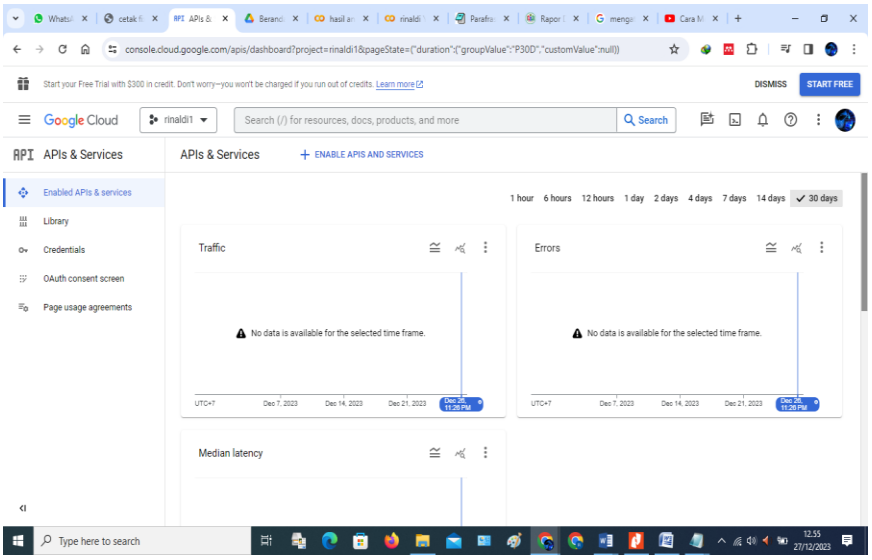
Proses Crawling Data Komentar Youtube

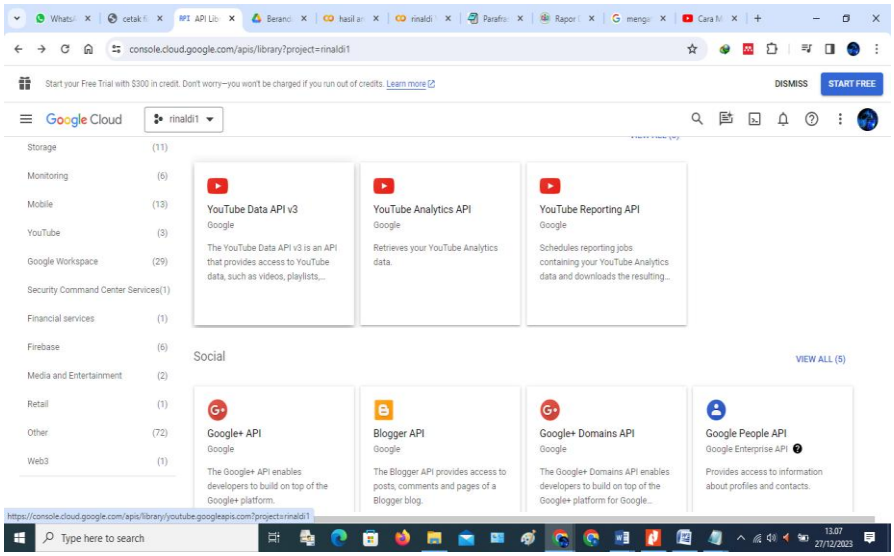
Data dikumpulkan melalui scraping web yang dilakukan pada laman video youtube dengan kata kunci “Kurikulum Merdeka” pada bulan Oktober 2023. Peneliti mengumpulkan data menggunakan bahasa pemrograman Python dan mengeksekusinya menggunakan Google Colab adapun Langkah-langkah yang dilakukan sebagai berikut:

a. Masuk ke akun Google yang kita miliki di https://accounts.google.com. Jika Anda tidak memiliki akun Google, daftar akun Google dan masuk ke akun Google tersebut.
b. Aktifkan layanan API Data YouTube dan dapatkan kunci API, yang nantinya akan digunakan untuk melakukan *Scraping* data. Aktifkan dengan mengunjungi https://console.cloud.google.com/apis/dashboard dengan akun Google yang sudah buat sebelumnya.



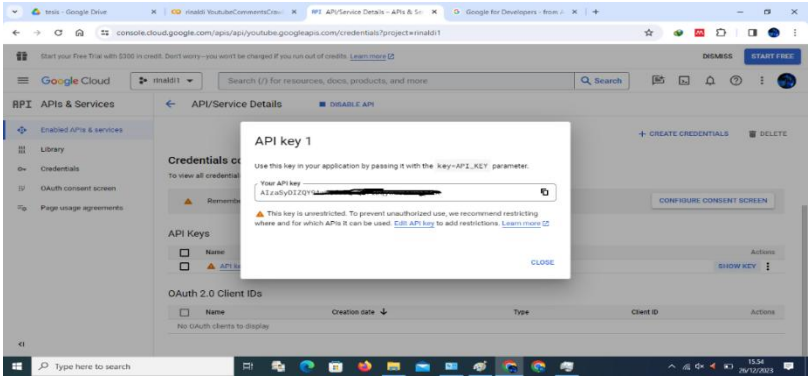
Gambar 4.1 Dasboard Google Developer

c. Sebelum mendapatkan YouTube API, peneliti harus mengaktifkan API di YouTube dengan mengklik tombol "+ *Enable Related* API and *Service*", yang kemudian akan mengarahkan ke perpustakaan Google API. Pilih YouTube Data API v3 dan aktifkan.



Gambar 4.2 *Library* API Youtube

d. Setelah YouTube Data API v3 diaktifkan, kembali ke halaman dasbor dan buka halaman Kredensial untuk mendapatkan kunci API. Selanjutnya tekan tombol "+ *CREATE CREDENTIALS*", lalu klik "API KEY" untuk menampilkan API key yang nantinya akan digunakan dalam proses pemrograman komentar YouTube.



Gambar 4.3 Pop-up API Key YouTube Data

Analisis Data Menggunakan Google Colabs

```
!pip install PySastrawi
!pip install gdown
!pip install gensim
```

```
Collecting PySastrawi
  Downloading PySastrawi-1.2.0-py2.py3-none-any.whl (210 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 210.6/210.6 kB 2.1 MB/s eta 0:00:00
Installing collected packages: PySastrawi
Successfully installed PySastrawi-1.2.0
Requirement already satisfied: gdown in /usr/local/lib/python3.10/dist-packages (4.7.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from gdown) (3.13.1)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.10/dist-packages (from gdown) (2.31.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from gdown) (1.16.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from gdown) (4.66.2)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from gdown) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->gdown) (2.5)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2024.2.2
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (1.7
Requirement already satisfied: gensim in /usr/local/lib/python3.10/dist-packages (4.3.2)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-packages (from gensim) (1.25.2)
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.10/dist-packages (from gensim) (1.11.4)
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.10/dist-packages (from gensim) (6.4.0)
```

```
#menginstal library
import re
import pandas as pd
import numpy as np
import datetime as dt
import string
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
from textblob import TextBlob
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, recall_score, precision_score
from sklearn.metrics import confusion_matrix, classification_report
import nltk
# import calendar
nltk.download('punkt')
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from wordcloud import WordCloud
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

```
#mengambil data dari google drive
!gdown --id 1AyDchiuPjRuQSjdstitkP4FnOciGYm8Z
```

```
/usr/local/lib/python3.10/dist-packages/gdown/cli.py:138: FutureWarning: Option `--id` was deprecated in version 4.3.1 and will be r
  warnings.warn(
Downloading...
From: https://drive.google.com/uc?id=1AyDchiuPjRuQSjdstitkP4FnOciGYm8Z
To: /content/kurmer_lebel_dataset.xlsx
100% 6.58M/6.58M [00:00<00:00, 115MB/s]
```

```
#membaca dataset
data_sentimen = pd.read_excel('kurmer_lebel_dataset.xlsx')
data_sentimen.head()
```

	koment	clean_review	normalization	final_text	token	stop_review
0	Sebagai guru.kurikulum merdeka ini dapat membe...	sebagai guru kurikulum merdeka ini dapat membe...	sebagai guru kurikulum merdeka ini dapat membe...	sebagai guru kurikulum merdeka dapat memberik...	['sebagai', 'guru', 'kurikulum', 'merdeka', 'd...	['guru', 'kurikulum', 'merdeka', 'motivasi', '...
1	Isinya buzzer semua kah ? Komenan kagak ada ya...	isinya buzzer semua kah komenan kagak ada yang...	isinya buzzer semua kah komenan tidak ada yang...	isinya buzzer semua komenan tidak yang kreat...	['isinya', 'buzzer', 'semua', 'komenan', 'tida...	['isinya', 'buzzer', 'komenan', 'kreatif', 'ti...
2	Asesmen awal di lakukan sebelum pembelajaran y...	asesmen awal di lakukan sebelum pembelajaran y...	asesmen awal di lakukan sebelum pembelajaran y...	asesmen awal lakukan sebelum pembelajaran yan...	['asesmen', 'awal', 'lakukan', 'sebelum', 'pem...	['asesmen', 'lakukan', 'pembelajaran', 'bertuj...
3	Terimakasih atas ilmunya kepada narasumber dan...	terimakasih atas ilmunya kepada narasumber dan...	terima kasih atas ilmunya kepada narasumber da...	terima kasih atas ilmunya kepada narasumber ...	['terima', 'kasih', 'atas', 'ilmunya', 'kepada...	['terima', 'kasih', 'ilmunya', 'narasumber', '...

```
polarity = []

for convert in data_sentimen['polarity']:
    if convert == "positif":
        num_polarity = 1
    elif convert == "negatif":
        num_polarity = -1
    else:  # Netral atau kategori lainnya
        num_polarity = 0

    polarity.append(num_polarity)

data_sentimen['polarity_numeric'] = polarity
```

```
data_sentimen['polarity']
```

```
0        positif
1        negatif
2        negatif
3        positif
4        positif
         ...
26480    negatif
26481     netral
26482    positif
26483    negatif
26484    negatif
Name: polarity, Length: 26485, dtype: object
```

```
data_sentimen.polarity.value_counts()
```

```
positif    19979
negatif     5108
netral      1398
Name: polarity, dtype: int64
```

```
# Hitung persentase sentimen
total_sampel = data_sentimen.shape[0]

# Hitung persentase positif
sampel_positif = data_sentimen['polarity_numeric'].value_counts().get(1, 0)
persentase_positif = (sampel_positif / total_sampel) * 100

# Hitung persentase negatif
sampel_negatif = data_sentimen['polarity_numeric'].value_counts().get(-1, 0)
persentase_negatif = (sampel_negatif / total_sampel) * 100

# Hitung jumlah sentimen netral
sampel_netral = data_sentimen['polarity_numeric'].value_counts().get(0, 0)
persentase_netral = (sampel_netral / total_sampel) * 100

# Tampilkan hasil
print("Persentase Sentimen:")
print(f"Persentase Sentimen Positif: {round(persentase_positif, 2)}%")
print(f"Persentase Sentimen Negatif: {round(persentase_negatif, 2)}%")
print(f"Persentase Sentimen Netral: {round(persentase_netral, 2)}%")
print(f"Perbedaan antara Sentimen Positif dan Sentimen Negatif: {round(persentase_positif - persentase_negatif, 2)}%")
print(f"Perbedaan antara Sentimen Positif dan Sentimen Netral: {round(persentase_positif - persentase_netral, 2)}%")
print(f"Perbedaan antara Sentimen Negatif dan Sentimen Netral: {round(persentase_negatif - persentase_netral, 2)}%")
```

```
Persentase Sentimen:
Persentase Sentimen Positif: 75.44%
Persentase Sentimen Negatif: 19.29%
Persentase Sentimen Netral: 5.28%
Perbedaan antara Sentimen Positif dan Sentimen Negatif: 56.15%
Perbedaan antara Sentimen Positif dan Sentimen Netral: 70.16%
Perbedaan antara Sentimen Negatif dan Sentimen Netral: 14.01%
```

```
data_sentimen = data_sentimen.replace('',np.nan, regex=True)
data_sentimen = data_sentimen.dropna()
data_sentimen = data_sentimen.reset_index(drop=True)
data_sentimen.to_excel("kurmer_fix_dataset.xlsx", index=False)
```

```
X = data_sentimen['stem_review']
y = data_sentimen['polarity']
```

```
data_sentimen.head()
```

	koment	clean_review	normalization	final_text	token	stop_review
0	Sebagai guru.kurikulum merdeka ini dapat membe...	sebagai guru kurikulum merdeka ini dapat membe...	sebagai guru kurikulum merdeka ini dapat membe...	sebagai guru kurikulum merdeka dapat memberik...	['sebagai', 'guru', 'kurikulum', 'merdeka', 'd...	['guru', 'kurikulum', 'merdeka', 'motivasi', '...
1	Isinya buzzer semua kah ? Komenan kagak ada ya...	isinya buzzer semua kah komenan kagak ada yang...	isinya buzzer semua kah komenan tidak ada yang...	isinya buzzer semua komenan tidak yang kreat...	['isinya', 'buzzer', 'semua', 'komenan', 'tida...	['isinya', 'buzzer', 'komenan', 'kreatif', 'ti...
2	Asesmen awal di lakukan sebelum pembelajaran y...	asesmen awal di lakukan sebelum pembelajaran y...	asesmen awal di lakukan sebelum pembelajaran y...	asesmen awal lakukan sebelum pembelajaran yan...	['asesmen', 'awal', 'lakukan', 'sebelum', 'pem...	['asesmen', 'lakukan', 'pembelajaran', 'bertuj...
3	Terimakasih atas ilmunya kepada narasumber dan...	terimakasih atas ilmunya kepada narasumber dan...	terima kasih atas ilmunya kepada narasumber da...	terima kasih atas ilmunya kepada narasumber ...	['terima', 'kasih', 'atas', 'ilmunya', 'kepada...	['terima', 'kasih', 'ilmunya', 'narasumber', '...
4	Kurikulum merdeka 1. Pendidikan berpusat pa...	kurikulum merdeka br pendidikan berpusat pada	kurikulum merdeka baru pendidikan berpusat pad...	kurikulum merdeka baru pendidikan berpusat	['kurikulum', 'merdeka', 'baru', 'pendidikan',...	['kurikulum', 'merdeka', 'pendidikan', 'berpus...

```
from sklearn.model_selection import train_test_split
#split data
# Misalnya, jika 'X' adalah fitur dan 'y' adalah label
X = data_sentimen['stem_review']
y = data_sentimen['polarity']

# Pisahkan data menjadi data latih (80%) dan data uji (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Tampilkan ukuran data latih dan data uji
print("Jumlah data latih:", len(X_train))
print("Jumlah data uji:", len(X_test))
```

```
Jumlah data latih: 21188
Jumlah data uji: 5297
```

```
import numpy as np
import pandas as pd
from sklearn.model_selection import StratifiedKFold, GridSearchCV, train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.metrics import confusion_matrix, accuracy_score, recall_score, precision_score

# Mengganti 'stem_review' dengan kolom yang sesuai pada dataset Anda
X = data_sentimen['stem_review']
y = data_sentimen['polarity']

# Membagi data menjadi data latih dan uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Pipeline dengan TfidfVectorizer dan Multinomial Naive Bayes
model = make_pipeline(TfidfVectorizer(), MultinomialNB())

# Parameter grid untuk dicari
param_grid = {
    'tfidfvectorizer__min_df': [1, 2, 3],
    'tfidfvectorizer__max_df': [0.9, 0.95, 1.0],
    'multinomialnb__alpha': [0.1, 0.5, 1.0]
}

# Menggunakan StratifiedKFold untuk validasi silang
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Melakukan pencarian parameter terbaik
grid_search = GridSearchCV(model, param_grid, cv=cv, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Evaluasi model dengan parameter terbaik pada data uji
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

# Mengukur metrik evaluasi
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred, average='weighted')
precision = precision_score(y_test, y_pred, average='weighted')

# Membuat DataFrame
data = {
    'test_size': [0.2],  # Sesuaikan dengan nilai yang sesuai
    'random_state': [42],  # Sesuaikan dengan nilai yang sesuai
    'confusion_matrix': [conf_matrix],
    'accuracy': [accuracy],
    'recall': [recall],
    'precision': [precision]
}

optimal = pd.DataFrame(data)
optimal
```

	test_size	random_state	confusion_matrix	accuracy	recall	precision
0	0.2	42	[[148, 8, 866], [7, 5, 267], [36, 5, 3955]]	0.775533	0.775533	0.750535

```
# mengatur desimal akurasi, recall, dan precision menjadi 2 angka dibelakang koma

optimal['accuracy'] = optimal['accuracy'].apply(lambda floats:format(float(floats), ".2f"))
optimal['recall'] = optimal['recall'].apply(lambda floats:format(float(floats), ".2f"))
optimal['precision'] = optimal['precision'].apply(lambda floats:format(float(floats), ".2f"))
optimal['test_size'] = optimal['test_size'].apply(lambda floats:format(float(floats), ".1f"))
optimal
```

	test_size	random_state	confusion_matrix	accuracy	recall	precision
0	0.2	42	[[148, 8, 866], [7, 5, 267], [36, 5, 3955]]	0.78	0.78	0.75

```
X_train.shape
```

```
(21188,)
```

```
X_test.shape
```

```
(5297,)
```

```
X_train
```

```
23867    riza lubis guru smpn blangpidie sekolah jalan ...
17302    alhamdulillah jalan kurikulum merdeka syukur c...
3443     guru konsep kurikulum merdeka konsep cocok ter...
18238    kurikulum merdeka guru harap merdeka tugas atm...
3291     tuju kurikulum merdeka ajar depan butuh siswa ...
                               ...
4305     konsep epmbeljaran kurikulum merdeka pusat mur...
6512     konsep kurikulum merdeka perhati prinsip orien...
20024    guru kreatif inovatif cipta ajar makna senang ...
10967    manfaat serta didik menumbuhkembangkan sikap m...
14278                                       materi bantu
Name: stem_review, Length: 21188, dtype: object
```

```
print(X_train)
```

```
23867    riza lubis guru smpn blangpidie sekolah jalan ...
17302    alhamdulillah jalan kurikulum merdeka syukur c...
3443     guru konsep kurikulum merdeka konsep cocok ter...
18238    kurikulum merdeka guru harap merdeka tugas atm...
3291     tuju kurikulum merdeka ajar depan butuh siswa ...
                               ...
4305     konsep epmbeljaran kurikulum merdeka pusat mur...
6512     konsep kurikulum merdeka perhati prinsip orien...
20024    guru kreatif inovatif cipta ajar makna senang ...
10967    manfaat serta didik menumbuhkembangkan sikap m...
14278                                       materi bantu
Name: stem_review, Length: 21188, dtype: object
```

```
import numpy as np
import pandas as pd
from gensim.models import Word2Vec
from nltk.tokenize import word_tokenize
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB


# Tokenisasi dan pelatihan model Word2Vec
tokenized_text = [word_tokenize(text.lower()) for text in X_train]
word2vec_model = Word2Vec(sentences=tokenized_text, vector_size=100, window=5, min_count=1, workers=4)

# Custom transformer untuk Word2Vec
class Word2VecVectorizer(BaseEstimator, TransformerMixin):
    def __init__(self, model):
        self.model = model

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        return np.array([self.get_vector(text) for text in X])

    def get_vector(self, text):
        tokens = word_tokenize(text.lower())
        vectors = [self.model.wv[token] for token in tokens if token in self.model.wv.key_to_index]
        if vectors:
            return np.mean(vectors, axis=0)
        else:
            return np.zeros(self.model.vector_size)

# Pipeline dengan Word2VecVectorizer dan Multinomial Naive Bayes
model = make_pipeline(
    Word2VecVectorizer(word2vec_model),
    MultinomialNB()
)

# Parameter grid untuk dicari
param_grid = {
    'multinomialnb__alpha': [0.1, 0.5, 1.0]
}
```

```
# Membuat model Word2Vec
tokenized_text = [word_tokenize(text.lower()) for text in data_sentimen['stem_review']]
word2vec_model = Word2Vec(sentences=tokenized_text, vector_size=100, window=5, min_count=1, workers=4)

# Melihat vektor representasi kata-kata
word_vectors = word2vec_model.wv

# Mendapatkan daftar kata yang ada dalam model
words_list = list(word_vectors.index_to_key)

# Membuat DataFrame vektor representasi kata-kata
word_vectors_df = pd.DataFrame(index=words_list, data=word_vectors.vectors)

# Menampilkan DataFrame
print(word_vectors_df)
```

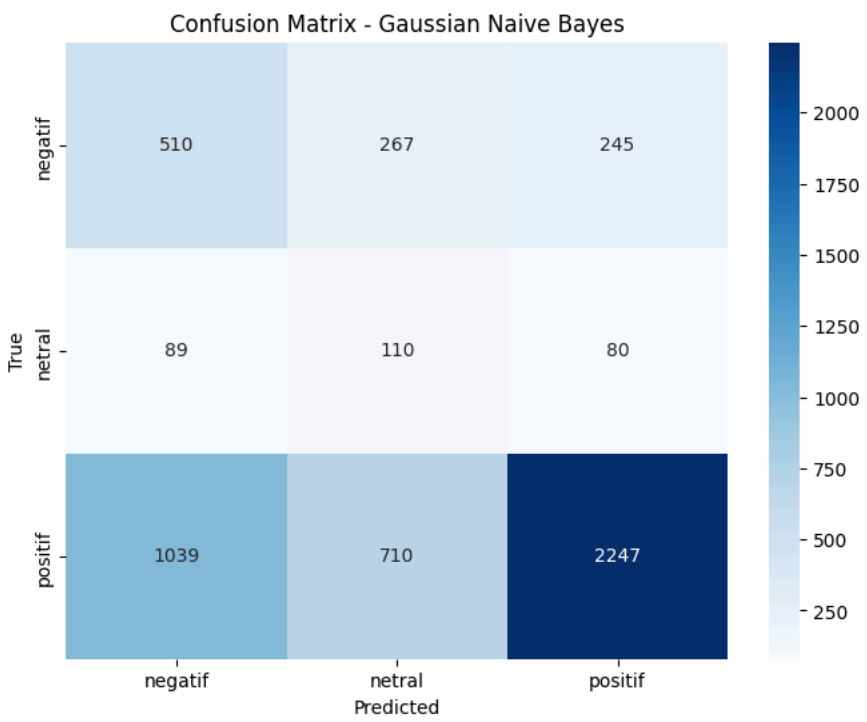
```
                     0         1         2         3         4         5   \
merdeka        -0.602381 -0.397389  0.840792 -0.573353 -0.173184 -0.277473
kurikulum      -1.285484  1.545251 -0.043064  0.115587  0.896082  1.098574
ajar           -0.288279  0.371303  0.459213 -0.508990  0.710769  0.488401
guru           -0.708380  0.646374 -0.132728 -0.912265  0.695120 -0.666043
didik          -0.603582 -0.067021 -0.145942 -0.222261  1.328904  0.970903
...                  ...       ...       ...       ...       ...       ...
berkuranhnya   -0.012254  0.014518  0.008450  0.005875  0.009652 -0.002119
sangatmembantu  0.009535 -0.002283 -0.005949 -0.013847  0.014449 -0.008140
rostina        -0.009571  0.013667 -0.004968 -0.005589  0.010039 -0.017263
ujungbatu      -0.000310  0.011913 -0.002819 -0.002208  0.033064 -0.027570
berbasia       -0.018258 -0.001594  0.002592 -0.007652 -0.008996 -0.015319

                       6         7         8         9   ...        90  \
merdeka         0.203261  0.123668  0.508184  0.856452  ...  0.719289
kurikulum      -0.117165  0.550560  0.906015  0.486924  ...  0.874083
ajar           -0.700056  0.135565  0.413046  0.631916  ... -0.465594
guru           -1.092234  0.336940  1.293927  1.565687  ... -0.729772
```

```
didik           0.718987  0.641613  0.297282  0.722270  ...  0.464853
...                  ...       ...       ...       ...  ...       ...
berkuranhnya    0.009798  0.014300 -0.028388  0.011859  ...  0.013970
sangatmembantu  0.010119  0.017858 -0.011812 -0.013669  ...  0.015287
rostina        -0.002461  0.030294 -0.009093 -0.006837  ...  0.011233
ujungbatu       0.014932  0.044421 -0.029396  0.003258  ...  0.018080
berbasia        0.005044  0.021701 -0.007548 -0.003320  ...  0.025538

                      91        92        93        94        95        96  \
merdeka        -1.069066  0.332191 -0.861775  0.940572  0.388253 -0.548947
kurikulum      -0.571599  0.072634 -1.159945  0.630463 -0.169726  0.888293
ajar           -1.179223 -0.332463 -0.239109  0.725214 -0.239048 -0.849011
guru            0.014097  0.480542 -0.267015  0.705767  0.176691  0.709974
didik          -1.806900  0.594170 -0.551120  0.471478 -0.589343 -0.067514
...                  ...       ...       ...       ...       ...       ...
berkuranhnya    0.001963 -0.008854  0.012882  0.011003  0.005648  0.008483
sangatmembantu  0.006759 -0.002626 -0.002262 -0.008948  0.026991  0.007509
rostina         0.005474  0.013630  0.022502  0.009353  0.027410  0.032360
ujungbatu       0.009167  0.007821  0.026878  0.018695  0.021899  0.018816
berbasia        0.004849  0.007208 -0.011451  0.008997  0.020433  0.016103

                      97        98        99
merdeka        -0.257511  0.799186  0.756706
kurikulum      -0.121598 -0.079471 -0.192417
ajar            0.098566 -0.219853 -0.009073
guru           -0.420025  0.357309  0.020831
didik          -0.949276 -0.029665 -0.071573
...                  ...       ...       ...
berkuranhnya   -0.020344  0.023676 -0.006364
sangatmembantu  0.000214  0.012384 -0.010081
rostina        -0.015926  0.030869 -0.007795
ujungbatu      -0.022908  0.038451 -0.027932
berbasia       -0.018691  0.020973  0.005447

[10301 rows x 100 columns]
```

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from gensim.models import Word2Vec
from nltk.tokenize import word_tokenize
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder


X = data_sentimen['stem_review']
y = data_sentimen['polarity']

# Membagi data menjadi data latih dan uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Tokenisasi dan pelatihan model Word2Vec
tokenized_text = [word_tokenize(text.lower()) for text in X_train]
word2vec_model = Word2Vec(sentences=tokenized_text, vector_size=100, window=5, min_count=1, workers=4)

# Mengubah data teks menjadi vektor Word2Vec
X_train_word2vec = []
X_test_word2vec = []

# Mengumpulkan vektor Word2Vec untuk setiap dokumen pada data latih
for text in X_train:
    vectors = [word2vec_model.wv[token] for token in word_tokenize(text.lower()) if token in word2vec_model.wv.key_to_index]
    if vectors:
        X_train_word2vec.append(np.mean(vectors, axis=0))
    else:
        # Jika tidak ada token yang ditemukan, tambahkan vektor nol
        X_train_word2vec.append(np.zeros(word2vec_model.vector_size))

# Mengumpulkan vektor Word2Vec untuk setiap dokumen pada data uji
for text in X_test:
    vectors = [word2vec_model.wv[token] for token in word_tokenize(text.lower()) if token in word2vec_model.wv.key_to_index]
    if vectors:
        X_test_word2vec.append(np.mean(vectors, axis=0))
    else:
        # Jika tidak ada token yang ditemukan, tambahkan vektor nol
        X_test_word2vec.append(np.zeros(word2vec_model.vector_size))

# Mengubah list vektor Word2Vec menjadi array numpy
X_train_word2vec = np.array(X_train_word2vec)
X_test_word2vec = np.array(X_test_word2vec)

# Label encoding untuk kelas sentimen
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Algoritma Gaussian Naive Bayes
gnb_model = GaussianNB()
gnb_model.fit(X_train_word2vec, y_train)
gnb_predictions = gnb_model.predict(X_test_word2vec)
gnb_accuracy = accuracy_score(y_test, gnb_predictions)

# Algoritma K-Nearest Neighbors (KNN)
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train_word2vec, y_train_encoded)
knn_predictions = knn_model.predict(X_test_word2vec)
knn_accuracy = accuracy_score(y_test_encoded, knn_predictions)

# Membandingkan akurasi kedua model
print("Gaussian Naive Bayes Accuracy (Word2Vec):", gnb_accuracy)
print("KNN Accuracy (Word2Vec):", knn_accuracy)
```

```
Gaussian Naive Bayes Accuracy (Word2Vec): 0.5412497640173684
KNN Accuracy (Word2Vec): 0.7961110062299415
```

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Confusion matrix untuk Gaussian Naive Bayes
cm_gnb = confusion_matrix(y_test, gnb_predictions)

# Visualisasi confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm_gnb, annot=True, fmt="d", cmap="Blues", xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.title("Confusion Matrix - Gaussian Naive Bayes")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```




```
import pandas as pd
from sklearn.metrics import confusion_matrix

# Confusion matrix untuk Gaussian Naive Bayes
cm_gnb = confusion_matrix(y_test, gnb_predictions)

# Membuat DataFrame dari confusion matrix
cm_df = pd.DataFrame(cm_gnb, index=label_encoder.classes_, columns=label_encoder.classes_)

# Tampilkan confusion matrix dalam bentuk DataFrame
print("Confusion Matrix - Gaussian Naive Bayes:")
print(cm_df)
```

```
Confusion Matrix - Gaussian Naive Bayes:
          negatif  netral  positif
negatif       510     267       245
netral         89     110        80
positif      1039     710      2247
```

```
import pandas as pd
from sklearn.metrics import confusion_matrix

# Confusion matrix untuk Gaussian Naive Bayes
cm_gnb = confusion_matrix(y_test, gnb_predictions)

# Membuat DataFrame dari confusion matrix dengan nama label untuk masing-masing kelas
label_names = label_encoder.classes_
cm_df = pd.DataFrame(cm_gnb, index=label_names, columns=label_names)

# Menghitung metrik True Positif (TP), True Negatif (TN), False Positif (FP), dan False Negatif (FN) untuk setiap kelas
TP = cm_df.values.diagonal()
TN = cm_df.values.sum() - cm_df.values.sum(axis=1) - cm_df.values.sum(axis=0) + TP
FP = cm_df.values.sum(axis=1) - TP
FN = cm_df.values.sum(axis=0) - TP

# Menambahkan kolom baru ke DataFrame untuk metrik TP, TN, FP, FN
cm_df['True Positif'] = TP
cm_df['True Negatif'] = TN
cm_df['False Positif'] = FP
cm_df['False Negatif'] = FN

# Menampilkan DataFrame yang lebih lengkap
print("Confusion Matrix - Gaussian Naive Bayes:")
print(cm_df)
```

```
Confusion Matrix - Gaussian Naive Bayes:
          negatif  netral  positif  True Positif  True Negatif  False Positif  \
negatif       510     267      245           510          3147            512
netral         89     110       80           110          4041            169
positif      1039     710     2247          2247           976           1749

          False Negatif
negatif            1128
netral              977
positif             325
```

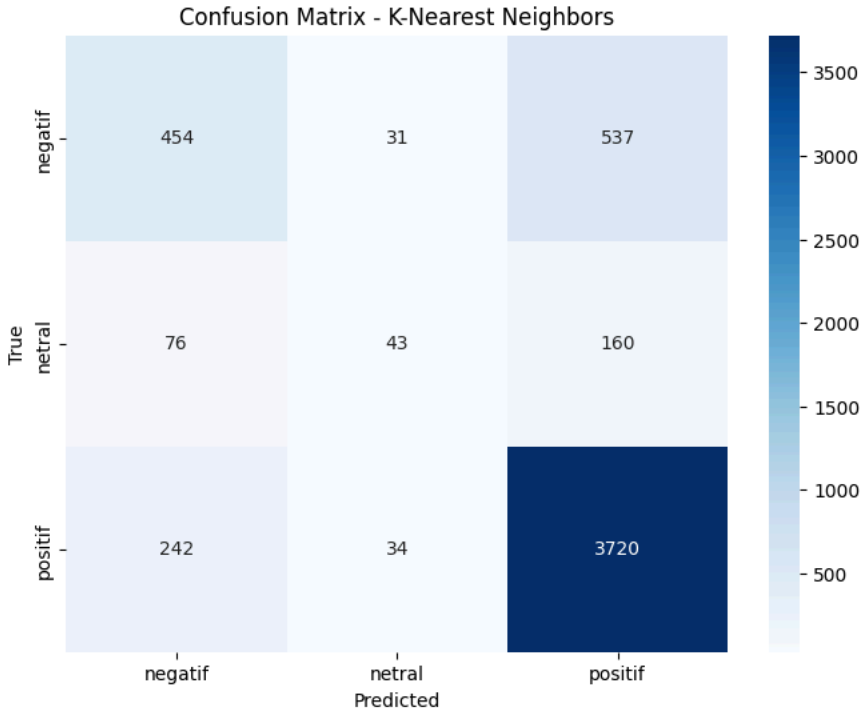
```
import pandas as pd
from sklearn.metrics import confusion_matrix

# Membuat confusion matrix untuk Gaussian Naive Bayes
cm_gnb = confusion_matrix(y_test, gnb_predictions)

# Membuat DataFrame dari confusion matrix
label_names = label_encoder.classes_
cm_df = pd.DataFrame(cm_gnb, index=label_names, columns=label_names)

# Menghitung metrik True Positif (TP), True Negatif (TN), False Positif (FP), dan False Negatif (FN) untuk setiap kelas
TP = cm_df.values.diagonal()
TN = cm_df.values.sum() - cm_df.values.sum(axis=1) - cm_df.values.sum(axis=0) + TP
FP = cm_df.values.sum(axis=1) - TP
FN = cm_df.values.sum(axis=0) - TP

# Membuat DataFrame baru untuk menampilkan metrik
metrics_df = pd.DataFrame({
    'Kelas': label_names,
    'True Positif': TP,
    'True Negatif': TN,
    'False Positif': FP,
    'False Negatif': FN
})

# Menampilkan DataFrame untuk metrik
print("Metrik untuk setiap kelas:")
print(metrics_df)
```

```
Metrik untuk setiap kelas:
     Kelas  True Positif  True Negatif  False Positif  False Negatif
0  negatif           510          3147            512           1128
1   netral           110          4041            169            977
2  positif          2247           976           1749            325
```

```
import pandas as pd
from sklearn.metrics import confusion_matrix

# Membuat confusion matrix untuk Gaussian Naive Bayes
cm_gnb = confusion_matrix(y_test, gnb_predictions)

# Membuat DataFrame dari confusion matrix
label_names = label_encoder.classes_
cm_df = pd.DataFrame(cm_gnb, index=label_names, columns=label_names)

# Menghitung metrik True Positif (TP), True Negatif (TN), False Positif (FP), dan False Negatif (FN) untuk setiap kelas
TP = cm_df.values.diagonal()
FP = cm_df.values.sum(axis=0) - TP
FN = cm_df.values.sum(axis=1) - TP

# Menghitung True Netral (TN) dan False Netral (FN) berdasarkan label kelas
TN = []
for label in label_names:
    if label == 'netral':
        TN.append(cm_df.values.sum() - TP.sum() - FP.sum() - FN.sum())
    else:
        TN.append(cm_df.values.sum() - TP[label_names != 'netral'].sum() - FP[label_names != 'netral'].sum() - FN[label_names != 'netral

# Membuat DataFrame baru untuk menampilkan metrik
metrics_df = pd.DataFrame({
    'Kelas': label_names,
    'True Positif': TP,
    'True Netral': TN,
    'True Negatif': cm_df.values.sum() - TP - FP - FN,
    'False Positif': FP,
    'False Netral': FN,
    'False Negatif': FN
})

# Menampilkan DataFrame untuk metrik
print("Metrik untuk setiap kelas:")
print(metrics_df)
```

```
Metrik untuk setiap kelas:
      Kelas  True Positif  True Netral  True Negatif  False Positif  \
0  negatif           510        -1174          3147           1128
1   netral           110        -2430          4041            977
2  positif          2247        -1174           976            325

   False Netral  False Negatif
0           512            512
1           169            169
2          1749           1749
```

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Confusion matrix untuk K-Nearest Neighbors
cm_knn = confusion_matrix(y_test_encoded, knn_predictions)

# Visualisasi confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm_knn, annot=True, fmt="d", cmap="Blues", xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.title("Confusion Matrix - K-Nearest Neighbors")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```

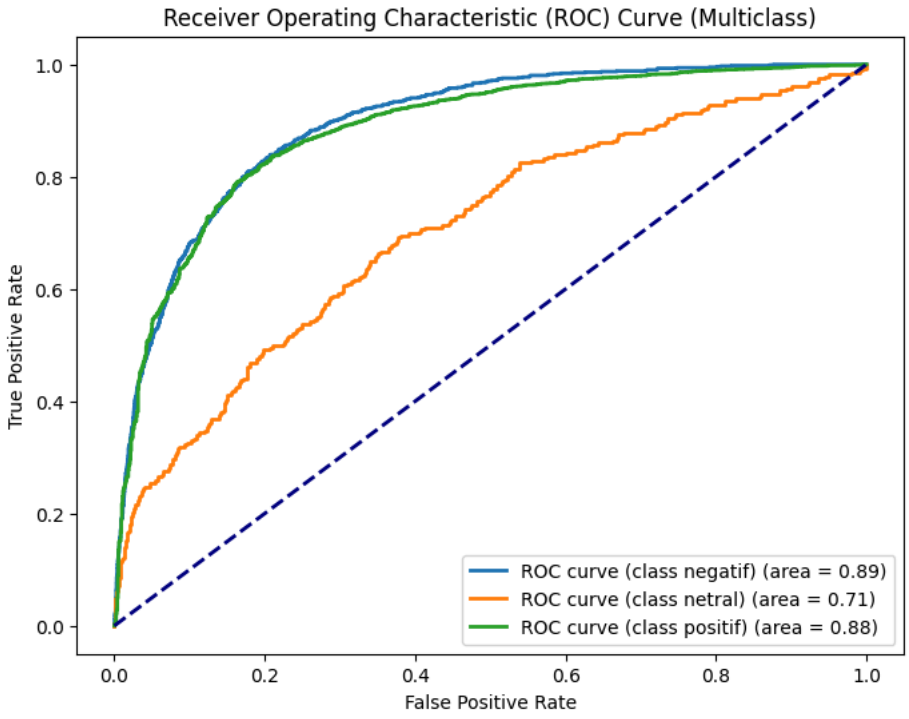



```
import pandas as pd
from sklearn.metrics import confusion_matrix

# Confusion matrix untuk K-Nearest Neighbors
cm_knn = confusion_matrix(y_test_encoded, knn_predictions)

# Membuat DataFrame dari confusion matrix
label_names = label_encoder.classes_
cm_df = pd.DataFrame(cm_knn, index=label_names, columns=label_names)

# Menampilkan DataFrame
print("Confusion Matrix - K-Nearest Neighbors:")
print(cm_df)
```

```
Confusion Matrix - K-Nearest Neighbors:
          negatif  netral  positif
negatif       454      31      537
netral         76      43      160
positif       242      34     3720
```

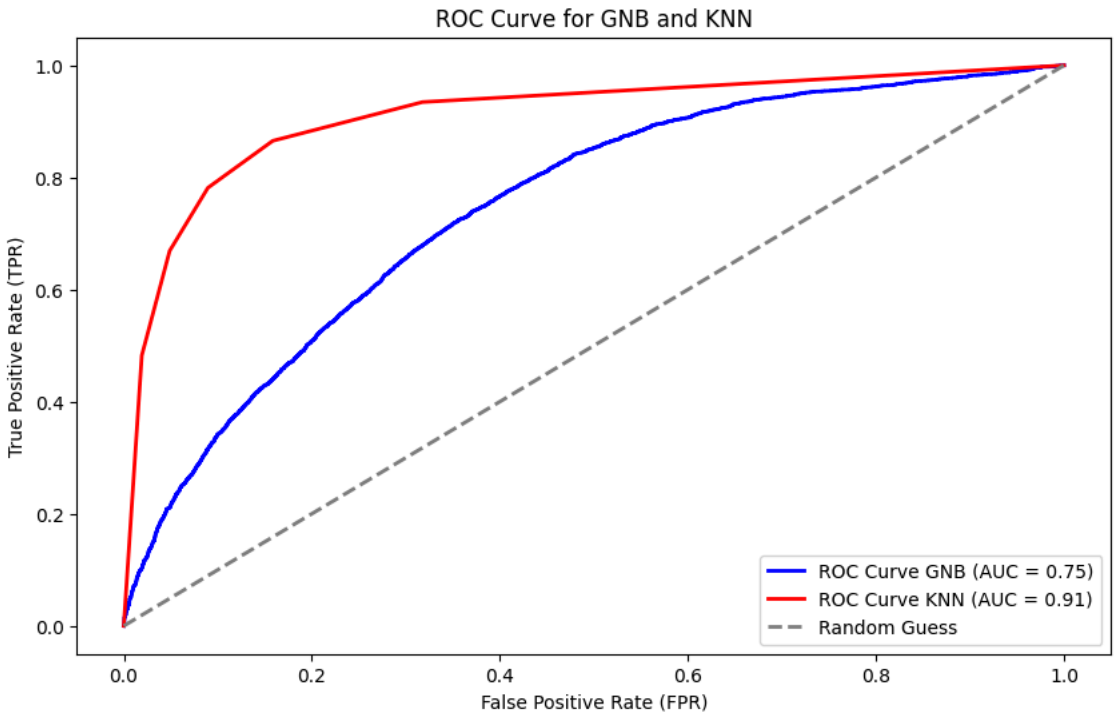
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from gensim.models import Word2Vec
from nltk.tokenize import word_tokenize
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, roc_curve, auc
from sklearn.preprocessing import LabelEncoder
from sklearn.multiclass import OneVsRestClassifier
import matplotlib.pyplot as plt


X = data_sentimen['stem_review']
y = data_sentimen['polarity']

# Membagi data menjadi data latih dan uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Tokenisasi dan pelatihan model Word2Vec
tokenized_text = [word_tokenize(text.lower()) for text in X_train]
word2vec_model = Word2Vec(sentences=tokenized_text, vector_size=100, window=5, min_count=1, workers=4)

# Mengubah data teks menjadi vektor Word2Vec
X_train_word2vec = []
X_test_word2vec = []

# Mengumpulkan vektor Word2Vec untuk setiap dokumen pada data latih
for text in X_train:
    vectors = [word2vec_model.wv[token] for token in word_tokenize(text.lower()) if token in word2vec_model.wv.key_to_index]
    if vectors:
        X_train_word2vec.append(np.mean(vectors, axis=0))
    else:
        # Jika tidak ada token yang ditemukan, tambahkan vektor nol
        X_train_word2vec.append(np.zeros(word2vec_model.vector_size))

# Mengumpulkan vektor Word2Vec untuk setiap dokumen pada data uji
for text in X_test:
    vectors = [word2vec_model.wv[token] for token in word_tokenize(text.lower()) if token in word2vec_model.wv.key_to_index]
    if vectors:
        X_test_word2vec.append(np.mean(vectors, axis=0))
    else:
        # Jika tidak ada token yang ditemukan, tambahkan vektor nol
        X_test_word2vec.append(np.zeros(word2vec_model.vector_size))

# Mengubah list vektor Word2Vec menjadi array numpy
X_train_word2vec = np.array(X_train_word2vec)
X_test_word2vec = np.array(X_test_word2vec)

# Label encoding untuk kelas sentimen
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Algoritma Logistic Regression dengan strategi One-vs-Rest (OvR)
logreg_model = OneVsRestClassifier(LogisticRegression(max_iter=1000))
logreg_model.fit(X_train_word2vec, y_train_encoded)

# Menghitung probabilitas prediksi pada data uji
y_prob_logreg = logreg_model.decision_function(X_test_word2vec)

# Menghitung false positive rate (FPR), true positive rate (TPR), dan area under the curve (AUC) untuk setiap kelas
fpr = dict()
tpr = dict()
roc_auc = dict()

for i in range(len(label_encoder.classes_)):
    fpr[i], tpr[i], _ = roc_curve((y_test_encoded == i).astype(int), y_prob_logreg[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Plotting kurva ROC untuk setiap kelas
plt.figure(figsize=(8, 6))
for i in range(len(label_encoder.classes_)):
    plt.plot(fpr[i], tpr[i], lw=2, label='ROC curve (class {}) (area = {:.2f})'.format(label_encoder.classes_[i], roc_auc[i]))

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve (Multiclass)')
plt.legend(loc='lower right')
plt.show()
```

```
# Menampilkan akurasi model
logreg_accuracy = logreg_model.score(X_test_word2vec, y_test_encoded)
print("Logistic Regression Accuracy (Word2Vec):", logreg_accuracy)
```




```
Logistic Regression Accuracy (Word2Vec): 0.8234849915046253
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from itertools import cycle

# ...

# Label encoding untuk kelas sentimen
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Algoritma Gaussian Naive Bayes
gnb_model = GaussianNB()
gnb_model.fit(X_train_word2vec, y_train)
gnb_probs = gnb_model.predict_proba(X_test_word2vec)

# Algoritma K-Nearest Neighbors (KNN)
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train_word2vec, y_train_encoded)
knn_probs = knn_model.predict_proba(X_test_word2vec)

# Mengubah kelas menjadi format biner untuk ROC
y_test_bin = label_binarize(y_test_encoded, classes=list(range(len(label_encoder.classes_))))

# Menghitung nilai TPR dan FPR untuk Gaussian Naive Bayes
fpr_gnb, tpr_gnb, _ = roc_curve(y_test_bin.ravel(), gnb_probs.ravel())
roc_auc_gnb = auc(fpr_gnb, tpr_gnb)

# Menghitung nilai TPR dan FPR untuk K-Nearest Neighbors
fpr_knn, tpr_knn, _ = roc_curve(y_test_bin.ravel(), knn_probs.ravel())
roc_auc_knn = auc(fpr_knn, tpr_knn)

# Menggambar kurva ROC
plt.figure(figsize=(10, 6))
plt.plot(fpr_gnb, tpr_gnb, color='blue', lw=2, label=f'ROC Curve GNB (AUC = {roc_auc_gnb:.2f})')
plt.plot(fpr_knn, tpr_knn, color='red', lw=2, label=f'ROC Curve KNN (AUC = {roc_auc_knn:.2f})')

plt.plot([0, 1], [0, 1], color='grey', lw=2, linestyle='--', label='Random Guess')

plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('ROC Curve for GNB and KNN')
plt.legend(loc='lower right')
plt.show()
```



```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from gensim.models import Word2Vec
from nltk.tokenize import word_tokenize
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder



X = data_sentimen['stem_review']
y = data_sentimen['polarity']

# Membagi data menjadi data latih dan uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Tokenisasi dan pelatihan model Word2Vec
tokenized_text = [word_tokenize(text.lower()) for text in X_train]
word2vec_model = Word2Vec(sentences=tokenized_text, vector_size=100, window=5, min_count=1, workers=4)

# Mengubah data teks menjadi vektor Word2Vec
X_train_word2vec = []
X_test_word2vec = []

# Mengumpulkan vektor Word2Vec untuk setiap dokumen pada data latih
for text in X_train:
    vectors = [word2vec_model.wv[token] for token in word_tokenize(text.lower()) if token in word2vec_model.wv.key_to_index]
    if vectors:
        X_train_word2vec.append(np.mean(vectors, axis=0))
    else:
        # Jika tidak ada token yang ditemukan, tambahkan vektor nol
        X_train_word2vec.append(np.zeros(word2vec_model.vector_size))

# Mengumpulkan vektor Word2Vec untuk setiap dokumen pada data uji
for text in X_test:
    vectors = [word2vec_model.wv[token] for token in word_tokenize(text.lower()) if token in word2vec_model.wv.key_to_index]
    if vectors:
        X_test_word2vec.append(np.mean(vectors, axis=0))
    else:
        # Jika tidak ada token yang ditemukan, tambahkan vektor nol
        X_test_word2vec.append(np.zeros(word2vec_model.vector_size))

# Mengubah list vektor Word2Vec menjadi array numpy
X_train_word2vec = np.array(X_train_word2vec)
X_test_word2vec = np.array(X_test_word2vec)

# Label encoding untuk kelas sentimen
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Algoritma Gaussian Naive Bayes
gnb_model = GaussianNB()
gnb_model.fit(X_train_word2vec, y_train_encoded)
gnb_predictions = gnb_model.predict(X_test_word2vec)
gnb_accuracy = accuracy_score(y_test_encoded, gnb_predictions)

# Algoritma K-Nearest Neighbors (KNN)
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train_word2vec, y_train_encoded)
knn_predictions = knn_model.predict(X_test_word2vec)
knn_accuracy = accuracy_score(y_test_encoded, knn_predictions)

# Menampilkan akurasi model
print("Gaussian Naive Bayes Accuracy (Word2Vec):", gnb_accuracy)
print("KNN Accuracy (Word2Vec):", knn_accuracy)
```

```
Gaussian Naive Bayes Accuracy (Word2Vec): 0.5372852558051727
KNN Accuracy (Word2Vec): 0.7934680007551445
```

Start coding or generate with AI.

```
#menghitung akurasi KNN dan Naive bayes jika vektorisasi dengan TF-IDF
# Memilih fitur dan label
X = data_sentimen['stem_review']
y = data_sentimen['polarity']

# Membagi data menjadi data latih dan uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Membuat TF-IDF vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000)  # Sesuaikan dengan kebutuhan Anda

# Melakukan vektorisasi pada data latih dan uji
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Label encoding untuk kelas sentimen
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Gaussian Naive Bayes dengan TF-IDF
gnb_tfidf_model = GaussianNB()
gnb_tfidf_model.fit(X_train_tfidf.toarray(), y_train_encoded)
gnb_tfidf_predictions = gnb_tfidf_model.predict(X_test_tfidf.toarray())
gnb_tfidf_accuracy = accuracy_score(y_test_encoded, gnb_tfidf_predictions)

# K-Nearest Neighbors (KNN) dengan TF-IDF
knn_tfidf_model = KNeighborsClassifier(n_neighbors=5)
knn_tfidf_model.fit(X_train_tfidf.toarray(), y_train_encoded)
knn_tfidf_predictions = knn_tfidf_model.predict(X_test_tfidf.toarray())
knn_tfidf_accuracy = accuracy_score(y_test_encoded, knn_tfidf_predictions)

# Menampilkan hasil akurasi
print("Gaussian Naive Bayes Accuracy (TF-IDF):", gnb_tfidf_accuracy)
print("KNN Accuracy (TF-IDF):", knn_tfidf_accuracy)
```

```
Gaussian Naive Bayes Accuracy (TF-IDF): 0.1848215971304512
KNN Accuracy (TF-IDF): 0.6801963375495563
```

```
#akurasi dengan TF IDF untuk semua jenis naive bayes
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB, BernoulliNB, GaussianNB, ComplementNB
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder


# Memilih fitur dan label
X = data_sentimen['stem_review']
y = data_sentimen['polarity']

# Membagi data menjadi data latih dan uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Membuat TF-IDF vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000)  # Sesuaikan dengan kebutuhan Anda

# Melakukan vektorisasi pada data latih dan uji
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Label encoding untuk kelas sentimen
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Inisialisasi model Naive Bayes
mnb_model = MultinomialNB()
bnb_model = BernoulliNB()
gnb_model = GaussianNB()
cnb_model = ComplementNB()

# Melatih model
mnb_model.fit(X_train_tfidf, y_train_encoded)
bnb_model.fit(X_train_tfidf, y_train_encoded)
gnb_model.fit(X_train_tfidf.toarray(), y_train_encoded)  # Gaussian Naive Bayes memerlukan array
cnb_model.fit(X_train_tfidf, y_train_encoded)

# Membuat prediksi
```

```
mnb_predictions = mnb_model.predict(X_test_tfidf)
bnb_predictions = bnb_model.predict(X_test_tfidf)
gnb_predictions = gnb_model.predict(X_test_tfidf.toarray())
cnb_predictions = cnb_model.predict(X_test_tfidf)

# Menampilkan hasil akurasi dan laporan klasifikasi
print("Multinomial Naive Bayes Accuracy:", accuracy_score(y_test_encoded, mnb_predictions))
print("Bernoulli Naive Bayes Accuracy:", accuracy_score(y_test_encoded, bnb_predictions))
print("Gaussian Naive Bayes Accuracy:", accuracy_score(y_test_encoded, gnb_predictions))
print("Complement Naive Bayes Accuracy:", accuracy_score(y_test_encoded, cnb_predictions))

# Laporan Klasifikasi
print("\nMultinomial Naive Bayes Classification Report:")
print(classification_report(y_test_encoded, mnb_predictions))

print("\nBernoulli Naive Bayes Classification Report:")
print(classification_report(y_test_encoded, bnb_predictions))

print("\nGaussian Naive Bayes Classification Report:")
print(classification_report(y_test_encoded, gnb_predictions))

print("\nComplement Naive Bayes Classification Report:")
print(classification_report(y_test_encoded, cnb_predictions))
```

```
Multinomial Naive Bayes Accuracy: 0.763262223900321
Bernoulli Naive Bayes Accuracy: 0.81026996413064
Gaussian Naive Bayes Accuracy: 0.1848215971304512
Complement Naive Bayes Accuracy: 0.7955446479139135

Multinomial Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       0.66      0.07      0.13      1022
           1       0.33      0.01      0.02       279
           2       0.77      0.99      0.86      3996

    accuracy                           0.76      5297
   macro avg       0.59      0.36      0.34      5297
weighted avg       0.72      0.76      0.68      5297


Bernoulli Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.56      0.58      1022
           1       0.17      0.05      0.07       279
           2       0.86      0.93      0.89      3996

    accuracy                           0.81      5297
   macro avg       0.55      0.51      0.52      5297
weighted avg       0.78      0.81      0.79      5297


Gaussian Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       0.23      0.23      0.23      1022
           1       0.06      0.73      0.11       279
           2       0.81      0.14      0.23      3996

    accuracy                           0.18      5297
   macro avg       0.37      0.36      0.19      5297
weighted avg       0.66      0.18      0.22      5297


Complement Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       0.61      0.54      0.57      1022
           1       0.19      0.10      0.13       279
           2       0.86      0.91      0.88      3996

    accuracy                           0.80      5297
   macro avg       0.55      0.52      0.53      5297
weighted avg       0.77      0.80      0.78      5297
```

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from gensim.models import Word2Vec
from sklearn.naive_bayes import MultinomialNB, BernoulliNB, GaussianNB, ComplementNB
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder

# Memilih fitur dan label
X = data_sentimen['stem_review']
y = data_sentimen['polarity']

# Membagi data menjadi data latih dan uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Training model Word2Vec
word2vec_model = Word2Vec(sentences=X_train, vector_size=100, window=5, min_count=1, workers=4)

# Fungsi untuk mengubah teks menjadi vektor Word2Vec
def word2vec_transformer(X, model):
    return np.array([np.mean([model.wv[word] for word in words.split() if word in model.wv] or [np.zeros(model.vector_size)], axis=0) fo

# Melakukan vektorisasi pada data latih dan uji
X_train_word2vec = word2vec_transformer(X_train, word2vec_model)
X_test_word2vec = word2vec_transformer(X_test, word2vec_model)

# Label encoding untuk kelas sentimen
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Inisialisasi model Naive Bayes
mnb_model = MultinomialNB()
bnb_model = BernoulliNB()
gnb_model = GaussianNB()
cnb_model = ComplementNB()

# Melatih model
mnb_model.fit(X_train_word2vec, y_train_encoded)
bnb_model.fit(X_train_word2vec, y_train_encoded)
gnb_model.fit(X_train_word2vec, y_train_encoded)
cnb_model.fit(X_train_word2vec, y_train_encoded)

# Membuat prediksi
mnb_predictions = mnb_model.predict(X_test_word2vec)
bnb_predictions = bnb_model.predict(X_test_word2vec)
gnb_predictions = gnb_model.predict(X_test_word2vec)
cnb_predictions = cnb_model.predict(X_test_word2vec)

# Menampilkan hasil akurasi dan laporan klasifikasi
print("Multinomial Naive Bayes Accuracy:", accuracy_score(y_test_encoded, mnb_predictions))
print("Bernoulli Naive Bayes Accuracy:", accuracy_score(y_test_encoded, bnb_predictions))
print("Gaussian Naive Bayes Accuracy:", accuracy_score(y_test_encoded, gnb_predictions))
print("Complement Naive Bayes Accuracy:", accuracy_score(y_test_encoded, cnb_predictions))

# Laporan Klasifikasi
print("\nMultinomial Naive Bayes Classification Report:")
print(classification_report(y_test_encoded, mnb_predictions))

print("\nBernoulli Naive Bayes Classification Report:")
print(classification_report(y_test_encoded, bnb_predictions))

print("\nGaussian Naive Bayes Classification Report:")
print(classification_report(y_test_encoded, gnb_predictions))

print("\nComplement Naive Bayes Classification Report:")
print(classification_report(y_test_encoded, cnb_predictions))
```

```
WARNING:gensim.models.word2vec:Each 'sentences' item should be a list of words (usually unicode strings). First item here is inst
Multinomial Naive Bayes Accuracy: 0.7543892769492165
Bernoulli Naive Bayes Accuracy: 0.7543892769492165
Gaussian Naive Bayes Accuracy: 0.192939399660185
Complement Naive Bayes Accuracy: 0.192939399660185

Multinomial Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1022
           1       0.00      0.00      0.00       279
           2       0.75      1.00      0.86      3996

    accuracy                           0.75      5297
   macro avg       0.25      0.33      0.29      5297
```

```
weighted avg       0.57      0.75      0.65      5297


Bernoulli Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1022
           1       0.00      0.00      0.00       279
           2       0.75      1.00      0.86      3996

    accuracy                           0.75      5297
   macro avg       0.25      0.33      0.29      5297
weighted avg       0.57      0.75      0.65      5297


Gaussian Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       0.19      1.00      0.32      1022
           1       0.00      0.00      0.00       279
           2       0.00      0.00      0.00      3996

    accuracy                           0.19      5297
   macro avg       0.06      0.33      0.11      5297
weighted avg       0.04      0.19      0.06      5297


Complement Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       0.19      1.00      0.32      1022
           1       0.00      0.00      0.00       279
           2       0.00      0.00      0.00      3996

    accuracy                           0.19      5297
   macro avg       0.06      0.33      0.11      5297
weighted avg       0.04      0.19      0.06      5297

/usr/local/lib/python3.10/dist-packages/sklearn/naive_bayes.py:515: RuntimeWarning: divide by zero encountered in log
  n_ij = -0.5 * np.sum(np.log(2.0 * np.pi * self.var_[i, :]))
/usr/local/lib/python3.10/dist-packages/sklearn/naive_bayes.py:516: RuntimeWarning: invalid value encountered in divide
  n_ij -= 0.5 * np.sum(((X - self.theta_[i, :]) ** 2) / (self.var_[i, :]), 1)
```

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from gensim.models import Word2Vec
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder

# Memilih fitur dan label
X = data_sentimen['stem_review']
y = data_sentimen['polarity']

# Membagi data menjadi data latih dan uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Training model Word2Vec
word2vec_model = Word2Vec(sentences=X_train, vector_size=100, window=5, min_count=1, workers=4)

# Fungsi untuk mengubah teks menjadi vektor Word2Vec
def word2vec_transformer(X, model):
    return np.array([np.mean([model.wv[word] for word in words.split() if word in model.wv] or [np.zeros(model.vector_size)], axis=0) for

# Melakukan vektorisasi pada data latih dan uji
X_train_word2vec = word2vec_transformer(X_train, word2vec_model)
X_test_word2vec = word2vec_transformer(X_test, word2vec_model)

# Label encoding untuk kelas sentimen
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Inisialisasi model K-Nearest Neighbors
knn_model = KNeighborsClassifier(n_neighbors=5)

# Melatih model K-Nearest Neighbors
knn_model.fit(X_train_word2vec, y_train_encoded)

# Membuat prediksi
knn_predictions = knn_model.predict(X_test_word2vec)

# Menampilkan hasil akurasi dan laporan klasifikasi
print("K-Nearest Neighbors Accuracy:", accuracy_score(y_test_encoded, knn_predictions))
```

```
print("\nK-Nearest Neighbors Classification Report:")
print(classification_report(y_test_encoded, knn_predictions))
```

```
WARNING:gensim.models.word2vec:Each 'sentences' item should be a list of words (usually unicode strings). First item here is instead
K-Nearest Neighbors Accuracy: 0.7543892769492165

K-Nearest Neighbors Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1022
           1       0.00      0.00      0.00       279
           2       0.75      1.00      0.86      3996

    accuracy                           0.75      5297
   macro avg       0.25      0.33      0.29      5297
weighted avg       0.57      0.75      0.65      5297

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
```

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report

# Menggunakan dataset data_sentimen
X = data_sentimen['stem_review']
y = data_sentimen['polarity']

# Membagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Menggunakan CountVectorizer untuk mengubah teks menjadi vektor fitur
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)

# Melatih model Gaussian Naive Bayes
nb_classifier = GaussianNB()
nb_classifier.fit(X_train_vectorized.toarray(), y_train)

# Melatih model K-Nearest Neighbors (KNN)
knn_classifier = KNeighborsClassifier()
knn_classifier.fit(X_train_vectorized, y_train)

# Membuat prediksi menggunakan model Gaussian Naive Bayes
nb_predictions = nb_classifier.predict(X_test_vectorized.toarray())

# Membuat prediksi menggunakan model K-Nearest Neighbors (KNN)
knn_predictions = knn_classifier.predict(X_test_vectorized)

# Menampilkan laporan klasifikasi untuk Gaussian Naive Bayes
print("Laporan Klasifikasi untuk Gaussian Naive Bayes:")
print(classification_report(y_test, nb_predictions))

# Menampilkan laporan klasifikasi untuk K-Nearest Neighbors (KNN)
print("Laporan Klasifikasi untuk K-Nearest Neighbors (KNN):")
print(classification_report(y_test, knn_predictions))
```

```
Laporan Klasifikasi untuk Gaussian Naive Bayes:
              precision    recall  f1-score   support

     negatif       0.23      0.27      0.25       992
      netral       0.06      0.66      0.11       301
     positif       0.79      0.14      0.24      4004

    accuracy                           0.20      5297
   macro avg       0.36      0.36      0.20      5297
weighted avg       0.64      0.20      0.23      5297

Laporan Klasifikasi untuk K-Nearest Neighbors (KNN):
              precision    recall  f1-score   support

     negatif       0.59      0.59      0.59       992
      netral       0.31      0.22      0.26       301
     positif       0.88      0.90      0.89      4004
```

```
          accuracy                           0.80      5297
         macro avg       0.59      0.57      0.58      5297
      weighted avg       0.79      0.80      0.79      5297
```

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from gensim.models import Word2Vec
from nltk.tokenize import word_tokenize
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder

# Membagi data menjadi data latih dan uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Tokenisasi dan pelatihan model Word2Vec
tokenized_text = [word_tokenize(text.lower()) for text in X_train]
word2vec_model = Word2Vec(sentences=tokenized_text, vector_size=100, window=5, min_count=1, workers=4)

# Mengubah data teks menjadi vektor Word2Vec
X_train_word2vec = []
X_test_word2vec = []

# Mengumpulkan vektor Word2Vec untuk setiap dokumen pada data latih
for text in X_train:
    vectors = [word2vec_model.wv[token] for token in word_tokenize(text.lower()) if token in word2vec_model.wv.key_to_index]
    if vectors:
        X_train_word2vec.append(np.mean(vectors, axis=0))
    else:
        # Jika tidak ada token yang ditemukan, tambahkan vektor nol
        X_train_word2vec.append(np.zeros(word2vec_model.vector_size))

# Mengumpulkan vektor Word2Vec untuk setiap dokumen pada data uji
for text in X_test:
    vectors = [word2vec_model.wv[token] for token in word_tokenize(text.lower()) if token in word2vec_model.wv.key_to_index]
    if vectors:
        X_test_word2vec.append(np.mean(vectors, axis=0))
    else:
        # Jika tidak ada token yang ditemukan, tambahkan vektor nol
        X_test_word2vec.append(np.zeros(word2vec_model.vector_size))

# Mengubah list vektor Word2Vec menjadi array numpy
X_train_word2vec = np.array(X_train_word2vec)
X_test_word2vec = np.array(X_test_word2vec)

# Label encoding untuk kelas sentimen
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Algoritma Gaussian Naive Bayes
gnb_model = GaussianNB()
gnb_model.fit(X_train_word2vec, y_train_encoded)
gnb_predictions = gnb_model.predict(X_test_word2vec)

# Algoritma K-Nearest Neighbors (KNN)
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train_word2vec, y_train_encoded)
knn_predictions = knn_model.predict(X_test_word2vec)

# Menampilkan laporan klasifikasi untuk Gaussian Naive Bayes
print("Laporan Klasifikasi untuk Gaussian Naive Bayes:")
print(classification_report(y_test_encoded, gnb_predictions))

# Menampilkan laporan klasifikasi untuk K-Nearest Neighbors (KNN)
print("Laporan Klasifikasi untuk K-Nearest Neighbors (KNN):")
print(classification_report(y_test_encoded, knn_predictions))
```

```
Laporan Klasifikasi untuk Gaussian Naive Bayes:
              precision    recall  f1-score   support

           0       0.32      0.48      0.38      1022
           1       0.11      0.43      0.18       279
           2       0.87      0.59      0.70      3996

    accuracy                           0.56      5297
   macro avg       0.43      0.50      0.42      5297
weighted avg       0.73      0.56      0.61      5297
```

```
Laporan Klasifikasi untuk K-Nearest Neighbors (KNN):
              precision    recall  f1-score   support

           0       0.58      0.44      0.50      1022
           1       0.45      0.17      0.25       279
           2       0.84      0.93      0.88      3996

    accuracy                           0.79      5297
   macro avg       0.62      0.51      0.54      5297
weighted avg       0.77      0.79      0.77      5297
```

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from gensim.models import Word2Vec
from nltk.tokenize import word_tokenize
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.preprocessing import LabelEncoder

# Anda akan memerlukan data_sentimen, pastikan untuk memuatnya sebelum menjalankan skrip ini.

X = data_sentimen['stem_review']
y = data_sentimen['polarity']

# Membagi data menjadi data latih dan uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Tokenisasi dan pelatihan model Word2Vec
tokenized_text = [word_tokenize(text.lower()) for text in X_train]
word2vec_model = Word2Vec(sentences=tokenized_text, vector_size=100, window=5, min_count=1, workers=4)

# Mengubah data teks menjadi vektor Word2Vec
X_train_word2vec = []
X_test_word2vec = []

# Mengumpulkan vektor Word2Vec untuk setiap dokumen pada data latih
for text in X_train:
    vectors = [word2vec_model.wv[token] for token in word_tokenize(text.lower()) if token in word2vec_model.wv.key_to_index]
    if vectors:
        X_train_word2vec.append(np.mean(vectors, axis=0))
    else:
        # Jika tidak ada token yang ditemukan, tambahkan vektor nol
        X_train_word2vec.append(np.zeros(word2vec_model.vector_size))

# Mengumpulkan vektor Word2Vec untuk setiap dokumen pada data uji
for text in X_test:
    vectors = [word2vec_model.wv[token] for token in word_tokenize(text.lower()) if token in word2vec_model.wv.key_to_index]
    if vectors:
        X_test_word2vec.append(np.mean(vectors, axis=0))
    else:
        # Jika tidak ada token yang ditemukan, tambahkan vektor nol
        X_test_word2vec.append(np.zeros(word2vec_model.vector_size))

# Mengubah list vektor Word2Vec menjadi array numpy
X_train_word2vec = np.array(X_train_word2vec)
X_test_word2vec = np.array(X_test_word2vec)

# Label encoding untuk kelas sentimen
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Algoritma Gaussian Naive Bayes
gnb_model = GaussianNB()
gnb_model.fit(X_train_word2vec, y_train_encoded)
gnb_predictions = gnb_model.predict(X_test_word2vec)
gnb_accuracy = accuracy_score(y_test_encoded, gnb_predictions)
gnb_precision = precision_score(y_test_encoded, gnb_predictions, average='weighted')
gnb_recall = recall_score(y_test_encoded, gnb_predictions, average='weighted')
gnb_f1_score = f1_score(y_test_encoded, gnb_predictions, average='weighted')

# Algoritma K-Nearest Neighbors (KNN)
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train_word2vec, y_train_encoded)
knn_predictions = knn_model.predict(X_test_word2vec)
knn_accuracy = accuracy_score(y_test_encoded, knn_predictions)
knn_precision = precision_score(y_test_encoded, knn_predictions, average='weighted')
knn_recall = recall_score(y_test_encoded, knn_predictions, average='weighted')
knn_f1_score = f1_score(y_test_encoded, knn_predictions, average='weighted')

# Menampilkan akurasi kedua model
print("Akurasi Gaussian Naive Bayes (Word2Vec):", gnb_accuracy)
print("Akurasi K-Nearest Neighbors (Word2Vec):", knn_accuracy)

# Rangkuman Data
summary_data = {
    'Metrik': ['Presisi', 'Recall', 'F1-Score'],
    'Gaussian Naive Bayes (GNB)': [gnb_precision, gnb_recall, gnb_f1_score],
    'K-Nearest Neighbors (KNN)': [knn_precision, knn_recall, knn_f1_score]
}
```

```
# Membuat dataframe dari struktur data
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from gensim.models import Word2Vec
from nltk.tokenize import word_tokenize
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.preprocessing import LabelEncoder

# Anda akan memerlukan data_sentimen, pastikan untuk memuatnya sebelum menjalankan skrip ini.

X = data_sentimen['stem_review']
y = data_sentimen['polarity']

# Membagi data menjadi data latih dan uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Tokenisasi dan pelatihan model Word2Vec
tokenized_text = [word_tokenize(text.lower()) for text in X_train]
word2vec_model = Word2Vec(sentences=tokenized_text, vector_size=100, window=5, min_count=1, workers=4)

# Mengubah data teks menjadi vektor Word2Vec
X_train_word2vec = []
X_test_word2vec = []

# Mengumpulkan vektor Word2Vec untuk setiap dokumen pada data latih
for text in X_train:
    vectors = [word2vec_model.wv[token] for token in word_tokenize(text.lower()) if token in word2vec_model.wv.key_to_index]
    if vectors:
        X_train_word2vec.append(np.mean(vectors, axis=0))
    else:
        # Jika tidak ada token yang ditemukan, tambahkan vektor nol
        X_train_word2vec.append(np.zeros(word2vec_model.vector_size))

# Mengubah list vektor Word2Vec menjadi array numpy
X_train_word2vec = np.array(X_train_word2vec)

# Label encoding untuk kelas sentimen
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)

# Algoritma Gaussian Naive Bayes
gnb_model = GaussianNB()
gnb_cv_scores = cross_val_score(gnb_model, X_train_word2vec, y_train_encoded, cv=10)

# Algoritma K-Nearest Neighbors (KNN)
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_cv_scores = cross_val_score(knn_model, X_train_word2vec, y_train_encoded, cv=10)

# Menampilkan nilai 10-fold cross validation
print("Nilai 10-fold cross validation untuk Gaussian Naive Bayes:")
print(gnb_cv_scores)
print("Rata-rata nilai cross validation:", np.mean(gnb_cv_scores))

print("\nNilai 10-fold cross validation untuk K-Nearest Neighbors:")
print(knn_cv_scores)
print("Rata-rata nilai cross validation:", np.mean(knn_cv_scores))
```

```
Nilai 10-fold cross validation untuk Gaussian Naive Bayes:
[0.52524776 0.51297782 0.50353941 0.52005663 0.52571968 0.50967437
 0.49174139 0.50920245 0.5368272  0.51935788]
Rata-rata nilai cross validation: 0.5154344589466854

Nilai 10-fold cross validation untuk K-Nearest Neighbors:
[0.79848985 0.79660217 0.79754601 0.80132138 0.79329873 0.79424257
 0.7881076  0.80132138 0.80028329 0.79131256]
Rata-rata nilai cross validation: 0.7962525528950042
```

```
import numpy as np
import pandas as pd
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
```