DAFTAR PUSTAKA

Aisah, Iis Siti, Bambang Irawan, and Tati Suprapti. 2023. 7 Jurnal Mahasiswa Teknik Informatika *ALGORITMA SUPPORT VECTOR MACHINE (SVM) UNTUK ANALISIS SENTIMEN ULASAN APLIKASI AL QUR'AN DIGITAL*.

Ansori, Yusuf, and Khadijah Fahmi Hayati Holle. 2022. 'Perbandingan Metode Machine Learning Dalam Analisis Sentimen Twitter'. *Jurnal Sistem dan Teknologi Informasi (JustIN)* 10(4): 429. doi:10.26418/justin.v10i4.51784.

Arisandi, Riza Rizqi Robbi, Budi Warsito, and Arief Rachman Hakim. 2022. 'APLIKASI NAÏVE BAYES CLASSIFIER (NBC) PADA KLASIFIKASI STATUS GIZI BALITA STUNTING DENGAN PENGUJIAN K-FOLD CROSS VALIDATION'. *Jurnal Gaussian* 11(1): 130–39. doi:10.14710/j.gauss.v11i1.33991.

Awangga, Rolly Maulana, and Nuha Hanifatul Khonsa'. 2022. 'Analisis Performa Algoritma Random Forest Dan Naive Bayes Multinomial Pada Dataset Ulasan Obat Dan Ulasan Film'. *InComTech : Jurnal Telekomunikasi dan Komputer* 12(1): 60. doi:10.22441/incomtech.v12i1.14770.

Canty, Angelo J. 2002. 2 *Resampling Methods in R: The Boot Package*.

Hamka, Muhammad. 2024. 1 Jurnal Rekayasa Sistem Informasi dan Teknologi *ANALISIS SENTIMEN PENGGUNA E-COMMERCE DAN MARKETPLACE MENGGUNAKAN SUPPORT VECTOR MACHINE*.

Mohamad Iqbal Jauhari, Nur, Resty Wulanningrum, and Ahmad Bagus Setiawan. 2024. 8 Seminar Nasional Inovasi Teknologi) 1331 INOTEK *Sistem Deteksi Kendaraan Menggunakan StreamLit Metode Yolo Universitas Nusantara PGRI Kediri*. Online.

Nurwahidah, Dalilah, Gifthera Dwilestari, Nisa Dienwati Nuris, Riri Narasati, and Teknik Informatika. 2023. 7 Jurnal Mahasiswa Teknik Informatika *ANALISIS SENTIMEN DATA ULASAN PENGGUNA APLIKASI GOOGLE KELAS PADA GOOGLE PLAY STORE MENGGUNAKAN ALGORITMA NAÏVE BAYES*.

Pristian Luthfy Romadloni, Bagus Adhi Kusuma, and Wiga Maulana Baihaqi. 2022. 'KOMPARASI METODE PEMBELAJARAN MESIN UNTUK IMPLEMENTASI PENGAMBILAN KEPUTUSAN DALAM MENENTUKAN PROMOSI JABATAN KARYAWAN'. *JATI (Jurnal Mahasiswa Teknik Informatika)* Vol. 6 No. 2.

Purbolaksono, Mahendra Dwifebri, Muhammad Irvan Tantowi, Adnan Imam Hidayat, and Adiwijaya Adiwijaya. 2021. 'Perbandingan Support Vector Machine Dan Modified Balanced Random Forest Dalam Deteksi Pasien Penyakit Diabetes'. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)* 5(2): 393–99. doi:10.29207/resti.v5i2.3008.

Samsir, Ambiyar, Unung Verawardina, Firman Edi, Ronal Watrianthos. 2021. 'Analisis Sentimen Pembelajaran Daring Pada Twitter Di Masa COVID-19 Menggunakan Metode Naïve Bayes'. *JURNAL MEDIA INFORMATIKA BUDIDARMA* 5(1): 149. doi:10.30865/mib.v5i1.2604.

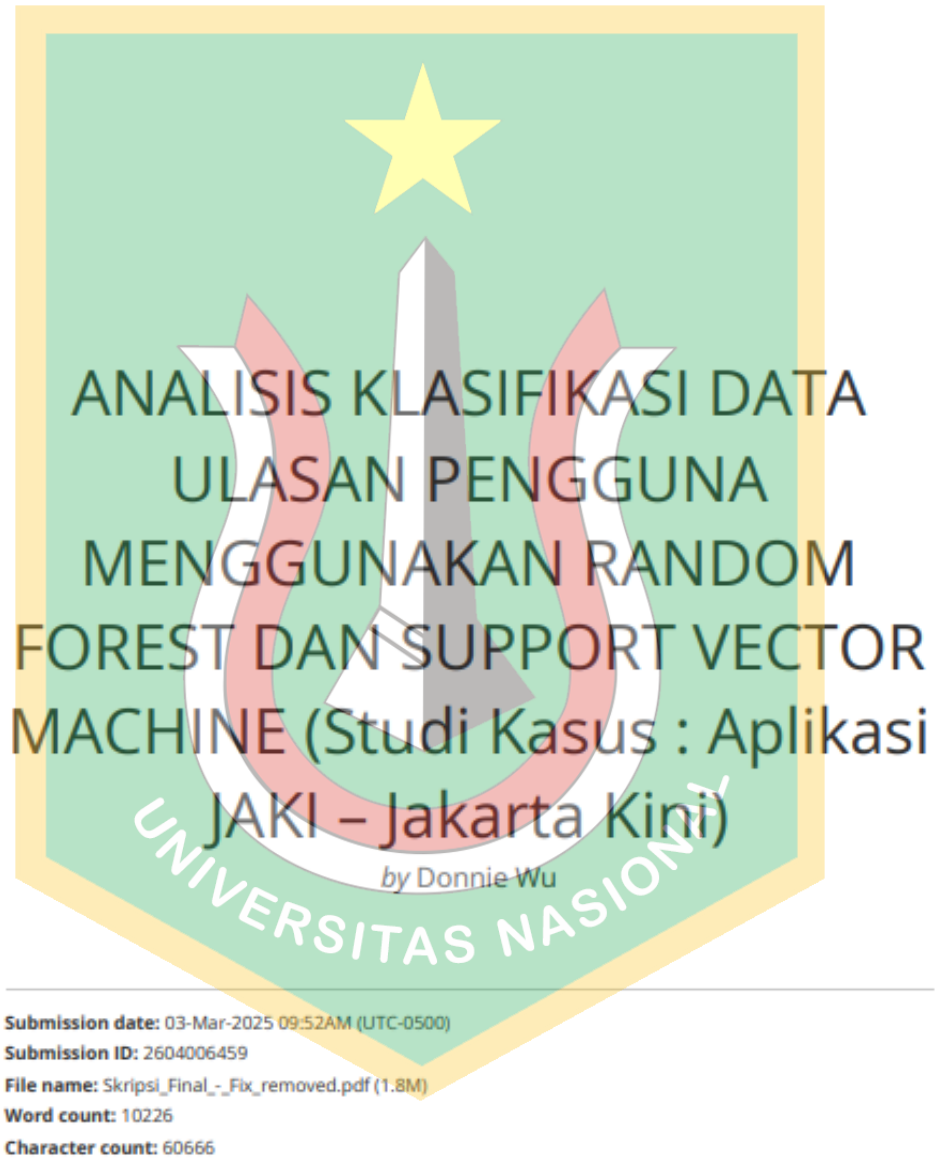
Suci Amaliah, Muhammad Nusrang, and Aswi Aswi. 2022a. 'Penerapan Metode Random Forest Untuk Klasifikasi Varian Minuman Kopi Di Kedai Kopi Konijiwa Bantaeng'. *VARIANSI: Journal of Statistics and Its application on Teaching and Research* 4(3): 121–27. doi:10.35580/variansiunm31.

Suci Amaliah, Muhammad Nusrang, and Aswi Aswi. 2022b. 'Penerapan Metode Random Forest Untuk Klasifikasi Varian Minuman Kopi Di Kedai Kopi Konijiwa Bantaeng'. *VARIANSI: Journal of Statistics and Its application on Teaching and Research* 4(3): 121–27. doi:10.35580/variansiunm31.

Yunita, Norma. 2016. ‘ANALISIS SENTIMEN BERITA ARTIS DENGAN MENGGUNAKAN ALGORITMA SUPPORT VECTOR MACHINE DAN PARTICLE SWARM OPTIMIZATION’. (AGUSTUS). www.tribunnews.com.

LAMPIRAN

Lampiran 1. Hasil Cek Plagiasi - Turnitin



ANALISIS KLASIFIKASI DATA ULASAN PENGGUNA MENGGUNAKAN RANDOM FOREST DAN SUPPORT VECTOR MACHINE (Studi Kasus : Aplikasi JAKI – Jakarta Kini)

ORIGINALITY REPORT

19%	16%	13%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

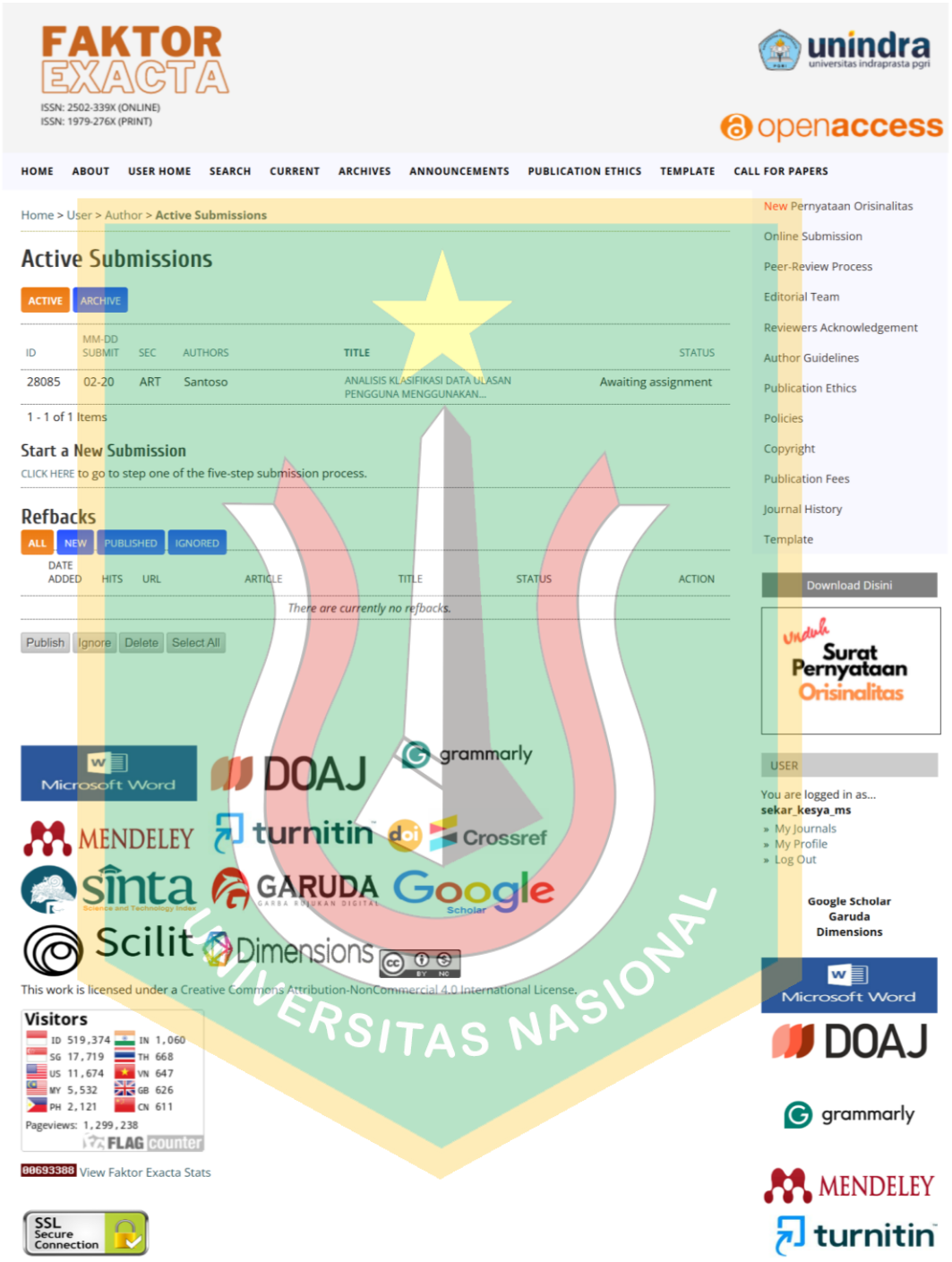
PRIMARY SOURCES

1	id.scribd.com Internet Source	1%
2	www.flr.at Internet Source	1%
3	doaj.org Internet Source	1%
4	repository.uin-suska.ac.id Internet Source	1%
5	Tarida Grace Wahyuni Margaretha Sidabutar, Didi Juardi. "ANALISIS SENTIMEN MASYARAKAT TERHADAP PENGGUNAAN HALODOC SEBAGAI LAYANAN TELEMEDICINE DI INDONESIA", Jurnal Informatika dan Teknik Elektro Terapan, 2025 Publication	1%
6	ejournal.medan.uph.edu Internet Source	<1%
7	Submitted to Universitas Maritim Raja Ali Haji Student Paper	<1%
8	Submitted to Universitas Brawijaya Student Paper	<1%
9	docplayer.info Internet Source	<1%

110	Ahmad Roihan, Po Abas Sunarya, Ageng Setiani Rafika. "Pemanfaatan Machine Learning dalam Berbagai Bidang: Review paper", IJCIT (Indonesian Journal on Computer and Information Technology), 2020 Publication	<1%
111	Cecep M Zakariya, Yulison Herry Chrisnanto, Gunawan Abdillah. "ANALISIS SENTIMEN TERKAIT PROGRAM KARTU PRAKERJA MENGGUNAKAN METODE K-NEAREST NEIGHBORS", Jurnal Informatika dan Rekayasa Elektronik, 2024 Publication	<1%
112	Herlawati Herlawati, Rahmadya Trias Handayanto, Prima Dina Atika, Fata Nidaul Khasanah et al. "Analisis Sentimen Pada Situs Google Review dengan Naïve Bayes dan Support Vector Machine", Jurnal Komtika (Komputasi dan Informatika), 2021 Publication	<1%
113	Ika Oktavia Suzanti, Fifin Ayu Mufarroha. "Implementasi Relevant Feedback Menggunakan Algoritma Genetika pada Dokumen Bahasa Indonesia (Implementation of Relevant Feedback Using Genetic Algorithm in Indonesian Documents)", JURNAL IPTEKKOM Jurnal Ilmu Pengetahuan & Teknologi Informasi, 2021 Publication	<1%
114	jurnal.polibatam.ac.id Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

Lampiran 2. Publikasi Jurnal



Lampiran 3. Kode Program Klasifikasi

```
!pip install google-play-scraper
!pip install nlp-id
!pip install --upgrade scikit-learn
!pip -q install sastrawi

import pandas as pd
import numpy as np
import re, string, unicodedata
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
nltk.download('stopwords')
import pickle
from google_play_scraper import app
from google_play_scraper import Sort, reviews
from nlp_id.lemmatizer import Lemmatizer
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score

"""## Scrapping Data"""

result, continuation_token = reviews(
    'id.go.jakarta.smartcity.jaki',
    lang='id',
    country='id',
    sort=Sort.MOST_RELEVANT,
    count=5000,
    filter_score_with=None
)

df = pd.DataFrame(np.array(result),columns=['review'])

df = df.join(pd.DataFrame(df.pop('review').tolist()))
```

```
df.head()

df.to_csv("scrapped_data.csv", index = False)

"""## Load Data"""

df = pd.read_csv('scrapped_data.csv')
df.head()

df['content']

"""## Text Preprocessing

### Case Folding
"""

def casefolding(text):
    text = text.lower()
    text = re.sub(r'[-+]?[0-9]+', '', text)
    text = re.sub(r'[^\w\s]', '', text)
    text = re.sub(r'\s+', ' ', text)
    text = text.strip()
    return text

raw_data = df['content'].iloc[65]
case_folding = casefolding(raw_data)

print('Raw Data \t\t  :', raw_data)
print('Case Folding \t\t  :', case_folding)

"""### Lematisasi"""

lemmatizer = Lemmatizer()

def lemmatizing(text):
    text = lemmatizer.lemmatize(text)
    return text

raw_data = df['content'].iloc[65]
case_folding = casefolding(raw_data)
lematisasi = lemmatizing(case_folding)

print('Raw Data \t\t  :', raw_data)
print('Case Folding \t\t  :', case_folding)
print('Lematisasi \t\t  :', lematisasi)
```

```
"""### Stemming"""

factory = StemmerFactory()
stemmer = factory.create_stemmer()

def stemming(text):
    text = stemmer.stem(text)
    return text

raw_data = df['content'].iloc[65]
case_folding = casefolding(raw_data)
lematisasi = lemmatizing(case_folding)
stem = stemming(lematisasi)

print('Raw Data \t\t  :', raw_data)
print('Case Folding \t\t  :', case_folding)
print('Lematisasi \t\t  :', lematisasi)
print('Stemming \t\t  :', stem)

"""### Slang Words Standarization"""

slang_dictionary =
pd.read_csv('https://raw.githubusercontent.com/nasalsabila/ka
mus-alay/master/colloquial-indonesian-lexicon.csv')
slang_dict =
pd.Series(slang_dictionary['formal'].values,index=slang_dicti
onary['slang']).to_dict()

def slangwords(text):
    for word in text.split():
        if word in slang_dict.keys():
            text = text.replace(word, slang_dict[word])
    return text

raw_data = df['content'].iloc[65]
case_folding = casefolding(raw_data)
lematisasi = lemmatizing(case_folding)
stem = stemming(lematisasi)
slangwords_standardization = slangwords (stem)

print('Raw Data \t\t\t  :', raw_data)
print('Case Folding \t\t\t  :', case_folding)
print('Lematisasi \t\t\t  :', lematisasi)
print('Stemming \t\t\t  :', stem)
```

```
print('Slangwords Standardization \t  :',
slangwords_standardization)

"""### Stopword Removal"""

stopwords_ind = stopwords.words('indonesian')

len(stopwords_ind)

stopwords_ind

def stopwords_removal(text):
    clean_words = []
    text = text.split()
    for word in text:
        if word not in stopwords_ind:
            clean_words.append(word)
    return " ".join(clean_words)

raw_data = df['content'].iloc[65]
case_folding = casefolding(raw_data)
lematisasi = lemmatizing(case_folding)
stem = stemming(lematisasi)
slangwords_standardization = slangwords (stem)
stopword_remove =
stopwords_removal(slangwords_standardization)

print('Raw Data \t\t\t  :', raw_data)
print('Case Folding \t\t\t  :', case_folding)
print('Lematisasi \t\t\t  :', lematisasi)
print('Stemming \t\t\t  :', stem)
print('Slangwords Standardization \t  :',
slangwords_standardization)
print('Stopword Removal \t\t  :', stopword_remove)

"""### Unwatedword Removal"""

unwatedword = ['aaaaaaapa', 'aaaaaaapasukan',
'aaaaaaapatria', 'aaaaamiin', 'aaahhhh', 'aaja', 'dong',
'aamata',  'agh', 'agr', 'ags', 'agustus', 'ah', 'sih', 'ya',
'no', 'nya', 'deh', 'aplikasi', 'app', 'apk', 'acu', 'aco',
'alpukat', 'anis', 'anies', 'aoa', 'apkikasi', 'apkjaki',
'apl', 'aplication', 'aplications', 'aplikaainyaa',
'aplikaksi', 'aplikas', 'aplikasi', 'aplikasibegitu',
'aplikasibenar', 'aplikasibergunanya', 'aplikasih',
```

```
'aplikasihanya', 'aplikasihh', 'aplikasihhnya',
'aplikasihkenapa', 'aplikasiklo', 'aplikasimasa',
'aplikasin', 'aplikasinyaa', 'aplikasinyw',
'aplikasipercumamalah', 'aplikasisemudah', 'aplikasitunggu',
'aplikeasi', 'aplikeasisaya', 'apliksi', 'aplisaki']

def unwantedword_removal(text):
    clean_words = []
    text = text.split()
    for word in text:
        if word not in unwatedword:
            clean_words.append(word)
    return " ".join(clean_words)

raw_data = df['content'].iloc[65]
case_folding = casefolding(raw_data)
lematisasi = lemmatizing(case_folding)
stem = stemming(lematisasi)
slangwords_standardization = slangwords (stem)
stopword_remove =
stopwords_removal(slangwords_standardization)
unwanted_removal = unwantedword_removal(stopword_remove)

print('Raw Data \t\t\t  :', raw_data)
print('Case Folding \t\t\t  :', case_folding)
print('Lematisasi \t\t\t  :', lematisasi)
print('Stemming \t\t\t  :', stem)
print('Slangwords Standardization \t  :',
slangwords_standardization)
print('Stopword Removal \t\t  :', stopword_remove)
print('Unwatedword Removal \t\t  :', unwanted_removal)

"""## Text Preprocessing Pipeline"""

def text_preprocessing_process(text):
    text = casefolding(text)
    text = slangwords(text)
    text = lemmatizing(text)
    text = stemming(text)
    text = stopwords_removal(text)
    text = unwantedword_removal(text)
    return text

# Commented out IPython magic to ensure Python compatibility.
# %%time
```

```
#
#
df['clean_teks']=df['content'].apply(text_preprocessing_proce
ss)

df[['content', 'clean_teks']]

"""### Split Word"""

df['clean_teks_split'] = df['clean_teks'].apply(lambda x: x
if isinstance(x, list) else x.split())

df[['content', 'clean_teks_split']]

"""## Labelling Ulasan"""

def analisis_sentimen(teks):
    df_positive = pd.read_csv('/content/positif_words.txt',
sep="\t")
    list_positive = list(df_positive.iloc[::,0])
    df_negative =
pd.read_csv('https://raw.githubusercontent.com/masdevid/ID-
Opinionwords/master/negative.txt', sep="\t")
    list_negative = list(df_negative.iloc[::,0])

    # Hitung skor sentimen
    score = 0
    for word in teks:
      if word in list_positive:
        score += 1
      elif word in list_negative:
        score -= 1

    # Tentukan polaritas
    if score > 0:
      polarity = "positif"
    else:
      polarity = "negatif"


    return score, polarity

df[['score', 'polarity']] =
df['clean_teks_split'].apply(lambda x:
pd.Series(analisis_sentimen(x)))
```

```
df['label'] = df['polarity'].apply(lambda x: 1 if x ==
'positif' else 0)

df[['clean_teks_split','score', 'polarity', 'label']]

df = df[~df['clean_teks_split'].apply(lambda x: isinstance(x,
list) and len(x) == 0)]
df = df.dropna(subset=['clean_teks_split'])

df[['clean_teks_split','score', 'polarity', 'label']]

df.polarity.value_counts()

fig, ax = plt.subplots(figsize=(6, 6))
sizes = [count for count in df['polarity'].value_counts()]
labels = list(df['polarity'].value_counts().index)
explode = [0.1, 0]
colors = ['#488ac7', '#84cfec']
ax.pie(x=sizes, labels=labels, colors=colors,
autopct='%1.1f%%', explode=explode, textprops={'fontsize':
14})
ax.set_title('Sentiment Polarity on Review Apps Data \n
(total 4500 review)', fontsize=16, pad=20)
plt.show()

"""## TF-IDF"""

X = df["clean_teks"]
y = df["label"]

vec_TF_IDF = TfidfVectorizer(ngram_range=(1,1))

vec_TF_IDF.fit(X)

X_tf_idf = vec_TF_IDF.transform(X)

pickle.dump(vec_TF_IDF.vocabulary_,open("feature_tf-
idf.sav","wb"))

vec_TF_IDF.vocabulary_


print(len(vec_TF_IDF.get_feature_names_out()))
```

```
features = list(vec_TF_IDF.get_feature_names_out())
print(features)

features = vec_TF_IDF.get_feature_names_out()
df_features = pd.DataFrame(features, columns=["Feature"])
df_features

# Mendapatkan daftar kata (fitur)
features = vec_TF_IDF.get_feature_names_out()

# Menghitung total bobot TF-IDF untuk setiap kata (sum dari
setiap kolom)
word_count = X_tf_idf.toarray().sum(axis=0)

# Membuat DataFrame dengan fitur dan jumlahnya
df_word_freq = pd.DataFrame({'Feature': features, 'Count':
word_count})

# Mengurutkan berdasarkan jumlah kemunculan kata
df_word_freq_sorted = df_word_freq.sort_values(by="Count",
ascending=False)

# Menampilkan hasil
print(df_word_freq_sorted)

X1 = vec_TF_IDF.transform(X).toarray()
data_tabular_tf_idf = pd.DataFrame(X1,
columns=vec_TF_IDF.get_feature_names_out())
data_tabular_tf_idf

"""## Feature Selection"""

X_trains = np.array(data_tabular_tf_idf)
y_trains = np.array(y)

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

chi2_features = SelectKBest(chi2, k=5000)
X_kbest_features = chi2_features.fit_transform(X_trains,
y_trains)

print('Original Fearures Number', X_trains.shape[1])
print('Reduced Fearures Number', X_kbest_features.shape[1])
```

```
data =
pd.DataFrame(chi2_features.scores_,columns=['total_bobot'])
data

feature = vec_TF_IDF.get_feature_names_out()

data['fitur'] = feature
data['fitur']

data[['fitur', 'total_bobot']]

df_sorted = data.sort_values(by='total_bobot',
ascending=False)

df_sorted['total_bobot'] = np.round(df_sorted['total_bobot'],
5)

top_20 = df_sorted.head(20)
print("20 Nilai Tertinggi:")
print(top_20)

bottom_20 = df_sorted.tail(20)
print("20 Nilai Terendah:")
print(bottom_20)

mask = chi2_features.get_support()

new_feature = []
for bool, f in zip(mask, feature):
    if bool :
        new_feature.append(f)
    selected_feature=new_feature
selected_feature

new_selected_feature = {}

for (k,v) in vec_TF_IDF.vocabulary_.items():
    if k in selected_feature:
        new_selected_feature[k]=v
new_selected_feature

len(new_selected_feature)

pickle.dump(new_selected_feature,
open("new_selected_feature_tf-idf.sav","wb"))
```

```
data_selected_feature = pd.DataFrame(X_kbest_features,
columns=selected_feature)
data_selected_feature


"""## Permodelan Klasifikasi"""

seleted_X = X_kbest_features

X = seleted_X
y = df["label"]

import random
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)


print("Total Keseluruhan Data : ", len(X))
print("Total Data Latih X     : ", len(X_train))
print("Total Data Test  X     : ", len(X_test))
print("Total Data Latih Y     : ", len(y_train))
print("Total Data Test  Y     : ", len(y_test))

"""### Random Forest"""

# Definisikan model Random Forest
rf = RandomForestClassifier(random_state=42)

# Melatih model Random Forest dengan data training
model_rf = rf.fit(X_train, y_train)

# Prediksi dengan Random Forest pada data uji
y_pred_rf = rf.predict(X_test)

data_input_rf = ("Saya mau vaksin yang ke 3 tp terjadi
kesalahan trs & harus log in..padahal nik KTP sudah
terverifikasi, cek status vaksin juga ga bisa kebuka.. udah
brp x d uninstall dulu, trs d install lagi tp ttp ga bisa d
buka 😖 aneh ini aplikasi, bukan nya membantu malah
menyulitkan .. ")
data_input_rf = text_preprocessing_process(data_input_rf)
```

```
#load
tfidf = TfidfVectorizer

loaded_vec = TfidfVectorizer(decode_error="replace",
vocabulary=set(pickle.load(open("new_selected_feature_tf-
idf.sav", "rb"))))

hasil =
model_rf.predict(loaded_vec.fit_transform([data_input_rf]))

if (hasil == 0):
    kategori = "Negatif"
elif(hasil == 1):
    kategori = "Positif"
else:
    kategori = "Negatif"

print("Hasil Prediksi : \n", kategori)

"""#### Confusion Matrix"""

print("Random Forest Classification Report : \n",
classification_report(y_test, y_pred_rf))

# Confusion Matrix
conf_matrix_rf = confusion_matrix(y_test, y_pred_rf)
plt.figure(figsize=(7, 5))
sns.heatmap(conf_matrix_rf, annot=True, fmt="d",
cmap="Blues")
plt.title("Confusion Matrix - Random Forest")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()

TN = conf_matrix_rf[0, 0]  # True Negative
FP = conf_matrix_rf[0, 1]  # False Positive
FN = conf_matrix_rf[1, 0]  # False Negative
TP = conf_matrix_rf[1, 1]  # True Positive

# Menampilkan hasil
print(f"True Positive (TP): {TP}")
print(f"True Negative (TN): {TN}")
print(f"False Positive (FP): {FP}")
print(f"False Negative (FN): {FN}")
```

```
# Menghitung correct dan incorrect predictions
correct_predictions = conf_matrix_rf[0, 0] +
conf_matrix_rf[1, 1]  # TN + TP
incorrect_predictions = conf_matrix_rf[0, 1] +
conf_matrix_rf[1, 0]  # FN + FP

# Membuat bar chart untuk correct vs incorrect predictions
categories = ["Correct Predictions", "Incorrect Predictions"]
values = [correct_predictions, incorrect_predictions]
colors = ['#488ac7', "#f8aba0"]

plt.figure(figsize=(7, 5))
plt.bar(categories, values, color=colors, width=0.6)
plt.title("Correct vs Incorrect Predictions Model Random
Forest")
plt.xlabel("Prediction Type")
plt.ylabel("Count")
plt.show()

"""### Support Vector Machine"""

# Definisikan model SVM
svm = SVC()

# Tidak perlu konversi jika X_train sudah dalam bentuk numpy
array
model_svm = svm.fit(X_train, y_train)

y_pred_svm = svm.predict(X_test)

data_input_svm = ("Saya mau vaksin yang ke 3 tp terjadi
kesalahan trs & harus log in..padahal nik KTP sudah
terverifikasi, cek status vaksin juga ga bisa kebuka.. udah
brp x d uninstall dulu, trs d install lagi tp ttp ga bisa d
buka 😔 aneh ini aplikasi, bukan nya membantu malah
menyulitkan .. ")
data_input_svm = text_preprocessing_process(data_input_svm)

#load
tfidf_svm = TfidfVectorizer

loaded_vec_svm = TfidfVectorizer(decode_error="replace",
vocabulary=set(pickle.load(open("new_selected_feature_tf-
idf.sav", "rb"))))
```

```
hasil_svm =
model_svm.predict(loaded_vec_svm.fit_transform([data_input_sv
m]).toarray())

if (hasil_svm == 0):
    kategori_svm = "Negatif"
elif(hasil_svm == 1):
    kategori_svm= "Positif"
else:
    kategori_svm = "Negatif"

print("Hasil Prediksi : \n", kategori_svm)

print("Support Vector Machine Classification Report : \n",
classification_report(y_test, y_pred_svm))

# Confusion Matrix
conf_matrix_svm = confusion_matrix(y_test, y_pred_svm)
plt.figure(figsize=(7, 5))
sns.heatmap(conf_matrix_svm, annot=True, fmt="d",
cmap="Blues")
plt.title("Confusion Matrix - Random Support Vector Machine")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()

TN = conf_matrix_svm[0, 0]  # True Negative
FP = conf_matrix_svm[0, 1]  # False Positive
FN = conf_matrix_svm[1, 0]  # False Negative
TP = conf_matrix_svm[1, 1]  # True Positive

# Menampilkan hasil
print(f"True Positive (TP): {TP}")
print(f"True Negative (TN): {TN}")
print(f"False Positive (FP): {FP}")
print(f"False Negative (FN): {FN}")

# Menghitung correct dan incorrect predictions
correct_predictions = conf_matrix_svm[0, 0] +
conf_matrix_svm[1, 1]  # TN + TP
incorrect_predictions = conf_matrix_svm[0, 1] +
conf_matrix_svm[1, 0]  # FN + FP

# Membuat bar chart untuk correct vs incorrect predictions
categories = ["Correct Predictions", "Incorrect Predictions"]
```

```
values = [correct_predictions, incorrect_predictions]
colors = ['#488ac7', "#f8aba0"]

plt.figure(figsize=(7, 5))
plt.bar(categories, values, color=colors, width=0.6)
plt.title("Correct vs Incorrect Predictions Model SVM")
plt.xlabel("Prediction Type")
plt.ylabel("Count")
plt.show()

# Konversi label y_test menjadi numerik
le = LabelEncoder()
y_test_numeric = le.fit_transform(y_test)

# Konversi prediksi menjadi numerik
y_pred_rf_numeric = le.transform(y_pred_rf)
y_pred_svm_numeric = le.transform(y_pred_svm)

# Menghitung metrik untuk Random Forest
rf_metrics = [
    accuracy_score(y_test_numeric, y_pred_rf_numeric),
    precision_score(y_test_numeric, y_pred_rf_numeric),
    recall_score(y_test_numeric, y_pred_rf_numeric),
    f1_score(y_test_numeric, y_pred_rf_numeric)
]

# Menghitung metrik untuk SVM
svm_metrics = [
    accuracy_score(y_test_numeric, y_pred_svm_numeric),
    precision_score(y_test_numeric, y_pred_svm_numeric),
    recall_score(y_test_numeric, y_pred_svm_numeric),
    f1_score(y_test_numeric, y_pred_svm_numeric)
]

# Menyiapkan plot bar
metrics = ['Accuracy', 'Precision', 'Recall', 'F1-Score']
x = np.arange(len(metrics))
width = 0.35

palette = sns.color_palette("Blues")

# Plot bar
fig, ax = plt.subplots(figsize=(10, 6))
rects1 = ax.bar(x - width/2, rf_metrics, width, label='Random
Forest', color=palette[0])
```

```
rects2 = ax.bar(x + width/2, svm_metrics, width,
label='Support Vector Machine', color=palette[1])

ax.set_ylabel('Scores')
ax.set_title('Random Forest vs Support Vector Machine')
ax.set_xticks(x)
ax.set_xticklabels(metrics)
ax.legend()

# Fungsi untuk menambahkan label pada bar chart
def add_labels(rects):
    for rect in rects:
        height = rect.get_height()
        ax.annotate(f'{height:.3f}',
                    xy=(rect.get_x() + rect.get_width() / 2,
height),
                    xytext=(0, 3),
                    textcoords="offset points",
                    ha='center', va='bottom')

# Menambahkan label ke masing-masing bar
add_labels(rects1)
add_labels(rects2)

plt.tight_layout()
plt.show()

"""## Menyimpan Model"""

pickle.dump(model_rf, open("model_rf.sav","wb"))

pickle.dump(model_svm, open("model_svm.sav", "wb"))
```

Lampiran 4. Kode Program Sistem Streamlit

```
import pickle
import streamlit as st
import pandas as pd
import plotly.express as px
import os
import warnings
from st_aggrid import AgGrid, GridOptionsBuilder
from streamlit_option_menu import option_menu
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from st_aggrid import AgGrid, GridOptionsBuilder
from PIL import Image

with st.sidebar:
    menu = option_menu(
        "Menu",
        ["Dashboard", "Dataset", "Klasifikasi", "Metode Klasifikasi",
     "Implementasi Algoritma"],
        default_index=0,
        styles={
            "container": {
                "width": "100%",
                "justify-content": "align left"
            },
            "nav-link": {
                "font-size": "18px",
                "padding": "12px 18px",
                "margin": "5px",
                "text-align": "align left",
                "white-space": "nowrap",
            },
            "nav-link-selected": {
                "background-color": "#6A89A7",
                "color": "white",
                "border-radius": "8px",
                "padding": "12px 18px",
                "margin": "5px",
            },
        }
    )
```

```
if menu == "Dashboard":
    df = pd.read_csv("C:/SkripsiApp/dataset_bersih.csv")

    total_reviews = len(df)
    positive_reviews = (df['polarity'] == 'positif').sum()
    negative_reviews = (df['polarity'] == 'negatif').sum()

    rf_correct, rf_incorrect = 826, 77
    svm_correct, svm_incorrect = 832, 71

    positive_review = " ".join(df[df['polarity'] ==
     'positif']['clean_teks_split'])
    negative_review = " ".join(df[df['polarity'] ==
     'negatif']['clean_teks_split'])


    total1, total2, total3 = st.columns(3, gap='small')
    with total1:
        st.markdown(f"""
            <div style='border: 1px solid white; padding: 1px;
     border-radius: 1px; background-color: #6A89A7;'>
                <h3 style='text-align: center;  font-size: 18px;
     '>Total Ulasan</h3>
            </div>
            <div style='border: 1px solid white; padding: 1px;
     border-radius: 1px;'>
                <h4 style='text-align: center;'>{total_reviews}</h4>
            </div>
            """, unsafe_allow_html=True)

    with total2:
        st.markdown(f"""
            <div style='border: 1px solid white; padding: 1px;
     border-radius: 1px; background-color: #6A89A7;'>
                <h3 style='text-align: center; font-size: 18px;
     '>Ulasan Positif</h3>
            </div>
            <div style='border: 1px solid white; padding: 1px;
     border-radius: 1px;'>
                <h4 style='text-align:
     center;'>{positive_reviews}</h4>
            </div>
            """, unsafe_allow_html=True)

    with total3:
```

```
    st.markdown(f"""
        <div style='border: 1px solid white; padding: 1px;
 border-radius: 1px; background-color: #6A89A7;'>
            <h3 style='text-align: center;  font-size: 18px;
 '>Ulasan Negatif</h3>
        </div>
        <div style='border: 1px solid white; padding: 1px;
 border-radius: 1px;'>
            <h4 style='text-align:
 center;'>{negative_reviews}</h4>
        </div>
        """, unsafe_allow_html=True)

st.write('')

chart1, chart2, chart3 = st.columns(3, gap='small')
# ------------ CHART 1: Pie Chart Sentimen Polarity ------------
with chart1:
    st.markdown("""
        <div style="border: 1px solid white; padding: 1px;
 border-radius: 1px; background-color: #6A89A7;">
            <h4 style="text-align: center;">Sentimen
 Polarity</h4>
        </div>
    """, unsafe_allow_html=True)

    fig1, ax1 = plt.subplots(figsize=(5, 5))

    sizes = df['polarity'].value_counts().tolist()
    labels = df['polarity'].value_counts().index.tolist()
    explode = [0.1, 0]
    colors = ['#66b3ff', '#ffcc99']

    fig1.patch.set_facecolor('none')
    ax1.set_facecolor('none')

    wedges, texts, autotexts = ax1.pie(
        x=sizes, labels=labels, colors=colors, autopct='%1.1f%%',
        explode=explode, textprops={'fontsize': 12},
        wedgeprops={'edgecolor': 'white', 'linewidth': 1.5}
    )

    for text in texts + autotexts:
        text.set_color("white")
```

```
        st.pyplot(fig1)
        st.markdown('</div>', unsafe_allow_html=True)



    # ------------ CHART 2: Bar Chart Correct vs Incorrect (RF) -----
     -------
    with chart2:
        st.markdown("""
            <div style="border: 1px solid white; padding: 1px;
     border-radius: 1px; background-color: #6A89A7;">
                <h4 style="text-align: center;">Prediksi RF</h4>
            </div>
        """, unsafe_allow_html=True)

        fig2, ax2 = plt.subplots(figsize=(5, 5))
        bars = ax2.bar(['Correct', 'Incorrect'], [rf_correct,
     rf_incorrect], color=['#66b3ff', '#ffcc99'], width=0.6,
     edgecolor='white', linewidth=1.5)

        for bar in bars:
            height = bar.get_height()
            ax2.text(bar.get_x() + bar.get_width()/2, height + 10,
     f"{height}", ha='center', color='white', fontsize=12,
     fontweight='bold')

        ax2.set_ylabel("Jumlah", fontsize=12, color="white")
        ax2.set_facecolor('none')
        fig2.patch.set_facecolor('none')
        ax2.spines['top'].set_color('white')
        ax2.spines['right'].set_color('white')
        ax2.spines['left'].set_color('white')
        ax2.spines['bottom'].set_color('white')
        ax2.tick_params(colors='white', labelsize=12)
        ax2.yaxis.grid(color='gray', linestyle='dashed',
     linewidth=0.5, alpha=0.5)
        st.pyplot(fig2)


    # ------------ CHART 3: Bar Chart Correct vs Incorrect (SVM) ----
     --------
    with chart3:
        st.markdown("""
```

```
        <div style="border: 1px solid white; padding: 1px;
    border-radius: 1px; background-color: #6A89A7;">
            <h4 style="text-align: center;">Prediksi SVM</h4>
        </div>
    """, unsafe_allow_html=True)
    fig3, ax3 = plt.subplots(figsize=(5, 5))
    bars = ax3.bar(['Correct', 'Incorrect'], [svm_correct,
    svm_incorrect], color=['#66b3ff', '#ffcc99'], width=0.6,
    edgecolor='white', linewidth=1.5)

    for bar in bars:
        height = bar.get_height()
        ax3.text(bar.get_x() + bar.get_width()/2, height + 10,
    f"{height}", ha='center', color='white', fontsize=12,
    fontweight='bold')

    ax3.set_ylabel("Jumlah", fontsize=12, color="white")
    ax3.set_facecolor('none')
    fig3.patch.set_facecolor('none')

    ax3.spines['top'].set_color('white')
    ax3.spines['right'].set_color('white')
    ax3.spines['left'].set_color('white')
    ax3.spines['bottom'].set_color('white')
    ax3.tick_params(colors='white', labelsize=12)
    ax3.yaxis.grid(color='gray', linestyle='dashed',
    linewidth=0.5, alpha=0.5)
    st.pyplot(fig3)


word_col1, word_col2 = st.columns(2, gap='small')

# ------------ WORD CLOUD 1: Ulasan Positif ------------
with word_col1:
    st.markdown("""
        <div style='border: 1px solid white; padding: 1px;
     border-radius: 1px; background-color: #6A89A7;'>
            <h4 style='text-align: center; color: white;'>Word
     Cloud Ulasan Positif</h4>
        </div>
    """, unsafe_allow_html=True)

    fig1, ax1 = plt.subplots(figsize=(5, 5))
    wordcloud_pos = WordCloud(
        width=400, height=400, colormap='YlOrBr',
```

```
            max_words=50, contour_color='white', contour_width=1
        ).generate(positive_review)

        ax1.imshow(wordcloud_pos, interpolation='bilinear')
        ax1.axis("off")
        st.pyplot(fig1)

    # ------------ WORD CLOUD 2: Ulasan Negatif ------------
    with word_col2:
        st.markdown("""
            <div style='border: 1px solid white; padding: 1px;
     border-radius: 1px; background-color: #6A89A7;'>
                <h4 style='text-align: center; color: white;'>Word
     Cloud Ulasan Negatif</h4>
            </div>
        """, unsafe_allow_html=True)

        fig2, ax2 = plt.subplots(figsize=(5, 5))
        wordcloud_neg = WordCloud(
            width=400, height=400, background_color='black',
     colormap='Reds',
            max_words=50, contour_color='white', contour_width=1
        ).generate(negative_review)

        ax2.imshow(wordcloud_neg, interpolation='bilinear')
        ax2.axis("off")
        st.pyplot(fig2)


elif menu == "Klasifikasi":
    st.title("📝 Klasifikasi Data Ulasan")
    st.write("___")
    with st.container():
        selected = option_menu(
            menu_title=None,
            options=['Random Forest', 'SVM'],
            icons=['tree', 'cpu'],
            orientation='horizontal',
            styles={
                "container": {"width": "100%", "justify-content":
     "center"},
                "nav-link": {
                    "font-size": "14px",
                    "padding": "8px 15px",
                    "margin": "0px",
```

```
                "text-align": "center",
                "white-space": "nowrap",
            },
            "nav-link-selected": {
                "background-color": "#6A89A7",
                "color": "white",
                "border-radius": "5px",
                "padding": "8px 15px",
                "margin": "0px",
            },
        }
    )

    if selected == 'Random Forest':
        model_rf_fraud = pickle.load(open('model_rf (1).sav',
'rb'))
        loaded_vec = TfidfVectorizer(decode_error="replace",
vocabulary=set(pickle.load(open("new_selected_feature_tf-idf
(1).sav", "rb"))))
        st.title("Menu Klasifikasi - Random Forest")
        clean_teks_rf = st.text_input('Masukan Teks Ulasan -
Random Forest', key="rf_input")
        fraud_detection_rf = ''

        if st.button('Hasil Deteksi - Random Forest',
key="rf_button"):
            predict_fraud_rf =
model_rf_fraud.predict(loaded_vec.fit_transform([clean_teks_rf])
.toarray())

            if predict_fraud_rf == 0:
                fraud_detection_rf = 'Ulasan Negatif'
            elif predict_fraud_rf == 1:
                fraud_detection_rf = 'Ulasan Positif'
            else:
                fraud_detection_rf = 'Hasil tidak diketahui'
        st.success(fraud_detection_rf)

    elif selected == 'SVM':
        model_svm_fraud = pickle.load(open('model_svm (1).sav',
'rb'))
        loaded_vec_svm = TfidfVectorizer(decode_error="replace",
vocabulary=set(pickle.load(open("new_selected_feature_tf-idf
(1).sav", "rb"))))
```

```
            st.title("Menu Klasifikasi - SVM")
            clean_teks_svm = st.text_input('Masukan Teks Ulasan -
    SVM', key="svm_input")
            fraud_detection_svm = ''

            if st.button('Hasil Deteksi - SVM', key="svm_button"):
                predict_fraud_svm =
    model_svm_fraud.predict(loaded_vec_svm.fit_transform([clean_teks
    _svm]).toarray())

                if predict_fraud_svm == 0:
                    fraud_detection_svm = 'Ulasan Negatif'
                elif predict_fraud_svm == 1:
                    fraud_detection_svm = 'Ulasan Positif'
                else:
                    fraud_detection_svm = 'Hasil tidak diketahui'
            st.success(fraud_detection_svm)


elif menu == "Dataset":
    st.title("💾 Dataset")
    st.markdown("""
            <hr style="border: 3px solid white; box-shadow: 0 4px 6px
     -4px gray;">
            """, unsafe_allow_html=True)

    file_dataset = 'C:\SkripsiApp\sentiment_analysis.csv'
    dataset = pd.read_csv(file_dataset)

    df_ulasan = pd.DataFrame(dataset)

    st.write("Pada dataset ini terdapat 4500 data. Kemudian Dataset
     melalui tahapan Text Preprocessing dan menghasilkan 4382 baris
     data. Data yang sudah melewati tahap Preprocessing kemudian
     dibagi menjadi Data Latih dan Data Uji, dengan proporsi 80:20.
     Untuk data latih sebesar 80% dan data uji sebesar 20%")

    st.write("Sumber Data:
     https://play.google.com/store/apps/details?id=id.go.jakarta.smar
     tcity.jaki&hl=id")
    st.write("___")
    st.subheader("Dataset Ulasan")

    st.table(df_ulasan[['content']].head(10))
```

```
    st.write("___")
    st.subheader("Dataset Ulasan Clean")
    st.table(df_ulasan[['clean_teks_split', 'score', 'label',
     'polarity']].head(10))



elif menu == "Metode Klasifikasi":
    st.title("🤖 Metode Klasifikasi")
    st.write("___")

    with st.container():
        selected = option_menu(
            menu_title=None,
            options=['Random Forest', 'SVM'],
            icons=['tree', 'cpu'],
            orientation='horizontal',
            styles={
                "container": {"width": "100%", "justify-content":
     "center"},
                "nav-link": {
                    "font-size": "14px",
                    "padding": "8px 15px",
                    "margin": "0px",
                    "text-align": "center",
                    "white-space": "nowrap",
                },
                "nav-link-selected": {
                    "background-color": "#6A89A7",
                    "color": "white",
                    "border-radius": "5px",
                    "padding": "8px 15px",
                    "margin": "0px",
                },
            }
        )

        if selected == 'Random Forest':
            st.subheader("🌳 Random Forest")
            st.write("___")
            st.write("""
            Random Forest (RF) merupakan algoritma ensemble yang
     bekerja dengan menggabungkan banyak pohon keputusan (decision
     tree) untuk menghasilkan prediksi yang lebih akurat dan stabil.
```

```
Kelebihan utama RF terletak pada kemampuannya menghasilkan
akurasi tinggi karena mengurangi risiko overfitting melalui
teknik bagging (Bootstrap Aggregating). RF juga tahan terhadap
outlier dan noise karena prediksi akhir dihasilkan dari rata-
rata atau voting mayoritas dari banyak pohon, sehingga model
tetap stabil meskipun ada data anomali.
        """)

        st.latex(r"""
            Gini(S_i) = 1 - \sum_{i=0}^{c-1} p_i^2
        """)

        st.write("Keterangan: ")
        st.latex(r"""
            \begin{aligned}
            & S   &\quad&: \text{Himpunan data} \\
            & 1   &\quad&: \text{Total kemurnian} \\
            & p_i &\quad&: \text{Proporsi sampel dalam kelas }
c_i \\
            & c   &\quad&: \text{Jumlah kelas dalam dataset} \\
            & i   &\quad&: \text{(Perhitungan dimulai dari kelas
pertama hingga kelas ke-} c)
            \end{aligned}
            """)

        st.latex(r"""
            Gini_{split} = \sum_{i=0}^{c-1} \left( \frac{n_i}{n}
\times Gini(S_i) \right)
        """)

        st.write("Keterangan")
        st.latex(r"""
            \begin{aligned}
            & c         &\quad&: \text{Jumlah subset setelah
pemisahan} \\
            & n_i       &\quad&: \text{Jumlah sampel di subset }
S_i \\
            & n         &\quad&: \text{Jumlah total sampel
sebelum pemisahan} \\
            & Gini(S_i) &\quad&: \text{Nilai Indeks Gini dari
subset } S_i
            \end{aligned}
            """)
```

```
        st.markdown("""
        Setelah pohon keputusan dalam Random Forest terbentuk,
proses prediksi dilakukan dengan menggabungkan hasil dari semua
pohon dalam hutan. Setiap pohon menghasilkan prediksi \( h_i (x)
\) untuk input \( x \), dan hasil akhir ditentukan berdasarkan
voting mayoritas dalam kasus klasifikasi atau rata-rata prediksi
dalam kasus regresi (Canty, 2002).
        """)

        st.latex(r"""
            y = \text{mode}\{h_1 (x), h_2 (x), \dots, h_M (x)\}
        """)

        st.write("Keterangan: ")
        st.latex(r"""
            \begin{aligned}
            & y           &\quad&: \text{Prediksi akhir dari
Random Forest} \\
            & h_i (x)     &\quad&: \text{Prediksi dari pohon
keputusan ke-} i \\
            & M           &\quad&: \text{Total jumlah pohon dalam
hutan} \\
            & \text{mode} &\quad&: \text{Fungsi yang memilih
prediksi yang paling sering muncul}
            \end{aligned}
            """)


    elif selected == 'SVM':
        st.subheader("⚙️ Support Vector Machine")
        st.write("___")
        st.write("""
        Sistem klasifikasi Support Vector Machine atau SVM
dikembangkan oleh seorang matematikawan asal Rusia bernama
Vladimir Vapnik. Support Vector Machines (SVM) adalah algoritma
machine learning yang kuat yang dapat digunakan untuk
klasifikasi atau regresi. SVM dapat digunakan untuk berbagai
masalah, seperti analisis sentimen (Aisah et al., 2023).
Algoritma permodelan Support Vector Machine (SVM) dirumuskan
sebagai berikut:
        """)

        st.latex(r"""
            f(x) = w^T x + b
```

```
            """)
            st.write("Keterangan: ")
            st.latex(r"""
                \begin{aligned}
                & f(x) &\quad&: \text{Output fungsi (prediksi atau
    skor keputusan)} \\
                & w^T  &\quad&: \text{Vektor bobot transpos
    (menentukan kontribusi setiap fitur)} \\
                & x    &\quad&: \text{Vektor fitur (input data yang
    akan diproses)} \\
                & b    &\quad&: \text{Bias (nilai konstanta yang
    menggeser keputusan model)}
                \end{aligned}
                """)


elif menu == "Implementasi Algoritma":
    st.title("🔬 Implementasi Algoritma")
    st.write("___")

    with st.container():
        selected = option_menu(
            menu_title=None,
            options=['Confusion Matrix', 'Random Forest', 'SVM',
    'Perbandingan'],
            icons=['bar-chart', 'tree', 'cpu', 'clipboard'],
            orientation='horizontal',
            styles={
                "container": {"width": "100%", "justify-content":
    "center"},
                "nav-link": {
                    "font-size": "14px",
                    "padding": "8px 15px",
                    "margin": "0px",
                    "text-align": "center",
                    "white-space": "nowrap",
                },
                "nav-link-selected": {
                    "background-color": "#6A89A7",
                    "color": "white",
                    "border-radius": "5px",
                    "padding": "8px 15px",
                    "margin": "0px",
                },
```

```
        }
    )

    if selected == 'Confusion Matrix':
        st.subheader("📊 Confusion Matrix")
        st.write("    ")
        st.write("Untuk mengukur kinerja Algoritma Random Forest
dan Support Vector Machine, dalam tahapan evaluasi model dengan
mengunakan metode Confusion Matrix. Confusion Matrix berguna
untuk memahami perbedaan antar kelas dalam klasifikasi. Matriks
ini memberikan gambaran yang jelas tentang bagaimana hasil
prediksi model dibandingkan dengan nilai aktual (ground truth)
pada dataset yang diuji. Confusion Matrix terdiri dari empat
komponen utama yaitu True Positive (TP), True Negative (TN),
False Positive (FP), dan False Negative (FN).")

        centered_table_css = """
            <style>
            .center-table {
                display: flex;
                justify-content: center;
                margin-top: 20px;
            }
            table {
                border-collapse: collapse;
                width: 50%;
                text-align: center;

                border: 1px solid #ddd;
            }
            th, td {
                padding: 12px;
                border: 1px solid #ddd;
            }
            th {
                background-color: #6A89A7;
                color: white;
            }
            </style>
        """

        # Tabel Confusion Matrix
        confusion_matrix_html = """
        <div class="center-table">
            <table>
```

```
                <tr>
                    <th>Prediksi \\ Aktual</th>
                    <th>Positif (P)</th>
                    <th>Negatif (N)</th>
                </tr>
                <tr>
                    <td><b>Positif (P')</b></td>
                    <td>True Positive (TP)</td>
                    <td>False Positive (FP)</td>
                </tr>
                <tr>
                    <td><b>Negatif (N')</b></td>
                    <td>False Negative (FN)</td>
                    <td>True Negative (TN)</td>
                </tr>
            </table>
        </div>
        """

        st.markdown(centered_table_css, unsafe_allow_html=True)
        st.markdown(confusion_matrix_html,
unsafe_allow_html=True)

        st.write(" ")
        st.write("   ")
        st.write("___")

        # Penjelasan Komponen
        st.markdown("### 📌 Penjelasan Komponen:")

        # True Positive (TP)
        st.markdown("- **True Positive (TP):**")
        st.markdown("<div style='text-indent: 30px;'>Prediksi
positif yang benar (model memprediksi positif, dan kenyataannya
memang positif).</div>", unsafe_allow_html=True)
        st.latex(r"TP = \text{Jumlah kasus positif yang
diprediksi benar}")

        # False Positive (FP)
        st.markdown("- **False Positive (FP):**")
        st.markdown("<div style='text-indent: 30px;'>Prediksi
positif yang salah (model memprediksi positif, tetapi
kenyataannya negatif).</div>", unsafe_allow_html=True)
        st.latex(r"FP = \text{Jumlah kasus negatif yang salah
diprediksi positif}")
```

```
        # False Negative (FN)
        st.markdown("- **False Negative (FN):**")
        st.markdown("<div style='text-indent: 30px;'>Prediksi
negatif yang salah (model memprediksi negatif, tetapi
kenyataannya positif).</div>", unsafe_allow_html=True)
        st.latex(r"FN = \text{Jumlah kasus positif yang salah
diprediksi negatif}")

        # True Negative (TN)
        st.markdown("- **True Negative (TN):**")
        st.markdown("<div style='text-indent: 30px;'>Prediksi
negatif yang benar (model memprediksi negatif, dan kenyataannya
memang negatif).</div>", unsafe_allow_html=True)
        st.latex(r"TN = \text{Jumlah kasus negatif yang
diprediksi benar}")

        st.write("___")

        # Rumus Evaluasi Berdasarkan Confusion Matrix
        st.markdown("### ☑ Rumus Evaluasi Berdasarkan Confusion
Matrix")

        # 1. Accuracy
        st.markdown("1. **Accuracy (Akurasi):**")
        st.markdown("<div style='text-indent: 30px;'> Mengukur
seberapa sering model membuat prediksi yang benar. </div>",
unsafe_allow_html=True)
        st.latex(r"Accuracy = \frac{TP + TN}{TP + TN + FP + FN}")

        # 2. Precision
        st.markdown("2. **Precision (Presisi):**")
        st.markdown("<div style='text-indent: 30px;'> Mengukur
proporsi prediksi positif yang benar. </div>",
unsafe_allow_html=True)
        st.latex(r"Precision = \frac{TP}{TP + FP}")

        # 3. Recall
        st.markdown("3. **Recall (Sensitivitas / TPR):**")
        st.markdown("<div style='text-indent: 30px;'> Mengukur
seberapa baik model dalam menemukan semua kasus positif.
</div>", unsafe_allow_html=True)
        st.latex(r"Recall = \frac{TP}{TP + FN}")

        # 4. F1-Score
```

```
        st.markdown("4. **F1-Score:**")
        st.markdown("<div style='text-indent: 30px;'> Harmonik
rata-rata dari Precision dan Recall. </div>",
unsafe_allow_html=True)
        st.latex(r"F1 = 2 \times \frac{Precision \times
Recall}{Precision + Recall}")


    elif selected == 'Random Forest':
        st.header("🌳 Pengujian Random Forest")
        st.markdown("""
        <hr style="border: 3px solid white; box-shadow: 0 4px 6px
-4px gray;">
        """, unsafe_allow_html=True)

        st.subheader("Confusion Matrix Random Forest")
        st.write('Pada implementasi Random Forest data training
yang di gunakan sebanyak 3505 data dan untuk testing digunakan
sebanyak 877 data. hasil pengujian yang diimplementasikan pada
metode Random Forest ditujukan dalam bentuk tabel Confusion
Matrix. Berikut adalah hasil pengujian yang didapatkan:    ')
        img6 = Image.open("C:\SkripsiApp\RF_CM.png")
        img6 = img6.resize((img6.width * 3, img6.height * 3),
Image.LANCZOS)

        col1, col2, col3 = st.columns([0.2, 0.8, 0.2])
        with col2:
            st.image(img6, caption="Confusion Matrix Hasil
Pengujian Support Vector Machine", width=450)

        st.write("   ")
        st.subheader("Evaluasi Model")
        st.write('Berdasarkan hasil confusion matrix tersebut,
dengan penggunaan Algoritma Random Forest didapatkan:')

        col1, col2, col3, col4 = st.columns([0.2, 1.1, 0.2,
0.2])

        with col2:
            st.latex(r"""
            Akurasi = \frac{TP + TN}{TP + TN + FP + FN} =
\frac{513 + 291}{513 + 291 + 48 + 25} = 0.9168 = 91.68\%
            """)

            st.latex(r"""
```

```
            Presisi = \frac{TP}{TP + FP} = \frac{513}{513 + 25} =
0.9535 = 95.35\%
            """)

            st.latex(r"""
            Recall = \frac{TP}{TP + FN} = \frac{513}{513 + 48} =
0.9144 = 91.44\%
            """)

            st.latex(r"""
            F1\text{-}Score = 2 \times \frac{Presisi \times
Recall}{Presisi + Recall} = 2 \times \frac{0.9144 \times
0.9535}{0.9144 + 0.9535} = 0.9336 = 93.36\%
            """)
        st.write("")
        img5 = Image.open("C:\SkripsiApp\RF_CR.png")
        img5 = img5.resize((img5.width * 3, img5.height * 3),
Image.LANCZOS)

        col1, col2, col3 = st.columns([0.2, 0.8, 0.2])
        with col2:
            st.image(img5, caption="Classification Report Random
Forest", width=450)

        st.write("___")
        st.subheader("Hasil Klasifikasi")
        st.write("Berdasarkan hasil klasifikasi, jumlah prediksi
yang sesuai sebanyak 826 data dan data yang tidak sesuai
sebanyak 77 data.")
        img4 = Image.open("C:\SkripsiApp\RF_Pred.png")
        img4 = img4.resize((img4.width * 3, img4.height * 3),
Image.LANCZOS)

        col1, col2, col3 = st.columns([0.2, 0.8, 0.2])
        with col2:
            st.image(img4, caption="Hasil Prediksi Random
Forest", width=450)


    elif selected == 'SVM':
        st.header("⚙️ Pengujian Support Vector Machine")
        st.markdown("""
        <hr style="border: 3px solid white; box-shadow: 0 4px 6px
-4px gray;">
        """, unsafe_allow_html=True)
```

```
        st.subheader("Confusion Matrix Support Vector Machine")
        st.write('Pada implementasi Support Vector Machine data
training yang di gunakan sebanyak 3505 data dan untuk testing
digunakan sebanyak 877 data. hasil pengujian yang
diimplementasikan pada metode Support Vector Machine ditujukan
dalam bentuk tabel Confusion Matrix. Berikut adalah hasil
pengujian yang didapatkan:   ')
        img3 = Image.open("C:\SkripsiApp\SVM_CM.png")
        img3 = img3.resize((img3.width * 3, img3.height * 3),
Image.LANCZOS)

        col1, col2, col3 = st.columns([0.2, 0.8, 0.2])
        with col2:
            st.image(img3, caption="Confusion Matrix Hasil
Pengujian Support Vector Machine", width=450)



        st.write("___")
        st.subheader("Evaluasi Model")
        st.write('Berdasarkan hasil confusion matrix tersebut,
dengan penggunaan Algoritma Support Vector Machine didapatkan:')
        col1, col2, col3, col4 = st.columns([0.2, 1.1, 0.2,
0.2])

        with col2:
            st.latex(r"""
             \text{Akurasi}  = \frac{TP + TN}{TP + TN + FP + FN} =
\frac{523 + 289}{523 + 289 + 38 + 27} = 0.9262 =
\mathbf{92.62\%}
            """)

            st.latex(r"""
            \text{Presisi}   = \frac{TP}{TP + FP} =
\frac{523}{523 + 27} = 0.9509 = \mathbf{95.09\%}
            """)

            st.latex(r"""
            \text{Recall}  = \frac{TP}{TP + FN} = \frac{523}{523
+ 38} = 0.9323 = \mathbf{93.23\%}
            """)

            st.latex(r"""
```

```
            F1\text{-}Score = 2 \times \frac{\text{Presisi}
\times \text{Recall}}{\text{Presisi} + \text{Recall}} = 2 \times
\frac{0.9323 \times 0.9509}{0.9323 + 0.9509} = 0.9396 =
\mathbf{93.96\%}
            """)
        st.write("")
        img2 = Image.open("C:\SkripsiApp\SVM_CR.png")
        img2 = img2.resize((img2.width * 3, img2.height * 3),
Image.LANCZOS)

        col1, col2, col3 = st.columns([0.2, 0.8, 0.2])
        with col2:
            st.image(img2, caption="Classification Report Support
Vector Machine", width=450)

        st.write("___")
        st.subheader("Hasil Klasifikasi")
        st.write("Berdasarkan hasil klasifikasi, jumlah prediksi
yang sesuai sebanyak 832 data dan data yang tidak sesuai
sebanyak 71 data.")
        img1 = Image.open("C:\SkripsiApp\SVM_Pred.png")
        img1 = img1.resize((img1.width * 3, img1.height * 3),
Image.LANCZOS)

        col1, col2, col3 = st.columns([0.2, 0.8, 0.2])
        with col2:
            st.image(img1, caption="Hasil Prediksi Support Vector
Machine", width=450)


    elif selected == 'Perbandingan':
        st.header("⚖️ Hasil Perbandingan Agoritma")
        st.markdown("""
        <hr style="border: 3px solid white; box-shadow: 0 4px 6px
-4px gray;">
        """, unsafe_allow_html=True)
        st.write("Dalam pengujian klasifikasi data ulasan,
dilakukan perbandingan antara dua algoritma machine learning,
yaitu Random Forest dan Support Vector Machine (SVM), dengan
menggunakan metrik evaluasi berupa akurasi, presisi, recall, dan
F1-score. Hasil pengujian menunjukkan bahwa Random Forest
memiliki akurasi sebesar 91.68%, presisi 95.35%, recall 91.44%,
dan F1-score 93.36%. Nilai presisi yang tinggi menunjukkan bahwa
model ini memiliki tingkat keandalan yang baik dalam
mengidentifikasi kelas positif, namun recall yang sedikit lebih
```

```
rendah mengindikasikan bahwa masih ada beberapa data positif
yang tidak terdeteksi dengan baik.")
        st.write("Di sisi lain, algoritma SVM menunjukkan kinerja
yang lebih unggul dengan akurasi sebesar 92.62%, presisi 95.09%,
recall 93.23%, dan F1-score 93.96%. Dengan nilai recall yang
lebih tinggi dibandingkan Random Forest, SVM lebih mampu
mendeteksi ulasan positif secara lebih menyeluruh, sehingga
menghasilkan keseimbangan yang lebih baik antara presisi dan
recall. F1-score yang lebih tinggi juga mengindikasikan bahwa
SVM lebih optimal dalam menangani klasifikasi data ulasan.")
        st.write("Berdasarkan hasil tersebut, dapat disimpulkan
bahwa SVM memiliki performa yang lebih baik dibandingkan dengan
Random Forest, terutama dalam hal akurasi, recall, dan
keseimbangan keseluruhan antara presisi dan recall.")
        st.image("C:/SkripsiApp/perbandingan_klasifikasi.png",
caption="Perbandingan Hasil Pengujian Random Forest & SVM",
use_container_width=True)
```