LAPORAN PROYEK AKHIR

Klasifikasi Risiko Stress Mahasiswa di Indonesia Berdasarkan Pola Akademik dan Aktivitas Harian Menggunakan Algoritma Random Forest

Disusun Oleh :

Nabil Hibban Hardian	(A11.2024.15947)
Muhammad Aldo Toni Saputra	(A11.2024.15664)
Muhammad Abid	(A11.2024.15597)
Naia Syafina Hikmayanti	(A11.2024.15554)
Ridlo Fanata Wicaksana	(A11.2024.16059)

FAKULTAS ILMU KOMPUTER

PROGRAM STUDI TEKNIK INFORMATIKA

UNIVERSITAS DIAN NUSWANTORO

SEMARANG

2025

ABSTRAK

Kesehatan mental mahasiswa merupakan isu penting yang dapat memengaruhi kinerja akademik dan kualitas hidup. Beban akademik, pola hidup yang tidak seimbang, serta kondisi sosial-ekonomi menjadi faktor utama yang berkontribusi terhadap munculnya stres pada mahasiswa. Penelitian ini bertujuan untuk membangun model prediksi risiko stres mahasiswa menggunakan pendekatan machine learning berbasis algoritma Random Forest. Dataset yang digunakan mencakup faktor akademik, gaya hidup harian, dan sosial-ekonomi mahasiswa yang telah melalui tahap prapemrosesan dan transformasi data. Model Random Forest dilatih dan dievaluasi menggunakan metrik klasifikasi untuk menilai performa prediksi risiko stres. Hasil penelitian menunjukkan bahwa model mampu mengklasifikasikan mahasiswa ke dalam kategori "Risiko Stres" dan "Sehat" dengan performa yang baik serta mengidentifikasi fitur-fitur dominan yang berpengaruh terhadap tingkat stres. Model yang telah dibangun kemudian diimplementasikan dalam aplikasi berbasis web menggunakan Streamlit sehingga dapat digunakan sebagai alat bantu skrining awal kondisi stres mahasiswa secara interaktif dan praktis di lingkungan kampus.

Kata kunci: kesehatan mental mahasiswa, stres, Random Forest, machine learning, Streamlit

DAFTAR ISI

DAFTAR GAMBAR

DFTAR TABEL

BAB I
PENDAHULUAN

a) Latar Belakang

Kesehatan mental merupakan aspek penting dalam kehidupan mahasiswa karena berpengaruh terhadap motivasi belajar, produktivitas akademik, serta kemampuan beradaptasi di lingkungan perguruan tinggi. Beban tugas akademik, tuntutan prestasi, dan perubahan pola hidup selama masa perkuliahan sering kali menimbulkan stres yang berkelanjutan. Data menunjukkan bahwa lebih dari 60% mahasiswa di Indonesia mengalami gangguan kesehatan mental ringan hingga sedang, sementara sekitar 15% lainnya berada pada kategori berat [1]. Kondisi ini menunjukkan bahwa mahasiswa merupakan kelompok yang rentan terhadap permasalahan stres dan memerlukan perhatian khusus.

Deteksi dini terhadap risiko stres mahasiswa menjadi penting agar institusi pendidikan dapat melakukan intervensi secara tepat waktu. Pendekatan berbasis data memungkinkan identifikasi mahasiswa berisiko dilakukan secara objektif melalui analisis pola akademik dan aktivitas harian, seperti jam belajar, jam tidur, IPK, serta frekuensi olahraga [2]. Oleh karena itu, diperlukan sistem prediksi yang mampu mengolah berbagai faktor tersebut secara akurat.

Berbagai algoritma machine learning telah digunakan dalam penelitian klasifikasi kesehatan mental mahasiswa, antara lain Decision Tree, Logistic Regression, dan Support Vector Machine (SVM) [2]. Namun, masing-masing metode memiliki keterbatasan, seperti kecenderungan overfitting pada Decision Tree, keterbatasan Logistic Regression dalam menangani hubungan nonlinier, serta sensitivitas SVM terhadap pemilihan parameter. Algoritma Random Forest dikembangkan untuk mengatasi kelemahan tersebut dengan menggabungkan banyak pohon keputusan sehingga menghasilkan prediksi yang lebih stabil dan akurat [3].

Pada penelitian ini, algoritma Random Forest digunakan untuk mengklasifikasikan tingkat risiko stres mahasiswa berdasarkan faktor akademik, gaya hidup, dan sosial-ekonomi. Model yang dibangun tidak hanya dievaluasi dari sisi performa prediksi, tetapi juga diimplementasikan dalam bentuk aplikasi berbasis web menggunakan Streamlit. Implementasi ini diharapkan dapat menjadi alat bantu skrining awal bagi pihak kampus dalam memantau kondisi kesehatan mental mahasiswa secara praktis dan berbasis data.

b) Rumusan Masalah

Berdasarkan latar belakang tersebut, rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana menerapkan algoritma Random Forest untuk mengklasifikasikan risiko stres mahasiswa berdasarkan data akademik dan aktivitas harian?
2. Bagaimana performa model Random Forest dalam memprediksi risiko stres mahasiswa berdasarkan metrik evaluasi seperti akurasi, precision, recall, dan confusion matrix?
3. Faktor apa saja yang paling berpengaruh terhadap klasifikasi risiko stres mahasiswa berdasarkan analisis feature importance pada model Random Forest?

c) Pendekatan Pemecahan Masalah

Penelitian ini menggunakan pendekatan klasifikasi biner dengan dua kelas, yaitu **"Risiko Stres"** dan **"Sehat"**. Dataset yang digunakan merepresentasikan karakteristik mahasiswa berdasarkan variabel akademik, gaya hidup, dan sosial-ekonomi, seperti jam belajar per hari, jam tidur per hari, IPK, jumlah tugas besar, frekuensi olahraga, serta kondisi ekonomi keluarga. Dataset diproses melalui tahap pembersihan data, transformasi, dan encoding variabel kategorikal agar dapat digunakan secara optimal oleh model pembelajaran mesin [4].

Model klasifikasi dibangun menggunakan algoritma Random Forest karena kemampuannya dalam menangani data multivariat, menjaga keseimbangan antara precision dan recall, serta mengurangi risiko overfitting melalui mekanisme ensemble learning[3] [4]. Model yang telah dilatih kemudian dievaluasi menggunakan metrik performa klasifikasi. Selanjutnya, model diintegrasikan ke dalam aplikasi berbasis web menggunakan Streamlit sehingga pengguna dapat melakukan prediksi risiko stres secara interaktif melalui antarmuka yang sederhana.

d) Kebaruan Penelitian

Kebaruan penelitian ini terletak pada:

1. Penggabungan faktor akademik, pola hidup harian, dan sosial-ekonomi dalam satu model klasifikasi risiko stres mahasiswa.
2. Pemanfaatan algoritma Random Forest tidak hanya untuk prediksi, tetapi juga untuk mengidentifikasi fitur-fitur dominan yang memengaruhi tingkat stres mahasiswa [5].
3. Implementasi model ke dalam aplikasi Streamlit sebagai sistem skrining awal yang praktis dan mudah digunakan di lingkungan kampus.

e) Peta Jalan Penelitian

Peta jalan penelitian ini dibagi menjadi tiga tahap. Tahap pertama meliputi studi literatur dan perancangan konsep penelitian. Tahap kedua berfokus pada pengolahan data,

pembangunan, serta evaluasi model Random Forest. Tahap ketiga merupakan tahap implementasi model ke dalam aplikasi berbasis Streamlit dan pengujian fungsional aplikasi sebagai sistem pendukung deteksi dini risiko stres mahasiswa.

Gambar 1. 1 Peta Jalan Penelitian

BAB II

Landasan Teori

f) 2.1 Tinjauan Studi

Penelitian mengenai klasifikasi tingkat stres dan kesehatan mental mahasiswa telah banyak dilakukan dengan memanfaatkan metode *machine learning*. Pendekatan ini dinilai efektif karena mampu mengolah data multidimensi yang berkaitan dengan aspek akademik, psikologis, dan sosial mahasiswa, serta mengenali pola kompleks yang sulit dianalisis menggunakan metode konvensional.

Penelitian pada jurnal pertama membahas prediksi tingkat stres dan kesehatan mental mahasiswa menggunakan algoritma **Support Vector Machine (SVM)**. Penelitian tersebut memanfaatkan variabel akademik, beban tugas, pola tidur, serta kondisi emosional mahasiswa sebagai fitur input. Hasil penelitian menunjukkan bahwa algoritma SVM mampu menghasilkan tingkat akurasi yang baik dalam mengklasifikasikan kondisi stres mahasiswa, terutama pada

data berdimensi tinggi dengan pola yang kompleks, sehingga SVM dinilai efektif dalam permasalahan klasifikasi kesehatan mental [5].

Penelitian selanjutnya mengkaji klasifikasi kesehatan mental mahasiswa dengan menggunakan beberapa algoritma *machine learning*, antara lain **Decision Tree**, **Logistic Regression**, dan **Random Forest**.

Hasil penelitian menunjukkan bahwa algoritma **Random Forest** memberikan performa yang paling optimal dibandingkan algoritma lainnya. Keunggulan ini disebabkan oleh kemampuan Random Forest dalam menangani hubungan nonlinier antar variabel serta kemampuannya dalam mengurangi risiko *overfitting* melalui mekanisme *ensemble learning*.

Penelitian lain yang menjadi rujukan menerapkan algoritma **Random Forest** untuk mendeteksi tingkat stres mahasiswa dengan mempertimbangkan faktor akademik, aktivitas harian, dan kondisi sosial. Studi tersebut menunjukkan bahwa Random Forest tidak hanya menghasilkan akurasi klasifikasi yang tinggi, tetapi juga mampu mengidentifikasi fitur-fitur yang paling berpengaruh terhadap tingkat stres mahasiswa melalui analisis *feature importance*.

Berdasarkan beberapa penelitian terdahulu tersebut, dapat disimpulkan bahwa algoritma *machine learning*, khususnya **Random Forest**, memiliki keunggulan dalam klasifikasi tingkat stres dan kesehatan mental mahasiswa. Algoritma ini mampu mengolah data yang kompleks, meningkatkan stabilitas prediksi, serta memberikan informasi penting terkait faktor-faktor dominan penyebab stres. Oleh karena itu, Random Forest dipilih sebagai metode utama dalam penelitian ini.

2.2 Tinjauan Pustaka

1.1.1. Data Mining

Data mining merupakan proses penggalian pola, hubungan, dan pengetahuan baru dari kumpulan data berukuran besar menggunakan teknik statistik, pembelajaran mesin, dan basis data. Tahapan utama dalam data mining meliputi *data cleaning, data integration, data transformation, data mining, serta pattern evaluation* [6].

Dalam penelitian ini, data mining digunakan untuk mengolah data mahasiswa yang terdiri dari atribut numerik dan kategorikal guna menemukan pola yang berkaitan dengan tingkat stres mahasiswa.

1.1.2. Machine Learning

Machine learning adalah cabang kecerdasan buatan yang memungkinkan sistem mempelajari pola dari data dan membuat keputusan tanpa instruksi eksplisit. Salah satu pendekatan yang umum digunakan adalah

supervised learning, yaitu metode pembelajaran yang menggunakan data berlabel sebagai acuan pelatihan model [6].

Pada penelitian ini, supervised learning diterapkan untuk membangun model klasifikasi yang dapat memprediksi label tingkat stres mahasiswa berdasarkan fitur-fitur akademik dan kebiasaan harian yang dimiliki.

1.1.3. Random Forest

Random Forest merupakan algoritma *ensemble learning* yang bekerja dengan membangun sejumlah *decision tree* selama proses pelatihan dan menghasilkan prediksi berdasarkan mekanisme *majority voting*. Algoritma ini memiliki keunggulan dalam mengurangi *overfitting*, mampu menangani data berdimensi tinggi, serta cukup robust terhadap *noise* pada data. Berdasarkan implementasi pada notebook penelitian ini, Random Forest digunakan sebagai model utama dengan pengaturan parameter seperti jumlah pohon (*n_estimators*) dan kedalaman pohon (*max_depth*) untuk meningkatkan performa klasifikasi.

1.1.4. Pra-pemrosesan Data

Pra-pemrosesan data merupakan tahap penting dalam data mining dan machine learning untuk meningkatkan kualitas data sebelum proses pelatihan model. Tahapan yang dilakukan dalam penelitian ini meliputi pemeriksaan *missing value*, duplikasi data, serta analisis *outlier* menggunakan metode visualisasi *boxplot*.

Selain itu, dilakukan **normalisasi data numerik** menggunakan metode *standardization* agar setiap fitur berada pada skala yang sebanding, sehingga tidak mendominasi proses pelatihan model [7][8].

1.1.5. One-Hot Encoding

One-Hot Encoding adalah teknik transformasi data kategorikal menjadi bentuk numerik dengan merepresentasikan setiap kategori sebagai vektor biner. Teknik ini banyak digunakan dalam algoritma machine learning yang tidak dapat memproses data kategorikal secara langsung.

Dalam penelitian ini, One-Hot Encoding diterapkan pada fitur kategorikal menggunakan pustaka *scikit-learn* untuk memastikan seluruh data dapat diproses oleh algoritma Random Forest secara optimal.

1.1.6. Klasifikasi

Klasifikasi merupakan proses penentuan kelas suatu objek berdasarkan atribut tertentu. Pada penelitian ini, klasifikasi digunakan untuk menentukan label tingkat stres mahasiswa berdasarkan variabel yang tersedia pada dataset. Evaluasi performa model klasifikasi dilakukan menggunakan metrik **akurasi**, **F1-score**, **confusion matrix**, dan **classification report**, yang umum digunakan untuk mengukur kinerja model supervised learning [5]

1.1.7. Python

Python merupakan bahasa pemrograman tingkat tinggi yang banyak digunakan dalam pengolahan data dan pengembangan model machine learning. Python didukung oleh berbagai pustaka seperti *pandas* untuk manipulasi data, *numpy* untuk komputasi numerik, serta *scikit-learn* untuk pembangunan dan evaluasi model machine learning [9]. Pada penelitian ini, Python digunakan sebagai bahasa utama dalam seluruh proses analisis dan pemodelan.

1.2.Perangkat Lunak Pendukung

1.2.1. Visual Studio Code

Visual Studio Code adalah *code editor* lintas platform yang mendukung berbagai bahasa pemrograman, termasuk Python. Aplikasi ini digunakan karena memiliki antarmuka yang ringan, fitur ekstensi yang lengkap, serta mendukung proses debugging dan pengelolaan proyek dengan baik [10].

1.2.2. Google Colab

Google Colab adalah lingkungan komputasi berbasis *cloud* yang digunakan untuk menjalankan eksperimen *machine learning*. Platform ini memudahkan proses pelatihan model tanpa memerlukan spesifikasi perangkat keras yang tinggi, serta mendukung kolaborasi dan penyimpanan berbasis daring.

1.2.3. Streamlit

Streamlit merupakan *framework* Python yang digunakan untuk membangun aplikasi web interaktif berbasis data. Dalam penelitian ini, Streamlit digunakan untuk menampilkan hasil klasifikasi tingkat stres mahasiswa sehingga dapat dimanfaatkan sebagai prototipe sistem pendukung keputusan.

BAB III
METODE PENELITIAN

3.1 Jenis Penelitian

Penelitian ini menggunakan pendekatan kuantitatif dengan metode **machine learning** untuk memprediksi risiko stres pada mahasiswa. Jenis penelitian ini termasuk dalam kategori penelitian terapan (applied research), di mana hasil penelitian dapat langsung diimplementasikan dalam bentuk sistem prediksi berbasis web.

Metode yang digunakan adalah klasifikasi supervised learning dengan algoritma Random Forest Classifier, yaitu metode ensemble learning yang menggabungkan beberapa decision tree untuk menghasilkan prediksi yang lebih akurat dan stabil.

3.2 Populasi dan Sampel Penelitian

Populasi dalam penelitian ini adalah mahasiswa aktif di berbagai perguruan tinggi di Indonesia dengan rentang usia 18-25 tahun. Teknik pengambilan sampel menggunakan purposive sampling, di mana responden dipilih berdasarkan kriteria sebagai mahasiswa aktif dengan latar belakang akademik yang bervariasi dari berbagai program studi.

3.3 Variabel Penelitian

3.3.1 Variabel Independen (Fitur)

Penelitian ini menggunakan dua jenis fitur:

A. Fitur Numerik:

No.	Nama Fitur	Deskripsi
1	Umur	Usia mahasiswa dalam tahun
2	Jam Belajar per Hari	Durasi waktu belajar setiap hari
3	Jam Tidur per Hari	Durasi waktu tidur setiap hari
4	IPK	Indeks Prestasi Kumulatif
5	Jumlah Tugas Besar per Minggu	Banyaknya tugas besar dalam seminggu

Tabel 3. 1 Fitur Numerik

B. Fitur Kategorikal:

No.	Nama Fitur	Deskripsi
1	Gender	Jenis kelamin mahasiswa
2	Jurusan/Program Studi	Program studi yang diikuti
3	Frekuensi Olahraga	Tingkat keseringan berolahraga

4	Pemasukan Keluarga	Tingkat ekonomi keluarga
5	Status Hubungan	Status relasi romantis

Tabel 3. 2 Fitur Kategorikal

3.3.2 Variabel Dependen (Label)

Variabel dependen adalah **Label** yang menunjukkan status risiko stres dengan dua kategori:

- **Sehat**: Mahasiswa dengan risiko stres rendah
- **Risiko Stres**: Mahasiswa dengan potensi risiko stres tinggi

3.4 Teknik Pengumpulan Data

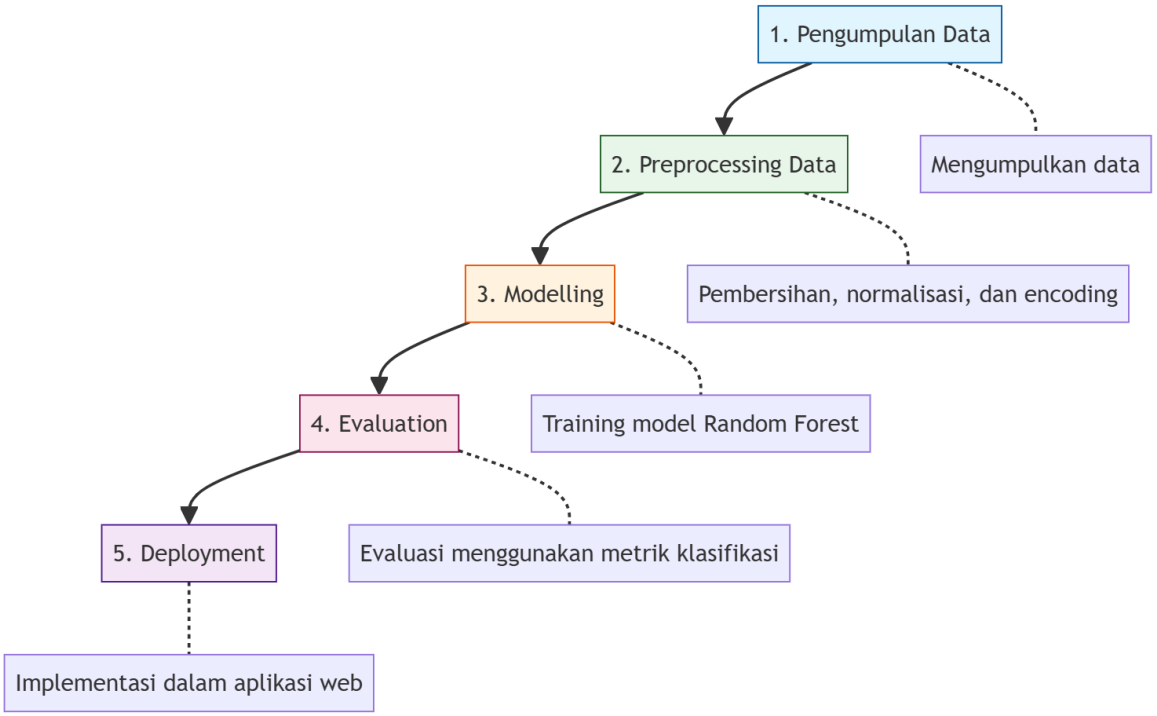
Data dikumpulkan dan mencakup:

1. Data demografi (gender, umur, program studi)
2. Kebiasaan akademik (jam belajar, IPK, jumlah tugas besar)
3. Gaya hidup (jam tidur, frekuensi olahraga)
4. Faktor sosial-ekonomi (pemasukan keluarga, status hubungan)

Data disimpan dalam format CSV (*Comma Separated Values*) untuk pemrosesan lebih lanjut.

3.5 Tahapan Penelitian

Penelitian ini mengikuti tahapan CRISP-DM **(***Cross-Industry Standard Process for Data Mining***)**:



Gambar 3. 1 Tahapan Penelitian

3.6 Teknik Analisis Data

3.6.1 Preprocessing Data

Preprocessing yang akan diterapkan meliputi:

- **Pembersihan data**: Penanganan missing value dan duplikasi
- **Normalisasi**: Menggunakan Z-Score Normalization untuk fitur numerik
- **Encoding**: Menggunakan One-Hot Encoding untuk fitur kategorikal
- **Splitting data**: Pembagian data menjadi training set dan testing set

3.6.2 Algoritma Klasifikasi

Random Forest Classifier dipilih karena:

1. Mampu menangani fitur campuran (numerik dan kategorikal)
2. Tidak mudah overfitting karena menggunakan ensemble dari banyak tree
3. Memberikan prediksi yang akurat dan stabil
4. Robust terhadap outlier

3.6.3 Metrik Evaluasi

Model akan dievaluasi menggunakan:

Metrik	Fungsi
Accuracy	Mengukur proporsi prediksi yang benar
F1-Score	Harmonic mean dari precision dan recall
Confusion Matrix	Menunjukkan distribusi prediksi (TP, TN, FP, FN)
Classification Report	Laporan precision, recall, F1 per kelas

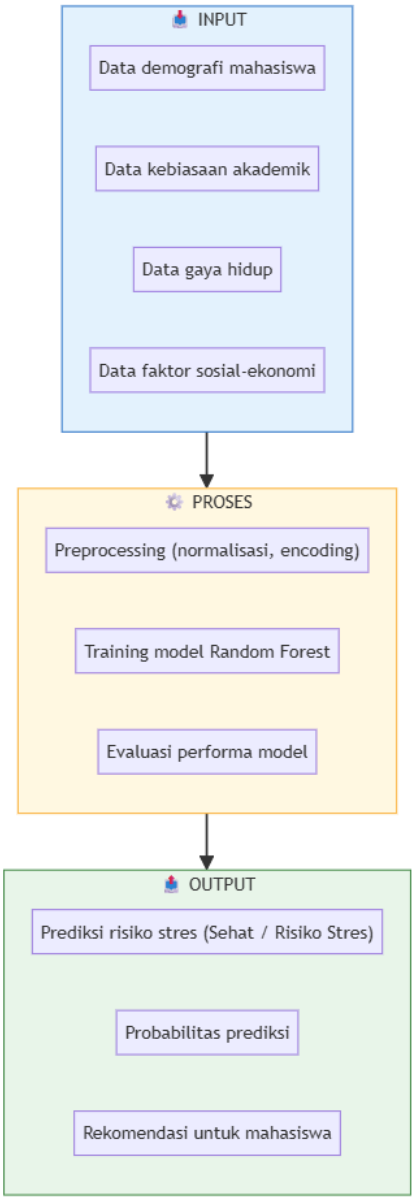
Tabel 3. 3 Metrik Evaluasi

3.7 Tools dan Teknologi

Komponen	Teknologi
Bahasa Pemrograman	Python
Manipulasi Data	Pandas, NumPy
Machine Learning	Scikit-learn
Web Framework	Streamlit
Visualisasi	Plotly

Tabel 3. 4 Tools dan Teknologi

3.8 Kerangka Berpikir



Gambar 3. 2 Kerangka Berpikir

BAB IV
HASIL DAN PEMBAHASAN

4.1 Pengumpulan Data

Pengumpulan dataset merupakan tahap pertama yang harus dilakukan pada proyek ini. Dataset yang digunakan dalam penelitian ini merupakan data sintetis yang dirancang untuk merepresentasikan karakteristik mahasiswa di Indonesia dengan total 3000 data. Dataset ini mencakup berbagai variabel yang berkaitan dengan faktor akademik, pola hidup harian, dan kondisi sosial-ekonomi mahasiswa. Dataset terdiri dari 10 atribut utama dan 1 atribut sebagai label yang meliputi:

1. Gender (jenis kelamin mahasiswa)
2. Umur (usia mahasiswa dalam rentang 18-25 tahun)
3. Jurusan/Program Studi (program studi yang diambil)
4. Jam Belajar per Hari (durasi belajar harian dalam jam)
5. Jam Tidur per Hari (durasi tidur harian dalam jam)
6. IPK (Indeks Prestasi Kumulatif dengan rentang 0.0-4.0)
7. Jumlah Tugas Besar per Minggu (beban tugas yang harus diselesaikan)
8. Frekuensi Olahraga (tingkat aktivitas fisik: Jarang/Kadang/Sering)
9. Pemasukan Keluarga (status ekonomi keluarga: Rendah/Sedang/Tinggi)
10. Status Hubungan (kondisi relasi romantis: Jomblo/Dalam hubungan)
11. Label (kategori risiko stres: Sehat/Risiko Stres).

Data disimpan dalam format CSV dengan pemisah semicolon (;) dan dimuat menggunakan fungsi read_csv() dari library Pandas. Proses loading data dilakukan dengan kode berikut

```
def load_data():
    """Load and preprocess the dataset"""
    df = pd.read_csv('dataset.csv', sep=';')
    return df
```

Gambar 4. 1 Proses load data

```
Gender;Umur;Jurusan/Program Studi;Jam Belajar per Hari;Jam Tidur per Hari;IPK;Jumlah Tugas Besar p
Laki-laki;20;Teknik Informatika;6;8;03.13;1;Jarang;Sedang;Dalam hubungan;Sehat
Perempuan;18;Hukum;5;6;0,188194444;1;Jarang;Rendah;Jomblo;Sehat
Perempuan;21;Desain Komunikasi Visual;6;4;0,127777778;1;Kadang;Sedang;Dalam hubungan;Risiko Stres
Laki-laki;24;Kedokteran;3;3;02.44;0;Sering;Sedang;Dalam hubungan;Sehat
Laki-laki;18;Hukum;4;4;03.35;5;Sering;Tinggi;Jomblo;Sehat
Perempuan;23;Desain Komunikasi Visual;2;7;03.52;5;Jarang;Tinggi;Dalam hubungan;Sehat
```

Gambar 4. 2 Dataset mahasiswa

Setelah data berhasil dimuat, dilakukan pemeriksaan awal terhadap struktur data untuk memahami karakteristik dataset yang akan digunakan dalam pembangunan model.

4.2 Prepocessing Data

Tahap preprocessing data merupakan proses krusial yang bertujuan untuk mempersiapkan data mentah agar layak digunakan dalam proses pemodelan machine learning. Preprocessing dilakukan untuk memastikan kualitas data, konsistensi format, serta kesesuaian tipe data dengan kebutuhan algoritma yang digunakan. Pada penelitian ini, tahapan preprocessing meliputi pemuatan dataset, pembersihan data IPK, normalisasi data numerik, pendefinisian fitur, pemisahan fitur dan label, serta encoding fitur kategorikal.

4.2.1 Loading Dataset

Dataset dimuat dari sebuah file berekstensi CSV menggunakan library Pandas. Proses pemuatan data dilakukan dengan menentukan separator berupa titik koma (;) karena format dataset menggunakan pemisah tersebut. Untuk meningkatkan efisiensi aplikasi berbasis Streamlit, digunakan decorator @st.cache_data agar dataset tidak perlu dimuat ulang setiap kali aplikasi dijalankan ulang.

```
@st.cache_data
def load_data():
    """Load and preprocess the dataset"""
    df = pd.read_csv('dataset.csv', sep=';')
    return df
```

Gambar 4. 3 Proses load dataset

Decorator @st.cache_data berfungsi menyimpan data ke dalam cache sehingga proses pemuatan dataset hanya dilakukan satu kali selama sesi berjalan. Parameter sep=';' digunakan untuk menyesuaikan format pemisah kolom pada file CSV.

4.2.2 Pembersihan Data IPK

Kolom IPK pada dataset memiliki variasi format penulisan, seperti penggunaan koma sebagai pemisah desimal serta adanya nilai dalam bentuk pecahan waktu. Oleh karena itu, diperlukan proses pembersihan dan standarisasi agar seluruh nilai IPK dapat dikonversi menjadi tipe data numerik yang valid. Proses ini diimplementasikan melalui fungsi clean_ipk().

```python
def clean_ipk(value):
    """Clean IPK value - convert various formats to float"""
    if pd.isna(value):
        return np.nan

    # Convert to string
    val_str = str(value).strip()

    # Replace comma with dot
    val_str = val_str.replace(',', '.')

    try:
        val_float = float(val_str)
        # If value is less than 1, it might be in decimal format like 0.188194444
        # These seem to be time fractions, convert to proper IPK scale (0-4)
        if val_float < 1:
            # These values appear to be time fractions, multiply by appropriate factor
            # Based on the data pattern, values like 0.188194444 should map to ~2.7-3.5 range
            val_float = val_float * 20  # Scale to approximate IPK range
            if val_float > 4:
                val_float = 4.0
        return val_float
    except:
        return np.nan
```

Gambar 4. 4 Proses clean IPK

Proses pembersihan data IPK dilakukan melalui serangkaian tahapan sistematis untuk memastikan konsistensi dan validitas nilai. Pertama, data IPK diperiksa untuk mendeteksi nilai kosong (NaN), yang kemudian tetap dipertahankan sebagai NaN. Selanjutnya, nilai IPK dikonversi ke dalam format string dan seluruh spasi yang tidak diperlukan dihapus. Untuk menyeragamkan format desimal, tanda koma (,) diganti dengan titik (.). Setelah itu, nilai IPK dikonversi ke dalam tipe data float. Apabila nilai IPK yang diperoleh kurang dari 1, nilai tersebut diasumsikan sebagai pecahan waktu dan dilakukan proses penskalaan dengan mengalikan nilai tersebut dengan 20 agar berada dalam rentang IPK 0–4. Jika pada salah satu tahap terjadi kesalahan dalam proses konversi, maka nilai IPK akan dikembalikan sebagai NaN.

4.2.3 Preprocessing Data Utama

Setelah pembersihan IPK, dilakukan preprocessing menyeluruh terhadap dataset menggunakan fungsi preprocess_data()

```python
def preprocess_data(df):
    """Preprocess the dataframe similar to notebook"""
    df = df.copy()

    # Clean IPK column
    df['IPK'] = df['IPK'].apply(clean_ipk)

    # Convert numeric columns to proper numeric type
    numeric_features = ['Jam Belajar per Hari', 'Jam Tidur per Hari', 'IPK', 'Jumlah Tugas Besar per Minggu']
    df[numeric_features] = df[numeric_features].apply(pd.to_numeric, errors='coerce')

    # Normalize numeric features (Z-score normalization like in notebook)
    df[numeric_features] = (df[numeric_features] - df[numeric_features].mean()) / df[numeric_features].std()

    return df
```

Gambar 4. 5 Preprocessing data

Proses preprocessing data dilakukan melalui beberapa tahapan terstruktur untuk memastikan kualitas data sebelum digunakan dalam pemodelan. Tahap awal dimulai dengan pembuatan salinan DataFrame guna menjaga keutuhan dan mencegah perubahan pada data asli. Selanjutnya, kolom IPK dibersihkan menggunakan fungsi clean_ipk() untuk menstandarkan format dan menangani nilai yang tidak valid. Setelah itu, seluruh fitur numerik dikonversi ke dalam tipe data numerik, di mana nilai yang tidak dapat dikonversi akan diubah menjadi NaN. Tahap berikutnya adalah normalisasi data numerik menggunakan metode Z-score normalization, yaitu dengan mengurangkan nilai fitur terhadap rata-rata (μ) dan membaginya dengan simpangan baku (σ). Proses normalisasi ini bertujuan untuk menyamakan skala antarfitur sehingga setiap variabel memiliki kontribusi yang seimbang dan pada akhirnya dapat meningkatkan performa model yang digunakan.

4.2.4 Definisi Fitur

Fitur dalam project ini dibagi menjadi dua kategori, yaitu fitur numerik dan fitur kategorikal.

```
# Define features exactly like notebook
numeric_features = ['Umur', 'Jam Belajar per Hari', 'Jam Tidur per Hari', 'IPK', 'Jumlah Tugas Besar per Minggu']
categorical_features = ['Gender', 'Jurusan/Program Studi', 'Frekuensi Olahraga', 'Pemasukan Keluarga', 'Status Hubungan']
```

Gambar 4. 6 Proses definisi fitur

Fitur numerik mencakup variabel-variabel yang dapat diukur secara langsung dalam bentuk angka. Variabel umur berada pada rentang 18 hingga 25 tahun. Jam belajar per hari memiliki rentang antara 1 hingga 7 jam, sedangkan jam tidur per hari berkisar antara 3 hingga 9 jam. Nilai IPK direpresentasikan dalam skala 0,0 hingga 4,0 sebagai indikator capaian akademik. Selain itu, jumlah tugas besar per minggu berada pada rentang 0 hingga 5 dan menggambarkan beban akademik yang diterima mahasiswa.

Fitur kategorikal merepresentasikan karakteristik non-numerik yang bersifat deskriptif. Fitur ini meliputi gender, jurusan atau program studi, frekuensi olahraga, tingkat pemasukan keluarga, serta status hubungan. Seluruh fitur kategorikal tersebut digunakan untuk menangkap variasi latar belakang dan gaya hidup responden yang berpotensi memengaruhi analisis maupun pemodelan yang dilakukan.

4.2.5 Encoding Variabel Kategorikal

Algoritma Random Forest memerlukan input dalam bentuk numerik. Oleh karena itu, variabel kategorikal perlu dikonversi menggunakan teknik One-Hot Encoding. Teknik ini mengubah setiap kategori menjadi kolom biner (0 atau 1).

Variabel kategorikal yang di-encode:

- Gender
- Jurusan/Program Studi
- Frekuensi Olahraga
- Pemasukan Keluarga
- Status Hubungan

Gambar Proses encoding dilakukan menggunakan *OneHotEncoder* dari scikit-learn:

```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', 'passthrough', numeric_features),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
    ],
    remainder='passthrough'
)
```

Gambar 4. 7 Proses One Hot Encoder

4.2.6 Pembagian Data

Tahap selanjutnya adalah split data, data dibagi menjadi data training (80%) dan data testing (20%) menggunakan fungsi train_test_split():

```
# Prepare data
X = df.drop('Label', axis=1)
y = df['Label']

# Split data - same as notebook
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Gambar 4. 8 Proses split data

Parameter random_state=42 digunakan untuk memastikan hasil pembagian data konsisten setiap kali kode dijalankan.

4.3 Modelling

Setelah proses preprocessing selesai dilakukan, tahap selanjutnya adalah pembangunan model klasifikasi. Pada penelitian ini digunakan algoritma **Random Forest Classifier** sebagai metode utama untuk memprediksi risiko stres mahasiswa.

4.3.1 Pemilian Algoritma

Random Forest dipilih karena memiliki beberapa keunggulan:

1. Dapat menggabungkan prediksi dari banyak decision tree sehingga menghasilkan prediksi yang lebih stabil
2. Random Forest tidak mudah overfitting
3. Mampu menangani kombinasi fitur numerik dan kategorikal dengan baik
4. Dapat mengidentifikasi fitur mana yang paling berpengaruh terhadap prediksi
5. Tidak terlalu sensitif terhadap nilai outlier dalam data

4.3.2 Arsitektur Model

Model Random Forest yang dibangun memiliki konfigurasi sebagai berikut:

```
model = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(
        n_estimators=200,
        max_depth=4,
        random_state=42,
        n_jobs=-1
    ))
])
```

Gambar 4. 9 Pipeline Random Forest

Model Random Forest yang dibangun menggunakan 200 pohon keputusan, artinya model akan membuat 200 pohon yang berbeda-beda untuk menghasilkan prediksi yang lebih akurat. Setiap pohon dibatasi kedalamannya maksimal 4 tingkat saja, pembatasan ini berguna untuk mencegah model terlalu "menghafal" data training yang bisa membuat prediksi pada data baru menjadi kurang akurat.

Agar hasil training bisa diulang dengan konsisten, digunakan nilai random_state sebesar 42. Parameter ini membuat proses acak dalam algoritma selalu menghasilkan pola yang sama setiap kali dijalankan. Selain itu, training juga memanfaatkan semua inti prosesor komputer (n_jobs=-1) sehingga proses berjalan lebih cepat.

4.3.3 Proses Training

Proses training dilakukan dengan melatih model menggunakan data training yang telah diproses:

```
# Train model
model.fit(X_train, y_train)
```

Gambar 4. 10 Proses Train

Selama proses training, Random Forest bekerja dengan beberapa tahapan. Pertama, algoritma membuat beberapa subset data training yang berbeda untuk setiap pohon. Kedua, 200 pohon keputusan dibangun secara bersamaan menggunakan semua inti prosesor. Ketiga, setiap pohon dilatih dengan memilih fitur-fitur secara acak agar setiap pohon memiliki karakteristik yang berbeda. Terakhir, hasil prediksi dari 200 pohon digabungkan melalui sistem voting, di mana hasil prediksi yang paling banyak dipilih akan menjadi prediksi akhir model.

4.4 Evaluation

Setelah model berhasil dibangun dan dilatih, tahap selanjutnya adalah evaluasi performa model menggunakan data testing. Evaluasi dilakukan untuk mengukur seberapa baik model dapat memprediksi risiko stres mahasiswa pada data yang belum pernah dilihat sebelumnya.

4.4.1 Metrik Evaluasi

Model dievaluasi menggunakan beberapa metrik standar klasifikasi:

- Accuracy (Akurasi) mengukur proporsi prediksi yang benar dari keseluruhan prediksi:

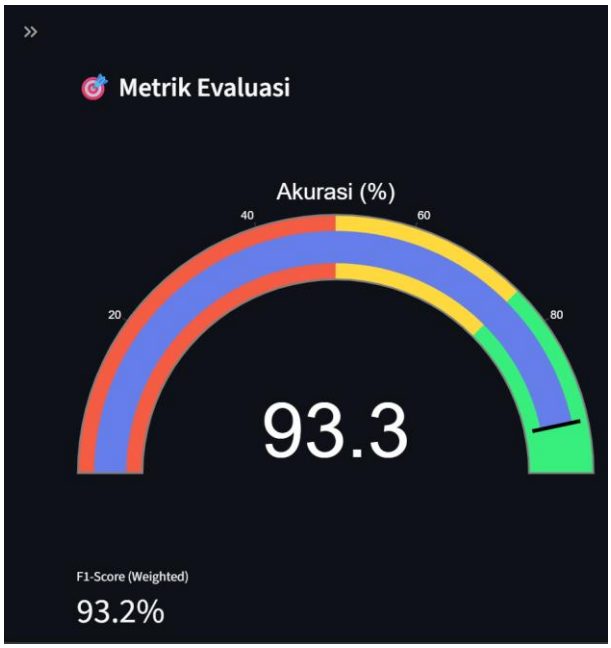
```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
```

Gambar 4. 11 Matrix Akurasi

- F1-Score merupakan harmonic mean dari precision dan recall, yang memberikan keseimbangan antara kedua metrik:

```
f1 = f1_score(y_test, y_pred, average='weighted')
```

Gambar 4. 12 MatrixF1-Score



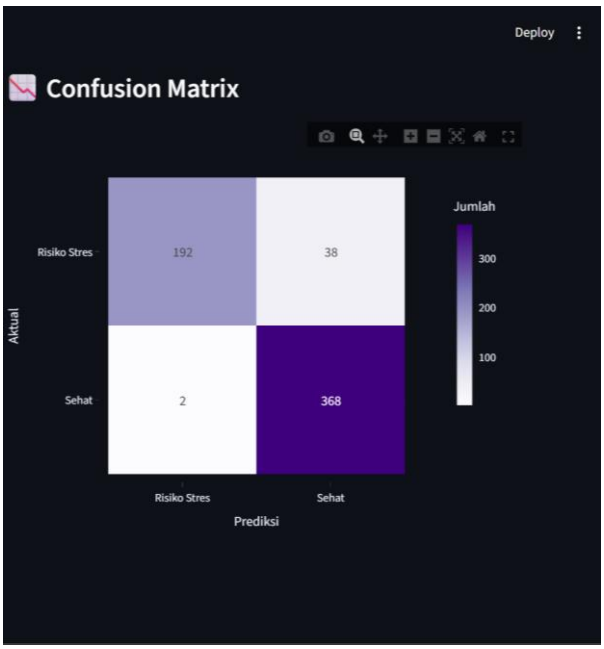
Gambar 4. 13 Matrix Evaluasi

4.4.2 Confusion Matrix

Confusion matrix memberikan gambaran detail tentang performa klasifikasi model dengan menampilkan jumlah prediksi yang benar dan salah untuk setiap kelas:

```
cm = confusion_matrix(y_test, y_pred)
```

Gambar 4. 14 Proses Confusion Matrix



Gambar 4. 15 Confusion Matrix

Confusion matrix menampilkan:

- True Positive (TP): Mahasiswa dengan risiko stres yang diprediksi benar
- True Negative (TN): Mahasiswa sehat yang diprediksi benar
- False Positive (FP): Mahasiswa sehat yang diprediksi risiko stres
- False Negative (FN): Mahasiswa risiko stres yang diprediksi sehat

4.5 Deployment

Setelah model berhasil dievaluasi dan menunjukkan performa yang baik, tahap selanjutnya adalah deployment aplikasi. Tahap ini penting dilakukan agar hasil penelitian dapat bermanfaat secara luas dan mudah diakses oleh pengguna.

4.5.1 Ferameworkk Deployment

Aplikasi dideploy menggunakan **Streamlit**, sebuah framework Python yang dirancang khusus untuk membangun aplikasi web interaktif berbasis data dengan cepat. Streamlit dipilih karena:

- Tidak memerlukan pengetahuan mendalam tentang web development
- Mendukung widget interaktif seperti slider, selectbox, dan button
- Dapat membuat aplikasi web dalam waktu singkat
- Aplikasi langsung ter-update ketika kode diubah
- Mendukung berbagai library visualisasi seperti Plotly, Matplotlib

4.5.2 Struktur Aplikasi

Aplikasi dibangun dengan struktur modular yang terdiri dari beberapa komponen utama:

- Caching System Untuk meningkatkan performa

  Digunakan decorator @st.cache_data dan @st.cache_resource untuk menyimpan data dan model di memory:

```python
@st.cache_data
def load_data():
    """Load and preprocess the dataset"""
    df = pd.read_csv('dataset.csv', sep=';')
    return df
```

Gambar 4. 16 st.cache_data

```python
@st.cache_resource
def train_model(_df):
    """Train the Random Forest model - exactly like the notebook"""
    # Preprocess data
    df = preprocess_data(_df)

    # Define features exactly like notebook
    numeric_features = ['Umur', 'Jam Belajar per Hari', 'Jam Tidur per Hari', 'IPK', 'Jumlah Tugas Besar per Minggu']
    categorical_features = ['Gender', 'Jurusan/Program Studi', 'Frekuensi Olahraga', 'Pemasukan Keluarga', 'Status Hubungan']

    # Prepare data
    X = df.drop('Label', axis=1)
    y = df['Label']

    # Split data - same as notebook
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Create preprocessor - matching notebook
    preprocessor = ColumnTransformer(
        transformers=[
            ('num', 'passthrough', numeric_features),
            ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
        ],
        remainder='passthrough'
    )

    # Create pipeline with same parameters as notebook
    model = Pipeline([
        ('preprocessor', preprocessor),
        ('classifier', RandomForestClassifier(
            n_estimators=200,
            max_depth=4,
            random_state=42,
            n_jobs=-1
        ))
    ])

    # Train model
    model.fit(X_train, y_train)

    # Evaluate
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred, average='weighted')
    cm = confusion_matrix(y_test, y_pred)

    # Store preprocessing stats for prediction
    raw_df = _df.copy()
    raw_df['IPK'] = raw_df['IPK'].apply(clean_ipk)
    numeric_cols = ['Jam Belajar per Hari', 'Jam Tidur per Hari', 'IPK', 'Jumlah Tugas Besar per Minggu']
    raw_df[numeric_cols] = raw_df[numeric_cols].apply(pd.to_numeric, errors='coerce')

    stats = {
        'mean': raw_df[numeric_cols].mean().to_dict(),
        'std': raw_df[numeric_cols].std().to_dict()
    }

    return model, accuracy, f1, cm, X_test, y_test, stats
```

Gambar 4. 17 st.cache_resource

- Page Navigation

  Aplikasi menggunakan sidebar navigation untuk berpindah antar halaman:

```python
with st.sidebar:
    st.image("https://img.icons8.com/fluency/96/brain.png", width=80)
    st.markdown("### 📊 Navigasi")
    page = st.radio(
        "Pilih Halaman:",
        ["🏠 Beranda", "🔮 Prediksi", "📈 Analisis Data", "📊 Performa Model"],
        label_visibility="collapsed"
    )
```

Gambar 4. 18 st.sidebar

- Custom CSS Styling
  Tampilan aplikasi dipercantik menggunakan custom CSS untuk meningkatkan user experience:

```python
st.markdown("""
<style>
    .main-header {
        font-size: 2.5rem;
        font-weight: 700;
        background: linear-gradient(90deg, #667eea 0%, #764ba2 100%);
        -webkit-background-clip: text;
        -webkit-text-fill-color: transparent;
        text-align: center;
        margin-bottom: 2rem;
    }
    .sub-header {
        font-size: 1.2rem;
        color: #6c757d;
        text-align: center;
        margin-bottom: 2rem;
    }
    .metric-card {
        background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
        border-radius: 15px;
        padding: 20px;
        color: white;
        text-align: center;
        box-shadow: 0 4px 15px rgba(102, 126, 234, 0.4);
    }
    .stButton>button {
        background: linear-gradient(90deg, #667eea 0%, #764ba2 100%);
        color: white;
        border: none;
        border-radius: 10px;
        padding: 0.75rem 2rem;
        font-weight: 600;
        transition: all 0.3s ease;
    }
    .stButton>button:hover {
        transform: translateY(-2px);
        box-shadow: 0 4px 15px rgba(102, 126, 234, 0.4);
    }
    .result-box {
        padding: 20px;
        border-radius: 15px;
        margin: 20px 0;
        text-align: center;
    }
    .result-sehat {
        background: linear-gradient(135deg, #11998e 0%, #38ef7d 100%);
        color: white;
    }
    .result-stres {
        background: linear-gradient(135deg, #eb3349 0%, #f45c43 100%);
        color: white;
    }
</style>
""", unsafe_allow_html=True)
```

Gambar 4. 19 Custom CSS

4.5.3 Integrasi Model

Model yang telah dilatih diintegrasikan ke dalam aplikasi dengan memastikan preprocessing yang konsisten antara tahap training dan prediksi. Ketika pengguna memasukkan data melalui form, nilai-nilai numerik seperti jam belajar, jam tidur, IPK, dan jumlah tugas harus dinormalisasi terlebih dahulu menggunakan statistik yang sama dengan data training (mean dan standar deviasi).

Proses normalisasi dilakukan dengan rumus Z-score untuk setiap fitur numerik. Setelah itu, seluruh data input digabungkan dalam satu DataFrame yang mencakup fitur numerik yang sudah dinormalisasi dan fitur kategorikal dalam format aslinya. Model kemudian melakukan prediksi

menggunakan fungsi predict() untuk mendapatkan kelas prediksi (Sehat atau Risiko Stres) dan predict_proba() untuk mendapatkan probabilitas setiap kelas. Sistem juga mengidentifikasi indeks kelas yang tepat untuk memastikan probabilitas ditampilkan sesuai dengan label yang benar.

```
# Make prediction
prediction = model.predict(input_data)[0]
proba = model.predict_proba(input_data)[0]
```

Gambar 4. 20 Prediksi dan Probabilitas

4.5.4 Error Handling

Implementasi error handling dilakukan untuk menangani berbagai kemungkinan kesalahan yang dapat terjadi saat aplikasi berjalan. Penanganan error dibagi menjadi beberapa tingkat untuk memberikan informasi yang jelas kepada pengguna.

```
# Load data and train model
try:
    df = load_data()
    model, accuracy, f1, cm, X_test, y_test, stats = train_model(df)
except FileNotFoundError:
    st.error("⚠ File dataset.csv tidak ditemukan. Pastikan file dataset berada di direktori yang sama dengan aplikasi.")
    return
except Exception as e:
    st.error(f"⚠ Terjadi kesalahan: {str(e)}")
    return
```

Gambar 4. 21 Error Handling

Error FileNotFoundError menangani kondisi ketika file dataset tidak ditemukan di direktori aplikasi. Sedangkan Exception umum menangani kesalahan lainnya seperti format data yang salah, masalah dalam proses training, atau error dalam preprocessing. Dengan error handling ini, aplikasi tidak akan crash secara tiba-tiba dan pengguna mendapatkan informasi yang membantu untuk mengatasi masalah.

4.5.5 Deployment Process

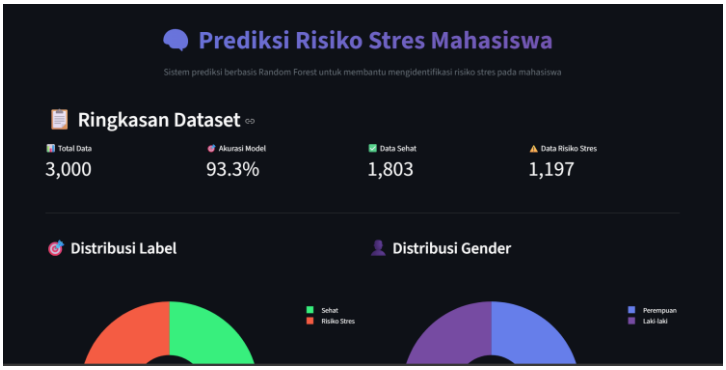
Aplikasi dapat dijalankan secara lokal maupun di-deploy ke cloud platform. Untuk menjalankan aplikasi secara lokal, cukup menggunakan perintah streamlit run app.py di terminal, kemudian aplikasi akan terbuka di browser pada alamat localhost port 8501.

Untuk deployment ke cloud, aplikasi dapat di-deploy ke berbagai platform seperti Streamlit Cloud, Heroku, Google Cloud Platform, atau AWS. Streamlit Cloud menjadi pilihan yang paling mudah karena terintegrasi langsung dengan GitHub dan tidak memerlukan konfigurasi server yang rumit. Proses deployment memastikan bahwa aplikasi dapat diakses oleh pengguna dari berbagai lokasi melalui internet browser, sehingga dapat digunakan oleh mahasiswa dan pihak kampus kapan saja dan di mana saja untuk melakukan skrining awal risiko stres.

4.6 Pengenalan Aplikasi

Aplikasi yang telah dibangun merupakan platform berbasis web yang dirancang untuk memberikan pengalaman yang intuitif dalam memprediksi risiko stres mahasiswa. Aplikasi ini menyediakan berbagai fitur yang memungkinkan pengguna untuk melakukan prediksi, menganalisis data, dan memahami faktor-faktor yang mempengaruhi kesehatan mental mahasiswa.

4.6.1 Halaman Beranda



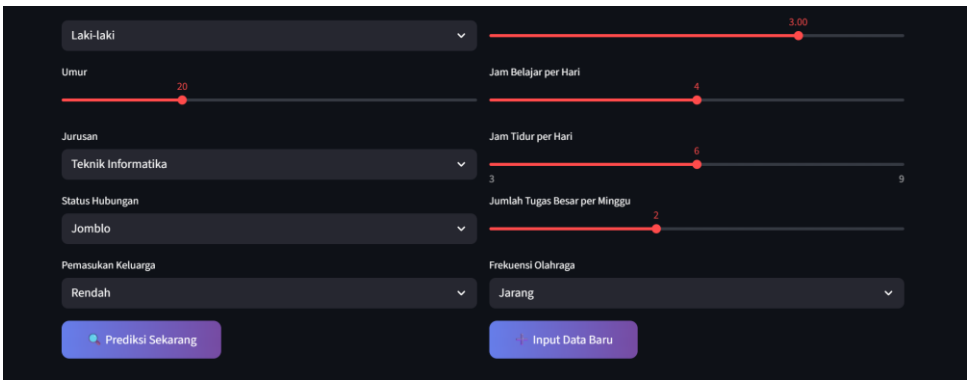
Gambar 4. 22 Tampilan Beranda 1

Gambar 4. 23 Tampilan Beranda 2

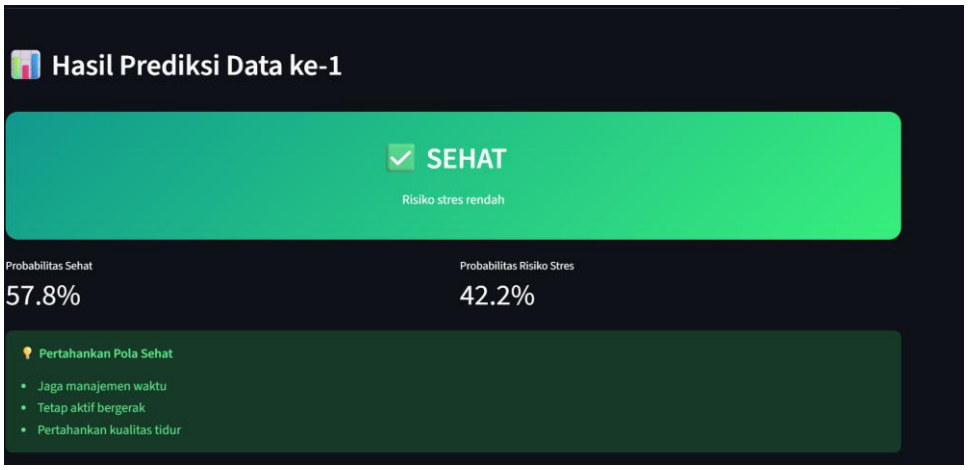
Halaman beranda merupakan tampilan utama aplikasi yang memberikan ringkasan informasi penting tentang dataset dan performa model. Pada bagian atas halaman terdapat header dengan judul aplikasi yang menggunakan gradient styling dan deskripsi singkat tentang tujuan aplikasi. Halaman ini menampilkan empat metrics cards yang menunjukkan total jumlah data dalam dataset, akurasi model yang telah dilatih, jumlah data mahasiswa sehat, dan jumlah data mahasiswa berisiko stres. Di bawah metrics cards, terdapat dua pie chart interaktif yang menampilkan distribusi label (Sehat vs Risiko Stres) dan distribusi gender mahasiswa. Bagian bawah halaman menampilkan tabel yang berisi 10 baris pertama dari dataset untuk memberikan gambaran struktur data kepada pengguna.

4.6.2 Halaman Predikdsi

Gambar 4. 24 Tampilan Prediksii 1



Gambar 4. 25 Tampilan Prediksi 2



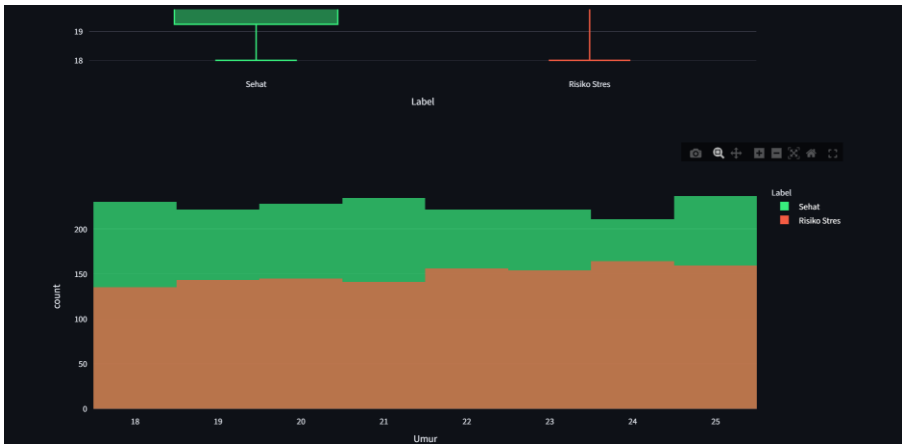
Gambar 4. 26 Tampilan Prediksi 3

Halaman prediksi merupakan fitur utama aplikasi yang memungkinkan pengguna untuk memasukkan data mahasiswa dan mendapatkan prediksi risiko stres secara real-time. Form input dibagi menjadi dua kolom untuk memudahkan pengisian data. Kolom pertama berisi data demografi (gender, umur, jurusan, status hubungan) dan status ekonomi (pemasukan keluarga), sedangkan kolom kedua berisi data akademik & kebiasaan (IPK, jam belajar per hari, jam tidur per hari, jumlah tugas besar per minggu) dan aktivitas fisik (frekuensi olahraga). Setelah pengguna mengklik tombol “Prediksi Risiko Stres", aplikasi menampilkan hasil prediksi dalam box berwarna hijau untuk hasil "Sehat" atau box

berwarna merah untuk hasil "Risiko Stres". Hasil prediksi dilengkapi dengan dua metric cards yang menampilkan probabilitas untuk masing-masing kelas. Di bagian bawah, aplikasi memberikan rekomendasi yang berbeda berdasarkan hasil prediksi, seperti saran untuk tidur cukup, olahraga teratur, dan konsultasi dengan konselor untuk mahasiswa berisiko stres, atau saran untuk mempertahankan pola hidup sehat untuk mahasiswa yang sehat. Di bagian paling bawah halaman tersedia tombol "Input Data Baru" yang memungkinkan pengguna untuk melakukan prediksi dengan data yang berbeda tanpa harus refresh halaman.

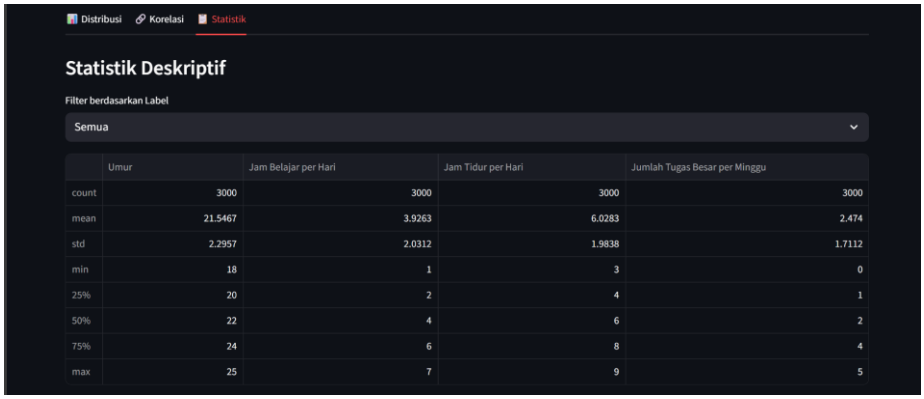
4.6.3 Halaman Analisis Data

Gambar 4. 27 Tampilan Analisis 1



Gambar 4. 28 Tampilan Analisis 2

Gambar 4. 29 Tampilan Analisis 3

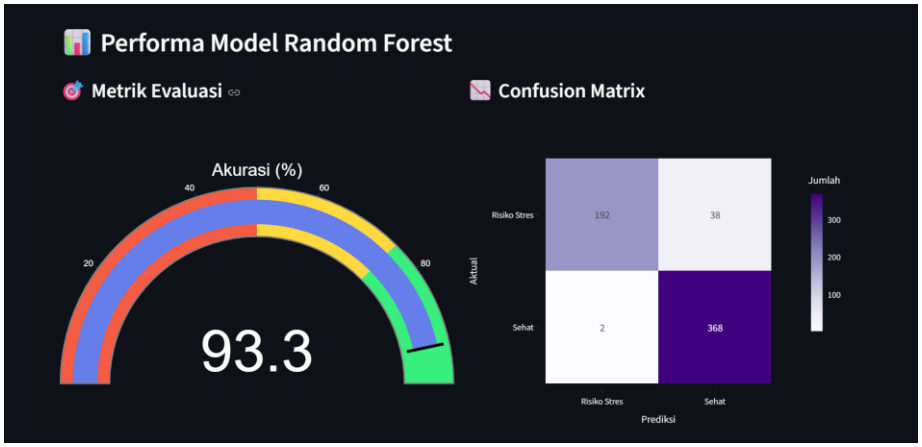


	Umur	Jam Belajar per Hari	Jam Tidur per Hari	Jumlah Tugas Besar per Minggu
count	3000	3000	3000	3000
mean	21.5467	3.9263	6.0283	2.474
std	2.2957	2.0312	1.9838	1.7112
min	18	1	3	0
25%	20	2	4	1
50%	22	4	6	2
75%	24	6	8	4
max	25	7	9	5

Gambar 4. 30 Tampilan Analisis 4

Halaman analisis data menyediakan berbagai visualisasi dan analisis eksplorasi untuk memahami pola dan karakteristik dataset. Halaman ini menggunakan tab-based navigation yang terdiri dari tiga tab utama. Tab "Distribusi" memungkinkan pengguna memilih fitur numerik (umur, jam belajar per hari, jam tidur per hari, jumlah tugas besar per minggu) untuk dianalisis, kemudian menampilkan box plot untuk melihat distribusi data berdasarkan label dan histogram overlay untuk membandingkan distribusi kedua kelas. Tab "Korelasi" menampilkan grouped bar chart yang menganalisis hubungan antara variabel kategorikal (jurusan, frekuensi olahraga, pemasukan keluarga, status hubungan, gender) dengan label, membantu mengidentifikasi pola berdasarkan kategori tertentu. Tab "Statistik" menampilkan tabel statistik deskriptif yang mencakup mean, standar deviasi, minimum, maksimum, dan quartiles dari fitur-fitur numerik, dengan fitur filter berdasarkan label (Semua, Sehat, atau Risiko Stres) untuk analisis yang lebih spesifik.

4.6.4 Halaman Performa Model

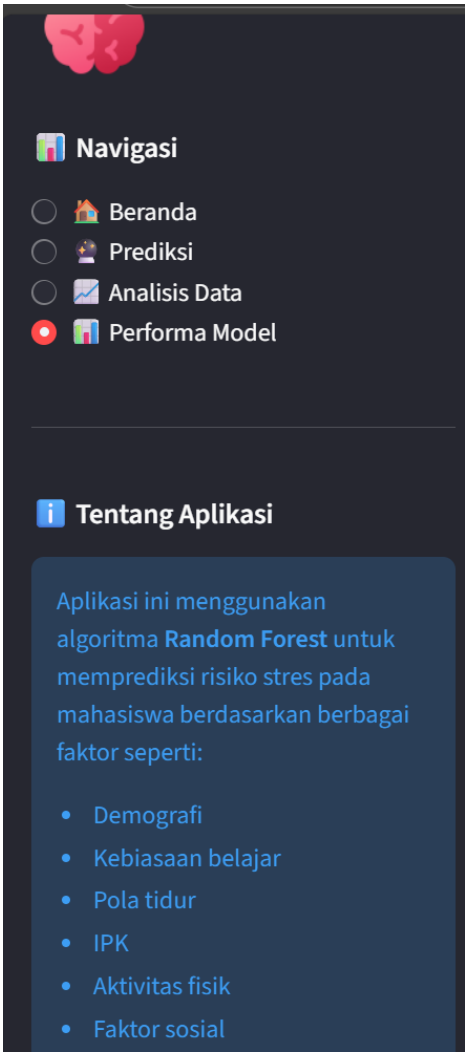


Gambar 4. 31 Tampilan Model 1

Gambar 4. 32 Tampilan Model 2

Halaman performa model memberikan informasi detail tentang performa model Random Forest yang telah dibangun dan dievaluasi. Halaman ini terbagi menjadi dua kolom utama. Kolom pertama menampilkan gauge chart berbentuk speedometer yang memvisualisasikan akurasi model dengan indikator warna merah (0-50%) untuk performa buruk, kuning (50-75%) untuk performa sedang, dan hijau (75-100%) untuk performa baik, serta menampilkan metric F1-Score weighted di bawahnya. Kolom kedua menampilkan confusion matrix dalam bentuk heatmap dengan gradasi warna yang menunjukkan jumlah True Positive, True Negative, False Positive, dan False Negative untuk memudahkan interpretasi hasil klasifikasi. Di bagian bawah halaman, terdapat dua kolom yang menampilkan informasi parameter model (algoritma Random Forest, jumlah trees 200, max depth 4, random state 42, test size 20%) dan daftar fitur yang digunakan yang terbagi menjadi fitur numerik (umur, jam belajar, jam tidur, IPK, jumlah tugas) dan fitur kategorikal (gender, jurusan, frekuensi olahraga, pemasukan keluarga, status hubungan).

4.6.5 Sidebar



Gambar 4. 33 Tampilan Sidebar

Sidebar merupakan komponen navigasi yang konsisten di setiap halaman aplikasi untuk memudahkan pengguna berpindah antar halaman. Di bagian atas sidebar terdapat icon brain yang merepresentasikan tema kesehatan mental. Di bawahnya terdapat menu navigasi berbentuk radio button yang terdiri dari empat pilihan halaman: Beranda, Prediksi, Analisis Data, dan Performa Model. Setelah menu navigasi, terdapat garis pemisah dan info box yang berisi informasi tentang aplikasi, menjelaskan bahwa aplikasi ini menggunakan algoritma Random Forest untuk memprediksi risiko stres mahasiswa berdasarkan berbagai faktor seperti demografi, kebiasaan belajar, pola tidur, IPK, aktivitas fisik, dan faktor sosial. Sidebar ini memastikan pengguna dapat dengan mudah mengakses seluruh fitur aplikasi tanpa harus kembali ke halaman utama.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil perancangan dan implementasi Aplikasi Klasifikasi Kesehatan Mental Mahasiswa, dapat disimpulkan bahwa algoritma Random Forest berhasil diterapkan dengan baik dalam

mengklasifikasikan kondisi kesehatan mental mahasiswa berdasarkan karakteristik yang dimiliki. Proses pengembangan aplikasi ini dilakukan melalui beberapa tahapan utama, yaitu pengumpulan data, preprocessing data, feature engineering, pembuatan model, evaluasi model, hingga deployment aplikasi.

Dataset yang digunakan terdiri dari 3000 data mahasiswa dengan berbagai atribut, antara lain gender, usia, jurusan, jam belajar, jam tidur, IPK, jumlah tugas, frekuensi olahraga, pemasukan keluarga, dan status hubungan. Tahap preprocessing dilakukan untuk memastikan kualitas data, seperti menghapus data yang memiliki nilai kosong *(missing values)* serta data yang tidak valid. Selanjutnya, fitur numerik dinormalisasi menggunakan Z-score normalization untuk menyeragamkan skala data, sedangkan variabel kategorikal diubah menjadi bentuk numerik menggunakan teknik One-Hot Encoding dengan parameter *handle_unknown='ignore'*

Model klasifikasi dibangun menggunakan algoritma Random Forest Classifier dengan parameter yang telah ditentukan, yaitu *n_estimators*=200 dan *max_depth*=4, serta *random_state*=42 untuk memastikan reproducibility hasil. Evaluasi model dilakukan menggunakan metrik accuracy, classification report (*precision, recall, dan F1-score*), serta confusion matrix, yang menunjukkan bahwa model mampu melakukan klasifikasi dengan performa yang cukup baik. Selain itu, analisis feature importance memberikan gambaran mengenai faktor-faktor yang paling berpengaruh terhadap kondisi kesehatan mental mahasiswa.

Hasil akhir dari aplikasi ini berupa klasifikasi kondisi kesehatan mental mahasiswa ke dalam dua kategori, yaitu “Sehat” dan “Risiko Stres”. Aplikasi kemudian dideploy menggunakan Streamlit sehingga dapat diakses melalui antarmuka web yang interaktif dan mudah digunakan. Dengan demikian, aplikasi ini diharapkan dapat berfungsi sebagai early warning system bagi mahasiswa untuk mengenali potensi risiko stres sejak dini, serta membantu pihak institusi pendidikan dalam memahami faktor-faktor yang memengaruhi kesehatan mental mahasiswa.

5.2 Saran

Berdasarkan hasil penelitian dan pengembangan aplikasi yang telah dilakukan, terdapat beberapa saran yang dapat dipertimbangkan untuk pengembangan lebih lanjut, antara lain sebagai berikut :

5.2.1 Perluasan Dataset

Untuk meningkatkan akurasi dan representativitas hasil prediksi, disarankan untuk memperluas dataset dengan mengumpulkan data dari berbagai universitas dan institusi pendidikan di Indonesia. Selain itu, penambahan variabel baru yang relevan seperti tingkat keaktifan organisasi, durasi penggunaan gadget, konsumsi kafein, serta riwayat kesehatan mental keluarga dapat memberikan informasi yang lebih komprehensif. Pembaruan dataset secara berkala juga penting untuk menyesuaikan dengan perubahan karakteristik mahasiswa dari waktu ke waktu.

5.2.2 Peningkatan Performa Model

Pengembangan model dapat ditingkatkan dengan melakukan hyperparameter tuning menggunakan metode Grid Search atau Random Search untuk memperoleh kombinasi parameter yang optimal. Selain itu, penggunaan algoritma lain seperti XGBoost, Gradient Boosting, atau Neural Network dapat dijadikan sebagai pembanding performa model. Penerapan teknik cross-validation serta penanganan ketidakseimbangan data menggunakan metode SMOTE juga disarankan untuk meningkatkan keandalan model.

5.2.3 Validasi dan Kolaborasi

Untuk meningkatkan kredibilitas dan dampak penggunaan aplikasi, disarankan untuk melakukan validasi dengan melibatkan pakar psikologi atau kesehatan mental. Kolaborasi dengan unit layanan konseling kampus serta pelaksanaan uji coba pengguna (user testing) dapat membantu dalam penyempurnaan fungsi dan tampilan aplikasi. Selain itu, aspek etika dan privasi data pengguna harus tetap diperhatikan dan disesuaikan dengan regulasi yang berlaku.

DAFTAR PUSTAKA

[1] A. T. Setyanto, "Deteksi Dini Prevalensi Gangguan Kesehatan Mental Mahasiswa di Perguruan Tinggi," *Wacana*, vol. 15, no. 1, p. 66, Feb. 2023, doi: 10.20961/wacana.v15i1.69548.

[2] A. Ananda Hapsari, A. Syafei Nursuwanda, H. Zuhriyah, and D. Junesco Vresdian, "Klasifikasi Kesehatan Mental Mahasiswa Model TMAS dengan Algoritma Decision Tree, Logistic Regression, dan Random Forest," vol. 7, 2024.

[3] M. E. Hasan *et al.*, "Prevalence, associated factors, and machine learning-based prediction of depression, anxiety, and stress among university students: a cross-sectional study from Bangladesh," *J. Health Popul. Nutr.*, vol. 44, no. 1, p. 361, Dec. 2025, doi: 10.1186/s41043-025-01095-8.

[4] I. Setiawan, I. Fatah Yasin, Y. Tri Desianti, P. Studi Sistem Dan Teknologi Informasi, F. Sains Dan Teknologi, and A. Surakarta, "Komparasi Kinerja Algoritma Random Forest, Decision Tree, Naïve Bayes, dan KNN dalam Prediksi Tingkat Depresi Mahasiswa Menggunakan Student Depression Dataset," 2025. [Online]. Available: http://creativecommons.org/licences/by/4.0/

[5] F. S. Jumeilah, "Penerapan Support Vector Machine (SVM) untuk Pengkategorian Penelitian," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 1, no. 1, pp. 19–25, 2017, doi: 10.29207/resti.v1i1.11.

[6] A. Shiri, "Introduction to Modern Information Retrieval (2nd edition)," *Library Review*, vol. 53, no. 9, pp. 462–463, 2004, doi: 10.1108/00242530410565256.

[7] Rifqi Mulyawan, "Text Preprocessing: Pengertian, Apa itu NLTK Library? Macam Tahapan Basic (Dasar) serta Contoh Simple dan Kodenya!," Rifqi Mulyawan. [Online]. Available: https://rifqimulyawan.com/blog/text-preprocessing/

[8] R. Tineges, "Tahapan Text Preprocessing dalam Teknik Pengolahan Data," dqlab. [Online]. Available: https://dqlab.id/tahapan-text-preprocessing-dalam-teknik-pengolahan-data

[9] dicoding, "Python: Pengertian, Contoh Penggunaan, dan Manfaat Mempelajarinya," dicoding. [Online]. Available: https://www.dicoding.com/blog/python-pengertian-contoh-penggunaan-dan-manfaat-mempelajarinya/

[10] N. Huda, "Visual Studio Code: Pengertian, Fitur, Keunggulan dan Jenisnya," dewaweb. [Online]. Available: https://blog.dewaweb.com/mengenal-visual-studio-code/