# Evaluasi Perbandingan Performa Model BERT dan Naïve Bayes pada Sentimen Analisis Pemilihan Presiden Indonesia 2024


**Kelompok 8**:

| | | |
|---|---|---|
| 1. | Gratia Yudika Morado Silalahi | 225150207111015 |
| 2. | Izzat Ikhwan Hadi | 225150200111010 |
| 3. | Muhammad Hasan Fadhlillah | 225150207111026 |
| 4. | Muhammad Husain Fadhlillah | 225150207111027 |

**Program Studi Teknik Informatika**
**Departemen Teknik Informatika**
**Fakultas Ilmu Komputer**
**Universitas Brawijaya**

**2024**

# DAFTAR ISI

# BAB 1 PENDAHULUAN

## 1.1 Deskripsi Studi Kasus

Di era digital yang semakin berkembang pesat, media sosial telah menjadi platform utama bagi masyarakat dalam menyuarakan opini dan pandangan mereka, termasuk dalam konteks politik. Twitter, sebagai salah satu platform media sosial terbesar, menjadi wadah diskusi signifikan terkait isu-isu politik, termasuk Pemilihan Presiden Indonesia 2024. Analisis sentimen terhadap tweet-tweet ini menjadi sangat penting untuk memahami persepsi publik, mengidentifikasi tren opini, dan memberikan wawasan berharga bagi berbagai pemangku kepentingan dalam proses demokrasi Indonesia.

Dalam studi kasus ini, kami mengembangkan sistem analisis sentimen otomatis yang mampu mengklasifikasikan tweet terkait kandidat presiden 2024 ke dalam kategori sentimen positif dan negatif. Sistem ini dirancang menggunakan pendekatan deep learning dengan model BERT (Bidirectional Encoder Representations from Transformers), yang telah terbukti efektif dalam tugas pemrosesan bahasa alami, khususnya analisis sentimen. Sebagai baseline, model tradisional Naive Bayes juga diimplementasikan untuk mengevaluasi keunggulan pendekatan modern dibandingkan metode konvensional.

### Rumusan Masalah

1. Bagaimana cara mengembangkan sistem analisis sentimen yang akurat untuk mengklasifikasikan sentimen masyarakat terhadap kandidat presiden 2024 di media sosial Twitter menggunakan model BERT?
2. Seberapa efektif perbandingan performa antara model deep learning BERT dengan model baseline Naive Bayes dalam analisis sentimen tweet?
3. Bagaimana karakteristik dan pola sentimen masyarakat terhadap masing-masing kandidat presiden berdasarkan hasil analisis sentimen menggunakan model yang dikembangkan?

### Tujuan Penelitian

1. Mengembangkan dan mengimplementasikan sistem analisis sentimen berbasis deep learning menggunakan model BERT untuk mengklasifikasikan tweet terkait kandidat presiden 2024.
2. Melakukan analisis komparatif performa antara model BERT dan model baseline Naive Bayes dalam tugas klasifikasi sentimen tweet politik.
3. Menganalisis dan menginterpretasikan pola sentimen masyarakat terhadap masing-masing kandidat presiden berdasarkan hasil klasifikasi model yang dikembangkan.

### Manfaat Penelitian

Penelitian ini memberikan beberapa manfaat signifikan:

- **Manfaat Akademis**:
    - Memberikan kontribusi pada pengembangan metode analisis sentimen, khususnya dalam konteks politik, dengan mengeksplorasi efektivitas model BERT.
    - Menyediakan benchmark perbandingan antara pendekatan deep learning modern dengan metode klasik dalam analisis sentimen.
    - Menghasilkan dataset tervalidasi yang dapat digunakan untuk penelitian lanjutan dalam bidang NLP.

- **Manfaat Praktis**:
    - Membantu tim kampanye dan strategi politik dalam memahami persepsi publik terhadap kandidat secara real-time dan objektif.
    - Memberikan wawasan bagi media dan analis politik dalam menginterpretasikan tren opini publik di media sosial.
    - Mendukung pengembangan sistem monitoring media sosial yang dapat digunakan untuk analisis sentimen dalam berbagai konteks politik di Indonesia.
- **Manfaat Sosial**:
    - Meningkatkan pemahaman tentang dinamika opini publik dalam konteks pemilihan presiden di era digital.
    - Mendorong transparansi dan keterbukaan dalam diskusi politik di media sosial.
    - Membantu masyarakat dan pemangku kepentingan dalam memahami polarisasi dan tren sentimen politik di media sosial.

**Data dan Karakteristiknya**

Dataset yang digunakan dalam penelitian ini berasal dari "Indonesia Presidential Candidates Dataset, 2024" yang tersedia di Kaggle dan Mendeley. Dataset ini memiliki karakteristik sebagai berikut:

1. **Sumber Data**
    - Platform: Twitter
    - Periode pengumpulan:
        - Ganjar Pranowo: Oktober 2022 - April 2023
        - Prabowo Subianto: Desember 2022 - April 2023
        - Anies Baswedan: Januari - April 2023
    - Jumlah total tweet: 30.000 (10.000 per kandidat)

2. **Cakupan Konten**

   Dataset mencakup berbagai aspek diskusi publik, seperti:
    - Visi dan misi kandidat
    - Track record dan pengalaman
    - Kebijakan yang diusung
    - Respon terhadap isu-isu nasional
    - Dukungan dari partai politik dan tokoh publik
    - Interaksi dengan masyarakat
    - Kampanye dan program kerja
    - Kritik dan tanggapan publik

3. **Nilai Penelitian**

   Dataset ini memiliki nilai penting untuk:
    - Pemahaman dinamika opini publik
    - Identifikasi isu-isu kunci yang mempengaruhi persepsi publik
    - Evaluasi efektivitas komunikasi politik
    - Prediksi tren dukungan publik
    - Studi komparatif pasca-pemilihan

**Struktur Dataset**

Dataset disimpan dalam format CSV dengan struktur yang kaya informasi:

1. **Kolom-kolom Utama:**
    - Date: Waktu tweet dibuat
    - Created: Waktu akun dibuat
    - User ID: Identifikasi unik pengguna
    - Followers: Jumlah pengikut akun
    - Following: Jumlah yang diikuti akun
    - Tweet Count: Jumlah tweet akun
    - TweetLocation: Lokasi pengguna (jika tersedia)
    - Text: Isi tweet
    - label: Kategori sentimen (positif/negatif)
    - Candidate: Nama kandidat yang dibahas

**Distribusi Dataset**

- Total Records: 30.000 tweet
- Distribusi per Kandidat: 10.000 tweet
- Distribusi Sentimen (setelah preprocessing):
    - Positif: ±73% (21.654 tweet)
    - Negatif: ±27% (8.074 tweet)

**Link Dataset**

Dataset dapat diakses melalui:

- Kaggle: https://www.kaggle.com/datasets/jocelyndumlao/indonesia-presidential-candidates-dataset-2024/
- Mendeley: https://data.mendeley.com/datasets/7w5zvr8jgp/5

**Paper Rujukan**

Korkmaz, A. Ç., 2023. Public's perception on nursing education during the COVID-19 pandemic: SENTIMENT analysis of Twitter data. *International Journal of Disaster Risk Reduction*, 99, p.104127. Available at: https://doi.org/10.1016/j.ijdrr.2023.104127 [Accessed 7 December 2024].

- Judul Paper: Public's perception on nursing education during the COVID-19 pandemic: SENTIMENT analysis of Twitter data
- Penulis: Ayşe Çiçek Korkmaz
- Jurnal: International Journal of Disaster Risk Reduction
- Volume: 99, Desember 2023
- URL: https://doi.org/10.1016/j.ijdrr.2023.104127

## 1.2 Metodologi Penyelesaian Studi Kasus

Metodologi ini dirancang untuk memberikan panduan sistematis dalam menyelesaikan studi kasus terkait analisis sentimen pada tweet politik, khususnya mengenai Pemilihan Presiden Indonesia 2024. Langkah-langkah yang dijabarkan mencakup instalasi lingkungan analisis, eksplorasi data, preprocessing, pembangunan model, hingga evaluasi kinerja. Dengan metodologi ini, diharapkan penelitian dapat dilakukan secara komprehensif dan menghasilkan sistem analisis sentimen yang akurat serta mampu memberikan wawasan berharga dari data yang dianalisis.

1. **Instalasi dan Persiapan Lingkungan**

- Menginstal pustaka dan dependencies yang dibutuhkan.
- Melakukan konfigurasi environment analisis.

2. **Import Library**
    - Pustaka untuk data manipulation: Digunakan untuk pengolahan data (contoh: pandas, numpy).
    - Pustaka untuk visualisasi: Digunakan untuk analisis visual (contoh: matplotlib, seaborn).
    - Pustaka untuk machine learning dan NLP: Digunakan untuk modeling dan analisis teks (contoh: sklearn, spaCy, transformers).
3. **Download dan Ekstraksi Data**
    - Mendownload dataset dari Google Drive.
    - Mengekstrak file dataset ke dalam folder kerja.
4. **Load Dataset**
    - Memuat tiga dataset kandidat presiden.
    - Memeriksa struktur data awal untuk memastikan integritas dataset.
5. **Kombinasi Dataset**
    - Menambahkan identitas kandidat ke dalam dataset.
    - Menggabungkan dataset menjadi satu kesatuan.
6. **Seleksi Kolom dan Penanganan Missing Value**
    - Memilih kolom yang relevan untuk analisis.
    - Membersihkan data kosong atau missing values.
7. **Eksplorasi Data Awal (EDA)**
    - Membuat visualisasi distribusi data.
    - Menganalisis panjang tweet dan pola temporal.
    - Menganalisis frekuensi kata untuk masing-masing kandidat.
8. **Data Processing dan Preprocessing Pipeline**
    - Membersihkan teks menggunakan cleantext.
    - Melakukan tokenisasi dan lemmatization menggunakan spaCy.
    - Menghapus emoji dari teks.
    - Melakukan normalisasi kata slang.
    - Membangun pipeline preprocessing yang terintegrasi.
9. **EDA Setelah Preprocessing**
    - Menampilkan preview data setelah preprocessing.
    - Membuat visualisasi frekuensi kata.
    - Membuat WordCloud berdasarkan kandidat dan sentimen.
10. **Data Sampling & Post-Processing**
    - Menganalisis distribusi label pada dataset.
    - Mengonversi label ke format biner.
    - Menangani ketidakseimbangan data dengan oversampling.
    - Membagi dataset ke dalam train, validation, dan test set.
    - Melakukan one-hot encoding pada label.
11. **Modelling dan Evaluasi Model**
    a. **Implementasi Baseline Model (Naive Bayes)**
        - Tokenisasi dengan CountVectorizer.
        - Mengonversi teks ke dalam representasi TF-IDF.
        - Melatih model Naive Bayes.
        - Melakukan prediksi dan menghitung probabilitas.
    b. **Evaluasi Model Naive Bayes**
        - Membuat evaluator untuk visualisasi evaluasi.
        - Menganalisis confusion matrix dan classification report.

- Membuat visualisasi ROC curve dan precision-recall curve.
- Menganalisis misclassifications untuk perbaikan model.

**c. Implementasi Model BERT**

- Menyiapkan tokenizer BERT pre-trained.
- Melakukan tokenisasi dataset dengan sequence length maksimum.
- Membuat model BERT pre-trained.
- Membangun arsitektur model dengan tambahan layer klasifikasi.
- Melatih model menggunakan dataset yang telah diproses.

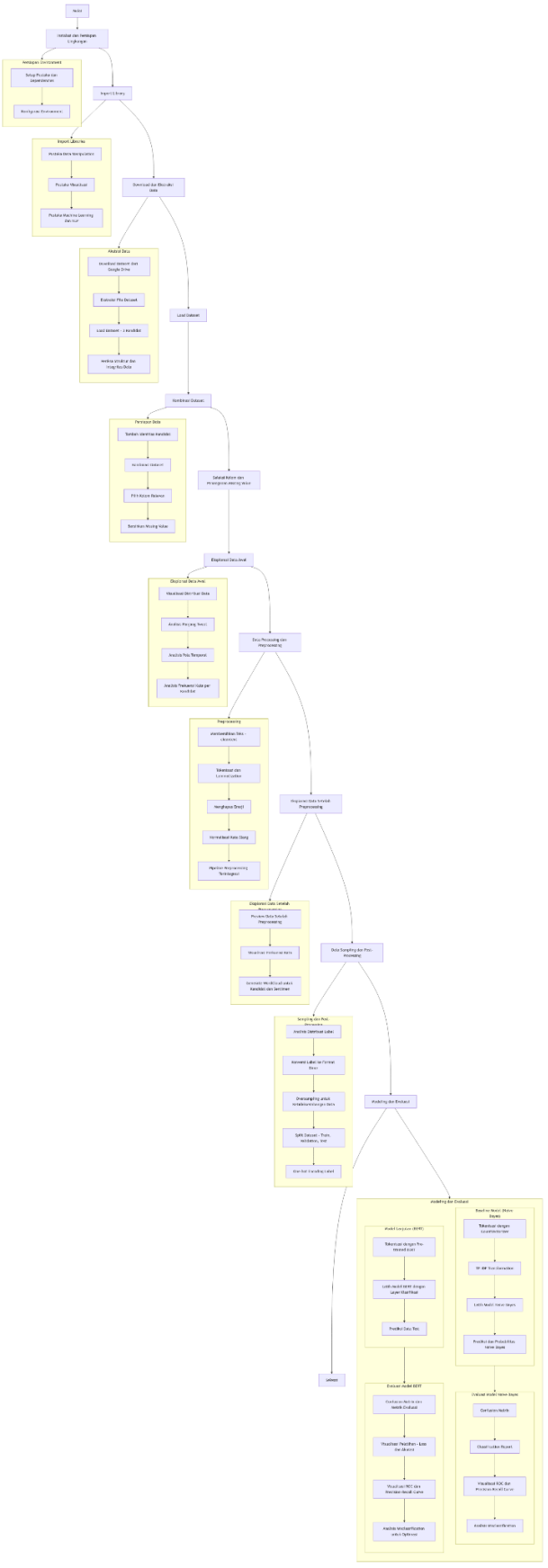
**d. Evaluasi Model BERT**

- Membuat prediksi pada data test.
- Memvisualisasikan history pelatihan (loss dan akurasi).
- Menganalisis confusion matrix dan metrik evaluasi.
- Membuat visualisasi ROC dan precision-recall curves.
- Menganalisis kesalahan klasifikasi untuk optimasi model.

Metodologi ini menawarkan pendekatan terstruktur yang mencakup seluruh tahapan dari persiapan data hingga evaluasi model. Dengan mengintegrasikan teknik eksplorasi data, preprocessing yang komprehensif, serta evaluasi mendalam terhadap model baseline dan model deep learning, metodologi ini diharapkan dapat menghasilkan sistem analisis sentimen yang optimal. Hasil dari penelitian ini tidak hanya berkontribusi dalam memahami persepsi publik tetapi juga memberikan solusi berbasis teknologi yang relevan untuk mendukung proses demokrasi di era digital.

# BAB 2 PERANCANGAN

## 2.1 Flowchart Penyelesaian Studi Kasus



*Link Drive Gambar Flowchart Penyelesaian Studi Kasus:*
*Flowchart Penyelesaian Studi Kasus.png*

Flowchart ini dirancang untuk memandu langkah-langkah analisis sentimen pada data Twitter, khususnya data terkait kandidat Pemilu Presiden Indonesia 2024. Dengan menggunakan teknologi Natural Language Processing (NLP), tujuan utama adalah memahami sentimen publik secara mendalam. Berikut penjelasan rinci setiap tahap:

**1. Persiapan dan Konfigurasi Lingkungan**
Tahap ini bertujuan untuk memastikan bahwa semua perangkat lunak dan pustaka yang diperlukan tersedia dan terkonfigurasi dengan baik.

- **Setup Pustaka dan Dependencies**: Instal pustaka Python seperti pandas, numpy, matplotlib, seaborn, scikit-learn, spaCy, NLTK, dan transformers. Langkah ini memastikan semua tools yang diperlukan untuk analisis tersedia.
- **Konfigurasi Environment**: Mempersiapkan lingkungan kerja seperti Jupyter Notebook atau IDE, termasuk pengaturan variabel lingkungan dan inisialisasi GPU jika diperlukan untuk model deep learning.

**Representasi Flowchart:** *Mulai → Instal Pustaka → Konfigurasi Lingkungan → Selesai Setup*

**2. Akuisisi Dataset**
Dataset yang akan digunakan diunduh dari sumber eksternal, misalnya Google Drive atau API Twitter. Langkah-langkah utamanya adalah:

- **Download Dataset**: Mengunduh dataset dari sumber yang sudah ditentukan.
- **Ekstraksi File Dataset**: Jika dataset dalam format terkompresi, file diekstrak ke direktori kerja.
- **Load Dataset**: Dataset dimuat ke dalam DataFrame menggunakan pustaka seperti pandas.
- **Periksa Struktur dan Integritas**: Memastikan dataset tidak memiliki anomali seperti format yang tidak konsisten atau data yang rusak.

**Representasi Flowchart:** *Download Dataset → Ekstraksi File → Load Dataset → Periksa Integritas*

**3. Kombinasi dan Pembersihan Data**
Setelah dataset berhasil diunduh dan dimuat, tahap berikutnya adalah menggabungkan data dari berbagai sumber serta membersihkannya.

- **Penambahan Identitas Kandidat**: Menambahkan label untuk membedakan data setiap kandidat.
- **Penggabungan Dataset**: Menggabungkan dataset menjadi satu kesatuan untuk analisis terpadu.
- **Seleksi Kolom**: Memilih kolom yang relevan, seperti teks tweet, tanggal, dan sentimen.
- **Penanganan Missing Value**: Menangani nilai kosong dengan strategi yang sesuai, misalnya interpolasi atau penghapusan baris.

**Representasi Flowchart:** *Labeling Data → Gabungkan Dataset → Seleksi Kolom → Tangani Missing Value*

**4. Eksplorasi Data Awal (EDA)**
Sebelum data diolah lebih lanjut, dilakukan analisis deskriptif untuk memahami pola dasar data.

- **Visualisasi Distribusi Data**: Menampilkan distribusi jumlah tweet per kandidat.
- **Analisis Panjang Tweet**: Mengamati panjang teks untuk menentukan preprocessing yang sesuai.
- **Analisis Pola Temporal**: Memahami pola tweet berdasarkan waktu.
- **Frekuensi Kata**: Mengidentifikasi kata-kata yang sering digunakan.

**Representasi Flowchart:** *EDA → Visualisasi Distribusi → Analisis Temporal → Analisis Frekuensi Kata*

**5. Preprocessing Data**
Tahap inti analisis NLP di mana teks diubah menjadi format yang siap untuk analisis lebih lanjut.

- **Membersihkan Teks**: Menghapus URL, tanda baca, emoji, dan elemen lain yang tidak relevan.
- **Tokenisasi dan Lemmatization**: Memecah teks menjadi token dan menyederhanakan kata ke bentuk dasarnya.
- **Penghapusan Stopwords**: Menghapus kata-kata umum yang tidak memiliki nilai analisis, seperti "dan" atau "atau".
- **Normalisasi Kata Slang**: Mengonversi kata tidak baku menjadi baku.
- **Pipeline Preprocessing**: Mengintegrasikan semua langkah preprocessing dalam pipeline otomatis.

**Representasi Flowchart:** *Cleaning → Tokenizing → Lemmatization → Stopword Removal → Normalisasi*

**6. Pembagian Data**
Setelah preprocessing, data dibagi menjadi subset untuk pelatihan, validasi, dan pengujian model.

- **Split Dataset**: Membagi data menjadi train, validation, dan test set dengan rasio standar (70:20:10).
- **Analisis Distribusi Label**: Memastikan distribusi label sentimen seimbang.
- **Oversampling**: Mengatasi ketidakseimbangan data menggunakan teknik seperti SMOTE.
- **One-hot Encoding Label**: Mengonversi label sentimen ke format numerik.

**Representasi Flowchart:** *Split Data → Balance Label (Oversampling) → Final Data Prep*

**7. Pemodelan dan Evaluasi**

Tahap ini melibatkan pembuatan dan evaluasi model analisis sentimen menggunakan pendekatan berikut:

- **Baseline Model**: Menggunakan metode sederhana seperti Naive Bayes dengan fitur TF-IDF sebagai pembanding awal.
- **Advanced Model**: Menggunakan model berbasis deep learning seperti BERT untuk hasil yang lebih akurat.
- **Evaluasi Model**: Mengukur performa model menggunakan metrik seperti akurasi, precision, recall, F1-score, dan ROC-AUC.

**Representasi Flowchart:** *Train Baseline Model -> Train Advanced Model -> Evaluasi Model -> Analisis Hasil*

**8. Iterasi dan Finalisasi**

Jika hasil model tidak memuaskan, iterasi dilakukan dengan mengulang beberapa tahap sebelumnya.

- **Revisi Preprocessing atau Model**: Memperbaiki preprocessing atau mencoba model lain.
- **Evaluasi Hasil Akhir**: Menentukan hasil terbaik untuk digunakan.

**Representasi Flowchart:** *Evaluasi Hasil → Revisi Preprocessing/Model → Selesai*

Flowchart ini mencakup langkah-langkah sistematis dari persiapan hingga evaluasi model. Setiap tahap dirancang untuk memastikan bahwa proses analisis berjalan efektif dan hasil yang diperoleh dapat diandalkan. Dengan pendekatan fleksibel ini, analisis sentimen dapat memberikan wawasan mendalam tentang opini publik terkait Pemilu Presiden Indonesia 2024.

# BAB 3 IMPLEMENTASI

## 3.1 Tautan *Google Colab*

*Project NLP B - Sentiment Analysis Kandidat Presiden Indonesia 2024 #v2*

## 3.2 Source Code

### 3.2.1 Import Library dan Modul yang diperlukan

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.util import ngrams
from collections import Counter
from wordcloud import WordCloud
import plotly.express as px
import plotly.graph_objects as go
from sklearn.feature_extraction.text import
CountVectorizer, TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import (
    accuracy_score, f1_score, classification_report,
confusion_matrix,
    roc_curve, auc, roc_auc_score,
precision_recall_curve, average_precision_score,
    precision_score, recall_score
)
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize
from sklearn.utils import shuffle
from imblearn.over_sampling import RandomOverSampler,
SMOTE
from transformers import BertTokenizerFast,
TFBertModel
import gdown
import zipfile
from itertools import cycle
import warnings
```

```
# Suppress warnings
warnings.filterwarnings('ignore')
```

Pada langkah awal dimulai dengan memuat pustaka dan modul yang relevan untuk pelaksanaan analisis teks dan pembelajaran mesin. Pada tahap ini, pustaka meliputi:

1. Pengolahan teks menggunakan pustaka seperti nltk dan transformers untuk proses tokenisasi, lemmatization, dan penghilangan noise pada teks mentah, yang penting untuk meningkatkan kualitas representasi teks.
2. Visualisasi alat seperti matplotlib, seaborn, dan plotly mendukung analisis data dengan visualisasi yang membantu memahami pola dan distribusi data.
3. Sklearn menyediakan berbagai alat untuk membangun pipeline klasifikasi, evaluasi kinerja model, dan eksplorasi matriks kebingungan.
4. Tensorflow digunakan untuk membangun arsitektur model berbasis neural network, termasuk implementasi transfer learning menggunakan BERT.
5. Penanganan ketidakseimbangan data seperti RandomOverSampler mendukung pengolahan data yang tidak seimbang, yang berkontribusi pada pengurangan bias dalam pelatihan model.

### 3.2.2 Download dan Ekstraksi Data

```
# ID file Google Drive
file_id = '16lB8ihk--e5Q1bBkVqIeHn2q_R-E-Bzd'
# URL untuk mendownload file
url = f'https://drive.google.com/uc?id={file_id}'
# Nama file output
output = 'dataset.zip'

# Download file
gdown.download(url, output, quiet=False)

# Ekstraksi file
with zipfile.ZipFile('dataset.zip', 'r') as
zip_ref:
    zip_ref.extractall('dataset_folder')
```

Proses ini dilakukan pengambilan data eksternal secara programatik untuk mempermudah aksesibilitas dan pengelolaan dataset. Dengan menggunakan library *gdown*, file diunduh langsung dari Google Drive menggunakan ID uniknya, kemudian disimpan sebagai file output lokal. Proses dekompresi data menggunakan library zipfile. File ZIP yang telah diunduh sebelumnya diekstrak ke dalam folder tujuan (dataset_folder).

### 3.2.3 Load Dataset

```
# Read dataset
anies_data = pd.read_csv('dataset_folder/Indonesia
Presidential Candidates Dataset, 2024/labeled
data/Anies Baswedan.csv')
prabowo_data = pd.read_csv('dataset_folder/Indonesia
Presidential Candidates Dataset, 2024/labeled
data/Prabowo Subianto.csv')
ganjar_data = pd.read_csv('dataset_folder/Indonesia
Presidential Candidates Dataset, 2024/labeled
data/Ganjar Pranowo.csv')

# Informasi dataset kandidat anies
anies_data.info()

# Informasi dataset kandidat prabowo
prabowo_data.info()

# Informasi dataset kandidat ganjar
ganjar_data.info()
```

Dataset memuat masing-masing kandidat presiden dan wakil presiden, yaitu Anies Baswedan, Prabowo Subianto, dan Ganjar Pranowo. File CSV yang disimpan dalam direktori hasil ekstraksi dibaca ke dalam tiga dataframe menggunakan pandas.

Pemaparan informasi mengenai deskripsi struktural setiap dataset, meliputi jumlah kolom, tipe data pada setiap kolom, serta jumlah data yang ada. Informasi ini digunakan untuk memahami komposisi dataset dan membantu mendeteksi potensi anomali seperti nilai kosong atau tipe data yang tidak sesuai.

### 3.2.4 Kombinasi Dataset

```
# Tambahkan kolom kandidat
anies_data['Candidate'] = 'Anies Baswedan'
prabowo_data['Candidate'] = 'Prabowo Subianto'
ganjar_data['Candidate'] = 'Ganjar Pranowo'

# Menggabungkan dataset
df = pd.concat([anies_data, prabowo_data,
ganjar_data])

# Informasi dataset setelah digabung
df.head()
```

Penggabungan dataset dilakukan dengan menambahkan atribut yang mengindikasikan kandidat pada setiap dataset individual. Penambahan kolom Candidate memungkinkan identifikasi sumber data setelah penggabungan.

### 3.2.5 Seleksi Kolom dan Penanganan Missing Value

```
# Melihat ukuran dataset
df.shape()

# Melihat missing value dalam dataset
df.isnull().sum()

# Seleksi kolom yang diperlukan
df = df.loc[:, [' Tweet Count', 'Text', 'label',
'Candidate', ' Date']]

# Hapus data kosong
df = df.dropna()

# Melihat kembali dataset setelah dilakukan seleksi
kolom
df.head()

# Melihat kembali struktur dataset
df.info()

# Melihat statistik deskriptif
df.describe()

# Mengecek kembali missing value dalam dataset
df.isnull().sum()

# Melihat kembali ukuran dataset
df.shape()
```

Pemeriksaan terhadap ukuran dan struktur dataset untuk mendapatkan gambaran umum. Kemudian, diperiksa apakah ada data yang hilang dan dilakukan pembersihan dengan menghapus data kosong. Selanjutnya, hanya kolom-kolom yang relevan yang dipilih untuk fokus analisis. Statistik deskriptif digunakan untuk memahami karakteristik umum data, dan terakhir, dataset diperiksa kembali untuk memastikan tidak ada data yang hilang serta untuk memastikan ukuran dataset sesuai dengan yang diinginkan.

### 3.2.6 Exploratory Data Analysis (EDA)

1. Distribusi Jumlah Tweet per Kandidat

```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Candidate',
palette="viridis")
plt.title("Distribusi Jumlah Tweet per Kandidat")
plt.show()
```

```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='label',
palette="coolwarm")
plt.title("Distribusi Sentimen Secara Keseluruhan")
plt.show()

plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='label',
palette="coolwarm")
plt.title("Distribusi Sentimen Secara Keseluruhan")
plt.show()
```

Visualisasi distribusi jumlah tweet per kandidat dan distribusi sentimen dilakukan untuk memberikan gambaran tentang sebaran data. Pertama, visualisasi jumlah tweet yang dibuat untuk masing-masing kandidat ditampilkan untuk melihat perbandingan antara Anies Baswedan, Prabowo Subianto, dan Ganjar Pranowo. Kemudian, distribusi sentimen secara keseluruhan divisualisasikan untuk memahami bagaimana persebaran sentimen positif, negatif, atau netral dalam dataset.

2. Analisis Panjang Tweet

```
df['Tweet Length'] = df['Text'].apply(lambda x:
len(x.split()))

plt.figure(figsize=(10, 6))
sns.histplot(df['Tweet Length'], bins=30, kde=True,
color='blue')
plt.title("Distribusi Panjang Tweet")
plt.xlabel("Jumlah Kata")
plt.show()

plt.figure(figsize=(8, 6))
sns.boxplot(x='label', y='Tweet Length', data=df,
palette="Set2")
plt.title("Panjang Tweet Berdasarkan Sentimen")
plt.show()
```

Visualisasi pertama menampilkan distribusi panjang tweet dengan histogram, yang membantu untuk melihat sebaran jumlah kata dalam tweet secara keseluruhan. Dengan histogram ini, kita dapat memahami apakah panjang tweet terdistribusi merata atau ada kecenderungan tertentu, misalnya banyak tweet yang sangat pendek atau panjang.

Visualisasi kedua menggunakan boxplot untuk menggambarkan panjang tweet berdasarkan sentimen. Boxplot ini memberikan gambaran tentang bagaimana panjang tweet berbeda antara kategori sentimen, serta mengidentifikasi apakah ada outlier yang signifikan dalam panjang tweet untuk setiap kategori sentimen.

3. Analisis Temporal

```
df[' Date'] = pd.to_datetime(df[' Date'],
errors='coerce')

plt.figure(figsize=(12, 6))
grouped_df = df.groupby([' Date', 'Candidate'])['
Tweet Count'].sum().reset_index()
sns.lineplot(data=grouped_df, x=' Date', y=' Tweet
Count', hue='Candidate')
plt.title("Volume Tweet dari Waktu ke Waktu")
plt.show()
```

Visualisasi untuk menunjukkan perubahan volume tweet dari waktu ke waktu untuk masing-masing kandidat. Dengan menggunakan lineplot untuk memantau bagaimana jumlah tweet yang terkait dengan setiap kandidat berkembang sepanjang periode waktu yang tercatat.

4. Frekuensi Kata Teratas

```
def plot_word_frequencies(data, label, candidate,
top_n=20):
    subset = data[(data['label'] == label) &
(data['Candidate'] == candidate)]
    if subset.empty:
        print(f"No data available for {candidate} -
{label}")
        return

    vectorizer =
CountVectorizer(stop_words='english',
max_features=top_n)
    counts =
vectorizer.fit_transform(subset['Text'])
    word_freq = pd.DataFrame({
        'word': vectorizer.get_feature_names_out(),
        'count': counts.toarray().sum(axis=0)
    }).sort_values('count', ascending=False)

    plt.figure(figsize=(10, 6))
    sns.barplot(data=word_freq, x='count', y='word',
palette="viridis")
    plt.title(f"Top {top_n} Words in {label} Tweets
- {candidate}")
    plt.show()

# Iterasi melalui setiap kandidat dan sentimen
candidates = df['Candidate'].unique()
sentiments = df['label'].unique()
```

```
for candidate in candidates:
    for sentiment in sentiments:
        plot_word_frequencies(df, sentiment,
candidate)
```

Visualisasi kata-kata yang paling sering muncul dalam tweet berdasarkan kategori sentimen dan kandidat. Untuk setiap kombinasi kandidat dan sentimen, dataset disaring sesuai dengan label sentimen dan kandidat yang dipilih. Kemudian, dilakukan penghitungan frekuensi kata menggunakan teknik CountVectorizer, yang hanya mempertimbangkan kata-kata yang paling sering muncul (top_n) dan mengabaikan kata-kata umum seperti stop words. Hasil frekuensi kata ini ditampilkan dalam bentuk grafik batang untuk memperlihatkan kata-kata yang paling dominan pada setiap kombinasi sentimen dan kandidat.

Proses ini dilakukan untuk setiap kombinasi kandidat dan sentimen, memberikan gambaran mengenai kata-kata yang sering digunakan dalam tweet terkait masing-masing kategori.

5. WordCloud

```
def plot_wordcloud(data, title):
    text = ' '.join(data)
    wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(text)
    plt.figure(figsize=(10, 6))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(title, fontsize=16)
    plt.show()

# Buat WordCloud per kandidat dan sentimen
candidates = df['Candidate'].unique()
sentiments = df['label'].unique()

for candidate in candidates:
    for sentiment in sentiments:
        subset = df[(df['Candidate'] == candidate) &
(df['label'] == sentiment)]
        title = f"WordCloud for {sentiment} Tweets -
{candidate}"
        plot_wordcloud(subset['Text'], title)
        print()
```

Fungsi ini digunakan untuk membuat visualisasi word cloud, yang memperlihatkan kata-kata yang paling sering muncul dalam tweet untuk setiap kombinasi kandidat dan sentimen. Pertama, data tweet yang sesuai dengan kandidat dan sentimen tertentu digabungkan menjadi satu teks panjang. Kemudian, teks tersebut diproses untuk menghasilkan word cloud, di mana kata-kata yang lebih sering muncul akan terlihat

lebih besar. Word cloud ini memberi gambaran visual yang mudah dipahami tentang kata-kata dominan yang digunakan dalam tweet untuk masing-masing kategori.

6. Analisis Kata dengan Batasan

```
def cari_dan_hitung_kata_dengan_batasan(df,
daftar_kata):
    def temukan_kata(text, daftar_kata):
        kata_ditemukan = []
        for kata in daftar_kata:
            pola = r'\b' + re.escape(kata) + r'\b'
            if re.search(pola, str(text),
re.IGNORECASE):
                kata_ditemukan.append(kata)
        return kata_ditemukan

    # Flatten list of lists dari semua baris
    all_words = [word for sublist in
df['Text'].apply(lambda x: temukan_kata(x,
daftar_kata)) for word in sublist]

    # Hitung kemunculan setiap kata
    word_counts = Counter(all_words)

    return word_counts

#
daftar_kata_slang = ['mr', 'dude', 'gue', 'gw',
'kece', 'anjay', 'anjir', 'banget', 'wkwk', 'cuek',
'mager', 'alay', 'gokil', 'kepo', 'jamet', 'pede',
'lebay', 'santai', 'gebuk', 'nongki', 'sok',
'bokap', 'nyokap', 'gila', 'kampret', 'ngeselin',
'baper', 'galau', 'KD', 'bucin', 'nangis', 'coi',
'yaudah', 'nih', 'sih', 'doang', 'ciye', 'woy',
'ngehe', 'u', 'r', 'idk', 'brb', 'smh', 'lol',
'omg', 'btw', 'fyi', 'thx', 'pls', 'plz', 'wtf',
'stfu', 'rofl', 'afk', 'asap', 'rn', 'imo', 'tbh',
'ikr', 'ngl', 'jk', 'yolo', 'ftw', 'tbf', 'yeet',
'sus', 'slay', 'stan', 'ghosting', 'simp', 'cap',
'cringe', 'lit', 'lowkey', 'highkey', 'gpp', 'bgt',
'kk', 'lu', 'bro', 'gw', 'deh', 'gt', 'yg', 'hrs',
'skrg', 'bwt', 'utk', 'dgn', 'w', 'y', 'k', 'ga',
'gk', 'emang', 'lah', 'kan', 'weh', 'gitu', 'oke',
'bro', 'coy', 'mantap', 'sumpah', 'asli', 'kawan',
'bang', 'bosque', 'udah', 'gede', 'bgt', 'jiwa',
'cekrek', 'hehe', 'huhu', 'rese', 'cemeng',
'gasken', 'puyeng', 'bokis', 'cazang', 'njir',
'mampus', 'biadab', 'goblok', 'tolol', 'sinting',
'brengsek', 'tai', 'goblog', 'yak', 'keren', 'deh',
'dong', 'kali', 'nya', 'tuh', 'baru', 'abis',
```

```
'buat', 'sama', 'duit', 'maap', 'makasih', 'oke',
'sip', 'wkw', 'mksdnya', 'cuman', 'kyknya',
'mungkin', 'soale', 'pdkt', 'cwk', 'cwk', 'jomblo',
'wibu', 'swag', 'yo', 'bruh', 'fam', 'deadass',
'lit', 'salty', 'oof', 'rip', 'legit', 'noob',
'hmm', 'meh', 'omw', 'rn', 'high-key', 'sis',
'fomo', 'yada', 'waduh', 'astaga', 'gak nyangka',
'ya elah', 'bete', 'sebel', 'mumet', 'anjrit',
'yahaha', 'wkwkwk', 'btw aja', 'yaelah',
'astagfirullah', 'subhanallah', 'masyaallah',
'guys', 'sis', 'bestie', 'mhm', 'hmph', 'uwu',
'owo', 'nye', 'trs', 'bru', 'ajg', 'anj', 'asu',
'jir', 'bego', 'deh', 'bener', 'ya ampun', 'aduh',
'woi', 'lu']
hasil_hitung =
cari_dan_hitung_kata_dengan_batasan(df,
daftar_kata_slang)

# Tampilkan hasil
for kata, jumlah in hasil_hitung.items():
    print(f"'{kata}'")
```

Fungsi ini mencari kata-kata yang terdapat dalam daftar kata (misalnya slang atau istilah populer) pada setiap tweet, kemudian menghitung berapa kali masing-masing kata muncul. Proses pencarian dilakukan dengan mencocokkan kata-kata tersebut secara sensitif terhadap huruf besar/kecil.

### 3.2.7 Data Preprocessing

```
## Import library yang dibutuhkan kembali
from cleantext import clean
import spacy
import emoji

def preprocess_text_with_cleantext(text):
    return clean(
        text,
        lower=True,                    # Konversi ke
huruf kecil
        no_line_breaks=True,           # Hapus \n dan
\r
        no_urls=True,                  # Hapus URL
        no_emails=True,                # Hapus email
        no_phone_numbers=True,         # Hapus nomor
telepon
        no_numbers=False,              # Tidak hapus
angka (opsional)
        no_punct=True,                 # Hapus tanda
baca
```

```
        no_emoji=True,                  # Hapus emoji
        replace_with_punct="",          # Tidak ganti
tanda baca
        replace_with_url="",            # Ganti URL
dengan string kosong
        replace_with_email="",          # Ganti email
dengan string kosong
    )

nlp = spacy.load("en_core_web_sm")  # Model bahasa
Inggris
def preprocess_with_spacy(text):
    doc = nlp(text.lower())
    tokens = [
        token.lemma_  # Lemmatization untuk
mendapatkan kata dasar
        for token in doc
        if not token.is_stop          # Hapus
stopword
        and not token.is_punct        # Hapus tanda
baca
        and not token.is_space        # Hapus spasi
tambahan
        and not token.like_url        # Hapus URL
        and not token.like_email      # Hapus email
    ]
    return " ".join(tokens)

def remove_emoji(text):
    return emoji.replace_emoji(text, replace="")

slang_dict = {
    "mr": "mister",
    "bang": "brother",
    "guys": "friends",
    "bro": "brother",
    "jk": "just kidding",
    "lol": "laughing out loud",
    "sama": "",
    "btw": "by the way",
    "hehe": "",
    "astagfirullah": "",
    "based": "",
    "masyaallah": "",
    "goblog": "",
    "yo": "hey",
    "gt": "got to",
    "subhanallah": "",
    "sis": "sister",
    "sip": "okay",
```

```
    "mumet": "",
    "ngehe": "",
    "lah": "",
    "r": "are",
    "kk": "sister",
    "cap": "",
    "coy": "",
    "nya": "",
    "nye": "",
    "pls": "please",
    "gede": "big",
    "weh": "",
    "ngl": "not gonna lie",
    "hmm": "",
    "meh": "",
    "bestie": "best friend",
    "lu": "you",
    "fyi": "for your information",
    "ga": "not",
    "kali": "maybe",
    "nih": "",
    "gk": "not",
    "tbh": "to be honest",
    "anjay": "",
    "kampret": "",
    "rip": "rest in peace",
    "dude": "man",
    "trs": "then",
    "gokil": "",
    "gpp": "no problem",
    "fomo": "fear of missing out",
    "gue": "i",
    "yak": "",
    "dong": "",
    "asli": "really",
    "baru": "new",
    "omg": "oh my god",
    "tai": "",
    "deh": "",
    "baper": "",
    "njir": "",
    "kece": "cool"
}

def normalize_slang(text):
    return " ".join([slang_dict.get(word, word) for
word in text.split()])

def full_preprocessing_pipeline(text):
    # Langkah 1: Clean text menggunakan cleantext
```

```
    text = preprocess_text_with_cleantext(text)
    # Langkah 2: Hapus emoji
    text = remove_emoji(text)
    # Langkah 3: Normalisasi slang
    text = normalize_slang(text)
    # Langkah 4: Tokenisasi, penghapusan stopwords,
dan lemmatization
    text = preprocess_with_spacy(text)
    return text

df['clean_text'] =
df['Text'].apply(full_preprocessing_pipeline)
```

Proses preprocessing yang dilakukan pada teks dalam dataset:

1. Pembersihan Awal dengan Cleantext
   Bertujuan untuk membersihkan teks dari elemen-elemen yang tidak relevan. Ini mencakup konversi teks menjadi huruf kecil, penghapusan baris baru, URL, email, nomor telepon, angka, tanda baca, dan emoji. Teks yang bersih akan memudahkan analisis lebih lanjut karena elemen-elemen tersebut bisa menambah kebisingan.

2. Penghapusan Emoji
   Emoji yang mungkin masih ada setelah pembersihan pertama dihapus dengan menggunakan pustaka emoji. Emoji dapat membawa informasi yang tidak relevan atau dapat menambah kompleksitas dalam analisis, sehingga penghapusannya diperlukan.

3. Normalisasi Slang
   Proses ini menggantikan istilah slang dengan bentuk yang lebih baku sesuai dengan kamus slang yang telah disiapkan. Misalnya, "mr" menjadi "mister", "lol" menjadi "laughing out loud". Hal ini dilakukan untuk mengurangi variasi kata yang sejenis dan membuat teks lebih konsisten, memudahkan analisis lebih lanjut.

4. Tokenisasi, Penghapusan Stopwords, dan Lematisasi dengan SpaCy
   Teks kemudian diproses menggunakan SpaCy, di mana teks pertama kali diturunkan ke bentuk huruf kecil. Kemudian dilakukan lemmatization untuk mendapatkan bentuk kata dasar dari setiap token (misalnya, "running" menjadi "run").
   Selama proses ini, stopwords (kata-kata umum seperti "dan", "di", yang tidak memberikan informasi penting) dan tanda baca dihapus. URL, email, dan spasi tambahan juga dibersihkan.

### 3.2.8 Exploratory Data Analysis (EDA) After Preprocessing

```
# Menampilkan 5 baris pertama dari data yang telah
diproses
df.head()
```

```
for candidate in candidates:
    for sentiment in sentiments:
        plot_word_frequencies(df, sentiment,
candidate)

for candidate in candidates:
    for sentiment in sentiments:
        subset = df[(df['Candidate'] == candidate) &
(df['label'] == sentiment)]
        title = f"WordCloud for {sentiment} Tweets -
{candidate}"
        plot_wordcloud(subset['Text'], title)
        print()
```

Menampilkan 5 baris pertama dari data yang telah diproses untuk memastikan bahwa proses pembersihan dan normalisasi berhasil dilakukan dengan baik. visualisasi frekuensi kata yang paling sering muncul dalam tweet untuk setiap kombinasi kandidat dan sentimen. Visualisasi ini membantu untuk memahami kata-kata yang paling sering digunakan oleh publik dalam diskusi tentang masing-masing kandidat, baik dalam sentimen positif maupun negatif. Selanjutnya, dilakukan pembuatan word cloud untuk setiap kombinasi kandidat dan sentimen.

### 3.2.9 Data Sampling

1. Exploratory Analysis of Label Distribution

```
# Mengecek distribusi awal label sentimen
df['label'].value_counts()
```

Pada tahap ini, kita menganalisis distribusi label sentimen (Positive dan Negative). Informasi ini penting untuk memahami keseimbangan dataset dan menentukan strategi penyeimbangan data yang diperlukan.

2. Label Mapping

```
print("Sebelum Mapping:")
print(df['label'].head())

# Mengonversi label 'Positive' menjadi 1 dan
'Negative' menjadi 0
df['label'] = df['label'].map({'Positive':1,
'Negative':0})

print("Setelah Mapping:")
print(df['label'].head())
```

Label Positive diubah menjadi 1 dan Negative menjadi 0 untuk mempermudah proses pemodelan. Pendekatan biner ini diperlukan untuk algoritma pembelajaran mesin yang membutuhkan representasi numerik.

3. Addressing Class Imbalance with Oversampling

```
# Mengecek kembali distribusi label setelah mapping
print(df['label'].value_counts())

from sklearn import preprocessing
from imblearn.over_sampling import RandomOverSampler

# Oversampling data
ros = RandomOverSampler()
train_x, train_y =
ros.fit_resample(np.array(df['clean_text']).reshape(
-1, 1), np.array(df['label']).reshape(-1, 1));
train_os = pd.DataFrame(list(zip([x[0] for x in
train_x], train_y)), columns = ['clean_text',
'label']);

# Mengecek distribusi label setelah oversampling
print(train_os['label'].value_counts())
```

Distribusi label yang tidak seimbang dapat menyebabkan model bias terhadap kelas mayoritas. Untuk mengatasi ini, kita akan menggunakan teknik Random Oversampling. Random oversampling menambahkan salinan dari sampel kelas minoritas hingga distribusinya seimbang dengan kelas mayoritas. Dataset hasil oversampling disimpan dalam variabel train_os.

4. Splitting Dataset: Train, Validation, and Test Sets

```
from sklearn.model_selection import train_test_split

# Memisahkan data menjadi train, validation, dan
test set
X = train_os.drop(columns=['label']).values
y = train_os['label'].values

X_train, X_temp, y_train, y_temp =
train_test_split(X, y, test_size=0.4,
random_state=42)
X_valid, X_test, y_valid, y_test =
train_test_split(X_temp, y_temp, test_size=0.5,
random_state=42)

# Menyimpan salinan label untuk referensi di masa
mendatang
y_train_le = y_train.copy()
y_valid_le = y_valid.copy()
y_test_le = y_test.copy()
```

```
# One-Hot Encoding label
from sklearn.preprocessing import OneHotEncoder
ohe = preprocessing.OneHotEncoder()

y_train =
ohe.fit_transform(np.array(y_train).reshape(-1,
1)).toarray()
y_valid =
ohe.fit_transform(np.array(y_valid).reshape(-1,
1)).toarray()
y_test=
ohe.fit_transform(np.array(y_test).reshape(-1,
1)).toarray()
```

Dataset dibagi menjadi:

- Training Set (60%): Untuk melatih model.
- Validation Set (20%): Untuk menyetel hyperparameter model.
- Test Set (20%): Untuk mengevaluasi performa akhir model.

One-hot encoding mengubah label biner menjadi format vektor untuk meningkatkan kompatibilitas dengan beberapa algoritma pembelajaran mesin dan deep learning.

5. Final Dataset Summary

```
# Melihat ringkasan jumlah data di setiap set
print(f"TRAINING DATA:
{X_train.shape[0]}\nVALIDATION DATA:
{X_valid.shape[0]}\nTESTING DATA: {X_test.shape[0]}"
)
```

Hasil akhir dari pembagian dataset adalah:

- Training Data: 25984 sampel
- Validation Data: 8662 sampel
- Testing Data: 8662 sampel

Setiap set digunakan untuk keperluan yang berbeda seperti pelatihan, validasi, dan evaluasi.

### 3.2.10 Modelling dan Model Evaluation

#### 3.2.10.1 Sentiment Analysis Using Baseline Model: Naive Bayes

1. Tokenisasi dengan CountVectorizer

```
# Mengimpor pustaka yang diperlukan
from sklearn.feature_extraction.text import
CountVectorizer
from sklearn.feature_extraction.text import
TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
```

```
# Metrik Evaluasi
from sklearn.metrics import accuracy_score, f1_score
from sklearn.metrics import classification_report,
confusion_matrix

# Tokenisasi teks
clf = CountVectorizer()
X_train_cv =
clf.fit_transform(X_train.ravel().astype(str)) #
Convert elements to strings before fitting
X_test_cv =
clf.transform(X_test.ravel().astype(str)) # Convert
elements to strings before transforming
```

Mengubah teks menjadi representasi numerik menggunakan CountVectorizer.

2. Mengonversi Tokenisasi ke TF-IDF

```
# Mengonversi ke TF-IDF
tf_transformer =
TfidfTransformer(use_idf=True).fit(X_train_cv)
X_train_tf = tf_transformer.transform(X_train_cv)
X_test_tf = tf_transformer.transform(X_test_cv)
```

Setelah tokenisasi, menerapkan TF-IDF Transformer untuk memberikan bobot pada kata berdasarkan frekuensi dalam dokumen.

3. Melatih Model Naive Bayes

```
# Melatih model Naive Bayes
nb_clf = MultinomialNB()
nb_clf.fit(X_train_tf, y_train_le)
```

Mendefinisikan dan melatih Multinomial Naive Bayes Classifier untuk klasifikasi sentimen.

4. Memprediksi dan Menghitung Probabilitas

```
# Prediksi dan probabilitas Model Naive Bayes
nb_pred = nb_clf.predict(X_test_tf)
nb_probs = nb_clf.predict_proba(X_test_tf)[:, 1]
```

Setelah pelatihan, kita menggunakan model untuk membuat prediksi pada data uji dan menghitung probabilitas kelas.

#### 3.2.10.2 Evaluasi Model Naive Bayes

```
class BinaryClassificationEvaluator:
    def __init__(self, model_name="Model"):
        """Initialize evaluator with model name for plot titles."""
```

```
    self.model_name = model_name
    plt.style.use('seaborn-v0_8-whitegrid')

    def plot_training_history(self, history):
        """Plot training & validation loss/accuracy
curves."""
        fig, (ax1, ax2) = plt.subplots(1, 2,
figsize=(15, 5))

        # Loss plot
        ax1.plot(history['loss'], 'b-',
label='Training Loss', linewidth=2)
        ax1.plot(history['val_loss'], 'r--',
label='Validation Loss', linewidth=2)
        ax1.set_title(f'{self.model_name}: Training
and Validation Loss', pad=15)
        ax1.set_xlabel('Epoch')
        ax1.set_ylabel('Loss')
        ax1.legend()
        ax1.grid(True, alpha=0.3)

        # Accuracy plot
        ax2.plot(history['accuracy'], 'b-',
label='Training Accuracy', linewidth=2)
        ax2.plot(history['val_accuracy'], 'r--',
label='Validation Accuracy', linewidth=2)
        ax2.set_title(f'{self.model_name}: Training
and Validation Accuracy', pad=15)
        ax2.set_xlabel('Epoch')
        ax2.set_ylabel('Accuracy')
        ax2.legend()
        ax2.grid(True, alpha=0.3)

        plt.tight_layout()
        plt.show()

    def plot_confusion_matrix(self, y_true,
y_pred):
        """Plot confusion matrix heatmap."""
        cm = confusion_matrix(y_true, y_pred)
        plt.figure(figsize=(8, 6))

        sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
                    xticklabels=['Negative',
'Positive'],
                    yticklabels=['Negative',
'Positive'])

        plt.title(f'{self.model_name}: Confusion
```

```
Matrix', pad=15)
        plt.xlabel('Predicted Label')
        plt.ylabel('True Label')
        plt.show()

        # Print classification metrics
        print("\nClassification Report:")
        print(classification_report(y_true, y_pred,
target_names=['Negative', 'Positive']))

    def plot_roc_curve(self, y_true, y_pred_proba):
        """Plot ROC curve."""
        fpr, tpr, _ = roc_curve(y_true,
y_pred_proba)
        roc_auc = auc(fpr, tpr)

        plt.figure(figsize=(8, 6))
        plt.plot(fpr, tpr, color='darkorange',
lw=2,
                 label=f'ROC curve (AUC =
{roc_auc:.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.title(f'{self.model_name}: Receiver
Operating Characteristic', pad=15)
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.legend(loc="lower right")
        plt.grid(True, alpha=0.3)
        plt.show()

    def plot_precision_recall_curve(self, y_true,
y_pred_proba):
        """Plot Precision-Recall curve."""
        precision, recall, _ =
precision_recall_curve(y_true, y_pred_proba)
        avg_precision =
average_precision_score(y_true, y_pred_proba)

        plt.figure(figsize=(8, 6))
        plt.plot(recall, precision,
color='darkorange', lw=2,
                 label=f'PR curve (AP =
{avg_precision:.2f})')
        plt.xlim([0.0, 1.0])
        plt.ylim([0.0, 1.05])
        plt.title(f'{self.model_name}:
Precision-Recall Curve', pad=15)
```

```
        plt.xlabel('Recall')
        plt.ylabel('Precision')
        plt.legend(loc="lower left")
        plt.grid(True, alpha=0.3)
        plt.show()

    def plot_misclassifications(self, y_true,
y_pred, X_test, max_samples=5):
        """Analyze and visualize misclassified
samples."""
        misclassified_idx = np.where(y_true !=
y_pred)[0]

        print(f"\nTotal misclassified samples:
{len(misclassified_idx)}")
        print(f"\nMisclassification rate:
{(len(misclassified_idx)/len(y_true))*100:.2f}%")

        if len(misclassified_idx) > 0:
            print("\nSample misclassified
instances:")
            for idx in
misclassified_idx[:max_samples]:
                print(f"\nText: {X_test[idx]}")
                print(f"True label: {y_true[idx]}")
                print(f"Predicted label:
{y_pred[idx]}")

        # Plot distribution of misclassifications
        plt.figure(figsize=(8, 6))
        sns.countplot(y=y_true[misclassified_idx])
        plt.title(f'{self.model_name}: Distribution
of Misclassified Samples', pad=15)
        plt.xlabel('Count')
        plt.ylabel('True Label')
        plt.show()

# Inisialisasi evaluator
nb_evaluator = BinaryClassificationEvaluator("Naive
Bayes")

# Plot confusion matrix
nb_evaluator.plot_confusion_matrix(y_test_le,
nb_pred)

# Plot ROC curve
nb_evaluator.plot_roc_curve(y_test_le, nb_probs)

# Plot Precision-Recall curve
nb_evaluator.plot_precision_recall_curve(y_test_le,
```

```
nb_probs)

# Plot misclassifications
nb_evaluator.plot_misclassifications(y_test_le,
nb_pred, X_test)
```

### 3.2.10.3 Sentiment Analysis Using BERT

1. Load Pre-trained Tokenizer

```
# Transformers
from transformers import BertTokenizerFast,
TFBertModel

# Inisialisasi tokenizer BERT
tokenizer =
BertTokenizerFast.from_pretrained('bert-base-uncase
d')
```

Menggunakan tokenizer BERT-base uncased untuk mengubah teks menjadi token yang sesuai dengan format input BERT.

2. Define Maximum Sequence Length

```
MAX_LEN=128
```

Mendefinisikan panjang maksimum token yang dapat diproses oleh model.

3. Tokenization Function

```
def tokenize(data,max_len=MAX_LEN) :
    input_ids = []
    attention_masks = []
    for i in range(len(data)):
        # Ensure data[i] is a string, handle
potential nested arrays
        text = data[i][0] if isinstance(data[i],
(np.ndarray, list)) and len(data[i]) > 0 else
str(data[i])
        encoded = tokenizer.encode_plus(
            text,  # Pass the string to encode_plus
            add_special_tokens=True,
            max_length=MAX_LEN,
            padding='max_length',
            return_attention_mask=True
        )
        input_ids.append(encoded['input_ids'])

attention_masks.append(encoded['attention_mask'])
```

```
    return
np.array(input_ids),np.array(attention_masks)
```

Mendefinisikan fungsi untuk melakukan tokenisasi teks, menambahkan token khusus, memastikan panjang token, dan mengembalikan input berupa input_ids dan attention_mask.

4. Tokenize Dataset

```
train_input_ids, train_attention_masks =
tokenize(X_train, MAX_LEN)
val_input_ids, val_attention_masks =
tokenize(X_valid, MAX_LEN)
test_input_ids, test_attention_masks =
tokenize(X_test, MAX_LEN)
```

Memproses dataset latih, validasi, dan uji untuk menghasilkan token numerik dan attention mask dengan mengaplikasikan fungsi tokenisasi.

5. Load Pre-trained BERT Model

```
# Inisialisasi model BERT
bert_model =
TFBertModel.from_pretrained('bert-base-uncased')
```

Memuat model BERT-base uncased untuk digunakan sebagai backbone.

6. Define Model Architecture

```
def create_model(bert_model, max_len=128):
    input_ids = tf.keras.Input(shape=(max_len,),
dtype='int32')
    attention_masks =
tf.keras.Input(shape=(max_len,), dtype='int32')

    embeddings = bert_model([input_ids,
attention_masks])[1]
    dropout =
tf.keras.layers.Dropout(0.3)(embeddings)
    output = tf.keras.layers.Dense(2,
activation="softmax")(dropout)

    model =
tf.keras.models.Model(inputs=[input_ids,
attention_masks], outputs=output)
    model.compile(

optimizer=tf.keras.optimizers.Adam(learning_rate=1e
-5),
        loss='categorical_crossentropy',
```

```
        metrics=['accuracy']
    )
    return model
```

Membangun arsitektur model dengan menambahkan lapisan klasifikasi di atas model BERT.
Arsitektur model melibatkan:

- Input Layer untuk token IDs dan attention mask.
- Pre-Trained BERT Layer untuk ekstraksi fitur.
- Dropout Layer untuk mencegah overfitting.
- Dense Layer untuk klasifikasi sentimen.

7. Initialize and Compile Model

```
model = create_model(bert_model, MAX_LEN)
model.summary()
```

Membuat model dan menampilkan ringkasan arsitekturnya.

8. Train Model

```
history_bert = model.fit(
    [train_input_ids, train_attention_masks],
    y_train,
    validation_data=([val_input_ids,
val_attention_masks], y_valid),
    epochs=10,
    batch_size=16
)
```

Melatih model menggunakan dataset latih dan memvalidasi performa menggunakan dataset validasi.

#### 3.2.10.4 Evaluasi Model BERT

```
# Prediksi dengan model BERT
y_pred_proba_bert = model.predict([test_input_ids,
test_attention_masks])
y_pred_bert = np.argmax(y_pred_proba_bert, axis=1)

# Inisialisasi evaluator
bert_evaluator =
BinaryClassificationEvaluator("BERT")

bert_evaluator.plot_training_history(history_bert.
history)

bert_evaluator.plot_confusion_matrix(y_test_le,
y_pred_bert)
```

```
bert_evaluator.plot_roc_curve(y_test_le,
y_pred_proba_bert[:, 1])

bert_evaluator.plot_precision_recall_curve(y_test_
le, y_pred_proba_bert[:, 1])

bert_evaluator.plot_misclassifications(y_test_le,
y_pred_bert, X_test)
```

Langkah-langkah Prediksi:

1. Proses Prediksi Probabilitas
    - Model mengeluarkan probabilitas untuk setiap sampel yang menunjukkan keyakinannya terhadap masing-masing kelas (misalnya, Negative atau Positive).
    - Probabilitas ini direpresentasikan dalam array dua dimensi di mana setiap baris mewakili satu sampel, dan kolom menunjukkan skor probabilitas untuk masing-masing kelas.

2. Mengambil Kelas dengan Probabilitas Tertinggi
    - Dengan menggunakan fungsi np.argmax, kita memilih indeks kelas dengan probabilitas tertinggi sebagai prediksi akhir.

3. Plot History Pelatihan
   Visualisasi ini menunjukkan bagaimana loss (fungsi kerugian) dan akurasi model berkembang selama setiap epoch pelatihan. Grafik ini penting untuk:
    - Memastikan bahwa model mengalami konvergensi, yaitu penurunan loss secara konsisten.
    - Mengidentifikasi potensi overfitting, yang terlihat jika akurasi pada dataset latih sangat tinggi tetapi akurasi pada dataset validasi stagnan atau menurun.

4. Confusion Matrix dan Classification Report
   Confusion Matrix memberikan gambaran detail tentang distribusi prediksi model. Matriks ini membantu untuk:
    - Mengetahui jumlah prediksi benar (True Positives dan True Negatives).
    - Mengidentifikasi jumlah kesalahan prediksi (False Positives dan False Negatives).

   Classification Report mencakup metrik penting seperti:
    - Precision: Proporsi prediksi benar terhadap semua prediksi positif.
    - Recall: Proporsi prediksi benar terhadap semua sampel positif yang sebenarnya.
    - F1-Score: Harmoni antara precision dan recall.

5. Plot ROC Curve
   ROC (Receiver Operating Characteristic) Curve menunjukkan trade-off antara True Positive Rate (TPR) dan False Positive Rate (FPR) pada berbagai

ambang batas probabilitas. Area di bawah kurva (AUC) adalah indikator seberapa baik model membedakan antara kelas.

Keunggulan dari analisis ini:

- Menggambarkan performa model secara menyeluruh pada berbagai ambang batas probabilitas.
- AUC tinggi menunjukkan kemampuan model untuk memprediksi dengan baik, bahkan pada dataset yang tidak seimbang.

6. Precision-Recall Curve
   Grafik ini sangat berguna jika dataset memiliki distribusi kelas yang tidak seimbang. Kurva ini menunjukkan hubungan antara Precision dan Recall pada berbagai ambang batas.

   Tujuan:

   - Membantu memilih ambang batas probabilitas yang optimal, terutama ketika trade-off antara precision dan recall menjadi penting.
   - Area di bawah Precision-Recall Curve juga menjadi indikator performa model.

7. Analisis Misclassifications
   Menampilkan sampel teks yang salah diklasifikasikan oleh model. Analisis ini penting untuk:

   - Mengidentifikasi pola kesalahan model, misalnya apakah model kesulitan memahami konteks tertentu.
   - Memberikan wawasan tentang langkah-langkah perbaikan, seperti pelatihan ulang pada data yang lebih beragam atau penambahan fitur tambahan.

### 3.3 Screenshot Output
### 3.3.1 Load Dataset

1. Informasi Dataset Kandidat Anies

```
 1   # Informasi dataset kandidat anies
 2   anies_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unnamed: 0.1    10000 non-null  int64
 1    Date           10000 non-null  object
 2    Created        10000 non-null  object
 3    User ID        10000 non-null  float64
 4    Followers      10000 non-null  int64
 5    Following      10000 non-null  int64
 6    Tweet Count    10000 non-null  int64
 7    TweetLocation  4960 non-null   object
 8   Text            9934 non-null   object
 9   label           9997 non-null   object
dtypes: float64(1), int64(4), object(5)
memory usage: 781.4+ KB
```

Informasi umum tentang dataset Anies Baswedan, dengan 10.000 entri dan 10 kolom. Beberapa kolom memiliki data yang hilang, seperti TweetLocation dan Text, yang mungkin perlu penanganan lebih lanjut. Kolom lainnya memiliki data lengkap.

2. Informasi Dataset Kandidat Prabowo

```
 1   # Informasi dataset kandidat prabowo
 2   prabowo_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unnamed: 0.1    10000 non-null  int64
 1    Date           10000 non-null  object
 2    Created        10000 non-null  object
 3    User ID        10000 non-null  float64
 4    Followers      10000 non-null  int64
 5    Following      10000 non-null  int64
 6    Tweet Count    10000 non-null  int64
 7    TweetLocation  3552 non-null   object
 8   Text            9912 non-null   object
 9   label           10000 non-null  object
dtypes: float64(1), int64(4), object(5)
memory usage: 781.4+ KB
```

Informasi umum tentang dataset Prabowo, dengan 10.000 entri dan 10 kolom. Kolom TweetLocation memiliki banyak data yang hilang (sekitar setengahnya), sementara kolom lainnya tidak memiliki data yang hilang.

3. Informasi Dataset Kandidat Ganjar

```
# Informasi dataset kandidat ganjar
ganjar_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unnamed: 0      10000 non-null  int64
 1    Date           10000 non-null  object
 2    Created        10000 non-null  object
 3    User ID        10000 non-null  float64
 4    Followers      10000 non-null  int64
 5    Following      10000 non-null  int64
 6    Tweet Count    10000 non-null  int64
 7    TweetLocation  5293 non-null   object
 8   Text            9885 non-null   object
 9   label           10000 non-null  object
dtypes: float64(1), int64(4), object(5)
memory usage: 781.4+ KB
```

Informasi tentang dataset Ganjar, yang memiliki 10.000 entri dan 10 kolom. Kolom TweetLocation memiliki sebagian besar data hilang, sementara kolom lainnya lengkap.

### 3.3.2 Kombinasi Dataset

```
# Informasi dataset setelah digabung
df.head()
```

| | Unnamed: 0.1 | Date | Created | User ID | Followers | Following | Tweet Count | TweetLocation | Text | label | Candidate | Unnamed: 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 2023-04-16 10:04:35+00:00 | 2023-03-27 08:59:57+00:00 | 1.640000e+18 | 981 | 531 | 22 | NaN | anies president info | Positive | Anies Baswedan | NaN |
| 1 | 1.0 | 2023-04-16 10:00:01+00:00 | 2012-08-21 07:34:04+00:00 | 7.710306e+08 | 10702 | 123 | 30121 | Palmerah, Jakarta | gerindra party politician sandiaga uno answers... | Positive | Anies Baswedan | NaN |
| 2 | 2.0 | 2023-04-16 09:14:46+00:00 | 2011-09-27 05:22:24+00:00 | 3.807551e+08 | 11 | 35 | 230 | North Jakarta | mr. anies continued, we will guard him until h... | Positive | Anies Baswedan | NaN |
| 3 | 3.0 | 2023-04-16 07:03:05+00:00 | 2017-10-19 12:32:28+00:00 | 9.210000e+17 | 37 | 47 | 2670 | Jakarta | may allah swt save the nation and state of the... | Positive | Anies Baswedan | NaN |
| 4 | 4.0 | 2023-04-16 06:03:59+00:00 | 2022-06-28 07:03:37+00:00 | 1.540000e+18 | 6 | 129 | 766 | Lebak, Banten | poor chotimah, uncle anies, that's why my fami... | Positive | Anies Baswedan | NaN |

Hasil setelah penggabungan kolom Candidate pada masing-masing dataset kandidat dan menggabungkan ketiga dataset menjadi satu. Kolom Candidate menunjukkan nama calon yang terkait dengan setiap entri, sedangkan kolom lainnya berisi data terkait tweet, termasuk teks dan label sentimen.

### 3.3.3 Seleksi Kolom dan Penanganan Missing Value

1. Melihat Ukuran dan Missing Value dalam Dataset

```
# Melihat ukuran dataset
df.shape
```

```
(30000, 12)
```

```
# Melihat missing value dalam dataset
df.isnull().sum()
```

| | 0 |
|---|---|
| Unnamed: 0.1 | 10000 |
| Date | 0 |
| Created | 0 |
| User ID | 0 |
| Followers | 0 |
| Following | 0 |
| Tweet Count | 0 |
| TweetLocation | 16195 |
| Text | 269 |
| label | 3 |
| Candidate | 0 |
| Unnamed: 0 | 20000 |

Ukuran dataset gabungan adalah 30.000 baris dan 12 kolom. Selanjutnya, informasi mengenai jumlah data yang hilang (missing values) mengungkapkan bahwa kolom TweetLocation memiliki banyak data kosong (16.195), diikuti oleh Text (269) dan label (3). Kolom Unnamed: 0.1 dan Unnamed: 0 masing-masing memiliki 10.000 dan 20.000 nilai kosong, menunjukkan bahwa kolom-kolom tersebut kemungkinan tidak relevan dan dapat dihapus untuk penyederhanaan analisis.

2. Seleksi Kolom yang diperlukan

```
# Melihat kembali dataset setelah dilakukan seleksi kolom
df.head()
```

| | Tweet Count | Text | label | Candidate | Date |
|---|---|---|---|---|---|
| 0 | 22 | anies president info | Positive | Anies Baswedan | 2023-04-16 10:04:35+00:00 |
| 1 | 30121 | gerindra party politician sandiaga uno answers... | Positive | Anies Baswedan | 2023-04-16 10:00:01+00:00 |
| 2 | 230 | mr. anies continued, we will guard him until h... | Positive | Anies Baswedan | 2023-04-16 09:14:46+00:00 |
| 3 | 2670 | may allah swt save the nation and state of the... | Positive | Anies Baswedan | 2023-04-16 07:03:05+00:00 |
| 4 | 766 | poor chotimah, uncle anies, that's why my fami... | Positive | Anies Baswedan | 2023-04-16 06:03:59+00:00 |

```
# Melihat kembali struktur dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 29728 entries, 0 to 9999
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0    Tweet Count  29728 non-null  int64 
 1   Text          29728 non-null  object
 2   label         29728 non-null  object
 3   Candidate     29728 non-null  object
 4    Date         29728 non-null  object
dtypes: int64(1), object(4)
memory usage: 1.4+ MB
```

Dataset yang telah diseleksi hanya menyisakan kolom-kolom relevan: jumlah tweet, teks, label sentimen, kandidat, dan tanggal. Setelah menghapus data kosong, jumlah baris menjadi 29.728. Struktur dataset kini lebih bersih, tanpa missing values, siap untuk proses analisis lebih lanjut.

3. Statistik Deskriptif

```
# Melihat statistik deskriptif
df.describe()
```

| | Tweet Count |
|---|---|
| count | 2.972800e+04 |
| mean | 3.605859e+04 |
| std | 1.712621e+05 |
| min | 1.000000e+00 |
| 25% | 5.720000e+02 |
| 50% | 2.447000e+03 |
| 75% | 1.432500e+04 |
| max | 2.310131e+06 |

Statistik deskriptif memberikan gambaran umum tentang distribusi jumlah tweet. Rata-rata jumlah tweet adalah 36.058, dengan nilai minimum 1 dan maksimum lebih dari 2,3 juta. Rentang yang luas dan standar deviasi tinggi menunjukkan adanya perbedaan signifikan dalam jumlah tweet di antara pengguna.

4. Cek kembali Missing Value dalam Dataset

```
# Mengecek kembali missing value dalam dataset
df.isnull().sum()
```

| | 0 |
|---|---|
| Tweet Count | 0 |
| Text | 0 |
| label | 0 |
| Candidate | 0 |
| Date | 0 |

dtype: int64

Dataset sudah bersih dari nilai kosong setelah proses seleksi kolom dan penghapusan data kosong. Semua kolom memiliki jumlah entri yang lengkap, memastikan dataset siap untuk analisis lebih lanjut atau pemodelan.

5. Melihat kembali Ukuran Dataset

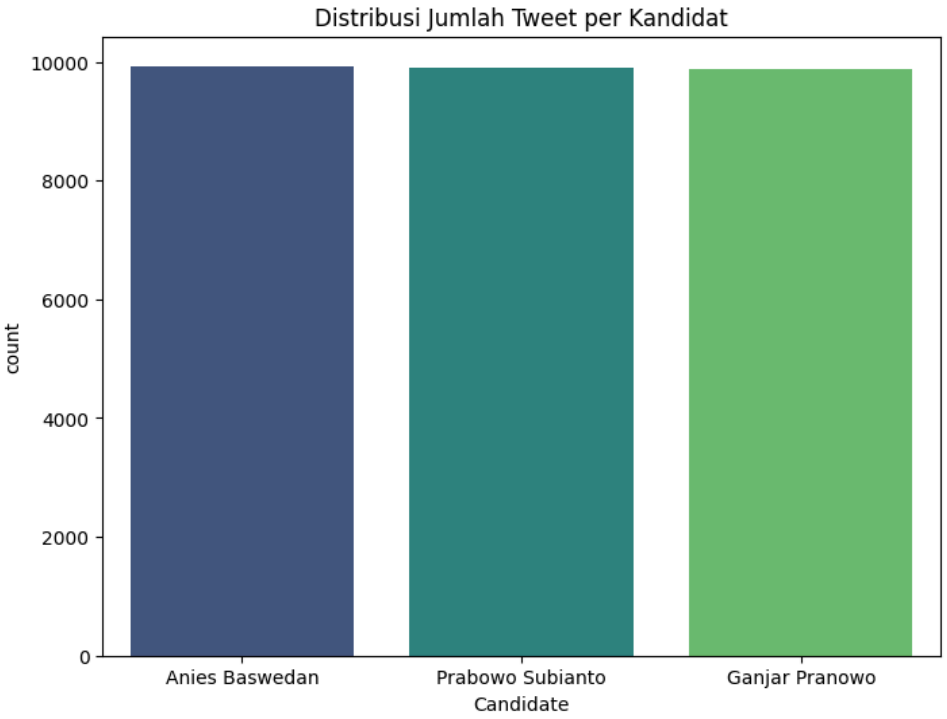
```
# Melihat kembali ukuran dataset
df.shape
```

```
(29728, 5)
```

Dataset memiliki 29.728 baris dan 5 kolom setelah proses seleksi kolom dan penghapusan data kosong. Ukuran ini menunjukkan jumlah data yang tersedia untuk analisis lebih lanjut.

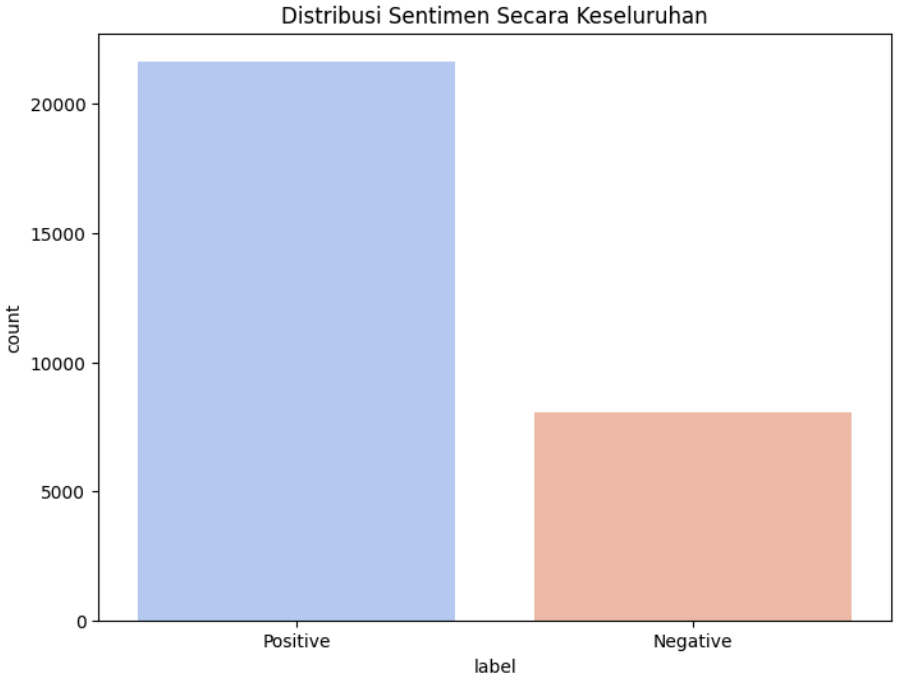
### 3.3.4 Exploratory Data Analysis (EDA)

1. Distribusi Jumlah Tweet per Kandidat



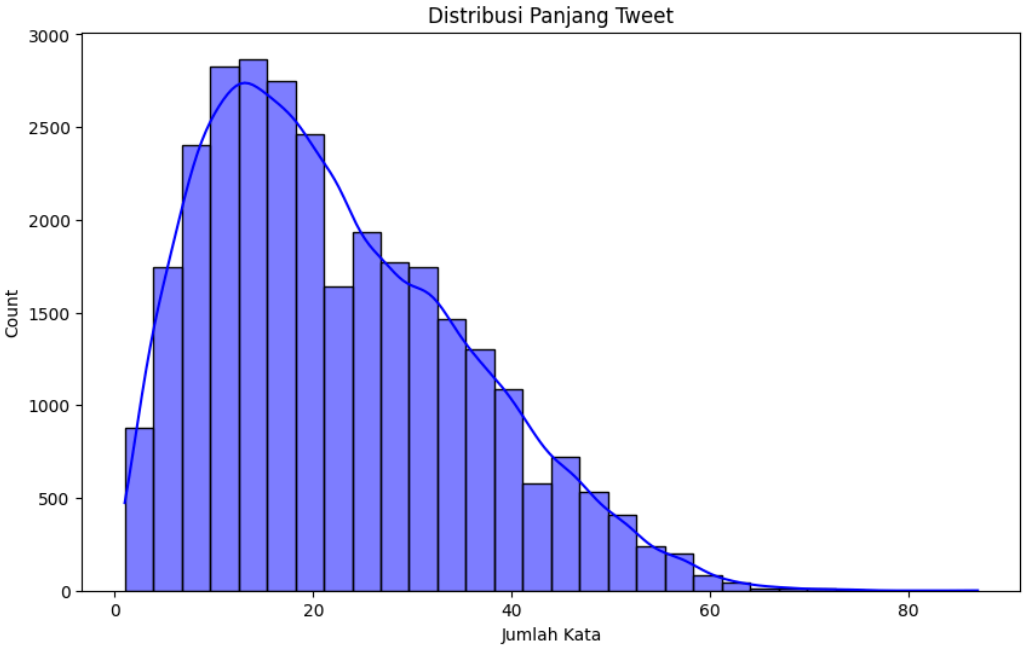

Dari plot bar tersebut dapat dilihat bahwa distribusi jumlah tweet masing-masing kandidat hampir memiliki persebaran yang sama.

2. Distribusi Keseluruhan Sentimen



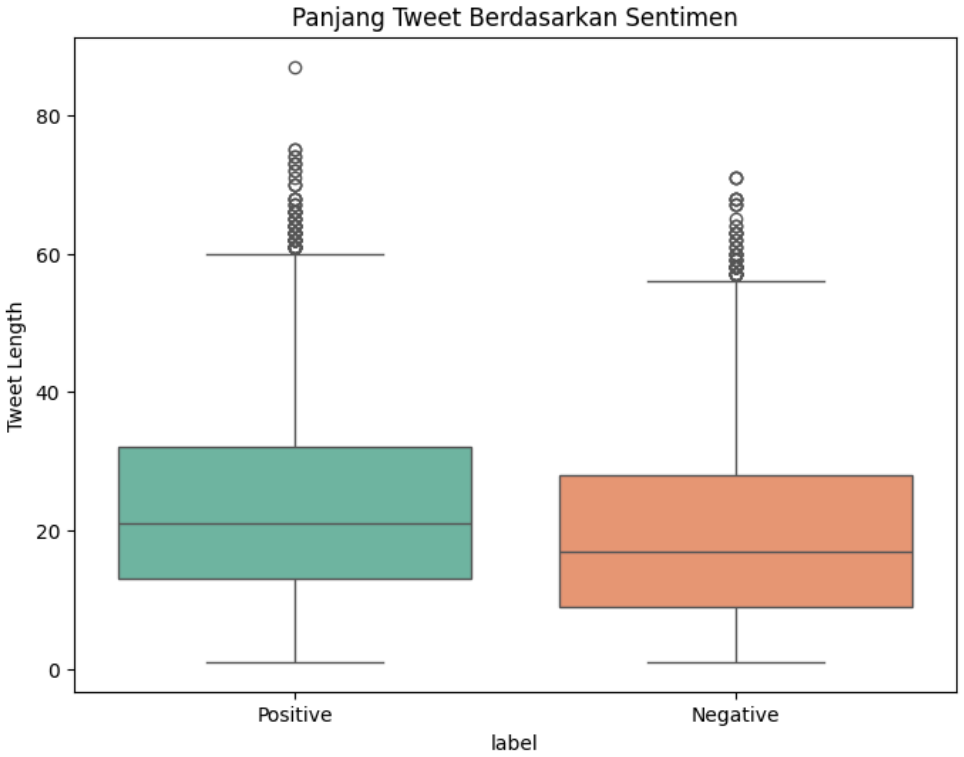

Dari plot bar tersebut dapat dilihat bahwa distribusi kelas ‘positive’ memiliki lebih banyak persebaran data dibandingkan dengan kelas ‘negative’.

3. Distribusi Panjang Tweet




Distribusi panjang tweet pada grafik menunjukkan bahwa sebagian besar tweet memiliki panjang sekitar 10–30 kata, dengan puncak distribusi berada di sekitar 20 kata. Kurva distribusi terlihat miring ke kanan, mengindikasikan adanya sejumlah kecil tweet dengan panjang kata yang jauh lebih banyak dari rata-rata.

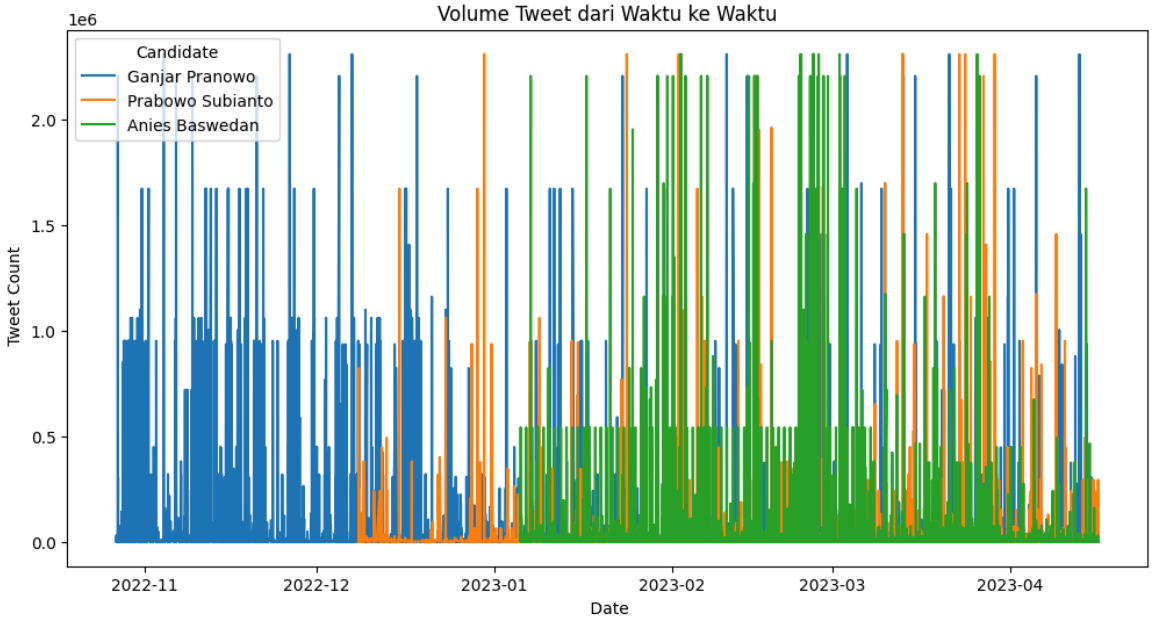
3. Panjang Tweet Berdasarkan Sentimen




Boxplot menunjukkan distribusi panjang tweet berdasarkan sentimen positif dan negatif. Median panjang tweet untuk kedua sentimen terlihat serupa, sekitar 20 kata. Namun, tweet positif cenderung memiliki lebih banyak outlier (tweet sangat panjang) dibandingkan tweet negatif. Distribusi panjang tweet untuk sentimen negatif lebih

simetris dibandingkan yang positif. Hal ini menunjukkan bahwa tweet dengan sentimen positif lebih bervariasi dalam panjangnya.

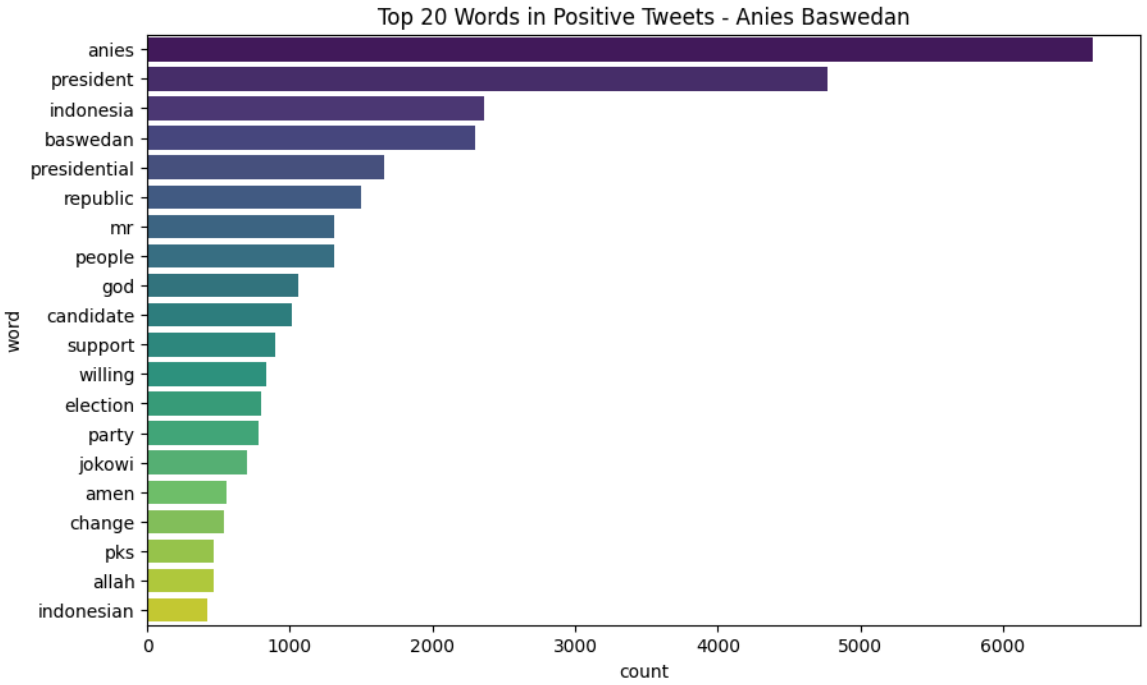
4. Volume Tweet dari Waktu ke Waktu



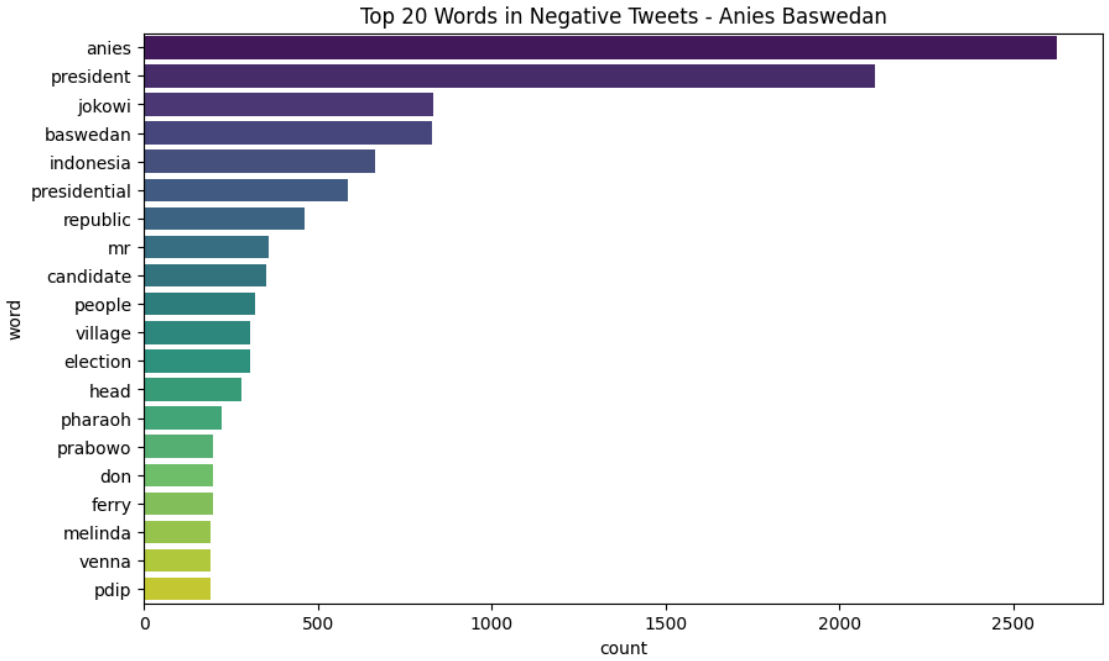

Grafik menunjukkan fluktuasi volume tweet dari waktu ke waktu untuk masing-masing kandidat.

- Ganjar Pranowo (biru): Volume tweet cenderung stabil dengan beberapa lonjakan signifikan di waktu tertentu di akhir tahun 2022, menunjukkan adanya momen atau peristiwa yang menarik perhatian publik.
- Prabowo Subianto (oranye): Volume tweet cenderung lebih rendah dibanding Ganjar, tetapi memiliki beberapa lonjakan tajam dan memiliki rentang waktu persebaran yang lebih luas dimulai dari awal tahun 2023.
- Anies Baswedan (hijau): Memiliki distribusi volume yang lebih terpusat di bulan-bulan tertentu, terutama menjelang periode tertentu, menunjukkan adanya diskusi yang meningkat terkait dirinya di media sosial.

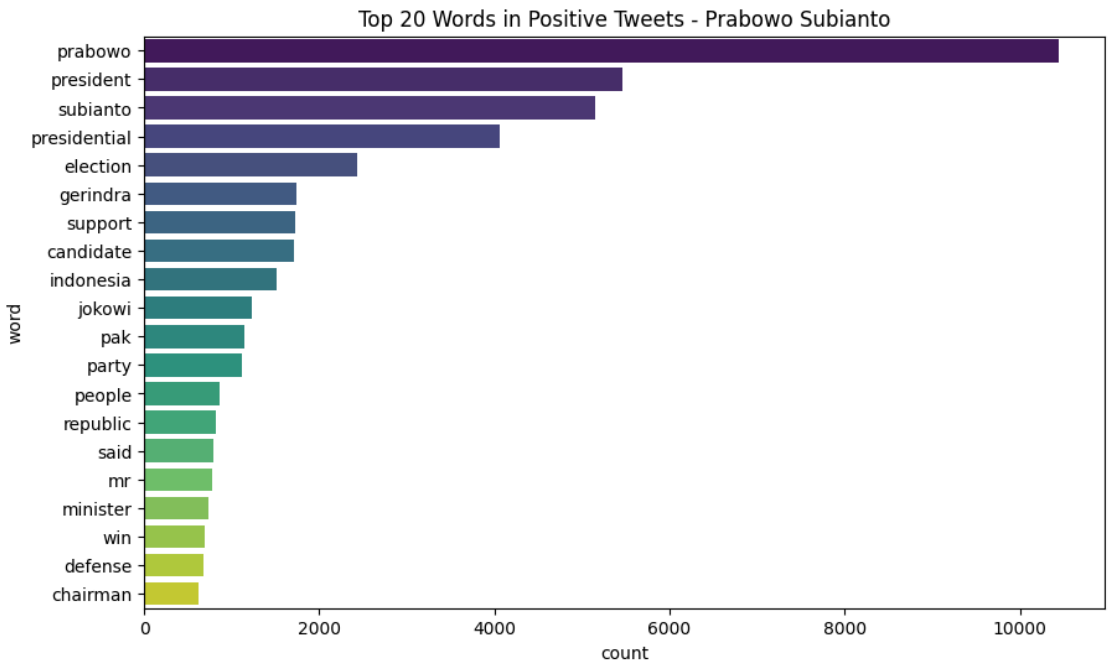
5. Frekuensi Kata Teratas



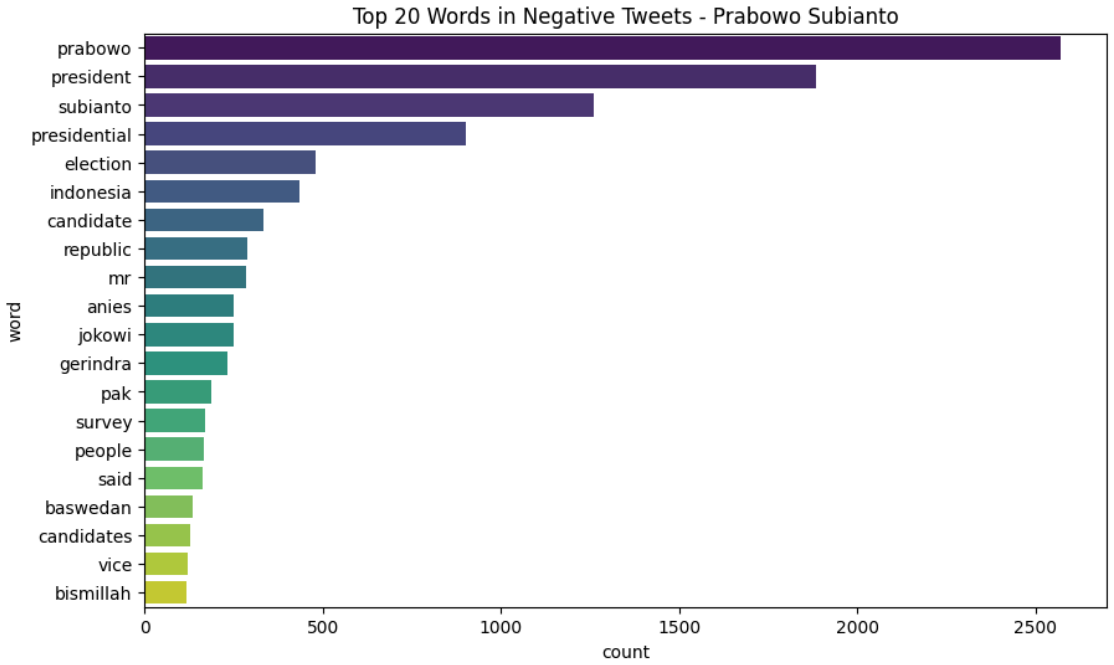

Grafik di atas menunjukkan jumlah 20 kata paling sering muncul pada tweets positif kandidat Anies.



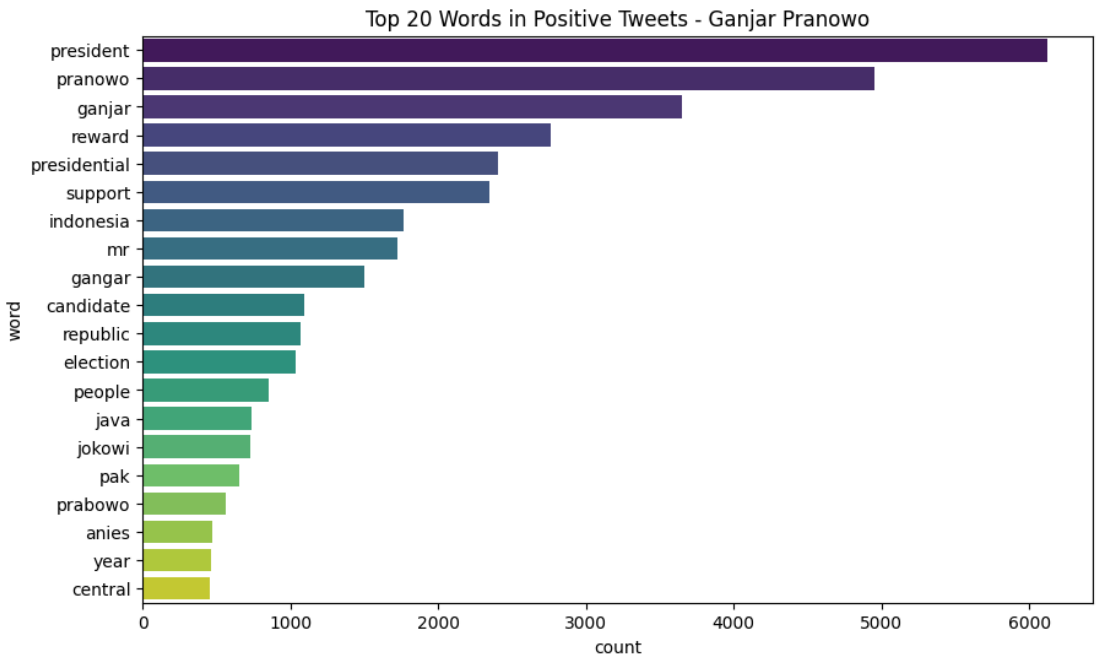

Grafik di atas menunjukkan jumlah 20 kata paling sering muncul pada tweets negatif kandidat Anies.



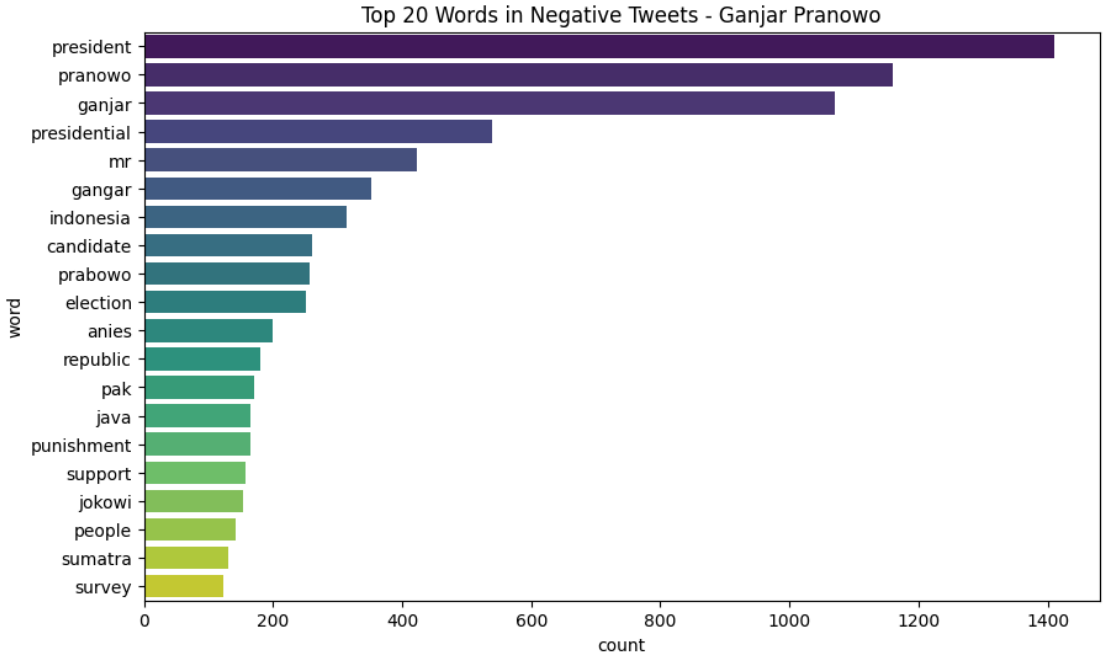

Grafik di atas menunjukkan jumlah 20 kata paling sering muncul pada tweets positif kandidat Prabowo.




Grafik di atas menunjukkan jumlah 20 kata paling sering muncul pada tweets negatif kandidat Prabowo.

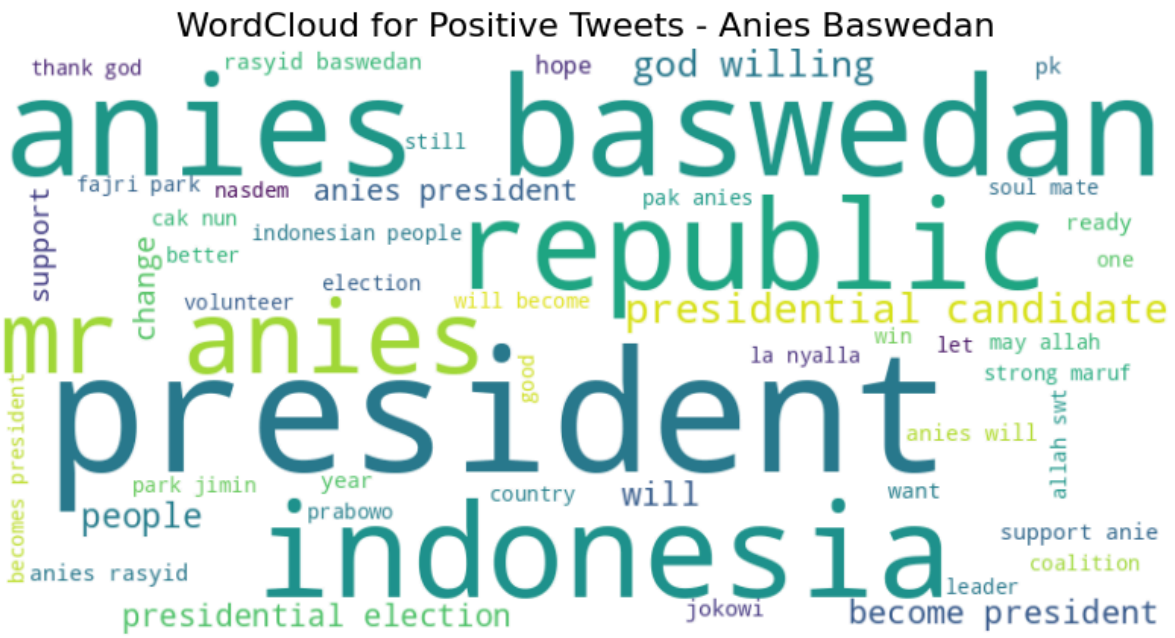



Grafik di atas menunjukkan jumlah 20 kata paling sering muncul pada tweets positif kandidat Ganjar.

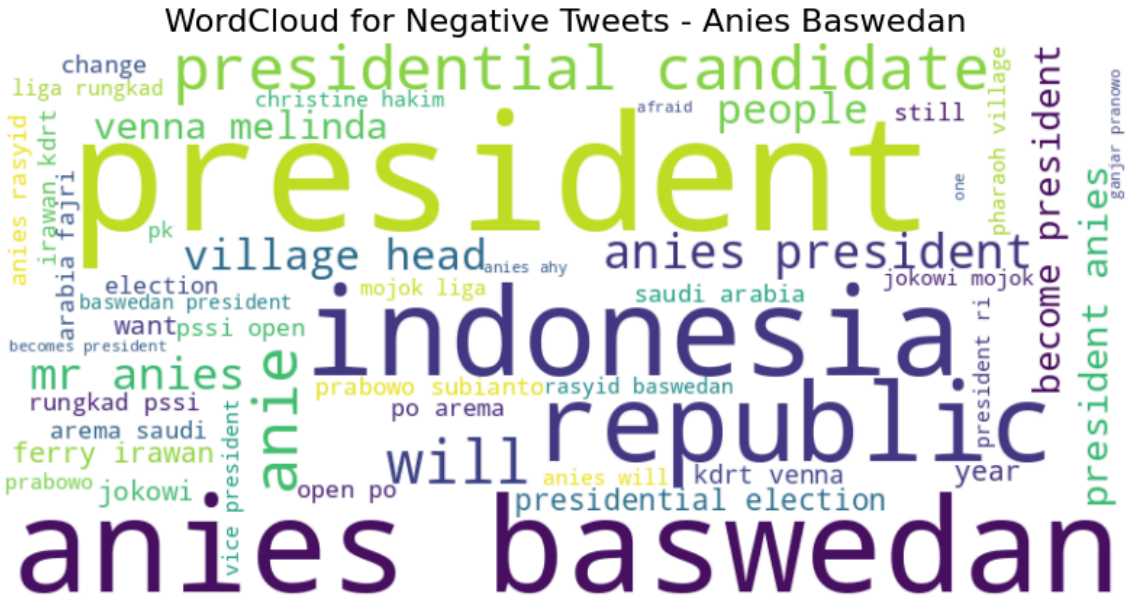



Grafik di atas menunjukkan jumlah 20 kata paling sering muncul pada tweets negatif kandidat Ganjar.

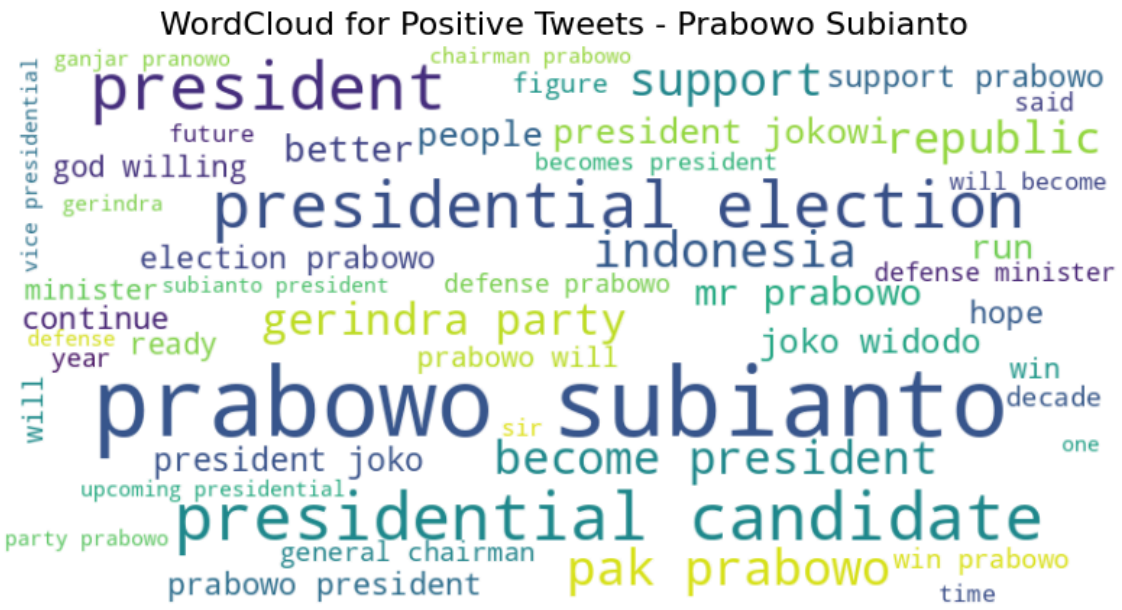
6. WorldCloud



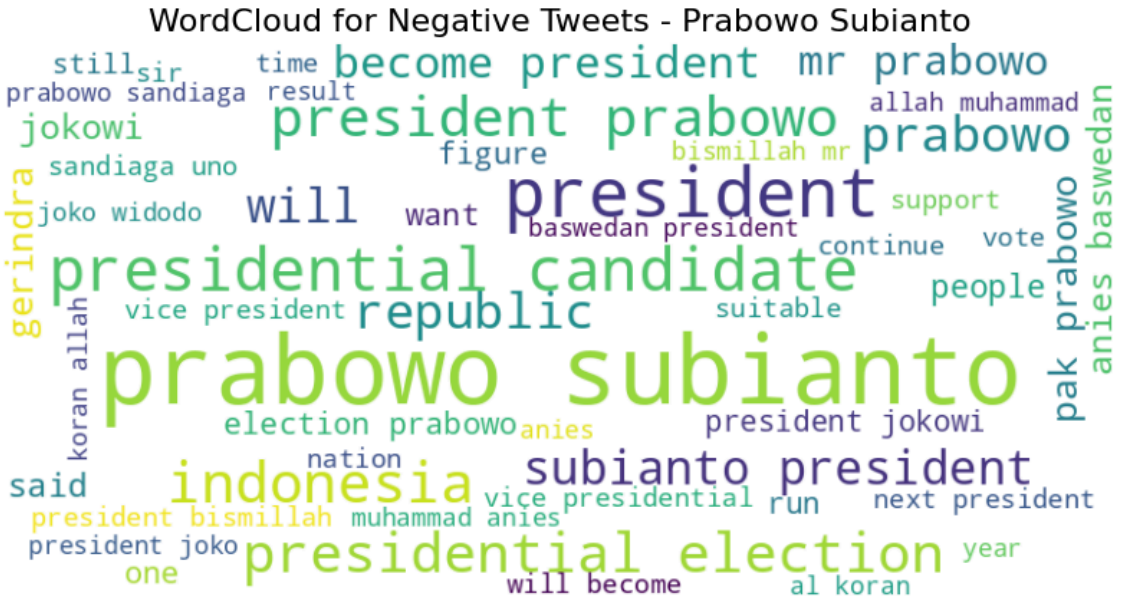

WordCloud berisi kata dengan ukuran yang sesuai dengan frekuensi kemunculannya untuk positif tweets kandidat Anies.



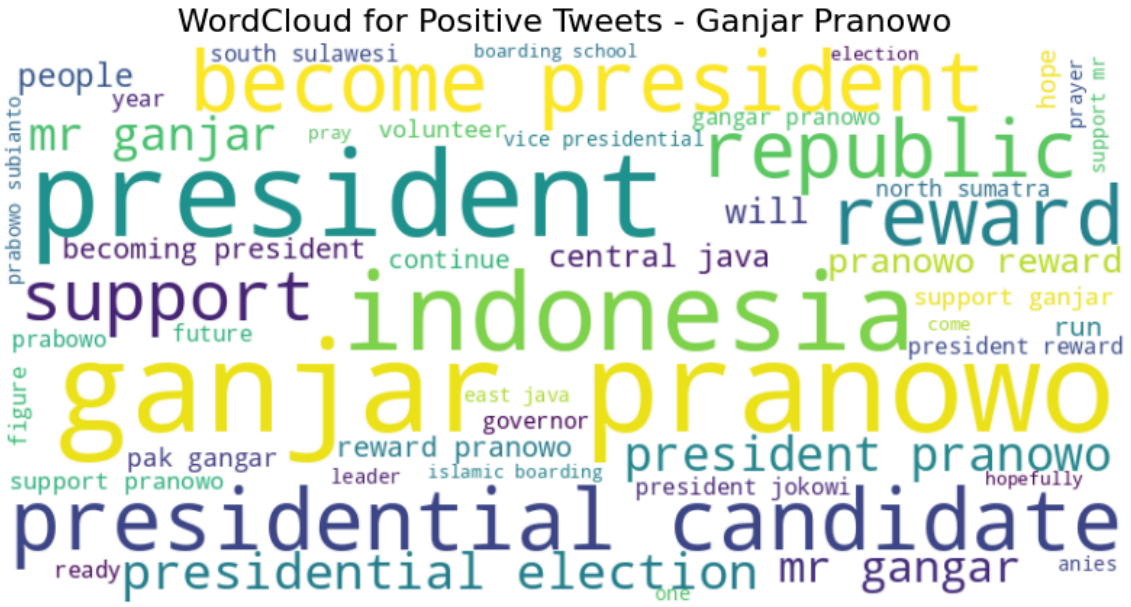

WordCloud berisi kata dengan ukuran yang sesuai dengan frekuensi kemunculannya untuk negatif tweets kandidat Anies.



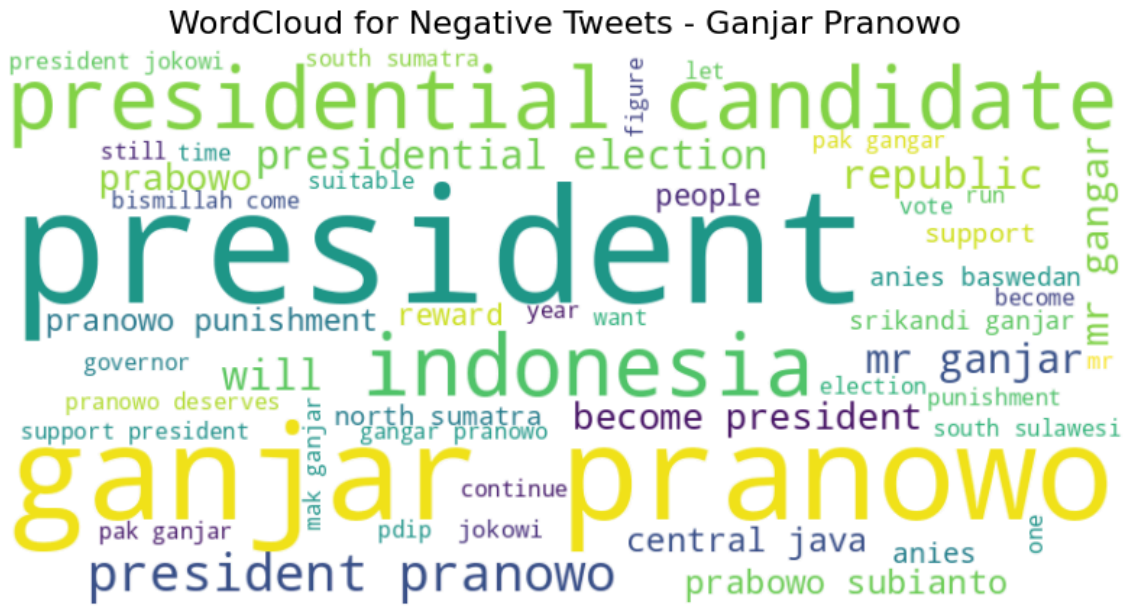

WordCloud berisi kata dengan ukuran yang sesuai dengan frekuensi kemunculannya untuk positif tweets kandidat Prabowo.




WordCloud berisi kata dengan ukuran yang sesuai dengan frekuensi kemunculannya untuk megatif tweets kandidat Prabowo.




WordCloud berisi kata dengan ukuran yang sesuai dengan frekuensi kemunculannya untuk positif tweets kandidat Ganjar.




WordCloud berisi kata dengan ukuran yang sesuai dengan frekuensi kemunculannya untuk negatif tweets kandidat Ganjar.

7. Analisis Kata dengan Batasan

```
'mr'    'bang'  'guys'  'bro'   'jk'    'lol'   'sama'  'btw'   'hehe'  'astagfirullah'
'masyaallah'    'goblog'        'yo'    'gt'    'subhanallah'   'sis'   'sip'   'mumet' 'ngehe' 'lah'
'r'     'kk'    'cap'   'coy'   'nya'   'nye'   'pls'   'gede'  'weh'   'ngl'
'hmm'   'meh'   'bestie'        'lu'    'fyi'   'ga'    'kali'  'nih'   'gk'    'tbh'
'anjay' 'kampret'       'rip'   'dude'  'trs'   'gokil' 'gpp'   'fomo'  'gue'   'yak'
'dong'  'asli'  'baru'  'omg'   'tai'   'deh'   'baper' 'njir'  'kece'
```

Mencari dan menghitung kata-kata spesifik seperti slang yang sudah ditetapkan sebagai acuan untuk preprocessing text.

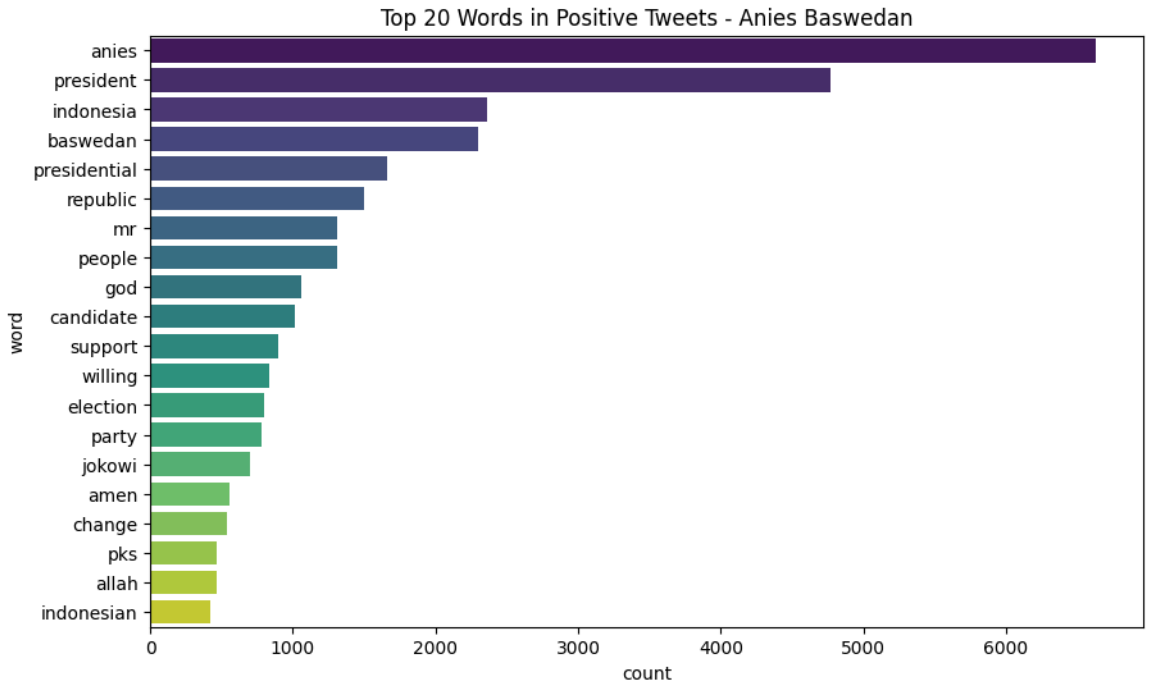
### 3.3.5 Data setelah Pre-processing

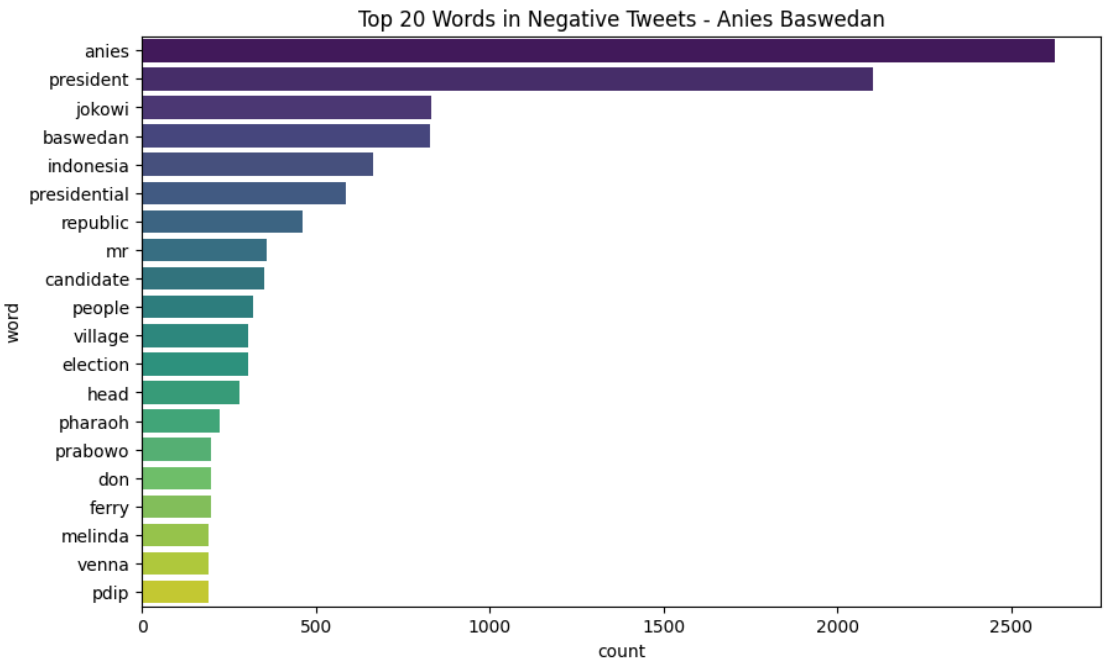
| | Tweet Count | Text | label | Candidate | Date | Tweet Length | clean_text |
|---|---|---|---|---|---|---|---|
| 0 | 22 | anies president info | Positive | Anies Baswedan | 2023-04-16 10:04:35+00:00 | 3 | anie president info |
| 1 | 30121 | gerindra party politician sandiaga uno answers... | Positive | Anies Baswedan | 2023-04-16 10:00:01+00:00 | 24 | gerindra party politician sandiaga uno answer ... |
| 2 | 230 | mr. anies continued, we will guard him until h... | Positive | Anies Baswedan | 2023-04-16 09:14:46+00:00 | 11 | mister anie continue guard president |
| 3 | 2670 | may allah swt save the nation and state of the... | Positive | Anies Baswedan | 2023-04-16 07:03:05+00:00 | 51 | allah swt save nation state republic indonesia... |
| 4 | 766 | poor chotimah, uncle anies, that's why my fami... | Positive | Anies Baswedan | 2023-04-16 06:03:59+00:00 | 16 | poor chotimah uncle anie s family decide elect... |

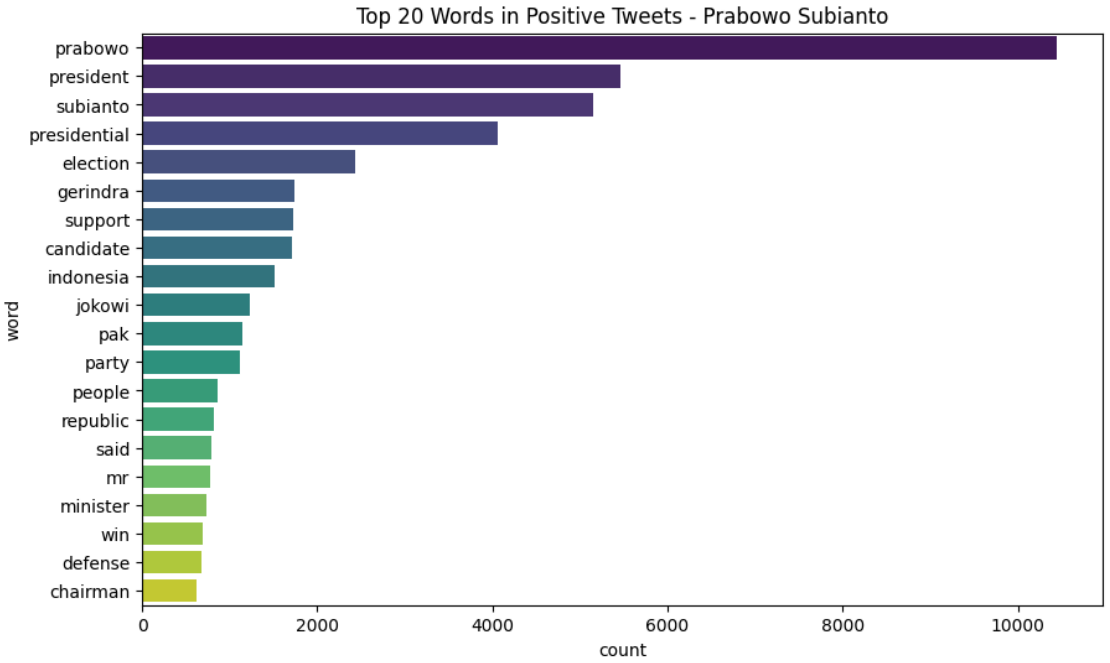
Menampilkan data baru setelah melewati preprocessing dan hasilnya tersimpan pada kolom clean_text.

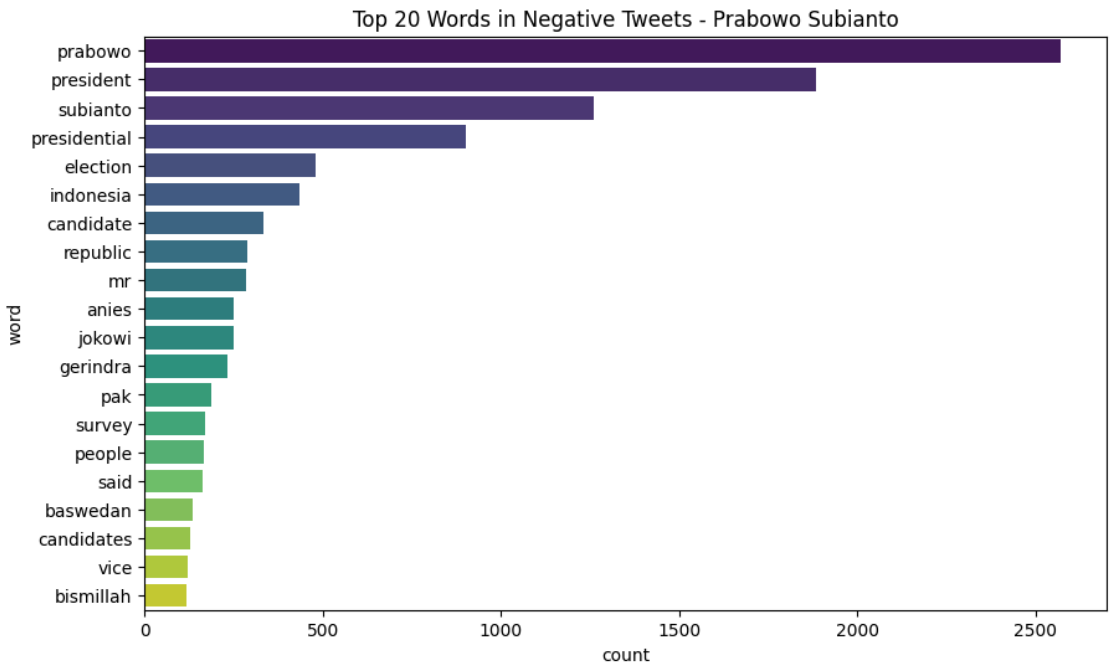
### 3.3.6 Exploratory Data Analysis after Preprocessing

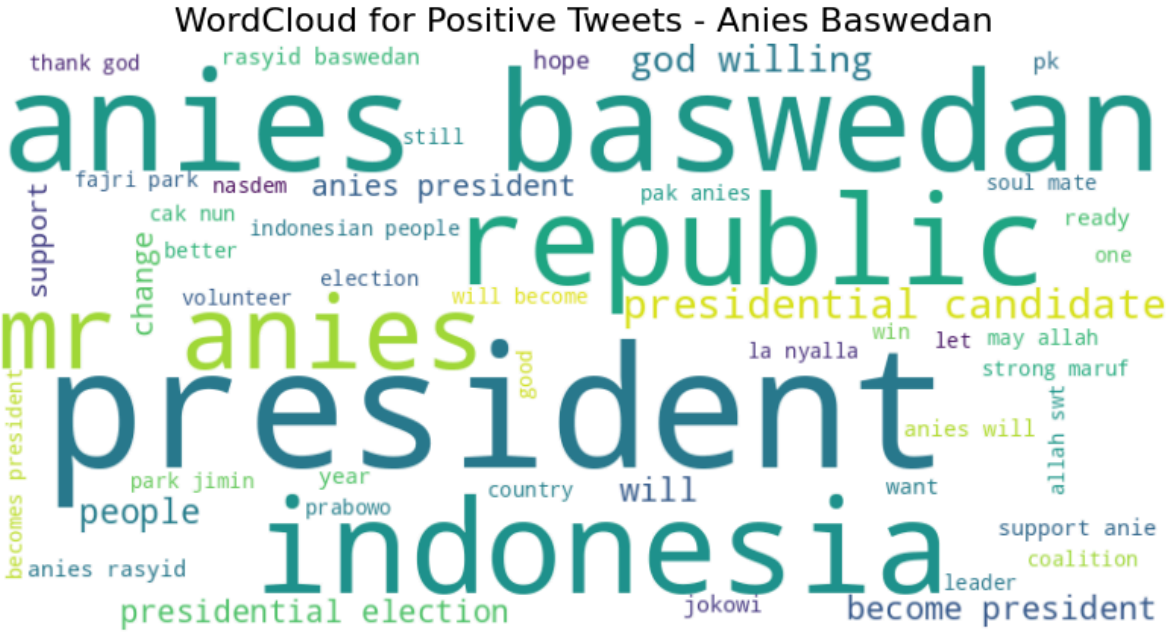
1. Visualisasi bar plot dan wordcloud untuk frekuensi kata yang sering muncul.



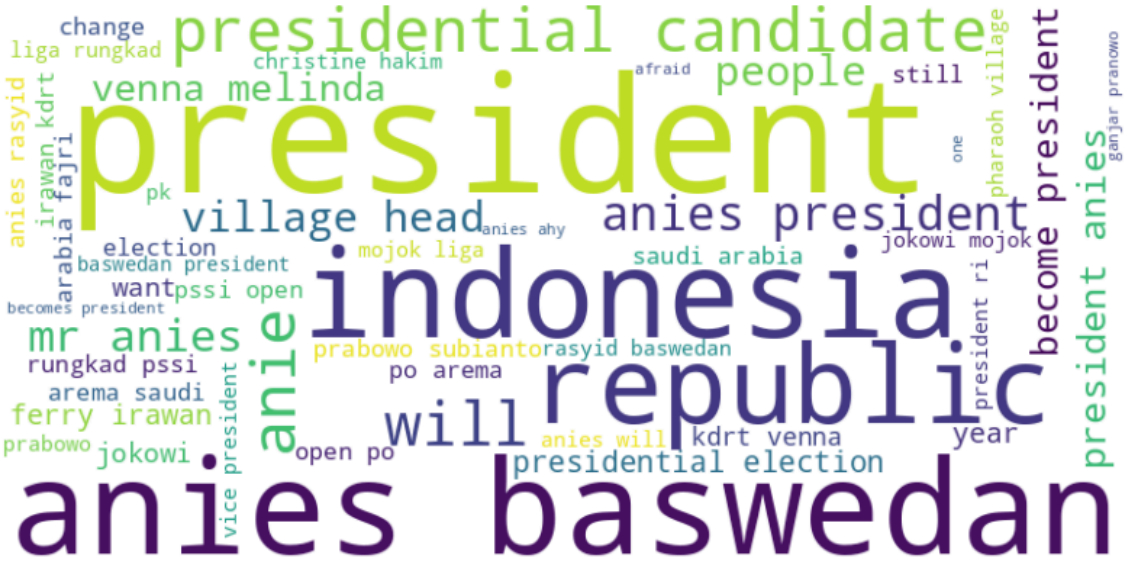

Top 20 Words in Negative Tweets - Anies Baswedan
anies
president
jokowi
baswedan
indonesia
presidential
republic
mr
candidate
people
village
election
head
pharaoh
prabowo
don
ferry
melinda
venna
pdip
word
0
500
1000
1500
2000
2500
count


Top 20 Words in Positive Tweets - Prabowo Subianto
prabowo
president
subianto
presidential
election
gerindra
support
candidate
indonesia
jokowi
pak
party
people
republic
said
mr
minister
win
defense
chairman
word
0
2000
4000
6000
8000
10000
count


Top 20 Words in Negative Tweets - Prabowo Subianto
prabowo
president
subianto
presidential
election
indonesia
candidate
republic
mr
anies
jokowi
gerindra
pak
survey
people
said
baswedan
candidates
vice
bismillah
word
0
500
1000
1500
2000
2500
count

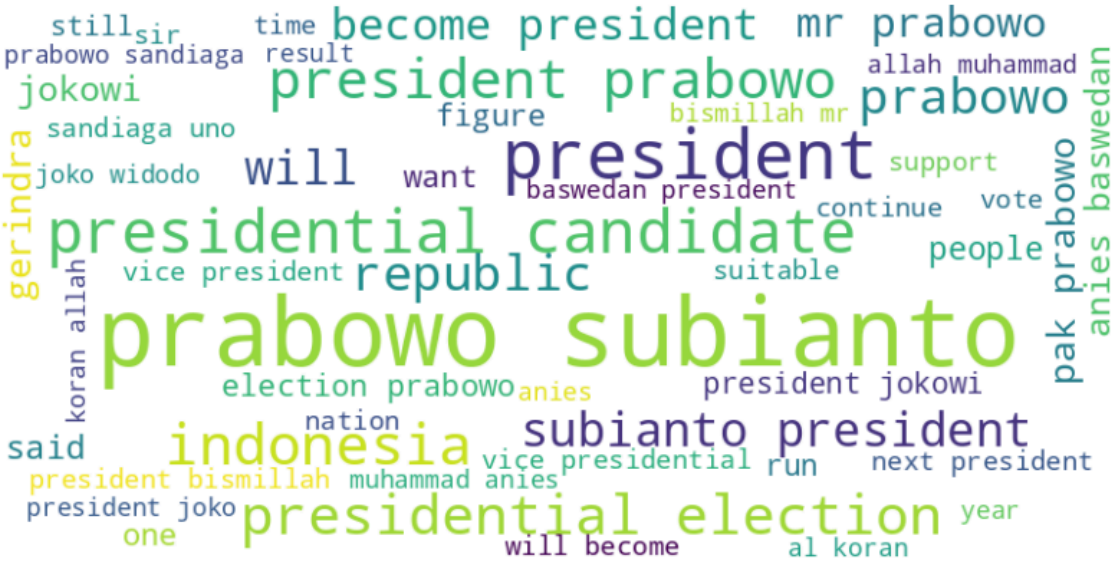

WordCloud for Positive Tweets - Anies Baswedan
thank god
rasyid baswedan
hope
god willing
pk
anies
baswedan
still
fajri park
nasdem
anies president
pak anies
soul mate
support
cak nun
indonesian people
republic
ready
change
better
one
election
volunteer
will become
presidential candidate
mr anies
win
let
may allah
la nyalla
president
good
strong maruf
anies will
allah swt
becomes president
park jimin
year
country
will
want
people
prabowo
support anie
coalition
anies rasyid
leader
indonesia
presidential election
jokowi
become president

WordCloud for Negative Tweets - Anies Baswedan

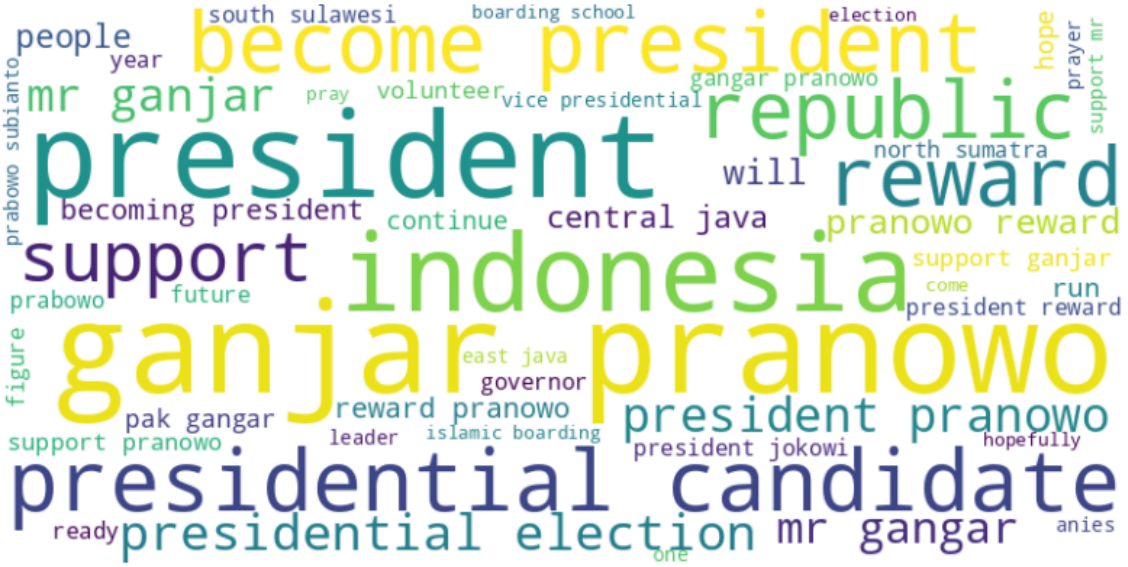


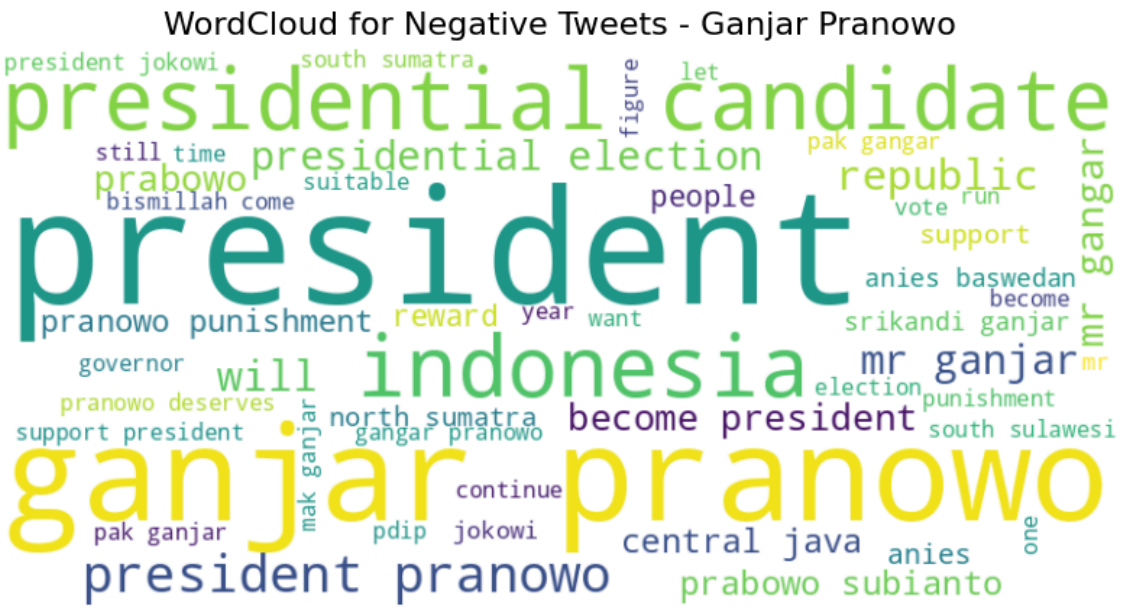

WordCloud for Positive Tweets - Prabowo Subianto

WordCloud for Negative Tweets - Prabowo Subianto




WordCloud for Positive Tweets - Ganjar Pranowo






Hasil eksplorasi data (EDA) setelah preprocessing, khususnya pada analisis frekuensi kata-kata dan WordCloud, menunjukkan pola yang konsisten. Setelah preprocessing, kata-kata yang lebih relevan dan sering digunakan dalam diskusi publik tentang kandidat dan topik politik akan lebih terlihat.

Meskipun proses WordCloud dan analisis frekuensi kata menggunakan pendekatan yang sedikit berbeda dalam representasi visual atau tabular, keduanya pada dasarnya menggambarkan tema yang sama. Kata-kata yang sering muncul (seperti nama kandidat, isu politik, atau opini) menjadi lebih terlihat setelah langkah-langkah pembersihan dan normalisasi diterapkan.

### 3.3.7 Data Sampling

1. Distribusi awal label sentimen

```
# Mengecek distribusi awal label sentimen
df['label'].value_counts()
```

| label | count |
|---|---|
| Positive | 21654 |
| Negative | 8074 |

dtype: int64

Hasil dari pengecekan distribusi label sentimen menunjukkan bahwa dataset memiliki ketidakseimbangan antara jumlah tweet yang berlabel "Positive" dan "Negative".

2. Label Mapping

```
Sebelum Mapping:
0    Positive
1    Positive
2    Positive
3    Positive
4    Positive
Name: label, dtype: object
```

```
Setelah Mapping:
0    1
1    1
2    1
3    1
4    1
Name: label, dtype: int64
```

Sebelum mapping label sentimen pada kolom label awalnya berisi nilai string, yaitu 'Positive' dan 'Negative'. Fungsi map() digunakan untuk mengganti nilai-nilai dalam kolom dengan pasangan yang sudah ditentukan, yaitu {'Positive': 1, 'Negative': 0}

3. Addressing Class Imbalance with Oversampling

```
label
1    21654
0     8074
Name: count, dtype: int64
```

```
label
1    21654
0    21654
Name: count, dtype: int64
```

Proses yang dilakukan dalam kode tersebut adalah oversampling menggunakan teknik RandomOverSampler untuk menangani masalah ketidakseimbangan kelas dalam dataset. Teknik ini bertujuan untuk menambah jumlah data pada kelas minoritas (dalam hal ini kelas Negative/0) dengan cara menggandakan sampel yang ada hingga jumlahnya setara dengan kelas mayoritas (kelas Positive/1).

4. Splitting dataset

```
TRAINING DATA: 25984
VALIDATION DATA: 8662
TESTING DATA: 8662
```

Dataset dibagi menjadi:

- Training Set (60%): Untuk melatih model dengan jumlah sampel sebanyak 25984 sampel.
- Validation Set (20%): Untuk menyetel hyperparameter model dengan jumlah sampel sebanyak 8662 sampel.
- Test Set (20%): Untuk mengevaluasi performa akhir model dengan jumlah sampel sebanyak 8662 sampel.

### 3.3.8 Modelling dan Model Evaluation

1. Initialize and Compile Model

```
Model: "model"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================================================
 input_1 (InputLayer)        [(None, 128)]             0         []

 input_2 (InputLayer)        [(None, 128)]             0         []

 tf_bert_model (TFBertModel  TFBaseModelOutputWithPooli 109482240 ['input_1[0][0]',
 )                           ngAndCrossAttentions(last_            'input_2[0][0]']
                             hidden_state=(None, 128, 7
                             68),
                              pooler_output=(None, 768)
                             , past_key_values=None, hi
                             dden_states=None, attentio
                             ns=None, cross_attentions=
                             None)

 dropout_37 (Dropout)        (None, 768)               0         ['tf_bert_model[0][1]']

 dense (Dense)               (None, 2)                 1538      ['dropout_37[0][0]']

=================================================================================================
Total params: 109483778 (417.65 MB)
Trainable params: 109483778 (417.65 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________________________________________
```

**Struktur Model**

1. **Input Layer**
    - Terdapat **dua input layer** (input_1 dan input_2).
    - Dimensi input: (None, 128).
        - **None** berarti jumlah sampel (batch size) dapat bervariasi.
        - **128** menunjukkan panjang maksimum token (maksimal 128 token per input teks setelah tokenisasi).
    - **Input_1** dan **Input_2** kemungkinan merupakan:
        - **Input_1**: Input token IDs (sekuens token teks).
        - **Input_2**: Input attention mask (penanda bagian teks yang relevan, biasanya 1 untuk token valid dan 0 untuk padding).
2. **BERT Layer**
    - **Layer Name**: tf_bert_model (TFBaseModelOutputWithPooling).
    - Ini adalah model inti **BERT (Bidirectional Encoder Representations from Transformers)**:
        - **Parameter Count**: 109,482,240 parameter (mayoritas parameter model ada di sini).
        - **Output Shape**:
            - **Hidden State**: (None, 128, 768)—Matriks representasi teks dari setiap token (768 dimensi untuk setiap token dalam sekuens).

        - **Pooler Output**: (None, 768)—Vektor representasi seluruh sekuens teks, biasanya dari token [CLS].
    - **Kegunaan**:
        - Hidden states dapat digunakan untuk tugas token-level (misalnya NER).
        - Pooler output digunakan untuk tugas sequence-level (misalnya klasifikasi sentimen).
3. **Dropout Layer**
    - Dropout rate diterapkan pada **pooler output** dari BERT.
    - **Output Shape**: (None, 768).
    - Dropout digunakan untuk mengurangi overfitting dengan mematikan unit-unit secara acak selama pelatihan.
4. **Dense (Output) Layer**
    - **Output Shape**: (None, 2).
    - Layer ini menggunakan unit **fully connected** untuk menghasilkan prediksi dua kelas (misalnya, **positif** dan **negatif**).
    - Aktivasi softmax kemungkinan diterapkan di layer ini untuk memberikan probabilitas output.

**Parameter dan Kapasitas Model**

- Total Parameters: 109,483,778
    - **Trainable Parameters:** Semua parameter bersifat trainable, termasuk yang ada di model BERT inti.
    - **Non-Trainable Parameters:** Tidak ada parameter non-trainable, artinya seluruh model termasuk fine-tuning.

# BAB 4 PENGUJIAN

Setelah kita melakukan proses implementasi pada proses sentimen analisis, diperlukan suatu parameter untuk menguji keberhasilan dari proses pengerjaan yang mana tahapan ini disebut sebagai Evaluasi Model. Proses evaluasi ini melibatkan beberapa tahapan mendalam terhadap hasil yang diperoleh. Evaluasi model sendiri melibatkan 2 model utama yang nantinya akan dibandingkan secara keseluruh performanya yakni baseline model Naive Bayes dan transformers model BERT.

## 4.1 Evaluasi Model Naive Bayes

### 4.1.1 Confusion Matrix

Visualisasi confusion matrix pada dasarnya bertujuan untuk memprediksi jumlah sampel yang benar dan sampel yang salah. Prediksi sampel yang benar dapat dilihat dari nilai 'True positif' dan 'True Negatif', sedangkan untuk kesalahan prediksi dapat dilihat dari 'False Positives' dan 'False Negatives'


Pada hasil visualisasi di atas, dapat dinilai bahwa distribusi prediksi model sudah bisa dikatakan baik. Hasil prediksi *true* memiliki akurasi yang cukup baik, dengan 3855 prediksi positif yang benar dan 3496 prediksi negatif yang benar. Meskipun demikian, model juga menghasilkan cukup banyak false positive (785) dan false negative (526 yang menunjukkan bahwa model masih memiliki ruang untuk peningkatan. Perbandingan rasio false positive dan false negative perlu dipertimbangkan, karena hal ini akan berdampak langsung pada metrik evaluasi model lainnya, seperti precision, recall, dan F1-score.

### 4.1.2 Classification Report

Merupakan bentuk evaluasi yang paling umum digunakan pada evaluasi model klasifikasi. Pada tahapan evaluasi ini terdapat 3 metrik utama sebagai penunjuk performa dari model yakni:

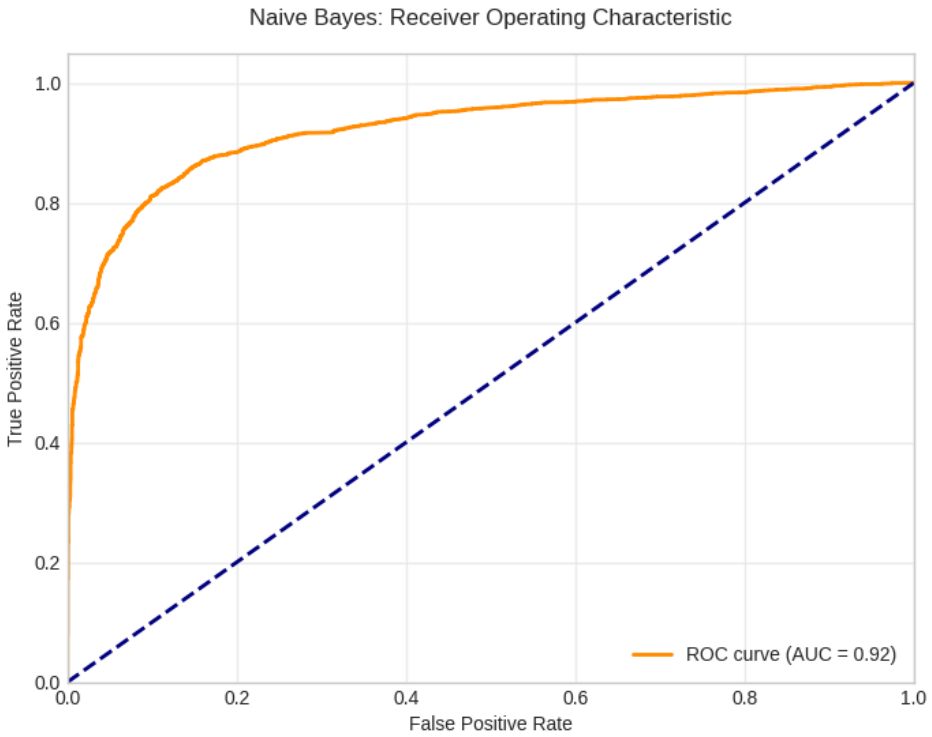
- Precision: Proporsi prediksi benar terhadap semua prediksi positif.
- Recall: Proporsi prediksi benar terhadap semua sampel positif yang sebenarnya.
- F1-Score: Harmoni antara precision dan recall.

```
Classification Report:
              precision    recall  f1-score   support

    Negative       0.87      0.82      0.84      4281
    Positive       0.83      0.88      0.85      4381

    accuracy                           0.85      8662
   macro avg       0.85      0.85      0.85      8662
weighted avg       0.85      0.85      0.85      8662
```

Dari hasil laporan di atas dapat disimpulkan bahwa nilai akurasi model naive bayes sudah cukup baik yakni mencapai nilai 0.85 atau 85% pada akurasinya. Nilai presisi / ketepatan sampel juga sudah cukup baik yakni 87% pada negatif dan 83% pada positif yang dilanjutkan dengan recall yang juga sudah cukup baik pada kedua model. Sehingga F1-score menunjukkan bahwa kedua sampel sudah cukup baik di angka 84% dan 85% namun masih terbuka pada peningkatan.

### 4.1.3 Plot ROC Curve

ROC (Receiver Operating Characteristic) Curve menunjukkan trade-off antara True Positive Rate (TPR) dan False Positive Rate (FPR) pada berbagai ambang batas probabilitas. Area di bawah kurva (AUC) adalah indikator seberapa baik model membedakan antara kelas.

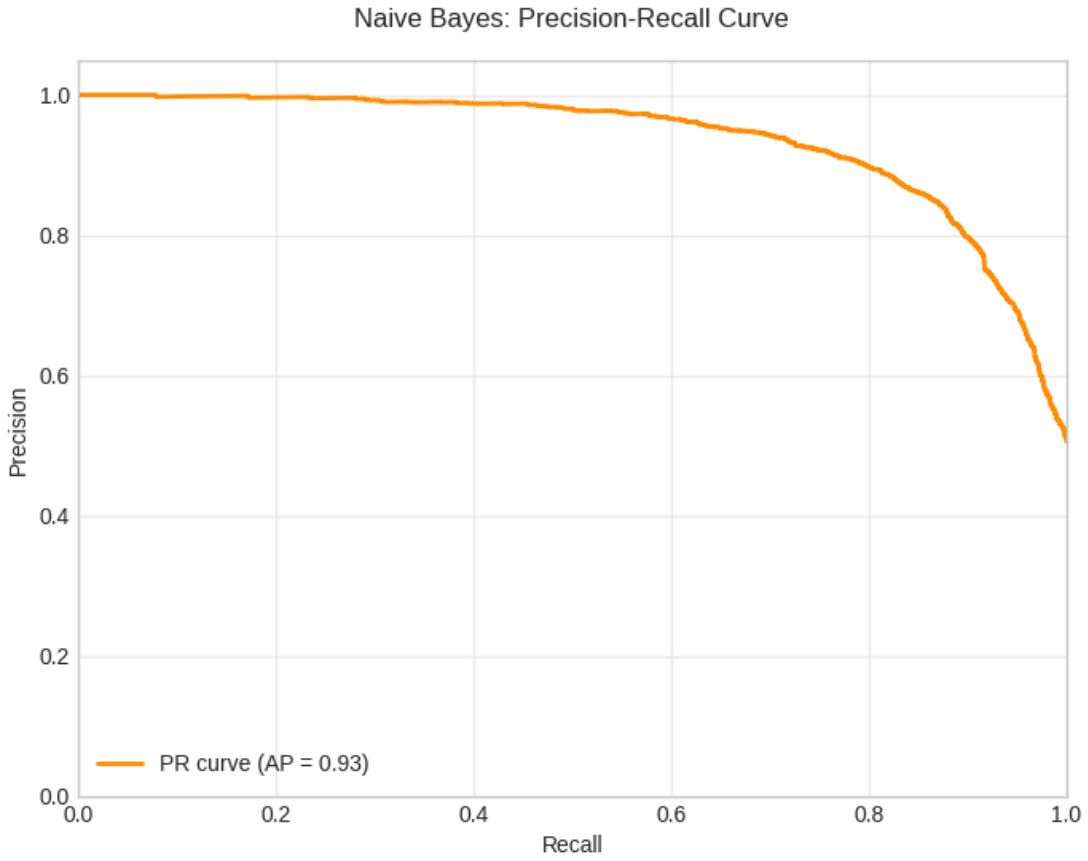



Nilai AUC (Area Under the Curve) pada kurva ROC adalah 0.92. Nilai AUC yang mendekati 1 menunjukkan bahwa model memiliki kemampuan klasifikasi yang sangat

baik. AUC 0.92 berarti model dapat membedakan dengan sangat baik antara kelas positif dan negatif.

### 4.1.4 Precision-Recall Curve

Grafik ini merupakan bentuk lain dari plotting yang berguna jika dataset memiliki distribusi kelas yang tidak seimbang. Kurva ini menunjukkan hubungan antara Precision dan Recall pada berbagai ambang batas dengan area di bawah Precision-Recall Curve juga menjadi indikator performa model.

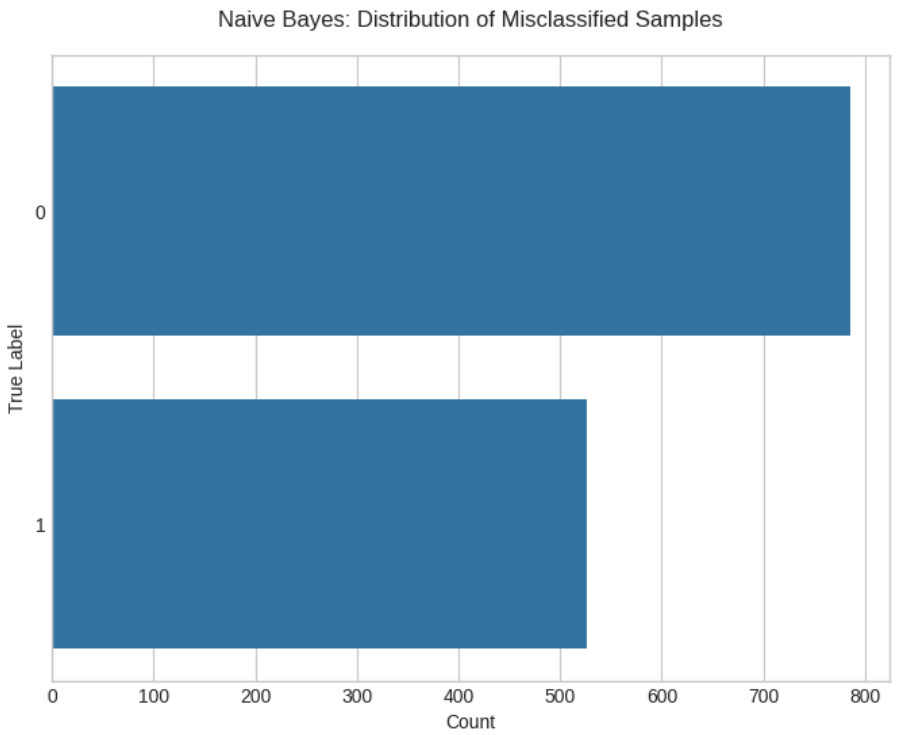



Nilai AP (Average Precision) pada kurva Precision-Recall adalah 0.93. AP yang mendekati 1 mengindikasikan bahwa model memiliki presisi yang sangat tinggi, artinya ketika model memprediksi suatu sampel sebagai positif, maka kemungkinan besar sampel tersebut memang benar-benar positif.

### 4.1.5 Analisis Misclassifications

Visualisasi ini merupakan salah satu bentuk analisis yang bertujuan untuk menampilkan sampel teks yang salah diklasifikasikan (missclassified) oleh model. Tujuan utamanya yakni memberikan wawasan mengenai langkah perbaikan yang mungkin masih perlu dilakukan untuk peningkatan performa dengan pelatihan data ulang ataupun penambahan fitur.

```
Total misclassified samples: 1311

Misclassification rate: 15.14%
```

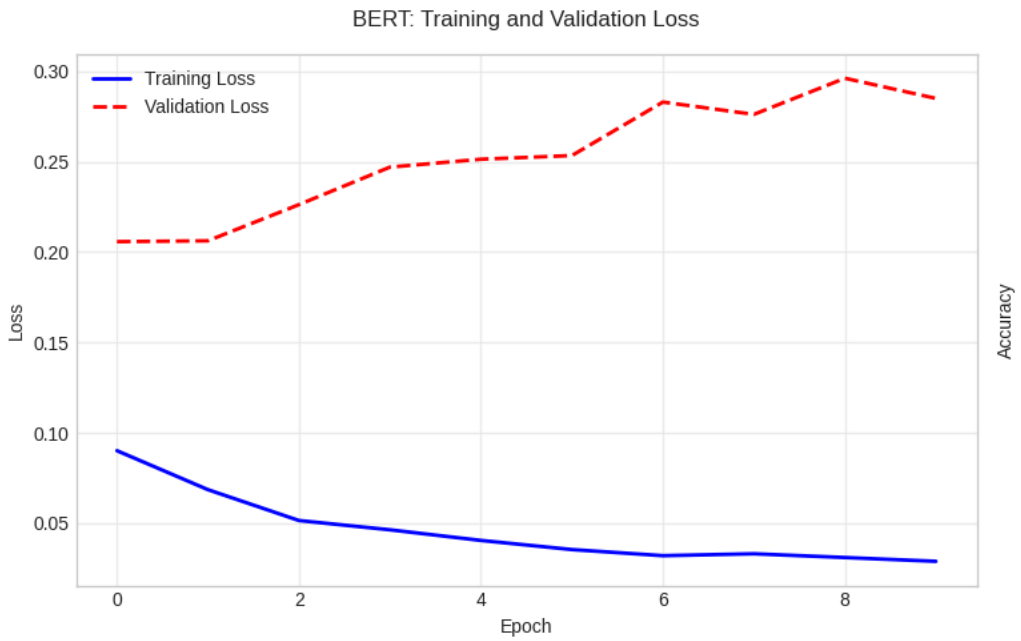


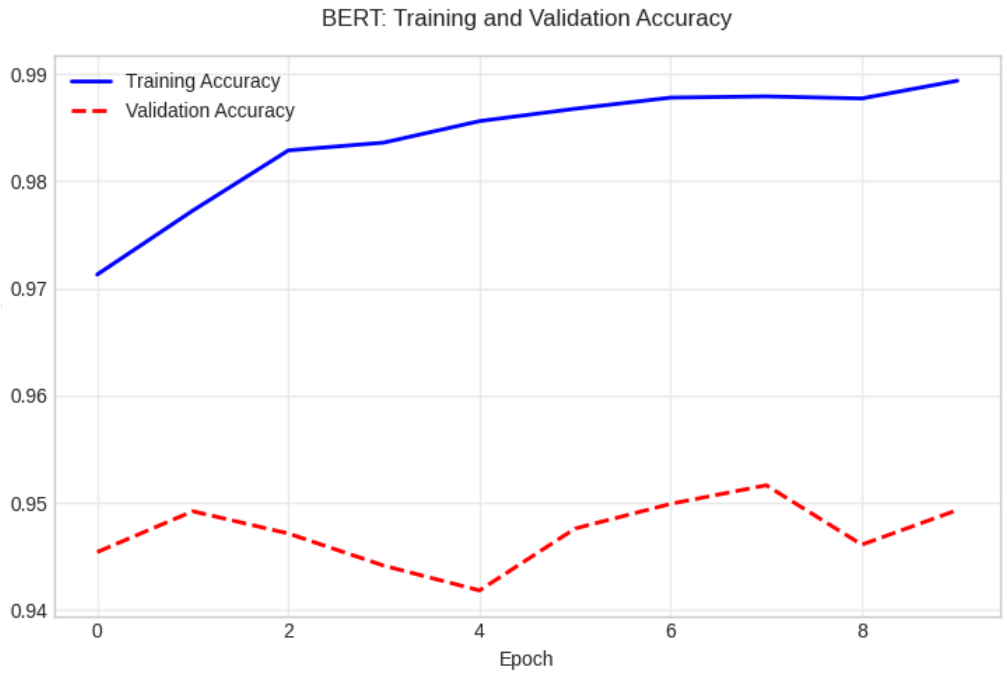

Total sampel yang diklasifikasikan salah mencatat 1311 sampel, di mana tingkat kesalahan klasifikasi ada pada 15.14% keseluruhan data. Sampel yang negatif diklasifikan sebagai 0 dan untuk positif memakai 1. Kesalahan ini bersifat wajar mengingat sentimen analisis mengambil dataset secara general di mana tidak semua perkataan akan mengandung makna yang sama jika dipadankan pada kalimat tertentu.

### 4.2 Evaluasi Model BERT

#### 4.2.1 Plot History Pelatihan

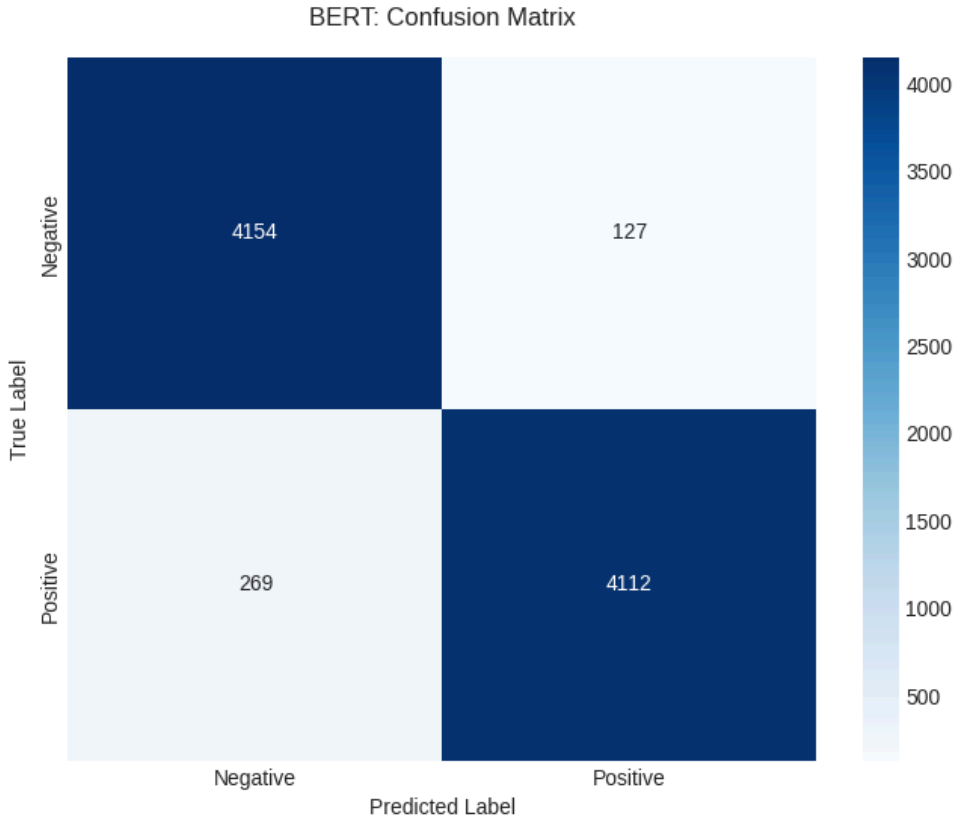
Model visualisasi ini merupakan salah satu jenis plotting yang menunjukkan bagaimana loss (fungsi kerugian) dan akurasi model berkembang selama setiap epoch pelatihan. Pada evaluasi ini terdapat 2 parameter utama yakni nilai ‘training’ dan juga ‘validation’ yang keduanya akan ditentukan dari status akurasi suatu model.




Dari visualisasi line chart di atas dapat disimpulkan bahwa model sudah cukup baik dalam mengeksekusi data. Training loss konstan mendekati 0 serta training akurasi mendekati 1 mengindikasikan bahwa BERT efektif dalam proses pengolahan sentimen. Meskipun, adanya sedikit perbedaan antara kinerja *training* dan *validation* mengindikasikan bahwa model mungkin saja mengalami sedikit overfitting yang dipengaruhi oleh beberapa faktor internal.

### 4.2.2 Confusion Matrix



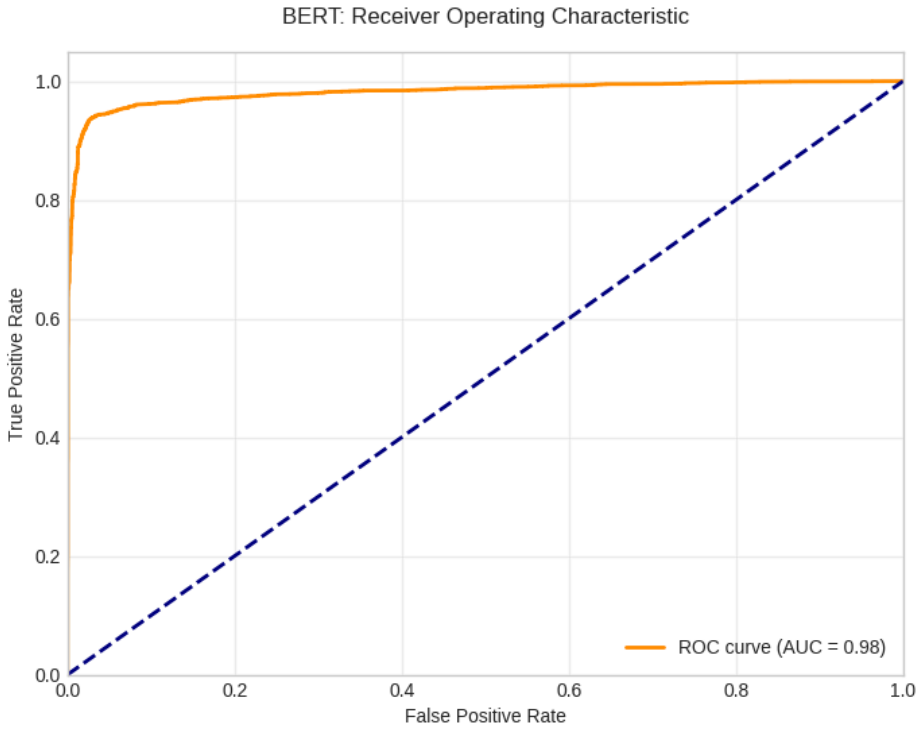

Berdasarkan hasil uji Naive bayes sebelumnya, dapat dilihat bahwa hasil confusion matrix BERT memiliki hasil distribusi yang jauh lebih baik. Data true yang sebelumnya berada di 3496 dan 3855, naik menjadi 4154 untuk negatif dan 4112 untuk positif. Hal ini membuktikan bahwa proses distribusi jauh lebih baik walau untuk Naive Bayes sendiri sudah bisa dikatakan baik juga.

### 4.2.3 Classification Report

```
Classification Report:
              precision    recall  f1-score   support

    Negative       0.94      0.97      0.95      4281
    Positive       0.97      0.94      0.95      4381

    accuracy                           0.95      8662
   macro avg       0.95      0.95      0.95      8662
weighted avg       0.95      0.95      0.95      8662
```

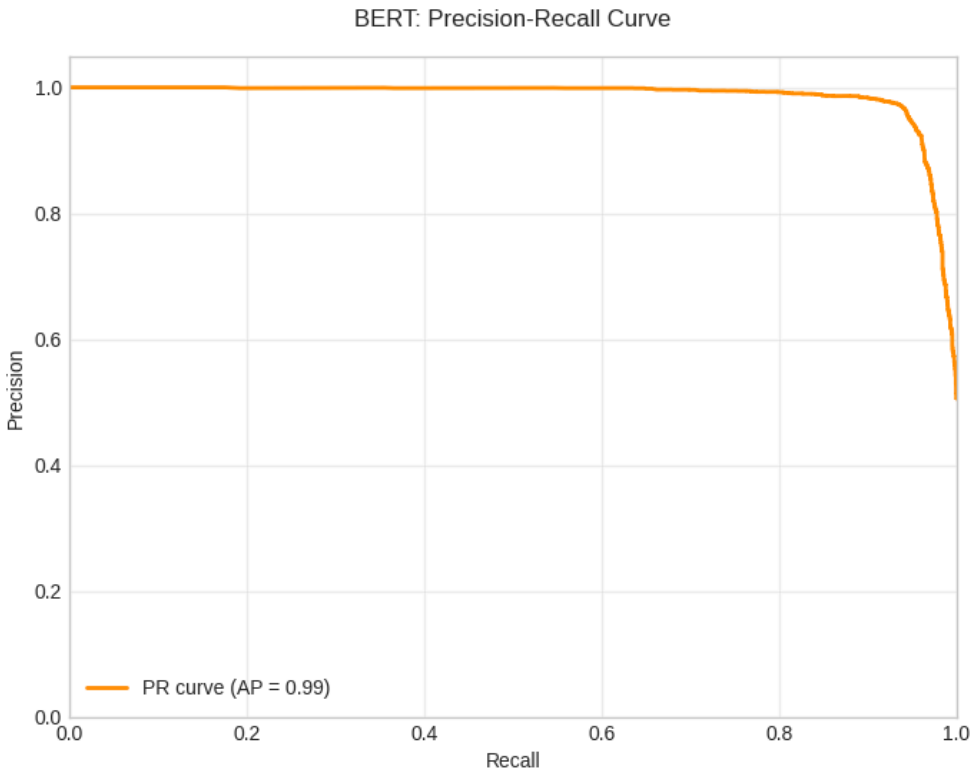
Untuk classification report juga sama, yakni hasil dari model BERT dapat dinilai perbedaan yang cukup signifikan dari pengujian sebelumnya. Hasil paling penting yaitu nilai akurasi berkembang dari yang sebelumnya 85% meningkat menjadi 95%. Hal itu juga sejalan dengan parameter lain seperti nilai precision, recall, dan juga f1-score.

### 4.2.4 Plot ROC Curve




Untuk ROC juga sudah pasti lebih baik di mana nilai dari AUC nya mencapai 0.98 yang dibanding dengan nilai sebelumnya hanya berkisar di 0.92

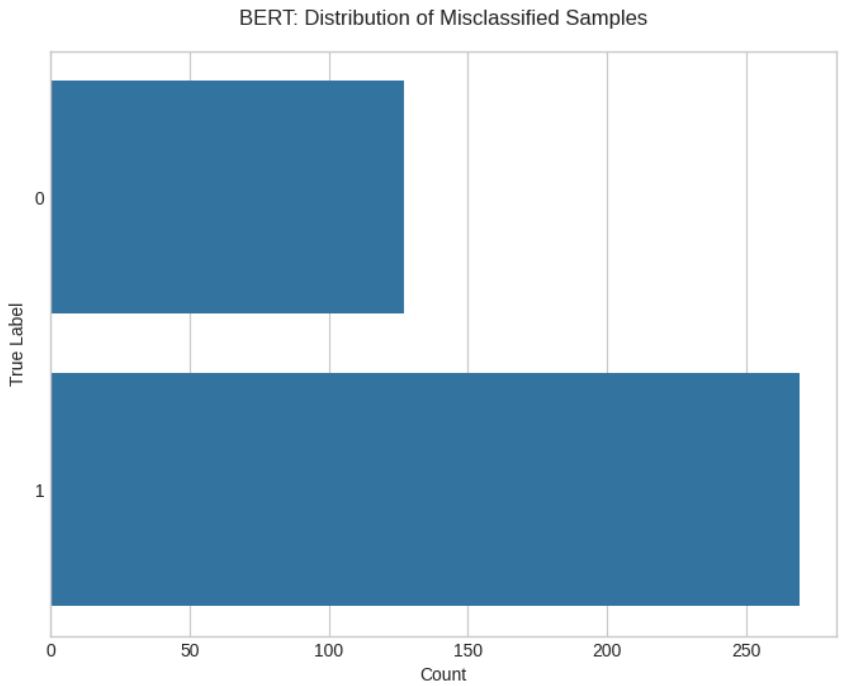
### 4.2.5 Precision-Recall Curve




Precision - recall juga meningkat dari yang sebelumnya 0.93 naik menjadi 0.99 Dengan begitu kemungkinan untuk hasil prediksi sampel berbeda semakin tidak ada.

### 4.2.6 Analisis Misclassifications

```
Total misclassified samples: 396

Misclassification rate: 4.57%
```




Pada misclassification, model BERT memberikan hasil yang jauh berbeda. Sampel misclassified turun dari yang sebelumnya di 15.14% dengan total 1311 sampel, turun menjadi 4.57% dengan total sampel mencapai di 396 sampel.

# BAB 5 KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Dalam analisis perbandingan model yang telah dilakukan, model BERT menunjukkan keunggulan yang sangat signifikan dibandingkan dengan model baseline Naive Bayes. Peningkatan performa terlihat jelas dari akurasi yang meningkat dari 85% menjadi 95%. Hal ini menunjukkan bahwa BERT mampu memahami konteks dan nuansa bahasa dengan jauh lebih baik. Tingkat misklasifikasi yang jauh lebih rendah pada BERT, yaitu hanya 4.57% dibandingkan dengan Naive Bayes yang mencapai 15.14%, membuktikan bahwa model ini lebih handal dalam mengklasifikasikan sentimen. Keseimbangan performa yang ditunjukkan melalui metrik evaluasi seperti precision, recall, dan F1-score untuk kedua kelas sentimen mengindikasikan bahwa model tidak bias terhadap kelas tertentu.

Berbicara tentang kualitas prediksi, model BERT memperlihatkan hasil yang sangat menjanjikan dengan tingkat precision yang tinggi untuk kedua sentimen. Pencapaian precision 94% untuk sentimen negatif dan 97% untuk sentimen positif menunjukkan kemampuan model dalam memberikan prediksi yang akurat. Nilai recall yang mencapai 97% untuk sentimen negatif dan 94% untuk sentimen positif mengindikasikan bahwa model mampu menangkap mayoritas kasus yang relevan. F1-score yang konsisten di angka 95% untuk kedua kelas memperkuat bukti bahwa model memiliki performa yang seimbang dan dapat diandalkan.

Dalam hal reliabilitas model, BERT menunjukkan peningkatan yang substansial dibandingkan dengan Naive Bayes. Kurva ROC mencapai nilai AUC 0.98, meningkat dari 0.92 pada model Naive Bayes, menandakan kemampuan diskriminatif yang sangat baik. Precision-Recall curve yang mencapai 0.99, naik dari 0.93, semakin menegaskan kehandalan model dalam memberikan prediksi yang akurat. Stabilitas model yang didukung oleh distribusi data yang seimbang memberikan keyakinan bahwa model ini dapat diandalkan untuk implementasi dalam skala yang lebih luas.

## 5.2 Saran

Untuk pengembangan sistem analisis sentimen ke depan, beberapa saran utama yang dapat diimplementasikan. Dari sisi teknis, diperlukan pembangunan sistem monitoring performa real-time dan fine-tuning berkala dengan data terbaru untuk memastikan model tetap relevan. Perluasan dataset yang mencakup lebih banyak variasi bahasa dan konteks juga menjadi prioritas untuk meningkatkan akurasi prediksi.

Dalam implementasi praktis, pengembangan dashboard monitoring sentiment real-time dengan sistem peringatan dini menjadi fokus utama. Hal ini perlu didukung dengan optimasi sistem untuk meningkatkan efisiensi waktu pemrosesan, terutama untuk analisis real-time, serta pengembangan API yang scalable.

Keberhasilan implementasi model BERT dalam analisis sentimen kandidat presiden Indonesia 2024 telah menunjukkan hasil yang sangat menjanjikan dengan tingkat akurasi dan reliabilitas yang tinggi. Dengan memperhatikan

saran-saran di atas, diharapkan sistem ini dapat terus dikembangkan untuk memberikan wawasan yang lebih mendalam dalam memahami sentimen publik terhadap dinamika politik di Indonesia, serta membuka peluang penerapan teknologi serupa dalam berbagai konteks analisis sentimen lainnya**.**