

Kode Program dan Hasil *Scraping* Menggunakan Google Colab dengan Bahasa Python

1. Import Library

```
import pandas as pd
from googleapiclient.discovery import build
```

```

def video_comments(video_id, api_key):
    # early list for storing reply
    replies = []

    # creating youtube resource object
    youtube = build('youtube', 'v3', developerKey=api_key)

    # retrieve youtube video results
    video_response = youtube.commentThreads().list(part='snippet,replies', videoId=video_id).execute()

    # iterate video response
    while video_response:
        # extracting required info
        # from each result object
        for item in video_response['items']:

            # Extracting comments {}
            published = item['snippet']['topLevelComment']['publishedAt']
            user = item['snippet']['topLevelComment']['snippet']['authorDisplayName']

            # Extracting comments
            comment = item['snippet']['topLevelComment']['snippet']['textDisplay']
            likecount = item['snippet']['topLevelComment']['snippet']['likeCount']

            replies.append([published, user, comment, likecount])

        # counting number of reply of comment
        replycount = item['snippet']['totalReplyCount']

        # if reply is there
        if replycount:
            # iterate through all reply
            for reply in item['replies']['comments']:

                # Extract reply
                published = reply['snippet']['publishedAt']
                user = reply['snippet']['authorDisplayName']
                rep = reply['snippet']['textDisplay']
                likecount = reply['snippet']['likeCount']

                # Store reply in list
                Replies.append(reply)
                replies.append([published, user, rep, likecount])

        # print comment with list of reply
        print(comment, replies, and = '\n\n')

    # empty reply list
    Replies = []

    # Again repeat
    if 'nextPageToken' in video_response:
        video_response = youtube.commentThreads().list(
            part = 'snippet,replies',
            pageToken = video_response['nextPageToken'],
            videoId = video_id
        ).execute()
    else:
        break
    return replies

```

[illegible][illegible]

3. Ubah Hasil *Scraping* ke *DataFrame*

4. Ubah Hasil Scraping ke Dataframe

```
df = pd.DataFrame(comments, columns=['publishedAt', 'authorDisplayName', 'textDisplay', 'likeCount'])
df
```

	publishedAt	authorDisplayName	textDisplay	likeCount
0	2026-03-28T00:43:53Z	@dennyload1149	yang paping pabiknya hijau atau coklat?	0
1	2026-02-03T04:55:49Z	@rizkisapari3113	Motis akaQn luhut ... di US aja udah mulai je...	0
2	2026-01-18T23:08:23Z	@wellymarch343	Investor : persasaan kemarin ngak gri deh pe...	0
3	2026-01-15T19:25:26Z	@kamujahat1384	Mending kasih insentif ev ...sh ...mangurangi...	0
4	2026-01-11T00:04:04Z	@aguspr1663	Harusnya jgn di hilangkan insentifnya, tapi di...	0
...	...	...	...	...
226	2025-12-20T10:08:36Z	@demadaarmand02948	Harga jual kembali lebih stabil kali Bq	1
227	2025-12-20T06:39:11Z	@henyDagabumomo5691	Konsistensi kebijakan barang langka di konoha	22
228	2025-12-19T22:01:28Z	@sogeking	Segitunya wkwk... Yg udah terlanjur beli mob...	8
229	2025-12-20T10:26:31Z	@persbday	@sogeking apakah Reviewer tentang mobil list...	0
230	2025-12-20T13:24:37Z	@waterheart123	Bukan nya bagus yg sudah beli ev yet. Sdh opt ...	1

231 rows x 4 columns

4. Simpan Hasil *Scraping* ke file *.csv*

5. Simpan Hasil Scraping ke file CSV

```
df.to_csv('youtube-comments_tespodotco.csv', index=False)
```



Lampiran 2 Kode Program dan Hasil *Preprocessing* menggunakan Google Colab dengan Bahasa Pemrograman Python

1. *Import Library* dan *Import Data*

```
1. Import Library

import pandas as pd
import numpy as np
from nltk.corpus import stopwords
import re
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

```
2. Import Data

from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

import pandas as pd

file_path = '/content/drive/MyDrive/realis Data Komentar YT/comments_youtube_2kelas_UJI_Labelcleaned.csv'

# Attempting to read the CSV directly with comma as separator.
# Removed on line 10 as it might interfere with correct parsing if header is complex.
data = pd.read_csv(file_path, encoding='latin1', sep=',')

# Rename 'label' column to 'pelabelan' for consistency with subsequent steps
if 'label' in data.columns:
    data.rename(columns={'label': 'pelabelan'}, inplace=True)

data.info()
data.head()
```

	I-jauthor	comment	pelabelan
0	@GendowarLowo	Omon lagi aja. 10	1.0
1	@ada5066	Tau kan ga kompetennya pemerintah razim ini. g...	1.0
2	@sacharbinmaun8997	KLO bisa semua di pajakin, lumayan hasil pajak...	1.0
3	@beimanspayungchanne8710	Gak adl dong mobil listrik o pajak pick up z...	2.0
4	@kpa262	Lagian udah kaya orang baru aja jadi WNI 0. 10	1.0

2. Proses *Cleaning*

PROSES CLEANING

```
data.info()
data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2278 entries, 0 to 2297
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ---
 0   I-jauthor   2278 non-null    object
 1   comment     2277 non-null    object
 2   pelabelan   2277 non-null    Float64
dtypes: float64(1), object(2)
memory usage: 54.0+ KB
```

	I-jauthor	comment	pelabelan
0	@GendowarLowo	Omon lagi aja. 10	1.0
1	@ada5066	Tau kan ga kompetennya pemerintah razim ini. g...	1.0
2	@sacharbinmaun8997	KLO bisa semua di pajakin, lumayan hasil pajak...	1.0
3	@beimanspayungchanne8710	Gak adl dong mobil listrik o pajak pick up z...	2.0
4	@kpa262	Lagian udah kaya orang baru aja jadi WNI 0. 10	1.0

```
# Menghapus duplikat dari kolom 'comment' di DataFrame 'data' utama.
# ini akan menghapus baris-baris duplikat tetapi mempertahankan semua kolom yang ada.
data.drop_duplicates(subset='comment', inplace=True)

data.info()
data.head(10)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2277 entries, 0 to 2267
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ---
 0   I-jauthor   2277 non-null    object
 1   comment     2276 non-null    object
 2   pelabelan   2276 non-null    Float64
dtypes: float64(1), object(2)
memory usage: 71.2+ KB
```

	I-jauthor	comment	pelabelan
0	@GendowarLowo	Omon lagi aja. 10	1.0
1	@ada5066	Tau kan ga kompetennya pemerintah razim ini. g...	1.0
2	@sacharbinmaun8997	KLO bisa semua di pajakin, lumayan hasil pajak...	1.0
3	@beimanspayungchanne8710	Gak adl dong mobil listrik o pajak pick up z...	2.0
4	@kpa262	Lagian udah kaya orang baru aja jadi WNI 0. 10	1.0
5	@aditya1168	Ditipu pemerintah ya	1.0
6	@BudiKeburan-az2	Bikin jalan sendiri aja	2.0
7	@hanantaominto5710	BATMAN: semua sudah diatur dari awal, kalian m...	1.0
8	@era_media_H2O	Itulah kalo dipimpin negara gagal, pasti blang...	1.0
9	@AHurumaSeptaMaiana17	Pengguna mobil listrik saat ini: Vn1. Pajak ta...	1.0

```
data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 2277 entries, 0 to 2267
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ---
 0   I-jauthor   2277 non-null    object
 1   comment     2276 non-null    object
 2   pelabelan   2276 non-null    Float64
dtypes: float64(1), object(2)
memory usage: 71.2+ KB
```

Data tersisa 2277

Menghapus url, emoji, angka, simbol, username yang ada di kolom **comment**

```
import re
import string
import nltk

# fungsi menghapus URL
def remove_url(text):
    url = re.compile(r'https?://(S+|www\.)S+')
    return url.sub(r'', text)

# fungsi menghapus emoji
def remove_emoji(text):
    if text is not None and isinstance(text, str):
        emoji_pattern = re.compile("["
            u'\U0001F600-\U0001F64F'  # emoticons
            u'\U0001F300-\U0001F5FF'  # symbols & pictographs
            u'\U0001F680-\U0001F6FF'  # transport & map symbols
            u'\U0001F700-\U0001F7FF'  # alchemical symbols
            u'\U0001F780-\U0001F7FF'  # Geometric Shapes Extended
            u'\U0001F800-\U0001F8FF'  # Supplemental Arrows-C
            u'\U0001F900-\U0001F9FF'  # Chess Symbols
            u'\U0001FA00-\U0001FAFF'  # Symbols and Pictographs Extended-A
            u'\U0001F000-\U0001F0FF'  # Additional emoticons
            u'\U0001F100-\U0001F1FF'  # flags
            "]")
        return emoji_pattern.sub(r'', text)
    else:
        return text

# fungsi menghapus simbol
def remove_symbols(text):
    if text is not None and isinstance(text, str):
        text = re.sub(r'["-@-9%]', '', text)
        return text
    else:
        return text

# fungsi menghapus angka
def remove_number(text):
    if text is not None and isinstance(text, str):
        text = re.sub(r'[0-9]', '', text)
        return text
    else:
        return text

# fungsi menghapus username
def remove_username(text):
    if text is not None and isinstance(text, str):
        return re.sub(r'@user', '', text)
    else:
        return text

# Fill NaN values in 'comment' column with empty strings before cleaning
data['comment'] = data['comment'].fillna('') # Use data instead of df

data['cleaning'] = data['comment'].apply(lambda x: remove_url(x)) # Use data
data['cleaning'] = data['cleaning'].apply(lambda x: remove_emoji(x)) # Use data
data['cleaning'] = data['cleaning'].apply(lambda x: remove_symbols(x)) # Use data
data['cleaning'] = data['cleaning'].apply(lambda x: remove_number(x)) # Use data
data['cleaning'] = data['cleaning'].apply(lambda x: remove_username(x)) # Use data
data[['comment', 'cleaning']].head(10)
```

	comment	cleaning
0	Omon lagi aja.👉👉👉	Omon lagi aja
1	Tau kan ga kompetennya pemerintah razim ini, g...	Tau kan ga kompetennya pemerintah razim ini ga...
2	KLO bisa semua di pajakin, lumayan hasil pajak...	KLO bisa semua di pajakin lumayan hasil pajak...
3	Gak adli dong mobil listrik o pajak pick up z ...	Gak adli dong mobil listrik o pajak pick up z b...
4	Lagian udah kaya orang baru aja jadi WNI 🇮🇩🇪	Lagian udah kaya orang baru aja jadi WNI
6	Ditipu pemerintah ya	Ditipu pemerintah ya
6	Bikin jalan sendiri aja	Bikin jalan sendiri aja
7	BATMAN: semua sudah diatur dari awal, kalian m...	BATMAN semua sudah diatur dari awal kalian mas...
8	Itulah kalo dipimpin negara gagal, pasti blang...	Itulah kalo dipimpin negara gagal pasti blangs...
9	Pengguna mobil listrik saat ini: 1.1. Pajak ta...	Pengguna mobil listrik saat ini 1.1. Pajak tahun...

3. Proses Case Folding

CASE FOLDING

```
def case_folding(text):
    if isinstance(text, str):
        lowercase_text = text.lower()
        return lowercase_text
    else:
        return text

data['case_folding'] = data['cleaning'].apply(case_folding) # Use data
data[['comment', 'cleaning', 'case_folding']].head(10)
```

	comment	cleaning	case_folding
0	Omon lagi aja.👉👉👉	Omon lagi aja	omon lagi aja
1	Tau kan ga kompetennya pemerintah razim ini, g...	Tau kan ga kompetennya pemerintah razim ini ga...	tau kan ga kompetennya pemerintah razim ini ga...
2	KLO bisa semua di pajakin, lumayan hasil pajak...	KLO bisa semua di pajakin lumayan hasil pajak...	klo bisa semua di pajakin lumayan hasil pajak...
3	Gak adli dong mobil listrik o pajak pick up z ...	Gak adli dong mobil listrik o pajak pick up z b...	gak adli dong mobil listrik o pajak pick up z b...
4	Lagian udah kaya orang baru aja jadi WNI 🇮🇩🇪	Lagian udah kaya orang baru aja jadi WNI	lagian udah kaya orang baru aja jadi wni
6	Ditipu pemerintah ya	Ditipu pemerintah ya	ditipu pemerintah ya
6	Bikin jalan sendiri aja	Bikin jalan sendiri aja	bikin jalan sendiri aja
7	BATMAN: semua sudah diatur dari awal, kalian m...	BATMAN semua sudah diatur dari awal kalian mas...	batman semua sudah diatur dari awal kalian mas...
8	Itulah kalo dipimpin negara gagal, pasti blang...	Itulah kalo dipimpin negara gagal pasti blangs...	itulah kalo dipimpin negara gagal pasti blangs...
9	Pengguna mobil listrik saat ini: 1.1. Pajak ta...	Pengguna mobil listrik saat ini 1.1. Pajak tahun...	pengguna mobil listrik saat ini 1.1. pajak tahun...

NORMALISASI

```
import pandas as pd
import requests
```

TOKENIZING

STOPWORD REMOVAL

fungsinya menghapus kata sambung

[illegible]

sebelum dilakukan tokenizing lanjut kita print atau kita cek frekuensi kata dengan jumlah kemunculan terbanyak sampai terkedi, dan akan coba ditampilkan 50 kata teratas

```
from collections import Counter

all_text = ''
for i,dataf in enumerate(csv_data): # looped di to data
    all_text += dataf['text'+str(i)] + '\n' # simplified regex, removed "" from previous all_text + re.sub(r'"|'|"|'",'', all_text.lower()) that sum removing letters

word_list = all_text.split()
word_freq = Counter(word_list)
sorted_freq = sorted(word_freq.items(), key=lambda x: x[1], reverse=True)
df_freq = pd.DataFrame(sorted_freq, columns=['kata', 'frekuensi'])

df_freq.to_csv('frekuensi_kata_mali.csv', index=False)
print('File "frekuensi_kata_mali.csv" berhasil disimpan')
df_freq.info()

#file "frekuensi_kata_mali.csv" berhasil disimpan.
cetak_gambar_csv(frekuensi_dataframe)
mengurutkan: kata tertinggi, & ke turun
data columns: [kita] 2 column(s)
# Column Non-Null Count (Type
non-null values
# Kata Non-Null Count (Type
1 frequency: kaka non-null object
df_freq.info() df_freq()
empty string: df_freq.isnull().sum()
```

```
kata frekuensi
0 papak 835
1 nudi 609
2 lera 506
3 ya 465
4 perawitrah 420
5 ngapan 322
6 meut 254
7 indonesia 250
8 jalan 245
9 bel 221
10 bayar 220
11 kerdewan 220
12 orang 221
13 su 210
14 kaka 208
15 kayak 190
16 paki 185
17 subdi 182
18 mika 176
19 kapten 136
20 insani 136
21 kerdwan 110
22 laka 105
23 alun 103
24 bira 102
```

[illegible]

[illegible]

9. Proses Pembobotan Kata TF-IDF

Pembobotan Kata (TF-IDF)

```

from sklearn.feature_extraction.text import TfidfVectorizer

# Inisialisasi TfidfVectorizer
# max_features dapat digunakan untuk membatasi jumlah kata yang dipertimbangkan
tfidf_vectorizer = TfidfVectorizer(max_features=1000)

# Fit dan transformasikan data
tfidf_matrix = tfidf_vectorizer.fit_transform(docs)

# Dapatkan nama-nama fitur (kata-kata)
feature_names = tfidf_vectorizer.get_feature_names_out()

# Buat DataFrame dari TF-IDF matrix
df_tfidf = pd.DataFrame(tfidf_matrix.toarray(), columns=feature_names)

print("Ukuran TF-IDF matrix (Barisan x Kolom):", df_tfidf.shape)
print("Ukuran 11 baris pertama dari DataFrame TF-IDF (Baris beberapa kata pertama):")
print(df_tfidf.iloc[:11, :10]) # Sempitkan 10 baris pertama dan 10 kata pertama

# Anda juga bisa melihat nilai TF-IDF rata-rata untuk setiap kata
# Ini akan memberikan kata-kata yang secara umum penting di seluruh korpus
mean_tfidf_scores = df_tfidf.mean(axis=0)
print("Rata-rata TF-IDF untuk setiap kata:")
print(mean_tfidf_scores.head(20))

"""
Ukuran TF-IDF matrix (Barisan x Kolom): (2277, 1000)

Carilah 10 baris pertama dari DataFrame TF-IDF (Baris beberapa kata pertama):
Barisan adalah string dari string abjad. Baris adalah:
0 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0
1 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0
2 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0
3 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0
4 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0
5 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0
6 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0
7 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0
8 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0
9 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0
10 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0
11 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0
12 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0
13 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0
14 0.0 0.0 0.000000 0.0 0.0 0.0 0.0 0.0

Baris adalah string dari string abjad. Baris adalah:
0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.000000
1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.000000
2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.000000
3 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.000000
4 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.000000
5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.000000
6 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.000000
7 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.000000
8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.000000
9 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.000000
10 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.000000
11 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.000000
12 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.000000
13 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.000000
14 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.000000

Baris adalah string dari string abjad. Baris adalah:
0 0.0 0.0
1 0.0 0.0
2 0.0 0.0
3 0.0 0.0
4 0.0 0.0
5 0.0 0.0
6 0.0 0.0
7 0.0 0.0
8 0.0 0.0
9 0.0 0.0
10 0.0 0.0
11 0.0 0.0
12 0.0 0.0
13 0.0 0.0
14 0.0 0.0

Rata-rata TF-IDF untuk setiap kata:
Baris adalah string dari string abjad. Baris adalah:
0 0.000000
1 0.000000
2 0.000000
3 0.000000
4 0.000000
5 0.000000
6 0.000000
7 0.000000
8 0.000000
9 0.000000
10 0.000000
11 0.000000
12 0.000000
13 0.000000
14 0.000000
"""

```



Lampiran 3 Kode Program dan Hasil Implementasi *Naive Bayes* dan *K-Nearest Neighbor* menggunakan Google Colab dengan Bahasa Pemrograman Python

1. Split Data Latih dan Data Uji (80 : 20)

Split Dataset dengan 80% Data Latih dan 20% Data Uji

```
from sklearn.model_selection import train_test_split

# Definisikan fitur (X) dan target (y)
# X akan menjadi kolom 'stemming_data' yang berisi teks yang sudah diproses
# y akan menjadi kolom 'pelabelan' yang merupakan label kelas
# Ilustrasi:
# Nilai 1 pada 'pelabelan' = Komentar Negatif
# Nilai 2 pada 'pelabelan' = Komentar Positif
X = data['stemming_data']
y = data['pelabelan']

# Bagi data menjadi 80% data latih dan 20% data uji
train_test_split adalah fungsi dari scikit-learn yang melakukan pembagian dataset.
# Parameter:
# - 'X': Fitur atau data input yang akan dibagi.
# - 'y': Target atau label yang sesuai dengan X.
# - 'test_size=0.2': Menentukan proporsi data uji. Dalam kasus ini, 20% dari total data akan menjadi data uji.
# - 'random_state=42': Mengisi seed untuk generator angka acak. Ini memastikan bahwa setiap kali kode dijalankan,
# pembagian data akan selalu menghasilkan set yang sama, sehingga hasil eksperimen menjadi reproducible (dapat direplikasi).
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print(f"Ukuran data latih (X_train): {X_train.shape}")
print(f"Ukuran label latih (y_train): {y_train.shape}")
print(f"Ukuran data uji (X_test): {X_test.shape}")
print(f"Ukuran label uji (y_test): {y_test.shape}")

Ukuran data latih (X_train): (1821,)
Ukuran label latih (y_train): (1821,)
Ukuran data uji (X_test): (456,)
Ukuran label uji (y_test): (456,)

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB

# Transformasi data latih (X_train) dan data uji (X_test) menggunakan TF-IDF
# fit_transform hanya dilakukan pada X_train untuk mempelajari vocabulary
X_train_tfidf = TfidfVectorizer().fit_transform(X_train)

# transform dilakukan pada X_test menggunakan vocabulary yang sudah dipelajari dari X_train
X_test_tfidf = TfidfVectorizer().transform(X_test)

print(f"Ukuran X_train_tfidf: {X_train_tfidf.shape}")
print(f"Ukuran X_test_tfidf: {X_test_tfidf.shape}")

Ukuran X_train_tfidf: (1821, 3588)
Ukuran X_test_tfidf: (456, 3588)
```

2. Implementasi *Naive Bayes*

Implementasi Naive Bayes

```
from sklearn.naive_bayes import MultinomialNB

# Inisialisasi model Naive Bayes (MultinomialNB cocok untuk fitur hitungan/frekuensi seperti TF-IDF)
model_naive_bayes = MultinomialNB()

# Handle NaN values in y_train before fitting
# Identifikasi indeks yang memiliki NaN di y_train
nan_indices = y_train.isna()

# Hapus baris dengan NaN dari X_train_tfidf dan y_train
# Untuk sparse matrix, gunakan .values untuk mendapatkan array boolean dari Series pandas
X_train_tfidf_cleaned = X_train_tfidf[~nan_indices.values]
y_train_cleaned = y_train[~nan_indices]

# Latih model menggunakan data latih TF-IDF yang sudah bersih dan label latih yang sudah bersih
model_naive_bayes.fit(X_train_tfidf_cleaned, y_train_cleaned)

print("Model Naive Bayes berhasil dilatih!")

Model Naive Bayes berhasil dilatih!
```


3. Evaluasi Naive Bayes

Evaluasi Naive Bayes

```
from sklearn.metrics import classification_report, roc_auc_score, RocCurveDisplay, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import numpy as np # Import numpy for where

# Prediksi pada data uji
y_pred = model_naive_bayes.predict(X_test_tf10f)

# Prediksi probabilitas untuk kurva ROC-AUC
# Pastikan urutan kelas sesuai. MultinomialNB biasanya mengurutkan kelas secara numerik.
# Jika kelas adalah 1.0 dan 2.0, maka probabilitas untuk kelas 2.0 (positif) adalah pada indeks 1.
y_pred_proba = model_naive_bayes.predict_proba(X_test_tf10f)[:, 1]

# --- Matrik Evaluasi ---
print("\n=== Evaluasi Model Naive Bayes ===")

# Akurasi
accuracy = accuracy_score(y_test, y_pred)
print(f"Akurasi: {accuracy:.4f}")

# Classification Report (Precision, Recall, F1 Score)
# Pastikan target_names sesuai dengan label di y_test, jika tidak, bisa menyebabkan output membingungkan.
# Karena label kita 1.0 dan 2.0, kita bisa langsung menggunakan classification_report.
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# ROC-AUC
# Transform y_test to binary (0 and 1) for roc_auc_score
# Assuming 2.0 is the positive class, map 2.0 to 1 and 1.0 to 0
y_test_binary = np.where(y_test == 2.0, 1, 0)

# The y_pred_proba[:, 1] already corresponds to the probability of class 2.0 (our positive class).
# So we can use it directly with the binarized y_test.
roc_auc = roc_auc_score(y_test_binary, y_pred_proba) # Removed pos_label
print(f"ROC AUC Score: {roc_auc:.4f}")

# --- Plot Kurva ROC ---
plt.figure(figsize=(8, 6))
# RocCurveDisplay.from_predictions expects binary labels for y, true (0/1)
RocCurveDisplay.from_predictions(y_test_binary, y_pred_proba, name='Naive Bayes ROC', ax=plt.gca()) # Removed pos_label
plt.title('Kurva ROC Naive Bayes')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.grid(True)
plt.show()
```

=== Evaluasi Model Naive Bayes ===

Akurasi: 0.8818

Classification Report:

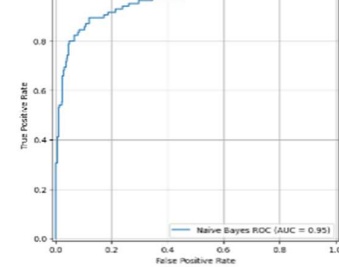
	precision	recall	f1-score	support
1.0	0.85	1.00	0.92	371
2.0	1.00	0.26	0.41	85
accuracy			0.88	456
macro avg	0.93	0.63	0.67	456
weighted avg	0.88	0.86	0.83	456

Confusion Matrix:

[[371 0]

[85 22]]

ROC AUC Score: 0.9489



```
from imblearn.over_sampling import SMOTE
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, roc_auc_score, RocCurveDisplay, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import numpy as np

print("\n=== Implementasi SMOTE untuk menyeimbangkan data latih ===")
# Inisialisasi SMOTE
# random_state untuk reproduktifitas
smote = SMOTE(random_state=42)

# Terapkan SMOTE pada data latih yang sudah bersih, setelah penanganan outlier
X_train_res, y_train_res = smote.fit_resample(X_train_cleaned, y_train_cleaned)

print(f"Jumlah sampel sebelum SMOTE: {len(X_train_cleaned)}")
print(f"Distribusi kelas sebelum SMOTE: {X_train_cleaned.value_counts()}")
print(f"Jumlah sampel setelah SMOTE: {len(X_train_res)}")
print(f"Distribusi kelas setelah SMOTE: {X_train_res.value_counts()}")

print("\n=== Melatih Model Naive Bayes dengan Data yang Diseimbangkan (SMOTE) ===")
# Inisialisasi dan latih model Naive Bayes baru dengan data yang sudah diseimbangkan
model_naive_bayes_smote = MultinomialNB()
model_naive_bayes_smote.fit(X_train_res, y_train_res)

print("Model Naive Bayes (dengan SMOTE) berhasil dilatih!")

print("\n=== Evaluasi Model Naive Bayes dengan Data yang Diseimbangkan (SMOTE) ===")
# Prediksi pada data uji (tetap menggunakan X_test_tf10f yang asli)
y_pred_smote = model_naive_bayes_smote.predict(X_test_tf10f)

# Prediksi probabilitas untuk kurva ROC-AUC
y_pred_proba_smote = model_naive_bayes_smote.predict_proba(X_test_tf10f)[:, 1]

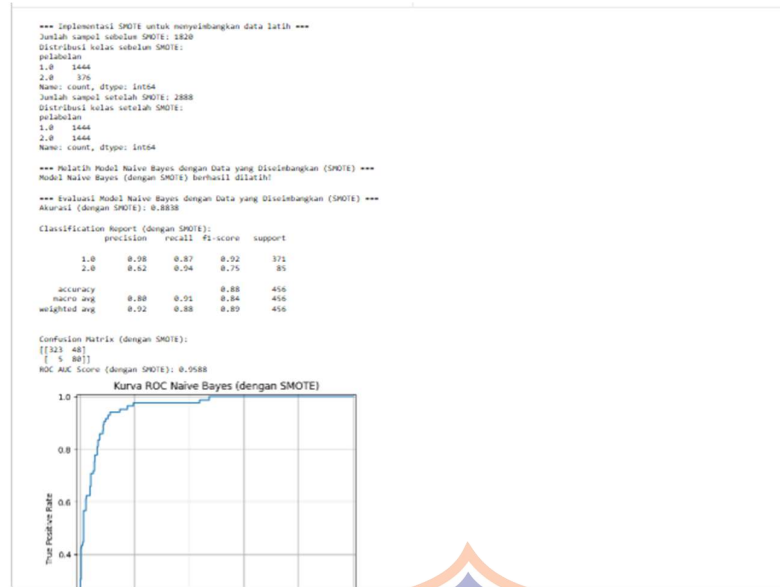
# Akurasi
accuracy_smote = accuracy_score(y_test, y_pred_smote)
print(f"Akurasi (dengan SMOTE): {accuracy_smote:.4f}")

# Classification Report
print("\nClassification Report (dengan SMOTE):")
print(classification_report(y_test, y_pred_smote))

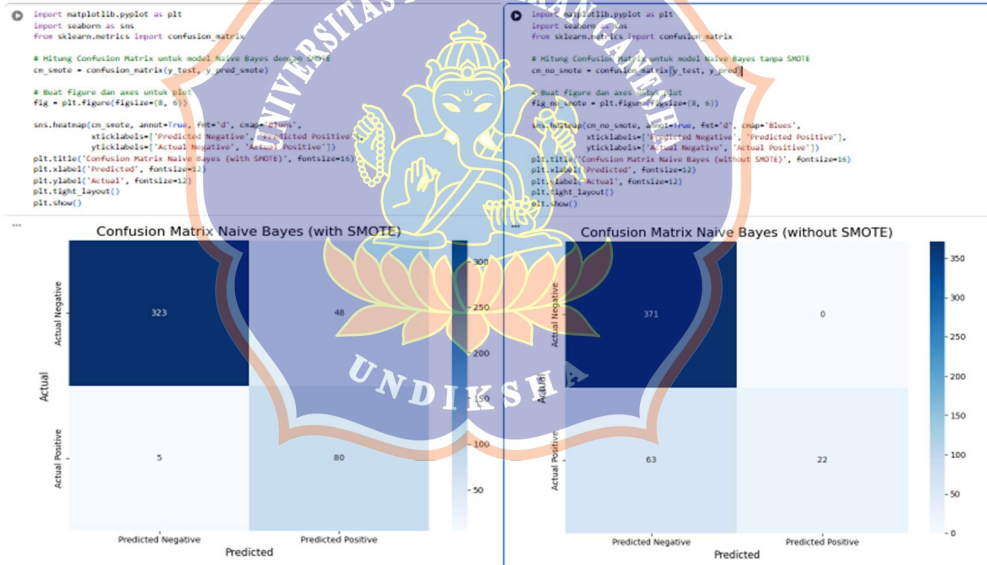
# Confusion Matrix
print("\nConfusion Matrix (dengan SMOTE):")
print(confusion_matrix(y_test, y_pred_smote))

# ROC-AUC
# Transform y_test to binary (0 and 1) for roc_auc_score (assuming 2.0 is positive class)
y_test_binary = np.where(y_test == 2.0, 1, 0)
roc_auc_smote = roc_auc_score(y_test_binary, y_pred_proba_smote)
print(f"ROC AUC Score (dengan SMOTE): {roc_auc_smote:.4f}")

# --- Plot Kurva ROC (dengan SMOTE) ---
plt.figure(figsize=(8, 6))
RocCurveDisplay.from_predictions(y_test_binary, y_pred_proba_smote, name='Naive Bayes ROC (SMOTE)', ax=plt.gca())
plt.title('Kurva ROC Naive Bayes (dengan SMOTE)')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.grid(True)
plt.show()
```



4. Visualisasi *Confusion Matrix* dari model *Naive Bayes*



5. Implementasi *K-Nearest Neighbor*

Implementasi dan Evaluasi Model K-Nearest Neighbors (KNN)

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, roc_auc_score, RocCurveDisplay, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import numpy as np

# 1. KNN Tanpa SMOTE dan Tuning n_neighbors
# ===== Melatih dan Mengevaluasi Model K-Nearest Neighbors (KNN) Tanpa SMOTE =====
n_neighbors_values = [1, 5, 7, 9, 11]
best_accuracy_no_smote = 0
best_n_no_smote = None
best_model_no_smote = None
best_y_pred_proba_no_smote = None
best_report_no_smote = ""
best_cm_no_smote = None
best_roc_auc_no_smote = 0

for n in n_neighbors_values:
    print(f"--- Evaluasi KNN dengan n_neighbors = {n} (Tanpa SMOTE) ---")
    model_knn_no_smote = KNeighborsClassifier(n_neighbors=n)

    # Latih model KNN menggunakan data latih yang sudah bersih (Tanpa SMOTE)
    model_knn_no_smote.fit(X_train_tfidf_cleaned, y_train_cleaned)

    # Prediksi pada data uji
    y_pred_knn_no_smote_current = model_knn_no_smote.predict(X_test_tfidf)

    # Prediksi probabilitas untuk kurva ROC-AUC
    y_pred_proba_knn_no_smote_current = model_knn_no_smote.predict_proba(X_test_tfidf)[:, 1]

    # Akurasi
    current_accuracy_no_smote = accuracy_score(y_test, y_pred_knn_no_smote_current)
    current_report_no_smote = classification_report(y_test, y_pred_knn_no_smote_current)
    current_cm_no_smote = confusion_matrix(y_test, y_pred_knn_no_smote_current)
    current_roc_auc_no_smote = roc_auc_score(y_test_binary, y_pred_proba_knn_no_smote_current)

    print(f"Akurasi (n_neighbors={n}): {current_accuracy_no_smote:.4f}")
    print("Classification Report:")
    print(current_report_no_smote)
    print("Confusion Matrix:")
    print(current_cm_no_smote)
    print(f"ROC AUC Score (n_neighbors={n}): {current_roc_auc_no_smote:.4f}")

    if current_accuracy_no_smote > best_accuracy_no_smote:
        best_accuracy_no_smote = current_accuracy_no_smote
        best_n_no_smote = n
        best_model_no_smote = model_knn_no_smote
        best_y_pred_proba_no_smote = y_pred_proba_knn_no_smote_current
        best_report_no_smote = current_report_no_smote
        best_cm_no_smote = current_cm_no_smote
        best_roc_auc_no_smote = current_roc_auc_no_smote

print(f"===== Hasil Evaluasi KNN Tanpa SMOTE =====")
print(f"KNN Terbaik Tanpa SMOTE adalah n_neighbors = {best_n_no_smote}")
print(f"Akurasi Terbaik (Tanpa SMOTE): {best_accuracy_no_smote:.4f}")
print("Classification Report Terbaik (Tanpa SMOTE):")
print(best_report_no_smote)
print("Confusion Matrix Terbaik (Tanpa SMOTE):")
print(best_cm_no_smote)
print(f"ROC AUC Score Terbaik (Tanpa SMOTE): {best_roc_auc_no_smote:.4f}")
print(f"===== Hasil Evaluasi KNN Tanpa SMOTE =====")

# --- Plot Kurva ROC (KNN Terbaik Tanpa SMOTE) ---
if best_y_pred_proba_no_smote is not None:
    plt.figure(figsize=(8, 6))
    RocCurveDisplay.from_predictions(y_test_binary, best_y_pred_proba_no_smote, name=f"KNN ROC (Tanpa SMOTE, n={best_n_no_smote})", ax=plt.gca())
    plt.title(f"Kurva ROC KNN (Tanpa SMOTE, n={best_n_no_smote})")
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.grid(True)
    plt.show()
```

6. Evaluasi *K-Nearest Neighbor*

```
# 2. KNN Dengan SMOTE dan Tuning n_neighbors
# ===== Melatih dan Mengevaluasi Model K-Nearest Neighbors (KNN) Dengan SMOTE =====
n_neighbors_values = [1, 5, 7, 9, 11]
best_accuracy = 0
best_n = 0
best_model_smote = None
best_y_pred_proba_smote = None
best_report = ""
best_cm = None
best_roc_auc = 0

for n in n_neighbors_values:
    print(f"--- Evaluasi KNN dengan n_neighbors = {n} (dengan SMOTE) ---")
    model_knn_smote = KNeighborsClassifier(n_neighbors=n)
    model_knn_smote.fit(X_train_smote, y_train_smote)

    y_pred_smote = model_knn_smote.predict(X_test_tfidf)
    y_pred_proba_smote = model_knn_smote.predict_proba(X_test_tfidf)[:, 1]

    current_accuracy = accuracy_score(y_test, y_pred_smote)
    current_report = classification_report(y_test, y_pred_smote)
    current_cm = confusion_matrix(y_test, y_pred_smote)
    current_roc_auc = roc_auc_score(y_test_binary, y_pred_proba_smote)

    print(f"Akurasi (n_neighbors={n}): {current_accuracy:.4f}")
    print("Classification Report:")
    print(current_report)
    print("Confusion Matrix:")
    print(current_cm)
    print(f"ROC AUC Score (n_neighbors={n}): {current_roc_auc:.4f}")

    if current_accuracy > best_accuracy:
        best_accuracy = current_accuracy
        best_n = n
        best_model_smote = model_knn_smote
        best_y_pred_proba_smote = y_pred_proba_smote
        best_report = current_report
        best_cm = current_cm
        best_roc_auc = current_roc_auc

print(f"===== Hasil Evaluasi KNN Dengan SMOTE =====")
print(f"KNN Terbaik dengan SMOTE adalah n_neighbors = {best_n}")
print(f"Akurasi Terbaik (dengan SMOTE): {best_accuracy:.4f}")
print("Classification Report Terbaik:")
print(best_report)
print("Confusion Matrix Terbaik:")
print(best_cm)
print(f"ROC AUC Score Terbaik (dengan SMOTE): {best_roc_auc:.4f}")
print(f"===== Hasil Evaluasi KNN Dengan SMOTE =====")

# --- Plot Kurva ROC (KNN Terbaik dengan SMOTE) ---
if best_y_pred_proba_smote is not None:
    plt.figure(figsize=(8, 6))
    RocCurveDisplay.from_predictions(y_test_binary, best_y_pred_proba_smote, name=f"KNN ROC (SMOTE, n={best_n})", ax=plt.gca())
    plt.title(f"Kurva ROC KNN (dengan SMOTE, n={best_n})")
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.grid(True)
    plt.show()
```

```
===== Melatih dan Mengevaluasi Model K-Nearest Neighbors (KNN) Tanpa SMOTE =====
Akurasi (n_neighbors=3): 0.8460
Classification Report:
      precision    recall  F1-score   support
0               1.00      0.89      0.95        371
1               0.20      0.64      0.40         85
-----
accuracy               0.85        456
macro avg             0.50      0.54      0.58        456
weighted avg          0.78      0.65      0.68        456

Confusion Matrix:
[[241 130]
 [ 23  42]]
ROC AUC Score (n_neighbors=3): 0.7117

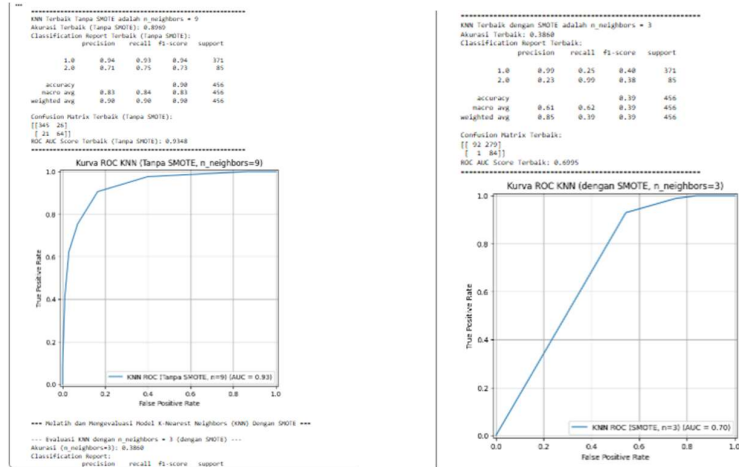
--- Evaluasi KNN dengan n_neighbors = 5 (Tanpa SMOTE) ---
Akurasi (n_neighbors=5): 0.8531
Classification Report:
      precision    recall  F1-score   support
0               1.00      0.91      0.88        371
1               0.18      0.73      0.35         85
-----
accuracy               0.85        456
macro avg             0.76      0.81      0.78        456
weighted avg          0.87      0.85      0.86        456

Confusion Matrix:
[[242 144]
 [ 23  42]]
ROC AUC Score (n_neighbors=5): 0.8404

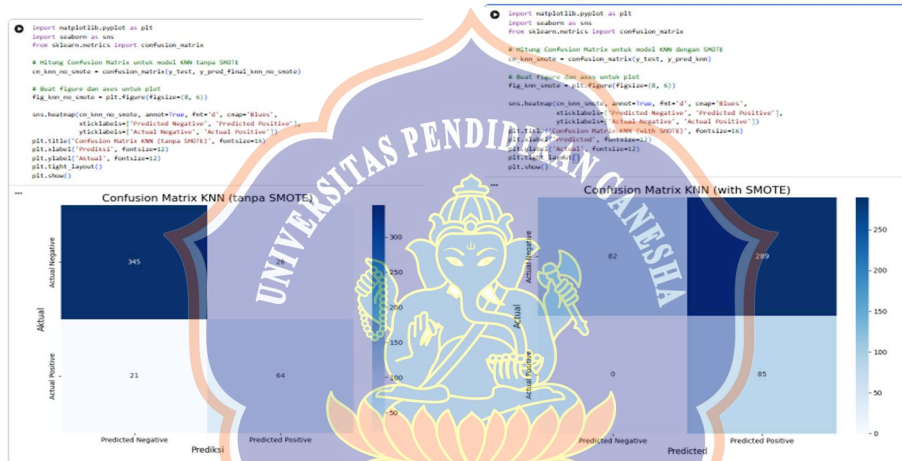
--- Evaluasi KNN dengan n_neighbors = 7 (Tanpa SMOTE) ---
Akurasi (n_neighbors=7): 0.8880
Classification Report:
      precision    recall  F1-score   support
0               1.00      0.94      0.92        371
1               0.18      0.74      0.35         85
-----
accuracy               0.88        456
macro avg             0.81      0.83      0.82        456
weighted avg          0.89      0.89      0.89        456

Confusion Matrix:
[[243 140]
 [ 22  43]]
ROC AUC Score (n_neighbors=7): 0.9047

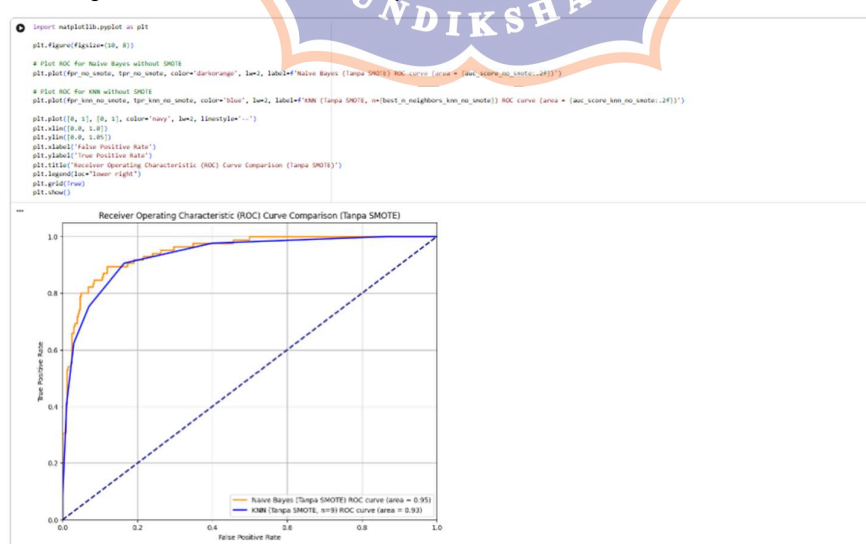
--- Evaluasi KNN dengan n_neighbors = 9 (Tanpa SMOTE) ---
Akurasi (n_neighbors=9): 0.8909
Classification Report:
```



7. Visualisasi Confusion Matrix dari K-Nearest Neighbor



8. Komparasi ROC-UAC Naïve Bayes dan KNN



```

from sklearn.metrics import roc_curve, auc, roc_auc_score
import matplotlib.pyplot as plt

# --- Calculate for Naive Bayes with SMOTE --- #
y_pred_proba_smote = model_naive_bayes_smote.predict_proba(X_test_tfidf)
positive_class_index = list(model_naive_bayes_smote.classes_).index(2.0)
y_prob_positive_class = y_pred_proba_smote[:, positive_class_index]

# Calculate False Positive Rate (FPR), True Positive Rate (TPR), and Thresholds for Naive Bayes with SMOTE
fpr, tpr, thresholds = roc_curve(y_test, y_prob_positive_class, pos_label=2.0)

# Calculate Area under the Curve (AUC) for Naive Bayes with SMOTE
roc_auc = auc(fpr, tpr)

print(f"AUC Score for Naive Bayes (with SMOTE): {roc_auc:.4f}")

# --- Calculate for KNN with SMOTE (using the best tuned model) ---
y_pred_proba_knn_smote = final_model_knn_smote.predict_proba(X_test_tfidf)

# Find the index of the position class (2.0) from the model's classes
positive_class_index_knn_smote = list(final_model_knn_smote.classes_).index(2.0)
y_prob_positive_class_knn_smote = y_pred_proba_knn_smote[:, positive_class_index_knn_smote]

# Calculate AUC score for the best KNN with SMOTE
auc_score_knn_smote = roc_auc_score(y_test, y_prob_positive_class_knn_smote)

# Calculate False Positive Rate (FPR), True Positive Rate (TPR), and Thresholds for KNN with SMOTE
fpr_knn_smote, tpr_knn_smote, thresholds_knn_smote = roc_curve(y_test, y_prob_positive_class_knn_smote, pos_label=2.0)

print(f"AUC Score for KNN (with SMOTE, n={best_n_neighbors_knn_smote}): {auc_score_knn_smote:.4f}")

plt.figure(figsize=(10, 8))

# Plot ROC for Naive Bayes with SMOTE
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'Naive Bayes (Dengan SMOTE) ROC curve (area = {roc_auc:.2f})')

# Plot ROC for KNN with SMOTE
plt.plot(fpr_knn_smote, tpr_knn_smote, color='green', lw=2, label=f'KNN (Dengan SMOTE, n={best_n_neighbors_knn_smote}) ROC curve (area = {auc_score_knn_smote:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve Comparison (Dengan SMOTE)')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()

```

```

--- AUC Score for Naive Bayes (with SMOTE): 0.9588
AUC Score for KNN (with SMOTE, n=3): 0.8095

```

