SENTIMEN ANALISIS INDIHOME DI X (TWITTER) DENGAN METODE *FASTTEXT* DAN *SUPPORT VECTOR MACHINE*

SKRIPSI

SARAH SYAFITRI

171401118

PROGRAM STUDI S1 ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UNIVERSITAS SUMATERA UTARA

MEDAN

2024

SENTIMEN ANALISIS INDIHOME DI X (TWITTER) DENGAN METODE *FASTTEXT* DAN *SUPPORT VECTOR MACHINE*

SKRIPSI

Diajukan untuk melengkapi Tugas Akhir dan memenuhi syarat memperoleh ijazah Sarjana S-1 Ilmu Komputer

SARAH SYAFITRI

171401118

PROGRAM STUDI S1 ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UNIVERSITAS SUMATERA UTARA

MEDAN

2024

PERSETUJUAN

PERSETUJUAN

Judul	:	SENTIMEN ANALISIS INDIHOME DI X (TWITTER) DENGAN METODE FASTTEXT DAN SUPPORT VECTOR MACHINE
Kategori	:	SKRIPSI
Nama	:	SARAH SYAFITRI
Nomor Induk Mahasiswa	:	171401118
Program Studi	:	SARJANA (S1) ILMU KOMPUTER
Fakultas	:	ILMU KOMPUTER DAN TEKNOLOGI INFORMASI UNIVERSITAS SUMATERA UTARA

Telah diuji dinyatakan lulus di Medan, 12 Juni 2024

Komisi Pembimbing :

Pembimbing 2

Dian Rachmawati S.Si., M.Kom.

NIP. 198307232009122004

Pembimbing 1

Dr. Amalia S.T., M.T.

NIP. 197812212014042001

Diketahui/disetujui oleh

Program Studi S1 Ilmu Komputer

Ketua,

Dr. Amalia S.T., M.T.

NIP. 197812212014042001

PERNYATAAN ORISINALITAS

SENTIMEN ANALISIS INDIHOME DI X (TWITTER) DENGAN METODE FASTTEXT DAN SUPPORT VECTOR MACHINE

SKRIPSI

Saya mengakui bahwa skripsi ini adalah hasil karya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, Maret 2024

Sarah Syafitri
171401118

PENGHARGAAN

Puji syukur saya sampaikan ke hadirat Tuhan Yang Maha Esa yang telah memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi ini dengan baik sebagai syarat untuk memperoleh gelar Sarjana Komputer Program Studi S-1 Ilmu Komputer Fasilkom-TI Universitas Sumatera Utara.

Dalam penyusunan skripsi ini, penulis menyadari semua proses bukan hanya usaha dan kemampuan dari penulis sendiri, melainkan dengan bantuan dan dukungan dari berbagai pihak. Pada kesempatan ini, penulis ingin menyampaikan rasa ungkapan terima kasih kepada semua pihak yang telah membantu penulis selama proses penyusunan dan penyelesaian skripsi ini. Penulis mengucapkan terima kasih kepada:

1. Bapak Dr. Muryanto Amin, S.Sos., M.Si selaku Rektor Universitas Sumatera Utara.
2. Bapak Dr. Mohammad Andri Budiman S.T., M.Comp.Sc., M.E.M. selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara.
3. Ibu Dr. Amalia, S.T., M.T selaku Ketua Program Studi S-1 Ilmu Komputer Universitas Sumatera Utara, Dosen Pembimbing I dan juga selaku Dosen Pembimbing Akademik yang telah memberikan banyak saran, dukungan, serta motivasi kepada penulis dalam menyelesaikan skripsi ini sehingga bisa terselesaikan dengan baik.
4. Ibu Sri Melvani Hardi S.Kom., M.Kom. selaku Sekretaris Program Studi S-1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara yang telah memberi dukungan dalam proses penyusunan skripsi ini.
5. Ibu Dian Rachmawati S.Si., M.Kom. selaku Dosen Pembimbing II yang telah memberikan saran, dukungan, serta motivasi kepada penulis dalam menyelesaikan skripsi ini sehingga bisa terselesaikan dengan baik.
6. Seluruh tenaga pengajar dan pegawai di Fakultas Ilmu Komputer dan Teknologi Informasi USU yang telah membantu penulis dalam proses pembuatan skripsi.
7. Kedua orang tua tersayang, Ibu Lusy Meidya yang senantiasa memberikan kasih sayang, doa, dukungan, motivasi serta dukungan finansial kepada penulis dan alm.

Bapak Hifni Rani yang dukungan, nasehat dan motivasinya akan selalu penulis kenang.

8. Kepada adik-adik tersayang, Siti Fadhila Syahrani dan Safira Syahrani yang selalu mendukung penulis.
9. Teman-teman yang telah bersama sejak masa awal perkuliahan sampai sekarang, Alya Andira Lubis, Evi Kristina Purba, Mellianta Sitepu, dan T. Arifah Inayah, yang selalu mendukung dan memotivasi penulis.
10. Teman-teman online dalam komunitas hobi yang selalu memberi hiburan dan semangat kepada penulis.
11. Semua pihak yang terlibat baik secara langsung maupun tidak langsung yang telah banyak membantu, yang tidak dapat disebutkan namanya satu per satu.
12. Sarah Syafitri sebagai penulis yang telah berhasil bertahan melewati berbagai hal, tidak menyerah selama proses penyusunan skripsi ini hingga dapat selesai dengan baik dan dapat menyelesaikan perkuliahan sampai akhir.

Semoga Allah SWT melimpahkan berkah kepada semua pihak yang telah memberikan bantuan, semangat, perhatian, serta dukungan kepada penulis dalam menyelesaikan skripsi ini dan penulis berharap semoga skripsi ini dapat memberikan manfaat ke depannya.

Medan, 02 April 2024

Penulis

SENTIMENT ANALISIS INDIHOME DI X (TWITTER) DENGAN METODE *FASTTEXT* DAN *SUPPORT VECTOR MACHINE*

ABSTRAK

Indonesia Digital Home atau biasa disingkat Indihome merupakan salah satu produk layanan internet, telepon rumah, dan televisi yang mengimplementasikan kabel fiber optik dari PT Telekomunikasi Indonesia. IndiHome merupakan penyedia layanan internet yang paling banyak digunakan di Indonesia karena jangkauan sinyalnya yang luas dan harga paket yang menarik. Tingginya pengguna berbanding lurus dengan kebutuhan masyarakat dalam mengakses berbagai informasi sebagaimana internet sudah menjadi kebutuhan utama. Namun, IndiHome tetap tidak lepas dari gangguan layanan. Interferensi pada IndiHome terdiri dari interferensi loss dan interferensi router. Setiap terjadi gangguan, pengguna akan melapor pada *customer service* Indihome yang tersedia melalui telepon 188, aplikasi *Whatsapp*, aplikasi MyIndihome dan *email*. Berbagai cara pengguna dalam berkeluh kesah dan mengkritik layanan IndiHome, salah satunya melalui media sosial X (Twitter). Pendapat yang terdapat pada X (Twitter) cukup beragam, hal ini dikarenakan pengguna X (Twitter) mempunyai latar belakang dan kelompok masyarakat yang berbeda. Sehingga perlu dilakukan sentimen analisis pengguna Indihome di X (Twitter) dengan menggunakan metode FastText dan Support Vector Machine (SVM). Proses klasifikasi dimulai dari input dataset, pre-processing data, pelabelan data, split data, word embedding, klasifikasi dan evaluasi. Hasil penelitian ini menunjukkan bahwa evaluasi menggunakan konfusi matriks tanpa menggunakan FastText (TF-IDF) dapat menghasilkan nilai akurasi sebesar 56.11% lebih tinggi dibandingkan dengan menggunakan FastText yaitu 55.78%.

Kata Kunci: Sentimen Analisis, IndiHome, FastText, Support Vector Machine

SENTIMENT ANALYSIS OF INDIHOME ON X (TWITTER) WITH FASTTEXT METHOD AND SUPPORT VECTOR MACHINE

ABSTRACT

Indonesia Digital Home or commonly abbreviated as Indihome is one of the internet, home telephone and television service products that implement fiber optic cable from PT Telekomunikasi Indonesia. IndiHome is the most widely used internet service provider in Indonesia due to its wide signal coverage and attractive package prices. The high number of users is directly proportional to the community's need to access various information as the internet has become a primary need. Despite all that, IndiHome still cannot be separated from service interruptions. Interference with IndiHome consists of loss interference and router interference. Every time a disturbance occurs, users will report to Indihome customer service available via telephone 188, Whatsapp application, MyIndihome application and email. Various ways users complain and criticize IndiHome services, one of which is using sosial media X (twitter). The opinions contained in X (tweet) are quite varied, this is because X (twitter) users have different backgrounds and come from different groups of society. So it is necessary to analyze the sentiment of Indihome users on X (Twitter) using the FastText and Support Vector Machine (SVM) methods. The classification process starts from dataset input, data pre-processing, data labeling, split data, word embedding, classification and evaluation. The results of this study show that evaluation using confusion matrix without using FastText (TF-IDF) can produce an accuracy value of 56.11% higher than using FastText which is 55.78%.

Keywords: *Sentiment Analysis, IndiHome, FastText, Support Vector Machine*

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Internet merupakan jaringan komunikasi elektronik yang menghubungkan berbagai jaringan komputer untuk mengakses informasi, berita, hiburan, pekerjaan dan kebutuhan lainnya dengan cepat (Mota and Cilento 2020). IndiHome merupakan salah satu dari banyak perusahaan layanan internet yang tersedia di Indonesia. Indonesia *Digital Home* atau yang biasa disingkat sebagai IndiHome merupakan satu dari PT Telekomunikasi Indonesia produk tersebut mencakup layanan internet, telepon rumah serta televisi yang mengimplementasikan kabel fiber optik. Dilansir dari *website* (PT Telkom 2022), IndiHome telah mencapai 9,2 juta pengguna di seluruh Indonesia sejak peluncurannya tahun 2015. Hal ini dikarenakan jangkauan sinyal yang luas dan harga paket yang menarik, IndiHome menjadi penyedia internet yang paling populer di Indonesia (Asosiasi Penyelenggara Jasa Internet Indonesia 2023). Tingginya pengguna berbanding lurus dengan kebutuhan masyarakat untuk mengakses berbagai informasi sebagaimana internet telah menjadi kebutuhan utama (Fitriasti and Priansa 2021). Terlepas dari semua itu, IndiHome tetap tidak bisa lepas dari gangguan layanan. Gangguan pada IndiHome terdiri dari gangguan *loss* dan gangguan pada *router* (Nastiti, Ramadan, and Tulloh 2021). Setiap terjadi gangguan, pengguna akan melapor pada *customer service* Indihome yang tersedia melalui telepon 188, aplikasi *Whatsapp*, aplikasi MyIndihome dan *email*. Berbagai cara pengguna dalam berkeluh kesah dan mengkritik layanan IndiHome salah satunya melalui media sosial X (*Twitter*).

Di X (*Twitter*) pengguna mengutarakan opini mereka dengan *post* (*tweet*) yang berhubungan dengan ide, pertanyaan, pujian, pendapat terhadap suatu topik, dan penilaian suatu produk (Raihan and Setiawan 2021). Pengguna Twitter X berasal dari berbagai latar belakang dan kelompok masyarakat yang berbeda, menyebabkan pendapat mereka cukup beragam (Aldisa and Maulana 2022). Pengumpulan data melalui media sosial sangat efisien karena bisa mendapatkan data secara real *time*,

menghasilkan data yang lebih detail yang menggambarkan pendapat masyarakat dan biaya minimal yang perlu dikeluarkan (Rachman and Pramana 2020).

Data yang diperoleh selanjutnya akan dilakukan analisis sentimen. Analisis sentimen merupakan proses memahami, mengekstraksi, dan memproses data teks secara otomatis. Analisis sentimen dapat digunakan untuk mengevaluasi, pendapat, penilaian, sikap, dan emosi individu terhadap barang, jasa, organisasi, orang, topik, dan peristiwa (Thavareesan and Mahesan 2020). Dalam analisis sentimen penting untuk memahami dan mengklasifikasikan teks ke dalam positif, negatif dan netral (Kilimci 2020).

Penelitian sebelumnya yang melakukan analisis sentimen dilakukan oleh (Nurdin and al 2020) mencoba membandingkan kinerja dari word embedding *GloVe, Word2Vec*, serta *FastText* menggunakan *Convolutional Neural Network*. Ketiga metode ini memiliki kemampuan untuk menangkap definisi, urutan dan konteks kata jika dibandingkan dengan *Bag of Words*. Hal ini merupakan alasan mengapa ketiga *word embedding* tersebut dipilih. Metode ini akan dibandingkan dengan klasifikasi berita dari kumpulan dua puluh *newsgroup* dan *Reuters Newswire.* Dengan nilai *F1-score* sebesar 0.979 untuk *dataset* 20 *Newsgroup* dan 0.715 untuk *Reuters, FastText* menunjukkan kinerja terbaik dari dua metode *word embedding* lainnya. Penelitian lain dilakukan oleh (Darwis, Pratiwi, and Pasaribu 2020) melihat bagaimana data Twitter dapat digunakan untuk mengenai Komisi Pemberantasan Korupsi Republik Indonesia dengan menggunakan metode Support Vector Machine (SVM). Penelitian ini menggunakan TF-IDF untuk ekstraksi fitur dan metode SVM untuk klasifikasi. Penelitian ini mengekstrak 1890 data dari 2000 data *crawling* Twitter dan 3846 kata atau term. Kemudian nilai kata dihitung untuk *labeling*, yang menghasilkan sentimen positif, negatif, dan netral. Hasil pengujian menunjukkan bahwa penggunaan metode SVM menghasilkan 82% akurasi dan 77% sentimen dengan label negatif, 8% sentimen dengan label positif, dan 25% sentimen dengan label netral.

Berdasarkan permasalahan yang telah dijelaskan diatas penulis melakukan penelitian dengan menggunakan metode FastText dan *Support Vector Machine* (SVM). Metode FastText digunakan sebagai *word embedding* karena dapat memetakan *subword* atau suku kata, yang menghasilkan representasi kata lebih baik ketika digunakan pada bahasa yang mempunyai banyak jenis kata (Sepriadi et al. 2023). Sedangkan Metode SVM dipilih karena teknik klasifikasi non-linearnya yang sangat baik untuk generalisasi

data dan dapat menghasilkan model klasifikasi yang sangat baik meskipun dilatih dengan himpunan data yang jumlahnya sedikit. (Asshiddiqi and Lhaksmana 2020). Oleh karena itu peneliti melakukan penelitian mengenai IndiHome terkait kualitas produk & layanan IndiHome dengan judul "Analisis Sentimen IndiHome di X *(Twitter)* dengan Metode *FastText* dan *Support Vector Machine* (SVM)".

1.2. Rumusan Masalah

Pada penelitian ini, peneliti menyadari bahwa ketika terjadi gangguan pada IndiHome masyarakat akan berkeluh kesah dan menyampaikan opininya ke media sosial X (Twitter), dengan ini penulis ingin melakukan sentimen analisis untuk menganalisis opini-opini masyarakat terkait kualitas produk & layanan IndiHome di X (*Twitter*) dengan menerapkan metode SVM dengan menggunakan FastText dan tanpa FastText (TF-IDF).

1.3. Batasan Masalah

Batasan masalah penelitian ini adalah sebagai berikut:

1. Penelitian dilakukan untuk analisis sentimen IndiHome pada X (*Twitter*).
2. Dataset yang digunakan merupakan teks bahasa Indonesia berasal dari media sosial X (Twitter) berupa *tweet* dengan *keyword* IndiHome.
3. Penelitian ini melakukan *pre-processing* data untuk mempersiapkan data agar mudah pada proses klasifikasi.
4. Pelabelan data yang dilakukan terdiri dari 3 label yaitu label Positif, Negatif dan Netral.
5. Metode *word embedding* yang digunakan yaitu *FastText*.
6. SVM merupakan metode klasifikasi teks yang digunakan.
7. *Output* penelitian yaitu akurasi metode SVM menggunakan FastText dan tanpa FastText dalam melakukan analisis sentimen IndiHome pada X (*Twitter)*.

1.4. Tujuan Penelitian

Tujuan penelitian ini, berdasarkan rumusan masalah yang telah dibuat, adalah untuk menganalisis atau mengetahui persentase sentimen positif, negatif dan netral serta

tingkat keakurasian penerapan metode SVM menggunakan FastText dan tanpa FastText dari sentimen IndiHome di X (*Twitter*).

1.5. Manfaat Penelitian

Manfaat dari penelitian yang dilakukan adalah sebagai berikut:

1. Mengetahui persentase sentimen positif, negatif dan netral serta tingkat keakurasian label terprediksi pada *keyword* IndiHome di X (*twitter)* berdasarkan metode SVM dengan kombinasi metode FastText dan tanpa FastText.
2. Dapat digunakan sebagai referensi untuk penelitian yang akan datang dalam melakukan analisis sentimen.

1.6. Metodologi Penelitian

Metodologi penelitian untuk menganalisis sentimen IndiHome di X (*Twitter*) menggunakan metode FastText dan SVM adalah sebagai berikut :

1. Pengumpulan Data

   Tahap pengumpulan data menggunakan dua buah teknik yaitu studi pustaka dan *web scraping*.

   a. Studi Pustaka

      Referensi dikumpulkan dari berbagai sumber yang dapat diandalkan seperti jurnal, buku dan ulasan literatur yang berkaitan dengan analisis *sentiment* menggunakan metode FastText dan teori sebagai bahan referensi dalam penelitian ini. Teori yang digunakan yaitu IndiHome Telkomsel, X (*Twitter*), Analisis Sentimen, Web Scraping, Natural Language Processing, Word Embedding, TextBlob, FastText, TF-IDF, Algoritma Support Vector Machine (SVM), Confusion Matrix, dan Google Colaboratory.

   b. *Web Scraping*

      Pada penelitian ini, website yang akan discraping adalah X (Twitter). Web scraping merupakan teknik untuk mengambil dokumen dari sebuah website.

2. Pre-processing Data

   Tahap pre-processing merupakan tahapan membersihkan data tweet yang terdiri dari kata atau simbol yang tidak penting, sehingga data tweet lebih mudah untuk

diolah. Tahapan pre-processing yang digunakan pada penelitian ini adalah sebagai berikut:

a. *Cleaning* Data, dilakukan untuk mengurangi *noise* dalam data. Proses ini menghapus kata-kata yang tidak penting seperti *hashtag, username, url, email, emoticon* dan tanda baca seperti koma, titik, dan lainnya.
b. *Case folding*, adalah proses perubahan huruf besar ke huruf kecil (*lowercase*) pada kalimat.
c. *Tokenizing*, adalah proses pemisahan atau pemenggalan kata berdasarkan spasi.
d. *Stopword Removal*, adalah proses penghapusan kata-kata yang tidak penting.
e. *Stemming*, adalah proses penghapusan imbuhan menjadi bentuk kata dasar.

3. Pelabelan Data

   Setelah dilakukan tahap *pre-processing* data yang telah bersih selanjutnya diberi label positif, negatif dan netral.

4. Split Data

   Setelah pelabelan selesai, data dibagi menjadi dua jenis, yaitu data train dan data test, dengan perbandingan 80:20. Data train digunakan untuk membangun model dan data test digunakan untuk testing model yang dibangun. Semakin banyak data train untuk melatih model maka model yang dibangun menjadi lebih baik (Mahdi and al 2021).

5. Word Embedding Metode TF-IDF & FastText

   Word embedding dilakukan untuk meng-convert kata menjadi bentuk vector. Dalam penelitian ini menggunakan metode TF-IDF & FastText. Data yang sudah dibagi menjadi data train dan data test selanjutnya akan dibuatkan kamus vektornya. Pada percobaan data embedding tanpa FastText data train dan data test akan dihitung menggunakan TF-IDF sedangkan percobaan kedua data pelatihan dan data uji akan di embedding dengan FastText. Vektor dari data pelatihan dan data uji akan menjadi input SVM.

6. Metode SVM untuk Klasifikasi Teks

   Pada tahap ini, penulis mengklasifikasikan data yang telah diolah di tahap sebelumnya menggunakan algoritma SVM, sehingga diperoleh hasil analisis sentimen IndiHome di X (Twitter). Kemudian, dilakukan pengujian model menggunakan data uji, Implementasi dilakukan menggunakan bahasa pemrograman Python dengan tools *Google Colaboratory*.

7. Evaluasi

   Evaluasi dilakukan untuk mengetahui tingkat keakuratan metode dalam klasifikasi. Pada penelitian ini, metode *confusion matrix* digunakan untuk menghasilkan nilai *accuracy, precision, recall* dan *f1-score.*

8. Dokumentasi

   Penelitian yang telah dilakukan dari tahap analisis hingga tahap pengujian, didokumentasikan dalam bentuk skripsi.

1.7. Sistematika Penyusunan

Saat menulis karya ilmiah seperti skripsi, struktur penyusunan digunakan. Berikut merupakan penjabaran mengenai sistematika penyusunan yang digunakan:

BAB 1 : PENDAHULUAN

Pada Bab Pendahuluan berisi latar belakang, rumusan masalah, tujuan, manfaat, dan metodologi penelitian.

BAB 2 : TINJAUAN PUSTAKA

Pada Bab Tinjauan Pustaka berisi pengetahuan dasar dan teori-teori yang relevan dengan penelitian. Teori-teori tersebut antara lain IndiHome Telkomsel, X (Twitter), Analisis Sentimen, Web Scraping, Natural Language Processing, Analisis Sentimen, Word Embedding, FastText, Support Vector Machine dan Confusion Matrix, *Google Colaboratory* dan Penelitian yang Relevan .

BAB 3 : ANALISIS DAN PERANCANGAN

Pada bab ini berisi penjelasan proses analisis sistem yang dibutuhkan dengan membagi sentimen menjadi tiga kelas yaitu label positif, label negatif dan label netral.

BAB 4 : IMPLEMENTASI DAN PENGUJIAN

Pada bab ini, pembahasan hasil penelitian yang telah diimplementasikan ke dalam sistem *Google Colaboratory*.

BAB 5 : KESIMPULAN DAN SARAN

Pada bab ini berisi kesimpulan pada penelitian yang telah dilakukan dan saran untuk penelitian selanjutnya.

BAB 2

TINJAUAN PUSTAKA

Tinjauan pustaka dalam penelitian ini menjelaskan teori-teori yang relevan dalam penelitian sentimen analisis IndiHome menggunakan metode FastText dan SVM.

2.1 IndiHome Telkomsel

IndiHome menjadi salah satu layanan populer PT. Telekomunikasi Indonesia (Persero) Tbk, juga dikenal sebagai Telkom Indonesia. Menurut situs website IndiHome, layanan digitalnya terdiri dari TV interaktif, internet rumah, dan telepon rumah yang dioperasikan melalui teknologi fiber optik (Fani, Sudarmaningtyas, and Nurcahyawati 2020).

Indihome terus memperoleh pelanggan baru setiap tahun, Dilansir dari website PT Telkom, IndiHome telah mencapai 9,2 juta pengguna di seluruh Indonesia sejak peluncurannya tahun 2015. Hal ini dikarenakan jangkauan sinyal yang luas dan harga paket yang menarik, IndiHome menjadi penyedia internet yang paling populer di Indonesia.

2.2 X (Twitter)

X, atau yang dulu dikenal sebagai *Twitter*, adalah platform media sosial yang dimiliki dan dioperasikan oleh *X Corp.* X *(Twitter)* mempunyai *microblogging* yang dapat mengirim pesan penggunanya secara *real-time*. Pesan yang dikirim di X (*Twitter*) dikenal dengan nama *tweet*. Untuk setiap *posting*, pengguna Twitter (X) dapat mengirimkan 280 karakter, yang dapat diisi dengan ide, pertanyaan, pujian, keluh kesah tentang topik tertentu dan penilaian suatu produk. Keterbatasan jumlah karakter yang dapat dikirim, menyebabkan *tweet* sering mengandung singkatan, bahasa slang, dan kesalahan pengejaan (Wulandari, Saedudin, and Andreswari 2021).

Penggunaan X (*Twitter*), terus berkembang pesat. Dengan menggunakan hashtag atau mencari topik *trending* dari berita terkini, orang-orang dapat dengan cepat

mendapatkan informasi (Hasri and Alita 2022). Oleh sebab itu, X (*Twitter*) menjadi sumber untuk penelitian klasifikasi teks karena berfungsi sebagai tempat mencari informasi yang singkat dan cepat.

2.3 Analisis Sentimen

Analisis sentimen adalah studi yang bertujuan untuk mengenali dan mengungkapkan ide, perasaan, atau perspektif yang terdapat dalam teks yang dapat digunakan untuk mengetahui seberapa banyak sentimen positif, negatif dan netral terhadap seseorang, organisasi, produk dan topik tertentu. Langkah-langkah klasifikasi analisis sentimen adalah mengumpulkan *dataset* seperti pendapat masyarakat atau penilaian tentang suatu produk, dimana pada penelitian ini dataset didapat dari web scraping. Langkah selanjutnya adalah *pre-processing*, seperti *cleaning* data, *case folding*, *tokenization, stopword removal* dan *stemming*. Berikutnya, *word embedding* yaitu pembobotan data teks, dimana pada penelitian ini menggunakan FastText dan TF-IDF. Berikutnya *classification* atau klasifikasi teks, pada penelitian ini menggunakan Support Vector Machine dengan Kernel RBF. Terakhir, *Evaluation* yaitu tahap menghitung nilai *accuracy, recall, precision* dan F1-*Score*. (Mara, Sediyono, and Purnomo 2021).

2.4 *Web Scraping*

Web scraping adalah proses pengambilan dokumen semi-terstruktur dari internet berupa HTML atau SHTML secara otomatis dengan program yang dapat digunakan untuk tujuan lain, seperti analisis (Khder 2021).

Proses dari *web scraping* yaitu :

1. Tahap Pengambilan Informasi Website: Penulis mempelajari dokumen HTML dari website yang akan digunakan untuk menggabungkan informasinya ke *tag* HTML. Lalu penulis mempelajari dan memilih *feature* yang akan diambil ke dalam aplikasi *scraper* yang akan dibuat. Pada penelitian ini penulis akan mengambil data berupa tweet dari website media sosial X (Twitter).
2. Tahap Ekstraksi: Aplikasi scraper dibuat untuk pengumpulan data secara otomatis dari website. Penelitian ini menggunakan Selenium sebagai *library*

dari Python untuk ekstraksi tweet dengan keyword “IndiHome” yang dibangun menggunakan *Streamlit*.

3. Tahap Transformasi: Data hasil scraping disimpan ke dalam format CSV yang selanjutnya akan di analisis.

2.5 *Natural Language Processing*

Tujuan dari *Natural Language Processing* (NLP) yaitu untuk memungkinkan komputer berinteraksi dengan manusia melalui bahasa manusia dengan cara yang lebih efisien. Tujuan dari *pre-processing* pada NLP adalah untuk memperbaiki bahasa pada teks karena banyak kata yang tidak baku dan singkatan di dalam dataset. *Pre-processing* yang dilakukan oleh NLP adalah sebagai berikut (Nurkholis 2021) :

1. *Cleaning Data*

   Tahap ini menghapus tanda baca, simbol, angka, *emoticons* dan *special characters* pada teks. Simbol yang dihapus adalah “ ~”, “ `”, “ !”, “ $”, “ %”,” ^”, “ &”, “ *”, “ (“, “)”,” _”,” –“, “ +”, “ =”, “ :”, “ “”, “ “”,” ”, “koma”, ” titik”, “ ?”, “ /”, “ \”, “ #”, “ dan “0-9”.

2. *Case Folding*

   Tahap *case folding* bertujuan untuk mengubah semua teks menjadi huruf kecil. Semua karakter antara "A" dan "Z" dalam data diubah menjadi karakter antara "a" dan "z".

3. *Tokenizing*

   Tahap *tokenizing* untuk pemisahan kata berdasarkan spasi. Tahapan ini bertujuan untuk memecah kata yang terdapat dalam satu kalimat menjadi kata-kata.

4. *Normalization*

   Tahap mengubah bahasa tidak baku, bahasa gaul (*slang)* dan kata singkatan menjadi bahasa baku.

5. *Stopword*

   Tahap ini untuk menghapus kata hubung yang sering muncul dan tidak penting dengan menggunakan algoritma *stoplist* untuk menghapus kata yang tidak penting atau *wordlist* untuk menyimpan kata yang penting. Kata hubung seperti "dan", "yang", "serta", "setelah", dan lainnya muncul pada *dataset* penelitian ini. Penggunaan stopword dapat mengurangi ukuran indeks, waktu pemrosesan dan *noise* data.

6. *Stemming*

   Tahap ini mengubah setiap kata berimbuhan menjadi kata dasar. Tujuan dari *Stemming* untuk mengurangi jumlah indeks yang berbeda yang ada dalam data, maka kata yang mempunyai awalan atau akhiran kembali ke kata aslinya. Selain itu, *Stemming* digunakan untuk mengelompokkan kata-kata yang memiliki imbuhan berbeda tetapi memiliki kata dasar dan makna yang sama.

2.6 *TextBlob*

Textblob adalah *library* yang digunakan untuk pemberian tag kata, ekstraksi kata benda, klasifikasi sentimen kata dan terjemahan kata. *Textblob* digunakan pada penelitian ini untuk pelabelan kata secara otomatis. *Textblob* akan menghitung nilai *polarity* yang berfungsi untuk melihat sentimen teks. Dari nilai *polarity,* akan diklasifikasikan menjadi 3 kelas, yaitu positif, negatif, dan netral (Baita, Pristyanto, and Cahyono 2021). Data pada penelitian ini merupakan data teks Bahasa Indonesia maka teks setelah melalui tahap *pre-processing* akan diterjemahkan ke terlebih dahulu ke Bahasa Inggris lalu dilakukan pelabelan dengan *textblob*.

2.7 *Word Embedding*

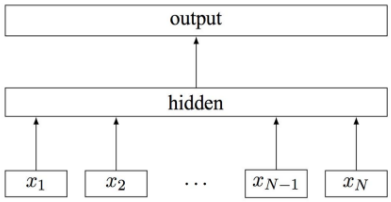
Word embedding merupakan proses mengubah setiap kata menjadi *vector* yang mewakili maknanya sendiri. Sebagian besar algoritma *machine learning* tidak dapat menganalisis input data yang berupa *string* atau teks, jadi kata-kata dimasukkan supaya masukkan dapat berupa angka. *Word embedding* bisa dikenal juga dengan nama *word vector representation* (Islamy, Indriati, and Adikara 2022). Kelebihan *word embedding* tidak membutuhkan anotasi, dapat langsung diturunkan dari *corpus* tak ter anotasi (Nurdin and al 2020).

2.8 *FastText*

FastText merupakan pengembangan dari word2vec *embedding*. Perbedaannya adalah melihat representasi *word* dengan melihat informasi *subword*. Setiap kata akan dipecah menjadi kumpulan karakter n-gram. Kata memiliki representasi vector berdasarkan jumlah representasi vector pada setiap n-gram(Nurdin and al 2020).

Keunggulan FastText yaitu dapat menangani kata yang tidak pernah ditemukan sebelumnya (OOV).

FastText memiliki arsitektur yang terdiri dari 3 lapisan, lapisan input (*embedding layer*), lapisan tersembunyi (*hidden layer*), serta lapisan keluaran (*output layer*). Untuk mencari representasi vector kata, FastText memiliki 2 metode, yaitu *Continous Bag of Words* (CBOW) dan *Skip-gram*. Pada CBOW konteks kalimat sebagai inputan untuk memprediksi satu kata, sedangkan *Skip-gram* memprediksi satu kata sebagai inputan untuk memprediksi konteks kalimat (Pardede and Pakpahan 2023). Pada penelitian ini, model *Skip-gram* dipilih karena lebih cocok digunakan pada dataset yang berukuran kecil dan dapat merepresentasikan kata-kata yang jarang muncul dengan lebih baik.



Gambar 2. 1 Arsitektur Skip-gram FastText

Setiap kata pada pada dataset direpresentasikan sebagai vektor kata x_1 sampai x_n. Selanjutnya akan dilakukan perhitungan *Hidden Layer* dan *Output Layer.* Untuk menghitung *Hidden Layer* dan *Output Layer* diperlukan inisialisasi parameter. Penentuan nilai pada setiap kata adalah nilai *random* dengan *range* -1 sampai 1 (Rifaldo, 2021). *Output Layer* merupakan output dari FastText yang terdiri dari Data *Train* dan OOV (Data Tes).

1. *Hidden Layer*

   Untuk menghitung hidden layer dengan persamaan rumus (2.2) berikut.

   $$z_{inj} = V_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad (2.2)$$

   Keterangan:

   z_{inj}= Hasil hidden layer

   V_{0j} = Bobot Bias

   X = Inputan nilai data

   $V_{i,j}$ = Bobot input ke hidden.

Nilai bias didapatkan dari inisialisasi bobot nilai di mana pada penelitian ini, bobot yang digunakan untuk bobot Bias adalah 0 (Putri, Yuhandri, and Nurcahyo 2021).

2. *Output Layer*

Selanjutnya adalah menghitung *output layer* menggunakan persamaan rumus FastText seperti (2.3) berikut (Pardede and Pakpahan 2023).

$$FastText_{word} = V_{word} + \sum g \in ngrams(word) \quad (2.3)$$

Keterangan:

V_{word} : Vektor Kata

$ngrams$: Nilai dari n-gram kata

word : nilai dari setiap kata (terdapat pada Tabel 3.19)

3. *Out of Vocabulary*

Jika kata yang akan dihitung tidak ada di dalam data *training*, maka dianggap sebagai keluar dari kosakata (*Out of Vocabulary;* OOV) dan vektor kata adalah total dari n-gram. Persamaan rumus (2.4) adalah persamaan rumus untuk menghitung OOV.

$$FastText_{oov\_word} = \sum g \in ngrams(oov\_word) \quad (2.4)$$

Keterangan:

$ngrams$: Nilai dari n-gram kata

oov_word : nilai dari kata yang tidak ada di dalam kamus/data training

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

Term Frequency-Inverse Document Frequency (TF-IDF) adalah metode pembobotan setiap kata dalam teks. *Term frequency* (TF) adalah kemunculan sebuah kata dalam suatu teks sedangkan *inverse document frequency* (IDF) adalah jumlah seluruh teks yang mengandung kata tertentu (Assidyk, Setiawan, and Kurniawan 2020). Maka metode TF-IDF dapat menghitung total bobot dari kata pada teks dengan persamaan rumus yang dapat dilihat pada (2.4) (Maulidina 2020).

$$TF - IDF = tf_t * IDF_t \quad (2.4)$$

$$IDF = \log\left(\frac{D}{DF}\right)$$

Keterangan:

tf_t = *Term Frequency*

IDF = *Inverse Document Frequency*

D = Total Dokumen

DF = Total kemunculan kata pada dari seluruh dokumen

Pada *Sklearn*, formula IDF yang digunakan agak berbeda. Formula IDF pada *sklearn* seperti pada persamaan (2.5) berikut (Widodo and Hartono 2023).

$$IDF = \ln\left(\frac{1+D}{1+DF}\right) + 1 \tag{2.5}$$

Selanjutnya pada *sklearn*, nilai TF-IDF juga akan dilakukan normalisasi L2 yang memiliki formula seperti pada (2.6) berikut (Widodo and Hartono 2023),

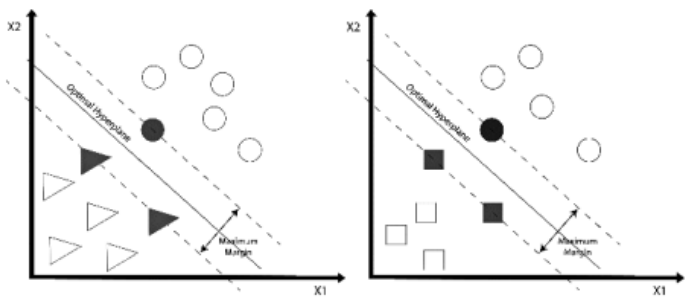
$$L2 - Norm = \sqrt[2]{TF - IDF_1^2 + TF - IDF_2^2 + \cdots + TF - IDF_i^2} \tag{2.6}$$

Sehingga hasil dari TF-IDF L2 Norm adalah seperti pada (2.7) berikut (Widodo and Hartono 2023).

$$TF - IDF\ L2 - Norm = \frac{TF - IDF\ sebelum\ Norm}{L2 - Norm} \tag{2.7}$$

2.10 Support Vector Machine (SVM)

Support Vector Machine (SVM) merupakan algoritma untuk klasifikasi dengan fungsi pemisahan data (*hyperplane*) yang akan mencari margin terbesar yang memungkinkan pemisahan data tweet dengan sentimen positif, negatif dan netral secara optimal. Ilustrasi SVM ditunjukkan pada Gambar 2.2 (Fremmuzar and Baita 2023).



Gambar 2. 2 Ilustrasi SVM

Persamaan rumus (2.5) untuk *hyperplane* yang terletak pada *support vector* (Ramlan and al 2023):

$$w.x_i + b = 0 \quad (2.5)$$

Nilai kelas -1 (sampel negatif) dapat dirumuskan dengan menggunakan persamaan rumus (2.6) berikut :

$$w.x_i + b \leq -1 \quad (2.6)$$

Nilai kelas 1 (sampel positif) dapat diwakili sebagai nilai yang memenuhi persamaan rumus (2.7) berikut :

$$w.x_i + b \geq 1 \quad (2.7)$$

Keterangan

w = bobot vector

xi = data ke-i

b = bias

SVM terdiri dari linear dan non-linear. SVM linear yaitu ketika data dapat dipisahkan dengan sempurna oleh sebuah garis linear. Sedangkan SVM non-linear yaitu jika data tidak dapat dipisahkan dengan sempurna secara linear. SVM non-linear akan menggunakan teknik kernel untuk mengubah data ke dalam dimensi fitur yang lebih tinggi dengan tujuan meningkatkan struktur data dan mempermudah proses pemisahan, sehingga menggunakan fungsi kernel di SVM akan menghasilkan hasil yang lebih akurat (Vina and al 2023). Jenis kernel dapat dilihat pada Tabel 2.1 berikut.

Tabel 2. 1 Jenis Kernel SVM

Jenis Kernel	**Model**
Linear	$K(x.x') = x.x'$
Polynomial	$K(x.x') = (x.x' + c)'$
RBF Gaussian	$K(x.x') = \exp(-\gamma\|x - x'\|^2)$
Sigmoid	$K(x.x') = \tanh(ax.x' + \beta)$

Sumber: (Isnain and al 2021)

1. Kernel RBF

   Kernel RBF menggunakan fungsi Gaussian yang dikenal sebagai fungsi basis radial untuk mengukur kesamaan antara dua vektor input dalam ruang fitur. Persamaan rumus (2.8) merupakan persamaan rumus dari kernel RBF (Karyawati 2023) :

   $$K(x_i, x_j) = \exp(-\gamma|x_i - x_j|^2) \quad (2.8)$$

   Keterangan:

   $K(x_i, x_j)$: Fungsi kernel RBF

   x_i : Data ke-i

   x_j : Data ke-j

   γ : Learning rate/gamma

2. Matrix Hessian

   Persamaan rumus (2.9) merupakan persamaan rumus untuk menghitung Matrix Hessian :

   $$D_{ij} = y_i y_j (K(x_i . x_j) + \lambda^2) \quad (2.9)$$

   Keterangan:

   D_{ij} : Elemen matriks ke i j

   y_y : kelas data ke-i

   y_j : kelas data ke-j

   λ : batas teoritis

3. Nilai Error

   Persamaan rumus (2.10) merupakan persamaan rumus untuk menghitung Nilai Error :

   $$E_i = \sum_{j=1}^{n} \alpha_i \, D_{i,j} \quad (2.10)$$

   Keterangan:

   Ei : Nilai error data ke-i

   D_{ij} : Elemen matriks ke i j

   a_j : Nilai parameter a pada Tabel 3.27

4. Delta Alpha

   Persamaan rumus (2.11) merupakan persamaan rumus untuk menghitung delta alpha :

$$\delta\alpha_i = \min\{\max[\gamma(1 - Ei) - \alpha_i]\, C - \alpha_i \quad (2.11)$$

   Keterangan:

   $\delta\alpha_i$: Nilai Delta Alpha ke-i

   Ei : Nilai error data ke-i

   Selanjutnya adalah menghitung nilai *Alpha* Baru dengan persamaan rumus (2.12) sebagai berikut :

$$a_i = a_i + \delta a_i \quad (2.12)$$

5. Klasifikasi

   Untuk proses klasifikasi 3 kelas menggunakan persamaan rumus (2.13) sebagai berikut (Fitriana, 2020) :

$$h(x) = Argmax(\sum_{t=1}^{n} ai.yi.K(xi,x) + b_{label\,1}; \sum_{t=1}^{n} ai.yi.K(xi,x) + b_{label\,2}; \sum_{t=1}^{n} ai.yi.K(xi,x) + b_{label\,3}) \quad (2.13)$$

   Keterangan:

   $\alpha_i = Nilai\ alpha\ baru$

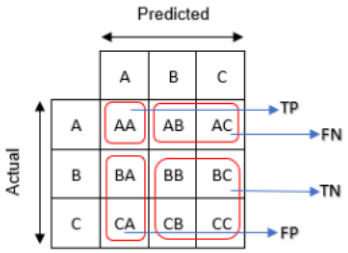
   $y_i = Label\ pada\ data\ train$

   $K(xi,x) = Nilai\ kernel\ dari\ dokumen\ yang\ dipilih$

   b = bias

2.11 Confusion Matrix

Confusion Matrix digunakan untuk menilai klasifikasi pada metode yang dipakai memiliki label yang baik atau buruk. Komponen yang dinilai adalah *accuracy*, *recall*, *precision*, dan F1-s*core* (Basar, Ratnawati, and Arwani 2022). Ilustrasi untuk *confusion matrix* bisa dilihat pada Gambar 2.3 (Wabang, Nurhayati, and Farikhin 2021).



Gambar 2. 3 Confusion Matrix 3x3

Nilai *True Positive* (TP) dan *True Negative* (TN) adalah hasil klasifikasi yang benar. Nilai *False Positive* (FP) adalah nilai yang hasilnya diprediksi positif tetapi sebenarnya negatif, sedangkan Nilai *False Negative* (FN) adalah nilai yang hasilnya diprediksi negatif tetapi sebenarnya positif (Fikri, Sabrila, and Azhar 2020). Berikut merupakan penjabaran (Pamungkas, Aridinanti, and Wibowo 2022) dan rumus perhitungan akurasi, *precision, recall* dan f1-*score* (Wabang, Nurhayati, & Farikhin, 2021) :

1. *Accuracy* adalah total persentase prediksi yang dinyatakan benar dengan keseluruhan data. Persamaan akurasi dapat dilihat pada persamaan rumus (2.14).

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.14}$$

2. *Precision* adalah total persentase prediksi yang memang benar bernilai positif dengan total jumlah data yang terprediksi positif. Persamaan *precision* dapat dilihat pada (2.15).

$$Precision = \frac{TP}{TP + FP} \tag{2.15}$$

3. *Recall* adalah total persentase prediksi yang memang benar bernilai positif dengan total jumlah data yang memang benar positif. Persamaan *recall* dapat dilihat pada (2.16).

$$Recall = \frac{TP}{TP + FN} \tag{2.16}$$

4. F-1 *score* adalah perbandingan rata-rata *precision* dan *recall.* Persamaan f1-*score* dapat dilihat pada (2.17).

$$F1 - Score \ = 2 \times \frac{Recall \times Precision}{Recall + Precison} \tag{2.17}$$

2.12 Google Colaboratory

Colaboratory atau yang biasa dikenal sebagai "*Colab*," merupakan produk *Google* yang bisa digunakan oleh siapa saja untuk menulis, mengeksekusi dan membagikan kode *Python* dengan menggunakan *browser*. Secara teknis *Google Colab* adalah layanan *notebook Jupyter* dengan ekstensi .ipynb yang di hosting dan dapat digunakan tanpa memerlukan instalasi khusus sehingga bisa langsung diakses di http://colab.research.google.com. *Google Colab* dapat membagikan *file* secara online dengan terhubung ke *Google Drive* dan dapat mengakses serta memilih sumber daya komputasi seperti CPU atau GPU secara gratis. Fitur-fitur pada ini membuat *Google Colab* cocok digunakan untuk kegiatan pembelajaran dan penelitian karena mudah diakses, proses pengolahan data yang cepat dan menghemat waktu serta biaya (Soen, Marlina, and Renny 2022).

2.13 Penelitian yang Relevan

Penelitian yang relevan digunakan sebagai bahan referensi dalam penelitian ini melakukan analisis sentimen IndiHome di X (*Twitter)* menggunakan metode *FastText* dan *Support Vector Machine* (SVM).

1. Penelitian yang dilakukan oleh (Darwis, Pratiwi, & Pasaribu, 2020) menggunakan metode SVM untuk menganalisis sentimen data Komisi Pemberantasan Korupsi (KPK) di Twitter. Banyak orang menggunakan Twitter untuk menyampaikan pendapat mereka tentang upaya KPK dalam memerangi korupsi. Studi ini menggunakan SVM dan ekstraksi TF-IDF. Penelitian ini mengekstrak 1890 data dan 3846 kata atau term dari 2000 data crawling Twitter. Kemudian mereka menghitung nilai kemunculan kata untuk labeling, yang menghasilkan sentimen netral, positif, dan negatif. Hasil pengujian menunjukkan bahwa metode SVM memberikan nilai akurasi sebesar 82%, sentimen dengan label negatif sebesar 77%, sentimen dengan label positif sebesar 8%, dan sentimen dengan label netral sebesar 25%.
2. Penelitian yang dilakukan oleh (Idris, Mustofa, and Salihi 2023) menggunakan algoritma SVM untuk menganalisis sentimen pengguna aplikasi *Shopee*. Tujuan penelitian ini adalah untuk mengklasifikasikan data komentar pengguna aplikasi

Shopee ke dalam komentar positif dan negatif dengan melihat ulasan pengguna tentang aplikasi Shopee dan untuk mengetahui seberapa baik metode pengklasifikasi yang digunakan. Tiga ribu data ulasan dibuat setelah dataset dikumpulkan melalui teknik scraping. Dengan skor f1 sebesar 0.98, atau 98% dan akurasi sebesar 98%, hasil penelitian yang menggunakan algoritma SVM menunjukkan kinerja yang cukup baik.

3. Penelitian yang dilakukan oleh (Asshiddiqi and Lhaksmana 2020) menggunakan algoritma SVM dengan kernel RBF dan *word embedding* untuk menganalisis sentimen judul berita terkait Pemilihan Presiden 2024 di Indonesia. Tujuan dari penelitian ini adalah untuk memahami opini *public* dan atmosfer politik selama masa kampanye pemilihan presiden serta membantu pembaca dalam memperoleh informasi yang relevan dari judul berita. Pada penelitian ini dilakukan percobaan untuk membandingkan kinerja algoritma SVM dalam melakukan klasifikasi sentimen dengan *word embedding* TF-IDF dan FastText. Penelitian ini menggunakan kernel rbf dan linear dengan 3 perbandingan data uji yaitu 70:30, 80:20 dan 90:10. Kernel yang digunakan adalah RBF dan linear. Nilai C yang digunakan adalah 0.1, 1, dan 10 dan nilai gamma yang digunakan adalah 0.1, 1, auto, dan *scale*. Hasil dari penelitian menunjukkan bahwa FastText mampu memberikan performa yang lebih baik dibandingkan TF-IDF sehingga dapat memberikan skor akurasi dan skor F1 yang lebih baik. Data dengan rasio 70:30 memiliki akurasi 98% dengan nilai F1-score 99%, data dengan rasio 80:20 memiliki akurasi 98% dengan nilai F1-score 98% dan data dengan rasio 90:10 memiliki akurasi 99% dengan nilai F1-score sebesar 99% .
4. Penelitian yang dilakukan oleh (Mustasaruddin, Fikry, and Yanto 2023) mengklasifikasikan ulasan pada aplikasi *MyPertamina* dengan metode SVM dan *word embedding* menggunakan FastText. Aplikasi ini menerima banyak ulasan dan komentar masyarakat, baik positif maupun negatif. Diharapkan evaluasi dan penilaian ini akan membantu pemerintah menerapkan program. Oleh karena itu, tujuan penelitian ini adalah untuk menilai aplikasi MyPertamina dengan menggunakan pengelompokan kelas sentimen 90:10, 80:20, dan 70:30. Porsi dataset yang digunakan adalah 90:10, 80:20, dan 70:30. Dengan 7200 data pelatihan dan 800 data pengujian, model SVM terbaik dibuat dengan porsi data

90:10. Tanpa menggunakan *undersampling*, diperoleh akurasi 80%, recall 50%, dan ketepatan 84%.

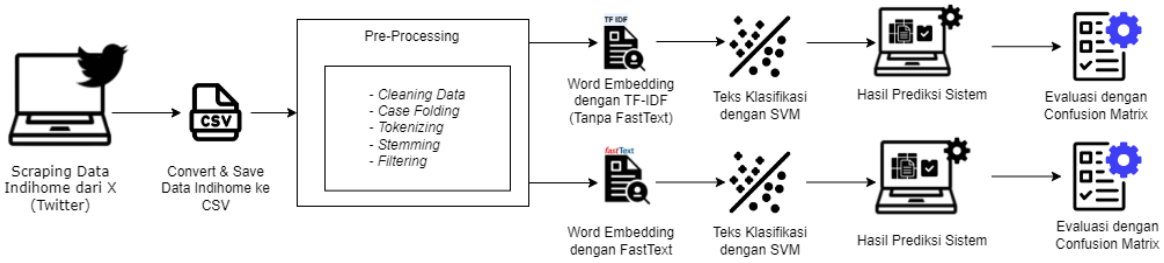
5. Penelitian yang dilakukan oleh (Zikri and Agustian 2023) melakukan deteksi *hate speech* dan *abusive* di Twitter. Kecenderungan negatif yang sering muncul di media sosial termasuk ujaran kebencian (*hate speech*) dan bahasa kasar (*abusive language*). Banyak pengguna mengunggah ujaran kebencian atau bahkan bahasa kasar, seperti di *Twitter*, yang menyebabkan konflik di *platform* media sosial. Oleh karena itu, untuk menentukan apakah sebuah *tweet* mengandung ujaran kebencian atau bahasa kasar, model klasifikasi menggunakan SVM dengan *word embedding* FastText. Penelitian ini bertujuan untuk meningkatkan kinerja metode *baseline* SVM dengan fitur *word embeddings* FastText. Pengujian yang dilakukan dengan model SVM yang paling *optimal* menunjukkan akurasi rata-rata 82,65% untuk kelas *hate speech, abusive language*, dan *hate speech*, masing-masing 84,92%, 86,60%, dan 76,43%.

BAB 3

ANALISIS DAN PERANCANGAN

3.1 Arsitektur Sistem

Arsitektur sistem merupakan gambaran keseluruhan sistem dimulai dari *input,* proses hingga *output* yang menghasilkan akurasi model analisis sentimen pada algoritma *FastText* dan SVM. General arsitektur sistem dapat dilihat pada Gambar 3.1.



Gambar 3.1 General Arsitektur Sistem

Dari Gambar 3.1 dapat dijelaskan bahwa hal yang pertama dilakukan adalah meng-*input* dataset *tweet* IndiHome yang didapatkan setelah melakukan *web scraping.* Selanjutnya, dilakukan *pre-processing* yang bertujuan untuk membersihkan dan mempersiapkan teks pada dataset agar memudahkan proses klasifikasi. Pada tahap *pre-processing* terdapat beberapa tahap yaitu *cleaning* data, *case folding, tokenizing, stopword removal* dan *stemming.* Setelah *pre-processing* dilakukan pelabelan data apakah positif maupun negatif dan dilakukan *split* data menjadi data *training* dan data *testing.* Tahap selanjutnya, jika menggunakan FastText akan dilakukan proses *word embedding* menggunakan FastText*,* jika tidak akan menggunakan TF-IDF. Setelah proses penentuan bobot atau *embedding* selanjutnya langsung ke pembentukan model SVM untuk klasifikasi teks. Sehingga dihasilkan klasifikasi SVM dengan FastText dan SVM tanpa FastText. Tahap selanjutnya, klasifikasi data *testing* menggunakan data *training* yang akan menghasilkan hasil prediksi label oleh sistem. Terakhir, hasil

prediksi ini akan dievaluasi menggunakan *confusion matrix,* sehingga menghasilkan nilai analisis sentimen SVM dengan FastText dan tanpa FastTex

3.2 Input *Dataset*

Dataset yang digunakan berasal dari web scraping sosial media X (Twitter) dengan *keyword* "Indihome" yang diekstrak menggunakan *library* Selenium dan Streamlit pada Python. Scraping data menghasilkan 1537 data dan 4 *feature*. Tabel 3.1 merupakan sampel dataset hasil scraping.

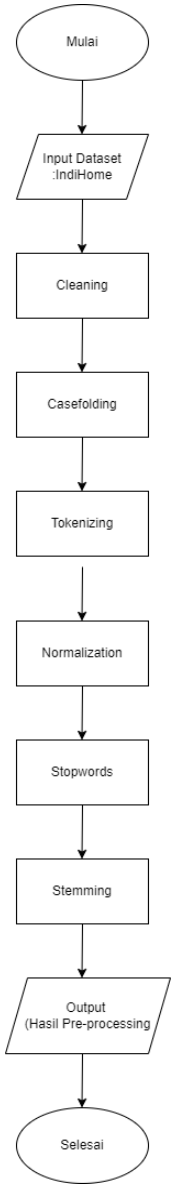
Tabel 3.1 Dataset

Index	Username	Date	Tweet
0	Fikel go	28/02/2023	Harimu Hampa? IndiHome Aja! Dijamin hari kamu tidak hampa lagi, karena bisa nonton Netflix, Disney Hotstar, HBO Go, Vidio dan Mola dengan IndiHome. Langganan sekarang di http://indihome.co.id @indihomeborneo #AktivitasTanpaBatas #internetnyaindonesia
1	IndiHomeCare	28/02/2023	Halo Kak Yani, apakah ada yang bisa Yurai bantu mengenai produk dan layanan IndiHome nya. Oh iya Kak untuk informasi dan laporan Kakak bisa menghubungi kami melalui DM di akun Twitter @IndiHomeCare atau klik Link : http://myih.ch/dm. Terima Kasih-Yurai

2	maung congkok	28/02/2023	@IndiHome ppppp
…	…	…	
…	…	…	
…	…	…	
			Ternyata kelakuan aneh bin ajaib keluarga Tamu Harap Sabar gak cuma di tayangan aja, di balik layar juga mereka sekocak itu kelakuannya!
1536	IndiHome	26/02/2024	Yuk tonton Tamu Harap Sabar, setiap hari Sabtu jam 16.00 WIB di YouTube IndiHome. #IndiHomebyTelkomsel #DariRumahTanpaBatas

3.3 *Pre-processing*

Pre-processing merupakan tahap penyiapan data sebelum diimplementasikan ke dalam suatu algoritma agar mendapatkan hasil yang lebih baik. Pada penelitian ini setelah tahap *input* data dilakukan *pre-processing* sebelum lanjut ke tahapan berikutnya. *Flowchart* tahap *pre-processing* dapat dilihat pada Gambar 3.2.



Gambar 3.2 *Flowchart Pre-Processing*

1. Input Dataset

   Dari 1537 data hasil scraping, diambil 9 data sampel dan tweet sebagai *feature* yang digunakan sebagai inputan, Tabel 3.2 merupakan sampel dataset yang digunakan dalam proses *pre-processing* ini.

Tabel 3. 2 Sampel Dataset

Index	Tweet
0	Semua mudah dengan IndiHome #IndiHome #AktifitasTanpaBatas #SambutHarapanBaru
1	Tiap jam segini pasti bapuk, heran @IndiHome
2	Eh serius? Becandaannya sungjin anak indihome ternyata beneran?
3	indihome kenapa lelet
4	Indihome trouble? @IndiHome
5	indihome tolol
6	tele, my indihome
7	sebel bgt sama Indihome lemottttttt
8	Pensi semenjak ganti wifi ke indihome

2. *Cleaning Data*

Setelah meng-input data selanjutnya data akan dibersihkan dengan menghapus simbol-simbol, digit angka dan emoticon pada teks. Tabel 3.3 merupakan perbandingan data sebelum dan sesudah dilakukan *cleaning*.

Tabel 3.3 Perbandingan data sebelum dan sesudah *Cleaning*

Index	Sebelum Cleaning	Sesudah Cleaning
0	Semua mudah dengan IndiHome #IndiHome #AktifitasTanpaBatas #SambutHarapanBaru	Semua mudah dengan IndiHome
1	Tiap jam segini pasti bapuk, heran @IndiHome	Tiap jam segini pasti bapuk heran
2	Eh serius? Becandaannya sungjin anak indihome ternyata beneran?	Eh serius Becandaannya sungjin anak indihome ternyata beneran
3	indihome kenapa lelet	indihome kenapa lelet

4	Indihome trouble? @IndiHome	Indihome trouble
5	indihome tolol	indihome tolol
6	tele, my indihome	tele my indihome
7	sebel bgt sama Indihome lemottttttt	sebel bgt sama Indihome lemottttttt
8	Pensi semenjak ganti wifi ke indihome	Pensi semenjak ganti wifi ke indihome

3. *Case Folding*

Tahap *case folding* bertujuan untuk mengubah semua isi teks menjadi huruf kecil. Tabel 3.4 merupakan perbandingan data hasil *cleaning* dan sesudah proses *case folding*.

Tabel 3.4 Dataset proses *Case Folding*

Index	**Hasil Cleaning**	**Sesudah Case Folding**
0	Semua mudah dengan IndiHome	semua mudah dengan indihome
1	Tiap jam segini pasti bapuk heran	tiap jam segini pasti bapuk heran
2	Eh serius Becandaannya sungjin anak indihome ternyata beneran	eh serius becandaannya sungjin anak indihome ternyata beneran
3	indihome kenapa lelet	indihome kenapa lelet
4	Indihome trouble	indihome trouble
5	indihome tolol	indihome tolol
6	tele my indihome	tele my indihome
7	sebel bgt sama Indihome lemottttttt	sebel bgt sama indihome lemottttttt
8	Pensi semenjak ganti wifi ke indihome	pensi semenjak ganti wifi ke indihome

4. *Tokenizing*

Tahap *tokenizing* untuk pemisahan kata berdasarkan spasi. Tabel 3.5 merupakan perbandingan data hasil *case folding* dan sesudah proses *tokenizing*.

Tabel 3.5 Proses *Tokenizing*

Index	Hasil Casefolding	Sesudah Tokenizing
0	semua mudah dengan indihome	semua mudah dengan indihome
1	tiap jam segini pasti bapuk heran	tiap jam segini pasti bapuk heran
2	eh serius becandaannya sungjin anak indihome ternyata beneran	serius becandaannya sungjin anak indihome ternyata beneran
3	indihome kenapa lelet	indihome kenapa lelet
4	indihome trouble	indihome trouble
5	indihome tolol	Indihome tolol
6	tele my indihome	tele indihome
7	sebel bgt sama indihome lemottttttt	sebel", "bgt", "sama", "indihome lemottttttt
8	pensi semenjak ganti wifi ke indihome	pensi semenjak ganti wifi indihome

5. *Normalization*

Normalization adalah tahap mengubah bahasa tidak baku, bahasa gaul (*slang)* dan kata singkatan menjadi bahasa baku. Tabel 3.6 merupakan perbandingan data hasil *tokenizing* dan sesudah proses *normalization.*

Tabel 3. 6 Proses Normalization

Index	Hasil Tokenizing	Sesudah Normalization
0	semua mudah dengan indihome	semua mudah dengan indihome
1	tiap jam segini pasti bapuk heran	setiap jam segini pasti bapuk heran
2	serius becandaannya sungjin anak indihome ternyata beneran	serius becandaannya sungjin anak indihome ternyata betulan
3	indihome kenapa lelet	indihome kenapa lelet
4	indihome trouble	indihome trouble
5	indihome tolol	indihome tolol
6	tele indihome	tele indihome

7	sebel bgt sama indihome lemottttttt	sebel begitu sama indihome lemot
8	pensi semenjak ganti wifi indihome	pensi semenjak ganti wifi indihome

6. *Stopwords*

Tahap ini untuk menghapus kata hubung yang sering muncul dan tidak penting. Pada penelitian ini penulis menggunakan stopwords tambahan yang dibuat secara manual mengacu pada dataset yang digunakan. Tabel 3.7 merupakan perbandingan data hasil *normalization* dan sesudah proses *stopwords*.

Tabel 3.7 Proses *Stopwords*

Index	**Hasil Normalization**	**Sesudah Stopwords (Filtering)**
0	semua mudah dengan indihome	semua mudah indihome
1	setiap jam segini pasti bapuk heran	setiap jam segini pasti bapuk heran
2	serius becandaannya sungjin anak indihome ternyata betulan	becandaannya sungjin anak indihome ternyata betulan
3	indihome kenapa lelet	indihome kenapa lelet
4	indihome trouble	indihome trouble
5	indihome tolol	indihome tolol
6	tele indihome	tele indihome
7	sebel begitu sama indihome lemot	sebel indihome lemot
8	pensi semenjak ganti wifi indihome	pensi semenjak ganti wifi indihome

7. *Stemming*

   Tahap *stemming* untuk mengubah setiap kata yang memiliki imbuhan menjadi kata dasar. Tabel 3.8 merupakan perbandingan data hasil *stopwords* dan sesudah proses *stemming*.

Tabel 3.8 Proses *Stemming*

Index	Hasil Stopwords (Filtering)	Sesudah Stemming
0	semua mudah indihome	semua mudah indihome
1	setiap jam segini pasti bapuk heran	tiap jam gin pasti bapuk heran
2	becandaannya sungjin anak indihome ternyata betulan	becandaannya sungjin anak indihome nyata betul
3	indihome kenapa lelet	indihome kenapa lelet
4	indihome trouble	indihome trouble
5	indihome tolol	indihome tolol
6	tele indihome	tele indihome
7	sebel indihome lemot	sebel indihome lot
8	pensi semenjak ganti wifi indihome	pensi semenjak ganti wifi indihome

8. *Output Pre-processing*

   Setelah melalui semua tahapan proses *pre-processing* untuk klasifikasi teks, didapatkan output *pre-processing* pada Tabel 3.9 berikut :

Tabel 3.9 Hasil *Pre-processing*

Index	Hasil Pre-processing
0	semua mudah indihome
1	tiap jam gin pasti bapuk heran
2	becandaannya sungjin anak indihome nyata betul
3	indihome kenapa lelet
4	indihome trouble
5	indihome tolol
6	tele indihome

7	sebel indihome lot
8	pensi semenjak ganti wifi indihome

3.4 Pelabelan Data

Proses pelabelan data menggunakan TextBlob. TextBlob akan melakukan perhitungan yang menghasilkan nilai polaritas pada setiap indeks kata. Jika nilai polaritas nilai polaritas > 0, maka bernilai positif. Jika nilai polaritas = 0, maka bernilai netral. Jika nilai polaritas < 0, maka bernilai negatif. Pada TextBlob hanya dapat memproses kata dalam Bahasa Inggris sehingga untuk memproses kata dalam Bahasa Indonesia perlu dilakukan penerjemahan dari Bahasa Indonesia ke Bahasa Inggris (Mas Diyasa et al. 2021). Berdasarkan pernyataan tersebut, dataset Bahasa Indonesia di terjemahkan terlebih dahulu ke Bahasa Inggris. Tabel 3.10 adalah hasil terjemahan menggunakan Python.

Tabel 3. 10 Hasil *Translate*

Index	**Translate**
0	all easily indihome
1	every hour gin must be surprised
2	the jokes of the indihome child sungjin are real
3	indihome why slow
4	indihome trouble
5	stupid indihome
6	tele indihome
7	resented indihome lot
8	pensi since replacing indihome wifi

Berdasarkan pernyataan tersebut, berikut adalah contoh perhitungan TextBlob pada setiap Data Index ke-0, Data Index ke-3 dan Data Indexs ke-6 :

$$Data\ Index_0 = if\ (4.3E{+}16) > 0\ \rightarrow\ positif$$

$$Data\ Index_3 = if\ (-3E{+}16) < 0\ \rightarrow negatif$$

$$Data\ Index_6 = if\ (0) = 0\ \rightarrow netral$$

Hasil dari *polarity score* didapatkan dari proses Textblob pada program. Tabel 3.11 adalah hasil dari Labeling menggunakan Textblob menggunakan Python.

Tabel 3. 11 Hasil *Labeling*

Index	Hasil Pre-processing	Polarity Score	Sentimen	Label
0	all easily indihome	4.3E+16	Positif	1
1	every hour gin must be surprised	1	Positif	1
2	the jokes of the indihome child sungjin are real	2	Positif	1
3	indihome why slow	-3E+16	Negatif	-1
4	indihome trouble	-2	Negatif	-1
5	stupid indihome	-8E+15	Negatif	-1
6	tele indihome	0	Netral	0
7	resented indihome lot	0	Netral	0
8	pensi since replacing indihome wifi	0	Netral	0

3.5 *Split Data*

Split data yang digunakan dalam penelitian ini adalah 80%:20%. Penelitian yang dilakukan (Gunawan 2021) melakukan perbandingan split data 80%:20% dan 70%:30% mendapatkan hasil akurasi untuk 80%:20% yaitu 70% akurasi, sedangkan split data 70%:30% mendapatkan hasil 69% akurasi. Maka dari itu Split data yang digunakan pada penelitian ini adalah 80%:20%. Tabel 3.12 adalah Data Train dan Tabel 3.13 adalah Data Test yang digunakan.

Tabel 3. 12 Data Train

Data Train		
Index	**Data**	**Label**
0	all easily indihome	1
1	every hour gin must be surprised	1

3	indihome why slow	-1
4	indihome trouble	-1
6	tele indihome	0
7	resented indihome lot	0

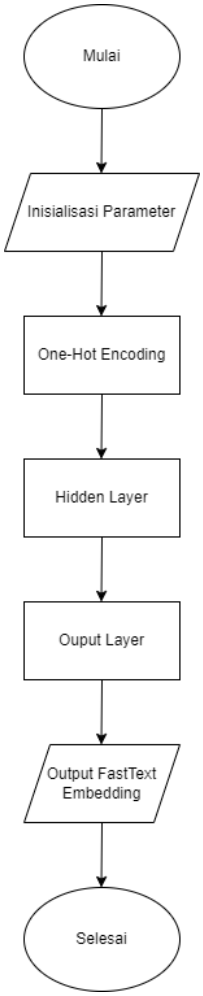
Tabel 3.13 Data Test

Data Tes		
Index	**Data**	**Label**
2	the jokes of the indihome child sungjin are real	1
5	stupid indihome	-1
8	pensi since replacing indihome wifi	0

3.6 *Word Embedding*

3.6.1 *Word Embedding Menggunakan Fasttext*

Gambar 3.3 merupakan tahapan dari implementasi FastText menggunakan arsitektur FastText dengan tipe Skip-gram. Skip-gram dipilih karena lebih cocok digunakan pada dataset yang berukuran kecil dan dapat merepresentasikan kata-kata yang jarang muncul dengan lebih baik. Berikut *flowchart* FastText dengan tipe skipgram pada Gambar 3.3.



Gambar 3.3 *Flowchart FastText*

1. Inisialisasi Parameter

   Untuk menghitung *Hidden layer* dan *Output layer*, diperlukan inisialisasi parameter. Index 0…14 dari Tabel 3.12 Data Train adalah nilai inisialisasi dari bobot *hidden layer*, bobot ini akan digunakan untuk perhitungan pada *hidden layer*. Untuk penentuan nilai W merupakan nilai *random* dengan *range* -1 sampai 1 (Rifaldo 2021). Tabel 3.14 adalah inisialisasi parameter yang digunakan untuk perhitungan Hidden layer

Tabel 3.14 Bobot *Hidden layer*

Index	0	1	…	14
W	-0,256	0,689	…	-0.193

Tabel 3.15 Bobot Output layer

Bobot	
n-grams	3

2. *One-Hot Encoding*

   Proses *One-Hot Encoding* menggunakan data hasil pre-processing pada Tabel 3.8 atau Tabel 3.11 dan Tabel 3.12. Tabel 3.16 dan Tabel 3.17 merupakan kata hasil dari pre-processing.

Tabel 3.16 Kata hasil pre-processing (Data Training)

Index	***Words***
0	All
1	Easily
2	Indihome
…	…
14	Lot

Tabel 3.17 Kata hasil pre-processing (Data Testing)

Index	***Words***
0	The
1	Jokes
2	Of
…	…
12	Wifi

Setelah mendapatkan setiap kata dari hasil pre-processing, maka selanjutnya adalah melakukan proses *one-hot encoding* (Khomsah 2021) dari Tabel 3.16 dan Tabel 3.17.

Tabel 3.18 Hasil *One-Hot Encoding* (Data Training)

Index	**Words**	**Encoding**			
		0	**1**	**…**	**14**
0	All	1	0	…	0
1	Easily	0	1	…	0
…	…	…	…	…	…
14	Lot	0	0	…	1

Pada Tabel 3.18, index 0 hingga 14 mengacu terhadap index setiap token (*words*) dan nilai 0 dan 1 menyatakan hasil encodernya, dimana nilai 1 menunjukkan keberadaan encoder terhadap kata tersebut dan nilai 0 mengacu ke hasil encoding yang tidak sesuai dengan index token. Contoh index 0 pada kolom Index dan index 0 pada kolom Encoding, memiliki nilai 1 sehingga index 1 hingga 14 memiliki nilai 0.

Tabel 3.19 Hasil *One-Hot Encoding* (Data Testing)

Index	**Words**	**Encoding**			
		0	**1**	**…**	**12**
0	The	1	0	…	0
1	Jokes	0	1	…	0
…	…	…	…	…	…
12	Wifi	0	0	…	1

Pada Tabel 3.19, index 0 hingga 12 mengacu terhadap index setiap token (words) dan nilai 0 dan 1 menyatakan hasil encodernya, dimana nilai 1 menunjukkan keberadaan encoder terhadap kata tersebut dan nilai 0 mengacu ke hasil encoding yang tidak sesuai dengan index token. Contoh index 0 pada kolom Index dan index 0 pada kolom Encoding memiliki nilai 1 sehingga index 1 hingga 12 memiliki nilai 0.

3. Hidden Layer

Berdasarkan persamaan rumus (2.1) tersebut, berikut adalah persamaan rumus untuk menghitung Hidden layer pada *Index* 0 “all” :

$$z_{inj} = V_{0j} + \sum_{i=1}^{n} x_i v_{ij}$$

$$z_{all} = 0 + \sum 1 * (-0{,}256) + 0 * 0{,}689 + .. + 0 * (\text{-}0.193) = -0{,}256$$

Perhitungan tersebut dilakukan terhadap semua index kata, Tabel 3.20 adalah hasil dari perhitungan hidden layer data train dan Tabel 3.21 adalah hasil hidden layer data test (OOV).

Tabel 3.20 Hasil Hidden Layer Data Train

Index	**Words**	**Hidden Layer**
0	All	-0,256
1	Easily	0,689
2	Indihome	0,124
…	…	…
14	Lot	-0,193

Tabel 3.21 Hasil Hidden Layer Data Tes (OOV)

Index	**Words**	**Hidden Layer**
0	The	0,456
1	Jokes	-0,789
2	Of	0,123
…	…	…
12	Wifi	-0,193

4. Output Layer

Selanjutnya adalah menghitung output layer menggunakan persamaan rumus (2.2) dan (2.3). Jika kata yang akan dihitung tidak ada di dalam kamus atau data training, kata dianggap keluar dari kosakata (OOV), maka vektor kata adalah total dari n-gram (Pardede and Pakpahan 2023).

Berdasarkan persamaan kedua rumus tersebut, berikut rumus menghitung FastText pada Index ke-0 yaitu kata “all”:

$$FastText_{word} = V_{word} + \sum g \in ngrams(word)$$

$$FastText_{all} = -0{,}256 + \sum 3\,(-0{,}256) = -1{,}024$$

Selanjutnya berikut adalah untuk menghitung OOV pada kata “the”:

$$FastText_{oov\_word} = \sum g \in ngrams(oov\_word)$$

$$FastText_{oov_{word},may} = \sum 3\,(0{,}456) = 1{,}368$$

Perhitungan tersebut dilakukan terhadap semua Index kata, Tabel 3.22 dan Tabel 3.23 adalah hasil dari perhitungan output layer dari FastText pada data training dan data testing (OOV).

Tabel 3.22 Hasil *Output layer*

Index	**Words**	**Output Layer**
0	All	-1,024
1	Easily	1,811
2	Indihome	0,116
…	…	…
14	Lot	-0,835

Tabel 3.23 Hasil *Output layer OOV*

Index	**Words**	**Output Layer**
0	The	1,368
1	Jokes	-2,367
2	Of	0,369
…	…	…
12	Wifi	-0,579

5. Output

Selanjutnya token kata tersebut menjadi input di dalam jaringan saraf skip-gram untuk menghasilkan nilai probabilitas konteks (Pardede and Pakpahan 2023). Angka token didapat dari index ke berapa kata tersebut, contohnya pada index ke-0 data train terdapat kata “all”, “easily” maka pada index ke-0 data train memiliki token 0, 1 atau bisa disebut juga index dari kata tersebut. Sesuai pada Tabel 3.20, dikarenakan kata “all” terdapat pada index ke-0, kata “easily” terdapat pada index ke-1. Tabel 3.24 dan Tabel 3.25 adalah hasil dari tahap token.

Tabel 3.24 Hasil *Token Data Train*

Data ke-0	Data ke-1	Data ke-2	Data ke-3	Data ke-4	Data ke-5
0	3	2	2	12	13
1	4	9	11	2	2
Label					
1	1	-1	-1	0	0

Tabel 3.25 Hasil *Token Data Tes*

Data ke-0	Data ke-1	Data ke-2
0	8	9
1	3	10
Label		
1	-1	0

Selanjutnya adalah Embedding yaitu, pembobotan tiap kata pada setiap data train dan data testing. Contohnya pada index ke-0 data train nilai token adalah 1 dan 2 maka dari token tersebut menjadi output FastText berdasarkan nilai token, contoh pada index ke-0 data train terdapat Token 1 bernilai -1,024, Token 2 memiliki nilai 1,811 dan seterusnya. Lalu dihasilkan pembobotan pada Tabel 3.26 dan Tabel 3.27.

Tabel 3.26 Hasil *Embedding* menggunakan *FastText* Pada *Data Train*

Data ke-0	Data ke-1	Data ke-2	Data ke-3	Data ke-4	Data ke-5
-1,024	-2,791	0,116	0,116	1,901	-2,119
1,811	-1,69	-3,037	-0,415	0,116	0,116

Tabel 3.27 Hasil *Embedding* menggunakan *Fasttext* pada *Data Tes*

Data ke-0	Data ke-1	Data ke-2
1,368	1,035	-0,036
-2,367	-1,701	1,368

3.6.2 *Word Embedding Menggunakan TF-IDF*

1. Menentukan Nilai TF-IDF

   Berdasarkan rumus (2.5), berikut adalah contoh perhitungan TF-IDF sebelum dilakukan normalisasi.

$$IDF = \ln\left(\frac{1+D}{1+DF}\right) + 1$$

$$IDF\ all = \ln\left(\frac{1+6}{1+6}\right) + 1 = \ 2{,}253$$

$$IDF\ easily = \ln\left(\frac{1+6}{1+6}\right) + 1 = 2{,}253$$

$$TF - IDF\ all, D0 = 1 * \ 2{,}253 = 2{,}253$$

$$TF - IDF\ easily, D0 = 1 * \ 2{,}253 = \ 2{,}253$$

Tabel 3. 28 Hasil TF-IDF Sebelum Normalisasi pada Data Train

Index	words	TF						DF(t)	IDF	TF-IDF					
		D0	D1	D2	D3	D4	D5			D0	D1	D2	D3	D4	D5
0	All	1	0	0	0	0	0	1	2,253	2,253	0	0	0	0	0
1	Easily	1	0	0	0	0	0	1	2,253	2,253	0	0	0	0	0
2	indiho me	1	0	1	1	1	1	5	1,154	1,154	0	1,154	1,154	1,154	1,154
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
26	Wifi	0	0	0	0	0	0	0	2,946	0	0	0	0	0	0

Pada tabel 3.28 hasil dari TF-IDF sebelum normalisasi pada data train menghasilkan jumlah kata sebanyak 27 dan 6 baris data, yaitu D0, D1, D2, D3, D4, D5. Perhitungan yang sama juga dilakukan pada data test sehingga menghasilkan pada Tabel 3.29.

Tabel 3. 29 Hasil TF-IDF sebelum Normalisasi pada Data Tes

Index	Words	TF			DF(t)	IDF	TF-IDF		
		D0	D1	D2			D0	D1	D2
0	All	1	1	1	1	2,253	2,253	2,253	2,253
1	Easily	0	0	0	0	2,946	0	0	0
2	Indihome	1	1	1	5	1,154	1,154	1,154	1,154
…	…	…	…	…	…	…	…	…	…
26	Wifi	0	0	1	0	2,946	0	0	2,946

2. Menentukan TF-IDF Normalisasi

Setelah mendapatkan nilai TF-IDF, selanjutnya menghitung L2-Norm dengan rumus (2.6). Berikut adalah contoh perhitungan L2-Norm :

$$L2 - Norm = \sqrt[2]{TF - IDF_1^2 + TF - IDF_2^2 + \cdots + TF - IDF_i^2}$$

$$L2 - Norm, D0 = \sqrt[2]{2{,}253^2 + 2{,}253^2 + \cdots + 0^2} = 3{,}389$$

$$L2 - Norm, D1 = \sqrt[2]{0^2 + 0^2 + \cdots + 0^2} = 5{,}518$$

$$L2 - Norm, D2 = \sqrt[2]{0^2 + 0^2 + \cdots + 0^2} = 3{,}389$$

Setelah mendapatkan nilai L2-Norm pada setiap indeks, maka selanjutnya adalah dilakukan perhitungan TF-IDF L2-Norm dengan rumus (2.7). Berikut adalah contoh perhitungan TF-IDF L2-Norm :

$$TF - IDF\ L2 - Norm = \frac{TF-IDF\ sebelum\ Norm}{L2-Norm}$$

$$TF - IDF\ L2 - Norm, all\ , D0 = \frac{2{,}253}{3{,}389} = 0{,}665$$

$$TF - IDF\ L2 - Norm, easily, D1 = \frac{0}{5{,}518} = 0$$

$$TF - IDF\ L2 - Norm, indihome, D2 = \frac{0}{3{,}389} = 0$$

Setelah dilakukan perhitungan pada seluruh data train dan data test, didapatkan hasil normalisasi data train pada Tabel 3.30 dan hasil normalisasi data test pada Tabel 3.31.

Tabel 3. 30 Hasil Normalisasi TF-IDF Data Train

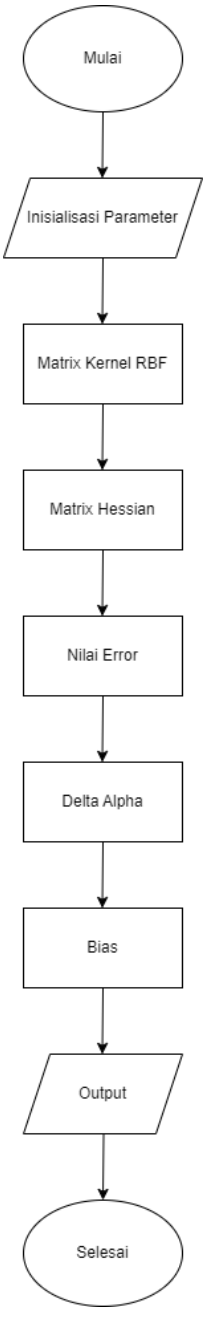
	D0	D1	D2	D3	D4	D5
All	0,665	0	0	0	0	0
Easily	0,665	0	0	0	0	0
Indihome	0,341	0	0,341	0,456	0,456	0,341
...	...	...	...	...	...	...
Wifi	0	0	0	0	0	0

Tabel 3. 31 Hasil Normalisasi TF-IDF Data Tes

	D0	D1	D2
All	0,295	0,580	0,351
Easily	0	0	0
Indihome	0,151	0,297	0,180
…	…	…	…
Wifi	0	0	0,459

3.7 Klasifikasi Teks menggunakan SVM

Metode SVM menggunakan nilai pendukung untuk membedakan kelas data yang berbeda. Pada penelitian ini menggunakan Kernel RBF, dikarenakan pada penelitian yang dilakukan oleh (Rabbani 2023) melakukan penelitian perbandingan Kernel pada SVM dan memperoleh hasil terbaik dengan kernel RBF pada perbandingan Split Data 80%:20% yaitu dengan akurasi 83%, sedangkan kernel Linier mendapatkan hasil akurasi 79% dan *Kernel Polynomial* mendapatkan hasil akurasi 76%. Berikut pada Gambar 3.4 merupakan *flowchart* SVM.



Gambar 3.4 *Flowchart* SVM

3.7.1 Klasifikasi SVM Menggunakan *FastText*

1. Inisialisasi Parameter

   Langkah awal yang dilakukan untuk melakukan klasifikasi dengan SVM adalah inisialisasi parameter untuk nilai α, C, lamda dan gamma (Rabbani 2023). Tabel 3.32 adalah hasil dari inisialisasi nilai untuk setiap parameternya.

Tabel 3.32 Inisialisasi Parameter

Parameter	Nilai	Keterangan
A	0.5	>=0 (Purnamasari et al. 2023)
C	2	> 0 (Purnamasari et al. 2023)
Λ	0.5	> 0 (Averina, Santoso, and Yudistira 2020)
Gamma	2	> 0 (Saadah, Z, and Z 2021)

2. Menghitung Kernel RBF

   Fungsi kernel non-linier sering digunakan untuk mengubah data inputan ke ruang fitur berdimensi tinggi di mana data input menjadi lebih dapat dipisahkan dari ruang input asli (Karyawati 2023). Berdasarkan persamaan rumus (2.11), berikut perhitungan kernel RBF pada data train ke-0 yang terdapat pada Tabel 3.32:

$$K(x_i, x_j) = \exp\left(-\gamma|x_i - x_j|^2\right)$$

$$K(x_0, x_0) = \exp\left(-2|\left(-1{,}024 - (-1{,}024)\right) * (1{,}811 - 1{,}811)\right)^2$$

$$K(x_0, x_0) = 1$$

$$K(x_0, x_1) = \exp\left(-2|(-1{,}024 - 1{,}901) * (1{,}811 - (-2{,}119))\right)^2$$

$$K(x_0, x_1) = 0$$

$$K(x_0, x_2) = \exp\left(-2|(-1{,}024 - 1{,}250) * \left(1{,}811 - (-1{,}264)\right)\right)^2$$

$$K(x_0, x_2) = 0$$

$$K(x_0, x_3) = \exp\left(-2|\left(-1{,}024 - (-1{,}024)\right) * (1{,}811 - 1{,}811)\right)^2$$
$$K(x_0, x_3) = 0$$

$$K(x_0, x_4) = \exp\left(-2|(-1{,}024 - 0{,}698) * (1{,}811 - (-2{,}878))\right)^2$$
$$K(x_0, x_4) = 0$$

$$K(x_0, x_5) = \exp\left(-2|\left(-1{,}024 - (-1{,}024)\right) * (1{,}811 - 1{,}811)\right)^2$$
$$K(x_0, x_5) = 0$$

Tabel 3.33 Hasil perhitungan kernel RBF

Index	Data Train ke-0	Data Train ke-1	Data Train ke-2	Data Train ke-3	Data Train ke-4	Data Train ke-5
Data Train ke-0	1	0	0	0	0	0
Data Train ke-1	0	1	0,000	0	0,000	0,001
Data Train ke-2	0	0,000	1	0,000	0,000	0
Data Train ke-3	0,000	0	0,000	1	0,001	0,000
Data Train ke-4	0	0,000	0,000	0,001	1	0
Data Train ke-5	0	0,001	0	0,000	0	1

3. Matrix Hessian

   Berdasarkan persamaan rumus (2.12), berikut contoh perhitungan dalam menghitung Matrix Hessian pada data train ke-0 :

$$D_{ij} = y_i y_j (K(x_i . x_j) + \lambda^2)$$
$$D_{0,0} = 1 * (1)(K(1) + 0.5^2) = 1{,}250$$
$$D_{0,1} = 1 * (1)(K(0) + 0.5^2) = 0{,}250$$
$$D_{0,2} = 1 * (-1)(K(0) + 0.5^2) = 0{,}250$$
$$D_{0,3} = 1 * (-1)(K(0) + 0.5^2) = 0{,}250$$
$$D_{0,4} = 1 * (-1)(K(0) + 0.5^2) = 0{,}250$$

$$D_{0,5} = 1 * (-1)(K(0) + 0.5^2) = 0{,}250$$

Nilai -1 pada $D_{0,5}$ merupakan label kelas pada data ke-5 (label sebenarnya 0). Seperti yang dijelaskan bahwasanya SVM memiliki anggota saat klasifikasi $[+1, -1]$.

Tabel 3.34 Hasil *Matrix Hessian*

Index	Data Train ke-0	Data Train ke-1	Data Train ke-2	Data Train ke-3	Data Train ke-4	Data Train ke-5
Data Train ke-0	1,250	0,250	0,250	0,250	0,250	0,250
Data Train ke-1	0,250	1,250	0,250	0,250	0,250	0,249
Data Train ke-2	0,250	0,250	1,250	0,250	0,250	0,250
Data Train ke-3	0,250	0,250	0,250	1,250	0,251	0,250
Data Train ke-4	0,250	0,250	0,250	0,251	1,250	0,250
Data Train ke-5	0,250	0,249	0,250	0,250	0,250	1,250

4. Nilai *Error*

   Berdasarkan persamaan rumus (2.13), berikut contoh perhitungan Nilai Error pada data train ke-0 Kelas 1 yang terdapat pada Tabel 3.35 :

$$E_i = \sum_{j=1}^{n} \alpha_i \, D_{i,j}$$

$$E0 = \sum_{j}^{i} = \big((1{,}250 * 1{,}250) + (0{,}250 * 0{,}250) + \cdots + (0{,}250 * (0{,}250))\big) * 0{,}5 = 1{,}250$$

Tabel 3.35 Hasil nilai *error*

Nilai Error	
E0	1,250
E1	1,250
E2	1,250
E3	1,250
E4	1,250
E5	1,250

5. Delta Alpha

Berdasarkan persamaan rumus (2.14) tersebut, berikut contoh perhitungan Delta Alpha pada data train ke-0 Kelas 1 yang terdapat pada Tabel 3.36 :

$$\delta\alpha_i = \min\{\max[\gamma(1 - Ei) - \alpha_i]\, C - \alpha_i$$

$$\delta\alpha_0 = \min\{\max[0.5(1 - 1{,}250) - 0{,}5]\, 2 - 0{,}05 = 0{,}500$$

Tabel 3.36 Hasil *Delta Alpha* pada setiap kelas

Delta Alpha Kelas 1		Delta Alpha Kelas -1		Delta Alpha Kelas 0	
a'0	0,500	**a'0**	0,500	**a'0**	0,500
a'1	0,500	**a'1**	0,500	**a'1**	0,500
a'2	0,500	**a'2**	0,500	**a'2**	0,500
a'3	0,500	**a'3**	0,500	**a'3**	0,500
a'4	0,500	**a'4**	0,500	**a'4**	0,500
a'5	0,500	**a'5**	0,500	**a'5**	0,500

Berdasarkan persamaan rumus (2.15), berikut adalah contoh perhitungan Alpha Baru pada data train ke-0 Kelas 1 yang terdapat pada Tabel 3.37. :

$$a_i = a_i + \delta a_i$$

$$a_0 = 0{,}500 + \ 0{,}5 = 1$$

Tabel 3.37 Hasil Alpha Baru Kelas 1

Alpha Baru	
a0	1
a1	1
a2	1
a3	1
a4	1
a5	1

6. Bias

Untuk klasifikasi positif, negatif, dan netral, pendekatan multi-kelas SVM yang menangani klasifikasi antara dua kelas diperlukan. *One Againts All* (OAA) adalah salah satu dari dua metode yang dapat digunakan untuk menerapkan metode multi-kelas SVM dengan menggabungkan beberapa SVM biner (Fitriana 2020). Berdasarkan hal tersebut maka dapat digambarkan seperti pada Tabel 3.38.

Tabel 3.38 Label Kelas Terkait dan Bukan Kelas Terkait pada Bias

Label	W.X+ (Kelas Terkait)	W.X- (Bukan Kelas Terkait)
1	D0	D3
-1	D3	D4
0	D4	D0

Setelah dilakukan proses Bias, maka didapatkan hasil pada Tabel 3.39.

Tabel 3.39 Hasil Bias

Label	Nilai Bias
1	0
-1	0
0	0

7. Klasifikasi

Proses Klasifikasi 3 kelas menggunakan persamaan rumus (2.16) sebagai berikut(Fitriana 2020). Berdasarkan persamaan rumus tersebut, berikut contoh perhitungan klasifikasi pada data test ke-0 yang terdapat pada Tabel 3.39 :

$$h(x) = Argmax(\sum_{t=1}^{n} ai.yi.K(xi,x) + b_{label\,1}; \sum_{t=1}^{n} ai.yi.K(xi,x) + b_{label\,2}; \sum_{t=1}^{n} ai.yi.K(xi,x) + b_{label\,3})$$

$$h(0) = Argmax(\sum_{t=1}^{n} \big((0{,}000 * -1 * 0{,}000) + \cdots + (0{,}000 * -1 * 0{,}000)\big) + (0); \sum_{t=1}^{n} \big((0{,}000 * -1 * 0{,}000) + \cdots + (0{,}000 * 1 * 0{,}000)\big) + (0); \sum_{t=1}^{n} \big((0{,}000 * 1 * 0{,}000) + \cdots + (000 * -1 * 0{,}000)\big) + (0)$$
$$= Argmax((0{,}136); (-0{,}136); (-0{,}139)) = 0{,}136$$

Tabel 3.40 Hasil Prediksi pada *Data Tes*

Prediksi (Data Testing)						
Data Testing	**-1**	**0**	**1**	**Max Prediksi**	**Prediksi Label**	**Kelas Sebenarnya**
0	0,136	-0,136	-0,139	0,136	-1	1
1	0,137	-0,137	-0,172	0,137	-1	-1
2	-0,063	-0,129	0,048	0,048	1	0

3.7.2 Klasifikasi SVM Menggunakan *TF-IDF*

1. Inisialisasi Parameter

   Langkah awal yang dilakukan untuk melakukan Klasifikasi SVM adalah inisialisasi parameter diantaranya, Menginisiasi nilai awal untuk nilai α, C, epsilon gamma dan lamda (Rabbani 2023). Tabel 3.41 adalah hasil dari inisialisasi nilai untuk setiap parameternya

Tabel 3. 41 Inisialisasi Parameter

Parameter	**Nilai**	**Keterangan**
A	0.5	>=0 (Purnamasari et al. 2023)
C	2	> 0 (Purnamasari et al. 2023)
Λ	0.5	> 0 (Averina et al. 2020)

Gamma	2	> 0 (Saadah et al. 2021)

2. Menghitung Kernel RBF

Fungsi kernel non-linier sering digunakan untuk mengubah data masukan ke ruang fitur berdimensi tinggi di mana data input menjadi lebih dapat dipisahkan dari ruang input asli (Karyawati 2023). Berdasarkan rumus (2.11), berikut perhitungan kernel RBF pada data train ke-0 yang terdapat pada Tabel 3.42:

$$K(x_i, x_j) = \exp(-\gamma |x_i - x_j|^2)$$

$$K(x_0, x_0) = \exp(2 * |(0{,}665 - 0{,}665) + (0{,}665 - 0{,}665) + \cdots + ((0 - 0|^2)$$

$$K(x_0, x_0) = 1$$

$$K(x_0, x_1) = \exp(2 * |(0{,}665 - 0) + (0{,}665 - 0) + \cdots + ((0 - 0|^2)$$

$$K(x_0, x_1) = 0.018$$

$$K(x_0, x_2) = \exp(2 * |(0{,}665 - 0) + (0{,}665 - 0) + \cdots + ((0 - 0|^2)$$

$$K(x_0, x_2) = 0.029$$

$$K(x_0, x_3) = \exp(2 * |(0{,}665 - 0) + (0{,}665 - 0) + \cdots + ((0 - 0|^2)$$

$$K(x_0, x_3) = 0.034$$

$$K(x_0, x_4) = \exp(2 * |(0{,}665 - 0) + (0{,}665 - 0) + \cdots + ((0 - 0|^2)$$

$$K(x_0, x_4) = 0.034$$

$$K(x_0, x_5) = \exp(2 * |(0{,}665 - 0) + (0{,}665 - 0) + \cdots + ((0 - 0|^2)$$

$$K(x_0, x_5) = 0.029$$

Tabel 3. 42 Hasil perhitungan kernel RBF

Index	Data Train ke-0	Data Train ke-1	Data Train ke-2	Data Train ke-3	Data Train ke-4	Data Train ke-5
Data Train ke-0	1	0,018	0,029	0,034	0,034	0,029
Data Train ke-1	0,018	1	0,018	0,018	0,018	0,018
Data Train ke-2	0,029	0,018	1	0,034	0,034	0,029
Data Train ke-3	0,034	0,018	0,034	1	0,042	0,034
Data Train ke-4	0,034	0,018	0,034	0,042	1	0,034
Data Train ke-5	0,029	0,018	0,029	0,034	0,034	1

3. Matrix Hessian

Berdasarkan rumus (2.12), berikut contoh perhitungan dalam menghitung Matrix Hessian pada data train ke-0 yang terdapat pada Tabel 3.43:

$$D_{ij} = y_i y_j (K(x_i . x_j) + \lambda^2)$$

$$D_{0,0} = 1 * -1(K(1) + 0.5^2) = 1.250$$

$$D_{0,1} = 1 * -1(K(0.018) + 0.5^2) = 0.268$$

$$D_{0,2} = -1 * -1(K(0.029) + 0.5^2) = 0.221$$

$$D_{0,3} = -1 * -1(K(0.034) + 0.5^2) = 0.216$$

$$D_{0,4} = -1 * -1(K(0.034) + 0.5^2) = 0.216$$

$$D_{0,5} = -1 * -1(K(0.029) + 0.5^2) = 0.221$$

Nilai -1 merupakan label kelas pada data ke-5 (label sebenarnya 0). Seperti yang dijelaskan bahwasanya SVM memiliki anggota saat klasifikasi $[+1, -1]$.

Tabel 3. 43 Hasil Matrix Hessian

Index	0	1	2	3	4	5
0	1,250	0,268	0,221	0,216	0,216	0,221
1	0,268	1,250	0,232	0,232	0,232	0,232
2	0,221	0,232	1,250	0,284	0,284	0,279
3	0,216	0,232	0,284	1,250	0,292	0,284
4	0,216	0,232	0,284	0,292	1,250	0,284
5	0,221	0,232	0,279	0,284	0,284	1,250

4. Nilai Error

   Berdasarkan rumus (2.13), berikut contoh perhitungan Nilai Error pada data train ke-0 Kelas 1 yang terdapat pada Tabel 3.44 :

$$E_i = \sum_{j=1}^{n} \alpha_i \, D_{i,j}$$

$$E0 = \sum_{j}^{i} = \big((0.5 * 1.250) + (0.5 * 0.268) + \cdots + (0.5 * 0.221)\big)$$

$$= 1.296$$

Tabel 3.44 Hasil nilai error

Nilai Error	
E0	1,196
E1	1,223
E2	1,275
E3	1,279
E4	1,279
E5	1,275

5. Delta Alpha

   Berdasarkan rumus (2.14), berikut contoh perhitungan Delta Alpha pada data train ke-0 pada data train ke-0 Kelas 1 yang terdapat pada Tabel 3.45 :

$$\delta\alpha_i = \min\{\max[\gamma(1 - Ei) - \alpha_i]\, C - \alpha_i$$

$$\delta\alpha_0 = \min\{\max[2 * (1 - 1.196); -0.5]\,; 2 - 0.5\} = -0.392$$

Tabel 3.45 Hasil Delta Alpha

Delta Alpha	
a'0	-0,392
a'1	-0,445
a'2	-0,500
a'3	-0,500
a'4	-0,500
a'5	-0,500

Berdasarkan rumus (2.15), berikut contoh perhitungan Alpha Baru pada data train ke-0 Kelas 1 yang terdapat pada Tabel 3.46 :

$$a_i = a_i + \delta a_i$$

$$a_0 = 0.5 + (-0.392) = 0.108$$

Tabel 3. 46 Hasil Alpha Baru Kelas 1

Alpha Baru	
a0	0,108
a1	0,055
a2	0
a3	0
a4	0
a5	0

6. Bias

Untuk klasifikasi positif, negatif, dan netral, pendekatan multi-kelas SVM yang menangani klasifikasi antara dua kelas diperlukan. *One Againts All* (OAA) adalah salah satu dari dua metode yang dapat digunakan untuk menerapkan metode multi-kelas SVM dengan menggabungkan beberapa SVM biner (Fitriana 2020). Berdasarkan hal tersebut maka dapat digambarkan seperti pada Tabel 3.47:

Tabel 3. 47 Label Kelas Terkait dan Bukan Kelas Terkait pada Bias

Label	W.X+ (Kelas Terkait)	W.X- (Bukan Kelas Terkait)
1	D0	D3
-1	D3	D4
0	D4	D0

Setelah dilakukan proses Bias, maka didapatkan hasil pada Tabel 3.48.

Tabel 3. 48 Hasil Bias

Bias Kelas 1	-0,057
Bias Kelas -1	-0,052
Bias Kelas 0	-0,051

7. Klasifikasi

Proses Klasifikasi 3 kelas menggunakan persamaan rumus (2.16). Berikut contoh perhitungan pada data test ke-0 yang terdapat pada Tabel 3.49 :

$$h(x) = Argmax(\sum_{t=1}^{n} ai.yi.K(xi,x) + b_{label\,1}; \sum_{t=1}^{n} ai.yi.K(xi,x) + b_{label\,2}; \sum_{t=1}^{n} ai.yi.K(xi,x) + b_{label\,3})$$

$$h(0) = Argmax(\sum_{t=1}^{n} \big((0*1*0) + \cdots + (0*-1*0)\big) + (-0.057); \sum_{t=1}^{n} \big(0*1*0) + \cdots + (0*-1*0)\big) + (-0.052); \sum_{t=1}^{n} \Big(0*1*0) + \cdots + \big((0*-1*0)\big)\Big) + (-0.051)$$

$$= Argmax((0);(0);(0)) = 0$$

Tabel 3. 49 Hasil Prediksi pada Data Tes

Prediksi (Data Testing)						
Data Testing	**-1**	**0**	**1**	**Max Prediksi**	**Prediksi Label**	**Kelas Sebenarnya**
0	0,000	0,000	0,000	0,000	0	1

1	0,000	0,000	0,000	0,000	0	-1
2	0,000	0,000	0,000	0,000	0	0

3.8 Evaluasi Confusion Matrix

3.8.1 Evaluasi Menggunakan *FastText*

Untuk menguji performa model prediksi dapat dilakukan evaluasi menggunakan confusion matrix pada FastText.

1. *Accuracy*

   Total persentase prediksi yang dinyatakan benar dengan keseluruhan data. Berdasarkan persamaan rumus (2.17) tersebut, berikut adalah perhitungan *Accuracy* :

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Accuracy = \frac{(1 + 1 + 1)}{(0 + 0 + 0 + \cdots + 1)} = 0.33$$

2. *Precision*

   Total persentase prediksi yang memang benar bernilai positif dengan total jumlah data yang terprediksi positif. Berdasarkan persamaan rumus (2.18) tersebut, berikut adalah perhitungan *Precision* :

$$Precision = \frac{TP}{TP + FP}$$

$$Precision = \frac{\frac{3}{(0 + 0 + 0 + .. + 1)}}{3} = 0.17$$

3. *Recall*

   Total persentase prediksi yang memang benar bernilai positif dengan total jumlah data yang memang benar positif. Berdasarkan persamaan rumus (2.19), berikut adalah perhitungan *Recall* :

$$Recall = \frac{TP}{TP + FN}$$

$$Recall = \frac{\frac{3}{(0 + 0 + 0 + .. + 1)}}{3} = 0.33$$

4. F1-*Score*

   Perbandingan rata-rata *precision* dan *recall*. Berdasarkan persamaan rumus (2.20) tersebut, berikut adalah perhitungan F1-Score :

$$F1 - Score = 2 \times \frac{Recall \times Precision}{Recall + Precison}$$

$$F1 - Score = \frac{\left(2\frac{0\% * 0\%}{0\% + 0\%}\right) + \left(2\frac{0\% * 0\%}{0\% + 0\%}\right) + \left(2\frac{100\% * 100\%}{100\% + 100\%}\right)}{3}$$

$$= 0.22$$

Setelah dilakukan proses Evaluasi menggunakan *Confusion Matrix*, maka didapatkan hasil pada Tabel 3.52.

Tabel 3.50 Hasil Evaluasi FastText

Akurasi	33%
Presisi	17%
Recall	33%
F1-Score	22%

3.8.2 Evaluasi Menggunakan *TF-IDF*

Untuk menguji performa model, maka dapat dilakukan evaluasi salah satunya menggunakan *confusion matrix* pada TF-IDF.

1. *Accuracy*

   Total persentase prediksi yang dinyatakan benar dengan keseluruhan data. Berdasarkan persamaan rumus (2.17) tersebut, berikut adalah perhitungan *Accuracy* :

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Accuracy = \frac{(1 + 0 + 0)}{(1 + 0 + 0 + \cdots + 0)} = 0.33$$

2. *Precision*

   Total persentase prediksi yang memang benar bernilai positif dengan total jumlah data yang terprediksi positif. Berdasarkan persamaan rumus (2.18) tersebut, berikut adalah perhitungan *Precision* :

$$Precision = \frac{TP}{TP + FP}$$

$$\text{Precision} = \frac{\frac{1}{(1+1+1+..+0)}}{3} = 0.11$$

3. *Recall*

Total persentase prediksi yang memang benar bernilai positif dengan total jumlah data yang memang benar positif. Berdasarkan persamaan rumus (2.19), berikut adalah perhitungan *Recall* :

$$Recall = \frac{TP}{TP + FN}$$

$$\text{Recall} = \frac{\frac{1}{(1+0+0+..+0)}}{3} = 0.33$$

4. F1-*Score*

Perbandingan rata-rata *precision* dan *recall*. Berdasarkan persamaan rumus (2.20) tersebut, berikut adalah perhitungan F1-Score :

$$F1 - Score = 2 \times \frac{Recall \times Precision}{Recall + Precison}$$

$$F1 - Score = \frac{\left(2\frac{0\% * 100\%}{0\% + 100\%}\right) + \left(2\frac{0\% * 0\%}{0\% + 0\%}\right) + \left(2\frac{0\% * 0\%}{0\% + 0\%}\right)}{3}$$

$$= 0.17$$

Setelah dilakukan proses Evaluasi menggunakan *Confusion Matrix* menggunakan TF-IDF, maka didapatkan hasil pada Tabel 3.53.

Tabel 3. 51 Hasil Evaluasi TF-IDF

Accuracy	33%
Presisi	11%
Recall	33%
F1-Score	17%

BAB 4

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Sistem

Pada penelitian ini terdapat sarana pendukung yang digunakan dalam pengimplementasian sentimen analisis pengguna IndiHome di X (*Twitter)* dengan Metode FastText dan Support Vector Machine.

4.1.1 Spesifikasi Perangkat Lunak *(Software)*

Perangkat lunak yang digunakan dalam menerapkan metode *FastText* dan SVM untuk sentimen analisis IndiHome sebagai berikut :

1. *Web Browser*: *Google Chrome*
2. Diagram Alir Penelitian: *Draw.io*
3. Bahasa Pemrograman: *Python*
4. Teks Editor: *Google Collab*
5. Pembuatan laporan: *Microsoft Word* 2019
6. *Microsoft Excel* 2019
7. Sistem Operasi: *Windows 10*

4.1.2 Spesifikasi Perangkat Keras *(Hardware)*

Perangkat keras (*hardware*) yang digunakan dalam menerapkan metode *FastText* dan SVM untuk sentimen analisis IndiHome sebagai berikut:

1. Laptop dengan *processor Intel i7*
2. *Memory/RAM* *:* 8,00 GB
3. *System Type* *: 64-bit operating system, x64-based processor*

4.2 Input Dataset

Tahap pertama dilakukan input dataset dari hasil scraping yang dilakukan dari sosial media X (Twitter*)*. Input dataset dilakukan di *Google Collab* dengan menggunakan bahasa pemrograman *Python.* Berikut merupakan tampilan *input dataset* pada Gambar 4.1.

	Unnamed: 0	Username	date	tweet
0	0	Fikel go	28-Feb-23	Harimu Hampa? IndiHome Aja!\n\nDijamin hari ka...
1	1	IndiHomeCare	28-Feb-23	Halo Kak Yani, apakah ada yang bisa Yurai bant...
2	2	maung congkok	28-Feb-23	@IndiHome\n ppppp
3	3	IndiHomeCare	28-Feb-23	Waalaikumsalam Kak Joma, apakah ada yang bisa ...
4	4	IndiHomeCare	28-Feb-23	Halo Kak Fitri, apakah ada yang bisa Yurai ban...
...	...	...	...	...
1532	1532	aw	26-Feb	mba indihome
1533	1533	Pahla	26-Feb	@IndiHomeCare\n @IndiHome\n pun10, gangguan
1534	1534	ipan	26-Feb	Maokla kak. Pake indihome ni bede 100an a
1535	1535	abaz.blesax	26-Feb	@IndiHomeCare\n hallo Indihome wifi rumah saya...
1536	1536	IndiHome	26-Feb	Ternyata kelakuan aneh bin ajaib keluarga Tamu...

1537 rows × 4 columns

Gambar 4.1 *Input Dataset*

Berdasarkan Gambar 4.1 tersebut, terdapat 1537 dataset yang didapat dari scraping dengan yang dibangun menggunakan Selenium dan terdiri dari 4 kolom yaitu kolom untuk *index, username, date,* dan *tweet.*

4.3 Pre-processing Data

Selanjutnya adalah tahap *pre-processing,* di mana dilakukan beberapa proses seperti *cleaning, casefolding, tokenizing, normalization, filtering, stemming,* sehingga dihasilkan *output pre-processing* dengan data sebanyak 1514 data seperti pada Gambar 4.2 berikut ini.

	username	tweet	cleaning	casefolding	normalization	tokenizing	filtering	stemming
0	Fikel go	Harimu Hampa? IndiHome Aja!\n\nDijamin hari ka...	Harimu Hampa IndiHome Aja\n\nDijamin hari kamu...	harimu hampa indihome aja\n\ndijamin hari kamu...	harimu hampa indihome saja dijamin hari kamu t...	['harimu', 'hampa', 'indihome', 'saja', 'dijam...	harimu hampa indihome dijamin hari kamu tidak ...	hari hampa indihome jamin hari kamu tidak hamp...
1	IndiHomeCare	Halo Kak Yani, apakah ada yang bisa Yurai bant...	Halo Kak Yani apakah ada yang bisa Yurai bantu...	halo kak yani apakah ada yang bisa yurai bantu...	halo kak yani apakah ada yang bisa yurai bantu...	['halo', 'kak', 'yani', 'apakah', 'ada', 'yang...	halo yani apakah bisa yurai bantu mengenai pro...	halo yani apakah bisa yura bantu kena produk l...
3	IndiHomeCare	Waalaikumsalam Kak Joma, apakah ada yang bisa ...	Waalaikumsalam Kak Joma apakah ada yang bisa Y...	waalaikumsalam kak joma apakah ada yang bisa y...	waalaikumsalam kak joma apakah ada yang bisa y...	['waalaikumsalam', 'kak', 'joma', 'apakah', 'a...	joma apakah bisa yurai bantu mengenai produk l...	joma apakah bisa yura bantu kena produk layan ...
4	IndiHomeCare	Halo Kak Fitri, apakah ada yang bisa Yurai ban...	Halo Kak Fitri apakah ada yang bisa Yurai bant...	halo kak fitri apakah ada yang bisa yurai bant...	halo kak fitri apakah ada yang bisa yurai bant...	['halo', 'kak', 'fitri', 'apakah', 'ada', 'yan...	halo fitri apakah bisa yurai bantu mengenai pr...	halo fitri apakah bisa yura bantu kena produk ...
5	Sky you	Semua mudah dengan IndiHome #IndiHome #Aktifit...	Semua mudah dengan IndiHome	semua mudah dengan indihome	semua mudah dengan indihome	['semua', 'mudah', 'dengan', 'indihome']	semua mudah indihome	semua mudah indihome
...	...	...	...	...	...	...	...	...
1510	aw	mba indihome	mba indihome	mba indihome	mba indihome	['mba', 'indihome']	mba indihome	mba indihome
1511	Pahla	@IndiHomeCare\n @IndiHome\n pun10, gangguan	\n \n pun gangguan	\n \n pun gangguan	pun gangguan	['pun', 'gangguan']	gangguan	ganggu
1512	ipan	Maokla kak. Pake indihome ni bede 100an a	Maokla kak Pake indihome ni bede an a	maokla kak pake indihome ni bede an a	maokla kak pakai indihome ini bede an a	['maokla', 'kak', 'pakai', 'indihome', 'ini', ...	maokla pakai indihome bede	maokla pakai indihome bede
1513	abaz.blesax	@IndiHomeCare\n hallo Indihome wifi rumah saya...	\n hallo Indihome wifi rumah saya dari tagal ...	\n hallo indihome wifi rumah saya dari tagal ...	hallo indihome wifi rumah saya dari tagal suda...	['hallo', 'indihome', 'wifi', 'rumah', 'saya',...	hallo indihome wifi rumah tagal sudah lporan g...	hallo indihome wifi rumah tagal sudah lporan g...
1514	IndiHome	Ternyata kelakuan aneh bin ajaib keluarga Tamu...	Ternyata kelakuan aneh bin ajaib keluarga Tamu...	ternyata kelakuan aneh bin ajaib keluarga tamu...	ternyata kelakuan aneh bin ajaib keluarga tamu...	['ternyata', 'kelakuan', 'aneh', 'bin', 'ajaib...	ternyata kelakuan aneh bin ajaib keluarga tamu...	nyata laku aneh bin ajaib keluarga tamu harap ...

1514 rows × 10 columns

Gambar 4.2 *Pre-processing Data*

Cleaning merupakan langkah yang bertujuan untuk menghapus karakter. *Case folding* merupakan teknik mengubah semua huruf kapital menjadi huruf kecil dengan memanfaatkan *library re* dengan fungsi *re.sub*(). *Tokenizing* merupakan proses pemecahan kata menjadi beberapa bagian. Hasil kata yang telah dipecah disebut dengan token. *Tokenizing* dilakukan dengan memanfaatkan *library nltk.tokenize. Filtering* yaitu dilakukan proses penghapusan kata yang tidak dibutuhkan berupa kata keterangan dan kata sambung dari hasil *token.* Tahap *stemming* merupakan proses mengubah kata dengan mengembalikan bentuknya ke dalam bentuk aslinya atau kata dasar.

4.4 Pelabelan Data

Selanjutnya setelah dilakukan *pre-processing data,* akan dilakukan pelabelan. Pelabelan pada penelitian ini dilakukan menggunakan *TextBlob.* Pada *labelling* menggunakan nilai *polarity*. Gambar 4.3 merupakan hasil dari pelabelan data.

	username	tweet	cleaning	casefolding	normalization	tokenizing	filtering	stemming	score	label
0	Fikel go	Harimu Hampa? IndiHome Aja!\n\nDijamin hari ka...	Harimu Hampa IndiHome Aja\n\nDijamin hari kamu...	harimu hampa indihome aja\n\ndijamin hari kamu...	harimu hampa indihome saja dijamin hari kamu t...	['harimu', 'hampa', 'indihome', 'saja', 'dijam...	harimu hampa indihome dijamin hari kamu tidak ...	hari hampa indihome jamin hari kamu tidak hamp...	0.000000	Netral
1	IndiHomeCare	Halo Kak Yani, apakah ada yang bisa Yurai bant...	Halo Kak Yani apakah ada yang bisa Yurai bantu...	halo kak yani apakah ada yang bisa yurai bantu...	halo kak yani apakah ada yang bisa yurai bantu...	['halo', 'kak', 'yani', 'apakah', 'ada', 'yang...	halo yani apakah bisa yurai bantu mengenai pro...	halo yani apakah bisa yura bantu kena produk l...	0.000000	Netral
3	IndiHomeCare	Waalaikumsalam Kak Joma, apakah ada yang bisa ...	Waalaikumsalam Kak Joma apakah ada yang bisa Y...	waalaikumsalam kak joma apakah ada yang bisa y...	waalaikumsalam kak joma apakah ada yang bisa y...	['waalaikumsalam', 'kak', 'joma', 'apakah', 'a...	joma apakah bisa yurai bantu mengenai produk l...	joma apakah bisa yura bantu kena produk layan ...	0.000000	Netral
4	IndiHomeCare	Halo Kak Fitri, apakah ada yang bisa Yurai ban...	Halo Kak Fitri apakah ada yang bisa Yurai bant...	halo kak fitri apakah ada yang bisa yurai bant...	halo kak fitri apakah ada yang bisa yurai bant...	['halo', 'kak', 'fitri', 'apakah', 'ada', 'yan...	halo fitri apakah bisa yurai bantu mengenai pr...	halo fitri apakah bisa yura bantu kena produk ...	0.000000	Netral
5	Sky you	Semua mudah dengan IndiHome #IndiHome #Aktifit...	Semua mudah dengan IndiHome	semua mudah dengan indihome	semua mudah dengan indihome	['semua', 'mudah', 'dengan', 'indihome']	semua mudah indihome	semua mudah indihome	0.433333	Positif
...	...	...	...	...	...	...	...	...	...	...
1510	aw	mba indihome	mba indihome	mba indihome	mba indihome	['mba', 'indihome']	mba indihome	mba indihome	0.000000	Netral
1511	Pahla	@IndiHomeCare\n @IndiHome\n pun10, gangguan	\n \n pun gangguan	\n \n pun gangguan	pun gangguan	['pun', 'gangguan']	gangguan	ganggu	0.000000	Netral
1512	ipan	Maokla kak. Pake indihome ni bede 100an a	Maokla kak Pake indihome ni bede an a	maokla kak pake indihome ni bede an a	maokla kak pakai indihome ini bede an a	['maokla', 'kak', 'pakai', 'indihome', 'ini', ...	maokla pakai indihome bede	maokla pakai indihome bede	0.000000	Netral
1513	abaz.blesax	@IndiHomeCare\n hallo Indihome wifi rumah saya...	\n hallo Indihome wifi rumah saya dari tagal ...	\n hallo indihome wifi rumah saya dari tagal ...	hallo indihome wifi rumah saya dari tagal suda...	['hallo', 'indihome', 'wifi', 'rumah', 'saya',...	hallo indihome wifi rumah tagal sudah lporan g...	hallo indihome wifi rumah tagal sudah lporan g...	0.000000	Netral
1514	IndiHome	Ternyata kelakuan aneh bin ajaib keluarga Tamu...	Ternyata kelakuan aneh bin ajaib keluarga Tamu...	ternyata kelakuan aneh bin ajaib keluarga tamu...	ternyata kelakuan aneh bin ajaib keluarga tamu...	['ternyata', 'kelakuan', 'aneh', 'bin', 'ajaib...	ternyata kelakuan aneh bin ajaib keluarga tamu...	nyata laku aneh bin ajaib keluarga tamu harap ...	0.100000	Positif

1514 rows × 10 columns

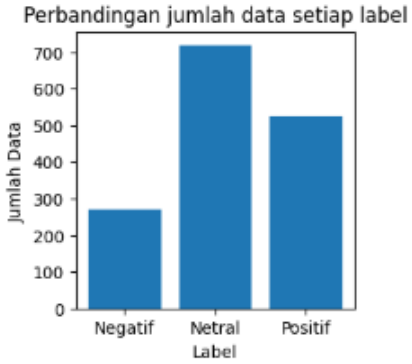
Gambar 4.3 Pelabelan Data

Berdasarkan hasil pelabelan pada Gambar 4.3, total label yang didapatkan tertera pada Gambar 4.4 berikut:

```
Total Label :
label
Netral     719
Positif    524
Negatif    271
Name: count, dtype: int64
```

Gambar 4.4 Total Label

Dari Gambar 4.4 tersebut didapatkan hasil pelabelan negatif sebanyak 271, netral sebanyak 719, dan positif sebanyak 524, dengan total 1514 data. Selanjutnya, dilakukan pengelompokan untuk melihat perbandingan jumlah data pada setiap label. Gambar 4.5 merupakan perbandingan jumlah data setiap labelnya.



Gambar 4. 5 Perbandingan Jumlah Data Setiap Label

4.5 Split Data

Setelah melakukan pelabelan, dilanjutkan *split data* dengan membagi data menjadi data train dan data test. Data akan terbagi menjadi 80% data train dan 20% data test. Gambar 4.8 merupakan tampilan data split data dari data train.

	stem	label
0	lag indihome komtol	Netral
1	vaii minta bisa kami bantu bisa jelas ingin mi...	Positif
2	bisa fathur minta pindah alamat bisa bantu lew...	Netral
3	wow mbps mulai ribu	Positif
4	padahal enak yak untuk bestee aku biar makin u...	Positif
...	...	...
1206	halo admin mau ganti paket wifi bagaimana cara...	Positif
1207	thankyou sudah sahabat jaring bukti banyak fla...	Positif
1208	kenapaa indihome adhan	Netral
1209	butuh netflix bulan nomor renewal bisa pakai i...	Netral
1210	indihome sudah hari loh	Netral

1211 rows × 2 columns

Gambar 4. 6 Data Train

Berdasarkan Gambar 4.6 dari data train, total label negatif sebanyak 220, label netral sebanyak 592, sedangkan label positif sebanyak 399, dengan total 1211 data training. Berikut pada Gambar 4.7 merupakan tampilan dari *split data* dari data test.

	stem	label
0	guna kece lebih lanjut kenan informasi nomor i...	Positif
1	sudah aku	Netral
2	contoh promo bisa kalian dapat misal tagih ind...	Positif
3	admin sudah login myindihome tapi tiap mau con...	Netral
4	halo bensutoyo seperti sudah informasi belum u...	Positif
...	...	...
298	indihome jelek malem padahal pakai doang laen ...	Negatif
299	pakai indihome	Netral
300	indihome temenku error boleh bantuuu	Netral
301	indihome kenapa tidak jelas	Negatif
302	benar namun hanya perangkat ambil kabel jadi h...	Positif

303 rows × 2 columns

Gambar 4. 7 Data Test

Berdasarkan Gambar 4.7 dari data test, total label negatif sebanyak 51, label netral sebanyak 127, sedangkan pada label positif terdapat 125, dengan total 303 data test.

4.6 Word Embedding

Word embedding merupakan proses mengubah setiap kata menjadi vektor yang mewakili maknanya sendiri. Berikut merupakan penjabaran mengenai proses tanpa menggunakan FastText dan menggunakan FastText.

4.6.1 Tanpa Menggunakan *FastText*

Proses *word embedding* tanpa menggunakan FastText dilakukan dengan menggunakan TF-IDF. Dalam prosesnya terdapat transformasi teks menjadi representasi numerik menggunakan dua pendekatan yaitu TF-IDF dan *Count Vectorizer.*

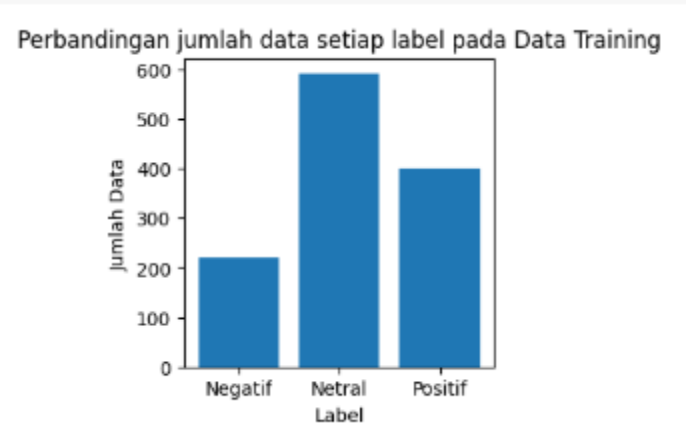
Pada bagian pertama, TF-IDF Vectorizer digunakan untuk mengubah teks ke dalam representasi vector berdasarkan bobot TF-IDF. Hal ini dilakukan dengan menggunakan *TfidfVectorizer(norm='l2')*. Kemudian data train dan data test diubah menjadi representasi vector menggunakan *fit_transform()* dan *transform()*. Selanjutnya terdapat *CountVectorizer()* digunakan untuk menghitung frekuensi kata-kata dalam teks. Dari hasil kedua transformasi tersebut disimpan ke dalam file CSV. Berikut Gambar 4.8 merupakan hasil representasi TF-IDF dari data train beserta labelnya.

	abangda	abdul	absen	abu	accs	action	ada	adalah	addon	adekku	...	yuujin	zayn	zhejiang	zikri	zoey	zonk	zoom	zte	zum	label
D1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Netral
D2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Positif
D3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Netral
D4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Positif
D5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Positif
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
D1207	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Positif
D1208	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Positif
D1209	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Netral
D1210	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Netral
D1211	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Netral

1211 rows × 2570 columns

Gambar 4. 8 Hasil Representasi TF-IDF Data Training

Setelah itu, dilakukan pengelompokan untuk melihat perbandingan jumlah data setiap label pada data training. Berikut pada Gambar 4.9 merupakan perbandingan jumlah data setiap label pada TF-IDF data training.



Gambar 4. 9 Perbandingan Jumlah Data Setiap Label pada Data Training TF-IDF

Dari Gambar 4.9 terlihat bahwa data tidak seimbang, maka perlu dilakukan penyeimbangan data menggunakan metode *Synthetic Minority Over-Sampling Technique* (SMOTE) (Alex et al. 2022). Proses SMOTE menggunakan library *imblearn.over_sampling*. Kemudian membuat objek SMOTE dan melakukan oversampling pada data latih. Dari pengubahan data latih tersebut, dilanjutkan dengan menyimpan file ke dalam CSV. Berikut pada Gambar 4.10 merupakan hasil oversampling pada data train

	abangda	abdul	absen	abu	accs	action	ada	adalah	addon	adekku	...	yuujin	zayn	zhejiang	zikri	zoey	zonk	zoom	zte	zum	label
D1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Netral
D2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Positif
D3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Netral
D4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Positif
D5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Positif
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
D1772	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Positif
D1773	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Positif
D1774	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Positif
D1775	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Positif
D1776	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Positif

1776 rows × 2570 columns

Gambar 4. 10 Hasil Oversampling pada Data Training TF-IDF

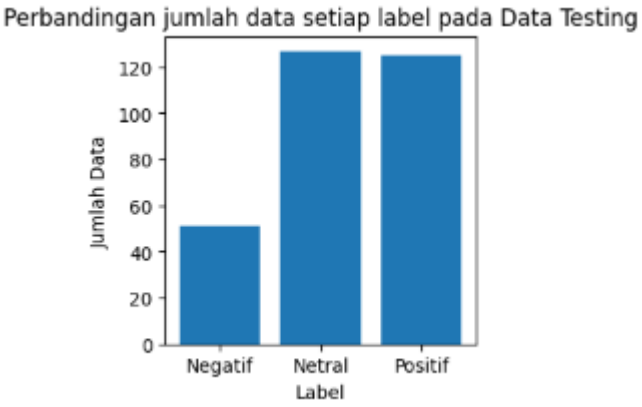
Setelah itu, dilakukan perhitungan total label pada data train dengan menghasilkan label negatif 592, label netral 592, dan label positif 592 dengan total 1776 label data training. Selanjutnya menampilkan hasil proses TF-IDF dari data testing. Berikut pada Gambar 4.14 merupakan hasil representasi TF-IDF dari data testing beserta labelnya.

	abangda	abdul	absen	abu	accs	action	ada	adalah	addon	adekku	...	yuujin	zayn	zhejiang	zikri	zoey	zonk	zoom	zte	zum	label
D1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	Positif
D2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	Netral
D3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	Positif
D4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	Netral
D5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.290949	0.0	0.0	0.0	0.0	0.0	Positif
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
D299	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	Negatif
D300	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	Netral
D301	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	Netral
D302	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	Negatif
D303	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	Positif

303 rows × 2570 columns

Gambar 4. 11 Hasil Representasi Data Testing TF-IDF

Setelah itu, dilakukan pengelompokan untuk melihat perbandingan jumlah data setiap label pada data testing dengan total 303 label pada data testing. Berikut pada Gambar 4.12 merupakan perbandingan jumlah data setiap label pada data testing.



Gambar 4. 12 Perbandingan Jumlah Data Setiap Label pada Data Testing TF-IDF

4.6.2 Menggunakan *FastText*

Dalam melakukan *word embedding* menggunakan FastText proses pertama yang dilakukan adalah mencari jumlah kata maksimum dengan variabel *max_len* untuk menyimpan jumlah kata maksimum. Pada setiap iterasi, teks akan dipecah menjadi kata-kata menggunakan fungsi split dan jika panjang daftar kata tersebut lebih besar dari nilai yang disimpan dalam *max_len*, maka nilai *max_len* akan diperbarui dengan panjang daftar kata yang baru. Dari proses tersebut menghasilkan nilai *max_len* yaitu 42.

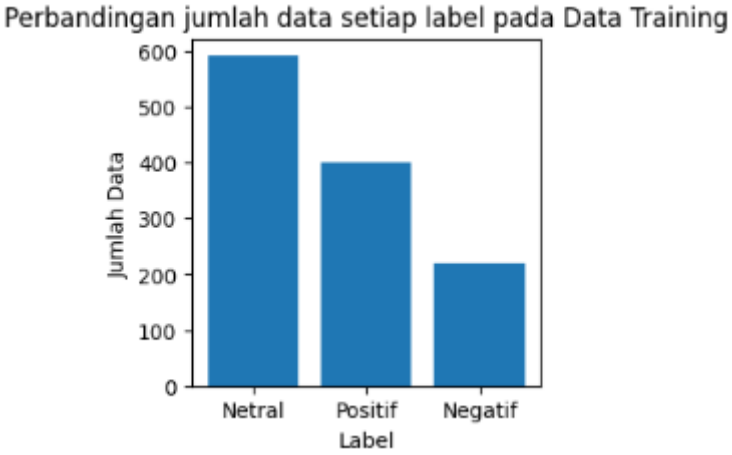
Selanjutnya untuk melatih model FastText menggunakan *genism.models* dan terdapat parameter pada proses tersebut seperti *sentences* yaitu data untuk pelatihan

model, *vector size* yang merupakan dimensi vector fitur yang akan dihasilkan oleh model. Berkutnya *window* dengan 3 sebagai jarak maksimum antara kata saat ini dan kata yang diprediksi dalam sebuah kalimat, *min_count* dengan nilai 3 (n-grams) sebagai frekuensi minimum kata yang harus ada agar dimasukkan ke dalam kosakata model, *sg* dengan nilai 1 untuk skip-gram model, *hs* dengan nilai 0 jika 1 *softmax* hierarkis akan digunakan untuk pelatihan model. Pada *hs* jika disetel ke 0 (default), dan negatif` bukan nol, pengambilan sampel negatif akan digunakan.

Lalu terdapat *negative* dengan nilai 5 (default), dimana jika lebih dari 0 maka pengambilan sampel negatif akan digunakan, parameter ini berfungsi untuk menentukan berapa banyak "noise word" yang harus dipakai (biasanya antara 5-20). Terakhir, terdapat *workers* dengan nilai 4 untuk pelatihan model dan *seed* dengan nilai 0 untuk penghasil angka acak.

Setelah itu dilakukan pembuatan vector untuk setiap kata dalam dokumen, lalu setelah menghasilkan vector dokumen, dan menyimpan vector dokumen dalam data train dan data test yang dihasilkan menggunakan fungsi *generate_document_vectors*. Lalu hasilnya adalah *array numpy* yang berisi vector dokumen untuk setiap dokumen dalam data train dan data test. Selanjutnya menghitung jumlah masing-masing label dalam data train yang berjumlah label negatif 220, label netral 592 dan label positif 399, dengan jumlah keseluruhan 1211 label.

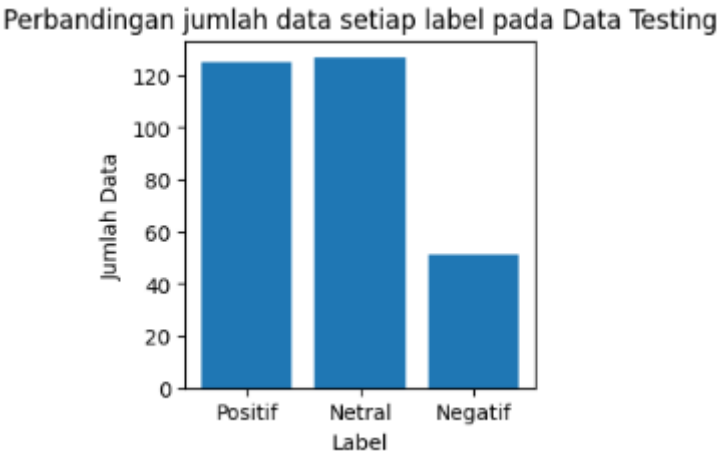
Setelah itu, dilakukan pengelompokan untuk melihat perbandingan jumlah data setiap label pada data training. Berikut pada Gambar 4.13 merupakan perbandingan jumlah data setiap label pada data training.



Gambar 4. 13 Perbandingan Jumlah Data Setiap Label pada Data Training FastText

Dari Gambar 4.13 terlihat bahwa data tidak seimbang, maka perlu dilakukan penyeimbangan data menggunakan metode *Synthetic Minority Over-Sampling Technique* (SMOTE) (Alex et al. 2022). Proses SMOTE menggunakan library *imblearn.over_sampling*. Kemudian membuat objek SMOTE dan melakukan oversampling pada data latih. Dari pengubahan data latih tersebut menghasilkan label negatif 592, label netral 592 dan label positif 592 dengan jumlah 1776 label.

Setelah dilakukan menampilkan hasil proses FastText dari data training, selanjutnya dilakukan menampilkan hasil proses FastText dari data testing yang menghasilkan label negatif sebanyak 51, label netral sebanyak 127 dan label positif sebanyak 125, dengan jumlah keseluruhan 303 label pada pada data testing. Setelah itu, dilakukan pengelompokan untuk melihat perbandingan jumlah data setiap label pada data testing. Berikut pada Gambar 4.14 merupakan perbandingan jumlah data setiap label pada data testing pada FastText.



Gambar 4. 14 Perbandingan Jumlah Data Setiap Label pada Data Testing FastText

4.7 Klasifikasi Teks Menggunakan SVM

Setelah dilakukan proses *word embedding* dilakukan proses klasifikasi teks menggunakan SVM, menggunakan FastText dan tanpa menggunakan FastText.

4.7.1 Tanpa Menggunakan *FastText*

Proses klasifikasi teks SVM Kernel RBF dengan embedding TF-IDF menggunakan parameter C=10, dan gamma=3. Kemudian dilakukan pelatihan menggunakan data yang telah di *resampled,* dan memprediksi kelas target

menggunakan model yang telah dilatih pada data train. Berikut pada Gambar 4.15 merupakan hasil dari proses tersebut.

```
Best parameter :  {'C': 10, 'gamma': 3}
```

Gambar 4. 15 Hasil Parameter Terbaik Tanpa Menggunakan FastText

4.7.2 Menggunakan *FastText*

Proses klasifikasi teks SVM Kernel RBF dengan embedding FastText, mengimplementasikan dari *Grid Search Cross Validation* untuk mencari kombinasi parameter terbaik. Gambar 4.16 merupakan hasil dari proses tersebut.

```
Best parameter :  {'C': 100, 'gamma': 5}
```

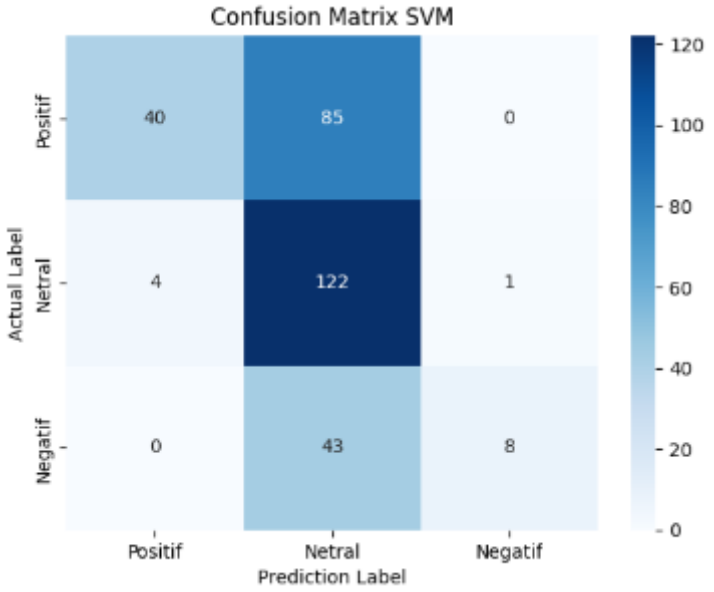
Gambar 4. 16 Hasil Grid Search Cross Validation menggunakan FastText

4.8 Evaluasi Confusion Matrix

Setelah dilakukan proses klasifikasi dilakukan proses evaluasi menggunakan *confusion matrix*, menggunakan FastText dan tanpa menggunakan FastText (TF-IDF). Berikut merupakan penjabarannya.

4.8.1 Tanpa Menggunakan *FastText*

Untuk proses evaluasi menggunakan *confusion matrix* menggunakan data test. Dalam proses ini terdapat label positif, netral, dan negatif. Berikut pada Gambar 4.17 merupakan tampilan *confusion matrix.*



Gambar 4. 17 Confusion Matrix Tanpa Menggunakan FastText

Berdasarkan Gambar 4.17 dapat dijelaskan bahwa algoritma SVM memperoleh hasil prediksi pada label positif terklasifikasikan dengan benar sebanyak 40 data, sedangkan kesalahan prediksi dengan 85 data yang masuk ke dalam label netral dan 0 data yang masuk ke dalam label negatif. Pada label netral dengan sebanyak 122 data terklasifikasikan dengan benar, sedangkan kesalahan prediksi dengan 4 data yang masuk ke dalam label positif dan 1 data yang masuk ke dalam label negatif. Pada label negatif dengan sebanyak 8 data terklasifikasikan dengan benar, sedangkan kesalahan prediksi dengan 0 data yang masuk ke dalam label positif, dan 43 data yang masuk ke dalam label netral. Selanjutnya dilakukan proses perhitungan *accuracy, precision, recall, f1-score.* Berikut pada Gambar 4.18 merupakan hasil dari *confusion matrix* tersebut.

```
Accuracy   : 56.11 %
Precission : 72.92 %
Recall     : 56.11 %
F-Score    : 51.14 %
```

Gambar 4. 18 Hasil Confusion Matrix Tanpa Menggunakan FastText

Berdasarkan Gambar 4.18 dapat dijelaskan nilai *accuracy* 56.77 %, *precision* 70.47 %, *recall* 56.77 %, dan *f1-score* 51.2%. Selanjutnya dilakukan visualisasi model klasifikasi SVM terhadap data test untuk mempermudah menganalisis performa model klasifikasi SVM dan memeriksa label yang diprediksi untuk data test. Berikut pada Gambar 4.19 merupakan hasil evaluasi prediksi untuk data uji.

	stem	label	pred_label_svm
0	guna kece lebih lanjut kenan informasi nomor i...	Positif	Positif
1	sudah aku	Netral	Netral
2	contoh promo bisa kalian dapat misal tagih ind...	Positif	Netral
3	admin sudah login myindihome tapi tiap mau con...	Netral	Netral
4	halo bensutoyo seperti sudah informasi belum u...	Positif	Positif
...	...	...	...
298	indihome jelek malem padahal pakai doang laen ...	Negatif	Netral
299	pakai indihome	Netral	Netral
300	indihome temenku error boleh bantuuu	Netral	Netral
301	indihome kenapa tidak jelas	Negatif	Netral
302	benar namun hanya perangkat ambil kabel jadi h...	Positif	Netral

303 rows × 3 columns

Gambar 4. 19 Hasil Evaluasi Prediksi Data Uji Tanpa Menggunakan FastText

Selanjutnya, penulis menambahkan *input text* untuk mengetahui sentimen dari teks tersebut dan melihat seberapa baik model yang sudah di train dan dapat dilihat pada Gambar 4.20.

Gambar 4. 20 Input Teks Baru

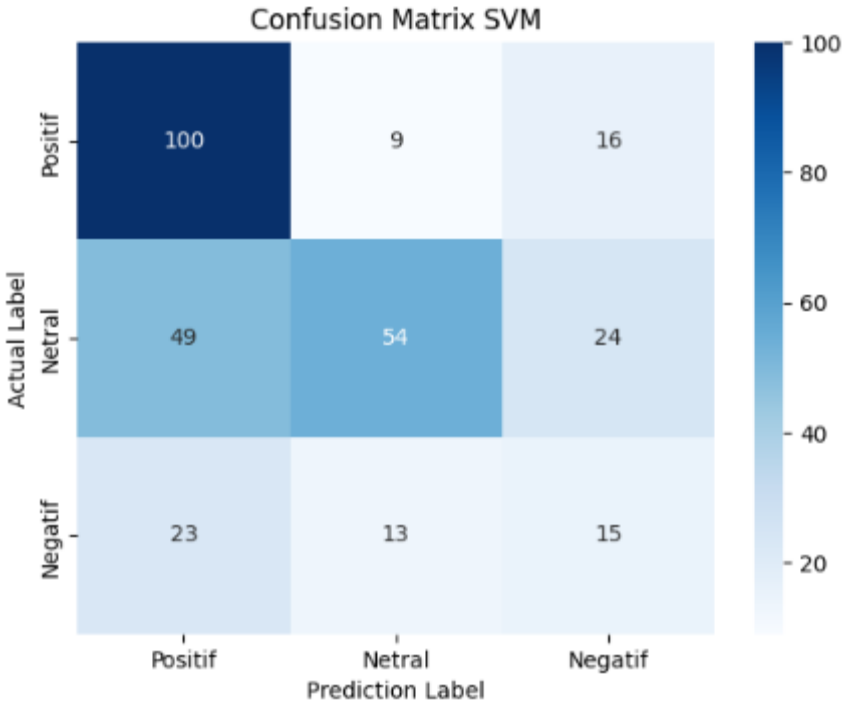
Berikut pada Gambar 4.21 merupakan hasil dari sentimen teks input yang dihasilkan dari TF-IDF menggunakan SVM.

```
Masukkan teks  :indihome kenapa sih gangguan mulu
['indihome', 'kenapa', 'ganggu', 'terus']
Label aktual   : Negatif
Label prediksi : Netral
```

Gambar 4. 21 Hasil Sentimen Analisis dari Input Teks Menggunakan TF-IDF dan SVM

4.8.2 Menggunakan FastText

Untuk proses evaluasi menggunakan *confusion matrix* menggunakan data uji. Dalam proses ini terdapat label positif, netral, dan negatif. Berikut pada Gambar 4.22 merupakan tampilan *confusion matrix.*



Gambar 4. 22 Tampilan Confusion Matrix Menggunakan FastText

Berdasarkan Gambar 4.22 dapat dijelaskan bahwa algoritma SVM diperoleh hasil prediksi pada label positif terklasifikasikan dengan benar sebanyak 100 data, sedangkan kesalahan prediksi dengan 9 data yang masuk ke dalam label netral dan 16 data masuk ke dalam label negatif. Pada label netral dengan sebanyak 54 data terklasifikasikan dengan benar, sedangkan kesalahan prediksi dengan 49 data yang masuk ke dalam label positif dan 24 data yang masuk ke dalam label negatif. Pada label negatif dengan sebanyak 15 data terklasifikasikan dengan benar, sedangkan kesalahan prediksi dengan 23 data yang masuk ke dalam label positif, dan 13 data yang masuk ke dalam label netral. Selanjutnya dilakukan proses perhitungan *accuracy, precision, recall, f1-score.* Berikut pada Gambar 4.23 merupakan hasil dari *confusion matrix* tersebut.

```
Accuracy   : 55.78 %
Precission : 58.36 %
Recall     : 55.78 %
F-Score    : 54.84 %
```

Gambar 4. 23 Hasil Confusion Matrix Menggunakan FastText

Berdasarkan Gambar 4.23 dapat dijelaskan nilai *accuracy* 56.77%, *precision* 59.9%, *recall* 56.77%, dan *f1-score* 56.12%. Selanjutnya dilakukan visualisasi model klasifikasi SVM terhadap data test. Dengan melakukan ini, dapat dengan mudah menganalisis performa model klasifikasi SVM dan memeriksa label yang diprediksi

untuk data test. Berikut pada Gambar 4.24 merupakan hasil evaluasi prediksi untuk data test.

	stem	label	pred_label_svm
0	guna kece lebih lanjut kenan informasi nomor i...	Positif	Positif
1	sudah aku	Netral	Positif
2	contoh promo bisa kalian dapat misal tagih ind...	Positif	Positif
3	admin sudah login myindihome tapi tiap mau con...	Netral	Netral
4	halo bensutoyo seperti sudah informasi belum u...	Positif	Positif
...	...	...	...
298	indihome jelek malem padahal pakai doang laen ...	Negatif	Negatif
299	pakai indihome	Netral	Netral
300	indihome temenku error boleh bantuuu	Netral	Negatif
301	indihome kenapa tidak jelas	Negatif	Netral
302	benar namun hanya perangkat ambil kabel jadi h...	Positif	Positif

303 rows × 3 columns

Gambar 4. 24 Hasil Evaluasi Prediksi Data Uji Menggunakan FastText

Selanjutnya, pengguna dapat melakukan *input text* untuk mengetahui sentimen dari teks tersebut. *Field input* dapat dilihat pada Gambar 4.25.

Gambar 4. 25 Input Teks Baru

Berikut pada Gambar 4.26 merupakan hasil dari sentiment teks input yang dihasilkan dari FastText menggunakan SVM.

```
Masukkan teks  :indihome kenapa sih gangguan mulu
Label aktual   : Negatif
Label prediksi : Negatif
```

Gambar 4. 26 Hasil Sentimen Analisis dari Input Teks Menggunakan FastText dan SVM

4.9 Pembahasan

Berdasarkan hasil pengujian yang telah dilakukan dalam mengimplementasikan algoritma SVM dengan word embedding tanpa FastText (TF-IDF) dan menggunakan FastText dalam analisis sentimen pengguna IndiHome pada media sosial X (Twitter).

Dataset yang digunakan merupakan data yang discrape dari X (Twitter) yang berisi *tweet* dengan keyword "indihome" berjumlah 1537 *tweet.*

Dataset yang telah disimpan selanjutnya dilakukan pre-processing lalu dilakukan pelabelan menggunakan TextBlob. Kemudian dilakukan pembagian dataset menjadi 80% data train dan 20% data test. Data yang telah dibagi selanjutnya diembedding dan dilanjutkan klasifikasi menggunakan SVM. Word embedding yang digunakan yaitu metode FastText dan tanpa menggunakan FastText (TF-IDF). Hasil evaluasi menggunakan confusion matrix, nilai akurasi yang diperoleh tanpa menggunakan FastText (TF-IDF) lebih tinggi dengan *accuracy* 56.11 %, *precision* 72.92 %, *recall* 56.11 %, dan *f-score* 51.14%. Sementara menggunakan *FastText* menghasilkan nilai *accuracy* 55.78%, *precision* 58.36%, *recall* 55.78%, dan *f-score* 54.84%. Dari hasil confusion matrix dapat dilihat pada Tabel 4.1.

Tabel 4. 1 Hasil Confusion Matrix

	Confusion Matrix			
	Accuracy	***Precision***	***Recall***	***F1-Score***
Menggunakan FastText	55.78%	58.36%	55.78%	54.84%
Tanpa Menggunakan FastText	56.11%	72.92%	56.11 %	51.14%

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah penulis lakukan pada analisis sentimen IndiHome didapatkan kesimpulan bahwa proses analisis sentimen berhasil dilakukan dengan SVM dengan menggunakan FastText dan tanpa FastText dan dibangun dengan menggunakan bahasa pemrograman python dan *Google Collab.* Berdasarkan hasil pengujian yang telah dilakukan dengan menggunakan *confusion matrix* didapatkan nilai akurasi dari metode SVM tertinggi dihasilkan tanpa menggunakan FastText (TF-IDF) dengan *accuracy* 56.11 %, *precision* 72.92 %, *recall* 56.11 %, dan *f-score* 51.14%. Sementara menggunakan FastText menghasilkan nilai *accuracy* 55.78%, *precision* 58.36%, *recall* 55.78%, dan *f-score* 54.84%.

5.2 Saran

Dari kesimpulan tersebut, dapat diberikan saran yaitu diharapkan untuk pengembangan penelitian berikutnya dapat melakukan perbandingan dengan metode word embedding lainnya seperti *word2vec.* Sehingga dapat diketahui mana word embedding terbaik dalam melakukan analisis sentimen pengguna IndiHome pada media sosial X (Twitter).

DAFTAR PUSTAKA

Aldisa, R., and P. Maulana. 2022. “Sentimen Opini Masyarakat Terhadap Vaksinasi Booster COVID-19Dengan Perbandingan Metode Naive Bayes, Decision Treedan SVM.” *Lding of Informatics, Technology and Science (BITS)* 106–9.

Alex, S. A., N. Jhanjhi, M. Humayun, A. O. Ibrahim, and A. W. Abulfaraj. 2022. “Deep LSTM Model for Diabetes Prediction with Class Balancing by SMOTE.” *Electronics* 1–17.

Asosiasi Penyelenggara Jasa Internet Indonesia. 2023. “Survei Internet APJII 2023.” *APJII*. Retrieved June 10, 2024 (https://survei.apjii.or.id/).

Asshiddiqi, M., and K. Lhaksmana. 2020. “Perbandingan Metode Decision Tree Dan Support Vector Machine.” *E-Proceeding of Engineering* 9936–48.

Assidyk, A. N., E. B. Setiawan, and I. Kurniawan. 2020. “Analisis Perbandingan Pembobotan TF-IDF Dan TF-RF Pada Trending Topic Di Twitter Dengan Menggunakan Klasifikasi K-Nearest Neighbor.” *E-Proceeding of Engineering* 7773–82.

Averina, Wa Ode May Zhara, Edy Santoso, and Novanto Yudistira. 2020. “Prediksi Persentase Penyelesaian Permohonan Hak Milik Menggunakan Metode Extreme Learning Machine (ELM) (Studi Kasus : Badan Pertanahan Nasional Kabupaten Malang).” 4(7):2236–42.

Baita, A., Y. Pristyanto, and N. Cahyono. 2021. “Analisis Sentimen Mengenai Vaksin Sinovac Menggunakan Algoritma Support Vector Machine (SVM) Dan K-Nearest Neighbor (KNN) .” *Information System Journal (INFOS)* 42–47.

Basar, T. F., D. E. Ratnawati, and I. Arwani. 2022. “Analisis Sentimen Pengguna Twitter Terhadap Pembayaran Cashless Menggunakan Shopeepay Dengan Algoritma Random Forest.” *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer* 1426–33.

Darwis, D., E. Pratiwi, and A. Pasaribu. 2020. “Penerapan Algoritma SVM Untuk Analisis Sentimen Pada Data Twitter Komisi Pemberantasan Korupsi Republik Indonesia.” *Jurnal Ilmiah Edutic* 1–11.

Fani, A. A., P. Sudarmaningtyas, and V. Nurcahyawati. 2020. “Analisis Sentimen Review Pelanggan Pada Layanan Indihome Menggunakan Algoritma Naive Bayer Classifier.” *JSIKA* 1–8.

Fikri, M., T. Sabrila, and Y. Azhar. 2020. “Perbandingan Metode Naïve Bayes Dan Support Vector Machine Pada Analisis Sentimen Twitter.” *SMATIKA Jurnal*.

Fitriana, D. N. 2020. “Klasifikasi Data Tweet Dengan Menggunakan Metode Klasifikasi Multi-Class Support Vector Machine (SVM).” *E-Proceeding of Engineering*.

Fitriasti, N., and D. Priansa. 2021. “Strategi Direct Marketing Dalam Rangka Merangsang Minat Penggunaan Produk Indihome (Studi Kasus Pada PT. Telekomunikasi Indonesia,Tbk. STO Dago Pada Tahun 2021).” *E-Proceeding of Applied Science* 665–71.

Fremmuzar, P., and A. Baita. 2023. “Uji Kernel SVM Dalam Analisis Sentimen Terhadap Layanan Telkomsel Di Media Sosial Twitter.” *Omputika: Jurnal Sistem Komputer* 117–86.

Gunawan, Y. 2021. “FastText Word Embedding and Random Forest Classifier for User Feedback Sentiment Classification in Bahasa Indonesia.” *Jurnal Teknik Informatika*.

Hasri, C. F., and D. Alita. 2022. “Penerapan Metode Naive Bayes Classifier Dan Support Vector Machine Pada Analisis Sentimen Terhadap Dampak Virus Corona Di Twitter.” *Jurnal Informatika Dan Rekayasa Perangkat Lunak (JATIKA)* 145–60.

Idris, I., Y. Mustofa, and I. Salihi. 2023. “Analisis Sentimen Terhadap Penggunaan Aplikasi Shopee Mengunakan Algoritma Support Vector Machine (SVM).” *JJEEE Jambura Journal of Electrical and Electronics Engineering* 32–35.

Islamy, M., Indriati, and P. Adikara. 2022. “Analisis Sentimen IMDB Movie Reviews Menggunakan Metode Long ShortTerm Memory Dan FastText.” *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer* 4106–15.

Isnain, A. R., and e al. 2021. “Sentimen Analisis Publik Terhadap Kebijakan Lockdown Pemerintah Jakarta Menggunakan Algoritma SVM.” *JDMSI* 31–37.

Karyawati, A. E. 2023. “Comparasion of Different Kernel Functions of SVM Claasifiacation Method for Spam Detection.” *JITK*.

Khder, Moaiad Ahmad. 2021. “Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application.” *International Journal of Advances in Soft Computing and Its Applications* 13(3):144–68. doi: 10.15849/ijasca.211128.11.

Khomsah, S. 2021. “Sentiment Analysis On YouTube Comments Using Word2Vec and Random Forest.” *Jurnal Informatika Dan Teknologi Informasi* 18(1):61–72.

Kilimci, Z. 2020. “Sentiment Analysis Based DirectionPrediction in Bitcoin Using Deep Learning Algorithms and Word Embedding Models.” *International Journal of Intelligent Systems and Applications in Engineering* 60–65.

Mahdi, F. A., and e al. 2021. “Pengaruh Principal Component Analysis Terhadap Akurasi Model Machine Learning Dengan Algoritma Artificial Neural Network Untuk Prediksi Kebangkrutan Perusahaan.” *Seminar Nasional Mahasiswa Ilmu Komputer Dan Aplikasinya (SENAMIKA)*.

Mara, A. T., E. Sediyono, and H. Purnomo. 2021. “Penerapan Algoritma K-Nearest Neighbors Pada Analisis Sentimen Metode Pembelajaran Dalam Jaringan (DARING) Di Universitas Kristen Wira Wacana Sumba.” *Journal of Informatics Engineering* 25–26.

Mas Diyasa, I. Gede Susrama, Ni Made Ika Marini Mandenni, Mohammad Idham Fachrurrozi, Sunu Ilham Pradika, Kholilul Rachman Nur Manab, and Nyoman Rahadi Sasmita. 2021. “Twitter Sentiment Analysis as an Evaluation and Service Base On Python Textblob.” *IOP Conference Series: Materials Science and Engineering* 1125(1):1–13. doi: 10.1088/1757-899x/1125/1/012034.

Maulidina, Mega Kurnia. 2020. “Analisis Sentimen Komentar Warganet Terhadap Postingan Instagram Menggunakan Metode Naive Bayes Classifier Dan TF-IDF (Studi Kasus: Instagram Gubernur Jawa Barat Ridwan Kamil).” *Naskah Publikasi Unversitas Teknologi Yogyakarta*.

Mota, F., and I. Cilento. 2020. “Competence for Internet Use: Integrating Knowledge, Skills, and Attitudes.” *Computers and Education Open*.

Mustasaruddin, Budianita, E., M. Fikry, and F. Yanto. 2023. “Klasifikasi Sentiment Review Aplikasi MyPertamina Menggunakan Word Embedding FastText Dan SVM (Support Vector Machine).” *Jurnal Sistem Komputer Dan Informatika (JSON)*.

Nastiti, S., D. Ramadan, and R. Tulloh. 2021. “Sistem Monitoring Untuk Laporan Gangguan Indihome Dengan Bot Telegram .” *E-Proceeding of Applied Science*.

Nurdin, A., and e al. 2020. “Perbandingan Kinerja Word Embedding Word2vec, Glove, Dan Fasttext Pada Klasifikasi Teks.” *Jurnal Teknokompak* 74–79.

Nurkholis, A. 2021. “Comparison of Kernel Support Vector MachineMulti-ClassinPPKM Sentiment Analysis on Twitter.” *Jurnal Resti*.

Pamungkas, K., L. Aridinanti, and W. Wibowo. 2022. “Analisis Sentimen Pelaporan Masyarakat Di Situs Media Centre Surabaya Dengan Naïve Bayes Classifier.” *Jurnal Teknik ITS*.

Pardede, J., and I. Pakpahan. 2023. “Analisis Sentimen Penanganan Covid-19 Menggunakan Metode Long Short-TermMemory Pada Media Sosial Twitter.” *Jurnal Publikasi Teknik Informatika* 12–25.

PT Telkom. 2022. “Laporan Tahunan Telkom 2022.” *PT Telkom*. Retrieved June 11, 2024 (https://www.telkom.co.id/sites/about-telkom/id_ID/page/ir-laporan-tahunan-150).

Purnamasari, Desi, Muhammad Adi, Khairul Anshary, Jurusan Informatika, Fakultas Teknik, and Universitas Siliwangi. 2023. “Particle Swarm Optimization Dan Genetic Algorithm Untuk Analisis Sentimen Pemekaran Papua Di Twitter Berbasis Support Vector Machine.” 20(2):177–90.

Putri, S. H., Yuhandri, and G. W. Nurcahyo. 2021. “Prediksi Pencapaian Target Peserta Keluarga Berencana Pasca Persalinan Menggunakan Algoritma Backpropagation.” *Jurnal Sistim Informasi Dan Teknologi* 3(3):176–82.

Rabbani, S. 2023. “Perbandingan Evaluasi Kernel SVM Untuk Klasifikasi Sentimen Dalam Analisis Kenaikan Harga BBM.” *MALCOM: Indonesian Journal of Machine Learning and Computer Science*.

Rachman, F., and S. Pramana. 2020. “Analisis Sentimen Pro Dan Kontra Masyarakat Indonesia Tentang Vaksin COVID-19 Pada Media Sosial Twitter.” *Indonesian of Health Information Management Journal* 100–109.

Raihan, M., and E. Setiawan. 2021. “Aspect Based Sentiment Analysis with FastText Feature Expansion and Support Vector Machine Method on Twitter.” *JURNAL RESTI (Rekayasa Sistem Dan Teknologi Informasi)*.

Ramlan, R., and e al. 2023. “Analisis Sentimen Pengguna Twitter Menggunakan Support Vector Machine Pada Kasus Kenaikan Harga BBM.” *Jambura Journal of Mathematics* 431–45.

Rifaldo, M. 2021. “Peramalan Kedatangan Wisatawan Mancanegara Ke Indonesia Menurut Kebangsaan Perbulannya Menggunakan Metode Multilayer Perceptron.” *Jurnal Computer Science and Information Technology.*

Saadah, Siti, Fakhira Zahra Z, and Hasna Haifa Z. 2021. “Support Vector Regression (SVR) Dalam Memprediksi Harga Minyak Kelapa Sawit Di Indonesia Dan Nilai Tukar Mata Uang EUR/USD.” *Journal of Computer Science and Informatics Engineering (J-Cosine)* 5(1):85–92. doi: 10.29303/jcosine.v5i1.403.

Sepriadi, N., E. Budianita, M. Fikry, and Pizaini. 2023. “Analisis Sentimen Review Aplikasi MyPertamina Menggunakan Word Embedding.” *INFORMASI (Jurnal Informatika Dan Sistem Informasi)*.

Soen, G. I., Marlina, and Renny. 2022. “Implementasi Cloud Computing Dengan Google Colaboratory Pada Aplikasi Pengolah Data Zoom Participants.” *JITU : Journal Informatic Technology And Communication* 24–30.

Thavareesan, S., and S. Mahesan. 2020. “Sentiment Analysis in Tamil Texts: A Study on Machine Learning Techniques and Feature Representation.” *International Conference on Industrial and Information Systems (ICIIS)* 320–25.

Vina, Fitriyana, and et al. 2023. “Analisis Sentimen Ulasan Aplikasi Jamsostek Mobile Menggunakan Metode Support Vector Machine.” *Jurnal Buana Informatika* 40–49.

Wabang, K., O. Nurhayati, and Farikhin. 2021. “Application of The Naïve Bayes Classifier Algorithm to Classify Community Complaints.” *JURNAL RESTI (Rekayasa Sistem Dan Teknologi Informasi)*.

Widodo, Slamet, and Budi Hartono. 2023. “Analisis Sentimen Pengguna Google Terhadap Destinasi Wisata Di Kota Semarang Menggunakan Metode K-Nearest Neighbor.” *Progresif: Jurnal Ilmiah Komputer* 19(2):545–54.

Wulandari, D., R. Saedudin, and R. Andreswari. 2021. “Analisis Sentimen Ulasan Aplikasi Jamsostek Mobile Menggunakan Metode Support Vector Machine.” *E-Proceeding of Engineering*.

Zikri, A., and Agustian. 2023. “Penerapan Support Vector Machine Dan FastText Untuk Mendeteksi Hate Speech Dan Abusive Pada Twitter.” *Jurnal Media Informatika Budidarma* 436–43.

LAMPIRAN

Lampiran 1. Listing Program

1. Scraping Data

```
import streamlit as st
import time
import numpy as np
# import datetime
from datetime import datetime, timedelta

import urllib
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager

import time
import pandas as pd
import numpy as np
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

st.set_page_config(page_title="Scraping Twitter")

st.markdown("# Scraping Twitter")

email    = st.text_input('Email', '')
_username = st.text_input('Username', '')
_password = st.text_input('Password', '')
title = st.text_input('Search Keyword', '')
s_date    =    st.date_input("Start    Date",    datetime.now()    -
timedelta(days=14))
e_date = st.date_input("End Date", datetime.now())
r_option     =     st.selectbox("Get     replies",("No",     "Yes"),
placeholder="Select if include replies")
# print(urllib.parse.quote(title))
```

```
# progress_bar = st.sidebar.progress(0)
# status_text = st.sidebar.empty()
# last_rows = np.random.randn(1, 1)

# chart = st.line_chart(last_rows)

# for i in range(1, 101):
#          new_rows  =  last_rows[-1,  :]  +  np.random.randn(5,
1).cumsum(axis=0)
#     status_text.text("%i%% Complete" % i)
#     chart.add_rows(new_rows)
#     progress_bar.progress(i)
#     last_rows = new_rows
#     time.sleep(0.05)

# progress_bar.empty()

# Streamlit widgets automatically run the script from top to
bottom. Since
# this button is not connected to any other logic, it just causes
a plain
# rerun.

# def get_driver():
#                                                          return
webdriver.Chrome(service=Service(ChromeDriverManager().install(
)))

@st.cache_data
def convert_df(df):
   return df.to_csv(index=False).encode('utf-8')

if st.button('Run Scraping'):
    if (len(title) < 1) or (len(email) < 1) or (len(_username) <
1) or (len(_password) < 1):
        st.write('Isi dulu semua form.!')
    else:
        # driver = webdriver.Chrome('../chromedriver')
        options = Options()
```

```
        options.add_argument('--disable-gpu')
        options.add_argument('--headless')

        driver                                                  =
webdriver.Chrome(service=Service(ChromeDriverManager().install(
)),  options=options)

        driver.get('https://twitter.com/i/flow/login')
        st.write("Opening Login Page complete.!")

        try:
            # Login twitter
            username = WebDriverWait(driver, 20).until(
                    EC.presence_of_element_located((By.XPATH,
'//*[@id="layers"]/div/div/div/div/div/div/div[2]/div[2]/div/di
v/div[2]/div[2]/div/div/div/div[5]/label/div/div[2]/div/input')
)
                )
            #    username    =    driver.find_element(By.XPATH,
'//*[@id="layers"]/div/div/div/div/div/div/div[2]/div[2]/div/di
v/div[2]/div[2]/div/div/div/div[5]/label/div/div[2]/div/input')
            # username.send_keys('') # Ini untuk email nya
            username.send_keys(email)
            st.write('Filling email complete.!')

            button_next     =     driver.find_element(By.XPATH,
'//*[@id="layers"]/div/div/div/div/div/div/div[2]/div[2]/div/di
v/div[2]/div[2]/div/div/div/div[6]/div')
            button_next.click()

            try:
                time.sleep(5)
                uname = WebDriverWait(driver, 5).until(
                    EC.presence_of_element_located((By.XPATH,
'//*[@id="layers"]/div/div/div/div/div/div/div[2]/div[2]/div/di
v/div[2]/div[2]/div[1]/div/div[2]/label/div/div[2]/div/input'))
                    )
                # uname.send_keys() # Ini untuk username nya
                uname.send_keys(_username) # Ini untuk username
nya
```

```
                st.write('Filling    username    identification
complete.!')
                button_next2   =   driver.find_element(By.XPATH,
'//*[@id="layers"]/div/div/div/div/div/div/div[2]/div[2]/div/di
v/div[2]/div[2]/div[2]/div/div/div/div/div')
                button_next2.click()
            except Exception as e:
                st.write('No Need')

            password = WebDriverWait(driver, 10).until(
                    EC.presence_of_element_located((By.XPATH,
'//*[@id="layers"]/div/div/div/div/div/div/div[2]/div[2]/div/di
v/div[2]/div[2]/div[1]/div/div/div[3]/div/label/div/div[2]/div[
1]/input'))
                )
            password.send_keys(_password)  #  Ini  untuk  password
nya
            st.write('Filling Password complete.!')
            time.sleep(1)
            button_login     =     driver.find_element(By.XPATH,
'//*[@id="layers"]/div/div/div/div/div/div/div[2]/div[2]/div/di
v/div[2]/div[2]/div[2]/div/div[1]/div/div/div/div')
            button_login.click()

            time.sleep(10)
            start_date = s_date.strftime('%Y-%m-%d')
            end_date   = e_date.strftime('%Y-%m-%d')
            if r_option == "No":

driver.get("https://twitter.com/search?q=%22"+urllib.parse.quot
e(title)+"%22%20until%3A"+end_date+"%20since%3A"+start_date+"%2
0-filter%3Alinks%20filter%3Areplies&src=typed_query&f=live")
            else:

driver.get("https://twitter.com/search?q=%22"+urllib.parse.quot
e(title)+"%22%20until%3A"+end_date+"%20since%3A"+start_date+"&s
rc=typed_query&f=live")
            time.sleep(10)
            list_tweet = [];
            tweetssss  = [];
```

```
            total_no_data = 0
            for i in range(200):
                st.write('Read Page ', i)
                driver.execute_script('window.scrollTo(0,
document.body.scrollHeight)') #scrolling
                time.sleep(10) #let content to load
                tweets  =  driver.find_elements(By.CSS_SELECTOR,
'[data-testid="cellInnerDiv"]')

                if total_no_data > 5:
                    break
                for tweet in tweets:
                    tweet_d = tweet.find_elements(By.CLASS_NAME,
'r-kzbkwu')
                    for t in tweet_d:
                        try:
                            not_now                          =
driver.find_element(By.XPATH,
'//*[@id="layers"]/div[2]/div/div/div/div/div/div[2]/div[2]/div
/div[2]/div/div[2]/div[2]/div[2]')
                            not_now.click()
                        except Exception as e:
                            pass

                        spans   =   t.find_elements(By.TAG_NAME,
'span')
                        try:
                            name                             =
spans[0].find_element(By.CLASS_NAME, 'r-qvutc0')
                            if name is not None:
                                time_s                       =
t.find_element(By.TAG_NAME, 'time')
                                tweet_s                      =
t.find_element(By.CSS_SELECTOR, '[data-testid="tweetText"]')

                                if    tweet_s.text    not    in
tweetssss:
                                    #
st.write('================')
                                    # st.write(name.text)
```

```
                                    # st.write(time_s.text)
                                    # st.write(tweet_s.text)

tweetssss.append(tweet_s.text)

list_tweet.append([name.text, time_s.text, tweet_s.text])
                            else:
                                total_no_data+=1
                                st.write('TOTAL    NO    DATA',
total_no_data)
                        except Exception as e:
                            pass
                st.write('Getting Data Complete.! Go to Next')


            df_data          =          pd.DataFrame(list_tweet,
columns=['Username', 'date', 'tweet'])
            # df_data.to_csv('Hasil Scraping.csv')
            driver.quit()

            st.dataframe(df_data)

            csv = convert_df(df_data)

            st.download_button(
               "Press to Download",
               csv,
               "file.csv",
               "text/csv",
               key='download-csv'
            )
        except Exception as e:
            st.write('Error', e)
            driver.quit()

        # return "Complete"
```

2. Input Data dan Pre-processing Data

```
import os
import re
import nltk
import numpy as np
import pandas as pd
from nltk.corpus import stopwords
from textblob import TextBlob
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
nltk.download('stopwords')
nltk.download('punkt')

data = pd.read_csv('dataset.csv') # Read dataset
data

From nltk.corpus import stopwords
# ----- START Setup Stopword and Additional Stopword
stopword             = set(stopwords.words('indonesian'))
additional_stopwords                                          =
pd.read_excel('corpus/Additional_Stopwords.xlsx')['Stopwords
'].tolist() #load kamus stopwords
new_stopwords        = stopword.union(additional_stopwords)
# ----- END

# ----- START setup stemmer kata
factory = StemmerFactory()
stemmer = factory.create_stemmer()
# ----- END

data_normalisasi   = pd.read_excel('corpus/Normalisasi.xlsx')
# load kamus kata normalisasi
kata_normalisasi   = data_normalisasi['Kata'].tolist() # kata
sebelum normalisasi
hasil_normalisasi = data_normalisasi['Normalisasi'].tolist()#
kata sesudah normalisasi

def remove_emojis(data):
    emoj = re.compile("["
```

```
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
        u"\U00002500-\U00002BEF"  # chinese char
        u"\U00002702-\U000027B0"
        u"\U00002702-\U000027B0"
        u"\U000024C2-\U0001F251"
        u"\U0001f926-\U0001f937"
        u"\U00010000-\U0010ffff"
        u"\u2640-\u2642"
        u"\u2600-\u2B55"
        u"\u200d"
        u"\u23cf"
        u"\u23e9"
        u"\u231a"
        u"\ufe0f"  # dingbats
        u"\u3030"
                      "]+", re.UNICODE)
    return re.sub(emoj, '', data)

def cleaning(text):
    # remove URLs
    text =
re.sub('((www\.[^\s]+)|(https?://[^\s]+)|(http?://[^\s]+))',
'', text)
    # remove usernames
    text = re.sub('@[^\s]+', '', text)
    # remove the # in #hashtag
    text = re.sub('#([^\s]+)', '', text)
    # Remove digits number
    text = re.sub("\d+", "", text)
    # Remove Punctuation
    text = re.sub(r'[^\w\s]', '', text)
    # Remove Emojis
    text = remove_emojis(text)
```

```
    # text = ' '.join(re.sub("[@#][A-Za-z0-9_]+|([^0-9A-Za-z
\t])|(\w+:\/\/\S+)+|<.*?>|^https?:\/\/.*[\r\n]*","         ",
text).split(' '))
    text = ' '.join(text.split(' '))
    return text

def caseFolding(text):
    text = str(text).lower()
    return text

def normalisasi_kata(kata):
    token = kata.split()
    for t in range(len(token)):
        if token[t] in kata_normalisasi:
            index_kata = kata_normalisasi.index(token[t])
             #print('Kata ', token[t], ' telah di normalisasi
menjadi ', hasil_normalisasi[index_kata])
            token[t] = hasil_normalisasi[index_kata].lower()
    return " ".join(token)

def tokenizing(text):
    min_len_token = 2 # Minimal huruf pada kata
    text = nltk.tokenize.word_tokenize(text)
    text = [t for t in text if len(t) > min_len_token]
    return text

def filtering(token):
       filter_  =  [t  for  t  in  token  if  t  not  in
additional_stopwords]
    return " ".join(filter_)\

def stemming(filtered_token):
    words = filtered_token.split()
    backstr = []
    for w in words:
        stemmed = stemmer.stem(w)
        backstr.append(stemmed)
    return ' '.join(backstr)
```

3. Pelabelan Data

```
def sentiment_analysis_textblob_lexicon(stemming_words):
    analysis = TextBlob(stemming_words)
    try:
             translate  =  analysis.translate(from_lang='id',
to='en') # Karena texblob digunakan untuk bahasa Inggris maka
perlu menerjemahkan dari bahasa Indonesi ke bahasa Inggris
        score = translate.sentiment.polarity
    except:
        score = analysis.sentiment.polarity

    polarity = ''
    if score < 0:
        polarity = 'Negatif'
    elif score == 0:
        polarity = 'Netral'
    else:
        polarity = 'Positif'
    return score, polarity

input_data = []
stem_data  = []
list_word  = []
disable_duplicate = True
for index, r in data.iterrows():
    print('Proccessing Index '+str(index)+'... ', end = '')
    text = r['tweet']
    _cleaning    = cleaning(text)
    _casefolding = caseFolding(_cleaning)
    _normalisasi  = normalisasi_kata(_casefolding)
    _tokenizing  = tokenizing(_normalisasi)
    _filtering   = filtering(_tokenizing)
    _stemming    = stemming(_filtering)
                          _labelling                          =
sentiment_analysis_textblob_lexicon(_stemming)
```

```
    _process = False
    if disable_duplicate == True:
        if _stemming not in stem_data: # If stem text not in
list
            _process = True
            stem_data.append(_stemming)
        else:
            print('Skip.! (Duplicate data)', end=' ')
    else:
        _process = True

    if _process == True:
        list_word.append(_filtering.split())
        input_data.append({
            "username"      : r['Username'],
            "tweet"         : text,
            "cleaning"      : _cleaning,
            "casefolding"   : _casefolding,
            "normalization" : _normalisasi,
            "tokenizing"    : _tokenizing,
            "filtering"     : _filtering,
            "stemming"      : _stemming,
            "score"         : _labelling[0],
            "label"         : _labelling[1]
        })
    print('Done.!')

df_dataset = pd.DataFrame(input_data)
df_dataset = df_dataset.dropna()
df_dataset



df_dataset.label.value_counts()

t_list = []
for t in list_word:
    for nt in t:
        t_list.append(nt)
```

```
df_t_list = pd.DataFrame(t_list, columns=['Word'])
word_data = df_t_list.groupby(['Word']).size()

df_list_words = pd.DataFrame(word_data,
columns=['Total']).reset_index().sort_values(by=['Total'],
ascending=False)
df_list_words.reset_index(inplace=True, drop=True)
df_list_words



df_dataset.to_csv('hasil_preprocessing.csv', index=False)
df_list_words.to_csv('list_words.csv', index=False)
print('Done')
```

4. Split Data

```
X = list()
y = list()
for index, row in data.iterrows():
    X.append(row['stemming'])
    y.append(row['label'])

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

# Visualisasi Train Data
train_data = {'stem':list(), 'label':list()}
for xt in range(len(X_train)):
    train_data['stem'].append(X_train[xt])
    train_data['label'].append(y_train[xt])
train_data = pd.DataFrame(train_data)
train_data.to_csv('Train data.csv')
train_data

# Menghitung jumlah data setiap label yang ada di data train
train_label_positive = 0
train_label_neutral  = 0
train_label_negative = 0
```

```
for index, row in train_data.iterrows():
    if(row['stem'] != ''):
        if(row['label'] == 'Negatif'):
            train_label_negative += 1
        elif(row['label'] == 'Positif'):
            train_label_positive  += 1
        elif(row['label'] == 'Netral'):
            train_label_neutral  += 1

print('Jumlah data positif pada data train : ',
train_label_positive)
print('Jumlah data netral pada data train : ',
train_label_neutral)
print('Jumlah data negatif pada data train : ',
train_label_negative)

# Visualisasi Test Data
test_data = {'stem':list(), 'label':list()}
for xt in range(len(X_test)):
    test_data['stem'].append(X_test[xt])
    test_data['label'].append(y_test[xt])
test_data = pd.DataFrame(test_data)
test_data.to_csv('Tes data.csv')
test_data

# Menghitung jumlah data setiap label yang ada di data test
test_label_positive = 0
test_label_neutral  = 0
test_label_negative = 0

for index, row in test_data.iterrows():
    if(row['stem'] != ''):
        if(row['label'] == 'Negatif'):
            test_label_negative += 1
        elif(row['label'] == 'Positif'):
            test_label_positive  += 1
        elif(row['label'] == 'Netral'):
            test_label_neutral  += 1
```

```
print('Jumlah data positif pada data test : ',
test_label_positive)
print('Jumlah data netral pada data test : ',
test_label_neutral)
print('Jumlah data negatif pada data test : ',
test_label_negative)
```

5. Word Embedding

a. Tanpa Menggunakan FastText (TF-IDF)

```
# TF-IDF Vectorizer
vectorizer = TfidfVectorizer(norm='l2') # Membuat TF ID-F
Vectorizer
X_train_vec = vectorizer.fit_transform(X_train).toarray()
# Mencari vocabulary
X_test_vec = vectorizer.transform(X_test).toarray()

# untuk memperoleh feature name kata
feature_names = vectorizer.get_feature_names_out() #
Mendapatkan Feature Names matrix

#menghitung frekuensi
count_vector = CountVectorizer()
X_train_count =
count_vector.fit_transform(X_train).toarray()
X_test_count = count_vector.transform(X_test).toarray()
tf_train = pd.DataFrame(X_train_count,index = [f'D{i+1}' for
i in range(len(X_train))],columns=feature_names)
tf_train.to_csv('tfidf/frekuensi data train.csv')

tfidf_train = pd.DataFrame(X_train_vec,index = [f'D{i+1}' for
i in range(len(X_train))],columns=feature_names)
tfidf_train['label'] = y_train
tfidf_train.to_csv('tfidf/TF-IDF data train.csv')
tfidf_train
names =
tfidf_train.groupby(['label']).size().index.to_list()
values = tfidf_train.groupby(['label']).size().to_list()

plt.figure(figsize=(3, 3))
```

```
plt.bar(names, values)
plt.xlabel('Label')
plt.ylabel('Jumlah Data')
plt.title('Perbandingan jumlah data setiap label pada Data
Training')
plt.show()
from imblearn.over_sampling import SMOTE

# Membuat objek SMOTE
smote = SMOTE(random_state=42)

# Melakukan oversampling pada data X_train dan y_train
X_train_resampled,            y_train_resampled            =
smote.fit_resample(X_train_vec, y_train)
tfidf_train_new  =  pd.DataFrame(X_train_resampled,index  =
[f'D{i+1}'              for              i              in
range(len(X_train_resampled))],columns=feature_names)
tfidf_train_new['label'] = y_train_resampled
tfidf_train_new.to_csv('tfidf/TF-IDF data train-SMOTE.csv')
tfidf_train_new
print('Total label data training setelah balancing data:')
tfidf_train_new.label.value_counts()
names                                                      =
tfidf_train_new.groupby(['label']).size().index.to_list()
values = tfidf_train_new.groupby(['label']).size().to_list()

plt.figure(figsize=(3, 3))
plt.bar(names, values)
plt.xlabel('Label')
plt.ylabel('Jumlah Data')
plt.suptitle('Perbandingan  label  data  training  setelah
balancing data')
plt.show()

tf_test = pd.DataFrame(X_test_count,index = [f'D{i+1}' for i
in range(len(X_test))],columns=feature_names)
tf_test.to_csv('tfidf/frekuensi data test.csv')

tfidf_test = pd.DataFrame(X_test_vec,index = [f'D{i+1}' for i
in range(len(X_test))],columns=feature_names)
```

```
tfidf_test['label'] = y_test
tfidf_test.to_csv('tfidf/TF-IDF data test.csv')
tfidf_test
names = tfidf_test.groupby(['label']).size().index.to_list()
values = tfidf_test.groupby(['label']).size().to_list()

plt.figure(figsize=(3, 3))
plt.bar(names, values)
plt.xlabel('Label')
plt.ylabel('Jumlah Data')
plt.title('Perbandingan jumlah data setiap label pada Data Testing')
plt.show()
```

b. Menggunakan FastText

```
max_len = 0 # Inisialisasi variabel max_len
for r in data['stemming'].tolist(): # Perulangan untuk list stemming
    if len(r.split(' ')) > max_len: # Jika jumlah kata lebih besar dari max_len
        max_len = len(r.split(' ')) # Menyimpan jumlah kata
max_len # Maximum jumlah kata
from gensim.models import FastText
train_data = list(map(lambda x: x.split(), data['stemming']))

model = FastText(sentences=train_data, # Data untuk pembelajaran
                 vector_size=max_len, # Dimensi vektor fitur
                 window=3, # jarak maksimum antara kata saat ini dan kata yang diprediksi dalam sebuah kalimat.
                 min_count=3, # mengabaikan semua kata dengan frekuensi total lebih rendah dari min_count.
                 sg = 1, # 1 untuk skip-gram model, 0 untuk Continous Bag of Word (CBOW)
                 hs = 0, #jika 1, softmax hierarkis akan digunakan untuk pelatihan model. Jika disetel ke 0 (default), dan `negatif` bukan nol, pengambilan sampel negatif akan digunakan.
```

```
                    negatif = 5, # jika > 0, pengambilan sampel
negatif akan digunakan, untuk menentukan berapa banyak "noise
word" yang harus digambar (biasanya antara 5-20). Standarnya
adalah 5. Jika disetel ke 0, tidak ada sisi negatif yang
digunakan.
                    workers= 4, # untuk pelatihan model
                    seed = 0) # untuk penghasil angka acak
# Fungsi untuk menghasilkan vektor dokumen
def generate_document_vectors(model, data):
    document_vectors = []
    # Iterasi melalui setiap dokumen dalam data
    for document in data:
        # Membuat vektor untuk setiap kata dalam dokumen
menggunakan word embeddings dari model
        vectors = [model.wv[word] for word in document if word
in model.wv]

        # Memeriksa apakah ada vektor yang dihasilkan untuk
dokumen (jika tidak ada, kemungkinan semua kata tidak
ditemukan dalam vocabulary model)
        if vectors:
            # Menghitung rata-rata vektor kata untuk
menghasilkan vektor representatif untuk dokumen
            document_vector = np.mean(vectors, axis=0)

            # Menambahkan vektor dokumen ke dalam list
            document_vectors.append(document_vector)
        else:
            # Jika tidak ada kata yang ditemukan dalam
kosakata model, anggap vektor nol
            X_test_vectors.append([0] * model.vector_size)

    # Mengubah list vektor dokumen menjadi array numpy
    return np.array(document_vectors)

# Menghasilkan vektor untuk data latih dan uji
X_train_vectors = generate_document_vectors(model, X_train)
X_test_vectors = generate_document_vectors(model, X_test)
from collections import Counter
# Menghitung jumlah masing-masing label
```

```
label_counts = Counter(y_train)

# Menampilkan jumlah masing-masing label
for label, count in label_counts.items():
    print(f"{label}: {count}")
names = list(label_counts.keys())
values = list(label_counts.values())

plt.figure(figsize=(3, 3))
plt.bar(names, values)
plt.xlabel('Label')
plt.ylabel('Jumlah Data')
plt.title('Perbandingan jumlah data setiap label pada Data
Training')
plt.show()
from imblearn.over_sampling import SMOTE

# Membuat objek SMOTE
smote = SMOTE(random_state=42)

# Melakukan oversampling pada data X_train dan y_train
X_train_resampled,            y_train_resampled            =
smote.fit_resample(X_train_vectors, y_train)
# Menghitung jumlah masing-masing label
label_counts = Counter(y_train_resampled)

# Menampilkan jumlah masing-masing label
for label, count in label_counts.items():
    print(f"{label}: {count}")
names = list(label_counts.keys())
values = list(label_counts.values())

plt.figure(figsize=(3, 3))
plt.bar(names, values)
plt.xlabel('Label')
plt.ylabel('Jumlah Data')
plt.suptitle('Perbandingan  label  data  training  setelah
balancing data')
plt.show()
from collections import Counter
```

```
# Menghitung jumlah masing-masing label
label_counts = Counter(y_test)

# Menampilkan jumlah masing-masing label
for label, count in label_counts.items():
    print(f"{label}: {count}")
names = list(label_counts.keys())
values = list(label_counts.values())

plt.figure(figsize=(3, 3))
plt.bar(names, values)
plt.xlabel('Label')
plt.ylabel('Jumlah Data')
plt.title('Perbandingan jumlah data setiap label pada Data
Testing')
plt.show()
```

6. Klasifikasi Teks Menggunakan SVM

a. Tanpa Menggunakan FastText (TF-IDF)

```
parameters = {'C': [1, 10, 100],
              'gamma': [5, 4, 3]}

Svm = GridSearchCV(SVC(kernel='rbf'), param_grid=parameters)
Svm.fit(X_train_resampled, y_train_resampled)
pred_svm = Svm.predict(X_test_vec)
# Menampilkan kombinasi parameter terbaik yang diperoleh
print('Best parameter : ', Svm.best_params_)
```

b. Menggunakan FastText

```
parameters = {'C': [1, 10, 100],
              'gamma': [5, 4, 3]}

Svm = GridSearchCV(SVC(kernel='rbf'), param_grid=parameters)
Svm.fit(X_train_resampled, y_train_resampled)
pred_svm = Svm.predict(X_test_vectors)
# Menampilkan kombinasi parameter terbaik yang diperoleh
print('Best parameter : ', Svm.best_params_)
```

7. Evaluasi Confusion Matrix

a. Tanpa Menggunakan FastText (TF-IDF)

```
cm_svm = confusion_matrix(y_test, pred_svm,
labels=['Positif', 'Netral', 'Negatif'])
class_label = ['Positif', 'Netral', 'Negatif']
df_confusion = pd.DataFrame(cm_svm, index = class_label,
columns = class_label)

sns.heatmap(df_confusion, annot=True, fmt='d',
cmap=plt.cm.Blues)
plt.title('Confusion Matrix SVM')
plt.xlabel('Prediction Label')
plt.ylabel('Actual Label')
plt.show()
_accuracy = round(accuracy_score(y_test, pred_svm)*100, 2)
_precision = round(precision_score(y_test, pred_svm,
average='weighted')*100, 2)
_recall = round(recall_score(y_test, pred_svm,
average='weighted')*100, 2)
_fscore = round(f1_score(y_test, pred_svm,
average='weighted')*100, 2)

print('Accuracy :', _accuracy, '%')
print('Precission :', _precision, '%')
print('Recall :', _recall, '%')
print('F-Score :', _fscore, '%')
test_data['pred_label_svm'] = pred_svm
test_data.to_csv('hasil.csv')
test_data
```

b. Menggunakan FastText

```
cm_svm = confusion_matrix(y_test, pred_svm,
labels=['Positif', 'Netral', 'Negatif'])
class_label = ['Positif', 'Netral', 'Negatif']
df_confusion = pd.DataFrame(cm_svm, index = class_label,
columns = class_label)

sns.heatmap(df_confusion, annot=True, fmt='d',
cmap=plt.cm.Blues)
```

```
plt.title('Confusion Matrix SVM')
plt.xlabel('Prediction Label')
plt.ylabel('Actual Label')
plt.show()
_accuracy   = round(accuracy_score(y_test, pred_svm)*100, 2)
_precision    =    round(precision_score(y_test,    pred_svm,
average='weighted')*100, 2)
_recall            =   round(recall_score(y_test,   pred_svm,
average='weighted')*100, 2)
_fscore               =   round(f1_score(y_test,   pred_svm,
average='weighted')*100, 2)

print('Accuracy   :', _accuracy, '%')
print('Precission :', _precision, '%')
print('Recall     :', _recall, '%')
print('F-Score    :', _fscore, '%')
      test_data['pred_label_svm'] = pred_svm
      test_data.to_csv('hasil.csv')
test_data
```

Lampiran 2. ***Curriculum Vitae***

Curriculum Vitae

BIODATA

Nama	:	Sarah Syafitri
Tempat & Tanggal Lahir	:	Batam, 07 Februari 1999
Jenis Kelamin	:	Perempuan
Alamat	:	Taman Kota Baloi Blok F5 No. 22B - Batam
Agama	:	Islam
Kewarganegaraan	:	Indonesia
No. HP	:	085977876491
Email	:	sarahfitri7299@gmail.com

RIWAYAT PENDIDIKAN

SD	:	SD Kartini 2 - Batam (2005-2011)
SMP	:	SMP Kartini 2 - Batam (2011-2014)
SMA	:	SMA Kartini - Batam (2014-2017)
Perguruan Tinggi	:	S1 Ilmu Komputer Universitas Sumatera Utara - Medan (2017 - 2024)

RIWAYAT PENDIDIKAN

SD	:	SD Kartini 2 - Batam (2005-2011)
SMP	:	SMP Kartini 2 - Batam (2011-2014)
SMA	:	SMA Kartini - Batam (2014-2017)
Perguruan Tinggi	:	S1 Ilmu Komputer Universitas Sumatera Utara - Medan (2017 - 2024)

PENGALAMAN KERJA

PT. McDermott Indonesia Yard IT Department (2020)	:	Meningkatkan Efisiensi Transaksi Aplikasi Toolhound di Consumable Warehouse PT McDermott Indonesia Lapangan Fabrikasi Batam