LAPORAN

MACHINE LEARNING

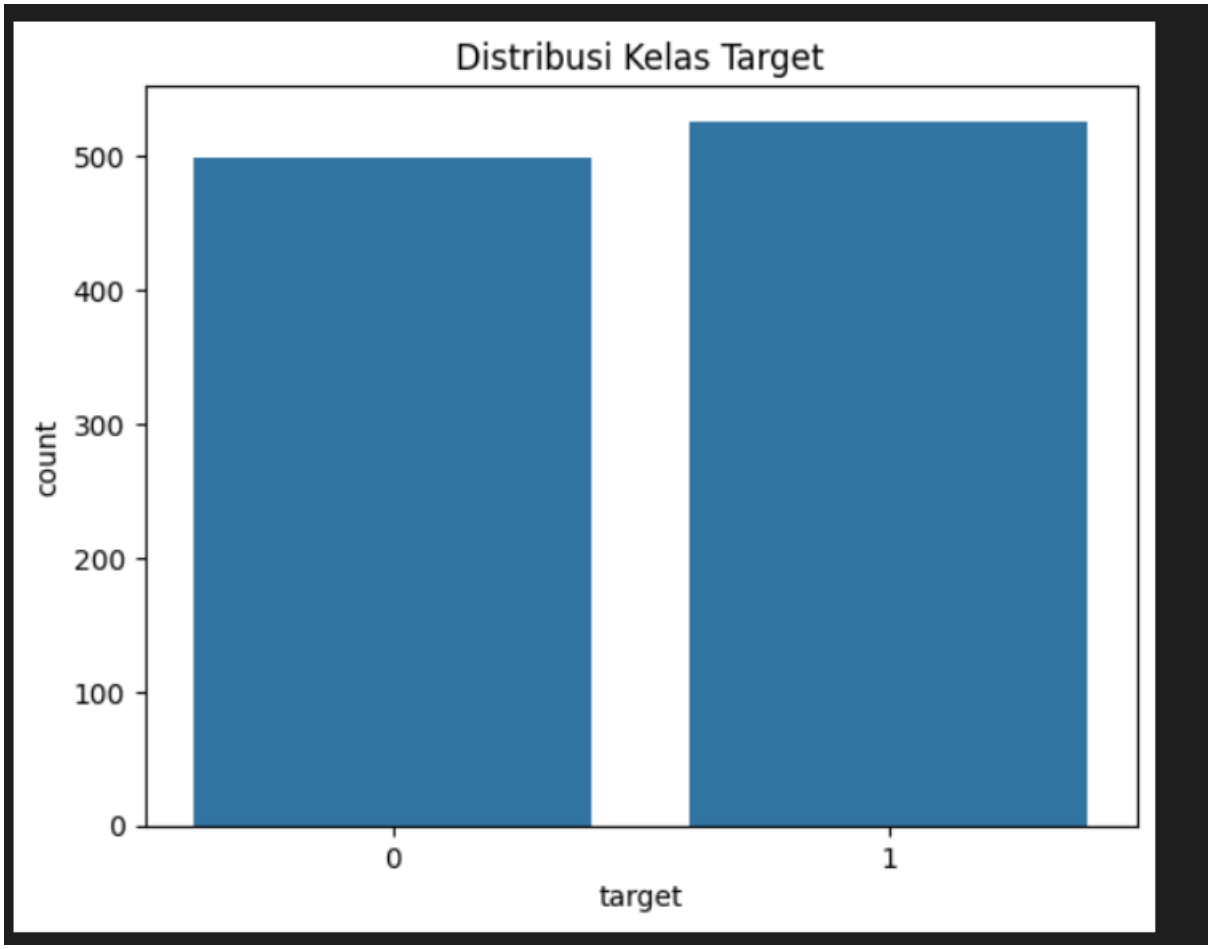
WEEK 11

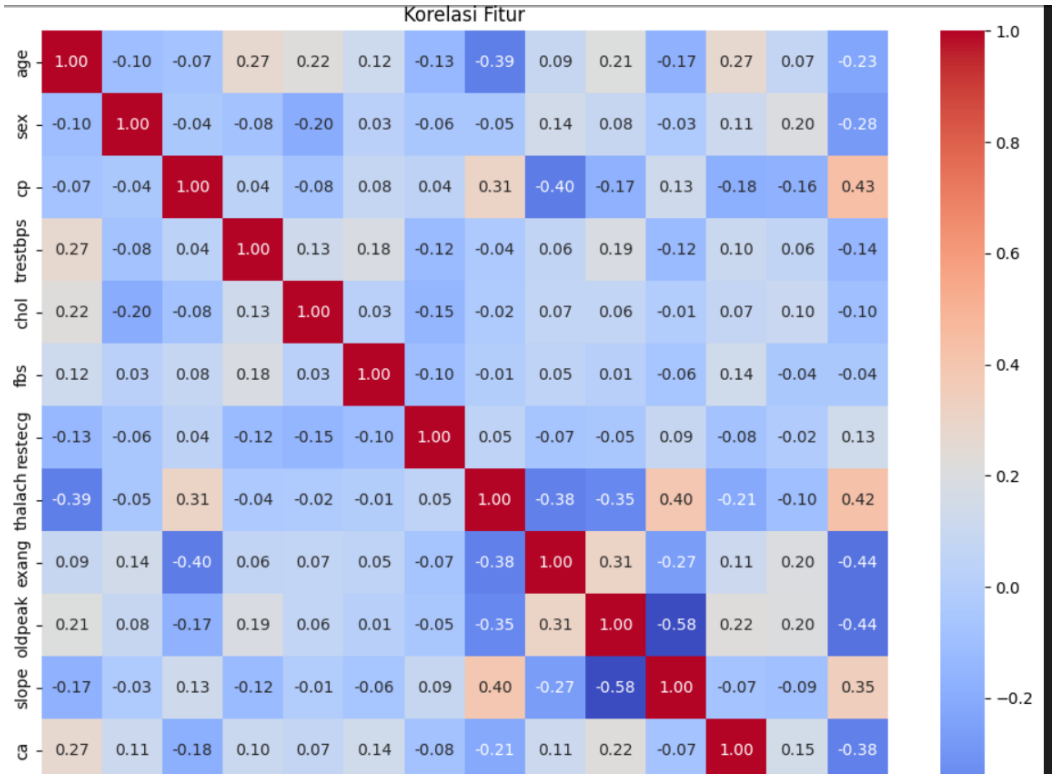
Deep Learning Dataset Heart Disease

```
# **Langkah 1: Membaca Dataset Heart Disease**
# Membaca dataset heart.csv
file_path = 'heart.csv'  # Sesuaikan dengan path file Anda
df = pd.read_csv(file_path)

# Menampilkan beberapa baris pertama data
print("Dataset Heart Disease:\n", df.head())

# Statistik dasar dari data
print("Statistik Deskriptif:\n", df.describe())
```

1. Exploratory Data Analysis (EDA)

```
Dataset Heart Disease:
    age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   52    1   0       125   212    0        1      168      0      1.0      2
1   53    1   0       140   203    1        0      155      1      3.1      0
2   70    1   0       145   174    0        1      125      1      2.6      0
3   61    1   0       148   203    0        1      161      0      0.0      2
4   62    0   0       138   294    1        1      106      0      1.9      1

   ca  thal  target
0   2     3       0
1   0     3       0
2   0     3       0
3   1     3       0
4   3     2       0
Statistik Deskriptif:
                age          sex           cp     trestbps        chol  \
count  1025.000000  1025.000000  1025.000000  1025.000000  1025.00000
mean     54.434146     0.695610     0.942439   131.611707   246.00000
std       9.072290     0.460373     1.029641    17.516718    51.59251
min      29.000000     0.000000     0.000000    94.000000   126.00000
25%      48.000000     0.000000     0.000000   120.000000   211.00000
50%      56.000000     1.000000     1.000000   130.000000   240.00000
75%      61.000000     1.000000     2.000000   140.000000   275.00000
max      77.000000     1.000000     3.000000   200.000000   564.00000
...
25%       1.000000     0.000000     2.000000     0.000000
50%       1.000000     0.000000     2.000000     1.000000
75%       2.000000     1.000000     3.000000     1.000000
max       2.000000     4.000000     3.000000     1.000000
```


Distribusi Kelas Target
count
500
400
300
200
100
0
0
1
target


Korelasi Fitur
age 1.00 -0.10 -0.07 0.27 0.22 0.12 -0.13 -0.39 0.09 0.21 -0.17 0.27 0.07 -0.23
sex -0.10 1.00 -0.04 -0.08 -0.20 0.03 -0.06 -0.05 0.14 0.08 -0.03 0.11 0.20 -0.28
cp -0.07 -0.04 1.00 0.04 -0.08 0.08 0.04 0.31 -0.40 -0.17 0.13 -0.18 -0.16 0.43
trestbps 0.27 -0.08 0.04 1.00 0.13 0.18 -0.12 -0.04 0.06 0.19 -0.12 0.10 0.06 -0.14
chol 0.22 -0.20 -0.08 0.13 1.00 0.03 -0.15 -0.02 0.07 0.06 -0.01 0.07 0.10 -0.10
fbs 0.12 0.03 0.08 0.18 0.03 1.00 -0.10 -0.01 0.05 0.01 -0.06 0.14 -0.04 -0.04
restecg -0.13 -0.06 0.04 -0.12 -0.15 -0.10 1.00 0.05 -0.07 -0.05 0.09 -0.08 -0.02 0.13
thalach -0.39 -0.05 0.31 -0.04 -0.02 -0.01 0.05 1.00 -0.38 -0.35 0.40 -0.21 -0.10 0.42
exang 0.09 0.14 -0.40 0.06 0.07 0.05 -0.07 -0.38 1.00 0.31 -0.27 0.11 0.20 -0.44
oldpeak 0.21 0.08 -0.17 0.19 0.06 0.01 -0.05 -0.35 0.31 1.00 -0.58 0.22 0.20 -0.44
slope -0.17 -0.03 0.13 -0.12 -0.01 -0.06 0.09 0.40 -0.27 -0.58 1.00 -0.07 -0.09 0.35
ca 0.27 0.11 -0.18 0.10 0.07 0.14 -0.08 -0.21 0.11 0.22 -0.07 1.00 0.15 -0.38
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

2. **Membagi Data Menjadi Train dan Test**

```
# **Langkah 3: Membagi Data menjadi Train dan Test Set**
# Memisahkan fitur dan target
X = df.drop(columns=['target'])
y = df['target']

# Komentar: Data dipecah menjadi data latih dan uji untuk evaluasi model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standarisasi data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Konversi data menjadi tensor PyTorch
X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train.values, dtype=torch.long)
y_test_tensor = torch.tensor(y_test.values, dtype=torch.long)
```

Membagi Data menjadi Train dan Test Set

1. Memisahkan Fitur dan Target
   - X berisi fitur (kolom data selain target).
   - y berisi target (kolom yang akan diprediksi, yaitu 'target').
2. Membagi Data
   - Menggunakan train_test_split dari sklearn untuk membagi data menjadi set latih (X_train, y_train) dan set uji (X_test, y_test) dengan rasio 80:20. Parameter random_state=42 memastikan pembagian data yang konsisten.
3. Standarisasi Data
   - Data X_train dan X_test dinormalisasi menggunakan StandardScaler agar nilai fitur memiliki distribusi normal dengan mean 0 dan standar deviasi 1. Hal ini penting untuk mempercepat konvergensi model.
4. Konversi ke Tensor PyTorch
   - Data X_train, X_test, y_train, dan y_test dikonversi menjadi tensor PyTorch menggunakan torch.tensor. Tipe data (dtype) untuk fitur adalah torch.float32, sedangkan untuk target adalah torch.long.

3. **Membuat Model dengan Variasi Parameter**

```python
# **Langkah 4: Membuat Model MLP dengan Variasi Parameter**
def create_mlp_model(input_dim, hidden_layers, activation_fn):
    layers = []
    for neurons in hidden_layers:
        layers.append(nn.Linear(input_dim, neurons))
        layers.append(activation_fn)
        input_dim = neurons
    layers.append(nn.Linear(input_dim, 2))  # Output layer untuk 2 kelas
    return nn.Sequential(*layers)

# Parameter yang akan dibandingkan
hidden_layer_configs = [[4], [8, 4], [16, 8, 4]]  # Variasi hidden layers
activation_functions = {
    "ReLU": nn.ReLU(),
    "Sigmoid": nn.Sigmoid(),
    "Tanh": nn.Tanh(),
    "Linear": nn.Identity(),
    "Softmax": nn.Softmax(dim=1)
}
epochs_options = [10, 25, 50, 100]
learning_rates = [0.1, 0.01, 0.001, 0.0001]
batch_sizes = [16, 32, 64, 128]

# Variabel untuk menyimpan hasil terbaik
best_accuracy = 0
best_config = None
```

1. Fungsi create_mlp_model

- Membuat model MLP secara dinamis berdasarkan:
  - Input Dimension (input_dim): Jumlah fitur input.
  - Hidden Layers (hidden_layers): Daftar jumlah neuron di setiap hidden layer.
  - Activation Function (activation_fn): Fungsi aktivasi yang digunakan di setiap hidden layer.
- Layer terakhir adalah output layer dengan 2 neuron untuk klasifikasi 2 kelas.

2. Parameter yang Akan Dibandingkan

- Hidden Layer Configs: Variasi jumlah neuron pada hidden layer, seperti [4], [8, 4], dan [16, 8, 4].
- Activation Functions: Fungsi aktivasi seperti ReLU, Sigmoid, Tanh, Linear, dan Softmax.
- Epochs Options: Pilihan jumlah epoch untuk pelatihan (10, 25, 50, 100).
- Learning Rates: Pilihan learning rate (0.1, 0.01, 0.001, 0.0001).
- Batch Sizes: Pilihan ukuran batch (16, 32, 64, 128).

3. Variabel Hasil Terbaik

- best_accuracy: Menyimpan nilai akurasi terbaik selama eksperimen.
- best_config: Menyimpan konfigurasi parameter yang menghasilkan akurasi terbaik.

4. **Output**

```
Config: HL=[4], Act=ReLU, Ep=10, LR=0.1, BS=16, Acc=0.8195
Config: HL=[4], Act=ReLU, Ep=10, LR=0.1, BS=32, Acc=0.8341
Config: HL=[4], Act=ReLU, Ep=10, LR=0.1, BS=64, Acc=0.8585
Config: HL=[4], Act=ReLU, Ep=10, LR=0.1, BS=128, Acc=0.8585
Config: HL=[4], Act=ReLU, Ep=10, LR=0.01, BS=16, Acc=0.8244
Config: HL=[4], Act=ReLU, Ep=10, LR=0.01, BS=32, Acc=0.8293
Config: HL=[4], Act=ReLU, Ep=10, LR=0.01, BS=64, Acc=0.7951
Config: HL=[4], Act=ReLU, Ep=10, LR=0.01, BS=128, Acc=0.8098
Config: HL=[4], Act=ReLU, Ep=10, LR=0.001, BS=16, Acc=0.7902
Config: HL=[4], Act=ReLU, Ep=10, LR=0.001, BS=32, Acc=0.7707
Config: HL=[4], Act=ReLU, Ep=10, LR=0.001, BS=64, Acc=0.7512
Config: HL=[4], Act=ReLU, Ep=10, LR=0.001, BS=128, Acc=0.5024
Config: HL=[4], Act=ReLU, Ep=10, LR=0.0001, BS=16, Acc=0.6439
Config: HL=[4], Act=ReLU, Ep=10, LR=0.0001, BS=32, Acc=0.4976
Config: HL=[4], Act=ReLU, Ep=10, LR=0.0001, BS=64, Acc=0.5317
Config: HL=[4], Act=ReLU, Ep=10, LR=0.0001, BS=128, Acc=0.5951
Config: HL=[4], Act=ReLU, Ep=25, LR=0.1, BS=16, Acc=0.8780
Config: HL=[4], Act=ReLU, Ep=25, LR=0.1, BS=32, Acc=0.7902
Config: HL=[4], Act=ReLU, Ep=25, LR=0.1, BS=64, Acc=0.8634
Config: HL=[4], Act=ReLU, Ep=25, LR=0.1, BS=128, Acc=0.8634
Config: HL=[4], Act=ReLU, Ep=25, LR=0.01, BS=16, Acc=0.8878
Config: HL=[4], Act=ReLU, Ep=25, LR=0.01, BS=32, Acc=0.8488
Config: HL=[4], Act=ReLU, Ep=25, LR=0.01, BS=64, Acc=0.8098
Config: HL=[4], Act=ReLU, Ep=25, LR=0.01, BS=128, Acc=0.8390
Config: HL=[4], Act=ReLU, Ep=25, LR=0.001, BS=16, Acc=0.7805
...
Config: HL=[16, 8, 4], Act=Softmax, Ep=100, LR=0.0001, BS=64, Acc=0.4976
Config: HL=[16, 8, 4], Act=Softmax, Ep=100, LR=0.0001, BS=128, Acc=0.4976
Hasil Terbaik: {'hidden_layers': [8, 4], 'activation': 'ReLU', 'epochs': 100, 'learning_rate': 0.01, 'batch_size': 64}
Akurasi Terbaik: 0.9902
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
# Menampilkan hasil terbaik
print("Hasil Terbaik:", best_config)
print(f"Akurasi Terbaik: {best_accuracy:.4f}")
```

```
Hasil Terbaik: {'hidden_layers': [8, 4], 'activation': 'ReLU', 'epochs': 100, 'learning_rate': 0.01, 'batch_size': 64}
Akurasi Terbaik: 0.9902
```

Hasil Terbaik

1. Konfigurasi Optimal:
    - Hidden Layers: [8, 4].
    - Activation: ReLU.
    - Epochs: 100.
    - Learning Rate: 0.01.
    - Batch Size: 64.
2. Akurasi Tertinggi:
    - Model mencapai akurasi 0.9902, yang merupakan hasil terbaik dari semua konfigurasi yang dicoba.

Analisis

1. Pentingnya Hyperparameter Tuning:
    - Eksperimen menunjukkan bahwa kombinasi parameter yang berbeda menghasilkan akurasi yang bervariasi. Hal ini menunjukkan pentingnya memilih hyperparameter yang tepat untuk meningkatkan performa model.
2. Fungsi Aktivasi dan Struktur Hidden Layer:
    - Struktur hidden layer [8, 4] dengan fungsi aktivasi ReLU menunjukkan performa yang sangat baik, mengindikasikan bahwa konfigurasi ini cocok untuk dataset yang digunakan.
3. Epochs dan Learning Rate:
    - Model dengan jumlah epoch lebih besar (100) dan learning rate sedang (0.01) menunjukkan performa yang stabil, memungkinkan konvergensi model secara optimal.
4. Batch Size:
    - Ukuran batch sedang (64) memberikan hasil terbaik, mungkin karena keseimbangan antara kecepatan komputasi dan stabilitas gradien.

Kesimpulan

Hasil eksperimen ini menunjukkan bahwa kombinasi struktur hidden layer `[8, 4]`, fungsi aktivasi `ReLU`, epochs `100`, learning rate `0.01`, dan batch size `64` memberikan akurasi terbaik. Langkah ini penting untuk mengoptimalkan performa model MLP pada dataset tertentu.