
- [7] H. H. Gatlin dan R. B. Rainsberger, *Software Testing in Handbook of Human-Computer Interaction*. 2013.
- [8] L. R. Rabiner dan B. H. Juang, "Fundamentals of Speech Recognition," 1993.