Jurnal Sifo Mikroskil (JSM)
Volume 26, No 2, Oktober 2025 – Hal 257 - 274
DOI: https://doi.org/10.55601/jsm.v26i2.1799

e-ISSN: 2622-8130
ISSN: 1412-0100

Evaluasi Teknik Resampling untuk Class Balancing dalam Analisis Sentimen Kesehatan Mental Berbasis Bi-LSTM

Cindy Sintiya[1], Grace Helena Hutagaol[2], David Bate`e[3], Syanti Irviantina[4*]
[1,2,3,4]Universitas Mikroskil, Jl. Thamrin No. 112, 124, 140, Telp. (061) 4573767
[1,2,3,4]Informatika, Teknik Informatika, Universitas Mikroskil, Medan
e-mail: [1]211110347@students.mikroskil.ac.id, [2]211110948@students.mikroskil.ac.id,
[3]211111930@students.mikroskil.ac.id, [4]syanti@mikroskil.ac.id

Dikirim: ***14-07-2025*** | *Diterima:* ***04-08-2025*** | *Diterbitkan:* ***31-10-2025***

Abstrak

Ketidakseimbangan data (*imbalanced data*) sering menjadi tantangan utama dalam analisis sentimen, terutama ketika model pembelajaran mesin cenderung mengabaikan kelas minoritas yang justru memuat informasi kritis. Penelitian ini mengevaluasi efektivitas teknik *class balancing* dengan metode *resampling*, yaitu *Random Under-Sampling* (RUS) dan *Synthetic Minority Over-sampling Technique* (SMOTE), dalam meningkatkan performa model LSTM (*Long Short-Term Memory*) untuk analisis sentimen kesehatan mental. Hasil penelitian menunjukkan bahwa metode SMOTE dapat meningkatkan *F1-score* pada kelas "*Suicidal*" dari 64,4% menjadi 72,6%, meskipun terjadi penurunan pada kelas "*Depression*" dari 73,4% menjadi 59,9%. Sementara itu, metode RUS cenderung menurunkan performa secara keseluruhan, dengan akurasi model turun dari 80,5% menjadi 77,8% akibat penghapusan data secara acak yang mengurangi kualitas representasi data. Penelitian ini menyimpulkan bahwa meskipun teknik *resampling* dapat membantu menyeimbangkan data, namun penerapannya pada data teks memerlukan kehati-hatian untuk menghindari efek negatif pada performa model.

Kata kunci: *class balancing*, *resampling*, analisis sentimen, kesehatan mental, LSTM.

Abstract

Imbalanced data is often a major challenge in sentiment analysis, especially when machine learning models tend to ignore minority classes that contain critical information. This research evaluates the effectiveness of class balancing techniques with resampling methods, such as Random Under-Sampling (RUS) and Synthetic Minority Over-sampling Technique (SMOTE), in improving the performance of LSTM (Long Short-Term Memory) models for mental health sentiment analysis. The results show that the SMOTE method increases the F1-score for the "Suicidal" class from 64.4% to 72.6%, although it causes a performance drop in the "Depression" class from 73.4% to 59.9%. Meanwhile, the RUS method tends to degrade overall performance, with model accuracy decreasing from 80.5% to 77.8% due to random data removal, which reduces the quality of data representation. This study concludes that while resampling techniques can help balance the data, their application to text data requires caution to avoid negative effects on model performance.

Keywords: *class balancing, resampling, sentiment analysis, mental health, LSTM.*

1. PENDAHULUAN

Kesehatan mental merupakan aspek penting dalam kesejahteraan manusia, namun kesadaran terhadap isu ini masih rendah karena stigma sosial dan minimnya literasi masyarakat. Seiring dengan meningkatnya penggunaan media sosial, sudah tersedia banyak data tekstual yang dapat dimanfaatkan untuk menganalisis sentimen publik terkait kesehatan mental [1], [2]. Namun, analisis sentimen terhadap data ini menghadapi tantangan utama berupa ketidakseimbangan distribusi kelas (*imbalanced data*) [3], di mana model cenderung memprioritaskan kelas mayoritas dan mengabaikan kelas minoritas yang memuat informasi kritis. Ketidakseimbangan ini umumnya terjadi karena sebagian besar pengguna media sosial lebih sering membagikan konten netral atau umum, sementara ekspresi terkait gangguan mental seperti depresi atau gangguan kecemasan lebih jarang diungkapkan secara eksplisit [4]. Selain itu, adanya stigma sosial membuat individu cenderung menyembunyikan kondisi mentalnya, sehingga data yang mencerminkan kondisi tersebut menjadi lebih sedikit [5]. Dalam konteks ini, diperlukan teknik penyeimbangan data agar performa model lebih optimal pada semua kelas [6], [7].

Salah satu pendekatan yang banyak digunakan untuk menangani masalah ketidakseimbangan data adalah teknik *resampling*, seperti *Random Under-Sampling* (RUS) dan *Synthetic Minority Over-sampling Technique* (SMOTE) [8]. Dengan *Random Under Sampling* (RUS), sejumlah data dari kelas mayoritas dihapus secara acak untuk mencapai distribusi yang lebih seimbang dengan kelas minoritas. Pendekatan ini secara efektif mengurangi bias terhadap kelas mayoritas, tetapi memiliki risiko kehilangan informasi berharga dari data yang dibuang. Sebaliknya, SMOTE menghasilkan data sintetis untuk kelas minoritas dengan menginterpolasi nilai dari sampel minoritas yang ada, sehingga meningkatkan ukuran sampel tanpa mengorbankan integritas data asli. Meskipun SMOTE mempertahankan informasi dari kelas mayoritas, SMOTE dapat meningkatkan risiko *overfitting* jika data minoritas terlalu kecil [9].

Pemilihan *Long Short-Term Memory* (LSTM) sebagai model utama didorong oleh kemampuannya yang secara efektif menangani data ekuensial, seperti teks. LSTM, salah satu jenis *Recurrent Neural Network* (RNN), dirancang untuk mengatasi masalah *vanishing gradient*, sehingga lebih efektif dalam mempelajari ketergantungan jangka panjang pada data. Dalam konteks analisis sentimen kesehatan mental, data tekstual sering kali mengandung pola dan informasi kontekstual yang menyebar di seluruh kalimat atau paragraf. Kemampuan LSTM menyimpan informasi dari sekuensial sebelumnya membuatnya sangat cocok untuk memahami konteks semantik dan sintaksis yang kompleks [10]. Selain itu, model LSTM mampu menangkap emosi atau sentimen yang tersembunyi di dalam teks, yang merupakan aspek penting dalam menganalisis konten yang berhubungan dengan kesehatan mental, di mana pola bahasa sering kali merefleksikan kondisi psikologis seseorang secara mendalam [11], [12]. Dengan kelebihan-kelebihan tersebut, LSTM merupakan pilihan yang optimal untuk menghasilkan prediksi yang akurat dan memahami seluk-beluk data tekstual yang digunakan dalam analisis ini.

Penerapan teknik *resampling* pada data tekstual, terutama ketika dianalisis menggunakan model LSTM masih belum banyak dieksplorasi. Hal ini disebabkan oleh kompleksitas data tekstual yang membutuhkan representasi dalam bentuk vektor atau *embedding* sebelum dapat diproses lebih lanjut secara efektif. Selain itu, pembuatan data sintetis berbasis teks membutuhkan pendekatan yang lebih hati-hati untuk menjaga konteks semantik, memastikan bahwa data sintetis tambahan tetap relevan dan bermakna. Dalam domain kesehatan mental, eksplorasi teknik *resampling* untuk data teks adalah sangat penting, mengingat tantangan yang terlibat dalam menganalisis teks yang menyampaikan emosi sering diwakili oleh kelompok data minoritas.

Penelitian ini bertujuan untuk mengevaluasi efektivitas teknik *resampling* dalam analisis sentimen kesehatan mental menggunakan model Bi-LSTM. Dengan membandingkan hasil performa model pada data sebelum dan setelah penerapan *resampling*, penelitian ini diharapkan dapat memberikan wawasan lebih mendalam terkait dampak teknik *class balancing* terhadap prediksi pada kelas minoritas serta potensi implikasinya dalam aplikasi nyata.

2. TINJAUAN PUSTAKA

2.1 Analisis Sentimen

Analisis sentimen merupakan salah satu cabang utama dari NLP (*Natural Language Processing*) yang fokus pada pemahaman opini, perasaan, atau emosi yang terkandung dalam teks. Tujuan utamanya adalah mengidentifikasi polaritas sentimen, seperti positif, negatif, atau netral, dalam sebuah konten, baik itu ulasan produk, media sosial, atau artikel berita. Teknik ini sangat berguna dalam berbagai bidang, mulai dari bisnis untuk analisis kepuasan pelanggan, politik untuk memahami opini publik, hingga bidang kesehatan untuk memantau kesejahteraan mental [2].

Namun, salah satu tantangan utama dalam analisis sentimen adalah masalah ketidakseimbangan data (*data imbalanced*). Data sering kali didominasi oleh sentimen tertentu yang membuat data sentiment pada kelas minoritas lainnya kurang terwakili. Ketidakseimbangan ini menyebabkan model *machine learning* cenderung lebih akurat pada kelas mayoritas dan mengabaikan kelas minoritas [3], [6].

2.2 LSTM (*Long-Short Term Memory*)

Long-Short Term Memory (LSTM) merupakan pengembangan lanjutan dari *Recurrent Neural Network* (RNN) yang dirancang untuk meningkatkan kemampuan dalam mengolah data sekuensial. Dengan adanya penambahan *memory cells*, LSTM memungkinkan jaringan untuk menyimpan dan mengelola informasi dalam jangka waktu yang lebih lama dibandingkan RNN biasa. *Memory cells* ini juga dilengkapi dengan mekanisme *gates* (gerbang) yang berfungsi untuk mengatur aliran informasi secara selektif, sehingga informasi yang relevan dapat dipertahankan, dan informasi yang tidak penting akan diabaikan. Pendekatan ini membantu LSTM mengatasi masalah *vanishing gradient*, sebuah kendala umum pada RNN saat memproses data sekuensial yang panjang. Oleh karena itu, LSTM lebih andal dan stabil untuk tugas-tugas yang memerlukan pemahaman konteks jangka panjang [10]. Struktur *gates* pada arsitektur LSTM, antara lain [13]:

1. *Forget Gate* (*ft*): menggunakan gabungan output pada waktu $t-1$ dan input pada waktu t kemudian melewati fungsi aktivasi sigmoid untuk menghasilkan nilai pada *range* 0 hingga 1.
$$ft = \sigma(W_f \times S_{t-1} + W_f \times X_t) \quad (1)$$
Ketika nilai *ft* semakin mendekati 0, *state* sebelumnya akan dilupakan (*forget*), sedangkan jika mendekati 1, maka *state* tersebut akan tetap dipertahankan.
2. *Input Gate* (*it*): memproses input baru dan output dari proses sebelumnya kemudian diteruskan ke lapisan sigmoid dan mengembalikan nilai antara 0 dan 1.
$$it = \sigma(W_i \times S_{t-1} + W_i \times X_t) \quad (2)$$
$$\tilde{C}_t = \tanh(W_c \times S_{t-1} + W_c \times X_t) \quad (3)$$
Proses ini digunakan untuk memperbarui *state* dengan menggabungkan *state* sebelumnya dengan kandidat *state* baru.
$$c_t = \left(it \times \tilde{C}_t + f_t \times c_{t-1}\right) \quad (4)$$
3. *Output Gate* (*ot*): mengatur seberapa besar informasi *state* yang dilewatkan untuk menghasilkan *state* baru (*ht*).
$$ot = \sigma(W_o \times S_{t-1} + W_o \times X_t) \quad (5)$$
$$ht = ot * tanh(ct) \quad (6)$$

Keterangan:
W_f, W_i, W_c, W_o= matriks bobot tiap *gate*
S_{t-1} = *hidden state* dari *state* sebelumnya
X_t = input saat ini
σ, $tanh$ = fungsi aktivasi (Sigmoid, Tangen Hiperbolik)

2.3 *Resampling*

Resampling merupakan salah satu teknik yang dapat digunakan untuk mengatasi permasalahan ketidakseimbangan data pada model yang akan dibangun [8]. Ketidakseimbangan kelas yang sering ditemui pada dataset dapat menyebabkan model lebih cenderung memprediksi kelas mayoritas dan mengabaikan kelas minoritas yang sebenarnya memiliki peranan penting. Untuk mengatasi hal ini, teknik *resampling* dapat dilakukan dalam dua bentuk, yaitu *oversampling* dan *undersampling*.

Pada *oversampling*, jumlah sampel dari kelas minoritas ditingkatkan dengan menggandakan data yang ada atau menggunakan teknik seperti SMOTE (*Synthetic Minority Over-sampling Technique*) yang menggunakan metode interpolasi linier antar-sampel aktual untuk menghasilkan data sintetik dengan cara mengambil nilai di antara dua sampel data asli, sehingga menciptakan data baru yang diharapkan merepresentasikan distribusi kelas minoritas lebih baik. Sebaliknya, pada *undersampling*, jumlah sampel dari kelas mayoritas dikurangi agar distribusi kelas menjadi lebih seimbang [9]. Penerapan teknik-teknik *resampling* ini bertujuan untuk meningkatkan akurasi model dalam mengidentifikasi kelas minoritas, yang sangat penting dalam konteks deteksi gangguan kesehatan mental yang jarang teridentifikasi.

2.4 Metrik Pengujian

Untuk mengevaluasi performa model yang telah dibangun, sejumlah metrik pengujian telah dipilih sebagai acuan untuk mengukur akurasi dan efektivitas model dalam melakukan klasifikasi data sentimen secara menyeluruh. Adapun metrik yang digunakan meliputi [14]:

1. *Accuracy*, adalah persentase prediksi yang benar terhadap keseluruhan data uji. Akurasi memberikan gambaran umum tentang seberapa baik model dapat mengklasifikasikan data. Akurasi dihitung dengan rumus:
$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} = \frac{TP + TN}{P + N} \quad (7)$$
2. *Precision*, mengukur keandalan model dalam memberikan prediksi positif yang benar. Metrik ini sangat penting dalam menangani ketidakseimbangan data. Presisi dihitung dengan rumus:
$$Precision = \frac{TP}{TP+FP} \quad (8)$$
3. *Recall/ Sensitivity*, disebut juga sebagai TPR (*True Positive Rate*), mengukur seberapa banyak data positif yang benar terdeteksi oleh model dibandingkan dengan seluruh data aktual yang positif. Metrik ini penting untuk memastikan kelas minoritas tidak terabaikan. *Recall* dihitung dengan rumus:
$$Recall = \frac{TP}{TP+FN} \quad (9)$$
4. *F1-score,* merupakan nilai rata-rata harmonis antara *precision* dan *recall*. *F1-score* dihitung dengan rumus:
$$F1\ score = \frac{2 \times recall \times precision}{recall + precision} \quad (10)$$

Istilah TP (*True Positive*), TN (*True Negative*), FP (*False Positive*), dan FN (*False Negative*) pada rumus diatas, merepresentasikan hasil klasifikasi yang diperoleh dari model yang telah dibangun. Untuk lebih memperjelas, hasil-hasil klasifikasi ini dapat divisualisasikan melalui *Confusion Matrix*, penataan tabel yang digunakan untuk menggambarkan evaluasi kinerja model klasifikasi dengan membandingkan hasil prediksi dengan nilai sebenarnya [15].

Tabel 1. *Confusion Matrix* untuk Data 3 Label

		Predicted		
		Normal	*Depression*	*Suicidal*
Actual	*Normal*	TN	FN	FN
	Depression	FD	TD	FD
	Suicidal	FS	FS	TS

3. METODE PENELITIAN

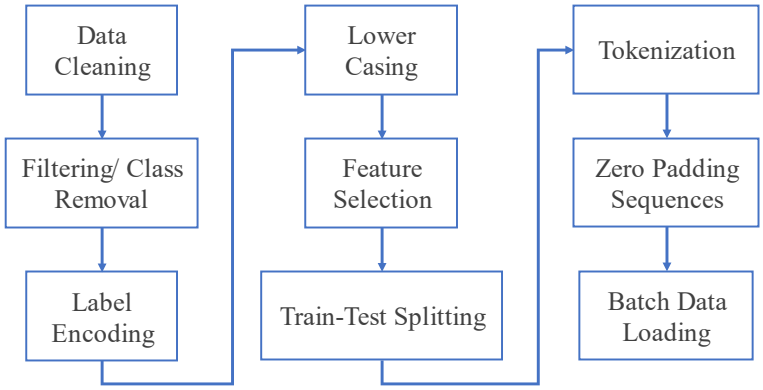
3.1 Deskripsi Dataset

Dataset yang digunakan diperoleh dari kaggle.com dan merupakan gabungan beberapa dataset hasil kurasi dari berbagai sumber data mentah. Data tersebut telah melalui proses pembersihan, penghapusan data yang tidak relevan, pelabelan, dan penggabungan untuk menghasilkan kumpulan data yang dapat dimanfaatkan. Dataset ini terdiri dari 53.043 baris data, dengan dua kolom utama, yaitu:

1. Kolom "*statement*"
   Berisi data teks sentimen dalam bentuk kalimat. Kolom ini merupakan fitur utama yang berisi informasi sentimen yang akan dianalisis.
2. Kolom "*status*"
   Berisi label kelas/ kategori sentimen, yaitu salah satu dari tujuh kelas: Normal, *Depression*, *Suicidal*, *Anxiety*, *Stress*, *Bipolar*, dan *Personality disorder*. Kolom ini akan digunakan sebagai label target dalam proses pelatihan dan evaluasi model.

3.2 *Preprocessing* Data

Untuk mendukung proses pelatihan model, dilakukan beberapa tahapan *preprocessing* yang dirangkum dalam Gambar 1 berikut.



Gambar 1. Tahapan *Preprocessing* Data

3.2.1 *Data Cleaning*

Pembersihan data adalah tahapan pertama yang wajib dilakukan untuk dataset yang sudah dikumpulkan. Untuk dataset yang diperoleh dari kaggle berisi 53.043 baris data. Penghapusan baris data yang mengandung nilai kosong (NaN) dilakukan karena data tersebut dianggap tidak *valid* untuk dianalisis dan dapat mengganggu proses *training* model. Selanjutnya, dilakukan pengecekan dan penghapusan duplikasi dengan menghapus baris yang memiliki isi kolom yang persis sama untuk mencegah *overfitting* akibat data berulang yang tidak memberikan informasi baru pada model.

Tabel 2. Perbandingan Jumlah Data Sebelum dan Setelah *Cleaning*

Before	*After*
`raw_data.shape` `(53043, 3)`	`df.sample(frac = 1).head()` `Total data (rows): 52681`

3.2.2 *Data Balancing with Class Removal*

Pengecekan jumlah data tiap kelas dilakukan untuk menghindari model yang terlalu condong ke salah satu atau beberapa kelas dengan jumlah data mayoritas, dan mengabaikan kelas dengan jumlah

data yang lebih sedikit. Ketidakseimbangan ini dapat memicu terjadinya *overfitting* dan menurunkan akurasi model [16].

Hasil pengecekan menunjukkan adanya ketidakseimbangan jumlah data antar kelas, dengan rentang yang cukup jauh, mulai dari sekitar 16.000 data untuk kelas "Normal" hingga hanya sekitar 1.000 data untuk kelas "*Personality disorder*". Hal ini mengindikasikan adanya ketidakseimbangan data pada kelas "*Anxiety*", "*Bipolar*", "*Stress*", dan "*Personality disorder*".

Untuk menyeimbangkan data tiap kelas, salah satu metode yang dipilih adalah *Filtering* atau *Class Removal*, yaitu dengan menghapus data pada kelas minoritas dan hanya menyisakan data dari kelas mayoritas. Pendekatan ini bertujuan agar model tetap dapat mempelajari pola dari dataset yang memiliki distribusi kelas yang lebih seimbang, sehingga dapat meningkatkan akurasi prediksi tanpa adanya bias terhadap kelas mayoritas [17].

Tabel 3. Perbandingan Jumlah Data Sebelum dan Setelah *Class Removal*

Before

status	count
Normal	16343
Depression	15404
Suicidal	10652
Anxiety	3841
Bipolar	2777
Stress	2587
Personality disorder	1077

dtype: int64

After

status	count
Normal	16343
Depression	15404
Suicidal	10652

dtype: int64

Penghapusan terhadap kelas "*Anxiety*", "*Bipolar*", "*Stress*", dan "*Personality disorder*" dilakukan berdasarkan pertimbangan distribusi data yang sangat timpang, di mana keempat kelas tersebut hanya mewakili kurang dari 15% dari total dataset. Ketimpangan ini membuat kelas-kelas tersebut tidak cukup representatif untuk dilatih secara efektif menggunakan model *deep learning*. Selain itu, kelas-kelas tersebut memiliki potensi tumpang tindih secara semantik, yang dapat menimbulkan ambiguitas dalam klasifikasi dan memperbesar risiko kesalahan prediksi. Oleh karena itu, penelitian ini difokuskan pada tiga kelas dengan distribusi data yang lebih dominan dan relevan secara klinis, yaitu "Normal", "*Depression*", dan "*Suicidal*".

3.2.3 *Label Encoding*

Setiap kelas yang masih bersifat kategorikal (teks) akan diubah menjadi label numerik dalam bentuk bilangan bulat, dengan label 0 untuk Normal, 1 untuk *Depression*, dan 2 untuk *Suicidal*.

	Class Name	Value
0	Normal	0
1	Depression	1
2	Suicidal	2

Gambar 2. Label Hasil *Encoding*

3.2.4 *Feature Selection* dan *Lower Casing*

Kolom yang akan digunakan sebagai label atau kelas adalah kolom "Status". Setelah pemilihan kolom tersebut, hanya kolom "*statement"* dan kolom "*Unnamed: 0*" (yang berfungsi sebagai indeks) yang tersisa. Oleh karena itu, fitur independen (X) hanya akan menggunakan kolom "*statement*".

```
x_data = df['statement'].apply(lambda x: str(x).lower())
y_data = df['status']
```

Gambar 3. Pemilihan Fitur Independen (x) dan Dependen (y)

Selain itu, untuk meminimalkan perbedaan yang tidak relevan dalam analisis teks, seluruh data teks yang terdapat dalam kolom "*statement*" perlu disamakan terlebih dahulu dalam hal huruf besar dan kecilnya. Hal ini dilakukan untuk memastikan konsistensi dalam pemrosesan teks dan menghindari perbedaan yang tidak perlu.

3.2.5 *Train-Test Splitting*

Meskipun tahapan *preprocessing* data belum sepenuhnya selesai, pembagian data menjadi set *training* dan *testing* dilakukan terlebih dahulu dengan beberapa pertimbangan sebagai berikut:

1. Mempertahankan distribusi data yang seimbang antara data *training* dan *testing*
2. Mencegah kebocoran data, di mana data *training* dan *testing* akan ditokenisasi secara terpisah

```
x_train, x_test, y_train, y_test = train_test_split(
    x_data.values,
    y_data.values,
    test_size = 0.3,
    random_state = 1000
)
```

Gambar 4. *Train-Test Split*

Dataset akan dibagi dengan rasio 7:3, di mana 70% data akan digunakan untuk *training* dan 30% sisanya akan digunakan sebagai data *testing*. Rasio ini memberikan jumlah data *testing* yang cukup besar untuk mengevaluasi performa model secara lebih representatif, tanpa mengorbankan jumlah data *training* yang dibutuhkan untuk pembelajaran fitur secara optimal. Rasio 7:3 juga umum digunakan dalam penelitian NLP dengan dataset yang relatif besar seperti pada penelitian ini.

Pembagian ini dilakukan dengan menggunakan *random state* 1.000 untuk memastikan variasi pengacakan yang konsisten setiap kali kode dijalankan.

3.2.6 *Tokenization*

Data teks sentimen memerlukan perlakuan khusus yang berbeda dengan data teks kategorikal yang dapat langsung dilakukan *encoding*. Untuk data teks sentimen, proses tokenisasi diperlukan untuk mengonversi teks menjadi rangkaian angka yang dapat dipahami dan diproses model *machine learning*.

```
tokenizer = Tokenizer(oov_token='UNK', lower = True)
tokenizer.fit_on_texts(x_data.values)

x_train_tokenized = tokenizer.texts_to_sequences(x_train)
x_test_tokenized = tokenizer.texts_to_sequences(x_test)
```

Gambar 5. Tokenisasi Data *Training* dan *Testing*

Proses tokenisasi dilakukan dengan membuat sebuah kamus (*vocabulary*) dari seluruh kata unik yang muncul dalam dataset *training*. Setiap kata dalam kamus tersebut diberi ID numerik unik secara berurutan berdasarkan urutan kemunculannya atau frekuensi tertentu. Sebagai contoh, jika kata "*happy*" berada di posisi ke-37 dalam kamus, maka ia akan dikonversi menjadi token 37. Dengan demikian, kalimat "*I feel so happy today*" akan diubah menjadi urutan angka {2, 15, 1, 37, 8} di mana setiap angka merepresentasikan indeks kata dalam kamus yang telah dibentuk dari data *training*.

3.2.7 *Zero Padding Sequence*

Karena model *machine learning*, terutama yang berbasis jaringan saraf (*neural networks*), seringkali mengharapkan input dalam bentuk sekuensial dengan panjang yang seragam, proses *padding* diperlukan untuk memastikan ukuran data yang dimasukkan sesuai dengan panjang yang konsisten [18].

Padding bertujuan untuk menambahkan nilai yang biasanya berupa angka 0, pada data dengan panjang lebih pendek sehingga sekuensial data tersebut memiliki panjang yang seragam, sesuai dengan panjang maksimal yang telah ditentukan [19]. Panjang maksimal ini biasanya ditentukan berdasarkan sekuensial yang dihasilkan dari proses tokenisasi, yaitu dengan memilih panjang data yang paling banyak mengandung token atau ukuran terpanjang.

```
x_train_tokenized_padded = pad_sequences(x_train_tokenized, maxlen = max_len)
x_test_tokenized_padded = pad_sequences(x_test_tokenized, maxlen = max_len)
```

Gambar 6. *Zero Padding* pada Data *Training* dan *Testing*

Sebagai contoh, jika kita memiliki dua sekuensial, seperti {2, 15, 1, 37, 8} dan {3, 29, 15}, dan panjang maksimal yang ditentukan adalah 5, maka sekuensial pertama tetap tidak berubah, sementara sekuensial kedua akan diubah menjadi {3, 29, 15, 0, 0} untuk memenuhi panjang yang seragam.

3.2.8 *Batch Data Loading*

Proses *batch data loading* dilakukan untuk memungkinkan model melakukan pelatihan dengan memproses beberapa data sekaligus dalam satu *batch*. Teknik ini bertujuan untuk mengurangi penggunaan memori dan waktu yang diperlukan selama proses pelatihan dan pengujian model, dengan cara memecah data besar menjadi bagian-bagian yang lebih kecil. Hal ini meningkatkan efisiensi serta mempercepat konvergensi model, karena proses pelatihan dapat dilakukan lebih cepat dengan beban memori yang lebih rendah.

```
batch_size = 24
train_dataloader = DataLoader(CustomizedDataset(x_train_tokenized_padded, y_train), shuffle = True, batch_size = batch_size)
test_dataloader = DataLoader(CustomizedDataset(x_test_tokenized_padded, y_test), shuffle = True, batch_size = batch_size)
```

Gambar 7. Pembagian *Batch* pada Data *Training* dan *Testing*

Pada implementasi model menggunakan PyTorch, pembagian *batch* harus dilakukan secara manual oleh pengguna. PyTorch menyediakan “DataLoader” sebagai alat untuk mempermudah proses ini, memungkinkan pembagian data training ke dalam *batch* yang lebih kecil, baik dengan pengacakan (*shuffling*) maupun tanpa pengacakan, sesuai kebutuhan model yang sedang dilatih [20].

3.3 Arsitektur Model

Setelah data siap digunakan, tahap selanjutnya adalah membangun arsitektur model *deep learning* yang akan digunakan untuk klasifikasi sentiment, yaitu Bi-LSTM, seperti pada Gambar 8 berikut:

```
SentimentAnalysisModel(
  (embd): Embedding(54741, 128, padding_idx=0)
  (lstm): LSTM(128, 32, bidirectional=True)
  (linear): Linear(in_features=403200, out_features=3, bias=True)
  (dropout): Dropout(p=0.35, inplace=False)
)
```

Gambar 8. Arsitektur Model *Bidirectional-LSTM*

1. *Embedding*
   Lapisan *embedding* digunakan untuk menangkap representasi vektor dari data token unik dari dataset yang akan diakses menggunakan indeks [21]. Model menerima input berupa token dengan ukuran kosakata sebesar 54.741 token unik, yang akan direpresentasikan dalam vektor berdimensi 128. Untuk memastikan panjang input seragam, *padding* dengan nilai 0 ditambahkan pada sekuensial yang lebih pendek. Teknik ini memungkinkan pengolahan data teks dalam format yang dapat diproses oleh *layer* berikutnya.

2. LSTM (*Long-Short Term Memory*)
   Lapisan utama dalam model ini adalah LSTM, yang dirancang untuk menangani data sekuensial berdimensi 128. Model menggunakan 32 unit neuron dalam setiap *hidden layer*-nya. *Layer* ini bekerja secara *bidirectional*, sehingga data diproses baik dalam arah *forward* maupun *backward*, memungkinkan model menangkap informasi dari konteks sebelum dan sesudahnya dalam data sekuensial.
3. *Linear* (*Fully Connected Layer*)
   Setelah data melewati lapisan LSTM, hasilnya diubah menjadi format *flattened* dengan ukuran input sebesar 403.200 (dihitung dari hasil 2×32×6300). *Layer* ini menghasilkan output berupa probabilitas untuk tiga kelas: *Normal, Depression,* dan *Suicidal*. Proses klasifikasi pada *layer* ini dilakukan dengan fungsi aktivasi Softmax yang sesuai untuk *multi-class*.
4. *Dropout*
   Lapisan *Dropout* diterapkan selama pelatihan model untuk mencegah *overfitting* dengan cara menghilangkan beberapa unit neuron secara acak pada setiap langkah *forward*. Proses ini dilakukan dengan probabilitas yang diambil dari distribusi Bernoulli, di mana elemen-elemen yang diubah menjadi nol dipilih secara independen untuk setiap langkah *forward*. Hal ini berarti bahwa setiap unit neuron memiliki peluang yang sama untuk dinonaktifkan, sementara sisanya tetap aktif [22]. Dalam arsitektur yang digunakan, *Dropout* ditambahkan dengan tingkat pembuangan neuron sebesar 35% untuk menjaga keseimbangan antara regularisasi dan kemampuan belajar model.

3.4 Pembuatan Dan Pelatihan Model

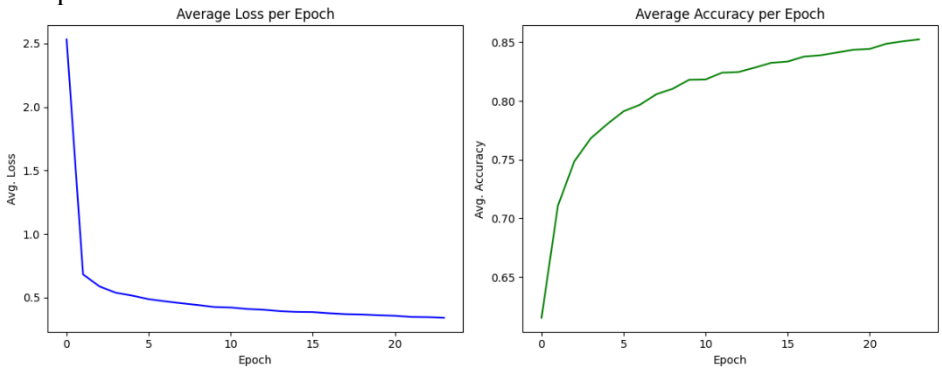
Berikut adalah beberapa prosedur yang diterapkan selama proses *training* model:

1. *Batch Size*
   Ukuran *batch* yang digunakan adalah 24, yang dipilih berdasarkan pertimbangan antara penggunaan memori dan waktu *training* yang efisien. Dengan *batch size* sebesar 24, model dapat menjalani pelatihan dengan memanfaatkan sumber daya komputasi secara optimal tanpa mengalami kekurangan memori.
2. *Epochs*
   Model akan dilatih selama 24 *epochs*. Pada titik ini, akurasi model mulai stabil, dan tidak ada peningkatan yang signifikan meskipun pelatihan dilanjutkan lebih lama. Oleh karena itu, 24 *epoch* dianggap sebagai jumlah yang optimal untuk memastikan model dapat belajar secara efektif tanpa mengalami *overfitting*.
3. *Loss Function*
   Model akan dievaluasi dengan *CrossEntropyLoss*, yang merupakan *loss function* yang tepat untuk masalah klasifikasi *multi-class* [23]. *CrossEntropyLoss* menghitung selisih antara distribusi probabilitas yang diprediksi dan distribusi yang benar (label kelas), kemudian memberikan penalti yang lebih besar saat model membuat prediksi yang lebih salah, sehingga mendorong model untuk belajar dengan lebih efektif [24].
4. *Optimizer*
   Model menggunakan Adagrad sebagai *optimizer*, dengan *learning rate* sebesar 0.02 dan *weight decay* sebesar 0.0001. Adagrad adalah salah satu *optimizer* yang memiliki kemampuan untuk menyesuaikan *learning rate* untuk setiap parameter secara otomatis. Dengan pengaturan ini, *optimizer* memberikan *learning rate* yang lebih besar untuk parameter yang jarang diperbarui, dan lebih kecil untuk parameter yang sering diperbarui, yang bisa mempercepat proses konvergensi. *Weight decay* diterapkan untuk mencegah *overfitting* dengan memberikan penalti terhadap parameter yang terlalu besar [25].
5. *Backpropagation*
   Setelah nilai *loss* dihitung untuk setiap *batch*, fungsi *backward* [23] dipanggil untuk menghitung gradien dari nilai *loss* terhadap parameter model (bobot/*weight*) melalui algoritma *backpropagation*.

Proses ini memungkinkan model untuk belajar dengan menyesuaikan bobotnya ke arah yang mengurangi nilai *loss*. Gradien yang dihitung akan digunakan oleh *optimizer* untuk memperbarui bobot model dengan efektif.

3.5 Evaluasi Performa Model Selama *Training*

Selama proses *training* model, nilai *loss* model mengalami penurunan secara bertahap hingga mencapai nilai 0,34, dengan penurunan paling tajam terjadi pada *epoch* kedua, dari 2,53 menjadi 0,61. Selain itu, akurasi model juga mengalami peningkatan yang signifikan pada *epoch* kedua, dengan kenaikan mencapai 10%.

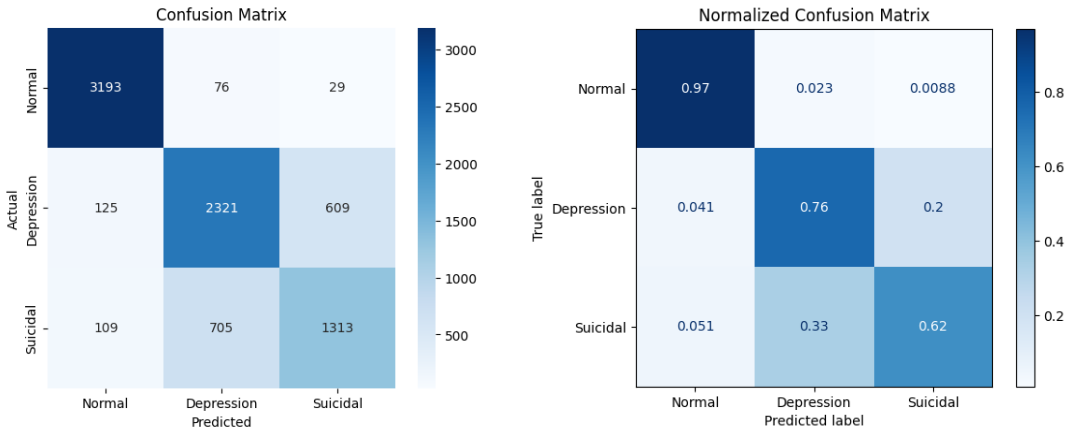


Gambar 9. Grafik Nilai *Loss* dan Akurasi Selama *Training* Model

Model kemudian diuji dan dievaluasi menggunakan berbagai metrik evaluasi, yang dapat dilihat pada tabel di bawah ini:

Tabel 4. Evaluasi Performa Model Hasil *Testing*

	Normal	*Depression*	*Suicidal*	*Accuracy*
Precision	0.931719	0.748227	0.672988	80.5071%
Recall/ Sensitivity	0.968163	0.759738	0.617301	80.5071%
F1-score	0.949591	0.753939	0.643943	80.5071%
Data size	3298	3055	2127	8480



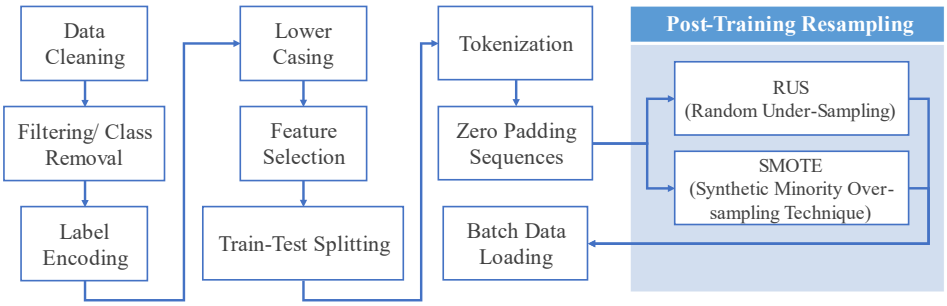
Gambar 10. Evaluasi Performa dengan *Confusion Matrix*

Berdasarkan hasil dari tabel metrik evaluasi dan *Confusion Matrix* yang ditampilkan, model menunjukkan kemampuan yang baik dalam mengklasifikasikan data pada kelas "Normal" dengan nilai

F1-score mencapai 95%. Hal ini sejalan dengan proporsi jumlah data pada kelas tersebut yang lebih banyak dibandingkan dengan kelas lainnya. Namun, untuk kelas minoritas seperti "*Depression*" dan "*Suicidal*", model masih menghasilkan sejumlah kesalahan klasifikasi, yang berdampak pada akurasi keseluruhan model yang tercatat sebesar 80,5%.

3.6 Implementasi *Data Balancing* Dengan *Resampling*

Melihat rendahnya akurasi model dalam memprediksi kelas minoritas (*Depression* dan *Suicidal)*, percobaan lebih lanjut dilakukan untuk menyeimbangkan jumlah data pada kelas-kelas minoritas menggunakan teknik *resampling* yang meliputi dua pendekatan utama, yaitu *undersampling dan oversampling*. Kedua metode ini diterapkan pada set *training* setelah tahapan *pad sequences* dilakukan.



Gambar 11. Tahapan *Resampling* Data (*Post-Training*)

Undersampling dilakukan menggunakan metode RUS (*Random Under-Sampling*), yaitu teknik pembuangan data pada kelas mayoritas secara acak hingga mencapai keseimbangan jumlah data pada keseluruhan kelas [9].

```
undersampler = RandomUnderSampler(random_state=42)
x_train_resampled, y_train_resampled = undersampler.fit_resample(x_train_tokenized_padded, y_train)

# Verify class distribution
print("Before:", Counter(y_train))
print("After :", Counter(y_train_resampled))
```

```
Before: Counter({'Normal': 11387, 'Depression': 10847, 'Suicidal': 7445})
After : Counter({'Depression': 7445, 'Normal': 7445, 'Suicidal': 7445})
```

Gambar 12. Hasil *Resampling* dengan RUS (*Random Under-Sampling*)

Oversampling dilakukan dengan metode SMOTE (*Synthetic Minority Over-sampling Technique*), yang menciptakan data sintetis pada kelas minoritas sehingga jumlah data pada kelas tersebut seimbang dengan kelas mayoritas [9].

```
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(x_train_tokenized_padded, y_train)

# Verify class distribution
print("Before SMOTE:", Counter(y_train))
print("After SMOTE:", Counter(y_train_smote))
```

```
Before SMOTE: Counter({'Normal': 11387, 'Depression': 10847, 'Suicidal': 7445})
After SMOTE: Counter({'Depression': 11387, 'Normal': 11387, 'Suicidal': 11387})
```

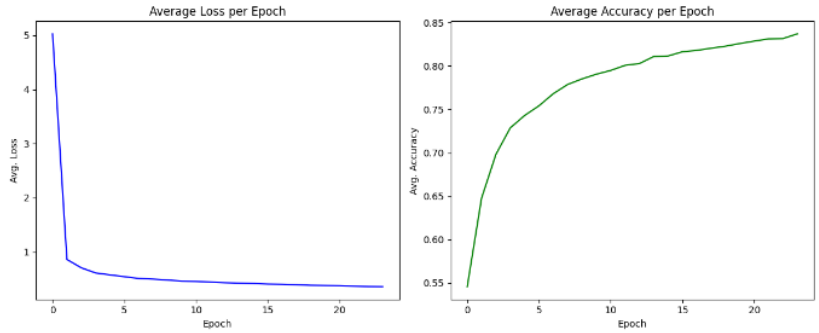
Gambar 13. Hasil *Resampling* dengan SMOTE

Parameter *random state* diatur sebesar 42 untuk menjaga konsistensi dalam pengacakan, serta memastikan proses pembuangan dan penambahan data berlangsung secara reproduktif.

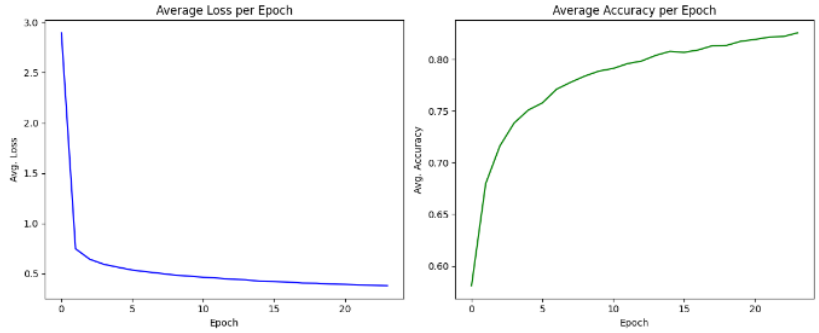
4. HASIL DAN PEMBAHASAN

4.1 Analisis Kinerja Model Dengan *Resampling*

Berdasarkan grafik yang dihasilkan selama proses *training* model, terlihat bahwa tidak ada perbedaan yang signifikan dalam bentuk garis grafik untuk *loss* dan akurasi antara data sebelum dan setelah penerapan teknik *resampling*. Hal ini menunjukkan bahwa meskipun teknik *resampling* berhasil menyeimbangkan distribusi kelas, tidak ada perubahan besar pada dinamika *training* model, baik dalam hal konvergensi *loss* maupun perkembangan akurasi.



Gambar 14. Grafik Nilai *Loss* dan Akurasi *Training* pada Data Hasil RUS



Gambar 15. Grafik Nilai *Loss* dan Akurasi *Training* pada Data Hasil SMOTE

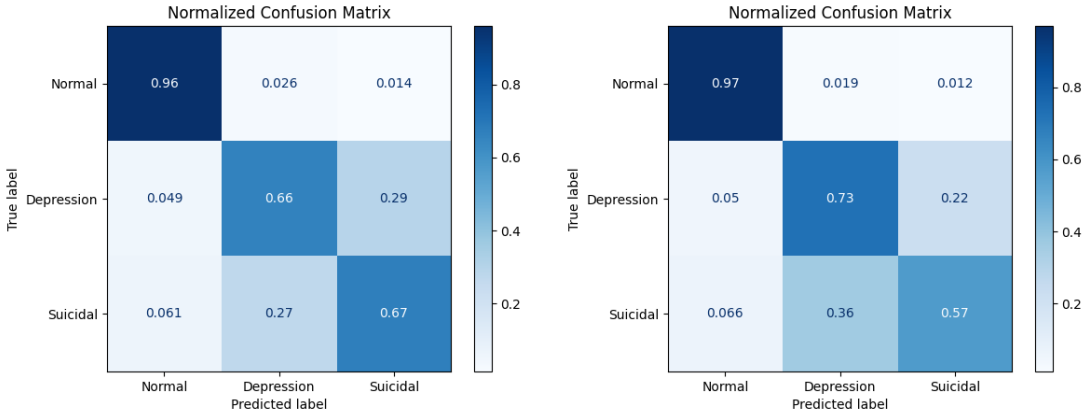
Meskipun grafik *loss* dan akurasi tidak menunjukkan perubahan signifikan, hasil pada evaluasi kelas minoritas dan keseluruhan akurasi dapat menunjukkan adanya efek *resampling* dalam meningkatkan deteksi untuk kelas minoritas.

4.2 Evaluasi Model Setelah *Resampling*

Rangkuman evaluasi hasil model setelah diterapkan teknik *resampling* dapat dilihat pada tabel di bawah ini:

Tabel 5. Evaluasi Perbandingan Performa Model dengan Data Hasil *Resampling*

Resampling Method	*Training*		*Testing*			
	Final Loss	*Final Accuracy*	*Accuracy*	*F1-Score*		
				Normal	*Depression*	*Suicidal*
Undersampling dengan *RUS*	0.355186	0.837027	0.778538	0.938925	0.701381	0.635597
Oversampling dengan *SMOTE*	0.381177	0.825737	0.782311	0.942249	0.599476	0.726038



Gambar 16. Perbandingan *Confusion Matrix* pada Data Hasil RUS (kiri) vs SMOTE (kanan)

Hasil evaluasi model setelah menerapkan teknik *resampling* menunjukkan bahwa ada peningkatan *F1-score* pada kelas "*Suicidal*" dengan teknik *undersampling*, dari 64,4% menjadi 72,6%. Namun untuk kelas lainnya, yaitu "*Depression*" dan "Normal", justru mengalami penurunan baik dengan teknik *undersampling* maupun *oversampling*. Hasil ini menunjukkan bahwa meskipun teknik *resampling* berhasil meningkatkan prediksi untuk kelas "*Suicidal*", penurunan performa pada kelas-kelas lainnya mengindikasikan bahwa model kesulitan dalam menyeimbangkan antara ketepatan prediksi untuk kelas minoritas dan mayoritas.

Selain itu, akurasi akhir *testing* model juga mengalami penurunan setelah penerapan *resampling*. Pada awalnya, model dapat mencapai akurasi sebesar 80,5%. Namun, setelah teknik *undersampling* diterapkan, akurasi menurun menjadi 77,85%, dan setelah menerapkan teknik *oversampling*, akurasi sedikit membaik menjadi 78,23%. Hal ini menunjukkan adanya *trade-off* antara meningkatkan kemampuan model dalam mendeteksi kelas minoritas dan mempertahankan prediksi yang akurat untuk kelas mayoritas. Penurunan ini juga mencerminkan bahwa upaya menyeimbangkan distribusi kelas tidak selalu berdampak positif terhadap keseluruhan performa, terutama ketika data mayoritas yang kaya variasi dikurangi (seperti pada RUS), atau ketika data sintetis yang ditambahkan tidak sepenuhnya merepresentasikan kompleksitas semantik data asli (seperti pada SMOTE). Dengan kata lain, model menjadi lebih "peka" terhadap kelas minoritas, namun mengorbankan ketepatan terhadap kelas yang sudah dipahami dengan baik sebelumnya.

Untuk memahami lebih dalam tantangan model dalam mengenali kelas minoritas, dilakukan analisis terhadap beberapa prediksi yang salah (*misclassification*). Sebagai contoh, pernyataan *"Akhir-akhir ini aku merasa hidup tidak berarti, tapi aku tetap tersenyum agar orang lain tidak khawatir."* diprediksi sebagai "Normal", padahal label sebenarnya adalah "*Depression*". Kalimat seperti ini menunjukkan bahwa ekspresi kondisi mental sering kali dinyatakan secara tersirat, sehingga menyulitkan model dalam membedakan dengan pernyataan netral yang sebenarnya tidak mengandung gejala psikologis tertentu.

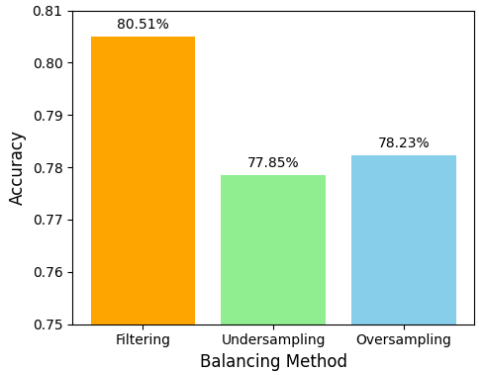
```
predict_sentiment(input("Masukkan sentimen Anda:\n"))

Masukkan sentimen Anda:
Akhir-akhir ini aku merasa hidup tidak berarti, tapi aku tetap tersenyum agar orang lain tidak khawatir.
('Normal', 0.956535279750824)
```

Gambar 17. Contoh *Misclassification* Sentimen "*Depression*" Menjadi "Normal"

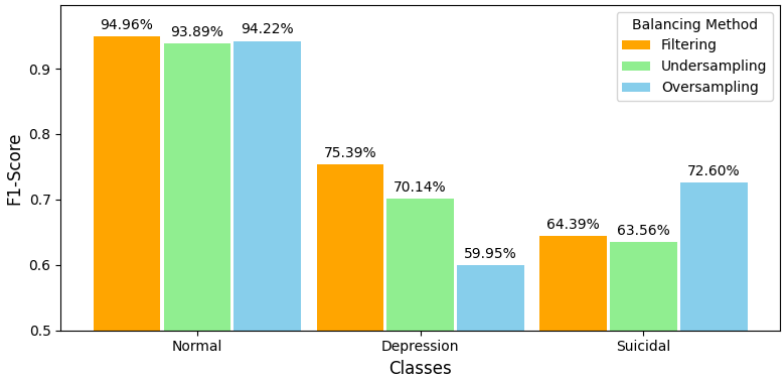
4.3 Perbandingan Hasil Performa Model

Untuk memahami pengaruh penerapan teknik *resampling* terhadap performa model, perbandingan hasil evaluasi dilakukan antara data sebelum *resampling*, setelah *undersampling*, dan setelah *oversampling*. Grafik berikut menyajikan perubahan nilai metrik utama, seperti akurasi dan nilai *F1-score*, untuk memberikan gambaran yang jelas mengenai efektivitas penerapan teknik *resampling*.



Gambar 18. Grafik Perbandingan Akurasi dengan 3 Teknik *Class Balancing*

Analisis performa model menunjukkan bahwa akurasi tertinggi diperoleh pada model yang dilatih dengan data hasil *filtering* tanpa penerapan teknik *resampling*, yakni sebesar 80,5%. Hal ini mengindikasikan bahwa model memiliki kemampuan yang lebih baik dalam menangkap pola dari distribusi data awal. Di sisi lain, akurasi model menurun ketika data hasil *resampling* digunakan, dengan nilai 78,2% pada *oversampling* menggunakan SMOTE dan 77,8% pada *undersampling* menggunakan RUS. Penurunan akurasi ini dapat dikaitkan dengan perubahan distribusi data, di mana *resampling* dapat memperkenalkan *noise* atau mengurangi variasi data pada kelas mayoritas.



Gambar 19. Grafik Perbandingan *F1-Score* dengan 3 Teknik *Class Balancing*

Analisis terhadap *F1-score* menunjukkan dinamika performa sebagai berikut:

1. Kelas "*Normal*"
   Tidak terdapat perubahan signifikan pada nilai *F1-score* untuk kelas ini setelah diterapkan teknik *resampling*. Hal ini wajar mengingat kelas "Normal" memiliki jumlah data yang dominan, sehingga model sudah mampu memprediksi kelas ini dengan baik bahkan sebelum *resampling* dilakukan.
2. Kelas "*Depression*"
   Metode *undersampling* dengan RUS memberikan dampak negatif pada nilai *F1-score* untuk kelas ini, yang menandakan bahwa penghapusan data secara acak pada kelas mayoritas dapat mengurangi kualitas representasi data. Di sisi lain, model yang dilatih dengan data hasil *oversampling* dengan

SMOTE penurunan performa yang lebih signifikan, dengan nilai *F1-score* yang turun drastis dari 73,4% menjadi 59,9%. Penurunan ini dapat diindikasikan sebagai efek samping dari penciptaan data sintetik yang mungkin tidak sepenuhnya merepresentasikan distribusi data asli kelas.

3. Kelas "*Suicidal*"
   Peningkatan nilai *F1-score* diamati setelah diterapkannya *oversampling* dengan SMOTE. Hal ini menunjukkan bahwa penambahan data sintetik melalui SMOTE membantu model mengenali pola pada kelas minoritas ini. Namun, peningkatan ini tidak terjadi pada *undersampling* RUS, di mana penghapusan data mayoritas tampaknya mengurangi kemampuan model untuk mempelajari pola-pola penting dari data.

5. KESIMPULAN

Dari analisis di atas, dapat disimpulkan bahwa meskipun teknik *oversampling* dengan SMOTE memberikan manfaat untuk meningkatkan *F1-score* pada kelas minoritas tertentu (seperti kelas "*Suicidal*"). Namun, hal ini tetap memiliki *trade-off*, yaitu potensi penurunan performa pada kelas lain (misalnya kelas "*Depression*"). Sementara itu, teknik *undersampling* dengan RUS cenderung memberi efek negatif secara keseluruhan, terutama dengan menurunkan *F1-score* pada kelas-kelas minoritas.

Teknik *undersampling* dengan RUS berkontribusi pada penurunan performa model karena penghapusan data secara acak dari kelas mayoritas. Meskipun dapat menyeimbangkan distribusi data, pendekatan ini sering kali menghilangkan informasi penting yang justru diperlukan untuk membedakan pola antar-kelas. Akibatnya, model kehilangan konteks penting dan kemampuan generalisasi terhadap data baru berkurang.

Di sisi lain, teknik *oversampling* dengan SMOTE, yang secara teori dirancang untuk memperbaiki representasi kelas minoritas, juga memiliki kelemahan. Penciptaan data sintetik melalui interpolasi linier antar-sampel aktual terkadang gagal merepresentasikan seluruh keragaman data secara akurat. Hal ini dapat menyebabkan model mengalami *local overfitting*, di mana model terlalu mengandalkan pola dari data sintetik tanpa memahami distribusi data yang lebih luas. Akibatnya, meskipun terdapat peningkatan pada kelas "*Suicidal*", performa model pada kelas "*Depression*" justru menurun secara signifikan.

Keterbatasan utama yang dihadapi terletak pada distribusi label dalam dataset yang sangat tidak seimbang, serta keputusan untuk menghapus sebagian kelas dengan jumlah data yang sangat sedikit. Meskipun langkah ini diambil untuk menjaga kestabilan model, ruang lingkup analisis menjadi terbatas pada kelas-kelas dominan, sementara kelas lainnya yang memiliki nilai penting dalam konteks kesehatan mental tidak terwakili secara optimal. Selain itu, dataset yang digunakan bersumber dari data kurasi terbuka, sehingga kualitas konsistensi anotasi dan keberagaman data tidak dapat dikendalikan sepenuhnya.

6. SARAN

Berdasarkan hasil penelitian ini, terdapat beberapa saran yang dapat dijadikan acuan untuk pengembangan penelitian di masa mendatang dalam analisis sentimen terhadap data teks yang tidak seimbang:

1. Memanfaatkan metode *resampling* yang lebih adaptif, seperti ADASYN (*Adaptive Synthetic Sampling*), *Borderline*-SMOTE, atau metode *hybrid* lainnya, yang dapat menghasilkan data sintetis berdasarkan distribusi yang lebih sesuai dengan kompleksitas kelas minoritas.
2. Menerapkan pendekatan *ensemble learning*, seperti *Balanced Random Forest* atau *Easy Ensemble*, yang dapat menjaga performa model terhadap kelas mayoritas sekaligus meningkatkan sensitivitas model terhadap kelas minoritas.

Analisis ini menekankan bahwa penerapan teknik *data balancing* harus disesuaikan dengan karakteristik dataset dan tujuan akhir analisis, sambil mempertimbangkan dengan cermat risiko

overfitting pada kelas minoritas dengan distribusi kelas yang sangat timpang. Dengan menerapkan saran-saran tersebut, diharapkan sistem klasifikasi dapat mencapai performa yang lebih optimal dan memberikan manfaat praktis yang lebih luas dalam berbagai skenario dunia nyata.

DAFTAR PUSTAKA

[1] T. Zhang, K. Yang, S. Ji, and S. Ananiadou, "Emotion fusion for mental illness detection from social media: A survey," *Information Fusion*, vol. 92, pp. 231–246, Apr. 2023, doi: 10.1016/j.inffus.2022.11.031.

[2] M. Wankhade, A. C. S. Rao, and C. Kulkarni, "A survey on sentiment analysis methods, applications, and challenges," *Artif Intell Rev*, vol. 55, no. 7, pp. 5731–5780, Oct. 2022, doi: 10.1007/s10462-022-10144-1.

[3] P. Shamsolmoali, M. Zareapoor, L. Shen, A. H. Sadka, and J. Yang, "Imbalanced data learning by minority class augmentation using capsule adversarial networks," *Neurocomputing*, vol. 459, pp. 481–493, Oct. 2021, doi: 10.1016/j.neucom.2020.01.119.

[4] E. Gentili *et al.*, "Machine learning from real data: A mental health registry case study," *Computer Methods and Programs in Biomedicine Update*, vol. 5, p. 100132, 2024, doi: 10.1016/j.cmpbup.2023.100132.

[5] P. Zweifel, "Mental health: The burden of social stigma," *Int J Health Plann Manage*, vol. 36, no. 3, pp. 813–825, May 2021, doi: 10.1002/hpm.3122.

[6] M. Nurul Puji, "Imbalanced Data - Data yang Tidak Seimbang pada Machine Learning." Accessed: Jan. 05, 2025. [Online]. Available: https://base.binus.ac.id/automotive-robotics-engineering/2024/10/07/imbalanced-data-data-yang-tidak-seimbang-pada-machine-learning/

[7] P. Kumar, R. Bhatnagar, K. Gaur, and A. Bhatnagar, "Classification of Imbalanced Data:Review of Methods and Applications," *IOP Conf Ser Mater Sci Eng*, vol. 1099, no. 1, p. 012077, Mar. 2021, doi: 10.1088/1757-899X/1099/1/012077.

[8] Z. A. Sayyed, "Study of sampling methods in sentiment analysis of imbalanced data," Jun. 2021.

[9] T. Wongvorachan, S. He, and O. Bulut, "A Comparison of Undersampling, Oversampling, and SMOTE Methods for Dealing with Imbalanced Classification in Educational Data Mining," *Information*, vol. 14, no. 1, p. 54, Jan. 2023, doi: 10.3390/info14010054.

[10] A. Hanafiah, Y. Arta, H. O. Nasution, and Y. D. Lestari, "Penerapan Metode Recurrent Neural Network dengan Pendekatan Long Short-Term Memory (LSTM) Untuk Prediksi Harga Saham," *Bulletin of Computer Science Research*, vol. 4, no. 1, pp. 27–33, Dec. 2023, doi: 10.47065/bulletincsr.v4i1.321.

[11] Joyeeta Dey and Dhyani Desai, "NLP Based Approach for Classification of Mental Health Issues using LSTM and GloVe Embeddings," *International Journal of Advanced Research in Science, Communication and Technology*, pp. 347–354, Jan. 2022, doi: 10.48175/IJARSCT-2296.

[12] K. Mao *et al.*, "Prediction of Depression Severity Based on the Prosodic and Semantic Features With Bidirectional LSTM and Time Distributed CNN," *IEEE Trans Affect Comput*, vol. 14, no. 3, pp. 2251–2265, Jul. 2023, doi: 10.1109/TAFFC.2022.3154332.

[13] D. Setiawan, K. Stefani, Y. J. Shandy, and C. A. F. Patra, "Sistem Analisa Harga Saham Menggunakan Algoritma Long Short Term Memory (LSTM)," *Media Informatika*, vol. 21, no. 3, pp. 264–279, Mar. 2023, doi: 10.37595/mediainfo.v21i3.159.

[14] Ž. Đ. Vujovic, "Classification Model Evaluation Metrics," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 6, 2021, doi: 10.14569/IJACSA.2021.0120670.

[15] J. Görtler *et al.*, "Neo: Generalizing Confusion Matrix Visualization to Hierarchical and Multi-Output Labels," in *CHI Conference on Human Factors in Computing Systems*, New York, NY, USA: ACM, Apr. 2022, pp. 1–13. doi: 10.1145/3491102.3501823.

[16] F. Thabtah, S. Hammoud, F. Kamalov, and A. Gonsalves, "Data imbalance in classification: Experimental evaluation," *Inf Sci (N Y)*, vol. 513, pp. 429–441, Mar. 2020, doi: 10.1016/j.ins.2019.11.004.
[17] A. Kanukolanu, D. S. P. Kumar, and A. V. S. Abhishek, "Augmentation Techniques Analysis with removal of Class Imbalance using PyTorch for Intel Scene Dataset," 2022.
[18] A. Lopez-del Rio, M. Martin, A. Perera-Lluna, and R. Saidi, "Effect of sequence padding on the performance of deep learning models in archaeal protein functional prediction," *Sci Rep*, vol. 10, no. 1, p. 14634, Sep. 2020, doi: 10.1038/s41598-020-71450-8.
[19] S. Ullah and S.-H. Song, "Design of compensation algorithms for zero padding and its application to a patch based deep neural network," *PeerJ Comput Sci*, vol. 10, p. e2287, Aug. 2024, doi: 10.7717/peerj-cs.2287.
[20] pytorch.org, "Preparing your data for training with Pytorch DataLoaders." Accessed: Dec. 26, 2024. [Online]. Available: https://pytorch.org/tutorials/beginner/basics/data_tutorial.html#preparing-your-data-for-training-with-dataloaders
[21] pytorch.org, "Pytorch Neural Network Layer: Embedding." Accessed: Dec. 26, 2024. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.Embedding.html#torch.nn.Embedding
[22] pytorch.org, "Pytorch Neural Network Layer: Dropout", Accessed: Dec. 26, 2024. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.Dropout.html#torch.nn.Dropout
[23] pytorch.org, "Pytorch Loss Function: CrossEntropyLoss." Accessed: Dec. 27, 2024. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html#torch.nn.CrossEntropyLoss
[24] A. Mao, M. Mohri, and Y. Zhong, "Cross-Entropy Loss Functions: Theoretical Analysis and Applications," in *Proceedings of the 40th International Conference on Machine Learning*, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., in Proceedings of Machine Learning Research, vol. 202. PMLR, Dec. 2023, pp. 23803–23828. [Online]. Available: https://proceedings.mlr.press/v202/mao23b.html
[25] A. Défossez, L. Bottou, F. Bach, and N. Usunier, "A Simple Convergence Proof of Adam and Adagrad," Mar. 2020.

2622-8130
Jurnal Sifo Mikroskil (JSM) - Volume 26, No 2, Oktober 2025
DOI: https://doi.org/10.55601/jsm.v26i2.1799