INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Reporte Práctica 9 "Caja de herramientas de red neuronal"

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

La práctica fue desarrollada en Python, la cual implementa una aplicación de interfaz gráfica utilizando herramientas de redes neuronales (TensorFlow y Keras) para crear, entrenar y evaluar una red neuronal artificial (RNA) mediante un conjunto de datos. El uso de estas herramientas es cada vez más habitual en la actualidad en diferentes ámbitos industriales, como el control de procesos, de calidad y la predicción de fallos operacionales. La aplicación de estas herramientas dentro de la práctica fue de suma importancia para su correcto funcionamiento, teniendo una interacción con el usuario mediante una interfaz gráfica para ingresar el número de capas y el algoritmo de entrenamiento a utilizar, así como gráficas de salida para la visualización de la RNA, entre otras funcionalidades.

II. Palabras clave

Python, red neuronal artificial (RNA), TensorFlow, Keras, inteligencia artificial.

III. Introducción

La idea de que una máquina que piensa se remonta a los antiguos griegos. Sin embargo, no fue hasta 1943 que Warren S. McCulloch y Walter Pitts investigaron sobre los patrones complejos entre las células cerebrales y su relación con un umbral binario con la lógica booleana. A partir de ahí otros investigadores han indagado sobre la analogía que hay entre las neuronas y la inteligencia artificial (IA), de tal forma que en la actualidad la conversación con una máquina es real, teniendo varias aplicaciones útiles.

Una RNA es un programa, modelo o técnica de aprendizaje automático que está diseñado para tomar decisiones de manera similar al cerebro humano,

mediante el uso de procesos que imitan la forma en que las neuronas biológicas trabajan juntas para obtener conclusiones.

Cada RNA está conformada por capas de nodos (neuronas artificiales), divididas de la siguiente manera: capa de entrada, capa(s) oculta(s) y capa de salida. Cada nodo trabaja con una correspondiente ponderación y umbral conectado a otros nodos, en cuanto la salida de cualquier nodo sobrepase el umbral, se activa y envía datos a la siguiente capa.

Las RNA se basan en datos de entrenamiento para aprender y mejorar su precisión con el paso del tiempo. En cuanto se ajustan a la precisión requerida se convierten en herramientas poderosas en informática e IA, permitiendo clasificar y agrupar datos a alta velocidad.

Su funcionamiento básico consiste en entrenarla para que aprenda a asociar características de entrada con las de salida. Además, presentan la capacidad de generalización que les permite pronosticar comportamientos con base a situaciones que no han visto antes.

Para poder crear, entrenar, visualizar y simular RNA se utilizan algoritmos proporcionados por la caja de herramientas de red neuronal (Neural Network Toolbox), esto permite realizar clasificación, regresión, agrupamiento, reducción de dimensionalidad, pronóstico de series de tiempo, modelado y control de sistemas.

Un análisis adecuado con el uso de RNA debe incluir los siguientes pasos:

1. **Recopilar datos:** la calidad de los datos recopilados permitirá extraer el conocimiento adecuadamente.
2. **Crear la red:** se usan una serie de funciones disponibles en la caja de herramientas que permiten crear una RNA a través de los algoritmos esperados.
3. **Configurar la red:** consiste en examinar los datos de entrada y salida, establecer las dimensiones de entrada y salida para que se ajusten a los datos.
4. **Inicializar los pesos y sesgos:** consiste en establecer los valores iniciales desde los cuales comenzar y luego entrenar la red.
5. **Capacitar a la red:** es la fase más importante de todo el proceso, donde los pesos y sesgos deben ajustarse para optimizar el rendimiento de la red.
6. **Validar la red:** implica la validación de la red, en la que una fracción de los datos recopilados aleatoriamente se pasa a la red para estimar qué tan bien se ha entrado el modelo.

7. **Probar la red:** se ocupa para estimar la tasa de error después de haber elegido el modelo final.

En el presente trabajo, se pretende abordar estos 7 pasos para llevar a cabo la creación de una RNA de manera eficaz. Contemplando los datos de entrada, las capas involucradas, así como los algoritmos para entrenarla.

IV. Desarrollo

A grandes rasgos, el funcionamiento del programa es construir una red neuronal artificial a partir de datos cargados y estandarizados, para posteriormente ser entrenada con un tamaño especifico de capas ocultas y un optimizador seleccionado. El proceso de entrenamiento muestra el progreso y los resultados finales (rendimiento y errores) mediante gráficas y un histograma. El programa es interactivo con el usuario, ya que involucra botones para cargar los datos, entrenar la red, probarla, ver el diagrama y mostrara las gráficas del rendimiento.

Su implementación se baso en los pasos de la caja de herramientas, así como funciones para mostrar las salidas y el uso de bibliotecas específicas para su correcto funcionamiento, mostradas a continuación:

A) Importación de bibliotecas

Son módulos de Python que se utilizan para definir, compilar y entrenar la red neuronal:

- **Tkinter:** se usa para crear la interfaz gráfica de usuario.
- **Numpy:** biblioteca para el manejo de matrices y operaciones numéricas.
- **Slearn:** se utiliza para cargar el conjunto de datos, dividirlos en conjuntos de entrenamiento y prueba, y estandarizarlos.
- **Tensorflow.keras: s**e utiliza para crear y entrenar la red neuronal.
- **Matplotlib:** se usa para genera gráficas que visualizan el rednimiento de la red.

B) Carga y preparación de datos

La función "load_data" Crga el conjunto de datos (viviendas de California), estandariza los datos utilizando "StandardScale" y guarda los datos estandarizados en variables globales, habilitando el botón de entrenamiento.

C) Creación de la red neuronal

La función "create_network" define una red neuronal secuencial con capas densas para compilar el modelo con un optimizador específico y una función de error cuadrático medio.

D) Progreso del entrenamiento

La función "callback" imprime el progreso del entrenamiento al final de cada etapa.

E) Entrenamiento de la red neuronal

La función "train_network" obtiene los parámetros de la red desde la interfaz gráfica para crear y entrenar la red neuronal. Posteriormente, muestra el progreso del entrenamiento y habilita el botón para probar la red.

F) Funciones para graficar el rendimiento

Define varias funciones ("plot_perfomance", "plot_train_state", "plot_fit", "plot_regression" y "plot_err_hist") para graficar el rendimiento y estado del entrenamiento, así como el ajuste del modelo, regresión y errores.

G) Visualización del diagrama de la red

La función "view_network" muestra el resumen de la red neuronal entenada y habilita el botón para mostrar las gráficas.

H) Prueba de la red neuronal

La función "test_network" evalúa y prueba la red neuronal, mostrando el rendimiento de la red neuronal en el conjunto de datos de prueba.

I) Interfaz gráfica de usuario (GUI)

Define y organiza los elementos de la GUI para proporcionar campos de entrada y botones para interactuar con la red neuronal; cargar datos, entrenar, probar, ver el diagrama y mostrara gráficas.

Pseudocódigo

```
INICIAR

IMPORTAR librerías necesarias (tkinter, numpy, sklearn, tensorflow.keras, matplotlib)

DEFINIR función load_data():
```

```
    CARGAR el conjunto de datos de viviendas de California
    SEPARAR entradas y objetivos
    ESTANDARIZAR las entradas
    GUARDAR datos estandarizados en variables globales
    MOSTRAR mensaje de éxito
    HABILITAR el botón de entrenar
DEFINIR función create_network(hidden_layer_size, num_hidden_layers, optimizer):
    CREAR un modelo secuencial
    AGREGAR capa densa con tamaño hidden_layer_size y activación tanh
    REPETIR (num_hidden_layers - 1) veces:
        AGREGAR capa densa con tamaño hidden_layer_size y activación tanh
    AGREGAR capa de salida con una neurona y activación lineal
    COMPILAR el modelo con el optimizador y la función de pérdida mean_squared_error
    DEVOLVER el modelo
DEFINIR función on_epoch_end(epoch, logs):
    IMPRIMIR progreso del entrenamiento al final de cada época
DEFINIR función train_network():
    OBTENER parámetros de la interfaz gráfica
    CREAR optimizador basado en el nombre seleccionado
    CREAR la red neuronal con los parámetros obtenidos
    DIVIDIR los datos en conjuntos de entrenamiento, validación y prueba
    DEFINIR callback para mostrar el progreso del entrenamiento
    ENTRENAR la red neuronal
    GUARDAR el historial del entrenamiento
    PREDECIR salidas con los datos completos
    CALCULAR errores y rendimiento en el conjunto de prueba
    MOSTRAR mensaje de éxito
    HABILITAR el botón de probar
DEFINIR función plot_performance():
    GRAFICAR el rendimiento del entrenamiento (MSE de entrenamiento y validación)
DEFINIR función plot_train_state():
```

```
    GRAFICAR el estado del entrenamiento (loss de entrenamiento y validación)
DEFINIR función plot_fit():
    GRAFICAR ajuste del modelo (targets vs outputs)
DEFINIR función plot_regression():
    GRAFICAR regresión (valores reales vs predicciones)
DEFINIR función plot_err_hist():
    GRAFICAR histograma de errores
DEFINIR función view_network():
    SI la red está entrenada:
        MOSTRAR resumen del modelo
    SINO:
        MOSTRAR mensaje de error
    HABILITAR el botón de mostrar gráficas
DEFINIR función test_network():
    SI la red está entrenada:
        PREDECIR salidas con los datos completos
        CALCULAR errores
        GRAFICAR histograma de errores
        EVALUAR rendimiento en los datos completos
        MOSTRAR mensaje de rendimiento
    SINO:
        MOSTRAR mensaje de error
    HABILITAR el botón de diagrama de red
CREAR interfaz gráfica:
    DEFINIR ventana principal
    AGREGAR etiquetas, campos de entrada, combobox y botones a la ventana
    CONFIGURAR comandos para cada botón
    DESHABILITAR botones que no se pueden usar inicialmente
    INICIAR bucle principal de la ventana gráfica
EJECUTAR bucle principal de la ventana gráfica
```

V. Resultados

Figura 1. Interfaz de usuario.

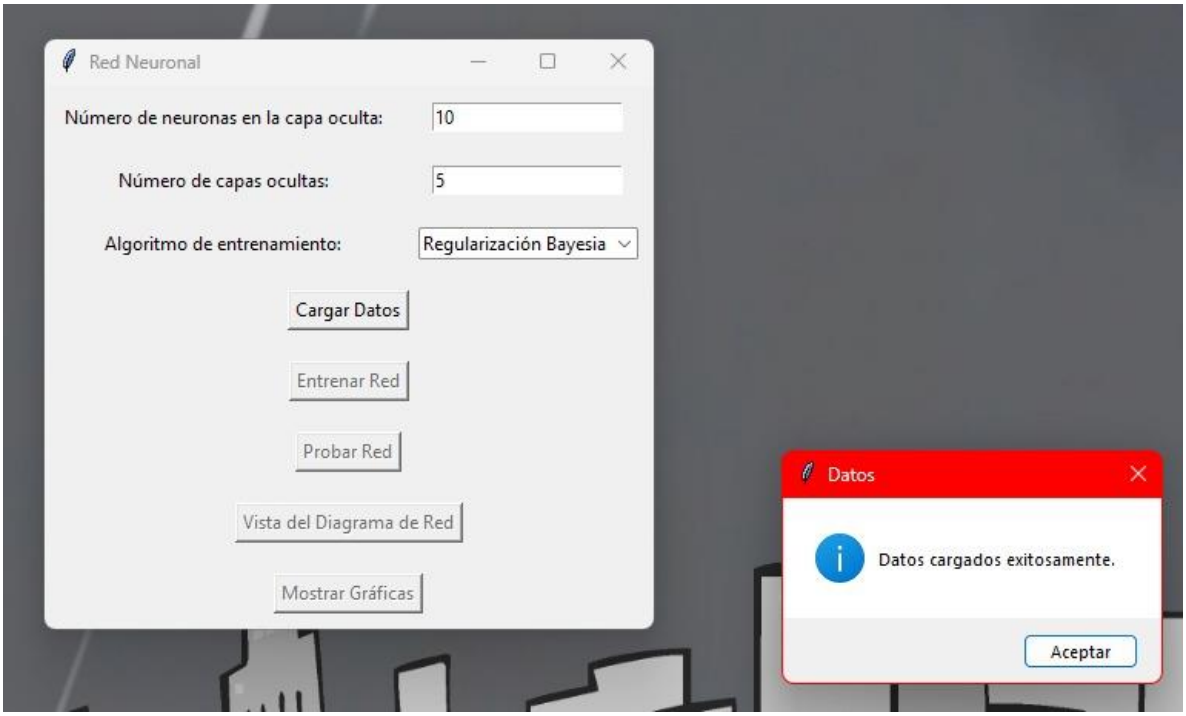

Figura 2. Etapa de ingreso de datos (número de neuronas, número de capas ocultas y algoritmo de entrenamiento).

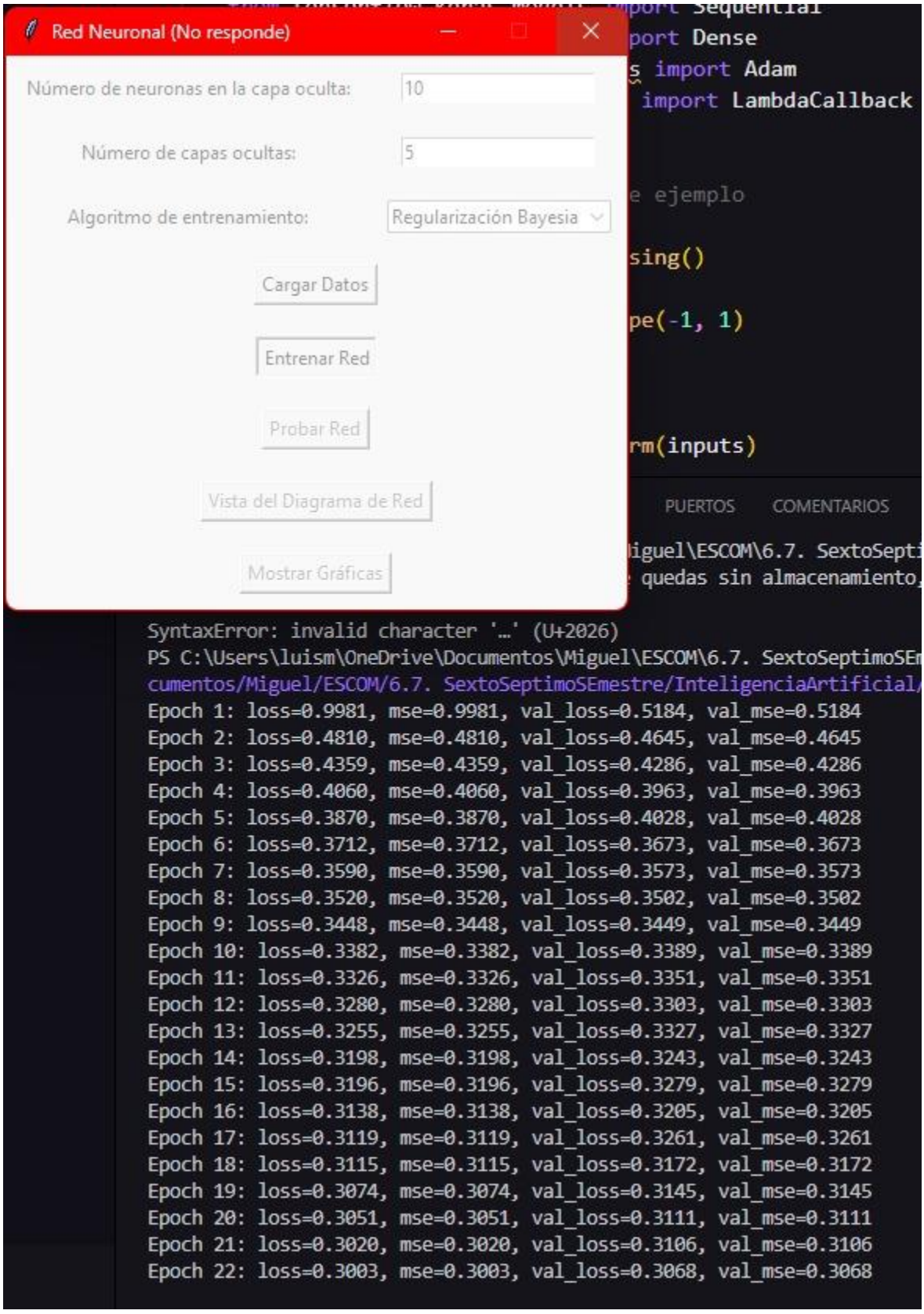

Figura 3. Etapa de entrenamiento de la red neuronal.

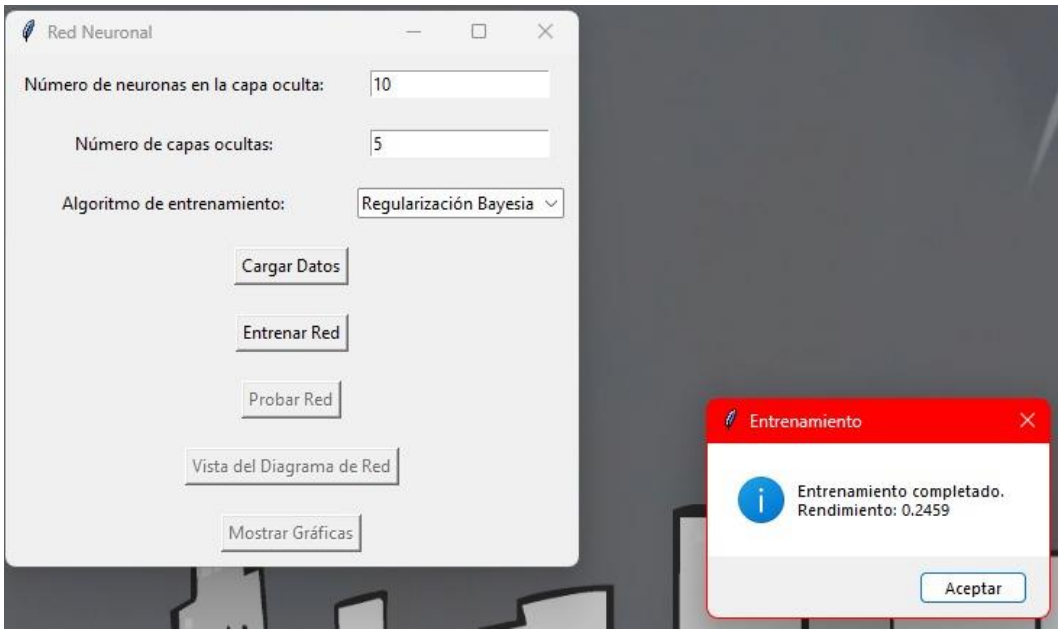

Figura 4. Visualización del estado del rendimiento (numérico) de la red neuronal.

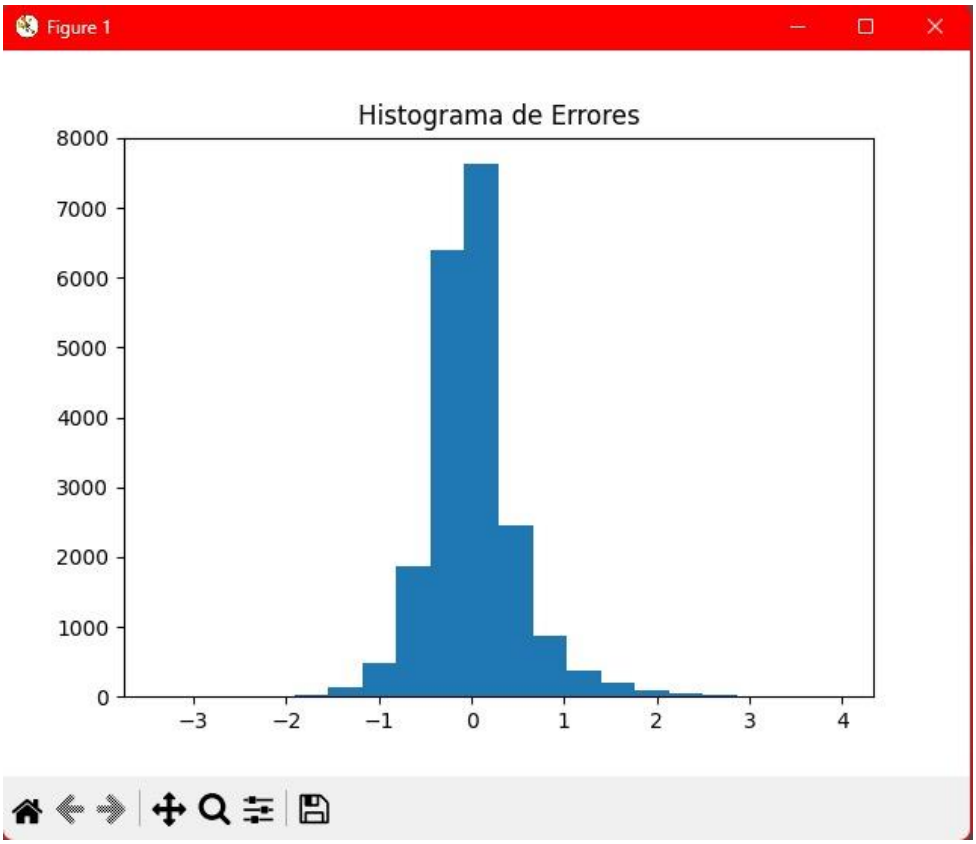

Figura 5. Histograma de errores.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 10)                90

 dense_1 (Dense)             (None, 10)                110

 dense_2 (Dense)             (None, 10)                110

 dense_3 (Dense)             (None, 10)                110

 dense_4 (Dense)             (None, 10)                110

 dense_5 (Dense)             (None, 1)                 11

=================================================================
Total params: 541
Trainable params: 541
Non-trainable params: 0
_________________________________________________________________
```

Figura 6. Parámetros de los datos.

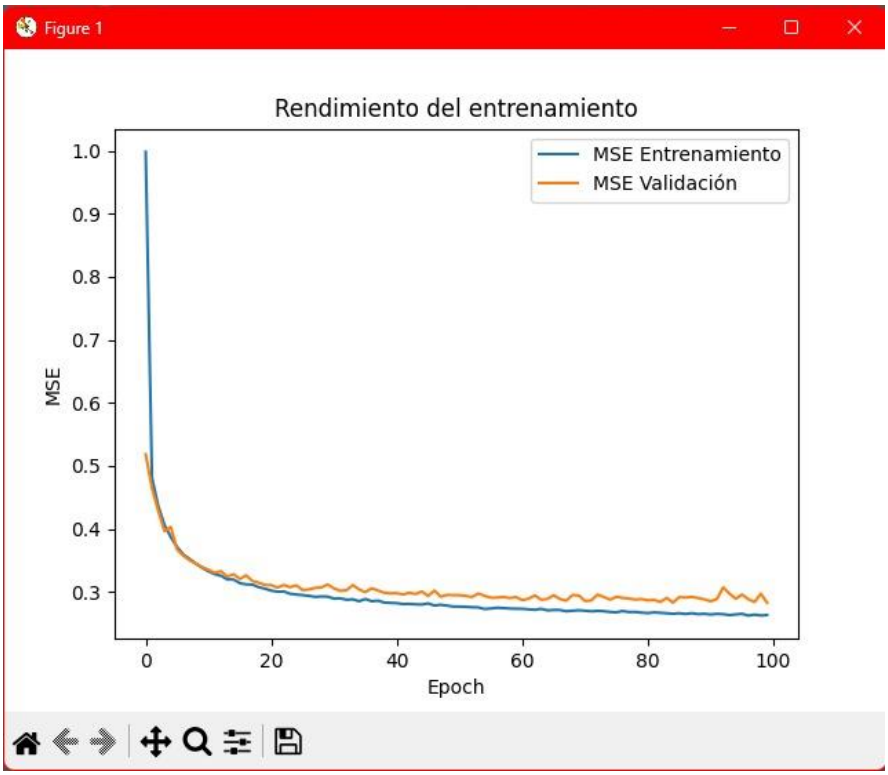

Figura 7. Rendimiento del entrenamiento.

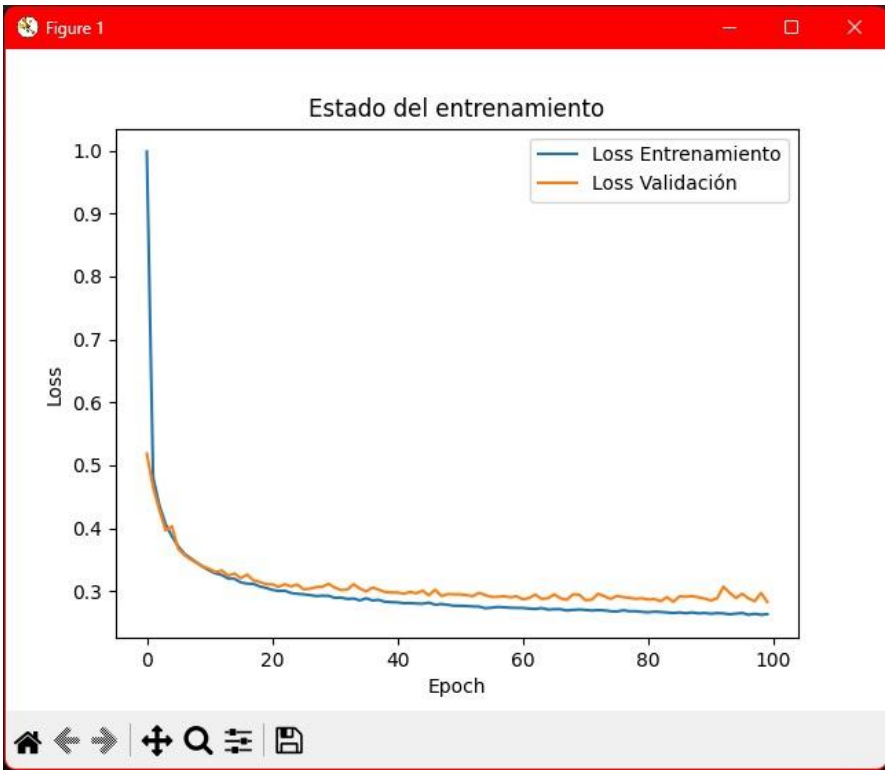

Figura 8. Estado del entrenamiento.

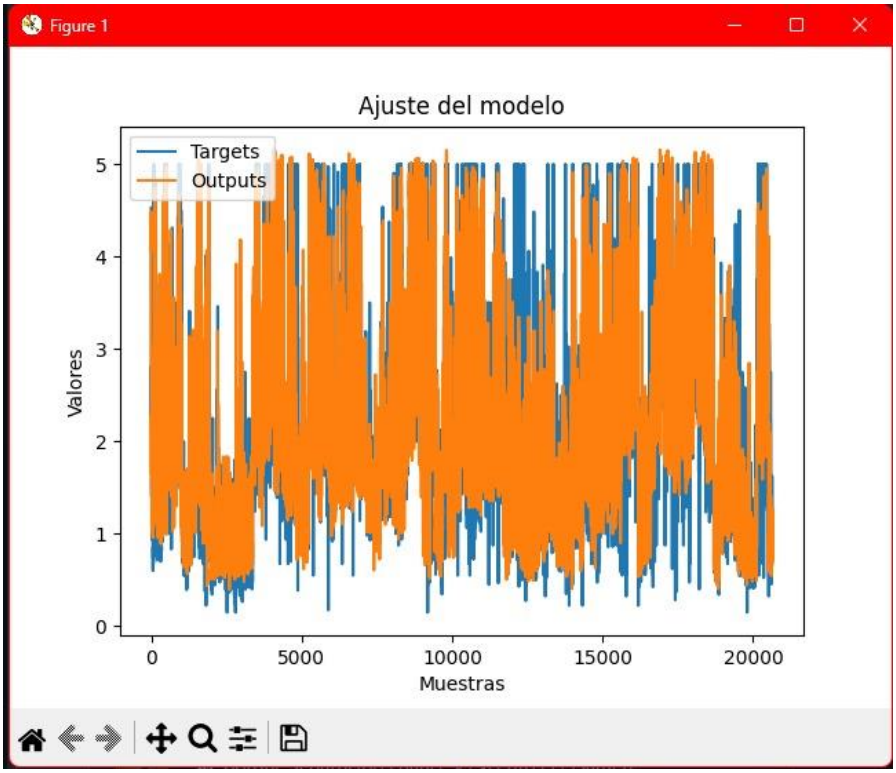

Figura 9. Ajuste del modelo.

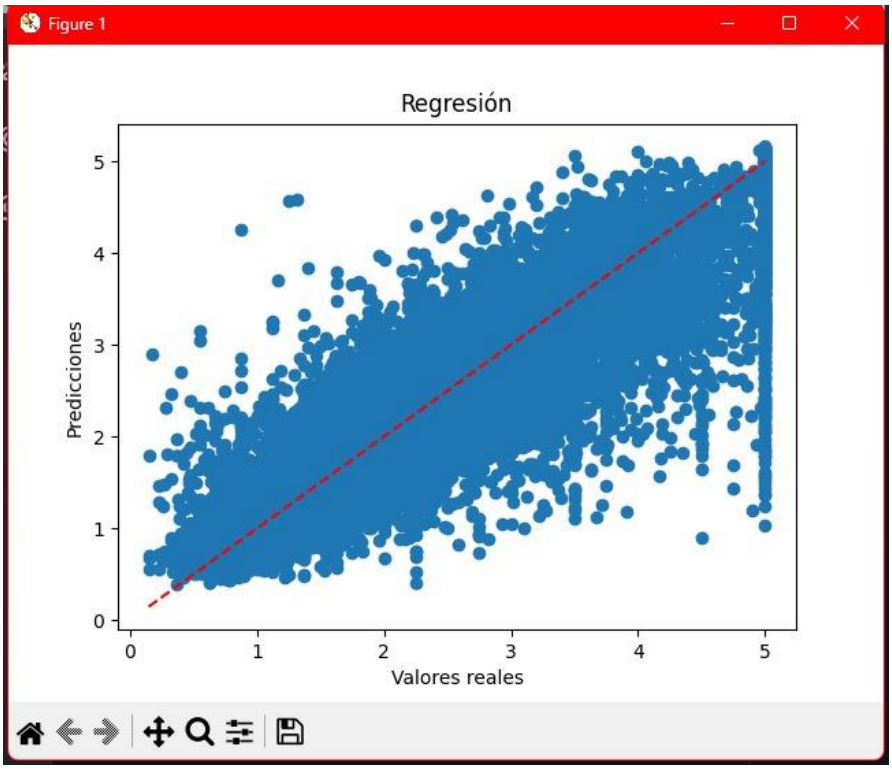

Figura 10. Regresión del modelo.

VI. Conclusión

El desarrollo de la práctica no solo demuestra cómo construir y entrenar una red neuronal artificial usando herramientas modernas, sino que también ilustra la importancia y versatilidad de las redes neuronales artificiales en la resolución de problemas complejos en múltiples dominios. La integración de una interfaz gráfica facilita la exploración interactiva, permitiendo a los usuarios experimentar con diferentes configuraciones y obtener una comprensión mas profunda del comportamiento y rendimiento de las redes neuronales.

La creación de estas redes neuronales es esencial en la actualidad en diversos campos: finanzas, entretenimiento, medicina y salud, automatización industrial, procesamiento de lenguaje natural (NLP), etc.

Por otro lado, cabe aclarar que el uso de la caja de herramientas para las redes neuronales es crucial en el desarrollo y experimentación con redes neuronales, proporcionando los siguientes factores: abstracción y facilidad de uso, optimización, flexibilidad y escalabilidad, callbacks y monitorización.

VII. Bibliografía

[1] *La base de las redes neuronales: desentrañamos el código*. (s. f.). ISO. https://www.iso.org/es/inteligencia-artificial/redes-neuronales

[2] Pascual, J. L. (2023, 5 mayo). Red neuronal artificial. Herramientas de inteligencia artificial. *Omnes*. https://omnesmag.com/recursos/reverendo-sos/herramientas-inteligencia-artificial/

[3] *¿Qué es una red neuronal? | IBM*. (s. f.). https://www.ibm.com/mx-es/topics/neural-networks

[4] *¿Qué es una red neuronal? - Explicación de las redes neuronales artificiales - AWS*. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/neural-network/