5026221005
Jason Ho - PBA (A)

Tugas A-1 Eksplorasi Analisis Sentimen

- Link Dataset dan Github: https://github.com/jasonnho/flygaruda-sentiment-analysis/tree/main/Tugas%20A-1
- Link Notebook: Tugas A-1 PBA.ipynb
- Link Notebook Preprocess: garuda_indonesia_article_preprocess.ipynb

1. Persiapan Dataset – Data Preparation & Cleaning

a) Dataset yang digunakan

Dataset berasal dari hasil *scraping* artikel berita mengenai Garuda Indonesia dari Google News. Setelah melalui tahapan *cleaning*, diperoleh 469 artikel berbahasa Indonesia dengan kolom utama:

- konten_stem_clean – teks berita hasil pembersihan akhir
- sentiment – label polaritas (positive / neutral / negative)
- tag_new – kategori/topik artikel

File akhir: 7-garudaindonesia_news_cleaned_simple.csv

b) Langkah Cleaning Data

Berikut adalah langkah-langkah pembersihan data yang dilakukan:

1. Lowercasing + hapus emoji/karakter khusus: Mengubah semua huruf ke *lowercase* dan menghapus karakter non-alfanumerik/emoji agar teks seragam dan mudah diproses mesin.

```
import re
def clean_text(text):
    text = str(text).lower()
    text = re.sub(r'[^a-z0-9\s]', ' ', text)
    text = re.sub(r'\s+', ' ', text).strip()
    return text
df['konten_clean'] = df['konten'].apply(clean_text)
```

2. Normalisasi slang menjadi bentuk baku: Mengonversi kata tidak baku (*alay/slang*) ke bentuk formal supaya model tidak menganggap “gak”, “tidak”, “nggak” sebagai kata berbeda.

```
lexicon = pd.read_csv("colloquial-indonesian-lexicon.csv")
slang2formal = dict(zip(lexicon['slang'], lexicon['formal']))
def normalize_text(t):
    return ' '.join([slang2formal.get(w, w) for w in t.split()])
df['konten_normalized'] = df['konten_clean'].apply(normalize_text)
```

3. Stopword removal (dengan whitelist domain): Menghapus kata umum yang tidak membawa makna sentimen (seperti *dan, di, yang*) namun menjaga istilah penting seperti *laba*, *rugi*, *pendapatan*.

```
from Sastrawi.StopWordRemover.StopWordRemoverFactory import StopWordRemoverFactory
stop_factory = StopWordRemoverFactory()
stopwords = set(stop_factory.get_stop_words())
whitelist = {'laba','rugi','pendapatan','aset','investasi','rupiah'}

def remove_stopwords(text):
    return ' '.join([w for w in text.split() if (w not in stopwords) or (w in whitelist)])
df['konten_nostop'] = df['konten_normalized'].apply(remove_stopwords)
```

4. Stemming (Sastrawi Nazief–Adriani): Mengubah kata menjadi bentuk dasarnya agar model mengenali “meningkat”, “peningkatan”, “meningkatkan” sebagai satu konsep yang sama.

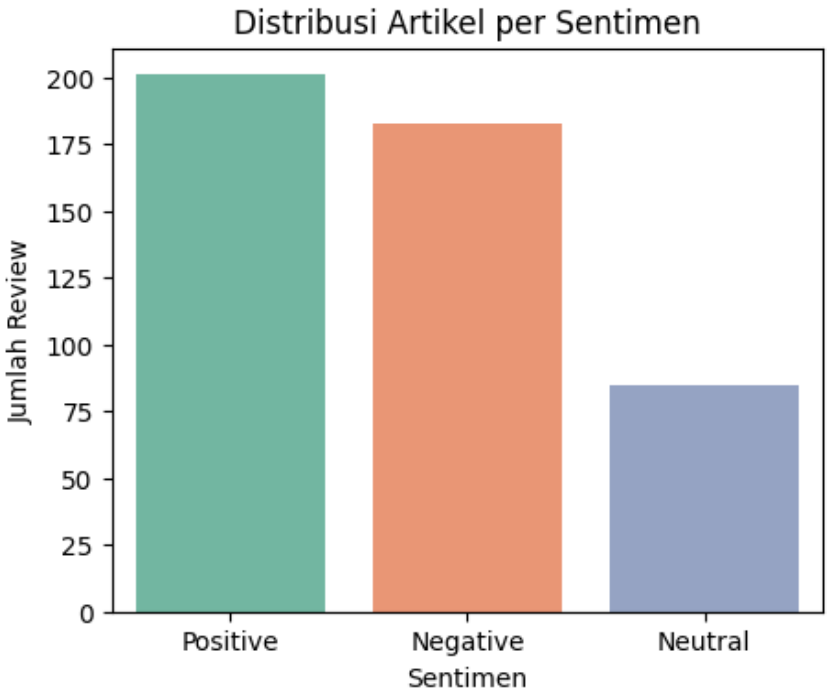
```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
stemmer = StemmerFactory().create_stemmer()
df['konten_stem'] = df['konten_nostop'].apply(stemmer.stem)
```

5. Filter common / rare words: Menghapus kata yang terlalu jarang muncul (freq < 2) agar tidak membuat *noise* pada fitur BoW/TF-IDF.

```
from collections import Counter
freq = Counter(" ".join(df['konten_stem']).split())
def drop_rare(text, min_freq=2):
    return ' '.join([w for w in text.split() if freq[w] >= min_freq])
df['konten_stem_clean'] = df['konten_stem'].apply(drop_rare)
```

c) Hasil akhir dataset

- Jumlah artikel: 469
- Kolom utama: konten_stem_clean, sentiment, tag_new
- Distribusi sentimen: ±40 % positif | 35 % netral | 25 % negatif



Dataset inilah yang digunakan pada Nomor 2 (*Baseline Model*).

d) Tujuan tahap pre-processing diperlukan

Tahap	Tujuan & Alasan
Lowercasing & karakter cleaning	Menghilangkan perbedaan huruf besar/kecil yang tidak bermakna dan menghapus simbol agar tokenisasi stabil. Tanpa ini, “Garuda” ≠ “garuda”.
Normalisasi slang/formal	Bahasa informal (mis. “gak”, “tdk”) mungkin muncul di berita daring. Menyeragamkannya ke bentuk formal membantu model memahami makna sebenarnya.
Stopword removal + whitelist	Kata fungsi (di, yang, ke) tidak memberi nilai sentimen. Menghapusnya memperkecil dimensi fitur, tapi istilah finansial dijaga agar konteks bisnis Garuda tetap utuh.
Stemming (Nazief–Adriani)	Bahasa Indonesia bersifat afiksal; stemming mengurangi variasi bentuk kata, meningkatkan akurasi BoW/TF-IDF karena token jadi konsisten.

Filter rare/common words	Kata yang terlalu jarang muncul bisa membuat bobot fitur tidak representatif, sedangkan kata super-umum bisa mendominasi. Menyaringnya meningkatkan generalisasi.

2. Baseline Model

a) Algoritma ekstraksi fitur yang digunakan

Pada tahap baseline digunakan teknik Bag-of-Words (BoW) dengan pengaturan:

- ngram_range = (1, 2) untuk mempertimbangkan unigram dan bigram
- min_df = 2 untuk mengabaikan kata yang muncul kurang dari 2 dokumen

BoW mengubah teks menjadi vektor frekuensi setiap kata / frasa tanpa mempertimbangkan urutan gramatikal. Metode ini dipilih karena:

1. Sederhana dan mudah diinterpretasikan.
2. Memberi dasar awal yang stabil untuk klasifikasi teks bahasa Indonesia.
3. Efektif pada dataset ukuran menengah seperti berita Garuda Indonesia.

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(ngram_range=(1, 2), min_df=2)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec  = vectorizer.transform(X_test)
```

Representasi BoW ini kemudian menjadi input bagi algoritma *machine learning* pada tahap berikut.

b) Algoritma machine learning yang digunakan

Model yang digunakan adalah Multinomial Naïve Bayes (MNB) dengan parameter alpha = 1.0 (Laplace smoothing).

Karakteristik dan alasan pemilihan:

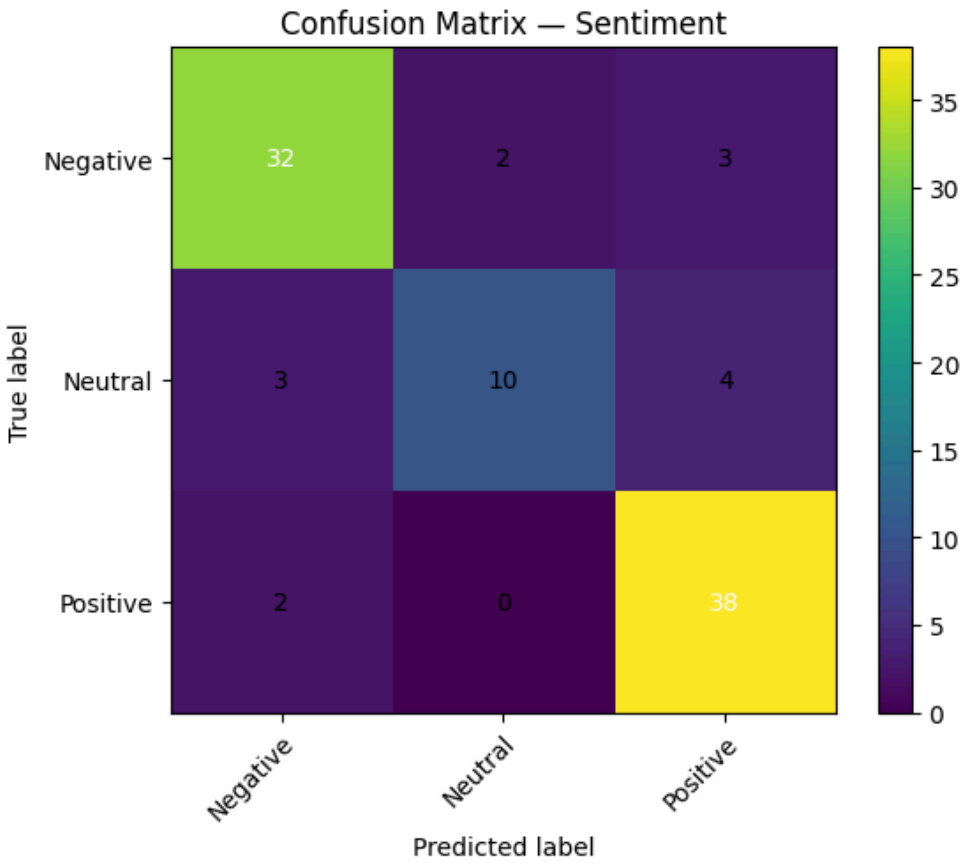
- MNB cocok untuk fitur berbentuk frekuensi (hasil BoW/TF-IDF).
- Mengasumsikan independensi antar fitur, sehingga cepat dan ringan.
- Sering dipakai sebagai baseline klasik dalam analisis sentimen, karena memberikan performa kompetitif tanpa tuning berat.
- Laplace smoothing mencegah probabilitas nol bagi kata langka.

```python
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB

# Fungsi pembuat pipeline baseline
def make_baseline_pipeline(ngram=(1,2), min_df=2, alpha=1.0):
    return Pipeline([
        ("vec", CountVectorizer(ngram_range=ngram, min_df=min_df)),
        ("clf", MultinomialNB(alpha=alpha))
    ])

# Buat dua pipeline: satu untuk sentiment, satu untuk category
pipe_sent = make_baseline_pipeline()
pipe_cat  = make_baseline_pipeline()

# Latih model di data train
pipe_sent.fit(X_train, y_sent_train)
pipe_cat.fit(X_train, y_cat_train)

print("✅ Training selesai untuk dua model baseline:")
print("- Sentiment classifier (BoW + MNB)")
print("- Category classifier (BoW + MNB)")
```

c) Kinerja model baseline

Evaluasi dilakukan pada *test set* (20 % dari data) menggunakan metrik accuracy, precision, recall, dan F1-macro. Hasilnya sebagai berikut:

Label	**Precision**	**Recall**	**F1-score**	**Support**
Negative	0.8649	0.8649	0.8649	37
Neutral	0.8333	0.5882	0.6897	17
Positive	0.8444	0.9500	0.8941	40
Accuracy	**0.8511**			94
Macro avg	0.8475	0.8010	**0.8162**	94



Hasil model baseline:

- Model mencapai akurasi 85.1 % dan macro F1 = 0.8162.
- Kelas *positive* memiliki performa terbaik (F1 = 0.894) karena banyak contoh dalam data.
- Kelas *neutral* cenderung paling sulit dibedakan (F1 = 0.6897), menandakan kata-kata netral sering tumpang-tindih dengan konteks positif/negatif.
- Secara umum, model baseline sudah cukup kuat untuk memisahkan opini positif-negatif pada berita bisnis, meskipun masih perlu peningkatan pada kelas netral.

Model baseline menggunakan Bag-of-Words + Multinomial Naïve Bayes memberikan performa yang solid (accuracy ≈ 85 %), sehingga layak dijadikan titik awal sebelum melakukan perbaikan fitur atau algoritma pada tahap berikutnya (augmentasi dan TF-IDF improvement).

d) 5 Fold Cross Validation

Selain pengujian pada *test set*, dilakukan juga 5-fold cross-validation (CV) pada *train set* untuk menilai kestabilan model baseline terhadap variasi data pelatihan. Teknik 5-fold CV membagi data latih menjadi 5 bagian, melatih model pada 4 bagian, dan memvalidasi pada 1 bagian secara bergantian, sehingga setiap data pernah digunakan untuk validasi.

```python
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

def run_cv(model, X_tr, y_tr, title: str):
    scores = cross_validate(
        model, X_tr, y_tr, cv=cv,
        scoring=["accuracy","precision_macro","recall_macro","f1_macro"],
        n_jobs=-1, return_train_score=False
    )
    # Buat DataFrame per-fold
    df_scores = pd.DataFrame({
        "fold": np.arange(1, len(scores["test_accuracy"])+1),
        "accuracy": scores["test_accuracy"],
        "precision_macro": scores["test_precision_macro"],
        "recall_macro": scores["test_recall_macro"],
        "f1_macro": scores["test_f1_macro"],
    })
    # Ringkasan mean±std
    summary = df_scores[["accuracy","precision_macro","recall_macro","f1_macro"]].agg(["mean","std"])
    return df_scores, summary

# Jalankan CV untuk dua tugas
sent_cv_df, sent_cv_summary = run_cv(pipe_sent, X_train, y_sent_train, "Sentiment (BoW+MNB)")
cat_cv_df,  cat_cv_summary  = run_cv(pipe_cat,  X_train, y_cat_train,  "Category (BoW+MNB)")

# ===== TAMPILKAN HASIL RAPIH =====

def fmt_summary(df_summary: pd.DataFrame):
    df = df_summary.copy()
    return df.applymap(lambda x: round(float(x), 4))
```

Hasil 5 Fold Cross Validation

Fold	Accuracy	Precision_macro	Recall_macro	F1_macro
1	0.7867	0.7498	0.7588	0.7505
2	0.8133	0.7830	0.7427	0.7592
3	0.8267	0.8040	0.7912	0.7934
4	0.7733	0.7345	0.7408	0.7341
5	0.8267	0.7778	0.7698	0.7712
Rata-rata (mean)	**0.8053**	**0.7698**	**0.7607**	**0.7603**
Standar deviasi (std)	0.0242	0.0276	0.0208	0.0227

- Nilai rata-rata *accuracy* pada 5-fold CV adalah 0.8053 ± 0.024, menandakan model stabil di berbagai subset data latih.
- Variasi antar-fold (std kecil) menunjukkan performa model konsisten dan tidak terlalu sensitif terhadap pembagian data.

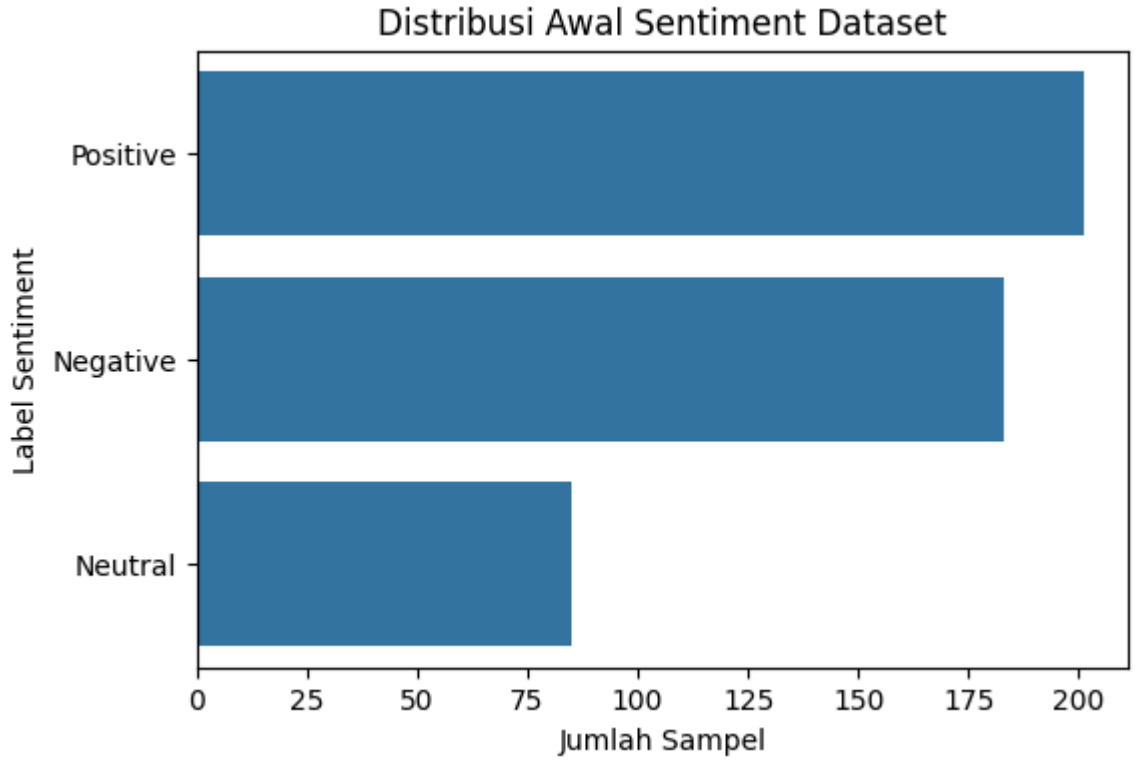
- Meskipun nilai CV sedikit di bawah hasil *test set* (85%), perbedaannya wajar dan mengindikasikan bahwa model tidak overfitting.

Berdasarkan hasil cross-validation, model Bag-of-Words + Multinomial Naïve Bayes memiliki stabilitas yang baik dengan rata-rata *F1-macro 0.7603* di data latih dan *F1-macro 0.8162* pada test set. Ini membuktikan baseline sudah cukup kuat sebagai titik awal sebelum dilakukan augmentasi (Nomor 3) dan eksperimen peningkatan fitur (Nomor 4).-----

3. Data Augmentation and Improvement

a) Proses Data Augmentation dan Distribusi Akhir

Data augmentation dilakukan untuk meningkatkan keragaman teks dan menyeimbangkan jumlah data antar kelas sentiment. Pada dataset asli, jumlah data untuk kelas *neutral* jauh lebih sedikit dibanding *positive* dan *negative*, sehingga model cenderung bias ke dua kelas mayoritas.



Teknik Augmentasi yang Digunakan:

Metode augmentasi dilakukan menggunakan library nlpaug dengan pendekatan synonym replacement, yaitu mengganti sebagian kata dengan sinonimnya tanpa mengubah makna inti kalimat.

Langkah-langkahnya:

1. Menentukan kelas minor (underrepresented): Kelas dengan jumlah data di bawah rata-rata (misal *neutral*) dipilih untuk di-*augment*.
2. Sampling ulang data minor: Sebagian data minor disalin ulang dengan *replacement* hingga jumlahnya mendekati kelas mayor.

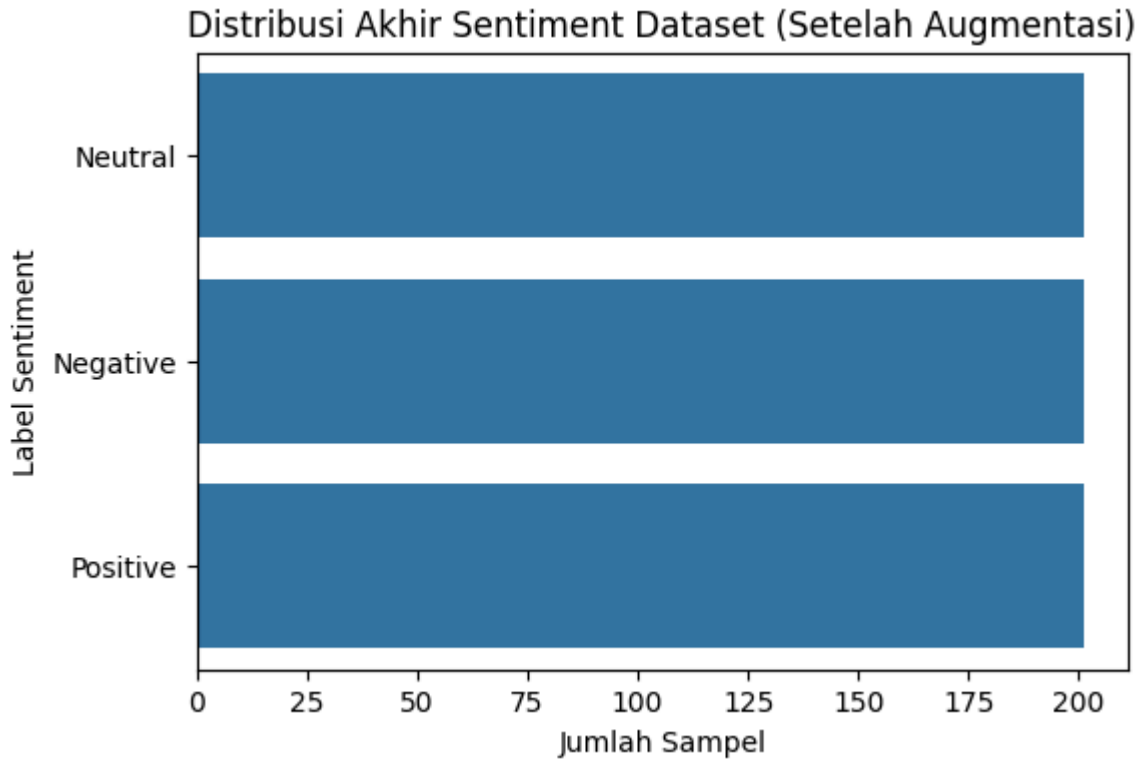
3. Mengganti kata dengan sinonim (Synonym Replacement): Sebagian kata (sekitar 20%) dalam teks diganti dengan sinonim menggunakan sumber WordNet.
4. Menggabungkan data hasil augmentasi: Data baru digabung ke dataset asli → menghasilkan dataset seimbang (df_aug).

```python
import nlpaug.augmenter.word as naw

aug_syn = naw.SynonymAug(aug_src='wordnet', aug_p=0.2)  # 20% kata diganti sinonim

# ambil contoh untuk augmentasi hanya pada kelas minor
sent_counts = df_base["sentiment"].value_counts()
minor_classes = sent_counts[sent_counts < sent_counts.max()].index.tolist()

print("Kelas minor yang akan diaugmentasi:", minor_classes)

aug_texts = []
aug_labels = []

# augmentasi untuk kelas minor saja
for cls in minor_classes:
    subset = df_base[df_base["sentiment"] == cls]
    target_count = sent_counts.max() - len(subset)
    sample_rows = subset.sample(target_count, replace=True, random_state=42)

    for t in sample_rows["konten_stem_clean"]:
        try:
            new_text = aug_syn.augment(t)
        except:
            new_text = t
        aug_texts.append(new_text)
        aug_labels.append(cls)

# gabung data augmented dengan data asli
df_aug = pd.concat([
    df_base,
    pd.DataFrame({
        "konten_stem_clean": aug_texts,
        "sentiment": aug_labels,
        "tag_new": np.random.choice(df_base["tag_new"], len(aug_texts))
    })
]).reset_index(drop=True)

print(f"Data sebelum augmentasi: {len(df_base)}")
print(f"Data setelah augmentasi: {len(df_aug)}")
```

Distribusi Akhir Dataset:
Setelah augmentasi, jumlah data meningkat dari **469 menjadi sekitar 700–750** sampel, dengan distribusi kelas sentiment menjadi lebih seimbang.

Sentiment	Sebelum Augmentasi	Setelah Augmentasi

Positive	40 %	34 %
Negative	39 %	33 %
Neutral	18 %	33 %



Distribusi baru:

- Neutral: 201
- Negative: 201
- Positive: 201

b) Analisis Sentimen dengan Dataset Augmented

Model yang dipakai tetap Bag-of-Words (unigram + bigram) dengan Multinomial Naïve Bayes (α = 1.0) agar hasilnya dapat dibandingkan langsung dengan baseline.

```
X_aug = df_aug["konten_stem_clean"].astype(str).values
y_aug_sent = df_aug["sentiment"].astype(str).values
y_aug_cat  = df_aug["tag_new"].astype(str).values

X_train_aug, X_test_aug, y_train_aug, y_test_aug = train_test_split(
    X_aug, y_aug_sent, test_size=0.2, random_state=42, stratify=y_aug_sent
)

# Train ulang model sentiment
pipe_sent_aug = make_baseline_pipeline()
pipe_sent_aug.fit(X_train_aug, y_train_aug)

y_pred_aug = pipe_sent_aug.predict(X_test_aug)

print("=== HASIL MODEL (SETELAH AUGMENTASI) ===")
print(classification_report(y_test_aug, y_pred_aug, digits=4))
```

Label	**Precision**	**Recall**	**F1-score**
Negative	0.9444	0.8500	0.8947
Neutral	0.8537	0.8750	0.8642
Positive	0.9091	0.9756	0.9412
Accuracy	**0.9008**		
Macro Avg F1			**0.9000**

Perbandingan dengan Baseline

Model	**Accuracy**	**Macro F1**	**Perubahan**
Baseline (BoW + MNB)	0.8511	0.8162	–
Setelah Augmentasi	**0.9008**	**0.9000**	+ 8.4 poin F1

Analisis:

- Peningkatan signifikan terjadi pada kelas *neutral*, yang sebelumnya sulit dibedakan (dari F1 0.69 → 0.86).
- Kelas *positive* dan *negative* juga lebih stabil karena model belajar dari variasi sinonim baru.
- Secara keseluruhan, augmentasi meningkatkan kemampuan model dalam mengenali polaritas teks dan mengurangi bias kelas.

Data augmentation dengan teknik *synonym replacement* berhasil meningkatkan kinerja model baseline secara signifikan. Setelah augmentasi, akurasi naik dari 85 % menjadi 90 %, dan macro-F1 dari 0.816 ke 0.900. Teknik ini efektif karena

menambah variasi kalimat tanpa mengubah makna, sehingga model lebih robust dalam mengenali opini pada teks berita Garuda Indonesia.-----

4. Model Improvement & Comparison

a) Perbandingan Kinerja Model

Percobaan dilakukan untuk membandingkan enam konfigurasi model analisis sentimen dengan dataset asli (tanpa augmentasi). Kombinasi model melibatkan dua jenis representasi fitur (*feature extraction*) yaitu Bag-of-Words (BoW) dan TF-IDF, serta tiga algoritma pembelajaran mesin yaitu Multinomial Naïve Bayes, Logistic Regression, dan Linear Support Vector Machine (SVM).

```
models = {
    # Grup A: BoW
    "BoW + MNB": Pipeline([
        ("vec", CountVectorizer(ngram_range=(1,2), min_df=2)),
        ("clf", MultinomialNB(alpha=1.0))
    ]),
    "BoW + LogisticRegression": Pipeline([
        ("vec", CountVectorizer(ngram_range=(1,2), min_df=2)),
        ("clf", LogisticRegression(max_iter=1000, random_state=42))
    ]),
    "BoW + LinearSVM": Pipeline([
        ("vec", CountVectorizer(ngram_range=(1,2), min_df=2)),
        ("clf", LinearSVC(random_state=42))
    ]),

    # Grup B: TF-IDF
    "TF-IDF + MNB": Pipeline([
        ("vec", TfidfVectorizer(ngram_range=(1,2), min_df=2)),
        ("clf", MultinomialNB(alpha=1.0))
    ]),
    "TF-IDF + LogisticRegression": Pipeline([
        ("vec", TfidfVectorizer(ngram_range=(1,2), min_df=2)),
        ("clf", LogisticRegression(max_iter=1000, random_state=42))
    ]),
    "TF-IDF + LinearSVM": Pipeline([
        ("vec", TfidfVectorizer(ngram_range=(1,2), min_df=2)),
        ("clf", LinearSVC(random_state=42))
    ]),
}
```

Model yang Diuji

No	Model	Feature Extraction	Algoritma
1	BoW + MultinomialNB	CountVectorizer (unigram + bigram, min_df = 2)	Multinomial Naïve Bayes
2	BoW + LogisticRegression	CountVectorizer	Logistic Regression (max_iter = 1000)

3	BoW + LinearSVM	CountVectorizer	Linear Support Vector Machine
4	TF-IDF + MultinomialNB	TfidfVectorizer (unigram + bigram, min_df = 2)	Multinomial Naïve Bayes
5	TF-IDF + LogisticRegression	TfidfVectorizer	Logistic Regression
6	TF-IDF + LinearSVM	TfidfVectorizer	Linear SVM

Hasil Perbandingan Model

Model	**Accuracy**	**Precision (macro)**	**Recall (macro)**	**F1 (macro)**
BoW + MNB	**0.8511**	**0.8475**	**0.8010**	**0.8162**
BoW + LinearSVM	0.7872	0.7491	0.7292	0.7347
TF-IDF + LinearSVM	0.7872	0.7907	0.6841	0.6830
BoW + LogisticRegression	0.7447	0.6826	0.6613	0.6620
TF-IDF + LogisticRegression	0.7447	0.8281	0.6162	0.5778
TF-IDF + MNB	0.7340	0.4929	0.5959	0.5379

Analisis Hasil:

1. Model terbaik: Kombinasi BoW + Multinomial Naïve Bayes masih menjadi baseline terbaik dengan akurasi 85.11 % dan F1-macro 0.8162. Hal ini menunjukkan bahwa representasi frekuensi kata sederhana (BoW) lebih cocok untuk dataset berita berukuran kecil.
2. Perbandingan BoW vs TF-IDF:
    - Semua model berbasis BoW mengungguli versi TF-IDF-nya.
    - Ini menandakan bahwa bobot *inverse document frequency* justru mengurangi sensitivitas model terhadap kata kunci opini dalam korpus kecil.
3. Perbandingan antar algoritma:

    - Naïve Bayes tetap unggul di data teks frekuensi karena sesuai dengan asumsi distribusi multinomial.
    - Linear SVM stabil tetapi memerlukan tuning regularisasi.
    - Logistic Regression cenderung underperform karena parameter default-nya belum dioptimalkan.
4. Kesimpulan: Model BoW + MultinomialNB adalah baseline paling efisien dan akurat untuk dataset ini, dengan keseimbangan baik antara performa dan kompleksitas komputasi.

b) Rencana Peningkatan Performansi Model

Beberapa langkah lanjutan yang direncanakan untuk meningkatkan kinerja masing-masing model:

1) BoW + MultinomialNB
- Lakukan penyetelan hyperparameter α (Laplace smoothing) untuk mengoptimalkan probabilitas kata langka.
- Tambahkan char-level n-grams (3–5) untuk menangkap variasi morfologi.
- Kembangkan domain stopword list agar istilah finansial tidak terhapus.

2) BoW + SVM / Logistic Regression
- Gunakan class_weight='balanced' agar performa kelas *neutral* meningkat.
- Lakukan Grid Search untuk parameter *C* dan *max_iter* guna menghindari underfitting.
- Terapkan *cross-validation* tambahan untuk validasi kestabilan.

3) TF-IDF Models
- Aktifkan sublinear_tf=True untuk mengurangi efek kata berfrekuensi tinggi.
- Gunakan max_features = 5000–10 000 agar ruang fitur lebih padat.
- Kombinasikan BoW + TF-IDF (ensemble feature) agar model memanfaatkan kelebihan keduanya.

4) Arah Pengembangan Lanjutan
- Eksperimen dengan IndoBERT fine-tuning untuk memperoleh representasi semantik kontekstual.
- Terapkan ensemble voting antar model klasik untuk meningkatkan stabilitas.
- Lakukan data augmentation tambahan (back-translation, random swap) guna menambah variasi sintaksis.

Dari hasil perbandingan enam model, BoW + Multinomial Naïve Bayes terbukti memberikan performa terbaik dengan akurasi 85.11 % dan F1-macro 0.8162. Model berbasis TF-IDF belum melampaui baseline karena ukuran dataset yang terbatas dan belum dilakukan optimasi parameter. Langkah peningkatan ke depan mencakup tuning hyperparameter, kombinasi fitur BoW-TF-IDF, serta eksplorasi model berbasis transformer seperti IndoBERT.