Nama : Raihana Fawaz

NIM :1103210102

Kelas : TK-45-05

Heart Dataset

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
```

Kode ini mengimpor berbagai library untuk membangun dan melatih model machine learning berbasis PyTorch. torch digunakan untuk neural network dan optimasi, sedangkan pandas dan numpy untuk pengelolaan data. train_test_split membagi dataset, StandardScaler menormalisasi data, dan accuracy_score menghitung akurasi model. DataLoader dan TensorDataset mempermudah pengelolaan data dalam batch selama pelatihan. Kombinasi ini mendukung proses preprocessing, pelatihan, dan evaluasi model.

```
[5]  # Praproses Data: Pisahkan fitur dan target
     X = df.drop('target', axis=1).values
     y = df['target'].values
```

- X = df.drop('target', axis=1).values: Menghapus kolom bernama 'target' dari DataFrame untuk mendapatkan fitur-fitur independen, lalu mengonversinya menjadi array numpy menggunakan .values.
- y = df['target'].values: Mengambil nilai dari kolom 'target' sebagai array numpy, yang akan digunakan sebagai variabel target dalam model machine learning.

```
# Normalisasi data
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Split data menjadi training dan testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
[7] # Mengonversi data ke dalam bentuk tensor
    X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
    y_train_tensor = torch.tensor(y_train, dtype=torch.long)
    X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
    y_test_tensor = torch.tensor(y_test, dtype=torch.long)
```

Kode tersebut mempersiapkan data untuk digunakan dalam model PyTorch melalui tiga langkah utama, yaitu:

1. Data pada variabel X dinormalisasi menggunakan StandardScaler agar setiap fitur memiliki mean 0 dan standar deviasi 1, sehingga meningkatkan stabilitas dan kinerja model.
2. Data dipisahkan menjadi data pelatihan (X_train, y_train) dan data pengujian (X_test, y_test) dengan pembagian 70:30 menggunakan fungsi train_test_split, dimana test_size=0.3 menunjukkan bahwa 30% data digunakan untuk pengujian, dan random_state=42 digunakan untuk memastikan hasil pemisahan konsisten setiap kali dijalankan.
3. Data yang sudah dipisahkan dikonversi menjadi tensor PyTorch menggunakan fungsi torch.tensor, dengan tipe data float32 untuk fitur dan long untuk target, agar dapat diproses oleh model PyTorch.

Langkah selanjutnya adalah membuat model MLP, melatih model, mengevaluasi model, dan menguji parameter sesuai dengan syarat yang diberikan.

```python
# Menampilkan best dan worst hyperparameter berdasarkan akurasi
print("\nBest Hyperparameter Configuration:")
print(f"Hidden Layers: {best_result['Hyperparameters']['Hidden Layers']}")
print(f"Hidden Neurons: {best_result['Hyperparameters']['Hidden Neurons']}")
print(f"Activation Function: {best_result['Hyperparameters']['Activation Function']}")
print(f"Epochs: {best_result['Hyperparameters']['Epochs']}")
print(f"Learning Rate: {best_result['Hyperparameters']['Learning Rate']}")
print(f"Batch Size: {best_result['Hyperparameters']['Batch Size']}")
print(f"Test Accuracy: {best_result['Test Accuracy'] * 100:.2f}%")
```

```
Best Hyperparameter Configuration:
Hidden Layers: 3
Hidden Neurons: 32
Activation Function: Softmax
Epochs: 100
Learning Rate: 0.0001
Batch Size: 256
Test Accuracy: 84.09%
```

Parameter terbaik

```python
# Menampilkan hyperparameter terburuk berdasarkan akurasi terendah
print("\nWorst Hyperparameter Configuration:")
print(f"Hidden Layers: {worst_result['Hyperparameters']['Hidden Layers']}")
print(f"Hidden Neurons: {worst_result['Hyperparameters']['Hidden Neurons']}")
print(f"Activation Function: {worst_result['Hyperparameters']['Activation Function']}")
print(f"Epochs: {worst_result['Hyperparameters']['Epochs']}")
print(f"Learning Rate: {worst_result['Hyperparameters']['Learning Rate']}")
print(f"Batch Size: {worst_result['Hyperparameters']['Batch Size']}")
print(f"Test Accuracy: {worst_result['Test Accuracy'] * 100:.2f}%")
```

```
Worst Hyperparameter Configuration:
Hidden Layers: 2
Hidden Neurons: 8
Activation Function: linear
Epochs: 10
Learning Rate: 0.0001
Batch Size: 256
Test Accuracy: 32.14%
```

Parameter terburuk