Laporan Akhir: Klasifikasi Bunga Iris Menggunakan MLP (PyTorch)

1. Pendahuluan

Proyek ini bertujuan untuk membangun model klasifikasi bunga Iris menggunakan arsitektur Multi-Layer Perceptron (MLP) dengan PyTorch. Dataset Iris merupakan dataset klasik dalam pembelajaran mesin yang terdiri dari tiga spesies bunga: Setosa, Versicolor, dan Virginica, dengan empat fitur utama: panjang dan lebar sepal serta panjang dan lebar petal.

2. Persiapan Data

a) Import Library: Menggunakan library seperti pandas, numpy, matplotlib, seaborn, dan modul dari scikit-learn untuk manipulasi data dan visualisasi.
b) Load Dataset: Dataset Iris dimuat menggunakan load_iris() dari scikit-learn.
c) Eksplorasi Data: Data divisualisasikan menggunakan pairplot dari seaborn untuk melihat distribusi fitur antar kelas.
d) Preprocessing:
   i. Label Encoding: Mengubah label string menjadi numerik menggunakan LabelEncoder.
   ii. Normalisasi: Fitur dinormalisasi menggunakan StandardScaler untuk meningkatkan stabilitas model saat pelatihan.
   iii. Split Data: Data dibagi menjadi data latih dan data uji dengan rasio 80:20 menggunakan train_test_split.

3. Arsitektur Model

Model MLP yang digunakan memiliki satu hidden layer dengan struktur sebagai berikut:

a) Input Layer: 4 neuron (sesuai dengan jumlah fitur).
b) Hidden Layer: 16 neuron dengan fungsi aktivasi ReLU.
c) Output Layer: 3 neuron (sesuai dengan jumlah kelas) dengan fungsi aktivasi Softmax.

```python
# Model MLP dengan 1 hidden layer
class MLP(nn.Module):
    def __init__(self):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(4, 16)       # Layer 1: input 4 fitur → 16 neuron
        self.relu = nn.ReLU()             # Fungsi aktivasi ReLU
        self.fc2 = nn.Linear(16, 3)       # Output: 3 neuron (kelas)
        self.softmax = nn.Softmax(dim=1)  # Softmax untuk klasifikasi multi-kelas

    def forward(self, x):
        x = self.relu(self.fc1(x))        # Forward ke hidden layer + aktivasi
        x = self.softmax(self.fc2(x))     # Output layer + softmax
        return x
```

4. Pelatihan Model

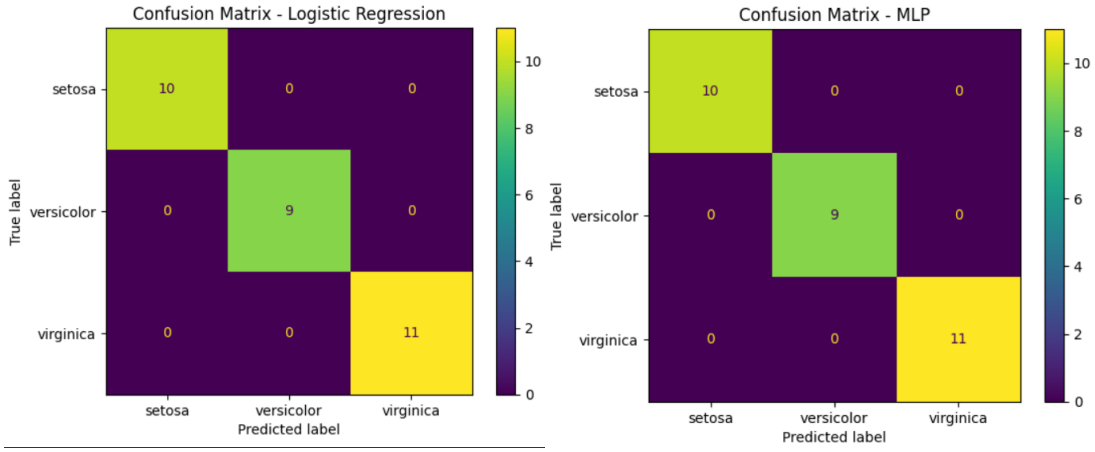
a) Inisialisasi:
   i. Model diinisialisasi menggunakan kelas MLP.
   ii. Fungsi loss yang digunakan adalah nn.CrossEntropyLoss(), yang menggabungkan LogSoftmax dan NLLLoss untuk klasifikasi multi-kelas.
   iii. Optimizer yang digunakan adalah Adam dengan learning rate 0.01.

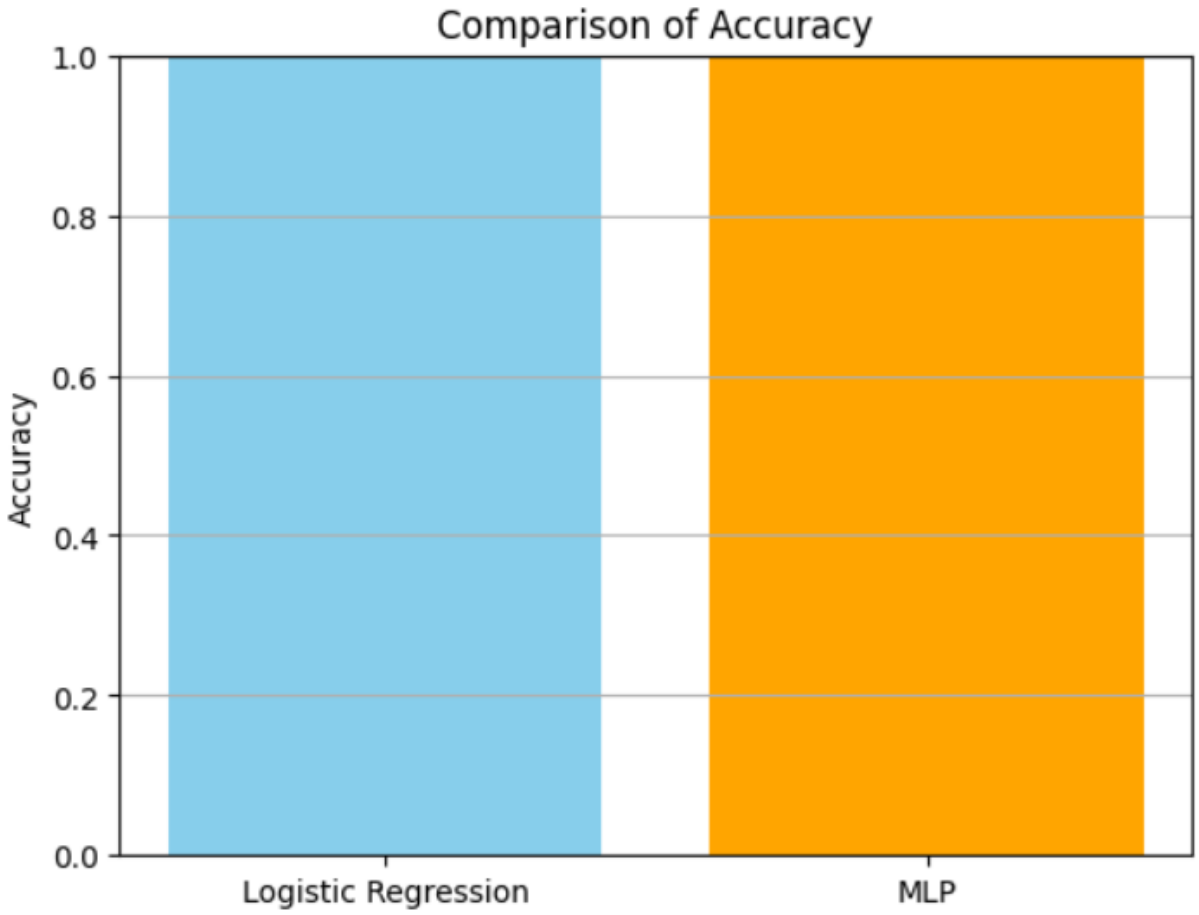
b) Training Loop:
   i. Model dilatih selama 100 epoch.
   ii. Pada setiap epoch, model melakukan forward pass, menghitung loss, melakukan backpropagation, dan mengupdate bobot.
   iii. Akurasi dan loss dicatat untuk setiap epoch.

5. Evaluasi Model

a) Akurasi:
   i. Akurasi model dievaluasi pada data uji.
   ii. Model MLP dibandingkan dengan model baseline Logistic Regression.

b) Visualisasi:
   i. Loss dan akurasi selama pelatihan divisualisasikan menggunakan matplotlib.
   ii. Confusion matrix ditampilkan untuk kedua model untuk melihat performa klasifikasi pada masing-masing kelas.





Visualisasi Perbandingan Regresi Logistic dan Model MLP

6. Prediksi Data Baru

Model dapat digunakan untuk memprediksi kelas dari data baru dengan langkah-langkah berikut:

a) Data baru dinormalisasi menggunakan StandardScaler yang telah dilatih.
b) Data dikonversi menjadi tensor PyTorch.
c) Model melakukan prediksi dan hasilnya dikonversi kembali ke label asli menggunakan LabelEncoder.

```python
def predict_new(sample):
    model.eval()
    with torch.no_grad():
        sample_scaled = scaler.transform([sample])  # Normalisasi
        sample_tensor = torch.tensor(sample_scaled, dtype=torch.float32)
        output = model(sample_tensor)
        _, predicted = torch.max(output, 1)
        return encoder.inverse_transform(predicted.numpy())[0]
```

7. Kesimpulan

a) Model MLP dengan satu hidden layer mampu mengklasifikasikan bunga Iris dengan akurasi yang kompetitif dibandingkan dengan Logistic Regression.
b) MLP memiliki keunggulan dalam menangkap hubungan non-linear dalam data, namun memerlukan waktu pelatihan yang lebih lama dan tuning parameter yang lebih kompleks.
c) Logistic Regression lebih cepat dan cocok untuk data yang memiliki hubungan linear.