LAMPIRAN

Lampiran 1. Surat Permohonan Pelabelan Data

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 : **1518**/UN48.11.1/KM/2025 Singaraja, 17 Juni 2025
Perihal : Surat Permohonan Pengambilan Data

Yth. Kepala SMA Negeri 1 Bebandem
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	: Ni Komang Arista Tri Wahyuni
NIM	: 2115091009
Program Studi	: Sistem Informasi
Jurusan	: Teknik Informatika
Data yang dibutuhkan	: Pelabelan data penelitian skripsi
Judul Penelitian	: Analisis Sentimen Ulasan Pengguna Terhadap Produk Skincare Korea Pada Platform Female Daily Menggunakan K-Nearest Neighbor

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

a.n Dekan
Wakil Dekan Bidang Akademik,

Made Windu Antara Kesiman
NIP 198211112008121001

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 : **1517**/UN48.11.1/KM/2025 Singaraja, 17 Juni 2025
Perihal : Surat Permohonan Pengambilan Data

Yth. Kepala SMP Negeri 6 Singaraja
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	: Ni Komang Arista Tri Wahyuni
NIM	: 2115091009
Program Studi	: Sistem Informasi
Jurusan	: Teknik Informatika
Data yang dibutuhkan	: Pelabelan data penelitian skripsi
Judul Penelitian	: Analisis Sentimen Ulasan Pengguna Terhadap Produk Skincare Korea Pada Platform Female Daily Menggunakan K-Nearest Neighbor

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

a.n Dekan
Wakil Dekan Bidang Akademik,

Made Windu Antara Kesiman
NIP 198211112008121001

Lampiran 2 Surat Pernyataan Validasi Pelabelan Data

SURAT PERNYATAAN HASIL VALIDASI DATA

Yang bertanda tangan di bawah ini:

Nama : Ni Nyoman Suci Mahartini, S.Pd., M.Pd.
NIP/NIK : 198403072011012006
Jabatan : Guru Bahasa Indonesia
Instansi : SMA Negeri 1 Bebandem

Dengan ini menyatakan bahwa saya telah melakukan validasi terhadap data ulasan produk skincare Korea Selatan yang telah diberikan oleh:

Nama Mahasiswa : Ni Komang Arista Tri Wahyuni
NIM : 2115091009
Program Studi : Sistem Informasi
Universitas : Universitas Pendidikan Ganesha

Data tersebut terdiri dari sejumlah kalimat ulasan yang telah diberi label sentimen positif dan negatif oleh mahasiswa. Berdasarkan hasil penelaahan terhadap konteks kalimat, diksi, serta makna yang terkandung di dalam ulasan, 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.

Bebandem, 30 Juli 2025
Guru Bahasa Indonesia,

Ni Nyoman Suci Mahartini, S.Pd., M.Pd.
NIP 198403072011012006

SURAT PERNYATAAN HASIL VALIDASI DATA

Yang bertanda tangan di bawah ini:

Nama : Putu Tri Noverawati, S.Pd.

NIP/NIK : 199911122024212009

Jabatan : Guru Bahasa Indonesia

Instansi : SMP Negeri 6 Singaraja

Dengan ini menyatakan bahwa saya telah melakukan validasi terhadap data ulasan produk skincare Korea Selatan yang telah diberikan oleh:

Nama Mahasiswa : Ni Komang Arista Tri Wahyuni

NIM : 2115091009

Program Studi : Sistem Informasi

Universitas : Universitas Pendidikan Ganesha

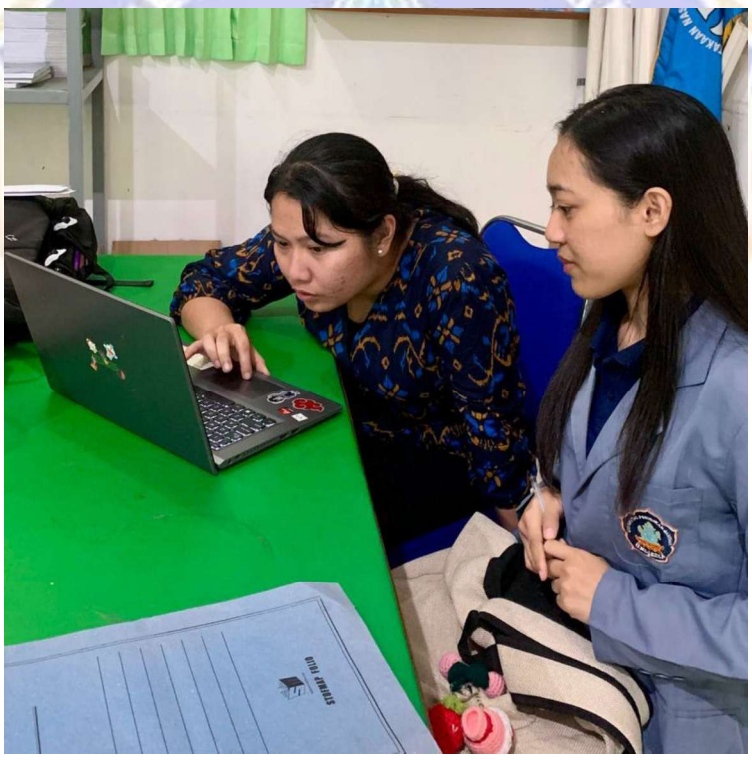
Data tersebut terdiri dari sejumlah kalimat ulasan yang telah diberi label sentimen positif dan negatif oleh mahasiswa. Berdasarkan hasil penelaahan terhadap konteks kalimat, diksi, serta makna yang terkandung di dalam ulasan, saya menyatakan bahwa pelabelan yang dilakukan sudah sesuai dengan kaidah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

Demikian surat ini dibuat untuk digunakan sebagaimana mestinya.

Singaraja, 19 Agustus 2025
Guru Bahasa Indonesia,

Putu Tri Noverawati, S.Pd.
NIP 199911122024212009

Lampiran 3. Dokumentasi Validasi Pelabelan Data



Lampiran 4. Implementasi Pre-Processing Dataset

1. *Case Folding*

```
import pandas as pd
df = pd.read_csv('full_data.csv')
# Case Folding
df['case_folding'] = df['Review'].str.lower()
# Hasil
df[['Review', 'case_folding']].head()
```

2. *Cleaning*

```
import re
import string
def clean_text(text):
    text = re.sub(r'\d+', '', text)
    # Hapus tanda baca
    text = text.translate(str.maketrans('', '',
string.punctuation))
    # Hapus whitespace berlebih
    text = text.strip()
    # hapus spasi di awal dan akhir
    text = re.sub(r'\s+', ' ', text)
    # ganti spasi berlebih jadi satu spasi
    text = re.sub("http\S+|www.\S+", '', text)
    # hapus URL
    text = re.sub("\n", " ", text)
    # hapus baris baru
    text = re.sub("[^\w\s]", "", text)
    # hapus karakter non-alfanumerik
    text = re.sub(r'\s+', " ", text).strip()
    # hapus spasi berlebih
    return text
# Penerapan Cleaning ke kolom hasil Case Folding
df['cleaned'] = df['case_folding'].apply(clean_text)
# Hasil
df[['case_folding', 'cleaned']].Sample(5)
```

3. *Tokenize*

```
import nltk
# Hapus cache
nltk.data.path.clear()
# Unduh punkt
nltk.download('punkt')
# Coba word_Tokenize
from nltk.Tokenize import word_Tokenize
# Tokenizer
df['tokens'] = df['cleaned'].apply(lambda x: x.split())
# Cek hasil tokenisasi
df[['cleaned', 'tokens']].Sample(5)
```

4. ***Normalize***

```
import pandas as pd
import ast
# Load kamus slang dari GitHub
url = "https://raw.githubusercontent.com/adeariniputri/text-
preprocesing/master/slang.csv"
df_github = pd.read_csv(url)
kamus_github = dict(zip(df_github['slang'],
df_github['formal']))
# Load kamus pribadi
df_kamus_pribadi = pd.read_csv('kamus_pribadi.csv')
kamus_pribadi = dict(zip(df_kamus_pribadi['slang'],
df_kamus_pribadi['formal']))
# Gabungkan kamus Normalize
kamus_normalisasi = {**kamus_github, **kamus_pribadi}
# Fungsi Normalize
def Normalize_text(tokens):
    Normalized = []
    for word in tokens:
        repl = kamus_normalisasi.get(word, word)
        Normalized.extend(repl.split())
    return Normalized
# Terapkan normalisasi
df['Normalized'] = df['tokens'].apply(Normalize_text)
# SAMPLE HASIL NORMALISASI
print(df[['tokens', 'Normalized']].Sample(min(5, len(df))))
```

5. **Handling Negasi**

```
import pandas as pd
import ast
import warnings
warnings.filterwarnings('ignore')
negation_words = [
    'tidak', 'tak', 'bukan', 'jangan', 'belum', 'tanpa',
    'gak', 'ga', 'nggak', 'ngga', 'enggak', 'gakusah'
]
print(f"\nKata negasi: {', '.join(negation_words)}\n")
def handle_negation(tokens):
    result = []
    i = 0
    while i < len(tokens):
        if tokens[i] in negation_words and i + 1 < len(tokens):
            negated_word = f"negasi_{tokens[i+1]}"
            result.append(negated_word)
            i += 2
        else:
            result.append(tokens[i])
            i += 1
    return result
print("Loading data...")
```

```
df = pd.read_csv('preprocessing_progress.csv', encoding='utf-
8')
print(f"✓ Data dimuat: {len(df)} baris\n")

# Convert string to list
df['normalized_tokens'] =
df['normalized_tokens'].apply(ast.literal_eval)
print("Proses Negation Handling (gabung negasi + kata)...\n")
df['negation_handled'] =
df['normalized_tokens'].apply(handle_negation)
print("📝 PERBANDINGAN (5 Sample dengan Negasi):\n")

# Cari data yang ada negasinya
has_negation = df['negation_handled'].apply(lambda x:
any('negasi_' in token for token in x))
samples_with_negation = df[has_negation].head(5)

if len(samples_with_negation) > 0:
    # Format untuk display
    samples_with_negation['before_display'] =
samples_with_negation['normalized_tokens'].apply(
        lambda x: str(x[:10])[1:-1] + '...'
    )
    samples_with_negation['after_display'] =
samples_with_negation['negation_handled'].apply(
        lambda x: str(x[:10])[1:-1] + '...'
    )
    display_df = pd.DataFrame({
        'Sebelum (Normalized)':
samples_with_negation['before_display'].values,
        'Sesudah (Negation Handled)':
samples_with_negation['after_display'].values
    })
    print(display_df.to_string(index=True))
else:
    # Jika tidak ada negasi, tampilkan 5 sample biasa
    print("⚠️  Tidak ada negasi terdeteksi, menampilkan sample
biasa:\n")

    samples = df.head(5).copy()
    samples['before_display'] =
samples['normalized_tokens'].apply(lambda x: str(x[:10])[1:-1]
+ '...')
    samples['after_display'] =
samples['negation_handled'].apply(lambda x: str(x[:10])[1:-1] +
'...')

    display_df = pd.DataFrame({
        'Sebelum (Normalized)': samples['before_display'],
        'Sesudah (Negation Handled)': samples['after_display']
    })
    print(display_df.to_string(index=True))
```

6. ***Stopword Removal***

```
!git clone
https://github.com/louisowen6/NLP_bahasa_resources.git
# Membaca daftar stopwords
with open('NLP_bahasa_resources/combined_stop_words.txt', 'r',
encoding='utf-8') as f:
    stopwords = set(f.read().splitlines())
# Menghapus stopwords
def remove_stopwords(tokens):
    return [word for word in tokens if word not in stopwords]
# Terapkan pada kolom 'Normalized'
df['no_stopwords'] = df['Normalized'].apply(remove_stopwords)
df['no_stopwords_str'] = df['no_stopwords'].apply(lambda x: "
".join(x))
# Hasil
print(df[['Normalized', 'no_stopwords']])
```

7. ***Stemming***

```
# Install library Sastrawi
!pip install Sastrawi
# Import modul untuk membuat stemmer
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
factory = StemmerFactory()
stemmer = factory.create_stemmer()
# Melakukan stemming
def stem_tokens(text):
    if not isinstance(text, str) or text.strip() == "":
        return []
    stemmed_text = stemmer.stem(text)
    return stemmed_text.split()
# Menerapkan fungsi stemming dan menyimpan di kolom baru
df['stemmed_tokens'] =
df['no_stopwords_str'].apply(stem_tokens)
# Hasil
print(df[['no_stopwords_str', 'stemmed_tokens']])
```

Lampiran 5. Ekstraksi Fitur TF-IDF

```
from sklearn.feature_extraction.text import TfidfVectorizer
#Menggabungkan list token menjadi string
df['stemmed_str'] = df['stemmed_tokens'].apply(lambda x: ' '.join(x))
#Inisialisasi TF-IDF Vectorizer
vectorizer = TfidfVectorizer()
#Transformasi ke bentuk TF-IDF
tfidf_matrix = vectorizer.fit_transform(df['stemmed_str'])
#Konversi ke DataFrame
tfidf_df = pd.DataFrame(tfidf_matrix.toarray(),
columns=vectorizer.get_feature_names_out())
#Menampilkan hasil TF-IDF
print(tfidf_df.head())
```

Lampiran 6. Implementasi KNN tanpa Resampling

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
import warnings
warnings.filterwarnings('ignore')

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import (classification_report, confusion_matrix,
                             accuracy_score, f1_score,
precision_score, recall_score)

print("="*70)
print("KNN BASELINE MODEL")
print("="*70)

print("\n[1/4] Loading data...")

with open('tfidf_train.pkl', 'rb') as f:
    X_train = pickle.load(f)
with open('tfidf_test.pkl', 'rb') as f:
    X_test = pickle.load(f)
with open('labels_train.pkl', 'rb') as f:
    y_train = pickle.load(f)
with open('labels_test.pkl', 'rb') as f:
    y_test = pickle.load(f)
with open('label_encoder.pkl', 'rb') as f:
    le = pickle.load(f)

print(f"✓ Training: {X_train.shape}, Testing: {X_test.shape}")

print("\n[2/4] Cross Validation (5-Fold)...")

k_range = [3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31]
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

results = {
    'k': [],
    'train_acc': [],
    'val_acc': [],
    'train_f1': [],
    'val_f1': [],
    'train_precision': [],
    'val_precision': [],
    'train_recall': [],
    'val_recall': []
}
```

```
for k in k_range:
    val_accs = []
    val_f1s = []
    val_precisions = []
    val_recalls = []

    # K-Fold CV
    for train_idx, val_idx in skf.split(X_train, y_train):
        X_tr = X_train[train_idx]
        X_val = X_train[val_idx]
        y_tr = y_train.iloc[train_idx] if hasattr(y_train, 'iloc')
else y_train[train_idx]
        y_val = y_train.iloc[val_idx] if hasattr(y_train, 'iloc') else
y_train[val_idx]

        knn = KNeighborsClassifier(n_neighbors=k, weights='uniform',
metric='euclidean')
        knn.fit(X_tr, y_tr)
        y_val_pred = knn.predict(X_val)

        val_accs.append(accuracy_score(y_val, y_val_pred))
        val_f1s.append(f1_score(y_val, y_val_pred,
average='weighted'))
        val_precisions.append(precision_score(y_val, y_val_pred,
average='weighted'))
        val_recalls.append(recall_score(y_val, y_val_pred,
average='weighted'))

    # Train pada full training set
    knn_full = KNeighborsClassifier(n_neighbors=k, weights='uniform',
metric='euclidean')
    knn_full.fit(X_train, y_train)
    y_train_pred = knn_full.predict(X_train)

    results['k'].append(k)
    results['train_acc'].append(accuracy_score(y_train, y_train_pred))
    results['val_acc'].append(np.mean(val_accs))
    results['train_f1'].append(f1_score(y_train, y_train_pred,
average='weighted'))
    results['val_f1'].append(np.mean(val_f1s))
    results['train_precision'].append(precision_score(y_train,
y_train_pred, average='weighted'))
    results['val_precision'].append(np.mean(val_precisions))
    results['train_recall'].append(recall_score(y_train, y_train_pred,
average='weighted'))
    results['val_recall'].append(np.mean(val_recalls))

# Tampilkan hasil CV lengkap
print("\n TABEL LENGKAP HASIL CV:")
df_results = pd.DataFrame(results)
print(df_results.to_string(index=False))
```

```
best_idx = np.argmax(results['val_f1'])
best_k = results['k'][best_idx]
print(f"\n⏳ Best K = {best_k} (Val F1:
{results['val_f1'][best_idx]:.4f})")

print("\n[3/4] Training final model...")

knn_best = KNeighborsClassifier(n_neighbors=best_k, weights='uniform',
metric='euclidean')
knn_best.fit(X_train, y_train)

y_train_pred = knn_best.predict(X_train)
y_test_pred = knn_best.predict(X_test)

train_acc = accuracy_score(y_train, y_train_pred)
test_acc = accuracy_score(y_test, y_test_pred)
test_f1 = f1_score(y_test, y_test_pred, average='weighted')
test_precision = precision_score(y_test, y_test_pred,
average='weighted')
test_recall = recall_score(y_test, y_test_pred, average='weighted')

print(f"\n PERFORMA MODEL (K={best_k}):")
print(f"   Train Accuracy: {train_acc:.4f}")
print(f"   Test Accuracy:  {test_acc:.4f}")
print(f"   Test F1-Score:  {test_f1:.4f}")
print(f"   Test Precision: {test_precision:.4f}")
print(f"   Test Recall:    {test_recall:.4f}")

print(f"\n CLASSIFICATION REPORT:")
print(classification_report(y_test, y_test_pred,
target_names=le.classes_))

print("\n[4/4] Creating visualizations...")

cm = confusion_matrix(y_test, y_test_pred)

fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# Plot 1: Accuracy vs K
ax1 = axes[0]
ax1.plot(results['k'], results['train_acc'], marker='o', label='Train
Accuracy', linewidth=2)
ax1.plot(results['k'], results['val_acc'], marker='s',
label='Validation Accuracy', linewidth=2)
ax1.axvline(x=best_k, color='red', linestyle='--', label=f'Best
K={best_k}', alpha=0.7)
ax1.set_xlabel('K (Number of Neighbors)')
ax1.set_ylabel('Accuracy')
ax1.set_title('KNN Performance: Accuracy vs K', fontweight='bold')
ax1.legend()
ax1.grid(True, alpha=0.3)
```

```
# Plot 2: Confusion Matrix
ax2 = axes[1]
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=le.classes_, yticklabels=le.classes_, ax=ax2)
ax2.set_xlabel('Predicted')
ax2.set_ylabel('True')
ax2.set_title(f'Confusion Matrix (K={best_k})', fontweight='bold')

plt.tight_layout()
plt.savefig('knn_baseline_results.png', dpi=300, bbox_inches='tight')
plt.show()
```

Lampiran 7. Implementasi KNN dengan ADASYN

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
import warnings
warnings.filterwarnings('ignore')

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import (classification_report, confusion_matrix,
                             accuracy_score, f1_score,
precision_score, recall_score)
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import ADASYN

print("="*70)
print("KNN + ADASYN MODEL")
print("="*70)

print("\n[1/5] Loading data...")

with open('tfidf_train.pkl', 'rb') as f:
    X_train = pickle.load(f)
with open('tfidf_test.pkl', 'rb') as f:
    X_test = pickle.load(f)
with open('labels_train.pkl', 'rb') as f:
    y_train = pickle.load(f)
with open('labels_test.pkl', 'rb') as f:
    y_test = pickle.load(f)
with open('label_encoder.pkl', 'rb') as f:
    le = pickle.load(f)

print(f"✓ Training: {X_train.shape}, Testing: {X_test.shape}")

# SIMPAN distribusi SEBELUM ADASYN
print("\n[2/5] Applying ADASYN...")

# Hitung distribusi sebelum ADASYN
unique_before, counts_before = np.unique(y_train, return_counts=True)
print(f"\n📊 Distribusi SEBELUM ADASYN:")
for label, count in zip(unique_before, counts_before):
    print(f"   {le.inverse_transform([label])[0]}: {count} samples")

adasyn = ADASYN(n_neighbors=3, random_state=42)
X_train_bal, y_train_bal = adasyn.fit_resample(X_train, y_train)

# Hitung distribusi setelah ADASYN
unique_after, counts_after = np.unique(y_train_bal,
return_counts=True)
```

```
print(f"\n📊 Distribusi SETELAH ADASYN:")
for label, count in zip(unique_after, counts_after):
    print(f"   {le.inverse_transform([label])[0]}: {count} samples")

print(f"\n✓ ADASYN selesai")
print(f"   Before: {X_train.shape[0]} samples")
print(f"   After:  {X_train_bal.shape[0]} samples (balanced)")

print("\n[3/5] Scaling data...")

scaler = StandardScaler(with_mean=False)
X_train_bal_scaled = scaler.fit_transform(X_train_bal)
X_test_scaled = scaler.transform(X_test)

print("✓ Scaling selesai")

print("\n[4/5] Cross Validation (5-Fold)...")

k_range = [3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29]
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

results = {
    'k': [],
    'train_acc': [],
    'val_acc': [],
    'train_f1': [],
    'val_f1': [],
    'train_precision': [],
    'val_precision': [],
    'train_recall': [],
    'val_recall': []
}

for k in k_range:
    train_accs = []
    val_accs = []
    train_f1s = []
    val_f1s = []
    train_precisions = []
    val_precisions = []
    train_recalls = []
    val_recalls = []

    # K-Fold CV
    for train_idx, val_idx in skf.split(X_train_bal_scaled,
y_train_bal):
        X_tr = X_train_bal_scaled[train_idx]
        X_val = X_train_bal_scaled[val_idx]
        y_tr = y_train_bal[train_idx]
        y_val = y_train_bal[val_idx]
```

```
        knn = KNeighborsClassifier(n_neighbors=k, weights='uniform',
metric='euclidean')
        knn.fit(X_tr, y_tr)

        y_tr_pred = knn.predict(X_tr)
        y_val_pred = knn.predict(X_val)

        train_accs.append(accuracy_score(y_tr, y_tr_pred))
        train_f1s.append(f1_score(y_tr, y_tr_pred,
average='weighted'))
        train_precisions.append(precision_score(y_tr, y_tr_pred,
average='weighted'))
        train_recalls.append(recall_score(y_tr, y_tr_pred,
average='weighted'))

        val_accs.append(accuracy_score(y_val, y_val_pred))
        val_f1s.append(f1_score(y_val, y_val_pred,
average='weighted'))
        val_precisions.append(precision_score(y_val, y_val_pred,
average='weighted'))
        val_recalls.append(recall_score(y_val, y_val_pred,
average='weighted'))

    results['k'].append(k)
    results['train_acc'].append(np.mean(train_accs))
    results['val_acc'].append(np.mean(val_accs))
    results['train_f1'].append(np.mean(train_f1s))
    results['val_f1'].append(np.mean(val_f1s))
    results['train_precision'].append(np.mean(train_precisions))
    results['val_precision'].append(np.mean(val_precisions))
    results['train_recall'].append(np.mean(train_recalls))
    results['val_recall'].append(np.mean(val_recalls))

# Tampilkan hasil CV lengkap
print("\n📊 TABEL LENGKAP HASIL CV:")
df_results = pd.DataFrame(results)
print(df_results.to_string(index=False))

best_idx = np.argmax(results['val_f1'])
best_k = results['k'][best_idx]
print(f"\n🏆 Best K = {best_k} (Val F1:
{results['val_f1'][best_idx]:.4f})")

print("\n[5/5] Training final model...")

knn_best = KNeighborsClassifier(n_neighbors=best_k, weights='uniform',
metric='euclidean')
knn_best.fit(X_train_bal_scaled, y_train_bal)

y_train_pred = knn_best.predict(X_train_bal_scaled)
y_test_pred = knn_best.predict(X_test_scaled)
```

```
train_acc = accuracy_score(y_train_bal, y_train_pred)
test_acc = accuracy_score(y_test, y_test_pred)
test_f1 = f1_score(y_test, y_test_pred, average='weighted')
test_precision = precision_score(y_test, y_test_pred,
average='weighted')
test_recall = recall_score(y_test, y_test_pred, average='weighted')

print(f"\n📊 PERFORMA MODEL (K={best_k}):")
print(f"   Train Accuracy: {train_acc:.4f}")
print(f"   Test Accuracy:  {test_acc:.4f}")
print(f"   Test F1-Score:  {test_f1:.4f}")
print(f"   Test Precision: {test_precision:.4f}")
print(f"   Test Recall:    {test_recall:.4f}")

print(f"\n CLASSIFICATION REPORT:")
print(classification_report(y_test, y_test_pred,
target_names=le.classes_))

print("\n📊 Creating visualizations...")

cm = confusion_matrix(y_test, y_test_pred)

# BUAT 3 PLOT: ADASYN Comparison + Accuracy vs K + Confusion Matrix
fig, axes = plt.subplots(1, 3, figsize=(20, 5))

# Plot 1: DIAGRAM BATANG PERBANDINGAN ADASYN
ax1 = axes[0]

labels = le.classes_
x = np.arange(len(labels))
width = 0.35

bars1 = ax1.bar(x - width/2, counts_before, width, label='Before
ADASYN', color='#ef4444', alpha=0.8)
bars2 = ax1.bar(x + width/2, counts_after, width, label='After
ADASYN', color='#10b981', alpha=0.8)

# Tambahkan nilai di atas bar
for bar in bars1:
    height = bar.get_height()
    ax1.text(bar.get_x() + bar.get_width()/2., height,
             f'{int(height)}',
             ha='center', va='bottom', fontsize=10, fontweight='bold')

for bar in bars2:
    height = bar.get_height()
    ax1.text(bar.get_x() + bar.get_width()/2., height,
             f'{int(height)}',
             ha='center', va='bottom', fontsize=10, fontweight='bold')

ax1.set_xlabel('Class', fontweight='bold')
ax1.set_ylabel('Number of Samples', fontweight='bold')
```

```
ax1.set_title('Data Distribution: Before vs After ADASYN',
fontweight='bold', fontsize=12)
ax1.set_xticks(x)
ax1.set_xticklabels(labels)
ax1.legend()
ax1.grid(True, alpha=0.3, axis='y')

# Plot 2: Accuracy vs K
ax2 = axes[1]
ax2.plot(results['k'], results['train_acc'], marker='o', label='Train
Accuracy', linewidth=2, color='#3b82f6')
ax2.plot(results['k'], results['val_acc'], marker='s',
label='Validation Accuracy', linewidth=2, color='#f59e0b')
ax2.axvline(x=best_k, color='red', linestyle='--', label=f'Best
K={best_k}', alpha=0.7, linewidth=2)
ax2.set_xlabel('K (Number of Neighbors)', fontweight='bold')
ax2.set_ylabel('Accuracy', fontweight='bold')
ax2.set_title('KNN + ADASYN Performance: Accuracy vs K',
fontweight='bold', fontsize=12)
ax2.legend()
ax2.grid(True, alpha=0.3)

# Plot 3: Confusion Matrix
ax3 = axes[2]
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=le.classes_, yticklabels=le.classes_, ax=ax3,
            cbar_kws={'label': 'Count'})
ax3.set_xlabel('Predicted', fontweight='bold')
ax3.set_ylabel('True', fontweight='bold')
ax3.set_title(f'Confusion Matrix (K={best_k})', fontweight='bold',
fontsize=12)

plt.tight_layout()
plt.savefig('knn_adasyn_results.png', dpi=300, bbox_inches='tight')
print("✓ Visualization saved: knn_adasyn_results.png")
plt.show()
```

RIWAYAT HIDUP

Ni Komang Arista Tri Wahyuni lahir di Karangasem pada 9 Mei 2003. Penulis lahir dari pasangan suami istri Bapak I Ketut Latra dan Ibu Ni Ketut Sari. Penulis berkebangsaan Indonesia dan beragama Hindu. Kini Penulis beralamat di Banjar Tanah Ampo, Desa Jungutan, Kecamatan Bebandem, Kabupaten Karangasem, Provinsi Bali. Penulis menyelesaikan pendidikan dasar di SD Negeri 3 Sibetan dan lulus pada tahun 2015. Kemudian penulis melanjutkan pendidikan di SMP Negeri 1 Bebandem dan lulus pada tahun 2018. Pada tahun 2021, penulis lulus dari SMA Negeri 1 Bebandem jurusan Ilmu Pengetahuan Alam dan melanjutkan studi (S1) di Universitas Pendidikan Ganesha dengan Program Studi Sistem Informasi, Jurusan Teknik Informatika.