Nama : Ameliani Kusmayadi
NIM : 1103213044
Kelas : TK-45-06

WEEK 10 MLP CLASSIFICATION

Import Library yang digunakan

```
[ ]  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 accuracy_score, classification_report
     import numpy as np
```

Menampilkan 5 baris dataset

```
# Memuat dataset
df = pd.read_csv('ObesityDataSet_raw_and_data_sinthetic.csv')
df.head()
```

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	MTRANS	NObeyesdad
0	Female	21.0	1.62	64.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	0.0	1.0	no	Public_Transportation	Normal_Weight
1	Female	21.0	1.52	56.0	yes	no	3.0	3.0	Sometimes	yes	3.0	yes	3.0	0.0	Sometimes	Public_Transportation	Normal_Weight
2	Male	23.0	1.80	77.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	2.0	1.0	Frequently	Public_Transportation	Normal_Weight
3	Male	27.0	1.80	87.0	no	no	3.0	3.0	Sometimes	no	2.0	no	2.0	0.0	Frequently	Walking	Overweight_Level_I
4	Male	22.0	1.78	89.8	no	no	2.0	1.0	Sometimes	no	2.0	no	0.0	0.0	Sometimes	Public_Transportation	Overweight_Level_II

Menampilkan informasi pada dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2111 entries, 0 to 2110
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Gender                          2111 non-null   object
 1   Age                             2111 non-null   float64
 2   Height                          2111 non-null   float64
 3   Weight                          2111 non-null   float64
 4   family_history_with_overweight  2111 non-null   object
 5   FAVC                            2111 non-null   object
 6   FCVC                            2111 non-null   float64
 7   NCP                             2111 non-null   float64
 8   CAEC                            2111 non-null   object
 9   SMOKE                           2111 non-null   object
 10  CH2O                            2111 non-null   float64
 11  SCC                             2111 non-null   object
 12  FAF                             2111 non-null   float64
 13  TUE                             2111 non-null   float64
 14  CALC                            2111 non-null   object
 15  MTRANS                          2111 non-null   object
 16  NObeyesdad                      2111 non-null   object
dtypes: float64(8), object(9)
memory usage: 280.5+ KB
```

Menampilkan kolom pada dataset

```
[ ]  df.columns
```

```
Index(['Gender', 'Age', 'Height', 'Weight', 'family_history_with_overweight',
       'FAVC', 'FCVC', 'NCP', 'CAEC', 'SMOKE', 'CH2O', 'SCC', 'FAF', 'TUE',
       'CALC', 'MTRANS', 'NObeyesdad'],
      dtype='object')
```

Melakukan encoding pada kolom kategorikal menggunakan LabelEncoder

```
[ ]  # Melakukan encoding pada kolom kategorikal menggunakan LabelEncoder
     label_cols = ['Gender', 'family_history_with_overweight', 'FAVC', 'CAEC', 'SMOKE', 'SCC', 'CALC', 'MTRANS', 'NObeyesdad']
     label_encoder = LabelEncoder()

     for col in label_cols:
         df[col] = label_encoder.fit_transform(df[col])
```

Menentukan fitur target untuk split data training dan data testing

```
[ ]  # Menentukan fitur (X) dan target (y)
     X = df.drop(['NObeyesdad'], axis=1)  # Menghapus kolom target
     y = df['NObeyesdad']  # Kolom target

[ ]  # Split data menjadi training dan testing (80% training, 20% testing)
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

Membuat standard scaler

```
[ ]  # Konversi data ke tensor
     X_train_tensor = torch.tensor(X_train_scaled, dtype=torch.float32)
     y_train_tensor = torch.tensor(y_train.values, dtype=torch.long)  # Untuk klasifi
     X_test_tensor = torch.tensor(X_test_scaled, dtype=torch.float32)
     y_test_tensor = torch.tensor(y_test.values, dtype=torch.long)
```

Grafik di atas menunjukkan hubungan antara hyperparameter yang diuji (jumlah lapisan tersembunyi, jumlah neuron, fungsi aktivasi, jumlah epoch, learning rate, dan ukuran batch) dengan rata-rata Mean Absolute Error (MAE). MAE digunakan untuk mengevaluasi performa model, dimana semakin kecil nilai MAE, semakin baik model dalam memprediksi hasil. Grafik ini memberikan gambaran bahwa kombinasi tertentu dari hyperparameter menghasilkan MAE yang lebih rendah, mengindikasikan model dengan konfigurasi tersebut lebih efektif dalam melakukan prediksi dibandingkan dengan konfigurasi lainnya.

```python
# Menyusun model MLP untuk klasifikasi
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

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

        # Menambahkan 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, len(y.unique())))  # Jumlah kelas ou
        self.model = nn.Sequential(*layers)

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

Kode di atas menggambarkan pengaruh berbagai kombinasi hyperparameter terhadap Mean Squared Error (MSE) dalam model MLP. MSE menunjukkan seberapa jauh prediksi model dari nilai sebenarnya, dengan nilai MSE yang lebih rendah menunjukkan kinerja model yang lebih baik. Dari grafik ini, terlihat bahwa beberapa kombinasi hyperparameter, seperti jumlah lapisan tersembunyi, jumlah neuron, fungsi aktivasi, jumlah epoch, learning rate, dan ukuran batch, menghasilkan nilai MSE yang lebih kecil. Hal ini menunjukkan bahwa pengaturan yang tepat dari hyperparameter sangat mempengaruhi akurasi dan kinerja model dalam memprediksi.

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

```python
[ ] # Hyperparameter yang akan diuji
    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]

    # Menyimpan hasil
    results = []
```

```python
[ ] # Melakukan eksperimen dengan kombinasi hyperparameter
    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:
                            # Membuat dan memindahkan model ke perangkat (GPU atau CPU)
                            model = MLPClassification(input_size=X_train_tensor.shape[1],
                                                      hidden_layers=layers,
                                                      neurons=neuron,
                                                      activation=activation).to(device)

                            # Mendefinisikan loss function dan optimizer
                            criterion = nn.CrossEntropyLoss()  # Untuk klasifikasi multi-kelas
                            optimizer = optim.Adam(model.parameters(), lr=lr)

                            # Training loop
```

Kode di atas bertujuan untuk melakukan eksperimen dengan berbagai kombinasi hyperparameter pada model Multi-Layer Perceptron (MLP) untuk tugas regresi. Hyperparameter yang diuji meliputi jumlah lapisan tersembunyi, jumlah neuron, fungsi aktivasi, jumlah epoch, learning rate, dan ukuran batch. Setiap kombinasi hyperparameter digunakan untuk melatih model, yang kemudian dievaluasi menggunakan metrik seperti Mean Absolute Error (MAE), Mean Squared Error (MSE), R-squared, dan akurasi. Hasil eksperimen, yang mencakup metrik kinerja dan

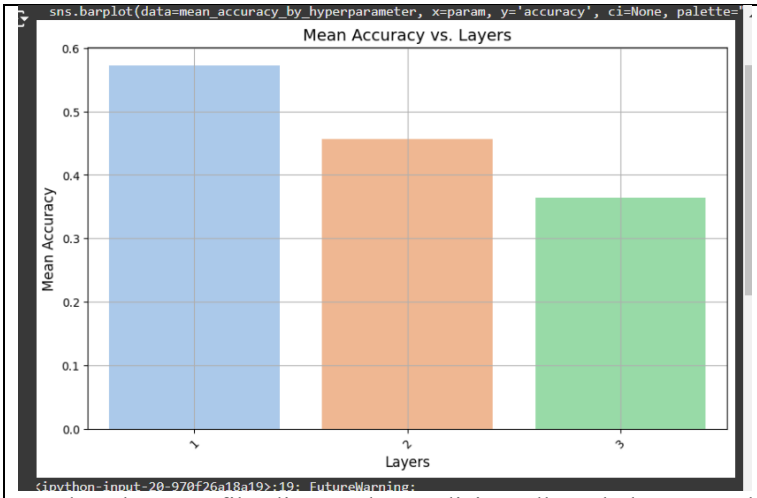
kombinasi hyperparameter, disimpan untuk analisis lebih lanjut, dengan tujuan mencari pengaturan terbaik yang memberikan kinerja model optimal.

```python
# Mengonversi hasil ke DataFrame dan menyimpannya ke CSV
results_df = pd.DataFrame(results)
results_df.to_csv("mlp_classification_hidden layer 123.csv", index=False)
print("All results have been saved to 'mlp_classification_hidden layer 123.csv'.")
```

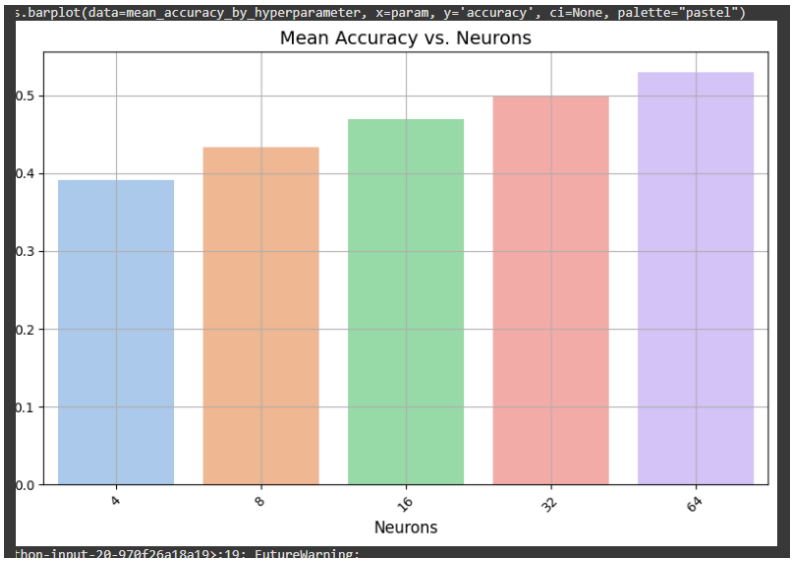
Kode di atas digunakan untuk mengkonversi hasil data frame dan menyimpannya ke file CSV

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

# Mengambil DataFrame hasil eksperimen
# Pastikan 'results_df' sudah ada setelah eksekusi kode sebelumnya
results_df = pd.read_csv("mlp_classification_hidden layer 123.csv")

# Calculate mean accuracy for each hyperparameter combination
# Changed 'mae' to 'accuracy' to calculate and plot accuracy
mean_accuracy_by_hyperparameter = results_df.groupby(['layers', 'neurons', 'activation', 'epo

# Plot mean accuracy vs. each hyperparameter
hyperparameters = ['layers', 'neurons', 'activation', 'epochs', 'lr', 'batch_size']

# Create bar plots for each hyperparameter
for param in hyperparameters:
    plt.figure(figsize=(10, 6))
    # Changed 'mae' to 'accuracy' in sns.barplot and ylabel
    sns.barplot(data=mean_accuracy_by_hyperparameter, x=param, y='accuracy', ci=None, palette=
    plt.title(f'Mean Accuracy vs. {param.capitalize()}', fontsize=14) # Changed title to refl
    plt.xlabel(param.capitalize(), fontsize=12)
    plt.ylabel('Mean Accuracy', fontsize=12) # Changed ylabel to reflect Accuracy
    plt.xticks(rotation=45)
    plt.grid(True)
    plt.show()
```

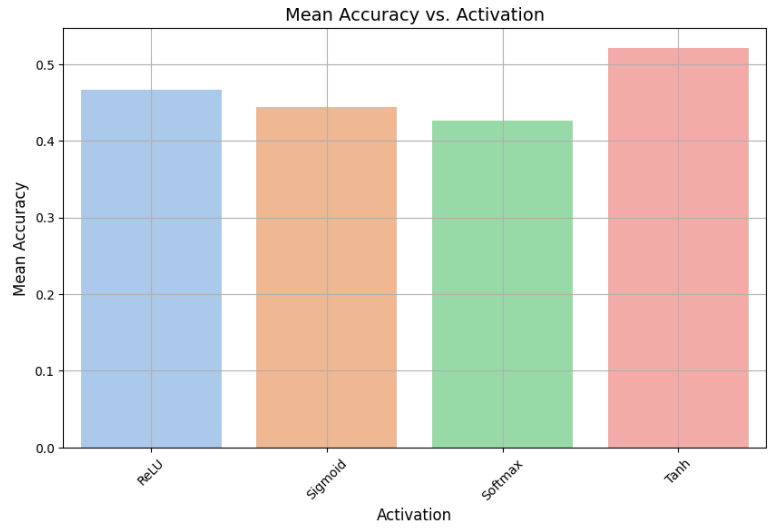
Kode ini digunakan untuk memvisualisasikan rata-rata akurasi model MLP berdasarkan kombinasi hiperparameter seperti layers, neurons, activation, epochs, lr, dan batch_size. Data dibaca dari file CSV, kemudian dikelompokkan dan dihitung rata-rata akurasi untuk setiap kombinasi hiperparameter. Setiap hiperparameter diplot menggunakan grafik batang untuk menunjukkan pengaruhnya terhadap akurasi model. Pengaturan visual tambahan, seperti rotasi label dan grid, digunakan untuk memperjelas tampilan plot, memudahkan analisis pengaruh hiperparameter terhadap kinerja model.



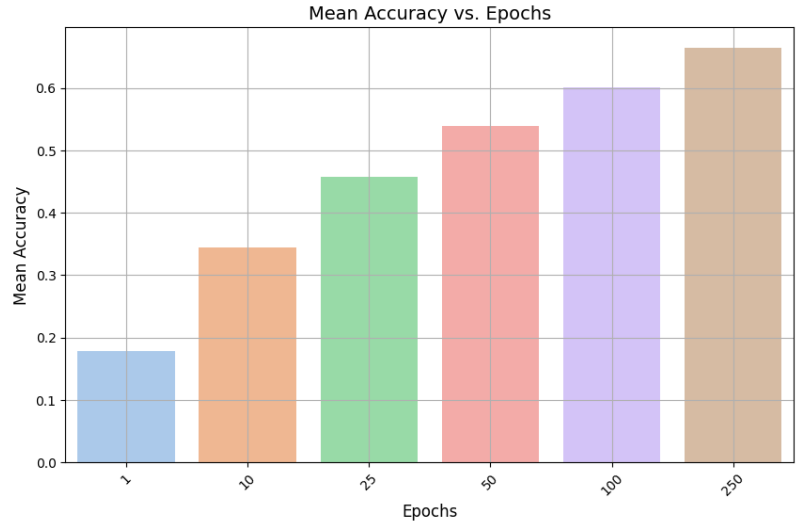
Berdasarkan grafik diatas, dapat disimpulkan bahwa untuk dataset yang digunakan dalam eksperimen ini, model dengan 1 lapisan memberikan akurasi terbaik. Hasil ini menekankan pentingnya melakukan tuning hyperparameter secara hati-hati untuk mendapatkan kinerja model yang optimal.



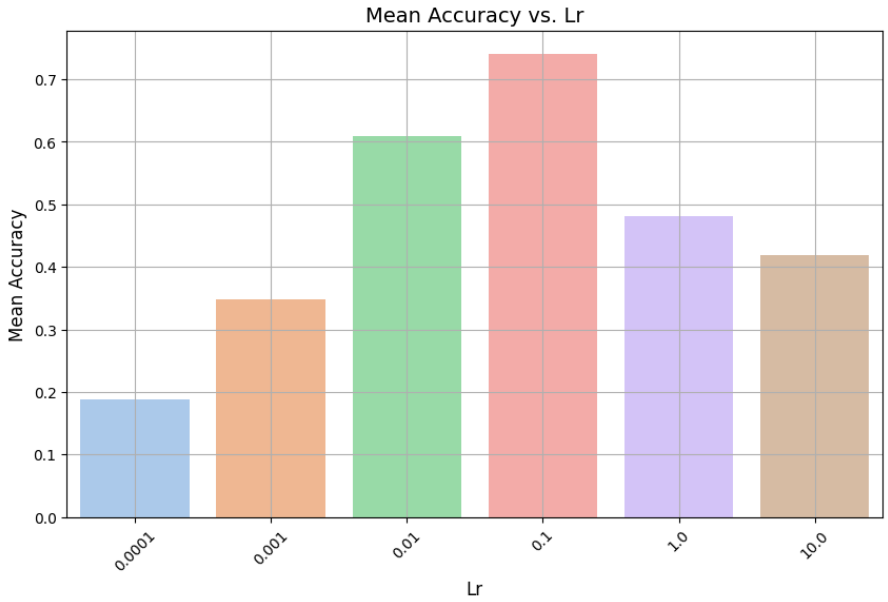
Berdasarkan grafik diatas, dapat disimpulkan bahwa untuk dataset yang digunakan dalam eksperimen ini, model dengan 64 neuron memberikan akurasi terbaik. Hasil ini menekankan pentingnya melakukan tuning hyperparameter secara hati-hati untuk mendapatkan kinerja model yang optimal.



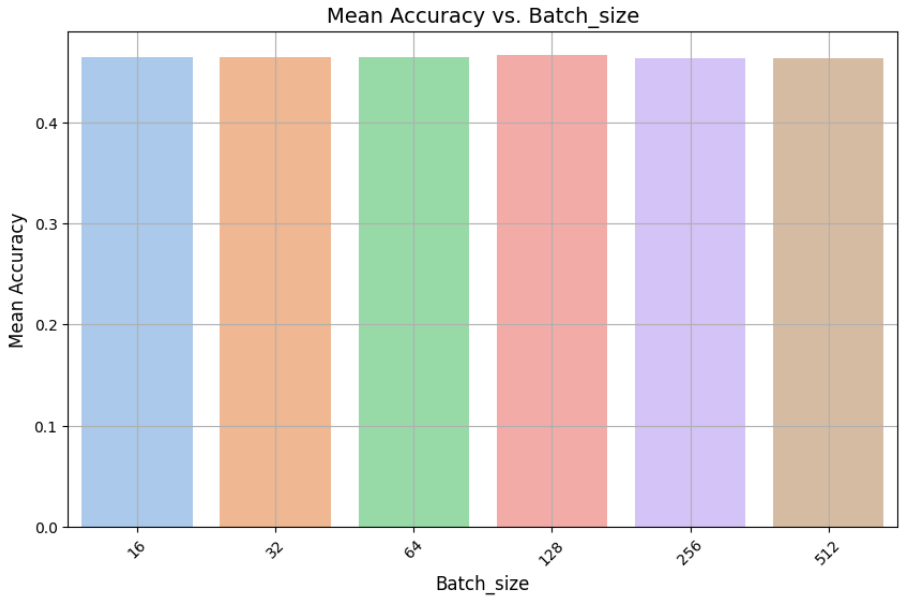
Berdasarkan grafik di atas, dapat disimpulkan bahwa fungsi aktivasi Tanh memberikan akurasi terbaik untuk dataset yang digunakan dalam eksperimen ini. Namun, pemilihan fungsi aktivasi yang tepat harus disesuaikan dengan karakteristik dataset dan masalah yang ingin diselesaikan.



Berdasarkan grafik di atas, dapat disimpulkan bahwa jumlah *epochs* yang optimal untuk model ini adalah sekitar 100. Setelah mencapai jumlah *epochs* tersebut, penambahan *epochs* selanjutnya tidak memberikan peningkatan yang signifikan pada akurasi.



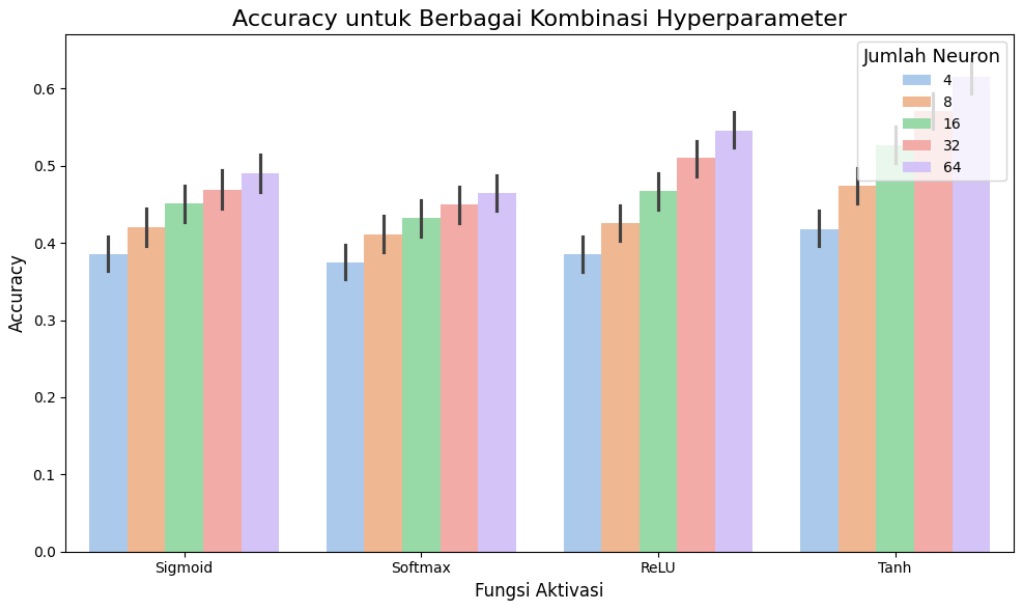
Berdasarkan grafik di atas, dapat disimpulkan bahwa nilai *learning rate* sebesar 0.1 memberikan akurasi terbaik untuk dataset yang digunakan dalam eksperimen ini. Namun, nilai *learning rate* yang optimal dapat berbeda untuk dataset dan model yang berbeda. Oleh karena itu, pemilihan nilai *learning rate* yang tepat perlu dilakukan secara eksperimental.



Berdasarkan grafik di atas, dapat disimpulkan bahwa untuk dataset dan model yang digunakan dalam eksperimen ini, perubahan ukuran *batch* tidak memberikan dampak yang signifikan pada akurasi. Namun, pemilihan ukuran *batch* yang tepat tetap penting untuk mempertimbangkan aspek-aspek seperti kecepatan pelatihan dan penggunaan memori.

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

# Mengambil DataFrame hasil eksperimen
# Pastikan 'results_df' sudah ada setelah eksekusi kode sebelumnya
results_df = pd.read_csv("mlp_classification_hidden layer 123.csv")

# Plot accuracy untuk setiap kombinasi hyperparameter
plt.figure(figsize=(10, 6))

# Mengubah 'mse' menjadi 'accuracy' pada parameter y
sns.barplot(data=results_df, x='activation', y='accuracy', hue='neurons', palette='pastel')

# Menambahkan judul dan label
plt.title('Accuracy untuk Berbagai Kombinasi Hyperparameter', fontsize=16) # Mengubah judul me
plt.xlabel('Fungsi Aktivasi', fontsize=12)
plt.ylabel('Accuracy', fontsize=12) # Mengubah label y menjadi Accuracy
plt.legend(title='Jumlah Neuron', title_fontsize='13', loc='upper right')

# Menampilkan grafik
plt.tight_layout()
plt.show()
```

Kode tersebut bertujuan untuk memvisualisasikan hasil eksperimen pada model jaringan saraf tiruan (JST). Eksperimen ini dilakukan dengan mencoba berbagai kombinasi hyperparameter, yaitu fungsi aktivasi (seperti Sigmoid, Softmax, ReLU, dan Tanh) dan jumlah neuron pada lapisan tersembunyi.



Grafik yang dihasilkan akan menampilkan perbandingan akurasi model untuk berbagai kombinasi fungsi aktivasi (Sigmoid, Softmax, ReLU, Tanh) dan jumlah neuron (4, 8, 16, 32, 64). Setiap batang mewakili rata-rata akurasi untuk kombinasi hyperparameter tertentu.