DECODE: JURNAL PENDIDIKAN TEKNOLOGI INFORMASI
P-ISSN: 2775-2984 | E-ISSN: 2775-1813
Vol. 5, No 2, Juli 2025, Hal: 348-364
Doi: http://dx.doi.org/10.51454/decode.v5i2.1170
https://journal.umkendari.ac.id/index.php/decode

Optimasi Hyperparameter Pada Algoritma K-Nearest Neighbor untuk Analisis Sentimen terhadap Pembelajaran Jarak Jauh

Ety Sutanty[1*], Esti Setiyaningsih[2], Sari Noorlima Yanti[2]
[1]Jurusan Sistem Informasi, Universitas Gunadarma, Indonesia.
[2]Jurusan Teknik Informatika, Universitas Gunadarma, Indonesia.

Artikel Info

Kata Kunci:

Akurasi;
Analisis Sentimen;
Metrik Jarak;
K-Nearest Neighbor;
Pembelajaran Jarak Jauh.

Keywords:

Accuracy;
Distance Metric;
Sentiment Analysis;
K-Nearest Neighbor;
Distance Learning.

Riwayat Artikel:

Submitted: 15 Juni 2025
Accepted: 17 Juli 2025
Published: 23 Juli 2025

Abstrak: Pandemi COVID-19 mendorong diberlakukannya kebijakan Pembelajaran Jarak Jauh (PJJ) secara masif di Indonesia. Kebijakan ini memunculkan beragam tanggapan dari masyarakat, baik positif maupun negatif, yang tersebar luas melalui media sosial. Namun, belum banyak penelitian yang secara sistematis menganalisis opini publik terhadap PJJ menggunakan pendekatan berbasis data. Penelitian ini bertujuan untuk mengidentifikasi dan menganalisis sentimen masyarakat terhadap kebijakan PJJ dengan memanfaatkan data dari Twitter. Penelitian ini menggunakan pendekatan kuantitatif dengan metode text mining dan analisis sentimen berbasis algoritma K-Nearest Neighbor (KNN). Data dikumpulkan melalui Application Programming Interface (API) Twitter dengan kata kunci tertentu, kemudian dilakukan proses prapengolahan yang mencakup pembersihan teks, tokenisasi, dan ekstraksi fitur menggunakan TF-IDF. Proses pelatihan dan pengujian model dilakukan menggunakan KNN, dengan eksplorasi hyperparameter melalui teknik *GridSearchCV*. Parameter yang diuji meliputi jumlah tetangga (k), metode pembobotan (uniform dan distance), serta metrik jarak (Euclidean, Minkowski, dan Cosine). Hasil menunjukkan konfigurasi terbaik pada $k = 22$, pembobotan distance, dan metrik Cosine, dengan akurasi pelatihan tertinggi mencapai 98,80%. Temuan ini menunjukkan bahwa pemilihan hyperparameter yang optimal serta tahapan prapengolahan yang tepat sangat berpengaruh dalam meningkatkan performa klasifikasi sentimen terhadap kebijakan PJJ.

Abstract: *The COVID-19 pandemic prompted the widespread implementation of Distance Learning (PJJ) policies across Indonesia. This policy sparked diverse public responses, both positive and negative, particularly on social media. However, few studies have systematically analyzed public opinion on PJJ using data-driven approaches. This research aims to identify and analyze public sentiment toward the PJJ policy by leveraging data from Twitter. A quantitative approach was employed, involving text mining and sentiment analysis using the K-Nearest Neighbor (KNN) algorithm. Data were collected via the Twitter Application Programming Interface (API) using specific keywords, followed by a comprehensive preprocessing stage including text cleaning, tokenization, and feature extraction using TF-IDF. The KNN model was then trained and tested with hyperparameter tuning performed using the GridSearchCV technique. The parameters evaluated included the number of neighbors (k), weighting methods (uniform and distance), and distance metrics (Euclidean, Minkowski, and Cosine). The best model configuration was achieved with k = 22, distance-based weighting, and the Cosine distance metric, yielding a training accuracy of 98.80%. These findings indicate that optimal hyperparameter selection and effective preprocessing significantly impact the performance of sentiment classification related to the PJJ policy.*

Corresponding Author:
Ety Sutanty
Email: ety_s@staff.gunadarma.ac.id

PENDAHULUAN

Dampak global yang signifikan di berbagai sektor, termasuk pendidikan. Untuk menekan laju penyebaran virus, Pemerintah Indonesia menerapkan kebijakan Pembelajaran Jarak Jauh (PJJ) mulai 16 Maret 2020 (Rukmini et al., 2023). Kebijakan ini mengalihkan seluruh aktivitas belajar mengajar ke sistem daring. Meskipun efektif dalam mengurangi interaksi fisik, PJJ juga menimbulkan berbagai tantangan, terutama bagi siswa(Uyun, 2025). Hasil survei UNICEF terhadap 4.000 siswa di Indonesia mengungkapkan bahwa 38% siswa mengalami kekurangan bimbingan dari guru, sementara 35% menghadapi kendala akses internet yang buruk (UNICEF Indonesia & Agency, National Development Planning, 2021). Di sisi lain, PJJ turut mempercepat pemanfaatan teknologi dan komunikasi digital dalam dunia pendidikan. Respons masyarakat terhadap kebijakan ini beragam, mencerminkan opini positif dan negatif yang tersebar luas melalui media sosial, khususnya Twitter. Twitter merupakan salah satu media sosial yang banyak digunakan untuk menyampaikan pendapat secara real-time, lengkap dengan fitur tagar (hashtag) yang memudahkan pelacakan isu-isu tertentu (Cano-Marin et al., 2023). Analisis opini publik dari media sosial dapat memberikan wawasan berharga melalui pendekatan sentiment analysis yaitu proses analitis yang bertujuan untuk mengidentifikasi dan mengkategorikan opini sebagai positif, negatif, atau netral (Rodríguez-Ibánez et al., 2023). Salah satu algoritma populer yang digunakan dalam klasifikasi sentimen adalah K-Nearest Neighbor (KNN), yaitu metode non-parametrik yang mengklasifikasikan data berdasarkan jarak terdekat ke sejumlah tetangga terdekat(Halder et al., 2024).

Penelitian (Iparraguirre-Villanueva et al., 2022) mengklasifikasikan dan menganalisis konten opini publik tentang Administrasi Dana Pensiun (AFP) yang dipublikasikan di Twitter menggunakan tagar #afp. Peneliti mengumpulkan tweet selama bulan Mei 2022 dan menerapkan berbagai teknik machine learning untuk proses data mining, pembersihan data, tokenisasi, analisis eksploratif, dan klasifikasi sentimen. Analisis dilakukan dalam tiga kategori polaritas: positif (22%), netral (4%), dan negatif (74%). Penelitian juga mencakup analisis frekuensi kata, lemmatisasi, dan visualisasi melalui word cloud. Untuk identifikasi pola, digunakan metode unsupervised learning dengan algoritma K-Means, dan jumlah klaster ditentukan menggunakan metode elbow. Hasil menunjukkan adanya penyebaran opini yang sangat bervariasi, serta jarak antar klaster yang tidak seragam meskipun telah dilakukan standardisasi data, mencerminkan tingginya keragaman sentimen masyarakat terhadap isu AFP. Penelitian (Rizki et al., 2024) meningkatkan akurasi algoritma KNN melalui optimasi hyperparameter *k* menggunakan algoritma PSO. Tidak seperti studi sebelumnya yang hanya menggunakan satu jenis dataset, penelitian ini memvalidasi efektivitas PSO pada tiga dataset berbeda: Iris, Wine, dan Breast Cancer, yang masing-masing memiliki kompleksitas dan jumlah kelas yang bervariasi. Sebelum optimasi, eksperimen dilakukan dengan nilai *k* default (3, 5, 7) untuk mengamati pola awal. Hasil awal menunjukkan bahwa dataset seperti Wine dan Breast Cancer mengalami penurunan akurasi pada *k* kecil karena kompleksitas atribut. Setelah dilakukan optimasi dengan PSO, akurasi meningkat secara signifikan, terutama pada dataset Wine dengan peningkatan sebesar 6,28% menggunakan metrik Manhattan. Temuan ini menunjukkan bahwa PSO efektif dalam mengatasi keterbatasan KNN, dan optimasi hyperparameter *k* dapat memberikan peningkatan kinerja bahkan pada dataset yang awalnya telah menunjukkan performa yang baik seperti Iris.

Penelitian lain juga dilakukan (Trianda et al., 2025) dalam mengembangkan model prediksi risiko hipertensi menggunakan algoritma K-Nearest Neighbor (KNN) yang dioptimasi dengan tiga teknik *hyperparameter tuning*: Grid SearchCV, Bayes SearchCV, dan Random SearchCV. Dataset yang digunakan berasal dari Kaggle, terdiri atas 520 sampel seimbang (260 positif dan 260 negatif) dengan 18 fitur kesehatan seperti usia, tekanan darah, kolesterol, dan glukosa. Setelah proses pra-pemrosesan data, model KNN diuji menggunakan kombinasi nilai *k*, jenis bobot, dan metrik jarak dari masing-

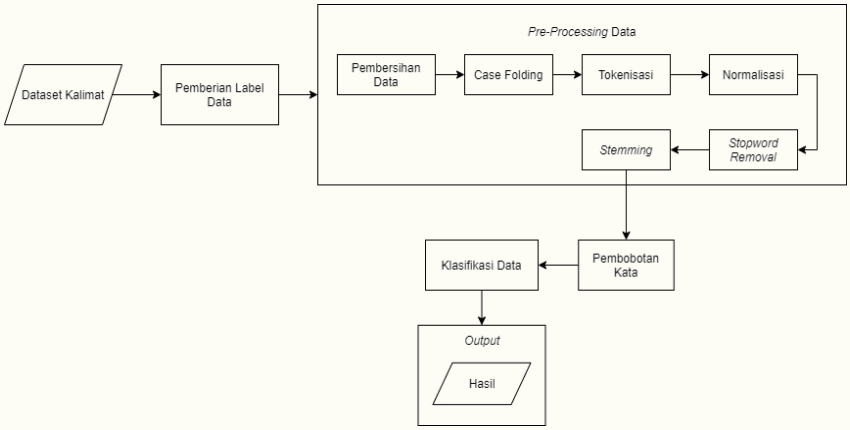
masing metode tuning. Hasil menunjukkan bahwa Bayes SearchCV memberikan kinerja terbaik dengan akurasi sebesar 92% dan skor ROC AUC sebesar 0.96191, mengungguli Grid dan Random SearchCV, serta baseline KNN yang hanya mencapai 85% akurasi. Penelitian ini menyimpulkan bahwa optimasi hyperparameter, khususnya menggunakan pendekatan Bayesian, secara signifikan meningkatkan akurasi prediksi dan efektivitas model KNN dalam klasifikasi risiko hipertensi, serta dapat mendukung pengambilan keputusan klinis lebih tepat di bidang kesehatan. Penelitian lain dalam membahas tantangan ketidakseimbangan data pada performa algoritma klasifikasi juga dilakukan (Achmad & Haris, 2023), khususnya pada data tweet dari Twitter terkait calon Presiden Indonesia 2024. Data yang dikumpulkan dari 8 Oktober 2022 hingga 10 Januari 2023 terdiri dari tweet Anies Baswedan (34.962), Ganjar Pranowo (39.796), dan Prabowo Subianto (12.398), yang kemudian diklasifikasikan menjadi sentimen positif dan negatif menggunakan metode Decision Tree, Naïve Bayes, dan Deep Learning. Proses preprocessing dan ekstraksi data dilakukan dengan RapidMiner, serta penerapan teknik seperti hyperparameter tuning, cross-validation, SMOTE upsampling, dan undersampling untuk menangani ketidakseimbangan kelas. Hasil terbaik diperoleh pada dataset Prabowo Subianto dengan model Deep Learning yang mencapai akurasi 85,42%, precision 63,30%, recall 91,77%, dan AUC 0,867, menunjukkan efektivitas model dalam analisis sentimen pada data yang tidak seimbang.

Kesenjangan utama dari penelitian-penelitian terdahulu terletak pada fokus dan konteks analisis sentimen serta pendekatan optimasi yang digunakan. Penelitian Iparraguirre-Villanueva et al. (2022) dan Achmad & Haris (2023) telah mengkaji sentimen di Twitter dengan berbagai algoritma dan teknik penanganan data tidak seimbang, namun masih terbatas pada topik dan dataset spesifik seperti opini publik terhadap dana pensiun atau kandidat presiden, tanpa eksplorasi mendalam terhadap hyperparameter tuning khusus pada algoritma K-Nearest Neighbor untuk topik pembelajaran jarak jauh. Selain itu, meskipun penelitian Rizki et al. (2024) dan Trianda et al. (2025) telah menunjukkan pentingnya optimasi hyperparameter pada KNN di berbagai domain dan dataset, penerapannya pada data sosial media berbahasa Indonesia, khususnya terkait isu pendidikan jarak jauh, masih minim dan belum eksplorasi secara komprehensif terhadap kombinasi metrik jarak dan pembobotan tetangga.

Penelitian ini berkontribusi dengan mengimplementasikan metode K-Nearest Neighbor yang dioptimasi melalui pengujian variasi hyperparameter nilai k, jenis metrik jarak (Euclidean, Minkowski, Cosine), dan bobot tetangga (uniform dan distance) secara sistematis dalam konteks analisis sentimen pembelajaran jarak jauh pada Twitter berbahasa Indonesia. Pendekatan ini berbeda dari penelitian sebelumnya karena fokus pada topik pembelajaran jarak jauh di masa pandemi, penggunaan data berbahasa Indonesia, serta eksplorasi mendalam terhadap parameter KNN untuk memperoleh performa klasifikasi terbaik. Penelitian ini akan melakukan preprocessing data tweet, pengujian model KNN dengan hyperparameter tuning, dan evaluasi akurasi hasil klasifikasi sentimen positif-negatif, diharapkan dapat memberikan rekomendasi yang valid bagi kebijakan pemerintah dalam pengelolaan pembelajaran jarak jauh.

METODE

Pada penelitian ini dilakukan beberapa tahap untuk membuat sistem analisis sentimen terhadap topik Pembelajaran Jarak Jauh. Metode yang digunakan pada penelitian ini dapat dilihat pada Gambar 1.



Gambar 1. Metode Penelitian Analisis Sentimen PJJ

Tahap awal yang dilakukan pada penelitian ini adalah tahap pengumpulan data pengambilan *tweet* dari media sosial Twitter menggunakan *hashtag* "#pembelajaranjarakjauh" dan kata kunci "pembelajaran jarak jauh". Tahap selanjutnya adalah pemberian label pada data secara manual, yang selanjutnya dilanjutkan dengan tahap *pre-processing*. Tahap *pre-processing* terdiri dari pembersihan data, *case folding*, tokenisasi, normalisasi, *stopword removal* serta *stemming*. Setelah dilakukkan tahap *pre-processing*, selanjutnya dilakukan pembobotan kata menggunakan TF-IDF. Data selanjutnya dibagi menjadi dua jenis, yaitu data *train* yang digunakan pada pelatihan model *K-Nearest Neighbor* dan data *test* yang digunakan untuk mengevaluasi model yang telah dilatih. Pembagian menggunakan rasio 8:2, dimana 80% digunakan sebagai data *train* dan 20% digunakan sebagai data *test*. Kemudian, dilakukan pembentukan model *K-Nearest Neighbor* dengan beberapa *hyperparameter* berbeda. Model yang sudah dirancang kemudian dilatih dengan data *train*. Keluaran dari proses ini akan menghasilkan model *K-Nearest Neighbor* yang sudah dilatih dan dapat digunakan untuk klasifikasi data. Model KNN tersebut akan dievaluasi menggunakan data *test*. Tahap terakhir adalah merancang aplikasi berupa *website* untuk pengimplementasian model KNN yang sudah dievaluasi.

Dataset Kalimat dengan *Hashtag*

Tahap pengumpulan data dilakukan dengan menggunakan *Twitter Archiving Google Sheet* dengan kata kunci Pembelajaran Jarak Jauh dan *hashtag* #pembelajaranjarakjauh. Pengumpulan data merupakan *tweet* yang membahas tentang Pembelajaran Jarak Jauh. Tweet yang berhasil diambil sebanyak 1014 data. Contoh hasil pengambilan data dapat dilihat pada Tabel 1.

Tabel 1. Hasil Pengambilan Data Tweet

Text	Created_at
saya introvert, saya bersedia memperpanjang pembelajaran jarak jauh dan tidak mengeluh 🙏	Mon Mar 01 11:51:46
Pembelajaran jarak jauh yang berkepanjangan akan mengakibatkan terjadinya terjadinya kehilangan pembelajaran pada siswa. #ingatpesanibu #cucitangan #pakaimasker #jagajarak https://t.co/JhlxqXxwDv	Mon Mar 01 10:12:00
Kondisi di pedesaan lebih buruk karena pembelajaran jarak jauh tidak berjalan https://t.co/IxlbuzNLo9	Mon Mar 01 10:52:23

Pemberian Label Data

Tahap pembentukan label pada penelitian ini dilakukan dengan cara manual (Azahra & Setiawan, 2023), yaitu dengan mengklasifikasikan data yang sudah dikumpulkan kedalam kategori positif dan negatif tanpa menggunakan program tertentu. Cara manual dipilih agar proses pelabelan pada setiap data lebih akurat walaupun memakan cukup banyak waktu (Parhusip et al., 2023). Data yang sudah dikumpulkan dan sudah berbentuk *spreadsheet* ditambahkan kolom untuk label yang

nantinya akan diisi oleh label positif atau negatif sesuai dengan muatan dari setiap data. Masing-masing label yang digunakan pada penelitian ini akan dijelaskan sebagai berikut:

1. Label positif: berupa data teks yang mengandung dukungan ataupun pembelaan terhadap kebijakan pembelajaran jarak jauh.
2. Label negatif: berupa data teks yang mengandung opini buruk, ejekan, sindiran, maupun hinaan terhadap kebijakan pembelajaran jarak jauh.

Hasil dari tahap ini menghasilkan 576 data positif dan 438 data negatif dari total keseluruhan 1014 data. Contoh data teks yang sudah diberi label dapat dilihat pada Tabel 2.

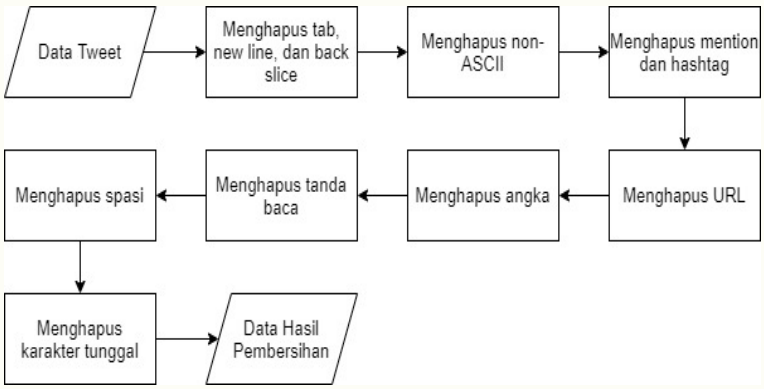
Tabel 2. Contoh Data yang Sudah Diberi Label

Text *Tweet*	Label
saya introvert, saya bersedia memperpanjang pembelajaran jarak jauh dan tidak mengeluh 🙏	Positif
Pembelajaran jarak jauh yang berkepanjangan akan mengakibatkan terjadinya terjadinya kehilangan pembelajaran pada siswa. #ingatpesanibu #cucitangan #pakaimasker #jagajarak https://t.co/JhlxqXxwDv	Negatif
Kondisi di pedesaan lebih buruk karena pembelajaran jarak jauh tidak berjalan https://t.co/IxlbuzNLo9	Negatif

Pre-Processing Data

Pre-processing dilakukan dengan tujuan mentransformasikan data mentah ke dalam format yang dibutuhkan untuk proses analisis sentimen. Tahap *pre-processing* ini terdiri dari beberapa tahap, yaitu pembersihan data, *case folding*, tokenisasi, normalisasi, *stopword removal*, dan *stemming* (Rifaldi et al., 2023). Beberapa proses tersebut akan dijelaskan sebagai berikut:

1. **Pembersihan Data.** Data *tweets* yang didapat dari Twitter seringkali mengandung komponen yang tidak diperlukan, seperti *delimiter*, simbol, *emoticon*, atau URL, sehingga data tersebut perlu dilakukan pembersihan terlebih dahulu (Dewi & Arianto, 2023). Langkah-langkah yang ada pada tahap pembersihan data dapat dilihat pada Gambar 2.



Gambar 2. Tahap Pembersihan Data

Pada Python, pembersihan data dapat dilakukan menggunakan *library* re (Oladipupo et al., 2023). Berikut ini penggalan sintaks dari pembersihan data yang dilakukan terhadap teks:

```
def cleaning(text):
    text = text.replace('\\t'," ").replace('\\n'," ").replace('\\u'," ").replace('\\',"")
    text = text.encode('ascii', 'replace').decode('ascii')
    text = ''.join(re.sub("([@#][A-Za-z0-9]+)|(\w+:\/\/\S+)"," ", text).split())
```

```
        text=text.replace("http://"," ").replace("https://", " ")
    text = re.sub(r"\d+", "", text)
        text = text.translate(str.maketrans("","",string.punctuation))
    text = text.strip()
    text = re.sub('\s+',' ',text)
    return re.sub(r"\b[a-zA-Z]\b", "", text)
```

Fungsi *def cleaning(text)* digunakan untuk membersihkan data teks pada tweet dengan mengganti delimiter dan karakter non-ASCII menjadi ASCII, menghapus mention dan hashtag, serta menghilangkan tanda baca dan spasi di awal maupun akhir teks. Fungsi ini juga mengganti karakter tunggal dengan spasi untuk memastikan teks lebih rapi dan siap digunakan dalam proses analisis selanjutnya.

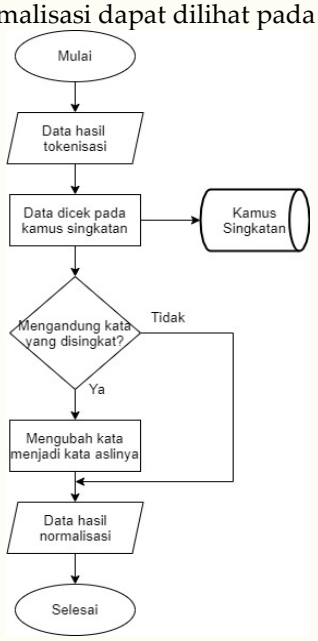
2. **Case Folding.** Tahap *case folding* merupakan tahapan untuk mengubah huruf kapital pada teks menjadi huruf kecil. Berikut ini penggalan sintaks dari tahap *case folding* untuk mengubah huruf kapital yang terdapat pada teks. Perintah *def casefolding(text)* adalah pendeklarasian fungsi untuk melakukan proses *case folding*. Perintah *return text.lower()* akan mengembalikan teks yang diubah menjadi huruf kecil.

```
def casefolding(text):
    return text.lower()
```

3. **Tokenisasi.** Tokenisasi adalah proses pemisahan setiap kata yang menyusun suatu teks. Teks yang sebelumnya sudah melalui tahap *case folding* akan dibagi atau dipotong berdasarkan kata menjadi potongan-potongan yang dapat disebut dengan token(Choo & Kim, 2023). Pada Python, tahap tokenisasi dapat dilakukan dengan fungsi *word_tokenize* pada *library* nltk. *Natural Language Toolkit* (NTLK) adalah *library* yang biasa digunakan untuk mempermudah kegiatan pemrosesan teks (Olabiyi et al., 2024). Berikut ini penggalan sintaks dari pembuatan fungsi tokenisasi untuk mengubah kata menjadi potongan token. Perintah *def tokenizing(text)* akan melakukan proses tokenisasi, dan *return word_tokenize(text)* merupakan fungsi untuk mengembalikan teks hasil tokenisasi.

```
def tokenizing(text):
    return word_tokenize(text)
```

4. **Normalisasi Data.** Normalisasi data digunakan untuk mengubah teks yang mengandung kata yang disingkat menjadi kata aslinya agar kata tersebut dapat dimengerti oleh model (Siino et al., 2024). Langkah-langkah pada tahap normalisasi dapat dilihat pada Gambar 3.



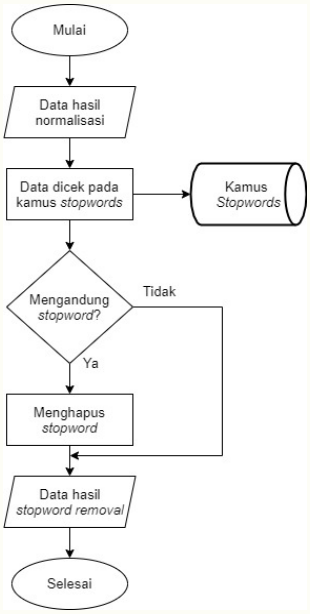
Gambar 3. Tahap Normalisasi Data

Berikut ini penggalan sintaks dari fungsi normalisasi untuk mengubah kata yang disingkat menjadi kata asli:

```
key_norm = pd.read_csv('data/kamus_singkatan.csv')
def normalizing(text):
    text = ' '.join([key_norm[key_norm['singkatan'] == word]['asli'].values[0] if
    (key_norm["singkatan"] ==
        word).any() else word for word in text])
return text.split(' ')
```

Fungsi *normalizing(text)* digunakan untuk melakukan normalisasi teks dengan mengganti kata-kata singkatan menjadi bentuk aslinya berdasarkan data dari file *kamus_singkatan* yang dibaca menggunakan *pd.read_csv*. Proses ini dilakukan dengan mencocokkan setiap kata dalam teks dengan entri dalam kamus dan mengganti jika ditemukan padanan, lalu hasilnya dikembalikan dalam bentuk daftar kata yang dipisahkan oleh spasi(Gunawan et al., 2019).

5. ***Stopwords Removal.*** Tahap *stopwords removal* ini digunakan untuk menghilangkan kata-kata yang tidak dibutuhkan dalam analisis sentimen, misalnya seperti kata penghubung. Pada tahap ini, setiap kata pada data *tweets* akan diperiksa(Pradana & Hayaty, 2019). Jika pada data terdapat kata sambung, kata depan, kata ganti atau kata yang tidak ada hubungannya dalam analisis sentimen, maka kata tersebut akan dihilangkan. Pada Python, tahap *stopwords removal* dapat dilakukan dengan menggunakan fungsi *stopwords* pada *library* nltk. Langkah-langkah pada tahap *Stopwords* ini dapat dilihat pada Gambar 4.



Gambar 4. Tahap *Stopwords Removal*

Berikut ini penggalan sintaks dari fungsi *Stopwords Removal* untuk menghilangkan kata yang tidak diperlukan:

```
stopwords = pd.read_csv('dataset/stopwordsID.csv', header=None)
extend = pd.DataFrame(["yg", "dg", "rt", "dgn", "ny", "d", 'klo', 'kalo', 'amp', 'biar', 'bikin',
'bilang', 'gak', 'ga', 'krn', 'nya', 'nih', 'sih', 'si', 'tau', 'tdk', 'tuh', 'utk', 'ya', 'jd', 'jgn', 'sdh',
'aja', 'n', 't', 'nyg', 'hehe', 'pen', 'u', 'nan', 'loh',
'rt','&amp','yah','my','rb','jr','rp','hr','di','kb','gb'])
stopwords = stopwords.append(extend, ignore_index=True)
list_stopwords = set(stopwords.iloc[:,0])
def stopwords_removing(words)
```

```
return [word for word in words if word not in list_stopwords]
```

Fungsi *stopwords_removing(words)* digunakan untuk menghapus kata-kata yang termasuk dalam daftar stopword Bahasa Indonesia. Kamus stopword dibaca dari file *stopwordsID* menggunakan pd.read_csv, kemudian dapat diperluas dengan kata tambahan yang dimasukkan melalui DataFrame extend. Setelah itu, semua stopword disimpan dalam list_stopwords, dan fungsi akan mengembalikan daftar kata yang tidak termasuk dalam daftar tersebut, sehingga hanya menyisakan kata-kata penting untuk analisis selanjutnya.

6. ***Stemming.*** Stemming adalah proses mengubah kata menjadi bentuk kata dasarnya dengan cara menghilangkan imbuhan pada kata dalam suatu teks. Pada tahapan ini, term-term hasil filtering kemudian diubah menjadi ke bentuk dasar term atau kata tersebut. Bentuk dasar yang dimaksud adalah bentuk dasar dengan menghilangkan imbuhan, mulai dari imbuhan awal sampai imbuhan akhir. Pada Python, proses *stemming* dilakukan menggunakan *library* sastrawi yang dilakukan dengan algoritma Nazief dan Adriani. Proses *stemming* dari algoritma Nazief dan Ardani adalah sebagai berikut :
   a. Kata yang belum dilakukan proses *stemming* akan dibandingkan dengan kamus kata dasar. Jika kata tersebut ditemukan pada kamus kata dasar, maka kata tersebut akan dianggap sebagai kata dasar dan proses akan berhenti. Jika kata tersebut tidak ditemukan pada kamus, maka lanjut ke langkah 2.
   b. Jika kata tersebut memiliki partikel (“-lah”, “-kah”, ”-ku”, “-mu”, atau “-nya) maka bagian tersebut dibuang. Langkah ini akan diulangi untuk menghapus kata ganti kepunyaan (“-ku”, “-mu”, atau “-nya”).
   c. Jika terdapat akhiran (“-i”, “-kan”, dan “-an”), maka bagian tersebut dibuang. Jika kata ditemukan pada kamus, maka proses akan berhenti. Jika tidak ditemukan, maka lanjut ke langkah 4.
   d. Jika terdapat awalan (“be-“, “di-“, “ke-“, “me-“, “pe-“, “se-“, dan “te-“), maka bagian tersebut dibuang dengan mengikuti aturan berikut:
      1) Algoritma akan berhenti jika: Awalan diidentifikasi bentuk sepasang imbuhan yang tidak diperbolehkan dengan akhiran (“be-i”, “di-an”, “ke-i-kan”, “me-an”, “se-i-kan”, “te-an”) yang dihapus, Diidentifikasi awalan yang sekarang identik dengan awalan yang telah dihapus sebelumnya atau, Kata tersebut sudah tidak memiliki awalan.
      2) Identifikasi jenis awalan dan peluruhannya bila diperlukan jenis awalan ditentukan dengan aturan : Jika awalan dari kata adalah “di-“, “ke-“, atau “se-“ maka awalan dapat langsung dihilangkan dan hapus awalan “te-“, “be-“, “me-“, atau “pe-“.
   e. Jika semua langkah di atas gagal, maka kata dianggap sebagai kata dasar dan proses akan berhenti.

   Berikut ini penggalan sintaks dari fungsi *stemming* yang mengubah kata menjadi bentuk dasar:

```
def stemming(words):
    stem_kalimat = []
for k in words:
    stem_kata = stemmer.stem(k)
    stem_kalimat.append(stem_kata)
stem_kalimat_str = ' '.join(stem_kalimat)
return stem_kalimat_str
```

Pembobotan Kata

Tahap pembobotan kata adalah tahap yang berguna untuk menghitung bobot dari setiap kata yang umum digunakan. Pada penelitian ini, pembobotan kata dilakukan menggunakan *Term Frequency Inverse Document Frequency* (TF-IDF). TF-IDF menggabungkan dua skema, yaitu *Term Frequency* (TF) dan *Inverse Document Frequency* (IDF). Pada *Term Frequency* (TF), potongan kata atau token yang akan diberi nilai sesuai dengan jumlah kemunculannya dalam satu dokumen dan akan didapatkan matriks *term frequency*, sedangkan pada proses *Invers Document Frequency* (IDF), potongan kata diberi bobot atau

nilai berdasarkan frekuensi kemunculannya pada seluruh dokumen. Frekuensi kemunculan setiap kata atau *term* pada setiap kalimat akan dihitung dengan menggunakan *Term Frequency* menggunakan Persamaan :

$$tf_{i,j} = \frac{n_{i,j}}{\Sigma_k n_{kj}} \quad (1)$$

Variabel $n_{i,j}$ adalah jumlah term- i yang muncul pada teks ke- j. $\Sigma_k n_{kj}$ adalah total kata pada teks ke- j. Setelah melakukan proses *Term Frequency* (TF), langkah selanjutnya adalah menghitung *Inverse Document Frequency* (IDF) untuk menghitung frekuensi kemunculan *term* pada seluruh teks menggunakan Persamaan :

$$idf_i = log\left(\frac{1+N}{1+df_i}\right) + 1 \quad (2)$$

Variabel N adalah jumlah seluruh teks, df_i adalah jumlah teks yang mengandung term- i. Setelah melakukan perhitungan dengan menggunakan *Inverse Document Frequency* (IDF), selanjutnya adalah mengkalikan nilai TF dengan IDF untuk mendapatkan *Term Frequency Inverse Document Frequency* (TF-IDF). Proses TF-IDF menggunakan Persamaan :

$$w_{i,j} = tf_{i,j}\ x\ idf_i \quad (3)$$

Variael $tf_{i,j}$ adalah hasil perhitungan kemunculan suatu *term* pada sebuah teks (TF) dan idf_i adalah hasil perhitungan kemunculan suatu *term* pada seluruh teks (IDF). Pada Python, pembobotan kata menggunakan TF-IDF dapat dilakukan menggunakan fungsi *TfidfVectorizer* yang terdapat pada *library* sklearn. Berikut ini penggalan sintaks fungsi pembobotan kata dengan menggunakan TF-IDF:

```
vectorizer = TfidfVectorizer()
tfidf_feat = vectorizer.fit_transform(np.array((list(data['text_stemming']))))
data_tfidf = pd.DataFrame(tfidf_feat.toarray(), columns =
vectorizer.get_feature_names())
```

Pembuatan Model *K-Nearest Neighbor*

Tahap selanjutnya yang dilakukan adalah pembentukan model dengan metode *K-Nearest Neighbor*. Sebelum melakukan tahap ini, hal pertama yang perlu dilakukan adalah melakukan *hyperparameter tuning*. *Hyperparameter tuning* dilakukan untuk menentukan *hyperparameter* terbaik yang sesuai dengan karakteristik data latih sehingga dapat menghasilkan model dengan kemampuan generalisasi yang baik. Pada penelitian ini, tahapan *hyperparameter tuning* dilakukan menggunakan *GridSearchCV*. *GridSearchCV* adalah metode untuk mencari kombinasi nilai *hyperparameter* terbaik dalam sebuah *grid*. *GridSearchCV* akan membagi data sesuai dengan jumlah grid yang ditentukan dan melatih model berdasarkan pembagian data tersebut. Kemudian *GridSearchCV* akan menghitung rata-rata akurasi pelatihan dan memilih parameter dengan akurasi terbaik.

Pada penelitian ini, jumlah grid yang digunakan dalam pelatihan model adalah 3 *grid*. Hyperparameter yang digunakan dalam penelitian ini adalah parameter nilai k, *weights*, dan metrik jarak. Nilai akurasi terbaik yang didapat pada *hyperparameter tuning* akan digunakan dalam pelatihan dan pengujian model. Berikut ini penggalan sintaks untuk menentukan nilai *hyperparameter* dan pembentukan model:

```
knn = KNeighborsClassifier()
k_range = list(range(1,31))
weights = ['uniform', 'distance']
metric = ['minkowski','euclidean','cosine']
param_grid = { 'n_neighbors' : k_range,
        'metric' : metric,
        'weights' : weights}
```

```
model = GridSearchCV(knn, param_grid, verbose = 1, cv=3, n_jobs = -
1,scoring='accuracy')
model.fit(X_train, y_train)
print(model.best_score_, model.score(X_test, y_test))
print(model.best_params_)
```

HASIL DAN PEMBAHASAN

Beberapa hasil yang akan dibahas adalah hasil dari tahap pengumpulan data, hasil *preprocessing*, hasil pembobotan kata, hasil pelatihan model, dan hasil evaluasi model, dan implementasi model *K-Nearest Neighbor* pada sistem yang dibuat.

Hasil Tahap *Preprocessing*

Pada hasil tahap *preprocessing* ini akan dibahas hasil yang didapat dari tahap *preprocessing* yang telah dilakukan. Pada tahap *preprocessing* ini terdapat beberapa proses, proses yang akan dibahas antara lain adalah pembersihan data, *case folding*, tokenisasi, normalisasi, *stopword removal*, dan *stemming*. Berikut ini adalah hasil dari proses yang sudah disebutkan.

Hasil Pembersihan Data

Pembersihan data dilakukan terhadap data teks yang sudah melalui tahap labelisasi untuk menghilangkan komponen yang tidak diperlukan dalam analisis sentimen. Berikut ini adalah perbandingan antara data yang belum dibersihkan dengan data yang sudah dibersihkan dapat dilihat pada Tabel 3.

Tabel 3. Perbandingan Data Sebelum dan Sesudah Pembersihan Data

Sebelum Pembersihan Data	Setelah Pembersihan Data
saya introvert, saya bersedia memperpanjang pembelajaran jarak jauh dan tidak mengeluh🙏	saya introvert saya bersedia memperpanjang pembelajaran jarak jauh dan tidak mengeluh
Pembelajaran jarak jauh yang berkepanjangan akan mengakibatkan terjadinya terjadinya kehilangan pembelajaran pada siswa. #ingatpesanibu #cucitangan #pakaimasker #jagajarak https://t.co/JhlxqXxwDv	Pembelajaran jarak jauh yang berkepanjangan akan mengakibatkan terjadinya kehilangan pembelajaran pada siswa
Kondisi di pedesaan lebih buruk karena pembelajaran jarak jauh tidak berjalan https://t.co/IxlbuzNLo9	Kondisi di pedesaan lebih buruk krn pembelajaran jarak jauh tidak berjalan

Berdasarkan perbandingan pada Tabel 3 dapat dilihat bahwa hasil dari pembersihan data ini dapat menghilangkan *emoticon* pada teks pertama, *hashtag* dan URL pada teks kedua, dan URL pada teks ketiga.

Hasil *Case Folding*

Case folding dilakukan terhadap data teks yang sebelumnya telah melalui proses pembersihan data. Berikut ini perbandingan antara data yang belum melalui tahap *case folding* dengan data yang sudah melalui tahap *case folding* yang dapat dilihat pada Tabel 4.

Tabel 4. Perbandingan Data Sebelum dan Sesudah Case Folding

Sebelum *Case Folding*	Setelah *Case Folding*
saya introvert saya bersedia memperpanjang pembelajaran jarak jauh dan tidak mengeluh	saya introvert saya bersedia memperpanjang pembelajaran jarak jauh dan tidak mengeluh

Pembelajaran jarak jauh yang berkepanjangan akan mengakibatkan terjadinya kehilangan pembelajaran pada siswa	pembelajaran jarak jauh yang berkepanjangan akan mengakibatkan terjadinya kehilangan pembelajaran pada siswa
Kondisi di pedesaan lebih buruk krn pembelajaran jarak jauh tidak berjalan	kondisi di pedesaan lebih buruk krn pembelajaran jarak jauh tidak berjalan

Berdasarkan perbandingan pada Tabel 4 dapat dilihat bahwa kata-kata seperti "Pembelajaran" pada teks kedua serta kata "Kondisi" pada teks ketiga diubah menjadi huruf kecil dalam proses *case folding*.

Hasil Tokenisasi

Tokenisasi dilakukan terhadap data teks yang sebelumnya sudah melalui proses *case folding*. Hasil dari proses tokenisasi ini akan mengubah data teks menjadi potongan-potongan kata yang disebut dengan token. Berikut ini perbandingan antara data yang belum ditokenisasi dengan data yang sudah ditokenisasi yang dapat dilihat pada Tabel 5.

Tabel 5. Perbandingan Data Sebelum dan Sesudah Tokenisasi

Sebelum Tokenisasi	Sesudah Tokenisasi
saya introvert saya bersedia memperpanjang pembelajaran jarak jauh dan tidak mengeluh	['saya', 'introvert', 'saya', 'bersedia', 'memperpanjang', 'pembelajaran', 'jarak', 'jauh', 'dan', 'tidak', 'mengeluh']
pembelajaran jarak jauh yang berkepanjangan akan mengakibatkan terjadinya kehilangan pembelajaran pada siswa	['pembelajaran', 'jarak', 'jauh', 'yang', 'berkepanjangan', 'akan', 'mengakibatkan', 'terjadinya', 'kehilangan', 'pembelajaran', 'pada', 'siswa']
kondisi di pedesaan lebih buruk krn pembelajaran jarak jauh tidak berjalan	['kondisi', 'di', 'pedesaan', 'lebih', 'buruk', 'krn', 'pembelajaran', 'jarak', 'jauh', 'tidak', 'berjalan']

Hasil Normalisasi Data

Normalisasi data dilakukan terhadap data yang sebelumnya sudah melalui proses tokenisasi. Hasil dari proses normalisasi akan mengubah teks yang mengandung kata yang disingkat menjadi kata aslinya. Berikut ini perbandingan antara data yang belum dinormalisasi dengan data yang sudah dinormalisasi yang dapat dilihat pada Tabel 6.

Tabel 6. Perbandingan Data Sebelum dan Sesudah Normalisasi

Sebelum Normalisasi	Sesudah Normalisasi
['saya', 'introvert', 'saya', 'bersedia', 'memperpanjang', 'pembelajaran', 'jarak', 'jauh', 'dan', 'tidak', 'mengeluh']	['saya', 'introvert', 'saya', 'bersedia', 'memperpanjang', 'pembelajaran', 'jarak', 'jauh', 'dan', 'tidak', 'mengeluh']
['pembelajaran', 'jarak', 'jauh', 'yang', 'berkepanjangan', 'akan', 'mengakibatkan', 'terjadinya', 'kehilangan', 'pembelajaran', 'pada', 'siswa']	['pembelajaran', 'jarak', 'jauh', 'yang', 'berkepanjangan', 'akan', 'mengakibatkan', 'terjadinya', 'kehilangan', 'pembelajaran', 'pada', 'siswa']

['kondisi', 'di', 'pedesaan', 'lebih', 'buruk', 'krn', 'pembelajaran', 'jarak', 'jauh', 'tidak', 'berjalan']	['kondisi', 'di', 'pedesaan', 'lebih', 'buruk', 'karena', 'pembelajaran', 'jarak', 'jauh', 'tidak', 'berjalan']

Berdasarkan perbandingan pada Tabel 6 dapat dilihat bahwa setelah melalui proses normalisasi teks ketiga yang memiliki kata yang disingkat, yaitu "krn" berubah menjadi kata aslinya yaitu "karena".

Hasil *Stopwords Removal*

Stopwords removal ini dilakukan terhadap data yang telah melalui proses normalisasi. Berikut ini adalah perbandingan antara data yang sebelum melalui *Stopwords Removal* dengan data yang sudah melalui *Stopwords Removal* dapat dilihat pada Tabel 7.

Tabel 7. Perbandingan Data Sebelum dan Sesudah Stopwords Removal

Sebelum *Stopwords Removal*	**Sesudah *Stopwords Removal***
['saya', 'introvert', 'saya', 'bersedia', 'memperpanjang', 'pembelajaran', 'jarak', 'jauh', 'dan', 'tidak', 'mengeluh']	['introvert', 'bersedia', 'memperpanjang', 'pembelajaran', 'jarak', 'mengeluh']
['pembelajaran', 'jarak', 'jauh', 'yang', 'berkepanjangan', 'akan', 'mengakibatkan', 'terjadinya', 'kehilangan', 'pembelajaran', 'pada', 'siswa']	['pembelajaran', 'jarak', 'berkepanjangan', 'mengakibatkan', 'kehilangan', 'pembelajaran', 'siswa']
['kondisi', 'di', 'pedesaan', 'lebih', 'buruk', 'karena', 'pembelajaran', 'jarak', 'jauh', 'tidak', 'berjalan']	['kondisi', 'pedesaan', 'buruk', 'pembelajaran', 'jarak', 'berjalan']

Berdasarkan perbandingan pada Tabel 7 dapat dilihat bahwa setelah melalui proses *stopword removal* pada teks pertama sejumlah kata akan terhapus seperti "saya", "jauh", "dan", dan "tidak". Pada teks kedua beberapa kata yang dihapus adalah "jauh", "akan", "terjadinya", dan "pada", sedangkan pada teks ketiga beberapa kata yang dihapus adalah "di", "lebih", "karena", "jauh", dan "tidak".

Hasil *Stemming*

Proses terakhir pada *preprocessing* adalah *stemming*. *Stemming* dilakukan terhadap data yang sebelumnya sudah melalui proses *stopword removal*. Berikut ini perbandingan antara data yang belum melalui *stemming* dengan data yang sudah melalui *stemming* yang dapat dilihat pada Tabel 8.

Tabel 8. Perbandingan Data Sebelum dan Sesudah Stemming

Sebelum *Stemming*	**Sesudah *Stemming***
['introvert', 'bersedia', 'memperpanjang', 'pembelajaran', 'jarak', 'mengeluh']	introvert sedia panjang ajar jarak keluh
['pembelajaran', 'jarak', 'berkepanjangan', 'mengakibatkan', 'kehilangan', 'pembelajaran', 'siswa']	ajar jarak panjang akibat hilang ajar siswa
['kondisi', 'pedesaan', 'buruk', 'pembelajaran', 'jarak', 'berjalan']	kondisi desa buruk ajar jarak jalan

Berdasarkan perbandingan pada Tabel 8 dapat dilihat bahwa setelah melalui proses *stemming* maka beberapa kata yang dihapus imbuhannya pada teks pertama adalah kata "bersedia", "memperpanjang", "pembelajaran". {ada teks kedua kata yang dihapus imbuhannya yaitu

"pembelajaran", "berkepanjangan", "mengakibatkan", "kehilangan", dan "pembelajaran", sedangkan pada teks ketiga terdapat kata "pedesaan", "pembelajaran" dan "berjalan".

Pembahasan

Text preprocessing melalui pembersihan, *case folding*, tokenisasi, normalisasi, *stopword removal*, dan *stemming* menghasilkan data bersih, konsisten, dan siap fitur. Sebagaimana dijelaskan oleh Cano-Marin et al. (2023), urutan dan keberlanjutan langkah preprocess terbukti sangat mempengaruhi performa model. Proses ini berhasil menghilangkan noise (emotikon, URL, hashtag) dan meminimalisir dimensi teks, sesuai temuan Gunawan et al. (2019) dan penelitian lain yang menunjukkan bahwa *preprocessing* mengurangi redundansi dan meningkatkan akurasi hingga signifikan.

Hasil Pembobotan Kata

Hasil dari pembobotan kata dari contoh teks yang dilakukan *pre-processing* dapat dilihat pada Tabel 9 Pada hasil pemobotan kata terdapat skor pada beberapa kata sesuai dengan perhitungan yang sudah dilakukan pada tahap sebelumnya.

Tabel 9. Hasil Term Frequency Inverse Document Frequency

Term	TF-IDF		
	Teks 1	Teks 2	Teks 3
Introvert	0,217272	0	0
Sedia	0,217272	0	0
Panjang	0,187865	0,160866	0
Ajar	0,167	0,143	0,167
Jarak	0,167	0,143	0,167
Keluh	0,217272	0	0
Akibat	0	0,160866	0
Hilang	0	0,160866	0
Siswa	0	0,160866	0
Kondisi	0	0	0,217272
Desa	0	0	0,217272
Buruk	0	0	0,217272
Jalan	0	0	0,217272

Pembahasan

Pemakaian TF-IDF mengubah teks menjadi representasi numerik yang informatif. Metode ini umum dan efektif dalam teks mining karena mengurangi dominasi kata umum dan menyoroti kata kunci—sejalan dengan praktik umum dalam text mining.

Hasil Pembuatan Model dengan Parameter Nilai *K*

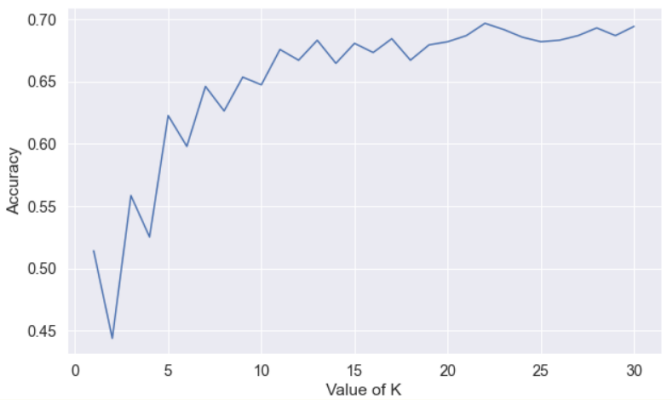
Nilai *k* pada metode K-Nearest Neighbor menunjukkan jumlah tetangga terdekat yang diperlukan dalam menentukan klasifikasi suatu kalimat. Parameter nilai *k* yang digunakan pada tahap ini terdiri dari 30 nilai dari 1 hingga 30. Parameter lain diberi nilai *default* yang digunakan pada *library* sklearn, yaitu *Minkowski* untuk parameter metrik jarak dan *uniform* untuk pembobotan. Hasil pelatihan model dengan parameter nilai *k* dapat dilihat pada Tabel 10.

Tabel 10. Hasil Training dengan Parameter Nilai K

Nilai *k*	Rerata Akurasi *Training* (%)	Nilai *k*	Rerata Akurasi *Training* (%)	Nilai *k*	Rerata Akurasi *Training* (%)
1	51,42%	11	67,57%	21	68,68%
2	44,39%	12	66,70%	22	69,67%
3	55,84%	13	68,31%	23	69,18%

Nilai *k*	Rerata Akurasi *Training* (%)	Nilai *k*	Rerata Akurasi *Training* (%)	Nilai *k*	Rerata Akurasi *Training* (%)
4	52,52%	14	66,46%	24	68,56%
5	62,26%	15	68,06%	25	68,19%
6	59,79%	16	67,32%	26	68,31%
7	64,61%	17	68,43%	27	68,68%
8	62,64%	18	66,71%	28	69,30%
9	65,35%	19	67,94%	29	68,68%
10	64,73%	20	68,19%	30	69,42%

Berdasarkan Tabel 10 pada *record* yang diberi kotak biru dapat dilihat bahwa akurasi *training* tertinggi dicapai ketika mengunakan nilai *k* = 22. Visualisasi hasil *training* menggunakan parameter nilai *k* ditampilkan dalam bentuk grafik dapat dilihat pada Gambar 5.



Gambar 5. Grafik *Training* dengan Parameter Nilai *K*

Gambar 5 menunjukkan grafik hasil akurasi berdasarkan nilai *k* yang digambarkan dengan garis berwarna biru. Garis biru yang bergerak naik menunjukkan bahwa semakin besar nilai *k*, maka semakin besar juga nilai akurasinya. Namun, pada saat nilai *k* sudah mencapai 22, nilai akurasi mulai menunjukkan penurunan, sehingga dapat disimpulkan bahwa 22 merupakan nilai *k* yang paling optimal untuk melakukan *training* data.

Hasil Pembuatan Model dengan Parameter *Weights*

Parameter *weights* pada metode K-Nearest Neighbor merupakan metode pembobotan tetangga dari setiap kalimat yang akan diklasifikasi. Parameter *weights* yang digunakan pada tahap ini adalah *uniform weights* dimana bobot yang diberikan untuk kalimat tetangga bernilai sama dan *distance weights* dimana bobot diberikan berdasarkan jarak dari suatu kalimat ke kalimat tetangganya. Parameter nilai k diberi nilai 22 karena memiliki akurasi tertinggi dari tahap sebelumnya dan parameter metrik jarak diberi nilai *default* yang digunakan pada *library* sklearn, yaitu *Minkowski*. Hasil pelatihan model dengan parameter *weights* dapat dilihat pada Tabel 11.

Tabel 11. Hasil Training dengan Parameter Weights

Weight	Rerata Akurasi *Training* (%)
uniform	89,67%
distance	90,66%

Berdasarkan Tabel 11, terlihat bahwa penggunaan *distance weights* pada algoritma KNN menghasilkan akurasi training tertinggi dibandingkan dengan metode pembobotan lainnya. Hal ini menunjukkan bahwa memberikan bobot lebih besar pada tetangga terdeka, yakni yang memiliki jarak

lebih kecil terhadap data yang dianalisis,sehingga meningkatkan performa model dalam mengenali pola sentimen pada data latih. Strategi ini memungkinkan model lebih responsif terhadap kedekatan data dalam ruang fitur, sehingga prediksi sentimen menjadi lebih tepat. Hasil ini mengindikasikan bahwa faktor jarak memiliki pengaruh signifikan terhadap kinerja KNN dalam konteks analisis sentimen.

Hasil Pembuatan Model dengan Parameter Metrik Jarak

Metrik jarak pada *K-Nearest Neighbor* merupakan metode yang digunakan untuk menghitung jarak antara kalimat yang akan diklasifikasi dengan tetangganya. Metrik jarak yang digunakan pada tahap ini adalah *Minkowski Distance*, *Euclidian Distance* dan *Cosine Distance*. Parameter nilai *k* menggunakan nilai 22 dan *weights* menggunakan *distance weights* karena memiliki akurasi tertinggi dari tes sebelumnya. Hasil pelatihan model dengan parameter nilai *k* dapat dilihat pada Tabel 12.

Tabel 12. Hasil Training dengan Parameter Metrik Jarak

Metrik Jarak	Rerata Akurasi *Training* (%)
Minkowski	90,66%
Euclidian	91,66%
Cosine	98,80%

Berdasarkan Tabel 12, terlihat bahwa pemilihan metrik jarak memberikan dampak signifikan terhadap akurasi training dalam model KNN untuk analisis sentimen. Metrik Cosine menghasilkan rerata akurasi training tertinggi sebesar 98,80%, jauh melampaui metrik Euclidian (91,66%) dan Minkowski (90,66%). Ini menunjukkan bahwa dalam konteks data yang dianalisis, metrik Cosine lebih efektif dalam mengukur kemiripan antar data, kemungkinan karena kemampuannya menangkap arah vektor dalam ruang fitur yang relevan untuk representasi teks atau sentimen. Oleh karena itu, penggunaan metrik Cosine terbukti lebih optimal dalam meningkatkan performa model pada tahap pelatihan.

Pembahasan

Eksperimen KNN dengan variasi nilai *k* (1–30) menunjukkan puncak akurasi pada *k* = 22 (~69,7% untuk *training*). Hal ini konsisten dengan teori bahwa nilai *k* sedang dapat menyeimbangkan bias dan varians, menghindari *overfitting* atau *underfitting*. Pengujian *weights* menunjukkan bahwa distance weighting (90,66%) unggul dibanding *uniform* (89,67%), menegaskan bahwa bobot berdasarkan kedekatan meningkatkan sensitivitas model terhadap fitur relevan. Temuan ini didukung oleh teori bahwa distance weighting memberikan bobot lebih pada tetangga terdekat sebagai strategi efektif untuk *sentiment classification*. Cosine memberikan akurasi signifikan tertinggi (98,80%), lebih unggul terhadap Euclidean (91,66%) dan Minkowski (90,66%). Konfirmasi riset sebelumnya menunjukkan Cosine lebih presisi untuk data berbasis TF-IDF karena mempertimbangkan orientasi vektor dan bukan magnitudo. Model optimal (k=22, *distance weights*, Cosine) diintegrasikan dalam sistem klasifikasi sentimen real-time. Hasil uji sistem menunjukkan respons cepat dan akurasi tinggi serupa training. Ini menunjukkan generalisasi dari pipeline yang baik dan keandalan dalam lingkungan operasional nyata.

KESIMPULAN

Penerapan metode KNN dalam penelitian ini membuktikan bahwa pemilihan hyperparameter yang tepat berpengaruh besar terhadap akurasi model dalam analisis sentimen. Dengan menggunakan teknik *GridSearchCV*, dilakukan eksplorasi kombinasi parameter seperti jumlah tetangga terdekat (*k*), metode pembobotan (*weights*), dan jenis metrik jarak. Hasil tuning menunjukkan bahwa konfigurasi optimal terjadi pada nilai *k* = 22, pembobotan *distance*, dan metrik jarak *cosine*, yang secara signifikan memberikan hasil akurasi pelatihan tertinggi sebesar 98,80%. Keberhasilan ini juga tidak lepas dari tahapan preprocessing data yang menyeluruh, yang mencakup pembersihan teks, normalisasi,

tokenisasi, hingga stemming. Proses tersebut berhasil menyaring kata-kata yang relevan dan menyederhanakan struktur teks, sehingga model dapat lebih akurat dalam mengenali dan mengklasifikasikan pola sentimen dalam data.

Penelitian selanjutnya disarankan untuk menguji model KNN pada dataset yang lebih besar, seimbang, dan bervariasi secara topik dan struktur bahasa untuk mengetahui sejauh mana model dapat melakukan generalisasi pada data nyata. Selain itu, pembandingan performa KNN dengan algoritma klasifikasi lain seperti Support Vector Machine (SVM), Naive Bayes, atau Random Forest penting dilakukan untuk menilai keunggulan relatif masing-masing metode. Disarankan pula untuk mengembangkan proses preprocessing dengan memperluas kamus normalisasi serta memperbaiki daftar *stopwords* agar lebih sesuai dengan konteks lokal dan bahasa yang digunakan, sehingga kualitas fitur yang diekstraksi semakin optimal.

DAFTAR PUSTAKA

Achmad, R. R., & Haris, M. (2023). Hyperparameter Tuning Deep Learning for Imbalanced Data. *Tepian*, *4*(2), 90–101. https://doi.org/10.51967/tepian.v4i2.2216

Azahra, N. M., & Setiawan, E. B. (2023). Sentence-Level Granularity Oriented Sentiment Analysis of Social Media Using Long Short-Term Memory (LSTM) and IndoBERTweet Method. *Jurnal Ilmiah Teknik Elektro Komputer Dan Informatika (JITEKI)*, *9*(1), 85–95. https://doi.org/10.26555/jiteki.v9i1.25765

Cano-Marin, E., Mora-Cantallops, M., & Sánchez-Alonso, S. (2023). Twitter as a predictive system: A systematic literature review. *Journal of Business Research*, *157*(December 2022). https://doi.org/10.1016/j.jbusres.2022.113561

Choo, S., & Kim, W. (2023). A study on the evaluation of tokenizer performance in natural language processing. *Applied Artificial Intelligence*, *37*(1). https://doi.org/10.1080/08839514.2023.2175112

Dewi, S., & Arianto, D. B. (2023). Twitter Sentiment Analysis Towards Qatar as Host of the 2022 World Cup Using Textblob. *Journal of Social Research*, *2*(2), 443–455. https://doi.org/10.55324/josr.v2i2.615

Gunawan, D., Saniyah, Z., & Hizriadi, A. (2019). Normalization of abbreviation and acronym on microtext in Bahasa Indonesia by using dictionary-based and longest common subsequence (LCS). *Procedia Computer Science*, *161*, 553–559. https://doi.org/10.1016/j.procs.2019.11.155

Halder, R. K., Uddin, M. N., Uddin, M. A., Aryal, S., & Khraisat, A. (2024). Enhancing K-nearest neighbor algorithm: a comprehensive review and performance analysis of modifications. *Journal of Big Data*, *11*(1). https://doi.org/10.1186/s40537-024-00973-y

Iparraguirre-Villanueva, O., Guevara-Ponce, V., Sierra-Liñan, F., Beltozar-Clemente, S., & Cabanillas-Carbonell, M. (2022). Sentiment Analysis of Tweets using Unsupervised Learning Techniques and the K-Means Algorithm. *International Journal of Advanced Computer Science and Applications*, *13*(6), 571–578. https://doi.org/10.14569/IJACSA.2022.0130669

Olabiyi, W., Olaoye, G., & Daniel, O. (2024). *Natural language processing (nlp) with nltk and spacy*. *November*.

Oladipupo, M. A., Obuzor, P. C., Bamgbade, B. J., Olagunju, K. M., Adeniyi, A. E., & Ajagbe, S. A. (2023). An Automated Python Script for Data Cleaning and Labeling using Machine Learning Technique. *Informatica (Slovenia)*, *47*(6), 219–232. https://doi.org/10.31449/inf.v47i6.4474

Parhusip, M., Sudianto, S., & Laksana, T. G. (2023). Sentiment Analysis of the Public Towards the Kanjuruhan Tragedy with the Support Vector Machine Method. *JUITA : Jurnal Informatika*, *11*(2), 241. https://doi.org/10.30595/juita.v11i2.17405

Pradana, A. W., & Hayaty, M. (2019). The Effect of Stemming and Removal of Stopwords on the Accuracy of Sentiment Analysis on Indonesian-language Texts. *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, *4*(3), 375–380. https://doi.org/10.22219/kinetik.v4i4.912

Rifaldi, D., Fadlil A., & Herman, H. (2023). Teknik Preprocessing Pada Text Mining Menggunakan Data Tweet "Mental Health." *Decode: Jurnal Pendidikan Teknologi Informasi*, *3*(2), 161–171. https://doi.org/10.51454/decode.v3i2.131

Rizki, M., Hermawan, A., & Avianto, D. (2024). Optimization of Hyperparameter K in K-Nearest Neighbor Using Particle Swarm Optimization. *JUITA : Jurnal Informatika*, *12*(1), 71. https://doi.org/10.30595/juita.v12i1.20688

Rodríguez-Ibánez, M., Casánez-Ventura, A., Castejón-Mateos, F., & Cuenca-Jiménez, P. M. (2023). A review on sentiment analysis from social media platforms. *Expert Systems with Applications*, *223*(July). https://doi.org/10.1016/j.eswa.2023.119862

Rukmini, E., Angelina, H., & Anggreni, V. C. (2023). Indonesia higher education's online learning during the pandemic state. *International Journal of Evaluation and Research in Education*, *12*(4), 2286–2301. https://doi.org/10.11591/ijere.v12i4.25103

Siino, M., Tinnirello, I., & LaCascia, M. (2024). Is text preprocessing still worth the time? A comparative survey on the influence of popular preprocessing methods on Transformers and traditional classifiers. *Information Systems*, *121*(December 2023), 102342. https://doi.org/10.1016/j.is.2023.102342

Trianda, D., Hartama, D., Engineering, I., Program, S., Systems, I., & Program, S. (2025). *Comparison of Hyperparameter Tuning Methods for Optimizing K-Nearest Neighbor Performance in Predicting Hypertension*. *18*(1), 111–121.

UNICEF Indonesia, & Agency, National Development Planning, P. (2021). *The situation of children and young people in indonesian cities*. 124.

Uyun, M. (2025). *From Isolation to Engagement : Understanding and Addressing Online Learning Challenges among University Students in Indonesia*. *17*, 1806–1818. https://doi.org/10.35445/alishlah.v17i1.6264