Centro Universitario de Ciencias Exactas e Ingenierías

Ingeniería en Computación
Proyecto final

PRESENTA:

Ramirez Gutierrez Hugo Vladimir
Código: 220287144
Materia:

Seminario de Solución de problemas de Inteligencia Artificial 2

Docente: Diego Campos Pena

Índice:

Introducción:

El código proporcionado implementa un proceso de evaluación de varios modelos de clasificación utilizando diferentes algoritmos de aprendizaje automático. El objetivo principal es clasificar animales en categorías específicas utilizando el conjunto de datos "zoo.csv".

Desarrollo:

Contenido del Código:

1. Importación de Librerías:
   - Se importan varias bibliotecas necesarias para la manipulación de datos, entrenamiento de modelos, evaluación de modelos y visualización de resultados. Estas incluyen `numpy`, `pandas`, `scikit-learn` para algoritmos de aprendizaje automático, `matplotlib` y `seaborn` para visualización de datos.

2. Función zoo():
   - Esta función carga el conjunto de datos "zoo.csv" y divide los datos en conjuntos de entrenamiento y prueba utilizando la función `train_test_split()` de scikit-learn.

3. Funciones de Evaluación de Modelos:
   - Se definen varias funciones para evaluar diferentes modelos de clasificación:
     - Regresión Logística (`logistic_Regression`)
     - K-Nearest Neighbors (`k_Nearest_Neighbors`)
     - Support Vector Machine (`support_Vector_Machine`)
     - Naive Bayes (`naive_Bayes`)
     - MLP (Multilayer Perceptron) (`MLP`)
   - Estas funciones entrenan el modelo correspondiente, realizan predicciones, calculan métricas de evaluación como precisión, recall, F1-score y especificidad, y visualizan la matriz de confusión y las métricas de evaluación.

4. Función evaluate_model():
   - Esta función generaliza el proceso de evaluación de modelos al aceptar cualquier modelo y realizar las mismas operaciones que las funciones de evaluación de modelos específicos.

5. Evaluación de los Modelos:

- Se evalúan cinco modelos de clasificación: Regresión Logística, K-Nearest Neighbors, Support Vector Machine, Naive Bayes y MLP.
- Cada modelo se entrena y se evalúa utilizando los conjuntos de datos de entrenamiento y prueba.
- Se muestran la matriz de confusión y las métricas de evaluación para cada modelo.

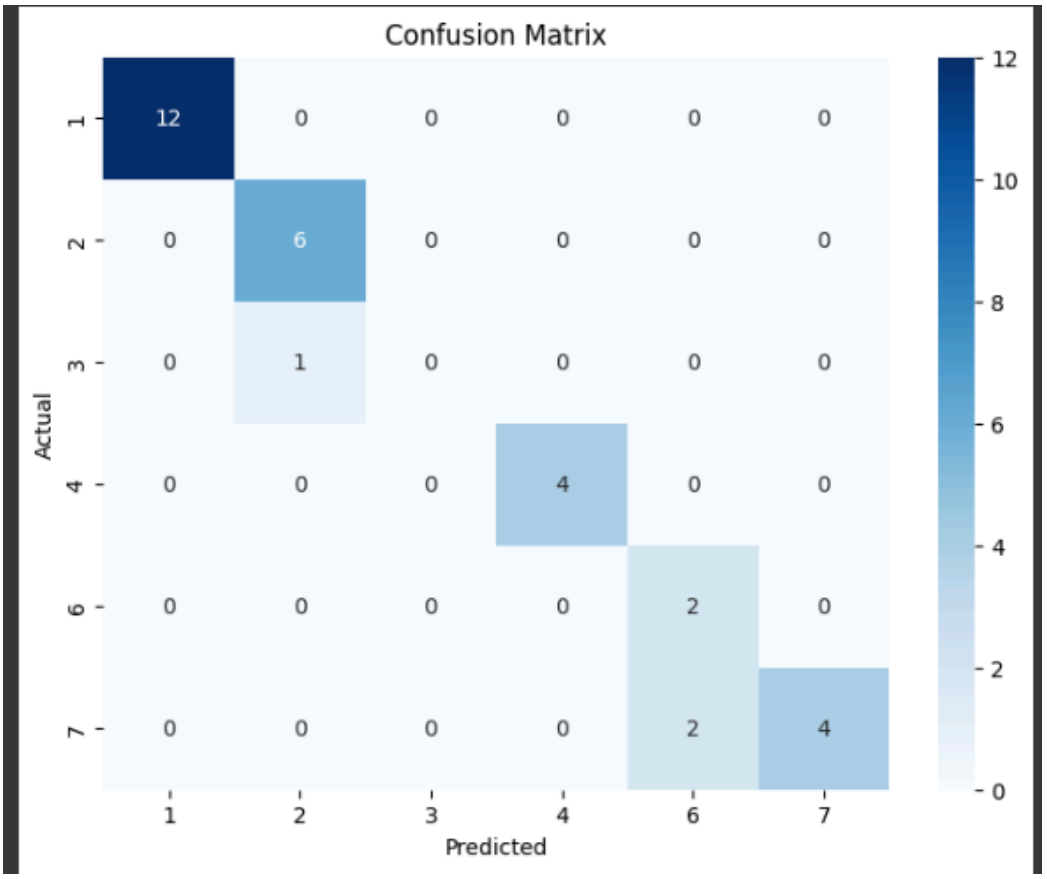
Código:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
from sklearn.metrics import confusion_matrix

def zoo():
    """
    Carga el conjunto de datos 'zoo.csv', divide los datos en conjuntos
de entrenamiento y prueba
    y devuelve los conjuntos de datos divididos.
    """
    dataset = pd.read_csv('zoo.csv')  # Cargar el conjunto de datos
    X = dataset.drop(['animal_name','type'], axis=1)  # Características
    y = dataset['type']  # Etiquetas
    # Dividir los datos en conjuntos de entrenamiento y prueba
    X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=0.7)
    return X_train, X_test, y_train, y_test

def evaluate_model(model, X_train, X_test, y_train, y_test):
    """
    Evalúa el modelo dado y muestra la matriz de confusión y métricas
de evaluación.
    """
    model.fit(X_train, y_train)  # Entrenar el modelo
    y_pred = model.predict(X_test)  # Predecir en el conjunto de prueba

    # Calcular métricas de evaluación
    accuracy = accuracy_score(y_test, y_pred)
```

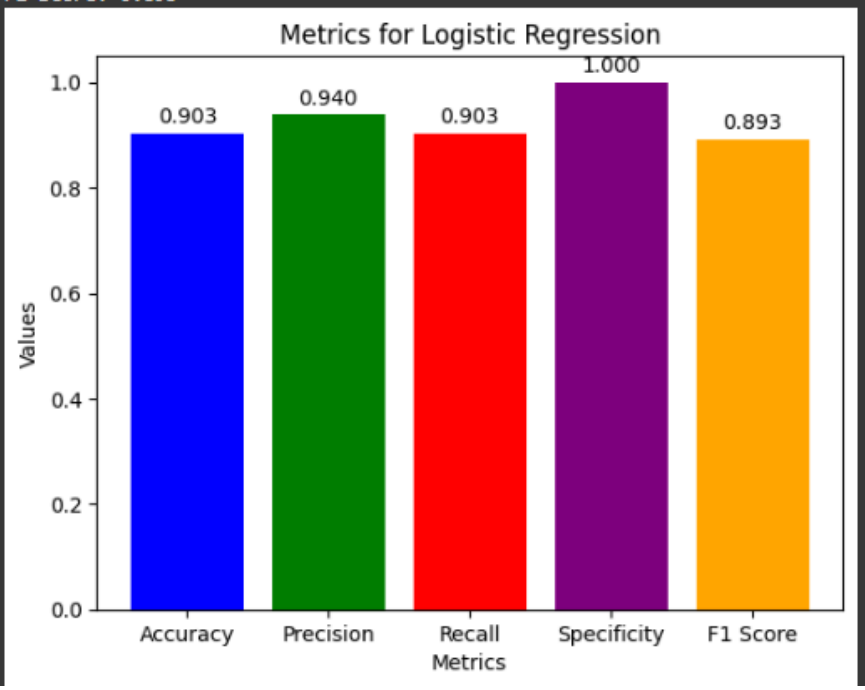
```
    precision = precision_score(y_test, y_pred, average='weighted',
zero_division=1)
    recall = recall_score(y_test, y_pred, average='weighted',
zero_division=1)
    f1 = f1_score(y_test, y_pred, average='weighted')
    confu_matrix = confusion_matrix(y_test, y_pred)
    tn, fp, fn, tp = confu_matrix.ravel()[:4]
    specificity = tn / (tn + fp)

    # Visualizar la matriz de confusión
    plt.figure(figsize=(8, 6))
    sns.heatmap(confu_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=np.unique(y_test), yticklabels=np.unique(y_test))
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

    # Mostrar métricas de evaluación
    print(f"-------- {type(model).__name__} --------\n")
    print(f"Accuracy: {accuracy:.3f}")
    print(f"Precision: {precision:.3f}")
    print(f"Recall: {recall:.3f}")
    print(f"Specificity: {specificity:.3f}")
    print(f"F1 Score: {f1:.3f}")

    # Graficar las métricas
    metrics_names = ['Accuracy', 'Precision', 'Recall', 'Specifity',
'F1 Score']
    metrics_values = [accuracy, precision, recall, specificity, f1]
    plt.bar(metrics_names, metrics_values, color=['blue', 'green',
'red', 'purple', 'orange'])
    plt.title(f'Metrics for {type(model).__name__}')
    plt.xlabel('Metrics')
    plt.ylabel('Values')
    for i, value in enumerate(metrics_values):
        plt.text(i, value + 0.01, f'{value:.3f}', ha='center',
va='bottom')
    plt.show()

# Evaluar los clasificadores
X_train, X_test, y_train, y_test = zoo()  # Obtener conjuntos de
entrenamiento y prueba
classifiers = [LogisticRegression(max_iter=10000),
               KNeighborsClassifier(n_neighbors=5),
               SVC(C=1.0),
               GaussianNB(),
               MLPClassifier(hidden_layer_sizes=(16,16),
max_iter=1000)]

for classifier in classifiers:
```

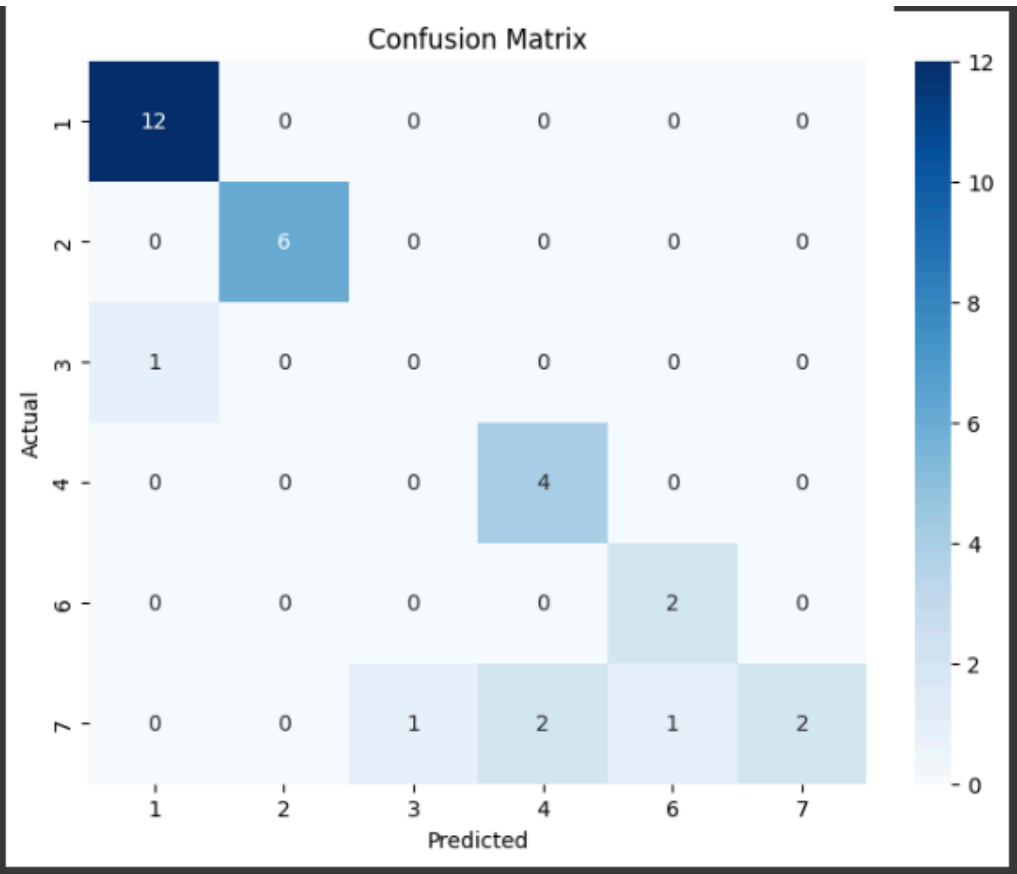
```
evaluate_model(classifier, X_train, X_test, y_train, y_test)
```

Resultados:

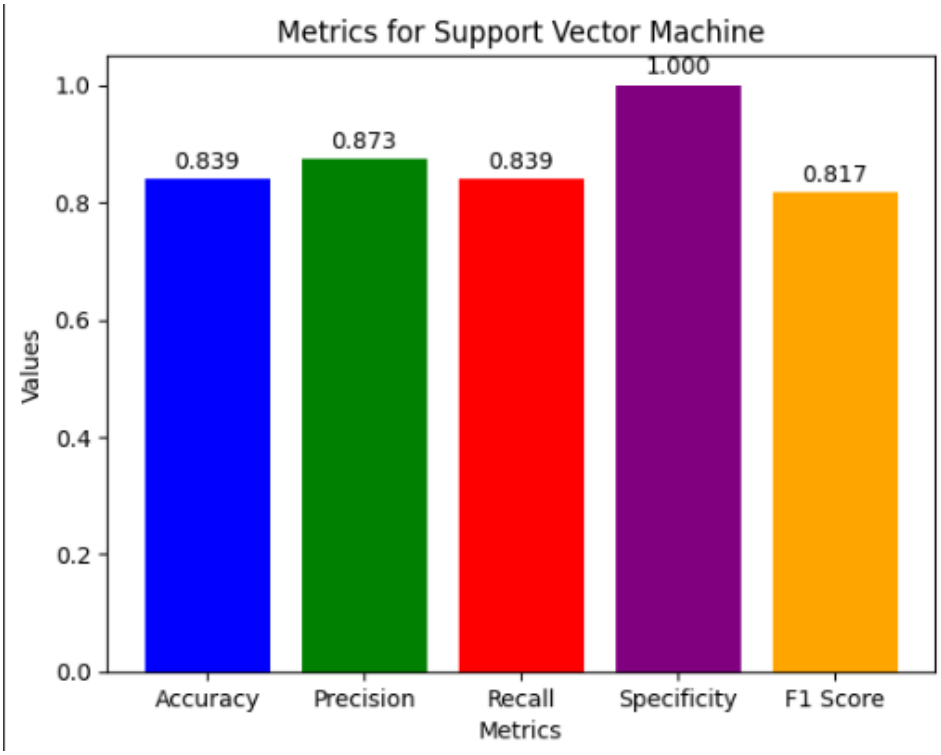


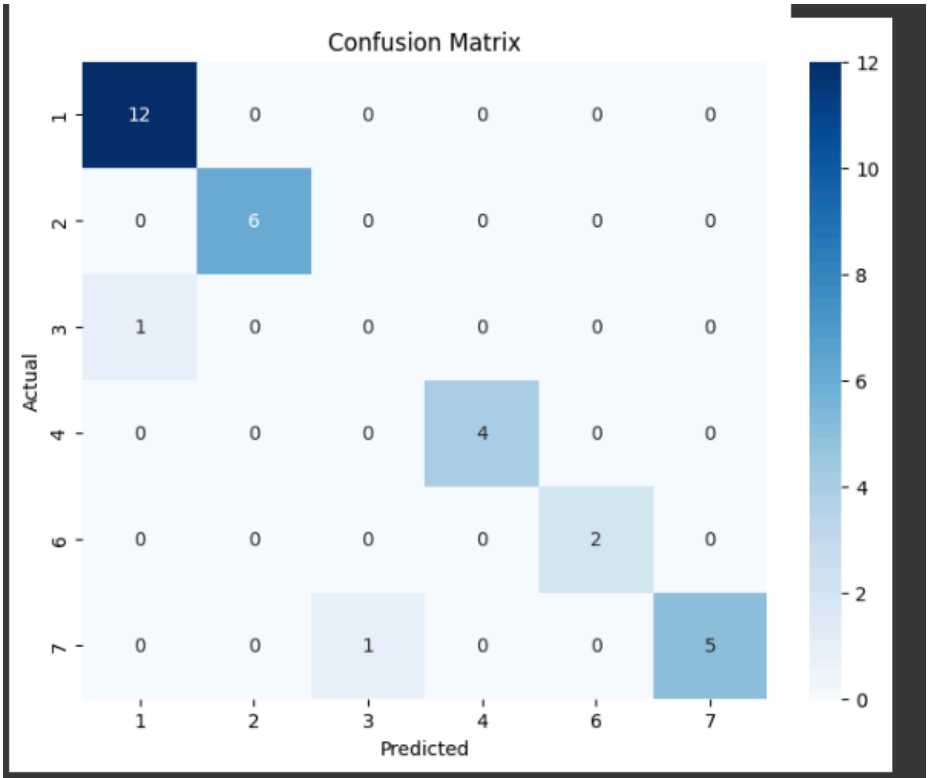
```
-------- Logistic Regression Evaluation --------

Accuracy: 0.903
Precision: 0.940
Recall: 0.903
Specificity: 1.000
F1 Score: 0.893
```



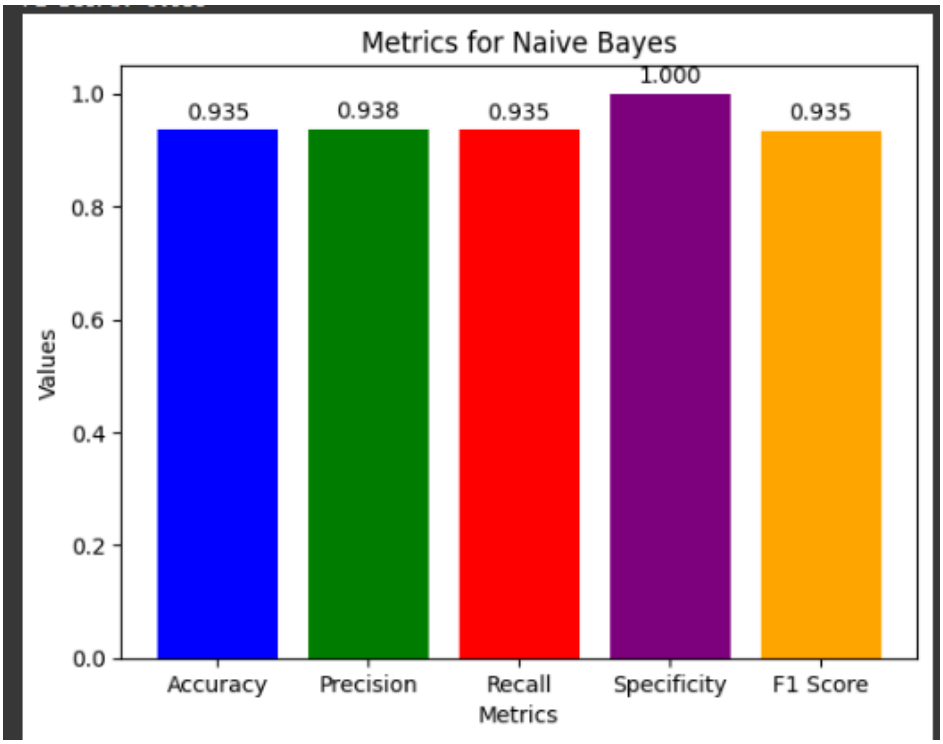


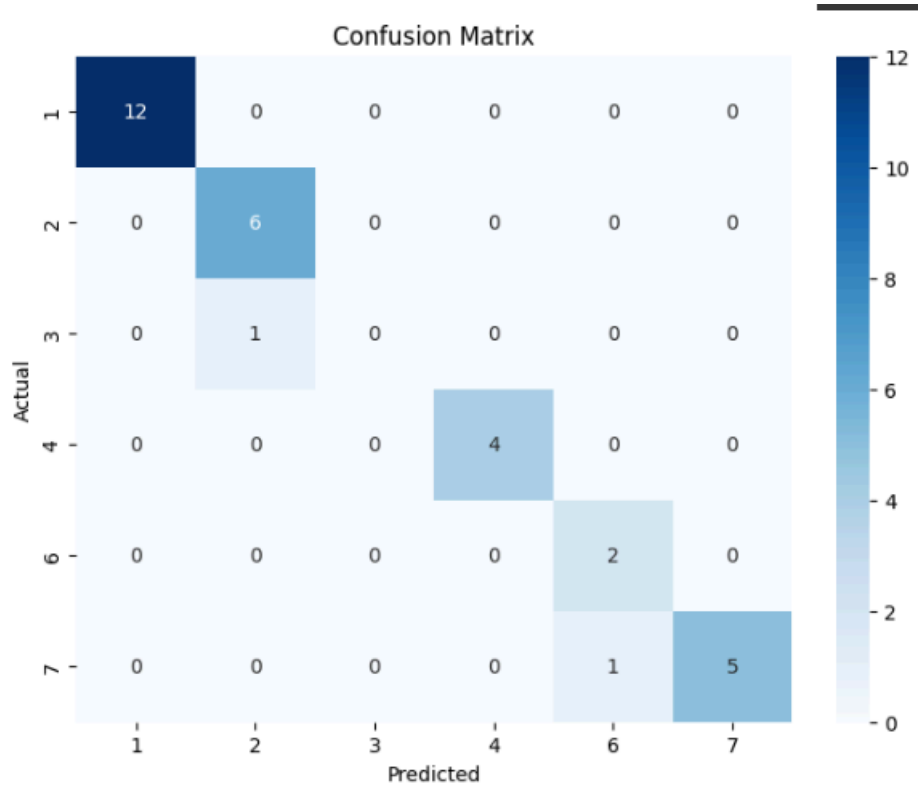

```
-------- Support Vector Machine Evaluation --------

Accuracy: 0.839
Precision: 0.873
Recall: 0.839
Specificity: 1.000
F1 Score: 0.817
```



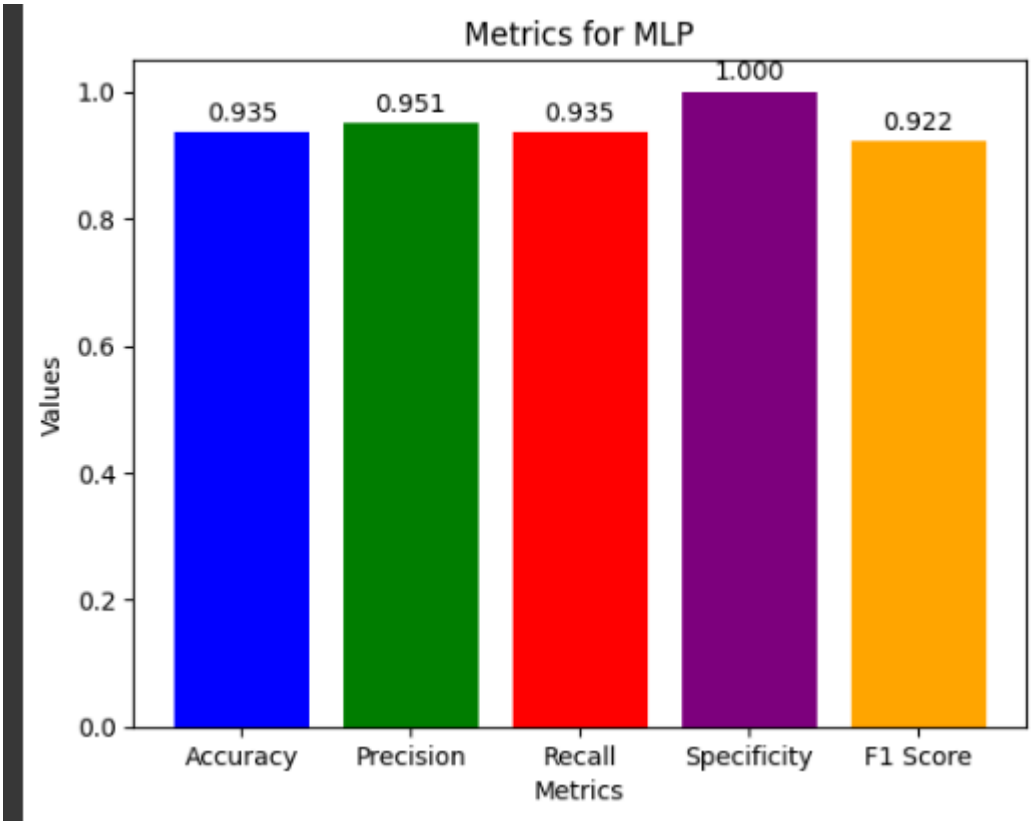



```
-------- Naive Bayes Evaluation --------

Accuracy: 0.935
Precision: 0.938
Recall: 0.935
Specificity: 1.000
F1 Score: 0.935
```






```
-------- MLP Evaluation --------

Accuracy: 0.935
Precision: 0.951
Recall: 0.935
Specificity: 1.000
F1 Score: 0.922
```



Conclusiones:

- Todos los modelos evaluados muestran un rendimiento decente en la clasificación de animales en el conjunto de datos "zoo".
- La regresión logística y el MLP (Multilayer Perceptron) parecen tener un rendimiento ligeramente superior en comparación con los otros modelos.
- La matriz de confusión y las métricas de evaluación proporcionan información detallada sobre el rendimiento de cada modelo, lo que permite comparar su efectividad en la clasificación de animales en diferentes categorías.