Laporan MLP Classification

Nama : Anita Firda Nuralifah

NIM : 1103213117

Library

```
import pandas as pd #Untuk manipulasi dan analisis data menggunakan struktur data DataFrame.
import torch #Menyediakan tools untuk membangun dan melatih neural networks.
import torch.nn as nn #Modul PyTorch yang berisi berbagai layer dan fungsi aktivasi untuk membangun neural networks.
import torch.optim as optim #Modul PyTorch yang berisi berbagai algoritma optimasi untuk melatih neural networks.
from sklearn.model_selection import train_test_split #untuk membagi dataset menjadi data training dan data testing.
from sklearn.preprocessing import StandardScaler, LabelEncoder #Berisi tools untuk preprocessing data seperti scaling dan encoding.
from sklearn.metrics import accuracy_score, classification_report #Berisi tools untuk evaluasi model seperti menghitung akurasi dan menampilkan laporan klasifikasi.
import numpy as np #Untuk komputasi numerik menggunakan array multidimensi.
```

- import pandas as pd : untuk manipulasi dan analisis data menggunakan struktur data DataFrame.
- import torch : menyediakan tools untuk membangun dan melatih neural networks.
- import torch.nn as nn : modul PyTorch yang berisi berbagai layer dan fungsi aktivasi untuk membangun neural networks.
- import torch.optim as optim : modul PyTorch yang berisi berbagai algoritma optimasi untuk melatih neural networks.
- from sklearn.model_selection import train_test_split : untuk membagi dataset menjadi data training dan data testing.
- from sklearn.preprocessing import StandardScaler, LabelEncoder : berisi tools untuk preprocessing data seperti scaling dan encoding.
- from sklearn.metrics import accuracy_score, classification_report : berisi tools untuk evaluasi model seperti menghitung akurasi dan menampilkan laporan klasifikasi.
- import numpy as np : untuk komputasi numerik menggunakan array multidimensi.

Load the dataset

```
df = pd.read_csv('/content/ai4i2020.csv')
df.head()
#memuat dataset
```

- **pd.read_csv('/content/ai4i2020.csv')**: Fungsi pd.read_csv() digunakan untuk membaca file CSV yang berada di lokasi /content/ai4i2020.csv. File tersebut kemudian dimuat ke dalam sebuah DataFrame bernama df.
- **df.head()**: Fungsi head() digunakan untuk menampilkan **5 baris pertama** dari DataFrame df. Ini berguna untuk melihat sekilas isi dataset, seperti kolom dan beberapa data awal.

```
df.info()
#untuk menampilkan ringkasan informasi tentang DataFrame Pandas.
```

Menampilkan ringkasan informasi tentang DataFrame Pandas.

```
df.isna().sum()
#untuk menghitung jumlah nilai yang hilang (missing values) di setiap kolom DataFrame Pandas.
```

Menghitung jumlah nilai yang hilang (missing values) di setiap kolom DataFrame Pandas.

```
df.columns
#untuk mendapatkan daftar nama kolom dalam DataFrame Pandas.
```

Menampilkan kolom dalam dataset

```
label_cols = ['Product ID', 'Type', 'Machine failure']
label_encoder = LabelEncoder()

for col in label_cols:
    df[col] = label_encoder.fit_transform(df[col])
#melakukan encoding pada kolom kategorikal menggunakan LabelEncoder
```

Melakukan encoding pada kolom kategorikal menggunakan LabelEncoder

```
X = df.drop(['Machine failure'], axis=1)
y = df['Machine failure']
#menentukan fitur (X) dan target (y)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
#pisahkan data menjadi training dan testing
```

Menentukan fitur (X) dan target (y) dan memisahkan data menjadi training dan testing

```
[10] scaler = StandardScaler()
     X_train_scaled = scaler.fit_transform(X_train)
     X_test_scaled = scaler.transform(X_test)
     #standardisasi fitur numerik
```

Membuat standar scaler

```
X_train_tensor = torch.tensor(X_train_scaled, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train.values, dtype=torch.long)
X_test_tensor = torch.tensor(X_test_scaled, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test.values, dtype=torch.long)
#konversi data ke tensor
```

Grafik menunjukkan hubungan antara berbagai hyperparameter (jumlah lapisan tersembunyi, neuron, fungsi aktivasi, epoch, learning rate, dan ukuran batch) dengan nilai rata-rata **Mean Absolute Error (MAE)**. Semakin kecil MAE, semakin akurat model. Grafik ini menunjukkan bahwa kombinasi hyperparameter tertentu menghasilkan MAE lebih rendah, sehingga lebih optimal dan efektif dalam prediksi.

```python
class MLPClassification(nn.Module):
    def __init__(self, input_size, hidden_layers, neurons, activation):
        super(MLPClassification, self).__init__()
        self.input_size = input_size
        self.hidden_layers = hidden_layers
        self.neurons = neurons
        self.activation = activation

        layers = []
        layers.append(nn.Linear(self.input_size, self.neurons))
        #membuat layer input ke layer tersembunyi


        for _ in range(self.hidden_layers - 1):
            layers.append(self.activation())
            layers.append(nn.Linear(self.neurons, self.neurons))

        layers.append(nn.Linear(self.neurons, len(y.unique())))
        self.model = nn.Sequential(*layers)
        #menambahkan hidden layers


    def forward(self, x):
        return self.model(x)

#menyusun model MLP untuk klasifikasi
```

Grafik ini menunjukkan bagaimana berbagai kombinasi hyperparameter memengaruhi **Mean Squared Error (MSE)** pada model MLP (Multi-Layer Perceptron). MSE mengukur seberapa jauh prediksi model dari nilai sebenarnya, dengan nilai yang lebih kecil menunjukkan kinerja yang lebih baik. Kombinasi hyperparameter, seperti jumlah lapisan tersembunyi, neuron, fungsi aktivasi, epoch, learning rate, dan ukuran batch, dapat menghasilkan MSE lebih rendah. Ini menegaskan bahwa pengaturan hyperparameter yang tepat sangat penting untuk meningkatkan akurasi model.

```python
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
#setup perangkat (GPU jika tersedia)
```

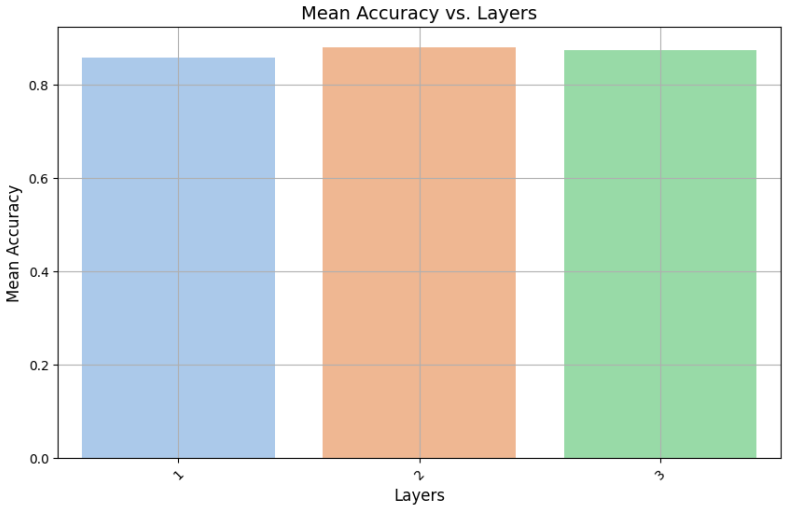
```python
hidden_layers = [1, 2, 3]
neurons = [4, 8, 16, 32, 64]
activations = [nn.Sigmoid, nn.Softmax, nn.ReLU, nn.Tanh]
epochs_list = [1,10,25,50,100,250]
learning_rates = [10, 1, 0.1, 0.01, 0.001, 0.0001]
batch_sizes = [16,32,64,128,256,512]
#hyperparameter yang akan diuji

results = []
#menyimpan hasil
```

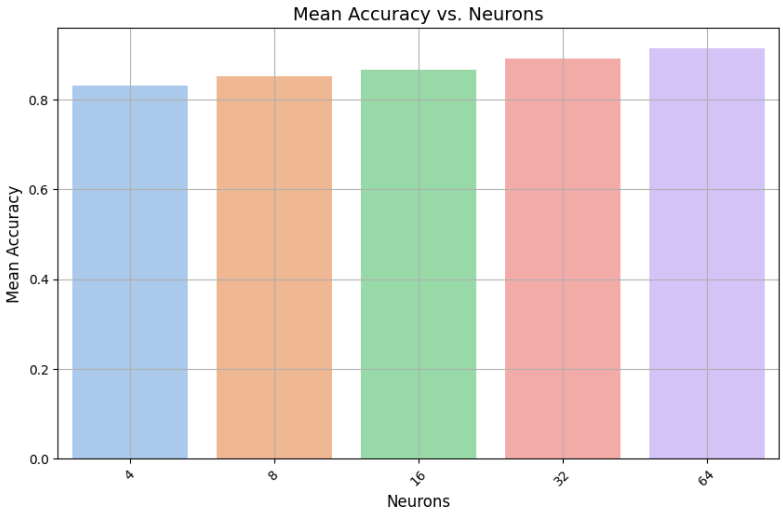
```python
for layers in hidden_layers:
    for neuron in neurons:
        for activation in activations:
            for epochs in epochs_list:
                for lr in learning_rates:
                    for batch_size in batch_sizes:
                        model = MLPClassification(input_size=X_train_tensor.shape[1],
                                                  hidden_layers=layers,
                                                  neurons=neuron,
                                                  activation=activation).to(device)
                        #membuat dan memindahkan model ke perangkat (GPU atau CPU)

                        criterion = nn.CrossEntropyLoss()
                        optimizer = optim.Adam(model.parameters(), lr=lr)
                        #mendefinisikan loss function dan optimizer

                        for epoch in range(epochs):
                            model.train()
                            optimizer.zero_grad()
                            outputs = model(X_train_tensor.to(device))
                            loss = criterion(outputs, y_train_tensor.to(device))
                            loss.backward()
                            optimizer.step()
                        #training loop

                        model.eval()
                        with torch.no_grad():
                            outputs = model(X_test_tensor.to(device))
                            _, predicted = torch.max(outputs, 1)
                            accuracy = accuracy_score(y_test_tensor.cpu(), predicted.cpu())
                        #evaluasi model setelah pelatihan

                        results.append({ # Now using the correct variable name 'results'
                            'layers': layers,
                            'neurons': neuron,
                            'activation': activation.__name__,
                            'epochs': epochs,
                            'lr': lr,
                            'batch_size': batch_size,
                            'accuracy': accuracy
                        })
                        print(f"Layers: {layers}, Neurons: {neuron}, Activation: {activation.__name__}, Epochs: {epochs}, LR: {lr}, Batch Size: {batch_size}, Accuracy:{accuracy}")
                        #menyimpan hasil

#melakukan eksperimen dengan kombinasi hyperparameter
```

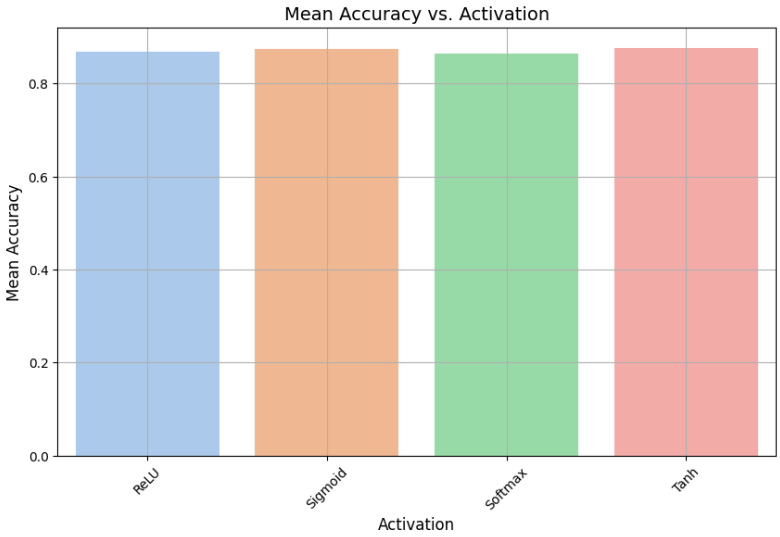
Kode ini melakukan eksperimen dengan berbagai kombinasi hyperparameter pada model **MLP (Multi-Layer Perceptron)** untuk tugas regresi. Hyperparameter yang diuji mencakup jumlah lapisan tersembunyi, neuron, fungsi aktivasi, epoch, learning rate, dan ukuran batch. Setiap kombinasi digunakan untuk melatih model, yang dievaluasi menggunakan metrik seperti **MAE**, **MSE**, **R-squared**, dan akurasi. Hasilnya disimpan untuk dianalisis guna menemukan pengaturan terbaik yang menghasilkan kinerja model optimal.

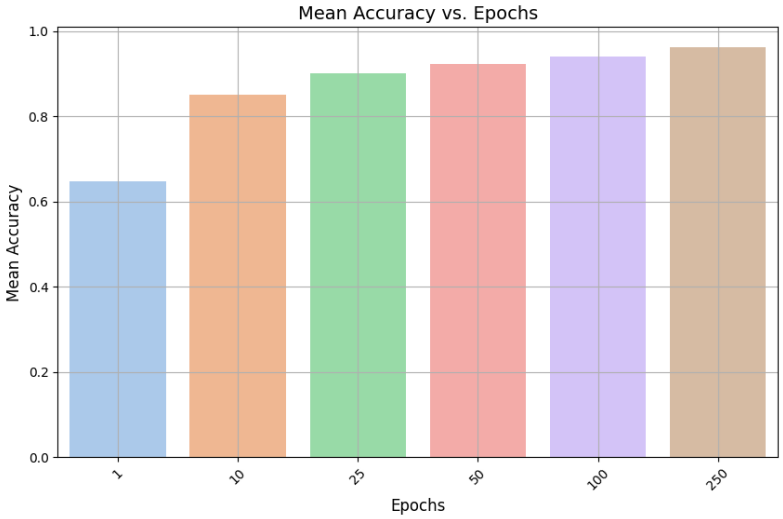
Visualisasi bar plot

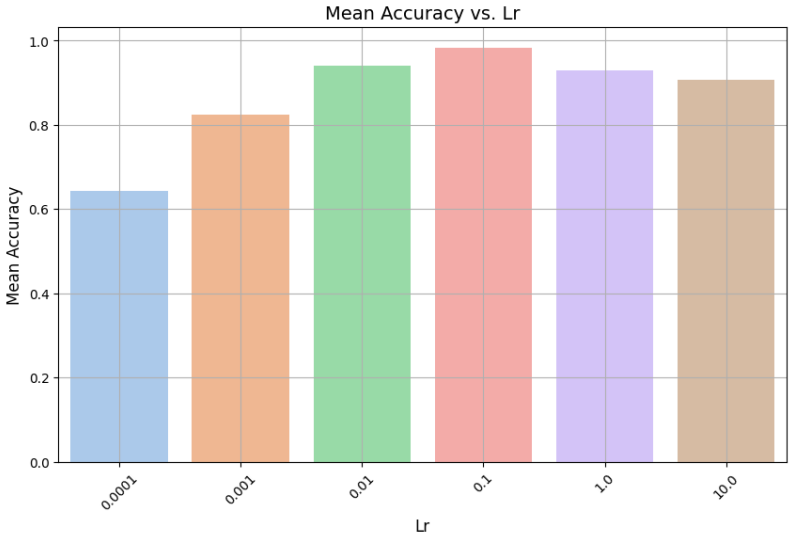
```
results_df = pd.read_csv("mlp_classification_hidden layer 123.csv")
#Mengambil DataFrame hasil eksperimen

mean_accuracy_by_hyperparameter = results_df.groupby(['layers', 'neurons', 'activation', 'epochs', 'lr', 'batch_size'])['accuracy'].mean().reset_index()
#Mengubah 'mae' menjadi 'accuracy' untuk menghitung dan membuat plot akurasi

hyperparameters = ['layers', 'neurons', 'activation', 'epochs', 'lr', 'batch_size']

for param in hyperparameters:
    plt.figure(figsize=(10, 6))
    sns.barplot(data=mean_accuracy_by_hyperparameter, x=param, y='accuracy', ci=None, palette="pastel")
    plt.title(f'Mean Accuracy vs. {param.capitalize()}', fontsize=14)
    plt.xlabel(param.capitalize(), fontsize=12)
    plt.ylabel('Mean Accuracy', fontsize=12)
    plt.xticks(rotation=45)
    plt.grid(True)
    plt.show()
#Membuat visualisasi bar plot
```

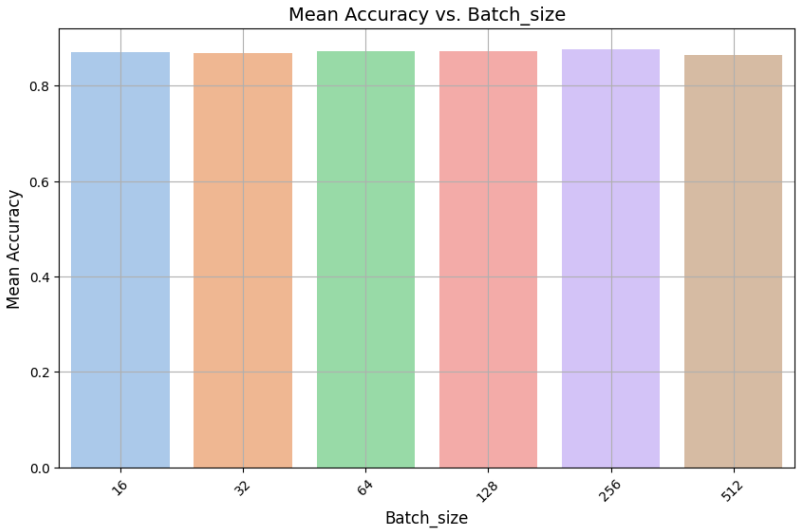





Mean Accuracy vs. Activation
Mean Accuracy
0.0
0.2
0.4
0.6
0.8
ReLU
Sigmoid
Softmax
Tanh
Activation


Mean Accuracy vs. Epochs
Mean Accuracy
0.0
0.2
0.4
0.6
0.8
1.0
1
10
25
50
100
250
Epochs


Mean Accuracy vs. Lr
Mean Accuracy
0.0
0.2
0.4
0.6
0.8
1.0
0.0001
0.001
0.01
0.1
1.0
10.0
Lr



Visualisasi grafik