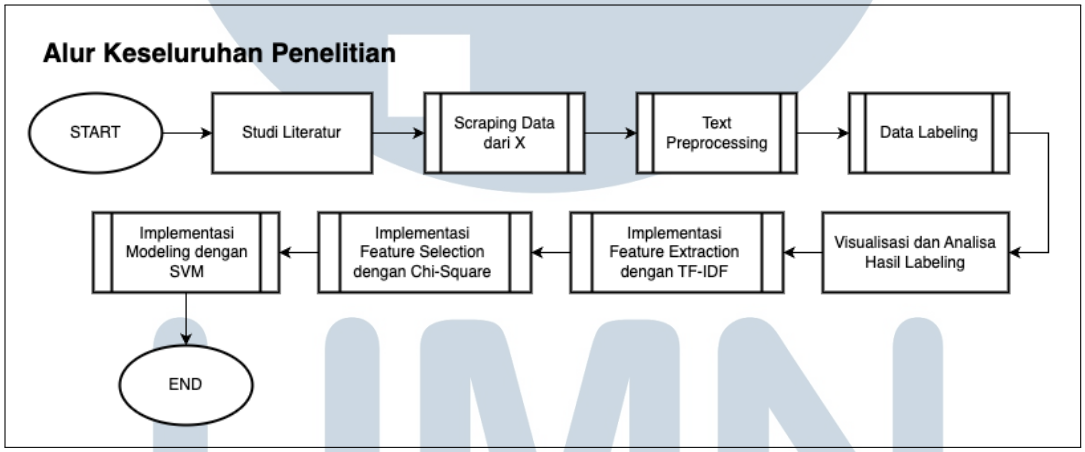
BAB 3
METODOLOGI PENELITIAN

3.1 Alur Diagram Penelitian

Keseluruhan alur penelitian dari awal hingga akhir mengacu pada langkah-langkah atau metodologi yang ditampilkan dalam *flowchart* berikut. Langkah-langkah tersebut dilakukan secara sistematis dimulai dari studi literatur, *scraping* data dari X dengan Tweet-Harvest, *text preprocessing*, data *labeling* dengan Sentistrength_id, visualisasi dan analisa hasil *labeling* dengan WordCloud, implementasi *feature extraction* dengan TF-IDF, implementasi *feature selection* dengan Chi Square, hingga implementasi *modeling* dengan SVM. Gambar 3.1 berikut menunjukan keseluruhan alur metodologi penelitian yang dilakukan.



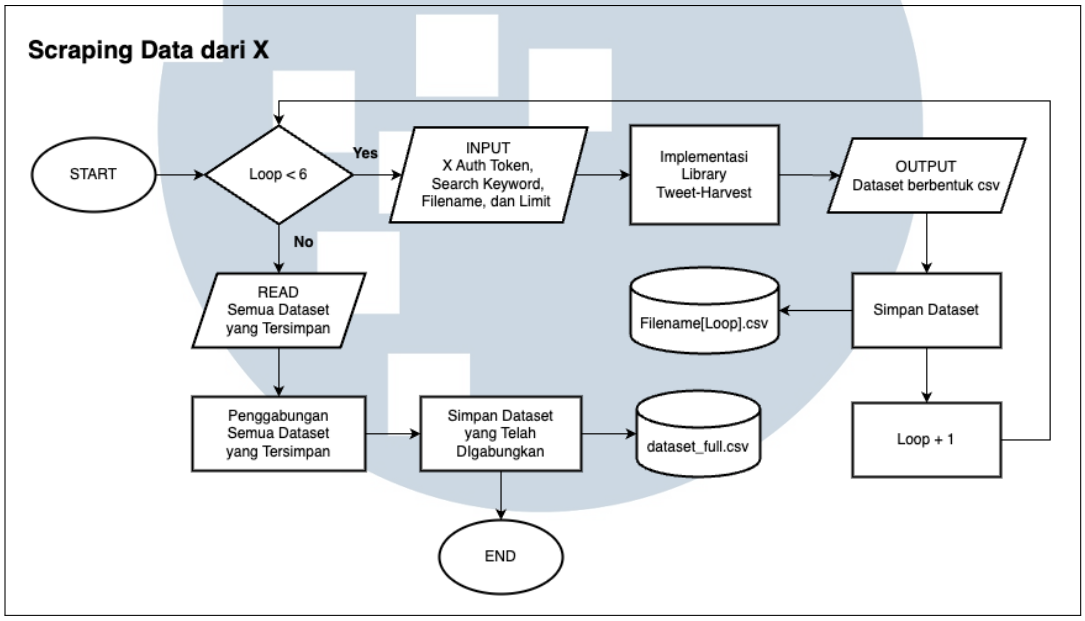
Gambar 3.1. Alur Metodologi Penelitian

3.2 Studi Literatur

Pada tahap awal penelitian ini, dilakukan telaah literatur untuk mendalami teori dan informasi yang diperlukan dalam penelitian yang diperoleh melalui pemahaman literatur ilmiah seperti jurnal dan buku. Literatur tersebut antara lain analisis sentimen, pengungsi Rohingya, *support vector machine*, TF-IDF, Chi Square, dan confusion matrix.

3.3 Data Scraping

Pada tahap ini dilakukan *scraping* data dari X dengan memanfaatkan *library* Tweet-Harvest. Tweet-Harvest merupakan sebuah *library open source* berbasis Node.js untuk pengambilan data *tweet* atau *post* X yang ditujukan hanya untuk keperluan edukasi. Gambar 3.2 berikut menunjukan alur data *scraping* yang dilakukan.



Gambar 3.2. Alur Data *Scraping*

Pada penelitian ini proses *scraping* data dilakukan sebanyak 6 kali dengan rentang waktu 5-6 hari per pengambilan data. Untuk menggunakan Tweet-Harvest diperlukan parameter utama yaitu X *authentication token*, *filename*, *search keyword*, dan *limit* pengambilan data.

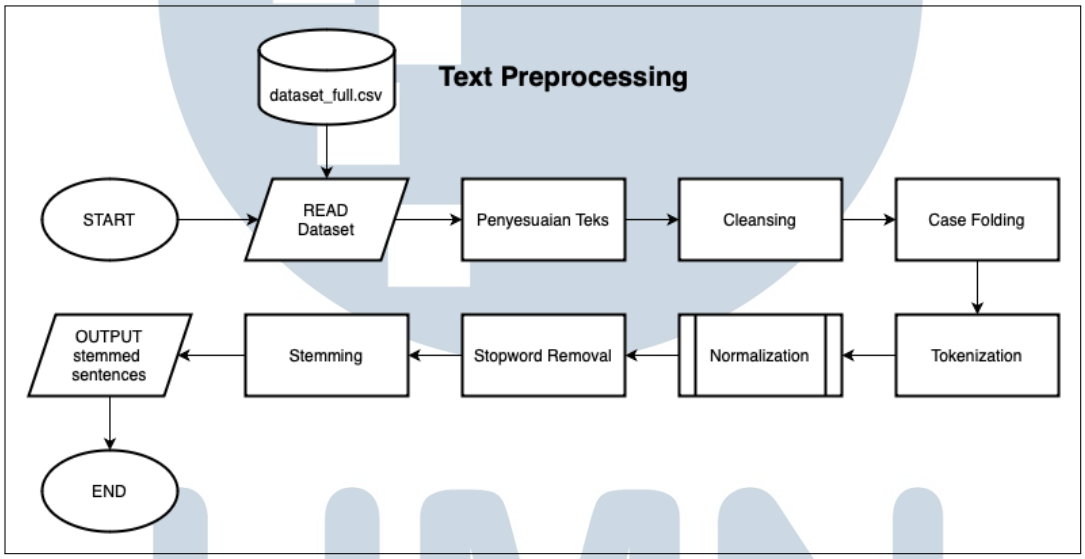
Pada parameter *search keyword* dipilih kata kunci ”pengungsi rohingya” atau ”rohingya aceh” untuk menemukan data *post* X yang relevan dan bersifat spesifik pada pengungsi rohingya Aceh tanpa mengurangi kemungkinan jangkauan data yang rendah. Selain itu pada *search keyword* juga terdapat rentang waktu pengambilan data yang ditentukan, ditandai dengan keyword ”*since*” dan ”*until*”. Pada parameter *limit* dipilih batas pengambilan data sebanyak 5000 data per pengambilan data untuk menghindari X *rate limit*, yaitu kebijakan baru X untuk yang membatasi pengguna untuk mengakses *post*.

Proses *scraping* data menghasilkan 6 *dataset* yang disimpan satu per satu

dalam bentuk file csv. Setelah semua data telah didapatkan, data kemudian digabungkan menjadi satu *dataset* penuh untuk proses selanjutnya.

3.4 Text Preprocessing

Pada tahap ini dilakukan *text preprocessing*, yaitu proses pembersihan data mentah berupa teks dengan mengeliminasi unsur-unsur tertentu yang tidak terstruktur dan tidak dibutuhkan dalam penelitian, seperti adanya frasa informal dan *noise* sehingga dapat memudahkan pemrosesan selanjutnya [21]. Gambar 3.3 berikut menunjukan alur *text preprocessing* yang dilakukan.



Gambar 3.3. Alur *Text Preprocessing*

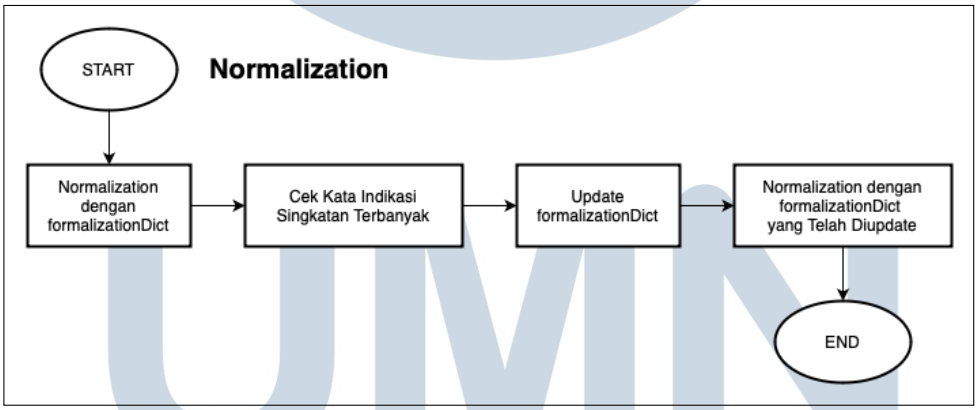
Tahap ini menggunakan dataset_full.csv hasil tahapan data *scraping* sebelumnya. Pada tahap awal dilakukan penyesuaian teks dengan tujuan untuk memudahkan pembacaan data pada proses berikutnya serta mengurangi bias pada proses analisis. Selanjutnya dilakukan 6 tahapan dalam *text preprocessing*, antara lain *cleansing*, *case folding*, *tokenizing*, *normalization*, *stopword removal*, dan *stemming*, dengan rincian sebagai berikut [22].

1. *Cleansing*: *Cleansing* bertujuan untuk mengeliminasi teks non alfabet seperti tanda baca, simbol, *emoticon*, dan alamat *website*.

2. *Case Folding*: *Case Folding* bertujuan untuk mengubah seluruh teks yang mengandung huruf kapital menjadi huruf kecil (teks *lower case*).

3. *Tokenizing*: *Tokenizing* bertujuan untuk memisahkan setiap kata dalam sebuah kalimat berdasarkan spasi menjadi kata tunggal yang disebut token.
4. *Normalization*: *Normalization* bertujuan untuk mengubah token singkatan dan token tidak baku menjadi token baku.
5. *Stopword Removal*: *Stopword Removal* bertujuan untuk menghapus token tidak bermakna contohnya adalah kata hubung seperti atau, dengan, dan tetapi.
6. *Stemming*: *Stemming* bertujuan untuk menghilangkan kata berimbuhan dalam token menjadi akar kata atau kata dasar.

Gambar 3.4 berikut menunjukkan alur *normalization* yang lebih rinci. Tahap awal dilakukan *normalization* menggunakan bantuan kamus formalizationDict yang diakses dari Github Pujangga Indonesian Natural Language Processing REST API. Kamus tersebut secara otomatis mengubah kata tidak baku menjadi kata baku selama kata tersebut tercantum dalam kamus.



Gambar 3.4. Alur *Normalization*

Selanjutnya dilakukan pengecekan kata indikasi singkatan (kata dengan panjang 2 huruf dan 3 huruf) dengan kemunculan terbanyak dalam data. Pengecekan menghasilkan 100 kata dengan panjang 2 huruf dan 100 kata dengan panjang 3 huruf. Berdasarkan pengecekan tersebut dilakukan *update* formalizationDict secara *manual*. Selanjutnya dilakukan *normalization* ulang dengan bantuan kamus formalizationDict yang telah di *update*. Tujuan dari *normalization manual* ini adalah meminimalisir adanya kata-kata bermakna yang tidak tercakup dalam kamus.

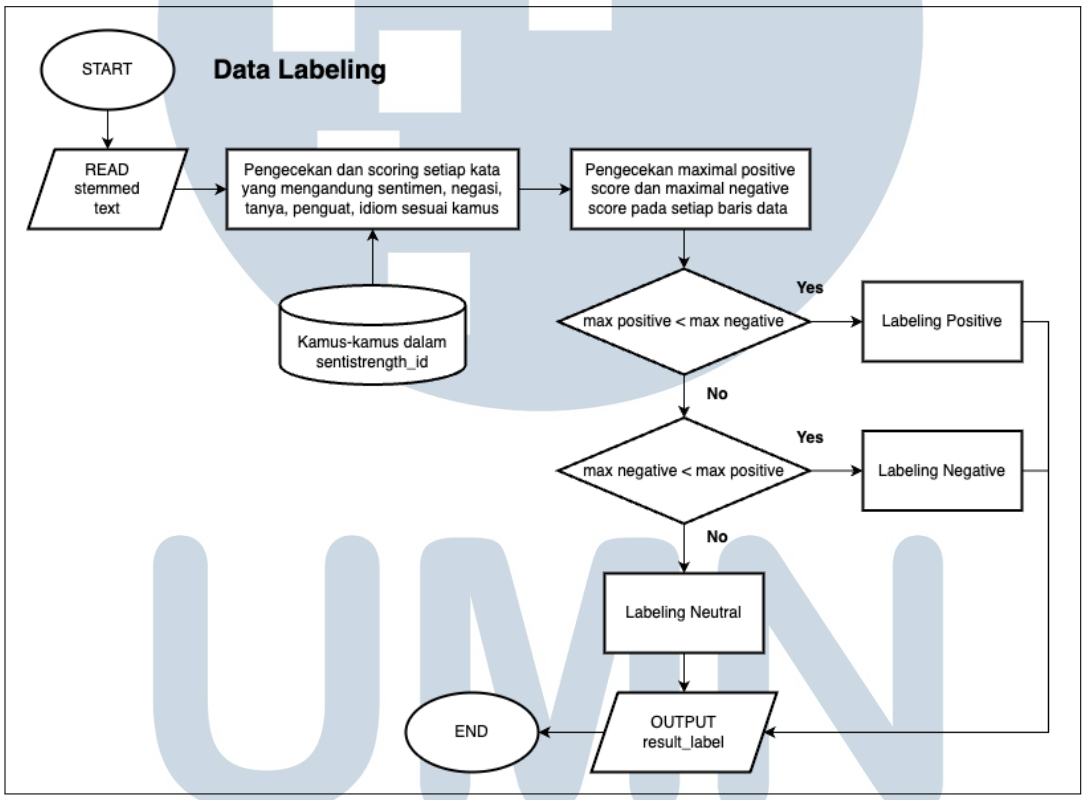
3.5 Data Labeling

Proses *labeling* data dilakukan secara otomatis menggunakan *library* lexicon sentistrength_id dari github Sentiment Strength Detection in Bahasa Indonesia [23]. Proses ini dimulai dengan membaca dataset dari proses sebelumnya. Pada tahap awal dilakukan pengecekan dan *scoring* setiap kata dari dataset yang mengandung sentimen, negasi, tanya, penguat, idiom sesuai kamus sentiwords_id, negatingword, questionword, idiom_id, dan boosterwords_id. Alur *scoring* dalam kode tersebut melibatkan beberapa tahap untuk menentukan *score* sentimen akhir dari sebuah teks. Berikut adalah alur *scoring* yang digunakan:

1. Iterasi Melalui Setiap Kata dalam Kalimat: Setiap kata dalam kalimat diperiksa satu per satu untuk menentukan *score* sentimennya.
2. Penanganan Kata Negasi: Jika kata sebelumnya adalah kata negasi, *score* sentimen kata saat ini dibalik.
3. Penanganan Kata Booster: Jika kata saat ini adalah kata booster dan *score* sentimen sebelumnya tidak netral, *score* sentimen kata saat ini ditambahkan atau dikurangkan dari *score* sentimen sebelumnya.
4. Penanganan Idiom: Jika *idiom* atau kiasan ditemukan, *score* sentimen kalimat diatur berdasarkan *score* kiasan yang ditemukan.
5. Penanganan Konsekuensi: Diperiksa apakah ada kata-kata sentimen yang berurutan. Jika ada, *score* sentimen kalimat ditingkatkan.
6. Penanganan Pengulangan: Jika terdapat karakter berulang lebih dari 2 kali dalam kata, *score* sentimen kata tersebut ditingkatkan atau diturunkan, tergantung pada apakah kata tersebut memiliki *score* positif atau negatif.
7. Penanganan Tanda Tanya dan Tanda Seru: Jika terdapat tanda tanya atau tanda seru dalam kalimat, *score* sentimen kalimat diatur sesuai dengan konfigurasi yang diberikan.
8. Perhitungan *Score* Maksimum Sentimen Kalimat: *Score* sentimen maksimum positif dan negatif untuk setiap kalimat dihitung berdasarkan *score* sentimen kata-kata dalam kalimat.

9. Klasifikasi Sentimen Keseluruhan Teks: Setelah semua kalimat diproses, *score* sentimen maksimum positif dan negatif dari semua kalimat diambil untuk menentukan klasifikasi sentimen keseluruhan teks.

Dari score tersebut dilakukan labelisasi yang terdiri dari 3 label, yaitu *negative*, *neutral*, dan *positive*. Score dengan nilai *maximal positive* >nilai *maximal negative* diberikan label *positive*, nilai *maximal negative* <nilai *maximal positive* diberikan label *negative*, dan ketika nilai sama diberikan label *neutral*. Gambar 3.5 berikut menunjukan alur *labeling* yang dilakukan.



Gambar 3.5. Alur *Labeling*

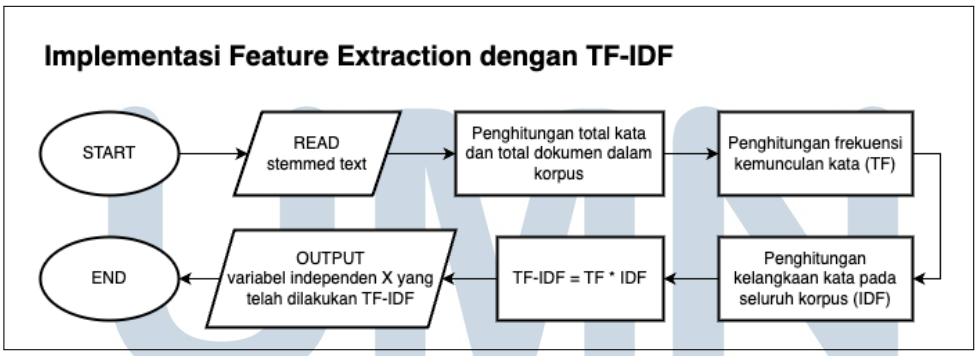
3.6 Visualisasi dan Analisa Hasil Labeling

Pada tahap ini dilakukan visualisasi dengan memanfaatkan *library* WordCloud dan matplot. Visualisasi dengan matplot menampilkan perbandingan jumlah data pada setiap label dalam bentuk *pie chart*. Sedangkan, visualisasi dengan WordCloud menampilkan frekuensi kata yang paling sering muncul dalam setiap sentimen dalam bentuk grafis, dimana semakin besar ukuran kata dalam

gambar menandakan semakin besar juga frekuensinya. Berdasarkan visualisasi hasil *labeling* sentimen ini dilakukan analisa terkait hubungan kata dengan dengan label sentimennya.

3.7 Implementasi Feature Extraction dengan TF-IDF

Pada tahap ini dilakukan *feature extraction* menggunakan TF-IDF. Tujuannya adalah untuk memberikan bobot dan mengubah data kategorikal menjadi data numerik. Term Frequency (TF) dipakai untuk mengukur seberapa sering sebuah kata muncul dalam dokumen, dimana kata-kata yang muncul lebih sering memiliki bobot yang lebih tinggi. Sementara Inverse Document Frequency (IDF) dipakai untuk mengukur seberapa umum atau langka sebuah kata di seluruh dokumen dalam dataset. Hasil nilai TF-IDF kemudian digunakan untuk membangun vektor representasi dokumen, di mana setiap komponen vektor mewakili bobot dari suatu kata dalam dokumen tersebut. Data yang digunakan adalah data kategorikal dari *stemmed text* hasil *text preprocessing* yang menghasilkan variabel independen berbobot X. Gambar 3.6 berikut menunjukan alur *feature extraction* yang dilakukan.

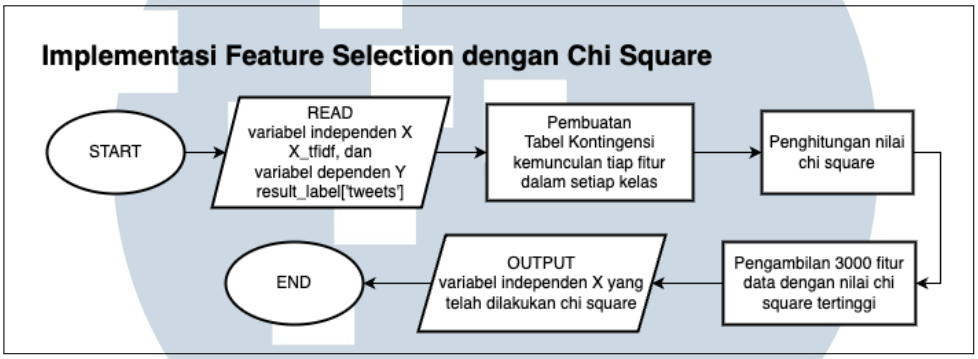


Gambar 3.6. Alur *Feature Extraction* dengan TF-IDF

3.8 Implementasi Feature Selection dengan Chi Square

Pada tahap ini dilakukan seleksi fitur menggunakan metode Chi Square. Tujuannya adalah untuk memilih fitur-fitur yang paling signifikan dalam menentukan kelas dokumen, mengurangi dimensi data, dan meningkatkan efisiensi. Chi-square mengukur seberapa signifikan kehadiran fitur tersebut dalam dokumen dikaitkan dengan label kelas. Pada tahap ini digunakan data X hasil tf-idf dan

data Y hasil labeling. Untuk setiap fitur dalam data dibuat tabel kontingensi yang mengukur frekuensi kemunculan fitur dalam dokumen dari masing-masing kelas. Nilai Chi Square dihitung berdasarkan tabel ini untuk menentukan ketergantungan antara fitur dan kelas. Fitur-fitur dengan nilai Chi Square tertinggi dipilih sebagai fitur terbaik yang paling signifikan dalam menentukan kelas dokumen. Gambar 3.7 berikut menunjukkan alur seleksi fitur yang dilakukan.



Gambar 3.7. Alur Seleksi Fitur dengan Chi-Square

3.9 Implementasi Modeling dengan Algoritma SVM

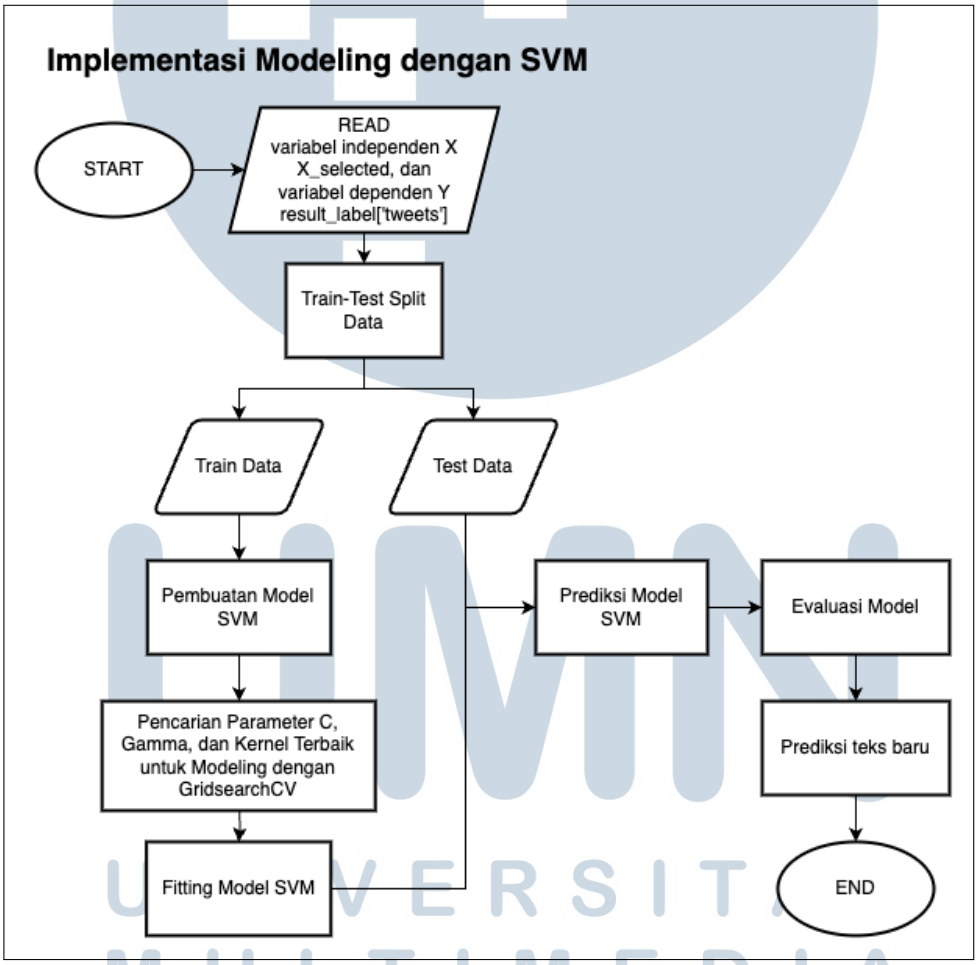
Pada tahap ini dilakukan implementasi model SVM dengan beberapa tahapan antaran lain *train-test split data*, pencarian parameter SVM terbaik, *modeling* SVM, evaluasi, dan prediksi teks baru yang dimasukkan secara manual. Pada tahap awal dilakukan pembagian data X hasil *feature extraction* dan *feature selection*, serta data y hasil *labeling* menjadi data *training* dan data *testing*. Data *training* digunakan dalam *modeling* menggunakan SVM, sedangkan data *testing* digunakan untuk proses prediksi dalam *modeling* dan evaluasi.

Selanjutnya dilakukan pencarian kombinasi *hyperparameter* C, gamma, dan kernel terbaik dalam *modeling* SVM digunakan *tools* gridsearchCV. GridsearchCV melakukan pelatihan dengan mencoba semua kombinasi parameter dan menggunakan *cross-validation* untuk mengevaluasi setiap kombinasi berdasarkan tingkat *accuracy*. Setelah didapatkan kombinasi *hyperparameter* terbaik dilakukan *modeling* dengan algoritma SVM menggunakan data *training* hasil *spliting* data. Data *testing* hasil *spliting* kemudian digunakan untuk melakukan prediksi.

Selanjutnya dilakukan evaluasi menggunakan Confusion Matrix. Performa *modeling* dari algoritma SVM dalam melakukan klasifikasi data tercermin dalam tabel Confusion Matrix. Tabel ini terdiri dari 3 kategori sentimen, yaitu positif,

netral, dan negatif. Dalam tabel ini, terdapat nilai True yang menunjukkan bahwa label yang diprediksi oleh model sama dengan label sebenarnya dari teks, dan nilai False yang menunjukkan bahwa label yang diprediksi oleh model berbeda dengan label sebenarnya dari teks. Hasil dari Confusion Matrix digunakan untuk menghitung metrik evaluasi penting seperti Akurasi (*Accuracy*), Presisi (*Precision*), *Recall*, dan *F1-score*.

Pada tahap akhir dilakukan prediksi pada teks baru yang dimasukkan secara manual untuk melihat keakuratan prediksi model pada kasus atau teks baru. Gambar 3.8 berikut menunjukan alur *modeling* secara keseluruhan.



Gambar 3.8. Alur *Modeling* dengan Algoritma SVM