BAB III

METODOLOGI PENELITIAN DAN PERANCANGAN SISTEM

3.1 Metodologi Penelitian

Penelitian "Analisis Sentimen Masyarakat terhadap E-commerce pada Media Sosial Menggunakan Multilayer Perceptron dengan Word Embedding" akan dilakukan dalam beberapa tahap, yaitu telaah literatur, analisis kebutuhan, pemrograman sistem, *testing* dan *debugging*, implementasi sistem, serta konsultasi dan penulisan laporan agar metodologi dan perancangan sistem dapat terpenuhi. Tahap-tahap yang dilaksanakan antara lain adalah sebagai berikut.

1. Studi literatur dilakukan dengan cara mempelajari teori-teori dengan mencari litelatur berupa prosiding ataupun jurnal ilmiah yang berhubungan dengan sentimen analisis, metode *Multilayer Perceptron*, dan *Word Embedding.*
2. Analisis Kebutuhan, pada tahap ini dilakukan untuk menganalisa dan merancang kebutuhan sistem. Hasil Analisa lalu akan diterjemahkan untuk melakukan penelitian.
3. Perancangan Sistem, tahap ini dilakukan berbagai perancangan yang dibutuhkan sistem, seperti *flowchart* ataupun yang lainnya jika dibutuhkan sistem.
4. Pemrograman Sistem, tahap ini merupakan kegiatan dilakukannya pembuatan coding, *training dataset*, dan modifikasi model dengan bahasa pemrograman Python dengan *integrated development and environment* (IDE) *Jupyter*

Notebook dan beberapa *library* untuk menunjang otomatisasi pembelajaran mesinnya.

5. *Testing* dan *debugging*, dilakukan bersamaan dengan pemrograman sistem dengan tujuan untuk mengetahui kekurangan dan melakukan perbaikan pada sistem yang dibuat serta melakukan pengukuran performa algoritma yang digunakan. Pengukuran performa akan menggunakan perhitungan seperti pada persamaan 2.12, 2.13, 2.14, dan 2.15 pada bab dua.
6. Konsultasi dan Penulisan Laporan, tahapan ini dilakukan secara bersamaan dengan proses penelitian agar terdokumentasi dengan baik dan tidak ada proses proses penelitian yang terlewatkan.

3.2 Teknik Pengumpulan Data

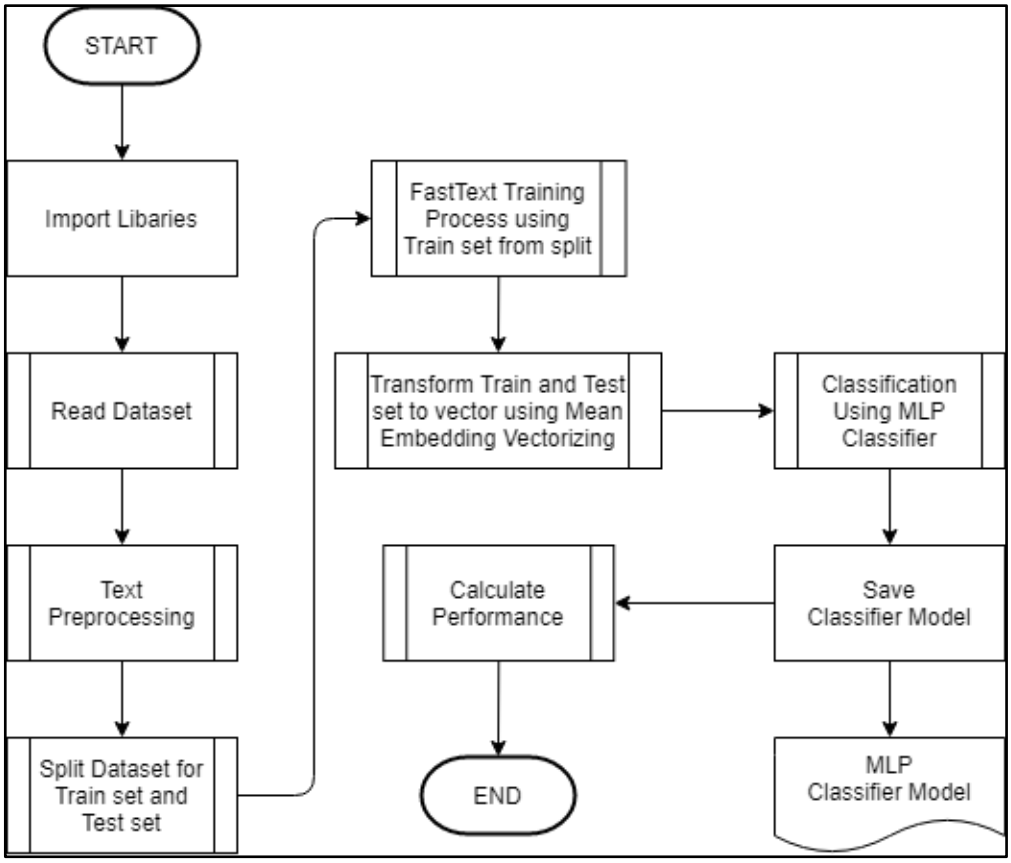
Teknik pengumpulan data yang dilakukan pada penelitian ini menggunakan dataset berbahasa Indonesia yang bersumber dari LIPI (Lembaga Ilmu Pengetahuan Indonesia) yang merupakan dataset hasil dari penelitian yang telah dilakukan oleh Rini Wijayanti dan Andria Arisal pada tahun 2017 yang berjudul *"Ensemble Approach for Sentiment Polarity Analysis in User-Generated Indonesian Text"* yang telah dipublikasikan di IEEE pada tahun 2018.

3.3 Perancangan Aplikasi

Perancangan sistem terdiri dari *flowchart*, perancangan struktur tabel, dan antarmuka aplikasi *web*. *Flowchart* akan terbagi menjadi 2 bagian, yaitu *flowchart* proses klasifikasi yang merupakan *flowchart* yang menggambarkan proses

klasifikasi menggunakan *Multilayer Percepcron* (MLP) *Classifier* dengan *Word Embedding*, dan *flowchart deployed version* yang menggambarkan proses penggunaan algoritma *Multilayer Perceptron* yang telah di-*deploy* ke dalam versi aplikasi *web.*

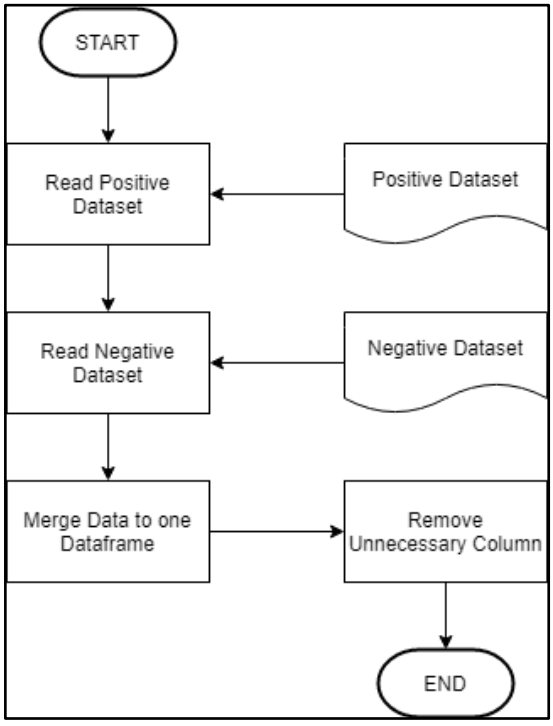
3.2.1 Flowchart Proses Klasifikasi



Gambar 3.1 *Flowchart* penelitian secara umum

Flowchart yang ditunjukkan oleh Gambar 3.1 merupakan *flowchart* dari proses klasifikasi menggunakan *Multilayer Perceptron* (MLP) *Classifier* dengan *Word Embedding* secara umum. Proses yang pertama dilakukan adalah meng-*import library* yang diperlukan untuk mendukung proses penelitian agar lebih efektif. Kemudian *dataset* yang akan digunakan akan dimasukkan dan dianalisa.

Dataset yang digunakan berasal dari LIPI (Lembaga Ilmu Pengetahuan Indonesia) yang sudah dijelasakan pada bagian Teknik Pengumpulan data. Kategori dari *dataset* yang didapat ternyata terpisah dan memiliki kolom-kolom yang dirasa tidak diperlukan untuk proses klasifikasi ini. Kolom yang digunakan pada dataset ini hanya dua kolom utama yaitu kolom komentar dan kolom kategori sentimennya (positif atau negatif). Karena itu dilakukan proses penyatuan *dataset* dan penghapusan kolom *dataset* yang tidak diperlukan untuk melakukan klasifikasi yang digambarkan pada *flowchart* pada Gambar 3.2.

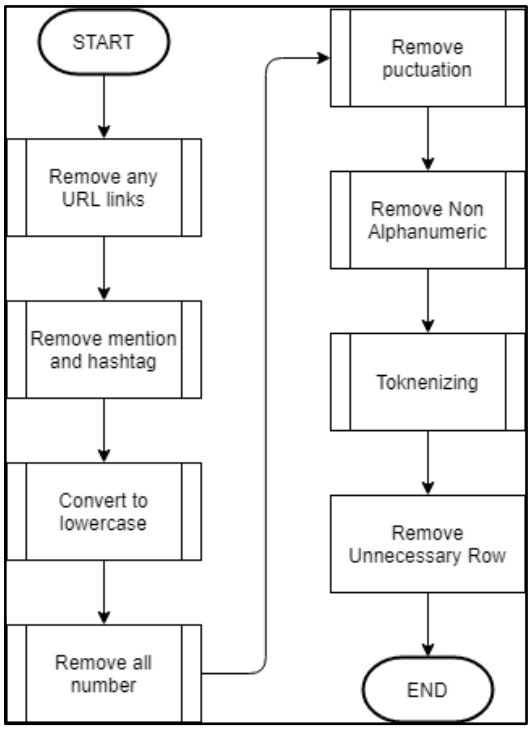


Gambar 3.2 *Flowchart* Proses *Read Dataset*

Dataset yang masuk akan dinormalisasikan pada tahap *text pre-processing*. Hal ini dilakukan agar data yang nantinya akan diolah sudah bersih dan sesuai dengan kebutuhan. Selain itu penelitian menjadi lebih efektif dan diharapkan performa yang lebih baik dan tepat sasaran. Tahapan yang dilakukan pada tahap *text pre-processing* antara lain:

1. *Remove any URL links* (menghapus seluruh link url yang ada pada dataset).
2. *Remove mention and hashtag* (menghapus semua *mention* dan *hastag* pada dataset).
3. *Convert to lower case* (membuat semua huruf pada dataset menjadi huruf kecil).
4. *remove all number* (menghapus semua angka).
5. *Remove punctuation* (menghapus seluruh tanda baca pada dataset).
6. *Remove non alphanumeric* (menghapus karakter selain alfanumerik).
7. *Tokenizing* (pemisahan kata dari kalimat).
8. *Remove unnecessary row* (menghapus baris dataset yang tidak diperlukan)

Tahapan *pre-processing* ini dapat dilihat di *flowchart* pada Gambar 3.3.



Gambar 3.3 *Flowchart Text Preprocessing*

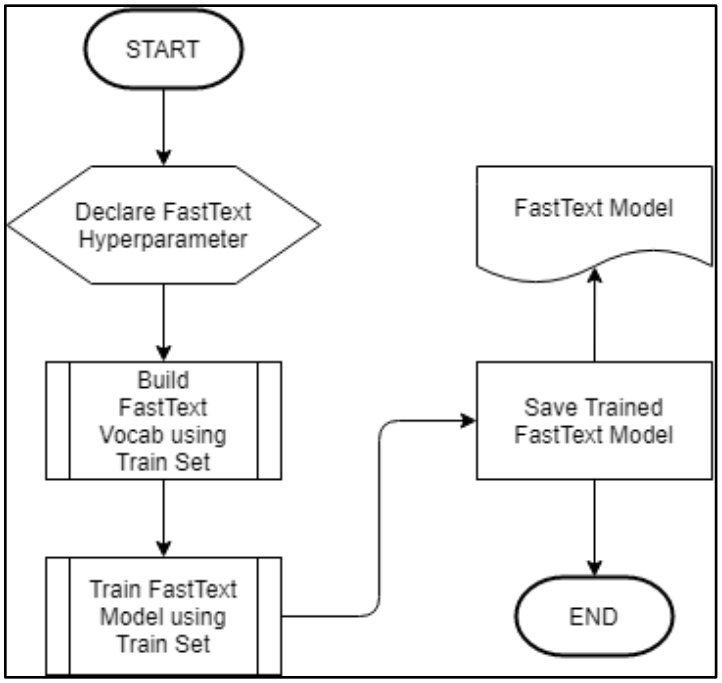
Tahapan *text pre-processing* dilakukan dengan substitusi dengan *pattern regex* (*Regular Expression*) *library*, *string* untuk *remove punctuation*, dan *nltk* (*Natural Language Toolkit*) untuk *tokenizing*. *Remove url link* dilakukan karena

tidak ada informasi yang dapat digunakan dalam *link website* untuk pembobotan nantinya. *Remove mention* dan *hashtag* dilakukan karena *dataset* yang digunakan merupakan gabungan antara *tweet* dan *app review*, sehingga pada *tweet* harus dihilangkan kata yang memiliki awalan *mention* (@) dan *hashtag* (#) karena tidak akan mempengaruhi. *Hashtag* mungkin akan berguna dalam pembobotan, namun ada *hastag* yang berisikan kata-kata yang digabung dan cukup panjang yang mengakibatkan makna kata yang rancu saat dilakukan pembobotan oleh karena itu diputuskan kata yang memiliki awalan *hashtag* (#) akan dihapus.

Remove number, *punctuation* (tanda baca), dan *non-alphanumeric* dilakukan karena angka atau nomor, tanda baca, dan karakter non-alfanumerik tidak memiliki makna apapun untuk pembobotan. Kemudian dilanjutkan dengan proses *tokenizing* dan terakhir *remove unnecessary rows* (menghapus baris *dataset* yang tidak diperlukan). Proses terakhir ini dilakukan karena setelah melewati *tokenizing* masih ada juga data yang kosong sehingga akan membuat error saat proses klasifikasi. *Stemming* dan *Stopword Removal* tidak dilakukan sebagaimana yang sudah dijelaskan pada latar belakang penelitian.

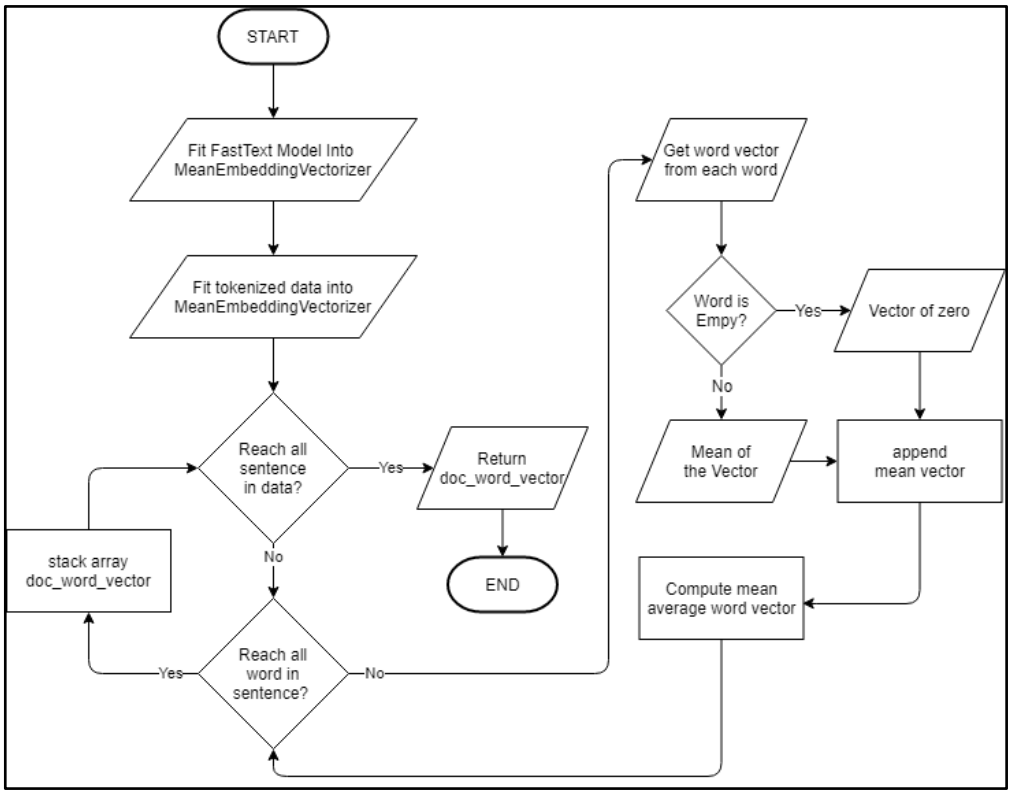
Setelah proses *text pre-processing*, kemudian dataset yang sudah diproses akan dibagi (*split*) menjadi *train set* dan *test set*. Tahapan *split* data ini dilakukan dengan bantuan *library* dari *Sklearn Train Test Split*. *Train set* adalah data yang digunakan untuk melatih model *machine learning* sebagai contoh agar model dapat bekerja sesua dengan parameter yang diberikan. *Test set* adalah data yang digunakan untuk memberikan evaluasi model berdasarkan pembelajaran yang sudah dilakukan dengan *train set*.

Setelah dataset di-*split*, proses selanjutnya adalah melakukan *training* model FastText *pre-trained* untuk melakukan *word embedding* seperti yang dapat dilihat pada Gambar 3.4. *Training* yang dilakukan hanya menggunakan data *train set* yang terbentuk dari proses *train test split*. Proses pertama dilakukan inisialisasi *hyperparameter* yang kemudian dilanjutkan dengan proses pembuatan *vocabulary* dan model *training* FastText. Kemudian hasil dari *training* model yang telah dilakukan dapat digunakan untuk proses vektorisasi dataset. Hasil model FastText *pre-train* juga akan di-*export* ke dalam bentuk .bin atau .vec yang dapat digunakan sebagai *feature extraction* untuk demo pada aplikasi web agar tidak diperlukan lagi *training* ulang model FastText data yang masuk karena hanya tinggal meng-*import* file model yang sudah dibuat. Proses pembuaatan model FastText *pre-trained* ini dibantu dengan *library Gensim*.

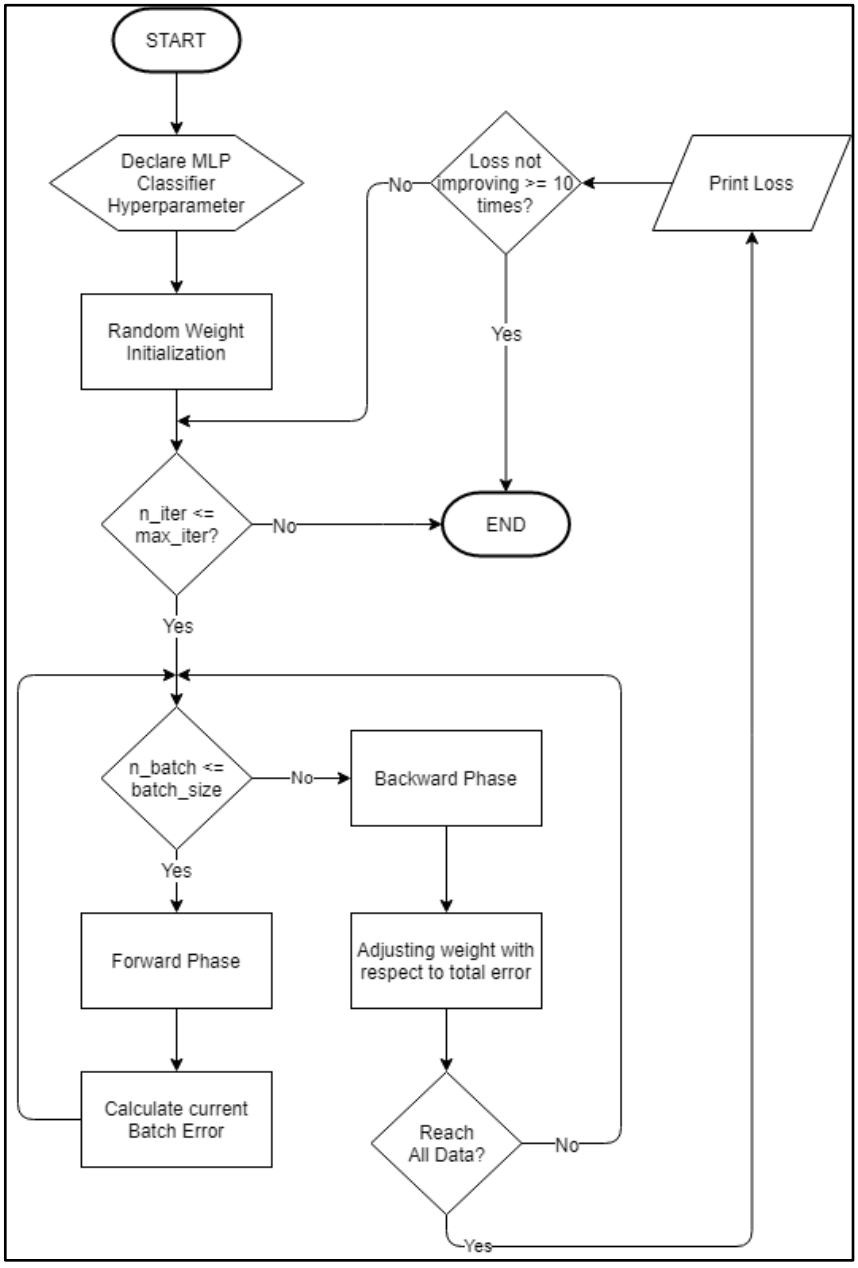


Gambar 3.4 *Flowchart* pembuatan model FastText *pre-trained*

Setelah dilakukan training model FastText maka langkah selanjutnya adalah proses vektorisasi *dataset* menggunakan *Mean Embedding Vectorizer*. Model FastText *pre-trained* kemudian di-*fit* ke dalam *Mean Embedding vectorizer* lalu setiap input kata yang masuk akan dihitung dan diambil nilai vektor rata-ratanya menggunakan model FastText *pre-trained* yang telah ditanamkan di dalam *Mean Embedding Vectorizer*. Jika input yang masuk adalah kata kosong (*empty word*) maka yang dikembalikan adalah nilai vektor 0. Namun, jika *input* merupakan sebuah kata maka output akan berupa nilai rata-rata dari vektor kata tersebut. Keluaran yang dihasilkan merupakan kumpulan vektor angka dari data yang masuk. *Flowchart* proses ini dapat dilihat pada Gambar 3.5.



Gambar 3.5 *Flowchart Mean Embedding Vectorizer*



Gambar 3.6 *Flowchart MLP Classifier*

Gambar 3.6 adalah *flowchart* yang menjabarkan proses lebih detail untuk klasifikasi menggunakan *MLP Classifier*. Proses pertama yaitu inisialisasi *hyperparameter* apa saja yang didigunakan untuk proses klasifikasi. Setelah itu bobot pada jaringan syaraf (*neural network*) akan diinisialisasi dengan bilangan kecil secara acak. Kemudian dilanjutkan dengan *Forward phase* dimana pada

langkah pelatihan model ini dilakukan untuk menghitung semua keluaran di unit tersembunyi dan juga menghitung semua keluaran jaringan di unit keluaran dengan hanya meneruskan input ke model dengan mengkalikannya dengan bobot pada setiap *layer* dan menemukan *output* yang dihitung model. Tahapan ini juga akan menghasilkan akumulasi eror berdasarkan *batch size* yang sudah ditentukan pada saat inisialisasi *hyperparameter*.

Setelah *batch size* yang sudah ditentukan terpenuhi tahap selanjutnya adalah *Backward phase* yaitu tahap penyesuaian atau perbaikan bobot koneksi berdasarkan hasil eror akumulatif yang didapat pada *Forward phase* dalam jaringan sehingga meminimalkan ukuran perbedaan antara vektor *output* aktual dari *net* dan vektor *output* yang diinginkan. Setelah *Backward phase* dilakukan maka kemudian akan kembali dilakukan *Forward Phase* untuk *batch* data berikutnya lalu *Backward Phase* lagi hingga seluruh data ter-*cover*. Seluruh proses sebelumnya akan dilakukan sebanyak jumlah *epoch* atau iterasi yang ditentukan pada saat inisialisasi *hyperparameter* untuk menemukan hasil *loss* hinggga mendekati 0. Jika *loss* dalam pengulangan *epoch* tidak ada perbaikan selama 10 iterasi berturut turut maka proses *training* akan dianggap selesai.

Proses pembelajaran *MLP Classifier* akan di-*export* (disimpan) menjadi sebuah file .pkl untuk nantinya dapat digunakan untuk demo pada aplikasi web agar tidak diperlukan lagi *training* data untuk melakukan prediksi karena hanya tinggal meng-*import* file model yang sudah dibuat. Proses *export* dan *import* model ini dibantu dengan *library pickle*.

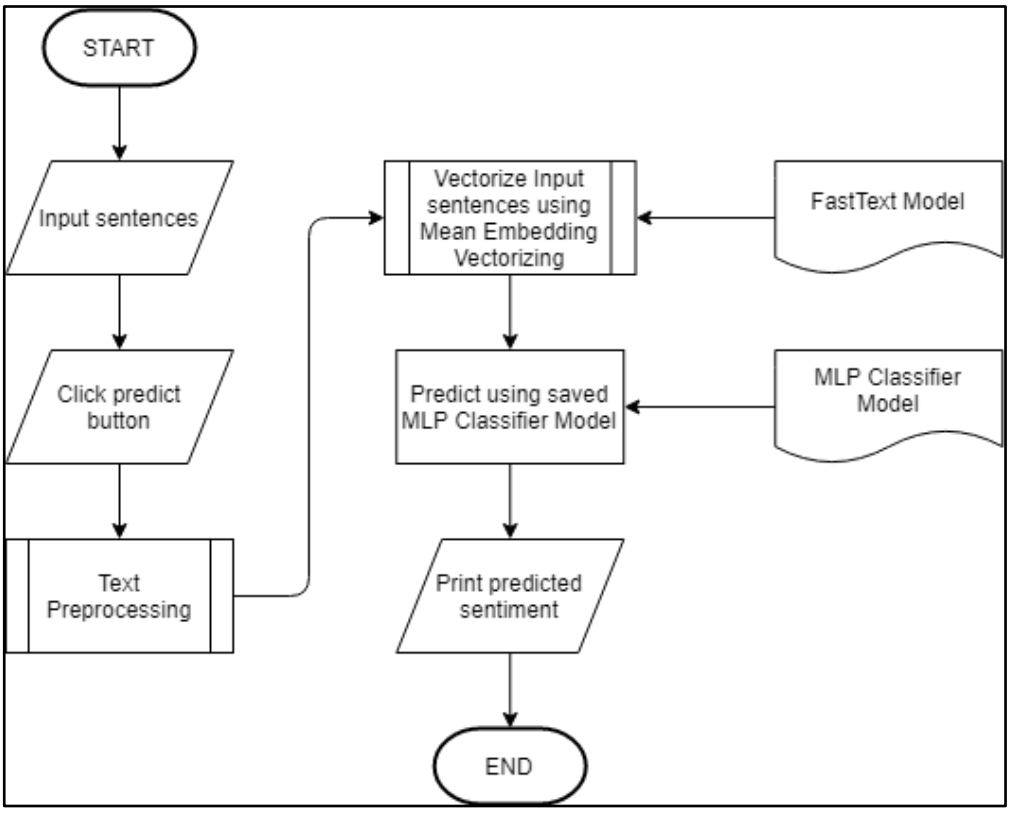
Proses berikutnya dilakukan evaluasi performa dengan menggunakan *confussion matrix*. Dengan *confussion matrix* akan diperoleh nilai *variable* untuk menghitung *accuracy*, *precision*, dan *recall* dengan menggunakan *classification report*. Nilai ini akan digunakan untuk melakukan perhitungan F1 *score*. Tahap ini dilakukan dengan bantuan *library Sklearn Metrics* yang dirancang khusus untuk evaluasi algoritma klasifikasi pada *machine learning*.

Pada tahapan setelah split dataset, selain melakukan feature extraction dengan training model FastText dan vektorisasi dataset dengan Mean Emdedding Vectorizer, proses feature extraction juga akan dilakukan dengan menggunakan TF-IDF untuk membuktikan apakah dengan menggunakan dataset ini performanya dapat lebih baik. TF-IDF hanya akan dilakukan pada skenario terakhir setelah ditemukan skenario terbaik jika menggunakan FastText dengan Mean Embedding Vectorizer.

3.2.2 Flowchart Deployed Version

Model klasifikassi hasil *training* yang telah dilakukan akan dievaluasi dan dijalankan atau di-*deploy* ke dalam aplikasi web demo agar lebih mudah dipahami dan lebih terlihat gambarannya. Dalam aplikasi web demo ini akan ada 2 jenis *input* yang bisa digunakan untuk menggambarkan hasil prediksi dari algoritma *MLP Classifier*, yaitu input berupa kalimat menggunakan *text box* dan berupa file .csv yang diunggah ke *web* menggunakan *form input file* yang akan menghasilkan presentase dari sentimen yang diprediksi.

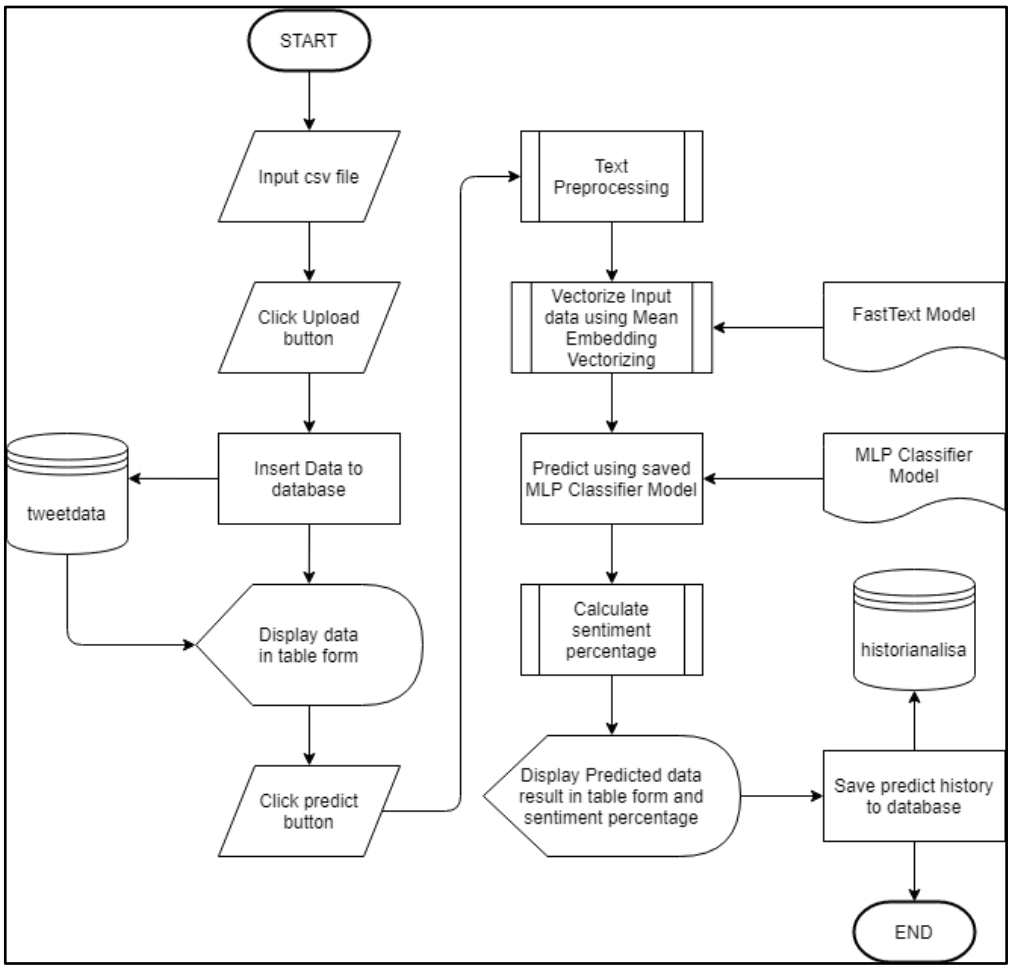
Pada Gambar 3.7 merupakan *flowchart* yang menjelaskan proses untuk memprediksi *input* dari kalimat *tweet* yang diberikan melalui *text box* pada aplikasi demo *web*. Setelah melakukan *input* kalimat kemudian menekan tombol prediksi. Setelah itu, *web* akan menjalankan proses yang sama seperti proses klasifikasi yang telah dijelaskakan sebelumnya tanpa melui proses *training* karena proses prediksi sudah menggunakan model hasil *export* dari proses *training* algoritma *MLP Classifier* yang telah dilakukan. Lalu, hasil prediksi akan ditampilkan pada halaman *web*.



Gambar 3.7 *Flowchart* prediksi *input* kalimat *tweet* pada *text box*

Flowchart yang ditunjukkan pada Gambar 3.8 merupakan proses prediksi dengan *input* berupa *file* .csv yang diunggah ke dalam apliaksi *web* demo. *File* yang

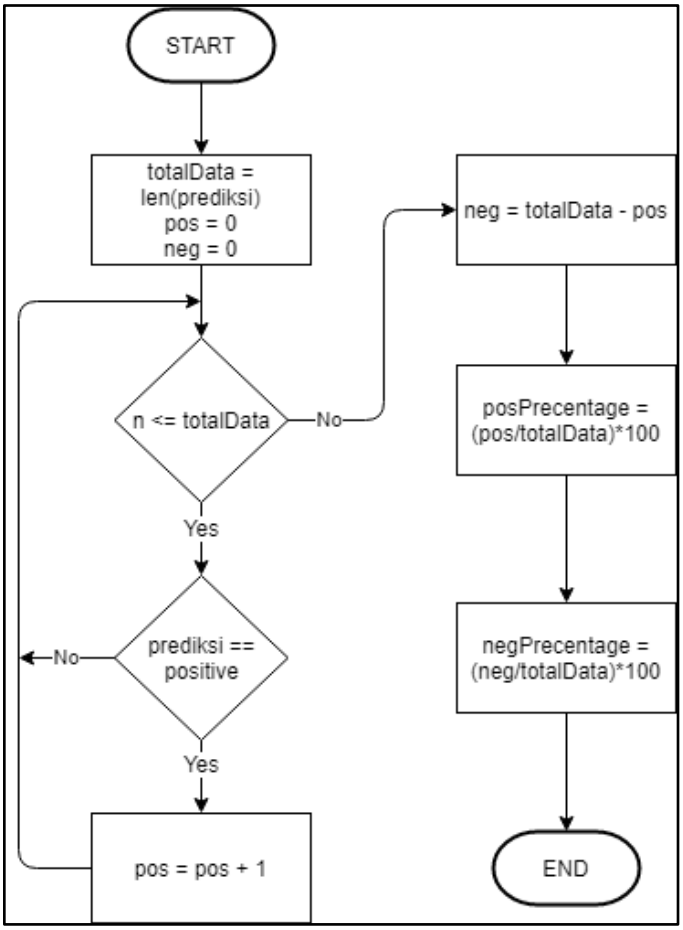
telah diunggah tersebut akan ditampung sementara ke dalam *database* dan ditampilan di halaman *web*. Penampungan sementara ke *database* ini dilakukan karena pada *framework* Flask tidak memungkinkan penampungan *file* berukuran besar untuk dijadikan *global variable* untuk penggunaan pada setiap *route* dari Flask. Setelah itu ketika tombol prediksi ditekan, maka proses yang sama seperti Gambar 3.7 akan dilakukan.



Gambar 3.8 *Flowchart* prediksi dengan *input* berupa *file*

Model *MLP Classifier* akan memprediksi setiap baris data yang terdapat pada *database*. Hasil prediksi akan ditampilkan pada halaman *web* dalam bentuk

tabel disertai persentase sentimen yang keluar dab akan disimpan historinya ke database. *Flowchart* proses penghitungan persentase dapat dilihat pada Gambar 3.9.



Gambar 3.9 Flowchart penghitungan presentase sentimen

3.2.3 Rancangan Struktur Tabel

Tabel 3.1 merupakan struktur tabel *Data* pada *database* yang menggunakan MYSQL. Tabel ini hanya memiliki 1 kolom. Tabel ini berfungsi untuk menampung sementara konten *tweet* dari *dataset* yang diunggah ke halaman *web* sebelum diprediksi. Penampungan sementara ke web ini dilakukan karena pada *framework*

Flask tidak memungkinkan penampungan file yang cukup besar untuk dijadikan *global variable* untuk penggunaan pada setiap *route* dari Flask.

Tabel 3.1 Struktur Tabel Data

Nama Kolom	Tipe Data	Atribut
tweet	text	NOT NULL

Tabel 3.2 merupakan struktur tabel Histori pada database yang menyimpan histori prediksi dari beberapa data yang dicoba. Data yang disimpan adalah nama dari data toko *online* dan presentasi positif negatifnya.

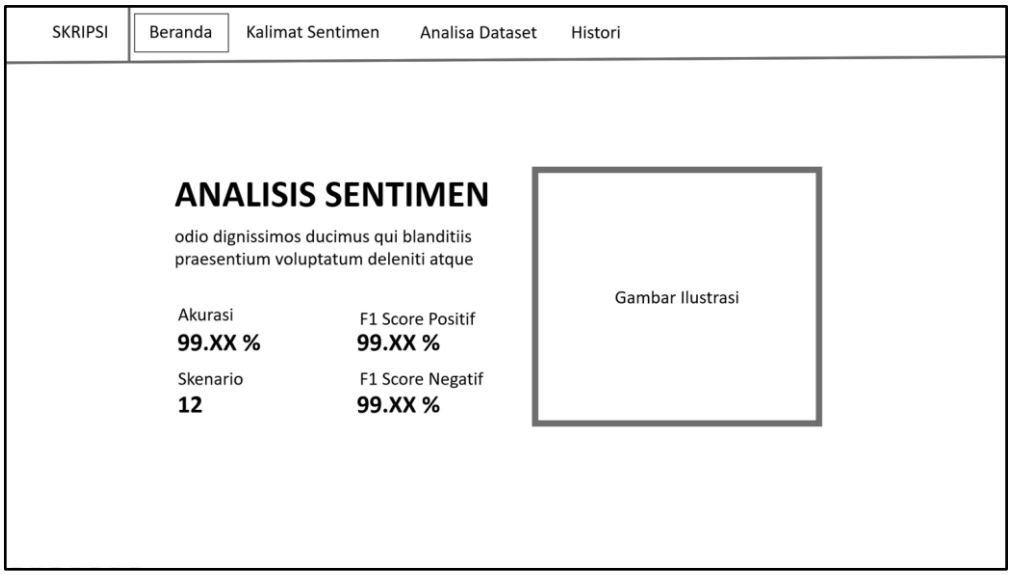
Tabel 3.2 Struktur Tabel Histori

Nama Kolom	Tipe Data	Atribut
namatoko	VARCHAR (255)	NOT NULL
positif	float	NOT NULL
negatif	float	NOT NULL

3.2.4 Rancangan Tampilan Antarmuka

Tampilan antarmuka aplikasi *web* dibuat secara *single page web application* yang dibagi menjadi beberapa *section* atau bagian, yaitu bagian Beranda, bagian Kalimat Sentimen, bagian Analisa Dataset, bagian Hasil Analisa, bagian Presentasi, dan bagian histori. Untuk bagian Hasil Analisa dan Presentase tidak dakan ditampilkan diawal halaman *web* dibuka, namun baru akan muncul ketika tombol prediksi pada halaman Anlisa Dataset ditekan.

Gambar 3.10 merupakan rancangan antarmuka untuk bagian Beranda. Pada bagian ini terdapat penjelasan singkat mengenai aplikasi web demo yang ditampilkan dan performa sistem yang dibuat.



Gambar 3.10 *Mockup* tampilan bagian Kalimat Sentimen

Pada Gambar 3.11 terdapat rancangan antarmuka untuk bagian Kalimat Sentimen. Di bagian inilah *user* dapat memasukkan *input* berupa kalimat *tweet* yang kemudian bisa diprediksi sentimennya secara langsung saat tombol prediksi ditekan.

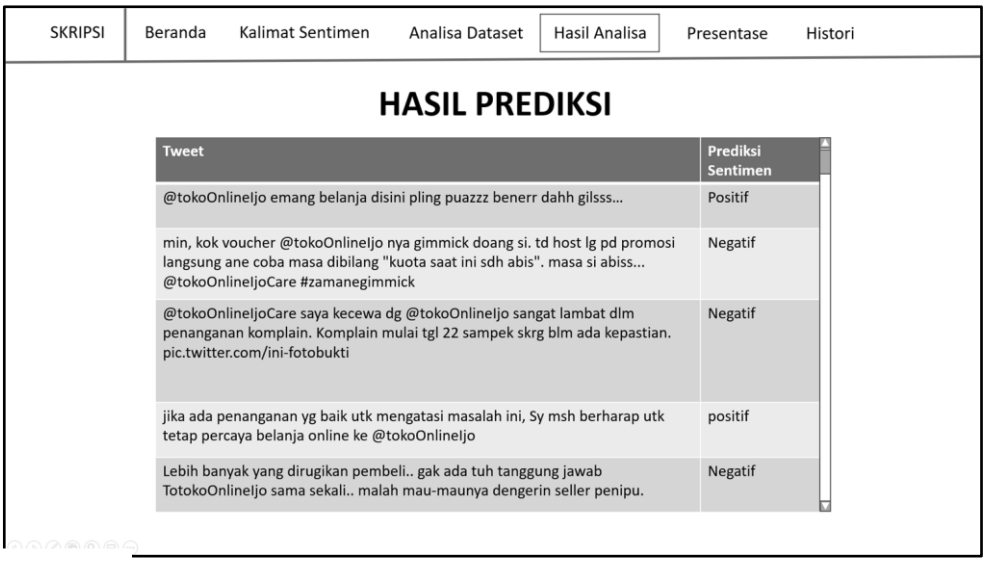
Gambar 3.11 *Mockup* tampilan bagian Kalimat Sentimen

Pada Gambar 3.12 terdapat rancangan antarmuka untuk bagian Analisa Dataset. Di halaman ini *user* dapat mengunggah *file* berekstensi .csv. Setelah tombol unggah di tekan akan menampilkan isi dari *file* kumpulan *tweet* seperti yang ditunjukkan pada Gambar 3.12.

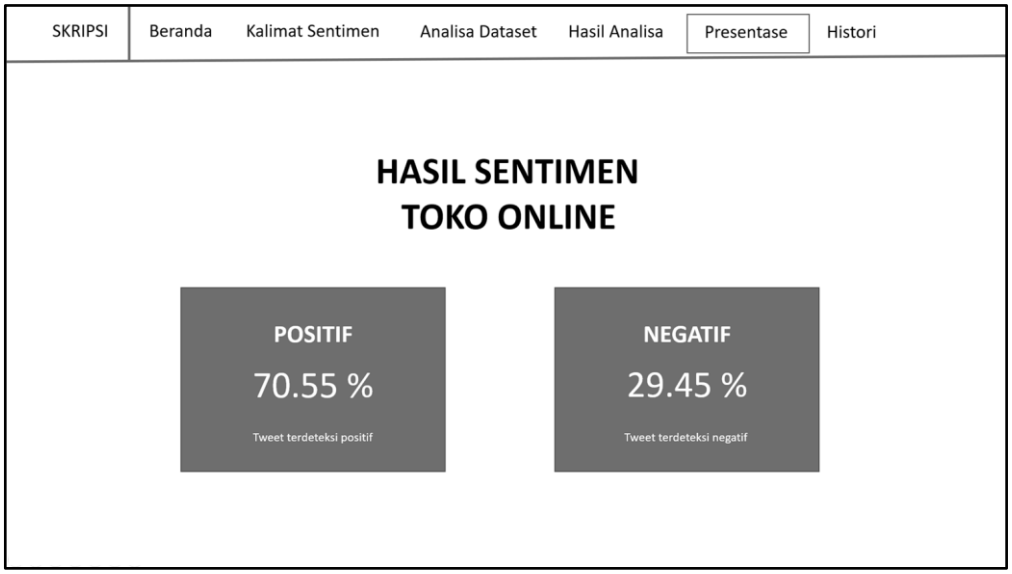
Gambar 3.12 *Mockup* tampilan bagian Analisa Dataset

Saat tombol prediksi ditekan maka selanjutnya akan memproses pengecekan satu persatu *tweet* untuk deprediksi yang hasilnya dapat dilihat pada

Gambar 3.13. Kemudian presentasi hasil sentimen juga akan ditampilkan seperti pada Gambar 3.14.

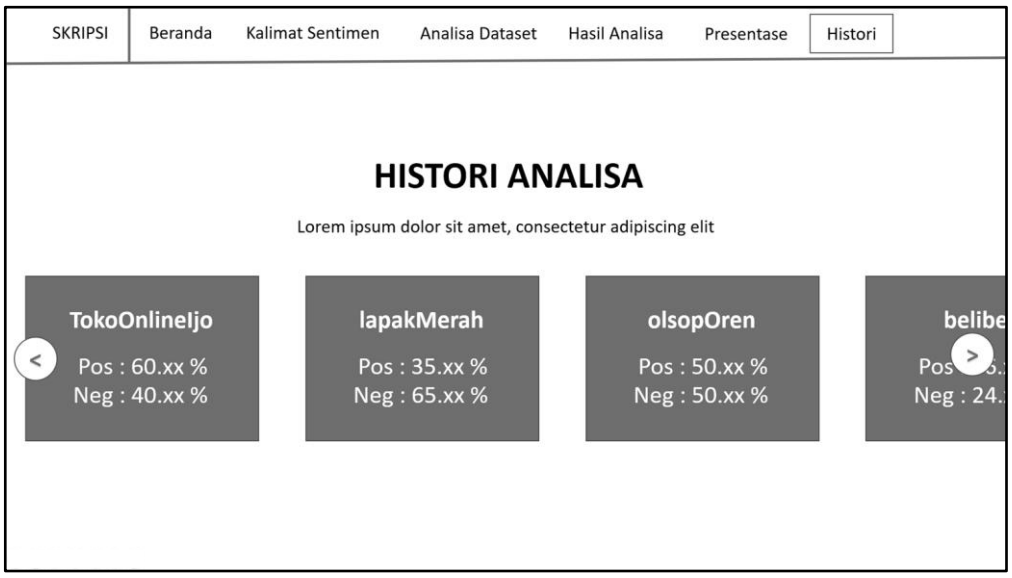


Gambar 3.13 *Mockup* tampilan bagian Hasil Analisa



Gambar 3.14 *Mockup* tampilan bagian Presentase

Histori dari setiap pengecekan akan disimpan ke database dan ditampilkan pada bagian Hisotri seperti yang ditunjukkan pada Gambar 3.15 dengan format slider otomatis.



Gambar 3.15 *Mockup* tampilan bagian Histori