BAB 3
METODOLOGI PENELITIAN

3.1 Tahapan Penelitian

Tahapan penelitian ini disusun secara sistematis guna memastikan bahwa proses penelitian dilaksanakan dengan terarah dan sesuai dengan tujuan yang telah ditetapkan. Berikut adalah tahapan yang dilakukan, dimulai dari studi literatur hingga penulisan laporan dan konsultasi.

1. Studi Literatur

   Studi literatur merupakan tahap awal dalam penyusunan laporan penelitian ini, dengan tujuan menggali informasi yang relevan dengan fokus penelitian. Informasi yang dibutuhkan diperoleh melalui pembacaan dan pemahaman berbagai materi dari sumber terpercaya, seperti jurnal ilmiah, artikel, buku, dan sumber lainnya.

2. Pengumpulan Data

   Dataset yang digunakan dalam penelitian ini merupakan data balita dari Kecamatan Mauk yang diperoleh dari Dinas Kesehatan Kabupaten Tangerang, dalam format berkas `.xls`. *Dataset* ini, bertanggal 28 November 2024, berisi 24 kolom dan mencakup 6.592 sampel data, termasuk informasi seperti identitas pribadi, identitas orang tua, informasi lokasi, data pengukuran, dan indeks pertumbuhan.

3. *Data Preprocessing*

   Tahap *data preprocessing* dilakukan untuk menyiapkan data sebelum digunakan dalam model. Dalam proses ini, *dataset* dibagi dengan proporsi 90% untuk pelatihan dan pengujian, serta 10% untuk validasi. Proses ini mencakup konversi format berkas dari `.xls` ke `.csv`, penghapusan kolom yang tidak relevan, dan penanganan nilai *null* guna menjaga konsistensi data. Selain itu, data bertipe tanggal dikonversi ke format numerik, sementara atribut kategori dikodekan menggunakan *Label Encoding* agar dapat diproses oleh algoritma pembelajaran mesin.

4. Pembangunan Model

Tahap ini bertujuan untuk mengembangkan dan mengoptimalkan model *Random Forest* dalam klasifikasi *stunting*. Proses dimulai dengan membagi data menggunakan metode *holdout* (70:30), diikuti dengan pencarian *hyperparameter* optimal menggunakan *Grid Search*. Selanjutnya, dilakukan *feature selection* dengan ANOVA *F-value* untuk memilih fitur terbaik. Model dilatih dalam empat skenario berbeda guna mengevaluasi pengaruh teknik penyeimbangan data (SMOTE) dan penyetelan *hyperparameter*.Selanjutnya, dilakukan evaluasi untuk membandingkan performa berbagai model dengan menggunakan metrik *Accuracy*, *Precision*, *Recall*, dan *F1-Score*. Tujuan dari evaluasi ini adalah untuk menentukan model terbaik yang akan diimplementasikan dalam program.

5. Evaluasi Kinerja Model

   Tahap evaluasi dilakukan terhadap model terbaik dengan menggunakan *Multi-class Confusion Matrix* untuk menganalisis distribusi prediksi model terhadap kelas yang sebenarnya. Selain itu, metrik evaluasi seperti *Accuracy*, *Precision*, *Recall*, dan *F1-Score* dihitung berdasarkan informasi yang diperoleh dari *Confusion Matrix* untuk memperoleh gambaran menyeluruh mengenai performa model.

6. Perancangan dan Implementasi Program

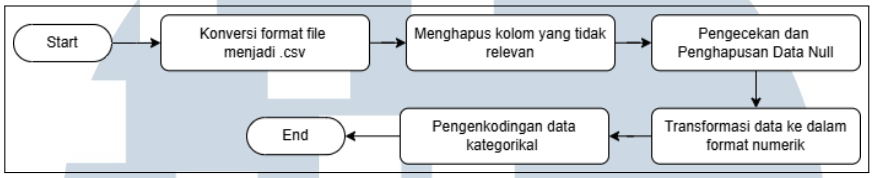
   Tahap perancangan program dilakukan untuk merancang alur kerja dan struktur program sebelum implementasi. Proses ini mencakup pembuatan *flowchart* untuk menggambarkan alur program serta perancangan *wireframe* sebagai representasi tampilan antarmuka pengguna. Setelah tahap perancangan selesai, langkah berikutnya adalah implementasi, di mana desain dan alur kerja yang telah dibuat diterjemahkan ke dalam kode program. Implementasi mencakup pengembangan fungsionalitas utama dan penerapan antarmuka pengguna sesuai *wireframe*.

7. Penulisan Laporan dan Konsultasi

   Pada tahap ini, dilakukan penulisan laporan untuk mendokumentasikan hasil penelitian. Laporan ini bertujuan untuk menyediakan informasi yang bermanfaat bagi penelitian serupa di masa mendatang. Selain itu, konsultasi dengan dosen pembimbing dilakukan secara rutin guna memastikan penelitian berjalan terarah dan menghasilkan hasil yang sesuai dengan tujuan yang telah ditetapkan.

3.2 Data Preprocessing

Gambar 3.1 merepresentasikan tahapan dalam *data preprocessing*. Tahap ini bertujuan untuk mempersiapkan data sebelum digunakan dalam pembangunan model.



Gambar 3.1. *Flowchart Data Preprocessing*

Langkah pertama dalam proses ini adalah mengonversi format *file* dari `.xls` menjadi `.csv` agar lebih mudah diolah. Selanjutnya, dilakukan penghapusan kolom yang tidak relevan, seperti identitas pribadi, identitas orang tua, dan informasi lokasi. Penghapusan ini bertujuan untuk menyederhanakan *dataset* serta mengurangi potensi *noise* yang dapat memengaruhi kinerja model, serta untuk menjaga privasi data. Setelah proses ini, jumlah kolom dalam *dataset* berkurang dari 24 menjadi 11. Daftar kolom yang terdapat pada *dataset* disajikan pada Tabel 3.1.

Tabel 3.1. Daftar Kolom yang digunakan pada *Dataset*

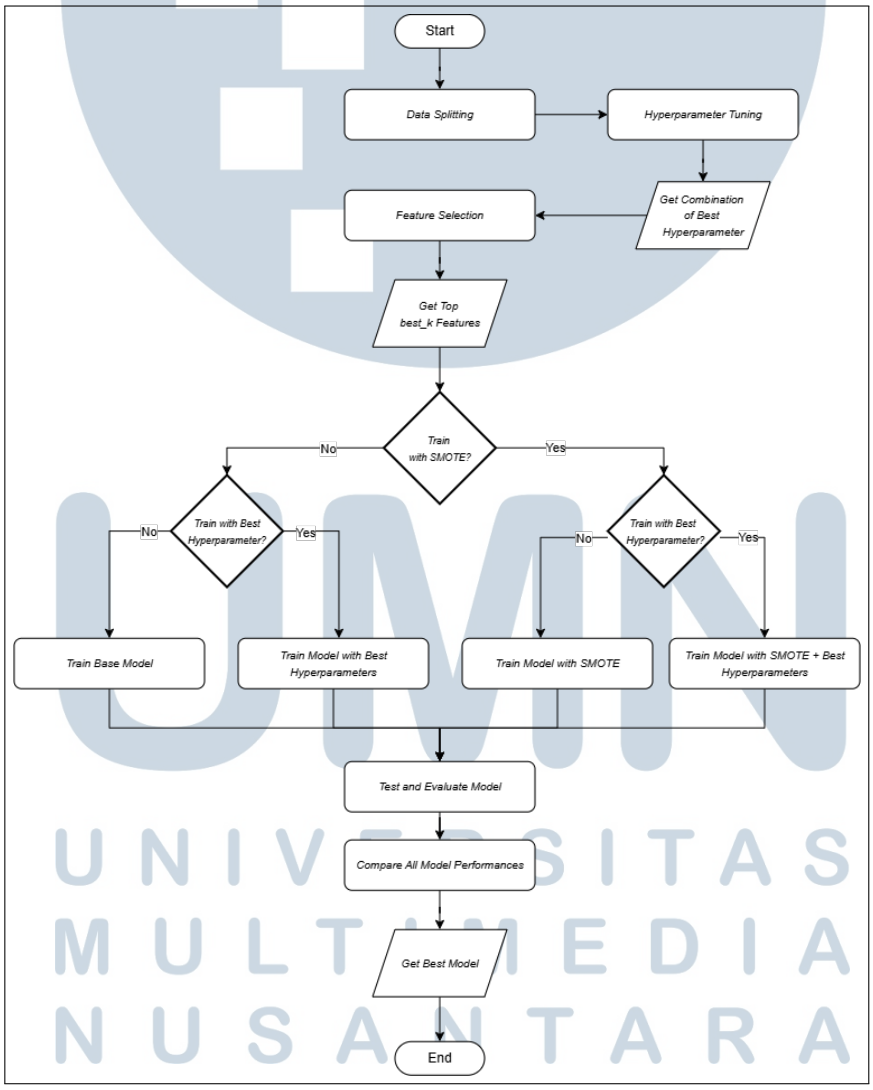
Nama Kolom	**Tipe Data**
JK	Categorical
BB Lahir	Float
TB Lahir	Float
Desa/Kel	Categorical
Usia Saat Ukur	String
Berat	Float
Tinggi	Float
Cara Ukur	Categorical
BB/U	Categorical
TB/U	Categorical
BB/TB	Categorical

Proses berikutnya adalah pemeriksaan dan penghapusan nilai *null*. Langkah ini dilakukan untuk memastikan konsistensi data dan mencegah bias dalam analisis, sehingga dapat meningkatkan performa model. Setelah itu, data dalam format

”Tahun - Bulan - Hari” ditransformasikan ke dalam format numerik agar kompatibel dengan algoritma pembelajaran mesin. Selanjutnya, dilakukan proses *categorical encoding* menggunakan metode *Label Encoding* agar data non-numerik dapat dipahami oleh algoritma pembelajaran mesin.

3.3 Pembangunan Model

Gambar 3.2 menggambarkan tahapan dalam pembangunan model *Random Forest*. Tahap ini bertujuan untuk mendapatkan model terbaik untuk digunakan pada program.



Gambar 3.2. *Flowchart* Pembangunan Model *Random Forest*

Proses dimulai dengan *Data Splitting* menggunakan metode *holdout* dengan

rasio 70:30, dimana 70% data digunakan untuk pelatihan model (*training set*) dan 30% sisanya digunakan untuk mengevaluasi kinerja model (*testing set*). Selanjutnya, dilakukan *Hyperparameter Tuning* menggunakan metode *Grid Search* untuk mengoptimalkan kombinasi *hyperparameter* dan meningkatkan akurasi model.

Proses berikutnya adalah *Feature Selection* untuk memilih fitur terbaik menggunakan metode ANOVA *F-value* dengan bantuan dari fungsi `SelectKBest()`, menghasilkan *Top best*$_k$ *Features*. Fitur terpilih digunakan untuk meningkatkan akurasi dan mengurangi kompleksitas model. Proses pelatihan model dilakukan dalam empat skenario: (1) *Train Base Model*, (2) *Train Model with SMOTE*, (3) *Train Model with SMOTE + Best Hyperparameters*, dan (4) *Train Model with Best Hyperparameters*.

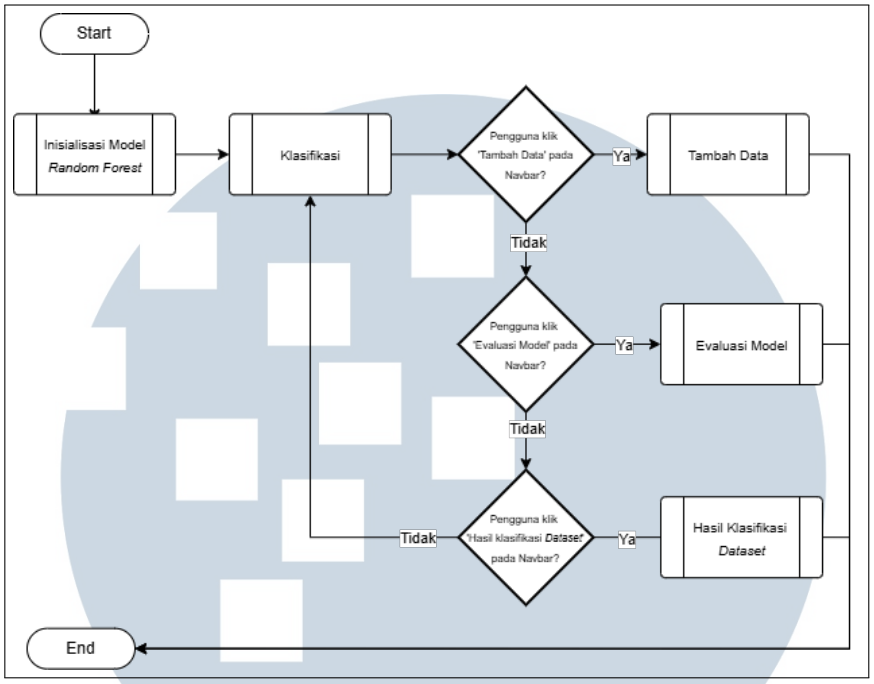
Setelah proses pelatihan, model dievaluasi menggunakan data pengujian dengan metrik *Accuracy*, *Precision*, *Recall*, dan *F1-Score*. Hasil evaluasi dari setiap model dibandingkan pada tahap *Compare All Model Performances* untuk menentukan model terbaik. Model dengan performa tertinggi dipilih sebagai *Best Model* dan dijadikan model klasifikasi dalam penelitian ini. Selanjutnya, model terpilih dievaluasi lebih lanjut menggunakan beberapa metrik tambahan dan diimplementasikan ke dalam sistem klasifikasi.

3.4 Perancangan Program

Pada proses ini, dilakukan perancangan program untuk memudahkan visualisasi proses dan hasil klasifikasi stunting pada balita. Tahapan dimulai dengan pembuatan *flowchart* untuk memetakan alur program secara keseluruhan, diikuti penyusunan *wireframe* sebagai rancangan awal tampilan antarmuka. Selanjutnya, program diimplementasikan berdasarkan *flowchart* dan *wireframe* yang telah disusun guna membantu pengguna dalam memahami informasi yang dihasilkan dari model klasifikasi *stunting*.

3.4.1 Flowchart

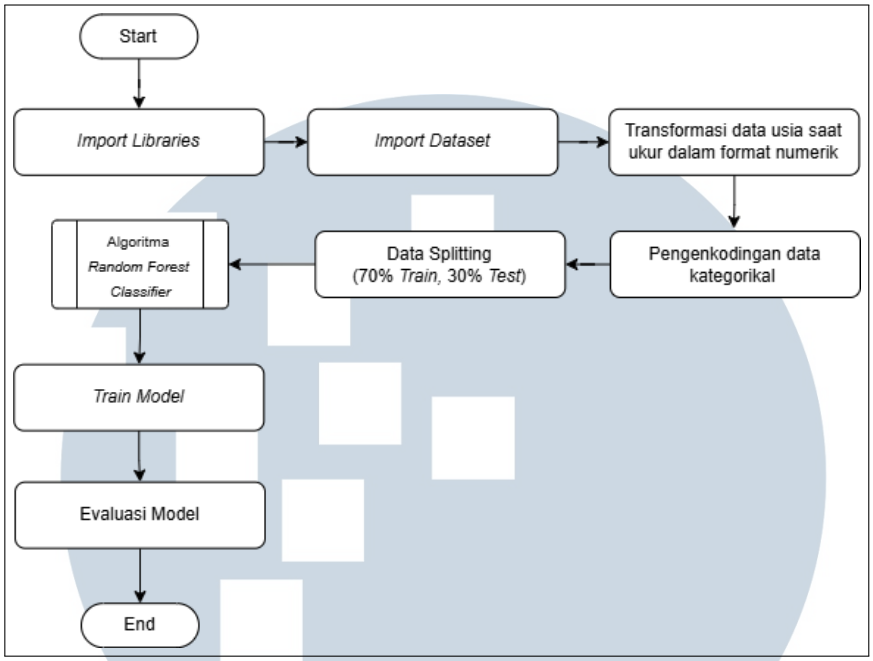
Flowchart digunakan untuk memvisualisasikan alur kerja program secara terstruktur, sehingga memudahkan pemahaman proses dan tahapan yang dilakukan serta menunjukkan cara pengguna berinteraksi dengan berbagai menu yang tersedia pada program.



Gambar 3.3. *Flowchart* Alur Program

Flowchart pada Gambar 3.3 menunjukkan alur program klasifikasi *stunting* pada balita menggunakan algoritma *Random Forest*. Proses dimulai dengan inisialisasi model, diikuti oleh opsi bagi pengguna untuk mengakses menu melalui *navbar*. Setelah inisialisasi model dilakukan, pengguna bisa melakukan klasifikasi. Pengguna juga dapat menambahkan data baru melalui menu ”Tambah Data”, mengevaluasi performa model menggunakan metrik seperti *Accuracy*, *Precision*, *Recall*, dan *F1-score* melalui menu ”Evaluasi Model”, atau melihat hasil klasifikasi berdasarkan *dataset* pada menu ”Hasil Klasifikasi *Dataset*”.

Selanjutnya, terdapat *flowchart* modul inisialisasi model *Random Forest* pada Gambar 3.4. Proses dimulai dengan mengimpor *libraries* yang diperlukan untuk *preprocessing* dan pembuatan model. Selanjutnya, *dataset* diimpor untuk digunakan dalam proses pelatihan dan pengujian. Tahap *preprocessing* melibatkan transformasi data usia ke dalam format numerik serta pengkodean variabel kategorikal agar kompatibel dengan algoritma *Random Forest*. Setelah tahap *preprocessing* selesai, *dataset* dipisahkan menjadi data pelatihan dan data pengujian dengan komposisi 70% untuk pelatihan dan 30% untuk pengujian. Kemudian dilakukan pelatihan model menggunakan data latih. Tahap akhir adalah evaluasi menggunakan data uji untuk menilai performa model.

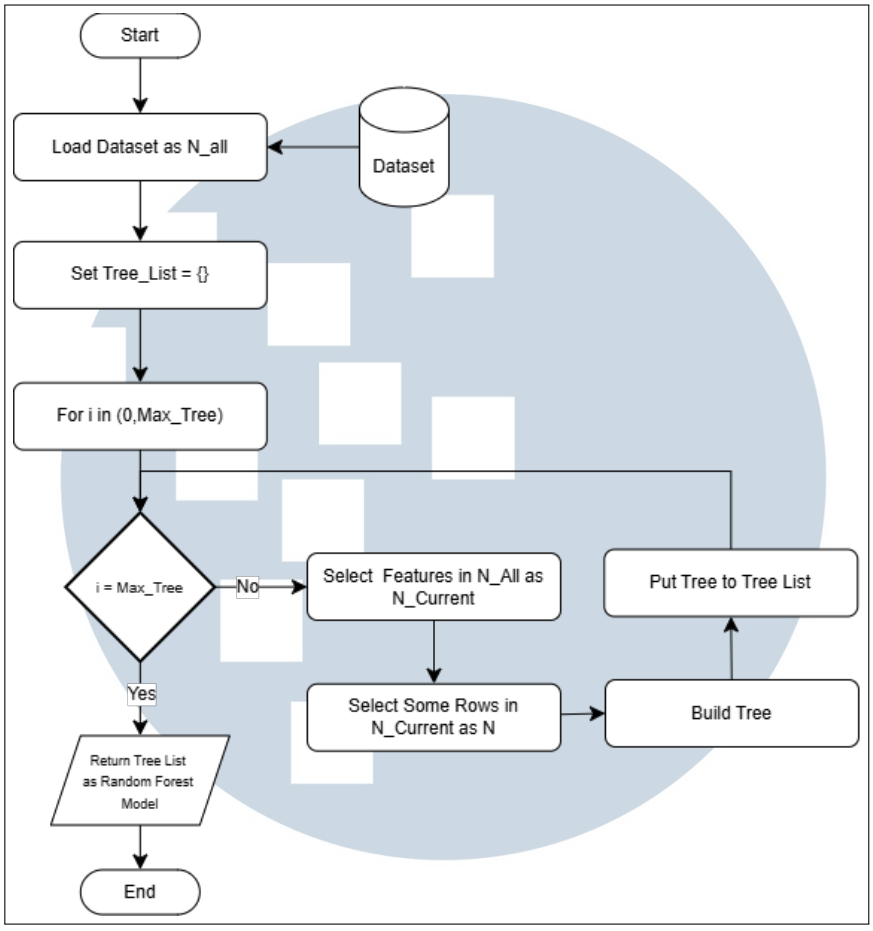


Gambar 3.4. *Flowchart* Modul Inisialisasi Model *Random Forest*

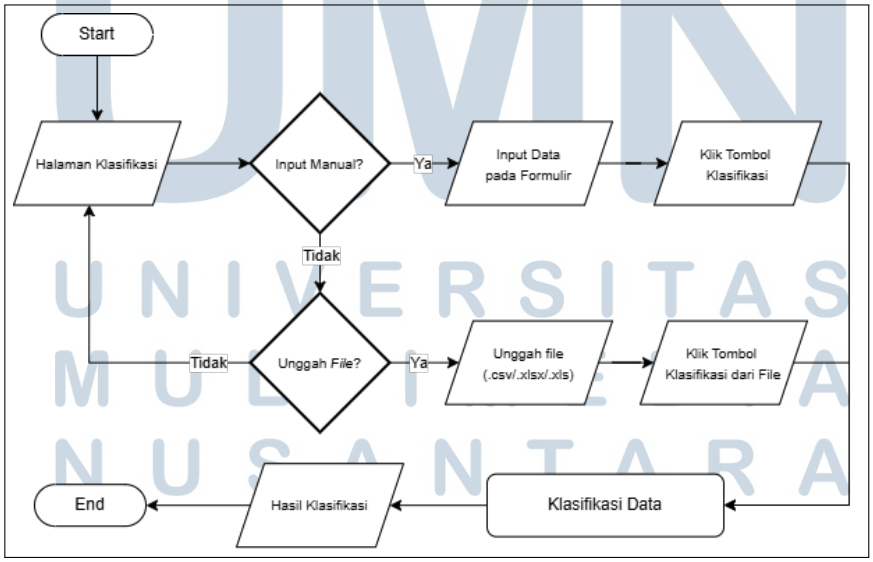
Pada program ini, algoritma *Random Forest* digunakan untuk membentuk model klasifikasi *stunting* pada balita ketika *Train Model*. *Flowchart* algoritma *Random Forest Classifier* digambarkan pada Gambar 3.5. Proses pembentukan model klasifikasi menggunakan algoritma *Random Forest* diawali dengan memuat *dataset* sebagai N_{all} dan menginisialisasi *Tree List* sebagai daftar kosong. Model kemudian dibangun melalui iterasi sebanyak *Max_Tree*, dengan setiap iterasi mencakup tahapan berikut:

1. Memilih sejumlah fitur secara acak dari N_{all} untuk membentuk *subse*t fitur N_{current}.
2. Mengambil sampel *dataset* secara acak dari N_{current} menggunakan teknik *bootstrap sampling*.
3. Membangun pohon keputusan berdasarkan *subset* fitur dan *dataset* yang telah dipilih.
4. Menyimpan pohon keputusan yang telah terbentuk ke dalam *Tree List*.

Proses ini berulang hingga jumlah pohon keputusan mencapai *Max_Tree*. Setelah seluruh pohon terbentuk, kumpulan *Tree List* dikembalikan sebagai model *Random Forest* yang kemudian digunakan untuk klasifikasi *stunting* pada balita.

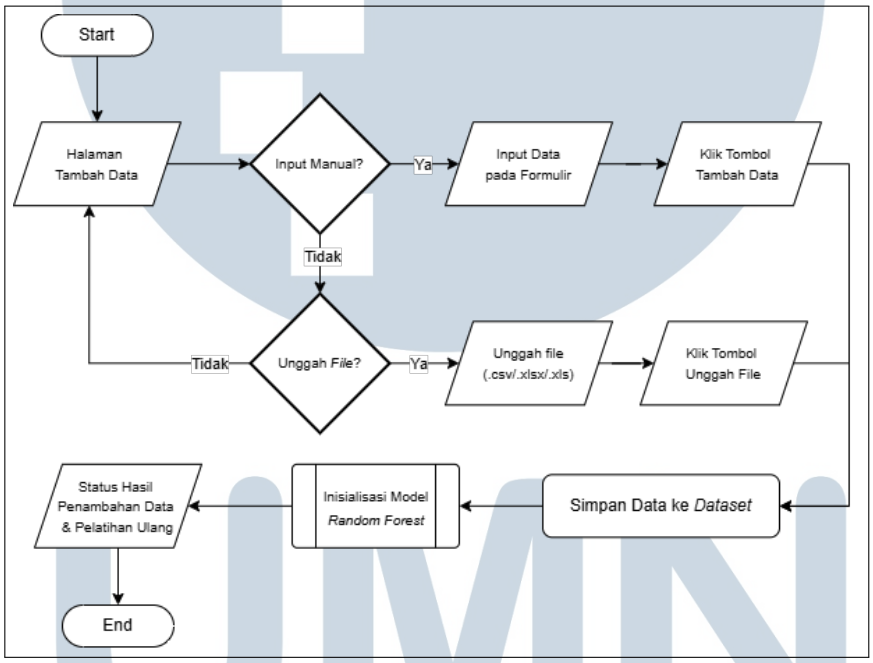


Gambar 3.5. *Flowchart* Modul Algoritma *Random Forest Classifier*



Gambar 3.6. *Flowchart* Modul Klasifikasi

Flowchart pada Gambar 3.6 menggambarkan alur kerja modul klasifikasi dalam program. Proses dimulai dari halaman klasifikasi, dimana pengguna diberikan dua opsi untuk memasukkan data, yaitu input manual atau unggah *file*. Jika pengguna memilih input manual, data dimasukkan melalui formulir yang tersedia dan proses klasifikasi dimulai setelah pengguna memilih tombol ”klasifikasi”. Jika pengguna tidak memilih input manual, sistem menawarkan opsi untuk mengunggah *file* dalam format `.csv`, `.xlsx`, atau `.xls`. Setelah *file* diunggah, pengguna dapat memulai klasifikasi dengan memilih tombol ”klasifikasi dari *file*”. Seluruh data yang diproses, baik dari input manual maupun *file*, kemudian diklasifikasikan, dan hasilnya ditampilkan kepada pengguna.

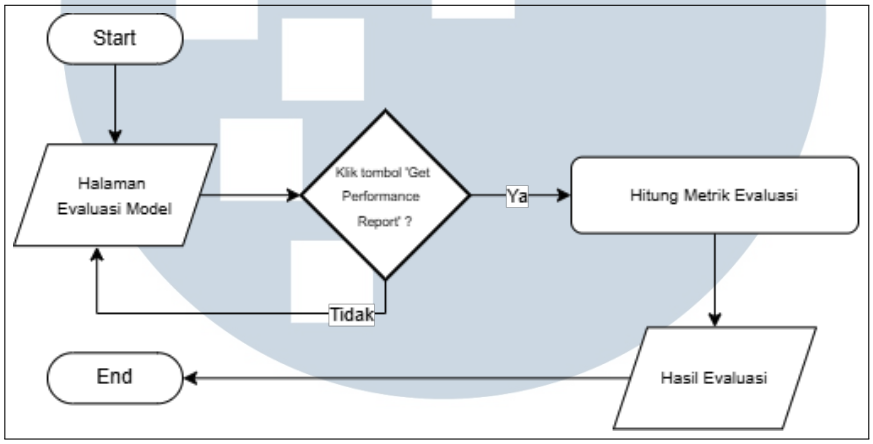


Gambar 3.7. *Flowchart* Modul Tambah Data

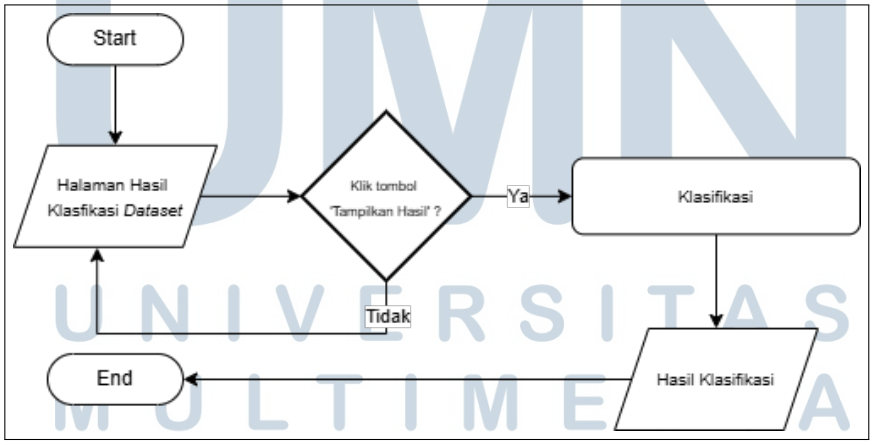
Flowchart pada Gambar 3.7 menggambarkan alur kerja modul tambah data dalam program. Proses dimulai dari halaman tambah data, dimana pengguna diberikan dua opsi untuk menambahkan data, yaitu input manual atau unggah *file*. Jika pengguna memilih input manual, data dimasukkan melalui formulir yang disediakan dan kemudian memilih tombol ”tambah data” untuk menyimpan entri tersebut. Jika pengguna tidak memilih input manual, sistem akan memeriksa apakah pengguna ingin mengunggah *file*. Apabila opsi unggah *file* dipilih, pengguna dapat mengunggah *file* dalam format `.csv`, `.xlsx`, atau `.xls`, kemudian memilih tombol ”unggah *file*” untuk memprosesnya. Setelah data berhasil ditambahkan melalui

salah satu metode, sistem akan menyimpan data ke dalam *dataset* dan memperbarui model *Random Forest* dengan data terbaru. Proses diakhiri dengan menampilkan pesan status hasil penambahan data dan pelatihan ulang model.

Flowchart pada Gambar 3.8 menggambarkan alur kerja untuk melakukan evaluasi model dalam program. Proses dimulai dari halaman evaluasi model, dimana pengguna dapat memilih untuk menghitung metrik evaluasi dengan memilih tombol "*Get Performance Report*". Jika pengguna memilih tombol tersebut, sistem akan menghitung metrik evaluasi seperti *Confusion Matrix*, *Accuracy*, *Precision*, *Recall*, *F1-score*, dan *Log Loss*, lalu menampilkan hasil evaluasi kepada pengguna.



Gambar 3.8. *Flowchart* Modul Evaluasi Model



Gambar 3.9. *Flowchart* Modul Hasil Klasifikasi *Dataset*

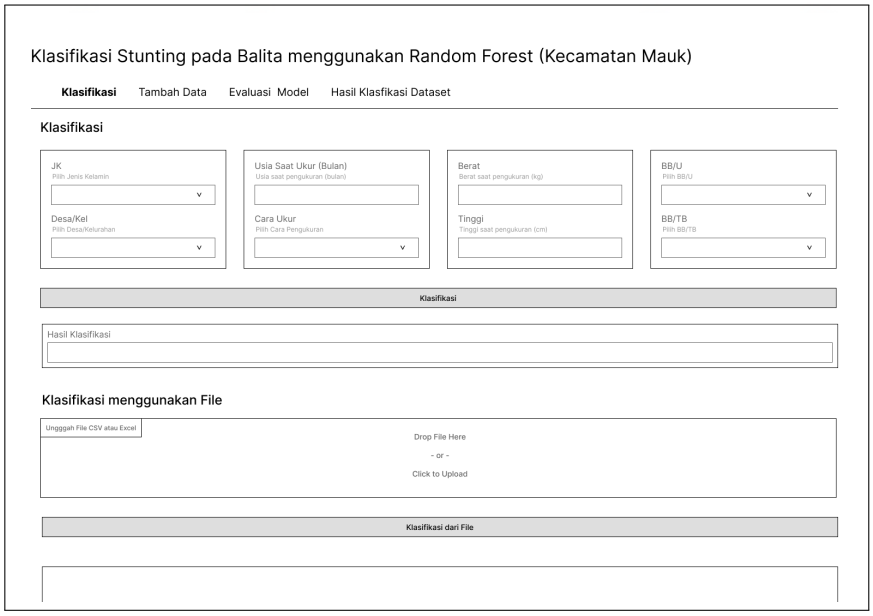
Flowchart pada Gambar 3.9 menggambarkan alur kerja untuk menampilkan hasil klasifikasi *dataset* dalam program. Proses dimulai dari halaman hasil klasifikasi *dataset*, dimana pengguna dapat memilih untuk menampilkan hasil

klasifikasi dengan memilih tombol ”Tampilkan Hasil”. Jika pengguna memilih tombol tersebut, sistem akan memproses data melalui model klasifikasi dan menghasilkan *output* berupa hasil klasifikasi dalam bentuk diagram lingkaran. Hasil ini kemudian ditampilkan kepada pengguna.

3.4.2 Wireframe

Wireframe merupakan representasi visual sederhana dari tampilan antarmuka pengguna yang akan dibangun. Umumnya, *wireframe* dibuat menggunakan garis, kotak, dan elemen dasar lainnya. Terdapat 4 *wireframe* yang digunakan, yaitu halaman klasifikasi, tambah data, evaluasi model, dan hasil klasifikasi *dataset*.

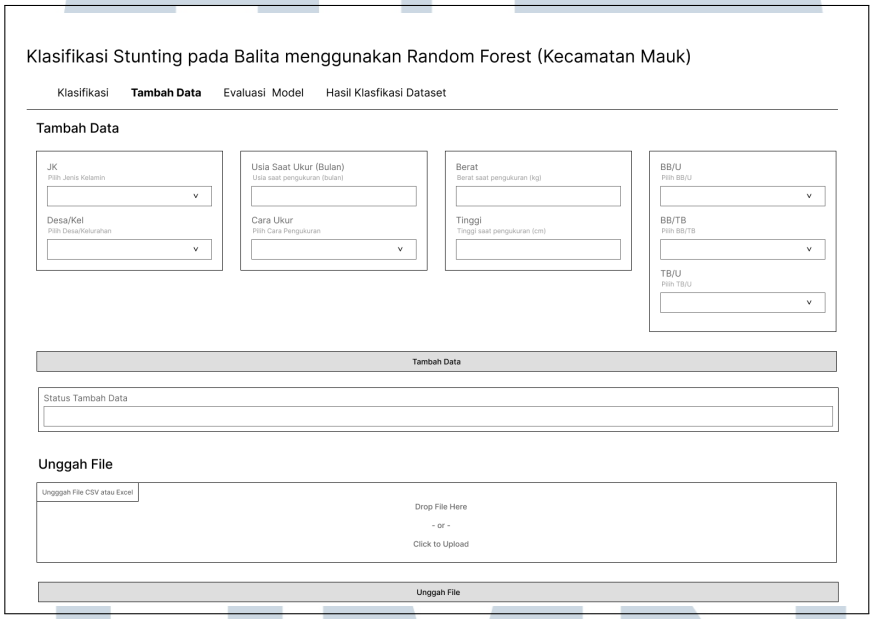
Gambar 3.10 menunjukkan *wireframe* halaman klasifikasi. Pada halaman ini, pengguna dapat memasukkan data balita secara manual melalui formulir yang tersedia, seperti jenis kelamin, desa/kelurahan, cara ukur, usia saat ukur, berat badan, tinggi badan, BB/U, dan BB/TB. Selain input manual, pengguna juga dapat mengunggah *file* berformat `.csv`, `.xlsx`, atau `.xls` untuk melakukan klasifikasi data secara massal. Hasil klasifikasi akan langsung ditampilkan setelah data diproses. Untuk klasifikasi massal, hasilnya disajikan dalam bentuk diagram lingkaran.



Gambar 3.10. *Wireframe* Halaman Klasifikasi

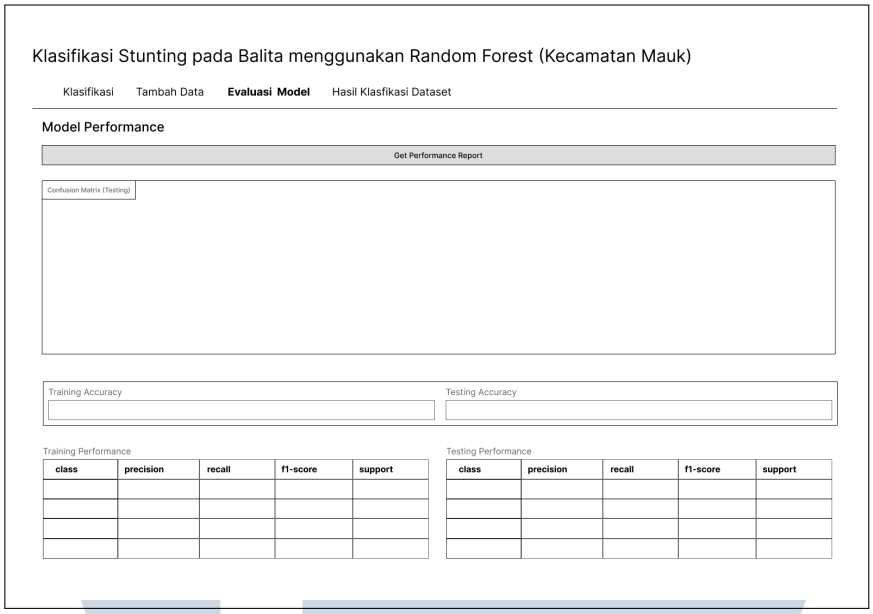
Gambar 3.11 memperlihatkan *wireframe* untuk halaman tambah data.

Halaman ini menawarkan fitur untuk memasukkan data balita baru ke dalam *dataset* guna memperluas data yang digunakan untuk pelatihan model. Pengguna dapat mengisi formulir yang serupa dengan halaman klasifikasi, dengan tambahan atribut TB/U yang dijadikan target dalam proses klasifikasi. Selain itu, tersedia opsi untuk menambahkan data melalui *file* yang diunggah. Pada proses ini, dilakukan juga pelatihan ulang model dengan tambahan data yang telah ditambahkan ke dalam *dataset*. Hasil dari proses ini akan menampilkan pesan mengenai keberhasilan penambahan data dan pelatihan ulang model.



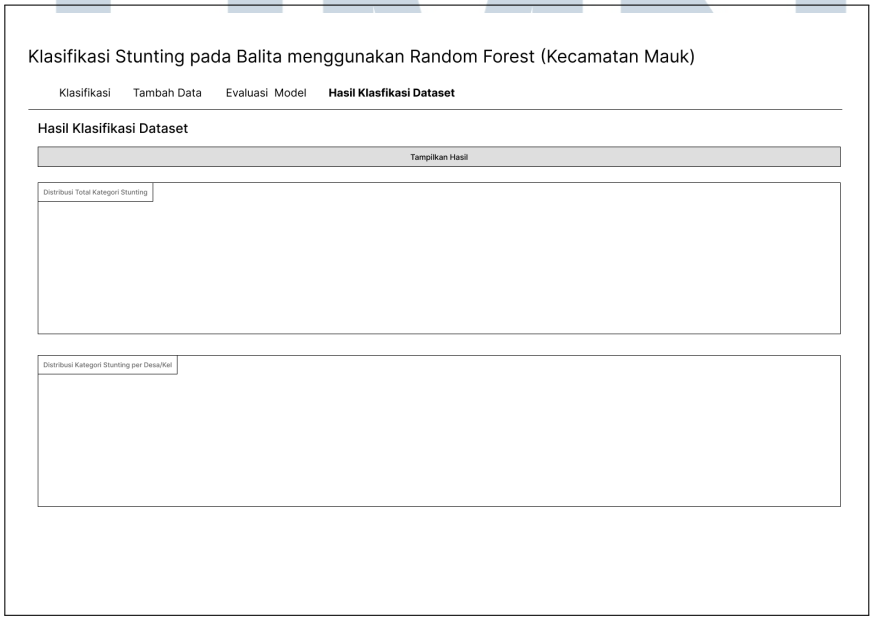
Gambar 3.11. *Wireframe* Halaman Tambah Data

Gambar 3.12 menunjukkan *wireframe* halaman evaluasi model. Halaman ini dirancang untuk mengevaluasi performa model dengan menyajikan berbagai metrik evaluasi, seperti *Confusion Matrix* untuk data pengujian serta nilai *Accuracy* untuk data pelatihan dan pengujian. Pengguna dapat menghasilkan laporan performa model dengan memlih tombol ”Get Performance Report”, yang akan menampilkan metrik seperti *Precision*, *Recall*, dan *F1-score* dalam tabel terpisah untuk data *training* dan *testing*.



Gambar 3.12. *Wireframe* Halaman Evaluasi Model

Gambar 3.13 menunjukkan *wireframe* halaman hasil klasifikasi *dataset*. Halaman ini menampilkan visualisasi distribusi hasil klasifikasi untuk memudahkan pengguna dalam memahami data. Pengguna dapat melihat distribusi total kategori *stunting* serta distribusi berdasarkan desa atau kelurahan dalam bentuk diagram lingkaran. Visualisasi ini akan ditampilkan setelah pengguna memilih tombol ”Tampilkan Hasil”.



Gambar 3.13. *Wireframe* Halaman Hasil Klasifikasi *Dataset*