Jonathan Espinoza 20022
Juan Andrés Galicia 20298
Oscar Méndez 20402
Juan Fernando Ramírez 20666
Jeyner Arango 201106

# Laboratorio 1

## Ejercicio 1

En este caso trabajamos con Anaconda y Jupyter Notebooks. Anaconda es una distribución de Python que facilita la gestión de paquetes y entornos, lo que es ideal para proyectos de Machine Learning. Con Anaconda, creamos un entorno virtual específico para nuestro proyecto, lo que nos permitió instalar solo las bibliotecas necesarias sin interferir con otros proyectos. Luego, utilizamos Jupyter Notebooks, que es una herramienta interactiva que nos permite escribir y ejecutar código en celdas, facilitando la visualización de resultados y la documentación del proceso. Jupyter también nos permite incluir texto, gráficos y visualizaciones, lo que hace que la exploración de datos y el desarrollo de modelos sean más intuitivos y colaborativos. Esta combinación de herramientas nos proporcionó un entorno de desarrollo robusto y flexible para nuestro trabajo en Machine Learning.

## Ejercicio 2

Para esta fase creamos la estructura básica de un proyecto de Machine Learning. Primero, organizamos los archivos en tres directorios principales: *data* para almacenar los datos crudos y procesados, *models* para los modelos entrenados, y *docs* para la documentación del proyecto. Luego, inicializamos un repositorio Git en la carpeta raíz del proyecto. Incluimos un archivo .gitignore para excluir archivos no deseados, y también añadimos un archivo README.md para describir brevemente el propósito y la estructura del proyecto.

## Ejercicio 3

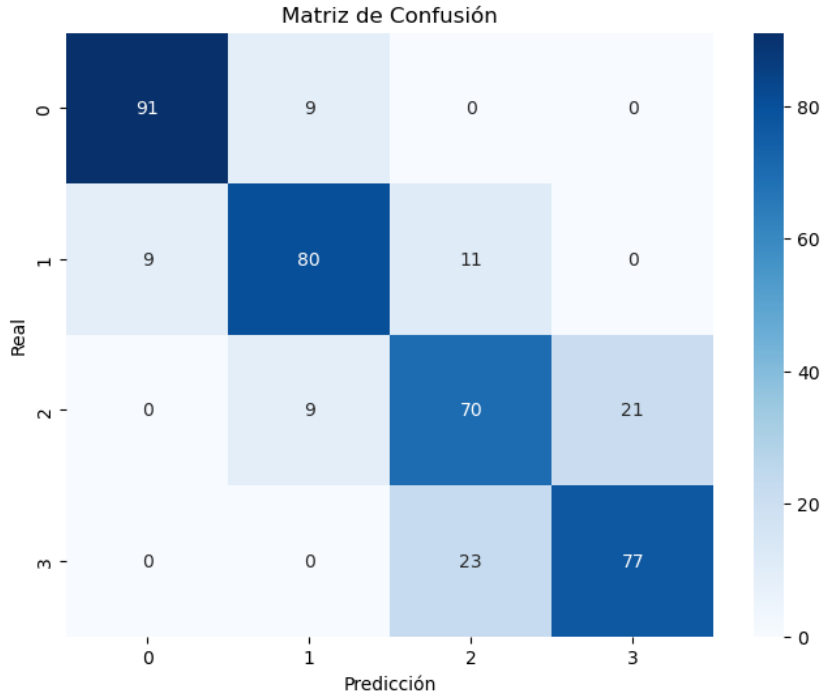
En este ejercicio, realizamos un análisis exploratorio de los datos con el objetivo de comprender mejor las características de nuestro conjunto de datos. Primero, exploramos la cantidad de valores nulos por columna, lo que nos permitió identificar posibles problemas de calidad de datos. También revisamos los tipos de datos de cada columna, lo que es esencial para aplicar las transformaciones adecuadas durante el preprocesamiento. Además, analizamos la cantidad de observaciones únicas en cada variable, lo que nos ayudó a comprender la variabilidad de los datos.

Un aspecto clave del EDA fue la distribución de la variable de respuesta, donde visualizamos cómo se distribuyen las diferentes categorías del target. Para esto, realizamos gráficos de caja (boxplots) y gráficos de barras apiladas al 100% que nos permitieron observar la incidencia de variables, ayudándonos a identificar patrones y relaciones entre las características del dataset y la variable de respuesta.

**Ejercicio 4**

En este ejercicio, comenzamos probando un modelo de árbol de decisión básico para establecer una línea base y evaluar su rendimiento. El clasificador proporcionó los siguientes resultados:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.91 | 0.91 | 0.91 | 100 |
| 1 | 0.82 | 0.80 | 0.81 | 100 |
| 2 | 0.67 | 0.70 | 0.69 | 100 |
| 3 | 0.79 | 0.77 | 0.78 | 100 |
| accuracy | | | 0.80 | 400 |
| macro avg | 0.80 | 0.80 | 0.80 | 400 |
| weighted avg | 0.80 | 0.80 | 0.80 | 400 |

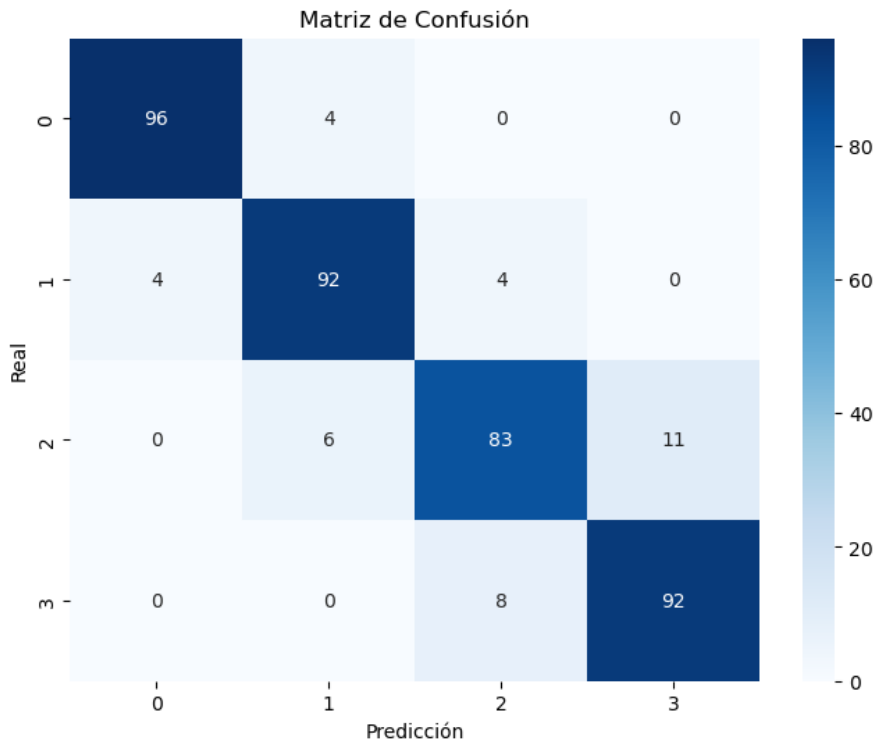



Los resultados mostraron que el modelo de árbol de decisión tuvo una precisión general del 80%. Sin embargo, notamos que las clases 2 y 3 tuvieron un rendimiento más bajo en comparación con las clases 0 y 1, lo que sugiere que hay oportunidades de mejora en el manejo de esas clases.

**Ejercicio 5**

Para mejorar las métricas del modelo, decidimos optimizar los hiperparámetros del modelo de Gradient Boosting Tree (GBT) utilizando XGBoost. Elegimos este modelo porque durante el análisis exploratorio de datos no se encontraron muchas relaciones lineales, lo que sugiere que un modelo no lineal podría ser más adecuado. Además, los GBT son efectivos al manejar una combinación de datos discretos, booleanos y continuos. Después de la optimización de los hiperparámetros, los resultados del modelo fueron los siguientes:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.96 | 0.96 | 0.96 | 100 |
| 1 | 0.90 | 0.92 | 0.91 | 100 |
| 2 | 0.87 | 0.83 | 0.85 | 100 |
| 3 | 0.89 | 0.92 | 0.91 | 100 |
| accuracy | | | 0.91 | 400 |
| macro avg | 0.91 | 0.91 | 0.91 | 400 |
| weighted avg | 0.91 | 0.91 | 0.91 | 400 |




Los resultados mostraron una mejora significativa en comparación con el modelo de árbol de decisión. La precisión general aumentó al 91%, y las métricas para todas las clases también mostraron mejoras, indicando que la optimización de hiperparámetros tuvo un impacto positivo en el rendimiento del modelo.

**Conclusiones**

- Aprendimos que el EDA es fundamental para comprender la estructura de los datos, identificar valores nulos y explorar la distribución de la variable de respuesta.

- Al implementar un árbol de decisión como modelo base, pudimos establecer un punto de referencia para evaluar el rendimiento de modelos más complejos.

- La calibración del modelo a través de la optimización de hiperparámetros en XGBoost nos permitió mejorar las métricas de rendimiento, destacando la relevancia de este proceso en la creación de modelos efectivos.

- La elección de XGBoost fue adecuada debido a la mezcla de datos discretos, booleanos y continuos, lo que demostró que los modelos no lineales pueden ser más efectivos en ciertos conjuntos de datos.

- La optimización resultó en una mejora significativa en la precisión general del modelo, pasando del 80% con el árbol de decisión al 91% con XGBoost, lo que resalta la importancia de ajustar los modelos para obtener mejores resultados.