KLASIFIKASI SPESIES IKAN HIU MENGGUNAKAN MODEL MULTI-LAYER PERCEPTION (MLP)

Mata Kuliah: Deep Learning

Dosen Pengampu : Harwikarya, Dr. M.T.

NAMA	: Muhammad Rizky
NIM	: 41522010097

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS MERCU BUANA

2024

Daftar Isi

BAB I
PENDAHULUAN

1. Latar Belakang

Ikan hiu adalah jenis ikan yang termasuk dalam kelompok ikan bertulang rawan (Chondrichthyes), yang berarti tulangnya terbuat dari kartilago (tulang rawan) dan bukan tulang keras seperti ikan pada umumnya. Hiu dikenal sebagai predator laut yang sangat kuat dan memiliki berbagai spesies yang bervariasi, dengan beberapa di antaranya menjadi pemangsa puncak di ekosistem laut. Tujuan Utama dari penelitian ini adalah membangun dan mengevaluasi model klasifikasi menggunakan algoritma Multi-Layer Perceptron (MLP).

1.1 Great White Shark

Great White Shark (*Carcharodon carcharias*) adalah salah satu spesies hiu terbesar dan terkenal sebagai predator puncak di ekosistem laut. Dapat tumbuh hingga panjang 6 hingga 7 meter dan berat lebih dari 1.000 kilogram, mereka memiliki tubuh ramping dengan warna punggung abu-abu gelap dan perut putih yang memberikan kamuflase. Hiu ini memangsa mamalia laut seperti anjing laut, singa laut, dan ikan besar lainnya, menggunakan serangan cepat dari bawah. Indera mereka sangat sensitif, memungkinkan mereka mendeteksi gerakan dan medan listrik makhluk hidup di sekitar mereka. Sebagai spesies vivipar, mereka melahirkan 2 hingga 14 bayi setelah masa kehamilan sekitar 11 bulan. Meskipun jarang menyerang manusia, mereka terancam oleh penangkapan berlebihan dan kerusakan habitat. Sebagai predator puncak, mereka menjaga keseimbangan ekosistem laut dengan memangsa hewan yang lemah atau sakit.

Gambar 1. Great White Shark

1.2 Sand Tiger Shark

Sand Tiger Shark (*Carcharias taurus*) adalah spesies hiu yang dikenal karena penampilannya yang menakutkan dengan gigi besar yang tampak mencuat, meskipun sebenarnya mereka lebih jinak dibandingkan dengan beberapa spesies hiu lainnya. Hiu ini ditemukan di perairan hangat hingga subtropis di seluruh dunia, sering kali di daerah terumbu karang dan pantai dengan kedalaman 10 hingga 50 meter. Mereka memiliki tubuh panjang dan ramping, dengan warna tubuh yang cenderung ke abu-abu kecoklatan atau keabu-abuan dengan perut lebih terang. Sand Tiger Shark adalah predator yang aktif dan umumnya memangsa ikan, invertebrata, dan kadang-kadang mamalia laut kecil. Meskipun memiliki gigi yang tajam dan besar, mereka lebih sering menggunakan kekuatan untuk menangkap mangsa mereka daripada menggigit dengan kekuatan besar seperti beberapa spesies hiu lain. Hiu ini juga dikenal karena perilakunya yang khas, yaitu sering muncul ke permukaan untuk mengambil napas, sehingga sering terlihat berenang mendekati permukaan air. Sebagai spesies vivipar, Sand Tiger Shark melahirkan bayi yang berkembang dalam rahim ibu mereka, dan dalam satu kali kelahiran, mereka dapat melahirkan hingga 2 hingga 4 bayi. Meskipun tampak menakutkan, mereka tidak agresif terhadap manusia dan jarang terlibat dalam insiden serangan. Namun, Sand Tiger Shark terancam oleh penangkapan berlebihan dan kerusakan habitat, yang menyebabkan penurunan jumlah populasinya di beberapa wilayah. Sebagai predator puncak, mereka memainkan peran penting dalam menjaga keseimbangan ekosistem laut.

Gambar 2. Sand Tiger Shark

2. Perumusan Masalah

Masalah Utama yang akan di bahas pada laporan ini adalah bagaimana cara membangun model klasifikasi untuk memprediksi spesies ikan hiu (Great White Shark, Sand Tiger Shark) Berdasarkan 10 fitur. Berikut adalah beberapa aspek penting yang akan dibahas dalam laporan ini:

- Pre-Processing Data: Beberapa hal yang penting yang wajib kita lakukan terlebih dahulu seperti proses normalisasi, dan proses pemisahan label.
- Model MLP: Pemilihan model dan teknik-teknik untuk mencegah overfitting, contohnya Dropout.
- Evaluasi Model: Penggunaan Teknik Evaluasi seperti Confusion Matrix

BAB II
PEMBAHASAN

1. Dataset

Dataset yang digunakan dalam penelitian ini adalah dataset ikan hiu yang mencakup beberapa fitur dari dua spesies ikan hiu: Great White Shark, dan Sand Tiger Shark. Fitur yang dianalisis itu meliputi:

- **Panjang Rata-rata (Average_Length_m)**
- **Panjang Maksimal (Maximum_Length_m)**
- **Berat Rata-rata (Average_Weight_kg)**
- **Berat Maksimal (Maximum_Weight_kg)**
- **Umur (Lifespan_years)**
- **Kecepatan Berenang (Swimming_Speed_kmh)**
- **Kekuatan Gigitan (Bite_Force_N)**
- **Kehamilan (Gestation)**
- **Jumlah Gigi (Number_of_Teeth)**
- **Diet Mingguan (Diet_kg_per_week)**

Species	Average_Length_m	Maximum_Length_m	Average_Weight_kg	Maximum_Weight_kg	Lifespan_years	Swimming_Speed_kmh	Bite_Force_N	Gestation	Number_of_Teeth	Diet_kg_per_week
Great White Shark	4.59	6.81	1019.25	2180.29	71	56.34	17981.54	12	291	190.36
Sand Tiger Shark	2.71	2.98	144.69	327.66	15	21.92	2849.02	8	180	45.51
Great White Shark	4.41	6.35	1090.19	2295.4	70	50.8	16624.95	11	306	191.58
Sand Tiger Shark	3.28	3.03	172.16	303.87	13	22.23	3187.8	9	205	50.71
Great White Shark	4.5	6.69	1084.09	2215.87	68	61.11	18739.49	12	288	196.7

Gambar 3. Contoh Tabel pada dataset yang digunakan

2. Multi-Layer-Perceptron

Multi-Layer Perceptron (MLP) adalah jenis jaringan saraf tiruan (artificial neural network, ANN) yang terdiri dari beberapa lapisan (layer) neuron yang saling terhubung. MLP sering digunakan dalam berbagai aplikasi pembelajaran mesin, termasuk pengenalan pola, prediksi, klasifikasi, dan regresi. Berikut adalah penjelasan elemen-elemen utamanya.

3. Struktur Multi-Layer Perceptron

1. **Input Layer (Lapisan Input)**: Lapisan ini menerima data masukan dari luar sistem. Jumlah neuron di lapisan ini biasanya sesuai dengan jumlah fitur dalam dataset.
2. **Hidden Layer (Lapisan Tersembunyi)**

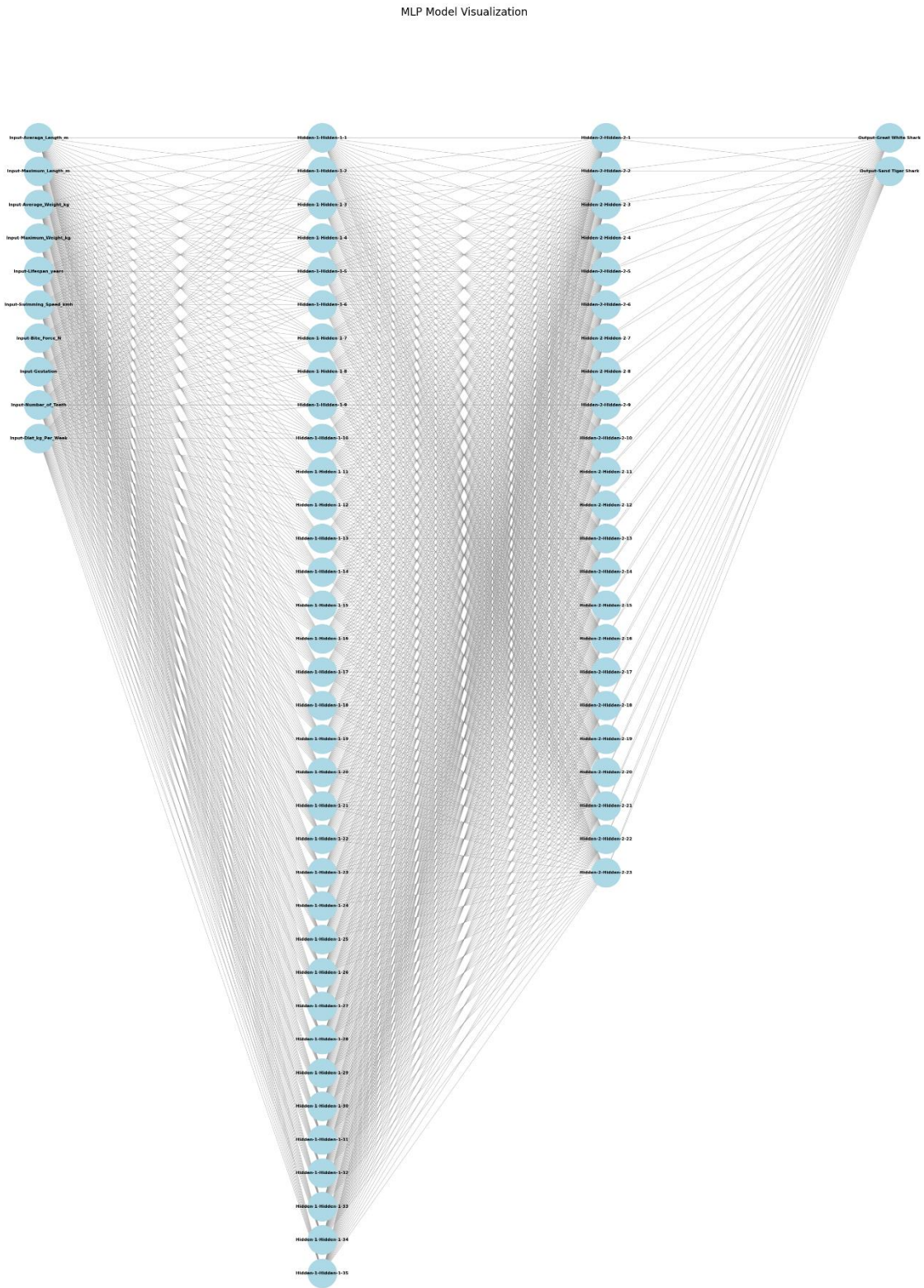
    - Terdiri dari satu atau lebih lapisan yang memproses data masukan menggunakan fungsi aktivasi non-linear seperti ReLU, Sigmoid, atau Tanh.
    - Fungsi ini membantu menangkap hubungan kompleks dalam data.

3. **Output Layer (Lapisan Output)**

    - Lapisan ini menghasilkan hasil akhir, seperti probabilitas klasifikasi atau nilai regresi.
    - Jumlah neuron pada lapisan ini bergantung pada tugas yang ingin diselesaikan, misalnya satu neuron untuk regresi atau beberapa neuron untuk klasifikasi multi-kelas.

4. Diagram Model MLP:

Diagram ini menggambarkan arsitektur model Multi-Layer Perceptron (MLP) yang digunakan untuk klasifikasi spesies Ikan Hiu. Model ini menerima 10 fitur sebagai input: Panjang Rata-rata (Average_Length_m), Panjang Maksimal (Maximum_Length_m), Berat Rata-rata (Average_Weight_kg), Berat Maksimal (Maximum_Weight_kg), Umur (Lifespan_years), Kecepatan Berenang (Swimming_Speed_kmh), Kekuatan Gigitan (Bite_Force_N), Kehamilan (Gestation), Jumlah Gigi (Number_of_Teeth), dan Diet Mingguan (Diet_kg_per_week). Terdapat dua hidden layer dengan 36 dan 24 neuron yang menggunakan fungsi aktivasi ReLU untuk mengolah informasi. Output Layer yang memiliki 2 kelas untuk memprediksi dua spesies ikan hiu, yaitu **Great White Shark,** dan **Sand Tiger Shark**, dengan menggunakan fungsi aktivasi softmax untuk klasifikasi categorical. Model ini dioptimalkan untuk mempelajari hubungan yang terjadi di antara layer input dan output, dengan maksud untuk memprediksi spesies ikan hiu berdasarkan data yang sudah diberikan.



Gambar 4. Visualisasi Dari Model MLP

5. Kalimat Deskriptif

1. Mulai
2. Memasukkan Dataset, bisa berupa Dataset CSV, Gambar, or Noise.
3. Melakukan Pre-processing data sebelum masuk ke algoritma, seperti Encoding, Normalisasi, dan split.
4. Hasil dari data Pre-processing tersebut, kita masukkan bagian yang telah di split itu ke dalam sebuah MLP.
5. Deklarasikan bahwa input layer berisi berapa neuron, hidden layer 1, dan 2 berisi berapa neuron, lalu deklarasikan output layer berisi berapa neuron.
6. Melakukan fiting terhadap data yang sudah di masukkan ke dalam model. Apa yang terjadi di dalam proses fittingnya:
   a. Forward Propagation
      Ini adalah proses data mengalir dari input layer sampai ke Output Layer. Jadi, di dalam proses ini input data yang ada di sebuah neuron itu diberikan sebuah bobot, dan bias yang random untuk iterasi-1, dan untuk masuk ke dalam hidden layer 1, maka akan terjadi perhitungan dengan rumus berikut:

      $z = W. x + b$
      setelah hasil dari z dapat maka akan diterapkan fungsi aktivasinya, di sini saya menggunakan relu, maka:

      $a = \max(0,z)$

      setelah hasilnya sudah di dapat maka itu adalah input dari hidden layer 1, untuk mau masuk ke hidden layer 2, itu prosesnya sama. Perhitungan Bobot, dan bias dengan rumus:

      $z2 = W1. x1 + b1$

      lalu untuk aktivasi menggunakan Relu maka:

      $a = \max(0,z).$

      Hasil dari Aktivasi itu maka akan di masukkan ke dalam Hidden layer 2 sebagai Input.

      Seperti di hidden layer sebelumnya, output dari hidden layer 2 ((a2)) akan dihitung menggunakan bobot ((W2)) dan bias ((b2)) untuk menghasilkan nilai

z3 = w2 . a2 + b2

Di output layer, fungsi aktivasi yang digunakan adalah Softmax (karena biasanya digunakan untuk masalah klasifikasi multi-kelas). Fungsi ini memastikan output berupa probabilitas yang jumlahnya 1. Fungsi Softmax diterapkan untuk menghasilkan output probabilitas untuk setiap kelas:

Softmax(z3) =e^z3/∑e^z3

Hasilnya adalah vektor probabilitas yang menunjukkan kemungkinan kelas dari input data.

b. Back Propagation
Setelah melakukan forward propagation dan menghasilkan output, langkah selanjutnya adalah melakukan backpropagation untuk mengoptimalkan bobot dan bias melalui perhitungan gradien dan pembaruan parameter dengan menggunakan gradient descent atau optimizer lain.

Menghitung Error (Loss):

Untuk mengetahui seberapa baik model melakukan prediksi, kita menghitung loss (kesalahan). Pada model klasifikasi multi-kelas, biasanya digunakan categorical crossentropy sebagai fungsi loss:

$$\text{Loss} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

Di mana:

- y_i adalah nilai aktual untuk kelas i,
- $\hat{y}_i$ adalah prediksi probabilitas untuk kelas i,
- C adalah jumlah kelas.

Menghitung Gradien:

Setelah loss dihitung, backpropagation digunakan untuk menghitung gradien dari loss terhadap setiap bobot dan bias di seluruh jaringan, dengan menggunakan aturan rantai (chain rule) dari kalkulus.

c. Pembaruan Bobot:

Setelah gradien dihitung, bobot dan bias diperbarui untuk meminimalkan loss. Proses pembaruan ini dilakukan dengan rumus:

$$W = W - \eta \cdot \frac{\partial \text{Loss}}{\partial W}$$

$$b = b - \eta \cdot \frac{\partial \text{Loss}}{\partial b}$$

Di mana η adalah **learning rate** yang mengontrol seberapa besar pembaruan yang dilakukan.

Pembaruan bobot dan bias ini dilakukan pada setiap layer mulai dari output layer menuju input layer (proses backward pass). Setiap Proses ini berlangsung dalam satu epoch.

Pada setiap epoch, perhitungan akurasi dilakukan melalui proses **forward propagation** (untuk menghasilkan prediksi), kemudian membandingkan prediksi dengan label yang sebenarnya untuk menghitung jumlah prediksi yang benar. Akurasi dihitung dengan rumus yang sederhana:

$$\text{Accuracy} = \frac{\text{Jumlah prediksi benar}}{\text{Jumlah total data}}$$

Akurasi ini dihitung untuk data pelatihan dan data validasi, yang menunjukkan seberapa baik model mengenali pola dalam data pelatihan dan seberapa baik model dapat mengeneralisasi ke data yang belum dilihat (validasi).

7. Setelah Proses Training Model Selesai Maka kita akan melakukan proses evaluasi model dengan data test yang sudah di split sebelumnya. Hasil dari Evaluasi ini akan menentukan Akurasi dari model yang sudah di buat.
8. Selesai

6. Penjelasan Program

Sel 1: Import Library

```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
```

- **TensorFlow**: Untuk membangun model Multi-Layer Perceptron (MLP) atau deep learning.
- **NumPy**: Untuk manipulasi array.
- **Pandas**: Untuk manipulasi data dalam bentuk tabel (DataFrame).
- **Matplotlib**: Untuk visualisasi data.
- **os**: Untuk operasi file/direktori.
- **to_categorical**: Untuk mengonversi label menjadi representasi kategori (one-hot encoding).
- **train_test_split**: Untuk membagi data menjadi data latih dan uji.
- **StandardScaler** dan **LabelEncoder**: Untuk normalisasi data dan encoding label.

Sel 2: Cek GPU

```
print("GPU is available:", tf.config.list_physical_devices('GPU'))
```

Kode ini memeriksa apakah GPU tersedia untuk mempercepat pelatihan model TensorFlow. Jika GPU ditemukan, hasilnya akan ditampilkan.

Sel 3: Konfigurasi Proses Deep Learning Menggunakan GPU

```
# # CPU Optimizations
# tf.config.experimental.set_virtual_device_configuration(
#     tf.config.experimental.list_physical_devices("CPU")[0],
#     [tf.config.experimental.VirtualDeviceConfiguration()],
# )

 # GPU Memory Limit Optimization
tf.config.set_logical_device_configuration(
    tf.config.list_physical_devices("GPU")[0],
    [tf.config.LogicalDeviceConfiguration(memory_limit=5899)],
)
```

Kode ini mengatur penggunaan GPU dan membatasi memori yang digunakan hingga 5899 MB. Komentar di bagian CPU mengindikasikan opsi untuk mengoptimalkan konfigurasi CPU jika diperlukan.

Sel 4: Load Dataset

```
data = pd.read_csv('Shark_Comparison_Dataset.csv')
```

Kode ini Membuat Agar Kode bisa membaca Dataset

Sel 5: Mencetak Kolom Dataset

```
print(data.columns)
```

Kode ini Mencetak Kolom-Kolom yang ada pada Dataset

Sel 6: Encode Kolom

```
le = LabelEncoder()
data['Species'] = le.fit_transform(data['Species'])
```

Kode ini Membuat Label Species Agar menjadi data numerical yang bisa di baca oleh computer seperti bilangan 0, dan 1 dalam bilangan biner.

Sel 7: Split data menjadi X features dan Label y

```
X = data.drop(columns='Species')
y = data['Species']
```

Kode ini Membuat agar X itu berisi semua fitur kecuali label, sedangkan y hanya berisi label saja

Sel 8: Split Data menjadi train, dan test

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=300 / len(X), random_state=69)
```

Kode ini Akan membagi data menjadi train, dan test. Sehingga yang train itu berisi 1700, dan yang test itu hanya berisi 300 saja.

Sel 9: Normalisasi fitur

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Kode tersebut menggunakan StandardScaler untuk menormalisasi fitur dalam dataset agar memiliki rata-rata (mean) = 0 dan standar deviasi = 1. Pertama, fit_transform diterapkan pada data latih (X_train) untuk menghitung statistik normalisasi (rata-rata dan standar deviasi) berdasarkan data latih, sekaligus mengubah data tersebut ke skala baru. Kemudian, transform diterapkan pada data uji (X_test) menggunakan statistik yang sama dari data latih, memastikan data uji berada dalam skala yang konsisten. Normalisasi ini penting untuk memastikan semua fitur berada dalam skala yang sama, sehingga model dapat belajar dengan lebih stabil dan efektif tanpa bias terhadap fitur dengan rentang nilai yang lebih besar.

Sel 10: Mengubah Label menjadi format to_categorical untuk binary Classification

```
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

Kode ini menggunakan fungsi `to_categorical` dari Keras untuk mengonversi label target (`y_train` dan `y_test`) menjadi representasi **one-hot encoding**. One-hot encoding adalah metode untuk merepresentasikan label kelas dalam bentuk vektor biner, di mana setiap kelas unik direpresentasikan oleh satu elemen bernilai 1 di posisi yang sesuai, sementara elemen lainnya adalah 0. Misalnya, jika ada tiga kelas (0, 1, 2),

label 1 akan direpresentasikan sebagai `[0, 1, 0]`. Langkah ini dilakukan karena banyak model pembelajaran mesin, seperti neural network, membutuhkan label dalam format ini untuk menghitung probabilitas pada output layer (biasanya dengan fungsi aktivasi seperti softmax). Dengan one-hot encoding, model dapat menghitung loss dengan lebih akurat dan memahami bahwa setiap sampel hanya milik satu kelas tertentu.

Sel 11: Pembuatan Model, dan Training Data dengan model tersebut

```
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
from tensorflow.keras.utils import plot_model

# Assuming X_train and y_train are already defined

# Define model
model = Sequential([
    Dense(36, input_dim=X_train.shape[1], activation='relu'),
    Dense(24, activation='relu'),
    Dense(2, activation='softmax')
])

optimizer = Adam(learning_rate=0.00001)
model.compile(optimizer=optimizer, loss='categorical_crossentropy',
metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, epochs=50, batch_size=32,
validation_split=0.2)

# Visualize training history
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title('Model Accuracy')
plt.show()

plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Model Loss')
plt.show()
```

Kode di atas adalah implementasi model neural network dengan Keras untuk klasifikasi, mencakup definisi model, pelatihan, dan visualisasi hasil. Berikut penjelasannya:

1. **Import Pustaka**:
    - matplotlib.pyplot untuk memvisualisasikan hasil pelatihan.
    - Modul dari Keras untuk membangun dan melatih model neural network, termasuk callback untuk pengoptimalan dan visualisasi model.

Definisi Model:

```
model = Sequential([
   Dense(36, input_dim=X_train.shape[1], activation='relu'),
   Dense(24, activation='relu'),
   Dense(2, activation='softmax')
])
```

Model MLP dibuat dengan API Sequential, memiliki:

- **Hidden layer 1**: 36 neuron dengan aktivasi relu, menerima input dengan dimensi yang sesuai dengan fitur data latih.
- **Hidden layer 2**: 24 neuron dengan aktivasi relu.
- **Output layer**: 2 neuron dengan aktivasi softmax, sesuai dengan jumlah kelas target (2 kelas).

Kompilasi Model:

```
optimizer = Adam(learning_rate=0.00001)
model.compile(optimizer=optimizer, loss='categorical_crossentropy',
metrics=['accuracy'])
```

Model dikompilasi menggunakan optimizer Adam dengan learning rate kecil (0.00001), loss categorical_crossentropy untuk klasifikasi multi-kelas, dan metrik akurasi.

Pelatihan Model:

```
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)
```

Model dilatih selama 50 epoch dengan batch size 32, menggunakan 80% data untuk pelatihan dan 20% untuk validasi.

Visualisasi Hasil Pelatihan:

- Akurasi dan loss selama pelatihan dan validasi diplot menggunakan matplotlib untuk memahami performa model.
- Grafik ini membantu mendeteksi masalah seperti **overfitting** atau **underfitting**.

Sel 12: Melihat Model History, dan Summary

```
model.summary(), history.history
```

Fungsi model.summary() digunakan untuk mencetak ringkasan arsitektur model neural network, menampilkan informasi tentang nama lapisan, dimensi keluaran, jumlah parameter pada setiap lapisan, dan total parameter dalam model, yang membantu memahami kompleksitas model. Sementara itu, atribut history.history adalah dictionary yang berisi data hasil pelatihan model, mencakup nilai loss dan akurasi untuk data pelatihan (loss, accuracy) serta validasi (val_loss, val_accuracy) di setiap epoch. Informasi ini berguna untuk menganalisis performa model, membantu mendeteksi masalah seperti overfitting atau underfitting, serta untuk membuat visualisasi seperti grafik akurasi dan loss untuk evaluasi lebih lanjut.

Sel 13: membuat prediksi dengan model neural network pada data uji (X_test) dan menghitung akurasinya

```
# Prediksi model
predictions = model.predict(X_test)
predicted_classes = (predictions > 0.5).astype(int)  # Jika model output-nya probabilitas

# Jika y_test dalam format one-hot encoded, ambil argmax untuk label yang diprediksi
y_test_labels = np.argmax(y_test, axis=1)
predicted_labels = np.argmax(predicted_classes, axis=1)

# Hitung akurasi
correct_predictions = np.sum(predicted_labels == y_test_labels)
accuracy = correct_predictions / len(y_test) * 100

print(f"Number of correct predictions: {correct_predictions}")
print(f"Accuracy: {accuracy:.2f}%")
```

Kode tersebut digunakan untuk membuat prediksi pada data uji (X_test) dan menghitung akurasi model. Model menghasilkan output berupa probabilitas untuk setiap kelas target, yang kemudian diubah menjadi kelas diskret menggunakan ambang batas 0.5. Jika data label (y_test) dan prediksi berbentuk one-hot encoding, np.argmax digunakan untuk mengonversinya ke label numerik asli. Akurasi dihitung dengan membandingkan jumlah prediksi yang benar dengan total data uji, lalu dinyatakan dalam persentase. Langkah ini penting untuk mengevaluasi performa model dalam memprediksi data baru, dengan akurasi sebagai metrik sederhana namun efektif untuk menilai hasil klasifikasi.

Sel 14: Melihat Hasil dari evaluasi model

```
# Evaluating the model on test data
results = model.evaluate(X_test, y_test, verbose=0)
print('Test loss, Test accuracy:', results)
```

Kode tersebut digunakan melihat Test loss, dan Test Accuracy

Sel 15: Confusion Matrix

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Generate predictions
predicted_classes = model.predict(X_test)
predicted_classes = np.argmax(predicted_classes, axis=1)

# Get true labels
true_labels = np.argmax(y_test, axis=1)

# Create confusion matrix
cm = confusion_matrix(true_labels, predicted_classes)

# Plot the confusion matrix using seaborn heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=le.classes_, yticklabels=le.classes_)
plt.title('Confusion Matrix of Model Predictions')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

Kode ini berfungsi untuk mengenerate confusion matrix berdasarkan prediksi yang telah dilakukan.

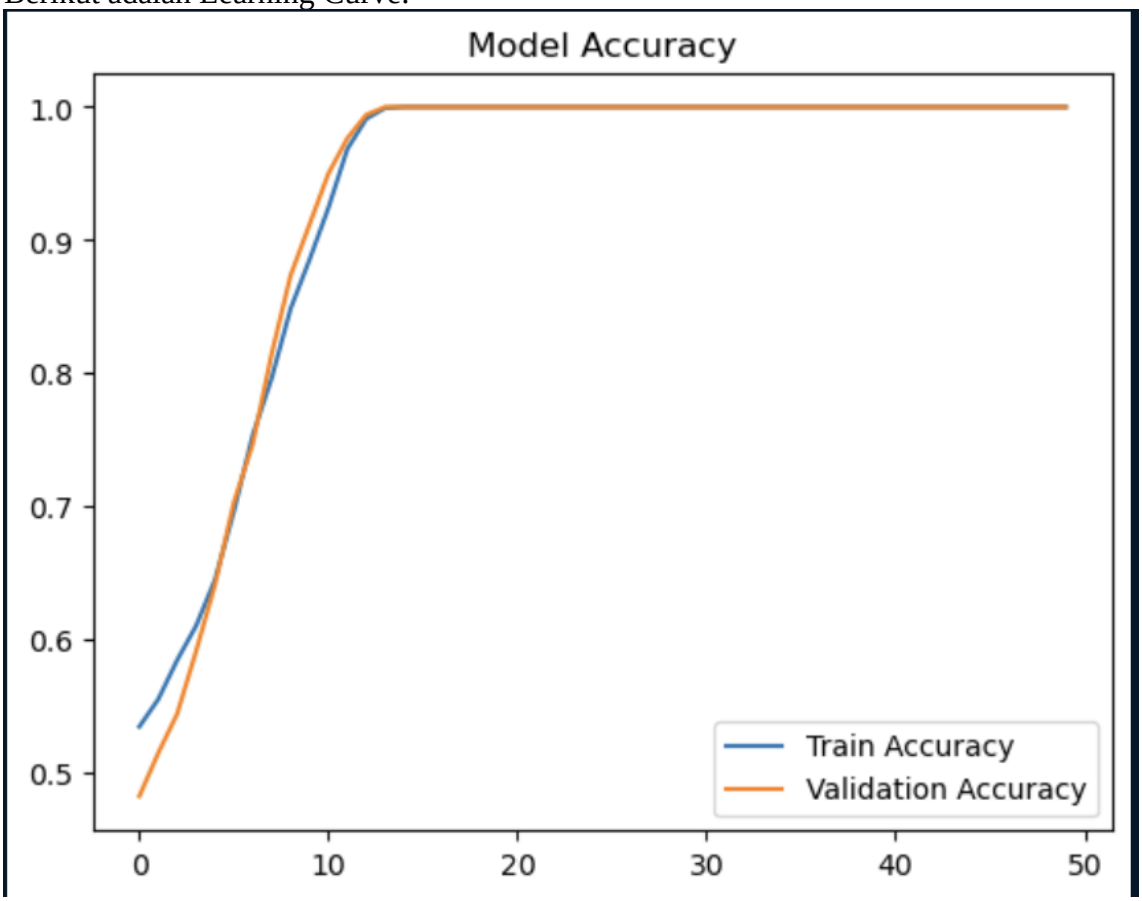
7. Output Program:

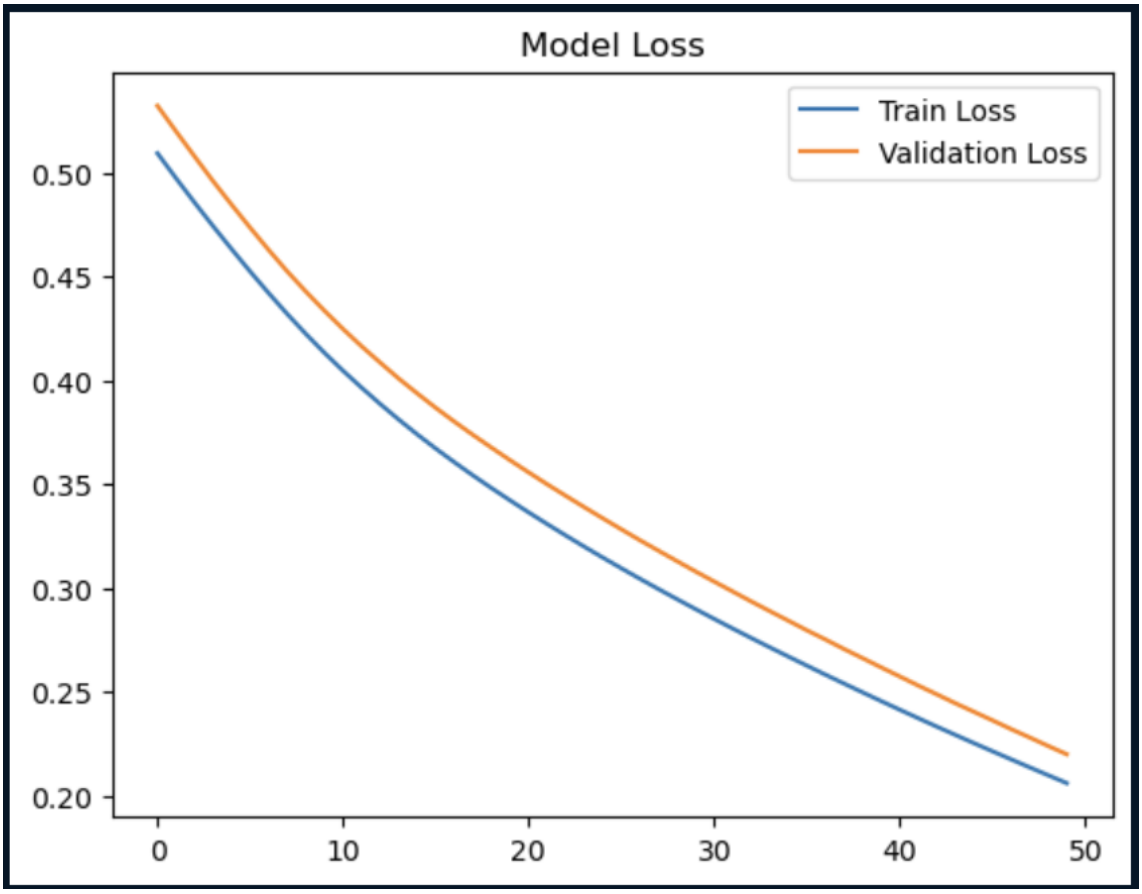
1. Berikut adalah proses Training Program yang di dapat ketika fitting terjadi:

```
Epoch 1/50
43/43 [==============================] - 3s 9ms/step - loss: 0.5094 - accuracy: 0.5346 - val_loss: 0.5323 - val_accuracy: 0.4824
Epoch 2/50
43/43 [==============================] - 0s 3ms/step - loss: 0.4974 - accuracy: 0.5551 - val_loss: 0.5200 - val_accuracy: 0.5147
Epoch 3/50
43/43 [==============================] - 0s 3ms/step - loss: 0.4857 - accuracy: 0.5846 - val_loss: 0.5079 - val_accuracy: 0.5441
Epoch 4/50
43/43 [==============================] - 0s 3ms/step - loss: 0.4743 - accuracy: 0.6103 - val_loss: 0.4960 - val_accuracy: 0.5912
Epoch 5/50
43/43 [==============================] - 0s 3ms/step - loss: 0.4632 - accuracy: 0.6449 - val_loss: 0.4846 - val_accuracy: 0.6412
Epoch 6/50
43/43 [==============================] - 0s 4ms/step - loss: 0.4524 - accuracy: 0.6963 - val_loss: 0.4735 - val_accuracy: 0.7029
Epoch 7/50
43/43 [==============================] - 0s 3ms/step - loss: 0.4420 - accuracy: 0.7537 - val_loss: 0.4628 - val_accuracy: 0.7471
Epoch 8/50
43/43 [==============================] - 0s 3ms/step - loss: 0.4319 - accuracy: 0.7963 - val_loss: 0.4525 - val_accuracy: 0.8147
Epoch 9/50
43/43 [==============================] - 0s 3ms/step - loss: 0.4224 - accuracy: 0.8485 - val_loss: 0.4428 - val_accuracy: 0.8735
Epoch 10/50
43/43 [==============================] - 0s 3ms/step - loss: 0.4133 - accuracy: 0.8853 - val_loss: 0.4335 - val_accuracy: 0.9118
Epoch 11/50
43/43 [==============================] - 0s 3ms/step - loss: 0.4047 - accuracy: 0.9243 - val_loss: 0.4248 - val_accuracy: 0.9500
Epoch 12/50
43/43 [==============================] - 0s 4ms/step - loss: 0.3965 - accuracy: 0.9684 - val_loss: 0.4164 - val_accuracy: 0.9765
Epoch 13/50
...
Epoch 49/50
43/43 [==============================] - 0s 4ms/step - loss: 0.2097 - accuracy: 1.0000 - val_loss: 0.2238 - val_accuracy: 1.0000
Epoch 50/50
43/43 [==============================] - 0s 3ms/step - loss: 0.2059 - accuracy: 1.0000 - val_loss: 0.2198 - val_accuracy: 1.0000
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Bisa di lihat nilai akurasi yang perlahan-lahan naik menjadi 1, dan nilai loss yang sedikit demi sedikit berkurang menjadi ketika 50 epoch maka loss yang terakhir adalah 0,2059 yang berarti itu adalah hasil paling optimal dari model tersebut.

2. Berikut adalah Learning Curve:





Kedua Learning Curve ini memiliki maksud bahwa model telah berhasil dalam berlatih yang bisa di lihat karena model accuracy, dan model loss yang perlahan-lahan mendekati satu sama lain dan sesama menurun.

3. Berikut adalah hasil dari history, dan summary dari pelatihan model:
Model: "sequential"

_____
_
Layer (type) Output Shape Param #
===
dense (Dense) (None, 36) 396

dense_1 (Dense) (None, 24) 888

dense_2 (Dense) (None, 2) 50

===
Total params: 1,334
Trainable params: 1,334
Non-trainable params: 0

```
_ (None,
 {'loss': [0.5094438195228577,
   0.49741968512535095,
   0.48569151759147644,
   0.47429725527763367,
   0.46322083473205566,
   0.4524175226688385,
   0.44198694825172424,
   0.43192991614341736,
   0.42236360907554626,
   0.4132729470729828,
   0.40465280413627625,
   0.39649587869644165,
   0.3886411786079407,
   0.3811361491680145,
   0.3739619851112366,
   0.36711788177490234,
   0.3605577349662781,
   0.3542180359363556,
   0.3481147289276123,
   0.34218326210975647,
   0.336402028799057,
   0.33076754212379456,
   0.32526829838752747,
   0.3198695778846741,
   0.3146055340766907,
   0.3094463050365448,
   0.3043957054615021,
   0.2994280457496643,
   0.2945614457130432,
   0.28976768255233765,
   0.2850625813007355,
   0.2804112136363983,
   0.2758404612541199,
   0.2713389992713928,
   0.26690033078193665,
   0.26252150535583496,
   0.2581593692302704,
   0.25389450788497925,
   0.2496458888053894,
   0.2454521656036377,
   0.2412944734096527,
   0.2372036874294281,
   0.233133926987648,
   0.2291482836008072,
   0.22517520189285278,
   0.2212640941143036,
```

0.21738536655902863,
0.21352581679821014,
0.20971477031707764,
0.2059294581413269],
'accuracy': [0.5345588326454163,
0.5551470518112183,
0.5845588445663452,
0.6102941036224365,
0.6448529362678528,
0.6963235139846802,
0.7536764740943909,
0.7963235378265381,
0.8485293984413147,
0.8852941393852234,
0.9242647290229797,
0.9683823585510254,
0.9911764860153198,
0.9992647171020508,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,

1.0,
1.0,
1.0,
1.0,
1.0,
1.0],
'val_loss': [0.532338559627533,
0.5200012922286987,
0.5079104900360107,
0.4959828853607178,
0.48460638523101807,
0.47353672981262207,
0.46279066801071167,
0.4524664878845215,
0.44277459383010864,
0.43354886770248413,
0.42477157711982727,
0.4164276123046875,
0.4085864722728729,
0.4008863568305969,
0.3937135636806488,
0.386741578578949,
0.38010892271995544,
0.37373700737953186,
0.3675881624221802,
0.3615317642688751,
0.3556874990463257,
0.34994202852249146,
0.3443881571292877,
0.3389071822166443,
0.3335198760032654,
0.3282192349433899,
0.3229708671569824,
0.31792423129081726,
0.3128856122493744,
0.30790671706199646,
0.3030512034893036,
0.2982582449913025,
0.2934642434120178,
0.28879478573799133,
0.28418800234794617,
0.2795429825782776,
0.27511584758758545,
0.2705265283584595,
0.26612910628318787,
0.2616892457008362,
0.2573172152042389,
0.25293734669685364,

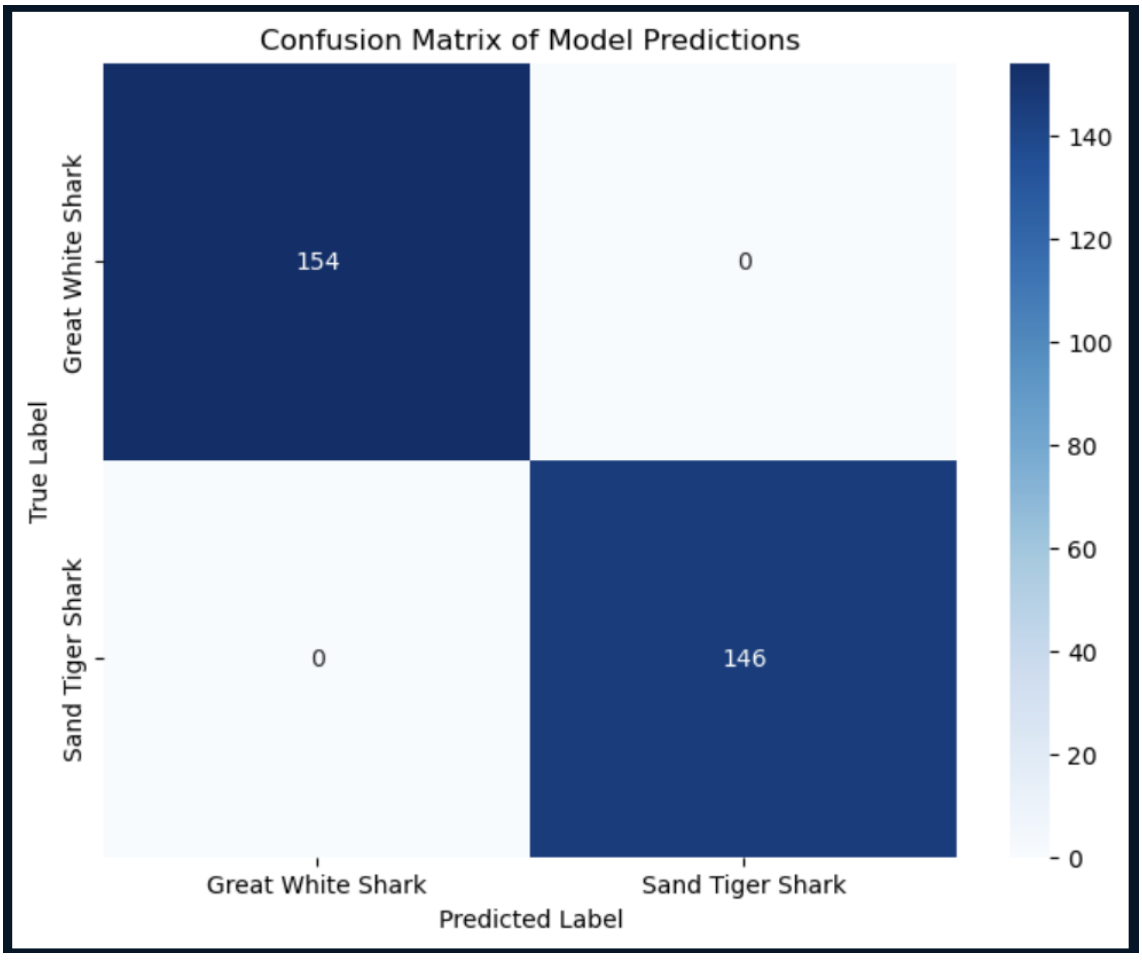
0.24869629740715027,
0.2443941831588745,
0.2402842789888382,
0.23616285622119904,
0.23195737600326538,
0.2279273271560669,
0.2238159477710724,
0.21981878578662872],
'val_accuracy': [0.48235294222831726,
0.5147058963775635,
0.5441176295280457,
0.591176450252533,
0.6411764621734619,
0.7029411792755127,
0.7470588088035583,
0.8147059082984924,
0.8735294342041016,
0.9117646813392639,
0.949999988079071,
0.9764705896377563,
0.9941176176071167,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,

1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0,
1.0]})

4. Hasil Test Prediksi, dan Evaluasi Model:

```
...   10/10 [==============================] - 0s 1ms/step
      Number of correct predictions: 300
      Accuracy: 100.00%
```

5. Hasil Dari Confusion Matrix



BAB III
PENUTUP

Kesimpulan

Model Multi-Layer Perceptron (MLP) yang dirancang untuk menganalisis dataset ini memanfaatkan berbagai fitur numerik yang menggambarkan karakteristik biologis dan fisik dari spesies hiu, seperti panjang rata-rata, berat maksimum, kecepatan renang, kekuatan gigitan, jumlah gigi, dan lainnya. Fitur-fitur tersebut terlebih dahulu melalui proses normalisasi untuk memastikan bahwa mereka berada dalam skala yang sama, sehingga model dapat belajar dengan lebih efektif tanpa dipengaruhi oleh perbedaan skala antara fitur-fitur yang ada. Selain itu, langkah encoding diterapkan pada target kelas untuk memastikan bahwa label spesies dapat dipahami oleh model dalam format one-hot encoding, yang memungkinkan model untuk mengklasifikasikan data secara lebih akurat.

Arsitektur model ini terdiri dari beberapa lapisan dense yang masing-masing diikuti oleh fungsi aktivasi non-linear seperti ReLU, yang memungkinkan model untuk menangkap hubungan yang lebih kompleks antara fitur input dan target kelas. Pada lapisan terakhir, fungsi aktivasi softmax digunakan untuk memberikan probabilitas untuk setiap kelas target, memungkinkan model untuk memilih spesies hiu dengan probabilitas tertinggi sebagai prediksi.
Hasil evaluasi model menunjukkan tingkat akurasi yang menggambarkan seberapa baik model dapat mengenali pola-pola yang membedakan spesies hiu berdasarkan karakteristik yang diberikan dalam dataset. Keakuratan ini mencerminkan kemampuan model dalam memahami dan memetakan data fitur ke dalam kelas yang tepat, yang sangat penting untuk tugas klasifikasi seperti ini. Dengan menggunakan teknik preprocessing yang tepat, seperti normalisasi dan encoding, serta membangun model dengan arsitektur yang sesuai, performa optimal dapat tercapai dalam mengklasifikasikan data hiu.

Secara keseluruhan, penggunaan algoritma MLP dalam pengklasifikasian spesies hiu ini memberikan hasil yang menjanjikan, dengan model yang mampu belajar dan memprediksi spesies berdasarkan berbagai atribut fisik yang ada dalam dataset. Hasil ini dapat digunakan untuk analisis lebih lanjut atau diterapkan dalam aplikasi yang membutuhkan identifikasi spesies hiu secara otomatis, seperti dalam penelitian ekologi laut atau sistem pengawasan kelautan.

Daftar Pustaka

Lorencin, I., Anđelić, N., Španjol, J., & Car, Z. (2020). Using multi-layer perceptron with Laplacian edge detector for bladder cancer diagnosis. *Artificial Intelligence in Medicine*, *102*. https://doi.org/10.1016/j.artmed.2019.101746

Moustafa, E. B., & Elsheikh, A. (2023). Predicting Characteristics of Dissimilar Laser Welded Polymeric Joints Using a Multi-Layer Perceptrons Model Coupled with Archimedes Optimizer. *Polymers*, *15*(1). https://doi.org/10.3390/polym15010233

Pang, B., Nijkamp, E., & Wu, Y. N. (2020). Deep Learning With TensorFlow: A Review. In *Journal of Educational and Behavioral Statistics* (Vol. 45, Issue 2). https://doi.org/10.3102/1076998619872761

Tutorials Point. (2019). TensorFlow Tutorial. *Tutorials Point (I) Pvt. Ltd.*
Vujović, Ž. (2021). Classification Model Evaluation Metrics. *International Journal of Advanced Computer Science and Applications*, *12*(6). https://doi.org/10.14569/IJACSA.2021.0120670