Instituto Tecnológico y de Estudios Superiores de Monterrey

Inteligencia Artificial Avanzada para la Ciencia de Datos I (Grupo 101)

Momento de Retroalimentación: Módulo 2 Uso de framework o biblioteca de aprendizaje máquina para la implementación de una solución. (Portafolio Implementación)

Daniel Kaled Bernal Ayala A01750047

10 de Septiembre del 2025

1. Introducción

En el presente documento se abordará la implementación de un árbol de decisiones. De igual manera, se realizará una comparativa entre una red neuronal utilizando el algoritmo de Back propagation. Además, se utiliza el dataset desarrollado para la prueba del anterior algoritmo. El árbol de decisión es un modelo de machine learning que toma decisiones en donde se evalúa cada una característica del dataset. Esto lo hace llegar a una hoja que corresponde a la clase de salida. Este modelo ayuda a explicar de mejor manera como se toman las decisiones y el por qué lo hace.

En este proyecto se emplea el framework **scikit-learn (sklearn)**. En particular, se hace uso de la clase **DecisionTreeClassifier** perteneciente al módulo **sklearn.tree**. Esta clase permite crear y entrenar un árbol de decisión para resolver problemas de clasificación supervisada, además, se utiliza la entropía para la división de los datos. De igual manera, se grafica el árbol de decisiones elaborado después del entrenamiento y testeo. Esta representación es de gran utilidad, ya que facilita la interpretación del modelo, mostrando los criterios de división en cada nodo y la profundidad alcanzada.

2. Base de Datos y División

La base de datos utilizada son números representados en una matriz 3x3 donde se expresan de la siguiente manera.

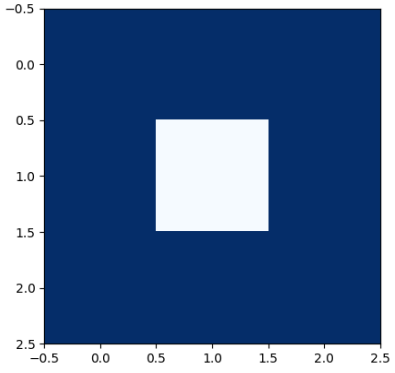

Figura 1. Matriz de 3x3 del número 0

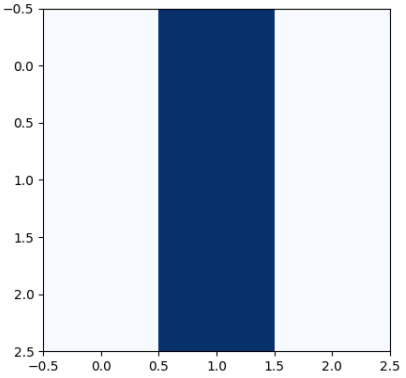

Figura 2. Matriz de 3x3 del número 1

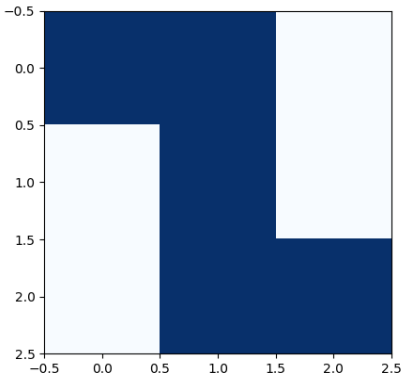

Figura 3. Matriz de 3x3 del número 2

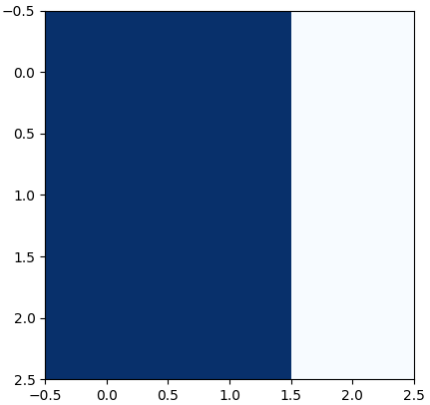

Figura 4. Matriz de 3x3 del número 3

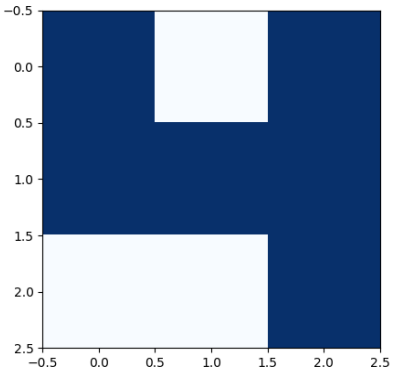

Figura 5. Matriz de 3x3 del número 4

La salida de las clases está dada en one-hot encoding donde (0,0,0,0,1) es la Figura 1, (0,0,0,1,0) para la Figura 2, (0,0,1,0,0) para la Figura 3, (0,1,0,0,0) para la Figura 4 y (1,0,0,0,0) para la Figura 5. Una vez establecido esto, se mantiene la expansión del dataset, en donde se agrega ruido Gaussiano para mantener una buena dispersión de datos. Así manteniendo la función **agregar_ruido().** Con esto hecho se realiza la distribución con el uso de la función **train_test_split()**, en donde se le asigna un 20% de los datos a testeo.

- **80% de los 105 datos = 84**
- **20% de los 105 datos = 21**

3. Implementación del modelo

Para la implementación del árbol de decisiones se utiliza la función **DecisionTreeClasifier**. Con ello se asigna el criterio de entropía para que los datos estén divididos de buena manera. Delimitamos la profundidad del árbol a 5 para evitar el overfitting, esto hace que generalice de buena forma para no aprenderse todo de memoria. Y por último, colocamos la semilla random a 42 para no perder el árbol hecho. Una vez elaborado esto, se llama a la función para realizar el entrenamiento y ajustar el árbol, creando las diferentes ramas con los atributos designados. Después, se evalúa el modelo con la función **clf.predict()** dando las variables **X del test.**

4. Evaluación y resultados

A continuación, se muestra el árbol de decisiones entrenado para poder analizar las decisiones tomadas.

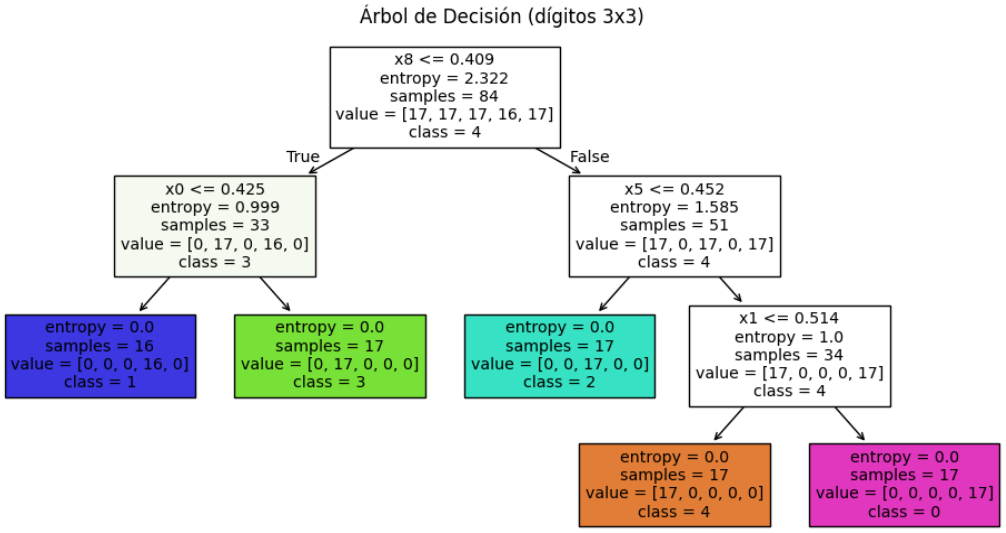

Figura 6. Árbol de decisión obtenido

Aquí podemos observar la manera en que hace las decisiones y se justifica que realiza para poder evaluar los resultados. Se menciona que si el valor de **x8** dentro de la lista dada de **X** es menor o igual a 0.409 este será clase 3. Esto se comprueba en el dataset original, ya que encuentra que tanto en la **Figura 2 y 4** que corresponde al número 1 y 3, en esa columna sus valores corresponden a 0. Haciendo la siguiente evaluación en donde el el x1 corresponde al primer valor del arreglo siendo el diferenciador entre si elegir el número 1 o 3. Lo mismo sucede con el lado derecho, donde en el primer nodo si no se cumple la decisión quiere decir que es mayor y se realiza la separación entre las 3 clases faltantes. Como lo es **x5**, que es el diferenciador de la clase 2 ya que tanto en la **Figura 5 y 1** tenemos los números 0 y 4 que tienen 1 en esa posición de **X**. Por último x1 es el diferenciador final entre decidir si es el número 0 o 4 respectivamente.

Una vez explicadas las decisiones que toma el árbol se tiene la matriz de confusión obtenida de los 21 datos evaluados.

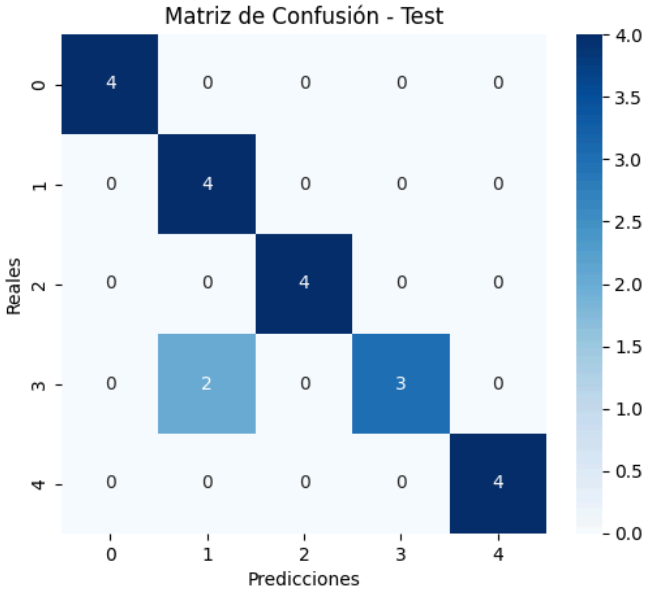

Figura 7. Matriz de confusión del test realizado

Con la matriz de confusión mostrada en la Figura 7, nos podemos dar cuenta que tuvo un accuracy del 90.5% lo que significa que el árbol generalizó bien los datos sin llegar al overfitting. La mayor confusión que tuvo fue con el número 3, ya que lo clasificaba como 1, a lo que se entiende, debido a su similitud entre ambos.

Tabla I. Métricas del modelo con el test realizado

	Precisión	Recall	F1-score	support
Cero	1.00	1.00	1.00	4
Uno	0.66	1.00	0.80	4
Dos	1.00	1.00	1.00	4
Tres	1.00	0.60	0.75	5
Cuatro	1.00	1.00	1.00	4

Con las métricas se observa esta confusión entre las Figuras anteriormente mencionadas. Sin embargo, el error no es mucho, lo que nos aseguraría un buen funcionamiento para la predicción de las clases derivadas de la base de datos.

5. Comparativa de resultados

Ahora bien, comparando los 2 modelos implementados como lo es el Backpropagation y el Decision Tree nos podemos dar cuenta que nos está tan alejado uno del otro. Para el Backpropagation se tienen los siguientes resultados con la matriz de confusión.

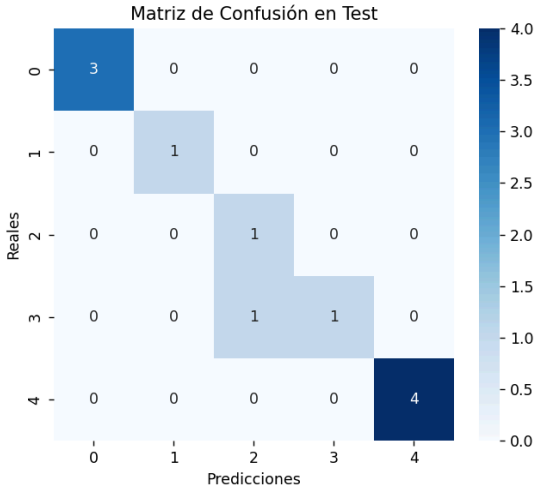

Figura 8. Matriz de confusión para el modelo Backpropagation.

El modelo mostró un accuracy del 90.91% al momento de procesar los datos. Cabe mencionar, que es una estructura más amplia en donde se abarca una red neuronal. Esta red neuronal tiene una capa oculta de 10 neuronas lo que permite generar las salidas correspondientes.

Tabla I. Métricas del modelo con el test realizado para Backpropagation

	Precisión	Recall	F1-score	support
Cero	1.00	1.00	1.00	3
Uno	1.00	1.00	1.00	1
Dos	0.50	1.00	0.67	1
Tres	1.00	0.50	0.67	2
Cuatro	1.00	1.00	1.00	4

En este caso, la confusión sucede en las Figuras 3 y 4 respectivamente. En donde, nos arrojaba resultados coincidentes a lo que se esperaba con el error. De igual forma se trabajó con el mismo modelo para evitar cambios en los resultados.

Ahora bien, en conclusiones con los dos modelos podemos observar gran diferencia en el comportamiento de cada uno y en los recursos que consumen independientemente. El primer detalle a destacar es que fue más fácil entender y justificar el modelo del árbol de decisión, debido a que se puede observar el flujo del mismo y ver hacia donde deriva el resultado de la clase que predice. Además, el procesamiento es menor al implementar una red neuronal, debido a los recursos que consume. Para aplicaciones sencillas como lo es el caso de este dataset, la implementación del Árbol es la mejor decisión debido a su fácil justificación, mejor procesamiento y ofreció un accuracy similar al compararlo con la red neuronal.

Las desventajas de implementarlo es que se puede sobreajustar rápido llegando al overfitting. También, los árboles muy profundos son difíciles de visualizar e interpretar. Además, no sería bueno si se busca una relación más compleja en los datos, lo que causaría que el modelo fuera muy sencillo para la complejidad que se busca. Sin embargo, para este dataset diseñado, es mejor implementación el árbol de decisión.

Referencias

Uresti, J. (2025). Presentación Tema 7 - Aprendizaje Supervisado. Recuperado del material visto en clase.