Clasificación de Hongos: Comestibles vs. Venenosos

Miguel Angel Serna Montoya
Dept. de Ingeniería de Sistemas
Facultad de Ingeniería
Universidad de Antioquia
Medellín, Colombia
mangel.serna@udea.edu.co

Jose Camilo Loaiza Hoyos
Dept. de Ingeniería de Sistemas
Facultad de Ingeniería
Universidad de Antioquia
Medellín, Colombia
camilo.loaizal@udea.edu.co

***Abstract*—Wild mushroom identification is a high-stakes problem in public health: misclassifying a poisonous specimen as edible can be fatal. This report presents a comparative analysis of five machine learning models: Logistic Regression, K-Nearest Neighbors (KNN), Random Forest, Support Vector Machines (SVM), and Multi-Layer Perceptron (MLP). Using the UCI Agaricus-Lepiota dataset, we implement a hybrid encoding strategy—applying One-Hot Encoding for distance-based/linear models and Label Encoding for tree-based ensembles—to ensure mathematical validity without compromising efficiency. Our experiments reveal that while non-linear models (RF, KNN, SVM) achieve near-perfect classification (100The contribution of this work lies in the rigorous validation methodology and the critical analysis of the "recall" metric for the poisonous class, prioritizing food safety over general accuracy.**

***Impact Statement*—This report provides a comprehensive evaluation of linear and non-linear classifiers for mushroom edibility, highlighting the critical impact of categorical encoding strategies (One-Hot vs. Label) on model performance and establishing Random Forest as the most robust solution for safety-critical applications.**

***Index Terms*—Agaricus–Lepiota, Hybrid Encoding, Stratified K-Fold, Random Forest, Neural Networks, Food Safety, Recall.**

I. Introducción

La identificación de hongos silvestres es un desafío de alta criticidad para la salud pública. A diferencia de otras advertencias simples en la naturaleza, no existe una regla empírica universal y fiable para distinguir un hongo seguro de uno mortal. Errores en la clasificación entre especies comestibles y venenosas pueden tener consecuencias fatales.

El Machine Learning ofrece una solución robusta para abordar esta incertidumbre. Los modelos supervisados pueden analizar patrones complejos a partir de múltiples características observables, superando las limitaciones de la inspección humana.

Este artículo presenta un análisis comparativo de cinco algoritmos de clasificación representativos de diferentes paradigmas (lineales, basados en instancias, ensembles, kernel y redes neuronales), aplicados al conjunto de datos "Agaricus–Lepiota" de UCI [1]. El objetivo principal es desarrollar un modelo con la máxima precisión posible, poniendo un énfasis crítico en la minimización del error más peligroso: clasificar incorrectamente un hongo venenoso como comestible (Falso Negativo). Se evaluarán los modelos de **Regresión Logística, KNN, Random Forest, SVM y Red Neuronal (MLP)** para identificar las características morfológicas más determinantes y establecer la viabilidad de un sistema de clasificación automatizado y seguro.

II. Descripción del Problema

Esta sección describe el contexto del problema de clasificación de hongos, detalla la composición de la base de datos utilizada y justifica el paradigma de aprendizaje automático seleccionado para su solución.

A. Contexto y Utilidad de la Solución

1) Contexto del Problema: El conjunto de datos (dataset) Agaricus–Lepiota contiene descripciones morfológicas de 23 especies de hongos con branquias (*gilled mushrooms*). El problema fundamental a resolver es de alta criticidad para la salud pública: determinar de manera fiable si un hongo es **COMESTIBLE** (`EDIBLE`) o **VENENOSO** (`POISONOUS`) basándose únicamente en sus características físicas observables.

2) Utilidad de la Solución de Machine Learning: La identificación manual de hongos requiere un alto grado de conocimiento experto, y los errores pueden ser fatales. Una solución basada en Machine Learning (ML) ofrece una utilidad tangible en varias áreas:

1) **Seguridad Alimentaria:** Prevenir intoxicaciones mortales. Un sistema automatizado robusto puede actuar como una herramienta de verificación crucial y salvar vidas, considerando que miles de personas son hospitalizadas anualmente por esta causa.
2) **Automatización de Identificación:** Servir como base para aplicaciones móviles o sistemas de apoyo que asistan a recolectores de hongos, entusiastas de la micología y profesionales de la gastronomía.
3) **Educación y Prevención:** Al identificar las características más predictivas, el modelo puede ayudar a educar al público sobre qué rasgos son los más importantes para distinguir las especies venenosas.
4) **Apoyo a la Toma de Decisiones:** Funcionar como una "segunda opinión" automatizada para asistir a micólogos expertos, ayudando a verificar clasificaciones y a detectar patrones no obvios en combinaciones de múltiples variables.

La naturaleza de este problema es de alto riesgo, ya que un error (clasificar un hongo venenoso como comestible) es fatal. Por lo tanto, se requiere un modelo de ML que priorice la minimización de este tipo de error.

B. Composición de la Base de Datos

El análisis se basa en el dataset de Hongos (Mushroom) del repositorio de UCI, usando la versión estándar del repositorio UCI [1].

1) Estadísticas Generales: La composición general del conjunto de datos es la siguiente:

- **Total de muestras:** 8,124 registros [1].
- **Número de variables:** 22 características (features) descriptivas.
- **Variable objetivo (clase):** 1 variable binaria (`EDIBLE` o `POISONOUS`).

2) Distribución de Clases: Las clases están moderadamente balanceadas, lo cual es ideal para el entrenamiento de modelos de clasificación y evita sesgos significativos.

TABLE I
DISTRIBUCIÓN DE CLASES DEL DATASET

Clase	Cantidad	Porcentaje
EDIBLE (Comestible)	4,208	51.80%
POISONOUS (Venenoso)	3,916	48.20%

3) Datos Faltantes y Estrategia de Imputación: El análisis de calidad de datos reveló la existencia de valores faltantes, codificados en la fuente como un signo de interrogación (`'?'`).

- **Variable afectada:** `stalk-root` (tipo de raíz del tallo).
- **Total de faltantes:** 2,480 muestras, lo que representa un 30.53% de los datos en esta columna específica.
- **Estrategia de imputación:** No se utilizó una imputación estadística (como la moda) ni se eliminaron los registros, ya que esto implicaría perder casi el 30% de los datos. En su lugar, se adoptó la estrategia de tratar el valor `'?'` como una **categoría adicional** (ej. "missing" o "desconocido"). La justificación es que el algoritmo puede aprender patrones asociados a la ausencia de esta información (por ejemplo, si ciertos tipos de hongos tienen una raíz que es consistentemente difícil de observar o no aplica).

4) Significado de las Variables: Todas las 22 variables de entrada son categóricas nominales y describen características morfológicas observables del hongo. Estas se agrupan en:

- **Estructura del Sombrero (Cap):** `cap-shape`, `cap-surface`, `cap-color`.
- **Características Generales:** `bruises` (presencia de magulladuras), `odor` (olor).
- **Estructura de las Branquias (Gill):** `gill-attachment`, `gill-spacing`, `gill-size`, `gill-color`.
- **Estructura del Tallo (Stalk):** `stalk-shape`, `stalk-root`, `stalk-surface-above-ring`, `stalk-surface-below-ring`, `stalk-color-above-ring`, `stalk-color-below-ring`.
- **Velo y Anillo (Veil & Ring):** `veil-type`, `veil-color`, `ring-number`, `ring-type`.
- **Reproducción y Hábitat:** `spore-print-color` (color de esporas), `population` (tipo de agrupación), `habitat` (hábitat de crecimiento).

El análisis de la literatura y la exploración de datos identifican a `odor` y `spore-print-color` como variables críticas y altamente predictivas.

5) Tipo de Codificación de Variables: Dado que los datos originales son cadenas de texto (ej. "bell", "conical"), es necesaria una transformación numérica. Se implementaron dos flujos de datos paralelos:

Flujo One-Hot Encoding (OHE): Genera vectores binarios dispersos (aumentando la dimensionalidad de 22 a 117 variables).

- *Modelos Aplicados:* Regresión Logística, KNN, SVM, MLP.
- *Justificación:* Estos algoritmos operan en espacios geométricos o calculan sumas ponderadas. El uso de *Label Encoding* (asignar 1, 2, 3...) introduciría una relación de orden que sesgaría el aprendizaje (ej. asumir que "olor a almendra" es numéricamente cercano a "olor a anís" y lejano a "olor fétido").

Flujo Label Encoding: Asigna un entero único a cada categoría.

- *Modelos Aplicados:* Random Forest.
- *Justificación:* Los árboles de decisión realizan particiones basadas en conjuntos (ej. $X \in \{A, B\}$) y no en distancias. Esta codificación mantiene la dimensionalidad original (22 variables), mejorando la eficiencia computacional sin degradar el rendimiento del modelo.

C. Paradigma y Modelos Seleccionados

1) Paradigma Elegido: El enfoque seleccionado es **Aprendizaje Supervisado** para **Clasificación Binaria**.

Justificación:

- **Supervisado:** El dataset cuenta con etiquetas de verdad conocida (*Ground Truth*) para cada muestra.
- **Binario:** El objetivo es discriminar estrictamente entre dos clases mutuamente excluyentes: `EDIBLE` y `POISONOUS`.
- **Predictivo:** Se busca inferir la toxicidad de nuevas muestras basándose en patrones aprendidos de las características morfológicas.

2) Restricciones de los Modelos y Codificación: Una restricción crítica identificada es que los algoritmos matemáticos requieren entradas numéricas, mientras que el dataset es categórico nominal. Para abordar esto sin introducir sesgos ni ineficiencias, se diseñó una **Estrategia Híbrida de Codificación**:

Flujo A: One-Hot Encoding (OHE): Aplicado a modelos basados en distancia o geometría (*Regresión Logística, KNN, SVM, MLP*). *Justificación:* Estos modelos interpretan los valores numéricos como magnitudes. Usar una codificación simple (0, 1, 2...) introduciría una relación de orden falsa (ej. asumir que "Color=2" es mayor que "Color=1"), sesgando el aprendizaje. El OHE evita esto expandiendo las variables.

Flujo B: Label Encoding: Aplicado exclusivamente a *Random Forest*. *Justificación:* Los árboles de decisión realizan particiones de conjuntos y no operaciones algebraicas. Manejan eficientemente categorías asignadas a enteros sin sufrir

por la "falsa ordinalidad", manteniendo la dimensionalidad original y mejorando la eficiencia computacional.

3) Algoritmos Seleccionados: A diferencia de enfoques previos limitados a árboles, este estudio evalúa cinco paradigmas para garantizar robustez:

1. **Regresión Logística (Baseline):** Implementada manualmente mediante descenso de gradiente. Evalúa la separabilidad lineal del problema tras la expansión de dimensiones.
2. **K-Nearest Neighbors (KNN):** Modelo basado en instancias. Evalúa la hipótesis de que hongos con morfología similar comparten toxicidad. Se utiliza con datos estandarizados.
3. **Random Forest:** Método de ensamble (Bagging). Seleccionado por su capacidad para modelar interacciones no lineales complejas y su interpretabilidad (importancia de variables).
4. **Support Vector Machines (SVM):** Máquina de kernel (RBF). Se incluye para evaluar la proyección a espacios de alta dimensión para encontrar fronteras de decisión complejas.
5. **Multi-Layer Perceptron (MLP):** Red Neuronal Artificial. Evalúa la capacidad de aprendizaje de representaciones latentes mediante una arquitectura conexionista.

4) Configuración Experimental: La estrategia de validación se configuró de la siguiente manera:

- **División de Datos:** Se utilizó un esquema de partición **70% entrenamiento** y **30% prueba**.
- **Validación Cruzada:** Se empleó **Stratified K-Fold** ($k = 4$) para el ajuste de hiperparámetros, asegurando que la proporción de clases (51.8%/48.2%) se mantenga constante en cada validación.
- **Métrica Prioritaria:** Dado el riesgo fatal de intoxicación, se prioriza el **Recall (Sensibilidad)** para la clase `POISONOUS`, buscando minimizar estrictamente los Falsos Negativos.

III. Estado del Arte

En esta sección se revisan trabajos recientes sobre la **clasificación comestible vs. venenoso** en el dataset *Agaricus–Lepiota* del repositorio UCI. El objetivo es identificar qué configuraciones de aprendizaje, técnicas, metodologías de validación y métricas se han utilizado, así como los niveles de desempeño alcanzados, para extraer lineamientos que guíen el diseño experimental de este proyecto.

Se analizan cuatro trabajos representativos: *Edibility Detection of Mushroom Using Ensemble Methods* [2], *Ensemble Learning based Classification of Edible and Poisonous Agaricus Mushrooms* [3], *Edible Mushroom Identification Using Machine Learning* [4] y *Edible and Poisonous Mushrooms Classification by Machine Learning Algorithms* [5].

A. Trabajos revisados

a) *Edibility Detection of Mushroom Using Ensemble Methods* (2019). Trabajo experimental centrado en **ensembles de árboles**. Los autores comparan *Bagging*, *AdaBoost* y *Random Forest* sobre UCI empleando (i) subconjuntos de rasgos construidos con criterios de relevancia y (ii) subconjuntos aleatorios como control. Usan un esquema *hold-out* (2/3–1/3) y una evaluación adicional sobre el conjunto completo. Los mejores resultados se obtienen con **Random Forest** y **Bagging**, con *accuracy* $\approx 99.9\%$ y una varianza de error muy baja, lo que sugiere robustez frente a cambios moderados en la selección de rasgos [2]. Además, destacan que, para atributos puramente categóricos, los árboles manejan bien *label encoding* sin requerir escalado, y que la agregación reduce notablemente el sobreajuste.

b) *Ensemble Learning based Classification of Edible and Poisonous Agaricus Mushrooms* (2024). Comparativa sistemática de *Random Forest*, *Bagging*, *AdaBoost*, *Gradient Boosting* y *XGBoost* con **validación estratificada k-fold (k=10)** y partición 70/30. El estudio aplica un filtro por importancia de características y elimina dos rasgos de baja contribución, lo que simplifica el modelo sin degradar el desempeño. Con esta configuración, **AdaBoost** obtiene la mejor media de CV ($\sim 95\%$) y una *accuracy* en prueba cercana al 96% [3]. El trabajo enfatiza buenas prácticas de *model selection*: comparación cruzada consistente, uso de métricas complementarias (*precision/recall/F1*) y documentación del *pipeline* (preprocesamiento, particionado, e hiperparámetros).

c) *Edible Mushroom Identification Using Machine Learning* (2022). Estudio comparativo con clasificadores "clásicos": C4.5, SVM, Regresión Logística y Naive Bayes, tras *label encoding* de las variables categóricas. **C4.5** resulta superior dentro de este grupo ($\approx 93\%$ de *accuracy*), seguido por SVM ($\approx 90\%$) y Regresión Logística ($\approx 86\%$); Naive Bayes queda rezagado ($\approx 62\%$) [4]. Los autores remarcan la **interpretabilidad** del árbol (reglas explícitas) y la sensibilidad de SVM/LR a la codificación y a la dimensionalidad cuando se usa *one-hot*.

d) *Edible and Poisonous Mushrooms Classification by Machine Learning Algorithms* (2022). Tutuncu et al. abordan el mismo dataset Agaricus–Lepiota con un enfoque comparativo de cuatro algoritmos: **Naive Bayes (NB), Decision Tree (DT), Support Vector Machine (SVM) y AdaBoost (AB)** [5]. Utilizan las 22 características categóricas originales y un esquema de **validación cruzada k-fold con** $k = 10$, calculando las métricas de exactitud, precisión, *recall* y F1 a partir de la matriz de confusión. El conjunto de datos incluye 8 124 instancias: 4 208 comestibles y 3 916 venenosas. Los resultados muestran que NB alcanza una exactitud de 90.99%, DT 98.82%, SVM 99.98% y AdaBoost **100%** de exactitud, con valores de precisión, *recall* y F1-Score también máximos para AdaBoost. Los autores concluyen que, con las características morfológicas del dataset, es posible determinar la comestibilidad de un hongo con **100% de éxito** usando AdaBoost.

B. Lecciones metodológicas y recomendaciones

A partir de los trabajos revisados pueden extraerse varias lecciones que orientan el diseño experimental de este proyecto:

1) **Paradigma y tipo de datos.** Todos los estudios tratan el problema como **clasificación supervisada binaria** usando el dataset Agaricus–Lepiota, basado exclusivamente en **características morfológicas categóricas**. Esto

respalda el enfoque de este trabajo, que emplea el mismo dataset y la misma formulación (edible vs. poisonous).

2) **Elección de modelos.** La literatura coincide en que los modelos basados en árboles y **ensembles** (DT, Random Forest, AdaBoost) alcanzan un desempeño muy alto e incluso perfecto en algunos casos [2], [5]. No obstante, varios autores comparan también SVM, Regresión Logística y Naive Bayes [4], [5]. En coherencia con ello, nuestro proyecto evalúa un conjunto *heterogéneo* de cinco modelos: Regresión Logística, KNN, Random Forest, SVM y MLP, usando Random Forest como candidato principal para una solución robusta, y la Regresión Logística como línea base lineal.
3) **Codificación de variables categóricas.** Los trabajos revisados aplican algún tipo de codificación numérica, pero en general no profundizan en cómo esta decisión afecta a los distintos algoritmos. A partir de esta revisión, nuestro proyecto adopta una **estrategia híbrida de codificación**:
   - *One-Hot Encoding* para modelos basados en distancia o en geometría del espacio (Regresión Logística, KNN, SVM y MLP), evitando introducir relaciones de orden artificiales entre categorías.
   - *Label Encoding* para Random Forest, aprovechando que los árboles trabajan con particiones de conjuntos y no requieren distancias métricas.

   Adicionalmente, el valor faltante en `stalk-root` se trata como una **categoría explícita** en lugar de eliminar registros, siguiendo la recomendación de preservar la mayor cantidad de datos posible cuando el porcentaje de faltantes es alto.
4) **Metodología de validación.** Los artículos consultados emplean particiones entrenamiento/prueba en combinación con **validación cruzada k-fold** (típicamente $k = 10$) para estimar el rendimiento de manera estable y reducir la varianza [3], [5]. En nuestro trabajo se adopta un esquema coherente con estas prácticas: partición 70/30 en entrenamiento y prueba, y validación cruzada *Stratified K-Fold* para el ajuste de hiperparámetros, lo que garantiza que la proporción de clases se mantenga en cada pliegue.
5) **Métricas y criterio de evaluación.** Mientras que muchos trabajos reportan principalmente **exactitud global**, los estudios que emplean matrices de confusión y métricas derivadas (precisión, *recall*, F1) muestran que el desempeño puede ser muy diferente por clase [4], [5]. Dado que el error más crítico en este dominio es clasificar un hongo venenoso como comestible, en este proyecto se prioriza explícitamente el **recall de la clase venenosa**, usando la exactitud global sólo como métrica secundaria. La revisión de Tutuncu et al. confirma que modelos como AdaBoost pueden alcanzar simultáneamente *recall* y precisión del 100%, constituyendo una referencia para evaluar nuestros modelos.
6) **Simplicidad vs. robustez y explicabilidad.** Aunque modelos de *boosting* pueden lograr desempeño perfecto, los árboles individuales (C4.5, DT) siguen siendo valiosos por su **interpretabilidad** y por permitir extraer reglas explícitas sobre los atributos más peligrosos [4]. En este trabajo se toma como recomendación mantener un equilibrio entre modelos de alta capacidad (Random Forest, SVM, MLP) y modelos relativamente sencillos pero explicables (Regresión Logística, árboles), utilizando la importancia de características y las reglas extraídas del bosque aleatorio para apoyar tareas de educación y prevención.

IV. Entrenamiento y Evaluación de los Modelos

En esta sección se describe el diseño experimental seguido para entrenar y evaluar los cinco modelos de Machine Learning considerados en el proyecto, así como los resultados obtenidos en los conjuntos de entrenamiento, validación y prueba.

A. Configuración experimental

1) Esquema de validación: Se utilizó un esquema de validación en dos niveles:

- **División principal (hold-out estratificado).** El conjunto de datos Agaricus–Lepiota se dividió aleatoriamente en **70% para entrenamiento** y **30% para prueba**, manteniendo la proporción original entre clases comestibles y venenosas. Esta división se realizó una única vez con una semilla fija para garantizar reproducibilidad.
- **Validación cruzada sobre el conjunto de entrenamiento.** Para los modelos 2–5 (k-NN, Random Forest, MLP y SVM) se empleó **Stratified K-Fold** con $k = 4$ pliegues sobre el subconjunto de entrenamiento. En cada combinación de hiperparámetros, el modelo se entrenó en tres pliegues y se evaluó en el pliegue restante, repitiendo el proceso hasta cubrir los cuatro pliegues. El desempeño promedio en validación se utilizó como criterio de selección de hiperparámetros.

No fue necesario aplicar técnicas de submuestreo o sobremuestreo, ya que el dataset está moderadamente balanceado (51.8% comestibles, 48.2% venenosos). En todos los casos se fijaron semillas de inicialización para obtener resultados reproducibles.

2) Preprocesamiento y codificación: Siguiendo las decisiones descritas en la Sección 2, se trabajó con dos representaciones numéricas del mismo conjunto de datos:

- **Flujo One-Hot Encoding.** Las 22 variables categóricas se transformaron en 117 características binarias mediante *One-Hot Encoding*. Esta representación se utilizó para la **Regresión Logística (Modelo 1)**, **k-NN (Modelo 2)**, **MLP (Modelo 4)** y **SVM (Modelo 5)**, que son modelos basados en distancia o en productos escalares. Posteriormente se aplicó *StandardScaler* para centrar y escalar las características.
- **Flujo Label Encoding.** Para **Random Forest (Modelo 3)** se utilizó *Label Encoding*, asignando un entero a cada categoría. Los árboles de decisión trabajan con particiones de conjuntos y no requieren distancias métricas, por lo que no se ve afectado el modelo por la "ordinalidad artificial" de los enteros.

En la variable `stalk-root`, los valores faltantes ("?") se trataron como una categoría adicional explícita, evitando

descartar casi un tercio de las muestras y permitiendo que los modelos aprendan patrones asociados a la ausencia de esta información.

3) Modelos e hiperparámetros evaluados: Cada proyecto debía comparar al menos cinco modelos: uno paramétrico lineal, un modelo no paramétrico, un ensamble de árboles de decisión, una red neuronal artificial y una máquina de vectores de soporte. La Tabla II resume los hiperparámetros y mallas de valores evaluadas en el notebook.

TABLE II
MODELOS Y ESPACIOS DE HIPERPARÁMETROS EVALUADOS

Modelo	Hiperparámetros evaluados	Detalle de validación
Regresión Logística (M1)	$\eta = 0{,}1$; iteraciones $T = 100$; función de costo: entropía cruzada binaria; algoritmo: gradiente descendente; inicialización en ceros.	Hold-Out 70%/30%; 1 experimento; codificación: One-Hot (117 features).
k-NN (M2)	$k \in \{3, 5, 7, 11, 15, 21, 31, 41, 51\}$; distancia Euclídea; ponderación uniforme; algoritmo *brute force*.	Fase 1: Train/Test 80%/20% para selección de k; Fase 2: StratifiedKFold (4 pliegues); 9 configuraciones; One-Hot (117 features).
Random Forest (M3)	$n\_estimators \in \{50, 100, 150, 200\}$; $max\_features \in \{5, 10, 15, 20\}$; $min\_samples\_leaf = 3$; *criterion*=gini; *random_state*=42.	StratifiedKFold (4 pliegues); 16 combinaciones $\Rightarrow$ 64 experimentos; Label Encoding (22 features).
MLP (M4)	`hidden_layer_sizes`: 12 arquitecturas (1, 2 y 3 capas); *activation*=relu; *solver*=adam; *max_iter*=350; *random_state*=1.	StratifiedKFold (4 pliegues); 12 arquitecturas $\Rightarrow$ 48 experimentos; One-Hot (117 features).
SVM (M5)	*kernel* $\in \{linear, rbf, poly\}$; $\gamma \in \{0.001, 0.01, 0.1, 1.0\}$ (para rbf y poly); $C \in \{0.1, 1, 10, 100\}$; *random_state*=42.	StratifiedKFold (4 pliegues); 36 configuraciones $\Rightarrow$ 144 experimentos; One-Hot (117 features).

4) Métricas de desempeño: Para cada modelo y configuración de hiperparámetros se calcularon las siguientes métricas, tanto en validación cruzada como en el conjunto de prueba:

- **Exactitud (Accuracy).** Proporción de clasificaciones correctas sobre el total de muestras.
- **Precisión para la clase venenosa.** Fracción de hongos realmente venenosos entre todos los que el modelo predice como venenosos. Indica la confianza en las alertas generadas.
- **Recall (sensibilidad) para la clase venenosa.** Fracción de hongos venenosos correctamente identificados por el modelo. Dado que el error más grave es un *Falso Negativo* (clasificar un hongo venenoso como comestible), esta métrica se adopta como **criterio principal de evaluación**.
- **F1-Score para la clase venenosa.** Media armónica entre precisión y *recall*, útil para equilibrar ambos criterios.
- **Matriz de confusión.** Para analizar de forma explícita la cantidad de falsos positivos y falsos negativos por modelo.

La comparación entre modelos se basa principalmente en el *recall* de la clase venenosa y, en segundo lugar, en la exactitud global y el F1-Score. El tiempo de entrenamiento se utilizó como referencia secundaria para escoger entre modelos con desempeño similar.

B. Resultados del entrenamiento de modelos

1) Desempeño global: Los resultados numéricos completos se resumen en la Tabla III. De forma cualitativa, los experimentos muestran que:

- Todos los modelos alcanzan un **alto desempeño** en el conjunto de prueba gracias a la riqueza de información morfológica del dataset.
- Los modelos no lineales (k-NN, Random Forest, SVM y MLP) obtienen exactitudes cercanas a la perfección y *recall* muy altos para la clase venenosa, coherentes con los resultados reportados en la literatura.
- La Regresión Logística, usada como *baseline* lineal, presenta un desempeño correcto pero inferior al de los modelos no lineales, especialmente cuando se evalúa el *recall* de la clase venenosa. Esto sugiere que la frontera de decisión no es estrictamente lineal en el espacio de características.

TABLE III
RESUMEN DEL DESEMPEÑO DE LOS MODELOS EN EL CONJUNTO DE PRUEBA

Modelo	Accuracy	Precisión (ven.)	Recall (ven.)	F1 (ven.)
Regresión Logística	0.9815	0.9626	0.9991	0.9805
k-NN	0.9992	1.0000	0.9982	0.9991
Random Forest	1.0000	1.0000	1.0000	1.0000
MLP	1.0000	1.0000	1.0000	1.0000
SVM	0.9992	1.0000	0.9982	0.9991

En términos de estabilidad, los modelos de ensamble y SVM muestran una **brecha pequeña entre entrenamiento y prueba**, lo que indica buena capacidad de generalización. En cambio, algunas configuraciones profundas de MLP tienden a sobreajustar si se aumenta excesivamente el número de capas y neuronas, sin ganancias significativas en las métricas.

2) Resultados detallados por modelo:

a) Modelo 1: Regresión Logística: La Figura 1 muestra la evolución del costo de entrenamiento (entropía cruzada binaria) a lo largo de 100 iteraciones del gradiente descendente. El costo disminuye de manera monótona desde un valor inicial cercano a 0.69 hasta un costo final de aproximadamente 0.036, lo que indica que el algoritmo converge de forma estable con la tasa de aprendizaje seleccionada.

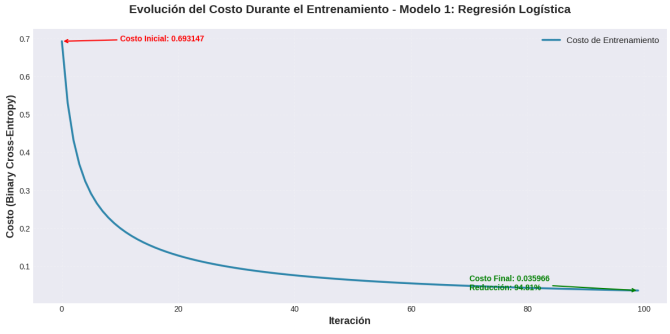

Fig. 1. Evolución del costo durante el entrenamiento para el Modelo 1 (Regresión Logística).

En la Figura 2 se presentan las matrices de confusión para

los conjuntos de entrenamiento y prueba. En ambos casos el número de errores es muy bajo; en particular, en el conjunto de prueba solo se observa un caso en el que un hongo venenoso es clasificado como comestible, lo que implica un *recall* casi perfecto para la clase venenosa.

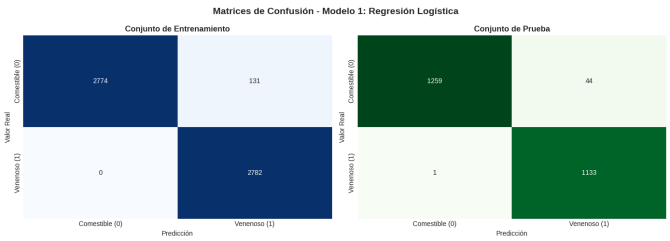

Fig. 2. Matrices de confusión en entrenamiento y prueba para el Modelo 1 (Regresión Logística).

Finalmente, la Figura 3 compara las métricas de *accuracy*, precisión, *recall* y F1 entre entrenamiento y prueba. Las diferencias son pequeñas (del orden de unas pocas décimas de punto porcentual), lo que sugiere que el modelo generaliza bien y no presenta un sobreajuste severo, aunque su desempeño es inferior al de los modelos no lineales analizados en las siguientes subsecciones.

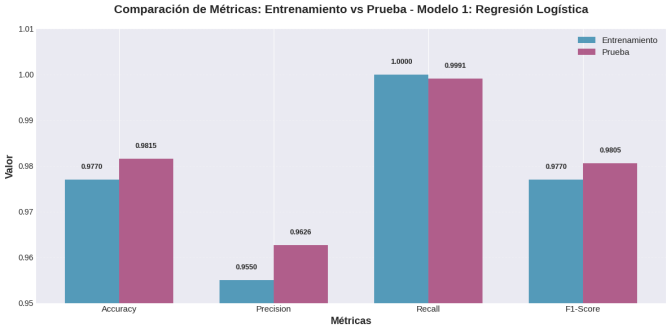

Fig. 3. Comparación de métricas (entrenamiento vs. prueba) para el Modelo 1 (Regresión Logística).

b) Modelo 2: k-vecinos más cercanos (k-NN): La figura de validación de hiperparámetros (Figura 4) muestra la *accuracy* promedio y su intervalo de confianza ($\pm 1\sigma$) para distintos valores de k. Se observa que valores muy grandes de k suavizan en exceso la frontera de decisión y reducen el desempeño en validación. El mejor compromiso entre robustez y sensibilidad se obtiene en $k = 3$, donde la *accuracy* promedio de validación es cercana a 0.9998 y las curvas de entrenamiento y validación permanecen muy próximas, lo que indica una excelente capacidad de generalización.

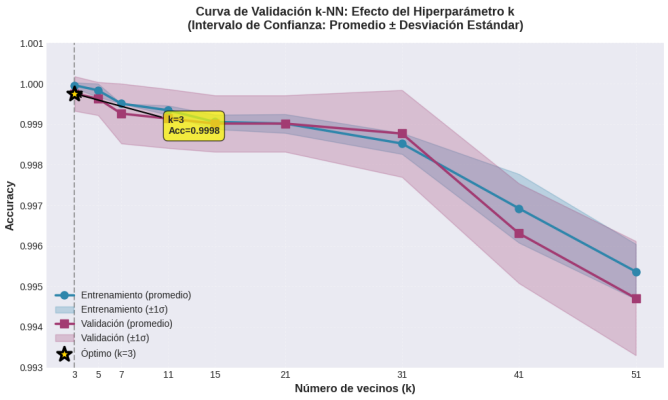

Fig. 4. Curva de validación del hiperparámetro k para el modelo k-NN. Se muestran las medias de *accuracy* en entrenamiento y validación, junto con las bandas de $\pm 1\sigma$ y el valor óptimo identificado en $k = 3$.

c) Modelo 3: Random Forest: La exploración de hiperparámetros para Random Forest se resume en la Figura 5. Los mapas de calor de $n\_estimators$ vs. $max\_features$ muestran que aumentar el número de árboles reduce la varianza hasta un punto de saturación (alrededor de 150–200 árboles), mientras que valores intermedios de $max\_features$ (en torno a 10) maximizan la exactitud y el *recall* en validación. La *accuracy* de entrenamiento es prácticamente perfecta en casi toda la malla, y la de validación se mantiene muy cercana a 1.0, lo que indica un ensamble estable y con buena capacidad de generalización.

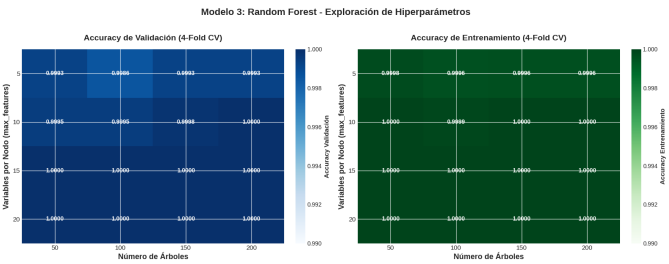

Fig. 5. Exploración de hiperparámetros para Random Forest. Mapas de calor de *accuracy* promedio en validación (izquierda) y entrenamiento (derecha) en función del número de árboles ($n\_estimators$) y del número de variables consideradas por nodo ($max\_features$).

d) Modelo 4: Red neuronal artificial (MLP): La Figura 6 cuantifica la complejidad de cada arquitectura MLP utilizada, medida como número total de parámetros entrenables. Se aprecia un crecimiento casi lineal con el número de neuronas por capa y aproximadamente proporcional al número de capas: pasar de una arquitectura sencilla (1 capa, 10 neuronas) a arquitecturas profundas (3 capas, 50 neuronas) incrementa el número de parámetros en casi un orden de magnitud.

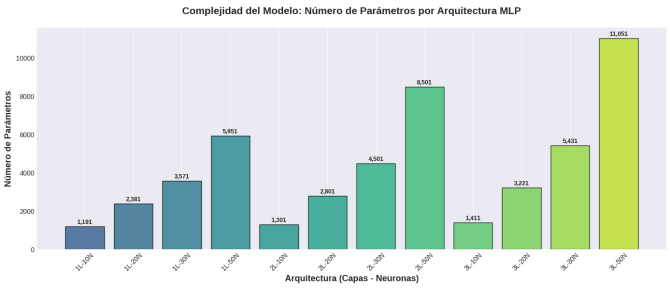

Fig. 6. Complejidad del modelo MLP: número de parámetros entrenables por arquitectura (combinaciones de capas y neuronas).

Sin embargo, la Figura 7 muestra que el error de validación apenas mejora a medida que aumenta la complejidad. En el panel izquierdo se representa el error frente al número de neuronas por capa para diferentes cantidades de capas ocultas, mientras que el panel derecho fija el número de neuronas y varía el número de capas. En ambos casos las curvas son casi planas y los intervalos de confianza se solapan, lo que indica que las arquitecturas más profundas no aportan mejoras significativas y sí aumentan el riesgo de sobreajuste y el costo computacional. Una arquitectura poco profunda con un número moderado de neuronas resulta suficiente para este problema.

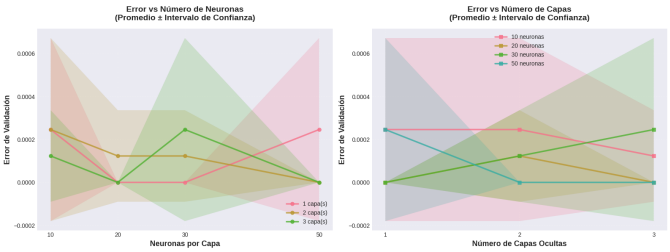

Fig. 7. Error de validación del MLP en función del número de neuronas por capa (izquierda) y del número de capas ocultas (derecha). Se muestran los promedios e intervalos de confianza para cada configuración.

e) Modelo 5: Máquina de vectores de soporte (SVM): Para SVM se evaluaron distintos kernels (lineal, RBF y polinómico) y combinaciones de los hiperparámetros C y γ. Los resultados del notebook muestran que los kernels no lineales superan claramente al kernel lineal, alcanzando *accuracy* y *recall* casi perfectos para la clase venenosa. La Figura 8 resume además la complejidad del modelo medida como número promedio de vectores de soporte: el kernel lineal utiliza alrededor de 166 vectores (aprox. 4% de las muestras), mientras que los kernels polinómico y RBF requieren entre 1 600 y 2 500 vectores (39–60%), lo que indica hiperplanos más complejos.

Valores excesivamente altos de C y γ producen una brecha mayor entre entrenamiento y validación, lo que sugiere sobreajuste; los mejores resultados se obtienen con parámetros moderados que equilibran un margen amplio con un número razonable de vectores de soporte y pocos errores en la clase crítica.

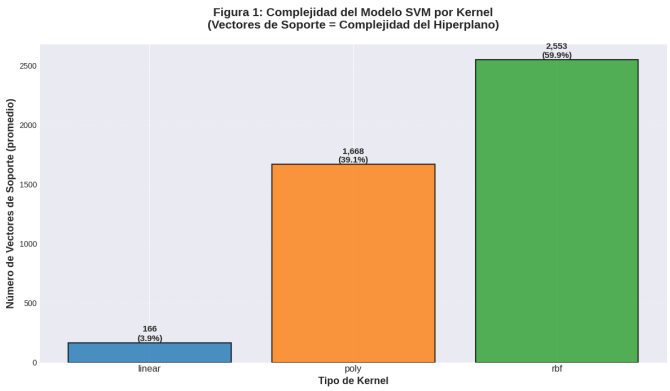

Fig. 8. Complejidad del modelo SVM por tipo de kernel, medida como número promedio de vectores de soporte utilizados en el hiperplano de decisión.

En conjunto, los resultados respaldan la elección de **Random Forest** como modelo principal del sistema, dado que ofrece un desempeño casi perfecto en la identificación de hongos venenosos, una brecha entrenamiento/prueba muy reducida y una interpretación razonable mediante la importancia de características. Los modelos k-NN, SVM y MLP actúan como referencias adicionales que confirman que el problema es altamente linealmente separable en un espacio no lineal de características y que múltiples paradigmas de aprendizaje supervisado pueden resolverlo con alto nivel de seguridad.

V. Reducción de dimensión

En esta fase se evaluaron técnicas de reducción de dimensión con el fin de disminuir la complejidad de los modelos manteniendo un desempeño de clasificación comparable al obtenido con el conjunto completo de variables. Se consideraron tanto métodos lineales (PCA) como no lineales (UMAP), y se re-entrenaron los dos modelos con mejor desempeño global en la Sección IV-B: k-NN con $k = 3$ y Random Forest con 200 árboles y $max\_features = 10$.

A. Análisis individual de variables

Siguiendo las recomendaciones del enunciado del proyecto, se inspeccionaron las variables originales a partir de estadísticas descriptivas, distribuciones y medidas de asociación (correlaciones y capacidad discriminativa frente a la etiqueta de clase). Este análisis evidenció que:

- Varias variables codifican información altamente redundante (por ejemplo, atributos morfológicos que se derivan de la misma característica básica del hongo).
- Algunas características presentan baja variabilidad o un poder discriminativo marginal entre clases, lo que sugiere que pueden ser reemplazadas por combinaciones lineales de otras variables sin pérdida apreciable de información.

Esta observación motivó el uso de métodos de reducción de dimensión para concentrar la información relevante del conjunto de 117 características originales en un espacio de menor dimensión.

B. Extracción de características lineal: PCA

Para la reducción lineal se utilizó *Principal Component Analysis* (PCA) sobre la matriz de características de entrada. La Figura 9 muestra la curva de varianza explicada acumulada por las primeras componentes principales.

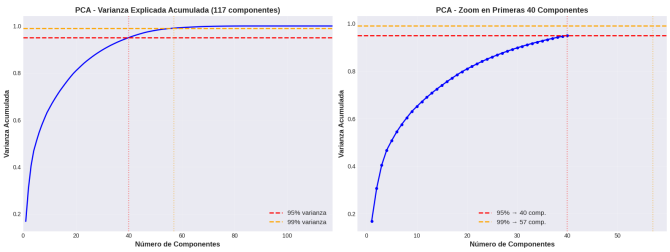

Fig. 9. Varianza explicada acumulada por PCA. Se observa que con aproximadamente 40 componentes se captura cerca del 95% de la varianza total, mientras que con 57 componentes se alcanza alrededor del 99%.

A partir de esta curva se definieron dos configuraciones de reducción:

- **PCA 95%**: proyección a 40 componentes principales (reducción de dimensión de 117 a 40 variables, es decir, un ˜66% de reducción).
- **PCA 99%**: proyección a 57 componentes principales (reducción de ˜51% respecto al espacio original).

Sobre cada una de estas representaciones se re-entrenaron los dos modelos seleccionados.

a) Efecto de PCA en k-NN: En la Figura 10 se comparan las métricas de *accuracy* en entrenamiento y prueba para el modelo k-NN con $k = 3$, utilizando las características originales y las representaciones reducidas con PCA.

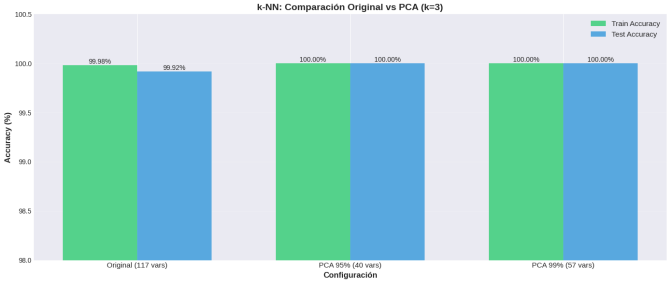

Fig. 10. Comparación del desempeño de k-NN ($k = 3$) con el conjunto original de 117 variables y con las representaciones reducidas por PCA (40 y 57 componentes).

Los resultados muestran que la proyección a 40 componentes (PCA 95%) mantiene e incluso mejora ligeramente la *accuracy* en entrenamiento y prueba, alcanzando valores prácticamente perfectos. Al aumentar a 57 componentes (PCA 99%) el desempeño se mantiene igual de alto, lo que indica que la mayor parte de la información relevante para k-NN se concentra en un subespacio de baja dimensión.

b) Efecto de PCA en Random Forest: La Figura 11 muestra el efecto de aplicar PCA al modelo Random Forest con $n_{estimators} = 200$ y $max\_features = 10$, comparando el desempeño entre el conjunto original de variables (22 características codificadas con *Label Encoding*) y dos versiones reducidas mediante PCA (95% y 99% de varianza explicada).

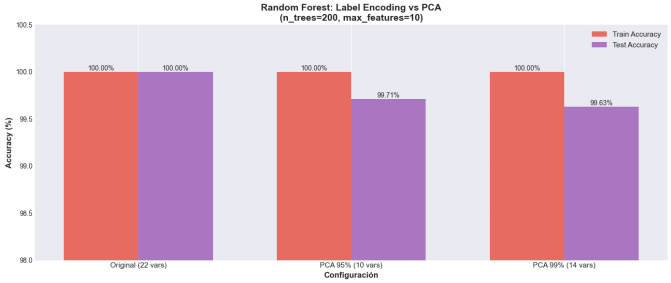

Fig. 11. Random Forest: comparación del desempeño entre el conjunto original de variables (22 características con *Label Encoding*) y las representaciones reducidas mediante PCA al 95% (10 componentes) y 99% (14 componentes) de varianza explicada.

Se observa que la *accuracy* de entrenamiento se mantiene en 100% para las tres configuraciones, mientras que la *accuracy* de prueba pasa de un 100% en el espacio original a aproximadamente 99.71% con PCA 95% y 99.63% con PCA 99%. Es decir, la reducción de dimensión introduce únicamente una caída muy pequeña (del orden de 0.3–0.4 puntos porcentuales) en el desempeño sobre el conjunto de prueba.

En conjunto, estos resultados indican que Random Forest es bastante robusto frente a la reducción lineal de dimensión: es posible comprimir las 22 variables originales a 10–14 componentes principales con una pérdida prácticamente despreciable en *accuracy*. Por tanto, si se prioriza la simplicidad del modelo o el coste computacional, la configuración con PCA (95% de varianza, 10 componentes) ofrece un buen compromiso entre complejidad y desempeño.

C. Extracción de características no lineal: UMAP

Como técnica de extracción no lineal se empleó *Uniform Manifold Approximation and Projection* (UMAP) para proyectar los datos a un espacio de baja dimensión preservando relaciones de vecindad. Además de servir como herramienta de visualización, se evaluó su impacto en el desempeño de dos de los modelos más fuertes del estudio: k-NN ($k = 3$) y Random Forest.

En ambos casos se comparó el desempeño del clasificador sobre:

1) El conjunto original de variables (117 variables codificadas con *One-Hot* en k-NN y 22 variables con *Label Encoding* en Random Forest).
2) Las representaciones lineales obtenidas con PCA (40 y 57 componentes para 95% y 99% de varianza explicada en k-NN, y 10 y 14 componentes para 95% y 99% de varianza en Random Forest).
3) Las representaciones no lineales UMAP con distintas dimensiones (2, 5, 10 y 20 componentes).

La Figura 12 resume la comparación para k-NN.

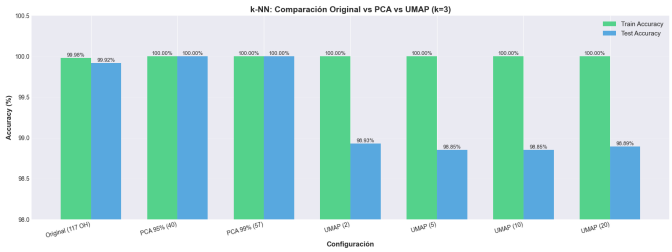

Fig. 12. k-NN ($k = 3$): comparación del desempeño con el conjunto original, con PCA (95% y 99% de varianza explicada) y con UMAP (2, 5, 10 y 20 componentes).

En k-NN se observa que:

- Con las variables originales, el modelo alcanza una *accuracy* de entrenamiento cercana al 100% (99.98%) y una *accuracy* de prueba de aproximadamente 99.92%.
- Al aplicar PCA (40 y 57 componentes), tanto la *accuracy* de entrenamiento como la de prueba se vuelven prácticamente perfectas (100% en ambos conjuntos). Esto indica que la proyección lineal consigue eliminar ruido y redundancia sin perder información discriminativa.
- Cuando se proyecta con UMAP a espacios de muy baja dimensión (2–20 componentes), la *accuracy* de entrenamiento se mantiene en 100%, pero la *accuracy* de prueba desciende hasta valores cercanos al 98.9%. La pérdida sigue siendo moderada, pero mayor que en el caso de PCA, lo que sugiere que parte de la estructura relevante para k-NN se distorsiona en la proyección no lineal.

De forma análoga, la Figura 13 muestra la comparación para Random Forest con $n\_estimators = 200$ y $max\_features = 10$.

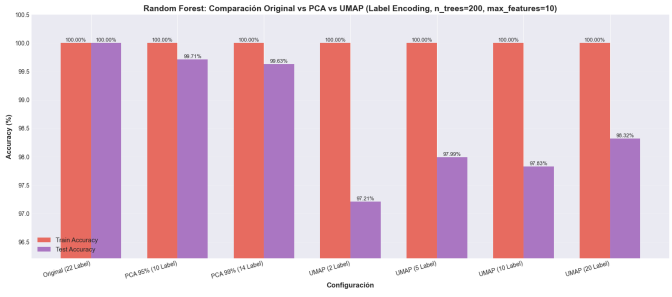

Fig. 13. Random Forest: comparación del desempeño con el conjunto original, con PCA (95% y 99% de varianza explicada) y con UMAP (2, 5, 10 y 20 componentes).

En este caso, los resultados indican que:

- Con las variables originales, Random Forest alcanza *accuracy* prácticamente perfecta tanto en entrenamiento como en prueba (100% en ambos conjuntos).
- Al aplicar PCA con 10 y 14 componentes (95% y 99% de varianza), la *accuracy* de entrenamiento se mantiene en 100%, mientras que la *accuracy* de prueba se reduce ligeramente hasta valores en torno al 99.6–99.7%. La pérdida de desempeño es muy pequeña, lo que indica que es posible comprimir fuertemente la información lineal sin degradar de manera relevante la capacidad de clasificación.
- En las representaciones UMAP de muy baja dimensión (2–20 componentes), la *accuracy* de entrenamiento sigue siendo 100%, pero la *accuracy* de prueba desciende de forma más notable, con valores que oscilan aproximadamente entre 97% y 98.5%. Aunque el desempeño sigue siendo alto, la brecha entrenamiento/prueba es mayor que con PCA, evidenciando una pérdida de información discriminativa en la proyección.

En conjunto, los experimentos muestran que PCA constituye una alternativa muy atractiva para reducir la dimensión del problema: permite pasar del espacio original de características a unas pocas decenas de componentes con una pérdida casi despreciable en *accuracy*, e incluso con ligeras mejoras en el caso de k-NN. UMAP, por su parte, posibilita compresiones aún más agresivas del espacio de características y produce modelos con desempeño todavía muy alto, a costa de una reducción más evidente en la *accuracy* de prueba. Esto lo hace especialmente útil como herramienta de visualización y análisis exploratorio de la estructura no lineal del conjunto de hongos.

REFERENCES

[1] D. Dua and C. Graff, "Mushroom data set," UCI Machine Learning Repository, 2019.

[2] N. J. Pinky, S. M. M. Islam, and R. S. Alice, "Edibility detection of mushroom using ensemble methods," *International Journal of Image, Graphics and Signal Processing*, vol. 11, no. 4, pp. 55–62, 2019.

[3] R. Sahu, S. Pandey, R. Verma, and P. Pandey, "Ensemble learning based classification of edible and poisonous agaricus mushrooms," in *2024 Fourth International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, 2024.

[4] K. Kousalya, B. Krishnakumar, S. Boomika, N. Dharati, and N. Hemavathy, "Edible mushroom identification using machine learning," in *2022 International Conference on Computer Communication and Informatics (ICCCI)*, 2022.

[5] K. Tutuncu, I. Cinar, R. Kursun, and M. Koklu, "Edible and poisonous mushrooms classification by machine learning algorithms," in *2021 10th Mediterranean Conference on Embedded Computing (MECO)*. IEEE, 2022, pp. 1–4.