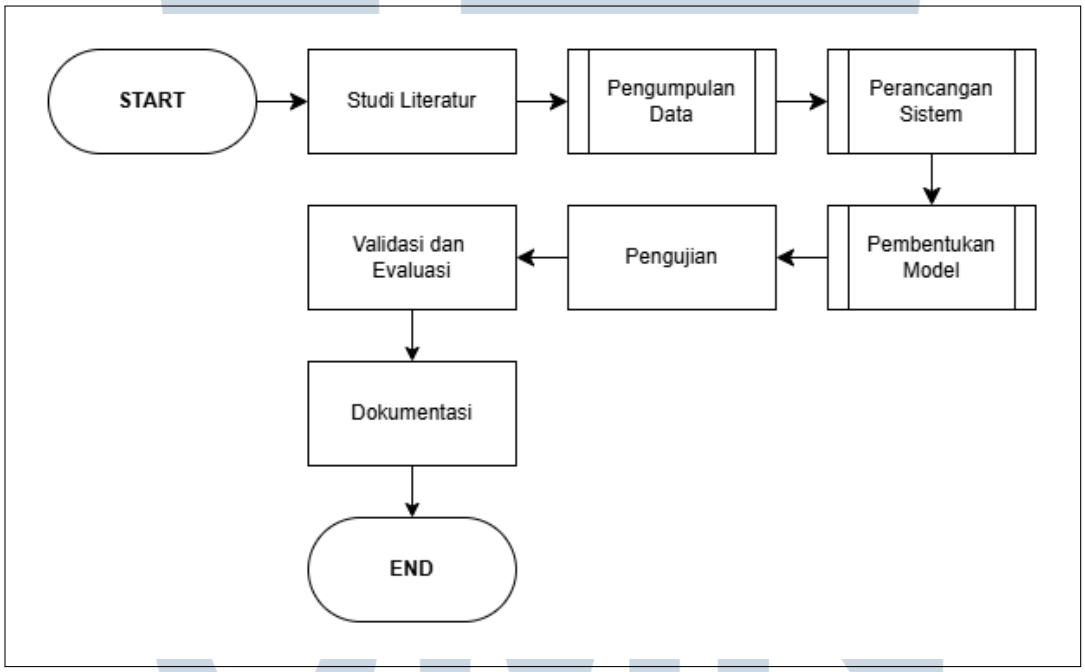
BAB 3
METODOLOGI PENELITIAN

Penelitian dilakukan menggunakan metodologi yang terdiri dari serangkaian langkah. Metodologi ini dirancang secara sistematis untuk memastikan validitas dan hasil penelitian. Setiap langkah dalam metodologi ini mencerminkan tahapan yang dilakukan mulai dari awal hingga akhir penelitian. Struktur metodologi yang digunakan dalam penelitian ini ditunjukkan pada Gambar 3.1 yang merupakan *flowchart* untuk memberikan gambaran mengenai alur kerja penelitian secara keseluruhan.



Gambar 3.1. *Flowchart* metodologi penelitian

3.1 Studi Literatur

Pada tahap ini dilakukan kajian terhadap penelitian-penelitian terdahulu yang berkaitan dengan analisis sentimen, khususnya analisis sentimen pada media sosial X (sebelumnya Twitter), serta implementasi algoritma *machine learning* serupa. Penelitian ini juga didukung oleh berbagai teori yang relevan, seperti teori mengenai analisis sentimen, media sosial Twitter, algoritma Naïve Bayes, SVM, *Ensemble Learning*, metode *Feature Extraction*, *cross validation*, dan *confusion matrix* sebagai dasar untuk pelaksanaan penelitian.

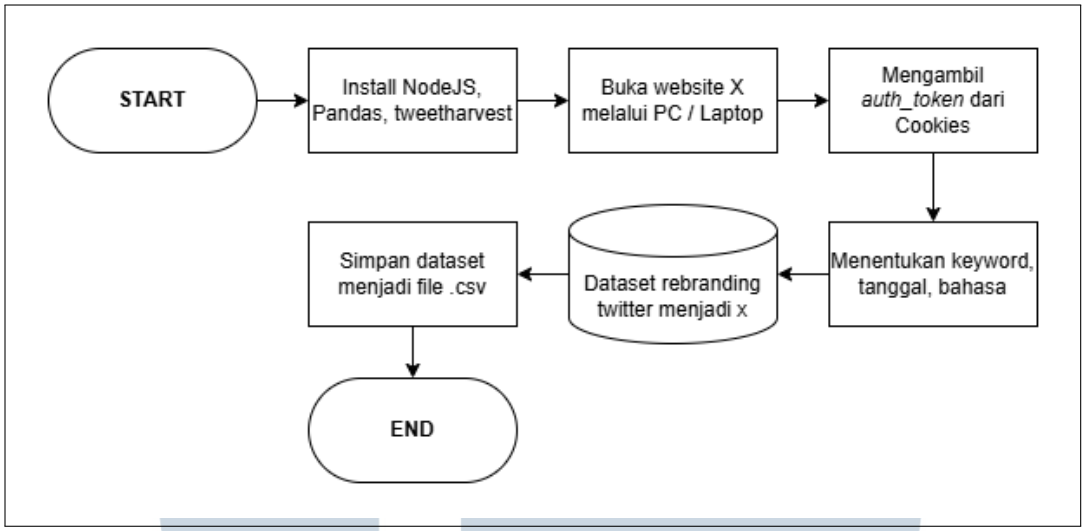
3.2 Pengumpulan Data

Tahap pengumpulan data yang digunakan untuk penelitian ini dilakukan dengan memanfaatkan Jupyter Notebook dan *tweet harvest*. *Tweet Harvest* merupakan *command-line tool* yang digunakan untuk memperoleh data dari *post* yang diunggah oleh pengguna X [37]. *Tweet Harvest* memungkinkan pengguna untuk mengumpulkan data berdasarkan kata kunci, rentang tanggal, dan bahasa tertentu. Data yang berhasil dikumpulkan akan disimpan dalam format file csv.

Tweet harvest bekerja dengan memanfaatkan *authorization token*. Token ini menyimpan informasi pribadi dari akun pengguna X dan memberikan izin akses untuk mengumpulkan data yang dibutuhkan. *Authorization token* ini didapatkan melalui halaman inspect yang dapat dibuka dengan melakukan klik kanan pada halaman X dan memilih opsi *Inspect*. Setelah halaman *inspect* terbuka, pilih halaman *Application*. Pada sisi kiri halaman ini, terdapat bagian *Storage* dan pilih pada bagian *Cookies* kemudian terdapat https://x.com. Pada bagian *Cookies X* ini terdapat *auth_token* yang memiliki nilai di dalamnya. Nilai pada *auth_token* tersebut dibutuhkan untuk dapat menggunakan *tool tweet harvest*.

Pengumpulan data dilanjutkan dengan memasukkan *auth_token* ke dalam kode. Untuk dapat menggunakan *tweet harvest*, diperlukan instalasi *NodeJS, Pandas,* dan *library tweet harvest*. Setelah instalasi berhasil dilakukan, pengumpulan data dapat dilakukan dengan menjalankan kode yang berisi kata kunci, rentang waktu, dan bahasa.

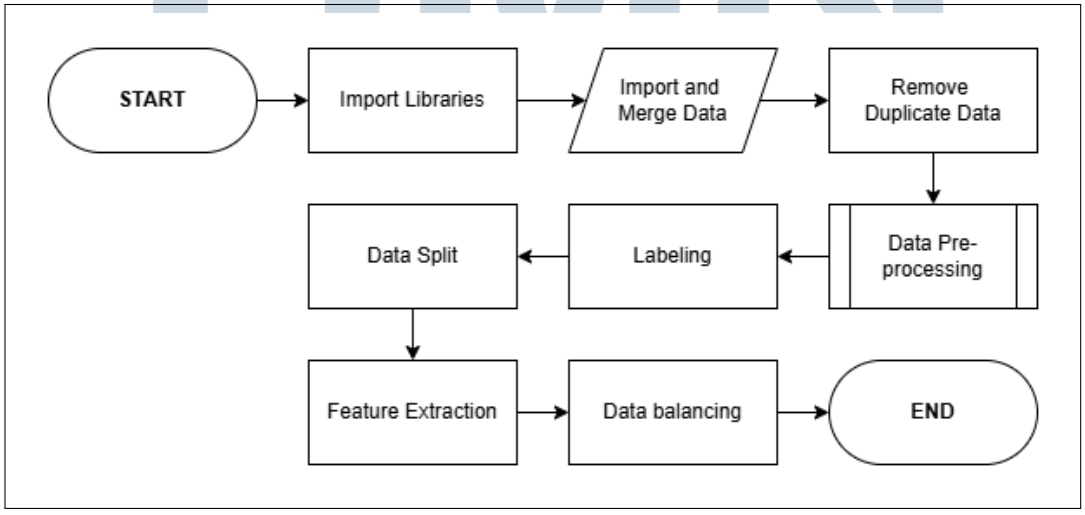
Untuk penelitian ini, data yang dikumpulkan merupakan *tweet* Bahasa Indonesia dalam rentang waktu 23 Juli 2023 sampai dengan 23 Juli 2024. Data yang dikumpulkan juga terdiri dari beberapa kata kunci yang berkaitan dengan *rebranding*, perubahan logo, perubahan nama dari Twitter menjadi X. Kata kunci yang digunakan dalam pengumpulan data ini terdiri dari ”twitter rebranding”, ”twitter berubah jadi x”, ”twitter ganti logo”, ”twitter ganti nama”, ”twitter jadi x”, ”logo twitter berubah”. Tahapan yang perlu dilakukan untuk mengumpulkan data ditunjukkan pada Gambar 3.2.



Gambar 3.2. *Flowchart* pengumpulan data

3.3 Perancangan Sistem

Setelah data berhasil dikumpulkan, tahap selanjutnya yang perlu dilakukan adalah perancangan sistem. Perancangan sistem dilakukan untuk membangun sistem yang dapat mengolah data mentah. Pengolahan data mentah ini perlu dilakukan supaya dapat menjadi *input* untuk model *machine learning*. Tahap ini juga terdapat proses *data pre-processing* yang merupakan proses membersihkan data supaya menghapus *noise* dan meningkatkan akurasi dari analisis yang dilakukan. Tahapan-tahapan dalam perancangan sistem pada penelitian ini dapat dilihat pada Gambar 3.3.



Gambar 3.3. *Flowchart* perancangan sistem

Perancangan sistem dilakukan melalui beberapa proses utama. Proses ini diperlukan untuk membangun sistem yang dapat mengolah data hingga siap menjadi *input* bagi model *machine learning*. Setiap proses dalam perancangan sistem dijelaskan sebagai berikut:

3.3.1 Import Libraries

Tahap pertama dalam perancangan sistem adalah *import libraries*. Langkah ini meliputi instalasi dan pemanggilan *library* yang dibutuhkan. Penggunaan *library* bertujuan untuk mempermudah proses perancangan sistem yang dilakukan. *Library* juga digunakan untuk keperluan lainnya, seperti pengolahan data, pembentukan model, validasi, dan evaluasi.

3.3.2 Import and Merge Data

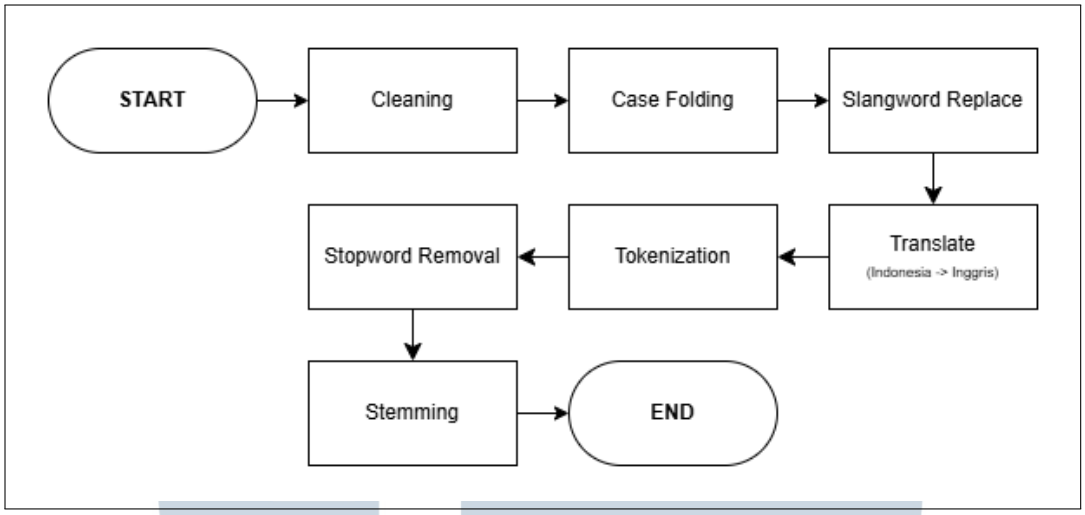
Tahap selanjutnya adalah *import and merge data* untuk melakukan impor dan penggabungan data. Tahap ini dilakukan dengan impor pada data yang telah dikumpulkan dan disimpan dalam format csv. Data yang dikumpul pada satu atau lebih dari file .csv akan digabungkan menjadi satu file untuk memudahkan pemrosesan data.

3.3.3 Remove Duplicate Data

Setelah data berhasil digabungkan, data akan melalui proses penghapusan data duplikat. Penghapusan data duplikat ini dilakukan karena data mentah yang dikumpul dan digabung memiliki kemungkinan terdapat data yang sama atau berulang. Penghapusan data duplikat dapat mengurangi potensi *bias* pada suatu kelas.

3.3.4 Data Pre-processing

Data yang telah berhasil digabung dan dihapus duplikatnya akan dilanjutkan ke tahap *data pre-processing*. Tahap ini bertujuan untuk meningkatkan akurasi model dengan membersihkan data dari *noise* atau informasi yang tidak relevan. Proses *data pre-processing* dilakukan agar data menjadi lebih bersih dan siap digunakan sebagai *input* untuk model. Tahapan-tahapan dalam *data pre-processing* ditunjukkan pada Gambar 3.4.



Gambar 3.4. *Flowchart* perancangan sistem

3.3.5 Labeling

Setelah melalui tahap data pre-processing, data yang telah bersih akan diberi label sentimen secara otomatis. Proses pelabelan otomatis ini dilakukan dengan memanfaatkan *library* TextBlob. TextBlob merupakan salah satu metode pelabelan yang menggunakan pendekatan *lexicon-based*. Setiap kata dalam kamus lexicon memiliki nilai polaritas yang telah ditentukan sebelumnya. Nilai polaritas merupakan angka dalam rentang -1 hingga 1 yang menentukan label pada data. Polaritas negatif menunjukkan sentimen negatif, polaritas nol menunjukkan sentimen netral, dan polaritas positif menunjukkan sentimen positif.

3.3.6 Data Split

Data bersih yang telah diberi label akan dilanjutkan ke tahap *data split*. Tahap ini merupakan pembagian dataset menjadi dua bagian, yaitu data latih (*train*) dan data uji (*test*). Pembagian ini dilakukan untuk memastikan bahwa model dilatih menggunakan data latih dan dievaluasi performanya dalam melakukan prediksi dengan data uji.

3.3.7 Feature Extraction

Data yang telah dibagi menjadi data latih dan data uji kemudian akan diubah menjadi representasi numerik melalui proses *feature extraction*. Metode *feature*

extraction yang digunakan dalam penelitian ini adalah CountVectorizer dan TF-IDF. Tahap ini penting dilakukan untuk memastikan data teks dapat diubah ke dalam bentuk yang dapat dipahami dan diproses oleh algoritma *machine learning* sebagai *input*.

3.3.8 Data Balancing

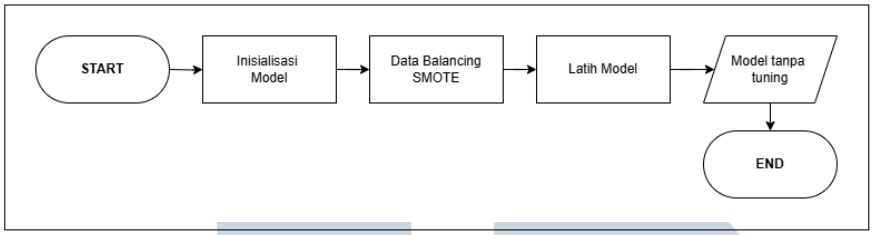
Data yang telah diubah ke dalam representasi numerik kemudian akan melewati proses *data balancing*. Proses ini dilakukan untuk menangani distribusi kelas yang tidak seimbang disaat salah satu kelas memiliki jumlah data yang jauh lebih banyak dibandingkan kelas lainnya. Distribusi kelas yang tidak seimbang ini dapat menyebabkan model menjadi bias ke kelas mayoritas sehingga menurunkan akurasi prediksi terhadap kelas minoritas. Sehingga dibutuhkan penyeimbangan data yang akan dilakukan dengan SMOTE melalui fungsi `SMOTE()` untuk membuat data buatan pada kelas minoritas.

3.4 Pembentukan Model

Pada tahap pembentukan model, algoritma Naïve Bayes dan SVM digunakan sebagai *base model* dalam penelitian ini. Setiap model akan dibentuk dengan menggunakan teknik *feature extraction* dan kondisi data yang berbeda. Skenario-skenario pembentukan model yang diterapkan dalam penelitian ini adalah sebagai berikut:

A Model tanpa Hyperparameter Tuning

Pembentukan model tanpa hyperparameter tuning dilakukan untuk mengetahui performa dasar dari masing-masing algoritma. Model akan dibentuk menggunakan dua teknik *feature extraction*, yaitu CountVectorizer dan TF-IDF. Selain itu, model juga akan dilatih dengan data yang tidak seimbang dan juga data yang telah diseimbangkan dengan SMOTE. Proses pembentukan model tanpa hyperparameter tuning dapat dilihat pada Gambar 3.5.

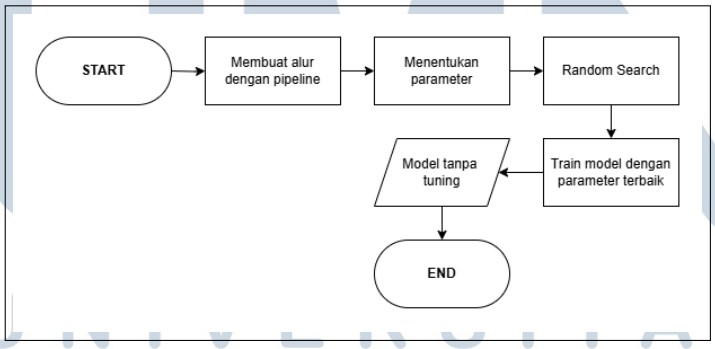


Gambar 3.5. *Flowchart* pembentukan model tanpa *hyperparameter tuning*

Pada pembentukan model akan dilakukan pemilihan *feature extraction* dan *data balancing*. Apabila *data balancing* diperlukan, maka SMOTE akan diterapkan pada data latih supaya data menjadi seimbang. Kemudian pemilihan implementasi algoritma Naïve Bayes atau SVM untuk lanjut ke pelatihan model. Pelatihan model ini menghasilkan model yang akan digunakan untuk melakukan prediksi pada data uji.

B Model dengan Hyperparameter Tuning

Pembentukan model dengan hyperparameter tuning dilakukan dengan mencari kombinasi parameter secara acak menggunakan Random Search. Sebelum membentuk model baru dengan parameter terbaik, terdapat hal-hal yang perlu dipersiapkan. Langkah-langkah pada pembentukan model dengan hyperparameter tuning ditunjukkan pada Gambar 3.6.



Gambar 3.6. *Flowchart* dengan *hyperparameter tuning*

Pembentukan model dengan *hyperparameter tuning* diawali dengan pembuatan alur pada *pipeline*. Di dalam pipeline ini akan didefinisikan alur pembentukan model, seperti *feature extraction, sampling*, dan algoritma yang digunakan. Setelah itu dilanjutkan dengan mendefinisikan parameter yang akan diuji dan dicari nilai terbaiknya.

Pencarian kombinasi parameter dilakukan pada parameter `CountVectorizer`, `TfidfVectorizer`, `MultinomialNB`, dan `SVC`. Setiap parameter memiliki nilai *default* yang dapat diketahui melalui pemanggilan fungsi default atau melalui dokumentasi resmi *sklearn*. Nilai parameter dari setiap fungsi ditunjukkan pada tabel-tabel berikut ini.

- **Parameter CountVectorizer**

  CountVectorizer merupakan metode ekstraksi fitur yang digunakan. Fungsi ini memiliki beberapa parameter untuk menentukan fitur yang diekstrak pada proses ini. Parameter dari fungsi ini dapat dilihat pada Tabel 3.1.

  Tabel 3.1. Parameter CountVectorizer()

Parameter	Nilai yang Diuji	Default
max_df	[0.25, 0.5, 0.75, 1.0]	1.0
max_features	[None, 1000, 2000, 5000]	None
min_df	[1, 3, 5, 10]	1
ngram_range	[(1,1), (1,2)]	(1,1)

  Terdapat beberapa parameter yang didefinikan untuk dicari nilai terbaiknya. Setiap parameter dari fungsi ini memiliki nilai dan kegunaannya masing-masing. Kegunaan dari setiap parameter pada fungsi `CountVectorizer` dijelaskan sebagai berikut:

  - `max_df = [0.25, 0.5, 0.75, 1.0]` Berfungsi untuk mengabaikan kata-kata yang muncul lebih dari persentase tertentu dari seluruh dokumen. Rentang nilai ini dipilih untuk melihat pengaruh kata-kata yang sering muncul terhadap kinerja model. Semakin kecil nilainya, semakin banyak kata umum yang dihapus.
  - `max_features = [None, 1000, 2000, 5000]` Berfungsi untuk menentukan jumlah maksimum kata (*feature*) pada proses ekstraksi. Rentang nilai ini dipilih untuk mengendalikan jumlah fitur yang diambil serta menghindari resiko *overfitting* yang dapat terjadi apabila terlalu banyak fitur yang dimasukkan dalam model.
  - `min_df = [1, 3, 5, 10]` Berfungsi untuk mengabaikan kata-kata yang jarang muncul dari persentasi tertentu. Rentang ini dipilih untuk

menguji seberapa besar pengaruh kata-kata yang unik dan langka pada model.

- `ngram_range = [(1,1), (1,2)]` Berfungsi untuk menentukan rentang n-gram yang diekstrak dari teks. Rentang (1,1) berarti *unigram* dan (1,2) berarti bigram. Penambahan ini ditambahkan untuk mengetahui apakah bigram dapat meningkatkan performa model.

- **Parameter TfidfVectorizer**

TfidfVectorizer merupakan metode ekstraksi fitur yang digunakan. Fungsi ini memiliki beberapa parameter untuk menentukan fitur yang diekstrak pada proses ini. Parameter dari fungsi ini dapat dilihat pada Tabel 3.2.

Tabel 3.2. Parameter TfidfVectorizer()

Parameter	**Nilai yang Diuji**	**Nilai *Default***
max_df	[0.25, 0.5, 0.75, 1.0]	1.0
max_features	[None, 1000, 2000, 5000]	None
min_df	[1, 3, 5, 10]	1
ngram_range	[(1,1), (1,2)]	(1,1)
smooth_idf	[True, False]	True
use_idf	[True, False]	True

Terdapat beberapa parameter yang didefinikan untuk dicari nilai terbaiknya. Setiap parameter dari fungsi ini memiliki nilai dan kegunaannya masing-masing. Kegunaan dari setiap parameter pada fungsi `TfidfVectorizer` dijelaskan sebagai berikut:

- `max_df = [0.25, 0.5, 0.75, 1.0]` Berfungsi untuk mengabaikan kata-kata yang muncul lebih dari persentase tertentu dari seluruh dokumen. Rentang nilai ini dipilih untuk melihat pengaruh kata-kata yang sering muncul terhadap kinerja model. Semakin kecil nilainya, semakin banyak kata umum yang dihapus.
- `max_features = [None, 1000, 2000, 5000]` Berfungsi untuk menentukan jumlah maksimum kata (*feature*) pada proses ekstraksi. Rentang nilai ini dipilih untuk mengendalikan jumlah fitur yang diambil serta menghindari resiko *overfitting* yang dapat terjadi apabila terlalu banyak fitur yang dimasukkan dalam model.

- `min_df = [1, 3, 5, 10]` Berfungsi untuk mengabaikan kata-kata yang jarang muncul dari persentasi tertentu. Rentang ini dipilih untuk menguji seberapa besar pengaruh kata-kata yang unik dan langka pada model.
- `ngram_range = [(1,1), (1,2)]` Berfungsi untuk menentukan rentang n-gram yang diekstrak dari teks. Rentang (1,1) berarti *unigram* dan (1,2) berarti bigram. Penambahan ini ditambahkan untuk mengetahui apakah bigram dapat meningkatkan performa model.
- `smooth_idf = [True, False]` Berfungsi untuk menambah 1 pada perhitungan IDF untuk menghindari pembagian dengan nol. Nilai ini dipilih untuk melihat pengaruh *smoothing* terhadap bobot dari fitur yang muncul sangat sedikit.
- `use_idf = [True, False]` Berfungsi untuk menentukan penggunaan bobot IDF pada proses perhitungan. Nilai ini dipilih untuk melihat pengaruh model pada penggunaan bobot IDF.

- **Parameter MultinomialNB**

MultinomialNB merupakan fungsi yang digunakan untuk pembentukan model menggunakan algoritma Naive Bayes dengan model Multinomial. Fungsi ini memiliki parameter alpha yang berfungsi untuk menentukan *smoothing* dengan menambahkan nilai tertentu. Nilai awal dan nilai yang akan diuji dari parameter ini ditunjukkan pada Tabel 3.3.

Tabel 3.3. Parameter MultinomialNB()

Parameter	**Nilai yang Diuji**	**Nilai *Default***
alpha	[0.1, 0.325, 0.55, 0.775, 1.0]	1.0

Terdapat beberapa alpha yang didefinikan untuk dicari nilai terbaiknya. Parameter dari fungsi ini memiliki nilai dan kegunaannya tersendiri. Kegunaan dari parameter pada fungsi `MultinomialNB` dijelaskan sebagai berikut:

- `alpha = [0.1, 0.325, 0.55, 0.775, 1.0]` Parameter ini berfungsi untuk mengatur tingkat *smoothing* pada model Naive Bayes yang dibentuk. *Smoothing* merupakan proses menambahkan

nilai tertentu pada perhitungan probabilitas untuk menghindari nilai nol. Rentang nilai ini dipilih untuk melihat nilai terbaik untuk melakukan *smoothing*.

- **Parameter SVC**

  Fungsi SVC merupakan fungsi yang digunakan untuk membentuk model menggunakan algoritma SVM. Pada fungsi ini terdapat parameter yang dapat dilakukan *hyperparameter tuning*. Parameter pada fungsi SVC yang dilakukan tuning ditunjukkan pada Tabel 3.4 berikut.

  Tabel 3.4. Parameter SVC()

Parameter	**Nilai yang Diuji**	**Nilai *Default***
C	[0.1, 1, 10, 100, 1000]	1.0
decision_function_shape	['ovo', 'ovr']	ovr

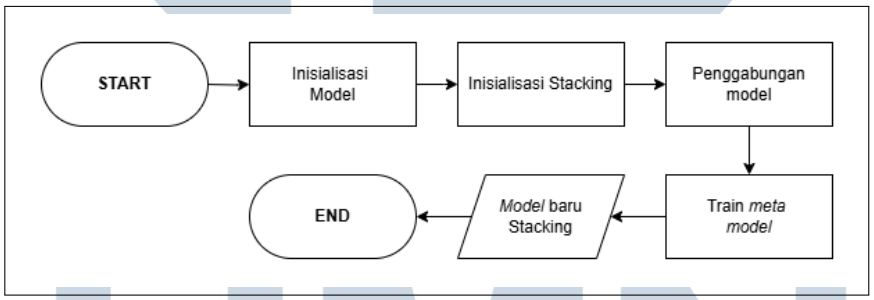
  Terdapat beberapa parameter yang didefinikan untuk dicari nilai terbaiknya. Setiap parameter dari fungsi ini memiliki nilai dan kegunaannya masing-masing. Kegunaan dari setiap parameter pada fungsi `SVC` dijelaskan sebagai berikut:

  - `C = [0.1, 1, 10, 100, 1000]` Berfungsi untuk mengatur seberapa besar model mengizinkan melakukan kesalahan dalam klasifikasi data latih. Nilai C dapat mengatur lebar dari margin dan seberapa banyak kesalahan klasifikasi diizinkan. Semakin kecil nilai C, maka semakin lebar margin dan model mengizinkan lebih banyak kesalahan pada data pelatihan. Sebaliknya, nilai C yang besar akan memperkecil margin dan memaksa model untuk melakukan semua klasifikasi data latih dengan benar yaang dapat memicu *overfitting*. Rentang nilai ini dipilih untuk mengetahui tingkat toleransi terhadap kesalahan dari model yang dihasilkan.
  - `decision_function_shape = ['ovo', 'ovr']` Berfungsi untuk menentukan jenis klasifikasi *multi-class* yang digunakan adalah *one-vs-one* (OVO) atau *one-vs-rest* (OVR). Secara default parameter ini memiliki nilai 'ovr' yang berarti jenis yang digunakan adalah *one-vs-rest*. Nilai pada parameter ini dipilih untuk menguji jenis klasifikasi *multi-class* yang paling optimal.

Pipeline dan parameter yang telah ditentukan akan menjadi input untuk melakukan Random Search menggunakan fungsi `RandomizedSearchCV()`. Proses random search ini menghasilkan model beserta kombinasi parameter dengan nilai akurasi tertinggi. Model tersebut nantinya dapat digunakan untuk melakukan prediksi terhadap data uji.

C Model dengan Ensemble Learning: Stacking

Pembentukan model dengan ensemble learning dilakukan dengan menggabungkan beberapa algoritma *machine learning* sebagai *base learner*. Penggabungan *base learner* ini merupakan model yang belum dilatih sebelumnya. *Base learner* ini nantinya akan dilatih bersama dengan *meta learner* sehingga terbentuk model akhir dan dapat digunakan untuk melakukan prediksi akhir. Proses pembentukan model dengan *stacking ensemble learning* ditunjukkan pada Gambar 3.7.



Gambar 3.7. *Flowchart* pengujian dengan *ensemble learning*

Pembentukan model untuk melakukan *stacking* dengan *ensemble learning* diawali dengan melakukan inisialisasi algoritma yang akan menjadi *base learner*. Setelah itu *base learner* yang telah diinisialisasi akan didefinisikan kembali pada fungsi `StackingClassifier()`. Di dalam fungsi ini juga terdapat algoritma lain yang berperan sebagai *meta learner* yang bertugas untuk melakukan prediksi akhir. Setelah itu model akhir akan terbentuk dan dapat digunakan untuk melakukan prediksi.

3.5 Pengujian

Pada tahap ini, model yang telah dibentuk dari berbagai metode *feature extraction* dan algoritma akan dilakukan pengujian. Pengujian yang dilakukan

adalah melakukan prediksi pada data uji yang telah dipisahkan dengan rasio sebesar 10% dari keseluruhan data.

3.6 Validasi dan Evaluasi

Validasi dan Evaluasi merupakan tahap yang dilakukan setelah pengujian dari berbagai skenario. Hasil pengujian akan divalidasi kembali menggunakan *cross-validation*. Metode *cross-validation* yang digunakan pada penelitian ini adalah *Stratified K-fold Cross Validatation* dengan 5 *fold* (lipatan).

Setelah proses validasi dilakukan, hasil dari model akan dievaluasi menggunakan *confusion matrix*. Melalui *confusion matrix* ini, diperoleh nilai evaluasi penting, seperti *accuracy, precision, recall, dan F1-score* yang berguna untuk menilai kinerja model.

3.7 Dokumentasi

Tahap terakhir adalah dokumentasi yang bertujuan untuk mencatat dan menyusun seluruh proses serta hasil yang telah dicapai selama penelitian. Dokumentasi ini dilakukan untuk memastikan setiap tahapan dari awal hingga akhir penelitian terdokumentasi dengan baik. Seluruh informasi penting pada penelitian ini akan disusun secara sistematis ke dalam laporan penelitian dalam bentuk skripsi.

3.8 Spesifikasi Sistem

Penelitian ini didukung oleh spesifikasi perangkat keras *(hardware)* dan perangkat lunak *(software)*. Perangkat keras berperan dalam menjalankan proses komputasi, sedangkan perangkat lunak digunakan untuk pemrograman, serta penyusunan laporan skripsi. Perangkat keras dan lunak yang digunakan dalam penelitian ini dirincikan sebagai berikut:

3.8.1 Hardware

1. Sistem Operasi: Windows 10 64-bit
2. RAM: 16 GB
3. Processor: 12th Gen Intel(R) Core(TM) i5-12450HX 2.40 GHz

3.8.2 Software

1. Jupyter Notebook
2. Anaconda
3. Google Chrome
4. draw.io