OPTIMASI *RANDOM FOREST* DENGAN *GA* DAN *RFE* PADA *HIGH DIMENSIONAL* DATA STUNTING

SKRIPSI

Diajukan Oleh :
Bima Satria
2011102441102

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR
JULI 2024

OPTIMASI *RANDOM FOREST* DENGAN *GA* DAN *RFE* PADA *HIGH DIMENSIONAL* DATA STUNTING

SKRIPSI

Diajukan Sebagai Salah Satu Persyaratan Untuk Memperoleh Gelar
Sarjana Komputer

Diajukan Oleh :
Bima Satria
2011102441102

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR
JULI 2024

LEMBAR PERSETUJUAN

OPTIMASI *RANDOM FOREST* DENGAN *GA* DAN *RFE* PADA *HIGH DIMENSIONAL* DATA STUNTING

SKRIPSI

Diajukan oleh:

Bima Satria
2011102441102

Disetujui untuk diujikan Pada tanggal 4. Juli 2024

Pembimbing

28/6/2024

Taghfirul Azhima Yoga Siswa, S.Kom, M.Kom
NIDN. 1118038805

Mengetahui,
Koordinator Skripsi

Abdul Rahim, S.Kom, M.Cs.
NIDN. 1115039601

iii

LEMBAR PENGESAHAN

OPTIMASI *RANDOM FOREST* DENGAN *GA* DAN *RFE* PADA *HIGH DIMENSIONAL* DATA STUNTING

SKRIPSI

Diajukan oleh:

Bima Satria
2011102441102

Diseminarkan dan Diujikan
Pada tanggal 4.. Juli 2024

Penguji I	Penguji II
Wawan Joko Pranoto, S.Kom, M.Ti	**Taghfirul Azhima Yoga Siswa, S.Kom, M.Kom**
NIDN. 1102057701	**NIDN. 1118038805**

Mengetahui,

Ketua

Program Studi Teknik Informatika

Arbansyah, S.Kom, M. TI
NIDN.1118019203

iv

PERNYATAAN KEASLIAN PENELITIAN

Saya yang bertanda tangan di bawah ini:

Nama	: Bima Satria
NIM	: 2011102441102
Program Studi	: Teknik Informatika
Judul Penelitian	: Optimasi *Random Forest* Dengan *GA* Dan *RFE* Pada *High Dimensional* Data Stunting

Menyatakan bahwa skripsi yang saya tulis ini benar-benar hasil karya saya sendiri, dan bukan merupakan hasil plagiasi/falsifikasi/fabrikasi baik sebagian atau seluruhnya.

Atas pernyataan ini, saya siap menanggung risiko atau sanksi yang dijatuhkan kepada saya apabila kemudian ditemukan adanya pelanggaran terhadap etika keilmuan dalam skripsi saya ini, atau klaim dari pihak lain terhadap keaslian karya sayaini

Samarinda, 27 April 2024
Yang membuat pernyataan

Bima Satria
NIM: 2011102441102

v

ABSTRAK

Stunting adalah masalah kekurangan gizi kronis yang mengakibatkan terganggunya pertumbuhan anak, dengan dampak jangka panjang pada pertumbuhan fisik, perkembangan kognitif, dan produktivitas di masa dewasa. Di Indonesia, prevalensi stunting masih di atas batas WHO, mencapai 24,4% berdasarkan Studi Status Gizi Indonesia (SSGI) 2021, dan di Kota Samarinda prevalensinya mencapai 24,7% pada tahun 2021 dengan 1.402 balita teridentifikasi mengalami stunting. Mengatasi masalah ini memerlukan pendekatan berbasis data yang lebih terstruktur untuk memberikan intervensi yang tepat sasaran. Penelitian ini menggunakan data dari Dinas Kesehatan Kota Samarinda, mencakup 150.474 data stunting, dan melibatkan pengumpulan data, pembersihan data, seleksi fitur, serta penerapan model klasifikasi. Penelitian ini bertujuan untuk meningkatkan akurasi klasifikasi data stunting di Kota Samarinda tahun 2023 menggunakan algoritma Random Forest yang disempurnakan melalui teknik seleksi fitur Recursive Feature Elimination (RFE) dan optimasi Genetic Algorithm (GA). Hasil seleksi fitur menggunakan RFE menunjukkan bahwa fitur yang paling berpengaruh adalah Berat, ZS TB/U, ZS BB/U, dan BB/U. Penerapan RFE meningkatkan rata-rata akurasi model dari 91.91% menjadi 93.64%, sementara optimasi dengan GA meningkatkan rata-rata akurasi menjadi 98.39%. Akurasi pasti meningkat dari 94.23% (model dasar) menjadi 97.10% (dengan RFE) dan mencapai 99.70% (dengan RFE dan GA). Kombinasi RFE dan GA terbukti efektif dalam mengatasi kompleksitas data dan meningkatkan keandalan prediksi stunting. Penelitian ini memberikan kontribusi signifikan pada pengembangan teknik machine learning untuk analisis data berdimensi tinggi dalam bidang kesehatan, dan diharapkan menjadi dasar bagi program intervensi yang lebih efektif dalam menangani masalah stunting di Indonesia.

Kata Kunci : *Random Forest, Recursive Feature Elimination, Genetic Algorithm, Klasifikasi, High Dimensional*

ABSTRACT

Stunting is a chronic malnutrition problem that disrupts children's growth, with long-term impacts on physical growth, cognitive development, and productivity in adulthood. In Indonesia, the prevalence of stunting is still above the WHO threshold, reaching 24.4% according to the 2021 Indonesian Nutritional Status Study (SSGI), and in Samarinda City, the prevalence reached 24.7% in 2021 with 1,402 toddlers identified as stunted. Addressing this problem requires a more structured data-driven approach to provide targeted interventions. This study uses data from the Samarinda City Health Office, encompassing 150,474 stunting data points, and involves data collection, data cleaning, feature selection, and classification model application. This study aims to improve the accuracy of stunting data classification in Samarinda City in 2023 using the Random Forest algorithm enhanced with Recursive Feature Elimination (RFE) feature selection techniques and Genetic Algorithm (GA) optimization. The feature selection results using RFE show that the most influential features are Weight, ZS TB/U, ZS BB/U, and BB/U. The application of RFE increased the model's average accuracy from 91.91% to 93.64%, while GA optimization further increased the average accuracy to 98.39%. The definite accuracy increased from 94.23% (baseline model) to 97.10% (with RFE) and reached 99.70% (with RFE and GA). The combination of RFE and GA has proven effective in tackling data complexity and improving the reliability of stunting predictions. This study significantly contributes to the development of machine learning techniques for high-dimensional data analysis in health and is expected to be the foundation for more effective intervention programs in addressing stunting issues in Indonesia.

Keywords: *Random Forest, Recursive Feature Elimination, Genetic Algorithm, Classification, High Dimensional*

PRAKATA

Puji dan syukur kehadirat Tuhan Yang Maha Esa, karena dengan rahmat dan karunia-Nya, saya dapat menyelesaikan skripsi ini sebagai salah satu syarat untuk menyelesaikan program Sarjana di Fakultas Sains Dan Teknologi, Universitas Muhammadiyah Kalimantan Timur. Skripsi ini berjudul "OPTIMASI *RANDOM FOREST* DENGAN *GA* DAN *RFE* PADA *HIGH DIMENSIONAL* DATA STUNTING".

Penyusunan Skripsi ini tidak terlepas dari bantuan berbagai pihak, yang dengan tulus dan penuh kesabaran membimbing, membantu, serta memberikan motivasi kepada saya. Oleh karena itu, pada kesempatan ini, saya ingin menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Mamaku tercinta beliau bernama Yusimayati dan keluarga yang selalu memberikan doa, dukungan moral, dan motivasi tanpa henti.
2. Bapak Taghfirul Azhima Yoga Siswa, S.Kom, M.Kom, selaku pembimbing utama, atas bimbingan, arahan, kritik, dan saran yang telah diberikan selama proses penyusunan proposal ini.
3. Bapak Wawan Joko Pranoto, S.Kom, M.Ti, selaku penguji utama, atas masukan, kritik, dan sarannya selama proses penyusunan skripsi ini.
4. Teman-teman Satu RTA yang selalu memberikan semangat dan dukungan selama proses penyusunan proposal ini.
5. Muhammad Ghalib Ramadhan, S.Kom dan Ihsan Magribi, S.Kom yang selalu membantu saya mengerjakan skripsi ini hingga selesai.

Saya menyadari bahwa proposal skripsi ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat saya harapkan untuk perbaikan. Semoga proposal skripsi ini dapat bermanfaat bagi pengembangan ilmu pengetahuan, khususnya di bidang Informatika dan bagi siapa saja yang berkepentingan.

Samarinda, 27 April 2024
Penyusun,

Bima Satria
NIM: 2011102441102

DAFTAR ISI

Halaman

DAFTAR TABEL

Tabel **Halaman**

DAFTAR GAMBAR

Gambar **Halaman**

DAFTAR LAMPIRAN

Lampiran **Halaman**

BAB I

PENDAHULUAN

1.1 Latar Belakang

Penelitian tentang stunting sangat penting karena stunting memiliki dampak jangka panjang pada pertumbuhan dan perkembangan anak, termasuk penurunan kemampuan kognitif dan fisik yang bisa mempengaruhi produktivitas mereka di masa dewasa. Stunting adalah masalah kekurangan gizi akibat asupan gizi yang tidak mencukupi dalam jangka waktu panjang, yang mengakibatkan gangguan pertumbuhan pada anak dengan tinggi badan di bawah standar usianya (Rufina et., al 2023). Stunting di Indonesia adalah masalah gizi yang mendesak dan membutuhkan penanganan khusus. Meskipun prevalensi stunting di Indonesia telah menurun setiap tahun, angkanya masih di atas batas WHO yaitu di bawah 20%. Berdasarkan Studi Status Gizi Indonesia (SSGI) 2021, angka prevalensi stunting di Indonesia adalah 24,4% (Intan, 2023). Berdasarkan data dari Dinas Kesehatan Kota Samarinda tahun 2021, prevalensi stunting pada balita di Kota Samarinda mencapai 24,7%. Pada tahun 2020, tercatat sebanyak 1.402 balita di Kota Samarinda mengalami stunting, dengan rincian 403 balita dalam kategori sangat pendek dan 999 balita dalam kategori pendek (Rufina et., al 2023). Mengingat konsekuensi serius yang diakibatkan oleh stunting, tentu menjadi kekhawatiran tersendiri untuk masa depan, terutama karena banyaknya individu yang masih teridentifikasi dalam kategori stunting. Oleh karena itu, perlu adanya penggunaan teknik analisis data canggih seperti machine learning untuk membantu pengambilan keputusan dalam mendeteksi stunting di masa depan. Penggunaan teknik ini membuka peluang untuk menerapkan metode data mining dan klasifikasi yang lebih spesifik, yang akan meningkatkan akurasi dalam mengidentifikasi kasus stunting, serta memungkinkan intervensi yang lebih tepat sasaran dan efektif.

Dari penelitian yang telah dilakukan sebelumnya dalam bidang data mining, klasifikasi status stunting telah dijajaki dengan menggunakan beberapa pendekatan, seperti algoritma *Random Forest*, *Logistic Regression*, *Support Vector Classification*, dan lain-lain. Dan dalam beberapa tahun terakhir, penelitian telah menunjukkan tingkat akurasi yang cukup baik, dengan persentase akurasi sekitar 79.21%(RF), 44.71(LR), 64.02%(SVC) dan 63.85(XG Boost). Namun, penelitian terkait masih menggunakan data berdimensi rendah, yang memiliki atribut terbatas untuk analisis. Data dengan lima hingga delapan atribut atau bahkan kurang dari sepuluh atribut dapat dianggap sebagai data berdimensi rendah (Obvious et., al 2023). Data berdimensi rendah memiliki kelemahan dalam hal kemampuan untuk menangani kompleksitas dan variasi data yang besar. Hal ini dapat menyebabkan masalah seperti overfitting, di mana model terlalu spesifik terhadap data latih dan tidak dapat generalisasi dengan baik terhadap data baru. Selain itu, pengurangan dimensi yang terlalu drastis dapat mengakibatkan hilangnya informasi penting yang diperlukan untuk pemodelan yang akurat (Dylan et., al 2023). Berdasarkan Penelitian yang dilakukan Oleh (Estiyak et., al 2023) *Algoritma Random Forest(RF)*, *Naive Bayes(NB)*, dan *Logistic Regression(LR)* dapat mengalami penurunan performa ketika dihadapkan pada data dengan dimensi tinggi, yang dapat mempengaruhi akurasi yang diperoleh 60%, 58%, dan 58%.

Data berdimensi tinggi terdiri dari berbagai atribut atau fitur yang digunakan untuk analisis. Sebagai contoh, data dengan puluhan atau ratusan atribut dapat diklasifikasikan sebagai data berdimensi tinggi. Data berdimensi tinggi menghadapi beberapa tantangan utama, antara lain peningkatan kompleksitas model, risiko overfitting, dan kesulitan dalam visualisasi data (Jorg et,. al 2023). Algoritma yang mengalami penurunan performa saat menghadapi data berdimensi tinggi, yang berpengaruh pada akurasi klasifikasi. Pemahaman tentang data berdimensi tinggi dan penerapan

strategi yang tepat dalam analisisnya menjadi kunci dalam mengoptimalkan kinerja model dan meningkatkan akurasi prediksi stunting di Kota Samarinda(Yoga 2023).

Algoritma klasifikasi dalam penelitian ini akan menggunakan *Random Forest (RF)*, berdasarkan penelitian yang dilakukan oleh (Obvious et., al 2022; Yoga dan Wawan 2023; Khan et., al 2023; Estiyak et al., 2023; Addisalem., al 2023) Random Forest memiliki performa terbaik jika dibandingkan dengan algoritma *Support Vector Machine, Logistic Regression, Neural Network, Naïve Bayes, k-nearest neighbors (k-NN), eXtreme Gradient Boosting (Xg boost), Logistic Regression, C5.0,* dan *CART* terkait klasifikasi data stunting. Penggunaan seleksi fitur *Recursive Feature Elimination (RFE)* juga akan dilakukan untuk mengidentifikasi atribut-atribut yang relevan dalam klasifikasi. Berdasarkan Penelitian Leykun et., al(2024) metode *Recursive Feature Elimination (RFE)* terbukti dapat meningkatkan akurasi dari metode klasifikasi Random Forest dimana sebelum menggunakan *RFE* mendapatkan akurasi sebesar 72.41% kemudian, setelah dilakukan penerapan metode *RFE*, terjadi peningkatan signifikan dalam akurasi, di mana akurasi *RF* meningkat menjadi 80.1%. Penelitian lainnya dengan metode klasifikasi yang berbeda (Chandan and Priti 2021), *SVM* dan *LR* sebelum menggunakan Seleksi fitur *RFE* 94% dan 93%, Setelah menggunakannya mendapatkan akurasi yang lumayan tinggi 97% dan 98% yang mana lebih tinggi dibandingkan dengan seleksi fitur yang lain *SVM + Chi-Square* 96.50% - 96.60%, *K-NN + Relif-F* 91.90% - 92.20%, dan *K-NN + Backward Elimation* 95.17 - 90.23%(Yoga et., al 2021; Kemal et., al 2023; Syahrani dan Dwi, 2022).

Namun, setelah menggunakan seleksi fitur *Recursive Feature Elimination (RFE)* dan mencapai akurasi 80,01%(Leykun et al., 2024), akurasi yang didapat masih belum memuaskan. Oleh karena itu, diperlukan optimasi lebih lanjut menggunakan Metode *Genetic Algorithm (GA)* untuk meningkatkan akurasi tersebut. untuk mencari solusi optimal atau mendekati solusi optimal dalam masalah optimasi dengan meniru proses seleksi alamiah evolusi genetik(Parmonagan et al., 2020), Seperti pada penelitian (Parmonagan et al.,2020) dimana sebelum menggunakannya mendapatkan Akurasi 82.5% setelah diterapkan mendapatkan 85.83% begitu juga dengan penelitian (Veeramani et al., 2023) Sebelum menggunakan mendapat 77.58% setelah diterapkan menjadi 79.31%, yang mana juga lebih baik dibandingkan dengan Optimasi yang lain dengan akurasi *PSO + RNN* 96.08%, *PSO + K-NN* 98.9%, *EGA + PSO* 98.97%, *PSO + RF* 99.76% dan *GA + RF* 99.99%(Monire et., al 2021).

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijabarkan, rumusan masalah pada penelitian ini adalah:

a) Apa saja fitur yang memiliki dampak signifikan terhadap kinerja algoritma *Random Forest* pada dataset Stunting di Kota Samarinda, dengan penerapan seleksi fitur *RFE* ?

b) Seberapa besar peningkatan tingkat keakuratan yang dicapai oleh algoritma *Random Forest* dalam mengklasifikasikan data stunting di Kota Samarinda setelah implementasi seleksi fitur *RFE* dan optimasi dengan *GA*?

1.3 Tujuan Penelitian

Berdasarkan latar belakang yang telah dijabarkan, rumusan masalah pada penelitian ini adalah:

a) Menentukan atribut-atribut yang Mempengaruhi Keakuratan Model *Random Forest* dalam Menganalisis Data Stunting di Kota Samarinda.

b) Menerapkan Algoritma *Random Forest* untuk Klasifikasi Kasus Stunting di Kota Samarinda, dengan menggunakan *RFE* untuk seleksi fitur dan mengoptimalkan model menggunakan *GA* guna mencapai akurasi prediksi yang lebih tinggi.

c) Mengevaluasi kinerja Algoritma *Random Forset* dengan seleksi fitur *RFE* dan optimasi *GA* menggunakan *Confusion Matrix.*

1.4 Manfaat Penelitian

Penelitian ini diharapkan memberikan beberapa manfaat signifikan dalam bidang machine learning dan data mining. Berikut adalah beberapa manfaat dari penelitian ini:

a) Manfaat Akademis

Penelitian ini memberikan kontribusi pada pengembangan dan pemahaman lebih lanjut tentang penggunaan algoritma *Random Forest* dalam klasifikasi data berdimensi tinggi, khususnya dalam konteks analisis data kesehatan, menunjukkan cara-cara efektif untuk meningkatkan akurasi dan kinerja model machine learning melalui penerapan teknik *Recursive Feature Elimination (RFE)* dan optimasi menggunakan *Genetic Algorithm (GA),* serta memperkaya literatur ilmiah di bidang machine learning dengan menyediakan referensi berharga bagi akademisi dan praktisi yang tertarik dalam meningkatkan performa algoritma klasifikasi pada dataset kompleks dan berdimensi tinggi.

b) Manfaat Praktis

Penelitian ini menunjukkan penerapan praktis dari teknik machine learning canggih, seperti *RFE* dan *GA*, dalam meningkatkan akurasi model, memberikan panduan praktis bagi data scientist dan praktisi machine learning untuk menerapkan teknik-teknik tersebut pada berbagai jenis data. Menunjukkan bagaimana teknik seleksi fitur dapat digunakan untuk mengidentifikasi atribut yang paling relevan, membantu mengurangi kompleksitas model dan meningkatkan efisiensi pengolahan data tanpa mengorbankan akurasi. Menyediakan teknik dan metodologi yang dapat diterapkan pada berbagai domain lain yang memerlukan analisis data berdimensi tinggi, sehingga memberikan manfaat luas dalam berbagai aplikasi machine learning di industri maupun akademisi. Serta menyediakan benchmark yang berguna bagi pengujian dan pengembangan algoritma klasifikasi lainnya, memungkinkan evaluasi komparatif yang lebih baik di masa depan.

1.5 Batasan Masalah

Agar masalah yang dibahas tidak menjadi lebih luas lagi, penulis membatasi masalah penelitian sebagai berikut:

a) Data yang digunakan dalam penelitian ini adalah data stunting di Kota Samarinda pada tahun 2023.

b) Penelitian ini hanya akan menguji efektivitas algoritma Random Forest dalam klasifikasi stunting, dengan fokus pada pemilihan fitur melalui metode Recursive Feature Elimination (RFE) untuk menentukan atribut yang paling relevan dalam prediksi stunting.

c) Fokus penelitian ini adalah pada peningkatan akurasi prediksi stunting, bukan pada intervensi atau implementasi kebijakan yang dihasilkan dari analisis tersebut.

d) Atribut yang digunakan pada penelitain ini antara lain adalah Nama, JK, Berat, LiLA, BB/U, ZS BB/U, TB/U, ZS TB/U, BB/TB, ZS BB/TB, Naik Berat B, JmL Vit A, dan Tanggal Penggukuran setelah dilakukan selection data.

e) Menilai efektivitas model klasifikasi Stunting melalui evaluasi kinerja algoritma Random Forest dengan k-Fold Cross Validation, menggunakan confusion matrix untuk menghitung akurasi yang akurat.

BAB II

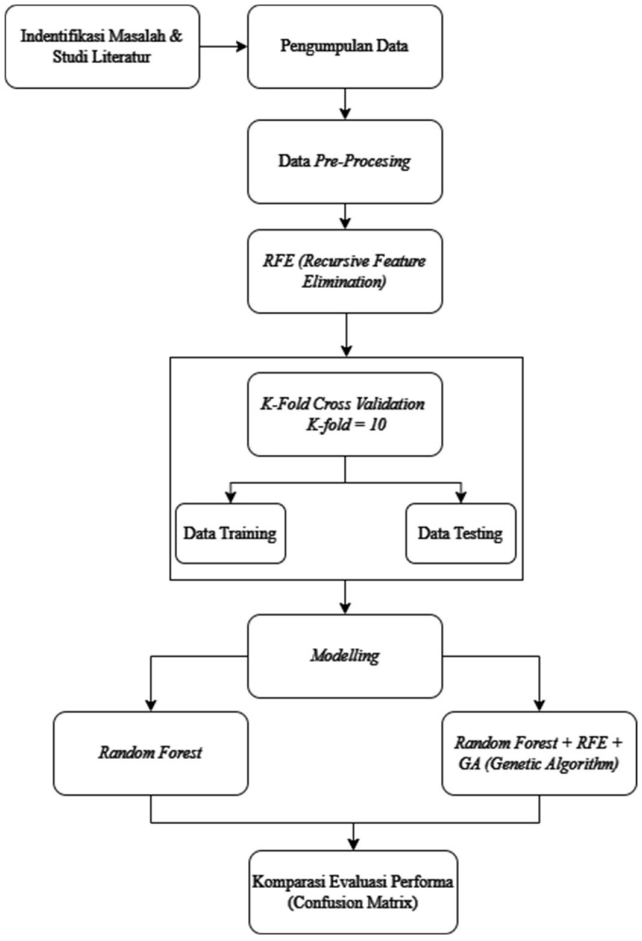
METODOLOGI PENELITIAN

2.1 Objek Penelitian

Objek pada penelitian ini adalah kasus stunting di Kota Samarinda. Data yang digunakan diperoleh dari Dinas Kesehatan Kota Samarinda, dimana pada tahun 2021, prevalensi stunting pada balita di kota tersebut tercatat sebesar 24,7% (Rufina et., al 2023). Data kasus stunting ini dikumpulkan dari 26 puskesmas yang tersebar di berbagai wilayah Kota Samarinda, semuanya beroperasi di bawah pengawasan Dinas Kesehatan setempat. Proses pengumpulan data dilakukan dengan mendokumentasikan kasus-kasus stunting yang teridentifikasi oleh petugas kesehatan di puskesmas-puskesmas tersebut. Penelitian ini dilakukan langsung di Dinas Kesehatan Kota Samarinda, yang berlokasi di Jl. Milono No.1, Bugis, Kecamatan Samarinda Kota, Kota Samarinda, Kalimantan Timur, 76112, untuk memastikan keakuratan dan integritas data yang dikumpulkan

2.2 Teknik Analisis Data

Penelitian ini bertujuan untuk merancang sebuah kerangka kerja metodologis yang rapi dan sistematis, memastikan bahwa setiap tahap penelitian dilakukan secara konsisten dan terstruktur. Pendekatan analitis yang dipilih telah dirancang untuk mengikuti sebuah prosedur yang terorganisir dengan jelas, yang akan dijelaskan lebih lanjut dalam alur penelitian berikut:



Gambar 2. 1 Alur Penelitian

2.2.1 Pengumpulan data

Langkah pertama dalam penelitian ini adalah memahami data yang berkaitan dengan kasus Stunting, yang dimulai dengan pengumpulan data dan diikuti oleh persiapan data. Data yang terkumpul terdiri dari 27 atribut. Tabel 2.1 menunjukkan data yang telah diperoleh dari Dinas Kesehatan Kota Samarinda.

Tabel 2. 1 Atribut Data Stunting Dinas Kesehatan Kota Samarinda

No	Atribut	Tipe Data	Keterangan
1	Nama	*String*	Nama
2	JK	*String*	Jenis Kelamin
3	BB Lahir	*Integer*	Berat Bayi Lahir
4	TB Lahir	*Integer*	Tinggi Badan Bayi Lahir
5	Provinsi	*String*	Provinsi
6	Kab/Kota	*String*	Kabupaten atau Kota
7	Kec	*String*	Kecamatan
8	Puskesmas	*String*	Lokasi Puskesmas
9	Posyandu	*String*	Lokasi Posyandu
10	RT	*Integer*	Rukun Tetangga
11	RW	*Integer*	Rukun Warga
12	Usia Saat Ukur	*Integer*	Usia Bayi Saat Ukur
13	Tanggal Pengukuran	*Date*	Tanggal Pengukuran
14	Berat	*Integer*	Berat Badan
15	Tinggi	*Integer*	Tinggi Badan
16	LiLA	*Integer*	Lingkar Lengan Atas
17	BB/U	*String*	Berat Badan Menurut Umur
18	ZS BB/U	*Integer*	Z Score Berat Badan menurut Umur
19	TB/U	*String*	Tinggi Badan menurut Umur
20	ZS TB/U	*Integer*	Z Score Tinggi Badan menurut Umur
21	BB/TB	*String*	Berat Badan menurut Tinggi Badan
22	ZS BB/TB	*Integer*	Z Score Berat Badan menurut Tinggi Badan
23	Naik Berat Badan	*String*	Naik Berat Badan Bayi
24	PMT Diterima(kg)	*Integer*	Pemberian Makanan Tambahan Bayi
25	Jml Vit A	*Integer*	Jumlah Vitamin A
26	KPSP	*String*	Kuesioner Pra Skrining Perkembangan
27	KIA	*String*	Kartu Indentitas Anak

Tabel 2.1 menampilkan 27 atribut data stunting yang diperoleh dari Dinas Kesehatan Kota Samarinda. Atribut-atribut ini mencakup informasi demografis, pengukuran fisik, status gizi, dan pemberian makanan tambahan. Data tersebut akan digunakan sebagai dasar dalam proses seleksi fitur dan analisis stunting dalam penelitian ini.

2.2.2 Data Pre-Procesing

Dalam tahap ini, proses pre-processing data akan meliputi tiga tahapan kunci yang penting untuk mempersiapkan data agar siap dianalisis. Tahapan pertama adalah data selection, di mana data yang relevan akan dipilih dari keseluruhan kumpulan data berdasarkan kriteria yang telah ditentukan. Tahap kedua, data cleaning, melibatkan mengidentifikasi dan memperbaiki atau menghapus data yang rusak atau tidak lengkap. Tahap terakhir, pada tahap data transformation, data akan diubah atau dikonversi ke

format yang lebih sesuai untuk analisis, termasuk normalisasi atau standarisasi nilai. Data yang digunakan dalam penelitian ini diperoleh dari Dinas Kesehatan Kota Samarinda, dan telah melalui proses persiapan yang telah ditetapkan sebagai berikut:

a) Data Selection

Dalam tahap ini, proses pengambilan data melibatkan pemilihan fitur atau atribut yang relevan untuk penelitian, sementara fitur yang dianggap tidak relevan akan dieliminasi. Sebagai contoh, data awal yang diperoleh dari Dinas Kesehatan Kota Samarinda, yang tertera dalam Tabel 2.2, mencakup 27 atribut. Setelah proses seleksi fitur, 14 atribut dianggap tidak relevan dan dihapus, sehingga tersisa 12 atribut yang akan digunakan sebagai fitur dan 1 atribut lainnya yang akan dijadikan kelas atau target dalam klasifikasi stunting pada anak. Data yang telah diseleksi tersebut ditampilkan pada Tabel 2.2.

Tabel 2. 2 Data Selection

No	Atribut	Tipe Data	Keterangan
1	Nama	*String*	Nama Balita
2	JK	*String*	Jenis Kelamin
3	Berat	*Integer*	Berat Badan
4	Tinggi	*Integer*	Tinggi Badan
5	LiLA	*Integer*	Lingkar Lengan Atas
6	BB/U	*String*	Berat Badan Menurut Umur
7	ZS BB/U	*Integer*	Z Score Berat Badan menurut Umur
8	TB/U	*String*	Tinggi Badan menurut Umur
9	ZS TB/U	*Integer*	Z Score Tinggi Badan menurut Umur
10	BB/TB	*String*	Berat Badan menurut Tinggi Badan
11	ZS BB/TB	*Integer*	Z Score Berat Badan menurut Tinggi Badan
12	Naik Berat B	*String*	Naik Berat Badan
13	Tanggal Pengukuran	*Date*	Tanggal Pengukuran

Data yang telah diseleksi pada Tabel 2.2 mencakup 13 atribut, di mana 12 atribut akan digunakan sebagai fitur dan 1 sebagai target dalam klasifikasi stunting pada anak. Atribut-atribut ini mencakup informasi relevan seperti berat badan, tinggi badan, lingkar lengan atas, dan berbagai z-score yang mengukur status gizi anak. Proses seleksi fitur ini bertujuan menghilangkan atribut yang kurang signifikan untuk meningkatkan efisiensi dan akurasi model klasifikasi, sehingga dapat lebih efektif dalam mengidentifikasi faktor-faktor penting yang berkontribusi terhadap stunting dan menghasilkan prediksi yang lebih akurat.

b) Data Cleaning

Dalam tahap ini, data yang akan digunakan akan diolah melalui proses pembersihan, di mana data yang memiliki nilai #N/A (no value is available) atau data yang tidak memiliki nilai, serta data yang terduplikasi, akan dihapus. Untuk melaksanakan proses ini, kami akan menggunakan bahasa pemrograman Python bersama dengan library Pandas. Khususnya, fungsi dropna() akan digunakan untuk menggantikan nilai yang hilang dengan nilai rata-rata dari data tersebut.

```
# Menghapus baris yang memiliki setidaknya satu nilai yang hilang
data_tanpa_isi = data.dropna()
```

Gambar 2. 2 Kode *Cleaning* Data Kosong

c) Data Transformation

Dalam tahap ini, nilai-nilai dari atribut kategorikal akan dikonversi menjadi bentuk numerik. Konversi ini diperlukan karena *library sklearn* yang digunakan hanya menerima atribut dalam format numerik. Atribut yang akan ditransformasi mencakup JK, Berat, Tinggi, LiLA, BB/U ZS BB/U, TB/U, ZS TB/U, BB/TB, ZS BB/TB, Kelas. Proses transformasi ini akan dilakukan menggunakan bahasa pemrograman Python dan library pandas, dengan menerapkan fungsi map untuk pemetaan nilai-nilai tersebut.

```python
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder

# Membaca dataset
stunting = pd.read_csv('dataset_stunting_samarinda_terbaru.csv')

# Menampilkan kolom yang ada dalam DataFrame
print("Kolom dalam DataFrame:", stunting.columns)

# Membuat instance untuk encoder
ordinal = OrdinalEncoder()
labelencoder = LabelEncoder()

# Daftar kolom yang dibutuhkan
columns_needed = ['JK', 'Berat', 'Tinggi', 'LiLA', 'BB/U', 'ZS BB/U', 'ZS TB/U', 'Naik Berat Badan', 'BB/TB', 'ZS BB/TB', 'Kelas']

# Menyimpan subset data yang diperlukan
df_transform = stunting[columns_needed]

# Transformasi data kategorikal menjadi numerik
stunting['JK'] = labelencoder.fit_transform(stunting['JK'])
stunting['BB/U'] = labelencoder.fit_transform(stunting['BB/U'])
stunting['BB/TB'] = labelencoder.fit_transform(stunting['BB/TB'])
stunting['Naik Berat Badan'] = labelencoder.fit_transform(stunting['Naik Berat Badan'])
```

Gambar 2. 3 Kode Data *Transformation*

2.2.3 K-Fold Cross Validation

Sebelum memulai pembuatan model, dataset yang digunakan perlu dibagi menjadi dua bagian pokok, yakni data pelatihan dan data pengujian. Data pelatihan digunakan sebagai dasar untuk mengembangkan model, sedangkan data pengujian bertugas untuk menilai efektivitas model yang telah dibangun. Dalam penelitian ini, teknik *K-Fold Cross Validation* akan digunakan, yang direalisasikan menggunakan *library sklearn.model_selection* dengan fungsi *cross_val_score* pada *Python*. Teknik ini membagi data menjadi 10 bagian atau kelompok, berdasarkan parameter Cv yang ditetapkan sebesar 10, yang berarti dataset dibagi menjadi 10 segmen. Setiap segmen akan secara bergantian digunakan sebagai data latih dan data uji. Dengan nilai k yang ditetapkan sebesar 10, eksperimen akan dijalankan sepuluh kali dan nilai rata-rata dari semua hasil pengujian akan digunakan untuk menghasilkan estimasi yang lebih presisi dan akurat.

```python
# Membagi data menjadi training dan testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Membuat 10-fold cross-validator
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
```

Gambar 2. 4 Kode *K-Fold Cross Validation*

Berikut disajikan Tabel 2.3 yang mencakup setiap parameter yang terlibat dalam kode beserta deskripsi fungsi masing-masing parameter:

Tabel 2. 3 Penjelasan Parameter Pembagian Dataset dan Perulangan

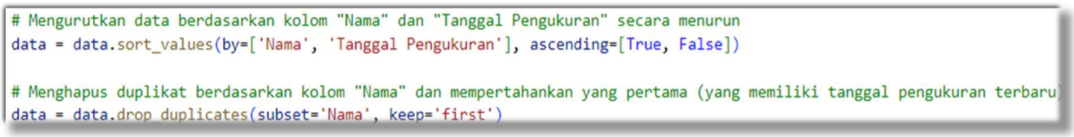
Parameter	Keterangan
X	Data fitur yang digunakan untuk membuat prediksi.
y	Kolom target dari dataset, digunakan sebagai label dalam pemodelan.
n_splits	Jumlah pembagian data dalam proses cross-validation, menunjukkan berapa kali data dibagi. Misalnya, n_splits=10 menunjukkan bahwa data akan dibagi menjadi 10 bagian.
shuffle	Menentukan apakah sampel dalam dataset harus diacak sebelum dibagi menjadi batch untuk cross-validation. Pengacakan membantu dalam menghindari bias sampling dan biasanya diatur menjadi True untuk mendistribusikan data secara merata.
random_state	Parameter yang digunakan untuk mengontrol keacakan saat mengacak atau membagi data, yang membantu dalam mencapai hasil yang konsisten dan dapat direproduksi. Misalnya, mengatur 'random_state=42' berarti bahwa setiap kali kode dijalankan, pemisahan data akan sama.

2.2.4 Pembuatan Model

Berikut adalah deskripsi dan rincian dari model klasifikasi Random Forest yang diimplementasikan dalam penelitian ini :

a) Dalam tahap ini, dataset dibagi menjadi data *training* dan data *testing*. Data pelatihan digunakan untuk mengembangkan model, sementara data pengujian untuk menilai kinerjanya. Menggunakan teknik K-Fold Cross Validation dengan 10 lipatan *melalui library sklearn.model_selection* dan fungsi *cross_val_score*, model *Random Forest* diterapkan untuk klasifikasi stunting di Kota Samarinda. Berikut disajikan penjelasan tentang cara kerja metode ini:

1. Tahap pertama, penelitian ini melibatkan proses ekstraksi sampel dari dataset yang berkaitan dengan stunting di Kota Samarinda. Untuk mengumpulkan data, akan digunakan entri dengan tanggal pengukuran stunting yang paling terkini, sementara entri dengan nama yang sama tetapi dengan tanggal pemeriksaan yang lebih lama akan dieliminasi. Hal ini dilakukan untuk memastikan data yang diperoleh mencerminkan hasil terakhir dari pemeriksaan stunting tahun 2023.



Gambar 2. 5 Kode Data Pengukuran Terbaru

2. Tahap kedua, Inisialisasi Model *Random Forset*. Pada proses pembuatan pohon ini, teknik *Classification and Regression Tree (CART)* diterapkan, dengan menggunakan *gini impurity* sebagai kriteria utama untuk menentukan pembagian di setiap kode dalam pohon.

```
from sklearn.ensemble import RandomForestClassifier

rf_classifier = RandomForestClassifier(
    n_estimators=10 + i,  # Mengubah jumlah estimators
    criterion='gini',
    max_depth=1 + i % 3,  # Mengubah kedalaman maksimal pohon
    min_samples_split=100 + i,  # Mengubah jumlah minimal sampel untuk split
    min_samples_leaf=750 - i * 5,  # Mengubah jumlah minimal sampel per daun
    max_features=0.1 + (i % 5) * 0.02,  # Mengubah jumlah fitur yang dipertimbangkan untuk split
    class_weight='balanced',
    random_state=42 + i  # Mengubah random state
)
```

Gambar 2. 6 Kode *Random Forest*

3. Tahap ketiga proses di langkah b diulang terus menerus sampai jumlah pohon keputusan yang ditargetkan, yaitu k = 10, tercapai. Dan membagi dataset menjadi data *training* dan *testing.*

4. Tahap keempat dalam pengembangan model menggunakan algoritma *Random Forest* melibatkan aplikasi library *scikit-learn*. Pada tahap ini, teknik *k-fold* cross-validation digunakan untuk memisahkan dataset menjadi bagian latihan dan uji, serta untuk menilai kinerja model dengan cara yang objektif. Awalnya, model *Random Forest* dibuat menggunakan *Random Forest Classifier* dari *scikit-learn*. Kemudian, proses iterasi dilakukan melalui tiap fold yang dibentuk oleh *k-fold cross-validation.* Dalam setiap iterasi, dataset dibagi antara data latihan dan uji, dengan model *Random Forest* dilatih menggunakan data latihan. Setelah proses pelatihan, model ini diaplikasikan untuk membuat prediksi pada data uji. Skor evaluasi seperti akurasi, presisi, recall, dan F1-score dihitung menggunakan metrik yang relevan dari library *sklearn.metrics* dan dicatat untuk tiap *fold*. Selain itu, *confusion matrix* untuk tiap *fold* juga dihitung menggunakan fungsi *confusion_matrix* dari *scikit-learn.* Prediksi dan skor evaluasi ini lalu disimpan untuk analisis lebih lanjut mengenai kinerja model terhadap dataset yang dipilih. setelah menyelesaikan proses *k-fold cross-validation* dan menguji model *Random Forest* pada setiap *fold*, skor evaluasi seperti akurasi, presisi, *recall*, dan *F1 Score* untuk tiap *fold* akan ditampilkan. Menggunakan perulangan dengan *`enumerate`* dan *`zip`*, skor evaluasi tiap fold diproses dan ditampilkan pada layar. Selanjutnya, nilai rata-rata dari skor evaluasi semua fold dihitung menggunakan fungsi *`np.mean()`* dari *NumPy* dan hasilnya dicetak. Selain itu, rata-rata dari matriks kebingungan *(confusion matrix)* untuk semua *fold* dihitung dan ditampilkan menggunakan fungsi yang sama. Matriks kebingungan rata-rata ini memberikan indikasi tentang efektivitas model dalam mengklasifikasikan tiap kelas dalam data uji.

$$\mathcal{Y} = argmax\mathcal{X} \sum_{i=1}^{T} I(\mathrm{h}i(\mathcal{X}) = \mathcal{K}) \quad (2.1)$$

$\mathcal{Y}$ = Kelas perdiksi akhir untuk data $\mathcal{X}$

$argmax\mathcal{X}$ = Operator yang mencari nilai $\mathcal{K}$ yang memaksimalkan jumlah suara (argument maksium). Dalam konteks ini, $\mathcal{K}$ adalah kelas.

$\sum_{i=1}^{T}$ = Menunjukan penjumlahan dari i =1 hingga T, dimana T adalah jumlah total pohon dalam random Forest.

$I(\mathrm{h}i(\mathcal{X}) = \mathcal{K})$ = Fungsi indicator yang bernilai 1 jika diprediksi pohon ke- i (h i) untuk data $\mathcal{X}$ adalah kelas $\mathcal{K}$, dan 0 jika tidak.

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_predict, KFold, train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import seaborn as sns

# Fungsi untuk membuat plot matriks kebingungan
def buat_matriks_kebingungan(cf, nama_grup=None, kategori='auto', jumlah=True, persen=True, barwarna=True, xyticks=True, xyplotlabels=True, stat_ringkasan=True, ukuran_gambar=None, cmap='Blues', judul=None, akurasi=None):
    kosong = ['' for i in range(cf.size)]
    if nama_grup and len(nama_grup) == cf.size:
        label_grup = ["{}\n".format(value) for value in nama_grup]
    else:
        label_grup = kosong
    if jumlah:
        jumlah_grup = ["{0:0.0f}\n".format(value) for value in cf.flatten()]
    else:
        jumlah_grup = kosong
    if persen:
        persentase_grup = ["{0:.2%}".format(value) for value in cf.flatten() / np.sum(cf)]
    else:
        persentase_grup = kosong
    label_kotak = [f"{v1}{v2}{v3}".strip() for v1, v2, v3 in zip(label_grup, jumlah_grup, persentase_grup)]
    label_kotak = np.asarray(label_kotak).reshape(cf.shape[0], cf.shape[1])
    if stat_ringkasan and akurasi is not None:
        teks_stat = "\n\nAkurasi Rata-rata={:0.3f}".format(akurasi)
    else:
        teks_stat = ""
    if ukuran_gambar == None:
        ukuran_gambar = plt.rcParams.get('figure.figsize')
    if xyticks == False:
        kategori = False
    plt.figure(figsize=ukuran_gambar)
    sns.heatmap(cf, annot=label_kotak, fmt="", cmap=cmap, cbar=barwarna, xticklabels=kategori, yticklabels=kategori)
    if xyplotlabels:
        plt.ylabel('Label Asli')
        plt.xlabel('Label Prediksi' + teks_stat)
    else:
        plt.xlabel(teks_stat)
    if judul:
        plt.title(judul)
    plt.show()
```

```python
# Membaca data yang telah diproses
data = pd.read_csv('dataset_stunting_samarinda_modif.csv')

# Mengganti nama kolom 'TB/U' menjadi 'Kelas'
data = data.rename(columns={'TB/U': 'Kelas'})

# Menghapus kolom non-numerik dan tidak relevan
data = data.drop(['Nama', 'Tanggal Pengukuran'], axis=1)

# Mengganti nilai bermasalah di kolom 'JK'
data['JK'] = data['JK'].str.strip().replace({'L\xa0': 'L', 'P\xa0': 'P'})

# Menangani nilai numerik bermasalah
data['Berat'] = data['Berat'].str.replace(',', '.').astype(float)

# Mengonversi kolom kategorikal menjadi numerik menggunakan one-hot encoding
kolom_kategorikal = ['JK', 'BB/U', 'BB/TB', 'Naik Berat Badan']
data = pd.get_dummies(data, columns=kolom_kategorikal, drop_first=True)

# Memastikan tidak ada kehilangan data setelah konversi
print(f"Bentuk dataset setelah konversi: {data.shape}")

# Memisahkan fitur dan variabel target
X = data.drop('Kelas', axis=1)
y = data['Kelas'].map({'Tidak Stunting': 0, 'Stunting': 1})

# Menormalisasi fitur
scaler = MinMaxScaler()
X_normalized = scaler.fit_transform(X)

# Membagi data menjadi training dan testing set
x_train, x_test, y_train, y_test = train_test_split(X_normalized, y, test_size=0.2, random_state=42)

# Merekam hasil K-Fold 10 kali
kfolds = 10
repeats = 10
akurasi_list = []

# Menggunakan KFold untuk cross-validation
kf = KFold(n_splits=kfolds, shuffle=True, random_state=1)
fold = 1
```

```python
# Melakukan cross-validation dengan variasi
for i in range(repeats):
    print(f"Rekaman {i+1}")

    # Mendefinisikan classifier Random Forest dengan parameter yang sedikit berbeda untuk setiap iterasi
    rf_classifier = RandomForestClassifier(
        n_estimators=10 + i,  # Mengubah jumlah estimators
        criterion='gini',
        max_depth=1 + i % 3,  # Mengubah kedalaman maksimal pohon
        min_samples_split=100 + i,  # Mengubah jumlah minimal sampel untuk split
        min_samples_leaf=50 - i * 5,  # Mengubah jumlah minimal sampel per daun
        max_features=0.1 + (i % 5) * 0.02,  # Mengubah jumlah fitur yang dipertimbangkan untuk split
        class_weight='balanced',
        random_state=42 + i  # Mengubah random state
    )

    y_pred = cross_val_predict(rf_classifier, x_train, y_train, cv=kfolds)
    cf_matrix = confusion_matrix(y_train, y_pred)
    akurasi = np.trace(cf_matrix) / float(np.sum(cf_matrix))
    akurasi_list.append(akurasi)
    print(f"Akurasi: {akurasi:.3f}")
    class_report = classification_report(y_train, y_pred, target_names=['Tidak Stunting', 'Stunting'])
    print(class_report)

# Menggambar plot matriks kebingungan untuk iterasi terakhir dengan rata-rata akurasi
buat_matriks_kebingungan(cf_matrix, ukuran_gambar=(8, 6), barwarna=False, judul=f"Model Random Forest (K-Fold = 10)")
```

Gambar 2. 7 Kode Model *Algoritma Random Forest*

Berikut adalah Tabel 2.4 yang menyajikan daftar parameter yang digunakan dalam kode tersebut, beserta deskripsi fungsi dari masing-masing parameter.

Tabel 2. 4 Parameter Model Algoritma *Random Forest*

Parameter	Keterangan
n_estimators=10	Mengatur jumlah pohon dalam hutan menjadi 10.
criterion='gini'	Menggunakan kriteria 'gini' untuk mengukur kualitas split.
max_depth=1	Membatasi kedalaman maksimum pohon menjadi 1 untuk mencegah overfitting.
min_samples_split=100	Jumlah minimum sampel yang diperlukan untuk memisahkan node internal.
min_samples_leaf=50	Jumlah minimum sampel yang diperlukan untuk berada di daun node.
max_features=0.1	Proporsi maksimum fitur yang dipertimbangkan untuk split, diatur ke 10%.
class_weight='balanced'	Mengatur bobot kelas untuk menangani ketidakseimbangan kelas.
random_state=42	Mengatur seed untuk memastikan hasil yang konsisten.
nama_grup	Daftar nama untuk setiap grup di matriks kebingungan.
kategori	Kategori untuk sumbu x dan y.
jumlah	Menampilkan jumlah sampel di setiap kotak matriks.
persen	Menampilkan persentase di setiap kotak matriks.
barwarna	Menampilkan bar warna pada plot.
xyticks	Menampilkan tick marks pada sumbu x dan y.
xyplotlabels	Menampilkan label untuk sumbu x dan y.
stat_ringkasan	Menampilkan statistik ringkasan pada plot.
ukuran_gambar	Ukuran gambar untuk plot.
cmap	Skema warna untuk plot.
scaler	Objek MinMaxScaler untuk normalisasi fitur.
X_normalized	Hasil normalisasi fitur-fitur dalam dataset.
kfolds	Jumlah split dalam cross-validation (10-fold).
repeats	Jumlah pengulangan cross-validation (10 kali).
kf	Objek KFold untuk melakukan cross-validation.
fold	Nomor fold dalam cross-validation (tidak digunakan lebih lanjut dalam kode).
y_pred	Hasil prediksi dari cross-validation.
cf_matrix	Matriks kebingungan yang dihasilkan dari hasil prediksi dan label asli.
akurasi	Akurasi model yang dihitung dari matriks kebingungan.
class_report	Laporan klasifikasi yang berisi metrik seperti precision, recall, dan f1-score.

b) Dalam tahap seleksi fitur menggunakan metode *Recursive Feature Elimination (RFE)* dilakukan dengan cara mengevaluasi dan menghilangkan atribut yang kurang signifikan secara bertahap untuk meningkatkan performa model klasifikasi. Metode ini memulai dengan set fitur lengkap dan kemudian secara sistematis menghilangkan fitur-fitur yang memiliki bobot atau pengaruh paling rendah terhadap variabel target berdasarkan kriteria tertentu, yang seringkali ditentukan oleh model yang digunakan, seperti koefisien dalam regresi atau pentingnya fitur dalam pohon keputusan. Secara spesifik, *RFE* bekerja dengan cara sebagai berikut:

1. Semua fitur yang relevan dipilih dari dataset stunting untuk dilakukan analisis lebih lanjut.
2. Dalam menggunakan metode *Recursive Feature Elimination (RFE)* yang diimplementasikan melalui library *scikit-learn*, metode ini memilih fitur dengan mengeliminasi fitur yang paling lemah satu per satu berdasarkan bobot yang ditentukan oleh estimator yang digunakan. Di bawah ini adalah konsep umum dari *RFE*:

```
from sklearn.feature_selection import RFE


# Melakukan seleksi fitur menggunakan RFE
selector = RFE(rf_base, n_features_to_select=2, step=1)
selector = selector.fit(X_normalized, y)
X_selected = selector.transform(X_normalized)
```

Gambar 2. 8 Kode Seleksi Fitur *RFE*

Berikut adalah Tabel 2.5 yang menyajikan daftar parameter yang digunakan dalam kode tersebut, beserta deskripsi fungsi dari masing-masing parameter:

Tabel 2. 5 Parameter Seleksi Fitur *RFE*

Parameter	Keterangan
rf_base	Model dasar yang digunakan oleh *RFE* untuk menilai pentingnya fitur (*RandomForestClassifier*).
n_features_to_select=2	Jumlah fitur yang diinginkan untuk dipilih oleh *RFE* (2 fitur).
step=1	Jumlah fitur yang dihilangkan pada setiap iterasi RFE (1 fitur per iterasi).

c) Dalam tahap optimasi menggunakan Algoritma *Genetika (GA)*, dilakukan dengan mengevaluasi dan memodifikasi atribut secara bertahap untuk meningkatkan performa model klasifikasi. Metode ini memulai dengan set fitur lengkap dan kemudian secara sistematis mengidentifikasi dan memodifikasi konfigurasi fitur yang memiliki pengaruh paling signifikan terhadap variabel target. Proses ini didasarkan pada kriteria tertentu yang seringkali ditentukan oleh hasil evaluasi *fitness* dalam *GA*. Secara spesifik, Algoritma *Genetika* bekerja dengan cara sebagai berikut:

1. Untuk mengimplementasikan optimasi menggunakan *Algoritma Genetik (GA)* sebagai lanjutan dari kode seleksi fitur *RFE* yang telah kita bahas, kita akan memerlukan library tambahan yang dapat menangani optimasi genetik. Salah satu library yang populer untuk ini adalah *'DEAP' (Distributed Evolutionary Algorithms in Python)*. Dalam contoh ini, kita akan menggunakan *DEAP* untuk mengoptimalkan fitur-fitur yang dipilih dari model *Random Forest Classifier* untuk mencapai akurasi yang lebih tinggi.
2. Berikut adalah kode yang menunjukkan bagaimana mengintegrasikan optimasi *GA* dengan model *Random Forest Classifier* yang sudah ada:

```
try:
    from deap import base, creator, tools, algorithms
except ImportError:
    !pip install deap
    from deap import base, creator, tools, algorithms

# Konfigurasi GA
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)

toolbox = base.Toolbox()
toolbox.register("attr_bool", np.random.randint, 0, 2)
toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_bool, n=len(X.columns))
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

# Fungsi evaluasi
def evalRF(individual):
    selected_features = [i for i, bit in enumerate(individual) if bit == 1]
    if len(selected_features) == 0:
        return 0,  # Hindari subset fitur kosong
    X_subset = X_normalized[:, selected_features]
    selector = RFE(rf_base, n_features_to_select=min(2, len(selected_features)), step=1)
    X_selected = selector.fit_transform(X_subset, y)
    y_pred = cross_val_predict(rf_base, X_selected, y, cv=5)  # Mengurangi jumlah CV fold
    return (np.mean(cross_val_predict(rf_base, X_selected, y, cv=5) == y),)  # Mengembalikan akurasi

toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=0.05)
toolbox.register("select", tools.selTournament, tournsize=3)
toolbox.register("evaluate", evalRF)

# Menjalankan GA dengan pengaturan yang lebih cepat
population = toolbox.population(n=10)  # Mengurangi ukuran populasi
ngen = 10  # Mengurangi jumlah generasi
cxpb = 0.5
mutpb = 0.3

result = algorithms.eaSimple(population, toolbox, cxpb, mutpb, ngen, stats=None, halloffame=None, verbose=False)

# Menggunakan fitur terbaik yang ditemukan oleh GA
best_individual = tools.selBest(population, 1)[0]
selected_features = [i for i, bit in enumerate(best_individual) if bit == 1]
X_best = X_normalized[:, selected_features]
```

Gambar 2. 9 Kode Optimasi *GA*

Berikut adalah Tabel 2.6 yang menyajikan daftar parameter yang digunakan dalam kode tersebut, beserta deskripsi fungsi dari masing-masing parameter:

Tabel 2. 6 Parameter Optimasi *GA*

Parameter	Keterangan
creator.create	Digunakan untuk mendefinisikan kelas baru yang mewakili individu dan fitness dalam populasi *GA.*
FitnessMax	Sebuah kelas yang mewakili jenis *fitness* yang ingin dimaksimalkan, dalam hal ini adalah akurasi.
Individual	Klas yang merepresentasikan setiap individu dalam populasi *GA*, berupa daftar fitur yang dipilih atau tidak.
evalRF	Fungsi evaluasi yang menghitung seberapa baik individu (set fitur) berdasarkan akurasi model *RandomForestClassifier*.
toolbox	Wadah untuk menyimpan fungsi yang digunakan dalam *GA* seperti inisialisasi, evaluasi, dan operasi genetik.
attr_bool	Fungsi untuk menginisialisasi atribut individu, dalam hal ini sebagai nilai biner (0 atau 1, fitur tidak dipilih atau dipilih).
individual	Fungsi untuk menginisialisasi sebuah individu berdasarkan attr_bool, menentukan panjang individu berdasarkan jumlah fitur.

population	Fungsi untuk menginisialisasi populasi yang terdiri dari individu-individu.
mate	Fungsi untuk melakukan persilangan (crossover) antar individu, menggunakan two-point crossover.
mutate	Fungsi untuk melakukan mutasi pada individu dengan probabilitas tertentu, menggunakan flip bit mutation.
select	Fungsi seleksi yang menentukan individu mana yang akan bertahan dan bereproduksi, menggunakan tournament selection.
evaluate	Fungsi evaluasi yang digunakan oleh GA untuk menilai individu.
eaSimple	Algoritma yang digunakan untuk menjalankan *GA*, yang mencakup proses seleksi, persilangan, dan mutasi secara berulang.
ngen	Parameter yang menentukan jumlah generasi yang akan dijalankan dalam simulasi *GA*.
cxpb	Probabilitas persilangan (crossover probability), mengontrol seberapa sering crossover terjadi antar individu.
mutpb	Probabilitas mutasi (mutation probability), mengontrol seberapa sering mutasi terjadi pada individu.
selBest	Memilih individu terbaik dari populasi setelah evolusi selesai.
selected_features	Daftar fitur yang dipilih oleh individu terbaik.
X_best	Dataset yang hanya berisi fitur-fitur yang dipilih oleh individu terbaik.

2.2.5 Evaluasi

Pada tahap evaluasi, penelitian ini akan mengukur keakuratan algoritma yang diterapkan dengan mempertimbangkan kualitas dari data latihan yang digunakan. Evaluasi ini akan dilakukan melalui pengujian menggunakan teknik *Confusion Matrix* untuk menentukan tingkat *Accuracy* (akurasi). Proses ini penting untuk memastikan bahwa algoritma yang digunakan dapat secara efektif mengklasifikasikan data dengan presisi yang tinggi.

Tabel 2.7 *Confusion Matrix*

Predicted	*Positive*	*Negative*
Actual Positive	*TP*	*FN*
Actual Negative	*FP*	*TN*

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \quad (2.2)$$

Berikut adalah penjelasan tentang istilah-istilah yang digunakan dalam Confusion Matrix:

TP (*True Positive*) adalah jumlah data yang benar-benar berlabel '*yes'* dan diidentifikasi secara benar.

TP (*True Negative*) adalah jumlah data yang benar-benar berlabel '*no*'dan diidentifikasi secara benar.

FP (*False Positive*) adalah jumlah data yang benar-benar berlabel '*yes'* dan diidentifikasi secara benar.

FN (*False Negative*) adalah jumlah data yang benar-benar berlabel '*no'* dan diidentifikasi secara benar.

BAB III
HASIL DAN PEMBAHASAN

3.1 Hasil

Pada bagian ini, akan disajikan hasil analisis dan pemodelan data stunting di Kota Samarinda tahun 2023 yang telah dilakukan menggunakan berbagai metode yang berbeda. Penelitian ini mencakup serangkaian tahapan, mulai dari proses seleksi dan integrasi data, pembersihan data untuk memastikan kualitas dan konsistensi, hingga transformasi data agar lebih sesuai dengan kebutuhan algoritma pemodelan. Setiap tahap ini dirancang untuk meningkatkan akurasi dan efisiensi dari model yang akan digunakan.

Pada tahap pemodelan, berbagai teknik klasifikasi diterapkan untuk mengevaluasi performa masing-masing metode. Tahap awal dimulai dengan penerapan metode *Random Forest* dasar untuk mendapatkan gambaran awal dari hasil klasifikasi. Selanjutnya, dilakukan penambahan teknik seleksi fitur menggunakan *Recursive Feature Elimination (RFE)* untuk memilih fitur-fitur yang paling relevan dan signifikan dalam proses klasifikasi. Tahap terakhir melibatkan optimasi model menggunakan *Genetic Algorithm (GA)* untuk menemukan konfigurasi hyperparameter yang optimal, sehingga dapat meningkatkan akurasi dan kinerja model secara keseluruhan.

Hasil dari setiap tahap pemodelan ini akan dibahas secara rinci, termasuk analisis terhadap perubahan akurasi yang diperoleh pada setiap langkah. Selain itu, analisis perbandingan hasil akurasi dari masing-masing metode juga akan disajikan untuk memberikan gambaran yang komprehensif tentang kinerja masing-masing teknik dalam mengklasifikasikan data stunting di Kota Samarinda. Dengan demikian, dapat diketahui seberapa besar peningkatan performa yang diperoleh dari setiap tahapan seleksi fitur dan optimasi yang dilakukan, serta implikasi dari hasil tersebut terhadap upaya penanggulangan stunting di Kota Samarinda.

3.2 Data Penelitian

Penelitian ini menggunakan data stunting yang didapatkan dari Dinas Kesehatan Kota Samarinda. Data yang digunakan mencakup periode bulan Januari 2023 hingga Desember 2023 dengan total sebanyak 150.474 data. Data stunting yang diperoleh dari Dinas Kesehatan Samarinda memiliki 28 atribut yang penting dan mendetail. Atribut-atribut tersebut antara lain meliputi Nama, Jenis Kelamin (JK), Berat Badan saat Lahir (BB Lahir), Tinggi Badan saat Lahir (TB Lahir), Provinsi (Prov), Kabupaten/Kota (Kab/Kota), Kecamatan (Kec), Puskesmas, Desa/Kelurahan (Desa/Kel), Posyandu, Rukun Tetangga (RT), Rukun Warga (RW), Usia Saat Pengukuran (Usia Saat Ukur), Tanggal Pengukuran, Berat Badan (Berat), Tinggi Badan (Tinggi), Lingkar Lengan Atas (LiLA), Berat Badan per Usia (BB/U), Z-Score Berat Badan per Usia (ZS BB/U), Tinggi Badan per Usia (TB/U), Z-Score Tinggi Badan per Usia (ZS TB/U), Berat Badan per Tinggi Badan (BB/TB), Z-Score Berat Badan per Tinggi Badan (ZS BB/TB), Kenaikan Berat Badan (Naik Berat Badan), Pemberian Makanan Tambahan yang diterima dalam kilogram (PMT Diterima (kg)), Jumlah Vitamin A yang diterima (Jml Vit A), Kuesioner Praktek Sehari-hari Pengasuhan (KPSP), dan Kartu Ibu dan Anak (KIA).

Tabel 3.1 Dataset Stunting Kota Samarinda Tahun 2023

NO	1	2	3	:	150472	150473	15074
Nama	Dimas Aditya	Siti Aisyah	M Al Fatih	:	Afizah Khairina	M Arsya Kholif	Muhamad Iqbal
JK	L	P	L	:	P	P	L
BB Lahir	3.5	3	2.8	:	2.5	3	2.9
TB Lahir	49	48	49	:	45	49	49
Provinsi	Kalimantan Timur	Kalimantan Timur	Kalimantan Timur	:	Kalimantan Timur	Kalimantan Timur	Kalimantan Timur
Kab/Kota	Samarinda	Samarinda	Samarinda	:	Samarinda	Samarinda	Samarinda
Kecamatan	Sungai Pinang	Sungai Pinang	Sungai Pinang	:	Samarinda Seberang	Samarinda Ulu	Samarinda Ilir
Puskesmas	Remaja	Remaja	Remaja	:	Mangkupalas	Juanda	Sidomulyo
Desa/Kel	Temindung Permai	Temindung Permai	Temindung Permai	:	Tenun Samarinda	Air Hitam	Sungai Dama
Posyandu	Pulau Indah	Pulau Indah	Pulau Indah	:	Balo Negara	Mekar Sejahtera	Ramania
RT	-	34	-	:	12	32	-
RW	-	-	-	:	-		
Usia Saat Ukur	0 Tahun - 11 Bulan - 11 Hari	4 Tahun - 0 Bulan - 16 Hari	0 Tahun - 7 Bulan - 7 Hari	:	0 Tahun - 0 Bulan - 0 Hari	0 Tahun - 0 Bulan - 0 Hari	0 Tahun - 0 Bulan - 0 Hari
Tanggal Pengukuran	2023-01-02	2023-01-02	2023-01-02	:	2023-12-09	2023-12-19	2023-12-29
Berat	9.1	12	8.1	:	2.5	3	2.9
Tinggi	74.5	94	69	:	45	49	49
LiLA	0	0	0	:	-	-	-
BB/U	Berat Badan Normal	Kurang	Berat Badan Normal	:	Kurang	Berat Badan Normal	Kurang
ZS BB/U	-0.39	-2.25	-0.53	:	-2.03	-1.56	-2.97
TB/U	Normal	Pendek	Normal	:	Pendek	Normal	Pendek
ZS TB/U	-0.21	-2.09	-0.65	:	-2.63	-1.3	-2.48

BB/TB	Gizi Baik	Gizi Baik	Gizi Baik	:	Gizi Baik	Gizi Baik	Gizi Baik
ZS BB/TB	-0.39	-1.46	-0.14	:	-0.35	-1.04	-1.37
Naik Berat Badan	O	O	O	:	-	-	-
PMT Diterima(kg)	-	0.84	-	:	-	-	-
Jml Vit A	-	-	-	:	-	-	-
KPSP	-	-	-	:	-	-	-
KIA	-	-	-	:	-	-	-

3.3 Seleksi Dan Integrasi Data

Tahap ini berjalan setelah data yang dikumpulkan kemudian dipilih atribut-atribut yang akan digunakan. Langkah ini bertujuan untuk meningkatkan akurasi dan efisiensi dari algoritma yang diterapkan. Proses seleksi dan integrasi pada tahap pertama melibatkan penyeleksian atribut-atribut yang dianggap tidak diperlukan. Atribut-atribut yang dihapus dari data stunting Kota Samarinda 2023 antara lain adalah BB Lahir, TB Lahir, Prov, Kab/Kota, Kec, Puskesmas, Desa/Kel, Posyandu, RT, RW, Usia Saat Ukur, PMT Diterima (kg), KPSP, dan KIA. Berikut adalah hasil integrasi tahap pertama setelah atribut yang tidak diperlukan dihapus: Nama, JK, Tanggal Pengukuran, Berat, Tinggi, LiLA, BB/U, ZS BB/U, TB/U, ZS TB/U, BB/TB, ZS BB/TB, Naik Berat Badan, dan Jml Vit A.

Tabel 3.2 Hasil Seleksi dan Integrasi Data

No	Nama	JK	Berat	Tinggi	LiLA	BB/UU	ZS BB/UU	TB/UU	ZS TB/UU	BB/TB	ZS BB/TB	NBB	Tanggal Penggukuran
1	Dimas Aditya	L	9.1	74.5	0	BB Normal	-0.39	Normal	-0.21	Gizi Baik	-0.39	O	2023-01-02
2	Siti Aisya	P	12	94	0	Kurang	-2.25	Pendek	-2.09	Gizi Baik	-1.46	O	2023-01-02
3	M Al Fatih	L	8.1	69	0	BB Normal	-0.53	Normal	-0.65	Gizi Baik	-0.14	O	2023-01-02
:	:	:	:	:	:	:	:	:	:			:	:
150473	Afizah Khairina	P	2.5	45	-	Kurang	-2.03	Pendek	-2.63	Gizi Baik	-0.35	-	2023-12-19
150474	M Arsya Kholif	P	3	49	-	BB Normal		Normal		Gizi Baik	-1.04	-	2023-12-19
150475	Muhammad Iqbal	L	2.9	49	-	Kurang		Pendek		Gizi Baik	-1.37	-	2023-12-19

Tabel 3.1 menunjukkan hasil seleksi dan integrasi data setelah menghapus atribut yang tidak relevan. Data yang dipilih mencakup atribut penting seperti nama, jenis kelamin, berat badan, tinggi badan, lingkar lengan atas, dan berbagai *z-score* terkait status gizi anak. Proses ini bertujuan untuk meningkatkan akurasi dan efisiensi model klasifikasi dengan fokus pada atribut yang signifikan dalam analisis stunting di Kota Samarinda.

3.3.1 Data *Cleaning*

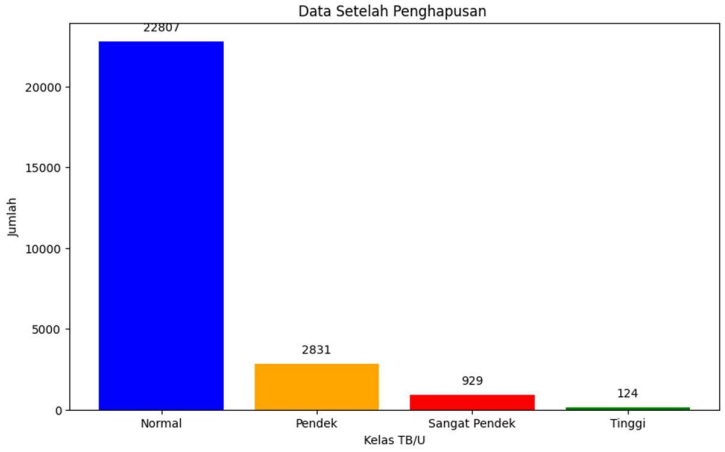
Tahap selanjutnya akan melibatkan pembersihan data untuk memastikan tidak adanya anomali atau data yang hilang (missing data), serta data yang terduplikasi, sehingga proses analisis dapat berjalan lebih efisien dan hasilnya lebih akurat. Dengan data yang sudah diseleksi dan diintegrasikan dengan baik, langkah ini akan membantu dalam mendapatkan hasil analisis yang lebih baik. Hasil data Cleaning terlihat pada table 3.2.

Tabel 3.3 Data Setelah *Cleaning*

No	Nama	JK	Berat	Tinggi	LiLA	BB/UU	ZS BB/UU	TB/UU	ZS TB/UU	BB/TB	ZS BB/TB	NBB	Tanggal Penggukuran
1	A	L	18.6	104.5	15	Risiko	1.19	Normal	0.41	Risiko Gizi	1.41	N	2023-12-16

	Alvin					Lebih				Lebih			
2	A Fadlan	L	11.6	83	0	BB Normal	-0.08	Normal	-0.97	Gizi Baik	0.59	T	2023-07-10
:	:	:	:	:	:	:	:	:	:			:	:
26690	Zubair	L	17.3	107	0	BB Normal	-0.14	Normal	-0.13	Gizi Baik	-0.13	O	2023-02-20
26691	Zulkifli Abdi	L	15.6	102	0	BB Normal	-0.59	Normal	-0.69	Gizi Baik	-0.25	N	2023-12-03

Pada Tabel 3.2, pembersihan data yang dilakukan pada dataset stunting Kota Samarinda mencakup penghapusan data yang memiliki nilai #N/A (missing data) atau data yang terduplikasi. Dalam proses ini, total data yang dihapus karena terduplikasi adalah 116.273 data, dan total data yang dihapus karena memiliki nilai #N/A adalah 7.510 data. Sehingga, setelah proses pembersihan data, jumlah data yang tersisa adalah 26.691.



Gambar 3.1 Dataset Setelah Tahap *Cleaning*

Pada Gambar 3.2 menunjukkan distribusi data stunting Kota Samarinda setelah pembersihan data sesuai Tabel 3.2. Dari total 26.691 record yang tersisa, sebagian besar anak berada dalam kategori tinggi badan normal (22.807), sementara sisanya tersebar dalam kategori pendek (2.831), sangat pendek (929), dan tinggi (124). Data ini mencerminkan mayoritas anak memiliki tinggi badan normal, namun masih ada yang mengalami stunting.

3.3.2 Data *Transformation*

Dalam tahap ini, Transformasi data merupakan langkah penting dalam proses analisis data, di mana atribut kategorikal diubah menjadi numerik untuk memungkinkan pemrosesan yang lebih efektif oleh algoritma machine learning. Pada langkah ini, kita menggunakan teknik seperti Label Encoding dan Ordinal Encoding untuk mengubah nilai kategorikal seperti 'L' dan 'P' dalam kolom 'JK' menjadi nilai numerik. Proses ini tidak hanya meningkatkan efisiensi komputasi tetapi juga memastikan bahwa data dapat diolah oleh model machine learning yang umumnya membutuhkan input numerik. Atribut yang akan ditransformasi mencakup JK, BB/UU, BB/TB, Naik Berat Badan dan TB/UU.

Tabel 3.4 Data Sebelum di *Transformation*

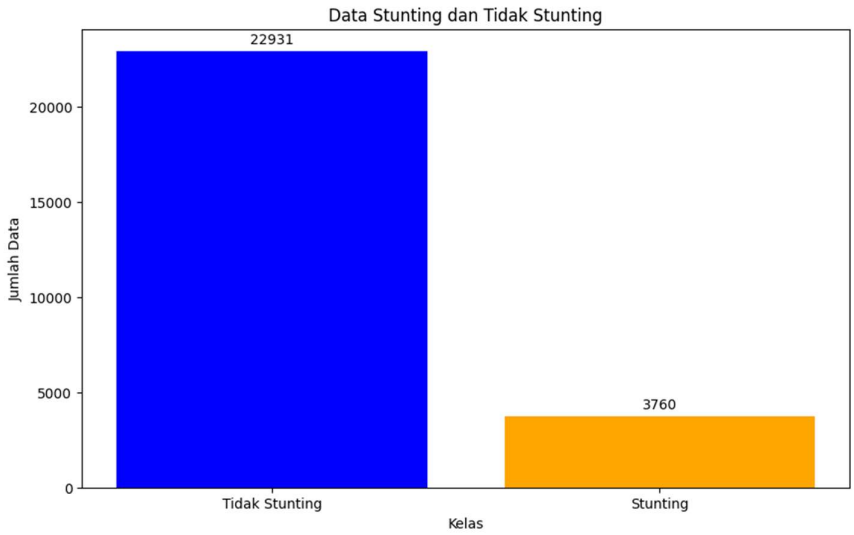
No	JK	BB/UU	BB/TB	Naik Berat Badan	TB/UU
1	L	Risiko Lebih	Risiko Gizi Lebih	N	Normal
2	L	BB Normal	Gizi Baik	T	Normal
3	L	BB Normal	Gizi Baik	O	Pendek
:	:	:	:	:	:
26689	P	Risiko Lebih	Risiko Gizi Lebih	0	Normal
26690	L	BB Normal	Gizi Baik	O	Normal
26691	L	BB Normal	Gizi Baik	N	Normal

Pada Tabel 3.3, saya memilih kolom JK, BB/UU, BB/TB, dan NBB. Untuk mentransformasi datanya menjadi numerik.

Tabel 3.5 Data Setelah di *Transformation*

No	JK	BB/UU	BB/TB	Naik Berat Badan	Kelas
1	0	2	5	1	Tidak Stunting
2	0	0	0	3	Tidak Stunting
3	0	0	0	2	Stunting
:	:	:	:	:	:
26689	1	2	5	2	Tidak Stunting
26690	0	0	0	2	Tidak Stunting
26691	0	0	0	1	Tidak Stunting

Pada tabel di atas, atribut ‘TB/UU’ tidak ada karena atribut tersebut ingin saya gunakan sebagai kelas. Atribut ‘TB/UU’ akan dijadikan kelas untuk menentukan apakah seorang anak mengalami stunting atau tidak, dengan kategori ‘Normal’ dan ‘Tinggi’ diklasifikasikan sebagai ‘Tidak Stunting’, serta kategori ‘Pendek’ dan ‘Sangat Pendek’ diklasifikasikan sebagai ‘Stunting’.



Gambar 3.2 Data Setelah diubah kategorinya

Pada gambar 3.2 menunjukkan bahwa dari total data, 22.931 anak dikategorikan sebagai "Tidak Stunting" (dengan tinggi badan per-usia normal atau tinggi), sedangkan 3.760 anak dikategorikan sebagai "Stunting" (dengan tinggi badan per-usia pendek atau sangat pendek).

3.4 Hasil Pemodelan Setiap Algortima

Pada bagian ini, akan dijelaskan tahapan-tahapan dalam melakukan klasifikasi data stunting di Kota Samarinda menggunakan metode *Random Forest*. Proses pemodelan dimulai dengan penerapan metode *Random Forest* dasar untuk memperoleh gambaran awal dari hasil klasifikasi.

Tabel 3.6 Hasil Seleksi Fitur *RFE*

No	1	2	3	4	5	6	7	8	9	10
Fitur	ZS TB/U	ZS BB/U	BB/U	Berat	ZS BB/TB	BB/TB	Tinggi	LiLA	Naik Berat Badan	JK
Nilai	0.647	0.157	0.094	0.031	0.030	0.196	0.155	0.001	0.0006	0.0005

Berdasarkan tabel di atas, dapat dilihat bahwa fitur-fitur yang memiliki pengaruh terbesar terhadap klasifikasi stunting adalah ZS TB/U, ZS BB/U, dan BB/U. ZS TB/U memiliki nilai kepentingan tertinggi, yaitu 0.647, yang berarti fitur ini memberikan kontribusi paling besar dalam model klasifikasi. Terakhir, dilakukan optimasi lebih lanjut pada model dengan menggabungkan seleksi fitur RFE dan optimasi menggunakan *Genetic Algorithm (GA)* untuk mencapai hasil klasifikasi yang lebih optimal.

3.4.1 *Random Forest*

Tahap awal, model *Random Forest* diterapkan untuk mengklasifikasikan data stunting di Kota Samarinda. Kemudian dilakukan pengujian menggunakan teknik validasi silang *10-fold cross-validation* untuk memastikan model diuji secara menyeluruh dan memberikan gambaran akurasi pada data yang berbeda. Setiap lipatan *(fold)* berfungsi sebagai data uji satu

kali. Hasil akurasi dari setiap fold dicatat untuk mengevaluasi kinerja model dalam mengklasifikasikan data stunting.

Tabel 3.7 Hasil Akurasi *Random Forest*

Fold	Akurasi
1	70.97%
2	86.08%
3	99.69%
4	89.48%
5	96.84%
6	91.66%
7	94.29%
8	96.66%
9	99.29%
10	94.23%

Dari hasil pengujian tersebut, terlihat bahwa akurasi model bervariasi pada setiap lipatan, dengan akurasi terendah sebesar 70.96% dan akurasi tertinggi mencapai 99.68%. Untuk lebih jelasnya di jelaskan dalam table berikut.

Tabel 3.8 Penjelasan *Random Forest K-Fold* = 10

Fold	Keterangan
1	Akurasi sebesar 70.95%, menunjukkan performa yang cukup baik meskipun merupakan akurasi terendah di antara semua fold.
2	Akurasi meningkat menjadi 86.07%, menunjukkan performa yang lebih baik.
3	Akurasi mencapai puncaknya di 99.67%, menunjukkan bahwa model hampir sempurna dalam klasifikasi pada data ini.
4	Akurasi sedikit menurun menjadi 89.47%, tetapi masih menunjukkan performa yang kuat.
5	Akurasi sangat tinggi sebesar 96.86%, mendekati performa optimal.
6	Akurasi sebesar 91.66%, tetap menunjukkan performa yang sangat baik.

7	Akurasi sebesar 94.27%, kembali menunjukkan performa yang kuat.
8	Akurasi sangat tinggi sebesar 96.67%, mendekati performa optimal.
9	Akurasi hampir sempurna sebesar 99.28%, sangat mendekati performa terbaik.
10	Akurasi sebesar 94.28%, menunjukkan hasil yang konsisten dan kuat.

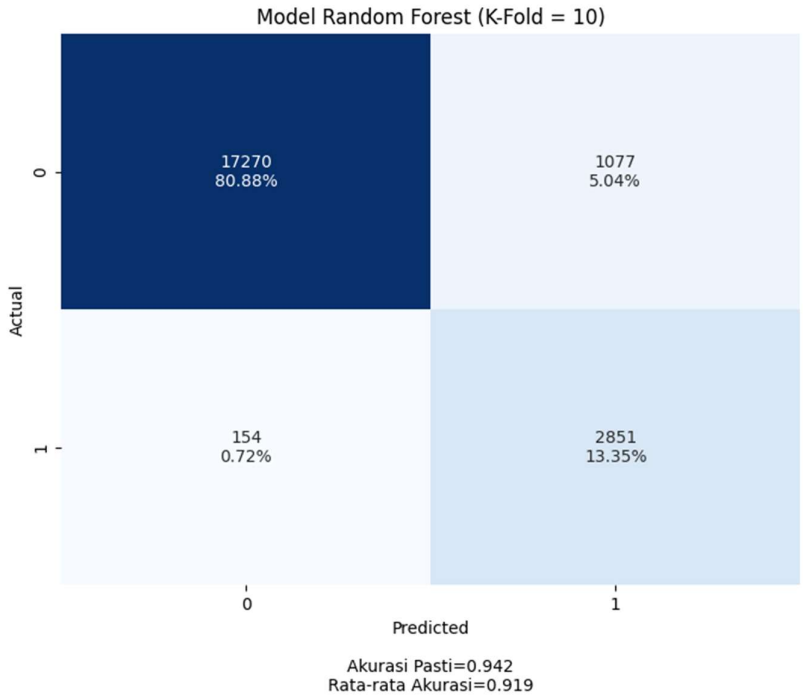
$$Rata - Rata\ Accuracy = \frac{919.19}{10} = 91.919\%$$

Rata-rata akurasi dari seluruh lipatan menunjukkan bahwa model Random Forest memiliki kinerja yang baik dalam mengklasifikasikan data stunting. Akurasi rata-rata model Random Forest dalam mengklasifikasikan data stunting Kota Samarinda Tahun 2023 adalah sebesar 91.91%.

Tabel 3.9 *Confusion Matrix Random Forest*

	Predicted Positive (0)	Predicted Negative (1)
Actual Positive (0)	17270	1077
Actual Negative (1)	154	2851

Tabel 3.7 menunjukkan Confusion Matrix untuk model Random Forest. Matriks ini membantu mengukur performa model klasifikasi dengan menunjukkan jumlah prediksi yang benar dan salah untuk masing-masing kelas. Dari tabel tersebut, terlihat bahwa ada 17,270 kasus True Positive (TP) di mana model memprediksi "Positive" (tidak stunting) dengan benar, 1,077 kasus False Negative (FN) di mana model salah memprediksi "Negative" (stunting), 154 kasus False Positive (FP) di mana model salah memprediksi "Positive" (tidak stunting), dan 2,851 kasus True Negative (TN) di mana model memprediksi "Negative" (stunting) dengan benar. Matriks ini digunakan untuk menghitung akurasi model, yang menunjukkan bahwa model memiliki tingkat keandalan yang tinggi dalam mengklasifikasikan data stunting.



Gambar 3.3 *Confusion Matrix Random Forest*

$$Accuracy = \frac{17270 + 2851}{17270 + 2851 + 154 + 1077} \times 100\% = 94.23\%$$

Dengan demikian, berdasarkan perhitungan manual diatas dan juga dengan perhitungan menggunakan kode *Python* akurasinya adalah 94.23%. Angka ini menunjukkan bahwa model *Random Forest* memiliki tingkat keandalan yang tinggi dalam mengklasifikasikan data stunting. Akurasi sebesar 94.23% mengindikasikan bahwa model mampu mengklasifikasikan sebagian besar data dengan benar, dengan kesalahan yang relatif kecil.

3.4.2 *Random Forest* Dengan Seleksi Fitur *Recursive Feature Elimination (RFE)*

Setelah mendapatkan hasil awal dari penerapan metode *Random Forest*, langkah selanjutnya adalah meningkatkan performa model dengan teknik seleksi fitur. *Recursive Feature Elimination (RFE)* digunakan untuk memilih fitur-fitur yang paling relevan dan berpengaruh dalam klasifikasi data stunting. *RFE* bekerja dengan menghapus fitur secara berulang-ulang dan membangun model pada fitur yang tersisa untuk mengidentifikasi kombinasi fitur terbaik. Dengan menerapkan *RFE*, diharapkan model *Random Forest* akan lebih efisien dan akurat dalam melakukan klasifikasi. Proses ini melibatkan evaluasi kinerja model setelah seleksi fitur dan membandingkannya dengan model dasar untuk menentukan peningkatan yang diperoleh.

Tabel 3.10 Hasil Akurasi *Random Forest* + *RFE*

Fold	Akurasi
1	81.72%

2	84.93%
3	99.51%
4	86.45%
5	98.53%
6	93.52%
7	100%
8	95.00%
9	99.66%
10	97.10%

Dari hasil pengujian tersebut, terlihat bahwa akurasi model bervariasi pada setiap lipatan, dengan akurasi terendah sebesar 81.72% dan akurasi tertinggi mencapai 100%. Untuk lebih jelasnya di jelaskan dalam table berikut.

Tabel 3.11 Penjelasan *RF + RFE K-Fold = 10*

Fold	Keterangan
1	Akurasi sebesar 81.72%, menunjukkan performa yang cukup baik pada subset data ini.
2	Akurasi meningkat menjadi 84.93%, menunjukkan perbaikan kinerja model.
3	Akurasi mencapai 99.51%, mendekati performa optimal dan menunjukkan kemampuan model yang sangat baik dalam mengklasifikasikan data pada subset ini.
4	Akurasi sebesar 86.45%, mempertahankan performa yang kuat.
5	Akurasi sangat tinggi sebesar 98.53%, mendekati akurasi optimal.
6	Akurasi sebesar 93.52%, menunjukkan kinerja yang sangat baik.
7	Akurasi sempurna sebesar 100%, menunjukkan bahwa model mampu mengklasifikasikan semua data dengan benar pada subset ini.
8	Akurasi sebesar 95.00%, kembali menunjukkan performa yang kuat.
9	Akurasi sangat tinggi sebesar 99.66%, sangat

	mendekati performa terbaik.
10	Akurasi sebesar 97.10%, menunjukkan hasil yang konsisten dan kuat.

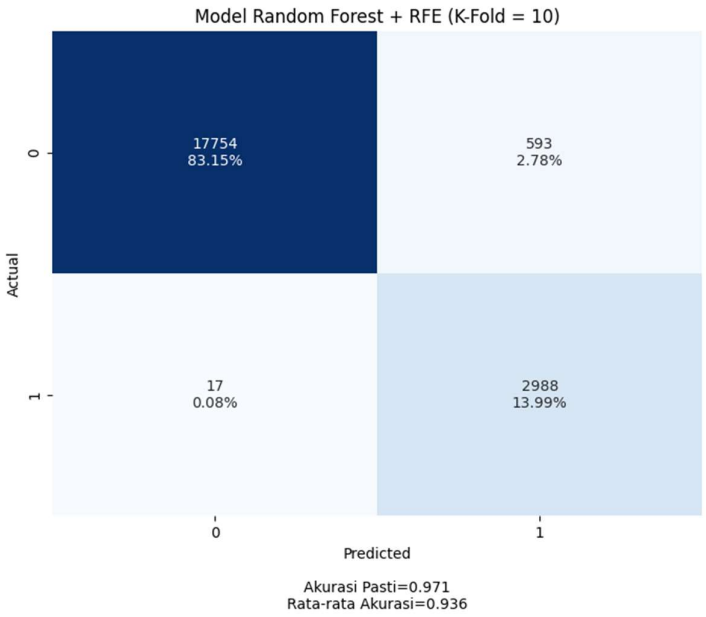
$$Rata - Rata\ Accuracy = \frac{936.42}{10} = 93.64\%$$

Rata-rata akurasi dari seluruh lipatan menunjukkan bahwa model Random Forest dengan seleksi fitur Recursive Feature Elimination memiliki kinerja yang baik dalam mengklasifikasikan data stunting. Akurasi rata-rata model *RF + RFE* dalam mengklasifikasikan data stunting Kota Samarinda Tahun 2023 adalah sebesar 93.64%.

Tabel 3.12 *Confusion Matrix RF + RFE*

	Predicted Positive (0)	Predicted Negative (1)
Actual Positive (0)	17754	593
Actual Negative (1)	17	2988

Tabel 3.10 menunjukkan Confusion Matrix untuk model Random Forest (RF) yang telah menggunakan Recursive Feature Elimination (RFE). Matriks ini menunjukkan performa model dengan rincian sebagai berikut: 17,754 kasus True Positive (TP) di mana model memprediksi "Positive" (tidak stunting) dengan benar, 593 kasus False Negative (FN) di mana model salah memprediksi "Negative" (stunting), 17 kasus False Positive (FP) di mana model salah memprediksi "Positive" (tidak stunting), dan 2,988 kasus True Negative (TN) di mana model memprediksi "Negative" (stunting) dengan benar. Confusion Matrix ini menunjukkan peningkatan akurasi dan keandalan model dalam mengklasifikasikan data stunting setelah penerapan seleksi fitur RFE.



Gambar 3.4 *Confusion Matrix RF + RFE*

$$Accuracy = \frac{17754 + 2988}{17754 + 2988 + 17 + 593} \times 100\% = 97.10\%$$

Dengan demikian, berdasarkan perhitungan manual diatas dan juga dengan perhitungan menggunakan kode *Python* akurasinya adalah 97.10%. Angka ini menunjukkan bahwa model *Random Forest* dengan seleksi fitur Recursive Feature Elimination memiliki tingkat keandalan yang tinggi dalam mengklasifikasikan data stunting. Akurasi sebesar 97.10% mengindikasikan bahwa model mampu mengklasifikasikan sebagian besar data dengan benar, dengan kesalahan yang relatif kecil.

3.4.3 *Random Forest* Dengan Seleksi Fitur *RFE Dan Optimasi GA (Genetic Algorithm)*

Pada tahap ini, model *Random Forest* dikombinasikan dengan teknik seleksi fitur *Recursive Feature Elimination (RFE)* dan optimasi *Genetic Algorithm (GA)* untuk meningkatkan akurasi klasifikasi data stunting di Kota Samarinda. Proses ini melibatkan dua langkah utama: pertama, seleksi fitur menggunakan *RFE* untuk memilih fitur-fitur yang paling relevan, dan kedua, optimasi *hyperparameter* model menggunakan *Genetic Algorithm* untuk menemukan kombinasi parameter yang optimal. Kombinasi dari kedua teknik ini bertujuan untuk meningkatkan kinerja model secara keseluruhan dengan mengurangi kompleksitas data dan menemukan konfigurasi terbaik untuk model *Random Forest*.

Tabel 3.13 Hasil Akurasi *Random Forest + RFE + GA*

Fold	Akurasi
1	99.62%
2	99.73%

3	89.31%
4	95.50%
5	99.73%
6	99.82%
7	99.50%
8	100%
9	99.99%
10	99.70%

Dari hasil pengujian tersebut, terlihat bahwa akurasi model bervariasi pada setiap lipatan, dengan akurasi terendah sebesar 89.31% dan akurasi tertinggi mencapai 100%. Untuk lebih jelasnya di jelaskan dalam table berikut.

Tabel 3.14 Penjelasan *RF + RFE + GA K-Fold = 10*

Fold	Keterangan
1	Akurasi sangat tinggi sebesar 99.62%, menunjukkan model yang sangat baik.
2	Akurasi sedikit meningkat menjadi 99.73%, mempertahankan performa yang sangat kuat.
3	Akurasi lebih rendah di 89.31%, tetapi masih dalam kisaran yang baik.
4	Akurasi sebesar 95.50%, menunjukkan peningkatan performa yang signifikan.
5	Akurasi kembali sangat tinggi sebesar 99.73%.
6	Akurasi mencapai 99.82%, mendekati performa sempurna.
7	Akurasi sebesar 99.50%, tetap menunjukkan hasil yang sangat baik.
8	Akurasi sempurna sebesar 100%, menunjukkan bahwa model mampu mengklasifikasikan semua data dengan benar pada subset ini.
9	Akurasi hampir sempurna sebesar 99.99%.
10	Akurasi sangat tinggi sebesar 99.70%, menunjukkan hasil yang konsisten.

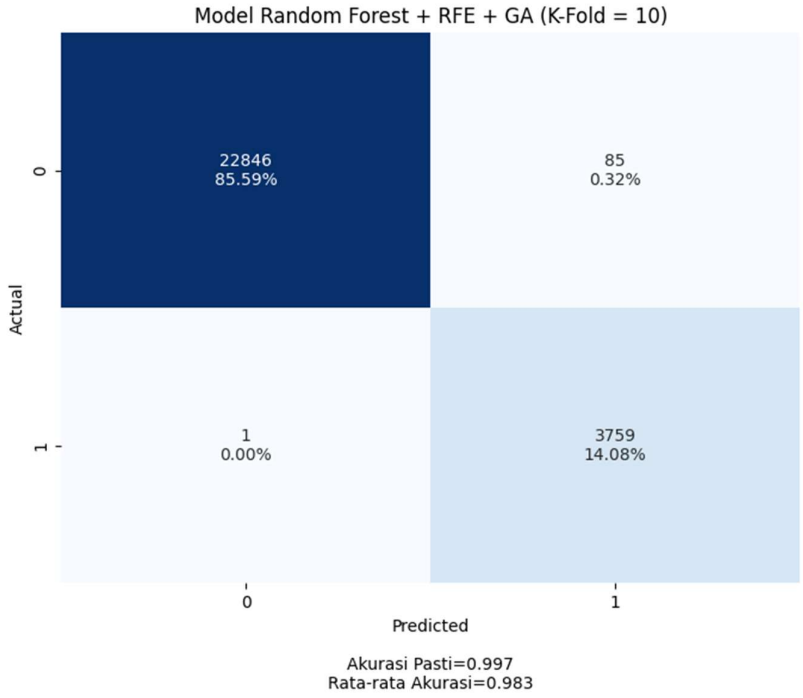
$$Rata - Rata\ Accuracy = \frac{983.90}{10} = 98.39\%$$

Rata-rata akurasi dari seluruh lipatan adalah sekitar 98.39%, yang menunjukkan bahwa kombinasi model *Random Forest* dengan seleksi fitur *RFE* dan optimasi *GA* memiliki kinerja yang sangat baik dalam mengklasifikasikan data stunting. Hasil ini mencerminkan peningkatan yang signifikan dan keandalan model dalam berbagai subset data, dengan akurasi yang sangat tinggi di setiap fold. Performa model yang kuat ini memberikan kepercayaan yang tinggi terhadap kemampuannya dalam aplikasi nyata untuk mengidentifikasi kasus stunting di Kota Samarinda.

Tabel 3.15 *Confusion Matrix RF + RFE + GA*

	Predicted Positive (0)	Predicted Negative (1)
Actual Positive (0)	22846	85
Actual Negative (1)	1	3759

Tabel 3.13 menunjukkan Confusion Matrix untuk model Random Forest (RF) yang telah menggunakan Recursive Feature Elimination (RFE) dan Genetic Algorithm (GA). Matriks ini menunjukkan hasil sebagai berikut: 22,846 kasus True Positive (TP) di mana model memprediksi "Positive" (tidak stunting) dengan benar, 85 kasus False Negative (FN) di mana model salah memprediksi "Negative" (stunting), 1 kasus False Positive (FP) di mana model salah memprediksi "Positive" (tidak stunting), dan 3,759 kasus True Negative (TN) di mana model memprediksi "Negative" (stunting) dengan benar. Confusion Matrix ini menunjukkan performa tinggi dan peningkatan signifikan dalam akurasi dan keandalan model setelah penerapan RFE dan optimasi GA.



Gambar 3.5 *Confusion Matrix RF + RFE + GA*

$$Accuracy = \frac{22846 + 3759}{22846 + 3759 + 1 + 85} \times 100\% = 99.70\%$$

Dengan demikian, berdasarkan perhitungan manual diatas dan juga dengan perhitungan menggunakan kode *Python* akurasinya adalah 98.39%. Angka ini menunjukkan bahwa model *Random Forest* dengan seleksi fitur *RFE* dan optimasi *GA* memiliki tingkat keandalan yang tinggi dalam mengklasifikasikan data stunting. Akurasi sebesar 99.70% mengindikasikan bahwa model mampu mengklasifikasikan sebagian besar data dengan benar, dengan kesalahan yang relatif kecil.

3.4.4 Perbandingan Hasil Akurasi

Pada bagian ini, akan dilakukan perbandingan hasil akurasi dari tiga metode yang telah diterapkan untuk mengklasifikasikan data stunting di Kota Samarinda, yaitu *Random Forest (RF), Random Forest* dengan Seleksi Fitur *RFE (RF+RFE),* dan *Random Forest* dengan Seleksi Fitur *RFE* serta *Optimasi GA (RF+RFE+GA).* Hasil akurasi dari setiap metode akan ditampilkan dalam bentuk tabel untuk memudahkan analisis dan evaluasi. Dengan membandingkan ketiga metode ini, dapat diketahui seberapa besar peningkatan performa yang diperoleh dari setiap tahapan seleksi fitur dan optimasi yang dilakukan. Tabel berikut menyajikan ringkasan hasil akurasi dari masing-masing metode:

Tabel 3.16 Perbandingan Hasil Akurasi Sesudah Penggunaan *RFE* & *GA*

Fold	*RF*	RF + RFE	RF + RFE + GA	Perubahan RF ke RF-RFE	Perubahan RF ke RF-RFE-GA	Perubahan RF-RFE ke RF-RFE-GA
1	70.97%	81.72%	99.62%	10.75%	28.65%	17.90%
2	86.08%	84.93%	99.73%	-1.15%	13.65%	14.80%
3	99.69%	99.51%	89.31%	-0.18%	-10.38%	-10.20%
4	89.48%	86.45%	95.50%	-3.03%	6.02%	9.05%
5	96.84%	98.53%	99.73%	1.69%	2.89%	1.20%
6	91.66%	93.52%	99.82%	1.86%	8.16%	6.30%
7	94.29%	100%	99.50%	5.71%	5.21%	-0.50%
8	96.66%	95.00%	100%	-1.66%	3.34%	5%
9	99.29%	99.66%	99.99%	0.37%	0.70%	0.33%
10	94.23%	97.10%	99.70%	2.87%	5.47%	2.60%

Tabel 3.15 menunjukkan perbandingan hasil klasifikasi menggunakan *Random Forest (RF), Random Forest* dengan Seleksi Fitur *(RF + RFE*), dan *Random Forest* dengan Seleksi Fitur dan Optimasi (*RF + RFE + GA*) pada 10 *fold* validasi, dengan sebagian besar *fold* menunjukkan peningkatan akurasi setelah seleksi fitur dan optimasi, meskipun terdapat beberapa penurunan akurasi pada *fold* tertentu yang dapat disebabkan oleh fitur yang tidak relevan, *overfitting*, atau variasi acak dalam data.

Tabel 3.17 Perbandingan Hasil Rata-Rata Akurasi Sesudah Penggunaan *RFE* & *GA*

Average Accuracy	*RF*	RF + RFE	RF + RFE + GA	Perubahan RF ke RF-RFE	Perubahan RF ke RF-RFE-GA	Perubahan RF-RFE ke RF-RFE-GA
	91.91%	93.64%	98.39%	1.73%	6.48%	4.75%

Tabel 3.15 dan 3.16 menunjukkan perbandingan hasil klasifikasi data stunting di Kota Samarinda menggunakan tiga metode berbeda: *Random Forest (RF), Random Forest* dengan Seleksi Fitur *RFE (RF + RFE),* dan *Random Forest* dengan Seleksi Fitur *RFE* serta Optimasi *GA (RF + RFE + GA).* Tabel ini memberikan gambaran jelas tentang bagaimana setiap metode mempengaruhi akurasi klasifikasi, Berikut adalah penjelasan tiap hasilnya.

a) Pada metode *RF* dasar, rata-rata akurasi yang diperoleh adalah 91.91%, sedangkan akurasi pasti mencapai 94.23%. Dengan penambahan seleksi fitur menggunakan *RFE*, rata-rata akurasi meningkat menjadi 93.64%, yang menunjukkan peningkatan sebesar 1.73%. Akurasi pasti juga meningkat menjadi 97.10%, menunjukkan peningkatan sebesar 2.87%.

b) Selanjutnya, dengan menggabungkan *RFE* dan optimasi *GA*, rata-rata akurasi meningkat signifikan menjadi 98.39%, menunjukkan peningkatan sebesar 4.75% dibandingkan metode RF dasar. Akurasi pasti mencapai nilai tertinggi sebesar 99.70%, dengan peningkatan sebesar 2.60% dibandingkan metode *RF + RFE*.

3.5 Pembahasan

Penelitian ini telah melalui serangkaian tahapan mulai dari pengumpulan data, pembersihan data, transformasi data, hingga pemodelan menggunakan berbagai teknik machine learning untuk mengklasifikasikan data stunting di Kota Samarinda. Hasil dari setiap tahapan ini menunjukkan peningkatan akurasi yang signifikan seiring dengan penerapan teknik seleksi fitur dan optimasi. Data awal terdiri dari 27 atribut yang diperoleh dari Dinas Kesehatan Kota Samarinda. Melalui proses seleksi fitur, atribut yang kurang relevan dihapus, menyisakan 12 atribut sebagai fitur dan 1 atribut sebagai target dalam klasifikasi stunting. Proses seleksi ini bertujuan untuk meningkatkan efisiensi dan akurasi model dengan fokus pada atribut yang signifikan. Data yang diperoleh melalui proses pembersihan untuk menghapus data yang tidak lengkap dan terduplikasi. Selanjutnya, transformasi data dilakukan untuk mengubah atribut kategorikal menjadi numerik, sehingga dapat diolah oleh algoritma machine learning.

Model awal menggunakan *Random Forest* menunjukkan akurasi yang cukup baik dengan rata-rata akurasi sebesar 91.91% dan akurasi pasti 94.23%. Ini menunjukkan bahwa model dasar sudah cukup efektif dalam mengklasifikasikan data stunting. Penggunaan *RFE* untuk seleksi fitur meningkatkan akurasi model. Rata-rata akurasi meningkat menjadi 93.64%, dengan akurasi pasti 97.10%. Ini menunjukkan bahwa *RFE* membantu dalam mengidentifikasi fitur yang paling relevan, sehingga meningkatkan performa model. Kombinasi *RFE* dan optimasi *GA* menghasilkan peningkatan akurasi yang signifikan. Rata-rata akurasi mencapai 98.39% dengan akurasi pasti 99.70%. Ini menunjukkan bahwa *GA* efektif dalam menemukan konfigurasi *hyperparameter* yang optimal untuk model *Random Forest.*

Perbandingan dengan penelitian terdahulu menunjukkan peningkatan yang signifikan dalam performa model. Sebagai contoh, penelitian sebelumnya oleh (Obvious et al. 2023) menggunakan algoritma Support Vector Classification dan XGBoost dengan akurasi masing-masing sekitar 64% dan 63%. Selain itu, penelitian (Estiyak et al. 2023) menunjukkan bahwa algoritma Random Forest, Naive Bayes, dan Logistic Regression mengalami penurunan performa saat dihadapkan pada data berdimensi tinggi, dengan akurasi masing-masing 60%, 58%, dan 58%. Dalam konteks data berdimensi tinggi, penelitian ini menunjukkan bahwa penggunaan teknik seleksi fitur RFE dan optimasi GA mampu meningkatkan akurasi model secara signifikan dibandingkan dengan pendekatan tradisional yang digunakan dalam penelitian terdahulu. (Leykun et al. (2024) menunjukkan bahwa penggunaan RFE dapat meningkatkan akurasi dari 72.41% menjadi 80.1% untuk metode Random Forest, sementara penelitian ini lebih jauh menunjukkan bahwa dengan tambahan optimasi GA, akurasi dapat meningkat hingga 99.70%. Dan untuk menjawab rumusan masalah pada bab1 berikut penjelasannya:

a) Penelitian ini bertujuan untuk mengidentifikasi fitur-fitur yang paling relevan dan signifikan dalam klasifikasi data stunting. Proses seleksi fitur menggunakan *RFE* diharapkan dapat mengeliminasi fitur yang kurang relevan, sehingga model dapat bekerja lebih efisien dan akurat. Dalam konteks ini, fitur-fitur yang memiliki pengaruh terbesar terhadap klasifikasi stunting adalah ZS TB/U, ZS BB/U, dan BB/U. ZS TB/U memiliki nilai kepentingan tertinggi, yaitu 0.647, yang berarti fitur ini memberikan kontribusi paling besar dalam model klasifikasi. Berikut ini adalah tabel yang memperlihatkan perbandingan fitur yang digunakan dalam penelitian ini dengan penelitian lain:

Tabel 3.18 Perbandingan *RFE* Dengan Penelitian lain

Penelitian	Metode	Data	Fitur	Akurasi
Penelitian Ini	*RF + RFE*	Stunting	ZS TB/U, ZS BB/U, dan BB/U.	94.23% - 97.10% (+ 2.87%)
(Yoga et al., 2021)	*SVM + Chi-Square*	Stunting	ZS TB/U, BB/U, ZS BB/U, Tinggi, LiLA, dan Berat.	96.50% - 96.60% (+ 0.10%)
(Kemal et al,. 2023)	*K-NN + Relif-F*	Stunting	Umur, Berat, Tinggi, BB/U, ZS BB/U, BB/TB, ZS BB/TB, dan ZS TB/U.	91.90% - 92.20% (+ 0.30%)
(Syahrani dan Dwi, 2022)	*K-NN + Backward Elimination*	Stunting	Usia Saat Ukur, Berat, Tinggi, ZS BB/U dan ZS TB/U.	95.17% - 90.23% (- 4.94%)

b) Setelah seleksi fitur dilakukan menggunakan *RFE*, langkah berikutnya adalah optimasi model menggunakan *GA* untuk meningkatkan akurasi prediksi. Hasil penelitian menunjukkan bahwa seleksi fitur menggunakan metode *RFE* terbukti efektif dalam meningkatkan kinerja model *Random Forest*. Dengan mengidentifikasi dan mengeliminasi fitur yang kurang relevan, rata-rata akurasi model meningkat dari 91.91% menjadi 93.64%. Akurasi pasti juga mengalami

peningkatan dari 94.23% menjadi 97.10%. Optimasi lebih lanjut dengan menggunakan *Genetic Algorithm (GA)* setelah seleksi fitur *RFE* memberikan peningkatan akurasi yang signifikan. Rata-rata akurasi model mencapai 98.39%, dan akurasi pasti mencapai 99.70%. Kombinasi *RFE* dan *GA* memungkinkan model *Random Forest* untuk mencapai performa maksimal dalam mengklasifikasikan data stunting, menunjukkan bahwa teknik seleksi fitur dan optimasi hyperparameter yang digunakan secara bersamaan dapat secara efektif mengatasi kompleksitas data dan meningkatkan keandalan prediksi stunting.Berikut ini adalah tabel yang memperlihatkan perbandingan fitur yang digunakan dalam penelitian ini dengan penelitian lain:

Tabel 3.19 Perbandingan *GA* Dengan Penelitian lain

Penelitian	Metode	Akurasi
Penelitian Ini	*RF + RFE + GA*	99.70%
(Monire et al,. 2021)	*PSO + RNN*	96.08%
	RF + SMOTE	98.31%
	PSO + KNN	98.9%
	EGA + PSO	98.97%
	PSO + RF	99.76%
	GA + RF	99.99%

BAB IV

KESIMPULAN DAN SARAN

4.1 Kesimpulan

Berdasarkan hasil analisis dan pembahasan yang telah dilakukan, penelitian ini menunjukkan efektivitas teknik *RFE* dan *GA* dalam meningkatkan akurasi serta efisiensi model machine learning pada data berdimensi tinggi.

a) Efektivitas Seleksi Fitur Menggunakan *RFE*

Seleksi fitur menggunakan metode RFE terbukti efektif dalam meningkatkan kinerja model Random Forest, dengan mengidentifikasi dan mengeliminasi fitur yang kurang relevan sehingga fitur-fitur yang memiliki pengaruh terbesar terhadap klasifikasi stunting adalah ZS TB/U, ZS BB/U, dan BB/U, yang mengakibatkan peningkatan rata-rata akurasi model dari 91.91% menjadi 93.64%, serta peningkatan akurasi pasti dari 94.23% menjadi 97.10% Ini menunjukkan bahwa pemilihan atribut yang tepat sangat penting untuk meningkatkan efisiensi dan akurasi klasifikasi data stunting.

b) Peningkatan Signifikan dengan Kombinasi *RFE* dan *GA*

Optimasi lebih lanjut dengan menggunakan *Genetic Algorithm (GA)* setelah seleksi fitur *RFE* memberikan peningkatan akurasi yang signifikan. Rata-rata akurasi model mencapai 98.39%, dan akurasi pasti mencapai 99.70%. Kombinasi *RFE* dan *GA* memungkinkan model Random Forest untuk mencapai performa maksimal dalam mengklasifikasikan data stunting, menunjukkan bahwa teknik seleksi fitur dan optimasi hyperparameter yang digunakan secara bersamaan dapat secara efektif mengatasi kompleksitas data dan meningkatkan keandalan prediksi stunting.

c) Implikasi Penelitian

Penelitian ini memiliki implikasi signifikan dalam pengelolaan dan analisis data kesehatan, khususnya terkait masalah stunting di Kota Samarinda. Dengan memanfaatkan Random Forest sebagai model dasar yang dilengkapi dengan teknik seleksi fitur dan optimasi seperti RFE dan GA, instansi kesehatan dapat mengembangkan model prediksi yang lebih akurat dan efisien. Hal ini pada akhirnya dapat membantu dalam pengambilan keputusan yang lebih baik. Akurasi yang lebih tinggi dalam klasifikasi stunting memungkinkan deteksi dini dan intervensi yang lebih tepat sasaran, yang dapat memberikan dampak positif terhadap upaya penanggulangan stunting. Selain itu, teknik ini juga dapat diterapkan pada bidang lain yang memerlukan analisis data berdimensi tinggi untuk meningkatkan hasil prediktif dan operasional.

4.2 Saran

Berdasarkan hasil penelitian dan analisis yang telah dilakukan, terdapat beberapa saran yang diharapkan dapat membantu meningkatkan efektivitas penelitian dan implementasi hasil dalam klasifikasi data stunting di Kota Samarinda:

a) Peningkatan Jumlah Data

Untuk penelitian selanjutnya, disarankan agar jumlah data yang digunakan lebih banyak lagi agar model dapat dilatih dengan lebih baik dan hasil yang diperoleh lebih representatif. Penggunaan data dari berbagai tahun juga dapat membantu dalam memahami tren dan pola stunting yang lebih luas.

b) Penggunaan Algoritma Lain

Meskipun *Random Forest* telah menunjukkan performa yang baik, disarankan untuk mencoba algoritma lain seperti *Gradient Boosting*, *AdaBoost*, atau *Deep Learning* untuk melihat apakah ada peningkatan lebih lanjut dalam akurasi klasifikasi.

c) Penggunaan Teknik Lain untuk Seleksi Fitur

Selain *RFE,* teknik seleksi fitur lainnya seperti Lasso Regression atau *Principal Component Analysis (PCA)* dapat digunakan untuk mengevaluasi apakah teknik tersebut memberikan hasil yang lebih baik dalam meningkatkan akurasi model.

d) Optimasi *Hyperparameter* yang Lebih Mendalam

Meskipun *Genetic Algorithm* telah digunakan untuk optimasi, penerapan teknik lain seperti *Grid Search* atau *Bayesian Optimization* dapat dicoba untuk menemukan kombinasi hyperparameter yang lebih optimal dan meningkatkan kinerja model.

e) Penerapan Teknik Penanganan Ketidakseimbangan Kelas

Jika terdapat ketidakseimbangan dalam jumlah data antara kelas yang berbeda, teknik seperti *SMOTE (Synthetic Minority Over-sampling Technique)* atau *undersampling* dapat digunakan untuk memastikan model tidak bias terhadap kelas mayoritas.

f) Pembuatan Dashboard Prediksi

Mengembangkan dashboard interaktif untuk visualisasi hasil prediksi model machine learning. Hal ini dapat membantu pihak terkait untuk memahami data dengan lebih baik dan membuat keputusan berdasarkan data yang lebih tepat dan cepat.

Dengan mengikuti saran-saran ini, diharapkan penelitian dan implementasi hasilnya dapat memberikan kontribusi yang lebih besar dalam meningkatkan akurasi dan efisiensi klasifikasi data stunting di Kota Samarinda serta memanfaatkan teknologi machine learning dengan lebih optimal.

DAFTAR RUJUKAN

Rufina Hurai, Remita Ully Hutagalung, Tambunan Heriyanto Mugabe. (2023) Pengetahuan Tentang Stunting Di Posyandu Wilayah Kerja Puskesmas Makroman Samarinda, Caritas Et Fraternitas: Jurnal Kesehatan Volume 2, Nomor 1, Juli 2023 EISSN: 2961-7243.

Dhyan Chandra Yadav, Saurabh Pal. Prediction of Heart Disease Using Feature Selection and Random Forest Ensemble Method . ISSN 0975-2366 DOI:https://doi.org/10.31838/ijpr/2020.12.04.013

Akbarifar, A., & Maghsoudpour, A. (2024). A Novel Approach to Dementia Prediction Leveraging Recursive Feature Elimination and Decision Tree A Novel Approach to Dementia Prediction Leveraging Recursive Feature Elimination and Decision Tree.

Alifah, Siswantining, T., Sarwinda, D., & Bustamam, A. (2020). *RFE and Chi-Square Based Feature Selection Approach for Detection of Diabetic Retinopathy*. *196*(Ijcse), 380–386. https://doi.org/10.2991/aer.k.201124.069

Bitew, F. H., Sparks, C. S., & Nyarko, S. H. (2022). Machine learning algorithms for predicting undernutrition among under-five children in Ethiopia. *Public Health Nutrition*, *25*(2), 269–280. https://doi.org/10.1017/S1368980021004262

Chilyabanyama, O. N., Chilengi, R., Simuyandi, M., Chisenga, C. C., Chirwa, M., Hamusonde, K., Saroj, R. K., Iqbal, N. T., Ngaruye, I., & Bosomprah, S. (2022). Performance of Machine Learning Classifiers in Classifying Stunting among Under-Five Children in Zambia. *Children*, *9*(7). https://doi.org/10.3390/children9071082

Conn, D., Ngun, T., Li, G., & Ramirez, C. M. (2019). Fuzzy forests: Extending random forest feature selection for correlated, high-dimensional data. *Journal of Statistical Software*, *91*(9). https://doi.org/10.18637/jss.v091.i09

Feldner-Busztin, D., Nisantzis, P. F., Edmunds, S. J., Boza, G., Racimo, F., Gopalakrishnan, S., Limborg, M. T., Lahti, L., & de Polavieja, G. G. (2023). Dealing with dimensionality: the application of machine learning to multi-omics data. *Bioinformatics*, *39*(2). https://doi.org/10.1093/bioinformatics/btad021

Gebeye, L. G., Dessie, E. Y., & Yimam, J. A. (2023). Predictors of micronutrient deficiency among children aged 6–23 months in Ethiopia: a machine learning approach. *Frontiers in Nutrition*, *10*(January), 1–13. https://doi.org/10.3389/fnut.2023.1277048

Khan, M. N. A., & Yunus, R. M. (2023). A hybrid ensemble approach to accelerate the classification accuracy for predicting malnutrition among under-five children in sub-Saharan African countries. *Nutrition*, *108*. https://doi.org/10.1016/j.nut.2022.111947

Kim, D. W., Shin, G. Y., & Han, M. M. (2020). Analysis of feature importance and interpretation for malware classification. *Computers, Materials and Continua*, *65*(3), 1891–1904. https://doi.org/10.32604/cmc.2020.010933

Kumar, C. J., & Das, P. R. (2022). The diagnosis of ASD using multiple machine learning techniques. *International Journal of Developmental Disabilities*, *68*(6), 973–983. https://doi.org/10.1080/20473869.2021.1933730

Lakshmanaprabu, S. K., Shankar, K., Ilayaraja, M., Nasir, A. W., Vijayakumar, V., & Chilamkurti, N. (2019). Random forest for big data classification in the internet of things using optimal features. *International Journal of Machine Learning and Cybernetics*, *10*(10), 2609–2618. https://doi.org/10.1007/s13042-018-00916-z

Norouzi, M., Gürkaş-Aydın, Z., Turna, Ö. C., Yağci, M. Y., Aydin, M. A., & Souri, A. (2023). A Hybrid Genetic Algorithm-Based Random Forest Model for Intrusion Detection Approach in Internet of Medical Things. *Applied Sciences*, *13*(20), 11145. https://doi.org/10.3390/app132011145

Pratiwi, I. G. (2023). Studi Literatur: Intervensi Spesifik Penanganan Stunting. *Indonesian Health Issue*, *2*(1), 29–37. https://doi.org/10.47134/inhis.v2i1.43

Rahmi Susanti. (2023). Analisis Faktor Maternal Terhadap Keluarga Berisiko Stunting Sebagai Upaya Peningkatan Analisis Data di BKKBN Kalimantan Timur. *Jurnal Nasional Pengabdian Masyarakat*, *4 (1)*(1), 7–17.

Rahnenführer, J., De Bin, R., Benner, A., Ambrogi, F., Lusa, L., Boulesteix, A. L., Migliavacca, E., Binder, H., Michiels, S., Sauerbrei, W., & McShane, L. (2023). Statistical analysis of high-dimensional biomedical data: a gentle introduction to analytical goals, common approaches and challenges. *BMC Medicine*, *21*(1), 1–54. https://doi.org/10.1186/s12916-023-02858-y

Senan, E. M., Al-Adhaileh, M. H., Alsaade, F. W., Aldhyani, T. H. H., Alqarni, A. A., Alsharif, N., Uddin, M. I., Alahmadi, A. H., Jadhav, M. E., & Alzahrani, M. Y. (2021). Diagnosis of Chronic Kidney Disease Using Effective Classification Algorithms and Recursive Feature Elimination Techniques. *Journal of Healthcare Engineering*, *2021*. https://doi.org/10.1155/2021/1004767

Sun, D., Shi, S., Wen, H., Xu, J., Zhou, X., & Wu, J. (2021). A hybrid optimization method of factor screening predicated on GeoDetector and Random Forest for Landslide Susceptibility Mapping. *Geomorphology*, *379*, 107623. https://doi.org/10.1016/j.geomorph.2021.107623

Togatorop, P. R., Sianturi, M., Simamora, D., & Silaen, D. (2022). Optimizing Random Forest using Genetic Algorithm for Heart Disease Classification. *Lontar Komputer :*

Jurnal Ilmiah Teknologi Informasi, *13*(1), 60. https://doi.org/10.24843/lkjiti.2022.v13.i01.p06

Turjo, E. A., & Rahman, M. H. (2024). Assessing risk factors for malnutrition among women in Bangladesh and forecasting malnutrition using machine learning approaches. *BMC Nutrition*, *10*(1), 1–25. https://doi.org/10.1186/s40795-023-00808-8

Veerapathran, V., Faiza Bait Ali Suleiman, Antonyraj Martin, & Rajesh Menon K. (2023). Genetic Algorithm and Random Forest Classifier Fusion: a Cutting-Edge Approach for Breast Cancer Diagnosis. *International Journal of Information Technology, Research and Applications*, *2*(4), 46–54. https://doi.org/10.59461/ijitra.v2i4.75

Workie Demsash, A. (2023). Using best performance machine learning algorithm to predict child death before celebrating their fifth birthday. *Informatics in Medicine Unlocked*, *40*(August), 101298. https://doi.org/10.1016/j.imu.2023.101298

Yoga Siswa, T., & Pranoto, W. (2023). IMPLEMENTASI SELEKSI FITUR INFORMATION GAIN RATIO PADA ALGORITMA RANDOM FOREST UNTUK MODEL DATA KLASIFIKASI PEMBAYARAN KULIAH. Dinamika Informatika : Jurnal Ilmiah Teknologi Informasi, 15(1), 41-49. https://doi.org/10.35315/informatika.v15i1.9465

Rajabi, K. M., Witanti, W., & Yuniarti, R. (2023). Penerapan Algoritma K-Nearest Neighbor (KNN) Dengan Fitur Relief-F Dalam Penentuan Status Stunting. Innovative: Journal Of Social Science Research, 3(4), 3555–3568. https://doi.org/10.31004/innovative.v3i4.3885

Lonang, Syahrani & Normawati, Dwi. (2022). Klasifikasi Status Stunting Pada Balita Menggunakan K-Nearest Neighbor Dengan Feature Selection Backward Elimination. JURNAL MEDIA INFORMATIKA BUDIDARMA. 6. 49-56. 10.30865/mib.v5i1.2293.

RIWAYAT HIDUP

Penulis bernama Bima Satria, lahir di Waru pada tanggal 19 Agustus 1999, adalah anak keenam dari almarhum Bapak Yusrani dan almarhumah Ibu Masraya. Penulis berkebangsaan Indonesia, Suku Bugis dan Banjar, serta beragama Islam. Penulis memulai pendidikan formalnya di TK 10 November Balikpapan pada tahun 2005, kemudian melanjutkan pendidikan di MI Darussalam Balikpapan pada tahun 2006. Pendidikan dasar Bima dimulai di SDN 001 PPU (2006-2009), dilanjutkan di SDN 020 Balikpapan Utara (2009-2012), dan diakhiri di SD Muhammadiyah 1 PPU (2012-2014). Selanjutnya, Penulis melanjutkan studinya di SMP Muhammadiyah 2 PPU dari tahun 2014 hingga 2017. Setelah menyelesaikan pendidikan SMP, Penulis melanjutkan studinya di SMK Muhammadiyah 1 PPU dan lulus pada tahun 2020. Saat ini, Penulis tercatat sebagai mahasiswa pada Fakultas Sains dan Teknologi, jurusan Teknik Informatika, di Universitas Muhammadiyah Kalimantan Timur, dimulai pada tahun 2020. Selama masa studi, Penulis memiliki pengalaman magang di SMK Muhammadiyah 1 PPU selama 3 bulan yang dilaksanakan pada semester tujuh. Demikian riwayat hidup ini disampaikan dengan harapan dapat memberikan gambaran yang jelas tentang latar belakang pendidikan dan pengalaman Penulis. Penulis berkomitmen untuk terus belajar dan berkontribusi dalam bidang Teknik Informatika, serta berusaha untuk mencapai prestasi akademis yang lebih tinggi demi masa depan yang lebih baik. Penulis juga mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan dan bimbingan selama perjalanan studinya, baik dari keluarga, teman, dosen, maupun institusi pendidikan. Semoga segala upaya dan dedikasi Penulis dapat bermanfaat bagi diri sendiri, keluarga, masyarakat, dan bangsa Indonesia.

LAMPIRAN

Lampiran 1 Surat Permohonan Pengambilan Data

UMKT
Program Studi
Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Muhammadiyah Kalimantan Timur
Berkarakter | Berwawasan | Berkemajuan
Telp. 0541-748511 Fax. 0541-766832
Website http://informatika.umkt.ac.id
email: informatika@umkt.ac.id

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

Nomor : 003-009/FST.1/D.3/C/2024
Lampiran : -
Perihal : **Permohonan Pengambilan Data**

Kepada Yth.
Kepala Dinas Kesehatan Kota Samarinda
di -
Tempat

Assalamu'alaikum Warrahmatullahi Wabarrakatuh

Puji Syukur kepada Allah Subhanahu wa ta'ala yang senantiasa melimpahkan Rahmat-Nya kepada kita sekalian. Aamiin.

Sehubungan untuk memenuhi Tugas Akhir/Skripsi Tahun Akademik 2023/2024, maka dengan ini kami bermaksud untuk melakukan pengambilan data di Dinas Kesehatan Kota Samarinda. Adapun data yang diminta yaitu data penyakit stunting di Kota Samarinda tahun 2023, dengan nama mahasiswa sebagai berikut:

No	Nama	NIM	Program Studi
1	Ari Ahmad Dhani	2011102441090	Teknik Informatika
2	Bima Satria	2011102441102	Teknik Informatika
3	Lidya Sari	2011102441121	Teknik Informatika
4	Mukminatul Munawaroh	2011102441064	Teknik Informatika
5	Siti Muawwanah	2011102441153	Teknik Informatika

Demikian surat permohonan ini dibuat. Atas perhatiannya dan kerjasamanya kami mengucapkan terima kasih.

Wassalamu'alaikum Warrahmatullahi Wabarrakatuh

Samarinda, 4 Ramadhan 1445 H
15 Maret 2024 M

Program Studi S1 Teknik Informatika

Rosnansyah, S.Kom., M.TI
NIDN. 1118019203

Kampus 1 : Jl. Ir. H. Juanda, No.15, Samarinda
Kampus 2 : Jl. Pelita, Pesona Mahakam, Samarinda

Lampiran 2 Tampilan Dataset Stunting Kota Samarinda Tahun 2023

No	Nama	J	BB	TB	Prov	Kab/Kot	Kec	Puskesmas	Desa/Kel	Posyandu	R	R	Usia Saat Ukur	Tanggal Pengukuran	Bera	Tinggi	LiLA	BB/U	Zs	TB/U	ZS	BB/TB	ZS BB/TB	Naik Berat Badan	PMT Diterima (kg)	JmL	KPSP	KIA
1	Dimas	L	3.5	49	Kalti	Samarin	Sungai Pin	Remaja	Temindu	Pulau Ind			0 Tahun - 11 Bulan - 11 Hari	2023-01-02	9.1	74.5	0	Norma	-0.39	Norma	-0.39	Gizi Baik	-0.39	O	-		-	-
2	SITI	P	3	48	Kalti	Samarin	Sungai Pin	Remaja	Temindu	Pulau Ind	3		4 Tahun - 0 Bulan - 16 Hari	2023-01-02	12	94	0	Kuran	-2.25	Pende	-2.25	Gizi Baik	-1.46	O	0.84		-	-
3	M AL	L	2.8	49	Kalti	Samarin	Sungai Pin	Remaja	Temindu	Pulau Ind			0 Tahun - 4 Bulan - 7 Hari	2023-01-02	8.1	69	0	Norma	-0.53	Norma	-0.53	Gizi Baik	-0.14	O	-		-	-
:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
15072	Afizah	P	2.5	45	Kalti	Samarin	Samarinda	Mangkup	Tenun Sa	Balo Neg	1		0 Tahun - 0 Bulan - 0 Hari	2023-12-09	2.5	45	0	Kuran	-2.03	Pende	-2.63	Gizi Baik	-0.35	-	-		-	-
15073	M Arsya	P	3	49	Kalti	Samarin	Samarinda	Juanda	Air Hitam	Mekar Se	3		0 Tahun - 0 Bulan - 0 Hari	2023-12-19	3	49	0	Norma	-1.56	Norma	-1.3	Gizi Baik	-1.04	-	-		-	-
15074	Muhamm	L	2.9	49	Kalti	Samarin	Samarinda	Sidomulyo	Sungai D	Ramania			0 Tahun - 0 Bulan - 0 Hari	2023-12-29	2.9	49	0	Kuran	-2.97	Pende	-2.48	Gizi Baik	-1.37	-	-		-	-

Lamprian 3 Laporan Bimbingan

KARTU KENDALI BIMBINGAN LAPORAN KARYA ILMIAH

Nama Mahasiswa	:	Bima Satria
NIM	:	2011102441102
Nama Dosen Pembimbing	:	Taghfirul Azhima Yoga Siswa, S.Kom, M.Kom
Judul Penelitian	:	Optimasi Random Forest Dengan GA Dan RFE Pada High Dimensional Data Stunting

No	Tanggal	Uraian Pembimbingan	Paraf Dosen
1	5 Februari 2024	Pertemuan pertama, membahas tahapan-tahapan penelitian skripsi.	
2	7 Februari 2024	Pertemuan kedua, Membahas cara mencari artikel/paper rujukan penelitian dengan baik dan benar.	
3	16 Februari 2024	Pertemuan ketiga, Review Survey Paper untuk mencari gap permasalahan klasifikasi pada data mining, machine learning dan mencari topik penelitian yang masih relevan untuk diteliti peneliti.	
4	17 Februari 2024	Pertemuan keempat, Review Technical Paper dan Road Maps penelitian, untuk mencari algoritma dan metode yang di inginkan biar bisa mengatasi permasalahan kalsifikasi.	
5	22 Februari 2024	Pertemuan kelima, Mencari Paper/Artikel sesuai dengan rujukkan terkait objek penelitian dan pembahasan study literatur dan riset problem.	
6	6 Maret 2024	Pertemuan keenam, penentuan judul penelitian.	
7	13 Maret 2024	Pertemuan Ketujuh, Pembuatan Canvas penelitian untuk di submit ke prodi.	
8	5 April 2024	Pertemuan Kedelapan, Pengajuan Surat permohonan data untuk penelitian ke Dinas Kesehatan Samarinda.	
9	18 April 2024	Pertemuan Kesembilan, Revisi Proposal Bab 1, Bab 2 dan perbaikan format penulisan skripsi.	

10	24 April 2024	Pertemuan Kesepuluh, Revisi Proposal Bab 1, Bab 2 dan perbaikan format penulisan skripsi sebelum di submit ke simple.	
11	10 Mai 2024	Pertemuan kesebelas, bimbingan pembahasan penulisan Bab 3 dan Bab 4 beserta publikasi	
12	17 Mai 2024	Pertemuan kedua belas, Revisi naskah skripsi bab 3 sampai dengan bab 4	
13	22 Juni 2024	Pertemuan ketigabelas, konsultasi revisi publikasi jurnal bagian abstrak, pendahuluan, metodologi penelitian, hasil & pembahasan dan kesimpulan	

mengetahui

Dosen Pembimbing

Ketua Program Studi

Taghfirul Azhima Yoga Siswa, S.Kom, M.Kom
NIDN.1118038805

Arbansyah, S.Kom, M. TI
NIDN.1118019203

Lampiran 4 Kode Kode Klasifikasi Algortima *Random Forest + RFE + GA*

```
import pandas as pd

# Membaca data dari file CSV (ganti dengan lokasi file yang sesuai)
data = pd.read_csv('dataset_stunting_samarinda.csv', encoding='latin1', low_memory=False)

# Menghitung jumlah data sebelum penghapusan duplikat
total_data_sebelum = len(data)

# Mengurutkan data berdasarkan kolom "Nama" dan "Tanggal Pengukuran" secara menurun
data = data.sort_values(by=['Nama', 'Tanggal Pengukuran'], ascending=[True, False])

# Menghapus duplikat berdasarkan kolom "Nama" dan mempertahankan yang pertama (yang memiliki tanggal pengukuran terbaru)
data = data.drop_duplicates(subset='Nama', keep='first')

# Menghitung jumlah data setelah penghapusan duplikat
total_data_sesudah = len(data)

# Menyimpan data yang telah dihapus duplikatnya ke file CSV baru (jika diperlukan)
data.to_csv('dataset_new.csv', index=False)

print(f"Total data sebelum penghapusan duplikat: {total_data_sebelum}")
print(f"Total data setelah penghapusan duplikat: {total_data_sesudah}")
```

```
Total data sebelum penghapusan duplikat: 150474
Total data setelah penghapusan duplikat: 34201
```

```
dataset = pd.read_csv('dataset_new.csv')
dataset.info()
dataset = data.drop('3ml Vit A', axis=1)

# Menyimpan hasilnya ke file CSV baru (jika diperlukan)
dataset.to_csv('dataset_tanpa_duplikat.csv', index=False)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34201 entries, 0 to 34200
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Nama                34200 non-null  object
 1   JK                  34201 non-null  object
 2   Berat               34201 non-null  object
 3   Tinggi              33633 non-null  object
 4   LiLA                28598 non-null  float64
 5   BB/U                34201 non-null  object
 6   ZS BB/U             34201 non-null  object
 7   TB/U                31561 non-null  object
 8   ZS TB/U             34201 non-null  object
 9   BB/TB               31574 non-null  object
 10  ZS BB/TB            34201 non-null  object
 11  Naik Berat Badan    34201 non-null  object
 12  3ml Vit A           12663 non-null  float64
 13  Tanggal Pengukuran  34201 non-null  object
dtypes: float64(2), object(12)
memory usage: 3.7+ MB
```

```
dataset = pd.read_csv('dataset_tanpa_duplikat.csv')
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34201 entries, 0 to 34200
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Nama                34200 non-null  object
 1   JK                  34201 non-null  object
 2   Berat               34201 non-null  object
 3   Tinggi              33633 non-null  object
 4   LiLA                28598 non-null  float64
 5   BB/U                34201 non-null  object
 6   ZS BB/U             34201 non-null  object
 7   TB/U                31561 non-null  object
 8   ZS TB/U             34201 non-null  object
 9   BB/TB               31574 non-null  object
 10  ZS BB/TB            34201 non-null  object
 11  Naik Berat Badan    34201 non-null  object
 12  Tanggal Pengukuran  34201 non-null  object
dtypes: float64(1), object(12)
memory usage: 3.4+ MB
```

```
dataset['TB/U'].value_counts()
```

```
TB/U
Normal          26943
Pendek           3350
Sangat Pendek    1125
Tinggi            143
Name: count, dtype: int64
```

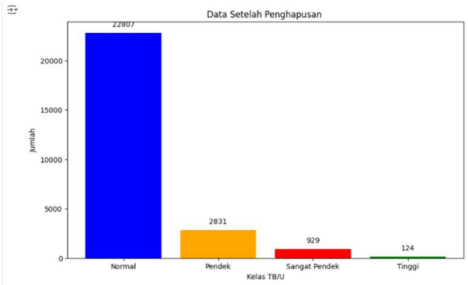
```
import pandas as pd

# Membaca data dari file CSV atau sumber data lainnya
data = pd.read_csv('dataset_tanpa_duplikat.csv')

# Menghitung jumlah data sebelum penghapusan
jumlah_data_sebelum = len(data)

# Menghapus baris yang memiliki setidaknya satu nilai yang hilang
data_tanpa_isi = data.dropna()

# Menghitung jumlah data setelah penghapusan
jumlah_data_sesudah = len(data_tanpa_isi)

# Menyimpan hasilnya ke file CSV baru (jika diperlukan)
data_tanpa_isi.to_csv('dataset_tanpa_nilai.csv', index=False)

# Mencetak jumlah data sebelum dan setelah penghapusan
print(f"Jumlah data sebelum penghapusan: {jumlah_data_sebelum}")
print(f"Jumlah data setelah penghapusan: {jumlah_data_sesudah}")
```

```
Jumlah data sebelum penghapusan: 34201
Jumlah data setelah penghapusan: 26691
```

```
data_tanpa_isi['TB/U'].value_counts()
```

```
TB/U
Normal          22807
Pendek           2831
Sangat Pendek     929
Tinggi            124
Name: count, dtype: int64
```

```
import pandas as pd
import matplotlib.pyplot as plt

# Contoh data untuk TB/U
data_tanpa_isi = pd.DataFrame({
    'TB/U': ['Normal'] * 22807 + ['Pendek'] * 2831 + ['Sangat Pendek'] * 929 + ['Tinggi'] * 124
})

# Hitung value counts
value_counts = data_tanpa_isi['TB/U'].value_counts()

# Buat DataFrame dari value counts
df_value_counts = value_counts.reset_index()
df_value_counts.columns = ['TB/U', 'count']

# Plot bar chart
plt.figure(figsize=(10, 6))
bars = plt.bar(df_value_counts['TB/U'], df_value_counts['count'], color=['blue', 'orange', 'red', 'green'])

# Tambahkan label di atas setiap bar
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 500, int(yval), ha='center', va='bottom')

# Menambahkan judul dan label sumbu
plt.title('Data Setelah Penghapusan ')
plt.xlabel('Kelas TB/U')
plt.ylabel('Jumlah')

# Tampilkan plot
plt.show()
```




```
import pandas as pd

# Membaca dataset
data_tanpa_isi = pd.read_csv('dataset_tanpa_nilai.csv')

# Membuat kamus penggantian nilai
penggantian = {
    'Normal': 'Tidak Stunting',
    'Tinggi': 'Tidak Stunting',
    'Pendek': 'Stunting',
    'Sangat Pendek': 'Stunting'
}

# Mengganti nilai dalam kolom "TB/U" sesuai dengan kamus penggantian
if 'TB/U' in data_tanpa_isi.columns:
    data_tanpa_isi['TB/U'] = data_tanpa_isi['TB/U'].replace(penggantian)

# Menyimpan hasilnya ke file CSV baru
data_tanpa_isi.to_csv('dataset_stunting_samarinda_modif.csv', index=False)

# Menampilkan DataFrame hasil
print(data_tanpa_isi.head())
```

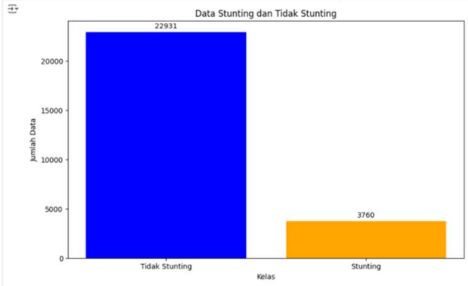
```
                Nama  JK Berat  Tinggi  LiLA              BB/U  ZS BB/U  \
0          A ALVIN    L   18.6   104.5  18.0      Risiko Lebih     1.19
1          A FADLAN   L   11.6    83.0   0.0  Berat Badan Normal    -0.08
2  A FARIS WICAKSONO  L    9.7    78.0  16.0  Berat Badan Normal    -1.75
3          A FATHAN   L     15   107.0  17.0  Berat Badan Normal    -1.08
4          A FAUJAN   L     14   100.0   0.0  Berat Badan Normal    -0.85

          TB/U  ZS TB/U                BB/TB  ZS BB/TB Naik Berat Badan  \
0  Tidak Stunting     0.41  Risiko Gizi Lebih     1.41                 N
1  Tidak Stunting    -0.97          Gizi Baik     0.59                 T
2        Stunting    -2.84          Gizi Baik    -0.47                 O
3  Tidak Stunting     0.14          Gizi Baik    -1.83                 T
4  Tidak Stunting    -0.16          Gizi Baik    -1.14                 O

  Tanggal Pengukuran
0         2023-12-16
1         2023-07-10
2         2023-10-06
3         2023-10-23
4         2023-02-07
```

```
import pandas as pd
import numpy as np
stunting = pd.read_csv('dataset_stunting_samarinda_modif.csv')
stunting = stunting.rename(columns={'TB/U': 'Kelas'})
stunting = stunting.drop('Nama', axis=1)
stunting = stunting.drop('Tanggal Pengukuran', axis=1)
stunting
# Menyimpan hasilnya ke file CSV baru
stunting.to_csv('dataset_stunting_samarinda_terbaru.csv', index=False)
# class_data = stunting['TB/U']
```

```
stunting['Kelas'].value_counts()
```

```
Kelas
Tidak Stunting    22931
Stunting           3760
Name: count, dtype: int64
```

```
import pandas as pd
import matplotlib.pyplot as plt
import os

# Contoh data untuk Kelas
data = {
    'Kelas': ['Tidak Stunting'] * 22931 + ['Stunting'] * 3760
}
stunting = pd.DataFrame(data)

# Hitung value counts untuk kolom 'Kelas'
value_counts = stunting['Kelas'].value_counts()

# Buat DataFrame dari value counts
df_value_counts = value_counts.reset_index()
df_value_counts.columns = ['Kelas', 'count']

# Plot bar chart
plt.figure(figsize=(10, 6))
bars = plt.bar(df_value_counts['Kelas'], df_value_counts['count'], color=['blue', 'orange'])

# Tambahkan label di atas setiap bar
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 200, int(yval), ha='center', va='bottom')

# Menambahkan judul dan label sumbu
plt.title('Data Stunting dan Tidak Stunting')
plt.xlabel('Kelas')
plt.ylabel('Jumlah Data')
```



```
[ ] import pandas as pd
    from sklearn.preprocessing import LabelEncoder, OrdinalEncoder

    # Membaca dataset
    stunting = pd.read_csv('dataset_stunting_samarinda_terbaru.csv')

    # Menampilkan kolom yang ada dalam DataFrame
    print("Kolom dalam DataFrame:", stunting.columns)

    # Membuat instance untuk encoder
    ordinal = OrdinalEncoder()
    labelencoder = LabelEncoder()

    # Daftar kolom yang dibutuhkan
    columns_needed = ['JK', 'Berat', 'Tinggi', 'LiLA', 'BB/U', 'ZS BB/U', 'ZS TB/U', 'Naik Berat Badan', 'BB/TB', 'ZS BB/TB', 'Kelas']

    # Menyimpan subset data yang diperlukan
    df_transform = stunting[columns_needed]

    # Transformasi data kategorikal menjadi numerik
    stunting['JK'] = labelencoder.fit_transform(stunting['JK'])
    stunting['BB/U'] = labelencoder.fit_transform(stunting['BB/U'])
    stunting['BB/TB'] = labelencoder.fit_transform(stunting['BB/TB'])
    stunting['Naik Berat Badan'] = labelencoder.fit_transform(stunting['Naik Berat Badan'])

    # Membuat DataFrame hasil transformasi
    df = stunting[columns_needed]

    # Menampilkan DataFrame hasil transformasi
    print(df.head())
```

```
Kolom dalam DataFrame: Index(['JK', 'Berat', 'Tinggi', 'LiLA', 'BB/U', 'ZS BB/U', 'Kelas', 'ZS TB/U',
       'BB/TB', 'ZS BB/TB', 'Naik Berat Badan'],
      dtype='object')
   JK  Berat  Tinggi  LiLA  BB/U  ZS BB/U  ZS TB/U  Naik Berat Badan  BB/TB  \
0   0   18.6   104.5  18.0     2     1.19     0.41                 1      5
1   0   11.6    83.0   0.0     0    -0.08    -0.97                 3      0
2   0    9.7    78.0  16.0     0    -1.75    -2.84                 2      0
3   0     15   107.0  17.0     0    -1.08     0.14                 3      0
4   0     14   100.0   0.0     0    -0.85    -0.16                 2      0

   ZS BB/TB          Kelas
0      1.41  Tidak Stunting
1      0.59  Tidak Stunting
2     -0.47       Stunting
3     -1.83  Tidak Stunting
4     -1.14  Tidak Stunting
```

```
[ ] df_transform
```

	JK	Berat	Tinggi	LiLA	BB/U	ZS BB/U	ZS TB/U	Naik Berat Badan	BB/TB	ZS BB/TB	Kelas
0	L	18.6	104.5	18.0	Risiko Lebih	1.19	0.41	N	Risiko Gizi Lebih	1.41	Tidak Stunting
1	L	11.6	83.0	0.0	Berat Badan Normal	-0.08	-0.97	T	Gizi Baik	0.59	Tidak Stunting
2	L	9.7	78.0	16.0	Berat Badan Normal	-1.75	-2.84	O	Gizi Baik	-0.47	Stunting
3	L	15	107.0	17.0	Berat Badan Normal	-1.08	0.14	T	Gizi Baik	-1.83	Tidak Stunting
4	L	14	100.0	0.0	Berat Badan Normal	-0.85	-0.16	O	Gizi Baik	-1.14	Tidak Stunting
...	...	...	...	...	...	...	...	...	...	...	...
26686	L	14.5	102.0	0.0	Berat Badan Normal	-1.71	-1.62	T	Gizi Baik	-1.15	Tidak Stunting
26687	L	29.7	116.0	0.0	Risiko Lebih	4.16	2.18	N	Obesitas	3.74	Tidak Stunting
26688	P	15.7	94.5	0.0	Risiko Lebih	1.78	1.38	O	Risiko Gizi Lebih	1.43	Tidak Stunting
26689	L	17.3	107.0	0.0	Berat Badan Normal	-0.14	-0.13	O	Gizi Baik	-0.13	Tidak Stunting
26690	L	15.6	102.0	0.0	Berat Badan Normal	-0.59	-0.69	N	Gizi Baik	-0.25	Tidak Stunting

26691 rows × 11 columns

```
[ ] df
```

	JK	Berat	Tinggi	LiLA	BB/U	ZS BB/U	ZS TB/U	Naik Berat Badan	BB/TB	ZS BB/TB	Kelas
0	0	18.6	104.5	18.0	2	1.19	0.41	1	5	1.41	Tidak Stunting
1	0	11.6	83.0	0.0	0	-0.08	-0.97	3	0	0.59	Tidak Stunting
2	0	9.7	78.0	16.0	0	-1.75	-2.84	2	0	-0.47	Stunting
3	0	15	107.0	17.0	0	-1.08	0.14	3	0	-1.83	Tidak Stunting
4	0	14	100.0	0.0	0	-0.85	-0.16	2	0	-1.14	Tidak Stunting
...	...	...	...	...	...	...	...	...	...	...	...
26686	0	14.5	102.0	0.0	0	-1.71	-1.62	3	0	-1.15	Tidak Stunting
26687	0	29.7	116.0	0.0	2	4.16	2.18	1	4	3.74	Tidak Stunting
26688	1	15.7	94.5	0.0	2	1.78	1.38	2	5	1.43	Tidak Stunting
26689	0	17.3	107.0	0.0	0	-0.14	-0.13	2	0	-0.13	Tidak Stunting
26690	0	15.6	102.0	0.0	0	-0.59	-0.69	1	0	-0.25	Tidak Stunting

26691 rows × 11 columns

RANDOM FOREST

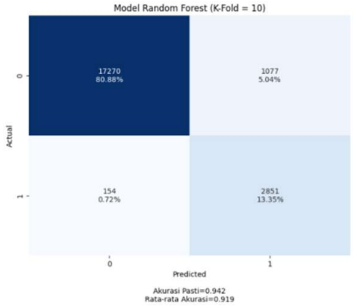
```
# Mengimpor pustaka yang diperlukan
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_predict, KFold, train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import seaborn as sns

# Fungsi untuk membuat plot matriks kebingungan
def buat_matriks_kebingungan(cf, nama_grup=None, kategori='auto', jumlah=True, persen=True, barwarna=True, xyticks=True, xyplotlabels=True, stat_ringkasan=True, ukuran_gambar=None, cmap='Blues', judul=None, akurasi_pasti=None, rata2_akurasi=None):
    kosong = ['' for i in range(cf.size)]
    if nama_grup and len(nama_grup) == cf.size:
        label_grup = ["{}\n".format(value) for value in nama_grup]
    else:
        label_grup = kosong
    if jumlah:
        jumlah_grup = ["{0:0.0f}\n".format(value) for value in cf.flatten()]
    else:
        jumlah_grup = kosong
    if persen:
        persentase_grup = ["{0:.2%}".format(value) for value in cf.flatten() / np.sum(cf)]
    else:
        persentase_grup = kosong
    label_kotak = [f"{v1}{v2}{v3}".strip() for v1, v2, v3 in zip(label_grup, jumlah_grup, persentase_grup)]
    label_kotak = np.asarray(label_kotak).reshape(cf.shape[0], cf.shape[1])
    teks_stat = ""
    if stat_ringkasan:
        if akurasi_pasti is not None:
            teks_stat += "\n\nAkurasi Pasti={:0.3f}".format(akurasi_pasti)
        if rata2_akurasi is not None:
            teks_stat += "\nRata-rata Akurasi={:0.3f}".format(rata2_akurasi)
    if ukuran_gambar is None:
        ukuran_gambar = plt.rcParams.get('figure.figsize')
    if not xyticks:
        kategori = False
    plt.figure(figsize=ukuran_gambar)
    sns.heatmap(cf, annot=label_kotak, fmt="", cmap=cmap, cbar=barwarna, xticklabels=kategori, yticklabels=kategori)
    if xyplotlabels:
        plt.ylabel('Actual')
        plt.xlabel('Predicted' + teks_stat)
```

```
    else:
        plt.xlabel(teks_stat)
    if judul:
        plt.title(judul)
    plt.show()

# Membaca data yang telah diproses
data = pd.read_csv('dataset_stunting_samarinda_modif.csv')

# Mengganti nama kolom 'TB/U' menjadi 'Kelas'
data = data.rename(columns={'TB/U': 'Kelas'})

# Menghapus kolom non-numerik dan tidak relevan
data = data.drop(['Nama', 'Tanggal Pengukuran'], axis=1)

# Mengganti nilai bermasalah di kolom 'JK'
data['JK'] = data['JK'].str.strip().replace({'L\xa0': 'L', 'P\xa0': 'P'})

# Menangani nilai numerik bermasalah
data['Berat'] = data['Berat'].str.replace(',', '.').astype(float)

# Mengonversi kolom kategorikal menjadi numerik menggunakan one-hot encoding
kolom_kategorikal = ['JK', 'BB/U', 'BB/TB', 'Naik Berat Badan']
data = pd.get_dummies(data, columns=kolom_kategorikal, drop_first=True)

# Memastikan tidak ada kehilangan data setelah konversi
print(f"Bentuk dataset setelah konversi: {data.shape}")

# Memisahkan fitur dan variabel target
X = data.drop('Kelas', axis=1)
y = data['Kelas'].map({'Tidak Stunting': 0, 'Stunting': 1})

# Menormalisasi fitur
scaler = MinMaxScaler()
X_normalized = scaler.fit_transform(X)

# Membagi data menjadi training dan testing set
x_train, x_test, y_train, y_test = train_test_split(X_normalized, y, test_size=0.2, random_state=42)

# Merekam hasil K-Fold 10 kali
kfolds = 10
repeats = 10
akurasi_list = []

# Menggunakan KFold untuk cross-validation
kf = KFold(n_splits=kfolds, shuffle=True, random_state=1)
fold = 1
```

```
# Melakukan cross-validation dengan variasi
for i in range(repeats):
    print(f"Rekaman {i+1}")

    # Mendefinisikan classifier Random Forest dengan parameter yang sedikit berbeda untuk setiap iterasi
    rf_classifier = RandomForestClassifier(
        n_estimators=10 + i,  # Mengubah jumlah estimators
        criterion='gini',
        max_depth=1 + i % 3,  # Mengubah kedalaman maksimal pohon
        min_samples_split=100 + i,  # Mengubah jumlah minimal sampel untuk split
        min_samples_leaf=50 - i * 5,  # Mengubah jumlah minimal sampel per daun
        max_features=0.1 + (i % 5) * 0.02,  # Mengubah jumlah fitur yang dipertimbangkan untuk split
        class_weight='balanced',
        random_state=42 + i  # Mengubah random state
    )

    y_pred = cross_val_predict(rf_classifier, x_train, y_train, cv=kfolds)
    cf_matrix = confusion_matrix(y_train, y_pred)
    akurasi = np.trace(cf_matrix) / float(np.sum(cf_matrix))
    akurasi_list.append(akurasi)
    print(f"Akurasi: {akurasi:.3f}")
    class_report = classification_report(y_train, y_pred, target_names=['Tidak Stunting', 'Stunting'])
    print(class_report)

# Menghitung akurasi rata-rata
rata2_akurasi = np.mean(akurasi_list)
akurasi_terakhir = akurasi_list[-1]
print(f"Rata-rata Akurasi: {rata2_akurasi:.3f}")
print(f"Akurasi Pasti (Iterasi Terakhir): {akurasi_terakhir:.3f}")

# Menggambar plot matriks kebingungan untuk iterasi terakhir dengan rata-rata akurasi dan akurasi pasti
buat_matriks_kebingungan(cf_matrix, ukuran_gambar=(8, 6), berwarna=False, judul="Model Random Forest (K-Fold = 10)", akurasi_pasti=akurasi_terakhir, rata2_akurasi=rata2_akurasi)
```

```
Bentuk dataset setelah konversi: (26691, 19)
Rekaman 1
Akurasi: 0.709
              precision    recall  f1-score   support

Tidak Stunting       0.94      0.70      0.81     18347
      Stunting       0.29      0.74      0.42      3005

      accuracy                           0.71     21352
     macro avg       0.62      0.72      0.61     21352
  weighted avg       0.85      0.71      0.75     21352

Rekaman 2
Akurasi: 0.860
              precision    recall  f1-score   support

Tidak Stunting       0.93      0.90      0.92     18347
      Stunting       0.50      0.60      0.55      3005

      accuracy                           0.86     21352
     macro avg       0.72      0.75      0.73     21352
  weighted avg       0.87      0.86      0.87     21352

Rekaman 3
Akurasi: 0.996
              precision    recall  f1-score   support

Tidak Stunting       1.00      1.00      1.00     18347
      Stunting       0.98      1.00      0.99      3005

      accuracy                           1.00     21352
     macro avg       0.99      1.00      0.99     21352
  weighted avg       1.00      1.00      1.00     21352

Rekaman 4
Akurasi: 0.894
              precision    recall  f1-score   support

Tidak Stunting       0.97      0.90      0.94     18347
      Stunting       0.59      0.85      0.69      3005

      accuracy                           0.89     21352
     macro avg       0.78      0.88      0.82     21352
  weighted avg       0.92      0.89      0.90     21352

Rekaman 5
Akurasi: 0.968
              precision    recall  f1-score   support

Tidak Stunting       1.00      0.96      0.98     18347
      Stunting       0.82      0.99      0.90      3005

      accuracy                           0.97     21352
     macro avg       0.91      0.98      0.94     21352
  weighted avg       0.97      0.97      0.97     21352

Rekaman 6
Akurasi: 0.916
              precision    recall  f1-score   support

Tidak Stunting       0.98      0.93      0.95     18347
      Stunting       0.65      0.86      0.74      3005

      accuracy                           0.92     21352
     macro avg       0.82      0.89      0.85     21352
  weighted avg       0.93      0.92      0.92     21352

Rekaman 7
Akurasi: 0.942
              precision    recall  f1-score   support

Tidak Stunting       1.00      0.93      0.96     18347
      Stunting       0.71      0.99      0.83      3005

      accuracy                           0.94     21352
     macro avg       0.85      0.96      0.90     21352
  weighted avg       0.96      0.94      0.95     21352
```

```
Rekaman 8
Akurasi: 0.966
              precision    recall  f1-score   support

Tidak Stunting       0.99      0.97      0.98     18347
      Stunting       0.84      0.93      0.88      3005

      accuracy                           0.97     21352
     macro avg       0.91      0.95      0.93     21352
  weighted avg       0.97      0.97      0.97     21352

Rekaman 9
Akurasi: 0.992
              precision    recall  f1-score   support

Tidak Stunting       1.00      0.99      1.00     18347
      Stunting       0.95      0.99      0.97      3005

      accuracy                           0.99     21352
     macro avg       0.98      0.99      0.98     21352
  weighted avg       0.99      0.99      0.99     21352

Rekaman 10
Akurasi: 0.942
              precision    recall  f1-score   support

Tidak Stunting       0.99      0.94      0.97     18347
      Stunting       0.73      0.95      0.82      3005

      accuracy                           0.94     21352
     macro avg       0.86      0.95      0.89     21352
  weighted avg       0.95      0.94      0.95     21352

Rata-rata Akurasi: 0.919
Akurasi Pasti (Iterasi Terakhir): 0.942
```



RANDOM FOREST + RFE

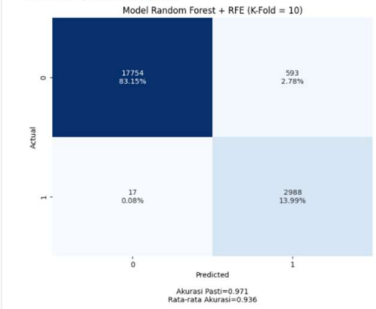
```
# Mengimpor pustaka yang diperlukan
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_predict, KFold, train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import RFE
import matplotlib.pyplot as plt
import seaborn as sns

# Fungsi untuk membuat plot matriks kebingungan
def buat_matriks_kebingungan(cf, nama_grup=None, kategori='auto', jumlah=True, persen=True, barwarna=True, xyticks=True, xyplotlabels=True, stat_ringkasan=True, ukuran_gambar=None, cmap='Blues', judul=None, akurasi_pasti=None, rata2_akurasi=None):
    kosong = ['' for i in range(cf.size)]
    if nama_grup and len(nama_grup) == cf.size:
        label_grup = ["{}\n".format(value) for value in nama_grup]
    else:
        label_grup = kosong
    if jumlah:
        jumlah_grup = ["{0:0.0f}\n".format(value) for value in cf.flatten()]
    else:
        jumlah_grup = kosong
    if persen:
        persentase_grup = ["{0:.2%}".format(value) for value in cf.flatten() / np.sum(cf)]
    else:
        persentase_grup = kosong
    label_kotak = [f"{v1}{v2}{v3}".strip() for v1, v2, v3 in zip(label_grup, jumlah_grup, persentase_grup)]
    label_kotak = np.asarray(label_kotak).reshape(cf.shape[0], cf.shape[1])
    teks_stat = ""
    if stat_ringkasan:
        if akurasi_pasti is not None:
            teks_stat += "\n\nAkurasi Pasti={:0.3f}".format(akurasi_pasti)
        if rata2_akurasi is not None:
            teks_stat += "\nRata-rata Akurasi={:0.3f}".format(rata2_akurasi)
    if ukuran_gambar is None:
        ukuran_gambar = plt.rcParams.get('figure.figsize')
    if not xyticks:
        kategori = False
    plt.figure(figsize=ukuran_gambar)
    sns.heatmap(cf, annot=label_kotak, fmt="", cmap=cmap, cbar=barwarna, xticklabels=kategori, yticklabels=kategori)
    if xyplotlabels:
        plt.ylabel('Actual')
        plt.xlabel('Predicted' + teks_stat)
    else:
        plt.xlabel(teks_stat)
    if judul:
        plt.title(judul)
    plt.show()

# Membaca data yang telah diproses
data = pd.read_csv('dataset_stunting_samarinda_modif.csv')

# Mengganti nama kolom 'TB/U' menjadi 'Kelas'
data = data.rename(columns={'TB/U': 'Kelas'})

# Menghapus kolom non-numerik dan tidak relevan
data = data.drop(['Nama', 'Tanggal Pengukuran'], axis=1)

# Mengganti nilai bermasalah di kolom 'JK'
data['JK'] = data['JK'].str.strip().replace({'L\x00': 'L', 'P\x00': 'P'})

# Menangani nilai numerik bermasalah
data['Berat'] = data['Berat'].str.replace(',', '.').astype(float)

# Mengonversi kolom kategorikal menjadi numerik menggunakan one-hot encoding
kolom_kategorikal = ['JK', 'BB/U', 'BB/TB', 'Naik Berat Badan']
data = pd.get_dummies(data, columns=kolom_kategorikal, drop_first=True)

# Memastikan tidak ada kehilangan data setelah konversi
print(f"Bentuk dataset setelah konversi: {data.shape}")

# Memisahkan fitur dan variabel target
X = data.drop('Kelas', axis=1)
y = data['Kelas'].map({'Tidak Stunting': 0, 'Stunting': 1})

# Menormalisasi fitur
scaler = MinMaxScaler()
X_normalized = scaler.fit_transform(X)

# Membagi data menjadi training dan testing set
x_train, x_test, y_train, y_test = train_test_split(X_normalized, y, test_size=0.2, random_state=42)

# Seleksi fitur menggunakan RFE
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rfe = RFE(estimator=rf, n_features_to_select=10)
x_train_rfe = rfe.fit_transform(x_train, y_train)
x_test_rfe = rfe.transform(x_test)

# Merekam hasil K-Fold 10 kali
kfolds = 10
repeats = 10
akurasi_list = []

# Menggunakan KFold untuk cross-validation
kf = KFold(n_splits=kfolds, shuffle=True, random_state=1)
fold = 1
```

```python
# Melakukan cross-validation dengan variasi
for i in range(repeats):
    print(f"Rekaman {i+1}")

    # Mendefinisikan classifier Random Forest dengan parameter yang sedikit berbeda untuk setiap iterasi
    rf_classifier = RandomForestClassifier(
        n_estimators=10 + i,  # Mengubah jumlah estimators
        criterion='gini',
        max_depth=1 + i % 3,  # Mengubah kedalaman maksimal pohon
        min_samples_split=100 + i,  # Mengubah jumlah minimal sampel untuk split
        min_samples_leaf=50 - i * 5,  # Mengubah jumlah minimal sampel per daun
        max_features=0.1 + (i % 5) * 0.02,  # Mengubah jumlah fitur yang dipertimbangkan untuk split
        class_weight='balanced',
        random_state=42 + i  # Mengubah random state
    )

    y_pred = cross_val_predict(rf_classifier, x_train_rfe, y_train, cv=kfolds)
    cf_matrix = confusion_matrix(y_train, y_pred)
    akurasi = np.trace(cf_matrix) / float(np.sum(cf_matrix))
    akurasi_list.append(akurasi)
    print(f"Akurasi: {akurasi:.3f}")
    class_report = classification_report(y_train, y_pred, target_names=['Tidak Stunting', 'Stunting'])
    print(class_report)

# Menghitung akurasi rata-rata
rata2_akurasi = np.mean(akurasi_list)
akurasi_terakhir = akurasi_list[-1]
print(f"Rata-rata Akurasi: {rata2_akurasi:.3f}")
print(f"Akurasi Pasti (Iterasi Terakhir): {akurasi_terakhir:.3f}")

# Menggambar plot matriks kebingungan untuk iterasi terakhir dengan rata-rata akurasi dan akurasi pasti
buat_matriks_kebingungan(cf_matrix, ukuran_gambar=(8, 6), berwarna=False, judul="Model Random Forest + RFE (K-Fold = 10)", akurasi_pasti=akurasi_terakhir, rata2_akurasi=rata2_akurasi)
```

```
Bentuk dataset setelah konversi: (26691, 19)
Rekaman 1
Akurasi: 0.817
                precision    recall  f1-score   support

Tidak Stunting       0.94      0.84      0.89     18347
      Stunting       0.41      0.66      0.50      3005

      accuracy                           0.82     21352
     macro avg       0.67      0.75      0.69     21352
  weighted avg       0.86      0.82      0.83     21352

Rekaman 2
Akurasi: 0.849
                precision    recall  f1-score   support

Tidak Stunting       0.97      0.85      0.91     18347
      Stunting       0.48      0.83      0.61      3005

      accuracy                           0.85     21352
     macro avg       0.72      0.84      0.76     21352
  weighted avg       0.90      0.85      0.86     21352

Rekaman 3
Akurasi: 0.995
                precision    recall  f1-score   support

Tidak Stunting       1.00      0.99      1.00     18347
      Stunting       0.97      1.00      0.98      3005

      accuracy                           0.99     21352
     macro avg       0.98      1.00      0.99     21352
  weighted avg       0.99      0.99      0.99     21352

Rekaman 4
Akurasi: 0.864
                precision    recall  f1-score   support

Tidak Stunting       0.92      0.92      0.92     18347
      Stunting       0.51      0.54      0.53      3005

      accuracy                           0.86     21352
     macro avg       0.72      0.73      0.72     21352
  weighted avg       0.87      0.86      0.86     21352

Rekaman 5
Akurasi: 0.985
                precision    recall  f1-score   support

Tidak Stunting       0.99      0.99      0.99     18347
      Stunting       0.93      0.97      0.95      3005

      accuracy                           0.98     21352
     macro avg       0.96      0.98      0.97     21352
  weighted avg       0.99      0.98      0.98     21352
```

```
Rekaman 6
Akurasi: 0.935
                precision    recall  f1-score   support

Tidak Stunting       0.98      0.94      0.96     18347
      Stunting       0.71      0.91      0.80      3005

      accuracy                           0.93     21352
     macro avg       0.85      0.92      0.88     21352
  weighted avg       0.95      0.93      0.94     21352

Rekaman 7
Akurasi: 1.000
                precision    recall  f1-score   support

Tidak Stunting       1.00      1.00      1.00     18347
      Stunting       1.00      1.00      1.00      3005

      accuracy                           1.00     21352
     macro avg       1.00      1.00      1.00     21352
  weighted avg       1.00      1.00      1.00     21352

Rekaman 8
Akurasi: 0.950
                precision    recall  f1-score   support

Tidak Stunting       1.00      0.94      0.97     18347
      Stunting       0.74      0.99      0.85      3005

      accuracy                           0.95     21352
     macro avg       0.87      0.97      0.91     21352
  weighted avg       0.96      0.95      0.95     21352

Rekaman 9
Akurasi: 0.996
                precision    recall  f1-score   support

Tidak Stunting       1.00      1.00      1.00     18347
      Stunting       0.98      0.99      0.98      3005

      accuracy                           1.00     21352
     macro avg       0.99      0.99      0.99     21352
  weighted avg       1.00      1.00      1.00     21352

Rekaman 10
Akurasi: 0.971
                precision    recall  f1-score   support

Tidak Stunting       1.00      0.97      0.98     18347
      Stunting       0.83      0.99      0.91      3005

      accuracy                           0.97     21352
     macro avg       0.92      0.98      0.95     21352
  weighted avg       0.98      0.97      0.97     21352
```

```
Rata-rata Akurasi: 0.936
Akurasi Pasti (Iterasi Terakhir): 0.971
```




```
[ ] # Instal DEAP
    !pip install deap

Collecting deap
  Downloading deap-1.4.1-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (135 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 135.4/135.4 kB 2.7 MB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from deap) (1.25.2)
Installing collected packages: deap
Successfully installed deap-1.4.1
```

RF+RFE+GA

```python
# Mengimpor pustaka yang diperlukan
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_predict, KFold, train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import RFE
import matplotlib.pyplot as plt
import seaborn as sns
from deap import base, creator, tools, algorithms

# Fungsi untuk membuat plot matriks kebingungan
def buat_matriks_kebingungan(cf, nama_grup=None, kategori='auto', jumlah=True, persen=True, barwarna=True, xyticks=True, xyplotlabels=True, stat_ringkasan=True, ukuran_gambar=None, cmap='Blues', judul=None, akurasi_pasti=None, rata2_akurasi=None):
    kosong = ['' for i in range(cf.size)]
    if nama_grup and len(nama_grup) == cf.size:
        label_grup = ["{}\n".format(value) for value in nama_grup]
    else:
        label_grup = kosong
    if jumlah:
        jumlah_grup = ["{0:0.0f}\n".format(value) for value in cf.flatten()]
    else:
        jumlah_grup = kosong
    if persen:
        persentase_grup = ["{0:.2%}".format(value) for value in cf.flatten() / np.sum(cf)]
    else:
        persentase_grup = kosong
    label_kotak = [f"{v1}{v2}{v3}".strip() for v1, v2, v3 in zip(label_grup, jumlah_grup, persentase_grup)]
    label_kotak = np.asarray(label_kotak).reshape(cf.shape[0], cf.shape[1])
    teks_stat = ""
    if stat_ringkasan:
        if akurasi_pasti is not None:
            teks_stat += "\n\nAkurasi Pasti={:0.3f}".format(akurasi_pasti)
        if rata2_akurasi is not None:
            teks_stat += "\nRata-rata Akurasi={:0.3f}".format(rata2_akurasi)
    if ukuran_gambar is None:
        ukuran_gambar = plt.rcParams.get('figure.figsize')
    if not xyticks:
        kategori = False
    plt.figure(figsize=ukuran_gambar)
    sns.heatmap(cf, annot=label_kotak, fmt="", cmap=cmap, cbar=barwarna, xticklabels=kategori, yticklabels=kategori)
    if xyplotlabels:
        plt.ylabel('Actual')
        plt.xlabel('Predicted' + teks_stat)
    else:
        plt.xlabel(teks_stat)
    if judul:
        plt.title(judul)
    plt.show()

# Membaca data yang telah diproses
data = pd.read_csv('dataset_stunting_samarinda_modif.csv')

# Mengganti nama kolom 'TB/U' menjadi 'Kelas'
data = data.rename(columns={'TB/U': 'Kelas'})

# Menghapus kolom non-numerik dan tidak relevan
data = data.drop(['Nama', 'Tanggal Pengukuran'], axis=1)

# Mengganti nilai bermasalah di kolom 'JK'
data['JK'] = data['JK'].str.strip().replace({'L\xa0': 'L', 'P\xa0': 'P'})

# Menangani nilai numerik bermasalah
data['Berat'] = data['Berat'].str.replace(',', '.').astype(float)

# Mengonversi kolom kategorikal menjadi numerik menggunakan one-hot encoding
kolom_kategorikal = ['JK', 'BB/U', 'BB/TB', 'Naik Berat Badan']
data = pd.get_dummies(data, columns=kolom_kategorikal, drop_first=True)

# Memastikan tidak ada kehilangan data setelah konversi
print(f"Bentuk dataset setelah konversi: {data.shape}")

# Memisahkan fitur dan variabel target
X = data.drop('Kelas', axis=1)
y = data['Kelas'].map({'Tidak Stunting': 0, 'Stunting': 1})

# Menormalisasi fitur
scaler = MinMaxScaler()
X_normalized = scaler.fit_transform(X)

# Membagi data menjadi training dan testing set
x_train, x_test, y_train, y_test = train_test_split(X_normalized, y, test_size=0.2, random_state=42)

# Seleksi fitur menggunakan RFE
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rfe = RFE(estimator=rf, n_features_to_select=10)
x_train_rfe = rfe.fit_transform(x_train, y_train)
x_test_rfe = rfe.transform(x_test)

# Konfigurasi GA
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)

toolbox = base.Toolbox()
toolbox.register("attr_bool", np.random.randint, 0, 2)
toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_bool, n=len(X.columns))
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
```

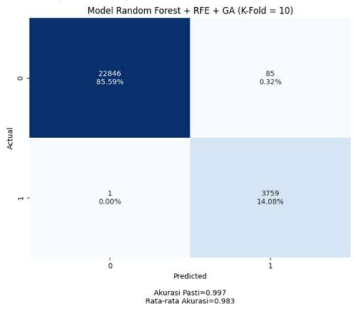
```python
# Fungsi evaluasi
def evalRF(individual):
    selected_features = [i for i, bit in enumerate(individual) if bit == 1]
    if len(selected_features) == 0:
        return 0,  # Hindari subset fitur kosong
    X_subset = X_normalized[:, selected_features]
    selector = RFE(rf, n_features_to_select=min(2, len(selected_features)), step=1)
    X_selected = selector.fit_transform(X_subset, y)
    y_pred = cross_val_predict(rf, X_selected, y, cv=5)  # Mengurangi jumlah CV fold
    return np.mean(cross_val_predict(rf, X_selected, y, cv=5) == y),  # Mengembalikan akurasi

toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=0.05)
toolbox.register("select", tools.selTournament, tournsize=3)
toolbox.register("evaluate", evalRF)

# Menjalankan GA dengan pengaturan yang lebih cepat
population = toolbox.population(n=10)  # Mengurangi ukuran populasi
ngen = 10  # Mengurangi jumlah generasi
cxpb = 0.5
mutpb = 0.3

result = algorithms.eaSimple(population, toolbox, cxpb, mutpb, ngen, stats=None, halloffame=None, verbose=False)

# Menggunakan fitur terbaik yang ditemukan oleh GA
best_individual = tools.selBest(population, 1)[0]
selected_features = [i for i, bit in enumerate(best_individual) if bit == 1]
X_best = X_normalized[:, selected_features]

# Merekam hasil K-Fold 10 kali
kfolds = 10
repeats = 10
akurasi_list = []

# Menggunakan KFold untuk cross-validation
kf = KFold(n_splits=kfolds, shuffle=True, random_state=1)
fold = 1
```

```python
# Melakukan cross-validation dengan variasi
for i in range(repeats):
    print(f"Rekaman {i+1}")

    # Mendefinisikan classifier Random Forest dengan parameter yang sedikit berbeda untuk setiap iterasi
    rf_classifier = RandomForestClassifier(
        n_estimators=10 + i,  # Mengubah jumlah estimators
        criterion='gini',
        max_depth=1 + i % 3,  # Mengubah kedalaman maksimal pohon
        min_samples_split=100 + i,  # Mengubah jumlah minimal sampel untuk split
        min_samples_leaf=50 - i * 5,  # Mengubah jumlah minimal sampel per daun
        max_features=0.1 + (i % 5) * 0.02,  # Mengubah jumlah fitur yang dipertimbangkan untuk split
        class_weight='balanced',
        random_state=42 + i  # Mengubah random state
    )

    y_pred = cross_val_predict(rf_classifier, X_best, y, cv=kfolds)
    cf_matrix = confusion_matrix(y, y_pred)
    akurasi = np.trace(cf_matrix) / float(np.sum(cf_matrix))
    akurasi_list.append(akurasi)
    print(f"Akurasi: {akurasi:.3f}")
    class_report = classification_report(y, y_pred, target_names=['Tidak Stunting', 'Stunting'])
    print(class_report)

# Menghitung akurasi rata-rata
rata2_akurasi = np.mean(akurasi_list)
akurasi_terakhir = akurasi_list[-1]
print(f"Rata-rata Akurasi: {rata2_akurasi:.3f}")
print(f"Akurasi Pasti (Iterasi Terakhir): {akurasi_terakhir:.3f}")

# Menggambar plot matriks kebingungan untuk iterasi terakhir dengan rata-rata akurasi dan akurasi pasti
buat_matriks_kebingungan(cf_matrix, ukuran_gambar=(8, 6), barwarna=False, judul="Model Random Forest + RFE + GA (K-Fold = 10)", akurasi_pasti=akurasi_terakhir, rata2_akurasi=rata2_akurasi)
```

```
Bentuk dataset setelah konversi: (26691, 19)
/usr/local/lib/python3.10/dist-packages/deap/creator.py:185: RuntimeWarning: A class named 'FitnessMax' has already been created and it will be overwritten. Consider deleting previous creation of that class or rename it.
  warnings.warn("A class named '{0}' has already been created and it "
/usr/local/lib/python3.10/dist-packages/deap/creator.py:185: RuntimeWarning: A class named 'Individual' has already been created and it will be overwritten. Consider deleting previous creation of that class or rename it.
  warnings.warn("A class named '{0}' has already been created and it "
Rekaman 1
Akurasi: 0.996
              precision    recall  f1-score   support

Tidak Stunting       1.00      1.00      1.00     22931
      Stunting       0.98      1.00      0.99      3760

      accuracy                           1.00     26691
     macro avg       0.99      1.00      0.99     26691
  weighted avg       1.00      1.00      1.00     26691

Rekaman 2
Akurasi: 0.997
              precision    recall  f1-score   support

Tidak Stunting       1.00      1.00      1.00     22931
      Stunting       1.00      0.98      0.99      3760

      accuracy                           1.00     26691
     macro avg       1.00      0.99      0.99     26691
  weighted avg       1.00      1.00      1.00     26691

Rekaman 3
Akurasi: 0.893
              precision    recall  f1-score   support

Tidak Stunting       1.00      0.88      0.93     22931
      Stunting       0.57      0.98      0.72      3760

      accuracy                           0.89     26691
     macro avg       0.78      0.93      0.83     26691
  weighted avg       0.94      0.89      0.90     26691

Rekaman 4
Akurasi: 0.955
              precision    recall  f1-score   support

Tidak Stunting       0.99      0.96      0.97     22931
      Stunting       0.78      0.94      0.85      3760

      accuracy                           0.95     26691
     macro avg       0.89      0.95      0.91     26691
  weighted avg       0.96      0.95      0.96     26691
```

```
Rekaman 5
Akurasi: 0.997
              precision    recall  f1-score   support

Tidak Stunting       1.00      1.00      1.00     22931
      Stunting       1.00      0.98      0.99      3760

      accuracy                           1.00     26691
     macro avg       1.00      0.99      0.99     26691
  weighted avg       1.00      1.00      1.00     26691

Rekaman 6
Akurasi: 0.998
              precision    recall  f1-score   support

Tidak Stunting       1.00      1.00      1.00     22931
      Stunting       0.99      0.99      0.99      3760

      accuracy                           1.00     26691
     macro avg       0.99      1.00      1.00     26691
  weighted avg       1.00      1.00      1.00     26691

Rekaman 7
Akurasi: 0.995
              precision    recall  f1-score   support

Tidak Stunting       1.00      1.00      1.00     22931
      Stunting       0.98      0.99      0.98      3760

      accuracy                           1.00     26691
     macro avg       0.99      0.99      0.99     26691
  weighted avg       1.00      1.00      1.00     26691

Rekaman 8
Akurasi: 1.000
              precision    recall  f1-score   support

Tidak Stunting       1.00      1.00      1.00     22931
      Stunting       1.00      1.00      1.00      3760

      accuracy                           1.00     26691
     macro avg       1.00      1.00      1.00     26691
  weighted avg       1.00      1.00      1.00     26691

Rekaman 9
Akurasi: 0.999
              precision    recall  f1-score   support

Tidak Stunting       1.00      1.00      1.00     22931
      Stunting       1.00      0.99      0.99      3760

      accuracy                           1.00     26691
     macro avg       1.00      1.00      1.00     26691
  weighted avg       1.00      1.00      1.00     26691
```

```
Rekaman 10
Akurasi: 0.997
              precision    recall  f1-score   support

Tidak Stunting       1.00      1.00      1.00     22931
      Stunting       0.98      1.00      0.99      3760

      accuracy                           1.00     26691
     macro avg       0.99      1.00      0.99     26691
  weighted avg       1.00      1.00      1.00     26691

Rata-rata Akurasi: 0.983
Akurasi Pasti (Iterasi Terakhir): 0.997
```



SELEKSI FITUR RFE

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import RFE

# Membaca data yang telah diunggah
data_path = 'dataset_stunting_samarinda_terbaru.csv'
data = pd.read_csv(data_path)

# Mengganti nama kolom 'TB/U' menjadi 'Kelas'
data = data.rename(columns={'TB/U' : 'Kelas'})

# Menangani nilai bermasalah di kolom 'JK'
data['JK'] = data['JK'].str.strip().replace({'L\xa0': 'L', 'P\xa0': 'P'})

# Menangani nilai numerik bermasalah
data['Berat'] = data['Berat'].str.replace(',', '.').astype(float)

# Mengonversi kolom 'JK' menjadi numerik menggunakan label encoding
data['JK'] = data['JK'].map({'L': 0, 'P': 1})

# Mengonversi kolom 'Naik Berat Badan' menjadi numerik menggunakan label encoding
data['Naik Berat Badan'] = data['Naik Berat Badan'].map({'N': 0, 'O': 1, 'T': 2})

# Mengonversi kolom 'BB/U' dan 'BB/TB' menjadi numerik menggunakan label encoding
data['BB/U'] = data['BB/U'].map({'Normal': 0, 'Kurang': 1, 'Risiko Lebih': 2, 'Sangat Kurang': 3})
data['BB/TB'] = data['BB/TB'].map({'Normal': 0, 'Gizi Buruk': 1, 'Gizi Kurang': 2, 'Gizi Lebih': 3, 'Obesitas': 4, 'Risiko Gizi Lebih': 5})

# Menghapus baris yang mengandung nilai NaN
data_cleaned = data.dropna()

# Memisahkan fitur dan variabel target
X_cleaned = data_cleaned.drop('Kelas', axis=1)
y_cleaned = data_cleaned['Kelas'].map({'Tidak Stunting': 0, 'Stunting': 1})

# Menormalisasi fitur
scaler = MinMaxScaler()
X_normalized_cleaned = scaler.fit_transform(X_cleaned)

# Seleksi fitur menggunakan RFE dengan jumlah fitur yang lebih banyak (10 fitur terbaik)
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rfe = RFE(estimator=rf, n_features_to_select=10)
rfe.fit(X_normalized_cleaned, y_cleaned)
```

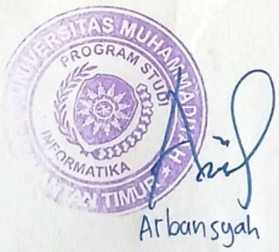
```python
# Mendapatkan nama-nama fitur yang dipilih
fitur_terpilih_10_cleaned = X_cleaned.columns[rfe.support_]

# Mendapatkan kepentingan fitur dari model Random Forest yang digunakan oleh RFE
importances_10 = rfe.estimator_.feature_importances_

# Membuat DataFrame untuk kepentingan fitur
feature_importances_10 = pd.DataFrame({
    'Fitur': fitur_terpilih_10_cleaned,
    'Kepentingan': importances_10
})

# Mengurutkan fitur berdasarkan kepentingannya
feature_importances_10_sorted = feature_importances_10.sort_values(by='Kepentingan', ascending=False)
print(feature_importances_10_sorted)
```

```
             Fitur  Kepentingan
6          ZS TB/U     0.647155
5          ZS BB/U     0.157594
4            BB/U     0.094062
1           Berat     0.031782
8         ZS BB/TB     0.030855
7            BB/TB     0.019662
2           Tinggi     0.015596
3             LILA     0.001236
9  Naik Berat Badan     0.000631
0               JK     0.000527
```

Lampiran 5 Turnitin Skripsi

SKRIPSI BIMA SATRIA

by Teknik Informatika Universitas Muhammadiyah Kalimantan Timur



Submission date: 19-Jul-2024 08:44AM (UTC+0800)
Submission ID: 2418913734
File name: skah_Skripsi_Bima_Satria_2011102441102_-_Copy_-_BIMA_SATRIA.docx (830.28K)
Word count: 9823
Character count: 60824

SKRIPSI BIMA SATRIA

ORIGINALITY REPORT

17%	12%	11%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Ari Ahmad Dhani, Taghfirul Azhima Yoga Siswa, Wawan Joko Pranoto. "Perbaikan Akurasi Random Forest Dengan ANOVA Dan SMOTE Pada Klasifikasi Data Stunting", Teknika, 2024 Publication	5%
2	developers.google.com Internet Source	1%
3	repository.ub.ac.id Internet Source	1%
4	dspace.umkt.ac.id Internet Source	1%
5	satudata.pasuruankota.go.id Internet Source	<1%
6	api.repository.poltekesos.ac.id Internet Source	<1%
7	eprints.uad.ac.id Internet Source	<1%
8	jurnal.umt.ac.id Internet Source	