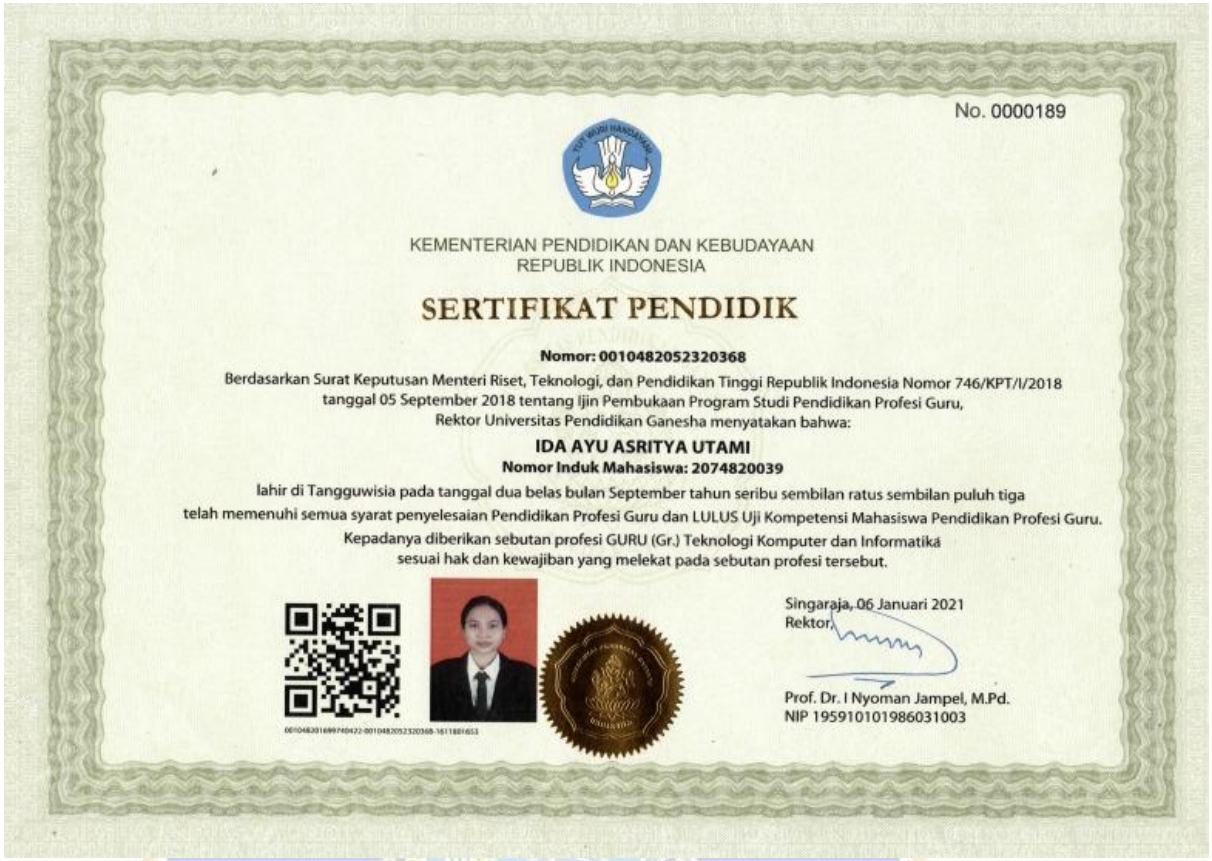
Lampiran 1. Riwayat Hidup

RIWAYAT HIDUP

Ida Ayu kadek Bintang Wijayanti lahir di Singaraja pada 5 Maret 2002. Penulis lahir dari pasangan suami istri, Bapak Ida Bagus Putu Adi Wijaya dan Ibu Luh Budi Wardani. Penulis berstatus Warga Negara Indonesia (WNI) dan beragama Hindu. Alamat tinggal penulis saat ini berada di Jalan Udayana No. 19 Singaraja, Kelurahan Kaliuntu, Kecamatan Buleleng, Kabupaten Buleleng, Provinsi Bali. Penulis menyelesaikan pendidikan dasar di SD Negeri 3 Banjar Jawa dan lulus pada tahun 2014. Kemudian penulis melanjutkan di SMP Lab Singaraja dan lulus pada tahun 2017. Setelah lulus dari jenjang SMP, kemudian penulis melanjutkan ke jenjang SMA di SMAN 4 Singaraja dan lulus pada tahun 2020. Setelah penulis lulus dari jenjang SMA, penulis melanjutkan pendidikan ke Perguruan Tinggi di Universitas Pendidikan Ganesha dengan mengambil Program Studi (S1) Sistem Informasi, Jurusan Teknik Informatika, Fakultas Teknik dan Kejuruan.

Lampiran 2. Sertifikat Pendidik



No. 0000189

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
REPUBLIK INDONESIA

SERTIFIKAT PENDIDIK

Nomor: 0010482052320368

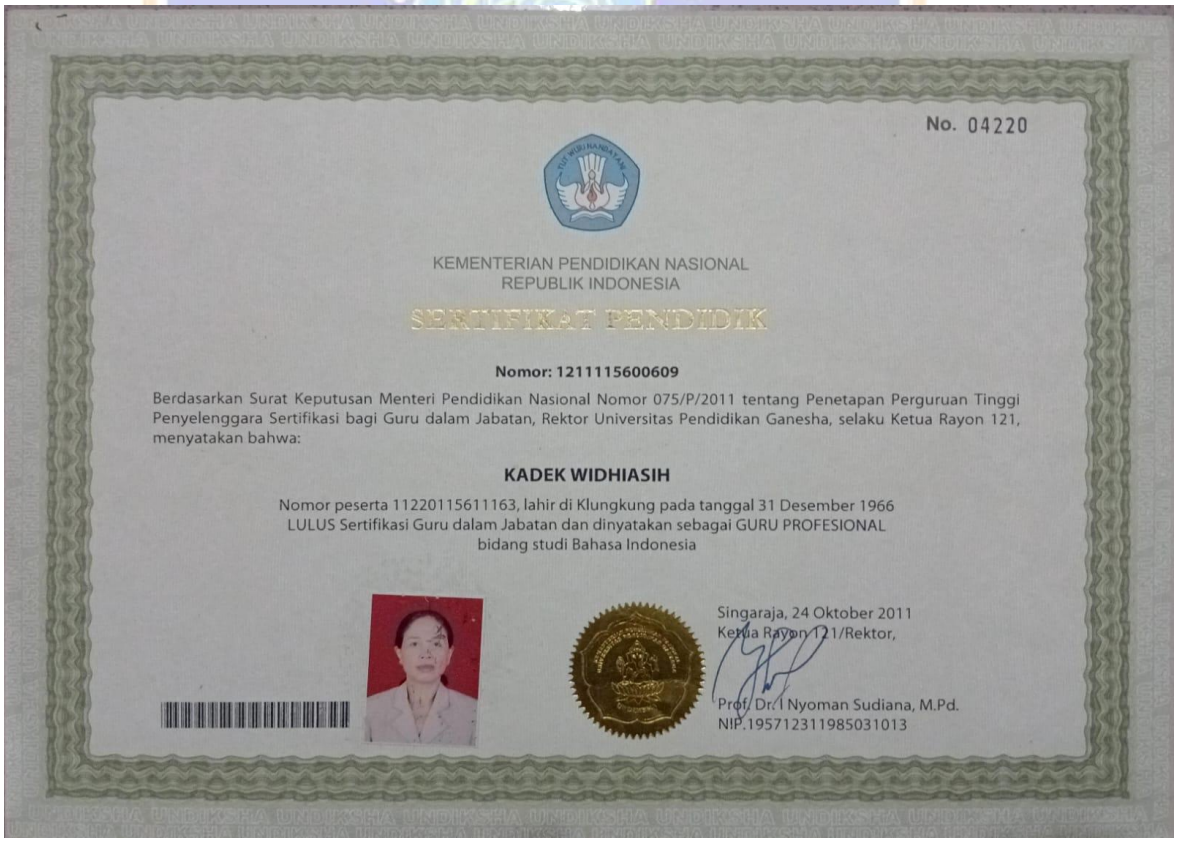
Berdasarkan Surat Keputusan Menteri Riset, Teknologi, dan Pendidikan Tinggi Republik Indonesia Nomor 746/KPT/I/2018 tanggal 05 September 2018 tentang Ijin Pembukaan Program Studi Pendidikan Profesi Guru, Rektor Universitas Pendidikan Ganesha menyatakan bahwa:

IDA AYU ASRITYA UTAMI
Nomor Induk Mahasiswa: 2074820039

lahir di Tangguwisia pada tanggal dua belas bulan September tahun seribu sembilan ratus sembilan puluh tiga telah memenuhi semua syarat penyelesaian Pendidikan Profesi Guru dan LULUS Uji Kompetensi Mahasiswa Pendidikan Profesi Guru. Kepadanya diberikan sebutan profesi GURU (Gr.) Teknologi Komputer dan Informatika sesuai hak dan kewajiban yang melekat pada sebutan profesi tersebut.

Singaraja, 06 Januari 2021
Rektor,

Prof. Dr. I Nyoman Jampel, M.Pd.
NIP 195910101986031003



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. 0002924

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
REPUBLIK INDONESIA

SERTIFIKAT PENDIDIK

Nomor: 0010482415620638

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

NI NENGAH SUDIANTARI
Nomor Induk Mahasiswa: 2374805033

lahir di Bangli pada tanggal enam belas bulan Juli tahun seribu sembilan ratus delapan puluh dua telah memenuhi semua syarat penyelesaian Pendidikan Profesi Guru dan LULUS Uji Kompetensi Mahasiswa Pendidikan Profesi Guru.

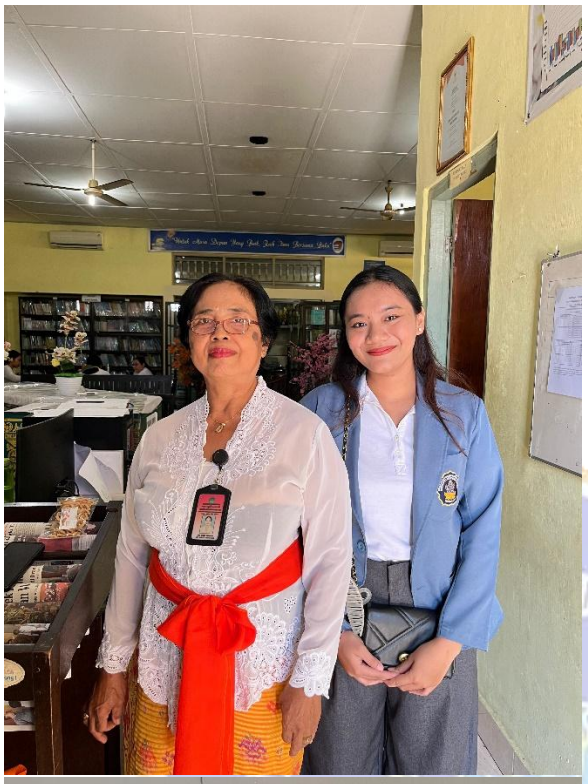
Kepadanya diberikan sebutan profesi GURU (Gr.) Bahasa Indonesia sesuai hak dan kewajiban yang melekat pada sebutan profesi tersebut.

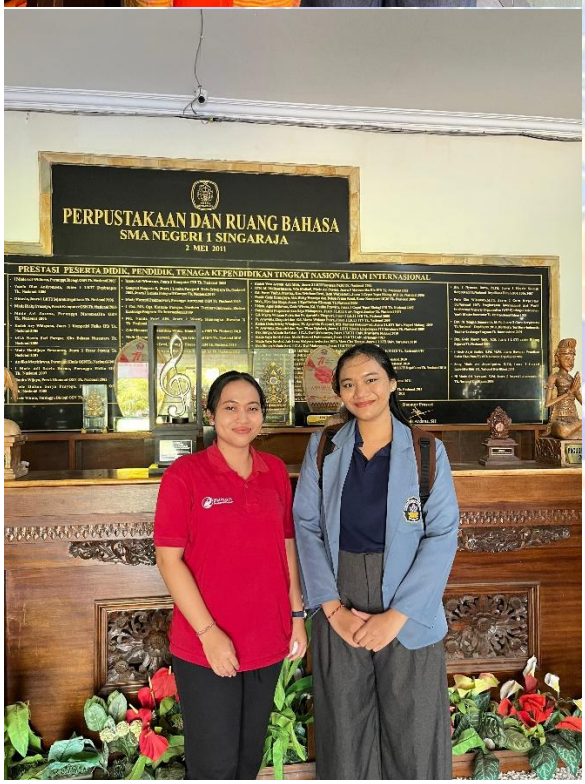
0010482015007010S8-0010482415620638-1717384193

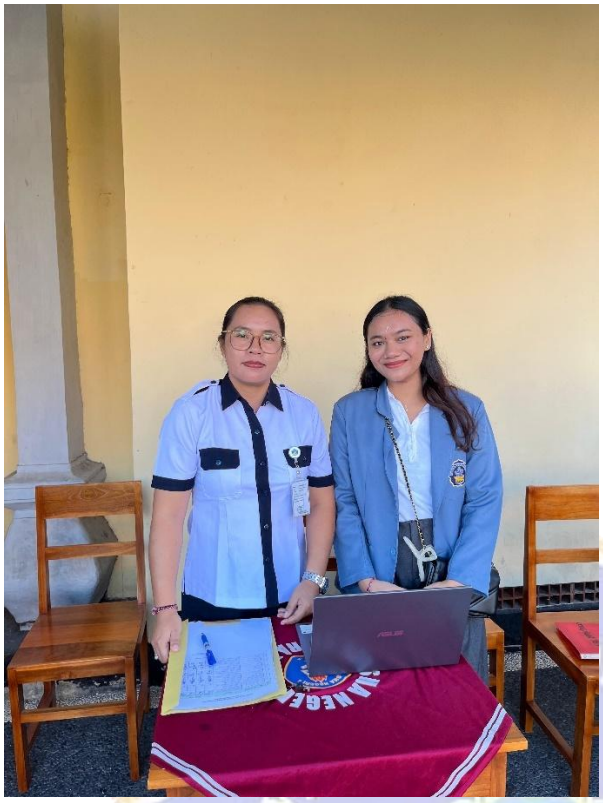
Singaraja, 20 Mei 2024
Rektor,

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

Lampiran 3. Foto Bersama Pakar (Guru Bahasa Indonesia)







Lampiran 4. Surat Validasi

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
Jalan Udayana, No. 11, Singaraja Bali Kode Pos 81116
Telepon (0362) 21541 Fax. (0362) 27561
Laman: https://undiksha.ac.id

SURAT KETERANGAN VALIDASI
LABELING DATASET

Yang bertanda tangan di bawah ini.

Nama : Kadek Widhiasih, S.Pd.
NIP : 196612311990702206

Menerangkan bahwa Mahasiswa Universitas Pendidikan Ganesha di bawah ini:

Nama : Ida Ayu Kadek Bintang Wijayanti
NIM : 2115091083
Prodi/Jurusan : S1 Sistem Informasi

Memang benar bahwa dataset yang sudah dilabeli telah divalidasi pada tanggal 12 Juni 2025. Demikian surat keterangan ini dibuat dengan sebenarnya untuk dapat digunakan sebagaimana mestinya.

Singaraja, 12 Juni 2025
Ahli Pakar,

Kadek Widhiasih, S.Pd.
NIP. 196612311990702206

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
Jalan Udayana, No. 11, Singaraja Bali Kode Pos 81116
Telepon (0362) 21541 Fax. (0362) 27561
Laman: https://undiksha.ac.id

SURAT KETERANGAN VALIDASI
LABELING DATASET

Yang bertanda tangan di bawah ini.

Nama : Ni Nengah Sudiantari, S.Pd.
NIP : 198207162009022001

Menerangkan bahwa Mahasiswa Universitas Pendidikan Ganesha di bawah ini:

Nama : Ida Ayu Kadek Bintang Wijayanti
NIM : 2115091083
Prodi/Jurusan : S1 Sistem Informasi

Memang benar bahwa dataset yang sudah dilabeli telah divalidasi pada tanggal 12 Juni 2025. Demikian surat keterangan ini dibuat dengan sebenarnya untuk dapat digunakan sebagaimana mestinya.

Singaraja, 12 Juni 2025
Ahli Pakar,

Ni Nengah Sudiantari, S.Pd.
NIP. 198207162009022001

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
Jalan Udayana, No. 11, Singaraja Bali Kode Pos 81116
Telepon (0362) 21541 Fax. (0362) 27561
Laman: https://undiksha.ac.id

SURAT KETERANGAN VALIDASI
LABELING DATASET

Yang bertanda tangan di bawah ini.

Nama : Ida Ayu Asritia Utami, S.Pd.
NIP : 199309122023212045

Menerangkan bahwa Mahasiswa Universitas Pendidikan Ganesha di bawah ini:

Nama : Ida Ayu Kadek Bintang Wijayanti
NIM : 2115091083
Prodi/Jurusan : S1 Sistem Informasi

Memang benar bahwa dataset yang sudah dilabeli telah divalidasi pada tanggal 12 Juni 2025. Demikian surat keterangan ini dibuat dengan sebenarnya untuk dapat digunakan sebagaimana mestinya.

Singaraja, 12 Juni 2025
Ahli Pakar,

Ida Ayu Asritia Utami, S.Pd.
NIP. 199309122023212045

Lampiran 5. Kode Program Proses *Crawling* Data

```
# 1. Set Twitter Auth Token
twitter_auth_token = 'd9fbcefc32cbb379c459749ceec558e071145dbf' # 1. Change this auth token with your own

# 2. Install pandas package
!pip install pandas # 2. Install pandas for data processing

# 3. Install Node.js (required for tweet-harvest)
!sudo apt-get update # 3. Update package list
!sudo apt-get install -y ca-certificates curl gnupg # 4. Install certificates, curl, gnupg
!sudo mkdir -p /etc/apt/keyrings # 5. Create keyrings directory
!curl -fsSL https://deb.nodesource.com/gpgkey/nodesource-repo.gpg.key | \
    sudo gpg --dearmor -o /etc/apt/keyrings/nodesource.gpg # 6. Download and save Node.js GPG key
!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 # 7. Add Node.js repository
!sudo apt-get update # 8. Update package list again
!sudo apt-get install nodejs -y # 9. Install Node.js
!node -v # 10. Check Node.js version

# 4. Crawl Data from Twitter using tweet-harvest
filename = 'ometv.csv' # 11. Output filename
search_keyword = 'ometv lang:id since:2020-01-01 until:2025-4-30' # 12. Search keyword with filters
limit = 6000 # 13. Limit data to crawl

# 5. Run tweet-harvest command
!npx -y tweet-harvest@2.6.1 \
-o "{filename}" \ # 14. Output file path
-s "{search_keyword}" \ # 15. Search keyword for crawling
--tab "LATEST" -l {limit} --token {twitter_auth_token} # 16. Additional options: latest tab, limit, and token

# 6. Import pandas for reading data
import pandas as pd # 17. Import pandas library

# 7. Set file path for reading the CSV
filename = '/content/tweets-data/ome.csv' # 18. File path (adjust if different in your drive)

# 8. Assign file path to variable
file_path = filename # 19. Assign path

# 9. Read CSV into pandas DataFrame
df = pd.read_csv(file_path, delimiter=",", encoding='utf-8', on_bad_lines='skip') # 20. Read CSV, skip bad lines

# 10. Display the DataFrame
display(df) # 21. Display crawled data for checking
```

Lampiran 6. Kode Program Proses *Cleaning* Data

```python
# 1. Install library yang dibutuhkan
!pip install emoji # 1. Install library emoji
!pip install langdetect # 2. Install library langdetect

# 2. Import library
import pandas as pd # 3. Import pandas
import re # 4. Import regex
import emoji # 5. Import emoji
from langdetect import detect, DetectorFactory # 6. Import language detection

DetectorFactory.seed = 0  # 7. Agar deteksi bahasa konsisten

# 3. BACA DATA
df = pd.read_csv('data_berlabel_awal.csv', encoding='latin1') # 8. Baca file data

print("=== DATA AWAL ===") # 9. Print judul data awal
print(df.head()) # 10. Tampilkan 5 data pertama
print(df.info()) # 11. Informasi data

# 4. HAPUS DATA MISSING & DUPLIKAT
df = df.dropna() # 12. Hapus data missing/null
df = df.drop_duplicates() # 13. Hapus data duplikat

# 5. HAPUS TWEET MENGANDUNG LINK
df = df[~df['tweet'].str.contains(r'http\S+|www\.\S+', na=False)] # 14. Hapus tweet mengandung link

# 6. HAPUS TWEET MENGANDUNG KATA IKLAN
pattern_iklan=r'RESULT|SHIO|JACKPOT|Para Pemenang|SYAIR|ANGKA' # 15. Pattern kata iklan
df = df[~df['tweet'].str.contains(pattern_iklan, case=False, na=False)] # 16. Hapus tweet iklan

# 7. HAPUS TWEET NON-BAHASA INDONESIA
def is_indonesian(text): # 17. Fungsi cek bahasa Indonesia
    try:
        return detect(text) == 'id'
    except:
        return False

df = df[df['tweet'].apply(is_indonesian)] # 18. Filter hanya bahasa Indonesia

# 8. HAPUS TWEET DENGAN SIMBOL BERULANG (!!!, ???, ***)
def has_repeated_symbols(text): # 19. Fungsi cek simbol berulang
    return bool(re.search(r'[\?\!\*\/\.\,]{3,}', text))

df = df[~df['tweet'].apply(has_repeated_symbols)] # 20. Hapus tweet dengan simbol berulang

# 9. CLEANING TEXT (MENTION, HASHTAG, EMOJI, ANGKA, TANDA BACA, SPASI GANDA)
def bersihkan_teks(text): # 21. Fungsi cleaning text
    text = re.sub(r'@\w+', '', text)  # 22. Hapus mention
    text = re.sub(r'http\S+|www\.\S+', '', text)  # 23. Hapus URL (jaga-jaga)
    text = re.sub(r'#\w+', '', text)  # 24. Hapus hashtag
    text = emoji.replace_emoji(text, replace='')  # 25. Hapus emoji
    text = re.sub(r'\d+', '', text)  # 26. Hapus angka
    text = re.sub(r'[^\w\s]', '', text)  # 27. Hapus tanda baca
    text = re.sub(r'\s+', ' ', text)  # 28. Spasi ganda jadi satu
    return text.strip() # 29. Hapus spasi awal/akhir

df['cleaning_data'] = df['tweet'].apply(bersihkan_teks) # 30. Terapkan cleaning ke kolom tweet

# 10. TAMPILKAN HASIL
print("\n=== DATA BERSIH ===") # 31. Judul data bersih
display(df[['tweet', 'cleaning_data']].head(10)) # 32. Tampilkan 10 data hasil cleaning

# 11. SIMPAN HASIL
df.to_csv('final_cleaning_data.csv', index=False) # 33. Simpan data hasil cleaning ke file
print("\n✅ Data cleaning selesai dan disimpan ke 'final_cleaning_data.csv'") # 34. Print selesai
```

Lampiran 7. Kode Program Proses Case Folding

```python
# 1. Import library pandas
import pandas as pd

#PROSES CASE FOLDING#

# 3. Membaca dataset final_cleaning_data.csv
df = pd.read_csv('final_cleaning_data.csv')

# 4. Fungsi case folding: ubah huruf menjadi huruf kecil
def case_folding(text):
    return str(text).lower()

# 5. Terapkan fungsi case folding ke kolom cleaning_data
df['case_folding'] = df['cleaning_data'].apply(case_folding)

# 6. Simpan hasilnya ke file baru (opsional)
df.to_csv('data_casefolding.csv', index=False)

# 7. Tampilkan hasil sebelum dan sesudah case folding (10 contoh)
from IPython.display import display
print("\n=== TABEL HASIL CASEFOLDING (SEBELUM & SESUDAH) ===") display(df[['cleaning_data', 'case_folding']].head(10))
```

Lampiran 8. Kode Program Proses *Stopword Removal*

```python
# 1. Import library yang dibutuhkan
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from IPython.display import display

# 2. Download stopwords bahasa Indonesia dari NLTK
nltk.download('stopwords')

# 3. Baca dataset hasil case folding
df = pd.read_csv('data_casefolding.csv')

# 4. Ambil stopwords bahasa Indonesia dari NLTK dan tambahkan kata tambahan
additional_stopwords = {
'aku','gue','gua','gw','eh','jg','anjir','anjirr','anjirrrr','sih','njir','bjir',
'wkwk','wkwkwkwk','wkkwkwkwk','nya','yang','yg','yuk','yukk','yokk','yok','ya',
'trs','tp','gw','pas','lg','tau','gitu','gtu','gt','lu','lo','pa','gak','aja',
'kek','ni','si','biar','tu','tuh','dah','najis','plis','pd','dri','dr','or','sm',
'bgt','bngt','bsa','udh','kalo','klo','mw','mo','prnh','sih','aduh','pada','juga','karena','atau','tapi','jadi','aku','gue','gw','gua','lu','org','huft','trus','trs','jadi'
}
stop_words = set(stopwords.words('indonesian')).union(additional_stopwords)

# 5. Fungsi untuk hapus stopword tanpa tokenisasi eksplisit
def remove_stopwords(text):
    text = str(text).lower()
    text = re.sub(r'[^\w\s]', '', text)
    words = text.split()
    filtered = [word for word in words if word not in stop_words]
    return ' '.join(filtered)

# 6. Terapkan fungsi ke kolom 'case_folding'
df['stopword_removal'] = df['case_folding'].apply(remove_stopwords)

# 7. Simpan hasil ke file baru
df.to_csv('data_stopword_removal.csv', index=False)

# 8. Tampilkan hasil sebelum dan sesudah stopword removal
print("✅ Stopword removal selesai!")
display(df[['case_folding', 'stopword_removal']].head(10))
```

Lampiran 9. Kode Program Proses *Tokenizing*

```python
# 1. Import library yang dibutuhkan
import re
import pandas as pd

# 2. Baca dataset hasil stopword removal
df = pd.read_csv('data_stopword_removal.csv')

# 3. Isi kolom stopword_removal dengan string kosong jika ada NaN
df['stopword_removal'] = df['stopword_removal'].fillna('')

# 4. Fungsi tokenize (contoh sederhana)
def tokenize(text):
    if not isinstance(text, str):
        return []
    return text.split()

# 5. Terapkan fungsi tokenize ke kolom 'stopword_removal'
df['tokens'] = df['stopword_removal'].apply(tokenize)

# 6. Simpan hasil ke file baru
df.to_csv('hasil_tokenizing.csv', index=False)

# 7. Tampilkan hasil sebelum dan sesudah tokenize
print("✅ Tokenizing selesai!")
display(df[['stopword_removal','tokens']].head(10))
```

Lampiran 10. Kode Program Proses *Normalization*

```python
# 1. Import library yang dibutuhkan
import pandas as pd
import json
import ast

# 2. Upload slang-indo.json dari lokal
from google.colab import files
uploaded = files.upload()

# 3. Baca dataset hasil tokenizing
df = pd.read_csv('hasil_tokenizing.csv')
df['tokens'] = df['tokens'].apply(ast.literal_eval)

# 4. Baca kamus slang dari file JSON
with open('slang-indo.json', 'r', encoding='utf-8') as f:
    slang_dict = json.load(f)

# 5. Fungsi normalisasi slang
def normalize_tokens(tokens):
    return [slang_dict.get(token.lower(), token) for token in tokens

# 6. Terapkan normalisasi ke kolom 'tokens'
df['normalized'] = df['tokens'].apply(normalize_tokens)

# 7. Simpan hasil ke file baru
df.to_csv('hasil_normalization.csv', index=False)

# 8. Tampilkan hasil sebelum dan sesudah normalisasi
print("✅ Normalized selesai!")
display(df[['tokens', 'normalized']].head(10))
```

Lampiran 11. Kode Program Proses *Stemming*

```python
# 1. Install dan import library yang dibutuhkan
!pip install PySastrawi
import pandas as pd
import ast
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

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

# 3. Baca file hasil normalisasi
df = pd.read_csv('hasil_normalization.csv')
df['normalized'] = df['normalized'].apply(ast.literal_eval)

# 4. Fungsi stemming untuk setiap token
def stem_tokens(tokens):
    return [stemmer.stem(token) for token in tokens]

# 5. Terapkan stemming ke kolom 'normalized'
df['stemmed'] = df['normalized'].apply(stem_tokens)

# 6. Simpan hasil ke file baru
df.to_csv('hasil_stemming.csv', index=False)

# 7. Tampilkan hasil sebelum dan sesudah stemming
print("✅ Stemming selesai!")
display(df[['normalized', 'stemmed']].head(10))
```

Lampiran 12. Kode Program Proses TF-IDF

```python
# 1. Import library yang dibutuhkan
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder

# 2. Baca dataset yang sudah pre-processing (kolom: 'stemmed', 'label')
df = pd.read_csv('hasil_stemming.csv')

# 3. TF-IDF dari kolom 'stemmed'
corpus = df['stemmed'].astype(str).tolist()
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(corpus)

# 4. Ubah hasil TF-IDF menjadi DataFrame
tfidf_df = pd.DataFrame(tfidf_matrix.toarray(), columns=vectorizer.get_feature_names_out())

# 5. Label Encoding dari kolom 'label'
le = LabelEncoder()
df['label_encoded'] = le.fit_transform(df['label'])

# 6. Print distribusi label asli
print("\nDistribusi label teks:")
print(df['label'].value_counts())

# 7. Gabungkan TF-IDF (fitur) + label_encoded (target)
final_df = tfidf_df.copy()
final_df['label'] = df['label_encoded']

# 8. Tampilkan 5 data pertama dari dataset final
print(final_df.head())

# 9. Simpan ke CSV jika perlu
final_df.to_csv('dataset_tfidf_label_baru.csv', index=False)

# 10. Menampilkan 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))

# 11. (Opsional) Lihat mapping label aslinya
label_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print("Mapping label ke angka:", label_mapping)
```

Lampiran 13. Kode Program *Naive Bayes* Fold-5 + *Confusion Matrix*

```python
# 1. Import library yang dibutuhkan
import numpy as np
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

 === NAIVE BAYES TANPA SMOTE ===

# 2. Load data
df = pd.read_csv('dataset_tfidf_label_baru.csv')
X = df.drop(columns=['label'])
y = df['label']

# 3. Tampilkan distribusi label di awal
print("=== Distribusi Label Awal ===")
print(y.value_counts())

# 4. Setup Stratified K-Fold
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# 5. List untuk menyimpan metrik setiap fold
macro_precisions, macro_recalls, macro_f1s, accuracies = [], [], [], []

# 6. Simpan semua label dan prediksi dari semua fold untuk report gabungan
all_y_true = []
all_y_pred = []

print("=== Naive Bayes Tanpa SMOTE + Confusion Matrix ===")

# 7. Loop K-Fold cross validation
for fold, (train_idx, test_idx) in enumerate(skf.split(X, y), 1):
    X_train, y_train = X.iloc[train_idx], y.iloc[train_idx]
    X_test, y_test = X.iloc[test_idx], y.iloc[test_idx]
    model = MultinomialNB()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
# 8. Hitung metrik per fold (macro avg & akurasi)
    report = classification_report(y_test, y_pred, output_dict=True)
    acc = accuracy_score(y_test, y_pred)

    macro_precisions.append(report['macro avg']['precision'])
    macro_recalls.append(report['macro avg']['recall'])
    macro_f1s.append(report['macro avg']['f1-score'])
    accuracies.append(acc)

# 9. Simpan label dan prediksi untuk report gabungan
    all_y_true.extend(y_test)
    all_y_pred.extend(y_pred)

# 10. === Classification Report Gabungan ===
print("\n=== Classification Report Gabungan (Tanpa SMOTE) ===")
print(classification_report(all_y_true, all_y_pred, digits=2))

# 11. === Rata-rata metrik ===
print("\n=== Rata-rata Hasil Evaluasi (Tanpa SMOTE) ===")
print(f"Precision: {np.mean(macro_precisions):.4f}")
print(f"Recall:    {np.mean(macro_recalls):.4f}")
print(f"F1-Score:  {np.mean(macro_f1s):.4f}")
print(f"Accuracy:  {np.mean(accuracies):.4f}")

# 12. === Confusion Matrix Gabungan ===
cm = confusion_matrix(all_y_true, all_y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap='Blues')
plt.title("Confusion Matrix (Tanpa SMOTE - Gabungan Semua Fold)") plt.show()
```

Lampiran 14. Kode Program *Naive Bayes* Fold-10 + *Confusion Matrix*

```python
# 1. Import library yang dibutuhkan
import numpy as np
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

#  === NAIVE BAYES TANPA SMOTE ===

# 2. Load data
df = pd.read_csv('dataset_tfidf_label_baru.csv')
X = df.drop(columns=['label'])
y = df['label']

# 3. Tampilkan distribusi label di awal
print("=== Distribusi Label Awal ===")
print(y.value_counts())

# 4. Setup Stratified K-Fold
skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# 5. List untuk menyimpan metrik setiap fold
macro_precisions, macro_recalls, macro_f1s, accuracies = [], [], [], []

# 6. Simpan semua label dan prediksi dari semua fold untuk report gabungan
all_y_true = []
all_y_pred = []
print("=== Naive Bayes Tanpa SMOTE + Confusion Matrix ===")

# 7. Loop K-Fold cross validation
for fold, (train_idx, test_idx) in enumerate(skf.split(X, y), 1):
    X_train, y_train = X.iloc[train_idx], y.iloc[train_idx]
    X_test, y_test = X.iloc[test_idx], y.iloc[test_idx]
    model = MultinomialNB()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

# 8. Hitung metrik per fold (macro avg & akurasi)
    report = classification_report(y_test, y_pred, output_dict=True)
    acc = accuracy_score(y_test, y_pred)

    macro_precisions.append(report['macro avg']['precision'])
    macro_recalls.append(report['macro avg']['recall'])
    macro_f1s.append(report['macro avg']['f1-score'])
    accuracies.append(acc)

# 9. Simpan label dan prediksi untuk report gabungan
    all_y_true.extend(y_test)
    all_y_pred.extend(y_pred)

# 10. === Classification Report Gabungan ===
print("\n=== Classification Report Gabungan (Tanpa SMOTE) ===")
print(classification_report(all_y_true, all_y_pred, digits=2))

# 11. === Rata-rata metrik ===
print("\n=== Rata-rata Hasil Evaluasi (Tanpa SMOTE) ===")
print(f"Precision: {np.mean(macro_precisions):.4f}")
print(f"Recall:    {np.mean(macro_recalls):.4f}")
print(f"F1-Score:  {np.mean(macro_f1s):.4f}")
print(f"Accuracy:  {np.mean(accuracies):.4f}")

# 12. === Confusion Matrix Gabungan ===
cm = confusion_matrix(all_y_true, all_y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap='Blues') # 40. Plot confusion matrix dengan cmap blue
plt.title("Confusion Matrix (Tanpa SMOTE - Gabungan Semua Fold)") plt.show()
```

Lampiran 15. Kode Program *Naive Bayes* SMOTE Fold-5 + *Confusion Matrix*

```python
# 1. Import library yang dibutuhkan
import numpy as np
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
import matplotlib.pyplot as plt
from imblearn.over_sampling import SMOTE

# 2. Load data
df = pd.read_csv('dataset_tfidf_label.csv')
X = df.drop(columns=['label'])
y = df['label']

# 3. Bersihkan target NaN
not_nan_mask = y.notna()
X = X[not_nan_mask]
y = y[not_nan_mask]

# 4. Target per label SESUDAH SMOTE
target_count = 2752

# 5. Pakai 1 kali SMOTE (split train test)
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
train_idx, test_idx = next(iter(skf.split(X, y)))
X_train, y_train = X.iloc[train_idx], y.iloc[train_idx]
X_test, y_test = X.iloc[test_idx], y.iloc[test_idx] # 18. Split test

# 6. SMOTE dengan target count per label
sampling_strategy = {label: target_count for label in y_train.unique()}
smote = SMOTE(sampling_strategy=sampling_strategy, random_state=42)
X_train_res, y_train_res = smote.fit_resample(X_train, y_train)

# 7. Print distribusi label setelah SMOTE
print("\n=== Distribusi Label DENGAN SMOTE ===")
print(y_train_res.value_counts())

# 8. Train model dengan data latih hasil SMOTE
model = MultinomialNB()
model.fit(X_train_res, y_train_res)

# 9. Prediksi di data latih hasil SMOTE
y_train_pred = model.predict(X_train_res)

# 10. Prediksi di data uji (yang asli, tanpa SMOTE)
y_test_pred = model.predict(X_test)

# 11. === Classification Report pada train set ===
print("\n=== Classification Report - Train (DENGAN SMOTE) ===")
print(classification_report(y_train_res, y_train_pred, digits=2))

# 12. === Rata-rata metrik ===
print("\n=== Rata-rata Hasil Evaluasi (DENGAN SMOTE) ===")
print(f"Precision: {np.mean(macro_precisions):.4f}")
print(f"Recall: {np.mean(macro_recalls):.4f}")
print(f"F1-Score: {np.mean(macro_f1s):.4f}")
print(f"Accuracy: {np.mean(accuracies):.4f}")

# 13. === Confusion Matrix pada data latih hasil SMOTE ===
labels = sorted(y_train_res.unique())
cm_train = confusion_matrix(y_train_res, y_train_pred, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm_train, display_labels=labels)
disp.plot(cmap='Blues')
plt.title("Confusion Matrix - Train (DENGAN SMOTE)")
plt.show()

# 14. Print total per label
print(f"Total per label seharusnya: {target_count} x {len(labels)} = {target_count * len(labels)}")
print(f"Total di confusion matrix (train): {cm_train.sum()}")
```

Lampiran 16. Kode Program *Naive Bayes* SMOTE Fold-10 + *Confusion Matrix*

```python
# 1. Import library yang dibutuhkan
import numpy as np
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
import matplotlib.pyplot as plt
from imblearn.over_sampling import SMOTE

# 2. Load data
df = pd.read_csv('dataset_tfidf_label.csv')
X = df.drop(columns=['label'])
y = df['label']

# 3. Bersihkan target NaN
not_nan_mask = y.notna()
X = X[not_nan_mask]
y = y[not_nan_mask]

# 4. Target per label SESUDAH SMOTE
target_count = 2752

# 5. Pakai 1 kali SMOTE (split train test)
skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
train_idx, test_idx = next(iter(skf.split(X, y)))
X_train, y_train = X.iloc[train_idx], y.iloc[train_idx]
X_test, y_test = X.iloc[test_idx], y.iloc[test_idx]

# 6. SMOTE dengan target count per label
sampling_strategy = {label: target_count for label in y_train.unique()}
smote = SMOTE(sampling_strategy=sampling_strategy, random_state=42)
X_train_res, y_train_res = smote.fit_resample(X_train, y_train)

# 7. Print distribusi label setelah SMOTE
print("\n=== Distribusi Label DENGAN SMOTE ===")
print(y_train_res.value_counts())

# 8. Train model dengan data latih hasil SMOTE
model = MultinomialNB()
model.fit(X_train_res, y_train_res)

# 9. Prediksi di data latih hasil SMOTE
y_train_pred = model.predict(X_train_res)

# 10. Prediksi di data uji (yang asli, tanpa SMOTE)
y_test_pred = model.predict(X_test)

# 11. === Classification Report pada train set ===
print("\n=== Classification Report - Train (DENGAN SMOTE) ===")
print(classification_report(y_train_res, y_train_pred, digits=2))

# 12. === Rata-rata metrik ===
# *Note*: macro_precisions, macro_recalls, macro_f1s, accuracies belum diinisialisasi di script ini.
print("\n=== Rata-rata Hasil Evaluasi (DENGAN SMOTE) ===")
print(f"Precision: {np.mean(macro_precisions):.4f}")
print(f"Recall:    {np.mean(macro_recalls):.4f}")
print(f"F1-Score:  {np.mean(macro_f1s):.4f}")
print(f"Accuracy:  {np.mean(accuracies):.4f}")

# 13. === Confusion Matrix pada data latih hasil SMOTE ===
labels = sorted(y_train_res.unique())
cm_train = confusion_matrix(y_train_res, y_train_pred, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm_train, display_labels=labels)
disp.plot(cmap='Blues')
plt.title("Confusion Matrix - Train (DENGAN SMOTE)")
plt.show()

# 14. Print total per label dan total confusion matrix
print(f"Total per label seharusnya: {target_count} x {len(labels)} = {target_count * len(labels)}")
print(f"Total di confusion matrix (train): {cm_train.sum()}")
```

Lampiran 17. Kode Program *Word Cloud*

```python
# 1. Import library
import pandas as pd
import re
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# 2. Baca data
df = pd.read_csv('hasil_stemming.csv')

# === POSITIF ===
# 3. Filter POSITIF
positif_df = df[df['label'] == 'positif']['stemmed'].astype(str)

# 4. Gabung dan bersihkan
positif_teks = " ".join(positif_df) jadi 1 string
positif_teks_bersih = re.sub(r"[^a-zA-Z\s]", " ", positif_teks)

# 5. Buat WordCloud POSITIF
wc = WordCloud(
    width=800,
    height=400,
    background_color='white',
    stopwords=None,
    max_words=100,
    collocations=False
).generate(positif_teks_bersih)

# 6. Tampilkan WordCloud POSITIF
plt.figure(figsize=(12, 6))
plt.imshow(wc, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud - Label Positif', fontsize=20)
plt.show()

# === NETRAL ===
# 7. Filter NETRAL
netral_df = df[df['label'] == 'netral']['stemmed'].astype(str)

# 8. Gabung dan bersihkan
netral_teks = " ".join(netral_df)
netral_teks_bersih = re.sub(r"[^a-zA-Z\s]", " ", netral_teks)

# 9. Buat WordCloud NETRAL
wc = WordCloud(
    width=800,
    height=400,
    background_color='white',
    stopwords=None,
    max_words=100,
    collocations=False
).generate(netral_teks_bersih)

# 10. Tampilkan WordCloud NETRAL
plt.figure(figsize=(12, 6))
plt.imshow(wc, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud - Label Netral', fontsize=20)
plt.show()

# === NEGATIF ===
# 11. Filter NEGATIF
negatif_df = df[df['label'] == 'negatif']['stemmed'].astype(str)

# 12. Gabung dan bersihkan
negatif_teks = " ".join(negatif_df)
negatif_teks_bersih = re.sub(r"[^a-zA-Z\s]", " ", negatif_teks)

# 13. Buat WordCloud NEGATIF
wc = WordCloud(
    width=800,
    height=400,
    background_color='white',
    stopwords=None,
    max_words=100,
    collocations=False
).generate(negatif_teks_bersih)

# 14. Tampilkan WordCloud NEGATIF
plt.figure(figsize=(12, 6))
plt.imshow(wc, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud - Label Negatif', fontsize=20)
plt.show()
```