Sistem Deteksi *Stunting* pada Balita Berbasis Web Menggunakan Metode *Random Forest*

Silvia Nanda Syafa Iswahyudi[1], Ricky Eka Putra[2],

[1,2] Program Studi S1 Teknik Informatika, Universitas Negeri Surabaya
[1]silvia.20070@mhs.unesa.ac.id
[2]rickyeka@unesa.ac.id

***Abstrak*— *Stunting* adalah kondisi gagal tumbuh yang sering dialami oleh balita di Indonesia dan dapat berdampak negatif jangka panjang terhadap perkembangan fisik serta kognitif anak. Penelitian ini bertujuan untuk mengembangkan sistem deteksi *stunting* berbasis web menggunakan *Random Forest*, yang dikenal efektif dalam menangani data yang tidak seimbang. *Dataset* yang digunakan dalam penelitian ini berasal dari situs web *Kaggle*, yang terdiri dari 10.000 data balita di Indonesia. Metodologi yang digunakan adalah CRISP-DM, dengan tahapan mulai dari pemahaman bisnis hingga implementasi aplikasi web menggunakan *Flask* dan *Railway* untuk *deployment*. Evaluasi model dilakukan menggunakan *confusion matrix* dan hasilnya menunjukkan bahwa model dengan *tuning hyperparameter* pada pembagian data 90:10 mencapai akurasi sebesar 85%. Model ini kemudian diintegrasikan ke dalam aplikasi web. Sistem yang dikembangkan menyediakan dua opsi deteksi, yaitu deteksi individu untuk mendeteksi status *stunting* pada satu anak, dan deteksi kelompok yang memungkinkan pengguna untuk mengunggah data beberapa anak sekaligus dalam bentuk *file* CSV. Aplikasi ini di-*deploy* menggunakan platform *Railway* yang memudahkan pengelolaan dan pemeliharaan aplikasi, serta memberikan kemampuan untuk melakukan *update* secara otomatis melalui *GitHub*. Diharapkan aplikasi ini dapat memberikan kontribusi dalam upaya deteksi aplikasi ini dapat memberikann kontribusi dalam upaya deteksi dini *stunting* secara efektif dan efisien, terutama di daerah dengan akses layanan kesehatan yang terbatas.**

***Kata Kunci*— *Stunting*, *Random Forest*, Deteksi dini, Aplikasi web, dan Balita.**

I. PENDAHULUAN

Stunting adalah masalah kesehatan yang sering ditemui di Indonesia, yang disebabkan oleh kekurangan gizi dan kekurangan asupan nutrisi dalam periode waktu yang lama. Kondisi ini menyebabkan anak tumbuh lebih pendek dibandingkan anak seusianya, mengalami keterlambatan kognitif, serta terganggunya perkembangan fisik dan otak. Salah satu metode pemeriksaan antropometri untuk mendeteksi *stunting* adalah mengukur tinggi badan, yang dapat mengindikasikan status gizi anak. Diagnosis *stunting* dilakukan dengan membandingkan *z-score* tinggi badan terhadap umur berdasarkan grafik pertumbuhan yang telah diakui secara global. Di Indonesia, grafik pertumbuhan yang digunakan untuk menegakkan diagnosis *stunting* adalah grafik yang disusun oleh *World Health Organization* (WHO) pada tahun 2005 [1].

Berdasarkan data statistik PBB tahun 2020, lebih dari 149 juta balita atau 22% dari semua balita di seluruh dunia mengalami *stunting*. Dari data tersebut, sebanyak 6,3 juta adalah balita di Indonesia. UNICEF menyatakan bahwa faktor-faktor yang menyebabkan *stunting* antara lain anak kekurangan gizi selama dua tahun usia, kekurangan nutrisi ibu selama kehamilan, dan kondisi sanitasi yang buruk [2]. Hasil Survei Status Gizi Indonesia (SSGI) yang diumumkan oleh Kementerian Kesehatan pada Rapat Kerja Nasional BKKBN, menunjukkan bahwa tingkat *stunting* di Indonesia turun dari 24,4% pada tahun 2021 menjadi 21,6% pada tahun 2022. Sedangkan standar WHO untuk prevalensi *stunting* adalah kurang dari 20%. Indonesia sendiri menarget tingkat prevalensi *stunting* 14% pada tahun 2024. *Stunting* bukakn hanya masalah tinggi badan, yang paling berbahaya adalah keterbelakangan mental, rendahnya kemampuan belajar, dan munculnya penyakit kronis juga [3].

Di era teknologi saat ini, kemajuan teknologi membuat semua aspek kehidupan manusia sangat bergantung pada teknologi komputasi. Seiring dengan meningkatnya kebutuhan untuk memproses data, diperlukan model yang mampu menjelaskan data tersebut melalui penerapan *machine learning*. *Machine learning* adalah bagian dari ilmu intelijen dalam teknologi perangkat lunak yang menggunakan algoritma kecerdasan buatan yang didasarkan pada pengetahuan data untuk menyelesaikan masalah secara mandiri ketika diberikan data baru. Saat ini, *machine learning* menjadi topik yang sangat populer dalam penelitian dan pengembangan teknologi [4]. Studi menunjukkan 86% organisasi kesehatan mengandalkan *machine learning* untuk meningkatkan operasional bisnis [5].

Decision Tree merupakan algoritma *machine learning* yang banyak digunakan untuk keperluan prediksi dan klasifikasi [6]. Metode ini menghasilkan pohon keputusan dan aturan. Namun, ketika data yang dikelola berukuran besar dan pengambilan keputusan membutuhkan banyak waktu, sering terjadi *overlapping*. Semakin dalam pohon dapat membuat model menjadi *overfitting* [7]. *Overfitting* terjadi ketika model menjadi terlalu kompleks dan mengandung *noise* atau perubahan acak dalam data pelatihan, sehingga performanya buruk pada data yang belum pernah dilihat (data uji) [8]. *Random Forest* adalah metode untuk menyelesaikan masalah ini. Metode *Random Forest* mengatasi masalah *overfitting* dengan membangun banyak pohon keputusan (*trees*) pada sub-sampel acak dari data pelatihan dan menggunakan rata-rata dari prediksi semua pohon untuk membuat keputusan akhir. *Random Forest* menggunakan teknik *bootstrap aggregating* (*bagging*), di mana data pelatihan diambil sampelnya dengan penggantian untuk melatih masing-masing pohon. Ini

memperkenalkan variasi di antara pohon dan lebih efektif dalam mengurangi varians, yang merupakan penyebab utama *overfitting*. Selain itu, pada setiap *node*, *Random Forest* memilih *subset* acak dari fitur-fitur untuk menemukan *split* terbaik. Ini memperkenalkan lebih banyak variasi di antara pohon dan lebih efektif dalam mengurangi korelasi di antara mereka [9].

Data *stunting* sering kali ditandai dengan *imbalance data*, di mana jumlah data anak *stunting* dan tidak *stunting* memiliki jumlah yang tidak seimbang. Ini dapat mengakibatkan model *machine learning* mengalami kecenderungan terhadap kelas mayoritas dan tidak akurat dalam mengidentifikasi anak *stunting.* Dalam mengatasi tantangan ini, penggunaan model *Random Forest* telah terbukti efektif. Menurut penelitian yang diterbitkan dalam *Journal of Machine Learning Research* oleh Leo Breiman, *Random Forest* memiliki keunggulan dalam menangani masalah *imbalance data*. Pendekatan *ensamble* memperhitungkan sampel acak dari setiap kelas saat membangun setiap pohon keputusan, sehingga mengurangi risiko kecenderungan terhadap kelas mayoritas dan meningkatkan kemampuan model untuk mengidentifikasi anak-anak yang menderita *stunting*.

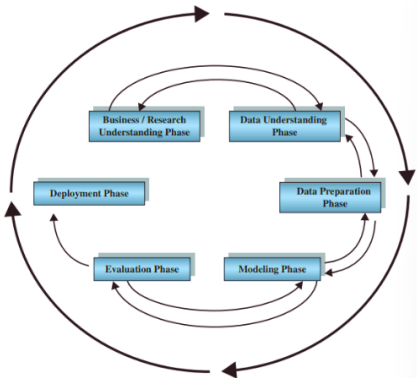
Seperti yang ditunjukkan dalam penelitian oleh [10] mengenai "Klasifikasi Penyakit Jantung Menggunakan *Decision Tree* dan *Random Forest*" menunjukkan bahwa model *Random Forest* memberikan akurasi yang lebih baik dibandingkan dengan *Decision Tree* dalam mengklasifikasikan penyakit jantung, dengan akurasi sebesar 81,82% untuk *Random Forest* dan 77,44% untuk *Decision Tree*. Penelitian tersebut menggunakan *Heart Disease Cleveland UCI Dataset* dengan 297 entri data. Hasil penelitian tersebut menunjukkan bahwa akurasi *Random Forest* lebih tinggi dibandingkan dengan *Decision Tree* dalam mengklasifikasikan penyakit jantung. Ini menunjukkan bahwa *Random Forest* lebih efektif dan dapat diandalkan dalam menangani variasi serta kompleksitas data dibandingkan dengan *Decision Tree*. *Random Forest* mencapai akurasi yang lebih tinggi karena menggabungkan hasil dari beberapa pohon keputusan yang dibangun secara acak dari *subset* data, yang membantu mengurangi risiko *overfitting* yang sering terjadi pada *Decision Tree* tunggal. Dengan menggabungkan hasil dari banyak pohon, *Random Forest* mampu menangkap lebih banyak variasi dalam data dan memberikan deteksi yang lebih akurat.

Pengembangan web melibatkan proses merancang, membangun, dan mengelola situs web. Dengan kemajuan teknologi saat ini, ada berbagai alat yang tersedia untuk pengembangan web, salah satunya yaitu kerangka kerja web yang dikembangkan dengan bahasa pemrograman *Python*. *Django* dan *Flask* adalah dua kerangka kerja web paling populer yang dibangun dengan *Python*. *Flask* adalah kerangka kerja web *Python* yang ringan dan membantu dalam pembuatan kerangka dasar situs web. *Flask* juga merupakan pilihan baik bagi pengembang pemula karena kemudahannya dalam dipelajari dan digunakan [11]. Untuk menerapkan model *machine learning* ke dalam sebuah aplikasi web, *Flask* digunakan sebagai kerangka kerja untuk membangun aplikasi web dan layanan *cloud Railway* akan digunakan untuk *deploy* aplikasi tersebut.

Berdasarkan penjelasan terkait penelitian sebelumnya tentang efektivitas model *Random Forest* menangani ketidakseimbangan kelas dan *overfitting*, penelitian ini bertujuan untuk menerapkan metode *Random Forest* dalam mendeteksi *stunting* pada balita. Dengan memanfaatkan dataset yang mencakup berbagai faktor yang berkaitan dengan pertumbuhan dan perkembangan balita, termasuk data sekunder yang relevan, penelitian ini akan menggunakan model *Random Forest* yang telah dilatih. Selain itu, penelitian ini juga akan mengimplementasikan model *Random Forest* ke dalam sebuah aplikasi web dengan memanfaatkan *framework Flask* dan *Railway* untuk mendeteksi *stunting* pada balita.

II. METODE PENELITIAN

Cross Industry Standard Process for Data Mining (CRISP-DM) adalah metode penelitian yang digunakan dalam penelitian ini karena dikenal sebagai pendekatan netral terhadap teknologi dan tidak terbatas pada industri tertentu, sehingga menjadi standar *de facto* dalam proses *data mining* [12]. Gbr. 1 adalah alur CRISP-DM yang digunakan dalam penelitian ini:



Gbr. 1 Alur Penelitian

A. *Business Understanding Phase*

Penelitian terkait pengembangan sistem deteksi *stunting* pada balita berbasis web menggunakan metode *Random Forest* bertujuan untuk memberikan solusi yang efektif dalam mengidentifikasi potensi *stunting* pada balita secara dini. Sasaran utama penelitian ini adalah para orang tua, tenaga kesehatan, dan pihak berkepentingan lainnya yang terlibat dalam pemantauan pertumbuhan anak. Dengan menyediakan informasi yang tepat dan cepat terkait risiko *stunting*, sistem ini diharapkan dapat membantu dalam memberikan perawatan dan intervensi yang tepat waktu untuk mencegah dampak negatifnya terhadap kesehatan dan perkembangan anak.

B. *Data Understanding Phase*

Tujuan dari tahap *data understanding* adalah untuk menganalisis data yang telah dikumpulkan. Data yang digunakan dalam penelitian ini diperoleh dari situs web *Kaggle*. *Dataset* tersebut berasal dari data *stunting* di Indonesia pada tahun 2022 dengan total sebanyak 10.000 entri. Tabel 1

merupakan tipe data dan deskripsi atribut *dataset* yang akan diterapkan dalam penelitian ini.

TABEL I
TIPE DATA DAN DESKRIPSI ATRIBUT

Nama Atribut	Tipe Data	Deskripsi
Gender	*String*	Jenis kelamin balita (*Male/Female*)
Age	*Integer*	Umur balita dalam bulan
Birth Weight	*Float*	Berat bayi saat lahir dalam kilogram
Birth Legth	*Integer*	Panjang bayi saat lahir dalam centimeter
Body Weight	*Float*	Berat badan balita saat ini dalam kilogram
Body Length	*Float*	Tinggi badan balita saat ini dalam centimeter
Breastfeeding	*String*	Status asi eksklusif (*Yes/No*)
Stunting	*String*	Status *stunting* balita (*Yes/No*)

C. *Data Preparation Phase*

Data preparation juga dikenal sebagai *data preprocessing*, adalah teknik yang diterapkan pada *dataset* untuk menghilangkan *noise*, *missing value*, *error*, serta data yang tidak relevan. Proses ini bertujuan untuk mempermudah pengolahan data dan mengekstrak informasi yang relevan. Pada tahap ini, *dataset* yang diperoleh akan dilakukan *data cleaning*, *feature encoding*, *data splitting*, dan *handling imbalance data*.

D. *Modeling Phase*

Tahap *modeling* dalam penelitian ini menggunakan *Random Forest* untuk deteksi *stunting,* yang menggabungkan banyak *Decision Tree* melalui teknik *bootstrap sampling* dan pemilihan fitur acak.

Modeling dalam penelitian ini dilakukan dengan beberapa pendekatan:

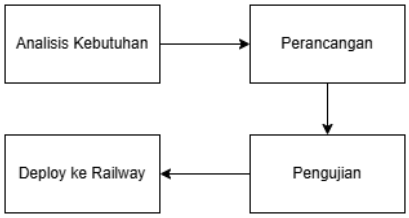
1. Data Asli: Model dibangun tanpa *resampling* sebagai *baseline*.
2. Data dengan *Resampling*: SMOTE digunakan untuk menyeimbangkan kelas, meningkatkan *recall* dan *f1-score* pada kelas minoritas.
3. *Tuning Hyperparameter*: Parameter seperti *n_estimators*, *max_features*, *max_depth*, *min_samples_split*, dan *min_samples_leaf* dioptimalkan menggunakan *Random Search* untuk hasil yang lebih akurat.

Pendekatan ini memungkinkan evaluasi model yang lebih komprehensif dalam deteksi *stunting*.

E. *Evaluation Phase*

Tahap evaluasi model memastikan kesesuaian dengan pemahaman bisnis yang ditetapkan dan menguji kinerja model terhadap tujuan penelitian. Data uji dan *K-Fold Cross Validation* dengan k yang ditentukan sebagai 5 dan 10 yang digunakan untuk menghindari *overfitting* dan menjaga konsistensi performa. Dalam *K-Fold Cross Validation*, data latih dibagi menjadi beberapa *subset*, memungkinkan tiap *fold* bergantian menjadi data uji untuk melatih model secara menyeluruh. Evaluasi kinerja model kemudian dilakukan menggunakan *confusion matrix* untuk menghitung akurasi, presisi, *recall*, dan *f1-score*. Jika hasil sesuai dengan tujuan awal, model dapat melanjutkan ke tahap *deployment*.

F. *Deployment Phase*



Gbr. 2 Alur *Deployment*

Gbr. 2 mencakup beberapa langkah penting dalam pengembangan aplikasi web deteksi *stunting*. Pertama, analisis kebutuhan dilakukan untuk mengidentifikasi kebutuhan fungsional, seperti penerimaan *input* untuk deteksi individu atau kelompok serta validasi data, dan non-fungsional, seperti antarmuka yang ramah pengguna dan kompatibilitas lintas perangkat. Kedua, tahap perancangan menggunakan *Flask* sebagai *framework* untuk membangun aplikasi dengan fitur deteksi individu dan kelompok serta integrasi model deteksi *stunting*. Setelah itu, pengujian dilakukan dengan metode *blackbox* untuk memastikan semua fitur berjalan sesuai spesifikasi. Terakhir, aplikasi di-*deploy* menggunakan *Railway*, yang mendukung aplikasi *Python* berbasis *Flask*, dengan proses penyiapan lingkungan server dan pengujian akhir di lingkungan produksi.

III. HASIL DAN PEMBAHASAN

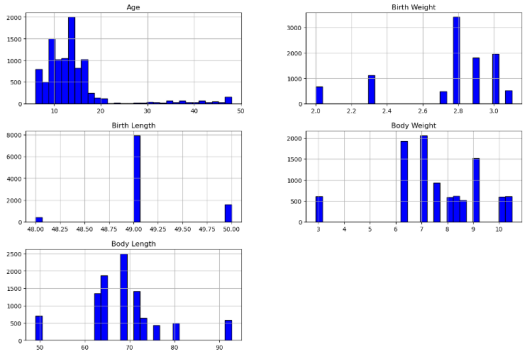
A. *Business Understanding Phase*

Penelitian ini dimulai dengan pemahaman masalah dari perspektif bisnis, yaitu kesulitan dalam mendeteksi *stunting* dini yang dapat menyebabkan intervensi terlambat dan menurunkan kualitas hidup anak. Untuk menjawab kebutuhan ini, data kesehatan anak, seperti jenis kelamin, usia, berat dan panjang bayi, berat dan tinggi badan, serta status gizi, dikumpulkan dan dianalisis. Data tersebut digunakan untuk membangun model deteksi berbasis *Random Forest* yang diintegrasikan ke dalam aplikasi web. Aplikasi ini dirancang untuk diakses oleh orang tua dan masyarakat umum, memberikan alat pemantauan mandiri yang efektif sekaligus meningkatkan kesadaran akan pentingnya nutrisi. Dengan solusi ini, penelitian diharapkan dapat berkontribusi dalam mendukung kesehatan anak secara lebih luas.

B. *Data Understanding Phase*

Tahap ini merupakan proses analisis yang bertujuan untuk memberikan pemahaman menyeluruh mengenai data yang digunakan dengan melihat distribusi tiap atribut, menganalisis korelasi antar atribut, memeriksa *missing value*, dan memeriksa data duplikat.

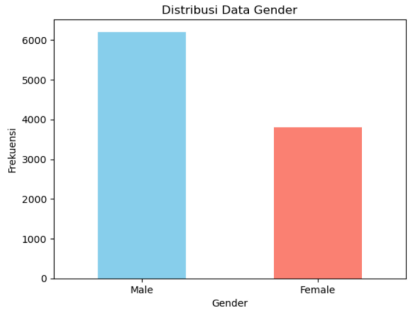
1) Distribusi Data dalam Dataset



Gbr. 3 Distribusi Data Numerik

Gbr. 3 merupakan distribusi data numerik pada *dataset stunting* balita yang menunjukkan beberapa pola menarik. Pertama, distribusi umur balita dalam *dataset* sebagian besar berkisar antara 0 hingga 20 bulan, sementara beberapa balita berusia lebih dari 20 bulan namun jumlahnya sangat sedikit. Berat bayi saat lahir menunjukkan nilai yang umum di sekitar 2.8 kg dan 3.0 kg, dengan beberapa balita memiliki berat lahir sekitar 2.0 atau lebih berat dari 3.0 kg, meskipun jumlahnya tidak signifikan. Panjang bayi saat lahir berkisar di sekitar 49 cm, dengan sedikit data balita memiliki panjang lahir 50 cm atau lebih rendah dari 48 cm. Distribusi berat badan balita menunjukkan puncak di sekitar 7 kg, dengan variasi signifikan, mencakup berat badan sekitar 3 kg hingga 10 kg. Beberapa kelompok berat badan menonjol di sekitar 6 kg, 7 kg, dan 9 kg. Tinggi badan balita sebagian besar berkisar antara 60 cm hingga 80 cm, dengan puncak distribusi di sekitar 70 cm, sementara beberapa balita memiliki tinggi badan sekitar 90 cm, namun jumlahnya relatif kecil.

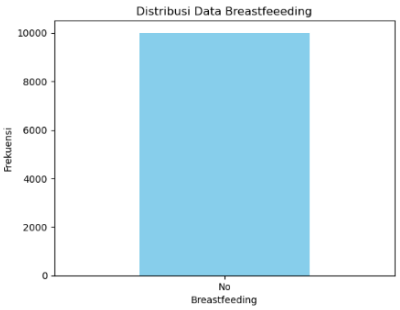
```
Male      6204
Female    3796
Name: Gender, dtype: int64
```



Gbr. 4 Distribusi Data *Gender*

Gbr. 4 menunjukkan distribusi atribut *gender* yang menunjukkan distribusi dengan *gender male* (laki-laki) sebanyak 6204 dan data dengan *gender female* (perempuan) sebanyak 3796.

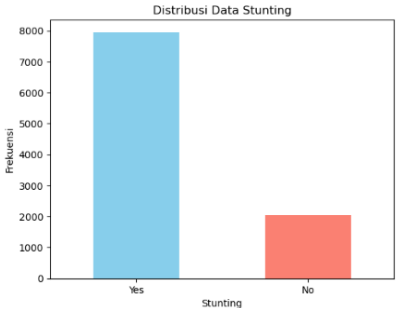
```
No    10000
Name: Breastfeeding, dtype: int64
```



Gbr. 5 Distribusi Data *Breastfeeding*

Gbr. 5 menunjukkan distribusi pada atribut *breastfeeding* yang memperlihatkan bahwa seluruh data dalam atribut ini ada pada satu kelas, yaitu "*No*". Dikarenakan atribut ini hanya memiliki satu kelas dan tidak memberikan informasi yang berguna untuk pemodelan, kolom ini akan dihilangkan dalam *dataset*.

```
Yes    7955
No     2045
Name: Stunting, dtype: int64
```

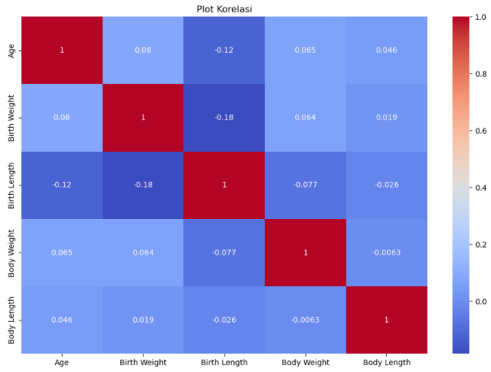


Gbr. 6 Distribusi Data *Stunting*

Gbr. 6 merupakan distribusi atribut *stunting* yang menunjukkan bahwa dari total 10.000 data balita, sebanyak 7955 data menunjukkan balita mengalami *stunting* dan 2045 data menunjukkan balita tidak mengalami *stunting*.

2) Plot Korelasi

Plot korelasi digunakan untuk menganalisis hubungan antara dua atribut dalam *dataset*. Dalam penelitian ini, plot korelasi membantu mengidentifikasi hubungan signifikan antar atribut dalam *dataset*. Korelasi dengan nilai antara 1 sampai -1 menggambarkan hubungan, di mana korelasi positif sempurna dengan nilai 1, korelasi negatif sempurna dengan nilai -1, dan tidak ada korelasi dengan nilai 0. Plot korelasi yang terbentuk dari *dataset* yang digunakan dapat dilihat pada Gbr. 7 berikut.



Gbr. 7 Plot Korelasi

3) Memeriksa Missing Value

Pemeriksaan *missing value* dilakukan untuk mengidentifikasi dan menangani data yang hilang pada setiap atribut.

```
Missing Values:
Gender            0
Age               0
Birth Weight      0
Birth Length      0
Body Weight       0
Body Length       0
Breastfeeding     0
Stunting          0
dtype: int64
```

Gbr. 8 Hasil Pemeriksaan *Missing Value*

Berdasarkan Gbr. 8 tidak ditemukan adanya *missing value* dalam *dataset*. Ini mengindikasikan bahwa data tersebut sudah lengkap dan tidak memerlukan penanganan khusus terkait *missing value.*

4) Memeriksa Data Duplikat

Selanjutnya, pemeriksaan data duplikat dilakukan untuk memastikan bahwa setiap data balita hanya muncul sekali dalam *dataset*. Proses pemeriksaan data duplikat ditunjukkan pada Gbr. 9 berikut.

```
# Memeriksa data duplikat
duplicate_rows = data[data.duplicated(keep=False)]

# Menampilkan baris duplikat
print(duplicate_rows)
```

Gbr. 9 Memeriksa Data Duplikat

Berdasarkan hasil eksekusi kode tersebut, terdapat 3816 baris duplikat dalam *dataset*, yang penting untuk dihapus guna memastikan keakuratan analisis lebih lanjut.

C. Data Preparation Phase

1) Data Cleaning

Data cleaning bertujuan untuk mengidentifikasi dan memperbaiki atau menghapus data yang duplikat atau menghilangkan kolom yang tidak memberikan informasi untuk analisis. Langkah pertama adalah mengubah kolom *dataset* ke dalam Bahasa Indonesia untuk mempermudah pemahaman dan interpretasi data. Proses *rename* kolom *dataset* ini ditunjukkan pada Gbr. 10 berikut.

```
# Mengubah kolom dataset ke bahasa indonesia
data.rename(columns={"Gender" : "Jenis Kelamin",
                     "Age" : "Umur",
                     "Birth Weight" : "Berat Bayi",
                     "Birth Length" : "Panjang Bayi",
                     "Body Weight" : "Berat Badan",
                     "Body Length" : "Tinggi Badan",
                     "Breastfeeding" : "Asi Eksklusif",
                     "Stunting" : "Gagal Tumbuh"}, inplace=True)
data.head()
```

Gbr. 10 *Rename* Kolom *Dataset*

Selanjutnya, menghapus data duplikat untuk memastikan bahwa analisis dan pemodelan dilakukan pada *dataset* yang unik dan akurat. Proses menghapus data duplikat ditunjukkan pada Gbr. 11 berikut.

```
data = data.drop_duplicates(keep='first')

data
```

Gbr. 11 Menghapus Data Duplikat

Dengan mengatur parameter *keep='first'*, hanya baris pertama dari setiap set duplikat yang dipertahankan sementara baris duplikat lainnya dihapus. Kemudian, menghapus kolom "Asi Eksklusif" dilakukan karena kolom tersebut semua nilainya adalah "*No*", sehingga tidak memberikan informasi yang bermanfaat untuk analisis lebih lanjut. Proses menghapus kolom "Asi Eksklusif" ditunjukkan pada Gbr. 12 berikut.

```
# Menghapus kolom "Asi Eksklusif"
data = data.drop(columns=['Asi Eksklusif'])

data.head()
```

Gbr. 12 Menghapus Kolom Asi Eksklusif

2) Feature Encoding

Feature encoding adalah proses mengubah data kategori atau teks menjadi format numerik yang dapat digunakan oleh algoritma *machine learning*. Pada penelitian ini, metode *mapping* digunakan untuk mengubah nilai kategori menjadi nilai numerik uang unik untuk setiap kategori. Pada proses ini nilai "*Female*" diubah menjadi 0 dan "*Male*" diubah menjadi 1 pada kolom "Jenis Kelamin". Demikian pula, nilai "*No*" diubah menjadi 0 dan nilai "*Yes*" diubah menjadi 1 pada kolom "Gagal Tumbuh". Proses *feature encoding* ditunjukkan pada Gbr. 13 berikut.

```
# Feature encoding menggunakan mapping dengan label encoding secara manual
data['Jenis Kelamin'] = data['Jenis Kelamin'].map({'Female': 0, 'Male': 1})
data['Gagal Tumbuh'] = data['Gagal Tumbuh'].map({'No': 0, 'Yes': 1})

data
```

Gbr. 13 *Feature Encoding* pada *Dataset*

3) Data Splitting

Data splitting merupakan proses membagi *dataset* menjadi dua sub set yaitu data latih dan data uji. Tujuan dari proses ini yaitu untuk memastikan bahwa model dilatih menggunakan satu set data dan diuji menggunakan set data yang berbeda, sehingga kinerjanya dapat dievaluasi dan secara objektif. Pada

penelitian ini, data dipisahkan ke dalam tiga skenario yaitu 90:10, 80:20, dan 70:30 untuk memeriksa performa model pada berbagai pembagian data. Proses ini dimulai dengan memisahkan fitur (X) dan target (y) dari *dataset*, di mana kolom "Gagal Tumbuh" dijadikan target (y) dan kolom lainnya sebagai fitur (X). Selanjutnya, *dataset* dibagi menjadi data latih dan data uji dengan memanfaatkan fungsi *train_test_split* dari pustaka *scikit-learn*. Proses pembagian *dataset* ditunjukkan pada Gbr. 14 di bawah ini.

```
from sklearn.model_selection import train_test_split

# Membuat variabel penentu (X) dan variabel target (y)
X = data.drop('Gagal Tumbuh', axis=1)
y = data['Gagal Tumbuh']

# Membagi data menjadi data latih (90%) dan data uji (10%)
X_train_90, X_test_90, y_train_90, y_test_90 = train_test_split(X, y, test_size=0.1)
print("Jumlah data latih (90:10):", X_train_90.shape[0])
print("Jumlah data uji (90:10):", X_test_90.shape[0])

# Membagi data menjadi data latih (80%) dan data uji (20%)
X_train_80, X_test_80, y_train_80, y_test_80 = train_test_split(X, y, test_size=0.2)
print("\nJumlah data latih (80:20):", X_train_80.shape[0])
print("Jumlah data uji (80:20):", X_test_80.shape[0])

# Membagi data menjadi data latih (70%) dan data uji (30%)
X_train_70, X_test_70, y_train_70, y_test_70 = train_test_split(X, y, test_size=0.3)
print("\nJumlah data latih (70:30):", X_train_70.shape[0])
print("Jumlah data uji (70:30):", X_test_70.shape[0])
```

Gbr. 14 Pemisahan *Dataset* Menjadi Data Latih dan Data Uji

4) Handling Imbalance Data

Handling imbalance data dalam penelitian ini dilakukan menggunakan *Syntjetic Minority Oversamping Technique* (SMOTE), sebuah metode *oversampling* yang bertujuan untuk menyeimbangkan distribusi kelas dengan membuat sampel sintetis untuk kelas minoritas. Proses ini ditunjukkan pada Gbr. 15 di bawah ini.

```
# Mengatasi imbalance dengan SMOTE
from imblearn.over_sampling import SMOTE

smote = SMOTE()
X_train_resampled_90, y_train_resampled_90 = smote.fit_resample(X_train_90, y_train_90)
X_train_resampled_80, y_train_resampled_80 = smote.fit_resample(X_train_80, y_train_80)
X_train_resampled_70, y_train_resampled_70 = smote.fit_resample(X_train_70, y_train_70)
```

Gbr. 15 *Resampling* dengan SMOTE

D. Modeling Phase

1) Modeling dengan Data Asli

Pada tahap ini, digunakan metode *Random Forest* dengan parameter *default* untuk membangun model. Tujuan dari proses ini adalah agar model dapat belajar dari data asli dan mendeteksi hasil berdasarkan pola yang ditemukan dalam data tersebut. Proses ini ditunjukkan pada Gbr. 16 di bawah ini.

```
# Modeling pada data asli
from sklearn.ensemble import RandomForestClassifier

model_90 = RandomForestClassifier()
model_80 = RandomForestClassifier()
model_70 = RandomForestClassifier()

# Modeling dengan split 90:10
model_90.fit(X_train_90, y_train_90)
y_pred_90 = model_90.predict(X_test_90)

# Modeling dengan split 80:20
model_80.fit(X_train_80, y_train_80)
y_pred_80 = model_80.predict(X_test_80)

# Modeling dengan split 70:30
model_70.fit(X_train_70, y_train_70)
y_pred_70 = model_70.predict(X_test_70)
```

Gbr. 16 *Modeling* dengan Data Asli

2) Modeling dengan Data Resampling

Modeling dengan data *resampling* bertujuan untuk membandingkan kinerja model *Random Forest* dalam mendeteksi *stunting* dengan menggunakan teknik *resampling* SMOTE. Proses ini ditunjukkan pada Gbr. 17 berikut.

```
# Modeling pada data resampling
model_resampling_90 = RandomForestClassifier()
model_resampling_80 = RandomForestClassifier()
model_resampling_70 = RandomForestClassifier()

# Modeling dengan split 90:10
model_resampling_90.fit(X_train_resampled_90, y_train_resampled_90)
y_pred_resampling_90 = model_resampling_90.predict(X_test_90)

# Modeling dengan split 80:20
model_resampling_80.fit(X_train_resampled_80, y_train_resampled_80)
y_pred_resampling_80 = model_resampling_80.predict(X_test_80)

# Modeling dengan split 70:30
model_resampling_70.fit(X_train_resampled_70, y_train_resampled_70)
y_pred_resampling_70 = model_resampling_70.predict(X_test_70)
```

Gbr. 17 *Modeling* dengan Data *Resampling*

3) Tuning Hyperparameter

Pada proses ini, dilakukan *tuning* terhadap lima *hyperparameter* pada model *Random Forest*, yaitu *n_estimators* (banyaknya pohon), *max_features* (jumlah fitur yang dipilih untuk setiap pohon), *min_samples_split* (jumlah minimum data untuk membagi simpul), *min_samples_leaf* (jumlah sampel minimal yang harus ada di daun), dan *max_depth* (kedalaman maksimum pohon). Untuk proses *tuning hyperparameter*, digunakan metode *Random Search* dengan mengevaluasi 20 kombinasi acak yang ditentukan dengan parameter *n_iter=20* dari rentang *hyperparameter* yang telah ditetapkan pada Tabel II berikut.

TABEL II
NILAI *HYPERPARAMETER* YANG DITUNING

Hyperparameter	Nilai
max_features	sqrt, log2
max_depth	2, 5, 10, 20, 50, None
n_estimators	100 – 1000 (interval 100)
min_samples_split	2-5
min_samples_leaf	1-4

E. Evaluation Phase

1) Evaluasi pada Data Asli

Evaluasi model *Random Forest* dilakukan pada data asli menggunakan dua pendekatan. Pertama, evaluasi langsung dilakukan pada data uji yang terpisah berdasarkan pembagian data 90:10, 80:20, dan 70:30. Kedua, evaluasi *K-Fold Cross Validation* dilakukan pada data latih untuk memastikan performa model konsisten dan tidak *overfitting*. Meskipun keduanya menggunakan data asli, pendekatan *K-Fold* bertujuan untuk mengukur performa model pada data latih melalui validasi silang. Hasil evaluasi dari kedua pendekatan ini ditunjukkan pada Tabel III dan Tabel IV berikut.

TABEL III
HASIL EVALUASI PADA DATA ASLI

Pembagian Data	Kelas	Akurasi	Presisi	*Recall*	*F1-Score*
90:10	1	0,78	0,42	0,29	0,35

	0		0,84	0,90	0,87
80:20	1	0,78	0,41	0,27	0,33
	0		0,83	0,91	0,87
70:30	1	0,78	0,41	0,28	0,34
	0		0,84	0,90	0,87

Hasil menunjukkan bahwa model memiliki performa yang lebih baik dalam mendeteksi Kelas 1 (*stunting*) dibandingkan Kelas 0 (tidak *stunting*), dengan *f1-score* untuk Kelas 1 lebih tinggi dari 0,86.

TABEL IV
HASIL EVALUASI DENGAN K-FOLD CROSS VALIDATION

Pembagian Data	Rata-Rata Akurasi	
	K=5	K=10
90:10	0,78	0,78
80:20	0,79	0,78
70:30	0,79	0,79

Hasil evaluasi menunjukkan bahwa performa model stabil dan tidak mengalami *overfitting* dengan akurasi berkisar antara 78%-79% pada semua pembagian data.

2) Evaluasi pada Data Resampling

Teknik SMOTE digunakan untuk menangani ketidakseimbangan kelas. Hasil evaluasi menggunakan data *resampling* ditunjukkan pada Tabel V berikut.

TABEL V
HASIL EVALUASI PADA DATA RESAMPLING

Pembagian Data	Kelas	Akurasi	Presisi	*Recall*	*F1-Score*
90:10	1	0,76	0,39	0,34	0,36
	0		0,84	0,87	0,85
80:20	1	0,77	0,40	0,33	0,36
	0		0,84	0,88	0,85
70:30	1	0,77	0,39	0,34	0,36
	0		0,85	0,87	0,86

Hasil evaluasi tersebut menunjukkan terdapat peningkatan *recall* pada kelas minoritas, sehingga akurasi keseluruhan mengalami sedikit penurunan akibat *trade-off* yang terjadi.

3) Evaluasi Setelah Tuning Hyperparameter

Proses *tuning hyperparameter* dilakukan menggunakan teknik *Random Search* untuk mendapatkan parameter terbaik. Hasil *tuning* pada pembagian 90:10 mencapai 85% di percobaan ke-5 sebagaimana terlihat pada Tabel VI berikut.

TABEL VI
HASIL TUNING HYPERPARAMETER PADA PEMBAGIAN 90:10

Percobaan	Parameter Terbaik	Akurasi
1	*n_estimators*: 200 *max_features*: log2 *max_depth*: 10 *min_samples_split*: 3 *min_samples_leaf*: 4	0,82
2	*n_estimators*: 200 *max_features*: sqrt *max_depth*: None *min_samples_split*: 5 *min_samples_leaf*: 4	0,82
3	*n_estimators*: 400 *max_features*: log2 *max_depth*: 10 *min_samples_split*: 5 *min_samples_leaf*: 4	0,83
4	*n_estimators*: 700 *max_features*: sqrt *max_depth*: 10 *min_samples_split*: 3 *min_samples_leaf*: 3	0,82
5	*n_estimators*: 200 *max_features:* log2 *max_depth*: 10 *min_samples_split*: 5 *min_samples_leaf*: 4	0,85

Selanjutnya, hasil *tuning* pada pembagian 80:20 menunjukkan dua percobaan menghasilkan akurasi tertinggi sebesar 0,83 yaitu pada uji coba ke-4 dan ke-5 sebagaimana terlihat pada Tabel VII berikut.

TABEL VII
HASIL TUNING HYPERPARAMETER PADA PEMBAGIAN 80:20

Percobaan	Parameter Terbaik	Akurasi
1	*n_estimators*: 500 *max_features*: sqrt *max_depth*: 10 *min_samples_split*: 4 *min_samples_leaf*: 4	0,82
2	*n_estimators*: 900 *max_features*: sqrt *max_depth*: 20 *min_samples_split*: 4 *min_samples_leaf*: 4	0,82
3	*n_estimators*: 800 *max_features*: log2 *max_depth*: 10 *min_samples_split*: 5 *min_samples_leaf*: 3	0,82
4	*n_estimators*: 700 *max_features*: sqrt *max_depth*: 10 *min_samples_split*: 4 *min_samples_leaf*: 4	0,83
5	*n_estimators*: 700 *max_features:* log2 *max_depth*: 10 *min_samples_split*: 4 *min_samples_leaf*: 3	0,83

Selanjutnya, hasil *tuning* pada pembagian 70:30 menujukkan akurasi tertinggi mencapai 0,84 pada percobaan ke-2 dan ke-5 yang ditunjukkan pada Tabel VIII berikut.

TABEL VIII
HASIL TUNING HYPERPARAMETER PADA PEMBAGIAN 70:30

Percobaan	Parameter Terbaik	Akurasi
1	*n_estimators*: 400 *max_features*: sqrt *max_depth*: 10 *min_samples_split*: 2	0,84

	min_samples_leaf: 4	
2	*n_estimators*: 400 *max_features*: log2 *max_depth*: 10 *min_samples_split*: 4 *min_samples_leaf*: 4	0,84
3	*n_estimators*: 600 *max_features*: sqrt *max_depth*: 50 *min_samples_split*: 4 *min_samples_leaf*: 4	0,83
4	*n_estimators*: 100 *max_features*: log2 *max_depth*: 5 *min_samples_split*: 4 *min_samples_leaf*: 4	0,84
5	*n_estimators*: 700 *max_features*: sqrt *max_depth*: 10 *min_samples_split*: 4 *min_samples_leaf*: 4	0,82

Model terbaik dipilih berdasarkan hasil *tuning hyperparameter* pada pembagian data 90:10. Model ini memiliki kemampuan yang baik dalam mendeteksi *stunting* dengan *recall* dan *f1-score* yang lebih tinggi pada kelas mayoritas. Selanjutnya, model disimpan menggunakan format *pickle* untuk integrasi ke dalam sistem web berbasis *Flask*, yang memungkinkan deteksi *stunting* secara *real-time*. Proses penyimpanan model ditunjukkan pada Gbr. 18 berikut.

```
# Menyimpan model dalam format pickle
import pickle

pickle.dump(best_model_90, open('best_model_90.pkl', 'wb'))
```

Gbr. 18 Menyimpan Model dengan Format *Pickle*

F. *Deployment Phase*

1) *Analisis Kebutuhan*

Pada tahap pengembangan aplikasi web deteksi *stunting*, kebutuhan perangkat terdiri dari perangkat keras (*hardware*) dan perangkat lunak (*software*) yang dibutuhkan untuk memastikan sistem mampu beroperasi secara optimal. Berikut adalah rincian kebutuhan perangkat untuk pengembangan web:

a. Perangkat Keras (*Hardware*)
- Prosesor Intel(R) Core(TM) i3-1005G1 dengan kecepatan 1,20 GHz (1,19 GHz).
- Memori RAM sebesar 4,00 GB.
- SSD sebesar 20GB.

b. Perangkat Lunak (*Software*)
- Sistem Operasi Windows 11 64-bit
- *Jupyter Notebook*
- *Visual Studio Code*
- *Google Chrome*

2) *Perancangan*

Tahap perancangan aplikasi web deteksi *stunting* mencakup tiga langkah utama yaitu instalasi *Flask* dan pustaka *Python*, pembuatan *template*, dan pengembangan logika aplikasi. Instalasi pustaka *Python* dilakukan menggunakan perintah *pip install* untuk *Flask, scikit-learn, pandas,* dan *numpy*. *Template* HTML dirancang untuk antarmuka pengguna, mencakup halaman utama, formulir deteksi individu, dan unggahan *file* CSV untuk deteksi kelompok. *Template* ini dibuat menggunakan HTML dan CSS untuk struktur dan desain yang *responsive*. Hasil perancangan antarmuka pengguna ditunjukkan pada Gbr. 19 hingga Gbr. 21.

Gbr. 19 Halaman Utama

Gbr. 20 Halaman Deteksi Individu

Gbr. 21 Halaman Deteksi Kelompok

Selanjutnya, logika aplikasi dikembangkan di *app.py*, mencakup integrasi model deteksi, penanganan *input* data, dan pengunduhan hasil deteksi. Model deteksi yang telah dilatih sebelumnya dimuat menggunakan *pickle*, sementara *Flask* digunakan untuk mengatur *routing*, memproses data, dan mengirim hasil kepada pengguna. Fitur keamanan seperti SECRET_KEY dikelola melalui variabel lingkungan untuk melindungi data sensitif. Data hasil deteksi disimpan dalam memori menggunakan *BytesIO* untuk efisiensi dan keamanan, memungkinkan pengguna mengunduh *file* tanpa menyimpan sementara di server.

3) *Pengujian*

Dalam penelitian ini, pengujian dilakukan menggunakan metode *blackbox testing*, di mana fokusnya adalah pada *input* dan *output* aplikasi tanpa memeriksa kode internalnya. Tabel IX di bawah ini mencakup berbagai skenario pengujian yang

diperlukan untuk memastikan aplikasi web berfungsi dengan baik sebelum dilakukan *deployment*.

TABEL IX
SKENARIO PENGUJIAN APLIKASI

No.	Skenario Pengujian	*Test Case*	Ekspektasi	Hasil
1	Form deteksi individu tidak diisi	Data *form*: (kosong)	Tombol deteksi tidak dapat diklik atau sistem memberikan pesan untuk mengisi data	Berhasil: sistem memberikan pesan peringatan untuk mengisi data yang diperlukan
2	Data tidak valid pada *form* individu	Isi *form* numerik dengan teks	Sistem menolak *input* dan memberikan pesan kesalahan yang jelas	Berhasil: sistem memberikan pesan peringatan untuk memasukkan *input* sesuai format
3	Data valid pada *form* individu	Jenis kelamin: Laki-laki, Umur: 16, Berat bayi: 2.9, Panjang bayi: 49, Berat badan: 8.5, Tinggi badan: 72.2	Sistem memproses deteksi dan menampilkan hasil yang sesuai (*Stunting* atau Tidak)	Berhasil: sistem menampilkan hasil deteksi sesuai data *input* yaitu 'Anak Mengalami *Stunting*'
4	Unggah *file* CSV tanpa data	*File*: (kosong)	Sistem menolak *file* dengan pesan kesalahan atau tidak memproses *file* kosong	Berhasil: sistem tidak memproses *file* dan menampilkan pesan kesalahan '*File* CSV kosong atau tidak valid. Mohon unggah *file* yang benar'
5	Unggah *file* CSV berisi *header* tanpa data	*File:* Hanya berisi *header* tanpa berisi data	Sistem menolak *file* dengan menampilkan pesan kesalahan	Berhasil: sistem tidak memproses *file* dan menampilkan pesan kesalahan '*File* CSV kosong. Mohon unggah *file* yang berisi data'
6	Unggah *file* CSV dengan format salah	*File* CSV: Nama kolom tidak sesuai	Sistem menolak *file* dan memberikan pesan bahwa format CSV tidak sesuai dengan ketentuan	Berhasil: sistem menampilkan pesan 'Format CSV tidak sesuai. Harus mengandung kolom: Jenis Kelamin, Umur, Berat Bayi, Panjang Bayi, Berat Badan, Tinggi Badan
7	Unggah *file* CSV dengan data yang hilang	*File* CSV: atribut umur pada indeks pertama kosong	Sistem menolak *file* dan menampilkan pesan kesalahan	Berhasil: sistem menampilkan pesan kesalahan 'Terdapat 1 nilai yang hilang dalam *file* CSV. Mohon lengkapi data sebelum diunggah'
8	Unggah *file* CSV dengan data valid	*File* CSV: Data sesuai dengan format yang diharapkan dan semua kolom terisi dengan benar	Sistem memproses *file* dan menampilkan hasil deteksi untuk setiap baris data dalam bentuk *file* CSV	Berhasil: sistem memproses *file* dan menampilkan hasil deteksi dalam bentuk *file* CSV yang dapat diunduh
9	Navigasi antar halaman	Pengguna meng-klik tautan navigasi ke halaman '*Home*', 'Individu', 'Kelompok'	Sistem berpindah halaman sesuai tautan navigasi tanpa adanya kesalahan	Berhasil: sistem berpindah halaman sesuai tautan navigasi tanpa kesalahan

4) Deploy ke Railway

Tahap akhir pengembangan aplikasi adalah *deployment* ke platform *Railway*, yang memudahkan pengelolaan aplikasi tanpa perlu infrastruktur kompleks. Proses ini dimulai dengan membuat *file* requirements.txt untuk mencatat semua pustaka yang digunakan menggunakan perintah *pip freeze > requrements.txt*. Selanjutnya, dibuat *file Procfile* yang berisi intruksi untuk menjalankan aplikasi menggunakan *gunicorn*, seperti *web: gunicorn app:app*. Kode kemudian diunggah ke *GitHub* melalui proses inisialisasi repositori, penambahan *file*, dan *push* ke repositori *GitHub*. *Railway* dihubungkan dengan repositori tersebut untuk mendeteksi perubahan dan melakukan *update* otomatis pada aplikasi. Sebelum *deployment*, konfigurasi variabel lingkungan seperti SECRET_KEY dilakukan melalui fitur *Environment Variables* di *Railway* untuk memastikan aplikasi berjalan dengan aman dan lancar.

IV. KESIMPULAN

Penelitian ini berhasil membangun model deteksi *stunting* pada balita berbasis web menggunakan metode *Random Forest*, dengan model terbaik diperoleh pada pembagian data 90:10 setelah *tuning hyperparameter*, menghasilkan akurasi sebesar 85%. Evaluasi model juga menunjukkan performa yang stabil, sebagaimana terlihat dari hasil *K-Fold Cross Validation* yang konsisten, mengindikasikan bahwa model tidak mengalami *overfitting* dan mampu memberikan prediksi yang andal pada data baru. Selain itu, aplikasi web yang dikembangkan menyediakan dua opsi deteksi, yaitu deteksi individu dan deteksi kelompok melalui unggahan *file* CSV. Aplikasi ini telah diimplementasikan di *Railway* untuk mempermudah pengelolaan dan pembaruan secara otomatis, sehingga berpotensi mendukung upaya deteksi dini *stunting* pada balita dengan lebih efektif dan efisien.

V. SARAN

Untuk pengembangan lebih lanjut, disarankan untuk mengeksplorasi factor-faktor yang berkontribusi terhadap *stunting*, seperti kondisi social ekonomi dan pola makan, guna meningkatkan akurasi deteksi. Selain itu, pengembangan aplikasi dapat ditingkatkan dengan menambahkan fitur visualisasi data dan analisis tren *stunting* di wilayah tertentu untuk memberikan informasi yang lebih komprehensif kepada pengguna. Pengujian dengan *dataset* yang lebih besar dan beragam juga perlu dilakukan untuk memastikan keandalan model dalam berbagai kondisi populasi. Selanjutnya, penerapan teknik *machine learning* lainnya dapat dipertimbangkan untuk membandingkan performa dengan metode yang diterapkan dalam penelitian ini, sehingga memungkinkan untuk memberikan wawasan lebih luas terkait pendekatan terbaik untuk deteksi *stunting*.

REFERENSI

[1] A. Candra, *Pencegahan dan Penanggulangan Stunting*. 2020.

[2] M. Amin, “Cegah Stunting, Wapres Minta Keluarga Indonesia Prioritaskan Kebutuhan Gizi Anak dan Sanitasi,” KEMENTERIAN SEKRETARIAT NEGARA RI SEKRETARIAT WAKIL PRESIDEN. Accessed: May 05, 2024. [Online]. Available: https://stunting.go.id/cegah-stunting-wapres-minta-keluarga-indonesia-prioritaskan-kebutuhan-gizi-anak-dan-sanitasi/

[3] N. S. Tarmizi, “Prevalensi Stunting di Indonesia Turun ke 21,6% dari 24,4%,” Sehat Negeriku. Accessed: Apr. 02, 2024. [Online]. Available: https://sehatnegeriku.kemkes.go.id/baca/rilis-media/20230125/3142280/prevalensi-stunting-di-indonesia-turun-ke-216-dari-244/

[4] J. J. P. Jansen, C. Heavey, T. J. M. Mom, Z. Simsek, and S. A. Zahra, “Scaling-up: Building, Leading and Sustaining Rapid Growth Over Time,” *J. Manag. Stud.*, vol. 60, no. 3, pp. 581–604, 2023, doi: 10.1111/joms.12910.

[5] S. Aminizadeh, A. Heidari, S. Toumaj, and M. Darbandi, “The Applications of Machine Learning Techniques in Medical Data Processing based on Distributed Computing and the Internet of Things,” *Comput. Methods Programs Biomed.*, 2023, doi: 10.1016/j.cmpb.2023.107745.

[6] P. T. R, “A Comparative Study on Decision Tree and Random Forest Using R Tool,” *Ijarcce*, vol. 4, no. 1, pp. 196–199, 2015, doi: 10.17148/ijarcce.2015.4142.

[7] N. Benediktus and R. S. Oetama, “Algoritma Klasifikasi Decision Tree C5.0 untuk Memprediksi Performa Akademik Siswa Natanael,” *Ultim. J. Tek. Inform.*, vol. 12, no. 1, pp. 14–19, 2020.

[8] L. Breiman, J. Friedman, R. . Olshen, and C. . Stone, *Classification and Regression Trees*. New York, 1984. doi: https://doi.org/10.1201/9781315139470.

[9] L. Breiman, “Random Forests,” in *Machine Learning*, 2001, pp. 5–32. doi: 10.1023/A:1010933404324.

[10] S. A. Kamila, R. R. S. Sulistijowati, I. Susanto, F. Matematika, P. Alam, and U. S. Maret, “Klasifikasi Penyakit Jantung Menggunakan Decision Tree dan Random Forest,” vol. 2, pp. 7–12, 2023.

[11] R. Irsyad, “Penggunaan Python Web Framework Flask Untuk Pemula,” 2018.

[12] A. Azevedo and M. F. Santos, “KDD, semma and CRISP-DM: A parallel overview,” *MCCSIS’08 - IADIS Multi Conf. Comput. Sci. Inf. Syst. Proc. Informatics 2008 Data Min. 2008*, no. January 2008, pp. 182–185, 2008.