INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
Reporte Práctica 5 “Evaluar un algoritmo de aprendizaje”
Integrantes: Cruz Barragan Ryan Nathanael
Rodríguez Vázquez Joshua Levi
Torres Abonce Luis Miguel
Correos: RNCB0963@gmail.com
joshualevirv@gmail.com
luiseishon9@gmail.com

I. Resumen

Evaluar un algoritmo de aprendizaje es crucial para determinar su efectividad, eficiencia y adecuación a un problema específico. Es por esto que esta práctica, la cual decidimos desarrollar en Python, muestra completamente el proceso de evaluación de los algoritmos de aprendizaje.

El objetivo principal de esta práctica es desarrollar un programa en python que utiliza un dataset, en este caso el dataset de iris, el cual contiene 150 registros de flores de iris, detallando la longitud y el ancho de los sépalos y pétalos de cada flor, junto con la clasificación de la especie, para poder entrenar y evaluar un algoritmo de aprendizaje.

El desarrollo de la práctica también incluye una interfaz gráfica mediante la cual se mostrarán las gráficas y también contiene un ejecutable en formato .exe para que el programa sea de fácil transporte y utilidad.

II. Palabras clave

- **MATLAB**: plataforma de programación y cálculo numérico utilizada para analizar datos, desarrollar algoritmos y realizar cálculos numéricos.
- **Algoritmo de aprendizaje:** es un método que permite a una máquina mejorar su rendimiento en tareas específicas mediante la experiencia y los datos.
- **Dataset:** Es un conjunto de datos organizados que se utiliza para análisis o entrenamiento de modelos.

III. Introducción

Evaluar un algoritmo de aprendizaje es un proceso esencial para determinar su eficacia, eficiencia y adecuación a un problema específico. Aprenderemos a realizar una evaluación exhaustiva de un algoritmo de aprendizaje, abordando desde la división del conjunto de datos hasta la comparación con un baseline sencillo.

Primero, dividiremos el conjunto de datos en tres partes: entrenamiento, validación y prueba, para entrenar el modelo, ajustar los hiperparámetros y evaluar el rendimiento final, respectivamente. Luego, utilizaremos diversas métricas de evaluación para clasificar y evaluar el modelo, como la exactitud, precisión, sensibilidad, F1 Score, matriz de confusión para clasificación, y MSE, RMSE, MAE, y R^2 para regresión.

A continuación, implementaremos la validación cruzada, utilizando K-Fold Cross-Validation y Leave-One-Out Cross-Validation (LOOCV) para asegurar la robustez del modelo. Analizaremos el sesgo y la varianza para comprender los errores del modelo y graficaremos curvas de aprendizaje para identificar problemas de sobreajuste o subajuste.

También mediremos el tiempo de entrenamiento y predicción, evaluaremos la interpretabilidad del modelo, y probaremos su robustez y capacidad de generalización en diferentes subconjuntos de datos. Aplicaremos técnicas de regularización, como L1 y L2 para regresión lineal y Dropout y Early Stopping para redes neuronales, para evitar el sobreajuste. Finalmente, compararemos el rendimiento del modelo con un baseline sencillo y evaluaremos el algoritmo en diferentes conjuntos de datos para verificar su capacidad de adaptación.

Para poner en práctica estos conceptos, trabajaremos con un ejemplo en python utilizando el conjunto de datos "iris". Cargaremos el conjunto de datos, extraemos características y etiquetas, y convertiremos las etiquetas categóricas a numéricas. Luego, dividiremos el conjunto de datos en entrenamiento y prueba, y entrenaremos un modelo de regresión logística para evaluar su rendimiento. Esta práctica nos permitirá aplicar y entender mejor las técnicas y conceptos fundamentales en la evaluación de algoritmos de aprendizaje en el campo de la inteligencia artificial.

IV. Desarrollo

La práctica fue desarrollada utilizando el lenguaje Python, y utilizando la librería tkinter para la realización de la interfaz gráfica, así como sklearn para el entrenamiento y evaluación.

Después cargamos los datos utilizando la librería scikitlearn y los preparamos dividiéndolos en dos subconjuntos: Datos de entrenamiento y datos de test. Y entrenamos el modelo utilizando un “RandomForestClassifier” y llamado la función “fit”

```
Pseudocódigo:
% Cargar y preparar los datos
iris = cargar_iris()
X = iris.data
y = iris.target
[X_entrenamiento, X_prueba, y_entrenamiento, y_prueba] =
dividir_datos_entrenamiento_prueba(X, y, tamaño_prueba=0.2, semilla=42)

% Crear el modelo
modelo = ClasificadorBosqueAleatorio(n_estimadores=100, semilla=42)
modelo.ajustar(X_entrenamiento, y_entrenamiento)
```

Después de esto, para mayor facilidad se decidió crear tres funciones, una para hacer predicciones, una para mostrar ayuda utilizando una interfaz gráfica y la última para mostrar la matriz de confusión, las cuales se mostraran a continuación:

```
Pseudocodigo:
% Crear la función para hacer una predicción
función hacer_predicción()
        muestra = [[5.1, 3.5, 1.4, 0.2]]  % Un ejemplo de entrada (puedes cambiar esto)
        predicción = modelo.predecir(muestra)
        messagebox.mostrar_información("Predicción", "La predicción del modelo es: " +
iris.nombres_objetivo[predicción][0])

% Crear la función de ayuda
función mostrar_ayuda()
        texto_ayuda = ("Esta aplicación utiliza un modelo de clasificación entrenado "
                "con el conjunto de datos de Iris. Al hacer clic en el botón 'EMPEZAR', "
                "se realiza una predicción usando una muestra de datos predefinida. "
                "También puedes generar una matriz de confusión y visualizar las curvas de
aprendizaje.")
        messagebox.mostrar_información("Ayuda", texto_ayuda)
```

```
% Crear la función para mostrar la matriz de confusión
función mostrar_matriz_confusión()
        predicciones_y = modelo.predecir(X_prueba)
        matriz_confusión = matriz_de_confusión(y_prueba, predicciones_y)
        visualización = MostrarMatrizDeConfusión(matriz_confusión=matriz_confusión,
etiquetas_visualización=iris.nombres_objetivo)
        visualización.plotear(cmap=plt.cm.Azules)
        plt.mostrar()
```

Por último, creamos la interfaz gráfica utilizando la librería tkinter, creamos todos los botones que queremos que se muestran en la interfaz, y llamamos la función “main loop” que corre siempre la interfaz hasta que el usuario decida salir de ella.

```
Pseudocódigo:
% Crear la ventana principal
raíz = tk.Tk()
raíz.título("Aplicación de ML con scikit-learn")

% Crear y agregar el botón "EMPEZAR"
botón_empezar = tk.Boton(raíz, texto="EMPEZAR", comando=hacer_predicción)
botón_empezar.pack(espacio_vertical=10)

% Crear y agregar el botón "Matriz de Confusión"
botón_matriz_confusión = tk.Boton(raíz, texto="Matriz de Confusión",
comando=mostrar_matriz_confusión)
botón_matriz_confusión.pack(espacio_vertical=10)

% Crear y agregar el botón "AYUDA"
botón_ayuda = tk.Boton(raíz, texto="AYUDA", comando=mostrar_ayuda)
botón_ayuda.pack(espacio_vertical=10)

% Iniciar el bucle principal de la interfaz gráfica
raíz.bucle_principal()
```

V. Resultados

La salida del sistema consiste en una pantalla principal la cual contiene tres botones, uno de los cuales permite empezar el entrenamiento, el otro mostrar la matriz de confusión y el tercero mostrar una ventana de ayuda.

Figura 4. Pantalla principal

Al presionar el botón de empezar, daremos inicio al entrenamiento ya mencionado anteriormente y para comprobar que este se entrenó, se mostrará una pantalla secundaria la cual muestra la predicción del modelo utilizando datos predeterminados que se pusieron en el código mostrado anteriormente.

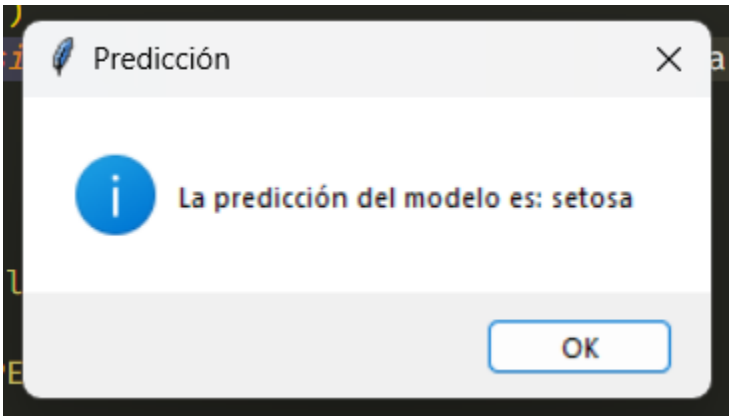

Figura 5. Predicción del modelo

Al presionar el botón llamado “Matriz de confusión” de la pantalla principal, se mostrará una pantalla secundaria que hace uso de la biblioteca matplotlib para poder visualizar dicha matriz de confusión.

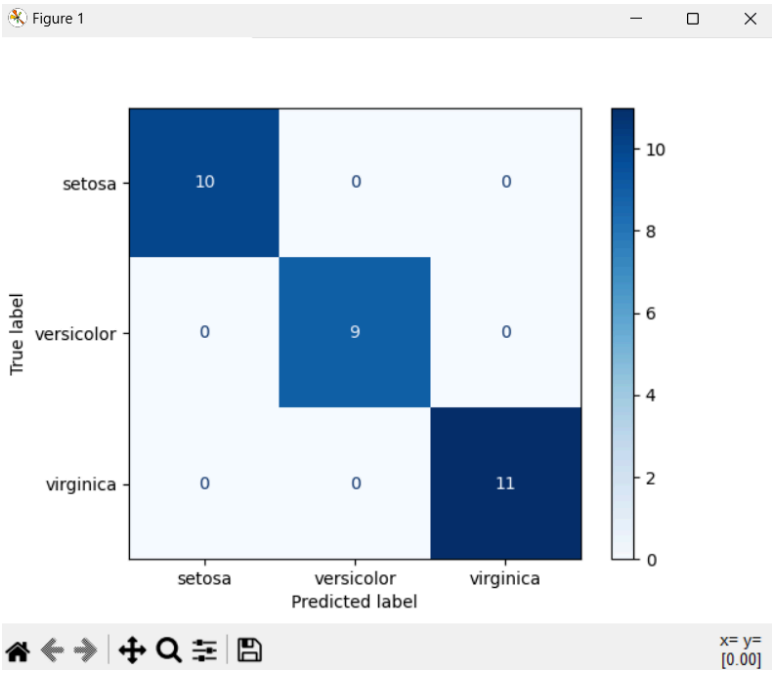

Figura 6. Matriz de confusión.

Por último al presionar el botón de “Ayuda”, este nos da una explicación de lo que hace el programa en una ventana nueva, como se muestra a continuación:

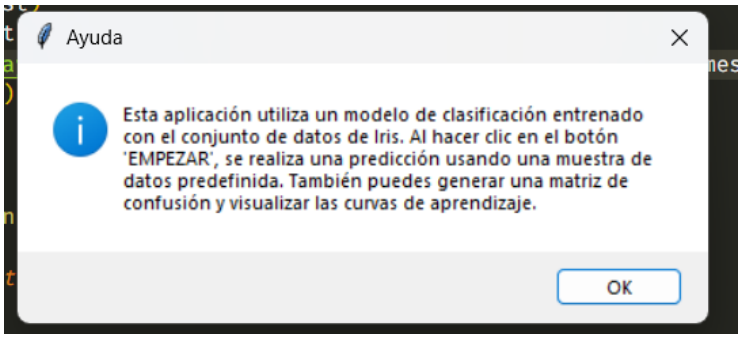

Figura 7. Botón de ayuda.

VI. Conclusión

Con el desarrollo de práctica, aprendimos a evaluar algoritmos de aprendizaje, un componente crucial para determinar su efectividad, eficiencia y adecuación a problemas específicos. Utilizando Python, implementamos y evaluamos un modelo de aprendizaje supervisado con el dataset de iris, compuesto por 150 registros de flores, detallando la longitud y el ancho de sépalos y pétalos, así como la clasificación de la

especie. A través de la división del conjunto de datos en entrenamiento, validación y prueba, logramos ajustar y evaluar el modelo de manera precisa.

Aplicamos diversas métricas de evaluación, como exactitud, precisión, sensibilidad y F1 Score para clasificación, y MSE, RMSE, MAE y R^2 para regresión. Utilizamos técnicas de validación cruzada como K-Fold y Leave-One-Out para asegurar la robustez del modelo y analizamos el sesgo y la varianza para comprender mejor los errores y optimizar el modelo. Las curvas de aprendizaje nos ayudaron a identificar problemas de sobreajuste y subajuste, mientras que la medición del tiempo de entrenamiento y predicción resultó esencial para aplicaciones en tiempo real.

VII. Bibliografía

[1] colaboradores de Wikipedia. (2024, 13 enero). *Conjuntos de datos de entrenamiento, validación y prueba*. Wikipedia, la Enciclopedia Libre. https://es.wikipedia.org/wiki/Conjuntos_de_datos_de_entrenamiento,_validaci%C3%B3n_y_prueba