LAPORAN PENELITIAN

PERBANDINGAN MODEL *MACHINE LEARNING* DAN *DEEP LEARNING* TERHADAP ANALISIS SENTIMEN PELANGGAN SHOPEE

Peneliti:

ALISSA SAUSAN NABILA

NIM. 190705057

Jenis Penelitian	**Penelitian Inter Disipliner**
Bidang Ilmu Kajian	**Machine Learning**
Dosen Peneliti	**Hendri Ahmadian, M.I.M**

UNIVERSITAS ISLAM NEGERI AR-RANIRY BANDA ACEH FAKULTAS SAINS DAN TEKNOLOGI

PRODI TEKNOLOGI INFORMASI

DESEMBER 2023

LEMBARAN IDENTITAS DAN PENGESAHAN LAPORAN PENELITIAN PUSAT PENELITIAN DAN PENERBITAN LP2M UIN AR-RANIRY TAHUN 2023

1.	a. Judul	:	Perbandingan Model Machine Learning dan Deep Learning terhadap analisis sentimen pelanggan Shopee
	b. Jenis Penelitian	:	Penelitian Inter Disipliner
	c. No. Registrasi	:	-
	d. Bidang Ilmu yang diteliti	:	Machine Learning
2.	Peneliti		
	a. Nama Lengkap	:	Alissa Sausan Nabila
	b. Jenis Kelamin	:	Perempuan
	c. NIM	:	190705057
	d. Fakultas/Prodi	:	Sains danTeknologi/Teknologi Informasi
	e. Anggota Peneliti 1		
	Nama Lengkap	:	Hendri Ahmadian, M.I.M
	Jenis Kelamin	:	Laki Laki
	Fakultas/Prodi	:	Sains danTeknologi/Teknologi Informasi
	f. Aggota Peneliti 2 *(Jika Ada)*		
	Nama Lengkap	:	Bustami, M.Sc
	Jenis Kelamin	:	Laki Laki
	Fakultas/Prodi	:	Sains danTeknologi/Teknologi Informasi
3.	Lokasi Kegiatan	:	Fakultas Sains dan Teknologi UIN Ar-Raniry Banda Aceh
4.	Jangka Waktu Pelaksanaan	:	6 (Enam) Bulan
5.	Tahun Pelaksanaan	:	2023
6.	Jumlah Anggaran Biaya	:	-
7.	Sumber Dana	:	Mandiri
8.	Output dan Outcome	:	-

Banda Aceh, 28 Desember 2023

Mengetahui,
Dosen Pembimbing I

Hendri Ahmadian, M.I.M
NIP. 198301272015032003

Pelaksana,

Alissa Sausan Nabila
NIM. 19705057

Menyetujui:

Ketua Prodi. Teknologi Informasi

Ima Dwitawati

NIP. 198210132014032002

ABSTRAK

Nama : Alissa Sausan Nabila
NIM : 190705057
Program Studi : Teknologi Informasi

Judul : Perbandingan Model *Machine Learning* dan *Deep Learning* Terhadap Analisis Sentimen Pelanggan Shopee

Tanggal Sidang : 28 Desember 2023
Jumlah Halaman : 81 Halaman
Pembimbing I : Hendri Ahmadian, M.I.M
Pembimbing II : Bustami, M. Sc
Kata Kunci : Analisis Sentimen, *Machine Learning, Deep, Accuracy*

Aplikasi Shopee adalah salah satu alat *e-commerce* yang dapat mendukung pemenuhan kebutuhan manusia. Analisis sentimen merupakan penentuan nada emosional pada komentar atau cuitan dari penilaian masyarakat terhadap sesuatu termasuk kinerja aplikasi shopee. Penelitian ini memfokuskan pada perbandingan model *machine learning* dan *deep learning* terhadap analisis sentimen pelanggan shopee. Dalam penelitian ini menggunakan dataset komentar pelanggan shopee sebanya 6.002 kalimat komentar. Pada model *machine learning*, metode *Logistic Regression* memperoleh nilai *accuracy* tertinggi yaitu sebesar 94.58%, selanjutnya untuk metode *naive bayes* mendapatkan nilai *accuracy* sebesar 93.67%, dan metode *Multinomial Naive Bayes* mendapatkan nilai *accuracy* sebesar 92.33%. Sedangkan pada model *deep learning*, metode *BERT* memperoleh nilai *accuracy* sebesar 92.83% dan metode *multilingual BERT* memperoleh nilai accuracy sebesar 97.41%. Pada penelitian ini, model *deep learning* memiliki nilai *accuracy* lebih tinggi jika dibandingkan dengan m*achine learning*.

Kata Kunci: Analisis Sentimen, *Machine Learning, Deep Learning, Accuracy*

ABSTRACT

Name : Alissa Sausan Nabila
Student Number : 190705057
Department : Information Technology

Title : Comparison of Machine Learning and Deep Learning Models for Shopee Customer Sentiment Analysis

Date : 28 Desember 2023
Number of pages : 81 Pages
Supervisor I : Hendri Ahmadian, M.I.M
Supervisor II : Bustami, M. Sc
Keywords : Sentiment Analysis, Machine Learning, Deep Learning, Accuracy

The Shopee application is an e-commerce tool that can support the fulfillment of human needs. Sentiment analysis is determining the emotional tone of comments or tweets from people's assessments of something, including the performance of the Shopee application. This research focuses on comparing machine learning and deep learning methods in analyzing shopee user sentiment. This research uses a dataset of Shopee customer comments of 6,002 comment sentences. In the machine learning method, the Logistic Regression method obtained the highest accuracy value, namely 94,58%, then the Naive Bayes method obtained an accuracy value of 93,67%, and the Multinomial Naive Bayes method obtained an accuracy value of 92,33%. While for the deep learning method, the BERT method obtained an accuracy value of 92,83% and the multilingual BERT method obtained an accuracy value of 97,41%. In this research, the deep learning model has a higher accuracy value when compared to machine learning.

Keywords: Sentiment Analysis, Machine Learning, Deep Learning, Accuracy

KATA PENGANTAR

Dengan menyebut nama Allah SWT yang Maha Pengasih lagi Maha Penyayang. Puji syukur penulis panjatkan atas kehadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan tugas akhir dengan judul "Perbandingan Model *Machine Learning* dan *Deep Learning* Terhadap Analisis Sentimen Pelanggan Shopee". Shalawat serta salam kepada Rasulullah SAW yang mengantarkan manusia dari zaman kegelapan ke zaman yang terang benderang.

Dalam proses penyusunan tugas akhir ini penulis menyadari bahwa tidak dapat menyelesaikan tugas akhir jika tidak adanya dukungan, bimbingan, motivasi dan bantuan dari berbagai pihak yang telah membantu penulis dalam menyelesaikan tugas akhir ini. Dengan kerendahan hati penulis ingin berterimakasih kepada:

1. Kedua orangtua dan keluarga yang senantiasa memberikan dukungan dan doa kepada saya dalam menyusun tugas akhir ini.
2. Bapak Hendri Ahmadian, M.I.M selaku pembimbing I dan Bapak Bustami, M.Sc selaku pembimbing II yang senantiasa memberikan arahan dan bimbingan dalam penyusunan tugas akhir ini.
3. Ketua dan sekretaris prodi Teknologi Informasi Ibu Ima Dwitawati, M.B.A dan Bapak Khairan AR, M.Kom yang telah memberikan bimbingan dan arahan dalam penyusunan tugas akhir ini.
4. Ibu Cut Ida Rahmadiana, S.Si selaku staff Prodi Teknologi Informasi yang senantiasa membantu penulis dalam pemberkasan administrasi.
5. Bapak Dr. Ir, M. Dirhamsyah, M.T.,IPU selaku Dekan Fakultas Sains dan Teknologi UIN Ar-Raniry yang telah mendukung dan memberi motivasi untuk kami.
6. Bapak dan Ibu dosen Prodi Teknologi Informasi yang tekah memberikan ilmu kepada kami dalam bidang Teknologi Informasi.

7. Teman-Teman yang selalu memberikan dukungan moral dalam menyelesaikan penyusunan tugas akhir ini.
8. Semua pihak yang telah membantu dalam menyelesaikan proposal tugas akhir yang tidak dapat disebutkan satu persatu.

Penulis menyadari bahwa masih banyak kekurangan dari tugas akhir ini. Oleh karena itu, penulis memohon maaf atas segala kekurangan yang terjadi selama proses penyusunan tugas akhir. Akhir kata, semoga tugas akhir ini dapat memberikan kontribusi yang bermanfaat bagi penulis dan pembaca.

Banda Aceh, 28 Desember 2023

Penulis

Alissa Sausan Nabila

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

I.1. Latar Belakang

Pengguna internet digunakan untuk lebih dari sekedar komunikasi di dunia yang berevolusi secara teknologi saat ini. Mereka juga dapat melakukan bisnis, perjalanan, pekerjaan, atau melakukan pembelian. Perilaku konsumen dalam membeli dan menggunakan produk yang dibeli secara online akan dipengaruhi oleh keinginan konsumen untuk melakukan penilaian produk secara cepat, baik secara langsung maupun tidak langsung. Kita dapat dengan mudah menemukan apa yang kita inginkan melalui *e-commerce*, yang juga dapat digunakan sebagai peluang bisnis untuk membuat toko online karena begitu banyak orang yang menggunakan internet.

Aplikasi Shopee adalah salah satu alat *e-commerce* yang dapat mendukung pemenuhan kebutuhan manusia. Pertumbuhan Shopee di Indonesia sangat cepat dibandingkan dengan pertumbuhan pasar lainnya. Pada kuartal I tahun 2022, Shopee masih memiliki keunikan di bisnis *e-commerce* Asia Tenggara. Shopee menjadi toko online dengan pengunjung terbanyak kedua setelah Tokopedia yaitu 132,8 Juta pengunjung pada kuartal I 2022, mengalahkan beberapa toko online terkenal lainnya. Jumlah kunjungan web shopee di indonesia meningkat +117% dibandingkan jumlah kunjangan pada kuartal I tahun 2018 (Leomongga, 2021). Namun aplikasi toko online tentu memiliki kelebihan dan kekurangan termasuk shopee. Pengalaman pengguna shopee sebagai aplikasi dengan pengguna paling aktif kedua sering diungkapkan melalui kolom komentar baik berupa kritik maupun kepuasan. Perasaan negatif atau positif pengguna tentang suatu produk berhubungan dengan berbagai aspek (Yasin, 2023). Masalah ini disebut *Aspect-Based Sentiment Analysis* (ABSA).

Analisis sentimen merupakan penentuan nada emosional pada komentar atau cuitan dari penilaian masyarakat terhadap sesuatu termasuk kinerja aplikasi shopee. Analisis sentimen dapat membantu untuk memperoleh gambaran umum persepsi

masyarakat dengan cara mengelompokkan jenis opini menjadi kategori positif, negatif atau netral terhadap pelayanan Shopee. Penelitian sebelumnya telah dilakukan untuk menganalisis sentimen pengguna aplikasi shopee. Diantaranya penelitian yang dilakukan oleh Aditya Hastami Ruger (Aditya, 2021) yang berjudul "*Sentimen Analisis* Pelanggan Shopee di Twitter menggunakan *Algoritma Naive Bayes*" dengan nilai akurasi sebesar 97%. Penelitian selanjutnya berjudul "Klasifikasi Ulasan Aplikasi Shopee Menggunakan Algoritma *Probabilistic Neural Network* dan *K-Nearest Neighbor*" oleh M. Afdal dkk (Afdal, 2022) dengan hasil akurasi KNN sebesar 91.43% dan PNN sebesar 85.71%.

Dalam melakukan sebuah analisis sentimen banyak model yang dapat digunakan seperti model *Machine learning* dan *Deep learning*. *Machine Learning* atau dikenal pembelajaran mesin adalah ilmu komputer yang bisa bekerja tanpa di program secara eksplisit. *Machine Learning* ini merupakan kecerdasan buatan yang mempelajari bagaimana membuat atau mengolah data. Secara definisi, *Machine Learning* merupakan ilmu yang mempelajari tentang algoritma dan model statistik yang digunakan oleh komputer untuk melakukan *task* tertentu (Rozzi, 2020).Pada umumnya menggunakan metode dengan pendekatan *machine learning* berbasis teks seperti *Naive Bayes* dan *Logistic Regression* karena tingkat akurasi dari kedua metode tersebut relatif tinggi. *Logistic Regression* adalah salah satu metode statistika yang sering digunakan untuk mengklasifikasikan sejumlah pengamatan dengan respon biner ke dalam beberapa kelompok berdasarkan satu atau lebih variabel prediktor. Melalui metode ini akan dihasilkan peluang dari masing-masing kategori respon yang akan dijadikan sebagai pedoman pengklasifikasian dan suatu pengamatan akan masuk ke dalam respon kategori tertentu berdasarkan nilai peluang yang terbesar. *Naive Bayes* merupakan salah satu metode klasifikasi yang memiliki kelebihan dalam perhitungan algoritma yang relatif sederhana dan menghasilkan nilai akurasi yang tinggi (Putro, 2020). *Naive Bayes* adalah metode yang digunakan dalam statistika untuk menghitung peluang dari suatu hipotesis, *naive bayes* menghitung peluang suatu variabel prediktor yang dimiliki dan menentukan variabel respon yang memiliki

peluang paling tinggi. Metode *naive bayes* juga memiliki beberapa keunggulan seperti mudah serta biaya perhitungan kecil, dapat menangani data missing, memiliki akurasi dan kecepatan yang tinggi saat diaplikasikan ke dalam database dengan data yang besar. Metode *naive bayes* terbukti dapat memberikan hasil yang cukup memuaskan ketika digunakan untuk klasifikasi teks. Salah satu metode dari *naive bayes* yang sering digunakan dalam klasifikasi teks adalah *multinomial naive bayes*. Metode *Multinomial Naive Bayes* dipilih karena memiliki tingkat akurasi yang tinggi dengan perhitungan sederhana. Metode *Multinomial Naive Bayes* sering digunakan dalam penelitian tentang klasifikasi teks karena kesederhanaan dan efektivitasnya yang menggunakan ide dasar probabilitas gabungan dari kata-kata dan kategori untuk memperkirakan probabilitas kategori pada suatu dokumen. Berdasarkan berbagai penelitian yang telah dilakukan, disimpulkan bahwa metode *multinomial naive bayes* serta metode TF-IDF memiliki hasil yang baik ketika digunakan untuk menganalisis sentimen.

Deep learning merupakan salah satu cabang dari *machine learning* yang algoritmanya memiliki abstraksi tingkat tinggi pada sekumpulan data. *Deep learning* adalah model yang memanfaatkan *Artificial Neural Network* atau jaringan saraf tiruan, dimana *Artificial Neural Network* dibuat mirip dengan otak manusia. Teknik *deep learning* adalah bagian dari *neural network* yang terkenal struktur *multilayer*, banyak dipakai karena dapat menangani banyak masalah sekaligus dan memberi solusi yang unik.

Salah satu penelitian yang menggunakan model *deep learning* ialah penelitian yang dilakukan oleh Alan Tusa Bagus dkk (Alan, 2021) dengan judul "Klasifikasi Emosi Pada teks dengan menggunakan Metode *Deep Learning*". Model *deep learning* yang digunakan pada penelitian ini ialah *BERT* dan mendapatkan hasil akurasi sebesar 89.2%. *BERT* merupakan teknik berbasis jaringan saraf untuk *pre-training natural language*. Cara kerja *BERT* adalah mampu melatih model Bahasa berdasarkan seluruh rangkaian kata dalam kalimat.

BERT dapat melakukan analisis sentimen dengan menggunakan kemampuan pembelajaran representasi yang kuat untuk menangkap fitur semantik dan sintaksis sebuah teks. Saat ini, model transformator digunakan untuk sebagian besar tugas analisis sentimen dalan *Natural Language Processing* (NLP). Alasannya karena model transformer memiliki mekanisme yang akan bekerja lebih baik daripada model pembelajaran mesin lainnya. Oleh karena itu upaya yang dapat dilakukan yaitu dengan menganalisis sentimen menggunakan metode *Multilingual BERT*, dimana *Multilingual BERT* bekerja dengan sangat baik pada tugas transfer lintas bahasa, lebih unggul dari penyisipan kata non-kontekstual statis. Model *machine learning* memiliki kelemahan dalam memilih metode *ekstraksi* fitur dan menemukan fitur yang lebih baik. Model tersebut juga memiliki kelemahan pada data yang berjumlah besar dapat mempengaruhi performa pada akurasi klasifikasi. *Deep learning* menunjukkan hasil yang bagus di bidang *speech recognition, vision* dan sebagainya sehingga banyak digunakan pada analisis sentimen untuk mendapatkan kinerja yang lebih baik.

Metode *Machine Learning* dan *Deep Learning* telah banyak dilakukan oleh peneliti sebelumnya. Namun, menganalisis sentimen dalam domain sosial media menghadirkan beberapa tantangan. Gaya linguistik memiliki jangkauan yang luas, representasi sentimen yang terjadi pada jutaan orang menjadi sulit untuk menyimpulkan keadaan sentimen seseorang secara konkret. Pada saat yang sama, membuat fitur standar melalui komentar-komentar pelanggan yang banyak juga sulit, oleh karena itu diperlukan metode yang tepat untuk menganalisis sentimen pada teks.

Untuk mengatasi permasalahan yang telah disebutkan sebelumnya, penelitian ini melakukan perbandingan antara metode yang ada di *machine learning* dan metode yang ada di *deep learning* sehingga didapatkan pemahaman mendalam terhadap metode yang cocok untuk analisis sentimen pada teks komentar pelanggan shopee. Model *machine learning* yang digunakan yaitu *Logistic Regression*, *Naive Bayes* dan *Multinomial Naive Bayes*. Sedangkan model *deep learning* yang digunakan pada penelitian ini ialah *BERT* dan *multilingual BERT*.

I.2. Rumusan Masalah

Deskripsi latar belakang menunjukkan bahwa pernyataan masalah untuk penelitian tugas akhir ini adalah :

1. Apakah model *machine learning* lebih baik dari model *deep learning* dalam analisis sentimen pelanggan shopee?
2. Bagaimana nilai prediksi akurasi dari model *machine learning* dan *deep learning* dalam analisis sentimen pelanggan shopee?

I.3 Tujuan Penelitian

Tujuan dari pengerjaan penelitian Tugas Akhir ini adalah sebagai berikut:

1. Membandingkan model *machine learning* dan model *deep learning* dalam analisis sentimen pelanggan shopee.
2. Mengetahui nilai prediksi akurasi dari model *machine learning* dan model *deep learning* dalam analisis sentimen pelanggan shopee.

I.4. Batasan Masalah

Berdasarkan permasalahan yang disebutkan di atas, batasan masalah pada penelitian tugas akhir ini adalah sebagai berikut:

1. Sumber data yang digunakan adalah data komentar pelanggan mengenai penilaian penggunaan aplikasi shopee yang diperoleh dari github.com/ekaminartimuhliss/analisis_sentimen_shopee.
2. Metode *machine learning* yang digunakan adalah *Logistic Regression, Naive Bayes* dan *Multinomial Naive Bayes*. Metode *deep learning* yang digunakan ialah *BERT (Bidirectional Encoder Representations from Transformers)* dan *Multilingual BERT*.

I.5. Manfaat Penelitian

Berdasarkan latar belakang, rumusan masalah dan tujuan penelitian yang telah di uraikan, maka manfaat yang diperoleh dari tugas akhir ini antara lain:

1. Dapat digunakan sebagai referensi baru untuk performa dari tiap model dalam melakukan analisis sentimen.
2. Dapat dimanfaatkan sebagai penilaian terhadap kualitas aplikasi shopee.

BAB II
KAJIAN KEPUSTAKAAN

II.2 Penelitian terdahulu

Penelitian mengenai perbandingan model *machine learning* dan *deep learning* sudah sering dilakukan pada berbagai macam bidang dan berbagai metode. Rona Nisa Sofia Amriza Dkk telah melakukan penelitian yang berjudul "Komparasi Model *Machine Learning* dan *Deep Learning* untuk Deteksi Emosi pada *Text* di Sosial Media". Metode pada model *machine learning* yang digunakan dalam penelitian ini adalah *Naïve Bayes, Random Forest, SVM, Gradient Boosting* dan *Logistic Regression*. Sedangkan metode pada model *deep learning* yang digunakan antara lain *LSTM, CNN, MLP, GRU* dan *RNN*. Pada penelitian ini diperoleh hasil bahwa *Deep learning* memiliki performa yang lebih baik dari *machine learning*, hal tersebut dapat dilihat dari nilai akurasi dari *deep learning* yang melebihi nilai akurasi dari *machine learning* (Rona, 2021).

M. Azmi Aris (Azmi, 2020) telah melakukan penelitian serupa dengan judul "Perbandingan Kinerja Model pada metode *Deep Learning* dan Model pada metode *non-Deep Learning* pada Analisis Sentimen Tweet Berbahasa Indonesia". Pada penelitian ini dilakukan perbandingan metode *CNN* dan *LSTM* dengan metode *Naive Bayes* dari sisi akurasi dan waktu komputasi untuk mengetahui apakah pengorbanan komputasi yang benar akan mendapatkan akurasi yang jauh lebih baik. Penelitian ini menghasilkan akurasi sebesar 60.92% untuk metode *LSTM*, 59.29% untuk metode *CNN*, 57.59% untuk metode *Naive Bayes*. Waktu komputasi rata-rata dalam 1 fold yang didapat oleh ketiga metode tersebut adalah 57 milidetik untuk *Naive Bayes*, 110 detik 500 milidetik untuk *CNN*, dan 293 detik 730 milidetik untuk *LSTM*. Setelah dilakukan pengujian hipotesis dari sisi akurasi dan waktu komputasi dapat disimpulkan bahwa kinerja metode *non-Deep Learning* yaitu *Naive Bayes* lebih baik daripada metode *Deep Learning* yakni *LSTM* dan *CNN*.

Selanjutnya adalah penelitian yang dilakukan oleh Yudicy Amelia Dkk (Yudicy, 2020) dengan judul "Perbandingan Metode *Deep Learning* dan *Machine Learning* untuk Klasifikasi (Uji Coba Pada Data Penyakit Kanker Payudara)". Tujuan dari penelitian ini adalah mendapatkan perbandingan metode yang paling akurat terhadap pengklasifikasian penyakit kanker payudara serta mencocokkan hasil klasifikasi kedua metode dengan pernyataan klasifikasi yang sudah ada sehingga didapatkan nilai perbandingan metode yang paling akurat. Dalam kasus klasifikasi penyakit kanker payudara, akurasi metode *machine learning* lebih kecil dibandingkan dengan akurasi *deep learning* yaitu 97.08% untuk *machine learning* dan 98.54% untuk *deep learning* yang menandakan bahwa *deep learning* lebih baik dibandingkan *machine learning*.

M. R. Adrian Dkk (Adrian, 2021) telah menggunakan penelitian yang berjudul "Perbandingan Metode Klasifikasi *Random Forest* dan *SVM* pada Analisis Sentimen PSBB". Dari objek tweet dan model algoritma yang diuji, dapat disimpulkan bahwa beragamnya bahasa pada twitter mampu menurunkan kemampuan model untuk memprediksi suatu sentimen pada tweet terkait PSBB. *SVM* dianggap lebih baik karena mampu mengenali tweet dengan label positif. Dari tes yang dilakukan untuk masing-masing model didapatkan model *Random Forest* memiliki akurasi yang lebih tinggi dibandingkan *Support Vector Machine* yaitu 0.58% untuk *Random Forest* dan 0.56% untuk *SVM*.

Nuli Giarsyani Dkk (Nuli, 2020) juga melakukan penelitian dengan judul "Komparasi Algoritma *Machine Learning* dan *Deep Learning* untuk *Named Entity Recognition* : Studi Kasus Data Kebencanaan". Penelitian ini bertujuan untuk melakukan *Named Entity Recognition* guna mengidentifikasi dan mengklasifikasi kata pada tweet yang memuat informasi bencana ke dalam entitas-entitas yang telah ditentukan. Entitas yang diidentifikasi yaitu jenis bencana, lokasi, waktu, magnitude dan lainnya. Adapun algoritma klasifikasi yang digunakan adalah *Machine Learning* dan *Deep Learning*. Algoritma *Deep Learning* yang digunakan yaitu *Long Short-Term Memory*, *Gated Recurrent Units*, dan *Convolutional Neural Network*.

Sedangkan algoritma *Machine Learning* yang digunakan yaitu *Naïve Bayes, Decision Tree, Support Vector Machine* dan *Random Forest*. Berdasarkan hasil eksperimen, *Deep Learning* memperoleh akurasi yang lebih unggul dari *Machine Learning*. Hal tersebut dilihat dari perolehan nilai *accuracy* terbaik *Deep Learning* dihasilkan dari algoritma *Gated Recurrent Units* dan *Long Short-Term Memory* dengan nilai 0.995%. Sedangkan perolehan *accuracy* terbaik *Machine Learning* dihasilkan dari algoritma *Random Forest* sebesar 0.98%

Selanjutnya penelitian dilakukan oleh Muhammad Naufal humam (Humam, 2021) dengan judul "Perbandingan Kinerja Algoritma *Convolutional Neural Network* dan *Naive Bayes* pada Analisis Sentimen Performa *Manchester United* di Twitter". Berdasarkan hasil penelitian, pada dataset berbahasa inggris *CNN* menunjukkan kinerja terbaik dengan nilai akurasi 94% dibandingkan dengan nilai *naive bayes* 79%. Sehingga dapat disimpulkan bahwa pada penelitian ini *CNN* lebih unggul dibandingkan *Naive Bayes*.

Selanjutnya Alfando Dkk melakukan penelitian berjudul "Klasifikasi Teks Berita Berbahasa Indonesia Menggunakan *Machine Learning* dan *Deep Learning:* Studi Literatur" (Alfando, 2023). Beberapa algoritma *Machine Learning* dan *Deep Learning* yang digunakan dalam penelitian ini adalah *K-Nearest Neighbor (KNN), Multinominal Naive Bayes, Long Short-Term Memory, Multi Layer Perceptron, Support Vector Machine (SVM)* dan lainnya. Hasil penelitian menunjukkan bahwa *SVM* memberikan kinerja yang terbaik diantara algoritma *Machine Learning* lainnya dengan nilai *accuracy* sebesar 94,24%. Sedangkan pada algoritma *Deep Learning*, *Long Short-Term Memory* mendapatkan nilai *accuracy* sebesar 95%.

Merinda Lestandy Dkk telah melakukan penelitian yang berjudul "Analisis Sentimen Tweet Vaksin COVID-19 Menggunakan *Recurrent Neural Network* dan *Naive Bayes*" (Merinda, 2021). Penelitian ini mengkaji kinerja *RNN* dan *Naïve Bayes* dengan menambahkan teknik *TF-IDF*. Hasil pengujian menunjukkan *RNN (TF-IDF)* memiliki akurasi lebih besar yaitu 97,77% dibandingkan *Naïve Bayes (TF-IDF)* sebesar 80%.

Tabel II.1. Perbandingan Penelitian Terdahulu

No	Peneliti	Metode	Tujuan Penelitian	Hasil
1	(Rona, 2021)	M*achine learning* dan D*eep learning*	Komparasi Metode *Machine Learning* dan metode *Deep Learning* untuk Deteksi Emosi pada Text di Sosial Media	*Deep learning* memiliki performa yang lebih baik dari *machine learning*
2	(Azmi, 2020)	*Deep Learning* dan *non-Deep Learning*	Perbandingan Kinerja Metode *Deep Learning* dengan Metode *Non-Deep Learning* pada Analisis Sentimen Tweet Bahasa Indonesia	Metode *non-Deep Learning* lebih baik daripada metode *Deep Learning*
3	(Yudicy, 2020)	*Machine learning* dan *deep learning*	Perbandingan Metode *Deep Learning* Dan metode *Machine Learning* Untuk Klasifikasi	Akurasi *machine learning* lebih kecil dibandingkan dengan akurasi *deep learning*
4	(Adrian, 2021)	*Machine Learning* dan *Deep learning*	Perbandingan Metode *Random Forest* dan *SVM* Pada Analisis Sentimen PSBB	Model *Random Forest* memiliki akurasi yang lebih tinggi
5	(Nuli, 2020)	- *Deep Learning* - *Machine Learning*	Komparasi Algoritma *Machine Learning* Dan *Deep Learning* Untuk *Named Entity Recognition*	*Deep Learning* memperoleh akurasi yang lebih unggul dari *Machine Learning*

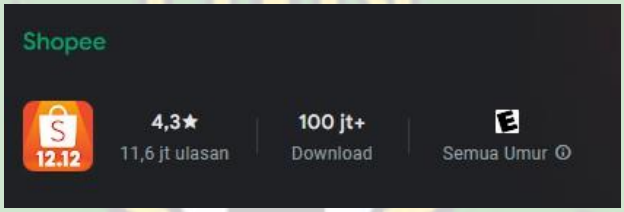
No	Peneliti	Metode	Tujuan Penelitian	Hasil
6	(Humam, 2021)	*CNN* dan *Naive Bayes*	Perbandingan Kinerja Algoritma *CNN* Dan *Naïve Bayes* Pada Analisis Sentimen Performa *Manchester United* Di Twitter	*CNN* menunjukkan kinerja terbaik dibandingkan dengan nilai *Naive Bayes*
7	(Alfando, 2023)	*Machine learning* dan *deep learning*	Klasifikasi Teks Berita Berbahasa Indonesia Menggunakan *Machine Learning* Dan *Deep Learning*	*Long Short-Term Memory* memberikan kinerja yang terbaik
8	(Merinda, 2021)	*Machine learning* dan *deep learning*	Analisis Sentimen Tweet Vaksin COVID-19 Menggunakan *Recurrent Neural Network* dan *Naïve Bayes*	Hasil pengujian menunjukkan *RNN* memiliki akurasi lebih besar dibandingkan *Naïve Bayes*

II.2 Teori Pendukung

II.2.1 Shopee

Shopee adalah platform perdagangan elektronik yang berkantor pusat di Singapura di bawah SEA Group yang didirikan pada tahun 2009 oleh Forest Li. Shopee pertama kali diluncurkan di Singapura pada tahun 2015 dan memperluas jangkauannya ke Malaysia, Thailand, Taiwan, Indonesia, Vietnam dan Filipina. Shopee merupakan anak perusahaan dari Ganera yang berbasis di Singapura. Shopee telah hadir di beberapa negara kawasan Asia Tenggara seperti Singapura, Malaysia, Vietnam, Thailand, Filiphina dan Indonesia.

Sasaran pengguna Shopee adalah kalangan muda yang saat ini terbiasa melakukan kegiatan dengan bantuan gadget termasuk kegiatan berbelanja. Saat ini angka unduhan shopee telah mencapai 100 Juta unduhan di *Google Play Store*.



Gambar II.1. Jumlah Unduhan Shopee di *Google Play Store*
(Sumber: Goog le Play Store, 2022)

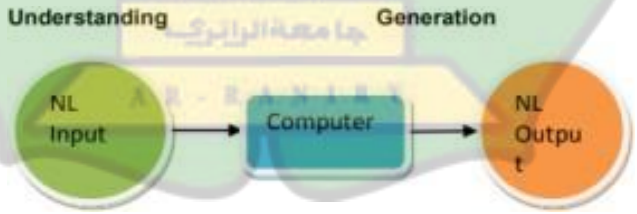
II.2.2 Analisis Sentimen

Analisis sentimen adalah salah satu cabang penelitian dari *text mining* yang bertujuan untuk mengklasifikasikan opini seseorang yang berupa dokumen text berdasarkan sentimen. Analisis sentimen merupakan salah satu bentuk dari pengaplikasian *text mining* yang digunakan untuk mengetahui opini dari sekumpulan data tekstual mengenai peristiwa atau topik tertentu (Rahmatullah, 2021). Fungsi utama dari analisis sentimen adalah untuk menggali informasi, mencari makna, serta opini pengguna dari data posting, baik berupa caption, post, komentar yang diunggah.

Dalam bisnis, penggunaan analisis sentimen sangat dibutuhkan dalam *role business intelligence* untuk mengetahui pendapat pelanggan mengenai produk atau jasa yang ditawarkan sehingga dapat digunakan sebagai pertimbangan dalam mengembangkan produk. Dalam sistem pemerintahan, analisis sentimen dapat membantu pemerintah untuk mengetahui opini yang dikeluarkan masyarakat mengenai kebijakan yang dikeluarkan oleh pemerintah. Proses penggalian sentimen atau opini publik terutama pada media sosial seperti Facebook, Twitter atau kolom komentar memiliki tantangan tersendiri. Ekstraksi data tekstual menjadi masalah yang harus dihadapi dalam menggali sentimen karena data yang bersifat dinamis, terdapat penggunaan kata yang tidak baku, juga format data yang bervariasi.

II.2.3 *Natural Language Processing* (NLP)

Bahasa alami atau *Natural Language* adalah bahasa yang digunakan oleh orang-orang. Pemrosesan Bahasa Alami menggabungkan semua yang dibutuhkan komputer untuk memahami bahasa alami dan juga untuk menghasilkan bahasa alami. NLP adalah cabang dari kecerdasan buatan dan linguistik yang berfokus untuk membuat komputer memahami pernyataan atau kata-kata yang ditulis dalam bahasa manusia. Bahasa alami juga dikenal sebagai bahasa biasa, diucapkan atau ditulis oleh manusia untuk komunikasi umum (Martinez, 2020).



Gambar II.2 Proses NLP
(Sumber: Alan, 2021)

Pada gambar diatas dapat dilihat bahwa ada dua proses yang terjadi yaitu:

1. *Natural Language Understanding* (NLU), bertugas untuk memahami bahwa didalam input adalah bahasa alami atau *Natural Language*.
2. *Natural Language Generation* (NLG), bertugas dalam sub generasi pemrosesan bahasa alami(Martinez, 2020).

II.2.4 *Machine Learning*

Machine learning merupakan ilmu atau studi yang mempelajari tentang algoritma dan model statistik yang digunakan oleh sistem komputer untuk melakukan *task* tertentu (Rozzi, 2020). *Machine learning* merupakan kecerdasan buatan yang mempelajari bagaimana membuat data. *Machine learning* atau disingkat ML ini dibutuhkan untuk menerapkan teknik yang cepat dan kuat dalam menemukan masalah baru. *Machine learning* adalah bidang keilmuan yang mempelajari bagaimana membuat program yang dapat menghasilkan pengetahuan baru dari pengetahuan yang sudah ada di luar pengetahuan yang "diprogram" secara langsung pada program.

II.2.5 *Deep Learning*

Deep learning adalah salah satu cabang dari *machine learning* yang algoritmanya memiliki abstrasksi tingkat tinggi pada sekumpulan data. *Deep learning* disebut juga dengan *Representations Learning. Deep learning* juga dapat melakukan model komputasi yang memilki beberapa layer pengolahan untuk mempelajari representasi data. *Deep learning* adalah model yang memanfaatkan *Artificial Neural Network* atau jaringan saraf tiruan, dimana *Artificial Neural Network* dibuat mirip dengan otak manusia Karena model ini dilatih dengan menggunakan dataset berlabel dengan jumlah yang banyak dan *Neural Network* dapat melakukan penyelesaian masalah secara akurat dan otomatis.

II.2.6 Algoritma *Logistic Regression*

Logistic regression atau Regresi Logistik merupakan jenis *supervised learning* yang digunakan untuk membuat sebuah model prediksi yang sama halnya dengan regresi linear. Bedanya ada pada nilai variabelnya yang bernilai ya/tidak, benar/salah, ataupun dalam bentuk bilangan biner 0/1. *Logistic Regression* sangat cocok digunakan untuk memprediksi ketika variabel dipenden atau output suatu data bersifat biner. *Logistic Regression* telah banyak digunakan dalam berbagai data masalah sentimen analisis, seperti prediksi hasil positif, negatif, atau netral (Aloysius, 2021). Berikut merupakan kelebihan algoritma logistic regression:

1) *Logistic Regression* lebih mudah diterapkan, diinterpretasikan dan sangat efisien untuk dilatih.
2) *Logistic regression* tidak membuat asumsi tentang distribusi kelas dalam ruang fitur.
3) Algoritma ini dapat dengan mudah diperluas ke beberapa kelas (*regresi multinomial*) dan pandangan probabilistik alami dari prediksi kelas.
4) *Logistic regression* tidak hanya memberikan ukuran seberapa tepat suatu prediktor, tetapi juga arah asosiasinya (positif/negatif).
5) Algoritma ini sangat cepat dalam mengklasifikasikan catatan yang tidak diketahui.
6) Akurasi yang baik untuk banyak kumpulan data sederhana dan berkinerja baik ketika kumpulan data dapat dipisahkan secara linier.
7) *Logistic regression* dapat menafsirkan koefisien model sebagai indikator pentingnya fitur.
8) *Logistic regression* tidak terlalu cenderung melakukan overfitting namun dapat melakukan overfitting pada kumpulan data berdimensi tinggi.

II.2.7 *Naive Bayes*

Naïve Bayes adalah suatu proses klasifikasi dengan probabilitas sederhana yang mengacu pada teori *bayes*. Teori *bayes* menyatakan bahwa kemungkinan terjadinya suatu peristiwa sama dengan *probabilitas intrinsik* dikalikan *probabilitas* bahwa hal serupa akan terjadi lagi di masa depan. *Naïve Bayes* adalah algoritma pembelajaran probabilitas yang berasal dari teori keputusan *bayesian* (Yasin, 2023).

Proses *training* berguna untuk membentuk pengetahuan berupa nilai probabilitas kata. Perlu diketahui bahwa pada proses *training*, tidak dijalankan modul klasifikasi, tetapi hanya menghasilkan dokumen yang mengandung kata untuk mengkarakteristik suatu kelas (Haryono, 2020). Klasifikasi algoritma *naïve bayes* akan dilakukan penentuan nilai probalitas dengan cara melakukan perhitungan jumlah kelas variabel. Kelebihan *naive bayes* ialah:

1) Menangani kuantitatif dan data diskrit.
2) Hanya memerlukan sejumlah kecil data pelatihan untuk mengestimasi parameter yang dibutuhkan untuk klasifikasi.
3) Menangani nilai yang hilang dengan mengabaikan instansi selama perhitungan estimasi peluang.
4) Cepat dan efisiensi ruang.
5) Kokoh terhadap atribut yang tidak relevan.
6) Kesederhanaan dalam proses komputerisasi dan nilai akurasi yang tinggi.

II.2.8 *Multinomial Naive Bayes*

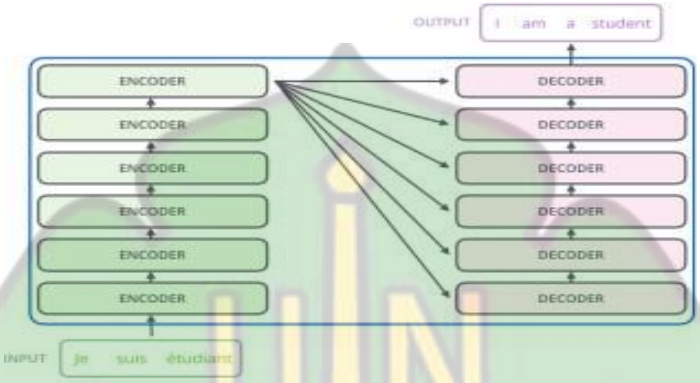
Multinomial Naïve Bayes merupakan variasi dari *Naïve Bayes* yang digunakan untuk mengklasifikasi dokumen atau teks ke dalam beberapa kategori atau kelas berdasarkan fitur-fitur yang ada dalam dokumen. Metode ini melakukan klasifikasi dengan pembelajaran *supervised* menggunakan model probabilistik. *Multinomial Naïve Bayes* dipengaruhi oleh serangkaian *term*, dengan kata lain jumlah *term* diperhitungkan. Peluang antara *term* satu dengan yang lain tidak bergantung. Model *Multinomial Naïve Bayes* memperhitungkan frekuensi setiap kata yang muncul pada dokumen (Dewi, 2021).

Metode *Multinomial Naïve Bayes* memanfaatkan teorema probabilitas yaitu teorema bayes dan fungsionalitas data mining yaitu klasifikasi *naïve bayesian. Multinomial Naïve Bayes* adalah salah satu metode bayes yang digunakan dengan memperhitungkan frekuensi masing-masing kemunculan data dalam sebuah dokumen dan probabilitas. Kelebihan *Multinomial Naïve Bayes* diantaranya adalah tingkat akurasi yang tinggi, mudah diimplementasikan, waktu komputasi yang rendah serta error rate yang minimum (Sriyano, 2021). Kelebihan Multinomial Naive Bayes ialah:

1) Penerapannya mudah karena haya perlu menghitung probabilitas.
2) Naive bayes dapat digunakan pada data kontinu dan diskrit.
3) Sederhana dan dapat digunakan untuk memprediksi aplikasi real-time.
4) Sangat skalabel dan dapat dengan mudah menangani kumpulan data besar.

II.2.9 *Transformers*

Transformers adalah model transmisi pertama yang sepenuhnya bergantung pada *Self-attention* untuk menghitung representasi input dan outputnya. Model ini dirancang untuk memproses tipe *data sequence* seperti bahasa alami untuk melakukan tugas seperti terjemah bahasa dan peringkasan teks (Rendragraha, 2021).



Gambar II.3 Arsitektur *Transformers*
(Sumber: Alfando, 2023)

BERT menggunakaan arsitektur neural network yang disebut transformers yang teerlihat pada gambar 2.2 bahwa *BERT* hanya menerima input berupa *vector* dengan teknik *word embeddings*. Proses *embeddings* setiap token dalam urutan input direpresentasikan melalui proses ini. *Stack encoder* bertanggung jawab untuk meng-code urutan input, sementara *stack decoder* bertanggung jawab untuk menghasilkan urutan *output*. Output dari *stack decoder* di proyeksikan menjadi *vector probabilitas* atas kosakata. *Distribusi probabilitas* ini digunakan untuk menghasilkan urutan output token per token (Rahmatullah, 2021).

II.2.10 *BERT*

Bidirectional Encoder Representations from Transformers (BERT) merupakan model representasi bahasa terlatih yang pertama kali di populerkan oleh Google pada 11 Oktober 2018. Model ini berbeda dengan model sebelumnya yang bersifat satu arah, model ini dibuat sebagai *pre-trained model* atau model yang sudah dilatih (Tandijaya, 2021). Metode *BERT* memiliki beberapa variasi berdasarkan ukuran *Hyperparameter* yang berbeda.

Transformer dapat belajar dan mengubah pemahaman yang diperoleh dari mekanisme *self-attention*. Mekanisme *self-attention* adalah cara transformator memodifikasi kata terkait dan diubah oleh mekanisme tersebut. *Transformer* terdiri dari dua mekanisme *encoder* dan *decoder*. *Encoder* digunakan untuk membaca data input teks. *Encoder* terdiri dari tumpukan N = 6 lapisan identic. Dengan lapisan *self-attention*, *encoder* dapat membantu *node* yang tidak hanya fokus pada kata yang di visualisasikan. *Decoder* berfungsi untuk menghasilkan urutan keluaran yang diprediksi. *Decoder* juga mencakup tumpukan N = 6 lapisan yang dapat di identifikasi. Lapisan *self-attention* di *decoder* memungkinkan setiap posisi di *decoder* untuk menangani semua posisi sebelumnya dan saat ini. Kelebihan Metode *BERT* ialah:

1) *BERT* dapat memanfaatkan sejumlah besar data tak berlabel untuk melakukan pra-pelatihan modelnya, sehingga mengurangi kebutuhan akan data khusus tugas dan meningkatkan kemampuan generalisasi model.
2) Konteks dua arah *BERT* memungkinkannya menangani ekspresi ambigu dan kompleks dengan lebih efektif daripada model satu arah.
3) *BERT* beradaptasi dengan berbagai tugas dengan menyempurnakan modelnya pada kumpulan data tertentu sehingga meningkatkan fleksibilitasnya.

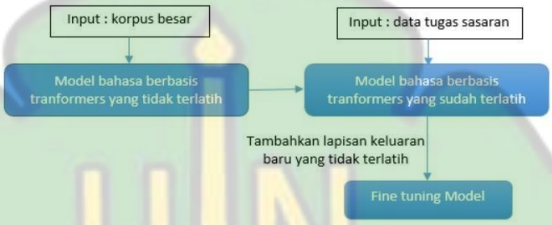
II.2.11 Multilingual BERT

Multilingual BERT adalah versi *BERT* tetapi dengan beberapa bahasa yang berbeda. Ini adalah salah satu model *BERT* terlatih. Namun, yang membedakan *Multilingual BERT* dan model lainnya adalah model ini dilatih dalah 104 bahasa, termasuk bahasa Indonesia. Model ini sangat berguna untuk memecahkan masalah multibahasa. Kelebihan *Multilingual BERT* ialah:

1) *BERT* menganalisis teks berbahasa inggris sedangkan *multilingual BERT* menganalisis teks dengan banyak bahasa termasuk bahasa indonesia.
2) Metode ini mampu mempelajari koneksi antara kata dan frasa delam kalimat sehingga menghasilkan terjemahan yang lebih akurat dan alami.

II.2.12 *Pre-Training* dan *Fine tunning*

BERT dan *Multilingual BERT* adalah *pre-trained* model artinya model tersebut sudah dibuat dan dilatih. Namun, model ini tidak sepenuhnya akurat dan siap untuk keperluan penelitian sehingga dibutuhkan proses *finetune and train model*. *Fine tuning* adalah proses memperbaharui semua parameter dalam model keperluan task yang baru. Sedangkan *train model* adalah proses belajar model dalam menentukan nilai *weights* dan *bias* yang terbaik dari label.



Gambar II.4 Tahapan *Pre-train* dan *Finetunning*
(Sumber: Azmi, 2020)

Model bahasa terlatih (*Pre-Train Model*) dikumpulkan dengan dataset yang besar. Model yang dihasilkan mampu memahami model bahasa yang diajarkan. *Finetunning* sangat mudah dilakukan karena mekanisme *self-attention* pada *Transformers* membuat *BERT* dapat berbagi tugas, baik pada kalimat tunggal atau kalimat berpasangan dengan menukar masukan dan keluaran yang sesuai.

II.2.13 *Confusion Matrix*

Confusion Matrix adalah suatu metode yang biasanya digunakan untuk melakukan perhitungan akurasi pada konsep *data mining*. *Confusion Matrix* digambarkan dengan tabel yang dinyatakan jumlah data uji yang benar diklasifikasikan dan jumlah data uji yang salah diklasifikasikan (Fadly, 2019). *Confusion matrix* merupakan hasil dari proses analisis berupa visualisasi data yang benar atau salah di prediksi. Pada penelitian ini berisi hasil pelabelan data komentar pelanggan shopee.

Nilai akurasi dihitung dari perhitungan jumlah prediksi benar yang sesuai / *True Positives* (TP) ditambah jumlah prediksi benar tidak sesuai / *True Negatives* (TN) dibandingkan dengan jumlah prediksi benar yang sesuai / *True Positives* (TP), jumlah prediksi benar tidak sesuai / *True Negatives* (TN), jumlah prediksi salah yang sesuai / *False Positives* (FP) dan jumlah prediksi salah tidak sesuai / *False Negatives* (FN).

II.3 Tools

Dalam Penelitian ini tools yang digunakan adalah bahasa pemograman *Phyton* dan *Google Colab*. Tools ini dapat digunakan secara online dan tidak berbayar pada *Website Google Colab*.

II.3.1 *Python*

Python merupakan salah satu bahasa pemrograman yang banyak digunakan oleh perusahaan besar maupun para developer untuk mengembangkan berbagai macam aplikasi berbasis dekstop. *Python* adalah bahasa pemrograman yang banyak digunakan dalam aplikasi web, pengembangan perangkat lunak, ilmu data dan machine learning.

Developer menggunakan *pyhton* karena efisien dan mudah dipelajari serta dapat dijalankan di berbagai platform. *Python* dapat diunduh secara gratis, terintegrasi baik dengan tipe sistem dan meningkatkan kecepatan pengembangan (Kadarina, 2019).

II.3.2 Google Colaboratory

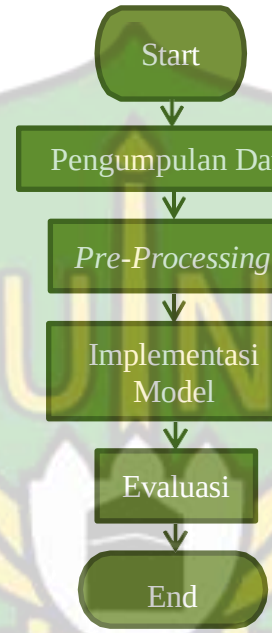
Google Colaboratory atau *Google Colab* ialah salah satu produk dari *Google Internal Research. Google Colab* merupakan salah satu *tools Google* yang membantu user dalam proses pemrograman dan pengolahan data. Ia memungkinkan penggunanya untuk menjalankan kode *phyton* tanpa perlu melakukan proses instalasi dan setup lainnya.

BAB III

METODOLOGI PENELITIAN

III.1 Tahapan Penelitian

Tahapan penelitian menjelaskan bagaimana alur penelitian yang digunakan dalam penelitian ini. *Flowchart* tahapan dapat dilihat pada Gambar III.1.



Gambar III.1. *Flowchart* Tahapan Penelitian

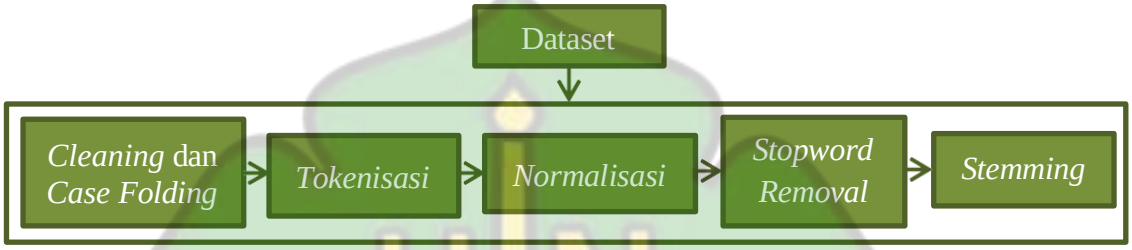
III.2 Metode Pengumpulan Data

Penelitian ini menggunakan metode kuantitatif deskriptif dengan pendekatan Analisis Data Sekunder (ADS). Penelitian Kuantitatif adalah penelitian dengan memperoleh data berbentuk angka atau data kualitatif yang di angkakan. ADS merupakan suatu metode dengan memanfaatkan data sekunder sebagai sumber data utama.

Sumber data berasal dari data komentar pelanggan Shopee yang diperoleh dari github.com/ekaminartimuhliss/analisis_sentimen_shopee. Jumlah dataset yang digunakan sebanyak 6.002 komentar berbahasa indonesia. Data komentar pelanggan shopee tersebut sudah diberikan label yaitu 5450 komentar berlabel positif dan 552 komentar berlabel negatif.

III.3 *Pre-Processing*

Pre-Processing merupakan proses awal dalam pengolahan data masukan sehingga akan menghasilkan data dengan format yang sesuai dan siap diproses pada tahap selanjutnya. Tujuan dilakukannya *pre-processing* agar data yang digunakan lebih presisi dan memudahkan dalam menjalankan analisis sentimen. Tahapan *pre-processing* dapat dilihat dalam Gambar III.2.



Gambar III.2. Tahapan *Pre-Processing*

III.3.1 Cleaning dan *Case Folding*

Tahap *cleaning* adalah proses pembersihan teks dari kata yang tidak diperlukan. Kata yang dihilangkan ialah karakter, simbol, huruf tunggal, link URL, dan sebagainya. Tahap ini adalah proses mengubah teks menjadi bentuk standar sehingga data masukkan akan diubah menjadi huruf kecil.

III.3.2 Tokenization

Tokenisasi merupakan proses menyortir dan membagi frase menjadi kata-kata individu yang dikenal sebagai tokenisasi. Token bisa berupa kata-kata, karakter, atau bagian-bagian kata.

III.3.3 Normalisasi

Pada tahap ini akan dilakukan penyeragaman kata yang memiliki makna yang sama, dapat diakibatkan dari penulisan yang salah, penyingkatan kata, ataupun bahasa gaul. Kata-kata singkatan tersebut dimasukkan ke dalam suatu dokumen teks/*text* yang nanti akan dibaca oleh kode pemograman. Proses ini bertujuan untuk menyederhakan teks sehingga lebih mudah untuk dianalisis, diproses maupun dibandingkan dengan teks lainnya.

III.3.4 Stopword Removal

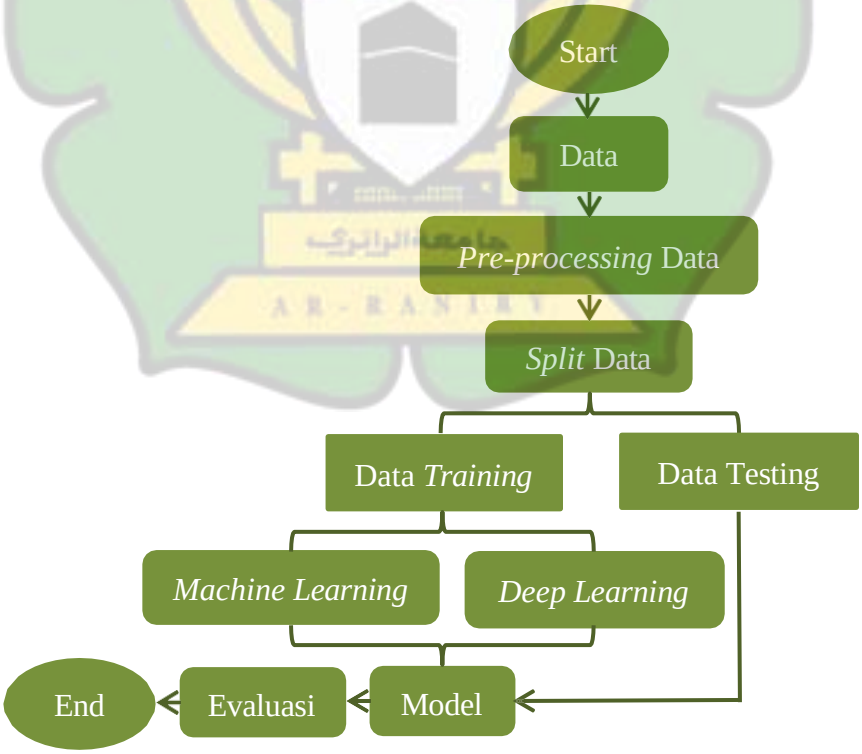
Stopword Removal berfungsi untuk menghapus kata yang tidak ada makna, seperti kata umum yang tidak memiliki nilai. Dalam tahap ini, kata yang tidak penting terhadap makna dokumen akan dihilangkan.

III.3.5 Stemming

Proses ini yaitu mengembalikan kata yang berimbuhan menjadi kata induk. Ini dilakukan untuk mengurangi kesalahan pada proses selanjutnya. Untuk stemming dalam bahasa indonesia menggunakan *library python sastrawi*. *Sastrawi* merupakan *library* yang dapat mengubah kata berimbuhan bahasa indonesia menjadi kata dasar.

III.4 Implementasi Metode

Tahap implementasi metode merupakan tahapan untuk mentransformasikan dan merealisasikan rumusan konsep penelitian yang telah dibangun sebelumnya. Alur sistem pada implementasi metode pada model *machine learning* dan *deep learning* dilihat pada gambar III.3.

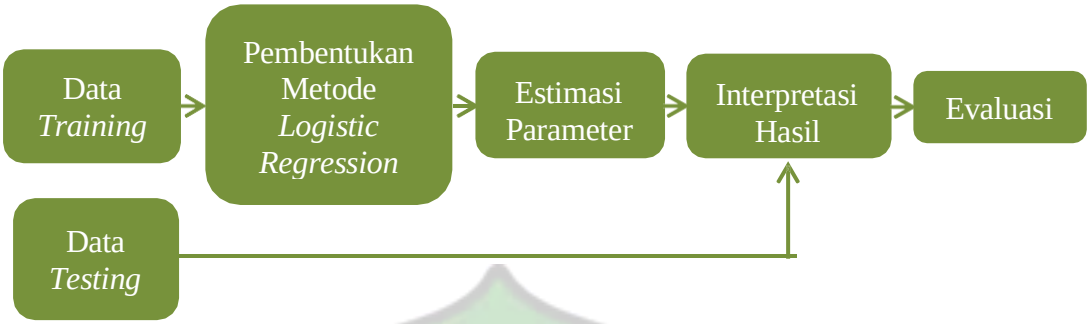


Gambar III.3 Alur Implementasi Metode

Keterangan dari alur diatas sebagai berikut:

1. Data diperoleh dari github milik Eka Minarti. Data ini di publish pada tanggal 26 Mei 2023 sebanyak 6.002 kalimat komentar. Data komentar yang akan dianalisis merupakan komentar berbahasa Indonesia.
2. *Pre-Processing* merupakan tahapan untuk membersihkan dataset yang akan diuji sehingga data menjadi lebih terstruktur. Dataset pada penelitian ini belum melewati proses *pre-processing* sehingga masih memiliki banyak kata tidak terstruktur dan tanda baca yang tidak perlu. Tujuan dilakukannya *pre-processing* ialah agar data dapat dibaca dengan baik ketika diuji sehingga hasil akhir yang diperoleh lebih akurat. Pada penelitian ini dibutuhkan beberapa tahapan *pre-processing* seperti tahap *Cleaning* dan *Case Folding*, Tokenisasi, Normalisasi, *Stopwords Removal*, dan *Stemming*.
3. *Split* data merupakan proses membagikan dataset menjadi dua kelompok yaitu data *testing* dan data *training*. Pembagian data menggunakan *library sklearn* dengan fungsi *train test split* untuk membagi data. Rasio pembagian data pada umumnya adalah 80:20 (80% adalah data *training* dan 20% adalah data *testing*). Pembagian data tersebut menggunakan metode *hold-out* yaitu 20% bagian dari dataset digunakan untuk data *testing* dan sisanya digunakan untuk data *training*.
4. Data *training* adalah sebuah data yang berguna untuk membangun dan melatih model pengklasifikasian. Pada penelitian ini data *training* yang digunakan sebanyak 4801 data.
5. Sedangkan data *testing* merupakan dataset yang akan diuji menggunakan pengklasifikasian yang sebelumnya sudah dibuat menggunakan data *training*. Pada penelitian ini data *testing* yang digunakan sebanyak 1201 data.
6. Metode pada model *machine learning* yang digunakan pada peneltian ini ialah *Logistic Regression, Naive Bayes* dan *Multinomial Naive bayes.* Sedangkan metode pada model *deep learning* yang akan digunakan ialah *BERT* dan *Multilingual BERT.* Adapun penjelasan mengenai pengklasifikasian metode tersebut sebagai berikut:

a. Metode *Logistic Regression*



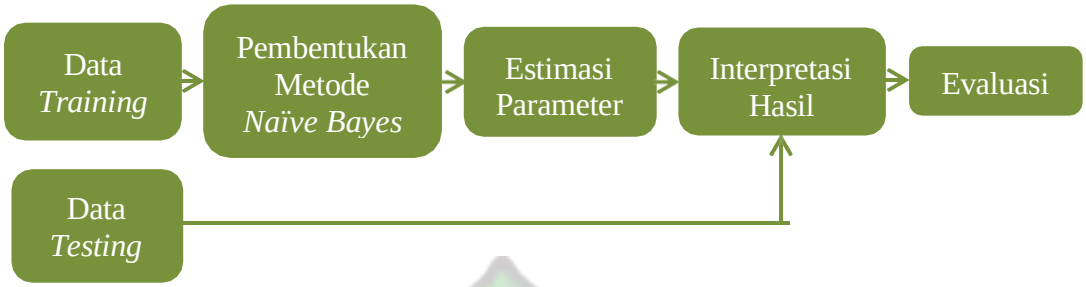
Gambar III.4 Proses Metode *Logistic Regression*

(1) Setelah dataset dibagi menjadi data *training* dan *testing*, tahap selanjutnya pada *Logistic Regression* ialah menentukan nilai probabilitas pada data.

(2) Tahap modeling data ialah menerapkan data ke dalam 2 kategori yang terdiri dari data x yang merupakan data teks dan data y yang merupakan data analisis sentimen. Tujuan di lakukan penerapan model ialah agar data tersebut mudah dibaca oleh algoritma pada saat analisis sentimen berlangsung. Pemodelan data tersebut dilakukan pada saat pengonversian data tipe objek ke dalam *Count Vectorizer* implementasi dari *library sklearn feature_extraction text* sehingga hasil dari implementasi tersebut menghasilkan data bilangan metriks. *Count Vectorizer* merupakan proses mengubah fitur teks menjadi bilangan metriks untuk ditugaskan sebagai penerjemah ke dalam bahasa mesin. Hasil implementasi tersebut akan dimasukkan kedalam data x.

(3) Selanjutnya penerapan *hyperparameter* yang digunakan sebagai pelatihan model untuk memperkirakan hasil prediksi parameter model Setelah dilakukan pemodelan dan penentuan parameter, selanjutnya data testing akan digunakan untuk mencari nilai akurasi dengan menggunakan metode *Logistic Regression*.

b. Metode *Naive Bayes*



Gambar III.5 Proses *Naïve Bayes*

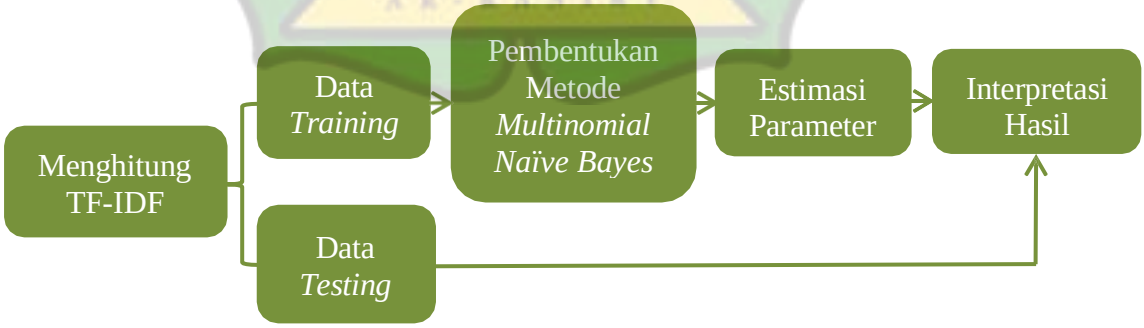
(1) Distribusi probabilitas *Naive Bayes* adalah pendekatan statistik yang didasarkan pada teorema *bayes*. Ini digunakan untuk memodelkan distribusi probabilitas dari suatu data berdasarkan data pelatihan.

(2) Aplikasi pada data teks, meskipun *naive bayes* dapat digunakan untuk klasifikasi teks, asumsi independen atribut sering tidak terpenuhi dalam data teks yang mengandung kata-kata atau fitur yang berkaitan erat satu sama lain.

(3) Setelah pembentukan model menggunakan metode *Naïve Bayes*, data akan dilakukan perhitungan parameter menggunakan *K-Fold Cross validation*. Pelatihan *K-Fold* akan dilakukan sebanyak 5-fold.

(4) Data *testing* digunakan untuk menghitung nilai akurasi dengan menggunakan *confusion matriks.*

c. Metode *Multinomial Naive Bayes*



Gambar III.6 Proses *Multinomial Naïve Bayes*

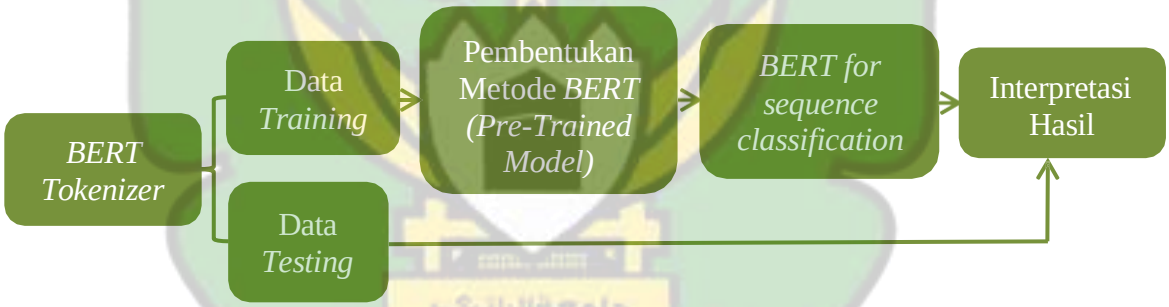
(1) Pendekatan khusus untuk data teks, *multinomial naive bayes* adalah variasi dari *naive bayes* yang secara khusus cocok untuk klasifikasi teks. Ini mempertimbangkan atribut sebagai jumlah kemunculan kata-kata dalam dokumen, oleh karena itu asumsi independensi digantikan dengan asumsi bahwa atribut ini mengikuti distribusi *multinomial*.

(2) Setelah dilakukan perhitungan TF-IDF, data dibagi menjadi data *training* dan *testing* dengan rasio 80:20. Kemudian dilakukan pemodelan dengan metode *multinomial naïve bayes*.

(3) Pada metode ini juga dilakukan perhitungan parameter dengan menggunakan *K-Fold cross validation* sebanyak *5-fold.*

(4) Setelah dilakukan pelatihan, dilakukan pengujian dengan menggunakan *confusion matriks*.

d. Metode *BERT*



Gambar III.7 Proses *BERT*

(1) *BERT* menggunakan arsitektur *neural network* yang disebut *transformers* sehingga *BERT* hanya menerima input berupa vector dengan teknik *word embeddings.*

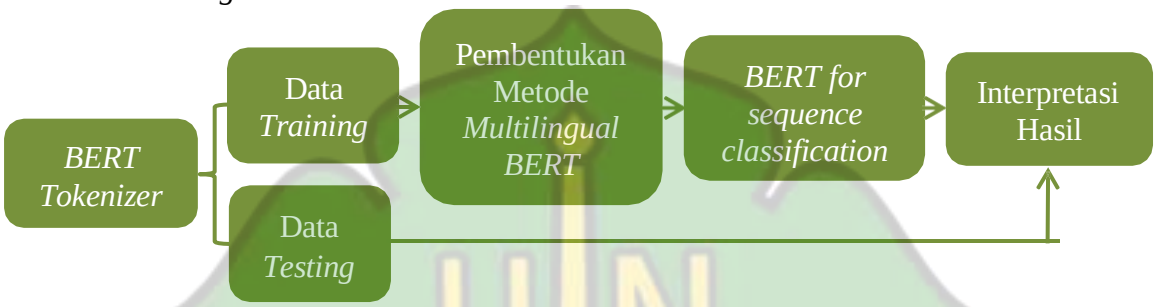
(2) Proses *embeddings* ialah merepresentasikan setiap token dalam urutan input. Karena arsitektur *transformers* tidak memiliki koneksi berulang, maka posisi dari setiap token dalam urutan input harus secara eksplisit direpresentasikan dengan menambahkan *vector positional encoding* kedalam *input embeddings*.

(3) Dataset akan diubah menjadi input yang dapat diterima oleh *BERT* yaitu dalam bentuk vector representasi kata menggunakan *Tokenizer*.

(4) Kemudian pada tahap *fine-tuning* dimana *pre-train* model *BERT* diadaptasi untuk melakukan analisis sentimen komentar shopee.

(5) Untuk model klasifikasi menggunakan *BERT For Sequence Classification* yang merupakan klasifikasi di atas 12 *layer BERT*. Kemudian akan dilakukan evaluasi menggunakan data *testing*.

e. *Multilingual BERT*



Gambar III.8 *Proses Multilingual BERT*

(1) Metode *Multilingual BERT* terdiri dari 13 *layer*. 12 lapisan tersembunyi *MBERT* dan 1 lapisan pengklasifikasi di bagian kelas.

(2) Selanjutnya input yang telah di sesuaikan kemudian diteruskan ke dalam jaringan *BERT* yaitu tumpukan 12 *layer Transformers Encoder* seperti yang tergambar pada gambar III.8

(3) *Library Transformers* memiliki kelas *BERTForSequenceClassification* yang didesain untuk tugas klasifikasi. *BERTForSequenceClassification* bekerja dengan cara memasukkan output untuk menghitung logits. Nilai logits yang dihasilkan digunakan untuk mendapat nilai prediksi menggunakan softmax.

7. Model adalah representasi matematis perhitungan probabilitas yang diperlukan untuk klasifikasi data. Data *Training* untuk melatih sistem model dengan tujuan agar model dapat mengenali pola dari data. Kemudian data *testing* digunakan untuk menguji performa akhir dari model.

8. Evaluasi dilakukan untuk mengetahui kelayakan dari model data yang sudah diuji dengan melihat hasil akurasi pada data *testing*. Evaluasi akan menghasilkan nilai *confusion matrix* pada tiap metode.

III.6 Evaluasi

Model *machine learning* dan *deep learning* akan di evaluasi menggunakan *confusion matrics* dengan melakukan perhitungan *accuracy*, *precission*, *recall*, dan *fl-score*. Rumus *confusion matrix* digambarkan dengan tabel III.1 sebagai berikut.

Tabel III.1. Tabel *Confusion Matrix*

Correct Classification	*Classified as*	
	Predicted "+/1"	*Predicted* "-/0"
Actual "+/1"	TP	FN
Actual "-/0"	FP	TN

Berdasarkan tabel *Confusion Matrix* diatas:

a. ***True Positives*(TP)** = Jumlah data positif yang diprediksi sebagai data positif
b. ***False Positives*(FP)** = Jumlah data positif yang diprediksi sebagai data negatif
c. ***False Negatives*(FN) =** Jumlah data negatif yang diprediksi sebagai data positif
d. ***True Negatives*(TN) =** Jumlah data negatif yang diprediksi sebagai data negatif

Menurut (Rahmatullah, 2021), *Confusion matrix* dapat ditentukan 4 nilai yaitu *accuracy, precission, recall,* dan *f1-score*. Berikut rumus untuk menghitung *accuracy, precission, recall* dan *f-1 score* :

$$Accuracy = \frac{TP+TN}{TP+FN+FP+TN} * 100\% \ldots\ldots\ldots\ldots\ldots (1)$$

$$Precission = \frac{TP}{TP+FP} * 100\% \ldots\ldots\ldots\ldots\ldots (2)$$

$$Recall = \frac{TP}{TP+FN} * 100\% \ldots\ldots\ldots\ldots\ldots (3)$$

$$F1\text{-}Score = 2 * \frac{Precission * Recall}{Precission + Recall} \ldots\ldots\ldots\ldots\ldots (4)$$

Keterangan:

- *Accuracy* merupakan hasil pengujian berdasarkan tingkat pendekatan antara nilai aktual dengan nilai prediksi
- *Recall* merupakan nilai prediksi dalam pernyataan benar antara data yang dimodelkan kedalam metode dengan data asli.

- *Precission* merupakan nilai kecocokan data antara data yang dimodelkan kedalam metode dengan data asli (data analisis sentimen) dalam pengklasifikasian
- *F1-score* merupakan nilai rata-rata antara precission dan recall yang dibobotkan

Hasil dari pengujian tersebut akan dibandingkan untuk melihat nilai tingkat akurasi dan kinerja pada ke-4 metode tersebut.

BAB IV

HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan tentang proses perbandingan metode pada model *machine learning* dan metode pada model *deep learning* dalam analisis sentimen. Metode pada model *machine learning* yang akan digunakan ialah *Logistic Regression*, *Naive Bayes* dan *Multinomial Naive Bayes*. Sedangkan metode pada model *deep learning* yang akan digunakan ialah *BERT* dan *Multilingual BERT*. Tujuan dilakukannya perbandingan tersebut ialah mengetahui tingkat akurasi dari tiap metode dan mengetahui metode apa yang cocok digunakan dalam proses analisis sentimen pada dataset tersebut.

IV.1 Hasil *Pre-processing* Data

Tabel IV.1 Komentar pelanggan shopee

Komentar pelanggan shopee
Suka belanja Dsni. Sesuai pesanan. ini sudah kali kedua belanja dsni. Pengemasan aman. pengiriman juga cepat üòé

Tabel IV.1 merupakan salah satu komentar pelanggan shopee yang akan dilakukan proses *pre-processing*. Tahapan *pre-processing* meliputi:

1) *Cleaning* dan *case folding*

Tabel IV.2 Proses *Cleaning* dan *Case Folding*

Cleaning* dan *Case Folding	
Teks	**Output teks**
Suka belanja Dsni. Sesuai pesanan. ini sudah kali kedua belanja dsni. Pengemasan aman. pengiriman juga cepat üòé	suka belanja dsni sesuai pesanan ini sudah kali kedua belanja dsni pengemasan aman pengiriman juga cepat

Tabel IV.2 menunjukkan hasil dari proses *cleaning* dan *case folding* dimana menghilangkan tanda baca yang tidak perlu dan menyeragamkan menjadi huruf kecil.

2) Tokenisasi

Tabel IV.3 Proses Tokenisasi

Tokenisasi	
Teks	**Output Teks**
suka belanja dsni sesuai pesanan ini sudah kali kedua belanja dsni pengemasan aman pengiriman juga cepat	suka, belanja, dsni, sesuai, pesanan, ini, sudah, kali, kedua, belanja, dsni, pengemasan, aman, pengiriman, juga, cepat

Tabel IV.3 menunjukkan hasil dari proses tokenisasi dimana kata disortir dan dibagi menjadi kata-kata individu yang dikenal sebagai token. Token bisa berupa kata-kata, karakter, atau bagian-bagian kata.

3) Normalisasi

Tabel IV.4 Proses Normalisasi

Normalisasi	
Teks	**Output Teks**
suka, belanja, dsni, sesuai, pesanan, ini, sudah, kali, kedua, belanja, dsni, pengemasan, aman, pengiriman, juga, cepat	suka, belanja, disini, sesuai, pesanan, ini, sudah, kali, kedua, belanja, disini, pengemasan, aman, pengiriman, juga, cepat

Tabel IV.4 menunjukkan hasil dari proses normalisasi dimana dilakukan penyeragaman kata sehingga lebih mudah untuk dianalisis, diproses maupun dibandingkan dengan teks lainnya.

4) Stopword Removal

Tabel IV.5 Proses *Stopword Removal*

Stopword Removal	
Teks	**Output Teks**
suka, belanja, disini, sesuai, pesanan, ini, sudah, kali, kedua, belanja, disini, pengemasan, aman, pengiriman, juga, cepat	suka, belanja, disini, sesuai, pesanan, kedua, belanja, pengemasan, aman, pengiriman, cepat

Tabel IV.5 menunjukkan hasil dari proses s*topword removal* dimana kata yang tidak ada makna akan di hapus, seperti kata "ini", "sudah", "kali" dan "juga".

5) Stemming

Tabel IV.6 Proses *stemming*

Stemming	
Teks	**Output Teks**
suka, belanja, disini, sesuai, pesanan, kedua, belanja, pengemasan, aman, pengiriman, cepat	suka belanja disini sesuai pesanan kedua belanja pengemasan aman pengiriman cepat

Tabel IV.6 menunjukkan hasil dari proses *stemming* dimana mengembalikan kata yang berimbuhan menjadi kata induk untuk mengurangi kesalahan pada proses selanjutnya.

IV.2 *Hyperparameter*

Tahap *hyperparameter* model merupakan tahap yang digunakan untuk mengukur parameter pada metode pada model *Machine Learning* dan metode pada model *Deep Learning*.

IV.2.1 Model *Machine Learning*

Tabel IV.7 Hasil pengujian *parameter machine learning*

No	*Fold*	*Logistic regression*	*Naive Bayes*	*Multinomial Naive Bayes*
1	1	92%	94%	94%
2	2	94%	92%	94%
3	3	93%	93%	91%
4	4	95%	95%	92%
5	5	93%	92%	93%

Tabel IV.7 merupakan hasil dari pengujian parameter model *machine learning* menggunakan *K-Fold Cross Validation* sebanyak 5 fold. *K-Fold Cross Validation* adalah implementasi dari *library sklearn model_selection*. Tujuannya adalah untuk memprediksi hasil probalitas metode agar terhindar dari *overfitting*. Berdasarkan tabel IV.7, *Logistic Regression* memiliki *accuracy* tertinggi sebesar 95% pada *fold* 4. Naïve Bayes memiliki *accuracy* tertinggi sebesar 95% pada *fold* 4. *Multinomial Naïve Bayes* memiliki nilai *accuracy* tertinggi sebesar 94% pada *fold* 1 dan 2.

IV.2.2 Model *Deep Learning*

Tabel IV.8 *Hyperparameter* model *deep learning*

No	*Hyperparameter*	Ukuran	
		BERT	*Multilingual BERT*
1	*Batch_size*	8	32
2	*Epochs*	4	10
3	*Learning rate*	2e-5	2e-5
4	*Epsilon*	1e-8	1e-8

Tabel IV.8 menunjukkan *hyperparameter* model *deep learning*. Berikut merupakan penjelasan dari parameter yang digunakan oleh metode penelitian:

a. Optimizer Adam W

Algoritma *optimasi Adam (Adaptive Moment Estimation)* yang mengatasi masalah pembelajaran berlebihan pada model jaringan saraf.

b. *Batch Size*

Batch size adalah ukuran jumlah sampel data yang diberikan ke model dalam satu iterasi selama proses pelatihan atau pengujian pada model *deep learning*.

c. Learning rate

Learning rate yang digunakan pada penelitian ini adalah 2e-5. 2e-5 mewakili 2 kali 10 pangkat -5 yang setara dengan 0.00002. Pemilihan *learning rate* yang optimal sangat tergantung pada model arsitektur, dataset, dan ukuran batch.

d. Epsilon

Epsilon digunakan untuk mencegah pembagian dengan nol saat menghitung gradient. Nilai epsilon ini biasanya sangat kecil untuk menghindari masalah *numeric.*

Tabel IV.9 Hasil pengujian parameter *BERT*

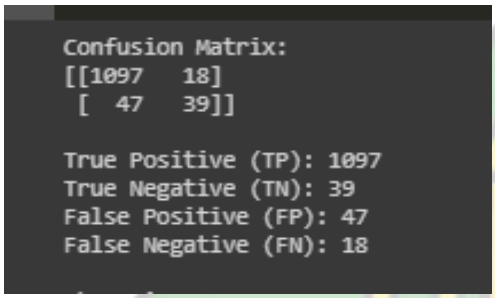
No	*Epoch*	*Accuracy*
1	1	91%
2	2	93%
3	3	94%
4	4	96%

Tabel IV.10 Hasil pengujian parameter *MBERT*

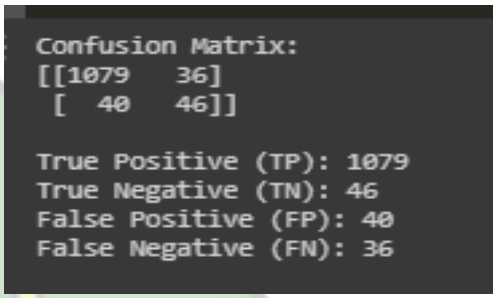
No	*Epoch*	*Accuracy*	No	*Epoch*	*Accuracy*
1	1	93%	6	6	94%
2	2	93%	7	7	93%
3	3	94%	8	8	94%
4	4	94%	9	9	94%
5	5	94%	10	10	94%

Tabel IV.9 dan IV.10 merupakan hasil pengujian parameter metode pada model *deep learning*. Berdasarkan tabel IV. 9 dan IV.10, *BERT* memiliki nilai *accuracy* tertinggi sebesar 96% pada *epoch* 4 dan MBERT memiliki nilai *accuracy* tertinggi sebesar 94% pada *epoch* 3, 4, 5, 6, 8, 9 dan 10.

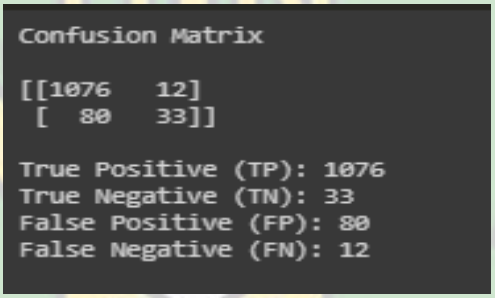
IV.3 Hasil Uji Coba *Confusion Matrix*



a. Logistic Regression

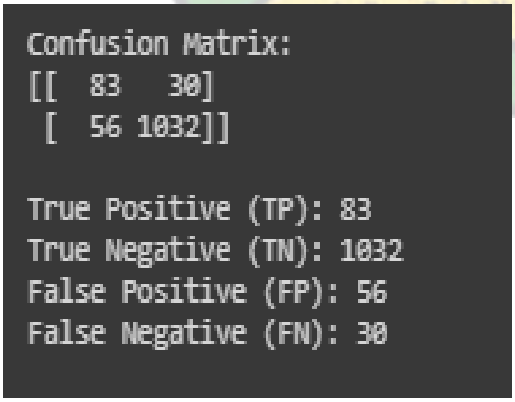


b. Naïve Bayes

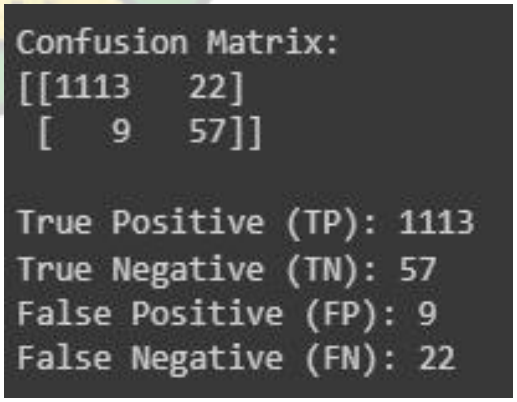


c. Multinomial Naïve Bayes

Gambar IV.1 *Confusion Matrix Machine Learning*



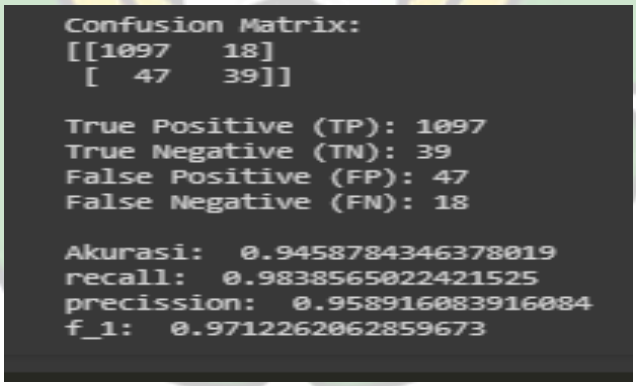
a. BERT



b. Multilingual BERT

Gambar IV.2 *Confusion Matrix Deep Learning*

Gambar IV.1 dan IV.2 merupakan hasil klasifikasi prediksi dari metode pada model *machine learning* dan *deep learning*. Sebelum ke tahap klasifikasi, data akan dilakukan pelatihan model metode dengan menggunakan data *training* x dan data *training* y. Hasil dari pelatihan metode akan dilakukan pengujian menggunakan data *testing* x kemudian akan diinput ke dalam data *y_pred* (data hasil pengujian metode). Setelah dilakukan pengujian model *y_pred*, data *testing* y akan dibandingan dengan data *y_pred* kemudian hasil perbandingan tersebut akan dimasukkan kedalam *confusion matrix* menggunakan *library sklearn_metrics* sebagai hasil klasifikasi model metode. Hasil klasifikasi metode tersebut akan menghasilkan prediksi TP (*True Positive*), TN (*True Negative*), FP (*False Positive*), dan FN (*False Negative*). Nilai predikasi tersebut akan menghasilkan nilai *accuracy, recall, precission* dan *f1-score* sesuai rumus. Dapat dilihat pada gambar IV.1 dan IV.2 terdapat nilai TP, TF, FP, dan FN yang akan digunakan ke dalam perhitungan nilai *accuracy, recall, precission* dan *f1 score*.



Gambar IV.3 *accuracy, recall, precission, f1-score Logistic Regression*

Gambar IV.3 merupakan hasil perhitungan *accuracy, recall, precission,* dan *f1 score* dari *Logistic Regression*. Untuk mendapatkan hasil seperti pada gambar IV.3, dilakukan perhitungan dengan rumus yang telah ditentukan. Berikut merupakan perhitungan untuk metode *logistic regression*.

Accuracy = ((TP+TN) / Total data *Testing*) * 100%

= ((1097+39) / 1201) * 100%

= (1136 / 1201)*100%

= 94,58%

Recall = (TP / (TP+FN))*100%

= (1097 / (1097+18))*100%

= (1097 / 1115)*100%

= 98,38%

Precission = (TP / (TP+FP))*100%

= (1097 / (1097+47))*100%

= (1097 / 1144)*100%

= 95,89 %

F1-Score = (2 x *recall* x *precission*) / (*recall + precission*)

= (2 x 98,38 % x 95,89%) / (98,38% + 95,89%)

= 97,12%

IV.2 Analisis Hasil Evaluasi

Analisis hasil evaluasi merupakan tahap penting dalam penelitian. Penelitian ini melakukan perbandingan beberapa metode pada model *Machine Learning* dengan metode pada model *Deep Learning* terhadap analisis sentimen pelanggan shopee. Hasil dari analisis ini akan memberikan gambaran yang jelas tentang kinerja dari metode-metode tersebut dalam mengerjakan tugas analisis sentimen. Metode pada model *machine learning* yang digunakan adalah *logistic regression, naive bayes* dan *multinomial naive bayes*. Sedangkan metode pada model *deep learning* yaitu *BERT* dan *Multilingual BERT*. Hasil dari pengujian metode tersebut dapat dilihat pada tabel IV.11 dan IV.12. Perbandingan kinerja masing-masing metode diukur dengan *accuracy, recall, precission* dan *f1-score.*

Tabel IV.11 Hasil pengujian model *machine learning*

No	Metode	Jumlah Data *Testing*	*Confusion Matrix*			
			Accuracy	***Recall***	***Precission***	***F1-Score***
1	*Logistic Regression*	1201 data	94.58%	98.38%	95.89%	97.12%
2	*Naive Bayes*		93.67%	96.77%	96.42%	96.59%
3	*Multinomial Naive Bayes*		92.33%	98.89%	93.07%	95.84%

Pada tabel IV.11 menunjukkan hasil pengujian model *machine learning*. Dari tabel IV.11 dapat dilihat bahwa *Logistic Regression* memperoleh *accuracy* sebesar 94.58%, *recall* sebesar 98.38%, *precission* sebesar 95.89% dan *f1-score* sebesar 97.12%. N*aive bayes* mendapatkan *accuracy* sebesar 93.67%, *recall* sebesar 96.77%, *precission* sebesar 96.42% dan *f1-score* sebesar 96.59%. *Multinomial Naive Bayes* mendapatkan *accuracy* sebesar 92.33%, *recall* sebesar 98.89%, *precission* sebesar 93.07% dan *f1-score* sebesar 95.48%. Dapat dilihat bahwa *logistic regression* mendapatkan hasil performa terbaik dibandingkan dengan metode-metode lainnya.

Tabel IV.12 Hasil pengujian model *Deep learning*

No	Metode	Jumlah Data *Testing*	*Confusion Matrix*			
			Accuracy	***Recall***	***Precission***	***F1-Score***
1	*BERT*	1201 data	92.83%	73.45%	59.71%	65.87%
2	*Multilingual BERT*		97.41%	98.06%	99.19%	98.62%

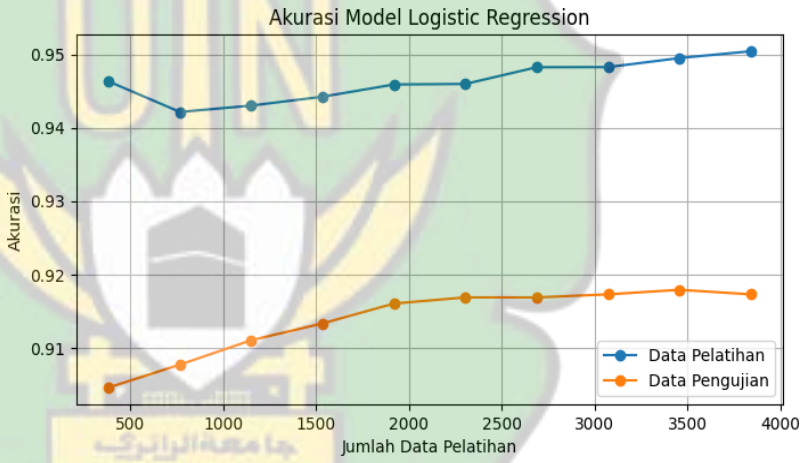
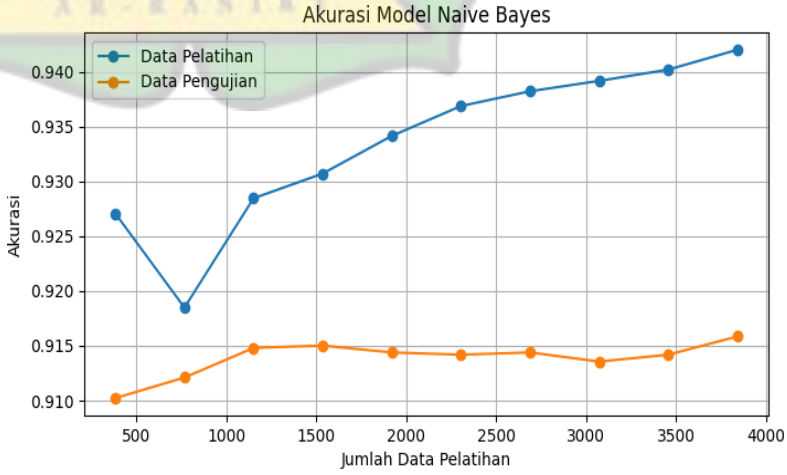
Pada tabel IV.12 menunjukkan hasil pengujian model deep learning diukur dengan nilai *accuracy, recall, precission* dan *f1 score*. Dari tabel IV.12 dapat dilihat bahwa *BERT* memperoleh nilai *accuracy* sebesar 92.83%, *recall* sebesar 73.45%, *precission* sebesar 59.71% dan *f1-score* sebesar 65.87%. Sedangkan *Multilingual BERT* mendapatkan nilai *accuracy* sebesar 97.41%, *recall* sebesar 98.06%, *precission* sebesar 99.19% dan *f1-score* sebesar 98.62%.

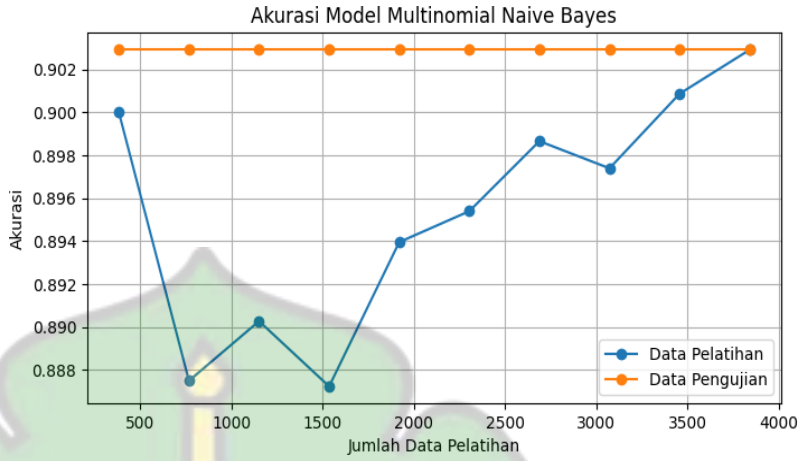
Dapat dilihat bahwa *MBERT* mendapatkan hasil performa terbaik pada *deep learing*. Hal ini terjadi karena *BERT* hanya dapat mendeteksi teks berbahasa inggris sedangkan teks yang digunakan dalam penelitian ini berbahasa indonesia. Sehingga nilai prediksi TP (True Positive) pada *BERT* sangat rendah dan mempengaruhi performa nilai *accuracy, recall, precission* dan *f1-score* pada metode tersebut. *MBERT* dapat mendeteksi banyak bahasa termasuk bahasa indonesia sehingga membuat metode ini lebih unggul dibandingkan dengan *BERT*.

Dari hasil pengujian metode, secara keseluruhan *deep learning* memiliki nilai *accuracy* lebih tinggi jika dibandingkan dengan *machine learning*. Terlihat bahwa performa *MBERT* mendapatkan hasil terbaik dengan nilai *accuracy* 97.41%. Seperti halnya dalam penelitian yang dilakukan oleh Rona nisa Sofia (Rona, 2021) dengan judul "Komparasi Metode *Machine learning* dan *Deep learning* untuk deteksi emosi pada text di sosial media" mendapatkan hasil bahwa *deep learning* memiliki performa yang lebih baik dari *machine learning* dibuktikan dengan hasil *accuracy deep learning* senilai 81.64% sedangkan *machine learning* senilai 58.10%. Selain itu, penelitian yang dilakukan oleh yudicy amelia (Yudicy, 2020) dengan judul "Perbandingan Metode *Deep learning* dan *Machine learning* untuk klasifikasi (uji coba pada data kanker payudara)" mendapatkan hasil bahwa *accuracy machine learning* sebesar 97.08% lebih kecil dibandingkan dengan *accuracy deep learning* yaitu mencapai 98.54%. Penelitian yang dilakukan oleh Nuli Giarsyani (Nuli, 2020) dengan judul "Komparasi Algoritma *Machine Learning* Dan *Deep Learning* Untuk *Named Entity Recognition"* mendapatkan hasil bahwa *deep learning* memperoleh *accuracy* yang lebih unggul dari *machine learning* yaitu sebesar 99.9% dan *machine learning* sebesar 98%. Selanjutnya penelitian yang dilakukan oleh Alfando (Alfando, 2023) dengan judul "Klasifikasi Teks Berita Berbahasa Indonesia Menggunakan *Machine Learning* Dan *Deep Learning"* mendapatkan hasil bahwa *deep learning* memberikan kinerja terbaik dengan *accuracy* sebesar 95% dibandingkan dengan *machine learning* sebesar 94.24%.

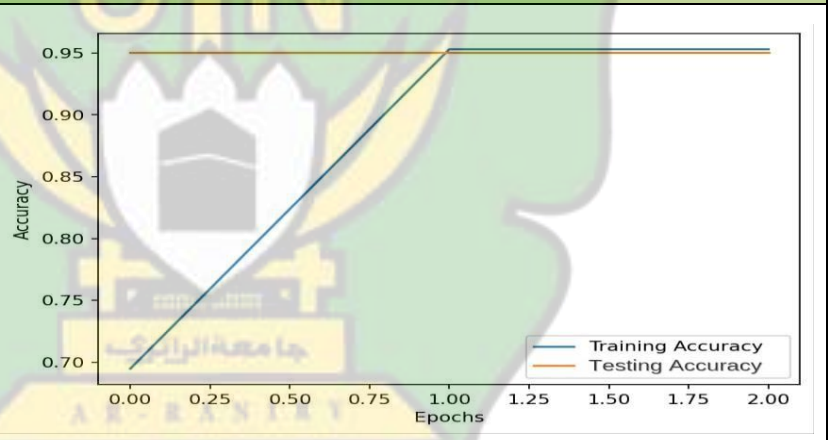
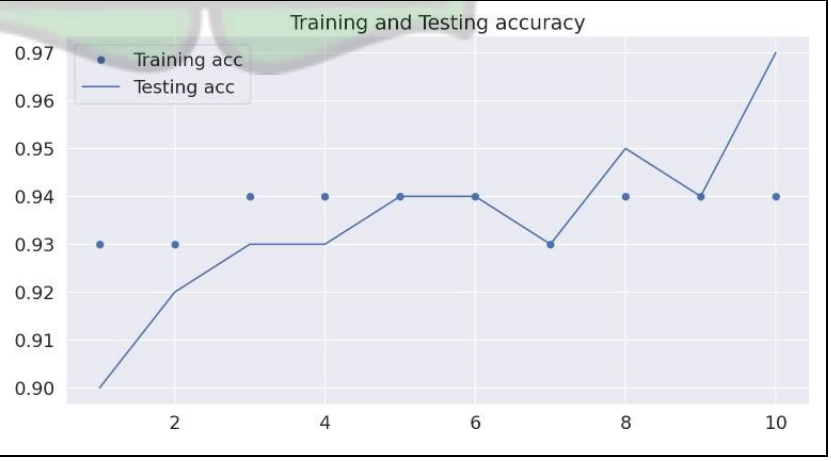
Berdasarkan penelitian yang dilakukan, *machine laerning* dan *deep learning* sudah mampu menganalisis kalimat komentar pelanggan shopee secara optimal sehingga hasil *accuracy* yang dilakukan terhadap ke-2 model ini hanya memiliki sedikit perbedaan yang membuat model-model tersebut layak digunakan untuk melakukan sebuah analisis sentimen. Tingkat *accuracy* juga dapat dipengaruhi oleh jumlah data yang digunakan. Semakin banyak data yang digunakan maka nilai *accuracy* yang dihasilkan akan semakin tinggi. Tabel IV.13 dan IV.14 merupakan grafik dari model *machine learning* dan *deep learning*.

Tabel IV.13 Grafik *Accuracy* model *machine learning*

No	Metode	Grafik
1	*Logistic Regression*	
2	*Naive Bayes*	

No	Metode	Grafik
3	*Multinomial Naive Bayes*	

Tabel IV.14 Grafik *Accuracy* model *deep learning*

No	Metode	Grafik
1	*BERT*	
2	*Multilingual BERT*	

BAB V
KESIMPULAN DAN SARAN

V.1 Kesimpulan

Berdasarkan hasil pembahasan pada bab sebelumnya, hasil dari penelitian ini dapat disimpulkan sebagai berikut:

1. Metode pada model *Machine learning* dan *deep learning* yaitu metode *Logistic Regression, Naive Bayes*, *Multinomial Naive Bayes*, *BERT* dan *Multilingual BERT* dapat digunakan dalam proses analisis sentimen pelanggan shopee melalui rangkaian proses implementasi metode yang terdiri dari pengumpulan data, *preprocessing* data, pembagian data *training* dan data *testing*, implementasi metode *machine learning* dan *deep learning*, pemodelan, dan evaluasi menggunakan *confusion matrix*.
2. Pada penelitian ini, *deep learning* memiliki nilai akurasi lebih tinggi jika dibandingkan dengan *machine learning*. Sehingga dapat disimpulkan bahwa model *deep learning* lebih baik dari model *machine learning* dalam analisis sentimen pelanggan shopee. Terlihat bahwa performa *MBERT* mendapatkan hasil terbaik dengan nilai *accuracy* sebesar 97.41%. sehingga dapat disimpulkan bahwa model deep learning
3. Pada model *machine learning*, metode *Logistic Regression* memperoleh nilai *accuracy* tertinggi yaitu sebesar 94.58%, selanjutnya untuk metode *naive bayes* mendapatkan nilai *accuracy* sebesar 3.674%, dan metode *Multinomial Naive Bayes* mendapatkan nilai *accuracy* sebesar 92.33%.
4. Pada model *deep learning*, metode *BERT* memperoleh nilai *accuracy* sebesar 92.83% dan metode *multilingual BERT* memperoleh nilai *accuracy* sebesar 97.41%.

V.2 Saran

Berdasarkan kesimpulan dari hasil dan pembahasan pada penelitian ini, saran yang dapat digunakan untuk pengembangan lanjutan dari penelitian ini adalah sebagai berikut:

1. Nilai *accuracy* analisis sentimen dapat dilakukan kembali dengan menggunakan metode –metode klasifikasi lainnya sehingga dapat mengetahui lebih banyak perbedaan tingkat *accuracy*. Terutama pada model *deep learning*, pada penelitian kali ini hanya menggunakan metode *BERT* dan *Multilingual BERT*.
2. Menambahkan jumlah dataset sehingga nilai *accuracy* yang didapat lebih akurat. Pada penelitian ini hanya menggunakan data sebanyak 6.002 data.

DAFTAR PUSTAKA

Aditya Hastami Ruger, M. S. (2021). Sentiment analisis Pelanggan Shopee di Twitter menggunakan Alogoritma Naive Bayes. *JifoTech*, 26-29.

Alan Tusa Bagus W, D. H. (2021). Klasifikais Emosi Pada teks dengan mengguanakan Metode Deep Learning. *Jurnal Ilmiah Indonesia*, 546-553.

Alfando, R. H. (2023). Klasifikasi Teks Berita Berbahasa Indonesia Menggunakan Machine Learning dan Deep Learning: Studi Literatur. *JATI*.

Aloysius Kurniawan Santoso, A. N. (2021). Klasifikasi Persepsi Pengguna Twitter Terhadap Kasus Covid-19 Menggunakan Metode Logistic Regression. *Jurnal Informatika Kaputama*, 234-241.

Aris, M. A. (2020). Perbandingan Kinerja Metode Deep Learning dan non-Deep Learning pada Analisis Sentimen Tweet Berbahasa Indonesia. *UGM*.

Dedi Darwis, N. S. (2021). Penerapan Alogirtma Naive Bayes untuk Analisis Sentimen Review Data Twitter BMKG Nasional. *Jurnal Tekno Kompak*, 131-145.

Humam, M. N. (2021). Perbandingan Kinerja Algoritma Convolutional Neural Network dan Nauve Bayes pada Analisis Sentimen Performa Manchester United Di Twitter.

Leomongga Oktaria Sihombing, H. B. (2021). Sentimen Analisis Costumer Review Produk Shopee Indonesia Menggunakan Algoritma Naive Bayes Classifier. *EDUMATIC*, 233-242.

M. Afdal, L. W. (2022). Klasifikasi Ulasan Aplikasi Shopee Menggunakan Algoritma Probabilistic Neural Network dan K-Nearest Negihbor. *IJIRSE*, 49-58.

M. Fadly Rahman, M. D. (2019). Klasifikasi untuk Diagnosa Diabtes Menggunakan Metode Byesian Regularization Neural Network (RBNN). *Jurnal Informatika*, 36-45.

M. R. Adrian, M. P. (2021). Perbandingan Metode Klasifikasi Random Forest dan SVM pada Analisis Sentimen PSBB. *Jurnal Informatika*.

Merinda Lestandy, A. L. (2021). Analisis Sentimen Tweet Vaksin COVID-19 Menggunakan Recurrent Neural Network dan Naive Bayes. *JURNAL RESTI*.

Muhammad Zidni Subarkah, M. H. (2021). Analisis Sentimen Review Tempat Wisata Pada Data Online Travel Agency Di Yogyakarta Menggunakan Model Neural Network IndoBERTweet Fine Tuning. *Seminar Nasional Official Statistics 2022*, 543-552.

Nabilah, S. (2022). *Analisis Sentimen Berbasis Aspek Pada Ulasan Aplikasi Novel di Media Sosial Menggunakan Latent Dirichlet Allocation dan Bidirectional Encoder Representation from Transformers.* Jakarta: UIN Syarif Hidayatullah.

Nofita Mahfudiyah, A. A. (2022). Analisis Persepsi Konsumen Terhadap Kualitas Layanan Gojek Menggunakan Sentiment Analysis Dan Topic Modeling Berdasarkan Deep Learning IndoBERT. *ISSN*, 1812-1817.

Nuli Giarsyani, A. F. (2020). Komparasi Algoritma Machine Learning dan Deep Learning Untuk Named Entity Recognition : Studi Kasus Data Kebencanaan. *JIRE*.

Rahmatullah, B. (2021). *Sentiment Analysis Pelaksanaan Work From Home di Indonesia Pada Masa Pandemi Covid-19 Menggunakan IndoBERT*. Surabaya: Departemen Matematika, Fakultas Sains dan Analitika Data.

Rohtman, D. (2021). Tranformers for Natural Language Processing. *Packt Publishing Ltd*, 1-355.

Rozzi Kesuma Dinata, N. H. (2020). *Machine Learning: Peroaduan memahami Data Science, Supervised Learning, Unsupervised Learning dan Reinforcement Learning.* Lhokseumawe: Unimal Press.

Sartini. (2020). Analisis Sentimen Twitter Bhasa Indonesia Menggunakan Algoritma Convolutional Neural Network. *Unnes*.

Setiadi, K. (2022). *Analisis Sentimen Pelanggan Terhadap Layanan ShopeeFood Pada Media Sosial Twitter Menggunakan Algoritma Naive Bayes dan Support Vector Machine (SVM).* Jakarta: Universitas Mercu Buana.

Rona Nisa S. A (2021). Komparasi Metode Machine Learning dan Deep Learning untuk Deteksi Emosi pada text di sosial media. *JUPITER*, 130-139.

Trie Maya Kadarina, M. H. (2019). Pengenalan Bahasa Pemrograman Phyton Menggunakan Aplikasi Games Untuk Siswa/I di Wilayah Kembangan Utara. *Jurnal Abdi Masyarakat*, 11-16.

Yasin Aril Mustofa, I. S. (2023). Analisis Sentimen Terhadap Penggunaan Aplikasi Shopee Menggunakan Algoritma Support Vector Machine (SVM). *JJEEE*, 32-35.

Yudicy Amelia, P. E. (2020). Perbandingan Metode Deep Learning untuk Klasifikasi (Uji Coba pada Data Penyakit Kanker Payudara). *SEMNATI*, 789-796.

Yusif Azhar, F. R. (2021). Analisis Klasifikasi SMS Spam Menggunakan Logistic Regression. *Jurnal Sistem Cerdas*, 155-160.

Putro, H.F., Vulandari, R.T., & Saptomo, W. L. Y. (2020). Penerapan Metode Naive Bayes Untuk Klaisifkasi Pelanggan. *Jurnal Teknologi Informasi Dan Komunikasi (TIKomSiN)*, 8(2). https://doi.org/10.30646/tikomsin.v8i2.500.

M. Azmi Aris (2020). Perbandingan Kinerja Metode Deep Learning dengan Metode Non-Deep Learning pada Analisis Sentimen Tweet Berbahasa Indonesia. Universitas Gadjah Mada.

Martinez, A. R. (2020). *Natural Language Processing. Wiley Interdisciplinary Reviews: Computational Statistics*, 2(3), 352-357.

Dewi, A. O. P. (2020). Kecerdasan Buatan sebagai Konsep Baru pada Perpustakaan. *Anuva: Jurnal Kajian Budaya, Perpustakaan, Dan Informasi*, 4(4), 453-460. https://doi.org/10.14710/anuva.4.4.453-460

Haryono, H., Palupiningsih, P., Asri, Y., & Handayani, A. N. S. (2020). Klasifikasi Pesan Gangguan Pelanggan Menggunakan Metode *Naive Bayes Classifier*. *Kilat*, 7(2), 100-108. https://doi.org/10.33322/kilat.v7i2.354

Sriyano, C. S., & Setiawan, E. B. (2021). Pendeteksi Berita Hoax Menggunakan *Naive Bayes Multinomial* pada Twitter dengan Fitur Pembobotan TF-IDF. *E-Proceeding of Engineering,* 8(2), 3396-3405.

Rendragraha, A. D., Bijaksana, M. A., & Romandhony, A. (2021). Pendekatan Metode *Transformers* untuk Deteksi Bahasa Kasar dalam Komentar Berita Online Indonesia. *E-Proceeding of Engineering*, 8(2), 3385-3395.

Tandijaya, J. H., Liliana, & Sugiarto, I. (2021). Klasifikasi dalam Pembuatan Portal Berita Online dengan Menggunakan Metode *BERT*. *Jurnal Infra*, 9(2), 320-325

LAMPIRAN

Adapun lampiran lengkap hasil dari penelitian ini sebagai berikut:

Lampiran 1. Source Code untuk Tahapan Load Dataset

Code dibawah digunakan untuk mengimport dataset dan menampilkan dataset.

```
def load_data():
    data = pd.read_excel('sudah_labeling (2).xlsx')
    return data
df = load_data()
df.head()
```

Lampiran 2. Source Code untuk tahapan Pre-processing

Code dibawah merupakan proses Cleaning dan Case Folding.

```
# proses cleaning
def remove_ulasan(Review):
    Review.encode('ascii','replace').decode('ascii')
    Review = ' '.join(re.sub("([@#][A-Zaz09]+)|(\w+:\/\/\s+)","
",Review).split())
    return Review.replace('http://', ' ').replace('https://',
df['shopee_review'] = df['shopee_review'].apply(remove_ulasan)
def remove_number(Review):
    return re.sub(r'\d+', ' ', Review)
df['shopee_review'] = df['shopee_review'].apply(remove_number)
def remove_punctuation(Review):
    return
Review.translate(str.maketrans('','',string.punctuation))
df['shopee_review'].apply(remove_punctuation)
def remove_whitespace(Review):
    return Review.strip()
df['shopee_review'] =
df['shopee_review'].apply(remove_whitespace)
def remove_whitespace_multiple(Review):
    return re.sub('\s+',' ', Review)
df['shopee_review'].apply(remove_whitespace_multiple)
def remove_singl_char(Review):
    return re.sub(r'\b[a-zA-Z]\b', '', Review)
df['shopee_review'].apply(remove_singl_char)
#case folding
df['shopee_review'] = df['shopee_review'].str.lower()
df.head(10)
```

Code dibawah ini merupakan proses Tokenisasi.

```
def word_tokenize_wrapper(Review):
  return word_tokenize(Review)
df['Tokenize'] = df['shopee_review'].apply(word_tokenize_wrapper)
df.head(5)
```

Code dibawah ini merupakan proses Normalisasi

```
normalizad_word = pd.read_csv('kbba.txt', sep='\t',
encoding='latin1')
normalizad_word_dict = {}
for index, row in normalizad_word.iterrows():
    if row[0] not in normalizad_word_dict:
        normalizad_word_dict[row[0]] = row[1]
def normalized_term(document):
    return [normalizad_word_dict[term] if term in
normalizad_word_dict else term for term in document]
df['Normalisasi'] = df['Tokenize'].apply(normalized_term)
df.head(5)
```

Code dibawah ini merupakan proses Stopword Removal

```
list_stopwords = stopwords.words('indonesian')
list_stopwords.extend(['pas','ya','sih','deh','loh','oiya','nih',
'ok'])
list_stopwords = set(list_stopwords)
def stopwords_removal(words):
    return [word for word in words if word not in list_stopwords]
df['Stopwords'] = df['Normalisasi'].apply(stopwords_removal)
df.head(5)
```

Code dibawah ini merupakan proses Stemming

```
# proses stemming
porter = PorterStemmer()
factory = StemmerFactory()
stemmer = factory.create_stemmer()
def stemmed_wrapper(term):
    return stemmer.stem(term)
term_dict = {}
for Review in df['Stopwords']:
    for term in Review:
        if term not in term_dict:
            term_dict[term] = " "
for term in term_dict:
    term_dict[term] = stemmed_wrapper(term)
```

```
def get_stemmed_term(Review):
    return [term_dict[term] for term in Review]
stem = df['Stopwords'].swifter.apply(get_stemmed_term)
def word_stemmer(text):
    stem_text = " ".join([stemmer.stem(i) for i in text])
    return stem_text
df['Stemming'] = stem.apply(word_stemmer)
df.head(5)
```

Lampiran 3. Source Code untuk tahapan Analisis Data

```
# Melihat jumlah data positif dan negatif pada data
df.dtypes
df['Label'].value_counts()
#Membuat grafik jumlah data
import seaborn as sns
import matplotlib.pyplot as plt
# Anda harus memiliki DataFrame 'df' dengan kolom 'label'
sns.countplot(data=df, x='Label')
plt.title("Category Counts")
plt.show()
```

Lampiran 4. Source Code untuk Tahapan Split Data

```
df['Stemming'] = df['Stemming'].astype(str)
counter = CountVectorizer().fit(df['Stemming'])
counter_transform = counter.transform(df['Stemming'])
# Menentukan features dan labels
x = counter_transform.toarray()
y = df['Label'].tolist()
# Menentukan jumlah data latih dan data uji
train_size = 0.8
num_training = int(train_size * len(df))
num_test = len(df) - num_training
# Menampilkan jumlah data latih dan data uji
print(f"Data-training: {num_training}")
print(f"Data-testing: {num_test}")
```

Lampiran 5. Source Code Untuk Tahapan Klasifikasi data dan menampilkan nilai akurasi pada metode Logistic Regression

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
LogisticRegression = LogisticRegression()
LogisticRegression.fit(x_train,y_train)
y_lr = LogisticRegression.predict(x_test)
y_lr
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix,
ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_lr)
cm
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=myclassifier.classes_)
disp.plot()
plt.show()
def find_evaluation_metrics(validation_set, validation_labels):
    tp, tn, fp, fn = 0, 0, 0, 0
    # Check if validation_set is a pandas DataFrame or a NumPy
array
    if isinstance(validation_set, pd.DataFrame):
        for i in range(len(validation_set)):
            guess = classify(validation_set.iloc[i])
            if guess == 1 and validation_labels[i] == 1:
                tp += 1
            elif guess == 0 and validation_labels[i] == 0:
                tn += 1
            elif guess == 1 and validation_labels[i] == 0:
                fp += 1
            else:
                fn += 1
    elif isinstance(validation_set, np.ndarray):
        for i in range(len(validation_set)):
            guess = classify(validation_set[0])
            if guess == 1 and validation_labels[i] == 1:
                tp += 1
            elif guess == 0 and validation_labels[i] == 0:
                tn += 1
```

```
            elif guess == 1 and validation_labels[i] == 0:
                fp += 1
            else:
                fn += 1
    accuracy = (tp + tn) / len(validation_set)
    recall = tp / (tp + fn)
    try:
        precision = tp / (tp + fp)
    except ZeroDivisionError:
        precision = 0
    try:
        f_1 = 2 * precision * recall / (precision + recall)
    except ZeroDivisionError:
        f_1 = 0
    return accuracy, recall, precision, f_1
# Rest of the code remains the same
# Memanggil fungsi find_evaluation_metrics
accuracy, recall, precision, f_1 =
find_evaluation_metrics(x_test, y_test)
# Menampilkan evaluasi matriks kinerja pengklasifikasian
print(f"Akurasi: {accuracy:.2f}")
print(f"Recall: {recall:.2f}")
print(f"Precision: {precision:.2f}")
print(f"F1-Score: {f_1:.2f}")
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd
# Mengganti nilai np.nan dengan string kosong pada x_train dan
x_test
x_train_list = [str(x) if pd.notna(x) else '' for x in x_train]
x_test_list = [str(x) if pd.notna(x) else '' for x in x_test]
counter = CountVectorizer()
# Selanjutnya, Anda dapat membuat vectorizer dan menghitung TF-
IDF seperti yang telah Anda lakukan
training_counts = counter.fit_transform(x_train_list)
test_counts = counter.transform(x_test_list)
# membuat model
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
# Multinomial Naive Bayes Classifier
lr=LogisticRegression()
```

```
# fit model
lr.fit(training_counts, y_train)
pred_y = lr.predict(test_counts)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
# Melakukan prediksi pada data uji
pred_y = lr.predict(test_counts)
# Menghitung confusion matrix
confusion = confusion_matrix(y_test, pred_y)
print("Confusion Matrix: ")
print(confusion)
tp = confusion[0, 0]
tn = confusion[1, 1]
fp = confusion[1, 0]
fn = confusion[0, 1]
print(f"\nTrue Positive (TP): {tp}")
print(f"True Negative (TN): {tn}")
print(f"False Positive (FP): {fp}")
print(f"False Negative (FN): {fn}")
accuracy = (tp + tn) / len(x_test)
recall = tp / (tp + fn)
precision = tp / (tp + fp)
f_1 = 2 * precision * recall / (precision + recall)
print("\nAkurasi: ", accuracy)
print("recall: ", recall)
print("precission: ", precision)
print("f_1: ", f_1)
# Membuat tampilan confusion matrix dengan seaborn
plt.figure(figsize=(5, 3))
sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues',
xticklabels=['positif', 'negatif'], yticklabels=['positif',
'negatif'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
```

```
# Melakukan prediksi pada data uji
pred_y = lr.predict(test_counts)
# Menyimpan akurasi dalam sebuah list
accuracies = []
# Menghitung akurasi untuk berbagai nilai split pada data latih
dan data uji
for split in range(80, 100, 1):
    x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=split/100, random_state=42)
    counter = CountVectorizer()
    x_train = x_train.fillna('')
    x_test = x_test.fillna('')
    train_counts = counter.fit_transform(x_train)
    test_counts = counter.transform(x_test)
    lr.fit(train_counts, y_train)
    pred_y = lr.predict(test_counts)
    accuracy = accuracy_score(y_test, pred_y)
    accuracies.append(accuracy)
# Menyimpan persentase split unatuk grafik
split_percentages = list(range(80, 100, 1))
# Membuat grafik akurasi
plt.figure(figsize=(5, 3))
plt.plot(split_percentages, accuracies, marker='o', linestyle='-
')
plt.title('Akurasi Model Logistic Regression')
plt.xlabel('Persentase Data Uji')
plt.ylabel('Akurasi')
plt.grid(True)
plt.show()
```

Lampiran 6. Source Code Untuk Tahapan Klasifikasi data dan menampilkan nilai akurasi pada metode Naive Bayes

```
print(counter_transform)
df['Label'].value_counts()
import seaborn as sns
import matplotlib.pyplot as plt
# Anda harus memiliki DataFrame 'df' dengan kolom 'label'
sns.countplot(data=df, x='Label')
plt.title("Category Counts")
plt.show()
from sklearn.model_selection import cross_val_score
```

```
from sklearn.naive_bayes import MultinomialNB
model_nb = MultinomialNB().fit(x_train, y_train)
cv_nb= cross_val_score(model_nb, x_test, y_test, cv=5)
print('Naive Bayes : ', cv_nb)
metodeBN = MultinomialNB().fit(x_train, y_train)
predictNB = metodeBN.predict(x_test)
print('Accuracy=>')
print('Naive Bayes : ', metodeBN.score(x_test, y_test))
from sklearn.metrics import confusion_matrix,
ConfusionMatrixDisplay
from sklearn.naive_bayes import MultinomialNB
import matplotlib.pyplot as plt
NaiveBayes = MultinomialNB()
NaiveBayes.fit(x_train, y_train)
y_nb = NaiveBayes.predict(x_test)
cm = confusion_matrix(y_test, y_nb)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=NaiveBayes.classes_)
disp.plot()
plt.show()
from sklearn.metrics import classification_report
print(classification_report(y_test, y_nb))
```

Lampiran 7. Source Code Untuk Tahapan Klasifikasi data dan menampilkan nilai akurasi pada metode Multinomial Naive Bayes

```
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd
# Mengganti nilai np.nan dengan string kosong pada x_train dan
x_test
x_train_list = [str(x) if pd.notna(x) else '' for x in x_train]
x_test_list = [str(x) if pd.notna(x) else '' for x in x_test]
counter = CountVectorizer()
# Selanjutnya, Anda dapat membuat vectorizer dan menghitung TF-
IDF seperti yang telah Anda lakukan
training_counts = counter.fit_transform(x_train_list)
test_counts = counter.transform(x_test_list)
# membuat model
from sklearn.naive_bayes import MultinomialNB
# Multinomial Naive Bayes Classifier
nb=MultinomialNB()
# fit model
```

```
nb.fit(training_counts, y_train)
print("Accuracy:", nb.score(test_counts, y_test))
# membuat model
from sklearn.naive_bayes import MultinomialNB
# Multinomial Naive Bayes Classifier
nb=MultinomialNB()
# fit model
nb.fit(tfidf_train, y_train)
print("Accuracy:",nb.score(tfidf_test, y_test))
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
# Melakukan prediksi pada data uji
y_pred = nb.predict(tfidf_test)
# Menghitung confusion matrix
confusion = confusion_matrix(y_test, y_pred)
# Membuat tampilan confusion matrix dengan seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues',
xticklabels=['non-spam', 'spam'], yticklabels=['non-spam',
'spam'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# print the labels
y_pred = nb.predict(tfidf_test)
# print the confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion  Matrix\n")
print(cm)
# print the Classification Report
cr = classification_report(y_test, y_pred)
print("\n\nClassification Report\n")
print(cr)
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

```
# print the labels
y_pred = nb.predict(tfidf_test)
# print the confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion  Matrix\n")
print(cm)
# Menampilkan nilai TP, TN, FP, FN
tp = cm[0, 0]
tn = cm[1, 1]
fp = cm[1, 0]
fn = cm[0, 1]
# Menampilkan nilai TP, TN, FP, FN
print(f"\nTrue Positive (TP): {tp}")
print(f"True Negative (TN): {tn}")
print(f"False Positive (FP): {fp}")
print(f"False Negative (FN): {fn}")
# menghitung akurasi
accuracy = (tp + tn) / len(x_test)
recall = tp / (tp + fn)
precision = tp / (tp + fp)
f_1 = 2 * precision * recall / (precision + recall)
print("\nAkurasi: ", accuracy)
print("recall: ", recall)
print("precision: ", precision)
print("f_1: ", f_1)
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
# Assuming x and y are your features and labels
# Replace this part with your actual data loading and processing
# Initialize Multinomial Naive Bayes model
nb = MultinomialNB()
# List to store accuracies
accuracies = []
# Initialize the CountVectorizer outside the loop
counter = CountVectorizer()
# Iterate over different split percentages
for split in range(80, 100, 1):
    # Split the data
```

```
    x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=split/100, random_state=42)
    # Fit and transform on the training data
    train_counts = counter.fit_transform(x_train.fillna(''))
    # Transform the test data using the same vectorizer
    test_counts = counter.transform(x_test.fillna(''))
    # Train the Multinomial Naive Bayes model
    nb.fit(train_counts, y_train)
    # Predict on the test data
    y_pred = nb.predict(test_counts)
    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)
    accuracies.append(accuracy)
# List of split percentages for the plot
split_percentages = list(range(80, 100, 1))
# Plot the accuracy
plt.figure(figsize=(5, 3))
plt.plot(split_percentages, accuracies, marker='o', linestyle='-
')
plt.title('Akurasi Model Multinomial Naive Bayes')
plt.xlabel('Persentase Data Uji')
plt.ylabel('Akurasi')
plt.grid(True)
plt.show()
```

Lampiran 8. Source Code Untuk Tahapan Klasifikasi data dan menampilkan nilai akurasi pada metode BERT

```
# Extract features (X) and labels (y) for training set
X_train_text = train_set['Teks'].tolist()
y_train = [1 if str(Label).lower() == '1' else 0 for Label in
train_set['Label']]
# Extract features (X) and labels (y) for test set
X_test_text = test_set['Teks'].tolist()
y_test = [1 if str(Label).lower() == '1' else 0 for Label in
test_set['Label']]
from transformers import BertTokenizer,
TFBertForSequenceClassification
num_classes = 2
bert_tokenizer = BertTokenizer.from_pretrained("bert-base-
uncased", do_lower_case=True)
y_train = np.array(y_train)
```

```
y_test = np.array(y_test)
def convert_example_to_feature(review):
    return bert_tokenizer.encode_plus(review,
                    add_special_tokens = True,
                    max_length = 512,
                    padding='max_length',
                    truncation=True,
                    return_attention_mask = True)
import tensorflow as tf
# map to the expected input to TFBertForSequenceClassification
def map_example_to_dict(input_ids, attention_masks,
token_type_ids, label):
    return {
        "input_ids": input_ids,
        "token_type_ids": token_type_ids,
        "attention_mask": attention_masks,
    }, label
def encode_examples(ds):
    # prepare list, so that we can build up final TensorFlow
dataset from slices.
    input_ids_list = []
    token_type_ids_list = []
    attention_mask_list = []
    label_list = []
    for review, label in ds:
        bert_input = convert_example_to_feature(review)
        input_ids_list.append(bert_input['input_ids'])
        token_type_ids_list.append(bert_input['token_type_ids'])
        attention_mask_list.append(bert_input['attention_mask'])
        label_list.append([label])
    return tf.data.Dataset.from_tensor_slices((input_ids_list,
attention_mask_list, token_type_ids_list,
label_list)).map(map_example_to_dict)
# hyper-parameters
batch_size = 8
# train dataset
ds_train = zip(X_train_text, y_train)
ds_test = zip(X_test_text, y_test)
ds_train_encoded =
encode_examples(ds_train).shuffle(len(X_train_text)).batch(batch_
size)
```

```
ds_test_encoded = encode_examples(ds_test).batch(batch_size)
## Initialize pre-built BERT-based classifier from transformers
bert_model = TFBertForSequenceClassification.from_pretrained(
    'bert-base-uncased', num_labels=num_classes)
bert_model.summary()
# recommended learning rate for Adam 5e-5, 3e-5, 2e-5
learning_rate = 2e-5
# multiple epochs might be better as long as we will not overfit
the model
number_of_epochs = 4
# choosing Adam optimizer
optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate,
epsilon=1e-08)
# we do not have one-hot vectors, we can use sparce categorical
cross entropy and accuracy
loss =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')
bert_model.compile(loss=loss,
                   optimizer=optimizer,
                   metrics=metric)
history = bert_model.fit(ds_train_encoded,
                         batch_size=batch_size,
                         epochs=number_of_epochs,
                         validation_data=ds_test_encoded)
import matplotlib.pyplot as plt
import matplotlib
import pandas as pd
matplotlib.rcParams['figure.dpi'] = 150
# Plotting results
def plot1(history):
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs = range(1, len(acc) + 1)
    ## Accuracy plot
    plt.plot(epochs, acc, 'bo', label='Training acc')
    plt.plot(epochs, val_acc, 'b', label='Validation acc')
    plt.title('Training and validation accuracy')
    plt.legend()
```

```
    ## Loss plot
    plt.figure()
    plt.plot(epochs, loss, 'bo', label='Training loss')
    plt.plot(epochs, val_loss, 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.legend()
    plt.show()
def plot2(history):
    pd.DataFrame(history.history).plot(figsize=(8, 5))
    plt.grid(True)
    #plt.gca().set_ylim(0,1)
    plt.show()
plot2(history)
bert_model.evaluate(ds_test_encoded,
                    batch_size=batch_size)
y_test_pred = bert_model.predict(ds_test_encoded,
                                 batch_size=batch_size)
y_test_pred_class = y_test_pred[0].argmax(axis=1)
print(y_test_pred_class[:10])
print(y_test[:10])
def plot_confusion_matrix(cm,
                          target_names,
                          title='Confusion matrix',
                          cmap=None,
                          normalize=True):
    import matplotlib.pyplot as plt
    import numpy as np
    import itertools
    accuracy = np.trace(cm) / float(np.sum(cm))
    misclass = 1 - accuracy
    if cmap is None:
        cmap = plt.get_cmap('Blues')
    plt.figure(figsize=(8, 6), dpi=150)
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    if target_names is not None:
        tick_marks = np.arange(len(target_names))
        plt.xticks(tick_marks, target_names, rotation=45)
        plt.yticks(tick_marks, target_names)
    if normalize:
```

```
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    thresh = cm.max() / 1.5 if normalize else cm.max() / 2
    for i, j in itertools.product(range(cm.shape[0]),
range(cm.shape[1])):
        if normalize:
            plt.text(j,
                     i,
                     "{:0.4f}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else
"black")
        else:
            plt.text(j,
                     i,
                     "{:,}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else
"black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label\naccuracy={:0.4f};
misclass={:0.4f}'.format(
        accuracy, misclass))
    plt.show()
import sklearn
cm = sklearn.metrics.confusion_matrix(y_test,
                                      y_test_pred_class,
                                      normalize=None)
plot_confusion_matrix(cm,
                      normalize=False,
                      target_names=['neg', 'pos'],
                      title="Confusion Matrix")
import numpy as np
from sklearn.metrics import confusion_matrix
# Menghitung confusion matrix
confusion = confusion_matrix(y_test, y_test_pred_class)
# Rest of your code remains unchanged
print("\nConfusion Matrix: ")
print(confusion)
tp = confusion[0, 0]
tn = confusion[1, 1]
```

```
fp = confusion[1, 0]
fn = confusion[0, 1]
print(f"\nTrue Positive (TP): {tp}")
print(f"True Negative (TN): {tn}")
print(f"False Positive (FP): {fp}")
print(f"False Negative (FN): {fn}")
accuracy = (tp + tn) / len(y_test_flat)
recall = tp / (tp + fn)
precision = tp / (tp + fp)
f_1 = 2 * precision * recall / (precision + recall)
print("\nAkurasi: ", accuracy)
print("Recall: ", recall)
print("Precision: ", precision)
print("F1 Score: ", f_1)
```

Lampiran 9. Source Code Untuk Tahapan Klasifikasi data dan menampilkan nilai akurasi pada metode Multilingual BERT

```
# Extract features (X) and labels (y) for training set
X_train_text = train_set['Teks'].tolist()
y_train = [1 if str(Label).lower() == '1' else 0 for Label in
train_set['Label']]
# Extract features (X) and labels (y) for test set
X_test_text = test_set['Teks'].tolist()
y_test = [1 if str(Label).lower() == '1' else 0 for Label in
test_set['Label']]
from transformers import BertTokenizer,
TFBertForSequenceClassification
num_classes = 2
bert_tokenizer = BertTokenizer.from_pretrained("bert-base-
uncased", do_lower_case=True)
y_train = np.array(y_train)
y_test = np.array(y_test)
def convert_example_to_feature(review):
    return bert_tokenizer.encode_plus(review,
                    add_special_tokens = True,
                    max_length = 512,
                    padding='max_length',
                    truncation=True,
                    return_attention_mask = True)
import tensorflow as tf
# map to the expected input to TFBertForSequenceClassification
```

```
def map_example_to_dict(input_ids, attention_masks,
token_type_ids, label):
    return {
        "input_ids": input_ids,
        "token_type_ids": token_type_ids,
        "attention_mask": attention_masks,
    }, label
def encode_examples(ds):
    # prepare list, so that we can build up final TensorFlow
dataset from slices.
    input_ids_list = []
    token_type_ids_list = []
    attention_mask_list = []
    label_list = []
    for review, label in ds:
        bert_input = convert_example_to_feature(review)
        input_ids_list.append(bert_input['input_ids'])
        token_type_ids_list.append(bert_input['token_type_ids'])
        attention_mask_list.append(bert_input['attention_mask'])
        label_list.append([label])
    return tf.data.Dataset.from_tensor_slices((input_ids_list,
attention_mask_list, token_type_ids_list,
label_list)).map(map_example_to_dict)
# hyper-parameters
batch_size = 8
# train dataset
ds_train = zip(X_train_text, y_train)
ds_test = zip(X_test_text, y_test)
ds_train_encoded =
encode_examples(ds_train).shuffle(len(X_train_text)).batch(batch_
size)
ds_test_encoded = encode_examples(ds_test).batch(batch_size)
## Initialize pre-built BERT-based classifier from transformers
bert_model = TFBertForSequenceClassification.from_pretrained(
    'bert-base-uncased', num_labels=num_classes)
bert_model.summary()
# recommended learning rate for Adam 5e-5, 3e-5, 2e-5
learning_rate = 2e-5
# multiple epochs might be better as long as we will not overfit
the model
number_of_epochs = 4
```

```
# choosing Adam optimizer
optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate,
epsilon=1e-08)
# we do not have one-hot vectors, we can use sparce categorical
cross entropy and accuracy
loss =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')
bert_model.compile(loss=loss,
                   optimizer=optimizer,
                   metrics=metric)
history = bert_model.fit(ds_train_encoded,
                         batch_size=batch_size,
                         epochs=number_of_epochs,
                         validation_data=ds_test_encoded)
import matplotlib.pyplot as plt
import matplotlib
import pandas as pd
matplotlib.rcParams['figure.dpi'] = 150
# Plotting results
def plot1(history):
    acc  = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs = range(1, len(acc) + 1)
    ## Accuracy plot
    plt.plot(epochs, acc, 'bo', label='Training acc')
    plt.plot(epochs, val_acc, 'b', label='Validation acc')
    plt.title('Training and validation accuracy')
    plt.legend()
    ## Loss plot
    plt.figure()
    plt.plot(epochs, loss, 'bo', label='Training loss')
    plt.plot(epochs, val_loss, 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.legend()
    plt.show()
def plot2(history):
    pd.DataFrame(history.history).plot(figsize=(8, 5))
    plt.grid(True)
```

```
    #plt.gca().set_ylim(0,1)
    plt.show()
plot2(history)
bert_model.evaluate(ds_test_encoded,
                    batch_size=batch_size)
y_test_pred = bert_model.predict(ds_test_encoded,
                                 batch_size=batch_size)
y_test_pred_class = y_test_pred[0].argmax(axis=1)
print(y_test_pred_class[:10])
print(y_test[:10])
def plot_confusion_matrix(cm,
                          target_names,
                          title='Confusion matrix',
                          cmap=None,
                          normalize=True):
    import matplotlib.pyplot as plt
    import numpy as np
    import itertools
    accuracy = np.trace(cm) / float(np.sum(cm))
    misclass = 1 - accuracy
    if cmap is None:
        cmap = plt.get_cmap('Blues')
    plt.figure(figsize=(8, 6), dpi=150)
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    if target_names is not None:
        tick_marks = np.arange(len(target_names))
        plt.xticks(tick_marks, target_names, rotation=45)
        plt.yticks(tick_marks, target_names)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    thresh = cm.max() / 1.5 if normalize else cm.max() / 2
    for i, j in itertools.product(range(cm.shape[0]),
range(cm.shape[1])):
        if normalize:
            plt.text(j,
                     i,
                     "{:0.4f}".format(cm[i, j]),
                     horizontalalignment="center",
```

```
                    color="white" if cm[i, j] > thresh else
"black")
        else:
            plt.text(j,
                     i,
                     "{:,}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else
"black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label\naccuracy={:0.4f};
misclass={:0.4f}'.format(
        accuracy, misclass))
    plt.show()
import sklearn
cm = sklearn.metrics.confusion_matrix(y_test,
                                      y_test_pred_class,
                                      normalize=None)
plot_confusion_matrix(cm,
                      normalize=False,
                      target_names=['neg', 'pos'],
                      title="Confusion Matrix")
import numpy as np
from sklearn.metrics import confusion_matrix
# Menghitung confusion matrix
confusion = confusion_matrix(y_test, y_test_pred_class)
# Rest of your code remains unchanged
print("\nConfusion Matrix: ")
print(confusion)
tp = confusion[0, 0]
tn = confusion[1, 1]
fp = confusion[1, 0]
fn = confusion[0, 1]
print(f"\nTrue Positive (TP): {tp}")
print(f"True Negative (TN): {tn}")
print(f"False Positive (FP): {fp}")
print(f"False Negative (FN): {fn}")
accuracy = (tp + tn) / len(y_test_flat)
recall = tp / (tp + fn)
precision = tp / (tp + fp)
```

```
f_1 = 2 * precision * recall / (precision + recall)
print("\nAkurasi: ", accuracy)
print("Recall: ", recall)
print("Precision: ", precision)
print("F1 Score: ", f_1)
from sklearn.metrics import classification_report
print(classification_report(y_test, y_test_pred_class))
```