Laporan MLP Regression

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 mean_absolute_error, mean_squared_error, r2_score #untuk mengevaluasi kinerja model regresi dalam machine learning.
import numpy as np #Untuk komputasi numerik menggunakan array multidimensi
import seaborn as sns #Untuk visualisasi data yang menarik dan informatif.
import matplotlib.pyplot as plt #membuat berbagai jenis visualisasi data seperti grafik garis, diagram batang, scatter plot, dan lainnya.
```

- 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 seaborn as sns : untuk visualisasi data yang menarik dan informatif.
- import matplotlib.pyplot as plt : untuk membuat berbagai jenis visualisasi data seperti grafik garis, diagram batang, scatter plot, dan lainnya.

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.float32)
X_test_tensor = torch.tensor(X_test_scaled, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test.values, dtype=torch.float32)
#konversi data ke tensor
```

Grafik menunjukkan hubungan antara hyperparameter (lapisan tersembunyi, neuron, fungsi aktivasi, epoch, learning rate, dan ukuran batch) dengan **Mean Absolute Error (MAE)**. Semakin kecil nilai MAE, semakin baik kinerja model dalam prediksi. Grafik ini menunjukkan bahwa kombinasi hyperparameter tertentu menghasilkan MAE lebih rendah, menandakan model lebih efektif dibandingkan konfigurasi lainnya.

```python
class MLPRegression(nn.Module):
    def __init__(self, input_size, hidden_layers, neurons, activation):
        super(MLPRegression, 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))
        #Menambahkan hidden layers

        layers.append(nn.Linear(self.neurons, 1))

        self.model = nn.Sequential(*layers)

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

#menyusun model MLP untuk regresi
```

Grafik ini menunjukkan bagaimana kombinasi hyperparameter memengaruhi **Mean Squared Error (MSE)** pada model MLP. MSE mengukur seberapa jauh hasil prediksi dari nilai sebenarnya, di mana nilai yang lebih rendah menandakan kinerja model yang lebih baik. Kombinasi hyperparameter, seperti jumlah lapisan tersembunyi, neuron, fungsi aktivasi, epoch, learning rate, dan ukuran batch, dapat menghasilkan MSE lebih rendah. Ini menunjukkan bahwa pengaturan hyperparameter yang tepat sangat memengaruhi akurasi dan kinerja model.

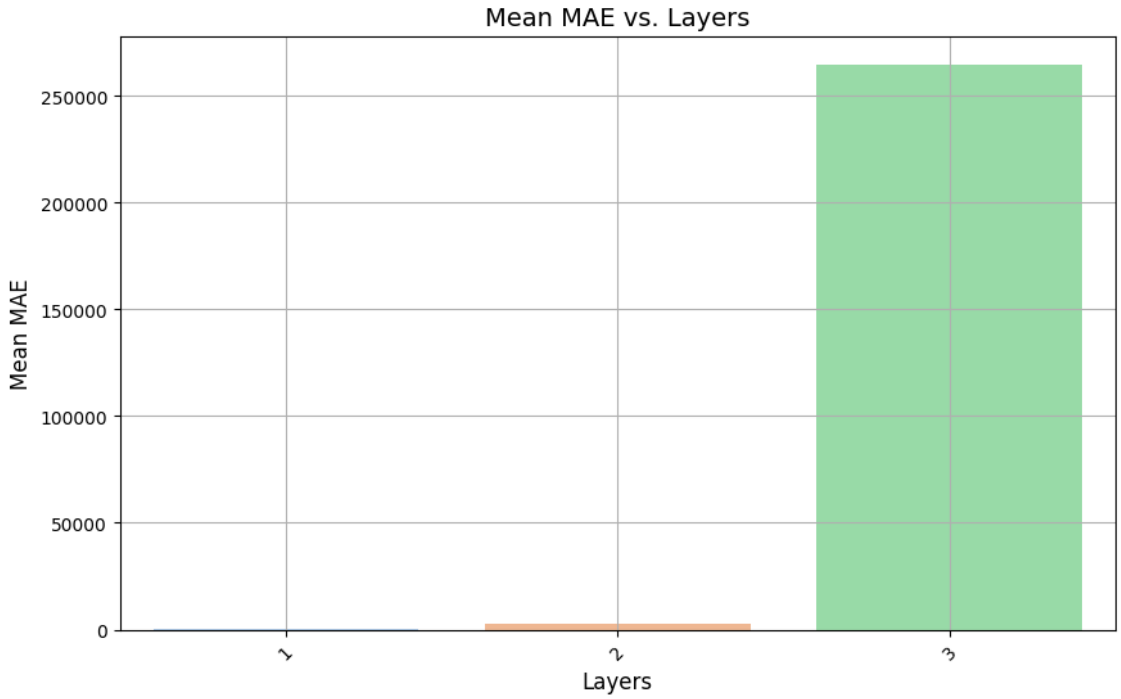
```
[13] device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
     #setup perangkat (GPU jika tersedia)

     hidden_layers = [1, 2, 3]
     neurons = [4, 8, 16, 32, 64]
     activations = [nn.Sigmoid, nn.ReLU, nn.Softmax, 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

     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 = MLPRegression(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)
     #melakukan eksperimen dengan kombinasi hyperparameter

                             criterion = nn.MSELoss()
                             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

                             # Evaluasi model setelah pelatihan
                             model.eval()
                             with torch.no_grad():
                                 y_pred = model(X_test_tensor.to(device)).cpu().numpy()
                                 mae = mean_absolute_error(y_test, y_pred)
                                 mse = mean_squared_error(y_test, y_pred)
                                 r2 = r2_score(y_test, y_pred)
                                 accuracy = (1 - mae / max(y_test)) * 100
                             #evaluasi model setelah pelatihan

                             results.append({
                                 'layers': layers,
                                 'neurons': neuron,
                                 'activation': activation.__name__,
                                 'epochs': epochs,
                                 'lr': lr,
                                 'batch_size': batch_size,
                                 'mae': mae,
                                 'mse': mse,
                                 'r2': r2,
                                 'accuracy': accuracy
                             })
                             print(f"Layers: {layers}, Neurons: {neuron}, Activation: {activation.__name__}, Epochs: {epochs}, LR: {lr}, Batch Size: {batch_size}, Accuracy:{accuracy}")
                             #menyimpan hasil
```

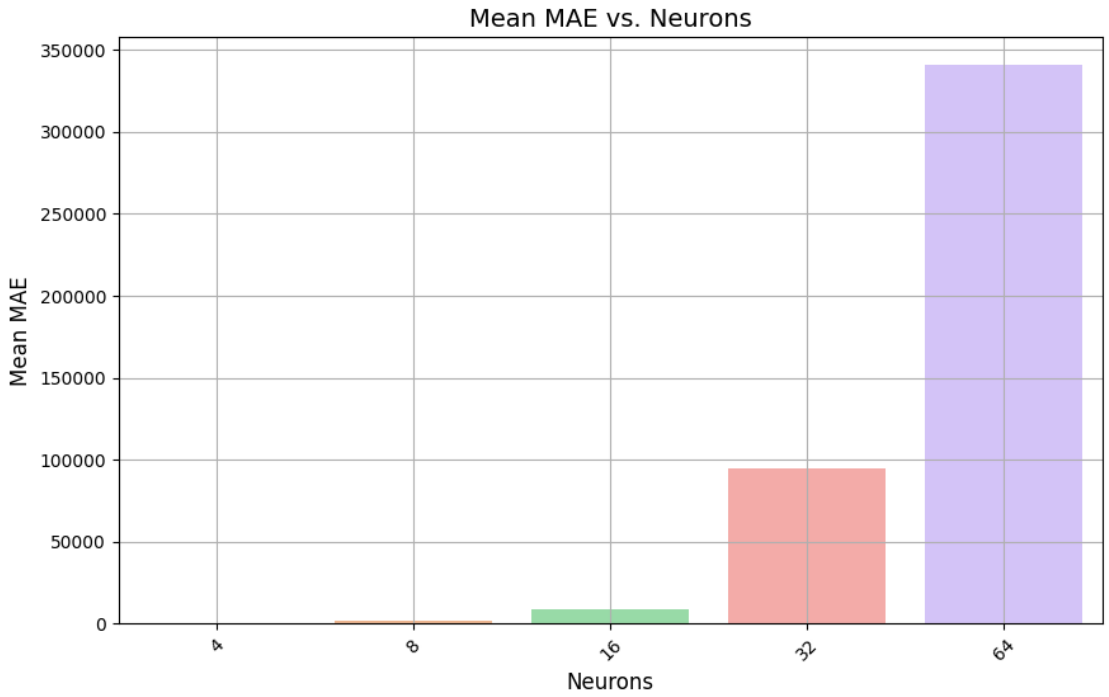
Kode ini bertujuan menguji berbagai kombinasi hyperparameter pada model **MLP (Multi-Layer Perceptron)** untuk tugas regresi. Hyperparameter yang diuji meliputi lapisan tersembunyi, neuron, fungsi aktivasi, epoch, learning rate, dan ukuran batch. Setiap kombinasi digunakan untuk melatih model dan dievaluasi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

Visualisasi bar plot

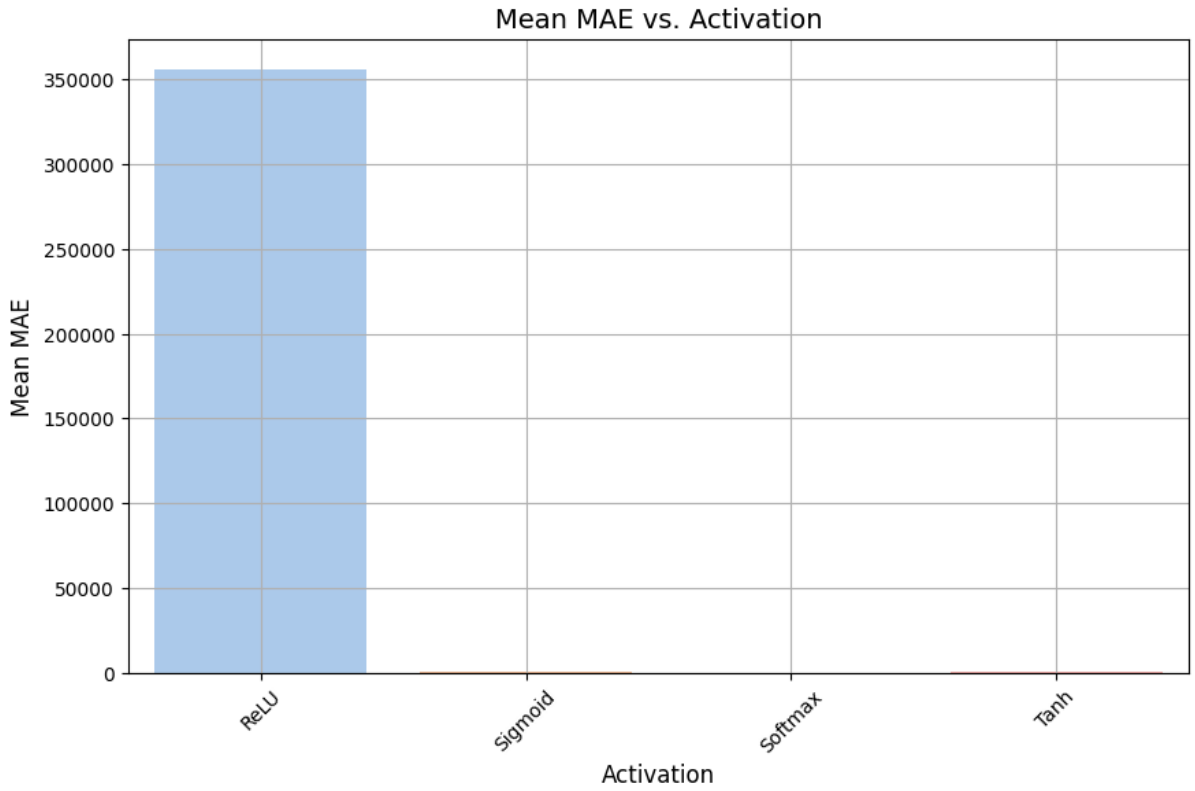
```
hyperparameters = ['layers', 'neurons', 'activation', 'epochs', 'lr', 'batch_size']
mean_mae_by_hyperparameter = results_df.groupby(hyperparameters)['mae'].mean().reset_index()
#memilih hyperparameter yang relevan dan menghitung rata-rata Mean Absolute Error (MAE)

for param in hyperparameters:
    plt.figure(figsize=(10, 6))
    sns.barplot(data=mean_mae_by_hyperparameter, x=param, y='mae', ci=None, palette="pastel")
    plt.title(f'Mean MAE vs. {param.capitalize()}', fontsize=14)
    plt.xlabel(param.capitalize(), fontsize=12)
    plt.ylabel('Mean MAE', fontsize=12)
    plt.xticks(rotation=45)
    plt.grid(True)
    plt.show()
#memvisualisasikan rata-rata MAE terhadap setiap hyperparameter.
```



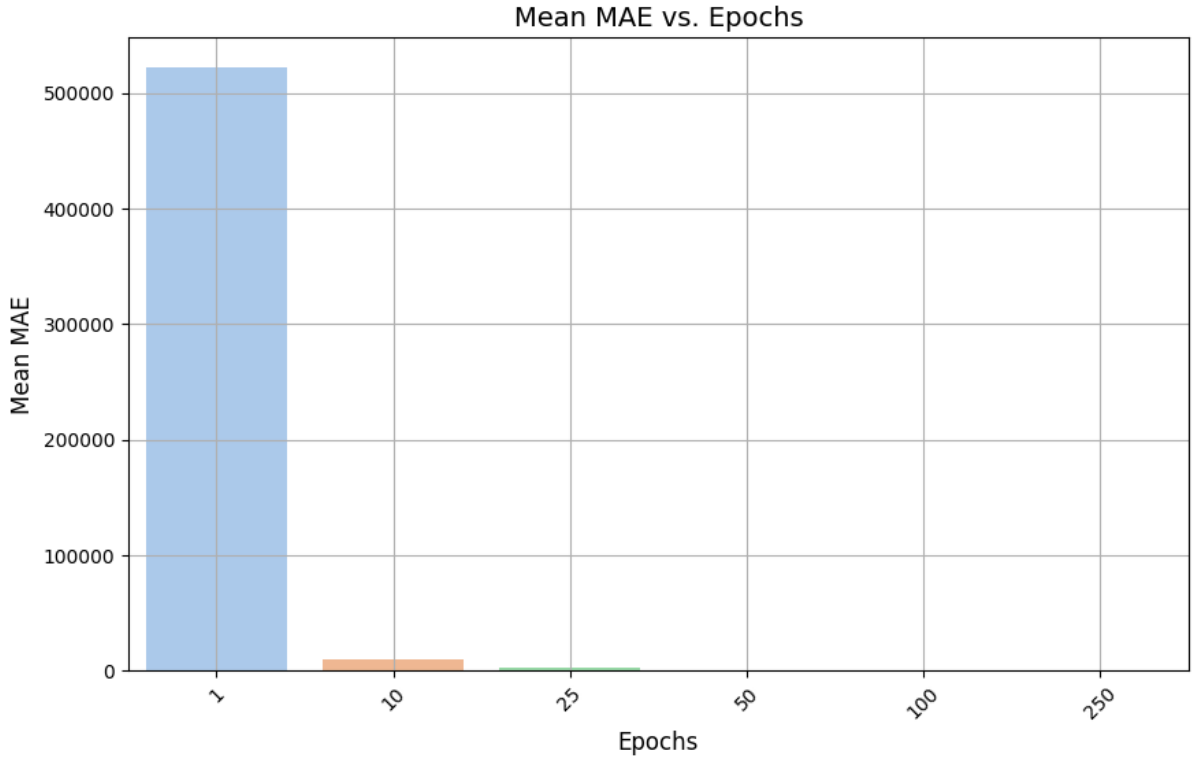
Pada grafik Mean MAE vs. Layers, terlihat bahwa peningkatan jumlah layer berdampak signifikan terhadap Mean Absolute Error (MAE). Dengan satu layer, MAE masih sangat rendah, sementara dua layer mulai meningkatkan MAE sedikit. Namun, ketika jumlah layer mencapai tiga, MAE melonjak drastis, menunjukkan bahwa penambahan layer yang berlebihan menyebabkan model menjadi kurang optimal atau sulit dilatih.



Pada grafik Mean MAE vs. Neurons, peningkatan jumlah neuron juga memberikan efek yang serupa. Dengan 1 dan 8 neuron, MAE tetap rendah, tetapi ketika jumlah neuron meningkat menjadi 16, 32, dan akhirnya 64, MAE naik tajam. Hal ini menunjukkan bahwa penambahan neuron yang berlebihan dapat menyebabkan model mengalami overfitting atau tidak stabil dalam proses pelatihan.

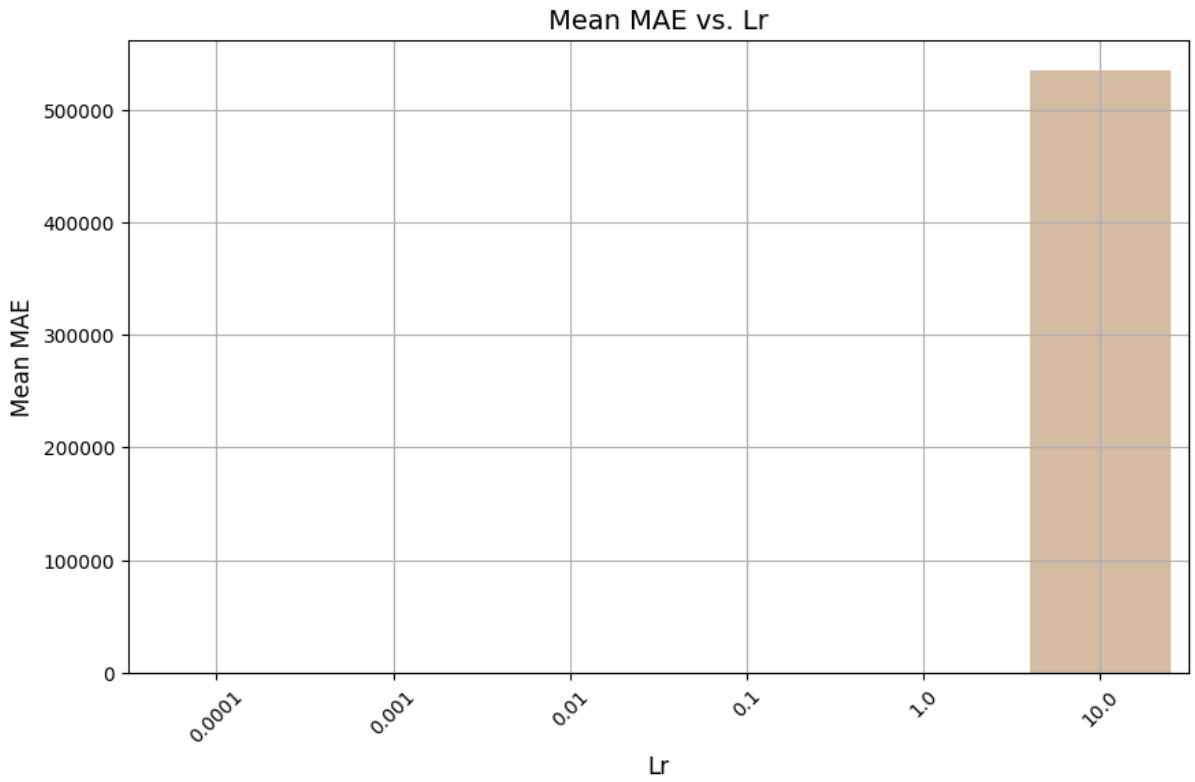


Pada grafik Mean MAE vs. Activation, fungsi aktivasi ReLU memiliki MAE tertinggi dibandingkan dengan fungsi aktivasi lainnya seperti Sigmoid, Softmax, dan Tanh. Hal ini mengindikasikan bahwa pemilihan fungsi aktivasi memainkan peran penting dalam performa model. Dalam hal ini, ReLU tampaknya menyebabkan masalah konvergensi atau overfitting, sedangkan fungsi aktivasi lainnya bekerja lebih baik.

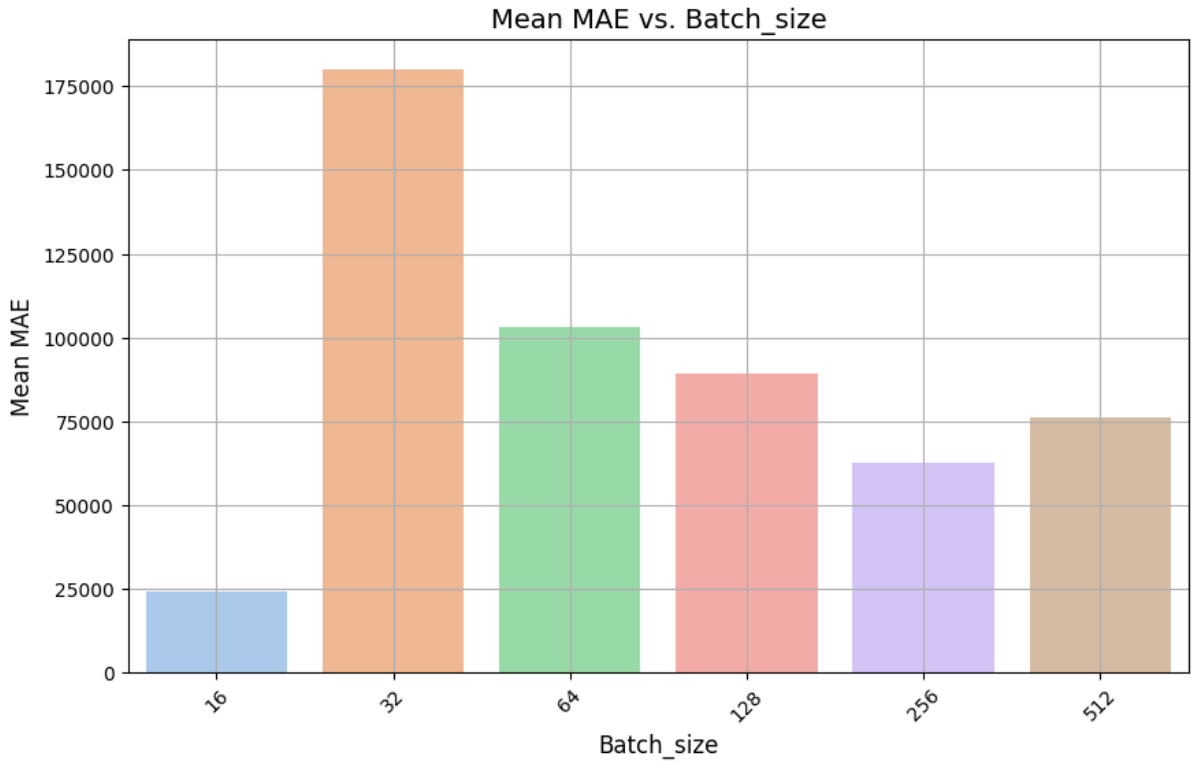


Mean MAE vs. Epochs menunjukkan bahwa jumlah epoch yang cukup penting untuk konvergensi model. Pada 1 epoch, MAE masih sangat tinggi karena model belum dilatih secara optimal. Dengan 10 hingga 25 epoch, MAE turun signifikan, dan pada 100 hingga 250 epoch, MAE tetap stabil di nilai yang rendah. Hal ini menunjukkan bahwa jumlah epoch yang moderat (sekitar 25) sudah cukup untuk mencapai performa terbaik.

Mean MAE vs. Learning Rate (Lr), terlihat bahwa learning rate yang terlalu besar, seperti 1.0 dan 10.0, menghasilkan MAE tertinggi, yang menandakan model gagal belajar atau menjadi tidak stabil. Sebaliknya, learning rate yang lebih kecil, seperti 0.0001 hingga 0.1, memberikan MAE yang rendah, menunjukkan proses pelatihan yang lebih stabil dan konvergen.



Mean MAE vs. Learning Rate (Lr), terlihat bahwa learning rate yang terlalu besar, seperti 1.0 dan 10.0, menghasilkan MAE tertinggi, yang menandakan model gagal belajar atau menjadi tidak stabil. Sebaliknya, learning rate yang lebih kecil, seperti 0.0001 hingga 0.1, memberikan MAE yang rendah, menunjukkan proses pelatihan yang lebih stabil dan konvergen.



Pada grafik Mean MAE vs. Batch Size, batch size kecil seperti 16 memberikan MAE yang rendah di awal, tetapi batch size 32 menyebabkan MAE melonjak tinggi. Dengan peningkatan batch size ke 64, 128, dan seterusnya, MAE mulai turun dan stabil, menunjukkan bahwa batch size yang lebih besar dapat memberikan keseimbangan antara stabilitas dan performa pelatihan model.