4. IMPLEMENTASI SISTEM

Bab ini menjelaskan mengenai implementasi dari sistem yang telah dirancang dan dibahas pada bab sebelumnya. Program untuk sistem yang dibuat disesuaikan dengan alur aplikasi dan alur sistem yang telah dirancang. Pada bab ini, akan dijelaskan proses implementasi yang dilakukan.

4.1 Implementasi Perangkat Lunak yang Digunakan

Implementasi sistem dibangun dengan menggunakan bahasa pemrograman Python dalam proses pembuatan model analisis sentimen. Model yang dibangun akan digunakan sebagai *backend* pada aplikasi *website* analisis sentimen. Untuk *frontend website*, akan digunakan HTML dan PHP dalam pembuatannya. Pembuatan kode menggunakan aplikasi *Visual Studio Code* dan dalam pembuatan model juga memanfaatkan *library* yang ada pada Python. Dalam implementasi sistem, *hardware* yang digunakan memiliki spesifikasi *processor* Intel i7-10510U, RAM sebesar 8 GB, dan kartu grafis NVIDIA GeForce MX350 dengan 2GB GDDR5 VRAM.

4.2 Implementasi Sistem

Implementasi sistem akan dirinci ke dalam beberapa segmen dan disesuaikan dengan alur yang sudah dirancang pada bab sebelumnya. Pemetaan segmen dalam implementasi sistem secara singkat dapat dilihat pada Tabel 4.1.

Tabel 4.1

Pemetaan Segmen Program

Sub-bab Desain	Proses	Keterangan	Segmen Program
3.3	Pengambilan Data Tweet	Pengambilan data menggunakan proses *scraping* dengan *library* TweetHarvest	4.1, 4.2
3.4.1	*Preprocessing* Data untuk metode *Lexicon-based*	Dilakukan proses *cleaning*, *case folding*, *filtering*	4.1, 4.3, 4.4

		(*username*, simbol, tanda baca, spasi berlebih), tokenisasi, transformasi, dan *stopword removal*	
	Preprocessing Data untuk deteksi sarkasme dengan metode SVM dan *Random Forest*	Dilakukan proses *cleaning*, filtering (hanya *username* dan spasi berlebih), tokenisasi, transformasi, dan *stopword removal*	4.1, 4.8, 4.9
3.4.2	*Feature Extraction*	Perhitungan jumlah kapital, inkonsisten kapital, perhitungan kata *slang*, perhitungan tanda baca, pengecekkan keberadaan emoji	4.1, 4.10, 4.11, 4.13
	Vektorisasi Data dengan *CountVectorizer*	Metode vektorisasi diterapkan untuk dibandingkan mana metode yang memiliki hasil terbaik	4.1, 4.12, 4.13, 4.14
	Vektorisasi Data dengan *TF-IDF*		4.1, 4.12, 4.13, 4.15
	Vektorisasi Data dengan *Word2Vec*		4.1, 4.12, 4.13, 4.16, 4.17
3.4.3	Evaluasi Sentimen dengan metode *Lexicon-based*	Pembuatan model analisis sentimen	4.1, 4.5, 4.6, 4.7
	Deteksi Sarkasme dengan metode SVM		4.1, 4.18, 4.19, 4.21, 4.22
	Deteksi Sarkasme dengan metode *Random Forest*		4.1, 4.20, 4.21, 4.22
3.5	Pengujian	Perhitungan dengan menggunakan *confusion matrix*	4.1, 4.23
		Percobaan implementasi model ke *dataset* yang	4.1, 4.24, 4.25, 4.26

		sudah diberi label	
3.6.2	Fitur Melihat Panduan pada Aplikasi *Website*	Pembuatan halaman menggunakan HTML dan PHP	4.1, 4.27, 4.28, 4.30, 4.31
3.6.3	Fitur Melakukan Analisis Sentimen pada Aplikasi *Website*	Pembuatan halaman menggunakan HTML dan PHP dan *backend* perhitungan menggunakan model yang sudah dibangun	4.1, 4.27, 4.29, 4.32, 4.33

4.3 Model

Pada sub-bab ini, dijelaskan program untuk pembuatan model analisis sentimen dengan deteksi sarkasme.

4.3.1 Instalasi *Library*

Sebelum memulai pembuatan model, diperlukan instalasi terhadap *library* yang digunakan. Segmen ini berguna untuk persiapan awal sebelum dilakukan pemrosesan data dan pembuatan model. Berikut penggunaan *library* yang digunakan.

Segmen Program 4.1 Instalasi *Library*

```
import subprocess
import re
import string
import pandas as pd
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import ast
import os
import math
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import MaxAbsScaler
from sklearn.preprocessing import MinMaxScaler
from scipy.sparse import hstack
import gensim
from gensim.models import Word2Vec
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split,
GridSearchCV
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score, classification_report
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
import joblib
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

- *Library* subprocess digunakan untuk menjalankan perintah pada *command prompt*.
- *Library* re digunakan untuk pemrosesan teks untuk menghilangkan karakter yang tidak diinginkan dalam *preprocessing* data.
- *Library* string digunakan untuk mendapatkan kumpulan karakter tanda baca.
- *Library* pandas digunakan untuk membaca file *excel* dari *dataset*.
- *Library* NLTK digunakan untuk kepentingan penghapusan *stopword* dan tokenisasi pada tahap *preprocessing* data.
- *Library* ast digunakan untuk mengevaluasi string pada kamus yang ada pada *preprocessing source*.
- *Library* os digunakan untuk melakukan operasi terkait sistem operasi, terutama untuk manipulasi *path file*.
- *Library* math digunakan untuk melakukan *ceiling* atau pembulatan ke atas dalam perhitungan *lexicon.*
- *Library* numpy digunakan dalam vektorisasi dengan metode Word2Vec dimana numpy digunakan untuk pembuatan vektor nol, penambahan vektor, dan pembagian skalar pada vektor.
- *Library* sklearn digunakan untuk vektorisasi *Bag-of-word* menggunakan *CountVectorizer* dan *TF-IDF* dan digunakan untuk implementasi metode untuk deteksi sarkasme yaitu menggunakan SVM dan *Random Forest*, serta *library* sklearn ini digunakan untuk menghitung nilai evaluasi matriks dan melakukan normalisasi pada fitur.
- *Library* scipy digunakan untuk menggabungkan beberapa matriks yang digunakan untuk menggabungkan ekstraksi fitur dan hasil vektorisasi.
- *Library* gensim digunakan untuk menggunakan model Word2Vec dalam vektorisasi data.
- *Library* imblearn digunakan untuk mengatasi *imbalance* atau ketidakseimbangan data pada *dataset*, dimana pada penelitian ini, kasusnya adalah data *Sarcasm* yang jauh lebih sedikit dibandingkan dengan data *NonSarcasm*, sehingga imblearn dimanfaatkan

untuk perlakuan *oversampling* dengan menggunakan SMOTE dan *undersampling* dengan menggunakan RandomUnderSampler pada *dataset*.

- *Library* joblib digunakan untuk menyimpan model yang sudah dilatih sebelumnya.
- *Library* Wordcloud digunakan untuk membuat wordcloud yang akan ditampilkan kepada user.
- *Library* matplotlib digunakan untuk memvisualisasikan hasil wordcloud.

4.3.2 Pengambilan Data Tweet

Pengambilan data *tweet* dilakukan dengan *scraping* menggunakan *library* TweetHarvest. Ketentuan dari penggunaan *library* ini adalah sudah memiliki Node.js pada *device* karena cara kerja TweetHarvest adalah membuka *chromium browser* dan masuk ke halaman pencarian Twitter untuk melakukan pencarian sesuai dengan parameter yang diberikan. *Library* ini akan mengekstrak *tweet* yang dihasilkan dan disimpan dalam sebuah file CSV. Pada program pengambilan data, dilakukan deklarasi variabel sebagai parameter yang akan digunakan untuk *search* yaitu variabel *keyword search* yang mengandung kata yang ingin dicari, *language* atau bahasa, dan rentang waktu *tweet* yang ingin dicari, selain itu juga dilakukan deklarasi variabel untuk nama *file*, limit *tweet* yang ingin diambil dan *twitter authentication token*. Token ini dapat diambil dengan *login* ke Twitter di *browser* dan mengekstrak *cookie* auth_token.

Segmen Program 4.2 *Scraping* Data Tweet

```
def run_command_on_cmd(command):
    try:
        # Menjalankan perintah di Command Prompt
        subprocess.run(command, shell=True, check=True)
    except subprocess.CalledProcessError as e:
        print(f"Error: {e}")
    except Exception as e:
        print(f"Unexpected error: {e}")

def main():
    # Deklarasi variabel
    filename = 'avoskinjanuari2023.csv'
    search_keyword = 'avoskin lang:id until:2024-01-31
since:2024-01-01' #isi dengan keyword
    limit = 1000
    twitter_auth_token = ' ' #isi dengan auth token twitter

    # Membuat perintah berdasarkan variabel yang sudah
dideklarasikan
```

```
    command = f'npx -y tweet-harvest@2.6.0 -o "{filename}" -s
"{search_keyword}" --tab "LATEST" -l {limit} --token
{twitter_auth_token}'

    # Menjalankan perintah di Command Prompt
    run_command_on_cmd(command)

if __name__ == "__main__":
    main()
```

Pada Segmen Program 4.2 terdapat 2 fungsi yaitu fungsi *run_command_on_cmd* dan fungsi *main*. Fungsi *run_command_on_cmd* digunakan untuk menjalankan perintah di *command prompt* dengan menggunakan *library subprocess*, perintah di *command prompt* perlu dijalankan karena TweetHarvest dijalankan melalui *command prompt*. Fungsi *main* merupakan fungsi utama untuk proses *scraping*, dimana fungsi ini berisi deklarasi variabel dan memanggil fungsi *run_command_on_cmd* untuk *run* TweetHarvest. Untuk mendapatkan data dengan jumlah yang cukup, proses *scraping* dilakukan *run* program berkali-kali karena Twitter sendiri memiliki pembatasan untuk setiap akun dalam melihat *tweet*. Pada penelitian ini, proses *scraping* dilakukan dengan beberapa auth_token yang berbeda untuk mempercepat waktu penelitian dan *scraping* dilakukan dengan rentang waktu satu bulan untuk setiap *keyword* untuk menghindari limit di tengah proses *scraping*. Setelah selesai, hasil *scraping* setiap bulan akan digabungkan menjadi satu *file dataset*. Pada penelitian ini, data hasil gabungan disimpan di file .xlsx dengan nama “Dataset2 Tweet Skincare 2023.xlsx”.

4.3.3 *Preprocessing* Data (untuk metode *Lexicon-based*)

Setelah mendapatkan *dataset* yang digunakan untuk melakukan pelatihan model, dilakukan *preprocessing data*. Seperti yang sudah dijelaskan pada bab sebelumnya, terdapat dua macam proses *preprocessing*, yaitu untuk metode *Lexicon-based dan* metode *machine learning*. Hal ini dikarenakan terdapat perlakuan yang berbeda untuk proses *preprocessing* yang dilakukan. Untuk *preprocessing* pertama yaitu *preprocessing* untuk metode *Lexicon-based*. Pada proses ini, sebelum data dimasukkan ke program *preprocessing* data, dilakukan penghapusan data yang kurang relevan pada *dataset*. Penghapusan data ini dilakukan sortir secara manual dari Microsoft Excel. Data yang dihapus adalah data yang merupakan iklan dan jual beli. Setelah disortir, *dataset* kemudian dimasukkan ke program untuk *preprocessing* berikutnya.

Segmen Program 4.3 *Preprocessing* Data untuk metode *Lexicon-based*

```
nltk.download('stopwords')

# Load Slang Words
file_1 = open("prepro_source/combined_slang_words.txt", "r")
content1 = file_1.read()
slang_words = ast.literal_eval(content1)
file_1.close()

# Load Colloquial Indonesian Lexicon
file_2 =
open("prepro_source/colloquial_indonesian_lexicon.txt", "r")
content2 = file_2.read()
colloquial_words = ast.literal_eval(content2)
file_2.close()

# Preprocessing teks
def preprocess_text_indonesia(text):
    # Mengubah teks menjadi huruf kecil
    text = text.lower() if isinstance(text, str) else ''  #
Menangani nilai NaN

    # Menghapus username
    tokens = text.split()
    tokens = [token for token in tokens if not
token.startswith('@')]
    text = ' '.join(tokens)

    # Menghapus spasi berlebih di awal teks setelah menghapus
username
    text = text.strip()

    # Menghapus karakter khusus, angka, tanda baca
    text = re.sub(r'\d+', '', text) # menghapus angka
    text = text.translate(str.maketrans('', '',
string.punctuation)) # menghapus tanda baca
    text = re.sub(r'\s+', ' ', text) # mengganti spasi berulang
dengan satu spasi

    # Tokenisasi teks
    tokens = word_tokenize(text)

    # Transformasi slang words
    for i in range(len(tokens)):
        if tokens[i] in slang_words:
            tokens[i] = slang_words[tokens[i]]

    # Transformasi colloquial words
    for i in range(len(tokens)):
        if tokens[i] in colloquial_words:
            tokens[i] = colloquial_words[tokens[i]]

    # Menghapus stop words (kata-kata umum yang tidak
```

```
berkontribusi pada makna)
    stop_words = set(stopwords.words('indonesian'))
    tokens = [word for word in tokens if word not in
stop_words]

    # Menggabungkan kembali token-token ke dalam bentuk string
    preprocessed_text = ' '.join(tokens)

    return preprocessed_text
```

Pada Segmen Program 4.3 terlihat bahwa sebelum masuk ke fungsi *preprocessing* dilakukan load *preprocessing source* yaitu *load* untuk *Slang Word* dan *Colloquial Word*. *Sources* ini digunakan untuk transformasi teks pada *preprocessing*. Selanjutnya, pada proses *preprocessing* dilakukan beberapa perlakuan terhadap data. Proses dimulai dengan *case-folding* atau mengubah teks menjadi seragam dengan huruf kecil, hal ini dilakukan karena *Lexicon* yang digunakan menggunakan *lowercase*. Kemudian dilakukan *filtering* yaitu dengan menghapus karakter yang tidak diperlukan, seperti *username* yang terkandung pada kalimat, spasi berlebih, karakter khusus, angka, tanda baca. Setelah *filtering*, dilakukan tokenisasi, yaitu proses memecah teks menjadi token. Selanjutnya, dilakukan transformasi kata dari token-token yang dihasilkan pada proses tokenisasi. Transformasi yang dilakukan adalah mengubah kata-kata yang ada pada *sources Slang-words* dan *Colloquial-words*. Transformasi dilakukan agar menaikkan kemungkinan untuk kata tersebut dapat ditemukan pada *Lexicon* yang digunakan untuk analisis sentimen. Setelah transformasi, dilakukan *stopword removal* untuk menghilangkan kata yang tidak berkontribusi terhadap makna. Proses *stopword removal* menggunakan *library* NLTK. Setelah semua proses perlakuan dilakukan, selanjutnya token-token tadi digabung kembali menjadi *string*.

Fungsi *preprocessing* diterapkan ke seluruh data *tweet* yang ada pada *dataset*. Sebelum memanggil fungsi *preprocessing*, *dataset* dipanggil dan fungsi *preprocessing* diterapkan pada data *tweet* pada *file*.

Segmen Program 4.4 Penerapan Fungsi *preprocessing* untuk Metode *Lexicon-based*

```
# Load dataset
fileFolder = "dataset/"
fileName = "Dataset2 Tweet Skincare 2023.xlsx" #ganti sesuai
nama folder dan nama file
fullFilePath = os.path.join(fileFolder, fileName)
df = pd.read_xlsx(fullFilePath)
```

```
# Apply ke dataset
df['Preprocessed Text'] =
df['full_text'].fillna('').apply(preprocess_text_indonesia)
```

Pemanggilan *dataset* dengan deklarasi *path* dari *dataset* yang sudah digabungkan dan disortir sebelumnya. Kemudian *library* pandas dipanggil untuk membaca *file* tersebut. File tersebut pada proses berikutnya akan dipanggil dengan *'df'*. Kemudian, dalam penerapan *preprocessing*, fungsi *preprocess_text_indonesia* yang sudah dibuat sebelumnya dipanggil untuk setiap data *tweet* yang berada pada kolom *'full_text'* pada file.

4.3.4 Evaluasi Sentimen (dengan metode *Lexicon-based*)

Proses analisis sentimen setelah *preprocessing* adalah evaluasi sentimen. Seperti yang sudah dijelaskan pada bab sebelumnya, terdapat dua kali evaluasi sentimen, yang pertama adalah dengan metode *Lexicon-based* dimana sentimen akan dicari dengan perhitungan *score* dari kalimat dan yang kedua adalah deteksi sarkasme dengan *machine learning*. Pada subbab ini akan dijelaskan implementasi sistem untuk evaluasi sentimen dengan metode *Lexicon-based*.

Segmen Program 4.5 *Load file* dan pembuatan fungsi untuk penanganan *Lexicon*

```
# Load keluhan kulit dari file
with open("prepro_source/masalahkulit.txt", "r") as file:
    keluhan_kulit = file.read().splitlines()

# Daftar kata-kata yang menunjukkan review baik
review_baik = ["baik", "bagus", "mantap", "sangat bagus",
"ampuh", "mantul"]

# Fungsi untuk menangani keluhan kulit
def handle_skin_complaints(text):
    for review_word in review_baik:
        if review_word in text:
            return True  # Kalimat mengandung review baik,
sehingga keluhan kulit tidak perlu ditangani
    for word in keluhan_kulit:
        if word in text:
            return False  # Kalimat tidak mengandung review
baik tetapi mengandung keluhan kulit, sehingga keluhan kulit
harus ditangani
    return True  # Kalimat tidak mengandung keluhan kulit,
tidak perlu ditangani
```

Pada bab 3, dijelaskan bahwa perhitungan *Lexicon* akan mengecualikan *score* untuk permasalahan kulit apabila terdapat kata yang termasuk ke dalam *review* baik. Pada Segmen Program 4.5 dilakukan *load* untuk *file* yang berisi *list* kata permasalahan kulit dan terdapat *array* review_baik yang berisi kata-kata yang termasuk kata *review* baik. Selanjutnya, terdapat fungsi untuk menangani keluhan kulit yaitu *handle_skin_complaints* dengan parameter teks dimana pada setiap teks akan dicek apakah terdapat kata dengan *review* baik, apabila terdapat maka keluhan kulit tidak perlu ditangani.

Selanjutnya, dilakukan perhitungan sentimen dengan kombinasi dua *Lexicon*, yaitu *InSet* dan *SentistrengthID*. Seperti yang sudah dijelaskan pada bab 3, bahwa penelitian ini mengkombinasikan kedua *Lexicon* guna untuk memperlengkap kata agar akurasi program dapat lebih tinggi.

Segmen Program 4.6 *Load Lexicon* dan mengkombinasikan *Lexicon*

```
# Inset Positive Lexicon
positive_lexicon_path = "lexicon/InSet/positive.tsv"
positive_lexicon = {}
with open(positive_lexicon_path, 'r', encoding='utf-8') as
file:
    next(file)  # skip header
    for line in file:
        word, weight = line.strip().split('\t')
        positive_lexicon[word] = int(weight)

# Inset Negative Lexicon
negative_lexicon_path = "lexicon/InSet/negative.tsv"
negative_lexicon = {}
with open(negative_lexicon_path, 'r', encoding='utf-8') as
file:
    next(file)  # skip header
    for line in file:
        word, weight = line.strip().split('\t')
        negative_lexicon[word] = int(weight)

# Additional lexicon dr Sentistrength
additional_lexicons = {}
sentistrength_lexicons =
["lexicon/sentistrength/sentiwords_id.txt",

"lexicon/sentistrength/idioms_id.txt",

"lexicon/sentistrength/boosterwords_id.txt"]
for lexicon_path in sentistrength_lexicons:
    with open(lexicon_path, 'r', encoding='utf-8') as file:
        for line in file:
```

```
            word, weight = line.strip().split(':')
            additional_lexicons[word] = int(weight)

# Combine lexicons
combined_lexicon = {}

for word, weight in positive_lexicon.items():
    if word in additional_lexicons:
        combined_lexicon[word] = math.ceil((weight +
additional_lexicons[word]) / 2) #opsi lain: pake round
    else:
        combined_lexicon[word] = weight

for word, weight in negative_lexicon.items():
    if word in additional_lexicons:
        combined_lexicon[word] = math.ceil((weight +
additional_lexicons[word]) / 2)
    else:
        combined_lexicon[word] = weight

for word, weight in additional_lexicons.items():
    if word not in positive_lexicon and word not in
negative_lexicon:
        combined_lexicon[word] = weight
```

Kedua *lexicon* disimpan dan selanjutnya dikombinasikan. Dalam kombinasi, apabila ditemukan kata yang sama pada kedua *lexicon*, bobot untuk kata tersebut di rata-ratakan kemudian hasilnya dibulatkan ke atas. Perlakuan ini didasarkan pada *paper* dari pembuatan *lexicon* tersebut. Cara pembuatan kedua *lexicon* sama, yaitu dengan menggunakan angka bobot yang diberikan kedua ahli dan untuk hasil akhir bobot pada kata yang ada pada *lexicon* adalah rata-rata dari angka yang diberikan kedua ahli dan dilakukan pembulatan ke atas. Sehingga, pada penelitian ini juga menggunakan cara yang sama dalam kombinasi *lexicon*. Pada *code* yang telah dibuat, *lexicon* disimpan ke dalam struktur data dan untuk kombinasi *lexicon* menggunakan pengulangan dan dilakukan perhitungan dengan hasil akhir perhitungan dibulatkan menggunakan *library math.ceil* dimana *ceil* ini akan membulatkan ke atas. Hasil akhir dari kombinasi *lexicon* disimpan pada *combined_lexicon*.

Segmen Program 4.7 Analisis Sentimen dengan metode *Lexicon-based*

```
# Hitung score sentiment
def calculate_sentiment(sentence):
    words = sentence.split()
    sentiment_score = 0
    for word in words:
```

```
        if word in combined_lexicon:
            # Penanganan keluhan kulit
            if not handle_skin_complaints(word):
                continue
            sentiment_score += combined_lexicon[word]
            # print(word, combined_lexicon[word])
    return sentiment_score

# Labeling sentiment
def label_sentiment(score):
    if score > 0:
        return 'Positive'
    elif score < 0:
        return 'Negative'
    else:
        return 'Neutral'

# Hitung score dari Lexicon
df['Sentiment Score'] = df['Preprocessed
Text'].apply(calculate_sentiment)
# Label sentimen dari score
df['Sentiment Label'] = df['Sentiment
Score'].apply(label_sentiment)

# Output path
output_folder = os.path.dirname(fullFilePath)
output_filename = os.path.splitext(fileName)[0] +
"_sentiment_labeled.xlsx"
output_path = os.path.join(output_folder, output_filename)

# Save hasil
df.to_excel(output_path, index=False)

print("Analisis Sentimen berhasil. Hasil Label disimpan di:",
output_path)
```

Setelah melakukan kombinasi pada *lexicon*, dilakukan perhitungan *score* untuk menentukan sentimen pada data. Perhitungan *score* menggunakan fungsi *calculate_sentiment* dimana pada fungsi ini dilakukan *split* pada kalimat dan dihitung *score* nya. Perhitungan *score* melibatkan fungsi *handle_skin_complaints* untuk mengecualikan perhitungan *score* untuk kata yang merupakan keluhan kulit apabila terdapat kata yang termasuk *review* baik. *Score* dari setiap kata pada kalimat dijumlahkan dengan acuan *score* dari *combined_lexicon*. Setelah didapatkan *score* untuk seluruh data pada *dataset*, dilakukan pemberian label pada data tersebut. Pemberian label didasarkan pada *paper* dari *lexicon*, dimana apabila *score* > 0 maka didapatkan label sentimen Positif, *score* = 0 maka didapatkan label sentimen *netral*, dan apabila *score < 0* didapatkan label sentimen negatif. Hasil perhitungan *score* disimpan pada kolom "*Sentiment*

Score" dan hasil label disimpan pada kolom "*Sentiment Label*". Untuk hasil akhir file akan disimpan dengan nama dan *path* sesuai dengan variabel *output_filename* dan *output_path*.

4.3.5 *Preprocessing* Data (untuk SVM dan RF)

Seperti yang sudah dijelaskan pada bab 3, data-data dengan hasil sentimen Positif dari metode *Lexicon-based* dipisahkan di file baru dengan nama "*dataset* sarkas" untuk dijadikan *dataset* untuk membangun model deteksi sarkasme dengan metode SVM dan RF. *Dataset* kemudian digabungkan dengan *dataset* tambahan dari sumber lain. Proses *preprocessing* data kali ini akan diterapkan pada *dataset* sarkas dan dilakukan pada data *tweet* awal atau data *tweet* asli (sebelum *preprocessing* pertama). *Preprocessing* mirip dengan *preprocessing* sebelumnya, namun terdapat beberapa perlakuan yang tidak dilakukan.

Segmen Program 4.8 *Preprocessing* Data untuk metode SVM dan RF

```
nltk.download('stopwords')

# Load Slang Words
file_1 = open("prepro_source/combined_slang_words.txt", "r")
content1 = file_1.read()
slang_words = ast.literal_eval(content1)
file_1.close()

# Load Colloquial Indonesian Lexicon
file_2 =
open("prepro_source/colloquial_indonesian_lexicon.txt", "r")
content2 = file_2.read()
colloquial_words = ast.literal_eval(content2)
file_2.close()

# Preprocessing text
def preprocess_text_indonesia(text):
    # Menghapus username
    tokens = text.split()
    tokens = [token for token in tokens if not
token.startswith('@')]
    text = ' '.join(tokens)

    # Menghapus spasi berlebih di awal teks setelah menghapus
username
    text = text.strip()

    # Mengganti spasi berulang dengan satu spasi
    text = re.sub(r'\s+', ' ', text)

    # Tokenisasi
```

```
    tokens = word_tokenize(text)

    # Transform slangword, colloquial word
    for i in range(len(tokens)):
        if tokens[i] in slang_words:
            tokens[i] = slang_words[tokens[i]]

    for i in range(len(tokens)):
        if tokens[i] in colloquial_words:
            tokens[i] = colloquial_words[tokens[i]]

    # Remove stopwords
    stop_words = set(stopwords.words('indonesian'))
    tokens = [word for word in tokens if word not in
stop_words]

    preprocessed_text = ' '.join(tokens)

    return preprocessed_text
```

Perlakuan yang dilakukan pada proses *preprocessing* ini adalah *load* data *slang-word* dan *colloquial-word* untuk proses transformasi. Kemudian, dalam fungsi *preprocessing* dilakukan penghapusan *username*, spasi berlebih dan dilakukan tokenisasi, transformasi *slang-word* dan *colloquial-word*, penghapusan *stopword* dan penggabungan kembali token menjadi kalimat. Karakter-karakter unik tidak dihapus dan tidak dilakukan *case-folding* karena berpengaruh terhadap deteksi sarkasme dan menjadi fitur dalam melatih model *machine learning*. Penggunaan nama fungsi yang sama seperti nama fungsi pada *preprocessing* sebelumnya karena fungsi berada pada *file* yang berbeda.

Segmen Program 4.9 Penerapan Fungsi *preprocessing* untuk Metode SVM dan RF

```
# Load dataset
fileFolder = "dataset/"
fileName = "dataset sarkas.xlsx"
fullFilePath = os.path.join(fileFolder, fileName)
df = pd.read_excel(fullFilePath)

df['Preprocessed Sarcasm'] =
df['full_text'].fillna('').apply(preprocess_text_indonesia)
```

Implementasi fungsi dari *preprocessing* seperti pada Segmen Program 4.9. Dimana akan ada proses *load dataset* sarkas dan fungsi *preprocessing* diimplementasikan ke kolom *"full_text"* pada *dataset*, yaitu kolom untuk tweet asli.

4.3.6 Vektorisasi Data

Proses vektorisasi dilakukan sebelum data dimasukkan ke dalam model *machine learning*. Dalam merubah data menjadi vektor, dilakukan *feature extraction* dengan ekstraksi fitur-fitur yang dapat berpengaruh terhadap pendeteksian sarkasme. Selain itu, pada penelitian ini juga dilakukan 3 macam metode vektorisasi untuk dibandingkan hasilnya dan dilihat metode manakah yang memberikan hasil yang lebih baik. Metode yang digunakan antara lain CountVectorizer, TF-IDF (menggunakan *library* sklearn) dan Word2Vec (menggunakan *library* gensim). Pada pembuatan model, akan dilakukan perbandingan vektorisasi dengan hanya ekstraksi fitur, ekstraksi fitur ditambah dengan setiap 3 macam metode vektorisasi, dan vektorisasi hanya dengan 3 macam metode dan tanpa ekstraksi fitur.

Segmen Program 4.10 Ekstraksi Fitur

```
# Fungsi untuk menghitung jumlah huruf kapital dalam sebuah
string
def count_capital_letters(text):
    # Menghitung jumlah huruf kapital dalam string
    capital_count = sum(1 for char in text if char.isupper())
    total_letters = len(text)
    return capital_count / total_letters

# Fungsi untuk mendeteksi apakah terdapat kombinasi huruf besar
dan kecil dalam sebuah string
def detect_inconsistent_case(text):
    inconsistent_count = 0
    for i in range(1, len(text) - 1):  # Mulai dari indeks 1
utk huruf pertama
        if text[i].isupper() and text[i + 1].islower() and
text[i - 1].islower():
            inconsistent_count += 1
    # Return 1 kalau > 1 utk menghindari camelCase terdetect
    return 1 if inconsistent_count > 1 else 0

# Fungsi untuk menghitung jumlah tanda baca dalam sebuah string
def count_punctuation(text):
    punctuation_count = sum(1 for char in text if char in
string.punctuation)
    return punctuation_count

# Fungsi untuk memeriksa apakah teks mengandung emoji tertentu
def check_emoji(text, emoji):
```

```
    return 1 if emoji in text else 0

# Fungsi untuk menghitung jumlah ekspresi slang dalam sebuah
teks
def count_slang(text, slang_words):
    count = 0
    for word in slang_words:
        count += text.lower().count(word)
    return count
# Daftar kata-kata slang yang umum
slangs = ['lol', 'rofl', 'lmao', 'LOL', 'ROFL', 'LMAO']

# Menambahkan kolom baru "CapitalNumber" dengan jumlah huruf
kapital dari kolom "Preprocessed Sarcasm"
df['CapitalNumber'] = df['Preprocessed
Sarcasm'].apply(count_capital_letters)
# Menambahkan kolom baru "InconsistentCap" untuk mendeteksi
huruf dengan case yang tidak konsisten
df['InconsistentCap'] = df['Preprocessed
Sarcasm'].apply(detect_inconsistent_case)
# Menambahkan kolom baru "PunctuationCount" dengan jumlah tanda
baca dari kolom "Preprocessed Sarcasm"
df['PunctuationCount'] = df['Preprocessed
Sarcasm'].apply(count_punctuation)
# Menambahkan kolom baru "AmountOfSlang" dengan jumlah ekspresi
slang dalam teks
df['AmountOfSlang'] = df['Preprocessed Sarcasm'].apply(lambda
x: count_slang(x, slangs))

# Memuat data emoji dari file CSV, skip header
emoji_df = pd.read_excel("dataset/filteredemoji.xlsx")

# Menambahkan kolom baru untuk setiap emoji
for emoji in emoji_df['emoji']:
    df[emoji] = df['full_text'].apply(lambda x: check_emoji(x,
emoji))

# Menyimpan DataFrame dengan kolom baru ke file Excel baru
output_file = "dataset/dataset_sarkas_with_feature.xlsx"
df.to_excel(output_file, index=False)
print("Data telah disimpan ke file:", output_file)
```

Setelah dilakukan *preprocessing* terhadap data, ekstraksi fitur dilakukan dengan fitur-fitur yang diambil adalah menghitung jumlah huruf kapital, pengecekan inkonsisten kapital, perhitungan jumlah tanda baca, perhitungan kata *slang* sarkas, dan pengecekan ketersediaan emoji. Setiap fitur disimpan pada kolom baru dan disimpan pada file *excel* baru. Kemudian, untuk penggunaan metode vektorisasi, digunakan *library* vektorisasi yang sudah tersedia pada python.

Segmen Program 4.11 Deklarasi nilai X dan Y serta pembagian data (hanya fitur)

```
# Define features and target
X = df[['CapitalNumber', 'InconsistentCap', 'PunctuationCount',
'AmountOfSlang'] + list(emoji_df['emoji'])]
y = df['Sarcasm Label']

# Normalisasi nilai X
scaler = MinMaxScaler()
X_normalized = scaler.fit_transform(X)

# Split dataset
X_train, X_test, y_train, y_test =
train_test_split(X_normalized, y, test_size=0.2,
random_state=42, stratify=y)
```

Segmen Program 4.12 Deklarasi nilai X dan Y serta pembagian data (hanya metode vektor)

```
# Split dataset
X = df['Preprocessed Sarcasm']
y = df['Sarcasm Label']

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

Segmen Program 4.13 Deklarasi nilai X dan Y serta pembagian data (kombinasi)

```
# Define features and target
X_numerical = df[['CapitalNumber', 'InconsistentCap',
'PunctuationCount', 'AmountOfSlang'] + list(emoji_df['emoji'])]

# Combine text features into one column
text_features = df['Preprocessed Sarcasm']

# Vectorize text features using CountVectorizer
vectorizer = # isi dengan metode vektorisasi seperti segmen
program berikutnya
X_text = vectorizer.fit_transform(text_features)

# Kombinasi numerical and text features
X_combined = hstack((X_numerical.values, X_text))

# Define target
y = df['Sarcasm Label']

# Normalize text features using MaxAbsScaler
scaler = MaxAbsScaler()
X_normalized = scaler.fit_transform(X_combined)
```

```
# Split dataset
X_train, X_test, y_train, y_test =
train_test_split(X_normalized, y, test_size=0.2,
random_state=42, stratify=y)
```

Sebelum dilakukan vektorisasi, perlu untuk mendapatkan nilai X dan Y dari data. Pada segmen program 4.11 hingga 4.13 dilakukan deklarasi nilai X dan Y dan dilakukan *train_test_split* yaitu pembagian data untuk menjadi data latih dan data uji. Parameter dalam pembagian data adalah 20% untuk data uji, *random_state* untuk mempertahankan pembagian data agar konsisten dan *stratify* untuk memastikan proporsi kelas yang sama di setiap subset data, sehingga meminimalkan risiko terjadinya bias pada pembagian data untuk pelatihan dan pengujian model. Untuk fitur, dilakukan normalisasi pada fitur agar setiap data memiliki rentang yang sama dan dapat mencegah salah satu fitur menjadi dominan.

Segmen Program 4.14 Vektorisasi dengan CountVectorizer

```
# Vectorization using CountVectorizer
vectorizer = CountVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

Vektorisasi menggunakan CountVectorizer dari *library* sklearn dengan memanggil metode fit_transform pada data latih dan data uji untuk membuat model vektorisasi dan mentransformasikan data pada nilai X menjadi vektor.

Segmen Program 4.15 Vektorisasi dengan TF-IDF

```
# Vectorization using TF-IDF
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf_vec = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf_vec = tfidf_vectorizer.transform(X_test)
```

Kemudian, vektorisasi dengan TF-IDF juga melakukan hal yang sama dan juga menggunakan *library* sklearn.

Untuk metode Word2Vec, dilakukan pelatihan terlebih dahulu dengan *corpus* Wikipedia Indonesia. Hal ini dikarenakan, Word2Vec yang tersedia dari gensim merupakan Word2Vec Bahasa Inggris. Pelatihan dilakukan dengan referensi pelatihan sebelumnya dari GitHub akun deryrahman (Ahaddienata, 2019) dan pelatihan Word2Vec penelitian ini menggunakan data

latih Wikipedia terbaru yang baru diunduh dari halaman Wikimedia (https://dumps.wikimedia.org/idwiki/latest/).

Segmen Program 4.16 Pelatihan model vektorisasi Word2Vec Bahasa Indonesia

```
import io
import time
from datetime import timedelta
import gensim
import multiprocessing
from gensim.models import word2vec

if __name__ == '__main__':

    start_time = time.time()
    print('Streaming wiki...')
    id_wiki = gensim.corpora.WikiCorpus(
        'idwiki-latest-pages-articles.xml.bz2', dictionary={},
lower=True
    )

    article_count = 0
    with io.open('idwiki_new_lower.txt', 'w', encoding='utf-8')
as wiki_txt:
        for text in id_wiki.get_texts():

            wiki_txt.write(" ".join(text) + '\n')
            article_count += 1

            if article_count % 10000 == 0:
                print('{} articles
processed'.format(article_count))
        print('total: {} articles'.format(article_count))

    finish_time = time.time()
    print('Elapsed time:
{}'.format(timedelta(seconds=finish_time-start_time)))

    start_time = time.time()
    print('Training Word2Vec Model...')
    sentences = word2vec.LineSentence('idwiki_new_lower.txt')
    id_w2v = word2vec.Word2Vec(sentences, vector_size=200,
workers=multiprocessing.cpu_count()-1)
    id_w2v.save('model/idwiki_word2vec_200_new_lower.model')
    finish_time = time.time()

    print('Finished. Elapsed time:
{}'.format(timedelta(seconds=finish_time-start_time)))
```

Pelatihan Word2Vec ini menghasilkan model dengan nama file "idwiki_word2vec_200_new_lower.model". Hasil dari model nantinya dapat dipanggil untuk diimplementasikan terhadap data yang akan di vektorisasi.

Segmen Program 4.17 Implementasi model vektorisasi Word2Vec

```
# Vectorization W2V Function
def vectorize_data_w2v(data, word2vec_model):
    vectors = []
    for sentence in data:
        vector = np.zeros(word2vec_model.vector_size)
        count = 0
        for word in sentence.split():
            if word in word2vec_model.wv:
                vector += word2vec_model.wv[word]
                count += 1
        if count != 0:
            vector /= count
        vectors.append(vector)
    return np.array(vectors)

# Load model
word2vec_model =
Word2Vec.load('model/idwiki_word2vec_200_new_lower.model')
# Vektorisasi
X_train_w2v_vec = vectorize_data_w2v(X_train, word2vec_model)
X_test_w2v_vec = vectorize_data_w2v(X_test, word2vec_model)
```

Penerapan model vektorisasi Word2Vec dituliskan pada Segmen Program 4.17. Fungsi *vectorize_data_w2v* bertujuan untuk mengkonversikan data teks menjadi vektor menggunakan model yang sudah dilatih dengan parameter yang diperlukan adalah data yang akan di vektorisasi dan model vektorisasi Word2Vec. Pada setiap iterasi, fungsi tersebut melakukan iterasi untuk setiap kalimat dalam data dan untuk setiap kata dalam kalimat, dicari vektor representasi kata tersebut dalam model Word2Vec. Jika kata tersebut ditemukan dalam model, maka vektor representasi kata tersebut ditambahkan ke vektor untuk kalimat dan jumlah kata yang ditemukan diperbaharui. Apabila iterasi sudah selesai untuk seluruh kalimat, vektor dinormalisasikan dengan membaginya dengan jumlah kata yang ditemukan. Proses ini dilakukan untuk setiap kalimat yang ada pada data dan hasilnya adalah *array numpy* yang berisi vektor representasi untuk setiap kalimat. Kemudian dalam penggunaannya, pertama dilakukan *load* dari model Word2Vec yang sudah dilatih, kemudian dilakukan vektorisasi terhadap nilai X dengan memanggil fungsi *vectorize_data_w2v* dengan parameter nilai X dan model Word2Vec.

4.3.7 Evaluasi Sentimen (deteksi sarkasme dengan SVM dan RF)

Proses analisis sentimen pada penelitian ini melibatkan deteksi sarkasme. Seperti yang sudah dijelaskan pada bab sebelumnya, bahwa dalam mendeteksi sarkasme digunakan metode *Support Vector Machine* (SVM) dan *Random Forest* (RF). Pada proses pembuatan model, dilakukan percobaan untuk data *no sampling*, *oversampling*, dan *undersampling* terhadap *dataset* dan diterapkan untuk seluruh model vektorisasi untuk dibandingkan kinerjanya. Perbandingan vektor meliputi vektorisasi yang digunakan untuk pelatihan model, dimana akan dibandingkan menggunakan hanya fitur, fitur dikombinasikan dengan metode vektorisasi dan hanya metode vektorisasi. Selain itu, untuk pemilihan kernel untuk SVM juga dibandingkan untuk kernel linear, poly, dan rbf.

Segmen Program 4.18 Perbandingan kernel SVM

```
# SVM model
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)

svm_model = SVC(kernel='poly')
svm_model.fit(X_train, y_train)

svm_model = SVC(kernel='rbf')
svm_model.fit(X_train, y_train)
```

Setiap kernel akan dicoba untuk perhitungan terhadap data dengan seluruh macam vektorisasi dan untuk setiap metode *sampling*. Pencarian kernel terbaik akan dihitung dengan membandingkan hasil *confusion matrix* yang dihasilkan oleh model.

Segmen Program 4.19 Deteksi Sarkasme dengan SVM hanya metode vektorisasi (No Sampling)

```
# SVM model No Sampling

# Model
# CountVectorizer
svm_model = SVC(kernel='linear')
svm_model.fit(X_train_vec, y_train)

model_filename = 'svm_model_cv_nosamp.pkl'
joblib.dump(svm_model, model_filename)

# TF-IDF
svm_model_tfidf = SVC(kernel='linear')
svm_model_tfidf.fit(X_train_tfidf_vec, y_train)
```

```
model_filename = 'svm_model_tfidf_nosamp.pkl'
joblib.dump(svm_model_tfidf, model_filename)

# Word2Vec
svm_model_w2v = SVC(kernel='linear')
svm_model_w2v.fit(X_train_w2v_vec, y_train)

model_filename = 'svm2_model_w2v_nosamp.pkl'
joblib.dump(svm_model_w2v, model_filename)

# Fitur
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)
joblib.dump(svm_model, 'n_fituronly_svmlin_model.pkl')

# Prediction

# CountVectorizer
y_pred = svm_model.predict(X_test_vec)

# TF-IDF
y_pred_tfidf = svm_model_tfidf.predict(X_test_tfidf_vec)

# Word2Vec
y_pred_w2v = svm_model_w2v.predict(X_test_w2v_vec)

# Fitur
y_pred_fitur = svm_model.predict(X_test)
```

Model SVM dengan setiap kernel digunakan untuk melatih setiap data hasil vektorisasi dengan model yang sudah dijelaskan sebelumnya (segmen program menggunakan contoh untuk kernel linear). Proses pelatihan menggunakan data latih yang telah diproses. Setelah pelatihan, model-model tersebut disimpan dalam file yang sesuai dengan masing-masing metode vektorisasi menggunakan *library* joblib. Kemudian, prediksi dilakukan pada data uji menggunakan masing-masing model yang telah dilatih, menghasilkan label prediksi untuk setiap jenis vektorisasi. Prediksi dilakukan untuk setiap hasil vektorisasi, untuk prediksi dengan mengganti variabel yang akan diprediksi sesuai nama variabel X_test yang sudah dideklarasikan saat vektorisasi dan pembagian data latih dan data uji.

Segmen Program 4.20 Deteksi Sarkasme dengan RF (No Sampling)

```
# Random Forest model with grid search
param_grid = {
    'n_estimators': [50, 100, 200, 300],
```

```
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'criterion': ['entropy']
}

rf = RandomForestClassifier()
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid,
cv=3, n_jobs=-1, verbose=2)

# No Sampling

# CountVectorizer
grid_search.fit(X_train_vec, y_train)
best_rf_count = grid_search.best_estimator_
y_pred_count_no_sampling = best_rf_count.predict(X_test_vec)
joblib.dump(best_rf_count, "rf2_cv_nosamp.pkl")

# TF-IDF
grid_search.fit(X_train_tfidf_vec, y_train)
best_rf_tfidf = grid_search.best_estimator_
y_pred_tfidf_no_sampling =
best_rf_tfidf.predict(X_test_tfidf_vec)

# Word2Vec
grid_search.fit(X_train_vec, y_train)
best_rf_w2v = grid_search.best_estimator_
y_pred_no_sampling = best_rf_w2v.predict(X_test_vec)
joblib.dump(best_rf_w2v, "rf2_w2v_nosamp.pkl")

# Fitur
grid_search.fit(X_train, y_train)
best_rf_classifier = grid_search.best_estimator_
joblib.dump(best_rf_classifier, 'n_rf_fitur_model.pkl')
# Prediction
y_pred = best_rf_classifier.predict(X_test)
```

Untuk pembuatan model dengan metode Random Forest, dilakukan dengan menggunakan *library* yang disediakan oleh sklearn dengan pencarian parameter menggunakan GridSearchCV. Proses ini berjalan paralel (n_jobs=-1) dan menampilkan informasi proses secara rinci (verbose=2). Sama seperti metode SVM, proses pelatihan juga menggunakan data latih yang telah diproses. Setelah pelatihan, model-model tersebut disimpan dalam file yang sesuai dengan masing-masing metode vektorisasi menggunakan library joblib. Kemudian, untuk pelatihan model juga sama dengan cara mengganti variabel X dan Y sesuai dengan variabel yang sudah dideklarasikan saat vektorisasi dan pembagian data uji dan data latih.

Segmen Program 4.21 Oversampling Dataset

```
# Oversampling dengan SMOTE
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled =
smote.fit_resample(X_train_vec, y_train)
```

Untuk *oversampling dataset*, menggunakan SMOTE dari *library* imblearn dimana *oversampling* ini akan menambah data yang minoritas, pada penelitian ini adalah data dengan label "Sarkasme". *Oversampling* akan menambah data minoritas menjadi sebanyak data mayoritas. Kemudian untuk pelatihan model dan prediksi menggunakan *code* yang sama seperti *code* untuk *no-sampling* dengan metode SVM dan RF, namun menggunakan variabel nilai X dan Y dari hasil *oversampling* yaitu *X_train_resampled* dan *y_train_resampled*.

Segmen Program 4.22 Undersampling Dataset

```
# Undersampling dengan mengurangi jumlah sampel dari kelas
mayoritas
rus = RandomUnderSampler(sampling_strategy=0.3,
random_state=42)
X_train_resampled, y_train_resampled =
rus.fit_resample(X_train_vec, y_train)
```

Untuk *undersampling* dataset, menggunakan RandomUnderSampler dari *library* imblearn dimana *undersampling* akan mengurangi data yang merupakan data mayoritas, pada penelitian ini adalah data dengan label "NonSarcasm". *Undersampling* akan mengurangi data mayoritas menjadi sebanyak data minoritas. Pelatihan model juga mirip dengan *code* pada *no-sampling* dengan metode SVM dan RF, namun menggunakan variabel nilai X dan Y dari hasil *undersampling* yaitu *X_train_resampled* dan *y_train_resampled*.

4.3.8 Pengujian

Setelah dilakukan prediksi dengan model, dilakukan pengujian untuk evaluasi terhadap model yang sudah dilatih. Evaluasi menggunakan *confusion matrix* dan juga mencoba menerapkan model dengan *dataset* baru yang berisi teks yang memiliki label sarkasme hasil *labeling* secara manual. Untuk penerapan model pada *dataset* baru akan menghitung kecocokan hasil prediksi model dengan kolom *self-labeled* untuk melihat seberapa tepat model dalam memprediksi.

Segmen Program 4.23 Evaluasi model SVM No Sampling

```
# Evaluation for SVM model No Sampling
# CountVectorizer
accuracy_no_sampling = accuracy_score(y_test, y_pred)
precision_no_sampling = precision_score(y_test, y_pred,
pos_label='Sarcasm')
recall_no_sampling = recall_score(y_test, y_pred,
pos_label='Sarcasm')
f1_no_sampling = f1_score(y_test, y_pred, pos_label='Sarcasm')
conf_matrix_no_sampling = confusion_matrix(y_test, y_pred)

# TF-IDF
accuracy_no_sampling_tfidf = accuracy_score(y_test,
y_pred_tfidf)
precision_no_sampling_tfidf = precision_score(y_test,
y_pred_tfidf, pos_label='Sarcasm')
recall_no_sampling_tfidf = recall_score(y_test, y_pred_tfidf,
pos_label='Sarcasm')
f1_no_sampling_tfidf = f1_score(y_test, y_pred_tfidf,
pos_label='Sarcasm')
conf_matrix_no_sampling_tfidf = confusion_matrix(y_test,
y_pred_tfidf)

# W2V
accuracy_no_sampling_w2v = accuracy_score(y_test, y_pred_w2v)
precision_no_sampling_w2v = precision_score(y_test, y_pred_w2v,
pos_label='Sarcasm')
recall_no_sampling_w2v = recall_score(y_test, y_pred_w2v,
pos_label='Sarcasm')
f1_no_sampling_w2v = f1_score(y_test, y_pred_w2v,
pos_label='Sarcasm')
conf_matrix_no_sampling_w2v = confusion_matrix(y_test,
y_pred_w2v)

# Print
print("Tanpa oversampling dan undersampling (Test):")
print(f"Accuracy: CountVectorizer = {accuracy_no_sampling},
TF-IDF = {accuracy_no_sampling_tfidf}, W2V =
{accuracy_no_sampling_w2v}")
```

```
print(f"Precision: CountVectorizer = {precision_no_sampling},
TF-IDF = {precision_no_sampling_tfidf}, W2V =
{precision_no_sampling_w2v}")
print(f"Recall: CountVectorizer = {recall_no_sampling}, TF-IDF
= {recall_no_sampling_tfidf}, W2V = {recall_no_sampling_w2v}")
print(f"F1 Score: CountVectorizer = {f1_no_sampling}, TF-IDF =
{f1_no_sampling_tfidf}, W2V = {f1_no_sampling_w2v}")
print(f"Confusion Matrix:\n CountVectorizer =  \n
{conf_matrix_no_sampling} \n TF-IDF = \n
{conf_matrix_no_sampling_tfidf} \n W2V = \n
{conf_matrix_no_sampling_w2v}")
```

Evaluasi performa dilakukan dengan menggunakan *library sklearn* untuk mendapatkan hasil perhitungan akurasi, presisi, *recall*, dan nilai *f1-score* untuk setiap jenis vektorisasi. Hasil evaluasi kemudian dimunculkan dengan *print* untuk dapat dibandingkan performanya. Untuk *dataset* dengan *oversampling* dan *undersampling* juga seperti *code* untuk *no-sampling*, namun variabel X dan Y diganti sesuai dengan variabel hasil *oversampling* dan *undersampling*. Kode yang sama juga diterapkan untuk percobaan vektorisasi dengan mengganti variabel sesuai variabel yang sudah dideklarasikan sebelumnya.

Segmen Program 4.24 Perhitungan kecocokan hasil prediksi model

```
# Load test dataset
test_data = pd.read_excel("dataset/buat test.xlsx")

# Load vectorizers
vectorizer_svmcv = joblib.load("vectorizernew_svmcv.pkl")

# Function to preprocess text
def preprocess_text(text):
   # isi code sama seperti code preprocessing untuk pembuatan
model

# Function to calculate kecocokan prediksi model dengan
self-labeled
def hitung_cocok(predictions, actual):
    return (predictions == actual).mean() * 100

# Preprocess test data
test_data['Preprocessed Text'] =
test_data['full_text'].fillna('').apply(preprocess_text)

# Predict with Random Forest models
def predict_rf(model_path, vectorizer, test_data):
    model = joblib.load(model_path)
    X_test_vec = vectorizer.transform(test_data['Preprocessed
```

```
Text'])
    predictions = model.predict(X_test_vec)
    return predictions

# Predict with SVM models
def predict_svm(model_path, vectorizer, test_data):
    model = joblib.load(model_path)
    X_test_vec = vectorizer.transform(test_data['Preprocessed
Text'])
    predictions = model.predict(X_test_vec)
    return predictions

# Vectorization W2V
def vectorize_data_w2v(data, word2vec_model):
    #code sama seperti code vektorisasi word2vec

word2vec_model =
Word2Vec.load('model/idwiki_word2vec_200_new_lower.model')

test_data['Word2Vec Vectors'] = test_data['Preprocessed
Text'].apply(lambda x: vectorize_data_w2v(word_tokenize(x),
word2vec_model))

# Predict with Word2Vec models
def predict_w2v(model_path, test_data):
    model = joblib.load(model_path)
    X_test = np.vstack(test_data['Word2Vec Vectors'].values)
    predictions = model.predict(X_test)
    return predictions

# Predict with SVM CountVectorizer models
svm_cv_nosamp_pred = predict_svm("svmnew_cv_nosamp.pkl",
vectorizer_svmcv, test_data)

# Predict with SVM Word2Vec models
svm_w2v_nosamp_pred = predict_w2v("svmnew_w2v_nosamp.pkl",
test_data)

# Add predictions to test_data
test_data['svm_cv_nosamp_pred'] = svm_cv_nosamp_pred
test_data['svm_w2v_nosamp_pred'] = svm_w2v_nosamp_pred

# Save predictions to excel
test_data.to_excel("hasil_prediksi_models.xlsx", index=False)

self_labeled = test_data['self labeled']

# Calculate kecocokan
svm_cv_nosamp_accuracy =
hitung_cocok(test_data['svm_cv_nosamp_pred'], self_labeled)
svm_w2v_nosamp_accuracy =
hitung_cocok(test_data['svm_w2v_nosamp_pred'], self_labeled)
```

```
print("Accuracy for each model:")
print(f"SVM CountVectorizer No Sampling Accuracy:
{svm_cv_nosamp_accuracy:.2f}%")
print(f"SVM Word2Vec No Sampling Accuracy:
{svm_w2v_nosamp_accuracy:.2f}%")
```

Pada segmen program 4.24 dilakukan perhitungan kecocokan hasil prediksi model dengan kolom "*self labeled*" atau kolom hasil label manual. Kode pada segmen tersebut merupakan contoh untuk penerapan model SVM dengan CountVectorizer dan Word2Vec. Untuk model lainnya dapat digunakan dengan mengganti variabel sesuai deklarasi model. Cara perhitungan kecocokan adalah dengan *load* model yang sudah dilatih sebelumnya dan prediksi data baru. Kemudian hitung kecocokan data hasil prediksi dengan kolom hasil label manual untuk dihitung akurasinya.

Segmen Program 4.25 Implementasi Ekstraksi Fitur

```
# Menambahkan kolom baru "CapitalNumber" dengan jumlah huruf
kapital dari kolom "Preprocessed Sarcasm"
df['CapitalNumber'] = df['Preprocessed
Sarcasm'].apply(count_capital_letters)
# Menambahkan kolom baru "InconsistentCap" untuk mendeteksi
huruf dengan case yang tidak konsisten
df['InconsistentCap'] = df['Preprocessed
Sarcasm'].apply(detect_inconsistent_case)
# Menambahkan kolom baru "PunctuationCount" dengan jumlah tanda
baca dari kolom "Preprocessed Sarcasm"
df['PunctuationCount'] = df['Preprocessed
Sarcasm'].apply(count_punctuation)
# Menambahkan kolom baru "AmountOfSlang" dengan jumlah ekspresi
slang dalam teks
df['AmountOfSlang'] = df['Preprocessed Sarcasm'].apply(lambda
x: count_slang(x, slangs))
```

Untuk pengujian implementasi dengan menggunakan *feature extraction*, dilakukan implementasi fungsi-fungsi ekstraksi fitur terlebih dahulu terhadap dataset baru sebelum diterapkan model prediksi.

Segmen Program 4.26 Implementasi Prediksi Model Fitur

```
#Prediksi
# Predict with SVM models
def predict_fituronly(model_path, test_data):
    model = joblib.load(model_path)
```

```
    X_numerical = test_data[['CapitalNumber',
'InconsistentCap', 'PunctuationCount', 'AmountOfSlang'] +
list(emoji_df['emoji'])]
    # Normalisasi nilai X
    scaler = MinMaxScaler()
    X_normalized = scaler.fit_transform(X_numerical)
    # Prediction
    y_pred = model.predict(X_normalized)
    return y_pred

def predict_fiturvector(model_path, vectorizer, test_data):
    model = joblib.load(model_path)
    X_text = vectorizer.transform(test_data['Preprocessed
Sarcasm'])
    X_numerical = test_data[['CapitalNumber',
'InconsistentCap', 'PunctuationCount', 'AmountOfSlang'] +
list(emoji_df['emoji'])]
    # Combine numerical and text features
    X_combined = hstack((X_numerical.values, X_text))
    scaler = MaxAbsScaler()
    X_normalized = scaler.fit_transform(X_combined)
    # Prediction
    y_pred = model.predict(X_normalized)
    return y_pred

word2vec_model =
Word2Vec.load('model/idwiki_word2vec_200_new_lower.model')

def predict_w2v(model_path, test_data):
    model = joblib.load(model_path)
    # Define features and target
    X_numerical = test_data[['CapitalNumber',
'InconsistentCap', 'PunctuationCount', 'AmountOfSlang'] +
list(emoji_df['emoji'])]
    text_features = test_data['Preprocessed Sarcasm']
    X_text = vectorize_data_w2v(text_features, word2vec_model)
    # Convert X_text to a sparse matrix
    X_text_sparse = csr_matrix(X_text)
    # Combine numerical and text features
    X_combined = hstack((X_numerical.values, X_text_sparse))
    # Normalize text features using MaxAbsScaler
    scaler = MaxAbsScaler()
    X_normalized = scaler.fit_transform(X_combined)
    y_pred = model.predict(X_normalized)
    return y_pred
```

Dalam implementasi model prediksi, menggunakan fungsi dengan beberapa parameter. Untuk hanya fitur membutuhkan parameter model yang sudah di *load* dan data, untuk fitur dan vektor, membutuhkan parameter model dan vektor yang sudah di *load* dan data. Untuk

Word2Vec membutuhkan model Word2Vec yang sudah dilatih, model prediksi yang sudah di *load* dan data.

4.4 *Website*

Pada sub-bab ini, dijelaskan program untuk pembuatan *website* untuk melakukan analisis sentimen dengan mengimplementasikan model yang sudah dibuat.

4.4.1 Fitur Melihat Panduan

Seperti yang sudah dijelaskan pada bab sebelumnya, halaman ini merupakan *landing page* yang juga berisi panduan dari penggunaan fitur analisis sentimen pada *website*.

Segmen Program 4.27 Bagian Head Kedua Fitur

```
<head>
    <title>Analisis Sentimen Skincare</title>

    <meta charset="UTF-8">
    <link rel="stylesheet" href="style.css">
    <!-- Boxicons CDN Link -->
    <link
href='https://unpkg.com/boxicons@2.0.7/css/boxicons.min.css'
rel='stylesheet'>
     <meta name="viewport" content="width=device-width,
initial-scale=1.0">


     <!-- Bootstrap CSS -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/
css/bootstrap.min.css" rel="stylesheet"

integrity="sha384-y3mdUoSZ+OW7LbJdxH9s6T0UomlUR8y2yrP6gjzoPLcGp
YPPCA2eD+5DrRP7G10F" crossorigin="anonymous">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
     <style>
        /* Googlefont Poppins CDN Link */
            @import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@200;
300;400;500;600;700&display=swap');
            *{
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            font-family: 'Poppins', sans-serif;
            }
            .sidebar{
            position: fixed;
```

```
height: 100%;
width: 240px;
background: #fa8fb1;
transition: all 0.5s ease;
}
.sidebar.active{
width: 60px;
}
.sidebar .logo-details{
height: 80px;
display: flex;
align-items: center;
}
.sidebar .logo-details i{
font-size: 28px;
font-weight: 500;
color: #fff;
min-width: 60px;
text-align: center
}
.sidebar .logo-details .logo_name{
color: #fff;
font-size: 24px;
font-weight: 500;
}
.sidebar .nav-links{
margin-top: 10px;
}
.sidebar .nav-links li{
position: relative;
list-style: none;
height: 50px;
}
.sidebar .nav-links li a{
height: 100%;
width: 100%;
display: flex;
align-items: center;
text-decoration: none;
transition: all 0.4s ease;
}
.sidebar .nav-links li a.active{
background: #f74780;
}
.sidebar .nav-links li a:hover{
background: #f74780;
}
.sidebar .nav-links li i{
min-width: 60px;
text-align: center;
font-size: 18px;
color: #fff;
```

```
}
.sidebar .nav-links li a .links_name{
color: #fff;
font-size: 15px;
font-weight: 400;
white-space: nowrap;
}
.sidebar .nav-links .log_out{
position: absolute;
bottom: 0;
width: 100%;
}
.home-section{
position: relative;
background: #d8d8d6;
min-height: 100vh;
width: calc(100% - 240px);
left: 240px;
transition: all 0.5s ease;
}
.sidebar.active ~ .home-section{
width: calc(100% - 60px);
left: 60px;
}
.home-section nav{
display: flex;
justify-content: space-between;
height: 80px;
background: #fff;
display: flex;
align-items: center;
position: fixed;
width: calc(100% - 240px);
left: 240px;
z-index: 100;
padding: 0 20px;
box-shadow: 0 1px 1px rgba(0, 0, 0, 0.1);
transition: all 0.5s ease;
}
.sidebar.active ~ .home-section nav{
left: 60px;
width: calc(100% - 60px);
}
.home-section nav .sidebar-button{
display: flex;
align-items: center;
font-size: 24px;
font-weight: 500;
}
nav .sidebar-button i{
font-size: 35px;
margin-right: 10px;
```

```
}

.home-section nav .profile-details{
display: flex;
align-items: center;
background: #F5F6FA;
border: 2px solid #EFEEF1;
border-radius: 6px;
height: 50px;
min-width: 190px;
padding: 0 15px 0 2px;
}
nav .profile-details img{
height: 40px;
width: 40px;
border-radius: 6px;
object-fit: cover;
}
nav .profile-details .admin_name{
font-size: 15px;
font-weight: 500;
color: #333;
margin: 0 10px;
white-space: nowrap;
}
nav .profile-details i{
font-size: 25px;
color: #333;
}
.home-section .home-content{
position: relative;
padding-top: 104px;
}
.home-content .overview-boxes{
/* display: flex; */
align-items: center;
justify-content: space-between;
flex-wrap: wrap;
padding: 0 20px;
margin-bottom: 26px;
}
.overview-boxes .box{
/* display: flex; */
align-items: center;
justify-content: center;
width: 100%;
background: #fff;
padding: 15px 14px;
border-radius: 12px;
box-shadow: 0 5px 10px rgba(0,0,0,0.1);
}
.overview-boxes .box-topic{
```

```
font-size: 20px;
font-weight: 500;
}

/* Responsive Media Query */
@media (max-width: 1240px) {
.sidebar{
    width: 60px;
}
.sidebar.active{
    width: 220px;
}
.home-section{
    width: calc(100% - 60px);
    left: 60px;
}
.sidebar.active ~ .home-section{
    /* width: calc(100% - 220px); */
    overflow: hidden;
    left: 220px;
}
.home-section nav{
    width: calc(100% - 60px);
    left: 60px;
}
.sidebar.active ~ .home-section nav{
    width: calc(100% - 220px);
    left: 220px;
}
}

@media (max-width: 1000px) {
.overview-boxes .box{
    width: calc(100% / 2 - 15px);
    margin-bottom: 15px;
}
}
@media (max-width: 700px) {
nav .sidebar-button .dashboard,
nav .profile-details .admin_name,
nav .profile-details i{
    display: none;
}
.home-section nav .profile-details{
    height: 50px;
    min-width: 40px;
}

}
@media (max-width: 550px) {
.overview-boxes .box{
    width: 100%;
```

```
                margin-bottom: 15px;
            }
            .sidebar.active ~ .home-section nav
.profile-details{
                display: none;
            }
            }
            @media (max-width: 400px) {
            .sidebar{
                width: 0;
            }
            .sidebar.active{
                width: 60px;
            }
            .home-section{
                width: 100%;
                left: 0;
            }
            .sidebar.active ~ .home-section{
                left: 60px;
                width: calc(100% - 60px);
            }
            .home-section nav{
                width: 100%;
                left: 0;
            }
            .sidebar.active ~ .home-section nav{
                left: 60px;
                width: calc(100% - 60px);
            }
            }
    </style>
  </head>
```

Pada bagian *head*, berisi *link* yang di-*import* yang nantinya akan berguna dalam pemanggilan *library* yang digunakan dalam pembuatan *website*, seperti Bootstrap dan Boxicons. Selain itu, juga terdapat CSS yang digunakan dalam membuat tampilan halaman.

Segmen Program 4.28 Bagian Body Fitur Melihat Panduan

```
<body>
  <div class="sidebar">
    <div class="logo-details">
      <i class='bx bx-user'></i>
      <span class="logo_name">Analisis Sentimen</span>
    </div>
      <ul class="nav-links">
        <li>
          <a href="index.php" class="active">
```

```
            <i class='bx bx-home-alt' ></i>
            <span class="links_name">Home</span>
          </a>
        </li>
        <li>
          <a href="analisis.php">
            <i class='bx bxs-analyse' ></i>
            <span class="links_name">Analisis</span>
          </a>
        </li>
      </ul>
  </div>
  <section class="home-section">
    <nav>
      <div class="sidebar-button">
        <i class='bx bx-menu sidebarBtn'></i>
        <span class="dashboard">Home</span>
      </div>
    </nav>

    <div class="home-content">
      <div class="overview-boxes">
        <div class="box">
          <div class="right-side">

            <div class="box-topic">Selamat Datang!</div>
                <p>Website ini bertujuan untuk membantu
pengguna dalam mengolah tweet review merek skincare lokal
dengan Analisis Sentimen.</p>
            </div>
            <br>
            <div class="box-topic">Panduan Penggunaan
Website</div>
                <p>Berikut adalah panduan dalam menggunakan
fitur Analisis pada Website ini:</p>
                <p>1. Buka halaman analisis</p>
                <p>2. Pilih metode yang ingin digunakan dalam
pengambilan data tweet</p>
                <p>3. Apabila mengambil data tweet dari Twitter
(X), maka isi form yang sudah disediakan</p>
                <p>4. Apabila mengunggah data tweet, maka
pastikan ketentuan kolom sudah sesuai</p>
                <p>5. Klik tombol "Submit" untuk memulai proses
analisis</p>
                <p>6. Hasil dari analisis sentimen akan
ditampilkan apabila proses sudah selesai</p>
            </div>


          </div>
        </div>
      </div>
```

```
    </div>


  </section>

  <script>
        let sidebar = document.querySelector(".sidebar");
        let sidebarBtn = document.querySelector(".sidebarBtn");
        sidebarBtn.onclick = function() {
        sidebar.classList.toggle("active");
        if(sidebar.classList.contains("active")){
        sidebarBtn.classList.replace("bx-menu"
,"bx-menu-alt-right");
        }else
        sidebarBtn.classList.replace("bx-menu-alt-right",
"bx-menu");
        }
 </script>
</body>
```

Bagian *body* dari fitur melihat panduan berisi *sidebar* dan berisi konten ucapan selamat datang dan list panduan tata cara penggunaan halaman analisis. Terdapat juga *script* yang berfungsi untuk *sidebar*.

4.4.2 Fitur Analisis Sentimen

Fitur ini berada pada halaman analisis yang digunakan untuk menjalankan program analisis sentimen. Pada fitur ini, terdapat 2 cara pengambilan sumber data dan pada halaman ini juga akan ditampilkan hasil proses analisis sentimen yang dijalankan.

Segmen Program 4.29 Konten Fitur Analisis Sentimen

```
<div class="home-content">
      <div class="overview-boxes">
          <div class="box">
            <div class="right-side">

                <div id="form-container" style="margin-top:
10px;">
                    <label for="data-source">Pilih Sumber
Data:</label>
                    <select class="form-control"
id="data-source" onchange="toggleForm()">
                      <option value="">-</option>
                      <option value="twitter">Ambil Data Tweet
Twitter</option>
                      <option value="upload">Unggah Data
```

```
Tweet</option>
                    </select>
                </div>

                <form id="twitter-form"
action="processpassing.php" method="post" class="hidden">
                  <br>
                  <label for="keyword">Keyword Twitter:</label>
                  <input type="text" class="form-control"
id="keyword" name="keyword" placeholder="Keyword">

                  <br>
                  <label for="keyword">Limit Tweet:</label>
                  <input type="number" class="form-control"
id="limit" name="limit" placeholder="Limit">

                  <div class="form-group">
                    <br>
                    <label for="date-range">Tanggal:</label>
                    <div class="input-group">
                      <div class="input-group-prepend">
                          <span class="input-group-text">Start
Date</span>
                      </div>
                      <input type="date" id="start-date"
name="start-date" class="form-control" placeholder="dd/mm/yyyy"
pattern="\d{2}/\d{2}/\d{4}">
                      <div class="input-group-append">
                        <span class="input-group-text">-</span>
                      </div>
                      <input type="date" id="end-date"
name="end-date" class="form-control" placeholder="dd/mm/yyyy"
pattern="\d{2}/\d{2}/\d{4}">
                      <div class="input-group-append">
                          <span class="input-group-text">End
Date</span>
                      </div>
                    </div>
                  </div>
                  <button class="btn btn-primary" type="submit"
style="background-color: #f74780; border-color:
#f74780;">Submit</button>
                </form>

                <form id="upload-form"
action="processpassingupload.php" method="post" class="hidden"
enctype="multipart/form-data">
                  <br>
                  <p id="triggermodal" style="color: blue;
cursor: pointer;" data-toggle="modal"
data-target="#modal"><u>Lihat ketentuan upload file</u></p>
```

```
                    <label for="tweet-file">Unggah File Data
Tweet Twitter:</label>
                    <br>
                    <input type="file" id="tweet-file"
name="tweet-file" accept=".csv,
application/vnd.openxmlformats-officedocument.spreadsheetml.she
et, application/vnd.ms-excel"><br>
                    <br>
                    <button class="btn btn-primary" type="submit"
style="background-color: #f74780; border-color:
#f74780;">Submit</button>
                </form>

                <!-- Modal -->
                <div class="modal fade" id="modal"
tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel"
aria-hidden="true">
                    <div class="modal-dialog" role="document">
                      <div class="modal-content">
                        <div class="modal-header">
                          <h5 class="modal-title"
id="exampleModalLabel">Ketentuan Upload File</h5>
                          <button type="button" class="close"
data-dismiss="modal" aria-label="Close">
                            <span
aria-hidden="true">&times;</span>
                          </button>
                        </div>
                        <div class="modal-body">
                          <p><b>Perhatikan kolom untuk file yang
akan diupload!</b></p>
                          <p style="color: red;">Kolom pada data
harus memiliki nama kolom seperti berikut:</p>
                          <table class="table table-striped">
                            <thead>
                              <tr>
                                <th scope="col">created_at</th>
                                <th scope="col">full_text</th>
                                <th scope="col">username</th>
                              </tr>
                            </thead>
                          </table>
                          <p>Keterangan:</p>
                          <p>1. created_at = kolom untuk tanggal
tweet</p>
                          <p>2. full_text = kolom untuk tweet</p>
                          <p>3. username = kolom untuk username
dari pemilik tweet</p>
                        </div>
                      </div>
                    </div>
                </div>
```

```
            </div>
          </div>
      </div>

      <div class="overview-boxes">
          <div class="box">
            <div class="right-side">
            <?php
              if (isset($_GET['csv_data'])) {
                  // Baca file CSV
                  $file = $_GET['csv_data'];
                  $handle = fopen($file, "r");

                  $rows = array_map('str_getcsv', file($file));
                  $header = array_shift($rows);
                  // Ambil index kolom sentimen
                  $sentimen_index = array_search("Sentiment
Label", $header);


                  // Inisialisasi variabel untuk menghitung
jumlah sentimen
                  $positifCount = 0;
                  $netralCount = 0;
                  $negatifCount = 0;
                  $totalCount = 0;

                  $isHeader = true;

                  // Baca baris per baris
                  while (($data = fgetcsv($handle, 1000, ","))
!== FALSE) {
                      if ($isHeader) {
                          // Skip header
                          $isHeader = false;
                          continue;
                      }

                      // Check if the indices exist before
accessing them
                      if (isset($data[$sentimen_index])) {
                          $sentimen = $data[$sentimen_index];
// kolom sentiment
                          // Hitung jumlah sentimen
                          if ($sentimen == 'Positif') {
                              $positifCount++;
                          } elseif ($sentimen == 'Netral') {
                              $netralCount++;
                          } elseif ($sentimen == 'Negatif') {
                              $negatifCount++;
                          }
```

```
                        $totalCount++;
                    }
                }

                // Tutup file CSV
                fclose($handle);

                // Hitung persentase
                $positifPercentage = ($positifCount /
$totalCount) * 100;
                $netralPercentage = ($netralCount /
$totalCount) * 100;
                $negatifPercentage = ($negatifCount /
$totalCount) * 100;

                // Tampilkan hasil
                echo "<table class='table-responsive'>";
                echo "<tbody>";
                echo "<tr>";
                echo "<td style='width:20%;'><label
for='hasilpositif'>Persentase Sentimen Positif:
<b>$positifPercentage%</b></label></td>";
                echo "<td style='width:20%;'><label
for='hasilnetral'>Persentase Sentimen Netral:
<b>$netralPercentage%</b></label></td>";
                echo "<td style='width:20%;'><label
for='hasilnegatif'>Persentase Sentimen Negatif:
<b>$negatifPercentage%</b></label></td>";
                echo "</tr>";

// Python script untuk generate word clouds
                // Ubah ke parent directory
                $originalDir = getcwd();
                chdir('..');
                // Run Python dan passing data
                $command = escapeshellcmd("python3.11
generate_wordcloud.py");
                $output = shell_exec($command);
                chdir($originalDir);

                echo "<tr>";
                // word cloud
                  echo "<td>";
                  if ($positifPercentage > 0){
                    echo "<img src='../wordcloud_Positif.png'
style='width:65%;'>";
                  }
                  echo "</td>";
                  echo "<td>";
                  if ($netralPercentage > 0){
                    echo "<img src='../wordcloud_Netral.png'
style='width:65%;'>";
```

```
                    }
                    echo "</td>";
                    echo "<td>";
                    if ($negatifPercentage > 0){
                      echo "<img src='../wordcloud_Negatif.png'
style='width:65%;'>";
                    }
                    echo "</td>";
                  echo "</tr>";
                  echo "</tbody>";
                  echo "</table>";
              }
            ?>


                <label for="hasilnegatif"><b>Tabel Data Tweet
setelah Analisis Sentimen</b></label>
                    <div class="table-responsive">
                          <div style="overflow-x: auto;">
                              <table id="tweet-table"
class="table table-striped" style="width:100%; text-align:
center;">
                                    <thead>
                                        <tr>
                                            <th>No</th>
                                            <th>Username</th>
                                            <th>Tanggal</th>
                                            <th>Tweet</th>
                                            <th>Sentimen</th>
                                            <th>Sarkasme</th>
                                        </tr>
                                    </thead>
                                    <tbody>
                                    <?php
                                      // Display CSV data in the
table body
                                      if
(isset($_GET['csv_data'])) {
                                          $csv_file =
$_GET['csv_data'];
                                          $rows =
array_map('str_getcsv', file($csv_file));

                                          if (count($rows) > 0) {
                                              $header =
array_shift($rows); // Mengambil baris header

                                              // Mencari indeks
kolom berdasarkan nama header
                                              $username_index =
array_search("username", $header);
                                              $tanggal_index =
```

```
array_search("created_at", $header);
                                                $tweet_index =
array_search("full_text", $header);
                                                $sentimen_index =
array_search("Sentiment Label", $header);
                                                $sarcasm_index =
array_search("Sarcasm Label", $header);

                                                // Menampilkan data
di tabel
                                                if (count($rows) >
0) {
                                                    $count = 1;
                                                    foreach ($rows
as $row) {
                                                        echo
"<tr>";
                                                        echo "<td>"
. $count . "</td>"; // Kolom untuk index
                                                        echo "<td>"
. (isset($row[$username_index]) ? $row[$username_index] : '') .
"</td>"; // Kolom username
                                                        echo "<td>"
. (isset($row[$tanggal_index]) ? $row[$tanggal_index] : '') .
"</td>"; // Kolom tanggal
                                                        echo "<td>"
. (isset($row[$tweet_index]) ? $row[$tweet_index] : '') .
"</td>"; // Kolom tweet
                                                        echo "<td>"
. (isset($row[$sentimen_index]) ? $row[$sentimen_index] : '') .
"</td>"; // Kolom sentiment
                                                        echo "<td>"
. (isset($row[$sarcasm_index]) ? $row[$sarcasm_index] : '') .
"</td>"; // Kolom sarcasm
                                                        echo
"</tr>";
                                                        $count++;
                                                    }
                                                } else {
                                                    echo "<tr><td
colspan='6'>No data available</td></tr>";
                                                }
                                            } else {
                                                echo "<tr><td
colspan='6'>No data available</td></tr>";
                                            }
                                        } else {
                                            echo "<tr><td
colspan='6'>No data available</td></tr>";
                                        }
                                    ?>
```

```
                                    </tbody>
                                </table>
                            </div>
                        </div>

                    </div>
                </div>
            </div>

        </div>

        </div>
```

Pada fitur analisis sentimen di *website*, untuk *head*, CSS, dan *sidebar* sama seperti fitur melihat panduan, yang berbeda adalah pada *home-content*. Konten yang ada pada fitur ini adalah terdapat dua *form* dimana *form* pertama adalah untuk pengambilan data *tweet* dengan mengambil langsung dari aplikasi Twitter atau X. *Form* ini memiliki id twitter-form dan ketika *user* sudah mengisi field dan submit, *form* akan POST data ke *processpassing*.php untuk memproses *input* yang sudah diisi oleh *user*.

Segmen Program 4.30 Program *processpassing.php*

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Retrieve form data
    $keyword = $_POST["keyword"];
    $start_date = $_POST["start-date"];
    $end_date = $_POST["end-date"];
    $limit = $_POST["limit"];

    // Ubah ke parent directory
    chdir('..');
    // Run Python dan passing data
    $command = escapeshellcmd("python3.11 sentimenweb.py
$keyword $start_date $end_date $limit");
    $output = shell_exec($command);
    echo $output;

    // Redirect dgn parameter csv hasil pengolahan
    header("Location:
analisis.php?csv_data=tweets-data/hasil_gabungan.csv");
    exit;

}
?>
```

Program ini akan mengambil data yang telah diisi oleh *user* untuk dikirimkan ke Python yang akan menjalankan proses *scraping* dan analisis sentimen dengan deteksi sarkasme. Ketika data sudah di *passing*, maka sistem akan menjalankan Python dan apabila sudah selesai dijalankan, maka akan mengembalikan hasil analisis sentimen dengan deteksi sarkasme ke halaman analisis. Hasil berupa *.csv* ini akan ditampilkan pada tabel yang ada pada halaman analisis.

Segmen Program 4.31 Penerimaan data di file Python

```
keyword = sys.argv[1]
start_date = sys.argv[2]
end_date = sys.argv[3]
limit = sys.argv[4]
```

Pengambilan data yang dikirimkan dari *form* ke file Python dapat dilihat pada segmen program 4.31 dimana masing-masing *input* akan disimpan ke variabel yang ada pada file Python.

Kemudian untuk *form* kedua, yaitu *form* untuk sumber data dengan mengunggah data untuk dianalisis sentimennya. *User* dapat melihat ketentuan pengunggahan file dengan modal yang muncul. Terdapat validasi *form* untuk melakukan pengecekkan *file* yang diunggah apakah sudah sesuai dengan ketentuan kolom. Apabila tidak, maka *input* tidak akan diproses. Apabila sudah sesuai, maka *form* akan POST ke *processpassingupload.php* untuk diproses juga ke file *Python*.

Segmen Program 4.32 Program *processpassingupload.php*

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Retrieve form data
    $info = pathinfo($_FILES['tweet-file']['name']);
    $ext = $info['extension']; // get  extension dari file
    $newname = "uploaduser.".$ext;

    $target = 'uploads/'.$newname;
    move_uploaded_file( $_FILES['tweet-file']['tmp_name'],
$target);

    // Ubah ke parent directory
    chdir('..');
    // Run Python dan passing data
    $command = escapeshellcmd("python3.11 sentimenweb2.py
$target");
    $output = shell_exec($command);
    echo $output;
```

```
    // Redirect dgn parameter csv hasil pengolahan
    header("Location:
analisis.php?csv_data=tweets-data/hasil_gabungan.csv");
    exit;
}
?>
```

Sama seperti *form* pengambilan data dari Twitter, untuk proses *passing* ini juga menjalankan Python, namun pada *form upload* ini, data disimpan ke *file* lokal dengan direktori dan nama *file* yang sudah ditentukan. Setelah tersimpan, maka Python akan dijalankan dengan *passing* direktori dari *file* yang tersimpan, sehingga dapat dikatakan bahwa proses *passing* nya tidak langsung ke Python.

Segmen Program 4.33 Penyesuaian data terunggah di file Python

```
fullFilePath = 'web/' + sys.argv[1]
 # Load dataset from uploaded file
   if fullFilePath.endswith('.csv'):
       df = pd.read_csv(fullFilePath)
   elif fullFilePath.endswith('.xlsx'):
       df = pd.read_excel(fullFilePath)
   else:
       raise ValueError("Unsupported file format")
```

Seperti pada Segmen Program 4.33, pada program Python, *file* akan di-*load* sesuai direktori yang di-*passing* sebelumnya. File akan dibaca dengan *library* Pandas untuk selanjutnya diproses. Dalam pemrosesan untuk kedua jenis sumber data, akan dilakukan *preprocessing* dan tahap-tahap evaluasi sentimen seperti pembuatan model. Untuk proses prediksi, digunakan salah satu model prediksi yang memiliki performa terbaik. Hasil dari analisis sentimen dengan deteksi sarkasme disimpan pada *file hasil_gabungan.csv* yang dikirimkan ke halaman analisis untuk ditampilkan kepada *user*. Ketika *hasil_gabungan.csv* sudah diterima di halaman utama, maka isi dari *.csv* tersebut akan di-*load* ke dalam tabel untuk ditampilkan kepada *user*, persentase untuk setiap sentimen juga dihitung, dan *run* program Python untuk membuat *wordcloud* yang juga akan ditampilkan kepada *user*.

Segmen Program 4.34 Program Python pembuatan WordCloud

```
def generate_wordcloud(text, sentiment):
    wordcloud = WordCloud(width=800, height=400,
```

```
background_color='white').generate(text)
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.savefig(f'wordcloud_{sentiment}.png')
    plt.close()

# Baca data dari argumen
csv_file = 'web/tweets-data/hasil_gabungan.csv'

# Baca CSV
df = pd.read_csv(csv_file)

# Generate Wordcloud
# Pastikan kolom yang diperlukan ada
if 'Sentiment Label' not in df.columns or 'full_text' not in
df.columns:
    print("Kolom yang diperlukan tidak ditemukan dalam CSV.")
    sys.exit(1)

# Pisahkan teks berdasarkan label sentimen
sentiments = df['Sentiment Label'].unique()
for sentiment in sentiments:
    sentiment_texts = df[df['Sentiment Label'] ==
sentiment]['full_text'].tolist()
    combined_text = ' '.join(sentiment_texts)
    generate_wordcloud(combined_text, sentiment)
```

Pemanggilan *run* pada Python pembuatan *wordcloud* adalah pada halaman analisis ketika analisis sentimen sudah selesai dilakukan. Pembuatan *wordcloud* diambil dari kolom *full_text* atau kolom yang berisi *tweet* pada data dengan pemisahan untuk setiap sentimen Positif, Negatif, dan Netral. Hasil dari *run* pembuatan *wordcloud* adalah foto *.png* dengan nama yang sudah pasti sehingga nanti di file *php* analisis dalam pemanggilan *source* nya sudah *fix*.