UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERIAS

Materia: Seminario de Solución de Problemas de Inteligencia Artificial II

Sección: D05

Profesor: Diego Campos Peña

Proyecto Final

Daniel Sánchez Zepeda 220286881

ANTACEDENTES

En el campo del aprendizaje automático, los clasificadores son herramientas esenciales para la asignación de categorías a distintas instancias de datos. Estos clasificadores varían ampliamente en sus enfoques y capacidades, lo que los hace adecuados para diferentes tipos de problemas y conjuntos de datos. Entre los clasificadores más comúnmente utilizados se encuentran la Regresión Logística, K-Vecinos Cercanos (K-Nearest Neighbors), Máquinas de Vectores de Soporte (Support Vector Machines) y Naive Bayes. Cada uno de estos métodos presenta ventajas y desventajas particulares que deben ser consideradas al seleccionar el modelo adecuado para una tarea específica.

La Regresión Logística es un modelo estadístico utilizado para predecir la probabilidad de una variable dependiente binaria. Es conocido por su simplicidad e interpretabilidad, aunque puede no ser adecuado para datos con relaciones no lineales. El K-Vecinos Cercanos (KNN) es un algoritmo de clasificación basado en la similitud que clasifica una instancia de datos en función de los K vecinos más cercanos. Aunque es fácil de implementar, su eficiencia puede disminuir con grandes conjuntos de datos y su desempeño depende de la correcta elección del valor de K. Las Máquinas de Vectores de Soporte (SVM) son métodos de aprendizaje supervisado que pueden ser utilizados tanto para clasificación como para regresión. Son especialmente efectivos en problemas de clasificación con alta dimensionalidad, pero pueden ser complejos y consumir muchos recursos computacionales. Naive Bayes, por otro lado, es un clasificador probabilístico que asume la independencia entre las características, lo que rara vez es el caso en la práctica. Sin embargo, es rápido y eficiente, particularmente con grandes volúmenes de datos.

La evaluación de estos clasificadores requiere el uso de diferentes conjuntos de datos que presenten características únicas. En este contexto, se ha seleccionado el conjunto de datos Zoo, disponible en el repositorio de aprendizaje automático de la UCI. Este conjunto de datos incluye diversas características de animales y se utiliza para clasificarlos en diferentes categorías. La implementación de estos clasificadores en el conjunto de datos Zoo permitirá un análisis comparativo de su desempeño y la identificación del método más adecuado para este tipo de datos.

Introducción

El aprendizaje automático ha revolucionado la manera en que se abordan los problemas de clasificación en diversas áreas, desde la biomedicina hasta la ingeniería y las ciencias sociales. En esta actividad, se explorarán y evaluarán algunos de los clasificadores más utilizados en aprendizaje automático, incluyendo la Regresión Logística, K-Vecinos Cercanos (K-Nearest Neighbors), Máquinas de Vectores de Soporte (Support Vector Machines) y Naive Bayes. El objetivo principal es familiarizarse con estos métodos, comprender sus diferencias y analizar sus ventajas y desventajas. Para este propósito, se empleará el conjunto de datos Zoo, que ofrece una rica variedad de características de animales, permitiendo una evaluación exhaustiva de cada clasificador. Se implementará cada método de clasificación y una red neuronal, generando modelos de aprendizaje automático adaptados a las particularidades del conjunto de datos. Además, se desarrollarán archivos CSV en caso de ser necesario para facilitar el análisis. La evaluación del rendimiento de los clasificadores se realizará mediante una serie de métricas estándar en el campo del aprendizaje automático, incluyendo Accuracy, Precision, Sensitivity, Specificity y F1 Score. Estas métricas proporcionarán una visión integral del desempeño de cada clasificador, permitiendo una comparación detallada y objetiva.

El análisis de los resultados se documentará en un informe detallado en formato PDF, donde se compararán los métodos implementados y se presentarán las conclusiones sobre el clasificador más adecuado para el conjunto de datos Zoo. Esta actividad no solo busca identificar el mejor método de clasificación, sino también proporcionar una comprensión profunda de las técnicas de evaluación y las consideraciones prácticas al aplicar diferentes clasificadores a conjuntos de datos reales.

Objetivos

- Conocer otros clasificadores comúnmente usados en aprendizaje maquina
- Analizar diferentes datasets
- Conocer las diferencias entre los métodos de aprendizaje automático
- Identificar las ventajas y desventajas de cada método de clasificación
- Conocer e implementar las métricas para evaluar a los clasificadores

CODIGO IMPLEMENTADO EN PYTHON COMENTADO

```
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.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
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
from sklearn.preprocessing import StandardScaler

def load_and_preprocess_data(file_path):
    """
    Carga y preprocesa el conjunto de datos Zoo.

    Args:
        file_path (str): Ruta del archivo CSV del conjunto de datos Zoo.

    Returns:
        tuple: Conjuntos de entrenamiento y prueba para características (X) y etiquetas (y).
    """
    dataset = pd.read_csv(file_path)
    X = dataset.drop(['animal_name', 'type'], axis=1)
    y = dataset['type']

    # Normalizar las características
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # Dividir los datos en conjuntos de entrenamiento y prueba
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
    return X_train, X_test, y_train, y_test

def evaluate_model(model_name, model, X_test, y_test):
    """
    Evalúa el modelo y muestra las métricas de rendimiento.

    Args:
        model_name (str): Nombre del modelo.
        model (sklearn estimator): Modelo entrenado.
        X_test (array): Datos de prueba.
        y_test (array): Etiquetas de prueba.
    """
    y_pred = model.predict(X_test)

    # Calcular métricas
```

```
    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')

    # Calcular specificity
    conf_matrix = confusion_matrix(y_test, y_pred)
    tn = conf_matrix[0][0]  # Verdaderos negativos
    fp = conf_matrix[0][1]  # Falsos positivos
    specificity = tn / (tn + fp) if (tn + fp) != 0 else 0

    # Imprimir métricas
    print(f"\n-------- {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}\n")

    # Graficar las métricas
    metrics_names = ['Accuracy', 'Precision', 'Recall', 'Specificity', 'F1 Score']
    metrics_values = [accuracy, precision, recall, specificity, f1]
    plt.bar(metrics_names, metrics_values, color=['blue', 'green', 'red', 'purple', 'orange'])
    plt.title(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()

def logistic_regression(X_train, X_test, y_train, y_test):
    model = LogisticRegression(max_iter=10000)
    model.fit(X_train, y_train)
    evaluate_model('Logistic Regression', model, X_test, y_test)

def k_nearest_neighbors(X_train, X_test, y_train, y_test):
    model = KNeighborsClassifier(n_neighbors=5)
    model.fit(X_train, y_train)
    evaluate_model('K-Nearest Neighbors', model, X_test, y_test)

def support_vector_machine(X_train, X_test, y_train, y_test):
    model = SVC(C=1.0)
    model.fit(X_train, y_train)
    evaluate_model('Support Vector Machine', model, X_test, y_test)

def naive_bayes(X_train, X_test, y_train, y_test):
    model = GaussianNB()
    model.fit(X_train, y_train)
    evaluate_model('Naive Bayes', model, X_test, y_test)
```

```
def neural_network(X_train, X_test, y_train, y_test):
    model = MLPClassifier(hidden_layer_sizes=(100,), max_iter=1000)
    model.fit(X_train, y_train)
    evaluate_model('Neural Network', model, X_test, y_test)

# Ruta del archivo CSV del conjunto de datos Zoo
file_path = 'zoo.csv'

# Cargar y preprocesar los datos
X_train, X_test, y_train, y_test = load_and_preprocess_data(file_path)

# Evaluación de los clasificadores
logistic_regression(X_train, X_test, y_train, y_test)
k_nearest_neighbors(X_train, X_test, y_train, y_test)
support_vector_machine(X_train, X_test, y_train, y_test)
naive_bayes(X_train, X_test, y_train, y_test)
neural_network(X_train, X_test, y_train, y_test)
```

Funcionalidad del código:

Función load_and_preprocess_data(file_path)
Objetivo: Cargar y preprocesar el conjunto de datos Zoo.
Parámetros: file_path (str) - Ruta del archivo CSV del conjunto de datos Zoo.
Retorna: Conjuntos de entrenamiento y prueba para características (X_train, X_test) y etiquetas (y_train, y_test).

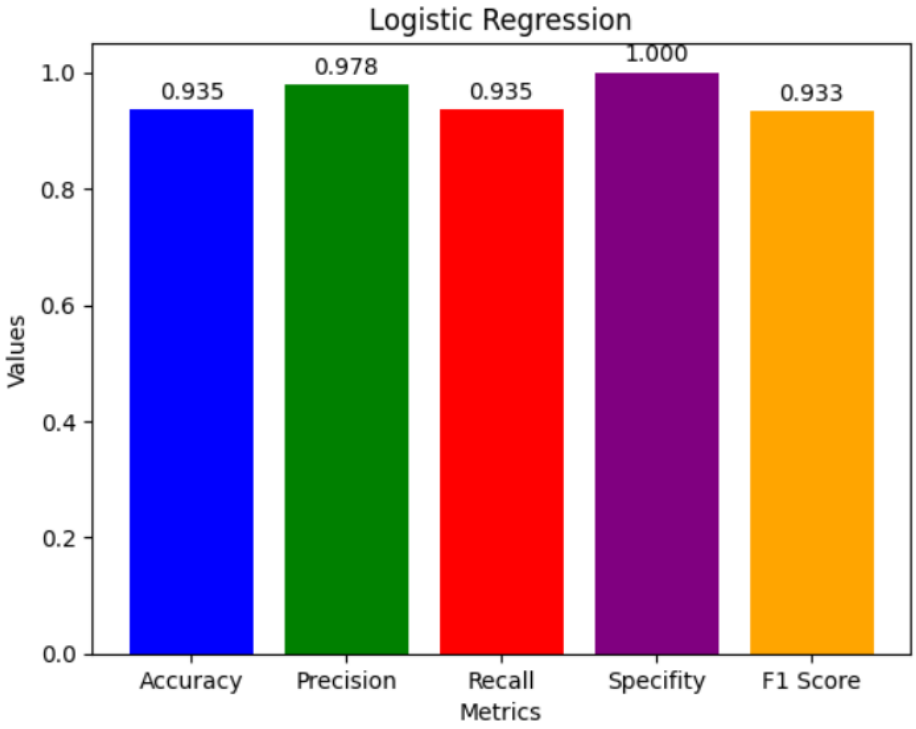
Función evaluate_model(model_name, model, X_test, y_test)
Objetivo: Evaluar el modelo y mostrar las métricas de rendimiento.
Parámetros:
model_name (str) - Nombre del modelo.
model (sklearn estimator) - Modelo entrenado.
X_test (array) - Datos de prueba.
y_test (array) - Etiquetas de prueba.

Funciones de Modelos (logistic_regression, k_nearest_neighbors, support_vector_machine, naive_bayes, neural_network)
Objetivo: Entrenar y evaluar cada modelo específico.
Parámetros:
X_train (array) - Datos de entrenamiento.
X_test (array) - Datos de prueba.
y_train (array) - Etiquetas de entrenamiento.
y_test (array) - Etiquetas de prueba.

Resultados

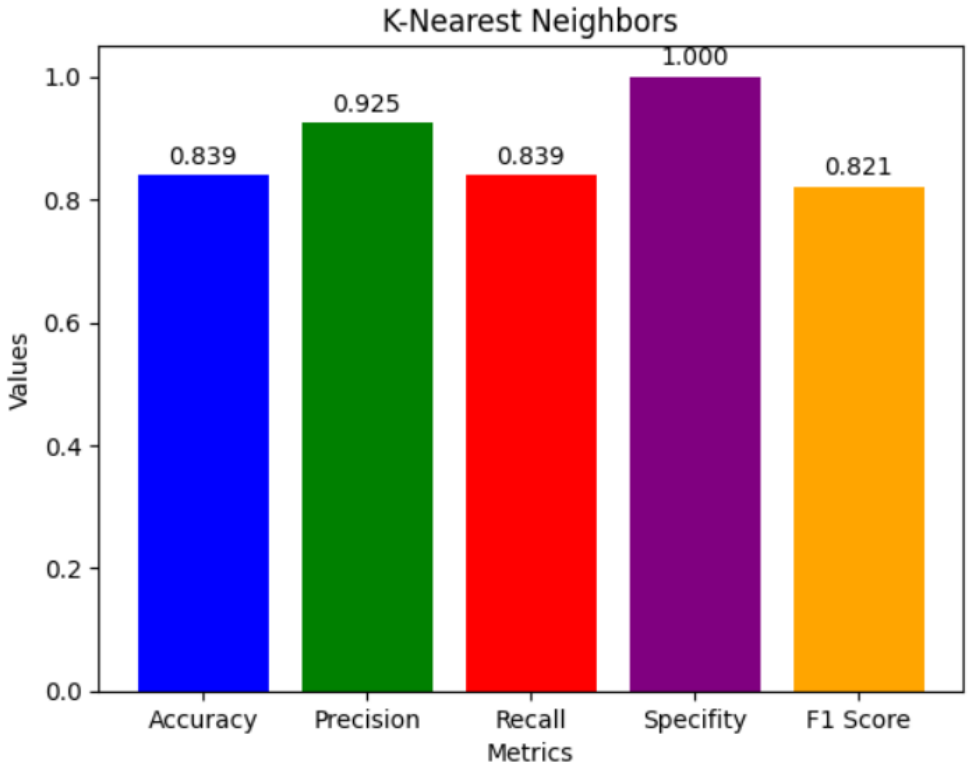
```
-------- Logistic Regression --------

Accuracy: 0.935
Precision: 0.978
Recall: 0.935
Specificity: 1.000
F1 Score: 0.933
```



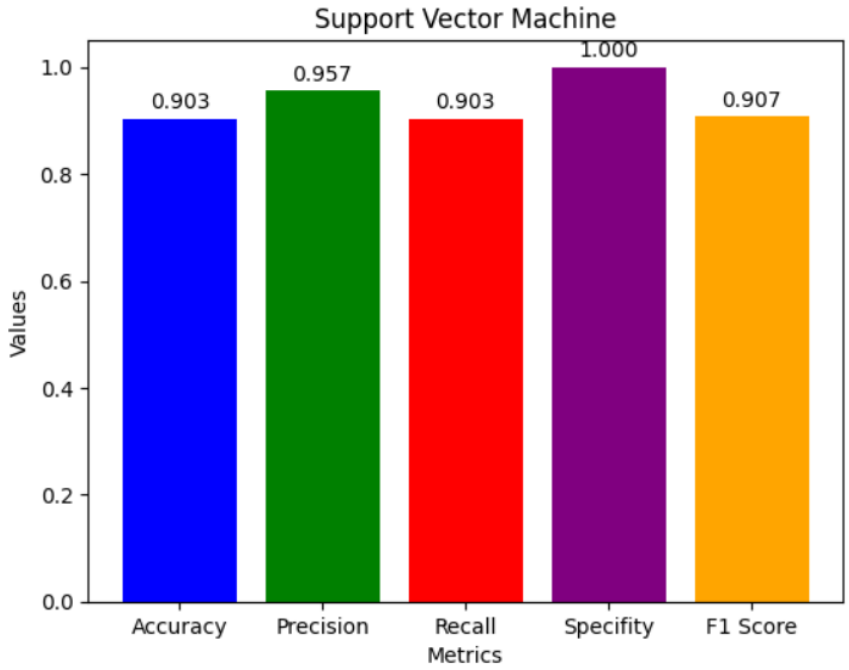
-------- K-Nearest Neighbors --------

Accuracy: 0.839
Precision: 0.925
Recall: 0.839
Specificity: 1.000
F1 Score: 0.821



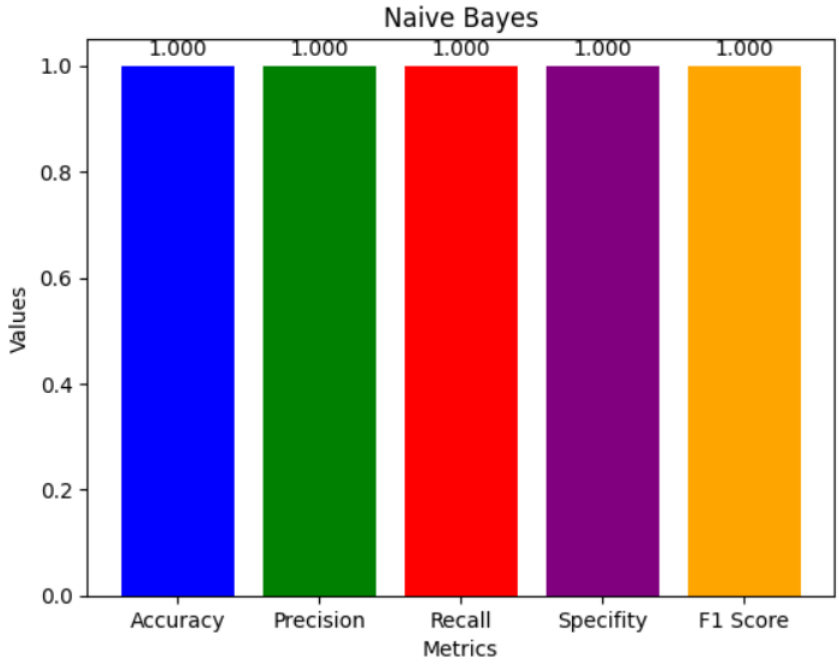
-------- Support Vector Machine --------

Accuracy: 0.903
Precision: 0.957
Recall: 0.903
Specificity: 1.000
F1 Score: 0.907



-------- Naive Bayes --------

Accuracy: 1.000
Precision: 1.000
Recall: 1.000
Specificity: 1.000
F1 Score: 1.000



MODELO	ACCURACY	PRECISION	RECALL	SPECIFITY	F1-SCORE
Regresión Logística	0.935	0.978	0.935	1.000	0.933
K-Vecinos Más Cercanos (KNN)	0.839	0.925	0.839	1.000	0.821
Máquinas de Vectores de Soporte (SVM)	0.903	0.957	0.903	1.000	0.907
Naive Bayes	1.000	1.000	1.000	1.000	1.000

CONCLUSION

En conclusión, este estudio ha subrayado la importancia de comprender las diferencias entre los métodos de clasificación y sus aplicaciones prácticas. La Regresión Logística, KNN, SVM, Naive Bayes y las redes neuronales son todas herramientas valiosas en el arsenal del aprendizaje automático, cada una adecuada para distintos escenarios y tipos de datos. La implementación y evaluación de estos clasificadores en el conjunto de datos Zoo ha proporcionado una comprensión profunda de sus ventajas y limitaciones, facilitando una toma de decisiones informada para futuras aplicaciones en diversos dominios. Este análisis comparativo no solo ha identificado el mejor método para el conjunto de datos Zoo, sino que también ha proporcionado una base sólida para la aplicación efectiva de técnicas de clasificación en problemas del mundo real.