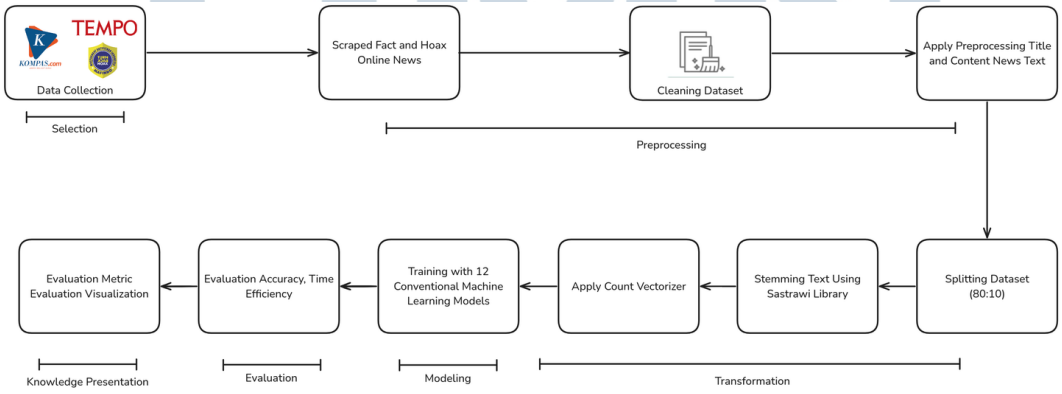
BAB 3
METODOLOGI PENELITIAN

3.1 Knowledge Discovery in Databases Framework

Penelitian ini diawali dengan studi literatur komprehensif dan diskusi mendalam bersama supervisi penelitian untuk mengidentifikasi pendekatan optimal dalam pengembangan model deteksi hoaks. Proses ini mencakup analisis perbandingan berbagai model *linear*, *ensemble* dan *large language model* (*LLM*), metode optimisasi, serta evaluasi *library* Python yang relevan untuk implementasi sistem. Fokus utama diberikan pada pemilihan metode yang efektif untuk pemrosesan teks bahasa Indonesia dan teknik klasifikasi yang sesuai. Hasil dari tahap ini mengarah pada desain arsitektur sistem dan pemilihan *tools* pengembangan yang optimal untuk implementasi model deteksi hoaks berbasis teks.

Penelitian ini menggunakan pendekatan metodologi *Knowledge Discovery in Databases* (*KDD*) untuk menemukan pengetahuan yang berguna dari data berita dalam jumlah besar dan kompleks. Gambar 3.1 menunjukkan alur penelitian yang dirancang berdasarkan kerangka kerja *KDD*. *KDD* adalah proses iteratif yang melibatkan beberapa tahapan, mulai dari pemahaman domain, pengumpulan data, pembersihan data, transformasi data, pemilihan fitur, pemodelan, evaluasi, hingga interpretasi hasil analisis [10].



Gambar 3.1. Research Framework based on Knowledge Discovery in Databases (KDD) Methodology

3.2 Pengumpulan Data

Proses pengumpulan data dilakukan melalui teknik *web crawling* dari portal berita terpercaya seperti Kompas, Tempo, serta basis data hoaks Mafindo. Dari hasil *crawling* tersebut, diperoleh lebih dari 29.553 artikel berita yang kemudian diproses untuk membentuk dataset pelatihan. Dataset ini mencakup artikel-artikel yang telah terverifikasi kebenarannya maupun yang telah dikonfirmasi sebagai hoaks. Perpaduan dari ketiga sumber ini memberikan dataset yang seimbang, seperti yang ditunjukkan dalam Tabel 3.1.

Tabel 3.1. Distribusi Dataset Berdasarkan Jenis Berita dan Sumber

Sumber	**Berita Fakta**	**Berita Hoaks**
Tempo & Kompas	15.495	-
Mafindo	495	14.057
Total	15.495	14.057

3.3 Pembersihan dan Pengolahan Data

Tahap *preprocessing* dilakukan terhadap seluruh dataset yang mencakup beberapa langkah penting, pembersihan teks dari karakter khusus, konversi ke huruf kecil (*case folding*), penghapusan *stopwords* bahasa Indonesia dengan *library* sastrawi, dan proses *tokenization*. Untuk melakukan *preprocessing* data, beberapa *library* populer dalam bahasa pemrograman Python digunakan. *Library re* dan *string* digunakan untuk menangani berbagai tugas pembersihan teks, seperti mengubah teks menjadi *lowercase*, menghapus teks dalam tanda kurung siku, menghapus tautan, menghapus tanda baca, dan menghapus kata-kata yang mengandung angka. Proses *tokenisasi*, yaitu memecah teks menjadi unit-unit kata individu, dilakukan dengan menggunakan *library nltk* (*Natural Language Toolkit*). Untuk menghapus *emoji* yang mungkin terdapat dalam teks, *library emot* digunakan. Dengan menggunakan kombinasi *library-library* ini, *preprocessing* data dapat dilakukan secara efisien dan efektif, menghasilkan dataset yang bersih dan siap untuk digunakan dalam pengembangan model *machine learning* untuk deteksi berita palsu. Tahapan tersebut merupakan metode umum sudah cukup untuk *preprocessing* sebagian besar kumpulan data teks [11].

3.4 Transformasi Data

Tahap transformasi data merupakan bagian penting dalam proses *Knowledge Discovery in Databases* (*KDD*). Pada tahap ini, data yang telah melalui proses pembersihan akan ditransformasi ke dalam format yang sesuai untuk proses analisis. Teknik transformasi yang digunakan dalam penelitian ini adalah *stemming* dan *feature extraction* dengan *count vectorizer* yang bertujuan untuk mengubah kata-kata menjadi bentuk dasarnya.

3.4.1 Sastrawi Stemmer

Dalam penelitian ini, proses *stemming* dilakukan menggunakan *library Sastrawi*, sebuah *library stemming* khusus untuk Bahasa Indonesia yang mengimplementasikan algoritma Nazief dan Adriani. Proses *stemming* ini diterapkan pada kolom judul (*title*) dan konten (*content*) dari dataset, baik untuk data latih maupun data uji. Implementasi *Sastrawi Stemmer* dilakukan dengan menggunakan kelas `StemmerFactory` yang menyediakan metode untuk membuat objek *stemmer*. Berikut pengimplementasian sastrawi stemmer:

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

    def stem_text(text):
        return stemmer.stem(text)

    df_train['title'] = df_train['title'].apply(stem_text)
    df_train['content'] = df_train['content'].apply(stem_text)
    df_test['title'] = df_test['title'].apply(stem_text)
    df_test['content'] = df_test['content'].apply(stem_text)

```

Listing 3.1: Inisialiasasi StemmerFactory

Proses *stemming* ini mengubah kata-kata seperti ”membaca”, ”dibaca”, ”pembacaan” menjadi kata dasar ”baca”. Transformasi ini penting untuk mengurangi variasi kata yang memiliki makna dasar yang sama, sehingga dapat meningkatkan efektivitas dalam analisis teks selanjutnya. Penggunaan *Sastrawi Stemmer* dipilih karena kemampuannya dalam menangani kompleksitas imbuhan dalam Bahasa Indonesia dengan tingkat akurasi yang baik.

3.4.2 Count Vectorizer

Proses vektorisasi teks menggunakan *CountVectorizer* dilakukan untuk mengubah dokumen teks menjadi vektor berdasarkan frekuensi kemunculan kata. Parameter yang dioptimalkan mencakup *min_df*, *max_df*, dan *max_features* untuk mengurangi dimensi fitur dan menghilangkan kata-kata yang terlalu jarang atau terlalu umum muncul dalam dataset.Implementasi *CountVectorizer* dalam penelitian ini menggunakan library *scikit-learn* dengan beberapa parameter yang disesuaikan untuk mengoptimalkan hasil vektorisasi. Pemilihan didasarkan pada pertimbangan untuk mengurangi dimensionalitas data dengan membatasi jumlah fitur, menghilangkan noise dari kata-kata yang terlalu umum atau terlalu jarang, meningkatkan efisiensi komputasi dengan mempertahankan fitur-fitur yang paling informatif, serta memastikan kualitas representasi teks yang dihasilkan tetap memadai untuk analisis selanjutnya. Berikut adalah parameter yang digunakan.

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(
    # Vocabulary options
    max_df=0.80,
    min_df=6,
    max_features=2500,

    # Stop words
    stop_words=stopwords,
    )
```

Listing 3.2: Implementasi CountVectorizer

3.5 Pembuatan Model

Pembangunan model klasifikasi hoaks dilakukan dengan mengimplementasikan dua algoritma: *Logistic Regression* dan *Ridge Classifier*. *Logistic Regression* dipilih karena kemampuannya dalam menangani klasifikasi biner dan interpretabilitas hasilnya, sementara *Ridge Classifier* digunakan untuk mengatasi masalah *overfitting* melalui regularisasi L2. Kedua model dilatih menggunakan dataset yang telah melalui proses vektorisasi, baik dengan *CountVectorizer* maupun *TF-IDF*.

3.6 Evaluasi Model

Evaluasi performa model dilakukan menggunakan metrik *accuracy*, *precision*, *recall*, *F1-score*, dan efisiensi waktu pelatihan. Penggunaan berbagai metrik evaluasi ini penting untuk memahami kemampuan model dalam mengidentifikasi berita hoaks maupun berita valid secara komprehensif. Model juga diuji ketahanannya melalui validasi silang untuk memastikan konsistensi performa pada berbagai subset data.

3.7 Pembuatan Laporan dan Dokumentasi

Tahap akhir penelitian mencakup penyusunan dokumentasi mengenai implementasi *Logistic Regression* dan *Ridge Classifier* dalam deteksi berita hoaks berbahasa Indonesia. Laporan penelitian ini menguraikan metodologi, hasil eksperimen, analisis performa model, serta rekomendasi untuk pengembangan lebih lanjut. Dokumentasi teknis juga mencakup detail implementasi dan konfigurasi model.