Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Estado de México

Implementación de una técnica de aprendizaje máquina sin el uso de un framework.

TC3006C. Inteligencia artificial avanzada para la ciencia de datos I

Grupo: 101

A01749667. Alan Contreras Prieto

Prof. Jorge Adolfo Ramírez Uresti

Fecha de entrega: 5 de septiembre de 2025

Semestre agosto – diciembre 2025

Modelo:

El modelo que hice fue una red neuronal

El modelo implementado consiste en una red neuronal artificial programada desde cero en Python sin hacer uso de frameworks especializados en machine learning, utilizando únicamente NumPy y Pandas. Para ello se definió una clase llamada RedNeuronal, en la cual se incluyen los siguientes elementos:

- Inicialización de pesos y sesgos aleatorios para cada capa.
- Función de activación sigmoide, que permite mapear las salidas a un rango entre 0 y 1, adecuado para problemas de clasificación binaria.
- Un método de propagación hacia adelante (feedforward) que calcula la salida de la red.
- Un método de backpropagation con descenso de gradiente, encargado de actualizar los pesos minimizando el error.
- Una función fit() que permite entrenar la red durante un número determinado de épocas y con una tasa de aprendizaje definida.

Dataset:

Para entrenar al modelo se utilizó un dataset sintético generado con la función make_classification() de sklearn. Los parámetros seleccionados fueron:

- n_samples = 10000 para contar con un conjunto de datos amplio.
- n_features = 5 de manera que la red tuviera múltiples entradas y se pudieran experimentar diferentes arquitecturas.
- n_classes = 2 para realizar una clasificación binaria.

La partición de datos se realizó con la función train_test_split() de sklearn, dividiendo el conjunto en 80% para entrenamiento y 20% para prueba.

Evaluación con matriz de confusión y métricas:

Para evaluar el desempeño de cada arquitectura se consideraron tanto la matriz de confusión como las principales métricas de clasificación:

1. Accuracy (exactitud): proporción de aciertos sobre el total de predicciones.
2. Precision (precisión): proporción de verdaderos positivos sobre las predicciones positivas.
3. Recall (sensibilidad): proporción de verdaderos positivos sobre los positivos reales.
4. F1-score: promedio armónico entre precision y recall.

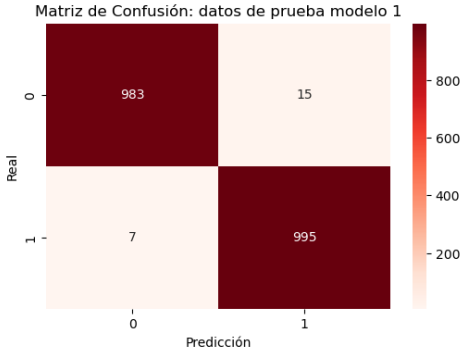
Se compararon varias arquitecturas, enfocándose en el desempeño final sobre el conjunto de prueba. Cabe resaltar que debido a la inicialización aleatoria de los pesos, los resultados pueden variar ligeramente en cada ejecución, pero la tendencia general se mantiene.

1. Capas: [3,5,3]

Este primer modelo dio resultados muy positivos, alcanzando un accuracy de 0.99 y valores altos en precision, recall y f1-score.

Reporte de Clasificación:

	precision	recall	f1-score	support
Clase 0	0.99	0.98	0.99	998
Clase 1	0.99	0.99	0.99	1002
accuracy			0.99	2000
macro avg	0.99	0.99	0.99	2000
weighted avg	0.99	0.99	0.99	2000

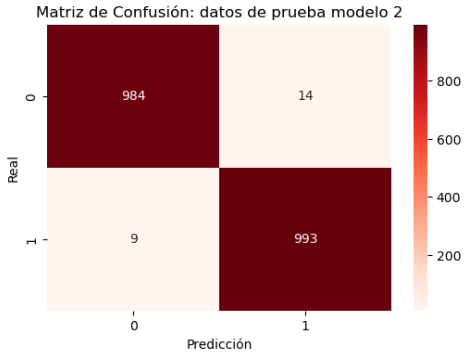


2. Capas: [8,10,5,3]

Una arquitectura más compleja que obtuvo un desempeño similar al modelo anterior, también con un accuracy de 0.99 y métricas consistentes.

Reporte de Clasificación:

	precision	recall	f1-score	support
Clase 0	0.99	0.99	0.99	998
Clase 1	0.99	0.99	0.99	1002
accuracy			0.99	2000
macro avg	0.99	0.99	0.99	2000
weighted avg	0.99	0.99	0.99	2000

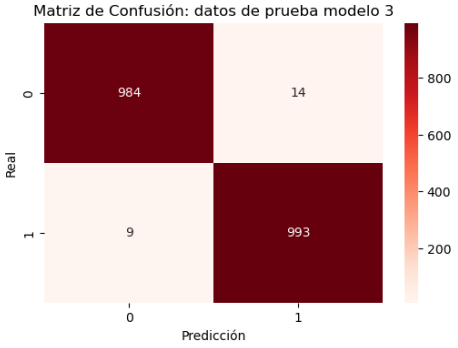


3. Capas: [30]

Se probó una red con una sola capa oculta pero con muchas neuronas. El desempeño se mantuvo en el mismo nivel (accuracy de 0.99), mostrando que incluso arquitecturas más simples pueden ser efectivas para este dataset.

Reporte de Clasificación:

	precision	recall	f1-score	support
Clase 0	0.99	0.99	0.99	998
Clase 1	0.99	0.99	0.99	1002
accuracy			0.99	2000
macro avg	0.99	0.99	0.99	2000
weighted avg	0.99	0.99	0.99	2000

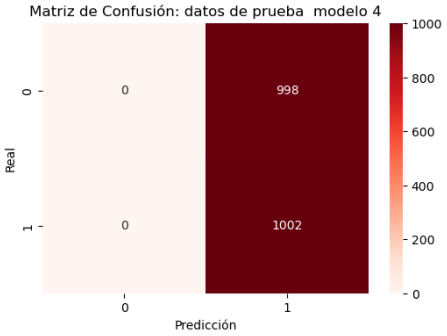


4. Capas: [3,3,3,3,3,3,3,3,3,3]

Por otro lado, ¿qué pasaría si se hace lo contrario? ¿Qué pasa cuando se usan muchísimas capas con pocas neuronas? Los resultados fueron simplemente malos, accuracy de 0.51 y eso solo porque las predicciones arrojaron todos los valores como cero y las clases están balanceadas.

Reporte de Clasificación:

	precision	recall	f1-score	support
Clase 0	0.00	0.00	0.00	998
Clase 1	0.50	1.00	0.67	1002
accuracy			0.50	2000
macro avg	0.25	0.50	0.33	2000
weighted avg	0.25	0.50	0.33	2000

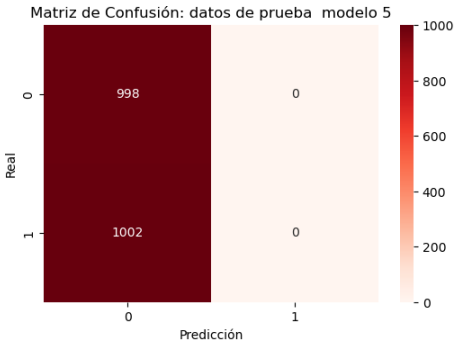


5. Capas: [10,10,10,10,10]

Finalmente ¿Qué pasaría si se busca un balance entre ambos? Al intentar combinar el estilo de las 2 redes anteriores (varias capas con varias neuronas) se llega a un resultado malo, al igual que el modelo 4 todas las predicciones apuntan solamente a una clase.

Reporte de Clasificación:

	precision	recall	f1-score	support
Clase 0	0.50	1.00	0.67	998
Clase 1	0.00	0.00	0.00	1002
accuracy			0.50	2000
macro avg	0.25	0.50	0.33	2000
weighted avg	0.25	0.50	0.33	2000



Adicionalmente se puede ver como durante el entrenamiento desde las primeras épocas se nota un estancamiento en la función de perdida, lo que se puede interpretar como que ya no está aprendiendo nada más.

Conclusiones:

Los resultados muestran que, para este problema en particular, modelos relativamente simples como [3, 5, 3] o incluso una sola capa con muchas neuronas ([30]) son suficientes para obtener un alto desempeño. En contraste, arquitecturas demasiado profundas o desbalanceadas tienden a sobreajustar o a no aprender adecuadamente, llevando a predicciones erróneas.