

**PERBANDINGAN ALGORITMA LOGISTIC REGRESSION DAN DECISION
TREE PADA DIAGNOSIS PENYAKIT JANTUNG**

SKRIPSI

**MUHAMMAD RIFKY AMRI MUNIR
181401061**



**PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2023**

**PERBANDINGAN ALGORITMA LOGISTIC REGRESSION DAN DECISION
TREE PADA DIAGNOSIS PENYAKIT JANTUNG**

SKRIPSI

**Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah
Sarjana Ilmu Komputer**

MUHAMMAD RIFKY AMRI MUNIR

181401061



**PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2023**

PERSETUJUAN

Judul : PERBANDINGAN ALGORITMA LOGISTIC
REGRESSION DAN DECISION TREE PADA
DIAGNOSIS PENYAKIT JANTUNG

Kategori : SKRIPSI

Nama : MUHAMMAD RIFKY AMRI MUNIR

Nomor Induk Mahasiswa : 181401061

Program Studi : SARJANA (S-1) ILMU KOMPUTER

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI
INFORMASI UNIVERSITAS SUMATERA
UTARA

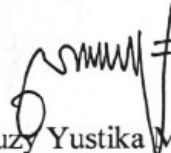
Komisi Pembimbing:

Pembimbing 2



Anandhini M Nababan, S.Kom., M.T.
NIP. 19930413 202102 2001

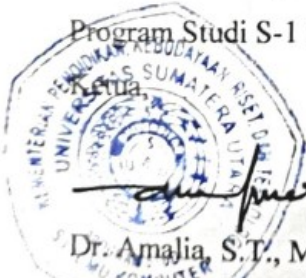
Pembimbing 1



Fuzy Yustika Manik, S.Kom., M.Kom.
NIP. 19871015 201903 2010

Diketahui/disetujui oleh

Program Studi S-1 Ilmu Komputer



Dr. Amalia, S.T., M.T.

NIP. 19781221 201404 2001

PERNYATAAN**PERBANDINGAN ALGORITMA LOGISTIC REGRESSION DAN DECISION
TREE PADA DIAGNOSIS PENYAKIT JANTUNG****SKRIPSI**

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan namanya.

Medan, Desember 2023



Muhammad Rifky Amri Munir

181401061

PENGHARGAAN

Alhamdulillahirabbil'alamin puji syukur di panjatkan kepada Allah Subhanahu Wa Ta'ala, dengan hidayah dan rahmatnya kepada penulis sehingga penulis dapat menyelesaikan penulisan tugas akhir ini sebagai syarat untuk memperoleh gelar Sarjana Komputer pada Program Studi S-1 Ilmu Komputer, Universitas Sumatera Utara.

Dalam pengerjaan tugas akhir ini penulis perlu menyampaikan terima kasih banyak kepada semua pihak yang telah memberikan dukungan dan doa. Penulis ingin mengucapkan terima kasih banyak kepada:

- 1 Kepada keluarga yang telah memberi doa dan dukungan kepada penulis selama penulis menjalankan perkuliahan sampai kepada penyusunan skripsi ini.
- 2 Bapak Dr. Muriyanto Amin, S.Sos., M.Si. selaku Rektor Universitas Sumatera Utara.
- 3 Ibu Dr. Maya Silvi Lydia, B.Sc., M.Sc. selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
- 4 Bapak Dr. Mohammad Andri Budiman, S.T., M.Comp.Sc., M.E.M., S.C.J.P. selaku Wakil Dekan 1 Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
- 5 Ibu Dr. Amalia, S.T., M.T. selaku Ketua Program Studi S-1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
- 6 Ibu Fuzy Yustika Manik, S.Kom., M. Kom. selaku Dosen Pembimbing I yang telah memberikan motivasi serta saran dan masukan yang luar kepada penulis sehingga penulis dapat menyelesaikan penulisan skripsi ini.
- 7 Ibu Anandhini Medianty Nababan S. Kom., M. T. selaku Dosen Pembimbing II yang telah memberikan semangat dan bimbingan kepada penulis, masukan dan saran dalam pengerjaan tugas akhir ini hingga selesai.

- 8 Seluruh teman-teman, abang dan kakak senior di S-1 Ilmu Komputer yang selalu memberikan motivasi kepada penulis.
- 9 Seluruh Bapak dan Ibu Dosen Program Studi S-1 Ilmu Komputer yang telah memberikan waktu dan tenaga untuk mengajar dan membimbing sehingga penulis dapat sampai kepada tahap penyusunan skripsi ini.
- 10 Seluruh pihak yang tidak bisa penulis sebutkan namanya satu per satu yang sudah memberikan dukungan kepada penulis.

Semoga Allah Subhanahu Wa Ta'ala memberikan keberkahan atas kebaikan dan bantuan yang telah diberikan.

Medan, Desember 2023

Penulis,



Muhammad Rifky Amri Munir

ABSTRAK

Penyakit jantung adalah penyakit yang menyumbang angka kematian yang tinggi. Oleh karena itu, perlu ada penilaian klinis untuk melakukan diagnosis penyakit jantung. Penilaian klinis penyakit jantung dapat dilakukan dengan menggunakan teknologi *Machine Learning*, yaitu algoritma yang melakukan pembelajaran dari suatu dataset yang kemudian setelah melakukan pembelajaran dapat melakukan prediksi. *Machine Learning* terdiri atas berbagai jenis algoritma, salah satunya adalah algoritma *Logistic Regression* dan *Decision Tree*. Algoritma *Logistic Regression* adalah algoritma yang menjelaskan hubungan variabel yang memiliki dua atau lebih kelas dengan satu atau lebih variabel bebas yang dapat dirumuskan dengan fungsi Sigmoid. Algoritma *Decision Tree* adalah suatu model algoritma yang dapat digunakan untuk membelah-belah kumpulan dataset yang banyak menjadi kumpulan yang lebih kecil berdasarkan aturan keputusan yang telah diatur. Dari kedua algoritma tersebut, akan dilakukan perbandingan untuk mencari tahu algoritma yang paling efektif. Berdasarkan hasil pengujian, didapatkan bahwa algoritma *Logistic Regression* lebih efektif daripada algoritma *Decision Tree*. Hal ini disebabkan karena nilai *accuracy* dan AUC algoritma *Logistic Regression* lebih tinggi daripada nilai *accuracy* dan AUC algoritma *Decision Tree*. Nilai *accuracy* algoritma *Logistic Regression* adalah sebesar 0.8698, sedangkan nilai *accuracy* algoritma *Decision Tree* adalah sebesar 0,8595. Nilai AUC algoritma *Logistic Regression* adalah sebesar 0.9441, sedangkan nilai AUC algoritma *Decision Tree* adalah sebesar 0,9267.

Kata kunci: Penyakit Jantung, Diagnosis, Algoritma *Logistic Regression*, Algoritma *Decision Tree*.

ABSTRACT

Heart disease is a disease that contributes to a high death rate. Therefore, there is a needs to have a clinical assessment to diagnose heart disease. Clinical assessment of heart disease can be carried out using Machine Learning technology, namely an algorithm that learns from a dataset which then after learning can make predictions. Machine Learning consists of various types of algorithms, one of which is Logistic Regression and Decision Tree algorithms. Logistic Regression algorithm is an algorithm that explains the relationship between variables that have two or more classes with one or more independent variables which can be formulated using the Sigmoid function. Decision Tree algorithm is an algorithm model that can be used to divide a large collection of datasets into smaller collections based on predetermined decision rules. Of the two algorithms, a comparison will be carried out to find out which algorithm is the most effective. Based on the test results, it was found that the Logistic Regression algorithm was more effective than the Decision Tree algorithm. This is because accuracy and AUC value of the Logistic Regression algorithm are higher than accuracy and AUC value of the Decision Tree algorithm. Accuracy value of the Logisitc Regression algorithm is 0.8698, while accuracy value of the Decision Tree algorithm is 0.8595. AUC value of the Logisitc Regression algorithm is 0.9441, while AUC value of the Decision Tree algorithm is 0.9267.

Keyword: Heart Disease, Diagnose, Logistic Regression Algorithm, Decision Tree Algorithm.

DAFTAR ISI

PERSETUJUAN	i
PERNYATAAN	ii
PENGHARGAAN.....	iii
ABSTRAK.....	v
ABSTRACT	vi
DAFTAR ISI	vii
DAFTAR GAMBAR.....	ix
DAFTAR TABEL	x
BAB 1 PENDAHULUAN.....	1
1.1. Latar Belakang	1
1.2. Rumusan Masalah	2
1.3. Batasan Masalah	2
1.4. Tujuan Penelitian	3
1.5. Manfaat Penelitian	3
1.6. Metodologi Penelitian	3
1.7. Sistematika Penulisan.....	4
BAB 2 LANDASAN TEORI.....	5
2.1. <i>Machine Learning</i>	5
2.2. Penyakit Jantung	5
2.3. <i>Logistic Regression</i>	6
2.4. <i>Decision Tree</i>	7
2.5. <i>Confusion Matrix</i>	9
2.6. <i>Accuracy</i>	10
2.7. <i>Receiver Operating Characteristics (ROC) & Area Under Curve (AUC)</i>	10
2.8. <i>G-Means</i>	12
2.9. <i>K-Fold Cross Validation</i>	12
2.10. Standarisasi Data.....	13
2.11. <i>Python</i>	13
2.12. <i>Scikit-Learn</i>	14
2.13. Penelitian yang Relevan	14

BAB 3 ANALISIS DAN PERANCANGAN	16
3.1. Analisis Sistem	16
3.1.1. Analisis Masalah	16
3.1.2. Analisis Kebutuhan.....	17
3.1.2.1. Kebutuhan Fungsional.....	17
3.1.2.2. Kebutuhan Non Fungsional	18
3.2. Analisis Data.....	18
3.3. <i>Flowchart</i>	20
3.3.1. <i>Flowchart</i> Sistem.....	20
3.3.2. <i>Flowchart</i> Algoritma <i>Logistic Regression</i>	22
3.3.3. <i>Flowchart</i> Algoritma <i>Decision Tree</i>	23
3.4. Perancangan Sistem	24
3.4.1. Arsitektur Umum Sistem	24
3.4.2. <i>Use Case Diagram</i>	25
3.4.3. <i>Activity Diagram</i>	25
3.4.4. <i>Sequence Diagram</i>	26
3.5. Perancangan <i>Interface</i>	27
BAB 4 IMPLEMENTASI DAN PENGUJIAN	29
4.1. Implementasi Algoritma.....	29
4.1.1. <i>Data Scaling</i>	29
4.1.2. Implementasi Algoritma <i>Logistic Regression</i>	34
4.1.3. Implementasi Algoritma <i>Decision Tree</i>	36
4.2. Implementasi Sistem	43
4.2.1. Halaman Utama Sistem	43
4.2.2. Halaman Input Dataset.....	45
4.3. Pengujian Sistem.....	46
4.3.1. Hasil Pengujian.....	46
4.3.2. Perbandingan Hasil Pengujian	49
BAB 5 KESIMPULAN DAN SARAN.....	52
5.1 Kesimpulan.....	52
5.2 Saran.....	53
DAFTAR PUSTAKA.....	54

DAFTAR GAMBAR

Gambar 2.1 Contoh Bentuk <i>Decision Tree</i>	8
Gambar 2.2 Contoh Bentuk <i>ROC Curve</i>	11
Gambar 2.3 Contoh Proses <i>k-Fold Cross Validation</i>	12
Gambar 3.1 Diagram Ishikawa.....	17
Gambar 3.2 <i>Flowchart</i> Sistem	21
Gambar 3.3 <i>Flowchart</i> Algoritma <i>Logistic Regression</i>	22
Gambar 3.4 <i>Flowchart</i> Algoritma <i>Decision Tree</i>	23
Gambar 3.5 Arsitektur Umum Sistem	24
Gambar 3.6 <i>Use Case Diagram</i>	25
Gambar 3.7 <i>Activity Diagram</i>	26
Gambar 3.8 <i>Sequence Diagram</i>	27
Gambar 3.9 Rancangan <i>Interface</i>	28
Gambar 4.1 Bentuk Awal <i>Decision Tree</i>	42
Gambar 4.2 <i>Decision Tree</i> dengan Depth Maksimal Sebesar 3.....	42
Gambar 4.3 Tampilan Halaman Sistem Sebelum Pemrosesan	44
Gambar 4.4 Tampilan Halaman Sistem Sesudah Pemrosesan.....	45
Gambar 4.5 Tampilan Halaman Input	45
Gambar 4.6 Grafik ROC algoritma <i>Logistic Regression</i>	46
Gambar 4.7 Grafik ROC algoritma <i>Decision Tree</i>	49

DAFTAR TABEL

Tabel 2.1 <i>Confusion Matrix</i>	9
Tabel 2.2 Daftar Ukuran	10
Tabel 3.1 Daftar Variabel	19
Tabel 3.2 Contoh Salah Satu Data Pasien	19
Tabel 3.3 Penjelasan Rancangan <i>Interface</i>	28
Tabel 4.1 Nilai Data <i>Training</i> Sebelum Standarisasi	29
Tabel 4.2 Nilai Rata-Rata dan Standar Deviasi Setiap Variabel.....	31
Tabel 4.3 Nilai Data <i>Training</i> Setelah Standarisasi	32
Tabel 4.4 Nilai Data <i>Test</i> Sebelum Standarisasi	33
Tabel 4.5 Nilai Data <i>Test</i> Setelah Standarisasi	34
Tabel 4.6. Nilai <i>Intercept</i> dan Koefisien	35
Tabel 4.7 Sampel Salah Satu Data Test.....	35
Tabel 4.8 Distribusi Variabel Independen	37
Tabel 4.9 Nilai <i>Gini Impurity</i> Setiap Variabel.....	41
Tabel 4.10 Sampel Salah Satu Data Test	43
Tabel 4.11 Nilai Pengujian Algoritma <i>Logistic Regression</i>	46
Tabel 4.12 Nilai Rata-Rata <i>Accuracy</i> dan AUC Setiap <i>Depth</i>	47
Tabel 4.13 Nilai Rata-Rata <i>G-Means</i> dan Rata-Rata k-Fold Setiap <i>Depth</i>	47
Tabel 4.14 Nilai Pengujian Algoritma <i>Decision Tree</i>	48
Tabel 4.15 Nilai Pengujian Kedua Algoritma.....	50

BAB I

PENDAHULUAN

1.1 Latar Belakang

Penyakit jantung adalah penyakit yang menyumbang angka kematian yang tinggi. Penyakit jantung di Indonesia mencapai angka 1,5% pada penduduk semua umur, yang berarti setiap 100 ribu orang penduduk semua umur 1,5% memiliki penyakit jantung (Tampubolon *et al.*, 2023). Angka kematian karena penyakit jantung tidak hanya tinggi di Indonesia, tetapi juga pada negara besar yang lain. Data dari World Health Organisation (WHO) menunjukkan bahwa angka penyakit jantung pada tahun 2013 di Amerika Serikat kurang lebih sekitar 550.000 kasus per tahun (Nursita & Pratiwi, 2020).

Sebagai respons dari besarnya angka kematian penyakit jantung, maka perlu ada penilaian klinis untuk mendiagnosis penyakit jantung. Caranya yaitu menggunakan imaging jantung yang non invasif, seperti menggunakan X-ray atau 2D-echo. Bisa juga menggunakan tes diagnostik, dimana akan dilakukan pemeriksaan laboratorium yaitu uji pemeriksaan darah lengkap, elektrolit, nitrogen urea darah, dan sebagainya (Gedela *et al.*, 2015). Penilaian klinis juga dapat dilakukan dengan teknologi lain, misalnya menggunakan dengan *Machine Learning*.

Machine Learning merupakan algoritma yang melakukan pembelajaran dari suatu dataset yang kemudian setelah melakukan pembelajaran dapat melakukan prediksi. *Machine Learning* memiliki berbagai macam variasi-variasi algoritma. Beberapa contohnya adalah *Logistic Regression*, *Decision Tree*, dan sebagainya (Alaoui *et al.*, 2018).

Algoritma *Logistic Regression* dan *Decision Tree* dapat digunakan untuk mendiagnosis penyakit jantung. Alasannya adalah karena kedua algoritma ini dapat belajar dan testing dengan data-data diagnosis

sebelumnya sehingga dapat menjadi pendiagnosis penyakit jantung. Kedua algoritma juga termasuk algoritma klasifikasi sehingga tepat digunakan untuk melakukan diagnosis. (Alaoui *et al.*, 2018).

Dalam penulisan ini, penulis akan membandingkan algoritma *Decision Tree* dan *Logistic Regression* untuk dicari tahu yang mana yang paling efektif dalam mendiagnosis penyakit jantung. Meskipun kedua algoritma bisa diimplementasikan dalam mendiagnosis penyakit jantung, kita tentunya perlu mengetahui mana yang paling efektif dalam mendiagnosis penyakit jantung.

1.2 Rumusan Masalah

Penyakit jantung adalah penyakit yang menyumbang angka kematian yang tinggi. Oleh karena itu, diperlukan metode untuk mendiagnosis penyakit jantung. Penelitian ini menggunakan *Machine Learning*. Dari algoritma-algoritma *Machine Learning* yang ada, peneliti menyoroti algoritma *Logistic Regression* dan algoritma *Decision Tree* untuk membantu mendiagnosis penyakit jantung. Kinerja kedua algoritma tersebut akan dibandingkan.

1.3 Batasan Masalah

Agar penulisan tugas akhir ini memiliki arah, diperlukan batasan masalah pada penelitian ini. Batasan masalah pada penelitian ini adalah sebagai berikut:

- 1 Menggunakan algoritma *Logistic Regression* dan *Decision Tree*.
- 2 Dataset yang digunakan diambil dari *Open Knowledge Foundation*.
- 3 Diagnosis hanya mendeteksi adanya keberadaan penyakit jantung tanpa mendiagnosis jenis penyakit jantung yang dialami pasien.
- 4 Program dirancang dengan bahasa *Python*.
- 5 Program yang dirancang berbasis *desktop*.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah membandingkan antara algoritma *Logistic Regression* dan *Decision Tree* untuk mengetahui algoritma mana yang paling efektif dalam mendiagnosis penyakit jantung.

1.5 Manfaat Penelitian

Manfaat yang diperoleh dari penelitian ini adalah:

- 1 Mengetahui perbandingan antara algoritma *Decision Tree* dan *Logistic Regression* dalam mendiagnosis penyakit jantung.
- 2 Memberikan pengetahuan tentang penyakit jantung.
- 3 Menjadi referensi untuk penelitian kedepan dalam hal perbandingan antara algoritma *Decision Tree* dan *Logistic Regression*.

1.6 Metodologi Penelitian

Metodologi penelitian yang dilakukan pada penelitian ini adalah:

1. Studi Pustaka

Pada penelitian ini, sebagai tahap awal penulis menggunakan metode studi pustaka untuk mencari referensi dari sumber-sumber yang terpercaya tentang hal-hal yang berkaitan dengan penelitian.

2. Analisis dan Perancangan

Pada tahap ini penulis akan menganalisis hal-hal yang bersangkutan dalam penelitian serta membuat rancangan yang divisualisasikannya dengan algoritma, diagram *Ishikawa*, *use case diagram*, *activity diagram*, *sequence diagram*, dan desain *interface*.

3. Implementasi

Pada tahap ini penulis akan membangun program berdasarkan rancangan yang sudah dirancang. Pembangunan aplikasi akan menggunakan bahasa pemrograman *Python*.

4. Pengujian

Pengujian merupakan proses menguji program yang telah dibangun terhadap hal-hal yang telah ditentukan sebelumnya dan memastikan

hasil yang diperoleh dari sistem sesuai dengan yang diharapkan.

5. Perbandingan Hasil

Hasil yang telah diperoleh akan dibandingkan sesuai dengan pedoman yang telah dipelajari melalui studi pustaka.

1.7 Sistematika Penulisan

Berikut merupakan sistematika yang digunakan pada penulisan ini:

BAB I PENDAHULUAN

Pendahuluan disusun atas latar belakang penelitian ini dilakukan, rumusan masalah penelitian, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

BAB II LANDASAN TEORI

Landasan teori berisi atas teori-teori dari berbagai sumber yang berfungsi sebagai dasar penelitian ini dapat dilakukan.

BAB III ANALISIS DAN PERANCANGAN

Analisis dan perancangan berisi analisis penulis tentang hal yang bersangkutan dengan penelitian serta rancangan-rancangan penulis dalam penelitian ini sebelum melakukan implementasi.

BAB IV IMPLEMENTASI DAN PENGUJIAN

Implementasi dan pengujian tersusun atas implementasi program yang terdiri atas implementasi algoritma, spesifikasi, dan *interface*, serta bagaimana pengujiannya.

BAB V KESIMPULAN DAN SARAN

Kesimpulan dan saran berisi kesimpulan yang diperoleh dari penelitian ini serta saran yang diberikan untuk sebagai referensi yang dapat digunakan untuk penelitian yang akan mendatang ke depan.

BAB II

LANDASAN TEORI

2.1 *Machine Learning*

Machine Learning merupakan algoritma yang melakukan pembelajaran dari suatu dataset yang kemudian setelah melakukan pembelajaran dapat melakukan prediksi. *Machine Learning* dalam memperoleh suatu kecerdasan akan melalui dua fase pembelajaran sebelum dapat melakukan fungsinya, yang antara lain adalah latihan (*training*) dan percobaan (*testing*). (Roihan *et al.*, 2020)

Machine learning terdiri dari beberapa jenis berdasarkan fungsinya, yaitu *Supervised Learning* dan *Unsupervised Learning*. Metode yang diterapkan oleh *Supervised Learning* adalah metode klasifikasi di mana kumpulan data akan diberikan label untuk melakukan proses klasifikasi kelas-kelas. *Unsupervised Learning* adalah metode yang dapat disebut dengan metode *clustering* disebabkan tidak perlu adanya kebutuhan pemberian label dalam kumpulan data, tidak seperti *Supervised Learning* (Roihan *et al.*, 2020).

Beberapa tahun belakangan ini, *Machine Learning* telah diaplikasikan dalam menyelesaikan banyak sekali masalah-masalah dunia nyata seperti pengecekan plagiarisme, rekognisi bahasa lisan, sistem rekomendasi, dan lain-lain. Salah satu pekerjaan yang dapat dilakukan oleh *Machine Learning* pula adalah melakukan diagnosis terhadap pasien yang terkena penyakit.

2.2 Penyakit Jantung

Penyakit jantung adalah penyakit yang menyebabkan jantung tidak mampu lagi dalam melakukan pekerjaannya seperti biasanya. Ini mengakibatkan akan adanya gangguan pada jantung mengingat pekerjaannya sebagai pemompa darah ke seluruh bagian tubuh. Penyebab dari penyakit jantung yang dikenai pengidapnya dikarenakan otot jantung yang sudah tidak sekuat seperti biasanya manusia normal atau jika terdapat celah di jantung, tepatnya di serambi kiri dan serambi kanan seseorang sehingga menyebabkan peercampuran antara darah bersih dan darah kotor di dalam tubuh. (Narvadi, 2019). Penyakit jantung juga disebabkan oleh kerusakan sel otot-otot jantung

dalam memompa aliran darah keseluruh tubuh yang diakibatkan oleh kurangnya oksigen yang dibawa darah ke pembuluh darah.

Penyakit jantung menyumbang angka kematian yang cukup tinggi. Penyakit jantung di Indonesia mencapai angka 1,5% pada penduduk semua umur, yang berarti setiap 100 ribu orang penduduk semua umur 1,5% memiliki penyakit jantung (Tampubolon *et al.*, 2023). Angka kematian karena penyakit jantung tidak hanya tinggi di Indonesia, tetapi juga pada negara besar yang lain. Data dari World Health Organisation (WHO) menunjukkan bahwa angka penyakit jantung pada tahun 2013 di Amerika Serikat kurang lebih sekitar 550.000 kasus per tahun (Nursita *et al.*, 2020).

Komplikasi penyakit jantung terdiri atas bermacam-macam bentuk. Salah satunya yaitu bisa berupa kematian sebagai hal yang paling membahayakan. Komplikasi lainnya dapat berupa harusnya pengidap penyakit jantung tinggal di rumah sakit, cacat fisik, dan peningkatan biaya perawatan yang diyakini akan terus meningkat hingga dekade mendatang. (Edgardo *et al.*, 2019)

Konsultasi diperlukan dalam menghadapi penyakit jantung, yang dimana dapat melibatkan dokter, perawat, ahli diet, ahli jantung, ahli saraf, dan spesialis lainnya. Selain itu, pasien juga perlu melakukan beberapa pencegahan dengan menjalani gaya hidup sehat, dan menerapkan pola makan yang tepat sedini mungkin. (Edgardo *et al.*, 2019).

2.3 *Logistic Regression*

Logistic Regression adalah algoritma yang menjelaskan hubungan variabel yang memiliki dua atau lebih kelas dengan satu atau lebih variabel bebas (Hendayana, 2012). *Logistic Regression* dapat dirumuskan dengan fungsi sigmoid dimana hasil prediksi atau variabel dependen dari fungsi adalah nilai antara 0 dan 1.

$$f(x) = \frac{1}{1+e^{-x}}$$

Dimana:

x = variabel independen

$f(x)$ = variabel dependen

Salah satu model *Logistic Regression* yang banyak digunakan adalah model *Logistic Regression* biner. Model *Logistic Regression* biner adalah model yang digunakan untuk mencari hubungan antara satu variabel dependen dan beberapa variabel independen yang berfungsi sebagai prediksi untuk variabel dependen. Variabel dependen memiliki respon berupa nilai 1 untuk menyatakan nilai *true* dan bernilai 0 untuk menyatakan nilai *false* (Tampil *et al.*, 2017).

Distribusi yang digunakan model *Logistic Regression* biner adalah distribusi *Bernouli* (Afifah, 2020). Model regresi logistik memperkirakan bahwasanya variabel biner harus saling bebas, sehingga variabel biner memiliki sebaran binom. Model regresi logistik dengan sebaran binom adalah sebagai berikut:

$$\pi(x) = \frac{e^{(\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n)}}{1 + e^{(\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n)}}$$

atau

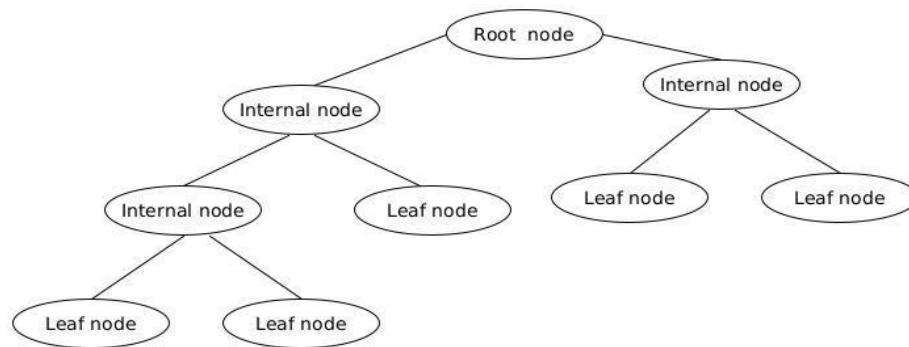
$$\pi(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n)}}$$

Model regresi biner membulatkan hasil dari fungsi sigmoid ke nilai-nilai terdekat. Umumnya, jawaban di bawah 0,5 dibulatkan menjadi 0, dan jawaban di atas 0,5 dibulatkan menjadi 1, sehingga fungsi logistik mengembalikan hasil biner. Oleh karena itu, model ini cocok digunakan untuk melakukan diagnosis penyakit jantung.

2.4 *Decision Tree*

Decision Tree adalah suatu model yang dapat digunakan untuk membelah-belah kumpulan dataset yang banyak menjadi kumpulan yang lebih kecil berdasarkan aturan keputusan yang telah diatur. *Decision Tree* mengubah dataset menjadi sebuah pohon keputusan yang merepresentasikan aturan-aturan. *Decision Tree* memiliki kegunaan untuk menemukan hubungan implisit beberapa variabel input atau independen dengan variabel target atau dependen. Oleh karena itu, *Decision Tree* dapat digunakan sebagai algoritma *Machine Learning*.

Decision Tree dapat direpresentasikan sebagai sebuah diagram yang berbentuk seperti pohon yang tersusun atas beberapa *node* (simpul). *Node* yang paling tinggi disebut sebagai *root node*. *Root node* tidak memiliki *edge* masuk dikarenakan posisinya, tetapi masih memiliki beberapa *edge* (rusuk) keluar. *Internal node* memiliki satu *edge* masuk dan beberapa *edge* keluar, sedangkan *leaf node* hanya memiliki satu *edge* masuk tanpa memiliki *edge* keluar. (Kasih, 2019).



Gambar 2.1 Contoh Bentuk Decision Tree

Dalam implementasinya, *Decision Tree* menggunakan suatu konsep bernama *Node Impurity* untuk menentukan variabel terbaik yang diletakkan di setiap *node* yang berada di *Decision Tree*. *Impurity* dapat dihitung dengan beberapa cara, contohnya menggunakan konsep seperti *Information Gain*, *Gain Ratio*, dan *Gini Index* (Muchai et al., 2014).

Dalam penelitian ini, akan digunakan *Gini Impurity (index)* sebagai penghitung *impurity* yang dimiliki variabel. *Gini Impurity* didapatkan dengan rumus sebagai berikut:

$$Gini\ Impurity = 1 - \sum_{i=1}^j p(i)^2$$

Dimana :

j = jumlah kelas variabel target

i = urutan kelas

p (i)= rasio kelas ke-i

Untuk menghitung nilai total *Gini Impurity* jika variabel dibagi dua kelas, maka dapat dirumuskan:

$$Gini\ total = \frac{a}{d} \times Gini(a) + \frac{b}{d} \times Gini(b)$$

Dimana:

a = jumlah yang berada di kelas a

b = jumlah yang berada di kelas b

d = total semua jumlah ($a+b$)

$Gini(a) = Gini$ kelas a

$Gini(b) = Gini$ kelas b

2.5 Confusion Matrix

Confusion Matrix adalah matriks yang mendefinisikan hasil perbandingan antara nilai prediksi dengan nilai kenyataan. *Confusion Matrix* akan merepresentasikan nilai kebenaran dari sebuah prediksi yang dilakukan. Oleh karena itu, *Confusion Matrix* dapat digunakan untuk mengetahui performa dari model algoritma. (Hutomo, 2020)

Karena diagnosis penyakit jantung hanya memiliki dua kelas, yaitu *true* dan *false*, maka *Confusion Matrix* berukuran 2×2 . Berikut tabel 2.1 adalah bentuk tabel *Confusion Matrix*-nya

Tabel 2.1 *Confusion Matrix*

		Hasil Prediksi	
		-	+
Kenyataan (Actual)	-	<i>True Negative (TN)</i>	<i>False Positive (FP)</i>
	+	<i>False Negative (FN)</i>	<i>True Positive (TP)</i>

1. *True Negative (TN)* : berapa kali benar memprediksi variabel bernilai *false*.
2. *False Positive (FP)* : berapa kali salah memprediksi variabel bernilai *false*.
3. *False Negative (FN)* : berapa kali salah memprediksi variabel bernilai *true*.
4. *True Positive (TN)* : berapa kali benar memprediksi variabel bernilai *true*.

Sejumlah ukuran dapat ditentukan berdasarkan rumusan-rumusan nilai yang tertera pada tabel 2.1 mengenai *Confusion Matrix*. Ukuran ini dapat digunakan sebagai dasar *output* hasil pengujian. Berikut pada tabel 2.2 mengenai rumus dan fungsi dari masing-masing ukuran tersebut.

Tabel 2.2 Daftar Ukuran

Ukuran	Rumus
<i>Sensitivity/Recall/Tprate</i>	$\frac{TP}{TP + FN}$
<i>Specificity/Tnrate</i>	$\frac{TN}{TN + FP}$
<i>Fprate</i>	$\frac{FP}{FP + TN}$

2.6 Accuracy

Accuracy adalah nilai yang banyak digunakan dalam menghitung performa suatu sistem belajar atau sistem *Machine Learning*. *Accuracy* dari suatu sistem yang dimana *Confusion Matrix*-nya berukuran 2x2 dapat dihitung dengan rumus sebagai berikut (Garcia *et al.*, 2009) :

$$Accuracy = \frac{TP+TN}{TP+FN+TN+FP}$$

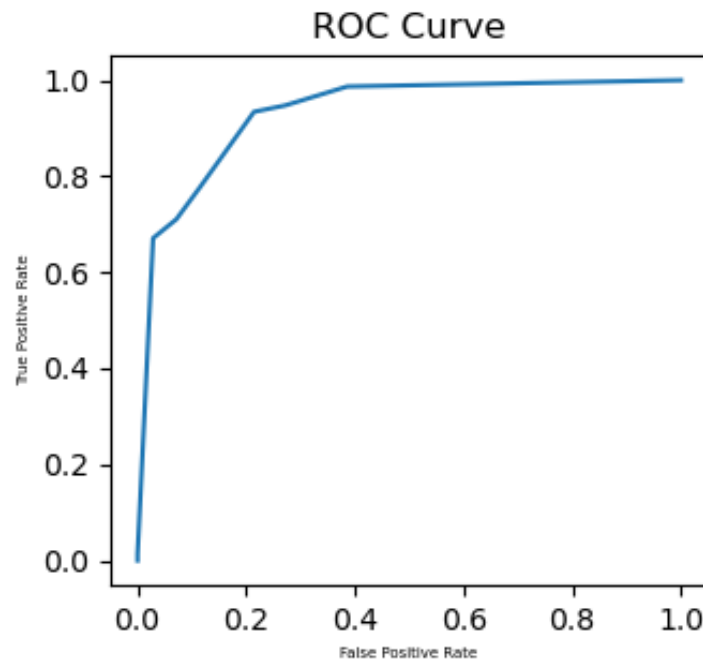
Accuracy memiliki definisi yang mudah dipahami untuk orang-orang. Tetapi, *accuracy* memiliki kelemahan jika suatu dataset memiliki data yang tidak *balance*. Misalkan, suatu dataset memiliki 95% data bernilai *true*, dan 5% bernilai *false*. Kemudian, dataset di-*split* dengan 10% dari dataset menjadi data *test* dengan persentase nilai *true* dan *false* mirip seperti sebelum di-*split*. Nilai *accuracy* bisa tetap tinggi jika sistem benar dalam memprediksi hampir semua nilai *true* meskipun sistem salah dalam memprediksi hampir semua nilai *false*. Perlu ada cara lain untuk menutupi kelemahan tersebut.

2.7 Receiver Operating Characteristics (ROC) Curve & Area Under Curve (AUC)

ROC curve merupakan grafik yang digunakan untuk menggambarkan performa algoritma *Machine Learning*. *ROC* ditampilkan dengan menggunakan grafik dua dimensi untuk mengukur perbedaan performa algoritma yang digunakan dalam proses *Machine Learning*. (Abdussomad, 2017).

Grafik *ROC* akan menampilkan hubungan antara *FPRate* (1 - *Specificity*) dan *TPRate* (*Sensitivity*) suatu algoritma *Machine Learning*. Grafik akan menampilkan sebuah kurva yang dimana menggambarkan hubungan antara *FPRate* dan *TPRate* model suatu algoritma.

Grafik ROC adalah teknik yang sangat populer digunakan dalam mengevaluasi suatu kinerja *Machine Learning* di sebuah domain yang tidak *balance* (Garcia *et al.*, 2009). Misalnya, jika 95% data bernilai *false*, dan sisanya yaitu 5% bernilai *true*. Oleh karena itu, grafik ROC sangat berguna untuk menutup kelemahan yang ada pada nilai akurasi.



Gambar 2.2 Contoh Bentuk ROC Curve

Melalui ROC curve kinerja *Machine Learning* dapat dihitung melalui nilai *Area Under Curve* (AUC). AUC dapat digunakan untuk mengevaluasi kemampuan diagnosis suatu algoritma dalam menentukan status penyakit jantung sebenarnya dari pasien.

Nilai AUC dapat dibagi menjadi beberapa kategori (Hutomo, 2020):

1. 0.90 – 1.00 = *Excellent classification*
2. 0.80 – 0.90 = *Good classification*
3. 0.70 – 0.80 = *Fair classification*
4. 0.60 – 0.70 = *Poor classification*
5. 0.50 – 0.60 = *Failure*

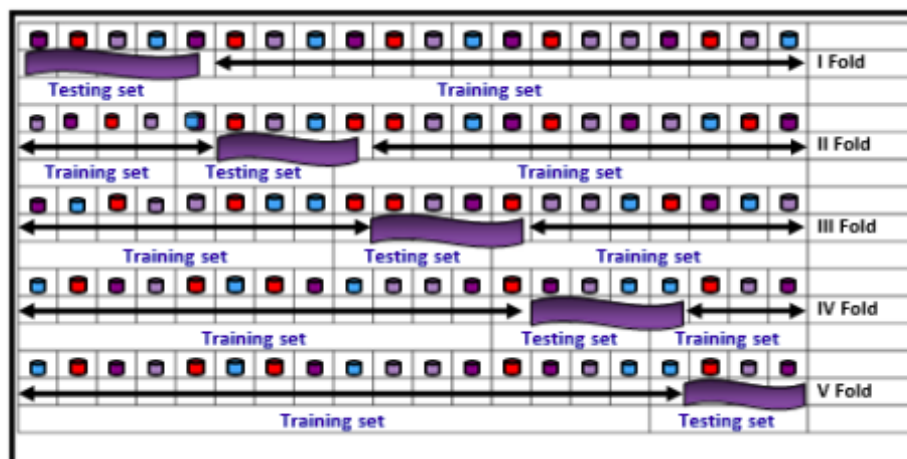
2.8 G-Means

Geometric Means (G-Means) adalah nilai yang dapat digunakan untuk merepresentasikan kemampuan suatu algoritma dalam melaksanakan tugasnya di kelas yang tidak seimbang (*imbalance*) pada dataset. Nilai *G-Means* didapatkan dengan akar kuadrat dari total nilai perkalian antara nilai *sensitivity (TPRate)* dan *specificity (TNRate)* (Hutomo, 2020).

$$G\text{-Means} = \sqrt{\text{Sensitivity} \times \text{Specificity}}$$

2.9 K-Fold Cross Validation

K-Fold Cross Validation adalah salah satu metode *Cross Validation*, suatu metode yang dipakai untuk mengevaluasi hasil suatu model algoritma, yang digunakan untuk mengecek apakah adanya kelemahan pada model algoritma saat melakukan prediksi dengan melakukan beberapa bentuk pengujian pada dataset. Oleh karena itu, permasalahan seperti *overfitting* dan *underfitting* bisa terselesaikan dengan menggunakan *k-Fold Cross Validation*. Proses akan dimulai dengan membagi dataset sebanyak *k* jumlah *fold*, yang dimana nilai *k* dapat ditentukan oleh pembuat sistem, yang kemudian dijadikan sebagai data *test* dan data *training* sebagai model evaluasi dari setiap *fold* yang dihasilkan oleh metode *k-Fold Cross Validation* (Firmansyach *et al.*, 2023). Berikut gambar 2.3 menunjukkan prosesnya.



Gambar 2.3 Contoh Proses *k-Fold Cross Validation*

2.10 Standarisasi Data

Standarisasi data adalah salah satu teknik data *scaling*, yaitu teknik yang mengubah nilai angka dalam dataset yang akan menjadi bahan penelitian menjadi nilai tertentu. Tentu saja rentang nilai dalam dataset tidak akan dikaburkan. Dalam melakukan proses data *scaling*, standarisasi data akan melibatkan perhitungan nilai rata-rata dan standar deviasi, yaitu dengan mengubah nilai rata-rata (*mean*) data suatu atribut adalah 0 dan standar deviasinya menjadi 1 (Ambarwari *et al.*, 2020).

Data yang distandarisasi dapat dicari dengan persamaan:

$$x' = \frac{x - \bar{x}}{\sigma}$$

Dimana :

x' = nilai data setelah distandarisasi

x = nilai data awal

\bar{x} = nilai rata-rata

σ = standar deviasi

2.11 Python

Python adalah bahasa pemrograman level tinggi yang banyak digunakan dekat-dekat ini. Diciptakan oleh Guido van Rossum dan dirilis pertama kali pada 1991, *Python* menekankan penggunaan kode yang mudah dibaca. Bahasanya distruktur dan dibangun dengan bertujuan membantu *Software Engineer* dalam melakukan koding dalam lingkup yang kecil dan besar.

Python memiliki beragam fitur yang memiliki kegunaan yang bermacam-macam, sehingga *Python* termasuk bahasa yang serbaguna. Beberapa contoh kegunaan *Python* adalah bahasa ini diterapkan di *Web Development*, *Game Development*, *Machine Learning*, sains, bisnis, sistem tanam, pengolahan citra, dan sebagainya (Jaiswal *et al.*, 2021).

Python dikenal sebagai bahasa yang dipilih dalam melakukan riset di bidang *Machine Learning*. Membangun dan melatih model sudah menjadi lebih mudah berkat metode dan alat-alat yang disediakan oleh *library* seperti *Scikit-learn*, *TensorFlow*, and *PyTorch* (Ranjan *et al.*, 2023).

2.12 Scikit-Learn

Scikit-learn adalah modul *Python* yang menggabungkan variasi-variasi algoritma komputer ultra modern dalam melakukan *learning*. Saat *Python* mulai banyak digunakan *Scikit-learn* di ranah industri, *Scikit-learn* memanfaatkan kesempatan tersebut untuk menyediakan implementasi-implementasi algoritma komputer, sekaligus juga tetap mempertahankan mudahnya penggunaan *interface* yang terintegrasi dengan bahasa *Python* (Chary & Singh, 2020).

Scikit-learn adalah *library machine learning Python* yang populer digunakan banyak orang. Berbagai macam jenis kegunaan dapat diperoleh dari *Scikit-learn*, yaitu pengguna dapat melakukan pekerjaan seperti klasifikasi, regresi, klustering, pengurangan dimensi, dan alat dan teknik pemilihan model. *Scikit-learn* juga memuat beberapa fungsi lain seperti *data preprocessing*, *model evaluation*, dan utilitas untuk *cross validation* (Ranjan *et al.*, 2023).

2.13 Penelitian yang Relevan

Pada penelitian ini terdapat beberapa penelitian yang terkait dengan metode yang dilakukan oleh penulis antara lain:

- 1 Berdasarkan hasil penelitian yang telah dilakukan oleh K. Polaraju dan D. Durga Prasad dengan judul “*Prediction of Heart Disease using Multiple Linear Regression Model*” menghasilkan kesimpulan bahwa model *Multiple Linear Regression* adalah model yang tepat untuk memprediksi kemungkinan penyakit jantung.
- 2 Berdasarkan hasil penelitian yang telah dilakukan oleh Madhumita Pal dan Smita Parija dengan judul “*Prediction of Heart Diseases using Random Forest*” menghasilkan kesimpulan bahwa diperoleh *Sensitivity Value* sebesar 90.6%, *Specificity Value* sebesar 82.7 dan *Accuracy Value* sebesar 86.9 untuk prediksi menggunakan *Random Forest*.
- 3 Berdasarkan hasil penelitian yang telah dilakukan oleh Nuraida Latif dan Ashara dengan judul “*Penerapan Model Decision Tree Algoritma Untuk Mengidentifikasi Penyakit Pencernaan Dengan Pengobatan Herbal*” menghasilkan kesimpulan bahwa pendiagnosaan penyakit pencernaan

dengan pengobatan cara herbal dengan memberikan hasil pendagnosaan dan hasilnya cukup baik berdasarkan gejala – gejala yang diberikan dengan cepat dan akurat.

- 4 Berdasarkan hasil penelitian yang telah dilakukan oleh Karnika Dwivedi, Dr.Hari Om Sharan dan Vinod Vishwakarma dengan judul “*Analysis Of Decision Tree For Diabetes Prediction*” menghasilkan kesimpulan bahwa model decision tree tersebut menunjukkan struktur pohon yang memungkinkan pengguna mengambil keputusan yang akurat berdasarkan input.
- 5 Berdasarkan hasil penelitian yang telah dilakukan oleh Amrin dan Omar Pahlevi dengan judul “*Implementasi Algoritma Klasifikasi Logistic Regression dan Naïve Bayes untuk Diagnosa Penyakit Hepatitis*” menghasilkan kesimpulan bahwa performa metode *Logistic Regression* untuk prediksi penyakit hepatitis memberikan tingkat akurasi kebenaran sebesar 84,62% dengan nilai *area under the curve* (AUC) sebesar 0,841. Sedangkan performa metode *Naïve Bayes* memberikan tingkat akurasi kebenaran sebesar 83,71% dengan nilai *area under the curve* (AUC) sebesar 0,816
- 6 Berdasarkan hasil penelitian yang telah dilakukan oleh Anisha P. R., Kishor Kumar Reddy C., K. Apoorva, dan Meghana Mangipudi C. dengan judul “*Early Diagnosis of Breast Cancer Prediction using Random Forest Classifier*” menghasilkan kesimpulan bahwa setelah melakukan pengetestan model, nilai AUC didapatkan sebesar 0.98 yang mengindikasikan akurasi sebesar 98%.
- 7 Berdasarkan penelitian yang dilakukan Nasser H. Sweilam, A.A. Tharwat, dan N.K. Abdel Moniem dengan judul “*Support vector machine for diagnosis cancer disease: A comparative study*” menghasilkan kesimpulan bahwa teknik PSO dan QPSO sedikit lebih tinggi nilai akurasinya dibandingkan teknik lainnya.

BAB III

ANALISIS DAN PERANCANGAN

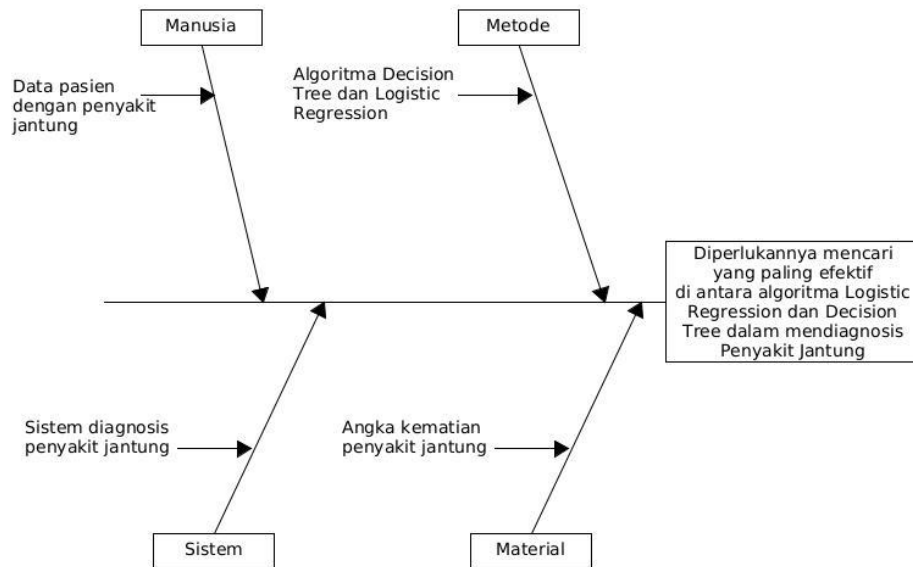
3.1 Analisis Sistem

Analisis sistem diperlukan untuk mengetahui perihai-perihai yang terkait dalam membuat sistem yang dilakukan di penelitian ini. Berikut analisis sistem yang dilakukan di penelitian ini.

3.1.1 Analisis Masalah

Proses perancangan sistem diawali oleh tahapan analisis masalah, yaitu tahapan yang dimana akan dilakukan pengidentifikasian permasalahan sehingga menjadi dasar dalam perancangan dan pembangunan sistem yang digunakan pada penelitian ini. Proses melakukan pengidentifikasian masalah dapat dilakukan dengan menerapkan konsep diagram Ishikawa.

Konsep diagram Ishikawa membagi permasalahan dalam penelitian ini menjadi beberapa bagian. Pada bagian material memaparkan bahwa tingginya angka mortalitas penyakit jantung yang menjadi urgensi dilakukannya perancangan sistem. Pada kategori manusia dijelaskan tentang permasalahan yang dialami manusia yaitu hasil diagnosis *Machin Learning* dapat menjadi data tambahan untuk pasien. Pada bagian metode memaparkan bahwa akan dilakukan perbandingan algoritma *Decision Tree* dan *Logistic Regression* dalam mendiagnosis penyakit jantung. Pada bagian sistem menjelaskan dibutuhkanannya pemilihan algoritma yang paling efektif jika diimplementasikan ke sistem diagnosis penyakit jantung.



Gambar 3.1 Diagram Ishikawa

3.1.2 Analisis Kebutuhan

Analisis kebutuhan adalah tahapan analisis yang bertujuan untuk memahami kebutuhan, batasan, serta hal lain yang diperlukan oleh sistem sehingga sistem dapat berfungsi sesuai yang diinginkan. Analisis kebutuhan akan dibagi menjadi beberapa bagian.

3.1.2.1 Kebutuhan Fungsional

Kebutuhan fungsional adalah kebutuhan sistem yang perlu dipenuhi dalam hal kemampuan sistem saat menerima inputan dari luar untuk masuk menuju ke dalam sistem. Berikut adalah kebutuhan fungsional yang diperlukan sistem.

1. Kemampuan sistem dalam memberikan informasi hasil perbandingan kedua algoritma.
2. Kemampuan sistem dalam menerima inputan dataset.
3. Kemampuan sistem dalam melakukan proses perhitungan dengan mengimplementasikan algoritma *Decision Tree* dan *Logistic Regression*.
4. Kemampuan sistem dalam mendiagnosis penyakit jantung berdasarkan inputan data

3.1.2.2 Kebutuhan Non Fungsional

Kebutuhan non fungsional adalah kebutuhan yang berdasarkan pada tingkat batasan sistem yang digunakan pada penelitian ini dalam memberikan layanan dan melakukan tugasnya. Pada penelitian ini, kebutuhan non fungsional adalah sebagai berikut:

- 1 Sistem yang bersangkutan adalah aplikasi *desktop*.
- 2 Sistem yang bersangkutan akan dijalankan secara *offline*.
- 3 Sistem yang bersangkutan dapat digunakan secara gratis.

3.2 Analisis Data

Analisis data merupakan tahap analisis dataset yang akan digunakan dalam penelitian ini. Dataset yang digunakan dalam penelitian ini berupa file csv yang berisi hasil test diagnosis penyakit jantung dari berbagai pasien. Dataset bersumber dari *Open Knowledge Foundation*, yaitu tempat dimana seseorang dapat mendapatkan data yang gratis digunakan. Dataset ini dikumpulkan dengan mengkombinasikan beberapa dataset yang berasal dari lima sumber yang berbeda, yaitu dari Cleveland, Hungarian, Swiss, Long Beach VA, dan Stalog Dataset. Dataset dibuat oleh orang-orang yang memiliki latar belakang akademik kedokteran, yaitu Andras Janosi, M.D., William Steinbrunn, M.D., Matthias Pfisterer, M.D., dan Robert Detrano, M.D., Ph.D.

Dataset berisi hasil tes diagnosis penyakit jantung yang terdiri dari 728 pasien. Dataset tersusun atas 11 variabel independen untuk yang memiliki korelasi dengan ada atau tidaknya penyakit jantung suatu pasien dan satu variabel yang menyatakan ada atau tidaknya penyakit jantung yang dimiliki pasien. Dataset dapat dilihat melalui link <https://github.com/rifkymunir/Tugas-Akhir-S1/blob/main/heartAfterCleaning.csv>.

Dataset memiliki beberapa variabel yang bertipe data non numerik. Untuk dapat diproses oleh algoritma *Machine Learning*, pada prosesnya nanti data akan di *encode* menjadi data numerik. Variabel yang akan melewati proses *encode* adalah *Sex*, *ChestPainType*, *RestingECG*, *ExerciseAngina*, dan *ST_Slope*. Berikut pada tabel 3.1 daftar variabelnya serta hasil *encoding* variabel yang membutuhkan *encoding* tersebut.

Tabel 3.1 Daftar Variabel

No	Variabel	Deskripsi
1	<i>Age</i>	Umur pasien
2	<i>Sex</i>	Jenis kelamin (0 untuk <i>female</i> , dan 1 untuk <i>male</i>)
3	<i>ChestPainType</i>	Jenis nyeri pada dada (0 untuk <i>Typical Angina</i> (TA), 1 untuk <i>Atypical Angina</i> (ATA), 2 untuk <i>Non-Anginal Pain</i> (NAP), dan 3 untuk <i>Asymptomatic</i> (ASY))
4	<i>RestingBP</i>	Tekanan darah di kondisi badan tidak aktif.
5	<i>Cholesterol</i>	Kadar kolesterol
6	<i>FastingBS</i>	Gula darah setelah puasa (1 untuk <i>FastingBS</i> >120 mg/dl, 0 untuk sebaliknya)
7	<i>RestingECG</i>	Elektrokardiogram di kondisi badan tidak aktif. (0 untuk normal, 1 untuk ST, dan 2 untuk LVH)
8	<i>MaxHR</i>	Detak jantung maksimal
9	<i>ExerciseAngina</i>	Terjadinya <i>Angina</i> pada saat olahraga (0 untuk <i>No</i> (N) dan 1 untuk <i>Yes</i> (Y))
10	<i>Oldpeak</i>	Depresi ST yang diakibatkan oleh olahraga relatif terhadap saat istirahat
11	<i>ST_Slope</i>	<i>Slope</i> saat olahraga maksimum (1 untuk <i>Up</i> (<i>slope</i> menaik), 2 untuk <i>Flat</i> (<i>slope</i> datar), dan 3 untuk <i>Down</i> (<i>slope</i> menurun))
12	<i>HeartDisease</i>	Adanya penyakit jantung pada pasien (1 untuk ada, 0 untuk tidak ada/normal)

Berikut ditampilkan detail salah satu data pasien yang telah di-*encode*.

Tabel 3.2 Contoh Salah Satu Data Pasien

No	Variabel	Nilai
1	<i>Age</i>	40
2	<i>Sex</i>	1
3	<i>Chest Pain Type</i>	1
4	<i>RestingBP</i>	140
5	<i>Cholesterol</i>	289
6	<i>FastingBS</i>	0
7	<i>RestingECG</i>	0
8	<i>Max HR</i>	172
9	<i>Exercise</i>	0

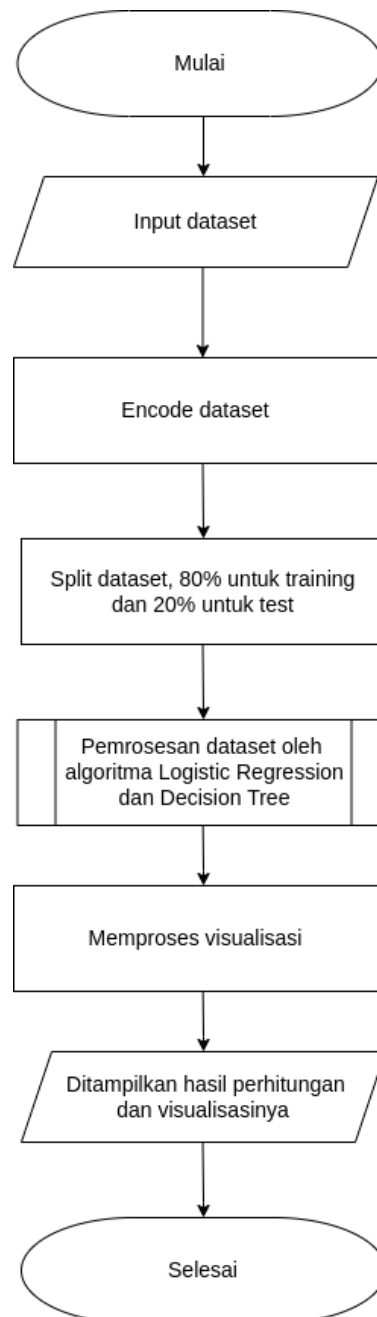
	<i>Angina</i>	
10	<i>Old peak</i>	0
11	<i>ST_Slope</i>	1
12	HeartDisease	0

3.3 Flowchart

Flowchart merupakan visualisasi berupa susunan diagram-diagram yang dibuat untuk menjelaskan urutan-urutan pekerjaan yang dilakukan dalam suatu sistem. Berikut ini ditampilkan *flowchart* dari sistem yang akan dirancang untuk digunakan di penelitian ini serta *flowchart* untuk kedua algoritma *machine learning* yang akan digunakan dalam sistem tersebut.

3.3.1 Flowchart Sistem

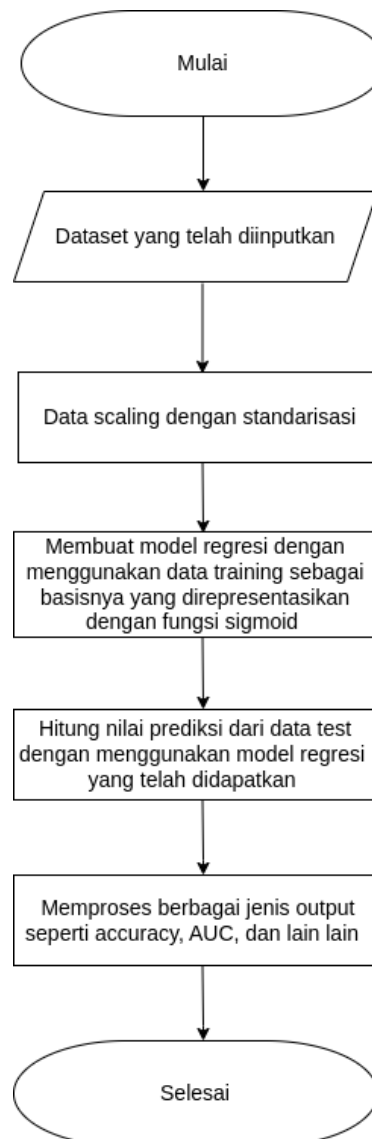
Pertama, akan dilakukan penginputan dataset dengan tipe data file csv. Kemudian, dataset akan melewati proses *encoding*, yaitu mengubah data bertipe non numerik menjadi numerik agar dapat diproses oleh sistem. Selanjutnya, dataset akan melewati proses *split*, yaitu membagi dataset menjadi data *training* dan data *test*. 80% dari dataset akan disimpan sebagai data *training*, sedangkan sisanya disimpan sebagai data *test*. Selanjutnya, dataset yang sudah melewati proses *split* akan diproses dengan kedua algoritma *Machine Learning*. Kemudian, akan diproses visualisasinya berdasarkan hasil yang didapatkan dari pemrosesan kedua algoritma *machine learning*. Kemudian, hasil perhitungan dan visualisasinya akan ditampilkan.



Gambar 3.2 *Flowchart* Sistem

3.3.2 Flowchart Algoritma Logistic Regression

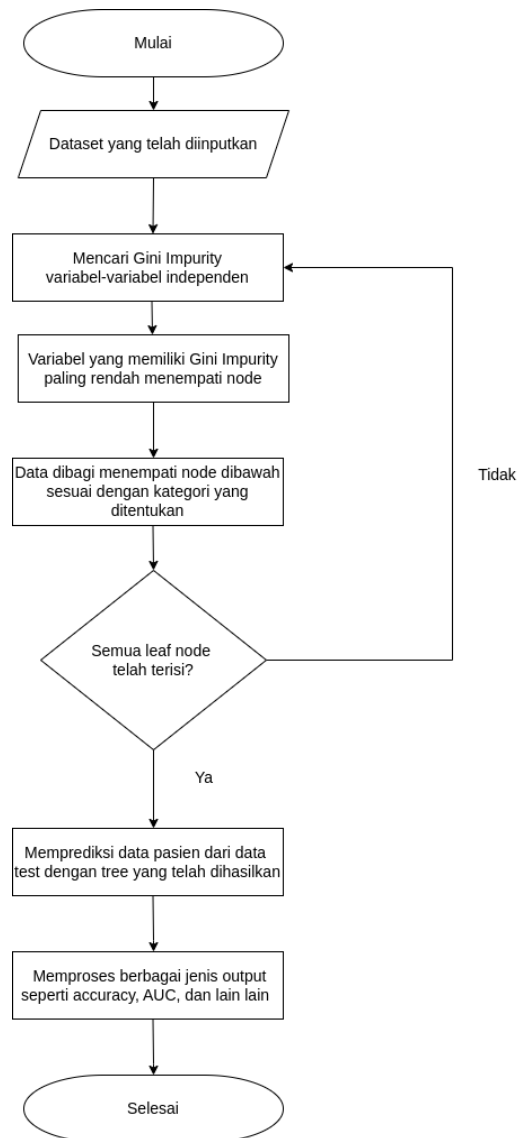
Dataset yang telah diinput akan dikirim untuk diproses. Sebelum diproses oleh algoritma *Logistic Regression*, dataset akan melalui proses *scaling* terlebih dahulu. Proses *scaling* akan dilakukan dengan menerapkan konsep standarisasi dataset. Setelah distandarisasi data dari dataset sudah bisa diproses oleh algoritma *Logistic Regression*. Data training akan digunakan sebagai dasar untuk membuat model regresi. Model regresi akan menjelaskan hubungan antara variabel independen dengan dependen yang direpresentasikan dengan fungsi sigmoid. Setelah itu, akan dicari nilai prediksi dari data *test* dengan menggunakan model regresi yang telah didapatkan. Setelah itu, maka akan diproses *output-output* seperti *accuracy*, *Confusion Matrix*, dan lain-lain.



Gambar 3.3 Flowchart Algoritma Logistic Regression

3.3.3 Flowchart Algoritma Decision Tree

Dataset yang telah diinput akan dikirim ke algoritma *Decision Tree* untuk diproses. Pemrosesan akan diawali dengan mencari *Gini Impurity* dari setiap variabel independen yang ada di dalam data *training*. Kemudian, akan dipilih variabel dengan *Gini Impurity* paling rendah untuk menempati *node*. Kemudian, data dibagi menempati *node* yang berada di bawah sesuai dengan kategori yang telah ditentukan. Proses ini diulang hingga *Decision Tree* tidak membutuhkan *node* baru. *Decision Tree* akhirnya sudah dibuat. *Decision Tree* yang telah dibuat akan digunakan untuk memprediksi data dari *data test*. Setelah diprediksi, berbagai jenis output akan dihasilkan seperti *accuracy*, *AUC*, dan lain-lain.

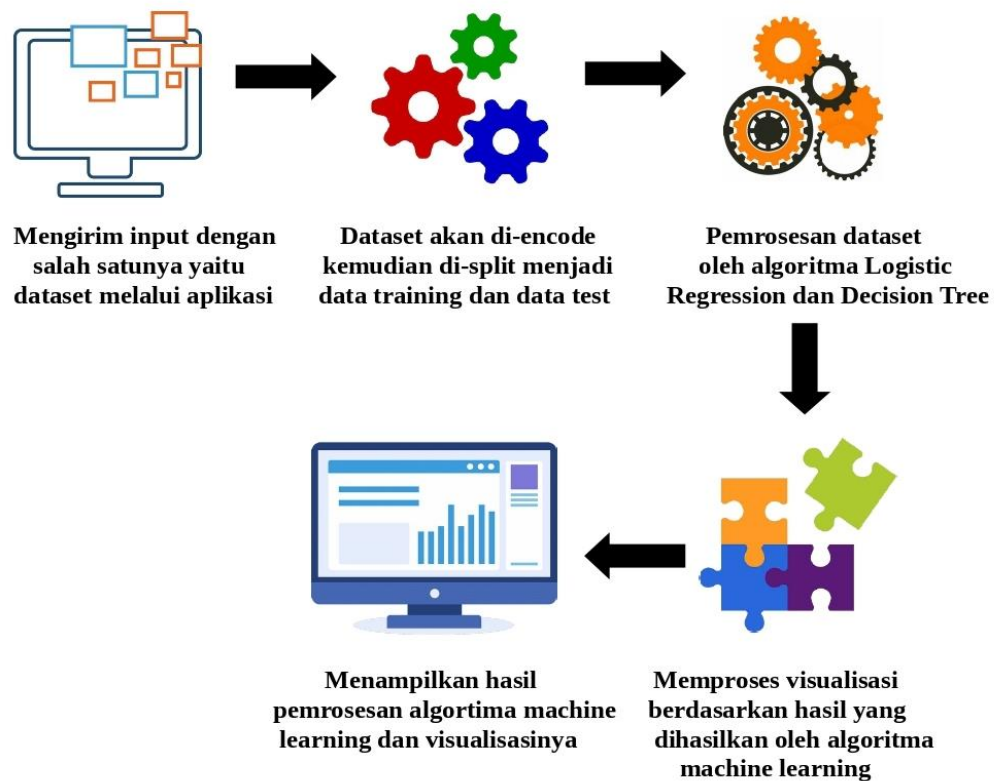


Gambar 3.4 Flowchart Algoritma Decision Tree

3.4 Perancangan Sistem

Perancangan sistem adalah tahapan membuat model suatu sistem yang terstruktur dari bagian-bagian dari sistem yang akan diciptakan, serta bagaimana interaksi model tersebut dengan pengguna sistem. Berikut akan dijelaskan bagaimana sistem akan dirancang.

3.4.1 Arsitektur Umum Sistem



Gambar 3.5 Arsitektur Umum Sistem

Pertama, pengguna akan memasukkan input yang diminta oleh aplikasi dengan dataset adalah salah satu inputan yang diminta. Kemudian, dataset akan melalui proses *encoding* agar dapat diproses oleh algoritma *Machine Learning*, serta melalui proses *split* agar dibedakan antara data *training* dan data *test*. Kemudian, barulah dataset akan diproses oleh algoritma *Machine Learning*. Lalu, visualisasi akan dibuat berdasarkan hasil yang didapatkan oleh pemrosesan algoritma *Machine Learning* tersebut. Setelah itu, barulah ditampilkan hasil pemrosesan algoritma *Machine Learning* serta visualisasinya.

3.4.2 Use Case Diagram

Use Case Diagram merupakan diagram yang memvisualisasikan hubungan antara sistem dengan pengguna agar memberikan penjelasan tentang bagaimana penggunaan sistem dari sisi seorang pengguna. Berikut pada gambar 3.6 *Use Case Diagram* sistem yang digunakan pada penelitian ini.

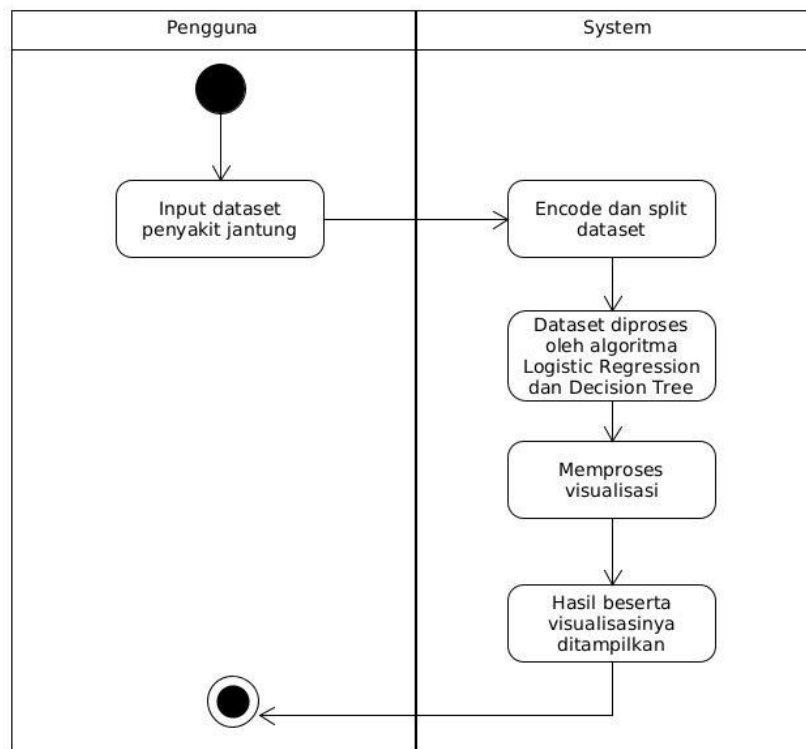


Gambar 3.6 *Use Case Diagram*

Pengguna akan menginputkan dataset penyakit jantung. Setelah itu, pengguna tinggal menekan tombol proses dan input akan diproses dan akan ditampilkan hasilnya.

3.4.3 Activity Diagram

Activity Diagram adalah diagram yang memberikan penjelasan mengenai bagaimana proses yang terjadi di dalam sebuah sistem yang digunakan di penelitian ini. Berikut pada gambar 3.7 divisualisasikan *Activity Diagram* sistem.

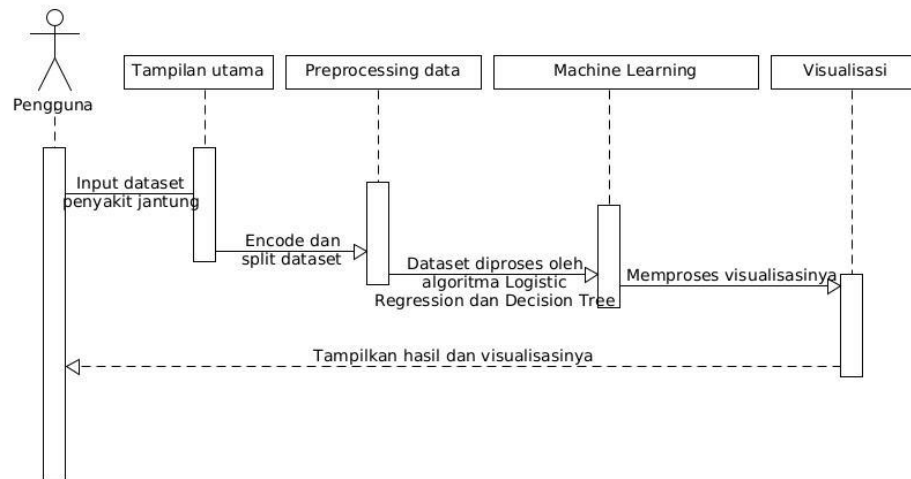


Gambar 3.7 *Activity Diagram*

Pengguna akan mengirim dataset penyakit jantung ke program. Kemudian, dataset akan melalui proses *encode* dan *split*. Kemudian, dataset akan diproses oleh algoritma *Logistic Regression* dan *Decision Tree*. Kemudian, visualisasinya akan diproses. Setelah diproses, maka akan ditampilkan hasil dan visualisasinya.

3.4.4 *Sequence Diagram*

Sequence Diagram merupakan diagram yang memvisualisasikan bagaimana pengguna dan objek-objek dalam sistem yang digunakan di penelitian ini berinteraksi yang digambarkan menurut urutan tertentu. Berikut pada gambar 3.8 *Sequence Diagram* sistem yang digunakan pada penelitian ini.

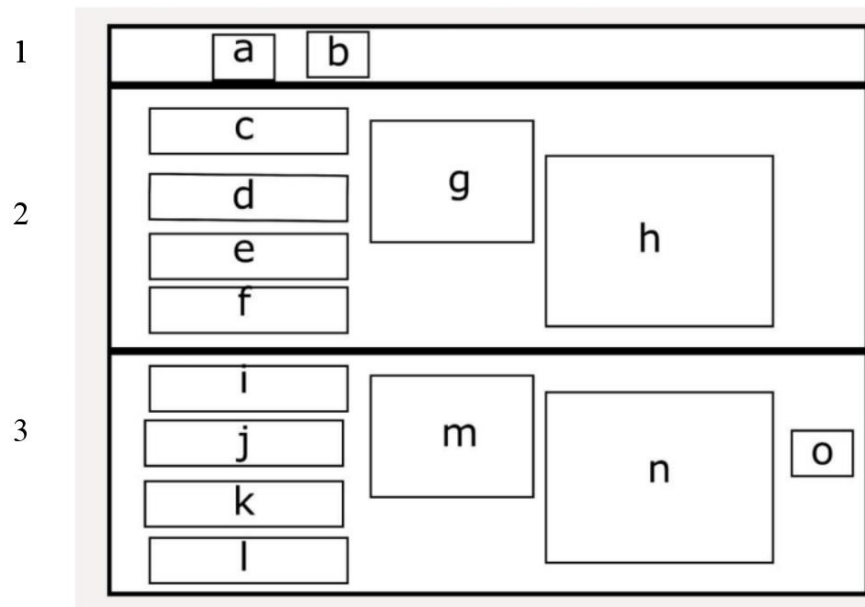


Gambar 3.8 *Sequence Diagram*

Terdapat beberapa tahap terjadi antara pengguna dengan aplikasi. Pada tahap awal pengguna akan menginput dataset penyakit jantung ke sistem. Setelah itu, dataset yang telah diinput akan melalui proses *encode* dan *split*. Kemudian, dataset akan diproses oleh algoritma *Logistic Regression* dan *Decision Tree*. Setelah itu, aplikasi akan membuat visualisasi berdasarkan hasil yang telah didapat. Setelah itu aplikasi akan menampilkan hasil serta visualisasinya kepada pengguna.

3.5 Perancangan *Interface*

Perancangan *interface* adalah tahap mendesain halaman sistem sesuai dengan kebutuhan yang dibutuhkan sistem dalam melaksanakan tugasnya. Rancangan *interface* pada sistem yang digunakan untuk penelitian ini akan menggunakan tampilan yang sederhana. Tampilan akan dibagi menjadi tiga bagian, yaitu bagian untuk menginputkan data, bagian hasil proses dari *Logistic Regression*, dan bagian hasil proses dari *Decision tree*.



Gambar 3.9 Rancangan *Interface*

Keterangan rancangan *interface* ditunjukkan pada tabel 3.3.

Tabel 3.3 Penjelasan Rancangan *Interface*

No	Nama	Penjelasan
1	Input data	(a) Bagian untuk menginputkan data. (b) Button untuk memproses perhitungan.
2	<i>Logistic Regression Section</i>	Bagian untuk menampilkan hasil proses dari <i>Logistic Regression</i> . Akan ditampilkan nilai: (c) <i>Accuracy</i> . (d) AUC. (e) <i>G-Means</i> . (f) <i>k-Fold</i> (g) <i>Confusion Matrix</i> . (h) Grafik ROC.
3	<i>Decision Tree Section</i>	Bagian untuk menampilkan hasil proses dari <i>Decision Tree</i> . Akan ditampilkan nilai: (i) <i>Accuracy</i> . (j) AUC. (k) <i>G-Means</i> . (l) <i>k-Fold</i> (m) <i>Confusion Matrix</i> . (n) Grafik ROC. <i>Accuracy</i> (o) Tempat menginputkan <i>depth</i> maksimal <i>tree</i> .

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma

Implementasi algoritma yang digunakan adalah menggunakan bahasa *Python* serta dengan menggunakan *library* yang telah disediakan oleh *Scikit-learn*. Berikut akan dijelaskan tentang implementasinya.

4.1.1 Data Scaling

Data *scaling* bertujuan bertujuan agar variabel memiliki skala yang sama, karena dataset memiliki variabel-variabel dengan *range* yang bermacam-macam. Data *scaling* berguna untuk algoritma *Logistic Regression* karena algoritma *Logistic Regression* bisa menghasilkan hasil yang lebih baik jika diubah skala datanya.

Scaling yang akan diterapkan pada dataset adalah standarisasi. Standarisasi akan mengubah nilai rata-rata variabel menjadi nol dan standar deviasi menjadi satu. Berikut adalah proses *scaling* dengan standarisasi.

Tabel 4.1 Nilai Data *Training* Sebelum Standarisasi

No	Variabel	Pasien 1	Pasien 2	Pasien 3	...	Pasien 580	Pasien 581	Pasien 582
1	<i>Age</i>	47	63	43	...	55	54	43
2	<i>Sex</i>	1	0	0	...	1	1	1
3	<i>ChestPain Type</i>	2	1	1	...	3	2	3
4	<i>RestingBP</i>	108	140	120	...	160	150	150
5	<i>Cholesterol</i>	243	195	266	...	289	232	247
6	<i>FastingBS</i>	0	0	0	...	0	0	0
7	<i>Resting ECG</i>	0	0	0	...	2	2	0
8	<i>MaxHR</i>	152	179	118	...	145	165	171
9	<i>Exercise Angina</i>	0	0	0	...	1	0	0
10	<i>Oldpeak</i>	0.0	0.0	0.0	...	0.8	1.6	1.5
11	<i>ST_Slope</i>	1	1	1	...	2	1	1

Terdapat 582 data pasien di dalam data *training*. Setiap variabel independen dari data training akan dicari rata-rata dan standar deviasinya. Untuk menghitungnya dapat menggunakan rumus:

$$\text{Rata-rata } (\bar{x}) = \frac{\sum x}{N} \qquad \text{Standar Deviasi } (\sigma) = \sqrt{\frac{\sum (x - \bar{x})^2}{N}}$$

Dimana : x = nilai variabel independen data pasien di data *training*

N = total data pasien di data *training*

Berikut perhitungannya untuk setiap variabel:

1. Rata-rata

$$\text{Rata-rata Age} = \frac{47+63+43+\dots+55+54+43}{582} = 52.6546$$

$$\text{Rata-rata Sex} = \frac{1+0+0+\dots+1+1+1}{582} = 0.75601$$

$$\text{Rata-rata ChestPainType} = \frac{2+1+1+\dots+3+2+3}{582} = 2.1357$$

$$\text{Rata-rata RestingBP} = \frac{108+140+120+\dots+160+150+150}{582} = 132.8419$$

$$\text{Rata-rata Cholesterol} = \frac{243+195+206+\dots+289+232+247}{582} = 246.4226$$

$$\text{Rata-rata FastingBS} = \frac{0+0+0+\dots+0+0+0}{582} = 0.1666$$

$$\text{Rata-rata RestingECG} = \frac{0+0+0+\dots+2+2+0}{582} = 0.6288$$

$$\text{Rata-rata MaxHR} = \frac{152+179+118+\dots+145+165+171}{582} = 140.9759$$

$$\text{Rata-rata ExerciseAngina} = \frac{0+0+0+\dots+1+0+0}{582} = 0.3659$$

$$\text{Rata-rata Oldpeak} = \frac{0.0+0.0+0.0+\dots+0.8+1.6+1.5}{582} = 0.8503$$

$$\text{Rata-rata ST_Slope} = \frac{1+1+1+\dots+2+1+1}{582} = 1.5635$$

2. Standar Deviasi

$$\text{Standar Deviasi Age} = \sqrt{\frac{(47-52.6546)^2+\dots+(43-52.6546)^2}{582}} = 9.5347$$

$$\text{Standar Deviasi Sex} = \sqrt{\frac{(1-0.75601)^2+\dots+(1-0.75601)^2}{582}} = 0.4298$$

$$\text{Standar Deviasi } ChestPainType = \sqrt{\frac{(2-2.1357)^2 + \dots + (3-2.1357)^2}{582}} = 0.9553$$

$$\text{Standar Deviasi } RestingBP = \sqrt{\frac{(108-132.8419)^2 + \dots + (150-132.8419)^2}{582}} = 17.0206$$

$$\text{Standar Deviasi } Cholesterol = \sqrt{\frac{(243-246.4226)^2 + \dots + (247-246.4226)^2}{582}} = 59.3334$$

$$\text{Standar Deviasi } FastingBS = \sqrt{\frac{(0-0.166667)^2 + \dots + (0-0.166667)^2}{582}} = 0.3729$$

$$\text{Standar Deviasi } RestingECG = \sqrt{\frac{(0-0.628866)^2 + \dots + (0-0.628866)^2}{582}} = 0.8398$$

$$\text{Standar Deviasi } MaxHR = \sqrt{\frac{(152-140.975945)^2 + \dots + (171-140.975945)^2}{582}} = 24.7519$$

$$\text{Standar Deviasi } ExerciseAngina = \sqrt{\frac{(0-0.3659)^2 + \dots + (0-0.3659)^2}{582}} = 0.4821$$

$$\text{Standar Deviasi } Oldpeak = \sqrt{\frac{(0.0-0.8503)^2 + \dots + (1.5-0.8503)^2}{582}} = 1.0360$$

$$\text{Standar Deviasi } ST_Slope = \sqrt{\frac{(1-1.5635)^2 + \dots + (1-1.5635)^2}{582}} = 0.5854$$

Maka telah didapatkan keseluruhan rata-rata dan standar deviasi dari setiap variabel independen.

Tabel 4.2 Nilai Rata-Rata dan Standar Deviasi Tiap Variabel

No	Variabel	Rata-rata	Standar Deviasi
1	<i>Age</i>	52.6546	9.5347
2	<i>Sex</i>	0.75601	0.42985
3	<i>ChestPainType</i>	2.1357	0.9553
4	<i>RestingBP</i>	132.8419	17.0206
5	<i>Cholesterol</i>	246.4226	59.3334
6	<i>FastingBS</i>	0.1666	0.3729
7	<i>Resting ECG</i>	0.6288	0.8398
8	<i>MaxHR</i>	140.9759	24.7519
9	<i>Exercise Angina</i>	0.3659	0.4821
10	<i>Oldpeak</i>	0.8503	1.0360
11	<i>ST_Slope</i>	1.5635	0.5854

Setelah mendapatkan rata-rata dan standar deviasi setiap variabel independen, maka sudah bisa kita dapatkan nilai yang telah distandarisasi. Rumus mendapatkan nilai standarisasi adalah:

$$x' = \frac{x - \bar{x}}{\sigma}$$

Misalkan penstandarisasi data untuk pasien 1 pada data *training*, maka

$$x' \text{ untuk } Age = \frac{47 - 52.6546}{9.5347} = -0.5930$$

$$x' \text{ untuk } Sex = \frac{1 - 0.75601}{0.4298} = 0.5680$$

$$x' \text{ untuk } ChestPainType = \frac{2 - 2.1357}{0.9553} = -0.1422$$

$$x' \text{ untuk } RestingBP = \frac{108 - 132.8419}{17.0206} = -1.460$$

$$x' \text{ untuk } Cholesterol = \frac{243 - 246.4226}{59.3334} = -0.0576$$

$$x' \text{ untuk } FastingBS = \frac{0 - 0.1666}{0.3729} = -0.4472$$

$$x' \text{ untuk } RestingECG = \frac{0 - 0.6288}{0.8398} = -0.749$$

$$x' \text{ untuk } MaxHR = \frac{152 - 140.9759}{24.7519} = 0.4453$$

$$x' \text{ untuk } ExerciseAngina = \frac{0 - 0.3659}{0.4821} = -0.7589$$

$$x' \text{ untuk } Oldpeak = \frac{0.0 - 0.8503}{1.0360} = -0.8214$$

$$x' \text{ untuk } ST_Slope = \frac{1 - 1.5635}{0.5854} = -0.9634$$

Berikut pada tabel 4.3 ditampilkan nilai-nilai dari data *training* yang telah melewati proses *scaling* dengan standarisasi data.

Tabel 4.3 Nilai Data *Training* Setelah Standarisasi

No	Varia bel	Pasien 1	Pasien 2	Pasien 3	...	Pasien 580	Pasien 581	Pasien 582
1	<i>Age</i>	-0.5930	1.0859	-1.013	...	0.2461	0.1412	-1.013
2	<i>Sex</i>	0.5680	-1.7602	-1.7602	...	0.5680	0.5680	0.5680
3	<i>Chest Pain Type</i>	-0.1422	-1.1898	-1.1898	...	0.9054	-0.1422	0.9054
4	<i>Restin</i>	-1.460	0.4209	-0.7551	...	1.5969	1.0089	1.0089

	<i>g BP</i>							
5	<i>Cholesterol</i>	-0.0576	-0.8674	0.3302	...	0.7182	-0.2432	0.0097
6	<i>Fasting BS</i>	-0.4472	-0.4472	-0.4472	...	-0.4472	-0.4472	-0.4472
7	<i>Resting ECG</i>	-0.74940	-0.74940	-0.74940	...	1.6339	1.6339	-0.74940
8	<i>Max HR</i>	0.4453	1.537	-0.9290	...	0.1627	0.9714	1.2140
9	<i>Exercise Angina</i>	-0.7597	-0.7597	-0.7597	...	1.3162	-0.7597	-0.7597
10	<i>Old peak</i>	-0.8214	-0.8214	-0.8214	...	-0.0486	0.7242	0.6276
11	<i>ST_Slope</i>	-0.9634	-0.9634	-0.9634	...	0.7460	-0.9634	-0.9634

Proses yang sama akan dilalui oleh data test yang menyimpan 146 data pasien. Namun, akan digunakan nilai rata-rata dan standar deviasinya dari sudah didapatkan dari data *training*. Berikut pada tabel 4.4 nilai dari data *test* sebelum distandarisasi dan tabel 4.5 setelah selesai distandarisasi

Tabel 4.4 Nilai Data *Test* Sebelum Standarisasi

No	Variabel	Pasien 1	Pasien 2	Pasien 3	...	Pasien 144	Pasien 145	Pasien 146
1	<i>Age</i>	58	67	58	...	55	39	60
2	<i>Sex</i>	1	0	1	...	1	1	0
3	<i>ChestPain Type</i>	2	3	3	...	3	2	2
4	<i>RestingBP</i>	132	106	128	...	140	160	102
5	<i>Cholesterol</i>	224	223	259	...	201	147	318
6	<i>FastingBS</i>	0	0	0	...	0	1	0
7	<i>Resting ECG</i>	2	0	2	...	0	0	0
8	<i>MaxHR</i>	173	142	130	...	130	160	160
9	<i>Exercise Angina</i>	0	0	1	...	1	0	0
10	<i>Oldpeak</i>	3.2	0.3	3	...	3	0	0
11	<i>ST_Slope</i>	1	1	2	...	2	1	1

Tabel 4.5 Nilai Data *Test* Setelah Standarisasi

No	Varia bel	Pasien 1	Pasien 2	Pasien 3	...	Pasien 144	Pasien 145	Pasien 146
1	<i>Age</i>	0.5611	1.5058	0.5611	...	0.2461	-1.4333	0.7710
2	<i>Sex</i>	0.5680	-1.7602	0.5680	...	0.5680	0.5680	-1.7602
3	<i>Chest Pain Type</i>	-0.1422	0.9054	0.9054	...	0.9054	-0.1422	-0.1422
4	<i>Restin g BP</i>	-0.0495	-1.5783	-0.2847	...	0.4209	1.5969	-1.8135
5	<i>Choles terol</i>	-0.3782	-0.3951	0.2121	...	-0.7662	-1.6771	1.2073
6	<i>Fastin g BS</i>	-0.4472	-0.4472	-0.4472	...	-0.4472	2.2360	-0.4472
7	<i>Restin g ECG</i>	1.6339	-0.7494	1.6339	...	-0.7494	-0.7494	-0.7494
8	<i>Max HR</i>	1.2949	0.0414	-0.4438	...	-0.4438	0.7692	0.7692
9	<i>Exerci se Angina</i>	-0.7597	-0.7597	1.3162	...	1.3162	-0.7597	-0.7597
10	<i>Old peak</i>	2.2699	-0.5316	2.0767	...	2.0767	-0.8214	-0.8214
11	<i>ST_ Slope</i>	-0.9634	-0.9634	0.7460	...	0.7460	-0.9634	-0.9634

Data telah selesai melalui proses *scaling*. Data siap untuk diproses oleh algoritma *Logistic Regression*.

4.1.2 Implementasi Algoritma *Logistic Regression*

Tahap dalam melakukan implementasi algoritma *Logistic Regression* akan diawali dengan menciptakan model regresi dengan menentukan hubungan antara variabel independen dengan variabel dependen yang dimana hubungan tersebut akan direpresentasikan dengan fungsi sigmoid.

$$f(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_{11} x_{11})}}$$

Dimana : β_0 = nilai *intercept*

$\beta_1, \beta_2, \dots, \beta_{11}$ = nilai koefisien setiap variabel independen

x_1, x_2, \dots, x_{11} = nilai variabel independen (yang telah melalui proses *scaling*)

Berikut pada tabel 4.6 ditampilkan nilai *intercept* dan koefisien setiap variabel independen.

Tabel 4.6 Nilai *Intercept* dan Koefisien

No	Variabel	Koefisien
1	<i>Intercept</i>	-0.28018385
2	<i>Age</i>	0.25765259
3	<i>Sex</i>	0.6390607
4	<i>ChestPainType</i>	0.55359806
5	<i>RestingBP</i>	0.12952012
6	<i>Cholesterol</i>	0.22780964
7	<i>FastingBS</i>	0.08745907
8	<i>Resting ECG</i>	0.18482642
9	<i>MaxHR</i>	-0.17984194
10	<i>Exercise Angina</i>	0.58003293
11	<i>Oldpeak</i>	0.27800528
12	<i>ST_Slope</i>	1.0141951

Setelah kita mendapatkan nilai *intercept* dan koefisien setiap variabel independen, maka model telah selesai dibuat. Maka kita dapat masuk ke tahap selanjutnya yaitu melakukan prediksi. Kita ambil salah satu data pasien dari data *test* yang telah melalui proses *scaling*, beserta nilai variabel *HeartDisease* yang menentukan ada atau tidaknya penyakit jantung pada pasien.

Tabel 4.7 Sampel Salah Satu Data Test

No	Variabel	Nilai data pasien
1	<i>Age</i>	0.5611
2	<i>Sex</i>	0.5680
3	<i>Chest Pain Type</i>	-0.1422
4	<i>Resting BP</i>	-0.0495
5	<i>Cholesterol</i>	-0.3782
6	<i>FastingBS</i>	-0.4472
7	<i>RestingECG</i>	1.6339

8	<i>Max HR</i>	1.2949
9	<i>Exercise Angina</i>	-0.7597
10	<i>Old peak</i>	2.2699
11	<i>ST_Slope</i>	-0.9634
12	<i>HeartDisease</i>	1

Dengan mengaplikasikan model, maka kita akan mendapatkan nilai prediksi dari data pasien.

$$\text{Nilai prediksi} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_{11} x_{11})}}$$

Misalkan $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_{11} x_{11} = n$, maka:

$$\begin{aligned} n &= (-0.28018385) + 0.25765259x_0.5611 + 0.6390607x_0.5680 + \\ &\quad 0.55359806x(-0.1422) + 0.12952012x(-0.0495) + 0.22780964x(-0.3782) \\ &\quad + 0.08745907x(-0.4472) + 0.18482642x1.6339 + (-0.17984194)x1.2949 \\ &\quad + 0.58003293x(-0.7597) + 0.27800528x2.2699 + 1.0141951x(-0.9634) \\ &= -0.7006025278 \end{aligned}$$

$$e^{-n} = 2,7182818^{0.7006025278} = 2.014966415$$

$$\text{Nilai prediksi} = \frac{1}{1 + 2.014966415} = 0.3316786532$$

Karena nilai prediksi < 0.5 , maka data pasien akan diprediksi *false*. Karena *HeartDisease* = 1 dan 1 menyatakan nilai *true*, maka hasil prediksi tidak sesuai.

4.1.3 Implementasi Algoritma *Decision Tree*

Algoritma *Decision Tree* di *Scikit-learn* akan melibatkan angka acak pada perhitungannya. Sehingga, di bawah ini akan dijelaskan gambaran dasar prosesnya di *Scikit-learn* jika diabaikan perhitungan angka acak pada prosesnya.

Implementasi algoritma *Decision Tree* diawali dengan mencari *root node*. Tahap awal dalam mencari *node* adalah dengan mencari distribusi setiap variabel independen di data *training* yang dibagi menjadi kategori-kategori yang dimana setiap kategorinya akan dicari berapa yang memiliki *HeartDisease* = 0 dan *HeartDisease* = 1.

Tabel 4.8 Distribusi Variabel Independen

No	Variabel	Kategori	Jumlah	<i>HeartDisease</i> =0	<i>HeartDisease</i> =1
1	<i>Age</i>	$\leq 54,5$	326	218	108
		$> 54,5$	256	98	158
2	<i>Sex</i>	$\leq 0,5$	142	108	34
		$> 0,5$	440	208	232
3	<i>ChestPain Type</i>	$\leq 2,5$	303	238	65
		$> 2,5$	279	78	201
4	<i>RestingBP</i>	$\leq 140,5$	449	267	182
		$> 140,5$	133	49	84
5	<i>Cholesterol</i>	$\leq 257,5$	362	220	142
		$> 257,5$	220	96	124
6	<i>FastingBS</i>	$\leq 0,5$	485	282	203
		$> 0,5$	97	34	63
7	<i>RestingECG</i>	$\leq 0,5$	353	217	136
		$> 0,5$	229	99	130
8	<i>MaxHR</i>	$\leq 134,5$	224	68	156
		$> 134,5$	358	248	110
9	<i>Exercise Angina</i>	$\leq 0,5$	369	275	94
		$> 0,5$	213	41	172
10	<i>Oldpeak</i>	$\leq 0,85$	329	248	81
		$> 0,85$	253	68	185
11	<i>ST_Slope</i>	$\leq 1,5$	282	247	37
		$> 1,5$	300	69	231

Seperti yang bisa dilihat di tabel, setiap variabel memiliki akan dibagi atas dua kategori. Kategori ini dihasilkan oleh algoritma bawaan dari *Scikit-learn*. Ini tidak seperti algoritma *Decision Tree* yang biasanya digunakan yang dimana jika ada variabelnya yang sudah memiliki kategori, maka akan digunakan kategori yang sudah ditentukan tersebut.

Langkah selanjutnya adalah menentukan *Gini Impurity* setiap variabel. Untuk mencarinya diperlukan *Gini Impurity* setiap kategori dalam suatu variabel. Berikut adalah perhitungannya:

1. Age

$$a. \text{ Gini Impurity Age (} \leq 54,5) = 1 - \left(\frac{218}{316}\right)^2 - \left(\frac{108}{316}\right)^2 = 0.443072$$

$$b. \text{ Gini Impurity Age (} > 54,5) = 1 - \left(\frac{98}{256}\right)^2 - \left(\frac{158}{256}\right)^2 = 0.472534$$

$$c. \text{ Gini Impurity Age} = \frac{316}{582} \times 0.443072 + \frac{256}{582} \times 0.472534 \\ = 0.448418$$

2. Sex

$$a. \text{ Gini Impurity Sex (} \leq 0,5) = 1 - \left(\frac{108}{142}\right)^2 - \left(\frac{34}{142}\right)^2 = 0.364213$$

$$b. \text{ Gini Impurity Sex (} > 0,5) = 1 - \left(\frac{208}{440}\right)^2 - \left(\frac{232}{440}\right)^2 = 0.498512$$

$$c. \text{ Gini Impurity Age} = \frac{142}{582} \times 0.364213 + \frac{440}{582} \times 0.498512 \\ = 0.465744$$

3. ChestPainType

$$a. \text{ Gini Impurity ChestpainType (} \leq 2,5) = 1 - \left(\frac{238}{303}\right)^2 - \left(\frac{65}{303}\right)^2 \\ = 0.337003$$

$$b. \text{ Gini Impurity ChestpainType (} > 2,5) = 1 - \left(\frac{78}{279}\right)^2 - \left(\frac{201}{279}\right)^2 \\ = 0.402821$$

$$c. \text{ Gini Impurity ChestpainType} = \frac{303}{582} \times 0.337003 + \frac{279}{582} \times 0.402821 \\ = 0.368554$$

4. RestingBP

$$a. \text{ Gini Impurity RestingBP (} \leq 140,5) = 1 - \left(\frac{267}{449}\right)^2 - \left(\frac{182}{449}\right)^2 \\ = 0.482080$$

$$b. \text{ Gini Impurity RestingBP (} > 140,5) = 1 - \left(\frac{49}{133}\right)^2 - \left(\frac{84}{133}\right)^2 \\ = 0.465373$$

$$c. \text{ Gini Impurity RestingBP} = \frac{449}{582} \times 0.482080 + \frac{133}{582} \times 0.465373$$

$$= 0.478262$$

5. Cholesterol

$$a. \text{ Gini Impurity Cholesterol (} \leq 257,5) = 1 - \left(\frac{220}{362}\right)^2 - \left(\frac{142}{362}\right)^2$$

$$= 0.476786$$

$$b. \text{ Gini Impurity Cholesterol (} > 257,5) = 1 - \left(\frac{96}{220}\right)^2 - \left(\frac{124}{220}\right)^2$$

$$= 0.491900$$

$$c. \text{ Gini Impurity Cholesterol} = \frac{362}{582} \times 0.476786 + \frac{220}{582} \times 0.491900 = 0.482499$$

6. FastingBS

$$a. \text{ Gini Impurity FastingBS (} \leq 0,5) = 1 - \left(\frac{282}{485}\right)^2 - \left(\frac{203}{485}\right)^2$$

$$= 0.486733$$

$$b. \text{ Gini Impurity FastingBS (} > 0,5) = 1 - \left(\frac{34}{97}\right)^2 - \left(\frac{63}{97}\right)^2$$

$$= 0.455308$$

$$c. \text{ Gini Impurity FastingBS} = \frac{485}{582} \times 0.486733 + \frac{97}{582} \times 0.455308 = 0.481495$$

7. RestingECG

$$a. \text{ Gini Impurity RestingECG (} \leq 0,5) = 1 - \left(\frac{217}{353}\right)^2 - \left(\frac{136}{353}\right)^2$$

$$= 0.473673$$

$$b. \text{ Gini Impurity RestingECG (} > 0,5) = 1 - \left(\frac{99}{229}\right)^2 - \left(\frac{130}{229}\right)^2$$

$$= 0.490837$$

$$c. \text{ Gini Impurity RestingECG} = \frac{353}{582} \times 0.473673 + \frac{229}{582} \times 0.490837 = 0.480426$$

8. MaxHR

$$a. \text{ Gini Impurity MaxHR (} \leq 134,5) = 1 - \left(\frac{68}{224}\right)^2 - \left(\frac{156}{224}\right)^2$$

$$= 0.422831$$

$$\text{b. Gini Impurity MaxHR (} > 134,5) = 1 - \left(\frac{248}{358}\right)^2 - \left(\frac{110}{358}\right)^2$$

$$= 0.425704$$

$$\text{c. Gini Impurity MaxHR} = \frac{224}{582} \times 0.422831 + \frac{358}{582} \times 0.425704$$

$$= 0.424598$$

9. ExerciseAngina

$$\text{a. Gini Impurity ExerciseAngina (} \leq 0,5) = 1 - \left(\frac{275}{369}\right)^2 - \left(\frac{94}{369}\right)^2$$

$$= 0.379697$$

$$\text{b. Gini Impurity ExerciseAngina (} > 0,5) = 1 - \left(\frac{41}{213}\right)^2 - \left(\frac{172}{213}\right)^2$$

$$= 0.310873$$

$$\text{c. Gini Impurity ExerciseAngina} = \frac{369}{582} \times 0.379697 + \frac{213}{582} \times 0.310873$$

$$= 0.354508$$

10. Oldpeak

$$\text{a. Gini Impurity Oldpeak (} \leq 0,85) = 1 - \left(\frac{248}{329}\right)^2 - \left(\frac{81}{329}\right)^2$$

$$= 0.371171$$

$$\text{b. Gini Impurity Oldpeak (} > 0,85) = 1 - \left(\frac{68}{253}\right)^2 - \left(\frac{185}{253}\right)^2$$

$$= 0.393069$$

$$\text{c. Gini Impurity Oldpeak} = \frac{329}{582} \times 0.371171 + \frac{253}{582} \times 0.393069$$

$$= 0.380690$$

11. ST_Slope

$$\text{a. Gini Impurity ST_Slope (} \leq 1,5) = 1 - \left(\frac{247}{282}\right)^2 - \left(\frac{35}{282}\right)^2$$

$$= 0.217418$$

$$\text{b. Gini Impurity ST_Slope (} > 1,5) = 1 - \left(\frac{69}{300}\right)^2 - \left(\frac{231}{300}\right)^2$$

$$= 0.354200$$

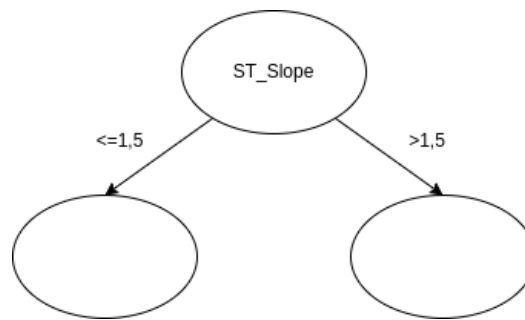
$$\begin{aligned} \text{c. Gini Impurity } ST_Slope &= \frac{282}{582} \times 0.217418 + \frac{300}{582} \times 0.354200 \\ &= 0.287924 \end{aligned}$$

Berikut adalah tabel 4.9 untuk list untuk *Gini Impurity* setiap variabel.

Tabel 4.9 Nilai *Gini Impurity* Setiap Variabel

No	Variabel	<i>Gini Impurity</i>
1	<i>Age</i>	0.448418
2	<i>Sex</i>	0.465744
3	<i>ChestPainType</i>	0.368554
4	<i>RestingBP</i>	0.478262
5	<i>Cholesterol</i>	0.482499
6	<i>FastingBS</i>	0.481495
7	<i>Resting ECG</i>	0.480426
8	<i>MaxHR</i>	0.424598
9	<i>Exercise Angina</i>	0.354508
10	<i>Oldpeak</i>	0.380690
11	<i>ST_Slope</i>	0.287924

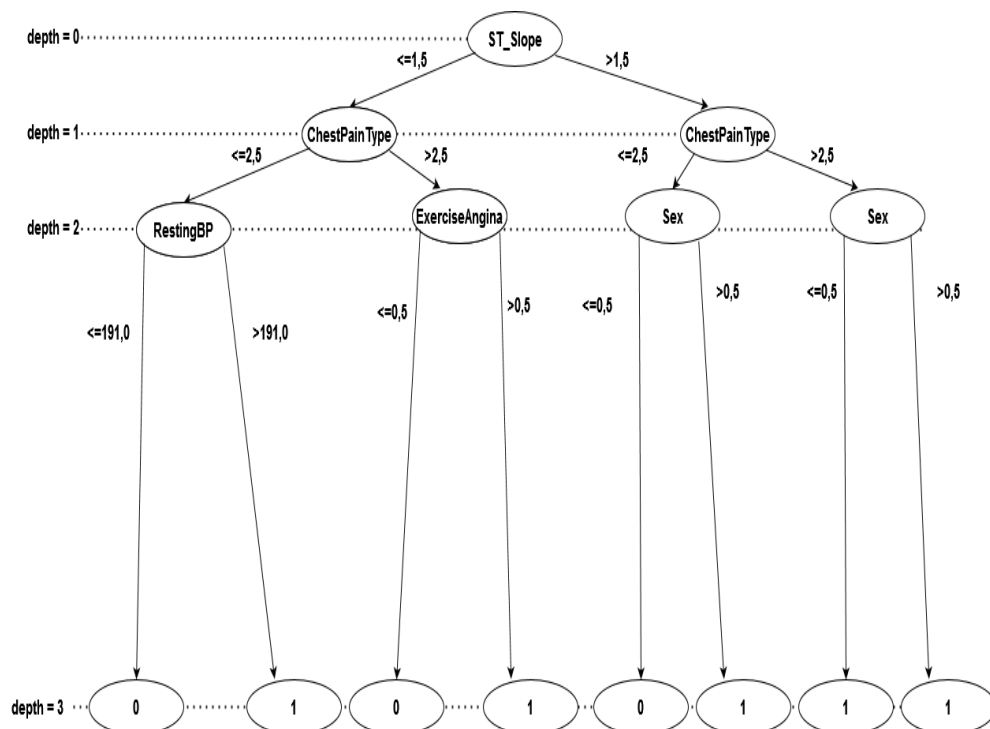
Root node akan diisi oleh variabel dengan *Gini Impurity* paling rendah. Oleh karena itu, *root node* akan diisi oleh *ST_Slope*.



Gambar 4.1 Bentuk Awal *Decision Tree*

Data pasien akan dibagi menempati *node* sesuai dengan kategori yang *root node* telah pisahkan. Kemudian, di *node* yang telah ditempati akan dicari ulang *Gini Impurity* yang baru, kemudian variabel yang memiliki *Gini impurity* paling rendah akan menempati *node*. Kemudian data akan dibagi lagi menempati *node* yang baru dan proses yang telah dilakukan sebelumnya akan diulang sampai pemrosesan mencapai di *leaf node*.

Berikut pada gambar 4.2 yaitu *Decision Tree* yang diatur *depth* maksimalnya 3.



Gambar 4.2 *Decision Tree* dengan *Depth* Maksimal Sebesar 3

Karena *Decision Tree* telah selesai diciptakan, maka program sudah dapat melakukan prediksi. Misalkan salah satu dari data *test* akan kita prediksi.

Tabel 4.10 Sampe Salah Satu Data *Test*

No	Variabel	Nilai data pasien
1	<i>Age</i>	67
2	<i>Sex</i>	0
3	<i>Chest Pain Type</i>	3
4	<i>RestingBP</i>	106
5	<i>Cholesterol</i>	223
6	<i>FastingBS</i>	0
7	<i>RestingECG</i>	0
8	<i>Max HR</i>	142
9	<i>Exercise Angina</i>	0
10	<i>Old peak</i>	0.3
11	<i>ST_Slope</i>	1
12	<i>HeartDisease</i>	0

Langkah awal adalah mengecek data pasien di *root node*. Didapati *ST_Slope* pasien lebih kecil dari 1.5, maka kita kirim data pasien ke *node* kiri. Kemudian, akan dicek *ChestPainType* pasien. Karena lebih besar dari 2.5, maka data dikirim ke kanan *node*. Kemudian akan dicek *ExerciseAngina* data pasien. Karena *ExerciseAngina* lebih kecil dari 0.5, maka data dikirim ke kiri, Data dikirim ke kiri berarti nilai nol adalah hasil prediksinya. Karena *HeartDisease* pasien adalah nol, maka data pasien diprediksi sesuai yang diinginkan.

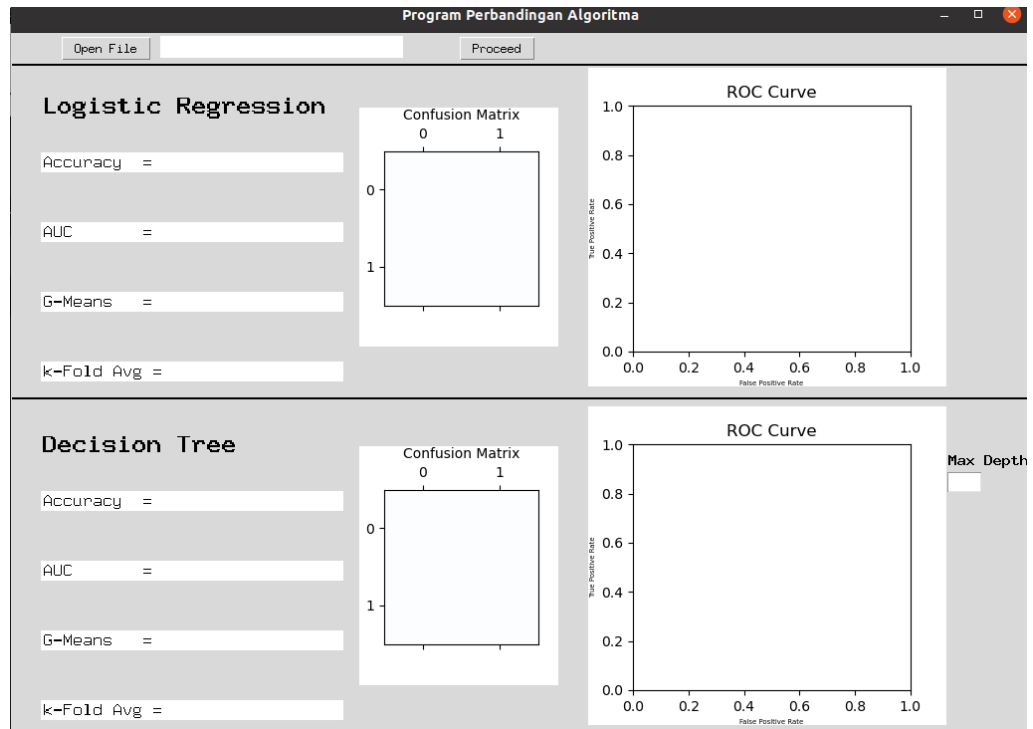
4.2 Implementasi Sistem

Pada bagian ini akan dijelaskan tampilan halaman dari sistem yang dibuat dengan bahasa pemrograman *Python*. Aplikasi tidak menggunakan jumlah halaman yang begitu banyak dalam melakukan prosesnya.

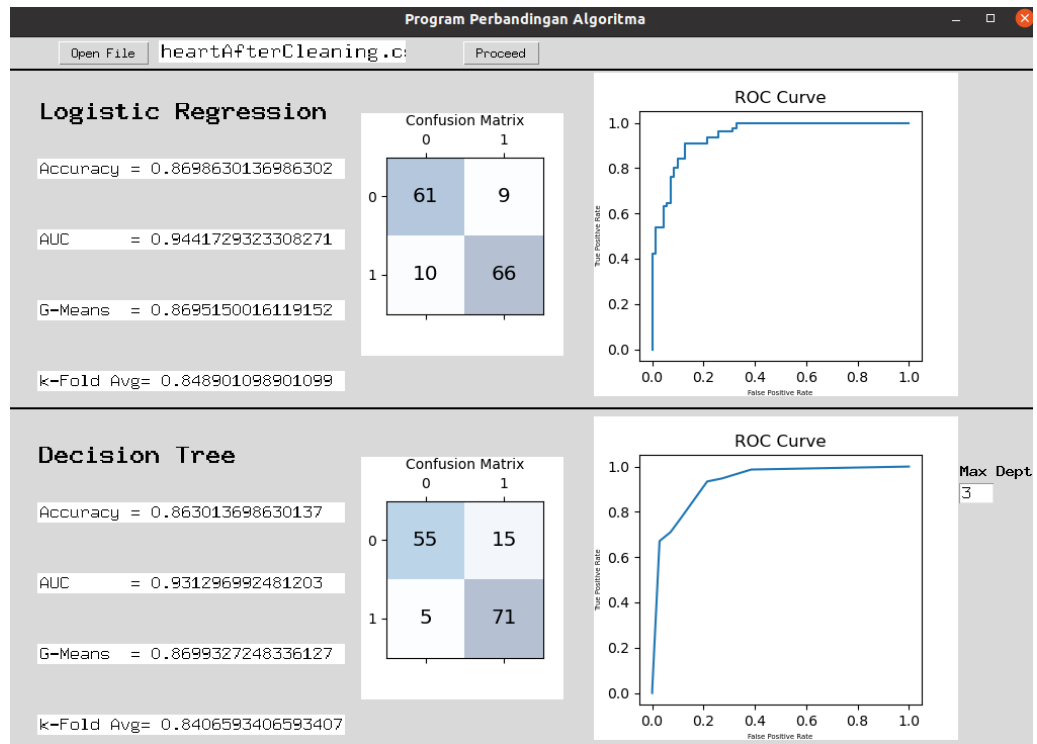
4.2.1 Halaman Utama Sistem

Halaman utama adalah halaman yang pertama kali dilihat oleh pengguna saat membuka aplikasi. Halaman ini akan menampilkan tampilan yang terbagi atas tiga bagian, yaitu bagian atas, bagian *Logistic Regression*, dan bagian

Decision Tree. Di bagian atas, pengguna dapat memilih file yang akan diproses oleh kedua algoritma *Machine Learning* untuk diproses. Di bagian *Logistic Regression*, pengguna dapat melihat hasil dan visualisasi pemrosesan algoritma *Logistic Regression*, dan di bagian *Decision Tree*, pengguna dapat melihat hasil dan visualisasi pemrosesan algoritma *Decision Tree*. Berikut diperlihatkan tampilan program sebelum dataset diproses pada gambar 4.3, dan sesudah diproses pada gambar 4.4.



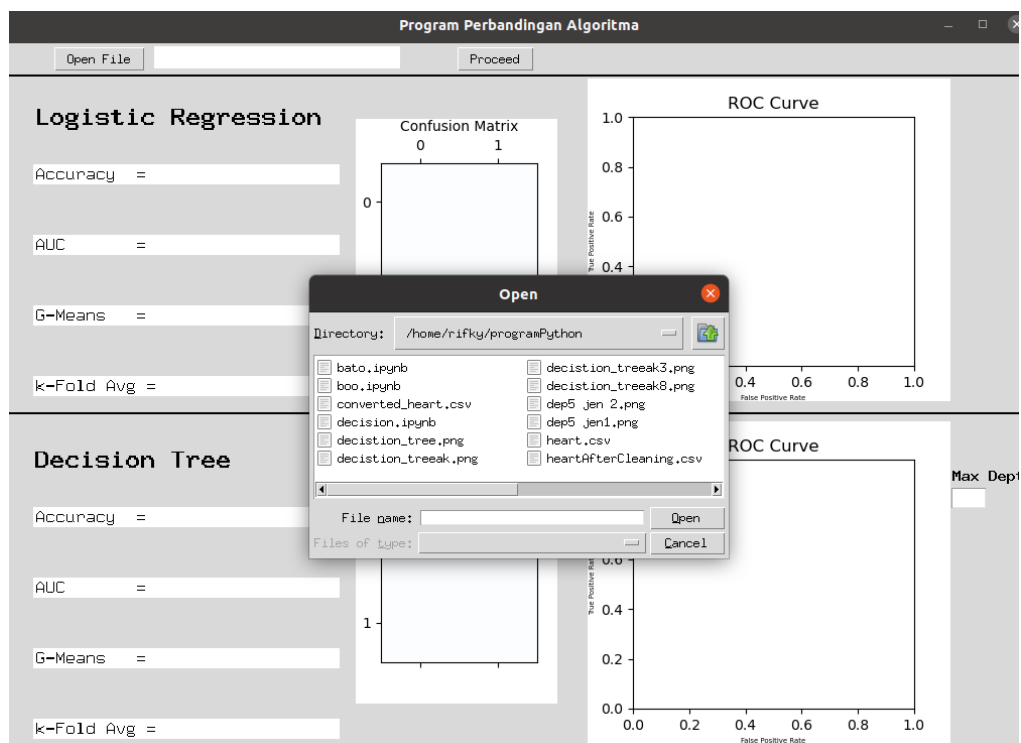
Gambar 4.3 Tampilan Halaman Sistem Sebelum Proses



Gambar 4.4 Tampilan Halaman Sistem Sesudah Proses

4.2.2 Halaman Input Dataset

Halaman input dataset adalah halaman kecil yang muncul ketika pengguna menekan tombol *open file* yang berguna untuk memilih file yang akan diproses oleh aplikasi. Berikut pada gambar 4.5 adalah tampilannya.



Gambar 4.5 Tampilan Halaman Input

4.3 Pengujian Sistem

Tahapan ini adalah proses untuk melakukan pengujian sistem dengan bertujuan mendapatkan hasil yang diinginkan dalam penelitian ini, yang kemudian nantinya akan dibandingkan. Berikut tahapan pengujian sistem.

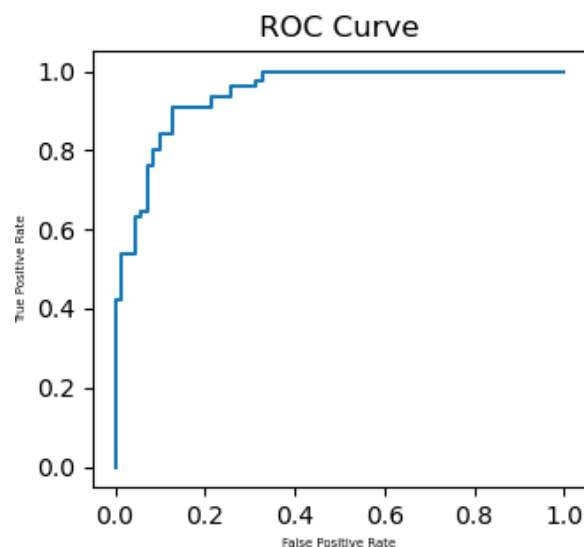
4.3.1 Hasil Pengujian

Berikut ditampilkan hasil pengujian dari kedua algoritma. Berikut tabel 4.11 menampilkan pengujian menggunakan algoritma *Logistic Regression*.

Tabel 4.11 Nilai Pengujian Algoritma *Logistic Regression*

Jenis Output	Nilai
<i>Accuracy</i>	0.8698630136986302
<i>Area Under Curve (AUC)</i>	0.9441729323308271
<i>G-Means</i>	0.8699235123234876
<i>Average k-Fold Cross Validation Score (8 Fold)</i>	0.848901098901099
<i>True Negative</i>	61
<i>False Positive</i>	9
<i>False Negative</i>	10
<i>True Positive</i>	66

Berikut ditampilkan pada gambar 4.6 grafik ROC dari pengujian menggunakan algoritma *Logistic Regression*.



Gambar 4.6 Grafik ROC algoritma *Logistic Regression*

Untuk algoritma *Decision Tree* akan ditentukan terlebih dahulu berapa *depth* maksimal *Decision Tree* yang memiliki hasil terbaik. Hasil terbaik dapat dilihat dari *depth* maksimal yang memiliki nilai *accuracy*, AUC, *G-Means*, dan rata-rata score *k-Fold* tertinggi.

Karena algoritma *Decision Tree* di *Scikit-learn* melibatkan perhitungan angka acak, maka akan dilakukan percobaan 10 kali untuk setiap *depth* maksimal dan dicari nilai rata-rata dari *accuracy*, AUC, *G-Means*, dan nilai rata-rata score *k-Fold* setelah 10 percobaan tersebut.

Berikut pada tabel 4.12 dan tabel 4.13 ditampilkan rata-rata nilainya setelah 10 kali percobaan.

Tabel 4.12 Nilai Rata-Rata *Accuracy* dan AUC Setiap *Depth*

Depth Maksimal	Accuracy	AUC
1	0,8356164383561644	0,8330827067669173
2	0,8356164383561644	0,906015037593985
3	0,8595890410958905	0,9267387218045112
4	0,821917808219178	0,9105263157894738
5	0,8404109589041097	0,9008646616541355
6	0,8554794520547944	0,8736654135338346
7	0,8465753424657535	0,8390695488721805
8	0,843835616438356	0,8281015037593985
9	0,8315068493150685	0,8123966165413533
10	0,8397260273972602	0,8278383458646617
11	0,832876712328767	0,8218609022556391
12	0,813013698630137	0,8081766917293233
13	0,7965753424657533	0,8094360902255639
14	0,8027397260273972	0,8031390977443609
15	0,8082191780821917	0,808515037593985

Tabel 4.13 Nilai Rata-Rata *G-Means* dan Rata-Rata *k-Fold* Setiap *Depth*

Depth Maksimal	G-Means	Rata-rata k-Fold
1	0,8307981487159049	0,8241758241758241
2	0,8307981487159049	0,8159340659340659
3	0,853723736429507	0,8413461538461538

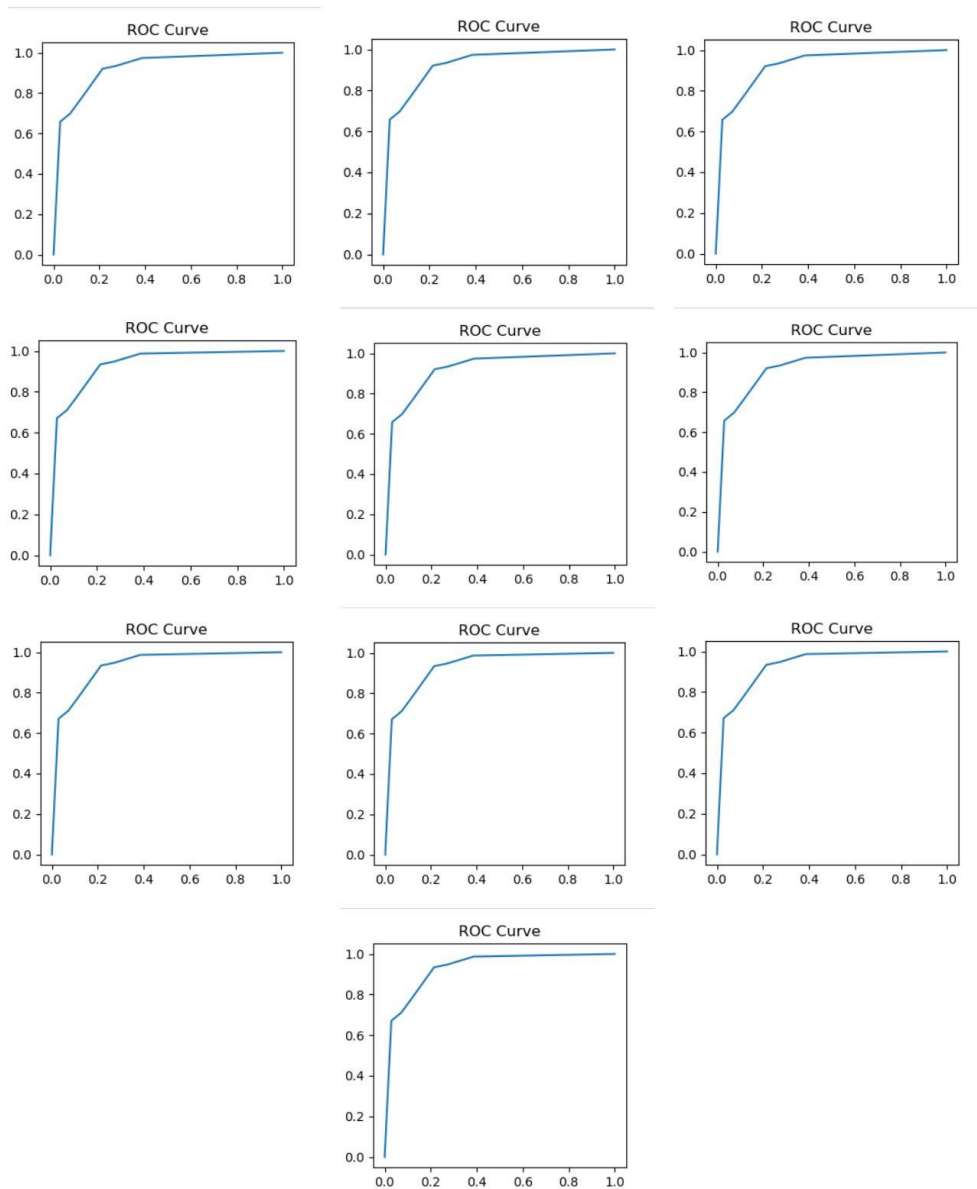
4	0,8207826816681234	0,8384615384615383
5	0,838695132150405	0,8254120879120878
6	0,8528119035426382	0,819368131868132
7	0,8439165736651393	0,8096153846153845
8	0,8419318648113494	0,8089285714285715
9	0,8297645965788876	0,8031593406593407
10	0,8388654462730111	0,7994505494505495
11	0,8311499213015418	0,7946428571428572
12	0,8125385254354057	0,7956043956043956
13	0,7967184912477824	0,7925824175824176
14	0,803015168191581	0,795054945054945
15	0,8081957164766179	0,7978021978021977

Terlihat bahwa dari nilai rata-rata *accuracy*, *AUC*, *G-Means*, dan nilai rata-rata *score k-Fold* setelah 10 percobaan tertinggi saat *depth* maksimal sama dengan 3. Maka, akan digunakan *Decision Tree* dengan *depth* maksimal 3 sebagai model yang digunakan dalam hasil pengujian di penelitian ini. Berikut adalah hasil pengujian menggunakan algoritma *Decision Tree* dengan *depth* maksimal sebesar 3 (hasil pengujian tabel di bawah menggunakan nilai rata-rata dari model *Decision Tree* yang telah didapatkan di percobaan sebelumnya).

Tabel 4.14 Nilai Pengujian Algoritma *Decision Tree*

Jenis Output	Nilai Rata-Rata
<i>Accuracy</i>	0,8595890410958905
<i>Area Under Curve (AUC)</i>	0,9267387218045112
<i>G-Means</i>	0,853723736429507
<i>Average k-Fold Cross Validation Score (8 Fold)</i>	0,8413461538461538
<i>True Negative</i>	55
<i>False Positive</i>	15
<i>False Negative</i>	5.5
<i>True Positive</i>	70.5

Berikut ditampilkan pada gambar 4.7 grafik ROC dari sepuluh percobaan *Decision Tree* dengan *depth* maksimal sebanyak 3 tersebut.



Gambar 4.7 Grafik ROC algoritma *Decision Tree*

4.3.2 Perbandingan Hasil Pengujian

Setelah hasil pengujian telah didapatkan, maka dilanjutkan ke tahap selanjutnya yaitu perbandingan hasil pengujian. Akan digunakan nilai setiap jenis output pada algoritma *Logistic Regression* dan nilai rata-rata setiap jenis output pada algoritma *Decision Tree*. Berikut pada tabel 4.15 ditampilkan hasil pengujian kedua algoritma.

Tabel 4.15 Nilai Pengujian Kedua Algoritma

Jenis Output	Hasil Algoritma <i>Logistic Regression</i>	Hasil Algoritma <i>Decision Tree</i>
<i>Accuracy</i>	0.8698630136986302	0,8595890410958905
<i>Area Under Curve (AUC)</i>	0.9441729323308271	0,9267387218045112
<i>G-Means</i>	0.8699235123234876	0,853723736429507
<i>Average k-Fold Cross Validation Score (8 Fold)</i>	0.848901098901099	0,8413461538461538
<i>True Negative</i>	61	55
<i>False Positive</i>	9	15
<i>False Negative</i>	10	5.5
<i>True Positive</i>	66	70.5

Terlihat bahwa nilai *accuracy* algoritma *Logistic Regression* lebih tinggi daripada nilai *accuracy* algoritma *Decision Tree*. Sehingga, menurut nilai *accuracy* performa algoritma *Logistic Regression* lebih baik daripada algoritma *Decision Tree* dalam mendiagnosis penyakit jantung.

Berdasarkan nilai AUC, algoritma *Logistic Regression* dan *Decision Tree* sama-sama memiliki nilai AUC antara 0.90-1.00. Oleh karena itu, kedua algoritma dalam mendiagnosis penyakit jantung menurut nilai AUC tergolong *Excellent classification*. Namun, algoritma *Logistic Regression* memiliki nilai AUC lebih tinggi daripada *Decision Tree*, sehingga menurut nilai AUC, algoritma *Logistic Regression* lebih baik performanya daripada algoritma *Decision Tree*. Nilai AUC algoritma *Logistic Regression* yang lebih tinggi juga menandakan bahwa algoritma *Logistic Regression* dapat mengatasi *class imbalance* lebih baik daripada algoritma *Decision Tree*. Nilai AUC algoritma *Logistic Regression* yang lebih tinggi juga pula menandakan bahwa algoritma *Logistic Regression* memiliki luas grafik ROC lebih besar daripada algoritma *Decision Tree*. Grafik ROC algoritma *Logistic Regression* dapat dilihat pada gambar 4.6 , sedangkan grafik ROC algoritma *Decision Tree* dapat dilihat pada gambar 4.7.

Seperti nilai AUC, nilai *G-Means* menandakan seberapa bagus suatu algoritma dalam mengatasi *class imbalance*. Berdasarkan tabel 4.15 , terlihat bahwa algoritma *Logistic Regression* memiliki nilai *G-Means* lebih tinggi

daripada algoritma *Decision Tree*. Sehingga menurut nilai *G-Means*, algoritma *Logistic Regression* dapat mengatasi *class imbalance* lebih baik daripada algoritma *Decision Tree*.

Nilai *average k-Fold Cross Validation* terlihat lebih tinggi pada algoritma *Logistic Regression* daripada algoritma *Decision Tree*. Sehingga, algoritma *Logistic Regression* lebih tidak rawan untuk terkena masalah *overfitting* dan *underfitting* daripada algoritma *Decision Tree*.

Nilai *True Negative* algoritma *Logistic Regression* lebih tinggi daripada algoritma *Decision Tree*. Ini menandakan bahwa saat tidak ada penyakit jantung pada pasien, algoritma *Logistic Regression* lebih bagus dalam mendeteksinya dibandingkan algoritma *Decision Tree*.

Nilai *False Positive* algoritma *Decision Tree* lebih tinggi daripada algoritma *Logistic Regression*. Ini menandakan bahwa saat tidak ada penyakit jantung pada pasien, algoritma *Decision Tree* lebih banyak salah dalam mendeteksinya dibandingkan algoritma *Logistic Regression*.

Nilai *False Negative* algoritma *Logistic Regression* lebih tinggi daripada algoritma *Decision Tree*. Ini menandakan bahwa saat ada penyakit jantung pada pasien, algoritma *Logistic Regression* lebih banyak salah dalam mendeteksinya dibandingkan algoritma *Decision Tree*.

Nilai *True Positive* algoritma *Decision Tree* lebih tinggi daripada algoritma *Logistic Regression*. Ini menandakan bahwa saat ada penyakit jantung pada pasien, algoritma *Decision Tree* lebih bagus dalam mendeteksinya dibandingkan algoritma *Logistic Regression*.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Mendasari hasil pengujian dan perbandingan yang telah dipaparkan di bab sebelumnya, maka dapat diambil beberapa kesimpulan atas penelitian ini.

1. Hasil nilai *accuracy* algoritma *Logistic Regression* yang sebesar 0.8698 lebih tinggi dari hasil nilai *accuracy* algoritma *Decision Tree* yang sebesar 0,8595. Sehingga, menurut nilai *accuracy* algoritma *Logistic Regression* lebih efektif daripada algoritma *Decision Tree* dalam mendiagnosis penyakit jantung.
2. Hasil nilai AUC algoritma *Logistic Regression* yang sebesar 0.9441 lebih tinggi dari hasil nilai AUC algoritma *Decision Tree* yang sebesar 0,9267. Sehingga, menurut nilai AUC, algoritma *Logistic Regression* lebih efektif dari algoritma *Decision Tree* dalam mendiagnosis penyakit jantung. Nilai AUC algoritma *Logistic Regression* yang lebih tinggi juga menandakan bahwa algoritma *Logistic Regression* dapat mengatasi *class imbalance* lebih baik daripada algoritma *Decision Tree*.
3. Hasil nilai *G-Means* algoritma *Logistic Regression* yang sebesar 0.8699 lebih tinggi dari hasil nilai *G-Means* algoritma *Decision Tree* yang sebesar 0,8537. Sehingga, menurut nilai *G-Means*, algoritma *Logistic Regression* dapat mengatasi *class imbalance* lebih baik daripada algoritma *Decision Tree* dalam melakukan diagnosis penyakit jantung.
4. Hasil nilai *average k-Fold Cross Validation* algoritma *Logistic Regression* yang sebesar 0.8489 lebih tinggi dari hasil nilai *average k-Fold Cross Validation* algoritma *Decision Tree* yang sebesar 0,8413. Sehingga, algoritma *Logistic Regression* lebih tidak rawan untuk terkena masalah *overfitting* dan *underfitting* daripada algoritma *Decision Tree* dalam melakukan diagnosis penyakit jantung.

5.2 Saran

Berikut saran membangun yang dapat dijadikan masukan untuk pengembangan selanjutnya.

1. Menambah variasi hasil pengujian yang dapat digunakan untuk memperdalam detail perbandingan algoritma *Machine Learning*.
2. Membuat *interface* yang lebih baik sehingga pengguna sistem lebih nyaman dalam menggunakan sistem.
3. Mencoba meneliti menggunakan algoritma *Machine Learning* yang lain sehingga dapat lebih memperluas wawasan dalam hal diagnosis penyakit jantung yang dilakukan *Machine Learning*.

DAFTAR PUSTAKA

- Abdillah, AA., Suwarno. 2016. Penerapan Algoritma Bayesian Regularization Backpropagation Untuk Memprediksi Penyakit Diabetes. *Jurnal MIPA*. **39** (2): 154. (Online) <https://www.neliti.com/publications/113706/penerapan-algoritma-bayesian-regularization-backpropagation-untuk-memprediksi-pe>(03 Oktober 2023)
- Abdussomad. 2017. Penerapan Algoritma Klasifikasi C4.5 Untuk Menghasilkan Pola Kelayakan Kredit. *Repositori BSI*. **2** (1): 22. (Online) <https://repository.bsi.ac.id/repo/files/238105/download/33-Article-Text-45-2-10-20180823.pdf> (24 November 2023)
- Afifah, D. N. 2020. Penerapan Metode Regresi Logistik Biner Pada Kesejahteraan Rumah Tangga Di Kabupaten Mojokerto. Skripsi. Universitas Islam Negeri Maulana Malik Ibrahim
- Ambarwari, A., Adrian, Q. J. & Herdiyeni, Y. 2021. Analisis Pengaruh Data Scaling Terhadap Performa Algoritme Machine Learning untuk Identifikasi Tanaman. *Jurnal RESTI*. **4** (1): 118-119. (Online) <https://jurnal.iaii.or.id/index.php/RESTI/article/view/1517> (26 November 2023)
- Alaoui, Safae S., Aksasse, B. & Farhaoui, Y. 2018. Classification algorithms in Data Mining. *ResearchGate*. **31** (4): 3-4. (Online) https://www.researchgate.net/publication/326866871_Classification_algorithms_in_Data_Mining (14 September 2022)
- Chary, P. D. & Singh, R. P. 2020. Review on Advanced Machine Learning Model: Scikit-Learn. *IJSRED*. **3** (4): 526. (Online) <http://www.ijsred.com/volume3/issue4/IJSRED-V3I4P66.pdf> (30 November 2023)
- Fedesoriano. (September 2021). Heart Failure Prediction Dataset. Retrieved [Date Retrieved] from <https://www.kaggle.com/fedesoriano/heart-failure-prediction>.

- Firmansyach, W. A., Hayati, U. & Wijaya, Y. A. 2023. Analisa Terjadinya Overfitting Dan Underfitting Pada Algoritma Naive Bayes Dan Decision Tree Dengan Teknik Cross Validation. *Jurnal Mahasiswa Teknik Informatika*. **7** (1): 262-264. (Online) <https://ejournal.itn.ac.id/index.php/jati/article/view/6329> (25 November 2023)
- Garcia, V., Mollineda, R. A. & Sanchez, J.S. 2009. Index of Balanced Accuracy: A Performance Measure for Skewed Class Distributions. *Core*. **1** (1): 2-3. (Online) <https://core.ac.uk/download/61392839.pdf> (14 November 2023)
- Gedela, M., Khan, M. & Jonsson, O. 2015. Heart Failure. *ResearchGate*. **68** (9): 405. (Online) https://www.researchgate.net/publication/283899687_Heart_Failure (14 September 2022)
- Hendayana, R. 2012. Penerapan Metode Regresi Logistik Dalam Menganalisis Adopsi Teknologi Pertanian. *Media Neliti*. **22** (1): 2. (Online) <https://www.neliti.com/publications/31101/penerapan-metode-regresi-logistik-dalam-menganalisis-adopsi-teknologi-pertanian> (07 Maret 2023)
- Hutomo, Alam W. 2020. Perbandingan Kinerja AUC Dan G-means Pada Machine Learning Berbasis Seleksi Fitur Algoritma Genetika Untuk Prediksi Cacat Perangkat Lunak. *Repository UINJKT*. 23-25. (Online) <https://repository.uinjkt.ac.id/dspace/bitstream/123456789/56253/1/ALAM%20WAHYU%20HUTOMO-FST.pdf> (12 April 2023)
- Jaiswal, A. & Dwivedi, A. 2021. Python: The Versatile Language. *Recent Trends in Programming language*. **8** (1): 24-27. (Online) <https://computers.stmjournals.com/index.php?journal=RTPL&page=article&op=view&path%5B%5D=2758> (30 November 2023)
- Kasih, P. 2019. *Pemodelan Data Mining Decision Tree Dengan Classification Error Untuk Seleksi Calon Anggota Tim Paduan Suara*. *Innovatics*. **1** (2): 64-65. (Online) <https://jurnal.unsil.ac.id/index.php/innovatics/article/view/918> (07 Maret 2023)

- Lawrynowicz, A. & Tresp, V. 2014. Introducing Machine Learning. *Perspectives on Ontology Learning*. **1** (1): 36. (Online) https://www.researchgate.net/publication/268804320_Introducing_Machine_Learning (17 Agustus 2023)
- Lopez, E. O. and Jan, A. 2019. Cardiovascular Disease. *StatPearls*. **1** (1): 5. (Online) https://www.researchgate.net/publication/337060450_Cardiovascular_Disease (23 Agustus 2023)
- Muchai, E. & Odongo, L. 2014. Comparison Of Crisp And Fuzzy Classification Trees Using Gini Index Impurity Measure On Simulated Data. *European Scientific Journal*. **10** (18): 2. (Online) <https://core.ac.uk/download/pdf/236414605.pdf> (24 November 2023)
- Naryadi, N. W. 2019. Hubungan Tingkat Pengetahuan, Tingkat Dukungan Keluarga dan Tingkat Kepatuhan Diet Pasien Jantung Pasca Rawat Inap di Rumah Sakit Umum Bangli. *Repository Poltekkes*. **1** (1): 6. (Online) <http://repository.poltekkes-denpasar.ac.id/3142/> (07 October 2022)
- Pratiwi, A. & Nursita., H. 2020. Peningkatan Kualitas Hidup Pada Pasien Gagal Jantung: a Narrative Review Article. *Jurnal Berita Ilmu Keperawatan*. **13** (1): 11. (Online) <https://www.neliti.com/publications/337440/peningkatan-kualitas-hidup-pada-pasien-gagal-jantung-a-narrative-review-article> (14 September 2022)
- Ranjan, M. K., Barot, K., Khairnar, V., Rawal, V., Pimpalgaonkar, A., Saxena, S. & Sattar, A. M. 2023. Python: Empowering Data Science Applications and Research. *Journal of Operating Systems Development & Trends*. **10** (1): 30-31 (Online) (30 November 2023)
- Rasyid, A. 2014. Sistem Prediksi Prestasi Akademik Mahasiswa Menggunakan Metode Decision Tree C4.5. *UMG Repository*. **1** (1): 11. (Online) <http://eprints.umg.ac.id/1533/> (29 September 2023)
- Roihan, A., Sunarya, P. A. & Rafika, A. S. 2020. Pemanfaatan Machine Learning dalam Berbagai Bidang: Review Paper. *IJCT*. **5** (1): 76-78. (Online) <https://ejournal.bsi.ac.id/ejurnal/index.php/ijcit/article/view/7951/0> (14 September 2022)

- Tampil, Y. A., Komalig, H. & Langi, Y. 2017. Analisis Regresi Logistik Untuk Menentukan Faktor-Faktor Yang Mempengaruhi Indeks Prestasi Kumulatif (IPK) Mahasiswa FMIPA Universitas Sam Ratulangi Manado. *Jurnal Matematika dan Aplikasi*. **6**(2): 57.(Online) <https://ejournal.unsrat.ac.id/index.php/decartesian/article/view/17023> (07 Maret 2023)
- Tampubolon, L. F., Ginting, A., & Turnip, F. E. S., 2023. Gambaran Faktor yang Mempengaruhi Kejadian Penyakit Jantung Koroner (PJK) di Pusat Jantung Terpadu (PJT). *Jurnal Ilmiah Permas: Jurnal Ilmiah STIKES Kendal*. **13** (3): 1044. (Online) <https://journal2.stikeskendal.ac.id/index.php/PSKM/article/download/1077/790> (28 Desember 2023)
- Yang, S. & Berdine, G. 2017. The receiver operating characteristic (ROC) curve. *ResearchGate*. **5** (19): 34 (Online) https://www.researchgate.net/publication/316751328_The_receiver_operating_characteristic_ROC_curve (03 Oktober 2023)

LISTING PROGRAM

```

import tkinter as tk
import pandas as pd
import math
import os
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from tkinter import filedialog as fd
from sklearn.model_selection import KFold, cross_val_score
import numpy as np
from sklearn import tree
from sklearn.tree import export_text
from sklearn.metrics import precision_recall_fscore_support

```

```

class Logistic_Regression_Algorithm:

```

```

    def __init__(self, xtrain, xtest, ytrain, ytest):
        self.xtrain = xtrain
        self.xtest = xtest
        self.ytrain = ytrain
        self.ytest = ytest

```

```

    def calculate(self):
        self.scaling()

        log_res = LogisticRegression(random_state=0)

        log_res.fit(self.xtrain, self.ytrain)

        #accuracy

```

```

        log_res.fit(self.xtrain, self.ytrain)

        #accuracy
        yPredResult = log_res.predict(self.xtest)
        accuracy = accuracy_score(self.ytest, yPredResult)

        #auc
        pred_proba = log_res.predict_proba(self.xtest)
        auc = roc_auc_score(self.ytest, pred_proba[:,1])

        cm = confusion_matrix(self.ytest, yPredResult)

        pre, rec, fs, _ = precision_recall_fscore_support(self.ytest, yPredResult)

        sens = rec[1]
        spec = rec[0]
        gMeans = math.sqrt(sens*spec)

        pred_proba2 = pred_proba[:,1]
        fpr, tpr, _ = metrics.roc_curve(self.ytest, pred_proba2)

        x = np.concatenate([self.xtrain, self.xtest])
        y = np.concatenate([self.ytrain, self.ytest])

        kf = KFold(n_splits=8)
        kFoldScore = cross_val_score(LogisticRegression(random_state=0), x, y, cv=kf)
        kFoldScoreAvg = kFoldScore.mean()

        return accuracy, auc, gMeans, cm, fpr, tpr, kFoldScoreAvg

```

```

    def scaling(self):
        scaler = StandardScaler()
        self.xtrain = scaler.fit_transform(self.xtrain)
        self.xtest = scaler.transform(self.xtest)

```

```

class Decision_Tree_Algorithm:

```

```

    def __init__(self, xtrain, xtest, ytrain, ytest, depth):
        self.xtrain = xtrain

```

```

        self.xtest = scaler.transform(self.xtest)

class Decision_Tree_Algorithm:

    def __init__(self, xtrain, xtest, ytrain, ytest, depth):
        self.xtrain = xtrain
        self.xtest = xtest
        self.ytrain = ytrain
        self.ytest = ytest
        self.depth = depth

    def calculate(self):

        des_tree = DecisionTreeClassifier(max_depth = self.depth)
        des_tree.fit(self.xtrain, self.ytrain)

        x = np.concatenate([self.xtrain, self.xtest])
        y = np.concatenate([self.ytrain, self.ytest])

        #accuracy
        yPredResult = des_tree.predict(self.xtest)
        accuracy = accuracy_score(self.ytest, yPredResult)

        #auc
        pred_proba = des_tree.predict_proba(self.xtest)
        auc = roc_auc_score(self.ytest, pred_proba[:,1])

        cm = confusion_matrix(self.ytest, yPredResult)

        pre, rec, fs, sd= precision_recall_fscore_support(self.ytest, yPredResult)

        sens = rec[1]
        spec = rec[0]
        gMeans = math.sqrt(sens*spec)

        pred_proba2 = pred_proba[:,1]
        fpr, tpr, _ = metrics.roc_curve(self.ytest, pred_proba2)

        kf = KFold(n_splits=8)
        kFoldScore = cross_val_score(DecisionTreeClassifier(max_depth = self.depth), x, y, cv=kf)

```

```

        pred_proba2 = pred_proba[:,1]
        fpr, tpr, _ = metrics.roc_curve(self.ytest, pred_proba2)

        kf = KFold(n_splits=8)
        kFoldScore = cross_val_score(DecisionTreeClassifier(max_depth = self.depth), x, y, cv=kf)
        kFoldScoreAvg = kFoldScore.mean()

        return accuracy, auc, gMeans, cm, fpr, tpr, kFoldScoreAvg

class Visualization:

    def __init__(self, size, frame):
        self.size = size
        self.frame = frame
        self.fig, self.ax = plt.subplots(figsize = size)
        self.canvas = FigureCanvasTkAgg(self.fig, master = self.frame)
        self.canvas.get_tk_widget().pack()

class Confusion_Matrix_Graph(Visualization):

    def __init__(self, size, frame):
        super().__init__(size, frame)

        self.ax.matshow([[0,0],[0,0]], cmap=plt.cm.Blues, alpha=0.3)
        self.ax.set_title('Confusion Matrix', fontsize=10)

    def draw(self, cm):
        self.canvas.get_tk_widget().pack_forget()
        self.ax.clear()

        self.ax.matshow(cm, cmap=plt.cm.Blues, alpha=0.3)
        for i in range(cm.shape[0]):
            for j in range(cm.shape[1]):
                self.ax.text(x=j, y=i, s=cm[i, j], va='center', ha='center', size='x-large')
        self.ax.set_title('Confusion Matrix', fontsize=10)

        self.canvas = FigureCanvasTkAgg(self.fig, master = self.frame)
        self.canvas.get_tk_widget().pack()

```

```

        self.ax.text(x=j, y=i, s=cm[i, j], va='center', ha='center', size='x-large')
self.ax.set_title('Confusion Matrix', fontsize=10)

self.canvas = FigureCanvasTkAgg(self.fig, master = self.frame)
self.canvas.get_tk_widget().pack()

class ROC_Graph(Visualization):

    def __init__(self, size, frame):
        super().__init__(size, frame)

        self.ax.set_xlabel('False Positive Rate', fontsize=5)
        self.ax.set_ylabel('True Positive Rate', fontsize=5)
        self.ax.set_title('ROC Curve', fontsize=12)

    def draw(self, fpr, tpr):
        self.canvas.get_tk_widget().pack_forget()
        self.ax.clear()

        self.ax.plot(fpr, tpr)
        self.ax.set_xlabel('False Positive Rate', fontsize=5)
        self.ax.set_ylabel('True Positive Rate', fontsize=5)
        self.ax.set_title('ROC Curve', fontsize=12)

        self.canvas = FigureCanvasTkAgg(self.fig, master = self.frame)
        self.canvas.get_tk_widget().pack()

class Button():

    def proceedOutput(self):

        dataset = pd.read_csv(file_name.cget("text"))
        input = text_box.get(1.0, "end-1c")

        if input == '':
            depth = None
        else:
            depth = int(input)

        #data encoding

```

```

        if input == '':
            depth = None
        else:
            depth = int(input)

        #data encoding
        dataset = self.dataEncoding(dataset)

        # input
        x = dataset.drop("HeartDisease", axis = 1)

        # output
        y = dataset['HeartDisease']

        #data split
        xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=0)

        self.proceedOutputLogisticRegression(xtrain, xtest, ytrain, ytest)
        self.proceedOutputDecisionTree(xtrain, xtest, ytrain, ytest, depth)

    def proceedOutputLogisticRegression(self, xtrain, xtest, ytrain, ytest):
        #logistic regression section
        logRes = LogisticRegressionAlgorithm(xtrain, xtest, ytrain, ytest)
        accuracy, auc, gMeans, cm, fpr, tpr, kFoldScoreAvg = logRes.calculate()

        logistic_accuracy.config(text = "Accuracy = " + str(accuracy))
        logistic_auc.config(text = "AUC = " + str(auc))
        logistic_gMeans.config(text = "G-Means = " + str(gMeans))
        logistic_kFold.config(text = "k-Fold Avg= " + str(kFoldScoreAvg))

        logRes_Confusion_Matrix_Graph.draw(cm)
        logRes_ROC_Graph.draw(fpr, tpr)

    def proceedOutputDecisionTree(self, xtrain, xtest, ytrain, ytest, depth):
        #decision tree section
        desTree = DecisionTreeAlgorithm(xtrain, xtest, ytrain, ytest, depth)
        accuracy, auc, gMeans, cm, fpr, tpr, kFoldScoreAvg = desTree.calculate()

```



```

def proceedOutputDecisionTree(self, xtrain, xtest, ytrain, ytest, depth):
    #decision tree section
    desTree = Decision_Tree_Algorithm(xtrain, xtest, ytrain, ytest, depth)
    accuracy, auc, gMeans, cm, fpr, tpr, kFoldScoreAvg = desTree.calculate()

    decision_accuracy.config(text = "Accuracy = " + str(accuracy) )
    decision_auc.config(text = "AUC = " + str(auc))
    decision_gMeans.config(text = "G-Means = " + str(gMeans))
    decision_kFold.config(text = "k-Fold Avg= " + str(kFoldScoreAvg))

    desTree_Confusion_Matrix_Graph.draw(cm)
    desTree_ROC_Graph.draw(fpr, tpr)

def openFile(self):
    filepath = fd.askopenfile()
    dataset_name = os.path.basename(filepath.name)
    file_name.config(text = dataset_name)

def dataEncoding(self, dataset):
    if 'Sex' in dataset.columns:
        dataset['Sex'] = dataset['Sex'].map({'F': 0, 'M': 1})

    if 'ChestPainType' in dataset.columns:
        dataset['ChestPainType'] = dataset['ChestPainType'].map({'TA': 0, 'ATA': 1, 'NAP': 2, 'ASY': 3})

    if 'RestingECG' in dataset.columns:
        dataset['RestingECG'] = dataset['RestingECG'].map({'Normal': 0, 'ST': 1, 'LVH': 2})

    if 'ExerciseAngina' in dataset.columns:
        dataset['ExerciseAngina'] = dataset['ExerciseAngina'].map({'N': 0, 'Y': 1})

    if 'ST_Slope' in dataset.columns:
        dataset['ST_Slope'] = dataset['ST_Slope'].map({'Up': 1, 'Flat': 2, 'Down': 3})

    return dataset

#visual design
root = tk.Tk()

```

```

        dataset['ST_Slope'] = dataset['ST_Slope'].map({'Up': 1, 'Flat': 2, 'Down': 3})

    return dataset

#visual design
root = tk.Tk()
root.geometry("1024x768")
root.title("Program Perbandingan Algoritma")

ButtonFunction = Button()

canvas_line = tk.Canvas(width=10000)
canvas_line.pack()
canvas_line.create_line(0,32,1024,32, fill="black", width=2)

proceed_button = tk.Button(root, text="Proceed", command = ButtonFunction.proceedOutput)
proceed_button.place(x=450, y=3)

file_name = tk.Label(root, text="          ", font=('Arial, 14'), bg = "White", width=20, height=1, anchor="w" )
file_name.place(x=150, y=3)

open_button = tk.Button(root, text="Open File", command = ButtonFunction.openFile, height=1)
open_button.place(x=50, y=3)

canvas_line2 = tk.Canvas(width=10000)
canvas_line2.pack()
canvas_line2.create_line(0,100,1024,100, fill="black", width=2)

#logistic regression section
logistic = tk.Label(root, text="Logistic Regression", font=('Arial 18 bold'))
logistic.place(x=30, y=60)
logistic_accuracy = tk.Label(root, text="Accuracy =          ", font=('Arial, 12'), bg = "White", width=30, anchor="w" )
logistic_accuracy.place(x=30, y=120)
logistic_auc = tk.Label(root, text="AUC =          ", font=('Arial, 12'), bg = "White", width=30, anchor="w" )
logistic_auc.place(x=30, y=180)
logistic_gMeans = tk.Label(root, text="G-Means =          ", font=('Arial, 12'), bg = "White", width=30, anchor="w" )
logistic_gMeans.place(x=30, y=260)
logistic_kFold = tk.Label(root, text="k-Fold Avg =          ", font=('Arial, 12'), bg = "White", width=30, anchor="w" )
logistic_kFold.place(x=30, y=330)

```

```

decision_gMeans.place(x=30, y=600)
decision_kFold = tk.Label(root, text="k-Fold Avg =          ", font=('Arial, 12'), bg = "White", width=30, anchor="w" )
decision_kFold.place(x=30, y=670)

dep = tk.Label(root, text="Max Depth", font=('Arial 11 bold'))
dep.place(x=940, y=420)
text_box = tk.Text(root, height=1, width=3, font=('Arial, 12'))
text_box.place(x=940, y=440)

frame3 = tk.Frame(root)
frame3.place(x=580, y=375)
desTree_ROC_Graph = ROC_Graph((3.6, 3.2), frame3)

frame4 = tk.Frame(root)
frame4.place(x=350, y=415)
desTree_Confusion_Matrix_Graph = Confusion_Matrix_Graph((2, 2.4), frame4)

root.mainloop()

```

OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER

rifky-HP-EliteBook-2560p:~/programPython\$

Ln 284, Col 1