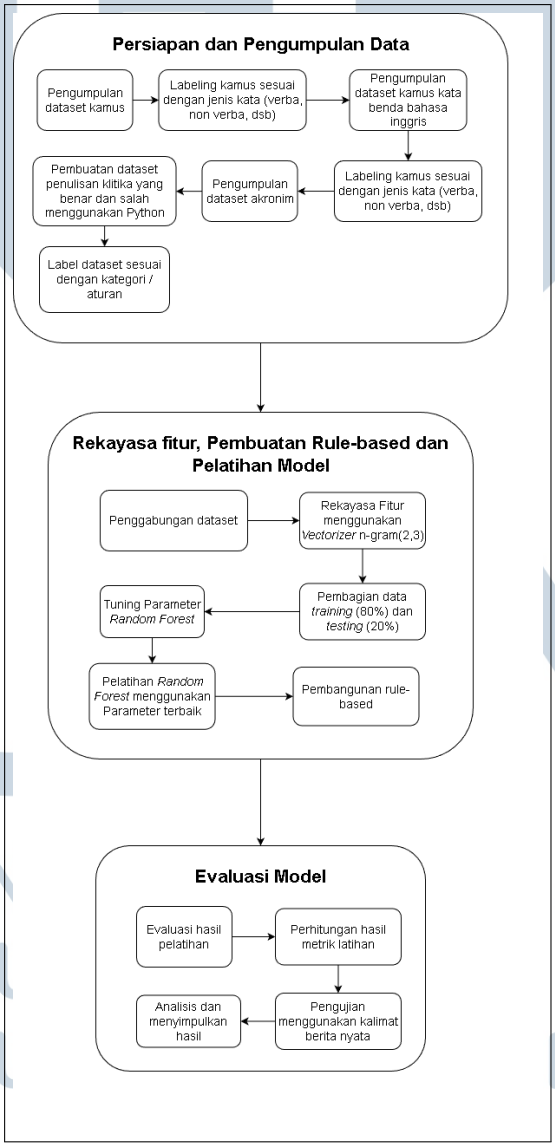
# BAB 3
# METODOLOGI PENELITIAN

## 3.1 Tahapan Penelitian

Penelitian ini dilakukan melalui beberapa tahapan, yaitu persiapan dan pengumpulan dataset, pra-pemrosesan, rekayasa fitur, pembuatan *rule-based*, pelatihan model, serta evaluasi metrik. Alur tahapan penelitian ditunjukkan pada Gambar 3.1.



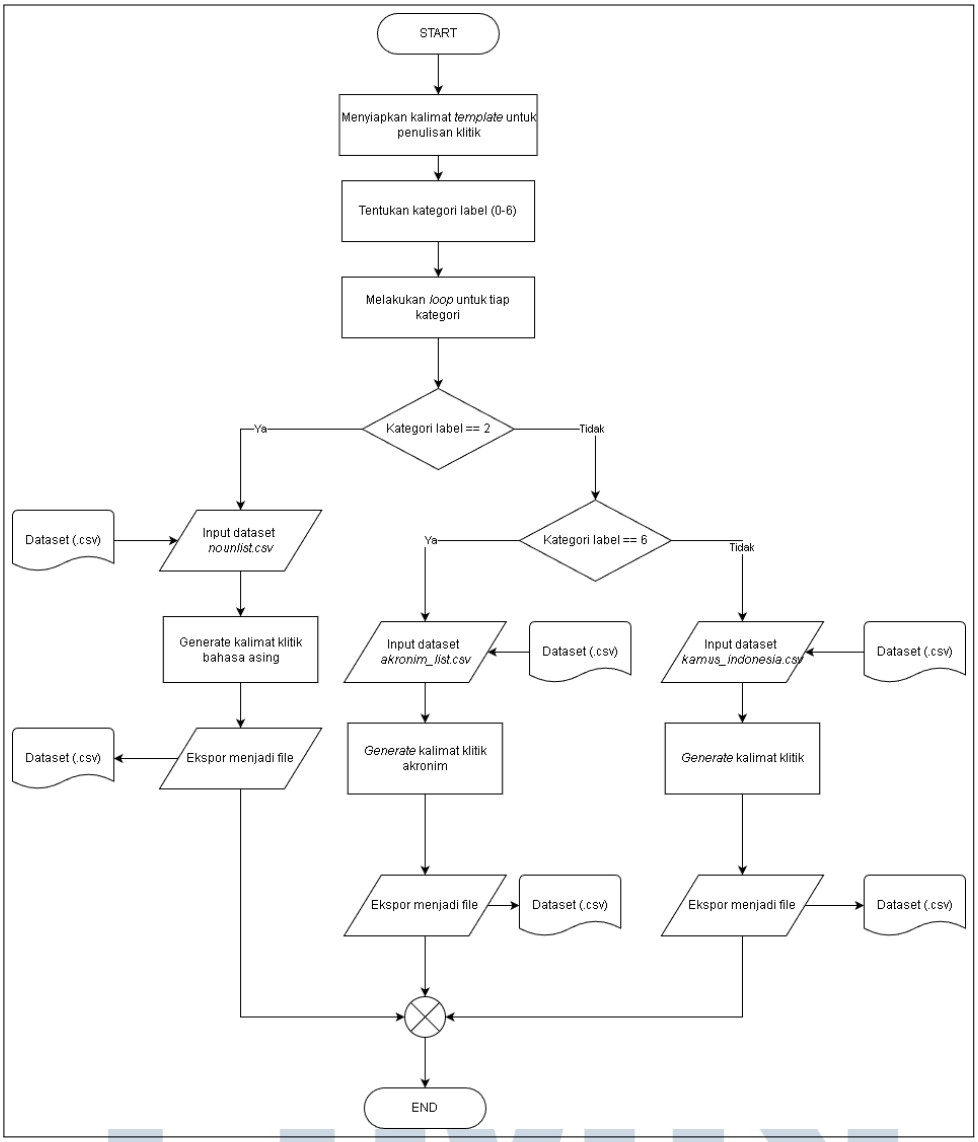

Gambar 3.1. *Pipeline* penelitian

### 3.1.1 Spesifikasi Sistem

Model ini dibuat pada hardware berupa laptop dengan spesifikasi *ASUS ROG Zephyrus G15 GA503QM* yang menjalankan Sistem Operasi *Windows 11 Home 64-bit (Build 22631)*. Perangkat ini dibekali prosesor *AMD Ryzen 9 5900HS* dengan 16 inti CPU berkecepatan 3.3 GHz, memori 16 GB RAM dengan ketersediaan 15.8 GB. Penelitian ini dilakukan dengan menggunakan *Jupyter Notebook* dan bahasa *Python* untuk mengembangkan model.

### 3.1.2 Persiapan dan Pengumpulan Data

Tahap persiapan dan pengumpulan data diawali dengan mencari dataset kamus Bahasa Indonesia dan Bahasa Inggris. Kamus Bahasa Indonesia diperoleh dari *GitHub* dengan jumlah 1.048.575 kata beserta jenis katanya (verba, nomina, dan sebagainya), sedangkan kamus Bahasa Inggris diperoleh dari *Kaggle* dengan jumlah 6.800 kata benda. Pengumpulan data dilanjutkan dengan memperoleh kumpulan dataset akronim atau singkatan melalui proses *scraping*, yang menghasilkan 3.178 singkatan. Tahap-tahap setelah itu dapat dilihat pada Gambar 3.2.




Gambar 3.2. *Flowchart* pembuatan dataset

Pembuatan dataset diawali dengan membuat *template* kalimat yang digunakan untuk membuat kalimat klitik yang bervariasi sesuai dengan kata-kata pada kumpulan dataset kamus dan akronim. Kumpulan dataset kamus dan akronim tersebut kemudian digunakan untuk membentuk dataset penulisan klitik benar dan salah. Dataset ini dibangun menggunakan bahasa pemrograman Python dengan memanfaatkan kata-kata yang berasal dari dataset yang telah dikumpulkan sebelumnya. Penulisan klitik salah dibagi menjadi enam kelas.

1. Akhiran *-ku, -mu, -nya* (*rumah ku*, *jalan-nya*).
2. Bahasa asing diakhiri klitik (*classnya*, *class nya*).

3. Kata religius diakhiri klitik (*TuhanKu*, *Tuhan Nya*).

4. Awalan *ku-* dipisah dari kata dasar (*ku ambil*).

5. Awalan *ku-*, *kau-* digabung dengan kata berimbuhan (*kumengerjakan*, *kaumengambil*).

6. Klitik pada akronim atau singkatan (*SIM ku*, *KTPnya*).

Dataset penulisan klitik benar terdiri dari 293.890 data, sedangkan dataset penulisan klitik salah (kelas satu hingga enam) masing-masing berjumlah 100.000, 100.000, 148.500, 109.599, 147.296, dan 129.690 data. Secara keseluruhan, jumlah data yang digunakan adalah 1.028.975. Contoh dataset dapat dilihat pada tabel 3.1.

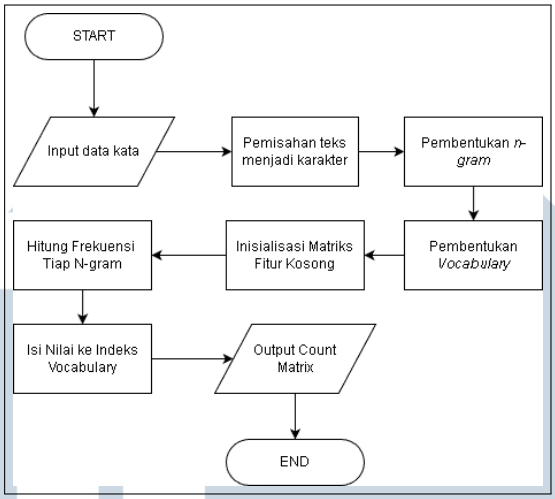
Tabel 3.1. Tabel Contoh Dataset

| kalimat | highlight_kata | label |
|---|---|---|
| Sejak dulu aku selalu merawat adat mu | adat mu | 1 |
| Kadang Aku menyerang tanpa alasan | Aku menyerang | 0 |
| Kadang kumerangkul tanpa alasan | kumerangkul | 5 |
| Dia selalu kau-tunjuk pagi hari | kau-tunjuk | 4 |

Pada dataset tersebut, terdapat kolom kalimat, *highlight_kata* dan label. Kolom kalimat merupakan contoh kalimat yang mengandung klitik. Kolom *highlight_kata* merupakan kata yang mengandung atau termasuk dalam kelas penggunaan klitik. Kolom *highlight_kata* yang akan digunakan untuk *train* model *random forest*. Kolom label, merupakan kolom kelas dari nol sampai enam sesuai dengan kelas-kelas yang sudah ditentukan sebelumnya.

### 3.1.3 Rekayasa Fitur

Alur rekayasa fitur dapat dilihat pada Gambar 3.3, rekayasa fitur pada penelitian ini dilakukan menggunakan metode *n-gram* berbasis karakter (*character n-gram*), yang mencakup *bigram* dan *trigram*, untuk menangkap pola lokal pada teks. Pendekatan ini memungkinkan model mengenali hubungan antar karakter yang membentuk klitik, sehingga model tidak hanya memahami karakter secara individu, tetapi juga urutan karakter yang muncul secara berurutan.




Gambar 3.3. Alur Rekayasa Fitur

Proses ekstraksi fitur diawali dengan pembacaan data teks, kemudian setiap teks diuraikan menjadi urutan karakter. Proses selanjutnya, membentuk *character n-gram* dengan rentang dua hingga tiga karakter. Seluruh *n-gram* yang dihasilkan dari data latih dikumpulkan dan disusun menjadi sebuah *vocabulary* yang berisi daftar unik *n-gram*.

Setiap data teks kemudian direpresentasikan ke dalam bentuk vektor numerik dengan menghitung frekuensi kemunculan masing-masing *n-gram* berdasarkan *vocabulary* yang telah dibentuk. Proses vektorisasi ini dilakukan menggunakan *CountVectorizer*, yang menghasilkan matriks fitur dalam bentuk *sparse matrix*, di mana setiap baris merepresentasikan satu data teks dan setiap kolom merepresentasikan satu *n-gram* tertentu. Penggunaan *bigram* dan *trigram* karakter dipilih karena pola penulisan klitik dalam bahasa Indonesia umumnya terbentuk dari dua hingga tiga karakter.

## 3.2 Pembuatan Model

Data dibagi menjadi dua bagian, yaitu 80% untuk data latih dan 20% untuk data uji, guna mengukur kemampuan generalisasi model terhadap data yang belum pernah dilihat sebelumnya. Sebelum tahap pelatihan, dilakukan proses *hyperparameter tuning* menggunakan metode *GridSearchCV* untuk mencari kombinasi parameter terbaik pada model yang digunakan, dengan total kombinasi sebanyak 144.

Model yang digunakan dalam penelitian ini adalah *Random Forest Classifier*. Setelah parameter terbaik diperoleh dari proses *tuning*, model dilatih menggunakan

data latih dan kemudian diuji menggunakan data uji. Evaluasi kinerja model dilakukan berdasarkan metrik *accuracy*, *precision*, *recall*, dan *f1-score*.

Proses *hyperparameter tuning* pada model *Random Forest* dilakukan dengan kombinasi parameter seperti yang ditunjukkan pada Tabel 3.2.

Tabel 3.2. Kombinasi Parameter *Random Forest*

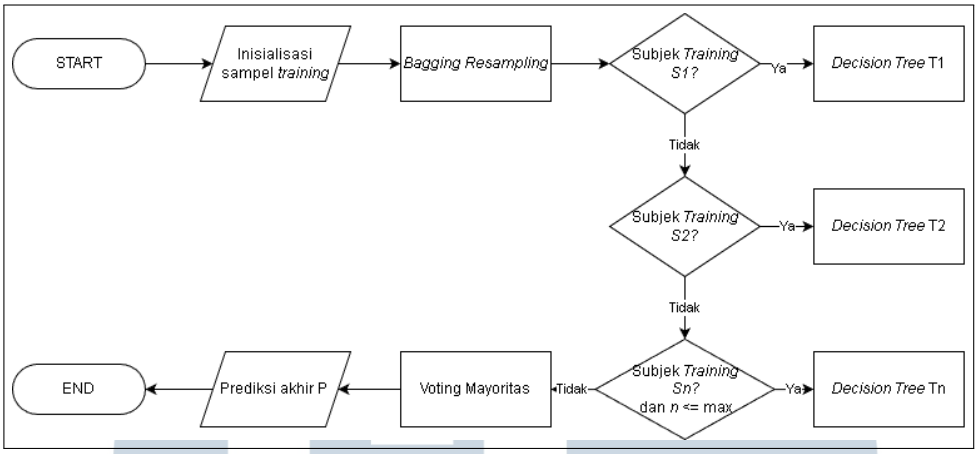
| | |
|---|---|
| *n_estimators* | 200, 300, 500 |
| *max_depth* | *None*, 20, 30 |
| *min_samples_split* | 2, 5 |
| *min_samples_leaf* | 1, 2 |
| *max_features* | *sqrt*, *log2* |
| *max_samples* | *None*, 0.8 |

Hasil proses *hyperparameter tuning* menghasilkan kombinasi parameter dengan nilai akurasi tertinggi sebagai berikut:

- *n_estimators*: 500
- *max_depth*: *None*
- *min_samples_split*: 2
- *min_samples_leaf*: 1
- *max_features*: *log2*
- *max_samples*: *None*

Parameter tersebut digunakan pada tahap pelatihan model *Random Forest* untuk membangun sistem deteksi kesalahan penggunaan klitik.

Pembuatan model mengikuti alur pada Gambar 3.4. Proses diawali dengan inisialisasi data latih yang telah direpresentasikan dalam bentuk vektor fitur hasil ekstraksi menggunakan metode *N-gram*. Model kemudian menerapkan metode *bagging* melalui proses *bootstrap sampling* untuk membentuk beberapa *subset* data latih secara acak dengan pengembalian. Setiap *subset* data digunakan untuk melatih sebuah *decision tree* secara independen sehingga dihasilkan sejumlah *decision tree* yang membentuk sebuah *Random Forest*. Setelah seluruh pohon keputusan selesai dilatih, masing-masing pohon menghasilkan prediksi terhadap data masukan, kemudian hasil prediksi tersebut digabungkan menggunakan mekanisme agregasi berupa *majority voting* untuk memperoleh prediksi akhir model.




Gambar 3.4. Alur Pembuatan Model

## 3.3 Penyusunan Rule-Based

Pendekatan *rule-based* dalam penelitian ini digunakan untuk melengkapi hasil prediksi model pembelajaran mesin dengan menambahkan seperangkat aturan yang dirancang berdasarkan karakteristik bahasa Indonesia. Pendekatan ini bertujuan untuk memperbaiki hasil klasifikasi pada kasus-kasus khusus yang sulit diprediksi hanya dengan model berbasis data.

Beberapa aturan (*rules*) disusun untuk mengidentifikasi pola-pola tertentu pada kata, seperti kata dasar atau akronim. Jika sebuah kata menggunakan klitik, sistem akan memisahkan bagian dasar dan bentuk klitik. Bagian dasar tersebut kemudian dibandingkan dengan daftar akronim maupun kamus kata. Apabila bagian dasar ditemukan dalam daftar akronim dan tidak memiliki sufiks seperti *-ku*, *-mu*, atau *-nya*, maka aturan akan menimpa prediksi model dengan label tertentu (misalnya kelas keenam) karena akronim yang benar harus diikuti dengan tanda hubung (-), contohnya *KTP-ku*.

Pola prefiks seperti *ku* atau *kau* di awal kata juga terdapat pola tertentu seperti prefiks *ku* dan *kau* digabung apabila kata yang mengikuti itu merupakan kata verba dasar. Jika pola ini ditemukan, prefiks tersebut dihapus, lalu bagian kata yang tersisa diperiksa dalam kamus kata dasar. Jika kata tersebut termasuk dalam kategori verba dan belum memiliki bentuk dasar, maka aturan ini menimpa hasil prediksi model dengan label lain (misalnya kelas kelima).

Penyusunan *rule-based* ini digunakan sebagai mekanisme pendukung untuk membantu model dalam melakukan deteksi dengan lebih akurat. *Rule-based* tidak dimaksudkan untuk menggantikan proses prediksi model, tetapi berperan sebagai

pengecekan tambahan pada kondisi tertentu.

Apabila kata yang terdeteksi oleh model ternyata ditemukan dalam salah satu *rule*, seperti kamus Bahasa Indonesia, Bahasa Inggris, atau daftar akronim, maka sistem akan menyesuaikan hasil deteksi berdasarkan rule tersebut. Penyesuaian ini dilakukan karena beberapa kelas, seperti bahasa asing atau akronim, memiliki pola yang sangat mirip sehingga model berpotensi membuat kesalahan. Jika kata yang terdeteksi tidak ditemukan pada sumber-sumber tersebut, maka hasil akhir tetap mengikuti prediksi model sepenuhnya.