LAMPIRAN

Lampiran 1 Pengumpulan Data atau Scrapping Data

Lampiran 2 Hasil dari Pengumpulan Data atau Scrapping Data

Lampiran 3 Melakukan Studi Literatur

Lampiran 4 Link Dataset

https://drive.google.com/file/d/1eRo4MdwMReWt0Flxna2dQ0L-n4prHfPv/view?usp=sharing

https://drive.google.com/file/d/15MORi1UZfn-6pIcUmQeBj_rKcYb97PBU/view?usp=sharing

Lampiran 5 *Source Code*

```
# -*- coding: utf-8 -*-
"""SVM X RF.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1H1T5FybSukBNDIa3ARjNjv6j7firBUCc

# Import Library
"""

import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score, KFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
import seaborn as sns
import matplotlib.pyplot as plt
import time
from wordcloud import WordCloud

"""# Import Data"""

data_clean = pd.read_csv('/content/drive/MyDrive/skripsi/output_sentimen_tweet.csv')
data_clean

data_clean.sentiment.unique()

data_clean = data_clean.astype({'sentiment':'category'})
data_clean = data_clean.astype({'tweet_text_cleaned':'string'})
data_clean.dtypes
```

```
# Membuat objek TfidfVectorizer dan mengubah teks menjadi matriks tf-
idf
tfidf = TfidfVectorizer(max_features=1000)
tfidf_matrix = tfidf.fit_transform(data_clean['tweet_text_cleaned'])

# Menampilkan jumlah counts
print("Counts:", tfidf_matrix.shape[1])

# Menampilkan shape dari matriks tf-idf
print("Shape:", tfidf_matrix.shape)

# Menampilkan vocabulary
print("Vocabulary:", tfidf.vocabulary_)

# Mendapatkan nilai IDF
idf_scores = tfidf.idf_

# Menampilkan nilai IDF untuk setiap kata dalam vocabulary
for word, idf in zip(tfidf.get_feature_names_out(), idf_scores):
    print(f"Word: {word}, IDF: {idf}")

# Mencetak matriks
print(tfidf_matrix.toarray())

"""# Split Data"""

# Memisahkan fitur dan label
X = tfidf_matrix
y = data_clean['sentiment']

# Membagi data menjadi data pelatihan dan data pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.9, random_state=42)

# Menampilkan jumlah data
print("Jumlah data pelatihan:", X_train.shape[0])
print("Jumlah data pengujian:", X_test.shape[0])

# Jumlah data negatif pada data pelatihan
jumlah_data_negatif_pelatihan = (y_train == "negatif").sum()

# Jumlah data positif pada data pelatihan
jumlah_data_positif_pelatihan = (y_train == "positif").sum()

# Jumlah data negatif pada data pengujian
```

```
jumlah_data_negatif_pengujian = (y_test == "negatif").sum()

# Jumlah data positif pada data pengujian
jumlah_data_positif_pengujian = (y_test == "positif").sum()

print("Jumlah data negatif pada data pelatihan:",
jumlah_data_negatif_pelatihan)
print("Jumlah data positif pada data pelatihan:",
jumlah_data_positif_pelatihan)
print("Jumlah data negatif pada data pengujian:",
jumlah_data_negatif_pengujian)
print("Jumlah data positif pada data pengujian:",
jumlah_data_positif_pengujian)

"""# Random Forest"""

from sklearn.ensemble import RandomForestClassifier
# Membuat objek Random Forest Classifier
rf_classifier = RandomForestClassifier(random_state=42)

# Melatih model dengan data pelatihan
rf_classifier.fit(X_train, y_train)

# Melakukan prediksi pada data pengujian
y_pred = rf_classifier.predict(X_test)

# Hitung metrik - Matriks kebingungan, akurasi, presisi, perolehan,
dan skor f1
confusion_mat = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Ubah metrik menjadi persentase dan bulatkan menjadi 2 tempat desimal
accuracy_percent = round(accuracy * 100, 2)
precision_percent = round(precision * 100, 2)
recall_percent = round(recall * 100, 2)
f1_percent = round(f1 * 100, 2)

print("Confusion Matrix:")
print(confusion_mat)

# Plot matriks konfusi sebagai heatmap
plt.figure(figsize=(8, 6))
```

```
sns.heatmap(confusion_mat, annot=True, fmt="d", cmap="Oranges",
xticklabels=["Negatif", "Positif"], yticklabels=["Negatif",
"Positif"])
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()

print("")
print("=== Hasil Evaluasi Random Forest ===")
print("Accuracy: {}%".format(accuracy_percent))
print("Precision: {}%".format(precision_percent))
print("Recall: {}%".format(recall_percent))
print("F1 Score: {}%".format(f1_percent))

"""# SVM"""

from sklearn.svm import SVC
# Membuat objek SVM
svm_classifier = SVC(kernel='linear', random_state=42)

# Melatih model dengan data pelatihan
svm_classifier.fit(X_train, y_train)

# Melakukan prediksi pada data pengujian
y_pred = svm_classifier.predict(X_test)

# Hitung metrik - Matriks kebingungan, akurasi, presisi, perolehan,
dan skor f1
confusion_mat = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Hitung metrik dalam persentase
accuracy_percent = accuracy * 100
precision_percent = precision * 100
recall_percent = recall * 100
f1_percent = f1 * 100

print("Confusion Matrix:")
print(confusion_mat)

# Plot matriks konfusi sebagai heatmap
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_mat, annot=True, fmt="d", cmap="Greens",
xticklabels=["Negatif", "Positif"], yticklabels=["NegatiF",
"Positif"])
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()

print("")
print("=== Hasil Evaluasi Support Vector Machine ===")
print("Accuracy:", accuracy_percent, "%")
print("Precision:", precision_percent, "%")
print("Recall:", recall_percent, "%")
print("F1 Score:", f1_percent, "%")

"""# Vader"""

from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk
data = data_clean
nltk.download('vader_lexicon')
# Inisialisasi SentimentIntensityAnalyzer
sid = SentimentIntensityAnalyzer()

# Melakukan analisis sentimen untuk setiap tweet dalam dataframe
sentiments = []
compound_scores = [] # Daftar untuk menyimpan skor gabungan
for tweet in data['tweet_text_cleaned']:
    sentiment_score = sid.polarity_scores(str(tweet))
    compound_scores.append(sentiment_score['compound']) # Simpan skor
gabungan
    if sentiment_score['compound'] >= 0.05:
        sentiments.append('positif')
    elif sentiment_score['compound'] <= -0.05:
        sentiments.append('negatif')
    else:
        sentiments.append('netral')

# Menambahkan kolom sentimen ke dalam dataframe
data['sentiment'] = sentiments
data['compound'] = compound_scores # Tambahkan skor gabungan sebagai
kolom

# Menyaring tweet dengan sentimen positif
```

```
positive_tweets = data[data['sentiment'] == 'positif']

# prompt: Diagram Persebaran Sentimen Pada Data

import matplotlib.pyplot as plt

# Menyiapkan data untuk diagram persebaran
x = data['compound']
y = data['sentiment']

# Membuat diagram persebaran
plt.scatter(x, y, alpha=0.5)

# Menambahkan label sumbu
plt.xlabel('Compound Score')
plt.ylabel('Sentiment')

# Menampilkan diagram persebaran
plt.show()

"""# WordCloud Positive"""

# Menggabungkan semua teks tweet positif
all_positive_text = ' '.join(tweet for tweet in
positive_tweets['tweet_text_cleaned'])

# Membuat objek WordCloud
wordcloud = WordCloud(width=800, height=400, max_font_size=100,
max_words=100, background_color='white').generate(all_positive_text)

# Menampilkan WordCloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')  # Menghilangkan axis
plt.title('Wordcloud Sentimen Positif')
plt.show()

all_positive_text

label_positif_data = data[data['sentiment'] == 'positif']

# Hitung frekuensi kata untuk setiap label
positif_word_freq =
label_positif_data['tweet_text_cleaned'].str.split(expand=True).stack(
).value_counts()
```

```
# Ambil 15 kata teratas untuk setiap label
top_positif_words = positif_word_freq.head(15)

# Visualisasikan dalam bentuk grafik batang
plt.figure(figsize=(12, 6))
bars = plt.bar(top_positif_words.index, top_positif_words,
color='green')
plt.title('Frekuensi Kata dalam Sentimen Positif')
plt.xlabel('Kata')
plt.ylabel('Frekuensi')
plt.xticks(rotation=45)

# Menambahkan label frekuensi di atas setiap bar
for index, value in enumerate(top_positif_words):
    plt.text(index, value + 0.1, str(value), ha='center', va='bottom')

plt.show()

"""# WordCloud Netral"""

neutral_tweets = data[data['sentiment'] == 'netral']

# Menggabungkan semua teks tweet netral
all_neutral_text = ' '.join(tweet for tweet in
neutral_tweets['tweet_text_cleaned'])

# Membuat objek WordCloud
wordcloud = WordCloud(width=800, height=400, max_font_size=100,
max_words=100, background_color='white').generate(all_neutral_text)

# Menampilkan WordCloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')  # Menghilangkan axis
plt.title('Wordcloud Sentimen Netral')
plt.show()

all_neutral_text

label_netral_df = data[data['sentiment'] == 'netral']

# Hitung frekuensi kata untuk setiap label
```

```
netral_word_freq =
label_netral_df['tweet_text_cleaned'].str.split(expand=True).stack().v
alue_counts()

# Ambil 15 kata teratas untuk setiap label
top_netral_words = netral_word_freq.head(15)

# Visualisasikan dalam bentuk grafik batang
plt.figure(figsize=(12, 6))
bars = plt.bar(top_netral_words.index, top_netral_words, color='blue')
plt.title('Frekuensi Kata dalam Sentimen Netral')
plt.xlabel('Kata')
plt.ylabel('Frekuensi')
plt.xticks(rotation=45)

# Menambahkan label frekuensi di atas setiap bar
for index, value in enumerate(top_netral_words):
    plt.text(index, value + 0.1, str(value), ha='center', va='bottom')

plt.show()

"""# WordCloud Negatif"""

import matplotlib.pyplot as plt
# Menyaring tweet dengan sentimen negatif
negative_tweets = data[data['sentiment'] == 'negatif']

# Menggabungkan semua teks tweet negatif
all_negative_text = ' '.join(tweet for tweet in
negative_tweets['tweet_text_cleaned'])

# Membuat objek WordCloud
wordcloud = WordCloud(width=800, height=400, max_font_size=100,
max_words=100, background_color='white').generate(all_negative_text)

# Menampilkan WordCloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')  # Menghilangkan axis
plt.title('Wordcloud Sentimen Negatif')
plt.show()

all_negative_text

label_negatif_df = data[data['sentiment'] == 'negatif']
```

```
# Hitung frekuensi kata untuk setiap label
negatif_word_freq =
label_negatif_df['tweet_text_cleaned'].str.split(expand=True).stack().
value_counts()

# Ambil 15 kata teratas untuk setiap label
top_negatif_words = negatif_word_freq.head(15)

# Visualisasikan dalam bentuk grafik batang
plt.figure(figsize=(12, 6))
bars = plt.bar(top_negatif_words.index, top_negatif_words,
color='red')
plt.title('Frekuensi Kata dalam Sentimen Negatif')
plt.xlabel('Kata')
plt.ylabel('Frekuensi')
plt.xticks(rotation=45)

# Menambahkan label frekuensi di atas setiap bar
for index, value in enumerate(top_negatif_words):
    plt.text(index, value + 0.1, str(value), ha='center', va='bottom')

plt.show()
```

DAFTAR RIWAYAT HIDUP

Muhammad Verel Prisco Alfito Devani adalah seorang mahasiswa Teknik Informatika di Universitas Islam Balitar, yang tinggal di Blitar, Jawa Timur. Perjalanan akademik saya bermula di SDN 5 Kademangan, di mana saya lulus pada tahun 2014 dengan fondasi kuat dalam disiplin ilmu dasar. Menyusul pendidikan dasar, saya melanjutkan studi di SMPN 2 Kademangan dan menyelesaikan pendidikan menengah pertama saya pada tahun 2017. Keingintahuan saya terhadap ilmu sosial dan teknologi semakin berkembang selama periode ini. Tahun 2020 menandai kelulusan saya dari SMAN 1 Kademangan dalam jurusan IPS, yang mempertajam minat saya dalam analisis data dan teknologi. Dorongan untuk menggali lebih dalam mengenai teknologi informasi membawa saya ke Universitas Islam Balitar, di mana saya saat ini mengembangkan kemampuan di bidang pemrograman dan manajemen data. Kompetensi ini saya perkuat untuk mendukung penelitian skripsi yang sedang saya kerjakan.