Nama : Raihana Fawaz

Kelas : TK-45-05

NIM : 1103210102

Dummy Data

```
[18] 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.datasets import make_classification
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import StandardScaler
     from sklearn.metrics import accuracy_score
```

Import library yang akan digunakan

```
[19] # Membuat Data Dummy untuk Klasifikasi data
     X, y = make_classification(n_samples=1000, n_features=20, n_classes=2, random_state=42)

     # Praproses Data
     scaler = StandardScaler()
     X = scaler.fit_transform(X)  # Normalisasi features

     # Membagi Data Menjadi Set Pelatihan dan Pengujian
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

     # Mengonversi Data ke Tensors PyTorch
     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 ini bertujuan untuk mempersiapkan data dummy untuk klasifikasi dengan PyTorch. Pertama, data dummy dibuat menggunakan fungsi make_classification dengan 1000 sampel, 20 fitur, dan 2 kelas. Kemudian, data X dinormalisasi menggunakan StandardScaler untuk memastikan setiap

fitur memiliki mean 0 dan standar deviasi 1, yang penting untuk stabilitas pelatihan model. Data selanjutnya dibagi menjadi data pelatihan (X_train, y_train) dan pengujian (X_test, y_test) dengan proporsi 70:30 menggunakan fungsi train_test_split. Akhirnya, data pelatihan dan pengujian dikonversi ke dalam bentuk tensor PyTorch dengan tipe data float32 untuk fitur dan long untuk label, sehingga dapat digunakan untuk melatih model neural network.

```
[21] # Menentukan model MLP
     def create_mlp(input_size, hidden_layers, hidden_neurons, activation_function):
         layers = []
         # Lapisan Input ke Lapisan Tersembunyi Pertama
         layers.append(nn.Linear(input_size, hidden_neurons))
         # Menambahkan Lapisan Tersembunyi
         for _ in range(hidden_layers - 1):
             layers.append(nn.Linear(hidden_neurons, hidden_neurons))

         # Memilih Fungsi Aktivasi
         if activation_function == 'linear':
             activation = nn.Identity()
         elif activation_function == 'Sigmoid':
             activation = nn.Sigmoid()
         elif activation_function == 'ReLU':
             activation = nn.ReLU()
         elif activation_function == 'Softmax':
             activation = nn.Softmax(dim=1)
         elif activation_function == 'Tanh':
             activation = nn.Tanh()

         # Lapisan Output
         layers.append(nn.Linear(hidden_neurons, 2))  # Menghasilkan 2 Kelas (0 atau 1)

         # Menggabungkan Semua Lapisan
         model = nn.Sequential(*layers)
         return model
```

Kode ini mendefinisikan fungsi create_mlp untuk membangun model Multi-Layer Perceptron (MLP) dengan PyTorch. Fungsi ini menerima parameter input_size, hidden_layers, hidden_neurons, dan activation_function. Pertama, lapisan input dan lapisan tersembunyi ditambahkan sesuai dengan jumlah layer dan neuron yang ditentukan. Fungsi aktivasi dipilih berdasarkan parameter activation_function (seperti linear, Sigmoid, ReLU, Softmax, atau Tanh). Lapisan output ditambahkan pada akhir dengan dua neuron untuk klasifikasi biner. Semua lapisan digabungkan menggunakan nn.Sequential, dan model dikembalikan untuk digunakan dalam pelatihan.

Setelah mengevaluasi model dan menjalankan hyperparameter. Maka didapat:

1. Konfigurasi terbaik untuk linear

```
Best Configuration for Activation Function linear:
Hidden Layers: 2
Hidden Neurons: 64
Activation Function: linear
Epochs: 250
Learning Rate: 0.1
Batch Size: 256
Test Accuracy: 87.33%
```

2. Konfigurasi terbaik untuk sigmoid

```
Best Configuration for Activation Function Sigmoid:
Hidden Layers: 3
Hidden Neurons: 8
Activation Function: Sigmoid
Epochs: 50
Learning Rate: 0.01
Batch Size: 128
Test Accuracy: 87.00%
```

3. Konfigurasi terbaik untuk RelU

```
Best Configuration for Activation Function ReLU:
Hidden Layers: 1
Hidden Neurons: 8
Activation Function: ReLU
Epochs: 25
Learning Rate: 0.1
Batch Size: 512
Test Accuracy: 87.00%
```

4. Konfigurasi terbaik untuk softmax

```
Best Configuration for Activation Function Softmax:
Hidden Layers: 2
Hidden Neurons: 16
Activation Function: Softmax
Epochs: 100
Learning Rate: 0.1
Batch Size: 128
Test Accuracy: 87.67%
```

5. Konfigurasi terbaik untuk tanh

```
Best Configuration for Activation Function Tanh:
Hidden Layers: 2
Hidden Neurons: 16
Activation Function: Tanh
Epochs: 25
Learning Rate: 0.01
Batch Size: 64
Test Accuracy: 87.33%
```

6. Hyperparameter terbaik

```
[33] # Menampilkan Konfigurasi Hyperparameter Terbaik dan Terburuk
     print("\nBest Hyperparameter Configuration (Overall):")
     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 (Overall):
Hidden Layers: 2
Hidden Neurons: 16
Activation Function: Softmax
Epochs: 100
Learning Rate: 0.1
Batch Size: 128
Test Accuracy: 87.67%
```

7. Hyperparameter terburuk

```
     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']['Bat                    ")
     print(f"Test Accuracy: {worst_result['Test Accuracy'] * 100:.2f}%")
```

Loading...

```
Worst Hyperparameter Configuration:
Hidden Layers: 1
Hidden Neurons: 8
Activation Function: ReLU
Epochs: 25
Learning Rate: 0.0001
Batch Size: 512
Test Accuracy: 30.00%
```