

***DEEP NEURAL NETWORK DENGAN PENYETELAN  
HYPERPARAMETER BAYESIAN OPTIMIZATION UNTUK  
DETEKSI PENYAKIT JANTUNG***

**Tesis**

untuk memenuhi sebagian persyaratan  
mencapai derajat Sarjana Magister

Program Studi Magister Teknologi Informasi  
Konsentrasi Pervasive Intelligence  
Departemen Teknik Elektro dan Teknologi Informasi



diajukan oleh  
**Fathania Firwan Firdaus**  
**18/437634/PTK/12667**

Kepada  
**PROGRAM PASCASARJANA  
FAKULTAS TEKNIK  
UNIVERSITAS G ADJAH MADA  
YOGYAKARTA  
2021**

**TESIS**  
**DEEP NEURAL NETWORK DENGAN PENYETELAN *HYPERPARAMETER***  
**BAYESIAN OPTIMIZATION UNTUK DETEKSI PENYAKIT JANTUNG**

Dipersiapkan dan disusun oleh

**Fathania Firwan Firdaus**  
18/437634/PTK/12667

Telah dipertahankan di depan dewan penguji

Pada tanggal : **15 Januari 2021**

**Susunan Dewan Penguji**

Ketua	Anggota
	
<b>Dr. Eng. Igi Ardiyanto, S.T., M.Eng.</b>	<b>Ir. Hanung Adi Nugroho, S.T., M.E., Ph.D., IPM.</b>
Anggota	Anggota
	
<b>Dr. Ir. Risanuri Hidayat, M.Sc., IPM.</b>	<b>Dr. Indah Socsanti, S.T., M.T.</b>

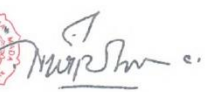

Tesis ini telah diterima sebagai salah satu persyaratan  
untuk memperoleh gelar Magister

Tanggal: **25 Januari 2021**

Pengelola Program Studi: Magister Teknologi Informasi

  
**Dr. Ir. Rudy Hartanto, M.T., IPM.**  
NIP. 196403151990031003

Mengetahui,  
Plt. Ketua Departemen/Wakil Penanggung Jawab Program Studi  
Teknik Elektro dan Teknologi Informasi

  
  
**Ir. Hanung Adi Nugroho, S.T., M.E., Ph.D., IPM.**  
NIP. 197802242002121001



## PERNYATAAN BEBAS PLAGIASI

Saya yang bertanda tangan di bawah ini:

Nama : Fathania Firwan Firdaus  
NIM : 18/437634/PTK/12667  
Tahun terdaftar : 2018  
Program Studi : Magister Teknologi Informasi  
Fakultas/Sekolah : Fakultas Teknik, Departemen Teknik Elektro dan Teknologi Informasi,  
Program Pascasarjana

Menyatakan bahwa dalam dokumen ilmiah Tesis ini tidak terdapat bagian dari karya ilmiah lain yang telah diajukan untuk memperoleh gelar akademik di suatu lembaga Pendidikan Tinggi, dan juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang/ lembaga lain, kecuali yang secara tertulis disitasi dalam dokumen ini dan disebutkan sumbernya secara lengkap dalam daftar pustaka.

Dengan demikian saya menyatakan bahwa dokumen ilmiah ini bebas dari unsur-unsur plagiasi dan apabila dokumen ilmiah Tesis ini di kemudian hari terbukti merupakan plagiasi dari hasil karya penulis lain dan/atau dengan sengaja mengajukan karya atau pendapat yang merupakan hasil karya penulis lain, maka penulis bersedia menerima sanksi akademik dan/atau sanksi hukum yang berlaku.

Yogyakarta, 22 Januari 2021



METERAI  
TEMPEL  
561B8AJX018068814

Fathania Firwan Firdaus  
18/437634/PTK/12667

## PRAKATA

Puji syukur ke hadirat Allah SWT yang telah melimpahkan rahmat dan barokah-Nya sehingga penulis dapat menyelesaikan tesis dengan judul “*Deep Neural Network* dengan Penyetelan *Hyperparameter Bayesian Optimization* untuk Deteksi Penyakit Jantung”. Laporan tesis ini disusun untuk memenuhi salah satu syarat dalam memperoleh gelar *Master of Engineering (M.Eng.)* pada Program Studi Magister Teknik Elektro Fakultas Teknik Universitas Gadjah Mada Yogyakarta.

Dalam melakukan penelitian dan penyusunan laporan tesis ini penulis telah mendapatkan banyak dukungan dan bantuan dari berbagai pihak. Penulis mengucapkan terima kasih yang tak terhingga kepada:

1. Bapak Ir. Hanung Adi Nugroho, S.T., M.E., Ph.D., IPM. selaku dosen pembimbing utama, dan Ibu Dr. Indah Soesanti, S.T., M.T. selaku dosen pembimbing pendamping, yang telah dengan penuh kesabaran dan ketulusan memberikan ilmu dan bimbingan terbaik kepada penulis.
2. Bapak Ir. Hanung Adi Nugroho, S.T., M.E., Ph.D., IPM. selaku Ketua Departemen Teknik Elektro dan Teknologi Informasi dan Bapak Dr. Ir. Rudy Hartanto, M.T., IPM. selaku Ketua Program Studi Magister Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada yang memberikan izin kepada penulis untuk belajar.
3. Para Dosen Program Studi Magister Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada yang telah memberikan bekal ilmu kepada penulis.
4. Para Karyawan/wati Departemen Tekni Elektro dan Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada yang telah membantu penulis dalam proses belajar.
5. Kedua Orang Tuaku Bapak Firwan Firdaus dan Ibu Sri Haryanti yang selalu memberikan doa, kasih sayang, semangat dan dukungan materi maupun moril yang tidak pernah putus selama proses perkuliahan hingga terselesaikannya

penelitian ini.

6. Fidela Firwan Firdaus dan Favian Firwan Firdaus selaku kakak dan seluruh keluargaku tercinta yang telah memberikan dukungan dan doa yang tulus selama proses perkuliahan hingga terselesaikannya penelitian ini.
7. Teman-teman Magister Teknologi Informasi angkatan 2018 yang telah berjuang bersama dan memberikan motivasi dalam menyelesaikan penelitian ini.
8. Kolega *Intelligent System Research Group* di Departemen Teknik Elektro dan Teknologi Informasi untuk diskusi dan motivasi yang menginspirasi.
9. Semua pihak yang telah memberikan dukungan dan bantuan pada penulisan tesis ini yang tidak dapat disebutkan satu-persatu

Penulis menyadari sepenuhnya bahwa laporan tesis ini masih jauh dari sempurna, untuk itu semua jenis saran, kritik dan masukan yang bersifat membangun sangat penulis harapkan. Akhir kata, semoga tulisan ini dapat memberikan manfaat dan memberikan wawasan tambahan bagi para pembaca dan khususnya bagi penulis sendiri.

Yogyakarta, 22 Januari 2021

Fathania Firwan Firdaus

## **ARTI LAMBANG DAN SINGKATAN**

WHO	=	World Health Organization
CVD	=	Cardiovascular Disease
PJK	=	Penyakit Jantung Koroner
CHD	=	Coronary Heart Disease
CAD	=	Coronary Artery Disease
IHD	=	Ischemic Heart Disease
DNN	=	Deep Neural Network
TP	=	True Positive
TN	=	True Negative
FP	=	False Positive
FN	=	False Negative

## ABSTRACT

Heart disease is a disease that most often causes death in the world. Detecting heart disease in the early stages of symptoms in patients is very important because then treatment can be done as soon as possible. Along with the development of information technology, many researchers are conducting research on systems for detecting computer-based heart disease. In the last decade, deep learning approaches have been widely implemented in terms of biomedicine and health care. A deep neural network (DNN) can improve accuracy and generalization and does not require manual feature engineering. In previous research of DNN to detect heart disease, the DNN hyperparameters were still tuned manually. Along with large number of hyperparameters and range of hyperparameter values, manual tuning is difficult to reproduce because it requires large human effort and can take a long time to produce the optimal hyperparameter.

In this study, Bayesian optimization is used to tune the DNN hyperparameters in detecting heart disease. Thus, it can improve the accuracy of DNN performance in detecting heart disease. Experiments is conducted 20 times using a randomized Cleveland dataset. The performance of DNN with hyperparameter tuning using Bayesian optimization is also compared with manual tuning, grid search, and random search techniques.

Bayesian optimization results in better accuracy than manual tuning, grid search, and random search to tune DNN hyperparameters in detecting heart disease. Performance evaluation of DNN with Bayesian optimization results in an average accuracy of 88%, a sensitivity of 86.31%, a precision of 86.89%, a specificity of 89.09%, and an F1 score of 86.48%. Bayesian optimization can tune hyperparameters faster than grid search, but slower than random search with less time difference. Thus, Bayesian optimization is preferred as a method of tuning the DNN hyperparameters in detecting heart disease.

**Keywords :** heart disease, deep neural network, Bayesian optimization

## INTISARI

Penyakit jantung merupakan penyakit yang paling sering menyebabkan kematian di dunia. Mendeteksi penyakit jantung pada gejala tahap awal pada pasien sangat penting karena dengan begitu pengobatan dapat dilakukan secepatnya. Seiring berkembangnya teknologi informasi, banyak peneliti yang melakukan penelitian mengenai sistem untuk mendeteksi penyakit jantung berbasis komputer. Dalam dekade terakhir, pendekatan *deep learning* banyak diimplementasikan dalam hal biomedis dan perawatan kesehatan. *Deep neural network* (DNN) dapat meningkatkan ketepatan dan generalisasi dan tidak memerlukan rekayasa fitur manual. Pada penelitian DNN dalam mendeteksi penyakit jantung sebelumnya, *hyperparameter* DNN masih disetel secara manual. Seiring dengan banyaknya jumlah *hyperparameter* dan jangkauan nilai *hyperparameter*, penyetelan manual sulit untuk direproduksi karena membutuhkan usaha manusia yang besar dan dapat membutuhkan waktu lama dalam menghasilkan *hyperparameter* yang optimal.

Pada penelitian ini, *Bayesian optimization* digunakan untuk menyetel *hyperparameter* DNN dalam mendeteksi penyakit jantung. Dengan demikian dapat meningkatkan akurasi kinerja DNN dalam mendeteksi penyakit jantung. Pengujian dilakukan sebanyak 20 kali menggunakan *dataset Cleveland* yang diacak. Kinerja DNN dengan penyetelan *hyperparameter* menggunakan *Bayesian optimization* juga dibandingkan dengan teknik penyetelan manual, *grid search* dan *random search*.

*Bayesian optimization* mendapatkan hasil akurasi lebih baik dibandingkan penyetelan manual, *grid search*, dan *random search* untuk menyetel *hyperparameter* DNN dalam mendeteksi penyakit jantung. Evaluasi kinerja DNN dengan *Bayesian optimization* menghasilkan rata-rata akurasi 88%, sensitivitas 86,31%, presisi 86,89%, spesifisitas 89,09%, dan skor F1 86,48%. *Bayesian optimization* menyetel *hyperparameter* lebih cepat *grid search*, tapi lebih lambat dari *random search* dengan selisih waktu yang sedikit. Dengan demikian, *Bayesian optimization* lebih dipilih sebagai metode penyetelan *hyperparameter* DNN dalam mendeteksi penyakit jantung.

**Kata kunci** – penyakit jantung, *deep neural network*, *Bayesian optimization*

## DAFTAR ISI

PERNYATAAN BEBAS PLAGIASI.....	iii
PRAKATA.....	iv
ARTI LAMBANG DAN SINGKATAN .....	vi
ABSTRACT.....	vii
INTISARI.....	viii
DAFTAR ISI.....	ix
DAFTAR GAMBAR .....	xi
DAFTAR TABEL.....	xii
BAB I PENDAHULUAN .....	1
1.1 Latar Belakang .....	1
1.2 Perumusan Masalah.....	5
1.3 Keaslian Penelitian .....	6
1.4 Tujuan Penelitian.....	12
1.5 Manfaat Penelitian.....	12
BAB II TINJAUAN PUSTAKA DAN LANDASAN TEORI.....	13
2.1 Tinjauan Pustaka.....	13
2.1.1 Metode Klasifikasi Penyakit Jantung.....	13
2.1.2 Teknik Penyetelan <i>Hyperparameter</i> Otomatis.....	17
2.2 Landasan Teori .....	20
2.2.1 Penyakit Jantung .....	20
2.2.2 <i>Data Mining</i> .....	22
2.2.3 <i>Deep Learning</i> .....	23
2.2.4 <i>Deep Neural Network</i> .....	24
2.2.5 <i>Hyperparameter</i> .....	30
2.2.5.1 Penyetelan <i>Hyperparameter</i> Manual.....	30
2.2.5.2 Penyetelan <i>Hyperparameter</i> Otomatis.....	31
2.2.6 <i>Standardization</i> .....	37
2.2.7 Kinerja Klasifikasi .....	38
2.3 Hipotesis .....	39
BAB III METODOLOGI.....	40
3.1 Alat dan Bahan .....	40
3.1.1 Alat.....	40
3.1.2 Bahan.....	41
3.2 Metode.....	42
3.2.1 <i>Data Pre-processing</i> .....	43
3.2.1.1 <i>Data Cleaning</i> .....	44
3.2.1.2 Konversi Data .....	45
3.2.1.3 Pembagian Data .....	45

3.2.1.4	<i>Standardization</i> .....	47
3.2.2	Pengembangan <i>Deep Neural Network</i> dengan Penyetelan <i>Hyperparameter</i> .....	47
3.2.2.1	Model <i>Deep Neural Network</i> .....	48
3.2.2.2	Penyetelan <i>Hyperparameter</i> .....	51
3.2.3	Klasifikasi Data.....	54
3.2.4	Evaluasi Kinerja.....	55
BAB IV HASIL DAN PEMBAHASAN.....		57
4.1	<i>Data Pre-processing</i> .....	57
4.1.1	<i>Data Cleaning</i> .....	57
4.1.2	Konversi Data.....	58
4.1.3	Pembagian Data .....	58
4.1.4	<i>Standardization</i> .....	59
4.2	Pengembangan <i>Deep Neural Network</i> dengan Penyetelan <i>Hyperparameter</i> .....	60
4.2.1	Model <i>Deep Neural Network</i> .....	60
4.2.2	Penyetelan <i>Hyperparameter</i> Manual .....	61
4.2.3	Penyetelan <i>Hyperparameter</i> Otomatis.....	63
4.2.3.1	<i>Grid Search</i> .....	63
4.2.3.2	<i>Random Search</i> .....	65
4.2.3.3	<i>Bayesian Optimization</i> .....	67
4.3	Klasifikasi.....	69
4.4	Evaluasi .....	75
BAB V KESIMPULAN DAN SARAN.....		83
5.1	Kesimpulan.....	83
5.2	Saran.....	84
DAFTAR PUSTAKA .....		85
LAMPIRAN.....		L-1

## DAFTAR GAMBAR

Gambar 2.1 Arteri normal versus arteri yang tersumbat .....	20
Gambar 2.2 Jantung dengan kerusakan otot dan arteri tersumbat .....	21
Gambar 2.3 Arsitektur <i>deep neural network</i> .....	25
Gambar 2.4 Kurva <i>sigmoid</i> .....	27
Gambar 2.5 <i>Grid search</i> .....	32
Gambar 2.6 <i>Random search</i> .....	33
Gambar 3.1 Diagram alir penelitian .....	43
Gambar 3.2 Diagram alir <i>data pre-processing</i> .....	44
Gambar 3.3 Pembagian data .....	46
Gambar 3.4 Alur pengembangan DNN dengan penyetelan <i>hyperparameter</i> .....	48
Gambar 3.5 Rancangan arsitektur DNN .....	48
Gambar 3.6 Diagram alir klasifikasi data .....	55
Gambar 4.1 <i>Code</i> untuk pembagian data .....	59
Gambar 4.2 <i>Code</i> untuk <i>standardization</i> .....	59
Gambar 4.3 Hasil <i>standardization</i> untuk <i>dataset training</i> dan <i>dataset testing</i> .....	60
Gambar 4.4 <i>Code</i> untuk membangun model DNN .....	61
Gambar 4.5 <i>Code</i> penyetelan manual .....	61
Gambar 4.6 <i>Code grid search</i> .....	64
Gambar 4.7 <i>Code random search</i> .....	66
Gambar 4.8 <i>Code Bayesian optimization</i> .....	68
Gambar 4.9 Perbandingan waktu penyetelan <i>hyperparameter</i> DNN oleh <i>grid search</i> , <i>random search</i> , dan <i>Bayesian optimization</i> .....	81

## DAFTAR TABEL

Tabel 1.1 Penelitian-penelitian terkait.....	7
Tabel 2.1 <i>Confusion matrix</i> .....	38
Tabel 3.1 Rincian data <i>missing values</i> .....	45
Tabel 3.2 <i>Confusion matrix</i> hasil klasifikasi .....	55
Tabel 4.1 Karakteristik <i>dataset Cleveland</i> sebelum <i>data cleaning</i> .....	58
Tabel 4.2 Karakteristik <i>dataset Cleveland</i> setelah <i>data cleaning</i> .....	58
Tabel 4.3 Karakteristik <i>dataset Cleveland</i> setelah konversi data.....	58
Tabel 4.4 Hasil penyetelan manual .....	62
Tabel 4.5 Ruang pencarian <i>hyperparameter</i> .....	63
Tabel 4.6 Hasil <i>grid search</i> .....	64
Tabel 4.7 Hasil <i>random search</i> .....	66
Tabel 4.8 Hasil <i>Bayesian optimization</i> .....	68
Tabel 4.9 <i>Confusion matrix</i> DNN dengan penyetelan manual.....	70
Tabel 4.10 <i>Confusion matrix</i> DNN dengan <i>grid search</i> .....	70
Tabel 4.11 <i>Confusion matrix</i> DNN dengan <i>random search</i> .....	71
Tabel 4.12 <i>Confusion matrix</i> DNN dengan <i>Bayesian optimization</i> .....	72
Tabel 4.13 Kombinasi <i>hyperparameter</i> dan akurasi klasifikasi model DNN dengan penyetelan manual, <i>grid search</i> , <i>random search</i> , dan <i>Bayesian optimization</i> .....	73
Tabel 4.14 Perbandingan akurasi model DNN dengan penyetelan manual, <i>grid search</i> , <i>random search</i> , dan <i>bayesian optimization</i> .....	75
Tabel 4.15 Perbandingan sensitivitas model DNN dengan penyetelan manual, <i>grid search</i> , <i>random search</i> , dan <i>bayesian optimization</i> .....	76
Tabel 4.16 Perbandingan presisi model DNN dengan penyetelan manual, <i>grid search</i> , <i>random search</i> , dan <i>bayesian optimization</i> .....	77
Tabel 4.17 Perbandingan spesifisitas model DNN dengan penyetelan manual, <i>grid search</i> , <i>random search</i> , dan <i>bayesian optimization</i> .....	78
Tabel 4.18 Perbandingan skor F1 model DNN dengan penyetelan manual, <i>grid search</i> , <i>random search</i> , dan <i>bayesian optimization</i> .....	79
Tabel 4.19 Rangkuman perbandingan kinerja model DNN dengan penyetelan manual, <i>grid search</i> , <i>random search</i> , dan <i>bayesian optimization</i> .....	80

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Menurut *World Health Organization* (WHO), penyakit jantung atau *Cardiovascular Disease* (CVD) menjadi penyebab kematian nomor satu di dunia. Diperkirakan 17,9 juta orang meninggal karena penyakit jantung atau mewakili 31% dari kematian di dunia [1]. Penyakit jantung menjadi penyakit yang paling banyak menyebabkan kematian dibandingkan penyakit lainnya. Data WHO menunjukkan bahwa jumlah kematian akibat penyakit jantung mengalami peningkatan dari tahun 2000 sampai 2016, yaitu sekitar 7 juta orang pada tahun 2000 dan sekitar 9,4 juta orang pada tahun 2016 [2].

*Cardiovascular Disease* (CVD) adalah istilah untuk sekelompok gangguan jantung dan pembuluh darah [1]. Penyakit Jantung Koroner (PJK) atau *Coronary Heart Disease* (CHD), sering juga disebut *Coronary Artery Disease* (CAD) atau *Ischemic Heart Disease* (IHD) [3] adalah penyakit jantung yang paling umum diderita manusia [4]. Faktor risiko penyakit jantung di antaranya adalah peningkatan tekanan darah, glukosa darah, lemak darah, dan obesitas [5]. Faktor risiko tersebut berasal dari gaya hidup yang tidak sehat, seperti kurangnya aktivitas fisik, penggunaan tembakau ataupun alkohol. Pasien dengan penyakit jantung memerlukan perawatan mahal seperti operasi *bypass* jantung, rehabilitasi pascaoperasi dan pengobatan seumur hidup [6]. Mendeteksi penyakit jantung pada tahap awal dari gejala pada pasien sangatlah penting, sehingga pengobatan dapat dilakukan sejak dini.

Peningkatan dan kompleksitas data di bidang kesehatan membuat teknologi diadopsi dalam praktik klinis [7]. Banyak peneliti telah melakukan studi tentang diagnosis berbantuan komputer untuk penyakit jantung menggunakan *data mining* dan *machine learning* [8]. *Data mining* adalah proses menganalisis data besar untuk

mengekstrak pengetahuan atau pola [9]. *Machine learning* memainkan peran kunci dalam *data mining* dengan melakukan analisis data dan penemuan pola [10]. Ada banyak teknik yang telah diterapkan dalam deteksi penyakit jantung, seperti *neural network* [11], *Linear Discriminant Analysis (LDA)* [12], *Support Vector Machine (SVM)* [13], *K-Nearest Neighbor (K -NN)* [14], metode *ensemble* [15] [16] [17] [18] dan teknik *hybrid* [19] [20].

Dalam dekade terakhir, penggunaan *deep learning* dalam domain medis telah menerima perhatian yang luar biasa [21]. *Deep learning* adalah cabang *machine learning* yang menggunakan lapisan multi-pemrosesan untuk memodelkan abstraksi tingkat tinggi dalam data [22]. Teknik *deep learning* dapat mengurangi kesalahan klasifikasi dan lebih kuat terhadap *noise* daripada pendekatan lainnya [23]. Konsep *deep learning* didasarkan pada arsitektur *neural network*. *Deep learning* memiliki berbagai arsitektur, seperti *deep neural network (DNN)*, *recurrent neural network (RNN)*, dan *convolutional neural network (CNN)* [24]. DNN adalah *neural network* dengan banyak *hidden layer* di antara *input layer* dan *output layer* dimana hubungan antar lapisan bersifat satu arah. DNN digunakan untuk menyelesaikan masalah yang menggunakan data terstruktur yang tersimpan dalam format tabel. RNN adalah jaringan dengan koneksi yang membentuk siklus terarah. RNN digunakan untuk mengatasi masalah pembelajaran yang melibatkan data *sequence* seperti pengenalan tulisan tangan, pengenalan ucapan, dan mesin penerjemah. CNN dirancang untuk secara otomatis dan adaptif mempelajari hierarki spasial fitur melalui algoritme *backpropagation* dengan menggunakan beberapa blok penyusun seperti *convolution layer*, *pooling layer*, and *fully connected layer*. CNN umumnya diterapkan untuk menganalisis citra visual.

*Deep neural network* menggunakan arsitektur *neural network* yang lebih dalam yang dapat mewakili fungsi dengan kompleksitas yang lebih tinggi [25]. Perbedaan antara *neural network* dan *deep neural network* terletak pada kedalaman model, *neural network* dengan dua atau lebih *hidden layer* disebut *deep neural*

*network* karena jaringan telah menjadi kompleks dengan lebih dari satu *hidden layer* [26]. *Deep neural network* memiliki manfaat dibandingkan dengan *neural network*, karena arsitektur yang lebih dalam dapat meningkatkan ketepatan dan generalisasi setelah mempelajari contoh baru [27]. *Deep neural network* memiliki keunggulan dibandingkan dengan teknik *machine learning* tradisional karena membutuhkan lebih sedikit pengetahuan domain untuk menyelesaikan masalah [28]. *Deep neural network* mempelajari fitur langsung dari data tanpa adanya rekayasa fitur manual. Model *machine learning* tradisional seperti *decision tree* dan *Support Vector Machine* (SVM) tidak efisien karena membutuhkan sejumlah besar upaya manusia untuk menentukan pengetahuan sebelumnya dalam model [29].

Penelitian yang menggunakan DNN untuk mendeteksi penyakit jantung sebelumnya sudah pernah dilakukan. Miao dan Miao [30] mengembangkan DNN untuk klasifikasi biner untuk deteksi penyakit jantung dengan arsitektur yang terdiri atas *input layer* yang menerima 28 input, dua *hidden layers* yang masing-masing memiliki 105 dan 42 neuron, dan *output layer* yang memiliki 1 neuron. Selain itu, Ashraf et al. [31] mengembangkan model DNN untuk melakukan klasifikasi *multiclass* untuk deteksi penyakit jantung dengan arsitektur yang terdiri atas *input layer* yang menerima 28 input, dua *hidden layers* yang masing-masing memiliki 105 dan 42 neuron, dan *output layer* yang memiliki 1 neuron. Akan tetapi, pada kedua penelitian tersebut [30] [31] *hyperparameter* DNN masih ditentukan secara manual. *Hyperparameter* adalah parameter yang nilainya digunakan untuk mengontrol proses pembelajaran. Nilai *hyperparameter* tidak didapatkan langsung saat proses *training* sehingga harus ditentukan sebelum memulai *training* model. DNN mempunyai beberapa *hyperparameter* yang perlu ditentukan sebelum *training*, seperti jumlah *hidden layer*, jumlah *hidden units*, ukuran *batch*, dan *epochs*. *Hyperparameter* pada DNN dapat berpengaruh pada akurasi kinerja model. Menentukan *hyperparameter* yang optimal untuk model DNN merupakan tantangan tersendiri. Penyetelan *hyperparameter* secara manual mencoba beberapa nilai *hyperparameter* secara manual dan dilakukan berkali-kali hingga mendapat

skor validasi yang optimal. Namun, seiring dengan banyaknya jumlah *hyperparameter* dan jangkauan nilai *hyperparameter*, penyetelan manual sulit untuk direproduksi karena membutuhkan usaha manusia yang besar dan dapat membutuhkan waktu lama dalam menghasilkan *hyperparameter* yang menghasilkan akurasi yang baik.

Pada penelitian lainnya, penyetelan *hyperparameter* otomatis telah dikembangkan untuk menangani kekurangan pada penyetelan manual. Shekar dan Dagnev [32] mengusulkan penyetelan *hyperparameter grid search* untuk mengoptimalkan *hyperparameter random forest* untuk klasifikasi data kanker *microarray*. *Grid search* mendapatkan *hyperparameter* optimal dengan melakukan pencarian secara menyeluruh dari ruang *hyperparameter* yang ditentukan. Namun, *grid search* menjadi lambat jika jumlah *hyperparameter* yang disetel dan rentang nilai *hyperparameter* cukup besar [33]. Mantovani et al. [34] membandingkan *random search* dengan *grid search*, dengan *Genetic Algorithm (GA)*, *Particle Swarm Optimization (PSO)*, dan *Estimation of Distribution Algorithms (EDA)* dalam penyetelan *hyperparameter SVM*. *Random search* mencoba kombinasi *hyperparameter* secara acak dari rentang nilai yang ditentukan. *Random search* mampu menghasilkan akurasi kinerja yang baik dengan biaya komputasi yang lebih rendah. Namun, *random search* kurang stabil digunakan untuk melatih model klasifikasi yang kompleks [33]. Padierna et. al [35] mengusulkan *Estimation of Distribution Algorithms (EDA)* untuk menyetel *hyperparameter SVM*. EDA memilih *hyperparameter* optimal dengan mengeksplorasi ruang solusi dengan secara berulang membangun dan mengambil sampel model probabilistik eksplisit dari solusi kandidat yang memandu pencarian. Namun, EDA memiliki beberapa kelemahan, yaitu hilangnya keragaman, kurangnya penggunaan informasi lokal solusi, dan terjebak dalam optima lokal. Wu et al. [36] melakukan penelitian mengenai penyetelan *hyperparameter* menggunakan *Bayesian optimization* untuk menemukan *hyperparameter* terbaik untuk model pembelajaran, seperti *random forest*, CNN, RNN, dan *Multi-Grained Cascade Forest*. *Bayesian optimization*

memilih *hyperparameter* optimal dengan memperhitungkan evaluasi sebelumnya saat memilih set *hyperparameter* untuk evaluasi berikutnya. *Bayesian optimization* merupakan algoritme optimasi yang sangat efektif dalam menyelesaikan masalah optimasi dimana fungsi tujuan optimasi tidak diketahui. *Bayesian optimization* mampu menghasilkan akurasi tinggi dengan waktu penyetalan yang singkat [36]. Hnin dan Jeenanunta [37] membandingkan *Bayesian optimization* dengan *Genetic Algorithm* (GA) dan *Particle Swarm Optimization* (PSO) untuk menyetel *hyperparameter* pada *Support Vector Regression* (SVR). *Bayesian optimization* menghasilkan kinerja lebih unggul daripada GA dan PSO. GA dan PSO mudah terjebak pada minimum lokal, sedangkan *Bayesian Optimization* mampu melampaui minimum lokal [37]. Ada pula, Gao dan Ding [38] meneliti mengenai *Bayesian optimization* untuk mengoptimalkan *hyperparameter* model *ensemble Extreme Gradient Boosting* (XGBoost) dalam prediksi penyakit kanker payudara dan jantung. Pada klasifikasi penyakit jantung, dilakukan proses seleksi fitur menggunakan *recursive feature elimination* (RFE) sebelum klasifikasi. *Bayesian optimization* memiliki kinerja yang stabil dalam penyetalan *hyperparameter* XGBoost [38].

## 1.2 Perumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan diatas, permasalahan yang dibahas adalah dalam penelitian sebelumnya mengenai deteksi penyakit jantung menggunakan *deep neural network*, *hyperparameter* masih disetel secara manual. Penyetalan *hyperparameter* dapat berpengaruh pada akurasi kinerja klasifikasi untuk mendeteksi penyakit jantung. Penyetalan secara manual tidak efisien karena membutuhkan banyak usaha manusia untuk menemukan *hyperparameter* yang dapat menghasilkan akurasi yang baik. Sehingga dibutuhkan teknik penyetalan *hyperparameter* yang efisien dan menghasilkan akurasi yang baik untuk *deep neural network* dalam mendeteksi penyakit jantung.

### 1.3 Keaslian Penelitian

Penelitian mengenai deteksi penyakit jantung menggunakan *machine learning* telah banyak dilakukan sebelumnya. Berbagai teknik *machine learning* seperti *machine learning* tradisional, *ensemble*, *hybrid*, dan *deep learning* telah diimplementasikan untuk mendeteksi penyakit jantung. Pada penelitian mendeteksi penyakit jantung dengan menggunakan *machine learning* tradisional [11] [12] [13] [14], *ensemble* [17] dan *hybrid* [19] [20], dilakukan proses seleksi fitur sebelum klasifikasi untuk mengetahui variabel yang paling berpengaruh dalam klasifikasi penyakit jantung. Adanya proses seleksi fitur secara manual membuat sistem kurang efisien. Hal ini berbeda dengan teknik *deep neural network* yang tidak memerlukan adanya seleksi fitur manual dalam menyelesaikan masalah. Akan tetapi, pada penelitian untuk mendeteksi penyakit jantung menggunakan *deep neural network* masih menggunakan penyetelan *hyperparameter* manual [30] [31]. Dalam membangun *deep neural network*, *hyperparameter* yang digunakan berpengaruh pada akurasi kinerja model. Penyetelan *hyperparameter* secara manual membutuhkan usaha manusia yang besar dan dapat menghabiskan waktu yang lama untuk mendapatkan *hyperparameter* yang menghasilkan akurasi yang baik. Pada penelitian lainnya, penyetelan *hyperparameter* otomatis dikembangkan untuk mengatasi keterbatasan penyetelan manual. Teknik *grid search* diusulkan untuk menyetel *hyperparameter* untuk *random forest* dan *K-Nearest Neighbour* (K-NN) [32] [39]. Namun, *grid search* menjadi lambat jika jumlah *hyperparameter* yang disetel dan rentang nilai *hyperparameter* cukup besar [33]. *Random search* diusulkan untuk menyetel *hyperparameter* pada SVM dan MLP [34] [40]. Namun, *random search* kurang stabil digunakan untuk melatih model klasifikasi yang kompleks [33]. Padierna et.al [35] mengusulkan *Estimation of Distribution Algorithms* (EDA) untuk menyetel *hyperparameter* SVM. Namun, EDA memiliki beberapa kelemahan yaitu hilangnya keragaman, kurangnya penggunaan informasi lokal solusi, dan terjebak dalam optima lokal. Wu et al. [36] mengusulkan penyetelan *hyperparameter* menggunakan *Bayesian optimization* untuk

mengoptimalkan *hyperparameter* beberapa model pembelajaran, yaitu *random forest*, CNN, RNN, dan *Multi-Grained Cascade Forest*. *Bayesian optimization* memilih *hyperparameter* optimal dengan memperhitungkan evaluasi sebelumnya saat memilih set *hyperparameter* untuk evaluasi berikutnya. *Bayesian optimization* mampu menghasilkan akurasi tinggi dengan waktu penyetalan yang singkat [36]. Hnin dan Jeenanunta [37] membandingkan *Bayesian optimization* dengan *Genetic Algorithm* (GA) dan *Particle Swarm Optimization* (PSO) untuk menyetal *hyperparameter* pada *Support Vector Regression* (SVR). *Bayesian Optimization* memiliki keunggulan dari pada PSO dan GA, yaitu mampu melampaui minimum lokal [37]. Gao dan Ding [38] mengoptimalkan *hyperparameter* model *ensemble XGBoost* dengan *Bayesian optimization* dalam prediksi penyakit kanker payudara dan jantung. *Bayesian optimization* memiliki kinerja yang stabil dalam penyetalan *hyperparameter* [38].

Tabel 1.1 menjabarkan penelitian-penelitian sebelumnya terkait topik penelitian ini dan serta perbedaan dibandingkan penelitian ini. Topik yang berhubungan pada penelitian ini adalah deteksi penyakit jantung, *deep neural network*, dan penyetalan *hyperparameter*.

Tabel 1.1 Penelitian-penelitian terkait

No.	Penulis, Tahun, Judul	Metode	Hasil	Perbedaan
1.	Mantovani et. al (2015), <i>Effectiveness of Random Search in SVM hyper-parameter tuning</i> [34]	<i>Hyperparameter</i> SVM disetel menggunakan <i>random search</i> diuji pada 70 <i>dataset</i> UCI dan dibandingkan dengan teknik <i>grid search</i> , <i>Genetic Algorithm</i> (GA), <i>Particle Swarm Optimization</i> (PSO), dan <i>Estimation of Distribution Algorithms</i> (EDA).	<i>Random search</i> mampu menghasilkan akurasi kinerja yang baik dengan biaya komputasi yang lebih rendah.	Metode klasifikasi berbeda, yaitu menggunakan SVM, penyetalan <i>hyperparameter</i> yang diusulkan menggunakan <i>random search</i> .

No.	Penulis, Tahun, Judul	Metode	Hasil	Perbedaan
2.	Padierna et. al (2017) <i>Hyper-Parameter Tuning for Support Vector Machines by Estimation of Distribution Algorithms</i> [35]	<i>Hyperparameter</i> SVM disetel menggunakan dua metode <i>Estimation of Distribution Algorithms</i> (EDA), yaitu <i>Univariate Marginal Distribution Algorithm</i> (UMDA) dan <i>the Boltzmann Univariate Marginal Distribution Algorithm</i> (BUMDA), dibandingkan dengan <i>random search</i> .	Metode EDA khususnya BUMDA mencapai hasil yang lebih baik daripada <i>random search</i> .	Metode klasifikasi menggunakan SVM, dan penyetelan <i>hyperparameter</i> menggunakan EDA.
3.	Maheswari dan Jasmine (2017), <i>Neural Network based Heart Disease Prediction</i> [11]	Deteksi penyakit jantung menggunakan seleksi fitur <i>logistic regression</i> dan pengklasifikasi <i>neural network</i> .	Model menghasilkan akurasi 84%, sensitivitas 91,14%, dan spesifisitas 77,5%.	Metode klasifikasi berbeda, yaitu menggunakan <i>neural network</i> , penyetelan <i>hyperparameter</i> secara manual, dan menggunakan seleksi fitur manual.
4.	Pouriyeh et. al (2017), <i>A Comprehensive Investigation and Comparison of Machine Learning Techniques in the Domain of Heart Disease</i> [15]	Prediksi penyakit jantung menggunakan <i>ensemble bagging</i> , <i>boosting</i> , <i>stacking</i> .	<i>Support Vector Machine</i> (SVM) dengan <i>boosting</i> mendapatkan hasil terbaik dengan akurasi 84,81%.	Metode berbeda, menggunakan <i>ensemble</i> dan pembagian data menggunakan <i>10-fold cross-validation</i> .
5.	El-Sayed (2018), <i>Linear Discriminant Analysis for An Efficient Diagnosis of Heart Disease via Attribute Filtering Based on Genetic Algorithm</i> [12]	Deteksi penyakit jantung menggunakan klasifikasi <i>Linear Discriminant Analysis</i> (LDA), SVM, K-NN, dan <i>Naive bayes</i> . Teknik seleksi fitur menggunakan GA dan <i>Principle Component Analysis</i> (PCA).	Model LDA dengan GA menghasilkan akurasi tertinggi, yaitu 89,07% untuk klasifikasi biner dan 67.22% untuk <i>multiclass</i> .	Metode klasifikasi berbeda dan masih menggunakan seleksi fitur manual.

No.	Penulis, Tahun, Judul	Metode	Hasil	Perbedaan
6.	Vijayashree dan Sultana (2018), <i>A Machine Learning Framework for Feature Selection in Heart Disease Classification Using Improved Particle Swarm Optimization with Support Vector Machine Classifier</i> [13]	Klasifikasi penyakit jantung menggunakan SVM, <i>random forest</i> , <i>naïve, bayes</i> , dan MLP. Seleksi fitur menggunakan <i>relief, correlation, filter, info gain, consistency, chi-squared, gain ratio</i> , dan PSO, PSO-SVM.	Model SVM dengan seleksi fitur PSO-SVM menghasilkan akurasi tertinggi 88,22%.	Metode klasifikasi berbeda dan masih menggunakan seleksi fitur manual.
7.	Pawlovsky (2018), <i>An Ensemble Based on Distances for a K-NN Method for Heart Disease Diagnosis</i> [16]	Prediksi penyakit jantung menggunakan <i>ensemble</i> berdasarkan jarak untuk metode <i>K-Nearest Neighbor</i> (K-NN). <i>Hyperparameter</i> jarak (k) dicoba dengan beberapa pengaturan.	Metode yang diusulkan memberikan akurasi rata-rata 85% untuk setiap konfigurasi yang diuji.	Metode berbeda, menggunakan <i>ensemble</i> K-NN dengan penyetelan <i>hyperparameter</i> secara manual.
8.	Normawati dan Winarti (2018), <i>Feature Selection with Combination Classifier use Rules-Based Data Mining for Diagnosis of Coronary Heart Disease</i> [19]	Deteksi penyakit jantung koroner dengan menggunakan klasifikasi VPRS, RIPPER, dan kombinasi VPRS+RIPPER. Seleksi fitur menggunakan MTF, VPRS, dan kombinasi MTF+VPRS.	Metode klasifikasi VPRS+RIPPER dengan seleksi fitur VPRS mendapatkan hasil terbaik dengan akurasi 88,89%, sensitivitas 100%, dan spesifisitas 77,08%.	Metode klasifikasi berbeda dan masih menggunakan seleksi fitur manual.
9.	Miao dan Miao (2018), <i>Coronary Heart Disease Diagnosis using Deep Neural Networks</i> [30]	Klasifikasi biner untuk penyakit jantung koroner menggunakan <i>deep neural network</i> (DNN).	Hasil evaluasi akurasi 83,67%, sensitivitas 93,51%, spesifisitas 72,86%, presisi 79,12%.	<i>Hyperparameter</i> disetel secara manual.
10.	Hnin dan Jeenanunta (2019), <i>Bayesian optimization in a support vector regression model for short-term electricity load forecasting</i> [37]	Menyetel <i>hyperparameter</i> SVR menggunakan <i>Bayesian optimization</i> dibandingkan dengan GA dan PSO untuk ramalan beban listrik jangka pendek.	<i>Bayesian optimization</i> menghasilkan kinerja lebih unggul dari GA dan PSO.	Metode menggunakan SVR dan <i>dataset</i> berbeda.

No.	Penulis, Tahun, Judul	Metode	Hasil	Perbedaan
11.	Gupta et. Al (2019), <i>HeartCare: IoT based heart disease prediction system</i> [14]	Klasifikasi penyakit jantung menggunakan <i>K-Nearest Neighbor (KNN), Random Forest, Decision Tree, Support Vector Machine, dan Naive Bayes</i> . Seleksi fitur menggunakan <i>correlation matrix</i> .	Model K-NN dengan mendapatkan hasil terbaik dengan akurasi 88,52% dan sensitivitas 91,17%.	Metode klasifikasi berbeda dan masih menggunakan seleksi fitur manual.
12.	Latha dan Jeeva (2019), <i>Improving the accuracy of prediction of heart disease risk based on ensemble classification techniques</i> [17]	Memprediksi penyakit jantung menggunakan <i>ensemble bagging, boosting, stacking, dan majority vote</i> . Pemilihan fitur menggunakan <i>brute force</i> .	Metode <i>ensemble majority vote</i> dengan <i>Naive Bayes, Bayes Net, Random Forest, dan Multilayer Perceptron</i> mendapatkan hasil terbaik dengan akurasi 85,48%.	Metode klasifikasi berbeda, yaitu <i>ensemble</i> dan masih menggunakan seleksi fitur manual.
13.	Ed-daoudy dan Maalmi (2019), <i>Performance evaluation of machine learning based big data processing framework for prediction of heart disease</i> [18]	Prediksi penyakit jantung menggunakan <i>SVM, Decision Tree, Random Forest, dan Logistic Regression</i>	<i>Random forest</i> memiliki akurasi tertinggi yaitu 87,50%, dengan sensitivitas dan spesifisitas sebesar 86,67% dan 88,37%.	Metode klasifikasi berbeda.
14.	Mohan et. al (2019), <i>Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques</i> [20]	Prediksi penyakit jantung menggunakan <i>Hybrid Random Forest with Linear Model (HRFLM)</i> dan seleksi fitur <i>decision tree entropy</i> .	Model menghasilkan akurasi 88,47%, sensitivitas 92,8%, dan spesifisitas 82,6%.	Metode klasifikasi berbeda, yaitu menggunakan metode <i>hybrid</i> dan masih menggunakan seleksi fitur manual.
15.	Ashraf et. al (2019), <i>Improved Heart Disease Prediction Using Deep Neural Network</i> [31]	Klasifikasi <i>multiclass</i> untuk penyakit jantung menggunakan <i>deep neural network (DNN)</i> .	Model menghasilkan akurasi 87,64%.	Klasifikasi data <i>multiclass</i> , <i>hyperparameter</i> disetel secara manual.

No.	Penulis, Tahun, Judul	Metode	Hasil	Perbedaan
16.	Shekar dan Dagnev (2019), <i>Grid Search-Based Hyperparameter Tuning and Classification of Microarray Cancer Data</i> [32]	Klasifikasi data kanker <i>microarray</i> menggunakan <i>random forest</i> dan penyetelan <i>hyperparameter grid search</i>	Metode yang diusulkan mampu menghasilkan akurasi klasifikasi hingga 100%.	Metode klasifikasi berbeda, penyetelan <i>hyperparameter</i> menggunakan <i>grid search</i> , dan <i>dataset</i> berbeda.
17.	Badriyah et. al (2019), <i>Improving stroke diagnosis accuracy using hyperparameter optimized deep learning</i> [40]	Deteksi penyakit stroke menggunakan <i>Multilayer Perceptron</i> (MLP) dengan penyetelan <i>hyperparameter random search</i> dan <i>Bayesian optimization</i> . Seleksi fitur menggunakan <i>genetic algorithm</i> .	MLP dengan <i>random search</i> mendapatkan hasil akurasi lebih baik, sedangkan dalam hal waktu komputasi, <i>Bayesian optimization</i> lebih unggul.	Metode klasifikasi berbeda, masih menggunakan seleksi fitur manual, dan <i>dataset</i> berupa citra CT scan.
18.	Wu et. al (2019), <i>Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization</i> [36]	Penyetelan <i>hyperparameter</i> menggunakan <i>Bayesian optimization</i> untuk <i>Random forest</i> , <i>Convolutional Neural Network</i> (CNN), <i>Recurrent Neural Network</i> (RNN), dan <i>Multi-Grained Cascade Forest</i> pada <i>dataset</i> UCI, MNIST, dan CIFAR-10.	<i>Bayesian optimization</i> dapat menghasilkan akurasi tinggi dengan waktu penyetelan yang singkat.	Metode klasifikasi dan <i>dataset</i> berbeda.
19.	Assegie (2020), <i>An optimized K-Nearest Neighbor based breast cancer detection</i> [39]	Deteksi kanker payudara menggunakan <i>K-Nearest Neighbor</i> (K-NN) dengan penyetelan <i>hyperparameter grid search</i> dibandingkan dengan penyetelan manual.	Model K-NN dengan <i>grid search</i> dapat meningkatkan akurasi sebesar 94,35%.	Metode klasifikasi berbeda, penyetelan <i>hyperparameter</i> menggunakan <i>grid search</i> , dan <i>dataset</i> berbeda.

No.	Penulis, Tahun, Judul	Metode	Hasil	Perbedaan
20.	Gao dan Ding (2020), <i>Disease prediction via Bayesian hyperparameter optimization and ensemble learning</i> [38]	Prediksi penyakit kanker payudara dan jantung menggunakan <i>ensemble Extreme Gradient Boosting (XGBoost)</i> dengan penyetelan <i>hyperparameter Bayesian optimization, random search, dan grid search</i> . Seleksi fitur menggunakan <i>correlation analysis</i> dan <i>recursive feature elimination (RFE)</i> .	XGBoost dengan <i>Bayesian optimization</i> mendapatkan hasil terbaik dengan akurasi 94,74% dan sensitivitas 93,69% pada <i>dataset</i> penyakit kanker payudara, serta akurasi 73,50% dan sensitivitas 69,54% pada <i>dataset</i> penyakit jantung.	Metode klasifikasi berbeda, masih menggunakan seleksi fitur manual, dan <i>dataset</i> penyakit jantung terdiri atas 11 fitur dan 1 <i>class</i> target.

#### 1.4 Tujuan Penelitian

Tujuan penelitian ini adalah mengembangkan model *deep neural network* dengan penyetelan *hyperparameter* yang efisien menggunakan *Bayesian optimization* untuk mendapatkan akurasi kinerja yang baik dalam mendeteksi penyakit jantung.

#### 1.5 Manfaat Penelitian

Penelitian ini dapat berkontribusi dalam meningkatkan akurasi metode deteksi penyakit jantung dengan menerapkan *deep neural network* dengan penyetelan *hyperparameter* menggunakan *Bayesian optimization*. Selain itu, penelitian ini diharapkan dapat menjadi rujukan dalam merancang sistem berbantuan komputer untuk membantu ahli medis dalam mendeteksi penyakit jantung, sehingga perawatan pasien pengidap penyakit jantung dapat dilakukan lebih dini.

## **BAB II**

### **TINJAUAN PUSTAKA DAN LANDASAN TEORI**

Pada bab ini dibahas mengenai tinjauan pustaka dan landasan teori yang digunakan. Pada tinjauan pustaka, dipaparkan mengenai penelitian-penelitian terkait metode berbantuan komputer untuk klasifikasi penyakit jantung dan teknik penyetulan *hyperparameter* otomatis untuk mengoptimalkan metode pembelajaran. Landasan teori berisi mengenai penjelasan metode yang digunakan.

#### **2.1 Tinjauan Pustaka**

##### **2.1.1 Metode Klasifikasi Penyakit Jantung**

Penelitian-penelitian mengenai pengembangan metode berbantuan komputer untuk mendeteksi penyakit jantung telah banyak dilakukan sebelumnya. Metode-metode tersebut dikategorikan mejadi beberapa pendekatan, yaitu *machine learning tradisional*, metode *ensemble*, *hybrid*, dan *deep learning*. Pada tiap pendekatan, dijelaskan metode-metode yang pernah dilakukan oleh peneliti, kinerja akurasi yang diraih, kelebihan dan kekurangan tiap metode secara singkat sebagai berikut.

Ada beberapa penelitian yang menerapkan *machine learning* tradisional dalam mendeteksi penyakit jantung [11] [12] [13] [14]. Teknik *machine learning* tradisional mempelajari data menggunakan algoritme tertentu untuk membuat prediksi. Metode ini dapat diimplementasikan untuk mendeteksi penyakit jantung dengan baik. Namun, metode ini umumnya perlu melakukan rekayasa fitur sebelum melakukan klasifikasi.

Maheswari dan Jasmine [11] mengembangkan prediksi penyakit jantung menggunakan *neural network*. Menggunakan *dataset* penyakit jantung *Cleveland*, data dibagi menjadi *dataset training* dan *testing*. Model ini menggunakan *logistic regression* untuk mendapatkan fitur signifikan prediksi penyakit jantung dan mengimplementasikan *neural network*. *Neural network* terdiri atas *input layer*, *hidden layer*, dan *output layer* dengan jumlah node tiap *layer*-nya secara berturut-

turut adalah 12, 10, dan 2. Model tersebut menghasilkan akurasi, sensitivitas, dan spesifisitas sebesar 84%, 91,14%, 77,5%. Metode ini memiliki kelebihan dalam mendeteksi semua kemungkinan interaksi antar variabel, dapat menoleransi kesalahan dan menemukan hubungan kompleks antar variabel dependen dan independen. Akan tetapi metode ini cenderung *over-fitting* dan mahal secara komputasi.

El-Sayed [12] mengusulkan sistem untuk mendeteksi penyakit jantung menggunakan *Genetic Algorithm* (GA) dan *Linear Discriminant Analysis* (LDA). Sistem yang diusulkan menggunakan teknik penyaringan atribut *genetic algorithm* untuk mengurangi jumlah atribut. Sistem yang diusulkan dievaluasi pada *dataset* penyakit jantung *Cleveland* dan menghasilkan akurasi 89,07% untuk klasifikasi biner dan 67,22% untuk *multiclass*. LDA memiliki batasan keputusan yang linear, sehingga mudah diimplementasikan dan klasifikasinya kuat. Namun, batasan keputusan linear tidak cukup memisahkan kelas, sehingga dibutuhkan batasan yang lebih umum.

Vijayashree dan Sultana [13] mengusulkan fungsi *Particle Swarm Optimization* (PSO) dengan pemilihan fitur *Support Vector Machine* (SVM) dan pengklasifikasi SVM untuk mengklasifikasikan penyakit jantung. Klasifikasi SVM menggunakan fitur terpilih yang dihasilkan oleh seleksi fitur PSO-SVM yang diusulkan pada *dataset* penyakit jantung *Cleveland* menghasilkan akurasi 88,22%. SVM [13] adalah teknik yang stabil dimana perubahan kecil pada data tidak terlalu mempengaruhi *hyperplane*. SVM juga memberikan fleksibilitas dengan menggunakan konsep kernel dan memiliki generalisasi yang baik. Namun, SVM tidak cocok untuk *dataset* yang besar, kinerja menurun bila ada banyak *noise*, dan sulit untuk diinterpretasikan.

Gupta et al. [14] mengembangkan model berbasis *machine learning* untuk mendeteksi penyakit jantung. Pada *dataset* penyakit jantung dari UCI, pemilihan fitur dilakukan menggunakan *correlation matrix*. Kemudian data dibagi menjadi *dataset training* dan *dataset testing*. *K-Nearest Neighbor* (KNN) mengungguli algoritme lainnya dengan akurasi 88,52% dan sensitivitas 91,17%. Metode ini

sederhana dan tahan terhadap *noise*. Namun, kinerja metode menurun pada *dataset* yang besar dan metode ini membutuhkan memori yang besar.

Beberapa metode *ensemble* juga telah diterapkan dalam mendeteksi penyakit jantung [15] [16] [17] [18]. *Ensemble* adalah agregasi prediksi dari beberapa pengklasifikasi [35]. Hasil agregat dari beberapa pengklasifikasi lebih sedikit *noisy* daripada pengklasifikasi individual yang membuatnya lebih stabil dan kuat. Metode *ensemble* menggabungkan beberapa algoritme pembelajaran untuk meningkatkan akurasi algoritme yang lemah. Namun, metode ini mahal secara komputasi dan algoritme menjadi sangat kompleks yang mengakibatkan kurangnya interpretabilitas.

Pouriyeh et al. [15] membandingkan dan menggunakan teknik *ensemble* untuk prediksi penyakit jantung. Metode *ensemble* menggunakan *10-fold cross-validation* pada *dataset* penyakit jantung *Cleveland*. Hasil percobaan menunjukkan bahwa *Support Vector Machine* (SVM) menggunakan teknik *boosting* mendapatkan hasil terbaik dengan akurasi 84,81%. *Ensemble boosting* mempunyai kelebihan yaitu dapat mengurangi kesalahan bias. Namun, metode ini cenderung *over-fitting* dan sensitif terhadap *outliers*.

Pawlovsky [16] mengusulkan *ensemble* berdasarkan jarak untuk metode *K-Nearest Neighbor* (K-NN) untuk prediksi penyakit jantung. *Ensemble* diimplementasikan dengan dua konfigurasi, yaitu tiga jarak dan lima jarak. Kemudian, mereka ditambahkan ke versi bobot berdasarkan akurasi rata-rata oleh setiap jarak ketika digunakan dalam metode K-NN. Metode yang diusulkan memberikan akurasi rata-rata 85% untuk setiap konfigurasi yang diuji pada *dataset* penyakit jantung *Cleveland*. Metode ini mudah diterapkan, tetapi dibutuhkan evaluasi untuk konfigurasi yang optimal untuk klasifikasi.

Latha dan Jeeva [17] mengusulkan metode *ensemble* dengan pemilihan fitur untuk meningkatkan akurasi prediksi penyakit jantung. Metode dievaluasi pada *dataset* penyakit jantung *Cleveland*. Metode *ensemble majority vote* dengan *Naive Bayes*, *Bayes Net*, *Random Forest*, dan *Multilayer Perceptron* mendapatkan hasil terbaik dengan akurasi 85,48%. *Ensemble majority vote* dapat meningkatkan akurasi pengklasifikasi yang lemah, tetapi mahal secara komputasi.

Ed-daoudy dan Maalmi [18] mengevaluasi teknik *machine learning* untuk prediksi penyakit jantung. Model *machine learning* diuji pada *dataset* penyakit jantung *Cleveland* dalam rasio 70:30. Hasil percobaan menunjukkan bahwa *random forest* memiliki akurasi tertinggi yaitu 87,50%, dengan sensitivitas dan spesifisitas sebesar 86,67% dan 88,37%. *Random forest* adalah metode *ensemble* yang membangun hutan dari beberapa *decision tree*. Keuntungan dari metode ini termasuk akurasi kinerja yang baik dan kemampuan untuk menghindari masalah *over-fitting* yang berlebihan. Namun, *random forest* lebih lambat dari pengklasifikasi lainnya, seperti SVM, *decision tree*, dan *logistic regression*.

Teknik *hybrid* juga telah diterapkan dalam deteksi penyakit jantung [19] [20]. Teknik *hybrid* mengintegrasikan beberapa metode *machine learning* untuk menghasilkan model prediktif. Teknik *hybrid* dapat mengurangi kekurangan pada model individu dan memanfaatkan metode generalisasinya. Menggunakan teknik *hybrid* untuk memprediksi penyakit jantung dapat meningkatkan kinerja prediksi, tetapi tergantung pada kombinasi teknik yang digunakan.

Normawati dan Winarti [19] mengusulkan metode untuk mendeteksi penyakit jantung koroner dengan menggunakan *Variable Precision Rough Set* (VPRS) sebagai metode pemilihan fitur dan klasifikasi menggunakan kombinasi VPRS dan *Repeated Incremental Pruning Error Reduction* (RIPPER). Metode pemilihan fitur VPRS dan kombinasi pengklasifikasian VPRS dan RIPPER dievaluasi menggunakan *dataset* penyakit jantung *Cleveland* yang menghasilkan akurasi, sensitivitas, dan spesifisitas sebesar 88,89%, 100%, 77,08%. Metode ini dapat meningkatkan kualitas penalaran dan meningkatkan kinerja klasifikasi. Akan tetapi, metode ini juga meningkatkan kompleksitas algoritme.

Mohan et al. [20] mengusulkan *Hybrid Random Forest with Linear Model* (HRFLM) untuk prediksi penyakit jantung. Pendekatan HRFLM menggabungkan karakteristik *random forest* dan metode linier. HRFLM diimplementasikan pada *dataset* penyakit jantung *Cleveland* yang menghasilkan akurasi, sensitivitas, dan spesifisitas sebesar 88,47%, 92,8%, dan 82,6%. Metode ini meningkatkan akurasi kinerja dan mengurangi kesalahan klasifikasi. Namun, metode ini juga meningkatkan kompleksitas komputasi.

Pendekatan *deep learning* telah diimplementasikan dalam mendeteksi penyakit jantung [30] [31]. *Deep learning* menghindari kebutuhan akan proses rekayasa fitur. Metode ini berkinerja sangat baik *dataset* yang besar. Namun, *deep learning* membutuhkan banyak contoh untuk mempelajari suatu konsep.

Miao dan Miao [30] mengembangkan *deep neural network* (DNN) untuk memprediksi penyakit jantung. Model DNN yang diusulkan terdiri atas 4 lapisan, yaitu *input layer*, 2 *hidden layers*, dan *output layer*. *Input layer* menerima 28 input, masing-masing *hidden layers* memiliki 105 dan 42 neuron, dan *output layer* memiliki 1 neuron. Model DNN digunakan untuk melakukan klasifikasi biner untuk penyakit jantung koroner. Model dievaluasi dalam *dataset testing* yang menghasilkan akurasi, sensitivitas, spesifisitas, dan presisi sebesar 83,67%, 93,51%, 72,86%, dan 79,12%. DNN mampu mengidentifikasi fitur secara langsung tanpa adanya rekayasa fitur manual. Namun, pada penelitian ini *hyperparameter* DNN ditentukan dengan cara manual dan akurasi kinerja yang dihasilkan oleh model DNN tersebut masih rendah.

Ashraf et al. [31] mengembangkan model *deep neural network* untuk deteksi penyakit jantung. Model ini terdiri atas 4 lapisan, yaitu *input layer*, 2 *hidden layers*, dan *output layer*. *Input layer* menerima 13 input, masing-masing *hidden layers* memiliki 6 dan 4 neuron, dan *output layer* memiliki 1 neuron. Model ini digunakan untuk melakukan klasifikasi *multiclass* untuk penyakit jantung dan mencapai akurasi 87,64%. DNN tidak membutuhkan rekayasa fitur manual. Akan tetapi, penentuan *hyperparameter* pada DNN masih dilakukan secara manual.

### 2.1.2 Teknik Penyetelan *Hyperparameter* Otomatis

Penelitian-penelitian mengenai teknik penyetelan *hyperparameter* otomatis untuk mengoptimalkan metode pembelajaran telah dilakukan sebelumnya. Shekar dan Dagnev [32] mengusulkan penyetelan *hyperparameter grid search* untuk mengoptimalkan *hyperparameter random forest* untuk klasifikasi data kanker *microarray*. *Grid search* mencari *hyperparameter* optimal dari nilai *hyperparameter* yang ditentukan berdasarkan *cross-validation*. *Hyperparameter*

yang dioptimalkan meliputi jumlah *tree*, jumlah fitur maksimum dalam *node*, kedalaman *tree* pada setiap *decision trees*, dan kriteria dalam *splitting*. Metode yang diusulkan mampu menghasilkan akurasi klasifikasi hingga 100%.

Assegie [39] mengusulkan pengoptimalan *K-Nearest Neighbor* (K-NN) menggunakan *grid search* untuk deteksi kanker payudara. *Grid search* digunakan untuk menemukan nilai *nearest neighbours* (*k*) terbaik yang dapat menghasilkan akurasi optimal dalam mendeteksi kanker payudara. Hasil penelitian menunjukkan bahwa *grid search* berpengaruh signifikan terhadap kinerja model K-NN. Kinerja K-NN dengan penyetelan *hyperparameter* menggunakan *grid search* diuji secara eksperimental menggunakan *dataset* kanker payudara *Wisconsin*. Teknik ini menghasilkan akurasi model sebesar 94.35% pada *dataset testing*.

Mantovani et al. [34] melakukan penelitian mengenai efektifitas *random search* dalam penyetelan *hyperparameter* SVM. *Random search* digunakan untuk mencari nilai *hyperparameter cost* (*C*) dan *gamma* ( $\gamma$ ). Kinerja SVM dengan *random search* diuji pada 70 *dataset* dan dibandingkan dengan teknik *grid search* dan tiga teknik meta-heuristik yang umum digunakan menyetel *hyperparameter* SVM yaitu, *Genetic Algorithm* (GA), *Particle Swarm Optimization* (PSO), dan *Estimation of Distribution Algorithms* (EDA). *Random search* mampu menghasilkan akurasi kinerja yang baik dengan biaya komputasi yang lebih rendah.

Badriyah et al. [40] melakukan penelitian untuk mengoptimalkan *hyperparameter Multilayer Perceptron* (MLP) untuk deteksi stroke. *Hyperparameter* yang dioptimalkan meliputi jumlah *hidden layer*, jumlah *node*, *epoch*, *activation function*, *learning rate*, *batch*, dan *dropout rate*. Penyetelan *hyperparameter* MLP menggunakan *random search* dan *Bayesian optimization*. Hasil pengujian menunjukkan bahwa *random search* mendapatkan hasil akurasi lebih baik, sedangkan dalam hal waktu penyetelan, *Bayesian optimization* lebih unggul.

Padierna et. al [35] mengusulkan *Estimation of Distribution Algorithms* (EDA) untuk menyetel *hyperparameter* SVM menggunakan 11 *dataset* dari UCI. EDA memilih *hyperparameter* optimal dengan mengeksplorasi ruang solusi dengan

secara berulang membangun dan mengambil sampel model probabilistik eksplisit dari solusi kandidat yang memandu pencarian. *Hyperparameter* SVM disetel menggunakan dua metode *Estimation of Distribution Algorithms* (EDA), yaitu *Univariate Marginal Distribution Algorithm* (UMDA) dan *the Boltzmann Univariate Marginal Distribution Algorithm* (BUMDA), dibandingkan dengan *random search*. Metode EDA khususnya BUMDA mampu mencapai hasil yang lebih baik daripada *random search* dengan tidak ada lagi kompleksitas algoritmik tambahan daripada pengurutan dan penghitungan statistik sederhana. Namun, EDA memiliki beberapa kelemahan yaitu hilangnya keragaman, kurangnya penggunaan informasi lokal solusi, dan terjebak dalam optima lokal.

Wu et al. [36] mengusulkan penyetelan *hyperparameter* menggunakan *Bayesian optimization* untuk model pembelajaran. *Bayesian optimization* menggabungkan menggabungkan informasi sebelumnya tentang fungsi objektif yang tidak diketahui dengan informasi sampel, untuk mendapatkan informasi posterior dari distribusi fungsi dengan menggunakan rumus *Bayesian*. Hasil eksperimen menunjukkan bahwa *Bayesian optimization* dapat menghasilkan akurasi yang tinggi untuk beberapa model pembelajaran, seperti *random forest*, *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN), dan *Multi-Grained Cascade Forest* dengan waktu penyetelan yang singkat.

Hnin dan Jeenanunta [37] membandingkan *Bayesian optimization* dengan *Genetic Algorithm* (GA) dan *Particle Swarm Optimization* (PSO) untuk menyetel *hyperparameter* pada *Support Vector Regression* (SVR) untuk ramalan beban listrik jangka pendek. Berdasarkan hasil MAPE, SVR dengan *Bayesian optimization* menghasilkan kinerja lebih unggul daripada GA dan PSO. GA dan PSO mudah terjebak pada minimum lokal, sedangkan *Bayesian Optimization* mampu melampaui minimum lokal.

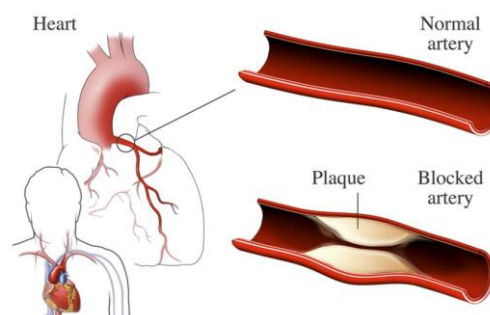
Gao dan Ding [38] mengusulkan *Bayesian optimization* untuk mengoptimalkan pembelajaran *ensemble* dalam prediksi penyakit kanker payudara dan jantung. *Bayesian optimization* digunakan untuk menyetel *hyperparameter* *Extreme Gradient Boosting* (XGBoost). Kinerja *Bayesian optimization*

dibandingkan dengan *grid search* dan *random search*. Pada klasifikasi penyakit jantung, dilakukan proses seleksi fitur menggunakan *recursive feature elimination* (RFE) sebelum klasifikasi. Sedangkan, pada klasifikasi penyakit kanker payudara menggunakan *correlation analysis* untuk menyeleksi fitur. XGBoost dengan *Bayesian optimization* menghasilkan akurasi 94,74% dan sensitivitas 93,69% pada *dataset* penyakit kanker payudara, serta akurasi 73,50% dan sensitivitas 69,54% pada *dataset* penyakit jantung. Hasil pengujian menunjukkan bahwa *Bayesian optimization* memiliki kinerja yang lebih stabil dalam menyetel *hyperparameter* XGBoost.

## 2.2 Landasan Teori

### 2.2.1 Penyakit Jantung

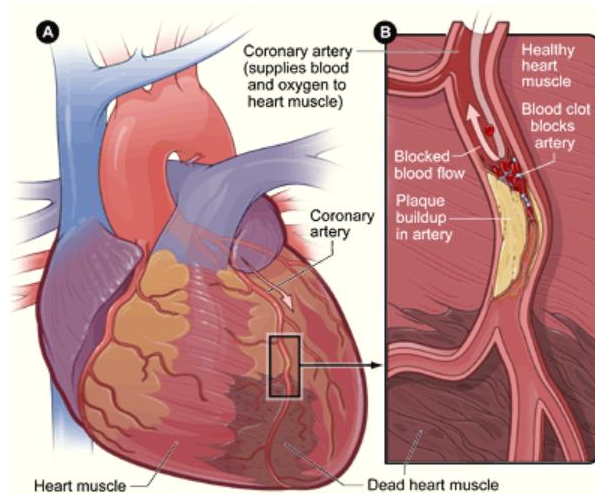
Penyakit Jantung Koroner (PJK) atau *Coronary Heart Disease* (CHD), disebut juga *Coronary Artery Disease* (CAD) atau *Ischemic Heart Disease* (IHD), secara singkat disebut penyakit jantung adalah permasalahan jantung yang disebabkan oleh penyempitan pembuluh darah jantung yang memasok darah ke otot jantung [3]. Penyempitan pembuluh darah jantung paling sering disebabkan oleh penumpukan plak atau aterosklerosis. Gambar 2.1 menunjukkan arteri koroner normal dengan aliran darah normal dan arteri koroner tersumbat yang menyempit oleh plak. Penumpukan plak membatasi aliran darah kaya oksigen melalui arteri.



Gambar 2.1 Arteri normal versus arteri yang tersumbat [41]

Jika aliran darah ke otot jantung terhambat sepenuhnya akan menyebabkan sel-sel otot jantung mati yang disebut serangan jantung atau *myocardial infarction* (MI) [3]. Kondisi jantung dan arteri akibat serangan jantung digambarkan seperti

Gambar 2.2. Gambar A merupakan gambaran jantung dan arteri koroner yang menunjukkan kerusakan (otot jantung mati) akibat serangan jantung. Gambar B adalah penampang arteri koroner dengan penumpukan plak dan bekuan darah.



Gambar 2.2 Jantung dengan kerusakan otot dan arteri tersumbat [42]

Kebanyakan orang dengan PJK dini (kurang dari 50 persen penyempitan) tidak mengalami gejala atau keterbatasan aliran darah. Namun, seiring perkembangan aterosklerosis, jika tidak ditangani, gejala dapat terjadi. Hal ini paling mungkin terjadi selama latihan atau stres emosional, ketika kebutuhan oksigen yang dibawa oleh darah meningkat.

Banyak faktor risiko penyebab penyakit jantung yang dipengaruhi oleh gaya hidup. Faktor risiko penyakit jantung di antaranya tekanan darah tinggi, kadar kolesterol darah tinggi, merokok, diabetes, obesitas, kurangnya aktivitas fisik, diet tidak sehat, dan stres [43]. Adapula faktor risiko yang tidak bisa dikontrol seperti, usia, jenis kelamin, riwayat keluarga, dan keturunan.

Penyakit jantung koroner dapat didiagnosis dengan beberapa cara. Pasien dengan riwayat serangan jantung atau angina khas menunjukkan diagnosis klinis PJK. Tes diagnostik tambahan juga dapat dilakukan untuk mengonfirmasi PJK di antaranya adalah *Exercise Stress Test (EST)*, *pharmacologic (nuclear and echocardiography) stress tests*, dan *coronary CT angiography*. EST biasanya dilakukan menggunakan treadmill atau ergometer sepeda sambil memantau

elektrokardiogram (EKG), ambang ventilasi, tekanan darah, detak jantung, konsumsi oksigen, laporan pasien secara umum, dan penampilan fisik termasuk nyeri dada dan dispne [44]. *Pharmacologic stress testing* adalah tes alternatif untuk mengevaluasi gejala PJK bagi pasien yang tidak dapat berolahraga [45]. *Coronary computed tomography (CT) angiography* adalah teknik pencitraan dimana pewarna kontras beryodium disuntikkan melalui vena perifer dan gambar arteri koroner diambil menggunakan sistem CT [3].

### 2.2.2 *Data Mining*

*Data mining* atau penambangan data adalah proses menemukan pola atau pengetahuan yang berguna dari kumpulan data yang besar [9]. *Data mining* pada umumnya mempunyai dua tujuan, prediksi dan deskripsi [46]. Prediksi melibatkan penggunaan beberapa variabel atau bidang dalam kumpulan data untuk memprediksi nilai variabel lain yang tidak diketahui atau yang akan datang. Deskripsi berfokus pada menemukan pola yang menggambarkan data yang dapat diinterpretasikan oleh manusia. Oleh karena itu, aktivitas *data mining* dibagi menjadi dua kategori, *data mining* prediktif dan *data mining* deskriptif. *Data mining* prediktif menghasilkan model sistem yang dijelaskan oleh kumpulan data yang diberikan. Beberapa tugas dari *data mining* prediktif di antaranya adalah klasifikasi, regresi, dan *deviation detection*. *Data mining* deskriptif menghasilkan informasi baru dan *non-trivial* berdasarkan kumpulan data yang tersedia. Beberapa tugas dari *data mining* deskriptif di antaranya adalah asosiasi, *clustering*, dan *summarization*.

Klasifikasi merupakan salah satu tugas *data mining* yang sering digunakan dalam dunia kesehatan [47]. Klasifikasi adalah mengklasifikasikan suatu item data ke dalam salah satu dari beberapa kelas yang telah ditentukan [9]. Dalam tugas klasifikasi, pertama-tama model dibangun dan dilatih menggunakan *dataset training* yang telah ditentukan dengan label kelas kemudian model tersebut digunakan untuk memprediksi label kelas untuk *dataset testing* guna memperkirakan keakuratan model pengklasifikasi [48]. Ada dua jenis tugas klasifikasi, yaitu klasifikasi biner dan klasifikasi *multiclass*. Contoh klasifikasi

biner dalam memprediksi penyakit adalah pasien dapat di kategorikan menjadi pasien sehat dan pasien yang mengidap penyakit.

### 2.2.3 *Deep Learning*

Dalam dekade terakhir, penggunaan *deep learning* dalam hal biomedis dan perawatan kesehatan telah menerima perhatian yang luar biasa [21]. Ukuran data medis terlalu besar untuk analisis komprehensif dengan alat analitik yang tersedia untuk memaksimalkan pengetahuan yang tersedia dalam data besar. Teknik *machine learning* tradisional memiliki kapasitas terbatas untuk memanfaatkan data besar dan dalam banyak kasus, solusinya menjadi kompleks. *Deep learning* memberikan solusi prospektif untuk tantangan ini. Keuntungan utama *deep learning* adalah bahwa kinerja arsitektur *deep learning* yang besar meningkat dengan bertambahnya ukuran data yang ada.

*Deep learning* adalah cabang dari *machine learning* yang menggunakan konsep *neural network* yang terinspirasi oleh struktur dan fungsi otak manusia [49]. *Deep learning* menggunakan beberapa lapisan unit pemrosesan *non-linear* untuk ekstraksi dan transformasi fitur. Lapisan bawah yang dekat dengan masukan data mempelajari fitur sederhana, sedangkan lapisan yang lebih tinggi mempelajari fitur yang lebih kompleks yang diturunkan dari fitur lapisan bawah. Arsitektur *deep learning* membentuk representasi fitur hierarkis dan kuat yang cocok untuk menganalisis dan mengekstraksi pengetahuan yang berguna dari data dalam jumlah besar dan data yang dikumpulkan dari berbagai sumber [50].

Beberapa arsitektur *deep learning* yang paling umum di antaranya adalah *deep neural network (DNN)*, *recurrent neural network (RNN)*, dan *convolutional neural network (CNN)* [24]. *Deep neural network (DNN)* adalah *neural network* dengan banyak *hidden layer* di antara *input layer* dan *output layer* dimana hubungan antar lapisan bersifat satu arah. DNN digunakan untuk menyelesaikan masalah yang menggunakan data terstruktur yang tersimpan dalam format tabel. *Recurrent neural network (RNN)* adalah jaringan dengan koneksi yang membentuk siklus terarah. RNN digunakan untuk mengatasi masalah pembelajaran yang

melibatkan data *sequence* seperti pengenalan tulisan tangan, pengenalan ucapan, dan mesin penerjemah. *Convolutional neural network (CNN)* dirancang untuk secara otomatis dan adaptif mempelajari hierarki spasial fitur melalui algoritme *backpropagation* dengan menggunakan beberapa blok penyusun seperti *convolution layer*, *pooling layer*, and *fully connected layer*. CNN umumnya diterapkan untuk menganalisis citra visual.

Teknik *deep learning* dapat memberikan hasil yang lebih baik, mengurangi tingkat kesalahan klasifikasi, dan lebih kuat terhadap *noise* daripada pendekatan lainnya [23]. *Deep learning* juga menghindari adanya rekayasa fitur dengan mengubah data menjadi representasi langsung yang ringkas dari komponen utama dan menghilangkan redundansi dalam representasi melalui struktur berlapis yang diturunkan [51]. Keuntungan lain dari *deep learning* adalah representasi invarian terhadap perubahan lokal dalam data masukan dan dapat menganalisis dan mempelajari data yang besar [52].

#### **2.2.4 Deep Neural Network**

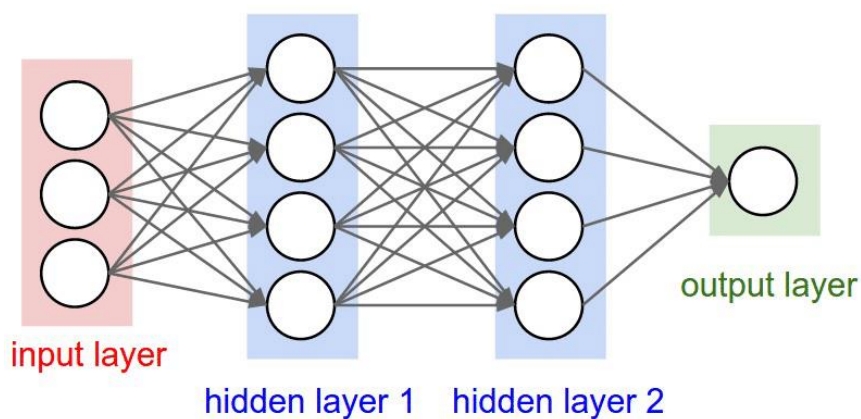
*Deep neural network* menggunakan arsitektur *neural network* yang lebih dalam yang dapat mewakili fungsi dengan kompleksitas yang lebih tinggi [25]. Perbedaan *neural network* dan *deep neural network* ditentukan pada kedalaman modelnya, *neural network* dengan dua atau lebih *hidden layer* disebut *deep neural network* karena jaringan telah menjadi kompleks dengan lebih dari satu *hidden layer* [26]. Keuntungan *deep neural network* dibandingkan dengan *neural network* adalah arsitekturnya yang lebih dalam dapat meningkatkan ketepatan dan generalisasi setelah mempelajari contoh baru [27]. Keuntungan *deep neural network* dibandingkan dengan teknik *machine learning* tradisional adalah membutuhkan lebih sedikit pengetahuan domain untuk menyelesaikan masalah [28]. *Deep neural network* dapat mempelajari fitur langsung dari data tanpa memerlukan rekayasa fitur manual. Penelitian yang menggunakan *deep neural network* di antaranya adalah Miao dan Miao [30] yang mengusulkan *deep neural network* untuk prediksi penyakit jantung dengan arsitektur yang terdiri atas *input*

layer yang menerima 28 input, dua *hidden layers* yang masing-masing memiliki 105 dan 42 neuron, dan *output layer* yang memiliki 1 neuron. Ashraf et. al [31] mengusulkan *deep neural network* untuk prediksi penyakit jantung dengan arsitektur yang terdiri atas *input layer* yang menerima 13 input, dua *hidden layers* yang masing-masing memiliki 4 dan 2 neuron, dan *output layer* yang memiliki 1 neuron. Ayon dan Islam [23] mengusulkan *deep neural network* untuk prediksi diabetes dengan arsitektur yang terdiri atas *input layer* yang menerima 8 input, empat *hidden layer* yang masing-masing memiliki 12,16,16, dan 14 neuron, dan *output layer* yang memiliki 1 neuron.

Arsitektur *deep neural network* dibangun dengan model sekuensial yang terdiri atas *input layer*, *hidden layer*, dan *output layer*. Tiap layer memiliki neuron yang mana setiap neuron menghitung jumlah bobot inputnya dan menambahkan bias. Neuron didefinisikan dengan fungsi (2-1).

$$y = Wx + b \quad (2-1)$$

Variabel  $W$  adalah bobot,  $x$  adalah masukan, dan  $b$  adalah suku bias. Bobot adalah kekuatan koneksi antara input dan output. Bentuk matriks  $W$  ditentukan oleh banyaknya unit pada *input layer* dan *output layer*. Nilai  $y$  dapat berupa apa saja dari  $-\infty$  hingga  $+\infty$ . Untuk memutuskan apakah neuron akan diaktifkan atau tidak, digunakan *activation function*. Arsitektur *deep neural network* dapat dilihat pada Gambar 2.3.



Gambar 2.3 Arsitektur *deep neural network* [23]

Lapisan-lapisan pada arsitektur *deep neural network* dijelaskan sebagai berikut:

### 1. *Input layer*

*Input layer* adalah memberikan informasi dari dunia luar ke jaringan. Pada lapisan ini, tidak ada komputasi yang dilakukan, *input nodes* hanya meneruskan informasi ke *hidden nodes*.

### 2. *Hidden layer*

*Hidden layer* adalah lapisan perantara yang melakukan komputasi dan mentransfer informasi dari *input layer* ke *output layer*. *Deep neural network* memiliki dua atau lebih *hidden layer*. Setiap *hidden layer* memiliki neuron-neuron atau *nodes* yang disebut *hidden nodes* atau *hidden units*.

Pada *hidden layer*, digunakan *activation function* ReLU (*Rectified Linear Unit*). ReLU memiliki kelebihan dalam hal kinerja dan generalisasi yang baik [53]. ReLU menjamin komputasi yang lebih cepat karena tidak menghitung eksponensial dan divisi, dengan kecepatan komputasi secara keseluruhan ditingkatkan. ReLU mewakili fungsi yang hampir linier dan mempertahankan properti model linier yang membuatnya mudah untuk dioptimalkan dengan metode penurunan gradien. ReLU menghasilkan *output* maksimum antara 0 dan x. Jika x negatif, *output*-nya 0 dan jika x positif, *output*-nya adalah x. ReLU didefinisikan dengan persamaan (2-2).

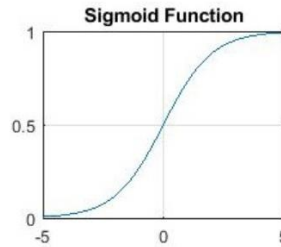
$$\text{ReLU}(x) = \max(0, x) \quad (2-2)$$

### 3. *Output layer*

*Output layer* bertanggung jawab atas komputasi dan pemindahan informasi dari jaringan ke dunia luar.

Pada *output layer*, *activation function* yang digunakan adalah *sigmoid*. Fungsi *sigmoid* digunakan untuk memprediksi *output* berbasis probabilitas dan telah berhasil diterapkan dalam masalah klasifikasi biner [53]. Fungsi *sigmoid* memiliki karakteristik kurva berbentuk "S" atau kurva

*sigmoid*, ditunjukkan pada Gambar 2.4.



Gambar 2.4 Kurva *sigmoid* [53]

*Sigmoid* berkisar antara 0 dan 1. *Sigmoid* didefinisikan dengan persamaan (2-3) dimana ketika  $x$  meningkat menuju tak terhingga positif, *output* mendekati 1. Jika  $x$  menurun ke tak terhingga negatif, *output* mendekati 0.

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (2-3)$$

Model *deep neural network* di-*training* menggunakan *loss function* untuk meminimalkan kesalahan antara *output* yang diprediksi dan *output* aktual sehingga mencapai solusi optimal untuk satu sampel pelatihan. Penelitian ini menggunakan *binary cross-entropy* sebagai *loss function*. *Binary cross-entropy* adalah *loss function* yang digunakan dalam klasifikasi biner yang didefinisikan sebagai (2-4), dimana  $y$  adalah *output* aktual dan  $p$  adalah *output* yang diprediksi.

$$L(y, p) = -(y \log(p) + (1 - y) \log(1 - p)) \quad (2-4)$$

*Optimizer* atau metode pengoptimalan digunakan pada saat pelatihan untuk mengubah atribut *neural network* seperti bobot dan kecepatan pembelajaran untuk mengurangi *loss*. Dalam penelitian ini, *optimizer* yang digunakan adalah *Adam* (*Adaptive Moment Estimation*). *Adam optimizer* adalah metode penurunan gradien stokastik yang didasarkan pada estimasi adaptif momen orde pertama dan orde dua. *Adam* menggabungkan keuntungan dari dua metode pengoptimalan, yaitu kemampuan *Adaptive Gradient Algorithm* (AdaGrad) untuk menangani gradien renggang dan kemampuan *Root Mean Square Propagation* (RMSProp) untuk menangani tujuan non-stasioner. *Adam* mudah digunakan, efisien secara komputasi,

membutuhkan sedikit memori, invarian terhadap penskalaan ulang gradien diagonal, dan cocok untuk masalah yang memiliki data yang besar [54]. *Adam* menyimpan perkiraan momen pertama dan kedua dari gradien dan menggunakannya untuk memperbarui parameter. Diberikan fungsi tujuan stokastik  $f(W)$  dengan parameter  $W_t$ , dimana  $t$  mengindeks iterasi pelatihan, momen pertama dan kedua didefinisikan sebagai (2-5) dan (2-6).

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (2-5)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (2-6)$$

Dimana, momentum pertama  $\beta_1 = 0,9$  dan momentum kedua  $\beta_2 = 0,999$ ,  $g_t = \nabla_W f_t(W_{t-1})$  menunjukkan gradien w.r.t  $W$  pada iterasi  $t$ . Dengan inisialisasi  $m_0 = 0$  dan  $v_0 = 0$ , perkiraan  $m_t$  dan  $v_t$  bias menuju nol. Bias ini diatasi dengan menghitung perkiraan momen pertama dan kedua bias yang dikoreksi sebagai (2-7) dan (2-8).

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (2-7)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (2-8)$$

Kemudian, aturan pembaruan terakhir untuk *Adam* akan didefinisikan sebagai (2-9).

$$W_t = W_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \quad (2-9)$$

Dimana,  $\epsilon$  adalah skalar kecil yang digunakan untuk mencegah pembagian dengan nol,  $\epsilon = 10^{-8}$  dan  $\eta$  adalah *learning rate* awal dengan pengaturan *default*  $\eta = 0,001$ . *Learning rate* menentukan ukuran langkah di setiap iterasi sambil bergerak menuju *loss function* minimum.

*Hyperparameter* adalah variabel yang menentukan struktur jaringan dan variabel yang menentukan bagaimana jaringan dilatih. *Hyperparameter* yang

disetel dalam penelitian ini di antaranya:

1. Jumlah *hidden layer* dan *hidden units*

Jumlah *hidden layer* yang digunakan dalam model dapat berpengaruh pada kinerja klasifikasi. *Hidden units* atau *hidden nodes* adalah neuron yang ada pada *hidden layer*. Jumlah *hidden units* yang terlalu banyak dengan dapat menyebabkan *overfitting*. Sedangkan, *hidden units* yang terlalu sedikit dapat menyebabkan *underfitting*.

2. *Dropout*

*Dropout* adalah teknik yang mencegah *over-fitting* dan menyediakan cara untuk menggabungkan secara eksponensial banyak arsitektur *neural network* yang berbeda secara efisien. *Dropout* secara acak menjatuhkan unit dari jaringan selama pelatihan (*training*) untuk mencegah unit beradaptasi secara berlebihan [55]. *Dropout* dapat diterapkan pada salah satu atau semua *hidden layer* di jaringan serta lapisan yang terlihat atau *input layer*. *Dropout rate* dapat bernilai antara 0 dan 1.

3. Ukuran *batch*

Ukuran *batch* menjadi salah satu *hyperparameter* yang disetel sebelum *training* DNN. Ukuran *batch* adalah *hyperparameter* yang mendefinisikan jumlah sampel pelatihan yang digunakan dalam satu iterasi. Menyetel *hyperparameter* ukuran *batch* terlalu tinggi dapat membuat jaringan membutuhkan waktu terlalu lama untuk mencapai konvergensi dimana tidak ada lagi peningkatan akurasi [56]. Namun, jika ukuran *batch* terlalu rendah, akan membuat jaringan terpelantak bolak-balik tanpa mencapai kinerja yang dapat diterima [56]. Jika semua sampel pelatihan digunakan untuk membuat satu *batch*, disebut *batch gradient descent*. Jika *batch* berukuran satu sampel, disebut *stochastic gradient descent*. Jika ukuran *batch* lebih dari satu sampel dan lebih kecil dari ukuran data pelatihan, disebut *mini-batch gradient descent*. Ukuran yang umum untuk *mini-batch gradient descent* adalah 32, 64, 128, dan sebagainya. Menurut Kandel dan Castelli [56] dalam menentukan ukuran *batch* yang optimal, disarankan

mencoba ukuran *batch* dari ukuran kecil terlebih dahulu seperti 32 dan 64.

#### 4. Jumlah *Epoch*

Jumlah *epoch* adalah frekuensi seluruh data pelatihan ditampilkan ke jaringan selama pelatihan. Jumlah *epoch* dapat ditingkatkan hingga akurasi validasi mulai menurun dan akurasi *training* meningkat atau mulai *overfitting*. Di akhir *epoch*, algoritme pembelajaran dapat membandingkan atau meninjau output aktual dari *dataset training* dan mengoptimalkan atau membuat penyesuaian pada parameternya untuk membuat prediksi yang lebih baik di siklus berikutnya.

### 2.2.5 *Hyperparameter*

*Hyperparameter* adalah parameter yang nilainya digunakan untuk mengontrol proses pembelajaran. Nilai *hyperparameter* tidak didapatkan langsung saat proses *training* sehingga harus ditentukan sebelum memulai *training* model. Contoh *hyperparameter* adalah parameter *regularization C* pada SVM, jumlah *nearest neighbours* (k) pada K-NN, jumlah *tree* pada *random forest*, dan jumlah *hidden units* pada *neural network*.

#### 2.2.5.1 **Penyetelan Hyperparameter Manual**

Penyetelan atau pengoptimalan *hyperparameter* adalah pemilihan sekumpulan *hyperparameter* yang optimal untuk algoritme pembelajaran. Penyetelan *hyperparameter* sangat penting karena akan menyebabkan perubahan besar dalam akurasi model. Secara tradisional, *hyperparameter* dapat disetel secara manual. Penyetelan manual atau *manual search* menetapkan nilai *hyperparameter* berdasarkan intuisi atau tebakan. Dengan menggunakan penyetelan manual, model dilatih dengan mencoba beberapa nilai *hyperparameter* secara manual dan dilakukan berkali-kali hingga mendapat skor validasi yang baik. Penyetelan manual mengharuskan pengguna memiliki pengetahuan latar belakang dan pengalaman praktis yang lebih profesional. Dan sulit untuk diterapkan oleh pengguna non-ahli. Proses penyetelan *hyperparameter* secara manual tidak mudah direproduksi. Selain itu, dengan bertambahnya jumlah *hyperparameter* dan jangkauan nilai, hal ini

menjadi sangat sulit untuk dikelola karena manusia tidak pandai menangani data berdimensi tinggi dan mudah salah menafsirkan atau melewatkan tren dan hubungan dalam *hyperparameter*.

#### **2.2.5.2 Penyetelan *Hyperparameter* Otomatis**

Penyetelan *hyperparameter* otomatis menyesuaikan *hyperparameter* model pembelajaran selama proses pelatihan secara otomatis. Prosedur penyetelan *hyperparameter* otomatis melibatkan penentuan ruang pencarian *hyperparameter*. Ruang pencarian *hyperparameter* dapat diartikan secara geometris sebagai volume  $n$ -dimensional, dimana setiap *hyperparameter* mewakili dimensi yang berbeda dan skala dari dimensi tersebut adalah nilai-nilai yang dapat diambil oleh *hyperparameter* tersebut, seperti nilai riil, nilai integer, atau *categorical*. Titik di ruang pencarian adalah vektor dengan nilai spesifik untuk setiap nilai *hyperparameter*. Tujuan dari prosedur penyetelan *hyperparameter* adalah untuk menemukan vektor yang menghasilkan performa model terbaik setelah pembelajaran.

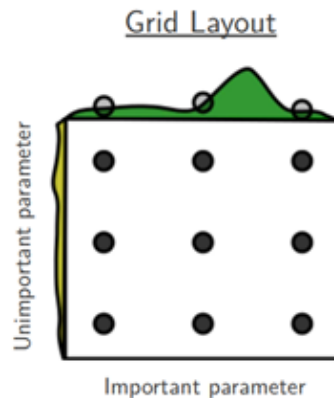
Pemilihan model dengan mengevaluasi berbagai pengaturan *hyperparameter* dapat dilihat sebagai cara untuk menggunakan data berlabel untuk melatih *hyperparameter*. Saat mengevaluasi model yang dihasilkan, penting untuk melakukannya pada sampel *hold-out* yang tidak terlihat selama proses penyetelan *hyperparameter*. Pada umumnya, sebelum melakukan penyetelan *hyperparameter*, data dibagi menjadi satu set pengembangan untuk digunakan dalam proses penyetelan *hyperparameter* dan satu set evaluasi untuk menghitung metrik kinerja. Pembagian data bisa dilakukan dengan menggunakan *train-test split*.

Pada penelitian ini, dilakukan penyetelan *hyperparameter* otomatis meliputi *grid search*, *random search*, dan *Bayesian optimization* yang dijelaskan sebagai berikut:

1. *Grid search*

*Grid search* merupakan metode penyetelan *hyperparameter* yang mudah dipahami dan mampu menghasilkan *hyperparameter* model yang optimal untuk mencapai kinerja prediksi yang akurat [57]. *Grid search*

melakukan pencarian secara menyeluruh melalui *subset* tertentu dari ruang *hyperparameter* algoritme pembelajaran. *Grid search* mengevaluasi kombinasi *hyperparameter* model pada set *training* dengan menggunakan *cross-validation*. Kombinasi *hyperparameter* yang mendapat skor rata-rata *cross-validation* paling tinggi dipilih sebagai *hyperparameter* terbaik.



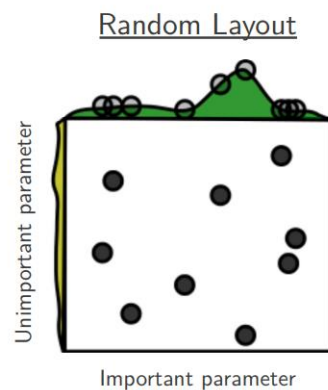
Gambar 2.5 *Grid search* [33]

Gambar 2.5 menggambarkan pencarian menggunakan *grid search*. *Grid search* dari sembilan percobaan untuk mengoptimalkan fungsi  $f(x,y) = g(x) + h(y) \approx g(x)$  dengan dimensi efektif rendah,  $g(x)$  ditampilkan dengan warna hijau, dan  $h(y)$  ditampilkan dengan warna kuning. Dengan *grid search*, dari sembilan percobaan hanya menguji  $g(x)$  di tiga tempat berbeda.

## 2. *Random search*

*Random search* adalah teknik dimana kombinasi acak dari *hyperparameter* digunakan untuk menemukan solusi terbaik untuk model yang dibangun. *Random search* mencoba kombinasi acak dari berbagai nilai. Jumlah kombinasi *hyperparameter* yang dicoba oleh *random search* dapat ditentukan secara eksplisit. Kombinasi *hyperparameter* model pada set *training* dievaluasi dengan menggunakan *cross-validation*. *Random search* lebih efisien daripada *grid search* di ruang berdimensi tinggi karena tidak semua kombinasi *hyperparameter* sama pentingnya untuk disetel [33]. *Random search* membutuhkan waktu yang lebih sedikit untuk menyetel *hyperparameter* dengan jumlah iterasi yang lebih sedikit.

Gambar 2.6 menggambarkan pencarian menggunakan *random search*. *Random search* dari sembilan percobaan untuk mengoptimalkan fungsi  $f(x,y) = g(x) + h(y) \approx g(x)$  dengan dimensi efektif rendah,  $g(x)$  ditampilkan dengan warna hijau, dan  $h(y)$  ditampilkan dengan warna kuning. Dengan *random search*, kesembilan percobaan mengeksplorasi nilai-nilai berbeda dari  $g$ .



Gambar 2.6 *Random search* [33]

### 3. *Bayesian optimization*

*Bayesian optimization* adalah pendekatan yang menggunakan Teorema Bayes untuk mengarahkan pencarian guna menemukan fungsi tujuan minimum atau maksimum. Penyetelan *hyperparameter* adalah masalah pengoptimalan di mana fungsi tujuan pengoptimalan tidak diketahui atau fungsi *black box*. *Bayesian optimization* adalah algoritme optimasi yang sangat efektif dalam menyelesaikan masalah optimasi semacam ini [36]. Pendekatan ini menggabungkan informasi sebelumnya tentang fungsi yang tidak diketahui dengan informasi sampel, untuk mendapatkan informasi posterior dari distribusi fungsi dengan menggunakan rumus Bayesian. Kemudian berdasarkan informasi posterior, dapat disimpulkan dimana fungsi tersebut memperoleh nilai optimal.

*Bayesian optimization* menggabungkan distribusi sebelumnya dari fungsi  $f(x)$  dengan informasi sampel untuk mendapatkan fungsi posterior; kemudian informasi posterior digunakan untuk mencari dimana fungsi  $f(x)$  dimaksimalkan menurut suatu kriteria. Kriteria tersebut diwakili oleh fungsi

utilitas  $u$  yang juga disebut fungsi akuisisi. Fungsi  $u$  digunakan untuk menentukan titik sampel berikutnya untuk memaksimalkan utilitas yang diharapkan. Saat mencari area pengambilan sampel, perlu mempertimbangkan eksplorasi (pengambilan sampel dari wilayah dengan ketidakpastian tinggi) dan eksploitasi (pengambilan sampel dari wilayah dengan nilai tinggi). Hal tersebut akan membantu mengurangi jumlah pengambilan sampel. Selain itu, performanya akan ditingkatkan meskipun fungsinya memiliki beberapa *local maxima*.

Selain informasi sampel, *Bayesian optimization* bergantung pada distribusi sebelumnya dari fungsi  $f$ , yang merupakan elemen yang sangat diperlukan dalam inferensi statistik dari distribusi posterior fungsi  $f$ . Distribusi apriori tidak harus menjadi dasar obyektif, baik sebagian atau seluruhnya berdasarkan keyakinan subjektif. Secara umum diasumsikan bahwa proses Gaussian sangat sesuai dengan distribusi sebelumnya dari pengoptimalan Bayesian. Proses Gaussian sangat fleksibel dan mudah ditangani, sehingga *Bayesian optimization* menerapkan proses Gaussian untuk menyesuaikan data dan memperbarui distribusi posterior.

Prosedur *Bayesian optimization* adalah sebagai berikut, dimana  $D_{1:t-1} = \{x_n, y_n\}_{n=1}^{t-1}$  merepresentasikan *dataset training* yang terdiri dari  $t-1$  observasi fungsi  $f$ . Pilih *hyperparameter*  $\eta \in \mathbb{R}^+$ , tentukan  $g_0^i = 0$  untuk  $i=1,2,\dots, N$ . Untuk  $t=1, 2,\dots$  dilakukan pengulangan:

- a. Nominasikan poin dari setiap fungsi akuisisi menggunakan persamaan (2-10).

$$x_t^i = \arg \max_x u_i(x|D_{1:t-1}) \quad (2-10)$$

- b. Pilih nominasi  $x_t = x_t^j$  dengan probabilitas persamaan (2-11).

$$p_t(j) = \frac{e^{\eta g_{t-1}^j}}{\sum_{l=1}^k e^{\eta g_{t-1}^l}} \quad (2-11)$$

- c. Dapatkan sampel dari fungsi objektif  $f$  dengan persamaan (2-12).

$$y_t = f(x_t) + \epsilon_t \quad (2-12)$$

- d. Tambahkan data dengan persamaan (2-13) dan perbarui posterior fungsi  $f$ .

$$D_{1:t} = \{D_{1:t-1}, (x_t, y_t)\} \quad (2-13)$$

- e. Menerima *rewards* dengan persamaan (2-14)

$$r_t^i = \mu_t(x_t^i) \quad (2-14)$$

- f. Perbarui *gains* dengan persamaan (2-15)

$$g_t^i = g_{t-1}^i + r_t^i \quad (2-15)$$

Prosedur *Bayesian optimization* terdiri atas dua bagian, yaitu memperbarui distribusi posterior dan memaksimalkan fungsi akuisisi. Saat pengamatan terakumulasi, distribusi posterior diperbarui terus menerus berdasarkan posterior baru, titik dimana fungsi akuisisi dimaksimalkan ditemukan dan ditambahkan ke *dataset training*. Fungsi akuisisi menggunakan *GP-hedge* dimana setelah setiap pilihan tindakan, algoritme menerima *reward* untuk setiap tindakan dan memperbarui vektor *gain*. Seluruh proses diulangi hingga jumlah iterasi maksimum tercapai atau selisih antara nilai saat ini dan nilai optimal yang diperoleh sejauh ini kurang dari ambang batas yang telah ditentukan.

Posterior fungsi  $f$  dapat ditentukan menggunakan proses Gaussian. Proses Gaussian adalah proses stokastik sehingga setiap sub-kumpulan hingga dari variabel acak memiliki distribusi Gaussian multivariat. Proses Gaussian mengasumsikan bahwa input yang serupa menghasilkan output yang serupa, dan dengan demikian mengasumsikan model statistik dari fungsi tersebut. Seperti distribusi Gaussian yang didefinisikan oleh *mean* dan kovarians, proses Gaussian ditentukan oleh fungsi *mean*  $m : x \rightarrow \mathbb{R}$  dan fungsi kovarian  $k : x \times x \rightarrow \mathbb{R}$ . Proses Gaussian didefinisikan dengan persamaan (2-16)

$$f(x) \sim GP(m(x), k(x_i, x_j)) \quad (2-16)$$

Proses Gaussian berbeda dengan distribusi Gaussian, yaitu fungsi kepadatan probabilitas  $f(x)$  untuk sembarang  $x$  bukan lagi skalar tetapi fungsi distribusi normal untuk semua kemungkinan nilai  $f(x)$ . Untuk memudahkan, asumsikan fungsi *mean* proses Gaussian  $m(x) = 0$ . Untuk fungsi kovarians  $k$ , fungsi kuadrat eksponensial didefinisikan dengan persamaan (2-17).

$$k(x_i, x_j) = \exp\left(-\frac{1}{2}\|x_i - x_j\|^2\right) \quad (2-17)$$

Dimana  $x_i$  dan  $x_j$  masing-masing mewakili sampel ke- $i$  dan ke- $j$ . Ketika  $x_i$  dan  $x_j$  berdekatan, nilai mendekati 1, jika tidak, mendekati 0. Jika dua titik pengambilan sampel berdekatan, keduanya memiliki korelasi yang kuat dan saling mempengaruhi. Ketika mereka semakin berpisah, pengaruh timbal balik itu lemah.

Proses penentuan distribusi posterior  $f(x)$  adalah sebagai berikut:

- a. Observasi sampel  $t$  sebagai set *training*  $D_{1:t} = \{x_n, f_n\}_{n=1}^t, f_n = f(x_n)$ , nilai fungsi  $f$  diambil berdasarkan distribusi normal multivariat  $f \sim N(0, K)$ , dimana  $K$  didefinisi sebagai (2-18).

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots & k(x_1, x_t) \\ k(x_2, x_1) & k(x_2, x_2) & \dots & k(x_2, x_t) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_t, x_1) & k(x_t, x_2) & \dots & k(x_t, x_t) \end{bmatrix} \quad (2-18)$$

Setiap elemen di dalam  $K$  dihitung dengan (2-17). Fungsi  $k$  mengukur derajat perkiraan antara dua sampel. Elemen diagonal  $k(x_i, x_i) = 1$  tanpa mempertimbangkan efek *noise*.

- b. Berdasarkan fungsi  $f$ , hitung nilai fungsi  $f_{t+1} = f(x_{t+1})$  pada titik sampel baru  $x_{t+1}$ . Berdasarkan asumsi proses Gaussian,  $f_{1:t}$  pada set *training* ditambah nilai fungsi  $f_{t+1}$  mengikuti distribusi normal dimensi  $t+1$  sebagai (2-19).

$$\begin{bmatrix} f_{1:t} \\ f_{t+1} \end{bmatrix} \sim N\left(0, \begin{bmatrix} K & k \\ k^T & k(x_{t+1}, x_{t+1}) \end{bmatrix}\right) \quad (2-19)$$

dimana:

$$f_{1:t} = [f_1, f_2, \dots, f_t]^T \quad (2-20)$$

$$k = [k(x_{t+1}, x_1)k(x_{t+1}, x_2) \dots k(x_{t+1}, x_t)] \quad (2-21)$$

$f_{t+1}$  mengikuti distribusi normal satu dimensi sebagai (2-22).

$$f_{t+1} \sim N(\mu_{t+1}, \sigma_{t+1}^2) \quad (2-22)$$

dimana sifat-sifat gabungan Gaussian (distribusi normal) sebagai (2-23)

dan (2-24).

$$\mu_{t+1}(x_{t+1}) = k^T K^{-1} f_{1:t} \quad (2-23)$$

$$\sigma_{t+1}^2(x_{t+1}) = -k^T K^{-1} + k(x_{t+1}, x_{t+1}) \quad (2-24)$$

Pada persamaan diatas, proses Gaussian tidak mengembalikan nilai skalar untuk  $f_{t+1}$ , tapi mengembalikan distribusi probabilitas atas semua kemungkinan  $f_{t+1}$ . Jika set *training* cukup besar, Gaussian dapat memperoleh perkiraan-perkiraan distribusi fungsi  $f(x)$ .

### 2.2.6 *Standardization*

*Standardization* atau standarisasi, disebut juga *Z-score normalization* merupakan metode *feature scaling* yang dilakukan saat *data pre-processing*. *Feature scaling* adalah metode yang digunakan untuk menormalkan rentang variabel independen atau fitur data. Standarisasi suatu *dataset* melibatkan pengubahan skala distribusi nilai, sehingga nilai rata-rata (*mean*) yang diamati adalah 0 dan standar deviasi adalah 1 [58]. Standarisasi fitur sehingga fitur berpusat di sekitar 0 dengan standar deviasi 1 tidak hanya penting jika kita membandingkan pengukuran yang memiliki unit berbeda, tetapi juga merupakan persyaratan umum untuk banyak algoritme pembelajaran. Skor standar dari vektor fitur  $x$  dihitung dengan persamaan (2-25), dimana  $u$  adalah *mean* dari vektor fitur dan  $s$  adalah standar deviasi.

$$z = \frac{(x - u)}{s} \quad (2-25)$$

### 2.2.7 Kinerja Klasifikasi

*Confusion Matrix* adalah alat visualisasi yang menggambarkan kinerja model klasifikasi pada serangkaian data uji yang nilai sebenarnya diketahui. Terdapat 4 istilah sebagai representasi dari hasil proses klasifikasi dalam *confusion matrix*, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Jika data positif diprediksi dengan benar, disebut *True Positive* (TP), tapi jika diprediksi dengan salah disebut *False Negative* (FN). Jika data negatif diprediksi dengan benar, disebut *True Negative* (TN), tapi jika diprediksi dengan salah, disebut *False Positive* (FP). *Confusion matrix* dapat dilihat pada Tabel 2.1.

Tabel 2.1 *Confusion matrix*

	<i>Predicted negative</i>	<i>Predicted positive</i>
<i>Actual negative</i>	<i>True Negative</i> (TN)	<i>False Positive</i> (FP)
<i>Actual positive</i>	<i>False Negative</i> (FN)	<i>True Positive</i> (TP)

Beberapa metrik kinerja digunakan untuk mengukur kinerja model pada penelitian ini, yaitu:

#### 1. Akurasi

Akurasi mengacu pada jumlah total rekaman yang diklasifikasikan dengan benar oleh pengklasifikasi. Akurasi dihitung menggunakan persamaan (2-26).

$$\text{Akurasi} = \frac{TP+TN}{TP+FP+FN+TN} \times 100\% \quad (2-26)$$

#### 2. Sensitivitas

Sensitivitas atau *recall*, disebut juga *true positive rate* adalah rasio prediksi positif benar dibandingkan dengan data benar positif secara keseluruhan. Sensitivitas dihitung menggunakan persamaan (2-27).

$$\text{Sensitivitas} = \frac{TP}{TP+FN} \times 100\% \quad (2-27)$$

### 3. Spesifisitas

Spesifisitas adalah rasio prediksi negatif benar dibandingkan dengan data benar negatif secara keseluruhan. Spesifisitas dihitung menggunakan persamaan (2-28).

$$\text{Spesifisitas} = \frac{TN}{TN+FP} \times 100\% \quad (2-28)$$

### 4. Presisi

Presisi adalah rasio positif benar terhadap total positif yang diprediksi. Presisi dihitung menggunakan persamaan (2-29).

$$\text{Presisi} = \frac{TP}{TP+FP} \times 100\% \quad (2-29)$$

### 5. Skor F1

Skor F1 atau *f-measure* merupakan perbandingan rata-rata pengukuran dari presisi dan *recall* yang dibobot. Skor F1 yang tinggi menunjukkan bahwa model memiliki nilai presisi dan *recall* yang baik. Skor F1 dihitung menggunakan persamaan (2-30).

$$\text{Skor F1} = \frac{2 \times \text{presisi} \times \text{recall}}{\text{presisi} + \text{recall}} \times 100\% \quad (2-30)$$

## 2.3 Hipotesis

Berdasarkan hasil tinjauan pustaka dan landasan teori, didapatkan hipotesis penelitian bahwa menggunakan metode *deep neural network* dengan *Bayesian optimization* dapat secara efisien menyetel *hyperparameter* dan menghasilkan akurasi kinerja klasifikasi yang baik untuk mendeteksi penyakit jantung.

## BAB III

### METODOLOGI

#### 3.1 Alat dan Bahan

Alat yang digunakan dalam penelitian ini meliputi *hardware* dan *software*. Sedangkan, bahan yang digunakan adalah *dataset* penyakit jantung *Cleveland* yang diperoleh dari *repository machine learning University of California at Irvine* (UCI).

##### 3.1.1 Alat

Alat yang digunakan dalam penelitian ini adalah sebagai berikut:

###### 1. *Hardware*

Hardware yang digunakan adalah sebuah laptop dengan spesifikasi sebagai berikut:

- a. *Processor intel core i5-10210U quad-core 1,6GHz*
- b. *Grafis Intel UHD Graphics 620 dan Nvidia GeForce MX130*
- b. RAM 8 GB
- c. SSD 512 GB

###### 2. *Software*

- a. Sistem operasi Windows 10
- b. Weka 3.8.4, digunakan untuk *data cleaning* dan konversi data.
- c. Bahasa pemrograman Python, sebagai bahasa pemrograman utama untuk mengembangkan model *deep neural network*.
- d. Google Colaboratory, digunakan sebagai *editor* untuk mengeksekusi *code* Python.
- e. Google Chrome, sebagai aplikasi peramban web untuk mengakses Google Colaboratory.
- f. TensorFlow, sebagai *framework* komputasional untuk membangun model *deep neural network*.
- g. Keras, sebagai API *deep learning* yang berjalan diatas TensorFlow.
- h. Pandas, sebagai alat analisis dan manipulasi data untuk bahasa

pemrograman Python.

- i. NumPy, sebagai *library* Python untuk pekerjaan komputasi numerik.
- j. Scikit-learn atau sklearn, sebagai *library* untuk bahasa pemrograman Python.
- k. Scikit-optimize, atau skopt, adalah *library* sederhana dan efisien untuk meminimalkan fungsi *black-box* yang mahal dan noisy. Skopt mengimplementasikan beberapa metode untuk optimasi berbasis model sekuensial.
- l. Matplotlib, sebagai pustaka komprehensif untuk membuat visualisasi dengan Python.

### 3.1.2 Bahan

Bahan yang akan digunakan dalam penelitian ini adalah *dataset* penyakit jantung *Cleveland* dari *repository machine learning University of California Irvine* (UCI). *Dataset* ini bersifat *multivariate*, terdapat 303 data dan hanya 14 atribut yang digunakan dari 76 atribut yang ada [59]. Dari 14 atribut yang digunakan, terdapat 13 atribut *input* dan 1 atribut *output*. Atribut-atribut yang ada dalam *dataset* penyakit jantung *Cleveland* dijelaskan sebagai berikut:

1. *Age*: Usia dalam tahun (numerik)
2. *Sex*: Jenis kelamin (1= laki-laki, 0= perempuan)
3. *Cp (chest pain type)*: *Chest pain* adalah tekanan atau rasa berat di dada, dikategorikan dengan 3 nilai (1= *typical angina*, 2= *atypical angina*, 3= *Non-anginal pain*, 4= *asymptomatic*). Penjelasan mengenai katogori *chest pain* sebagai berikut:
  - a. *Typical angina*: gejala umum nyeri dada yang kemungkinan tinggi terjadi penyumbatan arteri koroner.
  - b. *Atypical angina*: gejala tidak rinci yang dialami pasien sehingga kemungkinan penyumbatan arteri koroner rendah.
  - c. *Non-anginal pain*: rasa sakit menusuk seperti pisau yang

berlangsung lama atau pun sebentar.

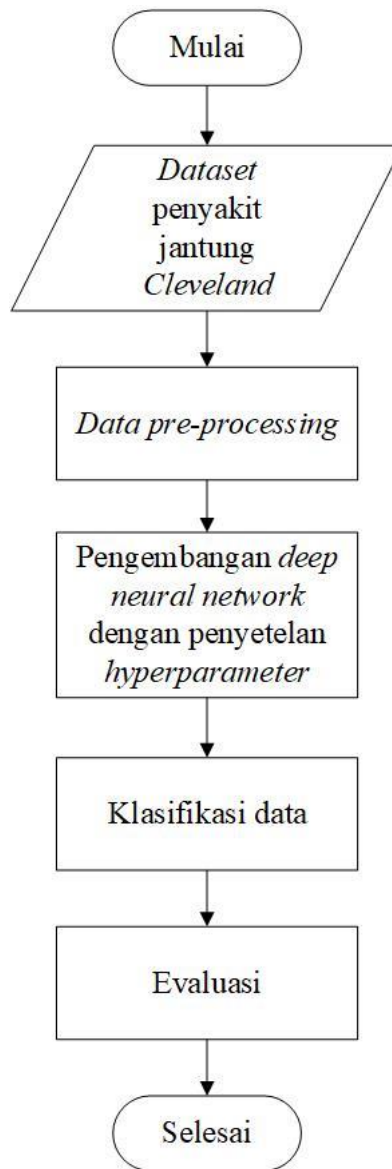
d. *Asyptomatic*: tidak menunjukkan adanya gejala penyakit.

4. *Trestbps (resting blood pressure)*: Tekanan sirkulasi darah pada dinding pembuluh darah dalam mm Hg saat masuk ke rumah sakit (numerik).
5. *Chol*: Serum kolesterol dalam mg/ dl (numerik).
6. *Fbs (fasting blood sugar)*: Variabel ini memiliki nilai boolean yang menunjukkan apakah tekanan darah puasa lebih besar dari 120 mg/dl atau tidak (1= *true*, 0= *false*)
7. *Restecg*: Hasil elektrokardiografi selama istirahat dengan tiga nilai (0= normal, 1= memiliki gelombang ST-T abnormal, 2= menunjukkan kemungkinan atau pasti hipertrofi ventrikel kiri).
8. *Thalach*: denyut jantung maksimum yang dapat dicapai (numerik).
9. *Exang*: Variabel dengan nilai boolean yang menunjukkan apakah latihan yang diinduksi angina terjadi (1= ya, 0= tidak)
10. *Oldpeak*: Depresi ST disebabkan oleh olahraga relatif terhadap istirahat.
11. *Slope*: Kemiringan segmen ST untuk latihan maksimum (puncak), dengan nilai 1= *upsloping*, 2= *flat*, 3= *downsloping*.
12. *Ca*: Jumlah pembuluh besar diwarnai oleh *flourosopy*, dengan 4 nilai (0, 1, 2, 3).
13. *Thal*: Hasil tes stres talium mengenai status jantung dengan tiga nilai (3 = normal, 6 = cacat tetap, 7 = cacat reversibel).
14. *Num*: Atribut *class* yang menunjukkan status *angiography* (0, 1, 2, 3, 4), dimana (0) berarti tidak ada penyakit jantung dan (1, 2, 3, 4) berarti memiliki penyakit jantung.

### 3.2 Metode

Penelitian ini bertujuan untuk mengembangkan model *deep neural network* dengan penyetulan *hyperparatemer Bayesian optimization* untuk menghasilkan

akurasi yang baik dalam mendeteksi penyakit jantung. Kinerja *Bayesian optimization* juga dibandingkan dengan penyetelan manual, *grid search* dan *random search*. Langkah-langkah penelitian dapat dilihat pada diagram alir Gambar 3.1.

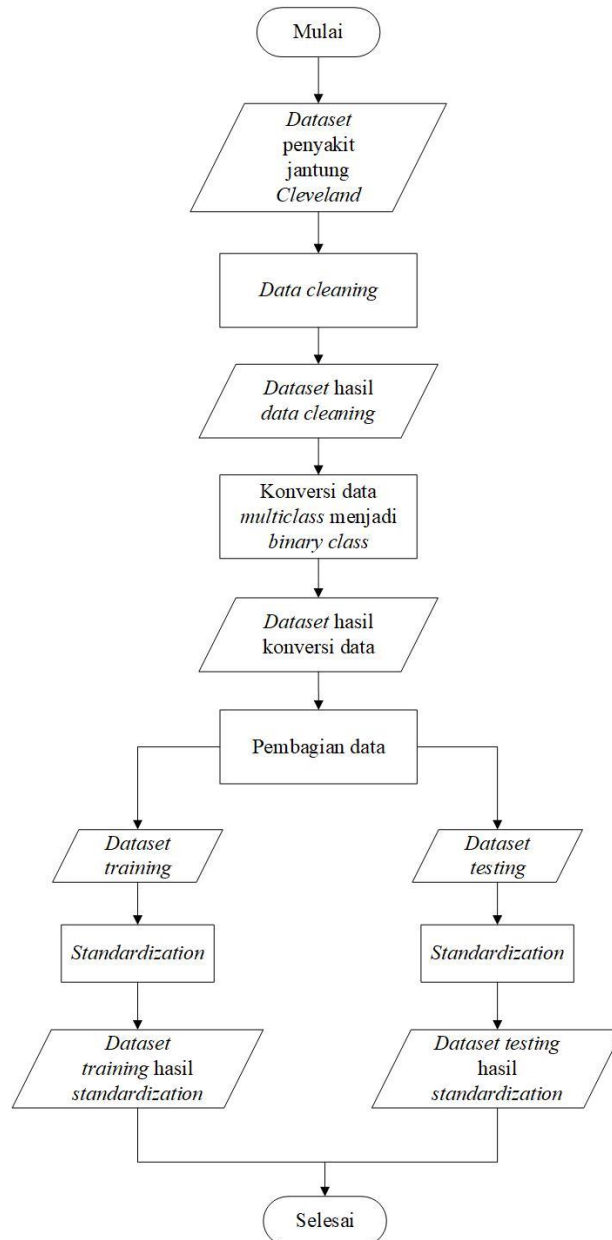


Gambar 3.1 Diagram alir penelitian

### 3.2.1 *Data Pre-processing*

*Data pre-processing* adalah tahapan awal dalam penelitian ini. *Data pre-processing* dilakukan dalam 4 tahapan, meliputi *data cleaning*, konversi data dari

data *multiclass* menjadi *binary class*, pembagian data dan *standardization*. Proses *data cleaning* dan konversi data *multiclass* ke *binary class* dilakukan menggunakan aplikasi Weka. Lalu, proses pembagian data dan *standardization* dilakukan menggunakan Google Colaboratory dan sklearn. Langkah-langkah *data pre-processing* dapat dilihat pada Gambar 3.2.



Gambar 3.2 Diagram alir *data pre-processing*

### 3.2.1.1 Data Cleaning

Pada proses *data cleaning*, dilakukan penghapusan data *missing values*.

*Missing value* adalah data yang tidak memiliki nilai tersimpan untuk suatu variabel. Data *missing value* akan dihapus agar tidak mempengaruhi validitas kesimpulan penelitian. Pada *dataset* penyakit jantung *Cleveland*, terdapat 6 data *missing values* dari 303 data yang ada. Menggunakan aplikasi Weka, data *missing value* dihapus dengan fungsi *RemoveWithValues*. Sehingga, data yang digunakan dalam penelitian ini berjumlah 297 data. Rincian data *missing values* yang terdapat pada *dataset* penyakit jantung *Cleveland* ditunjukkan pada Tabel 3.1.

Tabel 3.1 Rincian data *missing values*

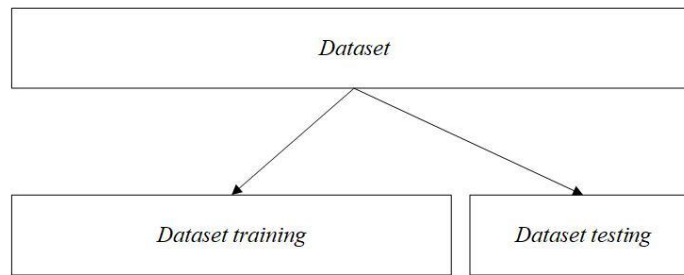
age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
53	0	3	128	216	0	2	115	0	0	1	0	?	0
52	1	3	138	223	0	0	169	0	0	1	?	3	0
43	1	4	132	247	1	2	143	1	0,1	2	?	7	1
52	1	4	128	204	1	0	156	1	1	2	0	?	2
58	1	2	125	220	0	0	144	0	0,4	2	?	7	0
38	1	3	138	175	0	0	173	0	0	1	?	3	0

### 3.2.1.2 Konversi Data

Atribut “*num*” merupakan atribut *class* yang mengkategorikan tingkat keparahan penyakit jantung pada pasien, yaitu (0, 1, 2, 3, dan 4). Penelitian ini berfokus pada klasifikasi biner, sehingga atribut “*num*” dikonversi dari *multiclass* ke *binary class*. Proses konversi data *multiclass* ke *binary class* dilakukan dengan asumsi (0) adalah pasien yang sehat dan (1, 2, 3, 4) dikonversi menjadi (1), yang dikategorikan sebagai pasien dengan penyakit jantung. Konversi data dilakukan menggunakan aplikasi Weka, pertama atribut “*num*” diubah dari mumerik menjadi nominal dengan fungsi *NumericToNominal*, lalu atribut “*num*” yang bernilai (1,2,3,4) dikonversi menjadi (1) dengan fungsi *MergeManyValues*.

### 3.2.1.3 Pembagian Data

Pada tahap ini, *dataset* dibagi menjadi 2, yaitu *dataset training* dan *dataset testing*. *Dataset training* dan *dataset testing* memiliki perbandingan 80:20. Pembagian data ditunjukkan seperti Gambar 3.3.



Gambar 3.3 Pembagian data

Penjelasan mengenai pembagian data tersebut sebagai berikut:

1. *Dataset training*

*Dataset training* atau data pelatihan adalah data yang digunakan dalam proses *training* model *deep neural network* dengan penyetelan *hyperparameter* manual, *grid search*, *random search*, dan *Bayesian optimization*. Pada penelitian ini, *dataset training* berisi 80% dari 297 data.

2. *Dataset testing*

*Dataset testing* atau data pengujian hanya digunakan setelah model selesai di-*training* sepenuhnya. *Dataset testing* digunakan untuk evaluasi kinerja model *deep neural network* dengan *hyperparameter* optimal yang didapat dari penyetelan *hyperparameter* manual, *grid search*, *random search*, dan *Bayesian optimization*. Pada penelitian ini, *dataset testing* berisi 20% dari 297 data.

Pada Google Colaboratory, proses pembagian data dilakukan menggunakan fungsi *train\_test\_split* dari *sklearn*. *Dataset* dibagi menjadi dua bagian, bagian pertama untuk *dataset training* yang memiliki 80% dari *dataset* dan bagian kedua untuk *dataset testing*, memiliki 20% dari *dataset*. Parameter *random state* digunakan untuk mengontrol pengacakan yang diterapkan ke data sebelum menerapkan pembagian data. Pada penelitian ini, dihasilkan 20 *dataset training* dan *testing* baru dengan urutan *instance* yang berbeda-beda untuk diproses dalam percobaan sebanyak 20 kali.

### 3.2.1.4 *Standardization*

Pada *dataset training* dan *dataset testing*, dilakukan proses *standardization*. *Standardization* perlu dilakukan karena variabel yang diukur pada skala yang berbeda tidak memberikan kontribusi yang sama dan dapat menciptakan bias. Pada Google Colaboratory, proses standardisasi dilakukan menggunakan fungsi *StandardScaler* dari *sklearn*. *StandardScaler* menstandarisasi fitur dengan menghapus *mean* dan mengskalakan ke varian unit. Skor standar dari vektor fitur  $x$  dihitung dengan persamaan (3-1), dimana  $u$  adalah *mean* dari vektor fitur dan  $s$  adalah standar deviasi.

$$z = \frac{(x - u)}{s} \quad (3-1)$$

### 3.2.2 **Pengembangan *Deep Neural Network* dengan Penyetelan *Hyperparameter***

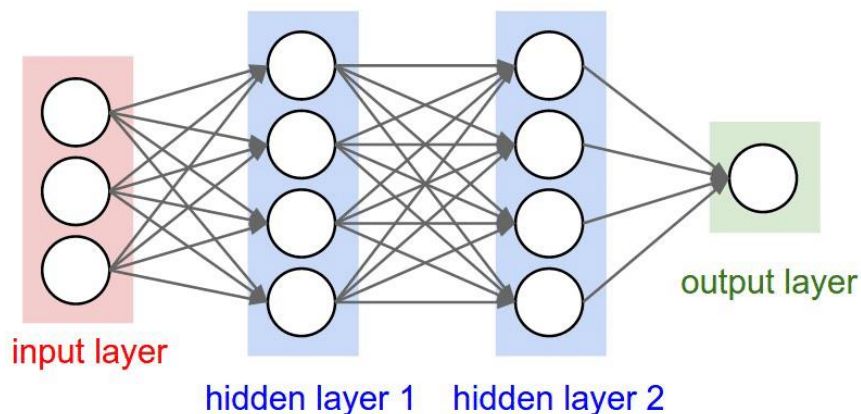
Setelah *data pre-processing*, selanjutnya adalah melakukan proses pengembangan *deep neural network* dengan penyetelan *hyperparameter* menggunakan penyetelan manual, *grid search*, *random search*, *Bayesian optimization* pada *dataset training* hasil *standardization*. Gambar 3.4 menunjukkan langkah-langkah proses pengembangan *deep neural network* dengan penyetelan *hyperparameter* menggunakan penyetelan manual, *grid search*, *random search* dan *Bayesian optimization*.



Gambar 3.4 Alur pengembangan DNN dengan penyetelan *hyperparameter*

### 3.2.2.1 Model *Deep Neural Network*

Pengembangan model *deep neural network* untuk klasifikasi biner penyakit jantung dilakukan menggunakan Google Colaboratory, Keras, dan TensorFlow. *Deep neural network* dibangun dengan model sekuensial, terdiri atas *input layer*, *hidden layer*, dan *output layer*. Perancangan arsitektur *deep neural network* dapat dilihat seperti pada Gambar 3.5.



Gambar 3.5 Rancangan arsitektur DNN

Arsitektur *deep neural network* dibangun dengan model sekuensial yang

terdiri atas *input layer*, *hidden layer*, dan *output layer*. *Tiap layer* memiliki neuron yang mana setiap neuron menghitung jumlah bobot inputnya dan menambahkan bias. Neuron didefinisikan dengan fungsi (3-2), dimana variabel  $W$  adalah bobot,  $x$  adalah masukan, dan  $b$  adalah suku bias.

$$y = Wx + b \quad (3-2)$$

Setiap lapisan pada rancangan arsitektur *deep neural network* dijelaskan sebagai berikut:

1. *Input layer*

*Input layer* adalah lapisan yang menerima input dari luar ke jaringan. Pada lapisan ini, tidak ada komputasi yang dilakukan, *input nodes* hanya meneruskan informasi ke *hidden nodes*. Pada penelitian ini, *input* yang diterima sebanyak 13, karena terdapat 13 fitur pada *dataset*.

2. *Hidden layer*

*Hidden layer* adalah lapisan yang berada di antara *input layer* dan *output layer*, dimana terjadi komputasi. *Deep neural network* memiliki lebih dari 1 *hidden layer*. Setiap *hidden layer* memiliki neuron-neuron atau *nodes* yang disebut *hidden nodes* atau *hidden units*. Pada penelitian ini, dilakukan penyetelan *hyperparameter* menggunakan pengaturan *beberapa hidden layer* dan *hidden units* untuk mendapat model dengan hasil yang optimal. Jumlah *hidden layer* dan *hidden units* yang digunakan dapat mempengaruhi kinerja model.

Pada *hidden layer*, digunakan *activation function* ReLU. ReLU menghasilkan *output* yang antara 0 dan  $x$ . ReLU didefinisikan dengan persamaan (3-3) dimana jika  $x$  negatif, outputnya 0 dan jika  $x$  positif, outputnya  $x$ .

$$\text{ReLU}(x) = \max(0, x) \quad (3-3)$$

Pada *hidden layer* terakhir, *dropout* digunakan untuk mencegah *overfitting* pada model. *Dropout* menjatuhkan unit dari jaringan selama *training* secara acak untuk mencegah unit beradaptasi secara berlebihan.

*Dropout rate* dapat bernilai antara 0 dan 1.

### 3. *Output layer*

*Output layer* memberikan hasil untuk *input* yang diberikan. Pada penelitian ini, *output layer* memiliki 1 *output node* yang menghasilkan unit *output* biner (0 atau 1).

Pada *output layer*, *activation function* yang digunakan adalah *sigmoid*. *Sigmoid* digunakan sebagai *activation function* pada *output layer* untuk klasifikasi kelas biner. *Sigmoid* didefinisikan dengan persamaan (3-4) dimana ketika  $x$  meningkat menuju tak terhingga positif, *output* mendekati 1. Jika  $x$  menurun ke tak terhingga negatif, *output* mendekati 0.

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (3-4)$$

Model *deep neural network* di-*training* menggunakan *binary cross-entropy* sebagai *loss function* dan *Adam* sebagai *optimizer*. *Binary cross-entropy* menghitung *cross-entropy loss* antara label aktual dan label yang diprediksi. *Binary cross-entropy* adalah *loss function* untuk tugas klasifikasi biner, dimana hanya ada dua label kelas (0 atau 1). *Binary cross-entropy* didefinisikan sebagai (3-5), dimana  $y$  adalah *output* aktual dan  $p$  adalah *output* yang diprediksi.

$$L(y, p) = -(y \log(p) + (1 - y) \log(1 - p)) \quad (3-5)$$

*Adam* menyimpan perkiraan momen pertama dan kedua dari gradien dan menggunakannya untuk memperbarui parameter. Diberikan fungsi tujuan stokastik  $f(W)$  dengan parameter  $W_t$ , dimana  $t$  mengindeks iterasi pelatihan, momen pertama dan kedua didefinisikan sebagai (3-6) dan (3-7).

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (3-6)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (3-7)$$

Dimana, istilah momentum pertama  $\beta_1 = 0,9$  dan momentum kedua  $\beta_2 = 0,999$ ,  $g_t = \nabla_W f_t(W_{t-1})$  menunjukkan gradien w.r.t  $W$  pada iterasi  $t$ . Dengan inisialisasi  $m_0 = 0$  dan  $v_0 = 0$ , perkiraan  $m_t$  dan  $v_t$  bias menuju nol. Bias ini diatasi dengan menghitung perkiraan momen pertama dan kedua bias yang dikoreksi sebagai (3-8)

dan (3-9).

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (3-8)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (3-9)$$

Kemudian, aturan pembaruan terakhir *Adam* akan didefinisikan sebagai (3-10).

$$W_t = W_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \quad (3-10)$$

Dimana,  $\epsilon$  adalah skalar kecil yang digunakan untuk mencegah pembagian dengan nol,  $\epsilon = 10^{-8}$  dan  $\eta$  adalah *learning rate* awal dengan pengaturan default  $\eta = 0,001$ .

*Deep neural network* dilatih menggunakan *epoch* dan ukuran *batch* yang ditentukan melalui penyetelan *hyperparameter*. *Epoch* adalah frekuensi semua *dataset training* ditampilkan ke jaringan selama pelatihan. Ukuran *batch* adalah jumlah sampel pelatihan yang digunakan dalam satu iterasi. Penelitian ini menggunakan *mini-batch gradient descent*, ukuran *batch* lebih dari satu sampel dan lebih kecil dari ukuran *dataset training*.

### 3.2.2.2 Penyetelan *Hyperparameter*

Penyetelan *hyperparameter* dilakukan untuk mendapatkan model dengan kinerja yang baik. Penyetelan *hyperparameter* dilakukan sebelum *training* model. *Hyperparameter* yang disetel dalam penelitian ini meliputi jumlah *hidden layer*, jumlah *hidden units*, *dropout rate*, *epochs*, dan ukuran *batch*. Penyetelan *hyperparameter* dilakukan menggunakan penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization*. Setiap teknik menghasilkan *hyperparameter* yang optimal untuk *deep neural network*. Teknik-teknik penyetelan *hyperparameter* yang digunakan pada penelitian ini, dijelaskan sebagai berikut:

#### 1. Penyetelan manual

Penyetelan manual menetapkan nilai *hyperparameter* berdasarkan intuisi atau tebakan. Dengan menggunakan peyetelan manual, model dilatih dengan mencoba beberapa nilai *hyperparameter* secara manual dan dilakukan berkali-kali hingga mendapat skor validasi yang optimal.

## 2. *Grid search*

*Grid search* mencoba kombinasi *hyperparameter* secara menyeluruh dari ruang *hyperparameter* yang ditentukan. Penyetelan *hyperparameter* menggunakan *grid search* dilakukan menggunakan fungsi *GridSearchCV* dari *library* sklearn.

## 3. *Random search*

*Random search* mencoba kombinasi secara acak dari berbagai nilai pada ruang *hyperparameter*. Jumlah kombinasi *hyperparameter* yang dicoba oleh *random search* ditentukan secara eksplisit. Penyetelan *hyperparameter* menggunakan *random search* dilakukan menggunakan fungsi *RandomizedSearchCV* dari *library* sklearn.

## 4. *Bayesian optimization*

*Bayesian optimization* adalah pendekatan yang berdasarkan *Teorema Bayes* dimana *hyperparameter* optimal dipilih dengan cara memperhitungkan evaluasi sebelumnya saat memilih set *hyperparameter* untuk evaluasi berikutnya. Jumlah kombinasi *hyperparameter* yang dicoba oleh *Bayesian optimization* ditentukan secara eksplisit. Penyetelan *hyperparameter* menggunakan *Bayesian optimization* dilakukan menggunakan fungsi *BayesSearchCV* dari *library* skopt

Prosedur *Bayesian optimization* adalah sebagai berikut, dimana  $D_{1:t-1} = \{x_n, y_n\}_{n=1}^{t-1}$  merepresentasikan *dataset training* yang terdiri dari  $t-1$  observasi fungsi  $f$ . Pilih *hyperparameter*  $\eta \in \mathbb{R}^+$ , tentukan  $g_0^i = 0$  untuk  $i=1,2,\dots, N$ . Untuk  $t=1, 2,\dots$  dilakukan pengulangan:

- a. Nominasikan poin dari setiap fungsi akuisisi menggunakan persamaan (3-11).

$$x_t^i = \arg \max_x u_i(x|D_{1:t-1}) \quad (3-11)$$

- b. Pilih nominasi  $x_t = x_t^j$  dengan probabilitas persamaan (3-12).

$$p_t(j) = \frac{e^{ng_{t-1}^j}}{\sum_{l=1}^k e^{ng_{t-1}^l}} \quad (3-12)$$

- c. Dapatkan sampel dari fungsi objektif  $f$  dengan persamaan (3-13).

$$y_t = f(x_t) + \epsilon_t \quad (3-13)$$

- d. Tambahkan data dengan persamaan (3-14) dan perbarui posterior fungsi  $f$ .

$$D_{1:t} = \{D_{1:t-1}, (x_t, y_t)\} \quad (3-14)$$

- e. Menerima *rewards* dengan persamaan (3-15)

$$r_t^i = \mu_t(x_t^i) \quad (3-15)$$

- f. Perbarui *gains* dengan persamaan (3-16)

$$g_t^i = g_{t-1}^i + r_t^i \quad (3-16)$$

Posterior fungsi  $f$  ditentukan menggunakan proses Gaussian yang didefinisikan dengan persamaan (3-17), dimana diasumsikan fungsi *mean*  $m(x)=0$ , fungsi kovarians  $k$  didefinisikan dengan persamaan (3-18),  $x_i$  dan  $x_j$  mewakili sampel ke- $i$  dan ke- $j$ .

$$f(x) \sim GP(m(x), k(x_i, x_j)) \quad (3-17)$$

$$k(x_i, x_j) = \exp\left(-\frac{1}{2}\|x_i - x_j\|^2\right) \quad (3-18)$$

Proses penentuan distribusi posterior  $f(x)$  adalah sebagai berikut:

- a. Observasi sampel  $t$  sebagai set *training*  $D_{1:t} = \{x_n, f_n\}_{n=1}^t$ ,  $f_n = f(x_n)$ , nilai fungsi  $f$  diambil berdasarkan distribusi normal multivariat  $f \sim N(0, K)$ , dimana  $K$  didefinisi sebagai (3-19).

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots & k(x_1, x_t) \\ k(x_2, x_1) & k(x_2, x_2) & \dots & k(x_2, x_t) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_t, x_1) & k(x_t, x_2) & \dots & k(x_t, x_t) \end{bmatrix} \quad (3-19)$$

Setiap elemen di dalam  $K$  dihitung dengan (3-18). Fungsi  $k$  mengukur derajat perkiraan antara dua sampel. Elemen diagonal  $k(x_i, x_i) = 1$  tanpa mempertimbangkan efek *noise*.

- b. Berdasarkan fungsi  $f$ , hitung nilai fungsi  $f_{t+1} = f(x_{t+1})$  pada titik sampel baru  $x_{t+1}$ . Berdasarkan asumsi proses Gaussian,  $f_{1:t}$  pada set *training* ditambah nilai fungsi  $f_{t+1}$  mengikuti distribusi normal dimensi  $t+1$  sebagai (3-20).

$$\begin{bmatrix} f_{1:t} \\ f_{t+1} \end{bmatrix} \sim N\left(0, \begin{bmatrix} K & k \\ k^T & k(x_{t+1}, x_{t+1}) \end{bmatrix}\right) \quad (3-20)$$

dimana:

$$f_{1:t} = [f_1, f_2, \dots, f_t]^T \quad (3-21)$$

$$k = [k(x_{t+1}, x_1)k(x_{t+1}, x_2) \dots k(x_{t+1}, x_t)] \quad (3-22)$$

$f_{t+1}$  mengikuti distribusi normal satu dimensi sebagai (3-23).

$$f_{t+1} \sim N(\mu_{t+1}, \sigma_{t+1}^2) \quad (3-23)$$

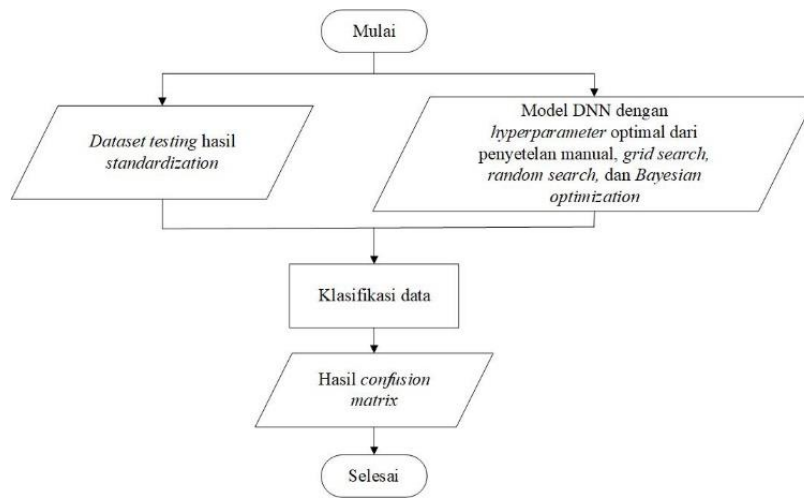
dimana sifat-sifat gabungan Gaussian (distribusi normal) sebagai (3-24) dan (3-25).

$$\mu_{t+1}(x_{t+1}) = k^T K^{-1} f_{1:t} \quad (3-24)$$

$$\sigma_{t+1}^2(x_{t+1}) = -k^T K^{-1} + k(x_{t+1}, x_{t+1}) \quad (3-25)$$

### 3.2.3 Klasifikasi Data

Setelah proses pengembangan *deep neural network* dengan penyetelan *hyperparameter*, dilakukan klasifikasi. Klasifikasi dilakukan menggunakan model *deep neural network* dengan *hyperparameter* yang optimal dari penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization* pada *dataset testing* hasil *standardization*. Setelah dilakukan klasifikasi, akan dihitung *confusion matrix* dari setiap model DNN dengan penyetelan *hyperparameter* menggunakan penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization*. Gambar 3.6 menunjukkan langkah-langkah klasifikasi kinerja model DNN dengan *hyperparameter* optimal dari penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization* pada *dataset testing* hasil *standardization*.



Gambar 3.6 Diagram alir klasifikasi data

### 3.2.4 Evaluasi Kinerja

Evaluasi kinerja model dilakukan untuk mengetahui sebaik apa kemampuan prediktif model, apakah klasifikasi mendekati kelas sebenarnya atau tidak. Evaluasi dilakukan menggunakan beberapa metrik kinerja, yaitu akurasi, sensitivitas, spesifisitas, presisi, dan skor F1. Metrik kinerja tersebut dihitung dengan mengetahui *confusion matrix*. Pada Google Colaboratory, *confusion matrix* hasil klasifikasi dapat dihitung menggunakan fungsi *confusion\_matrix* dari sklearn. Tabel *confusion matrix* hasil klasifikasi dapat dilihat pada Tabel 3.2.

Tabel 3.2 *Confusion matrix* hasil klasifikasi

		<i>Predicted Class</i>	
		<i>Negative (0)</i>	<i>Positive (1)</i>
<i>Actual Class</i>	<i>Negative (0)</i>	TN	FP
	<i>Positive (1)</i>	FN	TP

Akurasi adalah tingkat keberhasilan klasifikasi. Persamaan (3-26) merupakan formula untuk menghitung akurasi.

$$\text{Akurasi} = \frac{TP+TN}{TP+FP+FN+TN} \times 100\% \quad (3-26)$$

Sensitivitas atau *recall* mengukur proporsi positif benar yang diidentifikasi dengan benar. Persamaan (3-27) merupakan formula untuk menghitung sensitivitas.

$$\text{Sensitivitas} = \frac{TP}{TP+FN} \times 100\% \quad (3-27)$$

Spesifisitas mengukur proporsi negatif sebenarnya yang diidentifikasi dengan benar. Persamaan (3-28) merupakan formula untuk menghitung spesifisitas.

$$\text{Spesifisitas} = \frac{TN}{TN+FP} \times 100\% \quad (3-28)$$

Presisi mengukur rasio positif benar terhadap total positif yang diprediksi. Persamaan (3-29) merupakan formula untuk menghitung presisi.

$$\text{Presisi} = \frac{TP}{TP+FP} \times 100\% \quad (3-29)$$

Skor F1 mengukur rata-rata pengukuran dari presisi dan *recall* yang dibobot. Persamaan (3-30) merupakan formula untuk menghitung skor F1.

$$\text{Skor F1} = \frac{2 \times \text{presisi} \times \text{recall}}{\text{presisi} + \text{recall}} \times 100\% \quad (3-30)$$

Dimana:

TP (*True Positive*) : jumlah data positif yang diprediksi dengan benar.

TN (*True Negative*) : jumlah data negatif yang diprediksi dengan benar.

FP (*False Positive*) : jumlah data negatif, tapi diprediksi sebagai data positif.

FN (*False Negative*) : jumlah data positif, tapi diprediksi sebagai data negatif.

## **BAB IV**

### **HASIL DAN PEMBAHASAN**

Berdasarkan langkah-langkah yang telah dijelaskan pada bab 3, penelitian ini bertujuan untuk mengembangkan model *deep neural network* dengan penyetelan *hyperparameter* menggunakan *Bayesian optimization* untuk mendapatkan akurasi kinerja yang baik dalam mendeteksi penyakit jantung. Kinerja *Bayesian optimization* juga dibandingkan dengan penyetelan manual, *grid search*, dan *random search*. Pada bab ini, akan dibahas mengenai hasil penelitian yang diperoleh dari langkah-langkah dalam alur penelitian yang dilakukan. Pembahasan dilakukan secara berurutan mulai dari hasil proses *data pre-processing* yang terdiri atas *data cleaning*, konversi data, pembagian data, dan *standardization*, kemudian dilanjutkan dengan pengembangan model *deep neural network* dengan penyetelan *hyperparameter* dan evaluasi kinerja klasifikasi.

#### **4.1 Data Pre-processing**

Proses *data pre-processing* pada penelitian ini terbagi menjadi *data cleaning*, konversi data dari data *multiclass* menjadi *binary class*, pembagian data, dan *standardization*.

##### **4.1.1 Data Cleaning**

Langkah pertama pada proses *data pre-processing* adalah proses *data cleaning* yaitu menghapus data yang memiliki *missing value* pada *dataset Cleveland*. Proses penghapusan data yang memiliki *missing value* dilakukan dengan menggunakan aplikasi Weka. *Dataset* penyakit jantung *Cleveland* terdiri atas 303 *instances* dan terdapat 6 *instances* yang memiliki *missing value*, yaitu 4 *instances* pada atribut *ca* dan 2 *instances* pada atribut *thal*. Sehingga setelah menghapus data *missing value*, *dataset* yang digunakan pada penelitian ini sebanyak 297 *instances*. Tabel 4.1 dan Tabel 4.2 menunjukkan karakteristik *dataset* penyakit jantung *Cleveland* sebelum dan setelah dilakukan proses *data cleaning*.

Tabel 4.1 Karakteristik *dataset Cleveland* sebelum *data cleaning*

<i>Class</i>	Jumlah <i>instances</i>
0	164
1	55
2	36
3	35
4	13

Tabel 4.2 Karakteristik *dataset Cleveland* setelah *data cleaning*

<i>Class</i>	Jumlah <i>instances</i>
0	160
1	54
2	35
3	35
4	13

#### 4.1.2 Konversi Data

Pada tahap ini dilakukan konversi data *multiclass* menjadi *binary class* menggunakan aplikasi Weka. Pada *dataset Cleveland* terdapat 5 kelas yaitu (0, 1, 2, 3, dan 4). Proses konversi data *multiclass* menjadi *binary class* dilakukan dengan mengasumsikan bahwa kelas (0) adalah pasien sehat dan empat kelas lainnya (1, 2, 3, 4) dikonversi menjadi kelas (1) yaitu pasien dengan penyakit jantung. Tabel 4.3 menunjukkan karakteristik *dataset* penyakit jantung *Cleveland* setelah proses konversi data.

Tabel 4.3 Karakteristik *dataset Cleveland* setelah konversi data

<i>Class</i>	Jumlah <i>instances</i>
0	160
1	137

#### 4.1.3 Pembagian Data

Dataset dibagi menjadi *dataset training* dan *dataset testing* dengan perbandingan 80:20. Sehingga dari 297 data, didapatkan 237 *dataset training* dan 60 *dataset testing*. Proses pembagian data dilakukan menggunakan fungsi *train\_test\_split* dari *library* sklearn. Pengacakan data dilakukan terhadap urutan

*instance* pada *dataset* sebelum menerapkan pembagian. Pengacakan dilakukan dengan memasukkan angka acak pada parameter *random state*. Pada penelitian ini, dihasilkan 20 *dataset training* dan *testing* baru dengan urutan *instance* berbeda-beda. *Dataset training* dan *testing* tersebut digunakan untuk 20 kali percobaan. Gambar 4.1 menunjukkan *code* untuk pembagian data.

```
import numpy as np
from sklearn.model_selection import train_test_split
# fix random seed for reproducibility
seed = 10
np.random.seed(seed)
# Splitting the dataset into the training set and testing set
train,test = train_test_split(dataset, test_size=0.2, random_state=seed)
#save the data
train.to_csv('/content/drive/My Drive/Colab Notebooks/Heart/data/train1.csv',index=False)
test.to_csv('/content/drive/My Drive/Colab Notebooks/Heart/data/test1.csv',index=False)
```

Gambar 4.1 *Code* untuk pembagian data

#### 4.1.4 *Standardization*

Pada tahap ini, dilakukan *standardization* pada *dataset training* dan *dataset testing*, yaitu menstandarisasi fitur dengan menghapus *mean* dan mengskalakan ke varian unit. Pada Google Colaboratory, proses *standardization* dilakukan menggunakan fungsi *StandardScaler* dari *sklearn*. *Code* untuk *standardization* dan hasil *standardization* untuk *dataset training* dan *dataset testing* ditunjukkan pada Gambar 4.2 dan Gambar 4.3.

```
# split train data into input (X) and output (Y) variables
X_train = train.iloc[:, :-1]
Y_train = train.iloc[:, -1:]

# split test data into input (X) and output (Y) variables
X_test = test.iloc[:, :-1]
Y_test = test.iloc[:, -1:]

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Gambar 4.2 *Code* untuk *standardization*

```
X_train = pd.DataFrame(X_train)
X_train.columns = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']
X_train.head(5)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	1.252017	0.707107	0.863863	1.621746	-0.432360	-0.409253	1.000018	-0.515622	-0.700404	1.018193	-0.967630	-0.724139	0.658403
1	-0.211361	-1.414214	0.863863	0.364380	-0.318400	-0.409253	1.000018	0.473508	-0.700404	-0.914317	-0.967630	-0.724139	-0.886563
2	-0.661631	0.707107	-1.193781	-0.092843	0.289389	-0.409253	-1.008493	0.968073	-0.700404	-0.410184	-0.967630	-0.724139	-0.886563
3	-1.449604	0.707107	-2.222602	0.935910	-0.128466	-0.409253	1.000018	1.282796	-0.700404	-0.242140	-0.967630	1.343583	-0.886563
4	1.589720	0.707107	-2.222602	1.621746	-0.318400	2.443479	1.000018	-0.830346	-0.700404	-0.830295	0.603112	0.309722	-0.886563

```
X_test = pd.DataFrame(X_test)
X_test.columns = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']
X_test.head(5)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	0.576612	0.707107	0.863863	-0.092843	-0.850215	-0.409253	1.000018	-0.785385	1.427747	1.102215	0.603112	1.343583	1.173392
1	-0.098793	0.707107	0.863863	-0.664373	-1.192096	-0.409253	-1.008493	-1.639634	-0.700404	0.261993	0.603112	0.309722	1.173392
2	1.814855	-1.414214	-0.164959	-1.235903	0.270395	2.443479	1.000018	-0.875306	-0.700404	-0.914317	-0.967630	0.309722	-0.886563
3	1.026882	-1.414214	0.863863	-0.092843	0.992144	-0.409253	-1.008493	-1.234990	-0.700404	0.766126	0.603112	1.343583	-0.886563
4	0.126342	0.707107	-1.193781	-0.092843	-0.565314	-0.409253	1.000018	0.608389	-0.700404	-0.914317	-0.967630	-0.724139	1.173392

Gambar 4.3 Hasil *standardization* untuk *dataset training* dan *dataset testing*

## 4.2 Pengembangan *Deep Neural Network* dengan Penyetelan *Hyperparameter*

Pada tahap ini, dilakukan pembangunan model *deep neural network* dan penyetelan *hyperparameter* manual, *grid search*, *random search* dan *Bayesian optimization* pada *dataset training* hasil *standardization*.

### 4.2.1 Model *Deep Neural Network*

Proses pengembangan model *deep neural network* dilakukan menggunakan *framework* TensorFlow dan pustaka kode Keras. Model *deep neural network* terdiri atas *input layer*, *hidden layer*, dan *output layer*. *Input layer* menerima *input* untuk dilanjutkan ke *hidden layer*. *Input* yang diterima sebanyak 13, karena terdapat 13 fitur pada *dataset*. Lalu, proses komputasi terjadi pada *hidden layer*. Pada setiap *hidden layer*, terdapat *hidden units* dan *activation function* yang digunakan adalah ReLU. *Output layer* memberikan hasil klasifikasi untuk *input* yang diberikan. *Output layer* memiliki 1 *output node* yang menghasilkan unit *output* biner (0 atau 1). *Activation function* yang digunakan pada *output layer* adalah *sigmoid*. *Dropout* diterapkan di antara *hidden layer* terakhir dan *output layer* untuk mengurangi *overfitting*. Model *deep neural network* menggunakan *binary cross-entropy* sebagai *loss function* dan *Adam* sebagai *optimizer*. Code untuk membangun model *deep neural network* dapat dilihat pada Gambar 4.4.

```

# Importing the libraries
from keras.models import Sequential
from tensorflow import keras
from keras.layers import Dense, Activation, Dropout
from keras.activations import relu, sigmoid
from keras.wrappers.scikit_learn import KerasClassifier

def create_model(hidden_layers, neurons, dropout_rate):
    model = Sequential()
    for i in range(int(hidden_layers)):
        if i==0:
            model.add(Dense(neurons, input_shape = [13], activation= 'relu'))
        else:
            model.add(Dense(neurons, activation='relu'))

    model.add(Dropout(dropout_rate))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    return model

```

Gambar 4.4 Code untuk membangun model DNN

#### 4.2.2 Penyetelan *Hyperparameter* Manual

Penyetelan *hyperparameter* manual untuk model *deep neural network* dilakukan dengan cara mencoba beberapa nilai *hyperparameter* secara manual. *Hyperparameter* yang akan dicari nilai optimalnya meliputi jumlah *hidden layer*, jumlah *hidden units/neurons*, *dropout rate*, *epochs*, dan ukuran *batch*. Nilai jumlah *hidden layer* dicoba antara 2 sampai 5. Nilai jumlah *hidden units* dicoba antara 8, 16, 32, dan 64. Nilai *dropout rate* dicoba antara 0,2 dan 0,5. Nilai *epochs* dicoba antara 50 dan 100. Nilai ukuran *batch* dicoba antara 32 dan 64. Penyetelan *hyperparameter* secara manual divalidasi menggunakan *5-fold cross-validation*. Percobaan dilakukan secara berulang-ulang hingga mendapatkan skor validasi yang baik. Code untuk penyetelan manual ditunjukkan pada Gambar 4.5.

```

from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score

# Manual tuning
# create model
model_manual = KerasClassifier(build_fn=create_model, verbose=0, hidden_layers=2, neurons=32, dropout_rate=0.2, epochs=50, batch_size=32)
#fit the model
model_manual.fit(X_train, Y_train)
# evaluate using 5-fold cross validation
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
results = cross_val_score(model_manual, X_train, Y_train, cv=kfold)
print(results.mean())
print(results.std())

```

Gambar 4.5 Code penyetelan manual

Pada penelitian ini, dilakukan 20 kali percobaan menggunakan penyetelan manual untuk *hyperparameter deep neural network* dalam mendeteksi penyakit

jantung. Hasil kombinasi *hyperparameter* yang optimal berdasarkan penyetelan manual pada 20 kali percobaan ditunjukkan pada Tabel 4.4.

Tabel 4.4 Hasil penyetelan manual

No.	Jumlah <i>hidden layer</i>	Jumlah <i>hidden units</i>	<i>Dropout rate</i>	<i>Epochs</i>	Ukuran <i>batch</i>
1.	2	32	0,2	50	32
2.	2	8	0,2	50	32
3.	2	32	0,5	100	64
4.	2	32	0,5	50	32
5.	3	16	0,2	50	32
6.	5	8	0,2	50	32
7.	3	32	0,5	50	64
8.	2	16	0,2	50	32
9.	2	8	0,2	50	32
10.	2	64	0,2	50	64
11.	2	32	0,5	50	32
12.	4	16	0,2	50	32
13.	2	8	0,2	100	64
14.	2	16	0,5	100	64
15.	3	16	0,2	50	32
16.	3	16	0,2	50	64
17.	2	64	0,5	100	64
18.	3	16	0,5	50	32
19.	3	32	0,5	50	32
20.	2	32	0,2	50	64

Dari Tabel 4.4 diketahui kombinasi *hyperparameter* optimal untuk DNN menggunakan penyetelan manual pada 20 kali percobaan. Diketahui pada hasil 20 kali percobaan menggunakan penyetelan manual, jumlah *hidden layer* optimal yang paling sering muncul adalah 2, jumlah *hidden units* optimal yang paling sering muncul adalah 16 dan 32, *dropout rate* optimal yang paling sering muncul adalah 0,2, *epochs* optimal yang paling sering muncul adalah 50, dan ukuran *batch* optimal yang paling sering muncul adalah 32. Dalam penyetelan manual, waktu penyetelan tidak dapat didefinisi karena proses penyetelan *hyperparameter* dengan nilai yang berbeda-beda dilakukan secara manual. Model DNN dengan *hyperparameter* optimal dari penyetelan manual kemudian diuji pada *dataset testing* untuk mengetahui kinerjanya dalam mendeteksi penyakit jantung.

### 4.2.3 Penyetelan *Hyperparameter* Otomatis

Penyetelan *hyperparameter* otomatis untuk model *deep neural network* dilakukan menggunakan *grid search*, *random search*, dan *Bayesian optimization*. *Hyperparameter* yang akan dicari nilai optimalnya meliputi jumlah *hidden layer*, jumlah *hidden units/neurons*, *dropout rate*, *epochs*, dan ukuran *batch*. Dalam melakukan penyetelan *hyperparameter*, ruang pencarian *hyperparameter* didefinisikan terlebih dahulu. Ruang pencarian *hyperparameter* berisi nilai-nilai *hyperparameter* yang akan dicari oleh teknik penyetelan *hyperparameter*. Ruang pencarian *hyperparameter* ditunjukkan pada Tabel 4.5.

Tabel 4.5 Ruang pencarian *hyperparameter*

<b><i>Hyperparameter</i></b>	<b>Nilai</b>
Jumlah <i>hidden layer</i>	[2, 3, 4, 5]
Jumlah <i>hidden units</i>	[8, 16, 32, 64]
<i>Dropout rate</i>	[0,2, 0,5]
<i>Epochs</i>	[50, 100]
Ukuran <i>batch</i>	[32, 64]

#### 4.2.3.1 *Grid Search*

Penyetelan *hyperparameter deep neural network* menggunakan teknik *grid search* dilakukan menggunakan fungsi *GridSearchCV* dari *library sklearn*. *Grid search* menentukan kombinasi *hyperparameter* optimal dengan mencoba semua kombinasi *hyperparameter* pada ruang pencarian *hyperparameter*. *Grid search* mengevaluasi kombinasi *hyperparameter* dengan *5-fold cross-validation*. *Code* untuk *grid search* ditunjukkan pada Gambar 4.6.

```

from sklearn.model_selection import GridSearchCV
import time

# Grid search
# define the space of hyperparameters to search
params_grid = { 'hidden_layers': [2, 3, 4, 5],
                'neurons': [8, 16, 32, 64],
                'dropout_rate': [0.2, 0.5],
                'epochs': [50, 100],
                'batch_size': [32, 64] }

# Create grid search
model_grid = KerasClassifier(build_fn=create_model, verbose=0)
grid = GridSearchCV(estimator=model_grid, param_grid=params_grid, n_jobs=-1, cv=5)
# Fit grid search
start = time.time()
grid_result = grid.fit(X_train, Y_train)
end = time.time()

# Grid search summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
print('Grid search takes {:.2f} seconds to tune'.format(end - start))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))

```

Gambar 4.6 Code grid search

Pada penelitian ini, dilakukan 20 kali percobaan menggunakan *grid search* untuk menyetel *hyperparameter deep neural network* dalam mendeteksi penyakit jantung. Hasil kombinasi *hyperparameter* yang optimal dan waktu penyetelan teknik *grid search* pada 20 kali percobaan ditunjukkan pada Tabel 4.6.

Tabel 4.6 Hasil *grid search*

No.	Jumlah hidden layer	Jumlah hidden units	Dropout rate	Epochs	Ukuran batch	Waktu penyetelan (detik)
1.	3	8	0,2	50	64	621,06
2.	3	16	0,5	50	32	597,04
3.	4	16	0,5	50	64	649,41
4.	2	16	0,5	50	32	645,91
5.	3	16	0,2	100	64	640,34
6.	5	16	0,5	100	64	645,25
7.	5	8	0,5	100	32	594,99
8.	5	16	0,5	50	32	614,95
9.	2	8	0,2	100	64	622,43
10.	4	8	0,2	50	32	623,92
11.	3	16	0,2	50	64	614,71
12.	3	16	0,2	50	64	720,69
13.	2	16	0,2	100	64	682,01
14.	3	16	0,2	50	64	718,87
15.	3	16	0,5	50	32	610,71
16.	2	64	0,5	50	64	605,95

No.	Jumlah <i>hidden layer</i>	Jumlah <i>hidden units</i>	<i>Dropout rate</i>	<i>Epochs</i>	Ukuran <i>batch</i>	Waktu penyetelan (detik)
17.	3	32	0,5	50	64	641,34
18.	5	16	0,2	50	32	640,31
19.	2	16	0,5	50	64	602,91
20.	2	16	0,2	50	32	625,03
<b>Rata-rata</b>						<b>635,89</b>

Dari Tabel 4.6 diketahui kombinasi *hyperparameter* optimal untuk DNN menggunakan *grid search* pada 20 kali percobaan. Diketahui pada hasil 20 kali percobaan menggunakan *grid search*, jumlah *hidden layer* optimal yang paling sering muncul adalah 3, jumlah *hidden units* optimal yang paling sering muncul adalah 16, *dropout rate* optimal yang paling sering muncul adalah 0,2 dan 0,5, *epochs* optimal yang paling sering muncul adalah 50, dan ukuran *batch* optimal yang paling sering muncul adalah 64. Dari 20 kali percobaan, rata-rata waktu penyetelan oleh *grid search* adalah 635,89 detik atau 10,6 menit. Model DNN dengan *hyperparameter* optimal dari *grid search* kemudian diuji pada *dataset testing* untuk mengetahui kinerjanya dalam mendeteksi penyakit jantung.

#### 4.2.3.2 *Random Search*

Penyetelan *hyperparameter deep neural network* menggunakan teknik *random search* dilakukan menggunakan fungsi *RandomizedSearchCV* dari *library sklearn*. *Random search* menentukan kombinasi *hyperparameter* optimal dengan mencoba kombinasi acak dari berbagai nilai *hyperparameter* pada ruang pencarian *hyperparameter*. *Random search* mengevaluasi kombinasi *hyperparameter* dengan *5-fold cross-validation*. Jumlah kombinasi *hyperparameter* yang dicoba oleh *random search* ditentukan sebanyak 30. *Code* untuk *random search* ditunjukkan pada Gambar 4.7.

```

from sklearn.model_selection import RandomizedSearchCV

# Random search
# define the space of hyperparameters to search
params_random = { 'hidden_layers': [2, 3, 4, 5],
                  'neurons': [8, 16, 32, 64],
                  'dropout_rate': [0.2, 0.5],
                  'epochs': [50, 100],
                  'batch_size': [32, 64] }

# Create random search
model_random = KerasClassifier(build_fn=create_model, verbose=0)
random = RandomizedSearchCV(estimator=model_random, param_distributions=params_random, n_jobs=-1, cv=5, n_iter=30, random_state= 0)
# Fit random search
start = time.time()
random_result = random.fit(X_train, Y_train)
end = time.time()

# Random search summarize results
print("Best: %f using %s" % (random_result.best_score_, random_result.best_params_))
print('Random search takes {:.2f} seconds to tune'.format(end - start))
means = random_result.cv_results_['mean_test_score']
stds = random_result.cv_results_['std_test_score']
params = random_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))

```

Gambar 4.7 Code random search

Pada penelitian ini, dilakukan 20 kali percobaan menggunakan *random search* untuk menyetel *hyperparameter deep neural network* dalam mendeteksi penyakit jantung. Hasil kombinasi *hyperparameter* yang optimal dan waktu penyetelan teknik *random search* pada 20 kali percobaan ditunjukkan pada Tabel 4.7.

Tabel 4.7 Hasil *random search*

No.	Jumlah hidden layer	Jumlah hidden units	Dropout rate	Epochs	Ukuran batch	Waktu penyetelan (detik)
1.	2	32	0,2	50	32	149,93
2.	2	8	0,2	100	32	144,83
3.	4	16	0,2	50	64	168,95
4.	5	8	0,5	100	32	159,48
5.	2	8	0,2	100	32	158,13
6.	4	32	0,2	100	32	170,07
7.	2	16	0,5	50	32	145,4
8.	3	16	0,5	50	64	150,46
9.	4	8	0,2	50	32	153,8
10.	2	8	0,5	100	32	152,79
11.	2	16	0,5	50	32	150,57
12.	2	32	0,5	100	32	178,21
13.	4	16	0,2	50	64	162,38
14.	4	16	0,2	50	64	149,72
15.	2	32	0,5	100	32	150,67
16.	4	16	0,5	100	64	150,44

No.	Jumlah hidden layer	Jumlah hidden units	Dropout rate	Epochs	Ukuran batch	Waktu penyetelan (detik)
17.	4	16	0,2	50	64	158,03
18.	3	16	0,5	50	64	153,82
19.	4	8	0,2	50	32	151,27
20.	2	32	0,2	50	32	152,6
<b>Rata-rata</b>						<b>155,58</b>

Dari Tabel 4.7 diketahui kombinasi *hyperparameter* optimal untuk DNN menggunakan *random search* pada 20 kali percobaan. Diketahui pada hasil 20 kali percobaan menggunakan *random search*, jumlah *hidden layer* optimal yang paling sering muncul adalah 2, jumlah *hidden units* optimal yang paling sering muncul adalah 16, *dropout rate* optimal yang paling sering muncul adalah 0,2, *epochs* optimal yang paling sering muncul adalah 50, dan ukuran *batch* optimal yang paling sering muncul adalah 32. Dari 20 kali percobaan, rata-rata waktu penyetelan oleh *random search* adalah 155,58 detik atau 2,59 menit. Model DNN dengan *hyperparameter* optimal dari *random search* kemudian diuji pada *dataset testing* untuk mengetahui kinerjanya dalam mendeteksi penyakit jantung.

#### 4.2.3.3 Bayesian Optimization

Penyetelan *hyperparameter deep neural network* menggunakan teknik *Bayesian optimization* dilakukan menggunakan fungsi *BayesSearchCV* dari *library* skopt. *Bayesian optimization* menggunakan Teorema Bayes untuk mengarahkan pencarian dimana *hyperparameter* optimal dipilih dengan cara memperhitungkan evaluasi sebelumnya saat memilih set *hyperparameter* untuk evaluasi berikutnya. *Bayesian optimization* mengevaluasi kombinasi *hyperparameter* dengan *5-fold cross-validation*. Jumlah kombinasi *hyperparameter* yang dicoba oleh *Bayesian optimization* ditentukan sebanyak 30. *Code* untuk *Bayesian optimization* ditunjukkan pada Gambar 4.8.

```

from skopt import BayesSearchCV
from skopt.space import Categorical, Integer

# bayesian optimization
# define the space of hyperparameters to search
params_bayes= { 'hidden_layers': Integer(2, 5),
                'neurons': Categorical([8, 16, 32, 64]),
                'dropout_rate': Categorical([0.2, 0.5]),
                'epochs': Categorical([50, 100]),
                'batch_size': Categorical([32, 64]) }

# Create bayesian optimization
model_bayes = KerasClassifier(build_fn=create_model, verbose=0)
bayes = BayesSearchCV(estimator=model_bayes, search_spaces=params_bayes, n_jobs=-1, cv=5, n_iter=30, random_state= 0)
# Fit bayes search
start = time.time()
bayes_result = bayes.fit(X_train, Y_train)
end = time.time()

# Bayesian optimization summarize results
print("Best: %f using %s" % (bayes_result.best_score_, bayes_result.best_params_))
print('Bayesian optimization takes {:.2f} seconds to tune'.format(end - start))
means = bayes_result.cv_results_['mean_test_score']
stds = bayes_result.cv_results_['std_test_score']
params = bayes_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))

```

Gambar 4.8 Code Bayesian optimization

Pada penelitian ini, dilakukan 20 kali percobaan menggunakan *Bayesian optimization* untuk menyetel *hyperparameter deep neural network* dalam mendeteksi penyakit jantung. Hasil kombinasi *hyperparameter* yang optimal dan waktu penyetelan teknik *Bayesian optimization* pada 20 kali percobaan ditunjukkan pada Tabel 4.8.

Tabel 4.8 Hasil Bayesian optimization

No.	Jumlah hidden layer	Jumlah hidden units	Dropout rate	Epochs	Ukuran batch	Waktu penyetelan (detik)
1.	3	8	0,2	100	64	176,57
2.	3	32	0,5	50	32	173,25
3.	2	8	0,2	50	32	208,7
4.	2	8	0,2	100	64	184,53
5.	4	16	0,2	50	64	174,96
6.	2	8	0,2	100	64	178,44
7.	2	32	0,2	50	64	162,48
8.	3	16	0,5	100	64	179,87
9.	2	8	0,2	100	64	179,68
10.	2	8	0,2	100	64	175,08
11.	3	16	0,5	50	64	179,27
12.	2	32	0,2	50	32	190,63
13.	2	32	0,2	50	64	185,44
14.	2	64	0,2	50	64	163,01
15.	3	16	0,2	50	64	167,75
16.	5	8	0,2	100	32	173,67

No.	Jumlah <i>hidden layer</i>	Jumlah <i>hidden units</i>	<i>Dropout rate</i>	<i>Epochs</i>	Ukuran <i>batch</i>	Waktu penyetelan (detik)
17.	2	32	0,5	50	64	184,51
18.	2	8	0,5	100	64	173,58
19.	3	16	0,2	50	64	176,15
20.	2	32	0,5	50	64	166,69
<b>Rata-rata</b>						<b>177,71</b>

Dari Tabel 4.8 diketahui kombinasi *hyperparameter* optimal untuk DNN menggunakan *Bayesian optimization* pada 20 kali percobaan. Diketahui pada hasil 20 kali percobaan menggunakan *Bayesian optimization*, jumlah *hidden layer* optimal yang paling sering muncul adalah 2, jumlah *hidden units* optimal yang paling sering muncul adalah 8, *dropout rate* optimal yang paling sering muncul adalah 0,2, *epochs* optimal yang paling sering muncul adalah 50, dan ukuran *batch* optimal yang paling sering muncul adalah 64. Dari 20 kali percobaan, rata-rata waktu penyetelan oleh *Bayesian optimization* adalah 177,71 detik atau 2,96 menit. Model DNN dengan *hyperparameter* optimal dari *random search* kemudian diuji pada *dataset testing* untuk mengetahui kinerjanya dalam mendeteksi penyakit jantung.

### 4.3 Klasifikasi

Pada pengembangan model *deep neural network* dengan penyetelan *hyperparameter* untuk mendeteksi penyakit jantung didapatkan kombinasi *hyperparameter* yang optimal berdasarkan penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization*. Kemudian dilakukan klasifikasi menggunakan model DNN dengan *hyperparameter* optimal dari keempat teknik penyetelan *hyperparameter* ini pada *dataset testing*. Pengujian dilakukan 20 kali pada 20 *dataset testing* yang memiliki urutan *instance* yang berbeda-beda. Tabel 4.9 – 4.12 menunjukkan *confusion matrix* hasil klasifikasi dari model *deep neural network* dengan penyetelan *hyperparameter* manual, *grid search*, *random search*, dan *Bayesian optimization*.

Tabel 4.9 *Confusion matrix* DNN dengan penyetelan manual

No.	TN	FP	FN	TP
1.	29	6	3	22
2.	28	4	5	23
3.	31	3	5	21
4.	30	4	4	22
5.	28	6	3	23
6.	32	3	3	22
7.	22	5	4	29
8.	30	5	3	22
9.	32	5	2	21
10.	30	6	2	22
11.	25	4	6	25
12.	32	3	7	18
13.	33	3	5	19
14.	32	2	6	20
15.	28	2	8	22
16.	30	6	5	19
17.	30	5	5	20
18.	28	3	6	23
19.	26	6	5	23
20.	22	6	4	28
<b>Jumlah</b>	<b>578</b>	<b>87</b>	<b>91</b>	<b>444</b>

Tabel 4.9 menunjukkan komposisi nilai *confusion matrix* menggunakan metode DNN dengan penyetelan manual berupa nilai TN (*True Negative*), FP (*False Positive*), FN (*False Negative*), TP (*True Positive*) dari 20 kali pengujian. Nilai TN, FP, FN, dan TP masing-masing pengujian tersebut digunakan untuk menghitung nilai akurasi, sensitivitas, presisi, spesifisitas, dan skor F1 yang digunakan untuk mendeteksi penyakit jantung menggunakan metode DNN dengan penyetelan manual.

Tabel 4.10 *Confusion matrix* DNN dengan *grid search*

No.	TN	FP	FN	TP
1.	32	3	4	21
2.	28	4	4	24
3.	31	3	5	21
4.	29	5	5	21
5.	30	4	5	21
6.	31	4	4	21
7.	22	5	7	26

No.	TN	FP	FN	TP
8.	29	6	2	23
9.	32	5	3	20
10.	30	6	1	23
11.	24	5	5	26
12.	32	3	4	21
13.	33	3	5	19
14.	32	2	5	21
15.	27	3	7	23
16.	30	6	3	21
17.	29	6	4	21
18.	27	4	6	23
19.	29	3	6	22
20.	23	5	5	27
<b>Jumlah</b>	<b>580</b>	<b>85</b>	<b>90</b>	<b>445</b>

Tabel 4.10 menunjukkan komposisi nilai *confusion matrix* menggunakan metode DNN dengan *grid search* berupa nilai TN (*True Negative*), FP (*False Positive*), FN (*False Negative*), TP (*True Positive*) dari 20 kali pengujian. Nilai TN, FP, FN, dan TP masing-masing pengujian tersebut digunakan untuk menghitung nilai akurasi, sensitivitas, presisi, spesifisitas, dan skor F1 yang digunakan untuk mendeteksi penyakit jantung menggunakan metode DNN dengan *grid search*.

Tabel 4.11 *Confusion matrix* DNN dengan *random search*

No.	TN	FP	FN	TP
1.	30	5	4	21
2.	27	5	4	24
3.	32	2	7	19
4.	28	6	3	23
5.	32	2	6	20
6.	32	3	6	19
7.	22	5	2	31
8.	32	3	2	23
9.	33	4	2	21
10.	32	4	3	21
11.	23	6	4	27
12.	30	5	5	20
13.	33	3	3	21
14.	31	3	6	20
15.	27	3	7	23
16.	28	8	3	21
17.	31	4	5	20

No.	TN	FP	FN	TP
18.	25	6	4	25
19.	23	9	4	24
20.	21	7	4	28
<b>Jumlah</b>	<b>572</b>	<b>93</b>	<b>84</b>	<b>451</b>

Tabel 4.11 menunjukkan komposisi nilai *confusion matrix* menggunakan metode DNN dengan *random search* berupa nilai TN (*True Negative*), FP (*False Positive*), FN (*False Negative*), TP (*True Positive*) dari 20 kali pengujian. Nilai TN, FP, FN, dan TP masing-masing pengujian tersebut digunakan untuk menghitung nilai akurasi, sensitivitas, presisi, spesifisitas, dan skor F1 yang digunakan untuk mendeteksi penyakit jantung menggunakan metode DNN dengan *random search*.

Tabel 4.12 *Confusion matrix* DNN dengan *Bayesian optimization*

No.	TN	FP	FN	TP
1.	33	2	4	21
2.	28	4	4	24
3.	32	2	5	21
4.	30	4	4	22
5.	31	3	3	23
6.	33	2	5	20
7.	21	6	3	30
8.	31	4	2	23
9.	33	4	2	21
10.	32	4	0	24
11.	23	6	3	28
12.	31	4	4	21
13.	34	2	4	20
14.	31	3	5	21
15.	27	3	4	26
16.	33	3	4	20
17.	31	4	4	21
18.	28	3	5	24
19.	27	5	4	24
20.	25	3	4	28
<b>Jumlah</b>	<b>594</b>	<b>71</b>	<b>73</b>	<b>462</b>

Tabel 4.12 menunjukkan komposisi nilai *confusion matrix* menggunakan metode DNN dengan *Bayesian optimization* berupa nilai TN (*True Negative*), FP (*False Positive*), FN (*False Negative*), TP (*True Positive*) dari 20 kali pengujian. Nilai TN, FP, FN, dan TP masing-masing pengujian tersebut digunakan untuk

menghitung nilai akurasi, sensitivitas, presisi, spesifisitas, dan skor F1 yang digunakan untuk mendeteksi penyakit jantung menggunakan metode DNN dengan *Bayesian optimization*.

Perbandingan kombinasi *hyperparameter* dan akurasi klasifikasi model DNN dengan penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization* pada 20 kali pengujian dapat dilihat pada Tabel 4.13.

Tabel 4.13 Kombinasi *hyperparameter* dan akurasi klasifikasi model DNN dengan penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization*

No.	Penyetelan Manual		Grid search		Random search		Bayesian optimization	
	Kombinasi Hyperparameter	Akurasi (%)	Kombinasi Hyperparameter	Akurasi (%)	Kombinasi Hyperparameter	Akurasi (%)	Kombinasi Hyperparameter	Akurasi (%)
1.	Hidden layer= 2, hidden units= 32, dropout rate= 0,2, epochs= 50, ukuran batch= 32	85,00	Hidden layer= 3, hidden units= 8, dropout rate= 0,2, epochs= 50, ukuran batch= 64	88,33	Hidden layer= 2, hidden units= 32, dropout rate= 0,2, epochs= 50, ukuran batch= 32	85,00	Hidden layer= 3, hidden units= 8, dropout rate= 0,2, epochs= 100, ukuran batch= 64	90,00
2.	Hidden layer= 2, hidden units= 8, dropout rate= 0,2, epochs= 50, ukuran batch= 32	85,00	Hidden layer= 3, hidden units= 16, dropout rate= 0,5, epochs= 50, ukuran batch= 32	86,67	Hidden layer= 2, hidden units= 8, dropout rate= 0,2, epochs= 100, ukuran batch= 32	85,00	Hidden layer= 3, hidden units= 32, dropout rate= 0,5, epochs= 50, ukuran batch= 32	86,67
3.	Hidden layer= 2, hidden units= 32, dropout rate= 0,5, epochs= 100, ukuran batch= 64	86,67	Hidden layer= 4, hidden units= 16, dropout rate= 0,5, epochs= 50, ukuran batch= 64	86,67	Hidden layer= 4, hidden units= 16, dropout rate= 0,2, epochs= 50, ukuran batch= 64	85,00	Hidden layer= 2, hidden units= 8, dropout rate= 0,2, epochs= 50, ukuran batch= 32	88,33
4.	Hidden layer= 2, hidden units= 32, dropout rate= 0,5, epochs= 50, ukuran batch= 32	86,67	Hidden layer= 2, hidden units= 16, dropout rate= 0,5, epochs= 50, ukuran batch= 32	83,33	Hidden layer= 5, hidden units= 8, dropout rate= 0,5, epochs= 100, ukuran batch= 32	85,00	Hidden layer= 2, hidden units= 8, dropout rate= 0,2, epochs= 100, ukuran batch= 64	86,67
5.	Hidden layer= 3, hidden units= 16, dropout rate= 0,2, epochs= 50, ukuran batch= 32	85,00	Hidden layer= 3, hidden units= 16, dropout rate= 0,2, epochs= 100, ukuran batch= 64	85,00	Hidden layer= 2, hidden units= 8, dropout rate= 0,2, epochs= 100, ukuran batch= 32	86,67	Hidden layer= 4, hidden units= 16, dropout rate= 0,2, epochs= 50, ukuran batch= 64	90,00
6.	Hidden layer= 5, hidden units= 8, dropout rate= 0,2, epochs= 50, ukuran batch= 32	90,00	Hidden layer= 5, hidden units= 16, dropout rate= 0,5, epochs= 100, ukuran batch= 64	86,67	Hidden layer= 4, hidden units= 32, dropout rate= 0,2, epochs= 100, ukuran batch= 32	85,00	Hidden layer= 2, hidden units= 8, dropout rate= 0,2, epochs= 100, ukuran batch= 64	88,33
7.	Hidden layer= 3, hidden units= 32, dropout rate= 0,5, epochs= 50, ukuran batch= 64	85,00	Hidden layer= 5, hidden units= 8, dropout rate= 0,5, epochs= 100, ukuran batch= 32	80,00	Hidden layer= 2, hidden units= 16, dropout rate= 0,5, epochs= 50, ukuran batch= 32	88,33	Hidden layer= 2, hidden units= 32, dropout rate= 0,2, epochs= 50, ukuran batch= 64	85,00
8.	Hidden layer= 2, hidden units= 16, dropout rate= 0,2, epochs= 50, ukuran batch= 32	86,67	Hidden layer= 5, hidden units= 16, dropout rate= 0,5, epochs= 50, ukuran batch= 32	86,67	Hidden layer= 3, hidden units= 16, dropout rate= 0,5, epochs= 50, ukuran batch= 64	91,67	Hidden layer= 3, hidden units= 16, dropout rate= 0,5, epochs= 100, ukuran batch= 64	90,00
9.	Hidden layer= 2, hidden units= 8, dropout rate= 0,2, epochs= 50, ukuran batch= 32	88,33	Hidden layer= 2, hidden units= 8, dropout rate= 0,2, epochs= 100, ukuran batch= 64	86,67	Hidden layer= 4, hidden units= 8, dropout rate= 0,2, epochs= 50, ukuran batch= 32	90,00	Hidden layer= 2, hidden units= 8, dropout rate= 0,2, epochs= 100, ukuran batch= 64	90,00

No.	Penyetelan Manual		Grid search		Random search		Bayesian optimization	
	Kombinasi Hyperparameter	Akurasi (%)	Kombinasi Hyperparameter	Akurasi (%)	Kombinasi Hyperparameter	Akurasi (%)	Kombinasi Hyperparameter	Akurasi (%)
10.	Hidden layer= 2, hidden units= 64, dropout rate= 0,2, epochs= 50, ukuran batch= 64	86,67	Hidden layer= 4, hidden units= 8, dropout rate= 0,2, epochs= 50, ukuran batch= 32	88,33	Hidden layer= 2, hidden units= 8, dropout rate= 0,5, epochs= 100, ukuran batch= 32	88,33	Hidden layer= 2, hidden units= 8, dropout rate= 0,2, epochs= 100, ukuran batch= 64	93,33
11.	Hidden layer= 2, hidden units= 32, dropout rate= 0,5, epochs= 50, ukuran batch= 32	83,33	Hidden layer= 3, hidden units= 16, dropout rate= 0,2, epochs= 50, ukuran batch= 64	83,33	Hidden layer= 2, hidden units= 16, dropout rate= 0,5, epochs= 50, ukuran batch= 32	83,33	Hidden layer= 3, hidden units= 16, dropout rate= 0,5, epochs= 50, ukuran batch= 64	85,00
12.	Hidden layer= 4, hidden units= 16, dropout rate= 0,2, epochs= 50, ukuran batch= 32	83,33	Hidden layer= 3, hidden units= 16, dropout rate= 0,2, epochs= 50, ukuran batch= 64	88,33	Hidden layer= 2, hidden units= 32, dropout rate= 0,5, epochs= 100, ukuran batch= 32	83,33	Hidden layer= 2, hidden units= 32, dropout rate= 0,2, epochs= 50, ukuran batch= 32	86,67
13.	Hidden layer= 2, hidden units= 8, dropout rate= 0,2, epochs= 100, ukuran batch= 64	86,67	Hidden layer= 2, hidden units= 16, dropout rate= 0,2, epochs= 100, ukuran batch= 64	86,67	Hidden layer= 4, hidden units= 16, dropout rate= 0,2, epochs= 50, ukuran batch= 64	90,00	Hidden layer= 2, hidden units= 32, dropout rate= 0,2, epochs= 50, ukuran batch= 64	90,00
14.	Hidden layer= 2, hidden units= 16, dropout rate= 0,5, epochs= 100, ukuran batch= 64	86,67	Hidden layer= 3, hidden units= 16, dropout rate= 0,2, epochs= 50, ukuran batch= 64	88,33	Hidden layer= 4, hidden units= 16, dropout rate= 0,2, epochs= 50, ukuran batch= 64	85,00	Hidden layer= 2, hidden units= 64, dropout rate= 0,2, epochs= 50, ukuran batch= 64	86,67
15.	Hidden layer= 3, hidden units= 16, dropout rate= 0,2, epochs= 50, ukuran batch= 32	83,33	Hidden layer= 3, hidden units= 16, dropout rate= 0,5, epochs= 50, ukuran batch= 32	83,33	Hidden layer= 2, hidden units= 32, dropout rate= 0,5, epochs= 100, ukuran batch= 32	83,33	Hidden layer= 3, hidden units= 16, dropout rate= 0,2, epochs= 50, ukuran batch= 64	88,33
16.	Hidden layer= 3, hidden units= 16, dropout rate= 0,2, epochs= 50, ukuran batch= 64	81,67	Hidden layer= 2, hidden units= 64, dropout rate= 0,5, epochs= 50, ukuran batch= 64	85,00	Hidden layer= 4, hidden units= 16, dropout rate= 0,5, epochs= 100, ukuran batch= 64	81,67	Hidden layer= 5, hidden units= 8, dropout rate= 0,2, epochs= 100, ukuran batch= 32	88,33
17.	Hidden layer= 2, hidden units= 64, dropout rate= 0,5, epochs= 100, ukuran batch= 64	83,33	Hidden layer= 3, hidden units= 32, dropout rate= 0,5, epochs= 50, ukuran batch= 64	83,33	Hidden layer= 4, hidden units= 16, dropout rate= 0,2, epochs= 50, ukuran batch= 64	85,00	Hidden layer= 2, hidden units= 32, dropout rate= 0,5, epochs= 50, ukuran batch= 64	86,67
18.	Hidden layer= 3, hidden units= 16, dropout rate= 0,5, epochs= 50, ukuran batch= 32	85,00	Hidden layer= 5, hidden units= 16, dropout rate= 0,2, epochs= 50, ukuran batch= 32	83,33	Hidden layer= 3, hidden units= 16, dropout rate= 0,5, epochs= 50, ukuran batch= 64	83,33	Hidden layer= 2, hidden units= 8, dropout rate= 0,5, epochs= 100, ukuran batch= 64	86,67
19.	Hidden layer= 3, hidden units= 32, dropout rate= 0,5, epochs= 50, ukuran batch= 32	81,67	Hidden layer= 2, hidden units= 16, dropout rate= 0,5, epochs= 50, ukuran batch= 64	85,00	Hidden layer= 4, hidden units= 8, dropout rate= 0,2, epochs= 50, ukuran batch= 32	78,33	Hidden layer= 3, hidden units= 16, dropout rate= 0,2, epochs= 50, ukuran batch= 64	85,00
20.	Hidden layer= 2, hidden units= 32, dropout rate= 0,2, epochs= 50, ukuran batch= 64	83,33	Hidden layer= 2, hidden units= 16, dropout rate= 0,2, epochs= 50, ukuran batch= 32	83,33	Hidden layer= 2, hidden units= 32, dropout rate= 0,2, epochs= 50, ukuran batch= 32	81,67	Hidden layer= 2, hidden units= 32, dropout rate= 0,5, epochs= 50, ukuran batch= 64	88,33

Pada Tabel 4.13 dapat dilihat bahwa untuk DNN dengan penyetelan manual mendapatkan akurasi paling tinggi 90% pada pengujian ke 6 dengan kombinasi hyperparameter hidden layer= 5, hidden units= 8, dropout rate= 0,2, epochs= 50, ukuran batch= 32. DNN dengan grid search mendapatkan akurasi paling tinggi

88,33% pada pengujian ke 1 dengan kombinasi *hyperparameter hidden layer= 3, hidden units= 8, dropout rate= 0,2, epochs= 50*, ukuran *batch= 64*, pengujian ke 10 dengan kombinasi *hyperparameter hidden layer= 4, hidden units= 8, dropout rate= 0,2, epochs=50*, ukuran *batch= 32*, pengujian ke 12 dengan kombinasi *hyperparameter hidden layer= 3, hidden units= 16, dropout rate= 0,2, epochs= 50*, ukuran *batch=64*, dan pengujian ke 14 dengan kombinasi *hyperparameter hidden layer= 3, hidden units= 16, dropout rate= 0,2, epochs= 50*, ukuran *batch= 64*. DNN dengan *random search* mendapatkan akurasi tertinggi 91,67% pada pengujian ke 8 dengan kombinasi *hyperparameter hidden layer= 3, hidden units= 16, dropout rate= 0,5, epochs= 50*, ukuran *batch= 64*. Sedangkan, DNN dengan *Bayesian optimization* mendapatkan akurasi tertinggi 93,33% pada pengujian ke 10 dengan kombinasi *hyperparameter hidden layer= 2, hidden units= 8, dropout rate= 0,2, epochs= 100*, ukuran *batch= 64*.

#### 4.4 Evaluasi

Penerapan klasifikasi pada *dataset testing* menggunakan model *deep neural network* dengan penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization* telah dilakukan dengan baik. Dari hasil *confusion matrix* berupa nilai TP, TN, FP dan FN masing-masing teknik digunakan untuk menghitung nilai akurasi, sensitivitas, presisi, spesitivitas, dan skor F1.

Akurasi mengukur tingkat keberhasilan model dalam mengklasifikasikan pasien sehat dan sakit dengan benar dari keseluruhan *dataset* yang ada. Perbandingan akurasi DNN dengan penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization* dari 20 kali pengujian dapat dilihat pada Tabel 4.14.

Tabel 4.14 Perbandingan akurasi model DNN dengan penyetelan manual, *grid search*, *random search*, dan *bayesian optimization*

No.	Penyetelan Manual	<i>Grid search</i>	<i>Random search</i>	<i>Bayesian optimization</i>
1.	85,00%	88,33%	85,00%	90,00%
2.	85,00%	86,67%	85,00%	86,67%
3.	86,67%	86,67%	85,00%	88,33%
4.	86,67%	83,33%	85,00%	86,67%

No.	Penyetelan Manual	<i>Grid search</i>	<i>Random search</i>	<i>Bayesian optimization</i>
5.	85,00%	85,00%	86,67%	90,00%
6.	90,00%	86,67%	85,00%	88,33%
7.	85,00%	80,00%	88,33%	85,00%
8.	86,67%	86,67%	91,67%	90,00%
9.	88,33%	86,67%	90,00%	90,00%
10.	86,67%	88,33%	88,33%	93,33%
11.	83,33%	83,33%	83,33%	85,00%
12.	83,33%	88,33%	83,33%	86,67%
13.	86,67%	86,67%	90,00%	90,00%
14.	86,67%	88,33%	85,00%	86,67%
15.	83,33%	83,33%	83,33%	88,33%
16.	81,67%	85,00%	81,67%	88,33%
17.	83,33%	83,33%	85,00%	86,67%
18.	85,00%	83,33%	83,33%	86,67%
19.	81,67%	85,00%	78,33%	85,00%
20.	83,33%	83,33%	81,67%	88,33%
<b>Rata-rata</b>	<b>85,17%</b>	<b>85,42%</b>	<b>85,25%</b>	<b>88,00%</b>

Dari Tabel 4.14 diketahui perbandingan akurasi model DNN dengan penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization* dalam mendeteksi penyakit jantung. Model DNN dengan *Bayesian optimization* memiliki rata-rata akurasi lebih tinggi dibandingkan penyetelan manual, *grid search*, dan *random search*.

Sensitivitas atau *recall* mengukur sebaik apa model dalam mendeteksi pasien penyakit jantung dengan benar dari keseluruhan pasien yang benar mengidap penyakit jantung. Perbandingan sensitivitas DNN dengan penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization* dari 20 kali pengujian dapat dilihat pada Tabel 4.15.

Tabel 4.15 Perbandingan sensitivitas model DNN dengan penyetelan manual, *grid search*, *random search*, dan *bayesian optimization*

No.	Penyetelan Manual	<i>Grid search</i>	<i>Random search</i>	<i>Bayesian optimization</i>
1.	88,00%	84,00%	84,00%	84,00%
2.	82,14%	85,71%	85,71%	85,71%
3.	80,77%	80,77%	73,08%	80,77%
4.	84,62%	80,77%	88,46%	84,62%
5.	88,46%	80,77%	76,92%	88,46%

No.	Penyetelan Manual	<i>Grid search</i>	<i>Random search</i>	<i>Bayesian optimization</i>
6.	88,00%	84,00%	76,00%	80,00%
7.	87,88%	78,79%	93,94%	90,91%
8.	88,00%	92,00%	92,00%	92,00%
9.	91,30%	86,96%	91,30%	91,30%
10.	91,67%	95,83%	87,50%	100,00%
11.	80,65%	83,87%	87,10%	90,32%
12.	72,00%	84,00%	80,00%	84,00%
13.	79,17%	79,17%	87,50%	83,33%
14.	76,92%	80,77%	76,92%	80,77%
15.	73,33%	76,67%	76,67%	86,67%
16.	79,17%	87,50%	87,50%	83,33%
17.	80,00%	84,00%	80,00%	84,00%
18.	79,31%	79,31%	86,21%	82,76%
19.	82,14%	78,57%	85,71%	85,71%
20.	87,50%	84,38%	87,50%	87,50%
<b>Rata-rata</b>	<b>83,05%</b>	<b>83,39%</b>	<b>84,20%</b>	<b>86,31%</b>

Dari Tabel 4.15 diketahui perbandingan sensitivitas model DNN dengan penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization* dalam mendeteksi penyakit jantung. Model DNN dengan *Bayesian optimization* memiliki rata-rata sensitivitas lebih tinggi dibandingkan penyetelan manual, *grid search*, dan *random search*.

Presisi mengukur sebaik apa model dalam mendeteksi pasien yang benar mengidap penyakit jantung dari keseluruhan pasien yang diprediksi pasien penyakit jantung. Perbandingan presisi DNN dengan penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization* dari 20 kali pengujian dapat dilihat pada Tabel 4.16.

Tabel 4.16 Perbandingan presisi model DNN dengan penyetelan manual, *grid search*, *random search*, dan *bayesian optimization*

No.	Penyetelan Manual	<i>Grid search</i>	<i>Random search</i>	<i>Bayesian optimization</i>
1.	78,57%	87,50%	80,77%	91,30%
2.	85,19%	85,71%	82,76%	85,71%
3.	87,50%	87,50%	90,48%	91,30%
4.	84,62%	80,77%	79,31%	84,62%
5.	79,31%	84,00%	90,91%	88,46%
6.	88,00%	84,00%	86,36%	90,91%

No.	Penyetelan Manual	<i>Grid search</i>	<i>Random search</i>	<i>Bayesian optimization</i>
7.	85,29%	83,87%	86,11%	83,33%
8.	81,48%	79,31%	88,46%	85,19%
9.	80,77%	80,00%	84,00%	84,00%
10.	78,57%	79,31%	84,00%	85,71%
11.	86,21%	83,87%	81,82%	82,35%
12.	85,71%	87,50%	80,00%	84,00%
13.	86,36%	86,36%	87,50%	90,91%
14.	90,91%	91,30%	86,96%	87,50%
15.	91,67%	88,46%	88,46%	89,66%
16.	76,00%	77,78%	72,41%	86,96%
17.	80,00%	77,78%	83,33%	84,00%
18.	88,46%	85,19%	80,65%	88,89%
19.	79,31%	88,00%	72,73%	82,76%
20.	82,35%	84,38%	80,00%	90,32%
<b>Rata-rata</b>	<b>83,81%</b>	<b>84,13%</b>	<b>83,35%</b>	<b>86,89%</b>

Dari Tabel 4.16 diketahui perbandingan presisi model DNN dengan penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization* dalam mendeteksi penyakit jantung. Model DNN dengan *Bayesian optimization* memiliki rata-rata presisi lebih tinggi dibandingkan penyetelan manual, *grid search*, dan *random search*.

Spesifisitas mengukur sebaik apa model dalam mendeteksi pasien sehat dengan benar dari keseluruhan pasien yang benar sehat. Perbandingan spesifisitas DNN dengan penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization* dari 20 kali pengujian dapat dilihat pada Tabel 4.17.

Tabel 4.17 Perbandingan spesifisitas model DNN dengan penyetelan manual, *grid search*, *random search*, dan *bayesian optimization*

No.	Penyetelan Manual	<i>Grid search</i>	<i>Random search</i>	<i>Bayesian optimization</i>
1.	82,86%	91,43%	85,71%	94,29%
2.	87,50%	87,50%	84,38%	87,50%
3.	91,18%	91,18%	94,12%	94,12%
4.	88,24%	85,29%	82,35%	88,24%
5.	82,35%	88,24%	94,12%	91,18%
6.	91,43%	88,57%	91,43%	94,29%
7.	81,48%	81,48%	81,48%	77,78%
8.	85,71%	82,86%	91,43%	88,57%
9.	86,49%	86,49%	89,19%	89,19%

No.	Penyetelan Manual	Grid search	Random search	Bayesian optimization
10.	83,33%	83,33%	88,89%	88,89%
11.	86,21%	82,76%	79,31%	79,31%
12.	91,43%	91,43%	85,71%	88,57%
13.	91,67%	91,67%	91,67%	94,44%
14.	94,12%	94,12%	91,18%	91,18%
15.	93,33%	90,00%	90,00%	90,00%
16.	83,33%	83,33%	77,78%	91,67%
17.	85,71%	82,86%	88,57%	88,57%
18.	90,32%	87,10%	80,65%	90,32%
19.	81,25%	90,63%	71,88%	84,38%
20.	78,57%	82,14%	75,00%	89,29%
<b>Rata-rata</b>	<b>86,83%</b>	<b>87,12%</b>	<b>85,74%</b>	<b>89,09%</b>

Dari Tabel 4.17 diketahui perbandingan spesifisitas model DNN dengan penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization* dalam mendeteksi penyakit jantung. Model DNN dengan *Bayesian optimization* memiliki rata-rata spesifisitas lebih tinggi dibandingkan penyetelan manual, *grid search*, dan *random search*.

Skor F1 mengukur keseimbangan presisi dan *recall* pada model. Skor F1 yang tinggi menunjukkan bahwa model memiliki nilai presisi dan *recall* yang baik. Perbandingan skor F1 DNN dengan penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization* dari 20 kali pengujian dapat dilihat pada Tabel 4.18.

Tabel 4.18 Perbandingan skor F1 model DNN dengan penyetelan manual, *grid search*, *random search*, dan *bayesian optimization*

No.	Penyetelan Manual	Grid search	Random search	Bayesian optimization
1.	83,02%	85,71%	82,35%	87,50%
2.	83,64%	85,71%	84,21%	85,71%
3.	84,00%	84,00%	80,85%	85,71%
4.	84,62%	80,77%	83,64%	84,62%
5.	83,64%	82,35%	83,33%	88,46%
6.	88,00%	84,00%	80,85%	85,11%
7.	86,57%	81,25%	89,86%	86,96%
8.	84,62%	85,19%	90,20%	88,46%
9.	85,71%	83,33%	87,50%	87,50%
10.	84,62%	86,79%	85,71%	92,31%
11.	83,33%	83,87%	84,38%	86,15%
12.	78,26%	85,71%	80,00%	84,00%

No.	Penyetelan Manual	<i>Grid search</i>	<i>Random search</i>	<i>Bayesian optimization</i>
13.	82,61%	82,61%	87,50%	86,96%
14.	83,33%	85,71%	81,63%	84,00%
15.	81,48%	82,14%	82,14%	88,14%
16.	77,55%	82,35%	79,25%	85,11%
17.	80,00%	80,77%	81,63%	84,00%
18.	83,64%	82,14%	83,33%	85,71%
19.	80,70%	83,02%	78,69%	84,21%
20.	84,85%	84,38%	83,58%	88,89%
<b>Rata-rata</b>	<b>83,21%</b>	<b>83,59%</b>	<b>83,53%</b>	<b>86,48%</b>

Dari Tabel 4.18 diketahui perbandingan skor F1 model DNN dengan penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization* dalam mendeteksi penyakit jantung. Model DNN dengan *Bayesian optimization* memiliki rata-rata skor F1 lebih tinggi dibandingkan penyetelan manual, *grid search*, dan *random search*.

Rangkuman perbandingan kinerja teknik penyetelan *hyperparameter* manual, *grid search*, *random search*, dan *Bayesian optimization* untuk model DNN dalam mendeteksi penyakit jantung dapat dilihat pada Tabel 4.19.

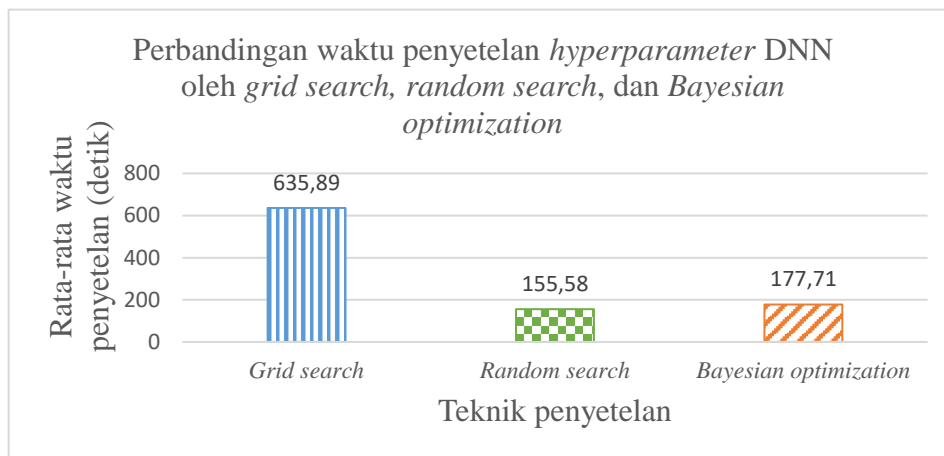
Tabel 4.19 Rangkuman perbandingan kinerja model DNN dengan penyetelan manual, *grid search*, *random search*, dan *bayesian optimization*

Teknik penyetelan	Akurasi	Sensitivitas	Presisi	Spesifisitas	Skor F1
Manual	85,17%	83,05%	83,81%	86,83%	83,21%
<i>Grid search</i>	85,42%	83,39%	84,13%	87,12%	83,59%
<i>Random search</i>	85,25%	84,20%	83,35%	85,74%	83,53%
<i>Bayesian optimization</i>	88,00%	86,31%	86,89%	89,09%	86,48%

Dari Tabel 4.19, diketahui bahwa DNN dengan penyetelan *hyperparameter* menggunakan *Bayesian optimization* menghasilkan akurasi, sensitivitas, presisi, spesifisitas, dan skor F1 lebih tinggi dibandingkan penyetelan manual, *grid search*, dan *random search* dalam mendeteksi penyakit jantung. *Bayesian optimization* mempertimbangkan evaluasi sebelumnya saat memilih set *hyperparameter* untuk evaluasi berikutnya, sehingga *hyperparameter* yang dipilih adalah *hyperparameter*

yang dapat meningkatkan akurasi model. Dari teknik-teknik yang digunakan, nilai spesifisitas selalu lebih tinggi dibandingkan nilai sensitivitas karena jumlah kelas negatif lebih banyak daripada jumlah kelas positif pada *dataset*. Hal tersebut memungkinkan klasifikasi kelas negatif dengan benar lebih besar dibandingkan klasifikasi untuk kelas positif dengan benar.

Perbandingan rata-rata waktu penyetelan *hyperparameter* oleh teknik penyetelan otomatis *grid search*, *random search*, dan *Bayesian optimization* untuk model DNN dalam mendeteksi penyakit jantung dapat dilihat pada Gambar 4.9.



Gambar 4.9 Perbandingan waktu penyetelan *hyperparameter* DNN oleh *grid search*, *random search*, dan *Bayesian optimization*

Berdasarkan Gambar 4.9, diketahui bahwa *random search* membutuhkan waktu lebih singkat dibandingkan *grid search* dan *Bayesian optimization* dalam menyetel *hyperparameter* DNN. *Random search* menghabiskan rata-rata 155,58 detik dalam menyetel *hyperparameter* DNN. Sedangkan, *Bayesian optimization* membutuhkan waktu rata-rata 177,71 detik dan *grid search* dengan rata-rata 635,89 detik dalam menyetel *hyperparameter* DNN. *Grid search* menghabiskan waktu paling lama karena mengevaluasi semua kombinasi *hyperparameter* yang ada. *Bayesian optimization* membutuhkan waktu yang lebih lama dibandingkan *random search* karena metode dalam menyetel *hyperparameter* lebih kompleks. Sedangkan, metode penyetelan *hyperparameter* oleh *random search* lebih sederhana, sehingga

menghabiskan waktu yang lebih singkat. Meskipun, *Bayesian optimization* membutuhkan waktu lebih lama dibanding *random search*, tapi selisih waktu hanya sedikit, yaitu 22,13 detik. Dengan mempertimbangkan hasil akurasi kinerja dan kecepatan waktu penyetelan, *Bayesian optimization* lebih dipilih sebagai metode penyetelan *hyperparameter* untuk DNN dalam mendeteksi penyakit jantung.

## BAB V KESIMPULAN DAN SARAN

### 5.1 Kesimpulan

Dengan tingginya angka kematian di dunia akibat penyakit jantung, menyebabkan perlunya mendeteksi penyakit jantung sedini mungkin. Dengan mendeteksi penyakit jantung pada tahap awal dari gejala pada pasien, dapat membantu pasien mendapatkan pengobatan lebih dini. Banyak peneliti telah memanfaatkan perkembangan teknologi informasi untuk mengembangkan sistem diagnosis penyakit jantung berbantuan komputer. Salah satu teknik yang mendapatkan perhatian dalam dekade terakhir ini adalah *deep learning*. Pada penelitian ini, dilakukan teknik *deep learning* menggunakan arsitektur *deep neural network* (DNN) untuk mendeteksi penyakit jantung. Arsitektur *deep neural network* terdiri atas *input layer*, *hidden layer*, dan *output layer*. Dilakukan pula penyetelan *hyperparameter* untuk mendapatkan model dengan akurasi yang optimal. Penyetelan *hyperparameter* dilakukan dengan menggunakan penyetelan manual, *grid search*, *random search*, dan *Bayesian optimization*. Berdasarkan hasil penelitian yang diperoleh, dapat diambil kesimpulan sebagai berikut:

1. Penyetelan *hyperparameter* DNN menggunakan *Bayesian optimization* mendapatkan hasil akurasi lebih baik dibandingkan penyetelan manual, *grid search*, atau pun *random search* dalam mendeteksi penyakit jantung. Evaluasi kinerja DNN dengan *Bayesian optimization* pada 20 kali pengujian menghasilkan rata-rata akurasi 88%, sensitivitas 86,31%, presisi 86,89%, spesifisitas 89,09%, dan skor F1 86,48%.
2. *Bayesian optimization* dapat menyetel *hyperparameter* untuk DNN dalam mendeteksi penyakit jantung lebih cepat dari *grid search*, tapi lebih lambat dari *random search*. Meskipun, *Bayesian optimization* membutuhkan waktu lebih lama dibanding *random search*, tapi selisih waktu hanya sedikit. Dengan mempertimbangkan hasil akurasi kinerja dan kecepatan waktu penyetelan, *Bayesian optimization* lebih dipilih sebagai metode penyetelan *hyperparameter* untuk DNN dalam mendeteksi penyakit jantung.

## 5.2 Saran

Dari penelitian yang telah dilakukan, terdapat beberapa saran yang dapat digunakan untuk penelitian selanjutnya, adapun saran tersebut adalah sebagai berikut:

1. Pada penelitian ini, *Bayesian optimization* untuk menyetel *hyperparameter* DNN lebih lambat dari *random search* dikarenakan metodenya yang lebih kompleks. Oleh karena itu, pada penelitian selanjutnya dapat diteliti mengenai metode untuk meningkatkan kecepatan waktu penyetelan *hyperparameter* dengan tetap menghasilkan akurasi kinerja yang baik.

2. Pada penelitian ini, jumlah datanya masih terbatas sehingga untuk penelitian selanjutnya dapat dilakukan pengujian dengan jumlah data yang lebih besar.

## DAFTAR PUSTAKA

- [1] World Health Organization, "WHO," WHO, 17 May 2017. [Online]. Available: [https://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases-\(cvds\)](https://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)). [Accessed 10 09 2020].
- [2] World Health Organization, "WHO," WHO, 2018. [Online]. Available: [https://www.who.int/gho/mortality\\_burden\\_disease/causes\\_death/top\\_10/en/](https://www.who.int/gho/mortality_burden_disease/causes_death/top_10/en/). [Accessed 10 09 2020].
- [3] Institute of Medicine (US) Committee on Social Security Cardiovascular Disability Criteria, "Ischemic Heart Disease," in *Cardiovascular Disability: Updating the Social Security Listings*, Washington (DC), National Academies Press (US), 2010.
- [4] "GBD 2013 Mortality and Causes of Death Collaborators. Global, regional, and national age-sex specific all-cause and cause-specific mortality for 240 causes of death, 1990-2013: a systematic analysis for the Global Burden of Disease Study 2013," *The Lancet*, vol. 385, no. 9963, pp. 117-171, 2015.
- [5] A. Rodgers, C. M. M. Lawes, T. Gaziano and T. Vos, "The Growing Burden of Risk from High Blood Pressure, Cholesterol, and Bodyweight," in *Disease Control Priorities in Developing Countries. 2nd edition*, New York, Oxford University Press, 2006, pp. 851-868.
- [6] H. S. Buttar, T. Li and N. Ravi, "Prevention of cardiovascular diseases: Role of exercise, dietary interventions, obesity and smoking cessation," *Exp Clin Cardiol*, vol. 10, no. 4, pp. 229-249, 2005.
- [7] T. Davenport and R. Kalakota, "The potential for artificial intelligence in healthcare," *Future Healthcare Journal*, vol. 6, no. 2, pp. 94-98, 2019.
- [8] J. K. Kim and S. Kang, "Neural Network-Based Coronary Heart Disease Risk Prediction Using Feature Correlation Analysis," *Journal of Healthcare Engineering*, vol. 2017, 2017.
- [9] F. Gullo, "From Patterns in Data to Knowledge Discovery: What Data Mining Can Do," *Physics Procedia*, vol. 62, pp. 18-22, 2015.
- [10] X. Teng and Y. Gong, "Research on Application of Machine Learning in Data Mining," *IOP Conference Series: Materials Science and Engineering*, vol. 392, no. 6, 2018.
- [11] U. Maheswari and J. Jasmine, "Neural Network based Heart Disease Prediction," *International Journal of Engineering Research & Technology (IJERT)*, vol. 5, no. 17, 2017.
- [12] R. S. El-Sayed, "Linear Discriminant Analysis for An Efficient Diagnosis of Heart Disease via Attribute Filtering Based on Genetic Algorithm," *Journal of Computers*, vol. 13, no. 11, pp. 1290-1299, 2018.
- [13] J. Vijayashree and H. P. Sultana, "A Machine Learning Framework for Feature Selection in Heart Disease Classification Using Improved Particle Swarm Optimization with Support Vector Machine Classifier," *Program Comput Soft*, vol. 44, no. 6, p. 388–397, 2018.

- [14] A. Gupta, S. Yadav, S. Shahid and V. U., "HeartCare: IoT based heart disease prediction system," in *2019 International Conference on Information Technology (ICIT)*, Bhubaneswar, India, 2019.
- [15] S. Pouriyeh, S. Vahid, G. Sannino, D. G. Pietro, H. Arabnia and J. Gutierrez, "A comprehensive investigation and comparison of Machine Learning Techniques in the domain of heart disease," in *2017 IEEE Symposium on Computers and Communications (ISCC)*, Heraklion, 2017.
- [16] A. P. Pawlovsky, "An Ensemble Based on Distances for a kNN method for heart disease diagnosis," in *2018 International Conference on Electronics, Information, and Communication (ICEIC)*, Honolulu, HI, 2018.
- [17] C. B. C. Latha and S. C. Jeeva, "Improving the accuracy of prediction of heart disease risk based on ensemble classification techniques," *Informatics in Medicine Unlocked*, vol. 16, p. 100203, 2019.
- [18] A. Ed-daoudy and K. Maalmi, "Performance evaluation of machine learning based big data processing framework for prediction of heart disease," in *2019 International Conference on Intelligent Systems and Advanced Computing Sciences (ISACS)*, Taza, Morocco, 2019.
- [19] D. Normawati and S. Winarti, "Feature Selection with Combination Classifier use Rules-Based Data Mining for Diagnosis of Coronary Heart Disease," in *12th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, Yogyakarta, Indonesia, 2018.
- [20] S. Mohan, C. Thirumalai and G. Srivastava, "Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques," *IEEE Access*, vol. 7, pp. 81542-81554, 2019.
- [21] I. Tobore, J. Li, L. Yuhang, Y. Al-Handarish, A. Kandwal, Z. Nie and L. Wang, "Deep Learning Intervention for Health Care Challenges: Some Biomedical Domain Considerations," *JMIR Mhealth and Uhealth*, vol. 7, no. 8, 2019.
- [22] B. B. Benuwa, Y. Zhan, B. Ghansah, D. K. Wornyo and F. B. Kataka, "A Review of Deep Machine Learning," *International Journal of Engineering Research in Africa*, vol. 24, pp. 124-136, 2016.
- [23] S. I. Ayon, "Diabetes Prediction: A Deep Learning Approach," *I.J. Information Engineering and Electronic Business*, vol. 11, no. 2, pp. 21-27, 2019.
- [24] I. Witten, E. Frank, M. Hall and C. Pal, "Deep Learning," in *Data Mining*, Elsevier, 2017, pp. 417-466.
- [25] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11-26, 2017.
- [26] O. . I. Abiodun, A. Jantan , A. . E. Omolara, K. V. Dada, N. A. Mohamed and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, 2018.
- [27] N. Kriegeskorte and T. Golan, "Neural network models and deep learning," *Current Biology*, vol. 29, no. 7, pp. R231-R236, 2019.
- [28] P. Bizopoulos and D. Koutsouris, "Deep Learning in Cardiology," *IEEE Reviews in Biomedical Engineering*, vol. 12, pp. 168-193, 2019.

- [29] Y. Bengio and Y. LeCun, "Scaling Learning Algorithms towards AI," in *Large-Scale Kernel Machines*, MIT Press, 2007.
- [30] K. H. Miao and J. H. Miao, "Coronary Heart Disease Diagnosis using Deep Neural Networks," (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 10, 2018.
- [31] M. Ashraf, M. A. Rizvi and H. Sharma, "Improved Heart Disease Prediction Using Deep Neural Network," *Asian Journal of Computer Science and Technology*, vol. 8, no. 2, pp. 49-54, 2019.
- [32] B. H. Shekar and G. Dagnev, "Grid Search-Based Hyperparameter Tuning and Classification of Microarray Cancer Data," in *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, Gangtok, India, 2019.
- [33] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research*, vol. 13, no. 10, pp. 281-305, 2012.
- [34] R. G. Mantovani, A. L. D. Rossi, J. Vanschoren, B. Bischl and A. C. P. L. F. d. Carvalho, "Effectiveness of Random Search in SVM hyper-parameter tuning," in *IEEE Proceedings of the 2015 International Joint Conference on Neural Networks*, Killarney, Ireland, 2015.
- [35] L. C. Padierna, M. Carpio, A. Rojas, H. Puga, R. Baltazar and H. Fraire, "Hyper-Parameter Tuning for Support Vector Machines by Estimation of Distribution Algorithms," in *Nature-Inspired Design of Hybrid Intelligent Systems*, Springer, Cham, 2017, pp. 787-800.
- [36] J. Wu, X. Y. Chen, H. Zhang, L. D. Xiong, H. Lei and S. H. Deng, "Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26-40, 2019.
- [37] S. W. Hnin and C. Jeenanunta, "Bayesian optimization in a support vector regression model for short-term electricity load forecasting," *Engineering and Applied Science Research*, vol. 46, no. 3, pp. 267-275, 2019.
- [38] L. Gao and Y. Ding, "Disease prediction via Bayesian hyperparameter optimization and ensemble learning," *BMC Res Notes*, vol. 13, no. 1, p. 205, 2020.
- [39] T. A. Assegie, "An optimized K-Nearest Neighbor based breast cancer detection," *Journal of Robotics and Control (JRC)*, vol. 2, no. 3, pp. 115-118, 2020.
- [40] T. Badriyah, D. B. Santoso, I. Syarif and D. R. Syarif , "Improving stroke diagnosis accuracy using hyperparameter optimized deep learning," *International Journal of Advances in Intelligent Informatics*, vol. 5, no. 3, pp. 256-272, 2019.
- [41] National Heart, Lung, and Blood Institute, "Coronary Heart Disease," [Online]. Available: <https://www.nhlbi.nih.gov/health-topics/coronary-heart-disease>. [Accessed 29 09 2020].
- [42] National Heart, Lung, and Blood Institute, "Heart Attack," [Online]. Available: <https://www.nhlbi.nih.gov/health-topics/heart-attack>. [Accessed 29 09 2020].
- [43] R. Hajar, "Risk Factors for Coronary Artery Disease: Historical Perspectives," *Heart Views*, vol. 18, no. 3, pp. 109-114, 2017.

- [44] S. Sharif and S. E. Alway, "The diagnostic value of exercise stress testing for cardiovascular disease is more than just st segment changes: A Review," *Journal of Integrative Cardiology*, vol. 2, no. 4, pp. 341-355, 2016.
- [45] E. H. Botvinick, "Current Methods of Pharmacologic Stress Testing and the Potential Advantages of New Agents," *Journal of Nuclear Medicine Technology*, vol. 37, no. 1, pp. 14-25, 2009.
- [46] M. Kantardzic, *DATA MINING Concepts, Models, Methods, and Algorithms*, Hoboken, New Jersey: John Wiley & Sons, Inc., 2011.
- [47] S. C. Pandey, "Data Mining Techniques for Medical Data: A Review," in *International conference on Signal Processing, Communication, Power and Embedded System (SCOPE)-2016*, Paralakhemundi, India, 2016.
- [48] A. H. Seh and P. K. Chaurasia, "A Review on Heart Disease Prediction Using Machine Learning Techniques," *International Journal of Management, IT & Engineering*, vol. 9, no. 4, pp. 208-224, 2019.
- [49] F. Ertam and G. Aydin, "Data Classification with Deep Learning using Tensorflow," in *2nd International Conference on Computer Science and Engineering (UBMK)*, Antalya, 2017.
- [50] P. P. Shinde and S. Shah, "A Review of Machine Learning and Deep Learning Applications," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCCAA)*, Pune, India, 2018.
- [51] V. Kumar and M. L. Garg, "Deep Learning as a Frontier of Machine Learning: A Review," *International Journal of Computer Applications*, vol. 182, no. 1, pp. 22-30, 2018.
- [52] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, no. 1, 2015.
- [53] C. E. Nwankpa, W. Ijomah, A. Gachagan and S. Marshall, "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning," *arXiv:1811.03378*, 2018.
- [54] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations*, 2014.
- [55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929-1958, 2015.
- [56] I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT Express*, vol. 6, no. 4, pp. 312-315, 2020.
- [57] J. Wong, T. Manderson, M. Abrahamowicz, D. L. Buckeridge and R. Tamblyn, "Can Hyperparameter Tuning Improve the Performance of a Super Learner?," *Epidemiology*, vol. 30, no. 4, pp. 521-531, 2019.
- [58] A. Ambarwari, Q. J. Adrian and Y. Herdiyeni, "Analisis Pengaruh Data Scaling Terhadap Performa Algoritme Machine Learning untuk Identifikasi Tanaman," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 4, no. 1, pp. 117-122, 2020.

[59] UCI, "Heart Disease Data Set," [Online]. Available: <https://archive.ics.uci.edu/ml/datasets/Heart+Disease>. [Accessed 10 12 2019].

## LAMPIRAN

### *Dataset Cleveland*

No.	age	sex	cp	trestbps	chol	fb	restecg	thalach	exang	oldpeak	slope	ca	thal	num
1	63	1	1	145	233	1	2	150	0	2,3	3	0	6	0
2	67	1	4	160	286	0	2	108	1	1,5	2	3	3	2
3	67	1	4	120	229	0	2	129	1	2,6	2	2	7	1
4	37	1	3	130	250	0	0	187	0	3,5	3	0	3	0
5	41	0	2	130	204	0	2	172	0	1,4	1	0	3	0
6	56	1	2	120	236	0	0	178	0	0,8	1	0	3	0
7	62	0	4	140	268	0	2	160	0	3,6	3	2	3	3
8	57	0	4	120	354	0	0	163	1	0,6	1	0	3	0
9	63	1	4	130	254	0	2	147	0	1,4	2	1	7	2
10	53	1	4	140	203	1	2	155	1	3,1	3	0	7	1
11	57	1	4	140	192	0	0	148	0	0,4	2	0	6	0
12	56	0	2	140	294	0	2	153	0	1,3	2	0	3	0
13	56	1	3	130	256	1	2	142	1	0,6	2	1	6	2
14	44	1	2	120	263	0	0	173	0	0	1	0	7	0
15	52	1	3	172	199	1	0	162	0	0,5	1	0	7	0
16	57	1	3	150	168	0	0	174	0	1,6	1	0	3	0
17	48	1	2	110	229	0	0	168	0	1	3	0	7	1
18	54	1	4	140	239	0	0	160	0	1,2	1	0	3	0
19	48	0	3	130	275	0	0	139	0	0,2	1	0	3	0
20	49	1	2	130	266	0	0	171	0	0,6	1	0	3	0
21	64	1	1	110	211	0	2	144	1	1,8	2	0	3	0
22	58	0	1	150	283	1	2	162	0	1	1	0	3	0
23	58	1	2	120	284	0	2	160	0	1,8	2	0	3	1
24	58	1	3	132	224	0	2	173	0	3,2	1	2	7	3
25	60	1	4	130	206	0	2	132	1	2,4	2	2	7	4
26	50	0	3	120	219	0	0	158	0	1,6	2	0	3	0
27	58	0	3	120	340	0	0	172	0	0	1	0	3	0
28	66	0	1	150	226	0	0	114	0	2,6	3	0	3	0
29	43	1	4	150	247	0	0	171	0	1,5	1	0	3	0
30	40	1	4	110	167	0	2	114	1	2	2	0	7	3
31	69	0	1	140	239	0	0	151	0	1,8	1	2	3	0
32	60	1	4	117	230	1	0	160	1	1,4	1	2	7	2
33	64	1	3	140	335	0	0	158	0	0	1	0	3	1
34	59	1	4	135	234	0	0	161	0	0,5	2	0	7	0
35	44	1	3	130	233	0	0	179	1	0,4	1	0	3	0
36	42	1	4	140	226	0	0	178	0	0	1	0	3	0
37	43	1	4	120	177	0	2	120	1	2,5	2	0	7	3
38	57	1	4	150	276	0	2	112	1	0,6	2	1	6	1
39	55	1	4	132	353	0	0	132	1	1,2	2	1	7	3
40	61	1	3	150	243	1	0	137	1	1	2	0	3	0
41	65	0	4	150	225	0	2	114	0	1	2	3	7	4
42	40	1	1	140	199	0	0	178	1	1,4	1	0	7	0
43	71	0	2	160	302	0	0	162	0	0,4	1	2	3	0
44	59	1	3	150	212	1	0	157	0	1,6	1	0	3	0
45	61	0	4	130	330	0	2	169	0	0	1	0	3	1
46	58	1	3	112	230	0	2	165	0	2,5	2	1	7	4
47	51	1	3	110	175	0	0	123	0	0,6	1	0	3	0
48	50	1	4	150	243	0	2	128	0	2,6	2	0	7	4

49	65	0	3	140	417	1	2	157	0	0,8	1	1	3	0
50	53	1	3	130	197	1	2	152	0	1,2	3	0	3	0
51	41	0	2	105	198	0	0	168	0	0	1	1	3	0
52	65	1	4	120	177	0	0	140	0	0,4	1	0	7	0
53	44	1	4	112	290	0	2	153	0	0	1	1	3	2
54	44	1	2	130	219	0	2	188	0	0	1	0	3	0
55	60	1	4	130	253	0	0	144	1	1,4	1	1	7	1
56	54	1	4	124	266	0	2	109	1	2,2	2	1	7	1
57	50	1	3	140	233	0	0	163	0	0,6	2	1	7	1
58	41	1	4	110	172	0	2	158	0	0	1	0	7	1
59	54	1	3	125	273	0	2	152	0	0,5	3	1	3	0
60	51	1	1	125	213	0	2	125	1	1,4	1	1	3	0
61	51	0	4	130	305	0	0	142	1	1,2	2	0	7	2
62	46	0	3	142	177	0	2	160	1	1,4	3	0	3	0
63	58	1	4	128	216	0	2	131	1	2,2	2	3	7	1
64	54	0	3	135	304	1	0	170	0	0	1	0	3	0
65	54	1	4	120	188	0	0	113	0	1,4	2	1	7	2
66	60	1	4	145	282	0	2	142	1	2,8	2	2	7	2
67	60	1	3	140	185	0	2	155	0	3	2	0	3	1
68	54	1	3	150	232	0	2	165	0	1,6	1	0	7	0
69	59	1	4	170	326	0	2	140	1	3,4	3	0	7	2
70	46	1	3	150	231	0	0	147	0	3,6	2	0	3	1
71	65	0	3	155	269	0	0	148	0	0,8	1	0	3	0
72	67	1	4	125	254	1	0	163	0	0,2	2	2	7	3
73	62	1	4	120	267	0	0	99	1	1,8	2	2	7	1
74	65	1	4	110	248	0	2	158	0	0,6	1	2	6	1
75	44	1	4	110	197	0	2	177	0	0	1	1	3	1
76	65	0	3	160	360	0	2	151	0	0,8	1	0	3	0
77	60	1	4	125	258	0	2	141	1	2,8	2	1	7	1
78	51	0	3	140	308	0	2	142	0	1,5	1	1	3	0
79	48	1	2	130	245	0	2	180	0	0,2	2	0	3	0
80	58	1	4	150	270	0	2	111	1	0,8	1	0	7	3
81	45	1	4	104	208	0	2	148	1	3	2	0	3	0
82	53	0	4	130	264	0	2	143	0	0,4	2	0	3	0
83	39	1	3	140	321	0	2	182	0	0	1	0	3	0
84	68	1	3	180	274	1	2	150	1	1,6	2	0	7	3
85	52	1	2	120	325	0	0	172	0	0,2	1	0	3	0
86	44	1	3	140	235	0	2	180	0	0	1	0	3	0
87	47	1	3	138	257	0	2	156	0	0	1	0	3	0
88	53	0	3	128	216	0	2	115	0	0	1	0	?	0
89	53	0	4	138	234	0	2	160	0	0	1	0	3	0
90	51	0	3	130	256	0	2	149	0	0,5	1	0	3	0
91	66	1	4	120	302	0	2	151	0	0,4	2	0	3	0
92	62	0	4	160	164	0	2	145	0	6,2	3	3	7	3
93	62	1	3	130	231	0	0	146	0	1,8	2	3	7	0
94	44	0	3	108	141	0	0	175	0	0,6	2	0	3	0
95	63	0	3	135	252	0	2	172	0	0	1	0	3	0
96	52	1	4	128	255	0	0	161	1	0	1	1	7	1
97	59	1	4	110	239	0	2	142	1	1,2	2	1	7	2
98	60	0	4	150	258	0	2	157	0	2,6	2	2	7	3
99	52	1	2	134	201	0	0	158	0	0,8	1	1	3	0
100	48	1	4	122	222	0	2	186	0	0	1	0	3	0
101	45	1	4	115	260	0	2	185	0	0	1	0	3	0

102	34	1	1	118	182	0	2	174	0	0	1	0	3	0
103	57	0	4	128	303	0	2	159	0	0	1	1	3	0
104	71	0	3	110	265	1	2	130	0	0	1	1	3	0
105	49	1	3	120	188	0	0	139	0	2	2	3	7	3
106	54	1	2	108	309	0	0	156	0	0	1	0	7	0
107	59	1	4	140	177	0	0	162	1	0	1	1	7	2
108	57	1	3	128	229	0	2	150	0	0,4	2	1	7	1
109	61	1	4	120	260	0	0	140	1	3,6	2	1	7	2
110	39	1	4	118	219	0	0	140	0	1,2	2	0	7	3
111	61	0	4	145	307	0	2	146	1	1	2	0	7	1
112	56	1	4	125	249	1	2	144	1	1,2	2	1	3	1
113	52	1	1	118	186	0	2	190	0	0	2	0	6	0
114	43	0	4	132	341	1	2	136	1	3	2	0	7	2
115	62	0	3	130	263	0	0	97	0	1,2	2	1	7	2
116	41	1	2	135	203	0	0	132	0	0	2	0	6	0
117	58	1	3	140	211	1	2	165	0	0	1	0	3	0
118	35	0	4	138	183	0	0	182	0	1,4	1	0	3	0
119	63	1	4	130	330	1	2	132	1	1,8	1	3	7	3
120	65	1	4	135	254	0	2	127	0	2,8	2	1	7	2
121	48	1	4	130	256	1	2	150	1	0	1	2	7	3
122	63	0	4	150	407	0	2	154	0	4	2	3	7	4
123	51	1	3	100	222	0	0	143	1	1,2	2	0	3	0
124	55	1	4	140	217	0	0	111	1	5,6	3	0	7	3
125	65	1	1	138	282	1	2	174	0	1,4	2	1	3	1
126	45	0	2	130	234	0	2	175	0	0,6	2	0	3	0
127	56	0	4	200	288	1	2	133	1	4	3	2	7	3
128	54	1	4	110	239	0	0	126	1	2,8	2	1	7	3
129	44	1	2	120	220	0	0	170	0	0	1	0	3	0
130	62	0	4	124	209	0	0	163	0	0	1	0	3	0
131	54	1	3	120	258	0	2	147	0	0,4	2	0	7	0
132	51	1	3	94	227	0	0	154	1	0	1	1	7	0
133	29	1	2	130	204	0	2	202	0	0	1	0	3	0
134	51	1	4	140	261	0	2	186	1	0	1	0	3	0
135	43	0	3	122	213	0	0	165	0	0,2	2	0	3	0
136	55	0	2	135	250	0	2	161	0	1,4	2	0	3	0
137	70	1	4	145	174	0	0	125	1	2,6	3	0	7	4
138	62	1	2	120	281	0	2	103	0	1,4	2	1	7	3
139	35	1	4	120	198	0	0	130	1	1,6	2	0	7	1
140	51	1	3	125	245	1	2	166	0	2,4	2	0	3	0
141	59	1	2	140	221	0	0	164	1	0	1	0	3	0
142	59	1	1	170	288	0	2	159	0	0,2	2	0	7	1
143	52	1	2	128	205	1	0	184	0	0	1	0	3	0
144	64	1	3	125	309	0	0	131	1	1,8	2	0	7	1
145	58	1	3	105	240	0	2	154	1	0,6	2	0	7	0
146	47	1	3	108	243	0	0	152	0	0	1	0	3	1
147	57	1	4	165	289	1	2	124	0	1	2	3	7	4
148	41	1	3	112	250	0	0	179	0	0	1	0	3	0
149	45	1	2	128	308	0	2	170	0	0	1	0	3	0
150	60	0	3	102	318	0	0	160	0	0	1	1	3	0
151	52	1	1	152	298	1	0	178	0	1,2	2	0	7	0
152	42	0	4	102	265	0	2	122	0	0,6	2	0	3	0
153	67	0	3	115	564	0	2	160	0	1,6	2	0	7	0
154	55	1	4	160	289	0	2	145	1	0,8	2	1	7	4

155	64	1	4	120	246	0	2	96	1	2,2	3	1	3	3
156	70	1	4	130	322	0	2	109	0	2,4	2	3	3	1
157	51	1	4	140	299	0	0	173	1	1,6	1	0	7	1
158	58	1	4	125	300	0	2	171	0	0	1	2	7	1
159	60	1	4	140	293	0	2	170	0	1,2	2	2	7	2
160	68	1	3	118	277	0	0	151	0	1	1	1	7	0
161	46	1	2	101	197	1	0	156	0	0	1	0	7	0
162	77	1	4	125	304	0	2	162	1	0	1	3	3	4
163	54	0	3	110	214	0	0	158	0	1,6	2	0	3	0
164	58	0	4	100	248	0	2	122	0	1	2	0	3	0
165	48	1	3	124	255	1	0	175	0	0	1	2	3	0
166	57	1	4	132	207	0	0	168	1	0	1	0	7	0
167	52	1	3	138	223	0	0	169	0	0	1	?	3	0
168	54	0	2	132	288	1	2	159	1	0	1	1	3	0
169	35	1	4	126	282	0	2	156	1	0	1	0	7	1
170	45	0	2	112	160	0	0	138	0	0	2	0	3	0
171	70	1	3	160	269	0	0	112	1	2,9	2	1	7	3
172	53	1	4	142	226	0	2	111	1	0	1	0	7	0
173	59	0	4	174	249	0	0	143	1	0	2	0	3	1
174	62	0	4	140	394	0	2	157	0	1,2	2	0	3	0
175	64	1	4	145	212	0	2	132	0	2	2	2	6	4
176	57	1	4	152	274	0	0	88	1	1,2	2	1	7	1
177	52	1	4	108	233	1	0	147	0	0,1	1	3	7	0
178	56	1	4	132	184	0	2	105	1	2,1	2	1	6	1
179	43	1	3	130	315	0	0	162	0	1,9	1	1	3	0
180	53	1	3	130	246	1	2	173	0	0	1	3	3	0
181	48	1	4	124	274	0	2	166	0	0,5	2	0	7	3
182	56	0	4	134	409	0	2	150	1	1,9	2	2	7	2
183	42	1	1	148	244	0	2	178	0	0,8	1	2	3	0
184	59	1	1	178	270	0	2	145	0	4,2	3	0	7	0
185	60	0	4	158	305	0	2	161	0	0	1	0	3	1
186	63	0	2	140	195	0	0	179	0	0	1	2	3	0
187	42	1	3	120	240	1	0	194	0	0,8	3	0	7	0
188	66	1	2	160	246	0	0	120	1	0	2	3	6	2
189	54	1	2	192	283	0	2	195	0	0	1	1	7	1
190	69	1	3	140	254	0	2	146	0	2	2	3	7	2
191	50	1	3	129	196	0	0	163	0	0	1	0	3	0
192	51	1	4	140	298	0	0	122	1	4,2	2	3	7	3
193	43	1	4	132	247	1	2	143	1	0,1	2	?	7	1
194	62	0	4	138	294	1	0	106	0	1,9	2	3	3	2
195	68	0	3	120	211	0	2	115	0	1,5	2	0	3	0
196	67	1	4	100	299	0	2	125	1	0,9	2	2	3	3
197	69	1	1	160	234	1	2	131	0	0,1	2	1	3	0
198	45	0	4	138	236	0	2	152	1	0,2	2	0	3	0
199	50	0	2	120	244	0	0	162	0	1,1	1	0	3	0
200	59	1	1	160	273	0	2	125	0	0	1	0	3	1
201	50	0	4	110	254	0	2	159	0	0	1	0	3	0
202	64	0	4	180	325	0	0	154	1	0	1	0	3	0
203	57	1	3	150	126	1	0	173	0	0,2	1	1	7	0
204	64	0	3	140	313	0	0	133	0	0,2	1	0	7	0
205	43	1	4	110	211	0	0	161	0	0	1	0	7	0
206	45	1	4	142	309	0	2	147	1	0	2	3	7	3
207	58	1	4	128	259	0	2	130	1	3	2	2	7	3

208	50	1	4	144	200	0	2	126	1	0,9	2	0	7	3
209	55	1	2	130	262	0	0	155	0	0	1	0	3	0
210	62	0	4	150	244	0	0	154	1	1,4	2	0	3	1
211	37	0	3	120	215	0	0	170	0	0	1	0	3	0
212	38	1	1	120	231	0	0	182	1	3,8	2	0	7	4
213	41	1	3	130	214	0	2	168	0	2	2	0	3	0
214	66	0	4	178	228	1	0	165	1	1	2	2	7	3
215	52	1	4	112	230	0	0	160	0	0	1	1	3	1
216	56	1	1	120	193	0	2	162	0	1,9	2	0	7	0
217	46	0	2	105	204	0	0	172	0	0	1	0	3	0
218	46	0	4	138	243	0	2	152	1	0	2	0	3	0
219	64	0	4	130	303	0	0	122	0	2	2	2	3	0
220	59	1	4	138	271	0	2	182	0	0	1	0	3	0
221	41	0	3	112	268	0	2	172	1	0	1	0	3	0
222	54	0	3	108	267	0	2	167	0	0	1	0	3	0
223	39	0	3	94	199	0	0	179	0	0	1	0	3	0
224	53	1	4	123	282	0	0	95	1	2	2	2	7	3
225	63	0	4	108	269	0	0	169	1	1,8	2	2	3	1
226	34	0	2	118	210	0	0	192	0	0,7	1	0	3	0
227	47	1	4	112	204	0	0	143	0	0,1	1	0	3	0
228	67	0	3	152	277	0	0	172	0	0	1	1	3	0
229	54	1	4	110	206	0	2	108	1	0	2	1	3	3
230	66	1	4	112	212	0	2	132	1	0,1	1	1	3	2
231	52	0	3	136	196	0	2	169	0	0,1	2	0	3	0
232	55	0	4	180	327	0	1	117	1	3,4	2	0	3	2
233	49	1	3	118	149	0	2	126	0	0,8	1	3	3	1
234	74	0	2	120	269	0	2	121	1	0,2	1	1	3	0
235	54	0	3	160	201	0	0	163	0	0	1	1	3	0
236	54	1	4	122	286	0	2	116	1	3,2	2	2	3	3
237	56	1	4	130	283	1	2	103	1	1,6	3	0	7	2
238	46	1	4	120	249	0	2	144	0	0,8	1	0	7	1
239	49	0	2	134	271	0	0	162	0	0	2	0	3	0
240	42	1	2	120	295	0	0	162	0	0	1	0	3	0
241	41	1	2	110	235	0	0	153	0	0	1	0	3	0
242	41	0	2	126	306	0	0	163	0	0	1	0	3	0
243	49	0	4	130	269	0	0	163	0	0	1	0	3	0
244	61	1	1	134	234	0	0	145	0	2,6	2	2	3	2
245	60	0	3	120	178	1	0	96	0	0	1	0	3	0
246	67	1	4	120	237	0	0	71	0	1	2	0	3	2
247	58	1	4	100	234	0	0	156	0	0,1	1	1	7	2
248	47	1	4	110	275	0	2	118	1	1	2	1	3	1
249	52	1	4	125	212	0	0	168	0	1	1	2	7	3
250	62	1	2	128	208	1	2	140	0	0	1	0	3	0
251	57	1	4	110	201	0	0	126	1	1,5	2	0	6	0
252	58	1	4	146	218	0	0	105	0	2	2	1	7	1
253	64	1	4	128	263	0	0	105	1	0,2	2	1	7	0
254	51	0	3	120	295	0	2	157	0	0,6	1	0	3	0
255	43	1	4	115	303	0	0	181	0	1,2	2	0	3	0
256	42	0	3	120	209	0	0	173	0	0	2	0	3	0
257	67	0	4	106	223	0	0	142	0	0,3	1	2	3	0
258	76	0	3	140	197	0	1	116	0	1,1	2	0	3	0
259	70	1	2	156	245	0	2	143	0	0	1	0	3	0
260	57	1	2	124	261	0	0	141	0	0,3	1	0	7	1

261	44	0	3	118	242	0	0	149	0	0,3	2	1	3	0
262	58	0	2	136	319	1	2	152	0	0	1	2	3	3
263	60	0	1	150	240	0	0	171	0	0,9	1	0	3	0
264	44	1	3	120	226	0	0	169	0	0	1	0	3	0
265	61	1	4	138	166	0	2	125	1	3,6	2	1	3	4
266	42	1	4	136	315	0	0	125	1	1,8	2	0	6	2
267	52	1	4	128	204	1	0	156	1	1	2	0	?	2
268	59	1	3	126	218	1	0	134	0	2,2	2	1	6	2
269	40	1	4	152	223	0	0	181	0	0	1	0	7	1
270	42	1	3	130	180	0	0	150	0	0	1	0	3	0
271	61	1	4	140	207	0	2	138	1	1,9	1	1	7	1
272	66	1	4	160	228	0	2	138	0	2,3	1	0	6	0
273	46	1	4	140	311	0	0	120	1	1,8	2	2	7	2
274	71	0	4	112	149	0	0	125	0	1,6	2	0	3	0
275	59	1	1	134	204	0	0	162	0	0,8	1	2	3	1
276	64	1	1	170	227	0	2	155	0	0,6	2	0	7	0
277	66	0	3	146	278	0	2	152	0	0	2	1	3	0
278	39	0	3	138	220	0	0	152	0	0	2	0	3	0
279	57	1	2	154	232	0	2	164	0	0	1	1	3	1
280	58	0	4	130	197	0	0	131	0	0,6	2	0	3	0
281	57	1	4	110	335	0	0	143	1	3	2	1	7	2
282	47	1	3	130	253	0	0	179	0	0	1	0	3	0
283	55	0	4	128	205	0	1	130	1	2	2	1	7	3
284	35	1	2	122	192	0	0	174	0	0	1	0	3	0
285	61	1	4	148	203	0	0	161	0	0	1	1	7	2
286	58	1	4	114	318	0	1	140	0	4,4	3	3	6	4
287	58	0	4	170	225	1	2	146	1	2,8	2	2	6	2
288	58	1	2	125	220	0	0	144	0	0,4	2	?	7	0
289	56	1	2	130	221	0	2	163	0	0	1	0	7	0
290	56	1	2	120	240	0	0	169	0	0	3	0	3	0
291	67	1	3	152	212	0	2	150	0	0,8	2	0	7	1
292	55	0	2	132	342	0	0	166	0	1,2	1	0	3	0
293	44	1	4	120	169	0	0	144	1	2,8	3	0	6	2
294	63	1	4	140	187	0	2	144	1	4	1	2	7	2
295	63	0	4	124	197	0	0	136	1	0	2	0	3	1
296	41	1	2	120	157	0	0	182	0	0	1	0	3	0
297	59	1	4	164	176	1	2	90	0	1	2	2	6	3
298	57	0	4	140	241	0	0	123	1	0,2	2	0	7	1
299	45	1	1	110	264	0	0	132	0	1,2	2	0	7	1
300	68	1	4	144	193	1	0	141	0	3,4	2	2	7	2
301	57	1	4	130	131	0	0	115	1	1,2	2	1	7	3
302	57	0	2	130	236	0	2	174	0	0	2	1	3	1
303	38	1	3	138	175	0	0	173	0	0	1	?	3	0