Implementasi Machine Learning dengan Algoritma CatBoostClassifier untuk Prediksi Risiko Stres pada Mahasiswa Berbasis Web

1. PENDAHULUAN

1.1 Latar Belakang Masalah

Mahasiswa merupakan kelompok usia produktif yang rentan mengalami tekanan psikologis, terutama dalam lingkungan akademik yang kompetitif. Tekanan tersebut dapat berasal dari tuntutan akademik, manajemen waktu yang buruk, tekanan pekerjaan paruh waktu, masalah finansial, hingga kurangnya dukungan sosial. Jika tidak ditangani dengan baik, tekanan ini dapat berkembang menjadi stres kronis yang berpengaruh negatif terhadap performa belajar, kesehatan mental, dan kualitas hidup mahasiswa.

Menurut Alaydrus et al. (2023), stres pada mahasiswa dapat menurunkan kemampuan kognitif, menurunkan konsentrasi, bahkan memicu gangguan psikologis lain seperti kecemasan dan depresi. Data yang dikumpulkan oleh Siregar dan Dewi (2022) menunjukkan bahwa sekitar 86,9% mahasiswa mengalami stres dalam kategori sedang hingga tinggi, dengan mayoritas merasa tidak mendapatkan dukungan psikologis yang memadai dari institusi pendidikan. Kondisi ini menunjukkan perlunya strategi deteksi dan penanganan dini terhadap stres di kalangan mahasiswa.

Namun, metode konvensional dalam mendeteksi stres seperti observasi klinis dan kuesioner manual memiliki keterbatasan. Pendekatan tersebut cenderung bersifat subjektif, memerlukan waktu lama untuk analisis, dan sangat bergantung pada tenaga profesional. Hal ini menyebabkan deteksi dini terhadap stres sulit dilakukan secara luas dan cepat. Dalam menghadapi tantangan ini, teknologi kecerdasan buatan khususnya machine learning menjadi solusi alternatif yang efisien dan skalabel.

Machine learning memungkinkan sistem untuk belajar dari data dan mengenali pola-pola kompleks yang tidak dapat dianalisis secara manual. Salah satu algoritma yang terbukti efektif dalam klasifikasi data kategorikal dan numerik adalah

CatBoostClassifier. Berdasarkan penelitian Zhou et al. (2021), algoritma CatBoost menunjukkan performa akurasi yang lebih baik dibandingkan model seperti Random Forest dan XGBoost dalam konteks prediksi kesehatan mental. CatBoost juga memiliki keunggulan dalam menangani fitur kategorikal secara otomatis tanpa perlu encoding tambahan, sehingga sangat sesuai untuk data survei mahasiswa yang mengandung banyak variabel non-numerik.

Dengan memanfaatkan machine learning dan data survei mahasiswa, sistem prediksi stres dapat dikembangkan untuk memberikan informasi probabilistik terhadap risiko stres secara personal. Model ini dapat diimplementasikan dalam aplikasi berbasis web agar mudah diakses oleh pengguna, tanpa perlu prosedur medis atau klinis yang kompleks. Oleh karena itu, penelitian ini mengembangkan model klasifikasi berbasis CatBoost untuk mendeteksi risiko stres mahasiswa yang diintegrasikan ke dalam aplikasi web interaktif.

1.2 Rumusan Masalah

1. Seberapa baik performa algoritma CatBoostClassifier dalam memprediksi risiko stres pada mahasiswa berdasarkan fitur input (Gender, Age, CGPA, Sleep Duration, Academic Pressure, Work Pressure, Financial Stress, Financial Problem, Health Issue, Family History of Mental Illness, Diet, Suicidal Thoughts)?
2. Bagaimana kinerja model CatBoostClassifier jika dievaluasi menggunakan metrik confusion matrix, classification report (precision, recall, F1-score), kurva ROC dan nilai AUC, serta analisis feature importance?
3. Bagaimana cara menyajikan hasil prediksi risiko stres dalam bentuk persentase yang informatif dan mudah dipahami sehingga dapat menjadi acuan awal dalam pengambilan keputusan atau langkah preventif?
4. Bagaimana merancang dan mengembangkan aplikasi web berbasis Flask yang dapat diakses oleh mahasiswa berusia 18–25 tahun untuk mendeteksi tingkat risiko stres secara mandiri tanpa pemeriksaan medis langsung?
5. Rekomendasi preventif apa saja yang dapat diberikan berdasarkan hasil prediksi model untuk mendukung manajemen stres mahasiswa, seperti

konseling psikologis, pengelolaan waktu, perbaikan kualitas tidur, serta intervensi nutrisi dan sosial?

1.3 Ruang Lingkup

Penelitian ini akan dibatasi pada hal-hal berikut antara lain :

1. Penelitian ini difokuskan pada perancangan dan pengembangan aplikasi web menggunakan framework Flask dan bahasa pemrograman Python untuk memprediksi tingkat risiko stres pada mahasiswa berusia 18–25 tahun sebagai pengguna utama.
2. Dataset yang digunakan dalam penelitian ini diambil dari platform Kaggle dengan judul *Student Depression Dataset*, yang berisi data hasil survei non-medis terhadap mahasiswa. Fitur input yang digunakan mencakup: *Gender, Age, CGPA, Sleep Duration, Academic Pressure, Work Pressure, Financial Stress, Financial Problem, Health Issue, Family History of Mental Illness, Diet, dan suicidal thoughts*.
3. Pengembangan sistem prediksi dilakukan dengan menerapkan algoritma *CatBoostClassifier*, yang dioptimasi menggunakan *RandomizedSearchCV* untuk memperoleh kombinasi *hyperparameter* terbaik dan meningkatkan akurasi model.
4. Evaluasi performa model dilakukan secara kuantitatif melalui metrik evaluasi seperti confusion matrix, classification report (precision, recall, f1-score), kurva ROC dan AUC, serta analisis feature importance untuk mengukur keandalan dan interpretabilitas model dalam memprediksi risiko stres.
5. Output dari sistem ini berupa persentase risiko stres, yang disajikan secara informatif dan mudah dipahami oleh pengguna untuk dijadikan acuan awal dalam mengambil langkah preventif.
6. Sistem yang dikembangkan bersifat non-klinis dan tidak dimaksudkan sebagai alat diagnosis medis, melainkan sebagai alat bantu deteksi dini untuk mendukung upaya promotif dan preventif kesehatan mental mahasiswa.

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk menerapkan metode machine learning, khususnya algoritma CatBoost Classifier, guna menghasilkan prediksi yang akurat terhadap tingkat stres pada mahasiswa berdasarkan berbagai fitur input seperti jenis kelamin, usia, IPK, durasi tidur, tekanan akademik, tekanan pekerjaan, stres finansial, masalah keuangan, kondisi kesehatan, riwayat keluarga dengan gangguan mental, pola makan, dan pemikiran bunuh diri. Selain itu, penelitian ini juga bertujuan untuk mengevaluasi performa model secara komprehensif melalui metrik evaluasi seperti confusion matrix, classification report (meliputi precision, recall, dan f1-score), kurva ROC, serta nilai AUC dan analisis feature importance, agar diperoleh model yang andal dan dapat diinterpretasikan dengan baik. Hasil prediksi risiko stres mahasiswa disajikan dalam bentuk persentase yang informatif dan mudah dipahami, sehingga dapat dimanfaatkan sebagai acuan awal dalam pengambilan keputusan maupun langkah-langkah preventif. Penelitian ini juga mencakup pengembangan aplikasi web berbasis framework Flask yang dapat diakses oleh mahasiswa berusia 18–25 tahun sebagai alat bantu deteksi mandiri tanpa memerlukan pemeriksaan medis langsung. Terakhir, penelitian ini diharapkan dapat memberikan rekomendasi berbasis hasil prediksi model untuk mendukung upaya promotif kesehatan mental mahasiswa, seperti konseling psikologis, pengelolaan waktu, peningkatan kualitas tidur, serta perbaikan pola makan dan dukungan sosial.

1.5 Sistematika Penulisan

Bagian ini menjelaskan sistematika penulisan yang digunakan dalam penelitian ini. Penelitian ini disusun dalam lima bab. Bab pertama, yaitu Bab 1 Pendahuluan, menguraikan latar belakang masalah, ruang lingkup, tujuan penulisan, dan sistematika penulisan. Bab kedua, Bab 2 Tinjauan Pustaka, membahas teori-teori dan pandangan para ahli yang relevan dengan topik penelitian serta menyajikan penelitian-penelitian terkait yang relevan. Selanjutnya, Bab 3 Analisis dan Perancangan, memberikan gambaran umum tentang penelitian, termasuk sumber data dan langkah-langkah yang dilakukan selama proses

penelitian. Bab 4 Implementasi dan Uji Coba, menyajikan hasil dari pembuatan dan pengujian program yang telah dikembangkan. Terakhir, pada Bab 5 Kesimpulan dan Saran, menyimpulkan hasil penelitian dan memberikan rekomendasi untuk penelitian serupa di masa depan.

2. TINJAUAN PUSTAKA

2.1 Stres

Stres pada hakikatnya merupakan kondisi adaptasi tubuh dan pikiran saat menghadapi tekanan yang melampaui kemampuan seseorang. Menurut Hans Selye, stres adalah respon tubuh yang bersifat non-spesifik terhadap setiap tuntutan atau beban yang dialaminya. Asosiasi Psikologi Indonesia (IKAPI) juga menyatakan bahwa stres merupakan reaksi non-spesifik manusia terhadap rangsangan atau tekanan (stressor) yang berasal dari lingkungan. Dalam bidang psikologi, Sarafino dan Smith (2011) menekankan bahwa stres timbul ketika terjadi interaksi antara individu dan lingkungannya, sehingga individu tersebut mempersepsikan adanya ketidaksesuaian antara tuntutan eksternal dengan kemampuan dirinya. Berdasarkan definisi-definisi di atas, stres dapat dipahami sebagai gabungan reaksi fisiologis dan psikologis yang muncul sebagai respons terhadap tekanan atau perubahan yang dihadapi individu.

Menurut Barseli, Ifdil, dan Nikmarijal (2017), stres akademik terjadi ketika seseorang merasa bahwa tuntutan akademik melebihi kapasitasnya untuk mengatasi, sehingga menimbulkan respon negatif secara fisik, emosional, dan kognitif. Dalam konteks ini, stres tidak selalu bersifat negatif, tetapi jika tidak dikelola dengan baik, dapat menyebabkan gangguan kesehatan mental maupun fisik. Afwillah, Amalia, dan Yunus (2025) menambahkan bahwa stres muncul sebagai reaksi terhadap ketidakseimbangan antara tuntutan yang datang dan kemampuan individu dalam menghadapinya, yang seringkali dipicu oleh tekanan waktu, tanggung jawab berlebihan, atau ekspektasi yang tinggi.

2.2.1 Jenis - Jenis Stres

Penggolongan stres terdiri dari dua jenis yaitu stres yang membangun atau dikenal dengan istilah eustress. Eustress membuat seseorang menjadi bersemangat dan meningkatkan konsentrasi belajar, sedangkan stres yang sifatnya tidak menyenangkan dan bersifat negatif dikenal dengan istilah distress. Distress memberikan berbagai dampak negatif seperti menjadi gugup, tekanan darah tinggi, mudah marah, dan sulit berkonsentrasi. Stres akademik tergolong stres negatif

(distress). Stres akademik terjadi didalam lingkungan pendidikan. Stres negsatif ini muncul ketika mahasiswa dihadapkan dengan tuntutan- tuntutan dan tugas yang harus dikerjakan oleh mahasiswa.

Menurut Jenita DT Donsu (2017) secara umum stress dibagi menjadi dua yaitu :

a. Stress Akut Stress yang dikenal juga dengan flight or flight response. Stress akut adalah respon tubuh terhadap ancaman tertentu, tantangan atau ketakutan. Response stress akut yang segera dan intensif di beberapa keadaan dapat menimbulkan gemetaran.
b. Stress Kronis Stress kronis adalah stress yang lebih sulit dipisahkan atau diatasi, dan efeknya lebih Panjang dan lebih.

Menurut Priyoto (2014) menurut gejalanya stress dibagi menjadi tiga yaitu:

a. Stress Ringan Stress ringan adalah stressor yang dihadapi setiap orang secara teratur, seperti banyak tidur, kemacetan lalu lintas, kritikan dari atasan. Situasi stress ringan berlangsung beberapa menit atau jam saja. Ciri-ciri stress ringan yaitu semangat meningkat, penglihatan tajam, energy meningkat namun cadangan energinya menurun, kemampuan menyelesaikan pelajaran meningkat, sering merasa letih tanpa sebab, kadang-kadang terdapat gangguan system seperti pencernaan, otak, perasaan tidak santai. Stress ringan berguna karena dapat memacu seseorang untuk berpikir dan berusaha lebih Tangguh menghadapi tantangan hidup.
b. Stress Sedang Stress sedang berlangsung lebih lama daripada stress ringan. Penyebab stress sedang yaitu situasi yang tidak terselesaikan dengan rekan, anak yang sakit, atau ketidakhadiran yang lama dari anggota keluarga. Ciri-ciri stress sedang yaitu sakit perut, mules, otot-otot tegang, perasaan tegang, gangguan tidur, badan terasa ringan.
c. Stress Berat Stress berat adalah situasi yang lama dirasakan oleh seseorang dapat berlangsung beberapa minggu sampai beberapa bulan, seperti perselisihan perkawinan secara terus menerus, kesulitan financial yang berlangsung lama karena tidak ada perbaikan, berpisah dengan keluarga, berpindah tempat tinggal mempunyai penyakit kronis dan termasuk perubahan

fisik, psikologis social pada usia lanjut. Ciri-ciri stress berat yaitu sulit beraktivitas, gangguan hubungan social, sulit tidur, negatifistic, penurunan konsentrasi, takut tidak jelas, keletihan meningkat, tidak mampu melaakukan pekerjaan sederhana, gangguan system meningkat, perasaan takut meningkat.

Karakteristik Stres menurut (Priyoto 2014):

a. Sering merasa letih tanpa sebab,
b. Penglihatan tajam,
c. Gangguan pencernaan (sakit perut),
d. Perasaan tegang,
e. Gangguan tidur,
f. Sulit beraktivitas krn penurunan konsentrasi
g. Negatifistic.

2.2.2 Faktor Penyebab Risiko Stres

Identifikasi faktor risiko stres pada mahasiswa menjadi langkah krusial dalam upaya memahami dan menanggulangi beban psikologis yang mereka alami. Berikut ini dipaparkan berbagai faktor risiko yang memengaruhi tingkat stres pada mahasiswa, yang menjadi dasar dalam penelitian prediksi menggunakan CatBoostClassifier. Faktor-faktor tersebut meliputi :

1. **Demografi**

   Banyak penelitian menunjukkan mahasiswa perempuan lebih rentan mengalami stres daripada laki-laki. Di Fakultas Kesehatan Masyarakat UNESA Surabaya, mahasiswa perempuan dilaporkan 2,2 kali lebih berisiko mengalami stres berat dibandingkan laki-laki. Demikian pula, studi lain menemukan mayoritas mahasiswa perempuan melaporkan stres akademik tinggi (71%). Dari sisi usia, stres akademik tinggi paling banyak ditemukan pada usia akhir masa studi; misalnya, 30,2% mahasiswa dengan stres akademik tinggi berusia 24 tahun. Secara umum, remaja akhir (usia sekitar 18-24) lebih rentan terhadap stres dibandingkan kelompok usia yang lebih tua.

2. **Akademik**

Tekanan akademik merupakan sumber stres utama mahasiswa. Stres akademik timbul dari tuntutan nilai tinggi, beban tugas yang banyak, waktu belajar yang lama, dan kecemasan menghadapi ujian. Misalnya, Barseli et al. (2018) mencatat bahwa tekanan untuk meraih nilai baik, banyaknya tugas, dan kekhawatiran ujian adalah stressor utama dalam pembelajaran. Prestasi akademik (IPK) juga berhubungan dengan stres: studi di Universitas Warmadewa menunjukkan bahwa bahkan mahasiswa ber-IPK tinggi pun banyak mengalami stres. Tabel distribusi stres menunjukkan 55,1% mahasiswa dengan predikat Pujian mengalami stres, 54% berpredikat Sangat Memuaskan mengalami stres, sedangkan pada kategori Memuaskan (IPK lebih rendah) proporsi yang stres mencapai 65%. Dengan kata lain, mahasiswa dengan prestasi lebih rendah cenderung menghadapi stres lebih tinggi.

3. **Durasi dan Kualitas Tidur**

   Pola tidur yang tidak memadai kerap memicu atau memperparah stres. Penelitian Amirotul & Kuncoro (2022) menunjukkan terdapat korelasi positif antara kualitas tidur dan tingkat stres pada mahasiswa akhir - semakin buruk kualitas tidurnya (yang sering akibat durasi tidur yang pendek), semakin tinggi stres yang dialami. Standar tidur ideal bagi dewasa muda adalah 7-9 jam per malam; durasi tidur di bawah itu dapat menurunkan kualitas tidur dan meningkatkan gangguan fisik serta psikologis yang berkontribusi pada stres.

4. **Tekanan Pekerjaan**

   Mahasiswa yang harus bekerja sambil kuliah menghadapi beban ganda akademik dan pekerjaan yang intensif. Lestari & Putri (2024) melaporkan bahwa kombinasi tanggung jawab kuliah dan pekerjaan banyak menimbulkan stres, mengganggu penyelesaian tugas serta menurunkan performa akademik mahasiswa. Tekanan dari pekerjaan (seperti keharusan lembur, tuntutan atasan, atau konflik peran) ditambah beban studi dapat memperbesar risiko stres karena waktu istirahat berkurang dan kewajiban bertambah.

5. **Finansial**

   Kesulitan ekonomi adalah pemicu stres yang umum di kalangan mahasiswa. Stres finansial diartikan sebagai tekanan psikologis akibat ketidakstabilan

keuangan atau masalah keuangan keluarga. Dwiasti & Sipayung (2024) menemukan bahwa aspek-aspek seperti masalah uang saku, tekanan biaya kuliah, beban biaya hidup, dan kesulitan keuangan keluarga secara signifikan memengaruhi konsentrasi dan prestasi akademik mahasiswa. Artinya, ketidakmampuan memenuhi kebutuhan finansial menimbulkan kecemasan tinggi yang mengganggu studi. Hal ini sejalan dengan teori bahwa kekhawatiran keuangan adalah faktor "off-job" yang memicu stres. Dengan kata lain, tekanan ekonomi (utang kuliah, biaya hidup mahal, dsb.) menambah beban stres mahasiswa, sehingga penting dukungan finansial dan literasi keuangan bagi mereka.

6. **Kesehatan**

   Masalah kesehatan fisik juga dapat memperburuk stres. Gangguan kesehatan (kronis maupun akut) mengganggu aktivitas sehari-hari dan menambah kecemasan, sehingga termasuk pemicu stres "off the job". Selain itu, riwayat keluarga dengan gangguan mental meningkatkan kerentanan mahasiswa terhadap stres. Tinjauan sistematis menunjukkan mahasiswa dengan riwayat keluarga gangguan jiwa memiliki risiko lebih tinggi mengalami masalah kesehatan mental (depresi, kecemasan, dsb.) di perguruan tinggi. Artinya, adanya gangguan psikologis dalam keluarga adalah faktor risiko genetik dan lingkungan yang membuat mahasiswa lebih mudah terpengaruh stres psikologis.

7. **Pola Makan**

   Pola makan yang tidak sehat atau tidak seimbang dapat memperparah tingkat stres melalui mekanisme fisiologis. Nikarli dkk. (2024) menjelaskan bahwa stres berat menyebabkan kondisi psikis memburuk dan pola makan menjadi tidak teratur serta kurang sehat, sedangkan stres rendah cenderung dikaitkan dengan pola makan yang lebih baik dan teratur. Konsumsi makanan yang tidak sehat juga mengganggu keseimbangan hormon tubuh (misalnya kortisol dan leptin), yang pada gilirannya dapat meningkatkan respons stres. Dengan kata lain, diet tinggi gula, garam, atau lemak tidak sehat dapat memicu peradangan

kronis dan ketidakseimbangan hormon, sehingga meningkatkan kerentanan seseorang terhadap stres.

8. **Pikiran Bunuh Diri**

   Meskipun pikiran bunuh diri lebih lazim dilihat sebagai akibat stres berat, keberadaan ide-ide tersebut juga menandakan tingkat stres yang tinggi. Iriani dkk. (2024) melaporkan bahwa mahasiswa yang memiliki ide bunuh diri biasanya dipicu oleh tingkat stres yang sangat tinggi, dan data penelitian tersebut menunjukkan korelasi positif yang sangat signifikan antara tingkat stres dan munculnya ide bunuh diri. Penelitian di Universitas Andalas (Putra dkk., 2023) juga menemukan bahwa stres akademik merupakan salah satu faktor signifikan yang berhubungan dengan munculnya ide bunuh diri pada mahasiswa. Dengan demikian, mahasiswa yang mengalami pikiran bunuh diri umumnya berada dalam tekanan psikologis yang berat, sehingga ide bunuh diri dapat dianggap sebagai salah satu indikator atau faktor risiko peningkatan stres pada individu tersebut. Hal ini menegaskan pentingnya deteksi dini stres tinggi untuk mencegah gejala psikologis berat seperti ide bunuh diri.

Dengan pemahaman terhadap berbagai faktor risiko tersebut, intervensi pencegahan seperti edukasi tentang gaya hidup sehat dan teknik manajemen stres dapat diterapkan secara strategis untuk menurunkan tingkat stres, khususnya di kalangan mahasiswa usia 17-25 tahun. Kesadaran akan faktor-faktor pemicu ini juga mendorong individu untuk mengenali kondisi berisiko dan mengambil langkah proaktif agar terhindar dari dampak negatif stres.

2.2 Artificial Intelligence

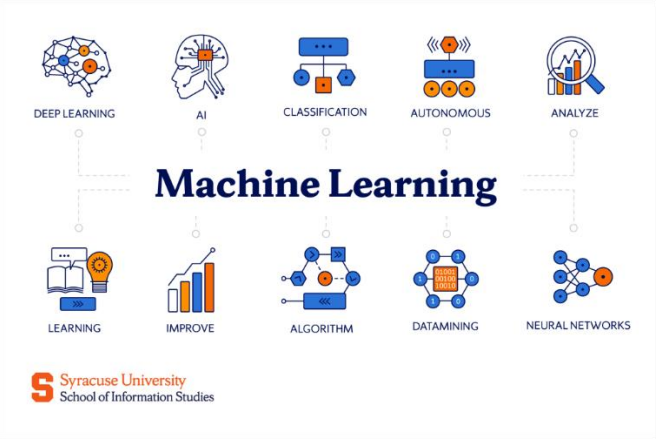
Artificial Intelligence (AI) merupakan suatu sistem yang dikembangkan agar memiliki kecerdasan layaknya manusia dalam suatu bidang tertentu, dan dikenal dengan istilah *soft computing*. Pendekatan ini memungkinkan sistem cerdas untuk beradaptasi terhadap perubahan lingkungan serta menangani kondisi yang tidak pasti, tidak tepat, atau bersifat parsial, sehingga solusi yang diberikan tetap relevan dengan kondisi nyata (Ahmad, 2017).

Secara umum, AI adalah teknologi yang dirancang untuk meniru kemampuan

berpikir makhluk hidup maupun objek tidak hidup dalam memecahkan permasalahan. Menurut Rich (1991), AI merupakan studi tentang bagaimana komputer dapat melakukan tugas-tugas yang sebelumnya hanya dapat dilakukan secara lebih baik oleh manusia. Tujuan utama AI adalah merepresentasikan proses berpikir manusia dan menciptakan mesin yang dapat berperilaku serupa, dengan dukungan pengetahuan dan kemampuan penalaran. Terdapat tiga metode utama dalam AI, yaitu:

1. *Machine Learning* (ML): metode ini meniru cara manusia belajar dan menggeneralisasi dalam menyelesaikan masalah, dan saat ini menjadi pendekatan yang paling banyak diterapkan (Ahmad, 2017).
2. *Fuzzy Logic* (FL): pendekatan ini meniru cara makhluk hidup dalam menghadapi ketidakpastian, dengan memberikan keputusan tidak secara biner (0 atau 1), tetapi berdasarkan logika *fuzzy*. Penerapan nyatanya dapat ditemukan dalam sistem pengereman kereta api di Jepang (Ahmad, 2017).
3. *Evolutionary Computing* (EC): metode ini mengadaptasi proses seleksi evolusioner untuk menghasilkan solusi optimal, seperti yang terlihat pada algoritma genetika, *Particle Swarm Optimization* (PSO), dan *Simulated Annealing* (Ahmad, 2017).

2.3 Machine Learning



Gambar 2.1 Machine Learning

Sumber : ischool.syracuse.edu/

Machine Learning (ML) atau pembelajaran mesin adalah cabang dari kecerdasan buatan (Artificial Intelligence/AI) yang memungkinkan sistem komputer untuk belajar secara otomatis dari data dan meningkatkan kinerjanya dalam menyelesaikan tugas tertentu tanpa harus diprogram secara eksplisit (Alzubaidi et al., 2021). ML berfokus pada pengembangan algoritma yang mampu mengidentifikasi pola serta membuat prediksi atau keputusan berdasarkan data historis.

Machine learning menawarkan banyak manfaat. Sistem ML dapat menangani data dalam jumlah besar dan kompleks dengan lebih cepat dibanding metode tradisional, serta belajar secara mandiri tanpa perlu pemrograman berulang oleh manusia. Hal ini berguna untuk menyelesaikan beragam masalah (misalnya deteksi penipuan, prediksi diagnosa medis, atau pemrosesan bahasa alami) dengan presisi tinggi. ML memungkinkan analisis data yang lebih mendalam dan deteksi pola tersembunyi dalam data, sehingga keputusan yang diambil lebih akurat dan adaptif terhadap perubahan data. Misalnya, dalam sektor kesehatan ML dapat meningkatkan presisi diagnosis dan mengurangi kesalahan manusia. Secara keseluruhan, ML dirancang untuk mempermudah proses pengolahan data dan memberikan wawasan yang sebelumnya sulit ditemukan, yang pada akhirnya meningkatkan kualitas dan kecepatan pengambilan keputusan.

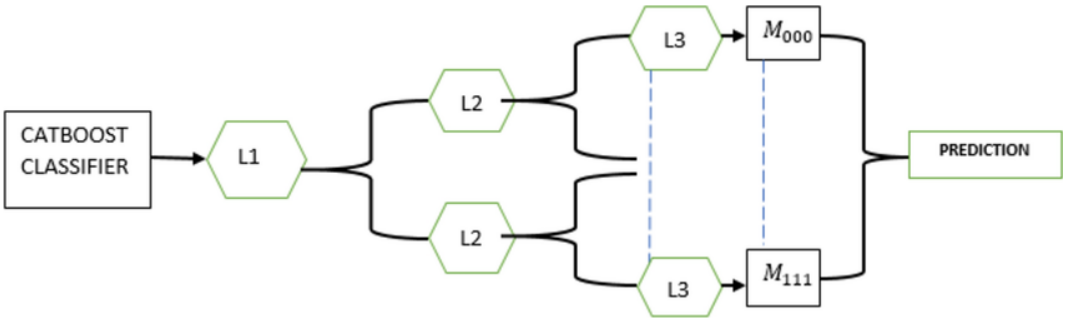
2.3.1 Algoritma CatboostClassifier

CatBoost adalah algoritma pembelajaran mesin yang termasuk dalam keluarga Gradient Boosted Decision Trees (GBDT) dan berada dalam lingkup ensemble learning. Algoritma ini bersifat *open source* serta dikembangkan untuk supervised machine learning dengan dua inovasi utama, yaitu Ordered Target Statistics dan Ordered Boosting (Hancock & Khoshgoftaar, 2020). Dalam beberapa kajian, penerapan gradient boosting ditemukan menghadapi beberapa masalah statistik. Model yang dibentuk dari tahap boosting berulang kali sangat tergantung pada keseluruhan data latih, menyebabkan pergeseran distribusi pada model tersebut. Selain itu, algoritma ini juga kurang optimal dalam menangani atribut kategorikal. Untuk mengatasi hal tersebut, teknik Ordered Boosting dimodifikasi

dari algoritma gradient boosting standar guna mencegah *target leakage*. Selain itu, CatBoost menyertakan algoritma khusus untuk memproses data kategorikal dengan lebih efisien. Kombinasi kedua pendekatan ini kemudian dikenal sebagai algoritma CatBoost, atau Categorical Boosting (Hancock & Khoshgoftaar, 2020).

Menurut tinjauan literatur oleh Gratton et al. (2020), CatBoost unggul dalam menangani dataset dengan variabel kategorikal dan heterogen, dan dalam banyak kasus outperform XGBoost atau LightGBM terutama pada data beragam (Gratton et al., 2020). Selain itu, CatBoost mengurangi kebutuhan praproses data seperti *one-hot encoding*, sehingga lebih efisien dalam pipeline analisis data (ML Journey, 2024).

Dalam konteks prediksi risiko stres mahasiswa (klasifikasi), CatBoostClassifier dipilih karena beberapa pertimbangan teknis. Pertama, dataset mahasiswa sering kali memiliki fitur kategorikal yang tidak dapat langsung digunakan oleh banyak algoritma ML. CatBoost mampu mengolah fitur kategorikal langsung tanpa perlu *one-hot-encoding* tambahan. Hal ini mengurangi beban praproses dan risiko kehilangan informasi.



Gambar 2.2 Struktur CatboostClassifier

Sumber : researchgate.net/

Dari gambar 2.2 diatas, CatBoost memulai dengan preprocessing fitur melalui *Ordered Target Statistics*, kemudian membangun serangkaian pohon keputusan simetris secara bertahap (symmetric trees). Setiap tahap pembentukan pohon selanjutnya memanfaatkan mekanisme *Ordered Boosting*, yakni penggunaan

permutasi data sehingga estimasi gradien hanya diperoleh dari data sebelumnya dalam urutan tersebut, sehingga menghindari *target leakage* dan mengurangi bias (Prokhorenkova, 2017). Kombinasi antara pengkodean kategorikal yang terstruktur dan boosting terurut ini memungkinkan CatBoost menghasilkan model akhir ditunjukkan pada diagram sebagai M_{000} dan M_{111} yang kemudian digabungkan untuk menghasilkan prediksi akhir dengan tingkat generalisasi yang kuat.

2.4 Evaluasi Model

Evaluasi model klasifikasi dalam penelitian ini dilakukan menggunakan beberapa metrik standar yang bertujuan untuk mengukur keakuratan, kemampuan diskriminasi, serta interpretabilitas model CatBoostClassifier. Pertama, confusion matrix digunakan untuk menampilkan jumlah prediksi benar dan salah True Positive (TP), True Negative (TN), False Positive (FP), dan False Negative (FN)sehingga dapat dihitung akurasi keseluruhan, presisi (TP/(TP+FP)), recall (TP/(TP+FN)), dan F1-Score sebagai rata-rata harmonik antara presisi dan recall (Dewi, 2021). Data confusion matrix juga membantu mengidentifikasi jenis kesalahan paling sering terjadi, misalnya kesalahan tipe I (FP) atau tipe II (FN) yang kritis dalam konteks deteksi stres mahasiswa.

Selain itu, classification report dari scikit-learn menampilkan metrik presisi, recall, F1-Score, dan support untuk masing-masing kelas (stres tinggi vs stres rendah), memberikan gambaran kuantitatif kinerja model pada setiap kategori (Tamba et al., 2023). Metrik ini sangat membantu dalam menilai apakah model lebih cenderung over-prediksi atau under-prediksi suatu kelas tertentu.

Selanjutnya, kurva ROC (Receiver Operating Characteristic) dan nilai AUC (Area Under the Curve) dievaluasi untuk menilai kemampuan model dalam membedakan kedua kelas di berbagai ambang batas (threshold). Kurva ROC memplot True Positive Rate (TPR) terhadap False Positive Rate (FPR), sedangkan AUC mengukur luas di bawah kurva tersebutsemakin mendekati 1, semakin baik diskriminasi model (Nisa, 2024).

Terakhir, feature importance CatBoost digunakan untuk menentukan fitur mana yang paling berkontribusi terhadap prediksi. CatBoost menyediakan nilai

kepentingan fitur, misalnya berdasarkan perubahan rata-rata prediksi saat suatu fitur diacak (Prediction Values Change). Analisis ini mempermudah pemahaman model (interpretability) dengan menyoroti variabel kritis seperti Academic Pressure atau Sleep Duration yang dominan memengaruhi estimasi risiko stres (Ananda, 2023).

Dengan kombinasi metrik kuantitatif (akurasi, presisi, recall, F1-Score, AUC) dan metode interpretasi (feature importance), evaluasi model ini memastikan bukan hanya performa yang tinggi, tetapi juga transparansi dalam pengambilan keputusan prediksi stres mahasiswa.

2.5 Shapley Additive Explanations (SHAP)

Metode Shapley Additive Explanations (SHAP) menyediakan pendekatan kuat untuk mengungkap kontribusi setiap fitur terhadap hasil prediksi (Rodríguez-Pérez & Bajorath, 2020). Penelitian sebelumnya (Permatasari et al., 2022; L. Wang et al., 2021) menunjukkan bahwa penerapan SHAP pada CatBoost mampu memberikan interpretasi kontribusi fitur yang baik pada prediksi diabetes melitus dan Analisis Stabilitas Seismik yang Efisien pada Lereng Tanggul yang Mengalami Perubahan Muka Air.

Metode Shapley Additive Explanations (SHAP) memungkinkan pemanfaatan nilai Shapley secara luas untuk tujuan interpretasi model secara keseluruhan. SHAP memberikan informasi tentang tingkat kepentingan variabel yang mendukung pemahaman model black box, sehingga sangat bermanfaat dalam menangani isu kerawanan pangan. Pemilihan fitur adalah elemen krusial dalam machine learning yang memastikan hanya variabel yang relevan dan terkait dengan variabel target yang dimasukkan ke dalam model, untuk menjaga tingkat akurasi (Sivasankar & Vijaya, 2019). Penggunaan variabel yang tidak relevan dapat menyebabkan penurunan akurasi model. Sebaliknya, pemilihan variabel yang tepat dapat meningkatkan kualitas prediksi (Heinze et al., 2018). Dengan mengurangi jumlah variabel, diharapkan model dapat mencapai akurasi yang lebih tinggi dan mengidentifikasi variabel yang paling signifikan, sehingga menghasilkan prediksi yang lebih akurat untuk masalah yang dihadapi (Nilsson et al., 2007).

Dalam penelitian ini, metode Shapley Additive Explanations (SHAP) diterapkan pada model CatBoostClassifier yang telah dilatih untuk memprediksi kemungkinan stres pada mahasiswa. SHAP berfungsi sebagai alat interpretasi model, memungkinkan identifikasi fitur-fitur mana yang paling berpengaruh terhadap hasil prediksi depresi. Dengan menghitung nilai Shapley, SHAP memberikan informasi mendalam tentang kontribusi setiap fitur (seperti tekanan akademik, tekanan kerja, durasi tidur, dll.) terhadap prediksi individu dan mendukung pemahaman yang lebih baik mengenai faktor-faktor kunci yang berkontribusi pada stres mahasiswa.

2.6 Bahasa Pemrograman Python

Gambar 2.6 Logo Python

Sumber :python.org

Python adalah bahasa pemrograman tingkat tinggi yang ditafsirkan, berorientasi objek, dan memiliki semantik dinamis. Bahasa ini mendukung pengembangan aplikasi cepat dan scripting berkat pengetikan dan pengikatan dinamisnya. Sintaks Python yang sederhana memudahkan pembelajaran dan meningkatkan keterbacaan, sehingga mengurangi biaya pemeliharaan. Python juga mendukung modul dan paket, yang mendukung modularitas dan penggunaan ulang kode. *Interpreter* Python serta pustaka standarnya yang luas tersedia secara gratis untuk berbagai platform utama (Python, 2024).

Adopsi Python tersebar luas karena sintaks yang jelas dan mudah dibaca. Sering digunakan dalam analisis data, pembelajaran mesin (ML), dan pengembangan *web*, Python menghasilkan kode yang mudah dibaca, dipahami, dan dipelajari. Persyaratan lekukan Python untuk pernyataan sumber membantu membuat kode konsisten dan mudah dibaca. Aplikasi yang dikembangkan dengan kode Python cenderung lebih kecil daripada perangkat lunak yang dibangun dengan bahasa pemrograman seperti Java. *Programmer* umumnya harus mengetik lebih

sedikit kode.

Dalam penelitian ini, Python berperan sebagai bahasa pemrograman utama yang mengintegrasikan seluruh tahapan pengolahan data, pengembangan, dan evaluasi model prediksi risiko stres pada mahasiswa. Kemampuan Python dalam menangani seluruh siklus proyek data mulai dari pemuatan, pembersihan, rekayasa fitur, hingga pelatihan dan evaluasi model CatBoostClassifier menjadikannya pilihan yang tepat dalam konteks penelitian ini. Pustaka Python seperti pandas, scikit-learn, catboost, matplotlib, seaborn, dan SHAP digunakan secara ekstensif untuk tugas-tugas spesifik di setiap tahapan.

2.7 Google Colab

Gambar 2.6 Logo Google Colab

Sumber :googlecolab.com

Google Colaboratory, atau Google Colab, adalah dokumen eksekusi yang memungkinkan penulisan, pengeditan, dan berbagi program *Python* yang tersimpan di Google Drive atau yang baru dibuat, tanpa perlu menginstal perangkat lunak tambahan. Hanya membutuhkan akun Google untuk *login* dan menyimpan *file* Python. Google Colab cocok untuk *programmer* maupun pemula yang ingin belajar Python, serta mendukung kolaborasi tim seperti Google Docs. Mirip dengan Jupyter Notebook yang berbasis browser, Google Colab memungkinkan menjalankan *program* Python tanpa perlu instalasi atau unduhan editor teks, karena seluruh *setup* dilakukan di *cloud* (Anendya, 2023).

Salah satu keunggulan utama Google Colab adalah kemampuannya untuk mendukung kolaborasi real-time, serupa dengan Google Docs, sehingga pengguna

dapat berbagi dan mengedit notebook secara *real-time*. Notebook yang disimpan di Google Drive juga bisa diakses dari perangkat apa pun dan kapan saja, memudahkan kerja tim dalam proyek penelitian. Selain itu, Colab juga menyediakan akses gratis ke GPU (Graphics Processing Unit) dan TPU (Tensor Processing Unit) untuk menjalankan proses komputasi berat seperti pelatihan model machine learning dan deep learning tanpa memerlukan perangkat keras khusus. Platform ini dilengkapi dengan beragam pustaka Python pra-instal, mulai dari pandas, scikit-learn, CatBoost, hingga TensorFlow, PyTorch, dan OpenCV, serta fleksibilitas untuk menginstal pustaka tambahan melalui perintah !pip install, menjadikannya lingkungan yang ideal untuk digunakan dalam berbagai proyek penelitian dan pengembangan.

Dalam penelitian ini, Google Colab digunakan sebagai lingkungan pengembangan utama untuk membangun, melatih, dan mengevaluasi model *CatBoostClassifier* secara efesien. Platform ini memfasilitasi pemuatan dan pembersihan dataset, pelaksanaan hyperparameter tuning, serta visualisasi metrik evaluasi tanpa memerlukan konfigurasi lokal yang kompleks. Menurut Prasanna et al. (2021), Google Colab merupakan salah satu platform paling banyak digunakan di dunia akademik karena aksesibilitas, fleksibilitas, dan kemampuan integrasinya yang tinggi terhadap berbagai teknologi komputasi modern.

2.8 Flask

Gambar 2.10 Logo Flask

Sumber : flask.palletsprojects.com/

Flask adalah *microframework* web berbasis Python yang dirancang untuk memudahkan pembuatan aplikasi web secara ringkas dan efisien. Sebagai *microframework*, Flask hanya menyediakan core minimal, sehingga pengembang

dapat menambahkan komponen lain sesuai kebutuhan melalui ekstensi. Evan dan Saian (2023) menyebut Flask digunakan untuk membangun aplikasi web dengan cepat dan efisien, sementara Adinda Amelia (2024) menegaskan bahwa Flask adalah kerangka kerja yang dirancang untuk memudahkan pengembang dalam membuat aplikasi web yang ringkas dan efisien. Dokumenasi Flask juga lengkap, mendukung pengembang dalam memelihara aplikasi.

Beberapa fitur utama Flask antara lain:

- Microframework minimalis - hanya berisi *core* ringan sehingga Flask sangat cepat dan hemat memori.
- Templating Jinja2 - Flask menggunakan *engine* templating Jinja2 untuk membuat halaman HTML dinamis dengan mudah.
- Routing dan HTTP - Pengembang dapat membuat *route* untuk menangani permintaan HTTP menggunakan decorator @app.route, menjadikan pengelolaan *request-response* sederhana.
- Ekstensi luas - Tersedia banyak ekstensi tambahan (misalnya Flask-SQLAlchemy untuk database, Flask-WTF untuk form, dll) yang memperkaya fungsionalitas aplikasi. Sebagai contoh, Flask-WTF disediakan untuk mempermudah pembuatan dan validasi form input pengguna.
- Dokumentasi dan komunitas - Flask memiliki dokumentasi komprehensif dan komunitas aktif, sehingga pengembang mendapat dukungan dalam pengembangan dan perawatan aplikasi.

Beberapa studi kasus di Indonesia mengilustrasikan penggunaan Flask untuk aplikasi prediksi. Bagastio (2023) membangun *website* prediksi harga saham menggunakan Flask dan model LSTM; dia mencatat bahwa *hasil prediksi ditampilkan melalui website* yang telah dibangun. Demikian pula, Fandi (2024) menerapkan model XGBoost dalam Flask sebagai backend untuk prediksi hasil pertandingan Dota 2, dengan antarmuka web berbasis HTML/TailwindCSS agar user-friendly. Contoh-contoh ini menunjukkan bahwa Flask memungkinkan pembuatan antarmuka input (form) dan penyajian output prediksi secara terintegrasi dalam satu aplikasi web.

Dalam penelitian ini, Flask digunakan sebagai kerangka *back-end* utama untuk membangun sistem prediksi risiko depresi pada mahasiswa. Melalui Flask, aplikasi web dikembangkan agar pengguna dapat dengan mudah menginput data pribadi dan faktor-faktor risiko seperti tekanan akademik, tekanan pekerjaan, kebiasaan tidur, dan riwayat keluarga. Data yang diinput kemudian diproses oleh model machine learning yang telah diintegrasikan ke dalam aplikasi, dan hasil prediksi risiko depresi ditampilkan secara langsung melalui antarmuka web yang responsif. Pada penelitian ini, Flask juga memanfaatkan fitur templating HTML (Jinja2) untuk membangun tampilan antarmuka pengguna yang interaktif dan informatif, sehingga seluruh proses input, prediksi, hingga penyajian hasil dapat dilakukan dalam satu platform yang terintegrasi. Dengan demikian, Flask berperan sebagai framework yang utuh, tidak hanya sebagai backend, tetapi juga dalam membangun frontend aplikasi prediksi dan edukasi kesehatan mental bagi mahasiswa.

2.10 Flowchart

Flowchart atau diagram alir adalah diagram yang terdiri atas gambar dan simbol untuk menggambarkan langkah-langkah penyelesaian suatu permasalahan yang ada pada sistem untuk mempermudah pengguna dalam mema- hami informasi dan rangkaian proses yang ada pada suatu prosedur. Simbol- simbol yang ada pada flowchart beserta keterangannya dapat dilihat pada gambar 2.11

No	Simbol	Nama	Keterangan
1		*Terminal*	Menyatakan awal dan akhir suatu proses.
2		Arus	Menghubungkan satu simbol dengan simbol lainnya.
3		Proses	Menyatakan suatu tindakan atau proses pada program.
4		*Input / Output*	Menyatakan proses *input* dan *output*.
5		*Preparation*	Menyiapkan tempat penyimpanan untuk hasil pengolahan data.
6		*Decision*	Menyatakan kondisi tertentu dengan dua kemungkinan antara ya atau tidak.
7		*Predefine*	Menyatakan suatu proses yang berada dalam sub program.
8		*Connector*	Menyatakan sambungan dari satu proses ke proses yang lain dalam halaman yang sama.
9		*Offline Connector*	Menyatakan sambungan dari satu proses ke proses yang lain pada halaman yang berbeda.

Gambar 2.11 Simbol dan keterangan flowchart

2.9 Visual Studio Code

Gambar 2.9 Logo Visual Studio Code

Sumber : visualstudiocode.com

Visual Studio Code adalah editor kode sumber gratis yang ringan namun kuat, tersedia untuk Windows, macOS, Linux, Raspberry Pi OS, serta versi web. Editor ini mendukung JavaScript, TypeScript, dan Node.js secara bawaan, serta memiliki ekosistem ekstensi yang luas untuk bahasa pemrograman lain, *runtime*, lingkungan, dan *platform cloud*. Visual Studio Code juga menyediakan fitur seperti *IntelliSense* untuk melengkapi kode, *debugging* grafis, linting, pengeditan multi-

kursor, petunjuk *parameter*, navigasi dan *refactoring* kode, serta kontrol versi bawaan termasuk dukungan Git. Banyak fitur ini diadaptasi dari teknologi Visual Studio (Heller, 2022).

Dalam penelitian ini, Visual Studio Code digunakan sebagai lingkungan pengembangan utama untuk membangun keseluruhan aplikasi web prediksi risiko stres pada mahasiswa. Seluruh proses pengembangan, mulai dari penulisan kode *backend* menggunakan Flask, pembuatan dan pengelolaan template HTML untuk antarmuka pengguna, hingga integrasi model machine learning ke dalam aplikasi, dilakukan di Visual Studio Code.. Dengan demikian, Visual Studio Code berperan penting sebagai pusat aktivitas pengembangan, memungkinkan seluruh tahapan pembuatan aplikasi mulai dari input data, pemrosesan model, hingga penyajian hasil prediksi dapat dilakukan secara terintegrasi dalam satu platform pengembangan yang modern dan fleksibel.

2.10 Penelitian Terdahulu

Penelitian ini dilakukan dengan mengkaji berbagai karya terdahulu yang relevan dalam memprediksi tingkat stres mahasiswa, sebagai dasar pembelajaran penelitian yang penulis akan lakukan dapat dilihat pada tabel 2.1 Rangkuman Penelitian Sejenis.

Tabel 2.1 Rangkuman Penelitian Sejenis

No	Penulis & Tahun	Judul (Sumber)	Algoritma & Dataset	Hasil Utama	GAP (Celah Penelitian)
1	Wafiqi dkk. (2024)	*Prediksi Tingkat Stres Pada Mahasiswa UNUGHA Cilacap Menggunakan Algoritma K-Nearest*	KNN; survei 90 mahasiswa	Akurasi 83,33%	Penelitian ini hanya menguji KNN tanpa optimasi, feature engineering, interpretasi model, atau penyimpanan

		Neighbor (Jurnal Tekno Kompak)			untuk deployment.
2	Alfiyan Aldi dkk. (2020)	*Deteksi Dini Tingkat Stres Pada Mahasiswa Menggunakan Metode Iterative Dichotomiser 3 dan K-Nearest Neighbour* (JIAC Univ. Pancasila)	ID3 & KNN; kuesioner 300 mahasiswa	KNN ≈ 91%, ID3 ≈ 61%	Hanya menggunakan dua algoritma dasar tanpa preprocessing mendalam, optimasi parameter, interpretasi hasil, atau perbandingan model lain.
3	Almira V. F. dkk. (2023)	*Implementasi Metode Machine Learning Untuk Mendeteksi Tingkat Stres Manusia Berdasarkan Kualitas Tidur* (SENAMIKA UPNVJ)	KNN, SVM, DT; dataset "SaYoPillow" (630 sampel, 8 fitur)	KNN & SVM 100%, DT 99%	Mengandalkan dataset eksternal tanpa penyesuaian lokal, tanpa feature engineering, tuning parameter, atau analisis interpretabilitas (SHAP).

4	Kurniawan dkk. (2024)	*Analisis Klasifikasi Tingkat Kesehatan Mental pada Mahasiswi Akhir dalam Menempuh Tugas Akhir Menggunakan Algoritma Support Vector Machine Studi Kasus: Universitas Darussalam Gontor* (Teknoka UHAMKA)	SVM (RBF & polinomial); kuesioner 249 mahasiswi	Akurasi 68%	Hanya menguji SVM tunggal dengan pengaturan default tanpa feature engineering, optimasi hyperparameter, interpretasi model, atau penyimpanan.
5	Harahap & Yamasari (2024)	*Klasifikasi Tingkat Stres Mahasiswa Menggunakan RMSProp untuk Arsitektur Artificial Neural Network* (JINACS Unesa)	ANN (RMSProp), SMOTE-N; data mahasiswa	Akurasi 96% (dgn SMOTE-N) vs 88% (tanpa)	Fokus pada ANN dan teknik balancing tanpa eksplorasi algoritma boosting modern, tanpa pembuatan fitur baru, tuning lanjutan, atau interpretasi.

6	Muhammad Reza (2024)	*Pengembangan Sistem Prediksi Kesehatan Mental Tingkat Stres Pada Mahasiswa Di Lingkungan Uin Ar-Raniry Menggunakan Algoritma C4.5 (Studi Kasus: Fakultas Tarbiyah Dan Keguruan)* (Skripsi UIN Ar-Raniry)	C4.5; kuesioner 135 mahasiswa	Akurasi 88,8%	Hanya menggunakan Decision Tree tanpa benchmarking model lain, preprocessing lanjutan, feature engineering, optimasi, atau analisis SHAP.

Berdasarkan tinjauan terhadap enam penelitian terdahulu, sebagian besar studi hanya menerapkan algoritma dasar seperti K-Nearest Neighbor (Wafiqi et al., 2024), ID3 dan KNN (Alfiyan Aldi et al., 2020), SVM (Kurniawan et al., 2024), Decision Tree C4.5 (Reza, 2024), q, SVM, serta Decision Tree pada dataset eksternal (Almira et al., 2023). Pendekatan tersebut umumnya belum membandingkan kinerja dengan metode boosting modern, belum menerapkan rangkaian preprocessing dan feature engineering yang komprehensif termasuk pembuatan fitur baru dan penanganan missing values serta terbatas pada konfigurasi parameter default tanpa optimasi hyperparameter. Selain itu, hampir tidak ada studi yang mengintegrasikan analisis interpretabilitas model, misalnya melalui SHAP, untuk mengungkap kontribusi masing-masing fitur, maupun mempertimbangkan aspek deployment seperti penyimpanan model untuk implementasi aplikasi nyata. Kondisi ini menunjukkan perlunya penelitian

selanjutnya yang menjalankan pipeline end-to-end secara menyeluruh, mulai dari data cleaning, eksplorasi, feature engineering, optimasi model, hingga interpretasi dan persiapan deployment.

3. METODOLOGI PENELITIAN

3.1 Gambaran Umum

Penelitian ini bertujuan mengembangkan sebuah aplikasi web yang dapat secara otomatis memprediksi tingkat risiko stres pada mahasiswa tanpa memerlukan keahlian khusus di bidang kesehatan mental. Dataset yang digunakan diambil dari platform Kaggle dengan judul Student Depression Dataset, berisi sekitar 27.901 entri hasil survei non-medis terhadap mahasiswa. Setiap entri mencakup fitur-fitur seperti Jenis Kelamin (Gender), Usia (Age), Indeks Prestasi (CGPA), Durasi Tidur (Sleep Duration), Tekanan Akademik (Academic Pressure), Tekanan Kerja (Work Pressure), Stres Finansial (Financial Stress), Masalah Finansial (Financial Problem), Masalah Kesehatan (Health Issue), Dukungan Sosial (Social Support), Riwayat Keluarga Gangguan Mental (Family History of Mental Illness), Pola Diet (Diet), Hilang Minat (Lost Interest), dan Pikiran Bunuh Diri (Suicidal Thoughts).

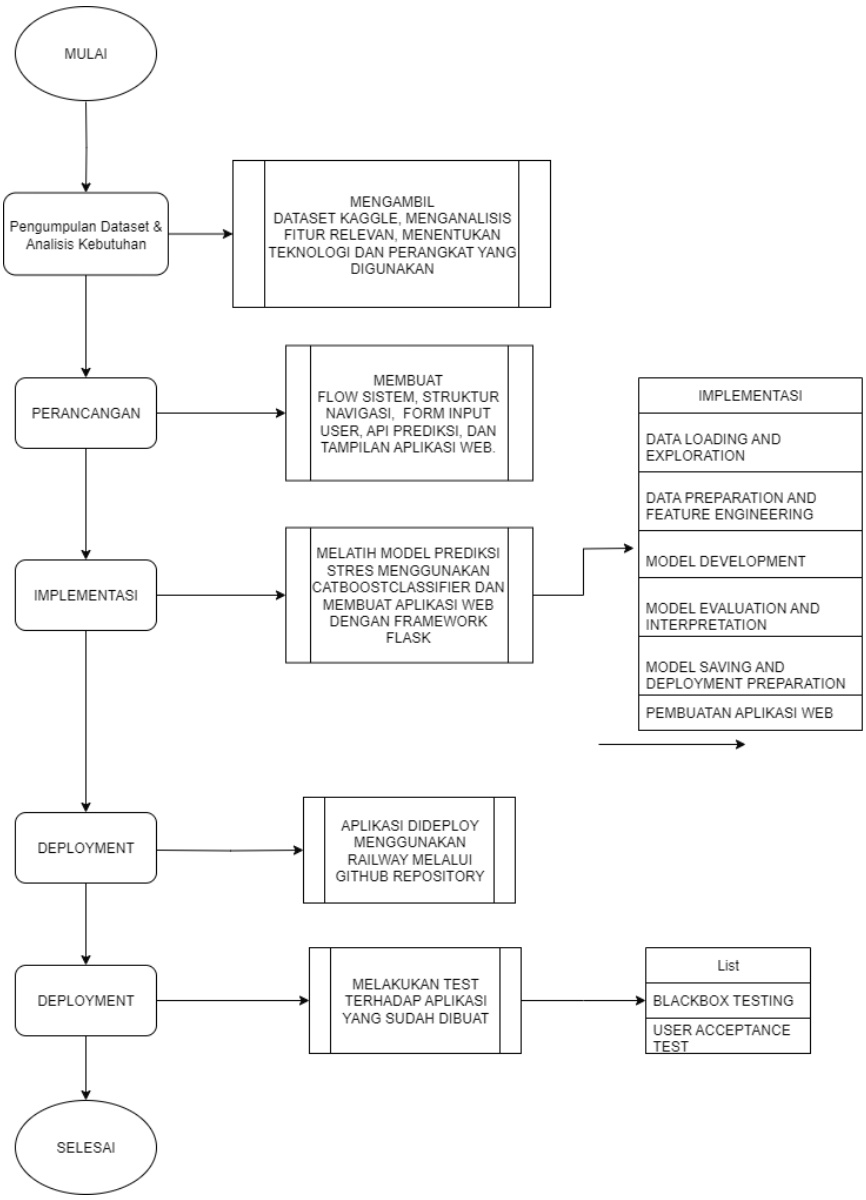
Algoritma yang digunakan adalah CatBoostClassifier, dipilih karena kemampuannya menangani data kategorikal dan numerik sekaligus tanpa perlu encoding eksplisit, serta performanya yang unggul pada tugas klasifikasi biner. Setelah preprocess, dataset dibagi ke dalam data latih (80 %) dan data uji (20 %) dengan stratifikasi target. Model CatBoost kemudian dioptimasi menggunakan RandomizedSearchCV untuk mencari kombinasi hyperparameter terbaik—meliputi jumlah iterasi (iterations), kedalaman pohon (depth), laju pembelajaran (learning_rate), dan regularisasi L2 (l2_leaf_reg) dengan tujuan meningkatkan AUC secara signifikan.

Selama pelatihan, proses optimasi dan evaluasi ditampilkan secara interaktif di notebook: setiap 100 iterasi CatBoost akan mengeluarkan progres, dan grafik ROC/AUC dapat terlihat secara real-time jika dijalankan dalam Jupyter Notebook atau Google Colab. Setelah model terlatih, performanya diukur secara kuantitatif menggunakan confusion matrix, classification report (precision, recall, F1-score), serta kurva ROC dan nilai AUC. Analisis feature importance juga diterapkan untuk mengidentifikasi fitur-fitur mana yang paling berkontribusi dalam prediksi.

Setelah validasi model memuaskan, hasil model di-serialisasi ke dalam file .cbm dan kemudian diintegrasikan ke dalam sebuah aplikasi web berbasis Flask. Melalui antarmuka yang menarik dan interaktif, mahasiswa dapat memasukkan data personal dan gaya hidup mereka (misal, usia, IPK, durasi tidur, tingkat tekanan akademik, dsb.), kemudian aplikasi akan mengembalikan nilai probabilitas risiko stres dalam bentuk persentase. Dengan begitu, diharapkan mahasiswa dapat mengetahui indikasi awal risiko stres secara cepat dan mandiri, serta mengambil langkah preventif seperti konseling, manajemen waktu, atau perbaikan pola tidur dan diet sesuai rekomendasi yang dihasilkan.

3.2 Metode dan Tahapan Penulisan

Metode yang digunakan dalam tugas akhir ini adalah System Development Life Cycle (SDLC), yaitu pendekatan terstruktur dalam proses pengembangan sistem informasi yang dilakukan secara sistematis, mulai dari tahap perencanaan hingga sistem siap digunakan. Penerapan SDLC bertujuan untuk memastikan bahwa setiap tahapan pengembangan sistem klasifikasi pneumonia berbasis web dapat dilakukan secara terarah, efisien, dan dapat dipertanggungjawabkan. Dalam penelitian ini, SDLC terdiri dari lima tahapan utama, yaitu: pengumpulan dataset dan analisis perangkat, perencanaan, implementasi, pengujian aplikasi (testing), dan deployment aplikasi. Rangkaian tahapan penelitian ini dapat dilihat pada Gambar 3.1.



Gambar 3.1 Tahapan Penulisan

Pada gambar 3.1 diatas menjelaskan tahapan-tahapan yang akan dilakukan dalam penelitian ini, yaitu pengumpulan dataset dan analisis kebutuhan, perancangan, implementasi, pengujian aplikasi, dan deployment. Penjelasan secara lengkap mengenai tahapan penulisan dapat dilihat pada sub bab selanjutnya.

3.2.1 Pengumpulan Dataset dan Analisis Kebutuhan Perangkat

Pada tahap Pengumpulan Dataset dan Analisis Kebutuhan Perangkat, dilakukan pengumpulan data yang relevan untuk mendukung proses

pengembangan sistem, baik berupa dataset utama yang akan digunakan dalam pelatihan model maupun data tambahan yang diperlukan untuk pengujian dan validasi. Dataset dapat diperoleh dari sumber terbuka (*open source*), atau dikumpulkan secara mandiri sesuai dengan tujuan sistem yang dikembangkan. Setelah itu, dilakukan analisis kebutuhan perangkat lunak dan perangkat keras untuk memastikan sistem dapat berjalan secara optimal. Analisis ini mencakup spesifikasi minimum perangkat keras maupun perangkat lunak, seperti jenis perangkat keras, kapasitas prosesor, RAM, penyimpanan, serta software pendukung seperti sistem operasi, library, dan framework yang dibutuhkan dalam proses pengembangan dan implementasi aplikasi. Berikut adalah rincian tentang pengumpulan dataset dan analisis kebutuhan perangkat.

3.2.2.1 Dataset Student Depression dari Kaggle

Dataset utama yang digunakan dalam penelitian ini diunduh dari platform Kaggle dengan judul *Student Depression Dataset* (https://www.kaggle.com/datasets/hopesb/student-depression-dataset/data). Dataset ini berisi sebanyak 27.901 entri hasil survei non-medis mahasiswa, dengan kolom-kolom sebagai berikut:

- Gender (Jenis Kelamin)
- Age (Usia)
- City (Kota) – akan dihilangkan setelah memuat data
- Profession (Status/Jenis Pekerjaan)
- CGPA (Indeks Prestasi Kumulatif dalam skala 0–4)
- Sleep Duration (Durasi Tidur dalam jam)
- Academic Pressure (Tekanan Akademik, skala 1–10)
- Work Pressure (Tekanan Kerja, skala 1–10)
- Financial Problem (Masalah Finansial; Yes/No)
- Financial Stress (Stres Finansial, skala 1–10)
- Health Issue (Masalah Kesehatan; Yes/No)
- Family History of Mental Illness (Riwayat Keluarga Gangguan Mental; Yes/No)

- Dietary Habits (Pola Diet; Good/Average/Poor)
- Have you ever had suicidal thoughts ? (Pernah Berpikir Bunuh Diri; Yes/No)

Proses pengumpulan data dimulai dengan mengunduh file Student Depression Dataset.csv ke dalam lingkungan Jupyter Notebook. Setelah itu, dilakukan verifikasi awal dengan menampilkan beberapa baris pertama untuk memastikan struktur kolom sesuai deskripsi. Kolom City kemudian dihapus karena tidak diperlukan dalam pemodelan.

3.2.2.2 Analisis Kebutuhan Teknologi dan Perangkat

Berdasarkan alur pemrosesan data dan pengembangan aplikasi, kebutuhan teknologi (perangkat lunak) untuk penelitian ini meliputi:

- Bahasa Pemrograman: Python (versi 3.7 ke atas)
- Library dan Framework Python:
    - pandas dan numpy untuk manipulasi data
    - scikit-learn untuk fungsi pembagian data (train_test_split), encoding, metrik evaluasi, dan RandomizedSearchCV
    - catboost untuk membangun dan melatih model CatBoostClassifier
    - matplotlib dan seaborn untuk visualisasi hasil eksplorasi dan evaluasi
    - shap untuk interpretasi model
    - joblib untuk menyimpan daftar kolom fitur dan model terlatih
- Framework Web:
    - Flask sebagai kerangka kerja (framework) untuk membangun aplikasi web yang dapat menerima input pengguna, memuat model yang sudah dilatih, dan menampilkan prediksi risiko stres
- Lingkungan Pengembangan:
    - Jupyter Notebook atau Google Colab untuk rangkaian eksplorasi, implementasi, dan evaluasi model
    - Visual Studio Code untuk pengembangan skrip Python dan Flask
- Data Repository:

- File Student Depression Dataset.csv yang disimpan di direktori proyek
- File catboost_depression_model.cbm dan model_columns.pkl sebagai keluaran (output) dari proses pelatihan model

Kebutuhan perangkat keras minimal yang direkomendasikan adalah komputer/laptop dengan prosesor multi-core (misalnya Intel i5 ke atas), RAM minimal 8 GB, dan ruang penyimpanan (hard disk/SSD) sekitar 10 GB untuk mengakomodasi instalasi library dan penyimpanan dataset serta model.

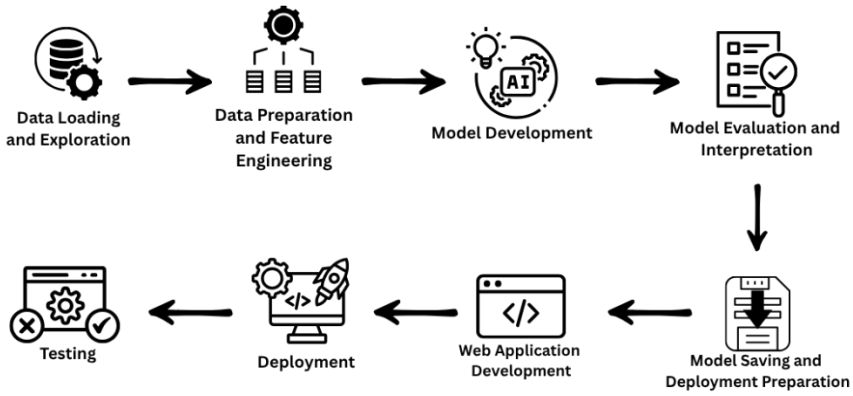
3.2.2 Perancangan

Pada tahap perancangan dilakukan serangkaian proses yang mencakup pembuatan alur kerja sistem, perancangan struktur navigasi halaman web, desain antarmuka pengguna, serta alur proses prediksi tingkat risiko stres pada mahasiswa. Tujuan dari tahap ini adalah untuk merancang sistem yang terstruktur, responsif, dan mudah digunakan oleh pengguna, sehingga mendukung proses deteksi stres secara mandiri dan informatif.

3.2.2.1 Alur Pembuatan Aplikasi

Tahapan alur pengembangan aplikasi yang dirancang dalam penelitian ini meliputi proses pemuatan dan pembersihan dataset, pembagian data, pelatihan model menggunakan algoritma CatBoostClassifier, evaluasi model dengan metrik kuantitatif, penyimpanan model terlatih, serta integrasi model ke dalam aplikasi web berbasis Flask. Selain itu, dirancang juga formulir input pengguna yang memuat berbagai fitur terkait gaya hidup dan kondisi psikologis mahasiswa. Aplikasi web kemudian akan menampilkan hasil prediksi berupa persentase risiko stres berdasarkan input yang diberikan.

Alur pembuatan aplikasi ini dapat dilihat pada Gambar 3.2.



Gambar 3.2 Alur Pembuatan Aplikasi

Berikut adalah penjelasan lengkap mengenai masing-masing alur pembuatan aplikasi pada *website* pendeteksi tingkat risiko stres pada mahasiswa.

1. ***Data Loading and Exploration***

   Pada tahap awal, dilakukan pemuatan dataset dengan nama Student Depression Dataset.csv ke dalam lingkungan analisis. Proses ini menjadi langkah pertama dalam memulai penelitian. Setelah data berhasil dimuat, dilakukan pemeriksaan struktur data secara umum untuk memastikan bahwa setiap kolom memiliki format dan nilai yang sesuai dengan dokumentasi. Salah satu langkah awal yang dilakukan adalah menghapus kolom yang dianggap tidak relevan, seperti kolom *City*, guna menyederhanakan analisis dan mengurangi beban komputasi pada tahap berikutnya.

   Selanjutnya, proses pembersihan data atau data cleaning dilakukan untuk meningkatkan kualitas data yang akan dianalisis. Dalam tahap ini, setiap kolom diperiksa tipe datanya dan dihitung jumlah nilai yang hilang. Untuk kolom bertipe numerik, nilai-nilai kosong diisi dengan menggunakan nilai median, sedangkan kolom bertipe kategorikal diisi menggunakan nilai modus. Hal ini dilakukan agar tidak perlu menghapus baris-baris data yang memiliki nilai kosong.

Tahap eksprlorasi data bertujuan memahami karakteristik dan distribusi data yang telah dibersihkan, termasuk visualisasi distribusi variabel target *Depression* dan analisis korelasi fitur numerik seperti *CGPA*, durasi tidur, dan tingkat stres. Fitur kategorikal seperti *Gender* dan *Academic Pressure* dianalisis melalui tabel silang dan plot distribusi, sementara outlier diperiksa menggunakan *boxplot* dan *scatter* plot untuk memastikan data tidak mengganggu pembangunan model.

2. ***Data Preparation and Feature Engineering***

Setelah proses eksplorasi awal, tahap persiapan data atau data wrangling dilakukan untuk memastikan bahwa data memiliki format yang sesuai dan siap digunakan oleh algoritma machine learning. Salah satu langkah penting dalam tahap ini adalah mengubah kolom target *Depression* dari format string berupa *Yes* dan *No* menjadi nilai numerik, yakni 1 untuk *Yes* dan 0 untuk *No*. Konversi ini dilakukan melalui proses pemetaan sederhana yang bertujuan untuk memudahkan proses klasifikasi selanjutnya. Selain itu, seluruh fitur kategorikal dalam dataset diidentifikasi dan dicatat agar bisa disesuaikan dengan kebutuhan model CatBoost, yang memiliki kemampuan untuk menangani fitur kategorikal secara otomatis tanpa memerlukan proses encoding manual yang kompleks.

Selanjutnya, proses feature engineering dilakukan untuk menambahkan informasi baru yang dapat meningkatkan kemampuan prediktif dari model. Salah satu fitur baru yang ditambahkan adalah Total_Pressure, yang dihitung dari penjumlahan nilai Academic Pressure dan Work Pressure. Fitur ini dirancang untuk menangkap tekanan yang dialami siswa secara keseluruhan. Selain itu, fitur Sleep_Quality juga dibentuk berdasarkan kategori dari durasi tidur. Jika siswa tidur kurang dari 5 jam, maka dikategorikan sebagai Poor, jika antara 5 hingga 8 jam maka dianggap Normal, dan jika lebih dari 8 jam dikategorikan sebagai Over. Transformasi ini bertujuan untuk menstandarkan data dan memperkaya informasi agar model dapat mengenali pola dengan lebih baik.

Pada tahap *data splitting*, dataset kemudian dibagi menjadi dua bagian, yakni data latih (train) dan data uji (test) dengan rasio pembagian sebesar 80:20. Pembagian ini dilakukan dengan teknik *stratified split*, yaitu dengan mempertahankan proporsi kelas target *Depression* agar tetap seimbang antara data latih dan data uji. Hal ini penting untuk menghindari bias model terhadap kelas mayoritas. Daftar nama kolom yang digunakan dalam proses pelatihan juga disimpan untuk memastikan bahwa format input tetap konsisten saat model digunakan dalam tahap evaluasi dan prediksi di akhir proses.

3. ***Model Development***

Pada tahap pelatihan model (*Model Training*), algoritma CatBoostClassifier digunakan karena kemampuannya dalam menangani fitur numerik dan kategorikal secara langsung tanpa memerlukan proses *encoding* yang kompleks. Model diinisialisasi dengan metrik Area Under the Curve (AUC) sebagai indikator utama performa, yang memungkinkan evaluasi kualitas pemisahan kelas oleh model. Proses pelatihan dilakukan dengan menggunakan data latih yang telah melalui tahap praproses, dan selanjutnya performa awal model diuji menggunakan data uji untuk memperoleh nilai AUC sebagai tolok ukur awal kemampuan generalisasi model terhadap data baru.

Pada tahap selanjutnya, yaitu *Model Optimization*, dilakukan penyetelan hyperparameter untuk meningkatkan akurasi dan kemampuan generalisasi model. Metode RandomizedSearchCV dipilih karena efisien dalam menjelajahi ruang parameter secara acak namun terkontrol. Parameter yang disetel mencakup jumlah iterasi (*iterations*), kedalaman pohon keputusan (*depth*), laju pembelajaran (*learning_rate*), serta koefisien regularisasi (*l2_leaf_reg*). Proses optimasi dilakukan dengan *cross-validation* pada data latih untuk menilai performa tiap kombinasi parameter. Setelah kombinasi terbaik ditemukan berdasarkan skor AUC tertinggi, model akhir dilatih ulang dengan parameter tersebut dan diuji kembali pada

data uji. Hasil pengujian akhir menunjukkan peningkatan stabilitas dan akurasi model dibandingkan sebelum optimasi dilakukan.

4. ***Model Evaluation and Interpretation***

Do not return selected text in word. Model yang telah dioptimalkan lalu dievaluasi pada data uji untuk menilai kemampuan klasifikasinya. Evaluasi model dilakukan secara komprehensif agar tidak hanya menghasilkan prediksi yang akurat, tetapi juga dapat dipahami secara interpretatif. Langkah awal evaluasi adalah menyusun confusion matrix yang menunjukkan distribusi prediksi benar dan salah untuk kelas *Depression* dan *No Depression*. Dari hasil matrix terlihat bahwa model lebih sering benar dalam memprediksi *Depression* (*True Positives*: 2892) dibanding *No Depression* (*True Negatives*: 1835), dengan sejumlah kesalahan pada *False Positives* (478) dan *False Negatives* (376).

Berdasarkan confusion matrix tersebut, classification report dihitung untuk mendapatkan metrik precision, recall, dan F1-score. Hasilnya menunjukkan bahwa model memiliki precision dan recall yang tinggi pada kelas "Depression" (masing-masing di atas 0.85 dan 0.88), yang berarti model cukup andal dalam mengenali mahasiswa yang mengalami depresi, serta minim dalam memberikan alarm palsu.

Selanjutnya, kemampuan model dalam membedakan dua kelas pada berbagai ambang batas dievaluasi menggunakan kurva ROC dan AUC (Area Under Curve). AUC sebesar 0.92 menunjukkan bahwa model memiliki performa klasifikasi yang sangat baik.

Untuk memahami kontribusi fitur terhadap prediksi, dilakukan visualisasi feature importance dari model *CatBoostClassifier*. Hasilnya menunjukkan bahwa fitur seperti *"Have you ever had suicidal thoughts?"*, *Financial Stress*, dan *Total_Pressure* merupakan yang paling berpengaruh dalam keputusan model. Sementara itu, fitur seperti *CGPA* atau *Degree* memiliki pengaruh relatif rendah.

Sebagai tahap interpretasi lanjutan, digunakan pendekatan SHAP (SHapley Additive exPlanations) untuk menggambarkan kontribusi tiap

fitur secara global dan lokal terhadap prediksi. Melalui summary plot, tampak bahwa nilai tinggi pada fitur seperti *suicidal thoughts* dan *financial stress* cenderung meningkatkan kemungkinan prediksi ke arah depresi, sementara nilai rendah pada *Age* atau *Work Pressure* bisa menurunkan skor prediksi tersebut. Dependence plot juga menunjukkan adanya interaksi antara fitur-fitur tertentu, misalnya antara tekanan akademik dan gender, di mana peningkatan tekanan lebih berdampak besar pada mahasiswa perempuan.

Kombinasi hasil dari confusion matrix, metrik evaluasi (precision/recall/F1), ROC AUC, feature importance, dan SHAP plot memberikan gambaran menyeluruh tentang performa dan karakteristik model dalam mendeteksi risiko depresi pada mahasiswa. Pendekatan ini tidak hanya menekankan akurasi model, namun juga transparansi dan interpretabilitasnya.

.

5. ***Model Saving and Deployment Preparation***

Setelah model terbaik diperoleh, langkah selanjutnya adalah tahap Model Saving, yang bertujuan untuk memastikan bahwa model dan konfigurasi fiturnya dapat digunakan kembali tanpa perlu melalui proses pelatihan ulang. Ini menjadi bagian penting dalam alur implementasi karena memungkinkan penggunaan model secara praktis dalam sistem produksi, seperti aplikasi web.

Model yang telah dilatih disimpan dalam format .cbm, yaitu format bawaan dari CatBoost. Sementara itu, daftar nama kolom fitur yang digunakan selama pelatihan juga disimpan secara terpisah dalam file .pkl menggunakan pustaka joblib. Penyimpanan ini dilakukan agar struktur input tetap konsisten saat model digunakan kembali untuk prediksi.

Kedua file ini, yaitu model dan fitur, menjadi komponen utama yang nantinya akan diintegrasikan ke dalam aplikasi web. Dengan demikian, aplikasi dapat memuat model secara langsung dan melakukan prediksi terhadap data pengguna secara real-time. Praktik ini juga mendukung

prinsip reproducibility, di mana model yang sama dapat dijalankan kembali dengan hasil prediksi yang konsisten.

6. ***Web Application Development***

Setelah model machine learning berhasil dipilih, dilatih, dan diekspor ke dalam format .cbm (CatBoost) beserta file daftar kolom fitur (model_columns.pkl), tahap selanjutnya adalah mengembangkan aplikasi web menggunakan framework Flask. Pada tahap ini, backend aplikasi dibuat untuk menerima input prediksi risiko stres dari pengguna melalui form yang tersedia di halaman utama. Ketika form disubmit, server akan memuat model CatBoost dan daftar kolom fitur, kemudian melakukan preprocessing data input agar sesuai dengan format yang digunakan saat pelatihan model. Data yang telah diproses kemudian dimasukkan ke model untuk menghasilkan probabilitas risiko stres, yang selanjutnya dikonversi ke dalam bentuk persentase agar mudah dipahami oleh pengguna. Hasil prediksi ini langsung ditampilkan di halaman web secara real-time, lengkap dengan kategori risiko dan saran yang sesuai. Seluruh proses ini memastikan bahwa prediksi yang diberikan konsisten, akurat, dan mudah diakses oleh pengguna.

7. ***Deployment***

Pada tahap deployment, website dirancang untuk dihosting menggunakan platform Railway. Railway dipilih karena menawarkan kemudahan dalam proses pengelolaan dan penyebaran aplikasi web berbasis Python, khususnya yang menggunakan framework Flask. Seluruh source code aplikasi, model machine learning, serta file pendukung lainnya dikumpulkan dalam satu folder project yang terorganisir dengan baik. Untuk memastikan seluruh dependensi Python dapat terinstal secara otomatis, disediakan file requirements.txt, sedangkan file Dockerfile digunakan untuk mendefinisikan langkah-langkah pembuatan image aplikasi secara otomatis. Railway yang mendukung integrasi dengan Docker memungkinkan proses deployment berjalan lebih efisien, konsisten, dan mudah direplikasi. Selain itu, platform ini juga menyediakan fitur

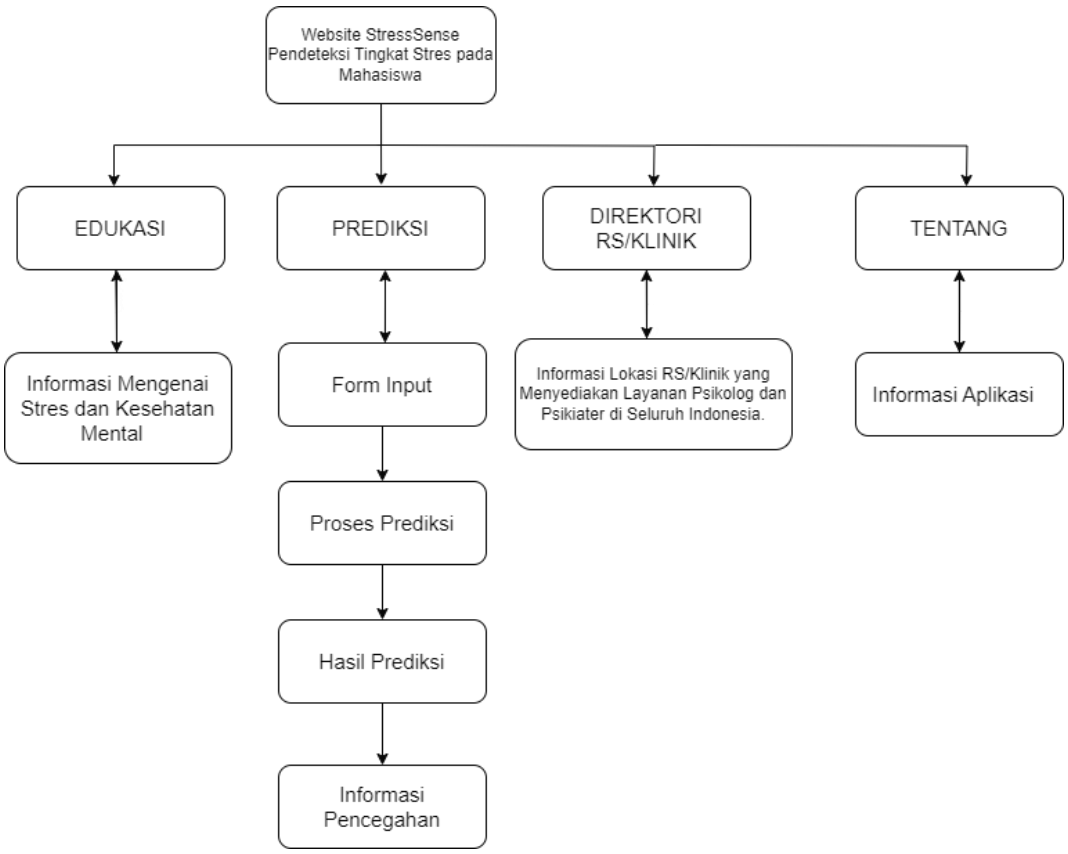
pengaturan environment variable, seperti PORT, yang dapat digunakan untuk menyesuaikan konfigurasi aplikasi saat dijalankan di lingkungan produksi. Dengan pendekatan ini, proses deployment aplikasi menjadi lebih sederhana, terstandarisasi, dan memudahkan pengelolaan serta pemeliharaan aplikasi di masa mendatang.

8. ***Testing***

Setelah aplikasi web selesai dikembangkan, tahap selanjutnya adalah melakukan pengujian untuk memastikan seluruh fitur berjalan dengan baik dan sesuai harapan. Pengujian dilakukan secara menyeluruh menggunakan *BlackBox* dan *User Acceptance Test*. Pengujian ini meliputi validasi hasil prediksi model dengan berbagai skenario input untuk memastikan akurasi dan konsistensi output yang dihasilkan. Selain itu, dilakukan pengujian pada antarmuka pengguna (user interface) untuk memastikan setiap elemen form, tampilan hasil prediksi, serta fitur reset dan navigasi dapat digunakan dengan mudah dan responsif oleh pengguna. Seluruh proses pengujian ini bertujuan untuk memastikan aplikasi tidak hanya memberikan hasil prediksi yang akurat, tetapi juga memberikan pengalaman pengguna yang baik, aman, dan andal sebelum akhirnya diimplementasikan secara luas.

3.2.2.2 Struktur Navigasi Aplikasi

Tujuan Struktur Navigasi dibuat adalah untuk mempermudah pengguna dalam menggunakan aplikasi serta mengetahui fitur-fitur dan informasi yang tersedia pada aplikasi. Struktur navigasi aplikasi dapat dilihat pada Gambar 3.3.



Gambar 3.3 Struktur Navigasi Aplikasi

Berikut adalah penjelasan mengenai masing-masing struktur navigasi aplikasi pada *website* pendeteksi penyakit daun padi.

1. Prediksi

   Halaman awal yang memperkenalkan aplikasi *StressSense* berisi penjelasan singkat mengenai fungsi utama aplikasi, yaitu membantu mahasiswa dalam mendeteksi risiko stres secara mandiri. Selain itu, halaman ini menyediakan navigasi ke fitur utama lainnya seperti Edukasi, Direktori RS/Klinik, dan Tentang Aplikasi. Pada laman Prediksi juga terdapat form untuk memasukkan data mahasiswa, seperti jenis kelamin, usia, tekanan akademik, tekanan kerja, pola tidur, dan faktor gaya hidup lainnya. Setelah data dikirim, sistem akan memproses input menggunakan model CatBoostClassifier dan menghasilkan estimasi risiko stres dalam bentuk persentase, serta memberikan informasi pencegahan berdasarkan hasil prediksi yang diperoleh.

2. Direktori RS/Klinik

   Halaman ini menyediakan daftar lokasi rumah sakit atau klinik di Indonesia yang menawarkan layanan psikologi dan psikiatri. Tujuannya adalah memberikan akses cepat bagi pengguna yang membutuhkan bantuan profesional setelah melihat hasil prediksi. Direktori ini membantu menjembatani pengguna menuju tindakan lanjutan yang lebih tepat.

3. Edukasi

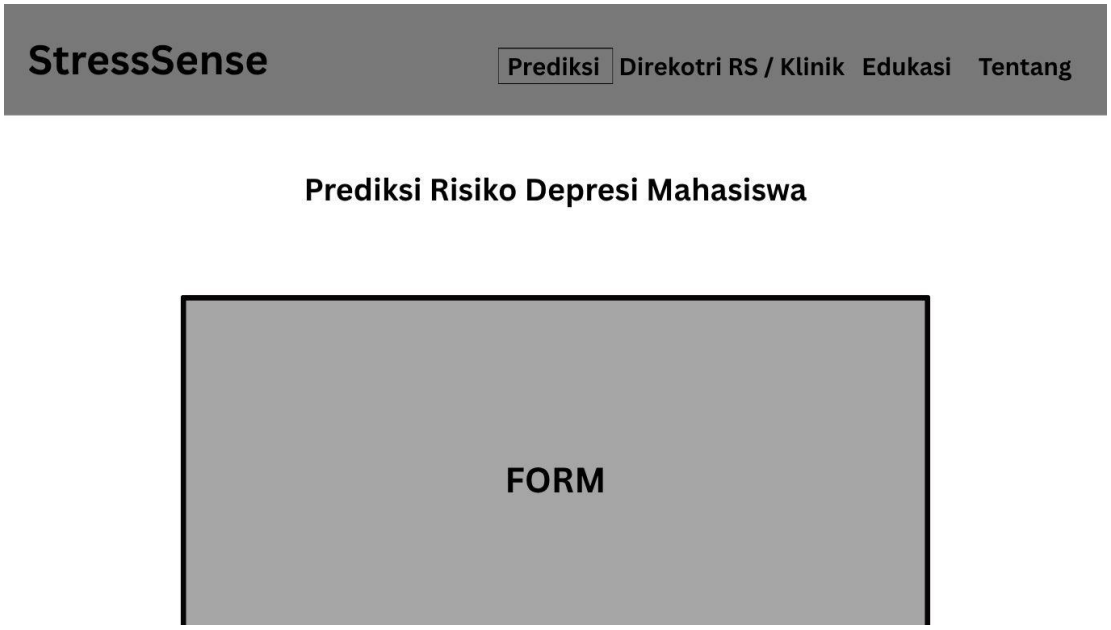
   Halaman ini berisi informasi penting mengenai stres dan kesehatan mental. Materi yang disediakan bertujuan untuk meningkatkan pemahaman pengguna terhadap penyebab, gejala, dan dampak stres, serta pentingnya menjaga kesehatan mental. Konten ini disajikan agar pengguna memiliki pengetahuan dasar sebelum atau sesudah menggunakan fitur prediksi.

4. Tentang

   Halaman ini menjelaskan secara singkat mengenai cara menggunakan aplikasi, latar belakang pengembangan aplikasi dan teknologi yang digunakan. Halaman ini juga dapat mencakup disclaimer bahwa hasil prediksi bersifat indikatif dan tidak menggantikan diagnosis medis resmi..

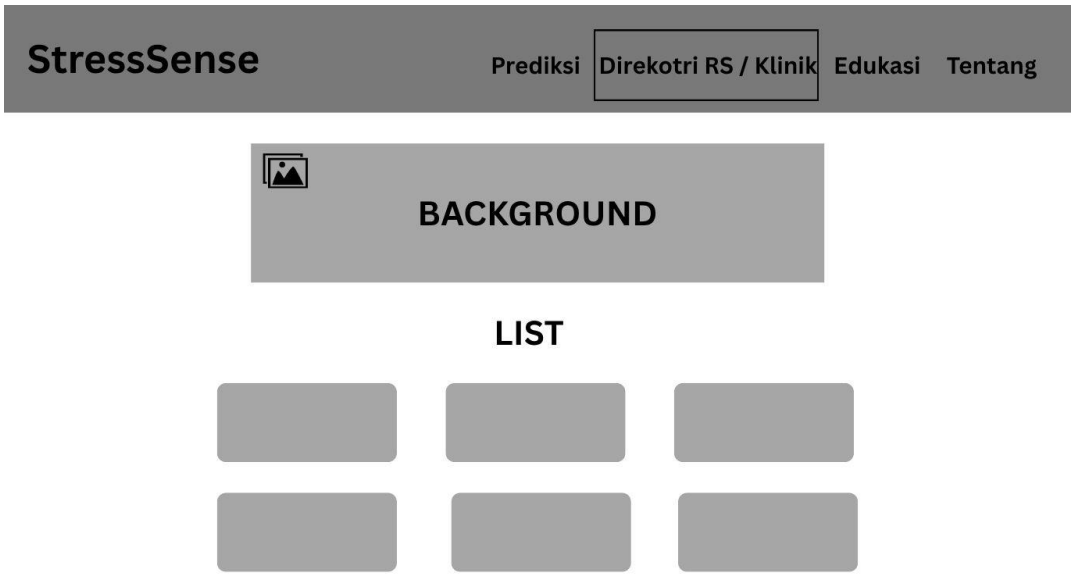
3.2.2.3 Desain Aplikasi

Desain Aplikasi terbagi menjadi 4 halaman yaitu halaman Prediksi, Direktori RS/Klinik, Edukasi, dan Tentang desain dibuat menggunakan Canva. Berikut adalah desain halaman pada *web* pendeteksi risiko stress pada mahasiswa. Desain aplikasi dapat dilihat pada Gambar 3.4 yang menjelaskan tentang halaman Prediksi, Gambar 3.5 tentang halaman Direktori RS/Klinik, Gambar 3.6 halaman edukasi, dan 3.7 adalah halaman tentang.



Gambar 3.4 Halaman Prediksi

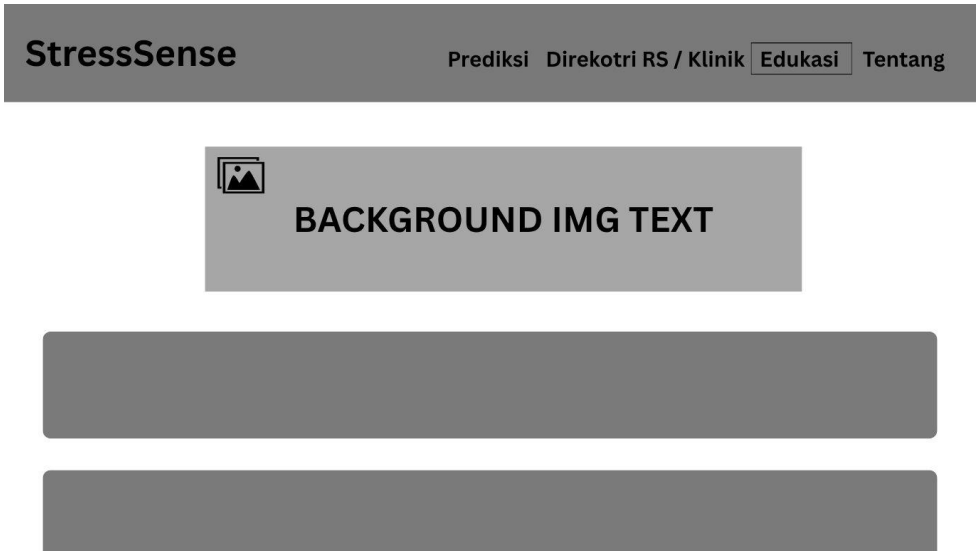
Desain halaman prediksi berisikan navigasi bar yang mengarahkan kepada halaman lain yaitu Edukasi, Direktori RS/Klinik, dan Tentang Aplikasi. Selain itu halaman beranda juga berisikan form untuk memprediksi persentase risiko stres mahasiswa dan menampilkan hasilnya.



Gambar 3.5 Halaman Direktori RS/Klinik

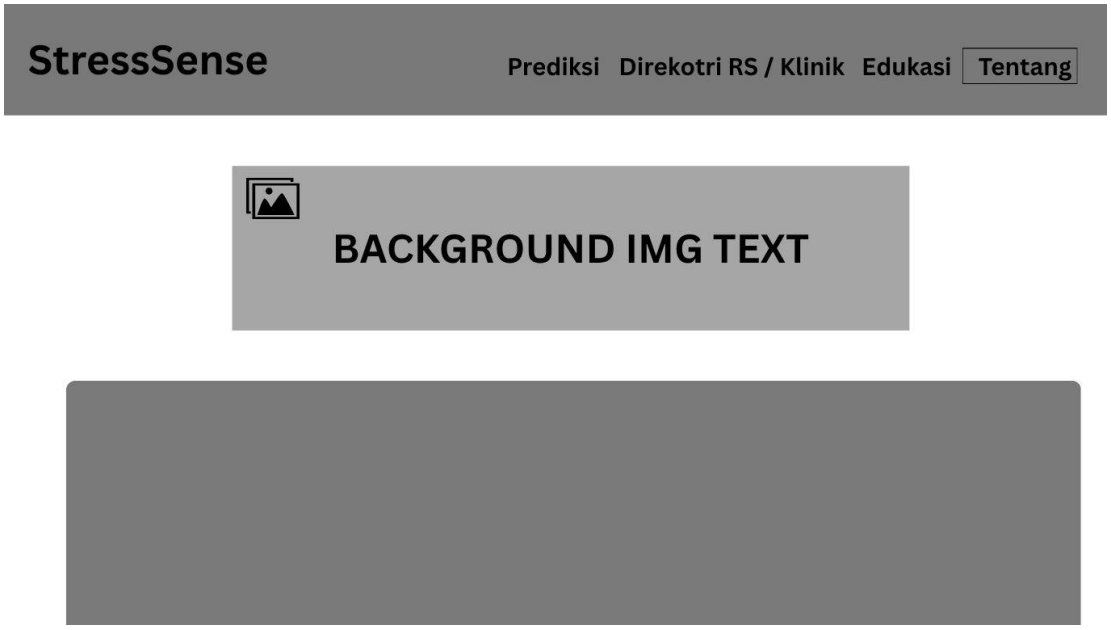
Selanjutnya pada halaman Direktori RS/Klinik berisikan navigasi bar yang mengarahkan kepada halaman lain. Lalu pada halaman ini terdapat daftar lokasi

rumah sakit atau klinik di Indonesia yang menawarkan layanan psikologi dan psikiatri beserta *hotline* yang bisa segera dihubungi



Gambar 3.6 Halaman Edukasi

Pada halaman Edukasi, berisikan navigasi bar yang mengarahkan ke halaman lain. Hal utama pada halaman edukasi ini adalah berisi informasi penting mengenai stres dan kesehatan mental.

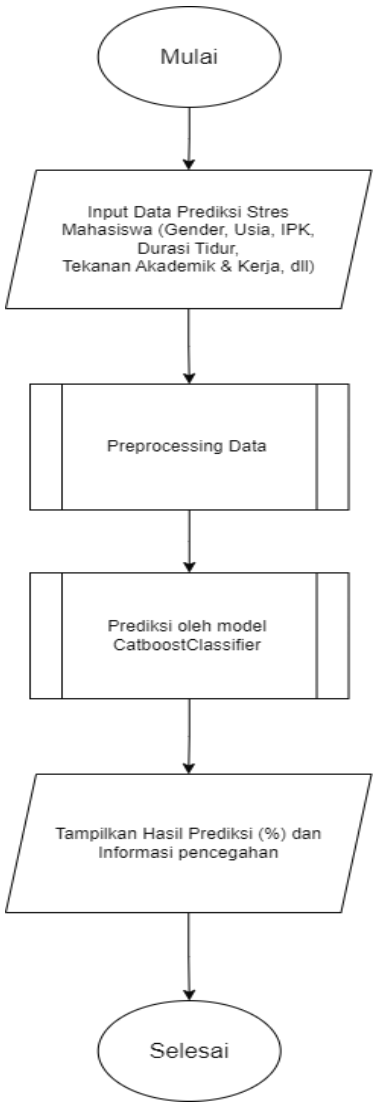


Gambar 3.7 Halaman Tentang

Pada halaman Tentang, berisikan navigasi bar yang mengarahkan ke halaman lain. Halaman ini juga memberikan penjelasan ringkas mengenai petunjuk penggunaan aplikasi, latar belakang pengembangannya, serta teknologi yang digunakan dalam proses prediksi.

3.2.2.4 Alur Proses Pendeteksi Risiko Stres

Pada tahap ini bertujuan untuk menunjukkan langkah-langkah yang harus dilakukan oleh pengguna dari awal hingga akhir dalam mengisi data diri dan gaya hidup melalui form input untuk kemudian memperoleh hasil prediksi tingkat risiko stres beserta informasi pencegahan yang disarankan. Alur proses prediksi stres pada mahasiswa dapat dilihat pada Gambar 3.8.



Gambar 3.8 Flowchart Pendeteksi Risiko Stres Pada Mahasiswa

Proses dimulai dari tahap Mulai, yang menjadi titik awal ketika pengguna mengakses aplikasi prediksi stres. Selanjutnya, pengguna diarahkan untuk mengisi data pribadi pada form input yang tersedia di halaman utama aplikasi. Data yang dimasukkan mencakup faktor-faktor yang memengaruhi tingkat stres mahasiswa seperti jenis kelamin, usia, IPK, tekanan akademik, tekanan kerja, durasi tidur, stres finansial, pola makan, riwayat keluarga dengan gangguan mental, serta pengalaman kehilangan minat dam pikiran bunuh diri.

Setelah semua data terkumpul, sistem akan melakukan tahap preprocessing data untuk memastikan format dan nilai input sesuai dengan kebutuhan model, termasuk perhitungan otomatis seperti total_pressure dan sleep_qualitu. Data yang

telah diproses selanjutnya dikirim ke model Machine Learning CatBoostClassifier yang telah dilatih sebelumnya. Model ini akan mengolah input untuk menghasilkan prediksi risiko stres dalam bentuk persentase.

Hasil prediksi ini kemudian ditampilkan kepada pengguna melalui tampilan yang informatif dan mudah dipahami. Sebagai pelengkap, sistem juga akan memberikan rekomendasi atau informasi pencegahan berdasarkan tingkat risiko yang terdeteksi, seperti pentingnya konsultasi psikologis, menjaga pola tidur, atau memperbaiki pola hidup sehat. Setelah seluruh proses selesai, alur sistem mencapai tahap Selesai, menandai bahwa pengguna telah menerima hasil prediksi dan solusi yang mudah dipahami.

4. HASIL DAN PEMBAHASAN

4.1 Data Loading and Exploration

Tahap awal dalam penelitian ini adalah *Data Loading and Exploration* yang bertujuan untuk memperoleh pemahaman awal terhadap dataset "Student Depression Dataset.csv". Proses dimulai dengan memuat data ke dalam struktur DataFrame, kemudian dilakukan pemeriksaan tipe data, distribusi variabel target, serta identifikasi nilai yang hilang. Beberapa transformasi awal juga dilakukan untuk menyesuaikan format data yang belum siap diproses. Tahap ini memberikan gambaran menyeluruh mengenai kondisi awal data yang akan digunakan, serta menjadi landasan untuk tahapan preprocessing dan pemodelan berikutnya

4.1.1 Data Loading

Pada tahap pertama, dataset Student Depression Dataset.csv dimuat ke dalam lingkungan Jupyter Notebook menggunakan fungsi pd.read_csv(). Setelah memverifikasi lima baris pertama data, kolom City yang tidak relevan dihapus dengan df.drop('City', axis=1) untuk menyederhanakan analisis.

```
import pandas as pd


try:
    df = pd.read_csv('Student Depression Dataset.csv')
    display(df.head())
except FileNotFoundError:
    print("Error: 'Student Depression Dataset.csv' not found. Please ensure
the file is in the correct directory or provide the full path.")
# Drop the 'City' column
df = df.drop('City', axis=1)
display(df.head())
```

	id	Gender	Age	Profession	Academic Pressure	Work Pressure	CGPA	Study Satisfaction	Job Satisfaction	Sleep Duration	Dietary Habits	Degree	ever had suicidal thoughts ?	Work/Study Hours	Financial Stress	History of Mental Illness	Depression
0	2	Male	33.0	Student	5.0	0.0	8.97	2.0	0.0	5-6 hours	Healthy	B.Pharm	Yes	3.0	1.0	No	1
1	8	Female	24.0	Student	2.0	0.0	5.90	5.0	0.0	5-6 hours	Moderate	BSc	No	3.0	2.0	Yes	0
2	26	Male	31.0	Student	3.0	0.0	7.03	5.0	0.0	Less than 5 hours	Healthy	BA	No	9.0	1.0	Yes	0
3	30	Female	28.0	Student	3.0	0.0	5.59	2.0	0.0	7-8 hours	Moderate	BCA	Yes	4.0	5.0	Yes	1
4	32	Female	25.0	Student	4.0	0.0	8.13	3.0	0.0	5-6 hours	Moderate	M.Tech	Yes	1.0	1.0	No	0

Gambar 4. 1 Data 5 baris pertama dari dataset

4.1.2 Data Cleaning

Selanjutnya, dilakukan pembersihan data dengan memeriksa tipe setiap kolom (df.dtypes) dan distribusi kelas target (df['Depression'].value_counts()). Semua nilai hilang pada kolom numerik diisi dengan median sedangkan kolom kategorikal diisi dengan modus. Khusus untuk kolom Sleep Duration, nilai seperti “5-6 hours” diekstrak ke angka terendah (misalnya menjadi 5.0) sebelum imputasi median. Setelah itu diverifikasi kembali bahwa tidak ada nilai hilang tersisa (df.isnull().sum()).

```
# Check data types and class balance
print("Data Types:\n", df.dtypes)
print("\nClass Balance (Depression_Status):\n", df['Depression'].value_counts())

# Handle missing values
for col in df.columns:
    if df[col].dtype == 'object':
        df[col] = df[col].fillna(df[col].mode()[0])
    elif col == 'Sleep Duration':
        try:
            df['Sleep Duration'] = df['Sleep Duration'].str.split('-').str[0].astype(float)
        except AttributeError:
            print("Sleep Duration column is already numeric.")
            pass
        df[col] = df[col].fillna(df[col].median())
```

```
    elif df[col].dtype in ['int64', 'float64']:
        df[col] = df[col].fillna(df[col].median())

# Verify imputation
print("\nMissing Values after Imputation:\n", df.isnull().sum())

# Data Type Check (Post-Imputation)
print("\nData Types after Imputation:\n", df.dtypes)
```

Langkah ini penting untuk memastikan bahwa data yang digunakan dalam pelatihan model tidak mengandung nilai kosong (*missing values*), karena sebagian besar algoritma pembelajaran mesin tidak dapat memproses data yang tidak lengkap. Pengisian menggunakan median untuk data numerik berguna agar tidak terlalu terpengaruh oleh pencilan (*outliers*), sementara pengisian dengan modus untuk data kategorikal bertujuan mempertahankan nilai yang paling umum atau representatif. Konversi nilai teks pada kolom *Sleep Duration* ke format numerik memastikan konsistensi tipe data agar dapat diolah lebih lanjut dalam tahap eksplorasi maupun pelatihan model.

Output dari proses data cleaning memberikan beberapa informasi penting terkait kondisi data sebelum dan sesudah dilakukan pembersihan. Bagian pertama yang ditampilkan adalah Data Types, yang menunjukkan tipe data awal dari setiap kolom. Informasi ini berguna untuk memahami format dari masing-masing fitur, apakah berupa numerik (seperti int64 atau float64) atau kategori/teks (object). Pemahaman tipe data ini menjadi penting karena akan menentukan metode yang digunakan pada tahap preprocessing berikutnya.

```
Data Types:
 id                                       int64
Gender                                   object
Age                                     float64
Profession                               object
Academic Pressure                       float64
Work Pressure                           float64
CGPA                                    float64
Study Satisfaction                      float64
Job Satisfaction                        float64
Sleep Duration                           object
Dietary Habits                           object
Degree                                   object
Have you ever had suicidal thoughts ?    object
Work/Study Hours                        float64
Financial Stress                        float64
Family History of Mental Illness         object
Depression                                int64
dtype: object

Class Balance (Depression_Status):
 Depression
1    16336
0    11565
Name: count, dtype: int64
```

Gambar 4. 2 Output Data Types dan Class Balance

Selanjutnya, bagian Class Balance (Depression_Status) menunjukkan distribusi nilai pada kolom target Depression. Output ini memperlihatkan bahwa kelas dengan label 1 (mengalami depresi) memiliki jumlah yang lebih banyak dibandingkan kelas 0 (tidak depresi), yaitu masing-masing sebanyak 16.336 dan 11.565 data. Informasi ini berguna untuk mendeteksi adanya class imbalance, yang dapat memengaruhi performa model klasifikasi dan perlu diperhatikan dalam tahap pemodelan.

```
Missing Values after Imputation:
 id                                       0
Gender                                    0
Age                                       0
Profession                                0
Academic Pressure                         0
Work Pressure                             0
CGPA                                      0
Study Satisfaction                        0
Job Satisfaction                          0
Sleep Duration                            0
Dietary Habits                            0
Degree                                    0
Have you ever had suicidal thoughts ?     0
Work/Study Hours                          0
Financial Stress                          0
Family History of Mental Illness          0
Depression                                0
dtype: int64
```

Gambar 4. 3 Output Missing Values after Imputation

Bagian Missing Values after Imputation menunjukkan bahwa semua nilai

kosong (NaN) pada dataset telah berhasil ditangani, ditandai dengan jumlah 0 pada seluruh kolom. Hal ini memastikan bahwa tidak ada lagi data yang hilang setelah proses imputasi dilakukan.

```
Data Types after Imputation:
 id                                          int64
Gender                                      object
Age                                        float64
Profession                                  object
Academic Pressure                          float64
Work Pressure                              float64
CGPA                                       float64
Study Satisfaction                         float64
Job Satisfaction                           float64
Sleep Duration                              object
Dietary Habits                              object
Degree                                      object
Have you ever had suicidal thoughts ?       object
Work/Study Hours                           float64
Financial Stress                           float64
Family History of Mental Illness            object
Depression                                   int64
dtype: object
```

Gambar 4. 4 Output Data Types after Imputation

Terakhir, bagian Data Types after Imputation menampilkan kembali tipe data dari setiap kolom setelah pembersihan. Tujuannya adalah untuk memastikan bahwa proses imputasi maupun transformasi tidak mengubah tipe data secara tidak diinginkan. Sebagai contoh, jika kolom Sleep Duration sebelumnya dalam format string atau teks, maka diharapkan sudah dikonversi ke tipe numerik agar dapat dianalisis lebih lanjut. Secara keseluruhan, output ini menegaskan bahwa tahap data cleaning telah selesai dilakukan dengan baik dan data siap untuk dianalisis lebih lanjut.

4.1.3 Data Exploratory

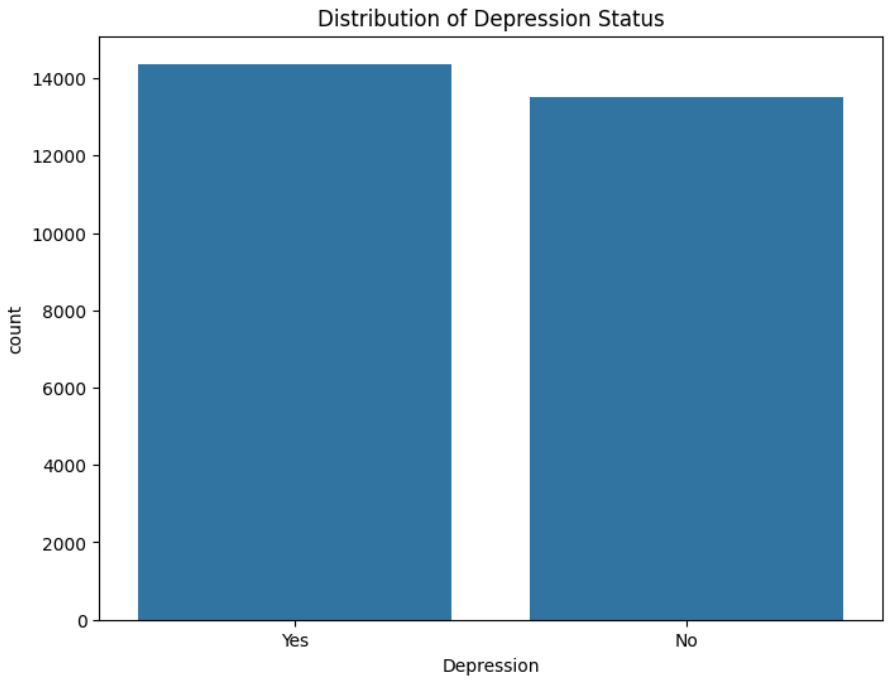
Tahap ini bertujuan untuk mengeksplorasi dataset dan mendapatkan wawasan visual mengenai distribusi variabel, hubungan antar fitur, dan potensi outlier. perpanjang bagian ini

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# 1. Target Variable Distribution
plt.figure(figsize=(8, 6))
sns.countplot(x='Depression', data=df)
plt.title('Distribution of Depression Status')
plt.show()

# 2. Correlation Matrix for Numerical Features
numerical_cols = df.select_dtypes(include=['number']).columns
correlation_matrix = df[numerical_cols].corr()
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of Numerical Features')
plt.show()

# 3. Relationship between Categorical Features and Target
categorical_cols = df.select_dtypes(include=['object']).columns
for col in categorical_cols:
    plt.figure(figsize=(10, 6))
    sns.countplot(x=col, hue='Depression', data=df)
    plt.title(f'Depression Status by {col}')
    plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
    plt.tight_layout() # Adjust layout to prevent labels from overlapping
    plt.show()

# 4. Outliers in 'CGPA'
plt.figure(figsize=(10, 6))
sns.boxplot(x='Depression', y='CGPA', data=df)
plt.title('CGPA Distribution with Depression Status')
plt.show()

plt.figure(figsize=(10, 6))
```

```
sns.scatterplot(x=df.index, y='CGPA', hue='Depression', data=df)
plt.title('CGPA Outliers Scatter Plot')
plt.show()
```

Langkah awal dalam eksplorasi data dilakukan dengan membuat plot hitungan (countplot) untuk kolom Depression menggunakan pustaka Seaborn. Visualisasi ini menunjukkan distribusi dari variabel target, yaitu jumlah mahasiswa yang tergolong mengalami depresi (Depression = 1) dan yang tidak (Depression = 0). Analisis ini penting untuk memahami apakah data memiliki distribusi kelas yang seimbang atau tidak.

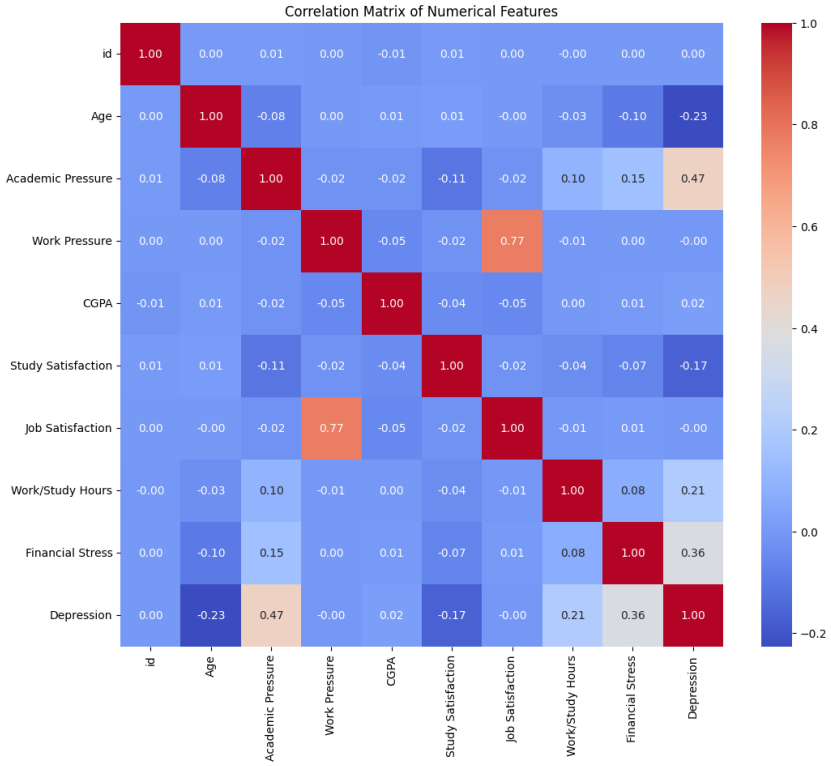


Gambar 4. 5 ***Distribussion of Depression Status***

Dari plot hitungan yang dihasilkan, terlihat jelas bahwa jumlah sampel untuk kedua kelas, yaitu 'Yes' (mengalami depresi) dan 'No' (tidak mengalami depresi), memiliki jumlah yang hampir setara. Distribusi yang relatif seimbang ini mengindikasikan bahwa dataset tidak mengalami masalah ketidakseimbangan kelas (class imbalance) yang signifikan, yang merupakan kondisi ideal untuk melatih

model klasifikasi tanpa memerlukan teknik penanganan *imbalance* yang kompleks secara berlebihan.

Selanjutnya, dilakukan perhitungan matriks korelasi untuk semua fitur numerik yang kemudian divisualisasikan dalam bentuk heatmap. Heatmap ini menampilkan nilai koefisien korelasi antar fitur numerik dengan pewarnaan yang mencerminkan kekuatan dan arah hubungan linier antar fitur. Adanya anotasi angka pada tiap sel membantu dalam mengidentifikasi fitur mana saja yang memiliki hubungan kuat satu sama lain, termasuk terhadap variabel target.



Gambar 4. 6 Heatmap Korelasi Fitur Numerik

Berdasarkan heatmap korelasi fitur numerik setelah feature engineering dan penghapusan beberapa kolom, terlihat beberapa hubungan penting. Academic Pressure menunjukkan korelasi positif yang paling kuat dengan Depression (≈ 0.47), mengindikasikan bahwa tekanan akademik merupakan faktor penting yang terkait dengan depresi pada dataset ini. Total_Pressure, yang merupakan kombinasi dari Academic Pressure dan Work Pressure, juga memiliki korelasi positif yang kuat dengan Depression (≈ 0.47), hampir sama dengan Academic Pressure saja,

menunjukkan bahwa Work Pressure dalam dataset ini mungkin tidak memiliki dampak tambahan yang signifikan terhadap depresi ketika dikombinasikan dengan Academic Pressure. Financial Stress juga berkorelasi positif sedang dengan Depression (≈ 0.36), menegaskan bahwa masalah finansial memiliki hubungan yang cukup kuat dengan kemungkinan depresi. Work/Study Hours menunjukkan korelasi positif yang lebih lemah dengan Depression (≈ 0.21). Sementara itu, Age memiliki korelasi negatif yang lemah dengan Depression (≈ -0.23), menyiratkan bahwa usia yang lebih tua sedikit terkait dengan kemungkinan depresi yang lebih rendah dalam sampel ini. Fitur lain seperti CGPA, Study Satisfaction, Job Satisfaction, dan Sleep Duration menunjukkan korelasi yang sangat lemah atau mendekati nol dengan Depression, menandakan bahwa dalam model linier sederhana, fitur-fitur ini mungkin kurang prediktif terhadap depresi dibandingkan faktor tekanan dan finansial. Analisis korelasi ini mengkonfirmasi bahwa tekanan akademik, tekanan gabungan (akademik dan pekerjaan), dan stres finansial adalah faktor-faktor kunci yang perlu mendapat perhatian dalam pemodelan prediksi depresi.Selain fitur numerik, fitur kategorikal juga dianalisis dengan pendekatan visual. Untuk setiap kolom kategorikal, dibuat plot hitungan yang dikelompokkan berdasarkan status depresi. Visualisasi ini menunjukkan distribusi masing-masing kategori dalam kaitannya dengan kondisi depresi mahasiswa, memberikan wawasan mengenai bagaimana prevalensi depresi berbeda di antara kategori-kategori tertentu, misalnya berdasarkan jenis kelamin, status pernikahan, atau pola tidur. Label pada sumbu x diputar untuk meningkatkan keterbacaan pada kategori yang panjang. Terakhir, untuk mendeteksi kemungkinan outlier pada fitur *CGPA*, digunakan *box plot* dan *scatter plot*. *Box plot* memperlihatkan distribusi nilai *CGPA* untuk setiap kelas depresi lengkap dengan median, kuartil, serta potensi outlier. Sementara itu, *scatter plot* menampilkan sebaran tiap data terhadap *CGPA* dan status depresinya untuk memperjelas pola dan outlier secara visual.

Secara keseluruhan, berbagai teknik visualisasi ini memberikan gambaran menyeluruh terhadap karakteristik dataset, baik dari segi distribusi kelas, korelasi antar fitur, maupun potensi pengaruh fitur-fitur terhadap status depresi. Temuan dari eksplorasi ini sangat penting untuk menjadi dasar dalam tahap pemodelan

selanjutnya.

4.2 Data Preparation and Feature Engineering

Pada bagian *Data preparation and Feature engineering*, fokus utama adalah mempersiapkan data mentah menjadi format yang siap digunakan untuk melatih model *machine learning* dan menciptakan fitur-fitur baru yang dapat meningkatkan kinerja model.

4.2.1 Data Wrangling

Pertama, di sub-bagian Data wrangling, dilakukan langkah awal persiapan data. Variabel target, yaitu kolom 'Depression', diubah dari nilai kategorikal menjadi numerik (0 atau 1) menggunakan LabelEncoder dari pustaka sklearn.preprocessing. Ini penting karena sebagian besar algoritma machine learning memerlukan target dalam bentuk numerik. Setelah itu, kode mengidentifikasi kolom-kolom lain yang bertipe 'object' (string) sebagai fitur kategorikal. Hasil dari langkah ini adalah daftar nama kolom kategorikal yang akan digunakan oleh model CatBoost, yang dapat menangani fitur kategorikal secara langsung.

```
[ ] from sklearn.preprocessing import LabelEncoder

    # 1. Label encode the 'Depression' column
    le = LabelEncoder()
    df['Depression'] = le.fit_transform(df['Depression'])

    # 2. Identify categorical features (excluding the target variable)
    categorical_features = [col for col in df.columns if df[col].dtype == 'object' and col != 'Depression']
    print(categorical_features)

    ler', 'Profession', 'Sleep Duration', 'Dietary Habits', 'Degree', 'Have you ever had suicidal thoughts ?', 'Family History of Mental Illness']
```

Gambar 4. 7 Kode dan Output Data Wrangler

Selanjutnya, kode tersebut mencari semua kolom bertipe *object* selain kolom target *Depression*, yang berarti kolom-kolom tersebut berisi data kategorikal seperti string atau teks. Daftar yang dihasilkan berisi nama-nama fitur kategorikal seperti *Gender*, *Profession*, *Sleep Duration*, *Dietary Habits*, *Degree*, *Have you ever had suicidal thoughts ?*, dan *Family History of Mental Illness*. Identifikasi ini berguna terutama jika menggunakan model seperti *CatBoost*, yang mampu menangani data kategorikal secara langsung tanpa perlu encoding manual seperti

one-hot encoding.

4.2.2 Feature Engineering

Selanjutnya, di sub-bagian *Feature Engineering*, dilakukan pembuatan fitur-fitur baru berdasarkan kolom yang sudah ada untuk memperkaya informasi yang dapat digunakan dalam pemodelan.

```
import pandas as pd

# Convert 'Sleep Duration' to numeric, handling errors
def convert_to_numeric(duration):
    try:
        if isinstance(duration, str):
            return float(duration.split('-')[0].split()[0]) #Extract first number
        elif isinstance(duration, (int, float)):
            return float(duration)
        else:
            return None  # Handle other data types
    except (ValueError, AttributeError, IndexError):
        return None

df['Sleep Duration'] = df['Sleep Duration'].apply(convert_to_numeric)

# Recalculate Sleep_Quality using the corrected numeric 'Sleep Duration'
def categorize_sleep(duration):
  if pd.isnull(duration):
    return "Unknown"
  if duration < 5:
    return 'Poor'
  elif 5 <= duration <= 8:
    return 'Normal'
```

```
  else:
    return 'Over'

df['Sleep_Quality'] = df['Sleep Duration'].apply(categorize_sleep)

display(df.head())
```

Fitur pertama yang dibuat adalah *Total_Pressure*, yang merupakan hasil penjumlahan dari *Academic Pressure* dan *Work Pressure*. Tujuannya adalah untuk menangkap total beban tekanan yang dialami oleh mahasiswa dalam konteks akademik maupun pekerjaan. Fitur kedua adalah *Sleep_Quality*, yang mengkategorikan durasi tidur menjadi tiga kelas, yaitu *Poor*, *Normal*, atau *Over*. Untuk menghasilkan fitur ini, digunakan fungsi kustom *convert_to_numeric()* yang bertugas mengubah nilai dalam kolom *Sleep Duration*—yang awalnya dalam format string seperti *5-6 hours*—menjadi angka desimal yang representatif, seperti 5.0 atau 7.0. Ini penting karena kolom tersebut awalnya memiliki format yang bervariasi, sehingga perlu distandarisasi terlebih dahulu sebelum dilakukan kategorisasi.

Setelah proses konversi ke nilai numerik, dilakukan pemetaan kualitas tidur dengan fungsi *categorize_sleep()* berdasarkan aturan tertentu: durasi kurang dari 5 jam dikategorikan sebagai *Poor*, antara 5 hingga 8 jam sebagai *Normal*, dan lebih dari 8 jam sebagai *Over*. Jika terdapat nilai yang hilang, maka kualitas tidur diberi label *Unknown*. Proses ini juga memastikan bahwa nilai-nilai tidak valid dan format string yang tidak sesuai dapat ditangani dengan baik. Output dari tahap ini adalah DataFrame yang telah diperbarui dan menampilkan dua kolom baru: *Total_Pressure* dan *Sleep_Quality*, yang siap digunakan dalam proses analisis lebih lanjut. Kehadiran dua fitur ini memperkaya dimensi data dan diharapkan dapat

meningkatkan performa model dalam memprediksi gejala depresi.

Output dari kode *feature engineering* menampilkan lima baris pertama *DataFrame* yang sudah diperbarui, menunjukkan penambahan dua kolom baru: *Total_Pressure*, yang merupakan penjumlahan *Academic Pressure* dan *Work*

	id	Gender	Age	Profession	Academic Pressure	Work Pressure	CGPA	Study Satisfaction	Job Satisfaction	Sleep Duration	Dietary Habits	Degree	Have you ever had suicidal thoughts ?	Work/Study Hours	Financial Stress	Family History of Mental Illness	Depression	Total_Pressure	Sleep_Quality
0	2	Male	33.0	Student	5.0	0.0	8.97	2.0	0.0	5.0	Healthy	B.Pharm	Yes	3.0	1.0	No	1	5.0	Normal
1	8	Female	24.0	Student	2.0	0.0	5.90	5.0	0.0	5.0	Moderate	BSc	No	3.0	2.0	Yes	0	2.0	Normal
2	26	Male	31.0	Student	3.0	0.0	7.03	5.0	0.0	NaN	Healthy	BA	No	9.0	1.0	Yes	0	3.0	Unknown
3	30	Female	28.0	Student	3.0	0.0	5.59	2.0	0.0	7.0	Moderate	BCA	Yes	4.0	5.0	Yes	1	3.0	Normal
4	32	Female	25.0	Student	4.0	0.0	8.13	3.0	0.0	5.0	Moderate	M.Tech	Yes	1.0	1.0	No	0	4.0	Normal

Gambar 4. 8 Output Feature Engineering

Pressure, serta *Sleep_Quality*, yang mengkategorikan durasi tidur menjadi *Poor*, *Normal*, *Over*, atau *Unknown*. Tampilan awal *DataFrame* ini berfungsi sebagai verifikasi visual bahwa fitur-fitur baru tersebut telah berhasil dibuat dan ditambahkan ke dataset dengan nilai-nilai yang sesuai, siap untuk digunakan dalam tahap pemodelan selanjutnya.

4.2.3 Data Splitting

Tahapan data splitting bertujuan untuk membagi dataset yang telah melalui proses pembersihan menjadi dua bagian utama, yaitu training set (set pelatihan) dan testing set (set pengujian). Training set digunakan untuk melatih model machine learning, sedangkan testing set digunakan untuk menguji kinerja model pada data yang belum pernah dilihat sebelumnya, sehingga dapat memberikan estimasi akurasi model di dunia nyata. Proses ini dilakukan menggunakan fungsi train_test_split dari pustaka sklearn.model_selection, sebagaimana ditunjukkan pada kode berikut.

```
from sklearn.model_selection import train_test_split


# Define features (X) and target (y)
X = df.drop('Depression', axis=1)
y = df['Depression']
```

```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

Dalam kode tersebut, terlebih dahulu didefinisikan fitur (*X*) dan target (*y*), di mana *X* merupakan semua kolom kecuali *'Depression'*, sedangkan *y* berisi kolom *'Depression'*. Fungsi *train_test_split* kemudian membagi *X* dan *y* menjadi empat bagian: *X_train*, *X_test*, *y_train*, dan *y_test*. Parameter *test_size=0.2* menunjukkan bahwa 20% data dialokasikan sebagai data pengujian, sementara 80% sisanya untuk pelatihan. Parameter *random_state=42* menjamin konsistensi pembagian saat kode dijalankan berulang, dan *stratify=y* memastikan distribusi kelas target tetap seimbang pada kedua set data. Dari total 27.901 baris data setelah pembersihan, diperoleh sekitar 22.320 baris untuk pelatihan dan 5.581 baris untuk pengujian. Selain itu, untuk menjaga konsistensi kolom pada data baru yang akan digunakan saat prediksi, daftar nama kolom pelatihan disimpan dalam file *model_columns.pkl* menggunakan pustaka *joblib*, seperti yang tampak pada gambar berikut.

```
import joblib

# Simpan kolom yang digunakan untuk pelatihan
joblib.dump(X_train.columns.tolist(), "model_columns.pkl")
```

```
['model_columns.pkl']
```

Gambar 4. 9 Kode dan Output Penyimpanan Kolom

Dengan pembagian ini, diperoleh keempat variabel penting (X_train, X_test, y_train, dan y_test) yang siap digunakan dalam tahap pelatihan dan evaluasi model. Meskipun tidak menampilkan output visual berupa tabel secara langsung, keberhasilan proses ini dapat dikonfirmasi melalui isi file model_columns.pkl dan penjelasan lanjutan pada sel markdown berikutnya yang mendeskripsikan hasil

pembagian data tersebut.

4.3 Model Development

Tahapan Model Development berfokus pada pembangunan dan pelatihan model machine learning untuk memprediksi status depresi mahasiswa. Langkah pertama dalam proses ini adalah memastikan bahwa pustaka CatBoost telah terinstal, karena pustaka ini dirancang khusus untuk menangani data kategorikal tanpa perlu proses encoding manual. Model yang digunakan adalah CatBoostClassifier, yang diinisialisasi dengan menentukan daftar fitur kategorikal berdasarkan hasil eksplorasi sebelumnya. Parameter eval_metric='AUC' digunakan untuk menilai kinerja model selama pelatihan dengan metrik Area Under the Curve (AUC), sedangkan verbose=100 memastikan log pelatihan ditampilkan setiap 100 iterasi. Model dilatih menggunakan data pelatihan X_train dan y_train, dan saat pelatihan berlangsung, CatBoost menampilkan log perkembangan, termasuk estimasi waktu dan metrik evaluasi.

```python
# Identify categorical features
categorical_features = ['Gender', 'Profession', 'Dietary Habits', 'D

# Inisialisasi model TANPA plot di konstruktor
model = CatBoostClassifier(
    cat_features=categorical_features,
    eval_metric='AUC',
    verbose=100, # Print progress every 100 iterations
    random_state=42
)

# Latih model dengan plot=True di sini
model.fit(X_train, y_train, plot=True)

# Prediksi probabilitas
y_pred_proba = model.predict_proba(X_test)[:, 1]

# Prediksi kelas untuk menghitung akurasi
y_pred = model.predict(X_test) # Define y_pred here

# Hitung AUC & Akurasi
auc_score = roc_auc_score(y_test, y_pred_proba)
accuracy = accuracy_score(y_test, y_pred) # Now y_pred is defined

print(f"AUC Score: {auc_score:.4f}")
print(f"Accuracy Score: {accuracy:.4f}")
```

```
Learning rate set to 0.0388
0:      total: 51.1ms   remaining: 51.1s
100:    total: 6.76s    remaining: 1m
200:    total: 9.16s    remaining: 36.4s
300:    total: 11.5s    remaining: 26.6s
400:    total: 13.8s    remaining: 20.7s
500:    total: 16.5s    remaining: 16.4s
600:    total: 19.7s    remaining: 13.1s
700:    total: 22.2s    remaining: 9.46s
800:    total: 24.7s    remaining: 6.13s
900:    total: 27.2s    remaining: 2.98s
999:    total: 30.9s    remaining: 0us
AUC Score: 0.9180
Accuracy Score: 0.8452
```

Gambar 4. 10 Kode dan Output Model Training

Setelah model selesai dilatih yang digunakan untuk menghasilkan prediksi probabilitas terhadap data pengujian (X_test), dan skor AUC dihitung menggunakan fungsi roc_auc_score. Skor AUC dan akurasi yang diperoleh dari model awal adalah 0.91 dan juga 0.84 , menunjukkan bahwa model memiliki kemampuan klasifikasi yang cukup sebelum optimasi dilakukan. Namun, untuk meningkatkan performa lebih lanjut, dilakukan proses hyperparameter tuning menggunakan RandomizedSearchCV. Proses ini mencari kombinasi terbaik dari parameter seperti iterations, depth, learning_rate, dan l2_leaf_reg dengan cara mencoba 100 kombinasi acak secara efisien melalui 6-fold cross-validation.

Parameter grid yang digunakan cukup luas, mencakup variasi yang realistis untuk setiap parameter penting. Model yang digunakan dalam proses tuning masih menggunakan CatBoost dan memanfaatkan fitur kategorikal secara langsung.

```
from sklearn.model_selection import RandomizedSearchCV
from catboost import CatBoostClassifier

# Define the parameter grid
param_grid = {
    'iterations': range(100, 1001),
    'depth': range(4, 11),
    'learning_rate': [round(x  0.01, 2) for x in range(1, 31)],
    'l2_leaf_reg': range(1, 11),
}

# Identify categorical features dynamically from X_train columns
categorical_features = [col for col in X_train.columns if X_train[col].dtype ==
'object']
print("Categorical features identified for RandomizedSearchCV:",
categorical_features)

# Initialize CatBoostClassifier
# Use verbose=0 to suppress output during the search
model = CatBoostClassifier(cat_features=categorical_features,
eval_metric='AUC', verbose=0, random_state=42)

# Initialize RandomizedSearchCV
# Use a smaller n_iter for faster execution during debugging, can increase later
random_search = RandomizedSearchCV(
    estimator=model,
    param_distributions=param_grid,
    scoring='roc_auc',
```

```
    n_iter=100,
    cv=5,
    verbose=1, # Show progress during search
    random_state=42,
    n_jobs=-1 # Use all available cores
)

print("Starting RandomizedSearchCV...")
# Fit RandomizedSearchCV
random_search.fit(X_train, y_train)

print("\nRandomizedSearchCV finished.")

# Print the best hyperparameters and AUC score
print("Best Hyperparameters:", random_search.best_params_)
print("Best AUC Score:", random_search.best_score_)

# Store the best estimator
best_model = random_search.best_estimator_

print("\nBest model stored in 'best_model' variable.")
```

```
Categorical features identified for RandomizedSearchCV: ['Sleep_Quality']
Starting RandomizedSearchCV with n_iter=100 and expanded ranges...
Fitting 6 folds for each of 100 candidates, totalling 600 fits

RandomizedSearchCV finished.
Best Hyperparameters: {'learning_rate': 0.125, 'l2_leaf_reg': 7, 'iterations': 154, 'depth': 4}
Best AUC Score: 0.9539632814834853

Best model stored in 'best_model' variable.
```

Gambar 4. 11 Output Model Optimization

Setelah proses pencarian selesai, RandomizedSearchCV memberikan hasil hyperparameter terbaik yang ditemukan, yaitu: learning_rate=0.125,

l2_leaf_reg=7, iterations=154, dan depth=4. Dengan kombinasi ini, model berhasil mencapai skor AUC terbaik sebesar 0.95, lebih tinggi dari model awal. Model terbaik ini kemudian disimpan dalam variabel best_model dan akan digunakan untuk tahap evaluasi lanjutan maupun prediksi data baru. Tahapan ini menunjukkan pentingnya kombinasi pelatihan awal dan tuning untuk menghasilkan model prediktif yang andal dan optimal dalam mengidentifikasi risiko depresi berdasarkan fitur-fitur yang tersedia.

4.4 Model Evaluation and Interpretation

Bagian Model Evaluation and Interpretation bertujuan untuk menilai seberapa baik kinerja model CatBoost yang telah dioptimalkan dan memahami faktor - faktor kunci yang berkontribusi pada prediksi model.

4.4.1 Model Evaluation

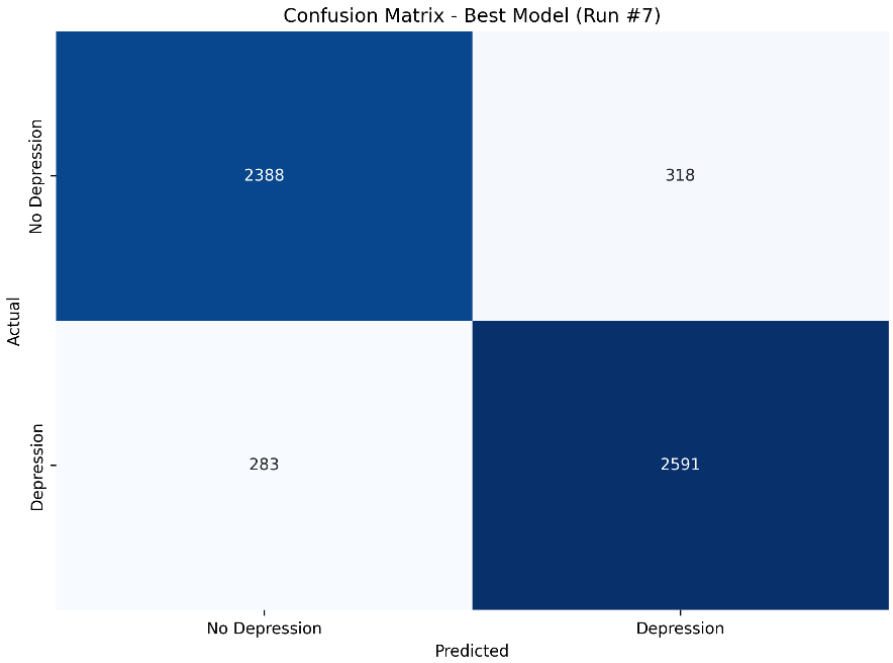
Tahapan evaluasi model bertujuan untuk menilai seberapa baik model CatBoostClassifier yang telah dioptimalkan (disimpan dalam variabel best_model) dalam memprediksi depresi berdasarkan data uji (X_test). Evaluasi dimulai dengan menghasilkan probabilitas prediksi untuk kelas positif ('Depression') menggunakan predict_proba, yang disimpan dalam y_pred_proba. Probabilitas ini diperlukan untuk menghitung ROC curve dan skor AUC yang merepresentasikan performa model dalam berbagai ambang keputusan (threshold). Selain itu, prediksi kelas juga dihitung menggunakan best_model.predict(X_test) sebagai dasar perhitungan metrik klasifikasi lainnya.

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report, roc_curve,
auc
import seaborn as sns
import numpy as np

# Predict probabilities on the test set
y_pred_proba = best_model.predict_proba(X_test)[:, 1]
```

```
# Confusion Matrix
cm = confusion_matrix(y_test, best_model.predict(X_test))
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False,
            xticklabels=['No Depression', 'Depression'],
            yticklabels=['No Depression', 'Depression'])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

# Classification Report
print(classification_report(y_test, best_model.predict(X_test)))

# ROC Curve and AUC
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =
{roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, 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')
plt.legend(loc="lower right")
plt.show()
```

```
# Feature Importance
feature_importance = best_model.feature_importances_
sorted_idx = np.argsort(feature_importance)
fig = plt.figure(figsize=(12, 6))
plt.barh(range(len(sorted_idx)), feature_importance[sorted_idx], align='center')
plt.yticks(range(len(sorted_idx)), np.array(X_test.columns)[sorted_idx])
plt.title('Feature Importance Scores')
plt.show()
```

Model yang telah dioptimasi dievaluasi menggunakan metrik evaluasi klasifikasi untuk mengukur kinerjanya dalam membedakan mahasiswa yang mengalami depresi dan yang tidak. Evaluasi ini mencakup pengukuran confusion matrix, precision, recall, F1-score, dan akurasi. Seluruh metrik dihitung berdasarkan hasil prediksi model terhadap data uji, sebagaimana ditampilkan dalam gambar confusion matrix berikut.



Gambar 4. 12 Hasil Confusion Matrix

Gambar confusion matrix di atas menunjukkan performa model dalam

memprediksi status depresi mahasiswa (apakah mengalami depresi atau tidak). Berdasarkan output tersebut:

- TP (True Positive): 2591. Model dengan benar memprediksi 2591 mahasiswa yang sebenarnya mengalami depresi (Aktual 1) sebagai mengalami depresi (Prediksi 1).
- TN (True Negative): 2388. Model dengan benar memprediksi 2388 mahasiswa yang sebenarnya tidak mengalami depresi (Aktual 0) sebagai tidak mengalami depresi (Prediksi 0).
- FP (False Positive): 318. Model salah memprediksi 318 mahasiswa yang sebenarnya tidak mengalami depresi (Aktual 0) sebagai mengalami depresi (Prediksi 1). Ini adalah kesalahan Tipe I.
- FN (False Negative): 283. Model salah memprediksi 283 mahasiswa yang sebenarnya mengalami depresi (Aktual 1) sebagai tidak mengalami depresi (Prediksi 0). Ini adalah kesalahan Tipe II.

Dari confusion matrix ini, dapat dihitung metrik utama: precision, recall, dan F1-score melalui classification_report.

	precision	recall	f1-score	support
0	0.89	0.88	0.89	2706
1	0.89	0.90	0.89	2874
accuracy			0.89	5580
macro avg	0.89	0.89	0.89	5580
weighted avg	0.89	0.89	0.89	5580

Gambar 4. 13 Classification Report

Dari gambar output classification report di atas, terlihat performa model dalam mengklasifikasikan dua kelas, yaitu No Depression (kelas 0) dan Depression (kelas 1). Nilai-nilai seperti precision, recall, dan f1-score ditampilkan secara lengkap untuk masing-masing kelas, beserta nilai rata-rata (macro avg dan weighted avg). Untuk memahami makna dan cara memperoleh nilai-nilai tersebut, dilakukan

perhitungan manual menggunakan rumus evaluasi klasifikasi sebagai berikut:

4.6.1 Precision

Presisi mengukur seberapa akurat prediksi positif model. Rumusnya adalah:

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

- Untuk Kelas Depression

$$P_1 = \frac{2591}{2591 + 318} = \frac{2591}{2909} \approx 0{,}89$$

- Untuk Kelas No Depression

$$P_0 = \frac{2388}{2388 + 283} = \frac{2388}{2671} \approx 0.89$$

4.6.2 Recall

Recall mengukur seberapa banyak data positif yang berhasil diprediksi dengan benar. Rumus recall adalah sebagai berikut::

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

- Untuk Kelas Depression

$$R_1 = \frac{2591}{2591 + 283} = \frac{2591}{2874} \approx 0.90$$

- Untuk Kelas No Depression

$$R_0 = \frac{2388}{2388 + 318} = \frac{2388}{2706} \approx 0.88$$

4.6.3 F1-Score

Recall mengukur seberapa banyak data positif yang berhasil diprediksi dengan benar. Rumus recall adalah sebagai berikut:

$$F1 = 2 \times \frac{P \times R}{P + R}$$

- Untuk Kelas Depression (1)

$$F1_1 = 2 \times \frac{0.89 \times 0.90}{0.89 + 0.90} = 2 \times \frac{0.801}{1.79} \approx 0.89$$

- Untuk Kelas No Depression (0)

$$F1_0 = 2 \times \frac{0.89 \times 0.88}{0.89 + 0.88} = 2 \times \frac{0.7832}{1.77} \approx 0.89$$

4.6.4 Accuracy

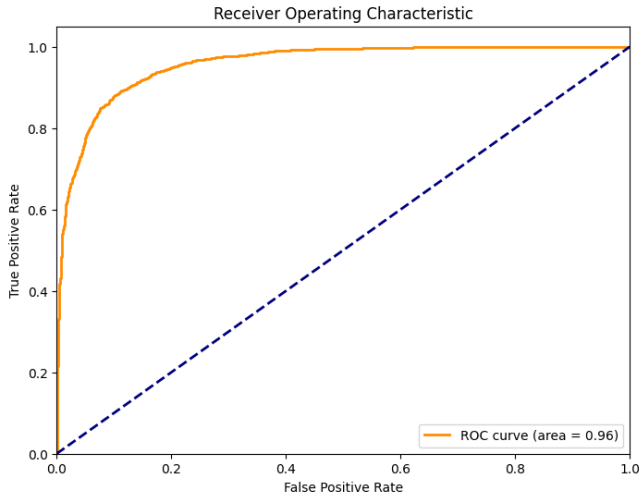
Akurasi mengukur proporsi prediksi yang benar terhadap seluruh data.

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

$$\text{Accuracy} = \frac{2591 + 2388}{2591 + 2388 + 318 + 283} = \frac{4979}{5580} \approx 0.89$$

Hasil evaluasi menunjukkan bahwa performa model dalam mengklasifikasikan data depresi cukup baik, dengan peningkatan akurasi yang signifikan. Sebelumnya, akurasi model hanya sebesar 0,84, namun setelah dilakukan evaluasi dan peningkatan, akurasi meningkat menjadi 0,89. Nilai precision dan recall pada kelas Depression juga tinggi, masing-masing sebesar 0,89 dan 0,90, yang menunjukkan bahwa model sangat efektif dalam mendeteksi kasus depresi. Hal ini penting dalam konteks medis atau psikologis karena mendeteksi depresi secara tepat lebih krusial daripada memprediksi orang yang tidak depresi.

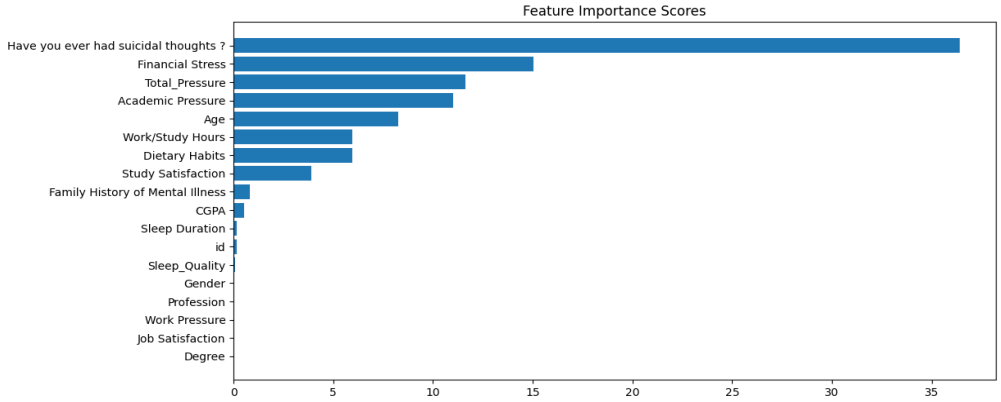
Sebagai pelengkap analisis performa klasifikasi, langkah berikutnya adalah mengevaluasi kemampuan diskriminatif model secara keseluruhan dengan menggunakan Receiver Operating Characteristic (ROC) curve dan menghitung Area Under the Curve (AUC).



Gambar 4. 14 Kurva ROC

ROC curve diplot dengan menghubungkan False Positive Rate (FPR) dan True Positive Rate (TPR) pada berbagai threshold, dan AUC (Area Under Curve) sebesar 0.96 memperlihatkan kemampuan sangat baik model dalam membedakan mahasiswa dengan dan tanpa depresi.

Selain metrik ROC-AUC yang menilai kemampuan diskriminatif model secara menyeluruh, analisis feature importance selanjutnya dilakukan untuk mengidentifikasi sejauh mana setiap fitur memberikan kontribusi terhadap keputusan prediksi.



Gambar 4. 15 Feature Importances Scores

Hasil feature importance dari CatBoostClassifier menunjukkan bahwa "Have you ever had suicidal thoughts?", Financial Stress, dan Total_Pressure adalah tiga faktor teratas yang paling memengaruhi prediksi, menegaskan

pentingnya indikator psikologis, tekanan emosional, dan kondisi finansial dalam menentukan risiko. Fitur-fitur seperti CGPA dan Degree berada di peringkat bawah dengan kontribusi yang sangat kecil, kemungkinan karena pengaruhnya terhadap hasil sudah terserap oleh variabel lain yang lebih relevan atau memiliki hubungan tidak langsung terhadap target. Temuan ini menyoroti bahwa model lebih mengutamakan faktor psikologis dan tekanan dibandingkan atribut akademik atau demografis murni.

4.4.2 Shapley Additive Explanation

Selanjutnya, digunakan metode SHAP (Shapley Additive Explanations) untuk menginterpretasikan hasil prediksi model CatBoost yang telah dibangun. SHAP merupakan pendekatan berbasis teori permainan yang menjelaskan kontribusi masing-masing fitur terhadap prediksi model, baik pada level individu (setiap baris data) maupun secara global. Tidak seperti Feature Importance biasa yang hanya menilai kontribusi rata-rata suatu fitur, SHAP dapat memberikan wawasan yang lebih rinci, termasuk bagaimana arah dan besar pengaruh suatu nilai fitur terhadap prediksi model.

```
# --- Interpretasi Model dengan SHAP ---
# Analisis kontribusi fitur dan interaksi, khususnya terhadap Gender

import shap
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

print("🔍 Menghitung nilai SHAP dari model.")

try:
    # Use the best_model trained with RandomizedSearchCV
    explainer = shap.TreeExplainer(best_model, data=X_train)
```

```
    # Calculate SHAP values for the test data
    # For binary classification (0/1), TreeExplainer returns a 2D array (n_samples, n_features)
    # This array's values correspond to the SHAP values for the positive class (class 1)
    shap_values = explainer.shap_values(X_test)

    # Verify the shape is 2D
    if len(shap_values.shape) == 2:
        shap_values_class_1 = shap_values
        print("✅ SHAP values berhasil dihitung (bentuk 2D).")
    elif isinstance(shap_values, list) and len(shap_values) > 1 and len(shap_values[1].shape) == 2:
        # Handle cases where shap_values is a list, take the positive class (index 1)
        shap_values_class_1 = shap_values[1]
        print("✅ SHAP values berhasil dihitung (dari list, bentuk 2D).")
    elif len(shap_values.shape) == 3 and shap_values.shape[2] > 1:
        # Handle cases where shap_values is a 3D array (n_samples, n_features, n_classes), take the positive class slice
        shap_values_class_1 = shap_values[:, :, 1]
        print("✅ SHAP values berhasil dihitung (dari 3D array, bentuk 2D).")
    else:
        raise ValueError(f"Unexpected SHAP values shape: {shap_values.shape}")

except Exception as e:
    print(f"❌ Gagal menghitung SHAP dengan background data: {e}, mencoba ulang tanpa background...")
    try:
        explainer = shap.TreeExplainer(best_model)
        shap_values = explainer.shap_values(X_test)
```

```
        # Verify the shape is 2D
        if len(shap_values.shape) == 2:
            shap_values_class_1 = shap_values
            print("✅ SHAP values berhasil dihitung (tanpa background, bentuk 2D).")
        elif isinstance(shap_values, list) and len(shap_values) > 1 and len(shap_values[1].shape) == 2:
            # Handle cases where shap_values is a list, take the positive class (index 1)
            shap_values_class_1 = shap_values[1]
            print("✅ SHAP values berhasil dihitung (tanpa background, dari list, bentuk 2D).")
        elif len(shap_values.shape) == 3 and shap_values.shape[2] > 1:
            # Handle cases where shap_values is a 3D array (n_samples, n_features, n_classes), take the positive class slice
            shap_values_class_1 = shap_values[:, :, 1]
            print("✅ SHAP values berhasil dihitung (tanpa background, dari 3D array, bentuk 2D).")
        else:
            raise ValueError(f"Unexpected SHAP values shape after retry: {shap_values.shape}")

    except Exception as e2:
        print(f"❌ Gagal total menghitung SHAP: {e2}")
        shap_values_class_1 = None # Ensure it's None if both attempts fail

# --- Visualisasi SHAP (Jika berhasil dihitung dan bentuknya benar) ---
if shap_values_class_1 is not None and len(shap_values_class_1.shape) == 2:
    print(f"\nFinal SHAP values shape for plotting:
```

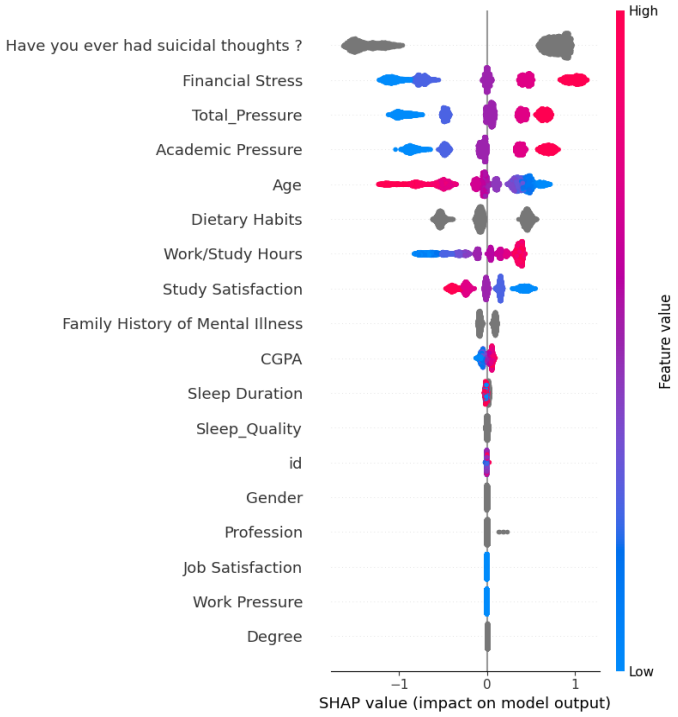
```
{shap_values_class_1.shape}")

    print("\n📊 SHAP Summary Plot (Semua fitur)")
    shap.summary_plot(shap_values_class_1, X_test)

    print("\n📊 SHAP Bar Plot (Rata-rata absolute SHAP)")
    shap.summary_plot(shap_values_class_1, X_test, plot_type="bar")

    # --- Dependence & Interaksi ---
    # ... (rest of the plotting code)
    for fitur in ['CGPA', 'Total_Pressure']:
        if fitur in X_test.columns:
            print(f"\n📈 Dependence Plot: {fitur}")
            shap.dependence_plot(fitur, shap_values_class_1, X_test)

    for fitur in ['Work Pressure', 'Sleep Duration']:
        if fitur in X_test.columns and 'Gender' in X_test.columns:
            print(f"\n📈 Interaksi {fitur} terhadap Gender")
                        shap.dependence_plot(fitur,    shap_values_class_1,    X_test,
interaction_index='Gender')

    # --- SHAP per Gender ---
    print("\n🧠 SHAP Summary Berdasarkan Gender")

    X_test_male = X_test[X_test['Gender'] == 'Male']
    X_test_female = X_test[X_test['Gender'] == 'Female']

    # Recalculate SHAP values for gender subsets to ensure correct shapes
    # This uses the explainer calculated above
    if not X_test_male.empty:
```

```
        print("\n👦 SHAP Summary Plot untuk Laki-laki")
        try:
            shap_values_male = explainer.shap_values(X_test_male)
            if len(shap_values_male.shape) == 2:
                shap.summary_plot(shap_values_male, X_test_male)
            elif isinstance(shap_values_male, list) and len(shap_values_male) > 1 and len(shap_values_male[1].shape) == 2:
                shap.summary_plot(shap_values_male[1], X_test_male)
            elif len(shap_values_male.shape) == 3 and shap_values_male.sha)pe[2] > 1:
                shap.summary_plot(shap_values_male[:, :, 1], X_test_male
            else:
                print(f"Warning: Unexpected SHAP shape for male subset: {shap_values_male.shape}")
        except Exception as e:
            print(f"Error plotting SHAP for male subset: {e}")

    if not X_test_female.empty:
        print("\n👧 SHAP Summary Plot untuk Perempuan")
        try:
            shap_values_female = explainer.shap_values(X_test_female)
            if len(shap_values_female.shape) == 2:
                shap.summary_plot(shap_values_female, X_test_female)
            elif isinstance(shap_values_female, list) and len(shap_values_female) > 1 and len(shap_values_female[1].shape) == 2:
                shap.summary_plot(shap_values_female[1], X_test_female)
            elif len(shap_values_female.shape) == 3 and shap_values_female.shape[2] > 1:
                shap.summary_plot(shap_values_female[:, :, 1], X_test_female)
            else:
```

```
                print(f"Warning: Unexpected SHAP shape for female subset: {shap_values_female.shape}")
        except Exception as e:
            print(f"Error plotting SHAP for female subset: {e}")

else:
    print("⚠️ SHAP values tidak berhasil dihitung atau memiliki bentuk yang salah. Lewati interpretasi visual.")
```

Setelah nilai SHAP berhasil dihitung, dilakukan visualisasi SHAP Summary Plot untuk semua fitur. Plot ini berbentuk *dot plot* yang menggambarkan sebaran nilai SHAP untuk setiap fitur terhadap seluruh data uji. Semakin jauh titik dari garis nol pada sumbu X, semakin besar pengaruh fitur terhadap prediksi. Warna titik menunjukkan nilai asli fitur (biru: rendah, merah: tinggi), sehingga kita bisa melihat apakah nilai tinggi atau rendah dari suatu fitur meningkatkan atau menurunkan kemungkinan depresi. Visualisasi ini memberi gambaran menyeluruh tentang fitur-fitur apa saja yang paling mempengaruhi hasil prediksi model.

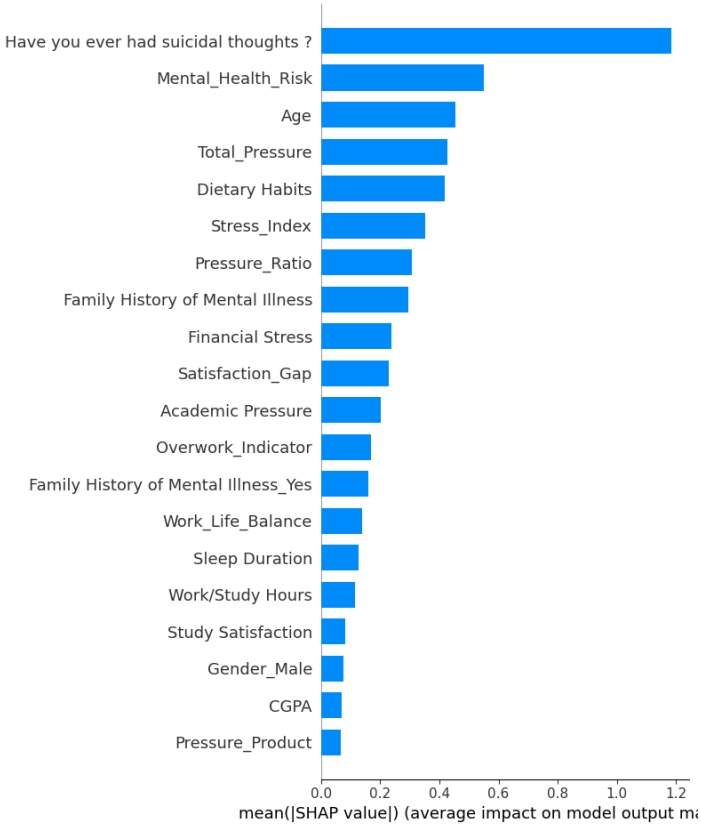
Proses dimulai dengan inisialisasi objek TreeExplainer dari pustaka shap, menggunakan model best_model yang telah dilatih. Sebagai data latar belakang (background), digunakan data pelatihan (X_train) untuk membantu perhitungan baseline nilai prediksi. Namun, jika perhitungan dengan background gagal karena kendala teknis atau kompatibilitas bentuk data, maka proses akan diulang tanpa data latar belakang, agar tetap dapat menghitung nilai SHAP. SHAP value yang dihitung fokus pada kelas positif (Depression = 1), dan hasilnya dipastikan berbentuk dua dimensi untuk memudahkan proses visualisasi.



Gambar 4. 16 Output SHAP Summary Plot

SHAP Summary Plot adalah alat visualisasi yang digunakan untuk memahami pengaruh masing-masing fitur terhadap prediksi model dalam konteks analisis data. Plot ini menampilkan setiap observasi sebagai titik, di mana posisi horizontal menunjukkan nilai SHAP yang merepresentasikan besarnya dan arah pengaruh fitur terhadap probabilitas terjadinya depresi. Nilai SHAP positif menunjukkan bahwa fitur tersebut meningkatkan kemungkinan prediksi ke arah "Depresi", sementara nilai negatif menunjukkan pengaruh ke arah "Tidak Depresi". Warna titik, dengan merah menunjukkan nilai fitur tinggi dan biru menunjukkan nilai fitur rendah, membantu memperjelas hubungan antara nilai fitur dan pengaruhnya terhadap prediksi. Fitur diurutkan dari atas ke bawah berdasarkan rata-rata magnitudo SHAP, sehingga fitur yang paling berpengaruh secara global berada di bagian atas plot. Fitur seperti "Have you ever had suicidal thoughts?", "Mental_Health_Risk", dan "Age" menunjukkan pengaruh terbesar, tidak hanya dari nilai SHAP rata-rata yang tinggi tetapi juga dari pola konsisten di mana nilai fitur yang tinggi cenderung meningkatkan probabilitas depresi. Selain menampilkan fitur penting, summary plot juga memperlihatkan variabilitas efek

antar individu dan potensi interaksi non-linier antar fitur, yang dapat membantu dalam memahami kompleksitas prediksi dan pengaruh fitur secara lebih mendalam.



Gambar 4. 17 SHAP Bar Plot

Melengkapi summary plot, SHAP Bar Plot menyajikan ringkasan global dengan menampilkan rata-rata nilai absolut SHAP untuk setiap fitur yang diurutkan dari yang paling penting hingga yang kurang penting. Panjang setiap batang menunjukkan mean(|SHAP value|), yaitu seberapa besar kontribusi rata-rata sebuah fitur dalam mengubah skor prediksi di seluruh dataset pengujian, sehingga plot ini memberikan gambaran cepat tentang prioritas fitur untuk keperluan feature engineering atau kebijakan intervensi. Karena plot menggunakan nilai absolut, ia hanya menggambarkan besarnya pengaruh fitur secara keseluruhan dan bukan arah pengaruhnya; untuk mengetahui apakah suatu fitur cenderung menaikkan atau menurunkan risiko perlu merujuk kembali ke SHAP Summary Plot. Pada kasus ini, bar plot menguatkan temuan summary plot, yakni indikator psikologis dan tekanan seperti riwayat pemikiran bunuh diri, indeks risiko mental, dan total tekanan

menempati peringkat teratas sebagai kontributor utama, sedangkan atribut seperti CGPA atau jenjang pendidikan memiliki dampak rata-rata yang relatif kecil terhadap prediksi model. Perlu dicatat bahwa SHAP menjelaskan apa yang model gunakan untuk membuat prediksi dan membantu interpretabilitas, namun bukan bukti hubungan sebab-akibat, sehingga rekomendasi intervensi klinis tetap memerlukan studi lanjutan dan penilaian dari tenaga kesehatan profesional.

4.5 Model Saving

Tahap Model saving adalah langkah penting untuk menyimpan model *machine learning* yang telah dilatih dan dioptimalkan ke dalam sebuah file, sehingga model tersebut dapat digunakan kembali di masa mendatang tanpa perlu melatihnya dari awal. Ini sangat berguna untuk deployment model, inferensi pada data baru, atau sekadar menyimpan hasil kerja.

```
import joblib
best_model.save_model("catboost_depression_model.cbm")
# Simpan kolom yang digunakan untuk pelatihan
joblib.dump(X_train.columns.tolist(), "model_columns.pkl")
```

```
['model_columns.pkl']
```

Gambar 4. 18 Kode Model Saving

Kode pada bagian ini menggunakan metode .save_model() yang merupakan bawaan dari objek model CatBoost (best_model). Metode ini menyimpan seluruh struktur model, termasuk parameter dan bobot yang telah dipelajari, ke dalam file biner dengan ekstensi .cbm (CatBoost Model). Dalam kasus ini, model disimpan dengan nama file "catboost_depression_model.cbm". Selain menyimpan model itu sendiri, kode ini juga menggunakan pustaka joblib untuk menyimpan daftar nama kolom fitur yang digunakan saat model dilatih.

4.6 Web Application Development

Pelaksanaan implementasi sistem prediksi risiko stres pada mahasiswa ini dilakukan melalui integrasi model ke dalam aplikasi berbasis website menggunakan framework Flask. Proses ini mencakup pengembangan antarmuka pengguna (UI)

yang responsif, pemrosesan backend yang efisien, integrasi model prediksi risiko depresi berbasis CatBoostClassifier, serta tampilan hasil prediksi secara real-time dengan klasifikasi tingkat risiko yang informatif.

Penggunaan Flask bertujuan untuk menyediakan platform yang fleksibel, cepat, dan mudah dikustomisasi sehingga sistem dapat diakses secara praktis oleh pengguna melalui browser tanpa perlu instalasi tambahan. Dengan antarmuka yang modern dan user-friendly menggunakan Tailwind CSS, pengguna cukup mengisi formulir data pribadi yang mencakup informasi demografis, gaya hidup, dan kondisi kesehatan mental, dan sistem akan langsung memberikan hasil prediksi berupa persentase risiko depresi (0-100%) beserta kategorisasi tingkat risiko (Rendah, Sedang, atau Tinggi).

Integrasi antara model CatBoostClassifier dan antarmuka web ini mendukung proses deteksi stres sedini mungkin secara efisien dan akurat, serta memungkinkan teknologi kecerdasan buatan digunakan lebih luas oleh mahasiswa, tenaga kesehatan, maupun masyarakat umum. Sistem ini juga dilengkapi dengan fitur edukasi kesehatan mental dan direktori bantuan yang memberikan informasi komprehensif kepada pengguna, sehingga tidak hanya berfungsi sebagai alat prediksi tetapi juga sebagai platform edukasi dan rujukan untuk layanan kesehatan mental yang tersedia.

4.6.1 Integrasi Model pada Website

Model CatBoostClassifier yang telah dilatih sebelumnya diintegrasikan ke dalam aplikasi web menggunakan framework Flask. Proses integrasi dilakukan dengan membuat file utama `app.py` yang berfungsi sebagai skrip backend yang menangani logika prediksi dan routing aplikasi.

Pada saat aplikasi dijalankan, model yang telah disimpan dalam format `.cbm` akan dimuat ke dalam memori menggunakan fungsi `load_model()` dari library CatBoost, sementara daftar kolom fitur yang diperlukan model dimuat dari file `model_columns.pkl` menggunakan library joblib. Pengguna dapat mengakses halaman aplikasi melalui browser, mengisi formulir data pribadi melalui antarmuka yang telah disediakan, dan aplikasi akan secara otomatis memproses input tersebut.

Data yang diinput oleh pengguna akan diproses melalui beberapa tahap: preprocessing data menggunakan fungsi `preprocess_input()`, konversi nilai-nilai kategorikal ke format yang sesuai dengan model, normalisasi nilai numerik seperti IPK (dari skala 0-4 ke skala 0-10), dan pengubahan bentuk data agar sesuai dengan input model. Setelah itu, data akan diprediksi menggunakan model CatBoostClassifier dengan bantuan objek Pool untuk menangani fitur kategorikal, dan hasil prediksi berupa probabilitas risiko depresi (dalam persentase) ditampilkan langsung di layar beserta tingkat risikonya (Rendah, Sedang, atau Tinggi).

Dengan pendekatan berbasis Flask ini, proses klasifikasi dapat dilakukan secara real-time, cepat, dan tanpa perlu intervensi manual tambahan. Selain itu, integrasi ini memungkinkan pengguna baik mahasiswa maupun tenaga kesehatan untuk menggunakan aplikasi secara langsung melalui browser, sehingga memperluas jangkauan dan kegunaan dari sistem prediksi risiko depresi yang telah dibangun. Antarmuka web yang responsif dan user-friendly juga memastikan pengalaman pengguna yang optimal dalam mengakses layanan prediksi kesehatan mental.

4.6.2 Implementasi *Backend*

Tahapan ini bertanggung jawab dalam menangani pemrosesan data input pengguna dan menjalankan proses klasifikasi menggunakan model machine learning. Backend dikembangkan menggunakan framework Flask yang menyediakan API endpoints untuk komunikasi dengan frontend. Model yang digunakan adalah CatBoostClassifier yang telah dilatih sebelumnya dan disimpan dalam format .cbm. Sistem secara otomatis akan memuat model dari direktori lokal saat aplikasi dijalankan, disertai dengan konfigurasi logging untuk monitoring dan debugging, sebagaimana ditunjukkan pada potongan kode berikut:

```
from flask import Flask, render_template, request
import pandas as pd
from catboost import CatBoostClassifier
import joblib
import logging
```

```
app = Flask(__name__)

# Configure logging
logging.basicConfig(level=logging.DEBUG)
logger = logging.getLogger(__name__)

# Load model and columns
model = CatBoostClassifier()
model.load_model("catboost_depression_model.cbm")
model_columns = joblib.load("model_columns.pkl")
```

Sistem logging dikonfigurasi pada level DEBUG untuk memungkinkan pencatatan detail aktivitas aplikasi, termasuk data input yang diterima dan error yang terjadi selama proses prediksi. Setelah model berhasil dimuat, sistem melakukan preprocessing terhadap data yang diinput pengguna melalui formulir web. Data yang diterima dari frontend akan diproses melalui fungsi `preprocess_input()` yang melakukan konversi tipe data, normalisasi nilai numerik seperti IPK (dari skala 0-4 ke skala 0-10), perhitungan fitur turunan, dan pengaturan nilai default untuk fitur-fitur yang tidak diinput pengguna. Proses preprocessing ditunjukkan dalam potongan kode berikut:

```
def preprocess_input(form):
    # Log form data
    logger.debug("Form data received: %s", dict(form))

    data = {
        "id": 0,
        "Gender": form["gender"],
        "Age": int(form["age"]),
        "Profession": "Student",
```

```
            "Academic Pressure": int(form["academic_pressure"]),
            "Work Pressure": int(form["work_pressure"]),
            "CGPA": (float(form["cgpa"]) / 4)  10,
            "Study Satisfaction": 3,
            "Job Satisfaction": 0,
            "Sleep Duration": float(form["sleep_duration"]),
            "Dietary Habits": form["diet"],
            "Degree": "BSc",
            "Have you ever had suicidal thoughts ?": form["suicidal_thoughts"],
            "Work/Study Hours": 5,
            "Financial Stress": int(form["financial_stress"]),
            "Family History of Mental Illness": form["family_history"],
            "Total_Pressure":                int(form["academic_pressure"])              +
int(form["work_pressure"]),
        }

        # Sleep Quality calculation berdasarkan durasi tidur
        if data["Sleep Duration"] < 5:
            data["Sleep_Quality"] = "Poor"
        elif 5 <= data["Sleep Duration"] <= 8:
            data["Sleep_Quality"] = "Normal"
        else:
            data["Sleep_Quality"] = "Over"

        # Menambahkan kolom yang mungkin tidak ada dengan nilai default
        for col in model_columns:
            if col not in data:
                data[col] = 0

        df = pd.DataFrame([data])
        df = df[model_columns]
```

```
    return df
```

Fungsi preprocessing ini melakukan beberapa transformasi penting, yaitu normalisasi IPK dengan mengkonversi skala IPK dari 0-4 menjadi 0-10 untuk kompatibilitas dengan model, perhitungan Total Pressure dengan menjumlahkan tekanan akademik dan tekanan kerja sebagai fitur gabungan, serta kategorisasi Sleep Quality yang menentukan kualitas tidur berdasarkan durasi tidur dimana Poor untuk durasi kurang dari 5 jam, Normal untuk 5-8 jam, dan Over untuk lebih dari 8 jam. Sistem juga melakukan penambahan fitur default dengan mengisi nilai 0 untuk kolom yang tidak ada dalam input namun dibutuhkan model.

Backend menyediakan beberapa endpoint untuk menangani berbagai fungsi aplikasi melalui routing Flask. Endpoint utama adalah route "/" yang menangani method GET dan POST untuk halaman prediksi, serta endpoint tambahan untuk halaman edukasi, direktori, dan tentang. Implementasi routing ditunjukkan dalam kode berikut:

```
@app.route("/", methods=["GET", "POST"])
def index():
    prediction = None
    form_data = {}
    if request.method == "POST":
        logger.debug("Request method: POST")
        logger.debug("Request form data: %s", dict(request.form))
        form_data = request.form.to_dict()
        try:
            input_df = preprocess_input(request.form)
            pred_prob = model.predict_proba(input_df)[0][1]
            prediction = round(pred_prob  100, 2)
        except Exception as e:
            logger.error("Error processing form: %s", str(e), exc_info=True)
```

```
            raise
        return        render_template("index.html",        prediction=prediction,
form_data=form_data)
```

Hasil prediksi dari model berupa nilai probabilitas risiko depresi (0-1), yang kemudian dikonversi menjadi persentase (0-100%). Sistem menggunakan metode `predict_proba()` dari CatBoostClassifier untuk mendapatkan probabilitas kelas positif (risiko depresi). Proses prediksi menggunakan kode `pred_prob = model.predict_proba(input_df)[0][1]` dimana `[0][1]` mengambil probabilitas kelas positif dari hasil prediksi, kemudian dikonversi menjadi persentase dengan `prediction = round(pred_prob  100, 2)` yang membulatkan hasil hingga 2 angka desimal.

Meskipun dalam implementasi saat ini sistem hanya mengembalikan nilai persentase, hasil ini dapat dikategorikan menjadi tiga tingkat risiko berdasarkan ambang batas yang telah ditentukan. Jika nilai prediksi ≥ 70%, maka sistem mengkategorikan sebagai Risiko Tinggi, jika ≥ 40% sebagai Risiko Sedang, dan sebaliknya sebagai Risiko Rendah. Backend dilengkapi dengan sistem error handling yang komprehensif dan logging untuk monitoring aplikasi. Setiap request yang masuk dicatat dalam log, termasuk data form yang diterima menggunakan `logger.debug("Request form data: %s", dict(request.form))`. Jika terjadi error selama proses prediksi, sistem akan mencatat detail error dengan stack trace lengkap melalui `logger.error("Error processing form: %s", str(e), exc_info=True)`.

Sistem melakukan validasi implisit melalui konversi tipe data yang akan menghasilkan error jika input tidak valid, seperti konversi string ke integer untuk field numerik. Selain itu, fungsi preprocessing memastikan bahwa semua kolom yang dibutuhkan model tersedia dengan mengisi nilai default untuk kolom yang tidak ada dalam input pengguna. Aplikasi dikonfigurasi untuk dapat berjalan dalam berbagai environment dengan pengaturan app.run(host='0.0.0.0', port=5000, debug=True) yang memungkinkan akses dari semua interface jaringan pada port 5000 dengan mode debug aktif.

Dengan demikian, backend sistem berperan penting dalam menjembatani data masukan pengguna dengan model klasifikasi CatBoostClassifier, sekaligus menangani aspek teknis dari proses prediksi secara otomatis dan efisien. Sistem ini memastikan bahwa data yang diproses sesuai dengan format yang diharapkan model, melakukan transformasi data yang diperlukan seperti normalisasi IPK dan perhitungan fitur turunan, serta memberikan hasil prediksi yang akurat dan informatif kepada pengguna melalui interface web yang responsif dengan dukungan logging dan error handling yang memadai untuk monitoring dan debugging aplikasi.

4.6.3 Implementasi *Frontend*

Subbab ini mendokumentasikan implementasi antarmuka pengguna (frontend) aplikasi StressSense. Setiap bagian memuat penjelasan mengenai tujuan dan fungsi halaman, cuplikan kode template (menggunakan Jinja2/HTML), serta uraian teknis terkait interaksi dengan backend berbasis Flask. Selain itu, disertakan catatan tambahan mengenai aspek aksesibilitas dan responsivitas, agar aplikasi dapat digunakan secara optimal di berbagai perangkat. Dokumentasi ini disusun untuk memberikan gambaran menyeluruh tentang bagaimana tampilan, alur input, dan proses prediksi risiko stres dirancang agar mudah dipahami, ramah pengguna, serta siap untuk deployment di lingkungan produksi.

- Base.html

  Template base.html digunakan sebagai kerangka dasar bagi seluruh halaman aplikasi. Struktur utamanya mencakup elemen navigasi (navbar), area konten, dan footer, serta memuat dependensi CSS dan JavaScript seperti Tailwind CSS dan Feather Icons. Penggunaan template induk ini memastikan konsistensi desain di seluruh halaman, mempermudah pemeliharaan, dan mempercepat proses pengembangan.

```
<!— Desktop Navigation —>
<div class="hidden md:flex items-center space-x-8">
  <a
    href="/"
    class="nav-link text-white hover:text-blue-100 font-medium {% if request.path == '/' %}active{% endif %}"
  >
    Prediksi
  </a>
  <a
    href="/edukasi"
    class="nav-link text-white hover:text-blue-100 font-medium {% if request.path == '/edukasi' %}active{% endif %}"
  >
    Edukasi
  </a>
  <a
    href="/direktori"
    class="nav-link text-white hover:text-blue-100 font-medium {% if request.path == '/direktori' %}active{% endif %}"
  >
    Direktori RS/Klinik
  </a>
  <a
    href="/tentang"
    class="nav-link text-white hover:text-blue-100 font-medium {% if request.path == '/tentang' %}active{% endif %}"
  >
    Tentang
  </a>
</div>
```

Gambar 4. 19 Potongan Kode Base.html

Dalam file base.html, terdapat beberapa blok yang dapat diisi sesuai kebutuhan halaman turunan. Blok-blok tersebut meliputi *title* dan *content*, yang memungkinkan penyesuaian konten seperti halaman Prediksi, Edukasi, Direktori RS/Klinik, dan Tentang. Selain itu, terdapat blok *scripts* yang memungkinkan setiap halaman menambahkan skrip JavaScript khusus tanpa mengubah struktur utama dari base.html. File base.html sendiri terletak di dalam folder *templates/* pada proyek Flask, sehingga dapat digunakan sebagai template dasar yang dapat di-extend oleh semua halaman lain dalam proyek tersebut.

Dalam penerapannya, pemanggilan skrip dan *library* ditempatkan pada bagian <head> atau sebelum penutup <body> untuk memastikan pemuatan yang optimal. Untuk keperluan produksi, file CSS dan JavaScript sebaiknya diminifikasi atau digabung menggunakan bundler seperti Webpack atau esbuild untuk meningkatkan performa.

- Index.html

Halaman index.html berperan sebagai beranda sekaligus pusat interaksi utama pengguna. Bagian awal halaman menampilkan deskripsi layanan dan fitur utama dalam bentuk kartu informasi (card), kemudian diikuti oleh formulir prediksi yang memungkinkan pengguna memasukkan data seperti jenis kelamin, umur, nilai IPK (CGPA), durasi tidur, tekanan akademik, serta faktor lainnya.

Form input dirancang responsif menggunakan utilitas Tailwind CSS sehingga tetap nyaman diakses pada berbagai ukuran layar. Validasi input dilakukan di sisi

klien dengan memanfaatkan fitur HTML5 seperti atribut required, min, dan max, serta JavaScript ringan untuk memperbaiki pengalaman pengguna. Hasil prediksi, berupa persentase risiko, akan ditampilkan secara langsung di bawah formulir setelah pengiriman data.

```
<!— Main Form Card —>
<div class="bg-white rounded-xl shadow-lg overflow-hidden">
  <div class="p-8">
    <form method="POST" class="space-y-6" id="stresssense-form">
      <!— Personal Information Section —>
      <div class="bg-blue-50 p-4 rounded-lg mb-6">
        <h2
          class="text-xl font-semibold text-blue-800 mb-4 flex items-center"
        >
          <i data-feather="user" class="w-5 h-5 mr-2"></i>
          Informasi Pribadi
        </h2>
        <div class="grid grid-cols-1 md:grid-cols-2 gap-6">
          <div>
            <label class="block text-sm font-medium text-gray-700 mb-2"      You, 3 weeks ago • Uncommitted changes
              >Jenis Kelamin</label
            >
            <select
              name="gender"
              class="w-full rounded-lg border-gray-300 focus:border-blue-500 focus:ring focus:ring-blue-200 focus:ring-opaci
              required
            >
              <option value="" disabled {% if not form_data.get('gender') %}selected{% endif %}>
                Pilih jenis kelamin...
              </option>
              <option value="Male" {% if form_data.get('gender') == 'Male' %}selected{% endif %}>Laki-laki</option>
              <option value="Female" {% if form_data.get('gender') == 'Female' %}selected{% endif %}>Perempuan</option>
            </select>
          </div>
```

Gambar 4. 20 Potongan kode Index.html

Integrasi form dengan backend dilakukan menggunakan metode POST yang diarahkan ke rute utama Flask. Backend kemudian memproses data melalui fungsi pra-pemrosesan (preprocess_input()), memanggil model CatBoost dengan predict_proba(), dan mengirimkan hasil kembali ke template. Nama atribut name pada elemen form harus konsisten dengan yang digunakan di sisi server. Apabila menggunakan Flask-WTF, token CSRF dapat disisipkan di template untuk meningkatkan keamanan.

- Edukasi.html

Halaman Edukasi Kesehatan Mental berfungsi sebagai pusat informasi yang menyediakan materi edukatif, tips praktis, dan referensi sumber bantuan terkait kesehatan mental mahasiswa. Konten pada halaman ini bersifat statis, tetapi dapat diubah menjadi dinamis menggunakan Markdown atau sistem manajemen konten sederhana jika dibutuhkan.

```
s > edukasi.html
{% extends 'base.html' %}

{% block title %}Edukasi Kesehatan Mental - StressSense{% endblock %}

{% block content %}
<div class="max-w-4xl mx-auto">
    <!— Hero Section —>
    <div class="text-center mb-12">
        <h1 class="text-4xl font-bold text-blue-800 mb-4">Edukasi Kesehatan Mental</h1>
        <p class="text-gray-600 text-lg max-w-2xl mx-auto">
            Pelajari lebih lanjut tentang kesehatan mental, tanda-tanda depresi, dan cara menjaga kesehatan mental Anda.
        </p>
    </div>

    <!— Main Content —>
    <div class="grid grid-cols-1 md:grid-cols-2 gap-8">
        <!— Understanding Depression —>
        <div class="bg-white rounded-xl shadow-lg overflow-hidden">
            <div class="p-6">
                <h2 class="text-2xl font-bold text-blue-800 mb-4 flex items-center">
                    <i data-feather="brain" class="w-6 h-6 mr-2"></i>
                    Memahami Depresi
                </h2>
                <div class="prose prose-blue max-w-none">
                    <p class="text-gray-600 mb-4">
                        Depresi adalah gangguan suasana hati yang serius yang mempengaruhi cara Anda merasa, berpikir, dan
                        Ini dapat menyebabkan berbagai masalah emosional dan fisik.
                    </p>
                    <h3 class="text-lg font-semibold text-blue-700 mt-6 mb-3">Tanda-tanda Umum:</h3>
                    <ul class="space-y-2 text-gray-600">
                        <li class="flex items-start">
                            <i data-feather="check-circle" class="w-5 h-5 text-blue-500 mr-2 mt-1"></i>
                            <span>Perasaan sedih, kosong, atau putus asa yang berkelanjutan</span>
                        </li>
```

Gambar 4. 21 Potongan Kode Edukasi.html

Secara teknis, halaman ini memanfaatkan *template inheritance* dari base.html, sehingga hanya bagian konten yang berbeda. Informasi ditata dalam elemen HTML semantik seperti <article> dan <section> untuk meningkatkan aksesibilitas, serta menggunakan *class utility* Tailwind CSS agar tampil responsif di berbagai perangkat.

Dalam penerapannya, setiap topik kesehatan mental dapat dipisahkan menjadi komponen atau modul yang dapat diperbarui secara berkala. Penyusunan konten sebaiknya menyertakan hyperlink menuju jurnal, artikel, atau situs resmi organisasi kesehatan mental untuk meningkatkan kredibilitas.

- Direktori.html

Halaman Direktori Bantuan menampilkan daftar kontak penting dan layanan profesional yang dapat dihubungi oleh pengguna saat memerlukan bantuan darurat atau konseling. Tujuan utama halaman ini adalah memastikan informasi layanan mudah diakses, jelas, dan langsung dapat digunakan (misalnya dengan link tel: untuk langsung melakukan panggilan dari perangkat seluler). Daftar kontak diurutkan dengan memprioritaskan nomor darurat atau layanan penting di bagian atas, diikuti oleh kontak layanan pendukung lainnya.

```
nplates > direktori.html
6   <div class="max-w-4xl mx-auto">
16      <div class="bg-white rounded-xl shadow-lg overflow-hidden mb-8">
38      </div>
39
40      <!— Facilities Grid —>
41      <div class="grid grid-cols-1 md:grid-cols-2 gap-6">
42          <!— RSJ Dr. Soeharto Heerdjan —>
43          <div class="bg-white rounded-xl shadow-lg overflow-hidden">
44              <div class="p-6">
45                  <div class="flex items-start justify-between">
46                      <div>
47                          <h3 class="text-xl font-bold text-blue-800 mb-2">RSJ Dr. Soeharto Heerdjan</h3>
48                          <p class="text-gray-600 mb-4">Rumah Sakit Jiwa</p>
49                      </div>
50                      <span class="bg-blue-100 text-blue-800 text-sm font-semibold px-3 py-1 rounded-full">Ru
51                  </div>
52                  <div class="space-y-4">
53                      <div class="flex items-start">
54                          <i data-feather="map-pin" class="w-5 h-5 text-gray-500 mr-2 mt-1"></i>
55                          <p class="text-gray-600">Jl. Prof. Dr. Latumeten No.1, Jakarta Barat</p>
56                      </div>
57                      <div class="flex items-start">
58                          <i data-feather="phone" class="w-5 h-5 text-gray-500 mr-2 mt-1"></i>
59                          <p class="text-gray-600">(021) 5682841</p>
60                      </div>
61                      <div class="flex items-start">
62                          <i data-feather="clock" class="w-5 h-5 text-gray-500 mr-2 mt-1"></i>
63                          <p class="text-gray-600">24 jam</p>
64                      </div>
65                      <div class="flex items-start">
66                          <i data-feather="info" class="w-5 h-5 text-gray-500 mr-2 mt-1"></i>
67                          <p class="text-gray-600">Layanan psikiatri, psikologi, dan rehabilitasi</p>
```

Gambar 4. 22 Potongan Kode Direktori.html

Secara teknis, halaman ini menampilkan informasi fasilitas layanan kesehatan dalam bentuk *card* menggunakan *utility class* Tailwind seperti shadow-lg, rounded-xl, dan grid-cols untuk tata letak yang rapi dan responsif. Setiap *card* memuat nama fasilitas, kategori, alamat, nomor telepon dengan *link* tel:, jam operasional, serta deskripsi singkat layanan. Ikon seperti *map pin*, telepon, jam, dan info digunakan untuk memperjelas setiap jenis informasi. Struktur ini memudahkan pengguna menemukan detail fasilitas secara cepat, dan jika jumlah fasilitas banyak, desain ini dapat dikembangkan menjadi *grid* dengan *filter* atau pencarian untuk navigasi yang lebih efisien.

- Tentang.html

Halaman *Tentang StressSense* berfungsi memberikan informasi umum mengenai aplikasi, mencakup tujuan pengembangan, teknologi yang digunakan, dan pernyataan privasi singkat. Bagian ini penting untuk transparansi dan membangun kepercayaan pengguna. Penjelasan disusun secara ringkas, jelas, dan disertai daftar teknologi agar pembaca memahami fondasi teknis aplikasi.

```html
{% extends 'base.html' %}

{% block title %}Tentang StressSense{% endblock %}

{% block content %}
<div class="max-w-4xl mx-auto">
    <!— Hero Section —>
    <div class="text-center mb-12">
        <h1 class="text-4xl font-bold text-blue-800 mb-4">Tentang StressSense</h1>
        <p class="text-gray-600 text-lg max-w-2xl mx-auto">
            Platform prediksi risiko depresi untuk membantu mahasiswa menjaga kesehatan mental mereka.
        </p>
    </div>

    <!— Main Content —>
    <div class="grid grid-cols-1 md:grid-cols-2 gap-8">
        <!— Our Mission —>
        <div class="bg-white rounded-xl shadow-lg overflow-hidden">
            <div class="p-6">
                <h2 class="text-2xl font-bold text-blue-800 mb-4 flex items-center">
                    <i data-feather="target" class="w-6 h-6 mr-2"></i>
                    Misi Kami
                </h2>
                <div class="prose prose-blue max-w-none">
                    <p class="text-gray-600 mb-4">
                        StressSense hadir untuk membantu mahasiswa dalam mengidentifikasi risiko depresi sejak d
                        Kami berkomitmen untuk memberikan akses ke informasi dan sumber daya yang diperlukan unt
                        menjaga kesehatan mental.
                    </p>
                    <ul class="space-y-2 text-gray-600">
                        <li class="flex items-start">
                            <i data-feather="check-circle" class="w-5 h-5 text-blue-500 mr-2 mt-1"></i>
```

Gambar 4. 23 Potongan Kode Tentang.html

Secara teknis, kode pada gambar tersebut membentuk halaman "Tentang StressSense" menggunakan struktur *template* dengan Tailwind CSS untuk pengaturan gaya. Bagian *Hero Section* menampilkan judul besar dan deskripsi singkat tentang fungsi platform sebagai alat prediksi risiko depresi bagi mahasiswa. Di bagian *Main Content*, terdapat *card* berjudul "Misi Kami" yang dilengkapi ikon, paragraf deskriptif, dan *utility class* seperti rounded-xl, shadow-lg, serta *grid layout* untuk tampilan rapi dan responsif. Struktur ini memudahkan pembaca memahami tujuan dan manfaat platform secara visual terorganisir, dan dapat dikembangkan lagi dengan tambahan poin misi atau ikon yang relevan untuk memperjelas pesan.

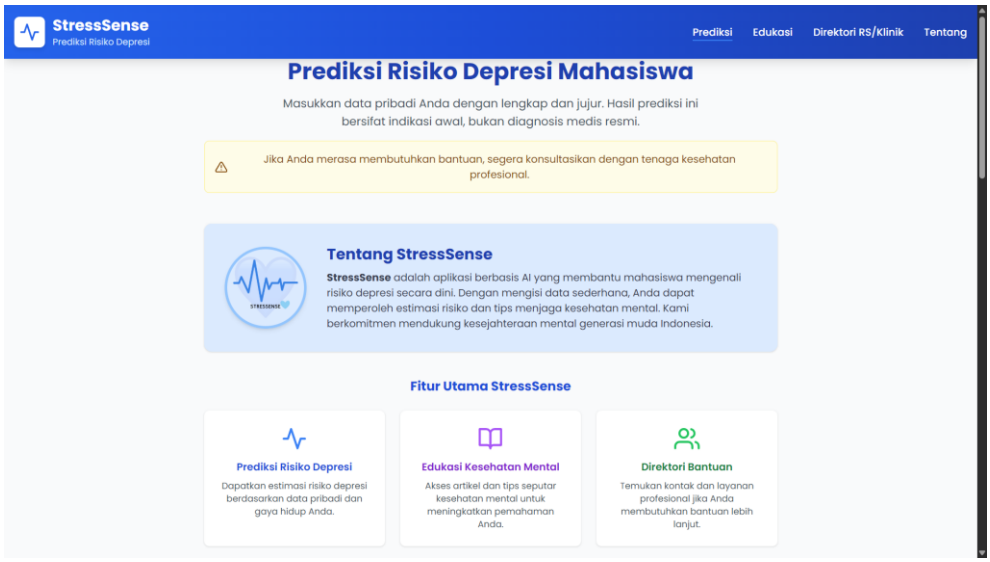
4.6.4 Tampilan Antarmuka Website

Antarmuka website *StressSense* dirancang agar mudah digunakan, informatif, dan modern dengan tampilan responsif berbasis Tailwind CSS. Website ini terdiri dari beberapa halaman utama seperti Beranda, Prediksi, Edukasi Kesehatan Mental, Direktori RS/Klinik, dan Tentang, sehingga memudahkan pengguna dalam mengakses seluruh fitur aplikasi.

- Halaman Beranda dan Prediksi

Halaman ini merupakan tampilan utama saat pengguna pertama kali mengakses website. Pada halaman ini, pengguna diperkenalkan dengan nama aplikasi, deskripsi singkat mengenai tujuan StressSense, serta tombol untuk

memulai proses prediksi risiko depresi.

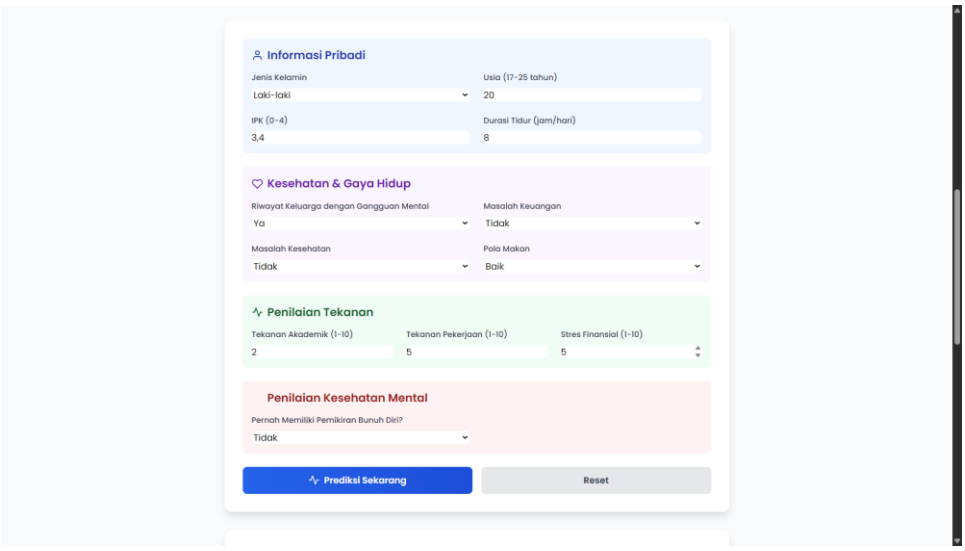


Gambar 4. 24 Halaman Beranda

Navigasi di bagian atas memudahkan pengguna untuk berpindah ke halaman Prediksi, Edukasi, Direktori, dan Tentang. Selain itu, terdapat penjelasan fitur utama, kutipan inspiratif, dan tips self-care untuk mendukung kesehatan mental pengguna.

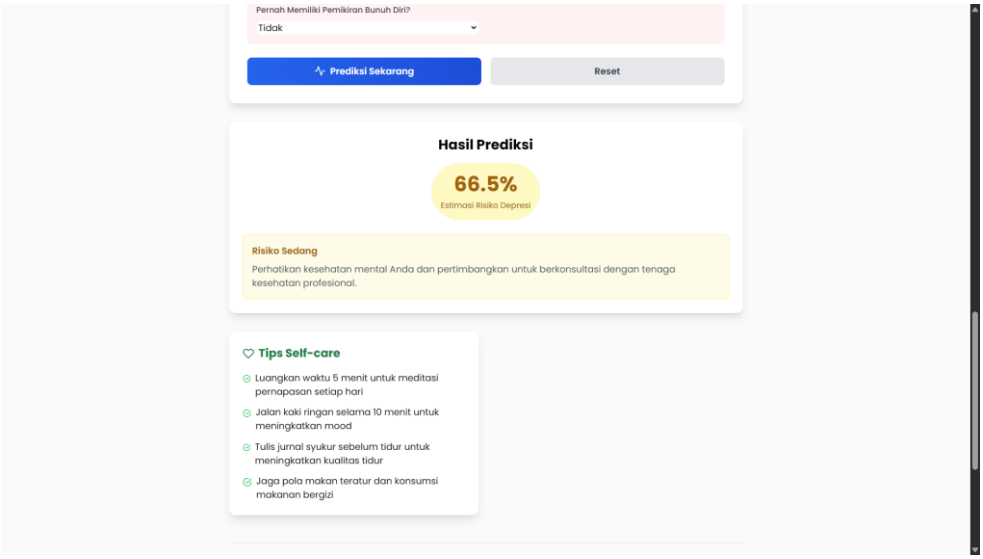
Pada bagian yang sama, tersedia formulir untuk memprediksi risiko depresi yang dapat langsung diisi oleh pengguna. Formulir ini dirancang secara user-friendly agar memudahkan pengguna dalam mengisi data secara lengkap dan akurat. Melalui pengisian formulir ini, pengguna dapat memperoleh estimasi risiko depresi secara cepat dan efisien, yang dapat membantu dalam deteksi dini dan

penanganan yang tepat.



Gambar 4. 25 Form Prediksi Stres di Halaman Beranda

Prediksi merupakan fitur utama yang memungkinkan pengguna melakukan estimasi risiko depresi dengan mengisi formulir data pribadi dan gaya hidup dengan jujur. Data yang dikumpulkan dari formulir ini meliputi berbagai aspek seperti riwayat kesehatan mental, tingkat stres, pola tidur, kebiasaan makan, serta faktor lingkungan yang mempengaruhi kesehatan mental.



Gambar 4. 26 Hasil Form Prediksi Stres

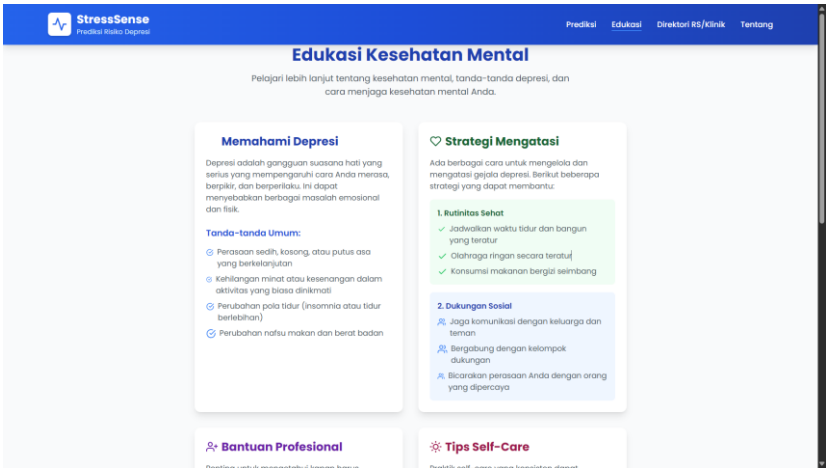
Setelah data diisi dan dikirim, sistem akan menampilkan hasil prediksi berupa persentase risiko depresi, kategori risiko (rendah, sedang, tinggi), serta saran atau

rekomendasi yang sesuai.

Pengguna juga diberikan tips self-care dan informasi penting terkait kesehatan mental. Dengan informasi yang lengkap dan akurat, sistem dapat memberikan hasil prediksi yang lebih tepat dan relevan, sehingga pengguna dapat mengambil langkah-langkah preventif atau mencari bantuan profesional jika diperlukan.

- Halaman Edukasi

Halaman edukasi dirancang untuk memberikan informasi yang komprehensif dan mudah dipahami mengenai kesehatan mental. Materi disajikan dalam bentuk kartu informatif, artikel singkat, dan daftar tips yang membahas berbagai topik penting seperti pengertian depresi, tanda-tanda dan gejala, strategi coping, serta langkah-langkah self-care yang dapat dilakukan sehari-hari.



Gambar 4. 27 Halamam Edukasi

Materi disajikan dalam bentuk kartu informatif dan daftar yang dirancang secara sistematis untuk memudahkan pemahaman pengguna. Pendekatan ini memungkinkan pengguna untuk dengan cepat mengidentifikasi poin-poin penting dan mendapatkan informasi secara efisien. Selain itu, materi ini dilengkapi dengan ilustrasi dan contoh nyata yang relevan, sehingga meningkatkan daya ingat dan pemahaman terhadap topik yang dibahas.

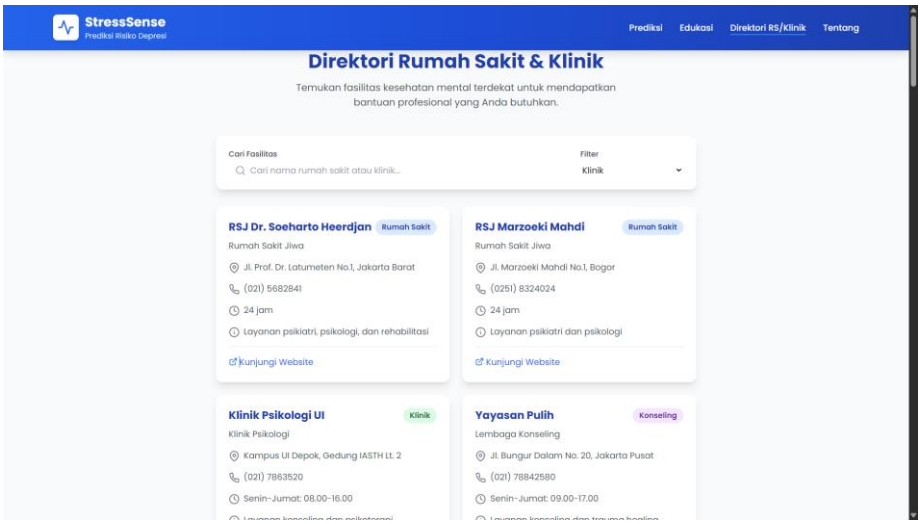
Halaman ini bertujuan untuk meningkatkan kesadaran dan pemahaman pengguna mengenai pentingnya menjaga kesehatan mental. Dalam konteks ini, pengguna diajarkan untuk mengenali tanda-tanda awal gangguan mental dan

memahami langkah-langkah yang harus diambil untuk menjaga kesejahteraan psikologis mereka. Selain itu, halaman ini juga memberikan panduan tentang kapan dan bagaimana mencari bantuan profesional, termasuk jenis-jenis layanan yang tersedia dan manfaatnya.

- Halaman Direktori RS/Klinik

Halaman ini menyediakan direktori lengkap rumah sakit, klinik, dan layanan psikologi di berbagai wilayah Indonesia. Pengguna dapat mencari fasilitas berdasarkan nama, lokasi, atau jenis layanan, serta memanfaatkan fitur filter untuk menampilkan kategori tertentu seperti rumah sakit jiwa, klinik psikologi, atau konselor.

Setiap entri fasilitas dilengkapi dengan informasi detail seperti alamat, nomor kontak, jam operasional, dan tautan ke website resmi, sehingga memudahkan pengguna untuk mendapatkan bantuan profesional sesuai kebutuhan.



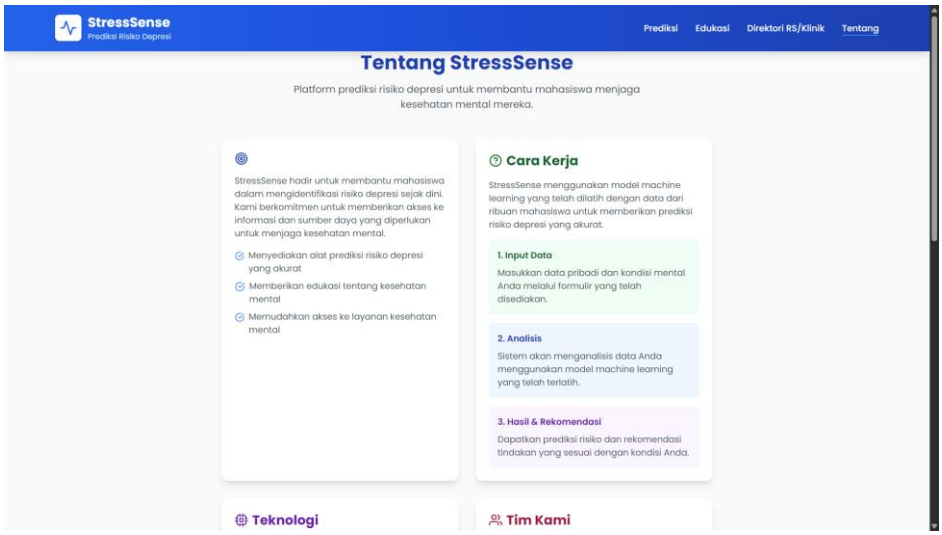
Gambar 4. 28 Halaman Direktori RS/Klinik

Selain itu, halaman ini juga menampilkan informasi penting terkait layanan darurat dan kontak bantuan krisis, sehingga pengguna dapat segera mengakses pertolongan jika berada dalam kondisi mendesak.

- Halaman Tentang

Halaman tentang berisi penjelasan mendalam mengenai tujuan, manfaat, dan latar belakang pengembangan aplikasi StressSense. Di sini, pengguna dapat mengetahui teknologi utama yang digunakan, seperti pemanfaatan machine

learning dan framework Flask untuk prediksi risiko depresi.



Gambar 4. 29 Halaman Tentang

Selain itu, halaman ini juga memperkenalkan tim pengembang, visi misi aplikasi, serta menyediakan informasi kontak untuk kolaborasi atau pertanyaan lebih lanjut. Terdapat pula disclaimer yang menegaskan bahwa hasil prediksi bukan merupakan diagnosis medis, melainkan sebagai alat bantu awal untuk meningkatkan kesadaran dan mendorong tindakan preventif.

Dengan desain yang responsif, navigasi yang jelas, dan konten yang informatif, StressSense diharapkan dapat menjadi solusi digital yang efektif dalam membantu mahasiswa mengenali risiko stres sedini mungkin, memperoleh edukasi kesehatan mental, serta mengakses layanan profesional dengan mudah dan cepat.

4.7 Deployment Website

Implementasi deployment website dilakukan menggunakan platform Railway sebagai media hosting aplikasi. Railway dipilih karena kemudahan integrasinya dengan aplikasi berbasis Python serta dukungan penuh terhadap deployment berbasis Docker. Berikut adalah tahapan implementasi deployment website pada project StressSense:

Pertama, seluruh source code aplikasi, model machine learning (catboost_depression_model.cbm), file pendukung seperti model_columns.pkl, serta file statis dan template telah dipersiapkan dalam satu folder project yang

terstruktur. File utama aplikasi adalah app.py, yang berisi kode program untuk menjalankan web server menggunakan framework Flask, serta fungsi untuk melakukan prediksi risiko depresi berdasarkan data yang diinputkan pengguna. Folder static digunakan untuk menyimpan file gambar dan style, sedangkan folder templates berisi file HTML yang digunakan sebagai tampilan website.

Selanjutnya, dibuat file requirements.txt yang berisi daftar seluruh dependensi Python yang diperlukan oleh aplikasi, seperti Flask, CatBoost, Pandas, dan Joblib. File ini sangat penting untuk memastikan seluruh library yang dibutuhkan dapat terinstall secara otomatis pada saat proses deployment.

Tahap berikutnya adalah pembuatan file Dockerfile. File ini berisi instruksi untuk membangun image Docker yang akan digunakan oleh Railway. Instruksi di dalam Dockerfile meliputi pemilihan base image Python, instalasi dependensi sistem yang diperlukan, penyalinan seluruh file project ke dalam image, serta perintah untuk menjalankan aplikasi Flask. Pada bagian akhir Dockerfile, aplikasi diatur agar dapat membaca environment variable PORT yang disediakan oleh Railway, sehingga aplikasi dapat berjalan pada port yang sesuai dengan konfigurasi platform.

Setelah seluruh file siap, project diunggah ke Railway. Proses upload dapat dilakukan dengan menghubungkan Railway ke repository GitHub yang berisi project, atau dengan mengunggah file secara manual melalui dashboard Railway. Railway kemudian akan secara otomatis mendeteksi file Dockerfile dan menjalankan proses build image. Jika proses build berhasil, Railway akan menjalankan container aplikasi dan memberikan URL publik yang dapat diakses oleh pengguna.

Setelah deployment selesai, aplikasi StressSense dapat diakses melalui URL yang disediakan oleh Railway, misalnya: https://stresssense.up.railway.app. Pengujian dilakukan dengan mengakses website, mengisi formulir prediksi risiko depresi, dan memastikan seluruh fitur berjalan dengan baik, mulai dari proses prediksi, navigasi halaman, hingga tampilan responsif pada berbagai perangkat. Dengan demikian, aplikasi telah siap digunakan secara online oleh penggun

4.8 Testing

Pengujian aplikasi website dilakukan untuk memastikan bahwa seluruh fitur yang tersedia pada Website StressSense dapat berjalan dengan baik, memberikan hasil prediksi risiko stres yang sesuai, serta menghadirkan pengalaman pengguna yang optimal. Pengujian ini sangat penting agar aplikasi benar-benar dapat digunakan secara efektif oleh pengguna sesuai dengan tujuan pengembangannya, yaitu membantu mahasiswa mengenali risiko stres secara dini dan mendapatkan akses edukasi serta layanan kesehatan mental.

Aspek yang diuji meliputi proses pengisian data pada formulir prediksi, akurasi hasil prediksi yang diberikan oleh sistem, tampilan dan interaksi antarmuka pengguna, serta keamanan dan responsivitas aplikasi di berbagai perangkat, baik desktop maupun mobile. Selain itu, pengujian juga mencakup fitur-fitur pendukung seperti akses ke halaman edukasi, direktori layanan kesehatan mental, dan halaman tentang aplikasi.

Pada sub-bab berikutnya, akan dijelaskan lebih lanjut mengenai metode pengujian yang digunakan, yaitu uji black box, untuk memastikan setiap fitur berjalan sesuai dengan fungsinya. Hasil pengujian akan diuraikan secara detail untuk menilai keandalan, kemudahan penggunaan, dan kualitas layanan yang diberikan oleh aplikasi Web StressSense.

4.8.1 Pengujian Fungsional (*Black Box Testing*)

Black box testing dilakukan untuk memastikan setiap fitur pada aplikasi website berjalan sesuai dengan fungsinya tanpa melihat kode program secara langsung. Pengujian ini berfokus pada input yang diberikan oleh pengguna dan output yang dihasilkan oleh sistem. Setiap fitur diuji berdasarkan skenario penggunaan yang telah ditentukan, sehingga dapat diketahui apakah aplikasi sudah memenuhi kebutuhan pengguna. Beberapa skenario pengujian yang dilakukan antara lain:

Tabel 4. 1 Pengujian Black Box

No	Fitur yang Diuji	Skenario Uji	Input/Skenario	Output yang Diharapkan	Hasil Uji

1	Formulir Prediksi Risiko Depresi	Pengguna mengisi dan mengirimkan formulir prediksi risiko depresi	Data pribadi dan gaya hidup valid	Hasil prediksi risiko (persentase & kategori) serta saran muncul di halaman	Berhasil
2	Navigasi Halaman	Pengguna mengakses menu Beranda, Edukasi, Direktori, dan Tentang	Klik menu navigasi	Halaman yang dipilih tampil dengan benar	Berhasil
3	Validasi Formulir	Pengguna mengirimkan formulir dengan data tidak lengkap/invalid	Data kosong/tidak valid	Muncul pesan error atau form tidak dapat dikirim	Berhasil
4	Tampilan Edukasi	Pengguna membuka halaman edukasi kesehatan mental	Akses halaman edukasi	Informasi edukasi tampil dengan baik dan mudah dibaca	Berhasil
5	Direktori RS/Klinik	Pengguna mencari fasilitas kesehatan mental melalui	Kata kunci/jenis layanan	Daftar fasilitas sesuai pencarian/filter tampil dengan benar	Berhasil

		fitur pencarian/filter			
6	Responsivitas Tampilan	Pengguna membuka aplikasi di perangkat mobile dan desktop	Akses dari berbagai perangkat	Tampilan tetap rapi, responsif, dan mudah digunakan	Berhasil
7	Keamanan Data	Pengguna mengirimkan data prediksi	Data pribadi dikirim	Data tidak ditampilkan ke publik, tidak ada error keamanan	Berhasil

Berdasarkan hasil pengujian fungsional di atas, seluruh fitur utama pada aplikasi website StressSense dapat berjalan dengan baik dan sesuai dengan yang diharapkan. Proses pengisian dan pengiriman formulir prediksi risiko depresi berhasil dilakukan tanpa kendala, dan sistem mampu menampilkan hasil prediksi berupa persentase risiko serta kategori (rendah, sedang, tinggi) secara otomatis setelah data dikirimkan. Navigasi antar halaman seperti Beranda & Prediksi, Edukasi Kesehatan Mental, Direktori RS/Klinik, dan Tentang juga dapat diakses dengan lancar tanpa error.

Validasi pada formulir prediksi berjalan sesuai harapan, di mana sistem akan menolak input yang tidak lengkap atau tidak valid dan memberikan pesan peringatan kepada pengguna. Fitur pencarian dan filter pada halaman direktori layanan kesehatan mental juga berfungsi dengan baik, menampilkan daftar fasilitas sesuai dengan kata kunci atau kategori yang dipilih. Dari sisi tampilan, aplikasi tetap responsif dan mudah digunakan baik pada perangkat desktop maupun mobile. Secara keseluruhan, hasil pengujian menunjukkan bahwa aplikasi StressSense telah memenuhi aspek fungsionalitas, keamanan, dan user experience yang baik, serta

siap digunakan oleh pengguna.

4.8.2 User Acceptance Test

User Acceptance Test (UAT) dilakukan untuk memastikan bahwa aplikasi web StressSense telah memenuhi kebutuhan dan harapan pengguna sebelum diimplementasikan secara resmi. UAT ini melibatkan responden dengan kriteria khusus, yaitu mahasiswa aktif berusia 17–25 tahun, yang sesuai dengan target pengguna aplikasi. Proses pengujian dilakukan menggunakan kuesioner berbasis Google Forms, di mana responden diminta untuk mencoba seluruh fitur yang tersedia, termasuk formulir prediksi risiko depresi, halaman edukasi kesehatan mental, direktori layanan bantuan, serta halaman tentang aplikasi. Setelah mencoba fitur-fitur tersebut, responden memberikan penilaian terkait kemudahan penggunaan (user-friendly), kejelasan informasi, kecepatan respon aplikasi, serta memberikan masukan atau saran untuk pengembangan lebih lanjut. Hasil dari UAT ini digunakan sebagai dasar evaluasi untuk meningkatkan kualitas aplikasi dan memastikan aplikasi siap digunakan secara efektif di lingkungan pengguna sebenarnya.

Tabel 4.2 Hasil *User Acceptance Testing*

No	Pertanyaan	Hasil
1	Apakah fitur prediksi risiko depresi menampilkan hasil dengan jelas dan akurat?	70% memilih nilai 5 (Rata-rata: 4,7)
2	Apakah navigasi antar halaman berjalan lancar tanpa error?	60% memilih nilai 5 (Rata-rata: 4,6)
3	Apakah pesan error muncul dengan jelas saat data tidak lengkap/tidak valid?	80% memilih nilai 5 (Rata-rata: 4,8)
4	Apakah halaman edukasi informatif dan mudah dipahami?	90% memilih nilai 5 (Rata-rata: 4,9)
5	Apakah fitur pencarian/filter pada halaman Direktori RS/Klinik bekerja dengan benar?	70% memilih nilai 5 (Rata-rata: 4,7)
6	Apakah tampilan antarmuka aplikasi ini	80% memilih nilai 5 (Rata-

	menurut Anda user-friendly (mudah digunakan)?	rata: 4,8)
7	Apakah Anda merasa aplikasi ini bermanfaat secara keseluruhan?	90% memilih nilai 5 (Rata-rata: 4,9)

Secara keseluruhan, aplikasi mendapat respons sangat positif dari pengguna dengan rata-rata penilaian 4,77 dari skala 5. Hampir semua fitur dinilai Sangat Baik, dan tidak ada nilai rata-rata di bawah 4, menunjukkan bahwa aplikasi ini telah memenuhi ekspektasi pengguna dalam hal kemudahan penggunaan, kejelasan informasi, dan manfaatnya.

Adapun saran yang diberikan oleh pengisi kuesioner dalam google form UAT yang telah dibuat, berikut merupakan saran yang diberikan oleh setiap 10 responden.

Tabel 4.3 Hasil Masukan atau Saran Pengguna Terhadap Webiste

No	**Nama**	**Apa masukan atau saran Anda untuk pengembangan aplikasi ini ke depannya?**
1	Darmawan	Aplikasi memberikan kesan modern dan bermanfaat
2	Faisal Shidqi Adli	Fitur sudah berjalan dengan baik dan gampang digunakan
3	Alyaa	Secara keseluruhan tampilan dan fungsinya sudah memuaskan
4	Alif Fathur Rahman	semoga makin bagus
5	Bagja	masih bisa dikembangkan dari sisi penggunaan agar mudah digunakan oleh berbagai kalangan
6	Shaun Patrick Hendra	Tampilan sudah bagus dan mudah digunakan. Prediksi sangat membantu beserta tips tipsnya
7	Nabila Aurora	Sebaiknya ditambahkan mode gelap agar nyaman digunakan di malam hari.

8	Raihan Fathan	sudah keren apknya
9	Ahmad Laahiq	tidak ada masukan tambahan
10	Ibra	Desainnya bagus dan fiturnya berjalan dengan lancar

Berdasarkan hasil tanggapan pengguna, secara umum website dinilai sudah sangat baik, bermanfaat, dan mudah digunakan, khususnya dalam membantu pendeteksian dini kondisi mental. Sebagian besar responden memberikan komentar positif terkait navigasi yang jelas, tampilan yang ramah pengguna, serta penggunaan bahasa yang tepat dan nyaman dibaca. Beberapa saran yang muncul meliputi penambahan tombol mulai prediksi di halaman utama agar akses lebih cepat, peningkatan variasi konten motivasi yang dibagi berdasarkan tema dan dapat disesuaikan dengan hasil analisis pengguna, penambahan fitur baru yang lebih menarik, penempatan link atau akses langsung ke halaman prediksi di beranda, serta penyempurnaan tampilan antarmuka agar semakin user-friendly. Saran-saran ini bersifat penyempurnaan dan pengembangan fitur, bukan perbaikan mendasar, sehingga dapat disimpulkan bahwa website telah memenuhi kebutuhan mayoritas pengguna, namun tetap memiliki ruang untuk peningkatan pengalaman penggunaan di masa mendatang.

5. Penutup

5.1 Kesimpulan

Penelitian ini berhasil merancang dan mengembangkan sebuah aplikasi web berbasis framework Flask dengan bahasa pemrograman Python untuk memprediksi tingkat risiko stres pada mahasiswa berusia 17–25 tahun. Model prediksi dibangun menggunakan algoritma CatBoostClassifier yang dioptimasi dengan RandomizedSearchCV, sehingga diperoleh kombinasi hyperparameter terbaik untuk meningkatkan akurasi. Dataset yang digunakan berasal dari Student Depression Dataset di platform Kaggle, yang berisi data survei non-medis dengan berbagai fitur relevan seperti gender, usia, IPK, durasi tidur, tekanan akademik, tekanan pekerjaan, stres finansial, masalah keuangan, kondisi kesehatan, riwayat keluarga dengan gangguan mental, pola makan, dan pemikiran bunuh diri.

Evaluasi model dilakukan melalui metrik kuantitatif seperti confusion matrix, classification report, kurva ROC, dan AUC, serta analisis feature importance untuk memahami kontribusi masing-masing variabel terhadap prediksi. Hasil prediksi disajikan dalam bentuk persentase risiko stres yang informatif dan mudah dipahami, disertai rekomendasi awal untuk langkah-langkah preventif, seperti konseling psikologis, pengelolaan waktu, perbaikan pola makan, peningkatan kualitas tidur, dan penguatan dukungan sosial.

Dengan sifatnya yang non-klinis, aplikasi ini tidak dimaksudkan sebagai alat diagnosis medis, melainkan sebagai sarana deteksi dini yang dapat mendukung upaya promotif dan preventif kesehatan mental mahasiswa. Penelitian ini membuktikan bahwa penerapan machine learning dengan algoritma CatBoostClassifier pada data survei non-medis dapat menghasilkan prediksi risiko stres yang akurat dan bermanfaat sebagai alat bantu pengambilan keputusan awal bagi mahasiswa maupun pihak terkait.

5.2 Saran

Pengembangan aplikasi StressSense ke depannya disarankan untuk menambahkan fitur *dark mode* dan sistem pengingat pengecekan rutin untuk meningkatkan pengalaman pengguna. Integrasi chatbot atau layanan konseling online pada halaman direktori yang sudah ada dapat memberikan bantuan lanjutan yang lebih cepat. Dataset perlu diperluas mencakup mahasiswa dari berbagai latar belakang untuk meningkatkan akurasi model prediksi, sementara integrasi data dari perangkat *wearable* dapat menambah parameter seperti detak jantung dan aktivitas fisik. Dari sisi teknis, model machine learning yang lebih canggih dapat menggantikan algoritma skoring sederhana yang digunakan saat ini. Aplikasi sebaiknya diuji langsung di lingkungan kampus untuk menilai manfaat dan kemudahan penggunaan.. Halaman direktori dapat diperluas dengan fitur pencarian berdasarkan lokasi untuk memudahkan akses ke layanan kesehatan mental yang tepat.