LAMPIRAN

```
def clean_pesan_data(text):
    text = re.sub(r'@[A-Za-z0-9_]+', '', text)# Hapus mention
(@username)
    text = re.sub(r'#\w+', '', text)# Hapus hashtag
    text = re.sub(r'RT[\s]+', '', text)# Hapus retweet (RT)
    text = re.sub(r'https?://\S+', '', text)# Hapus URL
    text = re.sub(r'\d+', '', text)# Hapus angka
    text = re.sub(r'[^A-Za-z ]', '', text)# Hapus karakter non-
alfabet
    text = re.sub(r'\s+', ' ', text).strip()# Hapus spasi
berlebih

    return text

final_df['full_text'] =
final_df['full_text'].apply(clean_pesan_data)

final_df['full_text'] = final_df['full_text' ].str.lower()

import json
def load_slang_dict(filename):
    with open(filename, "r", encoding="utf-8") as f:
        slang_dict = json.load(f)  # Membaca file sebagai JSON
    return slang_dict

# Load slang word dictionary dari file
slang_dict = load_slang_dict("combined_slang_words.txt")  #
Ganti dengan path file kamu

# Fungsi normalisasi menggunakan dictionary dari file
def normalisasi(str_text):
    for key, value in slang_dict.items():
        str_text = re.sub(r'\b' + re.escape(key) + r'\b', value,
str_text)  # Ganti kata utuh
    return str_text

# Terapkan normalisasi pada kolom 'text' di DataFrame
final_df['full_text'] = final_df['full_text'].apply(normalisasi)

from Sastrawi.StopWordRemover.StopWordRemoverFactory import
StopWordRemoverFactory

# Ambil stopword bawaan dari Sastrawi
factory = StopWordRemoverFactory()
default_stopwords = set(factory.get_stop_words())

# Fungsi hapus stopword
def remove_stopwords(text):
    words = text.split()
    filtered_words = [word for word in words if word.lower() not
in default_stopwords]
    return " ".join(filtered_words)

# Terapkan ke semua baris di train_data
final_df['full_text'] =
final_df['full_text'].apply(remove_stopwords)
```

```
import nltk
from nltk.tokenize import word_tokenize

nltk.download('punkt')  # Unduh tokenizer bahasa Indonesia jika
diperlukan

final_df['full_text'] = final_df['full_text'].apply(lambda x:
word_tokenize(x))

from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
factory = StemmerFactory()
stemmer = factory.create_stemmer()

# Fungsi stemming yang menerima input berupa list of tokens
def stemming(tokens):
    stemmed_tokens = [stemmer.stem(token) for token in tokens]
    return " ".join(stemmed_tokens)  # Gabungkan lagi jadi
string

# Terapkan setelah tokenisasi
final_df['full_text'] = final_df['full_text'].apply(stemming)
```

Source Code Preprocessing Data

```
X = final_df['full_text']   # Fitur (teks)
y = final_df['Label']  # Label kategori

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42, shuffle=True)

# TF-IDF Vectorizer
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)  # Belajar
dari X_train
X_test_tfidf = vectorizer.transform(X_test)

tfidf_df = pd.DataFrame(X_train_tfidf.toarray(),
columns=vectorizer.get_feature_names_out())
```

Source Code Pembobotan TF-IDF

```
# Model Naive Bayes
modelNB = MultinomialNB()
modelNB.fit(X_train_tfidf, y_train)

y_pred = modelNB.predict(X_test_tfidf)

import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score,
classification_report, confusion_matrix

# Evaluasi dasar
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred,
target_names=["negatif", "positif"])
```

```
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"\nAccuracy: {accuracy:.2f}")
print("Classification Report:\n", report)
print("Confusion Matrix:\n", conf_matrix)

# Visualisasi confusion matrix
plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap="Blues",
            xticklabels=["negatif", "positif"],
            yticklabels=["negatif", "positif"])
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix (Negatif vs Positif)")
plt.tight_layout()
plt.show()
```

Source Code Implementasi NB

```
from sklearn.neighbors import KNeighborsClassifier
modelknn = KNeighborsClassifier(n_neighbors=9)
modelknn.fit(X_train_tfidf, y_train)

y_pred = modelknn.predict(X_test_tfidf)

import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score,
classification_report, confusion_matrix

# Evaluasi dasar
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred,
target_names=["negatif", "positif"])
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"\nAccuracy: {accuracy:.2f}")
print("Classification Report:\n", report)
print("Confusion Matrix:\n", conf_matrix)

# Visualisasi confusion matrix
plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap="Blues",
            xticklabels=["negatif", "positif"],
            yticklabels=["negatif", "positif"])
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix (Negatif vs Positif) KNN")
plt.tight_layout()
plt.show()
```

Source Code Implementasi K-NN