Dokumentasi Teknis: MindMetric - Analisis Tingkat Stres (Versi 2.0)

Sumber Dataset:

https://www.kaggle.com/datasets/uom190346a/sleep-health-and-lifestyle-dataset

1. Pendahuluan

Proyek ini bertujuan membangun model *Machine Learning* menggunakan algoritma **K-Nearest Neighbors (KNN)** untuk memprediksi tingkat stres seseorang berdasarkan parameter kesehatan (tidur, langkah kaki, BMI, dll).

Pembaruan Utama di Versi ini:

1. **Pembersihan Data BMI:** Menggabungkan kategori "Normal Weight" dan "Normal" agar data lebih konsisten.
2. **Input Interaktif:** Menggunakan menu *dropdown* teks untuk uji coba manual, menggantikan input kode angka yang membingungkan.

---

2. Bedah Kode Per Bagian

Langkah 1: Import Library & Load Data

Menyiapkan "alat-alat" (library) dan memuat bahan baku (dataset).

Kode:

Python
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

filename = 'Sleep_health_and_lifestyle_dataset.csv'
df = pd.read_csv(filename)

print("Langkah 1 Berhasil: Dataset sudah dimuat.")
display(df.head())
```

Penjelasan Baris per Baris:

- `import pandas as pd`: Memanggil library Pandas (sebagai `pd`) untuk mengolah data tabel.
- `import numpy as np`: Memanggil library NumPy untuk operasi matematika.
- `import seaborn` & `matplotlib`: Library untuk membuat grafik visualisasi.
- `train_test_split`: Fungsi untuk membagi data menjadi soal latihan dan soal ujian.
- `StandardScaler`: Alat untuk menyamakan skala angka (misal: menyetarakan satuan "Langkah" ribuan dengan "Jam Tidur" satuan).
- `KNeighborsClassifier`: Algoritma otak buatan (AI) yang kita gunakan.
- `df = pd.read_csv(...)`: Membaca file Excel/CSV dan memasukkannya ke variabel `df`.
- `display(df.head())`: Menampilkan 5 baris teratas data agar kita bisa melihat isinya.

Langkah 2: Pembersihan dan Preprocessing (Updated)

Membersihkan data kotor dan mengubah huruf menjadi angka agar bisa dihitung matematika.

Kode:

Python
```
# 1. Mengisi nilai kosong (NaN)
df['Sleep Disorder'] = df['Sleep Disorder'].fillna('None')

# 2. Perbaikan Data BMI (Merge kategori)
df['BMI Category'] = df['BMI Category'].replace('Normal Weight', 'Normal')

# 3. Menghapus kolom tidak terpakai
df_bersih = df.drop(['Person ID', 'Blood Pressure'], axis=1)

# 4. Encoding (Huruf ke Angka)
le = LabelEncoder()
kolom_teks = ['Gender', 'Occupation', 'BMI Category', 'Sleep Disorder']

print("\n=== Proses Mengubah Huruf ke Angka ===")
for col in kolom_teks:
  df_bersih[col] = le.fit_transform(df_bersih[col])

  # Print kamus pemetaan untuk referensi
  mapping = dict(zip(le.classes_, range(len(le.classes_))))
  print(f"Kolom '{col}': {mapping}")

print("\n Langkah 2 Berhasil: Data sudah bersih dan berupa angka semua.")
display(df_bersih.head())
```

Penjelasan Baris per Baris:

- `.fillna('None')`: Kolom gangguan tidur yang kosong dianggap tidak punya gangguan ("None").
- `.replace('Normal Weight', 'Normal')`: **(Penting)** Menggabungkan data. Jika ada data tertulis "Normal Weight", diganti jadi "Normal". Ini mencegah AI bingung karena ada dua istilah untuk hal yang sama.
- `.drop(...)`: Membuang kolom `Person ID` (hanya nomor urut) dan `Blood Pressure` (format teks "120/80" sulit dibaca langsung).
- `for col in kolom_teks`: Melakukan perulangan untuk setiap kolom kategori.
    - `le.fit_transform(...)`: Mengubah "Male" jadi 1, "Female" jadi 0, dst.
    - `dict(zip(...))`: Membuat "kamus" contekan agar kita tahu angka berapa mewakili teks apa (misal: 0=Accountant).

Langkah 3: Pembagian Data & Normalisasi

Membagi data untuk ujian dan menyamakan satuan ukuran.

Kode:

Python
```
target = 'Stress Level'
X = df_bersih.drop(target, axis=1)
y = df_bersih[target]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=100,
random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

print(f"Jumlah Data Latih: {X_train.shape[0]} baris")
print(f"Jumlah Data Uji : {X_test.shape[0]} baris")
```

Penjelasan Baris per Baris:

- `X`: Data Fitur (Penyebab). Semua kolom kecuali Stress Level.
- `y`: Data Target (Akibat). Hanya kolom Stress Level.
- `train_test_split(..., test_size=100)`: Mengambil 100 data secara acak untuk disimpan sebagai soal ujian (`X_test`), sisanya 274 data untuk latihan (`X_train`).
- `scaler.fit_transform(X_train)`: Menghitung rata-rata dari data latih, lalu mengubah datanya ke skala standar (Z-score).
- `scaler.transform(X_test)`: Mengubah data uji menggunakan rata-rata dari data latih (agar adil/konsisten).

Langkah 4: Pelatihan Model

Proses komputer belajar mengenali pola stres.

Kode:

Python
```
k_value = 5
metrik = 'euclidean'

print(f"Sedang melatih AI dengan K={k_value}...")
knn = KNeighborsClassifier(n_neighbors=k_value, metric=metrik)
knn.fit(X_train, y_train)
```

Penjelasan Baris per Baris:

- `n_neighbors=5`: Jika ada data baru, AI akan melihat 5 tetangga terdekatnya untuk menyimpulkan.
- `knn.fit(...)`: Perintah untuk "Belajar". AI memetakan posisi data latih `X_train` dan menghafal kunci jawabannya `y_train`.

Langkah 5: Evaluasi

Memeriksa seberapa pintar model setelah belajar.

Kode:

Python
```
y_pred = knn.predict(X_test)

akurasi = accuracy_score(y_test, y_pred)
print(f"\n Akurasi Model: {akurasi * 100:.2f}%")

# (Kode visualisasi Confusion Matrix standar...)
print(classification_report(y_test, y_pred))
```

Penjelasan Baris per Baris:

- `knn.predict(X_test)`: AI disuruh mengerjakan soal ujian (100 data baru).
- `accuracy_score`: Membandingkan jawaban AI (`y_pred`) dengan kunci jawaban asli (`y_test`).

Langkah 6: Uji Coba Manual (Interaktif)

Bagian ini memungkinkan pengguna manusia mencoba model dengan mudah.

Kode & Penjelasan:

Bagian A: Input User (Widget)

Python
```
# @title Uji Coba Model dengan Data Baru
Age = 35 # @param {type: "integer"}
Gender_Input = "Male" # @param ["Male", "Female"]
# ... (variable input lainnya dengan @param)
```

- `# @title` & `# @param`: Kode khusus Google Colab untuk membuat formulir visual (kotak isian dan dropdown). Pengguna tidak perlu menyentuh kode, cukup klik opsi di layar.

Bagian B: Kamus Penerjemah

Python
```
gender_map = {"Female": 0, "Male": 1}
occupation_map = { "Accountant": 0, ..., "Software Engineer": 9, "Teacher":
10 }
# ...
```

- Ini adalah "Kamus" manual yang kita buat agar komputer mengerti pilihan dropdown.
- Contoh: Jika user memilih "Software Engineer", Python akan melihat kamus ini dan mengambil angka `9`.
- *Catatan:* Urutan angka di sini **WAJIB** sama dengan hasil `print` di Langkah 2.

Bagian C: Proses Prediksi

Python
```
try:
    # 1. Konversi Teks ke Angka
    Gender = gender_map[Gender_Input]
    # ... (lanjut untuk variabel lain)

    # 2. Buat DataFrame Satu Baris
    input_data = pd.DataFrame({ ... })

    # 3. Filter Kolom
    fitur_cols = ['Gender', 'Age', ... 'Sleep Disorder']
    input_siap = input_data[fitur_cols]

    # 4. Scaling (SANGAT PENTING)
    input_scaled = scaler.transform(input_siap)

    # 5. Prediksi
    prediksi_angka = knn.predict(input_scaled)[0]
```

- `fitur_cols`: Memastikan urutan kolom input user *persis sama* dengan urutan kolom saat AI latihan. Kalau urutan beda, prediksi akan ngawur.
- `scaler.transform`: Mengubah data input user (misal umur 35) menjadi skala standar (-0.5 sekian) menggunakan rumus yang sama dengan Langkah 3. Tanpa ini, AI tidak bisa membaca data.
- `[0]`: Hasil prediksi berupa list `[7]`. Kita ambil isinya saja (angka 7).

Bagian D: Output Manusiawi

Python

```
kategori_stres = { 3: "Sangat Rendah", ..., 8: "Sangat Tinggi" }
deskripsi = kategori_stres.get(prediksi_angka, "...")
print(f"Skor Stres : {prediksi_angka}")
print(f"Kategori   : {deskripsi}")
```

- Mengubah angka hasil prediksi (7) menjadi kalimat ("Tinggi/Bahaya") agar mudah dipahami pengguna awam.