Dokumentasi Code AI Rekomendasi Workout (FitAI)

1. train_model.py:

Script training AI.

Digunakan untuk: preprocessing data, training model, tuning hyperparameter, dan menyimpan model (.pkl)

```
import pandas as pd
import joblib

from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
from scipy.stats import randint

# ================================
# 1. LOAD DATASET
# ================================
df = pd.read_csv("dataset3.csv")

print("Jumlah data:", df.shape)
print("Kolom:", df.columns.tolist())
print("=" * 50)

# ================================
# 2. DEFINISI FITUR & TARGET
# ================================
TARGET_COL = "Workout_Type"
X = df.drop(columns=[TARGET_COL])
y = df[TARGET_COL]
```

```python
# Simpan urutan fitur (penting untuk inference)
FEATURE_COLS = list(X.columns)

# ================================
# 3. IDENTIFIKASI TIPE DATA
# ================================
numeric_features = X.select_dtypes(include=["int64", "float64"]).columns.tolist()
categorical_features = X.select_dtypes(include=["object"]).columns.tolist()

print("Numerical Features:", numeric_features)
print("Categorical Features:", categorical_features)
print("=" * 50)

# ================================
# 4. PREPROCESSING PIPELINE
# ================================
preprocessor = ColumnTransformer(
    transformers=[
        ("num", StandardScaler(), numeric_features),
        ("cat", OneHotEncoder(handle_unknown="ignore"), categorical_features)
    ]
)

# ================================
# 5. MODEL (KNN)
# ================================
knn = KNeighborsClassifier()

pipeline = Pipeline(
    steps=[
        ("preprocessor", preprocessor),
        ("model", knn)
    ]
)

# ================================
```

```python
# 6. HYPERPARAMETER TUNING
# ================================
param_dist = {
    "model__n_neighbors": randint(3, 15),
    "model__weights": ["uniform", "distance"],
    "model__p": [1, 2]  # Manhattan vs Euclidean
}

search = RandomizedSearchCV(
    pipeline,
    param_distributions=param_dist,
    n_iter=30,          # cukup besar untuk hasil bagus
    cv=5,
    n_jobs=-1,
    verbose=1,
    random_state=42
)

# ================================
# 7. SPLIT DATA
# ================================
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    stratify=y,
    random_state=42
)

# ================================
# 8. TRAINING MODEL
# ================================
print("Mulai training model...")
search.fit(X_train, y_train)

best_model = search.best_estimator_
```

```
print("Training selesai!")
print("=" * 50)

# ================================
# 9. EVALUASI MODEL
# ================================
y_pred = best_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print("AKURASI MODEL:", round(accuracy * 100, 2), "%")
print("\nCLASSIFICATION REPORT:\n")
print(classification_report(y_test, y_pred))

# ================================
# 10. SIMPAN MODEL & FITUR
# ================================
joblib.dump(best_model, "workout_model.pkl")
joblib.dump(FEATURE_COLS, "feature_cols.pkl")

print("=" * 50)
print("Model dan feature_cols berhasil disimpan!")
print("File output:")
print("- workout_model.pkl")
print("- feature_cols.pkl")
```

Penjelasan Code:

1. Import Library

```
import pandas as pd
import joblib
```

- pandas: membaca dan memanipulasi dataset dalam bentuk tabel (DataFrame)
- joblib: menyimpan dan memuat model machine learning dalam bentuk file (`.pkl`)

```
from sklearn.model_selection import train_test_split,
RandomizedSearchCV
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler,
OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score,
classification_report
from scipy.stats import randint
```

- Pembagian data (train-test split)
- Preprocessing data
- Pembuatan pipeline model
- Algoritma KNN
- Evaluasi performa model
- Hyperparameter tuning secara acak

2. Lod Dataset

```
df = pd.read_csv("dataset3.csv")
```

- Membaca file dataset `dataset3.csv`
- Menyimpannya ke dalam DataFrame `df`

```
print("Jumlah data:", df.shape)
print("Kolom:", df.columns.tolist())
```

- Menampilkan jumlah baris dan kolom dataset
- Menampilkan daftar nama kolom sebagai pengecekan awal struktur data

3. Definisi Fitur dan Target

```
TARGET_COL = "Workout_Type"
```

```
X = df.drop(columns=[TARGET_COL])
y = df[TARGET_COL]
```

- `Workout_Type` ditetapkan sebagai target (label) yang akan diprediksi
- `X` berisi seluruh fitur input
- `y` berisi kelas output (jenis workout)

```
FEATURE_COLS = list(X.columns)
```

- Menyimpan urutan fitur input
- Penting saat proses inference (prediksi) agar input konsisten dengan model

4. Identifikasi Tipe Data

```
numeric_features = X.select_dtypes(include=["int64",
"float64"]).columns.tolist()
categorical_features =
X.select_dtypes(include=["object"]).columns.tolist()
```

- Memisahkan fitur numerik (usia, berat, durasi, dll)
- Memisahkan fitur kategorikal (contoh: gender)

Pemisahan ini diperlukan agar setiap tipe data diproses dengan metode yang sesuai.

5. Preprocessing Pipeline

```
preprocessor = ColumnTransformer(
    transformers=[
        ("num", StandardScaler(), numeric_features),
        ("cat", OneHotEncoder(handle_unknown="ignore"),
categorical_features)
    ]
)
```

- StandardScaler: menormalkan fitur numerik agar berada pada skala yang sama
- OneHotEncoder: mengubah data kategorikal menjadi format numerik
- `handle_unknown="ignore"` mencegah error jika ada kategori baru saat prediksi

6. Pembuatan Model KNN

```
knn = KNeighborsClassifier()
```

Mendefinisikan algoritma K-Nearest Neighbors sebagai model klasifikasi.

```
pipeline = Pipeline(
    steps=[
        ("preprocessor", preprocessor),
        ("model", knn)
    ]
)
```

Pipeline ini memastikan:

- Preprocessing dan model berjalan dalam satu alur
- Tidak terjadi kebocoran data (data leakage)
- Proses training dan prediksi lebih konsisten

7. Hyperparameter Tuning

```
param_dist = {
    "model__n_neighbors": randint(3, 15),
    "model__weights": ["uniform", "distance"],
    "model__p": [1, 2]
}
```

Parameter yang diuji:

- `n_neighbors`: jumlah tetangga terdekat
- `weights`: bobot uniform atau berdasarkan jarak
- `p`: jenis jarak (Manhattan atau Euclidean)

```
search = RandomizedSearchCV(
    pipeline,
    param_distributions=param_dist,
    n_iter=30,
    cv=5,
    n_jobs=-1,
    verbose=1,
    random_state=42
)
```

- Mencari kombinasi parameter terbaik secara acak
- Menggunakan 5-fold cross validation
- Memanfaatkan semua core CPU untuk efisiensi

8. Pembagian Data

```
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    stratify=y,
    random_state=42
)
```

- 80% data digunakan untuk training
- 20% data digunakan untuk testing
- `stratify=y` menjaga proporsi kelas tetap seimbang

9. Training Model

```
search.fit(X_train, y_train)
```

- Melatih model menggunakan data training
- Sekaligus melakukan hyperparameter tuning

```
best_model = search.best_estimator_
```

Menyimpan model terbaik hasil tuning.

10. Evaluasi Model

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

- Melakukan prediksi pada data test
- Menghitung akurasi model

```
print(classification_report(y_test, y_pred))
```

Menampilkan metrik evaluasi:

- Precision
- Recall
- F1-score
- Support tiap kelas workout

11. Penyimpanan Model

```
joblib.dump(best_model, "workout_model.pkl")
joblib.dump(FEATURE_COLS, "feature_cols.pkl")
```

File yang dihasilkan:

- `workout_model.pkl`: model AI terlatih
- `feature_cols.pkl`: daftar urutan fitur input

File ini digunakan saat aplikasi dijalankan (di Hugging Face / Gradio).

Kesimpulan:

Kode membangun sistem AI yang:

- Mengolah data pengguna
- Melatih model KNN secara optimal
- Melakukan evaluasi berbasis data uji
- Menyimpan model siap pakai untuk aplikasi

2. **.gitattributes**
   Mengatur perilaku Git (penanganan file besar, line ending).
3. **README.md**
   Dokumentasi utama project. Berisi penjelasan singkat aplikasi, cara menjalankan, dan struktur project.
4. **app.py**
   File utama aplikasi (frontend + inference). Menjalankan UI Gradio, menerima input user, memanggil model, dan menampilkan hasil rekomendasi workout.
5. **dataset3.csv**
   Dataset mentah yang berisi data latihan (Age, Gender, Weight, dll). Digunakan saat training model, bukan saat app dijalankan.
6. **feature_cols.pkl**
   Menyimpan urutan dan nama fitur input. Penting agar input user di app sesuai dengan struktur model.
7. **requirements.txt**
   Daftar library Python yang dibutuhkan (pandas, scikit-learn, gradio, dll). Dipakai Hugging Face untuk install dependency otomatis.
8. **workout_model.pkl**
   Model AI yang sudah dilatih. File ini yang dipakai langsung oleh app.py untuk melakukan prediksi.