BAB III

METODOLOGI PENELITIAN DAN PERANCANGAN SISTEM

3.1 Metodologi Penelitian

Penelitian "Implementasi Algoritma Random Forest Menggunakan TF-IDF Untuk Analisis Sentimen Dengan Penerapan *Transfer Learning*" menggunakan beberapa tahap, tahap-tahap yang dilaksanakan antara lain adalah sebagai berikut.

1. Studi Literatur

   Tahap studi literatur dilakukan dengan mencari, membaca dan mempelajari sumber dari jurnal ilmiah dan karya tulis terkait dengan topik yang diteliti. Tahap studi literatur bertujuan untuk memahami teori mengenai TF-IDF, Random Forest Classifier, dan n-grams.

2. Pengumpulan Data

   Dataset yang didapatkan merupakan data ulasan pengguna kepada perusahaan Amazon dan Yelp yang tersedia di dalam situs Kaggle. *Dataset* yang digunakan terdiri dari 5000 baris data. *Dataset* tersebut telah diberi label sebelumnya berupa label positif dan label negatif. Namun pada umumnya jika *dataset* belum diberi label, maka cara memberikan label adalah dengan bantuan ahli bahasa dan dengan melakukan *majority voting* yaitu hasil *voting* terbanyak yang dilakukan oleh beberapa orang.

3. Perancangan dan Pengembangan Program

Tahap perancangan program terdiri atas pembuatan alur kerja untuk penerapan TF-IDF, Random Forest Classifier, *transfer learning*, dan alur antar muka pada aplikasi web.

4. *Testing* dan *debugging*

Tahap ini bertujuan untuk memastikan dan menguji apakah program *machine learning* dan web yang telah dibangun telah berjalan dengan baik serta melakukan perbaikan jika terdapat kesalahan. Pada program *machine learning* dilakukan pengujian pada hasil dari pembobotan TF-IDF, hasil dari tahap *preprocessing*, dan uji performa dari Random Forest Classifier. Pada program web, dilakukan pengujian apakah website dapat menerapkan model *machine learning* yang telah dikembangkan.

5. Penulisan Laporan.

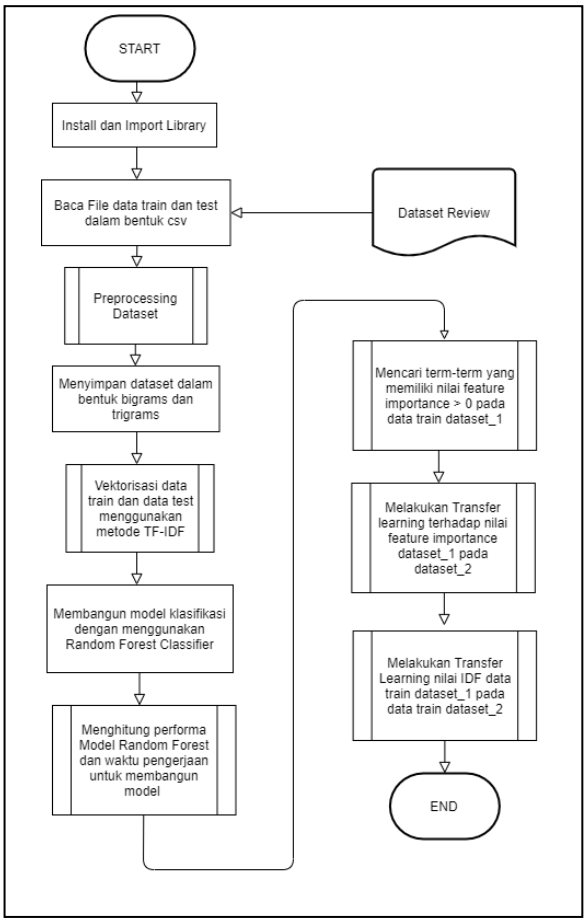
Laporan yang disusun bertujuan sebagai dokumentasi atas penelitian yang telah dilakukan. Penyusunan laporan dibuat secara bertahap, dimulai dari pendahuluan, latar belakang, kesimpulan, dan saran.

3.2 Perancangan Aplikasi

Perancangan aplikasi dilakukan dengan merancang *flowchart* yang menjabarkan tahapan dalam penerapan Random Forest Classifier dan metode *transfer leaning* dimulai dari *preprocessing* sampai dengan pembuatan model, serta menjabarkan tentang tahapan dalam penggunaan Random Forest Classifier untuk analisis sentiment dalam aplikasi web. Pada tahap ini juga dilakukan perancangan antarmuka aplikasi web.

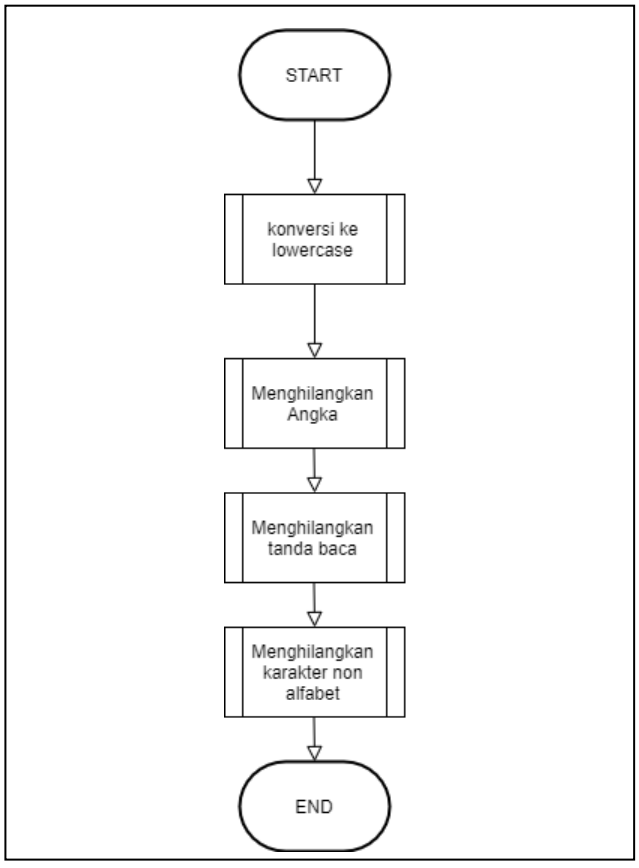
3.2.1 Flowchart

Gambar 3.1 adalah *flowchart* atau diagram alir dari proses klasifikasi sentimen ke dalam sentimen positif dan negatif menggunakan TF-IDF dan Random Forest Classifier dengan metode *transfer learning* yang digambarkan secara umum. Proses pertama yang dilakukan adalah *import library* yang dibutuhkan untuk mendukung proses penelitian. Lalu *dataset* dalam bentuk file *csv* yang telah diolah sebelumnya akan di-*import* yang selanjutnya akan diproses. *Dataset* yang digunakan adalah *dataset* sentiment ulasan pengguna kepada perusahaan Amazon, Yelp, dan IMDB.

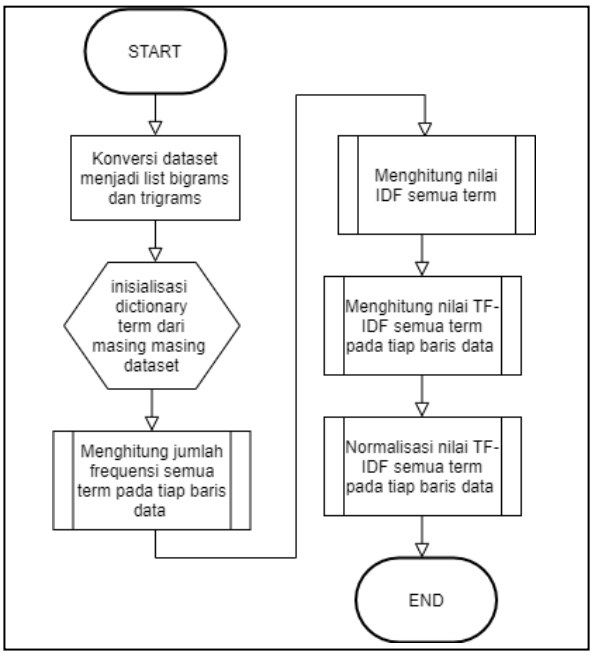


Gambar 3.1 Flowchart Utama

Dataset yang telah dibaca terdiri dari data *train* dan data *test* dari masing-masing data ulasan tiap perusahaan. Lalu tahap selanjutnya adalah melakukan *preprocessing* pada *dataset*. Dalam proses *preprocessing*, ada beberapa tahap yang dilakukan, seperti yang dapat dilihat pada Gambar 3.2 Tahap-tahap yang dilakukan antara lain *lowering case* (mengecilkan huruf), menghilangkan angka, menghilangkan *punctuation* (tanda baca), dan menghilangkan karakter yang bukan huruf alfabet. *Preprocessing* dilakukan dengan bantuan *library* string dan re *(regular expression)*.

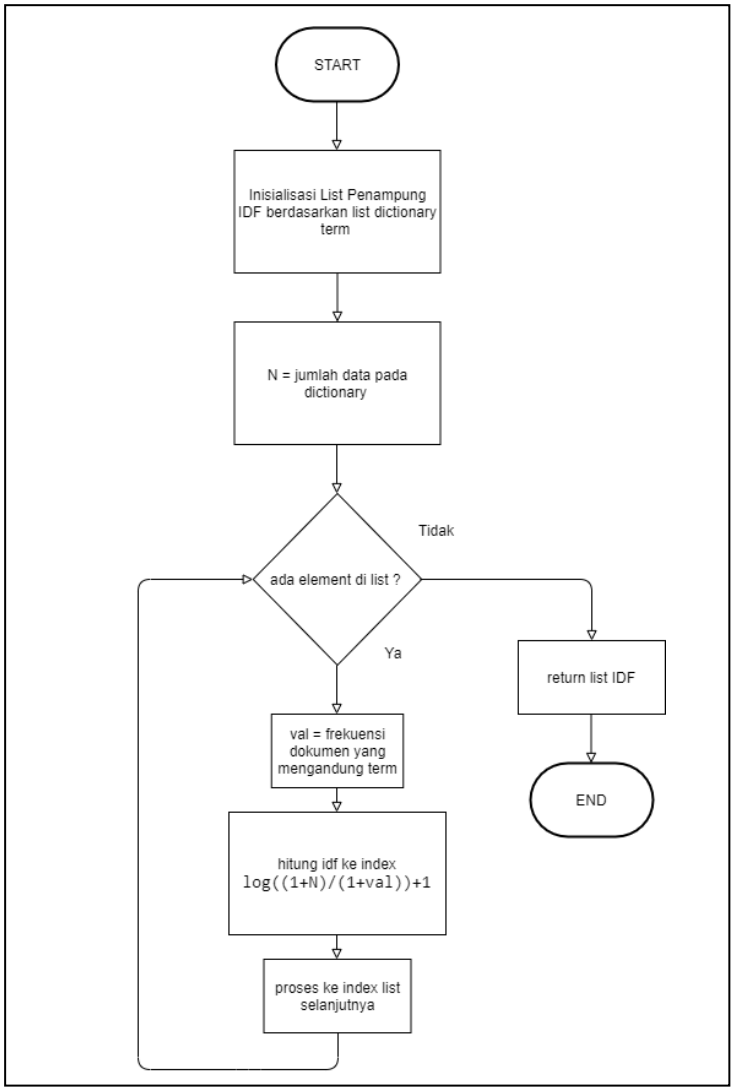


Gambar 3.2 Flowchart Tahap *Preprocessing*



Gambar 3.3 Flowchart Tahap Proses Vektorisasi TF-IDF

Setelah tahap *preprocessing*, akan dilakukan vektorisasi data menggunakan metode TF-IDF. Vektorisasi menggunakan TF-IDF mempunyai beberapa tahap yang dapat dilihat pada Gambar 3.3 yang merupakan penjabaran untuk proses vektorisasi menggunakan TF-IDF. Langkah pertama yang dilakukan adalah mengubah data sentimen pengguna pada masing masing data *train* dan data *test* menjadi bentuk n-grams. Dalam penelitian ini, bentuk n-grams yang digunakan adalah bigrams dan trigrams. Setelah data diubah dalam bentuk n-grams, langkah selanjutnya adalah menginisialisasi data *list dictionary* setiap data *train*. *List* tersebut memiliki indeks berupa *term* untuk setiap baris data yang didapatkan dari proses pengubahan bentuk *bigrams* dan trigrams data *train*. Lalu tahap selanjutnya adalah menghitung frekuensi kemunculan tiap *term* untuk setiap baris data. Langkah ini diperlukan untuk menghitung nilai TF (*term frequency*).



Gambar 3.4 Flowchart Tahap Proses Perhitungan Nilai IDF

Setelah membuat *list dictionary* untuk masing-masing data *train*, langkah selanjutnya adalah menghitung nilai IDF (*Inverse Document Frequency*) pada setiap *term* untuk masing-masing data *train*. Gambar 3.4 merupakan tahapan untuk menghitung nilai IDF setiap *term* pada semua data *train*. Langkah pertama yang dilakukan adalah inisialisasi *list* yang akan digunakan untuk menampung nilai IDF. Lalu dilakukan *looping* untuk setiap *term* yang terdapat pada *list dictionary*. Di

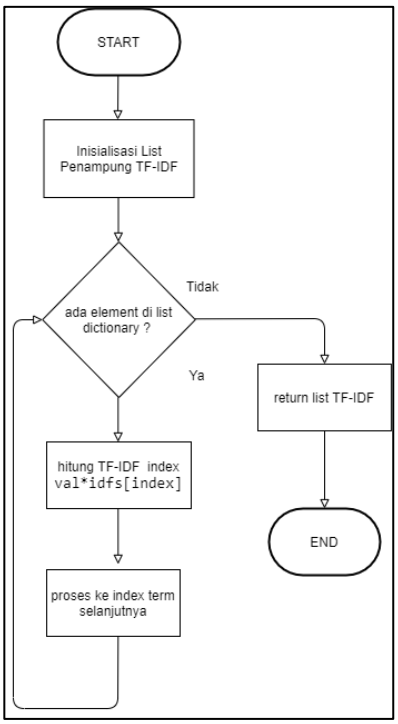
dalam *loop* tersebut, dilakukan perhitungan nilai IDF pada *term* yang sedang menjadi index *looping* tersebut. Rumus IDF yang digunakan adalah sebagai berikut.

$$IDF = \log((1 + N) / (1 + Val)) + 1 \quad (3.1)$$

Keterangan :

8. N = Banyaknya baris data pada data *train*
9. *Val* = banyaknya data yang mempunyai *term* ke index

Rumus yang digunakan diatas terdapat sedikit perbedaan pada rumus IDF yaitu penambahan angka ‘1’ pada pembilang dan penyebut, yang berguna jika menghasilkan nilai yang memicu kesalahan, seperti *division by zero*.



Gambar 3.5 Flowchart Tahap Proses Perhitungan Nilai TF-IDF

Setelah menghitung dan membuat *list* nilai IDF, tahap selanjutnya adalah menghitung nilai TF-IDF untuk semua *term* pada setiap baris data *train*. Gambar 3.5 merupakan penjabaran untuk proses perhitungan nilai TF-IDF. Langkah pertama yang dilakukan adalah inisialisasi *list* yang akan digunakan untuk menampung nilai TF-IDF. Lalu dilakukan *looping* untuk setiap *term* yang terdapat satu baris data pada data train. Di dalam *loop* tersebut, dilakukan perhitungan nilai TF-IDF non normalisasi pada *term* yang sedang menjadi index *looping* tersebut. Rumus TF-IDF yang digunakan adalah sebagai berikut

$$TF - IDF = Val * IDF \quad (3.2)$$

Keterangan :

1. Val = Frekuensi kemunculan *term* pada suatu data

Setelah menghitung nilai TF-IDF non normalisasi maka tahap selanjutnya adalah menormalisasi nilai TF-IDF di dalam *list* tersebut. Berikut rumus yang digunakan untuk normalisasi nilai TF-IDF.

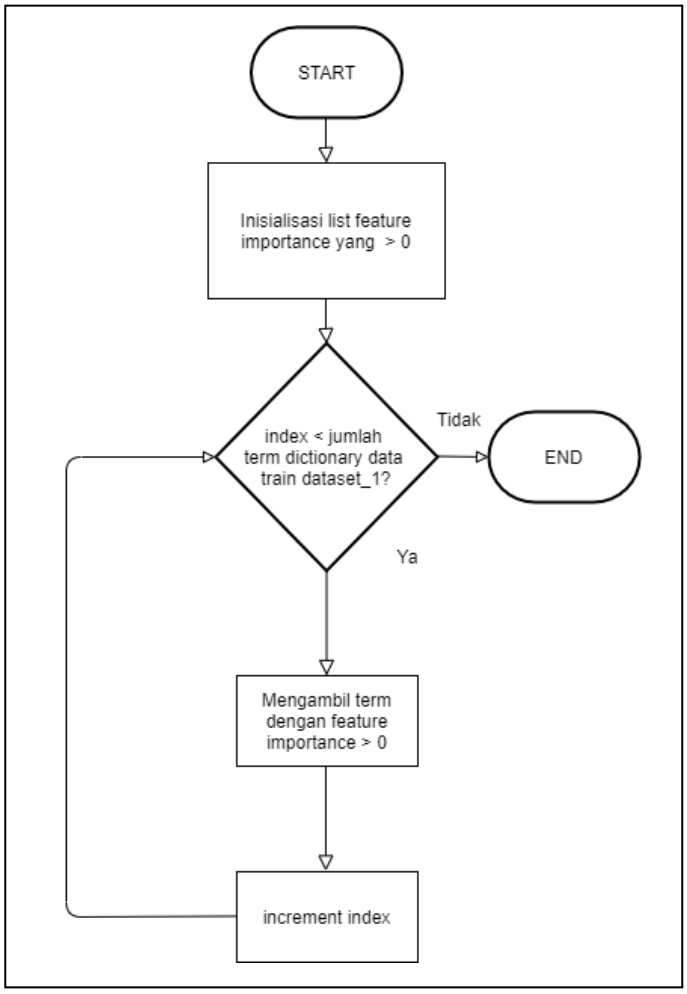
$$Vnorm = \frac{v}{\sqrt{v1^2 + v2^2 + \cdots + vn^2}} \quad (3.3)$$

Keterangan :

1. v = nilai TF-IDF non normalisasi

Setelah tahap perhitungan TF-IDF selesai, lalu dilakukan tahap selanjutnya yaitu membuat model Random Forest dengan menggunakan data train yang sudah diproses dan divektorisasi sebelumnya. Dalam penelitian ini model Random Forest dibuat dengan menggunakan *library* Random Forest Classifier dari scikit-learn,

karena *library* ini mempunyai fitur untuk mengembalikan nilai berupa *feature importance* tiap *term* yang nantinya akan digunakan untuk proses *transfer learning*.

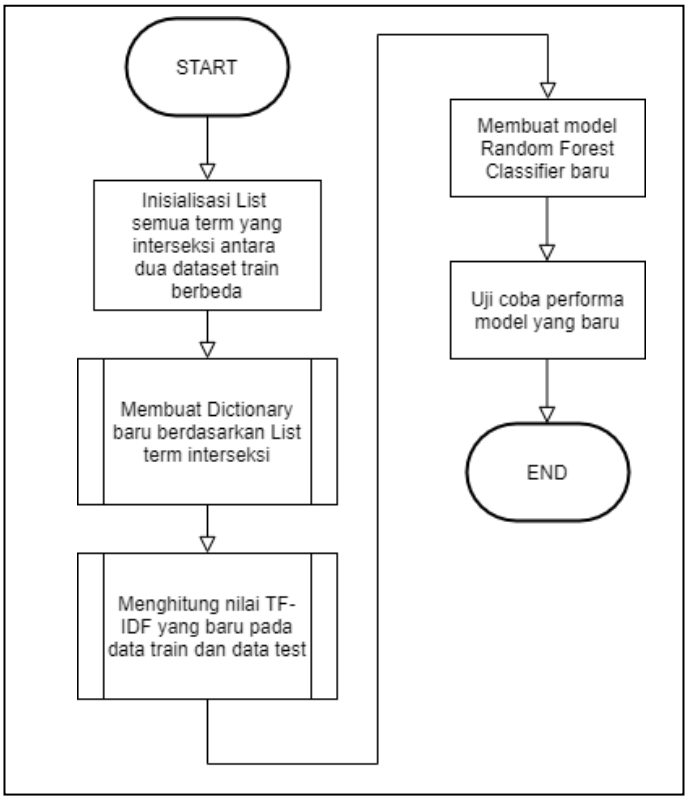


Gambar 3.6 Flowchart Tahap Proses Pembuatan List *Feature Importance*

Lalu setelah model selesai dibuat, tahap selanjutnya adalah melakukan uji coba pada model tersebut. Uji coba dilakukan dengan data *test* yang sudah divektorisasi sebelumnya dan berikut dengan label-nya. Setelah uji performa model dilakukan maka akan didapatkan skor performa model, seperti akurasi *train*, akurasi *test*, nilai *precision, recall,* dan *f1-score*. Serta juga dapat menghitung berapa lama model memerlukan waktu untuk *training* dan *testing* data. Dalam penelitian ini juga

dilakukan tahap *transfer learning* untuk mengetahui apakah dapat mempengaruhi performa dari model Random Forest yang telah dibangun.

Transfer learning pertama yang dilakukan adalah dengan menggunakan informasi nilai dari *feature importance* untuk setiap *term* pada model Random Forest yang di-*training* pada *dataset* pertama dan kemudian diterapkan pada dataset selanjutnya. Tahap pertama yang dilakukan adalah membuat *list* dari *term-term* yang memiliki nilai *feature importance* lebih dari 0 pada model Random Forest *dataset* pertama. Gambar 3.6 merupakan penjabaran langkah-langkah untuk membuat *list* tersebut. Langkah pertama yang dilakukan adalah inisialisasi *list* yang akan digunakan, lalu akan dilakukan *looping* untuk setiap *term* yang terdapat pada *dictionary dataset* pertama. Di dalam *looping* tersebut akan dilakukan pengecekan apakah *term* tersebut mempunyai *feature importance* yang lebih besar dari 0, jika lebih besar maka *term* tersebut akan ditambahkan ke dalam *list*. Setelah *looping* selesai mengakses semua *term* yang terdapat pada *dictionary* maka *list* akan di-*return*.

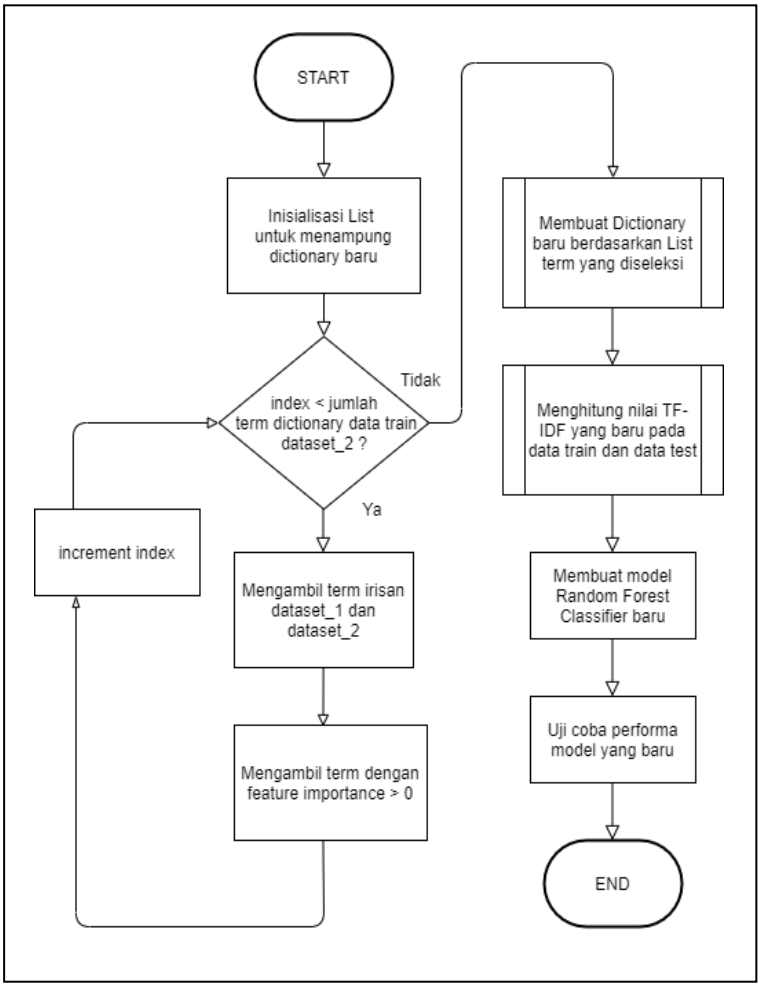


Gambar 3.7 Flowchart Implementasi *Transfer Learning* Pada Interseksi *Dataset*

Pada proses *transfer learning feature importance* akan dilakukan dua skenario uji coba, yaitu skenario pertama dengan menggunakan data *dictionary* yang dibuat berdasarkan *term-term* yang saling interseksi saja antara *dataset* pertama dan *dataset* kedua, dan skenario kedua, yaitu dengan seleksi data *term* pada *dataset* kedua berdasarkan informasi *feature importance* yang didapatkan dari model Random Forest *dataset* pertama. Gambar 3.7 merupakan *flowchart* penjabaran langkah yang dilakukan dalam implementasi *transfer learning feature importance* pada skenario pertama.

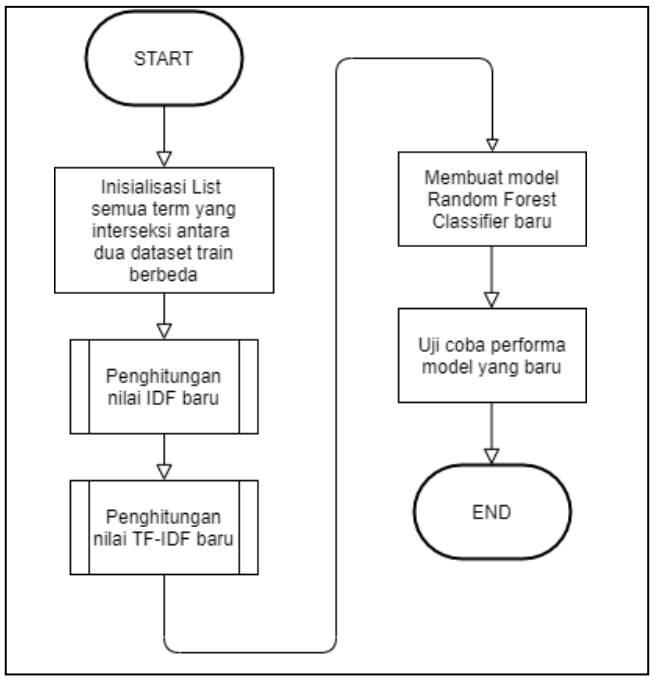
Langkah pertama yang dilakukan adalah inisialisasi *list* yang berisi semua *term* yang interseksi antara *dataset* pertama yang merupakan *dataset* yang telah di-

training menjadi *pretrained model* dan *dataset* kedua. Lalu langkah selanjutnya adalah membuat *dictionary term* berdasarkan *term* dari *list* tersebut dan mengisi nilai dari tiap indeks *term* dengan frekuensi kemunculan *term* tersebut pada setiap baris data. Selanjutnya dilakukan perhitungan nilai TF-IDF pada *dictionary* tersebut sesuai langkah pada Gambar 3.3 yang telah dideskripsikan sebelumnya. Setelah tahap perhitungan TF-IDF selesai, lalu dilakukan tahap selanjutnya yaitu membuat model Random Forest dan melakukan pengujian kembali model Random Forest baru tersebut.

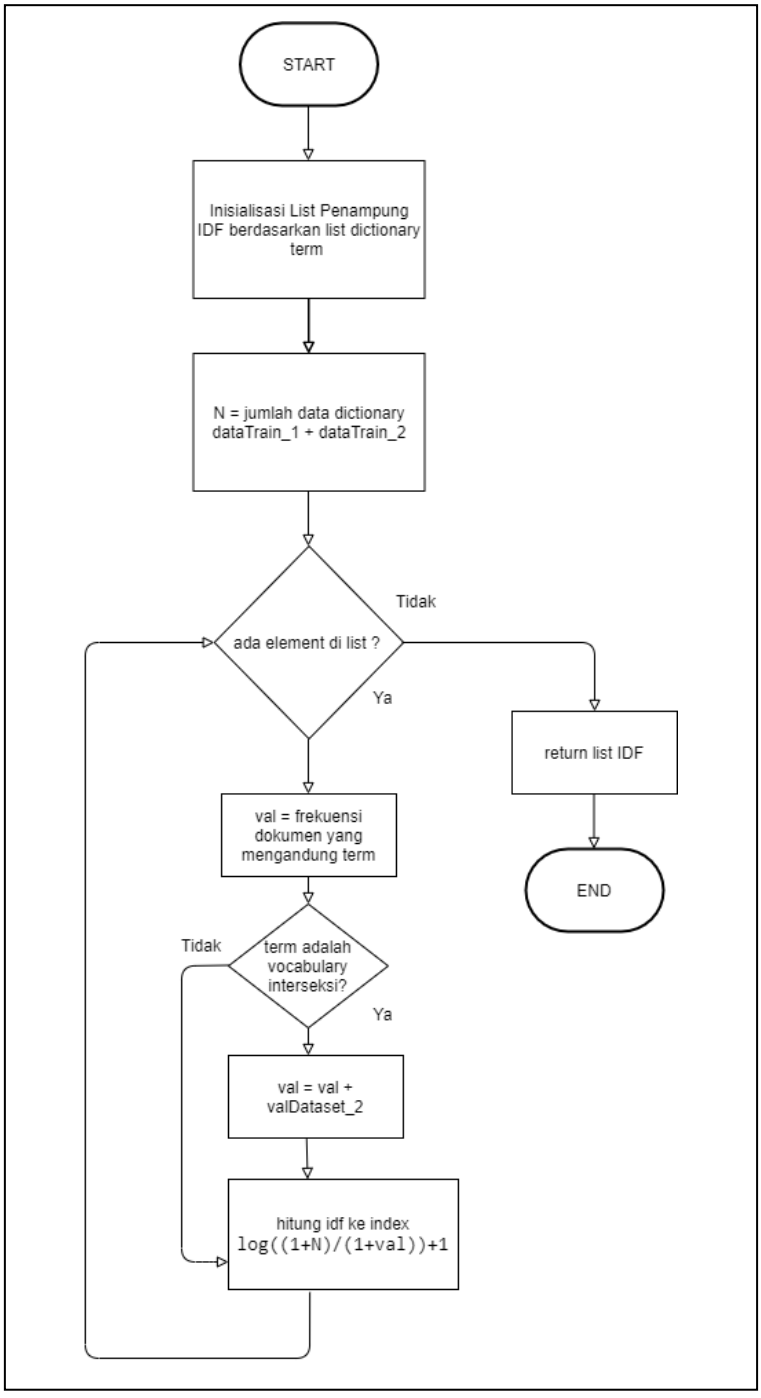


Gambar 3.8 Flowchart Implementasi *Transfer Learning* Dengan Seleksi Dataset

Gambar 3.8 merupakan langkah untuk implementasi *transfer learning* pada *feature importance* skenario kedua, yaitu dengan seleksi *term* atau *feature* pada *dataset* kedua. Langkah pertama yang dilakukan adalah inisialisasi *dictionary* yang akan menampung *term-term* baru hasil seleksi dari *dataset* kedua. Lalu akan dilakukan *looping* untuk setiap *term* terdapat pada *dictionary dataset* kedua. Pada *looping* tersebut dilakukan pengambilan data yang berinterseksi antara dataset_1 dan dataset_2 dan memiliki *feature importance* yang lebih besar dari nol . Setelah *looping* selesai mengakses semua *term* yang terdapat pada *dataset* kedua, maka proses dilanjutkan pada tahap membuat *dictionary term* yang baru berdasarkan *term* yang sudah diseleksi dan mengisi nilai dari tiap indeks *term* dengan frekuensi kemunculan *term* tersebut pada setiap baris data. Sama seperti skenario pertama, proses selanjutnya adalah menghitung nilai TF-IDF dan membangun model Random Forest yang baru berdasarkan nilai TF-IDF yang baru dan melakukan uji coba performa.



Gambar 3.9 Flowchart Implementasi *Transfer Learning* Dengan Nilai IDF



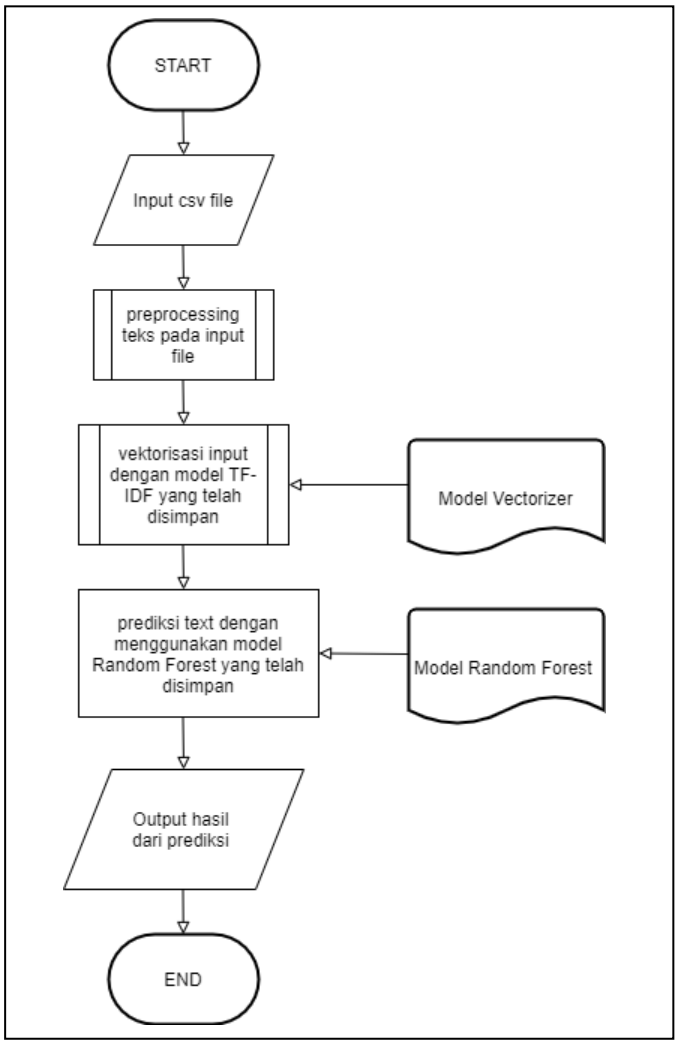
Gambar 3.10 Flowchart Perhitungan Nilai IDF *Transfer Learning*

Selain *transfer learning* Pada penelitian ini juga melakukan *transfer learning* nilai IDF pada *dataset* pertama dan menerapkan pada *dataset* kedua. Gambar 3.9 adalah *flowchart* yang menjabarkan langkah-langkah untuk menerapkan *transfer learning* IDF. Langkah pertama yang dilakukan adalah

inisialisasi *list* untuk menampung *term-term* yang saling interseksi antara *dataset* pertama dan *dataset* kedua. Lalu langkah selanjutnya adalah melakukan perhitungan terhadap nilai IDF baru pada *dictionary dataset* kedua dengan informasi pada nilai IDF pada *dataset* pertama.

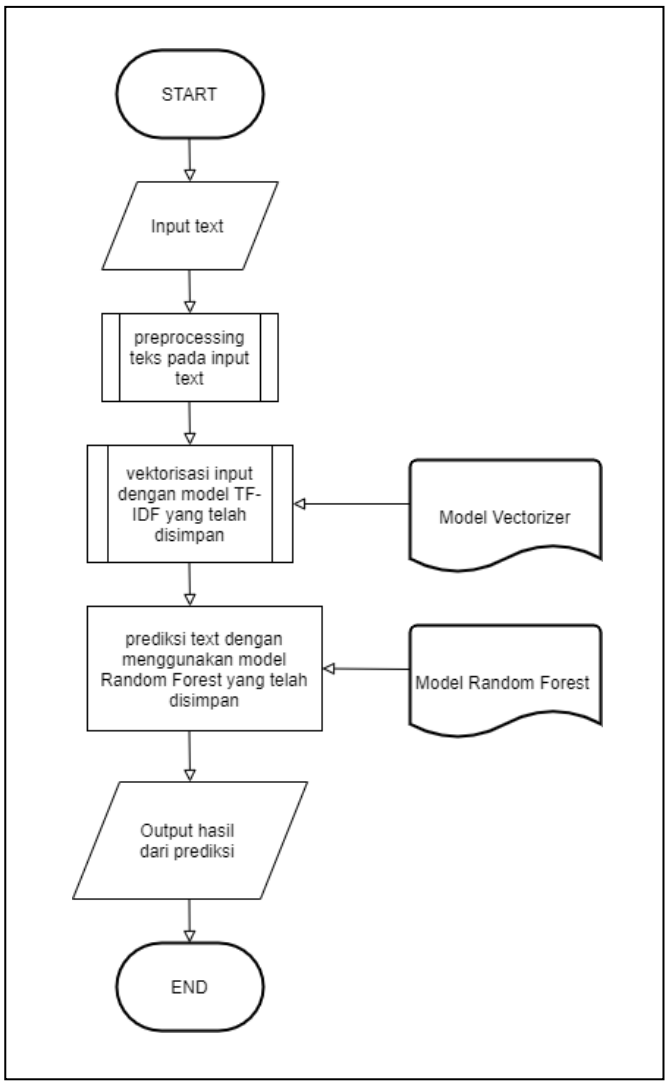
Gambar 3.10 merupakan *flowchart* yang menjabarkan langkah untuk menghitung nilai IDF yang baru pada *dictionary dataset* kedua. Langkah pertama yang dilakukan adalah me inisialisasi *dictionary* untuk menampung nilai IDF berdasarkan *term* dari *dictionary dataset* kedua. Selanjutnya akan dilakukan *looping* untuk setiap *term* yang terdapat pada *dictionary* yang telah diinisialisasi sebelumnya. Di dalam *looping* tersebut terdapat kondisi apakah *term* yang sedang diakses terdapat pada masing-masing *dataset* pertama dan *dataset* kedua. Jika kondisi tersebut bernilai *true*, maka nilai variabel *val* yang semula hanya menampung nilai frekuensi *term* indeks yang sedang diakses pada *dataset* kedua menjadi ditambahkan dengan *frekuensi term* indeks pada *dataset* pertama. Langkah selanjutnya adalah menghitung nilai IDF pada *term* yang sedang diakses dengan menggunakan rumus yang sama dengan perhitungan nilai IDF sebelumnya. Jika *looping* telah selesai mengakses semua indeks *term* maka *list dictionary* IDF akan di-*return*. Langkah selanjutnya adalah menghitung nilai TF-IDF dengan *list* nilai IDF yang baru dan membangun model Random Forest yang baru berdasarkan nilai TF-IDF yang baru dan melakukan uji coba performa.

Gambar 3.11 merupakan *flowchart* untuk proses prediksi *input* berupa data sentimen berbentuk file *csv* yang diunggah ke dalam aplikasi *web*. Ketika pengguna telah selesai *submit file csv*, maka akan dilakukan tahap preprocessing dengan langkah-langkah yang sama dengan sebelumnya. Selanjutnya dilakukan vektorisasi pada *input* dengan menggunakan model vectorizer TFIDF yang telah disimpan sebelumnya. Lalu dilakukan prediksi terhadap hasil vektorisasi tersebut dengan model Random Forest yang telah disimpan. Lalu hasil dari prediksi tersebut ditampilkan dengan menggunakan tabel.



Gambar 3.11 Flowchart Prediksi Input File Pada Aplikasi Web

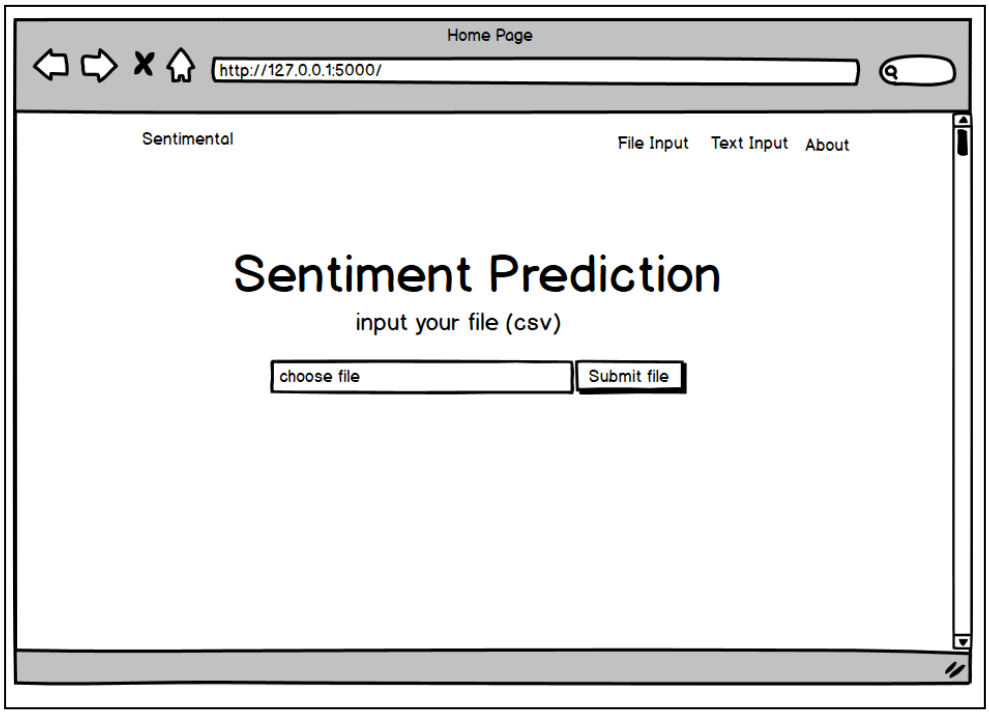
Gambar 3.12 merupakan *flowchart* untuk proses prediksi *input* berupa data sentimen berbentuk teks langsung. Ketika pengguna telah selesai *input* teks, maka proses yang sama seperti proses klasifikasi pada Gambar 3.11 akan dilakukan. Hasil prediksi akan ditampilkan pada halaman *web* apakah teks tersebut memiliki sentimen positif atau negative.



Gambar 3.12 Flowchart Prediksi Input Teks Pada Aplikasi Web

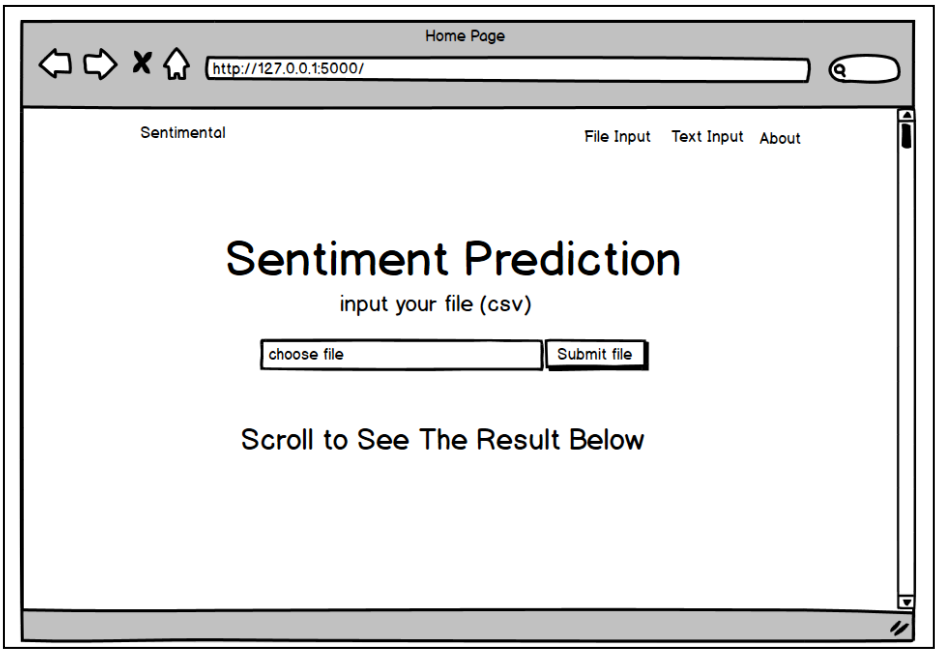
3.2.2 Rancangan Antarmuka

Rancangan antarmuka *website* terbagi menjadi tiga halaman, yaitu halaman *upload File, text input*, dan *about.* Pada Gambar 3.13 terdapat rancangan antarmuka untuk halaman *upload file*. Di halaman ini pengguna dapat *upload* dokumen dengan format *csv* yang mengandung data untuk diprediksi.



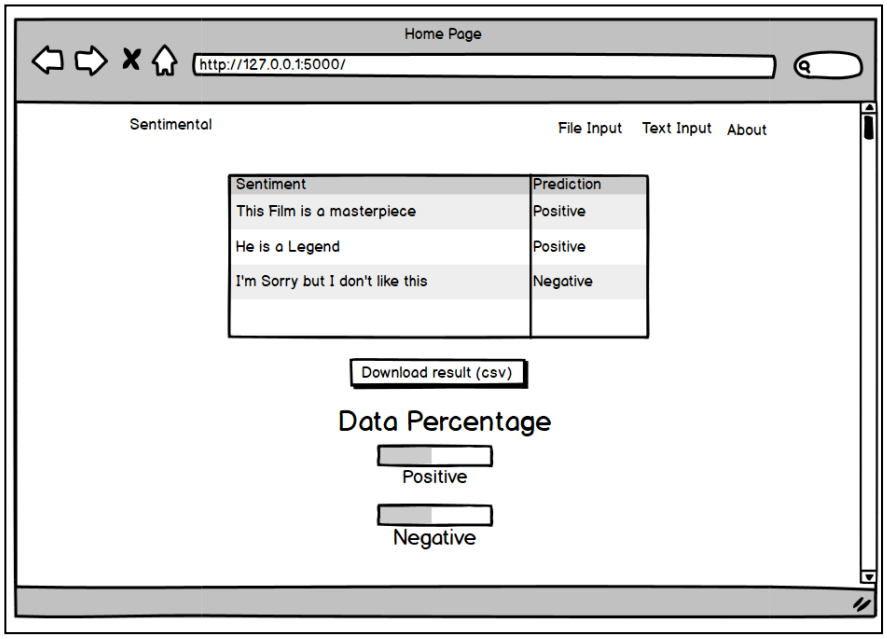
Gambar 3.13 Halaman *Upload File*

Pada Gambar 3.14 terdapat rancangan antarmuka pada halaman *upload file* jika pengguna telah berhasil *upload file csv*. Berbeda dengan antarmuka sebelumnya, pada antarmuka ini, terdapat informasi untuk pengguna agar *scroll* untuk melihat hasil dari prediksi *upload file* tersebut.



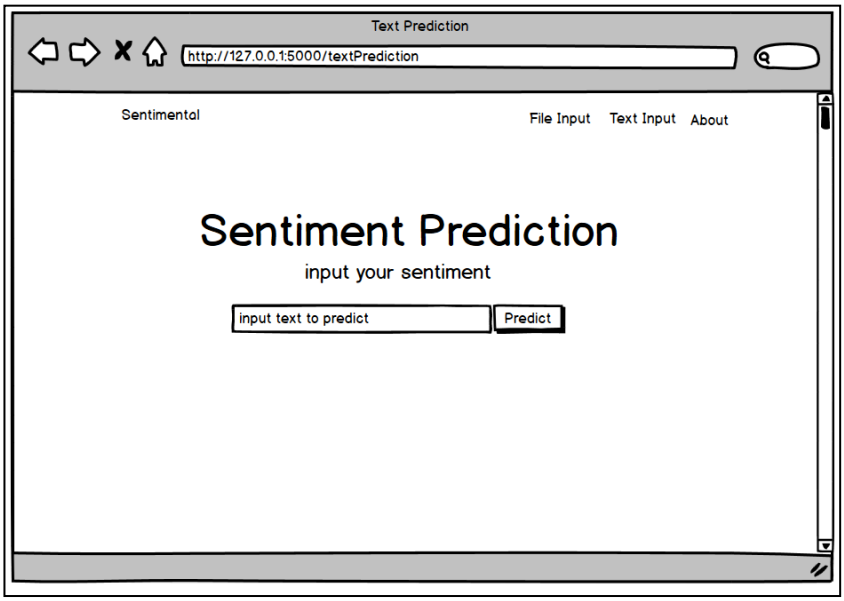
Gambar 3.14 Halaman *Upload File* Jika Telah Berhasil Upload

Gambar 3.15 adalah rancangan antarmuka pada halaman *upload file* bagian bawah. Pada bagian ini, pengguna dapat melihat hasil dari prediksi data file yang telah di-*upload* dan dapat *download* hasil dari prediksi tersebut. Pada bagian ini pengguna juga dapat melihat persentase hasil prediksi yang telah dimasukkan.

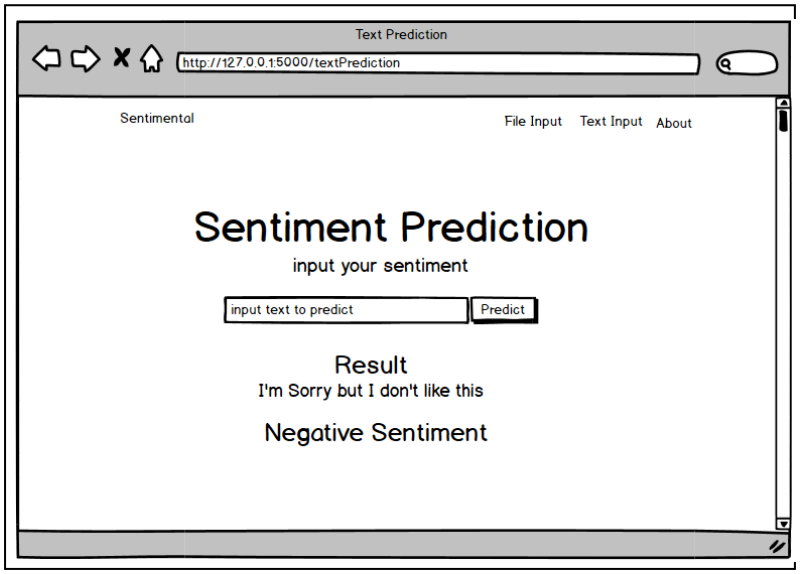


Gambar 3.15 Halaman *Upload File* Untuk Menampilkan Hasil Prediksi

Pada Gambar 3.16 terdapat rancangan antarmuka untuk halaman *text input*. Di halaman ini pengguna dapat memasukkan *input* sentimen yang kemudian bisa diprediksi dan ditampilkan hasilnya secara langsung. Gambar 3.17 merupakan tampilan halaman *text input* jika pengguna selesai memasukan input. Pada tampilan tersebut terdapat sentiment yang pengguna *input* berikut dengan hasil prediksi.

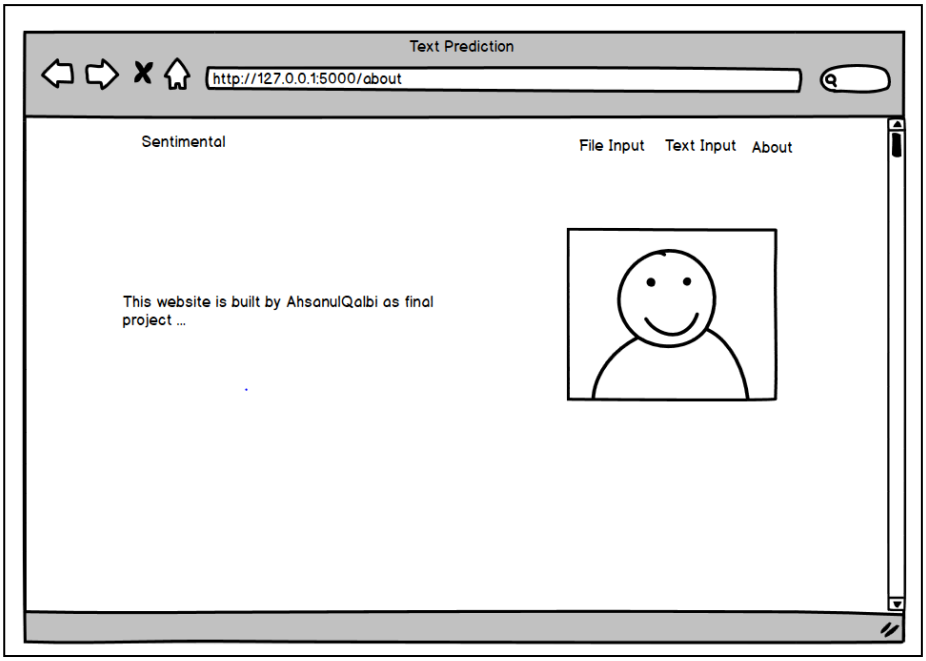


Gambar 3.16 Halaman *Input Text*



Gambar 3.17 Halaman *Input Text* Dengan Hasil Input

Pada Gambar 3.18 terdapat rancangan antarmuka untuk halaman *about me.* Di halaman ini terdapat informasi mengenai *website* berupa deskripsi dan alasan mengapa *website* dibuat. Pada halaman ini juga terdapat profil peneliti.



Gambar 3.18 Halaman *About Me*