

**FINAL PROJECT**  
**PENGANTAR PEMROSESAN DATA MULTIMEDIA**



**Disusun Oleh:**  
**KELOMPOK 1A**

<b>Ni Kadek Trisnawati</b>	<b>2108561026</b>
<b>I Komang Sutrisna Eka Wijaya</b>	<b>2108561032</b>
<b>I Putu Gede Mahardika Adi Putra</b>	<b>2108561055</b>

**PROGRAM STUDI INFORMATIKA**  
**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**  
**UNIVERSITAS UDAYANA**  
**JIMBARAN**  
**2023**

## DAFTAR ISI

DAFTAR ISI.....	ii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Problem Komputasi .....	2
1.3 Tujuan.....	2
1.4 Manfaat .....	2
BAB II ISI.....	3
2.1 Manual Aplikasi.....	3
2.1.1 Fitur Sistem Aplikasi .....	3
1. Halaman Awal .....	3
2. <i>Machine Modeling</i> .....	4
3. <i>Checking</i> .....	6
4. Fitur di luar Program .....	8
5. Kesimpulan .....	9
2.1.2 Antarmuka.....	10
1. Graphical User Interface (GUI) .....	10
2. Streamlit.....	10
2.1.3 Tuning Hyperparameter .....	11
2.2 Source Code.....	13
2.2.1 Design Pattern .....	13
2.2.2 Entry Point (Main Module).....	17
2.2.3 Model .....	18
2.2.4 View .....	19
2.2.5 Controller .....	26
2.2.6 Service.....	29
BAB III PENUTUP .....	32
DAFTAR PUSTAKA .....	33

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Emosi merupakan suatu aspek penting dalam kehidupan manusia berupa perasaan dan memiliki pengaruh pada perilaku manusia. Emosi dapat diekspresikan dengan berbagai cara, salah satunya adalah melalui suara. Dalam dunia teknologi suara juga digunakan untuk melakukan interaksi antara manusia dengan komputer. Dalam interaksi tersebut diperlukan kemampuan untuk melakukan pengenalan, penafsiran dan memberikan respons emosi yang diekspresikan dalam ucapan [1]. Terdapat banyak metode yang bisa digunakan untuk mengidentifikasi emosi dalam suara salah satunya yaitu melalui metode jaringan saraf tiruan.

*Convolutional Neural Network* (CNN) merupakan bagian dari jaringan saraf tiruan pada *machine learning* yang berfungsi untuk pengolahan data. CNN adalah jaringan saraf yang menggunakan konvolusi minimal di setiap lapisannya [2]. *Convolutional Neural Networks* (CNN) adalah kasus khusus Jaringan Syaraf Tiruan (JST) dan saat ini dianggap sebagai model terbaik untuk memecahkan masalah pengenalan dan deteksi objek [3]. Pada program ini CNN digunakan untuk mengidentifikasi emosi pada fitur data suara yang telah di ekstraksi. Proses ekstraksi fitur menggunakan *Mel-Frequency Cepstral Coefficient* (MFCC).

MFCC merupakan metode ekstraksi ciri untuk mendapatkan *cepstral coefficient* dan frame sehingga dapat digunakan untuk pemrosesan pengenalan suara agar lebih baik dalam ketepatan. Berdasarkan yang sudah dijelaskan maka program yang akan dirancang yaitu menggunakan *Convolutional Neural Networks* (CNN) untuk melakukan klasifikasi dan *Mel-Frequency Cepstral Coefficient* (MFCC) untuk ekstraksi fitur dengan program aplikasi berbentuk web.

## 1.2 Problem Komputasi

Membangun sistem aplikasi untuk mengidentifikasi sentimen/emosi dari beberapa suara orang

1. Terdapat dua sentimen yaitu sentimen positif (atau emosi *happy*) dan sentimen negatif (atau emosi *sad*).
2. Dataset merupakan data audio (suara orang)
3. Pembagian data menjadi 2 bagian: 80% untuk dataset training dan 20% untuk dataset testing
4. Terdapat 3 tahapan utama proses komputasi untuk menghasilkan model klasifikasi yaitu: 1) *Preprocessing*; 2) *Training*; dan 3) *Testing*.
5. Tahapan *preprocessing* menggunakan MFCC

## 1.3 Tujuan

Adapun tujuan dari laporan ini, yaitu:

1. Memenuhi tagihan tugas mata kuliah Pengantar Pemrosesan Data Multimedia.
2. Membangun sistem aplikasi berbasis web untuk mengidentifikasi sentimen/emosi positif (senang) atau negatif (sedih) dari audio suara.
3. Mengimplementasikan algoritma jaringan saraf tiruan *Convolutional Neural Network* (CNN) dan *Mel-Frequency Cepstral Coefficient* (MFCC) untuk ekstraksi fitur.

## 1.4 Manfaat

Adapun manfaat yang diperoleh dari laporan ini, yaitu:

1. Dapat menambah wawasan mengenai cara untuk membangun aplikasi identifikasi sentimen/emosi positif (senang) atau negatif (sedih) dari audio
2. Dapat mengimplementasikan algoritma jaringan saraf tiruan *confusional neural network* (CNN) dan *Mel-Frequency Cepstral Coefficient* (MFCC) untuk ekstraksi fitur
3. Dapat membangun sistem menggunakan *design pattern* dasar *Model-View-Controller* (MVC)

## BAB II

### ISI

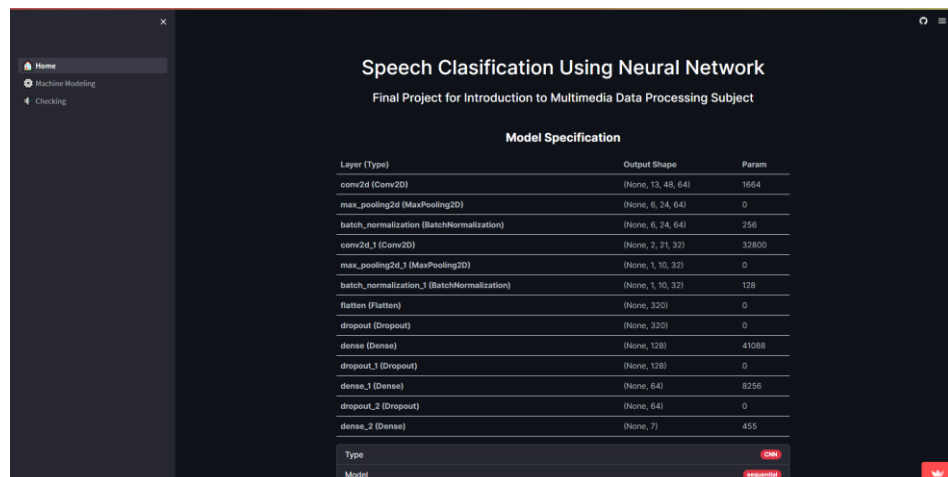
## 2.1 Manual Aplikasi

### 2.1.1 Fitur Sistem Aplikasi

Dalam konteks web atau aplikasi, fitur merujuk pada fitur atau kemampuan khusus yang diberikan oleh sistem. Fitur adalah komponen yang memberi pengguna nilai tambahan dan memungkinkan mereka melakukan tindakan atau memperoleh manfaat tertentu. Berikut adalah Fitur dari Program “*Speech Classification Using Neural Network*”, untuk website bisa di akses pada link berikut <https://speech-emotion-classification.streamlit.app/>

#### 1. Halaman Awal

Pada Bagian awal di program ini berisikan home page, yang di mana memuat informasi seperti *Model Spesification*, *How to Use*, dan *Contributors dari program*.

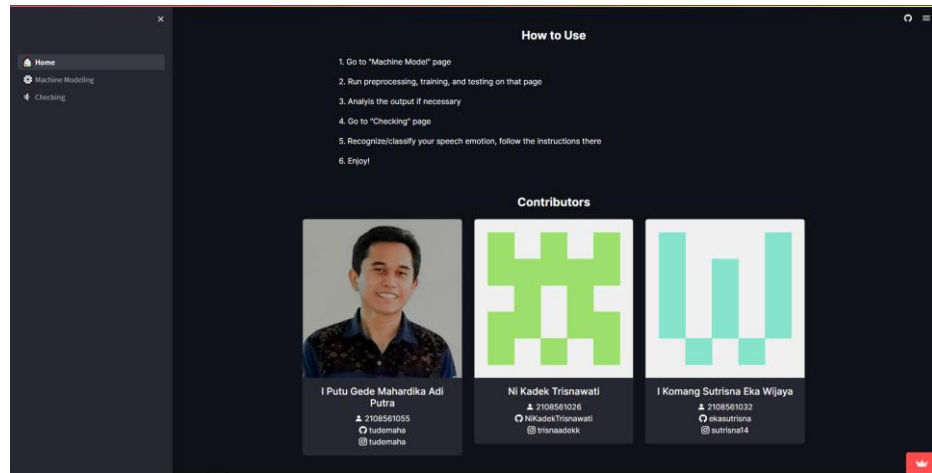


Layer (Type)	Output Shape	Param
conv2d (Conv2D)	(None, 13, 48, 64)	1664
max_pooling2d (MaxPooling2D)	(None, 6, 24, 64)	0
batch_normalization (BatchNormalization)	(None, 6, 24, 64)	256
conv2d_1 (Conv2D)	(None, 2, 21, 32)	32800
max_pooling2d_1 (MaxPooling2D)	(None, 1, 10, 32)	0
batch_normalization_1 (BatchNormalization)	(None, 1, 10, 32)	128
flatten (Flatten)	(None, 320)	0
dropout (Dropout)	(None, 320)	0
dense (Dense)	(None, 128)	41088
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8256
dropout_2 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 7)	455

Type

Model

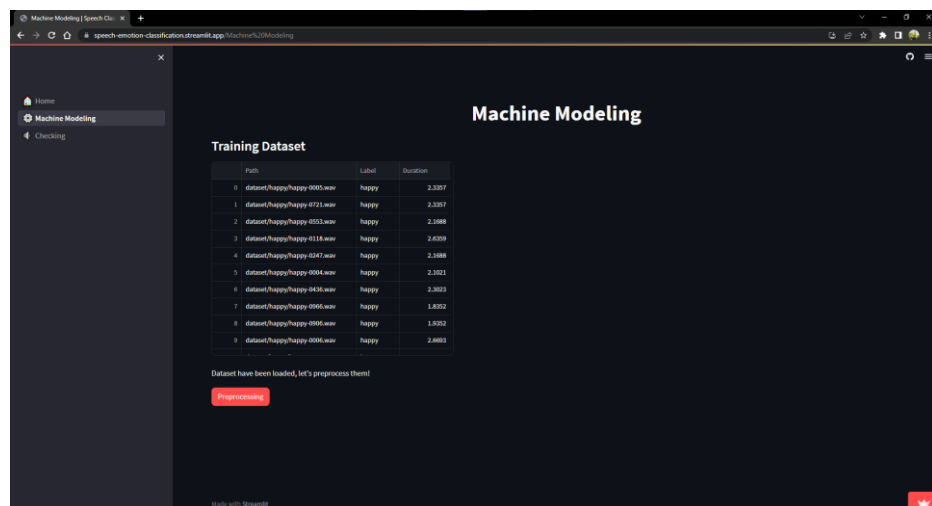
Gambar 2. Halaman Awal Model Spesifications



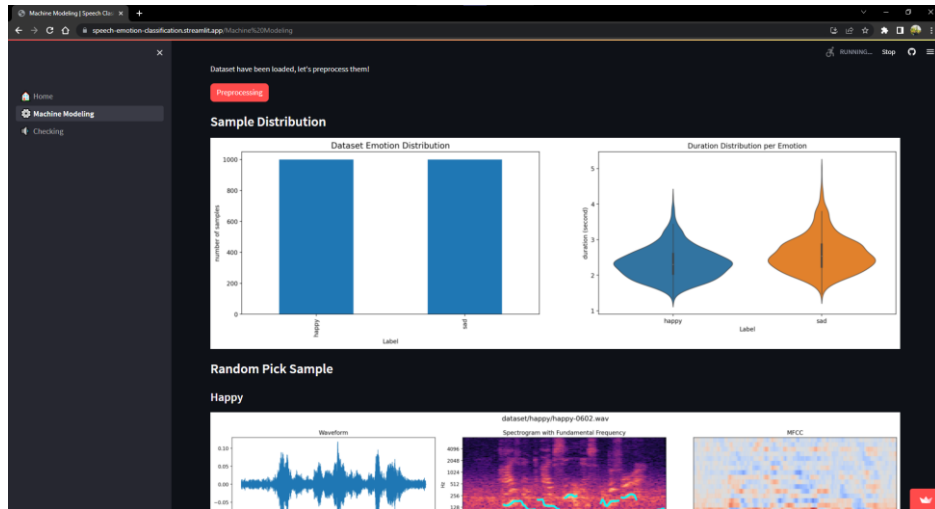
Gambar 3. Halaman Awal How to Use dan Contributors

## 2. Machine Modeling

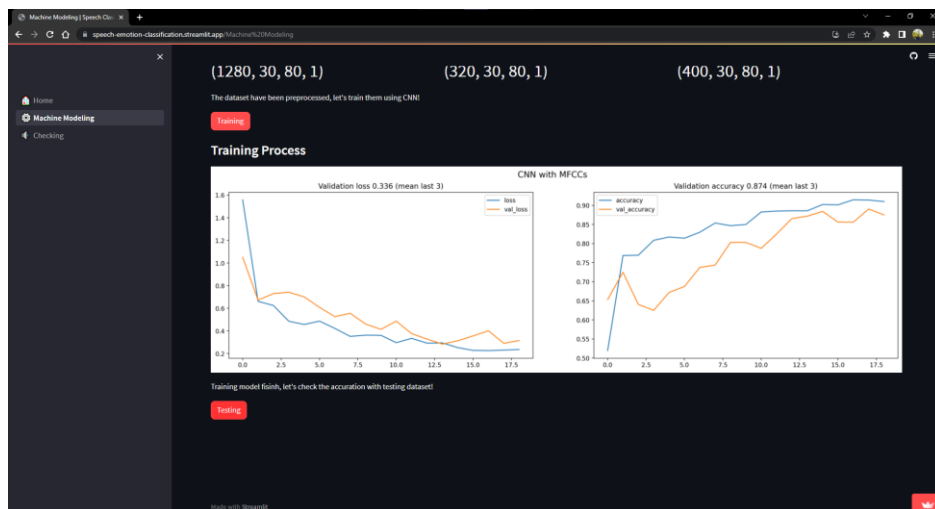
Kemampuan program untuk melatih dan menggunakan model jaringan saraf tiruan (*neural network*) untuk klasifikasi suara, seperti suara sedih atau bahagia, dimungkinkan oleh fitur model mesin.



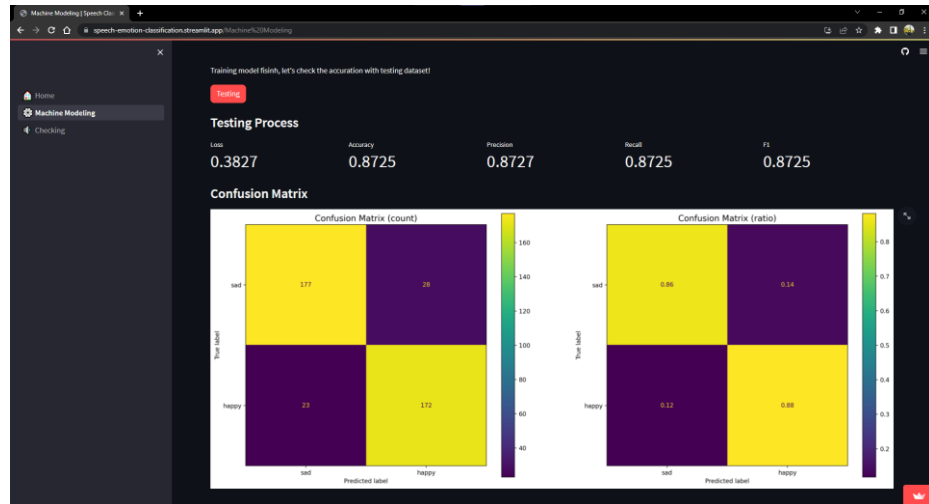
Gambar 4. Machine Modeling Training Dataset



*Gambar 5. Machine Modeling Sample Distributions Random pick sample Happy and Sad, Average MFCCs Column, MFCC Comparison, MFCCs Distribution*



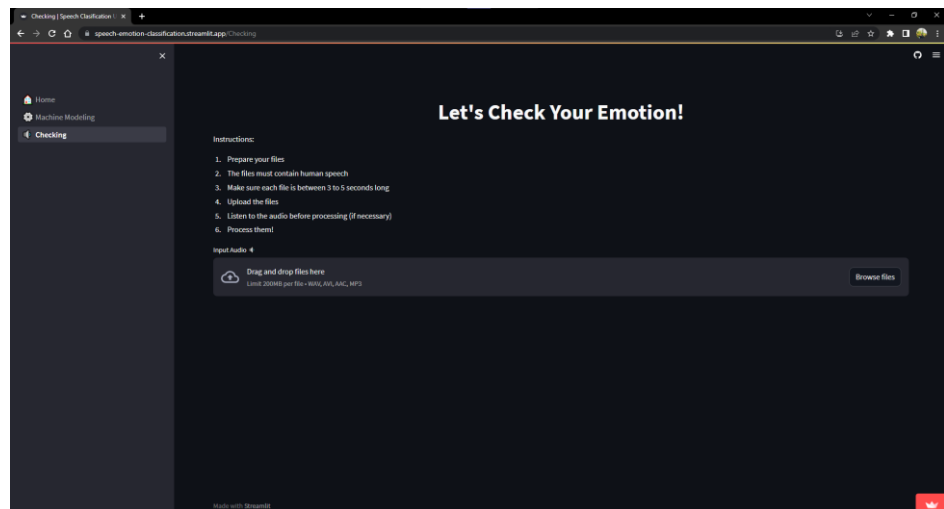
*Gambar 6. Training Process*



Gambar 7. Testing Process Confusion Matrix

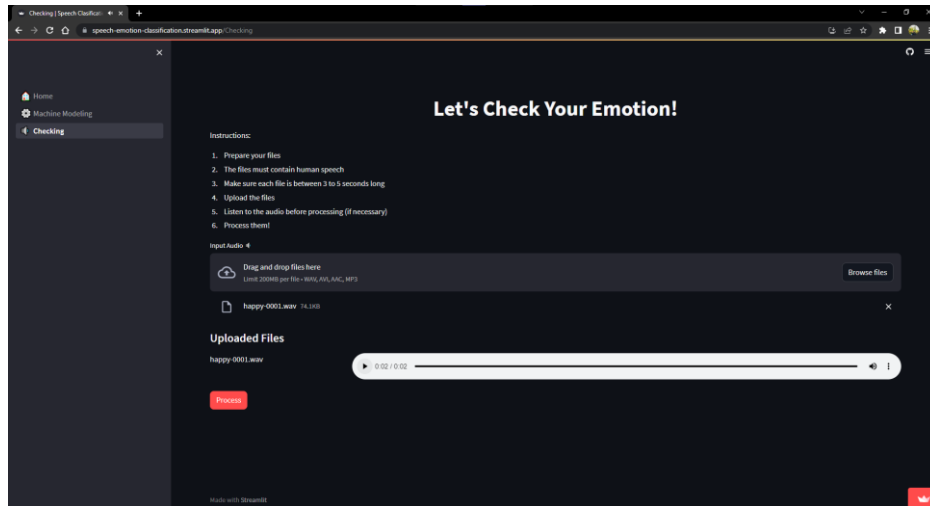
### 3. Checking

Kemampuan program untuk melatih dan menggunakan model jaringan saraf tiruan (*neural network*) untuk klasifikasi suara, seperti suara sedih atau bahagia, dimungkinkan oleh fitur model mesin. Berikut adalah tampilan *checking* pada program

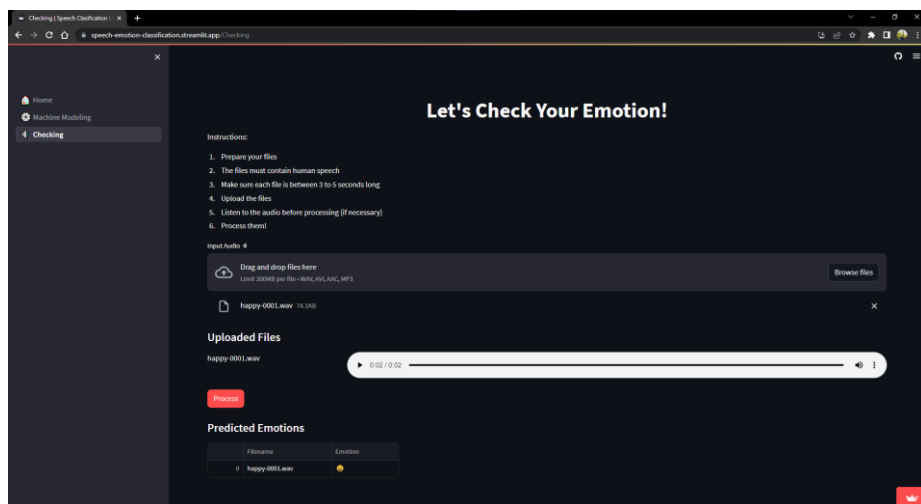


Gambar 8. Checking instruction and drop files data

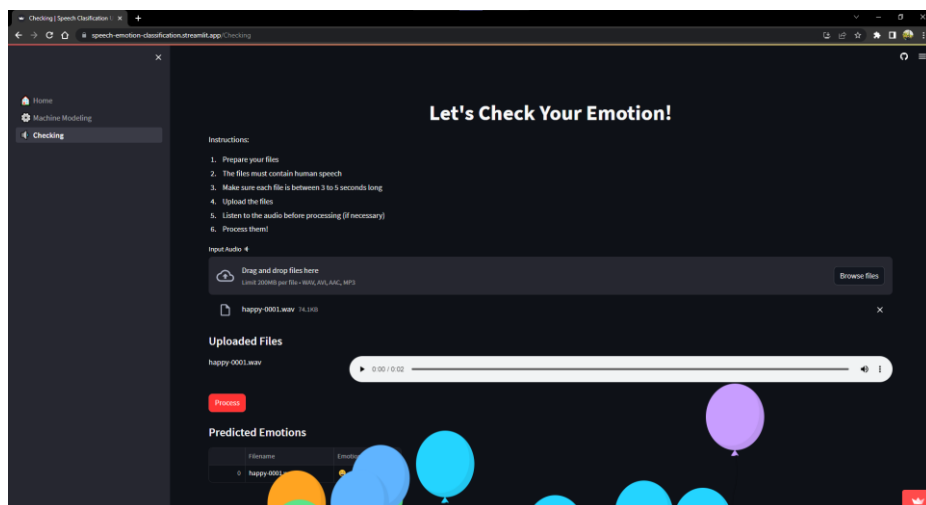




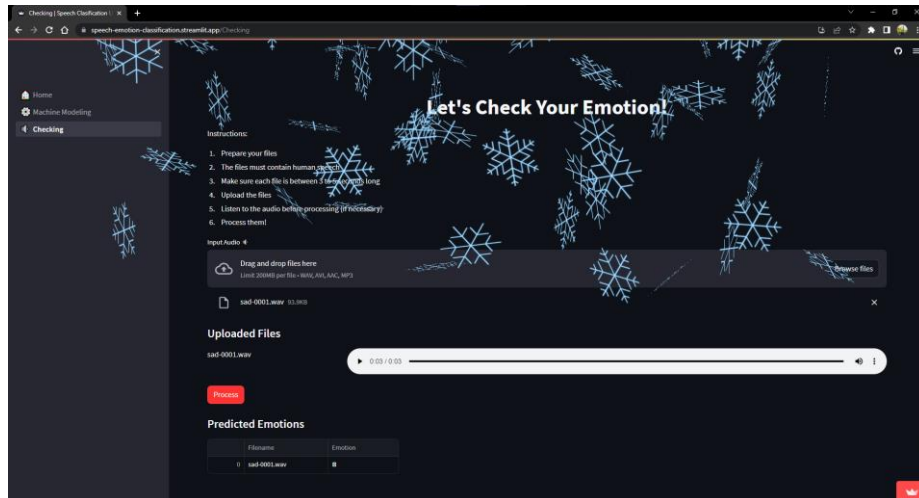
*Gambar 9. Contoh user input data happy-001.wav*



*Gambar 10. Checking Data Emotions*



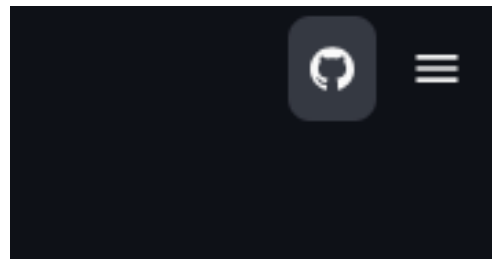
*Gambar 11. Checking Data Emotions. Happy emotion*



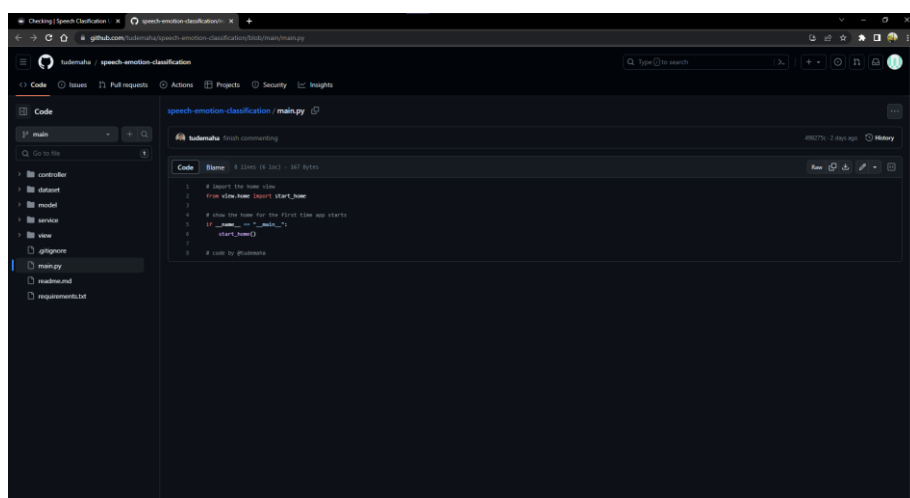
Gambar 12. Checking Data Emotionns, sad emotion

#### 4. Fitur di luar Program

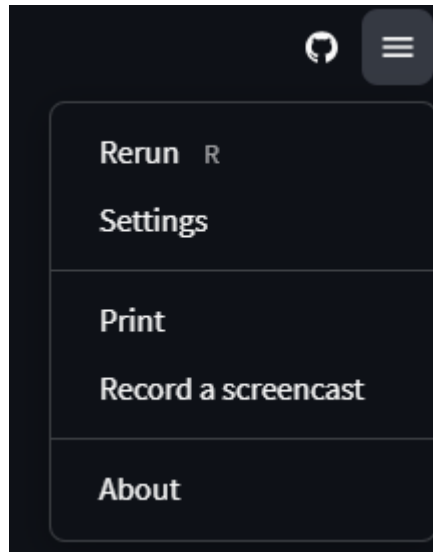
Fitur Luar Program yang di masukkan pada program ini adalah fitur *code github* atau kode program dan garis tiga yang berada pada pojok kana atas website, fitur ini merupakan fitur bawaan dari framework streamlit.



Gambar 13. Link to code github and streamlit default feature



Gambar 14. Github code for the website



Gambar 15. Streamlit default feature

## 5. Kesimpulan

Beberapa fitur program "*Speech Classification Using Neural Network*", dapat dijelaskan sebagai berikut:

- Fitur Sistem Aplikasi:
  - a. Halaman Awal: Menyajikan informasi tentang Model Spesifikasi, Cara Penggunaan, dan Kontributor program.
- Fitur *Machine Modeling*:
  - a. Melatih dan menggunakan model jaringan saraf tiruan, juga dikenal sebagai *neural network*, untuk klasifikasi suara sedih atau bahagia.
  - b. Menampilkan informasi tentang dataset pelatihan, serta proses pelatihan dan pengujian model..
  - c. Menampilkan distribusi sampel acak (senang dan sedih), kolom rata-rata MFCC, perbandingan MFCC, dan distribusi MFCC..
- Fitur *Checking*:
  - a. Memvalidasi dan memeriksa input suara sebelum memulai proses klasifikasi
  - b. Memeriksa format, durasi, dan kualitas suara yang diunggah
  - c. Memberikan instruksi dan fitur unggah file suara untuk pengecekan
  - d. Menampilkan hasil pemeriksaan emosi suara (senang atau sedih) berdasarkan model klasifikasi.

- Fitur di luar Program:
  - a. Pojok kanan atas halaman web memiliki tautan ke kode program di GitHub dan fitur *Streamlit* default..

Program ini memiliki pengalaman pengguna yang lebih interaktif dan informatif dengan fitur-fitur ini, yang memungkinkan pengguna menjelajahi program, mempelajari informasi tentang model, melatih dan menguji model, dan memeriksa validitas suara sebelum diklasifikasikan.

### **2.1.2 Antarmuka**

Antarmuka merupakan kegiatan yang sering dilakukan manusia pada saat menggunakan alat untuk meringankan pekerjaannya, interface dapat menerima komunikasi atau informasi dari user dan sebaliknya dapat memberikan komunikasi atau informasi kembali kepada user yang disebut dengan user interface. Sederhananya, antarmuka adalah desain antarmuka yang dapat dilihat secara langsung atau melalui perangkat tertentu

#### **1. Graphical User Interface (GUI)**

Graphical User Interface (GUI) adalah komponen sistem visual interaktif untuk aplikasi komputer pribadi. GUI menampilkan objek yang mampu menimbulkan masalah dan mewakili tindakan asal pengguna. Dengan menggunakan GUI, dapat mengetahui bahwa apa yang dimasukkan telah diterima dan responsnya ditampilkan secara visual. GUI dapat dilihat dari perubahan hue, size, visibility, dan sejenisnya saat terjadi koneksi. Secara sederhana fungsi GUI adalah untuk memudahkan pengguna dalam menggunakan software di komputer, karena GUI memudahkan pengguna dengan sekali klik untuk menampilkan sebuah aplikasi, tidak lagi dengan mengetiknya. Salah satu kelebihan dari GUI ini adalah mudah dipahami, hal ini dikarenakan menyederhanakan sistem komputer untuk semua orang, siapa pun dapat mengoperasikannya tanpa kesulitan yang berlebihan.

#### **2. Streamlit**

Streamlit adalah framework yang bersifat open source. Streamlit memudahkan pengguna untuk mengubah data skrip menjadi aplikasi berbasis web interaktif. Selain open source, Streamlit juga bersifat open sharing sehingga mudah untuk dibagikan. Oleh karena itu kami menggunakan Streamlit

ini. Streamlit adalah kerangka kerja berbasis Python dan sumber terbuka yang dibuat untuk memudahkan membangun aplikasi web di bidang ilmu data interaktif dan pembelajaran mesin. Salah satu hal hebat tentang framework ini adalah tidak perlu tahu banyak tentang teknologi pengembangan website. Tidak perlu pusing bagaimana mengatur tampilan website dengan CSS, HTML atau Javascript. Untuk menggunakan Streamlit, hanya perlu memiliki pengetahuan dasar tentang bahasa Python

### 2.1.3 Tuning Hyperparameter

Table 1. Tuning Hyperparameter

	epoch	batch _size	learning _rate	accuracy	val_accu racy	loss	val_loss
0	50	64	0.001	0.913542	0.851042	0.210227	0.339946
1	50	64	0.010	0.902083	0.842708	0.248580	0.1441923
2	50	128	0.001	0.895052	0.806250	0.257930	0.417920
3	50	128	0.010	0.914062	0.851042	0.227586	0.424495
4	100	64	0.001	0.938542	0.858333	0.162239	0.380261
5	100	64	0.010	0.888021	0.854167	0.285735	0.419266
6	100	128	0.001	0.898698	0.794792	0.242091	0.4 33788
7	100	128	0.010	0.917188	0.850000	0.212462	0.376208

Penjelasan :

1. Epoch mengacu pada satu putaran lengkap dari seluruh dataset pelatihan yang digunakan untuk melatih model mesin. Selama satu epoch, setiap sampel dalam dataset pelatihan akan diproses dan digunakan untuk memperbarui parameter model. Jumlah epoch yang diperlukan tergantung pada kompleksitas masalah dan ukuran dataset. Semakin besar jumlah epoch, semakin lama waktu yang dibutuhkan untuk melatih model.
2. Batch size (ukuran batch) adalah jumlah sampel yang diberikan kepada model untuk dievaluasi sebelum parameter diperbarui selama pelatihan. Dalam pelatihan menggunakan mini-batch stochastic gradient descent (SGD), dataset pelatihan dibagi menjadi batch-batch kecil, dan parameter model diperbarui setelah setiap batch dievaluasi. Batch size yang lebih

besar dapat mempercepat pelatihan tetapi memerlukan lebih banyak memori. Batch size yang lebih kecil dapat mengurangi penggunaan memori tetapi mungkin memperlambat pelatihan.

3. Learning rate (tingkat pembelajaran) adalah parameter yang mengontrol seberapa besar langkah yang diambil saat memperbarui parameter model selama pelatihan. Ini menentukan sejauh mana parameter diperbarui berdasarkan gradien yang dihitung selama proses pelatihan. Jika learning rate terlalu besar, pelatihan mungkin tidak stabil atau bahkan dapat menyebabkan model "melompati" optimum global. Jika learning rate terlalu kecil, pelatihan mungkin berjalan lambat atau sulit untuk mencapai hasil yang baik.
4. Akurasi adalah metrik evaluasi yang mengukur sejauh mana model berhasil memprediksi label yang benar. Ini dihitung sebagai persentase jumlah prediksi yang benar dibandingkan dengan total jumlah sampel. Akurasi yang tinggi menunjukkan bahwa model memiliki kemampuan yang baik dalam mengenali pola dan membuat prediksi yang tepat.
5. Val\_accuracy (validasi akurasi) adalah akurasi yang dihitung pada dataset validasi, yang merupakan dataset terpisah yang tidak digunakan dalam pelatihan tetapi digunakan untuk menguji kinerja model yang dilatih. Val\_accuracy memberikan informasi tentang sejauh mana model mampu melakukan generalisasi pada data yang tidak dilihat sebelumnya. Val\_accuracy yang tinggi menunjukkan bahwa model memiliki kinerja yang baik pada data baru.
6. Loss (kehilangan) adalah suatu angka yang menggambarkan sejauh mana model tidak cocok dengan data pelatihan saat ini. Tujuan pelatihan adalah untuk mengurangi kehilangan sebanyak mungkin. Umumnya, ada berbagai fungsi kehilangan yang dapat digunakan tergantung pada jenis masalah yang dihadapi, seperti regresi atau klasifikasi. Misalnya, dalam klasifikasi biner, fungsi kehilangan yang umum digunakan adalah cross-entropy loss.
7. Val\_loss (validasi kehilangan) adalah kehilangan yang dihitung pada dataset validasi. Ini mengukur sejauh mana model tidak cocok dengan data validasi yang tidak digunakan selama pelatihan. Val\_loss yang rendah

menunjukkan bahwa model memiliki kemampuan yang baik untuk memodelkan data yang tidak dilihat sebelumnya.

```
train_hyperparameter = {  
  "epoch": [50, 100],  
  "batch_size": [64, 128],  
  "learning_rate": [0.001, 0.01],  
}
```

Gambar 16. Parameter tuning yang digunakan

## 2.2 Source Code

### 2.2.1 Design Pattern

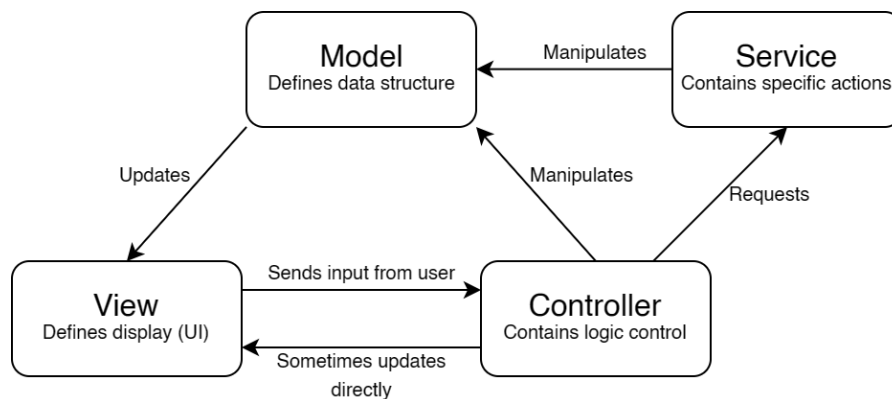
Sistem “*Speech Classification Using Neural Network*” dibuat menggunakan *design pattern* dasar *Model-View-Controller* (MVC). *Design pattern* ini kemudian dikembangkan dengan menambahkan *service layer* sehingga menjadi MVC + *Service layer* (MVCS). *Design pattern* MVC dibangun dengan tiga komponen utama: *model*, *view*, dan *controller*. Pembagian komponen ini menjadi dasar untuk membuat kode yang lebih mudah dikelola dan membantu pemrogram untuk mengembangkan aplikasi web yang dikelola dengan baik. Penggunaan MVC. *Pattern* ini dipilih saat pengembang membangun sistem yang mengandung komponen pemrograman *server-side* yang memerlukan akses ke basis data dan sekaligus dapat menampilkan *interface* ke pengguna [1].

Pengembangan MVC dilakukan menambahkan *service layer*. Layer ini bertujuan untuk menyederhanakan *controller* dengan memindahkan kode dengan tujuan yang lebih spesifik ke *service layer*. *Service layer* dapat diinterpretasikan dengan banyak cara. Biasanya, *layer* ini digunakan untuk inti dari *business logic* yang berada di bawah arsitektur MVC, namun di atas arsitektur akses data. *Service layer* dapat digunakan untuk beberapa keperluan. Contoh penggunaan *service layer* adalah membantu *controller* untuk mengambil atau membuat *model* dari beberapa sumber data, memperbarui nilai di berbagai repositori atau sumber data, dan melakukan logika dan manipulasi khusus aplikasi [2].

Secara ringkas, penjelasan mengenai elemen pada *design pattern* MVCS dapat dilihat pada Tabel 1 dan Gambar 1 [2], [3].

Tabel 2. Ringkasan MVCS

Elemen	Penjelasan
<i>Model</i>	Merepresentasikan dataset untuk disimpan dan dimanipulasi oleh sistem. Model diimplementasikan sebagai tabel basis data atau basis elemen basis data dalam bentuk lain.
<i>View</i>	Merepresentasikan konten yang dapat digunakan kembali atau tampilan yang dapat dilihat oleh pengguna untuk melakukan interaksi.
<i>Controller</i>	Mengoordinasikan interaksi antara elemen <i>view</i> dan <i>model</i> . <i>Controller</i> mencakup serangkaian metode tindakan yang dipicu oleh permintaan <i>view</i> .
<i>Service</i>	Menyediakan layanan tersendiri untuk <i>controller</i> atau <i>model</i> . <i>Service</i> juga dapat digunakan untuk berkomunikasi untuk layanan di luar sistem yang dibuat.



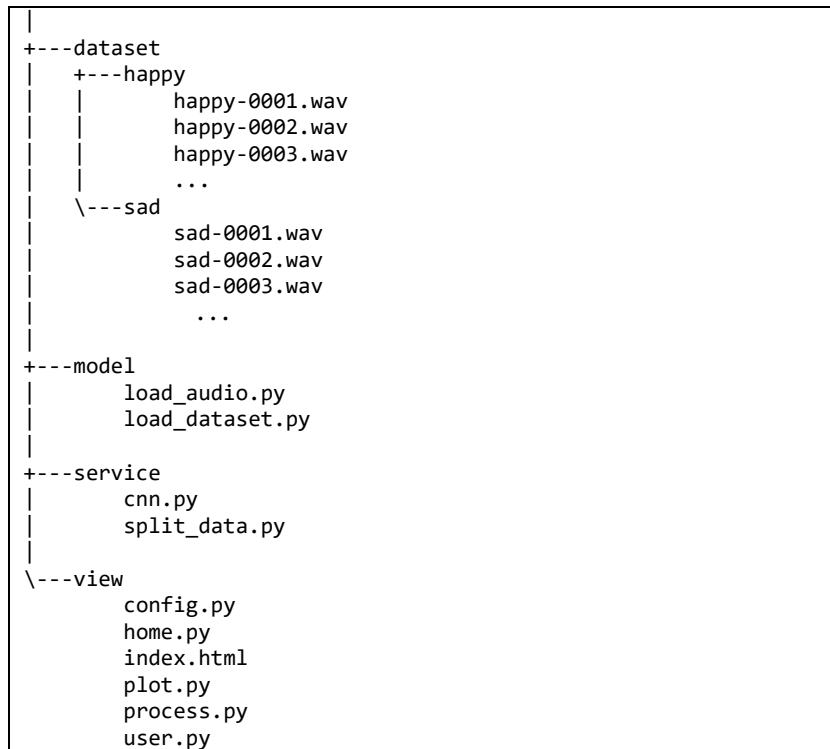
Gambar 1. Design Pattern MVCS

Struktur *file* dan *folder* arsitektur MVCS yang digunakan pada program “*Speech Classification Using Neural Network*” dapat dilihat pada Tabel 2.

Tabel 3. Struktur *File* dan *Folder* Aplikasi

speech-classification-using-neural-network
main.py
readme.md
requirements.txt
+---controller
checking.py
cnn.py
df_dataset.py
mfcc.py





Penjelasan struktur *file* dan *folder* pada Tabel 2 adalah sebagai berikut.

#### 1. *Folder root*

*Folder root* diberi nama “*speech-classification-using-neural-network*” sebagai *folder* induk program. *folder* ini berisi tiga *file* dan lima *folder*. Tiga *file* ini adalah “*main.py*”, “*requirements.txt*”, dan “*readme.md*”. “*main.py*” berfungsi sebagai *entry point* program dan merupakan *file* pertama yang dibaca ketika aplikasi dijalankan. “*requirements.txt*” berisi daftar modul yang digunakan untuk membangun aplikasi. Daftar modul ini juga berisi versi dari setiap modul yang ada di daftar. Modul-modul yang terdaftar di “*requirements.txt*” digunakan untuk membangun *virtual environment* untuk menjalankan aplikasi secara lokal atau pun saat aplikasi di-*deploy*. Terakhir, “*readme.md*” berisi penjelasan aplikasi, teknologi yang digunakan, cara penggunaan, dan kontributor pengembang aplikasi. Selanjutnya, *folder* “*controller*”, “*model*”, “*service*”, dan “*view*” bertindak untuk menampung *file-file* yang digunakan untuk membangun *design pattern* MVCS.

#### 2. *Folder controller*

*Folder* ini berisi *file* yang digunakan untuk melakukan kontrol antara *view* dan *model*. *File* “df\_dataset.py” adalah modul yang berfungsi untuk memuat *dataset* yang digunakan untuk proses *training* dan *testing*. *Dataset* dimuat melalui modul “load\_dataset.py” yang berada di *folder* “model” lalu disimpan dalam bentuk *data frame*. “mfcc.py” berfungsi untuk melakukan *preprocessing* MFCC untuk tiap data suara. “cnn.py” memiliki peran utama dalam aplikasi yang dibangun, yaitu untuk melakukan *training* dan *testing* pada model *machine learning* yang dibuat. Selain itu, proses *predicting* juga terjadi pada modul “cnn.py” sehingga dapat dihasilkan hasil prediksi dari data yang dimasukkan dan dapat dibentuk *confusion matrix*. Terakhir, “checking.py” digunakan untuk proses pengklasifikasian emosi dari *file* audio suara yang dimasukkan oleh pengguna ke aplikasi.

### 3. *Folder dataset*

*Folder dataset* digunakan untuk menyimpan data yang akan digunakan untuk proses *training* dan *testing*. *Dataset* dibagi menjadi dua, *happy* dan *sad* yang diletakkan di dalam *folder* yang bersesuaian. Setiap *dataset* berjumlah 1000 buah, sehingga total *dataset* yang digunakan adalah 2000 buah.

### 4. *Folder model*

*Folder* ini menangani manipulasi *dataset* secara langsung, baik data yang digunakan untuk *training* dan *testing*, maupun data yang diunggah pengguna. “load\_dataset.py” digunakan untuk memuat data audio untuk *training* dan *testing* berdasarkan *path*-nya. “load\_audio.py” digunakan untuk memuat data audio dari *widget File Uploader* milik Streamlit. Data-data yang telah dimuat ini selanjutnya akan digunakan oleh aplikasi di proses yang bersesuaian.

### 5. *Folder service*

*Folder* ini menyediakan modul dengan layanan atau logika yang spesifik. Terdapat dua *file*, yaitu “cnn.py” dan “split\_data.py”. Kedua modul ini berhubungan erat dengan pemodelan *machine learning*, yaitu “cnn.py” yang membuat model “kosong” yang belum di-*training* atau

belum di-fit dengan *dataset* yang menjadi masukan. Selanjutnya, “split\_data.py” menyediakan fungsi untuk melakukan pembagian data untuk proses *training*, *validation*, dan *testing*. Meskipun kedua *file* tersebut dapat dimasukkan ke *controller* untuk *cnn.py* dan ke *model* untuk *split\_data.py*, namun kedua *file* tersebut menyediakan tahapan yang spesifik. Oleh karena itu, akan lebih baik dimasukkan ke *service*.

#### 6. *Folder view*

*Folder* ini menyediakan modul-modul dengan fungsionalitas untuk menyediakan tampilan ke pengguna. “config.py” berfungsi untuk memuat *config default* untuk halaman *home*, *machine modeling*, dan *checking*. *File* “home.py” merupakan *file* yang berfungsi untuk menampilkan halaman *home*. Halaman ini berisi nama aplikasi, spesifikasi model, cara penggunaan aplikasi, dan kontributor yang disediakan oleh “index.html”. Selanjutnya, “plot.py” digunakan untuk menampilkan bagan pada halaman *machine modeling*. “process.py” berfungsi untuk menampilkan halaman *machine modeling* yang menjadi halaman untuk pengguna berinteraksi saat proses pembuatan model. Terakhir, “user.py” berfungsi untuk menampilkan halaman untuk pengecekan (mengklasifikasikan) suara yang dimasukkan oleh pengguna.

Pada bagian ini akan dijelaskan *source code* program yang terbagi sesuai dengan fungsinya, baik sebagai *entry point*, *model*, *view*, *controller*, dan *service*.

#### 2.2.2 Entry Point (Main Module)

Tabel 4. *Source Code* “main.py”

```
1. # import the home view
2. from view.home import start_home
3. # show the home for the first time app starts
4. if __name__ == "__main__":
5.     start_home()
```

Modul “main.py” merupakan *entry point* aplikasi. Modul ini menjadi modul yang pertama kali dimuat oleh saat aplikasi dimulai. “main.py” berisi pemanggilan fungsi *start\_home()* yang merupakan *graphical user interface* (GUI) untuk interaksi pengguna. GUI yang dibuat berbentuk web dengan

menggunakan *framework front-end* Streamlit. *Framework* ini dikelola pada *folder view*.

### 2.2.3 Model

#### 1. load\_audio.py

Tabel 5. Source Code "load\_audio.py"

```
1. # import librosa module
2. import librosa
3. # function to load audio files
4. def load(uploaded_files):
5.     # prepare empty list to store audio
6.     audio_librosa = []
7.     # load the audio data from uploaded files then append it to the list
8.     for uploaded_file in uploaded_files:
9.         audio, _ = librosa.load(uploaded_file, sr = 16000)
10.        audio_librosa.append(audio)
11.    # return the list
12.    return audio_librosa
```

Modul “load\_audio.py” berfungsi untuk memuat audio yang diunggah pengguna. Data audio didapatkan dari *widget file uploader* yang disediakan oleh Streamlit. Modul ini menggunakan utilitas dari modul *librosa* untuk mendapatkan *waveform* data audio yang diunggah. *Sampling rate* yang digunakan adalah 16.000 yang merupakan *sampling rate default*. *Sampling rate* ini juga digunakan pada proses *training model machine learning*. Saat semua data audio yang diunggah selesai dimuat, data audio tersebut akan dikembalikan ke fungsi pemanggil.

#### 2. load\_dataset.py

Tabel 6. Source Code "load\_dataset.py"

```
1. # import necessary modules
2. import os
3. import librosa
4. # function to load dataset from paths
5. def load(path):
6.     # prepare empty dictionary to store dataset (path, label, duration)
7.     dataset = {
8.         "paths": [],
9.         "labels": [],
10.        "durations": []
11.    }
12.    # walk through the path and append the data to the dictionary
13.    for dirname, _, filenames in os.walk(path):
14.        for filename in filenames:
15.            path = os.path.join(dirname, filename)
16.            dataset["paths"].append(path)
17.            label = filename.split("-")[0]
18.            dataset["labels"].append(label)
19.            duration = round(librosa.get_duration(path = path), 4)
20.            dataset["durations"].append(duration)
21.    # return the dictionary
22.    return dataset
```

Modul ini digunakan untuk memuat data audio berdasarkan *path* penyimpanannya. Data yang dimuat dan disimpan adalah *path*-nya, label emosi, dan durasi dari masing-masing data audio. *Path* diambil dari *path* relatif berdasarkan *root* aplikasi, label merupakan nama file pertama audio yang dipisahkan oleh tanda hubung, dan durasi diambil menggunakan modul *librosa*. Selanjutnya *dataset* yang telah dimuat, akan dikembalikan ke fungsi pemanggil.

## 2.2.4 View

### 1. config.py

Tabel 7. Source Code "config.py"

```

1. # import streamlit module
2. import streamlit as st
3. # get config function to start page config (for each page)
4. def get_config(page_title):
5.     # set app layout, set page title (on tab), set menu items in "about"
    section
6.     title = "Speech Clasification Using Neural Network"
7.     st.set_page_config(layout='wide', page_title=f"{page_title} |
{title}", menu_items={
8.         'About': f"""
9.             ### {title}
10.            Made with :heart: by Group 1 Class A
11.            Introduction to Multimedia Data Processing Subject
12.            GitHub: https://github.com/tudemaha/speech-emotion-
classification
13.            """
14.        })

```

Modul ini berfungsi untuk memuat *default config* untuk tiap halaman aplikasi web. Bagian yang diset oleh modul ini adalah *layout* web, judul halaman yang ditampilkan di *tab*, dan bagian “*about*” yang ada pada detail aplikasi.

### 2. home.py

Tabel 8. Source Code "home.py"

```

1. # import necessary modules
2. from st_pages import Page, show_pages
3. import streamlit.components.v1 as components
4. # import config view
5. from view.config import get_config
6. # start home function
7. def start_home():
8.     # prepare current page config, set the page's title
9.     get_config("Home")
10.    # open the html home page to be rendered
11.    render_page = open("view/index.html")
12.    # render the html home
13.    components.html(render_page.read(), height = 2000)
14.    # create pages (home, machine modeling and checking)
15.    show_pages(
16.        [

```

```

17.         Page("main.py", "Home", ":house:"),
18.         Page("view/process.py", "Machine Modeling", ":gear:"),
19.         Page("view/user.py", "Checking", ":sound:")
20.     ]
21. )

```

Modul ini berfungsi untuk *me-render* halaman *home* aplikasi. Halaman yang dirender berasal dari *file* “index.html” sebagai komponen tersendiri. Modul ini juga berfungsi untuk menyiapkan *menu multipage* untuk menuju ke halaman lain, yaitu *Machine Modeling* dan *Checking*. Modul ini juga memanggil *get\_config()* untuk mengeset nilai *default* halaman *home*.

### 3. plot.py

Tabel 9. Source Code "plot.py"

```

1. # import necessary modules
2. import pandas as pd
3. import numpy as np
4. import random
5. import librosa
6. import seaborn as sns
7. import streamlit as st
8. import matplotlib.pyplot as plt
9. from sklearn.metrics import ConfusionMatrixDisplay
10. from matplotlib.colors import Normalize
11. # function to plot dataset distribution
12. def distribution(dataset):
13.     # create figure and axes (left: emotion distribution, right: duration
distribution)
14.     fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize = (20, 5))
15.     # group the dataset by label and plot the distribution
16.     dataset.groupby(["Label"]).size().plot(kind = "bar", ax = ax[0])
17.     ax[0].set_title("Dataset Emotion Distribution", size = 14)
18.     ax[0].set_ylabel("number of samples")
19.     # group the dataset by label and plot the duration distribution using
violin plot
20.     sns.violinplot(x = dataset["Label"], y = dataset["Duration"],
linewidth = 1, ax = ax[1])
21.     ax[1].set_title("Duration Distribution per Emotion")
22.     ax[1].set_ylabel("duration (second)")
23.     # show the plot
24.     st.pyplot(fig)
25. # function to plot random dataset picked
26. def show_random_plot(df):
27.     # create figure and axes (left: waveform, middle: spectrogram, right:
mfcc)
28.     fig, ax = plt.subplots(nrows = 1, ncols = 3, figsize = (20, 4))
29.     # pick random index from dataset
30.     index = random.randint(0, df.shape[0])
31.     # load the audio file based on the path
32.     y, sr = librosa.load(df.Path[index], sr = 16000)
33.     # plot the waveform
34.     librosa.display.waveshow(y, sr = sr, ax = ax[0])
35.     ax[0].set_title("Waveform")
36.     # plot the spectrogram with fundamental frequency
37.     f0, _, _ = librosa.pyin(y, sr = sr, fmin = 50, fmax = 1500,
frame_length = 2048)
38.     timepoints = np.linspace(0, df.Duration[index], num = len(f0),
endpoint = False)
39.     x_stft = np.abs(librosa.stft(y))
40.     x_stft = librosa.amplitude_to_db(x_stft, ref = np.max)

```

```

41.     librosa.display.specshow(x_stft, sr = sr, x_axis = "time", y_axis =
"log", ax = ax[1])
42.     ax[1].plot(timepoints, f0, color = "cyan", linewidth = 4)
43.     ax[1].set_title("Spectrogram with Fundamental Frequency")
44.     # plot the mfcc
45.     x_mfcc = librosa.feature.mfcc(y = y, sr = sr, n_mfcc = 20)
46.     librosa.display.specshow(x_mfcc, sr = sr, x_axis = "time", norm =
Normalize(vmin = -50, vmax = 50), ax = ax[2])
47.     ax[2].set_title("MFCC")
48.     # show the path of picked audio file as title
49.     plt.suptitle(df.Path[index], size = 14)
50.     # use tight layout to prevent overlapping
51.     plt.tight_layout()
52.     # show the plot
53.     st.pyplot(fig)
54. # function to plot mfcc before and after equalization
55. def show_mfcc(df_mfcc_path, array_mfcc):
56.     # create figure and axes (left: mfcc before augmentation, right: mfcc
after equalization)
57.     fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize = (20, 4))
58.     # load the audio file based on the path
59.     y, sr = librosa.load(df_mfcc_path, sr = 16000)
60.     # plot the mfcc before augmentation
61.     x_mfcc = librosa.feature.mfcc(y = y, sr = sr, n_mfcc = 30)
62.     librosa.display.specshow(x_mfcc, sr = sr, x_axis = "time", norm =
Normalize(vmin = -50, vmax = 50), ax = ax[0])
63.     ax[0].set_title("Before")
64.     # plot the mfcc after equalization
65.     librosa.display.specshow(array_mfcc, sr = sr, x_axis = "time", norm =
Normalize(vmin = -50, vmax = 50), ax = ax[1])
66.     ax[1].set_title("After")
67.     # show the path of picked audio file as title
68.     plt.suptitle(df_mfcc_path, size = 14)
69.     # show the plot
70.     st.pyplot(fig)
71. # function to plot the model's training history
72. def plot_history(history):
73.     # create figure and axes (left: loss, right: accuracy)
74.     fig, ax = plt.subplots(ncols = 2, nrows = 1, figsize = (20, 5))
75.     plt.suptitle("CNN with MFCCs", size = 14)
76.     # make dataframe from training history
77.     results = pd.DataFrame(history.history)
78.     # plot the loss and validation loss for last 3 epochs
79.     results[["loss", "val_loss"]].plot(ax = ax[0])
80.     val_loss_mean = np.mean(history.history["val_loss"][-3:])
81.     ax[0].set_title(f"Validation loss {round(val_loss_mean, 3)} (mean
last 3)")
82.     # plot the accuracy and validation accuracy for last 3 epochs
83.     results[["accuracy", "val_accuracy"]].plot(ax = ax[1])
84.     val_accuracy_mean = np.mean(history.history["val_accuracy"][-3:])
85.     ax[1].set_title(f"Validation accuracy {round(val_accuracy_mean, 3)}
(mean last 3)")
86.     # show the plot
87.     st.pyplot(fig)
88. # function to plot the confusion matrix
89. def confusion_matrix(pred, y_te):
90.     # prepare numbered labels from string labels
91.     labels = {"sad": 0, "happy": 1}
92.     # create figure and axes (left: confusion matrix (count), right:
confusion matrix (ratio))
93.     fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize = (20, 8))
94.     # plot the confusion matrix (count)
95.     ConfusionMatrixDisplay.from_predictions(y_te, pred, display_labels =
labels, ax = ax[0])
96.     ax[0].set_title("Confusion Matrix (count)", size = 14)
97.     # plot the confusion matrix (ratio)

```

```

98.     ConfusionMatrixDisplay.from_predictions(y_te, pred, display_labels =
labels, normalize = "true", ax = ax[1])
99.     ax[1].set_title("Confusion Matrix (ratio)", size = 14)
100.    # show the plot
101.    st.pyplot(fig)

```

Modul ini memiliki fungsi spesifik untuk menampilkan bagan (*chart*) pada proses *preprocessing*. Bagan yang ditampilkan menggunakan beberapa modul, seperti *matplotlib* dan *seaborn*, *sklearn*, dan *librosa*. Modul ini memiliki lima fungsi, yaitu *distribution()*, *show\_random()*, *show\_mfcc()*, *plot\_history()*, dan *confusion\_matrix()*. Fungsi *distribution()* digunakan untuk menampilkan distribusi *dataset* berdasarkan label emosinya dan berdasarkan durasi masing-masing *dataset*. Fungsi *show\_random()* digunakan untuk menampilkan satu sampel *dataset* yang dipilih acak. Tujuan pengambilan sampel ini adalah untuk mengetahui perbedaan *dataset happy* dan *sad* berdasarkan *waveform*, *spectrogram*, dan MFCC-nya. Fungsi *show\_mfcc()* digunakan untuk menampilkan MFCC sebelum dan setelah disesuaikan. Fungsi *plot\_history()* untuk menampilkan riwayat saat *training* berlangsung. Terakhir, fungsi *confusion\_matrix()* digunakan untuk menampilkan *confusion matrix* berdasarkan proses *testing*.

#### 4. process.py

Tabel 10. Source Code "process.py"

```

1. # import necessary modules
2. import streamlit as st
3. import random
4. # import app modules (controller, service, and view)
5. from controller.df_dataset import dataset_df
6. from controller.mfcc import create_mfcc, create_resized_mfcc
7. from controller.cnn import train, test, predict
8. from service.split_data import make_train_test_split
9. from view.plot import distribution, show_random_plot, confusion_matrix,
show_mfcc, plot_history
10. from view.config import get_config
11. # prepare current page config, set the page's title
12. get_config("Machine Modeling")
13. # prepare all needed session states to make interactive interaction
14. if "preprocessing" not in st.session_state:
st.session_state["preprocessing"] = False
15. if "train" not in st.session_state: st.session_state["train"] = False
16. if "test" not in st.session_state: st.session_state["test"] = False
17. if "model" not in st.session_state: st.session_state["model"] = False
18. if "x_mean" not in st.session_state: st.session_state["x_mean"] = False
19. if "x_std" not in st.session_state: st.session_state["x_std"] = False
20. # start the machine modeling page
21. # used for preprocessing dataset, training and testing model
22. def start():
23.     # show the page title
24.     st.write("<h1 style='text-align: center;'>Machine Modeling</h1>",
unsafe_allow_html = True)
25.     # show training dataset section

```



```

26.     st.write("### Training Dataset")
27.     # set imported dataset to global
28.     global df
29.     # load dataset
30.     df = dataset_df('dataset')
31.     # show dataset
32.     st.dataframe(df, 500)
33.     # show message and button for next interaction (preprocessing)
34.     st.write("Dataset have been loaded, let's preprocess them!")
35.     # set the preprocessing state to true if user click the button
36.     if st.button("Preprocessing", type = "primary"):
37.         st.session_state["preprocessing"] = True
38. # preprocessing function to start preprocessing
39. def preprocessing():
40.     # show sample distribution
41.     st.write("### Sample Distribution")
42.     distribution(df)
43.     # random one pick sample
44.     st.write("### Random Pick Sample")
45.     # split dataset to happy and sad based on their label, reset the
dataframe index
46.     happy_df = df.loc[df["Label"] == "happy"].reset_index(drop = True)
47.     sad_df = df.loc[df["Label"] == "sad"].reset_index(drop = True)
48.     # plot 1 happy and 1 sad dataset
49.     st.write("#### Happy")
50.     show_random_plot(happy_df)
51.     st.write("#### Sad")
52.     show_random_plot(sad_df)
53.     # create the mfccs from each audio path
54.     mfccs = create_mfcc(df)
55.     # set the new_mfccs as global to be used in other function
56.     global new_mfccs
57.     # get the new mfccs (with the equal column number) and get the
average column number of old mfccs
58.     new_mfccs, average_cols = create_resized_mfcc(mfccs)
59.     # show the average mfccs column (old mfccs)
60.     st.write("### Average MFCCs Column")
61.     st.metric("Average", average_cols)
62.     # pick one random index to show the difference of old and new mfcc
63.     index = random.randint(0, df.shape[0])
64.     # show picked mfcc, before and after equalization
65.     st.write("### MFCC Comparison")
66.     show_mfcc(df.Path[index], new_mfccs[index])
67.     # x_tr => training data, y_tr => training label
68.     # x_va => validation data, y_va => validation label
69.     # x_te => testing data, y_te => testing label
70.     global x_tr, y_tr, x_va, y_va, x_te, y_te
71.     # split the dataset into training, validation, and testing
72.     x_tr, y_tr, x_va, y_va, x_te, y_te = make_train_test_split(new_mfccs,
df)
73.     # show the shape of mfccs distribution (training, validation,
testing)
74.     st.write("### MFCCs Distribution")
75.     col1, col2, col3 = st.columns(3)
76.     col1.metric("Train", str(x_tr.shape))
77.     col2.metric("Validation", str(x_va.shape))
78.     col3.metric("Test", str(x_te.shape))
79. # start training process
80. def start_train():
81.     # show the training process section
82.     st.write("### Training Process")
83.     # start training
84.     model, history = train(x_tr, y_tr, x_va, y_va)
85.     # store the trained model in session state to be used in another
function and page
86.     st.session_state["model"] = model

```

```

87.     # plot the training history (loss, validation loss, accuracy, and
validation accuracy)
88.     plot_history(history)
89. # start testing process
90. def start_test():
91.     # show the testing process section
92.     st.write("### Testing Process")
93.     # get the trained model from session state
94.     trained_model = st.session_state["model"]
95.     # test the model using testing data (return the loss and accuracy)
96.     loss, accuracy = test(trained_model, x_te, y_te)
97.     # get the prediction, precision, recall, and f1 score
98.     prediction, precision, recall, f1 = predict(trained_model, x_te,
y_te)
99.     # display the testing metrics (loss, accuracy, precision, recall, and
f1)
100.    col1, col2, col3, col4, col5 = st.columns(5)
101.    col1.metric("Loss", "{:.4f}".format(loss))
102.    col2.metric("Accuracy", "{:.4f}".format(accuracy))
103.    col3.metric("Precision", "{:.4f}".format(precision))
104.    col4.metric("Recall", "{:.4f}".format(recall))
105.    col5.metric("F1", "{:.4f}".format(f1))
106.    # plot the confusion matrix
107.    st.write("### Confusion Matrix")
108.    confusion_matrix(y_te, prediction)
109. # call start() function at the first time page load
110. if __name__ == "__main__":
111.     start()
112. # if the preprocessing state is true
113. if st.session_state["preprocessing"]:
114.     # start preprocessing
115.     preprocessing()
116.     # show message and button for next interaction (testing)
117.     st.write("The dataset have been preprocessed, let's train them using
CNN!")
118.     # if training button pressed, change train state to true
119.     if st.button("Training", type = "primary"):
120.         st.session_state["train"] = True
121. # if the train state is true
122. if st.session_state["train"]:
123.     # start training process
124.     start_train()
125.     # show message and button for next interaction (testing)
126.     st.write("Training model finish, let's check the accuraction with
testing dataset!")
127.     # if testing button pressed, change test state to true
128.     if st.button("Testing", type = "primary"):
129.         st.session_state["test"] = True
130. # if the test state is true
131. if st.session_state["test"]:
132.     # start testing process
133.     start_test()

```

Modul ini berfungsi untuk menampilkan halaman “*Machine Modeling*”.

Tahap yang terjadi pada halaman ini adalah *load dataset*, *preprocessing*, *training*, dan *testing*. Proses yang terjadi dikontrol dengan menggunakan *session state*, sehingga pengguna bisa berinteraksi dengan aplikasi dengan menggunakan tombol-tombol yang disediakan. Tujuan lain disediakanya tombol untuk pengguna adalah agar halaman tidak berat untuk dimuat saat

pertama kali. Proses akan terjadi hanya saat pengguna ingin menuju ke langkah yang spesifik.

#### 5. user.py

Tabel 11. Source Code "user.py"

```
1. # import necessary modules
2. import streamlit as st
3. from pandas import DataFrame
4. # import app module (controller and view)
5. from controller.checking import checking
6. from view.config import get_config
7. # get config function to start page config (for each page)
8. get_config("Checking")
9. # start the ckecking page
10. def start():
11.     # show the page title
12.     st.write("<h1 style='text-align: center;'>Let's Check Your Emotion!</h1>", unsafe_allow_html = True)
13.     # show instructions for user
14.     instructions = """
15.     Instructions:
16.     1. Prepare your files
17.     2. The files must contain human speech
18.     2. Make sure each file is between 3 to 5 seconds long
19.     3. Upload the files
20.     4. Listen to the audio before processing (if necessary)
21.     5. Process them!
22.     """
23.     st.markdown(instructions)
24.     # show file uploader to upload speech audio
25.     uploaded_files = st.file_uploader("Input Audio :sound:", ['wav', 'avi', 'aac', 'mp3'], accept_multiple_files = True)
26.     # if user upload the files
27.     if uploaded_files != []:
28.         # show the uploaded files with their name and audio player
29.         st.write("#### Uploaded Files")
30.         for uploaded_file in uploaded_files:
31.             col1, col2 = st.columns([0.2, 0.8])
32.             col1.write(uploaded_file.name)
33.             col2.audio(uploaded_file)
34.         # if user click the process button
35.         if st.button("Process", type = "primary"):
36.             # show the predicton result
37.             st.write("#### Predicted Emotions")
38.             # start prediction process
39.             pred = checking(uploaded_files)
40.             # show the result (if sad show snow, if happy show balloons)
41.             if 0 in pred: st.snow()
42.             if 1 in pred: st.balloons()
43.             # prepare the emoji dictionary
44.             emoji = {"0": "😞", "1": "😄"}
45.             # prepare the result list
46.             result = []
47.             # append the result to the list with their filename and predicted emotion's emoji
48.             for i, p in enumerate(pred):
49.                 result.append((uploaded_files[i].name, emoji[str(p)]))
50.
51.             # show the result in dataframe
52.             result_df = DataFrame(result, columns = ["Filename", "Emotion"])
53.             st.dataframe(result_df, 400)
54. # if the traing process has not been done
```

```

55. def show_warning():
56.     # show warning message for user to go to machine modeling page
57.     st.warning("Model not created yet! Go to \"Machine Modeling\" page, do preprocessing, training, and testing.", icon = "⚠")
58. # check if the model has been tested
59. if st.session_state["test"]:
60.     # start the checking page
61.     start()
62. else:
63.     # show warning message
64.     show_warning()

```

Modul ini digunakan untuk menampilkan halaman *Checking*. Halaman ini berfungsi sebagai tempat untuk mengunggah *file* audio dari pengguna. Pengguna bisa mengunggah lebih dari satu audio suara untuk diklasifikasikan emosinya berdasarkan model yang telah di-*training*. Pada halaman ini juga ditampilkan *player* yang dapat digunakan pengguna untuk memastikan audio yang diunggah. Hasil dari klasifikasi adalah tabel yang berisi nama *file* yang diunggah dan emosi audionya.

### 2.2.5 Controller

#### 1. checking.py

Tabel 12. Source Code "checking.py"

```

1. # import necessary modules
2. import numpy as np
3. import streamlit as st
4. # import app module (model and controller)
5. from model.load_audio import load
6. from controller.mfcc import create_audio_mfcc, create_resized_mfcc
7. from controller.cnn import make_prediction
8. # function to check/predict the uploaded audio
9. def checking(uploaded_files):
10.     # load the uploaded audio into librosa format
11.     audio_librosa = load(uploaded_files)
12.     # create mfccs from the audio
13.     mfccs = create_audio_mfcc(audio_librosa)
14.     # create resized mfccs from the mfccs (equalize the length of mfccs)
15.     new_mfccs, _ = create_resized_mfcc(mfccs)
16.     # convert the mfccs into numpy array
17.     new_mfccs = np.array([i for i in new_mfccs])
18.     # normalize the mfccs
19.     new_mfccs = (new_mfccs - st.session_state["x_mean"]) /
st.session_state["x_std"]
20.     # add channel dimension to the mfccs
21.     new_mfccs = new_mfccs[..., None]
22.     # load the model from session state
23.     model = st.session_state["model"]
24.     # make prediction from the model
25.     pred = make_prediction(model, new_mfccs)
26.     # return the prediction
27.     return pred

```

Modul “checking.py” berfungsi untuk mengatur proses prediksi emosi dari masukan pengguna. Modul ini menangani *flow* untuk menerima *file* dari

pengguna, melakukan *preprocessing*, memuat model *machine learning*, dan melakukan prediksi.

## 2. cnn.py

Tabel 13. Source Code "cnn.py"

```
1. # import necessary modules
2. import numpy as np
3. from tensorflow import keras
4. from sklearn.metrics import recall_score, precision_score, f1_score
5. import streamlit as st
6. # import app module (service)
7. from service.cnn import create_model
8. # function to train the model
9. @st.cache_resource
10. def train(x_tr, y_tr, x_va, y_va):
11.     # create the model
12.     model = create_model(x_tr)
13.     # prepare the Adam optimizer
14.     optimizer = keras.optimizers.Adam()
15.     # compile the model
16.     model.compile(optimizer = optimizer, loss =
"sparse_categorical_crossentropy", metrics = ["accuracy"])
17.     # prepare early stopping callback
18.     earlystopping_cb = keras.callbacks.EarlyStopping(patience = 5)
19.     # train the model
20.     history = model.fit(
21.         x = x_tr,
22.         y = y_tr,
23.         epochs = 100,
24.         batch_size = 32,
25.         validation_data = (x_va, y_va),
26.         callbacks = [earlystopping_cb]
27.     )
28.     # return the model and history
29.     return model, history
30. # function to test the model
31. def test(model, x_te, y_te):
32.     # evaluate the model
33.     loss_te, accuracy_te = model.evaluate(x_te, y_te)
34.     # return the loss and accuracy
35.     return loss_te, accuracy_te
36. # function to make prediction
37. def make_prediction(model, x_te):
38.     # make prediction from trained model
39.     predictions = model.predict(x_te)
40.     # prepare empty list to store the prediction
41.     pred = []
42.     # loop through the predictions and append the prediction to the list
43.     for i in predictions:
44.         pred.append(np.argmax(i))
45.     # return the prediction
46.     return pred
47. # function to predict the test set
48. def predict(model, x_te, y_te):
49.     # make prediction
50.     pred = make_prediction(model, x_te)
51.     # calculate the precision, recall, and f1 score
52.     precision = precision_score(y_te, pred, average = "weighted")
53.     recall = recall_score(y_te, pred, average = "weighted")
54.     f1 = f1_score(y_te, pred, average = "weighted")
55.     # return the prediction, precision, recall, and f1 score
56.     return pred, precision, recall, f1
```

Modul “cnn.py” berfungsi untuk menangani *flow* dari proses *modeling*, baik *training* maupun *testing*. Beberapa fungsi yang disediakan, yaitu *train()*, *test()*, *make\_prediction()*, dan *predict()*. Fungsi *train()* digunakan untuk melakukan *training* pada model *machine learning*. Fungsi *test()* untuk melakukan *testing*. Fungsi *make\_prediction()* untuk melakukan prediksi dengan mengembalikan label prediksi untuk tiap audio. Fungsi *predict()* untuk melakukan mengambil label prediksi, *precision*, *recall*, dan F1 score.

### 3. df\_dataset.py

Tabel 14. Source Code "df\_dataset.py"

```
1. # import necessary modules
2. import pandas as pd
3. import streamlit as st
4. # import app module (model)
5. from model.load_dataset import load
6. # function to create dataframe from dataset
7. @st.cache_data
8. def dataset_df(path):
9.     # load the dataset
10.    dataset_dict = load(path)
11.    # create dataframe from the dataset
12.    df = pd.DataFrame(dataset_dict)
13.    # rename the columns
14.    df.columns = ["Path", "Label", "Duration"]
15.    # return the dataframe
16.    return df
```

Modul “df\_dataset.py” berfungsi untuk berkomunikasi dengan model, yaitu “load\_dataset.py”. Modul ini menerima *dataset* yang telah dimuat, yaitu detail *path*, label, dan durasi dari masing-masing data. Selanjutnya, data yang telah dimuat dibentuk menjadi *data frame* untuk dikembalikan ke fungsi pemanggil.

### 4. mfcc.py

Tabel 15. Source Code "mfcc.py"

```
1. # import necessary modules
2. import numpy as np
3. import math
4. import librosa
5. import streamlit as st
6. # function to create mfccs from the dataframe
7. @st.cache_data
8. def create_mfcc(df):
9.     # prepare empty list to store the mfccs
10.    mfccs = []
11.    # loop through the dataframe and create mfccs from the audio
12.    for path in df.Path:
13.        y, sr = librosa.load(path, sr = 16000)
14.        mfcc = librosa.feature.mfcc(y = y, sr = sr, n_mfcc = 30)
15.        mfccs.append(mfcc)
16.    # return the mfccs
17.    return mfccs
```

```

18. # function to create mfccs from the uploaded audio
19. @st.cache_data
20. def create_audio_mfcc(librosa_audio):
21.     # prepare empty list to store the mfccs
22.     mfccs = []
23.     # loop through the uploaded audio and create mfccs from the audio
24.     for audio in librosa_audio:
25.         mfcc = librosa.feature.mfcc(y = audio, sr = 16000, n_mfcc = 30)
26.         mfccs.append(mfcc)
27.     # return the mfccs
28.     return mfccs
29. # function to resize the mfccs
30. @st.cache_resource
31. def resize_mfcc(array):
32.     # create empty array with shape (30, 80)
33.     new_mfcc = np.zeros((30, 80))
34.     # loop through the array and copy the value to the new array
35.     # if the mfcc length less than 80, the rest of the array will be
filled with 0
36.     # if the mfcc length more than 80, the rest of the array will be
ignored
37.     for i in range(30):
38.         for j in range(80):
39.             try:
40.                 new_mfcc[i][j] = array[i][j]
41.             except IndexError:
42.                 pass
43.     # return the new resized mfcc
44.     return new_mfcc
45. # function to create resized mfccs from the mfccs
46. @st.cache_data
47. def create_resized_mfcc(mfccs):
48.     # prepare empty list to store the resized mfccs
49.     new_mfccs = []
50.     # prepare variable to store the sum of mfccs column
51.     sum = 0
52.     # loop through the mfccs and create resized mfccs from the mfccs
53.     # also calculate the sum of mfccs column
54.     for mfcc in mfccs:
55.         new_mfccs.append(resize_mfcc(mfcc))
56.         sum += mfcc.shape[1]
57.     # calculate the average of mfccs column
58.     averate_column = math.ceil(sum / 2000)
59.     # return the resized mfccs and the average of mfccs column
60.     return new_mfccs, averate_column

```

Modul ini menangani pembuatan MFCC. Terdapat empat fungsi pada modul ini, yaitu *create\_mfcc()*, *create\_audio\_mfcc()*, *resize\_mfcc()*, dan *create\_resized\_mfcc()*. Fungsi *create\_mfcc()* digunakan untuk membuat MFCC dari *path* yang disediakan. Fungsi *create\_audio\_mfcc()* berfungsi mirip dengan fungsi *create\_mfcc()*, tetapi masukannya adalah audio yang berbentuk *UploadedFile* sehingga tidak perlu memuat audio terlebih dahulu berdasarkan *path*-nya. Fungsi *resize\_mfcc()* berfungsi untuk menyeragamkan kolom pada satu buah MFCC. Terakhir, fungsi *create\_resized\_mfcc()* untuk membuat *list* yang berisi semua MFCC yang telah diseragamkan kolomnya.

## 2.2.6 Service

### 1. cnn.py

Tabel 16. Source Code "cnn.py"

```

1. # import necessary modules
2. from tensorflow import keras
3. from keras.layers import (Conv2D, BatchNormalization, Dropout, Flatten,
Dense, MaxPool2D)
4. # function to create the model (CNN)
5. def create_model(x_tr):
6.     # create sequential model from keras
7.     model = keras.Sequential()
8.     # add first convolutional layer, max pooling layer, and batch
normalization layer
9.     model.add(Conv2D(filters = 64, kernel_size = 5, strides = (2, 2),
activation = "relu", input_shape = x_tr.shape[1:]))
10.    model.add(MaxPool2D(pool_size = 2))
11.    model.add(BatchNormalization())
12.    # add second convolutional layer, max pooling layer, and batch
normalization layer
13.    model.add(Conv2D(filters = 32, kernel_size = 4, strides = (2, 1),
activation = "relu"))
14.    model.add(MaxPool2D(pool_size = 2))
15.    model.add(BatchNormalization())
16.    # add flatten layer
17.    model.add(Flatten())
18.    # add first dense layer, dropout layer
19.    model.add(Dropout(0.5))
20.    model.add(Dense(128, activation = "relu"))
21.    # add second dense layer, dropout layer
22.    model.add(Dropout(0.5))
23.    model.add(Dense(64, activation = "relu"))
24.    # add third dense layer, dropout layer
25.    model.add(Dropout(0.5))
26.    model.add(Dense(units = 7, activation = "softmax"))
27.    # show the summary of the model
28.    model.summary()
29.    # return the model
30.    return model

```

Modul ini berfungsi untuk membuat model *Convolutional Neural Network* (CNN). Model yang dibuat menggunakan bantuan modul Keras yang disediakan oleh TensorFlow. Model “kosong” dibuat dengan mendefinisikan *input layer*, *hidden layer*, dan *output layer*-nya. Terakhir, ditampilkan ringkasan model yang dibuat lalu model dikembalikan ke fungsi pemanggil.

## 2. split\_data.py

Tabel 17. Source Code "split\_data.py"

```

1. # import necessary modules
2. from sklearn.model_selection import train_test_split
3. import numpy as np
4. import streamlit as st
5. # function to split the dataset into train, validation, and test set
6. @st.cache_data
7. def make_train_test_split(x, y):
8.     # replace string label with number
9.     y["Label"].replace({"sad": 0, "happy": 1}, inplace = True)
10.    # get the values from dataframe
11.    y = y.Label.values
12.    # copy the dataset input

```



```

13.     x = x.copy()
14.     # split the dataset into train and test set (80:20)
15.     x_tr, x_te, y_tr, y_te = train_test_split(x, y, train_size = 0.8,
shuffle = True, random_state = 0)
16.     # split the train set into train and validation set (80:20)
17.     x_tr, x_va, y_tr, y_va = train_test_split(x_tr, y_tr, test_size = 0.2,
shuffle = True, random_state = 0)
18.     # convert the dataset into numpy array
19.     x_tr = np.array([i for i in x_tr])
20.     x_va = np.array([i for i in x_va])
21.     x_te = np.array([i for i in x_te])
22.     # get the mean and standard deviation of the train set
23.     tr_mean = np.mean(x_tr, axis = 0)
24.     tr_std = np.std(x_tr, axis = 0)
25.     # save the mean and standard deviation of the train set to session
state
26.     st.session_state["x_mean"] = tr_mean
27.     st.session_state["x_std"] = tr_std
28.     # normalize the dataset
29.     x_tr = (x_tr - tr_mean) / tr_std
30.     x_va = (x_va - tr_mean) / tr_std
31.     x_te = (x_te - tr_mean) / tr_std
32.     # add channel dimension to the dataset
33.     x_tr = x_tr[..., None]
34.     x_va = x_va[..., None]
35.     x_te = x_te[..., None]
36.     # return the splitted dataset
37.     return x_tr, y_tr, x_va, y_va, x_te, y_te

```

Modul ini bertugas untuk melakukan *splitting dataset*. Data dibagi menjadi data *training*, *validation*, dan *testing*. Pengubahan bentuk data menjadi NumPy *array* juga dilangsungkan pada modul ini. Normalisasi dan penambahan *channel* merupakan *preprocessing* yang dilakukan pada modul ini. fungsi *make\_train\_test\_split()* mengembalikan data *training*, *validation*, dan *testing* beserta labelnya. Perincian *splitting* yang dilakukan adalah 80% untuk proses *training* dan 20% *dataset* untuk *testing*. Selanjutnya, 80% dari data *training* untuk keperluan *pure training* dan 20% dari data *training* untuk proses validasi.

## **BAB III**

### **PENUTUP**

#### **3.1 Kesimpulan**

Program aplikasi "*Speech Classification Using Neural Network*" dibangun menggunakan algoritma jaringan saraf tiruan *Convolutional Neural Network* (CNN), *Mel-Frequency Cepstral Coefficient* (MFCC) untuk ekstraksi fitur dan *design pattern* dasar *Model-View-Controller* (MVC). Aplikasi ini menyediakan fitur untuk mengetahui sentimen atau emosi dari positif (atau emosi *happy*) dan sentimen negatif (atau emosi *sad*) file audio dengan berbasis web. Pada aplikasi ini terdapat :

1. Fitur *Preprocessing* yaitu menampilkan *Sample Distributions*, menganalisis sentimen dengan memilih secara random file lalu melabeli dengan sentimen *happy* atau *sad*, menampilkan *waveform*, *spectrogram with fundamental frequency*, *MFCC* lalu menampilkan *Average MFCCs Column*, *MFCC Comparison*, *MFCCs Distribution*.
2. Fitur *Training* yaitu menampilkan *MFCCs Distribution* seperti *train*, *validation test* dan diagramnya
3. Fitur *Testing* menampilkan *loss*, *accuracy*, *precision*, *recall*, *f1*, *confusion matrix (count)* dan *confusion matrix (ratio)*

#### **3.2 Saran**

1. Sampling rate hendaknya diperhatikan agar tidak terjadi perubahan kualitas audio
2. Perlakuan yang sama terhadap data harus diterapkan pada saat proses training/testing dan prediction sehingga fitur mfcc yang dihasilkan dapat sama

## DAFTAR PUSTAKA

- [1] RESPATI, R. G. J. (2021). Identifikasi Emosi Melalui Suara Menggunakan Support Vector Machine Dan Convolutional Neural Network. (Skripsi Sarjana, Universitas Islam Indonesia)
- [2] Zhao, J., Mao, X., & Chen, L. (2019). Speech emotion recognition using deep 1D & 2D CNN LSTM networks. *Biomedical Signal Processing and Control*, 47, 312-323.
- [3] Lecun, Y., Bengio, & Y., Hinton, G. (2015). Deep Learning. *Nature* 521, 436-444.
- [4] Salsabila. (2018). PENERAPAN DEEP LEARNING MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK UNTUK KLASIFIKASI CITRA WAYANG PUNAKAWAN. (Skripsi Sarjana, Universitas Islam Indonesia)
- [5] John, C, B. Arsitektur Neural Network. <https://wiragotama.github.io/resources/ebook/parts/JWGP-intro-to-ml-chap13-secured.pdf>
- [6] Justin. 2022. Identifikasi Audio Ancaman Menggunakan Metode Convolutional Neural Network. *Jurnal Sistem dan Teknologi Informasi*. Vol. 10, No. 4,
- [7] Putra, D. dan Adi, R., 2011. Verifikasi Biometrika Suara Menggunakan Metode MFCC dan DTW. *Biometrika*, Universitas Udayana, 2(1), hal.8–21.
- [8] Chamidy, T., 2016. Metode Mel Frequency Cepstral Coeffisients (MFCC) Pada klasifikasi Hidden Markov Model (HMM) Untuk Kata Arabic pada Penutur Indonesia. *Matics*, 8(1), hal.36–39. Available at: <http://ejournal.uinmalang.ac.id/index.php/saintek/article/view/3482>.
- [9] Kiran,U. 2021. Teknik MFCC untuk Pengenalan Pidado. Blogathon Ilmu data. Diakses pada tanggal 3 Juli 2023. Dari Analytics Vidhya
- [10] Ilahiyah,S & Nilogiri, A. IMPLEMENTASI DEEP LEARNING PADA IDENTIFIKASI JENIS TUMBUHAN BERDASARKAN CITRA DAUN MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK.