RIWAYAT HIDUP

Trihana Santhi lahir di Singaraja pada tanggal 20 Mei 2003. Penulis lahir dari pasangan suami istri Bapak I Nyoman Surana dan Ibu Ni Gusti Ayu Putu Armini. Penulis berkebangsaan Indonesia dan beragama Hindu. Kini penulis beralamat di Jalan Parikesit 1 No. 5 Banjar Tegal, Kecamatan Buleleng, Kabupaten Buleleng, Provinsi Bali. Penulis menyelesaikan pendidikan dasar di SD Negeri 1 Banjar Jawa dan lulus pada tahun 2015. Kemudian penulis melanjutkan di SMP Negeri 2 Singaraja dan lulus pada tahun 2018. Pada tahun 2021, penulis lulus dari SMA Negeri 4 Singaraja dengan Jurusan Ilmu Pengetahuan Alam. Selanjutnya, penulis melanjutkan studi ke jenjang Sarjana (S1) di Universitas Pendidikan Ganesha, mengambil Program Studi Sistem Informasi di Jurusan Teknik Informatika.

LAMPIRAN

Lampiran 1. Surat Permohonan Pelabelan Data Guru Bahasa Indonesia

KEMENTERIAN PENDIDIKAN TINGGI,
SAINS, DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
Jalan Udayana Nomor 11 Singaraja Bali
Laman: http://ftk.undiksha.ac.id

Nomor : **1554**/UN48.11.1/KM/2025 Singaraja, 20 Juni 2025
Perihal : Surat Permohonan Pengambilan Data

Yth. Kepala SMP Negeri 6 Singaraja
c.q. Guru Bahasa indonesia (Putu Tri Noverawati, S. Pd)
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:

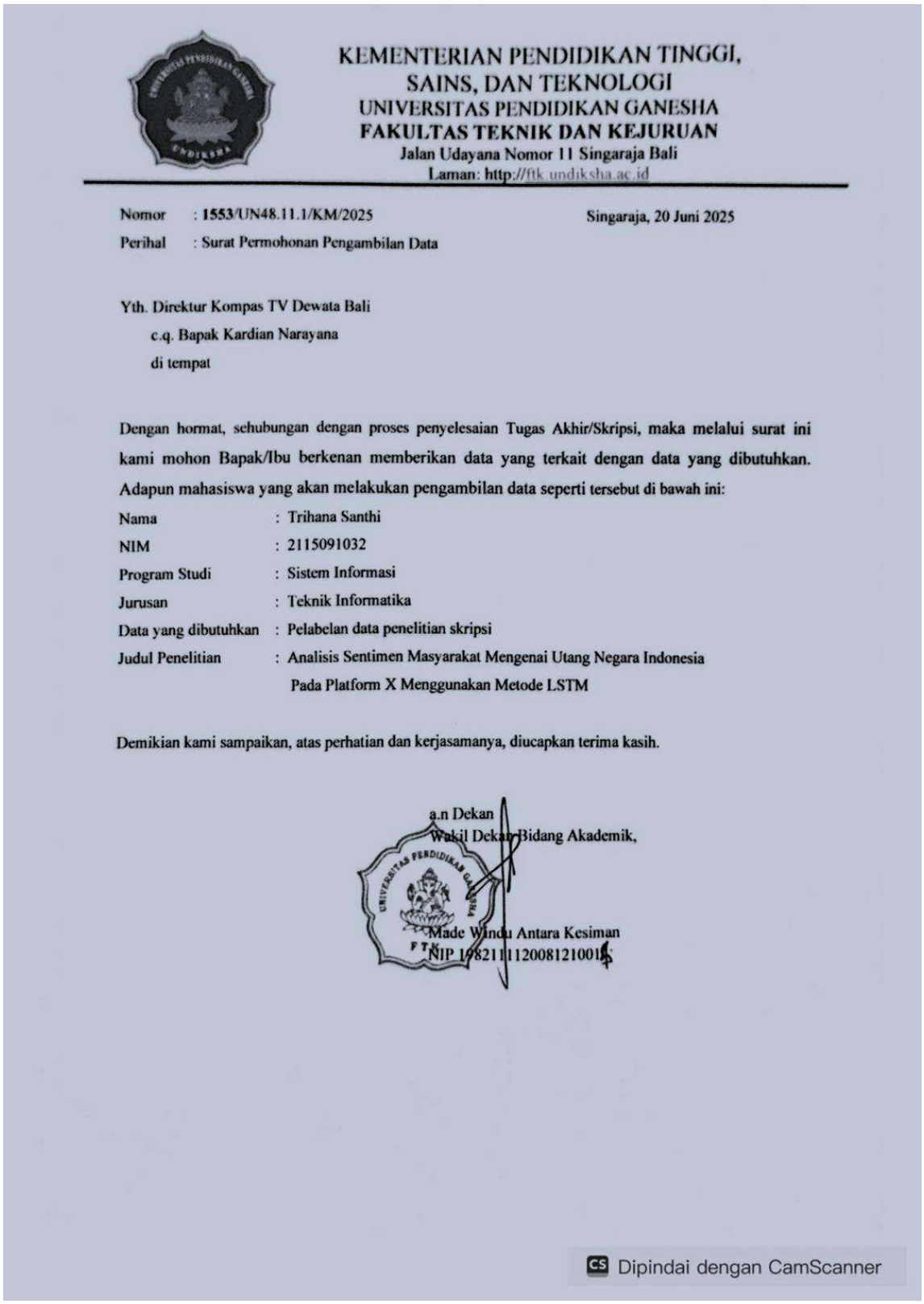
Nama	: Trihana Santhi
NIM	: 2115091032
Program Studi	: Sistem Informasi
Jurusan	: Teknik Informatika
Data yang dibutuhkan	: Pelabelan data penelitian skripsi
Judul Penelitian	: Analisis Sentimen Masyarakat Mengenai Utang Negara Indonesia Pada Platform X Menggunakan Metode LSTM

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

a.n Dekan
Wakil Dekan Bidang Akademik,

Made Windu Antara Kesiman
NIP 198211120081210015

Lampiran 2. Surat Permohonan Pelabelan Data Wartawan



KEMENTERIAN PENDIDIKAN TINGGI,
SAINS, DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
Jalan Udayana Nomor 11 Singaraja Bali
Laman: http://ftk.undiksha.ac.id

Nomor : 1553/UN48.11.1/KM/2025 Singaraja, 20 Juni 2025
Perihal : Surat Permohonan Pengambilan Data

Yth. Direktur Kompas TV Dewata Bali
c.q. Bapak Kardian Narayana
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	: Trihana Santhi
NIM	: 2115091032
Program Studi	: Sistem Informasi
Jurusan	: Teknik Informatika
Data yang dibutuhkan	: Pelabelan data penelitian skripsi
Judul Penelitian	: Analisis Sentimen Masyarakat Mengenai Utang Negara Indonesia Pada Platform X Menggunakan Metode LSTM

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

a.n Dekan
Wakil Dekan Bidang Akademik,

Made Windu Antara Kesiman
NIP 196211112008121001

Lampiran 3. Surat Pernyataan Validasi Pelabelan Data Guru Bahasa Indonesia

SURAT PERNYATAAN HASIL VALIDASI DATA

Yang bertanda tangan di bawah ini:

Nama : Putu Tri Noverawati, S.Pd.

Jabatan : Guru Bahasa Indonesia

Instansi : SMP Negeri 6 Singaraja

Dengan ini menyatakan bahwa saya telah melakukan validasi terhadap data *tweet* pada platform X mengenai *tweet* utang negara yang telah diberikan oleh:

Nama Mahasiswa : Trihana Santhi

NIM : 2115091032

Program Studi : S1 Sistem Informasi

Universitas : Universitas Pendidikan Ganesha

Data tersebut terdiri dari sejumlah kalimat *tweet* yang telah diberi label sentimen positif, netral, dan negatif oleh mahasiswa. Berdasarkan hasil penelaahan terhadap konteks kalimat, diksi, serta makna yang terkandung di dalam *tweet* tersebut, saya menyatakan bahwa pelabelan yang dilakukan sudah sesuai dengan kaidah Bahasa Indonesia dan dapat diterima secara makna dan konteks dalam analisis sentimen.

Demikian surat ini dibuat untuk digunakan sebagaimana mestinya.

Buleleng, 19 Agustus 2025

Guru Bahasa Indonesia,

Putu Tri Noverawati, S.Pd.

Lampiran 4. Surat Pernyataan Validasi Pelabelan Data Wartawan

SURAT PERNYATAAN HASIL VALIDASI DATA

Yang bertanda tangan di bawah ini:

Nama : Kardian Narayana

Jabatan : Wartawan

Instansi : KompasTv Dewata Bali

Dengan ini menyatakan bahwa saya telah melakukan validasi terhadap data *tweet* pada platform X mengenai *tweet* utang negara yang telah diberikan oleh:

Nama Mahasiswa : Trihana Santhi

NIM : 2115091032

Program Studi : S1 Sistem Informasi

Universitas : Universitas Pendidikan Ganesha

Data tersebut terdiri dari sejumlah kalimat *tweet* yang telah diberi label sentimen positif, netral, dan negatif oleh mahasiswa. Berdasarkan hasil penelaahan terhadap konteks kalimat, diksi, serta makna yang terkandung di dalam *tweet* tersebut, saya menyatakan bahwa pelabelan yang dilakukan sudah sesuai dengan kaidah Bahasa Indonesia dan dapat diterima secara makna dan konteks dalam analisis sentimen.

Demikian surat ini dibuat untuk digunakan sebagaimana mestinya.

Buleleng, 30 Agustus 2025

Wartawan KompasTv,

Kardian Narayana.

Lampiran 5. Dokumentasi Validasi Pelabelan Data Guru Bahasa Indonesia

Lampiran 6. Dokumentasi Validasi Pelabelan Data Wartawan

Lampiran 7. Implementasi *Crawling Data*

```
import datetime
import subprocess
import time
import pandas as pd
import os
start_date = datetime.date(2019, 10, 20)
end_date = datetime.date(2024, 10, 20)
limit = 10000
hashtag = '#bendunganjatigede'
lang = 'id'
twitter_auth_token = '...'
jeda = 60
tweet_mode = 'LATEST'
```

```
# Fungsi bantu untuk tanggal
def next_month(d):
    if d.month == 12:
        return datetime.date(d.year + 1, 1, 1)
    else:
        return datetime.date(d.year, d.month + 1, 1)
# Loop crawling per bulan
current_date = start_date
while current_date <= end_date:
    next_date = next_month(current_date)
    if next_date > end_date:
        next_date = end_date + datetime.timedelta(days=1)
    filename = f'tweet_{limit}.csv'  # tanpa tanggal
    search_keyword = f'{hashtag} since:{current_date}
until:{next_date} lang:{lang}'
    command = [
        'npx', '-y', 'npx', 'tweet-harvest@latest',
        '-o', filename,
        '-s', search_keyword,
        '--tab', tweet_mode,
        '-l', str(limit),
        '--token', twitter_auth_token ]
    subprocess.run(command)
    if os.path.exists(filename):
        df = pd.read_csv(filename)
        if 'text' in df.columns:
            df[['text']].to_csv(filename, index=False)
    time.sleep(jeda)
    current_date = next_date
```

Lampiran 8. Implementasi *Pre-Processing*

1. *Cleaning Data*

```
import re
import pandas as pd
# Cleaning Function
def cleaning(text):
text = re.sub(r'http\S+|www.\S+', ' ', str(text))   # hapus URL
text = re.sub(r'@\w+|#\w+', ' ', text) # hapus mention/hashtag
text = re.sub(r'[^a-zA-Z\s]', ' ', text) # hapus angka, tanda baca,
emot
df["cleaning"] = df["full_text"].apply(cleaning)
print(df[["full_text", "cleaning"]].head())
```

2. *Case Folding*

```
df["casefolding"] = df["cleaning"].str.lower()
print(df[["cleaning", "casefolding"]].head())
```

3. *Normalize*

```
import pandas as pd
import re
# === 1. Load data CSV ===
slang_csv = pd.read_csv(" slang.csv")
# === 2. Load data TXT ===
slang_txt = {}
```

```
with open("combined_slang_words.txt", "r") as f:
    for line in f:
        parts = line.strip().split(":", 1)
        if len(parts) == 2:
            slang_txt[parts[0].strip().lower()]
parts[1].strip().lower()
# === 3. Load data Excel ===
slang_excel = pd.read_excel("data_normalize.xlsx")
# === 4. Gabungkan kamus (TXT → CSV → Excel) ===
slang_dict = {}
slang_dict.update(slang_txt)
slang_dict.update(dict(zip(
    slang_csv["slang"].astype(str).str.strip().str.lower(),
    slang_csv["formal"].astype(str).str.strip().str.lower() )))
slang_dict.update(dict(zip(
    slang_excel["tidak baku"].astype(str).str.strip().str.lower(),
    slang_excel["baku"].astype(str).str.strip().str.lower() )))
print(f"Total entri kamus gabungan: {len(slang_dict)}")

# === 5. Fungsi Normalisasi ===
def normalize(text, kamus):
    words = re.findall(r"\w+", text.lower())  # ambil kata alfanumerik
saja
    return " ".join([kamus.get(w, w) for w in words])
# === 6. Terapkan ke DataFrame ===
df["normalize"] = df["casefolding"].apply(lambda x: normalize(str(x),
slang_dict))
print(df[["casefolding", "normalize"]].head())
# === 7. Analisis Sumber Normalisasi ===
sumber_dict = {}
# tandai dari TXT
for k in slang_txt.keys():
    sumber_dict[k] = "TXT"
# tandai dari CSV
for k in slang_csv["slang"].astype(str).str.strip().str.lower():
    sumber_dict[k] = "CSV"
# tandai dari Excel
for k in slang_excel["tidak
baku"].astype(str).str.strip().str.lower():
    sumber_dict[k] = "Excel"
# cek kata yang berhasil diganti
def cek_sumber(text, kamus, sumber):
    words = re.findall(r"\w+", text.lower())
    hasil = []
    for w in words:
        if w in kamus and kamus[w] != w:
            hasil.append((w, kamus[w], sumber.get(w, "Tidak
Diketahui")))
    return hasil
df["cek_normalisasi"] = df["casefolding"].apply(lambda x:
cek_sumber(str(x), slang_dict, sumber_dict))
print(df[["casefolding", "normalize", "cek_normalisasi"]].head())
```

4. *Tokenize*

```
df["tokenize"] = df["normalize"].apply(lambda x: x.split())
print(df[["normalize", "tokenize"]].head())
```

5. *Stopword Removal*

```
from Sastrawi.StopWordRemover.StopWordRemoverFactory import
StopWordRemoverFactory
import pandas as pd
# === 1. Stopword bawaan Sastrawi ===
stop_factory = StopWordRemoverFactory()
stopwords_sastrawi = set(stop_factory.get_stop_words())
# === 2. Fungsi Stopword Removal ===
def stopword_removal(tokens):
    return [t for t in tokens if t not in stopwords_sastrawi]
# === 3. Terapkan ke dataframe ===
df["stopword"] = df["tokenize"].apply(stopword_removal)
print(df[["tokenize", "stopword"]].head())
```

6. *Stemming*

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
stemmer = StemmerFactory().create_stemmer()
def stemming(tokens):
    return [stemmer.stem(t) for t in tokens]
df["stemming"] = df["stopword"].apply(stemming)
print(df[["stopword", "stemming"]].head())
```

Lampiran 9. Implementasi Ekstraksi Fitur TF-IDF

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
# Load data hasil preprocessing
path = "3a. Pre_processing.xlsx"
df = pd.read_excel(path)
print("Data awal:")
print(df.head())

texts = df['stemming'].astype(str)
# TF-IDF dengan max_features=2000
tfidf = TfidfVectorizer(max_features=2000)
X = tfidf.fit_transform(texts)
# Simpan hasil TF-IDF ke DataFrame + Label
tfidf_df = pd.DataFrame(X.toarray(),
columns=tfidf.get_feature_names_out())
tfidf_df['Label'] = df['Label'].values

# Simpan ke Excel
output_path = "4a. tf_idf.xlsx"
tfidf_df.to_excel(output_path, index=False)

print(tfidf_df.head())
print (f"\nHasil disimpan di: {output_path}")
```

Lampiran 10. Implementasi Model LSTM

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.metrics import (
    accuracy_score, precision_recall_fscore_support,
    confusion_matrix)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, LSTM, Reshape

results_summary = []
label_encoder = None
y_encoded = None

def create_model(input_dim, output_dim):
    """Membuat model Keras dengan LSTM"""
    model = Sequential()
    model.add(Reshape((1, input_dim), input_shape=(input_dim,)))
    model.add(LSTM(64))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(output_dim, activation='softmax'))
    model.compile(
        optimizer='adam',
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'] )
    return model
model = create_model(input_dim=X.shape[1],
output_dim=len(np.unique(y_encoded)))
model.summary()
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score,
precision_recall_fscore_support, confusion_matrix,
classification_report
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.callbacks import TensorBoard, EarlyStopping
import tensorflow as tf
from datetime import datetime
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

def train_and_evaluate_all_combinations(X, y, epochs_list=[100],
batch_size_list=[32, 64], n_splits=5, random_state=42):
    global results_summary, label_encoder, y_encoded
    label_encoder = LabelEncoder()
    y_encoded = label_encoder.fit_transform(y)
    summary_data = []
    timestamp = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
    main_folder = f"adasyn_training_results_{timestamp}"
    os.makedirs(main_folder, exist_ok=True)
    for epochs in epochs_list:
        for batch_size in batch_size_list:
            fold_results, subfolder = _train_single_combination(
                X, y_encoded, epochs, batch_size, n_splits,
random_state, main_folder            )
```

```
            best_fold = max(fold_results, key=lambda fr:
fr['f1_score'])
# === Rata-rata epoch terbaik antar fold ===
avg_best_epoch = int(np.mean([fr['best_epoch'] for fr in
fold_results]))
# === Confusion Matrix Fold Terbaik ===
cm = confusion_matrix(best_fold['y_val'], best_fold ['y_pred'])
cm_df = pd.DataFrame(cm,index=label_encoder.classes_,
columns=label_encoder.classes_)
cm_path_xlsx = os.path.join(subfolder, "folds_metrics.xlsx")
            with pd.ExcelWriter(cm_path_xlsx, engine="openpyxl",
mode="a", if_sheet_exists="replace") as writer:
cm_df.to_excel(writer, sheet_name=f"cm_best_fold", index=True)
plt.figure(figsize=(6, 5)) sns.heatmap(cm, annot=True, fmt="d",
cmap="Blues", xticklabels=label_encoder.classes_,
yticklabels=label_encoder.classes_)
  plt.xlabel("Predicted")
  plt.ylabel("Actual")
  plt.title(f"Confusion Matrix (Best Fold {best_fold ['fold']})")
  plt.tight_layout()
  plt.savefig(os.path.join(subfolder,"confusion_matrix_best_fold.png")
)plt.close()
  row_data = {'epochs_set': epochs,'best_epoch_avg':
avg_best_epoch,'batch_size': batch_size,'accuracy':
best_fold['accuracy'],'precision_macro':
best_fold['precision'],'recall_macro': best_fold['recall'],
'f1_score_macro': best_fold['f1_score']}
  report = classification_report(best_fold['y_val'],
best_fold['y_pred'], target_names=label_encoder.classes_,
                output_dict=True)
for class_name in label_encoder.classes_:
                row_data[f'precision_{class_name}'] =
report[class_name]['precision']
                 row_data[f'recall_{class_name}'] =
report[class_name]['recall']
                 row_data[f'f1_score_{class_name}'] =
report[class_name]['f1-score']
summary_data.append(row_data)
# === Simpan summary kombinasi ===
    df_summary = pd.DataFrame(summary_data)
    cols = ['epochs_set', 'best_epoch_avg', 'batch_size', 'accuracy',
'precision_macro', 'recall_macro', 'f1_score_macro']
    for class_name in label_encoder.classes_:
        cols.extend([f'precision_{class_name}',
f'recall_{class_name}', f'f1_score_{class_name}'])
    df_summary = df_summary[cols]
    df_summary.to_excel(os.path.join(main_folder,
"summary_results.xlsx"), index=False)
    print(f"Semua hasil tersimpan di folder: {main_folder}")
def _train_single_combination(X, y_encoded, epochs, batch_size,
n_splits, random_state, main_folder):
    kf = KFold(n_splits=n_splits, shuffle=True,
random_state=random_state)
    fold_results = [] all_histories = []
    all_y_val = np.array([]) all_y_pred = np.array([])
```

```
    subfolder = os.path.join(main_folder,
f"epochs_{epochs}_batch_{batch_size}") os.makedirs(subfolder,
exist_ok=True)
    tb_folder = os.path.join(subfolder, "tensorboard_logs")
    os.makedirs(tb_folder, exist_ok=True)
for fold, (train_idx, val_idx) in enumerate(kf.split(X), 1):
        print(f"\n--- Epochs: {epochs}, Batch Size: {batch_size},
Fold: {fold} ---")
X_train, X_val = X[train_idx], X[val_idx] y_train, y_val =
y_encoded[train_idx], y_encoded[val_idx]

        model = create_model(input_dim=X.shape[1],
output_dim=len(np.unique(y_encoded)))

        log_dir = os.path.join(tb_folder, f"fold_{fold}")
        tensorboard_cb = TensorBoard(log_dir=log_dir,
histogram_freq=1)

 early_stopping_cb = EarlyStopping( monitor='val_loss',
patience=10,verbose=1, restore_best_weights=True)
history = model.fit( X_train, y_train, validation_data =(X_val,
y_val),
  epochs=epochs, batch_size=batch_size, verbose=1,
callbacks=[tensorboard_cb, early_stopping_cb] )

   history_dict = history.history
 best_epoch = np.argmin(history.history['val_loss']) + 1 print(f"Fold
{fold} berhenti di epoch terbaik: {best_epoch}")

  y_pred = np.argmax(model.predict(X_val), axis=1)
       acc = accuracy_score(y_val, y_pred)

        precision, recall, f1, _ =
precision_recall_fscore_support(y_val, y_pred, average='macro',
zero_division=0)

        all_y_val = np.concatenate([all_y_val, y_val])
        all_y_pred = np.concatenate([all_y_pred, y_pred])
        all_histories.append(history_dict)

fold_result = {'fold': fold,'best_epoch': best_epoch,
   'accuracy': acc,'precision': precision,'recall': recall,
   'f1_score': f1,'history': history_dict,'y_val': y_val,
   'y_pred': y_pred}
        fold_results.append(fold_result)

        model.save(os.path.join(subfolder, f"model_fold{fold}.h5"))
        hist_df = pd.DataFrame(history_dict)
        hist_df.to_excel(os.path.join(subfolder,
f"history_fold{fold}.xlsx"), index=False)
  plt.figure(figsize=(12, 5)) plt.subplot(1, 2, 1)
  plt.plot(hist_df['loss'], label='Training Loss')
  plt.plot(hist_df['val_loss'], label='Validation Loss')
  plt.title(f'Loss (Fold {fold}) - Best Epoch: {best_epoch}')
        plt.xlabel('Epoch') plt.ylabel('Loss')plt.legend()

  plt.subplot(1, 2, 2)
```

```
  plt.plot(hist_df['accuracy'], label='Training Accuracy')
  plt.plot(hist_df['val_accuracy'], label='Validation Accuracy')
  plt.title(f'Accuracy (Fold {fold}) - Best Epoch:
{best_epoch}')plt.xlabel('Epoch') plt.ylabel('Accuracy')
  plt.legend()

   plt.tight_layout()plt.savefig(os.path.join(subfolder,
f"training_curves_fold{fold}.png"))
plt.close()

    combined_cm = confusion_matrix(all_y_val, all_y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(combined_cm, annot=True, fmt='d', cmap='Blues',
xticklabels=label_encoder.classes_,
    yticklabels=label_encoder.classes_)
    plt.title(f'Combined Confusion Matrix')
    plt.xlabel('Predicted Label')
    plt.ylabel('Actual Label')
    plt.savefig(os.path.join(subfolder,
"combined_confusion_matrix.png"))
    plt.close()
    class_report = classification_report(all_y_val,
all_y_pred,target_names=label_encoder.classes_,
    output_dict=True, zero_division=0)
    df_class_report = pd.DataFrame(class_report).transpose()
    excel_path = os.path.join(subfolder,"folds_metrics.xlsx")
    df_folds = pd.DataFrame([{'fold': fr['fold'],
    'best_epoch': fr['best_epoch'],
    'accuracy': fr['accuracy'],'precision': fr['precision'],
    'recall': fr['recall'],'f1_score': fr['f1_score']}
    for fr in fold_results])
with pd.ExcelWriter(excel_path, engine="openpyxl") as writer:
        df_folds.to_excel(writer, sheet_name="metrics_per_fold",
index=False)
        df_class_report.to_excel(writer,
sheet_name="combined_classification_report")
        pd.DataFrame(combined_cm, index=label_encoder.classes_,
columns=label_encoder.classes_).to_excel(writer,
sheet_name="combined_confusion_matrix")
return fold_results, subfolder
```

Lampiran 11. Implementasi *Resampling*

```
from sklearn.feature_extraction.text import TfidfVectorizer
from imblearn.over_sampling import SMOTE, ADASYN

# TF-IDF
tfidf = TfidfVectorizer(max_features=2000)
X = tfidf.fit_transform(df['stemming'])
y = df['Label']
# SMOTE
smote = SMOTE(random_state=42)
X_smote, y_smote = smote.fit_resample(X, y)
# Gabungkan ke DataFrame
df_smote = pd.DataFrame(X_smote.toarray(),
columns=tfidf.get_feature_names_out())
df_smote['Label'] = y_smote
```

```
# Simpan & print
df_smote.to_excel("output_smote.xlsx", index=False)
print("Hasil SMOTE:\n", df_smote['Label'].value_counts())
# ADASYN
adasyn = ADASYN(random_state=42)
X_adasyn, y_adasyn = adasyn.fit_resample(X, y)

df_adasyn = pd.DataFrame(X_adasyn.toarray(),
columns=tfidf.get_feature_names_out())
df_adasyn['Label'] = y_adasyn
df_adasyn.to_excel("output_adasyn.xlsx", index=False)
print("Hasil ADASYN:\n", df_adasyn['Label'].value_counts())
from imblearn.under_sampling import RandomUnderSampler
rus = RandomUnderSampler(random_state=42)
X_rus, y_rus = rus.fit_resample(X.values.reshape(-1,1), y)

df_rus_seq = pd.DataFrame({'stemming': X_rus.flatten(), 'Label':
y_rus})
df_rus_seq.to_excel("output_under.xlsx", index=False)
print("Hasil Undersampling tanpa TF-IDF:\n",
df_rus_seq['Label'].value_counts())
```

Lampiran 12. Implementasi GUI

```
import streamlit as st
import pickle
import numpy as np
import pandas as pd
from tensorflow.keras.models import load_model
from preprocessing import TextPreprocessor
import warnings
warnings.filterwarnings('ignore')

# ===== KONFIGURASI HALAMAN =====
st.set_page_config(
    page_title="Analisis Sentimen Utang Negara", page_icon="📊",
    layout="wide", initial_sidebar_state="expanded")
# ===== CUSTOM CSS =====
st.markdown("""
    <style>
    .main-header {
        font-size: 2.5rem; font-weight: bold; color: #1E88E5;
        text-align: center; margin-bottom: 1rem;}
    .prediction-box {
        padding: 20px; border-radius: 10px; margin: 20px 0;
        text-align: center; font-size: 1.5rem; font-weight: bold;}
    .positive {
        background-color: #4CAF50; color: white;}
    .negative {
        background-color: #F44336; color: white;}
    .neutral {
        background-color: #9E9E9E;color: white;}
    .confidence-score {
        font-size: 1.2rem; color: #555; margin-top: 10px;}
    .info-box {
        background-color: #E3F2FD; padding: 15px;
        border-radius: 8px; border-left: 5px solid #1E88E5;
```

```
        margin: 10px 0;} </style>
""", unsafe_allow_html=True)
# ===== LOAD MODEL DAN ARTIFACTS =====
@st.cache_resource
def load_artifacts():
    """Load model, vectorizer, dan label encoder"""
    try:
        model = load_model('model_best.h5')
        with open('tfidf_vectorizer.pkl', 'rb') as f:
            tfidf_vectorizer = pickle.load(f)
        with open('label_encoder.pkl', 'rb') as f:
            label_encoder = pickle.load(f)
        preprocessor = TextPreprocessor()
    return model, tfidf_vectorizer, label_encoder, preprocessor
    except Exception as e:
        st.error(f"❌ Error loading artifacts: {str(e)}")
        st.info("Pastikan file berikut ada dalam folder yang sama:
model_best.h5, tfidf_vectorizer.pkl, label_encoder.pkl")
        return None, None, None, None
# Load artifacts
model, tfidf_vectorizer, label_encoder, preprocessor =
load_artifacts()
# ===== FUNGSI PREDIKSI =====
def predict_sentiment(text):
    """ Fungsi untuk memprediksi sentimen dari teks input
    Args:
        text (str): Teks tweet yang akan diprediksi
    Returns:
        tuple: (predicted_class, confidence_score,
all_probabilities)"""
    # 1. Preprocessing
    processed_text = preprocessor.preprocess(text)
    # 2. Transform dengan TF-IDF
    tfidf_features =
tfidf_vectorizer.transform([processed_text]).toarray()
    # 3. Prediksi dengan model
    prediction_proba = model.predict(tfidf_features, verbose=0)
    # 4. Get predicted class
    predicted_class_idx = np.argmax(prediction_proba, axis=1)[0]
    predicted_class =
label_encoder.inverse_transform([predicted_class_idx])[0]
    # 5. Get confidence score
    confidence_score = prediction_proba[0][predicted_class_idx] *
100
    return predicted_class, confidence_score, prediction_proba[0]
# ===== HEADER =====
st.markdown('<p class="main-header">📊 Sentimen Analisis Utang
Negara</p>', unsafe_allow_html=True)
# ===== SIDEBAR =====
with st.sidebar:
    st.image("https://img.icons8.com/fluency/96/000000/bar-chart.png",
width=80)
    st.title("ℹ️ Informasi")
    st.markdown("""
    <div class="info-box">
    <b>Model:</b> LSTM + TF-IDF<br>
```

```
    <b>Pre-processing:</b>
    <ul>
        <li>Cleaning Data</li>
        <li>Case Folding</li>
        <li>Normalisasi</li>
        <li>Tokenisasi</li>
        <li>Stopword Removal</li>
        <li>Stemming</li> </ul>
    <b>Features:</b> TF-IDF (max 2000)<br>
    <b>Training:</b> K-Fold (k=5), Early Stopping<br>
    <b>Batch Size:</b> 32 </div>
    """, unsafe_allow_html=True)
    st.markdown("---")
    st.markdown("### 🎯 Cara Penggunaan")
    st.markdown("""
    1. Masukkan teks tweet
    2. Klik tombol **Prediksi**
    3. Lihat hasil sentimen dan confidence score """)
# ===== MAIN CONTENT =====
if model is not None:
    tab1, tab2 = st.tabs(["🔍 Prediksi Sentimen", "📈 About"])
    with tab1:
    st.subheader("Masukkan Teks Tweet")
    weet_text = st.text_area( "Ketikkan teks tweet di sini:",
    height=150,placeholder="Contoh: Pemerintah harus lebih transparan
dalam mengelola utang negara...",
    help="Masukkan tweet yang ingin dianalisis sentimennya")
    # Button untuk prediksi
    col1, col2, col3 = st.columns([1, 1, 1])
    with col2: predict_button = st.button("🔮 Prediksi",
type="primary", use_container_width=True)
        # Prediksi saat button diklik
        if predict_button:
        if tweet_text.strip() == "":
       st.warning("⚠️ Mohon masukkan teks tweet terlebih dahulu!")
         else:
         with st.spinner("Menganalisis sentimen..."):
         predicted_class, confidence, probabilities =
predict_sentiment(tweet_text)
       # Tampilkan hasil
       st.markdown("### 📊 Sentiment Result")
       # Box hasil prediksi - deteksi otomatis warna
       predicted_lower = predicted_class.lower()
       if 'positif' in predicted_lower or 'positive' in
predicted_lower:
       box_class = 'positive'
       elif 'negatif' in predicted_lower or 'negative' in
predicted_lower:
       box_class = 'negative'
       else:
       box_class = 'neutral'
       st.markdown(f"""
       <div class="prediction-box {box_class}">
         Tweet ini memiliki Sentimen "{predicted_class}" </div>
       <div class="confidence-score">
             Dengan confidence: {confidence:.2f}% </div>
```

```
                """, unsafe_allow_html=True)
        # Tampilkan detail preprocessing
        with st.expander("📝 Lihat Detail Preprocessing"):
        processed = preprocessor.preprocess(tweet_text)
        st.markdown(f"**Teks Original:**\n```\n{tweet_text}\n```")
                    st.markdown(f"**Setelah
Preprocessing:**\n```\n{processed}\n```")
        # Tampilkan probabilitas semua kelas
        with st.expander("📊 Lihat Probabilitas Semua Kelas"):
        prob_df = pd.DataFrame({
        'Sentimen': label_encoder.classes_,
        'Probabilitas (%)': [p * 100 for p in probabilities]})
         prob_df = prob_df.sort_values('Probabilitas (%)',
ascending=False
      # Bar chart
        st.bar_chart(prob_df.set_index('Sentimen'))
      # Table
        st.dataframe(prob_df, use_container_width=True)
      # Contoh tweet
         st.markdown("---")
         st.markdown("### 💡 Contoh Tweet")
         col1, col2, col3 = st.columns(3)
         with col1:
         if st.button("Contoh Positif"):
           st.session_state.example_text = "Pemerintah berhasil … "
       with col2:
       if st.button("Contoh Negatif"):
       st.session_state.example_text = "Utang negara semakin … "
         with col3:
             if st.button("Contoh Netral"):
                 st.session_state.example_text = "…"
    # Tampilkan contoh jika ada
        if 'example_text' in st.session_state:
            st.info(f"💬 Contoh: {st.session_state.example_text}")
    with tab2:
        st.subheader("📚 Tentang Aplikasi")
        st.markdown("""
        ### Analisis Sentimen Utang Negara ....
        **Dikembangkan dengan menggunakan Streamlit & TensorFlow**
        """)
else:
    st.error("❌ Gagal memuat model. Pastikan semua file artifacts
tersedia.")
    st.stop()
# ===== FOOTER =====
st.markdown("---")
st.markdown("""
<div style='text-align: center; color: #888; padding: 20px;'>
    <p>© 2025 Sentiment Analysis App | Powered by LSTM & Streamlit</p>
</div>
""", unsafe_allow_html=True)
```