DETEKSI PENYAKIT DIABETES MELALUI PLATFORM WEB MENGGUNAKAN ALGORITMA RANDOM FOREST

SKRIPSI

Diajukan untuk Memenuhi Sebagian Syarat Guna Memperoleh
Gelar Sarjana Strata Satu (S-1)
dalam Ilmu Teknologi Informasi

Oleh: **NUR UFAIRAH**

NIM: 2108096014

PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI WALISONGO SEMARANG

2025

PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini:

Nama : Nur Ufairah

NIM : 2108096014

Jurusan : Teknologi Informasi

Menyatakan bahwa skripsi yang berjudul:

DETEKSI PENYAKIT DIABETES MELALUI PLATFORM WEB MENGGUNAKAN ALGORITMA RANDOM FOREST

Secara keseluruhan adalah hasil penelitian/karya saya sendiri, kecuali bagian tertentu yang dirujuk sumbernya.

Semarang, 3 Juli 2025

Pembuat Pernyataan

Nur Ufairah

NIM. 2108096014

KEMENTERIAN AGAMA REPUBLIK INDONESIA
UNIVERSITAS ISLAM NEGERI WALISONGO SEMARANG
FAKULTAS SAINS DAN TEKNOLOGI
Jalan Prof Dr. Hamka Kampus III Ngaliyan Semarang 50185
Telp. 024-7601295 Fax.7615387

PENGESAHAN

Naskah skripsi berikut ini:

Judul	: Deteksi Penyakit Diabetes Melalui Platform Web Menggunakan Algoritma Random Forest
Penulis	: **Nur Ufairah**
NIM	: 2108096014
Jurusan	: Teknologi Informasi

Telah diujikan dalam sidang *tugas akhir* oleh Dewan Penguji Fakultas Sains dan Teknologi UIN Walisongo dan dapat diterima sebagai salah satu syarat memperoleh gelar sarjana dalam Ilmu Teknologi Informasi.

Semarang, 3 Juli 2025

DEWAN PENGUJI

Penguji I,

Hery Mustofa, M.Kom
NIP. 198703172019031007

Penguji II,

Dr. Masy Ari Ulinuha, M.T
NIP. 198108122011011007

Penguji III,

Mokhamad Iklil Mustofa, M.Kom
NIP. 198808072019031010

Penguji IV,

Nur Cahyo Hendro Wibowo, S.T., M.Kom
NIP. 197312222006041001

Pembimbing I,

Dr. Masy Ari Ulinuha, M.T
NIP. 198108122011011007

Pembimbing II,

Dr. Wenty Dwi Yuniarti, S.Pd., M.Kom
NIP. 197706222006042005

NOTA PEMBIMBING

Semarang, 8 Mei 2025

Yth. Ketua Program Studi Teknologi Informasi
Fakultas Sains dan Teknologi
UIN Walisongo Semarang

Assalamu'alaikum Wr. Wb.

Dengan ini diberitahukan bahwa saya telah melakukan bimbingan, arahan dan koreksi naskah skripsi dengan:

Judul	: Klasifikasi Penyakit Diabetes Melalui Platform
Web	Menggunakan Algoritma Random Forest
Nama	: **Nur Ufairah**
NIM	: 2108096014
Jurusan	: Teknologi Informasi

Saya memandang bahwa naskah skripsi tersebut sudah dapat diajukan kepada Fakultas Sains dan Teknologi UIN Walisongo untuk diajukan dalam Sidang Munaqasyah.

Wassalamu'alaikum Wr. Wb.

Pembimbing I,

Dr. Masy Ari Ulinuha S.T., M.T.

NIP: 198108122011011007

NOTA PEMBIMBING

Semarang, 13

Yth. Ketua Program Studi Teknologi Informasi
Fakultas Sains dan Teknologi
UIN Walisongo Semarang

Assalamu'alaikum Wr. Wb.

Dengan ini diberitahukan bahwa saya telah melakukan bimbingan, arahan dan koreksi naskah skripsi dengan:

Judul	: Klasifikasi Penyakit Diabetes Melalui Platform
Web	Menggunakan Algoritma Random Forest
Nama	: **Nur Ufairah**
NIM	: 2108096014
Jurusan	: Teknologi Informasi

Saya memandang bahwa naskah skripsi tersebut sudah dapat diajukan kepada Fakultas Sains dan Teknologi UIN Walisongo untuk diajukan dalam Sidang Munaqasyah.

Wassalamu'alaikum Wr. Wb.

Pembimbing II,

Dr. Wenty Dwi Yuniarti, S.Pd,.M.Kom

NIP. 197706222006042005

LEMBAR PERSEMBAHAN

Dengan rasa syukur yang tak terhingga, karya ini penulis persembahkan kepada:

1. Allah Subhanahu Wata'ala, Tuhan Yang Maha Esa, atas segala nikmat iman, kesehatan, kekuatan, dan limpahan rahmat yang tiada henti. Tanpa kehendak dan kasih-Nya, langkah ini tak akan pernah sampai di titik ini.
2. Mama dan Bapak tercinta, sumber kekuatan dan cinta yang tulus tanpa syarat. Terima kasih atas doa yang tak pernah putus, dukungan yang tak pernah surut, dan cinta yang tak tergantikan. Serta untuk kakak, adik, dan seluruh keluarga yang menjadi pelita dalam perjalanan ini, terima kasih telah menjadi tempat berpulang dalam lelah dan rindu.
3. Bapak dan Ibu dosen pembimbing, yang dengan sabar membimbing, mengarahkan, dan menginspirasi penulis hingga tugas akhir ini dapat terselesaikan. Terima kasih atas ilmu dan keikhlasan yang telah diberikan.
4. Untuk teman-teman KNB yang sudah bukan penghuni KNB lagi hehe, Mbak Rifda, Mbak Febi, Nadiah, Nanda, Etin dan Feli, yang telah menjadi keluarga kedua dalam suka dan duka. Terima kasih untuk kehangatan, dukungan, tawa, dan pelukan yang selalu hadir tepat saat dibutuhkan.

5. Terakhir untuk diri penulis sendiri, yang telah berjuang, bertahan, dan terus melawan rasa malas, lelah, mood, dan ego dalam proses penulisan skripsi ini. Terimakasih untuk tidak menyerah dan selalu tetap bangkit kembali setelah melewati hari yang kadang tidak sesuai ekspektasi, hari-hari burnout, tidak percaya diri, merasa kurang dalam berbagai hal, dan hari-hari lainnya yang penuh nano-nano seperti permen nano-nano yang berjuta rasanya.

MOTTO

“Ketika hidup memberi error 500, tetaplah tenang — mungkin server-nya cuma butuh restart, bukan kamu yang harus menyerah.”

“Saat usaha disertai doa, tak ada yang mustahil.”

ABSTRAK

Diabetes Melitus (DM) merupakan penyakit kronis dengan prevalensi yang terus meningkat. Deteksi dini sangat penting untuk mencegah komplikasi serius. Penelitian ini bertujuan mengembangkan sistem deteksi diabetes berbasis web menggunakan algoritma Random Forest. Dataset berjumlah 791 data pasien dengan 12 atribut medis dari Mendeley Data. Proses penelitian mengikuti metode CRISP-DM yang mencakup pemahaman data, pra-pemrosesan (pembersihan data, encoding, normalisasi, dan penyeimbangan kelas dengan SMOTE), pemodelan, evaluasi, dan penerapan sistem. Model dikembangkan dengan penyetelan hyperparameter menggunakan GridSearchCV. Evaluasi menunjukkan hasil akurasi 97,47%, presisi 97,83%, recall 99,26%, F1-score 98,54%, dan AUC 93,11%. Sistem diimplementasikan ke dalam platform web berbasis Streamlit dan memungkinkan input data manual maupun melalui file CSV. Pengujian dilakukan menggunakan metode *black-box testing* untuk memastikan fungsionalitas sistem berjalan sesuai harapan.

Kata Kunci: Diabetes Melitus, Random Forest, Deteksi, Streamlit

ABSTRACT

Diabetes Mellitus (DM) is a chronic disease with a steadily increasing prevalence. Early detection is essential to prevent serious complications. This study aims to develop a web-based diabetes detection system using the Random Forest algorithm. The dataset consists of 791 patient records with 12 medical attributes, obtained from Mendeley Data. The research follows the CRISP-DM methodology, covering data understanding, preprocessing (including data cleaning, encoding, normalization, and class balancing using SMOTE), modeling, evaluation, and system implementation. The model was optimized using hyperparameter tuning with GridSearchCV. Evaluation results show an accuracy of 97.47%, precision of 97.83%, recall of 99.26%, F1-score of 98.54%, and AUC of 93.11%. The system was deployed on a web platform built with Streamlit, allowing users to input data manually or via CSV file upload. The system was tested using black-box testing to ensure that all functionalities worked as expected.

Keywords: Diabetes Mellitus, Random Forest, Detection, Streamlit

KATA PENGANTAR

Alhamdulillah, puji syukur kehadirat Allah SWT karena atas berkat, rahmat, dan karunia-Nya, penulis dapat menyelesaikan penulisan skripsi ini yang berjudul "Deteksi Penyakit Diabetes Melalui Platform Web Menggunakan Algoritma Random Forest". Segala proses yang dilalui, dari awal hingga akhir, tidak terlepas dari pertolongan Allah serta dukungan dari berbagai pihak.

Tak ada hasil tanpa proses, tak ada karya tanpa luka. Skripsi ini bukan karya yang sempurna, namun merupakan bukti dari perjalanan panjang yang penuh pembelajaran. Penulis menyadari bahwa masih banyak kekurangan dalam penyusunan skripsi ini, baik dari segi isi maupun teknis. Maka dari itu, penulis sangat terbuka menerima kritik dan saran yang membangun agar skripsi ini bisa menjadi lebih baik.

Pada kesempatan ini, izinkan penulis menyampaikan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Prof. Dr. Nizar, M.Ag., Rektor UIN Walisongo beserta segenap jajarannya.
2. Prof. Dr. H. Musahadi, M.Ag., Dekan Fakultas Sains dan Teknologi UIN Walisongo beserta jajarannya.

3. Dr. Khothibul Umam, S.T., M.Kom., Ketua Jurusan Teknologi Informasi UIN Walisongo beserta para dosen jurusan Teknologi Informasi.
4. Dr. Masy Ari Ulinuha, S.T., M.T., selaku dosen pembimbing I dan Dr. Wenty Dwi Yuniarti, S.Pd., M.Kom., selaku dosen pembimbing II yang telah memberi arahan, masukan, dan dukungan kepada penulis selama proses pembuatan skripsi ini.
5. Kedua orang tua tercinta, yaitu bapak Abdussahid, M.Pd., dan ibu Rukaya, S.Ag. Berjuta terimakasih untuk semua pengorbanan yang telah dilakukan dan selalu mengupayakan apapun yang terbaik untuk diri penulis, yang selalu memberikan do'a, dukungan, dan motivasi, sehingga penulis dapat sampai di titik ini.
6. Kakak dan adik penulis, yaitu Apt. Husnul Ma'rifah, S.Farm dan Muhammad Wildan Masyhudul Haq yang selalu memberikan semangat kepada penulis walau hanya dapat bercanda tawa dan bertatap muka melalui panggilan video.
7. Seluruh dosen Teknologi Informasi dan staf Fakultas Sains dan Teknologi yang telah memberi pelajaran, pengetahuan, dan pengalaman yang tak ternilai.
8. Teman-teman jurusan Teknologi Informasi Angkatan 2021 khususnya kelas A. Terima kasih telah membersamai dalam perjalanan ini.

9. Berbagai pihak yang tidak dapat disebutkan satu per satu, yang telah membantu, memberi dukungan, dan sebagainya dalam proses penyusunan skripsi ini penulis ucapkan terimakasih. Semoga Allah limpahkan Rahmat dan berkah-Nya kepada kalian semua.

Ucapan terima kasih yang tulus penulis sampaikan kepada semua pihak yang telah mendukung, membimbing, dan menginspirasi sepanjang proses ini berlangsung. Akhir kata, semoga skripsi ini dapat memberi manfaat bagi penulis dan pembaca, serta menjadi batu pijakan untuk langkah-langkah berikutnya dalam pengabdian dan ilmu pengetahuan.

Semarang, 8 Mei 2025

Penulis

Nur Ufairah

NIM. 2108096014

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

A. Latar Belakang

Diabetes Melitus (DM) merupakan penyakit kronis yang prevalensinya terus meningkat secara global, termasuk di Indonesia. Menurut data dari *International Diabetes Federation* (IDF), jumlah penderita diabetes di seluruh dunia diperkirakan mencapai 463 juta orang pada tahun 2019, dan angka ini diproyeksikan akan terus meningkat dalam beberapa dekade mendatang (Dhani et al., 2024). Sedangkan di Indonesia sendiri jumlah penderita diabetes pada tahun 2021 telah mencapai 19,5 juta jiwa dan Indonesia berada diurutan ke-5 sebagai negara dengan jumlah penderita diabetes terbanyak di dunia. IDF memproyeksikan bahwa jumlah penderita diabetes di Indonesia akan mencapai 28,57 juta jiwa pada tahun 2024. Jumlah ini meningkat sebesar 47% dibandingkan dengan tahun 2021. Angka yang mengkhawatirkan ini menunjukkan betapa pentingnya langkah pencegahan dan penanganan dini untuk menghindari komplikasi serius seperti penyakit jantung, gagal ginjal, dan gangguan penglihatan.

Diabetes melitus adalah gangguan metabolisme yang ditandai dengan hiperglikemia atau tingginya kadar

glukosa dalam darah akibat kekurangan produksi insulin atau resistensi tubuh terhadap insulin (Hardianto, 2021). Kondisi ini menyebabkan gangguan metabolisme karbohidrat, lemak, dan protein. Deteksi dini menjadi kunci dalam mencegah komplikasi yang lebih serius dan meningkatkan kualitas hidup penderita diabetes. Sayangnya, metode konvensional dalam diagnosis sering kali memakan waktu dan membutuhkan sumber daya yang besar.

Kemajuan teknologi informasi dan penerapan *machine learning* dalam dunia medis membuka peluang baru dalam deteksi dini dan prediksi penyakit. Algoritma random forest merupakan salah satu metode *machine learning* yang terkenal efektif dalam klasifikasi data medis. Algoritma ini bekerja dengan membangun sejumlah pohon keputusan dan menggabungkan hasilnya untuk memberikan prediksi yang lebih akurat, serta memiliki keunggulan dalam menangani data berukuran besar dan kompleks (Finda & Utomo, 2024). Penelitian yang dilakukan oleh Kumar dan Singh (2022) menunjukkan bahwa algoritma *random forest* mampu mengungguli algoritma lain seperti *support vector machine* (SVM) dan *decision tree* dalam klasifikasi diabetes. Keunggulan *random forest* terletak pada kemampuannya untuk

mengatasi masalah *overfitting* serta mempertahankan performa yang stabil (Sari et al., 2023).

Penggunaan platform berbasis web untuk mendukung implementasi *machine learning* dalam dunia kesehatan telah terbukti bermanfaat. Teknologi ini menyediakan akses yang cepat dan mudah untuk menganalisis data secara *real-time*, sehingga tenaga medis dan masyarakat dapat menggunakannya untuk mendeteksi penyakit sejak dini. Menurut (Mirafuddin & Supriyatna, 2024), penerapan machine learning dalam aplikasi kesehatan dapat mengenali pola dalam data medis, membantu dokter dan pasien dalam memprediksi penyakit dengan lebih cepat dan akurat. Selain itu kecerdasan buatan juga berperan dalam meningkatkan efisiensi layanan kesehatan, mulai dari identifikasi penyakit hingga perencanaan perawatan yang lebih tepat dan berbasis data.

Agama islam sendiri telah menekankan pentingnya menggunakan ilmu dan teknologi untuk kebaikan umat manusia. Hal ini sebagaimana dengan firman Allah dalam Surat Al-Jumu'ah ayat 10:

فَإِذَا قُضِيَتِ ٱلصَّلَوٰةُ فَٱنتَشِرُوا۟ فِى ٱلْأَرْضِ وَٱبْتَغُوا۟ مِن فَضْلِ ٱللَّهِ وَٱذْكُرُوا۟ ٱللَّهَ كَثِيرًا لَّعَلَّكُمْ تُفْلِحُونَ

Artinya: *"Apabila telah ditunaikan shalat, maka bertebaranlah kamu di muka bumi; dan carilah karunia Allah dan ingatlah Allah banyak-banyak supaya kamu beruntung."*

Ayat ini menegaskan pentingnya usaha manusia dalam mencari pengetahuan dan inovasi untuk kemaslahatan bersama, termasuk dalam hal menjaga kesehatan dan mencegah penyakit.

Penelitian oleh (Sodikin, 2023) menyebutkan bahwa penerapan *machine learning* dapat meningkatkan akurasi dalam proses diagnosis penyakit. Evaluasi model menggunakan metrik kinerja seperti akurasi, sensitivitas, dan spesifisitas memberikan gambaran yang jelas tentang efektivitas *machine learning* dalam mendiagnosis penyakit. Dengan demikian, mengintegrasikan algoritma random forest dalam platform web untuk deteksi diabetes dapat menjadi solusi inovatif yang mendukung penanganan kesehatan yang lebih efektif.

Berdasarkan latar belakang yang telah penulis jelaskan di atas, maka fokus penelitian ini adalah "Bagaimana Membuat Deteksi Penyakit Diabetes melalui Platform Web Menggunakan Algoritma Random Forest" yang nantinya dapat memberikan hasil diagnosis apakah pasien tersebut terkena penyakit diabetes atau tidak dengan

menginputkan beberapa inputan pada platform website yang telah dirancang.

B. Rumusan Masalah

Berdasarkan latar belakang di atas, maka rumusan masalah untuk penelitian ini yaitu sebagai berikut:

1. Bagaimana menerapkan algoritma random forest untuk deteksi penyakit diabetes dan menyajikannya pada platform web?
2. Bagaimana performa algoritma random forest untuk deteksi penyakit diabetes tersebut?

C. Tujuan

Tujuan dari penelitian ini berdasarkan rumusan masalah di atas yaitu sebagai berikut:

1. Untuk menerapkan algoritma Random Forest untuk deteksi penyakit diabetes dan menyajikannya pada platform web.
2. Untuk mengetahui performa model klasifikasi penyakit diabetes yang dibangun melalui platform web dalam hal akurasi.

D. Batasan Masalah

Dalam penelitian ini, batasan masalah ditentukan untuk menjaga fokus dan ruang lingkup penelitian agar

tidak terlalu luas. Adapun batasan masalah pada penelitian ini yaitu sebagai berikut:

1. Sumber Data

   Data yang digunakan dalam penelitian ini adalah data sekunder yang diperoleh dari basis data publik *Mendeley Data*. Data ini berisi informasi terkait pasien diabetes yang terdiri dari 12 atribut yaitu jenis kelamin, umur, urea, *Creatinine ratio* (Cr), HbA1c, *Cholesterol* (Chol), *Triglycerides* (TG), *HDL Cholesterol*, LDL, VLDL, *Body Mass Index* (BMI), dan label *class* (kelas) diagnosis yang terdiri dari dua kelas yaitu Y (menunjukkan pasien terkena diabetes) dan N (pasien tidak terkena diabetes) (Rashid, 2020).

2. Jumlah dan Jenis Data

   Penelitian ini menggunakan 791 entri data pasien yang telah melalui proses pembersihan, termasuk penghapusan data kosong dan outlier guna meningkatkan akurasi model. Penelitian ini hanya menggunakan data pasien yang relevan untuk klasifikasi diabetes, sehingga data yang tidak terkait dikecualikan dari analisis.

3. Lingkup Implementasi

   Penelitian ini mengembangkan model klasifikasi menggunakan algoritma *Random Forest*, yang diimplementasikan dalam platform web berbasis

Streamlit. Platform ini dirancang untuk melakukan analisis dan prediksi penyakit diabetes berdasarkan data yang diinputkan oleh pengguna.

4. Evaluasi Model

   Evaluasi model akan dilakukan menggunakan metrik kinerja seperti akurasi, presisi, sensitivitas, F1-score, dan nilai *Area Under Curve-Receiver Operating Characteristic* (AUC-ROC). Penelitian ini tidak membandingkan algoritma lain di luar *Random Forest*.

E. Manfaat

Adapun manfaat yang didapat dari penelitian ini yaitu:

1. Manfaat Teoritis

   Penelitian ini diharapkan dapat menambah wawasan dan pengetahuan dalam bidang *machine learning* dan ilmu komputer khususnya terkait penerapan algoritma Random Forest dalam klasifikasi medis. Hasil penelitian ini juga diharapkan dapat menjadi referensi ilmiah bagi peneliti lain yang ingin mengembangkan atau memperdalam pemahaman mereka tentang penggunaan algoritma pembelajaran mesin untuk diagnosis penyakit.

2. Manfaat Praktis

   Penelitian ini diharapkan dapat memberikan kontribusi nyata dalam membantu tenaga medis untuk

melakukan diagnosis dini diabetes dengan lebih cepat dan akurat melalui platform web yang mudah diakses. Platform ini memungkinkan akses informasi yang lebih efisien, sehingga dapat mempercepat pengambilan keputusan medis dan penanganan pasien. Selain itu, masyarakat umum dapat memanfaatkan teknologi ini untuk memahami risiko kesehatan mereka, meningkatkan kewaspadaan, serta mendorong tindakan pencegahan dini melalui pemantauan yang lebih mudah.

BAB II

LANDASAN PUSTAKA

A. Kajian Teori

2.1. Diabetes Melitus (DM)

Diabetes Melitus (DM) adalah penyakit kronis yang ditandai dengan hiperglikemia yaitu kadar glukosa dalam darah akibat kekurangan produksi insulin atau resistensi tubuh terhadap insulin. Penyakit ini dibagi menjadi beberapa tipe yaitu diabetes melitus tipe 1, diabetes melitus tipe 2, diabetes gestasional, dan diabetes jenis lainnya yang mencakup berbagai macam kondisi langka seperti diabetes yang dipengaruhi oleh faktor genetik tertentu, maupun yang berkaitan dengan penyakit lain atau pengaruh dari pengunaan obat-obatan (Punthakee et al., 2018).

Gejala diabetes dapat bervariasi tergantung pada kadar glukosa darah dan tipe diabetes. Beberapa gejala umum diabetes yang perlu untuk kita waspadai yaitu sering merasa haus *(polydipsia)*, sering buang air kecil *(polyuria)*, kelelahan dan lemah yang tidak biasa *(weakness)*, cepat merasa lapar *(polyphagia)*, penurunan berat badan *(sudden weight loss)*, penglihatan kabur

(visual blurring), proses pemulihan luka lama atau sering mengalami infeksi *(delayed healing)*, dan warna kulit menggelap *(acanthosis nigricans)* (Siallagan & Fitriyani, 2021). Penderita diabetes sering merasa haus secara berlebihan karena tubuh kehilangan banyak cairan melalui urin dan kadar glukosa yang tinggi dalam darah sehingga menyebabkan ginjal bekerja lebih keras untuk mengeluarkannya.

Diabetes dapat menyebabkan berbagai komplikasi serius seperti penyakit jantung, gagal ginjal, neuropati, dan gangguan penglihatan yang dapat mengurangi kualitas hidup penderita. Penderita diabetes memiliki risiko lebih tinggi untuk mengalami penyakit jantung dan stroke, serta kerusakan pada pembuluh darah kecil di ginjal yang dapat menyebabkan gagal ginjal. Selain itu juga dapat menyebabkan kerusakan saraf yang menimbulkan rasa sakit, kesemutan atau kehilangan sensasi terutama di kaki juga merupakan komplikasi umum dari diabetes (Amelia, 2018).

Dalam jurnal (Iskandar et al., 2024a) dijelaskan bahwa faktor penyebab diabetes ditandai oleh beberapa atribut yang masing-masing

dikelompokkan dengan nilai kelas, diantaranya yaitu:

1. Usia, terdiri dari 6 macam kategori yaitu:

Tabel 2.1 *Age* (Usia)

Kategori	Rentang Usia (Tahun)	Deskripsi
0	31-40	Resiko rendah
1	41-50	Resiko menengah rendah
2	51-60	Resiko menengah
3	61-70	Resiko menengah tinggi
4	71-80	Resiko tinggi
5	> 81	Resiko sangat tinggi

2. BMI, terdiri dari 4 macam kategori yaitu:

Tabel 2.2 BMI

Kategori	Rentang BMI (kg/m^2)	Deskripsi
0	< 18.5	Berat badan kurang
1	18.5 ≦ BMI < 25	Berat badan normal
2	25 ≦ BMI < 30	Kelebihan berat badan
3	30 ≦ BMI < 9999	Obesitas

3. Jenis kelamin, terdiri dari 2 macam kategori yaitu:

Tabel 2.3 *Gender* (Jenis Kelamin)

Kategori	Deskripsi
0	*Female* (Perempuan)
1	*Male* (Laki-laki)

4. Label, terdiri dari 2 macam kategori yaitu:

Tabel 2.4 Label

Kategori	Deskripsi
0	Non-diabetes
1	Diabetes

Pada penelitian klasifikasi diabetes ini, data yang digunakan berjumlah 791 dan terdiri dari 12 atribut seperti yang disajikan dalam tabel 2.5. Atribut inilah yang digunakan untuk memprediksi seseorang terkena diabetes atau tidak.

Tabel 2.5 Atribut Dataset

Atribut	Deskripsi	Kode Pengisian
Gender	Jenis kelamin	F = *Female* M = *Male*
Age	Usia pasien	Angka
Urea	Kadar urea dalam darah	mg/dL
Creatinine ratio	Rasio kreatinin	mg/dL
HbA1c	Kadar glukosa	%
Cholesterol	Total kadar kolesterol	mg/dL

Triglycerides	Lemak dalam darah untuk energi	mg/dL
HDL *(High-Density Lipoprotein)*	Kolesterol baik yang melindungi dari penyakit jantung	mg/dL
LDL *(Low-Density Lipoprotein)*	Kolesterol jahat yang meningkatkan risiko penyakit kardiovaskular	mg/dL
VLDL *(Very-Low-Density Lipoprotein)*	Jenis lipoprotein berbahaya lainnya yang membawa trigliserida	mg/dL
BMI	Indeks massa tubuh	kg/m^2
Class	Kelas terkena DM atau tidak	Y = Diabetes N = Non-diabetes

2.2. *Machine Learning* (ML)

Machine Learning (ML) adalah cabang dari kecerdasan buatan *(artificial intelligence)* yang berfokus pada pengembangan algoritma yang memungkinkan komputer untuk belajar dari data tanpa pemrograman eksplisit. Dalam prosesnya, algoritma *machine learning* mempelajari pola-pola dalam data untuk membuat prediksi atau keputusan dengan sedikit intervensi manusia. Berdasarkan metode pembelajarannya *machine learning* dibagi menjadi tiga kategori utama yaitu *supervised learning, unsupervised learning, dan reinforcement learning* (Bintoro et al., 2024). Pada penelitian ini menggunakan *supervised learning* dimana model dilatih menggunakan dataset yang memiliki pasangan *input-output* seperti data kesehatan pasien dan label klasifikasi (diabetes atau non-diabetes).

Penerapan *machine learning* dalam diagnosis penyakit telah menjadi salah satu bidang yang berkembang pesat dalam beberapa tahun terakhir. Salah satu alasan utamanya adalah kemampuan ML untuk mengenali pola-pola kompleks yang sulit dianalisis oleh manusia (Gefy Fitry Wijaya & Dwi Yuniarto, 2024). Dalam konteks medis, data

yang dikumpulkan dari pasien seperti kadar glukosa, tekanan darah, indeks massa tubuh (BMI), dan usia, sering kali memiliki hubungan yang tidak linier dan kompleks. Algoritma ML mampu mempelajari hubungan ini untuk membuat prediksi yang akurat, misalnya untuk mengidentifikasi risiko penyakit tertentu seperti diabetes.

Keunggulan utama dari ML dalam diagnosis medis adalah kemampuannya untuk menangani data dalam jumlah besar dan kompleksitas tinggi (Attaran & Deb, 2018). Misalnya dataset kesehatan pasien sering kali terdiri dari banyak fitur dan beberapa diantaranya mungkin tidak relevan atau memiliki skala yang berbeda. Proses ML dapat menangani tantangan ini dengan teknik seperti normalisasi data, seleksi fitur, dan pengurangan dimensi. Selain itu, model ML yang telah dilatih dapat digunakan secara otomatis untuk melakukan prediksi pada data baru, yang menjadikannya sangat efisien untuk aplikasi di dunia nyata (Gde Agung Brahmana Suryanegara et al., 2021a).

Dalam penelitian ini, algoritma random forest digunakan untuk mengimplementasikan *machine*

learning. Random forest merupakan salah satu algoritma *supervised learning* yang unggul untuk tugas klasifikasi terutama karena kemampuannya dalam menangani dataset yang kompleks dan tidak seimbang. Model ini bekerja dengan membangun beberapa *decision trees* selama proses pelatihan dan menggabungkan hasil prediksi setiap *tree* menggunakan metode voting mayoritas. Pendekatan ini membuat random forest sangat tahan terhadap *overfitting* yang merupakan masalah umum pada algoritma berbasis *decision tree* tunggal (Sidiq Wijaya et al., 2024).

Salah satu penerapan signifikan ML dalam diagnosis penyakit adalah klasifikasi diabetes. Algoritma seperti random forest mampu mempelajari pola hubungan antara parameter kesehatan pasien (misalnya, kadar glukosa dan BMI) dengan hasil diagnosis diabetes (diabetes atau non-diabetes). Model ini kemudian dapat digunakan untuk memberikan prediksi yang akurat pada pasien baru berdasarkan data kesehatan mereka. Hasil prediksi tersebut tidak hanya memberikan klasifikasi misalnya "diabetes" atau "non-diabetes", tetapi juga dapat

memberikan informasi tambahan, seperti probabilitas prediksi, yang membantu dalam pengambilan keputusan medis (Apriliah et al., 2021).

2.3. Algoritma Random Forest (RF)

Algoritma *Random Forest* merupakan salah satu metode dalam machine learning yang digunakan untuk melakukan klasifikasi maupun regresi terhadap data dalam jumlah besar. Metode ini dikembangkan dari algoritma *Classification and Regression Tree* (CART), yang kemudian diperluas dengan penggunaan teknik agregasi *bootstrap* serta pemilihan fitur secara acak pada saat proses pelatihan model. Dengan pendekatan tersebut, *Random Forest* mampu meningkatkan akurasi prediksi serta mengurangi risiko *overfitting* dibandingkan dengan metode *decision tree* tunggal (Nugroho et al., 2019).

Algoritma sendiri merupakan sekumpulan langkah-langkah logis dan sistematis yang dirancang untuk menyelesaikan suatu permasalahan atau mencapai tujuan tertentu. Dalam bidang ilmu komputer dan pemrograman, algoritma memiliki peran penting dalam pemrosesan data, pelaksanaan perhitungan, serta

penyelesaian berbagai tugas komputasional (Andini & Yahfizham, 2024).

Algoritma *Random Forest* bekerja berdasarkan pendekatan *ensemble learning*, yakni metode yang menggabungkan hasil prediksi dari beberapa model pembelajaran secara kolektif untuk menghasilkan keputusan akhir yang lebih akurat dan stabil. Dalam konteks klasifikasi, keputusan akhir diambil berdasarkan mayoritas suara (*majority voting*) dari sejumlah *decision tree* yang dibentuk. Dengan kata lain, kelas yang paling sering diprediksi oleh pohon-pohon tersebut akan menjadi hasil akhir klasifikasi. Algoritma ini secara sederhana dapat dipandang sebagai kumpulan dari banyak *decision tree* yang bekerja secara paralel untuk mengklasifikasikan data. Semakin banyak jumlah pohon yang digunakan, maka semakin kuat dan stabil performa klasifikasi yang dihasilkan (Yoga Religia et al., 2021).

Sama halnya dengan *decision tree*, algoritma *Random Forest* menggunakan konsep *entropy* dan *information gain* dalam proses pembentukan pohon keputusan. *Entropy* merupakan ukuran ketidakpastian atau keragaman data dalam suatu kelompok. Jika seluruh data dalam satu kelompok

memiliki kelas yang seragam, maka nilai *entropy*-nya akan rendah. Sebaliknya, apabila data dalam kelompok tersebut beragam, maka nilai *entropy*-nya akan tinggi. Adapun rumus *entropy* ditunjukkan sebagai berikut:

$$E(S) = \sum_{i=1}^{n} - p_i log_2 p_i \quad (2.1)$$

Dimana:

- S adalah himpunan kasus atau data
- π (pi) adalah proporsi elemen dari kelas ke-*i* dalam dataset
- *n* adalah jumlah kelas dalam dataset

Sementara itu, *Information Gain* (IG) merupakan ukuran pengurangan ketidakpastian atau *entropy* ketika data dibagi berdasarkan atribut tertentu. Atribut yang menghasilkan nilai IG tertinggi akan dipilih untuk membagi data pada node tersebut karena dianggap paling optimal dalam memisahkan kelas. Adapun rumus *Information Gain* ditunjukkan sebagai berikut:

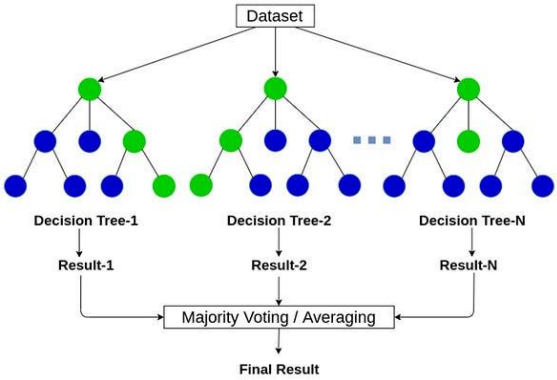
$$Gain(S, A) = Entropy(S) - \sum_{i=1}^{n} (\frac{|S_i|}{|S|} \times Entropy(S_i)) \quad (2.2)$$

Dengan:

- A sebagai atribut yang digunakan untuk membagi data
- S_i adalah subset data hasil pembagian berdasarkan nilai atribut A
- $|S_i|$ adalah jumlah data pada subset S_i
- $|S|$ adalah jumlah total data dalam dataset awal

Dalam proses pelatihan, setiap pohon dalam *Random Forest* dibentuk berdasarkan subset data yang diambil secara acak menggunakan teknik *bagging* (*bootstrap aggregating*). Selain itu, pemilihan fitur pada setiap node juga dilakukan secara acak. Tujuan dari langkah ini adalah untuk meningkatkan keberagaman antar pohon sehingga mengurangi kemungkinan *overfitting*. Setelah seluruh pohon selesai dilatih, masing-masing akan menghasilkan prediksi terhadap data baru. Untuk klasifikasi, hasil prediksi dari seluruh pohon akan digabungkan melalui mekanisme *majority voting*. Sedangkan pada kasus regresi, hasil akhirnya diperoleh dengan menghitung rata-rata dari seluruh prediksi. Gambar 2.1 berikut menggambarkan ilustrasi cara kerja algoritma *Random Forest* secara umum. Proses dimulai dari dataset utama yang dibagi menjadi beberapa

subset acak, kemudian masing-masing digunakan untuk melatih pohon keputusan secara independen. Setiap pohon memberikan hasil prediksi, dan keputusan akhir diperoleh dari hasil penggabungan semua prediksi tersebut.



Gambar 2.1 Ilustrasi Random Forest (RF)

Untuk memberikan pemahaman yang lebih intuitif, proses kerja algoritma *Random Forest* dapat dianalogikan dengan situasi sebagai berikut: Seorang mahasiswa ingin membeli makanan yang enak, namun ia tidak mengetahui pilihan yang terbaik. Ia kemudian bertanya kepada beberapa temannya. Masing-masing teman mengajukan beberapa pertanyaan dan memberikan rekomendasi makanan berdasarkan jawaban yang

diberikan. Proses ini menggambarkan satu *decision tree*. Setelah mendapatkan berbagai rekomendasi dari semua temannya, mahasiswa tersebut memilih makanan yang paling sering direkomendasikan. Proses ini mencerminkan konsep *majority voting* dalam algoritma *Random Forest*, di mana keputusan akhir diambil berdasarkan hasil terbanyak dari sejumlah pohon keputusan (Nugroho et al., 2019).

2.4. Pembagian Data Latih dan Data Uji

Dalam pemodelan *machine learning*, dataset umumnya dibagi menjadi dua bagian utama yaitu data latih *(training set)* dan data uji *(testing set)*. Data latih digunakan untuk membangun dan melatih model, sedangkan data uji digunakan untuk mengevaluasi kemampuan model dalam memprediksi data yang belum pernah dilihat sebelumnya. Tujuan dari pembagian ini adalah mengukur seberapa baik model dapat melakukan generalisasi, yaitu kemampuannya dalam memberikan prediksi yang akurat terhadap data baru, serta untuk mencegah *overfitting* yaitu kondisi di mana model terlalu menyesuaikan diri dengan data pelatihan dan gagal mengenali pola umum (Hakim et al., 2025).

Berdasarkan jurnal dari (Joseph, 2022) dijelaskan bahwa untuk menentukan rasio antara data latih dan data uji dapat dilakukan dengan membagi dataset dalam rasio $\sqrt{p} : 1$, di mana p adalah jumlah parameter. Karena penelitian ini memiliki 12 atribut (parameter), maka berikut adalah langkah perhitungan penentuan rasionya:

1. Hitung nilai $\sqrt{p}$

   $\sqrt{12} = 3{,}46$
2. Rasio

   Rasio pembagian data = 3,46 : 1
3. Total proporsi (jumlah rasio)

   3,46 + 1 = 4,46
4. Hitung presentase
   a. Data latih = $\frac{3{,}46}{4{,}46} \times 100 = 77{,}6\%$
   b. Data uji = $\frac{1}{4{,}46} \times 100 = 22{,}4\%$

Jadi, meskipun rasio terbaik untuk data penelitian ini berdasarkan perhitungan adalah 78:21, peneliti memilih rasio 80:20 untuk memudahkan proses klasifikasi karena rasio ini sudah umum digunakan dan memberikan keseimbangan yang baik antara jumlah data latih yang cukup untuk pelatihan model dan data uji

yang cukup untuk evaluasi, sehingga dapat menghasilkan model yang stabil dan efisien.

2.5. *Preprocessing* Data

Sebelum digunakan dalam pelatihan model machine learning, data perlu melalui tahap pra-pemrosesan guna menjamin kualitasnya. Proses ini mencakup pembersihan data dari nilai-nilai anomali, pemilihan fitur yang memiliki korelasi dengan target, serta transformasi nilai-nilai dalam dataset. Dalam buku *Secondary Analysis of Electronic Health Records*, dijelaskan bahwa tahap preprocessing terdiri dari beberapa langkah utama, yaitu:

a. *Data cleaning*, yaitu proses untuk menangani masalah seperti data yang hilang, data ganda, outlier, dan noise yang dapat mengganggu keakuratan analisis.
b. *Data integration*, yaitu proses menggabungan data dari berbagai sumber menjadi satu dataset yang utuh dan konsisten.
c. *Data transformation*, yaitu mengubah format atau struktur data agar sesuai dengan kebutuhan analisis termasuk normalisasi dan agregasi data.

d. *Data reduction*, yaitu langkah untuk menyederhanakan data yang kompleks dengan cara mengurangi jumlah data namun tetap mempertahankan informasi yang relevan. Proses ini dapat dilakukan melalui pemilihan fitur yang signifikan atau penggunaan metode kompresi data (MIT Critical Data, 2016).

Selain itu, terdapat proses resampling yang sangat penting ketika menghadapi ketidakseimbangan kelas pada dataset. Masalah ketidakseimbangan ini dapat menyebabkan model pembelajaran mesin cenderung memprediksi kelas mayoritas dengan akurasi tinggi, namun gagal mengenali kelas minoritas yang penting. Untuk mengatasi hal ini, teknik *Synthetic Minority Over-sampling Technique* (SMOTE) digunakan dengan cara membuat contoh sintetis baru untuk kelas minoritas melalui interpolasi antara contoh yang ada. Teknik ini bertujuan untuk menyeimbangkan distribusi kelas sehingga model tidak bias terhadap kelas mayoritas dan mampu

menghasilkan prediksi yang lebih seimbang di seluruh kelas (Sir & Soepranoto, 2022).

Selain resampling, proses penting lain dalam membangun model *machine learning* yang optimal adalah *tuning hyperparameter*, yaitu proses untuk mencari nilai parameter terbaik yang mengatur perilaku model saat pelatihan seperti jumlah pohon keputusan (n_estimators) atau kedalaman maksimal pohon (max_depth). Penelitian ini menggunakan *GridSearchCV*, yaitu metode validasi silang untuk mencari kombinasi parameter terbaik secara sistematis. Berdasarkan penelitian Muzayanah et al. (2024), *GridSearchCV* terbukti mampu meningkatkan akurasi model Random Forest hingga 0,74, meskipun membutuhkan waktu komputasi yang lebih tinggi. Hal ini menunjukkan efektivitasnya dalam meningkatkan performa model (Muzayanah et al., 2024).

2.6. Evaluasi Kinerja Model

Evaluasi kinerja model merupakan tahap penting dalam proses pengembangan model *machine learning*, karena menentukan seberapa baik model mampu melakukan prediksi terhadap

data yang belum pernah dilihat sebelumnya. Pada penelitian ini, digunakan tiga metrik evaluasi utama, yaitu *confusion matrix*, *F1-score*, dan AUC-ROC.

a. *Confusion Matrix*

Confusion matrix adalah matriks 2x2 yang menunjukkan jumlah prediksi yang benar dan salah yang dilakukan oleh model klasifikasi, dengan membandingkan hasil prediksi dengan label aktual. Matriks ini terdiri dari empat komponen:

- *True Positive* (TP): Jumlah data positif yang berhasil diklasifikasikan dengan benar.
- *True Negative* (TN): Jumlah data negatif yang berhasil diklasifikasikan dengan benar.
- *False Positive* (FP): Jumlah data negatif yang salah diklasifikasikan sebagai positif.
- *False Negative* (FN): Jumlah data positif yang salah diklasifikasikan sebagai negatif.

Dari *confusion matrix*, beberapa metrik turunan dapat dihitung sebagai berikut:

- *Accuracy* menunjukkan seberapa banyak prediksi yang benar dari seluruh data yang diuji.

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.3)$$

- *Precision* mengukur seberapa tepat model dalam memprediksi kelas positif (misalnya, seberapa banyak pasien yang diprediksi mengidap diabetes memang benar-benar mengidap diabetes).

$$Presisi = \frac{TP}{TP+FP} \quad (2.4)$$

- *Recall* melihat seberapa banyak dari seluruh kasus positif yang berhasil dikenali oleh model.

$$Recall = \frac{TP}{TP+FN} \quad (2.5)$$

Confusion matrix beserta turunannya banyak digunakan untuk mengevaluasi performa model klasifikasi karena memberikan informasi detail terkait jenis kesalahan yang terjadi (Sokolova & Lapalme, 2009).

b. *F1-Score*

F1-score adalah rata-rata harmonis dari presisi dan recall, yang digunakan untuk menilai keseimbangan antara keduanya, terutama ketika terjadi ketidakseimbangan kelas.

- $F1 - Score = 2 \times \frac{Presisi \times Recall}{Presisi + Recall}$ (2.6)

F1-score sangat berguna pada kondisi di mana kesalahan klasifikasi terhadap kelas minoritas memiliki konsekuensi besar, seperti pada data medis atau fraud detection (Encord , 2023).

c. AUC-ROC

AUC-ROC digunakan untuk mengukur kemampuan model dalam membedakan antara kelas positif dan negatif pada berbagai ambang batas (threshold).

- ROC Curve menggambarkan hubungan antara True Positive Rate (TPR) dan False Positive Rate (FPR) pada berbagai threshold:

$$TPR = \frac{TP}{TP+FN} \; dan \; FPR = \frac{FP}{FP+TN} \qquad (2.7)$$

- AUC adalah luas area di bawah kurva ROC. Nilai AUC berkisar dari 0 hingga 1:
  1) AUC = 0.5 menunjukkan performa acak.
  2) AUC mendekati 1 menunjukkan kemampuan klasifikasi yang sangat baik.

AUC-ROC sangat cocok digunakan pada dataset yang tidak seimbang karena mempertimbangkan berbagai threshold dan memberikan gambaran menyeluruh tentang trade-off antara TPR dan FPR (Google Developer Program, 2025).

2.7. Sistem Berbasis Web

Web ialah sekumpulan halaman yang memiliki topik terkait dan umumnya disimpan di server web yang dapat diakses melalui internet atau jaringan lokal. Sebuah situs web biasanya terdiri dari beberapa halaman yang saling terhubung dan dapat diakses melalui jaringan tersebut. Halaman-halaman ini dapat membahas topik yang sama atau berbeda tetapi tetap saling berhubungan. Situs web disimpan di server web dan dapat diakses menggunakan alamat URL yang unik. Situs

web dapat digunakan untuk berbagai tujuan, seperti memberikan informasi, menjual produk atau layanan, maupun sebagai sarana komunikasi antara penggguna dan organisasi (Wei et al., 2021).

Selain itu, web juga merupakan suatu aplikasi yang menyajikan berbagai dokumen multimedia seperti teks, gambar, animasi, dan video yang dapat diakses melalui protocol HTTP *(Hypertext Transfer Protocol)*. Untuk dapat mengakses halaman-halaman tersebut, dibutuhkan perangkat lunak yang dikenal dengan nama browser yang berfungsi untuk menampilkan halaman-halaman web kepada pengguna. Browser memungkinkan pengguna untuk menjelajah berbagai situs web dan mengakses informasi yang tersedia secara online dengan mudah dan efisien (Van Sluyters, 1997).

Dalam penelitian ini sistem akan dikembangkan dengan menggunakan streamlit. Streamlit adalah sebuah *framework open source* yang dirancang untuk memudahkan pengembang dalam membuat aplikasi web interaktif untuk analisis data dan *machine learning* (Snowflake Inc., n.d.). Beberapa fitur utama dari streamlit yaitu:

a. Kemudahan penggunaan

   Streamlit dirancang agar mudah digunakan dengan sintaks yang sederhana dan intuitif. Pengguna hanya perlu menulis kode python untuk membuat elemen antarmuka pengguna seperti grafik, tabel, dan input interaktif.

b. Interaktivitas

   Pengguna dapat menambahkan elemen interaktif seperti slider, dropdown, dan input teks yang memungkinkan pengguna untuk bisa berinteraksi dengan data secara langsung.

c. Integrasi dengan library python

   Streamlit mudah untuk diintegrasikan dengan berbagai library python popular seperti Pandas, NumPy, Matplotlib, dan Plotly, sehingga memberikan kemudahan bagi pengguna dalam memvisualisasikan dan menganalisis data (Panchal, 2024).

Selain itu, beberapa penelitian telah memanfaatkan aplikasi berbasis web untuk diagnosis penyakit menggunakan *machine learning*. Salah satunya adalah penelitian oleh (Mandal & Pradhan, 2023) yang mengembangkan

aplikasi *Predict2Protect* untuk deteksi dini penyakit jantung berbasis web. Aplikasi ini dirancang dengan antarmuka interaktif dan memungkinkan pengguna untuk menginput data medis guna mendapatkan prediksi secara instan. Pendekatan ini menunjukkan bahwa platform berbasis web dapat menjadi alat yang efektif dan praktis dalam mendukung diagnosis medis berbasis *machine learning*.

2.8. Black-Box Testing

Black-box testing merupakan metode pengujian perangkat lunak yang difokuskan pada fungsi sistem dari sisi pengguna, tanpa melihat struktur internal atau kode program dari sistem. Pengujian dilakukan dengan memberikan input tertentu dan mengamati output yang dihasilkan, guna memastikan bahwa sistem dapat berjalan sesuai dengan spesifikasi yang telah dirancang (Aqil Musthafa Ar Rachman et al., 2024).

Black-box testing dapat membantu memastikan bahwa semua fitur utama dalam sistem informasi seperti input data, pemrosesan, dan hasil keluaran telah berjalan sesuai alur logika yang dirancang, tanpa memerlukan pengetahuan

tentang kode program. Metode ini dinilai cocok diterapkan pada aplikasi berbasis web karena berfokus pada pengujian antarmuka dan interaksi pengguna, serta mampu mengidentifikasi kesalahan pada fungsionalitas sistem secara menyeluruh (Syafa Iswahyudi & Eka Putra, 2025).

Dalam pengujian sistem web berbasis Streamlit pada penelitian ini, black-box testing digunakan untuk mengevaluasi berbagai fitur utama seperti:

- Input data pasien melalui form
- Upload data melalui file CSV
- Proses klasifikasi dan hasil prediksi
- Validasi input tidak sesuai
- Navigasi halaman dan fungsionalitas tombol.

Metode ini memastikan bahwa sistem dapat merespon dengan tepat terhadap semua interaksi pengguna, serta menghasilkan keluaran sesuai yang diharapkan, tanpa harus memeriksa struktur internal algoritma atau kode program.

B. Kajian Penelitian yang Relevan

Berbagai penelitian telah dilakukan untuk mengeksplorasi efektivitas algoritma random forest dalam klasifikasi penyakit diabetes. Berikut ini adalah beberapa kajian yang relevan yang menunjukkan

penerapan dan hasil dari algoritma ini dalam mendiagnosis dan memprediksi risiko diabetes.

Tabel 2.6 Kajian Penelitian yang Relevan

No.	Nama Penulis	Judul Penelitian	Hasil Penelitian
1.	(Gde Agung Brahmana Suryanegara et al., 2021b)	Peningkatan Hasil Klasifikasi pada Algoritma Random Forest untuk Deteksi Pasien Penderita Diabetes Menggunakan Metode Normalisasi	Penelitian ini menunjukkan bahwa penerapan metode normalisasi (min-max scaling) pada dataset dapat meningkatkan akurasi Random Forest dalam mendeteksi pasien diabetes, dengan akurasi mencapai 92,5%.
2.	(Setiawan et al., 2024)	Klasifikasi Tingkat Risiko Diabetes Menggunakan	Hasil evaluasi model menunjukkan bahwa algoritma Random Forest dapat

		Random Forest.	mengklasifikasikan risiko diabetes dengan akurat. Penelitian ini juga mencakup analisis seleksi fitur untuk meningkatkan kinerja model.
3.	(Mubaqarah et al., 2024)	Comparison of Random Forest and XGBoost for Diabetes Classification	Perbandingan antara Random Forest dan XGBoost menunjukkan bahwa Random Forest memberikan hasil akurasi sebesar 92% pada dataset diabetes, sedikit lebih rendah dibandingkan XGBoost dengan akurasi 94%, namun tetap unggul dalam kecepatan pelatihan model.

4.	(Iskandar et al., 2024b)	Klasifikasi Menggunakan Metode Random Forest untuk Awal Deteksi Diabetes Melitus Tipe 2	Hasil dari penelitian ini menunjukkan akurasi rata-rata sebesar 97%, dengan evaluasi model menghasilkan nilai precision 95%, recall 97%, dan F1-Score 96%. Temuan ini menegaskan bahwa model Random Forest efektif dalam mendeteksi DM Tipe 2 dan memberikan prediksi yang akurat berdasarkan konfigurasi data yang sejenis.
5.	(Ibrahim, 2024)	Perbandingan Performa Algoritma Random	Penelitian ini membandingkan performa algoritma *Random*

		Forest Classifier Dan Naive Bayes Pada Penyakit Diabetes Melitus	*Forest Classifier* dengan *Naïve Bayes* dalam memprediksi penyakit diabetes. Hasilnya menunjukkan bahwa *Random Forest Classifier* memiliki akurasi, presisi, recall, dan F1-score yang lebih tinggi dibandingkan *Naïve Bayes*. Meskipun *Random Forest Classifier* memiliki waktu eksekusi yang lebih lama, performanya secara keseluruhan lebih baik dalam

			memprediksi penyakit diabetes.

Penelitian terkait klasifikasi penyakit diabetes menggunakan algoritma Random Forest menunjukkan berbagai pendekatan dan hasil yang efektif. Gde Agung Brahmana Suryanegara et al. (2021) dalam penelitiannya mengungkapkan bahwa penggunaan metode normalisasi, khususnya min-max scaling, dapat meningkatkan akurasi algoritma Random Forest dalam mendeteksi pasien diabetes. Hasilnya menunjukkan akurasi mencapai 92,5%, yang menunjukkan bahwa proses pengolahan data sebelum digunakan dalam model (*preprocessing*) memiliki pengaruh besar terhadap akurasi dan performa model itu sendiri.

Setiawan et al. (2024) juga mengembangkan model untuk klasifikasi tingkat risiko diabetes dengan menggunakan Random Forest. Dalam penelitian ini, algoritma berhasil mengklasifikasikan risiko diabetes dengan akurasi yang sangat baik. Selain itu, penelitian ini memperkenalkan analisis seleksi fitur untuk mengoptimalkan model, menandakan pentingnya pemilihan atribut yang relevan dalam meningkatkan performa model.

Sementara itu, Mubaqarah et al. (2024) membandingkan algoritma Random Forest dengan XGBoost dalam klasifikasi diabetes. Hasil penelitian menunjukkan bahwa meskipun XGBoost memiliki akurasi sedikit lebih tinggi (94%), Random Forest tetap unggul dalam kecepatan pelatihan dengan akurasi 92%. Temuan ini menunjukkan bahwa meskipun XGBoost mungkin lebih akurat, akan tetapi Random Forest lebih efisien dalam hal waktu yang dapat sangat berguna dalam aplikasi dunia nyata.

Iskandar et al. (2024) fokus pada penggunaan Random Forest untuk deteksi awal diabetes melitus tipe 2. Dalam penelitian ini, model Random Forest menghasilkan akurasi rata-rata sebesar 97%, dengan precision 95%, recall 97%, dan F1-Score 96%. Hasil ini menegaskan efektivitas Random Forest dalam memberikan prediksi yang akurat dan andal untuk deteksi dini diabetes tipe 2.

Terakhir, Ibrahim (2024) membandingkan algoritma Random Forest dengan Naïve Bayes dalam memprediksi diabetes melitus. Penelitian ini menunjukkan bahwa Random Forest memiliki performa yang lebih baik daripada Naïve Bayes, dengan akurasi, presisi, recall, dan F1-score yang lebih tinggi. Meskipun Random Forest membutuhkan waktu eksekusi yang lebih lama, hasil

keseluruhan menunjukkan bahwa algoritma ini lebih efektif dalam mengklasifikasikan diabetes dibandingkan Naïve Bayes.

Secara keseluruhan, penelitian-penelitian ini menunjukkan bahwa Random Forest merupakan algoritma yang sangat efektif dalam mendeteksi dan mengklasifikasikan diabetes, baik untuk deteksi awal maupun untuk prediksi lebih lanjut dengan berbagai pendekatan yang dapat meningkatkan akurasi dan efisiensi model.

Penelitian ini yaitu "Klasifikasi Penyakit Diabetes Melalui Platform Web Menggunakan Algoritma Random Forest", merupakan langkah lanjutan yang bertujuan untuk mengembangkan sistem berbasis web yang dapat mengklasifikasikan risiko diabetes secara *real-time*. Dalam penelitian ini, peneliti akan menerapkan algoritma Random Forest untuk menganalisis data pasien dan memberikan prediksi yang akurat mengenai kemungkinan terjadinya diabetes. Sistem ini diharapkan dapat memberikan kemudahan akses bagi pengguna untuk memantau kesehatan mereka dan mendapatkan rekomendasi yang sesuai. Dengan memanfaatkan teknologi web, penelitian ini bertujuan untuk meningkatkan kesadaran masyarakat tentang pentingnya

deteksi dini diabetes dan memberikan alat bantu yang efektif bagi tenaga medis dalam pengelolaan penyakit ini. Penelitian ini diharapkan dapat memberikan kontribusi signifikan dalam pengembangan sistem klasifikasi penyakit diabetes yang lebih efisien dan akurat.

BAB III

METODE PENELITIAN

A. Jenis Penelitian

Penelitian ini merupakan penelitian eksperimental kuantitatif yang menggunakan pendekatan *machine learning*. Penelitian eksperimental adalah pendekatan sistematis, cermat, dan logis yang melibatkan pengendalian terhadap suatu situasi atau kondisi (Musat, 2015). Penelitian ini bertujuan untuk mengembangkan dan mengimplementasikan algoritma random forest dalam sebuah platform berbasis web untuk mendeteksi penyakit diabetes berdasarkan data pasien. Algoritma ini dievaluasi untuk mengukur peforma dalam hal akurasi, sensivitas, dan kecepatan prediksi.

B. Data

Sumber data adalah elemen krusial yang harus dipertimbangkan saat memilih metode pengumpulan data. Dalam penelitian ini, jenis sumber data yang digunakan adalah data sekunder.

Sumber data sekunder merujuk pada informasi yang diperoleh dari sumber-sumber yang telah ada sebelumnya, yang berarti data tersebut tidak dikumpulkan secara langsung oleh peneliti, melainkan diambil dari penelitian, laporan, atau basis data yang telah

dipublikasikan. Dalam penelitian ini, pengumpulan data sekunder diperoleh dari basis data publik *Mendeley Data*. Data ini telah dikurasi dan relevan untuk mendukung klasifikasi penyakit diabetes menggunakan algoritma *Random Forest*. Berikut adalah penjelasan rinci terkait sumber, karakteristik, dan pengolahan data:

1. Sumber Data

   Data yang digunakan dalam penelitian ini menggunakan data sekunder yang diperoleh dari repositori *Mendeley Data* melalui tautan berikut: https://data.mendeley.com/datasets/wj9rwkp9c2/1. Data tersebut terdiri dari informasi pasien yang mencakup berbagai atribut medis yang relevan untuk diagnosis diabetes.

2. Karakteristik Data

   Dataset yang digunakan terdiri dari 12 atribut yaitu *gender* (jenis kelamin), *age* (umur), urea, *Creatinine ratio* (Cr), HbA1c, *Cholesterol* (Chol), *Triglycerides* (TG), *High-Density Lipoprotein* (HDL), *Low-Density Lipoprotein* (LDL), Very-*Low-Density Lipoprotein* (VLDL), *Body Mass Index* (BMI) dan label *class* (kelas) diagnosis yang terdiri dari dua kelas yaitu Y (menunjukkan pasien terkena diabetes) dan N (pasien tidak terkena diabetes) (Rashid, 2020).

3. Jumlah Data

Dataset yang diperoleh dari Mendeley Data berjumlah 948 entri. Setelah dilakukan pembersihan dan filter terhadap label kelas, hanya data dengan label 'Y' (diabetes) dan 'N' (non-diabetes) yang digunakan, sehingga total data yang dipakai dalam penelitian ini adalah 791 entri. Data dengan label 'P' dikecualikan karena tidak termasuk dalam tujuan klasifikasi biner penelitian ini. Berikut disajikan tabel 3.1 contoh data yang akan digunakan dalam penelitian ini.

Tabel 3.1 Dataset Penelitian

No.	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI	CLASS
1.	F	50	4.7	46	4.9	4.2	0.9	2.4	1.4	0.5	24	N
2.	M	26	4.5	62	4.9	3.7	1.4	1.1	2.1	0.6	23	N
3.	F	59	4.7	58	4.1	4.5	1.8	1.8	1.8	1.3	22.5	N
4.	F	57	4.6	97	0.9	3.2	1.3	0.9	3	1.1	22	N
5.	M	38	6.1	83	5.4	4.5	1.7	0.9	2.8	0.8	24.6	N
6.	M	31	3	60	12.3	4.1	2.2	0.7	2.4	15.4	37.2	Y
7.	M	30	7.1	81	6.7	4.1	1.1	1.2	2.4	8.1	27.4	Y
8.	M	45	4.1	63	10.2	4.8	1.3	0.9	3.3	9.5	34.3	Y
9.	M	45	4.1	63	10.2	4.8	1.3	0.9	3.3	9.5	34.3	Y
10.	M	45	5.3	77	11.2	3.9	1.5	1.3	2	10.4	29.5	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
791.	M	54	5	67	6.9	3.8	1.7	1.1	3	0.7	33	Y

4. Kualitas Data

Sebelum digunakan, dataset diolah melalui proses *preprocessing* untuk memastikan kualitas dan kelayakan data. Langkah-langkah yang dilakukan meliputi:

a. Menghapus data tidak valid dan duplikat

Data yang mengandung nilai kosong, *outlier*, atau duplikasi dihapus atau ditangani menggunakan metode imputasi.

b. Encoding fitur kategori

Atribut non-numerik seperti jenis kelamin diubah menjadi format numerik menggunakan teknik *One-Hot Encoding*.

5. Pengelolaan Data

Setelah *preprocessing*, data dibagi menjadi dua bagian utama:

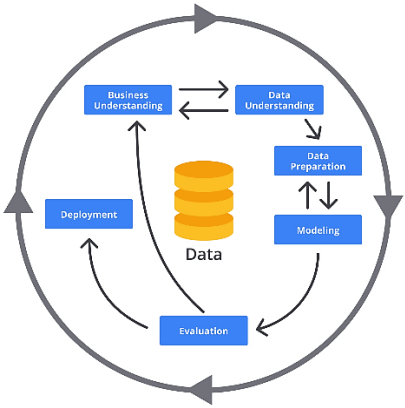
a. Data Latih (*Training Set*): Sebanyak 80% data digunakan untuk melatih model.

b. Data Uji (*Testing Set*): Sebanyak 20% data digunakan untuk mengevaluasi kinerja model. Proses pembagian dilakukan secara acak untuk memastikan distribusi data yang representatif pada masing-masing set.

C. Metode

Metode yang digunakan dalam penelitian ini yaitu CRISP-DM *(Cross Industry Standard Process for Data Mining)*. CRISP-DM adalah sebuah kerangka kerja yang terstruktur untuk mengelola proses pengolahan data. Metode ini mencakup tahapan yang dimulai dari pemahaman masalah bisnis hingga evaluasi model. Metode ini terdiri dari enam tahapan seperti pada gambar

3.1 yaitu *business understanding* (pemahaman bisnis), *data understanding* (pemahaman data), *data preparation* (persiapan data), *modelling* (pemodelan), *evaluation* (evaluasi), dan *deployment* (penerapan). Berikut penjelasan dari setiap tahapannya:



Gambar 3.1 Tahapan pada CRISP-DM

1. *Business Understanding* (Pemahaman Bisnis)

   Tahapan ini dimulai dengan mengidentifikasi masalah utama, yaitu meningkatnya prevalensi penyakit diabetes yang memerlukan sistem deteksi dini yang akurat dan mudah diakses. Tujuan dari proyek ini adalah mengembangkan model klasifikasi penyakit diabetes menggunakan algoritma Random Forest dan mengimplementasikannya dalam platform web berbasis Streamlit.

2. *Data Understanding* (Pemahaman Data)

Setelah pemahaman bisnis diperoleh, tahap berikutnya adalah memahami struktur dan karakteristik data yang digunakan. Dataset diambil dari repositori publik Mendeley Data, terdiri dari 948 entri, namun hanya 791 entri dengan label 'Y' dan 'N' yang digunakan. Data ini memiliki 12 atribut medis yang berkaitan dengan diagnosis diabetes yaitu *gender* (jenis kelamin), *age* (umur), urea, *Creatinine ratio* (Cr), HbA1c, *Cholesterol* (Chol), *Triglycerides* (TG), *HDL Cholesterol*, LDL, VLDL, dan *Body Mass Index* (BMI), dan label kelas diabetes (Y) atau non-diabetes (N).

3. *Data Preparation* (Persiapan Data)

   Setelah memahami data, dilakukan proses persiapan data. Tahapan ini mencakup pembersihan data (menghapus nilai kosong dan duplikat), transformasi data (encoding variabel kategorikal), normalisasi, dan pembagian data menjadi set pelatihan dan pengujian dengan rasio 80:20. Serta teknik SMOTE digunakan untuk menyeimbangkan kelas data.

4. *Modelling* (Pemodelan)

   Setelah data siap, proses modeling dilakukan dengan menggunakan algoritma Random Forest. Model dilatih menggunakan data training, dan untuk meningkatkan performa, dilakukan pencarian parameter terbaik menggunakan GridSearchCV.

5. *Evaluation* (Pengujian)

   Setelah model terbentuk, dilakukan evaluasi performa menggunakan data uji. Metrik yang digunakan mencakup akurasi, presisi, recall, F1-score, dan AUC-ROC. Confusion matrix juga digunakan untuk menggambarkan prediksi benar dan salah secara lebih detail.

6. *Deployment* (Penerapan)

   Tahapan akhir dalam pengembangan sistem ini adalah penerapan model klasifikasi ke dalam platform web menggunakan framework Streamlit. Sistem yang dibangun menyediakan antarmuka berbasis web yang memungkinkan pengguna untuk menginput data pasien secara manual melalui form atau secara massal melalui file CSV. Setelah data dimasukkan, sistem akan langsung menampilkan hasil klasifikasi berupa prediksi apakah pasien termasuk kategori diabetes atau non-diabetes.

   Pada tahapan ini juga mencakup pengujian sistem untuk memastikan bahwa seluruh fitur pada platform web berfungsi sebagaimana mestinya. Pengujian dilakukan dengan menggunakan metode black-box testing, yang difokuskan pada pengujian fungsionalitas utama sistem. antara lain:

   - Input data manual,

- Unggah file CSV,
- Validasi input,
- Output prediksi,
- Navigasi antarmuka.

BAB IV

HASIL DAN PEMBAHASAN

A. Data Mining

Pada tahap ini, dijelaskan proses data mining yang diterapkan dalam penelitian untuk membangun model klasifikasi penyakit diabetes. Seluruh proses mengikuti tahapan CRISP-DM yang telah dijabarkan pada Bab III. Penjelasan disusun secara sistematis mulai dari pemahaman masalah, pengolahan data, hingga penerapan model ke dalam platform web berbasis Streamlit.

4.1. *Business Understanding*

Penelitian ini bertujuan untuk mengklasifikasikan apakah seorang pasien menderita diabetes atau tidak menggunakan algoritma Random Forest. Model klasifikasi ini diimplementasikan dalam bentuk aplikasi berbasis web yang dibangun menggunakan framework Streamlit. Aplikasi ini dirancang agar mudah diakses oleh pengguna, baik tenaga medis maupun masyarakat umum.

4.2. *Data Understanding*

Data yang digunakan dalam penelitian ini diperoleh dari basis data publik Mendeley Data. Dataset awal berjumlah 948 entri yang berisi

informasi medis pasien, seperti usia, jenis kelamin, urea, kreatinin, HbA1c, kolesterol, trigliserida, HDL, LDL, VLDL, dan BMI, serta label kelas (diabetes atau non-diabetes). Namun, karena penelitian ini hanya memfokuskan klasifikasi pada dua kelas (Y untuk pasien dengan diabetes, dan N untuk non-diabetes), maka data yang digunakan hanya sebanyak 791 entri setelah menghapus entri dengan label kelas 'P'. Dataset ini selanjutnya dianalisis untuk mengetahui distribusi data, tipe atribut, dan kemungkinan adanya nilai kosong atau data duplikat yang perlu ditangani pada tahap berikutnya.

4.3. *Data Preparation*

Tahapan yang dilakukan untuk membangun sistem klasifikasi penyakit diabetes dengan algoritma random forest adalah mempersiapkan data yang akan diolah. Tahapan ini meliputi *import library*, pemuatan data, dan *preprocessing* data.

4.3.1. *Import Libraries*

Sebelum melakukan proses *preprocessing* dan pelatihan model, tahap awal yang perlu dilakukan adalah mengimpor berbagai pustaka (*libraries*) yang dibutuhkan dalam proses analisis data dan

pembangunan model machine learning. Pustaka-pustaka ini menyediakan fungsi dan tools yang diperlukan untuk membaca data, memproses atribut, membangun model, mengevaluasi performa, serta mengintegrasikan sistem ke dalam platform web.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import joblib
import time

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, roc_auc_score, confusion_matrix
from imblearn.over_sampling import SMOTE
```

Gambar 4.1 *Import Libraries*

Berikut penjelasan untuk setiap *libraries* yang digunakan dalam pemodelan ini:

a. *Pandas* adalah *library Python* yang digunakan untuk manipulasi dan analisis data, seperti mengolah data dalam bentuk DataFrame yang nantinya akan digunakan untuk membaca file berformat .csv. Data yang dibaca akan disajikan dalam bentuk tabel.
b. *NumPy* merupakan pustaka untuk komputasi numerik.

c. *Matplotlib* adalah pustaka untuk visualisasi data dalam bentuk grafik 2D. Submodul *pyplot* dari *Matplotlib* digunakan untuk membuat berbagai jenis plot, seperti plot garis, histogram, bar chart, dan scatter plot. Visualisasi ini berguna untuk memahami distribusi data dan hasil dari model yang telah dibangun.

d. *Seaborn* adalah pustaka visualisasi data yang dibangun di atas *Matplotlib*, dengan antarmuka yang lebih sederhana dan estetika yang lebih baik. *Seaborn* memudahkan pembuatan grafik statistik yang kompleks, seperti heatmap atau pairplot, serta memberikan tampilan visual yang lebih menarik dibandingkan dengan *Matplotlib*.

e. *Joblib* digunakan untuk menyimpan dan memuat objek *Python* yang besar, seperti model *machine learning* yang telah dilatih. Dengan menggunakan joblib, model yang telah dilatih dapat disimpan dalam format file dan dimuat kembali untuk digunakan tanpa perlu melatih ulang model tersebut.

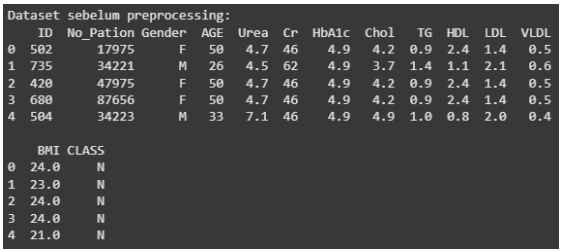
f. *Time* adalah pustaka standar *Python* yang digunakan untuk mengukur waktu eksekusi dan penundaan dalam program.
g. *Train_test_split* dan *GridSearchCV* digunakan untuk membagi data menjadi set pelatihan dan pengujian serta mencari hyperparameter terbaik dengan pencarian grid.
h. *StandardScaler* dan *LabelEncoder* digunakan untuk menstandardisasi data dan mengubah label kategori menjadi angka.
i. *RandomForestClassifier* merupakan algoritma *machine learning* berbasis ensemble untuk klasifikasi.
j. Evaluasi Model *(accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix)* digunakan untuk mengevaluasi kinerja model dengan berbagai metrik.
k. *SMOTE* merupakan teknik untuk menangani ketidakseimbangan kelas dengan menciptakan contoh sintetik dari kelas minoritas.

4.3.2. Membaca Dataset

Setelah seluruh pustaka (*library*) yang dibutuhkan berhasil diimpor, langkah selanjutnya adalah membaca dataset yang akan digunakan dalam penelitian ini. Proses pembacaan dilakukan dengan menggunakan fungsi read_csv() dari library Pandas. Adapun potongan kode program untuk membaca dataset adalah sebagai berikut:

```
data =
pd.read_csv('/content/drive/MyDrive
/Colab Notebooks/Dataset of
Diabetes  (1).csv')
print('Dataset sebelum
preprocessing:')
print(data.head())
```

Output dari kode tersebut ditampilkan pada Gambar 4.2 berikut:



```
Dataset sebelum preprocessing:
    ID  No_Pation Gender  AGE  Urea  Cr  HbA1c  Chol   TG  HDL  LDL  VLDL
0  502      17975      F   50   4.7  46    4.9   4.2  0.9  2.4  1.4   0.5
1  735      34221      M   26   4.5  62    4.9   3.7  1.4  1.1  2.1   0.6
2  420      47975      F   50   4.7  46    4.9   4.2  0.9  2.4  1.4   0.5
3  680      87656      F   50   4.7  46    4.9   4.2  0.9  2.4  1.4   0.5
4  504      34223      M   33   7.1  46    4.9   4.9  1.0  0.8  2.0   0.4

    BMI CLASS
0  24.0     N
1  23.0     N
2  24.0     N
3  24.0     N
4  21.0     N
```

Gambar 4.2 Output Membaca Dataset Asli

Dataset yang digunakan dalam penelitian ini dimuat dan dibaca menggunakan Pandas dengan tujuan untuk memverifikasi struktur dan isi awal data. Dataset tersebut berisi informasi pasien dengan sejumlah atribut, antara lain:

- ID pasien
- Nomor rekam medis
- Usia
- Rasio urea
- Rasio kreatinin
- Rasio HbA1c
- Rasio kolesterol (seperti TG, HDL, LDL, dan VLDL)
- Indeks Massa Tubuh (BMI)
- Label kelas (*CLASS*) sebagai target klasifikasi

Dataset ini akan menjadi dasar dalam proses *preprocessing*, analisis, dan pembuatan model klasifikasi yang dilakukan pada tahap-tahap selanjutnya dalam penelitian ini.

4.3.3. Menghapus Kolom yang Tidak Digunakan

Langkah selanjutnya dalam proses *data cleaning* adalah menghapus kolom-kolom yang tidak dibutuhkan dalam analisis. Pada tahap ini, kolom ID dan No_Pation dihapus karena tidak

memiliki pengaruh terhadap proses pemodelan atau hasil klasifikasi. Adapun potongan kode yang digunakan untuk menghapus kolom tersebut adalah sebagai berikut:

```
data = data.drop(columns=['ID', 'No_Pation'])
data.head()
```

Output dari kode tersebut ditampilkan pada Gambar 4.3

	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI	CLASS
0	F	50	4.7	46	4.9	4.2	0.9	2.4	1.4	0.5	24.0	N
1	M	26	4.5	62	4.9	3.7	1.4	1.1	2.1	0.6	23.0	N
2	F	50	4.7	46	4.9	4.2	0.9	2.4	1.4	0.5	24.0	N
3	F	50	4.7	46	4.9	4.2	0.9	2.4	1.4	0.5	24.0	N
4	M	33	7.1	46	4.9	4.9	1.0	0.8	2.0	0.4	21.0	N

Gambar 4.3 Menghapus Kolom ID dan No_Pation

Setelah kolom ID dan No_Pation dihapus, maka dataset hanya menyisakan atribut-atribut yang relevan dan akan digunakan dalam proses penelitian. Atribut-atribut tersebut meliputi:

- Gender (0 untuk perempuan/*female*, 1 untuk laki-laki/*male*)
- Age (usia)
- Urea

- Cr (kreatinin)
- HbA1c
- Chol (kolesterol total)
- TG (trigliserida)
- HDL (kolesterol baik)
- LDL (kolesterol jahat)
- VLDL
- BMI (indeks massa tubuh)
- CLASS (label kelas: *N* untuk tidak diabetes, *Y* untuk diabetes)

Dengan menyisakan hanya atribut-atribut yang relevan, proses analisis dan pemodelan dapat dilakukan dengan lebih efisien dan akurat.

4.3.4. Pemeriksaan dan Penghapusan Data Duplikat

Setelah sebelumnya dilakukan penghapusan terhadap kolom atribut yang tidak relevan, tahap selanjutnya adalah melakukan pemeriksaan terhadap kemungkinan adanya data duplikat dalam dataset. Data duplikat perlu diidentifikasi dan dihapus agar kualitas data tetap terjaga dan hasil analisis tidak bias. Adapun potongan kode

yang digunakan untuk mengecek jumlah data duplikat adalah sebagai berikut:

```
data.duplicated().sum()
```

Output dari kode di atas menunjukkan:

```
np.int64(169)
```

Artinya, terdapat 169 baris data duplikat dalam dataset.

Untuk menjaga integritas data dan mencegah terjadinya *overfitting* atau pengaruh yang tidak seimbang pada model, seluruh data duplikat tersebut kemudian dihapus. Berikut adalah potongan kode yang digunakan untuk menghapus data duplikat:

```
data.drop_duplicates().sum()
```

Output setelah penghapusan:

```
np.int64(0)
```

Hal ini menunjukkan bahwa seluruh data duplikat telah berhasil dihapus, dan tidak ada lagi baris yang terdeteksi sebagai duplikat. Dengan demikian, dataset telah dibersihkan dari duplikasi

dan siap digunakan untuk proses analisis lebih lanjut.

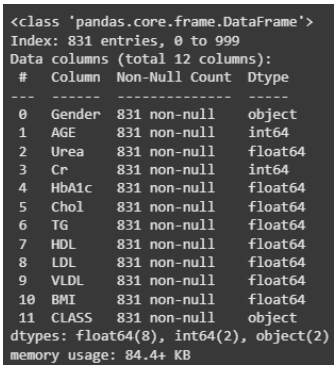
4.3.5. Cek Tipe Data dan *Missing Value*

Setelah dilakukan pembersihan terhadap kolom atribut yang tidak digunakan serta penghapusan data duplikat, langkah selanjutnya adalah memeriksa tipe data dari setiap atribut dan mengecek keberadaan nilai kosong (missing value) dalam dataset. Pemeriksaan ini penting dilakukan untuk memastikan bahwa data berada dalam format yang sesuai dan siap untuk tahap pemrosesan selanjutnya. Adapun potongan kode yang digunakan untuk melihat informasi umum tentang dataset adalah sebagai berikut:

```
data.info()
```

Output dari kode di atas ditampilkan pada Gambar 4.4 Berdasarkan output tersebut, diketahui bahwa dataset terdiri dari:

- 8 kolom dengan tipe data *float,*
- 2 kolom dengan tipe data *integer,* dan
- 2 kolom dengan tipe data *object.*



Gambar 4.4 Cek Tipe data

Kolom dengan tipe data *object* ini nantinya perlu dikonversi ke dalam format numerik agar dapat diproses oleh algoritma pembelajaran mesin. Selanjutnya, untuk melihat dimensi dari dataset, digunakan kode berikut:

```
data.shape
```

Output dari kode tersebut adalah:

```
(831,12)
```

Hasil tersebut menunjukkan bahwa dataset memiliki 831 baris (sampel data) dan 12 kolom (fitur atau variabel).

Untuk memastikan tidak terdapat nilai kosong, digunakan potongan kode berikut:

```
data.isnull().sum()
```

Output dari kode di atas dapat dilihat pada tabel berikut:

Tabel 4.1 Jumlah Missing Value per Atribut

Atribut	Jumlah Nilai Kosong
Age	0
Urea	0
Cr	0
HbA1c	0
TG	0
HDL	0
LDL	0
VLDL	0
BMI	0
Class	0

Pemeriksaan nilai kosong merupakan salah satu tahap penting dalam proses *data preprocessing*, karena nilai kosong dapat memengaruhi performa model yang akan dilatih. Berdasarkan hasil yang ditampilkan pada tabel 4.1, dapat disimpulkan bahwa tidak terdapat nilai kosong pada seluruh kolom dalam dataset,

sehingga tidak diperlukan penanganan khusus terhadap *missing value* pada tahap ini.

4.3.6. Membersihkan Label Target

Pada tahap sebelumnya telah diidentifikasi tipe data dan nilai yang hilang *(missing value)* pada dataset. Selanjutnya, pada tahap ini dilakukan proses pembersihan terhadap label target, yaitu kolom CLASS, yang mengandung data tidak konsisten atau tidak relevan. Langkah pertama adalah menampilkan distribusi awal dari nilai pada kolom CLASS dengan potongan kode berikut:

```
data['CLASS'].value_counts()
```

Hasil dari potongan kode tersebut disajikan pada Tabel 4.2 berikut:

Tabel 4.2 Jumlah Class Sebelum Dibersihkan

Class	Jumlah
Y	691
N	95
P	40
Y	4
N	1
Total	791

Dari tabel di atas terlihat bahwa terdapat duplikasi label kelas, seperti Y dan Y (dengan spasi) serta N dan N , yang seharusnya dianggap sebagai satu kategori. Untuk menyatukan label-label tersebut, maka dilakukan proses pembersihan spasi pada nilai-nilai dalam kolom CLASS menggunakan potongan kode berikut:

```
data['CLASS'].str.strip()
data['CLASS'].value_counts()
```

Hasil setelah pembersihan spasi ditampilkan dalam tabel 4.3 berikut:

Tabel 4.3 Jumlah Kelas Setelah Penghapusan Spasi

Class	Jumlah
Y	695
N	96
P	40

Selanjutnya, dilakukan penyaringan data untuk menghapus baris dengan label P, karena kelas ini tidak akan digunakan dalam proses analisis lebih

lanjut. Proses ini dilakukan dengan perintah berikut:

```
data = data[data['CLASS'] != 'P']
data['CLASS'].value_counts()
```

Perintah di atas akan menghapus seluruh entri yang memiliki label P, sehingga hanya kelas Y dan N yang dipertahankan. Distribusi akhir dari kelas setelah pembersihan dapat dilihat pada tabel 4.4 berikut:

Tabel 4.4 Jumlah Kelas Setelah Pembersihan Final

Class	Jumlah
Y	695
N	96
Total	791

Setelah melalui proses pembersihan, dataset telah memiliki label target yang konsisten dan hanya terdiri dari dua kelas utama yaitu Y dengan jumlah 695 dan N dengan jumlah 96.

4.3.7. Mengidentifikasi Kolom Kategorikal

Setelah label target berhasil dibersihkan dan hanya terdiri dari dua nilai, yaitu Y dan N, tahap

selanjutnya adalah mengidentifikasi kolom-kolom yang bertipe kategorikal (non-numerik). Identifikasi ini penting dilakukan karena kolom dengan tipe data non-numerik perlu dikonversi menjadi format numerik sebelum dapat digunakan dalam proses pelatihan model.

Adapun potongan kode yang digunakan untuk mengidentifikasi kolom kategorikal adalah sebagai berikut:

```
d_list = data.select_dtypes(include=
['object']).columns.tolist()
print(d_list)
```

Output dari kode tersebut adalah:

```
['Gender', CLASS']
```

Berdasarkan output tersebut, diketahui bahwa terdapat dua kolom bertipe *object* (non-numerik), yaitu Gender dan CLASS. Karena algoritma pembelajaran mesin hanya dapat memproses data numerik, maka kedua kolom tersebut perlu melalui proses encoding untuk mengubah nilainya menjadi bentuk numerik. Proses encoding ini akan dijelaskan lebih lanjut pada tahapan berikutnya dalam preprocessing data.

4.3.8. Encoding Data Kategorikal

Setelah sebelumnya diidentifikasi bahwa kolom Gender dan CLASS bertipe non-numerik (object), maka langkah selanjutnya adalah melakukan proses encoding untuk mengubah data kategorikal tersebut menjadi data numerik agar dapat digunakan dalam proses pemodelan. Adapun potongan kode program yang digunakan adalah sebagai berikut:

```
le = LabelEncoder()
for i in d_list:
   le.fit(data[i])
   data[i] =
le.fit_transform(data[i])
data.info()
```

Output dari proses tersebut ditampilkan pada Gambar 4.5

```
<class 'pandas.core.frame.DataFrame'>
Index: 791 entries, 0 to 999
Data columns (total 12 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Gender  791 non-null    int64
 1   AGE     791 non-null    int64
 2   Urea    791 non-null    float64
 3   Cr      791 non-null    int64
 4   HbA1c   791 non-null    float64
 5   Chol    791 non-null    float64
 6   TG      791 non-null    float64
 7   HDL     791 non-null    float64
 8   LDL     791 non-null    float64
 9   VLDL    791 non-null    float64
 10  BMI     791 non-null    float64
 11  CLASS   791 non-null    int64
```

Gambar 4.5 Proses Encoding Kolom Kategorikal

Proses encoding dilakukan dengan menggunakan LabelEncoder dari library *scikit-learn*. Setiap nilai unik dalam kolom kategorikal diubah menjadi angka integer. Tujuan dari langkah ini adalah untuk memastikan bahwa semua fitur dalam dataset berada dalam format numerik yang dapat diproses oleh algoritma pembelajaran mesin.

Setelah proses encoding selesai, dilakukan pengecekan kembali menggunakan perintah data.info() untuk memastikan bahwa tipe data pada kolom Gender dan CLASS telah berubah menjadi tipe data numerik. Hasil dari pengecekan tersebut ditampilkan pada gambar 4.5, yang menunjukkan bahwa kedua kolom tersebut kini memiliki tipe data integer.

Data Setelah Encoding:

	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI	CLASS
0	0	50	4.7	46	4.9	4.2	0.9	2.4	1.4	0.5	24.0	0
1	1	26	4.5	62	4.9	3.7	1.4	1.1	2.1	0.6	23.0	0
4	1	33	7.1	46	4.9	4.9	1.0	0.8	2.0	0.4	21.0	0
5	0	45	2.3	24	4.0	2.9	1.0	1.0	1.5	0.4	21.0	0
6	0	50	2.0	50	4.0	3.6	1.3	0.9	2.1	0.6	24.0	0

Gambar 4.6 Dataset Setelah Proses Encoding

Dataset yang telah melalui proses encoding, sebagaimana ditunjukkan pada Gambar 4.6, kini sepenuhnya terdiri dari data numerik dan siap digunakan untuk tahap *training*, analisis, atau pemodelan lebih lanjut.

4.4. *Modelling*

Setelah seluruh proses preprocessing selesai dilakukan dan data telah siap, langkah selanjutnya adalah membangun model klasifikasi menggunakan algoritma Random Forest Classifier. Model ini dilatih menggunakan data yang telah dibersihkan dan diproses sebelumnya. Untuk memperoleh performa model yang optimal, dilakukan proses tuning hyperparameter menggunakan metode GridSearchCV, yang bertujuan untuk menemukan kombinasi parameter terbaik dalam algoritma Random Forest.

4.4.1. Menyiapkan Fitur dan Target

Dataset yang telah bersih selanjutnya dibagi menjadi dua komponen utama, yaitu fitur (X) dan target (Y). Pemisahan ini diperlukan agar model dapat mempelajari hubungan antara fitur input dengan label output yang menjadi target

klasifikasi. Potongan kode program yang digunakan adalah sebagai berikut:

```
X = data.drop('CLASS', axis=1)
Y = data['CLASS']
```

Kode di atas memisahkan kolom CLASS sebagai target (Y), sedangkan seluruh kolom lainnya digunakan sebagai fitur (X). Proses ini penting agar algoritma dapat membangun model yang mampu mengidentifikasi pola dari fitur-fitur dan mengaitkannya dengan label kelas yang sesuai.

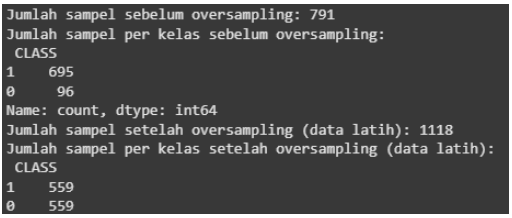
4.4.2. Membagi Data dan Melakukan Resampling

Tahap berikutnya dalam proses pemodelan adalah membagi data menjadi data latih (training set) dan data uji (testing set), serta melakukan proses resampling untuk menangani ketidakseimbangan kelas pada data latih. Berikut adalah potongan kode program yang digunakan:

```
#Bagi data menjadi data latih dan
data uji
X_train, X_test, y_train, y_test =
train_test_split(X, Y,
test_size=0.2, random_state=42)
# Inisialisasi SMOTE
```

```
smote = SMOTE(random_state=42)
# Oversampling data latih
X_train_resampled,
y_train_resampled =
smote.fit_resample(X_train,
y_train)
```

Hasil dari proses ini ditampilkan pada Gambar 4.7.



Gambar 4.7 Data Sebelum dan Sesudah Oversampling

Data dibagi dengan rasio 80:20, yaitu 80% sebagai data latih dan 20% sebagai data uji. Selanjutnya, untuk mengatasi permasalahan ketidakseimbangan kelas, digunakan teknik SMOTE *(Synthetic Minority Oversampling Technique)*. SMOTE merupakan metode oversampling yang menghasilkan sampel sintetis untuk kelas minoritas dengan cara membuat data baru berdasarkan tetangga terdekat, bukan sekadar menyalin data yang ada. Pendekatan ini bertujuan untuk memperbaiki distribusi kelas

tanpa meningkatkan risiko overfitting yang sering terjadi pada teknik oversampling konvensional.

Sebelum dilakukan oversampling, dataset terdiri dari 791 sampel, dengan distribusi kelas yang tidak seimbang:

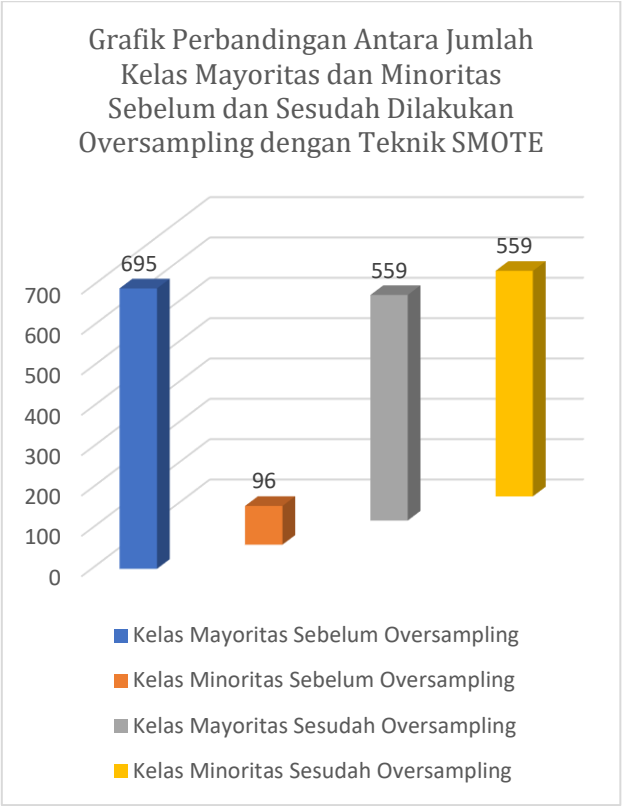
- Kelas mayoritas (1/diabetes) sebanyak 695 sampel
- Kelas minoritas (0/tidak diabetes) sebanyak 96 sampel

Ketidakseimbangan ini dapat mempengaruhi kinerja model, terutama dalam mengenali kelas minoritas. Oleh karena itu, teknik SMOTE diterapkan pada data latih untuk menyamakan jumlah sampel di setiap kelas.

Setelah proses oversampling, total data latih meningkat menjadi 1.118 sampel, dengan distribusi yang seimbang yaitu:

- 559 sampel untuk kelas 1
- 559 sampel untuk kelas 0

Untuk lebih jelas, berikut dilampirkan grafik perbandingan kelas mayoritas dan minoritas sebelum dan sesudah dilakukan oversampling:



Dengan distribusi kelas yang lebih seimbang, model diharapkan dapat belajar secara adil terhadap kedua kelas dan menghasilkan performa klasifikasi yang lebih baik.

4.4.3. Visualisasi Korelasi Antar Fitur

Visualisasi matriks korelasi dilakukan untuk memahami hubungan antar fitur dalam data latih.

Tujuan dari visualisasi ini adalah untuk mengetahui sejauh mana antar fitur saling berkorelasi, serta mendeteksi adanya korelasi tinggi yang berpotensi menimbulkan masalah multikolinearitas, yaitu kondisi di mana dua atau lebih fitur sangat berkaitan sehingga dapat mempengaruhi performa model prediktif. Berikut potongan kode program yang digunakan:

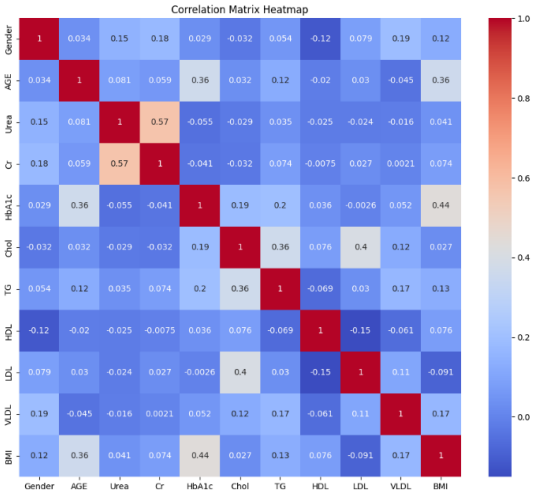
```
Plt.figure(figsize=(12,10))
sns.heatmap(pd.DataFrame(X_train).c
orr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix
Heatmap')
plt.show()
```

Visualisasi menggunakan heatmap dari pustaka *Seaborn* ini menyajikan hubungan antar fitur dalam bentuk representasi visual yang intuitif. Nilai korelasi Pearson ditampilkan secara langsung di setiap sel matriks, dengan gradasi warna dari skema coolwarm yang menunjukkan kekuatan dan arah hubungan antar variabel:

- Warna merah menunjukkan korelasi positif yang kuat (nilai mendekati +1).

- Warna biru menunjukkan korelasi negatif yang kuat atau hubungan yang lemah (nilai mendekati -1 atau 0).

Gambar hasil visualisasi ditampilkan pada Gambar 4.8. Pada tahap *exploratory data analysis*, pembuatan *Correlation Matrix Heatmap* digunakan untuk melihat kekuatan hubungan linier antar fitur dalam dataset menggunakan koefisien korelasi Pearson, yang berkisar antara -1 hingga 1.



Gambar 4.8 *Correlation Matix Heatmap*

Berdasarkan gambar 4.8, terlihat bahwa sebagian besar fitur memiliki tingkat korelasi yang rendah, yang berarti masing-masing fitur memberikan informasi yang relatif unik dan independen terhadap model. Namun, terdapat

beberapa pasangan fitur dengan korelasi cukup tinggi, antara lain:

- Urea dan Cr dengan korelasi sebesar 0,57.
- HbA1c dan BMI dengan korelasi sebesar 0,44.

Hubungan seperti ini penting untuk diperhatikan dalam proses seleksi fitur, karena korelasi tinggi antar fitur dapat menyebabkan redundansi informasi dan mengganggu stabilitas model, terutama dalam algoritma yang sensitif terhadap multikolinearitas seperti regresi linier.

4.4.4. Grid Search untuk Hyperparameter Tuning

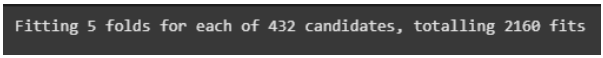
Untuk memperoleh kinerja model yang optimal, dilakukan pencarian kombinasi hyperparameter terbaik pada algoritma Random Forest Classifier menggunakan metode GridSearchCV. GridSearchCV bekerja dengan mengevaluasi semua kemungkinan kombinasi hyperparameter yang telah ditentukan, seperti jumlah pohon (n_estimators), kedalaman maksimum pohon (max_depth), jumlah fitur yang dipertimbangkan saat pemisahan (max_features), dan parameter lainnya. Berikut potongan kode yang digunakan:

```
params = {
   'n_estimators': [100, 200, 300],
   'criterion': ['gini', 'entropy'],
   'max_depth': [10, 20, 30, None],
   'min_samples_leaf': [1, 2, 4],
   'min_samples_split': [2, 5, 10],
   'max_features': ['sqrt', 'log2']
}
rf_model=RandomForestClassifier(ran
dom_state=42)
grid_search = GridSearchCV(
   estimator=rf_model,
   param_grid=params,
   cv=5,
   scoring='f1',
   verbose=2,
   n_jobs=-1
)
start_time = time.time()
grid_search.fit(X_train, y_train)
end_time = time.time()
```

Setiap kombinasi diuji menggunakan teknik validasi silang 5-fold cross-validation, dan dievaluasi berdasarkan F1-score. Metrik F1 dipilih karena lebih sesuai untuk menangani

ketidakseimbangan kelas dalam data, dengan mempertimbangkan keseimbangan antara *precision* dan *recall*.

Berdasarkan konfigurasi parameter yang digunakan, total terdapat 432 kombinasi yang diuji. Dengan validasi silang 5-fold, proses pelatihan dilakukan sebanyak 2.160 kali (432 × 5), sebagaimana ditunjukkan pada Gambar 4.9.



Gambar 4.9 Output Proses Fitting

Melalui proses ini, model Random Forest dapat dibangun dengan konfigurasi parameter yang paling sesuai berdasarkan data yang digunakan, sehingga diharapkan mampu menghasilkan performa prediksi yang lebih akurat dibandingkan jika menggunakan parameter default.

4.4.5. Mengambil Model Terbaik

Setelah dilakukan pencarian hyperparameter menggunakan teknik Grid Search, diperoleh model Random Forest terbaik dengan konfigurasi parameter yang memberikan performa paling optimal berdasarkan metrik F1-score. Berikut

adalah potongan kode untuk mengambil model tersebut:

```
best_model =
grid_search.best_estimator_
print(grid_search.best_estimator_)
```

Output:

```
RandomForestClassifier(max_depth=10
, min_samples_split=5,
random_state=42)
```

Model terbaik ini menggunakan pengaturan max_depth=10, min_samples_split=5, dan random_state=42.

- max_depth=10 berfungsi untuk membatasi kedalaman maksimum pohon keputusan. Pembatasan ini dilakukan agar model tidak terlalu kompleks dan terhindar dari overfitting pada data pelatihan.
- min_samples_split=5 mengatur jumlah minimum sampel yang dibutuhkan untuk memisahkan suatu node. Tujuannya adalah untuk menghindari pemisahan yang terlalu sering dan menjaga agar model tetap stabil.

- random_state=42 digunakan agar proses pelatihan dapat menghasilkan hasil yang konsisten setiap kali dijalankan.

Dengan pengaturan tersebut, model diharapkan mampu menghasilkan prediksi yang lebih seimbang dan dapat diandalkan pada data baru.

4.5. *Evaluation*

Setelah model selesai dilatih, langkah selanjutnya adalah mengujinya menggunakan data uji. Evaluasi dilakukan dengan menggunakan berbagai metrik seperti akurasi, presisi, recall, F1-score, dan AUC-ROC untuk mengetahui seberapa baik model bekerja.

4.5.1. Pengujian Model

Setelah mendapatkan model terbaik dari proses pelatihan dan tuning sebelumnya, model tersebut kemudian digunakan untuk memprediksi data uji. Berikut potongan kode yang digunakan:

```
y_pred = best_model.predict(X_test)
hasil_prediksi =
X_test.copy(deep=True)
hasil_prediksi['y_pred'] = y_pred
hasil_prediksi
```

Kode di atas menggunakan fungsi predict() dari model terbaik (best_model) untuk menghasilkan prediksi berdasarkan data uji (X_test). Hasil prediksi tersebut kemudian disalin ke dalam sebuah dataframe baru bernama hasil_prediksi, yang merupakan salinan dari X_test dan ditambahkan satu kolom baru bernama y_pred berisi label hasil prediksi.

	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI	y_pred
423	0	66	5.5	56	9.9	4.4	2.0	1.0	2.5	0.9	29.0	1
44	0	33	2.7	47	5.0	4.2	1.4	1.3	2.6	0.7	24.0	0
536	1	68	7.1	79	9.4	3.7	3.7	0.8	1.4	1.6	31.0	1
456	0	58	2.7	53	7.3	4.8	3.3	0.9	2.6	1.5	30.0	1
724	1	54	5.4	68	3.0	1.2	0.7	1.0	0.5	1.5	33.0	1

Gambar 4.10 Output Tes Model dengan Data Uji

Hasil prediksi ditampilkan pada Gambar 4.10, yang menunjukkan bagaimana model memetakan setiap sampel data uji ke dalam dua kategori:

- 1 menandakan pasien terdeteksi memiliki diabetes
- 0 menunjukkan pasien tidak terdeteksi memiliki diabetes

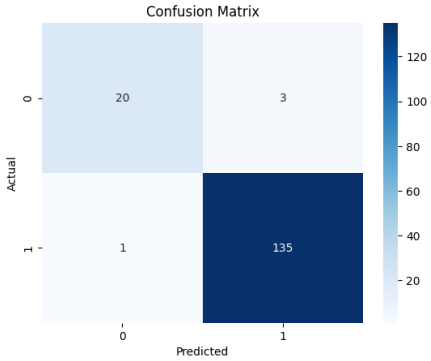
Berdasarkan hasil tersebut, model telah berhasil melakukan klasifikasi awal terhadap data uji secara otomatis dan siap dievaluasi menggunakan metrik yang telah disebutkan sebelumnya.

4.5.2. Evaluasi Model

Untuk mengetahui seberapa baik model bekerja dalam mengklasifikasikan data, digunakan beberapa metrik evaluasi, yaitu: accuracy, precision, recall, F1-score, dan ROC AUC. Berikut adalah potongan kode program yang digunakan untuk menghitung metrik tersebut:

```
# Confusion Matrix
cm = confusion_matrix(y_test,
y_pred)
sns.heatmap(cm, annot=True,
fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

Outputnya yaitu:



Gambar 4.11 Hasil *Confusion Matrix*

Tabel 4.5 Hasil *Confusion Matrix*

PREDIKSI	AKTUAL		
		Positif	Negatif
	Positif	135	1
	Negatif	3	20

Dari data tersebut diperoleh:

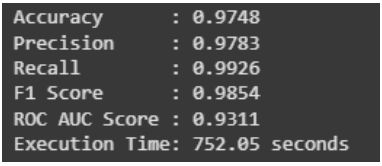
a. Jumlah data positif yang berhasil dideteksikan dengan benar (TP): 135
b. Jumlah data negatif yang berhasil dideteksikan dengan benar (TN): 20
c. Jumlah data negatif yang salah dideteksikan sebagai positif (FP): 1
d. Jumlah data positif yang salah dideteksikan sebagai negatif (FN): 3

Kemudian dilanjut dengan menghitung akurasi, presisi, recall, fi-score, dan juga auc roc.

```
accuracy = accuracy_score(y_test,
y_pred)
precision = precision_score(y_test,
y_pred)
recall = recall_score(y_test,
y_pred)
f1 = f1_score(y_test, y_pred)
```

```
roc_auc = roc_auc_score(y_test, y_pred)
execution_time = end_time - start_time
print("Best Parameters:", grid_search.best_params_)
print(f"Accuracy :{accuracy:.4f}")
print(f"Precision :{precision:.4f}")
print(f"Recall : {recall:.4f}")
print(f"F1 Score : {f1:.4f}")
print(f"ROC AUC Score : {roc_auc:.4f}")
print(f"Execution Time: {execution_time:.2f} seconds")
```

Output:



Gambar 4.12 Hasil Metrik Evaluasi

Dari hasil evaluasi model seperti yang ditampilkan pada Gambar 4.12, diperoleh nilai-nilai sebagai berikut:

- *Akurasi*

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

$$= \frac{135+20}{135+20+3+1}$$

$$= \frac{155}{159} = 0{,}9748\ atau\ 97{,}48\%$$

- *Presisi*

$$= \frac{TP}{TP + FP}$$

$$= \frac{135}{135 + 3}$$

$$= \frac{135}{138} = 0{,}9783\ atau\ 97{,}83\%$$

- *Recall*

$$= \frac{TP}{TP + FN}$$

$$= \frac{135}{135 + 1}$$

$$= \frac{135}{136} = 0{,}9926\ atau\ 99{,}26\%$$

- *F1-score*

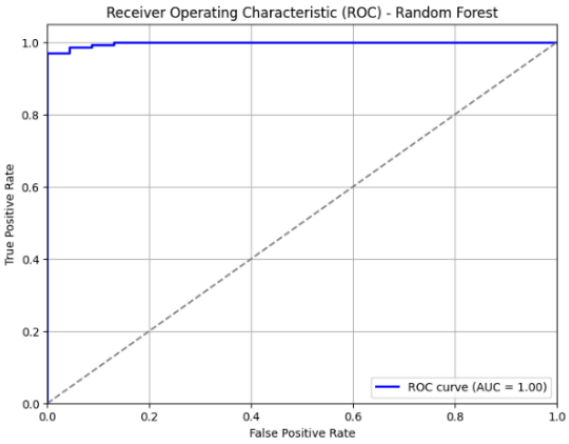
$$= 2 \times \frac{Presisi \times Recall}{Presisi + Recall}$$

$$= 2 \times \frac{0{,}9783 \times 0{,}9926}{0{,}9783 + 0{,}9926}$$

$$= 2 \times \frac{0{,}9712}{1{,}9709}$$

$$= 2 \times 0{,}4927 = 0{,}9854\ atau\ 98{,}54\%$$

- ROC AUC: 93,11%



Gambar 4.13 Hasil AUC-ROC

Kurva AUC-ROC menghasilkan nilai yang mendekati 1, menandakan bahwa model sangat baik dalam membedakan pasien yang terdiagnosis diabetes dan yang tidak.

- Waktu eksekusi pelatihan: **752,05 detik**

Hasil ini menunjukkan bahwa model bekerja sangat baik dalam mengenali dan mengklasifikasikan data, terutama dalam mendeteksi pasien yang terindikasi diabetes.

4.5.3. *Confusion Matrix*

Untuk melihat lebih detail performa model dalam mendeteksikan setiap kelas, digunakan *confusion matrix*. Matriks ini membantu

mengidentifikasi jumlah prediksi yang benar maupun salah untuk masing-masing kategori. Berikut adalah potongan kode untuk membuat confusion matrix:

```
cm = confusion_matrix(y_test,
y_pred)
sns.heatmap(cm, annot=True,
fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

Visualisasi confusion matrix ini menggunakan heatmap dengan skala warna biru (Blues), di mana setiap sel menampilkan jumlah prediksi untuk kombinasi kelas aktual dan kelas prediksi. Nilai-nilai diagonal pada matriks menunjukkan jumlah prediksi yang benar untuk masing-masing kelas, sedangkan nilai non-diagonal merepresentasikan jumlah kesalahan prediksi.

4.5.4. AUC-ROC

Untuk mengukur performa model dalam membedakan antara kelas positif dan negatif, digunakan *Receiver Operating Characteristic* (ROC)

Curve dan *Area Under the Curve* (AUC). ROC curve menggambarkan hubungan antara *False Positive Rate* (FPR) dan *True Positive Rate* (TPR) pada berbagai ambang batas (*threshold*), sedangkan AUC menunjukkan luas area di bawah kurva tersebut sebagai ukuran kinerja model secara keseluruhan. Berikut adalah potongan kode untuk menghasilkan ROC curve dan menghitung AUC:

```
from sklearn.metrics import
roc_curve, auc
import matplotlib.pyplot as plt

# Misal y_test dan prediksi
probabilitas dari model
y_pred_proba =
model.predict_proba(X_test)[:,1]  #
Ambil probabilitas kelas 1

# Hitung FPR, TPR
fpr, tpr, thresholds =
roc_curve(y_test, y_pred_proba)
# Hitung AUC
roc_auc = auc(fpr, tpr)
```

```
# Plot ROC Curve
plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, color='blue',
lw=2, label='ROC curve (AUC =
%0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1],
color='gray', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating
Characteristic (ROC) - Random
Forest')
plt.legend(loc="lower right")
plt.grid()
plt.show()
```

Pada grafik ROC curve, sumbu horizontal menunjukkan nilai *False Positive Rate* (FPR), sedangkan sumbu vertikal menunjukkan *True Positive Rate* (TPR). Garis diagonal berfungsi sebagai pembanding, mewakili performa model acak. Semakin dekat kurva ROC dengan sudut kiri atas grafik, semakin baik kemampuan model dalam membedakan antara kelas positif dan

negatif. Nilai AUC yang tinggi (mendekati 1) menunjukkan kinerja klasifikasi yang sangat baik.

4.5.5. Evaluasi Performa Model

Setelah dilakukan visualisasi *ROC Curve* dan perhitungan nilai AUC pada bagian sebelumnya, langkah selanjutnya adalah melakukan evaluasi menyeluruh terhadap performa model untuk mengidentifikasi potensi overfitting. Evaluasi dilakukan dengan membandingkan nilai AUC pada data latih dan data uji. Berikut adalah potongan kode yang digunakan:

```
from sklearn.model_selection import
train_test_split
from sklearn.ensemble import
RandomForestClassifier
from sklearn.metrics import
roc_auc_score

# Split data
X_train, X_test, y_train, y_test =
train_test_split(X, Y,
test_size=0.2, random_state=42)

# Train model
```

```
model =
RandomForestClassifier(random_state
=42)
model.fit(X_train, y_train)

# Predict
y_train_pred =
model.predict_proba(X_train)[:, 1]
y_test_pred =
model.predict_proba(X_test)[:, 1]

# Hitung AUC
train_auc = roc_auc_score(y_train,
y_train_pred)
test_auc = roc_auc_score(y_test,
y_test_pred)

print(f"AUC Training:
{train_auc:.2f}")
print(f"AUC Testing:
{test_auc:.2f}")
```

Output dari kode di atas akan menunjukkan nilai AUC untuk data latih dan data uji, yang divisualisasikan pada Gambar 4.14 berikut:

Gambar 4.14 Cek Overfitting Model

Data awal dibagi menjadi dua bagian, yaitu data latih (80%) dan data uji (20%) menggunakan fungsi *train_test_split()*. Model Random Forest dilatih menggunakan data latih, kemudian dilakukan prediksi probabilitas untuk kedua subset data dengan metode *predict_proba()*.

Nilai AUC *(Area Under the Curve)* kemudian dihitung untuk data latih dan data uji menggunakan *roc_auc_score()*. Perbandingan antara keduanya digunakan untuk menilai apakah terjadi overfitting, yaitu ketika model memiliki performa sangat baik pada data latih tetapi menurun pada data uji, yang menunjukkan kurangnya kemampuan generalisasi terhadap data baru.

4.6. *Deployment*

Model yang telah dilatih disimpan menggunakan library joblib dan siap digunakan dalam platform web. Proses deployment mencakup pengintegrasian model ke dalam dua halaman utama pada aplikasi, yaitu halaman input

manual dan halaman upload file CSV. Berikut potongan kode untuk menyimpan model:

```
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Save model-nya
Import joblib
Joblib.dump(model,  '/content/drive
/MyDrive/Colab
Notebooks/rfc_diabet_model.pkl')
```

Setelah model Random Forest Classifier dilatih menggunakan data latih, model disimpan dalam format .pkl menggunakan joblib.dump(). File model ini disimpan ke direktori yang telah ditentukan sehingga dapat digunakan kembali tanpa perlu proses pelatihan ulang. File .pkl inilah yang digunakan pada aplikasi web untuk prediksi data baru.

4.6.1. Integrasi Model pada Halaman Input Manual

Berikut potongan kode untukk memuat model dan membuat antarmuka input manual menggunakan streamlit:

```
import streamlit as st
import pandas as pd
```

```
import joblib

model =
joblib.load("model/rfc_diabet_
model.pkl")
model_accuracy = 0,09748

st.title("Input Manual Data
Pasien")
```

Pada bagian ini, Streamlit digunakan untuk membangun antarmuka pengguna yang memungkinkan pengguna memasukkan data pasien secara manual. Model yang telah disimpan sebelumnya dimuat kembali dengan joblib.load() dan digunakan untuk melakukan prediksi. Nilai akurasi model sebesar 97,48% juga disimpan dalam variabel model_accuracy sebagai informasi performa model.

4.6.2. Integrasi Model pada Halaman Upload File CSV

Berikut potongan kode untuk halaman upload file CSV:

```
import streamlit as st
import pandas as pd
```

```
import joblib

model =
joblib.load("model/rfc_diabet_
model.pkl")

st.title("Upload File CSV")
```

Pada bagian ini, antarmuka juga dibangun menggunakan Streamlit, yang memungkinkan pengguna untuk mengunggah file berformat .csv. Model yang telah disimpan sebelumnya dimuat kembali menggunakan joblib, dan digunakan untuk memproses serta memprediksi data dalam file yang diunggah.

4.6.3. Black-box Testing

Pengujian sistem dilakukan menggunakan metode Black-Box Testing, yaitu pengujian yang difokuskan pada fungsionalitas sistem dari sisi pengguna, tanpa mengetahui struktur internal kode program. Tujuan dari pengujian ini adalah untuk memastikan bahwa setiap fitur utama dalam aplikasi berfungsi

sebagaimana mestinya dan memberikan keluaran sesuai dengan yang diharapkan. Pengujian dilakukan dengan mencoba berbagai skenario interaksi pengguna, baik melalui input manual maupun upload file CSV, serta pengujian terhadap validasi dan navigasi antarmuka. Hasil dari pengujian dirangkum dalam tabel berikut:

Tabel 4.6 Pengujian Black-box Testing

No.	Fitur yang Diuji	Langkah Uji	Hasil yang Diharapkan	Status
1.	Halaman Input Data Manual	Pengguna mengisi form data medis pasien dan menekan tombol	Hasil klasifikasi dan nilai probabilitas tampil dengan benar	Berhasil

		“Prediksi”		
2.	Upload File CSV	Pengguna mengunggah file .csv berisi data pasien	Data terbaca dan hasil klasifikasi untuk tiap pasien muncul	Berhasil
3.	Validasi Format CSV	Pengguna mengunggah file CSV dengan format yang salah/rusak	Sistem menampilkan pesan kesalahan/hasil tidak terbaca (None)	Berhasil
4.	Output Hasil	Sistem menerima input (manual	Output prediksi ditampilkan jelas	Berhasil

	Prediksi	/CSV) yang valid	dengan label diabetes /tidak diabetes	
5.	Navigasi Antarmuka	Pengguna menekan tombol navigasi atau submit	Sistem berpindah ke tampilan /halaman sesuai alur	Berhasil

Hasil pengujian menunjukkan bahwa seluruh fitur utama pada sistem telah berjalan dengan baik sesuai spesifikasi. Setiap skenario uji menghasilkan keluaran yang sesuai, tidak ditemukan error atau bug kritis selama proses pengujian berlangsung

B. Implementasi Model ke Web

Sistem klasifikasi penyakit diabetes berbasis web ini diimplementasikan ke dalam platform berbasis web menggunakan framework streamlit.

1. *Tools* dan Teknologi

   Beberapa *tools* yang digunakan dalam pembangunan sistem ini meliputi:

   a. Python

      Bahasa pemrograman utama yang digunakan dalam pengembangan sistem, baik untuk pemodelan maupun implementasi web.

   b. Streamlit

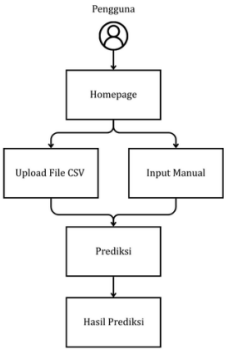
      Framework berbasis Python yang digunakan untuk membangun antarmuka web interaktif. Streamlit memudahkan proses integrasi model ke dalam aplikasi yang dapat diakses pengguna melalui browser.

   c. Joblib

      Digunakan untuk memuat model Random Forest yang telah dilatih, sehingga dapat digunakan langsung dalam sistem tanpa perlu melakukan pelatihan ulang.

2. Alur Interaksi Pengguna *(User Flow)*

   Pada tahap ini menjelaskan alur interaksi pengguna saat menggunakan sistem. Alur ini menggambarkan tahapan-tahapan utama dari awal pengguna membuka aplikasi hingga menerima hasil prediksi.



Gambar 4.15 *User Flow*

a. Pengguna mengakses halaman utama dari web.
b. Sistem menampilkan form input data pasien.
c. Setelah semua data atau file yang dibutuhkan telah diisi, pengguna kemudian menekan tombol “prediksi”.
d. Sistem akan memproses input menggunakan model random forest yang telah dilatih sebelumnya.
e. Hasil prediksi akan ditampilkan dalam bentuk teks yang menyatakan pasien menderita diabetes atau tidak diabetes.

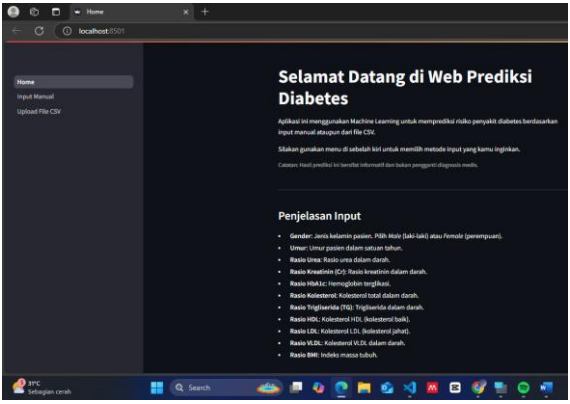
3. Antarmuka Pengguna *(User Interface)*

Untuk memudahkan interaksi, sistem dikembangkan dalam bentuk aplikasi web dengan tampilan antarmuka yang sederhana, intuitif, dan

mudah digunakan. Antarmuka ini dirancang agar pengguna dapat memahami cara kerja sistem serta melakukan proses klasifikasi secara mandiri, baik melalui input data secara manual maupun dengan mengunggah file CSV. Aplikasi web ini dapat diakses melalui tautan berikut: https://diabet-detection.streamlit.app/

a. Halaman Utama

Halaman utama merupakan tampilan awal aplikasi yang menyajikan informasi umum mengenai fungsi sistem serta petunjuk penggunaan. Tujuan dari halaman ini adalah memberikan orientasi awal bagi pengguna sebelum memulai proses deteksi. Tampilan antarmuka halaman ini dapat dilihat pada Gambar 4.16 berikut:



Gambar 4.16 Tampilan Halaman Utama

Selain informasi umum, halaman ini juga menjelaskan arti dari setiap fitur input yang digunakan dalam deteksi, yaitu:

- Gender: Jenis kelamin pasien (Male = laki-laki, Female = perempuan).
- Umur: Umur pasien dalam satuan tahun.
- Rasio Urea: Kandungan urea dalam darah.
- Rasio Kreatinin (Cr): Kadar kreatinin dalam darah.
- Rasio HbA1c: Persentase hemoglobin terglikasi.
- Rasio Kolesterol: Jumlah kolesterol total dalam darah.
- Rasio Trigliserida (TG): Kadar trigliserida dalam darah.
- Rasio HDL: Kadar kolesterol HDL (*kolesterol baik*).
- Rasio LDL: Kadar kolesterol LDL (*kolesterol jahat*).
- Rasio VLDL: Kadar kolesterol VLDL dalam darah.
- Rasio BMI: Indeks massa tubuh.

Lebih lanjut, halaman utama ini juga berfungsi sebagai dashboard analisis yang menampilkan

hasil deteksi dari model dan memberikan insight mendalam mengenai atribut-atribut yang berisiko tinggi terhadap diabetes. Bagian ini terdiri dari:

a) Tabel Hasil Deteksi Diabetes dari Model

Bagian ini menyajikan data tabular dari hasil deteksi yang dilakukan oleh model ML.

	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI	y_pred
0	0	66	5.5	56	9.9	4.4	2	1	2.5	0.9	29	1
1	0	33	2.7	47	5	4.2	1.4	1.3	2.6	0.7	24	0
2	1	68	7.1	79	9.4	3.7	3.7	0.8	1.4	1.6	31	1
3	0	58	2.7	53	7.3	4.8	3.3	0.9	2.6	1.5	30	1
4	1	54	5.4	68	3	1.2	0.7	1	0.5	1.5	33	1
5	0	57	4.3	41	10.2	5.5	1.1	1.4	1.7	2	31	1
6	1	61	0.5	59	7.5	4.5	1.8	1	2.7	0.8	38	1
7	1	55	2.6	48	11.8	5.1	1.2	1	3.6	1.6	30	1
8	1	51	10.4	76	7.3	3.3	2	1	1.2	0.9	33	1
9	1	61	4.1	44	10.3	6.4	11.6	1.1	0.7	5.2	31	1

Gambar 4.17 Tampilan Halaman Home Tabel Hasil Deteksi dari Model

Pengguna dapat melihat secara langsung data input beserta hasil deteksi (0 untuk non-diabetes, 1 untuk diabetes) untuk setiap baris data yang diproses. Tabel ini dirancang agar responsif dan mudah dipahami.

b) Analisis Atribut Berisiko Tinggi Terhadap Diabetes

Gambar 4.18 Tampilan Halaman Home Analisis Atribut Berisiko Tinggi

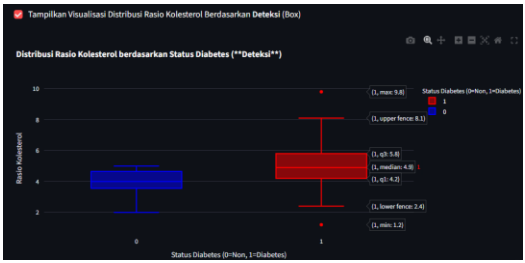
Bagian ini menyediakann analisis ringkas namun informatif untuk setiap atribut input. Tujuannya adalah untuk mengidentifikasi pola atau tren yang menunjukkan keterkaitan antara nilai atribut dengan status diabetes yang dideteksi.

- Untuk atribut kategorikal seperti gender: Sistem menghitung persentase individu yang dideteksi diabetes untuk setiap kategori. Pada gambar 4.18 diketahui bahwa dari data yang dideteksi, laki-laki menunjukkan persentase kasus diabetes (deteksi) yang paling tinggi,

yaitu 89.29% dari total individu dengan gender tersebut dalam dataset ini.

- Untuk atribut numerik seperti Umur, HbA1c, Urea, Cr, Kolesterol, Trigliserida, HDL, LDL, VLDL, BMI: Sistem menghitung rata-rata nilai atribut untuk dua kelompok, yaitu individu yang dideteksi diabetes dan individu yang dideteksi non-diabetes. Perbandingan rata-rata ini digunakan untuk menyimpulkan apakah nilai atribut yang lebih tinggi atau lebih rendah cenderung berkorelasi dengan status diabetes.

c) Visualisasi Distribusi Atribut



Gambar 4.19 Tampilan Halaman Utama Distribusi Atribut

Untuk mendukung pemahaman visual, halaman ini juga menyediakan opsi untuk menampilkan grafik interaktif. Pengguna dapat memilih untuk melihat:

- Box Plot: Grafik ini menampilkan distribusi statistik (median, kuartil, rentang, dan outlier) dari atribut numerik untuk kedua kelompok (dideteksi diabetes dan non-diabetes). Ini sangat membantu untuk membandingkan sebaran data antara kedua kelompok.
- Histogram: Grafik ini menunjukkan frekuensi atau kepadatan distribusi nilai-nilai atribut numerik. Dengan menumpuk histogram untuk kelompok diabetes dan non-diabetes,

pengguna dapat melihat perbedaan pola distribusi usia atau rasio lainnya.

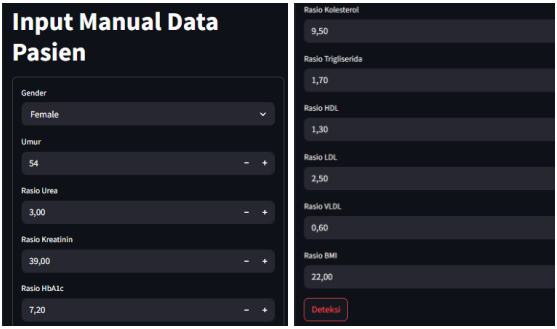
- Visualisasi ini dibuat menggunakan Plotly Express, sehingga bersifat interaktif (pengguna dapat zoom, pan, dan melihat detail data saat mengarahkan kursor).

b. Halaman Input Manual

Pada halaman ini, pengguna dapat mengisi formulir data medis pasien secara manual. Data yang diisikan meliputi jenis kelamin, usia, rasio urea, rasio kreatinin, HbA1c, kadar kolesterol, trigliserida, HDL, VLDL, dan BMI. Berikut ini merupakan contoh input manual data pasien untuk klasifikasi diabetes, seperti ditampilkan pada gambar 4.20 Data yang dimasukkan adalah sebagai berikut:

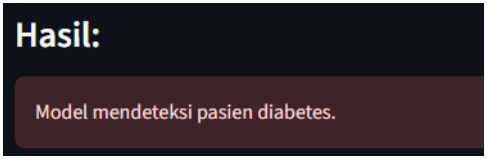
- Gender: Female
- Umur: 54
- Rasio Urea: 3,00
- Rasio Kreatinin: 39,00
- Rasio HbA1c: 7,20
- Rasio Kolesterol: 9,50
- Rasio Trigliserida: 1,70
- Rasio HDL: 1,30

- Rasio LDL: 2,50
- Rasio VLDL: 0,60
- Rasio BMI: 22,00



Gambar 4.20 Tampilan Halaman Input Manual

Setelah pengguna menekan tombol "Deteksi", sistem akan menampilkan hasil klasifikasi yaitu apakah terdeteksi positif atau negatif diabetes.

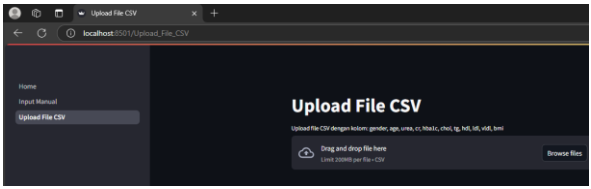


Gambar 4.21 Hasil Deteksi Input Manual

Pada contoh hasil prediksi pada gambar 4.21, sistem mengidentifikasi bahwa pasien terdeteksi menderita diabetes.

c. Halaman Upload CSV

Selain input manual, pengguna juga dapat mengunggah file berformat .csv yang berisi data beberapa pasien sekaligus.



Gambar 4.22 Tampilan Halaman Upload CSV

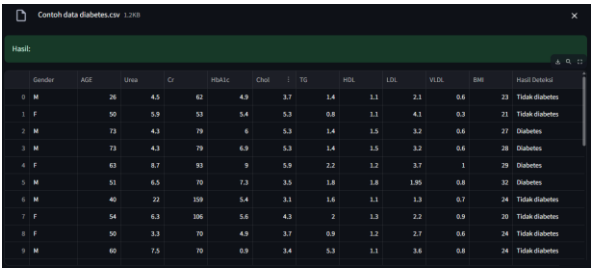
Berikut ini ditampilkan contoh file .csv yang memuat data input dari beberapa pasien dengan indikasi diabetes. Isi dari file tersebut ditampilkan dalam bentuk tabel 4.7 seperti pada ilustrasi berikut:

Tabel 4.7 Contoh Inputan dengan File .CSV

Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI
M	26	4.5	62	4.9	3.7	1.4	1.1	2.1	0.6	23
F	50	5.9	53	5.4	5.3	0.8	1.1	4.1	0.3	21
M	73	4.3	79	6	5.3	1.4	1.5	3.2	0.6	27
M	73	4.3	79	6.9	5.3	1.4	1.5	3.2	0.6	28
F	63	8.7	93	9	5.9	2.2	1.2	3.7	1	29
M	51	6.5	70	7.3	3.5	1.8	1.8	1.95	0.8	32
M	40	22	159	5.4	3.1	1.6	1.1	1.3	0.7	24
F	54	6.3	106	5.6	4.3	2	1.3	2.2	0.9	20
F	50	3.3	70	4.9	3.7	0.9	1.2	2.7	0.6	24
M	60	7.5	70	0.9	3.4	5.3	1.1	3.6	0.8	24
M	77	5	106	5.4	0	2.8	0.8	1.8	0.7	19
F	59	4.7	58	4.1	4.5	1.8	1.8	1.8	1.3	22.5
F	57	4.6	97	0.9	3.2	1.3	0.9	3	1.1	22

Setelah file diunggah, sistem akan memproses data dan menampilkan hasil klasifikasi untuk masing-masing pasien. Selain itu, pengguna

juga dapat mengunduh hasil prediksi dalam format file.



	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI	Hasil Deteksi
0	M	26	4.5	62	4.9	3.7	1.4	1.1	2.1	0.6	23	Tidak diabetes
1	F	50	5.9	53	5.4	5.3	0.8	1.1	4.1	0.3	21	Tidak diabetes
2	M	73	4.3	79	6	5.3	1.4	1.5	3.2	0.6	27	Diabetes
3	M	73	4.3	79	6.9	5.3	1.4	1.5	3.2	0.6	28	Diabetes
4	F	63	8.7	93	9	5.9	2.2	1.2	3.7	1	29	Diabetes
5	M	51	6.5	70	7.3	3.5	1.8	1.8	1.95	0.8	32	Diabetes
6	M	40	22	159	5.4	3.1	1.6	1.1	1.3	0.7	24	Tidak diabetes
7	F	54	6.3	106	5.6	4.3	2	1.3	2.2	0.9	20	Tidak diabetes
8	F	50	3.3	70	4.9	3.7	0.9	1.2	2.7	0.6	24	Tidak diabetes
9	M	60	7.5	70	0.9	3.4	5.3	1.1	3.6	0.8	24	Tidak diabetes

Gambar 4.23 Hasil Deteksi Upload File

C. Pembahasan

Model deteksi penyakit diabetes yang dibangun dalam penelitian ini menggunakan algoritma Random Forest dan berhasil menunjukkan performa yang sangat baik berdasarkan hasil evaluasi. Keputusan untuk menggunakan algoritma ini didasarkan pada kemampuannya dalam menangani data berdimensi tinggi, mencegah *overfitting* yaitu kondisi ketika model terlalu menyesuaikan diri dengan data latih sehingga kehilangan kemampuan untuk memprediksi data baru secara akurat, serta memberikan hasil deteksi yang stabil serta akurat. Dalam penelitian ini, algoritma Random Forest terbukti mampu mendeteksikan data pasien secara efisien dan efektif, baik dari sisi ketepatan maupun keseimbangan antara kelas.

Berdasarkan hasil evaluasi menggunakan *confusion matrix*, diperoleh hasil sebagai berikut:

- Sebanyak 135 data positif berhasil dideteksi dengan benar sebagai positif ***(True Positive)***.
- Sebanyak 20 data negatif berhasil dideteksi dengan benar sebagai negatif ***(True Negative)***.
- Terdapat 3 data negatif yang salah dideteksi sebagai positif ***(False Positive)***.
- Terdapat 1 data positif yang salah dideteksi sebagai negatif ***(False Negative)***.

Hasil ini menunjukkan bahwa model memiliki tingkat ketepatan yang sangat tinggi dalam mendeteksi kasus diabetes. Jumlah kesalahan klasifikasi khususnya *false negative* sangat kecil, yang merupakan hal penting dalam konteks medis untuk menghindari risiko keterlambatan diagnosis.

Selanjutnya, model dievaluasi menggunakan berbagai metrik performa lainnya. Berdasarkan hasil pengujian, hasil metrik diperoleh sebagai berikut:

- Akurasi model mencapai 97,48%, menunjukkan bahwa sebagian besar deteksi yang dilakukan model adalah benar.
- *Precision* sebesar 97,83%, yang berarti sebagian besar deteksi pasien positif diabetes memang benar-benar positif.

- *Recall* sebesar 99,26%, menunjukkan bahwa hampir seluruh pasien yang benar-benar mengidap diabetes berhasil dikenali oleh model.
- F1-Score sebesar 98,54%, menandakan keseimbangan yang sangat baik antara *precision* dan *recall*.
- ROC AUC sebesar 93,11%, menegaskan kemampuan model dalam membedakan antara kelas positif dan negatif secara efektif.

Proses pelatihan dan pencarian kombinasi hyperparameter terbaik menggunakan teknik GridSearchCV membutuhkan waktu eksekusi sekitar 752 detik. Secara keseluruhan dari 164 data uji, model berhasil mengklasifikasikan 160 data dengan benar, hanya mengalami empat kesalahan prediksi.

Pada tahap evaluasi performa model, dilakukan pengujian menggunakan kurva *Receiver Operating Characteristic* (ROC) dan penghitungan *Area Under Curve* (AUC) terhadap algoritma Random Forest yang telah dibangun. Hasilnya menunjukkan nilai AUC sebesar 1.00 pada data training dan 1.00 pada data testing. Nilai AUC 1.00 menunjukkan bahwa model memiliki kemampuan deteksi yang sempurna dalam membedakan antara kelas positif dan negatif, baik pada data training maupun testing. Kurva ROC yang dihasilkan juga mendekati titik

(0,1) pada grafik, yang menandakan tingkat *True Positive Rate* yang tinggi dengan *False Positive Rate* yang sangat rendah.

Konsistensi nilai AUC yang tinggi di kedua dataset ini mengindikasikan bahwa model tidak mengalami *overfitting* karena metrik lain seperti F1-Score dan hasil pada data uji tetap seimbang. Dengan demikian, dapat disimpulkan bahwa algoritma Random Forest yang diterapkan mampu membangun model deteksi yang akurat, stabil, dan mampu menggeneralisasi dengan baik terhadap dataset yang digunakan.

Tingginya performa model ini juga dipengaruhi oleh proses *preprocessing* data, khususnya dalam menangani masalah data tidak seimbang *(imbalanced data).* Pada dataset awal, jumlah pasien yang terdiagnosis diabetes lebih banyak dibandingkan dengan pasien non-diabetes, yang dapat menyebabkan bias pada model terhadap kelas mayoritas. Untuk mengatasi hal ini, digunakan teknik SMOTE *(Synthetic Minority Over-sampling Technique),* yang berhasil menyeimbangkan distribusi data latih sehingga model dapat mengenali kedua kelas secara lebih proporsional.

Nilai *recall* yang sangat tinggi semakin memperkuat pentingnya penggunaan teknik *balancing*, mengingat dalam konteks medis, mengurangi *false negative* sangat

krusial untuk mencegah risiko keterlambatan diagnosis. Selain itu, tingginya *precision* juga menunjukkan bahwa model dapat meminimalkan prediksi positif palsu, sehingga mengurangi kecemasan atau kesalahan tindak lanjut pada pasien.

Model ini diimplementasikan ke dalam aplikasi berbasis web menggunakan framework Streamlit. Platform ini dipilih karena kemudahan penggunaannya dan kemampuannya menyajikan model machine learning secara interaktif. Aplikasi menyediakan dua metode input, yaitu input data manual dan unggah file CSV, dengan hasil deteksi yang ditampilkan secara real-time. Antarmuka yang sederhana dan ramah pengguna memungkinkan sistem ini diakses oleh tenaga medis maupun masyarakat umum, menjadikannya alat yang tidak hanya fungsional secara teknis tetapi juga bermanfaat sebagai sarana edukatif dan pendukung skrining awal terhadap risiko diabetes.

Untuk memastikan bahwa sistem berjalan sesuai dengan fungsionalitas yang dirancang, dilakukan pengujian menggunakan metode *black-box testing*. Pengujian ini difokuskan pada bagaimana sistem merespons input dari sisi pengguna tanpa memperhatikan struktur internal kodenya. Seluruh fitur utama seperti input data manual, unggah file CSV, validasi input kosong,

hingga tampilan hasil deteksi diuji secara menyeluruh. Hasil pengujian menunjukkan bahwa sistem mampu menjalankan seluruh fungsi dengan baik dan memberikan output yang sesuai, sehingga sistem dinyatakan layak digunakan secara fungsional.

BAB V

KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan hasil penelitian dan pembahasan mengenai deteksi penyakit diabetes menggunakan algoritma Random Forest melalui platform web, maka diperoleh beberapa kesimpulan sebagai berikut:

1. Penerapan algoritma Random Forest untuk deteksi penyakit diabetes telah berhasil diimplementasikan ke dalam platform web berbasis Streamlit**.** Proses implementasi diawali dengan tahapan *preprocessing* data yang mencakup pembersihan data, encoding variabel kategorikal, normalisasi, dan penyeimbangan kelas menggunakan teknik SMOTE. Setelah proses pelatihan model dilakukan, sistem deteksi terintegrasi ke dalam antarmuka web yang menyediakan dua metode input, yaitu input manual dan upload file CSV. Platform ini memungkinkan pengguna untuk memperoleh hasil prediksi secara instan berdasarkan data medis yang diinputkan.
2. Performa algoritma Random Forest dalam mendeteksi penyakit diabetes menunjukkan hasil yang sangat baik. Berdasarkan hasil evaluasi

menggunakan data uji, model memperoleh metrik performa sebagai berikut:

a. Akurasi sebesar 97,47%
b. Presisi sebesar 97,83%
c. Recall sebesar 99,26%
d. F1-Score sebesar 98,54%
e. AUC-ROC sebesar 93,11%

Hasil ini membuktikan bahwa model memiliki kemampuan deteksi yang sangat akurat dan andal dalam membedakan antara pasien yang terkena diabetes dan tidak terkena diabetes. Selain itu, hasil pengujian menggunakan metode *black box testing* juga menunjukkan bahwa semua fungsionalitas sistem berjalan dengan baik, dengan Tingkat keberhasilan mencapai 100% sesuai dengan yang diharapkan.

B. Saran

Adapun saran berdasarkan hasil penelitian ini adalah:

1. Pengembangan Sistem Lebih Lanjut

   Diharapkan sistem ini dapat dikembangkan dengan menambahkan fitur tambahan seperti visualisasi riwayat kesehatan pasien, prediksi risiko berdasarkan tren data, atau integrasi dengan rekam medis elektronik (EMR).
2. Validasi Menggunakan Dataset Lebih besar dan Variatif

Untuk meningkatkan generalisasi model, perlu dilakukan pengujian lebih lanjut menggunakan dataset yang lebih besar dan beragam, serta berasal dari berbagai wilayah atau institusi kesehatan.

3. Penerapan Dalam Lingkungan Medis Riil

   Diperlukan kerja sama dengan tenaga medis dan instansi kesehatan agar sistem ini dapat dimanfaatkan secara langsung sebagai alat bantu diagnosis awal diabetes, dengan tetap memperhatikan aspek privasi dan etika data pasien.

DAFTAR PUSTAKA

Amelia, R. (2018). Hubungan Perilaku Perawatan Kaki dengan Terjadinya Komplikasi Luka Kaki Diabetes pada Pasien Diabetes Melitus Tipe 2 di Puskesmas Tuntungan Kota Medan. *Talenta Conference Series: Tropical Medicine (TM)*, *1*(1), 124–131. https://doi.org/10.32734/tm.v1i1.56

Andini, A. T., & Yahfizham. (2024). Analisis Algoritma Pemrograman Dalam Media Sosial Terhadap Pola Konsumsi Konten. *Jurnal Arjuna : Publikasi Ilmu Pendidikan, Bahasa Dan Matematika*, *2*, 286–296.

Apriliah, W., Kurniawan, I., Baydhowi, M., & Haryati, T. (2021). Prediksi Kemungkinan Diabetes pada Tahap Awal Menggunakan Algoritma Klasifikasi Random Forest. *SISTEMASI*, *10*(1), 163. https://doi.org/10.32520/stmsi.v10i1.1129

Aqil Musthafa Ar Rachman, M., Aulia Hanifah, N., Farah Fakhirah, S., Halya Alfrida, M., Shafa Salsabila, N., Wicaksono, A., & Parasti Mindara, G. (2024). PENERAPAN BLACK BOX TESTING UNTUK EVALUASI FUNGSIONALITAS WEBSITE MAGGOPLAST. *JATI (Jurnal Mahasiswa Teknik Informatika)*, *9*(1), 169–176. https://doi.org/10.36040/jati.v9i1.12177

Attaran, M., & Deb, P. (2018). Machine Learning: The New “Big Thing” for Competitive Advantage. *International Journal of Knowledge Engineering and Data Mining*, *5*(1), 1. https://doi.org/10.1504/IJKEDM.2018.10015621

Bintoro, P., Ratnasari, Wihardjo, E., Putri, I. P., & Asari, A. (2024). *Pengantar Machine Learning* (A. Asari, Ed.). PT MAFY MEDIA LITERASI INDONESIA. https://repository.um.ac.id/5619/1/fullteks.pdf

Dhani, A. A., Siswa, T. A. Y., & Pranoto, W. J. (2024). Perbaikan Akurasi Random Forest Dengan ANOVA Dan SMOTE Pada Klasifikasi Data Stunting. *Teknika*, *13*(2), 264–272. https://doi.org/10.34148/teknika.v13i2.875

Finda, S. M., & Utomo, D. W. (2024). Klasifikasi Stunting Balita menggunakan Metode Ensemble Learning dan Random Forest. *INFOTEKMESIN (Media Informasi Untuk Pengembangan Penelitian Teknik)*, *15*(02), 287–295. https://doi.org/10.35970/infotekmesin.v15i2.2326

Gde Agung Brahmana Suryanegara, Adiwijaya, & Mahendra Dwifebri Purbolaksono. (2021a). Peningkatan Hasil Klasifikasi pada Algoritma Random Forest untuk Deteksi Pasien Penderita Diabetes Menggunakan Metode Normalisasi. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *5*(1), 114–122. https://doi.org/10.29207/resti.v5i1.2880

Gde Agung Brahmana Suryanegara, Adiwijaya, & Mahendra Dwifebri Purbolaksono. (2021b). Peningkatan Hasil Klasifikasi pada Algoritma Random Forest untuk Deteksi Pasien Penderita Diabetes Menggunakan Metode Normalisasi. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *5*(1), 114–122. https://doi.org/10.29207/resti.v5i1.2880

Gefy Fitry Wijaya, & Dwi Yuniarto. (2024). Tinjauan Penerapan Machine Learning pada Sistem Rekomendasi Menggunakan Model Klasifikasi. *Populer: Jurnal Penelitian Mahasiswa*, *3*(4), 144–153. https://doi.org/10.58192/populer.v3i4.2798

Hakim, I., Asdi, A., Lubis, Mhd. D. S., Harahap, M. N., & Bara, L. R. B. (2025). Klasifikasi Kualitas Produk Mesin Pertanian Berdasarkan Evaluasi Kinerja Algoritma Random Forest.

KESATRIA: Jurnal Penerapan Sistem Informasi (Komputer & Manajemen), *6*(1), 343–356. https://tunasbangsa.ac.id/pkm/index.php/kesatria/article/download/577/572

Hardianto, D. (2021). TELAAH KOMPREHENSIF DIABETES MELITUS: KLASIFIKASI, GEJALA, DIAGNOSIS, PENCEGAHAN, DAN PENGOBATAN. *Jurnal Bioteknologi & Biosains Indonesia (JBBI)*, *7*(2), 304–317. https://doi.org/10.29122/jbbi.v7i2.4209

Ibrahim, F. (2024). *PERBANDINGAN PERFORMA ALGORITMA RANDOM FOREST CLASSIFIER DAN NAIVE BAYES PADA PENYAKIT DIABETES MELITUS*. UIN Syarif Hidayatullah.

Iskandar, R. F. N., Gutama, D. H., Wijaya, D. P., & Danianti, D. (2024a). Klasifikasi Menggunakan Metode Random Forest untuk Awal Deteksi Diabetes Melitus Tipe 2. *Jurnal Teknik Industri Terintegrasi*, *7*(3), 1620–1626. https://doi.org/10.31004/jutin.v7i3.26916

Iskandar, R. F. N., Gutama, D. H., Wijaya, D. P., & Danianti, D. (2024b). Klasifikasi Menggunakan Metode Random Forest untuk Awal Deteksi Diabetes Melitus Tipe 2. *Jurnal Teknik Industri Terintegrasi*, *7*(3), 1620–1626. https://doi.org/10.31004/jutin.v7i3.26916

Joseph, V. R. (2022). Optimal ratio for data splitting. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, *15*(4), 531–538. https://doi.org/10.1002/sam.11583

Mandal, A., & Pradhan, S. (2023). Predict2Protect: Machine Learning Web Application in Early Detection of Heart Disease. *Cureus*. https://doi.org/10.7759/cureus.49305

Mirafuddin, M., & Supriyatna, S. (2024). Penerapan Machine Learning pada Aplikasi Kesehatan HEALTIME. *Jurnal*

Nasional Komputasi Dan Teknologi Informasi (JNKTI), *7*(6), 1601–1606. https://doi.org/10.32672/jnkti.v7i6.8233

MIT Critical Data. (2016). *Secondary Analysis of Electronic Health Records*. Springer International Publishing. https://doi.org/10.1007/978-3-319-43742-2

Mubaqarah, Puteri, A. N., & Sumardin, A. (2024). Comparison of Random Forest and XGBoost for Diabetes Classification with SHAP and LIME Interpretation. *JTERA (Jurnal Teknologi Rekayasa)*, *9*, 121–130. https://doi.org/10.31544/jtera.v9.i1.2024.121-130

Musat, F. (2015). The anaerobic degradation of gaseous, nonmethane alkanes — From in situ processes to microorganisms. *Computational and Structural Biotechnology Journal*, *13*, 222–228. https://doi.org/10.1016/j.csbj.2015.03.002

Muzayanah, R., Pertiwi, D. A. A., Ali, M., & Muslim, M. A. (2024). Comparison of gridsearchcv and bayesian hyperparameter optimization in random forest algorithm for diabetes prediction. *Journal of Soft Computing Exploration*, *5*(1), 1–6.

Nugroho, K., Noersasongko, E., Purwanto, Muljono, Fanani, A. Z., Affandy, & Basuki, R. S. (2019). Improving Random Forest Method to Detect Hatespeech and Offensive Word. *2019 International Conference on Information and Communications Technology (ICOIACT)*, 514–518. https://doi.org/10.1109/ICOIACT46704.2019.8938451

Punthakee, Z., Goldenberg, R., & Katz, P. (2018). Definition, Classification and Diagnosis of Diabetes, Prediabetes and

Metabolic Syndrome. *Canadian Journal of Diabetes*, *42*, S10–S15. https://doi.org/10.1016/j.jcjd.2017.10.003

Rashid, A. (2020). Diabetes Dataset. *Mendeley Data*.

Sari, L., Romadloni, A., & Listyaningrum, R. (2023). Penerapan Data Mining dalam Analisis Prediksi Kanker Paru Menggunakan Algoritma Random Forest. *Infotekmesin*, *14*(1), 155–162. https://doi.org/10.35970/infotekmesin.v14i1.1751

Setiawan, A., Nst, Z. H., Khairi, Z., Rahmaddeni, & Efrizoni, L. (2024). KLASIFIKASI TINGKAT RISIKO DIABETES MENGGUNAKAN ALGORITMA RANDOM FOREST. *Jurnal Informatika & Rekayasa Elektronika)*, *7*(2), 263–271. http://e-journal.stmiklombok.ac.id/index.php/jireISSN.2620-6900

Siallagan, R. A., & Fitriyani. (2021). PREDIKSI PENYAKIT DIABETES MELLITUS MENGGUNAKAN ALGORITMA C4.5. *Jurnal Responsif : Riset Sains Dan Informatika*, *3*(1), 44–52. https://doi.org/10.51977/jti.v3i1.407

Sidiq Wijaya, F., Farhan Arotsid, M., Adriano, R., & Larasati, Z. D. (2024). Literatur Review: Penerapan Algoritma Random Forest untuk Klasifikasi Penyakit Diabetes. *BIIKMA : Buletin Ilmiah Ilmu Komputer Dan Multimedia*, *2*, 474–481. https://jurnalmahasiswa.com/index.php/biikma

Sir, Y. A., & Soepranoto, A. H. H. (2022). Pendekatan Resampling Data Untuk Menangani Masalah Ketidakseimbangan Kelas. *Jurnal Komputer Dan Informatika*, *10*(1), 31–38. https://doi.org/10.35508/jicon.v10i1.6554

Sodikin, M. I. (2023). Penerapan dan Manfaat Machine Learning di Rumah Sakit. *Multiverse: Open*

Multidisciplinary Journal, *2*(2), 262–265. https://doi.org/10.57251/multiverse.v2i2.1207

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, *45*(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

Syafa Iswahyudi, S. N., & Eka Putra, R. (2025). Sistem Deteksi Stunting pada Balita Berbasis Web Menggunakan Metode Random Forest. *Journal of Informatics and Computer Science (JINACS)*, *6*(03), 755–764. https://doi.org/10.26740/jinacs.v6n03.p755-764

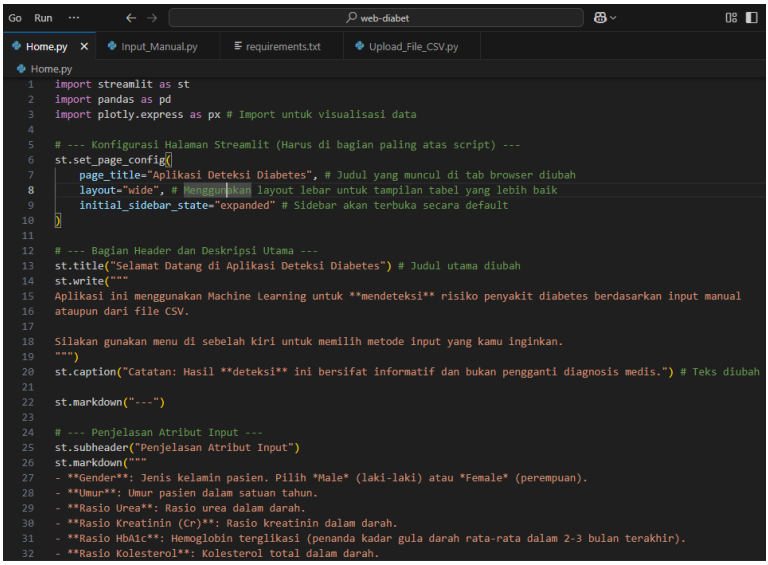
Van Sluyters, R. C. (1997). Introduction to the Internet and World Wide Web. *ILAR Journal*, *38*(4), 162–167. https://doi.org/10.1093/ilar.38.4.162

Wei, M.-F., Luh, Y.-H., Huang, Y.-H., & Chang, Y.-C. (2021). Young Generation's Mobile Payment Adoption Behavior: Analysis Based on an Extended UTAUT Model. *Journal of Theoretical and Applied Electronic Commerce Research*, *16*(1), 1–20. https://doi.org/10.3390/jtaer16010001

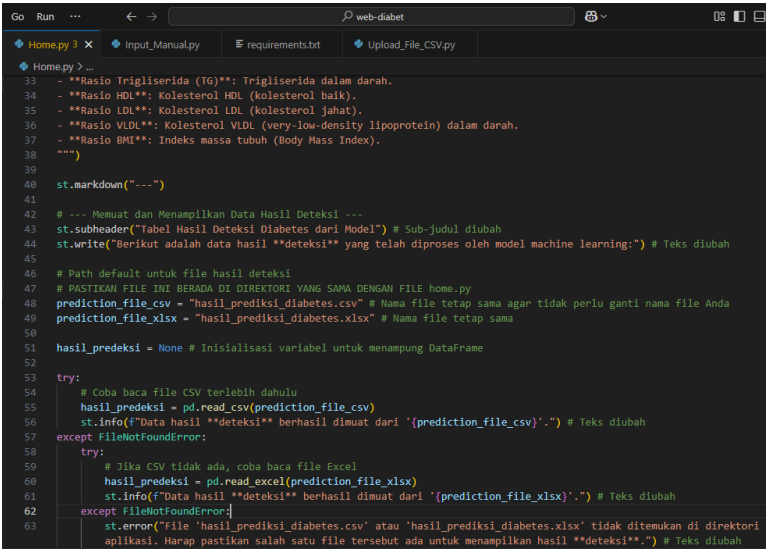
Yoga Religia, Agung Nugroho, & Wahyu Hadikristanto. (2021). Klasifikasi Analisis Perbandingan Algoritma Optimasi pada Random Forest untuk Klasifikasi Data Bank Marketing. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *5*(1), 187–192. https://doi.org/10.29207/resti.v5i1.2813

LAMPIRAN

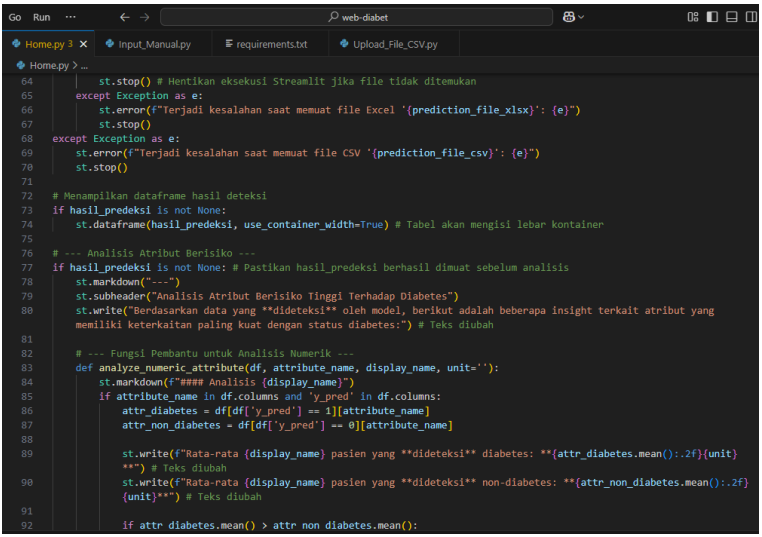
Lampiran 1: Codingan Halaman Home.py



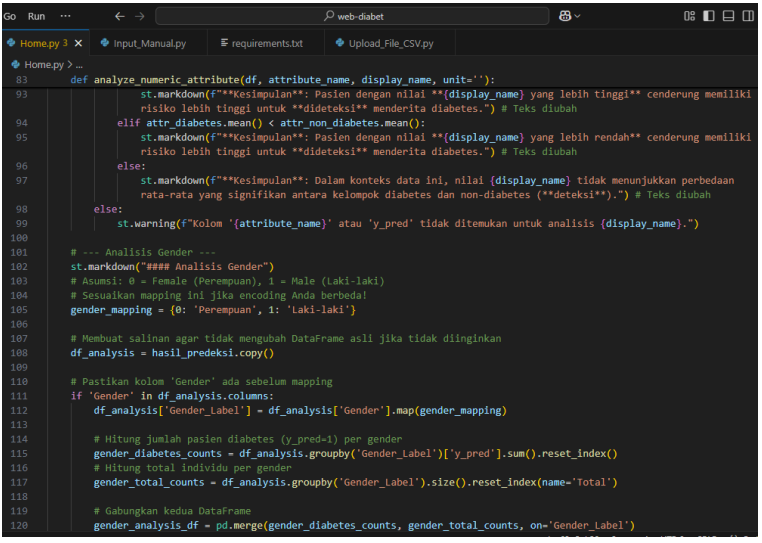
```python
import streamlit as st
import pandas as pd
import plotly.express as px # Import untuk visualisasi data

# --- Konfigurasi Halaman Streamlit (Harus di bagian paling atas script) ---
st.set_page_config(
    page_title="Aplikasi Deteksi Diabetes", # Judul yang muncul di tab browser diubah
    layout="wide", # Menggunakan layout lebar untuk tampilan tabel yang lebih baik
    initial_sidebar_state="expanded" # Sidebar akan terbuka secara default
)

# --- Bagian Header dan Deskripsi Utama ---
st.title("Selamat Datang di Aplikasi Deteksi Diabetes") # Judul utama diubah
st.write("""
Aplikasi ini menggunakan Machine Learning untuk **mendeteksi** risiko penyakit diabetes berdasarkan input manual
ataupun dari file CSV.

Silakan gunakan menu di sebelah kiri untuk memilih metode input yang kamu inginkan.
""")
st.caption("Catatan: Hasil **deteksi** ini bersifat informatif dan bukan pengganti diagnosis medis.") # Teks diubah

st.markdown("---")

# --- Penjelasan Atribut Input ---
st.subheader("Penjelasan Atribut Input")
st.markdown("""
- **Gender**: Jenis kelamin pasien. Pilih *Male* (laki-laki) atau *Female* (perempuan).
- **Umur**: Umur pasien dalam satuan tahun.
- **Rasio Urea**: Rasio urea dalam darah.
- **Rasio Kreatinin (Cr)**: Rasio kreatinin dalam darah.
- **Rasio HbA1c**: Hemoglobin terglikasi (penanda kadar gula darah rata-rata dalam 2-3 bulan terakhir).
- **Rasio Kolesterol**: Kolesterol total dalam darah.
```



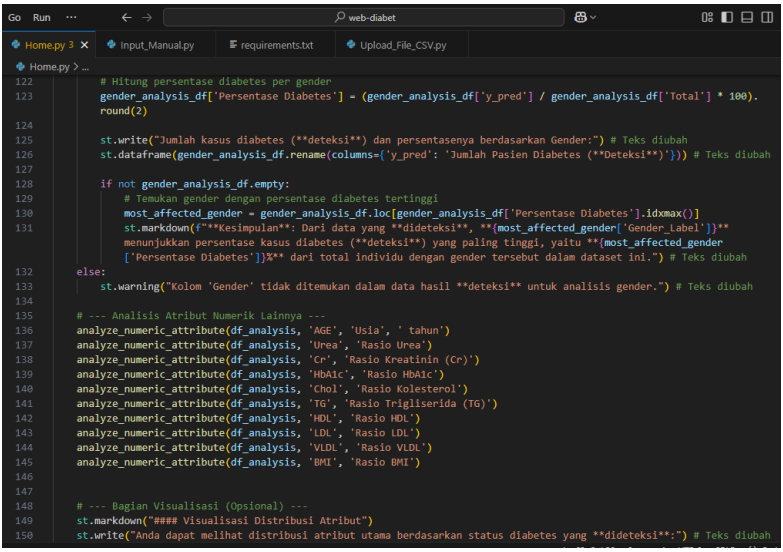
```python
- **Rasio Trigliserida (TG)**: Trigliserida dalam darah.
- **Rasio HDL**: Kolesterol HDL (kolesterol baik).
- **Rasio LDL**: Kolesterol LDL (kolesterol jahat).
- **Rasio VLDL**: Kolesterol VLDL (very-low-density lipoprotein) dalam darah.
- **Rasio BMI**: Indeks massa tubuh (Body Mass Index).
""")

st.markdown("---")

# --- Memuat dan Menampilkan Data Hasil Deteksi ---
st.subheader("Tabel Hasil Deteksi Diabetes dari Model") # Sub-judul diubah
st.write("Berikut adalah data hasil **deteksi** yang telah diproses oleh model machine learning:") # Teks diubah

# Path default untuk file hasil deteksi
# PASTIKAN FILE INI BERADA DI DIREKTORI YANG SAMA DENGAN FILE home.py
prediction_file_csv = "hasil_prediksi_diabetes.csv" # Nama file tetap sama agar tidak perlu ganti nama file Anda
prediction_file_xlsx = "hasil_prediksi_diabetes.xlsx" # Nama file tetap sama

hasil_predeksi = None # Inisialisasi variabel untuk menampung DataFrame

try:
    # Coba baca file CSV terlebih dahulu
    hasil_predeksi = pd.read_csv(prediction_file_csv)
    st.info(f"Data hasil **deteksi** berhasil dimuat dari '{prediction_file_csv}'.") # Teks diubah
except FileNotFoundError:
    try:
        # Jika CSV tidak ada, coba baca file Excel
        hasil_predeksi = pd.read_excel(prediction_file_xlsx)
        st.info(f"Data hasil **deteksi** berhasil dimuat dari '{prediction_file_xlsx}'.") # Teks diubah
    except FileNotFoundError:
        st.error("File 'hasil_prediksi_diabetes.csv' atau 'hasil_prediksi_diabetes.xlsx' tidak ditemukan di direktori
        aplikasi. Harap pastikan salah satu file tersebut ada untuk menampilkan hasil **deteksi**.") # Teks diubah
```



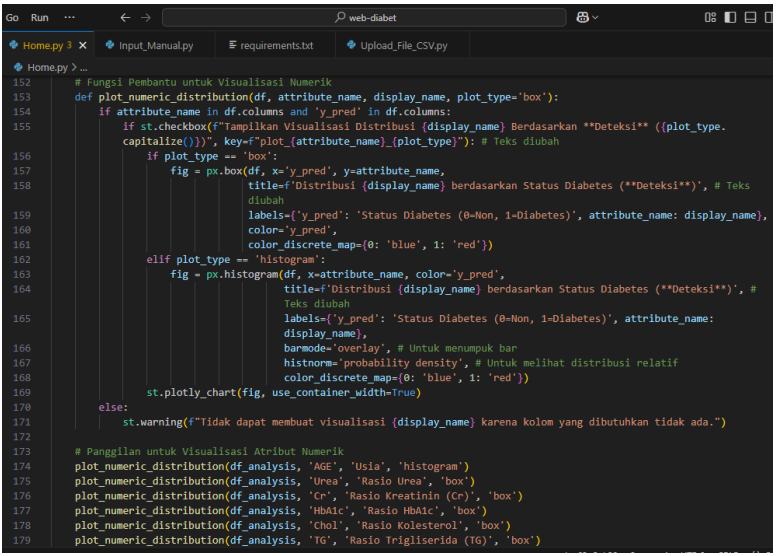
```
            st.stop() # Hentikan eksekusi Streamlit jika file tidak ditemukan
        except Exception as e:
            st.error(f"Terjadi kesalahan saat memuat file Excel '{prediction_file_xlsx}': {e}")
            st.stop()
except Exception as e:
    st.error(f"Terjadi kesalahan saat memuat file CSV '{prediction_file_csv}': {e}")
    st.stop()

# Menampilkan dataframe hasil deteksi
if hasil_predeksi is not None:
    st.dataframe(hasil_predeksi, use_container_width=True) # Tabel akan mengisi lebar kontainer

# --- Analisis Atribut Berisiko ---
if hasil_predeksi is not None: # Pastikan hasil_predeksi berhasil dimuat sebelum analisis
    st.markdown("---")
    st.subheader("Analisis Atribut Berisiko Tinggi Terhadap Diabetes")
    st.write("Berdasarkan data yang **dideteksi** oleh model, berikut adalah beberapa insight terkait atribut yang
    memiliki keterkaitan paling kuat dengan status diabetes:") # Teks diubah

    # --- Fungsi Pembantu untuk Analisis Numerik ---
    def analyze_numeric_attribute(df, attribute_name, display_name, unit=''):
        st.markdown(f"#### Analisis {display_name}")
        if attribute_name in df.columns and 'y_pred' in df.columns:
            attr_diabetes = df[df['y_pred'] == 1][attribute_name]
            attr_non_diabetes = df[df['y_pred'] == 0][attribute_name]

            st.write(f"Rata-rata {display_name} pasien yang **dideteksi** diabetes: **{attr_diabetes.mean():.2f}{unit}
            **") # Teks diubah
            st.write(f"Rata-rata {display_name} pasien yang **dideteksi** non-diabetes: **{attr_non_diabetes.mean():.2f}
            {unit}**") # Teks diubah

            if attr_diabetes.mean() > attr_non_diabetes.mean():
```



```
    def analyze_numeric_attribute(df, attribute_name, display_name, unit=''):
                st.markdown(f"**Kesimpulan**: Pasien dengan nilai **{display_name} yang lebih tinggi** cenderung memiliki
                risiko lebih tinggi untuk **dideteksi** menderita diabetes.") # Teks diubah
            elif attr_diabetes.mean() < attr_non_diabetes.mean():
                st.markdown(f"**Kesimpulan**: Pasien dengan nilai **{display_name} yang lebih rendah** cenderung memiliki
                risiko lebih tinggi untuk **dideteksi** menderita diabetes.") # Teks diubah
            else:
                st.markdown(f"**Kesimpulan**: Dalam konteks data ini, nilai {display_name} tidak menunjukkan perbedaan
                rata-rata yang signifikan antara kelompok diabetes dan non-diabetes (**deteksi**).") # Teks diubah
        else:
            st.warning(f"Kolom '{attribute_name}' atau 'y_pred' tidak ditemukan untuk analisis {display_name}.")

    # --- Analisis Gender ---
    st.markdown("#### Analisis Gender")
    # Asumsi: 0 = Female (Perempuan), 1 = Male (Laki-laki)
    # Sesuaikan mapping ini jika encoding Anda berbeda!
    gender_mapping = {0: 'Perempuan', 1: 'Laki-laki'}

    # Membuat salinan agar tidak mengubah DataFrame asli jika tidak diinginkan
    df_analysis = hasil_predeksi.copy()

    # Pastikan kolom 'Gender' ada sebelum mapping
    if 'Gender' in df_analysis.columns:
        df_analysis['Gender_Label'] = df_analysis['Gender'].map(gender_mapping)

        # Hitung jumlah pasien diabetes (y_pred=1) per gender
        gender_diabetes_counts = df_analysis.groupby('Gender_Label')['y_pred'].sum().reset_index()
        # Hitung total individu per gender
        gender_total_counts = df_analysis.groupby('Gender_Label').size().reset_index(name='Total')

        # Gabungkan kedua DataFrame
        gender_analysis_df = pd.merge(gender_diabetes_counts, gender_total_counts, on='Gender_Label')
```



```
        # Hitung persentase diabetes per gender
        gender_analysis_df['Persentase Diabetes'] = (gender_analysis_df['y_pred'] / gender_analysis_df['Total'] * 100).
        round(2)

        st.write("Jumlah kasus diabetes (**deteksi**) dan persentasenya berdasarkan Gender:") # Teks diubah
        st.dataframe(gender_analysis_df.rename(columns={'y_pred': 'Jumlah Pasien Diabetes (**Deteksi**)'})) # Teks diubah

        if not gender_analysis_df.empty:
            # Temukan gender dengan persentase diabetes tertinggi
            most_affected_gender = gender_analysis_df.loc[gender_analysis_df['Persentase Diabetes'].idxmax()]
            st.markdown(f"**Kesimpulan**: Dari data yang **dideteksi**, **{most_affected_gender['Gender_Label']}**
            menunjukkan persentase kasus diabetes (**deteksi**) yang paling tinggi, yaitu **{most_affected_gender
            ['Persentase Diabetes']}%** dari total individu dengan gender tersebut dalam dataset ini.") # Teks diubah
    else:
        st.warning("Kolom 'Gender' tidak ditemukan dalam data hasil **deteksi** untuk analisis gender.") # Teks diubah

    # --- Analisis Atribut Numerik Lainnya ---
    analyze_numeric_attribute(df_analysis, 'AGE', 'Usia', ' tahun')
    analyze_numeric_attribute(df_analysis, 'Urea', 'Rasio Urea')
    analyze_numeric_attribute(df_analysis, 'Cr', 'Rasio Kreatinin (Cr)')
    analyze_numeric_attribute(df_analysis, 'HbA1c', 'Rasio HbA1c')
    analyze_numeric_attribute(df_analysis, 'Chol', 'Rasio Kolesterol')
    analyze_numeric_attribute(df_analysis, 'TG', 'Rasio Trigliserida (TG)')
    analyze_numeric_attribute(df_analysis, 'HDL', 'Rasio HDL')
    analyze_numeric_attribute(df_analysis, 'LDL', 'Rasio LDL')
    analyze_numeric_attribute(df_analysis, 'VLDL', 'Rasio VLDL')
    analyze_numeric_attribute(df_analysis, 'BMI', 'Rasio BMI')


    # --- Bagian Visualisasi (Opsional) ---
    st.markdown("#### Visualisasi Distribusi Atribut")
    st.write("Anda dapat melihat distribusi atribut utama berdasarkan status diabetes yang **dideteksi**:") # Teks diubah
```



```
    # Fungsi Pembantu untuk Visualisasi Numerik
    def plot_numeric_distribution(df, attribute_name, display_name, plot_type='box'):
        if attribute_name in df.columns and 'y_pred' in df.columns:
            if st.checkbox(f"Tampilkan Visualisasi Distribusi {display_name} Berdasarkan **Deteksi** ({plot_type.
            capitalize()})", key=f"plot_{attribute_name}_{plot_type}"): # Teks diubah
                if plot_type == 'box':
                    fig = px.box(df, x='y_pred', y=attribute_name,
                                 title=f'Distribusi {display_name} berdasarkan Status Diabetes (**Deteksi**)', # Teks
                                 diubah
                                 labels={'y_pred': 'Status Diabetes (0=Non, 1=Diabetes)', attribute_name: display_name},
                                 color='y_pred',
                                 color_discrete_map={0: 'blue', 1: 'red'})
                elif plot_type == 'histogram':
                    fig = px.histogram(df, x=attribute_name, color='y_pred',
                                       title=f'Distribusi {display_name} berdasarkan Status Diabetes (**Deteksi**)', #
                                       Teks diubah
                                       labels={'y_pred': 'Status Diabetes (0=Non, 1=Diabetes)', attribute_name:
                                       display_name},
                                       barmode='overlay', # Untuk menumpuk bar
                                       histnorm='probability density', # Untuk melihat distribusi relatif
                                       color_discrete_map={0: 'blue', 1: 'red'})
                st.plotly_chart(fig, use_container_width=True)
            else:
                st.warning(f"Tidak dapat membuat visualisasi {display_name} karena kolom yang dibutuhkan tidak ada.")

    # Panggilan untuk Visualisasi Atribut Numerik
    plot_numeric_distribution(df_analysis, 'AGE', 'Usia', 'histogram')
    plot_numeric_distribution(df_analysis, 'Urea', 'Rasio Urea', 'box')
    plot_numeric_distribution(df_analysis, 'Cr', 'Rasio Kreatinin (Cr)', 'box')
    plot_numeric_distribution(df_analysis, 'HbA1c', 'Rasio HbA1c', 'box')
    plot_numeric_distribution(df_analysis, 'Chol', 'Rasio Kolesterol', 'box')
    plot_numeric_distribution(df_analysis, 'TG', 'Rasio Trigliserida (TG)', 'box')
```

```
    plot_numeric_distribution(df_analysis, 'HDL', 'Rasio HDL', 'box')
    plot_numeric_distribution(df_analysis, 'LDL', 'Rasio LDL', 'box')
    plot_numeric_distribution(df_analysis, 'VLDL', 'Rasio VLDL', 'box')
    plot_numeric_distribution(df_analysis, 'BMI', 'Rasio BMI', 'box')
```

Lampiran 2: Codingan Halaman Input_Manual.py

pages > Input_Manual.py > ...

```python
import streamlit as st
import pandas as pd
import joblib

model = joblib.load("model/rfc_diabet_model.pkl")

st.title("Input Manual Data Pasien")

with st.form("input_form"):
    gender = st.selectbox("Gender", ["Female", "Male"])
    age = st.number_input("Umur", min_value=1, max_value=120, step=1)
    urea = st.number_input("Rasio Urea", min_value=0.0, step=0.01)
    cr = st.number_input("Rasio Kreatinin", min_value=0.0, step=0.01)
    hba1c = st.number_input("Rasio HbA1c", min_value=0.0, step=0.01)
    chol = st.number_input("Rasio Kolesterol", min_value=0.0, step=0.01)
    tg = st.number_input("Rasio Trigliserida", min_value=0.0, step=0.01)
    hdl = st.number_input("Rasio HDL", min_value=0.0, step=0.01)
    ldl = st.number_input("Rasio LDL", min_value=0.0, step=0.01)
    vldl = st.number_input("Rasio VLDL", min_value=0.0, step=0.01)
    bmi = st.number_input("Rasio BMI", min_value=0.0, step=0.01)

    submitted = st.form_submit_button("Deteksi")
```

pages > Input_Manual.py > ...

```python
if submitted:
    gender_encoded = 1 if gender == "Male" else 0
    input_data = pd.DataFrame([{
        "Gender": gender_encoded,
        "AGE": age,
        "Urea": urea,
        "Cr": cr,
        "HbA1c": hba1c,
        "Chol": chol,
        "TG": tg,
        "HDL": hdl,
        "LDL": ldl,
        "VLDL": vldl,
        "BMI": bmi
    }])

    prediction = model.predict(input_data)[0]

    st.subheader("Hasil:")
    if prediction == 1:
        st.error("Model mendeteksi pasien diabetes.")
    else:
        st.success("Model mendeteksi pasien tidak diabetes.")
```

Lampiran 3: Codingan Halaman Upload_File_CSV

pages > Upload_File_CSV.py > ...

```python
import streamlit as st
import pandas as pd
import joblib

model = joblib.load("model/rfc_diabet_model.pkl")

st.title("Upload File CSV")

uploaded_file = st.file_uploader(
    "Upload file CSV dengan kolom: gender, age, urea, cr, hba1c, chol, tg, hdl, ldl, vldl, bmi",
    type=["csv"]
)

if uploaded_file is not None:
    try:
        df = pd.read_csv(uploaded_file, sep=";")

        rename_dict = {
            'gender': 'Gender',
            'age': 'AGE',
            'urea': 'Urea',
            'cr': 'Cr',
            'hba1c': 'HbA1c',
            'chol': 'Chol',
            'tg': 'TG',
            'hdl': 'HDL',
            'ldl': 'LDL',
            'vldl': 'VLDL',
            'bmi': 'BMI',
            'class': 'CLASS'
        }
```

pages > Upload_File_CSV.py > ...

```python
        df.columns = [col.strip().lower() for col in df.columns]
        df.rename(columns=rename_dict, inplace=True)

        gender_asli = df['Gender'].copy()
        df_for_model = df.copy()

        if 'Gender' in df_for_model.columns and df_for_model['Gender'].dtype == 'object':
            df_for_model['Gender'] = df_for_model['Gender'].apply(lambda x: 1 if str(x).strip().lower() == 'male' else 0)

        if 'CLASS' in df_for_model.columns:
            df_for_model.drop(columns=['CLASS'], inplace=True)

        required_features = ['Gender', 'AGE', 'Urea', 'Cr', 'HbA1c', 'Chol', 'TG', 'HDL', 'LDL', 'VLDL', 'BMI']
        missing_cols = [col for col in required_features if col not in df_for_model.columns]

        if missing_cols:
            st.error(f"File CSV kamu kekurangan kolom: {missing_cols}")
        else:
            predictions = model.predict(df_for_model[required_features])

            df['Gender'] = gender_asli
            df['Hasil Deteksi'] = ["Diabetes" if p == 1 else "Tidak diabetes" for p in predictions]

            st.success("Hasil:")
            st.dataframe(df)

    except Exception as e:
        st.error(f"Terjadi kesalahan saat memproses file: {e}")
```

RIWAYAT HIDUP

A. Identitas Diri

Nama Lengkap	: Nur Ufairah
Tempat & Tgl. Lahir	: Mataram, 16 Desember 2002
Alamat Rumah	: Tolotongga, RT.006/RW.OO3, Kel. Ule, Kec. Asakota, Kota Bima, NTB
HP	: 082340515727
E-mail	: nurufairah161@gmail.com

B. Riwayat Pendidikan

1. MIN Tolobali Kota Bima
2. MTsN 1 Kota Bima
3. MAN 2 Kota Bima

Semarang, 14 Juni 2025

Nur Ufairah

NIM: 2108096014