Nama : Az – Zahra Chikal E.

NIM : 1103213039

Kelas : TK-45-05

TUGAS WEEK 10 MLP CLASSIFICATION

Link Google Colab (Apabila yang di zip tidak dapat dibuka):

https://colab.research.google.com/drive/1ztl55VCbNTM7EqQEPo7fjRTwwbXzLVVK?usp=sharing

```python
# Import Library

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, roc_curve, auc, confusion_matrix
import torch
import torch.nn as nn
import torch.optim as optim
```

✓ 3.1s

```python
# membaca/load dataset
df = pd.read_csv('diabetes_binary_5050split_health_indicators_BRFSS2015.csv')

# menampilkan 5 baris pertama
df.head(5)
```

✓ 0.0s

```python
# informasi mengenai dataset
df.info()
```

✓ 0.0s

Output:

	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	...	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex	Age	Education	Income
0	0.0	1.0	0.0	1.0	26.0	0.0	0.0	0.0	1.0	0.0	...	1.0	0.0	3.0	5.0	30.0	0.0	1.0	4.0	6.0	8.0
1	0.0	1.0	1.0	1.0	26.0	1.0	1.0	0.0	0.0	1.0	...	1.0	0.0	3.0	0.0	0.0	0.0	1.0	12.0	6.0	8.0
2	0.0	0.0	0.0	1.0	26.0	0.0	0.0	0.0	1.0	1.0	...	1.0	0.0	1.0	0.0	10.0	0.0	1.0	13.0	6.0	8.0
3	0.0	1.0	1.0	1.0	28.0	1.0	0.0	0.0	1.0	1.0	...	1.0	0.0	3.0	0.0	3.0	0.0	1.0	11.0	6.0	8.0
4	0.0	0.0	0.0	1.0	29.0	1.0	0.0	0.0	1.0	1.0	...	1.0	0.0	2.0	0.0	0.0	0.0	0.0	8.0	5.0	8.0

5 rows × 22 columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70692 entries, 0 to 70691
Data columns (total 22 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Diabetes_binary       70692 non-null  float64
 1   HighBP                70692 non-null  float64
 2   HighChol              70692 non-null  float64
 3   CholCheck             70692 non-null  float64
 4   BMI                   70692 non-null  float64
 5   Smoker                70692 non-null  float64
 6   Stroke                70692 non-null  float64
 7   HeartDiseaseorAttack  70692 non-null  float64
 8   PhysActivity          70692 non-null  float64
 9   Fruits                70692 non-null  float64
 10  Veggies               70692 non-null  float64
 11  HvyAlcoholConsump     70692 non-null  float64
 12  AnyHealthcare         70692 non-null  float64
 13  NoDocbcCost           70692 non-null  float64
 14  GenHlth               70692 non-null  float64
 15  MentHlth              70692 non-null  float64
 16  PhysHlth              70692 non-null  float64
 17  DiffWalk              70692 non-null  float64
 18  Sex                   70692 non-null  float64
 19  Age                   70692 non-null  float64
 20  Education             70692 non-null  float64
 21  Income                70692 non-null  float64
dtypes: float64(22)
memory usage: 11.9 MB
```

Analisis:

Kode di atas digunakan untuk memuat library untuk analisis data, pembelajaran mesin, dan visualisasi, seperti pandas, PyTorch, dan seaborn. Dataset diabetes_binary_5050split_health_indicators_BRFSS2015.csv. diimpor menggunakan pd.read_csv(), dan metode seperti .head(), .info(), dan .isnull().sum() digunakan untuk mengeksplorasi data, termasuk menampilkan 5 baris pertama, informasi kolom, tipe data, dan jumlah nilai kosong.

```python
# cek missing value
df.isnull().sum()
```

✓ 0.0s Python

```python
# Drop duplicate rows
df = df.drop_duplicates()

# cek informasi dataset setelah menghapus yang duplicate
df.info()
```

✓ 0.0s Python

Output:

```
Diabetes_binary          0
HighBP                   0
HighChol                 0
CholCheck                0
BMI                      0
Smoker                   0
Stroke                   0
HeartDiseaseorAttack     0
PhysActivity             0
Fruits                   0
Veggies                  0
HvyAlcoholConsump        0
AnyHealthcare            0
NoDocbcCost              0
GenHlth                  0
MentHlth                 0
PhysHlth                 0
DiffWalk                 0
Sex                      0
Age                      0
Education                0
Income                   0
dtype: int64
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 69057 entries, 0 to 70691
Data columns (total 22 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Diabetes_binary       69057 non-null  float64
 1   HighBP                69057 non-null  float64
 2   HighChol              69057 non-null  float64
 3   CholCheck             69057 non-null  float64
 4   BMI                   69057 non-null  float64
 5   Smoker                69057 non-null  float64
 6   Stroke                69057 non-null  float64
 7   HeartDiseaseorAttack  69057 non-null  float64
 8   PhysActivity          69057 non-null  float64
 9   Fruits                69057 non-null  float64
 10  Veggies               69057 non-null  float64
 11  HvyAlcoholConsump     69057 non-null  float64
 12  AnyHealthcare         69057 non-null  float64
 13  NoDocbcCost           69057 non-null  float64
 14  GenHlth               69057 non-null  float64
 15  MentHlth              69057 non-null  float64
 16  PhysHlth              69057 non-null  float64
 17  DiffWalk              69057 non-null  float64
 18  Sex                   69057 non-null  float64
 19  Age                   69057 non-null  float64
 20  Education             69057 non-null  float64
 21  Income                69057 non-null  float64
dtypes: float64(22)
memory usage: 12.1 MB
```

Analisis:

Kode tersebut menghapus duplikasi data dalam dataset menggunakan df.drop_duplicates() dan mengecek kembali struktur data dengan df.info(), yang memberikan informasi seperti jumlah kolom, tipe data, dan jumlah entri non-kosong. Selanjutnya, df.describe() digunakan untuk menghitung statistik deskriptif seperti rata-rata, standar deviasi, nilai minimum, dan maksimum untuk kolom numerik, yang membantu memahami distribusi data.

```python
# Melihat jumlah data pada kolom
df['Diabetes_binary'].value_counts()
```

✓ 0.0s

Output:

```
Diabetes_binary
1.0    35097
0.0    33960
Name: count, dtype: int64
```

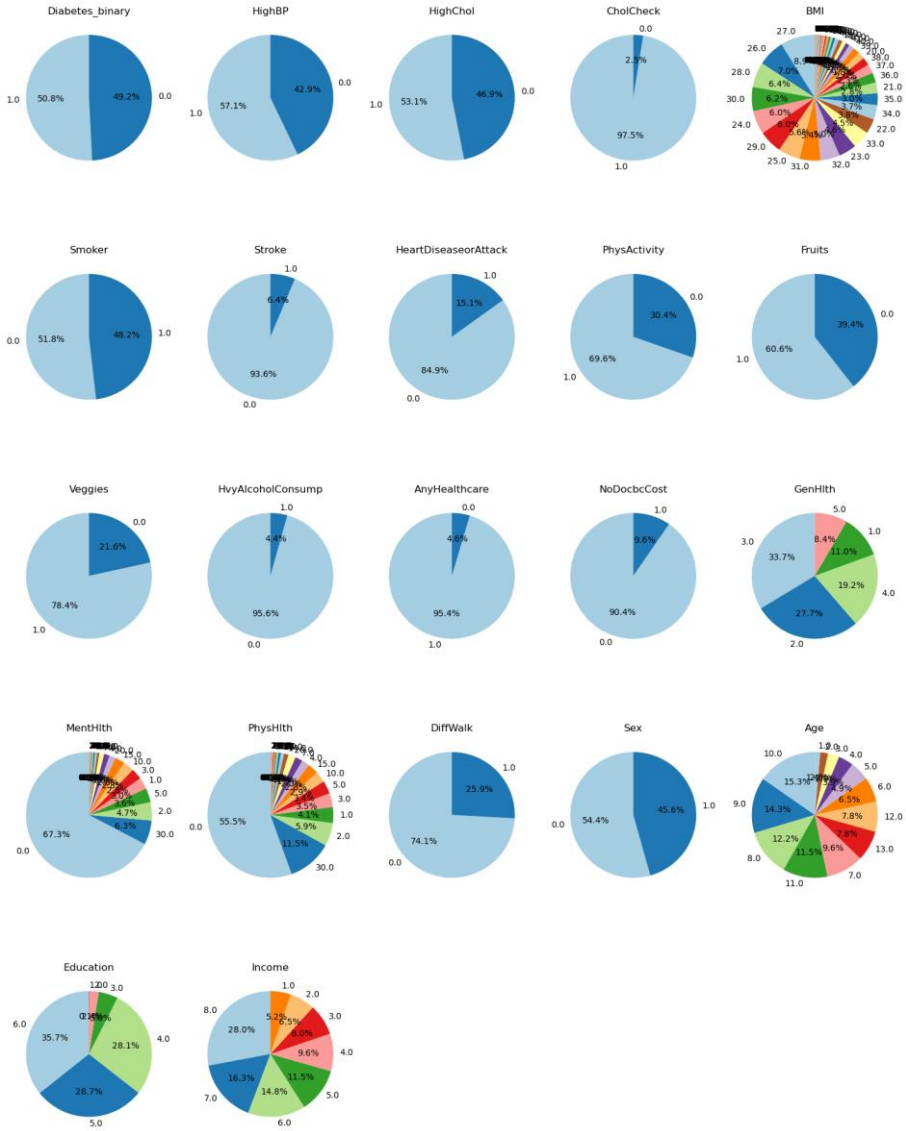
Analisis:

Kode di atas digunakan untuk menghitung nilai pada kolom 1 (Diabetes) dan 0 (Tidak diabetes). Untuk dataset ini digunakan yang sudah di split 5050 saat pengujian data yang digunakan sudah seimbang.

```python
# Function to plot pie charts for all columns in the dataset
def plot_pie_charts(dataframe, num_cols=5):
    cols = dataframe.columns
    total_plots = len(cols)
    rows = (total_plots + num_cols - 1) // num_cols  # Calculate required rows

    fig, axes = plt.subplots(rows, num_cols, figsize=(15, rows * 4))
    axes = axes.flatten()  # Flatten in case of multiple rows

    # Plot pie charts
    for idx, col in enumerate(cols):
        values = dataframe[col].value_counts()
        axes[idx].pie(values, labels=values.index, autopct='%1.1f%%', startangle=90, colors=plt.cm.Paired.colors)
        axes[idx].set_title(col)

    # Hide unused subplots
    for ax in axes[total_plots:]:
        ax.axis('off')

    plt.tight_layout()
    plt.show()

# Plot pie charts for all columns in the dataset
plot_pie_charts(df)
```

✓ 1.5s Python

Output:



Analisis:

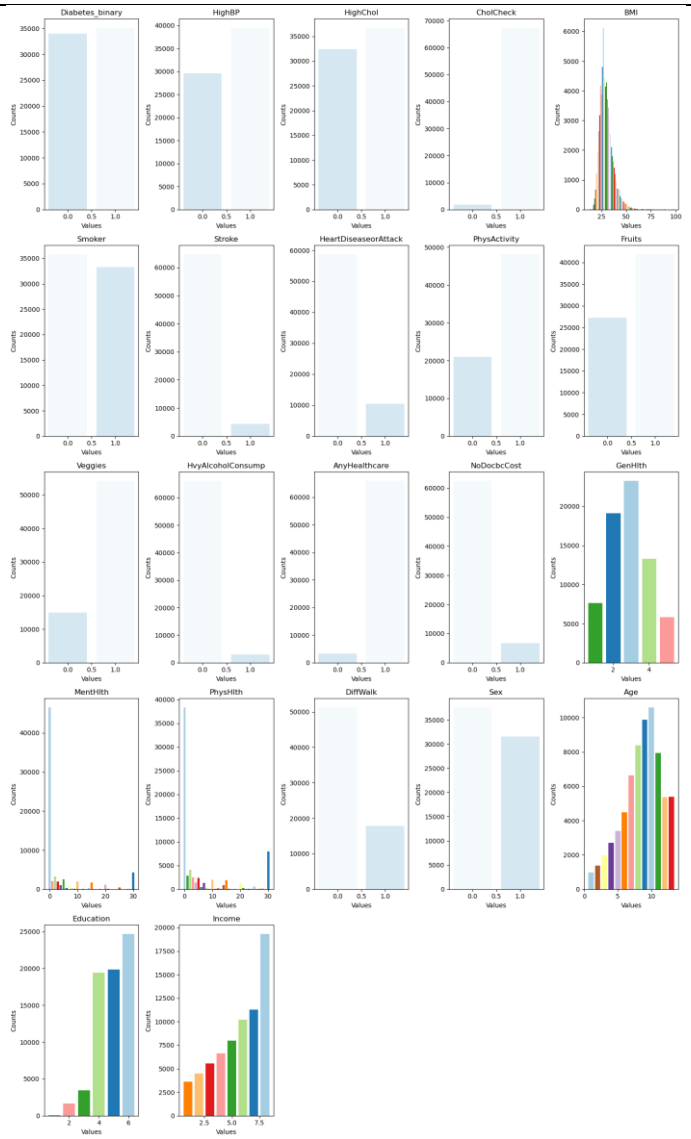
Kode diatas diguanakan untuk membuat diagram pie chart dari tiap kolom/fitur yang

terdapat pada dataset tersebut. dataframe (data yang ingin divisualisasikan) dan num_cols (jumlah kolom per baris dalam grid plot, default-nya adalah 5). Untuk setiap kolom, akan dihitung nilai – nilai nya dan akan ditampilkan dalam bentuk presentase.

```python
# Function to plot bar charts for all columns in the dataset
def plot_bar_charts(dataframe, num_cols=5):
    cols = dataframe.columns
    total_plots = len(cols)
    rows = (total_plots + num_cols - 1) // num_cols  # Calculate required rows

    fig, axes = plt.subplots(rows, num_cols, figsize=(15, rows * 5))
    axes = axes.flatten()  # Flatten in case of multiple rows

    for idx, col in enumerate(cols):
        values = dataframe[col].value_counts()
        axes[idx].bar(values.index, values, color=plt.cm.Paired.colors[:len(values)])
        axes[idx].set_title(col)
        axes[idx].set_xlabel('Values')
        axes[idx].set_ylabel('Counts')

    # Hide unused subplots
    for ax in axes[total_plots:]:
        ax.axis('off')

    plt.tight_layout()
    plt.show()

# Plot bar charts for all columns in the dataset
plot_bar_charts(df)
```

✓ 1.8s Python

Output:



Analisis:

Kode diatas diguanakan untuk membuat diagram bar chart dari tiap kolom/fitur yang terdapat pada dataset tersebut. dataframe (data yang ingin divisualisasikan) dan num_cols (jumlah kolom per baris dalam grid plot, default-nya adalah 5). Untuk setiap kolom, akan dihitung nilai – nilai nya dan akan ditampilkan. Untuk sumbu y menggunakan fitur count sedangkan untuk sumbu x dilabelkan fitur – fitur yang terdapat pada dataset tersebut.

```
# Deskripsi statistik
df.describe()
```

0.0s

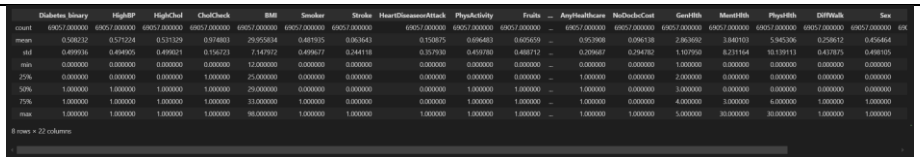
Python

```
import seaborn as sns
import matplotlib.pyplot as plt

def plot_heatmap(dataframe):
    numeric_cols = dataframe.select_dtypes(include=['int64', 'float64']) # Select numerical columns
    corr_matrix = numeric_cols.corr() # Compute correlation matrix

    plt.figure(figsize=(15, 15))
    sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap="coolwarm", cbar=True)
    plt.title("Heatmap of Correlation Between Numerical Columns")
    plt.show()

# Plot heatmap for the dataset
plot_heatmap(df)
```

0.7s

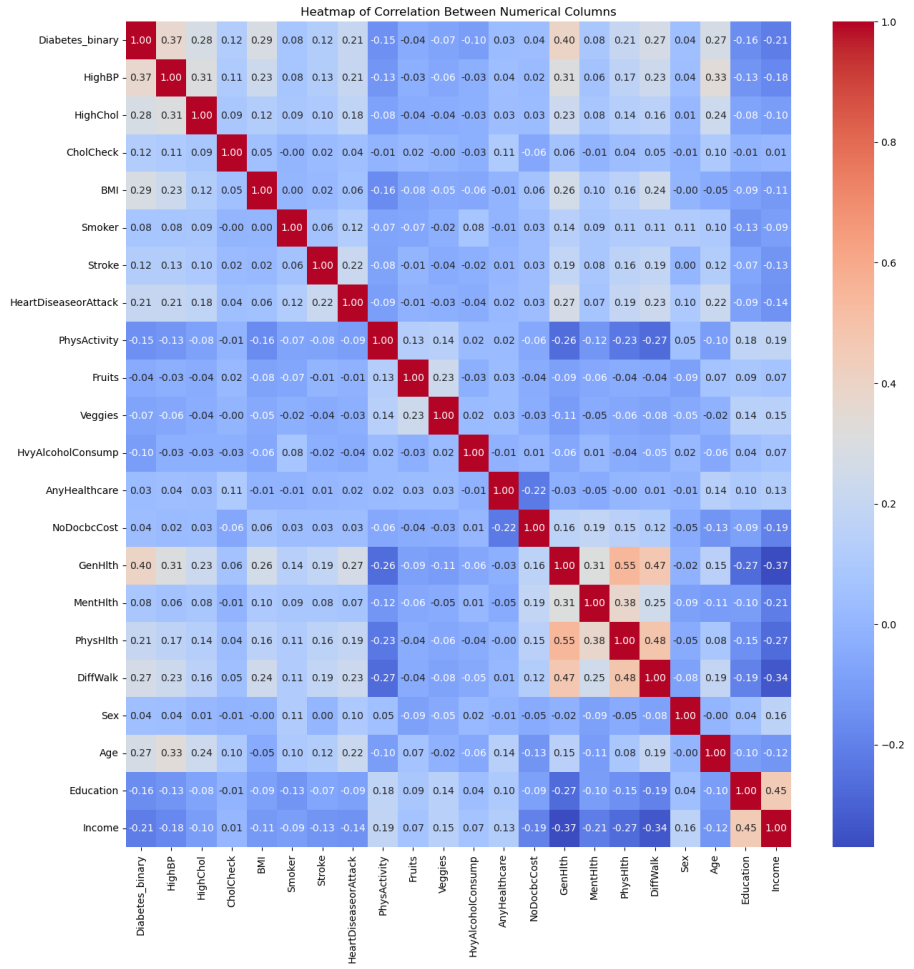
Python

Output:



	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	...	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex
count	69057.000000	69057.000000	69057.000000	69057.000000	69057.000000	69057.000000	69057.000000	69057.000000	69057.000000	69057.000000	...	69057.000000	69057.000000	69057.000000	69057.000000	69057.000000	69057.000000	69057.000000
mean	0.508232	0.571224	0.531329	0.974803	29.955834	0.481935	0.063643	0.159875	0.696483	0.605659	...	0.953908	0.096138	2.863692	3.840103	5.945306	0.258612	0.456464
std	0.499936	0.494905	0.499021	0.156723	7.147972	0.499677	0.244118	0.357930	0.459780	0.488712	...	0.209687	0.294782	1.107958	8.231164	10.139113	0.437875	0.498105
min	0.000000	0.000000	0.000000	0.000000	12.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	1.000000	25.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	1.000000	0.000000	2.000000	0.000000	0.000000	0.000000	0.000000
50%	1.000000	1.000000	1.000000	1.000000	29.000000	0.000000	0.000000	0.000000	1.000000	1.000000	...	1.000000	0.000000	3.000000	0.000000	0.000000	0.000000	0.000000
75%	1.000000	1.000000	1.000000	1.000000	33.000000	1.000000	0.000000	0.000000	1.000000	1.000000	...	1.000000	0.000000	4.000000	3.000000	6.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	98.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	5.000000	30.000000	30.000000	1.000000	1.000000

8 rows × 22 columns



Analisis:

Kode di atas mendefinisikan fungsi Python bernama plot_heatmap yang bertujuan untuk memvisualisasikan korelasi antar kolom numerik dalam sebuah dataset menggunakan heatmap. Fungsi ini pertama-tama memilih kolom dengan tipe data numerik (int64 dan float64) menggunakan metode select_dtypes, sehingga hanya fitur numerik yang dianalisis. Selanjutnya, matriks korelasi dihitung menggunakan fungsi corr, yang menghasilkan nilai-nilai korelasi antar kolom dalam rentang -1 hingga +1. Nilai +1 menunjukkan hubungan linier positif sempurna, sedangkan -1 menunjukkan hubungan linier negatif sempurna. Nilai mendekati 0 mengindikasikan tidak adanya hubungan linier.

```
# Menghitung matriks korelasi
correlation_matrix = df.corr()

# Menampilkan matriks korelasi sebagai tabel
print("Correlation Matrix:")
print(correlation_matrix)
```

0.0s Python

Output:

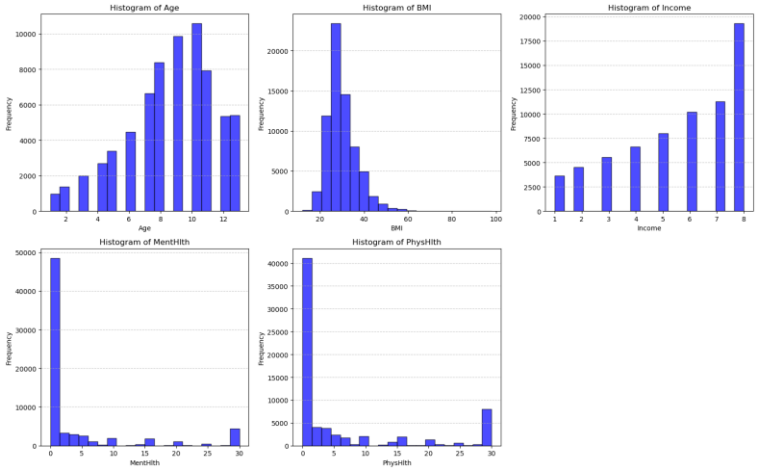
```
Correlation Matrix:
                      Diabetes_binary    HighBP  HighChol  CholCheck  \
Diabetes_binary              1.000000  0.372048  0.281399   0.118900
HighBP                       0.372048  1.000000  0.308987   0.106593
HighChol                     0.281399  0.308987  1.000000   0.088231
CholCheck                    0.118900  0.106593  0.088231   1.000000
BMI                          0.285643  0.232372  0.123917   0.047779
Smoker                       0.075853  0.078123  0.086522  -0.002854
Stroke                       0.122727  0.126869  0.098166   0.023368
HeartDiseaseorAttack         0.207229  0.206776  0.178207   0.044795
PhysActivity                -0.150281 -0.128307 -0.084469  -0.010072
Fruits                      -0.044560 -0.031818 -0.040783   0.015853
Veggies                     -0.072181 -0.059824 -0.037801  -0.001040
HvyAlcoholConsump           -0.098709 -0.029764 -0.027259  -0.026850
AnyHealthcare                0.027034  0.039659  0.034352   0.106549
NoDocbcCost                  0.036145  0.021802  0.029976  -0.061975
GenHlth                      0.396571  0.308459  0.227588   0.063116
MentHlth                     0.080688  0.058133  0.079929  -0.009365
PhysHlth                     0.206868  0.167821  0.138266   0.036442
DiffWalk                     0.267082  0.229638  0.157859   0.046421
Sex                          0.042538  0.037824  0.013250  -0.008116
Age                          0.274550  0.333721  0.235779   0.103414
Education                   -0.158522 -0.130037 -0.075364  -0.011266
Income                      -0.212846 -0.176360 -0.098712   0.005067
...
Education            -0.101774   1.000000  0.450376
Income               -0.124261   0.450376  1.000000

[22 rows x 22 columns]
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Analisis:

Kode di atas digunakan untuk menghitung dan menampilkan matriks korelasi antar kolom dalam sebuah dataset. nilai-nilai korelasi antar kolom dalam rentang -1 hingga +1. Nilai +1 menunjukkan hubungan linier positif sempurna, sedangkan -1 menunjukkan hubungan linier negatif sempurna. Nilai mendekati 0 mengindikasikan tidak adanya hubungan linier.

```
import matplotlib.pyplot as plt
# Creating histograms for relevant numerical columns
columns_to_plot = ['Age', 'BMI', 'Income', 'MentHlth', 'PhysHlth']

# Plotting histograms for each selected column
plt.figure(figsize=(16, 10))
for i, col in enumerate(columns_to_plot, 1):
    plt.subplot(2, 3, i)
    plt.hist(df[col], bins=20, color='blue', alpha=0.7, edgecolor='black')
    plt.title(f'Histogram of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
✓ 0.4s
```

Output:



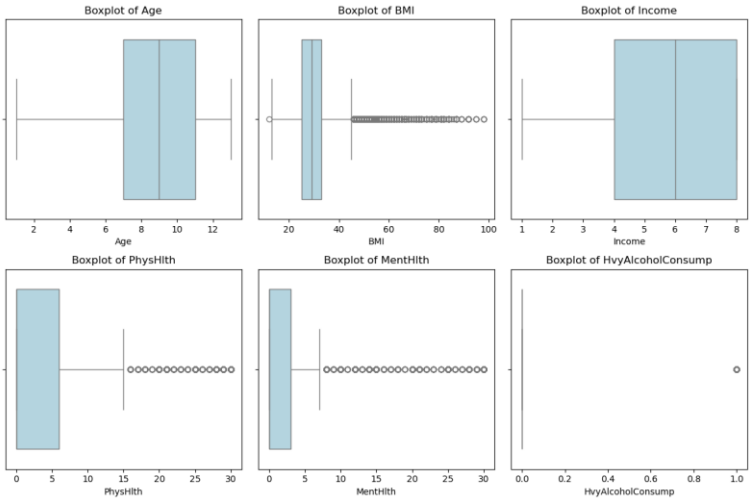
Analisis:

Kode ini menghasilkan boxplot untuk setiap kolom numerik dalam dataset menggunakan `sns.boxplot()`. Boxplot digunakan untuk mendeteksi outlier atau pencilan dalam data. Boxplot menunjukkan distribusi data melalui kuartil, dengan garis tengah (mediana), kotak yang mewakili rentang interkuartil, dan garis keluar (whiskers) yang menunjukkan rentang data yang dianggap normal. Titik-titik di luar whiskers dianggap sebagai outlier. Setiap boxplot membantu untuk memvisualisasikan sebaran data dan mengidentifikasi apakah ada nilai yang tidak biasa dalam setiap fitur numerik.

```python
# Daftar fitur  untuk box plot
columns_to_plot = ['Age', 'BMI', 'Income', 'PhysHlth', 'MentHlth','HvyAlcoholConsump']

# Set ukuran plot
plt.figure(figsize=(12, 8))

# Membuat boxplot untuk setiap fitur yang relevan
for i, column in enumerate(columns_to_plot, 1):
    plt.subplot(2, 3, i)  # Menampilkan dalam grid 2x3
    sns.boxplot(x=df[column], color='lightblue')
    plt.title(f'Boxplot of {column}')
    plt.xlabel(column)

plt.tight_layout()
plt.show()
```

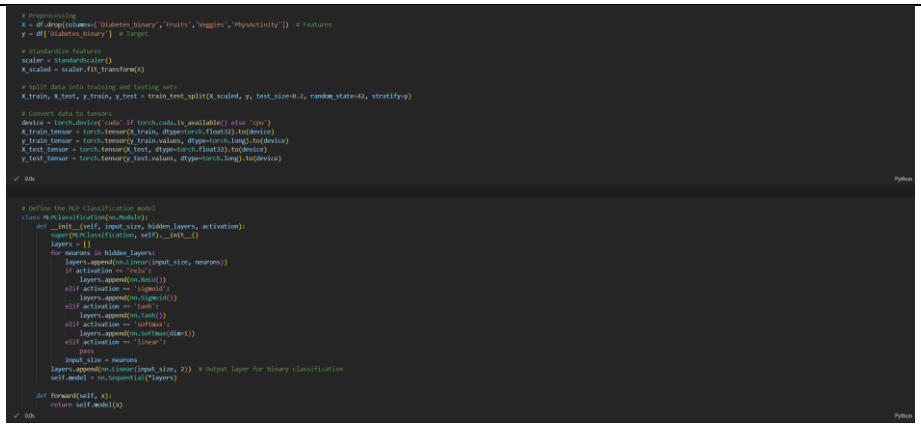
✓ 0.6s

Output:



Analisis:

Kode ini menghasilkan boxplot untuk setiap kolom numerik dalam dataset menggunakan `sns.boxplot()`. Boxplot digunakan untuk mendeteksi outlier atau pencilan dalam data. Boxplot menunjukkan distribusi data melalui kuartil, dengan garis tengah (mediana), kotak yang mewakili rentang interkuartil, dan garis keluar (whiskers) yang menunjukkan rentang data yang dianggap normal. Titik-titik di luar whiskers dianggap sebagai outlier. Setiap boxplot membantu untuk memvisualisasikan sebaran data dan mengidentifikasi apakah ada nilai yang tidak biasa dalam setiap fitur numerik.



```python
# Preprocessing
X = df.drop(columns=['Diabetes_binary','Fruits','Veggies','PhysActivity'])  # Features
y = df['Diabetes_binary']  # Target

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y)

# Convert data to tensors
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
X_train_tensor = torch.tensor(X_train, dtype=torch.float32).to(device)
y_train_tensor = torch.tensor(y_train.values, dtype=torch.long).to(device)
X_test_tensor = torch.tensor(X_test, dtype=torch.float32).to(device)
y_test_tensor = torch.tensor(y_test.values, dtype=torch.long).to(device)
```

0.0s Python

```python
# Define the MLP Classification model
class MLPClassification(nn.Module):
    def __init__(self, input_size, hidden_layers, activation):
        super(MLPClassification, self).__init__()
        layers = []
        for neurons in hidden_layers:
            layers.append(nn.Linear(input_size, neurons))
            if activation == 'relu':
                layers.append(nn.ReLU())
            elif activation == 'sigmoid':
                layers.append(nn.Sigmoid())
            elif activation == 'tanh':
                layers.append(nn.Tanh())
            elif activation == 'softmax':
                layers.append(nn.Softmax(dim=1))
            elif activation == 'linear':
                pass
            input_size = neurons
        layers.append(nn.Linear(input_size, 2))  # Output layer for binary classification
        self.model = nn.Sequential(*layers)

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

0.0s Python

Analisis:

Kode ini digunakan mempersiapkan data untuk membangun model klasifikasi menggunakan Multilayer Perceptron (MLP) di PyTorch. Pertama, data dibagi menjadi fitur

(X) dan target (y), dengan kolom yang tidak relevan seperti Fruits, Veggies, dan PhysActivity dihapus, dan kolom Diabetes_binary dijadikan target untuk klasifikasi. Fitur kemudian dinormalisasi menggunakan StandardScaler untuk memastikan semua fitur memiliki skala yang sama. Data kemudian dibagi menjadi set pelatihan dan pengujian, dengan 80% untuk pelatihan dan 20% untuk pengujian, menggunakan teknik stratified sampling untuk menjaga keseimbangan kelas. Setelah itu, data diubah menjadi tensor PyTorch agar dapat diproses di GPU atau CPU. Model MLP dibangun dengan beberapa lapisan tersembunyi, di mana setiap lapisan diikuti oleh fungsi aktivasi (seperti ReLU, Sigmoid, atau Tanh), dan lapisan output terdiri dari dua neuron untuk klasifikasi biner. Model ini kemudian akan digunakan untuk melatih dan memprediksi kelas target berdasarkan data yang diberikan.

```python
# 1 Hidden Layer
hidden_layers_options_1 = [[4], [16], [32], [64]]
activation_options = ["linear","softmax","relu", "sigmoid", "tanh"]
epoch_options = [1, 10, 25, 50, 100, 250]
learning_rate_options = [10, 1, 0.1, 0.01, 0.001, 0.0001]
batch_size_options = [16, 32, 64, 128, 256, 512]

results_hl1 = []
for hidden_layers in hidden_layers_options_1:
    for activation in activation_options:
        for epochs in epoch_options:
            for lr in learning_rate_options:
                for batch_size in batch_size_options:
                    # Initialize model
                    model = MLPClassification(input_size=X_train_tensor.shape[1],
                                              hidden_layers=hidden_layers,
                                              activation=activation).to(device)

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

                    # Training loop
                    total_loss = 0
                    for epoch in range(epochs):
                        model.train()
                        optimizer.zero_grad()
                        outputs = model(X_train_tensor)
                        loss = criterion(outputs, y_train_tensor)
                        loss.backward()
                        optimizer.step()
                        total_loss += loss.item()  # Sum up loss for all epochs

                    # Evaluate on test set
                    model.eval()
                    with torch.no_grad():
                        y_pred_tensor = model(X_test_tensor)
                        y_pred_prob = torch.softmax(y_pred_tensor, axis=1).cpu().numpy()[:, 1]
                        y_pred = torch.argmax(y_pred_tensor, axis=1).cpu().numpy()
                        accuracy = accuracy_score(y_test, y_pred)

                    # Store results
                    results_hl1.append({
                        'hidden_layers': hidden_layers,
                        'activation': activation,
                        'epochs': epochs,
                        'learning_rate': lr,
                        'batch_size': batch_size,
                        'accuracy': accuracy,
                        'total_loss': total_loss  # Store total loss
                    })
                    print("=======================================")
                    print(f"HL: {hidden_layers}, Act: {activation}, Epochs: {epochs}, LR: {lr}, BS: {batch_size}, Accuracy: {accuracy:.4f}, Loss: {total_loss:.4f}")
                    print("=======================================")

data_hl1 = pd.DataFrame(results_hl1)
data_hl1.to_csv("mlp_experiment_classification_hl1.csv", mode='w', header=True, index=False)
```

✓ 76m 1.0s

Output:

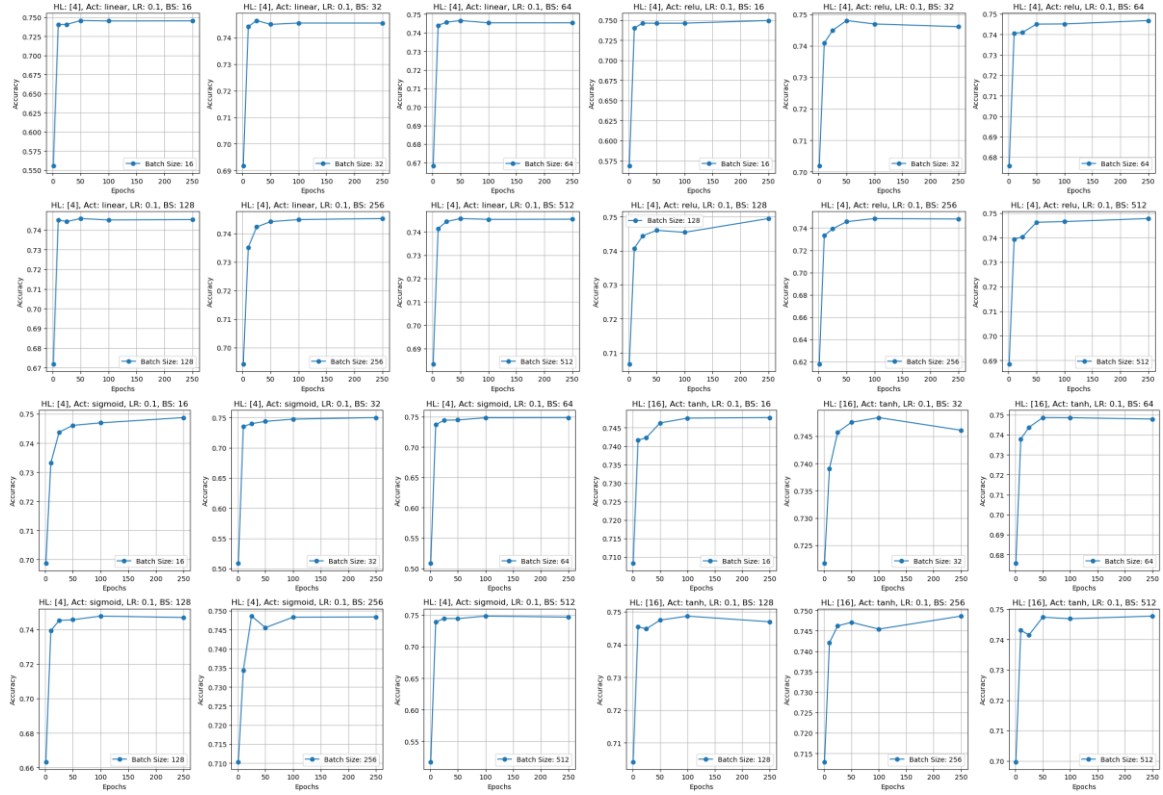
```
==========================================
HL: [4], Act: linear, Epochs: 1, LR: 10, BS: 16, Accuracy: 0.7165, Loss: 0.6939
==========================================
==========================================
HL: [4], Act: linear, Epochs: 1, LR: 10, BS: 32, Accuracy: 0.6807, Loss: 0.6982
==========================================
==========================================
HL: [4], Act: linear, Epochs: 1, LR: 10, BS: 64, Accuracy: 0.7101, Loss: 0.6711
==========================================
==========================================
HL: [4], Act: linear, Epochs: 1, LR: 10, BS: 128, Accuracy: 0.7044, Loss: 0.7372
==========================================
==========================================
HL: [4], Act: linear, Epochs: 1, LR: 10, BS: 256, Accuracy: 0.6733, Loss: 0.6701
==========================================
==========================================
HL: [4], Act: linear, Epochs: 1, LR: 10, BS: 512, Accuracy: 0.6754, Loss: 0.6963
==========================================
==========================================
HL: [4], Act: linear, Epochs: 1, LR: 1, BS: 16, Accuracy: 0.7231, Loss: 0.7312
==========================================
==========================================
HL: [4], Act: linear, Epochs: 1, LR: 1, BS: 32, Accuracy: 0.3168, Loss: 0.7244
==========================================
==========================================
...
==========================================
==========================================
HL: [64], Act: tanh, Epochs: 250, LR: 0.0001, BS: 512, Accuracy: 0.7252, Loss: 150.0839
==========================================
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Analisis:

Kode ini melakukan eksperimen untuk melatih model Multilayer Perceptron (MLP) dengan berbagai kombinasi parameter. Beberapa variasi yang diuji meliputi jumlah neuron di lapisan tersembunyi (misalnya, 4, 16, 32, 64), jenis fungsi aktivasi (seperti ReLU, Sigmoid, Softmax, dll.), jumlah epoch (dari 1 hingga 250), laju pembelajaran (dari 10 hingga 0.0001), dan ukuran batch (16 hingga 512). Untuk setiap kombinasi parameter, model dilatih menggunakan CrossEntropyLoss dan dioptimalkan dengan algoritma Adam. Setelah pelatihan, akurasi pada set pengujian dihitung, dan hasilnya disimpan bersama dengan total loss. Hasil eksperimen ini kemudian disimpan dalam file CSV yang mencatat setiap konfigurasi parameter yang diuji beserta akurasi dan loss-nya. Proses ini membantu dalam mencari konfigurasi terbaik untuk model dalam memprediksi target berdasarkan data yang diberikan.

```python
import pandas as pd
import matplotlib.pyplot as plt
import os

# Load the results dataset
results_df_1 = pd.read_csv("mlp_experiment_classification_hl1.csv")

# Loop melalui semua kombinasi Hidden Layers, Activation, dan Learning Rate
hidden_layers_list = results_df_1['hidden_layers'].unique()
activations_list = results_df_1['activation'].unique()
learning_rates = results_df_1['learning_rate'].unique()

for hidden_layers in hidden_layers_list:
    for activation in activations_list:
        for lr in learning_rates:
            # Filter data untuk kombinasi parameter ini
            filtered_data = results_df_1[
                (results_df_1['hidden_layers'] == str(hidden_layers)) &
                (results_df_1['activation'] == activation) &
                (results_df_1['learning_rate'] == lr)
            ]

            if not filtered_data.empty:
                # Plot akurasi terhadap epochs untuk setiap batch size
                batch_sizes = filtered_data['batch_size'].unique()

                plt.figure(figsize=(12, 12))
                for i, batch_size in enumerate(batch_sizes, 1):
                    plt.subplot(3, 3, i)  # Buat grid 3x3
                    data = filtered_data[filtered_data['batch_size'] == batch_size]
                    plt.plot(data['epochs'], data['accuracy'], marker='o', label=f'Batch Size: {batch_size}')
                    plt.title(f'HL: {hidden_layers}, Act: {activation}, LR: {lr}, BS: {batch_size}')
                    plt.xlabel('Epochs')
                    plt.ylabel('Accuracy')
                    plt.legend()
                    plt.grid()

                plt.tight_layout()
                plt.show()
```

Output:



Analisis:

Kode ini digunakan untuk menganalisis hasil eksperimen model MLP yang telah dilatih sebelumnya. Pertama, dataset hasil eksperimen dimuat, kemudian kode ini memeriksa setiap kombinasi dari parameter yang diuji: jumlah lapisan tersembunyi (hidden layers), fungsi aktivasi, dan laju pembelajaran (learning rate). Untuk setiap kombinasi, data difilter dan akurasi diuji untuk berbagai ukuran batch (batch size). Setelah itu, kode ini menghasilkan grafik yang menunjukkan perubahan akurasi sepanjang epoch untuk setiap ukuran batch yang berbeda, dengan tujuan untuk membandingkan bagaimana masing-

masing parameter mempengaruhi kinerja model. Setiap plot berisi informasi tentang kombinasi parameter tersebut, dengan grafik untuk masing-masing ukuran batch dalam grid 3x3.

```python
# 2 Hidden Layer
hidden_layers_options_2 = [[4,16], [16,32], [32,64], [64,128]]
activation_options = ["linear","softmax","relu", "sigmoid", "tanh"]
epoch_options = [1, 10, 25, 50, 100, 250]
learning_rate_options = [10, 1, 0.1, 0.01, 0.001, 0.0001]
batch_size_options = [16, 32, 64, 128, 256, 512]

results_hl2 = []
for hidden_layers in hidden_layers_options_2:
    for activation in activation_options:
        for epochs in epoch_options:
            for lr in learning_rate_options:
                for batch_size in batch_size_options:
                    # Initialize model
                    model = MLPClassification(input_size=X_train_tensor.shape[1],
                                              hidden_layers=hidden_layers,
                                              activation=activation).to(device)

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

                    # Training loop
                    total_loss = 0
                    for epoch in range(epochs):
                        model.train()
                        optimizer.zero_grad()
                        outputs = model(X_train_tensor)
                        loss = criterion(outputs, y_train_tensor)
                        loss.backward()
                        optimizer.step()
                        total_loss += loss.item()  # Sum up loss for all epochs

                    # Evaluate on test set
                    model.eval()
                    with torch.no_grad():
                        y_pred_tensor = model(X_test_tensor)
                        y_pred_prob = torch.softmax(y_pred_tensor, axis=1).cpu().numpy()[:, 1]
                        y_pred = torch.argmax(y_pred_tensor, axis=1).cpu().numpy()
                        accuracy = accuracy_score(y_test, y_pred)

                    # Store results
                    results_hl2.append({
                        'hidden_layers': hidden_layers,
                        'activation': activation,
                        'epochs': epochs,
                        'learning_rate': lr,
                        'batch_size': batch_size,
                        'accuracy': accuracy,
                        'total_loss': total_loss  # Store total loss
                    })
                    print(f"HL: {hidden_layers}, Act: {activation}, Epochs: {epochs}, LR: {lr}, BS: {batch_size}, Accuracy: {accuracy:.4f}, Loss: {total_loss:.4f}")
                    print("=============================================")

data_hl2 = pd.DataFrame(results_hl2)
data_hl2.to_csv("mlp_experiment_classification_hl2.csv", mode='w', header=True, index=False)
```

Output:

```
HL: [4, 16], Act: linear, Epochs: 1, LR: 10, BS: 16, Accuracy: 0.7124, Loss: 0.7356
=============================================
HL: [4, 16], Act: linear, Epochs: 1, LR: 10, BS: 32, Accuracy: 0.7018, Loss: 0.6867
=============================================
HL: [4, 16], Act: linear, Epochs: 1, LR: 10, BS: 64, Accuracy: 0.2878, Loss: 0.7252
=============================================
HL: [4, 16], Act: linear, Epochs: 1, LR: 10, BS: 128, Accuracy: 0.7103, Loss: 0.7014
=============================================
HL: [4, 16], Act: linear, Epochs: 1, LR: 10, BS: 256, Accuracy: 0.2832, Loss: 0.6837
=============================================
HL: [4, 16], Act: linear, Epochs: 1, LR: 10, BS: 512, Accuracy: 0.2878, Loss: 0.7371
=============================================
HL: [4, 16], Act: linear, Epochs: 1, LR: 1, BS: 16, Accuracy: 0.2818, Loss: 0.7070
=============================================
HL: [4, 16], Act: linear, Epochs: 1, LR: 1, BS: 32, Accuracy: 0.7123, Loss: 0.6643
=============================================
HL: [4, 16], Act: linear, Epochs: 1, LR: 1, BS: 64, Accuracy: 0.6926, Loss: 0.7006
=============================================
HL: [4, 16], Act: linear, Epochs: 1, LR: 1, BS: 128, Accuracy: 0.7070, Loss: 0.8333
=============================================
HL: [4, 16], Act: linear, Epochs: 1, LR: 1, BS: 256, Accuracy: 0.6959, Loss: 0.6756
=============================================
HL: [4, 16], Act: linear, Epochs: 1, LR: 1, BS: 512, Accuracy: 0.7066, Loss: 0.7456
=============================================
HL: [4, 16], Act: linear, Epochs: 1, LR: 0.1, BS: 16, Accuracy: 0.6928, Loss: 0.6953
...
HL: [64, 128], Act: tanh, Epochs: 250, LR: 0.0001, BS: 256, Accuracy: 0.7458, Loss: 140.0560
=============================================
HL: [64, 128], Act: tanh, Epochs: 250, LR: 0.0001, BS: 512, Accuracy: 0.7446, Loss: 138.4758
=============================================
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

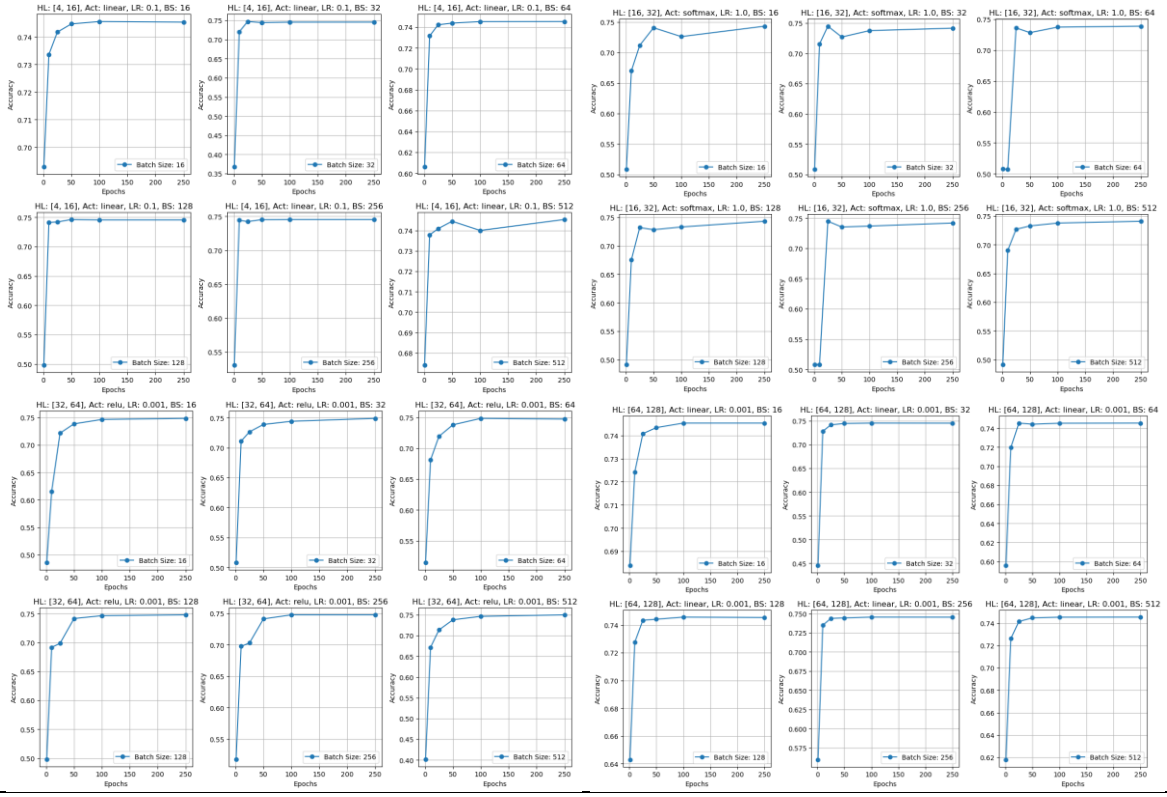
Analisis:

Kode ini melakukan eksperimen untuk melatih model Multilayer Perceptron (MLP) dengan berbagai kombinasi parameter. Beberapa variasi yang diuji meliputi jumlah neuron di lapisan tersembunyi (misalnya (4, 16), (32, 64), dst) jenis fungsi aktivasi (seperti ReLU, Sigmoid, Softmax, dll.), jumlah epoch (dari 1 hingga 250), laju pembelajaran (dari 10 hingga 0.0001), dan ukuran batch (16 hingga 512). Untuk setiap kombinasi parameter, model dilatih menggunakan CrossEntropyLoss dan dioptimalkan dengan algoritma Adam. Setelah pelatihan, akurasi pada set pengujian dihitung, dan hasilnya disimpan bersama dengan total

loss. Hasil eksperimen ini kemudian disimpan dalam file CSV yang mencatat setiap konfigurasi parameter yang diuji beserta akurasi dan loss-nya. Proses ini membantu dalam mencari konfigurasi terbaik untuk model dalam memprediksi target berdasarkan data yang diberikan.

```python
import pandas as pd
import matplotlib.pyplot as plt
import os

# Load the results dataset
results_df_2 = pd.read_csv("mlp_experiment_classification_hl2.csv")

# Loop melalui semua kombinasi Hidden Layers, Activation, dan Learning Rate
hidden_layers_list = results_df_2['hidden_layers'].unique()
activations_list = results_df_2['activation'].unique()
learning_rates = results_df_2['learning_rate'].unique()

for hidden_layers in hidden_layers_list:
    for activation in activations_list:
        for lr in learning_rates:
            # Filter data untuk kombinasi parameter ini
            filtered_data = results_df_2[
                (results_df_2['hidden_layers'] == str(hidden_layers)) &
                (results_df_2['activation'] == activation) &
                (results_df_2['learning_rate'] == lr)
            ]

            if not filtered_data.empty:
                # Plot akurasi terhadap epochs untuk setiap batch size
                batch_sizes = filtered_data['batch_size'].unique()

                plt.figure(figsize=(12, 12))
                for i, batch_size in enumerate(batch_sizes, 1):
                    plt.subplot(3, 3, i)  # Buat grid 3x3
                    data = filtered_data[filtered_data['batch_size'] == batch_size]
                    plt.plot(data['epochs'], data['accuracy'], marker='o', label=f'Batch Size: {batch_size}')
                    plt.title(f'HL: {hidden_layers}, Act: {activation}, LR: {lr}, BS: {batch_size}')
                    plt.xlabel('Epochs')
                    plt.ylabel('Accuracy')
                    plt.legend()
                    plt.grid()

                plt.tight_layout()
                plt.show()
```

Output:



Analisis:

Kode ini digunakan untuk menganalisis hasil eksperimen model MLP yang telah dilatih sebelumnya. Pertama, dataset hasil eksperimen dimuat, kemudian kode ini memeriksa setiap kombinasi dari parameter yang diuji: jumlah lapisan tersembunyi (hidden layers), fungsi aktivasi, dan laju pembelajaran (learning rate). Untuk setiap kombinasi, data difilter dan akurasi diuji untuk berbagai ukuran batch (batch size). Setelah itu, kode ini menghasilkan grafik yang menunjukkan perubahan akurasi sepanjang epoch untuk setiap ukuran batch yang berbeda, dengan tujuan untuk membandingkan bagaimana masing-masing parameter mempengaruhi kinerja model. Setiap plot berisi informasi tentang kombinasi parameter tersebut, dengan grafik untuk masing-masing ukuran batch dalam grid 3x3.

```
# 3 Hidden Layer
hidden_layers_options_3 = [[4,16,32], [16,32,64], [32,64,128], [64,128,256]]
activation_options = ["linear","softmax","relu", "sigmoid", "tanh"]
epoch_options = [1, 10, 25, 50, 100, 250]
learning_rate_options = [10, 1, 0.1, 0.01, 0.001, 0.0001]
batch_size_options = [16, 32, 64, 128, 256, 512]

results_hl3 = []
for hidden_layers in hidden_layers_options_3:
    for activation in activation_options:
        for epochs in epoch_options:
            for lr in learning_rate_options:
                for batch_size in batch_size_options:
                    # Initialize model
                    model = MLPClassification(input_size=X_train_tensor.shape[1],
                                              hidden_layers=hidden_layers,
                                              activation=activation).to(device)

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

                    # Training loop
                    total_loss = 0
                    for epoch in range(epochs):
                        model.train()
                        optimizer.zero_grad()
                        outputs = model(X_train_tensor)
                        loss = criterion(outputs, y_train_tensor)
                        loss.backward()
                        optimizer.step()
                        total_loss += loss.item()  # Sum up loss for all epochs

                    # Evaluate on test set
                    model.eval()
                    with torch.no_grad():
                        y_pred_tensor = model(X_test_tensor)
                        y_pred_prob = torch.softmax(y_pred_tensor, axis=1).cpu().numpy()[:, 1]
                        y_pred = torch.argmax(y_pred_tensor, axis=1).cpu().numpy()
                        accuracy = accuracy_score(y_test, y_pred)

                    # Store results
                    results_hl3.append({
                        'hidden_layers': hidden_layers,
                        'activation': activation,
                        'epochs': epochs,
                        'learning_rate': lr,
                        'batch_size': batch_size,
                        'accuracy': accuracy,
                        'total_loss': total_loss  # Store total loss
                    })
                    print(f"HL: {hidden_layers}, Act: {activation}, Epochs: {epochs}, LR: {lr}, BS: {batch_size}, Accuracy: {accuracy:.4f}, Loss: {total_loss:.4f}")
                    print("=========================================")

data_hl3 = pd.DataFrame(results_hl3)
data_hl3.to_csv("mlp_experiment_classification_hl3.csv", mode='w', header=True, index=False)
```

Output:

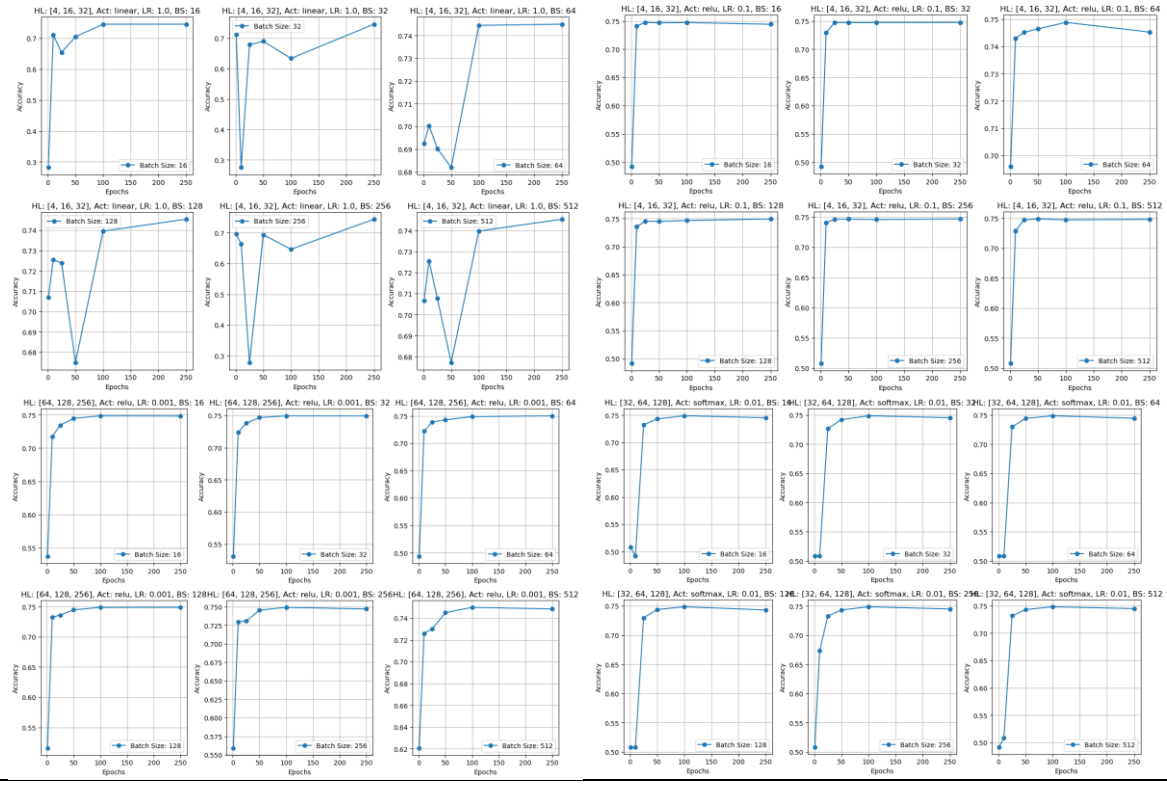
```
HL: [4, 16, 32], Act: linear, Epochs: 1, LR: 10, BS: 16, Accuracy: 0.2877, Loss: 0.6773
==========
HL: [4, 16, 32], Act: linear, Epochs: 1, LR: 10, BS: 32, Accuracy: 0.3172, Loss: 0.6799
==========
HL: [4, 16, 32], Act: linear, Epochs: 1, LR: 10, BS: 64, Accuracy: 0.3052, Loss: 0.7021
==========
HL: [4, 16, 32], Act: linear, Epochs: 1, LR: 10, BS: 128, Accuracy: 0.3226, Loss: 0.7718
==========
HL: [4, 16, 32], Act: linear, Epochs: 1, LR: 10, BS: 256, Accuracy: 0.4124, Loss: 0.7061
==========
HL: [4, 16, 32], Act: linear, Epochs: 1, LR: 10, BS: 512, Accuracy: 0.3051, Loss: 0.7398
==========
HL: [4, 16, 32], Act: linear, Epochs: 1, LR: 1, BS: 16, Accuracy: 0.6976, Loss: 0.6976
==========
HL: [4, 16, 32], Act: linear, Epochs: 1, LR: 1, BS: 32, Accuracy: 0.6968, Loss: 0.7327
==========
HL: [4, 16, 32], Act: linear, Epochs: 1, LR: 1, BS: 64, Accuracy: 0.3043, Loss: 0.7048
==========
HL: [4, 16, 32], Act: linear, Epochs: 1, LR: 1, BS: 128, Accuracy: 0.7103, Loss: 0.7021
==========
HL: [4, 16, 32], Act: linear, Epochs: 1, LR: 1, BS: 256, Accuracy: 0.7218, Loss: 0.6852
==========
HL: [4, 16, 32], Act: linear, Epochs: 1, LR: 1, BS: 512, Accuracy: 0.7152, Loss: 0.6849
==========
HL: [4, 16, 32], Act: linear, Epochs: 1, LR: 0.1, BS: 16, Accuracy: 0.3682, Loss: 0.7046
...
HL: [64, 128, 256], Act: tanh, Epochs: 250, LR: 0.0001, BS: 256, Accuracy: 0.7458, Loss: 135.5201
==========
HL: [64, 128, 256], Act: tanh, Epochs: 250, LR: 0.0001, BS: 512, Accuracy: 0.7451, Loss: 134.4791
==========
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Analisis:

Kode ini melakukan eksperimen untuk melatih model Multilayer Perceptron (MLP) dengan berbagai kombinasi parameter. Beberapa variasi yang diuji meliputi jumlah neuron di lapisan tersembunyi (misalnya (4, 16, 32), (32, 64, 128), dst) jenis fungsi aktivasi (seperti ReLU, Sigmoid, Softmax, dll.), jumlah epoch (dari 1 hingga 250), laju pembelajaran (dari 10 hingga 0.0001), dan ukuran batch (16 hingga 512). Untuk setiap kombinasi parameter, model dilatih menggunakan CrossEntropyLoss dan dioptimalkan dengan algoritma Adam. Setelah pelatihan, akurasi pada set pengujian dihitung, dan hasilnya disimpan bersama dengan total loss. Hasil eksperimen ini kemudian disimpan dalam file CSV yang mencatat setiap konfigurasi parameter yang diuji beserta akurasi dan loss-nya. Proses ini membantu dalam mencari konfigurasi terbaik untuk model dalam memprediksi target berdasarkan data yang diberikan.

```python
import pandas as pd
import matplotlib.pyplot as plt
import os

# Load the results dataset
results_df_3 = pd.read_csv("mlp_experiment_classification_hl3.csv")

# Loop melalui semua kombinasi Hidden Layers, Activation, dan Learning Rate
hidden_layers_list = results_df_3['hidden_layers'].unique()
activations_list = results_df_3['activation'].unique()
learning_rates = results_df_3['learning_rate'].unique()

for hidden_layers in hidden_layers_list:
    for activation in activations_list:
        for lr in learning_rates:
            # Filter data untuk kombinasi parameter ini
            filtered_data = results_df_2[
                (results_df_3['hidden_layers'] == str(hidden_layers)) &
                (results_df_3['activation'] == activation) &
                (results_df_3['learning_rate'] == lr)
            ]

            if not filtered_data.empty:
                # Plot akurasi terhadap epochs untuk setiap batch size
                batch_sizes = filtered_data['batch_size'].unique()

                plt.figure(figsize=(12, 12))
                for i, batch_size in enumerate(batch_sizes, 1):
                    plt.subplot(3, 3, i)  # Buat grid 3x3
                    data = filtered_data[filtered_data['batch_size'] == batch_size]
                    plt.plot(data['epochs'], data['accuracy'], marker='o', label=f'Batch Size: {batch_size}')
                    plt.title(f'HL: {hidden_layers}, Act: {activation}, LR: {lr}, BS: {batch_size}')
                    plt.xlabel('Epochs')
                    plt.ylabel('Accuracy')
                    plt.legend()
                    plt.grid()

                plt.tight_layout()
                plt.show()
```

Output:



Analisis:

Kode ini digunakan untuk menganalisis hasil eksperimen model MLP yang telah dilatih sebelumnya. Pertama, dataset hasil eksperimen dimuat, kemudian kode ini memeriksa setiap kombinasi dari parameter yang diuji: jumlah lapisan tersembunyi (hidden layers), fungsi aktivasi, dan laju pembelajaran (learning rate). Untuk setiap kombinasi, data difilter dan akurasi diuji untuk berbagai ukuran batch (batch size). Setelah itu, kode ini

menghasilkan grafik yang menunjukkan perubahan akurasi sepanjang epoch untuk setiap ukuran batch yang berbeda, dengan tujuan untuk membandingkan bagaimana masing-masing parameter mempengaruhi kinerja model. Setiap plot berisi informasi tentang kombinasi parameter tersebut, dengan grafik untuk masing-masing ukuran batch dalam grid 3x3.

```python
import pandas as pd

# Load result datasets
results_df_1 = pd.read_csv("mlp_experiment_classification_hl1.csv")
results_df_2 = pd.read_csv("mlp_experiment_classification_hl2.csv")
results_df_3 = pd.read_csv("mlp_experiment_classification_hl3.csv")

# Sort and select top 10 results for each hidden layer configuration
top_10_hl1 = results_df_1.sort_values(by='accuracy', ascending=False).head(10)
top_10_hl2 = results_df_2.sort_values(by='accuracy', ascending=False).head(10)
top_10_hl3 = results_df_3.sort_values(by='accuracy', ascending=False).head(10)

# Display top 10 results for each hidden layer in tabular format
print("Top 10 Results for Hidden Layer 1:")
display(top_10_hl1[['hidden_layers', 'activation', 'epochs', 'learning_rate', 'batch_size', 'accuracy', 'total_loss']])

print("\nTop 10 Results for Hidden Layer 2:")
display(top_10_hl2[['hidden_layers', 'activation', 'epochs', 'learning_rate', 'batch_size', 'accuracy', 'total_loss']])

print("\nTop 10 Results for Hidden Layer 3:")
display(top_10_hl3[['hidden_layers', 'activation', 'epochs', 'learning_rate', 'batch_size', 'accuracy', 'total_loss']])
```

✓ 0.0s

Output:

	hidden_layers	activation	epochs	learning_rate	batch_size	accuracy	total_loss
1635	[16]	relu	50	0.10	128	0.751738	26.254702
2755	[32]	relu	100	0.01	32	0.751738	52.018155
1677	[16]	relu	100	0.01	128	0.751376	52.194516
1668	[16]	relu	100	0.10	16	0.751376	51.358639
1421	[16]	softmax	50	0.10	512	0.751231	26.707046
2713	[32]	relu	50	0.10	32	0.751158	26.324679
1715	[16]	relu	250	0.01	512	0.751086	128.328839
3796	[64]	relu	50	0.10	256	0.751014	27.452455
1461	[16]	softmax	100	0.01	128	0.750941	56.962629
3794	[64]	relu	50	0.10	64	0.750869	30.256176

Top 10 Results for Hidden Layer 2:

	hidden_layers	activation	epochs	learning_rate	batch_size	accuracy	total_loss
1024	[4, 16]	tanh	100	0.10	256	0.752172	52.450121
1063	[4, 16]	tanh	250	0.01	32	0.751810	129.092148
1061	[4, 16]	tanh	250	0.10	512	0.751810	128.523728
2146	[16, 32]	tanh	250	0.01	256	0.751738	127.762700
1670	[16, 32]	relu	100	0.10	64	0.751303	52.355460
2755	[32, 64]	relu	100	0.01	32	0.750941	51.484025
1668	[16, 32]	relu	100	0.10	16	0.750869	51.966917
1678	[16, 32]	relu	100	0.01	256	0.750724	52.143462
2757	[32, 64]	relu	100	0.01	128	0.750724	51.350396
1679	[16, 32]	relu	100	0.01	512	0.750507	52.276919

Top 10 Results for Hidden Layer 3:

	hidden_layers	activation	epochs	learning_rate	batch_size	accuracy	total_loss
848	[4, 16, 32]	sigmoid	250	0.010	64	0.751665	134.764617
2109	[16, 32, 64]	tanh	100	0.010	128	0.751520	51.951709
3841	[64, 128, 256]	relu	100	0.001	32	0.751086	52.876278
2722	[32, 64, 128]	relu	50	0.010	256	0.751014	26.127391
841	[4, 16, 32]	sigmoid	250	0.100	32	0.750869	140.707145
1920	[16, 32, 64]	sigmoid	250	0.100	16	0.750724	140.760035
631	[4, 16, 32]	relu	250	0.010	32	0.750652	129.480720
3843	[64, 128, 256]	relu	100	0.001	128	0.750362	52.776987
1642	[16, 32, 64]	relu	50	0.010	256	0.750362	26.675895
4310	[64, 128, 256]	tanh	250	0.001	64	0.750290	128.475791

```python
import pandas as pd

# Load result datasets
results_df_1 = pd.read_csv("mlp_experiment_classification_hl1.csv")
results_df_2 = pd.read_csv("mlp_experiment_classification_hl2.csv")
results_df_3 = pd.read_csv("mlp_experiment_classification_hl3.csv")

# Sort and select top 10 results for each hidden layer configuration
top_10_hl1 = results_df_1.sort_values(by='accuracy', ascending=False).head(10)
top_10_hl2 = results_df_2.sort_values(by='accuracy', ascending=False).head(10)
top_10_hl3 = results_df_3.sort_values(by='accuracy', ascending=False).head(10)

# Add a column to indicate which Hidden Layer experiment the data is from
top_10_hl1['hidden_layer_experiment'] = 'HL1'
top_10_hl2['hidden_layer_experiment'] = 'HL2'
top_10_hl3['hidden_layer_experiment'] = 'HL3'

# Combine the top 10 results from all three experiments
combined_top_10 = pd.concat([top_10_hl1[['hidden_layers', 'activation', 'epochs', 'learning_rate', 'batch_size', 'accuracy', 'total_loss', 'hidden_layer_experiment']],
                             top_10_hl2[['hidden_layers', 'activation', 'epochs', 'learning_rate', 'batch_size', 'accuracy', 'total_loss', 'hidden_layer_experiment']],
                             top_10_hl3[['hidden_layers', 'activation', 'epochs', 'learning_rate', 'batch_size', 'accuracy', 'total_loss', 'hidden_layer_experiment']]],
                            ignore_index=True)

# Sort by accuracy to compare the top performing models across all experiments
combined_top_10_sorted = combined_top_10.sort_values(by='accuracy', ascending=False)

# Display the top 10 sorted table for better comparison
top_10_best = combined_top_10_sorted.head(10)

# Display the top 10 best results
top_10_best[['hidden_layers', 'activation', 'epochs', 'learning_rate', 'batch_size', 'accuracy', 'total_loss', 'hidden_layer_experiment']]
```

✓ 0.0s

Output:

	hidden_layers	activation	epochs	learning_rate	batch_size	accuracy	total_loss	hidden_layer_experiment
10	[4, 16]	tanh	100	0.10	256	0.752172	52.450121	HL2
12	[4, 16]	tanh	250	0.10	512	0.751810	128.523728	HL2
11	[4, 16]	tanh	250	0.01	32	0.751810	129.092148	HL2
0	[16]	relu	50	0.10	128	0.751738	26.254702	HL1
1	[32]	relu	100	0.01	32	0.751738	52.018155	HL1
13	[16, 32]	tanh	250	0.01	256	0.751738	127.762700	HL2
20	[4, 16, 32]	sigmoid	250	0.01	64	0.751665	134.764617	HL3
21	[16, 32, 64]	tanh	100	0.01	128	0.751520	51.951709	HL3
3	[16]	relu	100	0.10	16	0.751376	51.358639	HL1
2	[16]	relu	100	0.01	128	0.751376	52.194516	HL1