

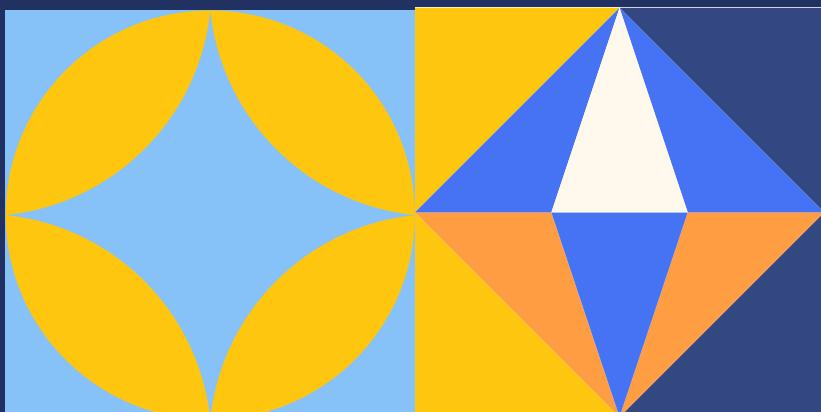
Data Mining

Analisis Sentimen Publik terhadap Isu Korupsi Proyek Kereta Cepat Whoosh Menggunakan TF-IDF dan IndoBERT



Anggota Kelompok

1. David Christian N (140810230027)
2. Hafizh Fadhl Muhammad (140810230070)
3. Farhan Zia Rizky (140810230074)
4. Gideon Tamba (140810230082)

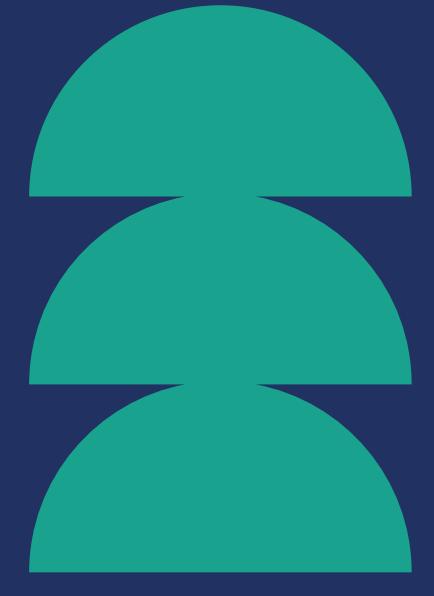


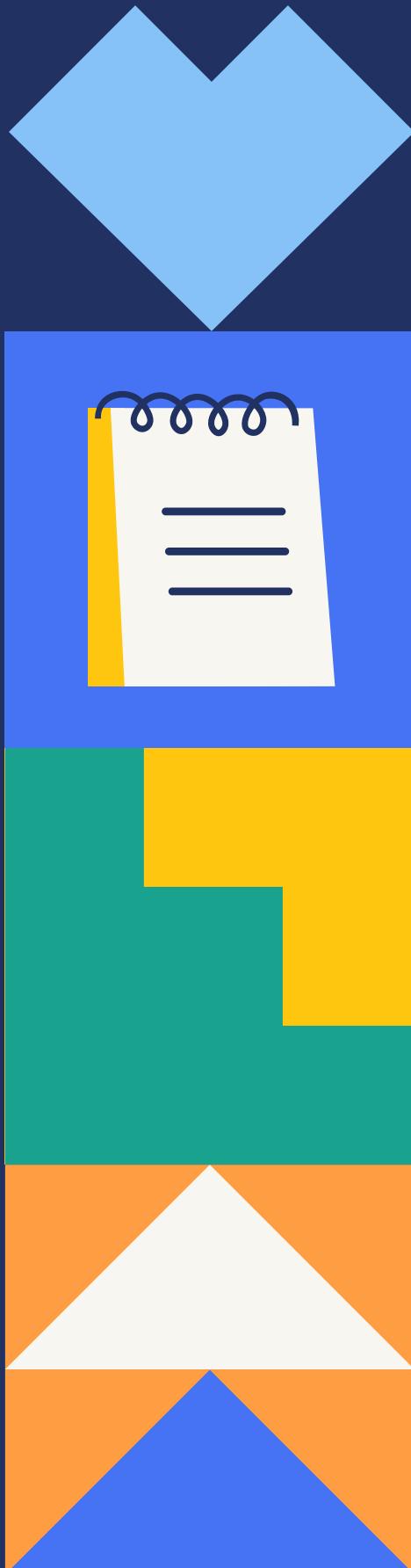


Latar Belakang

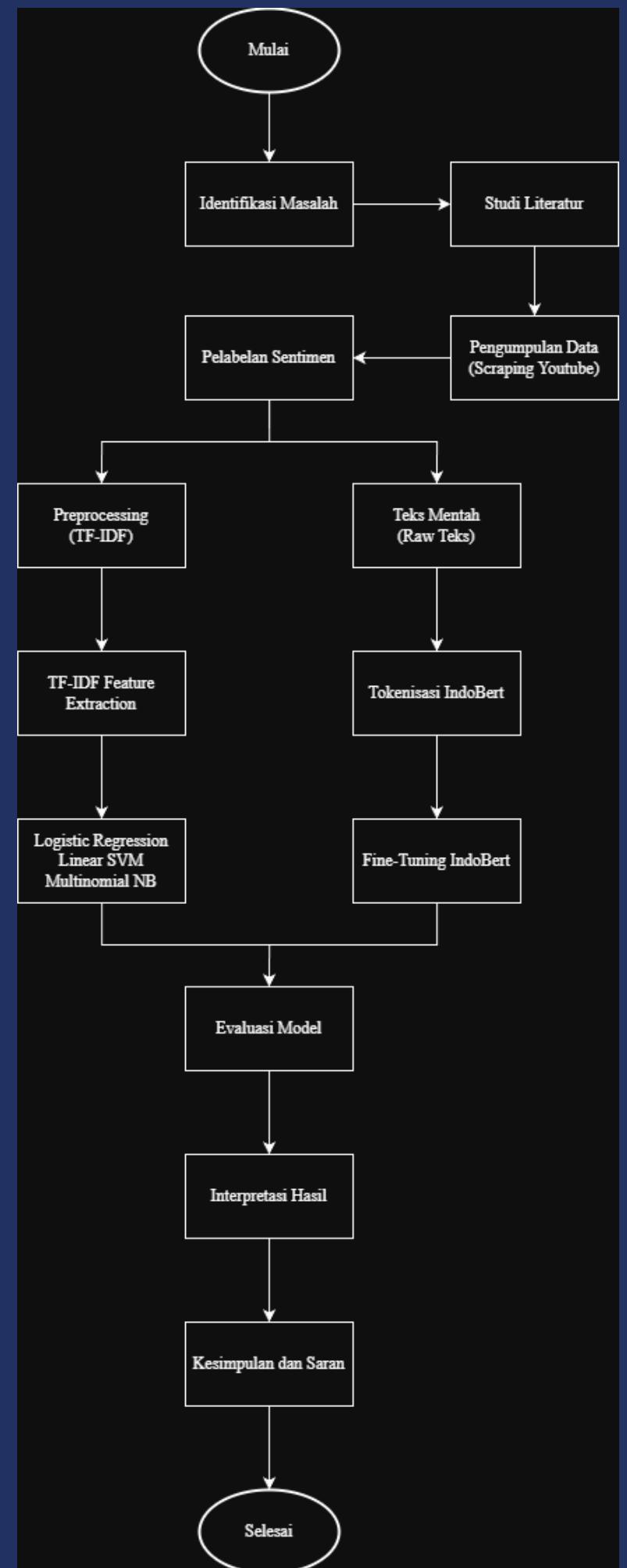
- Proyek Kereta Cepat Jakarta-Bandung (Whoosh) menuai beragam reaksi publik, mulai dari kebanggaan teknologi hingga kritik mengenai biaya dan hutang.
- Komentar di media sosial (YouTube) menjadi sumber data kaya untuk memahami persepsi masyarakat.

Tujuan Project

- Menganalisis sentimen publik terhadap isu korupsi pada proyek Kereta Cepat Whoosh melalui komentar YouTube yang telah di-scrape.
 - Dua pendekatan model digunakan:
 - TF-IDF + Machine Learning klasik (Logistic Regression, SVM, Naive Bayes)
 - Fine-tuning IndoBERT (transformer khusus Bahasa Indonesia)
 - Hasil kedua metode dibandingkan untuk melihat mana yang lebih efektif pada dataset berukuran kecil dan berlabel imbalance.
- 



Tahapan Penggerjaan



Scraping

Library yang dipakai

```
● ● ●  
import os  
import time  
import csv  
  
from googleapiclient.discovery import build  
from googleapiclient.errors import HttpError  
from dotenv import load_dotenv  
import pandas as pd
```

Set Up dan Autentikasi

```
● ● ●  
load_dotenv()  
API_KEY = os.getenv("YOUTUBE_API_KEY")  
  
if API_KEY is None:  
    raise ValueError(  
        "YOUTUBE_API_KEY tidak ditemukan.\n"  
        "Pastikan kamu sudah membuat file .env dan mengisinya seperti:\n"  
        "YOUTUBE_API_KEY=API_KEY_KAMU"  
)
```

Request API

```
● ● ●  
response = youtube.commentThreads().list(  
    part="snippet",  
    videoId=video_id,  
    maxResults=request_count,  
    textFormat="plainText",  
    pageToken=next_page_token  
).execute()
```

Yang Akan Menjadi Dataset

```
● ● ●  
comments.append({  
    "video_id": video_id,  
    "video_title": video_title,  
    "comment_id": comment_id,  
    "author": author,  
    "comment": comment_text,  
    "likes": like_count,  
    "published_at": published_at  
})
```

Menyimpan dalam Format CSV

```
● ● ●

for i, vid in enumerate(video_ids):
    print(f"[{i+1}/{len(video_ids)}] Video ID: {vid}")

    title = get_video_title(vid)
    print(f" -> Judul: {title}")

    comments = get_video_comments(vid, title, MAX_COMMENTS_PER_VIDEO)
    all_comments.extend(comments)

    time.sleep(0.5)

    print("-" * 50)
    print(f"Scraping selesai. Total komentar terkumpul: {len(all_comments)}")

    save_comments_to_csv(all_comments, output_filename)
```

Labelling

Library yang dipakai



```
import os
import re
import time
import glob
import pandas as pd
from google import genai
from dotenv import load_dotenv
from tqdm import tqdm
```

Func



```
def classify_sentiment_batch(comments):
    """
    Klasifikasi sentimen untuk batch komentar sekaligus.
    """
    numbered_comments = "\n".join([f"{i+1}. {c}" for i, c in enumerate(comments)])
    prompt = f"""
```

Pedoman Penilaian



Pedoman penilaian:

1. **Positive**

- Mendukung, membela, atau tidak percaya bahwa ada korupsi.
- Menganggap isu korupsi tidak benar, dilebih-lebihkan, atau ada pihak yang menyebarkan hoaks.
- Menilai proyek berjalan baik dan tidak berkaitan dengan korupsi.

2. **Negative**

- Menyatakan bahwa proyek Whoosh memang korup, merugikan negara, penuh penyimpangan.
- Menyalahkan pemerintah, pejabat, atau pihak tertentu terkait dugaan korupsi proyek tersebut.
- Mengkritik biaya, pembengkakan anggaran, atau tuduhan penyalahgunaan dana.

3. **Neutral**

- Tidak menunjukkan opini jelas.
- Hanya bertanya, bercanda, atau menceritakan informasi umum.
- Komentar tidak relevan dengan isu korupsi.

Jumlah komentar: {len(comments)}.

Berikut daftar komentar yang harus Anda klasifikasikan sesuai urutan:

{numbered_comments}

Func



```
response = client.models.generate_content(
    model="gemini-2.0-flash",
    contents=prompt
)

raw_output = response.text.strip()

# Bersihkan format markdown jika ada
raw_output = re.sub(r"^\``(?:python)?", "", raw_output)
raw_output = re.sub(r"\`$", "", raw_output).strip()

# Parsing hasil
try:
    labels = re.findall(r'".*?"', raw_output)
    if not labels: # Jika tidak ada tanda kutip, coba pisah berdasarkan koma
        labels = [w.strip(" []'\n") for w in raw_output.split(",") if w.strip()]

    # Validasi jumlah label
    if len(labels) != len(comments):
        print(f"    [WARNING] Jumlah label ({len(labels)}) ≠ jumlah komentar ({len(comments)})")
        if len(labels) < len(comments):
            labels += ["Netral"] * (len(comments) - len(labels))
        else:
            labels = labels[:len(comments)]

return labels
```

Preprocessing

Flowchart

- 1 Text Cleaning**
Menghapus emoji, video id, hashtag, tanda baca, dan angka.
- 2 Case Folding**
Mengubah seluruh teks menjadi huruf kecil (lowercase).
- 3 Normalisasi Kata**
Mengubah kata slang menjadi baku (contoh: "yg" -> "yang").
- 4 Tokenization**
Memecah kalimat menjadi deretan kata (token) berdasarkan spasi.
- 5 Stopword Removal**
Menghapus kata umum yang tidak memiliki makna analitis
- 6 Stemming**
Mengubah kata berimbuhan menjadi kata dasar menggunakan stemmer Bahasa Indonesia



Text Cleaning

Pada tahap ini, kita melakukan proses text cleaning menggunakan Python.

Pertama, kita menghapus URL, mention username, dan hashtag, karena elemen tersebut tidak memberikan makna dalam analisis teks. Selanjutnya, kita menghapus tag HTML, emoji, dan angka untuk memastikan teks yang tersisa benar-benar berupa kata. Setelah itu, kita gunakan fungsi `remove_symbols`, yang berfungsi untuk menyisakan hanya huruf dan spasi saja.

```
def remove_URL(tweet):
    if tweet is not None and isinstance(tweet, str):
        url = re.compile(r'https?://\S+|www\.\S+')
        return url.sub(r'', tweet)
    else:
        return tweet

# Fungsi untuk menghapus HTML
def remove_html(tweet):
    if tweet is not None and isinstance(tweet, str):
        html = re.compile(r'<.*?>')
        return html.sub(r'', tweet)
    else:
        return tweet

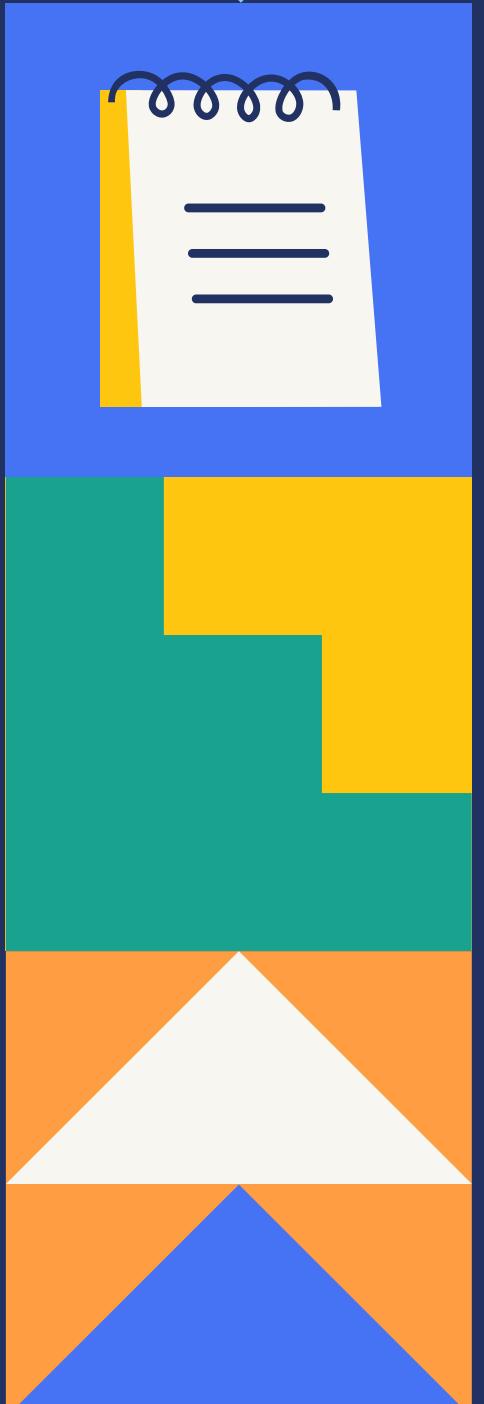
# Fungsi untuk menghapus emoji
def remove_emoji(tweet):
    if tweet is not None and isinstance(tweet, str):
        emoji_pattern = re.compile("["
            u"\U0001F600-\U0001F64F" # emoticons
            u"\U0001F300-\U0001F5FF" # symbols & pictographs
            u"\U0001F680-\U0001F6FF" # transport & map symbols
            u"\U0001F700-\U0001F77F" # alchemical symbols
            u"\U0001F780-\U0001F7FF" # Geometric Shapes Extended
            u"\U0001F800-\U0001F8FF" # Supplemental Arrows-C
            u"\U0001F900-\U0001F9FF" # Supplemental Symbols and Pictographs
            u"\U0001FA00-\U0001FA6F" # Chess Symbols
            u"\U0001FA70-\U0001FAFF" # Symbols and Pictographs Extended-A
            u"\U0001F004-\U0001F0CF" # Additional emoticons
            u"\U0001F1E0-\U0001F1FF" # flags
            "[]+", flags=re.UNICODE)
        return emoji_pattern.sub(r'', tweet)
    else:
        return tweet

# Fungsi untuk menghapus simbol
def remove_symbols(tweet):
    if tweet is not None and isinstance(tweet, str):
        tweet = re.sub(r'[^a-zA-Z0-9\s]', '', tweet)
    return tweet

# Fungsi untuk menghapus angka
def remove_numbers(tweet):
    if tweet is not None and isinstance(tweet, str):
        tweet = re.sub(r'\d', '', tweet)
    return tweet

# Fungsi hapus username
def remove_usernames(text):
    return re.sub(r'@\w+', '', text)
```

Case Folding



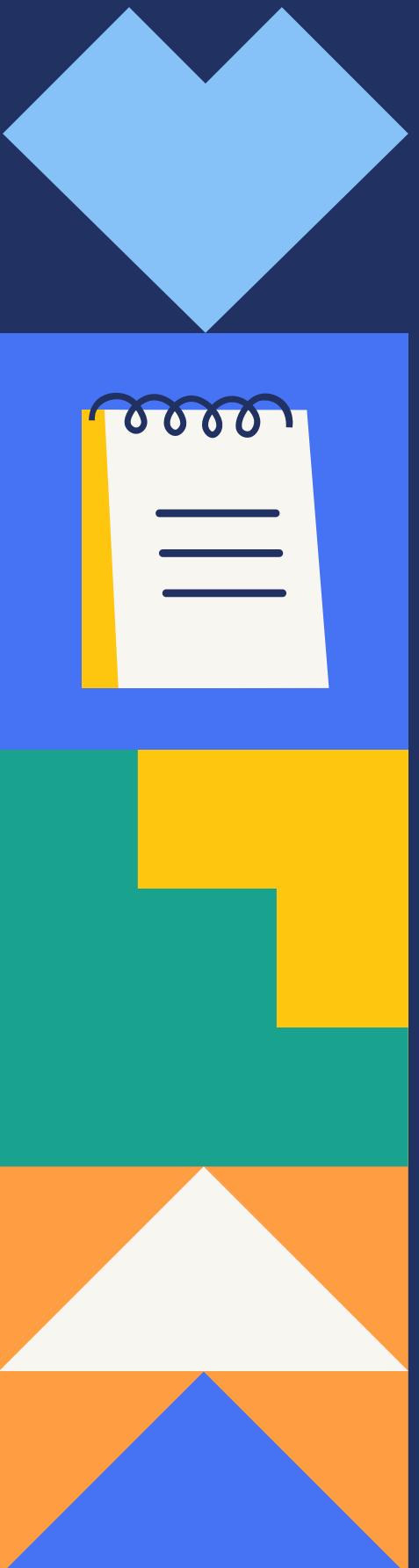
```
def case_folding(text):
    if isinstance(text, str):
        lowercase_text = text.lower()
        return lowercase_text
    else:
        return text

df['case_folding'] = df['cleaning'].apply(case_folding)

df = df[['comment', 'cleaning', 'case_folding', 'sentiment', 'label']]

df.head(5)
```

	comment	cleaning	case_folding
0	Yg benci ya apa aja salah.. \nYg seneng ya mak...	Yg benci ya apa aja salah \nYg seneng ya makin...	yg benci ya apa aja salah \nyg seneng ya makin...
1	Bandung akan miliki kereta pajajaran dgn beaya...	Bandung akan miliki kereta pajajaran dgn beaya...	bandung akan miliki kereta pajajaran dgn beaya...
2	SUDAH JELAS GENG SOLO YANG HARUS BERTANGGUNG J...	SUDAH JELAS GENG SOLO YANG HARUS BERTANGGUNG J...	sudah jelas geng solo yang harus bertanggung j...
3	Jokowi, Luhut, kroni2 yg harus bertanggungjawab ...	Jokowi Luhut kroni yg harus bertanggungjawab ...	jokowi luhut kroni yg harus bertanggungjawab ...
4	Yg ditangkap gorengan yg makan duduk manis	Yg ditangkap gorengan yg makan duduk manis	yg ditangkap gorengan yg makan duduk manis



Normalisasi

```
● ● ●

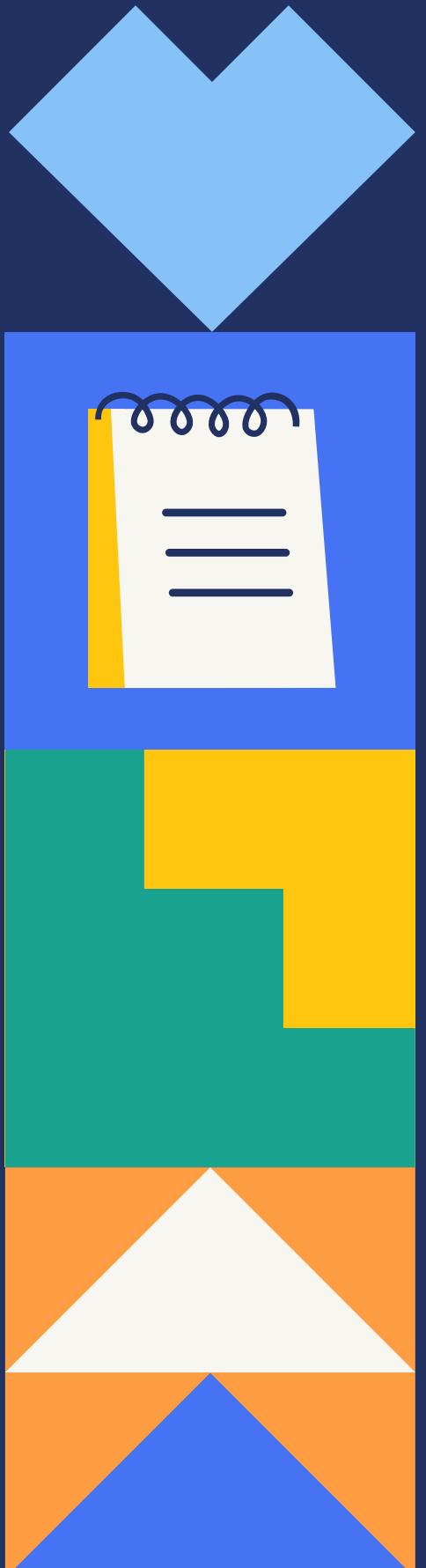
def replace_taboo_words(text, kamus_tidak_baku):
    if isinstance(text, str):
        words = text.split()
        replaced_words = []
        kalimat_baku = []
        kata_diganti = []
        kata_tidak_baku_hash = []

        for word in words:
            if word in kamus_tidak_baku:
                baku_word = kamus_tidak_baku[word]
                if isinstance(baku_word, str) and all(char.isalpha() for char in baku_word):
                    replaced_words.append(baku_word)
                    kalimat_baku.append(baku_word)
                    kata_diganti.append(word)
                    kata_tidak_baku_hash.append(hash(word))
            else:
                replaced_words.append(word)
        replaced_text = ' '.join(replaced_words)
    else:
        replaced_text = ''
        kalimat_baku = []
        kata_diganti = []
        kata_tidak_baku_hash = []

    return replaced_text, kalimat_baku, kata_diganti, kata_tidak_baku_hash

# Baca dataset kamu (pastikan df sudah tersedia)
data = pd.DataFrame(df[['comment','cleaning','case_folding', 'sentiment', 'label']])
data.head()
```

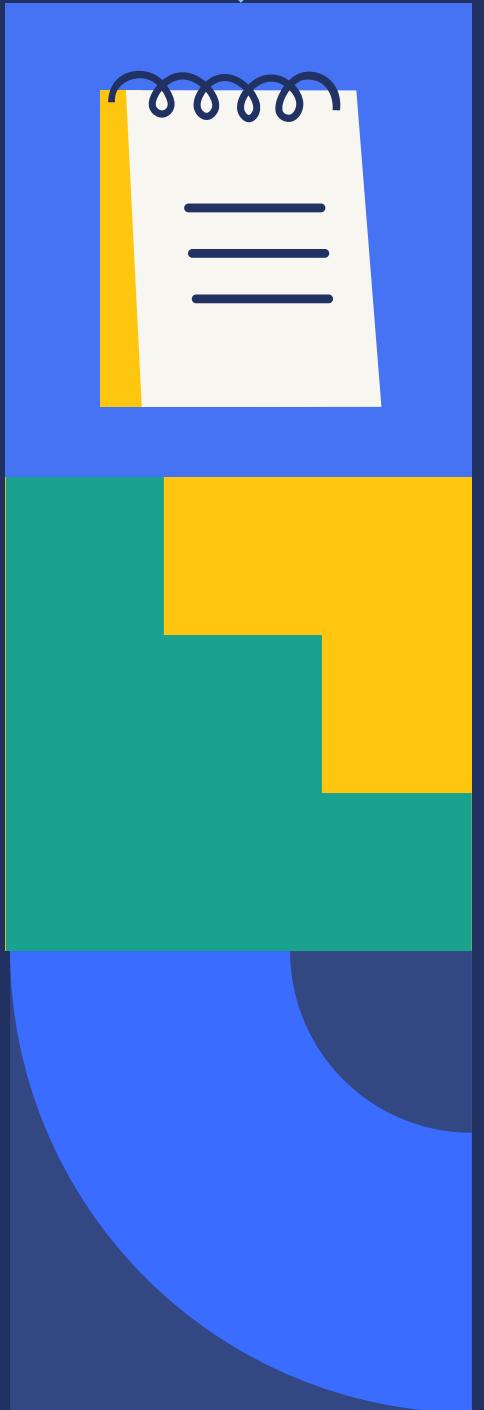
Tokenisasi



```
● ● ●  
def tokenize(text):  
    tokens = text.split()  
    return tokens  
  
# Tambah kolom tokenize ke df  
df['tokenize'] = df['normalisasi'].apply(tokenize)  
  
# Ambil kolom yang relevan - gunakan 'df' bukan 'data'  
df = df[['comment', 'cleaning', 'case_folding', 'normalisasi', 'tokenize', 'sentiment', 'label']]  
  
df.head(5)
```

	comment	cleaning	case_folding	normalisasi	tokenize
0	Yg benci ya apa aja salah.. \nYg seneng ya mak...	Yg benci ya apa aja salah \nYg seneng ya makin...	yg benci ya apa aja salah \n yg seneng ya makin...	yang benci ya apa saja salah yang senang ya ma...	[yang, benci, ya, apa, saja, salah, yang, sena...
1	Bandung akan miliki kereta pajajaran dgn beaya...	Bandung akan miliki kereta pajajaran dgn beaya...	bandung akan miliki kereta pajajaran dgn beaya...	bandung akan miliki kereta pajajaran dengan be...	[bandung, akan, miliki, kereta, pajajaran, den...
2	SUDAH JELAS GENG SOLO YANG HARUS BERTANGGUNG J...	SUDAH JELAS GENG SOLO YANG HARUS BERTANGGUNG J...	sudah jelas geng solo yang harus bertanggung j...	sudah jelas geng solo yang harus bertanggung j...	[sudah, jelas, geng, solo, yang, harus, bertan...
3	Jokowi, Luhut, kroni2 yg harus bertanggungjawab...	Jokowi Luhut kroni yg harus bertanggungjawab ...	jokowi luhut kroni yg harus bertanggungjawab ...	jokowi luhut kroni yang harus bertanggungjawab...	[jokowi, luhut, kroni, yang, harus, bertanggun...
4	Yg ditangkap gorengan yg makan duduk manis	Yg ditangkap gorengan yg makan duduk manis	yg ditangkap gorengan yg makan duduk manis	yang ditangkap gorengan yang makan duduk manis	[yang, ditangkap, gorengan, yang, makan, dudu...

Stopword Removal



```
● ● ●  
nltk.download('stopwords')  
stop_words = stopwords.words('indonesian')
```



```
● ● ●  
def remove_stopwords(text):  
    return [word for word in text if word not in stop_words]  
  
df['stopword removal'] = df['tokenize'].apply(lambda x: remove_stopwords(x))  
  
# Reorder kolom - tambahkan 'stopword removal' setelah 'tokenize'  
df = df[['comment', 'cleaning', 'case_folding', 'normalisasi', 'tokenize', 'stopword removal', 'sentiment', 'label']]  
  
df.head(5)
```

	comment	cleaning	case_folding	normalisasi	tokenize	stopword removal
0	Yg benci ya apa aja salah.. \nYg seneng ya mak...	Yg benci ya apa aja salah \nYg seneng ya makin...	yg benci ya apa aja salah \nnyg seneng ya makin...	yang benci ya apa saja salah yang senang ya ma...	[yang, benci, ya, apa, saja, salah, yang, sena...]	[benci, ya, salah, senang, ya, senang, nyinyir...]
1	Bandung akan miliki kereta pajajaran dgn beaya...	Bandung akan miliki kereta pajajaran dgn beaya...	bandung akan miliki kereta pajajaran dgn beaya...	bandung akan miliki kereta pajajaran dengan be...	[bandung, akan, miliki, kereta, pajajaran, den...]	[bandung, miliki, kereta, pajajaran, beaya, mu...]
2	SUDAH JELAS GENG SOLO YANG HARUS BERTANGGUNG J...	SUDAH JELAS GENG SOLO YANG HARUS BERTANGGUNG J...	sudah jelas geng solo yang harus bertanggung j...	sudah jelas geng solo yang harus bertanggung j...	[sudah, jelas, geng, solo, yang, harus, bertan...]	[geng, solo, bertanggung, tangkap]

Stemming

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

def stem_text(text):
    return [stemmer.stem(word) for word in text]

df['stemming_data'] = df['stopword removal'].apply(lambda x: ' '.join(stem_text(x)))

df = df[['comment', 'cleaning', 'case_folding', 'normalisasi', 'tokenize', 'stopword removal', 'stemming_data', 'sentiment', 'label']]

df.head(5)
```

	comment	cleaning	case_folding	normalisasi	tokenize	stopword removal	stemming_data
0	Yg benci ya apa aja salah.. \nYg seneng ya mak...	Yg benci ya apa aja salah \nYg seneng ya makin...	yg benci ya apa aja salah \nyg seneng ya makin...	yang benci ya apa saja salah yang senang ya ma...	[yang, benci, ya, apa, saja, salah, yang, sena...	[benci, ya, salah, senang, ya, senang, nyinyir...	benci ya salah senang ya senang nyinyir ya bah...
1	Bandung akan miliki kereta pajajaran dgn beaya...	Bandung akan miliki kereta pajajaran dgn beaya...	bandung akan miliki kereta pajajaran dengan beaya...	bandung akan miliki kereta pajajaran dengan be...	[bandung, akan, miliki, kereta, pajajaran, den...	[bandung, miliki, kereta, pajajaran, beaya, mu...	bandung milik kereta pajajaran beaya murah who...
2	SUDAH JELAS GENG SOLO YANG HARUS BERTANGGUNG J...	SUDAH JELAS GENG SOLO YANG HARUS BERTANGGUNG J...	sudah jelas geng solo yang harus bertanggung j...	sudah jelas geng solo yang harus bertanggung j...	[sudah, jelas, geng, solo, yang, harus, bertan...	[geng, solo, bertanggung, tangkap]	geng solo tanggung tangkap

Stemming

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

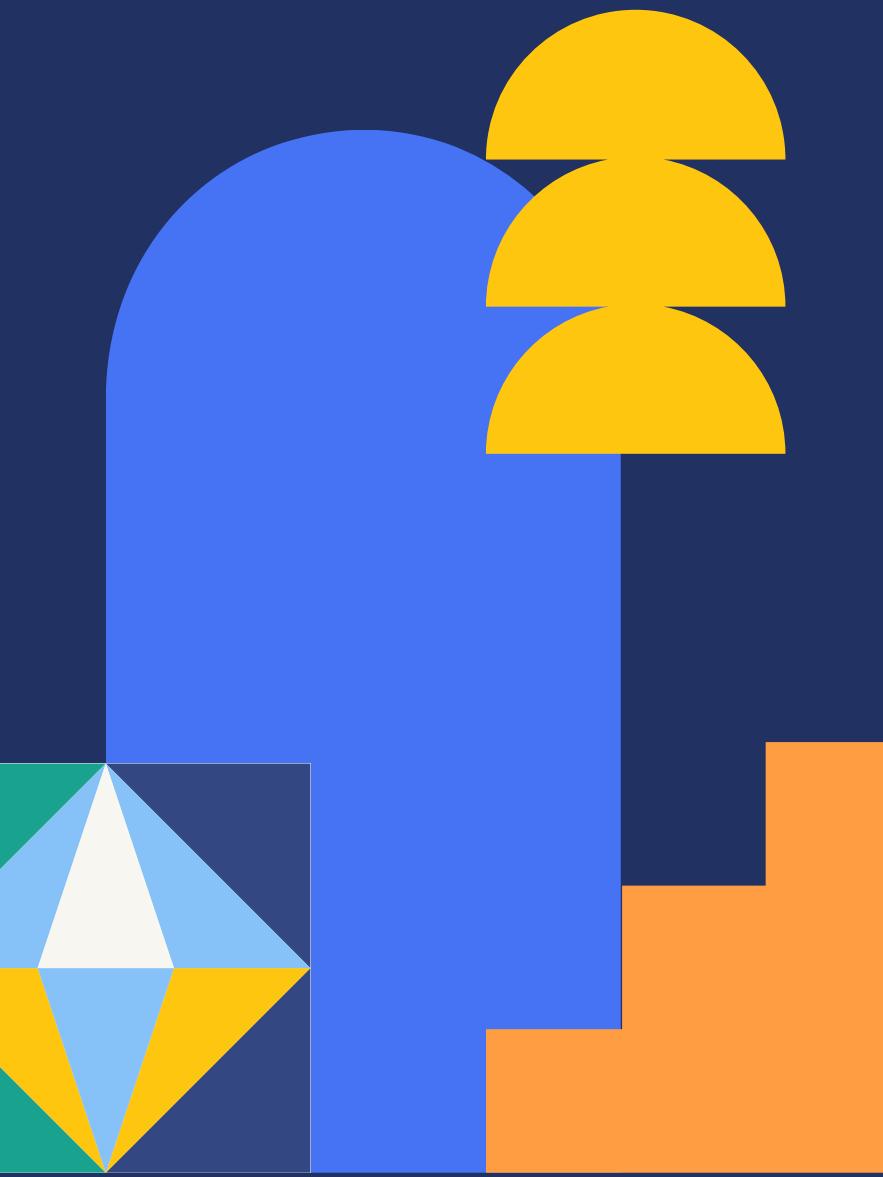
def stem_text(text):
    return [stemmer.stem(word) for word in text]

df['stemming_data'] = df['stopword removal'].apply(lambda x: ' '.join(stem_text(x)))

df = df[['comment', 'cleaning', 'case_folding', 'normalisasi', 'tokenize', 'stopword removal', 'stemming_data', 'sentiment', 'label']]

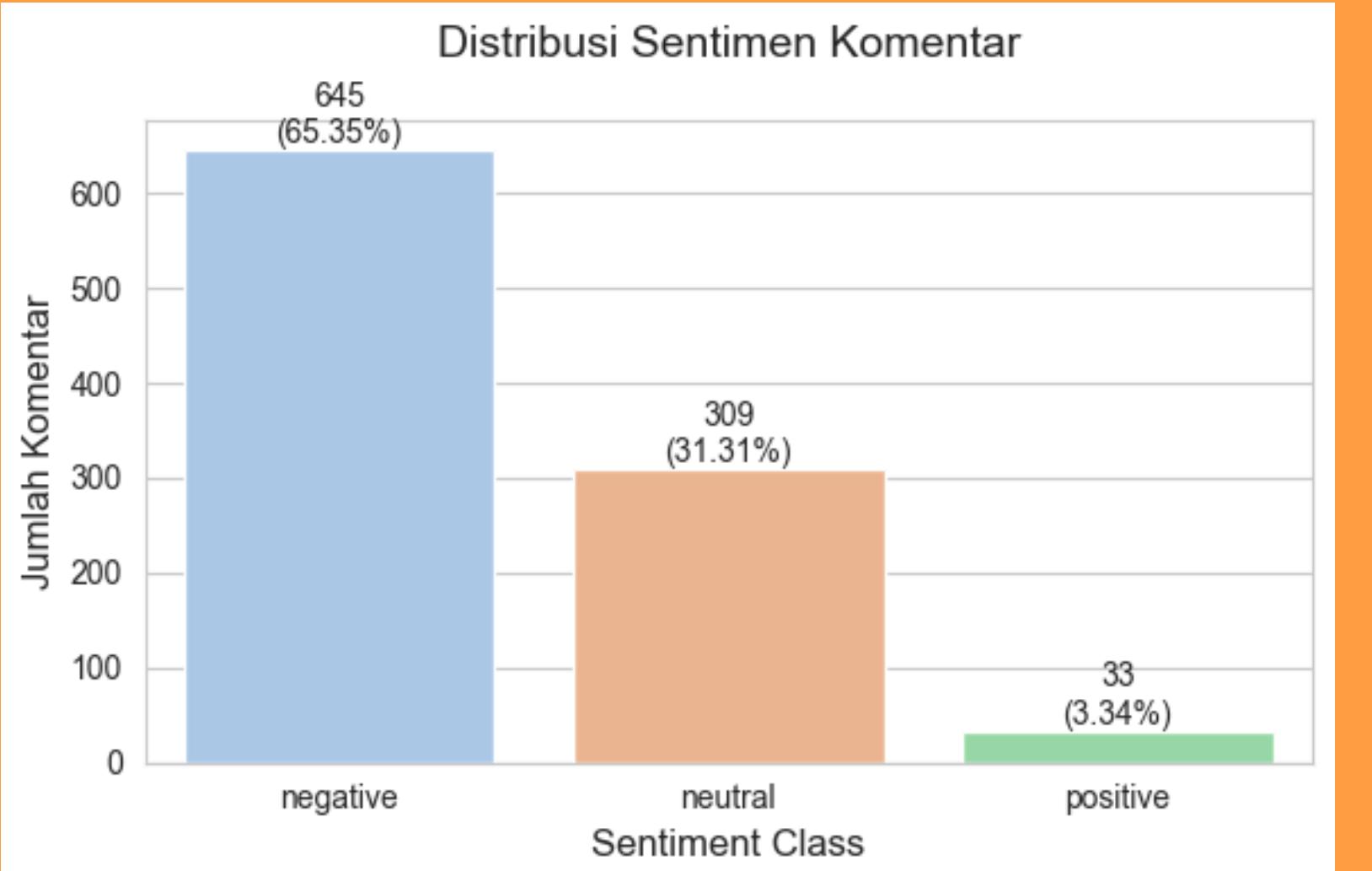
df.head(5)
```

	comment	cleaning	case_folding	normalisasi	tokenize	stopword removal	stemming_data
0	Yg benci ya apa aja salah.. \nYg seneng ya mak...	Yg benci ya apa aja salah \nYg seneng ya makin...	yg benci ya apa aja salah \nyg seneng ya makin...	yang benci ya apa saja salah yang senang ya ma...	[yang, benci, ya, apa, saja, salah, yang, sena...	[benci, ya, salah, senang, ya, senang, nyinyir...	benci ya salah senang ya senang nyinyir ya bah...
1	Bandung akan miliki kereta pajajaran dgn beaya...	Bandung akan miliki kereta pajajaran dgn beaya...	bandung akan miliki kereta pajajaran dengan beaya...	bandung akan miliki kereta pajajaran dengan be...	[bandung, akan, miliki, kereta, pajajaran, den...	[bandung, miliki, kereta, pajajaran, beaya, mu...	bandung milik kereta pajajaran beaya murah who...
2	SUDAH JELAS GENG SOLO YANG HARUS BERTANGGUNG J...	SUDAH JELAS GENG SOLO YANG HARUS BERTANGGUNG J...	sudah jelas geng solo yang harus bertanggung j...	sudah jelas geng solo yang harus bertanggung j...	[sudah, jelas, geng, solo, yang, harus, bertan...	[geng, solo, bertanggung, tangkap]	geng solo tanggung tangkap

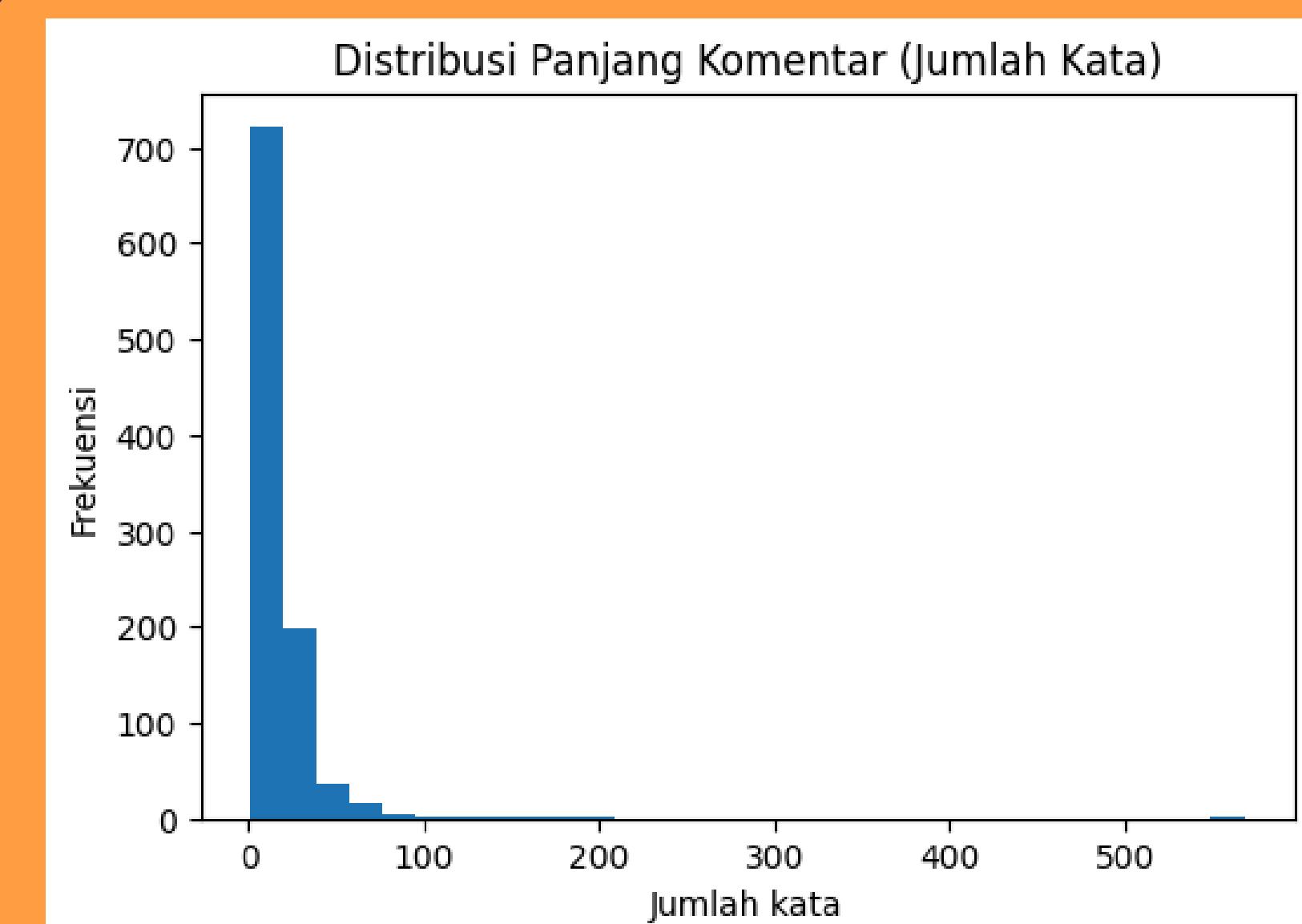


EDA

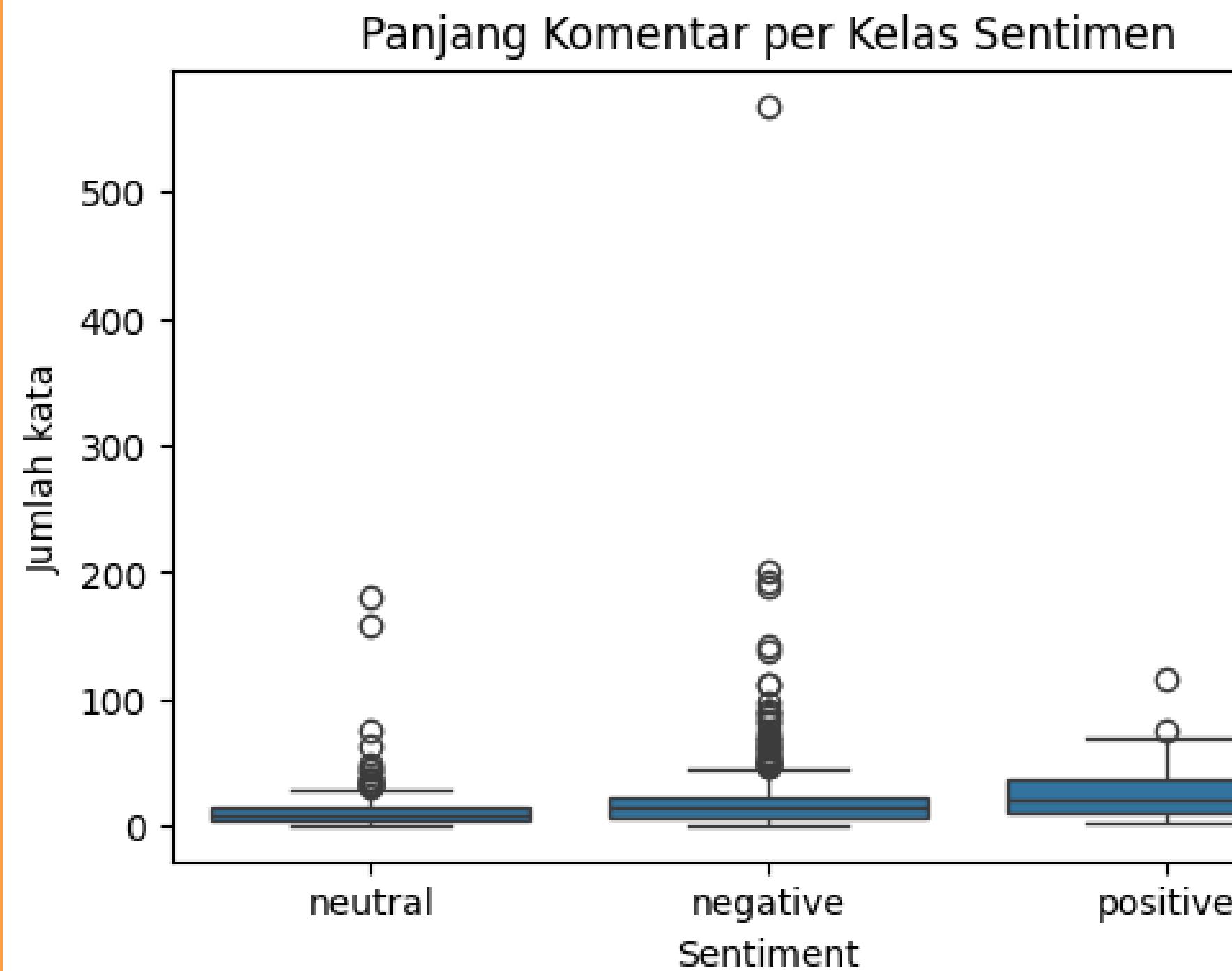
Distribusi Sentimen



Distribusi Panjang Komentar



Panjang Komentar



WordCloud

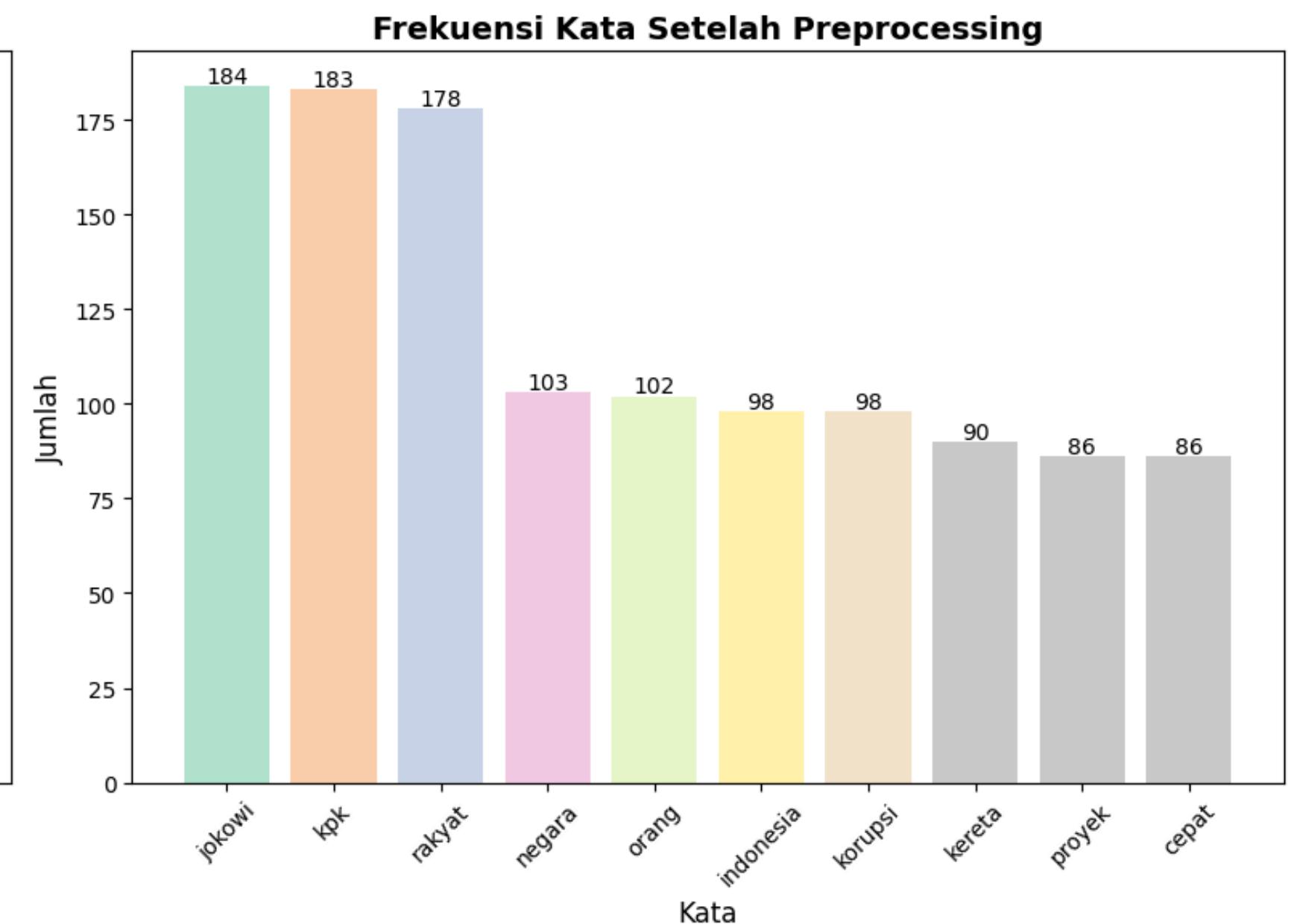
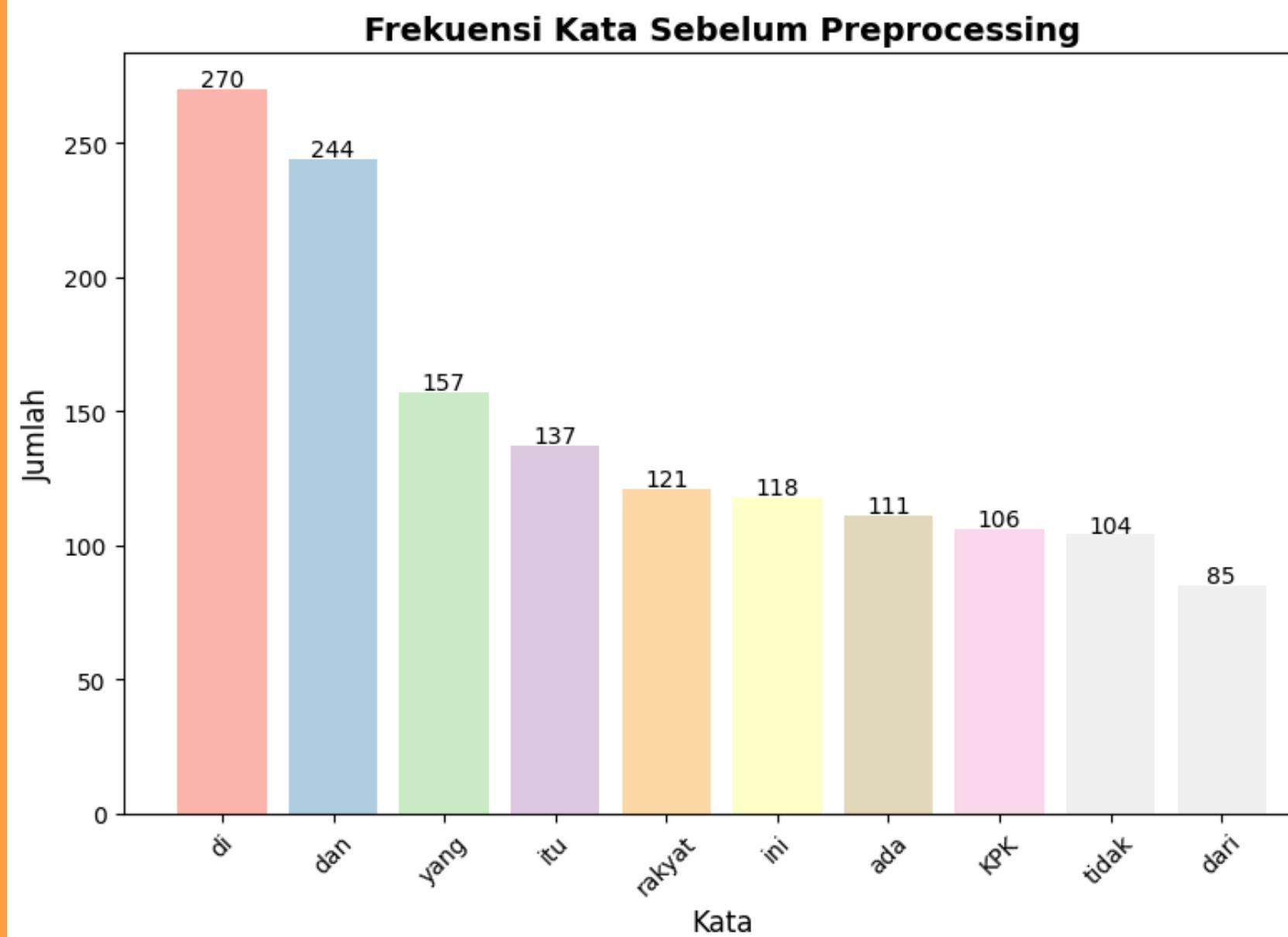
Before Preprocessing



After Preprocessing



Frekuensi Kata Sebelum dan Sesudah Preprocessing



Modeling : Baseline



TF-IDF Vectorizer



```
tfidf = TfidfVectorizer(  
    ngram_range=(1, 2),      # unigram + bigram  
    min_df=3,  
    max_features=10000,  
    sublinear_tf=True  
)  
  
X_train_tfidf = tfidf.fit_transform(X_train)  
X_test_tfidf = tfidf.transform(X_test)  
  
X_train_tfidf.shape, X_test_tfidf.shape
```

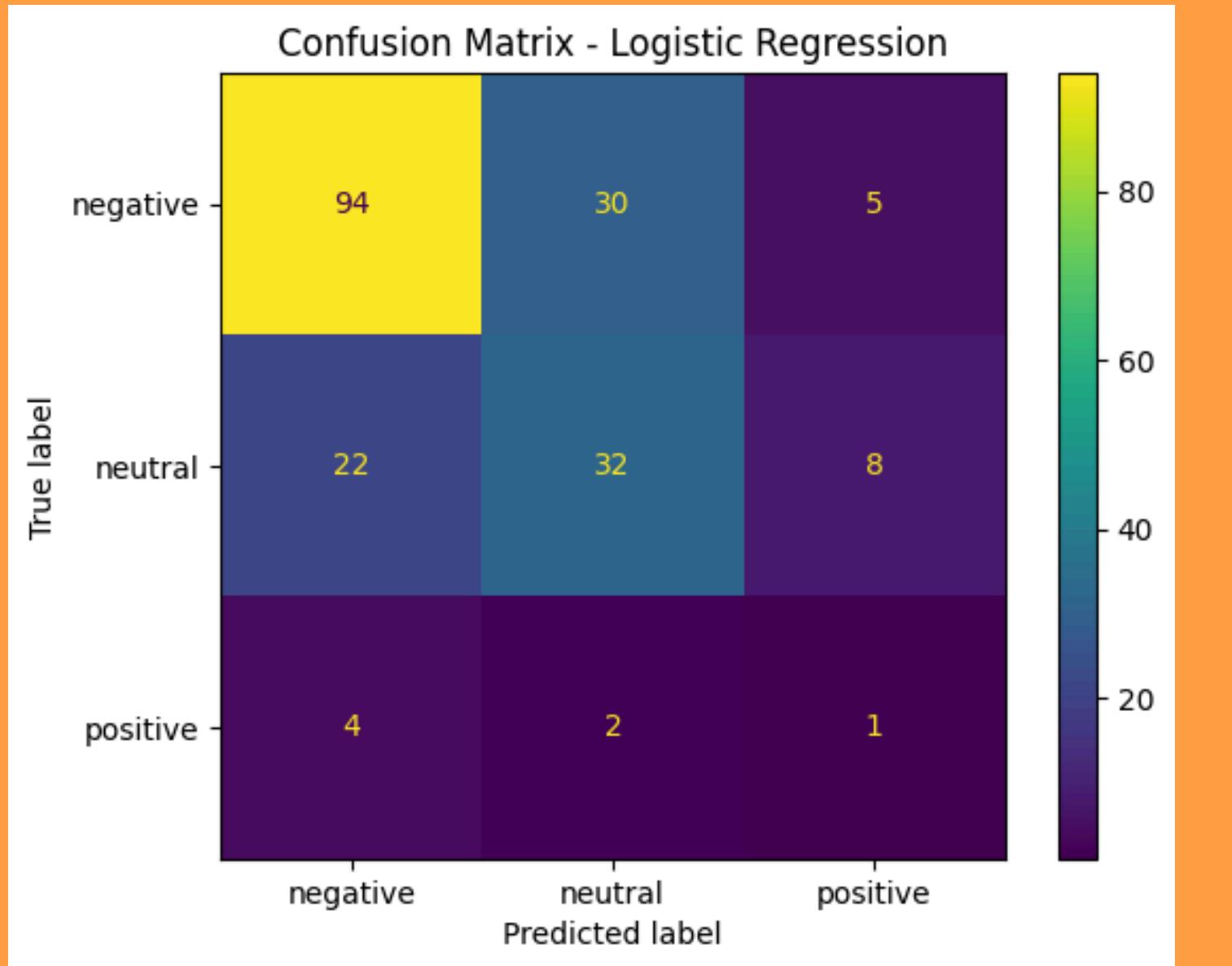
Train 3 Model



```
models = {}  
  
# 1. Logistic Regression  
logreg = LogisticRegression(  
    max_iter=500,  
    class_weight="balanced",  
    solver="liblinear"  
)  
logreg.fit(X_train_resampled, y_train_resampled)  
models["Logistic Regression"] = logreg  
  
# 2. Linear SVM  
svm = LinearSVC(  
    class_weight="balanced",  
    random_state=42  
)  
svm.fit(X_train_resampled, y_train_resampled)  
models["Linear SVM"] = svm  
  
# 3. Multinomial Naive Bayes  
mnb = MultinomialNB()  
mnb.fit(X_train_resampled, y_train_resampled)  
models["Multinomial NB"] = mnb
```

Confusion Matrix Model Terbaik

func Prediksi



```
def prediksi_sentimen(teks):
    vector = tfidf.transform([teks])
    pred = clf.predict(vector)[0]
    prob = clf.predict_proba(vector)[0]

    print(f"Teks: {teks}")
    print(f"Prediksi: {pred}")
    print(f"Probabilitas: {dict(zip(classes, prob))}")
    print("-" * 50)
```

```
prediksi_sentimen("kereta cepat whoosh keren banget")
prediksi_sentimen("ini proyek yang sangat merugikan negara")
prediksi_sentimen("biasa aja sih, ga terlalu relevan")
prediksi_sentimen("keren banget whoosh")
```

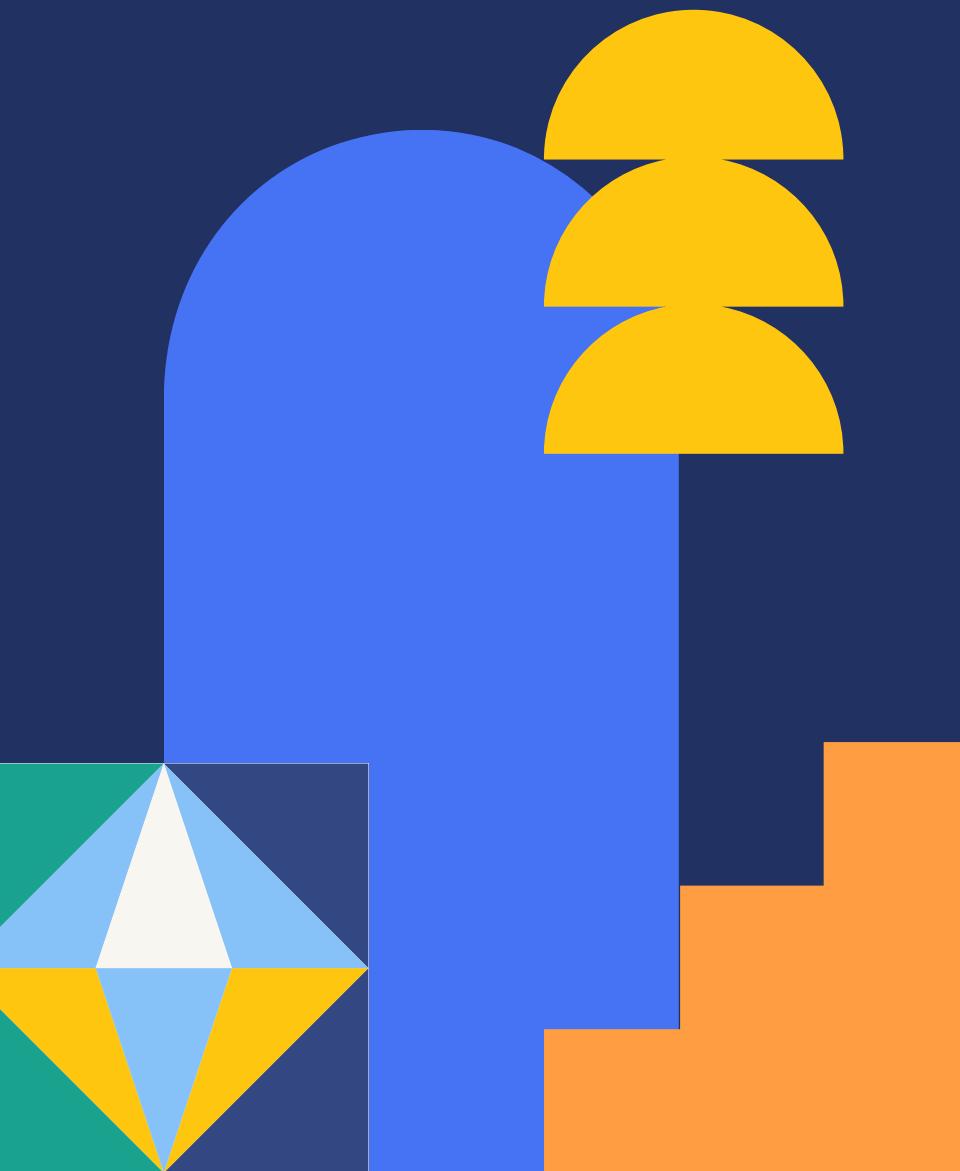
```
Teks: kereta cepat whoosh keren banget
Prediksi: 1
Probabilitas: {0: 0.35596185208918124, 1: 0.42714221146102294, 2: 0.2168959364497958}
```

```
Teks: ini proyek yang sangat merugikan negara
Prediksi: 0
Probabilitas: {0: 0.6083784201358473, 1: 0.21258604925061064, 2: 0.1790355306135421}
```

```
Teks: biasa aja sih, ga terlalu relevan
Prediksi: 1
Probabilitas: {0: 0.3398567435189311, 1: 0.6210336486967353, 2: 0.03910960778433361}
```

```
Teks: keren banget whoosh
Prediksi: 2
Probabilitas: {0: 0.18853730117696235, 1: 0.3718117258433964, 2: 0.4396509729796413}
```

Modeling : Fine-Tuning IndoBERT





```
trainer = Trainer(  
    model=model,  
    args=training_args,  
    train_dataset=tokenized_ds["train"],  
    eval_dataset=tokenized_ds["validation"],  
    compute_metrics=compute_metrics,  
)
```

Train Indobert

```
trainer.train()  
  
c:\Users\hafizh\AppData\Local\Programs\Python\Python310\lib\site-packages\torch\utils\data\dataloader.py:665  
    warnings.warn(warn_msg)  
  
[396/396 17:23, Epoch 4/4]  

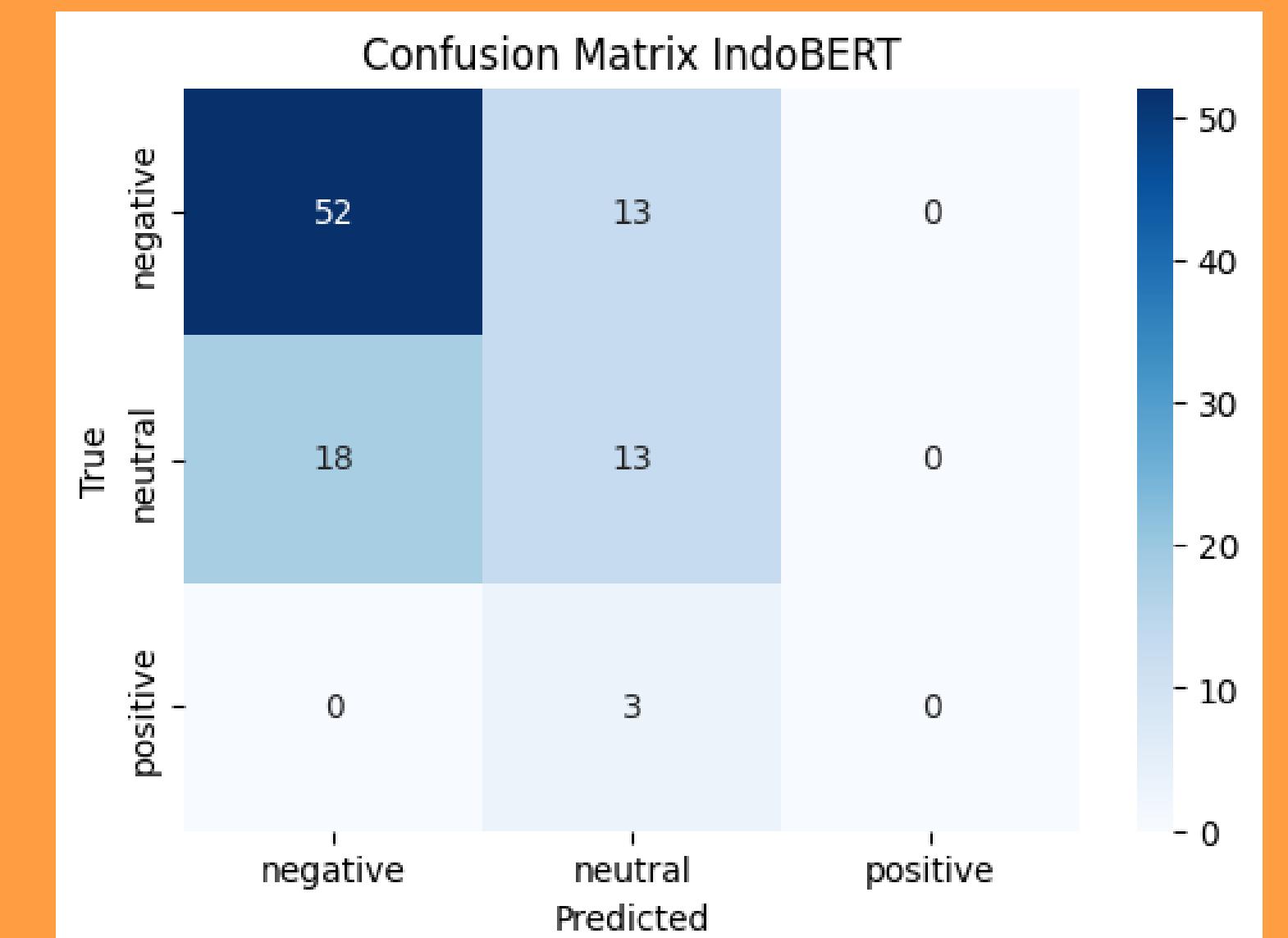

| Step | Training Loss |
|------|---------------|
| 100  | 0.689600      |
| 200  | 0.457500      |
| 300  | 0.250200      |


```

Classification Report

	precision	recall	f1-score	support
negative	0.74	0.80	0.77	65
neutral	0.45	0.42	0.43	31
positive	0.00	0.00	0.00	3
accuracy			0.66	99
macro avg	0.40	0.41	0.40	99
weighted avg	0.63	0.66	0.64	99

Confusion Matrix IndoBERT



Prediction

Prediksi Untuk 1 Teks

```
● ● ●  
def predict_sentiment(text: str):  
    # tokenisasi  
    inputs = tokenizer(  
        text,  
        return_tensors="pt",  
        truncation=True,  
        padding="max_length",  
        max_length=128  
    )  
  
    with torch.no_grad():  
        outputs = model(**inputs)  
        logits = outputs.logits  
        probs = torch.softmax(logits, dim=-1).cpu().numpy()[0]  
  
    pred_id = int(np.argmax(probs))  
    pred_label = id2label[pred_id]  
  
    prob_dict = {id2label[i]: float(probs[i]) for i in range(len(probs))}  
  
    print(f"Teks      : {text}")  
    print(f"Prediksi   : {pred_label}")  
    print("Probabilitas:")  
    for lbl, p in prob_dict.items():  
        print(f"  {lbl:<8}: {p:.4f}")  
    print("-" * 50)  
  
    return pred_label, prob_dict
```

```
Teks      : kereta cepat whoosh ini keren banget, bangga sama Indonesia  
Prediksi   : neutral  
Probabilitas:  
  negative: 0.0007  
  neutral : 0.9883  
  positive: 0.0110  
-----  
Teks      : proyek ini buang-buang uang rakyat, parah banget  
Prediksi   : negative  
Probabilitas:  
  negative: 0.9951  
  neutral : 0.0013  
  positive: 0.0036  
-----  
Teks      : ya biasa aja sih, nggak ngaruh juga ke hidup saya  
Prediksi   : neutral  
Probabilitas:  
  negative: 0.0010  
  neutral : 0.9924  
  positive: 0.0066  
-----  
Teks      : korupsi itu merugikan masyarakat kecil  
Prediksi   : negative  
Probabilitas:  
  negative: 0.9929  
  ...  
  negative: 0.0007  
  neutral : 0.9907  
  positive: 0.0086  
-----
```

Prediksi Untuk 1 Batch

```
● ● ●  
def predict_batch(text_list):  
    inputs = tokenizer(  
        text_list,  
        return_tensors="pt",  
        truncation=True,  
        padding=True,  
        max_length=128  
    )  
  
    with torch.no_grad():  
        outputs = model(**inputs)  
        logits = outputs.logits  
        probs = torch.softmax(logits, dim=-1).cpu().numpy()  
  
    pred_ids = np.argmax(probs, axis=1)  
    pred_labels = [id2label[int(i)] for i in pred_ids]  
  
    results = []  
    for text, lbl, p in zip(text_list, pred_labels, probs):  
        prob_dict = {id2label[i]: float(p[i]) for i in range(len(p))}  
        results.append({  
            "text": text,  
            "pred_label": lbl,  
            "prob_negative": prob_dict["negative"],  
            "prob_neutral": prob_dict["neutral"],  
            "prob_positive": prob_dict["positive"],  
        })  
  
    return pd.DataFrame(results)
```

[16]

	comment	sentiment
309	Orang zalim seringkali mengandalkan kekuasaan,...	neutral
139	mana bsa berani usut kereta cepat	negative
499	KPK.di kendalikan sama Jokowi\nSeharusnya tent...	negative
854	Muliono manusia berhati iblis	negative
88	Yg di tangkap & di hukum jokodok	negative
398	Aiman kompoorr	neutral
905	Cacat Konstitusi,Pemilu,Awal yang Buruk?	neutral
107	Hati2 jangan bangun rute woosh lagi ke surabay...	negative
59	Yang harus di tangkap yang buat kebijakan dan ...	negative
534	Oknum korupsi di whoosh harus diusut, biar dip...	negative

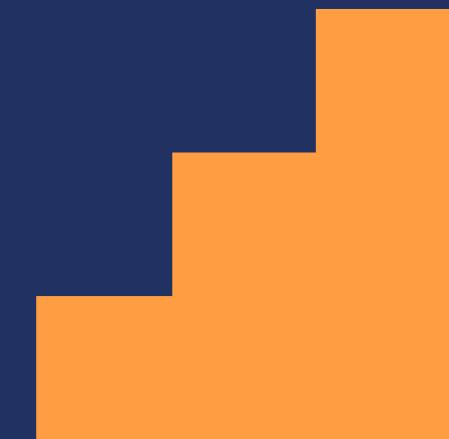
[17]

```
pred_df = predict_batch(sample_df["comment"].tolist())  
pred_df["true_label"] = sample_df["sentiment"].values  
pred_df
```

	text	pred_label	prob_negative	prob_neutral	prob_positive	true_label
0	Orang zalim seringkali mengandalkan kekuasaan,...	negative	0.982348	0.004064	0.013588	neutral
1	mana bsa berani usut kereta cepat	negative	0.992900	0.003107	0.003993	negative
2	KPK.di kendalikan sama Jokowi\nSeharusnya tent...	negative	0.996901	0.000986	0.002113	negative
3	Muliono manusia berhati iblis	neutral	0.097392	0.890316	0.012292	negative
4	Yg di tangkap & di hukum jokodok	negative	0.995841	0.001904	0.002255	negative
5	Aiman kompoorr	neutral	0.000740	0.995752	0.003507	neutral
6	Cacat Konstitusi,Pemilu,Awal yang Buruk?	neutral	0.000601	0.991494	0.007905	neutral
7	Hati2 jangan bangun rute woosh lagi ke surabay...	negative	0.994215	0.001052	0.004732	negative
8	Yang harus di tangkap yang buat kebijakan dan ...	negative	0.995586	0.002310	0.002104	negative
9	Oknum korupsi di whoosh harus diusut, biar dip...	negative	0.995099	0.000717	0.004185	negative



Link Penting

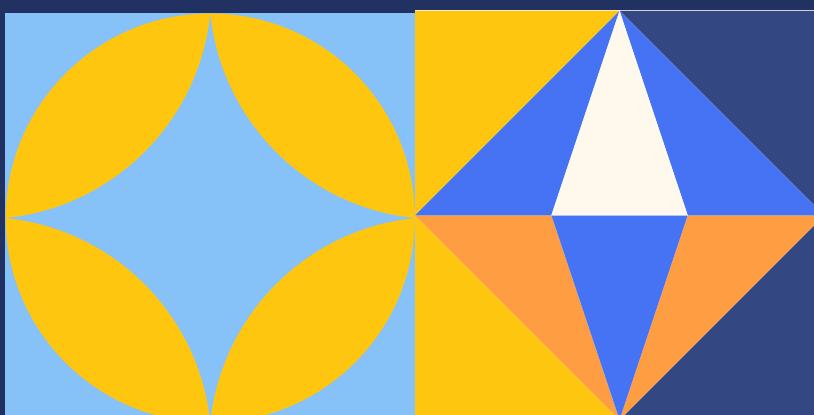


Link GitHub :

<https://github.com/HafizhFadhlMuhammad/project-sentimen-analisis-korupsi-whoosh.git>

Link Dataset + Hasil Model :

<https://drive.google.com/drive/folders/11bwnJ09TKnAplSwOYOOgNQI10Qzthx-Z?usp=sharing>



Komparasi Model

Komparasi :

- TF-IDF + Logistic Regression

Memberikan performa paling stabil dengan F1 Macro tertinggi (0.45) pada dataset imbalanced.

- TF-IDF + Linear SVM

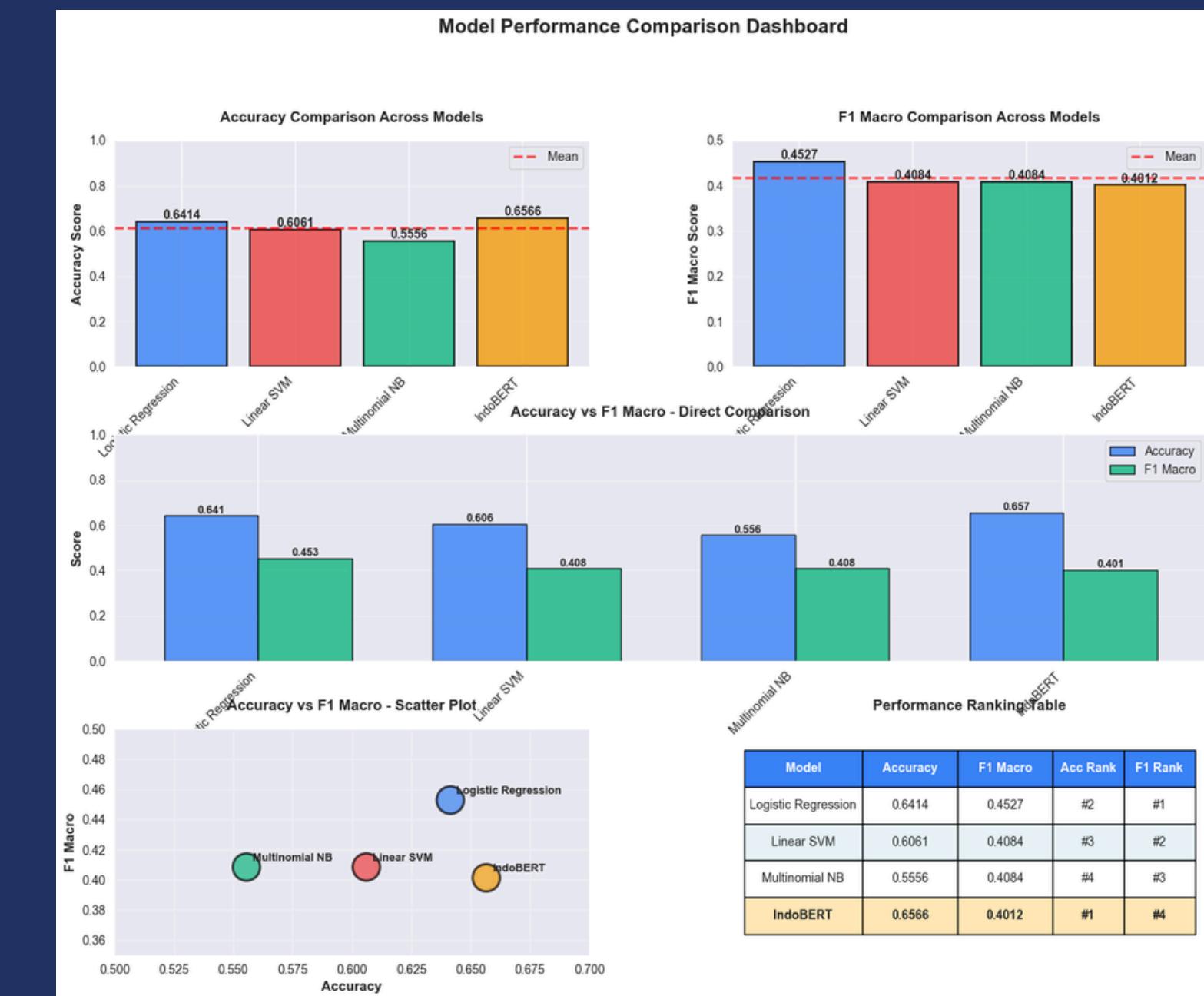
Memiliki akurasi kompetitif (0.61), namun F1 Macro lebih rendah dibanding Logistic Regression.

- TF-IDF + Multinomial Naive Bayes

Menunjukkan performa terendah, terutama pada kemampuan klasifikasi kelas minoritas.

- IndoBERT

Mencapai akurasi tertinggi (0.66), namun F1 Macro lebih rendah (0.40) akibat keterbatasan data dan ketidakseimbangan kelas.





Kesimpulan Teknis & Akhir

Teknis :

- Model klasik berbasis TF-IDF lebih stabil pada dataset berukuran kecil dan imbalanced.
- IndoBERT memiliki potensi lebih besar dalam memahami konteks bahasa Indonesia, namun membutuhkan dataset yang lebih besar dan seimbang.
- Perbedaan performa menunjukkan bahwa akurasi tinggi tidak selalu mencerminkan kualitas klasifikasi yang seimbang.
- Pemilihan model harus mempertimbangkan karakteristik data dan tujuan evaluasi.

Akhir :

Berdasarkan hasil analisis sentimen terhadap komentar YouTube terkait isu korupsi proyek Kereta Cepat Whoosh, dapat disimpulkan bahwa opini publik didominasi oleh sentimen negatif, yang menunjukkan tingginya sikap kritis dan ketidakpuasan masyarakat terhadap isu tersebut. Sentimen netral muncul sebagai bentuk diskusi informatif, sementara sentimen positif memiliki proporsi paling kecil.

Perbandingan metode menunjukkan bahwa model klasik berbasis TF-IDF memberikan performa yang stabil pada dataset imbalanced, sedangkan model NLP modern berbasis IndoBERT mencapai akurasi tertinggi namun belum optimal dalam menangani kelas minoritas akibat keterbatasan jumlah data. Hal ini mengindikasikan bahwa pemilihan metode analisis sentimen harus mempertimbangkan karakteristik data serta tujuan evaluasi.



Saran dan Rekomendasi



1. Penanganan Imbalanced Data: Dataset didominasi sentimen negatif (65,35%) yang menekan performa kelas minoritas (3,34%). Disarankan menggunakan teknik class weighting atau text augmentation untuk IndoBERT, dan hindari oversampling duplikasi untuk mencegah overfitting.
2. Penambahan Dataset: IndoBERT terbukti kurang optimal pada data kecil (~1.000 baris). Penelitian selanjutnya wajib memperbanyak data secara signifikan dari berbagai platform (YouTube, X, Instagram) untuk memaksimalkan potensi arsitektur Transformer.



Gracias!

