# LAPORAN AKHIR KLASIFIKASI BUNGA IRIS
# MATAKULIAH PEMBELAJARAN MESIN 2

Dosen Pengampu :

Dr. Oddy Virgantara Putra, S.Kom., M.T.


Disusun Oleh :

| | |
|---|---|
| Devianest Narendra | 442023618087 |
| Zainab Ahmad | 442023618107 |
| Adya Rusmalillah | 442023618093 |
| Naila Fatikhah | 442023618086 |

**PROGRAM STUDI TEKNIK INFORMATIKA**

**UNIVERSITAS DARUSSALAM GONTOR**

**2025/2026**

# A. PENDAHULUAN

## a. Klasifikasi Bunga Iris Menggunakan Multilayer Perceptron (MLP)

Multilayer Perceptron (MLP) merupakan salah satu jenis jaringan saraf tiruan (Artificial Neural Network) yang umum digunakan untuk permasalahan klasifikasi. Dalam penelitian ini, MLP digunakan untuk mengklasifikasikan tiga jenis bunga Iris, yaitu *Iris setosa*, *Iris versicolor*, dan *Iris virginica*, berdasarkan empat fitur utama: panjang sepal, lebar sepal, panjang petal, dan lebar petal.

## b. Struktur Dataset

Dataset Iris terdiri dari 150 data sampel, dengan masing-masing 50 data untuk setiap jenis bunga. Setiap sampel memiliki atribut sebagai berikut:

- **Sepal Length (cm)**: Panjang kelopak luar bunga
- **Sepal Width (cm)**: Lebar kelopak luar bunga
- **Petal Length (cm)**: Panjang mahkota bunga
- **Petal Width (cm)**: Lebar mahkota bunga

## c. Penerapan MLP

MLP dalam klasifikasi ini terdiri dari:

- **Input layer**: 4 neuron (untuk masing-masing fitur)
- **Hidden layer**: Satu atau lebih lapisan tersembunyi dengan jumlah neuron bervariasi
- **Output layer**: 3 neuron (mewakili tiga kelas bunga)

Model dilatih menggunakan algoritma backpropagation dengan fungsi aktivasi seperti ReLU pada hidden layer dan Softmax pada output layer. Fungsi loss yang digunakan adalah Cross-Entropy Loss, dan proses optimasi dilakukan dengan algoritma seperti Adam atau SGD.

## B. PENJELASAN

### a. DATASET

```
import pandas as pd
```

Penggunaan : Mengimpor library pandas untuk mengolah data dalam bentuk tabel (DataFrame).

```
from sklearn.datasets import load_iris
```

Penggunaan : Mengambil dataset **Iris**, dataset klasik untuk klasifikasi bunga iris (setosa, versicolor, virginica).

```
from sklearn.model_selection import train_test_split
```

Penggunaan : Berfungsi Digunakan untuk membagi data menjadi data **latih** dan **uji**

**Kode untuk menyiapkan Dataset :**

```
def main():
    iris = load_iris(as_frame=True)
    X = iris.data
    y = iris.target
    df = pd.concat([X, y.rename("target")], axis=1)

    # split train/test
    train_df, test_df = train_test_split(df, test_size=0.3, random_state=42, stratify=df["target"])

    # simpan ke CSV
    train_df.to_csv("train.csv", index=False)
    test_df.to_csv("test.csv", index=False)
    print("Saved train.csv and test.csv")

if __name__ == "__main__":
    main()
```

keterangan:

Di sini, data dan target digabung menjadi satu DataFrame untuk mempermudah pengolahan.

Data dibagi menjadi 70% training dan 30% testing.

- random_state=42: agar pembagian selalu konsisten setiap kali dijalankan.
- stratify=df["target"]: memastikan proporsi kelas tetap seimbang di training dan testing.

Menyimpan dua file hasil:

- train.csv → untuk melatih model.
- test.csv → untuk menguji model.

__name__ == "__main__" adalah cara standar di Python untuk menjalankan kode hanya jika file ini dijalankan langsung (bukan di-import dari file lain).

## b. TRAINING

```
def main():
    # 1) Persiapan data
    df = pd.read_csv("train.csv")  # pastikan dibuat dengan data_prep.py
    X = df.drop(columns="target").values.astype(float)
    y = df["target"].values.astype(int)

    # split train / validation
    X_train, X_val, y_train, y_val = train_test_split(
        X, y, test_size=0.2, random_state=42, stratify=y
    )

    # scaling
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_val = scaler.transform(X_val)
```

**Keterangan :**

- Membagi data menjadi **training (80%)** dan **validation (20%)**. Lalu distandarkan (mean = 0, std = 1) supaya pelatihan lebih stabil with open("scaler.pkl", "wb") as f
- pickle.dump(scaler, f)
- Scaler disimpan agar bisa digunakan lagi saat **inferensi/prediksi** nanti.
- TensorDataset & DataLoader, Membungkus data menjadi bentuk yang bisa diproses oleh PyTorch dengan batch size = 16.

```python
def main():
    # 1) Persiapan data
    df = pd.read_csv("train.csv")  # pastikan dibuat dengan data_prep.py
    X = df.drop(columns="target").values.astype(float)
    y = df["target"].values.astype(int)

    # split train / validation
    X_train, X_val, y_train, y_val = train_test_split(
        X, y, test_size=0.2, random_state=42, stratify=y
    )

    # scaling
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_val = scaler.transform(X_val)

    # simpan scaler untuk inferensi nanti
    with open("scaler.pkl", "wb") as f:
        pickle.dump(scaler, f)
    print("[INFO] Scaler disimpan di scaler.pkl")
```

**Keterangan :**

- Modelnya sederhana:
- Input: 4 neuron (karena 4 fitur)
- Hidden: 16 neuron + ReLU
- Output: 3 neuron (untuk 3 kelas)

criterion = nn.CrossEntropyLoss()

optimizer = optim.Adam(model.parameters(), lr=1e-3)

- Menggunakan CrossEntropyLoss untuk klasifikasi.
- Optimizer: Adam, learning rate = 0.001

```python
# 6) Loop pelatihan per-epoch
for epoch in range(1, n_epochs + 1):
    # training
    model.train()
    total_loss = 0.0
    for xb, yb in train_loader:
        xb, yb = xb.to(device), yb.to(device)
        optimizer.zero_grad()
        logits = model(xb)
        loss = criterion(logits, yb)
        loss.backward()
        optimizer.step()
        total_loss += loss.item() * xb.size(0)

    train_loss = total_loss / len(train_loader.dataset)

    # validation
    model.eval()
    correct = 0
    with torch.no_grad():
        for xb, yb in val_loader:
            xb, yb = xb.to(device), yb.to(device)
            logits = model(xb)
            preds = logits.argmax(dim=1)
            correct += (preds == yb).sum().item()
    val_acc = correct / len(val_loader.dataset)
```

**Keterangan :**

**Training Loop**

Melatih selama 100 epoch. Setiap epoch:

- Hitung loss training.
- Hitung akurasi validasi.
- Simpan model terbaik jika akurasi validasi meningkat.

plt.plot(epochs, train_losses)

plt.plot(epochs, val_accuracies)

- Menampilkan grafik:

- Penurunan loss selama training.
- Peningkatan akurasi selama validasi.

## c. PREDICT

```python
import pandas as pd
import pickle

def main():
    # load model pipeline
    with open("model.pkl", "rb") as f:
        pipeline = pickle.load(f)

    # load test data
    test_df = pd.read_csv("test.csv")
    X_test = test_df.drop(columns="target")
    y_true = test_df["target"]

    # prediksi
    y_pred = pipeline.predict(X_test)

    # simpan hasil prediksi
    out = X_test.copy()
    out["true_label"] = y_true
    out["pred_label"] = y_pred
    out.to_csv("predictions.csv", index=False)

    print("Hasil prediksi disimpan di predictions.csv")

if __name__ == "__main__":
    main()
```

**Keterangan :**

Digunakan untuk melakukan prediksi otomatis terhadap data uji (test.csv) menggunakan model Machine Learning yang sudah dilatih dan disimpan (model.pkl). Hasil prediksi tersebut kemudian disimpan dalam file CSV baru (predictions.csv) bersama label aslinya, sehingga bisa dievaluasi secara langsung.

Fungsi Utama:

1. Prediksi Data Uji: Menghasilkan label prediksi berdasarkan data fitur yang ada di test.csv.
2. Penyimpanan Hasil: Menyimpan hasil prediksi beserta label asli ke dalam CSV baru.
3. Evaluasi Mudah: Mempermudah pengecekan performa model dengan melihat hasil prediksi dan label asli secara berdampingan.

## d. MEMPREDIKSI DATA BARU

```
class IrisNet(nn.Module):
    def __init__(self, input_dim=4, hidden_dim=16, output_dim=3):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(input_dim, hidden_dim),
            nn.ReLU(),
            nn.Linear(hidden_dim, output_dim)
        )
    def forward(self, x):
        return self.net(x)

def main():
    # 1) Input fitur dari user
    try:
        sl = float(input("Masukkan sepal length  (cm): "))
        sw = float(input("Masukkan sepal width   (cm): "))
        pl = float(input("Masukkan petal length  (cm): "))
        pw = float(input("Masukkan petal width   (cm): "))
    except ValueError:
        print("Input tidak valid. Pastikan Anda memasukkan angka.")
        return
```

```
    with open("scaler.pkl", "rb") as f:
        scaler = pickle.load(f)

    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model = IrisNet().to(device)
    model.load_state_dict(torch.load("best_model.pth", map_location=device))
    model.eval()

    # 3) Preprocessing & prediksi
    import numpy as np
    X = np.array([[sl, sw, pl, pw]], dtype=float)
    X_scaled = scaler.transform(X)

    with torch.no_grad():
        inp = torch.tensor(X_scaled, dtype=torch.float32).to(device)
        logits = model(inp)
        pred = int(logits.argmax(dim=1).cpu().item())

    # 4) Map label ke nama kelas
    target_names = {0: "setosa", 1: "versicolor", 2: "virginica"}
    klas = target_names.get(pred, str(pred))

    print(f"\nHasil prediksi: {klas.capitalize()} (kelas {pred})")

if __name__ == "__main__":
    main()
```

**Keterangan :**

Kodingan diatas berfungsi sebagai **bagian inference** dari metode MLP (Multi-Layer Perceptron) yang sudah kita latih sebelumnya:

- **Definisi Arsitektur MLP**

  Membuat ulang jaringan 2-layer (4 → 16 → 3 neuron) persis seperti saat training.

- **Memuat Scaler & Bobot Model**
  Menggunakan `scaler.pkl` untuk menormalisasi data baru dengan skala yang sama, lalu memuat bobot `best_model.pth` ke MLP.
- **Preprocessing Input**
  Mengambil empat nilai fitur (sepal/petal length & width) dari pengguna, mengubah menjadi array, lalu menerapkan normalisasi.
- **Forward Pass & Prediksi**
  Menjalankan data yang sudah diskala ke dalam MLP, menghitung output (logits), dan memilih neuron dengan nilai tertinggi sebagai kelas prediksi.
- **Output Kelas**
  Menerjemahkan angka prediksi (0,1,2) menjadi nama bunga (setosa/versicolor/virginica) dan menampilkannya di layar.

Secara singkat, kode ini memungkinkan MLP yang sudah dilatih untuk **menerima input manual**, memprosesnya dengan **forward propagation**, dan **menghasilkan prediksi** jenis bunga Iris.