

LAPORAN TUGAS BESAR KECERDASAN BUATAN
Sistem Pendeteksi Jarak Kendaraan untuk Menghindari Tabrakan



Dibuat Oleh :

Alvin Aditya Bintang Raihan 122490040

Dosen Pengampu :

Ronal, M.Kom. dan Rizki Yustisia Sari, S.T., M.T.I.

PROGRAM STUDI REKAYASA INSTRUMENTASI DAN AUTOMASI

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI SUMATERA

2025

DAFTAR ISI

DAFTAR ISI.....	2
BAB I.....	3
PENDAHULUAN	3
1.1 Latar Belakang.....	3
1.2 Tujuan	3
1.3 Manfaat	4
BAB II.....	5
LANDASAN TEORI.....	5
2.1 Teori Dasar.....	5
2.1.1 Sistem Pendeteksi Jarak	5
2.1.2 Algoritma Decision Tree.....	5
2.1.3 Visual Studio Code (VSCode)	6
2.2 Parameter Dalam Pendeteksi Jarak Kendaraan.....	6
BAB III	7
METODOLOGI.....	7

BAB I

PENDAHULUAN

1.1 Latar Belakang

Keselamatan berkendara merupakan aspek penting yang harus diperhatikan dalam dunia transportasi. Tingginya angka kecelakaan lalu lintas yang disebabkan oleh kurangnya jarak aman antar kendaraan menjadi perhatian serius. Banyak kecelakaan terjadi karena pengemudi tidak menyadari adanya objek atau kendaraan lain di dekatnya, terutama pada saat kondisi cuaca buruk, jalan menurun, atau ketika pengemudi mengalami kelelahan.

Dengan perkembangan teknologi, sistem bantuan pengemudi seperti sensor jarak dan sistem peringatan dini menjadi solusi yang dapat membantu mengurangi risiko kecelakaan. Teknologi sensor seperti ultrasonik, inframerah, akselerometer (ADXL345), dan sensor tekanan udara (BMP280) dapat digunakan untuk mendeteksi kondisi sekitar kendaraan dan memproses data secara real-time untuk memberikan peringatan kepada pengemudi.

Proyek ini mengembangkan sistem pendeteksi jarak kendaraan berbasis algoritma Decision Tree yang mampu memproses berbagai data sensor untuk memprediksi potensi tabrakan. Sistem ini diharapkan dapat menjadi fitur keselamatan tambahan pada kendaraan modern, baik untuk kendaraan pribadi maupun transportasi umum.

1.2 Tujuan

Tujuan dari proyek ini adalah:

- Mengembangkan sistem pendeteksi jarak kendaraan yang mampu memantau kondisi sekitar secara real-time.
- Mengintegrasikan data dari berbagai sensor (ultrasonic, infrared, ADXL345, dan BMP280) untuk analisis potensi tabrakan.
- Mengimplementasikan algoritma Decision Tree untuk melakukan klasifikasi dan prediksi terhadap kondisi berbahaya.

1.3 Manfaat

Manfaat dari sistem ini antara lain:

- Meningkatkan keselamatan berkendara dengan memberikan peringatan dini terhadap potensi tabrakan.
- Mengurangi angka kecelakaan yang disebabkan oleh kelalaian dalam menjaga jarak aman antar kendaraan.
- Membantu pengemudi dalam pengambilan keputusan saat berkendara dalam kondisi berisiko tinggi.
- Mendorong penerapan teknologi sensor dan machine learning dalam sistem keselamatan transportasi.

BAB II

LANDASAN TEORI

2.1 Teori Dasar

2.1.1 Sistem Pendeteksi Jarak

Sistem pendeteksi jarak merupakan teknologi yang digunakan untuk mengukur jarak antara suatu objek dengan kendaraan menggunakan berbagai jenis sensor. Sistem ini sering diaplikasikan pada kendaraan untuk membantu pengemudi mengetahui keberadaan objek di sekitar kendaraan yang tidak terlihat secara langsung, terutama pada saat parkir atau berkendara di kondisi lalu lintas padat.

2.1.2 Algoritma Decision Tree

Algoritma Decision Tree adalah salah satu metode dalam machine learning yang digunakan untuk klasifikasi dan prediksi. Algoritma ini bekerja dengan membagi dataset ke dalam beberapa cabang berdasarkan nilai fitur tertentu hingga mencapai keputusan akhir berupa kelas tertentu. Decision Tree cocok digunakan karena:

- Mudah dipahami dan divisualisasikan
- Mampu menangani data dengan banyak fitur
- Tidak memerlukan normalisasi data

Dalam konteks sistem pendeteksi tabrakan, Decision Tree digunakan untuk mengklasifikasikan kondisi kendaraan berdasarkan input dari sensor, seperti kecepatan, jarak, gesekan ban, dan kecepatan angin.

2.1.3 Visual Studio Code (VSCode)

Visual Studio Code (VSCode) adalah editor kode sumber (source code editor) yang ringan, gratis, dan sangat populer di kalangan pengembang perangkat lunak dan data scientist. VSCode mendukung banyak bahasa pemrograman termasuk Python, dan menyediakan fitur-fitur seperti debugging, integrasi terminal, ekstensi untuk machine learning, dan manajemen proyek yang memudahkan pengembangan sistem pendeteksi jarak kendaraan untuk menghindari tabrakan berbasis Decision Tree. VSCode digunakan dalam pengembangan sistem ini untuk menulis, menjalankan, dan menguji kode program secara efisien

2.2 Parameter Dalam Pendeteksi Jarak Kendaraan

Beberapa parameter penting yang perlu diperhatikan dalam sistem pendeteksi jarak kendaraan antara lain:

- **Kecepatan (Speed):** Merupakan besarnya laju kendaraan yang diukur dalam satuan km/jam atau m/s. Kecepatan tinggi meningkatkan risiko tabrakan, terutama bila tidak dibarengi dengan jarak aman.
- **Jarak (Distance):** Merupakan jarak antara kendaraan dengan objek atau kendaraan lain di depannya. Diukur dengan sensor ultrasonik dan inframerah. Semakin dekat jarak, semakin besar risiko tabrakan.
- **Gesekan Ban (Friction/Grip):** Parameter ini mencerminkan kondisi cengkeraman ban terhadap permukaan jalan. Nilai gesekan dipengaruhi oleh jenis permukaan, kondisi ban, dan cuaca. Friksi yang rendah (misalnya saat hujan atau jalan licin) meningkatkan potensi tergelincir dan memperpanjang jarak pengereman.
- **Kecepatan Angin (Wind Speed):** Diukur menggunakan sensor BMP280 secara tidak langsung (melalui perbedaan tekanan udara). Kecepatan angin mempengaruhi kestabilan kendaraan, terutama kendaraan ringan atau saat melaju kencang di jalan terbuka.

BAB III METODOLOGI

3.1 Alur Pengembangan Sistem

Alur pengembangan sistem dilakukan secara bertahap dan terstruktur. Tahap pertama dimulai dengan pembuatan dataset berbasis data dummy yang merepresentasikan kondisi nyata dalam budidaya udang. Selanjutnya, dilakukan pra-pemrosesan data untuk memastikan kualitas dan konsistensi data yang akan digunakan dalam pelatihan model. Setelah itu, model Decision Tree dibangun dan dilatih menggunakan data tersebut. Kemudian, dilakukan evaluasi terhadap performa model dengan data pengujian. Terakhir, sistem dilengkapi dengan antarmuka pengguna (GUI) untuk memudahkan penggunaan sistem secara praktis.

3.2 Pembuatan Data Set

Dalam pengembangan sistem pendeteksi jarak kendaraan untuk menghindari tabrakan ini, data yang digunakan untuk melatih model Decision Tree tidak diperoleh langsung dari pembacaan sensor fisik, melainkan dibuat secara simulatif dalam bentuk data dummy. Pembuatan data dummy ini bertujuan untuk merepresentasikan berbagai kondisi nyata yang mungkin terjadi saat berkendara, dengan mengacu pada pengetahuan umum serta asumsi logis terkait parameter-parameter penting yang dapat memengaruhi risiko terjadinya tabrakan.

Dataset disusun dengan mempertimbangkan empat fitur utama, yaitu kecepatan kendaraan (dalam satuan km/jam), jarak kendaraan terhadap objek di depannya (dalam satuan cm), nilai gesekan ban yang merepresentasikan kondisi jalan atau ban (dalam bentuk nilai skala atau koefisien), serta kecepatan angin (dalam m/s) yang dapat memengaruhi kestabilan kendaraan. Keempat parameter ini dipilih karena dianggap paling relevan dalam menentukan tingkat keamanan berkendara.

Setiap baris data diberi label klasifikasi risiko tabrakan, yang terdiri dari tiga kategori, yaitu “Aman”, “Waspada”, dan “Bahaya”. Label ini ditentukan berdasarkan kombinasi nilai dari keempat fitur tersebut, dengan mempertimbangkan batasan-batasan tertentu yang ditetapkan secara logis. Selanjutnya, data dummy ini disimpan dalam format CSV dan dibaca menggunakan Python untuk keperluan pra-pemrosesan serta pelatihan model klasifikasi. Dengan pendekatan ini,

sistem tetap dapat dikembangkan dan diuji meskipun belum terhubung langsung dengan sensor atau perangkat keras fisik.

3.3 Pra-pemrosesan Data

Sebelum data dummy digunakan dalam proses pelatihan model klasifikasi Decision Tree, diperlukan tahapan pra-pemrosesan data agar data yang digunakan bersih, konsisten, dan sesuai dengan kebutuhan algoritma. Tahapan ini bertujuan untuk meningkatkan kualitas data serta memastikan bahwa model dapat mempelajari pola secara optimal tanpa gangguan akibat data yang tidak valid atau tidak seragam.

Langkah pertama yang dilakukan adalah pembersihan data, yaitu menghapus entri-entri yang kosong, duplikat, atau tidak logis. Misalnya, nilai kecepatan negatif atau jarak yang melebihi batas maksimal sensor dihapus dari dataset. Setelah itu, dilakukan pengecekan skala antar fitur untuk menentukan apakah diperlukan normalisasi atau standarisasi. Namun karena algoritma Decision Tree tidak sensitif terhadap perbedaan skala, proses normalisasi tidak wajib dilakukan.

Langkah berikutnya adalah mengubah label kategori (Aman, Waspada, Bahaya) ke dalam bentuk numerik agar dapat dikenali oleh algoritma pembelajaran mesin. Proses ini disebut dengan label encoding, misalnya “Aman” dikodekan sebagai 0, “Waspada” sebagai 1, dan “Bahaya” sebagai 2. Setelah semua data bersih dan siap, dataset dibagi menjadi dua bagian, yaitu data latih (training data) dan data uji (testing data), umumnya dengan perbandingan 80:20. Pembagian ini memungkinkan evaluasi performa model terhadap data yang belum pernah dilihat sebelumnya, sehingga akurasi dan generalisasi sistem dapat diuji secara objektif.

3.4 Pelatihan Model Decision Tree

Model kecerdasan buatan dalam sistem ini dibangun menggunakan algoritma Decision Tree dengan bantuan pustaka Scikit-learn dalam bahasa pemrograman Python. Model dilatih menggunakan data fitur dan label yang telah melalui proses pra-pemrosesan. Tujuan dari pelatihan ini adalah agar model mampu mengenali pola-pola dari data input seperti kecepatan kendaraan, jarak, gesekan ban, dan kecepatan angin, untuk kemudian membentuk sebuah struktur pohon keputusan. Pohon keputusan ini digunakan sebagai dasar dalam menentukan apakah kondisi kendaraan berada dalam status "Aman", "Waspada", atau "Bahaya". Proses pelatihan dilakukan

secara bertahap hingga model memperoleh akurasi yang optimal dalam memetakan input terhadap kategori risiko tabrakan.

3.5 Evaluasi Model

Setelah model selesai dilatih, dilakukan evaluasi terhadap performanya dengan menggunakan data uji, yaitu data yang sebelumnya tidak pernah dilihat oleh model. Evaluasi ini bertujuan untuk mengetahui kemampuan model dalam melakukan prediksi terhadap kondisi baru. Metrik utama yang digunakan dalam evaluasi adalah akurasi, yaitu persentase prediksi yang benar dibandingkan dengan total prediksi. Selain itu, digunakan juga confusion matrix untuk menganalisis distribusi prediksi benar dan salah dalam setiap kategori kelas. Evaluasi ini menjadi indikator penting dalam menilai keandalan sistem untuk digunakan dalam skenario nyata, khususnya dalam memberikan peringatan dini terhadap potensi tabrakan kendaraan.

3.6 Pembuatan Antarmuka (GUI)

Untuk meningkatkan kemudahan penggunaan sistem oleh pengguna akhir, dikembangkan sebuah antarmuka grafis (GUI) menggunakan pustaka Tkinter dalam bahasa Python. Antarmuka ini dirancang agar pengguna dapat dengan mudah memasukkan parameter-parameter seperti kecepatan kendaraan, jarak terhadap objek di depan, nilai gesekan ban, serta kecepatan angin. Setelah data dimasukkan, pengguna dapat menekan tombol prediksi untuk menjalankan model Decision Tree dan memperoleh hasil klasifikasi kondisi kendaraan, apakah dalam status “Aman”, “Waspada”, atau “Bahaya”.

Pembuatan GUI dilakukan di lingkungan pengembangan Visual Studio Code (VSCode). Seluruh proses pengembangan menggunakan bahasa Python 3.10 dengan memanfaatkan berbagai pustaka pendukung seperti Pandas, NumPy, Scikit-learn, dan Tkinter. Visual Studio Code dipilih karena mendukung integrasi pustaka secara fleksibel serta menyediakan tampilan interaktif yang memudahkan dalam proses debugging, pengujian model, dan visualisasi hasil klasifikasi.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi

Sistem dikembangkan melalui beberapa komponen utama, dimulai dari pembuatan dan pengolahan dataset hingga pembuatan antarmuka pengguna berbasis grafis. Dataset yang digunakan merupakan data simulasi (data dummy) yang dibuat berdasarkan asumsi logis mengenai kondisi kendaraan dalam situasi nyata. Data ini dimuat dan diolah menggunakan pustaka Pandas untuk proses manipulasi dan analisis.

Model klasifikasi dibangun menggunakan algoritma Decision Tree dari pustaka Scikit-learn, dengan memanfaatkan fitur-fitur utama seperti kecepatan kendaraan, jarak ke objek depan, gesekan ban, dan kecepatan angin. Proses pelatihan dilakukan dengan membagi data ke dalam data latih dan data uji, agar model dapat mengenali pola dan menghasilkan pohon keputusan yang mampu mengklasifikasikan kondisi kendaraan ke dalam kategori "Aman", "Waspada", atau "Bahaya".

Setelah proses pelatihan selesai, dikembangkan antarmuka pengguna (GUI) menggunakan pustaka Tkinter dalam bahasa pemrograman Python. GUI ini dirancang agar pengguna dapat memasukkan nilai-nilai parameter kondisi kendaraan secara manual. Setelah data dimasukkan, pengguna cukup menekan tombol "Prediksi" untuk menjalankan model dan menampilkan hasil klasifikasi berupa tingkat risiko tabrakan.

Seluruh proses pengembangan sistem dilakukan menggunakan lingkungan Visual Studio Code (VSCode). VSCode dipilih karena mendukung fleksibilitas pengelolaan proyek Python, kemudahan integrasi pustaka eksternal, serta tampilan interaktif yang memudahkan proses debugging dan pengujian sistem secara keseluruhan.

4.2 Pengujian Model

Pengujian model dilakukan untuk mengevaluasi performa algoritma Decision Tree dalam mengklasifikasikan tingkat risiko tabrakan kendaraan berdasarkan parameter-parameter yang diberikan. Dataset yang digunakan terdiri dari 200 baris data dummy yang disusun berdasarkan kemungkinan kombinasi kondisi nyata dalam berkendara. Setiap baris data terdiri dari fitur-fitur seperti kecepatan kendaraan, jarak ke objek di depan, gesekan ban, dan kecepatan angin, dengan label target berupa klasifikasi risiko tabrakan: "Aman", "Waspada", atau "Bahaya".

Model dilatih menggunakan pustaka Scikit-learn, kemudian dilakukan pengujian menggunakan data uji yang belum pernah digunakan selama pelatihan. Evaluasi dilakukan dengan menghitung akurasi prediksi model terhadap data uji, serta menyusun confusion matrix untuk melihat distribusi prediksi benar dan salah di tiap kelas. Hasil dari pengujian ini digunakan untuk menilai keandalan dan akurasi sistem dalam memberikan peringatan dini terhadap potensi tabrakan.

4.3 Analisis Hasil

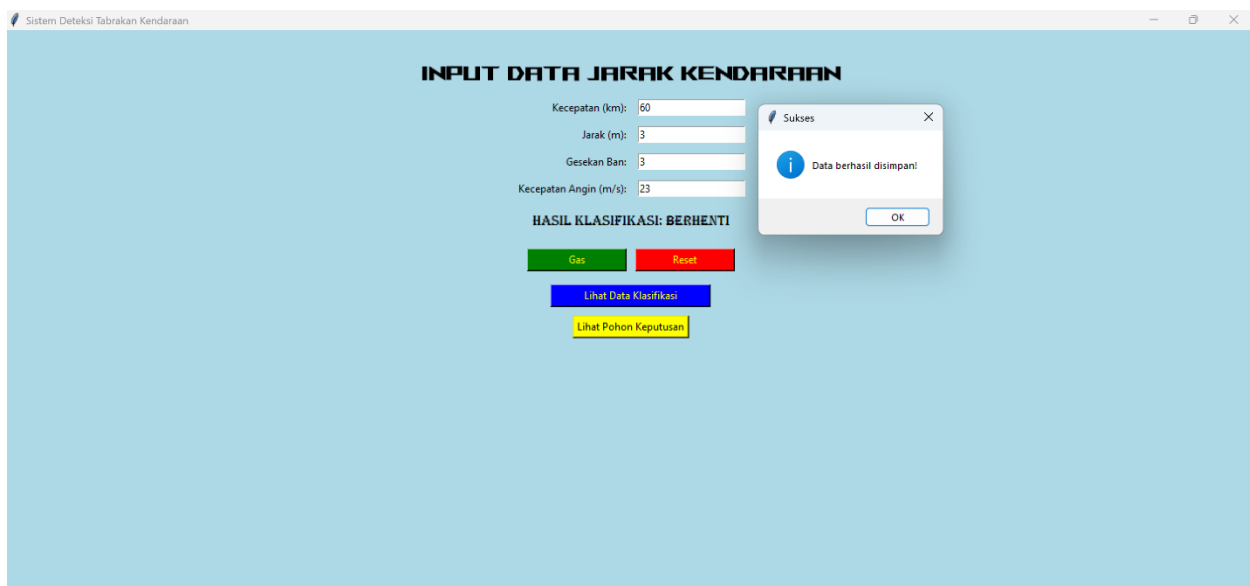
Berikut merupakan tampilan antarmuka dari aplikasi Sistem Pendeteksi Jarak Kendaraan untuk Menghindari Tabrakan yang dikembangkan sebagai alat bantu dalam pengambilan keputusan penghindaran tabrakan secara otomatis berdasarkan data sensor jarak dan kondisi kendaraan.



Gambar 4.1 Tampilan Antarmuka Sistem Pendeteksi Jarak Kendaraan

Gambar tersebut menunjukkan tampilan antarmuka grafis (GUI) dari sebuah aplikasi yang berjudul “Sistem Deteksi Tabrakan Kendaraan”. Aplikasi ini dirancang untuk memprediksi kemungkinan terjadinya tabrakan kendaraan berdasarkan beberapa parameter yang dimasukkan oleh pengguna. Tampilan utama terdiri dari bagian input data yang berada di tengah layar, di mana pengguna diminta untuk mengisi empat parameter penting, yaitu kecepatan kendaraan (dalam km/jam), jarak terhadap objek di depan (dalam meter), gaya gesekan ban, dan kecepatan angin (dalam m/s). Input ini kemungkinan digunakan sebagai fitur dalam algoritma klasifikasi, seperti Decision Tree, untuk menentukan apakah kondisi kendaraan aman atau berisiko.

Di bawah kolom input, terdapat bagian hasil klasifikasi yang ditampilkan dalam bentuk tombol berwarna, seperti tombol hijau bertuliskan "Gas" yang menunjukkan bahwa kendaraan berada dalam kondisi aman untuk melaju, dan tombol merah "Reset" untuk menghapus semua input. Selanjutnya, terdapat dua tombol tambahan yaitu "Lihat Data Klasifikasi" dan "Lihat Pohon Keputusan" yang memungkinkan pengguna untuk menampilkan data klasifikasi yang telah dilakukan serta visualisasi dari model pohon keputusan yang digunakan. Dari ikon-ikon yang terlihat di taskbar, terlihat bahwa aplikasi ini dikembangkan menggunakan Python dan kemungkinan besar memanfaatkan pustaka Tkinter untuk antarmuka pengguna, serta scikit-learn untuk implementasi model machine learning.

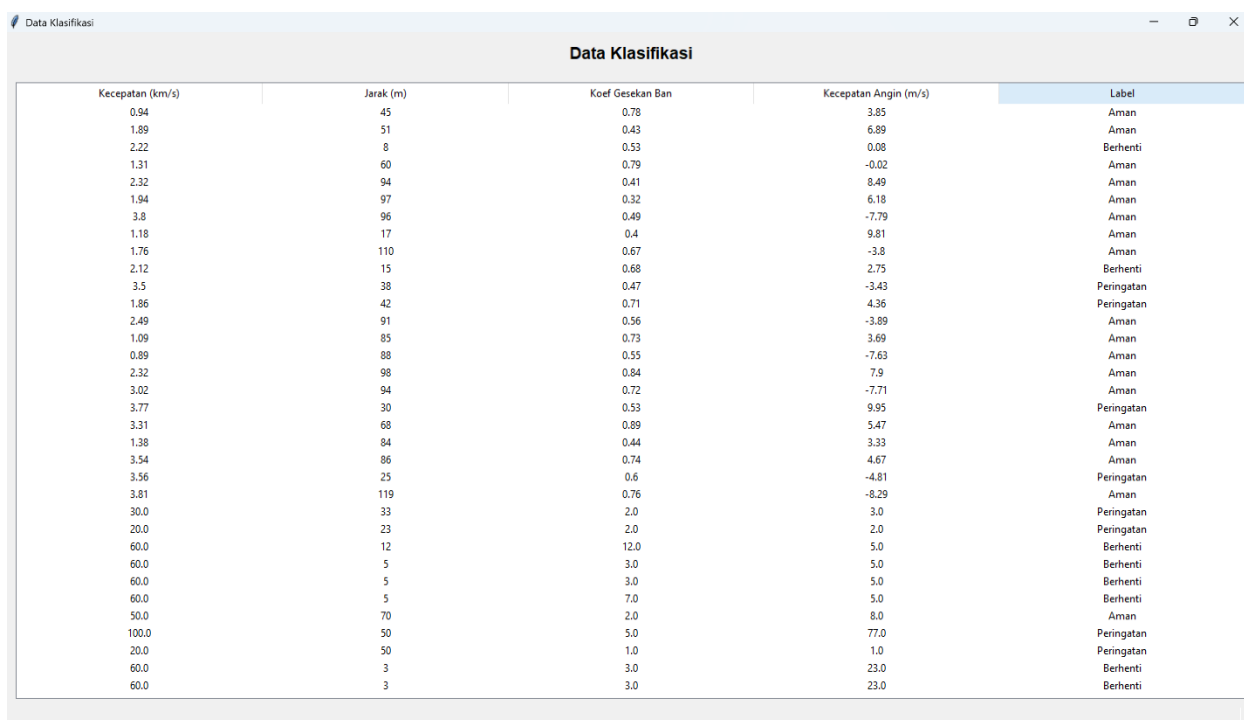


Gambar 4.2 Tampilan Antarmuka Sistem Pendeteksi Jarak Kendaraan

Gambar ini merupakan lanjutan dari tampilan antarmuka aplikasi Sistem Deteksi Tabrakan Kendaraan, yang menunjukkan hasil setelah pengguna melakukan input data dan proses klasifikasi selesai dilakukan. Di bagian kolom input, pengguna telah mengisi nilai-nilai sebagai berikut: Kecepatan = 60 km/jam, Jarak = 3 meter, Gesekan Ban = 3, dan Kecepatan Angin = 23 m/s. Berdasarkan input tersebut, sistem melakukan klasifikasi dan menghasilkan output pada bagian HASIL KLASIFIKASI, yang dalam hal ini bertuliskan “BERHENTI”. Ini menandakan bahwa berdasarkan parameter yang dimasukkan, sistem menyarankan agar kendaraan tidak melaju lebih lanjut karena kemungkinan besar berada dalam situasi berbahaya atau berisiko tinggi terhadap tabrakan.

Selain itu, muncul juga jendela pop-up bertuliskan “Sukses” dengan pesan “Data berhasil disimpan!”, yang menunjukkan bahwa data input beserta hasil klasifikasinya telah disimpan ke dalam file atau database untuk keperluan dokumentasi atau pelatihan model lebih lanjut. Hal ini mengindikasikan bahwa sistem memiliki fungsi penyimpanan data historis, yang sangat berguna untuk analisis lebih lanjut maupun pengembangan model klasifikasi yang lebih akurat.

Keseluruhan tampilan tetap mempertahankan desain antarmuka yang sederhana dan intuitif, dengan elemen-elemen penting seperti tombol "Gas", "Reset", "Lihat Data Klasifikasi", dan "Lihat Pohon Keputusan" yang masih tersedia. Dengan demikian, sistem ini tidak hanya mampu mengklasifikasikan kondisi kendaraan secara real-time berdasarkan input fisik, tetapi juga mendukung penyimpanan dan pelacakan data klasifikasi yang telah dilakukan sebelumnya. Ini merupakan fitur penting dalam sistem berbasis machine learning, khususnya dalam skenario deteksi dini kecelakaan kendaraan.

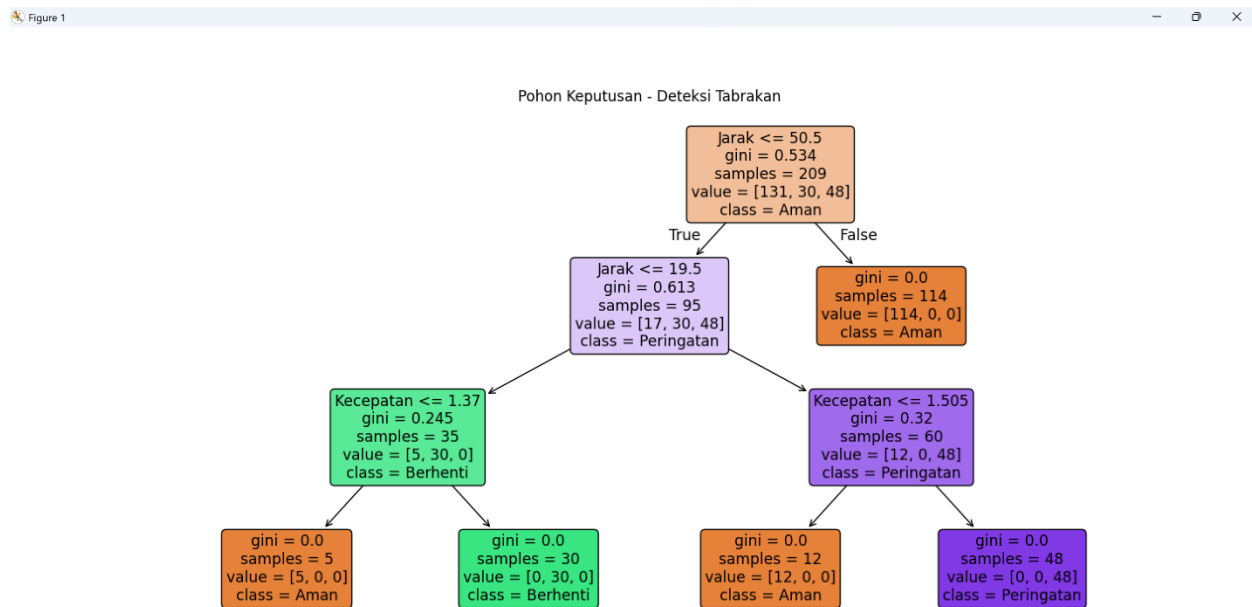


Kecepatan (km/s)	Jarak (m)	Koef Gesekan Ban	Kecepatan Angin (m/s)	Label
0.94	45	0.78	3.85	Aman
1.89	51	0.43	6.89	Aman
2.22	8	0.53	0.08	Berhenti
1.31	60	0.79	-0.02	Aman
2.32	94	0.41	8.49	Aman
1.94	97	0.32	6.18	Aman
3.8	96	0.49	-7.79	Aman
1.18	17	0.4	9.81	Aman
1.76	110	0.67	-3.8	Aman
2.12	15	0.68	2.75	Berhenti
3.5	38	0.47	-3.43	Peringatan
1.86	42	0.71	4.36	Peringatan
2.49	91	0.56	-3.89	Aman
1.09	85	0.73	3.69	Aman
0.89	88	0.55	-7.63	Aman
2.32	98	0.84	7.9	Aman
3.02	94	0.72	-7.71	Aman
3.77	30	0.53	9.95	Peringatan
3.31	68	0.89	5.47	Aman
1.38	84	0.44	3.33	Aman
3.54	86	0.74	4.67	Aman
3.56	25	0.6	-4.81	Peringatan
3.81	119	0.76	-8.29	Aman
30.0	33	2.0	3.0	Peringatan
20.0	23	2.0	2.0	Peringatan
60.0	12	12.0	5.0	Berhenti
60.0	5	3.0	5.0	Berhenti
60.0	5	3.0	5.0	Berhenti
60.0	5	7.0	5.0	Berhenti
50.0	70	2.0	8.0	Aman
100.0	50	5.0	77.0	Peringatan
20.0	50	1.0	1.0	Peringatan
60.0	3	3.0	23.0	Berhenti
60.0	3	3.0	23.0	Berhenti

Gambar 4.3 Tampilan Data Klasifikasi

Data Klasifikasi dari aplikasi *Sistem Deteksi Tabrakan Kendaraan* yang telah dikembangkan. Pada tampilan ini, ditampilkan tabel yang berisi data historis dari hasil klasifikasi sistem berdasarkan input parameter yang telah dimasukkan pengguna sebelumnya. Tabel ini terdiri dari lima kolom, yaitu Kecepatan (km/s), Jarak (m), Koef Gesekan Ban, Kecepatan Angin (m/s),

dan Label. Setiap baris pada tabel ini merepresentasikan satu kondisi kendaraan berdasarkan kombinasi parameter yang diberikan. Label yang dihasilkan oleh sistem terbagi ke dalam tiga kategori, yaitu "Aman", "Peringatan", dan "Berhenti". Label "Aman" menunjukkan kondisi kendaraan tidak berisiko dan masih berada dalam batas keselamatan, sementara "Peringatan" menunjukkan kondisi yang mulai membahayakan dan memerlukan perhatian pengemudi, serta "Berhenti" menunjukkan situasi yang sangat berbahaya dan sistem menyarankan kendaraan untuk berhenti. Sebagai contoh, ketika kecepatan kendaraan tinggi namun jaraknya sangat dekat dan nilai koefisien gesekan rendah, maka sistem akan mengklasifikasikan kondisi tersebut sebagai "Berhenti". Tampilan ini sangat berguna untuk memantau dan menganalisis hasil klasifikasi yang telah dilakukan, serta dapat digunakan sebagai basis data untuk pelatihan ulang model atau evaluasi kinerja sistem dalam mendeteksi potensi tabrakan kendaraan.



Gambar 4.4 Tampilan Struktur Pohon Keputusan

Dari gambar tersebut ditampilkan visualisasi struktur pohon keputusan (*decision tree*) yang digunakan dalam sistem deteksi tabrakan untuk mengklasifikasikan kondisi kendaraan ke dalam tiga kategori, yaitu Aman, Peringatan, dan Berhenti, berdasarkan dua fitur utama yaitu *jarak* dan *kecepatan*. Pohon keputusan ini dimulai dari node utama (root) yang memeriksa apakah nilai Jarak ≤ 50.5 . Jika kondisi ini terpenuhi (benar), maka data akan diarahkan ke cabang kiri untuk

pemeriksaan lebih lanjut. Jika tidak terpenuhi (salah), maka data diarahkan ke cabang kanan dan langsung diklasifikasikan sebagai *Aman*, karena pada simpul tersebut nilai $gini = 0.0$ dengan semua 114 sampel termasuk dalam kelas *Aman*, menandakan bahwa data pada node ini sepenuhnya homogen.

Cabang kiri dari simpul utama melakukan pembagian lanjutan berdasarkan nilai $Jarak \leq 19.5$. Jika kondisi ini benar, maka data diperiksa kembali berdasarkan $Kecepatan \leq 1.37$. Jika kecepatan memenuhi syarat ini, maka pohon akan berakhir di dua simpul: satu dengan 5 data diklasifikasikan sebagai *Aman* ($gini = 0.0$), dan satu lagi dengan 30 data diklasifikasikan sebagai *Berhenti* ($gini = 0.0$). Namun, jika $kecepatan > 1.37$, maka data tetap berada dalam simpul dengan $gini = 0.245$ dan mayoritas diklasifikasikan sebagai *Berhenti*, karena dari total 35 data, 30 di antaranya termasuk dalam kelas tersebut.

Jika nilai $Jarak > 19.5$, maka data diperiksa berdasarkan $Kecepatan \leq 1.505$. Jika memenuhi, data diklasifikasikan sebagai *Aman* dengan jumlah 12 data dan $gini = 0.0$. Jika kecepatan lebih dari 1.505, maka data diklasifikasikan sebagai *Peringatan* dengan 48 sampel, juga dengan $gini = 0.0$, menandakan bahwa semua data dalam simpul ini berasal dari satu kelas.

Setiap node dalam pohon ini menampilkan informasi penting, termasuk nilai gini, yaitu ukuran ketidakmurnian (*impurity*) dari data pada simpul tersebut. Nilai gini berkisar antara 0 hingga 1. Nilai $gini = 0$ menunjukkan bahwa semua data dalam simpul tersebut berasal dari satu kelas, sehingga simpul tersebut bersifat sepenuhnya murni (*pure*). Sebaliknya, semakin besar nilai gini, semakin bercampur data dari beberapa kelas dalam simpul tersebut. Contohnya, pada node utama dengan $gini = 0.534$, data terdiri dari kombinasi tiga kelas yang belum sepenuhnya terpisah. Nilai gini digunakan dalam algoritma decision tree untuk menentukan pemisahan data terbaik di setiap simpul.

Secara keseluruhan, struktur pohon ini menunjukkan bahwa kombinasi nilai *jarak* dan *kecepatan* sangat menentukan dalam pengambilan keputusan untuk mendeteksi potensi tabrakan. Setiap simpul membagi data berdasarkan kondisi tertentu untuk mengarahkan pada klasifikasi akhir yang akurat dan terukur.

4.4 Pembahasan

Tampilan antarmuka aplikasi “Sistem Deteksi Tabrakan Kendaraan” menampilkan sebuah GUI yang dirancang untuk membantu pengambilan keputusan otomatis dalam menghindari tabrakan. Pada tampilan utama, pengguna diminta mengisi empat parameter penting, yaitu kecepatan kendaraan dalam km/jam, jarak terhadap objek di depan dalam meter, koefisien gesekan ban, serta kecepatan angin dalam m/s. Parameter ini menjadi fitur utama dalam algoritma klasifikasi, seperti Decision Tree, yang akan memproses data tersebut untuk menentukan apakah kondisi kendaraan berada dalam status aman, perlu peringatan, atau berisiko tinggi. Di bagian bawah kolom input, terdapat tombol berwarna hijau bertuliskan “Gas” yang menandakan kondisi aman untuk melaju, serta tombol merah “Reset” untuk menghapus semua input. Dua tombol tambahan, yaitu “Lihat Data Klasifikasi” dan “Lihat Pohon Keputusan”, memungkinkan pengguna melihat data historis klasifikasi dan visualisasi model pohon keputusan yang digunakan dalam sistem.

Setelah pengguna memasukkan data, misalnya kecepatan 60 km/jam, jarak 3 meter, gesekan ban 3, dan kecepatan angin 23 m/s, sistem melakukan klasifikasi dan menghasilkan output “BERHENTI”. Output ini menunjukkan bahwa berdasarkan parameter yang dimasukkan, kendaraan berada dalam kondisi berisiko tinggi sehingga sistem menyarankan agar kendaraan berhenti untuk menghindari tabrakan. Selain itu, muncul jendela pop-up yang menginformasikan bahwa data berhasil disimpan, menandakan bahwa input dan hasil klasifikasi direkam ke dalam file atau database. Fitur penyimpanan data ini penting untuk dokumentasi, analisis, dan pelatihan ulang model agar akurasi sistem dapat terus ditingkatkan. Secara keseluruhan, antarmuka dirancang sederhana dan intuitif dengan tombol-tombol penting yang mudah diakses untuk mendukung pengguna dalam proses pengambilan keputusan.

Tampilan data klasifikasi yang tersimpan ditampilkan dalam bentuk tabel yang berisi beberapa kolom utama, yaitu kecepatan kendaraan, jarak, koefisien gesekan ban, kecepatan angin, serta label klasifikasi yang terdiri dari kategori Aman, Peringatan, dan Berhenti. Label tersebut memberikan gambaran tingkat risiko kondisi kendaraan, di mana “Aman” berarti kondisi kendaraan tidak berisiko, “Peringatan” menandakan kondisi yang mulai membahayakan dan memerlukan kewaspadaan, serta “Berhenti” menunjukkan situasi yang sangat berbahaya dan sistem menyarankan penghentian kendaraan. Tabel ini sangat berguna untuk memantau performa

sistem, menganalisis pola risiko, dan sebagai basis data dalam proses evaluasi dan pelatihan ulang model.

Selain itu, visualisasi struktur pohon keputusan (decision tree) yang digunakan dalam sistem juga disajikan dalam antarmuka. Pohon keputusan ini mengklasifikasikan kondisi kendaraan berdasarkan dua fitur utama, yaitu jarak dan kecepatan. Dimulai dari node akar yang memeriksa apakah jarak ≤ 50.5 meter, data akan diarahkan ke cabang yang sesuai untuk pengecekan lebih lanjut. Jika jarak lebih dari nilai tersebut, kendaraan diklasifikasikan sebagai aman. Pada cabang lain, kombinasi nilai jarak dan kecepatan akan menentukan apakah kondisi kendaraan termasuk kategori Aman, Peringatan, atau Berhenti. Setiap node dalam pohon ini juga menampilkan nilai gini yang menunjukkan tingkat kemurnian data pada node tersebut, yang berperan dalam menentukan batas pemisahan terbaik saat pelatihan model. Struktur pohon keputusan ini menggambarkan secara sistematis bagaimana variabel jarak dan kecepatan menjadi faktor utama dalam mengidentifikasi potensi risiko tabrakan kendaraan.

Secara keseluruhan, aplikasi ini mengintegrasikan input parameter fisik kendaraan dengan algoritma machine learning untuk memberikan rekomendasi secara real-time yang dapat meningkatkan keselamatan berkendara. Fitur penyimpanan data historis dan visualisasi model memberikan nilai tambah bagi sistem dalam mendukung evaluasi dan pengembangan model lebih lanjut. Dengan antarmuka yang sederhana namun fungsional, sistem ini dapat menjadi alat yang efektif dalam membantu pengemudi menghindari tabrakan dan meningkatkan keamanan di jalan.

BAB V

KESIMPULAN

5.1 Kesimpulan

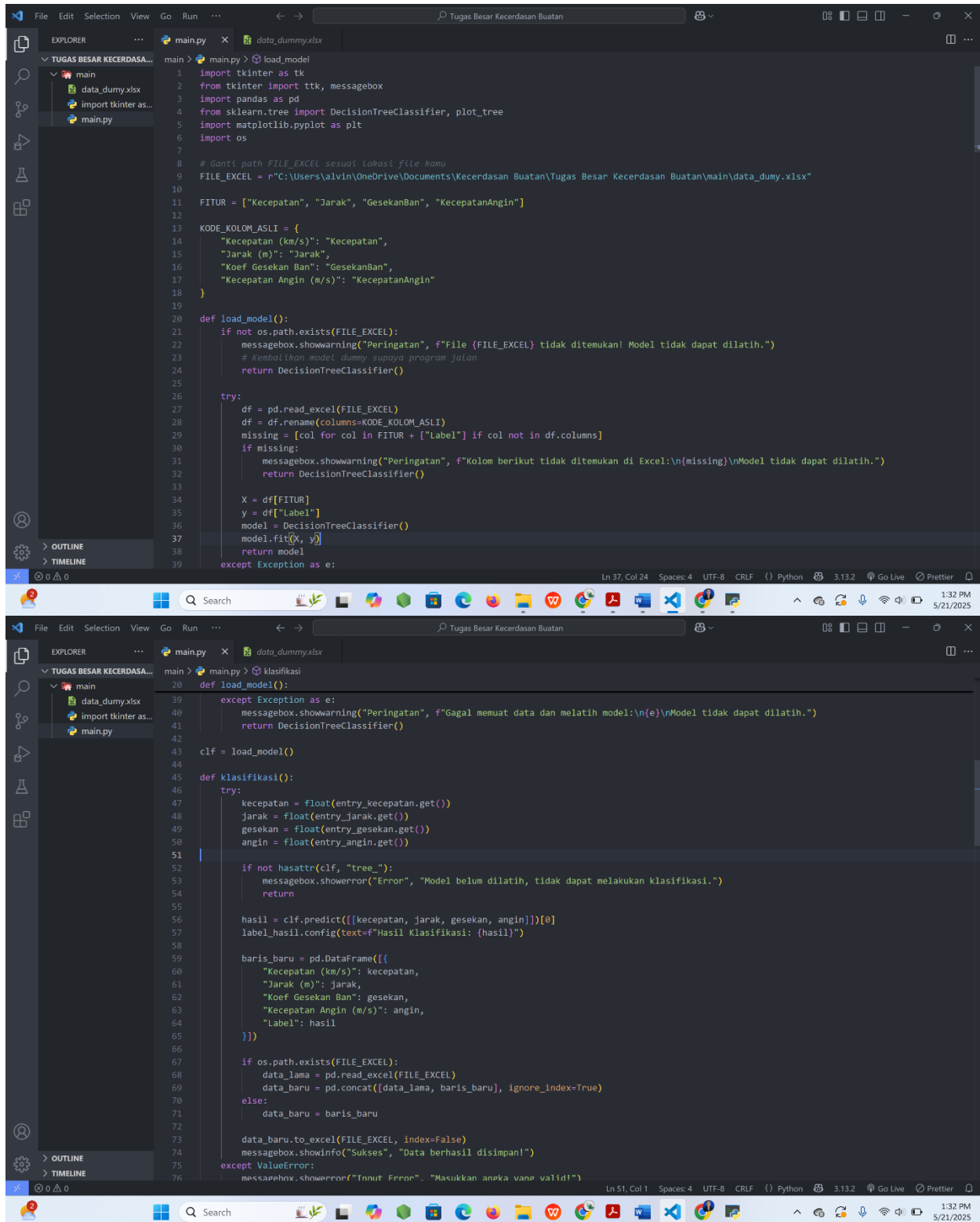
Aplikasi Sistem Pendeteksi Jarak Kendaraan yang dikembangkan berhasil menyediakan antarmuka pengguna yang sederhana dan intuitif untuk membantu pengambilan keputusan dalam menghindari tabrakan secara otomatis. Dengan memanfaatkan data parameter seperti kecepatan kendaraan, jarak terhadap objek depan, koefisien gesekan ban, dan kecepatan angin, sistem mampu mengklasifikasikan kondisi kendaraan ke dalam kategori Aman, Peringatan, dan Berhenti menggunakan algoritma Decision Tree. Fitur penyimpanan data historis dan visualisasi pohon keputusan memberikan dukungan penting untuk analisis performa dan pengembangan model lebih lanjut. Secara keseluruhan, sistem ini berpotensi meningkatkan keselamatan berkendara dengan memberikan rekomendasi yang tepat dan cepat berdasarkan kondisi aktual kendaraan.

5.2 Saran

Untuk pengembangan lebih lanjut, beberapa aspek dapat ditingkatkan agar sistem lebih optimal dan fungsional, antara lain:

1. Mengintegrasikan sensor jarak dan kecepatan secara langsung agar data diperoleh secara real-time tanpa input manual dari pengguna.
2. Menambahkan fitur notifikasi audio dan visual yang lebih responsif untuk meningkatkan efektivitas peringatan kepada pengemudi.
3. Melakukan pengujian sistem pada berbagai kondisi lingkungan dan variasi kecepatan kendaraan untuk memastikan keandalan dan akurasi klasifikasi.

LAMPIRAN



```
File Edit Selection View Go Run ... Tugas Besar Kecerdasan Buatan
EXPLORER main.py data_dummy.xlsx
TUGAS BESAR KECERDASAN BUATAN
main
data_dummy.xlsx
import tkinter as...
main.py
main.py
1 import tkinter as tk
2 from tkinter import ttk, messagebox
3 import pandas as pd
4 from sklearn.tree import DecisionTreeClassifier, plot_tree
5 import matplotlib.pyplot as plt
6 import os
7
8 # Ganti path FILE_EXCEL sesuai Lokasi file kamu
9 FILE_EXCEL = r"C:\Users\alvin\OneDrive\Documents\Kecerdasan Buatan\Tugas Besar Kecerdasan Buatan\main\data_dumy.xlsx"
10
11 FITUR = ["Kecepatan", "Jarak", "GesekanBan", "KecepatanAngin"]
12
13 KODE_KOLOM_ASLI = {
14     "Kecepatan (km/s)": "Kecepatan",
15     "Jarak (m)": "Jarak",
16     "Koef Gesekan Ban": "GesekanBan",
17     "Kecepatan Angin (m/s)": "KecepatanAngin"
18 }
19
20 def load_model():
21     if not os.path.exists(FILE_EXCEL):
22         messagebox.showwarning("Peringatan", f"File {FILE_EXCEL} tidak ditemukan! Model tidak dapat dilatih.")
23         # Kembali model dummy supaya program jalan
24         return DecisionTreeClassifier()
25
26     try:
27         df = pd.read_excel(FILE_EXCEL)
28         df = df.rename(columns=KODE_KOLOM_ASLI)
29         missing = [col for col in FITUR + ["Label"] if col not in df.columns]
30         if missing:
31             messagebox.showwarning("Peringatan", f"Kolom berikut tidak ditemukan di Excel:\n(missing)\nModel tidak dapat dilatih.")
32             return DecisionTreeClassifier()
33
34         X = df[FITUR]
35         y = df["Label"]
36         model = DecisionTreeClassifier()
37         model.fit(X, y)
38         return model
39     except Exception as e:
```

```
File Edit Selection View Go Run ... Tugas Besar Kecerdasan Buatan
EXPLORER main.py data_dummy.xlsx
TUGAS BESAR KECERDASAN BUATAN
main
data_dummy.xlsx
import tkinter as...
main.py
main.py
20 def load_model():
21     except Exception as e:
22         messagebox.showwarning("Peringatan", f"Gagal memuat data dan melatih model:\n(e)\nModel tidak dapat dilatih.")
23         return DecisionTreeClassifier()
24
25     clf = load_model()
26
27     def klasifikasi():
28         try:
29             kecepatan = float(entry_kecepatan.get())
30             jarak = float(entry_jarak.get())
31             gesekan = float(entry_gesekan.get())
32             angin = float(entry_angin.get())
33
34             if not hasattr(clf, "tree_"):
35                 messagebox.showerror("Error", "Model belum dilatih, tidak dapat melakukan klasifikasi.")
36                 return
37
38             hasil = clf.predict([[kecepatan, jarak, gesekan, angin]])[0]
39             label_hasil.config(text=f"Hasil Klasifikasi: {hasil}")
40
41             baris_baru = pd.DataFrame([
42                 {
43                     "Kecepatan (km/s)": kecepatan,
44                     "Jarak (m)": jarak,
45                     "Koef Gesekan Ban": gesekan,
46                     "Kecepatan Angin (m/s)": angin,
47                     "Label": hasil
48                 }
49             ])
50
51             if os.path.exists(FILE_EXCEL):
52                 data_lama = pd.read_excel(FILE_EXCEL)
53                 data_baru = pd.concat([data_lama, baris_baru], ignore_index=True)
54             else:
55                 data_baru = baris_baru
56
57             data_baru.to_excel(FILE_EXCEL, index=False)
58             messagebox.showinfo("Sukses", "Data berhasil disimpan!")
59         except ValueError:
60             messagebox.showerror("Tnntu Frron", "Masukkan angka vane valid!")
```

```
File Edit Selection View Go Run ... Tugas Besar Kecerdasan Buatan
EXPLORER
TUGAS BESAR KECERDASA...
main
data_dummy.xlsx
import tkinter as...
main.py
main.py x data_dummy.xlsx
main > main.py > klasifikasi
45 def klasifikasi():
76     messagebox.showerror("Input Error", "Masukkan angka yang valid!")
77 except Exception as e:
78     messagebox.showerror("Error", f"Terjadi kesalahan:\n{e}")
79
80 def reset_input():
81     entry_kecepatan.delete(0, tk.END)
82     entry_jarak.delete(0, tk.END)
83     entry_gesekan.delete(0, tk.END)
84     entry_angin.delete(0, tk.END)
85     label_hasil.config(text="Hasil Klasifikasi:")
86
87 def tampilkan_tree():
88     if not hasattr(clf, "tree_"):
89         messagebox.showerror("Error", "Model belum dilatih, tidak dapat menampilkan pohon keputusan.")
90         return
91
92     plt.figure(figsize=(10, 7))
93     plot_tree(clf, feature_names=FITUR,
94             class_names=clf.classes_, filled=True, rounded=True)
95     plt.title("Pohon Keputusan - Deteksi Tabrakan")
96     plt.show()
97
98 def tampilkan_data():
99     data_window = tk.Toplevel(root)
100     data_window.title("Data Klasifikasi")
101     data_window.geometry("900x500")
102
103     label_judul = tk.Label(data_window, text="Data Klasifikasi", font=("Helvetica", 14, "bold"))
104     label_judul.pack(pady=10)
105
106     global tabel
107     kolom = ["Kecepatan (km/s)", "Jarak (m)", "Koef Gesekan Ban", "Kecepatan Angin (m/s)", "Label"]
108     tabel = ttk.Treeview(data_window, columns=kolom, show="headings")
109     for col in kolom:
110         tabel.heading(col, text=col)
111         tabel.column(col, anchor="center", width=140)
112         tabel.pack(pady=10, padx=10, fill=tk.BOTH, expand=True)
113
Ln 51, Col 1 Spaces: 4 UTF-8 CRLF Python 3.13.2 Go Live Prettier
```

```
File Edit Selection View Go Run ... Tugas Besar Kecerdasan Buatan
EXPLORER
TUGAS BESAR KECERDASA...
main
data_dummy.xlsx
import tkinter as...
main.py
main.py x data_dummy.xlsx
main > main.py > klasifikasi
114 scrollbar = ttk.Scrollbar(data_window, orient="vertical", command=tabel.yview)
115 tabel.configure(yscrollcommand=scrollbar.set)
116 scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
117
118 def refresh_data():
119
120 def refresh_data():
121     for item in tabel.get_children():
122         tabel.delete(item)
123
124     if os.path.exists(FILE_EXCEL):
125         try:
126             data = pd.read_excel(FILE_EXCEL)
127             data = data.rename(columns={
128                 "Kecepatan (m/s)": "Kecepatan (km/s)",
129                 "Jarak (cm)": "Jarak (m)"
130             })
131             data = data[["Kecepatan (km/s)", "Jarak (m)", "Koef Gesekan Ban", "Kecepatan Angin (m/s)", "Label"]]
132             for _, row in data.iterrows():
133                 tabel.insert("", "end", values=list(row))
134         except Exception as e:
135             messagebox.showerror("Error", f"Gagal memuat data: {e}")
136         else:
137             messagebox.showinfo("Info", "Belum ada data disimpan")
138
139 # == GUI Setup ==
140 root = tk.Tk()
141 root.title("Sistem Deteksi Tabrakan Kendaraan")
142 root.geometry("500x430")
143 root.configure(bg="lightblue")
144
145 frame_main = tk.Frame(root, bg="lightblue")
146 frame_main.pack(padx=20, pady=20)
147
148 judul = tk.Label(frame_main, text="Input Data Jarak Kendaraan", font=("ROG FONTS", 17, "bold"), bg="lightblue", fg="black")
149 judul.pack(pady=10)
150
151 frame_input = tk.Frame(frame_main, bg="lightblue")
152 frame_input.pack()
153
Ln 51, Col 1 Spaces: 4 UTF-8 CRLF Python 3.13.2 Go Live Prettier
```

