LAPORAN

MACHINE LEARNING

WEEK 11

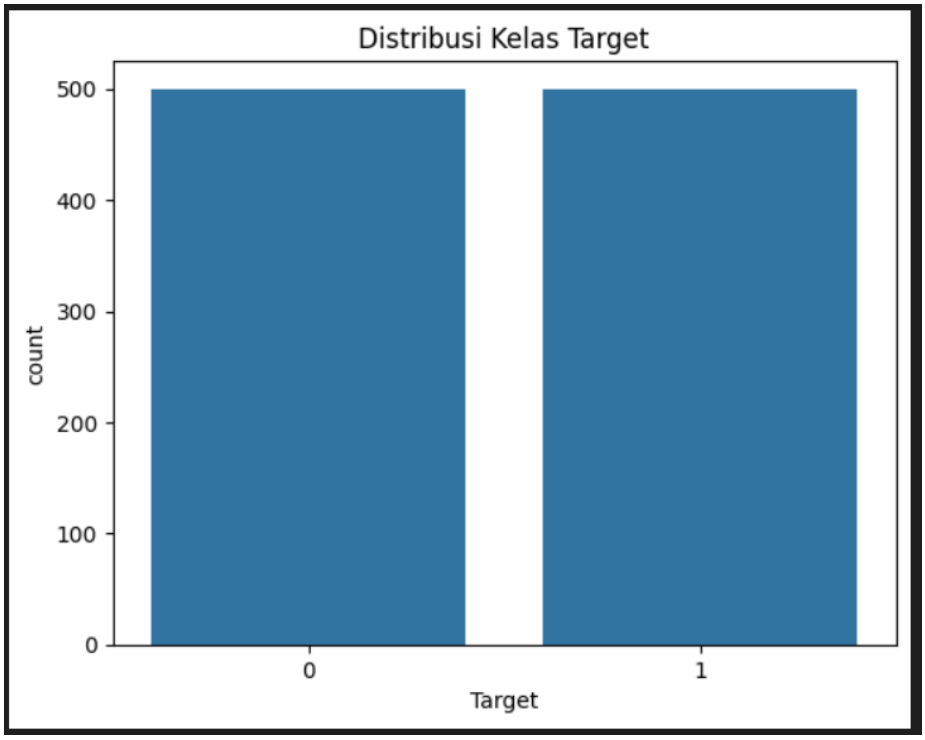
1. Classification Dummy Data

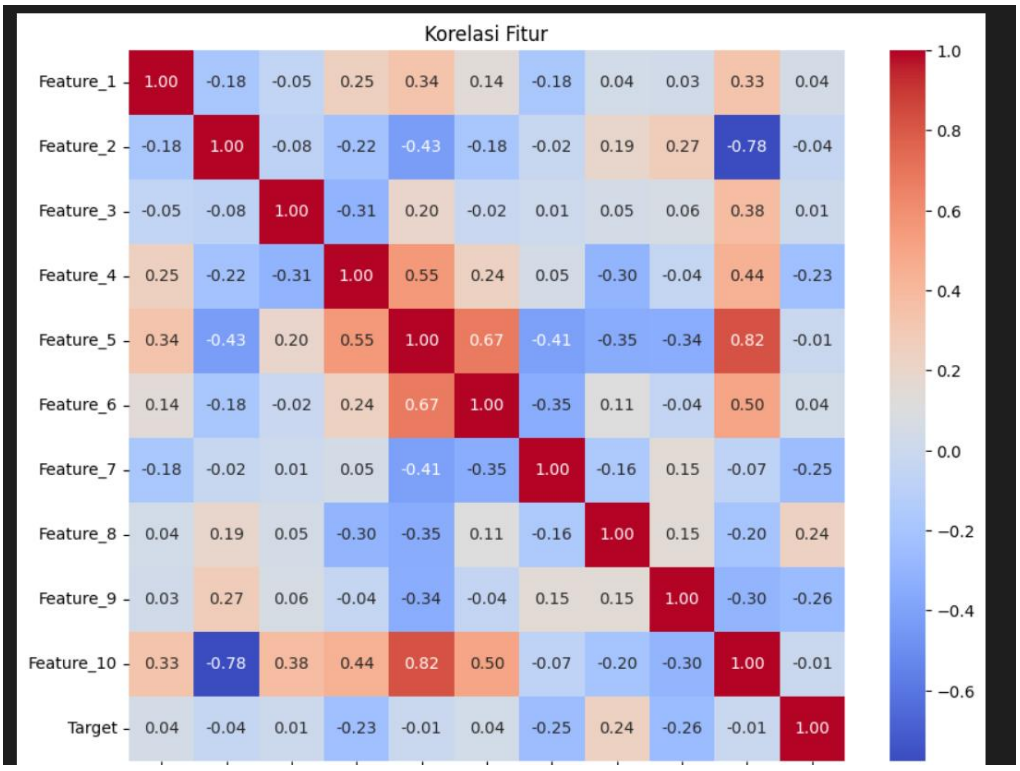
Dummy data pada Machine Learning adalah data yang dibuat secara sintetis untuk keperluan eksperimen, pelatihan, atau pengujian model machine learning. Dummy data digunakan terutama ketika data asli tidak tersedia, data sensitif tidak boleh digunakan, atau saat ingin membuat prototipe model dengan cepat.

```
# **Langkah 1: Membuat Dummy Data untuk Klasifikasi**
# Membuat dataset dummy dengan 2 kelas
# Komentar: Dataset ini dibuat untuk simulasi tugas klasifikasi
X, y = make_classification(n_samples=1000, n_features=10, n_informative=8, n_redundant=2, n_classes=2, random_state=42)
df = pd.DataFrame(X, columns=[f"Feature_{i+1}" for i in range(X.shape[1])])
df['Target'] = y
```

```
Dataset Dummy:
    Feature_1  Feature_2  Feature_3  Feature_4  Feature_5  Feature_6  \
0  -0.224515  -0.648598   2.805591   0.569525   2.676771   0.685312
1   1.038548  -1.316803   2.303387   1.233593   4.972675   1.086905
2   1.148988   1.794382   2.835693  -1.114858  -1.247120  -1.899268
3  -1.177318   2.420294  -0.363514  -1.086646  -2.824750  -2.722123
4   1.346717   0.089373   2.056613  -0.428365   0.285387   1.140716

   Feature_7  Feature_8  Feature_9  Feature_10  Target
0   0.306675   1.147291  -2.145905    2.477879       1
1  -3.548788   0.342810  -0.469005    2.787109       1
2  -0.381470   0.837209   0.163223   -0.997959       1
3  -0.225266   0.515079  -3.317272   -2.801517       1
4   0.721945   1.110152   2.426608    0.880496       0
Statistik Deskriptif:
         Feature_1    Feature_2    Feature_3    Feature_4    Feature_5  \
count  1000.000000  1000.000000  1000.000000  1000.000000  1000.000000
mean     -0.077343     1.115814    -0.020803    -0.573632    -1.732748
std       1.851235     1.587144     2.125429     1.846328     3.065499
min      -5.824009    -3.551943    -7.409478    -5.283366   -11.818684
25%      -1.227815     0.018902    -1.503226    -1.850948    -3.783477
50%       0.100892     1.145004    -0.086342    -0.696424    -1.626836
75%       1.148416     2.136035     1.468941     0.569953     0.281290
max       5.105162     7.011590     6.336243     5.517741     8.260046
...
25%       0.00000
50%       0.50000
75%       1.00000
max       1.00000
```


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


Korelasi Fitur
Feature_1 1.00 -0.18 -0.05 0.25 0.34 0.14 -0.18 0.04 0.03 0.33 0.04
Feature_2 -0.18 1.00 -0.08 -0.22 -0.43 -0.18 -0.02 0.19 0.27 -0.78 -0.04
Feature_3 -0.05 -0.08 1.00 -0.31 0.20 -0.02 0.01 0.05 0.06 0.38 0.01
Feature_4 0.25 -0.22 -0.31 1.00 0.55 0.24 0.05 -0.30 -0.04 0.44 -0.23
Feature_5 0.34 -0.43 0.20 0.55 1.00 0.67 -0.41 -0.35 -0.34 0.82 -0.01
Feature_6 0.14 -0.18 -0.02 0.24 0.67 1.00 -0.35 0.11 -0.04 0.50 0.04
Feature_7 -0.18 -0.02 0.01 0.05 -0.41 -0.35 1.00 -0.16 0.15 -0.07 -0.25
Feature_8 0.04 0.19 0.05 -0.30 -0.35 0.11 -0.16 1.00 0.15 -0.20 0.24
Feature_9 0.03 0.27 0.06 -0.04 -0.34 -0.04 0.15 0.15 1.00 -0.30 -0.26
Feature_10 0.33 -0.78 0.38 0.44 0.82 0.50 -0.07 -0.20 -0.30 1.00 -0.01
Target 0.04 -0.04 0.01 -0.23 -0.01 0.04 -0.25 0.24 -0.26 -0.01 1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

```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]
learning_rates = [0.1, 0.01, 0.001]
batch_sizes = [16, 32, 64]

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

Fungsi create_mlp_model digunakan untuk membangun model MLP berdasarkan jumlah fitur input, konfigurasi hidden layer, dan fungsi aktivasi. Setiap hidden layer ditambahkan secara dinamis sesuai dengan jumlah neuron yang didefinisikan dalam parameter hidden_layers, diikuti dengan fungsi aktivasi yang dipilih, dan diakhiri dengan layer output dengan 2 neuron untuk klasifikasi 2 kelas. Selanjutnya, berbagai parameter eksperimen ditentukan, termasuk konfigurasi hidden layer seperti [4], [8, 4], dan [16, 8, 4], serta fungsi aktivasi seperti ReLU, Sigmoid, Tanh, Linear, dan Softmax. Parameter lain seperti jumlah epoch, learning rate, dan ukuran batch juga divariasikan untuk menemukan konfigurasi terbaik. Variabel best_accuracy dan best_config disiapkan untuk menyimpan akurasi tertinggi dan konfigurasi parameter terbaik selama proses eksperimen. Pendekatan ini dirancang untuk melakukan hyperparameter tuning guna memilih model dengan performa terbaik untuk tugas klasifikasi.

```
Config: HL=[4], Act=ReLU, Ep=10, LR=0.1, BS=16, Acc=0.8850
Config: HL=[4], Act=ReLU, Ep=10, LR=0.1, BS=32, Acc=0.8900
Config: HL=[4], Act=ReLU, Ep=10, LR=0.1, BS=64, Acc=0.8600
Config: HL=[4], Act=ReLU, Ep=10, LR=0.01, BS=16, Acc=0.8700
Config: HL=[4], Act=ReLU, Ep=10, LR=0.01, BS=32, Acc=0.8900
Config: HL=[4], Act=ReLU, Ep=10, LR=0.01, BS=64, Acc=0.8450
Config: HL=[4], Act=ReLU, Ep=10, LR=0.001, BS=16, Acc=0.6900
Config: HL=[4], Act=ReLU, Ep=10, LR=0.001, BS=32, Acc=0.6150
Config: HL=[4], Act=ReLU, Ep=10, LR=0.001, BS=64, Acc=0.6400
Config: HL=[4], Act=ReLU, Ep=25, LR=0.1, BS=16, Acc=0.8800
Config: HL=[4], Act=ReLU, Ep=25, LR=0.1, BS=32, Acc=0.9050
Config: HL=[4], Act=ReLU, Ep=25, LR=0.1, BS=64, Acc=0.8800
Config: HL=[4], Act=ReLU, Ep=25, LR=0.01, BS=16, Acc=0.8750
Config: HL=[4], Act=ReLU, Ep=25, LR=0.01, BS=32, Acc=0.8700
Config: HL=[4], Act=ReLU, Ep=25, LR=0.01, BS=64, Acc=0.8850
Config: HL=[4], Act=ReLU, Ep=25, LR=0.001, BS=16, Acc=0.7900
Config: HL=[4], Act=ReLU, Ep=25, LR=0.001, BS=32, Acc=0.7950
Config: HL=[4], Act=ReLU, Ep=25, LR=0.001, BS=64, Acc=0.7350
Config: HL=[4], Act=ReLU, Ep=50, LR=0.1, BS=16, Acc=0.8600
Config: HL=[4], Act=ReLU, Ep=50, LR=0.1, BS=32, Acc=0.8500
Config: HL=[4], Act=ReLU, Ep=50, LR=0.1, BS=64, Acc=0.8900
Config: HL=[4], Act=ReLU, Ep=50, LR=0.01, BS=16, Acc=0.9050
Config: HL=[4], Act=ReLU, Ep=50, LR=0.01, BS=32, Acc=0.8800
Config: HL=[4], Act=ReLU, Ep=50, LR=0.01, BS=64, Acc=0.8900
Config: HL=[4], Act=ReLU, Ep=50, LR=0.001, BS=16, Acc=0.8600
...
Config: HL=[16, 8, 4], Act=Softmax, Ep=50, LR=0.001, BS=32, Acc=0.8850
Config: HL=[16, 8, 4], Act=Softmax, Ep=50, LR=0.001, BS=64, Acc=0.8100
Hasil Terbaik: {'hidden_layers': [4], 'activation': 'Sigmoid', 'epochs': 50, 'learning_rate': 0.1, 'batch_size': 16}
Akurasi Terbaik: 0.9500
```

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

```
Hasil Terbaik: {'hidden_layers': [4], 'activation': 'Sigmoid', 'epochs': 50, 'learning_rate': 0.1, 'batch_size': 16}
Akurasi Terbaik: 0.9500
```

Gambar ini menampilkan hasil eksperimen dengan berbagai konfigurasi hyperparameter untuk model MLP. Berikut penjelasannya:

1. Format Konfigurasi
   Setiap baris mencantumkan konfigurasi parameter yang digunakan dalam satu eksperimen:
   - HL: Konfigurasi hidden layer (contoh: [4] berarti ada satu hidden layer dengan 4 neuron).
   - Act: Fungsi aktivasi yang digunakan (contoh: ReLU atau Softmax).
   - Ep: Jumlah epoch yang digunakan untuk pelatihan (contoh: 10, 25, 50).
   - LR: Learning rate yang digunakan (contoh: 0.1, 0.01, 0.001).
   - BS: Ukuran batch (contoh: 16, 32, 64).
   - Acc: Akurasi yang diperoleh pada konfigurasi tersebut.
2. Hasil Akurasi
   Untuk setiap kombinasi parameter, akurasi model ditampilkan. Akurasi tertinggi dalam eksperimen ini adalah 0.9500.
3. Konfigurasi Terbaik
   Pada akhir eksperimen, konfigurasi terbaik ditampilkan:

- Hidden layers: [4] (satu hidden layer dengan 4 neuron).
- Activation: Sigmoid (fungsi aktivasi sigmoid).
- Epochs: 50.
- Learning rate: 0.1.
- Batch size: 16.

4. Analisis Hasil
    - Kombinasi hidden layer yang sederhana ([4]) dengan fungsi aktivasi Sigmoid memberikan hasil terbaik.
    - Learning rate yang lebih besar (0.1) menunjukkan kinerja lebih baik dibandingkan learning rate kecil dalam eksperimen ini.
    - Ukuran batch kecil (16) lebih optimal dalam mencapai akurasi tinggi, mungkin karena lebih sering memperbarui bobot selama pelatihan.

Hasil ini menunjukkan pentingnya eksperimen untuk memilih hyperparameter yang tepat dalam membangun model MLP dengan performa optimal.