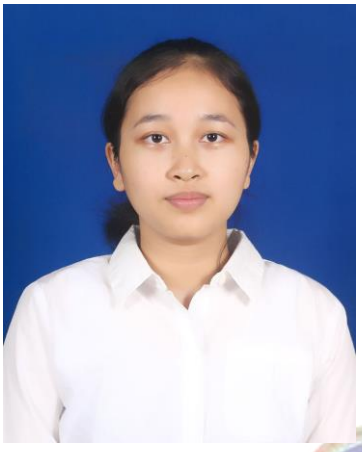
LAMPIRAN

Lampiran 1. Riwayat Hidup

RIWAYAT HIDUP



Made Donita Maharani lahir di Singaraja pada 5 Desember 2002. Penulis lahir dari pasangan suami istri Bapak Alm. I Gede Sudirtha dan Ibu Nyoman Kartini. Penulis berkebangsaan Indonesia dan beragama Hindu. Kini penulis beralamat di BTN Griya Permai Blok B/12, Baktiseraga, Kabupaten Buleleng, Provinsi Bali. Penulis menyelesaikan pendidikan dasar di SD Laboratorium Undiksha Singaraja dan lulus tahun 2015. Kemudian penulis melanjutkan pendidikan di SMP N 1 Singaraja dan lulus tahun 2018. Pada tahun 2021, penulis lulus dari SMA N 1 Singaraja jurusan Ilmu Pengetahuan Alam dan melanjutkan ke studi (S1) di Universitas Pendidikan Ganesha dengan Program Studi Sistem Informasi, Jurusan Teknik Informatika.

Lampiran 2. Surat Ketersediaan Pelabelan Data

KEMENTERIAN PENDIDIKAN TINGGI, SAINS, DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
Jalan Udayana Nomor 11 Singaraja - Bali Kode Pos 81116
Telepon (0362) 22570 Email: ftk@undiksha.ac.id Laman: http://ftk.undiksha.ac.id

Nomor : **2902**/UN48.11.1/DI.03.00/2025 Singaraja, 20 Oktober 2025
Perihal : Surat Permohonan Pengambilan Data

Yth. Kepala SMA Negeri 4 Singaraja
c.q. Guru Bahasa Indonesia
di tempat

Dengan hormat, sehubungan dengan proses penyelesaian Tugas Akhir/Skripsi, maka melalui surat ini kami mohon Bapak/Ibu berkenan memberikan data yang terkait dengan data yang dibutuhkan. Adapun mahasiswa yang akan melakukan pengambilan data seperti tersebut di bawah ini:

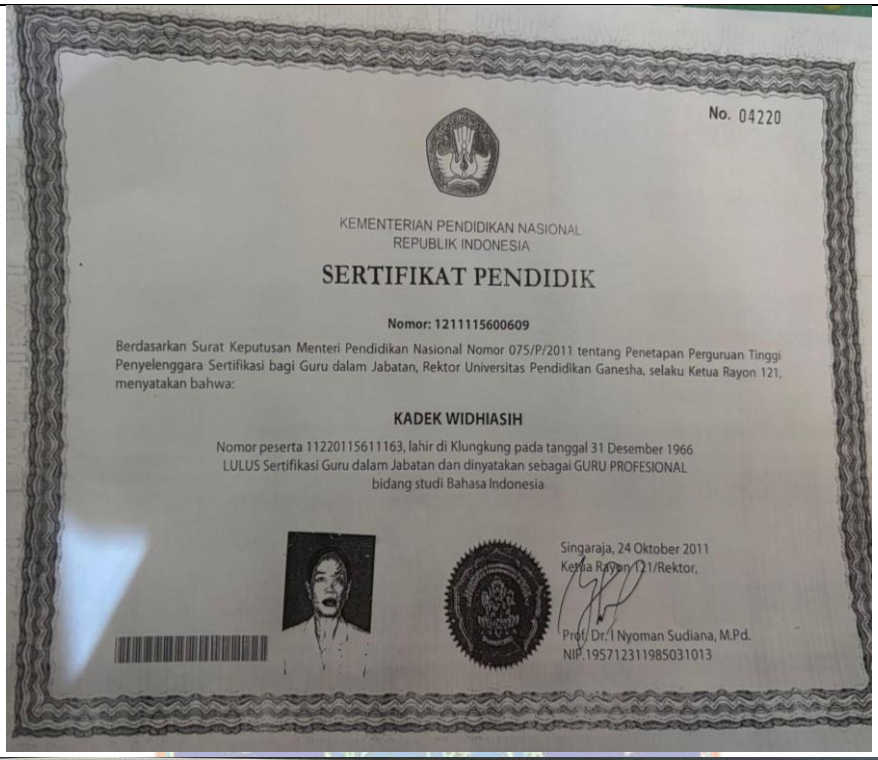
Nama	:	Made Donita Maharani
NIM	:	2115091065
Program Studi	:	Sistem Informasi
Jurusan	:	Teknik Informatika
Data yang dibutuhkan	:	Mengisi label Sentimen (Negatif dan Positif) pada komentar yang ada di formulir yang telah diberikan.
Judul Penelitian	:	Perbandingan Kinerja Ekstraksi Fitur TF-IDF & BOW Dalam Sentimen Analisis BPJS Kesehatan Dengan Algoritma XGBoost.

Demikian kami sampaikan, atas perhatian dan kerjasamanya, diucapkan terima kasih.

a.n Dekan
Wakil Dekan Bidang Akademik,

Made Windu Antara Kesiman
NIP 198211112008121001

Lampiran 3. Sertifikat Pendidik



No. 04220

KEMENTERIAN PENDIDIKAN NASIONAL
REPUBLIK INDONESIA

SERTIFIKAT PENDIDIK

Nomor: 1211115600609

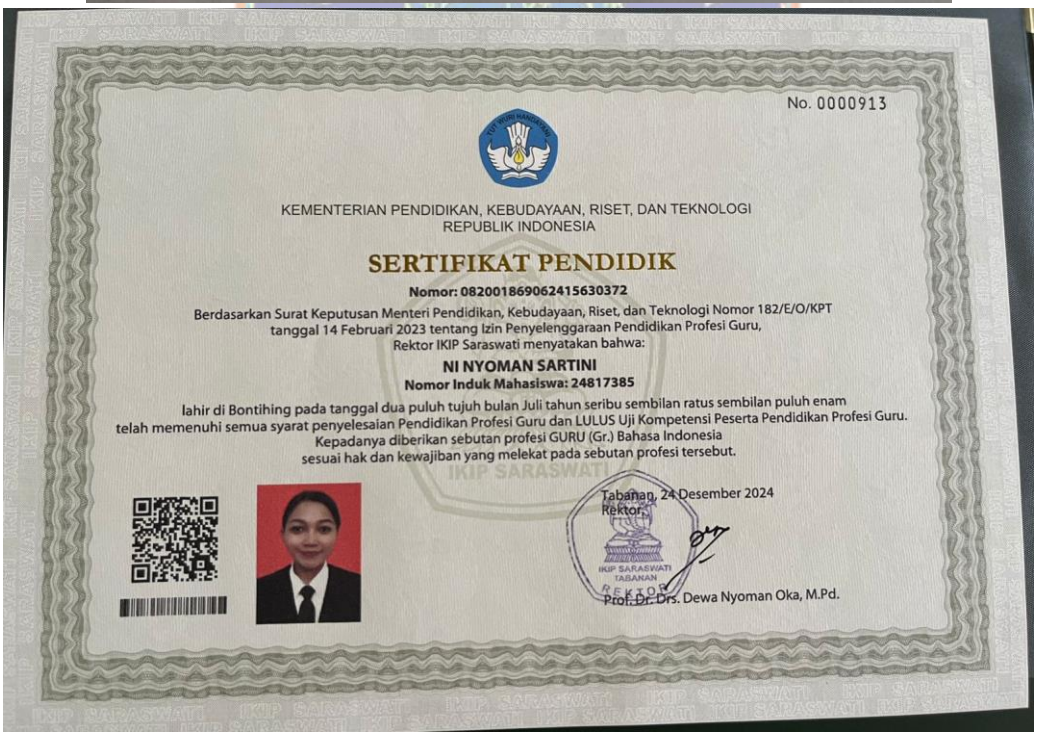
Berdasarkan Surat Keputusan Menteri Pendidikan Nasional Nomor 075/P/2011 tentang Penetapan Perguruan Tinggi Penyelenggara Sertifikasi bagi Guru dalam Jabatan, Rektor Universitas Pendidikan Ganesha, selaku Ketua Rayon 121, menyatakan bahwa:

KADEK WIDHIASIH

Nomor peserta 11220115611163, lahir di Klungkung pada tanggal 31 Desember 1966
LULUS Sertifikasi Guru dalam Jabatan dan dinyatakan sebagai GURU PROFESIONAL
bidang studi Bahasa Indonesia

Singaraja, 24 Oktober 2011
Ketua Rayon 121/Rektor,

Prof. Dr. I Nyoman Sudiana, M.Pd.
NIP.195712311985031013



No. 0000913

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
REPUBLIK INDONESIA

SERTIFIKAT PENDIDIK

Nomor: 082001869062415630372

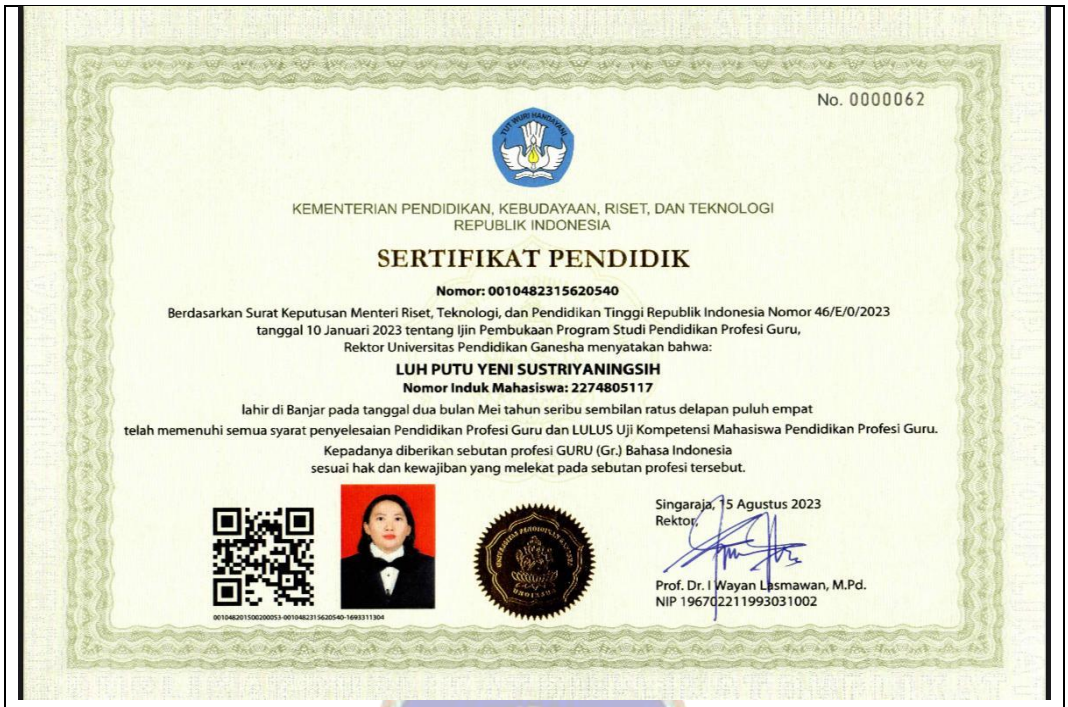
Berdasarkan Surat Keputusan Menteri Pendidikan, Kebudayaan, Riset, dan Teknologi Nomor 182/E/O/KPT tanggal 14 Februari 2023 tentang Izin Penyelenggaraan Pendidikan Profesi Guru,
Rektor IKIP Saraswati menyatakan bahwa:

NI NYOMAN SARTINI
Nomor Induk Mahasiswa: 24817385

lahir di Bontihing pada tanggal dua puluh tujuh bulan Juli tahun seribu sembilan ratus sembilan puluh enam telah memenuhi semua syarat penyelesaian Pendidikan Profesi Guru dan LULUS Uji Kompetensi Peserta Pendidikan Profesi Guru.
Kepadanya diberikan sebutan profesi GURU (Gr.) Bahasa Indonesia
sesuai hak dan kewajiban yang melekat pada sebutan profesi tersebut.

Tabanan, 24 Desember 2024
Rektor,

Prof. Dr. Drs. Dewa Nyoman Oka, M.Pd.



No. 0000062

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
REPUBLIK INDONESIA

SERTIFIKAT PENDIDIK

Nomor: 0010482315620540

Berdasarkan Surat Keputusan Menteri Riset, Teknologi, dan Pendidikan Tinggi Republik Indonesia Nomor 46/E/0/2023
tanggal 10 Januari 2023 tentang Ijin Pembukaan Program Studi Pendidikan Profesi Guru,
Rektor Universitas Pendidikan Ganesha menyatakan bahwa:

LUH PUTU YENI SUSTRIYANINGSIH
Nomor Induk Mahasiswa: 2274805117

lahir di Banjar pada tanggal dua bulan Mei tahun seribu sembilan ratus delapan puluh empat
telah memenuhi semua syarat penyelesaian Pendidikan Profesi Guru dan LULUS Uji Kompetensi Mahasiswa Pendidikan Profesi Guru.
Kepadanya diberikan sebutan profesi GURU (Gr.) Bahasa Indonesia
sesuai hak dan kewajiban yang melekat pada sebutan profesi tersebut.

Singaraja, 15 Agustus 2023
Rektor,

Prof. Dr. I Wayan Lasmawan, M.Pd.
NIP 196702211993031002

Lampiran 4. Surat Keterangan Validasi

KEMENTERIAN PENDIDIKAN TINGGI, SAINS,
DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
Jalan Udayana Nomor 11 Singaraja - Bali Kode Pos 81116
Telepon (0362) 22570 Email: ftk@undiksha.ac.id Laman: http://ftk.undiksha.ac.id

SURAT KETERANGAN VALIDASI

LABELING DATASET

Yang bertanda tangan di bawah ini:

Nama	: Dra. Kadek Widhiasih
NIP	: 196612311997022006

Menerangkan bahwa Mahasiswa Universitas Pendidikan Ganesha di bawah ini:

Nama	: Made Donita Maharani
NIM	: 2115091065
Prodi/Jurusan	: Sistem Informasi/Teknik Informatika

Memang benar bahwa dataset yang sudah dilabeli telah divalidasi pada tanggal 20 Oktober 2025. Demikian surat keterangan ini dibuat dengan sebenarnya untuk dapat digunakan sebagaimana mestinya.

Singaraja, 20 Oktober 2025
Ahli pakar II,

Dra. Kadek Widhiasih
NIP 196612311997022006

KEMENTERIAN PENDIDIKAN TINGGI, SAINS,
DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
Jalan Udayana Nomor 11 Singaraja - Bali Kode Pos 81116
Telepon (0362) 22570 Email: ftk@undiksha.ac.id Laman: http://ftk.undiksha.ac.id

SURAT KETERANGAN VALIDASI
LABELING DATASET

Yang bertanda tangan di bawah ini:

Nama	: Ni Nyoman Sartini, S.Pd
NIP	: 199607272023212016

Menerangkan bahwa Mahasiswa Universitas Pendidikan Ganesha di bawah ini:

Nama	: Made Donita Maharani
NIM	: 2115091065
Prodi/Jurusan	: Sistem Informasi/Teknik Informatika

Memang benar bahwa dataset yang sudah dilabeli telah divalidasi pada tanggal 20 Oktober 2025. Demikian surat keterangan ini dibuat dengan sebenarnya untuk dapat digunakan sebagaimana mestinya.

Singaraja, 20 Oktober 2025
Ahli pakar I,

Ni Nyoman Sartini, S.Pd
NIP 199607272023212016

KEMENTERIAN PENDIDIKAN TINGGI, SAINS,
DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
Jalan Udayana Nomor 11 Singaraja - Bali Kode Pos 81116
Telepon (0362) 22570 Email: ftk@undiksha.ac.id Laman: http://ftk.undiksha.ac.id

SURAT KETERANGAN VALIDASI
LABELING DATASET

Yang bertanda tangan di bawah ini:

Nama	: Luh Putu Yeni Sustriyaningsih, S.Pd
NIP	: 198405022023212016

Menerangkan bahwa Mahasiswa Universitas Pendidikan Ganesha di bawah ini:

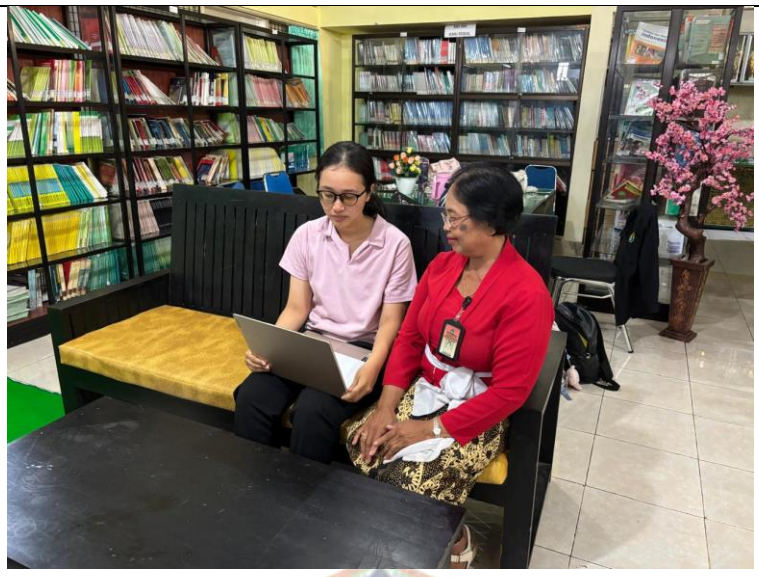
Nama	: Made Donita Maharani
NIM	: 2115091065
Prodi/Jurusan	: Sistem Informasi/Teknik Informatika

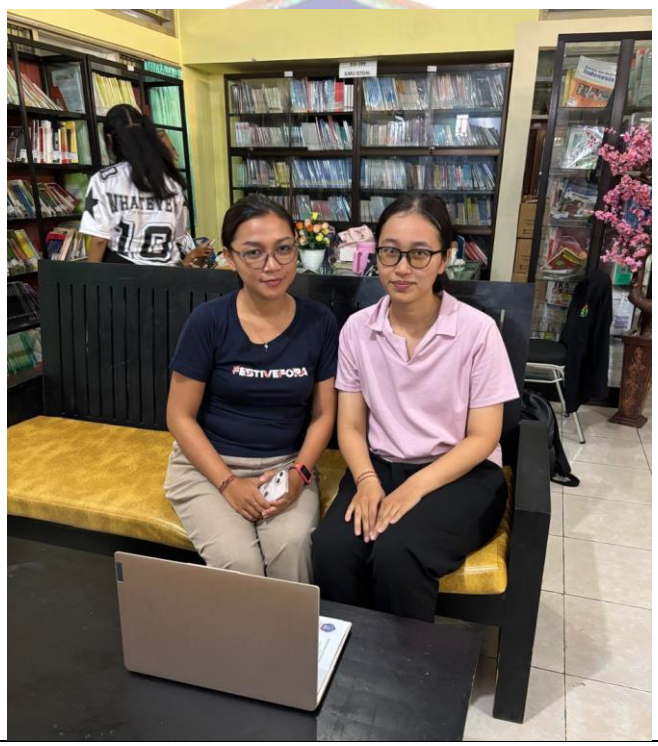
Memang benar bahwa dataset yang sudah dilabeli telah divalidasi pada tanggal 20 Oktober 2025. Demikian surat keterangan ini dibuat dengan sebenarnya untuk dapat digunakan sebagaimana mestinya.

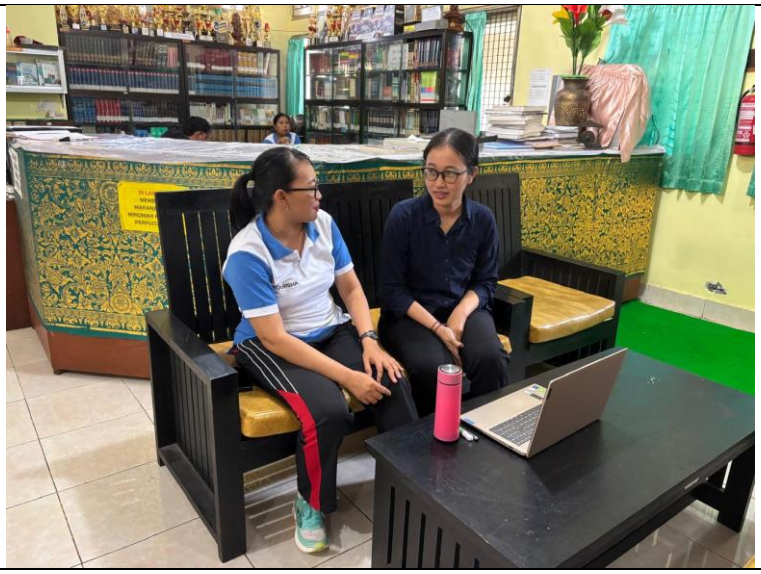
Singaraja, 20 Oktober 2025
Ahli pakar III,

Luh Putu Yeni Sustriyaningsih, S.Pd
NIP 198405022023212016

Lampiran 5. Dokumentasi bersama Validator







Lampiran 6. Proses Scraping Data di Aplikasi X

```
# Masukkan Twitter Auth Token
twitter_auth_token = "TOKEN"  # ganti dengan token masing-masing

# Install Node.js untuk tweet-harvest
!sudo apt-get update -y
!sudo apt-get install -y ca-certificates curl gnupg
!sudo mkdir -p /etc/apt/keyrings
!curl -fsSL https://deb.nodesource.com/gpgkey/nodesource-repo.gpg.key | sudo gpg --dearmor -o /etc/apt/keyrings/nodesource.gpg

!NODE_MAJOR=20 && echo "deb [signed-by=/etc/apt/keyrings/nodesource.gpg] https://deb.nodesource.com/node_$NODE_MAJOR.x nodistro main" | sudo tee /etc/apt/sources.list.d/nodesource.list

!sudo apt-get update -y
!sudo apt-get install -y nodejs
!node -v
filename = "bpjs.csv"  # nama file hasil scraping
search_keyword = 'BPJS Kesehatan since:2021-01-01 until:2025-07-31'
limit = 400  # jumlah maksimal tweet
```

```
# Jalankan tweet-harvest
!npx -y tweet-harvest@2.6.1 -o "{filename}" -s
"{search_keyword}" --tab "LATEST" -l {limit} --token
{twitter_auth_token}

import pandas as pd

file_path = "/content/tweets-data/bpjs.csv"  # path lengkap
df = pd.read_csv(file_path, delimiter=",")
print("Jumlah kolom awal:", df.shape[1])
print("Nama kolom tersedia:", df.columns.tolist())

# === Deteksi otomatis nama kolom teks ===
text_column = None
for col in ["full_text", "content", "text", "tweet"]:
    if col in df.columns:
        text_column = col
        break

if text_column is None:
    raise ValueError("⚠ Tidak ditemukan kolom teks
(full_text/content/text/tweet) di CSV")

print("☑ Kolom teks terdeteksi:", text_column)

# === 0. Install & Import Library ===
!pip install transformers pandas

import pandas as pd
from transformers import pipeline

# === 1. Load model deteksi bahasa ===
detector = pipeline("text-classification",
model="alexneakameni/language_detection")

# === 2. Baca file CSV hasil scraping/bersih ===
file_path = "/content/tweets-data/bpjs.csv"   # ganti sesuai
file
df = pd.read_csv(file_path)
```

```
print("Jumlah tweet awal:", len(df))

# === 3. Tentukan kolom teks (cek otomatis) ===
text_column = None
for col in ["full_text", "content", "text", "tweet"]:
    if col in df.columns:
        text_column = col
        break
if text_column is None:
    raise ValueError("⚠ Tidak ditemukan kolom teks di CSV")

# === 4. Deteksi bahasa setiap tweet ===
def detect_lang(text):
    try:
        result = detector(str(text)[:500])[0]  # batasi 500
karakter biar cepat
        return result["label"], result["score"]
    except:
        return "unknown", 0.0

df[["language", "lang_score"]] = df[text_column].apply(lambda
x: pd.Series(detect_lang(x)))

# === 5. Tampilkan distribusi bahasa ===
print("\nDistribusi bahasa yang terdeteksi:")
print(df["language"].value_counts())

# === 6. Filter hanya bahasa Indonesia ===
df_id = df[df["language"].str.contains("ind",
case=False)].copy()
print("\nJumlah tweet berbahasa Indonesia:", len(df_id))

# ☑ === 7. Pilih hanya kolom 'created_at' dan kolom teks ===
kolom_akhir = []
if "created_at" in df_id.columns:
    kolom_akhir.append("created_at")
kolom_akhir.append(text_column)

df_final = df_id[kolom_akhir + ["language",
"lang_score"]].copy()
```

```
# ☑ === 8. Simpan hasil filter ===
df_final.to_csv("bpjsk_clean_id.csv", index=False,
encoding="utf-8-sig")

print("\n☑ File akhir disimpan sebagai: buruk_clean_id.csv")
print("Kolom yang disimpan:", df_final.columns.tolist())
print("\nPreview 5 baris pertama:")
print(df_final.head())

# === 0. Import Library ===
import pandas as pd
import re
from transformers import pipeline

# === 1. Load CSV hasil deteksi bahasa ===
file_path = "/content/bpjsk_clean_id.csv"
df = pd.read_csv(file_path)

text_column = "full_text"
if text_column not in df.columns:
    raise ValueError(f"Kolom '{text_column}' tidak
ditemukan!")

# === 2. Filter hanya Bahasa Indonesia ===
df_id = df[df["language"].str.contains("ind", case=False,
na=False)].copy()
print("☑ Jumlah tweet bahasa Indonesia:", len(df_id))

# === 3. Preprocessing text ===
def preprocess_text(text):
    text = str(text).lower()
    text = re.sub(r"http\S+|www\S+|https\S+", "", text)  #
hapus link
    text = re.sub(r"@\w+", "", text)                      #
hapus mention
    text = re.sub(r"[^a-z\s]", "", text)                  #
hapus simbol/angka
    text = re.sub(r"\s+", " ", text).strip()              #
normalisasi spasi
```

```
    return text

df_id[text_column] =
df_id[text_column].apply(preprocess_text)

# === 4. Filter RT dan reply ===
df_id = df_id[~df_id[text_column].str.startswith(("rt",),
na=False)].copy()
print("☑ Setelah filter RT/reply:", len(df_id))

# === 5. Zero-shot classification ===
classifier = pipeline("zero-shot-classification",
model="joeddav/xlm-roberta-large-xnli")
candidate_labels = ["komentar masyarakat", "berita atau
promosi"]
batch_size = 32

# Keyword regex lebih spesifik
pattern_berita =
r"\b(berita|pengumuman|dirut|official|jurnalis|promosi|press
release|siaran pers|diumumkan oleh)\b"

def cek_keyword_berita(tweet):
    return bool(re.search(pattern_berita, tweet))

# Fungsi untuk label tweet
def label_tweet(tweet, res):
    max_score = max(res["scores"])
    max_label = res["labels"][res["scores"].index(max_score)]

    # Keyword kuat → otomatis berita/promosi
    if cek_keyword_berita(tweet):
        return "berita atau promosi"
    # Skor classifier sangat yakin → berita/promosi
    elif max_label == "berita atau promosi" and max_score >=
0.85:
        return "berita atau promosi"
    # Tweet panjang (>30 kata) cenderung berita/promosi
    elif len(tweet.split()) > 30 and max_label == "komentar
masyarakat":
```

```
        return "berita atau promosi"
    else:
        return "komentar masyarakat"

all_labels = []
for i in range(0, len(df_id), batch_size):
    batch = df_id[text_column].iloc[i:i+batch_size].tolist()
    results = classifier(batch, candidate_labels)

    for tweet, res in zip(batch, results):
        all_labels.append(label_tweet(tweet, res))

df_id["klasifikasi"] = all_labels

# === 6. Pisahkan hasil ===
df_komentar = df_id[df_id["klasifikasi"] == "komentar
masyarakat"]
df_berita_iklan = df_id[df_id["klasifikasi"] == "berita atau
promosi"]

# === 7. Simpan hasil akhir ===
df_komentar.to_csv("tweet_komentar.csv", index=False,
encoding="utf-8-sig")
df_berita_iklan.to_csv("tweet_berita_iklan.csv", index=False,
encoding="utf-8-sig")

print("\n☑ HASIL AKHIR:")
print("💬 Jumlah komentar masyarakat:", len(df_komentar))
print("📰 Jumlah berita/promosi:", len(df_berita_iklan))
print("\n➡ File disimpan sebagai:")
print(" ☑ tweet_komentar.csv")
print(" ☑ tweet_berita_iklan.csv")
```

Lampiran 7. Proses Preprocessing

```
# === 1. Cleansing ===
import pandas as pd
import re
from google.colab import files

# Upload file awal
print("Silakan upload file labeled_paling_fix.xls...")
uploaded = files.upload()
file_path = list(uploaded.keys())[0]

df = pd.read_excel(file_path)

# Fungsi cleansing
def cleansing(text):
    text = str(text)
    text = re.sub(r"http\S+|www\S+", "", text)
    text = re.sub(r"[^a-zA-Z\s]", " ", text)
    text = re.sub(r"\s+", " ", text).strip()
    return text

df['cleansing'] = df['ULASAN'].apply(cleansing)

# Simpan hasil cleansing, tetap sertakan kolom label
df[['ULASAN','cleansing','LABEL']].to_csv('/content/cleansing
.csv', index=False)
print("☑ Cleansing selesai. Hasil disimpan di cleansing.csv")

# Tampilkan hasil
display(df[['ULASAN', 'cleansing']].head(5))

# === 2. Case Folding ===
import pandas as pd

# 1. Baca hasil cleansing
df = pd.read_csv('/content/cleansing.csv')

# 2. Tambah kolom case folding
df['case_folding'] = df['cleansing'].str.lower()
```

```
# 3. Simpan hasil case folding (bawa cleansing + case_folding
+ LABEL)
df[['cleansing','case_folding','LABEL']].to_csv('/content/cas
e_folding.csv', index=False)

print("☑ Case folding selesai. Hasil disimpan di case_folding.csv")

# Tampilkan hasil
display(df[['cleansing', 'case_folding' ]].head(5))

# === 3. Tokenizing ===
import pandas as pd
import nltk
from nltk.tokenize import word_tokenize

# ☑ Download resource tokenizer NLTK
nltk.download('punkt')
nltk.download('punkt_tab')

# === 1. Baca file case_folding.csv ===
df = pd.read_csv('/content/case_folding.csv')

# === 2. Tokenizing ===
df['tokenizing'] = df['case_folding'].apply(lambda x:
word_tokenize(str(x)))

# === 3. Simpan hasil tokenizing ===
df[['case_folding', 'tokenizing',
'LABEL']].to_csv('/content/tokenizing.csv', index=False)
print("☑ Tokenizing selesai. Hasil disimpan di tokenizing.csv")

# === 4. Tampilkan hasil ===
df[['case_folding', 'tokenizing']].head(5)

# === 4. Stopword Removal ===
import pandas as pd
from Sastrawi.StopWordRemover.StopWordRemoverFactory import
StopWordRemoverFactory
import ast
```

```
# 1. Baca hasil tokenizing
df = pd.read_csv('/content/tokenizing.csv')

# 2. Pastikan kolom tokenizing berupa list
df['tokenizing'] = df['tokenizing'].apply(ast.literal_eval)

# 3. Ambil daftar stopword dari Sastrawi
stop_factory = StopWordRemoverFactory()
stopwords = set(stop_factory.get_stop_words())

# 4. Fungsi hapus stopword
def remove_stopwords(tokens):
    return [word for word in tokens if word not in stopwords]

# 5. Terapkan fungsi ke dataset
df['stopword_removal'] =
df['tokenizing'].apply(remove_stopwords)

# 6. Simpan hasil stopword removal (bawa tokenizing +
stopword_removal + label)
df[['tokenizing','stopword_removal','LABEL']].to_csv('/conten
t/stopword_removal.csv', index=False)

print("☑ Stopword removal selesai. Hasil disimpan di stopword_removal.csv")

# Tampilkan hasil (tanpa label, cukup 5 data)
display(df[['tokenizing', 'stopword_removal']].head(5))

# === 5. NORMALISASI ===
import pandas as pd
import ast

# 1. Baca hasil STOPWORD REMOVAL (sudah ada kolom LABEL di
dalamnya)
df = pd.read_csv('/content/stopword_removal.csv')
df['stopword_removal'] =
df['stopword_removal'].apply(ast.literal_eval)

# 2. Ambil kamus alay dari GitHub
url_alay =
```

```
"https://raw.githubusercontent.com/nasalsabila/kamus-
alay/master/colloquial-indonesian-lexicon.csv"
alay_df = pd.read_csv(url_alay)

# 3. Buat dictionary slang -> formal
slang_dict = dict(zip(alay_df['slang'], alay_df['formal']))

# 4. Fungsi normalisasi
def normalize_tokens(tokens):
    return [slang_dict.get(word.lower(), word) for word in
tokens]

# 5. Terapkan normalisasi ke dataset
df['normalisasi'] =
df['stopword_removal'].apply(normalize_tokens)

# 6. Simpan hasil normalisasi (tetap bawa kolom LABEL asli)
df[['stopword_removal', 'normalisasi',
'LABEL']].to_csv('/content/normalisasi.csv', index=False)

print("☑ NORMALISASI selesai. Hasil disimpan di normalisasi.csv")

# Tampilkan hasil
display(df[['stopword_removal', 'normalisasi' ]].head(5))

# === 6. STEMMING ===
import pandas as pd
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
import ast

# === 1. Baca file hasil normalisasi (sudah ada kolom LABEL)
===
# Pastikan file ada di path yang benar
df = pd.read_csv('/content/normalisasi.csv')

# Pastikan kolom 'normalisasi' berisi list (jika masih
string, ubah)
def convert_to_list(x):
    try:
        return ast.literal_eval(x) if isinstance(x, str) else
```

```
x
    except:
        return []

df['normalisasi'] = df['normalisasi'].apply(convert_to_list)

# === 2. Buat stemmer ===
stemmer = StemmerFactory().create_stemmer()

# === 3. Fungsi stemming ===
def stemming(tokens):
    return [stemmer.stem(word) for word in tokens if
isinstance(word, str)]

# === 4. Terapkan stemming pada setiap baris ===
df['stemming'] = df['normalisasi'].apply(stemming)

# === 5. Simpan hasil stemming + LABEL ===
df[['normalisasi', 'stemming',
'LABEL']].to_csv('/content/stemming.csv', index=False)
print("☑ Stemming selesai. Hasil akhir disimpan di 'stemming.csv'")

# === 6. Tampilkan hasil ===
df[['normalisasi', 'stemming']].head(5)
```

Lampiran 8. Proses TF-IDF (Term Frequency–Inverse Document Frequency)

```
import pandas as pd
from google.colab import files
from sklearn.feature_extraction.text import TfidfVectorizer

# 1. Upload file CSV hasil stemming
print("Silakan upload file stemming.csv...")
uploaded = files.upload()
file_path = list(uploaded.keys())[0]

# 2. Baca file
df = pd.read_csv(file_path)

# 3. Gabungkan token hasil stemming jadi string
df['stemming_joined']  =  df['stemming'].apply(lambda  x:  "
".join(eval(x)) if isinstance(x, str) else "")

# 4. TF-IDF Vectorizer
tfidf = TfidfVectorizer()
tfidf_matrix = tfidf.fit_transform(df['stemming_joined'])

# 5. Konversi hasil TF-IDF ke DataFrame
tfidf_df        =        pd.DataFrame(tfidf_matrix.toarray(),
columns=tfidf.get_feature_names_out())

# 6. Gabungkan dengan kolom label
tfidf_labeled = pd.concat([df[['LABEL']], tfidf_df], axis=1)

# 7. Simpan ke CSV
tfidf_labeled.to_csv("hasil_tfidf.csv", index=False)
print("☑ TF-IDF berhasil disimpan ke 'hasil_tfidf.csv'")

# 8. Tampilkan 10 kata dengan rata-rata TF-IDF tertinggi
mean_tfidf = tfidf_df.mean(axis=0)
top_tfidf = mean_tfidf.sort_values(ascending=False)
print("\n10 kata dengan rata-rata TF-IDF tertinggi:")
print(top_tfidf.head(10))
```

Lampiran 9. Proses BoW (Bag of Words)

```
import pandas as pd
from google.colab import files
from sklearn.feature_extraction.text import CountVectorizer

# 1. Upload file CSV hasil stemming
print("Silakan upload file stemming.csv...")
uploaded = files.upload()
file_path = list(uploaded.keys())[0]

# 2. Baca file
df = pd.read_csv(file_path)

# 3. Gabungkan token hasil stemming jadi string
df['stemming_joined'] = df['stemming'].apply(lambda x: "
".join(eval(x)) if isinstance(x, str) else "")

# 4. Inisialisasi BoW (CountVectorizer)
bow = CountVectorizer()
bow_matrix = bow.fit_transform(df['stemming_joined'])

# 5. Konversi hasil BoW ke DataFrame
bow_df = pd.DataFrame(bow_matrix.toarray(),
columns=bow.get_feature_names_out())

# 6. Gabungkan dengan kolom label
bow_labeled = pd.concat([df[['LABEL']], bow_df], axis=1)

# 7. Simpan ke CSV
bow_labeled.to_csv("/content/hasil_bow.csv", index=False)
print("☑ BoW berhasil disimpan ke 'hasil_bow.csv'")

# 8. Download file hasil
files.download("/content/hasil_bow.csv")

# 9. Tampilkan 10 kata yang paling sering muncul
word_freq = bow_df.sum(axis=0).sort_values(ascending=False)
print("\n10 kata yang paling sering muncul (BoW):")
print(word_freq.head(10))
```

Lampiran 10. Proses XGBoost + TF-IDF/BoW + scale_post_weight

```
# === 1. Import Library ===
import pandas as pd
import numpy as np
from sklearn.model_selection import RandomizedSearchCV,
KFold, cross_val_predict
from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score, precision_score,
recall_score, f1_score
from sklearn.preprocessing import LabelEncoder
import xgboost as xgb
import joblib
from google.colab import files
import seaborn as sns
import matplotlib.pyplot as plt
import random, os

# === 1A. Set seed agar hasil stabil ===
SEED = 42
np.random.seed(SEED)
random.seed(SEED)
os.environ['PYTHONHASHSEED'] = str(SEED)

# === 2. Load Data ===
df = pd.read_csv("hasil_tfidf.csv")    # sesuaikan nama file
X = df.drop(columns=['LABEL'])
y = df['LABEL']

# === 3. Encode Label ===
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# === 4. Hitung rasio kelas ===
neg = np.sum(y_encoded == 0)
pos = np.sum(y_encoded == 1)
scale_pos_weight = neg / pos
print(f"Rasio kelas (negatif:positif) = {neg}:{pos}")
print(f"scale_pos_weight = {scale_pos_weight:.2f}")

# === 5. Definisi model dasar XGBoost ===
```

```
model = xgb.XGBClassifier(
    objective='binary:logistic',
    eval_metric='logloss',
    random_state=SEED,
    use_label_encoder=False,
    tree_method='hist'
)

# === 6. Parameter untuk RandomizedSearch ===
param_dist = {
    'n_estimators': [200, 300, 400],
    'learning_rate': [0.05, 0.08, 0.1],
    'max_depth': [5, 7, 9],
    'min_child_weight': [1, 3],
    'gamma': [0, 0.05, 0.1],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0],

    'scale_pos_weight': [scale_pos_weight, scale_pos_weight *
1.1, scale_pos_weight * 0.9]
}

# === 7. Definisi 10-Fold Cross Validation ===
cv = KFold(n_splits=10, shuffle=True, random_state=SEED)

# === 8. RandomizedSearchCV ===
rand_search = RandomizedSearchCV(
    estimator=model,
    param_distributions=param_dist,
    n_iter=15,
    scoring='f1',
    cv=cv,
    n_jobs=-1,
    verbose=2,
    random_state=SEED
)

# === 9. Jalankan pencarian parameter terbaik ===
print("\n🚀 Mulai proses tuning dengan 10-Fold Cross
Validation ...")
```

```
rand_search.fit(X, y_encoded)
print("☑ Selesai tuning.\n")

# === 10. Hasil parameter terbaik ===
print("=== Parameter Terbaik dari RandomizedSearchCV ===")
print(rand_search.best_params_)
print(f"F1-Score rata-rata CV terbaik:
{rand_search.best_score_:.4f}")

# === 11. Evaluasi model dengan 10-Fold Cross Validation ===
best_model = rand_search.best_estimator_
y_pred_cv = cross_val_predict(best_model, X, y_encoded,
cv=cv, n_jobs=-1)

cm = confusion_matrix(y_encoded, y_pred_cv)
tn, fp, fn, tp = cm.ravel()

accuracy = accuracy_score(y_encoded, y_pred_cv)
precision = precision_score(y_encoded, y_pred_cv)
recall = recall_score(y_encoded, y_pred_cv)
f1 = f1_score(y_encoded, y_pred_cv)

print("\n=== Hasil Evaluasi Model XGBoost (10-Fold CV) ===")
print(f"TP: {tp}, FP: {fp}, FN: {fn}, TN: {tn}")
print(f"Akurasi : {accuracy:.4f}")
print(f"Presisi : {precision:.4f}")
print(f"Recall  : {recall:.4f}")
print(f"F1-Score: {f1:.4f}")

print("\nClassification Report:")
print(classification_report(y_encoded, y_pred_cv,
target_names=le.classes_, digits=4))

# === 12. Visualisasi Confusion Matrix ===
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=["Prediksi Negatif","Prediksi
Positif"],
            yticklabels=["Aktual Negatif","Aktual Positif"])
plt.xlabel("Prediksi")
```

```
plt.ylabel("Aktual")
plt.title("Confusion Matrix - XGBoost (10-Fold Cross
Validation)")
plt.show()
```

Lampiran 11. Proses XGBoost + TF-IDF/BoW + sample_weight

```
# === 1. Import Library ===
import pandas as pd
import numpy as np
from sklearn.model_selection import RandomizedSearchCV, KFold
from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score, precision_score,
recall_score, f1_score
from sklearn.preprocessing import LabelEncoder
import xgboost as xgb
import joblib
from google.colab import files
import seaborn as sns
import matplotlib.pyplot as plt
import random, os

# === 1A. Set seed agar hasil stabil ===
SEED = 42
np.random.seed(SEED)
random.seed(SEED)
os.environ['PYTHONHASHSEED'] = str(SEED)

# === 2. Load Data ===
df = pd.read_csv("hasil_bow.csv")    # sesuaikan nama file
X = df.drop(columns=['LABEL'])
y = df['LABEL']

# === 3. Encode Label ===
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# === 4. Hitung rasio kelas ===
neg = np.sum(y_encoded == 0)
```

```
pos = np.sum(y_encoded == 1)
ratio = neg / pos
print(f"Rasio kelas (negatif:positif) = {neg}:{pos}")

# === 5. Buat sample_weight untuk tiap instance ===
# Bobot kelas mayoritas = 1, minoritas = rasio
class_weights = {0: 1, 1: ratio}
sample_weights = np.array([class_weights[class_id] for
class_id in y_encoded])

# === 6. Definisi model dasar XGBoost ===
model = xgb.XGBClassifier(
    objective='binary:logistic',
    eval_metric='logloss',
    random_state=SEED,
    use_label_encoder=False,
    tree_method='hist'
)

# === 7. Parameter untuk RandomizedSearch ===
param_dist = {
    'n_estimators': [200, 300, 400],
    'learning_rate': [0.05, 0.08, 0.1],
    'max_depth': [5, 7, 9],
    'min_child_weight': [1, 3],
    'gamma': [0, 0.05, 0.1],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}

# === 8. Definisi 10-Fold Cross Validation ===
cv = KFold(n_splits=10, shuffle=True, random_state=SEED)

# === 9. RandomizedSearchCV dengan sample_weight ===
rand_search = RandomizedSearchCV(
    estimator=model,
    param_distributions=param_dist,
    n_iter=15,
    scoring='f1',
    cv=cv,
```

```
    n_jobs=-1,
    verbose=2,
    random_state=SEED
)

# === 10. Jalankan pencarian parameter terbaik ===
print("\n🚀 Mulai proses tuning dengan 10-Fold Cross
Validation ...")
rand_search.fit(X, y_encoded, sample_weight=sample_weights)
print("☑ Selesai tuning.\n")

# === 11. Hasil parameter terbaik ===
print("=== Parameter Terbaik dari RandomizedSearchCV ===")
print(rand_search.best_params_)
print(f"F1-Score rata-rata CV terbaik:
{rand_search.best_score_:.4f}")

# === 12. Evaluasi model dengan 10-Fold Cross Validation
(manual, support sample_weight) ===
best_model = rand_search.best_estimator_

y_pred_cv = np.zeros_like(y_encoded)

print("\n🚀 Mulai evaluasi dengan 10-Fold Cross Validation
(manual)...")
for fold, (train_idx, test_idx) in enumerate(cv.split(X), 1):
    X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
    y_train, y_test = y_encoded[train_idx],
y_encoded[test_idx]
    sample_weight_train = sample_weights[train_idx]

    best_model.fit(X_train, y_train,
sample_weight=sample_weight_train)
    y_pred_cv[test_idx] = best_model.predict(X_test)
    print(f"Fold {fold} selesai ☑")

print("☑ Evaluasi selesai.\n")

# === 13. Hitung metrik evaluasi ===
```

```
cm = confusion_matrix(y_encoded, y_pred_cv)
tn, fp, fn, tp = cm.ravel()

accuracy = accuracy_score(y_encoded, y_pred_cv)
precision = precision_score(y_encoded, y_pred_cv)
recall = recall_score(y_encoded, y_pred_cv)
f1 = f1_score(y_encoded, y_pred_cv)

print("=== Hasil Evaluasi Model XGBoost (10-Fold CV dengan
sample_weight) ===")
print(f"TP: {tp}, FP: {fp}, FN: {fn}, TN: {tn}")
print(f"Akurasi : {accuracy:.4f}")
print(f"Presisi : {precision:.4f}")
print(f"Recall  : {recall:.4f}")
print(f"F1-Score: {f1:.4f}")

print("\n" + classification_report(y_encoded, y_pred_cv,
target_names=le.classes_, digits=4))

# === 14. Visualisasi Confusion Matrix ===
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=["Prediksi Negatif","Prediksi
Positif"],
            yticklabels=["Aktual Negatif","Aktual Positif"])
plt.xlabel("Prediksi")
plt.ylabel("Aktual")
plt.title("Confusion Matrix - XGBoost (10-Fold CV,
sample_weight)")
plt.show()
```

Lampiran 12. Proses XGBoost + TF-IDF/BoW + sample_weight + SMOTE

```
# === 1. Import Library ===
import pandas as pd
import numpy as np
from sklearn.model_selection import RandomizedSearchCV,
StratifiedKFold
```

```
from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score, precision_score,
recall_score, f1_score
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
import xgboost as xgb
import joblib
from google.colab import files
import seaborn as sns
import matplotlib.pyplot as plt
import random, os

# === 1A. Set seed untuk reproduktifitas ===
SEED = 42
np.random.seed(SEED)
random.seed(SEED)
os.environ['PYTHONHASHSEED'] = str(SEED)

# === 2. Load Data ===
df = pd.read_csv("hasil_tfidf.csv")   # bisa diganti ke
hasil_bow.csv
X = df.drop(columns=['LABEL'])
y = df['LABEL']

# === 3. Encode Label ===
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# === 4. Hitung rasio kelas sebelum SMOTE ===
neg = np.sum(y_encoded == 0)
pos = np.sum(y_encoded == 1)
ratio = neg / pos
print(f"Rasio kelas (negatif:positif) sebelum SMOTE =
{neg}:{pos}")

# === 5. Terapkan SMOTE untuk penyeimbangan data ===
smote = SMOTE(random_state=SEED)
```

```
X_res, y_res = smote.fit_resample(X, y_encoded)
print(f"Setelah SMOTE: {np.bincount(y_res)} (Seimbang ✅)")

# === 6. Buat sample_weight ===
class_weights = {0: 1, 1: ratio}  # meskipun sudah seimbang,
tetap untuk stabilitas
sample_weights = np.array([class_weights[c] for c in y_res])

# === 7. Definisi model dasar XGBoost ===
model = xgb.XGBClassifier(
    objective='binary:logistic',
    eval_metric='logloss',
    random_state=SEED,
    use_label_encoder=False,
    tree_method='hist'
)

# === 8. Parameter RandomizedSearchCV ===
param_dist = {
    'n_estimators': [200, 300, 400],
    'learning_rate': [0.05, 0.08, 0.1],
    'max_depth': [5, 7, 9],
    'min_child_weight': [1, 3],
    'gamma': [0, 0.05, 0.1],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}

# === 9. 10-Fold Cross Validation ===
cv = StratifiedKFold(n_splits=10, shuffle=True,
random_state=SEED)

# === 10. RandomizedSearchCV dengan sample_weight ===
rand_search = RandomizedSearchCV(
    estimator=model,
    param_distributions=param_dist,
    n_iter=15,
```

```
    scoring='f1',
    cv=cv,
    n_jobs=-1,
    verbose=2,
    random_state=SEED
)

print("\n🚀 Mulai RandomizedSearchCV (10-Fold CV, SMOTE +
sample_weight)...")
rand_search.fit(X_res, y_res, sample_weight=sample_weights)
print("✅ Selesai tuning.\n")

# === 11. Ambil best model ===
best_model = rand_search.best_estimator_
print("=== Parameter Terbaik ===")
print(rand_search.best_params_)
print(f"F1-Score rata-rata CV terbaik:
{rand_search.best_score_:.4f}")

# === 12. Evaluasi model dengan 10-Fold CV manual ===
y_pred_cv = np.zeros_like(y_res)
for fold, (train_idx, test_idx) in enumerate(cv.split(X_res,
y_res), 1):
    X_train, X_test = X_res.iloc[train_idx],
X_res.iloc[test_idx]
    y_train, y_test = y_res[train_idx], y_res[test_idx]
    sample_weight_train = sample_weights[train_idx]

    best_model.fit(X_train, y_train,
sample_weight=sample_weight_train)
    y_pred_cv[test_idx] = best_model.predict(X_test)
    print(f"Fold {fold} selesai ✅")

# === 13. Hitung metrik evaluasi ===
cm = confusion_matrix(y_res, y_pred_cv)
tn, fp, fn, tp = cm.ravel()
```

```
accuracy = accuracy_score(y_res, y_pred_cv)
precision = precision_score(y_res, y_pred_cv)
recall = recall_score(y_res, y_pred_cv)
f1 = f1_score(y_res, y_pred_cv)

print("\n=== Hasil Evaluasi XGBoost (10-Fold CV, SMOTE +
sample_weight) ===")
print(f"TP: {tp}, FP: {fp}, FN: {fn}, TN: {tn}")
print(f"Akurasi : {accuracy:.4f}")
print(f"Presisi : {precision:.4f}")
print(f"Recall  : {recall:.4f}")
print(f"F1-Score: {f1:.4f}")
print(classification_report(y_res, y_pred_cv,
target_names=le.classes_, digits=4))

# === 14. Visualisasi Confusion Matrix ===
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=["Prediksi Negatif","Prediksi
Positif"],
            yticklabels=["Aktual Negatif","Aktual Positif"])
plt.xlabel("Prediksi")
plt.ylabel("Aktual")
plt.title("Confusion Matrix - XGBoost (10-Fold CV, SMOTE +
sample_weight)")
plt.show()

# === 15. Simpan model ===
joblib.dump(best_model, 'model_xgb_best_SMOTE.pkl')
files.download('model_xgb_best_SMOTE.pkl')

print("\n📁 Model akhir berhasil disimpan!")
```

Lampiran 13. Proses XGBoost + TF-IDF/BoW

```
# === 1. Import Library ===
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import KFold
from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score, precision_score,
recall_score, f1_score
from sklearn.preprocessing import LabelEncoder
import xgboost as xgb
import joblib
from google.colab import files
import seaborn as sns
import matplotlib.pyplot as plt
import random, os

# === 1A. Set seed agar hasil stabil ===
SEED = 42
np.random.seed(SEED)
random.seed(SEED)
os.environ['PYTHONHASHSEED'] = str(SEED)

# === 2. Load Data TFIDF ===
df = pd.read_csv("hasil_tfidf.csv") # sesuaikan nama file
X = df.drop(columns=['LABEL'])
y = df['LABEL']

# === 3. Encode Label ===
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# === 4. Tampilkan rasio kelas ===
neg = np.sum(y_encoded == 0)
pos = np.sum(y_encoded == 1)
print(f"Rasio kelas (negatif:positif) = {neg}:{pos}")

# === 5. Definisi Model XGBoost (PARAMETER DEFAULT RESMI)
model = xgb.XGBClassifier()

# === 6. Definisi 10-Fold Cross Validation ===
cv = KFold(n_splits=10, shuffle=True, random_state=SEED)
```

```
# === 7. Jalankan 10-Fold CV ===
y_pred_cv = np.zeros_like(y_encoded)

print("\n🚀 Mulai evaluasi 10-Fold Cross Validation (default
resmi XGBoost)...")
for fold, (train_idx, test_idx) in enumerate(cv.split(X), 1):

    X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
    y_train, y_test = y_encoded[train_idx],
y_encoded[test_idx]

    model.fit(X_train, y_train)
    y_pred_cv[test_idx] = model.predict(X_test)
    print(f"Fold {fold} selesai ☑")

print("☑ Evaluasi selesai.\n")

# === 8. Hitung metrik evaluasi ===
cm = confusion_matrix(y_encoded, y_pred_cv)
tn, fp, fn, tp = cm.ravel()

accuracy = accuracy_score(y_encoded, y_pred_cv)
precision = precision_score(y_encoded, y_pred_cv)
recall = recall_score(y_encoded, y_pred_cv)
f1 = f1_score(y_encoded, y_pred_cv)

print("=== Hasil Evaluasi Model XGBoost (10-Fold CV, Default
Resmi) ===")
print(f"TP: {tp}, FP: {fp}, FN: {fn}, TN: {tn}")
print(f"Akurasi : {accuracy:.4f}")
print(f"Presisi : {precision:.4f}")
print(f"Recall  : {recall:.4f}")
print(f"F1-Score: {f1:.4f}\n")

print(classification_report(y_encoded, y_pred_cv,
target_names=le.classes_, digits=4))

# === 9. Visualisasi Confusion Matrix ===
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
```

```
            xticklabels=["Prediksi Negatif","Prediksi
Positif"],
            yticklabels=["Aktual Negatif","Aktual Positif"])
plt.xlabel("Prediksi")
plt.ylabel("Aktual")
plt.title("Confusion Matrix - XGBoost (10-Fold CV, Default
Resmi)")
plt.show()
```

Lampiran 14. Proses XGBoost + TF-IDF/BoW + scale_post_weight (split data 80:20)

```
# === 1. Import Library ===
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, KFold,
RandomizedSearchCV
from sklearn.metrics import (
    classification_report, confusion_matrix,
    accuracy_score, precision_score, recall_score, f1_score
)
from sklearn.preprocessing import LabelEncoder
import xgboost as xgb
import seaborn as sns
import matplotlib.pyplot as plt
import joblib
import random, os

# === 1A. Set seed ===
SEED = 42
np.random.seed(SEED)
random.seed(SEED)
os.environ['PYTHONHASHSEED'] = str(SEED)

# === 2. Load Data ===
df = pd.read_csv("hasil_tfidf.csv")    # sesuaikan nama file
X = df.drop(columns=['LABEL'])
y = df['LABEL']

# === 3. Encode Label ===
```

```
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# === 4. Split Data (80% train, 20% test) ===
X_train, X_test, y_train, y_test = train_test_split(
    X, y_encoded, test_size=0.2, stratify=y_encoded,
random_state=SEED
)

# === 5. Hitung scale_pos_weight ===
neg = np.sum(y_train == 0)
pos = np.sum(y_train == 1)
scale_pos_weight = neg / pos
print(f"Rasio kelas (negatif:positif) = {neg}:{pos}")
print(f"scale_pos_weight = {scale_pos_weight:.2f}")

# === 6. Model dasar ===
model = xgb.XGBClassifier(
    objective='binary:logistic',
    eval_metric='logloss',
    use_label_encoder=False,
    random_state=SEED
)

# === 7. Parameter untuk RandomizedSearchCV ===
param_dist = {
    'n_estimators': [200, 300, 400],
    'learning_rate': [0.05, 0.08, 0.1],
    'max_depth': [5, 7, 9],
    'min_child_weight': [1, 3],
    'gamma': [0, 0.05, 0.1],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0],
    'scale_pos_weight': [scale_pos_weight, scale_pos_weight *
1.1]
}

# === 8. Inner CV: 10-fold pada data training ===
inner_cv = KFold(n_splits=10, shuffle=True,
random_state=SEED)
```

```
rand_search = RandomizedSearchCV(
    estimator=model,
    param_distributions=param_dist,
    n_iter=15,
    scoring='f1',
    cv=inner_cv,
    n_jobs=-1,
    verbose=2,
    random_state=SEED
)

# === 9. Jalankan RandomizedSearch pada 80% data training ===
rand_search.fit(X_train, y_train)

# === 10. Dapatkan best model ===
best_model = rand_search.best_estimator_
print("\n=== Parameter Terbaik dari RandomizedSearchCV (Inner
10-Fold) ===")
print(rand_search.best_params_)
print(f"F1-Score terbaik (cross-val):
{rand_search.best_score_:.4f}")

# === 11. Latih ulang best model pada seluruh data training
(80%) ===
best_model.fit(X_train, y_train)

# === 12. Prediksi pada data uji (20%) ===
y_pred = best_model.predict(X_test)

# === 13. Evaluasi Model di Data Uji ===
print("\n=== Laporan Kinerja Model di Data Uji (20%) ===")
print(classification_report(y_test, y_pred, digits=4))

# === 14. Metrik tambahan ===
cm = confusion_matrix(y_test, y_pred)
tn, fp, fn, tp = cm.ravel()

print("\n=== Hasil Evaluasi Model di Data Uji ===")
print(f"TP: {tp}, FP: {fp}, FN: {fn}, TN: {tn}")
```

```
print(f"Akurasi : {accuracy_score(y_test, y_pred):.4f}")
print(f"Presisi : {precision_score(y_test, y_pred):.4f}")
print(f"Recall  : {recall_score(y_test, y_pred):.4f}")
print(f"F1-Score: {f1_score(y_test, y_pred):.4f}")

# === 15. Visualisasi Confusion Matrix ===
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=["Prediksi Negatif","Prediksi
Positif"],
            yticklabels=["Aktual Negatif","Aktual Positif"])
plt.xlabel("Prediksi")
plt.ylabel("Aktual")
plt.title("Confusion Matrix - XGBoost (Data Uji 20%)")
plt.show()
```

Lampiran 15. XGBoost + TF-IDF/BoW + sample_weight (split data 80:20)

```
# === 1. Import Library ===
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, KFold,
RandomizedSearchCV
from sklearn.metrics import (
    classification_report, confusion_matrix,
    accuracy_score, precision_score, recall_score, f1_score
)
from sklearn.preprocessing import LabelEncoder
import xgboost as xgb
import seaborn as sns
import matplotlib.pyplot as plt
import joblib
import random, os

# === 1A. Set seed ===
SEED = 42
np.random.seed(SEED)
random.seed(SEED)
os.environ['PYTHONHASHSEED'] = str(SEED)
```

```
# === 2. Load Data ===
df = pd.read_csv("hasil_bow.csv")    # sesuaikan nama file
X = df.drop(columns=['LABEL'])
y = df['LABEL']

# === 3. Encode Label ===
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# === 4. Split Data (80% train, 20% test) ===
X_train, X_test, y_train, y_test = train_test_split(
    X, y_encoded, test_size=0.2, stratify=y_encoded,
random_state=SEED
)

# === 5. Buat Sample Weight ===
neg = np.sum(y_train == 0)
pos = np.sum(y_train == 1)

# bobot kebalikan proporsi kelas
weight_pos = neg / pos
weight_neg = 1

sample_weight_train = np.array([weight_pos if label == 1 else
weight_neg for label in y_train])

print(f"Jumlah data kelas negatif = {neg}")
print(f"Jumlah data kelas positif = {pos}")
print(f"Bobot sample weight positif = {weight_pos:.2f}")
print(f"Bobot sample weight negatif = {weight_neg}")

# === 6. Model dasar (tanpa scale_pos_weight) ===
model = xgb.XGBClassifier(
    objective='binary:logistic',
    eval_metric='logloss',
    use_label_encoder=False,
    random_state=SEED
)
```

```
# === 7. Parameter RandomizedSearchCV ===
param_dist = {
    'n_estimators': [200, 300, 400],
    'learning_rate': [0.05, 0.08, 0.1],
    'max_depth': [5, 7, 9],
    'min_child_weight': [1, 3],
    'gamma': [0, 0.05, 0.1],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}

# === 8. Inner CV: 10-fold pada data training ===
inner_cv = KFold(n_splits=10, shuffle=True,
random_state=SEED)

rand_search = RandomizedSearchCV(
    estimator=model,
    param_distributions=param_dist,
    n_iter=15,
    scoring='f1',
    cv=inner_cv,
    n_jobs=-1,
    verbose=2,
    random_state=SEED
)

# === 9. Jalankan RandomizedSearch (dengan sample_weight) ===
rand_search.fit(X_train, y_train,
sample_weight=sample_weight_train)

# === 10. Dapatkan best model ===
best_model = rand_search.best_estimator_
print("\n=== Parameter Terbaik dari RandomizedSearchCV (Inner
10-Fold) ===")
print(rand_search.best_params_)
print(f"F1-Score terbaik (cross-val):
{rand_search.best_score_:.4f}")

# === 11. Latih ulang best model pada seluruh data training
(dengan sample_weight) ===
```

```
best_model.fit(X_train, y_train,
sample_weight=sample_weight_train)

# === 12. Prediksi pada data uji (20%) ===
y_pred = best_model.predict(X_test)

# === 13. Laporan Kinerja Model di Data Uji ===
print("\n=== Laporan Kinerja Model di Data Uji (20%) ===")
print(classification_report(y_test, y_pred, digits=4))

# === 14. Metrik tambahan ===
cm = confusion_matrix(y_test, y_pred)
tn, fp, fn, tp = cm.ravel()

print("\n=== Hasil Evaluasi Model di Data Uji ===")
print(f"TP: {tp}, FP: {fp}, FN: {fn}, TN: {tn}")
print(f"Akurasi : {accuracy_score(y_test, y_pred):.4f}")
print(f"Presisi : {precision_score(y_test, y_pred):.4f}")
print(f"Recall  : {recall_score(y_test, y_pred):.4f}")
print(f"F1-Score: {f1_score(y_test, y_pred):.4f}")

# === 15. Visualisasi Confusion Matrix ===
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=["Prediksi Negatif","Prediksi
Positif"],
            yticklabels=["Aktual Negatif","Aktual Positif"])
plt.xlabel("Prediksi")
plt.ylabel("Aktual")
plt.title("Confusion Matrix - XGBoost (Sample Weight, Data
Uji 20%)")
plt.show()
```

Lampiran 16. XGBoost + TF-IDF/BoW+ sample_weight + SMOTE (split data 80:20)

```
# === 1. Import Library ===
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, KFold,
RandomizedSearchCV
from sklearn.metrics import (
    classification_report, confusion_matrix,
    accuracy_score, precision_score, recall_score, f1_score
)
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from sklearn.utils.class_weight import compute_sample_weight
import xgboost as xgb
import seaborn as sns
import matplotlib.pyplot as plt
import joblib
import random, os
from google.colab import files

# === 1A. Set seed untuk hasil konsisten ===
SEED = 42
np.random.seed(SEED)
random.seed(SEED)
os.environ['PYTHONHASHSEED'] = str(SEED)

# === 2. Load Data ===
df = pd.read_csv("hasil_tfidf.csv")    # sesuaikan nama file
X = df.drop(columns=['LABEL'])
y = df['LABEL']

# === 3. Encode Label ===
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# === 4. Split Data (80% train, 20% test) ===
X_train, X_test, y_train, y_test = train_test_split(
    X, y_encoded, test_size=0.2, stratify=y_encoded,
random_state=SEED
)
print(f"☑ Data latih: {X_train.shape}, Data uji: {X_test.shape}")
```

```
# === 5. Terapkan SMOTE pada data training saja ===
print("\n🔄 Melakukan oversampling SMOTE pada data
training...")
smote = SMOTE(random_state=SEED)
X_train_res, y_train_res = smote.fit_resample(X_train,
y_train)
print(f"Sebelum SMOTE: {np.bincount(y_train)}")
print(f"Sesudah SMOTE: {np.bincount(y_train_res)}")

# === 6. Hitung sample weights (agar memperhatikan distribusi
asli) ===
sample_weights =
compute_sample_weight(class_weight='balanced', y=y_train_res)

# === 7. Definisi model dasar ===
model = xgb.XGBClassifier(
    objective='binary:logistic',
    eval_metric='logloss',
    use_label_encoder=False,
    random_state=SEED
)

# === 8. Parameter untuk RandomizedSearchCV ===
param_dist = {
    'n_estimators': [200, 300, 400],
    'learning_rate': [0.05, 0.08, 0.1],
    'max_depth': [5, 7, 9],
    'min_child_weight': [1, 3],
    'gamma': [0, 0.05, 0.1],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}

# === 9. Inner CV: 10-fold pada data training hasil SMOTE ===
inner_cv = KFold(n_splits=10, shuffle=True,
random_state=SEED)

rand_search = RandomizedSearchCV(
    estimator=model,
```

```
    param_distributions=param_dist,
    n_iter=15,
    scoring='f1',
    cv=inner_cv,
    n_jobs=-1,
    verbose=2,
    random_state=SEED
)

# === 10. Jalankan RandomizedSearchCV dengan sample weight +
SMOTE data ===
print("\n🚀 Menjalankan RandomizedSearchCV (10-fold)...")
rand_search.fit(X_train_res, y_train_res,
sample_weight=sample_weights)

# === 11. Dapatkan best model ===
best_model = rand_search.best_estimator_
print("\n=== Parameter Terbaik dari RandomizedSearchCV (Inner
10-Fold) ===")
print(rand_search.best_params_)
print(f"F1-Score terbaik (cross-val):
{rand_search.best_score_:.4f}")

# === 12. Latih ulang best model pada seluruh data training
hasil SMOTE ===
best_model.fit(X_train_res, y_train_res,
sample_weight=sample_weights)

# === 13. Prediksi pada data uji (20%) ===
y_pred = best_model.predict(X_test)

# === 14. Evaluasi Model di Data Uji ===
print("\n=== Laporan Kinerja Model di Data Uji (20%) ===")
print(classification_report(y_test, y_pred, digits=4))

# === 15. Metrik tambahan ===
cm = confusion_matrix(y_test, y_pred)
tn, fp, fn, tp = cm.ravel()

print("\n=== Hasil Evaluasi Model di Data Uji ===")
```

```
print(f"TP: {tp}, FP: {fp}, FN: {fn}, TN: {tn}")
print(f"Akurasi : {accuracy_score(y_test, y_pred):.4f}")
print(f"Presisi : {precision_score(y_test, y_pred):.4f}")
print(f"Recall  : {recall_score(y_test, y_pred):.4f}")
print(f"F1-Score: {f1_score(y_test, y_pred):.4f}")

# === 16. Visualisasi Confusion Matrix ===
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=["Prediksi Negatif","Prediksi
Positif"],
            yticklabels=["Aktual Negatif","Aktual Positif"])
plt.xlabel("Prediksi")
plt.ylabel("Aktual")
plt.title("Confusion Matrix - XGBoost (SMOTE + Sample Weight,
Data Uji 20%)")
plt.show()
```

Lampiran 17. UI (User Interface)

```
import streamlit as st
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
from wordcloud import WordCloud

from sklearn.feature_extraction.text import
CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier

from Sastrawi.StopWordRemover.StopWordRemoverFactory
import StopWordRemoverFactory
from Sastrawi.Stemmer.StemmerFactory import
StemmerFactory
```

```
from nltk.tokenize import word_tokenize
import nltk
nltk.download('punkt')

# Page Config
st.set_page_config(page_title="Analisis Sentimen
BPJS", layout="wide")
# Header
st.markdown("""
<div style="background-
color:#2E3A87;color:white;padding:60px 0 30px 0;
text-align:center;font-size:32px;font-weight:700;">
APLIKASI ANALISIS SENTIMEN BPJS KESEHATAN
</div>
""", unsafe_allow_html=True)
# Tabs
tab1, tab2, tab3 = st.tabs([
    "Dataset & Preprocessing",
    "Fitur Ekstraksi",
    "Analisis Sentimen"
])
#Tab 1: Dataset & Preprocessing
with tab1:
    st.subheader("Upload Dataset & Preprocessing")
    uploaded = st.file_uploader("Upload CSV/XLSX",
type=["csv", "xlsx"])

    if uploaded:
        df = pd.read_csv(uploaded) if
uploaded.name.endswith(".csv") else
pd.read_excel(uploaded)
```

```
        if "ULASAN" not in df.columns or "LABEL" not
in df.columns:
            st.error("Dataset harus memiliki kolom
ULASAN dan LABEL")
            st.stop()

        df = df[["ULASAN", "LABEL"]].copy()
        st.success("Dataset berhasil dimuat")

        # Cleansing & Case Folding
        def cleansing(text):
            text = str(text).lower()
            text = re.sub(r"http\S+|www\S+", "",
text)
            text = re.sub(r"[^a-z\s]", " ", text)
            text = re.sub(r"\s+", " ", text).strip()
            return text

        df["Cleansing"] =
df["ULASAN"].apply(cleansing)

        # Tokenizing
        df["Tokenizing"] =
df["Cleansing"].apply(word_tokenize)

        # Normalisasi
        slang = pd.read_csv(
            "https://raw.githubusercontent.com/nasals
abila/kamus-alay/master/colloquial-indonesian-
lexicon.csv"
        )
```

```
        slang_dict =
dict(zip(slang["slang"].str.lower(),
slang["formal"].str.lower()))
        df["Normalisasi"] =
df["Tokenizing"].apply(lambda x: [slang_dict.get(w,
w) for w in x])

        # Stopword Removal
        stopwords =
set(StopWordRemoverFactory().get_stop_words())
        df["Stopword_Removal"] =
df["Normalisasi"].apply(lambda x: [w for w in x if w
not in stopwords])

        # Stemming
        stemmer = StemmerFactory().create_stemmer()
        df["Stemming"] =
df["Stopword_Removal"].apply(lambda x:
[stemmer.stem(w) for w in x])

        # Label Encoding
        label_map = {"negatif": 0, "positif": 1}
        df["LABEL_NUM"] = df["LABEL"].map(label_map)

        st.subheader("Hasil Lengkap Preprocessing")
        st.dataframe(df[[
            "ULASAN",
            "Cleansing",
            "Tokenizing",
            "Normalisasi",
            "Stopword_Removal",
            "Stemming",
```

```
            "LABEL"
        ]].head())

        st.session_state["preprocessed"] = df
        st.session_state["ekstraksi_selesai"] = False

# Tab 2 : Fitur Ekstraksi
with tab2:
    st.subheader("Fitur Ekstraksi")

    if "preprocessed" not in st.session_state:
        st.warning("Jalankan Tab 1 terlebih dahulu")
        st.stop()

    df = st.session_state["preprocessed"]
    df["JOINED"] = df["Stemming"].apply(lambda x: "
".join(x))

    metode = st.radio("Pilih metode ekstraksi:",
["TF-IDF", "BoW"], horizontal=True)

    if st.button("Mulai Ekstraksi"):
        if metode == "TF-IDF":
            vectorizer =
TfidfVectorizer(max_features=5000)
        else:
            vectorizer =
CountVectorizer(max_features=5000)

        X = vectorizer.fit_transform(df["JOINED"])

        st.session_state["X"] = X
```

```
        st.session_state["y"] = df["LABEL_NUM"]
        st.session_state["vectorizer"] = vectorizer
        st.session_state["ekstraksi_selesai"] = True

        st.success(f"Ekstraksi {metode} selesai")

        # Preview Hasil Ekstraksi
        preview_df = pd.DataFrame(
            X.toarray(),
            columns=vectorizer.get_feature_names_out(
)
        ).head()

        st.subheader("Preview Hasil Ekstraksi")
        st.dataframe(preview_df)

# Tab 3 : Analisis Sentimen
with tab3:
    st.subheader("Analisis Sentimen")

    if not st.session_state.get("ekstraksi_selesai",
False):
        st.info("Lakukan ekstraksi fitur di Tab 2
terlebih dahulu")
        st.stop()

    X = st.session_state["X"]
    y = st.session_state["y"]
    df = st.session_state["preprocessed"].copy()

    # Train Model
    model = XGBClassifier(
```

```
            objective='binary:logistic',
            eval_metric='logloss',
            random_state=42,
            subsample=1.0,
            n_estimators=400,
            min_child_weight=1,
            max_depth=9,
            learning_rate=0.08,
            gamma=0.1,
            colsample_bytree=0.8,
            tree_method='hist',
            use_label_encoder=False
        )

        model.fit(X, y)

        # Prediksi
        y_pred_all = model.predict(X)
        df["Prediksi"] = pd.Series(y_pred_all).map({0:
"Negatif", 1: "Positif"})

        st.subheader("Hasil Prediksi")
        st.dataframe(df[["ULASAN", "Prediksi"]].head(10))

        # Diagram Batang
        st.subheader("Distribusi Sentimen")
        counts = df["Prediksi"].value_counts()

        fig, ax = plt.subplots()
        ax.bar(
            counts.index,
            counts.values,
```

```
        color=["red", "green"]
    )
    ax.set_xlabel("Sentimen")
    ax.set_ylabel("Jumlah")
    st.pyplot(fig)

    #WordCloud
    st.subheader("WordCloud Sentimen")

    col1, col2 = st.columns(2)

    with col1:
        st.markdown("WordCloud Positif")
        text_pos = " ".join([" ".join(x) for x in
df[df["Prediksi"] == "Positif"]["Stemming"]]) or
"positif"
        wc_pos = WordCloud(width=400, height=300,
background_color="white",
colormap="Greens").generate(text_pos)
        fig, ax = plt.subplots()
        ax.imshow(wc_pos)
        ax.axis("off")
        st.pyplot(fig)

    with col2:
        st.markdown("WordCloud Negatif")
        text_neg = " ".join([" ".join(x) for x in
df[df["Prediksi"] == "Negatif"]["Stemming"]]) or
"negatif"
        wc_neg = WordCloud(width=400, height=300,
background_color="white",
colormap="Reds").generate(text_neg)
```

```
        fig, ax = plt.subplots()
        ax.imshow(wc_neg)
        ax.axis("off")
        st.pyplot(fig)
```