BAB 3

METODOLOGI PENELITIAN

3.1 Metodologi Penelitian

Metodologi yang diterapkan dalam penelitian ini terdiri dari beberapa tahap, yakni sebagai berikut:

1. Telaah literatur

   Telaah literatur dilakukan untuk mempelajari dan memperdalam pengetahuan serta teori terkait metode penanggulangan dataset tidak seimbang, representasi teks, *pre-trained* ELMo, algoritma SMOTE, algoritma UMAP, model MLP, algoritma PCA, metode Grid Search, dan metrik evaluasi. Tahap ini dilakukan dengan melakukan studi dari berbagai referensi, seperti jurnal dan buku terkait.

2. Analisis kebutuhan

   Analisis kebutuhan dilakukan untuk mengetahui perangkat lunak dan perangkat keras yang dibutuhkan dalam melaksanakan penelitian.

3. Pengumpulan dan analisa data

   Data yang digunakan untuk melakukan penelitian diperoleh dari Kaggle. Setelah memperoleh data, dilakukan analisa data untuk mengetahui informasi dan fitur-fitur yang dapat digunakan dalam penelitian. Kemudian data diproses sehingga dapat diubah menjadi vektor menggunakan *pre-trained* ELMo.

4. Perancangan dan pembuatan sistem

   Perancangan dilakukan agar rancangan aplikasi dapat dibuat sesuai dengan kebutuhan dan memiliki proses yang terstruktur. Tahapan ini mencakup

pembuatan *flowchart* dari penelitian yang dilakukan, dilanjutkan dengan pembuatan sistem berdasarkan hasil perancangan tersebut. Pembuatan mencakup penulisan kode dan visualisasi dataset menggunakan bahasa pemrograman Python, *training data set*, dan modifikasi model.

5. Pengujian dan evaluasi

   Pengujian dilakukan untuk memastikan sistem berjalan dengan baik dan memiliki fungsionalitas yang sesuai dengan tujuan awal. Evaluasi dilakukan untuk menguji dan melakukan validasi terhadap hasil klasifikasi dari sistem. Nilai *precision*, *recall,* dan *f-measure* dihitung dan dievaluasi untuk membuat kesimpulan atas rumusan masalah yang telah dijabarkan sebelumnya.

6. Penulisan naskah penelitian dan konsultasi

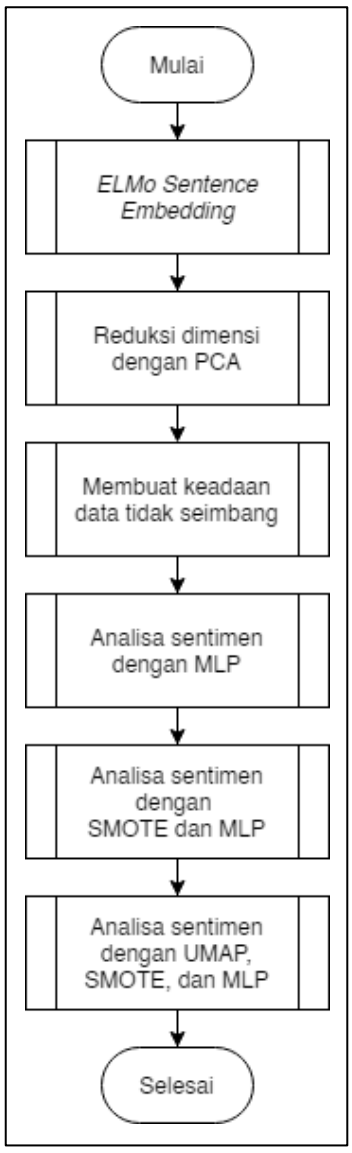
   Penulisan naskah dilakukan sebagai bentuk dokumentasi sehingga dapat dijadikan sebagai referensi dan sarana ilmu pengetahuan untuk penelitian selanjutnya. Konsultasi dilakukan agar penelitian tetap terarah dan memperoleh saran sehingga dapat berjalan dengan baik.

3.2 Perancangan Aplikasi

Perancangan aplikasi direpresentasikan menggunakan *flowchart*. Terdapat *flowchart* utama yang menjelaskan alur kerja sistem secara keseluruhan, kemudian untuk setiap modul dalam *flowchart* utama dijelaskan kembali secara lebih mendetail pada bagian selanjutnya.

3.2.1 Flowchart Utama

Alur sistem utama dapat dilihat pada Gambar 3.1. Seluruh proses wajib dijalani dalam melakukan percobaan, kecuali reduksi dimensi dengan PCA. Reduksi dimensi hanya dilakukan karena adanya keterbatasan sumber daya yang tersedia ketika menggunakan Google Colaboratory sehingga percobaan yang dilakukan tidak dapat dijalankan menggunakan data dengan jumlah dimensi yang sebenarnya, yaitu 1024.



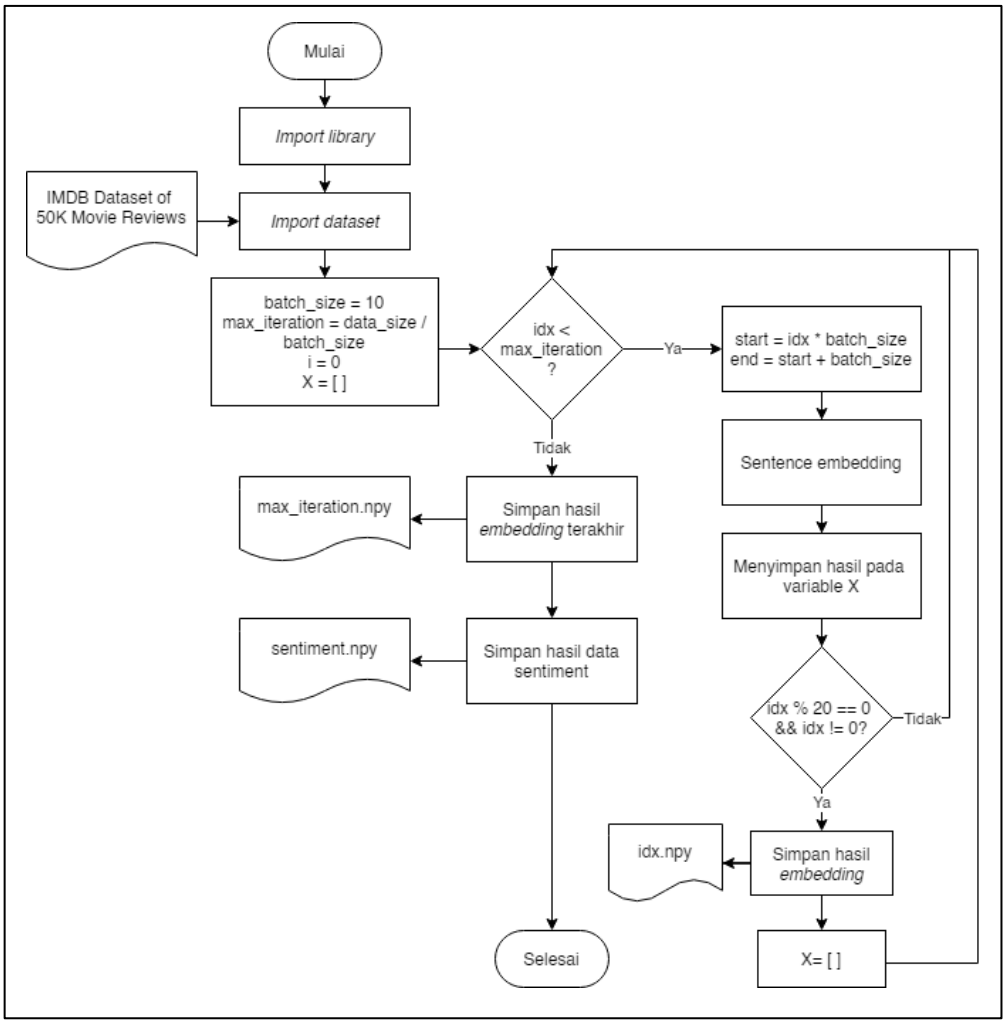
Gambar 3.1 *Flowchart* Utama

Sistem diawali dengan melakukan *sentence embedding* menggunakan ELMo, kemudian hasil *embedding* tersebut digunakan sebagai dataset. Akibat dataset yang digunakan merupakan dataset seimbang, maka perlu dilakukan proses pembagian data *training* menjadi tidak seimbang. Selanjutnya data yang tidak seimbang tersebut diklasifikasi sebanyak 3 kali menggunakan MLP. Pertama, dataset langsung diklasifikasi menggunakan MLP. Kedua, klasifikasi dengan MLP diawali dengan *resampling* menggunakan SMOTE. Kemudian yang terakhir, melakukan reduksi dimensi menggunakan UMAP terlebih dahulu, dilanjutkan dengan *resampling* SMOTE, *invert* dimensi reduksi menggunakan UMAP, lalu melakukan klasifikasi menggunakan MLP.

3.2.2 Flowchart Sentence Embedding

Sentence embedding dilakukan untuk mengubah dataset yang berbentuk kalimat menjadi vektor sehingga dapat dimengerti oleh mesin. Proses ini diawali dengan melakukan *import library* yang diperlukan, salah satunya ELMo versi kedua dari TensorFlow dan *import* dataset IMDB Dataset of 50k Movie Review dari Kaggle.

Proses *sentence embedding* dilakukan dalam *batch* yang masing-masing terdiri dari 10 data. Proses *batch* bertujuan mengurangi sumber daya yang diperlukan, karena jika dilakukan secara langsung untuk seluruh data yang berjumlah 50.000, maka sumber daya yang diperlukan sangat besar. Melalui proses *batch* ini, *embedding* dapat dilanjutkan bahkan ketika sumber daya yang digunakan kehabisan memori.



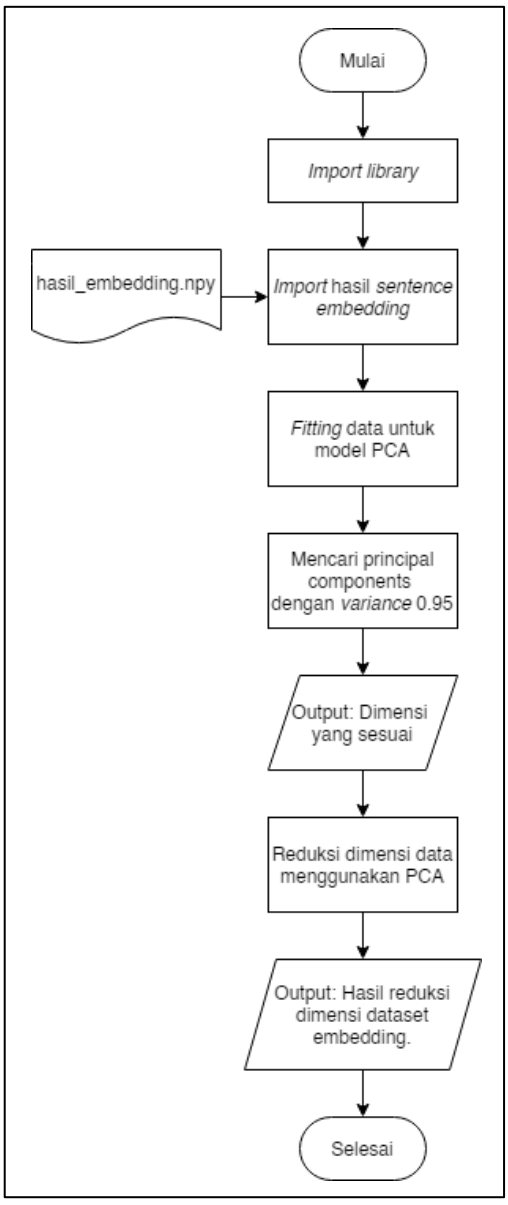
Gambar 3.2 *Flowchart Sentence Embedding*

Selain itu *batch* yang dilakukan dalam *loop* juga berperan dalam menyimpan hasil *embedding* dengan *format* npy dan nama file sesuai dengan iterasi, untuk setiap 200 data yang sudah berhasil di-*embedding*. Jumlah data yang disimpan dapat disesuaikan dengan sumber daya yang tersedia. Semakin besar sumber daya, maka jumlah *batch* dan penyimpanan hasil *embedding* dapat dilakukan dalam frekuensi yang lebih rendah.

Selanjutnya ketika iterasi sudah selesai, dilakukan penyimpanan terakhir dengan nama file iterasi maksimum dalam format .npy. Hal ini dikarenakan *loop* yang sebelumnya dijalankan tidak akan menyimpan hasil *embedding* untuk 20 data terakhir, sehingga harus dilakukan secara manual untuk kali terakhir.

3.2.3 Flowchart Reduksi Dimensi dengan PCA

Berdasarkan penjelasan sebelumnya, proses ini hanya dilakukan karena adanya keterbatasan sumber daya yang tersedia sehingga tidak dapat melakukan percobaan menggunakan dimensi asli dari data hasil *sentence embedding*, yaitu 1024. Berikut merupakan alur proses reduksi dimensi yang dilakukan.



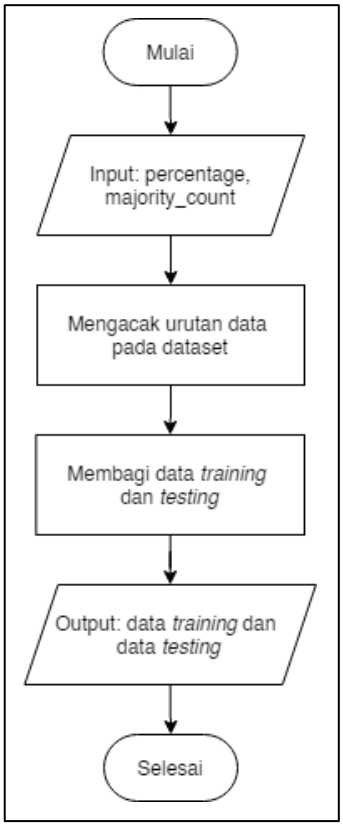
Gambar 3.3 *Flowchart* Reduksi Dimensi dengan PCA

Setelah melakukan *import library* dan hasil *sentence embedding*, data yang diperoleh akan di-*fitting* terhadap PCA yang disediakan oleh *library* sklearn.

Kemudian menampilkan nilai *explained variance* untuk setiap jumlah komponen dari PCA melalui grafik. Nilai *variance* menunjukkan tingkat kegunaan komponen dalam dimensi terkait, sehingga semakin tinggi nilai *explained variance*, semakin baik. Biasanya nilai *explained variance* minimal yang dipilih adalah 80% atau 0.8, namun dalam penelitian ini nilai *variance* yang dipilih adalah 0.95 atau 95%. Setelah jumlah dimensi yang sesuai ditemukan, dilakukan reduksi dimensi untuk seluruh dataset yang diperoleh.

3.2.4 Flowchart Dataset Tidak Seimbang

Dataset yang digunakan terdiri atas 25.000 ulasan bersentimen positif dan 25.000 ulasan bersentimen negatif, sehingga dataset ini merupakan dataset yang sangat seimbang. Untuk menyebabkan adanya ketidakseimbangan dalam dataset yang digunakan untuk *training*, maka dilakukan beberapa proses sebagai berikut.



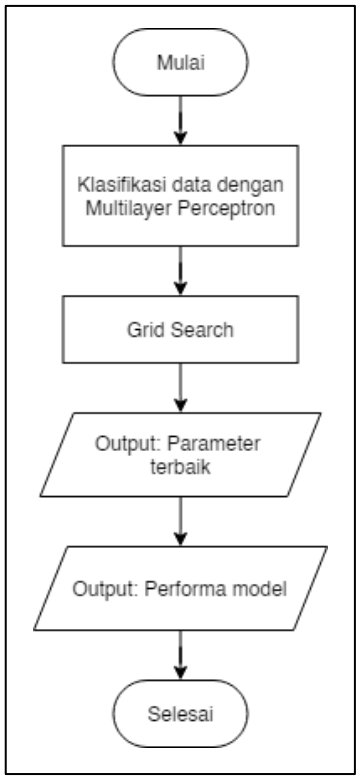
Gambar 3.4 *Flowchart* Dataset Tidak Seimbang

Proses ini dilakukan dalam fungsi yang menerima 2 parameter, yakni ‘percentage’ yang merupakan persentase perbandingan data tidak seimbang dan ‘majority_count’ yang merupakan jumlah data kelas mayor. Dalam percobaan ini, data tidak seimbang hanya diterapkan pada data *training* dengan data kelas mayor adalah data bersentimen positif sementara data kelas minor adalah data bersentimen negatif. Sebaliknya, data *testing* terdiri dari 10.000 data yang terbagi secara merata.

Sebelum melakukan proses pembagian data, dataset diacak terlebih dahulu untuk memastikan data yang dihasilkan dari fungsi ini memiliki hasil yang selalu berbeda. Kemudian *output* dari fungsi ini adalah data *training* sesuai dengan parameter *input* dan data *testing* sejumlah 10.000 data dalam bentuk array yang telah dispesifikasikan untuk *variable* X_train, X_test, y_train, dan y_test secara berurutan.

3.2.5 Flowchart Analisa Sentimen dengan MLP

Setelah membagi data menjadi data *training* dan data *testing*, dilakukan klasifikasi menggunakan Multilayer Perceptron. Klasifikasi ini dioptimasi dengan mencari parameter terbaik menggunakan metode Grid Search yang disediakan oleh sklearn. Setelah proses Grid Search selesai, sistem menampilkan parameter terbaik disertai dengan performa model, yakni nilai *precision*, *recall*, dan *f-measure* dari data yang telah diberikan.



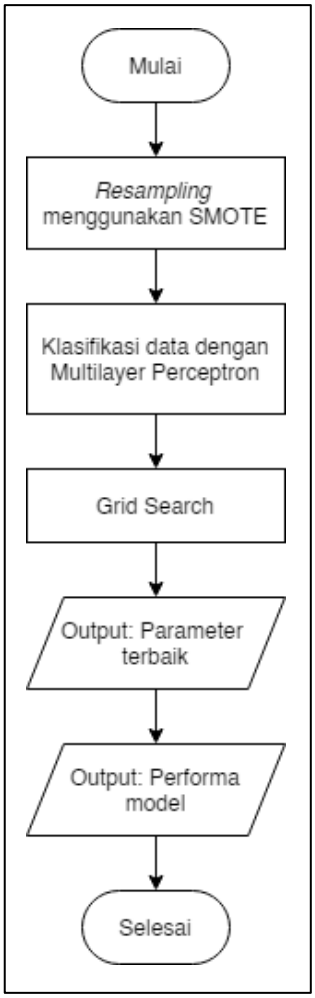
Gambar 3.5 *Flowchart* Analisa Sentimen dengan MLP

3.2.6 Flowchart Analisa Sentimen dengan SMOTE dan MLP

Analisa sentimen menggunakan SMOTE dan MLP dilakukan menggunakan data *training* dan data *testing* yang sama dengan data yang digunakan untuk melakukan klasifikasi dengan MLP saja. Sebagian besar alur proses yang dijalankan-pun serupa. Perbedaannya hanya terletak pada proses *resampling* menggunakan SMOTE yang dilakukan sebelum data diklasifikasi menggunakan MLP.

Kemudian dalam proses ini menggunakan *pipeline* untuk mencari parameter terbaik melalui Grid Search. Pipeline berguna untuk membuat SMOTE dan MLP menjadi satu kesatuan sehingga pencarian parameter terbaik dari SMOTE dan MLP dapat dilakukan. Jika dilakukan pencarian tanpa menggunakan *pipeline*, maka

parameter terbaik yang ditemukan melalui Grid Search hanya mencakup parameter dari MLP saja, tidak termasuk parameter dari SMOTE.



Gambar 3.6 *Flowchart* Analisa Sentimen dengan SMOTE dan MLP

3.2.7 Flowchart Analisa Sentimen dengan UMAP, SMOTE, dan MLP

Analisa sentimen dalam proses ini juga tidak berbeda jauh dengan analisa sentimen yang dilakukan menggunakan MLP saja. Perbedaannya terdapat pada proses awal sebelum melakukan klasifikasi. Data *training* yang diperoleh sebelumnya direduksi menjadi 2 dimensi (atau disesuaikan dengan parameter n_components jika disertakan) menggunakan UMAP. Selanjutnya dilakukan *resampling* menggunakan SMOTE, sehingga data yang di-*generate* oleh SMOTE

merupakan data 2 dimensi. Kemudian hasil *resampling* di-*invert* kembali menggunakan UMAP sehingga dimensinya kembali ke dimensi semula, yaitu 1024 atau kurang dari 1024 jika dimensi dataset awal telah direduksi menggunakan PCA.

Sama seperti proses analisa sentimen dengan SMOTE dan MLP, proses ini juga menggunakan *pipeline* untuk mencari parameter terbaik melalui Grid Search, *Pipeline* digunakan untuk menggabungkan UMAP dan SMOTE dengan MLP sehingga parameter terbaik dari ketiga algoritma dapat ditemukan.

Gambar 3.7 *Flowchart* Analisa Sentimen dengan UMAP, SMOTE, dan MLP