

# Clasificación de Hongos Venenosos y Comestibles Mediante Aprendizaje de Máquina

Arturo Cristián Díaz López  
Instituto Tecnológico y de Estudios Superiores de Monterrey

06-Sep-2024

## Resumen

*Este estudio presenta la implementación de modelos de regresión logística y random forest para la clasificación de hongos como comestibles o venenosos. Utilizando un conjunto de datos que describe diversas características físicas y químicas de los hongos, se entrenaron ambos modelos para predecir la clase de un hongo basado en sus atributos. Se analizaron las tasas de precisión de los modelos tanto en los conjuntos de entrenamiento como de prueba, y se realizaron visualizaciones para evaluar la distribución de las probabilidades predichas y la efectividad de los algoritmos en la clasificación binaria. Los resultados obtenidos muestran que ambos modelos son capaces de realizar una clasificación precisa, lo que podría ser útil en aplicaciones de seguridad alimentaria y micología.*

## 1 Introducción

Los hongos son organismos de gran diversidad, con especies que varían ampliamente en sus propiedades nutricionales y toxicológicas. Mientras que algunas especies de hongos son comestibles y forman parte esencial de la dieta en muchas culturas alrededor del mundo, otras pueden ser altamente tóxicas e incluso letales si se consumen. La identificación precisa de hongos venenosos es, por lo tanto, de vital importancia para prevenir intoxicaciones alimentarias. Sin embargo, la distinción entre especies comestibles y venenosas puede ser desafiante, ya que muchas comparten características morfológicas similares.

Tradicionalmente, la clasificación de hongos ha dependido de la experiencia de micólogos, quienes utilizan claves de identificación basadas en la morfología y otras características visibles. No obstante, con el avance de la ciencia de datos y el aprendizaje automático, se han desarrollado nuevas herramientas que permiten automatizar y mejorar la precisión en la identificación de hongos. En este contexto, algoritmos como la regresión logística y el random forest pueden servir como excelentes herramientas para problemas de clasificación.

El presente trabajo tiene como objetivo implementar modelos de regresión logística y random forest para la clasificación de hongos utilizando un conjunto de datos que contiene múltiples características físicas y químicas de los hongos. Toda la solución ha sido construida utilizando *Python*, un lenguaje de programación ampliamente utilizado en el rubro.

Además, se analizará el rendimiento de los modelos mediante métricas comunes como la precisión y se visualizarán las probabilidades predichas para comprender mejor el comportamiento de los algoritmos. Esta investigación tiene como objetivo verificar si estos enfoques son técnicas viables para la clasificación de hongos, contribuyendo así como herramientas útiles para profesionales en micología y seguridad alimentaria.

## 2 ETL

### 2.1 ¿Qué es ETL?

En el ámbito de la ciencia de datos y la ingeniería de datos, ETL es un acrónimo que se refiere a los procesos de Extracción, Transformación y Carga de datos (Extract, Transform, Load en inglés). Este proceso es esencial para preparar y organizar los datos que serán utilizados en análisis posteriores o en modelos de aprendizaje de máquina. ETL juega un papel crucial en garantizar que los datos sean consumibles, precisos y estén libres de inconsistencias o sesgos antes de ser utilizados en cualquier análisis.

### 2.2 Fuente de Datos

Para este estudio, se utilizó un dataset ampliamente conocido y utilizado en la comunidad de ciencia de datos: el *Mushroom Dataset* (Dataset de Hongos), disponible en la UCI Machine Learning Repository.

Este conjunto de datos contiene información detallada sobre distintas características de hongos, las cuales pueden ser utilizadas para clasificarlos como comestibles o venenosos. El dataset consta de 8,124 registros, donde cada registro representa un hongo único, y 22 atributos categóricos que describen características como el color del sombrero, el tamaño del tallo, la forma de las branquias, entre otros. Cada registro también está etiquetado con una clase binaria que indica si el hongo es comestible ('e' para *edible*) o venenoso ('p' para *poisonous*). La información detallada sobre los features o categorías se presenta en el siguiente cuadro.

Feature	Descripción
cap-shape	Forma del sombrero
cap-surface	Superficie del sombrero
cap-color	Color del sombrero
bruises?	Presencia de magulladuras
odor	Olor
gill-attachment	Unión de las láminas
gill-spacing	Espaciado de las láminas
gill-size	Tamaño de las láminas
gill-color	Color de las láminas
stalk-shape	Forma del tallo
stalk-root	Tipo de raíz del tallo
stalk-surface-above-ring	Superficie del tallo por encima del anillo
stalk-surface-below-ring	Superficie del tallo por debajo del anillo
stalk-color-above-ring	Color del tallo por encima del anillo
stalk-color-below-ring	Color del tallo por debajo del anillo
veil-type	Tipo de velo
veil-color	Color del velo
ring-number	Número de anillos
ring-type	Tipo de anillo
spore-print-color	Color de la impresión de esporas
population	Población
habitat	Hábitat

**Cuadro 1:** Descripción de los features del Mushroom Dataset

El dataset es accesible al público a través de la siguiente liga: <https://archive.ics.uci.edu/dataset/73>.

## 2.3 Proceso

Se comenzó extrayendo el dataset directamente del UCI Machine Learning Repository mediante la liga anteriormente proporcionada. A continuación, se empleó la librería *pandas* para cargar los datos en un *DataFrame*.

Dada la naturaleza categórica del dataset y con el objetivo de avanzar hacia la etapa de transformación, se realizó un mapeo de las características para convertir las variables categóricas en valores numéricos. Este mapeo facilita el análisis y procesamiento de los datos en las etapas siguientes. La distribución de categorías se puede analizar en el siguiente cuadro.

Feature	Valores Posibles
cap-shape	bell, conical, convex, flat, knobbed, sunken
cap-surface	fibrous, grooves, scaly, smooth
cap-color	brown, buff, cinnamon, gray, green, pink, purple, red, white, yellow
bruises?	bruises, no
odor	almond, anise, creosote, fishy, foul, musty, none, pungent, spicy
gill-attachment	attached, free
gill-spacing	close, crowded, distant
gill-size	broad, narrow
gill-color	black, brown, buff, chocolate, gray, green, orange, pink, purple, red, white, yellow
stalk-shape	enlarging, tapering
stalk-root	bulbous, club, cup, equal, rhizomorph, root
stalk-surface-above-ring	fibrous, scaly, silky, smooth
stalk-surface-below-ring	fibrous, scaly, silky, smooth
stalk-color-above-ring	brown, buff, cinnamon, gray, orange, pink, red, white, yellow
stalk-color-below-ring	brown, buff, cinnamon, gray, orange, pink, red, white, yellow
veil-type	partial, universal
veil-color	brown, orange, white, yellow
ring-number	none, one, two
ring-type	cobwebby, evanescent, flaring, large, none, pendant, sheathing, zone
spore-print-color	black, brown, buff, chocolate, green, orange, pink, purple, red, white, yellow
population	abundant, clustered, numerous, scattered, several, solitary
habitat	grasses, leaves, meadows, paths, urban, waste, woods

**Cuadro 2:** Valores posibles para cada feature del Mushroom Dataset

Para llevar a cabo el proceso de mapeo se definió

un diccionario que asigna un valor numérico a cada categoría. Es decir, cada categoría dentro de una característica se transformó en un número entero, comenzando desde 0 y aumentando secuencialmente. Este proceso asegura que cada categoría única se represente de manera uniforme.

Finalmente, según los autores del dataset, existen 2480 registros en el dataset sin valor para el feature 11 (*stalk-root*), lo cual representa aproximadamente un 30.5% de instancias del dataset. Con tal de abordar esta problemática, se consideraron múltiples opciones:

1. Imputación por moda.
2. Imputación condicional de moda por clase.
3. Eliminar instancias con valor faltante.
4. Eliminar la columna *stalk-root*.
5. Tratar el valor faltante como una categoría en sí.

Tras evaluar las alternativas, se tomó la decisión que la imputación condicional de moda por clase era la mejor opción. Esta solución toma en cuenta la distribución del feature *stalk-root* dentro de cada clase, lo que minimiza el sesgo que podría introducir la imputación simple con la moda global.

El proceso ETL concluye con una simple carga de los datos transformados en un nuevo archivo CSV. Este archivo, que ahora contiene solo valores numéricos, está listo para ser utilizado en las etapas siguientes del análisis y modelado. La carga final de los datos en el archivo permite la independencia del conjunto de datos transformado y su integración eficiente en los módulos posteriores del proyecto.

### 3 Regresión Logística

Para abordar el problema de clasificación de hongos como venenosos o comestibles, se implementó un modelo de regresión logística. Este modelo es una técnica de aprendizaje automático ampliamente utilizada para problemas de clasificación binaria, donde el objetivo es predecir una de dos posibles clases. En este caso, la regresión logística ayuda a distinguir entre hongos venenosos y comestibles basándose en las características del dataset.

La implementación del modelo de regresión logística se lleva a cabo a través de múltiples funciones que realizan cálculos matemáticos, cada una con un papel específico en el proceso de entrenamiento y predicción. A continuación, se presenta una breve descripción de cada una de ellas.

#### Función Sigmoide

La función sigmoide se encarga de transformar el resultado lineal de la combinación de características y pesos en una probabilidad que varía entre 0 y 1. La fórmula para la función sigmoide es:

$$h = \frac{1}{1 + e^{-z}}$$

Donde  $z$  es la combinación lineal de características y pesos. La función sigmoide garantiza que el modelo proporcione probabilidades interpretables para la clasificación binaria.

#### Función de Costo

Se encarga de calcular el costo o la pérdida, de la regresión logística. Este costo mide qué tan bien el modelo se ajusta a los datos de entrenamiento y se basa en la entropía cruzada binaria, siendo la fórmula la siguiente:

$$\text{Costo} = -\frac{1}{m} \sum_{i=1}^m [y_i \log(h_i) + (1 - y_i) \log(1 - h_i)]$$

Donde  $h_i$  es la predicción para la  $i$ -ésima muestra y  $y_i$  es la etiqueta verdadera. Un costo bajo indica que el modelo está haciendo buenas predicciones.

#### Descenso de Gradiente

Esta función implementa el algoritmo de descenso de gradiente para optimizar los pesos del modelo. Este proceso ajusta los pesos iterativamente para minimizar el costo calculado. La actualización de los pesos se realiza utilizando la fórmula:

$$\theta - = \theta - \alpha \times \frac{1}{m} \sum_{i=1}^m (h_i - y_i) X_i$$

Donde:

- $\theta$  representa el vector de pesos del modelo.
- $\alpha$  es la tasa de aprendizaje.
- $m$  es el número de muestras en el conjunto de datos.
- $h_i$  es la predicción para la  $i$ -ésima muestra.
- $y_i$  es la etiqueta verdadera para la  $i$ -ésima muestra.
- $X_i$  es el vector de características para la  $i$ -ésima muestra.

Además, la función de descenso de gradiente se ha mejorado para incluir monitoreo del conjunto de validación y un mecanismo de detención temprana en caso de sobreajuste. El algoritmo ahora monitorea el costo en el conjunto de validación y detiene el entrenamiento si no se observa una mejora durante un número especificado de épocas. Esta modificación ayuda a prevenir el sobreajuste y asegura que el modelo generalice mejor en datos no vistos. La función devuelve los pesos optimizados y las listas de costos de entrenamiento y validación durante el proceso de entrenamiento.

### Función de División del Dataset

Esta función se encarga de dividir el dataset en tres conjuntos: entrenamiento, validación y prueba. La inclusión de un conjunto de validación permite evaluar el rendimiento del modelo durante el proceso de ajuste y evitar problemas de sobreajuste antes de realizar la evaluación final en el conjunto de prueba. La función utiliza proporciones especificadas tanto para el conjunto de prueba como para el conjunto de validación, junto con un valor opcional para la semilla del generador de números aleatorios, garantizando la reproducibilidad de la división.

### Función de Predicción

Finalmente, se creó una función que realiza predicciones utilizando el modelo de regresión logística entrenado. Transforma las características de entrada en probabilidades a través de la función sigmoide y clasifica las muestras como pertenecientes a la clase positiva si la probabilidad es mayor o igual a 0.5.

## 3.1 Resultados

En esta sección se presentan los resultados obtenidos al aplicar el modelo de regresión logística sobre el dataset de hongos. Los parámetros e hiperparámetros utilizados se detallan en la siguiente lista:

- $\alpha = 0.01$
- Epochs = 10000
- Test Size = 20 %
- Validation Size = 20 %
- Random State = 42
- Patience = 15

### Proceso

Primeramente, se cargaron los datos transformados en un *DataFrame* de *pandas*, donde posteriormente se separaron en dos matrices, una para las características y otra para las etiquetas. Una vez que los datos fueron cargados, se separaron en conjuntos de entrenamiento, prueba y validación, utilizando la función definida en el modelo, con una proporción de 20 % del dataset para pruebas, 10 % para validación y el resto para entrenamiento. Para incluir el término de *bias* en el modelo, se añadió una columna de unos a las matrices de características para ambos conjuntos de datos (pruebas y entrenamiento). De esta manera, este término se maneja como un peso adicional dentro del modelo.

Los pesos del modelo fueron inicializados en 0 y posteriormente se les aplicó el algoritmo de gradiente descendiente para optimizarlos y reducir el costo. Esto se ejecutó durante el número total de épocas, ajustando los pesos en cada iteración.

### Predicciones

Tras optimizar los pesos, se realizaron predicciones en los tres conjuntos dados (train, test y validation). La medida inicial de éxito del modelo fue la *precisión*, obteniendo los siguientes resultados:

- Precisión en el conjunto de test : 92.68 %
- Precisión en el conjunto de train : 93.17 %
- Precisión en el conjunto de validation : 93.54 %

Otro valor importante, es el *bias*, el cual representa el ajuste adicional necesario para mejorar la capacidad del modelo de hacer predicciones precisas. El valor final de este componente fue de  $-0.0736$ . El que sea negativo indica que la clase de valor 0 (edible), se suele predecir con más frecuencia que la clase 1 (poisonous). A su vez, el obtener un valor bajo es un indicador de que el modelo es capaz de ajustar sus predicciones para reflejar adecuadamente la distribución de las clases en el dataset.

### Interpretaciones

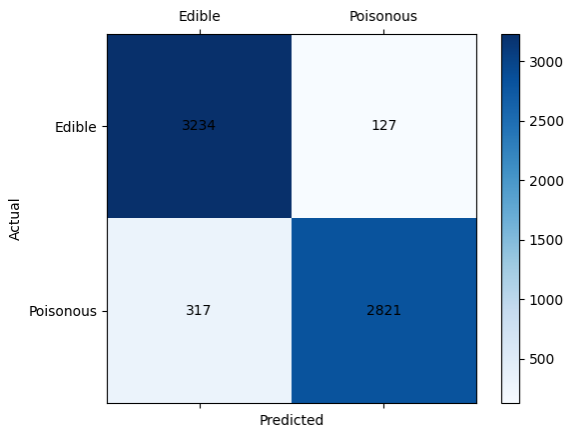
Aunque los valores anteriores proporcionan una indicación general del desempeño del modelo, no son suficientes para saber con certeza su precisión. Para interpretar adecuadamente los resultados generados por el modelo, es recomendando utilizar herramientas

visuales como las gráficas. La implementación de librerías como *matplotlib* o *seaborn* permite la creación de gráficos que facilitan una comprensión profunda y detallada de los resultados obtenidos. Las gráficas permiten observar patrones, distribuciones y posibles áreas de mejora que podrían no ser evidentes solo a partir de los valores numéricos.

### Matriz de Confusión

La matriz de confusión es una herramienta fundamental en la evaluación de modelos de clasificación, ya que permite visualizar el desempeño del modelo al comparar las predicciones realizadas con las etiquetas reales. En una matriz de confusión, las filas representan las etiquetas reales, mientras que las columnas muestran las predicciones realizadas por el modelo.

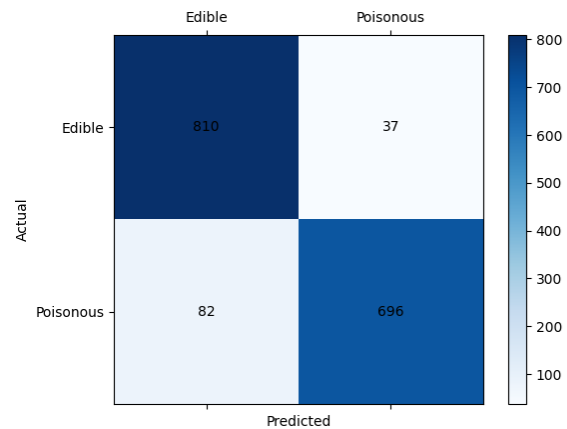
Cada celda de la matriz indica la cantidad de instancias clasificadas en una categoría específica, proporcionando una visión clara de cuántas predicciones fueron correctas y cuántas fueron erróneas. En este caso, el modelo fue entrenado para clasificar hongos como comestibles (*edible*) o venenosos (*poisonous*). Las diagonales de la matriz reflejan las predicciones correctas, mientras que los valores fuera de la diagonal representan los errores de clasificación.



**Figura 1:** Matriz de confusión del conjunto de entrenamiento del modelo de Regresión Logística.

Esta matriz nos muestra que 3234 hongos fueron predichos correctamente como comestibles, mientras que 317 se predijeron erróneamente como venenosos. A su vez, 2821 hongos se predijeron correctamente como venenosos, y 127 se predijeron erróneamente como comestibles. Podemos observar que, el modelo cuenta

con una precisión más alta de predicción para los hongos de clase venenosa, con un valor de 95.69 % para este conjunto de datos. La precisión es menor para la clase comestible, sin embargo, se considera más importante predecir correctamente los hongos venenosos con el objetivo de evitar posibles envenenamientos ante especies de hongos desconocidas.

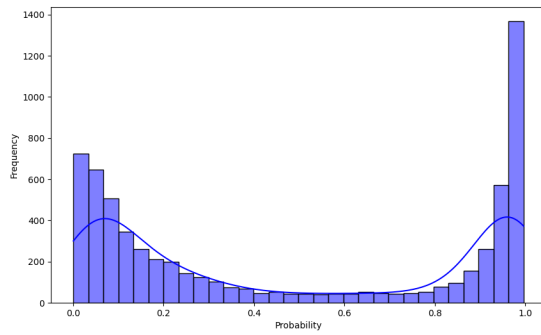


**Figura 2:** Matriz de confusión del conjunto de prueba del modelo de Regresión Logística.

La segunda matriz de confusión generada muestra una distribución similar a la primera, siendo las predicciones de hongos venenosos más precisas que las de hongos comestibles.

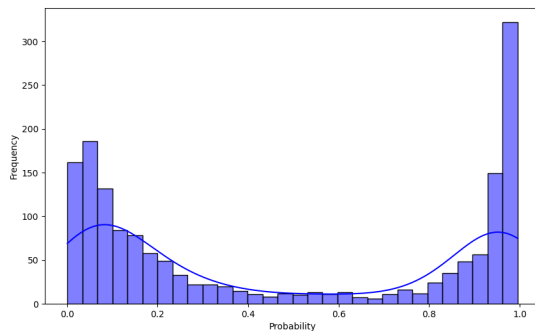
### Distribución de Probabilidades Predichas

Esta gráfica permite visualizar la confianza del modelo en sus predicciones al mostrar cómo se distribuyen las probabilidades asignadas a cada clase. En el eje X, se encuentran las probabilidades asignadas por el modelo a cada muestra del conjunto de datos, mientras que en el eje Y se muestra la frecuencia con la que el modelo asigna esas probabilidades. Según el contexto presentado, las probabilidades cercanas a 0 indican una mayor confianza en que el hongo es comestible, y las cercanas a 1, una mayor confianza en que es venenoso.



**Figura 3:** Distribución de probabilidades predichas del conjunto de entrenamiento del modelo de Regresión Logística.

Podemos observar que los valores tienden a agruparse cerca de los extremos, lo que demuestra que el modelo tiene una alta certeza en la mayoría de sus predicciones. El contar con pocos valores en el medio es un indicador de que existen pocos casos en los que el modelo no tiene certeza sobre la clase a predecir en una instancia.

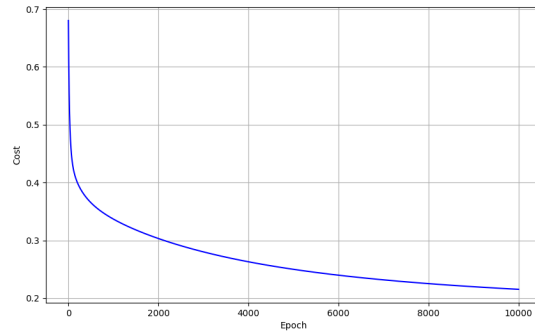


**Figura 4:** Distribución de probabilidades predichas del conjunto de prueba del modelo de Regresión Logística.

Para el conjunto de prueba podemos observar la misma tendencia, agrupaciones en los extremos y mínimos en el medio.

### Evolución del costo durante el entrenamiento

Finalmente, para lograr comprender cómo es que el modelo avanza en cada iteración, podemos analizar la evolución del costo durante el entrenamiento. Esta gráfica muestra cómo el costo varía a lo largo de las épocas durante el entrenamiento del modelo. El costo calculado en cada época, refleja la diferencia entre las predicciones del modelo y los valores verdaderos.



**Figura 5:** Evolución del costo durante el entrenamiento del modelo de Regresión Logística.

En esta gráfica, el eje X representa el número de épocas, mientras que el eje Y muestra el valor del costo. Una disminución constante del costo a medida que avanzan las épocas indica que el modelo está aprendiendo y ajustando correctamente sus pesos para minimizar el error en las predicciones. Esta tendencia es un buen signo de que el proceso de entrenamiento está siendo efectivo.

La gráfica revela una reducción significativa en el costo a lo largo del entrenamiento, lo que indica que el modelo está optimizando sus parámetros adecuadamente.

## 3.2 Diagnóstico

El análisis del desempeño del modelo de regresión logística revela que el modelo está **bien ajustado**, evitando tanto el sobreajuste (overfitting) como el subajuste (underfitting). Los resultados obtenidos muestran precisiones muy similares en los conjuntos de entrenamiento, validación y prueba, lo cual indica que el modelo generaliza bien a datos no vistos durante el entrenamiento. Específicamente, la precisión en el conjunto de prueba es del 92.62 %, en el conjunto de entrenamiento es del 93.18 %, y en el conjunto de validación es del 93.54 %. Esta consistencia en las métricas de precisión sugiere que el modelo no está ajustado en exceso a las peculiaridades del conjunto de entrenamiento, sino que mantiene un rendimiento estable a través de diferentes subconjuntos de datos.

Asimismo, los costos calculados en los diferentes conjuntos también presentan una tendencia similar, con valores de 0.2170 en el conjunto de prueba, 0.2152 en el conjunto de entrenamiento y 0.2140 en el conjunto de validación. La cercanía entre estos valores de costo refuerza la idea de que el modelo está bien ajustado, ya que no hay una gran discrepancia entre el costo en

los datos de entrenamiento y los datos de prueba o validación.

Además, el término de bias calculado es  $-0.0756$ , que indica una leve tendencia hacia una predicción sistemáticamente menor. Sin embargo, este valor es relativamente pequeño, sugiriendo que el modelo tiene un bajo sesgo y está capturando la relación entre las características de los hongos y su clasificación de manera efectiva.

Es importante mencionar que durante el entrenamiento del modelo, no se levantó la excepción de sobreajuste (excepción contenida en la función de descenso de gradiente), lo que indica que el modelo se ha entrenado con efectividad durante el número total de épocas. Esto asegura que el modelo no solo se ha ajustado adecuadamente a los datos de entrenamiento, sino que también ha mantenido su capacidad para generalizar a nuevos datos.

### 3.3 Tuning

Con la finalidad de presentar una mejora en el desempeño de este modelo, se realizó un proceso de búsqueda iterativa para identificar los hiperparámetros que mejor ajustan al mismo. Siendo los hiperparámetros a ajustar, el número de épocas y la tasa de aprendizaje. Se probaron los siguientes valores para las anteriores variables:

- Tasas de Aprendizaje (Learning Rates): 0.0001, 0.001, 0.01, 0.1, 0.003, 0.03, 0.005, 0.05
- Número de Épocas (Epochs): 10,000, 20,000, 30,000, 50,000

Tras correr la evaluación, se concluyó que los mejores hiperparámetros para el modelo eran los siguientes:

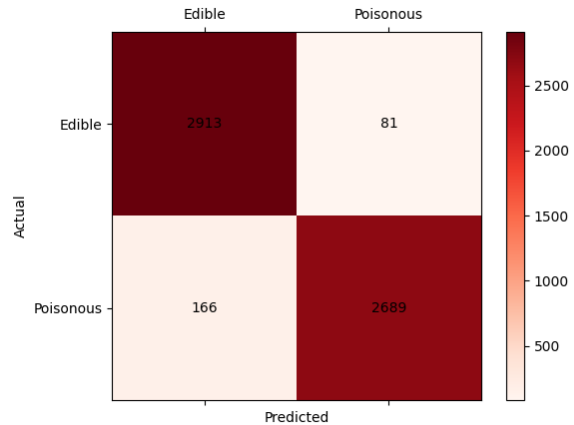
- Tasa de Aprendizaje (Learning Rate): 0.05
- Número de Épocas (Epochs): 50,000

Con estos hiperparámetros, el modelo alcanzó una precisión de **96.06 %**, **95.78 %** y **95.23 %** en los conjuntos de prueba, entrenamiento y validación respectivamente. El ajuste de hiperparámetros ha permitido mejorar significativamente el rendimiento del modelo, alcanzando una precisión cercana al 96 % en el conjunto de prueba. Estos resultados reflejan una buena capacidad de generalización del modelo y una mejora en la optimización del costo.

#### Interpretación

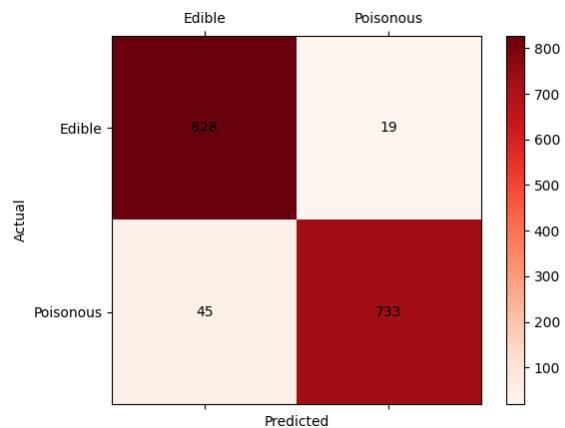
Finalmente, podemos apreciar los resultados de este tuning utilizando las gráficas presentadas

anteriormente, con la información del nuevo modelo entrenado.



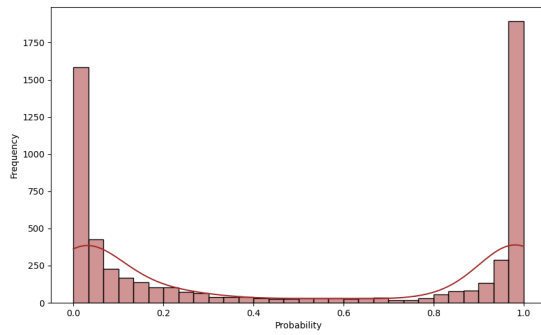
**Figura 6:** Matriz de confusión del conjunto de entrenamiento del modelo de Regresión Logística con Tuning.

Al compararla con la matriz del modelo sin Tuning, podemos apreciar que se han recortado en casi un 50 % las predicciones incorrectas de clases comestibles y venenosas, lo cual representa una mejora significativa.



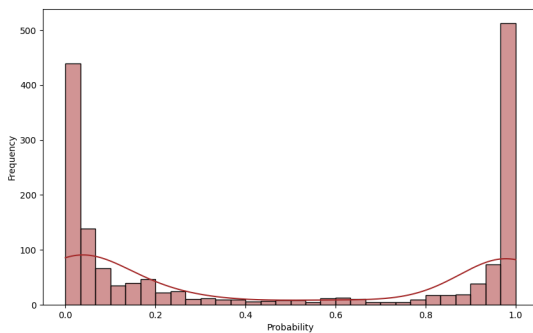
**Figura 7:** Matriz de confusión del conjunto de prueba del modelo de Regresión Logística con Tuning.

De la misma manera, podemos observar resultados similares en el conjunto de prueba.



**Figura 8:** Distribución de probabilidades predichas del conjunto de entrenamiento del modelo de Regresión Logística con Tuning.

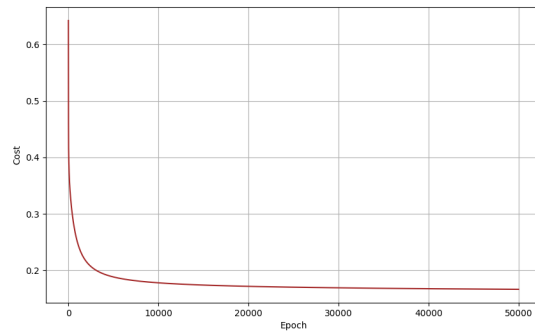
La gráfica de probabilidades predichas muestra que el modelo cuenta con una mucha mayor seguridad al momento de realizar una predicción, es decir, la cantidad de predicciones cercanas a 0.5 se redujo considerablemente.



**Figura 9:** Distribución de probabilidades predichas del conjunto de prueba del modelo de Regresión Logística con Tuning.

Podemos observar la misma tendencia en la gráfica del conjunto de prueba.

Finalmente, las gráficas del costo nos muestran el ritmo en el que el modelo aprendió con estos nuevos hiperparámetros.



**Figura 10:** Evolución del costo durante el entrenamiento del modelo de Regresión Logística con Tuning.

Aunque los resultados de desempeño son buenos, podemos apreciar que alrededor de la época 20000, el costo comienza a estabilizarse, lo que indica que el modelo ha alcanzado una fase de aprendizaje más lento o ha llegado a un punto donde las mejoras son muy pequeñas. Este comportamiento es típico cuando el modelo ya ha encontrado un buen conjunto de pesos y está realizando ajustes finos.

## 4 Random Forest

El algoritmo de Random Forest es un potente método de aprendizaje automático utilizado para clasificación y regresión. Este modelo se basa en la idea de construir un "bosque" de árboles de decisión y combinar sus predicciones para mejorar la precisión y robustez del modelo global. La técnica consiste en crear múltiples árboles de decisión durante el entrenamiento y hacer que cada árbol vote para la predicción final, utilizando la mayoría de votos en el caso de clasificación o la media de las predicciones en el caso de regresión.

Cada árbol en el bosque se construye a partir de una muestra aleatoria del conjunto de datos de entrenamiento, con reemplazo, y se seleccionan aleatoriamente un subconjunto de características en cada nodo para la toma de decisiones. Este enfoque de "bagging" (bootstrap aggregating) y la aleatorización de características ayudan a reducir la varianza y evitar el sobreajuste, ofreciendo así un modelo que es menos sensible a la variabilidad en los datos y más generalizable a nuevos conjuntos de datos.

El modelo de Random Forest tiene varias ventajas, incluyendo la capacidad de manejar grandes cantidades de datos con alta dimensionalidad, y su habilidad para proporcionar una estimación de la importancia de las características, lo cual es útil para la interpretación



del modelo. En esta investigación, hemos implementado un modelo de Random Forest para la clasificación de hongos, con el objetivo de evaluar su rendimiento y compararlo con el modelo de regresión logística previamente desarrollado.

#### 4.1 Entrenamiento

Los parámetros utilizados para el entrenamiento del Random Forest fueron los siguientes:

- Test Size = 20 %
- Validation Size = 10 %
- Random State = 42
- Patience = 15

La función `train_random_forest` se encarga de entrenar un modelo de Random Forest para la clasificación. Este proceso se lleva a cabo utilizando la clase `RandomForestClassifier` del paquete `sklearn`. La función recibe como entrada las variables predictoras (`X_train`) y la variable objetivo (`y_train`) del conjunto de entrenamiento.

Se pueden ajustar dos parámetros principales en esta función: `random_state` y `n_estimators`. El parámetro `random_state` se utiliza para asegurar la reproducibilidad del modelo, permitiendo que los resultados sean consistentes en cada ejecución. El parámetro `n_estimators` determina el número de árboles en el bosque.

Durante el entrenamiento, el modelo `RandomForestClassifier` construye múltiples árboles de decisión a partir de diferentes subconjuntos del conjunto de datos de entrenamiento. Cada árbol se ajusta a una muestra aleatoria de los datos y realiza predicciones basadas en la mayoría de votos o la media de las predicciones en el caso de regresión. Una vez entrenado, el modelo puede ser utilizado para hacer predicciones en nuevos datos y evaluar su efectividad en la clasificación de hongos.

La función retorna el modelo de Random Forest entrenado, el cual puede ser utilizado en etapas posteriores para realizar predicciones y analizar su efectividad en la clasificación de hongos.

#### 4.2 Resultados

El modelo de Random Forest ha demostrado un rendimiento excepcional en la clasificación de hongos. A continuación, se presentan las métricas de precisión obtenidas en los diferentes conjuntos de datos utilizados en el estudio:

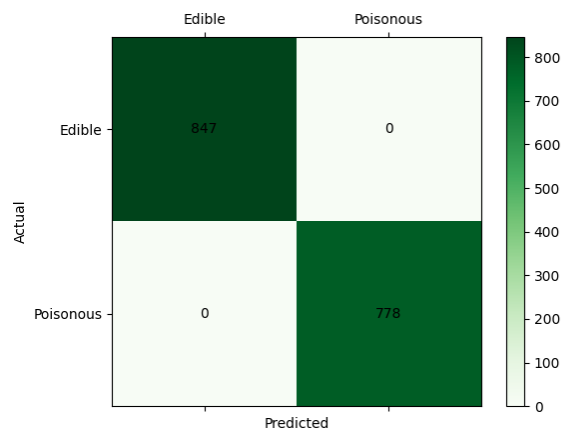
- Precisión en el conjunto de test: 100.00 %
- Precisión en el conjunto de train: 100.00 %
- Precisión en el conjunto de validation: 100.00 %

Estos resultados indican que el modelo de Random Forest ha logrado una clasificación perfecta en todos los conjuntos de datos, lo cual sugiere una alta capacidad de generalización y una ausencia de sobreajuste. La consistencia en las tasas de precisión entre los conjuntos de entrenamiento, validación y prueba sugiere que el modelo está bien ajustado y no presenta problemas de sobreajuste, ya que ha mantenido un rendimiento perfecto en cada fase del proceso de evaluación.

Es importante señalar que estos resultados ideales podrían ser indicativos de un dataset que se ajusta excepcionalmente bien al modelo, o bien de una posible falta de variabilidad en los datos. A pesar de estos resultados sobresalientes, es recomendable seguir evaluando el modelo con diferentes métricas y conjuntos de datos para garantizar la robustez y la aplicabilidad del modelo en escenarios del mundo real.

#### 4.3 Interpretaciones

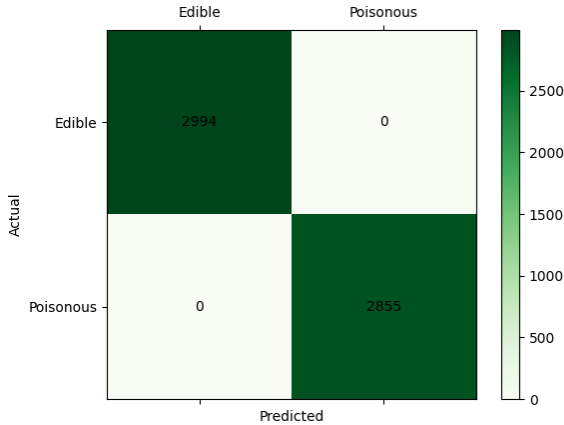
En esta subsección se presentan las gráficas que ilustran el desempeño del modelo de Random Forest en la clasificación de hongos. A continuación se muestra la matriz de confusión obtenida para el conjunto de prueba, la cual proporciona una visualización detallada de las predicciones correctas e incorrectas realizadas por el modelo.



**Figura 11:** Matriz de confusión del conjunto de prueba del modelo Random Forest.

La matriz de confusión permite observar cómo el modelo ha clasificado cada una de las categorías, mostrando el número de verdaderos positivos, verdaderos

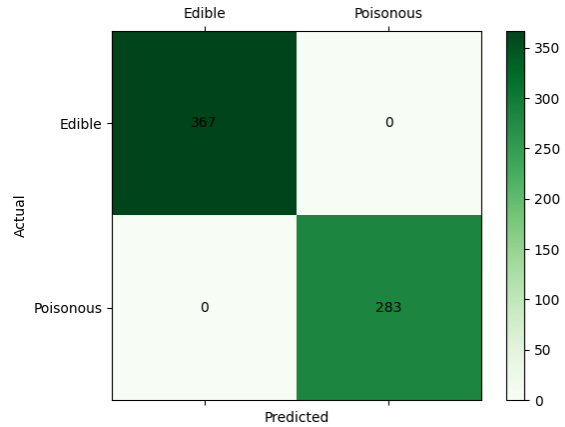
negativos, falsos positivos y falsos negativos. Dado que el modelo ha mostrado una precisión del 100.00 % en el conjunto de prueba, es de esperar que la matriz de confusión refleje un desempeño perfecto, sin errores de clasificación.



**Figura 12:** Matriz de confusión del conjunto de entrenamiento del modelo Random Forest.

La matriz de confusión del conjunto de entrenamiento también muestra un desempeño perfecto con una precisión del 100.00 %. Este resultado sugiere que el modelo ha logrado clasificar correctamente todos los ejemplos en el conjunto de entrenamiento, lo cual es esperado dado el alto rendimiento del modelo en el conjunto de prueba.

Dado que tanto en el conjunto de entrenamiento como en el de prueba se observa una precisión del 100.00 %, podemos inferir que el modelo no está sobreajustado (overfitting). Sin embargo, es importante corroborar estos resultados con el conjunto de validación para asegurarse de que el modelo generaliza bien a datos no vistos. La consistencia en la precisión entre todos los conjuntos sugiere que el modelo de Random Forest ha aprendido bien las características relevantes para la clasificación y es capaz de generalizar eficazmente a nuevas instancias.



**Figura 13:** Matriz de confusión del conjunto de validación del modelo Random Forest.

La matriz de confusión del conjunto de validación, con una precisión del 100.00 %, confirma que el modelo también ha realizado clasificaciones correctas en el conjunto de validación. Este resultado es consistente con las precisiones observadas en los conjuntos de entrenamiento y prueba, indicando que el modelo está generalizando bien y no presenta sobreajuste.

## 5 Conclusiones

Es factible concluir que el desempeño del modelo fue mejor que el esperado, dado que no se utilizó ningún framework, y todas las funciones de aprendizaje de máquina fueron implementadas desde cero. Como pasos siguientes, considero que diagnosticar el modelo a profundidad sería una buena dirección, para definir pasos siguientes. En caso de overfitting, se podría aplicar alguna técnica de regularización al modelo, y en caso de underfitting, consideraría realizar un algoritmo de red neuronal que permita una configuración más profunda de parámetros.

## 6 Bibliografía

- [1] UCI Machine Learning Repository. *Mushroom Dataset*.  
<https://archive.ics.uci.edu/dataset/73/mushroom>
- [2] Wikipedia. *Extract, transform, load*.  
[https://en.wikipedia.org/wiki/Extract,\\_transform,\\_load](https://en.wikipedia.org/wiki/Extract,_transform,_load)

- [3] Wikipedia. *Logistic regression*.  
[https://en.wikipedia.org/wiki/Logistic\\_regression](https://en.wikipedia.org/wiki/Logistic_regression)
- [4] Beyond PhD Coaching. *Binary Logistic Regression*.  
<https://www.beyondphdcoaching.com/dissertation/binary-logistic-regression/>
- [5] Wikipedia. *Sigmoid function*.  
[https://en.wikipedia.org/wiki/Sigmoid\\_function](https://en.wikipedia.org/wiki/Sigmoid_function)
- [6] Wikipedia. *Cross-entropy*.  
<https://en.wikipedia.org/wiki/Cross-entropy>
- [7] Machine Learning Mastery. *Cross-entropy for Machine Learning*.  
<https://machinelearningmastery.com/cross-entropy-for-machine-learning/>
- [8] Towards Data Science. *Understanding Binary Cross-Entropy (Log Loss) - A Visual Explanation*.  
<https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a>
- [9] Analytics Vidhya. *Binary Cross-Entropy (aka Log Loss) – The Cost Function used in Logistic Regression*.  
<https://www.analyticsvidhya.com/blog/2020/11/binary-cross-entropy-aka-log-loss-the-cost-function-used-in-logistic-regression/>
- [10] Wikipedia. *Gradient descent*.  
[https://en.wikipedia.org/wiki/Gradient\\_descent](https://en.wikipedia.org/wiki/Gradient_descent)
- [11] GeeksforGeeks. *How to Implement a Gradient Descent in Python to Find a Local Minimum*.  
<https://www.geeksforgeeks.org/how-to-implement-a-gradient-descent-in-python-to-find-a-local-minimum/>
- [12] Induraj. *Implementing Gradient Descent in Python*.  
<https://induraj2020.medium.com/implementing-gradient-descent-in-python-d1c6aeb9a448>
- [13] Machine Learning Mastery. *How to Choose the Right Test Options When Evaluating Machine Learning Algorithms*.  
<https://machinelearningmastery.com/how-to-choose-the-right-test-options-when-evaluating-machine-learning-algorithms/>
- [14] V7 Labs. *Train, Validation, and Test Set*.  
<https://www.v7labs.com/blog/train-validation-test-set>
- [15] Wikipedia. *Confusion matrix*.  
[https://en.wikipedia.org/wiki/Confusion\\_matrix](https://en.wikipedia.org/wiki/Confusion_matrix)
- [16] Towards Data Science. *Understanding Confusion Matrix*.  
<https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>
- [17] Wikipedia. *Random Forest*.  
[https://en.wikipedia.org/wiki/Random\\_forest](https://en.wikipedia.org/wiki/Random_forest)
- [18] Scikit-learn. *RandomForestClassifier*.  
<https://scikit-learn.org/stable/modules/generated/sklearn.en>
- [19] IBM. *Random Forest: Overview and Use Cases*.  
<https://www.ibm.com/topics/random-forest~:text=Random%20forest%20is%20a%20commonly,Decis>