In [23]:
```
# NAME = KRISHAN KUMAR RAI
# ROLL NO. = 231030029
# FOLLOWING CODE USES LEAKY RELU TO AVOID DYING RELU PROBLEM AND BATCH NORMALIZA
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder, PolynomialFeatur
from sklearn.metrics import accuracy_score, classification_report, confusion_mat
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization, LeakyReL
from scikeras.wrappers import KerasClassifier  # Use scikeras instead of tensorf
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.regularizers import l2
import numpy as np
import re

df = pd.read_csv('raveling_image_features_mahotas.csv')

df = df.sample(frac=1, random_state=42).reset_index(drop=True)

X = df.iloc[:, :-1]
y = df.iloc[:, -1]
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2, ra
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
def create_ann_model(optimizer='adam', dropout_rate=0.4, l2_reg=0.01):
    model = Sequential()
    model.add(Dense(256, input_dim=X_train_scaled.shape[1], kernel_regularizer=l
    model.add(LeakyReLU(alpha=0.1))
    model.add(BatchNormalization())
    model.add(Dropout(dropout_rate))

    model.add(Dense(128, kernel_regularizer=l2(l2_reg)))
    model.add(LeakyReLU(alpha=0.1))
    model.add(BatchNormalization())
    model.add(Dropout(dropout_rate))

    model.add(Dense(64, kernel_regularizer=l2(l2_reg)))
    model.add(LeakyReLU(alpha=0.1))
    model.add(BatchNormalization())
    model.add(Dropout(dropout_rate))

    model.add(Dense(32, kernel_regularizer=l2(l2_reg)))
    model.add(LeakyReLU(alpha=0.1))
    model.add(Dropout(dropout_rate))

    model.add(Dense(1, activation='sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['acc
    return model

model = KerasClassifier(model=create_ann_model, verbose=0)
param_grid = {
```

```
    'batch_size': [32, 64],
    'epochs': [100, 150],
    'optimizer': ['adam', 'nadam'],
    'model__dropout_rate': [0.3, 0.4, 0.5],
    'model__l2_reg': [0.001, 0.01, 0.1]
}
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, scoring='f1',
grid_search.fit(X_train_scaled, y_train, callbacks=[reduce_lr])
best_ann_model = grid_search.best_estimator_
print(f"Best Hyperparameters: {grid_search.best_params_}")
y_pred_prob = best_ann_model.predict_proba(X_test_scaled)[:, 1]
precision, recall, thresholds = precision_recall_curve(y_test, y_pred_prob)
f1_scores = 2 * (precision * recall) / (precision + recall)
optimal_threshold = thresholds[np.argmax(f1_scores)]
print(f"Optimal Threshold: {optimal_threshold}")
y_pred = (y_pred_prob >= optimal_threshold).astype(int)
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print(f"Accuracy of the model: {accuracy:.4f}")
print(f"F1 Score of the model: {f1:.4f}")
print("Classification Report:")
print(classification_report(y_test, y_pred))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
data_test = pd.read_csv('test-raveling_image_features_mahotas.csv')

def natural_key(string):
    return [int(text) if text.isdigit() else text.lower() for text in re.split(r

data_test['filename'] = data_test['filename'].apply(str)
data_test_sorted = data_test.sort_values(by='filename', key=lambda x: x.map(natu

X_test_features = data_test_sorted.iloc[:, 1:].values
X_test_poly = poly.transform(X_test_features)
X_test_scaled = scaler.transform(X_test_poly)
y_test_pred = (best_ann_model.predict_proba(X_test_scaled)[:, 1] > optimal_thres

output_df = pd.DataFrame({
    'filename': data_test_sorted['filename'].values,
    'class': y_test_pred.flatten()
})

output_df['class'] = output_df['class'].map({1: 'Raveling', 0: 'Non_raveling'})
output_csv_path = 'Assignment_ANN_02.csv'
output_df.to_csv(output_csv_path, index=False)

print(f"Predictions saved to {output_csv_path}")
```

```
C:\Users\krishan\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib
\externals\loky\process_executor.py:752: UserWarning: A worker stopped while some
jobs were given to the executor. This can be caused by a too short worker timeout
or by a memory leak.
  warnings.warn(
C:\Users\krishan\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras
\src\layers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Input(shap
e)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\krishan\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras
\src\layers\activations\leaky_relu.py:41: UserWarning: Argument `alpha` is deprec
ated. Use `negative_slope` instead.
  warnings.warn(
Best Hyperparameters: {'batch_size': 64, 'epochs': 150, 'model__dropout_rate': 0.
3, 'model__l2_reg': 0.001, 'optimizer': 'nadam'}
Optimal Threshold: 0.8071919083595276
Accuracy of the model: 0.9286
F1 Score of the model: 0.9359
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.92      0.92        62
           1       0.94      0.94      0.94        78

    accuracy                           0.93       140
   macro avg       0.93      0.93      0.93       140
weighted avg       0.93      0.93      0.93       140

Confusion Matrix:
[[57  5]
 [ 5 73]]
C:\Users\krishan\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklear
n\base.py:493: UserWarning: X does not have valid feature names, but PolynomialFe
atures was fitted with feature names
  warnings.warn(
Predictions saved to Assignment_ANN_02.csv
```

In [ ]: