

**Penerapan Algoritma *SVM* dan *Gradient Boosting* dalam
Prediksi *Stunting* pada Balita di Kabupaten Bekasi**

Tugas Akhir

diajukan untuk memenuhi salah satu syarat memperoleh gelar sarjana

dari Program Studi S1 Informatika

Fakultas Informatika

Universitas Telkom

1301213387

Ken Arvian Narasoma



Program Studi Sarjana S1 Informatika

Fakultas Informatika

Universitas Telkom

Bandung

2025

LEMBAR PENGESAHAN

**Penerapan Algoritma *SVM* dan *Gradient Boosting* dalam Prediksi
Stunting pada Balita di Kabupaten Bekasi**

***Application of SVM and Gradient Boosting Algorithms in Predicting
Stunting in Toddlers in Bekasi Regency***

1301213387

Ken Arvian Narasoma

Tugas akhir ini telah diterima dan disahkan untuk memenuhi sebagai syarat
memperoleh gelar pada Program Studi Sarjana S1 Informatika

Fakultas Informatika

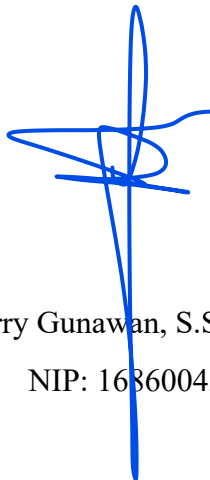
Universitas Telkom

Bandung, 5 Agustus 2025

Menyetujui

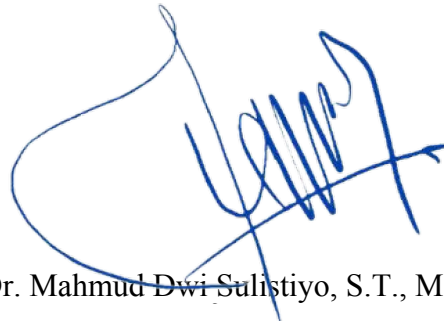
Pembimbing 1

Kaprodi S1 Informatika



Dr. Putu Harry Gunawan, S.Si., M.Si., M.Sc.

NIP: 16360043



Dr. Mahmud Dwi Sulistiyo, S.T., M.T.

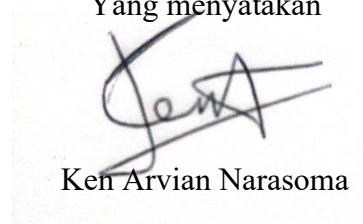
NIP: 13880017

LEMBAR ORISINALITAS

Dengan ini saya, Ken Arvian Narasoma, menyatakan sesungguhnya bahwa Tugas Akhir saya dengan judul Penerapan Algoritma *SVM* dan *Gradient Boosting* dalam Prediksi *Stunting* pada Balita di Kabupaten Bekasi berserta dengan seluruh isinya merupakan hasil karya saya sendiri, dengan tidak melakukan penjiplakan yang tidak sesuai dengan etika keilmuan yang berlaku dengan masyarakat keilmuan, serta produk dari tugas akhir ini bukan merupakan hasil dari *Generative AI*. Saya siap menanggung risiko/sanksi yang diberikan jika di kemudian hari ditemukan pelanggaran terhadap etika keilmuan dalam Laporan Tugas Akhir, atau jika ada klaisssssm dari pihak lain terhadap keaslian karya.

Bandung, 10 Juli 2025

Yang menyatakan



Ken Arvian Narasoma

NIM 1301213387

ABSTRAK

Stunting pada balita di Indonesia adalah masalah serius yang sangat umum dan berdampak signifikan pada pertumbuhan fisik dan kognitif anak-anak. Tujuan penelitian ini adalah untuk mengembangkan model prediksi *Stunting* pada balita yang menggunakan algoritma pembelajaran mesin, terutama *Support Vector Machine (SVM)* dan *Gradient Boosting*. Sebagai studi kasus, data BBLR dan PBLR Kabupaten Bekasi digunakan. Sebelum mengajarkan model *SVM* dan *Gradient Boosting*, kami memberikan penjelasan tentang proses pemrosesan *dataset*, yang mencakup *preprocessing* data dan *splitting* data. Dengan menggunakan data pengujian, evaluasi kinerja model dilakukan dengan metrik seperti akurasi, presisi, *recall*, skor F1 dan kurva *ROC*. Skenario pengujian juga digunakan untuk menguji bagaimana respons model terhadap variasi dalam beberapa parameter dan kondisi. Selain itu, mereka juga digunakan untuk menguji bagaimana model dapat digeneralisasi ke kumpulan data yang tidak dapat dilihat. Penelitian ini diharapkan dapat memberikan pemahaman yang lebih baik tentang kinerja *SVM*, *Gradient Boosting* dalam memprediksi *Stunting* pada balita serta komponen yang mempengaruhi *Stunting*.

Kata Kunci: *machine learning; Stunting; Support Vector Machine; Gradient Boosting*

ABSTRACT

Stunting in infants in Indonesia is a serious and widespread problem that has a significant impact on children's physical and cognitive growth. The objective of this study is to develop a predictive model for Stunting in infants using machine learning algorithms, specifically Support Vector Machine (SVM) and Gradient Boosting. As a case study, data from the BBLR and PBLR of Bekasi Regency were used. Before training the SVM and Gradient Boosting models, we provided an explanation of the dataset processing process, which includes data preprocessing and data splitting. Using the test data, model performance evaluation was conducted using metrics such as accuracy, precision, recall, F1 score, and ROC curve. Testing scenarios were also used to assess how the model responds to variations in certain parameters and conditions. Additionally, they were used to test how the model can be generalized to unseen datasets. This research is expected to provide a better understanding of the performance of SVM and Gradient Boosting in predicting Stunting in infants, as well as the factors influencing Stunting.

Keyword: *machine learning; Stunting; Support Vector Machine; Gradient Boosting*

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan ke hadirat *Allah Subhanahu wa Ta'ala* atas limpahan rahmat, karunia, serta petunjuk-Nya, sehingga penulis dapat menyelesaikan laporan Tugas Akhir ini dengan lancar. *Shalawat* serta salam semoga senantiasa tercurah kepada junjungan kita Nabi *Muhammad Shallallahu 'Alaihi Wasallam*, keluarga beliau, para sahabat, serta seluruh umat Islam hingga akhir zaman.

Laporan ini disusun sebagai bentuk pemenuhan salah satu syarat kelulusan pada Program Studi Sarjana Informatika, Fakultas Informatika, Universitas Telkom. Adapun penelitian yang dilakukan mengangkat topik “Penerapan Algoritma *SVM* dan *Gradient Boosting* dalam Prediksi *Stunting* pada Balita di Kabupaten Bekasi”.

Penelitian ini bertujuan untuk membangun model klasifikasi berbasis *machine learning* guna memprediksi kondisi *Stunting* pada balita, dengan harapan dapat memberikan kontribusi nyata dalam upaya penanggulangan *Stunting* di Indonesia. Pemanfaatan algoritma *Support Vector Machine (SVM)* dan *Gradient Boosting* dipilih karena efektivitasnya dalam analisis data kompleks, khususnya di bidang kesehatan masyarakat.

Ucapan terima kasih penulis sampaikan kepada dosen pembimbing atas bimbingan, arahan, dan waktu yang telah diberikan sepanjang proses penelitian. Tak lupa, penulis juga menyampaikan rasa terima kasih dan penghargaan setinggi-tingginya kepada keluarga, sahabat, dan semua pihak yang telah memberikan dukungan moral maupun spiritual selama penulisan laporan ini berlangsung.

Penulis menyadari bahwa laporan ini masih jauh dari kata sempurna. Oleh karena itu, segala bentuk kritik dan saran yang membangun sangat penulis harapkan demi perbaikan di masa mendatang. Semoga karya ini dapat memberikan manfaat dan menjadi bagian kecil dari kontribusi terhadap pengembangan ilmu pengetahuan serta mendapat keberkahan dari *Allah Subhanahu wa Ta'ala*.

UCAPAN TERIMA KASIH

Dengan penuh rasa syukur ke hadirat *Allah Subhanahu wa Ta'ala*, penulis menyampaikan ucapan terima kasih yang setulus-tulusnya kepada seluruh pihak yang telah memberikan dukungan, doa, serta kontribusi dalam penyusunan Tugas Akhir ini. Tanpa kehadiran dan peran mereka, penyusunan laporan ini tentu tidak akan berjalan sebagaimana mestinya. Penulis menyampaikan penghargaan dan terima kasih yang sebesar-besarnya kepada:

1. Ibu Ken Savitri Anugrahini, ibu kandung tercinta, yang dengan ketulusan hati, doa tanpa henti, dan kasih sayang yang tak ternilai menjadi sumber kekuatan utama dalam setiap langkah penulis. Semoga Allah membalas segala kebaikan, kesabaran, dan pengorbanan beliau dengan balasan terbaik di dunia dan akhirat.
2. Bapak Sujono dan Ibu Amilia Husnah, yang senantiasa memberikan perhatian, dorongan moral, serta dukungan spiritual selama penulis menempuh studi. Semangat dan motivasi dari keduanya telah menjadi pendorong kuat dalam menyelesaikan amanah akademik ini.
3. Dr. Putu Harry Gunawan, S.Si., M.Si., M., selaku dosen pembimbing, yang dengan penuh kesabaran dan keikhlasan telah membimbing penulis dengan arahan yang jelas dan mendalam. Nasihat dan ilmu yang diberikan sangat berharga dan membentuk kualitas dari laporan ini.
4. Seluruh dosen Fakultas Informatika Universitas Telkom, yang telah memberikan ilmu, bimbingan, serta teladan dalam proses perkuliahan. Semoga ilmu yang telah diberikan menjadi amal *jariyah* dan terus memberi manfaat bagi umat.
5. Rekan-rekan seperjuangan, yang turut membantu penulis dalam proses pembelajaran, berbagi semangat di kala lelah, dan menguatkan di kala ragu. Ucapan terima kasih secara khusus penulis sampaikan kepada:
 - a. Raihan Fadhillah Hafiizh
 - b. Syahrani Hauli
 - c. Kamelia Khoirunnisa
 - d. Ashiva Prameswara
 - e. Andini Aprilia Putri
6. Seluruh pihak lainnya, yang tidak dapat penulis sebutkan satu per satu, namun telah memberikan bantuan, doa, dan perhatian selama proses penyusunan tugas akhir ini.

Semoga segala bentuk kebaikan, bimbingan, dan dukungan yang telah diberikan menjadi amal saleh dan mendapatkan balasan yang berlimpah dari *Allah Subhanahu wa Ta'ala*. Penulis berharap karya sederhana ini dapat memberikan manfaat dan menjadi kontribusi kecil dalam pengembangan ilmu pengetahuan.

DAFTAR ISI

LEMBAR PENGESAHAN	ii
LEMBAR ORISINALITAS	iii
ABSTRAK	iv
ABSTRACT	v
KATA PENGANTAR.....	vi
UCAPAN TERIMA KASIH.....	vii
DAFTAR ISI.....	viii
DAFTAR GAMBAR.....	x
DAFTAR TABEL.....	xi
DAFTAR LAMPIRAN	xii
BAB I PENDAHULUAN	1
1.1. Latar Belakang.....	1
1.2. Perumusan Masalah.....	3
1.3. Tujuan	3
1.4. Batasan Masalah	3
BAB II TINJAUAN PUSTAKA.....	4
2.1. Studi Terkait.....	4
2.1.1. Studi Terkait Tentang <i>Stunting</i> pada Balita.....	4
2.1.2. Studi Terkait Tentang Klasifikasi untuk Prediksi <i>Stunting</i>	4
2.2. <i>Stunting</i> pada Balita	6
2.3. <i>Support Vector Machine (SVM)</i>	7
2.4. <i>Gradient Boosting</i>	9
2.5. <i>SMOTE-ENN</i>	10
BAB III PERANCANGAN DAN IMPLEMENTASI SISTEM	12
3.1. Bagan Alir	12
3.2. <i>Dataset</i> Status Gizi Balita	13
3.3. Pemrosesan <i>Dataset</i>	14
3.3.1. <i>Preprocessing</i>	14
3.3.2. Pemisahan Data <i>Test</i>	15
3.3.3. <i>Handling Imbalance</i>	16
3.3.4. <i>Splitting Data Train</i> dan Validasi.....	16
3.4. Model <i>SVM</i>	17

3.5.	Model <i>Gradient Boosting</i>	17
3.6.	Metrik Evaluasi	18
BAB IV HASIL DAN PEMBAHASAN.....		20
4.1.	Hasil Pengujian.....	21
4.1.1.	Model <i>Support Vector Machine (SVM)</i>	21
4.1.2.	Model <i>Gradient Boosting</i>	24
4.2	Analisis Pengujian	28
BAB V KESIMPULAN DAN SARAN		32
5.1.	Kesimpulan	32
5.2.	Saran.....	32
DAFTAR PUSTAKA		34
LAMPIRAN.....		37

DAFTAR GAMBAR

Gambar 2.1 Ilustrasi SVM Linear vs Non-linear	7
Gambar 3.1 Bagan Alir Penelitian	12
Gambar 4.1 Perbandingan Kinerja Model SVM dan Gradient Boosting pada Data Validasi	28
Gambar 4.2 Kurva ROC untuk Model SVM dan Gradient Boosting pada Data Validasi	28
Gambar 4.3 Perbandingan Kinerja Model SVM dan Gradient Boosting pada Data Test.....	29
Gambar 4.4 Kurva ROC untuk Model SVM dan Gradient Boosting pada Data Test.....	31

DAFTAR TABEL

Tabel 3.1 Dataset Status Gizi Balita Kabupaten Bekasi berdasarkan BBLA dan PBLA pada Februari 2024	13
Tabel 4. 1 Tabel Distribusi Sebelum dan Setelah Penerapan SMOTE-ENN	21
Tabel 4.2 Hasil Confusion Matrix dari Evaluasi Model SVM terhadap Data Validasi pada Berbagai Rasio Pembagian.....	21
Tabel 4.3 Hasil Evaluasi Model SVM terhadap Data Validasi pada Berbagai Rasio Pembagian.....	22
Tabel 4.4 Hasil Confusion Matrix dari Evaluasi Model SVM terhadap Data Test Terpisah.....	23
Tabel 4.5 Hasil Evaluasi Model SVM terhadap Data Test	23
Tabel 4.6 Hasil Confusion Matrik dari Model Gradient Boosting terhadap Data Validasi pada Berbagai Rasio Pembagian.....	24
Tabel 4.7 Hasil Evaluasi tiap Kelas Model Gradient Boosting	25
Tabel 4.8 Hasil Confusion Matrix dari Evaluasi Model Gradient Boosting terhadap Data Test.....	26
Tabel 4.9 Hasil Evaluasi Model Gradient Boosting terhadap Data Test	26

DAFTAR LAMPIRAN

Lampiran 1: Preprocessing.....	37
Lampiran 2: Pembagian Data dan Oversampling (SMOTE-ENN).....	40
Lampiran 3: Fungsi Evaluasi dan Visualisasi Model.....	41
Lampiran 4: Pelatihan Model dan Evaluasi Model.....	43

BAB I

PENDAHULUAN

1.1. Latar Belakang

Kondisi gizi yang dikenal sebagai *stunting* diidentifikasi melalui indeks PB/U atau TB/U, yang menilai status gizi anak menggunakan pengukuran *antropometri*. Ketika skor Z berada antara -2 dan -3 SD, hal itu dianggap sebagai *stunting*; ketika skor Z berada di bawah -3 SD, hal itu dianggap sebagai *stunting* berat [1]. *stunting* pada balita telah menjadi permasalahan serius di Indonesia, dengan prevalensi dan dampak yang signifikan terhadap pertumbuhan fisik dan kognitif anak-anak [2]. *World Health Organization* (WHO) mengidentifikasi bahwa Indonesia memiliki peringkat yang tinggi dalam tingkat prevalensi *stunting*, menempatkannya sebagai salah satu negara dengan masalah *stunting* yang signifikan di kawasan Asia [3].

Penelitian sebelumnya telah mencoba mempelajari faktor apa saja yang berpengaruh terhadap kejadian *stunting*. Faktor-faktor seperti status gizi ibu, pola asuh, ketersediaan pangan di rumah tangga, dan akses terhadap pelayanan kesehatan telah diidentifikasi sebagai penyebab utama *stunting* [2]. Pemerintah Indonesia telah melaksanakan Studi Status Gizi Indonesia sejak tahun 2021 dan 2022 sesuai dengan upaya tersebut. Studi ini mencakup survei status gizi sampel balita di seluruh Indonesia, termasuk Kota dan Kabupaten Bekasi. Dengan angka prevalensi *stunting* sebesar 13,8 persen pada tahun 2021, wilayah Bekasi menempati peringkat kedua terendah di Jawa Barat; namun, pada tahun 2022, prevalensi *stunting* di wilayah tersebut turun secara signifikan menjadi 6 persen, menunjukkan efektivitas inisiatif pencegahan *stunting* lokal. Selain itu, data tahun 2022 menunjukkan bahwa prevalensi berat badan rendah (*underweight*) sebesar 4,8 persen, atau 6.374 anak, dan prevalensi *wasting* (berat badan rendah) sebesar 3,9 persen, atau 5.145 anak. Meskipun demikian, masih ada tantangan besar yang perlu diatasi untuk mencapai target nasional yang ditetapkan [4].

Penelitian ini bertujuan untuk mengembangkan model prediksi pertumbuhan *stunting* pada balita menggunakan teknik *machine learning*, khususnya *Support Vector Machine (SVM)* dan *Gradient Boosting*. *Support Vector*

Machine (SVM) merupakan teknik *machine learning* yang sangat efektif dalam tugas klasifikasi dan regresi, namun *SVM* menunjukkan *recall* yang tinggi dan presisi yang rendah [5][6], di sisi lain *Gradient Boosting* merupakan teknik yang menunjukkan akurasi yang unggul dibandingkan dengan algoritma lain dalam tugas klasifikasi [7]. Pemilihan kedua teknik ini bertujuan untuk membandingkan performa algoritma klasifikasi yang memiliki karakteristik berbeda dalam memprediksi kejadian *stunting* pada balita.

Penelitian ini akan menggunakan data BBLR (Berat Badan Lahir Rendah) dan PBLR (Panjang Badan Lahir Rendah) di Kabupaten Bekasi sebagai studi kasus. Kabupaten Bekasi dipilih karena memiliki prevalensi *stunting* yang cukup tinggi dan merupakan representasi dari masalah *stunting* di daerah perkotaan di Indonesia. Berdasarkan eksplorasi awal terhadap *dataset* tersebut, ditemukan adanya ketidakseimbangan distribusi kelas antara balita yang mengalami *stunting* dan yang tidak. Dengan memanfaatkan data yang tersedia, diharapkan penelitian ini dapat memberikan wawasan yang lebih dalam tentang faktor-faktor risiko yang berkontribusi terhadap *stunting* pada anak-anak di wilayah tersebut.

Penelitian ini akan menggabungkan pemahaman yang mendalam tentang pentingnya *preprocessing* data dalam hal kesehatan anak dan berbagai masalah yang terkait dengannya. Penelitian ini akan memprioritaskan keakuratan prediksi sebagai pijakan untuk intervensi dini yang efektif dalam menghadapi *stunting*, dengan fokus pada mengatasi kesalahan yang berpotensi fatal akibat data yang tidak terproses dengan baik. Untuk mengatasi masalah ketidakseimbangan kelas dalam data, penelitian ini menerapkan pendekatan *SMOTE-ENN* (*Synthetic Minority Over-sampling Technique + Edited nearest neighbor*). Metode *SMOTE* secara umum efektif dalam meningkatkan performa model *machine learning* pada *dataset* dengan distribusi kelas yang tidak seimbang [8]. Di sisi lain, *SMOTE-ENN* dapat menghasilkan data pelatihan yang lebih bersih dan seimbang sambil juga secara signifikan meningkatkan kemampuan model untuk generalisasi pada *dataset* yang sangat tidak seimbang [9]. Penelitian diharapkan menggunakan metode ini untuk menghasilkan model prediksi yang lebih akurat, yang dapat digunakan sebagai dasar untuk mendukung upaya yang lebih tepat sasaran untuk menangani *stunting*.

1.2. Perumusan Masalah

Rumusan masalah dari penelitian ini adalah:

1. Bagaimana kinerja algoritma *Support Vector Machine (SVM)* dalam memprediksi status *stunting* pada balita?
2. Bagaimana kinerja algoritma *Gradient Boosting* dalam memprediksi status *stunting* pada balita?
3. Bagaimana perbandingan performa prediksi antara algoritma *Support Vector Machine (SVM)* dan *Gradient Boosting* berdasarkan metrik evaluasi akurasi, presisi, *recall*, *F1-score*, dan *AUC ROC*?

1.3. Tujuan

Tujuan dari penelitian ini adalah:

1. Mengevaluasi kinerja algoritma *Support Vector Machine (SVM)* dalam memprediksi status *stunting* pada balita.
2. Mengevaluasi kinerja algoritma *Gradient Boosting* dalam memprediksi status *stunting* pada balita.
3. Membandingkan performa prediksi kedua algoritma berdasarkan metrik akurasi, presisi, *recall*, *F1-score*, dan *AUC ROC*.

1.4. Batasan Masalah

Berikut adalah batasan yang terdapat pada penelitian ini:

1. Penelitian ini hanya mengambil data balita dari Kabupaten Bekasi, tidak mencakup wilayah lain di Indonesia.
2. Model prediksi yang digunakan dalam penelitian ini hanya terbatas pada dua metode *machine learning*, yaitu *Support Vector Machine (SVM)* dan *Gradient Boosting*.
3. Evaluasi kinerja model dilakukan menggunakan metrik akurasi, presisi, *recall*, *F1-score*, serta *ROC-AUC* untuk memberikan gambaran lebih menyeluruh terhadap performa klasifikasi.

BAB II

TINJAUAN PUSTAKA

2.1. Studi Terkait

2.1.1. Studi Terkait Tentang *Stunting* pada Balita

Penelitian telah menemukan bahwa beberapa faktor dapat berkorelasi dengan tingkat *Stunting* bayi. Penelitian Rusliani et al. [2] menemukan bahwa berat badan lahir rendah (LBW), pemberian ASI eksklusif, vaksinasi dasar yang tidak memadai, dan kadar zat besi dan seng yang cukup semuanya terkait dengan *stunting* pada bayi berusia 6 hingga 23 bulan. Komalasari et al. [3] juga mencapai hasil serupa. Mereka menekankan betapa pentingnya memberikan pelatihan gizi kepada ibu hamil dan ibu yang mengandung anak balita, serta kerja sama antara lembaga kesehatan untuk memastikan bahwa balita mendapatkan makanan tambahan.

Namun, menurut penelitian oleh Sumardilah dkk. [10], insiden *stunting* pada balita di lokasi penelitian secara signifikan terkait dengan faktor-faktor seperti asupan energi, menyusui eksklusif, riwayat penyakit menular, kelahiran prematur, dan pendidikan ibu. Di sisi lain, *stunting* pada balita di lokasi penelitian tidak secara signifikan terkait dengan faktor-faktor termasuk status gizi ibu, usia saat melahirkan, IMD, asupan protein dan seng, atau jarak kelahiran. Peningkatan pengetahuan tentang nutrisi ibu hamil dan peningkatan penyuluhan tentang sumber makanan yang didasarkan pada menu gizi seimbang adalah dua rekomendasi dari studi-studi ini. Meskipun berbagai studi sebelumnya telah mengidentifikasi banyak faktor penyebab *Stunting*, namun atribut-atribut tersebut tidak tersedia dalam *dataset* yang digunakan pada penelitian ini, sehingga tidak dapat disertakan secara langsung dalam proses pemodelan.

2.1.2. Studi Terkait Tentang Klasifikasi untuk Prediksi *Stunting*

Berbagai studi telah meneliti metode klasifikasi yang efektif dalam memprediksi *Stunting* pada balita, dengan algoritma *Support Vector Machine*

(*SVM*) dan *Gradient Boosting* sebagai pendekatan yang cukup populer. Penelitian oleh Nur Fitriyani Sahamony et al. [6] menunjukkan bahwa algoritma *Naive Bayes* lebih baik dalam prediksi *Stunting*. Namun, penelitian lain, seperti yang dilakukan oleh Similien Ndagijimana et al. [11], menunjukkan bahwa algoritma *Gradient Boosting* berhasil dalam mengklasifikasikan kasus-kasus *Stunting* dengan akurasi uji yang tinggi dan kemampuan untuk membedakan antara kasus *Stunting* dan tidak *Stunting*.

Penelitian yang dilakukan oleh Amanda Iksanul Putri et al. [12] menunjukkan bahwa teknik *Support Vector Machine (SVM)* memiliki performa yang lebih rendah dibandingkan algoritma *Decision Tree* yang menjadi *base learner Gradient Boosting* dalam memprediksi *Stunting* pada balita. Sementara itu, dalam penelitian yang dilakukan oleh Motaz M. H. Khorshid et al. [13], dilakukan eksperimen dengan membagi *dataset* menjadi dua bagian, yaitu 66% untuk pelatihan dan 34% untuk pengujian, hasilnya menunjukkan bahwa algoritma *SVM* mengklasifikasikan sekitar 40% data pelatihan dengan benar, sedangkan algoritma C4.5, yang merupakan salah satu bentuk dari algoritma *Decision Tree*, menunjukkan akurasi kurang dari 31% pada data pelatihan. Kedua algoritma ini dilibatkan dalam analisis komparatif bersama beberapa metode lainnya pada studi tersebut [13]. Temuan-temuan dari kedua penelitian ini meningkatkan pemahaman mengenai kemampuan algoritma *SVM* dan *Gradient Boosting* dalam menangani permasalahan *Stunting*, dan dapat menjadi dasar bagi pengembangan lebih lanjut dalam upaya penanganan dan pencegahan *Stunting* pada balita.

Namun, fitur-fitur dari data, metode *preprocessing*, dan konfigurasi parameter memiliki dampak yang signifikan terhadap seberapa baik kinerja algoritma pembelajaran mesin dalam klasifikasi. Menurut studi oleh Burdack dkk. [14], kinerja model klasifikasi sangat dipengaruhi oleh kombinasi prosedur *preprocessing* data seperti normalisasi, pengurangan dimensi menggunakan *PCA*, dan penyesuaian skala data. Mereka menemukan bahwa, tergantung pada bagaimana data dipersiapkan sebelum proses pelatihan, bahkan algoritma yang sama pun dapat memberikan hasil yang sangat

berbeda. Oleh karena itu, meskipun beberapa penelitian sebelumnya telah menemukan bahwa algoritma seperti *Gradient Boosting* secara umum lebih baik, hal ini tidak dapat dikatakan dengan pasti karena hasil prediksi sangat spesifik secara konteks dan bergantung pada sejumlah aspek teknis serta jenis set data yang digunakan [14].

2.2. *Stunting* pada Balita

Balita yang memiliki skor Z untuk panjang atau tinggi badan sesuai usia (TB/U) yang lebih dari dua simpangan baku dari median standar pertumbuhan anak dan di bawah pedoman perkembangan anak *World Health Organization* (WHO) dikatakan mengalami *stunting* [3].

Nilai *Z-Score* TB/U dihitung menggunakan parameter *LMS (Lambda-Mu-Sigma)* sesuai dengan ketentuan yang ditetapkan WHO untuk usia setiap anak dalam satuan bulan, dengan rumus sebagai berikut [15]:

$$Z\ Score = \frac{\left(\left(\frac{y}{M(t)}\right)^{L(t)} - 1\right)}{L(t) \times S(t)} \quad (2.1)$$

Keterangan:

- y : nilai pengukuran tinggi badan anak
- t : umur balita dalam satuan bulan
- $M(t)$: nilai median tinggi badan anak untuk usia t bulan
- $L(t)$: nilai parameter transformasi Box-Cox untuk usia t bulan
- $S(t)$: simpangan baku relatif untuk usia t bulan

Kondisi gizi seorang anak ditentukan menggunakan *Z-Score*. Jika *Z-Score* TB/U seorang anak kurang dari -2, anak tersebut dianggap mengalami *stunting*. Dokumen Standar Pertumbuhan Anak WHO yang resmi memuat parameter standar L, M, dan S untuk setiap usia.

Nilai *Z-Score* yang digunakan dalam penelitian ini berasal dari *dataset* yang telah dihitung sebelumnya oleh instansi penyedia data, sehingga perhitungan manual tidak dilakukan kembali. Pemahaman terhadap konsep dan

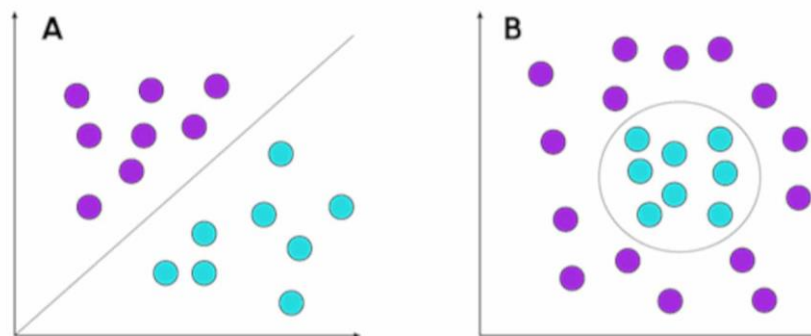
rumus *Z-Score* ini tetap penting sebagai landasan teoretis dalam menginterpretasikan status gizi balita.

2.3. Support Vector Machine (SVM)

SVM adalah metode pembelajaran mesin yang diterapkan pada tugas-tugas yang melibatkan regresi dan klasifikasi. Tujuan dari *SVM* adalah menemukan *hyperplane* optimal untuk membagi dua kelas dalam ruang *input*. *Hyperplane* ideal terletak di tengah antara dua himpunan objek dari dua kelas tersebut. *Hyperplane* terbaik diidentifikasi dengan menghitung margin, yaitu jarak antara pola terdekat dari masing-masing kelas dan *hyperplane*; vektor pendukung adalah pola terdekat [12].

Algoritma *SVM* meliputi *kernel* yang mencakup *SVM* linier dan *nonlinier*. *SVM* secara *default* hanya dapat berfungsi pada *dataset* yang dapat dipisahkan secara sederhana/*linier*. *SVM* dengan *kernel* linier dapat mengklasifikasikan data yang dapat dipisahkan secara *linier* dengan akurat dengan menciptakan *hyperplane* linier yang secara efisien memisahkan dua kelas data di ruang fitur berdimensi tinggi. Namun, *non linear SVM* digunakan untuk data dengan distribusi *non-linear* dengan menggunakan metode *kernel* pada set fitur awal.[16][17].

Untuk melihat perbedaan antara data yang dapat dipisahkan secara *linear* dan *non-linear*, bisa dilihat dalam ilustrasi berikut:



Gambar 2.1 Ilustrasi *SVM Linear vs Non-linear*

Data A menunjukkan data yang *linear*, merupakan data yang bisa di

pisahkan dengan 1 garis lurus, sedangkan data B menggambarkan data yang *tidak linear*, di mana pemisah antar kelas hanya bisa dilakukan dengan pendekatan *non-linear*. *Kernel non-linear* yang umum digunakan antara lain *Polinomial*, *Sigmoid*, dan *Radial Basis Function (RBF)* Salah satu *kernel non-linear* tersebut adalah *Radial Basis Function (RBF)*, yang memiliki persamaan sebagai berikut [18]:

$$k(x, y) = \exp (g \| x - y \|^2) \quad (2.2)$$

Keterangan:

x = Data *input*

y = Data yang akan diklasifikasikan

g = *gamma* (parameter *kernel*)

Secara matematis, fungsi keputusan dalam *SVM* dinyatakan dengan rumus berikut [12]:

$$f(x_d) = \sum_{i=1}^n a_i y_i x_i x_d + b \quad (2.3)$$

Keterangan:

ns = Jumlah *support vector*

ai = Besaran tiap titik data

yi = Kelas pada data

x_i = *Support Vector*

x_d = Data yang akan diklasifikasikan

b = Nilai *error*

Melakukan penggunaan *kernel* seperti *Radial Basis Function (RBF)*, keputusan *SVM* tidak lagi menggunakan hasil analisis linier antara vektor, melainkan ditentukan oleh hasil *kernel* antara vektor pendukung dan data yang akan diklasifikasikan. Oleh karena itu, fungsi dari *SVM* dengan

kernel RBF digambarkan sebagai berikut:

$$f(x_d) = \sum_{i=1}^{ns} a_i y_i \exp(-g \|x_i - x_d\|^2) + b \quad (2.4)$$

Mengembangkan model klasifikasi untuk mengidentifikasi situasi *Stunting* secara tepat membutuhkan pemahaman tentang dasar-dasar dan aplikasi *kernel SVM*, khususnya *kernel RBF*.

2.4. Gradient Boosting

Gradient Boosting Machines (GBM) adalah teknik *machine learning* yang kuat dan telah menunjukkan hasil yang sangat baik dalam berbagai aplikasi praktis. Metode ini dapat disesuaikan dengan kebutuhan aplikasi tertentu, seperti dipelajari dengan memperhatikan berbagai *loss function* [19]. *Gradient Boosting* digunakan untuk membangun model pembelajar dasar baru yang berkorelasi maksimal dengan gradien negatif dari *loss function* yang terkait dengan seluruh *ensemble*, untuk memberikan perkiraan yang lebih akurat dari variabel respons. Metode *Gradient Boosting* secara signifikan mengungguli algoritma pembelajaran mesin lainnya dalam memprediksi status kerdil pada anak-anak usia 0-59 bulan, dengan kekuatan diskriminasi tinggi sebesar 94,49%[11].

Secara umum, *Gradient Boosting* bekerja secara iteratif dengan membangun model prediksi awal $f(x)$. Kemudian, secara bertahap, memperbaiki model ini dengan menambahkan fungsi baru yang dilatih untuk memprediksi *pseudo-residual*, yaitu turunan negatif dari *loss function* terhadap prediksi saat ini. Proses ini mirip dengan pendekatan optimasi berbasis turunan gradien, tetapi berbeda karena turunan dihitung terhadap *output* model daripada parameter model [20]. Proses dasar algoritma *Gradient Boosting* adalah sebagai berikut:

1. Inisiasi model awal dengan meminimalkan fungsi loss:

$$f_0(x) = \arg \sum_{i=1}^n L(y_i, \gamma) \quad (2.5)$$

2. Hitung gradien negatif (*pseudo-residual*) pada setiap iterasi ke- m :

$$r_{im} = -\frac{\partial(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)} \quad (2.6)$$

3. Latih model lemah $h_m(x)$ untuk memetakan residual r_{im} .
4. Hitung nilai optimal γ_m :

$$\gamma_m = \arg \sum_{i=1}^n L(y_i, f_{m-1}(x_i) + \gamma h_m(x_i)) \quad (2.7)$$

5. Perbarui model secara iteratif:

$$F_m(x) = F_{m-1} + \gamma_m h_m(x) \quad (2.8)$$

Landasan teori untuk menggunakan *Gradient Boosting* sebagai salah satu teknik prediksi dalam penelitian ini disediakan melalui penjelasan tahapan dan prinsip-prinsip panduannya.

2.5. SMOTE-ENN

Ketidakseimbangan data (*imbalanced dataset*) merupakan kondisi ketika jumlah sampel antara kelas mayoritas dan minoritas tidak seimbang, yang dapat mengakibatkan model pembelajaran mesin bias terhadap kelas mayoritas. Salah satu pendekatan yang umum digunakan untuk menangani masalah ini adalah teknik *SMOTE-ENN*.

SMOTE-ENN menggabungkan kelebihan *SMOTE* dan *ENN*. Untuk menyeimbangkan distribusi kelas, *SMOTE* melakukan interpolasi antara titik data dan tetangga terdekatnya untuk menyediakan data sintetis bagi kelas minoritas [8]. Sementara itu, *ENN* mengurangi noise, menghilangkan outlier, dan memperjelas batas keputusan dengan menyaring sampel yang dikategorikan secara berbeda oleh mayoritas tetangganya, baik data sintetis maupun data aktual [9].

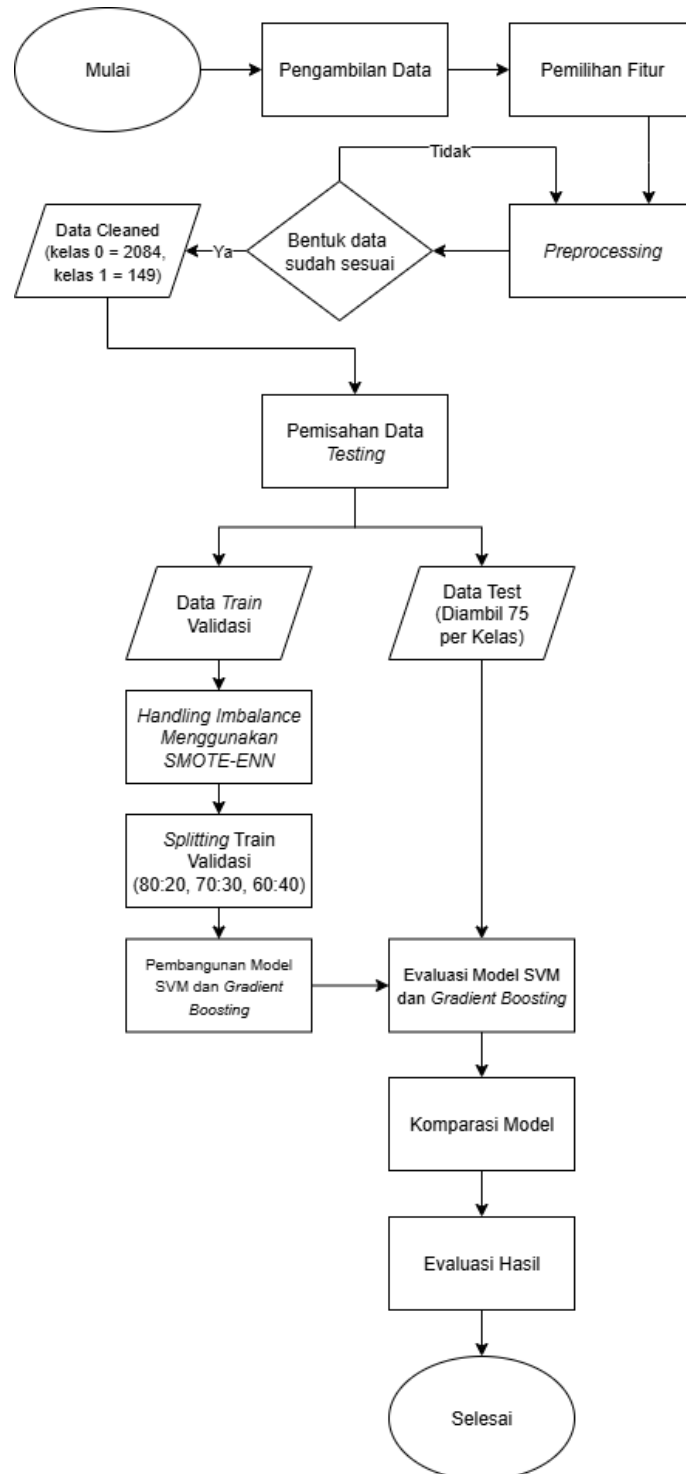
Banyak algoritma pembelajaran mesin, termasuk *XGBoost*, *SVM*, dan *K-NN*, telah menunjukkan peningkatan akurasi dan skor *ROC-AUC* setelah penerapan *SMOTE* dan seleksi fitur [21]. Namun, *SMOTE-ENN* dianggap lebih

baik karena tidak hanya menyeimbangkan data tetapi juga menyaring data yang membingungkan. Dibandingkan dengan *SMOTE* saja, penelitian menunjukkan bahwa *SMOTE-ENN* dapat meningkatkan akurasi, mengurangi kesalahan, dan menghasilkan model dengan potensi generalisasi yang lebih besar [9].

BAB III

PERANCANGAN DAN IMPLEMENTASI SISTEM

3.1. Bagan Alir



Gambar 3.1 Bagan Alir Penelitian

3.2. Dataset Status Gizi Balita

Berikut adalah contoh data yang digunakan:

Tabel 3.1 *Dataset Status Gizi Balita Kabupaten Bekasi berdasarkan BBLA dan PBLA pada Februari 2024*

Jenis Kelamin	Berat Lahir	Tinggi Lahir	Usia Saat Ukur	Berat Saat Ukur	Tinggi Saat Ukur	TB/U
L	2.2	45	4 Tahun-2 Bulan-13 Hari	20.00	116.0	Normal
L	2.4	46	2 Tahun-8 Bulan-5 Hari	22.95	98.0	Normal
P	1.93	40	3 Tahun -0 Bulan-13 Hari	15.76	93.0	Normal
P	2.2	46	3 Tahun-7 Bulan-25 Hari	16.20	112.0	Normal
P	2	42	4 Tahun-4 Bulan-2 Hari	15.00	99.0	Normal
L	2.3	46	4 Tahun-8 Bulan-5 Hari	17.10	106.0	Normal
P	2.4	45	2 Tahun-5 Bulan-28 Hari	14.00	97.0	Normal
P	2.2	43	2 Tahun-5 Bulan-28 Hari	13.70	97.0	Normal
P	2.4	41	2 Tahun-6 Bulan-11 Hari	10.80	84.0	Normal

Dataset ini terdiri dari 2.255 data yang mencakup informasi tentang individu yang lahir dengan kondisi BBLR dan PBLR di Kabupaten Bekasi. Data tersebut diperoleh dari Badan Kesatuan Bangsa dan Politik (Bankesbangpol) Kabupaten Bekasi dan terdiri dari 35 atribut yang mencakup berbagai informasi penting terkait individu tersebut. Atribut-atribut tersebut mencakup nama lengkap individu, jenis kelamin, tanggal lahir, berat badan lahir, panjang badan lahir, nama orang tua atau wali, lokasi tempat tinggal, serta status gizi dan risiko kesehatan.

Berdasarkan label status target TB/U (Tinggi Badan Menurut Usia) yang digunakan untuk menentukan kondisi *stunting*, diketahui bahwa dalam *dataset* ini terdapat dua kelas: normal dan *stunting*. Kelas normal memiliki 2.084 titik data (92,4%), sedangkan kelas *stunting* memiliki 149 titik data (6,6%). Hal ini menunjukkan adanya ketidakseimbangan kelas dalam *dataset*.

Dalam penelitian ini, tidak semua atribut digunakan untuk proses analisis dikarenakan Ada 4 alasan utama mendorong pemilihan atribut-atribut ini sebagai berikut:

- Tidak memuat atribut yang secara eksplisit mewakili faktor penyebab *Stunting* sebagaimana dijelaskan dalam literatur sebelumnya
- Karena relevan secara langsung dengan indikator *Stunting*
- Sederhana dan mudah diukur di lapangan
- Dimaksudkan untuk menghindari multikolinearitas dan redundansi yang dapat

Berikut adalah atribut yang dipilih:

- Jenis kelamin (JK) – L/P
- Berat badan lahir (BB Lahir) – Satuan: Kilogram (kg)
- Tinggi badan lahir (TB Lahir) – Satuan: Sentimeter (cm)
- Usia saat ukur – Satuan: Bulan (setelah konversi)
- Berat saat ukur – Satuan: Kilogram (kg)
- Tinggi saat ukur – Satuan: Sentimeter (cm)

Lalu untuk label yang menjadi target atau *variable output* yang digunakan adalah TB/U (tinggi badan per usia), yang menunjukkan apakah termasuk dalam kategori *stunting* atau normal. Seperti yang telah dibahas pada bagian latar belakang, distribusi kelas pada label ini menunjukkan ketidakseimbangan antara kategori "normal" dan "*stunting*". Permasalahan ini kemudian ditangani lebih lanjut dalam proses *preprocessing* data, sebagaimana dijelaskan pada Subbab 3.3.3.

3.3. Pemrosesan *Dataset*

Bagian ini akan menjelaskan langkah-langkah pemrosesan *dataset* sebelum digunakan untuk melatih model *SVM* (*Support Vector Machine*) dan *Gradient Boosting*.

3.3.1. *Preprocessing*

Sebelum memasuki proses pelatihan model, tahapan *preprocessing* dilakukan untuk memastikan kualitas data yang optimal. Proses yang dilakukan termasuk:

- Pembersihan data: menghapus kolom yang berisi data identitas seperti NIK, nama, alamat, dll. Selain itu, baris usia seperti "0 Tahun - 0 Bulan - 0 Hari" dihapus.
- Transformasi fitur usia: Kolom "Usia Saat Ukur" diubah dari teks menjadi nilai numerik dalam satuan bulan.
- Mengkodekan variabel kategori: Jenis kelamin (JK) dikodekan ke dalam nilai numerik (L = 0, P = 1). Status gizi TB/U juga dikodekan ke dalam nilai biner, dengan nilai normal/tinggi = 0, dan pendek/sangat pendek = 1.
- Penanganan *missing value*: Baris dengan nilai kosong (NaN) pada kolom penting (misalnya, BB Lahir dan TB Lahir) dihapus.
- Konversi tipe data: membulatkan atau mengubah tipe data, mengubah tinggi dan berat lahir dari karakter non-numerik dan mengubah format ke format numerik.
- Penanganan outlier: Untuk mengurangi efek nilai ekstrim, digunakan kombinasi metode *Interquartile Range* (IQR) dan *Winsorization*.
- Normalisasi fitur: Metode skala *Min-Max* digunakan untuk normalisasi fitur numerik agar berada dalam rentang 0 dan 1.
- Penanganan ketidakseimbangan kelas: Sebelum pelatihan model dimulai, distribusi kelas target TB/U diseimbangkan dengan teknik *SMOTE-ENN*.

3.3.2. Pemisahan Data *Test*

Untuk membentuk data *test*, 75 sampel positif (kelas 1) dan 75 sampel negatif (kelas 0) dipilih dari seluruh *dataset*. Proses pemisahan ini dilakukan secara seimbang antar kelas, yang berarti bahwa masing-masing kelas memiliki jumlah yang sama. Hal ini penting untuk memastikan bahwa evaluasi model tidak bias terhadap salah satu kelas

dan bahwa hasil pengukuran kinerja model lebih sesuai dengan keadaan di dunia nyata.

Sangat penting untuk memisahkan data *test* karena mereka berfungsi sebagai data independen dan digunakan untuk mengetahui seberapa kemampuan generalisasi suatu model pada data tidak dikenali sebelumnya. Evaluasi yang dilakukan dengan data *test* memberikan gambaran tentang bagaimana model berfungsi di dunia nyata. Selain itu, sangat penting untuk memisahkan data tes, terutama ketika menggunakan teknik penyeimbangan data seperti *oversampling*.

3.3.3. *Handling Imbalance*

Penanganan ketidakseimbangan kelas dilakukan sebelum proses pelatihan dan pembagian data *training* dan validasi (*splitting*), yaitu dengan menyeimbangkan distribusi kelas target TB/U menggunakan teknik *SMOTE-ENN* (*Synthetic Minority Over-sampling Technique + Edited Nearest Neighbors*). Proses ini dilakukan setelah pemisahan data *test*, sehingga teknik penyeimbangan hanya diterapkan pada data pelatihan dan validasi yang tidak mencakup data uji.

3.3.4. *Splitting Data Train dan Validasi*

Setelah menerapkan *SMOTE-ENN* untuk menyeimbangkan kelas, data dibagi menjadi data *training* untuk melatih model dan data validasi model. Beberapa rasio pembagian data digunakan, yaitu 80:20, 70:30, dan 60:40 antara data pelatihan dan validasi. Rasio 80:20 menggunakan 80% data untuk pelatihan model dan 20% untuk validasi model, memberikan cukup data untuk melatih dan mengevaluasi model. Rasio 70:30 menggunakan 70% data untuk pelatihan dan 30% untuk validasi, memberikan lebih banyak data untuk evaluasi. Rasio 60:40 menggunakan 60% data untuk pelatihan dan 40% untuk validasi, memberikan porsi data uji yang lebih besar

guna menguji ketahanan model terhadap data yang belum pernah dilihat sebelumnya. Pembagian ini memastikan bahwa model dilatih dengan data yang memadai dan dapat diuji secara objektif untuk mengetahui seberapa baik kinerjanya pada data baru.

3.4. Model SVM

Penelitian ini menggunakan *kernel Radial Basis Function (RBF)* karena memiliki kemampuan untuk menangani data dengan distribusi yang tidak *linear*, model *Support Vector Machine (SVM)* dirancang untuk menggunakannya. Sementara *gamma* diatur ke "*scale*" untuk menyesuaikan nilai secara otomatis berdasarkan jumlah fitur, parameter C akan diuji dalam kisaran 0,1 hingga 10. Untuk memungkinkan penggunaan metode kelompok di tahap lanjutan, fitur *probability=True* akan diaktifkan. Untuk memastikan bahwa hasil pelatihan konsisten, parameter *random_state* juga diatur. Selain itu, *class_weight="balanced"* diterapkan untuk membantu meningkatkan kinerja model dengan menyesuaikan bobot kelas secara otomatis berdasarkan distribusi data pelatihan. Untuk melihat dampak parameter tersebut terhadap kinerja model dalam klasifikasi, pemilihan parameter tersebut dilakukan secara bertahap.

3.5. Model Gradient Boosting

GradientBoostingClassifier digunakan untuk membangun model *Gradient Boosting*, dan berbagai kombinasi parameter akan diuji. Agar model belajar secara bertahap, *learning_rate* akan diuji antara 0.01 dan 0.1. Jumlah estimator (*n_estimators*) akan mengeksplorasi dari tiga puluh hingga enam puluh pohon, dan *max_depth* akan dibatasi menjadi 2 hingga 5 untuk menjaga kedalaman pohon tetap moderat. Untuk menjaga reproduisibilitas hasil, parameter *random_state* diatur. Selain itu, pembobotan sampel (*sample_weight*) digunakan untuk membantu meningkatkan kinerja model dengan menyesuaikan kontribusi setiap sampel selama pelatihan berdasarkan distribusi kelas. Pemilihan parameter dilakukan secara bertahap untuk menilai

pengaruhnya terhadap kemampuan model dalam menyelesaikan tugas klasifikasi.

3.6. Metrik Evaluasi

Rumus yang digunakan adalah sebagai berikut:

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (3.1)$$

$$precision = \frac{TP}{TP + FP} \quad (3.2)$$

$$recall = \frac{TP}{TP + FN} \quad (3.3)$$

$$F1\ Score = 2x\left(\frac{precision \times recall}{precision + recall}\right) \quad (3.4)$$

$$accuracy\ (per\ kelas) = \frac{TP}{TP + FP + FN} \quad (3.5)$$

Model akan di evaluasi menggunakan metrik *accuracy*, *precision*, *recall*, dan *F1-score*. Semua metrik itu akan dihitung berdasarkan nilai *True Positive*, *True Negative*, *False Positive*, dan *False Negative* yang nantinya akan diperoleh dari hasil prediksi Model *Gradient Boosting* dan *SVM* yang menggunakan data validasi yang telah di *split*. Selain menampilkan nilai metrik, visualisasi kinerja model juga ditampilkan dalam bentuk *confusion matrix* menggunakan *heatmap*. Visualisasi ini membantu menjelaskan distribusi prediksi benar dan salah masing-masing model terhadap dua kelas target, yaitu pendek/sangat pendek dan normal/tinggi. Akurasi per kelas dalam penelitian ini dihitung dengan menggunakan rumus (3.5), yang menunjukkan rasio prediksi benar terhadap seluruh prediksi yang terkait dengan kelas tersebut. Rumus ini tidak memasukkan True Negative (TN), yang berarti bahwa itu bukan akurasi klasik; sebaliknya, itu lebih tepat disebut *Intersection over Union (IoU)* atau rasio *Jaccard* per kelas.

Selain metrik klasifikasi umum, evaluasi performa model juga dilakukan dengan menggunakan *Receiver Operating Characteristic (ROC)* dan

nilai *Area Under Curve (AUC)*. Kurva *ROC* menunjukkan hubungan antara *true positive rate (TPR)* dan *false positive rate (FPR)* pada berbagai *threshold* probabilitas. Untuk memudahkan perbandingan performa visual, kedua kurva *ROC* dipetakan dalam satu plot.

BAB IV

HASIL DAN PEMBAHASAN

Penelitian ini dilakukan menggunakan Bahasa pemrograman *python* versi 3.12 yang disusun di *code editor Visual Studio Code (VSCode)* dengan format file *IPython Notebook (.ipynb)* untuk memudahkan pengolahan data. Beberapa *library* yang digunakan pada penelitian antara lain:

- ***re***: untuk pengolahan teks dan *regex*
- ***numpy***: untuk pengolahan numerik dan array
- ***pandas***: untuk membuka file *dataset*
- ***matplotlib.pyplot***: untuk visualisasi data
- ***seaborn***: untuk visualisasi data
- ***winsorize***: untuk penanganan outlier
- ***MinMaxScaler***: untuk normalisasi fitur
- ***train_test_split***: untuk membagi data menjadi data *train* dan data *test*
- ***SMOTEENN***: untuk penanganan keseimbangan data
- ***SVC***: untuk membangun model *Support Vector Machine*
- ***GradientBoostingClassifier***: untuk membangun model *Gradient Boosting*
- ***classification_report***: untuk menghasilkan ringkasan metrik evaluasi per *class*
- ***accuracy_score***: untuk menghitung nilai akurasi keseluruhan dari model
- ***confusion_matrix***: untuk membuat *confusion matrix*
- ***precision_recall_fscore_support***: untuk evaluasi model pada masing-masing *class*
- ***ROC_curve***: untuk membangun kurva *ROC (Receiver Operating Characteristic)*
- ***AUC***: untuk mengukur kualitas model berdasarkan kurva *ROC*

4.1. Hasil Pengujian

Tabel 4. 1 Tabel Distribusi Sebelum dan Setelah Penerapan *SMOTE-ENN*

Kelas TB/U	Sebelum Penerapan <i>SMOTE-ENN</i>	Setelah Penerapan <i>SMOTE-ENN</i>
0 (Negatif)	2009	1746
1 (Positif)	74	1907
Total	2083	3653

Sebagaimana telah dijelaskan pada Bab 3, data *train* dan validasi telah melalui proses penyeimbangan kelas menggunakan metode *SMOTE-ENN* untuk mengatasi ketimpangan distribusi antara kelas target. Hasil distribusi kelas setelah penerapan metode di jabarkan pada Tabel 4.1.

4.1.1. Model *Support Vector Machine (SVM)*

Tabel 4.2 Hasil *Confusion Matrix* dari Evaluasi Model *SVM* terhadap Data Validasi pada Berbagai Rasio Pembagian

Rasio Pembagian	<i>Predicted</i>		
	<i>Actual</i>	0 (Negatif)	1 (Positif)
80:20	0 (Negatif)	348	25
	1 (Positif)	4	354
70:30	0 (Negatif)	523	36
	1 (Positif)	5	532
60:40	0 (Negatif)	673	45
	1 (Positif)	5	739

Pada Tabel 4.2 menunjukkan *confusion matrix* dari model *SVM* terhadap data validasi yang didasarkan pada tiga rasio pembagian data: 80:20, 70:30, dan 60:40. Pada rasio 80:20, model berhasil mengklasifikasikan 348 data kelas negatif dan 354 data kelas positif. Namun demikian, pada rasio 70:30, model mengklasifikasikan 523 data negatif dan 532 data positif, tetapi masih terdapat 36 data negatif dan 5 data positif yang salah klasifikasi. Pada rasio 60:40, model

berhasil mengklasifikasikan 673 data negatif dan 739 data positif, tetapi masih terdapat 45 data negatif dan 5 data positif yang salah klasifikasi.

Tabel 4.3 Hasil Evaluasi Model *SVM* terhadap Data Validasi pada Berbagai Rasio Pembagian

Tipe Evaluasi	Metric	80:20	70:30	60:40
Overall	<i>Accuracy</i>	96.03%	96.26%	96.58%
	<i>Precision</i>	96.13%	96.36%	96.76%
	<i>Recall</i>	96.09%	96.31%	96.53%
	<i>F1-Score</i>	96.03%	96.26%	96.57%
0 (Negatif)	<i>Accuracy</i>	92.31%	92.73%	93.08%
	<i>Precision</i>	98.86%	99.05%	99.26%
	<i>Recall</i>	93.30%	93.56%	93.73%
	<i>F1-Score</i>	96.00%	96.23%	96.42%
	<i>Data Support</i>	373	559	718
1 (Positif)	<i>Accuracy</i>	92.43%	92.84%	93.66%
	<i>Precision</i>	93.40%	93.66%	94.26%
	<i>Recall</i>	98.88%	99.07%	99.33%
	<i>F1-Score</i>	96.07%	96.29%	96.73%
	<i>Data Support</i>	358	537	744

Pada Tabel 4.3 menunjukkan kinerja model *SVM* terhadap data validasi berdasarkan metrik evaluasi untuk tiga rasio pembagian data 80:20, 70:30, dan 60:40. Metrik yang ditampilkan termasuk akurasi, presisi, *recall*, dan skor F1 secara keseluruhan dan per kelas.

Secara keseluruhan, akurasi model adalah 96,03% pada rasio 80:20, meningkat menjadi 96,26% pada rasio 70:30, dan kembali meningkat menjadi 96,58% pada rasio 60:40. Nilai presisi masing-masing adalah 96,13%, 96,36%, dan 96,62%, sedangkan nilai *recall* masing-masing adalah 96,09%, 96,31%, dan 96,53%. *F1-Score* juga meningkat, yaitu 96,03% pada rasio 80:20, 96,26% pada rasio 70:30, dan 96,57% pada rasio 60:40.

Untuk kelas 0 (negatif), akurasi model mencapai 92,31% pada rasio 80:20, meningkat menjadi 92,73% pada rasio 70:30, dan 93,08% pada rasio 60:40. Nilai

presisi kelas ini juga cukup tinggi dan terus meningkat, yaitu 98,86%, 99,05%, dan 99,15%. Nilai F1-nya adalah 96,00%, 96,23%, dan 96,42%.

Untuk kelas 1 (Positif), nilai akurasi meningkat, yaitu 92,43% (80:20), 92,84% (70:30), dan 93,66% (60:40). Nilai presisi meningkat, yaitu 93,40%, 93,66%, dan 94,26%, sementara nilai *recall* meningkat dari 98,88% (80:20) menjadi 99,06% (70:30), dan 99,33% (60:40). Skor F1 juga meningkat secara konsisten, yaitu 96,07%, 96,29%, dan 96,73%.

Secara keseluruhan, model menunjukkan kinerja yang sangat baik dalam mengklasifikasikan kelas *stunting* dan *non-stunting* pada ketiga skenario, dan seiring dengan perubahan rasio pembagian data, metrik evaluasi meningkat secara keseluruhan.

Tabel 4.4 Hasil *Confusion Matrix* dari Evaluasi Model *SVM* terhadap Data *Test* Terpisah

Actual \ Predicted	Predicted	
	0 (Negatif)	1 (Positif)
0 (Negatif)	66	9
1 (Positif)	14	61

Tabel 4.5 Hasil Evaluasi Model *SVM* terhadap Data *Test*

Kategori Evaluasi	Metric	Nilai (%)
Overall	<i>Accuracy</i>	84.67%
	<i>Precision</i>	84.82%
	<i>Recall</i>	84.67%
	<i>F1-Score</i>	84.65%
0 (Negatif)	<i>Accuracy</i>	74.16%
	<i>Precision</i>	82.50%
	<i>Recall</i>	88.00%
	<i>F1-Score</i>	85.16%
	<i>Data Support</i>	75
1 (Positif)	<i>Accuracy</i>	72.62%
	<i>Precision</i>	87.14%
	<i>Recall</i>	81.33%

	<i>F1-Score</i>	84.14%
	<i>Data Support</i>	75

Evaluasi terhadap data *test* dilakukan menggunakan model *SVM* terbaik dari hasil pelatihan dengan rasio pembagian data 60:40. Data *test* terdiri atas 75 sampel kelas 0 (Negatif) dan 75 sampel kelas 1 (Positif) yang telah dipisahkan sebelum proses pelatihan dan tidak terkena *oversampling*.

Confusion matrix dari hasil evaluasi tersebut perlihatkan dalam Tabel 4.4. Terdapat 66 data kelas 0 dan 61 data kelas 1 yang diklasifikasikan dengan benar, 9 data kelas 0 yang salah, dan 14 data kelas 1 yang salah. Ini menunjukkan bahwa kelas positif lebih rentan terhadap kesalahan klasifikasi.

Tabel 4.5 memperlihatkan rincian metrik evaluasi. Secara keseluruhan, model menunjukkan akurasi sebesar 84,67%, presisi sebesar 84,82%, *recall* sebesar 84,67%, dan skor F1-sebesar 84,65%. Untuk kelas 0 (negatif), model menunjukkan akurasi sebesar 74,16%, presisi sebesar 82,50%, *recall* sebesar 88,00%, dan skor F1-sebesar 85,16%. Untuk kelas 1, model menunjukkan akurasi sebesar 72,62%, presisi sebesar 87,14%, *recall* sebesar 81,33%, dan skor F1-sebesar 84,14%. Hasil ini menunjukkan bahwa model memiliki performa yang cukup baik saat diuji dengan data *test*.

4.1.2. Model *Gradient Boosting*

Tabel 4.6 Hasil *Confusion Matrik* dari Model *Gradient Boosting* terhadap Data Validasi pada Berbagai Rasio Pembagian

Rasio Pembagian	<i>Predicted</i>		
	<i>Actual</i>	0 (Negatif)	1 (Positif)
80:20	0 (Negatif)	341	32
	1 (Positif)	25	333
70:30	0 (Negatif)	507	52
	1 (Positif)	35	502
60:40	0 (Negatif)	654	64
	1 (Positif)	53	691

Pada Tabel 4.6 menunjukkan *confusion matrix* dari model *Gradient Boosting* yang didasarkan pada tiga rasio pembagian data: 80:20, 70:30, dan 60:40. Pada rasio 80:20, model berhasil mengklasifikasikan 341 data kelas negatif dan 333 data kelas positif. Namun demikian, pada rasio 70:30, model mengklasifikasikan 507 data negatif dan 502 data positif, tetapi masih terdapat 52 data negatif dan 35 data positif yang salah klasifikasi. Pada rasio 60:40, model berhasil mengklasifikasikan 654 data negatif dan 691 data positif, tetapi masih terdapat 64 data negatif dan 53 data positif yang salah klasifikasi.

Tabel 4.7 Hasil Evaluasi tiap Kelas Model *Gradient Boosting*

<i>Class</i>	<i>Metric</i>	80:20	70:30	60:40
Overall	<i>Accuracy</i>	92.20%	92.06%	92.00%
	<i>Precision</i>	92.20%	92.08%	92.01%
	<i>Recall</i>	92.22%	92.09%	91.98%
	<i>F1-Score</i>	92.20%	92.06%	91.99%
0 (Negatif)	<i>Accuracy</i>	85.68%	85.35%	84.82%
	<i>Precision</i>	93.17%	93.54%	92.50%
	<i>Recall</i>	91.42%	90.70%	91.09%
	<i>F1-Score</i>	92.29%	92.10%	91.79%
	<i>Data Support</i>	373	559	718
1 (Positif)	<i>Accuracy</i>	85.38%	85.23%	85.52%
	<i>Precision</i>	91.23%	90.61%	91.52%
	<i>Recall</i>	93.02%	93.48%	92.88%
	<i>F1-Score</i>	92.12%	92.03%	92.19%
	<i>Data Support</i>	358	537	744

Tabel 4.7 menunjukkan hasil evaluasi performa model *Gradient Boosting* terhadap data validasi berdasarkan tiga rasio pembagian data: 80:20, 70:30, dan 60:40. Akurasi, presisi, *recall*, dan skor F1 digunakan untuk menilai setiap kelas dan secara keseluruhan. Secara keseluruhan, model mencatatkan akurasi sebesar 92,20% pada rasio 80:20, sedikit turun menjadi 92,06% pada rasio 70:30, dan 92,00% pada rasio 60:40. Nilai presisi juga relatif stabil, yaitu 92,20%, 92,08%,

dan 92,01% pada masing-masing rasio. Nilai *recall* juga tercatat sebesar 92,22% pada rasio 80:20, 91,98% pada rasio 70:30, dan 91,98% pada rasio 60:40. Sementara F1-score untuk ketiga rasio tersebut adalah 92,20%, 92,02%, dan 91,99%.

Untuk kelas 0 (negatif), akurasi sebesar 85,68% pada rasio 80:20, kemudian sedikit meningkat menjadi 85,35% pada rasio 70:30, dan 84,82% pada rasio 60:40. Presisi untuk kelas ini berkisar antara 93,17% dan 92,50%, sedangkan *recall* berkisar antara 91,42% pada rasio 80:20, 90,70% pada rasio 70:30, dan 90,39% pada rasio 60:40. Nilai F1-nya adalah 92,28% pada rasio 80:20, 91,90% pada rasio 70:30, seperti yang ditampilkan pada bagian 0 (Negatif) di Tabel 4.7.

Kelas 1 (Positif) menunjukkan akurasi 85,38% (80:20), 85,23% (70:30), dan 85,52% (60:40). Presisi berkisar antara 91,23% dan 91,52%, dan *recall* sebesar 93,02% pada rasio 80:20, naik menjadi 93,48% pada rasio 70:30, dan 92,88% pada rasio 60:40. Nilai F1-score masing-masing 92,03%, 92,12%, dan 92,19%. Hasil ini menunjukkan bahwa model *Gradient Boosting* berhasil mengklasifikasikan kedua kelas dengan cukup baik pada berbagai rasio pembagian data.

Tabel 4.8 Hasil *Confusion Matrix* dari Evaluasi Model *Gradient Boosting* terhadap Data Test

<i>Actual</i> \ <i>Predicted</i>	0 (Negatif)	1 (Positif)
0 (Negatif)	61	14
1 (Positif)	18	57

Tabel 4.9 Hasil Evaluasi Model *Gradient Boosting* terhadap Data Test

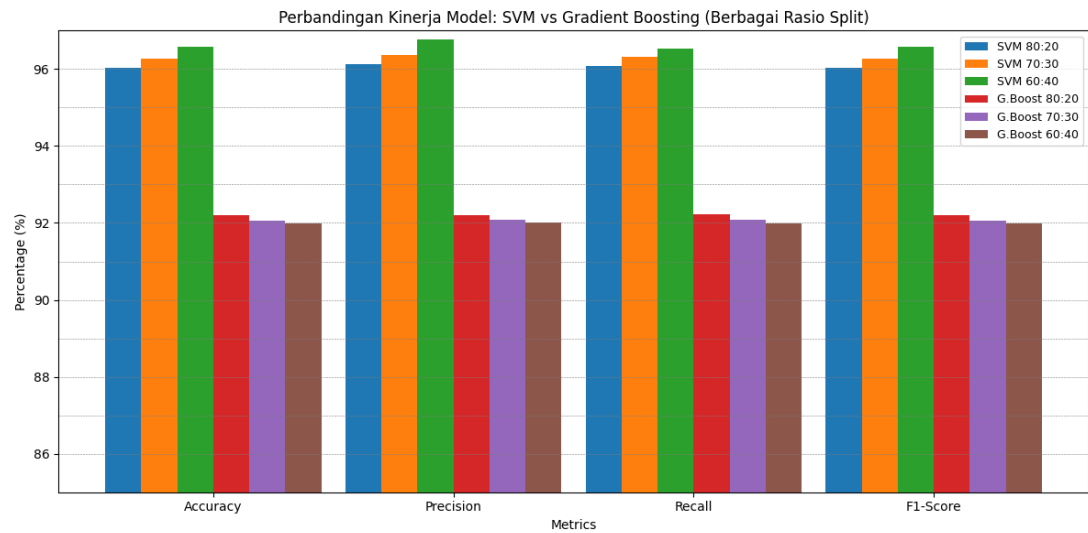
Kategori Evaluasi	Metric	Nilai (%)
Overall	<i>Accuracy</i>	78.67%
	<i>Precision</i>	78.75%
	<i>Recall</i>	78.67%
	<i>F1-Score</i>	78.65%
0 (Negatif)	<i>Accuracy</i>	65.59%
	<i>Precision</i>	77.22%
	<i>Recall</i>	81.33%

	<i>F1-Score</i>	79.22%
	<i>Data Support</i>	75
1 (Positif)	<i>Accuracy</i>	64.04%
	<i>Precision</i>	80.28%
	<i>Recall</i>	76.00%
	<i>F1-Score</i>	78.08%
	<i>Data Support</i>	75

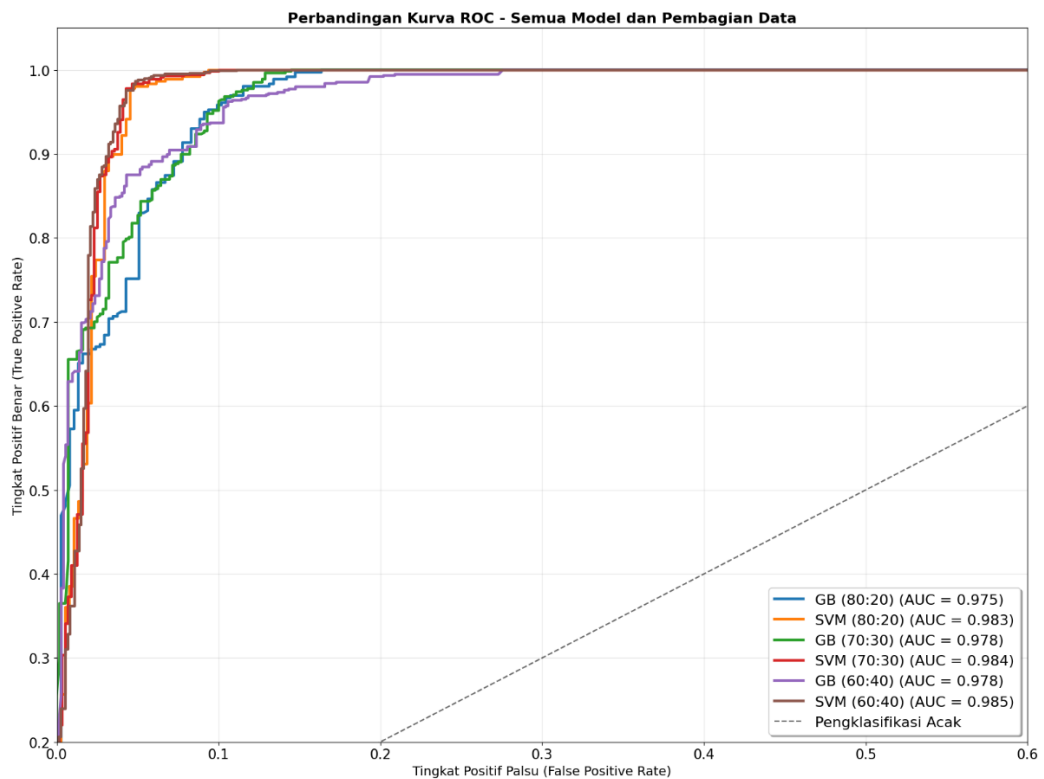
Evaluasi dilakukan terhadap data *test* menggunakan model *Gradient Boosting* terbaik dari pelatihan dengan rasio pembagian data 80:20. *Confusion matrix* dari hasil penilaian tersebut ditunjukkan pada Tabel 4.8. Dari 75 data aktual kelas 0 (negatif), 61 data berhasil dianggap benar, 14 data salah dianggap kelas 1. Dari 75 data kelas 1 (positif), 57 data berhasil dianggap benar, dan 18 data salah dianggap kelas 0.

Secara keseluruhan, model menunjukkan akurasi sebesar 78,67%, presisi sebesar 78,75%, *recall* sebesar 78,67%, dan skor F1-sebesar 78,65%. Untuk kelas 0 (negatif), nilai akurasi mencapai 65,59%, dengan presisi 77,22%, *recall* 81,33%, dan skor F1-sebesar 79,22%. Untuk kelas 1, nilai akurasi mencapai 64,04%, dengan presisi 80,28%, *recall* 76,00%, dan skor F1-sebesar 78,08%. Hasil ini menunjukkan bahwa model *Gradient Boosting* memiliki performa yang cukup baik saat diuji menggunakan data *test*, meskipun performanya berada di bawah model *SVM*.

4.2 Analisis Pengujian



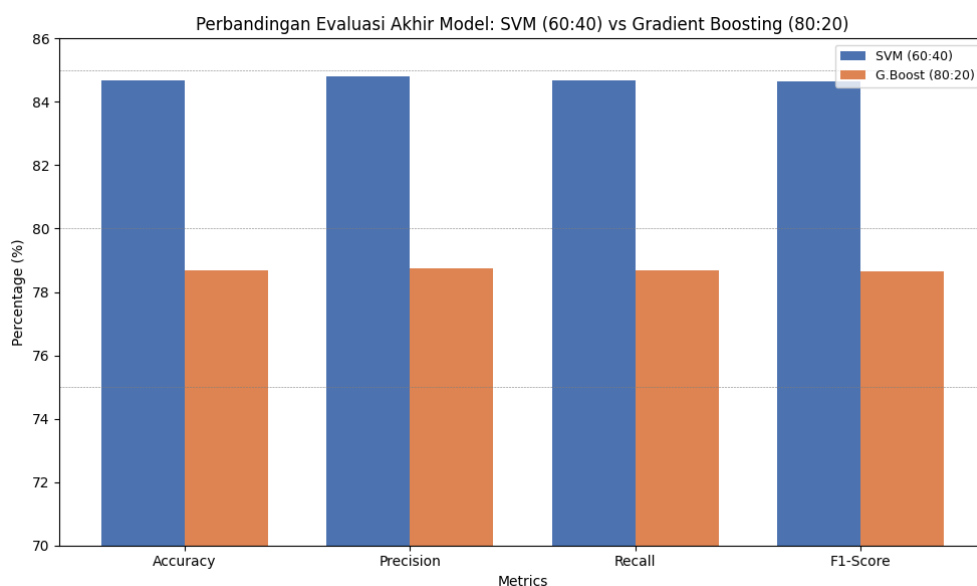
Gambar 4.1 Perbandingan Kinerja Model *SVM* dan *Gradient Boosting* pada Data Validasi



Gambar 4.2 Kurva *ROC* untuk Model *SVM* dan *Gradient Boosting* pada Data Validasi

Gambar 4.1 menunjukkan perbandingan nilai akurasi, presisi, *recall*, dan skor F1 dari model *SVM* dan *Gradient Boosting* pada tiga rasio pembagian data yang berbeda: 80:20, 70:30, dan 60:40. Dari visualisasi tersebut, terlihat bahwa model *SVM* secara konsisten memperoleh nilai metrik yang lebih tinggi dibandingkan *Gradient Boosting* di semua rasio. Pada rasio 60:40, model *SVM* memiliki performa tertinggi dengan nilai akurasi, presisi, *recall*, dan F1-Score masing-masing melebihi 96%. Sebaliknya, *Gradient Boosting* memiliki performa yang lebih rendah, dengan nilai metrik berkisar di 92% dan sedikit menurun pada rasio 60:40.

Kurva *ROC* ditampilkan pada Gambar 4.2 untuk setiap model dan rasio pembagian data. Melalui nilai *AUC* (*Area Under Curve*), kurva *ROC* mengevaluasi kinerja klasifikasi dan menunjukkan korelasi antara *true positive rate* (TPR) dan *false positive rate* (FPR). Nilai *AUC* untuk model *SVM* adalah 0.983 (80:20), 0,984 (70:30), dan 0,985 (60:40), sedangkan nilai *AUC* untuk *Gradient Boosting* adalah 0,975 (80:20), 0,978 (70:30), dan 0,978 (60:40). Kedua model dapat memisahkan kelas dengan sukses, seperti terlihat dari nilai *AUC* yang tinggi untuk ketiga rasio; namun, *SVM* secara konsisten berkinerja lebih baik daripada *Gradient Boosting* dalam hal ini.

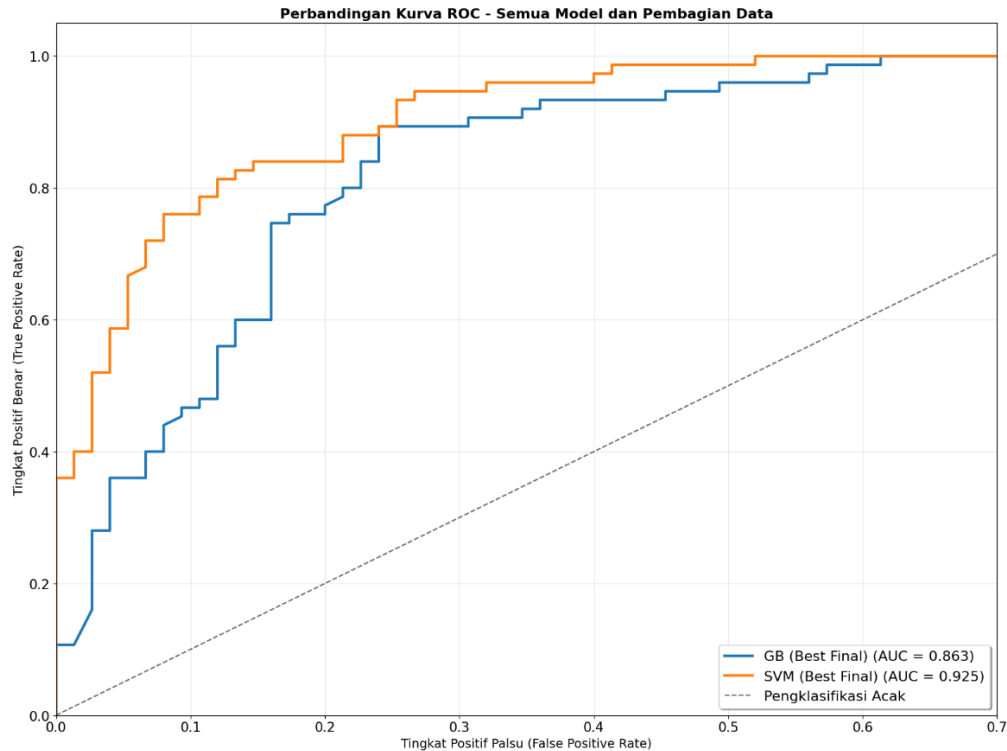


Gambar 4.3 Perbandingan Kinerja Model *SVM* dan *Gradient Boosting* pada Data Test

Model terbaik dari masing-masing algoritma digunakan untuk mengevaluasi data uji. Pelatihan dengan rasio 60:40 menghasilkan model *SVM* terbaik, sedangkan pelatihan dengan rasio 80:20 menghasilkan model *Gradient Boosting* terbaik. Perbandingan nilai metrik kedua model berdasarkan data uji ditampilkan pada Gambar 4.3. Seperti yang dapat diamati, *SVM* memperoleh skor F1 sebesar 84,65%, akurasi 84,67%, presisi 84,82%, dan *recall* 84,67%. *Gradient Boosting*, di sisi lain, mencapai akurasi 78,67%, presisi 78,75%, *recall* 78,67%, dan skor F1 78,65%. Hal ini menunjukkan bahwa model *SVM* berkinerja lebih baik dalam analisis akhir data yang belum pernah dilihat sebelumnya.

Analisis lebih lanjut per kelas menunjukkan bahwa akurasi model *SVM* untuk kelas positif (1) adalah 72,62% dan 74,16% untuk kelas negatif (0). Skor F1 adalah 85,16%, *recall* 88,00%, dan presisi 82,50% untuk kelas negatif. Skor F1 untuk kelas positif adalah 84,14%, *recall* 81,33%, dan *precision* 87,14%.

Gradient Boosting, di sisi lain, menunjukkan akurasi sebesar 64,04% untuk kelas positif dan 65,59% untuk kelas negatif. Pada kelas negatif, skor F1 sebesar 79,22%, *recall* sebesar 81,33%, dan *precision* sebesar 77,22%. Nilai F1-score sebesar 78,08%, *recall* sebesar 76,00%, dan *precision* sebesar 80,28% untuk kelas positif. Meskipun dianggap memiliki kinerja yang baik, angka-angka ini masih di bawah metrik model *SVM*.



Gambar 4.4 Kurva ROC untuk Model SVM dan Gradient Boosting pada Data Test

Kurva ROC dari penilaian model optimal pada data uji ditampilkan pada Gambar 4.4. *Gradient Boosting* mencatat *AUC* sebesar 0,863, sedangkan model *SVM* memperoleh *AUC* sebesar 0,925. Angka-angka ini menunjukkan bahwa model *SVM* mempertahankan kinerja klasifikasi yang lebih baik pada data baru sepenuhnya dan lebih andal dalam membedakan antara kelas positif dan negatif.

Dalam hal akurasi, presisi, *recall*, skor F1, dan pemisahan kelas berdasarkan nilai *AUC ROC*, model *SVM* menunjukkan kinerja yang lebih baik daripada *Gradient Boosting*, sebagaimana didukung oleh hasil penilaian data uji secara keseluruhan.

Temuan ini kemungkinan besar dipengaruhi oleh sifat data setelah *oversampling SMOTE*, yang memperbaiki distribusi data dan memudahkan *SVM* berbasis margin untuk membedakan antara data tersebut. Efektivitas model *SVM* juga didukung oleh fase eksplorasi, yang mencakup pemilihan atribut, penyesuaian *hyperparameter*, dan evaluasi strategi berbeda untuk menangani data yang tidak seimbang. Hasil evaluasi menunjukkan bahwa model *SVM* memberikan kinerja terbaik dalam penelitian ini, meskipun teknik ini bersifat eksploratif.

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Penelitian ini berhasil menerapkan algoritma *Support Vector Machine (SVM)* dan *Gradient Boosting* untuk memprediksi *stunting* pada balita berdasarkan fitur-fitur yang relevan pada *Dataset* Status Gizi Balita. Evaluasi dilakukan menggunakan data uji dan validasi yang sepenuhnya dipisahkan dari data pelatihan.

Hasil eksperimen menunjukkan bahwa algoritma *Support Vector Machine (SVM)* secara konsisten *outperform Gradient Boosting* dalam semua skenario pembagian data. Hal ini dibuktikan oleh akurasi, presisi, *recall*, *F1-score*, dan metrik *AUC-ROC* yang lebih unggul pada model *SVM*. Selain itu, *SVM* menunjukkan kemampuan generalisasi yang kuat saat diuji pada data baru, yang mendukung keandalan model dalam memprediksi status *stunting*. Sementara itu, algoritma *Gradient Boosting* menunjukkan performa yang relatif lebih rendah, terutama dalam aspek akurasi dan kemampuan generalisasi pada data uji.

Hasilnya bervariasi dibandingkan dengan penelitian sebelumnya, meskipun beberapa studi mengklaim bahwa *Gradient Boosting* lebih unggul. Perbedaan dalam sifat *dataset*, pemilihan fitur, teknik *preprocessing*, dan strategi penyeimbangan data dapat menjadi penyebab ketidaksamaan ini. Mengingat *dataset*, pemilihan fitur, dan metode *preprocessing* yang digunakan dalam penelitian ini, *SVM* menunjukkan kinerja yang lebih unggul. Oleh karena itu, berdasarkan data yang dikumpulkan, temuan penelitian ini membantu menunjukkan bahwa *SVM* adalah metode yang valid dan efisien untuk menentukan status *stunting* seseorang.

5.2. Saran

Untuk memperluas cakupan analisis, penelitian selanjutnya disarankan untuk melakukan studi lebih lanjut guna mengeksplorasi algoritma alternatif seperti *XGBoost*, *Random Forest*, atau teknik berbasis *deep learning*. Selain itu, kinerja model dapat ditingkatkan dengan menggunakan metode pencarian *hyperparameter* yang lebih canggih seperti *Grid Search* atau *Bayesian Optimization*. Untuk

meningkatkan kualitas *input*, teknik seleksi dan transformasi fitur yang lebih efisien dapat dikembangkan. Untuk meningkatkan kemampuan generalisasi model pada data aktual, disarankan agar penelitian mendatang fokus pada keseimbangan distribusi kelas dan meningkatkan volume data. Hal ini penting karena batasan studi ini meliputi distribusi kelas yang tidak seimbang dan jumlah data yang sedikit, yang dapat mempengaruhi hasil pelatihan dan evaluasi model.

DAFTAR PUSTAKA

- [1] K. Rahmadhita, “Permasalahan *Stunting* dan Pencegahannya,” *Jurnal Ilmiah Kesehatan Sandi Husada*, vol. 11, no. 1, pp. 225-229, 2020, doi: 10.35816/jiskh.v10i2.253.
- [2] N. Rusliani, W.R Hidayani, dan H. Sulistyoningsih, “Literature Review: Faktor-Faktor yang Berhubungan dengan Kejadian *Stunting* pada Balita,” *Buletin Ilmu Kebidanan dan Keperawatan (BIKK)*, vol. 01, no. 01, p. 32-40, 2022, doi: 10.56741/bikk.v1i01.39.
- [3] Komalasari, E. Supriati, R. Sanjaya, dan H. Ifayanti, “Faktor-Faktor Penyebab Kejadian *Stunting* Pada Balita,” *Majalah Kesehatan Indonesia*, vol. 1, Issue 2, p. 51-56, 2020.
- [4] Pemerintahan Kota Bekasi, “Pemkot Bekasi Gelar Sejumlah Program Guna Tekan Angka *Stunting*,” 2023. Tersedia pada: <https://www.bekasikota.go.id/detail/pemkot-bekasi-gelar-sejumlah-program-guna-tekan-angka-Stunting>. [Diakses 22 April 2024]
- [5] A.T. Zy, dan W. Hadikristanto, “Implementasi Algoritma Metode *Naive Bayes* dan *Support Vector Machine* Tentang Pembobolan dan Kebocoran Data di Twitter,” *Bulletin of Information Technology (BIT)*, vol. 4, no. 1, hal 49 – 56, 2023, doi: 10.47065/bit.v3i1.493.
- [6] N.F. Sahamony, Terttiaavini, dan H. Rianto, “Analisis Perbandingan Kinerja Model *Machine Learning* untuk Memprediksi Risiko *Stunting* pada Pertumbuhan Anak,” *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 4, pp: 413-422, 2024, doi: <https://doi.org/10.57152/malcom.v4i2.1210>.
- [7] H. Priyono, R. Sari, dan T. Mardiana, “Klasifikasi Pemilihan Jurusan Sekolah Menengah Kejuruan Menggunakan *Gradient Boosting Classifier*,” *Jurnal Informatika*, vol. 9, no. 2, Halaman 131-139, 2022.
- [8] J. Brandt, dan E. Lanzén, “*A Comparative Review of SMOTE and ADASYN in Imbalanced Data Classification*,” *Department of Statistics Uppsala University*, 2020.

- [9] G. Husain, et al., “*SMOTE vs. SMOTEENN: A study on the performance of resampling algorithms for addressing class imbalance in regression models*,” *Algorithms*, 18(1), 2025, doi: <https://doi.org/10.3390/a18010037>
- [10] D.S. Sumardilah, dan A. Rahmadi, “Risiko *Stunting* Anak Baduta (7-24 bulan,” *Jurnal Kesehatan*, Volume 10, Nomor 1, 2019.
- [11] S. Ndagijimana, I.H. Kabano, E. Masabo, dan J.M. Ntaganda, “*Prediction of Stunting Among Under-5 Children in Rwanda Using Machine Learning Techniques*,” *Journal of Preventive Medicine & Public Health*, 56:41-49, 2023, doi: <https://doi.org/10.3961/jpmph.22.388>.
- [12] A.I. Putri, Y. Syarif, P. Jayadi, F. Arrazak, dan F.N. Salisah, “Implementasi Algoritma *Decision Tree* dan *Support Vector Machine (SVM)* untuk Prediksi Risiko *Stunting* pada Keluarga,” *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, Vol. 3, pp: 349-357, 2023, doi: <https://doi.org/10.57152/malcom.v3i2.1228>.
- [13] M.M.H Khorshid, T.H.M Abou-El-Enien, dan G.M.A Soliman, “*A Comparison among Support Vector Machine and other Machine Learning Classification Algorithms*,” *IPASJ International Journal of Computer Science (IIJCS)*, Volume 3, Issue 5, 2015.
- [14] J. Burdack, F. Horst, S. Giesselbach, I. Hassan, S. Daffner, dan W.I Schollhorn, “*Systematic Comparison of the Influence of Different Data Preprocessing Methods on the Performance of Gait Classifications Using Machine Learning*,” *Front. Bioeng. Biotechnol.* 8:260, 2020, doi: [10.3389/fbioe.2020.00260](https://doi.org/10.3389/fbioe.2020.00260).
- [15] *WHO Multicentre Growth Reference Study Group*, “*WHO Child Growth Standards: Length/height-for-age, weight-for-age, weight-for-length, weight-for-height and body mass index-for-age: Methods and development*”. Geneva: *World Health Organization*, pp 302, 2006.
- [16] Indriyanti, N. Ichsan, H. Fatah, T. Wahyuni, dan E. Ermawati, “Implementasi *Orange Data Mining* untuk Prediksi Harga *Bitcoin*,” *Jurnal Responsif*, Vol. 4 No.2, pp. 118-125, 2022.

- [17] I.S. Al-Mejibli¹, J.K Alwan, dan H.A Dhafar, “*The effect of gamma value on support vector machine performance with different kernels,*” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 5, pp. 5497-5506, 2020, doi: 10.11591/ijece.v10i5.pp5497-5506.
- [18] I.A. Muis, dan M. Affandes, “Penerapan Metode *Support Vector Machine (SVM)* Menggunakan *Kernel Radial Basis Function (RBF)* Pada Klasifikasi Tweet,” *Jurnal Sains, Teknologi dan Industri*, vol. 12, no. 2, pp.189–197, 2015.
- [19] A. Natekin, dan A. Knoll, “*Gradient Boosting machines, a tutorial,*” *Frontiers in Neurorobotics*, volume 7, article 21, 2013, doi: 10.3389/fnbot.2013.00021.
- [20] Z. Zhang, Y. Zhao, A. Canes, D. Steinberg, dan O. Lyshevska, “*Predictive analytics with Gradient Boosting in clinical medicine,*” *Annals of Translational Medicine*, 7(7):152, 2019, doi: 10.21037/atm.2019.03.29.
- [21] T. Sugihartono, B Wijaya, Marini, A.F Alkayes, dan H.A. Anugrah, “*Optimizing Stunting Detection through SMOTE and Machine Learning: a Comparative Study of XGBoost, Random Forest, SVM, and K-NN,*” *Journal of Applied Data Sciences*, Vol. 6, No. 1, pp. 667–682, 2025, doi: <https://doi.org/10.47738/jads.v6i1.494>.

LAMPIRAN

Lampiran 1: Preprocessing

```
# Load dataset
file_path = '(MENTAH) Kab Bekasi berdasarkan BBLA PBLA Februari 2024
(2255).xlsx'
df = pd.read_excel(file_path)

# Drop unnecessary columns
df.drop(['No', 'NIK', 'Nama', 'Tgl Lahir', 'Nama Ortu', 'Prov', 'Kab/Kota',
'Kec', 'Pukesmas', 'Desa/Kel', 'Posyandu', 'RT', 'RW', 'Alamat', 'Tanggal
Pengukuran', 'LiLA', 'Naik Berat Badan', 'PMT Diterima (kg)', 'Jml Vit A',
'KPSP', 'KIA', 'ZS BB/U', 'ZS TB/U', 'ZS BB/TB', 'BB/U', 'BB/TB'], axis=1,
inplace=True)
# Remove data with "0 Tahun - 0 Bulan - 0 Hari" in the 'Usia Saat Ukur' column
df = df[df['Usia Saat Ukur'] != '0 Tahun - 0 Bulan - 0 Hari']
df.index = range(1, len(df) + 1)

# Convert 'Usia Saat Ukur' to months
def convert_to_months(age_str):
    pattern = r'(\d+)\s*Tahun\s*-\s*(\d+)\s*Bulan\s*-\s*(\d+)\s*Hari'
    match = re.match(pattern, age_str)
    if match:
        years = int(match.group(1))
        months = int(match.group(2))
        total_months = years * 12 + months
        return total_months
    return None
df['Usia Saat Ukur'] = df['Usia Saat Ukur'].apply(convert_to_months)

# Check missing value and data type
missing_values = df.isnull().sum()
missing_values = missing_values[missing_values > 0]
print(missing_values) if not missing_values.empty else print("Tidak ada
missing values.")
print(df.dtypes)

# Encode gender (JK) and normalize 'TB/U'
df['JK'] = df['JK'].str.strip().map({'L': 0, 'P': 1})
df['TB/U'] = df['TB/U'].map({'Tinggi': 0, 'Normal': 0, 'Pendek': 1, 'Sangat
Pendek': 1})
# Handle missing values and outliers
df = df.dropna(subset=['TB/U', 'BB Lahir', 'TB Lahir'])
df['TB/U'] = df['TB/U'].astype(int)
# Remove any unwanted non-numeric characters
df['BB Lahir'] = df['BB Lahir'].replace({r'\s+': ''}, regex=True)
df['TB Lahir'] = df['TB Lahir'].replace({r'\s+': ''}, regex=True)
df['BB Lahir'] = pd.to_numeric(df['BB Lahir'], errors='coerce').round(1)
df['TB Lahir'] = pd.to_numeric(df['TB Lahir'], errors='coerce').astype(int)
print(df.isnull().sum())
```

```

# Menghitung jumlah data 0 dan 1 pada kolom 'TB/U'
class_counts = df['TB/U'].value_counts()
# Menampilkan jumlah data dengan label 0 dan 1
print(f"Jumlah data dengan label 0 (Tinggi/Normal): {class_counts[0]}")
print(f"Jumlah data dengan label 1 (Pendek/Sangat Pendek): {class_counts[1]}")

features = ['Usia Saat Ukur', 'BB Lahir', 'TB Lahir', 'Berat Saat Ukur',
'Tinggi Saat Ukur']
plt.figure(figsize=(8, 10))
for i, feature in enumerate(features, 1):
    ax = plt.subplot(2, 3, i)
    sns.boxplot(y=df[feature], linewidth=1.5, color='grey',
boxprops=dict(edgecolor='grey', facecolor='none'), ax=ax)
    ax.set_title(feature)
    ax.set_ylabel('')
plt.tight_layout()
plt.subplots_adjust(wspace=0.3)
plt.show()

# Handle outliers and normalizing data
def handle_outliers_with_iqr(df, features):
    for feature in features:
        Q1 = df[feature].quantile(0.25)
        Q3 = df[feature].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.8 * IQR
        upper_bound = Q3 + 1.8 * IQR
        median = df[feature].median()
        df[feature] = df[feature].apply(lambda x: median if x < lower_bound or
x > upper_bound else x)
    return df
def apply_winsorization(df, features, limits=[0.05, 0.075]):
    for feature in features:
        df[feature] = winsorize(df[feature], limits=limits)
    return df
def scale_features(df, features):
    scaler = MinMaxScaler()
    df[features] = scaler.fit_transform(df[features])
    return df

# Visualize boxplots before and after handling outliers
def plot_boxplots(df, features):
    plt.figure(figsize=(8, 10))
    for i, feature in enumerate(features, 1):
        ax = plt.subplot(2, 3, i)
        sns.boxplot(y=df[feature], linewidth=1.5, color='grey',
                    boxprops=dict(edgecolor='grey', facecolor='none'), ax=ax)
        ax.set_title(feature)
        ax.set_ylabel('')

```

```
lt.tight_layout()
plt.subplots_adjust(wspace=0.3)
plt.show()

# List of features to handle
features = ['Usia Saat Ukur', 'BB Lahir', 'TB Lahir', 'Berat Saat Ukur',
'Tinggi Saat Ukur']

# Handle outliers, winsorization, and normalization
df = handle_outliers_with_iqr(df, features)
df = apply_winsorization(df, features)
df = scale_features(df, features)

# Visualize boxplots
plot_boxplots(df, features)
```

Lampiran 2: Pembagian Data dan *Oversampling* (SMOTE-ENN)

```
print("Jumlah per kelas pada Data Sebelum Pemisahan:")
print(df['TB/U'].value_counts())

# Select 100 positive and 100 negative samples for testing
positive_samples = df[df['TB/U'] == 1].sample(n=75, random_state=42)
negative_samples = df[df['TB/U'] == 0].sample(n=75, random_state=42)
data_testing = pd.concat([positive_samples, negative_samples])
# Pisahkan data testing dari data default
data_default = df.drop(data_testing.index)

print("\nJumlah per kelas pada Data Testing:")
print(data_testing['TB/U'].value_counts())
print("\nJumlah per kelas pada Data Trainig/Validasi:")
print(data_default['TB/U'].value_counts())

# Apply SMOTE to balance the classes in data_default
X_default = data_default.drop(columns=['TB/U']) # Fitur dari data default
y_default = data_default['TB/U'] # Target dari data default
# Terapkan SMOTE untuk oversampling kelas minoritas
smote = SMOTEENN(random_state=42)
X_default_resampled, y_default_resampled = smote.fit_resample(X_default,
y_default)
# Lihat distribusi kelas setelah SMOTE
print("Distribusi kelas setelah SMOTE (Train):")
print(y_default_resampled.value_counts())

# Split data_default_resampled menjadi training dan testing dengan rasio 80:20
X_train_80, X_val_80, y_train_80, y_val_80 =
train_test_split(X_default_resampled, y_default_resampled, test_size=0.2,
random_state=42)
# Split data_default_resampled menjadi training dan testing dengan rasio 70:30
X_train_70, X_val_70, y_train_70, y_val_70 =
train_test_split(X_default_resampled, y_default_resampled, test_size=0.3,
random_state=42)
# Split data_default_resampled menjadi training dan testing dengan rasio 60:40
X_train_60, X_val_60, y_train_60, y_val_60 =
train_test_split(X_default_resampled, y_default_resampled, test_size=0.4,
random_state=42)
```

Lampiran 3: Fungsi Evaluasi dan Visualisasi Model

```
# Fungsi untuk menampilkan metrik evaluasi
def display_evaluation_metrics_without_support(model_name, y_true, y_pred):
    accuracy = accuracy_score(y_true, y_pred)
    precision, recall, f1, _ = precision_recall_fscore_support(
        y_true, y_pred, average='macro')
    print(f"\nModel: {model_name}")
    print(f"Accuracy\t: {accuracy*100:.2f}%")
    print(f"Precision\t: {precision*100:.2f}% (macro average)")
    print(f"Recall\t\t: {recall*100:.2f}% (macro average)")
    print(f"F1-Score\t: {f1*100:.2f}% (macro average)")
    # Confusion matrix
    cm = confusion_matrix(y_true, y_pred)
    class_labels = sorted(set(y_true))
    n_samples = len(y_true)
    # Classification report
    report = classification_report(y_true, y_pred, output_dict=True)
    print("\nMetrik Per Kelas:")
    for idx, label in enumerate(class_labels):
        label_str = str(label)
        # Elemen confusion matrix untuk kelas ini
        TP = cm[idx, idx] # True Positives
        FN = cm[idx, :].sum() - TP
        FP = cm[:, idx].sum() - TP
        TN = n_samples - (TP + FN + FP)
        true_class_accuracy = TP / (TP + FP + FN) if (TP + FP + FN) > 0 else 0
        print(f"Class {label_str}:")
        print(f"  Akurasi      : {true_class_accuracy*100:.2f}%")
        print(f"  Precision    : {report[label_str]['precision']*100:.2f}%")
        print(f"  Recall       : {report[label_str]['recall']*100:.2f}%")
        print(f"  F1-Score     : {report[label_str]['f1-score']*100:.2f}%")
    return accuracy, precision, recall, f1, cm

# Fungsi untuk menggambar confusion matrix
def plot_confusion_matrix(y_true, y_pred, model_name):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(6,6))
    sns.heatmap(cm, annot=True, fmt='g', cmap='Blues', xticklabels=['Class 0',
'Class 1'], yticklabels=['Class 0', 'Class 1'])
    plt.title(f"Confusion Matrix for {model_name}")
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()
    return cm

# fungsi untuk menggambar ROC curve
def plot_combined_roc_curve(models, X_test, y_test):
    plt.figure(figsize=(8, 6))
    for model_name, model in models.items():
        # Get the predicted probabilities for the positive class
        y_pred_prob = model.predict_proba(X_test)[:, 1]
```

```
# Compute ROC curve and AUC
fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)
# Plot ROC curve
plt.plot(fpr, tpr, lw=2, label=f'{model_name}
        (AUC = {roc_auc:.2f})')
# Plot the diagonal line (chance level)
plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.title('Combined ROC Curve')
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```

Lampiran 4: Pelatihan Model dan Evaluasi Model

```
# Gradient Boosting (80:20 split)
gb_80_20 = GradientBoostingClassifier(learning_rate=0.05, max_depth=3,
n_estimators=50, random_state=42)
weights_80 = compute_sample_weight(class_weight='balanced', y=y_train_80)
gb_80_20.fit(X_train_80, y_train_80, sample_weight=weights_80)
gb_pred_80 = gb_80_20.predict(X_val_80)
# Gradient Boosting (70:30 split)
gb_70_30 = GradientBoostingClassifier(learning_rate=0.05, max_depth=3,
n_estimators=50, random_state=42)
weights_70 = compute_sample_weight(class_weight='balanced', y=y_train_70)
gb_70_30.fit(X_train_70, y_train_70, sample_weight=weights_70)
gb_pred_70 = gb_70_30.predict(X_val_70)
# Gradient Boosting (60:40 split)
gb_60_40 = GradientBoostingClassifier(learning_rate=0.05, max_depth=3,
n_estimators=50, random_state=42)
weights_60 = compute_sample_weight(class_weight='balanced', y=y_train_60)
gb_60_40.fit(X_train_60, y_train_60, sample_weight=weights_60)
gb_pred_60 = gb_60_40.predict(X_val_60)
# Evaluasi Gradient Boosting
print("\nEvaluating Gradient Boosting (80:20)")
accuracy_80, precision_80, recall_80, f1_80, cm_80 =
display_evaluation_metrics_without_support("Gradient Boosting (80:20)",
y_val_80, gb_pred_80)
print("\nEvaluating Gradient Boosting (70:30)")
accuracy_70, precision_70, recall_70, f1_70, cm_70 =
display_evaluation_metrics_without_support("Gradient Boosting (70:30)",
y_val_70, gb_pred_70)
print("\nEvaluating Gradient Boosting (60:40)")
accuracy_60, precision_60, recall_60, f1_60, cm_60 =
display_evaluation_metrics_without_support("Gradient Boosting (60:40)",
y_val_60, gb_pred_60)
plot_confusion_matrix(y_val_80, gb_pred_80, "Gradient Boosting (80:20)")
plot_confusion_matrix(y_val_70, gb_pred_70, "Gradient Boosting (70:30)")
plot_confusion_matrix(y_val_60, gb_pred_60, "Gradient Boosting (60:40)")

# SVM (80:20 split)
svm_80_20 = SVC(C=1.0, kernel='rbf', gamma='scale', probability=True,
random_state=42, class_weight='balanced')
svm_80_20.fit(X_train_80, y_train_80)
svm_pred_80 = svm_80_20.predict(X_val_80)
# SVM (70:30 split)
svm_70_30 = SVC(C=1.0, kernel='rbf', gamma='scale', probability=True,
random_state=42, class_weight='balanced')
svm_70_30.fit(X_train_70, y_train_70)
svm_pred_70 = svm_70_30.predict(X_val_70)
# SVM (60:40 split)
svm_60_40 = SVC(C=1.0, kernel='rbf', gamma='scale', probability=True,
random_state=42, class_weight='balanced')
```

```

svm_60_40.fit(X_train_60, y_train_60)
svm_pred_60 = svm_60_40.predict(X_val_60)
# Evaluasi SVM
print("\nEvaluating SVM (80:20)")
accuracy_svm_80, precision_svm_80, recall_svm_80, f1_svm_80, cm_svm_80 =
display_evaluation_metrics_without_support("SVM (80:20)", y_val_80,
svm_pred_80)
print("\nEvaluating SVM (70:30)")
accuracy_svm_70, precision_svm_70, recall_svm_70, f1_svm_70, cm_svm_70 =
display_evaluation_metrics_without_support("SVM (70:30)", y_val_70,
svm_pred_70)
print("\nEvaluating SVM (60:40)")
accuracy_svm_60, precision_svm_60, recall_svm_60, f1_svm_60, cm_svm_60 =
display_evaluation_metrics_without_support("SVM (60:40)", y_val_60,
svm_pred_60)
plot_confusion_matrix(y_val_80, svm_pred_80, "SVM (80:20)")
plot_confusion_matrix(y_val_70, svm_pred_70, "SVM (70:30)")
plot_confusion_matrix(y_val_60, svm_pred_60, "SVM (60:40)")

# SVM terbaik (60:40)
svm_best = SVC(C=1.0, kernel='rbf', gamma='scale', probability=True,
class_weight='balanced', random_state=42)
svm_best.fit(X_train_60, y_train_60)
# Gradient Boosting terbaik (80:20)
gb_best = GradientBoostingClassifier(learning_rate=0.05, max_depth=3,
n_estimators=50, random_state=42)
gb_best.fit(X_train_80, y_train_80, sample_weight=weights_80)
# Pisahkan fitur dan label dari data_testing
X_test_final = data_testing.drop(columns=['TB/U'])
y_test_final = data_testing['TB/U']
# Prediksi dan evaluasi
svm_pred_final = svm_best.predict(X_test_final)
gb_pred_final = gb_best.predict(X_test_final)
# Evaluasi akhir
print("\nEvaluasi Akhir: SVM (Terbaik - 60:40)")
display_evaluation_metrics_without_support("SVM (Final Testing)",
y_test_final, svm_pred_final)
print("\nEvaluasi Akhir: Gradient Boosting (Terbaik - 80:20)")
display_evaluation_metrics_without_support("Gradient Boosting (Final
Testing)", y_test_final, gb_pred_final)
plot_confusion_matrix(y_test_final, svm_pred_final, "SVM (Final Testing)")
plot_confusion_matrix(y_test_final, gb_pred_final, "Gradient Boosting (Final
Testing)")

# Dataframe untuk plotting
data_plot = {
    'Metrics': ['Accuracy', 'Precision', 'Recall', 'F1-Score'], 'SVM 80:20':
[accuracy_svm_80, precision_svm_80, recall_svm_80, f1_svm_80], 'SVM 70:30':

```



```

[accuracy_svm_70, precision_svm_70, recall_svm_70, f1_svm_70], 'SVM 60:40':
[accuracy_svm_60, precision_svm_60, recall_svm_60, f1_svm_60], 'GBoost 80:20':
[accuracy_80, precision_80, recall_80, f1_80], 'GBoost 70:30': [accuracy_70,
precision_70, recall_70, f1_70], 'GBoost 60:40': [accuracy_60, precision_60,
recall_60, f1_60]]
df_plot = pd.DataFrame(data_plot)
for col in df_plot.columns[1:]:
    df_plot[col] = df_plot[col] * 100
x = np.arange(len(df_plot['Metrics']))
width = 0.15
fig, ax = plt.subplots(figsize=(12, 6))
ax.bar(x - 2.5*width, df_plot['SVM 80:20'], width, label='SVM 80:20')
ax.bar(x - 1.5*width, df_plot['SVM 70:30'], width, label='SVM 70:30')
ax.bar(x - 0.5*width, df_plot['SVM 60:40'], width, label='SVM 60:40')
ax.bar(x + 0.5*width, df_plot['GBoost 80:20'], width, label='G.Boost 80:20')
ax.bar(x + 1.5*width, df_plot['GBoost 70:30'], width, label='G.Boost 70:30')
ax.bar(x + 2.5*width, df_plot['GBoost 60:40'], width, label='G.Boost 60:40')
for y in range(85, 97):
    ax.axhline(y=y, color='gray', linestyle='--', linewidth=0.4)
ax.set_xlabel('Metrics')
ax.set_ylabel('Percentage (%)')
ax.set_title('Perbandingan Kinerja Model: SVM vs Gradient Boosting (Berbagai
Rasio Split)')
ax.set_xticks(x)
ax.set_xticklabels(df_plot['Metrics'])
ax.set_ylim(85, 97)
ax.legend(loc='upper right', fontsize=9, frameon=True)
plt.tight_layout()
plt.show()

# Data evaluasi akhir (dari data testing)
data_final_plot = {'Metrics': ['Accuracy', 'Precision', 'Recall', 'F1-Score'],
'SVM (60:40)': [84.67, 84.82, 84.67, 84.65], 'G.Boost (80:20)': [78.67, 78.75,
78.67, 78.65]}
df_final_plot = pd.DataFrame(data_final_plot)
x = np.arange(len(df_final_plot['Metrics']))
width = 0.37
fig, ax = plt.subplots(figsize=(10, 6))
ax.bar(x - width/2, df_final_plot['SVM (60:40)'], width, label='SVM (60:40)',
color='#4c72b0')
ax.bar(x + width/2, df_final_plot['G.Boost (80:20)'], width, label='G.Boost
(80:20)', color='#dd8452')
for y in range(70, 91, 5):
    ax.axhline(y=y, color='gray', linestyle='--', linewidth=0.4)
ax.set_xlabel('Metrics')
ax.set_ylabel('Percentage (%)')
ax.set_title('Perbandingan Evaluasi Akhir Model: SVM (60:40) vs Gradient
Boosting (80:20)')

```

```

ax.set_xticks(x)
ax.set_xticklabels(df_final_plot['Metrics'])
ax.set_ylim(70, 86)
ax.legend(loc='upper right', fontsize=9, frameon=True)
plt.tight_layout()
plt.show()

# ROC Curve (Set Validasi)
all_models = {"GB (80:20)": (gb_80_20, X_val_80, y_val_80), "SVM (80:20)":
(svm_80_20, X_val_80, y_val_80), "GB (70:30)": (gb_70_30, X_val_70, y_val_70),
"SVM (70:30)": (svm_70_30, X_val_70, y_val_70), "GB (60:40)": (gb_60_40,
X_val_60, y_val_60), "SVM (60:40)": (svm_60_40, X_val_60, y_val_60),}
plt.figure(figsize=(16, 12))
for model_name, (model, X_test, y_test) in all_models.items():
    if hasattr(model, "predict_proba"):
        y_pred_proba = model.predict_proba(X_test)[: , 1]
    else:
        y_pred_proba = model.decision_function(X_test)
    # Calculate ROC curve
    fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
    auc_score = roc_auc_score(y_test, y_pred_proba)
    plt.plot(fpr, tpr, linewidth=3, label=f'{model_name} (AUC =
{auc_score:.3f})')
plt.plot([0, 1], [0, 1], 'k--', alpha=0.6, label='Pengklasifikasi Acak')
plt.xlim([0.0, 0.6])
plt.ylim([0.2, 1.05])
plt.xlabel('Tingkat Positif Palsu (False Positive Rate)', fontsize=14)
plt.ylabel('Tingkat Positif Benar (True Positive Rate)', fontsize=14)
plt.title('Perbandingan Kurva ROC - Semua Model dan Pembagian Data',
fontsize=16, fontweight='bold')
plt.legend(loc="lower right", fontsize=16, frameon=True, fancybox=True,
shadow=True)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()

# ROC Curve (Set Validasi)
all_models = {"GB (Best Final)": (gb_best, X_test_final, y_test_final), "SVM
(Best Final)": (svm_best, X_test_final, y_test_final),}
plt.figure(figsize=(16, 12))
for model_name, (model, X_test, y_test) in all_models.items():
    # Get predictions
    if hasattr(model, "predict_proba"):
        y_pred_proba = model.predict_proba(X_test)[: , 1]
    else:
        y_pred_proba = model.decision_function(X_test)
    # Calculate ROC curve
    fpr, tpr, _ = roc_curve(y_test, y_pred_proba)

```

```

    auc_score = roc_auc_score(y_test, y_pred_proba)
    plt.plot(fpr, tpr, linewidth=3, label=f'{model_name} (AUC =
{auc_score:.3f})')
    plt.plot([0, 1], [0, 1], 'k--', alpha=0.6, label='Pengklasifikasi Acak')
    plt.xlim([0.0, 0.7])
    plt.ylim([0.0, 1.05])
    plt.xlabel('Tingkat Positif Palsu (False Positive Rate)', fontsize=14)
    plt.ylabel('Tingkat Positif Benar (True Positive Rate)', fontsize=14)
    plt.title('Perbandingan Kurva ROC - Semua Model dan Pembagian Data',
    fontsize=16, fontweight='bold')
    plt.legend(loc="lower right", fontsize=16, frameon=True, fancybox=True,
    shadow=True)
    plt.grid(True, alpha=0.3)
    plt.tight_layout()
    plt.show()

```