Inteligencia Artificial Avanzada para la Ciencia de Datos 1

Adrián Galván Díaz
Escuela de Ingeniería y Ciencias
Instituto Tecnológico y de Estudios Superiores de Monterrey
Querétaro, México
A01704076@tec.mx

***Abstract* — Este proyecto se enfoca en la implementación manual de un algoritmo de regresión logística utilizando gradiente descendente, sin la asistencia de marcos de trabajo o bibliotecas externas. El objetivo principal fue desarrollar un modelo capaz de predecir con precisión si un hongo es comestible o venenoso, basándose en un conjunto de datos categóricos que describe 23 especies de hongos de las familias Agaricus y Lepiota. A través de un riguroso Análisis Exploratorio de Datos (EDA), se preparó y transformó el set de datos, lo que permitió la creación y evaluación de los modelos de clasificación.**

Para evaluar la capacidad del modelo y comparar su rendimiento con soluciones basadas en frameworks, se implementó un modelo adicional utilizando Random Forest Tree con la librería scikit-learn. Esta solución alcanzó una precisión del 100% en el conjunto de prueba, lo que motivó un esfuerzo adicional por mejorar el rendimiento del modelo manual. Este enfoque permitió explorar el potencial de mejora en la precisión y la generalización del modelo sin frameworks.

Al final del reporte, también se decidió profundizar en el análisis del algoritmo de Random Forest Tree. Esto incluyó mover hiperparámetros y realizar los mismos análisis detallados que se aplicaron al modelo manual. Sin embargo, fue necesario limitar el modelo de Random Forest Tree para que este análisis fuera factible, debido a la complejidad inherente del algoritmo y su alto rendimiento desde el inicio.

Los resultados obtenidos demostraron un alto porcentaje de precisión, confirmando la eficacia de ambos algoritmos en la tarea de clasificación, subrayando la importancia de la validación continua y la optimización de hiperparámetros para garantizar un rendimiento óptimo. Este proyecto no solo ofrece una contribución significativa al entendimiento y aplicación de algoritmos de Machine Learning sin frameworks, sino que también establece una base sólida para la escalabilidad de técnicas de clasificación en conjuntos de datos categóricos complejos.

Keywords* — *Regresión Logística, Gradiente Descendente, Análisis Exploratorio de Datos (EDA), Machine Learning (ML), Inteligencia Artificial (IA), Split Test, Matriz de Confusión, Épocas/Epochs, Learning Rate, Parámetros, Hiperparámetros, Python, Instancias, Clase(s), Features, Entropía Cruzada, Random Forest Trees/Bosque de árboles aleatorio, Frameworks, Árboles de decisión, Varianza, Sesgo, Sigmoide, Clasificación Binaria, Overfitting/Underfitting, One Hot Encoding

I. Introducción

La inteligencia artificial (IA) ha evolucionado significativamente desde sus inicios a mediados del siglo XX [1]. Originalmente, la IA se centraba en la creación de sistemas que pudieran realizar tareas que normalmente requerirían inteligencia humana, como la resolución de problemas y el procesamiento del lenguaje natural. Con el tiempo, la IA se ha diversificado en varios subcampos, siendo el aprendizaje automático o machine learning (ML) uno de los más prominentes. ML se refiere a la capacidad de las máquinas para aprender de datos y mejorar su rendimiento sin ser programadas explícitamente para cada tarea específica [2].

El desarrollo del aprendizaje automático comenzó con el reconocimiento de patrones y se ha expandido para incluir una amplia gama de técnicas que permiten a los sistemas aprender y tomar decisiones basadas en datos. Hoy en día, ML se clasifica principalmente en tres categorías: aprendizaje supervisado, no supervisado y por refuerzo.

Aprendizaje supervisado: Involucra entrenar un modelo en un conjunto de datos etiquetado, donde la respuesta correcta es conocida. Es ideal para tareas de clasificación y regresión. [3]

Aprendizaje no supervisado: Se utiliza cuando los datos no están etiquetados y el objetivo es identificar estructuras ocultas o patrones en los datos, como en la agrupación o reducción de dimensionalidad. No necesita de intervención humana en el proceso de aprendizaje. [3]

Aprendizaje por refuerzo: Implica entrenar un modelo para tomar decisiones secuenciales a través de un sistema de recompensas y castigos, comúnmente utilizado en áreas como el control robótico y los videojuegos. [3]

Hoy en día, la IA y el ML tienen un impacto profundo en diversos sectores, desde la medicina hasta la industria, mejorando procesos, optimizando recursos y generando soluciones innovadoras.

A. Contexto general

En el contexto de este proyecto, los algoritmos de machine learning, en particular los de clasificación, son fundamentales para resolver problemas de decisión binaria. Este proyecto se enfoca en el uso de un algoritmo de regresión logística, una técnica de aprendizaje supervisado, para predecir si un hongo es comestible o venenoso basándose en sus características observables.

La regresión logística es uno de los métodos más comunes para problemas de clasificación binaria, mientras que la regresión lineal se utiliza generalmente para problemas de predicción continua. La capacidad de estos modelos para hacer predicciones precisas y basadas en datos los convierte en herramientas esenciales en la toma de decisiones en diversas áreas.

Aplicar machine learning al análisis de hongos, una rama de la biología, permite no solo avanzar en la identificación precisa de especies, sino también abrir la puerta a aplicaciones en otras áreas biológicas, como la clasificación de plantas, la detección de enfermedades en cultivos, y el análisis de patrones de comportamiento animal.

B. Base de Datos

El conjunto de datos utilizado en este proyecto proviene del UC Irvine Machine Learning Repository [4], un recurso ampliamente reconocido y de acceso público. Esta base de datos contiene descripciones de muestras hipotéticas de 23 especies de hongos pertenecientes a las familias Agaricus y Lepiota. En total, la base de datos cuenta con 8124 instancias y 22 atributos, todos nominalmente valorados. A continuación, se listan las variables:

Tabla 1. Tabla de Variables

Nombre de la Variable	**Descripción**	**Valores Posibles**
cap-shape	Forma del sombrero	bell (b), conical (c), convex (x), flat (f), knobbed (k), sunken (s)
cap-surface	Superficie del sombrero	fibrous (f), grooves (g), scaly (y), smooth (s)
cap-color	Color del sombrero	brown (n), buff (b), cinnamon (c), gray (g), green (r), pink (p), purple (u), red (e), white (w), yellow (y)
bruises	Presencia de moretones	bruises (t), no (f)
odor	Olor	almond (a), anise (l), creosote (c), fishy (y), foul (f), musty (m), none (n), pungent (p), spicy (s)
gill-attachment	Adherencia de las branquias	attached (a), descending (d), free (f), notched (n)
gill-spacing	Espaciado de las branquias	close (c), crowded (w), distant (d)
gill-size	Tamaño de las branquias	broad (b), narrow (n)
gill-color	Color de las branquias	black (k), brown (n), buff (b), chocolate (h), gray (g), green (r), orange (o), pink (p), purple (u), red (e), white (w), yellow (y)
stalk-shape	Forma del tallo	enlarging (e), tapering (t)
stalk-root	Raíz del tallo	bulbous (b), club (c), cup (u), equal (e), rhizomorphs (z), rooted (r), missing (?)
stalk-surface-above-ring	Superficie del tallo sobre el anillo	fibrous (f), scaly (y), silky (k), smooth (s)
stalk-surface-below-ring	Superficie del tallo bajo el anillo	fibrous (f), scaly (y), silky (k), smooth (s)
stalk-color-above-ring	Color del tallo sobre el anillo	brown (n), buff (b), cinnamon (c), gray (g), orange (o), pink (p), red (e), white (w), yellow (y)
stalk-color-below-ring	Color del tallo bajo el anillo	brown (n), buff (b), cinnamon (c), gray (g), orange (o), pink (p), red (e), white (w), yellow (y)
veil-type	Tipo de velo	partial (p), universal (u)
veil-color	Color del velo	brown (n), orange (o), white (w), yellow (y)
ring-number	Número de anillos	none (n), one (o), two (t)
ring-type	Tipo de anillo	cobwebby (c), evanescent (e), flaring (f), large (l), none (n), pendant (p), sheathing (s), zone (z)

spore-print-color	Color de la impresión de las esporas	black (k), brown (n), buff (b), chocolate (h), green (r), orange (o), purple (u), white (w), yellow (y)
population	Población	abundant (a), clustered (c), numerous (n), scattered (s), several (v), solitary (y)
habitat	Hábitat	grasses (g), leaves (l), meadows (m), paths (p), urban (u), waste (w), woods (d)

II. Análisis Exploratorio de Datos

El objetivo principal del Análisis Exploratorio de Datos (EDA) es conocer y comprender la estructura del conjunto de datos utilizado en este proyecto. Este proceso incluye la identificación de valores nulos, la visualización de la distribución de las variables mediante gráficas, y la utilización de técnicas estadísticas como una matriz de correlación para tomar decisiones informadas sobre las variables más relevantes para el modelo. La finalidad del EDA es preparar los datos de manera adecuada para la construcción y evaluación del modelo de Machine Learning.

A. Transformación de los Datos

Para preparar los datos para el modelo de Machine Learning, se realizó una transformación de las variables categóricas utilizando One Hot Encoding [5]. Dado que las variables en el dataset no tenían una composición vectorial, sino que eran meramente clasificatorias (colores, formas), el One Hot Encoding fue la técnica más adecuada para este caso. De esta forma se transformaron las variables donde cada una representa la presencia o ausencia de valor. Por ejemplo: bruises (1. bruises_t, 2. bruises_f) (Tabla 1 y Figura 2).

B. Revisión de los Datos

Durante la revisión de los datos, se identificó la presencia de valores nulos en la variable stalk_root, que representa la forma de la raíz del tallo. Esta variable contenía 2480 valores nulos, lo que representa una proporción significativa del total de 8124 instancias en el dataset. Los valores nulos estaban representados por un signo de interrogación (?). A continuación, se graficó la distribución de esta variable para entender mejor su comportamiento en el dataset.

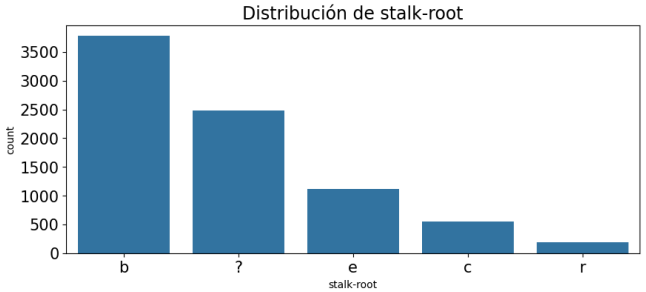

Fig. 1. Gráfico de barras para “stalk_root”

Como se observa en la Figura 1, una gran porción de los datos cuenta con un stalk_root indefinido. Inicialmente, se consideró la posibilidad de eliminar estas instancias, pero se optó por realizar un análisis más profundo a través de una matriz de correlación.

La matriz de correlación reveló una correlación positiva significativa de 0.784 entre los valores nulos en stalk_root_? y el color de las branquias negras (gill_color_b). Esta correlación sugiere que los valores nulos en stalk_root no son completamente aleatorios, sino que podrían estar asociados a características importantes de los hongos, posiblemente relacionados con morfologías específicas que afectan tanto la raíz como el color de las branquias. Por lo tanto, se decidió conservar estos valores nulos en el dataset, ya que podrían contener información útil para la predicción.

Como última observación, se encontró una correlación negativa fuerte de -0.78 entre la ausencia de olor (odor_n) y la comestibilidad de los hongos, indicando que los hongos sin olor tienden a ser comestibles. Esta información fue considerada valiosa y se mantuvo para su inclusión en el modelo.

C. Carga de Datos

Inicialmente, contábamos con una base de datos con 8124 instancias, 22 features o variables independientes y 1 columna de clasificación de clase. Después de la transformación y revisión de los datos contamos con la misma cantidad de instancias y nuestra columna de clasificación de clase pero con 118 features diferentes, cada una representando un valor presente dentro de una variable. Como se mencionó en la sección A, se transformaron las variables donde cada una representa la presencia o ausencia de valor, esto se puede observar al ver la variable cap-shape en la Tabla 1 y en la Figura 2. En este caso, el primer, segundo, cuarto y quinto hongo registrado tenían un “sombrero” convexo y el tercer hongo registrado contaba con un “sombrero” en forma de campana.

class	cap-shape_b	cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	cap-shape_x
1	0.0	0.0	0.0	0.0	0.0	1.0
0	0.0	0.0	0.0	0.0	0.0	1.0
0	1.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	1.0
0	0.0	0.0	0.0	0.0	0.0	1.0

Fig. 2. Representación visual de la transformación de los datos

III. Modelo

A. Justificación

a) Regresión

La regresión logística es un tipo de modelo de clasificación utilizado para predecir la probabilidad de que una observación pertenezca a una de dos categorías posibles [6]. A diferencia de la regresión lineal, que es adecuada para problemas de predicción continua, la regresión logística es ideal para situaciones donde la variable de resultado es binaria, como en este caso, donde queremos predecir si un hongo es comestible o venenoso.

La función central en la regresión logística es la función sigmoide, que mapea cualquier valor real a un rango entre 0 y 1, permitiendo interpretar el resultado como una probabilidad. La fórmula de la función sigmoide se muestra en la Figura 3 [6].

$$f(x) = \frac{1}{1 + e^{-x}}$$

Fig. 3. Fórmula de una sigmoide

El uso de regresión lineal en este contexto no sería adecuado porque la regresión lineal asume una relación lineal entre las variables independientes y la variable dependiente. Sin embargo, en un problema de clasificación binaria, esta suposición no se sostiene, ya que el rango de salida de la regresión lineal no se limita a 0 y 1, como lo hace la función de una sigmoide (Figura 4) [6].

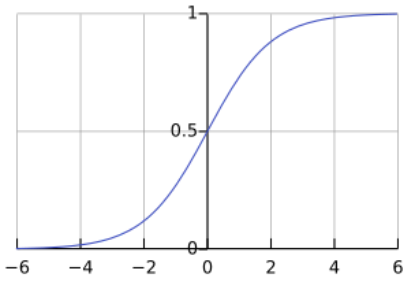

Fig. 4. Función representativa de una sigmoide

b) Gradiente Descendente

El gradiente descendente es un algoritmo de optimización utilizado para minimizar la función de pérdida, ajustando los parámetros del modelo (en este caso, los pesos **w** y el sesgo **b**). En cada iteración, el algoritmo calcula la pendiente de la función de pérdida con respecto a los parámetros y ajusta estos parámetros para reducir el error. El objetivo de este algoritmo es reducir la pendiente de la función de pérdida y acercarse a 0 de esta manera reduciendo el "costo" (Figura 5) [7].

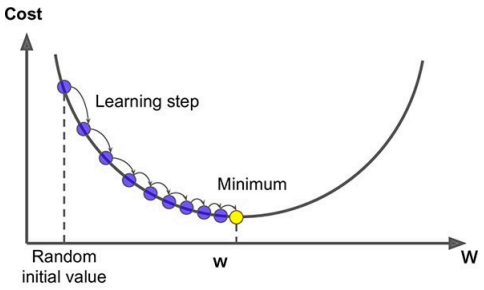

Fig. 5. Representación visual del funcionamiento del gradiente descendente.

c) Entropía Cruzada

La entropía cruzada es una medida de la diferencia entre dos distribuciones de probabilidad. En el contexto de la regresión logística, se utiliza como función de pérdida para evaluar qué tan bien el modelo está prediciendo las probabilidades correctas, de esta manera la función de gradiente descendente nos ayuda a disminuir el error de manera iterativa (Figura 7) [10]. La fórmula de la entropía cruzada para un modelo de clasificación binaria se puede ver en la Figura 6 [8, 9].

$$-\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))$$

Fig. 6. Fórmula de la entropía cruzada

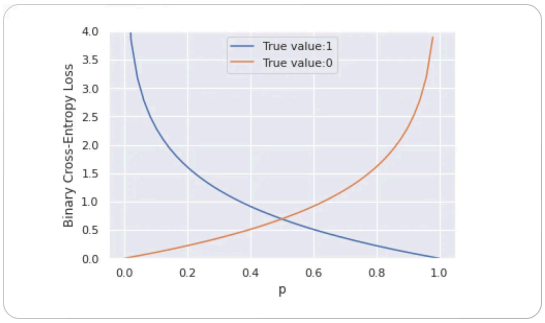

Fig. 7. Representación visual de la entropía cruzada.

B. Preparación de los Datos

En la preparación de los datos, es crucial separar el conjunto de datos en tres subconjuntos: entrenamiento, validación y prueba. Esto es debido a que cuando se lleva a cabo un entrenamiento con el mismo conjunto de datos de prueba no sabrás cómo funcionará el algoritmo con datos con los que no se entrenó el modelo, es decir, los datos reales. El proceso de separar los datos, conocido como split test, implica dividir los datos en dos partes [11]. El conjunto de validación se utiliza para evaluar el modelo durante el entrenamiento y ajustar los hiperparámetros (como la tasa de aprendizaje y el número de épocas). Este enfoque permite prevenir problemas como el sobreajuste, ya que proporciona una evaluación del modelo en datos no vistos,

permitiendo así realizar ajustes antes de la evaluación final en el conjunto de prueba. Este proceso se lleva a cabo en dos etapas:

1. *El conjunto de datos original se divide en dos partes*

- 80% de los datos se utilizan para el conjunto de entrenamiento y validación (train_val).
- 20% de los datos se utilizan para el conjunto de prueba (test).

2. *El conjunto train_val se divide nuevamente*

- 75% de train_val se utiliza para el entrenamiento (train).
- 25% de train_val se utiliza para la validación (validation).

C. Descripción del modelo

En el código se plantearon las funciones y métodos necesarios para llevar a cabo esta implementación. A continuación, describimos las funciones y parámetros utilizados en el modelo, organizándolos en una tabla para mayor claridad (Tabla 2).

Tabla 2. Tabla de Funciones

Nombre	**Función**	**Descripción**
sigmoid(z)	Mapea cualquier valor real a un rango entre 0 y 1 usando la función sigmoide.	Permite que el modelo prediga probabilidades entre 0 y1.
GD (x, y, w, b)	Implementa el gradiente descendente para actualizar los pesos y el sesgo.	Optimiza los parámetros del modelo minimizando la función de pérdida a través de iteraciones.
loss (x, y, w, b)	Calcula la pérdida utilizando entropía cruzada. Los errores son guardados en un arreglo para posteriormente graficarlos.	Proporciona una medida del error del modelo en cada época, que es minimizada durante el entrenamiento.
predict(x, w, b)	Predice la clase de nuevas instancias basándose en los parámetros entrenados.	Utiliza la salida de la función sigmoide para clasificar las instancias como 0 (comestible) o 1 (venenoso).

Parámetros:

Pesos (w): Inicializados en 0 y actualizados durante el entrenamiento.

Sesgo (b): Inicializado en 0 y actualizado junto con los pesos.

Hiperparámetros:

Learning Rate: Tasa de aprendizaje, que controla qué tan grandes son los pasos del gradiente descendente. Inicializado en 0.01.

Epochs: Número de iteraciones para ajustar los pesos del modelo. Una época equivale a una vuelta completa a la base de datos.

IV. Resultados Iniciales

A. Resultados

El modelo fue entrenado en dos configuraciones diferentes, utilizando 3000 y 6000 epochs. Los resultados muestran que con un mayor número de epochs, el modelo logra una mejor precisión en el conjunto de prueba.

Precisión en el conjunto de prueba (3000 epochs): 97.85%

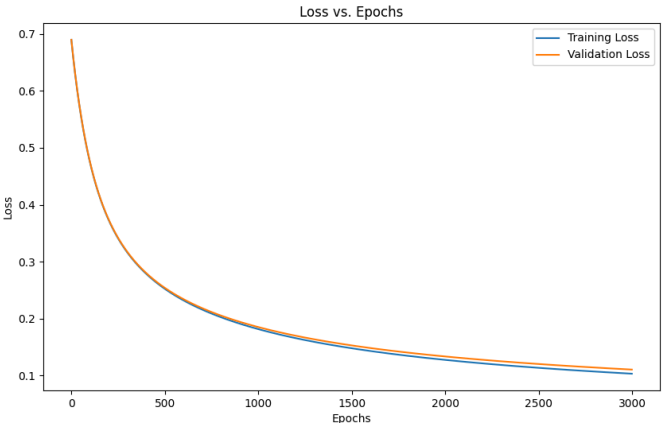

Fig. 8. Gráfica de pérdida a lo largo del tiempo 3000 epochs

Precisión en el conjunto de prueba (6000 epochs): 98.34%

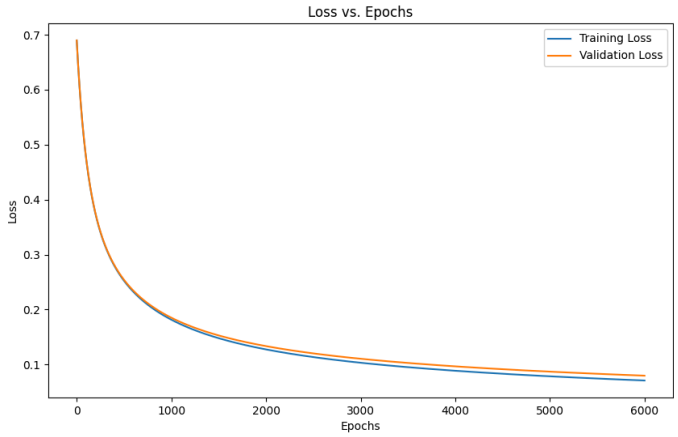

Fig. 9. Gráfica de pérdida a lo largo del tiempo 6000 epochs

Para la evaluación del modelo se decidió graficar una matriz de confusión para ambas pruebas de entrenamiento [12].

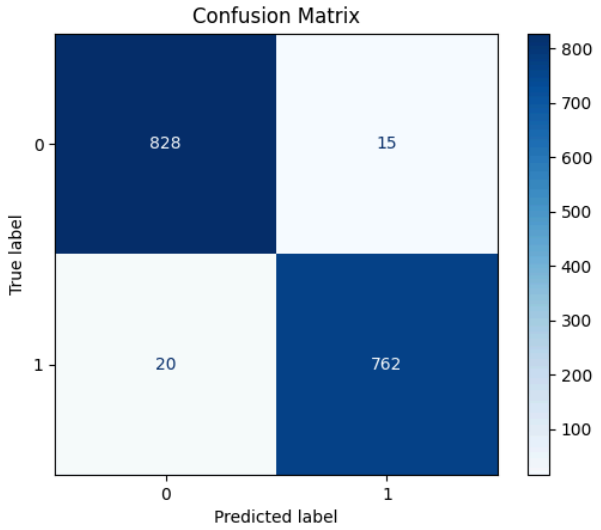

Fig. 10. Gráfica de matriz de confusión 3000 epochs

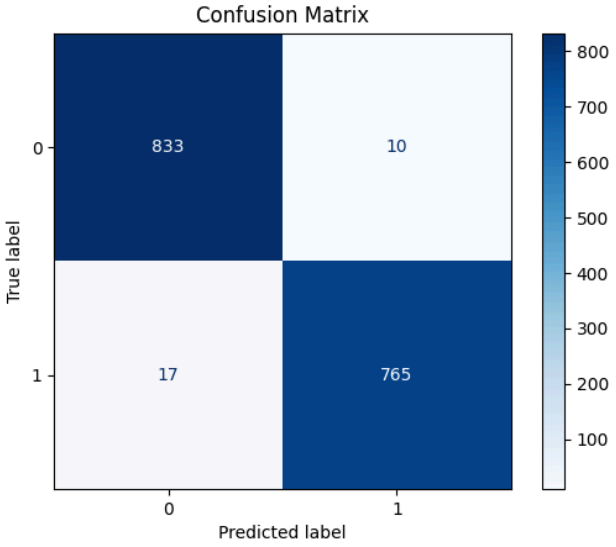

Fig. 11. Gráfica de matriz de confusión 6000 epochs

1. **3000 Epochs**
   TP (True Positives): 828
   TN (True Negatives): 762
   FP (False Positives): 15
   FN (False Negatives): 20

   Final Training Loss: 0.1032
   Final Validation Loss: 0.1105
   Final Test Loss: 0.1116

   Precisión en el conjunto de entrenamiento: 97.83%
   Precisión en el conjunto de validación: 97.48%
   Precisión en el conjunto de prueba: 97.60%

2. **6000 Epochs**
   TP (True Positives): 833
   TN (True Negatives): 765
   FP (False Positives): 10
   FN (False Negatives): 17

   Final Training Loss: 0.0709
   Final Validation Loss: 0.0797
   Final Test Loss: 0.0779

   Precisión en el conjunto de entrenamiento: 98.42%
   Precisión en el conjunto de validación: 97.78%
   Precisión en el conjunto de prueba: 98.28%

B. Interpretación

El aumento en las epochs resulta en una ligera reducción de errores, como lo demuestran las matrices de confusión, con menos falsas predicciones cuando se utilizan 6000 epochs. Sin embargo, la mejora es mínima, lo que sugiere que el modelo podría estar acercándose a su límite de rendimiento con este conjunto de datos. De igual manera, el modelo tiene un muy buen desempeño con ambos conjuntos de epochs, con más de 97% de precisión en ambos casos.

Las matrices de confusión refuerzan esta observación. Con 3000 epochs, el modelo logra identificar correctamente 828 instancias como verdaderos positivos (TP) y 762 como verdaderos negativos (TN). Aumentar a 6000 epochs reduce ligeramente los errores; los falsos positivos (FP) pasan de 15 a 10, y los falsos negativos (FN) de 20 a 17. Estas pequeñas mejoras indican que el modelo está capturando más patrones de los datos con más entrenamiento, pero las diferencias son mínimas.

C. *Diagnóstico final*

La baja tasa de error en ambos conjuntos de epochs, tanto en los verdaderos positivos como en los verdaderos negativos, sugiere que el modelo tiene un bajo sesgo. Está capturando adecuadamente las relaciones en los datos, lo que indica que no está subajustado (underfitting).

Aunque se observa una mejora con 6000 epochs, el modelo no muestra signos evidentes de sobreajuste (overfitting). El hecho de que la diferencia entre la precisión en el conjunto de entrenamiento y el conjunto de prueba sea mínima en ambos casos sugiere que el modelo está desempeñándose de manera efectiva en datos que no ha visto antes. La leve mejora en los falsos positivos y falsos negativos sugiere que el modelo aún está aprendiendo, aunque con mejoras decrecientes.

Es importante destacar que, dado el alto rendimiento del modelo incluso con un número moderado de epochs, se cree que este conjunto de datos es relativamente sencillo de entrenar. La clara separación entre las clases y la naturaleza categórica de las características pueden estar contribuyendo a la facilidad con la que el modelo logra altas precisiones sin requerir configuraciones más complejas o mayor cantidad de ajustes.

V. Modelo Con Frameworks (Random Forest Tree)

A. *Contexto*

Los Random Forest Trees son un conjunto de métodos de aprendizaje de máquina que combinan múltiples árboles de decisión para mejorar la precisión y robustez de un modelo. La idea central es que un "bosque" de árboles, entrenados de manera independiente en diferentes subconjuntos de datos, puede capturar mejor las relaciones entre los datos y reducir la varianza en comparación con un único árbol de decisión. [16]

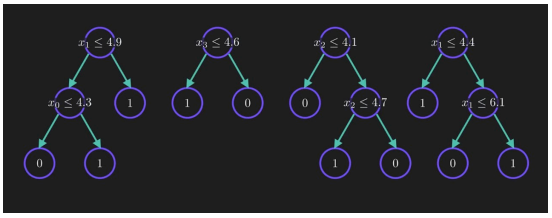

Fig. 12. Representación visual de un Random Forest Tree

Esta técnica se considera una mejora significativa sobre los árboles de decisión tradicionales, ya que estos últimos tienden a sobreajustarse (overfitting) fácilmente, especialmente en conjuntos de datos con muchas características o cuando las relaciones entre características son complejas. Los Random Forest Trees, al promediar las predicciones de múltiples árboles y ser un conjunto de muchos árboles de decisión, logran una generalización [19] mejorada y una menor sensibilidad al ruido en los datos [16].

En este proyecto, se implementó un modelo de Random Forest Tree para evaluar su rendimiento en comparación con la solución manual previamente desarrollada sin el uso de frameworks. El objetivo es determinar si el uso de este enfoque basado en frameworks puede ofrecer una mejora significativa en la precisión y el ajuste del modelo.

B. *Modelo*

El modelo de Random Forest Tree fue implementado utilizando la librería scikit-learn, que proporciona herramientas robustas para la construcción y evaluación de modelos de aprendizaje de máquina.

El conjunto de datos fue dividido en tres subconjuntos: entrenamiento, validación y prueba, manteniendo la misma proporción utilizada en el modelo manual. El 20% de los datos se reservó para el conjunto de prueba, mientras que el 80% restante se dividió en 75% para entrenamiento y 25% para validación. El modelo fue entrenado utilizando el clasificador RandomForestClassifier de scikit-learn, con variaciones en los hiperparámetros max_depth y n_estimators.

Hiperparámetros:

n_estimators: Indica el número de árboles a construir.

max_depth: Indica la profundidad máxima de cada árbol en el bosque.

VI. Resultados (Random Forest Tree)

A. *Resultados*

El primer modelo fue entrenado sin hiperparámetros definidos y al segundo se le aplicaron parámetros. Los resultados demuestran el potencial del algoritmo debido a sus resultados perfectos.

Prueba 1: Precisión en el conjunto de prueba: 100%

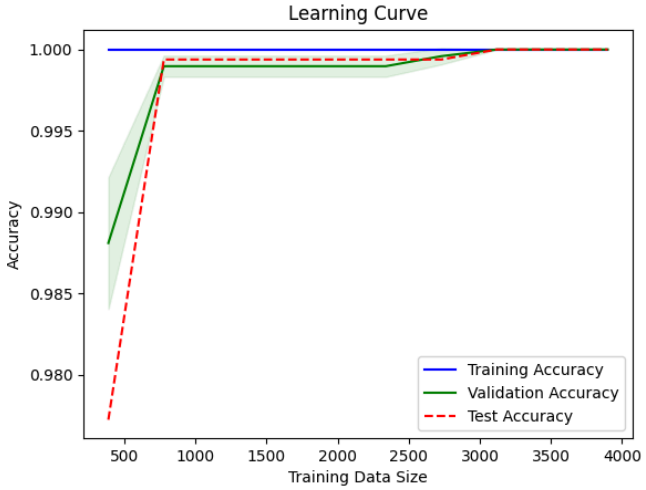

Fig. 13. Gráfica de precisión primera prueba RFT

Prueba 2: Precisión en el conjunto de prueba: 100%

n_estimators = 50
max_depth = 10
random_state = 42

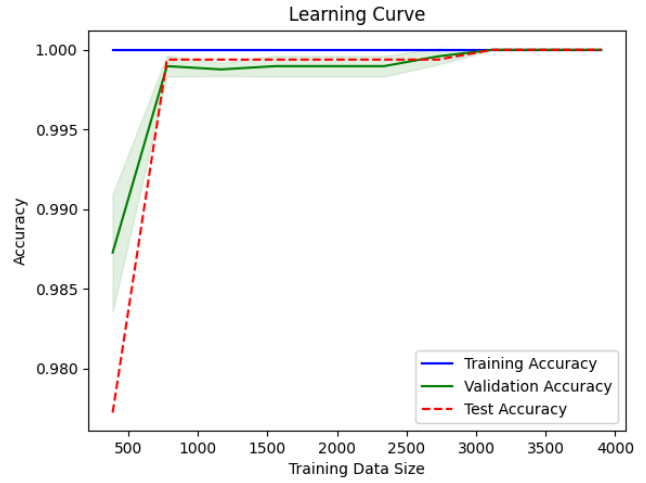

Fig. 14. Gráfica de precisión segunda prueba RFT

Para la evaluación de ambos modelos también se graficaron matrices de confusión, sin embargo, debido al 100% de precisión ambas son iguales [12].

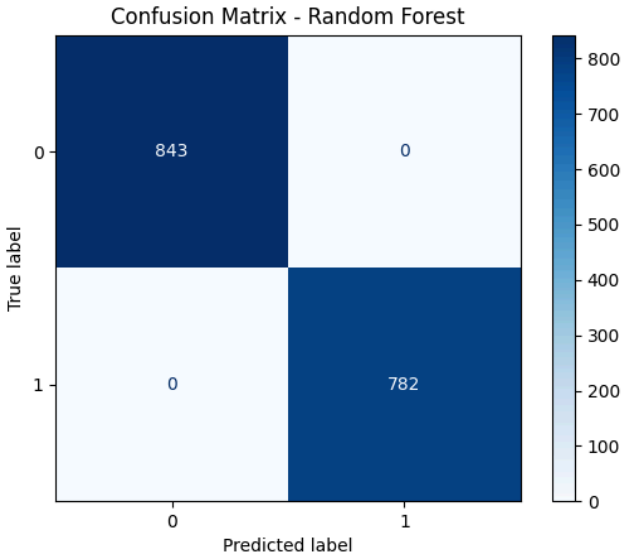

Fig. 15. Gráfica de matriz de confusión modelos RFT

Pruebas RTF
TP (True Positives): 843
TN (True Negatives): 782
FP (False Positives): 0
FN (False Negatives): 0

B. Interpretación

Los resultados obtenidos con el modelo de Random Forest Tree demuestran un rendimiento excepcional, alcanzando una precisión del 100% en el conjunto de prueba en ambas pruebas. Esto significa que el modelo fue capaz de clasificar correctamente todas las instancias de hongos como comestibles o venenosos sin cometer errores. Tanto los verdaderos positivos (TP) como los verdaderos negativos (TN) fueron identificados correctamente, mientras que no hubo falsos positivos (FP) ni falsos negativos (FN).

La alta precisión obtenida con Random Forest Tree respalda la hipótesis de que el conjunto de datos es relativamente sencillo de entrenar. Por otro lado, este resultado también demuestra la eficacia del algoritmo Random Forest para abordar problemas de clasificación binaria en datos categóricos, siendo capaz de reducir la varianza y el sesgo, evitando un mal ajuste.

Las líneas de precisión en el conjunto de validación y prueba siguen una trayectoria similar, acercándose al 100% después de un número inicial de muestras de entrenamiento (Figura 13). La pequeña diferencia observada al inicio entre el conjunto de entrenamiento y el de validación indica que el modelo tuvo una ligera dificultad para generalizar inicialmente, pero esto se corrige rápidamente a medida que se agregan más datos de entrenamiento.

Es relevante señalar que el área sombreada en la gráfica representa la variabilidad en la precisión. A medida que el tamaño del conjunto de entrenamiento aumenta, esta variabilidad disminuye, lo que sugiere que el modelo se vuelve más estable con más datos.

Así mismo, en la segunda gráfica (Figura 14), se observa un comportamiento muy similar al de la prueba anterior. Al aplicar los hiperparámetros n_estimators: 50, max_depth: 10, y random_state: 42, la curva de precisión del conjunto de entrenamiento alcanza rápidamente el 100%, y las del conjunto de validación y prueba se estabilizan en torno al 100% también.

La principal diferencia en esta gráfica es que la curva de precisión en el conjunto de validación es más suave y cercana a la de entrenamiento desde el principio, lo que indica que los ajustes en los hiperparámetros permitieron al modelo aprender de manera más eficiente y generalizar mejor desde el principio.

C. Diagnóstico final

Dado que el modelo de Random Forest Tree ha alcanzado una precisión del 100% en las pruebas, no hay indicios de que el modelo esté subajustado o sobreajustado. Los errores son inexistentes en todas las categorías, lo que implica que no hay necesidad de ajustes adicionales. En este caso, la naturaleza del conjunto de datos ha permitido que el modelo logre un fitting perfecto.

VII. Mejora de Modelo Inicial

Dado que los resultados obtenidos con el modelo de Random Forest Tree (RFT) muestran una precisión del 100% en el conjunto de prueba, podemos concluir que el conjunto de datos es relativamente sencillo de clasificar. El modelo de RFT logra un rendimiento perfecto con configuraciones comunes o sin ajustar parámetros, lo que indica que las características del dataset permiten una clara separación entre clases, facilitando el proceso de aprendizaje.

A partir de esta conclusión, surge la justificación de intentar mejorar el modelo manual que se desarrolló inicialmente con gradiente descendente. Aunque este modelo manual alcanzó un alto nivel de precisión (superior al 97%), existe margen para intentar optimizarlo y igualar el rendimiento del 100% logrado por el modelo basado en frameworks.

El enfoque de esta mejora se centrará en el ajuste los hiperparámetros del modelo manual con el objetivo de reducir aún más los errores de entrenamiento y validación. También se realizará un análisis detallado de los errores en ambos conjuntos de datos para evaluar la varianza y el sesgo.

Sabemos que la varianza mide la sensibilidad del modelo a los cambios en los datos de entrenamiento. Un modelo con alta varianza tiende a sobreajustarse a los datos de entrenamiento, mostrando un buen rendimiento en este conjunto pero un rendimiento pobre en el conjunto de validación o prueba. El sesgo, por otro lado, mide los errores sistemáticos que comete el modelo. Un alto sesgo indica que el modelo no está capturando bien la relación en los datos, lo que resulta en un subajuste.

Para evaluar el ajuste del modelo, se implementó un diagnóstico basado en la comparación de las pérdidas (losses) en los conjuntos de entrenamiento, validación y evaluación. Este diagnóstico se llevó a cabo al final del proceso de entrenamiento, calculando las pérdidas finales de todos los conjuntos.

A. Criterios de Evaluación

Underfitting (Alto Sesgo, Baja Varianza): Si se observan errores altos en los conjuntos de entrenamiento, validación y el de prueba, indica que el modelo no está capturando bien las relaciones en los datos. [13-15]

Overfitting (Bajo Sesgo, Alta Varianza): Si el error en el conjunto de entrenamiento es bajo pero los errores en el conjunto de validación y en el de prueba son significativamente mayores, esto indica que el modelo está sobreajustado, capturando ruido en los datos de entrenamiento pero fallando en generalizar a nuevos datos. [13-15]

Appropriate Fitting: Si los errores son moderadamente bajos en todos los conjuntos de datos, esto indica que el modelo está capturando bien las relaciones en los datos y generaliza adecuadamente. [13-15]

B. Justificación

Para realizar el diagnóstico y entender mejor el fitting de nuestro modelo, se realizaron tres pruebas con diferentes número de épocas: 1000, 3000 y 6000 epochs. Este enfoque nos permite analizar cómo las épocas afectan el desempeño del modelo y determinar si estamos ante un caso de subajuste (underfitting), sobreajuste (overfitting) o ajuste adecuado (appropriate fitting). Cabe resaltar que por el momento estamos haciendo pruebas control sin modificar el parámetro de aprendizaje, eso no significa que el error dependa de la cantidad de épocas que el modelo entrenará. El propósito de esto es mediante pruebas identificar áreas de oportunidad en el modelo.

Las preguntas que tenemos en este momento y queremos resolver son: ¿cómo sabemos qué tan altos o bajos están los errores comparados unos con otros si mi modelo ya obtuvo precisiones adecuadas sin señales de un mal ajuste? ¿Se puede mejorar el modelo? Queremos observar si con el cambio de épocas logramos identificar un cambio significativo en los valores como para poder hacer un análisis entre errores que nos permita identificar un área de oportunidad. Si los errores a lo largo de las 3 pruebas se comportan igual en todos los subconjuntos de datos entonces inferimos que el modelo tiene ajuste adecuado en todos los casos a pesar de la cantidad de épocas y no se puede mejorar.

C. *Resultados*

1. 1000 Epochs

Final Training Loss: 0.1817
Final Validation Loss: 0.1853
Final Test Loss: 0.1912

Precisión en el conjunto de entrenamiento: 95.61%
Precisión en el conjunto de validación: 95.57%
Precisión en el conjunto de prueba: 95.14%

2. 3000 Epochs

Final Training Loss: 0.1033
Final Validation Loss: 0.1105
Final Test Loss: 0.1116

Precisión en el conjunto de entrenamiento: 97.83%
Precisión en el conjunto de validación: 97.48%
Precisión en el conjunto de prueba: 97.60%

3. 6000 Epochs

Final Training Loss: 0.0710
Final Validation Loss: 0.0797
Final Test Loss: 0.0780

Precisión en el conjunto de entrenamiento: 98.42%
Precisión en el conjunto de validación: 97.78%
Precisión en el conjunto de prueba: 98.28%

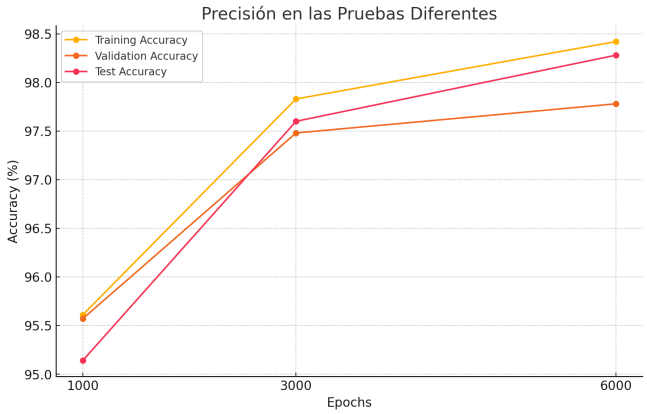

Fig. 16. Gráfica precisión a lo largo de las pruebas

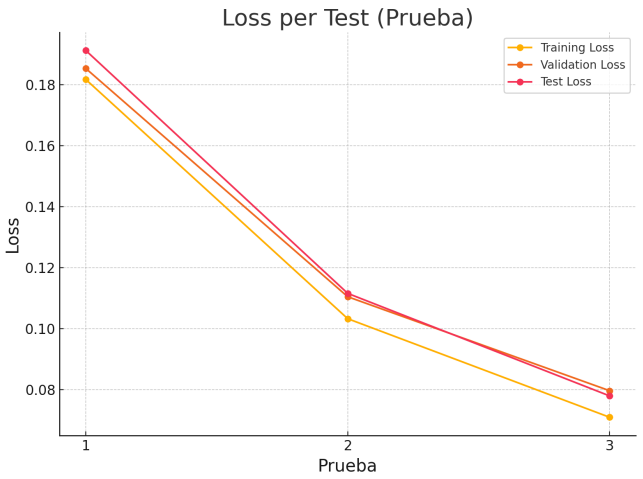

Fig. 17. Gráfica de pérdida a lo largo de las pruebas

TABLA 3. TABLA DE DIFERENCIA DE ERRORES

Prueba	1	2	3
Training Loss	0.1817	0.1033	0.071
Validation Loss	0.1853	0.1105	0.0797
Test Loss	0.1912	0.1116	0.0780
Diferencia (Validation vs Training)	0.0036	0.0072	0.0087
Diferencia (Test vs Validation)	0.0059	0.0011	- 0.0017

D. *Interpretación*

Las diferentes pruebas realizadas proporcionan una visión integral sobre el comportamiento del modelo y su ajuste. A medida que se incrementa el número de épocas, se observa una mejora consistente en la precisión (Figura 16) y un ligero decremento en los valores de pérdida (Tabla 3), lo que nos permite inferir el fitting del modelo.

En la prueba con 1000 épocas, el modelo alcanzó una precisión del 95.14% en el conjunto de prueba, con valores de pérdida más altos en comparación con las pruebas de 3000 y 6000 épocas. Esto sugiere que el modelo podría no haber tenido tiempo suficiente para aprender completamente de los datos.

Aumentando a 3000 épocas, se observa una mejora significativa en la precisión, alcanzando el 97.60% en el conjunto de prueba. La pequeña diferencia entre las pérdidas en los conjuntos de entrenamiento, validación y prueba sugieren que el modelo está bien ajustado, sin embargo, la

diferencia entre validación y entrenamiento es mayor a la prueba anterior.

Finalmente, con 6000 épocas, el modelo logra su mejor rendimiento, alcanzando una precisión del 98.28% en el conjunto de prueba. Los valores de pérdida entre los conjuntos de entrenamiento y validación se separan aún más. De igual manera, la diferencia de errores entre el test y el validación set aumentó (en valor absoluto) y no disminuyó como en la prueba de 3000 epochs. Estas dos observaciones pueden indicar que el modelo está tendiendo hacia un muy ligero sobreajuste (overfitting) dentro del pequeño margen de errores que estamos analizando.

VIII. Mejora de Modelo Inicial II

Como se pudo observar en los resultados anteriores, el análisis de las pérdidas y precisiones sugiere que el modelo no tiene necesariamente un mal ajuste. El único problema es que al aumentar el número de épocas, el modelo no muestra una mejora positiva en la diferencia de las pérdidas de los diferentes conjuntos a pesar de que los errores si disminuyen (Figura 17). La gráfica de precisión en las diferentes pruebas (Figura 16) muestra una tendencia clara de mejora en la precisión conforme aumentan las épocas de entrenamiento, tanto para los conjuntos de entrenamiento, validación y prueba. Sin embargo, se puede observar una reducción en la pendiente de mejora, lo que sugiere que entrenar por más épocas después de las 6000 no tendría un impacto significativo en la mejora del modelo.

Dado el diagnóstico, se decidió optar por probar un ajuste en la tasa de aprendizaje. La tasa de aprendizaje es un hiperparámetro crucial en el proceso de optimización mediante gradiente descendente, ya que controla el tamaño de los pasos que el algoritmo toma para minimizar la función de pérdida. Si la tasa de aprendizaje es demasiado baja, el modelo puede aprender de manera excesivamente lenta, lo que impide que alcance un rendimiento óptimo en un tiempo razonable. Por otro lado, si la tasa de aprendizaje es demasiado alta, el modelo puede oscilar alrededor del mínimo global o incluso divergir. Con el objetivo de prueba y mejorar la capacidad del modelo, se decidió ajustar la tasa de aprendizaje para optimizar el proceso de entrenamiento de 0.01 a 0.1 [13].

A. Resultados

El modelo nuevamente fue entrenado en dos configuraciones diferentes, utilizando 3000 y 6000 epochs, dejando a un lado la prueba de 1000 epochs.

Precisión en el conjunto de prueba (3000 epochs): 99.88%

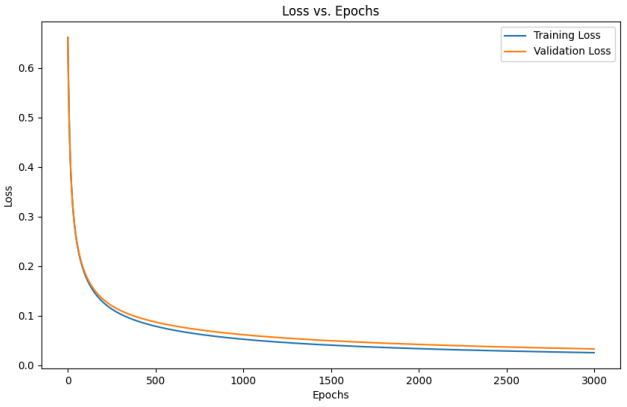

Fig. 18. Gráfica de pérdida a lo largo del tiempo 3000 epochs

Precisión en el conjunto de prueba (6000 epochs): 99.88%

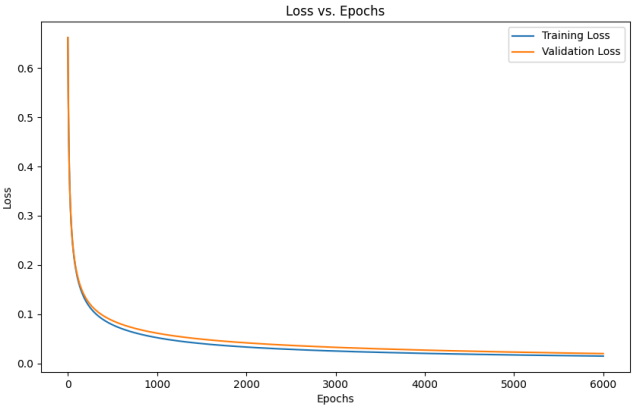

Fig. 19. Gráfica de pérdida a lo largo del tiempo 6000 epochs

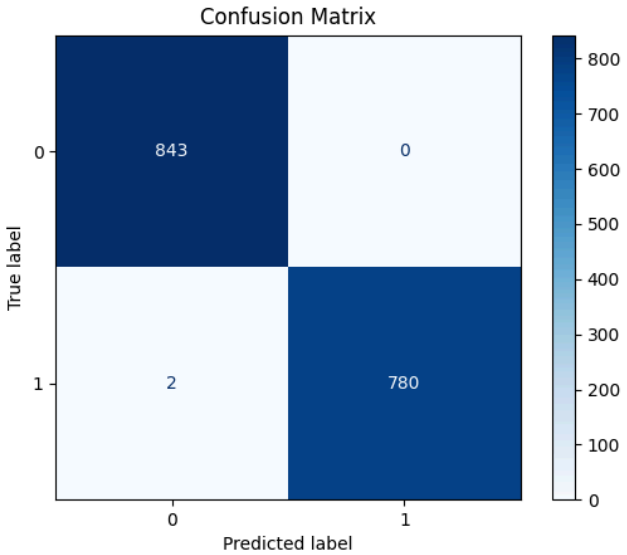

Fig. 20. Gráfica de matriz de confusión 3000 y 6000 epochs

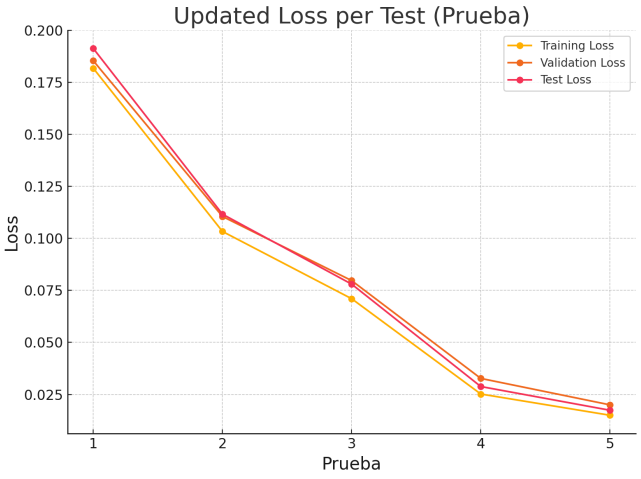

Fig. 21. Gráfica actualizada de pérdida a lo largo de las pruebas

4. **3000 Epochs**
   TP (True Positives): 843
   TN (True Negatives): 780
   FP (False Positives): 0
   FN (False Negatives): 2

   Final Training Loss: 0.0252
   Final Validation Loss: 0.0327
   Final Test Loss: 0.0288

   Precisión en el conjunto de entrenamiento: 99.84%
   Precisión en el conjunto de validación: 99.63%
   Precisión en el conjunto de prueba: 99.88%

5. **6000 Epochs**
   TP (True Positives): 843
   TN (True Negatives): 780
   FP (False Positives): 0
   FN (False Negatives): 2

   Final Training Loss: 0.0150
   Final Validation Loss: 0.0200
   Final Test Loss: 0.0174

   Precisión en el conjunto de entrenamiento: 99.84%
   Precisión en el conjunto de validación: 99.63%
   Precisión en el conjunto de prueba: 99.88%

B. Interpretación

Después de realizar un ajuste en la tasa de aprendizaje de 0.01 a 0.1, los resultados obtenidos mostraron una mejora en el rendimiento del modelo en ambos casos, logrando una precisión del 99.88% en el conjunto de prueba, lo que representa una mejora notable en comparación con los resultados anteriores.

A través de las pruebas realizadas con 3000 y 6000 épocas con un learning rate de 0.1, se observa una consistencia en los porcentajes de precisión en los tres subconjuntos de datos. La precisión en el conjunto de prueba alcanza un máximo de en ambos casos, mientras que la diferencia de errores en los diferentes conjuntos de datos disminuye a medida que aumentamos las épocas (Tabla 4). Por otro lado, vemos también cómo hemos logrado que los errores en los conjuntos de datos se comporten de manera muy similar, acercándose unos con otros (Figura 21). Esta consistencia y patrón indica que el modelo ha alcanzado un nivel óptimo de ajuste, donde el incremento en las épocas ya no produce mejoras en la precisión y se disminuyeron los errores de los conjuntos sin aumentar la diferencia entre ellos (Tabla 4).

TABLA 4. TABLA DE DIFERENCIA DE ERRORES ACTUALIZADA

Prueba	1 (1k epochs - 0.01 lr)	2 (3k epochs - 0.01 lr)	3 (6k epochs - 0.01 lr)	4 (3k epochs - 0.1 lr)	5 (6k epochs - 0.1 lr)
Training Loss	0.1817	0.1033	0.071	0.0252	0.0150
Validation Loss	0.1853	0.1105	0.0797	0.0327	0.02
Test Loss	0.1912	0.1116	0.0780	0.0288	0.0174
Diferencia (Validation vs Training)	0.0036	0.0072	0.0087	0.0075	0.005
Diferencia (Test vs Validation)	0.0059	0.0011	- 0.0017	-0.0039	-0.0026

C. Diagnóstico final

Con el ajuste de la tasa de aprendizaje a 0.1, el modelo muestra una notable mejora en la reducción del error tanto en los conjuntos de validación como de prueba en comparación con las pruebas realizadas con la tasa de 0.01. En ambas pruebas, no se observan señales de overfitting ni de underfitting, ya que los errores son bajos en todos los conjuntos y las diferencias en los conjuntos de datos dejan de incrementar.

La modificación en la tasa de aprendizaje ha demostrado ser una mejora efectiva en el ajuste del modelo. Los nuevos resultados indican que el modelo ha alcanzado un nivel óptimo de ajuste, manteniendo la misma precisión en el modelo así como en los valores obtenidos en la matriz de confusión. Esta falta de mejora adicional en la matriz de confusión refuerza la conclusión de que el modelo ha alcanzado su capacidad máxima de entrenamiento con el modelo sin uso de frameworks.

IX. MEJORA DE MODELO RFT (ADICIONAL)

Después de obtener resultados perfectos con la implementación inicial del modelo Random Forest Tree

(RFT), se decidió profundizar en el análisis y optimización del algoritmo. A pesar de que el modelo alcanzó una precisión del 100% sin ajustes profundos en los hiperparámetros, el objetivo de esta sección es explorar cómo la variación en dichos parámetros afecta el ajuste y la capacidad de generalización del modelo, profundizando en su funcionamiento.

Debido a la complejidad inherente del algoritmo de Random Forest Tree y su alto rendimiento inicial, fue necesario limitar ciertos aspectos del modelo para llevar a cabo este análisis. Se probaron diferentes combinaciones de estos hiperparámetros para evaluar cómo impactan en la precisión del modelo. Para cada combinación, se calcularon las precisiones en los conjuntos de entrenamiento y validación. Además, se utilizó un heatmap para visualizar los scores obtenidos, lo que permitió identificar rápidamente las combinaciones de parámetros que ofrecían un balance óptimo entre el ajuste del modelo y la generalización. Posteriormente, se seleccionó el modelo con mejor rendimiento en el conjunto de validación para evaluar su precisión en el conjunto de prueba [18].

C. Resultados iniciales

Como se mencionó en la sección anterior, se realizaron diferentes pruebas de entrenamiento. En total fueron 9 pruebas diferentes, cada una con una combinación de hipeparámetros diferente (Tabla 5).

TABLA 5. PRUEBAS PARA MODELO CON FRAMEWORKS

Prueba	n_estimators (eje x)	max_depth (eje y)
1	1	1
2	5	1
3	10	1
4	1	3
5	5	3
6	10	3
7	1	5
8	5	5
9	10	5

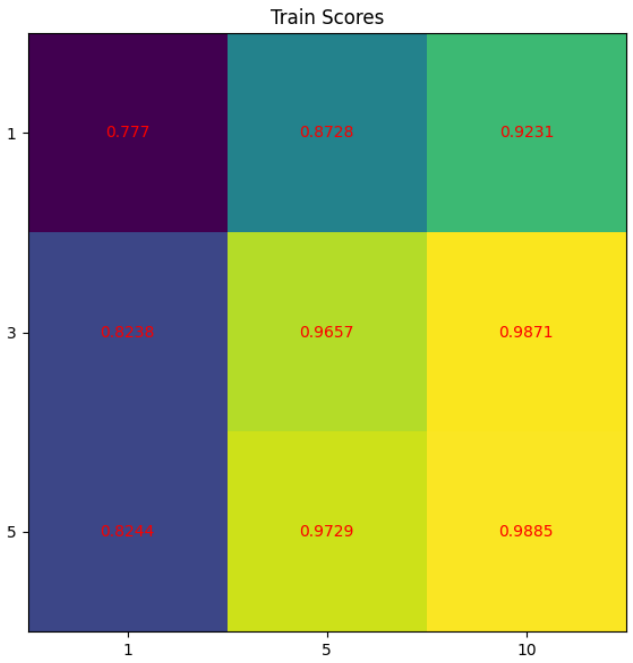

Fig. 22. Resultados de precisión para el set de entrenamiento

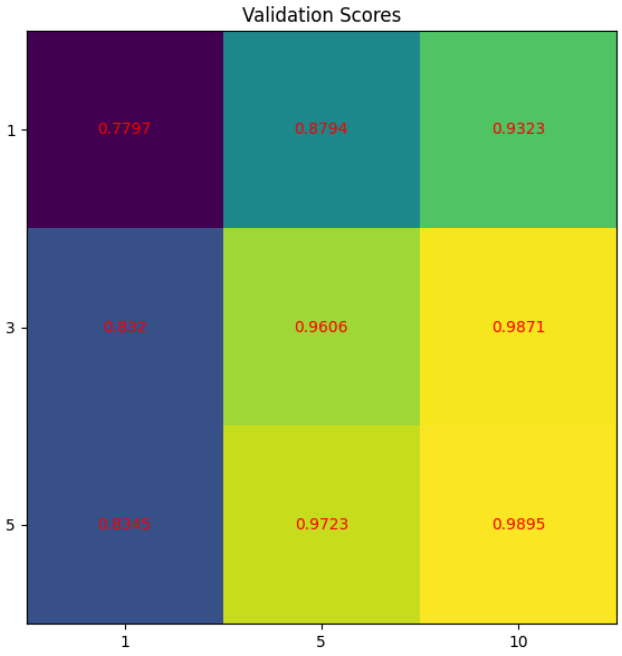

Fig. 23. Resultados de precisión para el set de validación

D. Interpretación

Los resultados obtenidos en las pruebas sugieren que el modelo de Random Forest Tree alcanza un rendimiento óptimo con configuraciones específicas de los hiperparámetros y se puede ver su mejora a lo largo de las pruebas.

En la prueba número 1 (max_depth = 1, n_estimators = 1) ambos scores (train y validation) fueron relativamente bajos, lo que indica que el modelo está subajustado (underfitting). Esto se debe a que la baja profundidad de los

árboles no permite capturar la complejidad de los datos. Sin embargo, en la prueba número 3 (max_depth = 1, n_estimators = 10), se observa una mejora significativa en ambos scores, lo que sugiere que un mayor número de estimadores ayuda a capturar mejor los patrones. A pesar de esta mejora, el modelo sigue siendo relativamente simple debido a la baja profundidad.

En la prueba 5 (max_depth = 3, n_estimators = 5) los scores son un poco más altos y muy cercanos entre sí, indicando un ajuste adecuado. Con esta información podemos asumir que el modelo tiene la complejidad suficiente para capturar los patrones sin sobreajustar los datos. Ahora bien, en la prueba número 9 (max_depth = 5, n_estimators = 10), ambos scores son extremadamente altos y casi idénticos, lo que indica que el modelo se ajusta muy bien a los datos sin señales de sobreajuste o subajuste.

El análisis de los resultados sugiere que las configuraciones con max_depth = 3 o 5 y un número razonable de n_estimators = 5 o 10 ofrecen el mejor rendimiento.

E. Diagnóstico final

Aunque el modelo alcanzó un ajuste aparentemente "óptimo" con max_depth = 5 y n_estimators = 10, se realizó una validación adicional para confirmar este ajuste. Se implementó un nuevo código para graficar la curva de aprendizaje del modelo, evaluando cómo los scores de entrenamiento y validación se comportan a medida que aumenta el tamaño del conjunto de datos.

Este análisis es crucial para confirmar que el modelo no tiene un mal ajuste y que su rendimiento se mantiene estable al aumentar la cantidad de datos de entrenamiento.

F. Validación

En este nuevo código se realizaron dos pruebas en el intento de saber el ajuste del modelo. La primera validación fue una matriz de confusión (Figura 24), por otro lado, la segunda validación fue la visualización de la curva de aprendizaje del modelo de entrenamiento como la de validación (Figura 25).

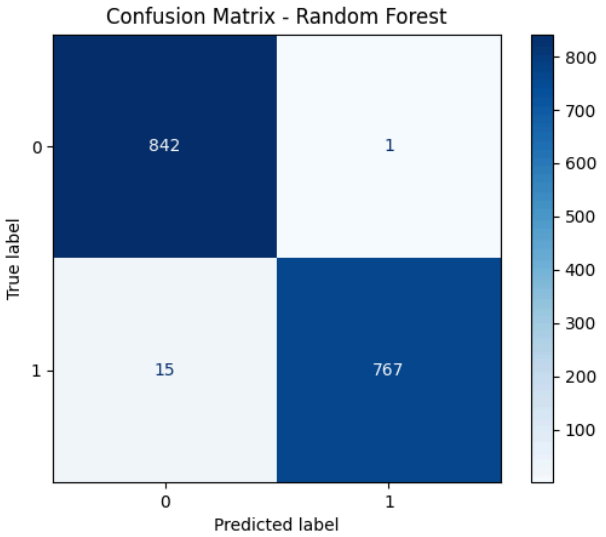

Fig. 24. Matriz de confusión para prueba "óptima" (max_depth = 5 y n_estimators = 10)

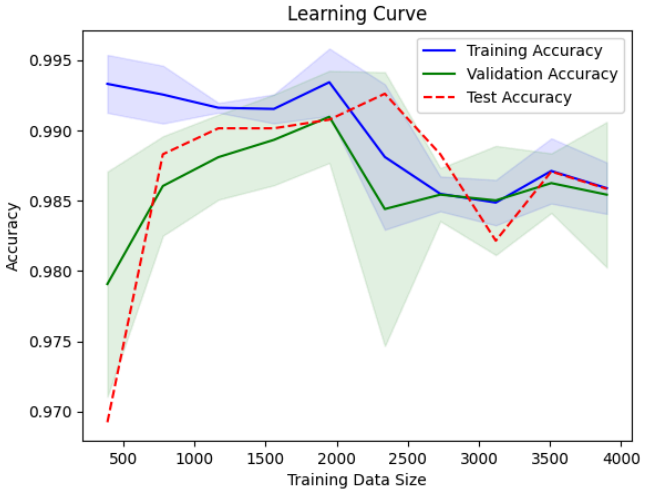

Fig. 25. Curva de aprendizaje para prueba "óptima" (max_depth = 5 y n_estimators = 10)

G. Interpretación

Después de evaluar la gráfica de la curva de aprendizaje y la matriz de confusión, es posible sugerir que el modelo no tiene un ajuste completamente adecuado. Un posible indicador es que tanto la precisión en los subconjuntos de datos tienden a disminuir ligeramente cuando el tamaño de los datos de entrenamiento aumenta. Esto podría ser una señal de que el modelo, con estos hiperparámetros limitando al mismo, está siendo "sobreajustado" al agregar más datos. Aunque las curvas están bastante cerca, hay variabilidad en las primeras etapas con un menor tamaño de datos de entrenamiento. Idealmente, en un fitting adecuado, las dos curvas convergen y permanecen estables a medida que aumenta el tamaño del conjunto de entrenamiento.

Debido a este análisis, aunque el modelo muestre un score muy bueno, es posible que estemos enfrentando un problema. Por esta razón, se realizaron dos pruebas adicionales, aumentando la complejidad del modelo.

H. *Resultados post-validación*

Para explorar más a fondo el ajuste del modelo, se realizaron dos pruebas adicionales con configuraciones aumentadas de hiperparámetros.

Prueba 1

n_estimators = 20
max_depth = 5
random_state = 42
Precisión en el conjunto de prueba: 99.08%

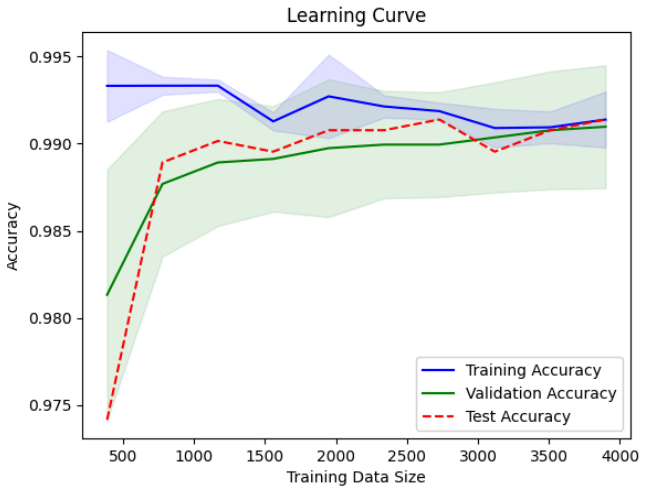

Fig. 26. Curva de aprendizaje prueba 1 post-validación

Prueba 2

n_estimators = 20
max_depth = 10
random_state = 42
Precisión en el conjunto de prueba: 100.00%

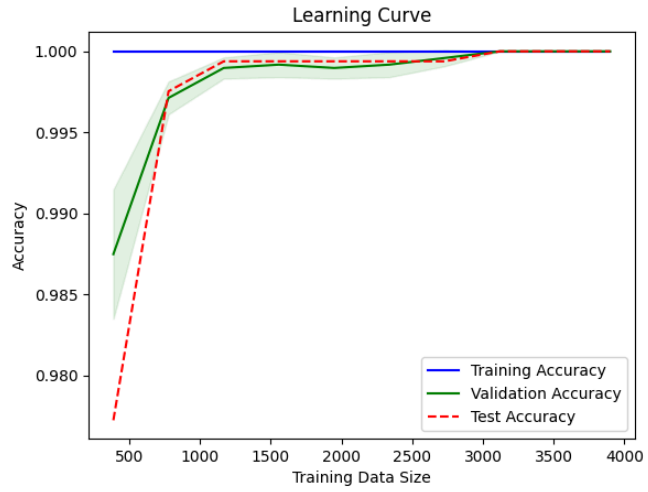

Fig. 27. Curva de aprendizaje prueba 2 post-validación

I. *Interpretación y diagnóstico final*

Las diferentes pruebas con Random Forest Tree muestran que es posible obtener un ajuste adecuado mediante la combinación correcta de hiperparámetros, con una configuración óptima alcanzada con un mayor número de estimadores y mayor profundidad en los árboles.

En la prueba número 1 post validación (Figura 26), la curva de aprendizaje muestra una ligera mejora, pero sigue habiendo una pequeña variabilidad en las primeras etapas con un tamaño de conjunto de datos más pequeño en comparación a la prueba 2 (Figura 27).

La curva de precisión en la segunda prueba (Figura 27) alcanza el 100%. Este modelo muestra un ajuste adecuado, logrando un rendimiento perfecto con un mayor número de estimadores y una mayor profundidad de árboles. No se observan señales de sobre o subajuste. Gracias a los resultados que obtuvimos con este algoritmo en un inicio, es posible que la alta precisión se deba a la naturaleza simple de los datos que estamos manejando.

Si las características utilizadas para entrenar el modelo son extremadamente informativas o tienen una relación directa con la variable objetivo, incluso un modelo simple puede alcanzar alta precisión. Así mismo, otro factor muy importante a considerar es que la redundancia en los datos es evidente debido al uso de one hot encoding. Esta redundancia facilita al modelo capturar las relaciones entre las características y la clase, lo que lleva a un alto rendimiento. Tomando esto en cuenta, podemos decir que el modelo ha conseguido un score perfecto pero no es debido a un mal ajuste en el modelo si no que se ha llegado al límite de entrenamiento, por lo que el ligero incremento en la complejidad del modelo ha ayudado a mejorar el rendimiento del mismo.

X. Conclusiones

En este proyecto, se implementó manualmente un algoritmo de regresión logística utilizando gradiente descendente para clasificar hongos como comestibles o venenosos. A lo largo del desarrollo del modelo, se realizaron varias iteraciones y ajustes, incluyendo el análisis del fitting del modelo y la optimización de la tasa de aprendizaje.

Los resultados finales muestran que el modelo sin frameworks ha alcanzado un ajuste adecuado, con una precisión del 99.88% en el conjunto de prueba tanto con 3000 como con 6000 épocas, sin signos de sobreajuste o subajuste. La mejora en la tasa de aprendizaje fue crucial para lograr este rendimiento, permitiendo al modelo capturar de manera efectiva las relaciones en las diferentes variables.

Por otro lado, el análisis adicional con Random Forest Trees sugirió un posible sobreajuste en las pruebas iniciales y cuando se aumentó la complejidad del modelo, alcanzando una precisión perfecta. Aunque esta precisión puede deberse

a la naturaleza de los datos altamente separables y características muy informativas, también indica la importancia de la validación continua y la optimización cuidadosa de los hiperparámetros.

Este proyecto no solo demuestra la capacidad de ambos modelos para realizar predicciones precisas en problemas de clasificación binaria, sino que también subraya la importancia de la optimización de hiperparámetros y la evaluación continua del modelo para garantizar su rendimiento óptimo. Con estos resultados, los modelos se presentan como una herramienta robusta y confiable para la clasificación de hongos, con potencial para ser aplicado en otros contextos biológicos.

Anexos

1. Repositorio de GitHub:
https://github.com/AdrianGalvanDiaz/Proyecto_ML/tree/branch1

Referencias

[1] Evolución de la Inteligencia Artificial. (s. f.). Iberdrola. https://www.iberdrola.com/innovacion/evolucion-inteligencia-artificial

[2] ¿Qué es machine learning? | Definición, tipos y ejemplos | SAP. (s. f.). SAP. https://www.sap.com/latinamerica/products/artificial-intelligence/what-is-machine-learning.html#:~:text=El%20machine%20learning%20es%20un,experiencia%20sin%20ser%20programadas%20expl%C3%ADcitamente.

[3] Richardson, D. (2024, 8 mayo). La IA y el ML y su importancia para las empresas. Blog de Red Hat. https://www.redhat.com/es/blog/what-aiml-and-why-does-it-matter-your-business#:~:text=La%20inteligencia%20artificial%2C%20el%20aprendizaje,en%20las%20funciones%20del%20sistema.

[4] UCI Machine Learning Repository. (s. f.). https://archive.ics.uci.edu/dataset/73/mushroom

[5] Novack, G. (2024, 30 abril). Building a One Hot Encoding Layer with TensorFlow - Towards Data Science. Medium. https://towardsdatascience.com/building-a-one-hot-encoding-layer-with-tensorflow-f907d686bf39

[6] ¿Qué es la regresión logística? - Explicación del modelo de regresión logística - AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/logistic-regression/

[7] Andrés, D., & Andrés, D. (2023, 18 enero). Gradiente descendente - ML Pills. ML Pills - Machine Learning Pills. https://mlpills.dev/machine-learning-es/gradiente-descendente/

[8] The cross-entropy error function in neural networks. (s. f.). Data Science Stack Exchange. https://datascience.stackexchange.com/questions/9302/the-cross-entropy-error-function-in-neural-networks

[9] Binary Cross Entropy: Where to use log loss in model Monitoring. (2023, 9 junio). Arize AI. https://arize.com/blog-course/binary-cross-entropy-log-loss/

[10] Shah, D. (2024, 2 julio). Cross entropy loss: intro, applications, code. V7. https://www.v7labs.com/blog/cross-entropy-loss-guide

[11] How To Choose The Right Test Options When Evaluating Machine Learning Algorithms. (s. f.). Machine Learning Mastery. https://machinelearningmastery.com/how-to-choose-the-right-test-options-when-evaluating-machine-learning-algorithms/

[12] Markham, K. (2024, 4 junio). Simple guide to confusion matrix terminology. Data School. https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

[13] GeeksforGeeks. (2024, 11 marzo). ML | Underfitting and Overfitting. GeeksforGeeks. https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

[14] NStatum. (2022, 12 agosto). Underfitting & overfitting - explained [Vídeo]. YouTube. https://www.youtube.com/watch?v=o3DztvnfAJg

[15] Anarthal. (2020, 15 julio). Underfitting, overfitting and model complexity. Anarthal Kernel. https://anarthal.github.io/kernel/posts/underfitting-overfitting/

[16] Normalized Nerd. (2021, 21 abril). Random Forest algorithm clearly explained! [Vídeo]. YouTube. https://www.youtube.com/watch?v=v6VJ2RO66Ag

[17] TechTour. (2019, 4 abril). Logistic Regression vs Random Forest Classifier [Vídeo]. YouTube. https://www.youtube.com/watch?v=goPiwckWE9M

[18] Yclaudel. (2020, 2 enero). Overfitting vs Underfitting with RandomForest. Kaggle. https://www.kaggle.com/code/yclaudel/overfitting-vs-underfitting-with-randomforest

[19] Qué es Generalización Concepto y definición. Glosario. (s. f.). GAMCO, SL. https://gamco.es/glosario/generalizacion/#:~:text=La%20generalizaci%C3%B3n%20se%20refiere%20a,su%20conjunto%20de%20entrenamiento%20original.