

Nama : Khalishah

NIM : 1103213045

## Laporan MLP Regression Machine Learning Week 10

Import library yg butuhkan:


Import libraries

```
[ ] Generated code may be subject to a license | UncleThree0402/PyTorch_FFN_HeartDisease | LPapakostas/welding_temp...
import pandas as pd
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Memuat dataset dan menampilkan 5 baris dataset:

Load dataset

```
[ ] df = pd.read_csv('/content/heart_failure_clinical_records_dataset.csv')
df.head()
```



	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_p
0	75.0	0	582	0	20	
1	55.0	0	7861	0	38	
2	65.0	0	146	0	20	
3	50.0	1	111	0	20	
4	65.0	1	160	1	20	

Next  
steps:

[Generate code with df](#)



[View recommended plots](#)

[New interactive sheet](#)

Menampilkan informasi dataset:

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
#   Column                                Non-Null Count  Dtype  
---  -
0   age                                   299 non-null   float64
1   anaemia                              299 non-null   int64  
2   creatinine_phosphokinase             299 non-null   int64  
3   diabetes                             299 non-null   int64  
4   ejection_fraction                   299 non-null   int64  
5   high_blood_pressure                  299 non-null   int64  
6   platelets                            299 non-null   float64
7   serum_creatinine                     299 non-null   float64
8   serum_sodium                         299 non-null   int64  
9   sex                                  299 non-null   int64  
10  smoking                              299 non-null   int64  
11  time                                 299 non-null   int64  
12  DEATH_EVENT                          299 non-null   int64  
dtypes: float64(3), int64(10)
memory usage: 30.5 KB
```

Memberikan ringkasan statistik dari kolom numerik dalam DataFrame, seperti rata-rata, standar deviasi, dan nilai minimum/maksimum:

```
[ ] df.describe()
```

```

      age  anaemia  creatinine_phosphokinase  diabetes  ejection_fraction
count 299.000000  299.000000                299.000000  299.000000  299.000000
mean  60.833893   0.431438                  581.839465   0.418060   38.083612
std   11.894809   0.496107                  970.287881   0.494067   11.834847
min   40.000000   0.000000                   23.000000   0.000000   14.000000
25%   51.000000   0.000000                  116.500000   0.000000   30.000000
50%   60.000000   0.000000                  250.000000   0.000000   38.000000
75%   70.000000   1.000000                  582.000000   1.000000   45.000000
max   95.000000   1.000000                 7861.000000   1.000000   80.000000
```

Menghitung jumlah nilai NaN(Not a Number) atau data yang hilang di setiap kolom dalam DataFrame:

```
[ ] df.isna().sum()
```



	0
age	0
anaemia	0
creatinine_phosphokinase	0
diabetes	0
ejection_fraction	0
high_blood_pressure	0
platelets	0
serum_creatinine	0
serum_sodium	0
sex	0
smoking	0
time	0
DEATH_EVENT	0

dtype: int64

Menampilkan kolom pada dataset:

```
[ ] df.columns
```



```
Index(['age', 'anaemia', 'creatinine_phosphokinase', 'diabetes',  
      'ejection_fraction', 'high_blood_pressure', 'platelets',  
      'serum_creatinine', 'serum_sodium', 'sex', 'smoking', 'time',  
      'DEATH_EVENT'],  
      dtype='object')
```

Mengonversi kolom-kolom dalam yang berisi kategori:

Encoding categorical columns using LabelEncoder

```
[ ] label_cols = ['anaemia', 'diabetes', 'high_blood_pressure', 'sex', 'smoking', 'DEATH_EVENT']  
    label_encoder = LabelEncoder()  
  
    for col in label_cols:  
        df[col] = label_encoder.fit_transform(df[col])
```

Memisahkan fitur (X) dan target (y) dalam dataset:

Determine features (X) and targets (y)

```
[ ] x = df.drop(['DEATH_EVENT'], axis=1)
    y = df['DEATH_EVENT']
```

```
[ ] x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Melakukan standaeisasi fitur agar memiliki skala yang sama (mean = 0, standar deviasi = 1):

Standardization of numeric features

```
[ ] scaler = StandardScaler()
    x_train_scaled = scaler.fit_transform(X_train)
    x_test_scaled = scaler.transform(X_test)
```

Mengonversi data latih dan uji menjadi tensor PyTorch, yang diperlukan saat bekerja dengan model pembelajaran mesin di PyTorch:

Convert data to tensor

```
[ ] 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)
```

Kelas MLPRegression adalah implementasi model jaringan saraf multilayer perceptron (MLP) untuk tugas regresi menggunakan PyTorch. Model ini terdiri dari lapisan input, beberapa lapisan tersembunyi, dan satu lapisan output. Pada inisialisasi, pengguna dapat menentukan jumlah fitur input, jumlah lapisan tersembunyi, jumlah neuron di setiap lapisan tersembunyi, dan fungsi aktivasi yang digunakan (misalnya, ReLU). Fungsi forward mendefinisikan bagaimana input melewati model untuk menghasilkan output, di mana data mengalir melalui lapisan linear dan fungsi aktivasi. Model ini cocok untuk masalah regresi dengan output kontinu, karena lapisan output hanya menghasilkan satu nilai.

## Constructing an MLP model for regression

```
[ ] 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

        # Create input to hidden layer
        layers = []
        layers.append(nn.Linear(self.input_size, self.neurons))

        # Add hidden layers
        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, 1)) # Output layer

        self.model = nn.Sequential(*layers)

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

Kode ini melakukan eksperimen untuk menguji berbagai kombinasi hiperparameter dalam pelatihan model MLPRegression. Hyperparameter yang diuji meliputi jumlah lapisan tersembunyi, jumlah neuron, fungsi aktivasi, jumlah epoch, laju pembelajaran, dan ukuran batch. Model dilatih menggunakan algoritma Adam dan dihitung menggunakan Mean Squared Error (MSE) sebagai fungsi kerugian. Setelah pelatihan, model dievaluasi menggunakan metrik seperti Mean Absolute Error (MAE), Mean Squared Error (MSE), dan  $R^2$  Score pada data uji. Hasil eksperimen, termasuk kombinasi hiperparameter dan metrik kinerja, disimpan dalam sebuah daftar untuk analisis lebih lanjut, memungkinkan untuk menentukan konfigurasi terbaik untuk model.

[illegible]

```

# Define loss function and optimizer
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=lr)

# Training loop
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()

# Evaluate model after training
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 # Accuracy estimation

# Store results
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 eksperimen dalam sebuah DataFrame pandas dan kemudian mengekspor data tersebut ke dalam file CSV:

Convert the results to a DataFrame and save them to CSV.

```

[ ] results_df = pd.DataFrame(results)
    results_df.to_csv("mlp_regression_hidden layer 123.csv", index=False)
    print("All results have been saved to 'mlp_regression_hidden layer 123.csv'.")

```

➡ All results have been saved to 'mlp\_regression\_hidden layer 123.csv'.

Menganalisis pengaruh masing-masing hiperparameter terhadap Mean Absolute Error (MAE) yang dihitung selama eksperimen, dengan memvisualisasikannya menggunakan bar plot:

Select relevant hyperparameters and mean MAE

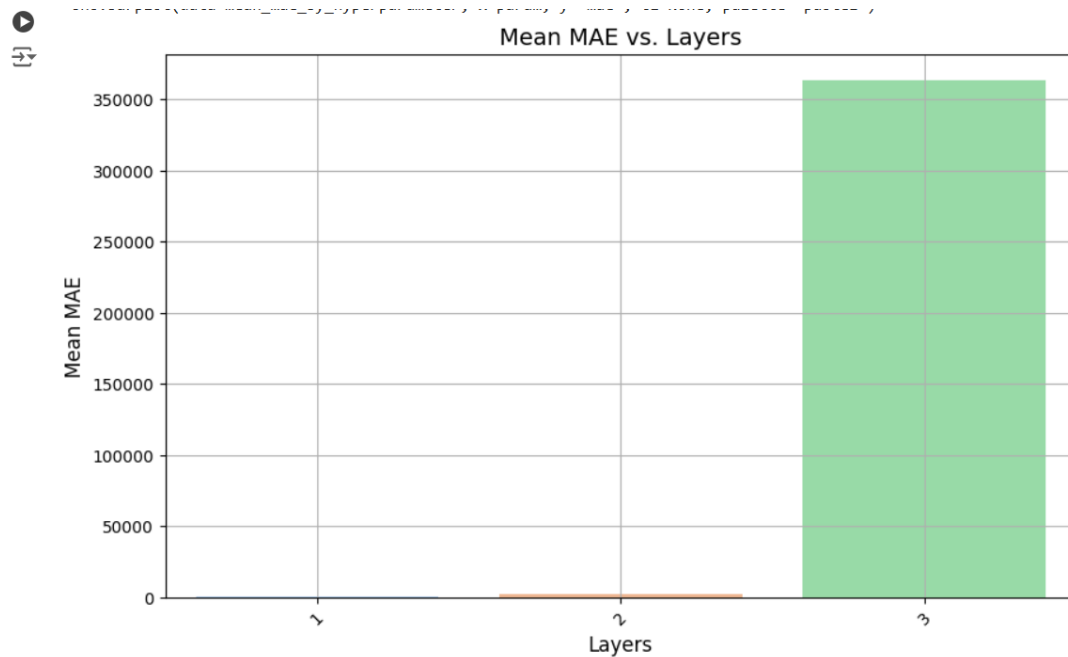
```

hyperparameters = ['layers', 'neurons', 'activation', 'epochs', 'lr', 'batch_size']
mean_mae_by_hyperparameter = results_df.groupby(hyperparameters)['mae'].mean().reset_index()

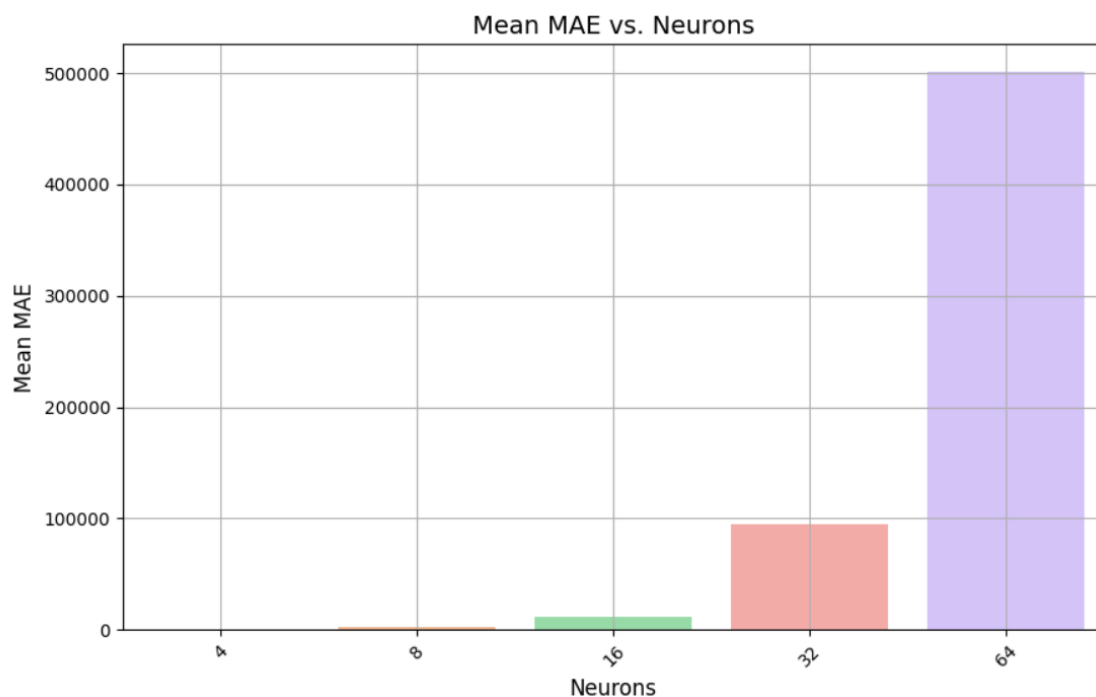
# Plot mean MAE against each hyperparameter
for param in hyperparameters:
    plt.figure(figsize=(10, 6))
    sns.barplot(data=mean_mae_by_hyperparameter, x=param, y='mae', ci=None, palette="pastel")
    plt.title(f'Mean MAE vs. {param.capitalize()}', fontsize=14)
    plt.xlabel(param.capitalize(), fontsize=12)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

```

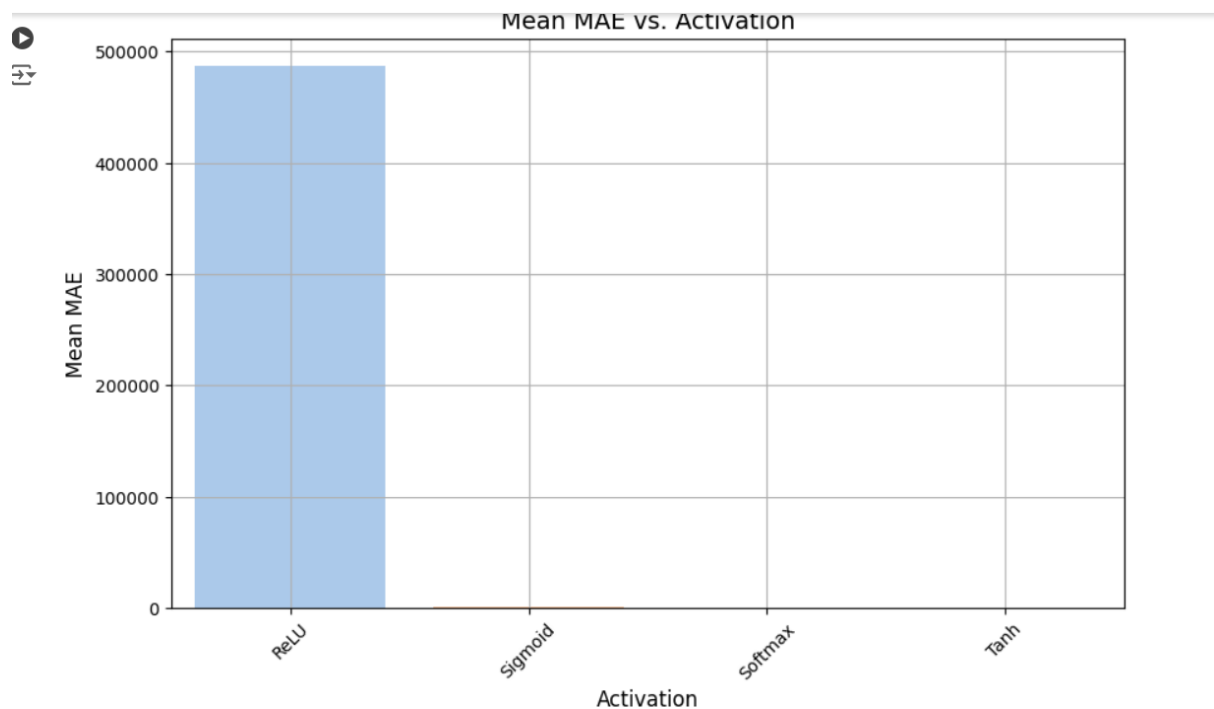
Mean MAE vs. Layers: Diagram ini menunjukkan hubungan antara jumlah lapisan tersembunyi dan MAE rata-rata. Tampaknya, MAE sangat tinggi untuk model dengan 3 lapisan tersembunyi, sedangkan untuk 1 dan 2 lapisan MAE jauh lebih rendah.



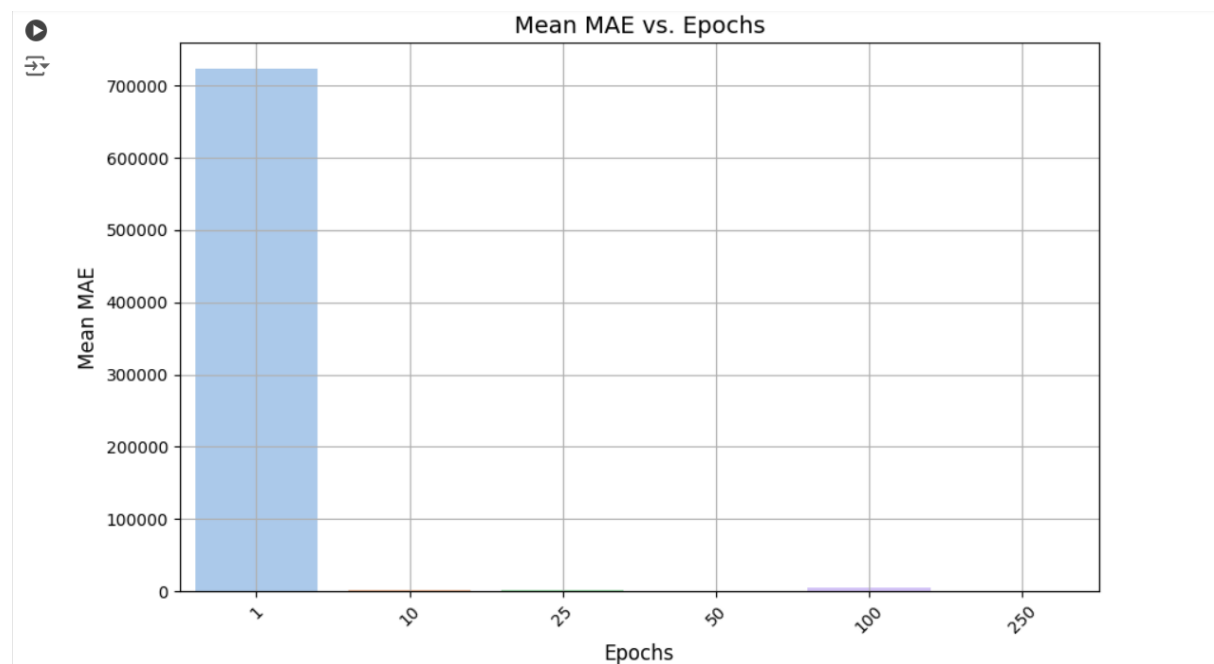
Mean MAE vs. Neurons: Diagram ini menggambarkan pengaruh jumlah neuron di setiap lapisan terhadap MAE. Dapat dilihat bahwa untuk jumlah neuron yang lebih tinggi (misalnya, 64 neuron), MAE meningkat secara signifikan, sedangkan jumlah neuron yang lebih rendah menghasilkan MAE yang lebih rendah.



Mean MAE vs. Activation: Diagram ini memperlihatkan pengaruh fungsi aktivasi terhadap MAE. Semua fungsi aktivasi (ReLU, Sigmoid, Softmax, Tanh) menghasilkan MAE yang sangat tinggi, tetapi ReLU tampaknya sedikit lebih baik dibandingkan dengan yang lain, meskipun MAE tetap sangat tinggi.

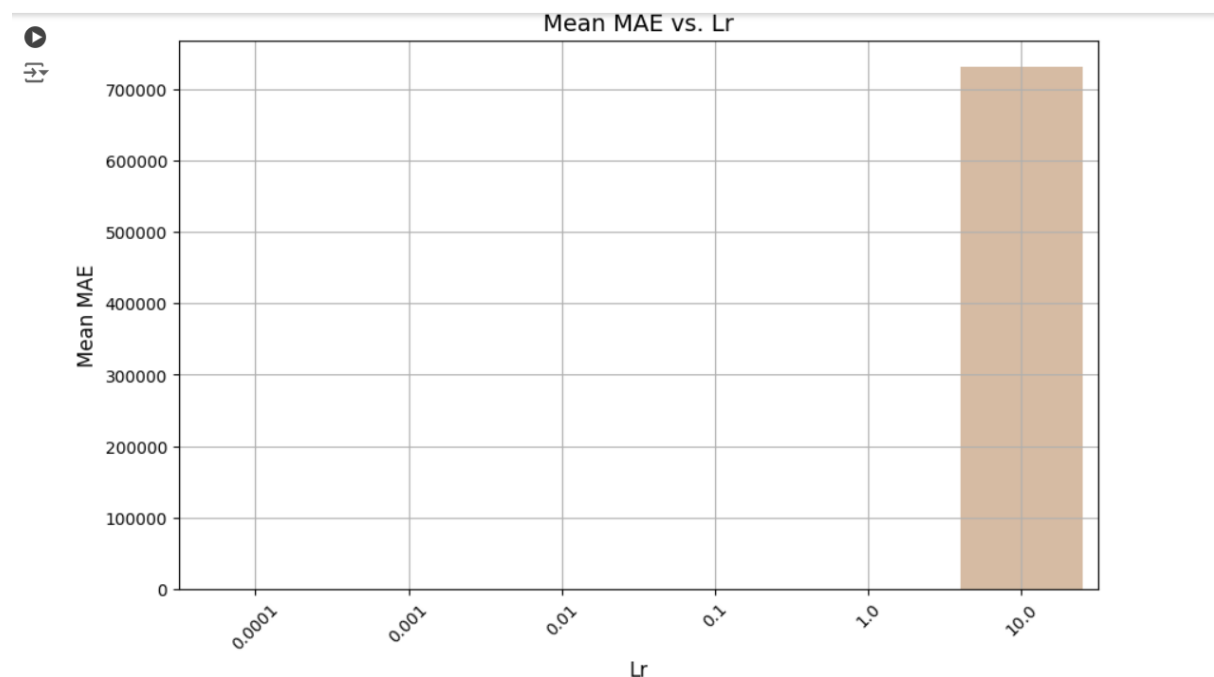


Mean MAE vs. Epochs: Diagram ini menunjukkan dampak jumlah epoch terhadap MAE. Terlihat bahwa meskipun jumlah epoch meningkat (hingga 250 epoch), MAE tetap tinggi dan hampir tidak berubah, yang menunjukkan model mungkin tidak terlatih dengan baik atau ada masalah dalam pelatihan.





Mean MAE vs. Learning Rate (LR): Diagram ini menggambarkan hubungan antara laju pembelajaran dan MAE. Laju pembelajaran yang sangat tinggi (1 dan 10) menghasilkan MAE yang sangat tinggi, sementara laju pembelajaran yang lebih rendah (seperti 0.001 dan 0.0001) menghasilkan MAE yang jauh lebih rendah.



Mean MAE vs. Batch Size: Diagram ini memperlihatkan pengaruh ukuran batch terhadap MAE. Ukuran batch 64 memberikan hasil terbaik dengan MAE yang lebih rendah dibandingkan dengan ukuran batch lainnya. Ukuran batch yang lebih besar atau lebih kecil menyebabkan MAE yang lebih tinggi.

