**IMPLEMENTASI ALGORITMA RANDOM FOREST MENGGUNAKAN WORD EMBEDDING UNTUK ANALISIS SENTIMEN DENGAN PENERAPAN TRANSFER LEARNING**

**SKRIPSI**

**Diajukan sebagai salah satu syarat untuk memperoleh gelar**

**Sarjana Komputer (S.Kom.)**

****

**Ahsanul Qalbi Fajar Islami**

**00000015655**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS TEKNIK DAN INFORMATIKA**

**UNIVERSITAS MULTIMEDIA NUSANTARA**

**TANGERANG**

**2020**

# **HALAMAN PERSETUJUAN**

**Skripsi dengan judul**

**IMPLEMENTASI ALGORITMA RANDOM FOREST MENGGUNAKAN WORD EMBEDDING UNTUK ANALISIS SENTIMEN DENGAN PENERAPAN TRANSFER LEARNING**

oleh

Ahsanul Qalbi Fajar Islami

00000015655

**telah disetujui untuk diajukan pada**

**Sidang Ujian Skripsi Universitas Multimedia Nusantara**

Tangerang, 5 April 2020

Menyetujui,

Dosen Pembimbing 1 Dosen Pembimbing 2

Julio Christian Young, S.Kom., M.Kom. Andre Rusli, S.Kom., M.Sc.

Ketua Program Studi Informatika

Nunik Afriliana, S.Kom., MMSI

# **PERNYATAAN TIDAK MELAKUKAN PLAGIAT**

Dengan ini saya:

Nama : Ahsanul Qalbi Fajar Islami

NIM : 00000015655

Program Studi : Informatika

Fakultas : Teknik dan Informatika

Menyatakan bahwa Skripsi yang berjudul “**Implementasi Algoritma Random Forest Menggunakan Word Embedding Untuk Analisis Sentimen Dengan Penerapan Transfer Learning**” ini adalah karya ilmiah saya sendiri, bukan plagiat dari karya ilmiah yang ditulis oleh orang lain atau lembaga lain, dan semua karya ilmiah orang lain atau lembaga lain yang dirujuk dalam Skripsi ini telah disebutkan sumber kutipannya serta dicantumkan di Daftar Pustaka.

Jika di kemudian hari terbukti ditemukan kecurangan/ penyimpangan, baik dalam pelaksanaan Skripsi maupun dalam penulisan laporan Skripsi, saya bersedia menerima konsekuensi dinyatakan TIDAK LULUS untuk mata kuliah Skripsi yang telah saya tempuh.

Tangerang, 05 April 2020

Ahsanul Qalbi Fajar Islami

# **PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS**

Sebagai sivitas akademik Universitas Multimedia Nusantara, saya yang bertanda tangan di bawah ini:

Nama : Ahsanul Qalbi Fajar Islami

NIM : 00000015655

Program Studi : Informatika

Fakultas : Teknik dan Informatika

Jenis Karya : Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui dan memberikan izin kepada **Universitas Multimedia Nusantara** hak Bebas Royalti Non-eksklusif (*Non-exclusive Royalty-Free Right*) atas karya ilmiah saya yang berjudul:

**Implementasi Algoritma Random Forest Menggunakan Word Embedding Untuk Analisis Sentimen Dengan Penerapan Transfer Learning**

beserta perangkat yang diperlukan.

Dengan Hak Bebas Royalti Non-eksklusif ini, pihak **Universtias Multimedia Nusantara** berhak menyimpan, mengalihmedia atau *format*-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mendistribusi dan menampilkan atau mempublikasikan karya ilmiah saya di internet atau media lain untuk kepentingan akademis, tanpa perlu meminta izin dari saya maupun memberikan royalty kepada saya, selama tetap mencantumkan nama saya sebagai penulis karya ilmiah tersebut.

Demikian pernyataan ini saya buat dengan sebenarnya untuk dipergunakan sebagaimana mestinya.

Tangerang, 05 April 2020

Ahsanul Qalbi Fajar Islami

# **HALAMAN PERSEMBAHAN/ MOTO**

*[INSERT QUOTES]*

# **KATA PENGANTAR**

Puji syukur kepada Tuhan Yang Maha Esa karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan skripsi dan laporan skripsi yang berjudul “Implementasi Algoritma Random Forest Menggunakan Word Embedding Untuk Analisis Sentimen Dengan Penerapan Transfer Learning” dengan tepat waktu.

Proses penyelesaian laporan skripsi tidak terlepas dari bantuan berbagai pihak. Penulis mengucapkan terima kasih kepada:

1. Dr. Ninok Leksono, Rektor Universitas Multimedia Nusantara, yang memberi inspirasi bagi penulis untuk berprestasi,
2. Dr. Eng. Niki Prastomo, S.T., M.Sc. Dekan Fakultas Teknik dan Informatika Universitas Multimedia Nusantara,
3. Nunik Afriliana, S.Kom., MMSI., Ketua Program Studi Informatika Universitas Multimedia, yang menerima penulis dengan baik untuk berkonsultasi,
4. Julio Christian Young, S.Kom., M.Kom. dan Andre Rusli, S.Kom., M.Sc. yang membimbing pembuatan skripsi dan yang telah mengajar penulis tata cara menulis karya ilmiah dengan benar.
5. Ayah dan Ibu, serta keluarga besar yang telah memberikan dukungan moral dan spiritual selama proses penulisan skripsi,
6. Sahabat-sahabat mahasiswa terkasih yang telah membantu memberikan dukungan langsung maupun tidak langsung, bahkan ada yang menemani penulis selama proses penulisan skripsi.

Semoga laporan skripsi ini dapat bermanfaat, baik sebagai sumber informasi maupun sumber inspirasi, bagi para pembaca.

Tangerang, 05 April 2020

Ahsanul Qalbi Fajar Islami

**IMPLEMENTASI ALGORITMA RANDOM FOREST MENGGUNAKAN WORD EMBEDDING UNTUK ANALISIS SENTIMEN DENGAN PENERAPAN TRANSFER LEARNING**

# **ABSTRAK**

Kata Kunci: -----------------------------------------------------------------------------

**RANDOM FOREST ALGORITHM IMPLEMENTATION USING WORD EMBEDDING FOR SENTIMENT ANALYSIS WITH TRANSFERS LEARNING APPLICATION**

# **ABSTRACT**

Keywords: -----------------------------------------------------------------------------

# **DAFTAR ISI**

[**HALAMAN PERSETUJUAN** ii](#_Toc35598713)

[**PERNYATAAN TIDAK MELAKUKAN PLAGIAT** iii](#_Toc35598714)

[**PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS** iv](#_Toc35598715)

[**HALAMAN PERSEMBAHAN/ MOTO** v](#_Toc35598716)

[**KATA PENGANTAR** vi](#_Toc35598717)

[**ABSTRAK** viii](#_Toc35598718)

[**ABSTRACT** ix](#_Toc35598719)

[**DAFTAR ISI** x](#_Toc35598720)

[**DAFTAR GAMBAR** xii](#_Toc35598721)

[**DAFTAR TABEL** xiii](#_Toc35598722)

[**PENDAHULUAN** 1](#_Toc35598723)

[**1.1** **Latar Belakang Masalah** 1](#_Toc35598724)

[**1.2** **Rumusan Masalah** 3](#_Toc35598725)

[**1.3** **Batasan Masalah** 4](#_Toc35598726)

[**1.4** **Tujuan Penelitian** 4](#_Toc35598727)

[**1.5** **Manfaat Penelitian** 5](#_Toc35598728)

[**LANDASAN TEORI** 6](#_Toc35598729)

[**2.1** **Rekayasa Kebutuhan** **Kesalahan! Bookmark tidak ditentukan.**](#_Toc35598730)

[**2.2** **User Feedback** 6](#_Toc35598731)

[**2.3** **Analisa Sentimen** **Kesalahan! Bookmark tidak ditentukan.**](#_Toc35598732)

[**2.4** **Word Embedding** **Kesalahan! Bookmark tidak ditentukan.**](#_Toc35598733)

[**2.5** **FastText** **Kesalahan! Bookmark tidak ditentukan.**](#_Toc35598734)

[**2.6** **Decision Tree** **Kesalahan! Bookmark tidak ditentukan.**](#_Toc35598735)

[**2.7** **Random Forest Classifier** **Kesalahan! Bookmark tidak ditentukan.**](#_Toc35598736)

[**METODOLOGI PENELITIAN DAN PERANCANGAN SISTEM** 11](#_Toc35598737)

[**3.1** **Metodologi Penelitian** 11](#_Toc35598738)

[**3.1.1.** **Studi Literatur** 11](#_Toc35598739)

[**3.1.2.** **Pengumpulan Data** 11](#_Toc35598740)

[**3.1.3.** **Prosedur Penelitian** **Kesalahan! Bookmark tidak ditentukan.**](#_Toc35598741)

[**3.1.4.** **Penulisan Laporan** 11](#_Toc35598742)

[**3.2** **Perancangan Aplikasi** 12](#_Toc35598743)

[**3.2.1** **Flowchart Proses Klasifikasi** 13](#_Toc35598744)

[**3.2.1** **Flowchart Deployed Version** **Kesalahan! Bookmark tidak ditentukan.**](#_Toc35598745)

[**IMPLEMENTASI DAN ANALISIS** 33](#_Toc35598746)

[**4.1** **Spesifikasi Sistem** 33](#_Toc35598747)

[**4.2** **Implementasi Sistem** 34](#_Toc35598748)

[**6.2.1** **Hasil Implementasi CNN** **Kesalahan! Bookmark tidak ditentukan.**](#_Toc35598749)

[**6.2.2** **Hasil Tampilan Aplikasi** **Kesalahan! Bookmark tidak ditentukan.**](#_Toc35598750)

[**SIMPULAN DAN SARAN** 49](#_Toc35598751)

[**5.1** **Simpulan** 49](#_Toc35598752)

[**5.2** **Saran** 49](#_Toc35598753)

[**DAFTAR PUSTAKA** 50](#_Toc35598754)

[**DAFTAR RIWAYAT HIDUP** 51](#_Toc35598755)

# **DAFTAR GAMBAR**

[Gambar 3. 1 13](#_Toc35599633)

# **DAFTAR TABEL**

[Tabel 3. 1 Struktur Tabel Data **Kesalahan! Bookmark tidak ditentukan.**](#_Toc35599669)

**BAB** **I**

# **PENDAHULUAN**

## **Latar Belakang Masalah**

Analisis sentimen adalah salah satu bidang studi yang menganalisis pendapat, sentimen, evaluasi, penilaian, sikap, dan emosi seseorang terhadap suatu entitas tertentu seperti, jasa, organisasi, individu, masalah, peristiwa, topik dan atribut lainnya (Liu, 2012). Analisis sentimen dapat dimanfaatkan untuk melakukan ekstraksi opini-opini dari dokumen, komentar, sosial media, *review blog*, dan data-data lainnya. Analisis sentimen memliki beberapa metode, salah satu diantaranya adalah metode pendekatan *machine learning*. Analisis sentimen dengan pendekatan *machine learning* dapat menggunakan teknik klasifikasi.

Teknik klasifikasi adalah teknik dalam *data mining* untuk mengelompokkan data berdasarkan keterikatan data terhadap data sampel (Irvi Oktanisa, dan Afif Supianto 2018). Salah satu teknik klasifikasi adalah Random Forest. Random Forest adalah salah satu metode berbasis klasifikasi dan regresi dimana terdapat proses agregasi pohon keputusan (Dhawangkhara, dan Riksakomara 2017). Kelebihan dari algoritma Random Forest diantaranya adalah dapat menghindari *overfitting,* meminimalisir waktu *training data*, berjalan secara efisien pada data yang banyak, dan dapat mempertahankan akurasi walaupun sebagian data hilang.

Penelitian terkait mengenai klasifikasi teks dengan algoritma Random Forest dan TF-IDF telah banyak dilakukan salah satunya adalah penelitian oleh M Ali

Fauzi. (2018) yang mengklasifikasikan teks Bahasa Indonesia dengan algoritma Random Forest menggunakan fitur Bag-of-word dan metode pembobotan Term Frequency Inverse Document Fequency (TF.IDF). Disimpulkan bahwa penelitian tersebut memberikan hasil akurasi yang baik yaitu kinerja dengan skor out-of-bag (OOB) 0,829. Penelitian terkait dengan penggunaan TF-IDF selanjutnya adalah penelitian yang dilakukan oleh Yulius Denny dkk (2019) yang menerapkan algoritma TF-IDF untuk *text mining*. Penelitian tersebut menyimpulkan bahwa penggunaan TF-IDF merupakan metode yang tepat untuk digunakan dalam pencarian kata ditiap dokemen dan dapat membantu pengguna mendapatkan dokumen terkait sesuai dengan *query* yang telah di inputkan.

Penelitian terkait mengenai analisis sentimen dengan menggunakan teknik N-gram telah banyak dilakukan, salah satunya adalah penelitian oleh Wahyu Candra Inddhiarta (2017) yang melakukan analisis sentimen pemilihan kepala daerah Jakarta dengan menggunakan N-gram dan algoritma Naïve Bayes. Dari penelitian tersebut, disimpulkan bahwa berdasarkan ketiga jenis token N-gram yaitu unigram, bigram, dan trigram dengan metode Naïve Bayes, nilai akurasi tertinggi terdapat pada penggunaan bigram yaitu 0,823, menunjukan bahwa dengan menggunakan bigram ketepatan akurasi dari sistem lebih baik dari pada unigram dan trigram. Nilai presisi tertinggi juga terdapat pada penggunaan bigram yaitu 0,76. Namun pada nilai *recall* nilai tertinggi terdapat pada penggunaan trigram yaitu 0,898, sehingga disimpulkan penggunaan bigram dalam pengklasifikasian data lebih baik daripada menggunakan unigram atau trigram

Analisis sentimen dengan menggunakan *machine learning*, diperlukan *datasets* sebagai data *training* dan data *testing*. Metode pendekatan *machine learning* menghasilkan akurasi yang baik, akan tetapi performa dari klasifikasinya bergantung pada dataset yang digunakan untuk data *training* masalah ini berkaitan dengan *transfer learning. Transfer learning* adalah metode Deep Learning yang menerapkan pengetahuan atau *knowledge* dari domain yang berbeda namun terkait ke domain tujuan (Di Zhang dkk, 2019). Berdasarkan penelitian yang telah ada sebelumnya, penelitian ini mencoba untuk menganalisis seberapa besar pengaruh *transfer learning* terhadap akurasi klasifikasi analisis sentimen dengan menggunakan algoritma Random Forest dengan bantuan N-gram dan TF-IDF.

## **Rumusan Masalah**

Berdasarkan latar belakang yang telah dijabatkan sebelumnya, rumusan masalah dalam penelitian ini adalah sebagai berikut.

1. Bagaimana mengimplementasikan algoritma Random Forest dengan N- gram dan TF-IDF pada analisis sentimen?
2. Seberapa besar tingkat performa model yang dirancang dengan menggunakan metode *transfer learning* ?

## **Batasan Masalah**

Batasan masalah dalam penelitian ini dapat dijabarkan menjadi beberapa poin sebagai berikut.

1. *Datasets* terdiri dari kumpulan *review* dari pengguna pada produk yang dimiliki oleh perusahaan Amazon, dan Yelp.
2. *Datasets review* yang digunakan berupa teks Bahasa Inggris.
3. *Transfer learning* akan dilakukan dengan mengirimkan pengetahuan dari satu model Random Forest Classifier berupa fitur-fitur dan nilai kepentingan setiap fitur-fitur tersebut kepada model baru yang akan dibangun, sehingga model tersebut tidak menggunakan fitur yang dianggap tidak penting.

## **Tujuan Penelitian**

Berdasarkan rumusan masalah yang dijelaskan sebelumnya, penelitian ini memiliki tujuan sebagai berikut.

1. Menerapkan algoritma Random Forest dengan menggunakan metode Bigrams dan TF-IDF untuk klasifikasi analisis sentimen *review* oleh *user*.
2. Mengukur dan mengetahui seberapa besar hasil performa model jika diterapkan metode *transfer learning* pada analisis sentimen dengan algoritma Random Forest.

## **Manfaat Penelitian**

Hasil dari penelitian ini diharapkan dapat menerapkan metode *transfer learning* dengan algortima Random Forest pada analisis sentimen sehingga jika memberi pengaruh yang positif untuk performa model diharapkan dapat menanggulangi keterbatasan set data untuk membangun model yang serupa.

.

**BAB II**

# **LANDASAN TEORI**

**2.1 Analisis Sentimen**

Sentimen analisis adalah riset komputasional dari opini sentimen dan emosi yang diekspresikan secara tekstual (Ira Zulfa, dan Edi Winarko 2017). Dalam analisis sentimen, teks data yang didapatkan akan diklasifikasikan menjadi beberapa jenis seperti teks sentimen “positif”, “negatif”, dan “netral”. Pada penerapannya, analisis sentimen dimanfaatkan untuk memberikan nilai reputasi pada pelayanan pelanggan, produk perusahaan, dan reputasi seorang tokoh publik.

**2.2 Word Embedding**

*Word Embedding* adalah istilah yang digunakan untuk teknik mengubah sebuah kata menjadi sebuah *vector* atau *array* yang terdiri dari kumpulan angka. *Word Embedding* adalah sebuah pendekatan yang digunakan untuk merepresentasikan *vector* kata. *Word Embedding* merupakan pengembangan komputasi permodelan kata-kata yang sederhana seperti perhitungan menggunakan jumlah dan frekuensi kemunculan kata dalam sebuah dokumen (Yulius Denny dkk, 2019)

Contoh cara tradisional untuk membaca teks dan mengubah menjadi vektor angka, misalnya terdapat sebuah kalimat yakni “sore ini merupakan sore yang indah”. Langkah pertama adalah membuat sebuah *dictionary* yang berisi *list* dari seluruh kata yang *unique* atau tidak berulang, sehingga *dictionary* yang terbentuk adalah [“Sore”, “ini”, “merupakan”, “yang”, “indah”]. Langkah selanjutnya adalah menggunakan metode *one-shot encoding* yang akan mengeluarkan *output* vektor berupa vektor ‘1’ merepresentasikan tempat kata tersebut pada *list*, dan vektor ‘0’ untuk merepresentasikan tempat kata lainnya. Contoh vektor representasi pada kata ‘merupakan’ mengacu pada metode *one-shot encoding* adalah [0, 0, 1, 0 ,0].

**2.3 Teknik N-gram**

Menurut Wahyu Candra Indhiarta (2017) N-gram merupakan penggabungan kata sifat yang sering muncul untuk menunjukan suatu sentimen. Teknik N-gram memiliki jenis-jenisnya berupa Unigram (n = 1), Bigram (n = 2), Trigram (n = 3), dan seterusnya. Berikut merupakan contoh penerapan N-gram pada kalimat “Pembelajaran mesin merupakan salah satu mata kuliah jurusan informatika”.

Tabel 1. Jenis-jenis N-Gram

|  |  |
| --- | --- |
| **N-Gram** | **Hasil Penerapan** |
| Unigram | Pembelajaran, mesin, merupakan, salah, satu, mata, kuliah, jurusan, informatika |
| Bigram | Pembelajaran mesin, mesin merupakan, merupakan salah, salah satu, satu mata, mata kuliah, kuliah jurusan, jurusan informatika |
| Trigram | Pembelajaran mesin merupakan, mesin merupakan salah, merupakan salah satu, salah satu mata, satu mata kuliah, mata kuliah jurusan, kuliah jurusan informatika |

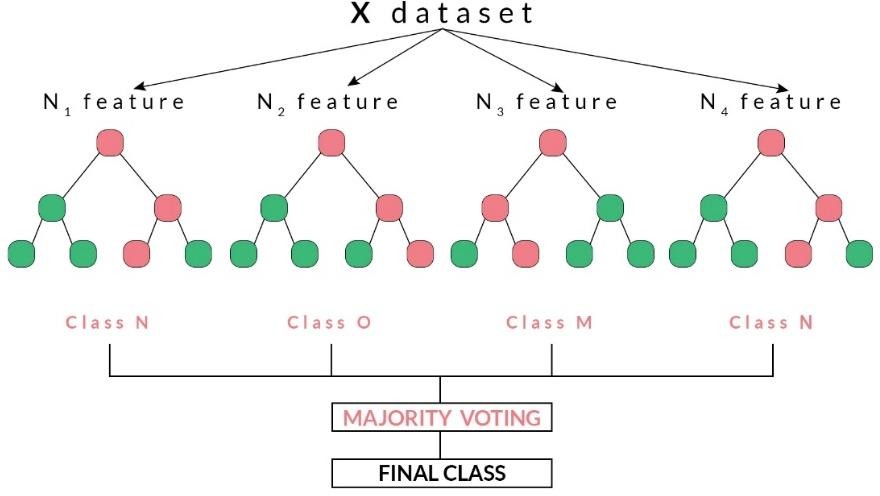
**2.4 Term Frequency-Inverse Document Frequency (TF-IDF)**

Term Frequency-Inverse Document Frequency atau TF-IDF merupakan algoritma yang berguna untuk mengetahui bobot setiap kata atau seberapa sering kata tersebut. Musfiroh Nurjannah, dkk. (2013) menyatakan bahwa metode ini menggabungkan perhitungan bobot, yaitu frekuensi kemunculan sebuah kata di dalam sebuah dokumen (Term Frequency) tertentu dan *inverse* frekuensi dokumen yang mengandung kata tersebut (Inverse Document Frequency).

Frekuensi kemunculan kata (Term Frequency) di dalam dokumen menunjukkan seberapa penting kata tersebut di dalam dokumen tersebut. Frekuensi dokumen yang mengandung kata tersebut (Inverse Document Frequency) menunjukkan seberapa umum kata tersebut. Sehingga bobot hubungan antara sebuah kata dan sebuah dokumen akan tinggi apabila frekuensi kata tersebut tinggi di dalam dokumen dan frekuensi keseluruhan dokumen yang mengandung kata tersebut yang rendah pada kumpulan dokumen Musfiroh Nurjannah, dkk. (2013).

**2.5 Random Forest Classifier**

Random Forest adalah salah satu teknik *machine learning* yang dapat digunakan untuk melakukan klasifikasi. Random Forest merupakan salah satu metode dalam *decision tree*. Menurut (Aditya Yanuar, 2018) *decision tree* atau pohon pengambil keputusan adalah sebuah diagram alir yang berbentuk seperti pohon yang memiliki sebuah *root node* yang digunakan untuk mengumpulkan data, Sebuah *inner node* yang berada pada *root node* yang berisi tentang pertanyaan tentang data dan sebuah *leaf node* yang digunakan untuk memecahkan masalah serta membuat keputusan. Random Forest memiliki beberapa *decision tree*, kemudian algoritma Random Forest mengambil keputusan berdasarkan hasil *voting* terbanyak dari semua *decision tree*. Kelebihan dari Random Forest adalah jika terdapat data yang hilang dengan jumlah tertentu, Random Forest masih dapat melakukan klasifikasi dengan akurasi yang stabil karena tidak bergantung dengan satu *decision tree* saja melainkan membandingkan data *voting decision tree* lainnya.

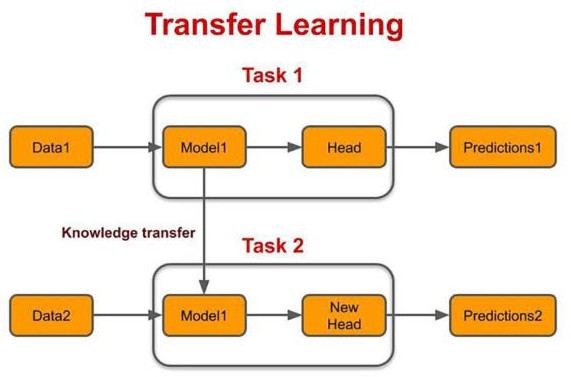


Gambar 1 Struktur Algortima Random Forest

(Shagufta, 2019)

**2.6 Transfer Learning**

*Transfer learning* adalah metode Deep Learning yang menerapkan pengetahuan atau *knowledge* dari domain yang berbeda namun terkait ke domain tujuan (Di Zhang dkk, 2019). Menurut (Reza Fuad, 2018) *transfer learning* bertujuan untuk mengurangi penggunaan set data dengan skala besar. Pada penelitian ini *transfer learning* akan dilakukan dengan membangun satu model dasar Random Forest Classifier dan mengirimkan pengetahuan dari model tersebut berupa *feature importance* kepada model baru yang akan dibangun, sehingga tidak menggunakan fitur yang dianggap tidak penting.



Gambar 2 Visualisasi gambaran *transfer learning*

(Pratik, 2019)

**BAB III**

# **METODOLOGI PENELITIAN DAN PERANCANGAN SISTEM**

## **Metodologi Penelitian**

Penelitian “Implementasi Algoritma Random Forest Menggunakan *Word Embedding* Untuk Analisis Sentimen Dengan Penerapan *Transfer Learning*” menggunakan beberapa tahap, tahap-tahap yang dilaksanakan antara lain adalah sebagai berikut.

### 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 Ngrams.

### Pengumpulan Data

Dataset didapatkan dari .. . Dataset yang didapatkan merupakan data ulasan pengguna kepada perusahaan Amazon dan Yelp. Dataset yang digunakan terdiri dari 10000 baris data berupa 5000 berlabel positif dan 5000 berlabel negative.

### Perancangan dan Pengembangan Program

Tahap perancangan program terdiri atas pembuatan alur kerja untuk penerapan *word embedding,* Random Forest Classifier, dan *transfer learning*, pembuatan struktur tabel, dan alur antar muka pada aplikasi web

1. *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.

1. Penulisan Laporan.

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

## **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 dan struktur tabel aplikasi web.

### **Flowchart**

Sebuah gambar berisi cuplikan layar

Deskripsi dibuat secara otomatisGambar 3.1 adalah *flowchart* atau diagram alir dari proses klasifikasi sentimen menggunakan *word embedding* 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

Sebuah gambar berisi cuplikan layar, jam

Deskripsi dibuat secara otomatis*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, manghilangkan *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*

Sebuah gambar berisi cuplikan layar

Deskripsi dibuat secara otomatis

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 ngrams. Dalam penelitian ini, bentuk ngrams yang digunakan adalah bigrams dan trigrams. Setelah data diubah dalam bentuk ngrams, 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*).

Sebuah gambar berisi teks

Deskripsi dibuat secara otomatis

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

(3.1)

Keterangan :

* N = Banyaknya baris data pada data *train*
* *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*.

Sebuah gambar berisi teks

Deskripsi dibuat secara otomatis

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. Didalam *loop* tersebuat, dilakukan perhitungan nilai TF-IDF non normalisasi pada *term* yang sedang menjadi index *looping* tersebut. Rumus TF-IDF yang digunakan adalah sebagai berikut

(3.2)

Keterangan :

* Val = Frekuensi kemunculan *term* pada suatu data

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

(3.3)

Keterangan :

* 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*. 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 Sebuah gambar berisi teks

Deskripsi dibuat secara otomatisperforma dari model Random Forest yang telah dibangun.

Gambar 3.6 Flowchart tahap proses pembuatan list *feature importance*

*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. Didalam *looping* tersebut akan dilakukan pengecekan apakah *term* tersebut mempunyai *feature importance* yang lebih besar dari 0, jika lebih besar maka *term* tersebut akan ditambahkan kedalam *list*. Setelah *looping* selesai mengakses semua *term* yang terdapat pada *dictionary* maka *list* akan di *return*.

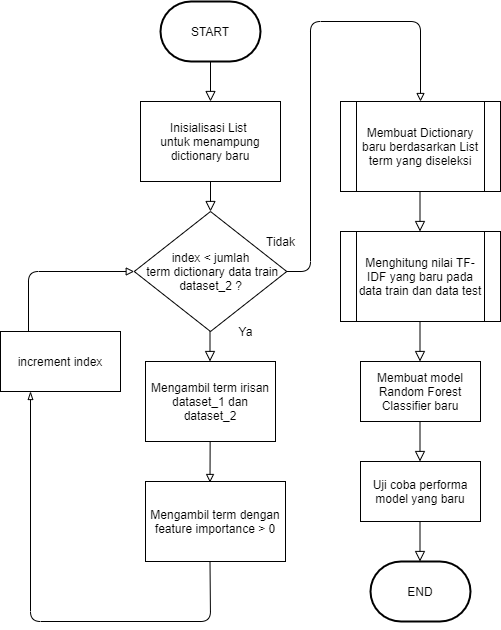
Sebuah gambar berisi cuplikan layar

Deskripsi dibuat secara otomatis

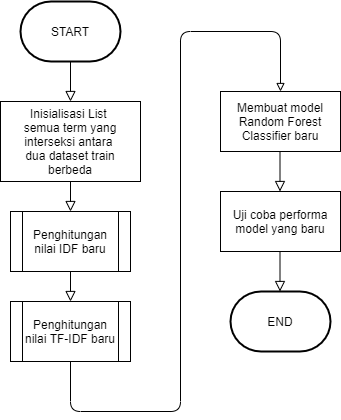
Gambar 3.7 Flowchart implementasi *transfer learning* pada interseksi *dataset*

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 berinterseksi saja antara *dataset* pertama dan *dataset* kedua, dan skenario kedua yaitu dengan meseleksi 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.

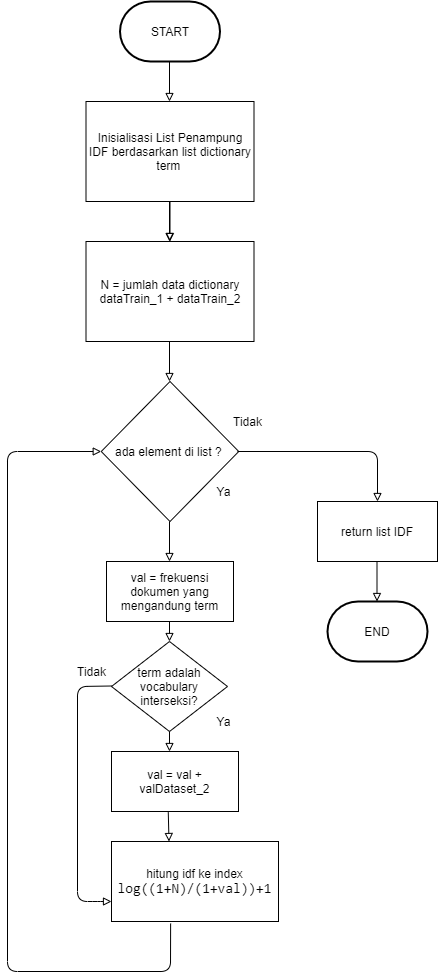
Langkah pertama yang dilakukan adalah inisialisasi *list* yang berisi semua *term* yang berinterseksi antara *dataset* pertama yang merupakan *dataset* yang telah di *training* menjadi *pretrained model* dan *dataset* kedua. Lalu langkah selanjutnya adalah membuat *dictionary term* berdasarkan *term* 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 / 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

Selain *trainsfer 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 Flowchart perhitungan nilai IDF *transfer learning*

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. Didalam *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.

Sebuah gambar berisi teks

Deskripsi dibuat secara otomatisGambar 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 menggunkan tabel.

Gambar 3.11 Flowchart Prediksi Input File pada aplikasi web

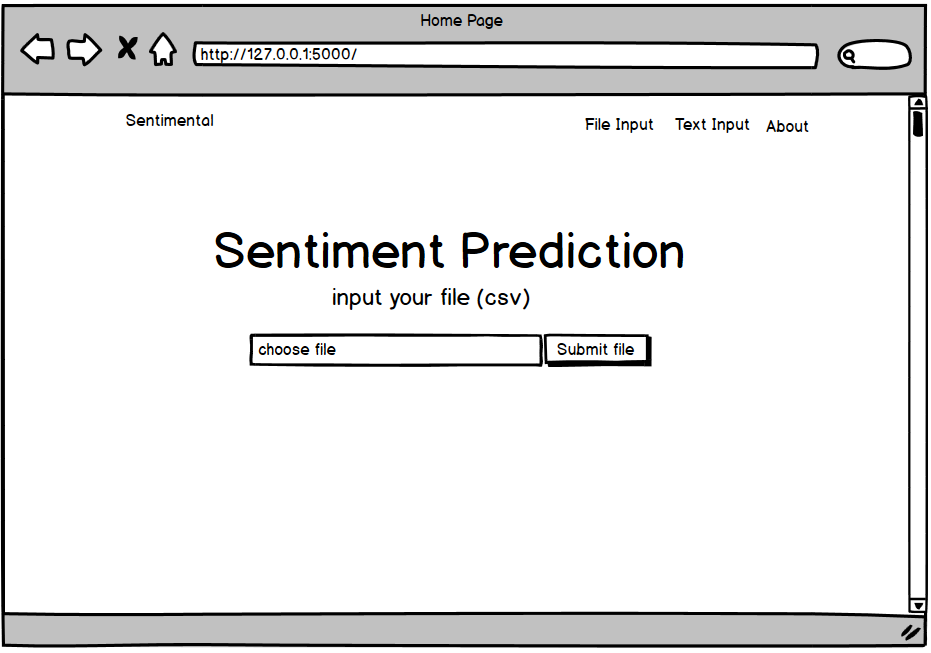
**Sebuah gambar berisi teks

Deskripsi dibuat secara otomatis**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

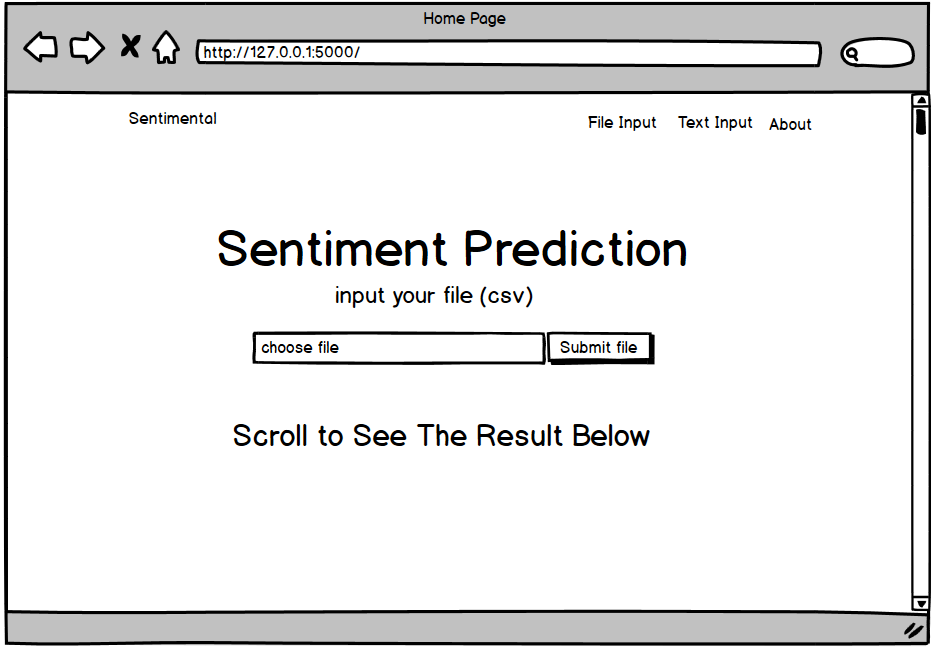
### **Rancangan Antarmuka**

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.

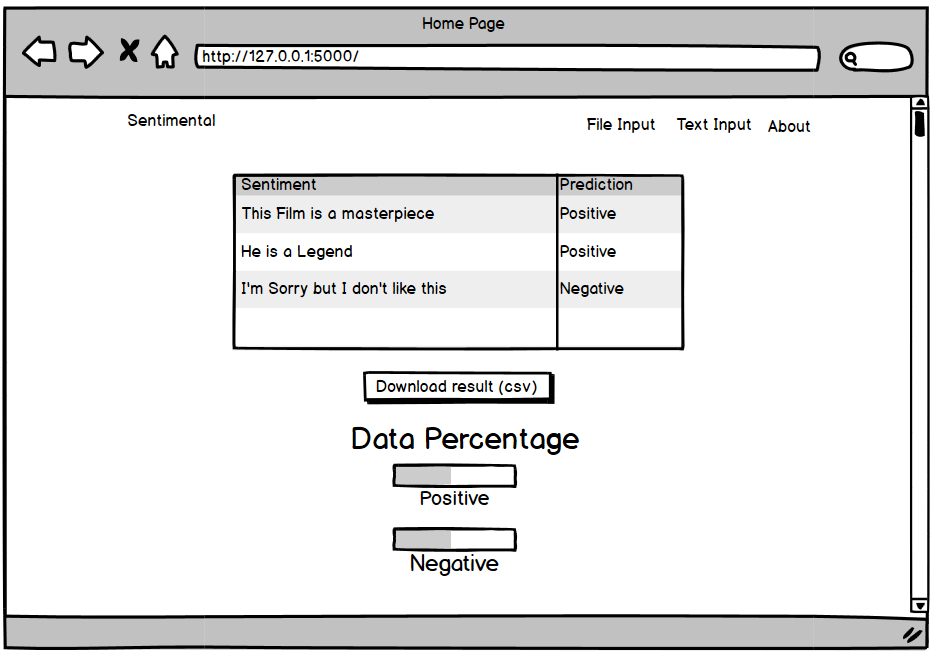


Gambar 3.13 Halaman *upload file*

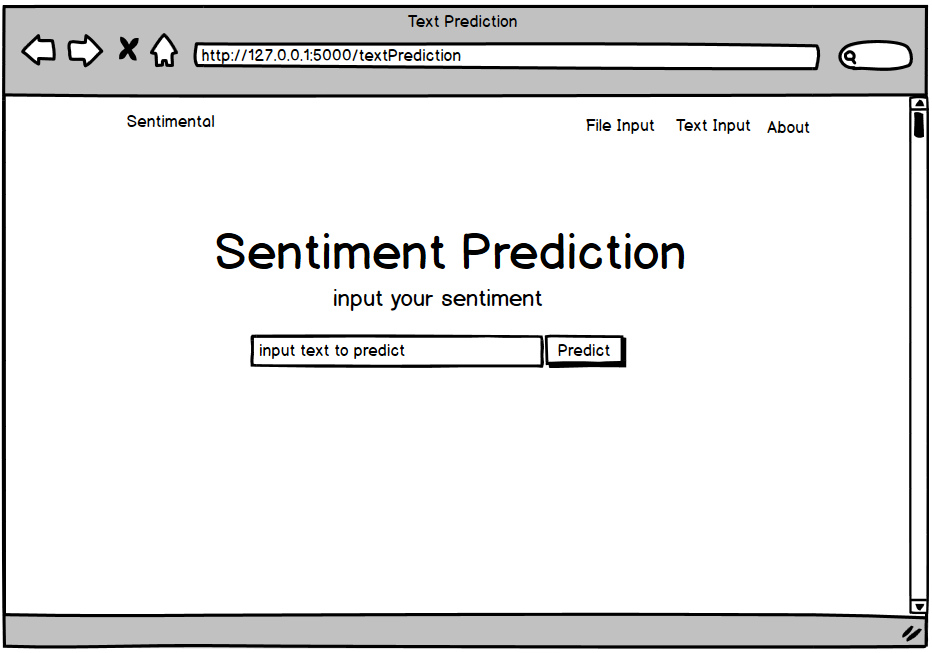
Pada Gambar 3.14 terdapat rancangan antarmuka pada halaman *upload file* jika pengguna telah berhasil *upload file csv*. Bebeda dengan antarmuka sebelumnya, pada antarmuka ini, terdapat informasi untuk pengguna agar *scroll* untuk melihat hasil dari prediksi *upload file* tersebut.

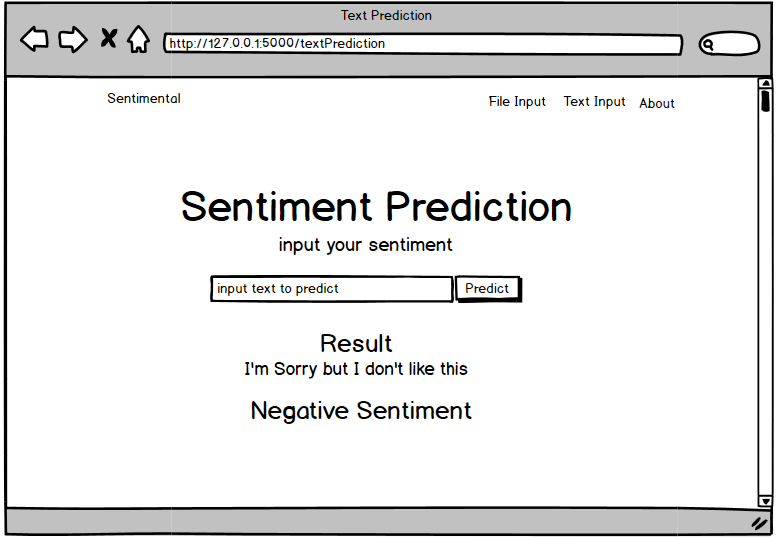


Gambar 3.14 Halaman *upload file* jika telah berhasil upload

Gambar 3.15 adalah rancangan antarmuka pada halaman *upload file* bagian bawah. Pada bagian ini, pengguna dapat melihat hasil dari prediksi data file yang terlah 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

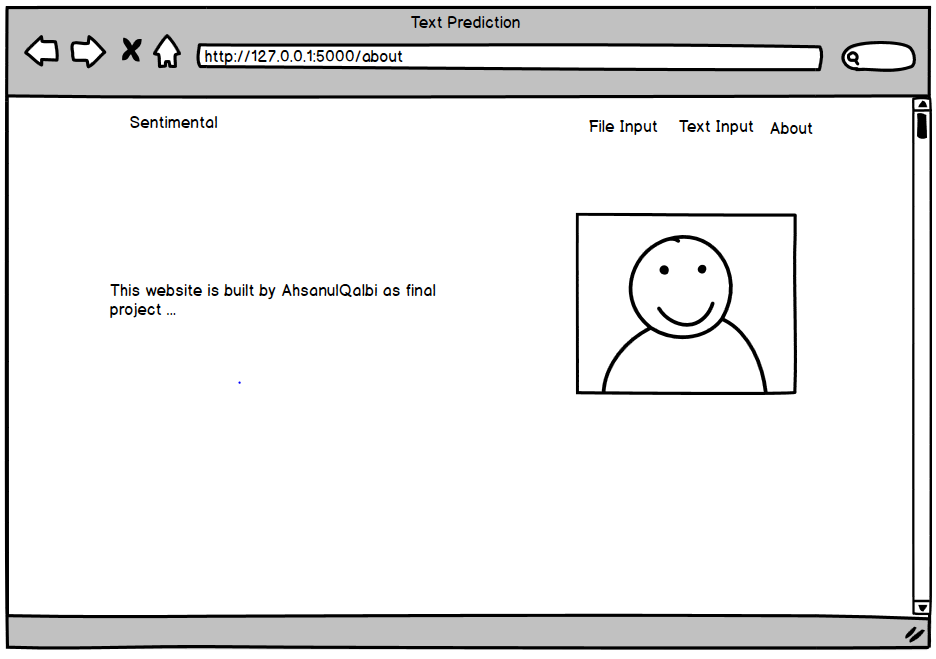
Pada Gambar 3.16 terdapat rancangan antarmuka untuk halaman *text input*. Di halaman ini pengguna dapat memasukkan *input* sentimen yang kemudian bisa diprediksi dan ditampilkan hasilnya secara langsung. Gambar 3.17 merupakan tampilan halaman *text input* jika pengguna selesai memasukan input. Pada tampilan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



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*

**BAB IV**

# **IMPLEMENTASI DAN ANALISIS**

## **Spesifikasi Sistem**

Pada pelaksanaan penelitian ini digunakan beberapa *tool* atau alat untuk mendukung pelaksanaan penelitian baik itu perangkat keras *(hardware)* maupun perangkat lunak *(software)*.Berikut merupakan komponen perangkat keras yang digunakan selama pengerjaan penelitian :

1. Laptop : MSI GL63 8SD
2. Processor : Intel Core 17-8750H CPU
3. Graphic Processing Unit : Nvidia GeForce GTX 1660 Ti
4. Memory : 16 GB RAM

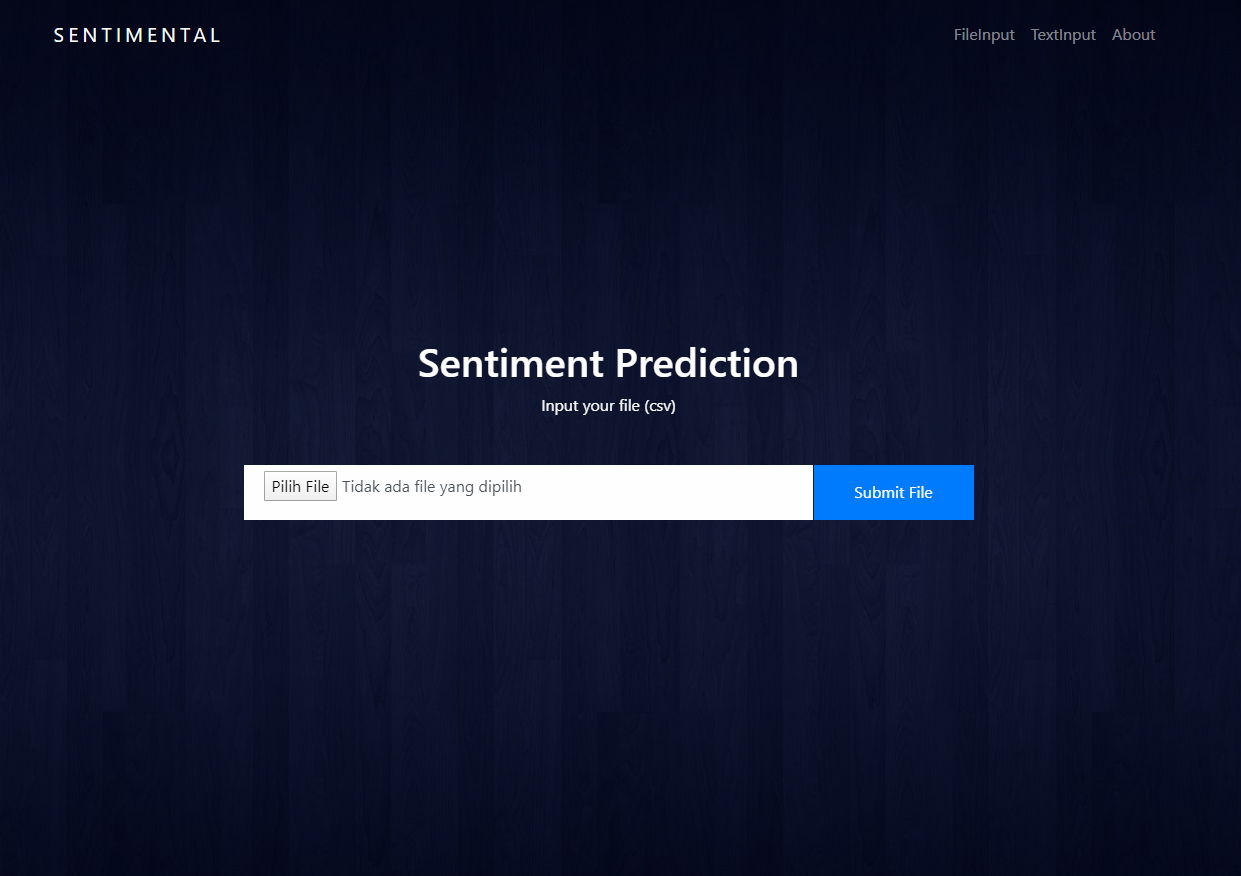
Adapun *software* yang digunakan selama penelitian antara lain :

1. Python 3.7.4
2. Anaconda Promt
3. Jupyter Notebook
4. Visual Studio Code
5. Framework Flask

## **Implementasi**

Dalam proses implementasi penelitian, aplikasi *web* dibangun dengan menggunakan bahasa pemrograman Python dengan bantuan *framework* Flask. Berikut adalah tampilan aplikasi dan potongan kode hasil implementasi aplikasi.

**4.2.1 Implementasi Aplikasi Web**

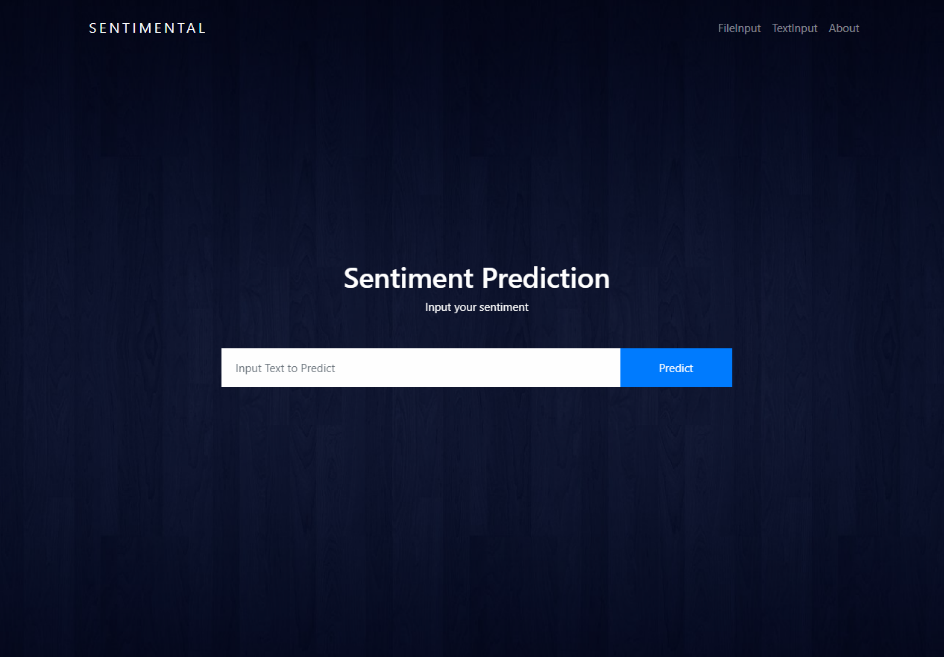
Gambar 4.1 merupakan halaman utama aplikasi *web* yaitu halaman *file upload*. Pada halaman utama ini memiliki *bar* navigasi berupa nama aplikasi, tombol navigasi halaman utama (*file upload*), tombol navigasi halaman *text upload* dan tombol navigasi halaman *about*. Ketiga tombol navigasi tersebut berfungsi untuk berpindah ke halaman *web* lainnya, *bar* navigasi tersebut juga terdapat di setiap halaman lainnya. Pada halaman utama ini juga memiliki *form* yang berguna agar pengguna dapat *upload file csv* yang berisi data sentiment yang ingin diprediksi.

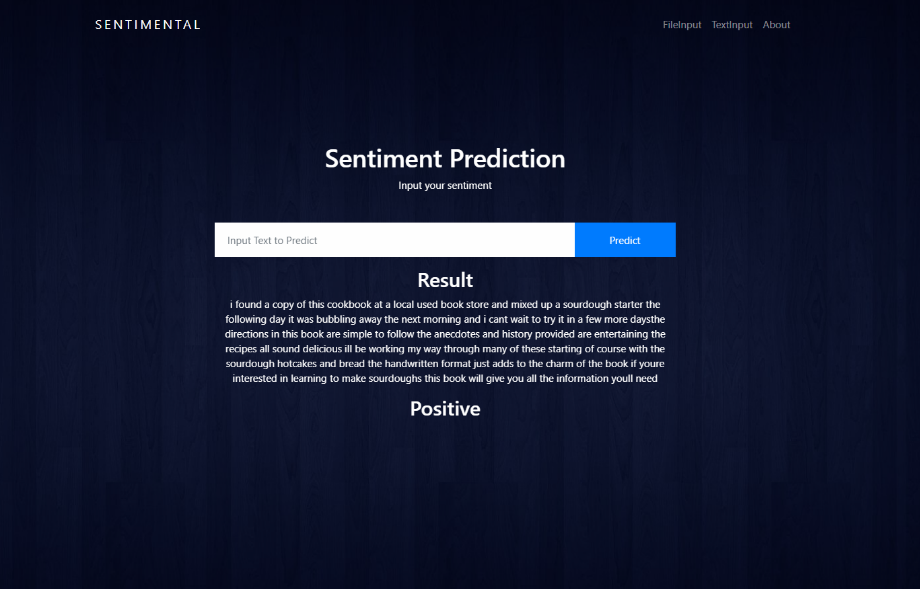
Gambar 4.1 Halaman utama aplikasi *web*

 Gambar 4.2 merupakan bagian bawah halaman utama aplikasi *web* ketika pengguna telah selesai *submit file csv*. Pada bagian ini terdapat hasil dari prediksi sentimen yang telah di-*upload* oleh pengguna. Hasil prediksi tersebut ditampilkan dengan menggunakan tabel yang terdiri atas kolom sentimen dan hasil prediksi. Tabel tersebut dibangun menggunakan *template* dari DataTables. Selanjutnya terdapat tombol *download result file*, yang berguna agar pengguna dapat mengunduh hasil dari prediksi sentimen dalam bentuk *file csv*. Selanjutya juga terdapat tampilan persentase label data hasil dari prediksi.

Gambar 4.2 Halaman utama web hasil prediksi file

Gambar 4.3 merupakan tampilan halaman input teks pada aplikasi *web*. Pada halaman ini terdapat *form* berupa *input* teks. *Form* tersebut berguna untuk pengguna dapat memasukan sentimen secara langsung dengan *input* teks. Jika pengguna telah melakukan *submit* pada *form* teks, maka akan ditampilkan hasil prediksi seperti yang digambarkan oleh Gambar 4.4.



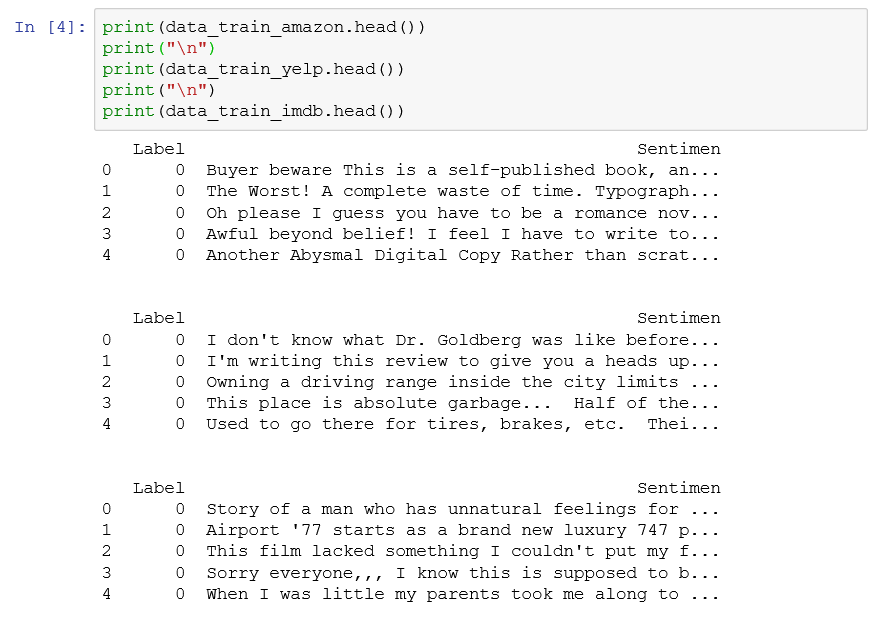
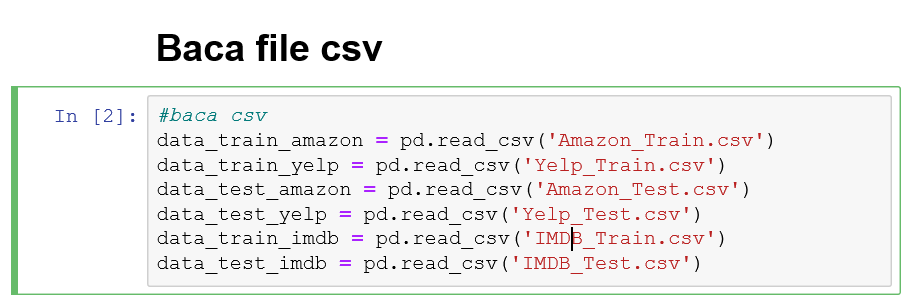


Gambar 4.4 Halaman input teks hasil prediksi

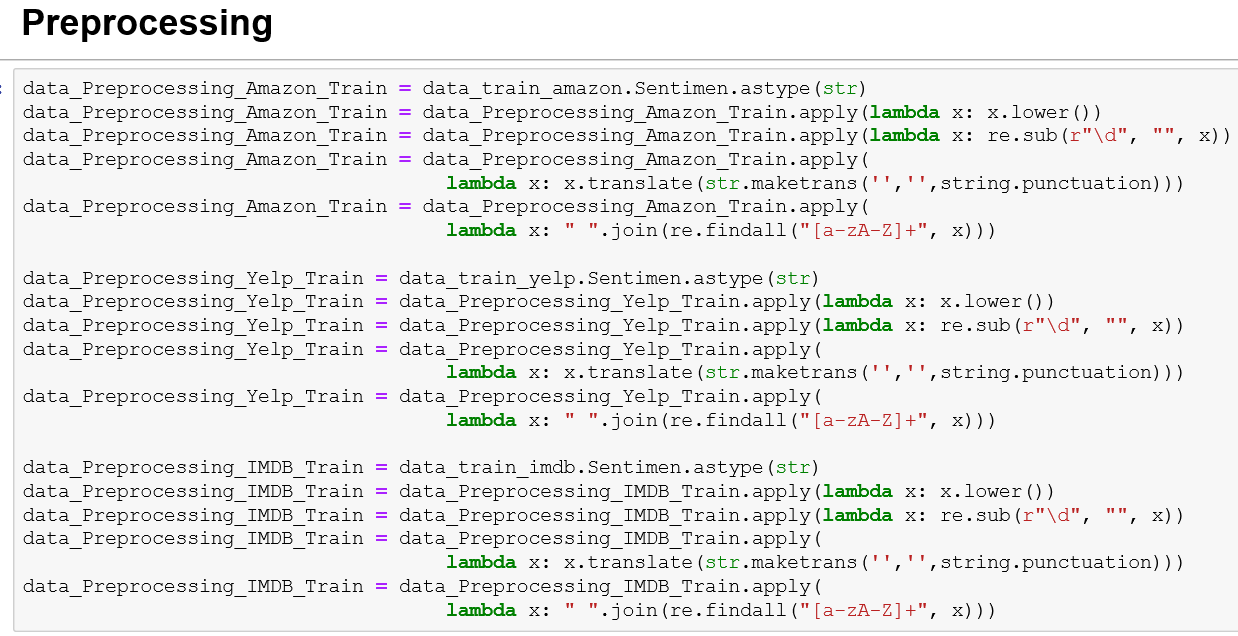
Gambar 4.3 Halaman input teks

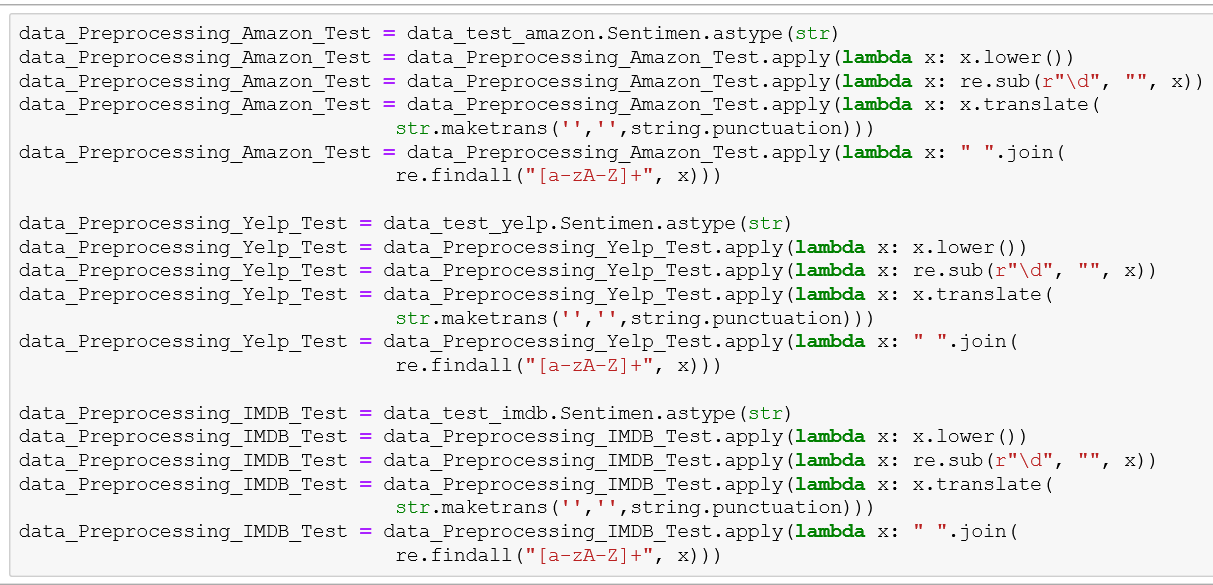
**4.2.2 Implementasi Algoritma**

Gambar 4.5 menunjukkan proses pengambilan data *train* dan data *test* dari *file* CSV untuk *dataset* Amazon, IMDB, dan Yelp. Tiap *file* tersebut terdiri dari kolom sentimen dan kolom label. Kolom sentimen terdiri dari data sentimen *review* tiap *dataset* yang belum diproses, kolom label terdiri dari label-label setiap sentimen yang terdiri dari angka satu (1) dan angka (0) yang merepresentasikan sentimen positif atau sentimen negatif.



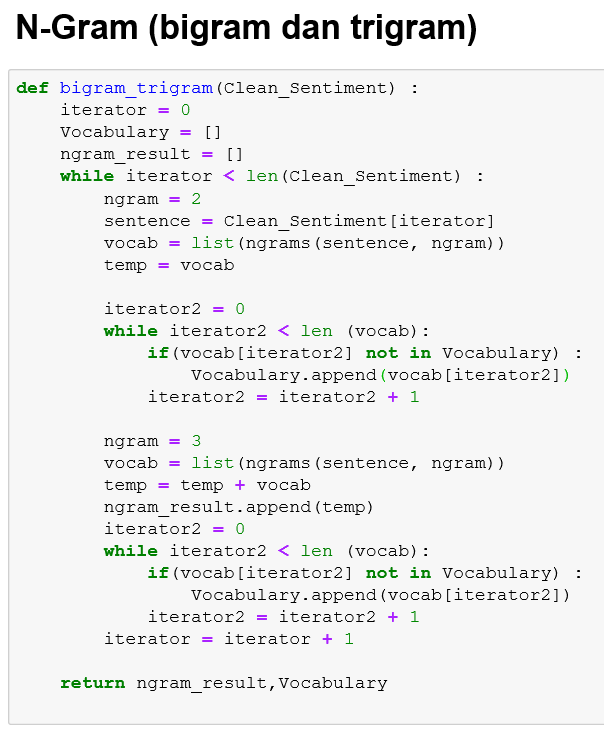
Gambar 4.5 Proses membaca *dataset*

 Gambar 4.6 merupakan proses *preprocessing* data *train* dan data *test* pada masing-masing dataset. Proses ini terdiri dari konversi ke *lowercase*, menghilangkan angka, menghilangkan tanda baca, dan menghilangkan karakter non alfabet dimana proses tersebut menggunakan *library* regular expression dan string. *Preprocessing* ini dilakukan pada setiap baris data pada masing-masing *dataset* dengan menggunakan fungsi *apply* dan fungsi *lamda* yang berguna untuk mengubah semua data pada *dataframe pandas* menjadi data dengan nilai yang diinginkan.

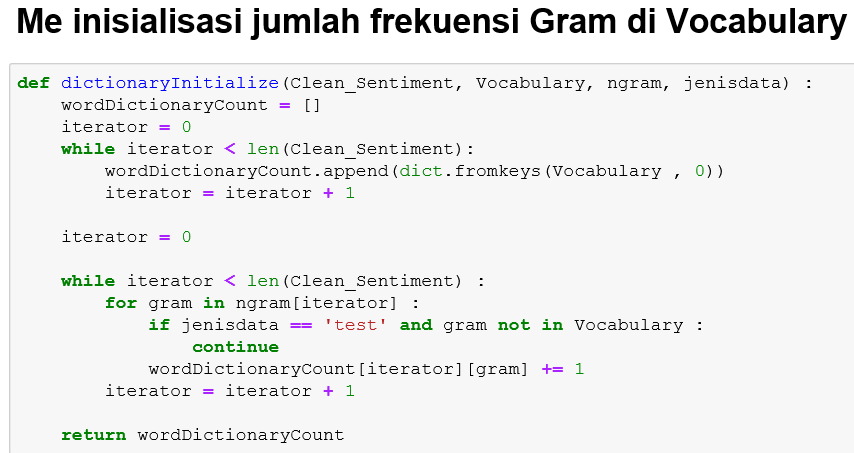


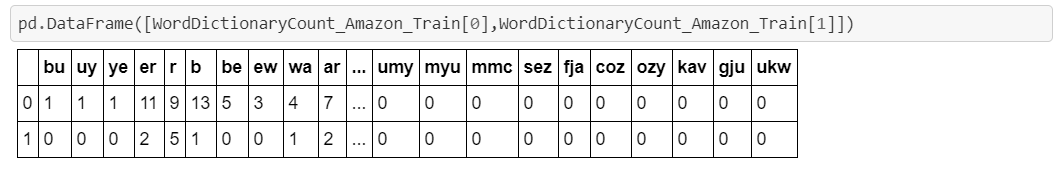
Gambar 4.5 Tahap preprocessing dataset

Gambar 4.6 menunjukkan sebuah fungsi yang digunakan untuk mengubah data sentimen yang telah diproses menjadi bentuk *bigrams* dan *trigrams*. Dalam penggunaan fungsi tersebut, akan mengembalikan *list* ngram (*bigrams* dan *trigrams*) untuk setiap data pada *dataset* yang di-*passing* dan mengembalikan *vocabulary* atau semua *term* ngram (*bigrams* dan *trigrams*) unik yang terdapat pada *dataset* tersebut. Dalam penerapannya, *list term* ngram didapatkan dengan menggunakan *library* ngrams dari nltk.



Gambar 4.6 pengubahan sentimen dalam bentuk ngrams

Gambar 4.7 menunjukkan sebuah fungsi yang digunakan untuk inisialisasi sebuah *dictionary* untuk data *train* dan data *test* untuk setiap *dataset*. Kolom dari *dictionary* tersebut merupakan *vocabulary* atau semua *term* yang terdapat dari suatu *dataset* dan baris dari *dictionary* tersebut merupakan frekuensi kemunculan setiap *term* pada suatu baris data. Gambar 4.8 menunjukan dua contoh hasil dari inisialisasi *dictionary* yang didapatkan dari proses pada Gambar 4.7 pada penerapan *data train* Amazon.



Gambar 4.8 Hasil inisialisasi *dictionary* ngram

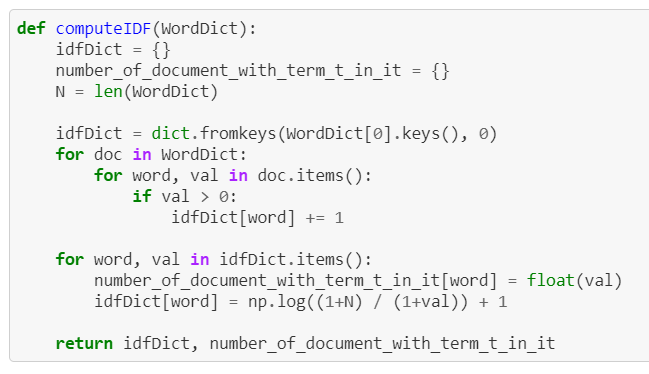
Gambar 4.7 fungsi inisialisasi dictionary ngram

Gambar 4.9 menunjukkan sebuah fungsi yang digunakan untuk menghitung nilai inverse document frequency atau IDF untuk setiap term pada data *train* dan data *test*. Langkah pertama yang dilakukan adalah membuat sebuah *list* yang berfungsi untuk menampung jumlah data atau dokumen yang memiliki *term t* di dalamnya. Lalu dihitunglah nilai IDF untuk setiap *term* dengan menggunakan rumus sebagai berikut :

(4.1)

Keterangan :

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

Fungsi tersebut akan mengembalikan nilai berupa *list* nilai IDF untuk setiap *term* dan list yang menampung jumlah data atau dokumen yang memiliki *term t* di dalamnya yang akan digunakan untuk melakukan *transfer learning* nilai IDF.

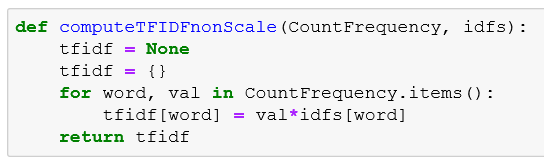
Gambar 4.9 Fungsi untuk inisialisasi nilai IDF

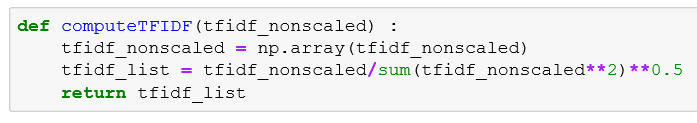
Gambar 4.10 menunjukkan sebuah fungsi yang digunakan untuk menghitung nilai **Term Frequency- Inverse Document Frequency (TF-IDF) sebelum normalisasi untuk setiap *term* pada setiap baris data suatu *dataset*. Nilai TF-IDF non normalisasi pada suatu *term* dihitung dengan cara mengalikan nilai frekuensi kemunculan *term* tersebut pada suatu baris data dengan nilai IDF *term* tersebut yang telah dihitung pada fungsi sebelumnya. Fungsi ini mengembalikan nilai hasil perhitungan TF-IDF tersebut. Langkah selanjutnya adalah menormalisasi nilai TF-IDF tersebut. Gambar 4.11 menunjukkan sebuah fungsi yang digunakan untuk menormalisasi nilai TF-IDF pada setiap *term*. Untuk menormalisasi nilai TF-IDF, digunakan rumus sebagai berikut :**

(4.2)

Keterangan :

* v = nilai TF-IDF non normalisasi

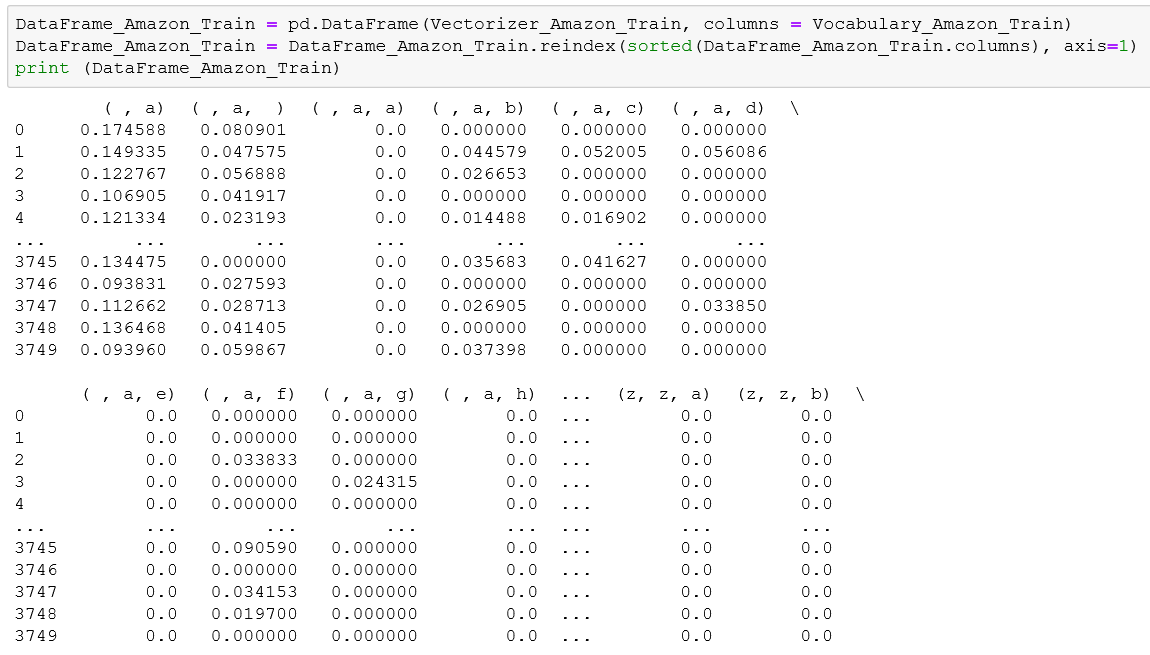




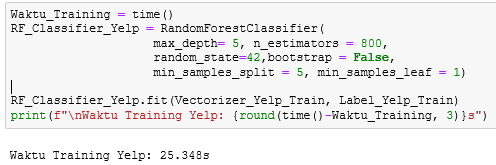
Gambar 4.11 Fungsi untuk normalisasi nilai TF-IDF

Gambar 4.10 Fungsi untuk inisialisasi nilai TF-IDF

Hasil dari normalisasi nilai TF-IDF ditunjukkan pada Gambar 4.12 berupa *dataframe* dari *dictionary* nilai TF-IDF untuk setiap *term* pada setiap baris data. Gambar 4.13 menunjukkan proses deklarasi model klasifikasi dengan menggunakan Random Forest Classifier dari scikit-learn dengan *hyperparameter* :

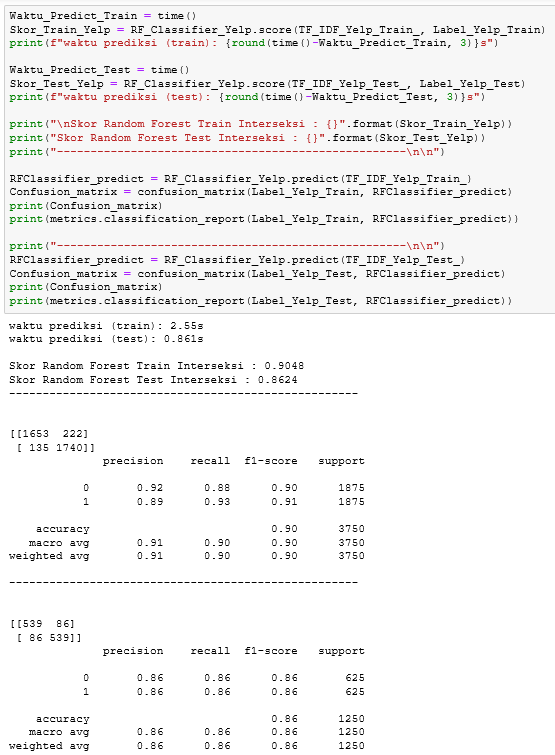
* max\_depth, yaitu kedalaman maksimal dari *tree* yang digunakan adalah lima tingkat.
* n\_estimators, yaitu banyaknya *tree* yang digunakan di dalam *forest* adalah sebanyak 800 *tree*.
* min\_samples\_split*,* yaitu jumlah sampel minimal yang diperlukan untuk membelah *node* internal adalah sebanyak lima sampel.
* min\_samples\_leaf, yaitu jumlah minimum sampel yang diperlukan untuk berada pada *leaf node* sebanyak satu sampel.
* bootstrap, yang bernilai *false* yang berarti seluruh *data* digunakan untuk membuat setiap *tree*.
* random\_state, yaitu indeks untuk mengontrol pengacakan angka yang di *generate* adalah 42.

Gambar 4.12 Hasil dari normalisasi nilai TF-IDF



Gambar 4.13 Inisialisasi model klasifikasi Random Forest

Gambar 4.14 menggambarkan proses yang digunakan untuk mengukur performa dari model Random Forest Classifier yang sudah dibangun sebelumnya, pada contoh ini, diambil uji coba pada model *dataset* Yelp. Performa yang di uji antara lain adalah waktu yang diperlukan untuk membangun model klasifikasi, waktu prediksi data, skor akurasi, precision, recall, dan f1-score. Untuk menghitung skor akurasi digunakan fitur pengembalian *score* pada Random Forest Classifier, sedangkan untuk menghitung nilai precision, recall, dan f1-score digunakan fitur *metrics classification report* dari scikit learn.

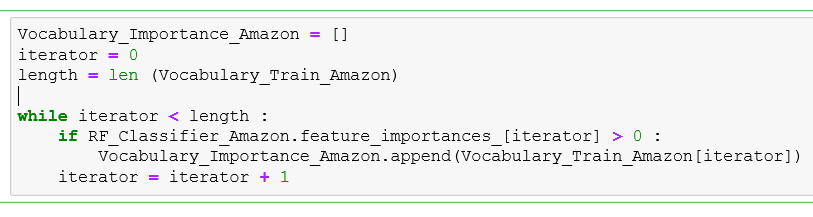


Gambar 4.14 Uji performa model klasifikasi

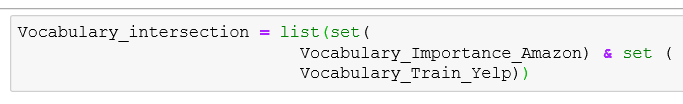
**4.3 Uji Coba Metode Transfer Learning**

Pada penelitian ini dilakukan metode *transfer learning* pada informasi atau *knowledge* yang diterima pada model yang telah dibangun dengan menggunakan suatu *dataset* akan diterapkan untuk membangun model klasifikasi lainnya dengan menggunakan *dataset* selanjutnya. Pada penerapannya, *transfer learning* akan dilakukan dengan menggunakan informasi dari *list feature­\_importances* yang didapatkan dari fitur pengembalian Random Forest Classifier. *Transfer learning* juga diterapkan untuk mengirim informasi berupa nilai IDF pada suatu model dan menerapkan pada model berikutnya. Untuk masing-masing penerapan *transfer learning* akan dilakukan dengan dua kali uji coba yakni dengan menggunakan informasi model klasifikasi dari *dataset* Amazon yang diterapkan kepada *dataset* Yelp, dan menggunakan informasi dari model klasifikasi *transfer learing* tersebut untuk digunakan pada *dataset* IMDB.

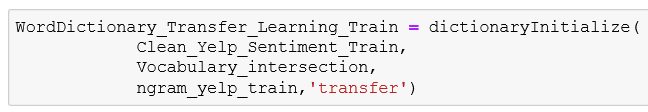
**4.3.1 Uji Coba Transfer Learning Feature Importances**

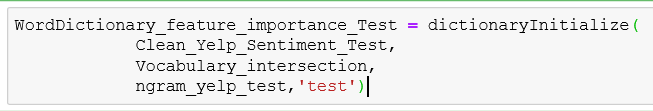
Untuk melakukan *transfer learning* *feature importance* diperlukan *list* yang terdiri dari daftar *term-term* yang memiliki nilai *feature importance* yang lebih dari nol. Gambar 4.15 menunjukkan langkah untuk inisialisasi *list* tersebut. *List* tersebut diinisialisasi dengan cara *looping* setiap *term* yang terdapat pada *vocabulary* masing-masing *dataset*, dan mendapatkan *term* yang memiliki *feature importance* yang lebih dari nol.

Gambar 4.15 Fungsi untuk inisialisai *list term feature importance*

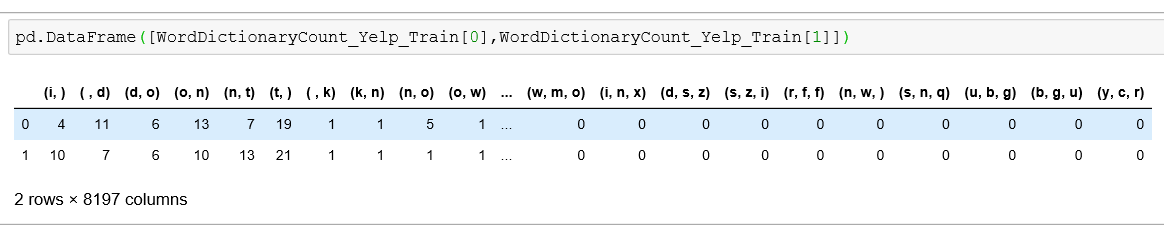
Skenario pertama dilakukan dengan hanya mengambil *term-term* yang interseksi pada kedua *dataset*. Uji coba pertama dalam penerapan *transfer learning* *feature importance*  skenario ini adalah dengan menggunakan model klasifikasi dengan *dataset* Amazon dan menerapkannya pada *dataset* Yelp yang akan dijadikan model klasifikasi. Gambar 4.16 menunjukkan langkah untuk mendapatkan *vocabulary* interseksi antara *dataset* Amazon dan Yelp. Setelah *list vocabulary term* interseksi didapatkan, maka langkah selanjutnya adalah membuat *dictionary* dan menghitung frekuensi masing-masing *term* disetiap baris data. Gambar 4.17 merupakan proses untuk membuat *dictionary* frekuensi tiap *term* pada data *train* dan data *test* Yelp dengan *vocabulary* interseksi yang memanggil fungsi dictionaryInitialize yang telah ditunjukkan sebelumnya pada Gambar 4.7.

Gambar 4.16 Proses pembuatan *list term* interseksi



Gambar 4.18 menunjukkan hasil dari *dictionary* *term* pada data *train* untuk membangun model klasifikasi dari dataset Yelp sebelum dan setelah diberlakukan pemilihan *term* secara interseksi pada data *train* Yelp dan data *train* Amazon yang memiliki nilai *feature\_importances* lebih besar dari nol. Sebelum diberlakukannya seleksi *term*, jumlah fitur *term* pada data *train* Yelp adalah sebanyak 8197 fitur, namun setelah diberlakukan seleksi *term* menjadi 2822 fitur. Hal ini menunjukkan terdapat sebanyak 5375 fitur yang memiliki nilai *feature importances* lebih kecil atau sama dengan nol.

Gambar 4.17 Proses pembuatan *dictionary* *train test* baru





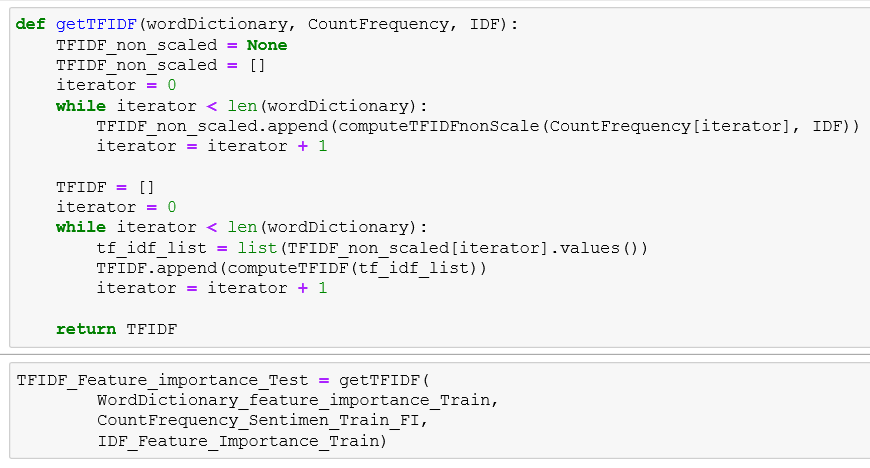
Gambar 4.18 Hasil pemilihan *term* dengan interseksi *feature importance*

Setelah membuat *dictionary* baru, langkah selanjutnya adalah menghitung nilai IDF berdasarkan *list term* pada *dictionary* baru. Gambar 4.19 merupakan proses untuk membuat *list* IDF tersebut dengan memanggil fungsi yang telah diinisialisasi yang ditunjukkan pada Gambar 4.9. Fungsi tersebut akan mengembalikan nilai IDF yang baru.



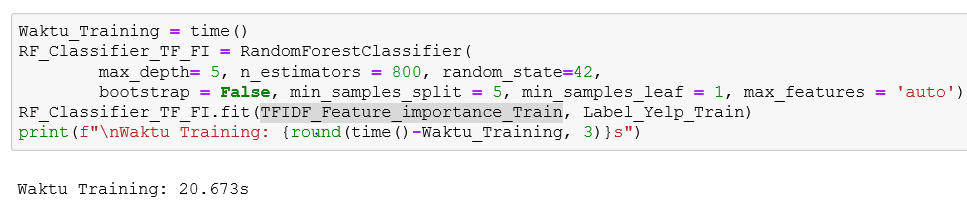
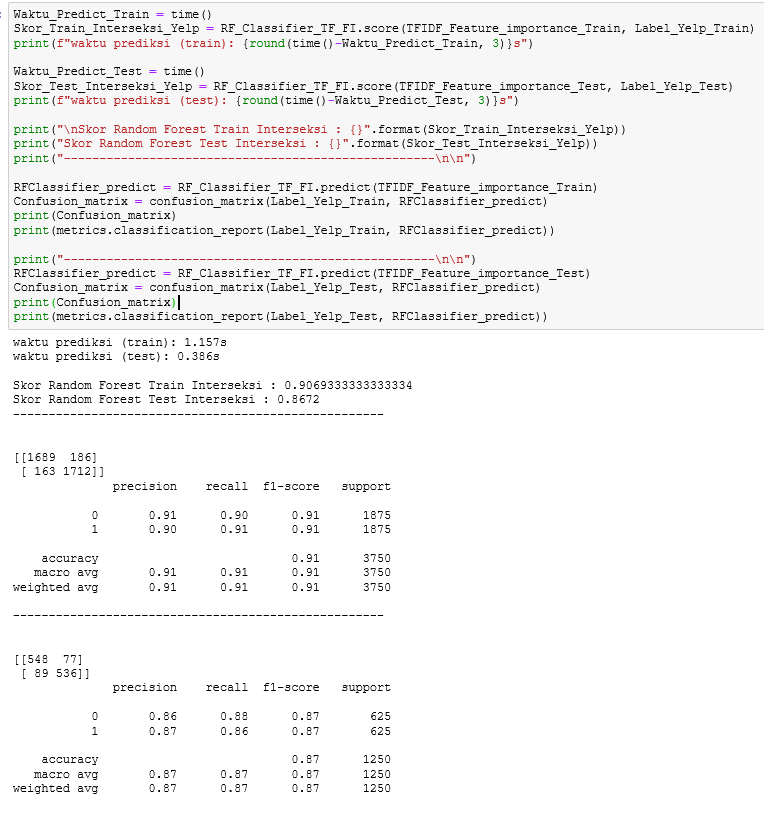
Gambar 4.19 Proses perhitungan nilai IDF *feature importances*

Setelah membuat *list* nilai IDF baru, langkah selanjutnya adalah menghitung nilai TF-IDF berdasarkan *list* nilai IDF yang baru tersebut. Gambar 4.20 merupakan proses untuk membuat *list* TF-IDF tersebut dengan memanggil fungsi yang telah diinisialisasi yang ditunjukkan pada Gambar 4.10 dan Gambar 4.11.



Gambar 4.20 Proses perhitungan nilai TF-IDF *feature importance*

Setelah membuat *list* nilai TF-IDF baru, langkah selanjutnya adalah membuat model klasifikasi dari Random Forest Classifier berdasarkan *list* nilai TF-IDF yang baru. Gambar 4.21 menunjukkan proses deklarasi model klasifikasi dengan menggunakan Random Forest Classifier berdasarkan nilai TF-IDF tersebut. Model klasifikasi yang dibangun menggunakan *hyperparameter* yang sama dengan sebelumnya. Dengan waktu training sekitar 20 detik, lebih cepat sekitar 5 detik dibandingkan sebelum diterapkan penyisihan *term* yang dapat dilihat pada Gambar 4.13 Hasil uji performa yang dihasilkan oleh model yang baru ditunjukkan pada Gambar 4.22.

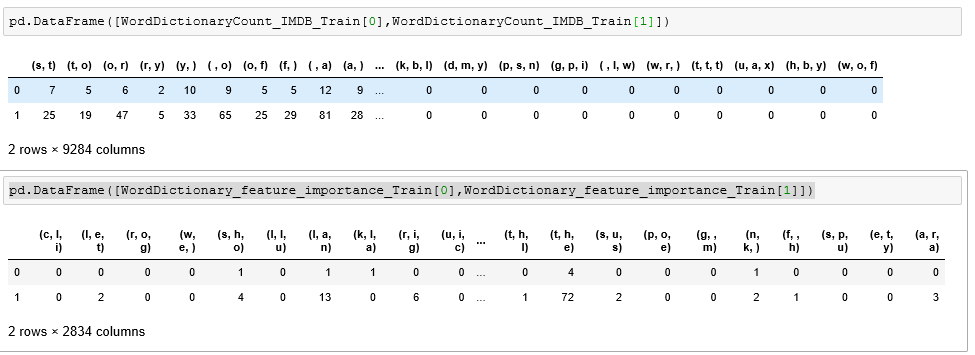
 

Gambar 4.20 Inisialisasi model klasifikasi

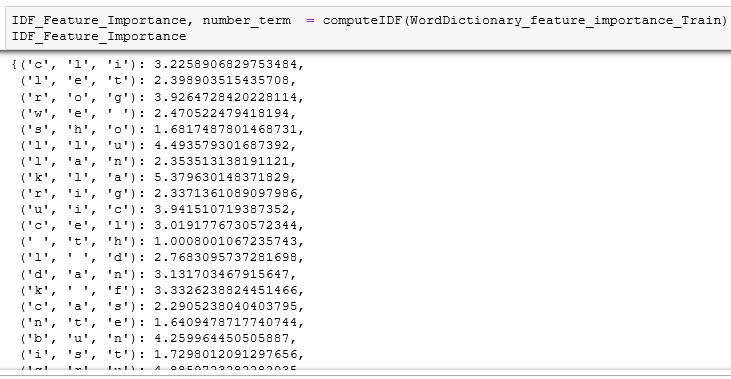
Gambar 4.22 Hasil uji performa model

Uji coba kedua dalam penerapan *transfer learning* *feature importance*  skenario ini adalah dengan menggunakan informasi model klasifikasi dengan *dataset* Amazon dan menerapkannya pada *dataset* IMDB yang akan dijadikan model klasifikasi. Proses pengerjaannya sama seperti uji coba pertama pada skenario ini. Prosesnya yaitu mencari *vocabulary* interseksi antara kedua *dataset* (Amazon dan IMDB), membuat *dictionary* dan menghitung frekuensi masing-masing *term* disetiap baris data berdasarkan *vocabulary* yang baru, menghitung nilai IDF baru pada masing-masing, menghitung nilai TF-IDF, dan membuat model klasifikasi yang baru.

Gambar 4.23 menunjukkan hasil dari *dictionary* *term* pada data *train* untuk membangun model klasifikasi dari dataset IMDB sebelum dan setelah diberlakukan pemilihan *term* secara interseksi pada data *train* IMDB dan data *train* Amazon yang memiliki nilai *feature\_importances* lebih besar dari nol. Sebelum diberlakukannya seleksi *term*, jumlah fitur *term* pada data *train* Yelp adalah sebanyak 9284 fitur, namun setelah diberlakukan seleksi *term* menjadi 2834 fitur. Hal ini menunjukkan terdapat sebanyak 6450 fitur yang memiliki nilai *feature importances* lebih kecil atau sama dengan nol. Gambar 4.24 hasil dari pembuatan nilai IDF yang baru berdasarkan *dictionary term* yang baru.

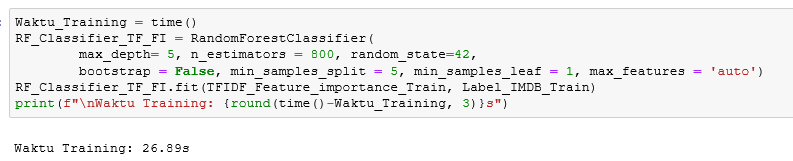
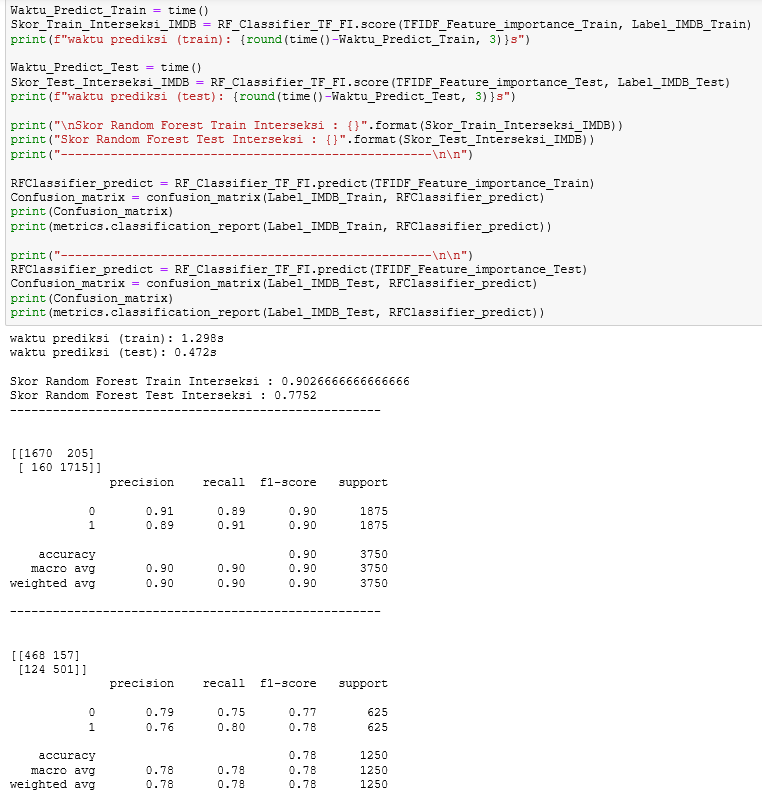


Gambar 4.23 Hasil pemilihan *term* dengan interseksi *feature importance*



Gambar 4.24 Proses perhitungan nilai IDF *feature importances*

Sama dengan uji coba pertama, langkah selanjutnya adalah menghitung kembali nilai TF-IDF dan inisialisasi model klasifikasi Random Forest Classifier dengan *list* nilai TF-IDF tersebut yang ditunjukkan oleh Gambar 4.25. Dengan waktu training sekitar 26 detik, lebih cepat sekitar 5 detik dibandingkan sebelum diterapkan penyisihan *term* yang dapat dilihat pada Gambar 4.26. Hasil uji coba performa model klasifikasi baru ditunjukkan pada Gambar 4.27.

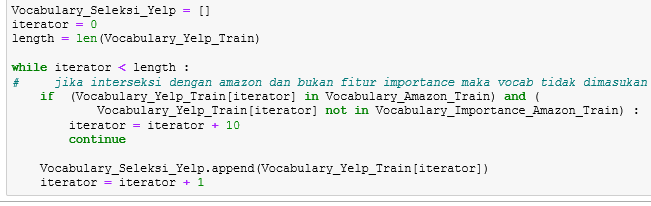


Gambar 4.27 Hasil uji performa model

Gambar 4.26 Inisialisasi model klasifikasi sebelum diterapkan seleksi *term*

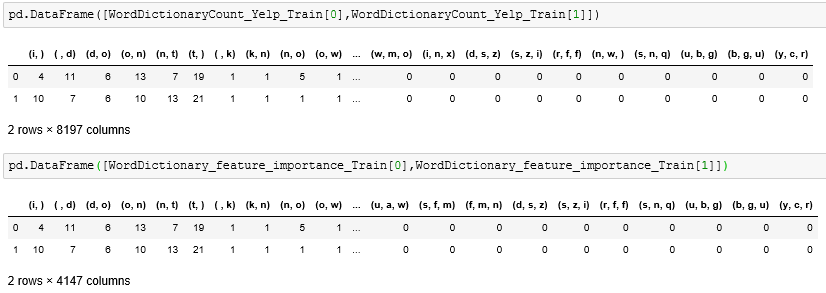
Gambar 4.25 Inisialisasi model klasifikasi setelah diterapkan seleksi *term*

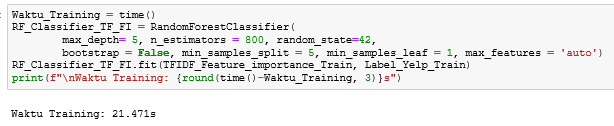
Skenario selanjutnya yaitu skenario kedua dilakukan dengan menyeleksi *term* dengan tidak memasukan *term* yang memiliki nilai *feature importances* kurang dari atau sama dengan nol pada *dataset* yang akan dijadikan model klasifikasi. Sama seperti sebelumnya, uji coba pertama dalam penerapan *transfer learning* *feature importance*  skenario ini adalah dengan menggunakan model klasifikasi dengan *dataset* Amazon dan menerapkannya pada *dataset* Yelp yang akan dijadikan model klasifikasi. Gambar 4.28 menunjukkan langkah untuk menyeleksi *term* pada *dataset* Yelp berdasarkan informasi *feature importances* dari *dataset* Amazon. Proses tersebut akan *looping* tiap *term* dalam *vocabulary dataset* Yelp dan mengecek *term* tersebut jika terdapat pada *vocabulary* Amazon dan bukan merupakan *term* yang memiliki nilai *feature importance* lebih dari nol, maka tidak dimasukan ke dalam *list vocabulary* yang baru. Setelah inisialisasi *vocabulary* yang baru, maka proses selanjutnya adalah sama seperti skenario sebelumnya yaitu mulai dari membuat *dictionary* baru, hingga menguji performa model klasifikasi yang baru.



Gambar 4.28 Inisialisasi model klasifikasi sebelum diterapkan seleksi *term*

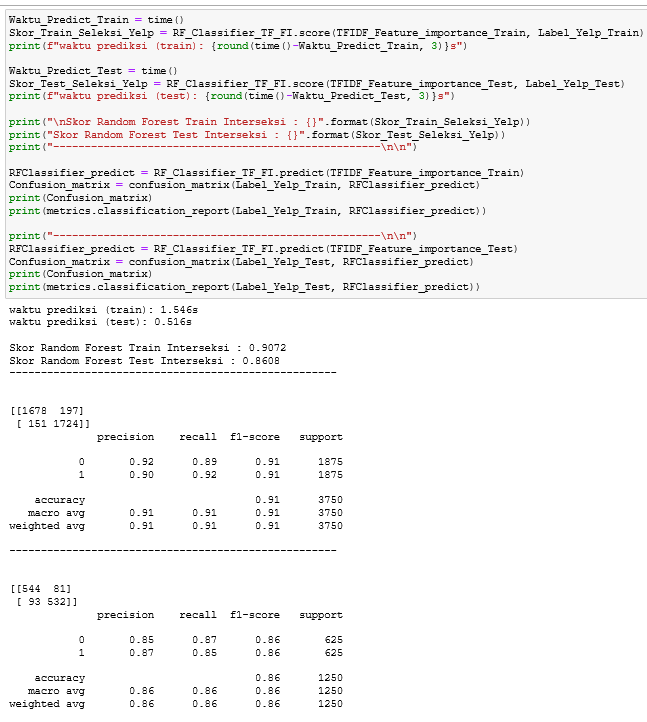
Gambar 4.29 menunjukkan hasil dari *dictionary* *term* pada data *train* untuk membangun model klasifikasi dari dataset Yelp sebelum dan setelah diberlakukan seleksi *term*. Hal ini menunjukkan bahwa terdapat sebanyak 4050 *term* diseleksi dan tidak dipergunakan. Gambar 4.30 menunjukkan perbandingan waktu yang diperlukan untuk inisialisasi model dengan menggunakan *dataset* Yelp sebelum diseleksi dan sesudah diseleksi. Gambar 4.31 merupakan hasil uji coba performa model klasifikasi yang baru.





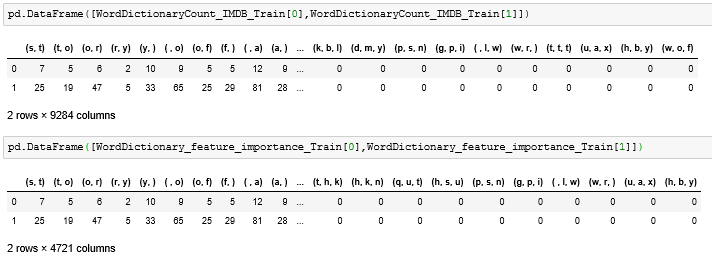
Gambar 4.30 Inisialisasi model klasifikasi sebelum dan setelah diterapkan seleksi *term*

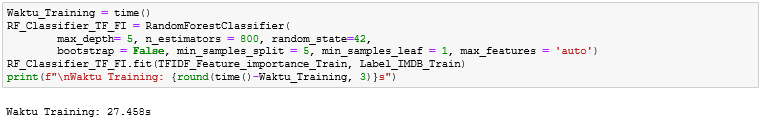
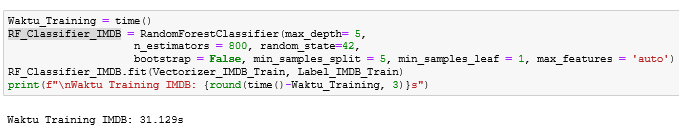
Gambar 4.29 Hasil pemilihan *term* dengan seleksi *feature importance*



Gambar 4.31 Hasil uji performa model

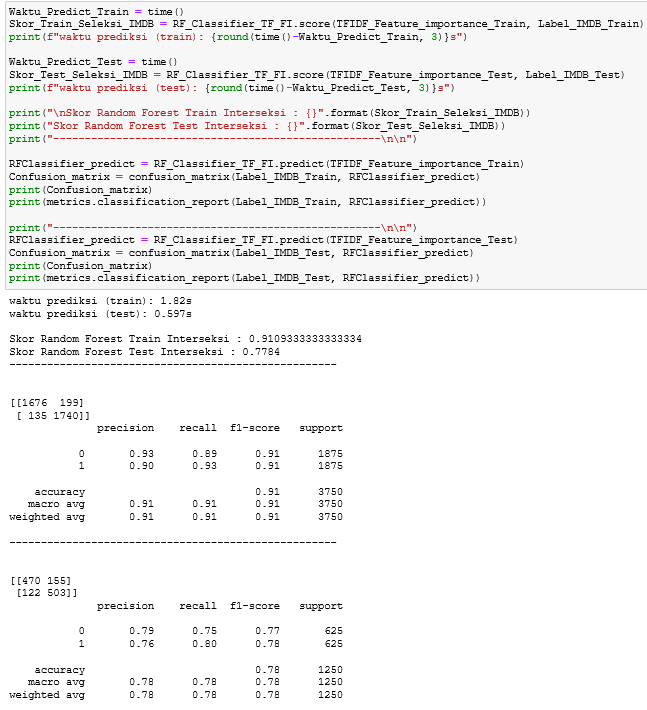
Sama seperti skenario sebelumnya, uji coba kedua dalam penerapan *transfer learning* *feature importance*  skenario ini adalah dengan menggunakan informasi model klasifikasi dengan *dataset* Amazon dan menerapkannya pada *dataset* IMDB yang akan dijadikan model klasifikasi. Gambar 4.32 menunjukkan hasil dari *dictionary* *term* pada data *train* untuk membangun model klasifikasi dari dataset IMDB sebelum dan setelah diberlakukan pemilihan *term* secara seleksi. Gambar 4.33 menunjukkan perbandingan waktu yang diperlukan untuk inisialisasi model dengan menggunakan *dataset* IMDB sebelum diseleksi dan sesudah diseleksi. Gambar 4.34 merupakan hasil uji coba performa model klasifikasi yang baru.





Gambar 4.33 Inisialisasi model klasifikasi sebelum dan setelah diterapkan seleksi *term*

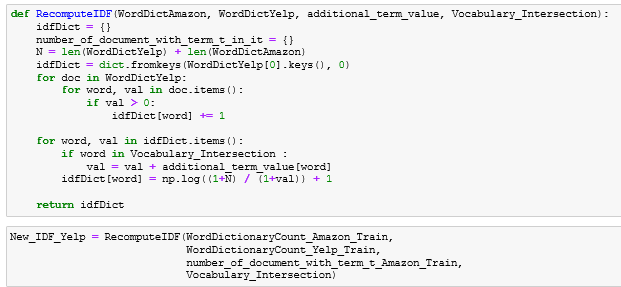
Gambar 4.32 Hasil pemilihan *term* dengan seleksi *feature importance*



Gambar 4.34 Hasil uji performa model

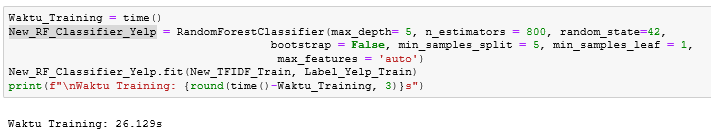
**4.3.2 Uji Coba Transfer Learning Nilai IDF**

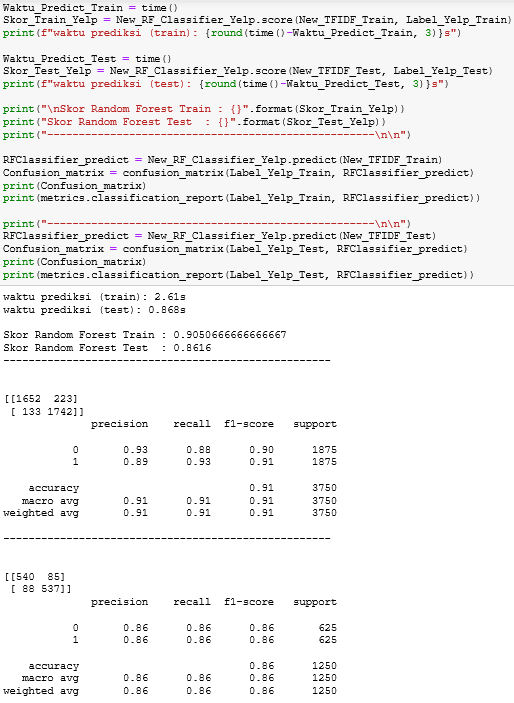
Pada Uji coba *transfer learning* nilai IDF hal yang dilakukan adalah melakukan penambahan nilai frekuensi dokumen yang memiliki suatu *term* *x* pada *dataset* yang akan menjadi model klasifikasi jika *term x* tersebut juga terdapat pada *dataset* yang telah di*-training* menjadi model klasifikasi. Gambar 4.35 menunjukkan sebuah fungsi untuk menghitung kembali nilai IDF pada suatu *data train*. Berbeda dengan perhitungan IDF sebelumnya (Gambar 4.9) pada fungsi tersebut, jumlah dokumen (N) pada *dataset* tersebut akan ditambahkan dengan jumlah dokumen pada *dataset* yang telah di-*training* menjadi model klasifikasi. Fungsi tersebut juga akan melakukan pengecekan apakah sebuah *term x* berinterseksi antar *dataset,* jika ya, maka nilai frekuensi dokumen kemunculan *term x* akan ditambahkan dengan nilai yang terdapat pada *dataset* yang telah di-*training*.



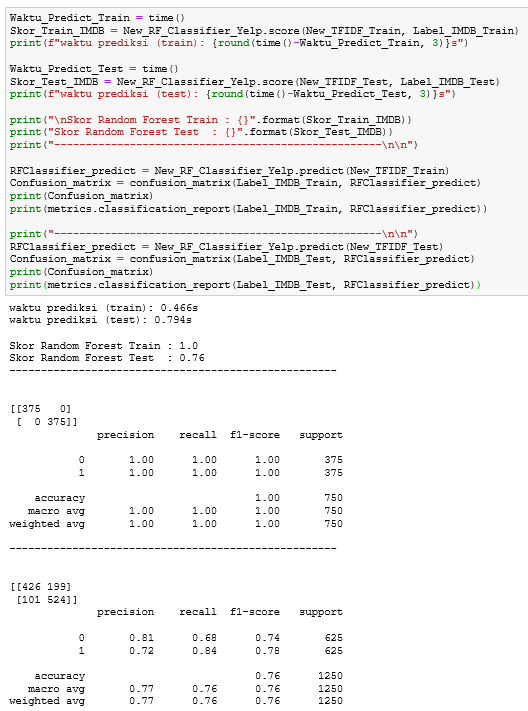
Gambar 4.35 Fungsi inisialisasi ulang nilai IDF

Setelah selesai menghitung kembali nilai IDF, langkah selanjutnya sama seperti pada skenario sebelumnya, yaitu dengan menghitung kembali nilai TF-IDF sampai dengan menguji kembali model klasifikasi baru yang telah dibuat. Gambar 4.36 merupakan proses inisialisasi model klasifikasi baru yang dibangun dengan menggunakan data *train* Yelp dengan *transfer* learning informasi IDF dari data *train* Amazon dan uji coba performa model tersebut, sedangkan Gambar 4.37 dibangun dengan menggunakan data *train* IMDB sebanyak 1000 baris data dengan *transfer* learning informasi IDF model yang dibangun dengan data *train* Amazon sebanyak 5000 baris data**.**





Gambar 4.36 Uji coba model klasifikasi model Yelp

**4.3 Hasil Uji Coba**

Gambar 4.37 Uji coba model klasifikasi model IMDB

Hasil dari kedua jenis uji coba *transfer learning* (*feature importance* dan nilai IDF) dengan masing-masing skenario adalah sebagai berikut :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Skenario | Waktu Training  (sebelum / sesudah) | Akurasi  (sebelum / setelah penerapan) | Label | Precision  (sebelum / setelah penerapan) | Recall  (sebelum / setelah penerapan) | F1  (sebelum / setelah penerapan) |
| Interseksi  Amazon &  Yelp | 29, 272 /  21, 217 Detik | 0.8624 / 0.8648 | positif | 0.86 / 0.87 | 0.86/0.86 | 0.86/0.86 |
| negatif | 0.86 / 0.86 | 0.86/0.87 | 0.86/0.87 |
| Interseksi  Amazon  & IMDB | 31, 174 / 26, 664 Detik | 0.7816 / 0.7704 | positif | 0.77/0.76 | 0.81/0.79 | 0.79/0.78 |
| negatif | 0.80/0.78 | 0.75/0.75 | 0.78/0.76 |
| Seleksi Amazon & Yelp | 25, 727 / 21,14 Detik | 0.8624 / 0.8608 | positif | 0.86/0.87 | 0.86/0.85 | 0.86/0.86 |
| negatif | 0.86/0.85 | 0.86/0.87 | 0.86/0.86 |
| Seleksi Amazon & IMDB | 30, 614 / 26, 154 Detik | 0.7816/ 0.7784 | positif | 0.77/ 0.76 | 0.81/0.80 | 0.79/0.78 |
| negatif | 0.80/0.79 | 0.75/0.75 | 0.78/0.77 |

Tabel 4.1 Hasil Uji Coba penerapan *transfer learning feature importance*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Skenario | Jumlah data *train* | Akurasi  (sebelum / setelah penerapan) | Label | Precision  (sebelum / setelah penerapan) | Recall  (sebelum / setelah penerapan) | F1  (sebelum / setelah penerapan) |
| dataset  Amazon &  IMDB | 200 | 0.6672/ 0.6696 | positif | 0.68 / 0.68 | 0.63/0.64 | 0.66/0.66 |
| negatif | 0.66 / 0.66 | 0.70/0.70 | 0.68/0.68 |
| dataset  Amazon  & IMDB | 1000 | 0.7382 / 0.7448 | positif | 0.73/0.75 | 0.75/0.75 | 0.74/0.75 |
| negatif | 0.74/0.74 | 0.73/0.74 | 0.74/0.74 |

Tabel 4.2 Hasil Uji Coba penerapan *transfer learning* nilai IDF

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| dataset  Amazon &  Yelp | 200 | 0.8248/ 0.8296 | positif | 0.82 / 0.84 | 0.83/0.82 | 0.83/0.83 |
| negatif | 0.83 / 0.82 | 0.82/0.84 | 0.82 / 0.83 |
| dataset  Amazon  & Yelp | 1000 | 0.8544 / 0.8512 | positif | 0.85/0.85 | 0.87/0.86 | 0.86/0.85 |
| negatif | 0.86/0.86 | 0.84/0.84 | 0.85/0.85 |

Tabel 4.2 Hasil Uji Coba penerapan *transfer learning* nilai IDF (lanjutan)

Dari hasil uji performa penerapan *transfer learning feature importances* yang ditunjukkan oleh tabel 4.1, dapat disimpulkan bahwa pada setiap skenario *dataset* memiliki selisih durasi waktu *training* yang cukup signifikan, dikarenakan jumlah fitur *term* yang digunakan jauh berkurang. Namun akurasi, presisi, *recall*, dan f1-score memiliki nilai yang hampir serupa, tidak jauh berbeda. Dari hasil uji performa penerapan *transfer learning* nilai IDF yang ditunjukkan oleh tabel 4.2, dapat disimpulkan bahwa….

**BAB V**

# **SIMPULAN DAN SARAN**

## **Simpulan**

## **Saran**

# **DAFTAR PUSTAKA**

# **DAFTAR RIWAYAT HIDUP**