Clasificación de Hongos Venenosos y Comestibles Mediante Regresión Logística

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

24 de agosto 2024

Resumen

Este estudio presenta la implementación de un modelo de regresión logística 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 entrenó un modelo para predecir la clase de un hongo basado en sus atributos. Se analizaron las tasas de precisión del modelo tanto en los conjuntos de entrenamiento como de prueba, y se realizaron visualizaciones para evaluar la distribución de las probabilidades predichas y la efectividad del modelo en la clasificación binaria. Los resultados obtenidos muestran que el modelo es capaz 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, la regresión logística, una técnica de modelado estadístico ampliamente utilizada para problemas de clasificación binaria, se presenta como una herramienta eficaz para diferenciar entre hongos comestibles y venenosos.

El presente trabajo tiene como objetivo implementar un modelo de regresión logística para la clasificación de hongos utilizando un conjunto de datos que contiene múltiples características de los hongos. Toda la solución ha sido construida utilizando *Python*, un lenguaje de programación ampliamente utilizado en el análisis de datos y aprendizaje automático.

Además, se analizará el rendimiento del modelo mediante métricas comunes como la precisión y se visualizarán las probabilidades predichas para comprender mejor el comportamiento del modelo. Esta investigación tiene como objetivo verificarsi la regresión logística es una técnica viable para la clasificación de hongos y así ser una herramienta útil 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 carac-

terística 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 Modelo

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.

3.1 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.

3.2 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:

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

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.

3.3 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:

- θ representa el vector de pesos del modelo.
- α 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.

3.4 Función de División del Dataset

Esta función se encarga de dividir el dataset en conjuntos de entrenamiento y prueba. Esta división es esencial para entrenar y evaluar el modelo de manera efectiva. La función utiliza una proporción especificada para el conjunto de prueba y un valor opcional para la semilla del generador de números aleatorios, garantizando la reproducibilidad de la división.

3.5 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.

4 Resultados

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

- α = 0.01
- Epochs = 10000
- Test Size = 20 %
- Random State = 42

4.1 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 y prueba, utilizando la función definida en el modelo, utilizando una proporción de 20 % del dataset para pruebas 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.

4.2 Predicciones

Tras optimizar los pesos, se realizaron predicciones tanto en el conjunto de prueba como en el de entrenamiento. La medida inicial de éxito del modelo fue la *precisión*, obteniendo los siguientes resultados:

- Precisión en el conjunto de prueba : 92.68 %
- Precisión en el conjunto de entrenamiento : 93.17 %

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.

4.3 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.

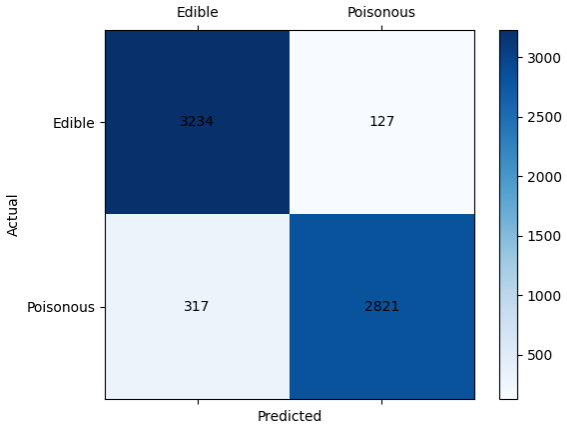

Figura 1: *Matriz de confusión del conjunto de entrenamiento.*

Esta matriz nos muestra que 3234 hongos fueron predecidos 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.

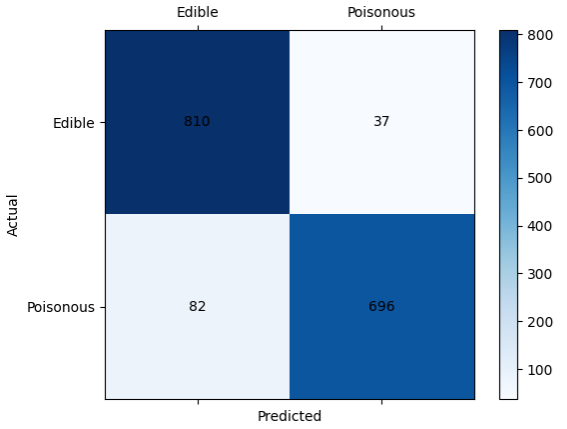

Figura 2: *Matriz de confusión del conjunto de prueba.*

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.

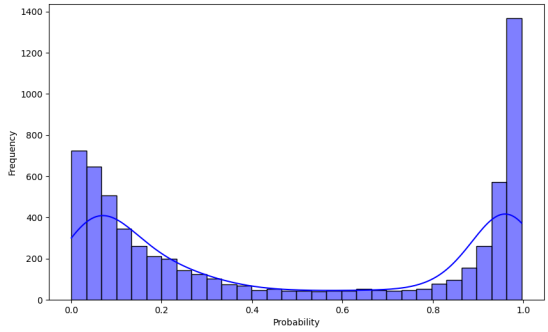

Figura 3: *Distribución de probabilidades predichas del conjunto de entrenamiento.*

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.

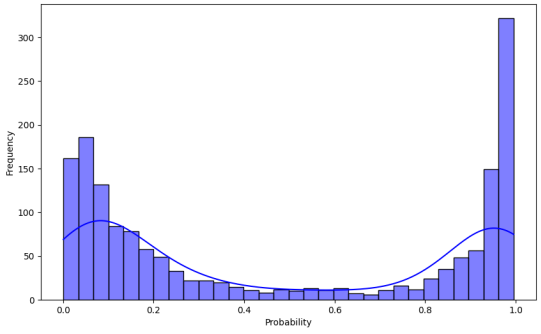

Figura 4: *Distribución de probabilidades predichas del conjunto de prueba.*

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.

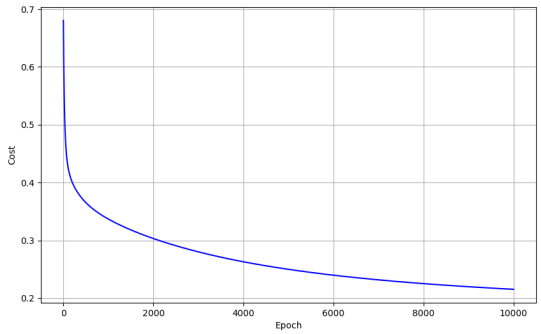

Figura 5: *Evolución del costo durante el entrenamiento.*

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.

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,$_t ransform,_l oad$

[3] Wikipedia. *Logistic regression*. https://en.wikipedia.org/wiki/Logistic$_r egression$

[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$_f unction$

[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$_d escent$

[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$_m atrix$

[16] Towards Data Science. *Understanding Confusion Matrix*. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62