Nama : Khalishah

NIM : 1103213045

Dummy Data Week 11

```
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
```

Dalam kode ini, berbagai pustaka diimpor untuk membangun dan melatih model klasifikasi. Pustaka seperti torch digunakan untuk membangun model neural network, sementara sklearn digunakan untuk pembuatan data dummy, pra-pemrosesan, dan evaluasi. Pustaka lain seperti pandas dan numpy digunakan untuk manipulasi data.

```
# Membuat data dummy untuk klasifikasi
X, y = make_classification(n_samples=1000, n_features=20, n_classes=2, random_state=42)

# Praproses data dengan normalisasi fitur
scaler = StandardScaler()
X = scaler.fit_transform(X)  # Normalisasi data fitur

# 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 menjadi tensor 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)
```

Data dummy untuk klasifikasi dibuat menggunakan make_classification dari sklearn, menghasilkan dataset dengan 1000 sampel dan 20 fitur. Setelah itu, data dinormalisasi menggunakan StandardScaler, kemudian dibagi menjadi data latih dan uji dengan train_test_split. Akhirnya, data diubah menjadi tensor PyTorch agar kompatibel dengan model neural network.

```python
# Menentukan model MLP
def create_mlp(input_size, hidden_layers, hidden_neurons, activation_function):
    layers = []
    # Menambahkan lapisan input ke lapisan tersembunyi pertama
    layers.append(nn.Linear(input_size, hidden_neurons))

    # Menambahkan lapisan tersembunyi tambahan
    for _ in range(hidden_layers - 1):
        layers.append(nn.Linear(hidden_neurons, hidden_neurons))

    # Menentukan 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()

    # Menambahkan fungsi aktivasi ke setiap lapisan tersembunyi
    layers = [nn.Sequential(layer, activation) for layer in layers]

    # Menambahkan lapisan output
    layers.append(nn.Linear(hidden_neurons, 2))  # Menghasilkan 2 kelas (0 atau 1)

    # Membuat model sebagai Sequential dari semua lapisan
    model = nn.Sequential(*layers)
    return model
```

Fungsi create_mlp mendefinisikan model Multi-Layer Perceptron (MLP). Fungsi ini memungkinkan pengguna untuk menyesuaikan jumlah lapisan tersembunyi, jumlah neuron per lapisan, dan fungsi aktivasi seperti Linear, Sigmoid, ReLU, Softmax, atau Tanh. Model dibangun secara modular dengan PyTorch, menggunakan kelas nn.Sequential.

```
Best Configuration for Activation Function linear:
Hidden Layers: 2
Hidden Neurons: 32
Activation Function: linear
Epochs: 50
Learning Rate: 0.1
Batch Size: 512
Test Accuracy: 87.00%

Best Configuration for Activation Function Sigmoid:
Hidden Layers: 3
Hidden Neurons: 4
Activation Function: Sigmoid
Epochs: 100
Learning Rate: 0.001
Batch Size: 64
Test Accuracy: 87.00%

Best Configuration for Activation Function ReLU:
Hidden Layers: 1
Hidden Neurons: 8
Activation Function: ReLU
Epochs: 100
Learning Rate: 0.001
Batch Size: 128
Test Accuracy: 87.33%

Best Configuration for Activation Function Softmax:
Hidden Layers: 3
Hidden Neurons: 4
Activation Function: Softmax
Epochs: 50
Learning Rate: 0.01
Batch Size: 512
Test Accuracy: 87.00%

Best Configuration for Activation Function Tanh:
Hidden Layers: 3
Hidden Neurons: 4
Activation Function: Tanh
Epochs: 100
Learning Rate: 0.001
Batch Size: 256
Test Accuracy: 86.67%
```

Hasil menunjukkan konfigurasi terbaik dari beberapa fungsi aktivasi (Linear, Sigmoid, ReLU, Softmax, Tanh). Masing-masing konfigurasi memiliki kombinasi tertentu dari hyperparameter seperti jumlah lapisan tersembunyi, neuron, epoch, learning rate, dan batch size. Test accuracy menunjukkan efektivitas model dengan konfigurasi tersebut.

c

```
# 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: 1
Hidden Neurons: 8
Activation Function: ReLU
Epochs: 100
Learning Rate: 0.001
Batch Size: 128
Test Accuracy: 87.33%
```

Menampilkan konfigurasi hyperparameter terbaik berdasarkan test accuracy. Untuk model ini, kombinasi terbaik adalah dengan 1 hidden layer, 8 neuron per layer, fungsi aktivasi ReLU, learning rate 0.001, dan batch size 128, menghasilkan akurasi 87.33%.

```
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: 1
Hidden Neurons: 16
Activation Function: linear
Epochs: 100
Learning Rate: 0.0001
Batch Size: 512
Test Accuracy: 40.00%
```

Konfigurasi terburuk memiliki 1 hidden layer, 16 neuron, fungsi aktivasi Linear, learning rate 0.0001, dan batch size 512. Konfigurasi ini hanya mencapai test accuracy sebesar 40%, menunjukkan bahwa pemilihan hyperparameter sangat memengaruhi performa mode