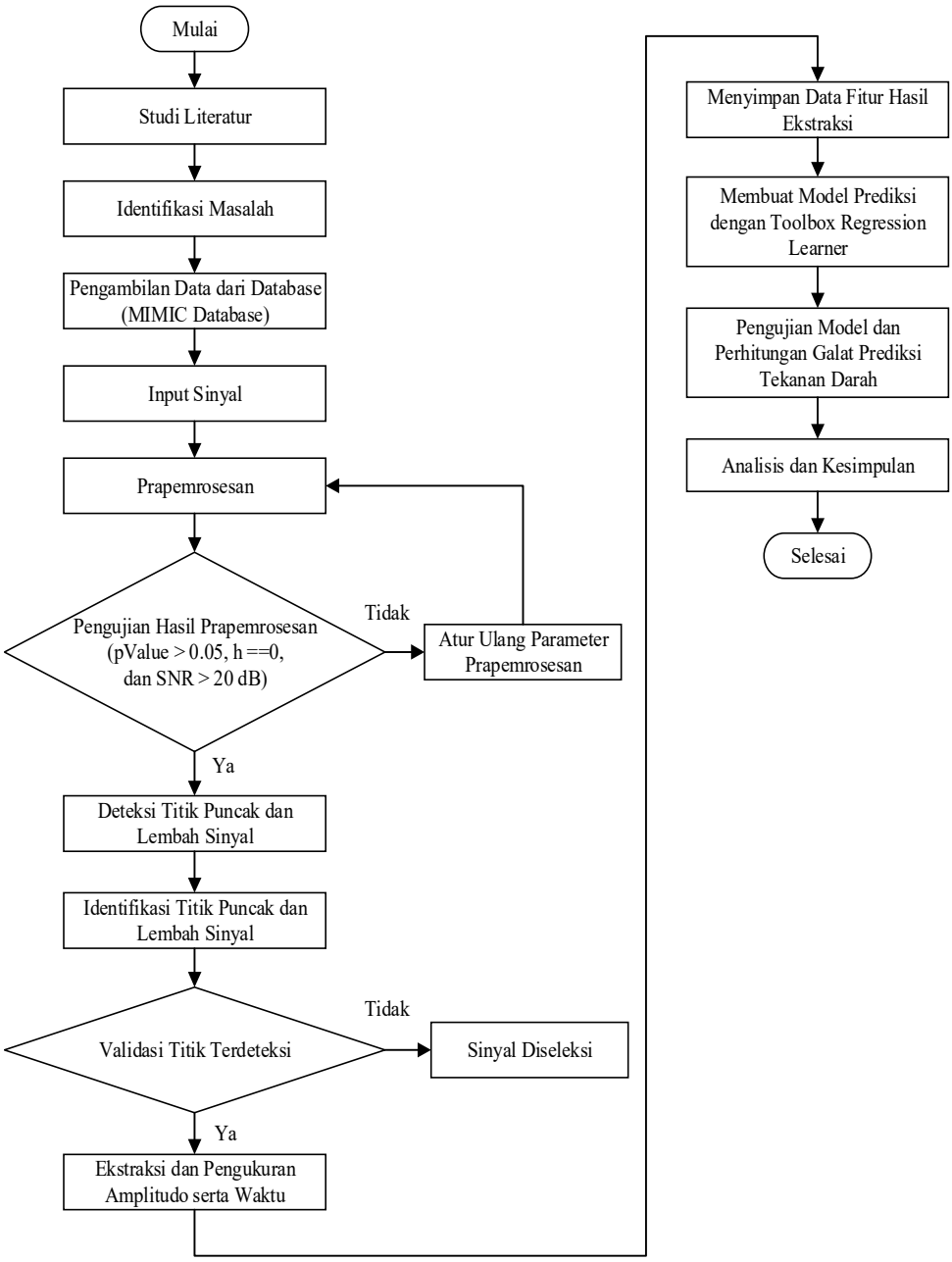
BAB III

METODE PENELITIAN

3.1 *Flowchart* Penelitian



Gambar 3.1 *Flowchart* Metode Penelitian

Gambar 3.1 menunjukkan metodologi penelitian yang diterapkan dalam tugas akhir ini, yang mengadopsi alur kerja (*workflow*) machine learning untuk membangun model prediksi tekanan darah dari sinyal *Photoplethysmography* (PPG). Setiap tahapan penelitian dijelaskan bersamaan dengan konsep machine learning yang relevan, mengidentifikasi secara jelas keterkaitannya dengan tahapan utama dalam *machine learning*. Secara konseptual, alur penelitian ini mencakup tahapan-tahapan *machine learning*: *Problem Statement* (Tahap 1-2), *Data Collection* (Tahap 3 - Tahap 12), *Model Selection & Training* (Tahap 13), Model *Evaluation* (Tahap 14), dan *Analysis & Conclusion* (Tahap 15). Dijelaskan sebagai berikut:

1. Tahap pertama adalah melakukan studi literatur. Tahap awal ini melibatkan studi literatur mendalam mengenai prinsip dasar PPG, metode pengukuran tekanan darah, teknik pra-pemrosesan sinyal PPG, analisis kualitas sinyal, deteksi dan identifikasi fitur sinyal, hubungan antara fitur PPG dan tekanan darah, serta model-model prediksi berbasis PPG. Studi ini secara langsung berkontribusi pada tahap *Problem Statement* dalam *machine learning*, di mana tujuan penelitian (pengembangan metode pengukuran tekanan darah non-invasif dan berkelanjutan menggunakan PPG) dan batasan masalah diidentifikasi.
2. Tahap kedua adalah identifikasi masalah. Berdasarkan studi literatur, tantangan spesifik dalam memproses sinyal PPG (seperti *noise* dan *baseline wander*) serta mendeteksi fitur-fitur penting untuk analisis diidentifikasi. Tahap ini memperjelas dan memperinci definisi masalah yang akan

diselesaikan melalui aplikasi *machine learning*, yang masih termasuk dalam tahap *Problem Statement*.

3. Tahap ketiga adalah pengambilan data dari basis data. Data sinyal PPG dan data tekanan darah fisiologis yang sesuai dikumpulkan dari basis data eksternal MIMIC yang tersedia di PhysioNet. Proses ini merupakan implementasi dari tahap *Data Collection* dalam *machine learning*, yang menyediakan data mentah yang diperlukan untuk langkah-langkah selanjutnya.
4. Tahap keempat adalah input sinyal. Data sinyal PPG dan tekanan darah yang telah diunduh dimasukkan ke dalam lingkungan pemrosesan (MATLAB). Langkah ini termasuk dalam tahap *Data Collection*, sebagai langkah awal dalam mempersiapkan data.
5. Tahap kelima adalah pra-pemrosesan. Sinyal PPG dipra-pemrosesan melalui serangkaian langkah seperti *detrending*, *smoothing*, dan normalisasi. Tahap ini merupakan bagian dari tahap *Data Collection*, yang bertujuan untuk membersihkan dan menstandarisasi data.
6. Tahap keenam adalah pengujian hasil pra-pemrosesan. Kualitas sinyal setelah pra-pemrosesan dievaluasi secara visual. Dalam kerangka *Data Collection* langkah ini memastikan bahwa proses pembersihan memberikan hasil yang diharapkan.
7. Tahap ketujuh adalah validasi hasil pengujian. Validasi kuantitatif menggunakan *p-value* dan SNR dilakukan untuk memastikan kualitas sinyal yang memadai. Ini adalah langkah lebih lanjut dalam tahap *Data Collection*,

di mana kualitas data dinilai berdasarkan metrik kuantitatif sebelum fitur diekstraksi.

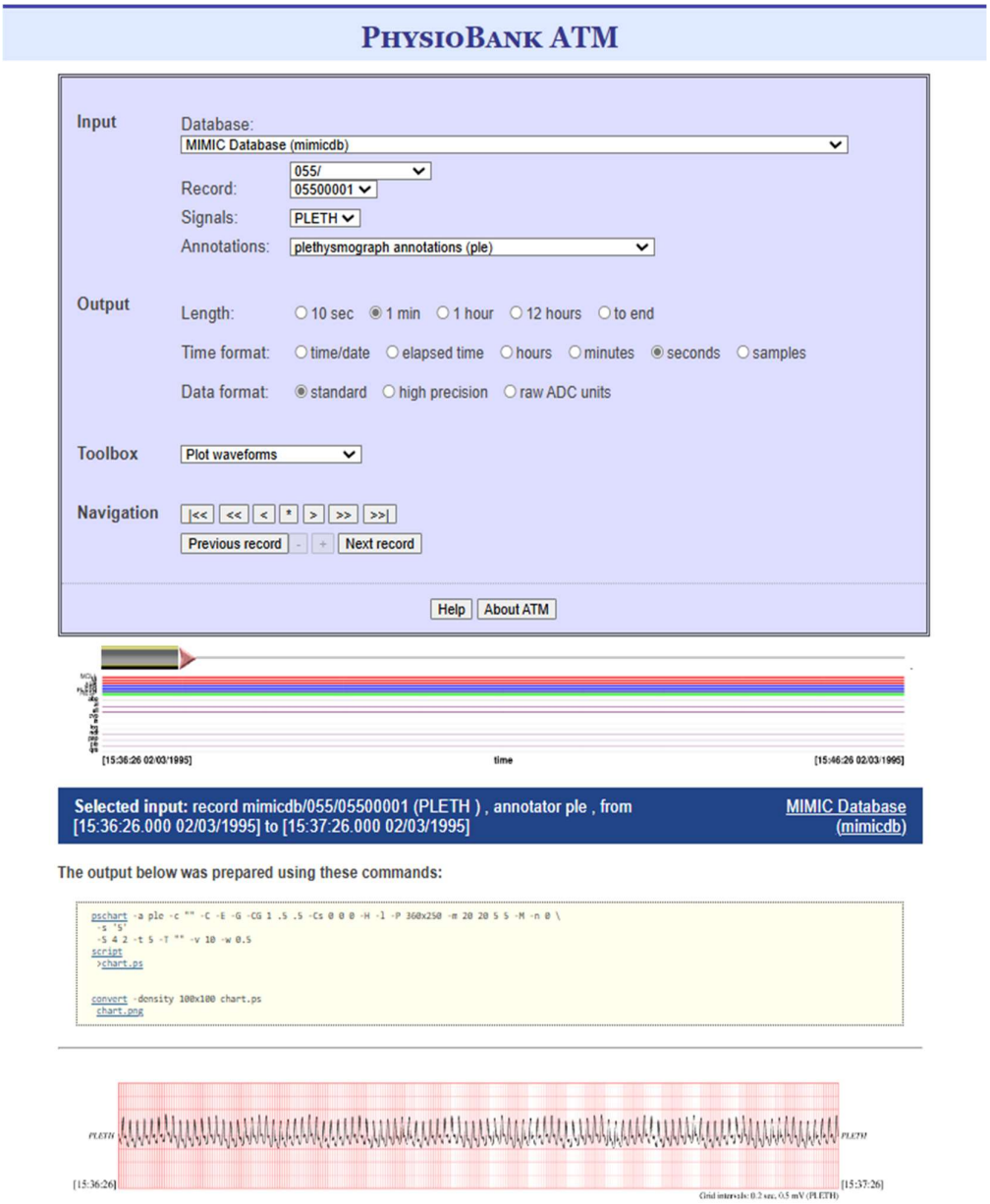
8. Tahap kedelapan adalah deteksi titik puncak dan lembah sinyal. Algoritma diterapkan untuk mendeteksi titik-titik penting (puncak dan lembah) pada sinyal PPG. Tahap ini termasuk dalam *Data Collection* sebagai langkah awal dalam mengidentifikasi informasi yang relevan.
9. Tahap kesembilan adalah identifikasi titik puncak dan lembah sinyal. Titik-titik yang terdeteksi diklasifikasikan lebih lanjut. Ini adalah bagian lanjutan dari tahap *Data Collection*, menyaring informasi yang lebih spesifik dan bermakna dari sinyal PPG.
10. Tahap kesepuluh adalah validasi titik terdeteksi. Akurasi deteksi titik-titik penting divalidasi. Ini merupakan langkah kontrol kualitas dalam tahap *Data Collection*, memastikan bahwa fitur-fitur yang akan digunakan akurat.
11. Tahap kesebelas adalah ekstraksi dan pengukuran amplitudo serta waktu. Amplitudo dan interval waktu antara titik-titik penting diukur dan diekstraksi. Tahap ini merupakan bagian dari *Data Collection*, menghasilkan sekumpulan fitur numerik yang akan digunakan sebagai *input* model.
12. Tahap kedua belas adalah menyimpan data fitur hasil ekstraksi. Data fitur yang telah diekstraksi disimpan untuk digunakan dalam pelatihan dan pengujian model. Ini adalah langkah akhir dalam tahap *Data Collection*, menyiapkan data yang telah direkayasa untuk pemodelan.
13. Tahap ketiga belas adalah membuat model prediksi dengan *toolbox regression learner*. Fitur-fitur yang diekstraksi digunakan untuk melatih

model prediksi tekanan darah. Tahap ini mencakup tahap *Model Selection* (pemilihan algoritma regresi yang tersedia di *toolbox*) dan tahap *Model Training* (melatih model menggunakan data fitur).

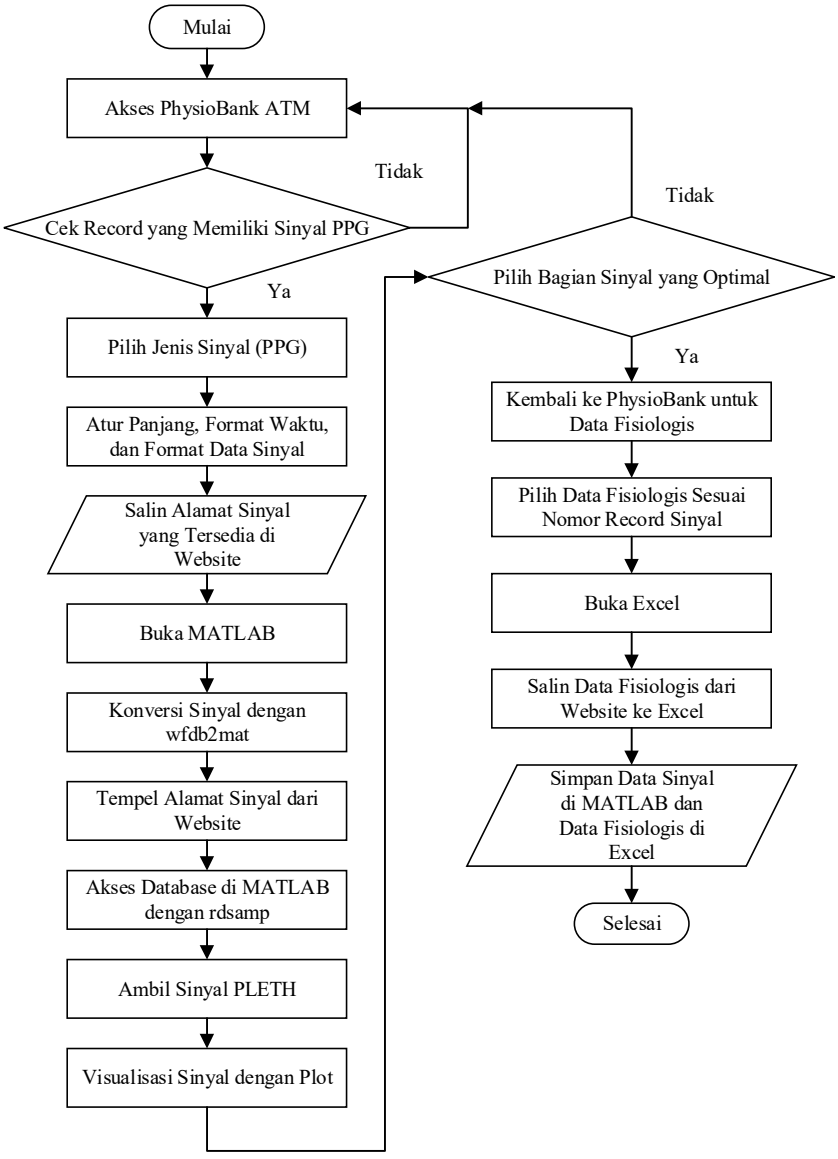
14. Tahap keempat belas adalah pengujian model dan perhitungan galat prediksi tekanan darah. Model yang telah dilatih diuji kinerjanya menggunakan data uji. Galat prediksi dihitung untuk tahap *Model Evaluation*, mengukur kemampuan model dalam memprediksi tekanan darah pada data yang belum pernah dilihat.
15. Tahap kelima belas adalah analisis dan kesimpulan. Hasil evaluasi model dianalisis. Tahap ini merupakan bagian dari evaluasi dan interpretasi hasil dalam alur kerja *machine learning*.

3.2 Pengambilan Data dari Basis data

Pada penelitian ini, sinyal PPG diambil dari basis data MIMIC, salah satu basis data fisiologis berkualitas tinggi yang tersedia secara publik melalui *PhysioNet* Gambar 3.2. Basis data ini sering digunakan dalam penelitian di bidang kesehatan. Pengambilan sinyal dilakukan menggunakan fungsi khusus di MATLAB untuk mengekstraksi data dari rekaman yang relevan. Proses pengambilan data dijelaskan secara rinci melalui diagram alir pada Gambar 3.3 yang menggambarkan setiap langkah, mulai dari akses ke basis data hingga data siap dianalisis lebih lanjut.



Gambar 3.2 Contoh Proses Akses Sinyal PPG melalui Situs web PhysioBank ATM



Gambar 3.3 *Flowchart* Pengambilan Data dari Basis data

Proses pengambilan data dari basis data dilakukan melalui beberapa langkah sistematis berikut:

1. Akses *PhysioBank* ATM

   Buka *situs web PhysioNet*, tempat *PhysioBank* ATM tersedia. ATM ini adalah antarmuka yang memudahkan akses berbagai set data sinyal fisiologis seperti PPG.

2. Cek *Record* yang Memiliki Sinyal PPG

   Cari dan pilih *record* dalam basis data MIMIC yang memiliki sinyal PPG, biasanya disimpan dalam bentuk file *PLETH* dalam setiap set data.

3. Pilih Jenis Sinyal (PPG)

   Setelah diketahui bahwa *record* yang dipilih memiliki sinyal PPG, maka pilih file *PLETH* tersebut untuk diproses pada tahap selanjutnya.

4. Atur Panjang, Format Waktu, dan Format Data Sinyal

   Sesuaikan panjang sinyal, format waktu, dan format data sesuai kebutuhan penelitian, karena hal ini akan memengaruhi pemrosesan selanjutnya.

5. Salin Alamat Sinyal yang Tersedia di *Situs web*

   Setelah konfigurasi sinyal selesai, salin alamat atau tautan sinyal yang disediakan di *situs web PhysioNet*.

6. Buka MATLAB

   Buka aplikasi MATLAB sebagai *platform* utama untuk mengelola dan menganalisis data sinyal PPG yang akan diambil.

7. Konversi Sinyal dengan *wfdb2mat*

   Gunakan perintah *wfdb2mat* di MATLAB untuk mengonversi file sinyal dari *PhysioNet* ke format MATLAB, memudahkan akses dan pengolahan data.

8. Tempel Alamat Sinyal dari *Situs web*

   Masukkan alamat sinyal yang telah disalin ke dalam perintah *wfdb2mat* atau fungsi yang relevan di MATLAB agar data dapat diakses langsung.

9. Akses Basis data di MATLAB dengan *rdsamp*

   Jalankan perintah *rdsamp* untuk mengambil sinyal dari basis data yang telah dikonversi. Ini memungkinkan MATLAB membaca dan mengekstrak sinyal.

10. Ambil Sinyal *PLETH*

    Pilih sinyal *PLETH* yang sesuai dengan file yang dipilih pada tahap pilih jenis sinyal PPG (tahap ketiga).

11. Visualisasi Sinyal dengan Plot

    Buat plot dari sinyal yang diambil untuk melihat visualisasi awal. Ini membantu memverifikasi kualitas dan kelengkapan data sinyal PPG.

12. Pilih Bagian Sinyal yang Optimal

    Tinjau plot sinyal untuk memilih bagian sinyal yang optimal, yaitu bagian dengan puncak sistolik dan diastolik yang terlihat jelas, untuk analisis lebih lanjut.

13. Kembali ke *PhysioBank* untuk Data Fisiologis

    Buka kembali *PhysioBank* untuk mengambil data fisiologis yang berkaitan dengan *record* sinyal PPG, seperti data tekanan darah yang relevan.

14. Pilih Data Fisiologis Sesuai Nomor *Record* Sinyal

    Pilih data fisiologis yang sesuai dengan nomor rekaman sinyal yang telah dipilih sebelumnya. Pastikan waktu pengukuran sinyal PPG dan tekanan darah sama untuk menjamin kesesuaian data dalam analisis bersama sinyal PPG.

15. Buka Excel

    Buka aplikasi Excel sebagai media untuk menyimpan dan mengelola data fisiologis yang diambil dari *PhysioNet*.

16. Salin Data Fisiologis dari Situs Web ke Excel

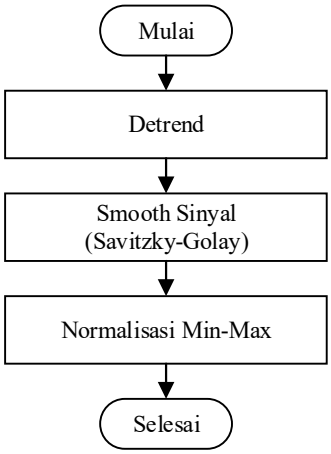
    Salin data fisiologis dari situs web *PhysioNet* dan tempelkan di Excel untuk disimpan sebagai data tambahan yang melengkapi data sinyal PPG.

17. Simpan Data Sinyal di MATLAB dan Data Fisiologis di Excel

    Simpan data sinyal PPG yang telah diproses di MATLAB dan data fisiologis di Excel. Dengan demikian, keduanya siap untuk analisis lebih lanjut.

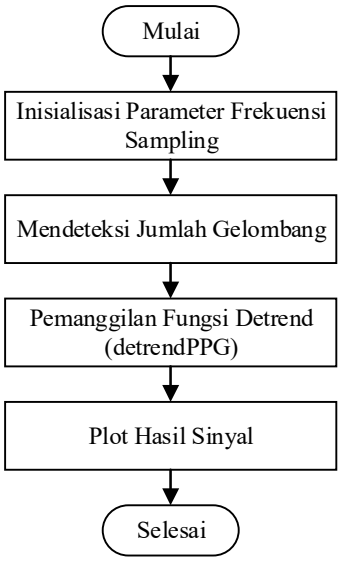
3.3 Pra-pemrosesan

Tahapan pra-pemrosesan merupakan langkah awal dalam pengolahan sinyal PPG, yang bertujuan untuk mengurangi efek *noise* dan artefak gerak yang dapat mempengaruhi kualitas informasi dalam sinyal asli. Proses pra-pemrosesan dimulai dengan *detrend*, yang bertujuan untuk menghilangkan tren atau *baseline* pada sinyal, sehingga menghasilkan sinyal yang lebih stabil. Setelah itu, dilakukan *smoothing* untuk mengurangi *noise* dengan menggunakan metode *Savitzky-Golay*. Langkah terakhir adalah normalisasi sinyal ke rentang [0, 1] untuk membuat proses ekstraksi fitur lebih efisien dan memastikan bahwa nilai fitur yang diekstraksi bermakna. Adapun Gambar 3.4 menggambarkan urutan dan rincian tahapan dalam perancangan model pra-pemrosesan ini.



Gambar 3.4 *Flowchart* Pra-premrosesan Sinyal PPG Asli

3.3.1 *Detrend*



Gambar 3.5 *Flowchart* Detrend

Proses *detrend* yang digunakan dalam pra-pemrosesan merujuk pada studi literatur yang telah dibahas pada Subbab 2.6.1, Implementasi langkah-langkahnya ditunjukkan pada Gambar 3.5, yang mencakup tahapan-tahapan berikut:

1. Inisialisasi Parameter Frekuensi Sampling

   Mengatur parameter awal, seperti frekuensi sampling (Fs = 500) untuk mendeteksi puncak sinyal dengan *findpeaks*.

2. Mendeteksi Jumlah Gelombang

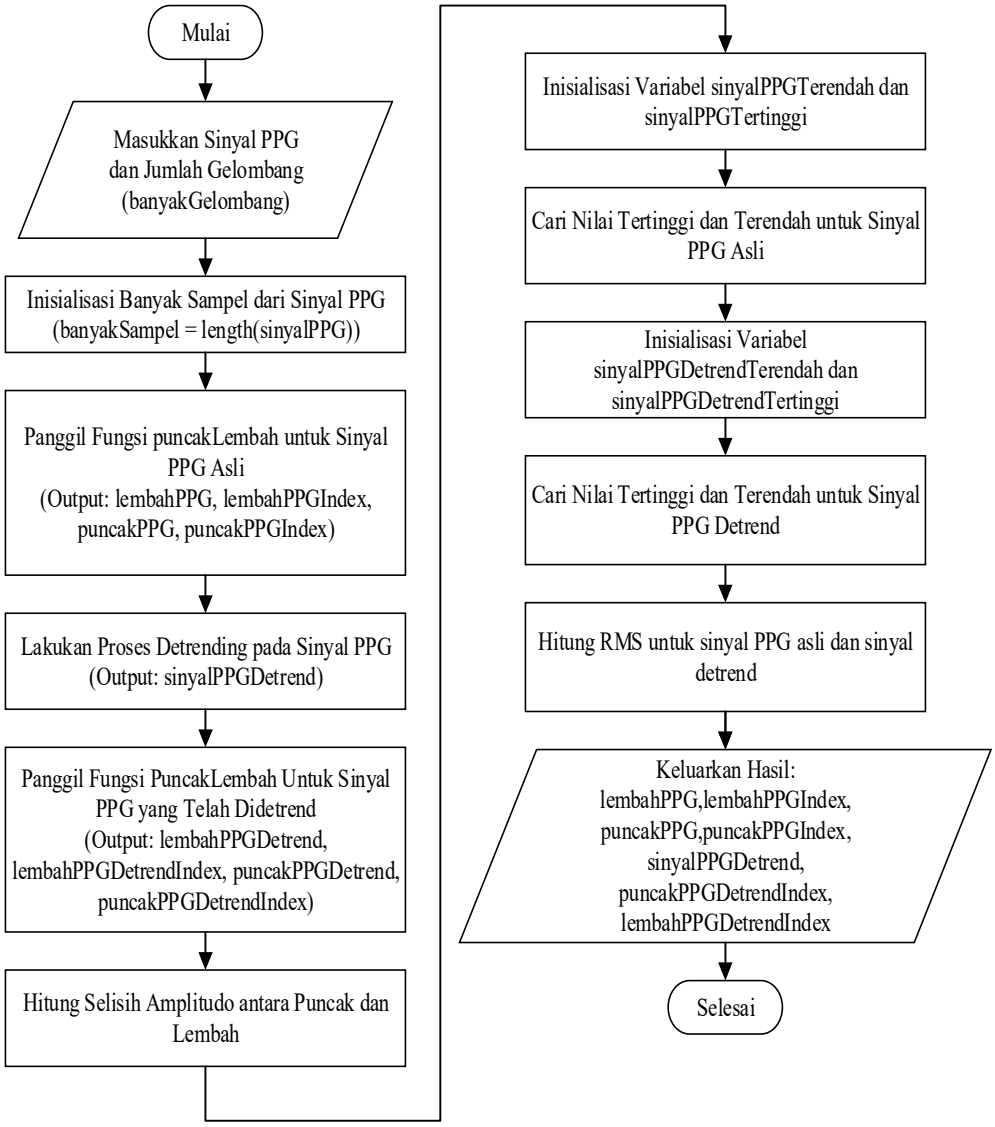
   Fungsi *findpeaks* digunakan untuk mendeteksi puncak pada sinyal PPG dengan ambang batas ketinggian puncak sebesar 0.3 mV. Batas ini ditetapkan karena pada semua sinyal PPG dari basis data, puncak sistolik selalu memiliki amplitudo lebih besar dari 0.3 mV. Nilai batas ini dapat disesuaikan sesuai dengan karakteristik sinyal PPG yang digunakan.

3. Panggil Fungsi *detrendPPG*

   Fungsi *detrendPPG* dipanggil untuk menghilangkan *baseline* sinyal, mengembalikan sinyal yang lebih stabil dan menyediakan berbagai *output* berupa titik puncak, lembah, dan sinyal *detrend*.

4. Plot Hasil Sinyal

3.3.1.1 Fungsi *Detrend*



Gambar 3.6 *Flowchart* Fungsi Detrend

Fungsi *detrendPPG* yang digunakan dalam pra-pemrosesan didasarkan pada persamaan (2.10) dan diimplementasikan melalui beberapa tahapan berikut:

1. Input Sinyal PPG dan Banyak Gelombang

   Masukkan sinyal PPG (*sinyalPPG*) dan jumlah gelombang (*banyakGelombang*) yang ingin dianalisis.

2. Hitung Panjang Sinyal

   Menghitung jumlah sampel dalam sinyal PPG dengan menggunakan *banyakSampel = length*(*sinyalPPG*).

3. Panggil Fungsi *puncakLembah* untuk Sinyal Asli

   Gunakan fungsi *puncakLembah* untuk mendeteksi puncak dan lembah dari sinyal PPG asli. Hasil yang diperoleh berupa variabel seperti *lembahPPG*, *lembahPPGIndex*, *puncakPPG*, dan *puncakPPGIndex*.

4. Lakukan *Detrend* pada Sinyal PPG

   Proses ini menghilangkan tren dari sinyal PPG untuk memfokuskan analisis pada perubahan kecil. *Output* proses ini disebut sinyalPPGDetrend.

5. Panggil Fungsi *puncakLembah* untuk Sinyal yang Didetrend

   Menggunakan fungsi *puncakLembah* pada sinyal yang telah didetrend untuk mendeteksi puncak dan lembah yang lebih bersih. Hasilnya adalah *lembahPPGDetrend, lembahPPGDetrendIndex, puncakPPGDetrend, dan puncakPPGDetrendIndex*.

6. Hitung Selisih Puncak-Lembah dengan Interpolasi

   Proses ini menghitung jarak antara puncak dan lembah melalui interpolasi, untuk menghasilkan nilai yang lebih akurat.

7. Inisialisasi Nilai Terendah dan Tertinggi untuk Sinyal Asli

   Menetapkan nilai awal minimum dan maksimum dari sinyal PPG asli untuk perbandingan lebih lanjut.

8. Cari nilai minimum dan maksimum

   Mencari nilai minimum dan maksimum dalam sinyal dengan melakukan perulangan pada setiap sampel sinyal.

9. Inisialisasi Nilai Terendah dan Tertinggi untuk Sinyal yang Didetrend

   Menetapkan nilai awal minimum dan maksimum dari sinyal PPG *detrend* untuk perbandingan lebih lanjut.

10. Cari Nilai Terendah dan Tertinggi dari Sinyal yang Didetrend

    Serupa dengan langkah sebelumnya, tahap ini mencari nilai minimum dan maksimum dari sinyal yang telah didetrend.

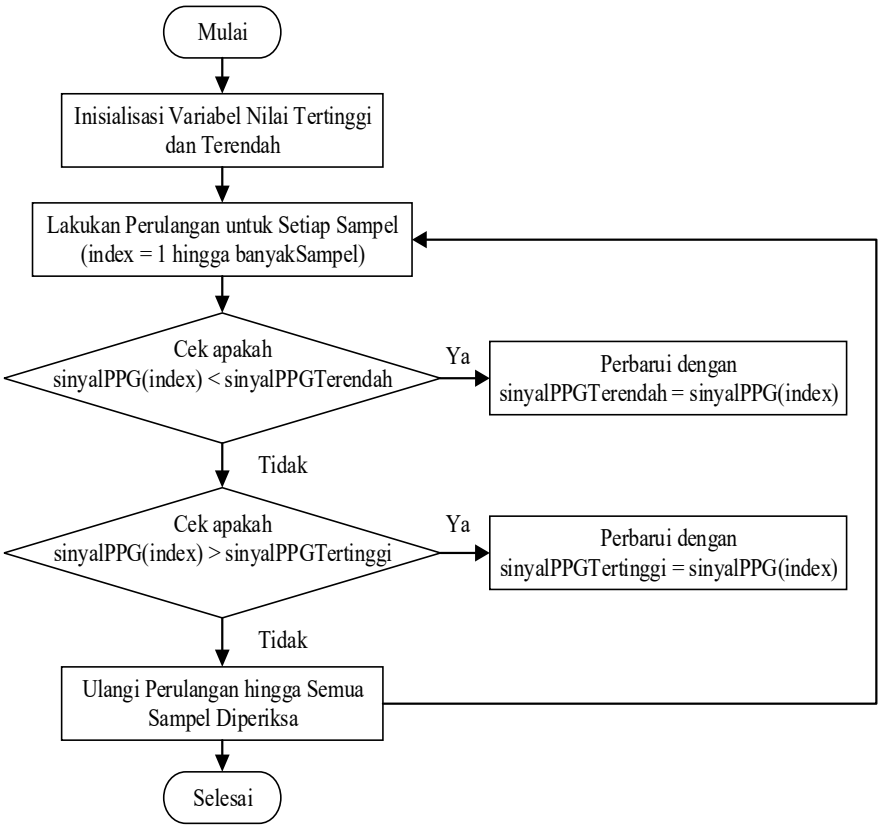
11. Hitung RMS untuk Sinyal Asli dan Sinyal yang Didetrend

    Menghitung nilai RMS (*Root Mean Square*) untuk sinyal asli dan yang didetrend. Nilai RMS memberikan ukuran kekuatan sinyal rata-rata.

12. *Output* Seluruh Hasil

    Proses ini menghasilkan *output* berupa semua variabel yang dihitung (seperti *lembahPPG, lembahPPGIndex, puncakPPG, puncakPPGIndex, sinyalPPGDetrend, puncakPPGDetrendIndex, lembahPPGDetrendIndex*).

3.3.1.2 Mencari Nilai Terendah dan Tertinggi Sinyal Asli



Gambar 3.7 *Flowchart* Mencari Nilai Terendah dan Tertinggi Sinyal Asli

Proses perulangan dalam mencari nilai minimun dan maksimum untuk sinyal PPG di dalam fungsi *detrendPPG* Gambar 3.7, dijelaskan sebagai berikut:

1. Inisialisasi Variabel

   Menginisialisasi variabel *sinyalPPGTerendah*, *sinyalPPGTertinggi* (untuk sinyal asli) dengan nilai yang sangat tinggi dan sangat rendah.

2. Lakukan Perulangan untuk Setiap Sampel pada Sinyal PPG

   Perulangan dilakukan untuk mengecek setiap elemen dalam sinyal asli *sinyalPPG*, mulai dari *index = 1* hingga *banyakSampel*.

3. Cek *sinyalPPG(index) < sinyalPPGTerendah*

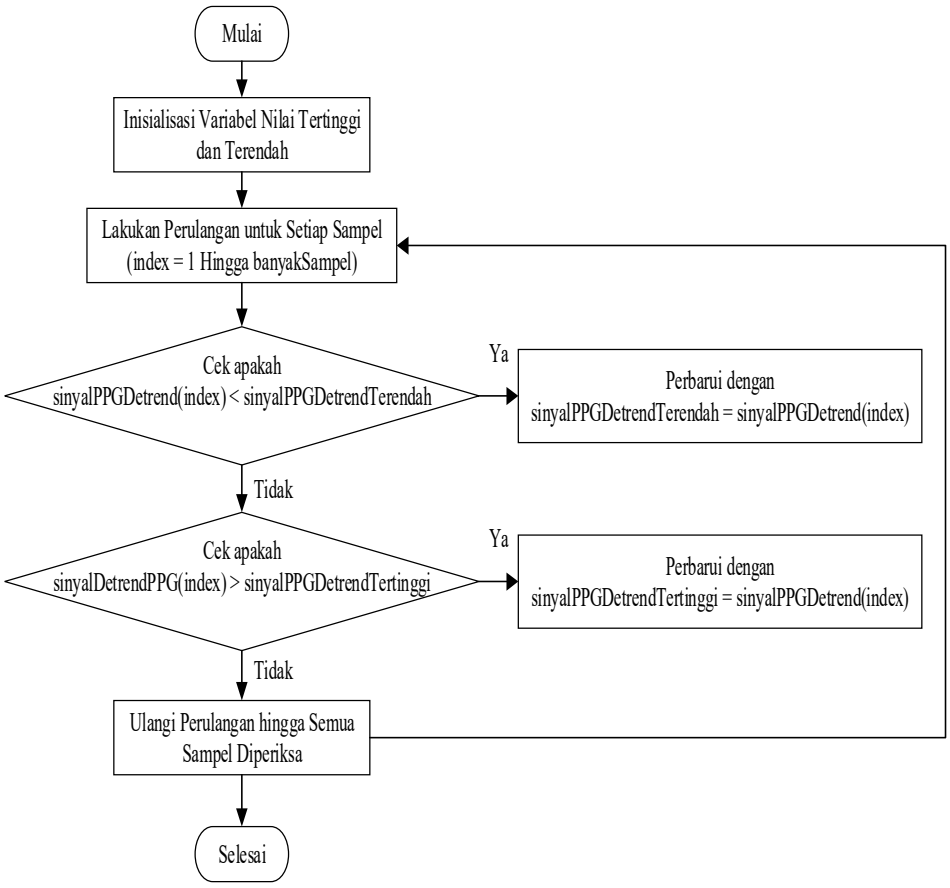
Mengecek nilai *sinyalPPG* pada indeks yang lebih kecil dari *sinyalPPGTerendah*. Jika kondisi terpenuhi, perbarui nilai *sinyalPPGTerendah*, jika tidak lanjutkan ke kondisi berikutnya.

4. Cek *sinyalPPG(index) > sinyalPPGTertinggi*

   Mengecek nilai *sinyalPPG* pada indeks yang lebih besar dari *sinyalPPGTertinggi*. Jika kondisi terpenuhi, perbarui nilai *sinyalPPGTertinggi*, jika tidak kembali ke proses perulangan untuk memproses indeks berikutnya.

5. Ulangi Perulangan hingga Semua Sampel Diperiksa

3.3.1.3 Mencari Nilai Terendah dan Tertinggi Sinyal Detrend



Gambar 3.8 *Flowchart* Mencari Nilai Terendah dan Tertinggi Sinyal Detrend

Proses perulangan dalam mencari nilai minimun dan maksimum untuk sinyal PPG di dalam fungsi *detrendPPG* Gambar 3.8, dijelaskan sebagai berikut:

1. Inisialisasi Variabel

   Menginisialisasi variabel *sinyalPPGDetrendTerendah*, *sinyalPPGDetrendTertinggi* (untuk sinyal *detrend*) dengan nilai yang sangat tinggi dan sangat rendah.

2. Lakukan Perulangan untuk Setiap Sampel pada Sinyal PPG

   Perulangan dilakukan untuk mengecek setiap elemen dalam sinyal asli *sinyalPPG*, mulai dari *index = 1* hingga *banyakSampel*.

3. Cek *sinyalPPGDetrend(index) < sinyalPPGDetrendTerendah*

   Mengecek nilai *sinyalPPGDetrend* pada indeks yang lebih kecil dari *sinyalPPGDetrendTerendah*. Jika kondisi terpenuhi, perbarui nilai *sinyalPPGDetrendTerendah*, jika tidak lanjutkan ke kondisi berikutnya.

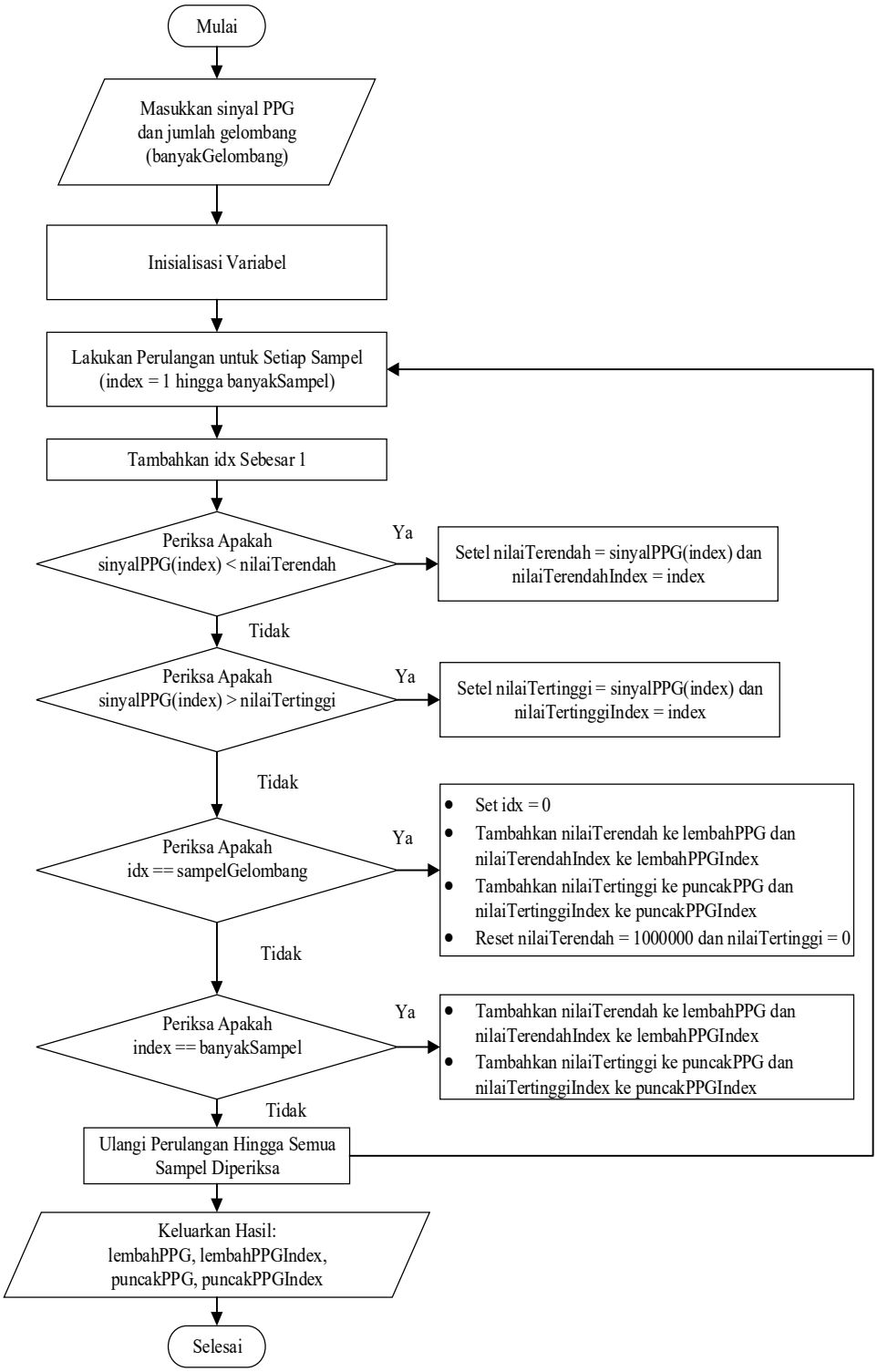
4. Cek *sinyalPPGDetrend (index) > sinyalPPGDetrendTertinggi*

   Mengecek nilai *sinyalPPGDetrend* pada indeks yang lebih besar dari *sinyalPPGDetrendTertinggi*. Jika kondisi terpenuhi, perbarui nilai *sinyalPPGDetrendTertinggi*, jika tidak kembali ke proses perulangan untuk memproses indeks berikutnya.

5. Ulangi Perulangan hingga Semua Sampel Diperiksa

   Mengulangi langkah-langkah dalam perulangan ini sampai semua sampel dalam sinyal selesai dianalisis.

3.3.1.5 Fungsi *puncakLembah*



Gambar 3.9 *Flowchart* Fungsi puncakLembah

Fungsi *puncakLembah* yang digunakan dalam pra-pemrosesan melibatkan beberapa langkah Gambar 3.9, dijelaskan sebagai berikut:

1. Input Sinyal PPG dan Banyak Gelombang

   Masukkan sinyal PPG (*sinyalPPG*) dan jumlah gelombang (*banyakGelombang*) yang ingin dianalisis.

2. Inisialisasi Variabel

   Inisialisasi variabel dilakukan dengan cara berikut: *banyakSampel = length(sinyalPPG)* untuk menghitung jumlah sampel dalam sinyal PPG, *sampelGelombang = round(banyakSampel / banyakGelombang)* untuk menentukan jumlah sampel per gelombang, dan *idx = 0* untuk menginisialisasi indeks. Kemudian variabel *lembahPPG, lembahPPGIndex, puncakPPG, puncakPPGIndex* diinisiasi sebagai *array* kosong.

3. Lakukan Perulangan untuk Setiap Sampel

   Melakukan perulangan untuk memproses setiap sampel dalam sinyal PPG, dari *indeks 1* hingga *banyakSampel*.

4. Tambahkan *idx* Sebesar 1

   Meningkatkan nilai *idx* dengan 1 pada setiap perulangan, untuk melacak jumlah sampel dalam satu gelombang.

5. Cek *sinyalPPG(index) < nilaiTerendah*

   Mengecek nilai *sinyalPPG* pada indeks yang lebih kecil dari *nilaiTerendah*. Jika kondisi terpenuhi, perbarui *nilaiTerendah* dengan nilai sinyal saat ini dan simpan indeksnya sebagai *nilaiTerendahIndex*, jika kondisi tidak terpenuhi lanjutkan ke langkah berikutnya tanpa mengubah *nilaiTerendah*.

6. Cek *sinyalPPG(index) > nilaiTertinggi*

   Mengecek nilai *sinyalPPG* pada indeks yang lebih besar dari *nilaiTertinggi*. Jika kondisi terpenuhi, perbarui *nilaiTertinggi* dengan nilai sinyal saat ini dan simpan indeksnya sebagai *nilaiTertinggiIndex*, jika kondisi tidak terpenuhi lanjutkan ke langkah berikutnya tanpa mengubah *nilaiTertinggi*.

7. Cek *index == sampelGelombang*

   Mengecek apakah *idx* sama dengan jumlah sampel per gelombang (satu gelombang telah terproses). Jika kondisi terpenuhi, maka:

   a. Setel ulang *idx* ke 0 untuk memulai gelombang baru
   b. Simpan *nilaiTerendah* dan *nilaiTerendahIndex* di *lembahPPG* dan *lembahPPGIndex*.
   c. Simpan *nilaiTertinggi* dan *nilaiTertinggiIndex* di *puncakPPG* dan *puncakPPGIndex*.
   d. Reset *nilaiTerendah* ke nilai yang sangat tinggi (1000000) dan *nilaiTertinggi* ke 0 sebagai persiapan untuk gelombang berikutnya.

   Jika kondisi tidak terpenuhi, lanjutkan ke sampel berikutnya

8. Cek *index == banyakSampel*

   Mengecek apakah *idx* sama dengan jumlah total sampel (*banyakSampel*). Jika kondisi terpenuhi, maka:

   a. Tambahkan nilai *nilaiTerendah* dan *nilaiTerendahIndex* ke dalam *lembahPPG* dan *lembahPPGIndex*.
   b. Tambahkan nilai *nilaiTertinggi* dan *nilaiTertinggiIndex* ke dalam *puncakPPG* dan *puncakPPGIndex*.

   Jika kondisi tidak terpenuhi, lanjutkan ke sampel berikutnya

9. Ulangi Perulangan hingga Semua Sampel Diperiksa

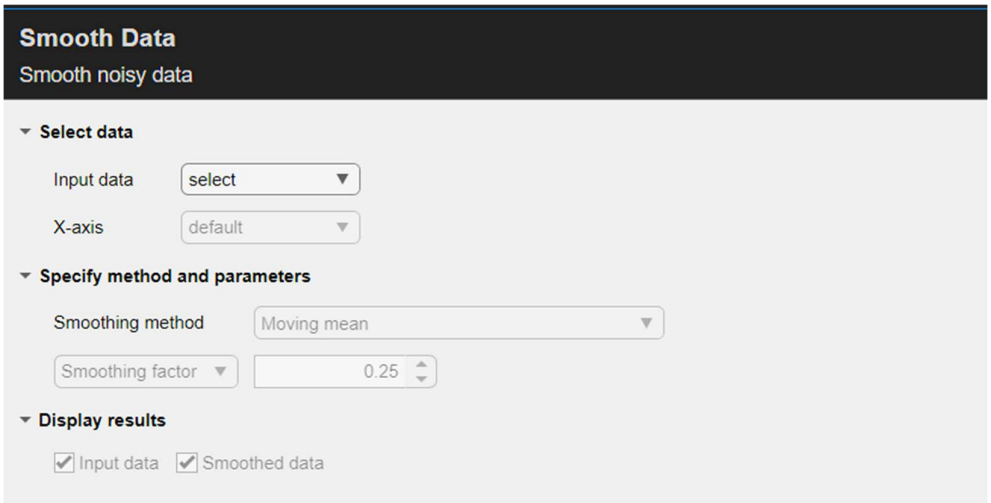
Mengulangi langkah-langkah dalam perulangan ini sampai semua sampel dalam sinyal selesai dianalisis.

10. Keluarkan Hasil

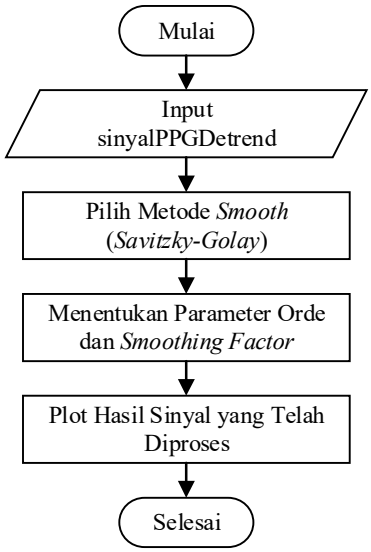
    Menampilkan atau menyimpan hasil ekstraksi puncak (*puncakPPG* dan *puncakPPGIndex*) dan lembah (*lembahPPG* dan *lembahPPGIndex*).

3.3.2 *Smooth* Sinyal

Dalam penelitian ini, proses *smoothing* sinyal dilakukan menggunakan *toolbox Smooth* Data pada MATLAB. *Toolbox* ini menyediakan berbagai metode penghalusan data, salah satunya adalah metode Savitzky-Golay yang digunakan dalam penelitian ini karena efektif dalam mereduksi *noise* tanpa mengubah karakteristik utama sinyal, sebagaimana dijelaskan pada Subbab 2.6.2. Proses *smoothing* ini bertujuan untuk meningkatkan kualitas sinyal PPG agar tahap analisis dan ekstraksi fitur dapat dilakukan dengan lebih akurat. Implementasi *smoothing* ditunjukkan pada Gambar 3.10 yang menampilkan antarmuka *toolbox Smooth* Data beserta opsi metode dan parameter yang dapat disesuaikan sesuai kebutuhan analisis.



Gambar 3.10 Tampilan *Toolbox Smooth* Data di MATLAB



Gambar 3.11 *Flowchart Smooth* Sinyal

Proses *smooth* sinyal yang digunakan dalam pra-pemrosesan melibatkan beberapa langkah Gambar 3.11, dijelaskan sebagai berikut:

1. Input Sinyal yang Telah Didetrend

   Masukkan sinyal PPG yang telah didetrend (*sinyalPPGDetrend*) yang akan diproses ke dalam sistem.

2. Pilih Metode *Smooth*

   Pilih *Savitzky-Golay* sebagai metode smooth yang akan diterapkan.

3. Menentukan Parameter

   Tentukan nilai parameter orde sebesar 3 dan *smoothing factor* sebesar 0.2. Nilai ini dapat disesuaikan jika hasil pra-pemrosesan sinyal masih mengandung *noise* yang memengaruhi kualitas sinyal.

4. Plot Hasil Sinyal

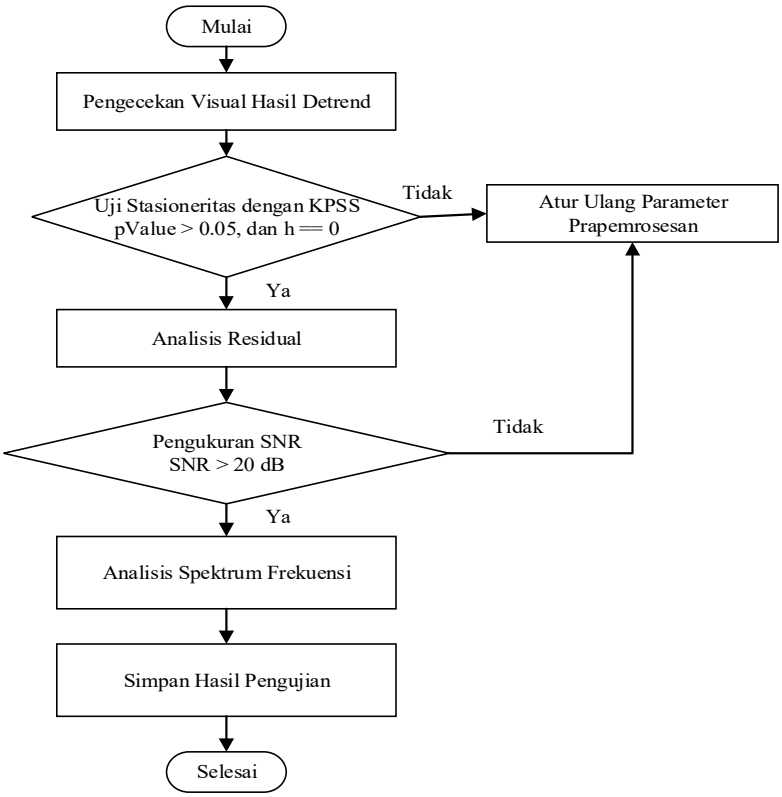
3.3.3 Normalisasi Sinyal

Proses normalisasi sinyal PPG dalam penelitian ini dilakukan dengan mengubah nilai sinyal ke dalam rentang [0, 1] menggunakan metode *min-max*,

sebagaimana dijelaskan pada Subbab 2.6.3 dan ditunjukkan dalam Persamaan (2.11). Langkah ini bertujuan untuk memastikan skala data seragam, sehingga proses analisis dan ekstraksi fitur dapat dilakukan secara lebih konsisten dan akurat.

Konversi waktu dari detik ke menit dilakukan dengan menyesuaikan titik awal data. Setiap set data waktu dihitung relatif terhadap waktu awal, yaitu dengan mengurangi nilai waktu awal dari seluruh data waktu, kemudian membaginya dengan 60. Rentang waktu pada setiap set data tidak selalu dimulai dari nol, melainkan bervariasi, seperti 120-180 detik atau 60-120 detik, tergantung pada pemotongan sinyal dari basis data. Proses ini memastikan skala waktu lebih seragam dalam rentang [0,1] menit dan mempermudah interpretasi dalam analisis durasi sinyal.

3.4 Pengujian Hasil Pra-pemrosesan



Gambar 3.12 *Flowchart* Pengujian Hasil Pra-pemrosesan

Pengujian hasil pra-pemrosesan melibatkan beberapa tahap Gambar 3.12, yang dijelaskan sebagai berikut:

1. Pengecekan Visual Hasil *Detrend*

   Tampilkan sinyal PPG asli dan *baseline* yang telah dihilangkan dengan *baseline = sinyalPPG – sinyalPPGDetrend*. Kemudian tampilkan sinyal asli dan *baseline* pada *subplot* pertama, serta plot sinyal yang telah didetrend pada *subplot* kedua.

2. Uji Stasionaritas dengan KPSS

   Lakukan Uji KPSS pada sinyal yang didetrend. Jika *pValue* > 0.05, nyatakan sinyal stasioner, jika tidak, nyatakan sinyal non-stasioner dan atur ulang parameter pra-pemrosesan.

3. Analisis Residual

   Hitung residual dengan *residual = sinyalPPGDetrend – sinyalPPGSmooth*. Kemudian tampilkan tampilkan residual dalam plot untuk melihat sisa *noise* setelah *detrending*.

4. Pengukuran SNR (*Signal-to-Noise Ratio*)

   Hitung daya *sinyal P_signal* dan daya *noise P_noise*, kemudian hitung SNR sebelum *smoothing* (SNRSebelum) dan setelah *smoothing* (SNRSesudah). Jika SNR > 20 dB, sinyal dapat digunakan untuk proses selanjutnya, jika tidak, lakukan pengaturan ulang parameter *pra_pemrosesan*.

5. Analisis Spektrum Frekuensi

   Hitung FFT sinyal asli (*fftSinyalAsli*) dan sinyal pra-pemrosesn (*fftPrapemrosesan*). Kemudian tampilkan plot spektrum frekuensi sebelum

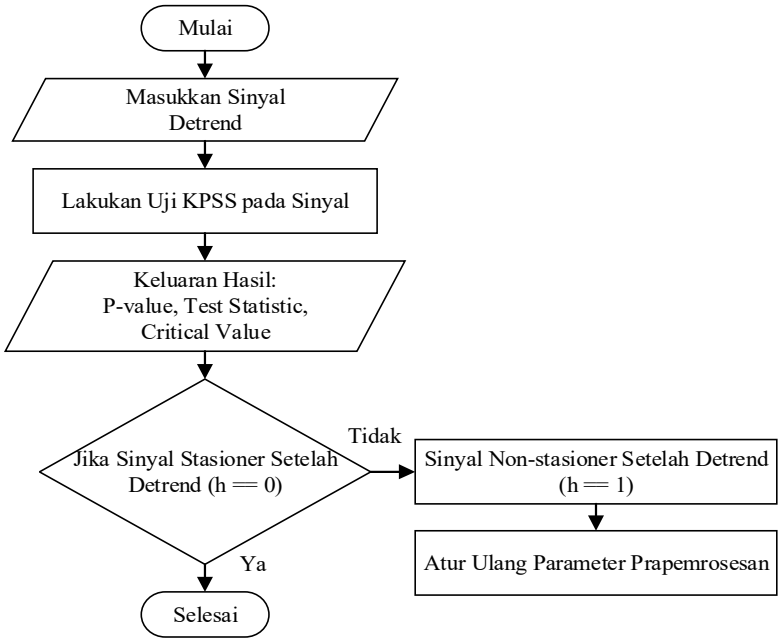
dan sesudah prapemrosesan. Tandai puncak-puncak frekuensi pada plot spektrum.

6. Simpan Hasil Pengujian

   Buat tabel *hasilPrapemrosesan* yang berisi hasil pengujian KPSS dan SNR, dan simpan ke file Excel *hasilPrapemrosesan.xlsx*. Kemudian gambar plot residual dan plot spektrum frekuensi dalam format png.

3.4.1 Uji Stasioneritas dengan KPSS

Uji KPSS dalam penelitian ini diterapkan secara khusus sebagai alat validasi terhadap kinerja algoritma *detrending*, dengan pendekatan yang berbeda dari penggunaan konvensional. Metode ini difokuskan pada analisis residual, yaitu selisih antara sinyal asli dan sinyal hasil *detrending*, di mana hipotesis nol menyatakan bahwa residual bersifat stasioner. Nilai p-value >0.05 mengindikasikan bahwa algoritma berhasil menghilangkan komponen non-stasioner tanpa mengganggu informasi fisiologis, sebagaimana dijelaskan pada Subbab 2.7.1. Proses validasi ini diperkuat dengan dua pendekatan tambahan: (1) perhitungan *signal-to-noise ratio* (SNR) untuk menilai peningkatan kualitas sinyal, dan (2) evaluasi terhadap hasil ekstraksi titik fisiologis. Kombinasi ketiga metode ini dirancang untuk memastikan bahwa algoritma pra-pemrosesan berfungsi sesuai dengan prinsip dasar yang telah ditetapkan, sekaligus mempertahankan integritas karakteristik fisiologis sinyal PPG.



Gambar 3.13 *Flowchart* Uji Stationeritas dengan KPSS

Proses uji KPSS dalam pengujian hasil pra-pemrosesan melibatkan beberapa langkah Gambar 3.13, dijelaskan sebagai berikut:

1. Masukkan Sinyal *Detrend*

   Masukkan sinyal yang telah melalui proses *detrend* untuk diuji stasioneritasnya.

2. Lakukan Uji KPSS pada Sinyal

   Mengaplikasikan uji KPSS untuk menentukan apakah sinyal bersifat stasioner atau non-stasioner.

3. Keluarkan Hasil

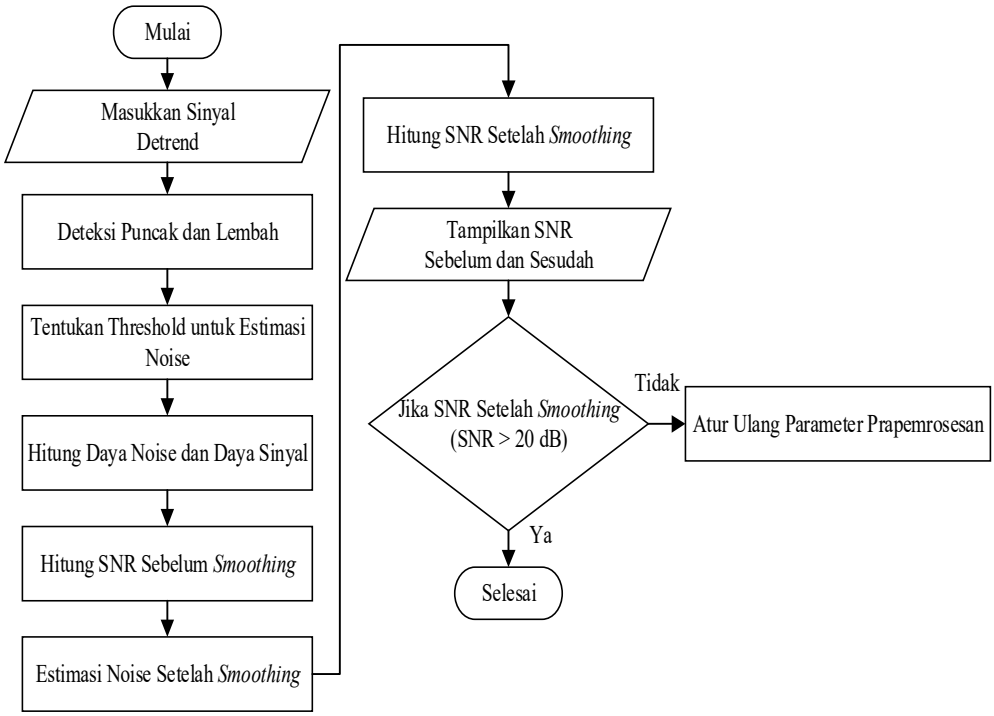
   Menyimpan atau menampilkan nilai *P-value*, *test statistic*, dan *critical value* dari hasil uji KPSS sebagai bagian dari hasil pengujian.

4. Cek Stasioneritas Sinyal

a Jika h = 0: Sinyal bersifat stasioner setelah *detrend*, sehingga dapat dilanjutkan ke proses berikutnya.

b Jika h = 1: Sinyal bersifat non-stasioner setelah *detrend*, maka pengaturan ulang parameter pra-pemrosesan harus dilakukan.

3.4.2 Proses Pengukuran SNR



Gambar 3.14 *Flowchart* Proses Pengukuran SNR

Proses Pengukuran SNR dalam pengujian hasil pra-pemrosesan sebagaimana dijelaskan pada bagian 2.7.3, melibatkan beberapa langkah Gambar 3.14, yang dijelaskan sebagai berikut:

1. Masukkan Sinyal *Detrend*

   Masukkan sinyal yang telah melalui proses *detrend* untuk proses perhitungan SNR menggunakan persamaan (2.12).

2. Deteksi Puncak dan Lembah

Menentukan titik puncak dan lembah pada sinyal dengan bantuan fungsi *findpeaks*.

3. Menentukan *Threshold* untuk Estimasi *Noise*

   Menentukan ambang batas (*threshold*) untuk membedakan komponen sinyal utama dan *noise*.

4. Menghitung Daya *Noise* dan Daya Sinyal

   Menghitung komponen daya sinyal dan *noise* berdasarkan sinyal yang didapat, untuk estimasi SNR.

5. Menghiturng SNR Sebelum *Smoothing*

   Menghitung nilai SNR (sebelum dilakukan *smoothing* atau perataan) untuk mengetahui kualitas sinyal awal.

6. Estimasi *Noise* Setelah *Smoothing*

   Mengestimasi tingkat *noise* setelah sinyal mengalami *smoothing*, untuk melihat apakah ada peningkatan kualitas sinyal.

7. Hitung SNR Setelah *Smoothing*

   Menghitung nilai SNR setelah proses *smoothing* untuk perbandingan kualitas sinyal sebelum dan sesudah proses perataan.

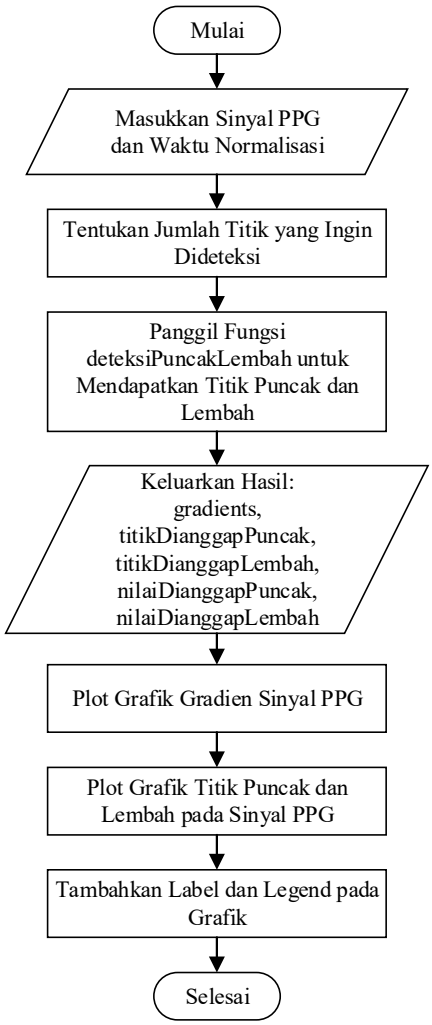
8. Tampilkan SNR Sebelum dan Sesudah

   Menampilkan atau menyimpan hasil perhitungan SNR sebelum dan sesudah *smoothing* sebagai hasil akhir dari proses ini.

9. Cek SNR Setelah *Smoothing*

Mengecek nilai SNR>20 dB. Jika kondisi terpenuhi, sinyal dapat digunakan untuk tahap selanjutnya, jika kondisi tidak terpenuhi, atur ulang parameter pra-pemrosesan.

3.5 Deteksi Puncak Lembah



Gambar 3.15 *Flowchart* Deteksi Puncak Lembah

Proses deteksi puncak lembah melibatkan beberapa langkah Gambar 3.15, yang dijelaskan sebagai berikut:

1. Masukkan Sinyal PPG dan Waktu Normalisasi

Masukkan sinyal dan waktu normalisasi untuk proses deteksi puncak dan lembah.

2. Menentukan Jumlah Titik yang Ingin Dideteksi

   Menentukan jumlah titik yang akan dianalisis pada sinyal PPG penting untuk membatasi titik yang dianalisis secara optimal. Jumlah titik tidak boleh terlalu sedikit agar semua titik terdeteksi, namun juga tidak terlalu banyak agar proses tidak memakan waktu lama. Jika jumlah titik terlalu besar, proses pencarian akan terus mengulang pada data terakhir hingga seluruh sampel selesai diproses.

3. Panggil Fungsi *deteksiPuncakLembah*

   Gunakan fungsi *deteksiFuncakLembah* untuk mendeteksi puncak dan lembah pada sinyal PPG normalisasi. Hasil yang diperoleh berupa variabel seperti *lembahPPG, lembahPPGIndex, puncakPPG, dan puncakPPGIndex*.

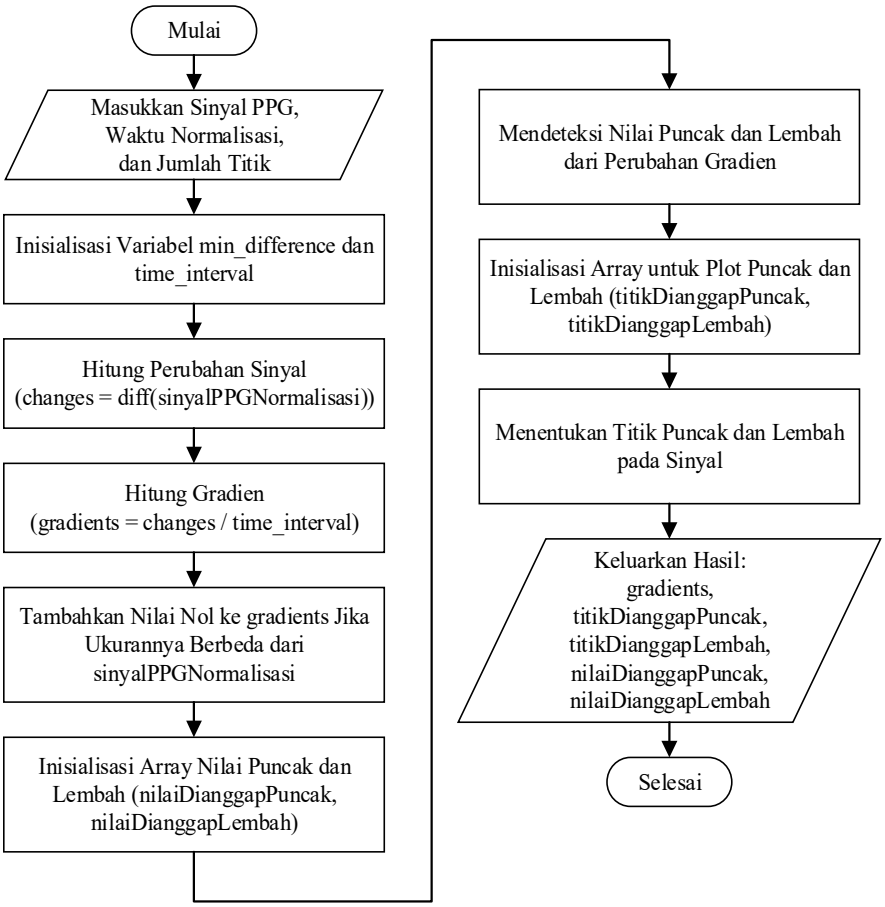
4. Keluarkan Hasil

   Hasil yang diperoleh berupa variabel seperti *gradients, titikDianggapPuncak, titikDianggapLembah, nilaiDianggapPuncak, nilaiDianggapLembah*.

5. Plot Hasil

   Visualisasikan hasil deteksi puncak dan lembah dengan menampilkan grafik gradien sinyal PPG yang menunjukkan perubahan gradien terhadap waktu. Selanjutnya, buat grafik utama sinyal PPG yang telah dinormalisasi, termasuk titik puncak dan lembah. Terakhir, tambahkan judul, label sumbu waktu dan amplitudo, serta legenda untuk membedakan puncak dan lembah, sehingga grafik menjadi lebih informatif dan mudah dipahami.

3.5.1 Fungsi *deteksiPuncakLembah*



Gambar 3.16 *Flowchart* Fungsi deteksiPuncakLembah

Fungsi *deteksiPuncakLembah* yang digunakan dalam proses deteksi puncak dan lembah melibatkan beberapa langkah Gambar 3.16, yang dijelaskan sebagai berikut:

1. Masukkan Sinyal PPG dan Waktu Normalisasi serta Jumlah Titik

   Masukkan data *sinyalPPGNormalisasi*, *waktuNormalisasi*, dan *jumlah_titik*, yang akan digunakan dalam proses deteksi puncak dan lembah.

2. Inisialisasi Variabel

   Inisialisasi *min_difference* = 0.05 sebagai ambang batas minimum selisih antara puncak dan lembah, serta *time_interval* = 1 sebagai interval waktu.

Nilai *min_difference* dapat disesuaikan agar tetap lebih kecil dari selisih antara puncak sistolik dan diastolik, sesuai karakteristik sinyal. Variabel *min_difference* bertujuan untuk menghindari deteksi puncak kecil (selain puncak sistolik dan diastolik) yang mungkin lolos dari proses pra-pemrosesan.

3. Hitung Perubahan Sinyal (*changes*)

   Hitung perubahan antara setiap dua titik bertetangga dalam *sinyalPPGNormalisasi* menggunakan fungsi *diff*. Ini memberikan informasi tentang perubahan nilai sinyal pada setiap waktu.

4. Hitung Gradien

   Gradien dihitung dengan membagi *changes* dengan *time_interval* sebagaimana dijelaskan dalam persamaan (2.13). Gradien ini digunakan untuk menentukan titik puncak dan lembah.

5. Tambahkan Nilai Nol ke *Gradient*

   Jika ukuran *gradients* lebih kecil dari *sinyalPPGNormalisasi*, tambahkan nilai nol untuk menyamakan ukuran.

6. Inisialisasi *Array* Nilai Puncak dan Lembah

   Buat *array nilaiDianggapPuncak* dan *nilaiDianggapLembah* untuk menyimpan nilai puncak dan lembah yang terdeteksi selama proses.

7. Mendeteksi Nilai Puncak dan Lembah dari Perubahan Gradien

   Ketika gradien berbalik dari positif ke negatif, maka titik puncak terdeteksi. Ketika gradien berbalik dari negatif ke positif, maka titik lembah terdeteksi.

8. Inisialisasi *Array* untuk Plot Puncak dan Lembah

   Buat *array* untuk menyimpan posisi titik-titik yang dianggap sebagai puncak dan lembah untuk keperluan visualisasi atau analisis selanjutnya.

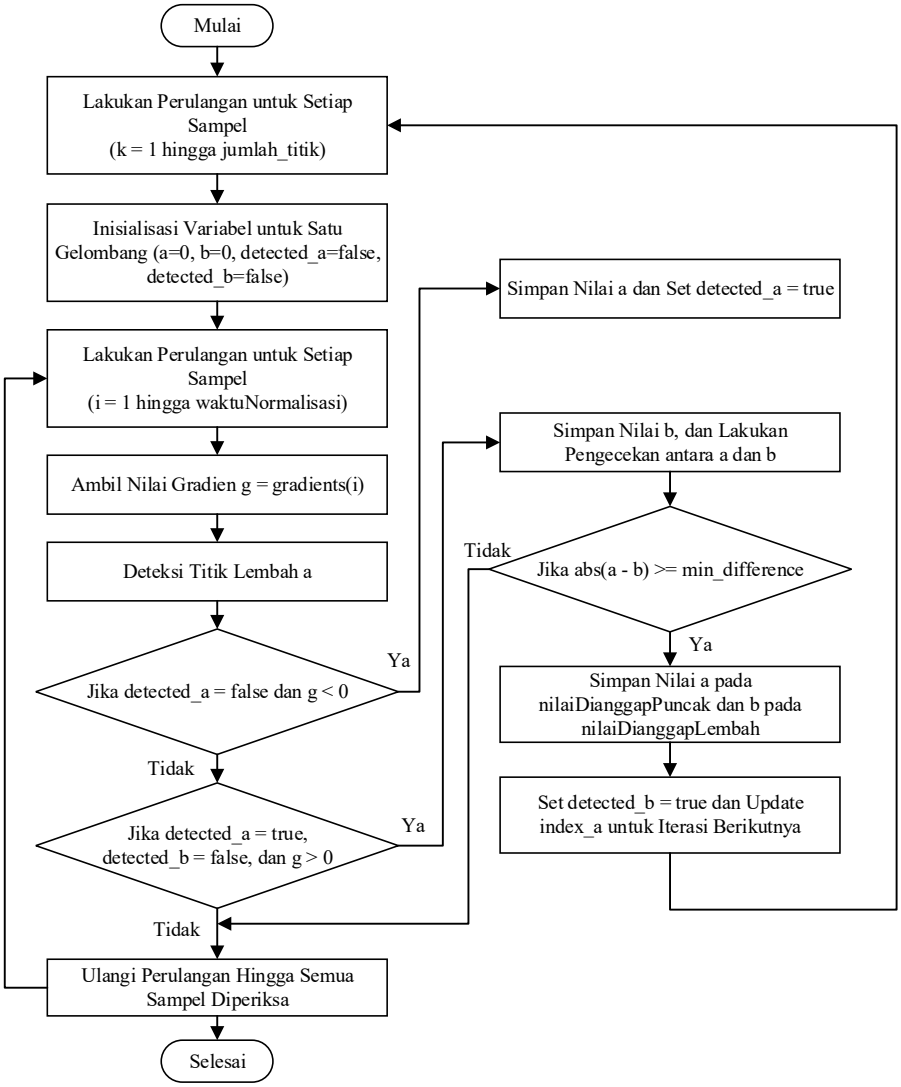
9. Menentukan Titik Puncak dan Lembah pada Sinyal

   Mencari nilai puncak dan lembah pada sinyal PPG yang telah dinormalisasi untuk menentukan titik puncak dan lembah pada sinyal.

10. Keluarkan Hasil

    Hasil yang diperoleh berupa variabel seperti *gradients, titikDianggapPuncak, titikDianggapLembah, nilaiDianggapPuncak, nilaiDianggapLembah*.

3.5.1.1 Deteksi Nilai Puncak Lembah



Gambar 3.17 *Flowchart* Deteksi Nilai Puncak Lembah

Proses deteksi nilai puncak dan lembah yang digunakan dalam fungsi *deteksiPuncakLembah* melibatkan beberapa langkah Gambar 3.17, sebagai berikut:

1. Lakukan Perulangan untuk Setiap Sampel pada Sinyal PPG

   Perulangan dilakukan untuk mendeteksi sejumlah *jumlah_titik* dari puncak dan lembah. Ini dilakukan untuk setiap titik dalam sinyal PPG.

2. Inisialisasi Variabel

   Tetapkan variabel untuk mendeteksi puncak (a) dan lembah (b) dari sinyal dengan nilai awal (*a = 0, b = 0, detected_a = false, detected_b = false*).

3. Lakukan Perulangan untuk Setiap Sampel pada Sinyal PPG

   Perulangan dilakukan untuk memproses setiap titik data, mulai dari *i = 1* hingga *waktuNormalisasi*.

4. Ambil Nilai Gradien

   Ambil nilai gradien *g* pada titik indeks *i* untuk menentukan perubahan arah sinyal.

5. Deteksi Titik Lembah (*a*)

   Jika *detected_a = false* dan *g < 0*, artinya titik lembah ditemukan. Simpan nilai lembah ini sebagai *a* dan setel *detected_a = true*.

6. Deteksi Titik Puncak (*b*)

   Jika *detected_a = true, detected_b = false*, dan *g > 0*, artinya titik puncak ditemukan. Simpan nilai puncak ini sebagai *b*.

7. Pengecekan Selisih antara *a* dan *b*

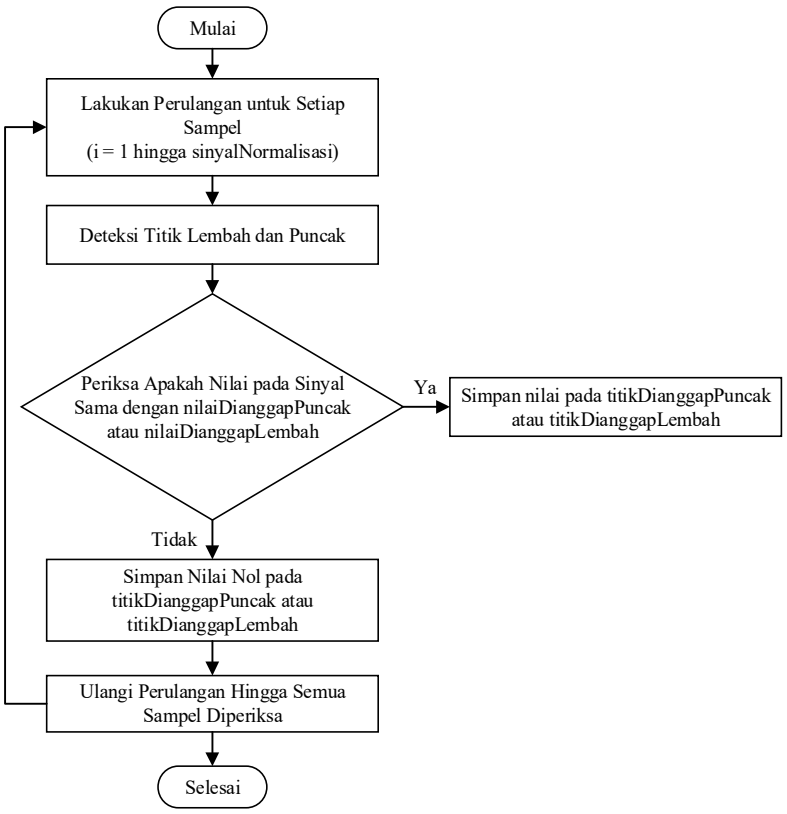
   Periksa apakah perbedaan absolut antara *a* dan *b* lebih besar atau sama dengan min_difference. Jika ya, simpan nilai *a* dalam *nilaiDianggapPuncak* dan *b*

dalam *nilaiDianggapLembah*, *set detected_b = true*, dan perbarui *index_a* untuk iterasi berikutnya. Keluar dari loop untuk lanjut ke deteksi gelombang berikutnya.

8. Ulangi Perulangan hingga Semua Sampel Diperiksa

   Mengulangi langkah-langkah dalam perulangan ini sampai semua sampel dalam sinyal selesai dianalisis.

3.5.1.2 Deteksi Titik Puncak Lembah



Gambar 3.18 *Flowchart* Deteksi Titik Puncak Lembah

Proses menentukan titik puncak dan lembah yang digunakan dalam fungsi *deteksiPuncakLembah* melibatkan beberapa langkah Gambar 3.18, sebagai berikut:

1. Lakukan Perulangan untuk Setiap Sampel pada Sinyal PPG

   Melakukan perulangan untuk memproses setiap sampel dalam sinyal PPG, dari *I = 1* hingga *sinyalPPGNormalisasi*.

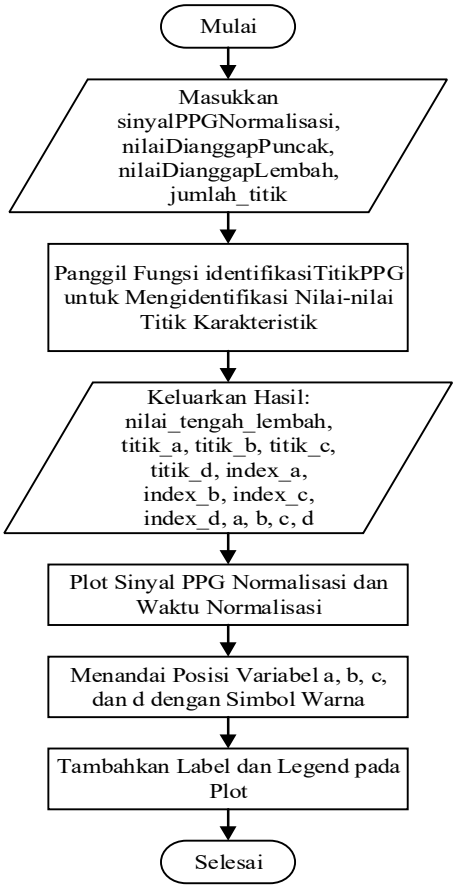
2. Deteksi Titik Lembah dan Puncak

   Jika nilai sinyal sama dengan nilai dalam *nilaiDianggapPuncak* atau *nilaiDianggapLembah*. Jika kondisi terpenuhi, simpan titik tersebut ke dalam *array titikDianggapPuncak* atau *titikDianggapLembah*. Jika kondisi tidak terpenuhi, simpan nilai nol pada *array titikDianggapPuncak* atau *titikDianggapLembah*.

3. Ulangi Perulangan hingga Semua Sampel Diperiksa

   Mengulangi langkah-langkah dalam perulangan ini sampai semua sampel dalam sinyal selesai dianalisis.

3.6 Identifikasi Titik Lembah dan Puncak



Gambar 3.19 *Flowchart* Identifikasi Titik Lembah dan Puncak

Proses identifikasi ttitik puncak dan lembah melibatkan beberapa langkah Gambar 3.19, yang dijelaskan sebagai berikut:

1. Masukkan Data yang Akan Digunakan

   Masukkan *sinyalPPGNormalisasi,waktuNormalisasi, nilaiDianggapPuncak, nilaiDianggapLembah* untuk proses identifikasi titik PPG.

2. Panggil Fungsi identifikasiTitikPPG

   Gunakan fungsi *identifikasiTitikPPG* untuk identifikasi titik pada sinyal PPG normalisasi. Hasil yang diperoleh berupa variabel seperti *nilai_tengah_lembah, titik_a, titik_b, titik_c, titik_d, index_a, index_b, index_c, index_d, a, b, c,* dan *d*.

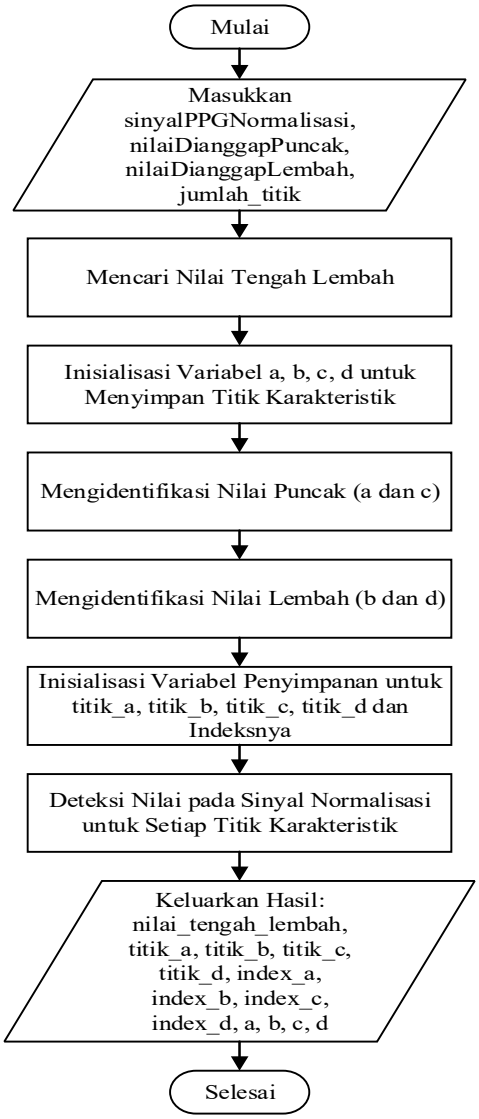
3. Keluarkan Hasil

   Hasil yang diperoleh berupa variabel seperti *nilai_tengah_lembah, titik_a, titik_b, titik_c, titik_d, index_a, index_b, index_c, index_d, a, b, c*, dan *d*.

4. Plot Hasil

   Visualisasikan hasil identifikasi titik PPG dengan menandai posisi variabel *a, b, c*, dan *d*. Terakhir, tambahkan judul, label sumbu waktu dan amplitudo, serta legenda untuk membedakan setiap titik, sehingga grafik menjadi lebih informatif dan mudah dipahami.

3.6.1 Fungsi *identifikasiTitikPPG*



Gambar 3.20 *Flowchart* Fungsi identifikasiTitikPPG

Fungsi *identifikasiTitikPPG* Gambar 3.20 yang digunakan dalam proses identifikasi titik PPG melibatkan beberapa langkah berikut:

1. Masukkan Sinyal PPG dan Waktu Normalisasi serta Jumlah Titik

   Masukkan data *sinyalPPGNormalisasi, waktuNormalisasi*, dan *jumlah_titik*, yang akan digunakan dalam proses identifikasi titik PPG.

2. Mencari Nilai Tengah Lembah

   Menghitung nilai tengah lembah dari sinyal sebagai referensi untuk memisahkan lembah *onset* dan lembah *dicrotic notch* dalam satu siklus.

3. Inisialisasi Variabel a, b, c, d untuk Menyimpan Titik Karakteristik

   Variabel a, b, c, dan d diinisialisasi untuk menyimpan nilai dan indeks dari titik karakteristik yang terdeteksi. Variabel ini akan digunakan dalam proses identifikasi puncak dan lembah.

4. Mengidentifikasi Nilai Puncak (a dan c)

   Puncak a dan c dideteksi dengan menggunakan nilai ambang. Nilai ini digunakan sebagai referensi untuk memisahkan puncak sistolik dan puncak diastolik.

5. Mengidentifikasi Nilai Lembah (b dan d)

   Lembah b dan d dideteksi dengan menggunakan nilai tengah. Nilai ini digunakan sebagai referensi untuk memisahkan lembah *onset* dan lembah *dicrotic notch*.

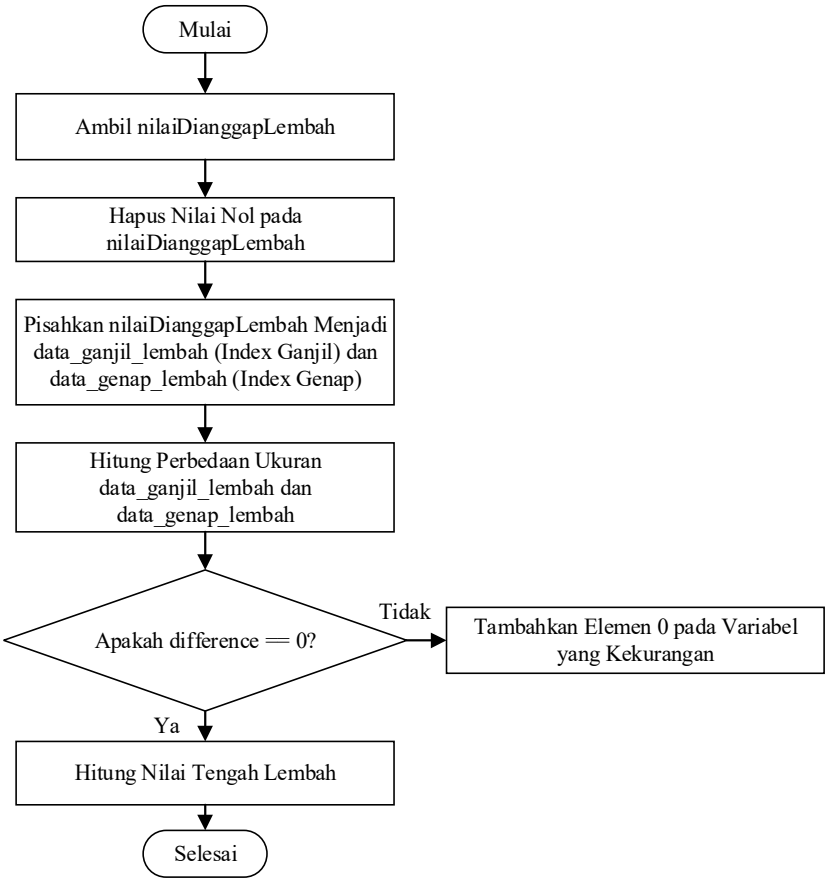
6. Inisialisasi Variabel Penyimpanan titik dan Indeksnya

   Membuat variabel penyimpanan untuk mencatat nilai dari masing-masing titik karakteristik (a, b, c, d) serta indeksnya. Variabel ini akan digunakan untuk menyimpan hasil deteksi dengan struktur yang terorganisir.

7. Keluarkan Hasil

   Hasil yang diperoleh berupa variabel seperti *nilai_tengah_lembah, titik_a, titik_b, titik_c, titik_d, index_a, index_b, index_c, index_d, a, b, c, d*.

3.6.1.1 Mencari Nilai Tengah Lembah



Gambar 3.21 *Flowchart* Mencari Nilai Tengah Puncak

Proses mencari nilai tengah lembah yang digunakan dalam proses identifikasi titik PPG melibatkan beberapa langkah Gambar 3.21, sebagai berikut:

1. Ambil *nilaiDianggapLembah*

   Ambil *nilaiDianggapLembah* untuk menghitung nilai tengah.

2. Hapus nilai nol pada *nilaiDianggapLembah*

   Menghapus nilai nol pada *nilaiDianggapLembah* suapaya tersisa data nilai lembahnya.

3. Pisahkan *nilaiDianggapLembah*

   Lakukan pembagian *nilaiDianggapLembah* menjadi *data_ganjil_lembah* dan *data_genap_lembah* untuk menghitung nilai tengah lembah.

4. Hitung Perbedaan Ukuran

   Menghitung selisih ukuran *data_ganjil_lembah* dan *data_genap_lembah*.

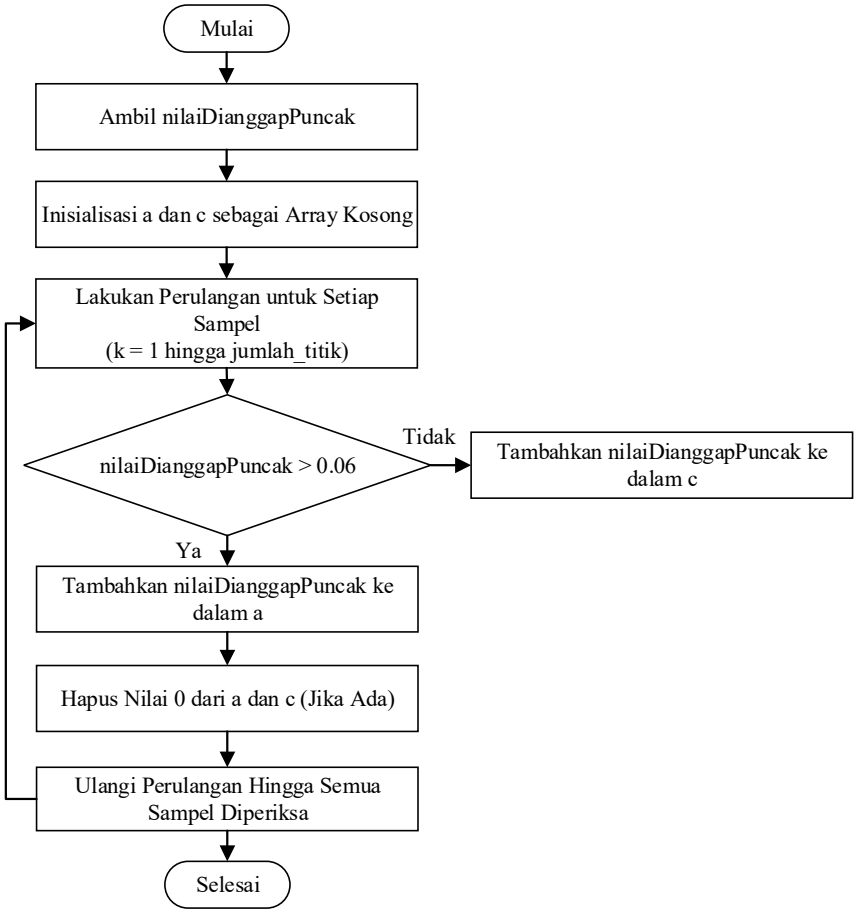
5. Cek *difference == 0*

   Jika ukuran *data_ganjil_lembah* lebih kecil atau lebih besar dari *data_genap_lembah*, tambahkan nilai nol untuk menyamakan ukuran.

6. Hitung Nilai Tengah Lembah

   Nilai tengah lembah dihitung dengan menjumlahkan *data_ganjil_lembah* dan *data_genap_lembah*, kemudian hasil penjumlahan tersebut dibagi 2.

3.6.1.2 Mengidentifikasi Nilai Puncak (a dan c)



Gambar 3.22 *Flowchart* Mengidentifikasi Nilai Puncak

Proses mengidentifikasi nilai puncak (*a* dan *c*) yang digunakan dalam proses identifikasi titik PPG melibatkan beberapa langkah Gambar 3.22, sebagai berikut:

1. Ambil *nilaiDianggapPuncak*

   Ambil *nilaiDianggapPuncak* untuk mengidentifikasi nilai puncak (a dan c).

2. Inisialisasi *a* dan *c* sebagai *array* kosong

   Menginisialisasi variabel *a* dan *c* sebagai *array* kosong, untuk diisi nilai dari puncak sistolik (a) dan puncak diastolik (c).

3. Lakukan Perulangan untuk Setiap Sampel

   Perulangan dilakukan untuk mengecek setiap elemen dalam *nilaiDianggapPuncak*, mulai dari indeks *k = 1* hingga *jumlah_titik*.

4. Cek *nilaiDianggapPuncak > 0.06*

   Periksa nilai *nilaiDianggapPuncak* pada indeks yang lebih besar dari 0.06, yang digunakan sebagai ambang batas (*threshold*) untuk memisahkan puncak sistolik dan diastolik. Jika kondisi terpenuhi, simpan *nilaiDianggapPuncak* ke dalam *array a*. Jika kondisi tidak terpenuhi, simpan *nilaiDianggapPuncak* ke dalam *array c*. Nilai *threshold* ini dapat disesuaikan dengan karakteristik sinyal PPG. Penetapan nilai *threshold* dilakukan melalui pengecekan visual sinyal PPG dengan mengidentifikasi puncak diastolik tertinggi. Setelah ditemukan, atur nilai *threshold* di atas puncak diastolik tertinggi namun di bawah puncak sistolik terendah.

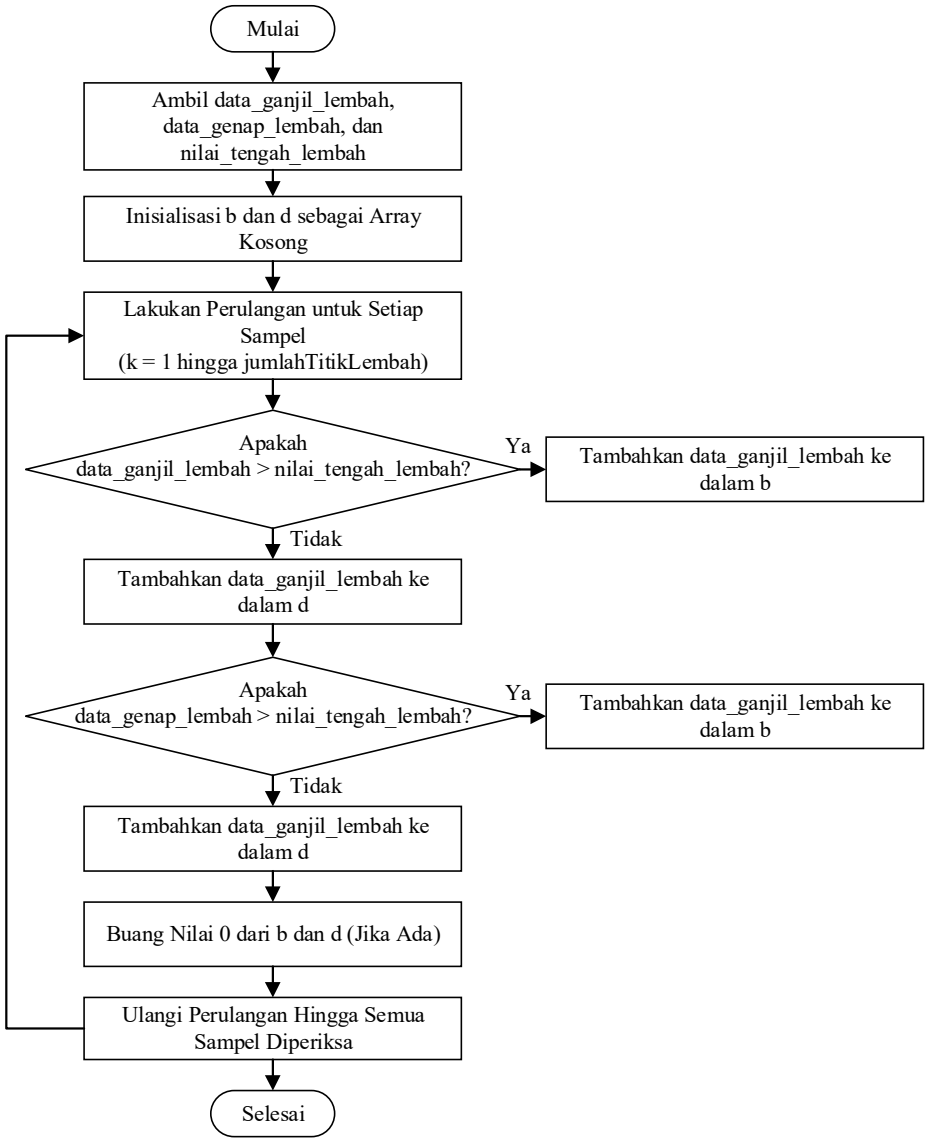
5. Hapus nilai 0 dari *a* dan *c* (Jika Ada)

   Jika terdapat nilai nol pada *a* dan *c*, maka nilai nol tersebut harus dihapus.

6. Ulangi Perulangan hingga Semua Sampel Diperiksa

   Mengulangi langkah-langkah dalam perulangan ini sampai semua sampel dalam sinyal selesai dianalisis.

3.6.1.3 Mengidentifikasi Nilai Lembah (b dan d)



Gambar 3.23 *Flowchart* Mengidentifikasi Nilai Lembah

Proses mengidentifikasi nilai lembah (*b* dan *d*) yang digunakan dalam proses identifikasi titik PPG melibatkan beberapa langkah Gambar 3.23, sebagai berikut:

1. Ambil Data dari Proses Mencari Nilai Tengah Lembah

   Ambil *data_ganjil_lembah*, *data_genap_lembah*, dan *nilai_tengah_lembah* untuk mengidentifikasi nilai lembah (*b* dan *d*).

2. Inisialisasi *b* dan *d* sebagai *array* kosong

   Menginisialisasi variabel *b* dan *d* sebagai *array* kosong, untuk diisi nilai dari titik *dicrotic notch (b)* dan titik *onset (d)*.

3. Lakukan Perulangan untuk Setiap Sampel

   Perulangan dilakukan untuk mengecek setiap elemen dalam *nilaiDianggapPuncak*, mulai dari indeks *k = 1* hingga *jumlahTitikLembah*.

4. Cek *data_ganjil_lembah* > *nilai_tengah_lembah*

   Periksa nilai *data_ganjil_lembah* pada indeks yang lebih besar dari *nilai_tengah_lembah*. Jika kondisi terpenuhi, simpan *data_ganjil_lembah* ke dalam *array b*. Jika kondisi tidak terpenuhi, simpan *data_ganjil_lembah* ke dalam *array d*.

5. Cek *data_genap_lembah* > *nilai_tengah_lembah*

   Periksa nilai *data_genap_lembah* pada indeks yang lebih besar dari *nilai_tengah_lembah*. Jika kondisi terpenuhi, simpan *data_genap_lembah* ke dalam *array b*. Jika kondisi tidak terpenuhi, simpan *data_genap_lembah* ke dalam *array d*.

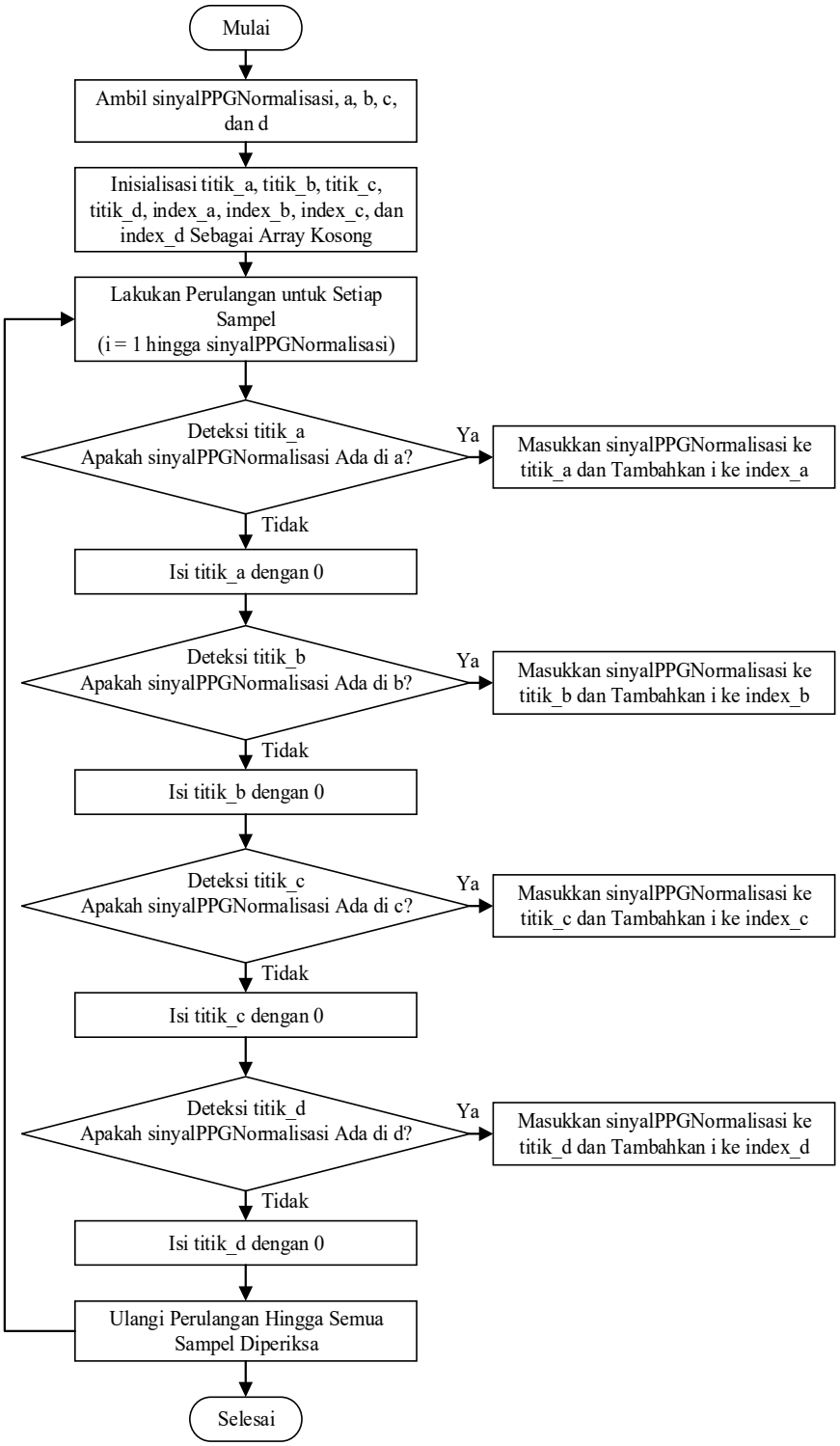
6. Hapus nilai 0 dari *b* dan *d* (Jika Ada)

   Jika terdapat nilai nol pada *a* dan *c*, maka nilai nol tersebut harus dihapus.

7. Ulangi Perulangan hingga Semua Sampel Diperiksa

   Mengulangi langkah-langkah dalam perulangan ini sampai semua sampel dalam sinyal selesai dianalisis.

3.6.1.4 Proses Deteksi Setiap Titik Karakteristik pada Sinyal Normalisasi



Gambar 3.24 *Flowchart* Deteksi Setiap Titik Karakteristik

Proses deteksi nilai setiap titik karakteristik pada sinyal normalisasi yang digunakan dalam proses identifikasi titik PPG melibatkan beberapa langkah Gambar 3.24, yang dijelaskan sebagai berikut:

1. Ambil *sinyalPPGNormalisasi* dan titik teridentifikasi

   Ambil *sinyalPPGNormalisasi*, *a, b, c*, dan *d* untuk mendeteksi setiap titik karakteristik pada sinyal normalisasi.

2. Inisialisasi Variabel

   Menginisialisasi variabel *titik_a, titik_b, titik_c, titik_d, index_a, index_b, index_c,* dan *index_d* sebagai *array* kosong.

3. Lakukan Perulangan untuk Setiap Sampel

   Perulangan dilakukan untuk mendeteksi setiap titik, mulai dari indeks *i = 1* hingga *sinyalPPGNormalisasi*.

4. Deteksi *titik_a*

   Cek apakah *a* ada di *sinyalPPGNormalisasi*, jika ya, masukkan *sinyalPPGNormalisasi* ke *titik_a* dan tambahkan *i* ke *index_a*. Jika tidak, isi *titik_a* dengan 0.

5. Deteksi *titik_b*

   Cek apakah *b* ada di *sinyalPPGNormalisasi*, jika ya, masukkan *sinyalPPGNormalisasi* ke *titik_b* dan tambahkan *i* ke *index_b*. Jika tidak, isi *titik_b* dengan 0.

6. Deteksi *titik_c*

   Cek apakah *c* ada di *sinyalPPGNormalisasi*, jika ya, masukkan *sinyalPPGNormalisasi* ke *titik_c* dan tambahkan *i* ke *index_c*. Jika tidak, isi *titik_c* dengan 0.

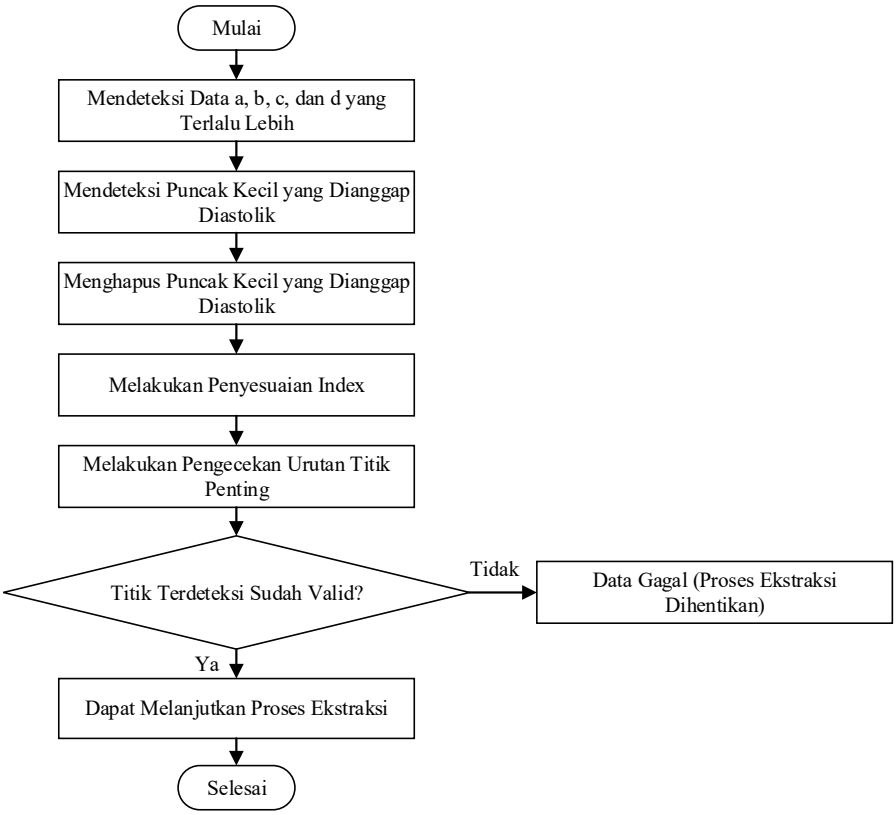
7. Deteksi *titik_d*

   Cek apakah d ada di *sinyalPPGNormalisasi*, jika ya, masukkan *sinyalPPGNormalisasi* ke *titik_d* dan tambahkan *i* ke *index_d*. Jika tidak, isi *titik_d* dengan 0.

8. Ulangi Perulangan hingga Semua Sampel Diperiksa

   Mengulangi langkah-langkah dalam perulangan ini sampai semua sampel dalam sinyal selesai dianalisis.

3.7 Validasi Titik Terdeteksi



Gambar 3.25 *Flowchart* Validasi Titik Terdeteksi

Proses validasi titik terdeteksi melibatkan beberapa langkah Gambar 3.25, yang dijelaskan sebagai berikut:

1. Mendeteksi Kelebihan Data

Data untuk titik a, b, c, dan d diperiksa untuk mengidentifikasi kelebihan yang mungkin muncul akibat puncak kecil yang dianggap sebagai titik diastolik.

2. Mendeteksi Puncak Kecil yang Dianggap Diastolik

   Titik yang tidak sesuai dipersiapkan untuk dihapus pada langkah berikutnya.

3. Menghapus Puncak Kecil yang Dianggap Diastolik

   Data yang telah diidentifikasi untuk eliminasi kemudian dihapus, sehingga hanya data valid yang tersisa.

4. Melakukan Penyesuaian Index

   Indeks titik terdeteksi diselaraskan agar urutannya dimulai dari titik d. Penyesuaian ini bertujuan untuk mendukung proses ekstraksi, perhitungan amplitudo, dan analisis waktu yang lebih akurat.

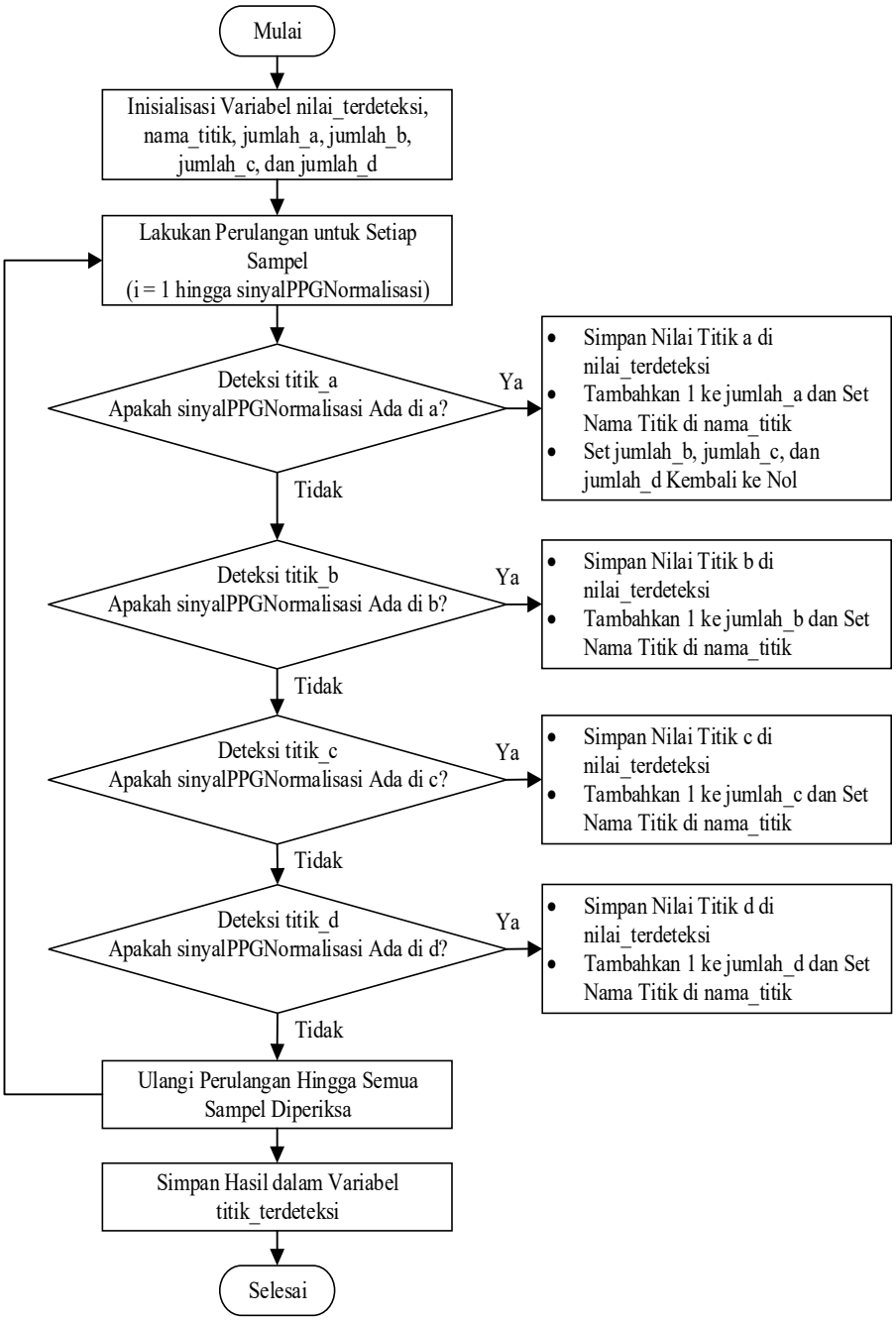
5. Memeriksa Urutan Titik Penting

   Setelah semua puncak kecil yang keliru diidentifikasi dan indeks disesuaikan, sistem memeriksa urutan titik penting (*a, b, c, d*) guna memastikan data berada dalam urutan yang benar.

6. Memvalidasi Titik Terdeteksi

   Jika pengecekan urutan menunjukkan ketidaksesuaian, data seleksi, dan proses ekstraksi dihentikan. Namun, jika urutan data benar, proses ekstraksi dilanjutkan.

3.7.1 Deteksi Titik a, b, c, dan d yang Berlebih



Gambar 3.26 *Flowchart* Deteksi Titik a, b, c, dan d yang Berlebih

Proses deteksi titik *a, b, c*, dan *d* yang berlebih dalam validasi titik terdeteksi melibatkan langkah-langkah Gambar 3.26, sebagai berikut:

1. Inisialisasi Variabel

   Menginisialisasi variabel seperti *nilai_terdeteksi, nama_titik, jumlah_a, jumlah_b, jumlah_c,* dan *jumlah_d* untuk melacak titik yang terdeteksi.

2. Lakukan Perulangan untuk Setiap Sampel

   Melakukan perulangan pada setiap sampel sinyal mulai dari indeks *i = 1* hingga akhir data *sinyalPPGNormalisasi*.

3. Deteksi *titik_a*

   Periksa keberadaan titik *a* pada *sinyalPPGNormalisasi*. Jika terdeteksi, lakukan langkah berikut:

   a. Atur ulang *idx* ke 0 untuk memulai gelombang baru.

   b. Simpan nilai titik *a* ke dalam *nilai_terdeteksi*.

   c. Tambahkan 1 ke *jumlah_a* dan perbarui *nama_titik*.

   d. Setel *jumlah_b, jumlah_c,* dan *jumlah_d* kembali ke nol.

   Jika kondisi tidak terpenuhi, lanjutkan ke sampel berikutnya.

4. Deteksi *titik_b*

   Periksa keberadaan titik *b* pada *sinyalPPGNormalisasi*. Jika terdeteksi, lakukan langkah berikut:

   a. Simpan nilai titik *b* ke dalam *nilai_terdeteksi*.

   b. Tambahkan 1 ke *jumlah_b* dan perbarui *nama_titik*.

   Jika kondisi tidak terpenuhi, lanjutkan ke sampel berikutnya.

5. Deteksi *titik_c*

   Periksa keberadaan titik *c* pada *sinyalPPGNormalisasi*. Jika terdeteksi, lakukan langkah berikut:

a. Simpan nilai titik *c* ke dalam *nilai_terdeteksi*.

b. Tambahkan 1 ke *jumlah_c* dan perbarui *nama_titik*.

Jika kondisi tidak terpenuhi, lanjutkan ke sampel berikutnya.

6. Deteksi *titik_d*

   Periksa keberadaan titik *d* pada *sinyalPPGNormalisasi*. Jika terdeteksi, lakukan langkah berikut:

   a. Simpan nilai titik *d* ke dalam *nilai_terdeteksi*.

   b. Tambahkan 1 ke *jumlah_d* dan perbarui *nama_titik*.

   Jika kondisi tidak terpenuhi, lanjutkan ke sampel berikutnya.

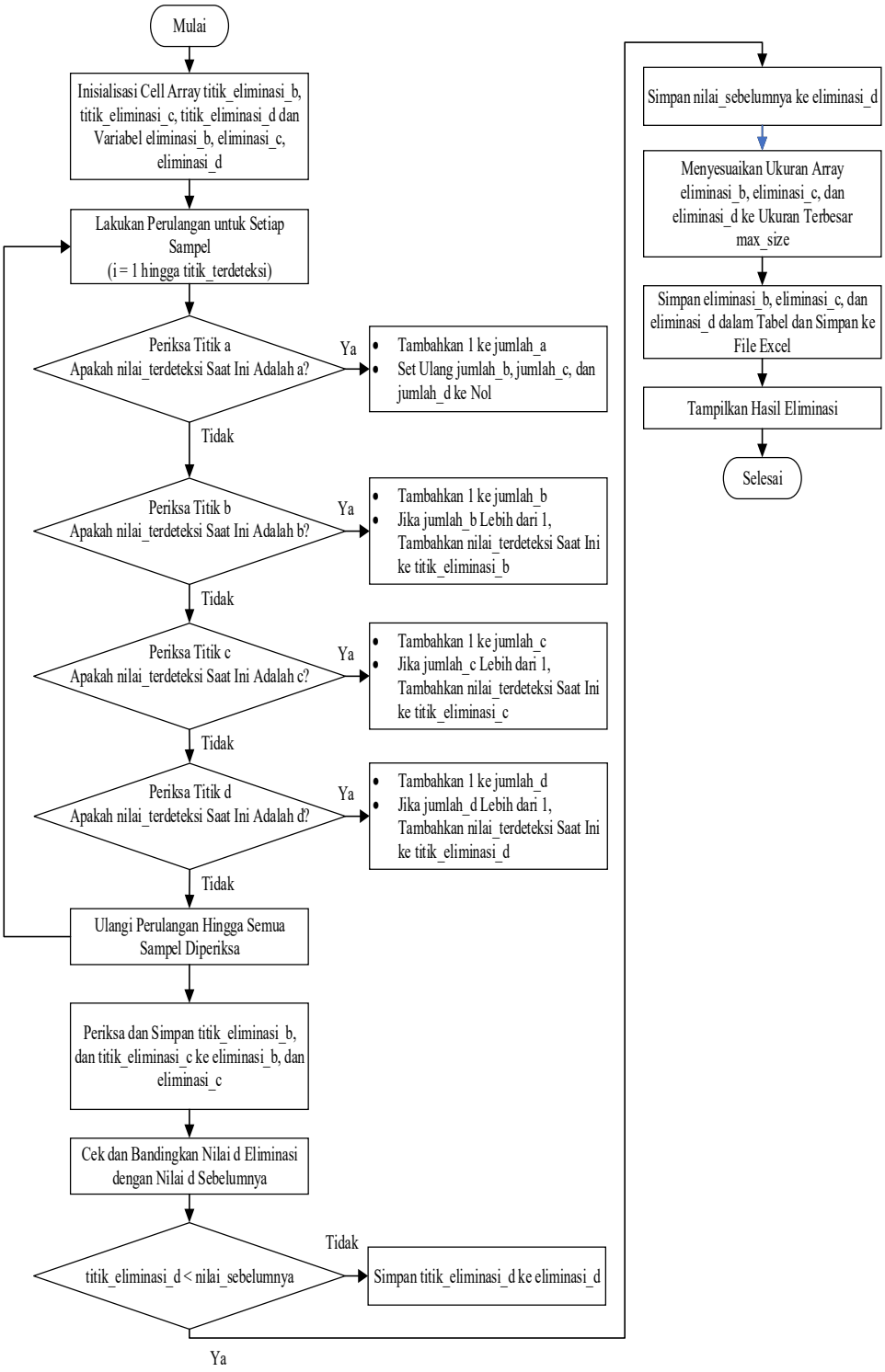
7. Ulangi Perulangan hingga Semua Sampel Diperiksa

   Ulangi seluruh langkah deteksi pada setiap sampel hingga semua data dalam *sinyalPPGNormalisasi* selesai dianalisis.

8. Simpan Hasil dalam Variabel *titik_terdeteksi*

   Simpan hasil deteksi sebagai tabel *titik_terdeteksi*, yang mencakup kolom *nilai_terdeteksi* dan *nama_titik*.

3.7.2 Mendeteksi Puncak Kecil yang Dianggap Diastolik



Gambar 3.27 *Flowchart* Mendeteksi Puncak Kecil yang Dianggap Diastolik

Proses deteksi puncak kecil yang dianggap diastolik dalam validasi titik terdeteksi melibatkan langkah-langkah Gambar 3.27, sebagai berikut:

1. Inisialisasi Variabel

   Menginisialisasi variabel seperti *eliminasi_b, eliminasi_c, eliminasi_d, serta cell array titik_eliminasi_b, titik_eliminasi_c,* dan *titik_eliminasi_d* untuk melacak puncak kecil yang terdeteksi.

2. Perulangan untuk Setiap Sampel

   Melakukan perulangan pada setiap sampel sinyal dari indeks *i = 1* hingga akhir data *sinyalPPGNormalisasi*.

3. Periksa Titik *a*

   Memeriksa keberadaan titik *a* pada *nilai_terdeteksi*. Jika ditemukan:

   a. Tambahkan 1 ke *jumlah_a*.

   b. Atur ulang *jumlah_b, jumlah_c, dan jumlah_d* ke nol untuk memulai analisis baru.

   Jika kondisi tidak terpenuhi, lanjutkan ke sampel berikutnya.

4. Periksa Titik *b*

   Memeriksa keberadaan titik *b* pada *nilai_terdeteksi*. Jika ditemukan:

   a. Tambahkan 1 ke *jumlah_b*.

   b. Jika *jumlah_b* lebih dari 1, tambahkan nilai *nilai_terdeteksi* saat ini ke *titik_eliminasi_b*.

   Jika kondisi tidak terpenuhi, lanjutkan ke sampel berikutnya.

5. Periksa Titik *c*

   Memeriksa keberadaan titik *c* pada *nilai_terdeteksi*. Jika ditemukan:

   a. Tambahkan 1 ke *jumlah_c*.

b. Jika *jumlah_c* lebih dari 1, tambahkan nilai *nilai_terdeteksi* saat ini ke *titik_eliminasi_c*.

Jika kondisi tidak terpenuhi, lanjutkan ke sampel berikutnya.

6. Periksa Titik *d*

Memeriksa keberadaan titik *d* pada *nilai_terdeteksi*. Jika ditemukan:

a. Tambahkan 1 ke *jumlah_d*.

b. Jika *jumlah_d* lebih dari 1, tambahkan nilai *nilai_terdeteksi* saat ini ke *titik_eliminasi_d*.

Jika kondisi tidak terpenuhi, lanjutkan ke sampel berikutnya.

7. Lanjutkan Perulangan hingga Semua Sampel Diperiksa

Ulangi seluruh langkah deteksi pada setiap sampel hingga semua data pada *sinyalPPGNormalisasi* selesai diperiksa.

8. Bandingkan dan Simpan Nilai *d* pada Eliminasi dengan Nilai Sebelumnya

Periksa *titik_eliminasi_b* dan *titik_eliminasi_c* serta simpan ke dalam *eliminasi_b* dan *eliminasi_c*.

9. Periksa Kondisi *titik_eliminasi_d* < *nilai_sebelumnya*

Periksa nilai *titik_eliminasi_d* pada indeks yang lebih kecil dari *nilai_sebelumnya*. Jika kondisi terpenuhi, simpan *nilai_sebelumnya* ke dalam *eliminasi_d*. Jika tidak terpenuhi, simpan *titik_eliminasi_d* ke dalam *eliminasi_d*.

10. Sesuaikan Ukuran *Array*

Menyesuaikan ukuran *array eliminasi_b, eliminasi_c*, dan *eliminasi_d* agar sesuai dengan ukuran terbesar (*max_size*).

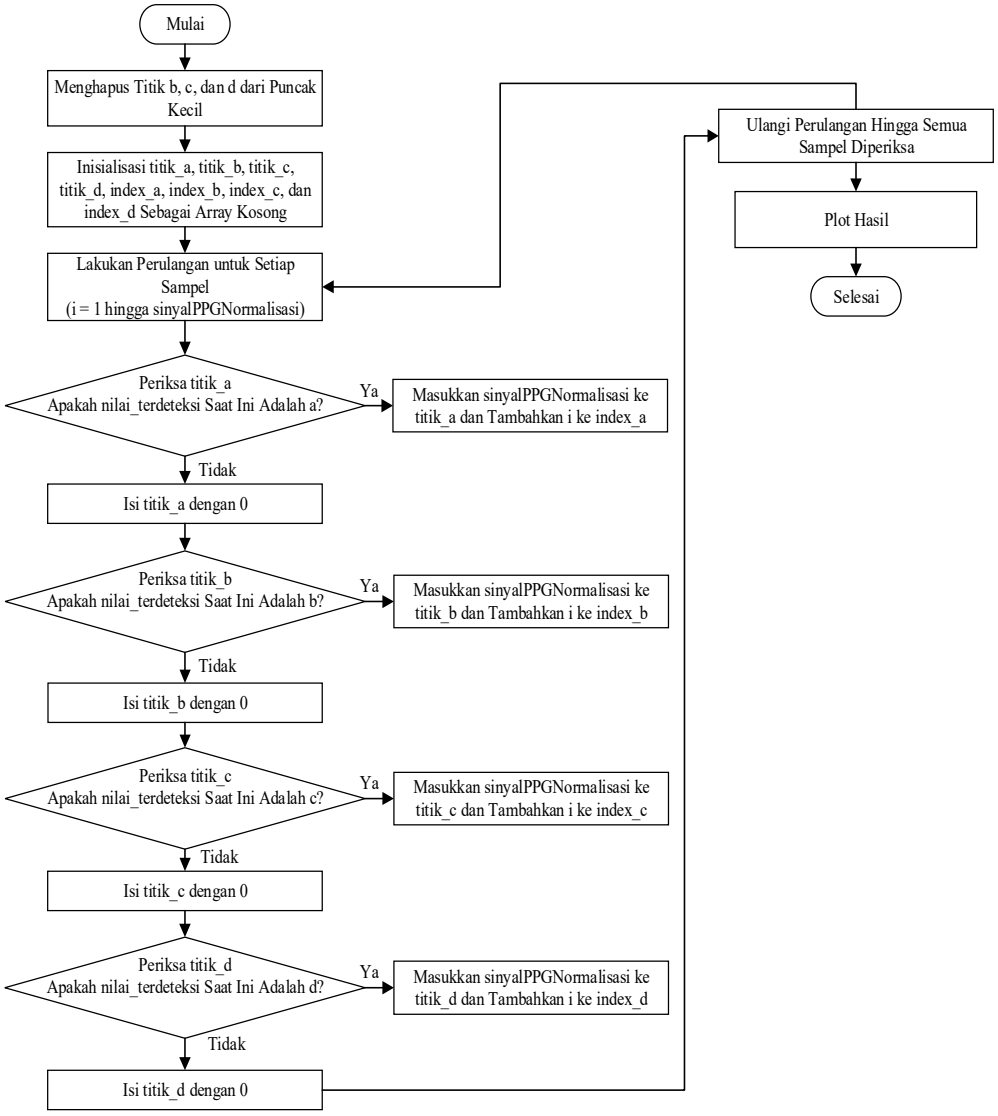
11. Simpan Hasil ke Tabel dan Ekspor ke Excel

    Simpan hasil deteksi sebagai tabel *errorDeteksiTitik*, yang mencakup kolom *eliminasi_b, eliminasi_c*, dan *eliminasi_d*, kemudian simpan tabel ini ke file Excel.

12. Tampilkan Hasil Eliminasi

    Menampilkan hasil deteksi dan eliminasi puncak kecil sebagai *output*.

3.7.3 Menghapus Puncak Kecil yang Dianggap Diastolik



Gambar 3.28 *Flowchart* Menghapus Puncak Kecil Dianggap Diastolik

Proses menghapus puncak kecil yang dianggap diastolik dalam validasi titik terdeteksi melibatkan langkah-langkah Gambar 3.28, sebagai berikut:

1. Menghapus Titik *b, c*, dan d dari Puncak Kecil

   Menginisialisasi variabel seperti *eliminasi_b, eliminasi_c, eliminasi_d, serta cell array titik_eliminasi_b, titik_eliminasi_c,* dan *titik_eliminasi_d* untuk melacak puncak kecil yang terdeteksi.

2. Inisialisasi Variabel

   Inisialisasi *titik_a, titik_b, titik_c, titik_d, index_a, index_b, index_c,* dan *index_d* sebagai *array* kosong.

3. Perulangan untuk Setiap Sampel

   Melakukan perulangan pada setiap sampel sinyal dari indeks *i = 1* hingga akhir data *sinyalPPGNormalisasi*.

4. Periksa Titik *a*

   Memeriksa keberadaan titik *a* pada *nilai_terdeteksi*. Jika ditemukan, masukkan *sinyalPPGNormalisasi* ke *titik_a* dan tambahkan i ke *index_a*. Jika kondisi tidak terpenuhi, isi *titik_a* dengan nol.

5. Periksa Titik *b*

   Memeriksa keberadaan titik *b* pada *nilai_terdeteksi*. Jika ditemukan, masukkan *sinyalPPGNormalisasi* ke *titik_b* dan tambahkan i ke *index_b*. Jika kondisi tidak terpenuhi, isi *titik_b* dengan nol.

6. Periksa Titik *c*

   Memeriksa keberadaan titik *c* pada *nilai_terdeteksi*. Jika ditemukan, masukkan *sinyalPPGNormalisasi* ke *titik_c* dan tambahkan i ke *index_c*. Jika kondisi tidak terpenuhi, isi *titik_c* dengan nol.

7. Periksa Titik *d*

   Memeriksa keberadaan titik *d* pada *nilai_terdeteksi*. Jika ditemukan, masukkan *sinyalPPGNormalisasi* ke *titik_d* dan tambahkan i ke *index_d*. Jika kondisi tidak terpenuhi, isi *titik_d* dengan nol.

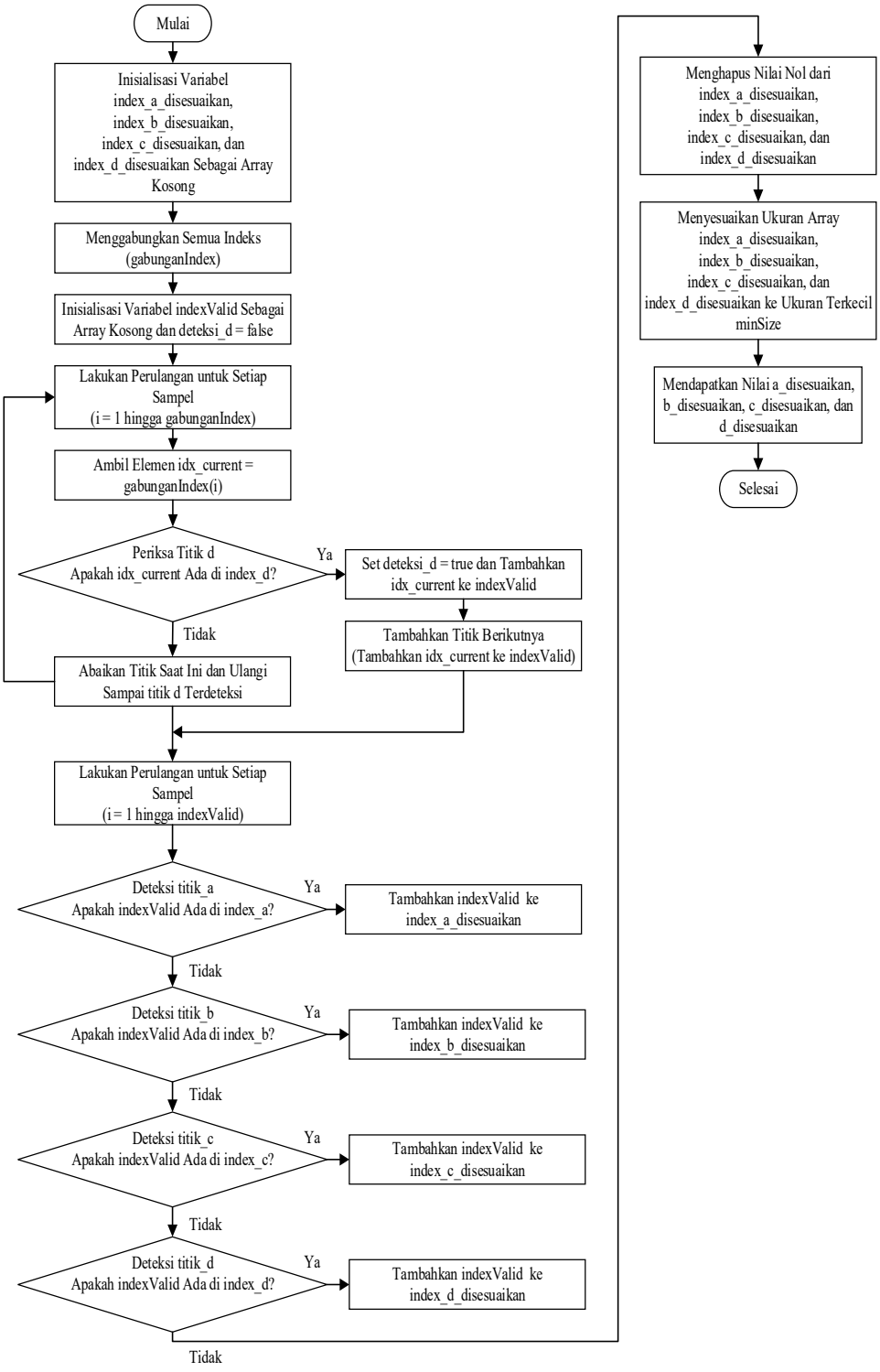
8. Lanjutkan Perulangan hingga Semua Sampel Diperiksa

   Ulangi seluruh langkah deteksi pada setiap sampel hingga semua data pada *sinyalPPGNormalisasi* selesai diperiksa.

9. Plot Hasil

   Visualisasikan sinyal PPG dan waktu normalisasi dengan menandai posisi variabel *a, b, c*, dan *d* yang sudah sesuai, dengan simbol warna. Terakhir, tambahkan judul, label sumbu waktu dan amplitudo, serta legenda untuk membedakan setiap titik, sehingga grafik menjadi lebih informatif dan mudah dipahami.

3.7.4 Melakukan Penyesuaian Index



Gambar 3.29 *Flowchart* Melakukan Penyesuaian Index

Proses penyesuaian titik dalam validasi titik terdeteksi melibatkan langkah-langkah Gambar 3.29, yang dijelaskan sebagai berikut:

1. Inisialisasi Variabel

   Menginisialisasi variabel seperti *index_a_disesuaikan*, *index_b_disesuaikan, index_c_disesuaikan,* dan *index_d_disesuaikan* sebagai *array* kosong.

2. Menggabungkan Semua Indeks (*gabunganIndex*)

   Menggabungkan seluruh indeks untuk digunakan dalam mendeteksi titik *d* awal.

3. Inisialisasi Variabel

   Menginisialisasi indexValid Sebagai *Array* Kosong dan *deteksi_d = false*.

4. Perulangan untuk Setiap Sampel

   Melakukan perulangan pada setiap sampel sinyal dari indeks *i = 1* hingga akhir data *gabunganIndex*.

5. Ambil elemen *idx_current = gabunganIndex*(i)

6. Periksa Titik *d*

   Memeriksa keberadaan *index_d* pada *idx_current.* Jika ditemukan, atur *deteksi_d = true d*an tambahkan *idx_current* ke *indexValid*, serta tambahkan titik berikutnya (tambahkan *idx_current* ke *indexValid*). Jika kondisi tidak terpenuhi, abaikan titik saat ini dan ulangi sampai titik d terdeteksi.

7. Perulangan untuk Setiap Sampel

   Melakukan perulangan pada setiap sampel sinyal dari indeks *i = 1* hingga akhir data *indexValid*.

8. Periksa Titik *a*

   Memeriksa keberadaan *index_a* pada *indexValid*. Jika ditemukan, tambahkan *indexValid* ke *index_a_disesuaikan*. Jika tidak terpenuhi, isi *titik_a* dengan nol.

9. Periksa Titik *b*

   Memeriksa keberadaan *index_b* pada *indexValid*. Jika ditemukan, tambahkan *indexValid* ke *index_b_disesuaikan*. Jika tidak terpenuhi, isi *titik_b* dengan nol.

10. Periksa Titik *c*

    Memeriksa keberadaan *index_c* pada *indexValid*. Jika ditemukan, tambahkan *indexValid* ke *index_c_disesuaikan*. Jika tidak terpenuhi, isi *titik_c* dengan nol.

11. Periksa Titik *d*

    Memeriksa keberadaan *index_d* pada *indexValid*. Jika ditemukan, tambahkan *indexValid* ke *index_d_disesuaikan*. Jika tidak terpenuhi, isi *titik_d* dengan nol.

12. Hapus Nilai Nol

    Menghapus nilai nol pada *index_a_disesuaikan*, *index_b_disesuaikan, index_c_disesuaikan*, dan *index_d_disesuaikan* supaya tersisa data nilai lembahnya.

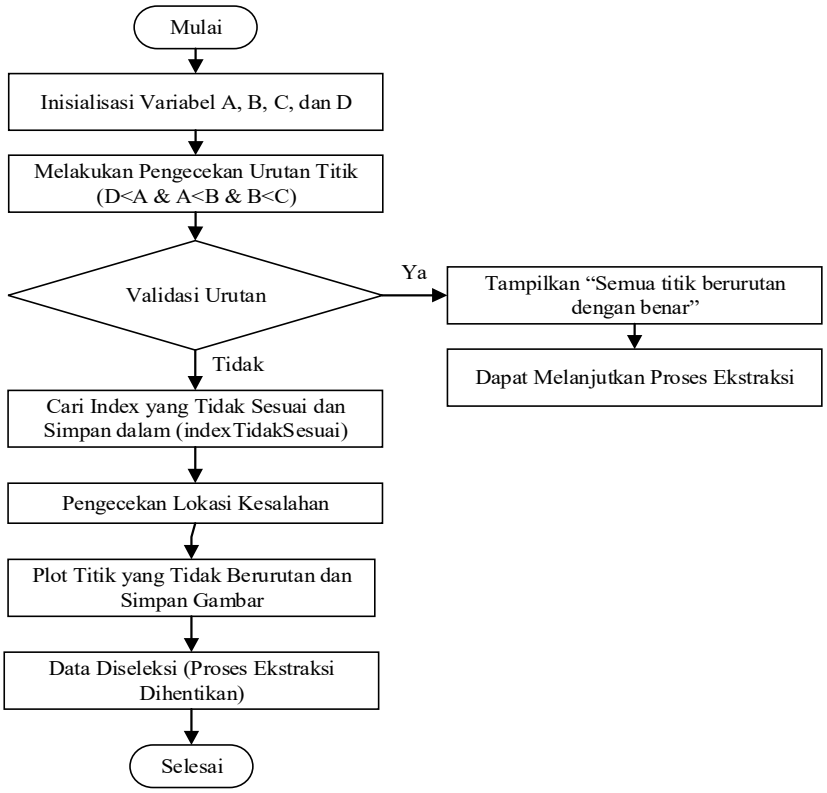
13. Menyesuaikan Ukuran *Array*

    Menyesuaikan ukuran *array index_a_disesuaikan, index_b_disesuaikan, index_c_disesuaikan, dan index_d_disesuaikan* ke Ukuran Terkecil *minSize*.

14. Mendapatkan Nilai *a_disesuaikan, b_disesuaikan, c_disesuaikan,* dan *d_disesuaikan*

    Mendapatkan nilai *a_disesuaikan, b_disesuaikan, c_disesuaikan, dan d_disesuaikan* untuk mencari nilai *a_disesuaikan, b_disesuaikan, c_disesuaikan, dan d_disesuaikan*.

3.7.5 Melakukan Pengecekan Urutan Titik Penting



Gambar 3.30 *Flowchart* Melakukan Pengecekan Urutan Titik Penting

Pengecekkan urutan titik pentig melibatkan beberapa langkah Gambar 3.30, yang dijelaskan sebagai berikut:

1. Inisialisasi Variabel

   Menginisialisasi variabel *A, B, C,* dan *D* dengan *index_a_disesuaikan, index_b_disesuaikan, index_c_disesuaikan,* dan *index_d_disesuaikan* sebagai *array* kosong.

2. Melakukan Pengecekan Urutan Titik

   Periksa apakah kondisi ($D<A$ & $A<B$ & $B<C$) terpenuhi. Jika ya, tampilkan "Semua titik berurutan dengan benar" dan lanjutkan proses ekstraksi. Jika tidak, cari index yang tidak sesuai dan simpan dalam (*indexTidakSesuai*).

3. Pengecekkan Lokasi Kesalahan

   Pengecekkan dilakukan untuk mengetahui apakah kelasahan berada di gelombang awal atau tidak. Jika kesalahan berada ditengah-tengah sinyal maka sinyal tersebut dikatakan benar-benar gagal dan tidak bisa digunakan kembali.

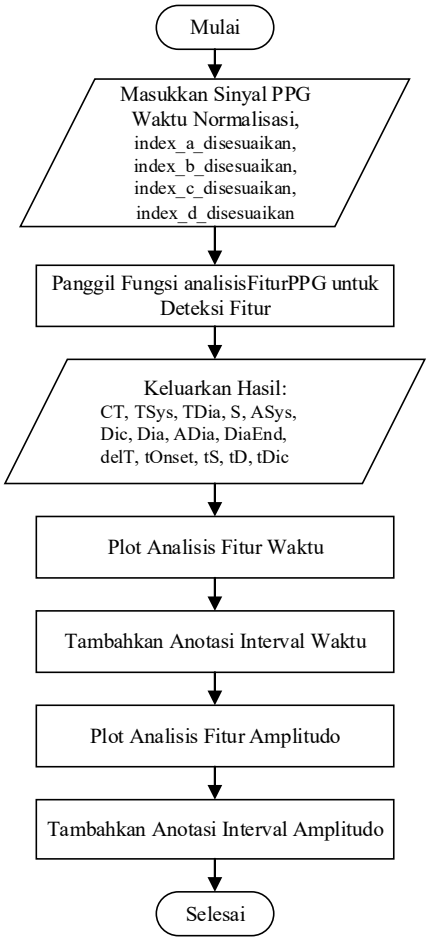
4. Plot Titik yang Tidak Berurutan dan Simpan Gambar

   Visualisasikan sinyal PPG dan waktu normalisasi dengan menandai posisi variabel *a, b, c*, dan *d*. Kemudian lihat bagian sinyal yang memiliki titik tidak seusa, kemudian atur supaya sumbu x dengan *xlim* supaya titik yang tidak sesuai terlihat dengan jelas.

5. Data Diseleksi (Proses Ekstraksi Dihentikan)

   Hasil visualisasi data diseleksi kemudian akan disimpan dalam format png, dan data tersebut tidak bisa digunakan untuk proses selanjutnnya.

3.8 Ekstraksi dan Perhitungan Amplitudo serta Waktu



Gambar 3.31 *Flowchart* Ekstraksi dan Perhitungan Amplitudo serta Waktu

Sebagaimana dijelaskan pada Subbab 2.9, sinyal PPG memiliki komponen amplitudo dan interval waktu yang berkaitan erat dengan tekanan darah. Proses ekstraksi serta perhitungan fitur amplitudo dan waktu dalam penelitian ini dilakukan melalui beberapa langkah terstruktur, sebagaimana ditunjukkan pada Gambar 3.31, dan dijelaskan sebagai berikut:

1. Masukkan Sinyal PPG dan Waktu Normalisasi serta Indeks yang Telah Disesuaikan

   Masukkan data *sinyalPPGNormalisasi*, *waktuNormalisasi, index_a_disesuaikan, index_b_disesuaikan, index_c_disesuaikan*, dan

index_d_disesuaikan, yang akan digunakan dalam proses ekstraksi dan perhitungan amplitudo serta waktu.

2. Panggil Fungsi *analisisFiturPPG*

   Gunakan fungsi analisisFiturPPG untuk menghitung fitur amplitudo dan waktu dari sinyal PPG.

3. Keluarkan Hasil

   Hasil yang diperoleh berupa variabel seperti *CT, TSys, TDia, S, ASys, Dic, Dia, ADia, DiaEnd, delT, tOnset, tS, tD, tDic.*

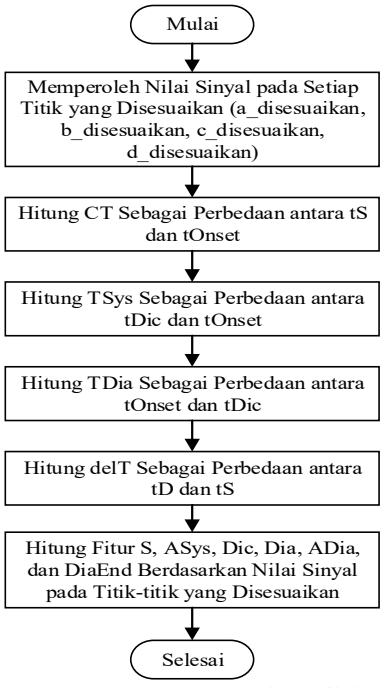
4. Plot Fitur Waktu

   Visualisasikan hasil fitur waktu pada sinyal PPG Normalisai. Terakhir, tambahkan anotasi interval waktu.

5. Plot Fitur Amplitudo

   Visualisasikan hasil fitur amplitudo pada sinyal PPG Normalisai. Terakhir, tambahkan anotasi interval amplitudo.

3.8.1 Fungsi *analisisFiturPPG*



Gambar 3.32 *Flowchart* Fungsi analisisFiturPPG

Fungsi *analisisFiturPPG* yang digunakan dalam proses ekstraksi dan perhitungan amplitudo serta waktu melibatkan langkah-langkah Gambar 3.32, yang dijelaskan sebagai berikut:

1. Memperoleh Nilai Titik yang Disesuaikan

   Nilai titik yang disesuaikan (*a_disesuaikan, b_disesuaikan, c_disesuaikan, dan d_disesuaikan*) diperoleh dengan mencari nilai berdasarkan indeks yang telah disesuaikan (*index_a_disesuaikan, index_b_disesuaikan, index_c_disesuaikan, dan index_d_disesuaikan*).

2. Hitung *CT*

   Hitung *CT* dengan melakukan pengurangan antara *tS* dan *tOnset*. Dimana *tS* adalah waktu titik puncak sistolik, dan *tOnset* adalah waktu titik *onset*.

3. Hitung *TSys*

   Hitung *TSys* dengan melakukan pengurangan antara *tDic* dan *tOnset*. Dimana *tDic* adalah waktu titik *dicrotic*, dan *tOnset* adalah waktu titik *onset*.

4. Hitung *TDia*

   Hitung *TDia* dengan melakukan pengurangan antara *tOnset* dan *tDic*. Dimana *tOnset* adalah waktu titik *onset*, dan *tDic* adalah waktu titik *dicrotic*.

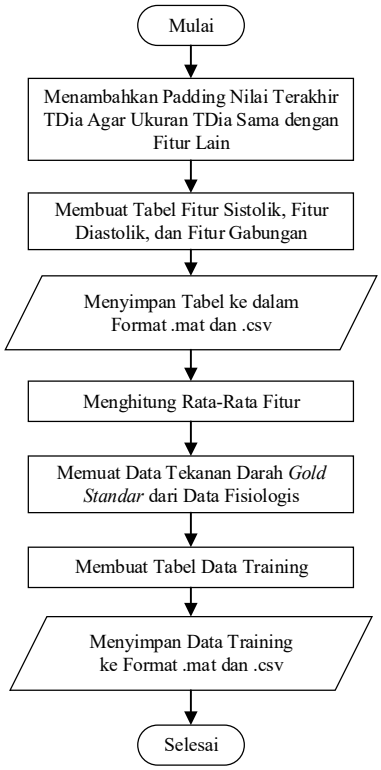
5. Hitung *delT*

   Hitung *delT* dengan melakukan pengurangan antara *tD* dan *tS*. Dimana *tD* adalah waktu titik puncak diastolik, dan *tS* adalah waktu titik puncak sistolik.

6. Hitung Fitur Amplitudo

   Hitung fitur *S, ASys, Dic, Dia, ADia*, dan *DiaEnd* berdasarkan nilai sinyal pada titik-titik yang disesuaikan (*a_disesuaikan, b_disesuaikan, c_disesuaikan, dan d_disesuaikan*).

3.9 Menyimpan Data Fitur Hasil Ekstraksi



Gambar 3.33 *Flowchart* Menyimpan Data Fitur Hasil Ekstraksi

Proses penyimpanan data fitur hasil ekstraksi disesuaikan dengan kebutuhan format dataset untuk keperluan ML, sebagaimana dijelaskan pada Subbab 2.5. Tahapan penyimpanan data fitur dilakukan secara sistematis sebagaimana ditunjukkan pada Gambar 3.33, dan dijelaskan sebagai berikut:

1. Menambahkan *Padding* pada Fitur *TDia*

   Menambahkan *padding* nilai terakhir *TDia* agar ukuran *TDia* sama dengan fitur lain. Hal ini dilakukan karena *TDia* dihitung dari pengurangan antara *tDic* dan *tOnset* gelombang selanjutnya. Sehingga ukuran *TDia* pasti akan berselisih *1* dengan fitur lainnya.

2. Membuat Tabel Fitur Sistolik, Fitur Diastolik, dan Fitur Gabungan

   Simpan hasil fitur sistolik sebagai tabel *fiturSistolik*, yang mencakup kolom *S, ASys, tOnset, CT, TSys*, dan *delT*. Simpan hasil fitur diastolik sebagai tabel

fiturDiastolik, yang mencakup kolom *Dia, ADia, Dic, CT, TDia*, dan *delT*. Simpan hasil fitur gabungan sebagai tabel *fiturSinyal*, yang mencakup kolom *S, ASys, tOnset, CT, TSys, Dia, ADia, Dic, CT, TDia,* dan *delT*.

3. Menyimpan Tabel ke dalam Format.mat dan .xlsx

   Simpan setiap tabel fitur tersebut ke dalam format file mat dan xlsx.

4. Menghitung Rata-Rata Fitur

   Setiap fitur yang diperoleh akan dihitung rata-ratanya. Hal ini dilakukan karena dalam satu sinyal PPG menghasilkan kumpulan nilai fitur, dan perhitungan rata-rata ini digunakan untuk mempersiapkan data *training*.

5. Memuat Data Tekanan Darah Gold Standar dari Data Fisiologis

   Setelah nilai rata-rata fitur diperoleh maka selanjutnya mengambil data tekanan darah *gold standard* (*BPsys* dan *Bpdias*) dari data fisiologis masing-masing partisipan. Nilai tekanan darah tersebut kemudian akan digabungkan dengan data fitur rata-rata.

6. Membuat Tabel Data *Training*

   Simpan hasil data *training* sistolik sebagai tabel *dataTrainFS*, yang mencakup kolom *S_mean, ASys_mean, tOnset_mean, CT_mean, TSys_mean, delT_mean,* dan *BPsys*. Simpan hasil data *training* sistolik sebagai tabel *dataTrainFD*, yang mencakup kolom *Dia_mean, ADia_mean, Dic, CT_mean, TDia_mean, delT_mean*, dan *BPdias*. Simpan hasil data training gabungan sebagai tabel *dataTrain*, yang mencakup kolom *S_mean, ASys_mean, tOnset_mean, CT_mean, TSys_mean, ADia_mean, Dic, CT_mean, TDia_mean, delT_mean, BPsys*, dan *BPdias*.

7. Menyimpan Data *Training* ke Format.mat dan .xlsx

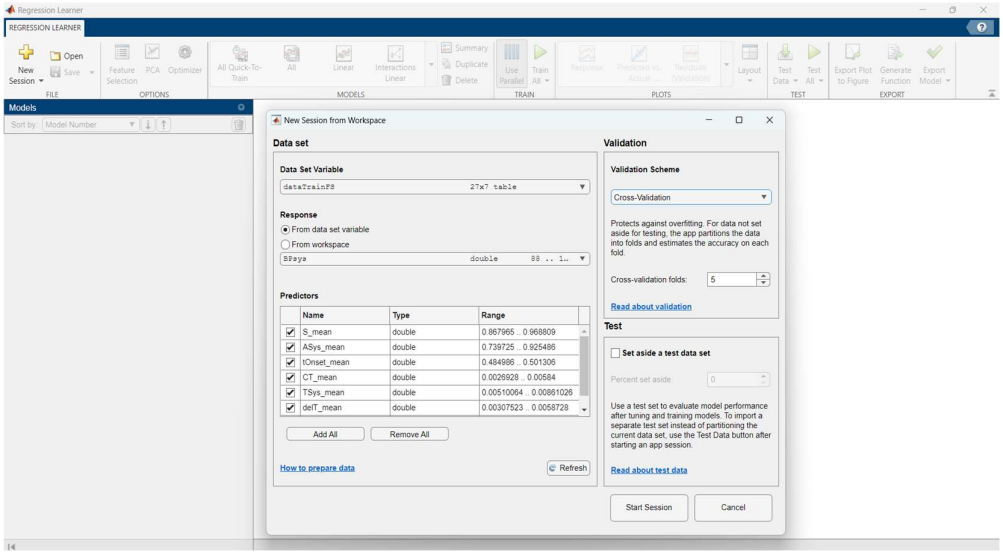
   Simpan setiap tabel fitur tersebut ke dalam format file mat dan xlsx.

3.10 Melatih Model Prediksi

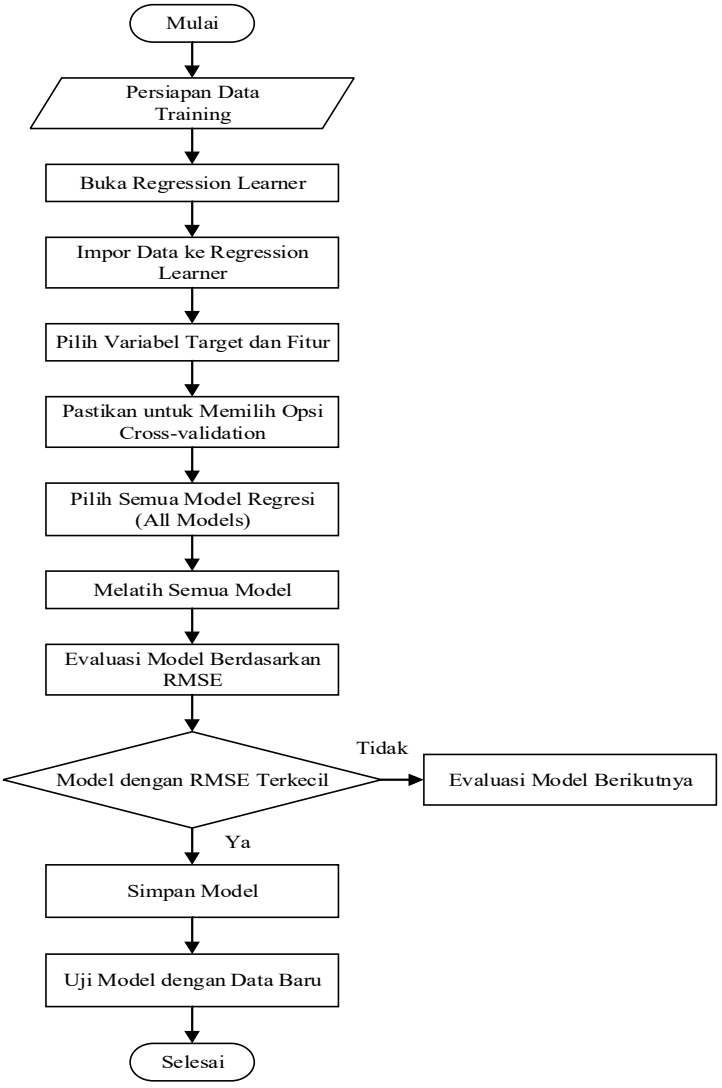
Dalam penelitian ini, proses pembuatan model regresi untuk analisis data dilakukan menggunakan *toolbox Regression Learner* di MATLAB Gambar 3.34. *Toolbox* ini menyediakan berbagai algoritma pembelajaran mesin, seperti regresi linier, pohon regresi, dan metode *ensemble*, yang memungkinkan pengguna untuk melatih, memvalidasi, dan membandingkan model secara efisien.

Regression Learner mempermudah eksplorasi performa model dengan fitur visualisasi interaktif dan metrik evaluasi seperti RMSE dan R^2. *Toolbox* ini digunakan untuk mengembangkan model prediksi berdasarkan fitur-fitur yang diekstraksi dari sinyal PPG, sehingga mendukung analisis hubungan antara fitur dengan parameter fisiologis tertentu.

Tahapan penggunaan *toolbox* ini dijelaskan secara rinci melalui diagram alir Gambar 3.34, yang menggambarkan langkah-langkah mulai dari pemilihan dataset, pemrosesan data, pelatihan model, hingga evaluasi performa model prediksi.



Gambar 3.34 Penggunaan *Toolbox Regression Learner* di MATLAB untuk Melatih Model Prediksi



Gambar 3.35 *Flowchart* Pelatihan Model Prediksi

Proses Pelatihan model prediksi tekanan darah Gambar 3.35, di *regression learner* melibatkan beberapa langkah berikut:

1. Persiapan Data

   Memuat data tekanan darah dan fitur-fitur terkait dari *workspace* MATLAB atau dari file eksternal (seperti .mat atau .csv). Memastikan data mencakup variabel target (*BPsys* untuk tekanan darah sistolik atau *BPdia* untuk tekanan

darah diastolik) serta variabel fitur yang relevan (seperti *CT_mean, TSys_mean, delT_mean,* dll).

2. Buka *Regression Learner*

   Membuka aplikasi *Regression Learner Toolbox* di MATLAB. Ini bisa dilakukan melalui menu *Apps* atau dengan mengetik *regressionLearner* di *Command Window* MATLAB.

3. Impor Data ke *Regression Learner*

   Memilih data yang sudah disiapkan dan memasukkannya sebagai set data di *Regression Learner* untuk diproses lebih lanjut.

4. Pilih Variabel Target dan Fitur

   Menentukan variabel target sebagai tekanan darah (*BPsys* atau *BPdia*). Memilih variabel-variabel fitur yang akan digunakan untuk memprediksi variabel target.

5. Pastikan untuk memilih opsi *cross-validation*

   Memastikan opsi *cross-validation* aktif agar model terlatih lebih baik dan mengurangi risiko *overfitting*.

6. Pilih Semua Model Regresi (*All Models*)

   Menggunakan opsi *All Models* di *Regression Learner* untuk mencoba berbagai model regresi yang tersedia, seperti *linear regression, support vector machines* (SVM), *decision trees, ensemble*, dan *Gaussian process regression* (GPR).

7. Melatih Semua Model

   Mengklik *Train All* untuk melatih semua model regresi yang tersedia dengan set data yang telah dipilih. Kemudian tunggu hingga pelatihan selesai dan melihat nilai *Root Mean Squared Error* (RMSE) yang dihasilkan setiap model.

8. Evaluasi Model Berdasarkan RMSE

   Mengevaluasi hasil pelatihan dengan nilai RMSE dari setiap model yang dilatih. Model dengan nilai RMSE terkecil dipilih sebagai model terbaik.

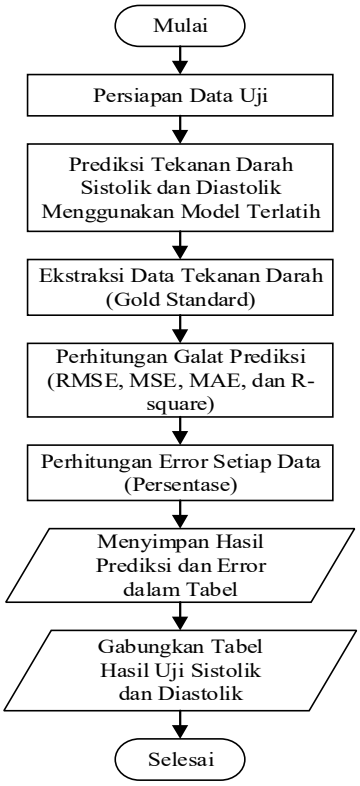
9. Simpan Model Terbaik

   Menyimpan model terbaik ke dalam *workspace* MATLAB sebagai fungsi prediktor atau mengekspornya untuk digunakan pada langkah-langkah selanjutnya.

10. Uji Model dengan Data Baru (Opsional)

    Menguji model yang telah disimpan menggunakan data uji atau data baru untuk memastikan model dapat melakukan prediksi tekanan darah dengan baik di luar data pelatihan.

3.11 Pengujian Model dan Perhitungan Galat Prediksi Tekanan Darah



Gambar 3.36 *Flowchart* Pengujian Model dan Perhitungan Galat Prediksi Tekanan Darah

Proses pengujian model dan perhitungan galat melibatkan beberapa langkah Gambar 3.36, yang dijelaskan sebagai berikut:

1. Persiapan Data Uji

   Ambil data uji untuk tekanan darah sistolik (*dataUjiSis*) dan diastolik (*dataUjiDia*) dari set data untuk digunakan dalam proses prediksi.

2. Prediksi Tekanan Darah Menggunakan Model Terlatih

   Gunakan model yang telah dilatih untuk memprediksi tekanan darah:

   a. Prediksi tekanan darah sistolik (*ySfit*) menggunakan model sistolik (*trainedModelFS.predictFcn(dataUjiSis)*).

b. Prediksi tekanan darah diastolik (*yDfit*) menggunakan model diastolik (*trainedModelFD.predictFcn(dataUjiDia)*).

3. Ekstraksi Data Tekanan Darah Nyata (*Gold Standard*)

   Ekstrak nilai tekanan darah sebenarnya sebagai data pembanding:

   a. Tekanan darah sistolik (*yS*) diambil dari kolom yang sesuai dalam *dataUjiSis*.

   b. Tekanan darah diastolik (*yD*) diambil dari kolom yang sesuai dalam *dataUjiDia*.

4. Perhitungan Galat Prediksi

   Hitung *error* atau galat prediksi untuk setiap model menggunakan metrik berikut:

   a. *Root Mean Squared Error* (RMSE) dihitung menggunakan persamaan (2.21):

      Sistolik: *RMSESis = sqrt(mean((yS - yS_pred).^2))*

      Diastolik: *RMSEDia = sqrt(mean((yD - yD_pred).^2))*

   b. *Mean Squared Error* (MSE) dihitung menggunakan persamaan (2.22):

      Sistolik: *MSESis = mean((yS - yS_pred).^2)*

      Diastolik: *MSEDia = mean((yD - yD_pred).^2)*

   c. *Mean Absolute Error* (MAE) dihitung menggunakan persamaan (2.23):

      Sistolik: *MAESis = mean(abs(yS - yS_pred))*

      Diastolik: *MAEDia = mean(abs(yD - yD_pred))*

   d. *R-square* (R^2) dihitung menggunakan persamaan (2.24):

      Sistolik: *SS_resSis = sum((yS - yS_pred).^2)*

SS_totSis = sum((yS - mean(yS)).^2)

R2Sis = 1 - (SS_resSis / SS_totSis)

Diastolik: *SS_resDia = sum((yD - yD_pred).^2)*

SS_totDia = sum((yD - mean(yD)).^2)

R2Dia = 1 - (SS_resDia / SS_totDia)

5. Perhitungan *Error* Setiap Data (Persentase)

   Perhitungan *error* dilakukan secara individual untuk masing-masing data, menggunakan persamaan (2.25) sebagai acuan:

   a. *Error* Sistolik:

      Hitung selisih antara prediksi dan nilai asli:

      selisihSis = abs(yS_pred - yS)

      Hitung persentase *error*: *errorSis = (selisihSis * 100) ./ yS*

   b. *Error* Diastolik:

      Hitung selisih antara prediksi dan nilai asli:

      selisihDia = abs(yD_pred - yD)

      Hitung persentase *error*: errorDia = *(selisihDia * 100) ./ yD*

6. Menyimpan Hasil Prediksi dan *Error* dalam Tabel

   Buat tabel untuk menyimpan data hasil prediksi dan *error*:

   a. *hasilUjiSis* menyimpan nilai asli, prediksi, dan *error* persentase untuk tekanan darah sistolik (*yS, yS_pred, errorSis*).

   b. *hasilUjiDia* menyimpan nilai asli, prediksi, dan *error* persentase untuk tekanan darah diastolik (*yD, yD_pred, errorDia*).

7. Gabungkan Tabel Hasil Uji Sistolik dan Diastolik

   Gabungkan tabel hasil uji sistolik dan diastolik (*hasilUjiSis* dan *hasilUjiDia*) menjadi satu tabel, *hasilUji*, untuk kemudahan analisis.