KLASIFIKASI SINYAL JANTUNG *PHONOCARDIOGRAM* MENGGUNAKAN *ENSEMBLE LEARNING* DENGAN PENDEKATAN *SOFT VOTING* BERBASIS *WEB*

TUGAS AKHIR

Program Studi
S1 TEKNIK KOMPUTER

Oleh:
Adisaputra Zidha Noorizki
20410200018

FAKULTAS TEKNOLOGI DAN INFORMATIKA
UNIVERSITAS DINAMIKA
2024

KLASIFIKASI SINYAL JANTUNG *PHONOCARDIOGRAM* MENGGUNAKAN *ENSEMBLE LEARNING* DENGAN PENDEKATAN *SOFT VOTING* BERBASIS *WEB*

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk menyelesaikan
Program Sarjana

Oleh:

Nama	**:**	**Adisaputra Zidha Noorizki**
NIM	**:**	**20410200018**
Program Studi	**:**	**S1 Teknik Komputer**

FAKULTAS TEKNOLOGI DAN INFORMATIKA
UNIVERSITAS DINAMIKA
2024

TUGAS AKHIR

KLASIFIKASI SINYAL JANTUNG *PHONOCARDIOGRAM* MENGGUNAKAN *ENSEMBLE LEARNING* DENGAN PENDEKATAN *SOFT VOTING* BERBASIS *WEB*

Dipersiapkan dan disusun oleh:

Adisaputra Zidha Noorizki

NIM : 20410200018

Telah diperiksa, dibahas, dan disetujui oleh Dewan Pembahas

Pada: Senin, 22 Juli 2024

Susunan Dewan Pembahas

Pembimbing:

I. Heri Pratikno, M.T., MTCNA., MTCRE.

NIDN: 0716117302

II. Weny Indah Kusumawati, S.Kom., M.MT.

NIDN: 0721047201

Digitally signed by Heri Pratikno, M.T.
DN: cn=Heri Pratikno, M.T., o=Universitas Dinamika, ou=S1 Teknik Komputer, email=heri@dinamika.ac.id, c=ID
Date: 2024.07.22 13:04:52 +07'00'
Adobe Acrobat version: 11.0.23

cn=Weny Indah Kusumawati, o=Undika, ou=Prodi S1 TK - FTI, email=weny@dinamika.ac.id, c=ID
2024.07.22 12:03:49 +07'00'

Pembahas:

Pauladie Susanto, S.Kom., M.T.

NIDN: 0729047501

cn=Pauladie Susanto, o=Universitas Dinamika, ou=PS S1 Teknik Komputer, email=pauladie@dinamika.ac.id, c=ID
2024.07.22 13:11:58 +07'00'

Tugas Akhir ini telah diterima sebagai salah satu persyaratan

untuk memperoleh gelar Sarjana

Digitally signed by Anjik Sukmaaji
Date: 2024.07.29 11:42:47 +07'00'

Dr. Anjik Sukmaaji, S.Kom., M.Eng.

NIDN: 0731057301

Dekan Fakultas Teknologi dan Informatika

UNIVERSITAS DINAMIKA

"*Hate has four letters, but so does love. Enemies has seven letters, but so does friends. Lying has five letters, but so does truth.* ***Failure has seven letters, but so does success****. Cry has three letters, but has does joy. And negativity has ten letters, but so does positivity.* ***You always have a choice, so choose the better side of it***."

- Zaky Tyree

"*Dipersembahkan kepada Diri Sendiri yang telah membuktikan keberanian, ketekunan, dan kerja keras dalam menghadapi setiap tantangan hingga titik ini. Kepada Ibu dan Ayah, yang senantiasa memberikan doa, dukungan, dan motivasi tanpa henti. Juga kepada seluruh keluarga, yang selalu memberikan semangat dan dukungan yang tulus. Tidak lupa juga kepada semua pihak yang telah memberikan bantuan, bimbingan, dan kontribusi dalam setiap langkah penyusunan laporan ini.*"

PERNYATAAN

PERSETUJUAN PUBLIKASI DAN KEASLIAN KARYA ILMIAH

Sebagai mahasiswa **Universitas Dinamika**, Saya :

Nama	:	**Adisaputra Zidha Noorizki**
NIM	:	**20410200018**
Program Studi	:	**S1 Teknik Komputer**
Fakultas	:	**Teknologi dan Informatika**
Jenis Karya	:	**Laporan Tugas Akhir**
Judul Karya	:	**KLASIFIKASI SINYAL JANTUNG *PHONOCARDIOGRAM* MENGGUNAKAN *ENSEMBLE LEARNING* DENGAN PENDEKATAN *SOFT VOTING* BERBASIS *WEB***

Menyatakan dengan sesungguhnya bahwa :

1. Demi pengembangan Ilmu Pengetahuan, Teknologi dan Seni, Saya menyetujui memberikan kepada **Universitas Dinamika** Hak Bebas Royalti Non-Eksklusif (***Non-Exclusive Royalty Free Right***) atas seluruh isi/sebagian karya ilmiah Saya tersebut diatas untuk disimpan, dialihmediakan, dan dikelola dalam bentuk pangkalan data (***database***) untuk selanjutnya didistribusikan atau dipublikasikan demi kepentingan akademis dengan tetap mencantumkan nama Saya sebagai penulis atau pencipta dan sebagai pemilik Hak Cipta.
2. Karya tersebut diatas adalah hasil karya asli Saya, bukan plagiat baik sebagian maupun keseluruhan. Kutipan, karya, atau pendapat orang lain yang ada dalam karya ilmiah ini semata-mata hanya sebagai rujukan yang dicantumkan dalam Daftar Pustaka Saya.
3. Apabila dikemudian hari ditemukan dan terbukti terdapat tindakan plagiasi pada karya ilmiah ini, maka Saya bersedia untuk menerima pencabutan terhadap gelar kesarjanaan yang telah diberikan kepada Saya.

Demikan surat pernyataan ini Saya buat dengan sebenar-benarnya.

Surabaya, 26 Juni 2024

Adisaputra Zidha Noorizki

NIM: 20410200018

ABSTRAK

Penelitian ini bertujuan untuk mengklasifikasikan sinyal jantung *phonocardiogram* ke dalam kelas normal dan abnormal menggunakan *ensemble learning* dengan pendekatan *soft voting*, serta membangun antarmuka pengguna berbasis *web* untuk memantau kinerja model. Model dasar yang diuji meliputi *Long Short-Term Memory* (LSTM), *Gated Recurrent Unit* (GRU), dan *Temporal Convolutional Network* (TCN), yang dilatih dengan variasi jumlah *epoch* dan nilai *learning rate*. Model GRU menunjukkan performa terbaik dengan *epoch* 20 serta nilai akurasi, presisi, *recall*, dan *f1-score* masing-masing sebesar 94%. Disisi lain, penerapan variansi bobot pada metode *ensemble learning* tidak menghasilkan peningkatan yang signifikan dalam kinerja model, dengan akurasi dan *f1-score* terbaik masing-masing hanya sebesar 60% dan 40%. Implementasi antarmuka berbasis *web* memungkinkan pengguna untuk melihat prediksi model dan memantau metrik performa seperti akurasi, presisi, *recall*, dan *f1-score* secara *real-time*. Kesimpulan dari penelitian ini menunjukkan bahwa model dasar GRU lebih unggul dibandingkan dengan model *ensemble* terbaik yang menggunakan pendekatan *soft voting*, menunjukkan perlunya eksplorasi lebih lanjut terhadap metode *ensemble* alternatif dan model dasar yang lebih bervariasi dalam klasifikasi sinyal jantung *phonocardiogram*.

Kata kunci : *Ensemble learning*, *soft voting*, sinyal jantung *phonocardiogram*.

KATA PENGANTAR

Puji syukur peneliti panjatkan kepada Allah Ta'ala yang senantiasa memberikan berkat, rahmat, serta hidayah-Nya, sehingga peneliti dapat menyelesaikan Laporan Tugas Akhir yang berjudul **KLASIFIKASI SINYAL JANTUNG *PHONOCARDIOGRAM* MENGGUNAKAN *ENSEMBLE LEARNING* DENGAN PENDEKATAN *SOFT VOTING* BERBASIS *WEB***. Laporan ini disusun sebagai salah satu bentuk tanggung jawab yang diberikan kepada peneliti untuk memenuhi persyaratan dalam memperoleh gelar Sarjana di Universitas Dinamika. Peneliti sangat menyadari bahwa penyelesaian laporan ini tidak akan mungkin tercapai tanpa bantuan, bimbingan, serta arahan dari berbagai pihak yang telah mendukung dan memfasilitasi proses penelitian ini. Oleh karena itu, dengan segala kerendahan hati, peneliti ingin mengucapkan rasa terima kasih yang sebesar-besarnya kepada :

1. Bapak Prof. Dr. Budi Jatmiko, M.Pd., selaku Rektor Universitas Dinamika, yang telah memberikan arahan strategis dan kebijakan yang mendukung terciptanya lingkungan akademik yang kondusif untuk penelitian ini.
2. Bapak Dr. Anjik Sukmaaji, S.Kom., M.Eng., selaku Dekan Fakultas Teknologi dan Informatika Universitas Dinamika, yang telah menyediakan fasilitas dan dukungan akademik yang diperlukan untuk kelancaran proses penelitian ini.
3. Bapak Pauladie Susanto, S.Kom., M.T., selaku Ketua Program Studi Strata Satu Teknik Komputer sekaligus Dewan Pembahas pada penelitian ini, yang telah memberikan masukan konstruktif dan evaluasi yang mendalam, sehingga peneliti dapat meningkatkan kualitas dari hasil penelitian ini.
4. Bapak Heri Pratikno, M.T., MTCNA., MTCRE., selaku Dosen Pembimbing I yang telah memberikan bimbingan kepada penulis, baik selama proses penelitian hingga proses penyusunan laporan ini.
5. Ibu Weny Indah Kusumawati, S.Kom., M.MT., selaku Dosen Pembimbing II yang telah memberikan bimbingan kepada penulis, baik selama proses penelitian hingga proses penyusunan laporan ini.

6. Kedua orang tua, Ibu dan Ayah, yang selalu memberikan dukungan moral, doa, dan motivasi tanpa henti, sehingga peneliti dapat menyelesaikan Laporan Tugas Akhir ini dengan lancar dan penuh keyakinan.
7. Rekan-rekan yang telah memberikan saran, dan diskusi yang konstruktif, serta kebersamaan yang tak ternilai selama menempuh pendidikan hingga menyelesaikan penelitian ini.
8. Serta semua pihak yang tidak bisa peneliti sebutkan satu persatu.

Dalam penulisan laporan ini, peneliti sangat menyadari bahwa masih terdapat kekurangan dan ruang untuk perbaikan. Oleh karena itu, peneliti sangat mengharapkan adanya saran dan kritik yang membangun, sehingga laporan ini dapat diperbaiki dan dikembangkan lebih lanjut di masa mendatang. Akhir kata, peneliti berharap laporan ini dapat memberikan pengetahuan yang bermanfaat dan berguna bagi para pembaca serta menjadi referensi yang berarti dalam bidang terkait. Terima kasih atas segala perhatian dan dukungannya.

Surabaya, 22 Juli 2024

Peneliti

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR LAMPIRAN

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Jantung adalah organ vital manusia yang berperan krusial dalam memompa darah ke seluruh tubuh dan menerima kembali darah setelah melewati paru-paru (Handayani, 2021). Seiring bertambahnya usia, kinerja jantung cenderung menurun (Pujiastuti, et al., 2023). Oleh karena itu, menjaga kesehatan jantung menjadi penting untuk meminimalkan risiko gangguan kesehatan dan menjalani hidup yang sehat. Namun, data dari *World Health Organization* (WHO) pada tahun 2021 menunjukkan bahwa penyakit jantung menyebabkan 17,8 juta kematian setiap tahun, atau satu dari tiga kematian di seluruh dunia. *American Heart Association* juga melaporkan bahwa penyakit jantung kardiovaskular adalah penyebab kematian global tertinggi (Amal, M. A., et al., 2023). Di Indonesia, menurut *Institute for Health Metrics and Evaluation* (IHME) dari 2014 hingga 2019, penyakit jantung menduduki peringkat tertinggi sebagai penyebab kematian. Data Riset Kesehatan Dasar (Riskesdas) menunjukkan peningkatan insiden penyakit jantung menjadi 1,5% pada tahun 2018, dengan pembiayaan kesehatan terbesar oleh BPJS Kesehatan mencapai Rp 7,7 triliun untuk penyakit jantung.

Dalam bidang medis, teknik seperti *electrocardiogram* (ECG) dan *phonocardiogram* (PCG) digunakan untuk memantau aktivitas jantung dan mendiagnosis penyakit kardiovaskular. ECG menilai kondisi jantung secara langsung, tetapi tidak selalu mencakup semua kelainan, seperti murmur jantung (Wang, M., et al., 2022). Oleh karena itu, penelitian ini menggunakan data sinyal *phonocardiogram* (PCG). Secara tradisional, praktisi medis mendengarkan suara jantung dengan stetoskop, namun PCG mampu mencerminkan kondisi sistem kardiovaskular secara menyeluruh, sehingga signifikan dalam mendukung diagnosis penyakit jantung (Triyani, Y., et al., 2021). PCG adalah metode untuk memonitoring aktivitas katup jantung dengan menghitung arus peredaran darah menggunakan sensor (Miskiyanto, 2021). PCG merekam suara jantung dan murmur yang terjadi selama siklus kerja jantung. *Murmur* merupakan suara tambahan yang menunjukkan adanya gangguan jantung. Normalnya, PCG merekam suara jantung

utama, S1 dan S2, tetapi dalam kondisi abnormal, *murmur* yang muncul sebagai suara S3 dan S4 juga direkam (Lee, S. Y., et al., 2019).

Penggunaan stetoskop untuk diagnosis tradisional memiliki kendala karena bergantung pada keterampilan dan pengalaman praktisi medis dalam menginterpretasi suara jantung. Oleh karena itu, teknologi menjadi penting untuk mempermudah dan mempercepat diagnosis. Penelitian ini fokus pada implementasi diagnosis sinyal PCG dengan klasifikasi menggunakan teknologi *machine learning*. Dalam *machine learn*ing, *ensemble learning* merupakan teknik yang menggabungkan beberapa model untuk meningkatkan kinerja prediksi (Ganaie, M. A., et al., 2022). Salah satu pendekatan umumnya adalah *soft voting*, yang membandingkan hasil prediksi setiap model dan memberi bobot untuk menentukan kelas prediksi yang paling sering muncul. Pendekatan ini meningkatkan akurasi dan keandalan prediksi model *machine learning*, sehingga populer dalam mengklasifikasikan sebuah objek.

Dalam beberapa penelitian sebelumnya telah menunjukkan potensi besar dalam klasifikasi sinyal PCG. Penelitian yang dilakukan oleh Miskiyanto (2021) yaitu mengklasifikasikan sinyal jantung PCG menggunakan metode *Long Short-Term Memory* (LSTM) dengan akurasi 91% untuk data latih dan 81.7% untuk data validasi. Penelitian yang dilakukan oleh Rizky, M. G. (2021) membandingkan metode LSTM dan BiLSTM untuk klasifikasi sinyal PCG dan menemukan bahwa BiLSTM mencapai akurasi 89%, lebih tinggi dibandingkan LSTM dengan 81%. Sedangkan penelitian yang dilakukan oleh Amal, M. A., et al. (2023) menggunakan *Short Time Fourier Transform* dan *Convolutional Neural Network* untuk klasifikasi sinyal PCG, mencapai akurasi 88,11% dengan penggunaan *hamming window*.

Dari beberapa penelitian sebelumnya, penelitian ini bertujuan untuk meningkatkan akurasi sistem diagnosis kesehatan (normal) dan kelainan jantung (abnormal) melalui klasifikasi sinyal PCG menggunakan *ensemble learning* dengan pendekatan *soft voting* berbasis *website*. Diharapkan penelitian ini dapat berkontribusi pada perkembangan sistem diagnosis yang lebih efisien dan akurat di dunia medis, mempermudah deteksi dini kelainan jantung, dan meningkatkan kualitas hidup pasien.

1.2 Rumusan Masalah

Berdasarkan semua latar belakang dan uraian terkait penelitian terdahulu sebagaimana tersebut di atas, maka rumusan masalah utama dalam penelitian ini adalah bagaimana kinerja model pada metode *ensemble learning* dengan pendekatan *soft voting* secara signifikan dapat meningkatkan nilai persentase akurasi dari proses klasifikasi sinyal jantung *phococardiogram* melalui antarmuka aplikasi berbasis *website*.

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini telah ditentukan dengan mempertimbangkan latar belakang, penelitian terdahulu, dan rumusan masalah sebelumnya. Berikut adalah batasan-batasan yang diterapkan:

1) *Dataset* sinyal jantung *phonocardiogram* (PCG) terdiri atas 3.240 sampel data mentah yang berasal dari laman *open-source Physionet*. Di mana, *dataset* telah dikelompokkan menjadi dua (2) kelas, yaitu normal dan abnormal, dengan format file '.wav'.
2) *Dataset* akan dilatih hanya menggunakan tiga (3) jenis model *machine learning* yang berbeda sebelum menerapkan metode *ensemble learning* dengan pendekatan *soft voting*.
3) Antarmuka berbasis *website* akan digunakan sebagai *platform* untuk menampilkan kinerja model yang telah dilatih, serta hanya menerima *input* berupa file berekstensi '.wav'.

1.4 Tujuan

Berdasarkan latar belakang, rumusan masalah, dan batasan masalah yang sebelumnya telah dipaparkan, berikut tujuan dari penelitian ini:

1) Dapat mengklasifikasikan sinyal jantung *phonocardiogram* (PCG) ke dalam dua (2) kelas yaitu normal dan abnormal dengan tiga (3) model *machine learning* yang berbeda yaitu LSTM, GRU, dan TCN.
2) Dapat meningkatkan kinerja model *machine learning* dalam mengklasifikasi sinyal jantung PCG dengan menerapkan metode *ensemble learning* dengan pendekatan *soft voting.*

3) Dapat mengetahui hasil perbandingan antara model secara individu dan hasil model yang telah menerapkan metode *ensemble learning* dengan pendekatan *soft voting*.

1.5 Manfaat

Dengan beberapa hal yang telah dipaparkan di atas, maka terkait manfaat dari terlaksananya penelitian ini antara lain:

1) Memahami apakah penerapan metode *ensemble learning* dalam mengklasifikasikan sinyal jantung PCG dapat meningkatkan keakuratan model yang dilatih.
2) Mengoptimalkan sistem pendeteksian penyakit kasdiovaskular dengan lebih efektif, sehingga dapat mempermudah para praktisi medis dan para pasien untuk mendeteksi dini kelainan pada jantung.

BAB II
LANDASAN TEORI

2.1 Jantung

Memiliki peran utama untuk memompa darah ke seluruh tubuh, jantung menjadi salah satu organ vital pada tubuh manusia yang penting untuk diperhatikan. Jantung pada manusia terbagi menjadi empat (4) ruang utama yaitu dua (2) atrium/serambi di bagian atas dan dua (2) bilik/*ventrikel* pada bagian bawah (Whiteman, S., et al., 2021). Ruang-ruang tersebut pun memiliki fungsi dan tugas yang berbeda-beda. Seperti atrium kanan memiliki peran untuk menerima darah kotor yang berasal dari seluruh tubuh, sementara atrium kiri menerima darah bersih yang berasal dari paru-paru.

Berbeda dengan peran atrium/serambi pada jantung yang menerima darah bersih maupun kotor, peran bilik/*ventrikel* pada jantung adalah memompa darah. Ventrikel kanan memompa darah ke paru-paru untuk dilakukan proses pertukaran gas oksigen (O2) dan karbon dioksida (CO2), sementara bilik/*vetrikel* kiri memompa darah bersih ke seluruh tubuh. Di mana untuk setiap proses yang sebelumnya telah dipaparkan, darah dipindahkan dari ruang ke ruang yang lain menggunakan otot jantung, serta adanya peran katup jantung yang secara otomatis membuka dan menutup saat proses pemompaan darah terjadi.

2.1.1 Sinyal Jantung Jantung

Sinyal suara jantung atau yang biasa dikenal sebagai bunyi jantung, merupakan salah satu fenomena yang terjadi dan ada akibat aktifitas kontraksi dan relaksasi dari organ jantung. Aktifitas ini akan menghasilkan dua suara berbeda yang dinamakan sebagai '*lub*' dan '*dub*'. Para praktisi medis umumnya menggunakan bantuan alat bernama stetoskop untuk mendengarkan dua (2) bunyi yang dihasilkan jantung ini. Stetoskop memiliki dua bagian (*earpiece* dan *chestpiece*) yang memungkinkan mereka mendengarkan dengan lebih jelas dan fokus pada suara internal tubuh (Vincent, R., 2022).

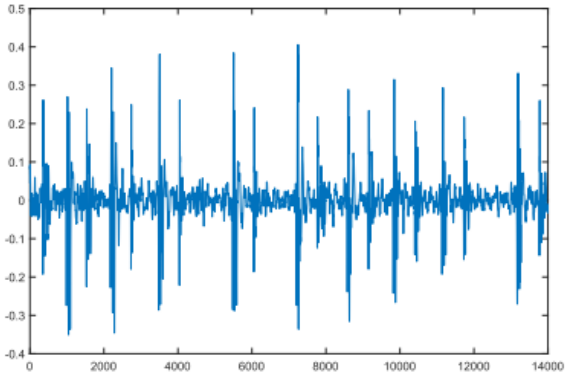
Bunyi pertama (S1) atau '*lub*' terjadi selama kontraksi ventrikel/bilik jantung. Bunyi ini disebabkan oleh penutupan katup mitral dan katup *trikuspid*

(katup *atrioventrikuler*) saat darah dipompa keluar dari atrium (ruang atas jantung) ke *ventrikel*. Sementara '*dub*' atau bunyi kedua (S2) terjadi selama relaksasi *ventrikel*. Bunyi ini disebabkan oleh penutupan katup aorta dan katup pulmonal saat *ventrikel* berkontraksi untuk memompa darah keluar dari jantung ke dalam sirkulasi tubuh. Selain S1 dan S2, jantung juga memiliki bunyi ketiga (S3) dan bunyi keempat (S4), di mana kedua sinyal ini menandakan adanya sinyal aktivitas tambahan pada jantung.

Bunyi jantung S3 umumnya akan terdengar pada anak-anak dan pasien dewasa yang sedang menyidap penyakit jantung. Bunyi S3 atau *ventricular gallop* memiliki nada rendah yang terjadi saat *ventrikel* terisi dengan cepat pada periode *diastol*, sehingga menghasilkan sebuah getaran. Sementara bunyi jantung S4 atau derap atrium umumnya terdengar pada area *trikuspid* atau *mitral* yang dapat terjadi pada para pasien di usia lanjut, pasien hipertensi, dan pasien yang memiliki riwayat *miokard infark*. Bunyi S4 terjadi sebelum suara jantung S1 akibat adanya kontraksi atrium pada akhir periode *diastol*, sehingga menghasilkan getaran yang disebabkan oleh ejeksi atrium yang kuat dari darah (Nurhayati, 2021).

2.1.2 *Phonocardiogram*

Phonocardiogram (PCG) merupakan teknik pencatatan grafis dari suara jantung yang dihasilkan selama siklus jantung. Dalam menghasilkan *phonocardiogram*, sinyal suara jantung yang diperoleh melalui stetoskop direkam dan direpresentasikan secara visual. Teknologi ini memberikan gambaran grafis yang jelas tentang pola bunyi jantung, memungkinkan para praktisi dalam bidang medis untuk melakukan analisis lebih mendalam terhadap aktivitas jantung pasien. Dalam proses perekamannya, sensor suara berupa mikrofon akan difungsikan untuk mendeteksi dan mengkonversi getaran suara jantung menjadi sinyal listrik berbentuk analog yang kemudian diubah menjadi sinyal digital melalui proses *sampling* dan kuantisasi (Oktivasari, P., et al., 2020). Sinyal digital inilah yang direkam dan diolah menjadi grafik *phonocardiogram*, di mana merepresentasikan waktu pada sumbu X (horizontal) dan amplitudo pada sumbu Y (vertikal).



Gambar 2.1 Sinyal jantung *phonocardiogram* (PCG)
(Sumber: Yang, L., et al., 2020)

2.2 *Ensemble Learning*

Metode *ensemble learning* merupakan pendekatan *machine learning* di mana sejumlah model atau yang dikenal juga sebagai *base learner* dikombinasikan untuk menghasilkan satu model prediksi yang lebih akurat (Dachi, J. M. A. S., & Sitompul, P., 2023). *Ensemble learning* bekerja dengan cara melatih beberapa model menggunakan algoritma pembelajaran yang sama atau berbeda pada dataset yang sama, kemudian hasil prediksi dari masing-masing model digabungkan sehingga menghasilkan satu (1) keputusan untuk menentukan prediksi akhir.

Model atau *base learner* yang menjadi komponen dalam *ensemble* dapat berasal dari berbagai jenis algoritma pembelajaran mesin seperti *decision tree*, *support vector machine*, *neural network*, dan lainnya. Pemilihan dan kombinasi *base learner* yang tepat memungkinkan untuk menangkap kekuatan dari berbagai pendekatan algoritma. Masing-masing *base learner* dilatih secara independen menggunakan seluruh atau sebagian data latih. Proses ini memungkinkan setiap model menangkap pola dan informasi unik dari dataset. Setelah proses pelatihan, hasil prediksi dari setiap model kemudian dikombinasikan dengan berbagai pendekatan seperti *voting*, *weighted averaging*, *stacking*, dan lainnya untuk menghasilkan prediksi akhir. Dengan menggabungkan hasil dari beberapa model, *ensemble learning* dapat berpotensi untuk mengurangi *overfitting* pada data (Saluza, I., & Hartati, H., 2022).

2.3 *Soft Voting*

Salah satu metode yang sering digunakan dalam *ensemble learning*, khususnya pada tahap penggabungan hasil prediksi adalah *soft voting*. *Soft voting* merupakan pendekatan di mana model atau *base learner* memberikan prediksi dengan menghasilkan nilai probabilitas untuk setiap kelas pada studi kasus klasifikasi (Budiman, F., & Awaludin, Y. M., 2023). Kemudian, prediksi akhir diambil dengan menghitung rata-rata tertimbang dari probabilitas kelas yang dihasilkan oleh masing-masing model. Secara matematis, *soft voting* dapat direpresentasikan sebagai berikut:

$$P(c_i) = \frac{\sum_{j=1}^{M} w_j \,.\, P_j(c_i)}{\sum_{j=1}^{M} w_j} \quad (1)$$

Keterangan:

$P(c_i)$: Probabilitas prediksi akhir untuk suatu kelas c_i
M : Jumlah model/*base learner* dalam *ensemble*
w_j : Bobot yang diberikan pada model/*base learner* ke-j
$P_j(c_i)$: Probabilitas prediksi dari model/*base learner* ke-j untuk kelas c_i

Rumus 1 menjelaskan bahwa setiap model memberikan kontribusi pada prediksi akhir berdasarkan probabilitas kelas yang dihasilkannya. Bobot (w_j) dapat diberikan untuk mengatur kontribusi relatif dari setiap model dalam pengambilan keputusan akhir. Probabilitas prediksi akhir kemudian dihitung dengan menjumlahkan probabilitas yang telah diboboti dari semua model.

2.4 *Fast Fourier Transform* (FFT)

Algoritma yang sering juga disingkat FFT ini merupakan algoritma yang digunakan untuk menghitung transformasi *fourier* dari suatu sinyal dalam cara yang efisien secara komputasional. FFT secara signifikat mengurangi jumlah operasi yang diperlukan dibandingkan dengan metode konvensional sebelumnya yaitu *Discrete Fourier Transform* (DFT). Umumnya FFT akan digunakan untuk mentransformasikan sinyal dari domain waktu ke domain frekuensi, yang mana dapat memungkinkan analisis komponen frekuensi yang terdapat dalam suatu sinyal (Wang, Y., et al., 2020).

Proses FFT dimulai dengan menguraikan sinyal dalam domain waktu pada titik N menjadi N sinyal domain waktu yang lebih kecil. Selanjutnya, dilakukan

perhitungan untuk menentukan N frekuensi spektrum yang berkorelasi dengan N sinyal domain waktu tersebut. Proses ini membentuk spektrum frekuensi yang mencerminkan distribusi energi frekuensi dalam sinyal. Akhirnya, spektrum N ini disintesis menjadi satu spektrum frekuensi tunggal yang merepresentasikan sinyal dalam domain frekuensi. Secara matematis, FFT atau *Fast Fourier Transform* dapat direpresentasikan sebagai berikut:

$$X[k] = \sum_{n=0}^{N} x[n] \, . \, e^{-j\frac{2\pi}{N}nf} \tag{2}$$

Keterangan:

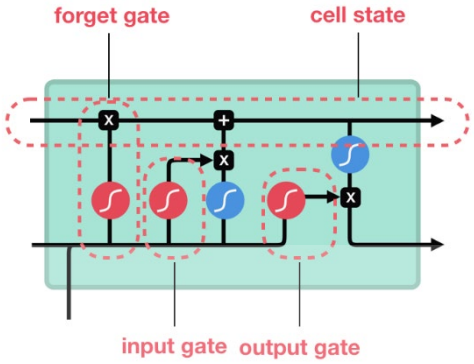
$X[k]$: Frekuensi hasil FFT pada indeks frekuensi k.
$x[n]$: Amplitudo sinyal pada waktu n.
$e^{-j\frac{2\pi}{N}nf}$: Eksponensial kompleks yang menciptakan rotasi sudut di sepanjang lingkaran kompleks.
j : Satuan kompleks $\sqrt{-1}$ (satuan imajiner).
N : Panjang sinyal/jumlah sampel dalam domain waktu.
n : Indeks waktu dari 0 hingga $N - 1$.
f : Indeks frekuensi dari 0 hingga $N - 1$.

Rumus 2 merepresentasikan bagaimana setiap komponen frekuensi pada sinyal waktu $x[n]$ dihitung dalam domain frekuensi menggunakan FFT (Wang, Y., et al., 2020). Proses ini memecahkan sinyal menjadi komponen frekuensi yang lebih sederhana, memungkinkan analisis lebih lanjut terhadap karakteristik frekuensi dari sinyal tersebut.

2.5 *Long Short-Term Memory* (LSTM)

LSTM atau *Long Short-Term Memory* merupakan jenis arsitektur jaringan saraf tiruan (JST) yang dirancang khusus untuk mengatasi tantangan memori jangka panjang dan pendek dalam analisis dan prediksi data sekuensial. Pertama kali dikembangkan oleh Hochreiter dan Schmidhuber pada tahun 1997. LSTM memiliki kemampuan untuk menangani masalah *vanishing gradient* dan *exploding gradient* yang sering ditemui dalam JST konvensional saat memproses urutan data yang cukup panjang (Rehmer, A., & Kroll, A., 2020). *Vanishing gradient problems* merupakan masalah ketika *gradient* menyusut seiring proses *back propagation* melalui waktu. Sedangkan *exploding gradient* merupakan kebalikannya, di mana *gradient* menjadi sangat besar selama proses *back propagation*.

LSTM menggunakan unit memori khusus yang disebut '*cell*', yang memiliki kemampuan untuk menyimpan, menghapus, dan mengakses informasi dalam jangka waktu yang lama. Hal ini memungkinkan LSTM untuk menangkap dependensi jangka panjang dalam urutan data, sehingga sangat efektif dalam memodelkan pola yang kompleks dan hubungan temporal (Wang, J., et al., 2023). Unit gerbang, seperti gerbang lupa (*forget gate*), gerbang *input*, dan gerbang *output*, memainkan peran penting dalam mengontrol aliran informasi dan mengatur sejauh mana informasi baru diintegrasikan ke dalam sel memori.



Gambar 2.2 Struktur jaringan LSTM
(Sumber: Lambert, 2019)

Input gate berfungsi mengatur jumlah informasi baru yang akan disertakan ke dalam sel memori. Dengan menggunakan fungsi *sigmoid*, *input gate* menghasilkan nilai antara 0 dan 1, yang menentukan sejauh mana informasi baru akan diintegrasikan. Nilai ini kemudian dikalikan dengan nilai yang dihasilkan oleh fungsi *hyperbolic tangent* (tanh) dari *input*, menciptakan vektor informasi baru yang ditambahkan ke dalam sel memori. Sementara, *forget gate* berguna mengatur seberapa banyak informasi dari sel memori sebelumnya harus dihapus atau dilupakan. Dengan memanfaatkan fungsi *sigmoid*, *forget gate* menghasilkan nilai antara 0 dan 1, menentukan sejauh mana informasi sebelumnya harus dipertahankan atau dibuang. Nilai ini kemudian digunakan untuk mengalikan sel memori sebelumnya.

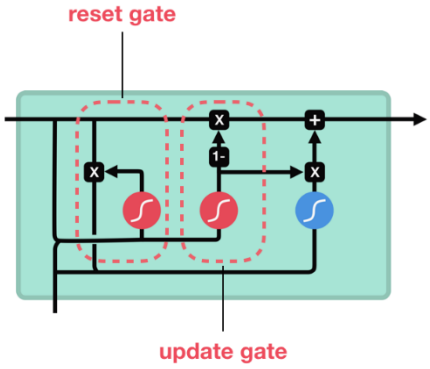
Sel memori menyimpan informasi jangka panjang, yang diperlukan oleh model untuk memahami konteks dalam data berurutan. Informasi baru dari *input* gate ditambahkan ke dalam sel memori, dan informasi lama yang akan dilupakan

dihitung melalui *forget gate*. Dengan demikian, sel memori memperbarui isinya pada setiap langkah waktu. Sementara, *output gate* berguna untuk mengontrol informasi yang akan dikirim ke lapisan berikutnya atau berfungsi sebagai *output* model. Dengan menggunakan fungsi *sigmoid*, *output gate* menghasilkan nilai antara 0 dan 1, menentukan seberapa besar sel memori akan memengaruhi *output*. Nilai ini kemudian dikalikan dengan nilai *hyperbolic tangent* dari sel memori untuk menghasilkan *output* akhir pada langkah waktu tertentu.

Secara keseluruhan, kinerja LSTM terletak pada kemampuannya untuk menyimpan informasi jangka panjang dan mendeteksi pola temporal dalam urutan data. Fungsi setiap *gate* memastikan LSTM dapat secara efektif mengelola dan memproses informasi dalam konteks sekuensial yang panjang, membuatnya cocok untuk tugas-tugas seperti prediksi deret waktu ataupun pemrosesan bahasa alami.

2.6 *Gated Recurrent Unit* (GRU)

GRU atau *Gated Recurrent Unit* merupakan salah satu jenis arsitektur jaringan pada *Recurrent Neural Network* yang dirancang untuk mengatasi beberapa kendala dalam model LSTM sekaligus mempertahankan kemampuan untuk menangkap dependensi temporal pada data berurutan. Apabila dibandingkan dengan LSTM, GRU memiliki struktur yang lebih sederhana dengan hanya dua gerbang utama, yaitu gerbang *reset* (*reset gate*) dan gerbang pembaruan (*update gate*). Gerbang *reset* (rt) dan gerbang *update* (zt) pada GRU dihitung menggunakan fungsi sigmoid dan memungkinkan model untuk memutus atau mempertahankan informasi yang disimpan pada langkah waktu sebelumnya (Wu, Z., et al., 2023).



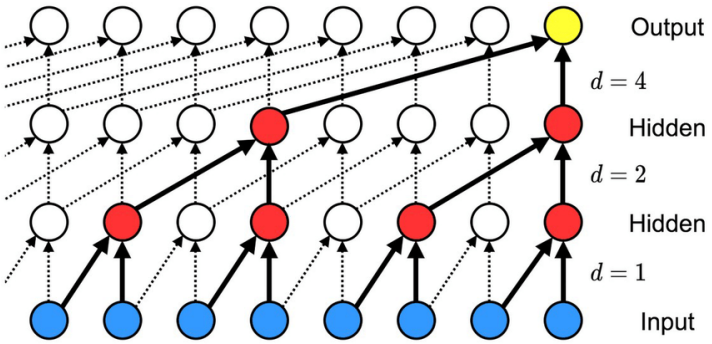
Gambar 2.3 Struktur jaringan GRU
(Sumber: Lambert, 2019)

Proses pada *Gated Recurrent Unit* (GRU) dimulai dengan inisialisasi parameter, termasuk matriks bobot W dan vektor bias b. Selanjutnya, input data berurutan x_t dan *hidden state* sebelumnya $h_{t\text{-}1}$ digunakan untuk menghitung nilai dari dua gerbang utama, yaitu *reset gate* (r_t) dan *update gate* (z_t), menggunakan fungsi aktivasi sigmoid. *Reset gate* (r_t) memungkinkan model untuk memilih apakah akan memutus atau mempertahankan informasi dari langkah waktu sebelumnya. Di sisi lain, *update gate* (z_t) menentukan seberapa banyak informasi baru yang akan diintegrasikan ke dalam *hidden state*.

Setelah itu, kandidat aktivasi ($\tilde{h}_t$) dihitung melalui fungsi aktivasi *hyperbolic tangent* (tanh), yang mempertimbangkan *reset gate* untuk mengatur pengaruh informasi dari *hidden state* sebelumnya. *Update gate* kemudian digunakan untuk menggabungkan kandidat aktivasi ($\tilde{h}_t$) dengan *hidden state* sebelumnya ($h_{t\text{-}1}$), sehingga menghasilkan *hidden state* yang diperbarui ($\tilde{h}_t$) untuk langkah waktu saat ini. Proses ini memungkinkan GRU untuk secara adaptif mempelajari dan menyimpan informasi relevan dari masa lalu, sehingga dapat mengatasi masalah *vanishing* atau *exploding gradient*. Selain itu, GRU secara efektif mampu menangkap dependensi temporal dalam data berurutan. Selama pelatihan, parameter-parameter GRU diperbarui melalui proses pembelajaran berbasis gradien untuk meningkatkan kemampuan model dalam memahami pola dan hubungan dalam data temporal.

2.7 *Temporal Convolutional Network* (TCN)

TCN atau singkatan dari *Temporal Convolutional Network* merupakan suatu arsitektur *neural network* yang dirancang khusus untuk menangani data berurutan dengan efisien. Berbeda dengan *Convolutional Neural Network* (CNN), yang umumnya diterapkan pada data gambar, TCN dirancang untuk menangkap pola dan dependensi temporal dalam data berurutan. Salah satu komponen utama dalam TCN adalah lapisan *dilated convolution*, yang memungkinkan model untuk memperluas jangkauan temporal tanpa mengorbankan efisiensi komputasional (Shaikh., A. K., et al., 2023).



Gambar 2.4 Struktur jaringan TCN
(Sumber: Lee, K., 2021)

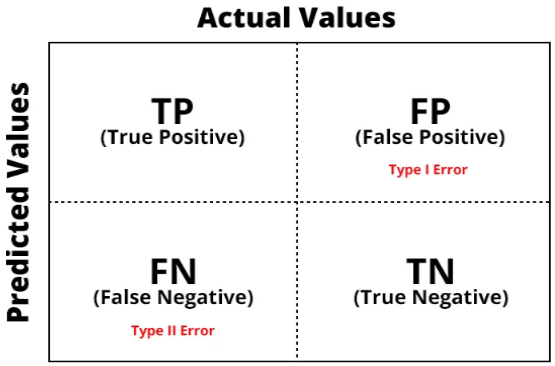
Lapisan *dilated convolution* pada TCN memungkinkan model melibatkan informasi dari jarak yang lebih jauh dalam sejarah data temporal. Hal ini memberikan kemampuan adaptif kepada TCN untuk menyesuaikan diri dengan konteks temporal yang lebih luas, sebuah aspek yang sangat berharga dalam analisis data deret waktu. Sebagai contoh, ketika menganalisis rangkaian waktu panjang, TCN dapat secara efektif menangkap pola jangka panjang tanpa menimbulkan masalah komputasi berlebihan.

Keunggulan TCN membuatnya menjadi pilihan yang sangat efisien dan efektif dalam berbagai tugas, termasuk prediksi deret waktu, analisis data deret waktu, dan tugas-tugas lain yang melibatkan data berurutan (Li, S., et al., 2023). Dengan kemampuannya dalam menangkap informasi temporal yang kompleks, TCN memberikan kontribusi penting dalam pengembangan model yang dapat memahami dan memproses data deret waktu dengan akurasi tinggi.

2.8 *Confusion Matrix*

Confusion matrix yang juga dikenal sebagai matriks kebingungan merupakan alat evaluasi penting dalam analisis kinerja model klasifikasi (Valero-Carreras, D., et al., 2023). Matriks ini mengorganisir hasil prediksi model ke dalam empat kategori utama yaitu *true positive* (TP), *true negative* (TN), *false positive* (FP), dan *false negative* (FN). *True positive* mewakili data positif yang benar-benar diklasifikasikan dengan benar oleh model, sedangkan *true negative* mencerminkan data negatif yang diklasifikasikan dengan benar. Di sisi lain, *false positive* adalah

jumlah data negatif yang salah diklasifikasikan sebagai data positif, sementara *false negative* adalah jumlah data positif yang salah diklasifikasikan sebagai data negatif.



Gambar 2.5 *Confusion matrix*

2.9 Python

Python dikenal sebagai salah satu bahasa pemrograman tingkat tinggi, sebab kefleksibilitasannya terhadap berbagai bidang pengembangan teknologi. Python telah menjadi pilihan utama para pengembang di berbagai sektor berkat keunggulan yang dimilikinya. Penulisan sintaksis yang bersahabat, kemudahan dalam pembacaan kode, dan dukungan komprehensif terhadap berbagai paradigma pemrograman membuat Python sangat populer dewasa ini. Keluasan perpustakaan Python dan komunitas yang besar turut memperkaya ekosistem pengembangan, memudahkan pengguna dalam mengakses dan mengimplementasikan fungsionalitas yang kompleks. Dikenalkan oleh penciptanya Guido van Rossum pada akhir 1980-an, Python sukses menciptakan ekosistem yang ramah pengembang, menjadikannya pilihan utama bagi *programmer* yang mengutamakan efisiensi, kejelasan, dan keberlanjutan proyek (Kaswan, K. S., et al., 2023).

Keras merupakan salah satu *library* yang sangat populer dalam pengembangan *machine learning* dan *artificial intelligence*. Hal ini disebabkan oleh antarmuka yang mudah digunakan dan kemampuannya untuk berjalan di atas berbagai *backend* seperti TensorFlow, Theano, dan Microsoft Cognitive Toolkit (CNTK). Keras menyediakan berbagai alat dan fungsi yang memungkinkan pengembang untuk membangun, melatih, dan menyebarkan model *machine learning* dengan lebih cepat dan efisien.

2.10 *Website Development*

Salah satu percabangan pada pengembangan teknologi yang termasuk salah satu cabang yang memiliki minat paling banyak di dunia adalah pengembangan *website*. *Website development*, atau pengembangan *website*, mencakup proses perancangan, pembangunan, dan pemeliharaan halaman *website*. Fenomena ini muncul seiring dengan perkembangan internet, di mana situs *website* menjadi sarana utama bagi individu, bisnis, dan organisasi untuk berkomunikasi, berbagi informasi, dan menyediakan layanan secara daring.

Pengembangan situs *website* melibatkan berbagai aspek, termasuk desain antarmuka pengguna, pemrograman, manajemen basis data, dan optimisasi kinerja. Desain antarmuka pengguna bertujuan menciptakan tata letak yang menarik dan mudah digunakan, sementara pemrograman memastikan fungsi-fungsi situs berjalan dengan lancar. Manajemen basis data diperlukan untuk menyimpan dan mengelola informasi yang akan ditampilkan di situs *website*, sementara optimisasi kinerja fokus pada meningkatkan kecepatan dan efisiensi situs.

2.9.1 *Wireframe* Antarmuka

Wireframe antarmuka merupakan tahap awal dalam pengembangan situs web. Ini adalah representasi sketsa atau kerangka kerja visual dari tata letak halaman web tanpa desain atau elemen grafis terperinci. *Wireframe* membantu desainer dan pengembang dalam merencanakan struktur dan navigasi situs sebelum masuk ke tahap pengembangan yang lebih mendalam, sehingga menghindari *miss*-komunikasi yang dapat saja terjadi disaat proses pengembangan suatu sistem.

Figma merupakan alat desain antarmuka yang populer digunakan untuk membuat *wireframe*. Figma memungkinkan kolaborasi waktu nyata antara desainer dan pengembang, sehingga memudahkan proses perencanaan dan iterasi desain. Dengan fitur-fitur seperti *prototyping*, *version control*, dan integrasi dengan berbagai alat desain lainnya, Figma menjadi pilihan utama dalam pengembangan *wireframe* yang efektif dan efisien.

2.9.2 Pengembangan *Backend*

Pengembangan *backend* melibatkan pembangunan *endpoint Application Programming Interface* (API) yang mendukung fungsi-fungsi *website*. Pengembang *backend* bertanggung jawab untuk memastikan bahwa data disimpan dan diakses dengan aman dan fungsi-fungsi server berjalan secara efisien. Bahasa pemrograman seperti Python, Ruby, dan Node.js sering digunakan dalam pengembangan *backend*.

1) Flask

   Flask merupakan *framework* web berbasis Python yang ringan dan fleksibel. Flask digunakan untuk membuat API dan layanan *backend* yang mudah diatur dan dapat diskalakan sesuai kebutuhan. Dengan arsitektur mikro yang memungkinkan penambahan komponen tambahan hanya jika diperlukan, Flask memberikan kebebasan dan fleksibilitas yang besar dalam pengembangan aplikasi *web*.

2) PhpMyAdmin

   PhpMyAdmin merupakan alat berbasis *web* yang ditulis dalam PHP dan digunakan untuk menangani administrasi MySQL melalui antarmuka *web*. PhpMyAdmin mendukung berbagai operasi MySQL, seperti mengelola basis data, tabel, kolom, relasi, indeks, pengguna, izin, dan banyak lagi. Dengan PhpMyAdmin, pengembang dapat menjalankan kueri SQL, mengelola basis data, dan melakukan tugas administrasi lainnya dengan lebih mudah melalui antarmuka grafis yang *user-friendly*.

2.9.3 Pengembangan *Frontend*

Pengembangan *frontend* fokus pada implementasi desain antarmuka pengguna ke dalam kode-kode yang dapat dijalankan oleh browser. Pengembang *frontend* bekerja dengan HTML, CSS, dan JavaScript untuk menciptakan pengalaman pengguna yang interaktif dan menarik. Pengembangan ini akan memastikan tampilan laman *website* dapat responsif dan dapat diakses di berbagai perangkat.

1) *HyperText Markup Language* (HTML)

HTML digunakan untuk membuat struktur dasar halaman *web*. HTML memungkinkan penyusunan elemen-elemen konten seperti teks, gambar, dan tautan. Ini merupakan fondasi dari semua halaman *web* dan digunakan untuk memberikan kerangka dasar yang akan diisi dengan elemen-elemen visual dan interaktif lainnya.

2) *Cascading Style Sheets* (CSS)

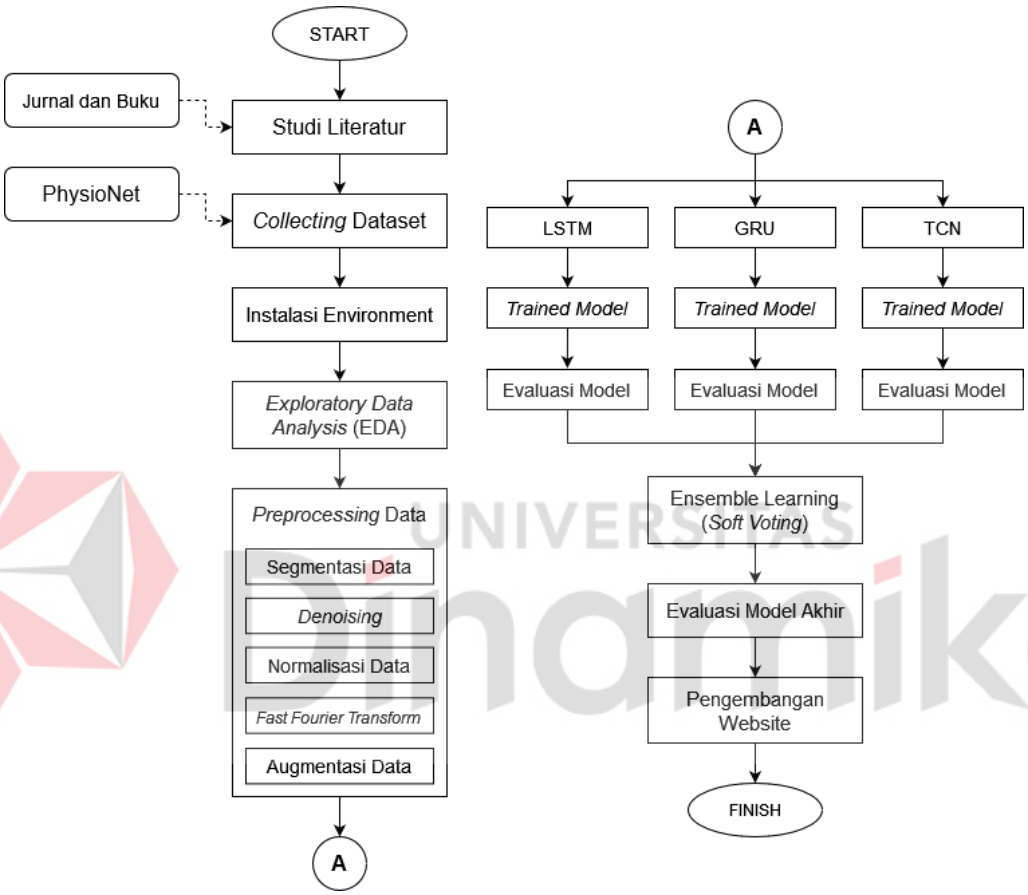
CSS digunakan untuk mengatur tata letak dan tampilan halaman *web*. CSS memungkinkan pengembang untuk menyesuaikan desain agar responsif dan menarik di berbagai perangkat. Dengan CSS, elemen HTML dapat diatur secara visual untuk menciptakan tampilan yang konsisten dan menarik di seluruh situs.

3) JavaScript

JavaScript digunakan untuk menambah interaktivitas pada halaman *web*. JavaScript memungkinkan pengembang untuk membuat fitur dinamis seperti formulir yang dapat divalidasi secara langsung, animasi, dan manipulasi DOM (*Document Object Model*). Dengan JavaScript, halaman *web* menjadi lebih hidup dan responsif terhadap interaksi pengguna.

BAB III
METODOLOGI PENELITIAN

Melalui diagram alir berikut, peneliti bermaksud menggambarkan alur dari penelitian ini. Diagram tersebut mencakup langkah-langkah dan proses yang akan dilakukan selama penelitian.



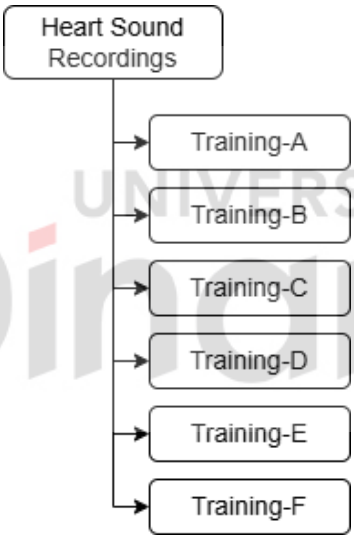
Gambar 3.1 Diagram alir penelitian

3.1 Studi Literatur

Tahap ini bertujuan untuk mengetahui dan memahami lebih lanjut terkait topik dan judul penelitian. Peneliti akan mengumpulkan dan menganalisis literatur-literatur terkini serta sumber informasi relevan selama lima (5) tahun terakhir yang dapat memberikan dasar pemahaman yang kokoh terhadap topik penelitian. Mulai dari dasar teori variabel yang akan dijadikan sebagai objek penelitian, penerapan dan penggunaan metode, hingga beberapa algoritma yang akan digunakan pada penelitian ini.

3.2 *Collecting* Dataset

Penelitian ini melibatkan pemanfaatan sumber data *open-source* dari PhysioNet, terkhusus dataset berjudul '*Normal/Abnormal Heart Sound Recordings: the PhysioNet/Computing in Cardiology Challenge 2016*'. Dataset ini merupakan kumpulan rekaman suara jantung yang mencakup suara jantung normal dan abnormal, dan diambil sebagai bagian dari tantangan *Computing in Cardiology* pada tahun 2016. Dataset ini terdiri dari sekitar 3240 file audio dengan ekstensi '.wav'. Rekaman suara jantung ini telah diberikan label dan telah dibagi menjadi beberapa bagian untuk keperluan analisis. Informasi label dan metadata lainnya terkait dataset ini disajikan dalam file excel terpisah, sehingga dapat memberikan pemahaman yang lebih rinci tentang karakteristik setiap rekaman suara.



Gambar 3.2 Struktur folder dataset

3.3 Instalasi *Environment*

Penelitian ini memanfaatkan salah satu bahasa pemrograman yang cukup populer, yaitu Python. Pada tahap ini, peneliti mempersiapkan lingkungan kerja atau *environment* yang diperlukan untuk menjalankan analisis dan pengolahan data menggunakan Python. Mulai dari pemilihan versi bahasa pemrograman hingga instalasi pustaka dan dependensi pendukung lainnya dengan menggunakan pengelola paket yang biasa dikenal dengan nama 'pip'. Setelah itu juga perlunya pembuatan file dengan nama 'requirements.txt' yang berguna untuk menyimpan

semua pustaka atau *library* yang diperlukan beserta versi yang digunakan selama penelitian ini. Langkah ini bertujuan untuk mempermudah replikasi lingkungan kerja oleh penelitian di masa yang akan datang.

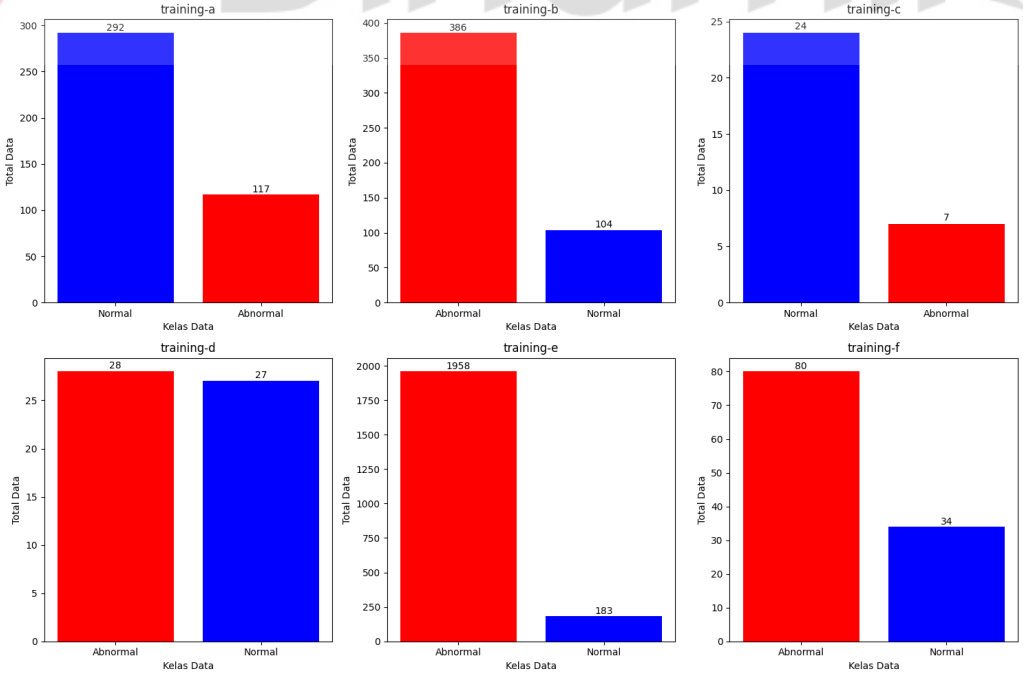
```
python --version
pip install --upgrade pip
pip install virtualenv

python -m venv myenvironment
myenvironment\Scripts\activate
pip install -r requirements.txt
```

Gambar 3.3 *Scripts* instalasi *environment*

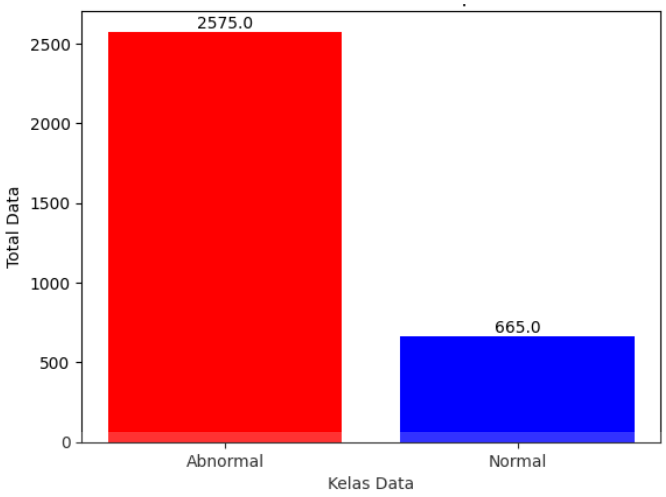
3.4 *Exploratory Data Analysis* (EDA)

Tahapan ini berfokus pada analisis dataset dengan bantuan Python untuk mendapatkan pemahaman yang mendalam mengenai data yang akan digunakan. Dengan memahami dataset secara menyeluruh, diharapkan dapat menentukan serta melakukan tahapan selanjutnya dengan tepat sesuai dengan kondisi dataset yang digunakan.



Gambar 3.4 Distribusi data kelas tiap folder

Gambar 3.4 menunjukkan bahwa jumlah label normal dan abnormal pada setiap subfolder dalam dataset memiliki variasi yang signifikan. Selain itu, terdapat perbedaan yang cukup besar dalam jumlah masing-masing label, sehingga diperlukan proses tambahan pada tahap selanjutnya untuk menyeimbangkan jumlah antar label agar tidak terlalu jauh berbeda.



Gambar 3.5 Distribusi total data setiap kelas

Gambar 3.5 mengonfirmasi bahwa jumlah data pada label abnormal lebih banyak, dengan 2575 rekaman suara, sedangkan label normal hanya memiliki 665 rekaman suara. Perbedaan jumlah yang cukup signifikan ini akan menjadi salah satu fokus utama pada tahap *preprocessing* data, untuk memastikan data latih/*training* optimal dan mencapai performa yang baik pula.

Tabel 3.1 Durasi data berdasarkan folder dan kelas

Nama folder	Kelas	Min durasi	Max durasi	Rata-rata durasi
training-a	abnormal	9.265	36.293	32.53
	normal	12.795	36.502	32.61
training-b	abnormal	6.986	8.00	7.992
	normal	5.306	8.00	7.935
training-c	abnormal	16.966	71.812	39.442
	normal	9.65	121.99	52.349
training-d	abnormal	6.61	29.077	11.75
	normal	8.043	48.54	18.426
training-e	abnormal	8.06	101.673	23.30
	normal	8.127	91.701	20.624
training-f	abnormal	29.376	59.616	33.322
	normal	30.00	42.336	32.637

Data pada Tabel 3.1 menunjukkan sebaran durasi untuk setiap kelas (abnormal dan normal) dalam tiap folder pada dataset. Durasi direpresentasikan dalam tiga metrik dengan menggunakan satuan detik, antara lain minimum (Min durasi), maksimum (Max durasi), dan rata-rata (Rata-rata durasi). Perbedaan dalam durasi minimum, maksimum, dan rata-rata antar kelas dan folder ini menunjukkan perlunya langkah khusus dalam tahap *preprocessing* data. Ketidakseimbangan dalam jumlah data antar kelas, seperti yang sebelumnya disebutkan, juga perlu ditangani untuk menghindari bias dalam model. Teknik seperti menggunakan metode *ensemble* yang dapat menangani ketidakseimbangan data dapat menjadi sebuah solusi.

Tabel 3.2 Durasi dataset secara menyeluruh

Kelas	Min durasi	Max durasi	Rata-rata durasi
abnormal	5.306	121.00	27.43
normal	6.608	101.67	24.72

Tabel 3.2 menunjukkan durasi dataset secara menyeluruh untuk masing-masing kelas (abnormal dan normal). Dari tabel tersebut, terlihat bahwa durasi rata-rata rekaman suara untuk kelas abnormal adalah 27.43 detik dan untuk kelas normal adalah 24.72 detik. Meskipun terdapat rekaman dengan durasi hingga 121 detik untuk kelas abnormal dan 101.67 detik untuk kelas normal, banyak rekaman juga yang memiliki durasi jauh lebih pendek.

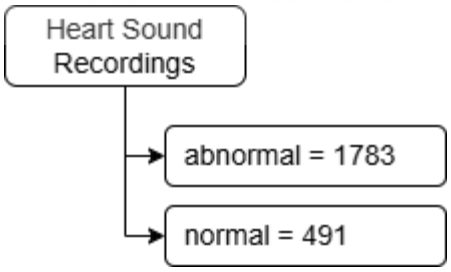
Dengan mempertimbangkan informasi ini, pengambilan sampel dengan durasi 15 detik dapat dipertimbangkan karena beberapa alasan. Pertama, sampel masih berada dalam rentang durasi yang lebih panjang dari durasi minimum namun lebih pendek dari durasi rata-rata, sehingga tetap mewakili karakteristik penting dari data. Kedua, sampel dapat secara signifikan mengurangi jumlah data yang perlu diproses, yang akan mempercepat waktu pelatihan model. Ketiga, dengan durasi 15 detik, sampel tersebut masih mencakup sebagian besar informasi penting dalam rekaman sehingga tidak menghilangkan informasi kritis. Sehingga, penggunaan sampel 15 detik dapat dipertimbangkan agar tetap menghasilkan model yang berkualitas dan representatif.

3.5 *Preprocessing* Data

Tahapan ini memiliki tujuan untuk memastikan bahwa data yang digunakan untuk pelatihan model memiliki kualitas yang baik dan representatif. Pada penelitian ini, *preprocessing* data dilakukan dalam lima tahapan utama antara lain segmentasi data, *denoising*, normalisasi, *fast fourier transform* (FFT), dan augmentasi data.

3.5.1 Segmentasi Data

Segmentasi data merupakan teknik yang melibatkan pemisahan dataset menjadi segmen-segmen yang lebih kecil atau kelompok-kelompok berdasarkan kriteria tertentu. Pada penelitian ini, proses segmentasi pada sinyal PCG melibatkan pengenalan fase-fase spesifik dalam siklus jantung, sehingga memungkinkan pemisahan sinyal PCG menjadi bagian-bagian yang dapat mewakili setiap kelas yang telah ditentukan. Pada tahap ini, data akan disaring berdasarkan durasi dengan mengambil data yang memiliki durasi minimal 15 detik. Jika durasi melebihi 15 detik, maka hanya 15 detik pertama yang akan diambil. Sehingga apabila diperhatikan berdasarkan tahapan sebelumnya, dataset akan mengalami perubahan pada jumlah data disetiap kelasnya.

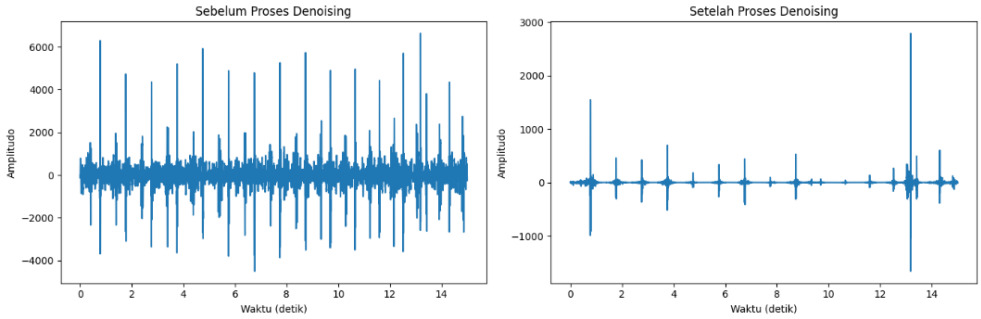


Gambar 3.6 Struktur folder dan jumlah data setelah proses segmentasi

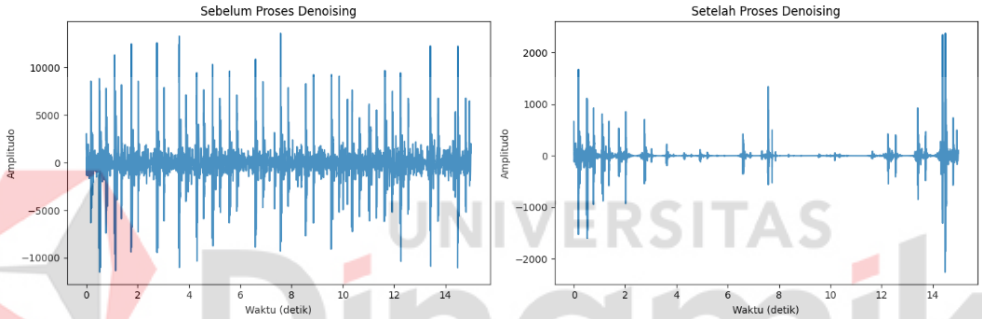
3.5.2 Denoising

Denoising merupakan proses penghilangan *noise* atau gangguan yang tidak diinginkan dari suatu sinyal. Tujuan utama tahap ini adalah untuk meningkatkan kualitas sinyal, membuatnya lebih jelas, dan memfasilitasi analisis atau pemodelan lebih lanjut dengan menghilangkan gangguan yang tidak relevan. Proses *denoising* pada sinyal PCG pada penelitian ini melibatkan penggunaan salah satu teknik

pemrosesan sinyal, yaitu teknik filtering yang dapat mengurangi atau menghilangkan *noise* yang mungkin terkandung dalam sinyal tersebut.



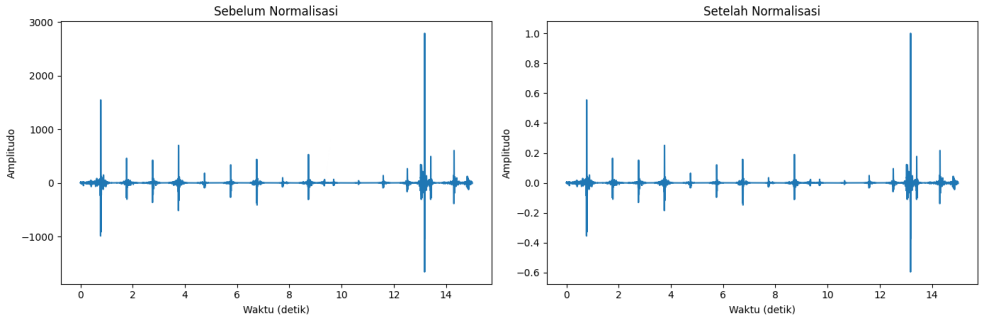
Gambar 3.7 Sinyal dengan kelas normal pada proses *denoising*



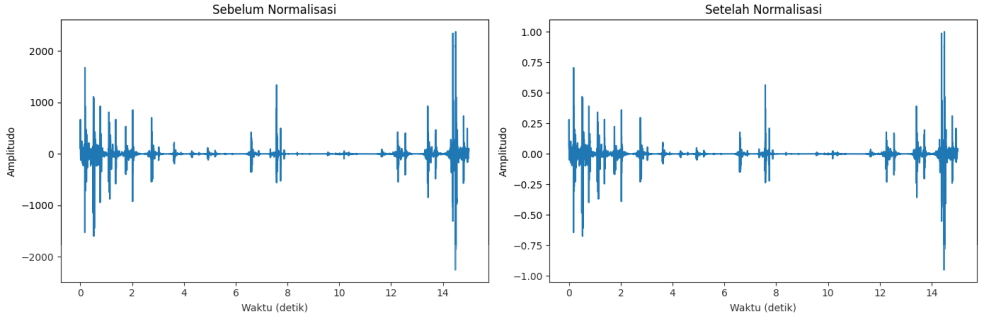
Gambar 3.8 Sinyal dengan kelas abnormal pada proses *denoising*

3.5.3 Normalisasi Data

Normalisasi data merupakan proses untuk mengubah skala atau rentang variabel dalam suatu dataset sehingga seluruh variabel memiliki skala yang seragam. Tujuan utama dari normalisasi adalah untuk memastikan bahwa setiap fitur atau variabel berkontribusi secara seimbang dalam analisis atau pemodelan, terlepas dari perbedaan skala asli. Proses ini membantu dalam menghindari dominasi variabel yang memiliki rentang nilai lebih besar dan memungkinkan model untuk menangkap pola yang lebih akurat. Pada umumnya, data yang telah dinormalisasi akan berada dalam rentang -1 hingga 1, sehingga mempermudah proses analisis dan meningkatkan performa algoritma.



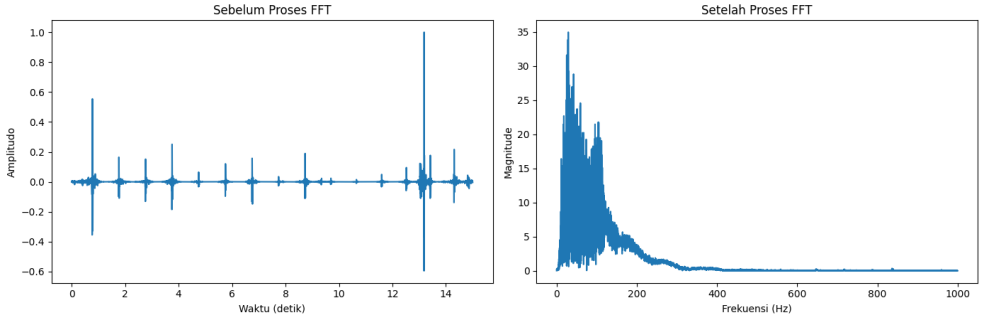
Gambar 3.9 Sinyal dengan kelas normal pada proses normalisasi data



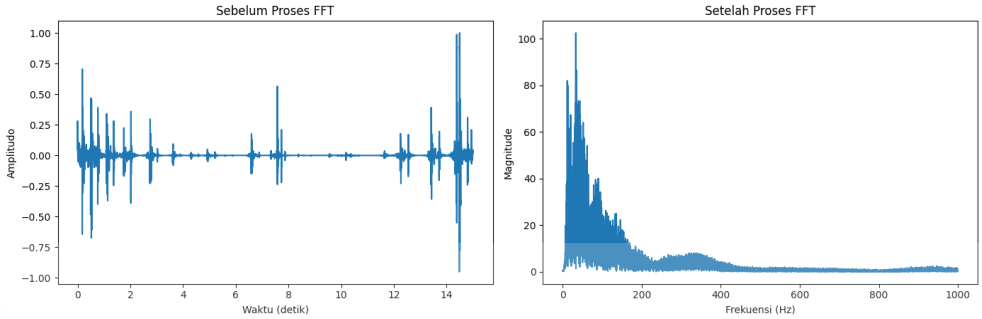
Gambar 3.10 Sinyal dengan kelas abnormal pada proses normalisasi data

3.5.4 *Fast Fourier Transform* (FFT)

Fast Fourier Transform (FFT) merupakan metode untuk mengonversi sinyal dari domain waktu ke domain frekuensi. Proses transformasi ini memungkinkan analisis distribusi frekuensi dari sinyal *phonocardiogram* (PCG) yang merekam bunyi jantung. Dengan menerapkan FFT, peneliti dapat mengevaluasi komponen frekuensi yang mendasari sinyal jantung, termasuk identifikasi bunyi jantung pertama (S1) dan bunyi jantung kedua (S2). Analisis spektrum frekuensi yang dihasilkan oleh FFT memberikan wawasan lebih dalam mengenai karakteristik suara jantung, yang mencakup informasi penting tentang kehadiran dan kekuatan puncak-puncak frekuensi. Selain itu, FFT dapat mengungkapkan adanya murmur, yang merupakan indikasi potensial dari kondisi abnormal pada sinyal PCG. Teknik ini tidak hanya memfasilitasi pemahaman komprehensif terhadap struktur frekuensi suara jantung, tetapi juga mendukung diagnosis dini melalui deteksi fitur yang mungkin tidak mudah dikenali dalam domain waktu.



Gambar 3.11 Sinyal dengan kelas normal pada proses *fast fourier transform*



Gambar 3.12 Sinyal dengan kelas abnormal pada proses *fast fourier transform*

Gambar 3.11 dan 3.12 memperlihatkan visualisasi sinyal dengan kelas normal dan abnormal setelah melalui proses *fast fourier transform* (FFT). Perbedaan dalam distribusi frekuensi antara sinyal normal dan abnormal dapat diidentifikasi dengan lebih mudah, seperti adanya puncak tambahan atau perubahan dalam intensitas frekuensi yang mengindikasikan kemunculan murmur pada sinyal abnormal. Identifikasi dan pengenalan karakteristik frekuensi ini menjadi krusial dalam pelatihan model karena memberikan pengetahuan yang lebih kuat bagi model untuk belajar dan mengenali pola spesifik yang membedakan sinyal normal dan sinyal abnormal. Dengan demikian, FFT membantu dalam mengekstraksi informasi yang lebih mendalam dari sinyal, yang dapat meningkatkan kemampuan model untuk mendeteksi pola dan anomali selama proses pelatihan, sehingga memaksimalkan efektivitas dan akurasi model dalam klasifikasi sinyal jantung.

3.5.5 Augmentasi Data

Augmentasi data merupakan langkah terakhir dalam proses *preprocessing* data sebelum data dilibatkan dalam pemodelan lebih lanjut. Teknik augmentasi data

digunakan untuk menciptakan variasi tambahan dalam dataset dengan memanipulasi, mengubah, atau menambahkan variasi pada data yang sudah ada. Tujuannya adalah untuk meningkatkan keberagaman dataset, sehingga model yang dibangun memiliki performa yang lebih baik dalam mengenali pola-pola secara akurat. Dengan augmentasi data, model diharapkan dapat lebih generalis terhadap variasi data yang tidak terlihat selama fase pelatihan.

Tabel 3.3 Jumlah data setiap kelas pada proses augmentasi

Kelas	Sebelum proses augmentasi	Setelah proses augmentasi
abnormal	1783	1000
normal	491	1000

Tabel 3.3 menunjukkan hasil proses augmentasi data. Proses ini dilakukan dengan menyeimbangkan jumlah data dalam setiap kelas sebelum pemodelan lebih lanjut. Dua kelas, ‘normal’ dan ‘abnormal’, diproses dengan memastikan folder output ada, menghitung jumlah file yang sudah ada, dan menambahkan file secara acak dari folder input hingga mencapai jumlah target. Sebelum augmentasi, kelas ‘abnormal’ memiliki 1783 data dan kelas ‘normal’ memiliki 491 data. Setelah augmentasi, kedua kelas memiliki masing-masing 1000 data, memastikan keseimbangan data untuk pelatihan model.

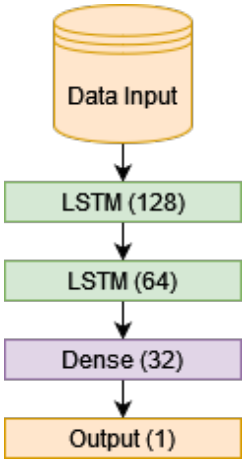
3.6 *Training* Data dengan Model

Setelah proses *preprocessing* data pada tahapan sebelumnya, di mana data dipersiapkan agar lebih siap untuk menjadi input model yang akan dibangun, pada tahap ini peneliti menentukan jumlah lapisan (*layer*) dan jumlah *neuron* dari setiap model yang digunakan.

3.6.1 Model Dasar/*Base Learner*

1) *Long Short Term Memory* (LSTM)

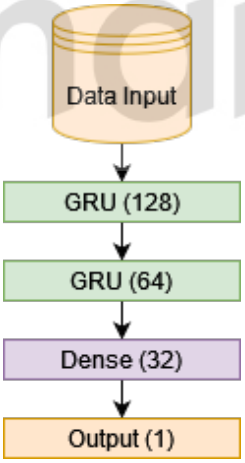
Model LSTM diterapkan dengan sejumlah lapisan dan *neuron* yang disesuaikan untuk menangani data urutan yang telah diproses. Parameter seperti jumlah lapisan tersembunyi dan *neuron* per lapisan dioptimalkan untuk mencapai performa terbaik.



Gambar 3.13 Diagram alir algoritma LSTM

2) *Gated Recurrent Unit* (GRU)

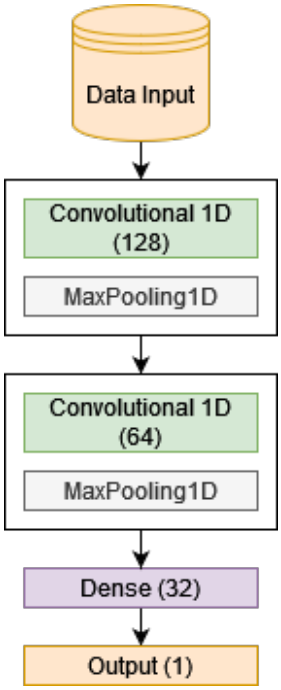
Model GRU, yang merupakan varian lebih sederhana dari LSTM, juga digunakan. Konfigurasi model mencakup penyesuaian jumlah lapisan dan *neuron* untuk memastikan efisiensi dan keefektifan dalam memproses data urutan sinyal jantung.



Gambar 3.14 Diagram alir algoritma GRU

3) *Temporal Convolutional Network* (TCN)

TCN digunakan untuk memproses data sekuensial dengan pendekatan *convolutional*. Jumlah lapisan dan filter ditentukan dan dioptimalkan untuk menangani *long-term dependencies* secara efisien.



Gambar 3.15 Diagram alir algoritma TCN

Gambar 3.13, 3.14, dan 3.15 menampilkan konfigurasi jumlah lapisan/*layer* dan *neuron* untuk setiap algoritma yang digunakan dalam penelitian ini. Peneliti menerapkan fungsi aktivasi *ReLU* untuk lapisan tersembunyi dan *sigmoid* untuk lapisan output, sesuai dengan praktik umum dalam jaringan saraf. Sebagai optimizer, 'Adam' dipilih dengan *learning rate* (lr) yang disesuaikan. Setiap model, yaitu LSTM, GRU, dan TCN, dilatih menggunakan variasi empat (4) jumlah *epoch* 10, 20, 30, dan satu percobaan menggunakan bantuan 'EarlyStopping' untuk menentukan nilai *epoch* yang optimal. Di mana untuk setiap variasi *epoch*, model dilatih sebanyak empat (4) kali dengan nilai *learning rate* yang berbeda-beda.

Pada 3 percobaan awal, nilai *learning rate* (lr) dipilih secara manual dengan nilai 0.01, 0.001, dan 0.0001. Percobaan keempat menggunakan strategi *auto learning rate* dengan bantuan 'ReduceLROnPlateau' dari *callback* yang termasuk dalam library Keras, yang memantau 'val_loss' (*loss* pada data validasi) dengan *patience*/atau tingkat kesabaran sebanyak 2 *epoch*. Jika 'val_loss' tidak menurun selama 2 *epoch* berturut-turut, *learning rate* akan dikurangi sebesar 0.01. Selama pelatihan, setiap model di-*compile* menggunakan *optimizer* 'Adam' dengan *learning rate* yang sesuai dengan percobaan yang sedang dilakukan, fungsi *loss*

'binary_crossentropy' untuk tugas klasifikasi biner, dan beberapa metrik untuk evaluasi performa model.

3.6.2 Model *Ensemble Learning*

Setelah model dasar (*base learner*) menjalani tahap pelatihan, prediksi probabilitas dari setiap model (LSTM, GRU, dan TCN) digabungkan menggunakan metode *soft voting* untuk menghasilkan prediksi akhir. Percobaan dilakukan empat kali dengan jumlah *epoch* 10, 20, 30, dan satu percobaan menggunakan bantuan 'EarlyStopping' untuk menentukan nilai *epoch*. Setiap *epoch* telah dilakukan dua kali dengan membedakan nilai bobot: 1:1:1 dan 2:1:1, di mana bobot 2 diberikan kepada model yang memiliki performa terbaik berdasarkan metrik evaluasi.

Prediksi probabilitas dari setiap model diberikan bobot sesuai dengan kontribusinya dalam *ensemble*. Bobot-bobot ini dinormalisasi sehingga total bobot menjadi 1. Prediksi probabilitas dari masing-masing model kemudian dikalikan dengan bobot yang sesuai dan dijumlahkan untuk mendapatkan prediksi *ensemble*. Probabilitas gabungan ini dibandingkan dengan ambang batas 0.5 untuk menentukan kelas akhir. Pendekatan ini diharapkan dapat meningkatkan akurasi prediksi dengan memanfaatkan kekuatan dari setiap model dasar yang berkontribusi dalam *ensemble*.

3.7 Evaluasi Model

Pada tahap ini, peneliti menggunakan salah satu metode yang cukup populer untuk mengetahui kinerja atau performa yang dimiliki oleh model yang telah dirancang dan melalui proses pelatihan, yaitu menggunakan metode *confusion matrix*. Dengan menggunakan nilai yang terdapat pada *confusion matrix*, beberapa metrik evaluasi populer dapat dihitung untuk memberikan gambaran komprehensif tentang kinerja model. *Accuracy* mencerminkan persentase total kelas yang diklasifikasikan dengan benar, sementara *precision* mengukur seberapa baik model mengidentifikasi positif dari kelas yang diklasifikasikan sebagai positif. *Recall* menggambarkan seberapa baik model mengidentifikasi semua kelas positif yang ada. Selanjutnya, *f1-score* menggabungkan nilai *precision* dan *recall* menjadi satu nilai yang memberikan gambaran menyeluruh tentang kinerja model. Apabila

keempat indikator metrik evaluasi dituliskan secara matematis, dapat dijabarkan sebagai berikut:

Accuracy (Akurasi)

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (3)$$

Precision (Presisi)

$$precision = \frac{TP}{TP + FP} \quad (4)$$

Recall (Sensitivitas)

$$recall = \frac{TP}{TP + FN} \quad (5)$$

F1-score

$$f1 - score = \frac{2 \times precision \times recall}{precision + recall} \quad (6)$$

Selain keempat metrik di atas, terdapat metrik lain yang umumnya Digunakan untuk memonitor seberapa baik kinerja model yang telah dibangun dan melalui proses pelatihan, yaitu ROC AUC (*Receiver Operating Characteristic Area Under the Curve*). ROC AUC merupakan metrik yang mengukur kemampuan model dalam membedakan antara kelas positif dan negatif. ROC *Curve* atau kurva ROC merupakan grafik yang menunjukkan hubungan antara *True Positive Rate* (TPR) dan *False Positive Rate* (FPR) pada berbagai *threshold* klasifikasi. AUC merupakan ukuran atau dimensi area di bawah kurva ROC. Nilai AUC berkisar antara 0 dan 1, di mana apabila nilainya semakin mendekati angka 1, maka model dapat dianggap sempurna karena selalu memprediksi kelas dengan benar. Secara matematis, metrik ROC AUC dapat dijabarkan sebagai berikut:

True Positive Rate (TPR)

$$TPR = \frac{TP}{TP + FN} \quad (7)$$

False Positive Rate (FPR)

$$FPR = \frac{FP}{FP + TN} \quad (8)$$

ROC AUC

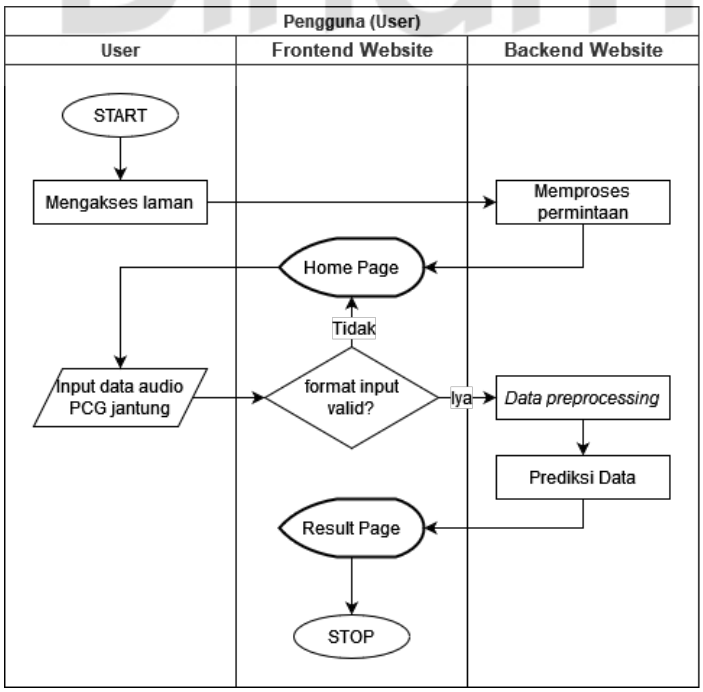
$$\text{ROC AUC} = \sum_{i=1}^{n-1} \left(\frac{TPR_i + TPR_{i+1}}{2} \right) . (FPR_{i+1} - FPR_i) \quad (9)$$

3.8 Pengembangan *Website*

Tahapan dalam pengembangan *website* dimulai dengan perancangan alur sistem hingga pengujian *website* guna menghasilkan *website* yang sesuai dengan tujuan utama pada penelitian ini. Proses pengembangan *website* ini meliputi perancangan alur sistem (*system flow*), pembuatan *wireframe*, penyusunan *entity relationship diagram* (ERD), serta diakhiri dengan implementasi pengembangan *backend* dan *frontend*.

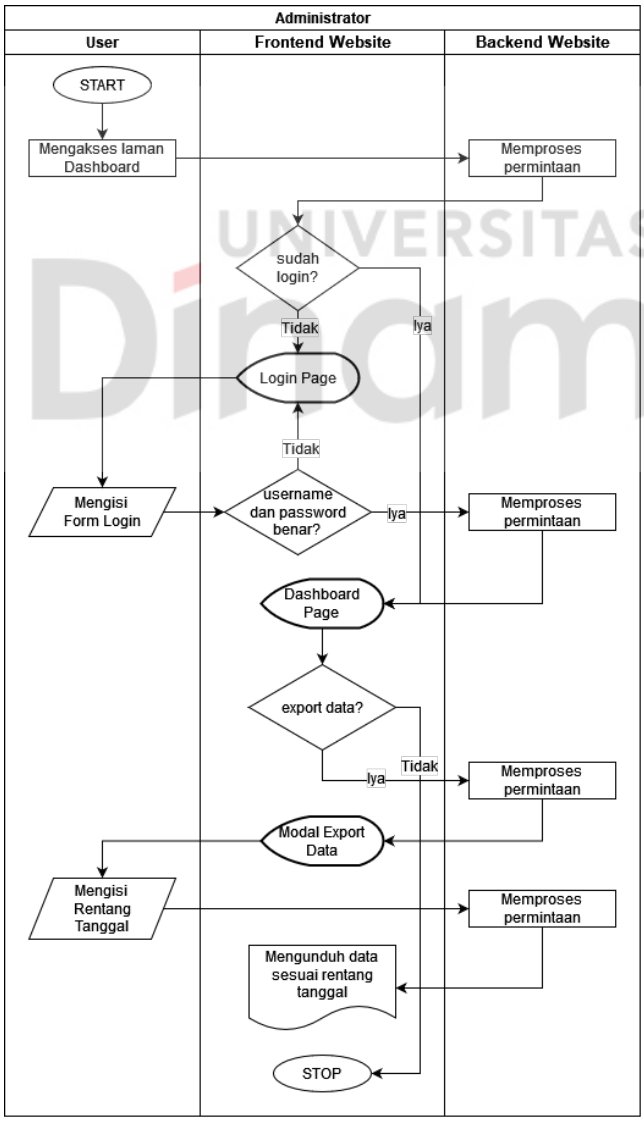
3.8.1 Alur Sistem (*System Flow*)

Alur sistem merupakan representasi visual dari proses kerja sistem yang menggambarkan bagaimana data mengalir dan bagaimana pengguna berinteraksi dengan sistem. Dalam konteks penelitian ini, terdapat dua alur sistem yang dirancang untuk pengguna (*user*) dan administrator.



Gambar 3.16 Diagram alur sistem untuk pengguna

Pertama, pengguna mengakses halaman utama website melalui peramban *web* mereka. Setelah tampilan halaman utama ditampilkan, pengguna diberikan opsi untuk mengunggah file yang ingin diproses oleh sistem. Setelah pengguna memilih dan mengunggah file, sistem *website* akan melakukan verifikasi terhadap format file yang diunggah untuk memastikan kesesuaiannya dengan format yang telah ditentukan sebelumnya. Jika sistem menentukan bahwa format file sesuai, pengguna akan diarahkan ke halaman hasil yang menampilkan output atau informasi yang relevan berdasarkan data yang diunggah. Namun, jika sistem mendeteksi bahwa format file tidak sesuai, pengguna akan diberitahu melalui pesan peringatan yang jelas dan dipandu untuk kembali ke halaman utama dengan instruksi untuk mengunggah kembali file dengan format yang benar.



Gambar 3.17 Diagram alur sistem untuk administrator

Administrator memulai dengan proses *login* ke dalam sistem menggunakan kredensial yang sah. Setelah kredensial dimasukkan, sistem melakukan verifikasi untuk memvalidasi informasi *login* yang diberikan. Jika informasi yang dimasukkan tidak sesuai dengan data yang tersimpan, sistem akan memberikan pesan kesalahan autentikasi yang jelas dan meminta administrator untuk mencoba lagi. Jika proses autentikasi berhasil, administrator akan diarahkan ke halaman *dashboard* yang merupakan pusat kontrol administratif. Di dalam *dashboard*, administrator dapat melihat *log* kegiatan pengunggahan file oleh pengguna dalam bentuk tabel yang mencakup detail seperti waktu pengunggahan, nama file, dan status verifikasi format. Selain itu, administrator memiliki kemampuan untuk mengekspor data *log* kegiatan dalam format Excel sesuai kebutuhan.

Selama penelitian, terdapat satu halaman tambahan yang hanya dapat diakses oleh administrator. Halaman ini dirancang khusus untuk memonitor kinerja model dalam mengklasifikasikan file atau dokumen yang diunggah. Administrator dapat membandingkan hasil klasifikasi model dengan nilai label aktual dari data tersebut. Halaman ini memberikan administrator akses untuk memantau performa model secara langsung dan mengambil tindakan atau keputusan berdasarkan hasil yang diperoleh. Penambahan halaman khusus ini memperluas fungsionalitas sistem yang dibangun untuk memenuhi kebutuhan pengawasan dan evaluasi yang mendalam dalam konteks pengembangan *website* dan model *machine learning* pada penelitian ini.

3.8.2 *Wireframe*

Wireframe menjadi komponen yang cukup penting dalam pengembangan *website* karena dapat menyediakan representasi visual awal dari tata letak dan struktur elemen antarmuka pengguna tanpa detail desain visual. Hal ini dapat membantu peneliti merencanakan navigasi dan interaksi pengguna secara efisien dan memvalidasi konsep fungsionalitas sebelum implementasi lebih lanjutl. Dengan demikian, *wireframe* tidak hanya mengarah pada pengembangan antarmuka yang lebih terstruktur dan efektif, tetapi juga meminimalkan risiko perubahan di tahap akhir pengembangan.

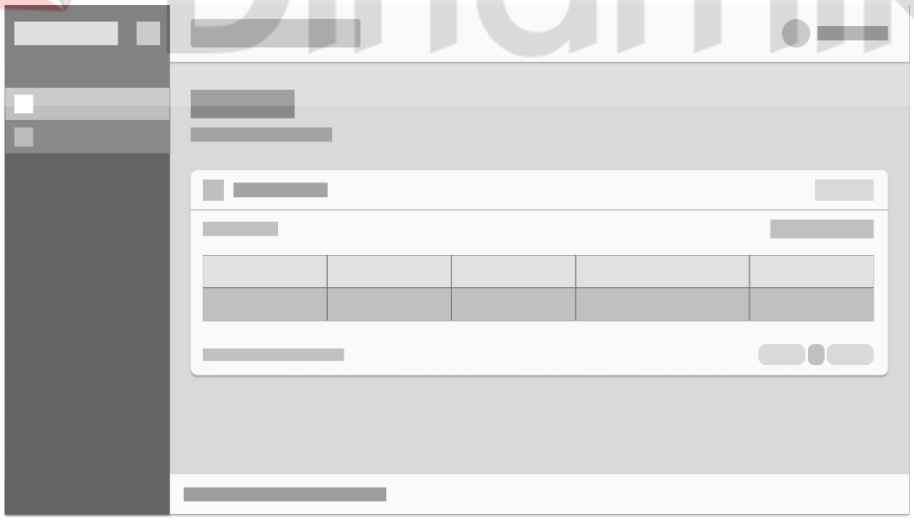
Gambar 3.18 *Wireframe* halaman utama

Gambar 3.19 *Wireframe* halaman hasil prediksi

Gambar 3.18 dan Gambar 3.19 menggambarkan *wireframe* dari halaman utama dan halaman hasil prediksi yang dapat diakses oleh pengguna. Pada Gambar 3.18, halaman utama menampilkan formulir sederhana yang memungkinkan pengguna untuk mengunggah file audio yang akan diprediksi oleh model. Pengguna dapat memilih file dari perangkat mereka dan mengunggahnya melalui antarmuka ini. Sedangkan pada Gambar 3.19, halaman hasil prediksi menampilkan hasil yang dihasilkan oleh model setelah memproses file audio yang diunggah. Hasil prediksi akan muncul dalam bentuk label yang mengindikasikan status file tersebut, apakah 'normal' atau 'abnormal'.

Gambar 3.20 *Wireframe* halaman *login*

Gambar 3.20 menunjukkan *wireframe* halaman *login* yang digunakan oleh administrator untuk mengakses bagian administratif dari sistem. Halaman ini mencakup elemen seperti kolom input untuk username dan password serta tombol untuk mengirimkan data login. Jika kredensial yang dimasukkan benar, administrator akan diarahkan ke halaman *dashboard*.

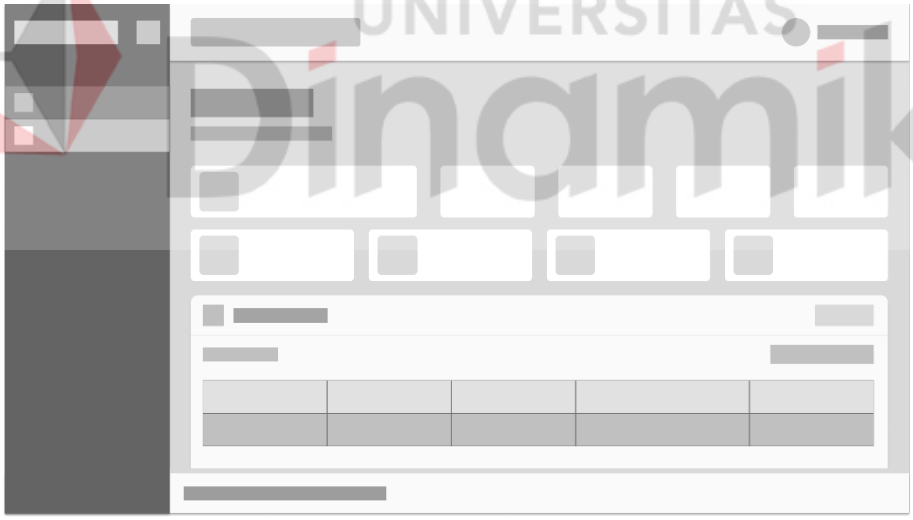


Gambar 3.21 *Wireframe* halaman *dashboard*

Gambar 3.21 memperlihatkan *wireframe* halaman *dashboard*, di mana administrator dapat memantau aktivitas pengguna dan mengelola sistem. *Dashboard* ini menampilkan *log* kegiatan pengunggahan file oleh pengguna dalam

format tabel yang mencakup detail seperti waktu pengunggahan, nama file, dan status verifikasi format. *Dashboard* juga menyediakan fitur untuk mengekspor data *log* kegiatan dalam format Excel, yang ditunjukkan pada Gambar 3.22.

Gambar 3.22 *Wireframe* aksi *export* data



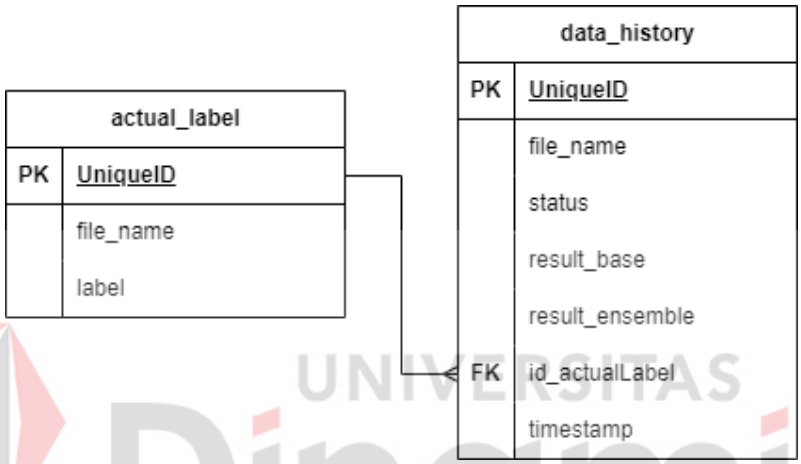
Gambar 3.23 *Wireframe* halaman pengembangan

Gambar 3.23 menggambarkan *wireframe* halaman pengembangan, di mana administrator dapat melakukan pemantauan terkait performa model dalam memprediksi data yang diunggah oleh pengguna. Halaman ini menampilkan nilai dari *confusion matrix*, termasuk perhitungan empat (4) metrik yang paling umum digunakan untuk pemantauan performa model yang telah dilatih. Metrik tersebut

meliputi *accuracy*, *precision*, *recall*, dan *f1-score*, yang semuanya memberikan wawasan penting mengenai efektivitas model dalam melakukan klasifikasi.

3.8.3 *Entity Relational Diagram* (ERD)

Entity Relational Diagram (ERD) merupakan representasi visual dari struktur basis data yang digunakan dalam pengembangan *website*. ERD memodelkan entitas, atribut, dan hubungan antar entitas untuk memastikan desain basis data yang efisien dan terstruktur.



Gambar 3.24 *Entity relational diagram*

Tabel ‘actual_label’ dibuat untuk menyimpan informasi mengenai label aktual dari file-file yang diunggah ke dalam sistem. Tabel ini terdiri dari tiga kolom utama: ‘UniqueID’ sebagai kunci utama yang unik, ‘file_name’ untuk menyimpan nama file, dan ‘label’ untuk menyimpan label yang terkait dengan file tersebut.

Tabel ‘data_history’ dirancang untuk mencatat detail pengunggahan file dan proses pengklasifikasian yang dilakukan oleh sistem. Struktur tabel ini mencakup beberapa kolom kunci: ‘UniqueID’ sebagai kunci utama unik, ‘file_name’ untuk menyimpan nama file, ‘status’ yang mencerminkan status terkini dari proses pengolahan file, ‘result_base’ menyimpan hasil klasifikasi model dasar dari file yang telah diunggah, ‘result_ensemble’ menyimpan hasil klasifikasi model *ensemble* dari file yang telah diunggah, dan ‘timestamp’ yang mencatat waktu pengunggahan file.

Tabel ‘data_history’ juga memuat kolom ‘id_actualLabel’ yang menghubungkan dengan tabel ‘actual_label’. Kolom ini memastikan setiap entri dalam tabel ‘data_history’ terhubung dengan entri yang sesuai dalam tabel ‘actual_label’, menggunakan referensi ke id dalam tabel ‘actual_label’.

3.8.4 Implementasi Pengembangan *Backend* dan *Frontend*

Dalam pengembangan sistem berbasis *website* untuk klasifikasi sinyal *phonocardiogram* (PCG) menggunakan teknik *ensemble learning*, langkah-langkah implementasi *backend* dan *frontend* juga cukup krusial. Bagian ini menjelaskan bagaimana tahapan aplikasi Flask digunakan untuk membangun komponen *backend* dan *frontend*, termasuk penggunaan *library*, dan struktur folder pada aplikasi *website*. Di mana pada tahap sebelumnya, instalasi *environment* telah dilakukan. Berikut adalah tahapan yang dilakukan sebelum membangun *backend* dan *frontend* antarmuka *website*:

1) Mengaktifkan *virtual environment* dan instalasi *library*

   Langkah pertama adalah mengaktifkan *virtual environment* dan menginstal *library* yang diperlukan untuk pengembangan. *Virtual environment* membantu mengisolasi dependensi proyek sehingga konsistensi lingkungan pengembangan terjaga. Perintah untuk mengaktifkan *virtual environment* dan instalasi beberapa *library* adalah sebagai berikut:

```
source venv/bin/activate    # Untuk Linux/macOS
venv\Scripts\activate       # Untuk Windows

pip install -r requirements.txt
```

   Di mana dokumen atau file ‘requirements.txt’ adalah berkas yang berisi daftar *library* yang diperlukan untuk proyek, antara lain:

```
Flask==2.3.3
python-dotenv==1.0.0
Flask-MySQLdb==0.2.0
pandas==2.0.3
xlsxwriter==3.1.4
noisereduce==2.0.1
scipy==1.11.2
pydub==0.25.1
```

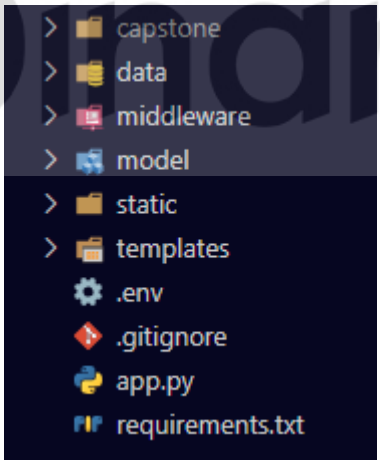
```
keras==2.13.1
tensorflow==2.13.0
numpy==1.24.3
Werkzeug==3.0.1
```

2) Konfigurasi dokumen .env

Berkas .env digunakan untuk menyimpan informasi konfigurasi sensitif seperti kredensial basis data. Berikut adalah isi dari berkas .env pada penelitian ini:

```
MYSQL_HOST=localhost
MYSQL_USER=root
MYSQL_PASSWORD=password
MYSQL_DB=heartdisease
SECRET_KEY=supersecretkey
```

Setelah melakukan kedua hal di atas, selanjutnya peneliti melakukan implementasi logika yang digunakan pada proses antarmuka *website*, serta mengimplementasikan *wireframe* yang sebelumnya telah disusun. Sehingga susunan folder dokumen pada *website* yang dapat dilihat pada Gambar 3.25.



Gambar 3.25 Struktur folder pada pengembangan antarmuka *website*

Di mana folder ‘capstone’ merupakan *virtual environment* yang digunakan, folder ‘data’ yang menyimpan *preprocessing* data, folder ‘middleware’ yang berisi beberapa logika untuk manajemen *login* pengguna, folder ‘model’ yang menyimpan model *machine learning*, folder ‘static’ menyimpan berkas *static* seperti file css dan javascript, dan folder ‘templates’ berisi *template* HTML untuk ditampilkan kepada

pengguna. Selain itu juga terdapat beberapa file, file '.env' berisi informasi konfigurasi sensitif seperti *database*, file '.gitignore' yang menyimpan daftar file dan folder yang diabaikan oleh Git, file 'app.py' merupakan berkas utama aplikasi Flask yang berisi logika *routing* dan penanganan permintaan HTTP dari pengguna, serta file 'requirements.txt' menyimpan daftar dependensi Python yang diperlukan.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil Proses *Training* Model Dasar/*Base Learner*

Pada penelitian ini, proses *training* model dasar dilakukan menggunakan tiga jenis model, yaitu LSTM, GRU, dan TCN. Setiap model diuji dalam empat (4) percobaan berbeda dengan variasi jumlah *epoch*, yaitu 10, 20, 30, dan satu percobaan menggunakan bantuan 'EarlyStopping'. 'EarlyStopping' merupakan salah satu fungsi *callback* dari *library* Keras yang digunakan untuk menghentikan proses *training* secara otomatis ketika performa model berhenti membaik setelah sejumlah *epoch* tertentu. Masing-masing percobaan juga dilakukan dengan empat (4) variasi nilai *learning rate*, yaitu 0.01, 0.001, 0.0001, dan auto_lr. Variabel auto_lr merupakan implementasi dari *callback* 'ReduceLROnPlateau' yang termasuk dalam *library* Keras, yang secara otomatis menentukan nilai *learning rate* paling optimal untuk proses pelatihan tersebut.

Indikator yang digunakan sebagai parameter penggunaan fungsi *callback* 'EarlyStopping' dan 'ReduceLROnPlateau' adalah nilai *val_loss*, atau nilai *loss* untuk data validasi saat proses *training*, dengan nilai *patience* untuk 'EarlyStopping' adalah 5, sedangkan untuk 'ReduceLROnPlateau' adalah 2, dengan nilai minimum *learning rate* yang ditentukan adalah sebesar 1.0000e-04. Untuk mengetahui model yang memiliki performa terbaik di setiap percobaan dengan variasi nilai *epoch*, penelitian ini menggunakan nilai dari metrik evaluasi akurasi dan *f1-score* sebagai indikatornya, dengan berbagai persentase bobot antara lain 50:50, 60:40, 70:30, 80:20, hingga 90:10.

Keputusan memilih akurasi dan *f1-score*, karena akurasi memberikan gambaran umum mengenai seberapa baik model dapat mengklasifikasikan data secara keseluruhan, sedangkan *f1-score* memberikan keseimbangan antara *precision* dan *recall*, yang penting untuk menangani kelas yang tidak seimbang. Dengan menggunakan kedua metrik ini, dapat diidentifikasi tidak hanya sejauh mana model dapat mengidentifikasi data dengan benar tetapi juga seberapa baik model menangani keseimbangan nilai antara *false positives* dan *false negatives*.

4.1.1 Hasil Model *Long Short Term Memory* (LSTM)

1) *Epoch* 10

Pada percobaan dengan menggunakan 10 *epoch*, Tabel 4.1 menunjukkan hasil variasi kinerja model LSTM dengan berbagai laju pembelajaran (*learning rate*). Penurunan *learning rate* terbukti berkontribusi pada peningkatan kinerja model, dengan *learning rate* bernilai 0.0001 memberikan performa terbaik dalam mendeteksi kedua kelas. Selain itu, metode otomatisasi *learning rate* juga cukup efektif sehingga mencapai hasil yang mendekati optimal.

Tabel 4.1 Hasil pelatihan model LSTM dengan 10 *epoch*

Kelas	*Learning rate*		*Accuracy*	*Precision*	*Recall*	*F1-score*
normal	manual	0.01	0.78	0.86	0.67	0.75
abnormal				0.73	0.89	0.80
normal	manual	0.001	0.83	0.78	0.91	0.84
abnormal				0.89	0.75	0.81
normal	manual	0.0001	0.90	0.87	0.95	0.91
abnormal				0.94	0.86	0.90
normal	auto_lr	1.0000e-04	0.88	0.86	0.91	0.89
abnormal				0.90	0.95	0.88

Tabel 4.2 Perbandingan performa model LSTM dengan 10 *epoch*

Bobot presentase	*Learning rate*			
	0.01	0.001	0.0001	1.0000e-04
accuracy : 50% *f1-score* : 50%	0.765	0.835	0.905	0.885
accuracy : 60% *f1-score* : 40%	0.768	0.834	0.904	0.884
accuracy : 70% *f1-score* : 30%	0.771	0.833	0.903	0.883
accuracy : 80% *f1-score* : 20%	0.774	0.832	0.902	0.882
accuracy : 90% *f1-score* : 10%	0.777	0.831	0.901	0.881

Tabel 4.2 menunjukkan bahwa performa model LSTM setelah melalui proses pelatihan dengan 10 *epoch* bervariasi berdasarkan kombinasi bobot presentase antara akurasi dan *f1-score*. Ketika menggunakan nilai *learning rate* sebesar 0.0001, model LSTM menunjukkan performa terbaik di semua kombinasi bobot presentase, baik itu dengan penekanan yang lebih tinggi pada metrik akurasi, maupun nilai bobot yang sama antara akurasi dan *f1-score*.

2) *Epoch* 20

Pada percobaan dengan menggunakan 20 *epoch*, Tabel 4.3 menunjukkan hasil variasi kinerja model LSTM dengan berbagai laju pembelajaran (*learning rate*). Pola penurunan nilai *learning rate* yang diamati mirip dengan temuan pada percobaan dengan 10 *epoch*. Model dengan *learning rate* 0.0001 menunjukkan performa terbaik dalam hal akurasi dan *f1-score*, mirip dengan hasil sebelumnya. Metode otomatisasi *learning rate* juga memberikan hasil yang mendekati performa optimal, menunjukkan bahwa metode ini masih efektif pada nilai *epoch* yang semakin tinggi.

Tabel 4.3 Hasil pelatihan model LSTM dengan 20 *epoch*

Kelas	*Learning rate*		*Accuracy*	*Precision*	*Recall*	*F1-score*
normal	manual	0.01	0.83	0.82	0.84	0.83
abnormal				0.83	0.82	0.83
normal	manual	0.001	0.84	0.90	0.76	0.82
abnormal				0.79	0.91	0.85
normal	manual	0.0001	0.92	0.90	0.95	0.92
abnormal				0.94	0.89	0.92
normal	auto_lr	1.0000e-04	0.91	0.88	0.94	0.91
abnormal				0.93	0.87	0.90

Tabel 4.4 Perbandingan performa model LSTM dengan 20 *epoch*

Bobot presentase	*Learning rate*			
	0.01	0.001	0.0001	1.0000e-04
accuracy : 50% *f1-score* : 50%	0.83	0.83	0.92	0.91
accuracy : 60% *f1-score* : 40%	0.83	0.832	0.92	0.91
accuracy : 70% *f1-score* : 30%	0.83	0.834	0.92	0.91
accuracy : 80% *f1-score* : 20%	0.83	0.836	0.92	0.91
accuracy : 90% *f1-score* : 10%	0.83	0.838	0.92	0.91

Tabel 4.4 menunjukkan bahwa performa model LSTM setelah melalui proses pelatihan dengan 20 *epoch* bervariasi berdasarkan kombinasi bobot presentase antara akurasi dan *f1-score*. Model dengan *learning rate* sebesar 0.0001 mencapai nilai terbaik dalam hal akurasi dan *f1-score* pada semua kombinasi bobot

presentase. Selain itu, penggunaan metode otomatisasi *learning rate* (auto_lr) juga menunjukkan hasil yang konsisten dengan performanya yang mendekati optimal.

3) *Epoch* 30

Pada percobaan dengan menggunakan 30 *epoch*, Tabel 4.5 menunjukkan hasil variasi kinerja model LSTM dengan berbagai laju pembelajaran (*learning rate*), sejalan dengan pola yang telah diamati pada *epoch* 10 dan 20. Model dengan *learning rate* 0.0001 terus menunjukkan performa terbaik dalam hal akurasi dan *f1-score*, mirip dengan hasil sebelumnya. Selain itu, metode otomatisasi *learning rate* mengalami penurunan, sebab memberikan hasil yang lebih rendah dari nilai *learning rate* 0.001.

Tabel 4.5 Hasil pelatihan model LSTM dengan 30 *epoch*

Kelas	*Learning rate*		*Accuracy*	*Precision*	*Recall*	*F1-score*
normal	manual	0.01	0.79	0.86	0.70	0.77
abnormal				0.74	0.88	0.81
normal	manual	0.001	0.89	0.86	0.93	0.89
abnormal				0.92	0.85	0.88
normal	manual	0.0001	0.92	0.90	0.95	0.92
abnormal				0.94	0.90	0.92
normal	auto_lr	1.0000e-04	0.88	0.86	0.92	0.89
abnormal				0.91	0.85	0.88

Tabel 4.6 Perbandingan performa model LSTM dengan 30 *epoch*

Bobot presentase	*Learning rate*			
	0.01	0.001	0.0001	1.0000e-04
accuracy : 50% *f1-score* : 50%	0.78	0.89	0.92	0.885
accuracy : 60% *f1-score* : 40%	0.782	0.89	0.92	0.884
accuracy : 70% *f1-score* : 30%	0.784	0.89	0.92	0.883
accuracy : 80% *f1-score* : 20%	0.786	0.89	0.92	0.882
accuracy : 90% *f1-score* : 10%	0.788	0.89	0.92	0.881

Tabel 4.6 menunjukkan bahwa performa model LSTM setelah melalui proses pelatihan dengan 30 *epoch* tetap konsisten dengan pola yang telah diamati pada *epoch* sebelumnya. Model dengan *learning rate* sebesar 0.0001 terus

memberikan hasil terbaik dalam hal akurasi dan *f1-score* pada semua kombinasi bobot presentase. Metode otomatisasi *learning rate* juga tetap memberikan performa yang mendekati optimal, menunjukkan bahwa metode ini masih efektif pada *epoch* yang lebih tinggi.

4) Auto *Epoch*

Pada percobaan ini, nilai *epoch* ditentukan secara otomatis menggunakan *callback* 'EarlyStopping'. Hasil pelatihan model LSTM menunjukkan variasi kinerja yang bergantung pada *learning rate* yang digunakan. Secara umum, model menunjukkan performa yang baik dengan akurasi dan *f1-score* yang konsisten. Namun, hasil terbaik dicapai dengan *learning rate* yang lebih kecil. *Epoch* yang bernilai 15 dengan *learning rate* 0.0001 memberikan performa terbaik, dengan akurasi dan *f1-score* tertinggi. Ini menunjukkan bahwa semakin kecil nilai *learning rate* membantu model dalam mencapai deteksi kelas yang lebih baik. Sebaliknya, *epoch* yang bernilai 96 dengan *learning rate* (auto_lr) memberikan hasil yang konsisten tetapi sedikit lebih rendah dibandingkan dengan *learning rate* 0.0001. Ini menegaskan bahwa pemilihan *learning rate* yang tepat memainkan peran penting dalam mencapai performa model yang optimal.

Tabel 4.7 Hasil pelatihan model LSTM dengan auto *epoch*

Epoch	Kelas	*Learning rate*		*Accuracy*	*Precision*	*Recall*	*F1-score*
8	normal	manual	0.01	0.83	0.85	0.81	0.83
	abnormal				0.82	0.85	0.84
8	normal	manual	0.001	0.87	0.86	0.89	0.87
	abnormal				0.88	0.85	0.87
15	normal	manual	0.0001	0.91	0.90	0.94	0.91
	abnormal				0.93	0.89	0.91
96	normal	auto_lr	1.0000e-04	0.87	0.86	0.88	0.87
	abnormal				0.88	0.86	0.87

Tabel 4.8 menunjukkan bahwa performa model LSTM dengan auto *epoch* tetap konsisten dalam memberikan hasil terbaik pada *learning rate* 0.0001 dan memberikan performa yang mendekati optimal pada *learning rate* (auto_lr). Hal ini menegaskan bahwa penggunaan *callback* 'EarlyStopping' bersama dengan variansi

nilai *learning rate* yang sesuai sangat penting untuk mencapai hasil pelatihan yang efektif.

Tabel 4.8 Perbandingan performa model LSTM dengan auto *epoch*

Bobot presentase	*Learning rate*			
	0.01	0.001	0.0001	1.0000e-04
accuracy : 50% *f1-score* : 50%	0.83	0.87	0.91	0.87
accuracy : 60% *f1-score* : 40%	0.83	0.87	0.91	0.87
accuracy : 70% *f1-score* : 30%	0.83	0.87	0.91	0.87
accuracy : 80% *f1-score* : 20%	0.83	0.87	0.91	0.87
accuracy : 90% *f1-score* : 10%	0.83	0.87	0.91	0.87

Secara keseluruhan pada empat (4) percobaan yang membedakan nilai *epoch*, model LSTM menunjukkan hasil yang konsisten dalam respons terhadap perubahan laju pembelajaran (*learning rate*). Pada setiap pengujian dengan nilai *epoch* 10, 20, 30, dan dengan penggunaan *callback* 'EarlyStopping', penurunan *learning rate* secara jelas berkontribusi pada peningkatan kinerja model. *Learning rate* bernilai 0.0001 secara konsisten memberikan performa terbaik dalam hal akurasi dan *f1-score* di semua *epoch* yang diuji. Metode otomatisasi *learning rate* (auto_lr) yang diterapkan dengan *callback* 'EarlyStopping', juga memberikan hasil yang mendekati optimal.

4.1.2 Hasil Model *Gated Recurrent Unit* (GRU)

1) *Epoch* 10

Tabel 4.9 menunjukkan bahwa menurunkan nilai *learning rate* berkontribusi pada peningkatan kinerja model GRU, dengan *learning rate* bernilai 0.0001 memberikan performa terbaik dalam mendeteksi kedua kelas. Model dengan *learning rate* 0.0001 mencapai akurasi dan *f1-score* tertinggi dibandingkan dengan *learning rate* lainnya. Meskipun metode otomatisasi *learning rate* (auto_lr) juga efektif dalam menyesuaikan laju pembelajaran, hasilnya tidak mencapai performa terbaik yang ditunjukkan oleh *learning rate* 0.0001.

Tabel 4.9 Hasil pelatihan model GRU dengan 10 *epoch*

Kelas	*Learning rate*		*Accuracy*	*Precision*	*Recall*	*F1-score*
normal	manual	0.01	0.81	0.79	0.86	0.82
abnormal				0.84	0.76	0.80
normal	manual	0.001	0.87	0.85	0.90	0.87
abnormal				0.89	0.84	0.87
normal	manual	0.0001	0.88	0.81	0.96	0.88
abnormal				0.95	0.78	0.86
normal	auto_lr	1.0000e-04	0.87	0.83	0.92	0.87
abnormal				0.91	0.81	0.86

Tabel 4.10 Perbandingan performa model GRU dengan 10 *epoch*

Bobot presentase	*Learning rate*			
	0.01	0.001	0.0001	1.0000e-04
accuracy : 50% *f1-score* : 50%	0.815	0.87	0.88	0.87
accuracy : 60% *f1-score* : 40%	0.814	0.87	0.88	0.87
accuracy : 70% *f1-score* : 30%	0.813	0.87	0.88	0.87
accuracy : 80% *f1-score* : 20%	0.812	0.87	0.88	0.87
accuracy : 90% *f1-score* : 10%	0.811	0.87	0.88	0.87

Tabel 4.10 menunjukkan bahwa performa model GRU setelah melalui proses pelatihan dengan 10 *epoch* bervariasi berdasarkan kombinasi bobot presentase antara akurasi dan *f1-score*. *Learning rate* yang lebih kecil, terutama 0.0001, memberikan hasil terbaik pada semua kombinasi bobot presentase, sedangkan auto_lr menghasilkan performa yang lebih rendah.

2) *Epoch* 20

Pada percobaan dengan menggunakan 20 *epoch*, penurunan *learning rate* secara konsisten meningkatkan kinerja model GRU pada setiap pengujian. Model mencapai performa terbaik pada *learning rate* 0.0001, dengan peningkatan yang signifikan dalam metrik *precision* dan *recall* untuk kedua kelas. Metode otomatisasi *learning rate* (auto_lr) juga memberikan hasil yang mendekati optimal, namun tidak melampaui performa terbaik yang dicapai dengan *learning rate* 0.0001.

Tabel 4.11 Hasil pelatihan model GRU dengan 20 *epoch*

Kelas	*Learning rate*		*Accuracy*	*Precision*	*Recall*	*F1-score*
normal	manual	0.01	0.80	0.85	0.74	0.79
abnormal				0.77	0.86	0.81
normal	manual	0.001	0.85	0.88	0.82	0.85
abnormal				0.83	0.89	0.86
normal	manual	0.0001	0.94	0.93	0.94	0.94
abnormal				0.94	0.93	0.93
normal	auto_lr	1.0000e-04	0.89	0.88	0.92	0.90
abnormal				0.91	0.87	0.89

Tabel 4.12 Perbandingan performa model GRU dengan 20 *epoch*

Bobot presentase	*Learning rate*			
	0.01	0.001	0.0001	1.0000e-04
accuracy : 50% *f1-score* : 50%	0.795	0.85	0.94	0.895
accuracy : 60% *f1-score* : 40%	0.796	0.85	0.94	0.894
accuracy : 70% *f1-score* : 30%	0.797	0.85	0.94	0.893
accuracy : 80% *f1-score* : 20%	0.798	0.85	0.94	0.892
accuracy : 90% *f1-score* : 10%	0.799	0.85	0.94	0.891

Tabel 4.12 menunjukkan bahwa performa model GRU setelah melalui proses pelatihan dengan 20 *epoch* bervariasi berdasarkan kombinasi bobot presentase antara akurasi dan *f1-score*. *Learning rate* sebesar 0.0001 memberikan hasil terbaik dalam hal akurasi dan *f1-score* di semua kombinasi bobot presentase. Metode otomatisasi *learning rate* (auto_lr) menghasilkan performa yang mendekati optimal, namun tetap di bawah hasil terbaik yang dicapai dengan *learning rate* bernilai 0.0001.

3) *Epoch* 30

Pada percobaan menggunakan 30 *epoch*, penurunan *learning rate* secara konsisten meningkatkan kinerja model GRU pada setiap pengujian. Model mencapai performa terbaik pada *learning rate* 0.0001, dengan peningkatan signifikan dalam metrik *precision* dan *f1-score* untuk kedua kelas. Metode otomatisasi *learning rate* juga memberikan hasil yang mendekati optimal,

meskipun tidak melampaui performa terbaik yang dicapai dengan *learning rate* bernilai 0.0001.

Tabel 4.13 Hasil pelatihan model GRU dengan 30 *epoch*

Kelas	*Learning rate*		*Accuracy*	*Precision*	*Recall*	*F1-score*
normal	manual	0.01	0.78	0.84	0.71	0.77
abnormal				0.74	0.86	0.80
normal	manual	0.001	0.83	0.78	0.93	0.85
abnormal				0.91	0.73	0.81
normal	manual	0.0001	0.93	0.91	0.94	0.93
abnormal				0.94	0.91	0.92
normal	auto_lr	1.0000e-04	0.92	0.90	0.94	0.92
abnormal				0.93	0.89	0.91

Tabel 4.14 Perbandingan performa model GRU dengan 30 *epoch*

Bobot presentase	*Learning rate*			
	0.01	0.001	0.0001	1.0000e-04
accuracy : 50% *f1-score* : 50%	0.775	0.84	0.93	0.92
accuracy : 60% *f1-score* : 40%	0.776	0.838	0.93	0.92
accuracy : 70% *f1-score* : 30%	0.777	0.836	0.93	0.92
accuracy : 80% *f1-score* : 20%	0.778	0.834	0.93	0.92
accuracy : 90% *f1-score* : 10%	0.779	0.832	0.93	0.92

Tabel 4.14 menunjukkan bahwa performa model GRU setelah melalui proses pelatihan dengan 30 *epoch* bervariasi berdasarkan kombinasi bobot presentase antara akurasi dan *f1-score*. *Learning rate* sebesar 0.0001 memberikan hasil terbaik dalam hal akurasi dan *f1-score* di semua kombinasi bobot presentase.

4) Auto *Epoch*

Percobaan ini menggunakan auto *epoch* yang menunjukkan bahwa performa model GRU sangat dipengaruhi oleh jumlah *epoch* dan *learning rate* yang digunakan. Model melakukan pelatihan dengan durasi *epoch* yang bervariasi secara otomatis untuk mencapai performa terbaik. Dan model dapat mencapai performa terbaik pada saat *learning rate* bernilai 0.0001.

Tabel 4.15 Hasil pelatihan model GRU dengan auto *epoch*

Epoch	Kelas	*Learning rate*		*Accuracy*	*Precision*	*Recall*	*F1-score*
18	normal	manual	0.01	0.82	0.87	0.77	0.81
	abnormal				0.79	0.88	0.83
16	normal	manual	0.001	0.87	0.85	0.90	0.88
	abnormal				0.89	0.84	0.87
14	normal	manual	0.0001	0.92	0.90	0.95	0.92
	abnormal				0.95	0.89	0.92
48	normal	auto_lr	1.0000e-04	0.89	0.87	0.93	0.90
	abnormal				0.92	0.86	0.89

Tabel 4.16 Perbandingan performa model GRU dengan auto *epoch*

Bobot presentase	*Learning rate*			
	0.01	0.001	0.0001	1.0000e-04
accuracy : 50% *f1-score* : 50%	0.815	0.875	0.92	0.895
accuracy : 60% *f1-score* : 40%	0.816	0.874	0.92	0.894
accuracy : 70% *f1-score* : 30%	0.817	0.873	0.92	0.893
accuracy : 80% *f1-score* : 20%	0.818	0.872	0.92	0.892
accuracy : 90% *f1-score* : 10%	0.819	0.871	0.92	0.891

Tabel 4.15 menunjukkan hasil pelatihan model GRU dengan auto *epoch*, di mana durasi pelatihan ditentukan secara otomatis untuk mencapai kinerja terbaik. Dengan *epoch* bernilai 14 dengan *learning rate* 0.0001 memberikan performa terbaik dengan akurasi 0.92 dan *f1-score* 0.92 pada kelas normal. Kelas abnormal juga menunjukkan kinerja yang sangat baik dengan *recall* 0.95 dan *f1-score* 0.92.

Learning rate 0.0001 terbukti memberikan performa yang paling optimal dalam hal akurasi dan *f1-score* pada semua percobaan yang dilakukan. Meskipun metode otomatisasi *learning rate* (auto_lr) juga efektif, hasilnya tidak dapat melampaui performa yang dicapai dengan *learning rate* manual 0.0001. Penambahan jumlah *epoch* pada model juga berkontribusi pada peningkatan performa, dengan hasil terbaik tercapai pada 30 *epoch* untuk *learning rate* 0.0001. Secara keseluruhan, model GRU konsisten dalam merespons penurunan *learning rate* dengan peningkatan performa, dan penggunaan *learning rate* yang lebih rendah bersama dengan jumlah *epoch* dapat memberikan hasil yang lebih baik.

4.1.3 Hasil Model *Temporal Convolutional Network* (TCN)

1) *Epoch* 10

Pada percobaan dengan 10 *epoch*, penurunan *learning rate* secara konsisten meningkatkan kinerja model TCN pada setiap pengujian. Model mencapai performa terbaik dengan *learning rate* 1.0000e-04, sesuai dengan penerapan metode otomatisasi *learning rate* menggunakan *library* 'ReduceLROnPlateau'. Hal ini menunjukkan bahwa penyesuaian otomatis dapat mengoptimalkan performa model lebih baik dibandingkan dengan nilai *learning rate* manual lainnya.

Tabel 4.17 Hasil pelatihan model TCN dengan 10 *epoch*

Kelas	*Learning rate*		*Accuracy*	*Precision*	*Recall*	*F1-score*
normal	manual	0.01	0.89	0.88	0.90	0.89
abnormal				0.90	0.87	0.89
normal	manual	0.001	0.90	0.90	0.91	0.90
abnormal				0.91	0.89	0.90
normal	manual	0.0001	0.91	0.88	0.95	0.91
abnormal				0.95	0.87	0.91
normal	auto_lr	1.0000e-04	0.93	0.91	0.95	0.93
abnormal				0.94	0.91	0.93

Tabel 4.18 Perbandingan performa model TCN dengan 10 *epoch*

Bobot presentase	*Learning rate*			
	0.01	0.001	0.0001	1.0000e-04
accuracy : 50% *f1-score* : 50%	0.89	0.90	0.91	0.93
accuracy : 60% *f1-score* : 40%	0.89	0.90	0.91	0.93
accuracy : 70% *f1-score* : 30%	0.89	0.90	0.91	0.93
accuracy : 80% *f1-score* : 20%	0.89	0.90	0.91	0.93
accuracy : 90% *f1-score* : 10%	0.89	0.90	0.91	0.93

Tabel 4.18 menunjukkan bahwa performa model TCN setelah pelatihan dengan 10 *epoch* bervariasi tergantung pada kombinasi bobot presentase antara akurasi dan *f1-score*. Penurunan *learning rate*, terutama pada nilai 1.0000e-04, memberikan hasil yang optimal dalam hal *f1-score*, menunjukkan efektivitas metode otomatisasi *learning rate* dalam meningkatkan kinerja model TCN.

2) *Epoch* 20

Pada percobaan dengan 20 *epoch*, hasil menunjukkan bahwa penurunan nilai *learning rate* tidak secara signifikan meningkatkan kinerja model TCN. Tabel 4.19 menunjukkan bahwa performa model tidak mengalami perubahan yang mencolok dibandingkan dengan percobaan sebelumnya menggunakan 10 *epoch*. Tiga dari empat percobaan menghasilkan hasil yang konsisten, dengan *learning rate* 1.0000e-04 menunjukkan performa yang serupa dengan *learning rate* manual lainnya. Namun, model tetap mencapai performa terbaik pada *learning rate* 1.0000e-04, yang diterapkan melalui metode otomatisasi *learning rate*.

Tabel 4.19 Hasil pelatihan model TCN dengan 20 *epoch*

Kelas	*Learning rate*		*Accuracy*	*Precision*	*Recall*	*F1-score*
normal	manual	0.01	0.91	0.87	0.97	0.92
abnormal				0.96	0.86	0.91
normal	manual	0.001	0.91	0.90	0.94	0.91
abnormal				0.93	0.89	0.91
normal	manual	0.0001	0.91	0.89	0.94	0.91
abnormal				0.94	0.88	0.91
normal	auto_lr	1.0000e-04	0.92	0.90	0.94	0.92
abnormal				0.94	0.90	0.92

Tabel 4.20 Perbandingan performa model TCN dengan 20 *epoch*

Bobot presentase	*Learning rate*			
	0.01	0.001	0.0001	1.0000e-04
accuracy : 50% *f1-score* : 50%	0.915	0.91	0.91	0.92
accuracy : 60% *f1-score* : 40%	0.914	0.91	0.91	0.92
accuracy : 70% *f1-score* : 30%	0.913	0.91	0.91	0.92
accuracy : 80% *f1-score* : 20%	0.912	0.91	0.91	0.92
accuracy : 90% *f1-score* : 10%	0.911	0.91	0.91	0.92

Tabel 4.20 menunjukkan bahwa performa model TCN setelah pelatihan dengan 20 *epoch* menunjukkan konsistensi yang tinggi dalam hal akurasi pada semua kombinasi bobot presentase. *Learning rate* 1.0000e-04 memberikan hasil terbaik dengan akurasi 0.92, yang lebih tinggi dibandingkan dengan *learning rate* lainnya pada semua bobot presentase. Sementara itu, nilai akurasi sedikit menurun

seiring penurunan *learning rate*, menunjukkan bahwa *learning rate* yang lebih kecil membantu model dalam mencapai performa optimal yang konsisten.

3) *Epoch* 30

Percobaan dengan menggunakan 30 *epoch* menunjukkan bahwa penurunan *learning rate* tidak menghasilkan peningkatan signifikan dalam kinerja model TCN, namun sedikit berbeda dengan hasil yang ditemukan pada percobaan dengan 20 *epoch*. Model pada *epoch* 30 mencapai performa terbaiknya pada *learning rate* 0.001, dengan nilai akurasi dan *f1-score* 0.92.

Tabel 4.21 Hasil pelatihan model TCN dengan 30 *epoch*

Kelas	*Learning rate*		*Accuracy*	*Precision*	*Recall*	*F1-score*
normal	manual	0.01	0.91	0.88	0.94	0.91
abnormal				0.94	0.87	0.90
normal	manual	0.001	0.91	0.89	0.94	0.91
abnormal				0.94	0.88	0.91
normal	manual	0.0001	0.91	0.89	0.95	0.92
abnormal				0.95	0.88	0.91
normal	auto_lr	1.0000e-04	0.92	0.90	0.95	0.92
abnormal				0.94	0.90	0.92

Tabel 4.22 Perbandingan performa model TCN dengan 30 *epoch*

Bobot presentase	*Learning rate*			
	0.01	0.001	0.0001	1.0000e-04
accuracy : 50% *f1-score* : 50%	0.91	0.91	0.915	0.92
accuracy : 60% *f1-score* : 40%	0.91	0.91	0.914	0.92
accuracy : 70% *f1-score* : 30%	0.91	0.91	0.913	0.92
accuracy : 80% *f1-score* : 20%	0.91	0.91	0.912	0.92
accuracy : 90% *f1-score* : 10%	0.91	0.91	0.911	0.92

Tabel 4.22 menunjukkan bahwa performa model TCN setelah pelatihan dengan 30 *epoch* mencapai hasil terbaik pada *learning rate* 1.0000e-04, dengan *f1-score* yang konsisten sebesar 0.92 di semua kombinasi bobot presentase. Meskipun *learning rate* 0.001 dan 0.0001 juga menunjukkan performa yang baik dengan akurasi 0.91, *learning rate* 1.0000e-04 memberikan akurasi tertinggi dan

kinerja optimal secara keseluruhan. Performa model tidak menunjukkan peningkatan signifikan dengan penurunan *learning rate* setelah 30 *epoch*, menunjukkan bahwa *learning rate* 1.0000e-04 adalah yang paling efektif untuk model TCN pada durasi pelatihan ini.

4) Auto *Epoch*

Percobaan ini menggunakan auto *epoch* yang menunjukkan bahwa performa model TCN sangat dipengaruhi oleh jumlah *epoch* dan *learning rate* yang digunakan. Hasil pelatihan mengindikasikan bahwa model dapat mencapai performa terbaik pada *learning rate* sebesar 0.0001. Pengaturan *epoch* secara otomatis memungkinkan model untuk menentukan durasi pelatihan yang optimal, dengan hasil yang menunjukkan bahwa *learning rate* ini memberikan hasil terbaik dibandingkan dengan *learning rate* lainnya yang diuji.

Tabel 4.23 Hasil pelatihan model TCN dengan auto *epoch*

Epoch	Kelas	*Learning rate*		*Accuracy*	*Precision*	*Recall*	*F1-score*
10	normal	manual	0.01	0.89	0.91	0.86	0.88
	abnormal				0.87	0.91	0.89
12	normal	manual	0.001	0.90	0.89	0.91	0.90
	abnormal				0.91	0.89	0.90
11	normal	manual	0.0001	0.92	0.90	0.95	0.92
	abnormal				0.94	0.90	0.92
9	normal	auto_lr	1.0000e-04	0.85	0.83	0.90	0.86
	abnormal				0.89	0.81	0.85

Tabel 4.24 Perbandingan performa model TCN dengan auto *epoch*

Bobot presentase	*Learning rate*			
	0.01	0.001	0.0001	1.0000e-04
accuracy : 50% *f1-score* : 50%	0.885	0.90	0.92	0.855
accuracy : 60% *f1-score* : 40%	0.886	0.90	0.92	0.854
accuracy : 70% *f1-score* : 30%	0.887	0.90	0.92	0.853
accuracy : 80% *f1-score* : 20%	0.888	0.90	0.92	0.852
accuracy : 90% *f1-score* : 10%	0.889	0.90	0.92	0.851

Tabel 4.24 menunjukkan bahwa performa model TCN setelah pelatihan dengan bantuan *callback* 'EarlyStopping' bervariasi tergantung pada kombinasi *learning rate*. Dari tabel, dapat dilihat bahwa *learning rate* 1.0000e-04 memberikan *f1-score* tertinggi sebesar 0.92 pada berbagai bobot presentase, menunjukkan bahwa model mencapai hasil terbaik pada variansi ini. Meskipun *learning rate* 0.0001 memberikan akurasi yang konsisten tinggi dan performa yang baik, *f1-score* untuk *learning rate* ini sedikit lebih rendah dibandingkan dengan 0.0001. *Learning rate* 0.01 dan 0.001 menunjukkan performa yang lebih rendah dalam hal *f1-score* dibandingkan dengan *learning rate* 1.0000e-04. Dengan demikian, model TCN menunjukkan bahwa *learning rate* 1.0000e-04 adalah yang paling efektif dalam memaksimalkan *f1-score* selama pelatihan dengan *epoch* yang disesuaikan.

4.2 Evaluasi Performa Model Dasar/*Base Learner*

Berdasarkan hasil proses pelatihan model dasar/*base learner* yang telah dilakukan sebelumnya, serta mengacu pada Tabel 4.2, 4.4, 4.6, 4.8, 4.10, 4.12, 4.14, 4.16, 4.18, 4.20, 4.22, dan 4.24 yang telah dipaparkan, dapat disimpulkan bahwa bobot presentase dari dua metrik yang digunakan dalam penelitian ini, yaitu akurasi dan *f1-score*, dari perbandingan bobot 50:50 hingga 90:10 tidak menunjukkan perbedaan hasil yang signifikan. Oleh karena itu, peneliti dapat lebih mudah menentukan model terbaik untuk setiap percobaan berdasarkan nilai *epoch* yang digunakan. Sehingga, pada Tabel 4.25, 4.26, dan 4.27, dapat diamati model dengan performa terbaik dari setiap percobaan berdasarkan variansi nilai *epoch*.

Tabel 4.25 Model LSTM dengan performa terbaik berdasarkan *epoch*

Epoch		Kelas	*Learning rate*	*Accuracy*	*Precision*	*Recall*	*F1-score*
manual	10	normal	0.0001	0.90	0.87	0.95	0.91
		abnormal			0.94	0.86	0.90
manual	20	normal	0.0001	0.92	0.90	0.95	0.92
		abnormal			0.94	0.89	0.92
manual	30	normal	0.0001	0.92	0.90	0.95	0.92
		abnormal			0.94	0.90	0.92
early stopping	15	normal	0.0001	0.91	0.90	0.94	0.91
		abnormal			0.93	0.89	0.91

Tabel 4.26 Model GRU dengan performa terbaik berdasarkan *epoch*

Epoch		Kelas	*Learning rate*	*Accuracy*	*Precision*	*Recall*	*F1-score*
manual	10	normal	0.0001	0.88	0.81	0.96	0.88
		abnormal			0.95	0.78	0.86
manual	20	normal	0.0001	0.94	0.93	0.94	0.94
		abnormal			0.94	0.93	0.93
manual	30	normal	0.0001	0.93	0.91	0.94	0.93
		abnormal			0.94	0.91	0.92
early stopping	14	normal	0.0001	0.93	0.91	0.95	0.93
		abnormal			0.95	0.90	0.93

Tabel 4.27 Model TCN dengan performa terbaik berdasarkan *epoch*

Epoch		Kelas	*Learning rate*	*Accuracy*	*Precision*	*Recall*	*F1-score*
manual	10	normal	1.0000e -4	0.93	0.91	0.95	0.93
		abnormal			0.94	0.91	0.93
manual	20	normal	1.0000e -4	0.92	0.90	0.94	0.92
		abnormal			0.94	0.90	0.92
manual	30	normal	1.0000e -4	0.92	0.90	0.95	0.92
		abnormal			0.94	0.90	0.92
early stopping	11	normal	0.0001	0.92	0.90	0.95	0.92
		abnormal			0.94	0.90	0.92

Berdasarkan analisis data dari Tabel 4.25, 4.26, dan 4.27, dapat disimpulkan bahwa pemilihan *epoch* dan *learning rate* yang tepat berpengaruh signifikan terhadap performa model. Untuk model LSTM, *epoch* 20 dengan *learning rate* 0.0001 menunjukkan performa terbaik dengan akurasi, precision, recall, dan *f1-score* yang tinggi. Model GRU mencapai hasil terbaik pada *epoch* 20 dengan *learning rate* 0.0001. Model TCN mencapai performa terbaik pada *epoch* 10 dengan *learning rate* 0.0001, dan penggunaan *early stopping* pada *epoch* 11 dengan *learning rate* 0.0001 menghasilkan performa yang setara dengan *epoch* 20.

Secara keseluruhan, penggunaan *epoch* yang lebih tinggi seringkali memberikan hasil yang baik, namun penting untuk mempertimbangkan kombinasi *learning rate* yang sesuai untuk mencapai hasil optimal pada masing-masing model. Dari Tabel 4.25, 4.26, dan 4.27, juga dapat diamati bahwa penggunaan fungsi *callback* 'EarlyStopping' cukup memberikan performa yang baik. Selain mempersingkat waktu pelatihan model, penggunaan *callback* ini juga dapat memberikan performa yang cukup maksimal, meskipun performa terbaik tetap

dicapai pada *epoch* yang ditentukan secara manual. Sehingga, kombinasi yang tepat antara *epoch* dan *learning rate* memastikan bahwa model dapat memaksimalkan akurasi, *precision*, *recall*, dan *f1-score* dalam berbagai kondisi.

4.3 Hasil Proses *Training* Model *Ensemble*

Setelah menentukan model terbaik dari setiap percobaan dengan variasi nilai *epoch*, langkah selanjutnya adalah mengintegrasikan prediksi dari model dasar tersebut menggunakan teknik *ensemble learning* dengan pendekatan *soft voting*. Proses pelatihan pada model *ensemble* ini akan dilakukan beberapa kali percobaan, dengan pengelompokan berdasarkan nilai *epoch* dan variasi bobot yang diberikan kepada setiap model dasar/*base learner*. Percobaan variansi bobot akan dilakukan dua (2) kali dengan membedakan bobot yang diberikan kepada masing-masing model dasar. Pada percobaan pertama, setiap model akan diberikan bobot yang sama. Sedangkan pada percobaan kedua, salah satu model akan diberikan bobot yang lebih berat, yaitu bernilai 2, dibandingkan dengan dua model dasar lainnya. Rincian simulasi pemberian bobot untuk setiap percobaan yang dilakukan, dapat dilihat lebih lanjut pada Tabel 4.28.

Tabel 4.28 Simulasi pemberian bobot pada model dasar

Percobaan ke	Bobot		
	LSTM	GRU	TCN
1	1	1	1
2	2	1	1
3	1	2	1
4	1	1	2

Tabel 4.28 menunjukkan contoh pemberian bobot yang lebih berat kepada satu model serta bobot yang lebih rendah dan sama pada dua model lainnya. Pada percobaan ke-1, setiap model dasar diberikan bobot yang sama yaitu satu (1). Sedangkan pada percobaan ke-2, 3, dan 4, bobot yang lebih berat diberikan kepada model dasar yang memiliki nilai performa yang lebih baik dibandingkan dengan dua model lainnya. Dengan demikian, dalam penelitian ini, urutan pemberian bobot pada model *ensemble* secara berurutan adalah LSTM:GRU:TCN.

Penentuan performa terbaik didasarkan pada perbandingan nilai metrik akurasi dan *f1-score* yang telah dipaparkan sebelumnya, serta telah tercatat pada

Tabel 4.25, 4.26, dan 4.27 untuk setiap model terbaik berdasarkan nilai *epoch*. Model *ensemble* akan diuji menggunakan data *testing* atau data yang berbeda dari data pelatihan model dasar. Hal ini dilakukan untuk menilai performa model *ensemble* terhadap data yang belum dikenali. Evaluasi performa model *ensemble* akan menggunakan dua metrik yang sebelumnya digunakan pada model dasar, yaitu akurasi dan *f1-score*, serta menambahkan satu metrik evaluasi yaitu ROC AUC, dengan berbagai persentase bobot antara lain 40:30:30, 50:25:25, 60:20:20, 70:15:15, dan 80:10:10.

Penambahan parameter ROC AUC diharapkan mampu menilai performa model *ensemble* secara lebih menyeluruh, karena fungsi dari ROC AUC adalah menghitung area di bawah kurva ROC. Kurva ROC adalah plot yang menggambarkan *trade-off* antara *true positive rate* dan *false positive rate* pada berbagai *threshold*. ROC AUC memberikan gambaran keseluruhan mengenai kemampuan model dalam membedakan antara kelas positif dan negatif. Hal ini akan membantu mengidentifikasi performa model dalam berbagai kondisi dan memberikan pandangan yang lebih lengkap tentang performanya.

4.3.1 Hasil Model *Ensemble* Pada *Epoch* 10

Dapat dilihat dari Tabel 4.29, bahwa nilai performa model TCN secara signifikan lebih baik dibandingkan dengan model LSTM dan GRU. Oleh karena itu, bobot/*weight* dua (2) pada percobaan kedua akan diberikan pada model TCN pada percobaan dengan nilai *epoch* 10.

Tabel 4.29 Model dasar terbaik dengan 10 *epoch*

Model	Kelas	*Learning rate*	*Accuracy*	*Precision*	*Recall*	*F1-score*
LSTM	normal	0.0001	0.90	0.87	0.95	0.91
	abnormal			0.94	0.86	0.90
GRU	normal	0.0001	0.88	0.81	0.96	0.88
	abnormal			0.95	0.78	0.86
TCN	normal	1.0000e-04	0.93	0.91	0.95	0.93
	abnormal			0.94	0.91	0.93

Tabel 4.30 menunjukkan hasil pelatihan model *ensemble* dengan menggunakan model dasar yang dilatih dengan nilai *epoch* 10. Pada konfigurasi bobot 1:1:1, di mana setiap model (LSTM, GRU, TCN) memiliki bobot yang sama,

ensemble mencapai akurasi sebesar 92%. Sementara itu, pada konfigurasi bobot 1:1:2, yang memberikan bobot 2 pada model TCN dan bobot 1 pada model LSTM dan GRU, ensemble juga mencapai akurasi sebesar 92%. Dengan demikian, konfigurasi bobot 1:1:2 tidak secara signifikan meningkatkan akurasi dibandingkan konfigurasi bobot 1:1:1.

Tabel 4.30 Hasil pelatihan model *ensemble* dengan 10 *epoch*

Weight LSTM:GRU:TCN	Kelas	*Accuracy*	*Precision*	*Recall*	*F1-score*	*ROC AUC*
1:1:1	normal	0.92	0.90	0.95	0.91	0.9604
	abnormal		0.94	0.90	0.92	
1:1:2	normal	0.92	0.89	0.95	0.92	0.9636
	abnormal		0.95	0.88	0.92	

Tabel 4.31 Perbandingan performa model *ensemble* dengan 10 *epoch*

Weight LSTM:GRU:TCN	Persentase bobot *accuracy* : *f1-score* : *roc auc*				
	40:30:30	50:25:25	60:20:20	70:15:15	80:10:10
1:1:1	0.9291	0.9276	0.9261	0.9246	0.9230
1:1:2	0.9331	0.9309	0.9287	0.9265	0.9244

Tabel 4.31 menunjukkan bahwa bobot evaluasi akurasi, *f1-score*, dan ROC AUC bervariasi dari 40:30:30 hingga 80:10:10. Hasilnya menunjukkan bahwa konfigurasi bobot 1:1:1 dan 1:1:2 menghasilkan nilai performa yang sedikit berbeda pada setiap persentase bobot evaluasi, dengan performa terbaik tetap pada model dengan bobot 1:1:2. Meskipun ada variasi dalam hasil, perbedaan performa antara kedua konfigurasi bobot tidak signifikan.

4.3.2 Hasil Model *Ensemble* Pada *Epoch* 20

Seperti yang terlihat pada Tabel 4.32, dalam percobaan dengan *epoch* 20, model GRU menunjukkan performa yang lebih unggul dibandingkan dengan model LSTM dan TCN. Oleh karena itu, pada percobaan *ensemble* kedua, bobot 2 akan diberikan kepada model GRU yang menggunakan nilai *epoch* 20.

Tabel 4.32 Model dasar terbaik dengan 20 *epoch*

Model	Kelas	*Learning rate*	*Accuracy*	*Precision*	*Recall*	*F1-score*
LSTM	normal	0.0001	0.92	0.9	0.95	0.92
	abnormal			0.94	0.89	0.92
GRU	normal	0.0001	0.94	0.93	0.94	0.94
	abnormal			0.94	0.93	0.93
TCN	normal	1.0000e-04	0.92	0.9	0.94	0.92
	abnormal			0.94	0.9	0.92

Tabel 4.33 menunjukkan hasil pelatihan model *ensemble* dengan menggunakan model dasar yang dilatih dengan nilai *epoch* 20. Pada konfigurasi bobot 1:1:1, di mana setiap model (LSTM, GRU, TCN) memiliki bobot yang sama, *ensemble* mencapai akurasi sebesar 92%. Sementara itu, pada konfigurasi bobot 1:2:1, yang memberikan bobot 2 pada model GRU dan bobot 1 pada model LSTM dan TCN, *ensemble* juga mencapai akurasi sebesar 92%. Dengan demikian, konfigurasi bobot 1:2:1 tidak secara signifikan meningkatkan akurasi dibandingkan konfigurasi bobot 1:1:1.

Tabel 4.33 Hasil pelatihan model *ensemble* dengan 20 *epoch*

Weight LSTM:GRU:TCN	Kelas	*Accuracy*	*Precision*	*Recall*	*F1-score*	*ROC AUC*
1:1:1	normal	0.92	0.90	0.95	0.92	0.92
	abnormal		0.94	0.90	0.92	
1:2:1	normal	0.92	0.90	0.95	0.92	0.92
	abnormal		0.94	0.89	0.91	

Tabel 4.34 Perbandingan performa model *ensemble* dengan 20 *epoch*

Weight LSTM:GRU:TCN	Persentase bobot *accuracy* : *f1-score* : *roc auc*				
	40:30:30	50:25:25	60:20:20	70:15:15	80:10:10
1:1:1	0.9327	0.9306	0.9285	0.9263	0.9242
1:2:1	0.9333	0.9311	0.9288	0.9266	0.9244

Tabel 4.34 menunjukkan bahwa bobot evaluasi akurasi, *f1-score*, dan ROC AUC bervariasi dari 40:30:30 hingga 80:10:10. Hasilnya menunjukkan bahwa konfigurasi bobot 1:1:1 dan 1:2:1 menghasilkan perbedaan performa yang relatif kecil, dengan performa terbaik tercapai pada konfigurasi bobot 1:2:1.

4.3.3 Hasil Model *Ensemble* Pada *Epoch* 30

Tabel 4.35 memaparkan perbandingan performa ketiga model dengan nilai *epoch* 30, model GRU kembali menunjukkan performa yang lebih unggul dibandingkan dengan model LSTM dan TCN. Oleh karena itu, pada percobaan *ensemble* kedua, bobot 2 juga diberikan kepada model GRU yang menggunakan nilai *epoch* 30.

Tabel 4.35 Model dasar terbaik dengan 30 *epoch*

Model	Kelas	*Learning rate*	*Accuracy*	*Precision*	*Recall*	*F1-score*
LSTM	normal	0.0001	0.92	0.9	0.95	0.92
	abnormal			0.94	0.9	0.92
GRU	normal	0.0001	0.93	0.91	0.94	0.93
	abnormal			0.94	0.91	0.92
TCN	normal	1.0000e-04	0.92	0.9	0.95	0.92
	abnormal			0.94	0.90	0.92

Tabel 4.36 menunjukkan hasil pelatihan model *ensemble* dengan menggunakan model dasar yang dilatih pada nilai *epoch* 30. Pada konfigurasi bobot 1:1:1, *ensemble* mencapai akurasi 92%. Sedangkan pada konfigurasi bobot 1:2:1, akurasi tetap sama yaitu 92%. Kembali lagi seperti percobaan sebelumnya, nilai akurasi yang dihasilkan tetap bernilai sama, tidak ada peningkatan. Dengan demikian, konfigurasi bobot 1:2:1 tidak secara signifikan meningkatkan akurasi dibandingkan konfigurasi bobot 1:1:1, tetapi tetap model dengan bobot 1:2:1 menjadi lebih unggul karena nilai ROC AUC yang dimilikinya.

Tabel 4.36 Hasil pelatihan model *ensemble* dengan 30 *epoch*

Weight LSTM:GRU:TCN	Kelas	*Accuracy*	*Precision*	*Recall*	*F1-score*	*ROC AUC*
1:1:1	normal	0.92	0.90	0.95	0.92	0.9658
	abnormal		0.94	0.90	0.92	
1:2:1	normal	0.92	0.90	0.95	0.92	0.9687
	abnormal		0.94	0.90	0.92	

Tabel 4.37 Perbandingan performa model *ensemble* dengan 30 *epoch*

Weight LSTM:GRU:TCN	Persentase bobot *accuracy* : *f1-score* : *roc auc*				
	40:30:30	50:25:25	60:20:20	70:15:15	80:10:10
1:1:1	0.9337	0.9315	0.9292	0.9269	0.9246
1:2:1	0.9346	0.9322	0.9297	0.9273	0.9249

Tabel 4.37 menunjukkan bahwa bobot evaluasi akurasi, *f1-score*, dan ROC AUC bervariasi dari 40:30:30 hingga 80:10:10. Hasilnya menunjukkan bahwa konfigurasi bobot 1:1:1 dan 1:2:1 menghasilkan perbedaan performa yang relatif kecil, dengan performa terbaik tercapai pada konfigurasi bobot 1:2:1.

4.3.4 Hasil Model *Ensemble* Dengan *Callback EarlyStopping*

Tabel 4.38 memaparkan perbandingan performa ketiga model dengan nilai *epoch* yang telah ditentukan oleh *callback* 'EarlyStopping', kembali seperti pada *epoch* 20 dan 30 sebelumnya, model GRU kembali menunjukkan performa yang lebih unggul dibandingkan dengan model LSTM dan TCN. Oleh karena itu, pada percobaan *ensemble* kedua, bobot 2 juga diberikan kepada model GRU.

Tabel 4.38 Model dasar terbaik dengan auto *epoch*

Model	Kelas	*Epoch*	*Learning rate*	*Accuracy*	*Precision*	*Recall*	*F1-score*
LSTM	normal	15	0.0001	0.91	0.90	0.94	0.91
	abnormal				0.93	0.89	0.91
GRU	normal	14	0.0001	0.93	0.91	0.95	0.93
	abnormal				0.95	0.90	0.93
TCN	normal	11	1.0000e-04	0.92	0.90	0.95	0.92
	abnormal				0.94	0.90	0.92

Tabel 4.39 menunjukkan hasil pelatihan model *ensemble* dengan menggunakan model dasar yang dilatih dengan menggunakan fungsi *callback* 'EarlyStopping'. Pada konfigurasi bobot 1:1:1, *ensemble* mencapai akurasi 92%. Sedangkan pada konfigurasi bobot 1:2:1, akurasi tetap sama yaitu 92%. Kembali lagi seperti percobaan sebelumnya, nilai akurasi yang dihasilkan tetap bernilai sama, tidak ada peningkatan. Dengan demikian, model dengan konfigurasi bobot 1:2:1 menjadi lebih unggul karena nilai ROC AUC yang dimilikinya.

Tabel 4.39 Hasil pelatihan model *ensemble* dengan auto *epoch*

Weight LSTM:GRU:TCN	Kelas	*Accuracy*	*Precision*	*Recall*	*F1-score*	*ROC AUC*
1:1:1	normal	0.92	0.90	0.95	0.90	0.9632
	abnormal		0.94	0.89	0.92	
1:2:1	normal	0.92	0.90	0.95	0.92	0.9638
	abnormal		0.94	0.89	0.92	

Tabel 4.40 Perbandingan performa model *ensemble* dengan auto *epoch*

Weight LSTM:GRU:TCN	Persentase bobot *accuracy* : *f1-score* : *roc auc*				
	40:30:30	50:25:25	60:20:20	70:15:15	80:10:10
1:1:1	0.9270	0.9258	0.9246	0.9235	0.9223
1:2:1	0.9331	0.9310	0.9288	0.9266	0.9244

Tabel 4.40 menunjukkan bahwa bobot evaluasi akurasi, *f1-score*, dan ROC AUC bervariasi dari 40:30:30 hingga 80:10:10. Hasilnya menunjukkan bahwa konfigurasi bobot 1:1:1 dan 1:2:1 menghasilkan perbedaan performa yang relatif kecil, dengan performa terbaik tercapai pada konfigurasi bobot 1:2:1. Sehingga secara keseluruhan percobaan yang telah dilakukan, model dengan konfigurasi bobot 1:1:2 untuk *epoch* 10 dan 1:2:1 untuk *epoch* 20, 30, serta 'auto' dengan menggunakan bantuan fungsi *callback* 'EarlyStopping' memiliki performa yang lebih unggul dibandingkan dengan konfigurasi bobot setara atau 1:1:1.

4.4 Evaluasi Performa Model *Ensemble*

Setelah melakukan serangkaian percobaan dan melakukan variasi bobot pada model-model dasar berdasarkan performa terbaik dari variansi *epoch*, di mana model TCN pada *epoch* 10, dan model GRU pada *epoch* 20, 30, dan 'auto' dengan menggunakan *callback* 'EarlyStopping', tidak terlihat peningkatan yang signifikan dalam performa model *ensemble*. Metrik akurasi tetap konsisten pada nilai yang sama antara kedua konfigurasi bobot. Hal ini menunjukkan bahwa kontribusi setiap model dalam *ensemble* adalah seimbang, tanpa memberikan keuntungan tambahan dari pemberian bobot yang lebih tinggi pada model dengan performa terbaik.

Dengan kata lain, variansi nilai bobot pada model *ensemble* mampu mempertahankan performa yang stabil tanpa adanya peningkatan signifikan dalam metrik akurasi, *f1-score*, ataupun ROC AUC. Namun, jika dilihat pada Tabel 4.34, 4.37, dan 4.40, dapat disimpulkan bahwa konfigurasi bobot dengan memberikan model terbaik bobot yang lebih tinggi, performanya lebih baik dibandingkan dengan model *ensemble* yang diberikan bobot sama. Hal ini terlihat dari nilai metrik ROC AUC, di mana setiap model memiliki perbedaan nilai meskipun sangat kecil. Perbedaan nilai tersebut baru dapat dilihat pada angka keempat di belakang koma untuk setiap nilai ROC AUC yang dimiliki model *ensemble*. Kesimpulan ini

memberikan wawasan penting tentang bagaimana penyesuaian bobot dapat mempengaruhi performa model *ensemble*, walaupun tidak begitu besar efeknya.

Tabel 4.41 Model *ensemble* dengan performa terbaik

Weight	*Epoch*	Kelas	*Accuracy*	*Precision*	*Recall*	*F1-score*	*ROC AUC*
1:2:1	10	normal	0.92	0.89	0.95	0.92	0.9636
		abnormal		0.95	0.88	0.92	
1:2:1	20	normal	0.92	0.90	0.95	0.92	0.9642
		abnormal		0.94	0.89	0.91	
1:2:1	30	normal	0.92	0.90	0.95	0.92	0.9687
		abnormal		0.94	0.90	0.92	
1:2:1	*Early stopping*	normal	0.92	0.90	0.95	0.92	0.9638
		abnormal		0.94	0.89	0.92	

Tabel 4.41 menunjukkan model *ensemble* terbaik dari keseluruhan percobaan yang telah dilakukan dengan variansi nilai bobot/*weight*. Performa model *ensemble* tetap konsisten, mencapai akurasi 92%. pada *epoch* 10, model *ensemble* mencapai ROC AUC sebesar 0.9636. Pada *epoch* 20, nilai ROC AUC sedikit meningkat menjadi 0.9642. Peningkatan lebih lanjut terlihat pada *epoch* 30, di mana ROC AUC mencapai 0.9687. Metode penggunaan fungsi *callback* 'EarlyStopping' juga menunjukkan performa yang cukup konsisten dengan nilai ROC AUC 0.9638. Meskipun nilai ROC AUC yang dihasilkan oleh 'EarlyStopping' cukup tinggi, model *ensemble* terbaik diperoleh dari *epoch* 30, yang memberikan performa terbaik secara keseluruhan. Sehingga yang akan dijadikan sebagai acuan untuk membandingkan performa model *ensemble* dengan model dasar pada subbab berikutnya adalah model *ensemble* dengan *epoch* 30.

4.5 Perbandingan Performa Model Dasar Dengan Model *Ensemble*

Setelah melakukan proses pelatihan dari model dasar/*base learner* hingga model *ensemble*, langkah berikutnya adalah membandingkan hasil dari berbagai model dengan performa terbaik yang dikelompokkan berdasarkan nilai *epoch*. Hasil dari setiap model dalam setiap kelompok *epoch* yang terbaik akan dibandingkan kembali untuk menentukan model dengan performa terbaik dari semua percobaan yang telah dilakukan dalam penelitian ini.

Evaluasi model difokuskan pada metrik akurasi dengan bobot 60% dan *f1-score* dengan bobot 40%. Mengingat percobaan yang telah dilakukan dengan memberikan variansi bobot pada metrik evaluasi selama proses pelatihan di model dasar dan model *ensemble* tidak menghasilkan hal yang cukup signifikan, maka pada perbandingan performa antar model ini akan dilakukan dengan hanya satu nilai persentase bobot tetap, yaitu 60:40. Hal ini dilakukan untuk memastikan bahwa perbandingan antar model dilakukan secara konsisten dan adil, sehingga dapat memperoleh pemahaman yang jelas mengenai model mana yang memberikan performa terbaik berdasarkan akurasi dan *f1-score*.

4.5.1 Perbandingan Performa Model Dengan 10 *Epoch*

Dari Tabel 4.42, dapat diamati bahwa model TCN memiliki nilai performa tertinggi dibandingkan dengan nilai performa model lainnya untuk nilai *epoch* 10, di mana nilai performa mencapai 93%.

Tabel 4.42 Perbandingan performa model dengan 10 *epoch*

Metrik	Model			
	LSTM	GRU	TCN	*Ensemble*
accuracy	0.90	0.88	0.93	0.92
f1-score	0.91	0.88	0.93	0.92
performance	0.90	0.88	0.93	0.92

4.5.2 Perbandingan Performa Model Dengan 20 *Epoch*

Dari Tabel 4.43, dapat diamati bahwa model GRU memiliki nilai performa tertinggi dibandingkan dengan nilai performa model lainnya untuk nilai *epoch* 20, di mana nilai performa mencapai 94%.

Tabel 4.43 Perbandingan performa model dengan 20 *epoch*

Metrik	Model			
	LSTM	GRU	TCN	*Ensemble*
accuracy	0.92	0.94	0.92	0.92
f1-score	0.92	0.94	0.92	0.92
performance	0.92	0.94	0.92	0.92

4.5.3 Perbandingan Performa Model Dengan 30 *Epoch*

Dari Tabel 4.44, dapat diamati bahwa model GRU memiliki nilai performa tertinggi dibandingkan dengan nilai performa model lainnya untuk nilai *epoch* 30, di mana nilai performa mencapai 93%.

Tabel 4.44 Perbandingan performa model dengan 30 *epoch*

Metrik	Model			
	LSTM	GRU	TCN	*Ensemble*
accuracy	0.92	0.93	0.92	0.92
f1-score	0.92	0.93	0.92	0.92
performance	0.92	0.93	0.92	0.92

4.5.4 Perbandingan Performa Model Dengan *Callback EarlyStopping*

Dari Tabel 4.45, dapat diamati bahwa model GRU memiliki nilai performa tertinggi dibandingkan dengan nilai performa model lainnya untuk nilai *epoch* 14, di mana nilai performa mencapai 93%.

Tabel 4.45 Perbandingan performa model dengan auto *epoch*

Metrik	Model			
	LSTM	GRU	TCN	*Ensemble*
accuracy	0.91	0.93	0.92	0.92
f1-score	0.91	0.93	0.92	0.92
performance	0.91	0.93	0.92	0.92

4.5.5 Perbandingan Performa Model Secara Keseluruhan

Setelah mengidentifikasi model dengan performa terbaik berdasarkan nilai *epoch* yang ditampilkan pada Tabel 4.42, 4.43, 4.44, dan 4.45, selanjutnya adalah membandingkan model terbaik dari setiap *epoch*. Tujuannya untuk menentukan model dengan performa terbaik dari semua percobaan yang telah dilakukan.

Tabel 4.46 Perbandingan performa model secara keseluruhan

Epoch	Model	*Accuracy*	*F1-score*	*Performance*
10	TCN	0.93	0.93	0.93
20	GRU	0.94	0.94	0.94
30	GRU	0.93	0.93	0.93
Early stopping	GRU	0.93	0.93	0.93

Setelah melakukan evaluasi performa dari model dasar dan model *ensemble* pada berbagai nilai *epoch*, serta menganalisis hasil yang ditampilkan pada Tabel

4.46, dua model terbaik telah dipilih untuk diimplementasikan pada antarmuka *website*. Kedua model tersebut adalah model dasar GRU dengan *epoch* 20, serta model *ensemble* dengan bobot 1:2:1 dan *epoch* 30, di mana bobot dua (2) diberikan kepada model dasar GRU. Penggunaan dua model ini bertujuan untuk menyediakan bahan pertimbangan yang lebih komprehensif dalam pengambilan keputusan.

4.6 Uji Performa Model Secara Keseluruhan

Pengujian performa model merupakan tahap kritis dalam mengevaluasi kehandalan dan akurasi prediktif dari model yang telah diimplementasikan. Uji performa dilakukan dalam sepuluh kali percobaan di mana model-model yang telah diintegrasikan akan diuji dengan data *testing* yang dipilih secara acak dari dataset yang digunakan selama proses pelatihan. Hasil prediksi yang dihasilkan, selanjutnya dibandingkan langsung dengan label atau kelas aktual dari data tersebut sesuai dengan dataset asli.

Tabel 4.47 Uji performa model dalam mengklasifikasikan data *testing*

Percobaan ke	Nama File Audio	Aktual Kelas	Hasil Prediksi Model	
			GRU	*Ensemble*
1	d0023	abnormal	normal	abnormal
2	c0026	normal	normal	abnormal
3	c0009	normal	normal	normal
4	d0003	normal	normal	abnormal
5	a0026	normal	normal	normal
6	b0035	abnormal	normal	abnormal
7	e00130	abnormal	normal	normal
8	f0112	abnormal	normal	abnormal
9	a0304	abnormal	normal	normal
10	b0098	abnormal	normal	abnormal

Berdasarkan hasil uji performa model pada Tabel 4.47, dapat disimpulkan bahwa model *ensemble*, yang menggabungkan prediksi dari model LSTM, GRU, dan TCN dengan bobot tertentu, menunjukkan keunggulan dalam memprediksi kelas ‘normal’ pada data audio. Meskipun demikian, terdapat beberapa percobaan di mana model *ensemble* dan model GRU mengalami kesulitan dalam mengidentifikasi kelas ‘abnormal’ dengan tepat. Hal ini menunjukkan bahwa meskipun *ensemble* memberikan prediksi yang lebih stabil secara keseluruhan,

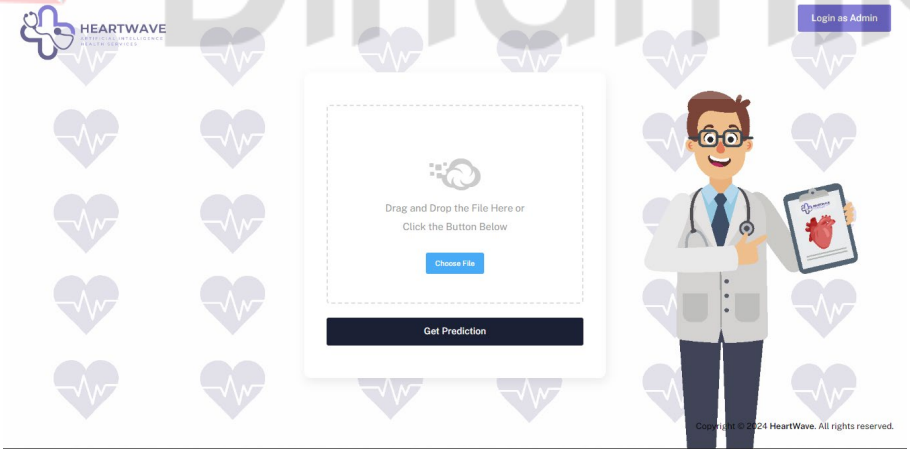
masih diperlukan penyesuaian lebih lanjut pada strategi pembobotan atau pengembangan model individual untuk meningkatkan ketepatan dalam mengklasifikasikan kelas minoritas seperti ‘abnormal’.

4.7 Hasil Implementasi Antarmuka *Website*

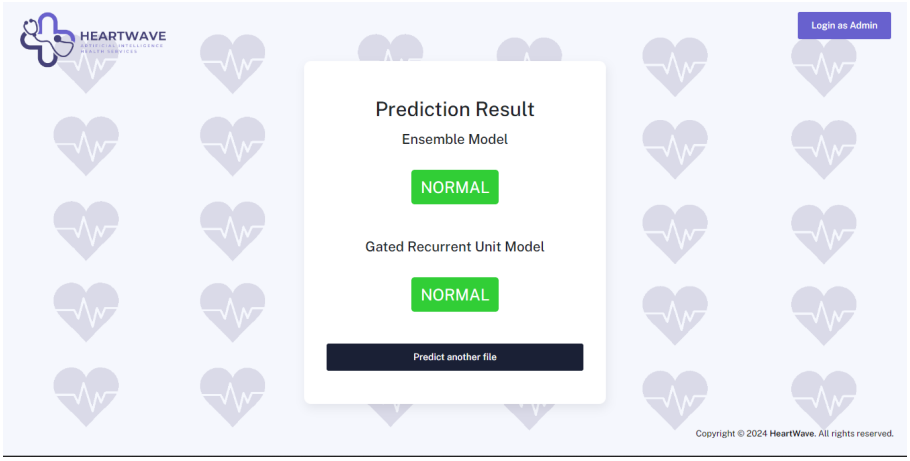
Pada bagian ini, dijelaskan hasil implementasi dua model yang telah dipilih untuk diintegrasikan ke dalam antarmuka *website*. Hasil ini mencakup pemaparan tentang *wireframe* yang telah dirancang sebelumnya serta logika pemrograman yang diterapkan.

4.7.1 Implementasi Desain Antarmuka

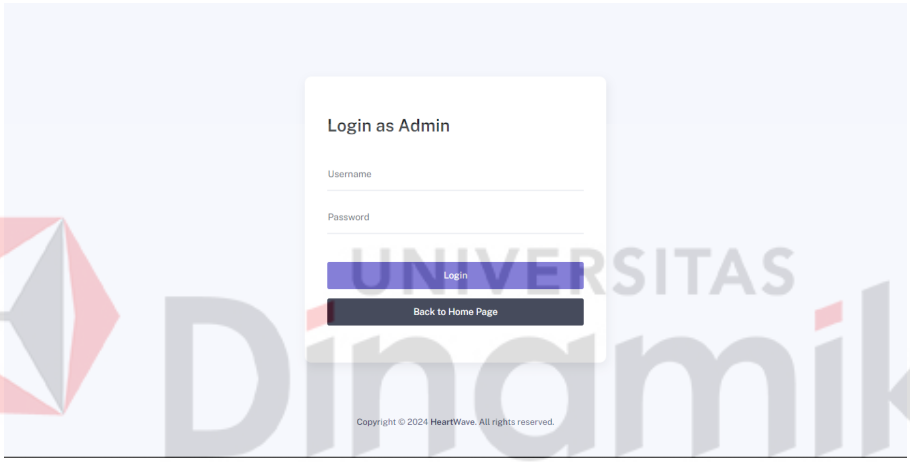
Wireframe yang sebelumnya dirancang telah digunakan sebagai acuan utama dalam pengembangan antarmuka. Setiap elemen dalam *wireframe* ditransformasikan menjadi komponen fungsional pada *website*, mencakup tata letak halaman, elemen interaktif, dan jalur pengguna. Di bawah ini ditampilkan dari desain implementasi antarmuka *website*, yang mencerminkan integrasi elemen-elemen tersebut:



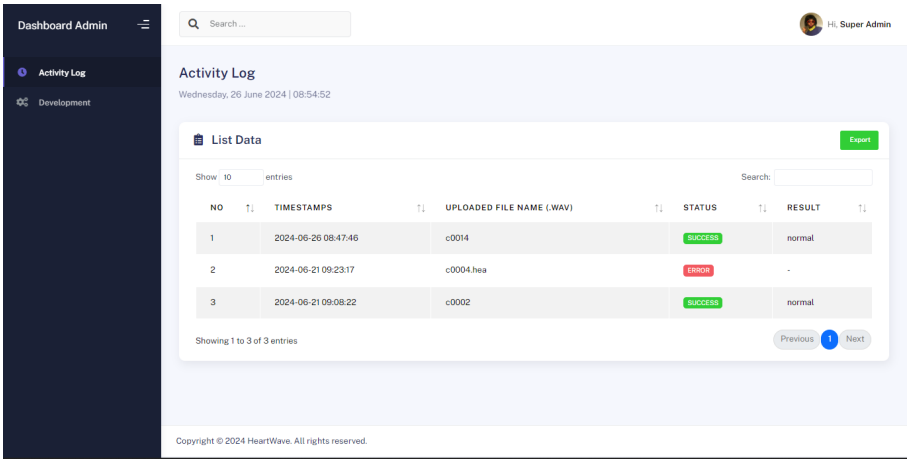
Gambar 4.1 Tampilan halaman utama



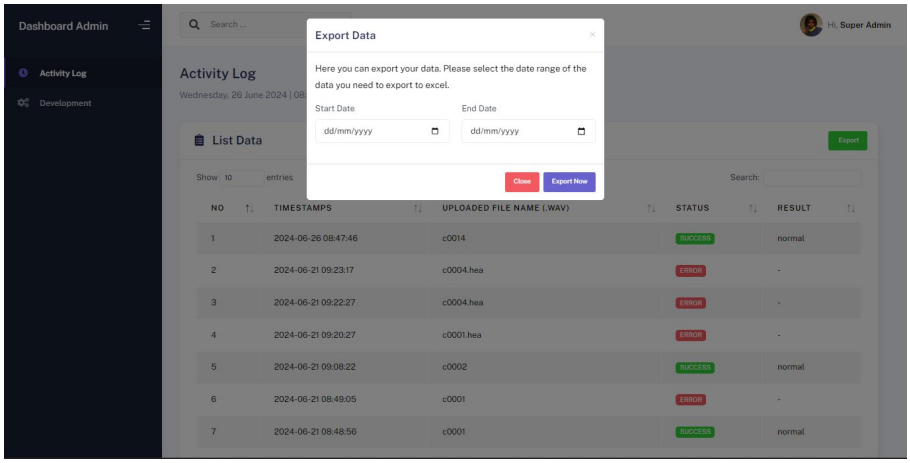
Gambar 4.2 Tampilan halaman hasil prediksi



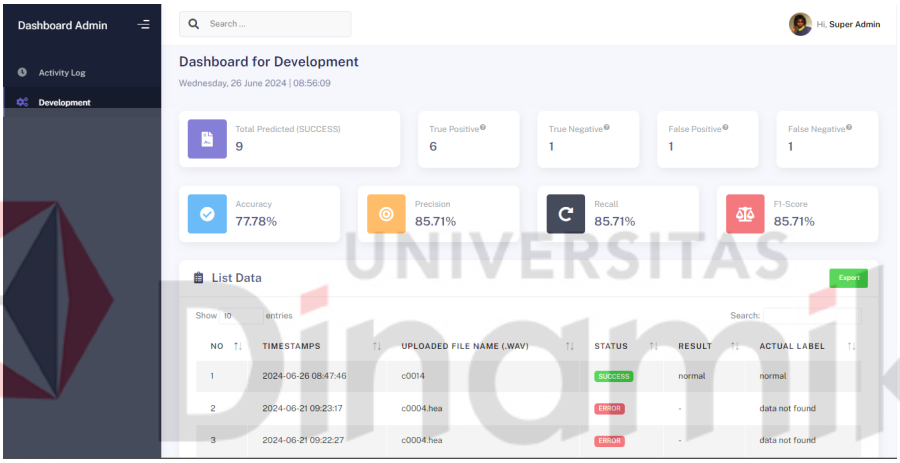
Gambar 4.3 Tampilan halaman *login*



Gambar 4.4 Tampilan halaman *dashboard*



Gambar 4.5 Tampilan aksi *export* data



Gambar 4.6 Tampilan halaman pengembangan

4.7.2 Uji Fungsionalitas *Website*

Untuk memastikan bahwa *website* berfungsi sesuai dengan yang diharapkan, dilakukan serangkaian uji coba yang berfokus pada proses *upload* data audio. Sebanyak 10 kali percobaan dilakukan untuk mengevaluasi kehandalan dan responsivitas sistem dalam menerima, memproses, dan menanggapi data yang diunggah oleh pengguna. Di mana dapat dilihat secara rinci pada Tabel 4.48 berikut.

Tabel 4.48 Uji fungsionalitas fitur *upload* pada *website*

Percobaan ke	Nama File yang Diunggah	Status	Keterangan
1	c0001.wav	Berhasil	Benar
2	c0001.hea	Gagal	Benar
3	c0001.wav	Berhasil	Benar
4	a0021.wav	Berhasil	Benar
5	d0032.wav	Berhasil	Benar
6	models.py	Gagal	Benar
7	tes_file.pdf	Gagal	Benar
8	e00001.wav	Berhasil	Benar
9	erd_sistem.png	Gagal	Benar
10	f0008.wav	Berhasil	Benar

Tabel 4.48 menunjukkan bahwa fungsionalitas antarmuka *website* yang difokuskan pada proses upload data audio telah berhasil dalam semua percobaan tanpa adanya kesalahan.

BAB V
PENUTUP

5.1. Kesimpulan

Berdasarkan hasil dan analisis yang telah dipaparkan, berikut adalah rangkuman kesimpulan yang dapat diambil dari penelitian ini:

1) Model dasar LSTM, GRU, dan TCN berhasil mengklasifikasikan data audio sinyal jantung *phonocardiogram* pada kelas normal dan abnormal. Evaluasi dilakukan dengan variasi jumlah *epoch* dan nilai *learning rate*. Pada *epoch* 10, model TCN menunjukkan performa terbaik dengan *akurasi* dan *f1-score* masing-masing sebesar 93%. Pada *epoch* 20, model GRU mencapai nilai tertinggi dengan akurasi dan *f1-score* mencapai 94%. Pada *epoch* 30, model GRU menjadi yang terbaik dengan nilai akurasi dan *f1-score* sebesar 93%. Sementara pada percobaan dengan menggunakan *callback* 'Early Stopping' memberikan nilai *epoch* 14, model GRU kembali menjadi yang terbaik dengan nilai akurasi dan *f1-score* sebesar 93%.
2) Ketika dikelompokkan berdasarkan jenis model, model LSTM menunjukkan performa terbaik pada *epoch* 30 dengan *learning rate* 0.0001, mencapai akurasi dan *f1-score* 92%. Model GRU unggul pada *epoch* 20 dengan *learning rate* 0.0001, mencapai akurasi dan *f1-score* 94%. Sedangkan model TCN memiliki performa optimalnya pada *epoch* 10 dengan *learning rate* 1.0000e-4 yang merupakan hasil dari proses otomatisasi dengan bantuan *callback* 'ReduceLROnPlateau', di mana akurasi dan *f1-score* masing-masing mencapai 93%.
3) Penggunaan variansi bobot/*weight* pada proses pelatihan model ensemble tidak menghasilkan perbedaan yang signifikan dalam metrik-metrik yang digunakan, terutama pada akurasi, presisi, *recall*, dan *f1-score*, di mana nilai tersebut berada di kisaran 92% saja. Nilai-nilai metrik ini menunjukkan konsistensi yang sama tanpa peningkatan yang berarti, seiring bertambahnya nilai *epoch*. Satu-satunya perbedaan yang jelas terlihat terjadi pada metrik ROC AUC, di mana perbedaan tersebut hanya berada di empat desimal di belakang koma.

4) Perbandingan performa antara model dasar, khususnya model *Gated Recurrent Unit* (GRU) dengan *epoch* 20 dan *learning rate* 0.0001, serta model *ensemble* telah dilakukan menggunakan beberapa metrik seperti akurasi dan *f1-score*. Hasilnya menunjukkan bahwa model dasar GRU memiliki performa yang lebih baik dibandingkan dengan model *ensemble* terbaik yang diuji. Hal ini mengindikasikan bahwa dalam konteks percobaan yang telah dilakukan, model GRU memberikan hasil yang sedikit lebih baik yaitu dengan nilai performa sebesar 94%.
5) Implementasi model yang telah dilatih menggunakan antarmuka pengguna berbasis *website* juga telah berhasil dibangun, di mana selain dapat melihat hasil prediksi dari model-model terbaik, antarmuka yang dibangun juga dapat memantau bagaimana kinerja model dalam mengklasifikasikan data audio dengan menampilkan beberapa metrik yaitu akurasi, *precision*, *recall*, dan *f1-score* untuk setiap modelnya.

Berdasarkan beberapa poin kesimpulan di atas, dapat disimpulkan secara menyeluruh bahwa, dengan tujuan penelitian ini untuk mengklasifikasikan sinyal jantung *phonocardiogram* menggunakan metode *ensemble learning* dengan pendekatan *soft voting* belum dapat meningkatkan kinerja atau performa model secara signifikan. Di mana pada kenyataannya dengan menggunakan nilai metrik akurasi sebesar 60% dan *f1-score* 40%, performa yang ditunjukkan oleh model dasar GRU lebih unggul dibandingkan dengan model terbaik yang menerapkan metode *ensemble learning* dengan pendekatan *soft voting*. Model GRU menunjukkan performa dengan nilai 94%, sedangkan model *ensemble* hanya memberikan performa sebesar 92% saja.

5.2. Saran

Untuk penelitian selanjutnya, beberapa saran yang dapat dipertimbangkan adalah sebagai berikut:

1) Menggunakan variasi model dasar lainnya di luar LSTM, GRU, dan TCN. Penggunaan model seperti Transformer dapat diuji untuk melihat apakah ada peningkatan kinerja dalam klasifikasi sinyal jantung *phonocardiogram*.

2) Menggunakan pendekatan selain *soft voting* yang telah digunakan, metode *ensemble learning* lain seperti *bagging* (Bootstrap Aggregating), *boosting* (AdaBoost, Gradient Boosting), dan *stacking* dapat dieksplorasi.
3) Melakukan pencarian *hyperparameter* yang lebih variatif untuk menemukan kombinasi yang optimal antara jumlah *epoch*, *learning rate*, dan parameter lainnya.
4) Mengembangkan lebih lanjut antarmuka pengguna yang telah dibangun sehingga dapat diintegrasikan dengan sistem pemantauan klinis untuk aplikasi langsung dalam lingkungan medis.

DAFTAR PUSTAKA

Amal, M. A., Zulherman, D., & Widadi, R. (2023). Klasifikasi Sinyal Phonocardiogram Menggunakan Short Time Fourier Transform dan Convolutional Neural Network. *Jurnal Teknologi Informasi dan Ilmu Komputer*, 10(2), 237. https://doi.org/10.25126/jtiik.20231015424.

Budiman, F., & Awaludin, Y. M. (2023). Optimasi Analisis Kesuburan Tanah dengan Pendekatan Soft Voting Ensemble. Jurnal Teknik Mesin, Elektro dan Ilmu Komputer, 14(2), 261. https://doi.org/10.24176/simet.v14i2.11285.

Dachi, J. M. A. S., & Sitompul, P. (2023). Analisis Perbandingan Algoritma XGBoost dan Algoritma Random Forest Ensemble Learning pada Klasifikasi Keputusan Kredit. *Jurnal Riset Rumpun Matematika dan Ilmu Pengetahuan Alam*, 2(2), 87. https://doi.org/10.55606/jurrimipa.v2i2.1336.

Ganaie, M. A., Hu, M., Malik, A. K., Tanveer, M., & Suganthan, P. N. (2022). Ensemble Deep Learning: A Review. *Engineering Applications of Artificial Intelligence*, *115*, 105151. https://doi.org/10.1016/j.engappai.2022.105151.

Handayani, S. (2021). *Anatomi dan Fisiologi Tubuh Manusia*. Media Sains Indonesia. https://repository.stikes-yogyakarta.ac.id/id/eprint/24/3/buku-anatomi-dan-fisiologi-tubuh-manusia.pdf.

Kaswan, K. S., Dhatterwal, J. S., & Balamurugan, B. (2023). *Python for Beginners*. Chapman and Hall/CRC.

Kemenkes. (2021). *Satu Dari Tiga Kematian Disebabkan Oleh Jantung, Ayo Cegah Serangan Jantung*. Unit Pelayanan Kesehatan - Kemenkes. https://upk.kemkes.go.id/new/satu-dari-tiga-kematian-disebabkan-oleh-jantung-ayo-cegah-serangan-jantung.

Lambert. (2019). *Comprendre Le fonctionnement d'un LSTM et d'un GRU en schemas*. Pensée Artificielle. https://penseeartificielle.fr/comprendre-lstm-gru-fonctionnement-schema.

Li, S., Zhang, W., & Wang, P. (2023). TS2ARCFoRMer: A Multi-Dimensional Time Series Forecasting Framework for Short-Term Load Prediction. *Energies*, 16(15), 5825. https://doi.org/10.3390/en16155825.

Lee, K., Ray, J. & Safta, C. (2021). The Predictive Skill of Convolutional Neural Networks Models for Disease Forecasting. *PLoS ONE*. 16(7). e0254319. https://doi.org/10.1371/journal.pone.0254319.

Lee, S. Y., Huang, P. W., Chiou, J. R., Tsou, C., Liao, Y. Y., & Chen, J. Y. (2019). Electrocardiogram and Phonocardiogram Monitoring System for Cardiac Auscultation. *IEEE transactions on biomedical circuits and systems*, *13*(6), 1471-1482. https://doi.org/10.1109/tbcas.2019.2947694.

Miskiyanto. (2021). Klasifikasi Sinyal Jantung Phonocardiogram Menggunakan Metode *Long Short-Term Memory* (LSTM) [Tugas Akhir Sarjana, Universitas Dinamika]. *Repositori Universitas Dinamika*. https://repository.dinamika.ac.id/id/eprint/6804.

Nurhayati. (2021). *Sistem Kardiovaskuler: Keperawatan Dewasa*. Syiah Kuala University Press.

Oktivasari, P., Haryanto, F., Salman, A. H., Riandini, R., & Suprijadi, S. (2020). A Real-Time Heart Rate Signal Detection using an Electronic Stethoscope with Labview. *Journal of Biomedical Physics and Engineering*, 10(3). https://doi.org/10.31661/jbpe.v0i0.1183.

Pujiastuti, M., Hizkia, I., & Munthe, W. (2023). Gambaran Tekanan Darah pada Masyarakat yang Mengikuti Senam Jantung Sehat di Rambung Merah Tahun 2022. *Jurnal Cakrawala Ilmiah*, *2*(7), 2905. https://doi.org/10.53625/jcijurnalcakrawalailmiah.v2i7.5274.

Rehmer, A., & Kroll, A. (2020). On the Vanishing and Exploding Gradient Problem in Gated Recurrent Units. *IFAC-PapersOnLine*, *53*(2), 1243. https://doi.org/10.1016/j.ifacol.2020.12.1342.

Rizky, M. G. (2021). Analisis Perbandingan Metode LSTM Dan BiLSTM untuk Klasifikasi Sinyal Jantung Phonocardiogram [Tugas Akhir Sarjana, Universitas Dinamika]. *Repositori Universitas Dinamika*. https://repository.dinamika.ac.id/id/eprint/5962.

Rokom. (2022). *Penyakit Jantung Penyebab Utama Kematian, Kemenkes Perkuat Layanan Primer.* Sehat Negeriku - Kemenkes. https://sehatnegeriku.kemkes.go.id/baca/rilis/media/20220929/0541166/penyakit-jantung-penyebab-utama-kematian-kemenkes-perkuat-layanan-primer.

Saluza, I., & Hartati, H. (2022). Neural Network Optimization using Ensemble Method in Forecasting Financial Data. *Jurnal Sistem dan Teknologi Informasi*, *10*(4), 381. https://doi.org/10.26418/justin.v10i4.50771

Šegota, S. B., Lorencin, I., Musulin, J., Štifanić, D., & Car, Z. (2020). Frigate Speed Estimation using CODLAG Propulsion System Parameters and Multilayer Perceptron. *Naše More (Dubrovnik)*, 67(2), 117. https://doi.org/10.17818/nm/2020/2.4.

Shaikh, A. K., Nazir, A., Khalique, N., Shah, A. S., & Adhikari, N. (2023). A New Approach to Seasonal Energy Consumption Forecasting using

Temporal Convolutional Networks. *Results in Engineering*, *19*, 101296. https://doi.org/10.1016/j.rineng.2023.101296.

Triyani, Y., Styorini, W., & Harpawi, N. (2021). Computer Aided Diagnosis (CAD) untuk Phonocardiogram (PCG) Berbasis Fast Fourier Tranform. *Jurnal Elektro dan Mesin Terapan*, *7*(1), 66. https://doi.org/10.35143/elementer.v7i1.4454.

Vincent, R. (2022). I Look Into the Chest: History and Evolution of Stethoscope. *Journal of the Practice of Cardiovascular Sciences*, 8(3), 168-173. https://doi.org/10.4103/jpcs.jpcs_77_22.

Wang, J., Qiang, X., Ren, Z., Wang, H., Wang, Y., & Wang, S. (2023). Time-Series WeLl Performance Prediction Based on Convolutional and Long Short-Term Memory Neural Network Model. Energies, 16(1), 499. https://doi.org/10.3390/en16010499.

Wang, M., Guo, B., Hu, Y., Zhao, Z., Liu, C., & Tang, H. (2022). Transfer Learning Models for Detecting Six Categories of Phonocardiogram Recordings. *Journal of Cardiovascular Development and Disease*, 9(3), 86. https://doi.org/10.3390/jcdd9030086.

Wang, Y., Zheng, L., Gao, Y., & Li, S. (2020). Vibration Signal Extraction Based on FFT and Least Square Method. *IEEE Access*, 8, 224092. https://doi.org/10.1109/access.2020.3044149.

Whiteman, S., Alimi, Y., Carrasco, M., Gielecki, J., Żurada, A., & Loukas, M. (2021). Anatomy of the Cardiac Chambers: A Review of the Left Ventricle. *Translational Research in Anatomy*, 23, 100095. https://doi.org/10.1016/j.tria.2020.100095.

Wu, Z., Lu, C., Sun, Q., Wen, L., He, X., Qin, T., Yan, L., & Wu, C. (2023). Predicting Groundwater Level Based on Machine Learning: A Case Study of the Hebei Plain. Water, 15(4), 823. https://doi.org/10.3390/w15040823.

Yang, L., Li, S., Zhang, Z., & Yang, X. (2020). Classification of Phonocardiogram Signals Based on Envelope Optimization Model and Support Vector Machine. *Journal of Mechanics in Medicine and Biology, 20(1), 1950062.* https://doi.org/10.1142/s0219519419500623.