Laporan Akhir: Klasifikasi Bunga Iris dengan Multilayer Perceptron (MLP)

Mata Kuliah: Pembelajaran Mesin 2 **Dosen Pengampu:** Dr. Oddy Virgantara Putra, S.Kom., M.T.

Kelompok:

- **Rizky Cahyono Putra**
- **Raffa Arvel**
- **Syaifan Nur**
- **Irfansyah**
- **Galang Alvian**

Laporan ini menyajikan hasil implementasi dan evaluasi model klasifikasi bunga Iris menggunakan jaringan syaraf tiruan **Multilayer Perceptron (MLP)** dengan PyTorch, serta perbandingannya dengan model baseline **Logistic Regression**.

Jika kamu ingin saya bantu masukkan ini ke dalam template PDF atau markdown untuk konversi otomatis ke PDF, cukup beri tahu saja!

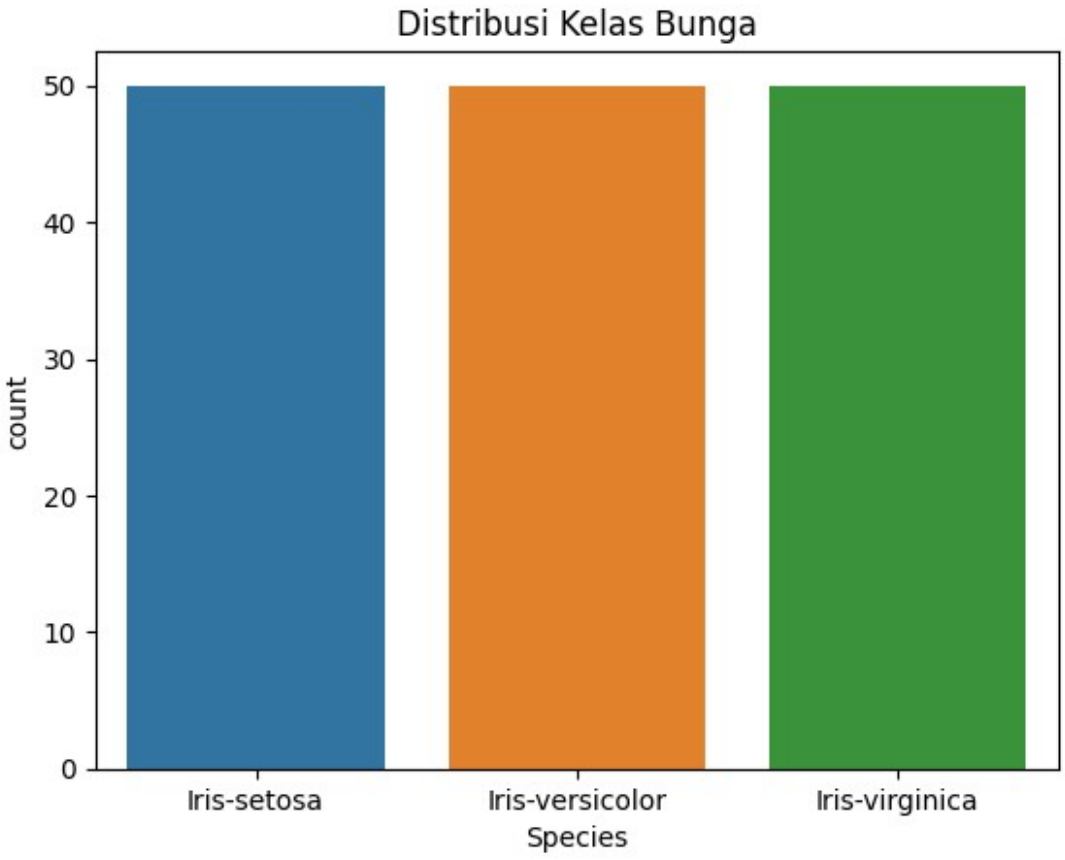
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
```

Eksplorasi Data

```
df = pd.read_csv('/kaggle/input/iris/Iris.csv')
df.head()
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
Species
0   1            5.1           3.5            1.4           0.2  Iris-
setosa
1   2            4.9           3.0            1.4           0.2  Iris-
setosa
2   3            4.7           3.2            1.3           0.2  Iris-
setosa
3   4            4.6           3.1            1.5           0.2  Iris-
setosa
4   5            5.0           3.6            1.4           0.2  Iris-
setosa
```

```
sns.countplot(x='Species', data=df)
plt.title('Distribusi Kelas Bunga')
plt.show()
```

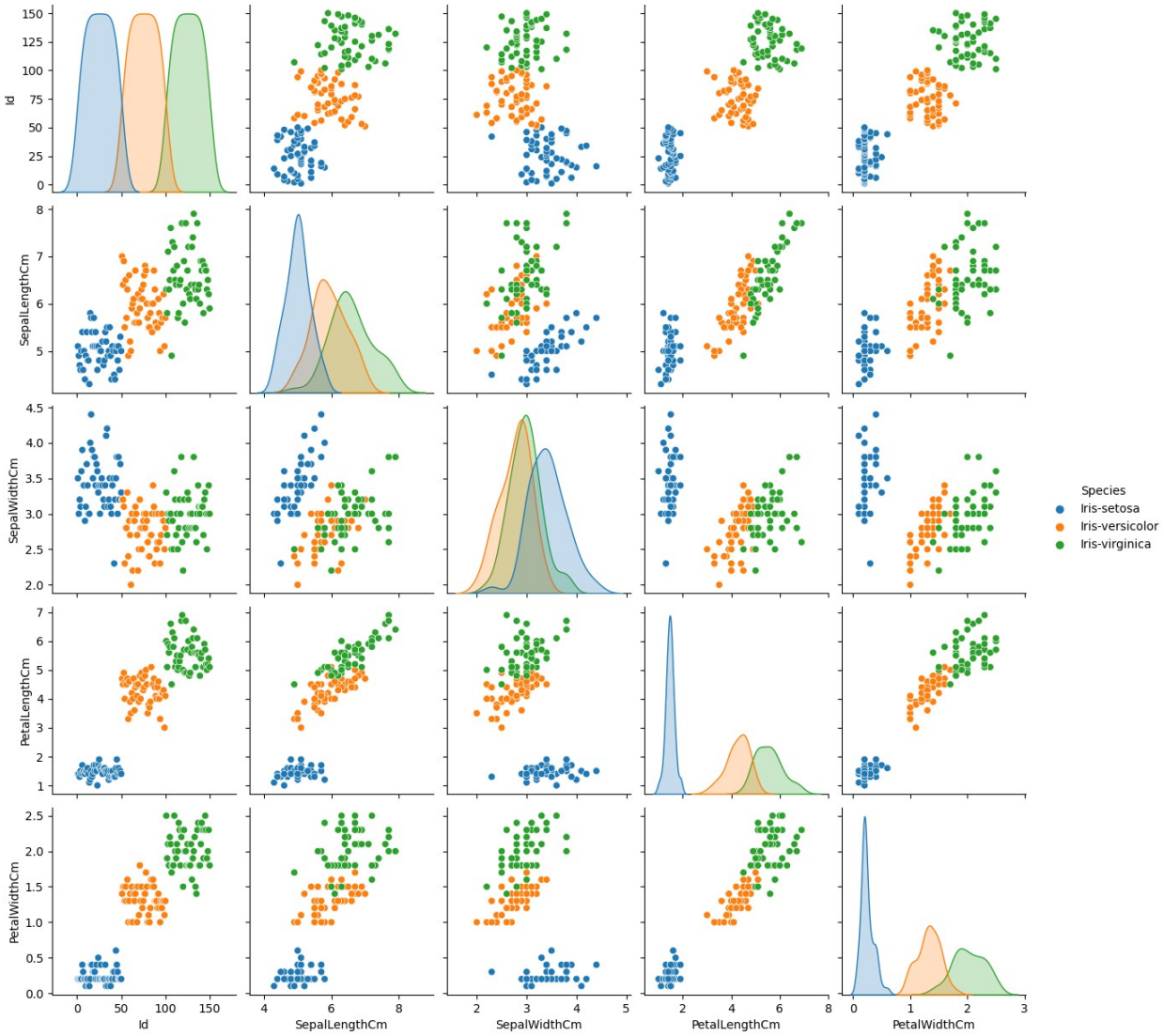



```
sns.pairplot(df, hue='Species')
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
```

```
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
```

```
/usr/local/lib/python3.11/dist-packages/seaborn/axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```




```
plt.figure(figsize=(8,6))
sns.heatmap(df.drop(columns=['Id']).corr(), annot=True,
cmap='coolwarm')
plt.title('Korelasi antar Fitur')
plt.show()

---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
/tmp/ipykernel_35/693317647.py in <cell line: 0>()
```

```
      1 plt.figure(figsize=(8,6))
----> 2 sns.heatmap(df.drop(columns=['Id']).corr(), annot=True,
cmap='coolwarm')
      3 plt.title('Korelasi antar Fitur')
      4 plt.show()

/usr/local/lib/python3.11/dist-packages/pandas/core/frame.py in
corr(self, method, min_periods, numeric_only)
  11047         cols = data.columns
  11048         idx = cols.copy()
> 11049         mat = data.to_numpy(dtype=float, na_value=np.nan,
copy=False)
  11050
  11051         if method == "pearson":

/usr/local/lib/python3.11/dist-packages/pandas/core/frame.py in
to_numpy(self, dtype, copy, na_value)
   1991         if dtype is not None:
   1992             dtype = np.dtype(dtype)
-> 1993         result = self._mgr.as_array(dtype=dtype, copy=copy,
na_value=na_value)
   1994         if result.dtype is not dtype:
   1995             result = np.asarray(result, dtype=dtype)

/usr/local/lib/python3.11/dist-packages/pandas/core/internals/managers
.py in as_array(self, dtype, copy, na_value)
   1692                 arr.flags.writeable = False
   1693         else:
-> 1694             arr = self._interleave(dtype=dtype,
na_value=na_value)
   1695             # The underlying data was copied within
_interleave, so no need
   1696             # to further copy if copy=True or setting na_value

/usr/local/lib/python3.11/dist-packages/pandas/core/internals/managers
.py in _interleave(self, dtype, na_value)
   1751             else:
   1752                 arr = blk.get_values(dtype)
-> 1753             result[rl.indexer] = arr
   1754             itemmask[rl.indexer] = 1
   1755

ValueError: could not convert string to float: 'Iris-setosa'

<Figure size 800x600 with 0 Axes>
```

Preprocessing

```
X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
```

```
'PetalWidthCm']]
y = df['Species']

le = LabelEncoder()
y_encoded = le.fit_transform(y)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled,
y_encoded, test_size=0.2, random_state=42)
```

Definisi Model MLP

```
class IrisMLP(nn.Module):
    def __init__(self):
        super(IrisMLP, self).__init__()
        self.fc1 = nn.Linear(4, 10)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(10, 3)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        return x
```

Training Model MLP

```
X_train_tensor = torch.FloatTensor(X_train)
y_train_tensor = torch.LongTensor(y_train)

train_dataset = TensorDataset(X_train_tensor, y_train_tensor)
train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True)

model = IrisMLP()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)

epochs = 100
losses = []
accuracies = []

for epoch in range(epochs):
    total_loss = 0
    correct = 0
    model.train()
```

```
    for batch_x, batch_y in train_loader:
        optimizer.zero_grad()
        outputs = model(batch_x)
        loss = criterion(outputs, batch_y)
        loss.backward()
        optimizer.step()

        total_loss += loss.item()
        _, predicted = torch.max(outputs, 1)
        correct += (predicted == batch_y).sum().item()

    acc = correct / len(train_dataset)
    losses.append(total_loss)
    accuracies.append(acc)

    if (epoch+1) % 10 == 0:
        print(f"Epoch {epoch+1}, Loss: {total_loss:.4f}, Accuracy:
{acc:.4f}")
```

Visualisasi Loss dan Akurasi

```
plt.plot(losses, label='Loss')
plt.plot(accuracies, label='Accuracy')
plt.xlabel('Epoch')
plt.legend()
plt.title('Loss dan Akurasi Selama Training')
plt.show()
```

Evaluasi Model MLP

```
model.eval()
with torch.no_grad():
    X_test_tensor = torch.FloatTensor(X_test)
    outputs = model(X_test_tensor)
    _, predicted = torch.max(outputs, 1)
    print("Accuracy:", accuracy_score(y_test, predicted))
    print(confusion_matrix(y_test, predicted))
    print(classification_report(y_test, predicted,
target_names=le.classes_))
```

Perbandingan dengan Logistic Regression

```
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred_logreg = logreg.predict(X_test)
```

```
print("Logistic Regression Accuracy:", accuracy_score(y_test,
y_pred_logreg))
print(confusion_matrix(y_test, y_pred_logreg))
print(classification_report(y_test, y_pred_logreg,
target_names=le.classes_))
```

Prediksi Data Baru (Opsional)

```
sample = [[5.8, 2.7, 5.1, 1.9]]
sample_scaled = scaler.transform(sample)
sample_tensor = torch.FloatTensor(sample_scaled)

model.eval()
with torch.no_grad():
    output = model(sample_tensor)
    _, predicted = torch.max(output, 1)
    print("Prediksi:", le.inverse_transform(predicted.numpy()))
```

Analisis dan Kesimpulan

- Model MLP berhasil mencapai akurasi tinggi pada dataset Iris.
- ReLU digunakan sebagai fungsi aktivasi hidden layer untuk memperkenalkan non-linearitas.
- Fungsi loss yang digunakan adalah CrossEntropyLoss, cocok untuk multi-class classification.
- Meskipun Logistic Regression juga tampil baik, MLP memiliki potensi lebih untuk generalisasi jika data lebih kompleks.
- Tantangan di dunia nyata meliputi: kebutuhan data besar, tuning hyperparameter, dan overfitting.

Model ini dapat diterapkan dalam sistem klasifikasi bunga otomatis, seperti aplikasi edukasi botani atau sistem rekomendasi tanaman.