BAB III

METODOLOGI PENELITIAN

3.1 Metodologi Penelitian

Penelitian sentimen analisis ini dilakukan dalam beberapa tahap, yaitu telaah literatur, analisis kebutuhan, perancangan dan pembuatan sistem, *testing*, serta konsultasi dan penulisan laporan.

1. Telaah literatur dilakukan dengan mempelajari dan mengumpulkan teori-teori yang memiliki keterkaitan dengan topik penelitian yang berasal dari jurnal, *conference* dan buku mengenai klasifikasi teks, sentimen analisis, *text preprocessing,* TF-IDF, Support Vector Machine, Logistic Regression, Multinomial Naïve Bayes, Ensemble Learning, Voting Classifier dan evaluasi performa.
2. Analisis kebutuhan dilakukan dengan menganalisa dan merancang sistem yang di dapat dari telaah literatur. Analisis kebutuhan ini dilakukan untuk mendapatkan requirements, baik *requirements* sistem ataupun *requirements* perangkat keras yang akan digunakan dalam pengimplementasian algoritma dan metode yang dipakai. Selain itu juga kebutuhan data yang akan digunakan.
3. Perancangan dan pembuatan sistem dimulai dengan merapihkan data yang akan diteliti, lalu *preprocessing* data, dilanjutkan dengan *training* data. Tahapan ini menggunakan bahasa pemrograman Python dengan *tools* Jupyter Notebook dan beberapa *library* untuk membantu dalam otomatisasi

pembelajaran. Serta pembuatan halaman *website* untuk kebutuhan demonstrasi klasifikasi kalimat. *Dataset* yang digunakan hanyalah *dataset* hasil kuesioner terkait pengalaman *work from home* dari department Human Resource and General Affairs Universitas Multimedia Nusantara, dengan total data 859 data dalam kelas positif sebanyak 571 data dan kelas negatif sebanyak 288 data.

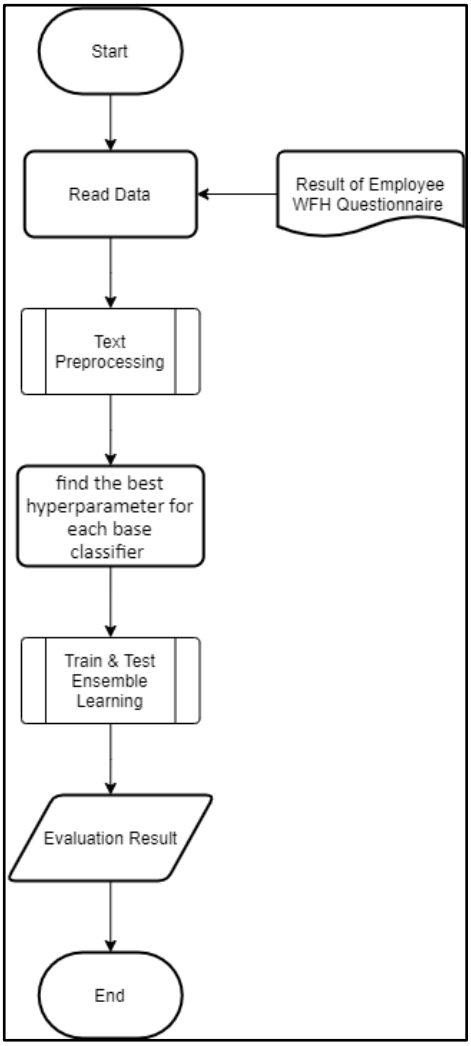
4. Uji Coba dan Evaluasi, tahap ini dilakukan dengan data yang telah disiapkan untuk uji coba terhadap model dari hasil *training* yang sudah dilakukan sebelumnya. Berdasarkan uji coba akan dilakukan evaluasi terhadap performa sistem dengan menganalisis performa.
5. Penulisan laporan dilakukan dengan tujuan untuk mendokumentasi hasil dari penelitian dan pembuatan sistem.

3.2 Perancangan Sistem

Perancangan sistem dalam penelitian ini akan menggunakan *flowchart* serta rancang bangun antarmuka aplikasi berbasis *web*.

3.2.1 Flowchart Utama

Gambar 3.1 merupakan tahapan-tahapan dari seluruh proses pembuatan sistem, dimulai dari proses *read data* dalam format .xlsx yang di dalamnya terdiri dari *feedback* dan juga label dari *feedback* tersebut dalam bahasa Indonesia. Setelah *read data*, kemudian data akan menuju tahap *text preprocessing* yaitu membersihkan kalimat atau teks. Manfaat dari dilakukannya *text preprocessing* adalah untuk mempermudah dalam pemrosesan di tahap selanjutnya agar lebih efektif dan dapat menghemat waktu.

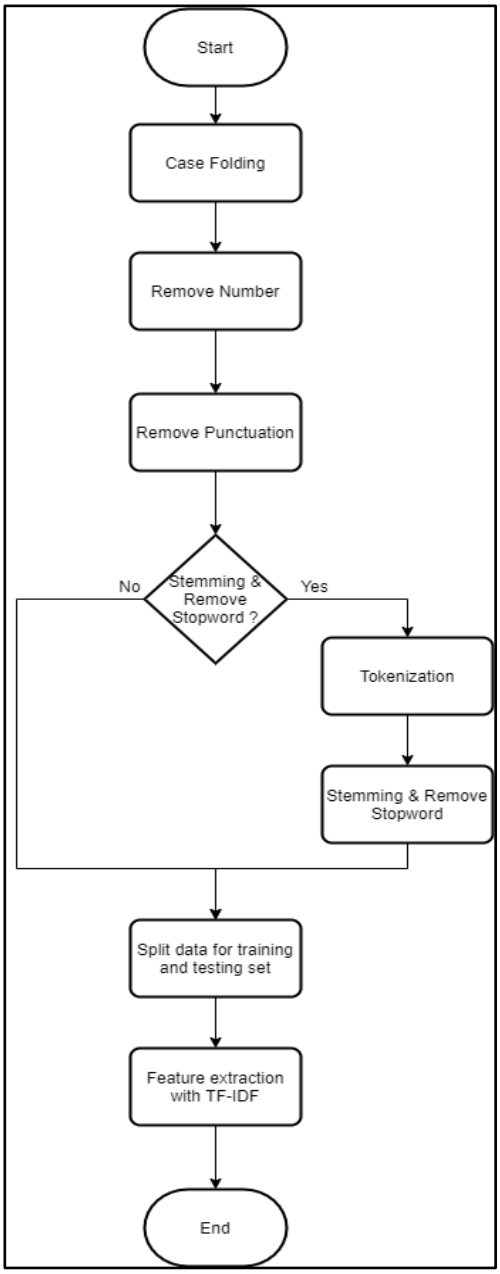


Gambar 3. 1 Flowchart Utama

Tahap *optimize hyperparameter for each base classifier* merupakan tahapan untuk mencari hyperparameter optimal dari setiap algoritma klasifikasi. Setiap algoritma klasifikasi akan membuat model dengan menggunakan hyperparameter yang berbeda-beda dalam *grid* yang sudah ditentukan. Kemudian hasil dari pencarian tersebut akan dilaporkan hyperparameter mana yang memiliki hasil performa paling baik. Tahap ini menggunakan *library* dari sckitLearn yaitu GridSearchCV.

Berikutnya merupakan tahapan di mana melatih model dengan *training* data. Setelah model latih jadi lalu akan dilakukan uji coba prediksi dengan menggunakan data *testing*. Kemudian tahap terakhir yaitu *evaluation result*.

3.2.2 Flowchart Text Preprocessing



Gambar 3. 2 Flowchart Text Preprocessing

Gambar 3.2 merupakan tahapan-tahapan *preprocessing* di mana tiap baris data akan melewati beberapa tahap seperti *case folding*, *remove number*, *remove punctuation*, melakukan *stemming* dan *remove stopword*, *split* data menjadi data *training* dan data *testing* dan *feature extraction*.

Tahap pertama yaitu *case folding*, di mana setiap kata yang memiliki huruf kapital akan diubah kedalam bentuk non-kapital dengan memanggil fungsi lower(). Tujuan dari *case folding* yaitu untuk menghindari adanya kemungkinan salah klasifikasi antara kata dengan huruf kapital dan non kapital yang sebenarnya merupakan kata yang sama. Selanjutnya yaitu menggunakan *regular expression* (*regex*) untuk tahap *remove number* dan *remove punctuation*. Setiap baris kalimat akan dilakukan pengecekan, apabila terdapat angka "\d" maka akan dihilangkan karena angka umumnya tidak diperlukan pada klasifikasi teks konvensional. Lalu pada tahapan berikutnya menghapus tanda baca atau karakter spesial. Apabila terdapat tanda baca akan di cek dengan "[^a-zA-Z0-9\s]" kemudian menggantinya dengan spasi atau dengan kata lain menghilangkan tanda baca. Lalu pada tahap berikutnya data akan digunakan untuk dua tujuan yang berbeda yaitu membuat sistem dengan menggunakan data hasil *stemming* dan *stopword* dan membuat sistem dengan menggunakan data tanpa *stemming* dan *stopword*. Dengan kata lain, data yang sama akan melalui tahap *stemming* dan *remove stopword* kemudian dengan data yang sama tidak melalui tahap *stemming* dan *remove stopword*.

Data yang akan melalui tahap *stemming* dan *remove stopword* data tersebut terlebih dahulu akan dipisah perkata sehingga menjadi *array of words*. Selanjutnya, *stemming* digunakan untuk mengubah kata berimbuhan menjadi kata dasar, lalu

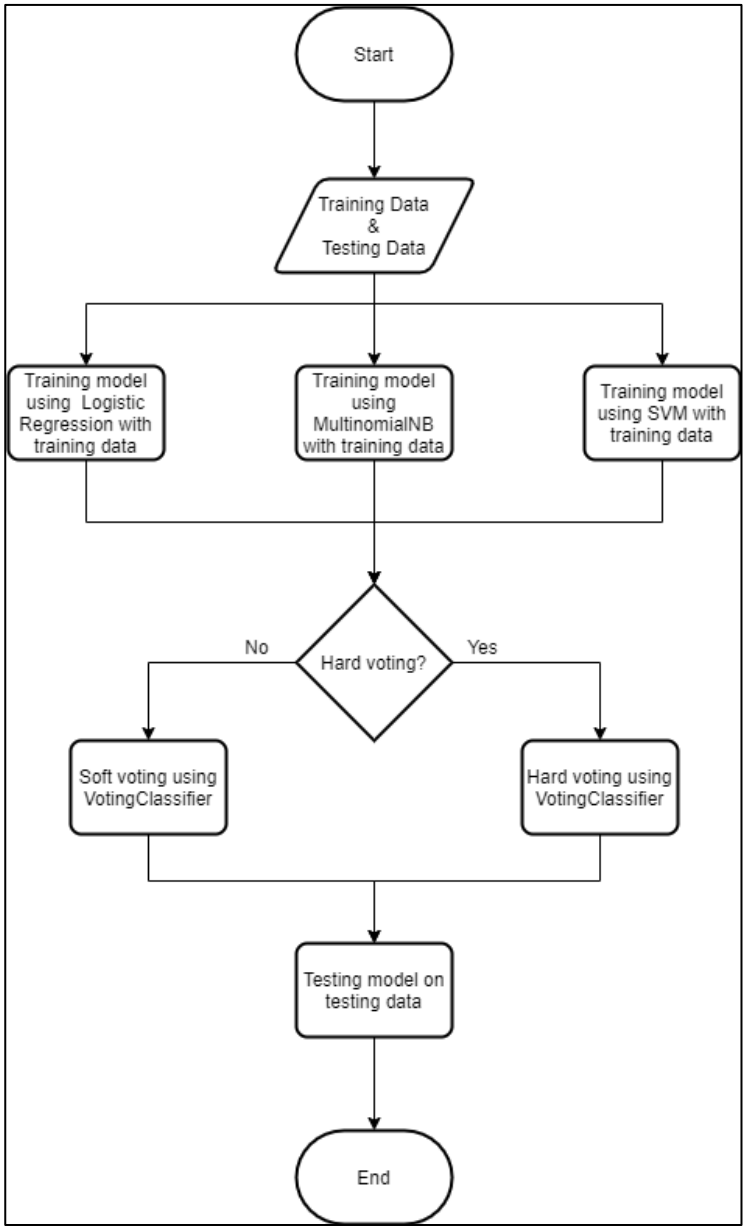
remove stopword digunakan untuk menghilangkan kata sambung, kata depan dan kata-kata yang tidak memiliki makna. Kemudian di tahap yang sama data yang sebelumnya berbentuk *array of words* akan digabungkan kembali menjadi kalimat. Proses ini menggunakan *library* python yaitu Sastrawi.

Setelah data tersebut dibersihkan, kemudian masuk ke tahap *split data for training and testing set*. Tahap ini bertujuan untuk memisahkan data awal menjadi data untuk data *training* dan data *testing* yang nantinya akan digunakan pada tahap *train & test ensemble learning*. Selanjutnya, data *feedback* baik itu pada *training* data maupun *testing* data akan diubah menjadi bentuk vektor agar dapat diproses oleh komputer. Proses ini menggunakan Term Frequency-Inverse Document Frequency (TF-IDF). Dalam penelitian kali ini TF-IDF *feature extraction* yang akan digunakan didapatkan dari *library* Scikit Learn yaitu TfidfVectorizer.

3.2.3 Flowchart Train and Test Ensemble Learning

Gambar 3.3 menunjukkan proses perancangan dan pelatihan model *ensemble learning*. Data yang sudah dipisahkan di dalam tahap *text preprocessing* menjadi *input* dalam tahap kali ini. *Training* dan *testing* data yang diguanakan sudah dalam bentuk vektor sehingga dapat diproses oleh komputer. *Training* data akan digunakan dalam proses pelatihan model pada setiap *base classifier*. Dalam penelitian kali ini *base classifier* yang digunakan adalah Logistic Regression, Multinomial Naïve Bayes, dan Support Vector Machine dengan menggunakan *library* dari Scikit Learn.

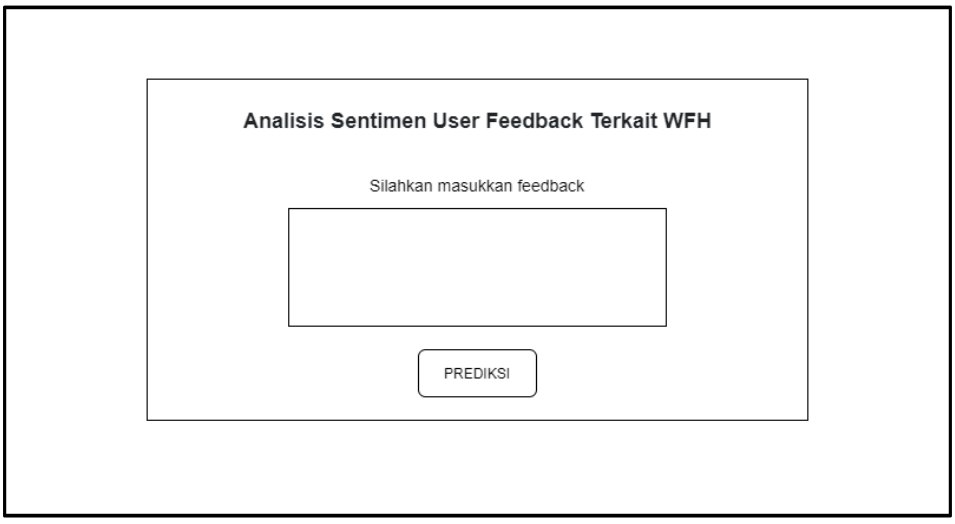
Tahapan selanjutnya yaitu setiap *classifier* akan melatih model dengan menggunakan *training* data dan *hyperparameter*. *Training* data yang digunakan sudah dalam bentuk vektor. *Hyperparameter* yang digunakan merupakan *hyperparameter* yang sudah optimal karena sudah didapatkan dari proses sebelumnya yaitu pada tahap *optimize hyperparameter for each base classifier*.



Gambar 3. 3 Flowchart Train & Test Ensemble Learning

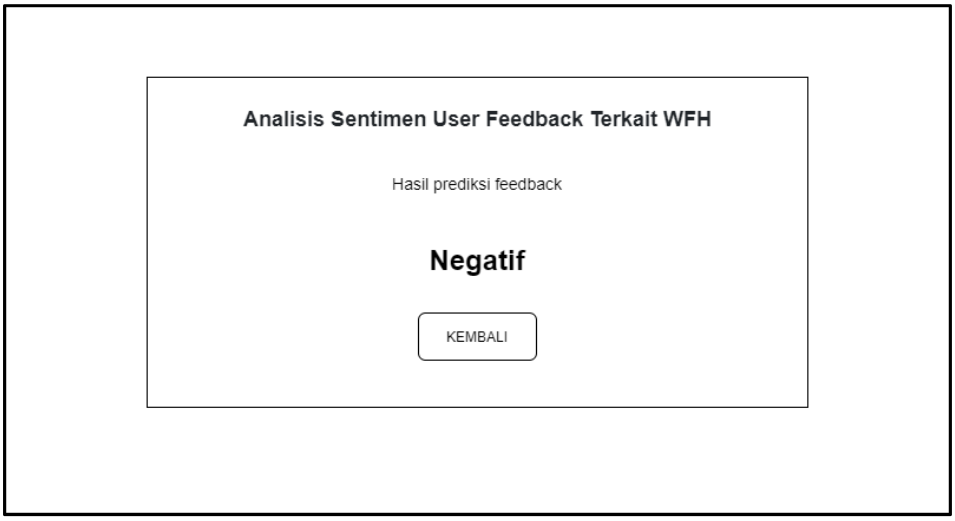
Hasil dari melatih model setiap *base classifier* akan menghasilkan model dari *base classifier* tersebut. Dalam tahap ini berarti terdapat 3 model yaitu model Logistic Regression, model Multinomial Naïve Bayes, dan model Support Vector Machine. Dalam melatih model *ensemble learning* akan menggunakan Voting Classifier yang didapatkan dari *library* Scikit Learn. Dalam melatih model Voting Classifier, menggunakan model-model dari *base classifier* untuk melakukan *voting*. Dalam penelitian kali ini akan menggunakan *soft voting* dan *hard voting*. *Soft voting* merupakan *voting* yang dilakukan dengan menghitung argmax dari jumlah prediksi probabilitas kelas atau target. *Hard voting* merupakan *voting* yang dilakukan dengan menghitung mayoritas *vote* dari *base classifier*. Model Voting Classifier yang telah selesai dilatih akan digunakan untuk *testing* model dengan menggunakan *testing* data untuk mendapatkan performa dari model tersebut.

3.2.4 Rancangan Tampilan Antarmuka



Gambar 3. 4 Kerangka Tampilan Utama

Antarmuka dibuat sebagai alat demonstrasi atau visualisasi dari hasil pembuatan model. Antarmuka dibuat berbasis *web* dengan menggunakan bahasa pemrograman python dan *web framework* Flask. Tampilan antarmuka pada Gambar 3.4 yaitu untuk memasukkan kalimat baru terkait *work from home* yang nantinya akan diprediksi untuk mengetahui apakah ada kesalahan atau dikenal dengan istilah *debugging*. Pada Gambar 3.5 yaitu untuk menampilkan hasil prediksi label dari kalimat yang sudah dimasukkan sebelumnya.



Gambar 3. 5 Kerangka Tampilan Hasil