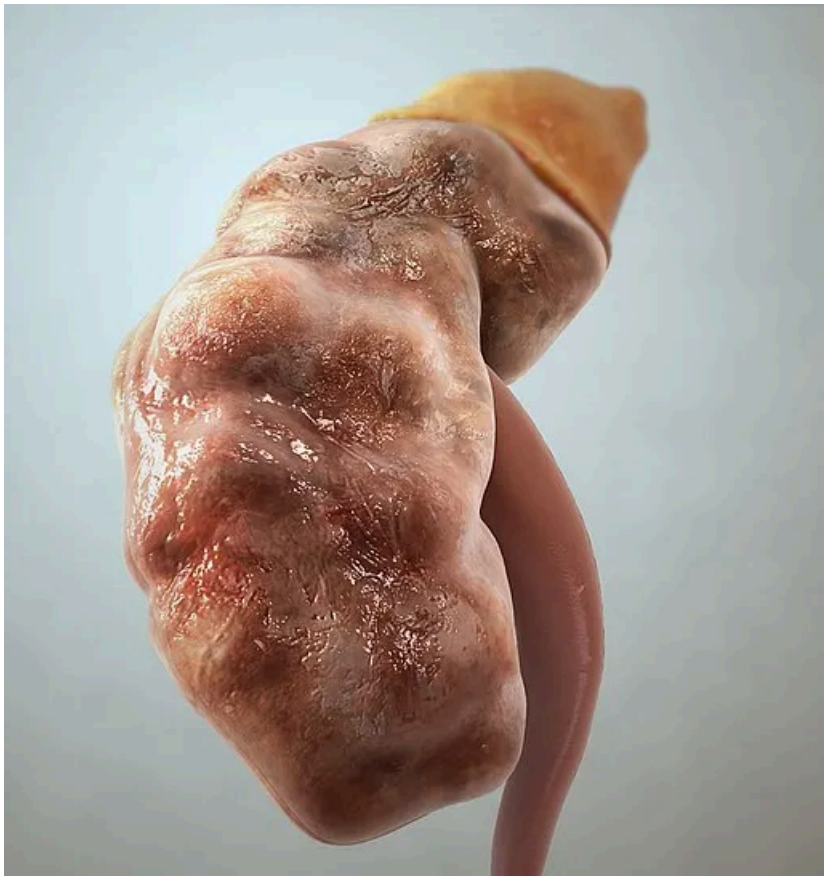
Chronic Kidney Disease, aprendizaje de modelos de decisión para una tarea de clasificación



Índice

In [1]:
```
# TODOS LOS IMPORTS NECESARIOS PARA EL PROYECTO

import tensorflow as tf
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
import arff
import joblib

# Modelos
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier

# Preprocesado y métricas
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler, MinMaxScaler
from sklearn.impute import SimpleImputer
from sklearn.decomposition import PCA
from sklearn.feature_selection import RFE, SelectFromModel
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV, cross_val_score,
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, c

# Para redes neuronales en Keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from scikeras.wrappers import KerasClassifier
from tensorflow.keras.optimizers import Adam

# Para manejo de desequilibrio de clases
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler

# Modelos avanzados
import xgboost as xgb
```

1. Introducción y objetivos

El objetivo propuesto para esta práctica final de la asignatura es aplicar todo lo aprendido durante este curso para lograr crear modelos de decisión para una tarea de clasificación. En este caso como me gusta el ámbito de la salud y la medicina he decidido elegir el dataset Chronic Kidney Disease, que refleja un conjunto de datos sobre la Enfermedad Renal Crónica (CKD) en población india. Para ello se basa en diferentes tipos de datos médicos de diferente índole, puediendo ser estos de tipo numérico como la presión de la sangre medida en mm/Hg o de tipo

nominal como el azúcar (0,1,2,3,4,5). Además, cabe destacar que el dataset cuenta con numerosos campos vacíos y errores de tipado en las columnas con los que deberemos lidiar para un correcto uso de este dataset.

Para poder lograr con éxito el objetivo de crear modelos de decisión, debemos intentar conseguir el mayor porcentaje de éxito posible a la hora de dictaminar un resultado, por lo que para ello, trataremos el dataset y conseguiremos cuatro muestras diferentes del mismo, aplicando diferentes técnicas para así tener una mayor posibilidad de éxito. Además contaremos con diferentes modelos creados con distintos métodos para así tener un estudio más completo y ver cuales funcionan mejor a la hora de evaluarlos. Con todo esto esperamos crear un estudio lo más completo posible acerca de la Enfermedad Renal Crónica (CKD) y aprender lo máximo posible mientras se realiza.

2. Análisis inicial y preparación de los datos

En primer lugar debemos darnos cuenta de que el dataset no es un archivo .csv sino que es un archivo .arff, un tipo de archivo no tan común que debe ser tratado de una forma un poco diferente a lo habitual. Para poder leer el archivo .arff debemos tener instalada la librería liac-arff. Una vez instalada, al intentar abrir el archivo nos damos cuenta de que no es posible debido a varios errores con los que cuenta.

En algunas líneas existen más columnas de las que en un principio se declaran ya que algunas de estas lineas finalizan con una , extra al final.

BadDataFormat: Bad @DATA instance format in line 215:
26,70,1.015,0,4,?,normal,notpresent,notpresent,250,20,1.1,?,?,15.6,52,6900,6.0,no,yes,no,good,no,no,ckd,

Por lo que debemos crear un script en Python para analizar y limpiar el dataset actual.

Una vez que tenemos ya el dataset corregido nos toca ver si todo está en orden y si podemos visualizar de forma corretca los datos en el fichero del notebook. Vemos que cuenta con multitud de valores vacios y faltantes por lo que reemplazaremos dichos valores faltantes representados por ? con NaN y realizaremos una exploración inicial. El dataset, despues de estos cambios se encuentra a la orden del día y podemos continuar con el proceso, que será proponer 4 datasets diferentes con distintos métodos de preprocesamiento.

In [2]:
```python
arff_file = r"C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Chronic_Kidney_Di
corrected_arff_file = r"C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Chronic_

with open(arff_file, 'r') as f:
    lines = f.readlines()

# Identificar la sección @DATA
data_start = False
corrected_lines = []
for line in lines:
    if line.strip().lower().startswith('@data'):
        data_start = True
        corrected_lines.append(line)
        continue
    if data_start:
        if line.strip() == '' or line.strip().startswith('%'):
            continue
        # Eliminar comas finales
        if line.strip().endswith(','):
            line = line.rstrip(',\n') + '\n'
        # Verificar el número de campos
        fields = line.strip().split(',')
        if len(fields) != 25:
            print(f"Advertencia: Línea con {len(fields)} campos: {line.strip()}")
            continue
        corrected_lines.append(line)
    else:
        corrected_lines.append(line)

with open(corrected_arff_file, 'w') as f:
    f.writelines(corrected_lines)

print(f"Archivo corregido guardado en: {corrected_arff_file}")
```

```
Advertencia: Línea con 26 campos: 75,70,1.020,0,0,normal,normal,notpresent,notpresent,107,48,0.8,144,3.5,
13.6,46,10300,4.8,no,,no,no,good,no,no,notckd
Archivo corregido guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\C
hronic_Kidney_Disease\chronic_kidney_disease_full_clean.arff
```

In [3]:
```python
with open(corrected_arff_file, 'r') as f:
    dataset = arff.load(f)

# Convertir a DataFrame
df_original = pd.DataFrame(dataset['data'], columns=[attr[0] for attr in dataset['attributes']])

# Reemplazar '?' por NaN
df_original.replace('?', np.nan, inplace=True)

# Identificar columnas numéricas y categóricas
numeric_cols = ['age', 'bp', 'bgr', 'bu', 'sc', 'sod', 'pot',
                'hemo', 'pcv', 'wbcc', 'rbcc']
categorical_cols = ['sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba',
                    'htn', 'dm', 'cad', 'appet', 'pe', 'ane']
```

3. Preprocesado y transformación

3.1 Dataset 1: Eliminar filas con valores ausentes

- Eliminar todas las filas que contengan al menos un valor faltante.
- Label Encoding para variables binarias y One-Hot Encoding para variables con más de dos categorías.
- Estandarización de variables numéricas.

In [4]:
```python
# ---------------------- Versión 1 ----------------------
# Eliminar filas con valores faltantes
df_v1 = df_original.copy()
df_v1.dropna(inplace=True)
print(f"\n[Versión 1] Instancias después de eliminar filas con valores faltantes: {df_v1.shape[0]}")

# Convertir columnas numéricas a float
for col in numeric_cols:
    df_v1[col] = pd.to_numeric(df_v1[col], errors='coerce')

# Codificación de variables categóricas
binary_cols_v1 = ['rbc', 'pc', 'pcc', 'ba', 'htn', 'dm', 'cad', 'pe', 'ane', 'class']
le_v1 = LabelEncoder()
for col in binary_cols_v1:
    df_v1[col] = le_v1.fit_transform(df_v1[col])

# One-Hot Encoding para variables con más de dos categorías
df_v1 = pd.get_dummies(df_v1, columns=['sg', 'al', 'su', 'appet'], drop_first=True)

# Escalado de variables numéricas
scaler_v1 = StandardScaler()
df_v1[numeric_cols] = scaler_v1.fit_transform(df_v1[numeric_cols])


df_v1.to_csv('ckd_dataset1.csv', index=False)
print("**Dataset 1** guardado como 'ckd_dataset1.csv'")
```

```
[Versión 1] Instancias después de eliminar filas con valores faltantes: 157
**Dataset 1** guardado como 'ckd_dataset1.csv'
```

3.2 Dataset 2: Imputar valores ausentes (Media/Mediana para numéricos, Moda para categóricos)

- Imputar valores faltantes con la mediana para variables numéricas y la moda para variables categóricas.
- One-Hot Encoding para todas las variables categóricas.
- Estandarización de variables numéricas.

In [5]:
```python
# ---------------------- Versión 2 ----------------------
# Imputación de valores faltantes
df_v2 = df_original.copy()
```

```
# Convertir columnas numéricas a float
for col in numeric_cols:
    df_v2[col] = pd.to_numeric(df_v2[col], errors='coerce')

# Imputar numéricos con la mediana
imputer_num_v2 = SimpleImputer(strategy='median')
df_v2[numeric_cols] = imputer_num_v2.fit_transform(df_v2[numeric_cols])

# Imputar categóricos con la moda
imputer_cat_v2 = SimpleImputer(strategy='most_frequent')
df_v2[categorical_cols] = imputer_cat_v2.fit_transform(df_v2[categorical_cols])

# One-Hot Encoding para todas las variables categóricas
df_v2 = pd.get_dummies(df_v2, columns=['sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba',
                                       'htn', 'dm', 'cad', 'appet', 'pe', 'ane'], drop_first=True)

# Codificación de la clase
df_v2['class'] = df_v2['class'].map({'notckd': 0, 'ckd': 1})

# Escalado de variables numéricas
scaler_v2 = StandardScaler()
df_v2[numeric_cols] = scaler_v2.fit_transform(df_v2[numeric_cols])

df_v2.to_csv('ckd_dataset2.csv', index=False)
print("**Dataset 2** guardado como 'ckd_dataset2.csv'")
```

```
**Dataset 2** guardado como 'ckd_dataset2.csv'
```

3.3 Dataset 3: Eliminar características con muchos valores faltantes + Imputación simple + Label encoding + Escalado

- Eliminación de variables removiendo columnas con una alta cantidad de valores faltantes (más del 30%), además de eliminar filas con valores vacíos.
- Codificación con Label Encoding para variables categóricas binarias.
- Escalado con StandardScaler para variables numéricas.

In [6]:
```
# ---------------------- Versión 3 ----------------------
# Eliminar características con >30% de valores faltantes
df_v3 = df_original.copy()
missing_percent_v3 = df_v3.isnull().mean() * 100
threshold_v3 = 30
cols_to_drop_v3 = missing_percent_v3[missing_percent_v3 > threshold_v3].index.tolist()
print(f"\n[Versión 3] Columnas a eliminar (>{threshold_v3}% faltantes): {cols_to_drop_v3}")
df_v3.drop(columns=cols_to_drop_v3, inplace=True)

# Identificar columnas numéricas y categóricas después de la eliminación
numeric_cols_v3 = ['age', 'bp', 'bgr', 'bu', 'sc', 'sod', 'pot',
                   'hemo', 'pcv', 'wbcc', 'rbcc']
categorical_cols_v3 = ['sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba',
                       'htn', 'dm', 'cad', 'appet', 'pe', 'ane']

# Actualizar las listas de columnas para incluir solo las que permanecen en el DataFrame
numeric_cols_v3 = [col for col in numeric_cols_v3 if col in df_v3.columns]
categorical_cols_v3 = [col for col in categorical_cols_v3 if col in df_v3.columns]

print(f"\n[Versión 3] Columnas numéricas después de eliminación: {numeric_cols_v3}")
print(f"[Versión 3] Columnas categóricas después de eliminación: {categorical_cols_v3}")

# Convertir columnas numéricas a float
for col in numeric_cols_v3:
    df_v3[col] = pd.to_numeric(df_v3[col], errors='coerce')

# Verificar valores faltantes después de la conversión
print("\n[Versión 3] Valores faltantes después de convertir a numérico:")
print(df_v3[numeric_cols_v3].isnull().sum())

# Imputar numéricos con la mediana
imputer_num_v3 = SimpleImputer(strategy='median')
df_v3[numeric_cols_v3] = imputer_num_v3.fit_transform(df_v3[numeric_cols_v3])

# Verificar valores faltantes después de la imputación numérica
```

```
print("\n[Versión 3] Valores faltantes después de imputar numéricos:")
print(df_v3[numeric_cols_v3].isnull().sum())

# Imputar categóricos con la moda
imputer_cat_v3 = SimpleImputer(strategy='most_frequent')
df_v3[categorical_cols_v3] = imputer_cat_v3.fit_transform(df_v3[categorical_cols_v3])

# Verificar valores faltantes después de la imputación categórica
print("\n[Versión 3] Valores faltantes después de imputar categóricos:")
print(df_v3[categorical_cols_v3].isnull().sum())

# Codificación de variables categóricas binarias y clase
binary_cols_v3 = ['rbc', 'pc', 'pcc', 'ba', 'htn', 'dm', 'cad', 'pe', 'ane', 'class']
# Verificar cuáles de estos están presentes en df_v3
binary_cols_v3 = [col for col in binary_cols_v3 if col in df_v3.columns]
le_v3 = LabelEncoder()
for col in binary_cols_v3:
    df_v3[col] = le_v3.fit_transform(df_v3[col])
    print(f"[Versión 3] '{col}' codificado correctamente.")

if 'appet' in df_v3.columns:

    le_appet = LabelEncoder()
    df_v3['appet'] = le_appet.fit_transform(df_v3['appet'])
    print("[Versión 3] 'appet' codificado correctamente.")

# Verificar si quedan NaNs después de la codificación
print("\n[Versión 3] Valores faltantes después de codificación:")
print(df_v3.isnull().sum())

if df_v3.isnull().sum().sum() > 0:
    print("[Versión 3] Existen NaNs después de la codificación. Eliminando filas con NaNs.")
    df_v3.dropna(inplace=True)
    print("[Versión 3] Filas con NaNs eliminadas.")
else:
    print("[Versión 3] No hay NaNs después de la codificación.")

# Escalado de variables numéricas
scaler_v3 = StandardScaler()
df_v3[numeric_cols_v3] = scaler_v3.fit_transform(df_v3[numeric_cols_v3])

print("\n[Versión 3] Valores faltantes después del escalado:")
print(df_v3.isnull().sum())

# Si hay NaNs restantes, eliminarlos
if df_v3.isnull().sum().sum() > 0:
    print("[Versión 3] Existen NaNs después del escalado. Eliminando filas con NaNs.")
    df_v3.dropna(inplace=True)
    print("[Versión 3] Filas con NaNs eliminadas.")
else:
    print("[Versión 3] No hay NaNs después del escalado.")

total_nans = df_v3.isnull().sum().sum()
if total_nans == 0:
    print("\n[Versión 3] Verificación final: No hay valores faltantes en el dataset.")
else:
    print(f"\n[Versión 3] Verificación final: Existen {total_nans} valores faltantes en el dataset.")

# Guardar Dataset 3
df_v3.to_csv('ckd_dataset3.csv', index=False)
print("**Dataset 3** guardado como 'ckd_dataset3.csv'")
```

```
[Versión 3] Columnas a eliminar (>30% faltantes): ['rbc', 'rbcc']

[Versión 3] Columnas numéricas después de eliminación: ['age', 'bp', 'bgr', 'bu', 'sc', 'sod', 'pot', 'he
mo', 'pcv', 'wbcc']
[Versión 3] Columnas categóricas después de eliminación: ['sg', 'al', 'su', 'pc', 'pcc', 'ba', 'htn', 'd
m', 'cad', 'appet', 'pe', 'ane']

[Versión 3] Valores faltantes después de convertir a numérico:
age       9
bp       12
bgr      44
bu       19
sc       17
sod      87
pot      88
hemo     52
pcv      71
wbcc    106
dtype: int64

[Versión 3] Valores faltantes después de imputar numéricos:
age     0
bp      0
bgr     0
bu      0
sc      0
sod     0
pot     0
hemo    0
pcv     0
wbcc    0
dtype: int64

[Versión 3] Valores faltantes después de imputar categóricos:
sg       47
al       46
su       49
pc       65
pcc       4
ba        4
htn       2
dm        2
cad       2
appet     1
pe        1
ane       1
dtype: int64
[Versión 3] 'pc' codificado correctamente.
[Versión 3] 'pcc' codificado correctamente.
[Versión 3] 'ba' codificado correctamente.
[Versión 3] 'htn' codificado correctamente.
[Versión 3] 'dm' codificado correctamente.
[Versión 3] 'cad' codificado correctamente.
[Versión 3] 'pe' codificado correctamente.
[Versión 3] 'ane' codificado correctamente.
[Versión 3] 'class' codificado correctamente.
[Versión 3] 'appet' codificado correctamente.

[Versión 3] Valores faltantes después de codificación:
age       0
bp        0
sg       47
al       46
su       49
pc        0
pcc       0
ba        0
bgr       0
bu        0
sc        0
sod       0
pot       0
hemo      0
pcv       0
wbcc      0
```

```
htn       0
dm        0
cad       0
appet     0
pe        0
ane       0
class     0
dtype: int64
[Versión 3] Existen NaNs después de la codificación. Eliminando filas con NaNs.
[Versión 3] Filas con NaNs eliminadas.

[Versión 3] Valores faltantes después del escalado:
age      0
bp       0
sg       0
al       0
su       0
pc       0
pcc      0
ba       0
bgr      0
bu       0
sc       0
sod      0
pot      0
hemo     0
pcv      0
wbcc     0
htn      0
dm       0
cad      0
appet    0
pe       0
ane      0
class    0
dtype: int64
[Versión 3] No hay NaNs después del escalado.

[Versión 3] Verificación final: No hay valores faltantes en el dataset.
**Dataset 3** guardado como 'ckd_dataset3.csv'
```

3.4 Dataset 4: Técnicas de reducción de dimensionalidad y balanceo de clases

- Imputación con la mediana para numéricos y la moda para categóricos.
- One-Hot Encoding para todas las variables categóricas.
- Estandarización de variables numéricas.
- PCA para reducir la dimensionalidad manteniendo la mayor varianza posible.
- Balanceo de clases, Over-sampling con SMOTE.

In [7]:

```python
# ---------------------- Versión 4 ----------------------
# Imputación de valores faltantes
df_v4 = df_original.copy()

# Convertir columnas numéricas a float
for col in numeric_cols:
    df_v4[col] = pd.to_numeric(df_v4[col], errors='coerce')

# Imputar numéricos con la mediana
imputer_num_v4 = SimpleImputer(strategy='median')
df_v4[numeric_cols] = imputer_num_v4.fit_transform(df_v4[numeric_cols])

# Imputar categóricos con la moda
imputer_cat_v4 = SimpleImputer(strategy='most_frequent')
df_v4[categorical_cols] = imputer_cat_v4.fit_transform(df_v4[categorical_cols])

# One-Hot Encoding para todas las variables categóricas
df_v4 = pd.get_dummies(df_v4, columns=['sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba',
                                       'htn', 'dm', 'cad', 'appet', 'pe', 'ane'], drop_first=True)

# Codificación de la clase
df_v4['class'] = df_v4['class'].map({'notckd': 0, 'ckd': 1})
```

```python
# Separar características y variable objetivo
X_v4 = df_v4.drop('class', axis=1)
y_v4 = df_v4['class']

# Escalado de variables numéricas
scaler_v4 = StandardScaler()
X_v4[numeric_cols] = scaler_v4.fit_transform(X_v4[numeric_cols])

# Reducción de Dimensionalidad con PCA
pca = PCA(n_components=0.95, random_state=42)  # Mantener el 95% de la varianza
X_pca_v4 = pca.fit_transform(X_v4)

print(f"\n[Versión 4] Número de componentes después de PCA: {X_pca_v4.shape[1]}")

# Balanceo de Clases con SMOTE
smote = SMOTE(random_state=42)
X_balanced_v4, y_balanced_v4 = smote.fit_resample(X_pca_v4, y_v4)

print("\n[Versión 4] Distribución de clases después de SMOTE:")
print(pd.Series(y_balanced_v4).value_counts())

# Crear un DataFrame con las nuevas características y la variable objetivo
df_v4_final = pd.DataFrame(X_balanced_v4, columns=[f'PC{i+1}' for i in range(X_pca_v4.shape[1])])
df_v4_final['class'] = y_balanced_v4


df_v4_final.to_csv('ckd_dataset4.csv', index=False)
print("**Dataset 4** guardado como 'ckd_dataset4.csv'")
```

```
[Versión 4] Número de componentes después de PCA: 20

[Versión 4] Distribución de clases después de SMOTE:
class
1    250
0    250
Name: count, dtype: int64
**Dataset 4** guardado como 'ckd_dataset4.csv'
```

3.5 Resumen de las cuatro versiones del dataset

Versión	Manejo de valores faltantes	Codificación categórica	Reducción de dimensionalidad	Balanceo de clases	Archivo guardado
1	Eliminar filas con cualquier valor faltante	Label + One-Hot Encoding	No	No	cdk_dataset1.csv
2	Imputar con mediana/moda	One-Hot Encoding	No	No	cdk_dataset2.csv
3	Eliminar columnas con >30% faltantes, imputar	Label Encoding	No	No	cdk_dataset3.csv
4	Imputar con mediana/moda	One-Hot Encoding	PCA	SMOTE	cdk_dataset4.csv

4. Construcción y entrenamiento de modelos

Una vez que hemos limpiado el dataset y a partir de él hemos creado diferentes versiones, es hora de usarlos para crear diferentes modelos de IA. La construcción y entrenamiento de modelos es una etapa crítica en el desarrollo de modelos, ya que determina la capacidad de los algoritmos para aprender patrones significativos a partir de los datos preprocesados y realizar predicciones precisas. En este proyecto, hemos optado por una variedad de modelos de clasificación que abarcan desde algoritmos sencillos hasta modelos avanzados de ensemble y redes neuronales, con el objetivo de evaluar su desempeño en la clasificación de la enfermedad renal crónica (CKD) a través de diferentes versiones de datasets.

La selección de estos modelos se ha basado en su robustez, interpretabilidad, capacidad de generalización y adecuación al tipo de datos y problema en cuestión. Además, se ha implementado una estructura organizada mediante funciones especializadas que facilitan la carga de datos, la creación de directorios, el almacenamiento de modelos, y la generación de reportes y visualizaciones. Este enfoque modular no solo optimiza el proceso de entrenamiento, sino que también garantiza la reproducibilidad y facilidad de gestión de los resultados obtenidos.

A continuación, se detalla el proceso seguido para seleccionar, configurar, entrenar, optimizar y evaluar los modelos de clasificación, así como la organización estructurada de los datos y resultados para cada versión del dataset.

4.1 Selección de modelos

Para abordar el problema de clasificación de CKD, hemos seleccionado una gama diversa de modelos de machine learning, cada uno con características únicas que los hacen aptos para diferentes tipos de datos y escenarios. La selección se realizó considerando factores como la interpretabilidad, la capacidad de manejar relaciones no lineales, y la eficiencia computacional. Los modelos seleccionados son:

- **k-Nearest Neighbors (k-NN):**

Algoritmo de clasificación basado en la proximidad, donde una instancia se clasifica según la mayoría de sus vecinos más cercanos.

Simple de implementar, no paramétrico, y eficaz en datasets bien estructurados.

Sensible a la escala de los datos y al ruido, y puede ser computacionalmente costoso en datasets grandes.

- **Decision Tree:**

Modelo interpretativo que divide el espacio de características en regiones homogéneas mediante decisiones basadas en umbrales.

Fácil de interpretar, manejar variables tanto numéricas como categóricas, y capturar relaciones no lineales.

Propenso al overfitting, especialmente con árboles profundos.

- **Random Forest:**

Conjunto de árboles de decisión entrenados con diferentes subconjuntos de datos y características, combinando sus predicciones para mejorar la precisión.

Alta precisión, robusto frente al overfitting, y capaz de manejar grandes volúmenes de datos con muchas características.

Menos interpretativo que un único árbol de decisión, y puede ser computacionalmente intensivo.

- **Gradient Boosting:**

Modelo de ensemble que construye árboles de decisión secuencialmente, cada uno enfocándose en corregir los errores del anterior.

Alta precisión, capacidad para manejar relaciones complejas y no lineales, y flexibilidad en la optimización de funciones de pérdida.

Sensible a la selección de hiperparámetros y propenso al overfitting si no se regulariza adecuadamente.

- **Logistic Regression:**

Modelo lineal utilizado para clasificación binaria que estima la probabilidad de pertenencia a una clase.

Simple de implementar, eficiente en datasets linealmente separables, y fácil de interpretar.

Limitado en capturar relaciones no lineales, y sensible a la multicolinealidad.

- **Support Vector Machine (SVM):**

Modelo que encuentra el hiperplano óptimo que separa las clases con el mayor margen posible.

Eficaz en espacios de alta dimensionalidad, y flexible mediante el uso de diferentes kernels.

Computacionalmente intensivo en datasets grandes, y menos interpretativo.

- **MLP Classifier (Multilayer Perceptron):**

Red neuronal multicapa capaz de capturar relaciones no lineales complejas en los datos.

Alta capacidad de aprendizaje, flexible en la arquitectura, y eficaz en problemas de clasificación complejos.

Requiere un mayor tiempo de entrenamiento, propenso al overfitting, y menos interpretativo.

4.2. Configuración de hiperparámetros

Para cada modelo seleccionado, se ha definido un conjunto de hiperparámetros que serán ajustados mediante técnicas de búsqueda para optimizar su rendimiento. La configuración de hiperparámetros para cada modelo es la siguiente:

- **k-NN:**
  - **n_neighbors:** Número de vecinos a considerar (1 a 30).
  - **weights:** Método de ponderación ('uniform', 'distance').
  - **metric:** Métrica de distancia utilizada ('euclidean', 'manhattan', 'minkowski').
- **Decision Tree:**
  - **criterion:** Función de calidad de la división ('gini', 'entropy').
  - **max_depth:** Profundidad máxima del árbol (None, 5, 10, 20, 30).
  - **min_samples_split:** Número mínimo de muestras para dividir un nodo (2, 5, 10).
  - **min_samples_leaf:** Número mínimo de muestras en una hoja (1, 2, 4).
  - **ccp_alpha:** Parámetro de poda de complejidad (0.0, 0.01, 0.02, 0.05).
- **Random Forest:**
  - **n_estimators:** Número de árboles en el bosque (100, 200, 300).
  - **max_depth:** Profundidad máxima de cada árbol (None, 10, 20, 30).
  - **min_samples_split:** Número mínimo de muestras para dividir un nodo (2, 5, 10).
  - **min_samples_leaf:** Número mínimo de muestras en una hoja (1, 2, 4).
  - **bootstrap:** Método de muestreo ('True', 'False').
- **Gradient Boosting:**
  - **n_estimators:** Número de etapas de boosting (100, 200, 300).
  - **learning_rate:** Tasa de aprendizaje (0.01, 0.05, 0.1).
  - **max_depth:** Profundidad máxima de los árboles individuales (3, 5, 7).
  - **min_samples_split:** Número mínimo de muestras para dividir un nodo (2, 5, 10).
  - **min_samples_leaf:** Número mínimo de muestras en una hoja (1, 2, 4).
- **Logistic Regression:**
  - **C:** Parámetro de regularización (0.01, 0.1, 1, 10, 100).
  - **penalty:** Tipo de penalización ('l1', 'l2').
  - **solver:** Algoritmo de optimización ('liblinear').
- **SVM:**
  - **C:** Parámetro de penalización (0.1, 1, 10, 100).
  - **gamma:** Coeficiente del kernel ('scale', 'auto').
  - **kernel:** Tipo de kernel utilizado ('linear', 'rbf', 'poly').
- **Random Forest (Avanzado):**
  - **n_estimators:** Número de árboles en el bosque (100, 200, 300, 400, 500).
  - **max_features:** Número de características a considerar en cada división ('sqrt', 'log2').
  - **max_depth:** Profundidad máxima de cada árbol (None, 10, 20, 30).
  - **min_samples_split:** Número mínimo de muestras para dividir un nodo (2, 5, 10).
  - **min_samples_leaf:** Número mínimo de muestras en una hoja (1, 2, 4).
- **MLP Classifier:**
  - **hidden_layer_sizes:** Tamaño de las capas ocultas ((50,), (100,), (100, 50), (100, 100)).
  - **activation:** Función de activación ('relu', 'tanh', 'logistic').
  - **solver:** Método de optimización ('adam', 'sgd').
  - **alpha:** Parámetro de regularización (0.0001, 0.001, 0.01).

- **learning_rate**: Estrategia de tasa de aprendizaje ('constant', 'adaptive').

In [8]:
```
models_params = {
    'k-NN': {
        'model': KNeighborsClassifier(),
        'params': {
            'n_neighbors': list(range(1,31)),
            'weights': ['uniform', 'distance'],
            'metric': ['euclidean', 'manhattan', 'minkowski']
        }
    },
    'Decision Tree': {
        'model': DecisionTreeClassifier(random_state=42),
        'params': {
            'criterion': ['gini', 'entropy'],
            'max_depth': [None, 5, 10, 20, 30],
            'min_samples_split': [2, 5, 10],
            'min_samples_leaf': [1, 2, 4],
            'ccp_alpha': [0.0, 0.01, 0.02, 0.05]
        }
    },
    'Random Forest': {
        'model': RandomForestClassifier(random_state=42),
        'params': {
            'n_estimators': [100, 200, 300],
            'max_depth': [None, 10, 20, 30],
            'min_samples_split': [2, 5, 10],
            'min_samples_leaf': [1, 2, 4],
            'bootstrap': [True, False]
        }
    },
    'Gradient Boosting': {
        'model': GradientBoostingClassifier(random_state=42),
        'params': {
            'n_estimators': [100, 200, 300],
            'learning_rate': [0.01, 0.05, 0.1],
            'max_depth': [3, 5, 7],
            'min_samples_split': [2, 5, 10],
            'min_samples_leaf': [1, 2, 4]
        }
    },
    'Logistic Regression': {
        'model': LogisticRegression(random_state=42, max_iter=1000),
        'params': {
            'C': [0.01, 0.1, 1, 10, 100],
            'penalty': ['l1', 'l2'],
            'solver': ['liblinear']
        }
    },
    'SVM': {
        'model': SVC(probability=True, random_state=42),
        'params': {
            'C': [0.1, 1, 10, 100],
            'gamma': ['scale', 'auto'],
            'kernel': ['linear', 'rbf', 'poly']
        }
    },
    'Random Forest (Avanzado)': {
        'model': RandomForestClassifier(random_state=42),
        'params': {
            'n_estimators': [100, 200, 300, 400, 500],
            'max_features': ['sqrt', 'log2'],
            'max_depth': [None, 10, 20, 30],
            'min_samples_split': [2, 5, 10],
            'min_samples_leaf': [1, 2, 4]
        }
    },
    'MLP Classifier': {
        'model': MLPClassifier(random_state=42, max_iter=1000),
        'params': {
            'hidden_layer_sizes': [(50,), (100,), (100,50), (100,100)],
            'activation': ['relu', 'tanh', 'logistic'],
            'solver': ['adam', 'sgd'],
            'alpha': [0.0001, 0.001, 0.01],
```

```
            'learning_rate': ['constant', 'adaptive']
        }
    }
}
```

4.3. Entrenamiento de modelos

El entrenamiento de los modelos se ha realizado de manera estructurada, utilizando funciones especializadas que facilitan la gestión del proceso y la organización de los resultados. A continuación, se describen las funciones implementadas y cómo se integran en el flujo de trabajo.

4.3.1 Funciones implementadas

- load_dataset(file_path, target_column='class'):

Esta función facilita la carga de datasets, separando automáticamente las características de la variable objetivo, lo que simplifica el flujo de trabajo posterior al entrenamiento de modelos.

In [9]:
```
def load_dataset(file_path, target_column='class'):

    df = pd.read_csv(file_path)
    X = df.drop(target_column, axis=1)
    y = df[target_column]
    return X, y
```

- create_directory(path):

Garantiza la creación de directorios necesarios para almacenar modelos, gráficos y reportes, evitando errores si las carpetas ya existen.

In [10]:
```
def create_directory(path):

    os.makedirs(path, exist_ok=True)
```

- save_model(model, filepath):

Permite el almacenamiento eficiente de modelos entrenados en formato .pkl, facilitando su reutilización y despliegue en el futuro.

In [11]:
```
def save_model(model, filepath):

    joblib.dump(model, filepath)
    print(f"Modelo guardado en: {filepath}")
```

- save_report(report, filepath):

Guarda el reporte de clasificación en un archivo de texto, proporcionando una documentación detallada del rendimiento del modelo. Almacena los reportes en las carpetas de Reports, organizados por dataset y modelo.

In [12]:
```
def save_report(report, filepath):

    with open(filepath, 'w') as f:
        f.write(report)
    print(f"Reporte guardado en: {filepath}")
```

- train_evaluate_model(model, X_train, X_test, y_train, y_test, model_name, dataset_name, output_dirs):

Entrena el modelo, realiza predicciones, evalúa el rendimiento mediante diversas métricas, y guarda las visualizaciones de las matrices de confusión y curvas de rendimiento. Centraliza el proceso de entrenamiento y evaluación, asegurando consistencia y organización en los resultados.

In [13]:
```
def train_evaluate_model(model, X_train, X_test, y_train, y_test, model_name, dataset_name, output_dirs)
```

```
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    # Intentar obtener probabilidades o scores
    if hasattr(model, "predict_proba"):
        y_proba = model.predict_proba(X_test)[:,1]
    elif hasattr(model, "decision_function"):
        y_proba = model.decision_function(X_test)
    else:
        y_proba = None

    # Evaluar el modelo
    accuracy = accuracy_score(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)
    report = classification_report(y_test, y_pred, output_dict=True)
    report_text = classification_report(y_test, y_pred)

    print(f"\n--- {model_name} en {dataset_name} ---")
    print(f"Precisión (Accuracy): {accuracy:.4f}")
    print("Reporte de Clasificación:")
    print(report_text)

    # Guardar el reporte de clasificación
    report_filepath = os.path.join(output_dirs['Reports'], f"{model_name}_{dataset_name}_Report.txt")
    save_report(report_text, report_filepath)

    # Visualizar y guardar la matriz de confusión
    plt.figure(figsize=(6,5))
    sns.set(style="whitegrid")
    ax = sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
                     xticklabels=np.unique(y_test),
                     yticklabels=np.unique(y_test))

    ax.set_xlabel('Predicción', fontsize=12)
    ax.set_ylabel('Realidad', fontsize=12)
    ax.set_title(f'Matriz de Confusión - {model_name} en {dataset_name}', fontsize=14, pad=20)
    plt.yticks(rotation=0)


    cm_filepath = os.path.join(output_dirs['Plots'], f"{model_name}_{dataset_name}_Confusion_Matrix.png"
    plt.savefig(cm_filepath)
    plt.close()
    print(f"Matriz de Confusión guardada en: {cm_filepath}")

    # Visualizar y guardar Curvas ROC y Precision-Recall si están disponibles
    if y_proba is not None and len(np.unique(y_test)) > 1:
        # Curva ROC
        fpr, tpr, _ = roc_curve(y_test, y_proba)
        roc_auc = auc(fpr, tpr)

        plt.figure(figsize=(6,5))
        plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
        plt.plot([0,1], [0,1], color='navy', lw=2, linestyle='--')
        plt.xlim([0.0,1.0])
        plt.ylim([0.0,1.05])
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.title(f'ROC Curve - {model_name} en {dataset_name}')
        plt.legend(loc="lower right")

        roc_filepath = os.path.join(output_dirs['Plots'], f"{model_name}_{dataset_name}_ROC_Curve.png")
        plt.savefig(roc_filepath)
        plt.close()
        print(f"Curva ROC guardada en: {roc_filepath}")

        precision_vals, recall_vals, _ = precision_recall_curve(y_test, y_proba)
        pr_auc = auc(recall_vals, precision_vals)

        plt.figure(figsize=(6,5))
        plt.plot(recall_vals, precision_vals, color='blue', lw=2, label=f'Precision-Recall curve (AUC =
        plt.xlabel('Recall')
        plt.ylabel('Precision')
        plt.title(f'Precision-Recall Curve - {model_name} en {dataset_name}')
        plt.legend(loc="upper right")
```

```
        pr_filepath = os.path.join(output_dirs['Plots'], f"{model_name}_{dataset_name}_Precision_Recall_
        plt.savefig(pr_filepath)
        plt.close()
        print(f"Curva Precision-Recall guardada en: {pr_filepath}")

    return y_pred, y_proba, report, accuracy, cm
```

- visualize_results(results_dict):

Crea visualizaciones gráficas de las métricas almacenadas en el diccionario de resultados, facilitando la comparación entre modelos y datasets. Genera gráficos de barras para cada métrica, almacenándolos en una carpeta Summary para una visión global del desempeño.

In [14]:
```
def visualize_results(results_dict):

    results_df = pd.DataFrame(results_dict).T

    print("\n=== Resultados de Modelos ===")
    print(results_df)

    summary_dir = os.path.join(BASE_OUTPUT_DIR, "Summary")
    create_directory(summary_dir)

    summary_csv_path = os.path.join(summary_dir, "models_results_summary.csv")
    results_df.to_csv(summary_csv_path)
    print(f"Resumen de resultados guardado en: {summary_csv_path}")

    # Plotear las métricas
    metrics = ['Accuracy', 'Precision', 'Recall', 'F1-Score', 'AUC']

    for metric in metrics:
        plt.figure(figsize=(12,6))
        sns.barplot(x=results_df.index, y=results_df[metric])
        plt.xticks(rotation=90)
        plt.title(f'{metric} por Modelo y Dataset')
        plt.ylabel(metric)
        plt.xlabel('Modelo - Dataset')
        plt.tight_layout()

        metric_plot_path = os.path.join(summary_dir, f"{metric}_Barplot.png")
        plt.savefig(metric_plot_path)
        plt.close()
        print(f"Gráfico de {metric} guardado en: {metric_plot_path}")
```

In [15]:
```
# Definir las versiones de los datasets
datasets = {
    'Dataset_1': 'ckd_dataset1.csv',
    'Dataset_2': 'ckd_dataset2.csv',
    'Dataset_3': 'ckd_dataset3.csv',
    'Dataset_4': 'ckd_dataset4.csv'
}
BASE_OUTPUT_DIR = r'C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2'

# Crear el directorio base si no existe
create_directory(BASE_OUTPUT_DIR)

# Crear un diccionario para almacenar los resultados
results = {}
```

- record_results(model_name, dataset_name, accuracy, precision, recall, f1, auc_score):

Registra las métricas de rendimiento de cada modelo y dataset en un diccionario para análisis posterior. Almacena de manera estructurada las métricas obtenidas durante el entrenamiento.

In [16]:
```
def record_results(model_name, dataset_name, accuracy, precision, recall, f1, auc_score):

    key = f"{model_name} - {dataset_name}"
    results[key] = {
        'Accuracy': accuracy,
        'Precision': precision,
```

```
        'Recall': recall,
        'F1-Score': f1,
        'AUC': auc_score
    }
```

- hyperparameter_tuning(model, param_grid, X_train, y_train, cv=5, search_type='grid', scoring='accuracy'):

Ajusta los hiperparámetros de un modelo utilizando GridSearchCV o RandomizedSearchCV, optimizando el rendimiento mediante búsqueda exhaustiva o aleatoria. Identifica la mejor configuración de hiperparámetros para cada modelo antes del entrenamiento final.

In [17]:
```
def hyperparameter_tuning(model, param_grid, X_train, y_train, cv=5, search_type='grid', scoring='accura

    if search_type == 'grid':
        search = GridSearchCV(estimator=model, param_grid=param_grid, cv=cv, n_jobs=-1, scoring=scoring,
    elif search_type == 'random':
        search = RandomizedSearchCV(estimator=model, param_distributions=param_grid, cv=cv, n_jobs=-1, s
    else:
        raise ValueError("search_type debe ser 'grid' o 'random'")

    search.fit(X_train, y_train)

    print("Mejores Parámetros:", search.best_params_)
    print(f"Mejor {search.scoring} (CV): {search.best_score_:.4f}")

    return search.best_estimator_, search.best_params_, search.best_score_
```

4.3.2 Flujo de entrenamiento

El proceso de entrenamiento de los modelos se ha realizado iterando sobre cada dataset y cada modelo, siguiendo los siguientes pasos:

Carga del dataset:

Utilizando load_dataset, se cargan las características (X) y la variable objetivo (y) desde los archivos CSV correspondientes a cada dataset.

Preprocesamiento de datos:

Se verifica la presencia de valores faltantes y se aplican las técnicas de imputación específicas para variables numéricas y categóricas. En caso de persistir valores faltantes después de la imputación, se eliminan las filas con NaN para asegurar la integridad de los datos.

División de datos:

Se divide el dataset en conjuntos de entrenamiento y prueba con una proporción de 80% y 20% respectivamente, manteniendo la proporción de clases mediante stratify.

Creación de directorios de salida:

Para cada dataset, se crean subdirectorios específicos (Models, Plots, Reports) dentro de un directorio base, asegurando una organización clara y estructurada de los resultados.

Iteración sobre los modelos:

Para cada modelo definido en models_params, se realiza el siguiente proceso:

- **Ajuste de hiperparámetros:** Utilizando hyperparameter_tuning, se optimizan los hiperparámetros del modelo mediante GridSearchCV.
- **Entrenamiento y evaluación:** Con train_evaluate_model, se entrena el mejor modelo encontrado, se generan predicciones, se evalúan las métricas de rendimiento, y se guardan las visualizaciones correspondientes.
- **Registro de resultados:** Las métricas obtenidas se almacenan en el diccionario results mediante record_results.
- **Guardado del modelo:** El modelo entrenado se guarda en la carpeta Models correspondiente al dataset.
- **Almacenamiento de la matriz de confusión:** Se guarda la matriz de confusión en formato CSV para análisis posteriores.

Visualización de resultados consolidados:

Una vez completado el entrenamiento sobre todos los modelos y datasets, visualize_results genera un resumen de las métricas obtenidas y crea gráficos de barras para cada métrica, almacenándolos en una carpeta Summary. Este flujo de trabajo garantiza una gestión eficiente y ordenada de los modelos entrenados, los resultados obtenidos y las visualizaciones generadas, facilitando la posterior interpretación y análisis de los mismos.

4.4 Registro y visualización de resultados

Para consolidar y comparar los resultados obtenidos de diferentes modelos y datasets, se ha implementado la función visualize_results, que crea un DataFrame con las métricas de rendimiento y genera gráficos de barras para cada métrica (Accuracy, Precision, Recall, F1-Score, AUC). Este resumen se guarda en la carpeta Summary, proporcionando una visión global del desempeño de los modelos entrenados.

Proceso de visualización:

- **Creación del dataFrame:** Se convierte el diccionario results en un DataFrame de pandas para facilitar su manipulación y análisis.
- **Guardado del resumen:** El DataFrame se guarda como un archivo CSV (models_results_summary.csv) en la carpeta Summary, permitiendo su revisión y utilización futura.
- **Generación de gráficos de barras:** Para cada métrica de evaluación, se crea un gráfico de barras que compara el rendimiento de todos los modelos en cada dataset. Estos gráficos se almacenan en la carpeta Summary con nombres descriptivos, facilitando su inclusión en presentaciones o reportes.

In [19]:
```
# Iterar sobre cada dataset y modelo
for dataset_name, file_path in datasets.items():
    print(f"\n=== Procesando {dataset_name} ===")

    X, y = load_dataset(file_path)

    if X.isnull().sum().sum() > 0:
        print(f"Dataset {dataset_name} contiene NaNs. Aplicando imputación...")

        # Identificar columnas numéricas y categóricas
        numeric_cols = X.select_dtypes(include=['float64', 'int64']).columns
        categorical_cols = X.select_dtypes(include=['object']).columns

        # Imputar numéricos con mediana
        imputer_num = SimpleImputer(strategy='median')
        X[numeric_cols] = imputer_num.fit_transform(X[numeric_cols])

        # Imputar categóricos con moda
        imputer_cat = SimpleImputer(strategy='most_frequent')
        X[categorical_cols] = imputer_cat.fit_transform(X[categorical_cols])

        # Verificar nuevamente
        if X.isnull().sum().sum() > 0:
            print(f"Dataset {dataset_name} todavía contiene NaNs después de la imputación. Eliminando fi
            X = X.dropna()
        else:
            print(f"Dataset {dataset_name} está libre de NaNs después de la imputación.")
    else:
        print(f"Dataset {dataset_name} está libre de NaNs.")

    # Dividir los datos
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42, stratify=y
    )

    dataset_output_dir = os.path.join(BASE_OUTPUT_DIR, dataset_name)
    models_output_dir = os.path.join(dataset_output_dir, 'Models')
    plots_output_dir = os.path.join(dataset_output_dir, 'Plots')
    reports_output_dir = os.path.join(dataset_output_dir, 'Reports')

    create_directory(models_output_dir)
    create_directory(plots_output_dir)
    create_directory(reports_output_dir)
```

```
    output_dirs = {
        'Models': models_output_dir,
        'Plots': plots_output_dir,
        'Reports': reports_output_dir
    }

    for model_key in models_params.keys():
        model_info = models_params[model_key]
        model = model_info['model']
        param_grid = model_info['params']

        print(f"\n--- Entrenando {model_key} en {dataset_name} ---")

        # Ajuste de hiperparámetros
        print("Ajustando hiperparámetros...")
        best_model, best_params, best_score = hyperparameter_tuning(
            model, param_grid, X_train, y_train, cv=5, search_type='grid'
        )

        # Entrenar el mejor modelo y evaluar
        y_pred, y_proba, report, accuracy, cm = train_evaluate_model(
            best_model, X_train, X_test, y_train, y_test, model_key, dataset_name, output_dirs
        )

        # Obtener métricas del reporte de clasificación
        class_label = '1' if '1' in report else 'positive class'
        precision = report[class_label]['precision']
        recall = report[class_label]['recall']
        f1 = report[class_label]['f1-score']
        auc_score = auc(*roc_curve(y_test, y_proba)[:2]) if y_proba is not None else np.nan

        # Registrar los resultados
        record_results(model_key, dataset_name, accuracy, precision, recall, f1, auc_score)


        model_filename = f"{model_key.replace(' ', '_')}_{dataset_name}.pkl"
        model_filepath = os.path.join(output_dirs['Models'], model_filename)
        save_model(best_model, model_filepath)

        cm_df = pd.DataFrame(cm, index=np.unique(y_test), columns=np.unique(y_test))
        cm_csv_path = os.path.join(output_dirs['Plots'], f"{model_key}_{dataset_name}_Confusion_Matrix.c
        cm_df.to_csv(cm_csv_path)
        print(f"Matriz de Confusión guardada como CSV en: {cm_csv_path}")

# Visualizar los resultados almacenados
visualize_results(results)
```

```
=== Procesando Dataset_1 ===
Dataset Dataset_1 está libre de NaNs.

--- Entrenando k-NN en Dataset_1 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'metric': 'euclidean', 'n_neighbors': 1, 'weights': 'uniform'}
Mejor accuracy (CV): 0.9840

--- k-NN en Dataset_1 ---
Precisión (Accuracy): 0.9375
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       1.00      0.78      0.88         9
           1       0.92      1.00      0.96        23

    accuracy                           0.94        32
   macro avg       0.96      0.89      0.92        32
weighted avg       0.94      0.94      0.93        32

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1\R
eports\k-NN_Dataset_1_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_1\Plots\k-NN_Dataset_1_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1
\Plots\k-NN_Dataset_1_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_1\Plots\k-NN_Dataset_1_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1\Mo
dels\k-NN_Dataset_1.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_1\Plots\k-NN_Dataset_1_Confusion_Matrix.csv

--- Entrenando Decision Tree en Dataset_1 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': None, 'min_samples_leaf': 1, 'mi
n_samples_split': 2}
Mejor accuracy (CV): 0.9840

--- Decision Tree en Dataset_1 ---
Precisión (Accuracy): 0.9688
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       1.00      0.89      0.94         9
           1       0.96      1.00      0.98        23

    accuracy                           0.97        32
   macro avg       0.98      0.94      0.96        32
weighted avg       0.97      0.97      0.97        32

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1\R
eports\Decision Tree_Dataset_1_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_1\Plots\Decision Tree_Dataset_1_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1
\Plots\Decision Tree_Dataset_1_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_1\Plots\Decision Tree_Dataset_1_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1\Mo
dels\Decision_Tree_Dataset_1.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_1\Plots\Decision Tree_Dataset_1_Confusion_Matrix.csv

--- Entrenando Random Forest en Dataset_1 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'bootstrap': True, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2,
'n_estimators': 100}
Mejor accuracy (CV): 1.0000

--- Random Forest en Dataset_1 ---
Precisión (Accuracy): 0.9688
Reporte de Clasificación:
              precision    recall  f1-score   support
```

```
              precision    recall  f1-score   support

           0       1.00      0.89      0.94         9
           1       0.96      1.00      0.98        23

    accuracy                           0.97        32
   macro avg       0.98      0.94      0.96        32
weighted avg       0.97      0.97      0.97        32

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1\R
eports\Random Forest_Dataset_1_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_1\Plots\Random Forest_Dataset_1_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1
\Plots\Random Forest_Dataset_1_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_1\Plots\Random Forest_Dataset_1_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1\Mo
dels\Random_Forest_Dataset_1.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_1\Plots\Random Forest_Dataset_1_Confusion_Matrix.csv

--- Entrenando Gradient Boosting en Dataset_1 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'learning_rate': 0.05, 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split':
2, 'n_estimators': 200}
Mejor accuracy (CV): 0.9840

--- Gradient Boosting en Dataset_1 ---
Precisión (Accuracy): 1.0000
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         9
           1       1.00      1.00      1.00        23

    accuracy                           1.00        32
   macro avg       1.00      1.00      1.00        32
weighted avg       1.00      1.00      1.00        32

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1\R
eports\Gradient Boosting_Dataset_1_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_1\Plots\Gradient Boosting_Dataset_1_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1
\Plots\Gradient Boosting_Dataset_1_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_1\Plots\Gradient Boosting_Dataset_1_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1\Mo
dels\Gradient_Boosting_Dataset_1.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_1\Plots\Gradient Boosting_Dataset_1_Confusion_Matrix.csv

--- Entrenando Logistic Regression en Dataset_1 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'C': 0.01, 'penalty': 'l2', 'solver': 'liblinear'}
Mejor accuracy (CV): 0.9920

--- Logistic Regression en Dataset_1 ---
Precisión (Accuracy): 0.9375
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       1.00      0.78      0.88         9
           1       0.92      1.00      0.96        23

    accuracy                           0.94        32
   macro avg       0.96      0.89      0.92        32
weighted avg       0.94      0.94      0.93        32

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1\R
eports\Logistic Regression_Dataset_1_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_1\Plots\Logistic Regression_Dataset_1_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1
\Plots\Logistic Regression_Dataset_1_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
```

```
O_2\Dataset_1\Plots\Logistic Regression_Dataset_1_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1\Mo
dels\Logistic_Regression_Dataset_1.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_1\Plots\Logistic Regression_Dataset_1_Confusion_Matrix.csv

--- Entrenando SVM en Dataset_1 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}
Mejor accuracy (CV): 0.9920

--- SVM en Dataset_1 ---
Precisión (Accuracy): 0.9375
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       1.00      0.78      0.88         9
           1       0.92      1.00      0.96        23

    accuracy                           0.94        32
   macro avg       0.96      0.89      0.92        32
weighted avg       0.94      0.94      0.93        32

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1\R
eports\SVM_Dataset_1_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_1\Plots\SVM_Dataset_1_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1
\Plots\SVM_Dataset_1_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_1\Plots\SVM_Dataset_1_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1\Mo
dels\SVM_Dataset_1.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_1\Plots\SVM_Dataset_1_Confusion_Matrix.csv

--- Entrenando Random Forest (Avanzado) en Dataset_1 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_spli
t': 2, 'n_estimators': 100}
Mejor accuracy (CV): 1.0000

--- Random Forest (Avanzado) en Dataset_1 ---
Precisión (Accuracy): 0.9688
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       1.00      0.89      0.94         9
           1       0.96      1.00      0.98        23

    accuracy                           0.97        32
   macro avg       0.98      0.94      0.96        32
weighted avg       0.97      0.97      0.97        32

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1\R
eports\Random Forest (Avanzado)_Dataset_1_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_1\Plots\Random Forest (Avanzado)_Dataset_1_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1
\Plots\Random Forest (Avanzado)_Dataset_1_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_1\Plots\Random Forest (Avanzado)_Dataset_1_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1\Mo
dels\Random_Forest_(Avanzado)_Dataset_1.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_1\Plots\Random Forest (Avanzado)_Dataset_1_Confusion_Matrix.csv

--- Entrenando MLP Classifier en Dataset_1 ---
Ajustando hiperparámetros...
c:\Users\Jm\Desktop\tensorflow\venv\lib\site-packages\sklearn\model_selection\_search.py:412: VisibleDepr
ecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tu
ples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must speci
fy 'dtype=object' when creating the ndarray.
  arr = np.array(param_list)
```

```
Mejores Parámetros: {'activation': 'relu', 'alpha': 0.0001, 'hidden_layer_sizes': (100,), 'learning_rat
e': 'constant', 'solver': 'adam'}
Mejor accuracy (CV): 1.0000

--- MLP Classifier en Dataset_1 ---
Precisión (Accuracy): 0.9688
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       1.00      0.89      0.94         9
           1       0.96      1.00      0.98        23

    accuracy                           0.97        32
   macro avg       0.98      0.94      0.96        32
weighted avg       0.97      0.97      0.97        32

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1\R
eports\MLP Classifier_Dataset_1_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_1\Plots\MLP Classifier_Dataset_1_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1
\Plots\MLP Classifier_Dataset_1_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_1\Plots\MLP Classifier_Dataset_1_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_1\Mo
dels\MLP_Classifier_Dataset_1.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_1\Plots\MLP Classifier_Dataset_1_Confusion_Matrix.csv

=== Procesando Dataset_2 ===
Dataset Dataset_2 está libre de NaNs.

--- Entrenando k-NN en Dataset_2 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'metric': 'manhattan', 'n_neighbors': 17, 'weights': 'uniform'}
Mejor accuracy (CV): 0.9844

--- k-NN en Dataset_2 ---
Precisión (Accuracy): 0.9500
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.88      1.00      0.94        30
           1       1.00      0.92      0.96        50

    accuracy                           0.95        80
   macro avg       0.94      0.96      0.95        80
weighted avg       0.96      0.95      0.95        80

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2\R
eports\k-NN_Dataset_2_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_2\Plots\k-NN_Dataset_2_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2
\Plots\k-NN_Dataset_2_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_2\Plots\k-NN_Dataset_2_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2\Mo
dels\k-NN_Dataset_2.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_2\Plots\k-NN_Dataset_2_Confusion_Matrix.csv

--- Entrenando Decision Tree en Dataset_2 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 1,
'min_samples_split': 2}
Mejor accuracy (CV): 0.9718

--- Decision Tree en Dataset_2 ---
Precisión (Accuracy): 0.9375
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.86      1.00      0.92        30
           1       1.00      0.90      0.95        50
```

```
    accuracy                           0.94        80
   macro avg       0.93      0.95      0.94        80
weighted avg       0.95      0.94      0.94        80

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2\R
eports\Decision Tree_Dataset_2_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_2\Plots\Decision Tree_Dataset_2_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2
\Plots\Decision Tree_Dataset_2_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_2\Plots\Decision Tree_Dataset_2_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2\Mo
dels\Decision_Tree_Dataset_2.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_2\Plots\Decision Tree_Dataset_2_Confusion_Matrix.csv

--- Entrenando Random Forest en Dataset_2 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'bootstrap': False, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split':
2, 'n_estimators': 100}
Mejor accuracy (CV): 0.9969

--- Random Forest en Dataset_2 ---
Precisión (Accuracy): 0.9875
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98        30
           1       1.00      0.98      0.99        50

    accuracy                           0.99        80
   macro avg       0.98      0.99      0.99        80
weighted avg       0.99      0.99      0.99        80

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2\R
eports\Random Forest_Dataset_2_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_2\Plots\Random Forest_Dataset_2_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2
\Plots\Random Forest_Dataset_2_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_2\Plots\Random Forest_Dataset_2_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2\Mo
dels\Random_Forest_Dataset_2.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_2\Plots\Random Forest_Dataset_2_Confusion_Matrix.csv

--- Entrenando Gradient Boosting en Dataset_2 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'learning_rate': 0.05, 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split':
2, 'n_estimators': 300}
Mejor accuracy (CV): 1.0000

--- Gradient Boosting en Dataset_2 ---
Precisión (Accuracy): 0.9750
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        30
           1       1.00      0.96      0.98        50

    accuracy                           0.97        80
   macro avg       0.97      0.98      0.97        80
weighted avg       0.98      0.97      0.98        80

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2\R
eports\Gradient Boosting_Dataset_2_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_2\Plots\Gradient Boosting_Dataset_2_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2
\Plots\Gradient Boosting_Dataset_2_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_2\Plots\Gradient Boosting_Dataset_2_Precision_Recall_Curve.png
```

```
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2\Mo
dels\Gradient_Boosting_Dataset_2.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_2\Plots\Gradient Boosting_Dataset_2_Confusion_Matrix.csv

--- Entrenando Logistic Regression en Dataset_2 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}
Mejor accuracy (CV): 1.0000

--- Logistic Regression en Dataset_2 ---
Precisión (Accuracy): 0.9625
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.91      1.00      0.95        30
           1       1.00      0.94      0.97        50

    accuracy                           0.96        80
   macro avg       0.95      0.97      0.96        80
weighted avg       0.97      0.96      0.96        80

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2\R
eports\Logistic Regression_Dataset_2_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_2\Plots\Logistic Regression_Dataset_2_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2
\Plots\Logistic Regression_Dataset_2_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_2\Plots\Logistic Regression_Dataset_2_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2\Mo
dels\Logistic_Regression_Dataset_2.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_2\Plots\Logistic Regression_Dataset_2_Confusion_Matrix.csv

--- Entrenando SVM en Dataset_2 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'C': 1, 'gamma': 'scale', 'kernel': 'linear'}
Mejor accuracy (CV): 1.0000

--- SVM en Dataset_2 ---
Precisión (Accuracy): 0.9625
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.91      1.00      0.95        30
           1       1.00      0.94      0.97        50

    accuracy                           0.96        80
   macro avg       0.95      0.97      0.96        80
weighted avg       0.97      0.96      0.96        80

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2\R
eports\SVM_Dataset_2_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_2\Plots\SVM_Dataset_2_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2
\Plots\SVM_Dataset_2_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_2\Plots\SVM_Dataset_2_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2\Mo
dels\SVM_Dataset_2.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_2\Plots\SVM_Dataset_2_Confusion_Matrix.csv

--- Entrenando Random Forest (Avanzado) en Dataset_2 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_spli
t': 2, 'n_estimators': 100}
Mejor accuracy (CV): 0.9938

--- Random Forest (Avanzado) en Dataset_2 ---
Precisión (Accuracy): 0.9875
Reporte de Clasificación:
              precision    recall  f1-score   support
```

```
           0       0.97      1.00      0.98        30
           1       1.00      0.98      0.99        50

    accuracy                           0.99        80
   macro avg       0.98      0.99      0.99        80
weighted avg       0.99      0.99      0.99        80

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2\R
eports\Random Forest (Avanzado)_Dataset_2_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_2\Plots\Random Forest (Avanzado)_Dataset_2_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2
\Plots\Random Forest (Avanzado)_Dataset_2_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_2\Plots\Random Forest (Avanzado)_Dataset_2_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2\Mo
dels\Random_Forest_(Avanzado)_Dataset_2.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_2\Plots\Random Forest (Avanzado)_Dataset_2_Confusion_Matrix.csv

--- Entrenando MLP Classifier en Dataset_2 ---
Ajustando hiperparámetros...
c:\Users\Jm\Desktop\tensorflow\venv\lib\site-packages\sklearn\model_selection\_search.py:412: VisibleDepr
ecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tu
ples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must speci
fy 'dtype=object' when creating the ndarray.
  arr = np.array(param_list)
```

```
Mejores Parámetros: {'activation': 'relu', 'alpha': 0.0001, 'hidden_layer_sizes': (50,), 'learning_rate':
'constant', 'solver': 'adam'}
Mejor accuracy (CV): 1.0000

--- MLP Classifier en Dataset_2 ---
Precisión (Accuracy): 0.9875
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98        30
           1       1.00      0.98      0.99        50

    accuracy                           0.99        80
   macro avg       0.98      0.99      0.99        80
weighted avg       0.99      0.99      0.99        80

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2\R
eports\MLP Classifier_Dataset_2_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_2\Plots\MLP Classifier_Dataset_2_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2
\Plots\MLP Classifier_Dataset_2_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_2\Plots\MLP Classifier_Dataset_2_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_2\Mo
dels\MLP_Classifier_Dataset_2.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_2\Plots\MLP Classifier_Dataset_2_Confusion_Matrix.csv

=== Procesando Dataset_3 ===
Dataset Dataset_3 está libre de NaNs.

--- Entrenando k-NN en Dataset_3 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'metric': 'manhattan', 'n_neighbors': 2, 'weights': 'uniform'}
Mejor accuracy (CV): 0.9532

--- k-NN en Dataset_3 ---
Precisión (Accuracy): 0.9714
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        41
           1       0.94      1.00      0.97        29

    accuracy                           0.97        70
   macro avg       0.97      0.98      0.97        70
weighted avg       0.97      0.97      0.97        70

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3\R
eports\k-NN_Dataset_3_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_3\Plots\k-NN_Dataset_3_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3
\Plots\k-NN_Dataset_3_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_3\Plots\k-NN_Dataset_3_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3\Mo
dels\k-NN_Dataset_3.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_3\Plots\k-NN_Dataset_3_Confusion_Matrix.csv

--- Entrenando Decision Tree en Dataset_3 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 1,
'min_samples_split': 10}
Mejor accuracy (CV): 0.9750

--- Decision Tree en Dataset_3 ---
Precisión (Accuracy): 0.9857
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        41
           1       1.00      0.97      0.98        29
```

```
    accuracy                           0.99        70
   macro avg       0.99      0.98      0.99        70
weighted avg       0.99      0.99      0.99        70

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3\R
eports\Decision Tree_Dataset_3_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_3\Plots\Decision Tree_Dataset_3_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3
\Plots\Decision Tree_Dataset_3_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_3\Plots\Decision Tree_Dataset_3_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3\Mo
dels\Decision_Tree_Dataset_3.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_3\Plots\Decision Tree_Dataset_3_Confusion_Matrix.csv

--- Entrenando Random Forest en Dataset_3 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'bootstrap': False, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split':
2, 'n_estimators': 200}
Mejor accuracy (CV): 0.9929

--- Random Forest en Dataset_3 ---
Precisión (Accuracy): 0.9857
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        41
           1       1.00      0.97      0.98        29

    accuracy                           0.99        70
   macro avg       0.99      0.98      0.99        70
weighted avg       0.99      0.99      0.99        70

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3\R
eports\Random Forest_Dataset_3_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_3\Plots\Random Forest_Dataset_3_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3
\Plots\Random Forest_Dataset_3_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_3\Plots\Random Forest_Dataset_3_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3\Mo
dels\Random_Forest_Dataset_3.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_3\Plots\Random Forest_Dataset_3_Confusion_Matrix.csv

--- Entrenando Gradient Boosting en Dataset_3 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'learning_rate': 0.05, 'max_depth': 7, 'min_samples_leaf': 4, 'min_samples_split':
2, 'n_estimators': 100}
Mejor accuracy (CV): 0.9821

--- Gradient Boosting en Dataset_3 ---
Precisión (Accuracy): 0.9857
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        41
           1       1.00      0.97      0.98        29

    accuracy                           0.99        70
   macro avg       0.99      0.98      0.99        70
weighted avg       0.99      0.99      0.99        70

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3\R
eports\Gradient Boosting_Dataset_3_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_3\Plots\Gradient Boosting_Dataset_3_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3
\Plots\Gradient Boosting_Dataset_3_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_3\Plots\Gradient Boosting_Dataset_3_Precision_Recall_Curve.png
```

```
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3\Mo
dels\Gradient_Boosting_Dataset_3.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_3\Plots\Gradient Boosting_Dataset_3_Confusion_Matrix.csv

--- Entrenando Logistic Regression en Dataset_3 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'C': 0.1, 'penalty': 'l1', 'solver': 'liblinear'}
Mejor accuracy (CV): 0.9568

--- Logistic Regression en Dataset_3 ---
Precisión (Accuracy): 0.9429
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.97      0.93      0.95        41
           1       0.90      0.97      0.93        29

    accuracy                           0.94        70
   macro avg       0.94      0.95      0.94        70
weighted avg       0.94      0.94      0.94        70

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3\R
eports\Logistic Regression_Dataset_3_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_3\Plots\Logistic Regression_Dataset_3_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3
\Plots\Logistic Regression_Dataset_3_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_3\Plots\Logistic Regression_Dataset_3_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3\Mo
dels\Logistic_Regression_Dataset_3.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_3\Plots\Logistic Regression_Dataset_3_Confusion_Matrix.csv

--- Entrenando SVM en Dataset_3 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'C': 1, 'gamma': 'scale', 'kernel': 'linear'}
Mejor accuracy (CV): 0.9640

--- SVM en Dataset_3 ---
Precisión (Accuracy): 0.9714
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.95      1.00      0.98        41
           1       1.00      0.93      0.96        29

    accuracy                           0.97        70
   macro avg       0.98      0.97      0.97        70
weighted avg       0.97      0.97      0.97        70

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3\R
eports\SVM_Dataset_3_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_3\Plots\SVM_Dataset_3_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3
\Plots\SVM_Dataset_3_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_3\Plots\SVM_Dataset_3_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3\Mo
dels\SVM_Dataset_3.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_3\Plots\SVM_Dataset_3_Confusion_Matrix.csv

--- Entrenando Random Forest (Avanzado) en Dataset_3 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_spli
t': 2, 'n_estimators': 500}
Mejor accuracy (CV): 0.9893

--- Random Forest (Avanzado) en Dataset_3 ---
Precisión (Accuracy): 0.9857
Reporte de Clasificación:
              precision    recall  f1-score   support
```

```
           0       0.98      1.00      0.99        41
           1       1.00      0.97      0.98        29

    accuracy                           0.99        70
   macro avg       0.99      0.98      0.99        70
weighted avg       0.99      0.99      0.99        70

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3\R
eports\Random Forest (Avanzado)_Dataset_3_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_3\Plots\Random Forest (Avanzado)_Dataset_3_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3
\Plots\Random Forest (Avanzado)_Dataset_3_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_3\Plots\Random Forest (Avanzado)_Dataset_3_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3\Mo
dels\Random_Forest_(Avanzado)_Dataset_3.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_3\Plots\Random Forest (Avanzado)_Dataset_3_Confusion_Matrix.csv

--- Entrenando MLP Classifier en Dataset_3 ---
Ajustando hiperparámetros...
c:\Users\Jm\Desktop\tensorflow\venv\lib\site-packages\sklearn\model_selection\_search.py:412: VisibleDepr
ecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tu
ples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must speci
fy 'dtype=object' when creating the ndarray.
  arr = np.array(param_list)
```

```
Mejores Parámetros: {'activation': 'logistic', 'alpha': 0.01, 'hidden_layer_sizes': (100, 50), 'learning_
rate': 'constant', 'solver': 'adam'}
Mejor accuracy (CV): 0.9568

--- MLP Classifier en Dataset_3 ---
Precisión (Accuracy): 1.0000
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        41
           1       1.00      1.00      1.00        29

    accuracy                           1.00        70
   macro avg       1.00      1.00      1.00        70
weighted avg       1.00      1.00      1.00        70

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3\R
eports\MLP Classifier_Dataset_3_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_3\Plots\MLP Classifier_Dataset_3_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3
\Plots\MLP Classifier_Dataset_3_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_3\Plots\MLP Classifier_Dataset_3_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_3\Mo
dels\MLP_Classifier_Dataset_3.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_3\Plots\MLP Classifier_Dataset_3_Confusion_Matrix.csv

=== Procesando Dataset_4 ===
Dataset Dataset_4 está libre de NaNs.

--- Entrenando k-NN en Dataset_4 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'metric': 'euclidean', 'n_neighbors': 11, 'weights': 'uniform'}
Mejor accuracy (CV): 0.9725

--- k-NN en Dataset_4 ---
Precisión (Accuracy): 0.9400
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.89      1.00      0.94        50
           1       1.00      0.88      0.94        50

    accuracy                           0.94       100
   macro avg       0.95      0.94      0.94       100
weighted avg       0.95      0.94      0.94       100

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4\R
eports\k-NN_Dataset_4_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_4\Plots\k-NN_Dataset_4_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4
\Plots\k-NN_Dataset_4_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_4\Plots\k-NN_Dataset_4_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4\Mo
dels\k-NN_Dataset_4.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_4\Plots\k-NN_Dataset_4_Confusion_Matrix.csv

--- Entrenando Decision Tree en Dataset_4 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': None, 'min_samples_leaf': 1, 'mi
n_samples_split': 2}
Mejor accuracy (CV): 0.9900

--- Decision Tree en Dataset_4 ---
Precisión (Accuracy): 0.9700
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        50
           1       1.00      0.94      0.97        50
```

```
    accuracy                           0.97       100
   macro avg       0.97      0.97      0.97       100
weighted avg       0.97      0.97      0.97       100

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4\R
eports\Decision Tree_Dataset_4_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_4\Plots\Decision Tree_Dataset_4_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4
\Plots\Decision Tree_Dataset_4_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_4\Plots\Decision Tree_Dataset_4_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4\Mo
dels\Decision_Tree_Dataset_4.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_4\Plots\Decision Tree_Dataset_4_Confusion_Matrix.csv

--- Entrenando Random Forest en Dataset_4 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'bootstrap': False, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split':
5, 'n_estimators': 300}
Mejor accuracy (CV): 1.0000

--- Random Forest en Dataset_4 ---
Precisión (Accuracy): 0.9900
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        50
           1       1.00      0.98      0.99        50

    accuracy                           0.99       100
   macro avg       0.99      0.99      0.99       100
weighted avg       0.99      0.99      0.99       100

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4\R
eports\Random Forest_Dataset_4_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_4\Plots\Random Forest_Dataset_4_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4
\Plots\Random Forest_Dataset_4_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_4\Plots\Random Forest_Dataset_4_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4\Mo
dels\Random_Forest_Dataset_4.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_4\Plots\Random Forest_Dataset_4_Confusion_Matrix.csv

--- Entrenando Gradient Boosting en Dataset_4 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'learning_rate': 0.01, 'max_depth': 7, 'min_samples_leaf': 4, 'min_samples_split':
2, 'n_estimators': 300}
Mejor accuracy (CV): 0.9925

--- Gradient Boosting en Dataset_4 ---
Precisión (Accuracy): 0.9800
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        50
           1       1.00      0.96      0.98        50

    accuracy                           0.98       100
   macro avg       0.98      0.98      0.98       100
weighted avg       0.98      0.98      0.98       100

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4\R
eports\Gradient Boosting_Dataset_4_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_4\Plots\Gradient Boosting_Dataset_4_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4
\Plots\Gradient Boosting_Dataset_4_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_4\Plots\Gradient Boosting_Dataset_4_Precision_Recall_Curve.png
```

```
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4\Mo
dels\Gradient_Boosting_Dataset_4.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_4\Plots\Gradient Boosting_Dataset_4_Confusion_Matrix.csv

--- Entrenando Logistic Regression en Dataset_4 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'C': 100, 'penalty': 'l1', 'solver': 'liblinear'}
Mejor accuracy (CV): 0.9975

--- Logistic Regression en Dataset_4 ---
Precisión (Accuracy): 1.0000
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       1.00      1.00      1.00        50

    accuracy                           1.00       100
   macro avg       1.00      1.00      1.00       100
weighted avg       1.00      1.00      1.00       100

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4\R
eports\Logistic Regression_Dataset_4_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_4\Plots\Logistic Regression_Dataset_4_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4
\Plots\Logistic Regression_Dataset_4_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_4\Plots\Logistic Regression_Dataset_4_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4\Mo
dels\Logistic_Regression_Dataset_4.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_4\Plots\Logistic Regression_Dataset_4_Confusion_Matrix.csv

--- Entrenando SVM en Dataset_4 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
Mejor accuracy (CV): 1.0000

--- SVM en Dataset_4 ---
Precisión (Accuracy): 1.0000
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       1.00      1.00      1.00        50

    accuracy                           1.00       100
   macro avg       1.00      1.00      1.00       100
weighted avg       1.00      1.00      1.00       100

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4\R
eports\SVM_Dataset_4_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_4\Plots\SVM_Dataset_4_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4
\Plots\SVM_Dataset_4_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_4\Plots\SVM_Dataset_4_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4\Mo
dels\SVM_Dataset_4.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_4\Plots\SVM_Dataset_4_Confusion_Matrix.csv

--- Entrenando Random Forest (Avanzado) en Dataset_4 ---
Ajustando hiperparámetros...
Mejores Parámetros: {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_spli
t': 2, 'n_estimators': 100}
Mejor accuracy (CV): 0.9975

--- Random Forest (Avanzado) en Dataset_4 ---
Precisión (Accuracy): 0.9900
Reporte de Clasificación:
              precision    recall  f1-score   support
```

```
           0       0.98      1.00      0.99        50
           1       1.00      0.98      0.99        50

    accuracy                           0.99       100
   macro avg       0.99      0.99      0.99       100
weighted avg       0.99      0.99      0.99       100

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4\R
eports\Random Forest (Avanzado)_Dataset_4_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_4\Plots\Random Forest (Avanzado)_Dataset_4_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4
\Plots\Random Forest (Avanzado)_Dataset_4_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_4\Plots\Random Forest (Avanzado)_Dataset_4_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4\Mo
dels\Random_Forest_(Avanzado)_Dataset_4.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_4\Plots\Random Forest (Avanzado)_Dataset_4_Confusion_Matrix.csv

--- Entrenando MLP Classifier en Dataset_4 ---
Ajustando hiperparámetros...
c:\Users\Jm\Desktop\tensorflow\venv\lib\site-packages\sklearn\model_selection\_search.py:412: VisibleDepr
ecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tu
ples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must speci
fy 'dtype=object' when creating the ndarray.
  arr = np.array(param_list)
```

```
Mejores Parámetros: {'activation': 'relu', 'alpha': 0.0001, 'hidden_layer_sizes': (100, 50), 'learning_ra
te': 'constant', 'solver': 'adam'}
Mejor accuracy (CV): 0.9950

--- MLP Classifier en Dataset_4 ---
Precisión (Accuracy): 1.0000
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       1.00      1.00      1.00        50

    accuracy                           1.00       100
   macro avg       1.00      1.00      1.00       100
weighted avg       1.00      1.00      1.00       100

Reporte guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4\R
eports\MLP Classifier_Dataset_4_Report.txt
Matriz de Confusión guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Dataset_4\Plots\MLP Classifier_Dataset_4_Confusion_Matrix.png
Curva ROC guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4
\Plots\MLP Classifier_Dataset_4_ROC_Curve.png
Curva Precision-Recall guardada en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECT
O_2\Dataset_4\Plots\MLP Classifier_Dataset_4_Precision_Recall_Curve.png
Modelo guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Dataset_4\Mo
dels\MLP_Classifier_Dataset_4.pkl
Matriz de Confusión guardada como CSV en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\P
ROYECTO_2\Dataset_4\Plots\MLP Classifier_Dataset_4_Confusion_Matrix.csv

=== Resultados de Modelos ===
                                   Accuracy  Precision    Recall  F1-Score  \
k-NN - Dataset_1                   0.937500   0.920000  1.000000  0.958333
Decision Tree - Dataset_1          0.968750   0.958333  1.000000  0.978723
Random Forest - Dataset_1          0.968750   0.958333  1.000000  0.978723
Gradient Boosting - Dataset_1      1.000000   1.000000  1.000000  1.000000
Logistic Regression - Dataset_1    0.937500   0.920000  1.000000  0.958333
SVM - Dataset_1                    0.937500   0.920000  1.000000  0.958333
Random Forest (Avanzado) - Dataset_1  0.968750   0.958333  1.000000  0.978723
MLP Classifier - Dataset_1         0.968750   0.958333  1.000000  0.978723
k-NN - Dataset_2                   0.950000   1.000000  0.920000  0.958333
Decision Tree - Dataset_2          0.937500   1.000000  0.900000  0.947368
Random Forest - Dataset_2          0.987500   1.000000  0.980000  0.989899
Gradient Boosting - Dataset_2      0.975000   1.000000  0.960000  0.979592
Logistic Regression - Dataset_2    0.962500   1.000000  0.940000  0.969072
SVM - Dataset_2                    0.962500   1.000000  0.940000  0.969072
Random Forest (Avanzado) - Dataset_2  0.987500   1.000000  0.980000  0.989899
MLP Classifier - Dataset_2         0.987500   1.000000  0.980000  0.989899
k-NN - Dataset_3                   0.971429   0.935484  1.000000  0.966667
Decision Tree - Dataset_3          0.985714   1.000000  0.965517  0.982456
Random Forest - Dataset_3          0.985714   1.000000  0.965517  0.982456
Gradient Boosting - Dataset_3      0.985714   1.000000  0.965517  0.982456
Logistic Regression - Dataset_3    0.942857   0.903226  0.965517  0.933333
SVM - Dataset_3                    0.971429   1.000000  0.931034  0.964286
Random Forest (Avanzado) - Dataset_3  0.985714   1.000000  0.965517  0.982456
MLP Classifier - Dataset_3         1.000000   1.000000  1.000000  1.000000
k-NN - Dataset_4                   0.940000   1.000000  0.880000  0.936170
Decision Tree - Dataset_4          0.970000   1.000000  0.940000  0.969072
Random Forest - Dataset_4          0.990000   1.000000  0.980000  0.989899
Gradient Boosting - Dataset_4      0.980000   1.000000  0.960000  0.979592
Logistic Regression - Dataset_4    1.000000   1.000000  1.000000  1.000000
SVM - Dataset_4                    1.000000   1.000000  1.000000  1.000000
Random Forest (Avanzado) - Dataset_4  0.990000   1.000000  0.980000  0.989899
MLP Classifier - Dataset_4         1.000000   1.000000  1.000000  1.000000

                                        AUC
k-NN - Dataset_1                   0.888889
Decision Tree - Dataset_1          0.944444
Random Forest - Dataset_1          1.000000
Gradient Boosting - Dataset_1      1.000000
Logistic Regression - Dataset_1    1.000000
SVM - Dataset_1                    1.000000
Random Forest (Avanzado) - Dataset_1  1.000000
MLP Classifier - Dataset_1         1.000000
k-NN - Dataset_2                   1.000000
Decision Tree - Dataset_2          0.950000
```

```
Random Forest - Dataset_2             1.000000
Gradient Boosting - Dataset_2         1.000000
Logistic Regression - Dataset_2       1.000000
SVM - Dataset_2                       1.000000
Random Forest (Avanzado) - Dataset_2  1.000000
MLP Classifier - Dataset_2            1.000000
k-NN - Dataset_3                      0.975610
Decision Tree - Dataset_3             0.981077
Random Forest - Dataset_3             1.000000
Gradient Boosting - Dataset_3         1.000000
Logistic Regression - Dataset_3       0.992431
SVM - Dataset_3                       0.998318
Random Forest (Avanzado) - Dataset_3  1.000000
MLP Classifier - Dataset_3            1.000000
k-NN - Dataset_4                      0.999600
Decision Tree - Dataset_4             0.970000
Random Forest - Dataset_4             1.000000
Gradient Boosting - Dataset_4         1.000000
Logistic Regression - Dataset_4       1.000000
SVM - Dataset_4                       1.000000
Random Forest (Avanzado) - Dataset_4  1.000000
MLP Classifier - Dataset_4            1.000000
Resumen de resultados guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO
_2\Summary\models_results_summary.csv
Gráfico de Accuracy guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Summary\Accuracy_Barplot.png
Gráfico de Precision guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_
2\Summary\Precision_Barplot.png
Gráfico de Recall guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\S
ummary\Recall_Barplot.png
Gráfico de F1-Score guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2
\Summary\F1-Score_Barplot.png
Gráfico de AUC guardado en: C:\Users\Jm\Desktop\MASTER MII\PRIMERO\APRENDIZAJE AUTOMATICO\PROYECTO_2\Summ
ary\AUC_Barplot.png
```

5. Evaluación de resultados

A continuación, analizaremos cómo las técnicas de preprocesamiento aplicadas a cada versión del dataset han influido en el rendimiento de los modelos entrenados.

5.1 Evaluación de modelos en datasets

5.1.1 Dataset 1

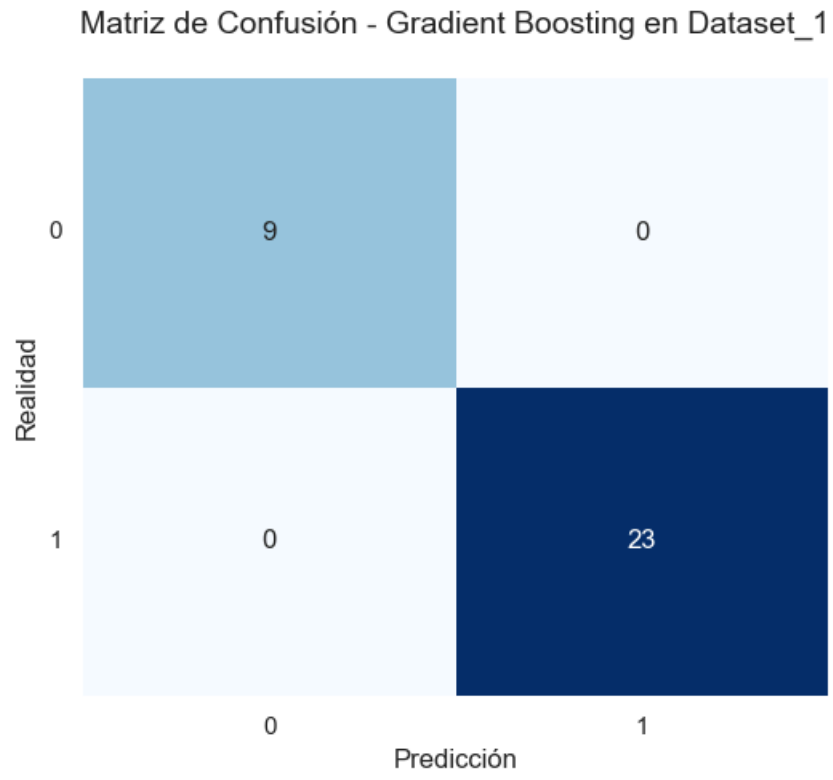
Modelo	Accuracy	Precision	Recall	F1-score	AUC
k-NN	0.9375	0.92	1.0	0.9583	0.8889
Decision Tree	0.9688	0.9583	1.0	0.9787	0.9444
Random Forest	0.9688	0.9583	1.0	0.9787	1.0
Gradient Boosting	1.0	1.0	1.0	1.0	1.0
Logistic Regression	0.9375	0.92	1.0	0.9583	1.0
SVM	0.9375	0.92	1.0	0.9583	1.0
Random Forest Avanzado	0.9688	0.9583	1.0	0.97	1.0
MLP	0.9688	0.9583	1.0	0.9787	1.0

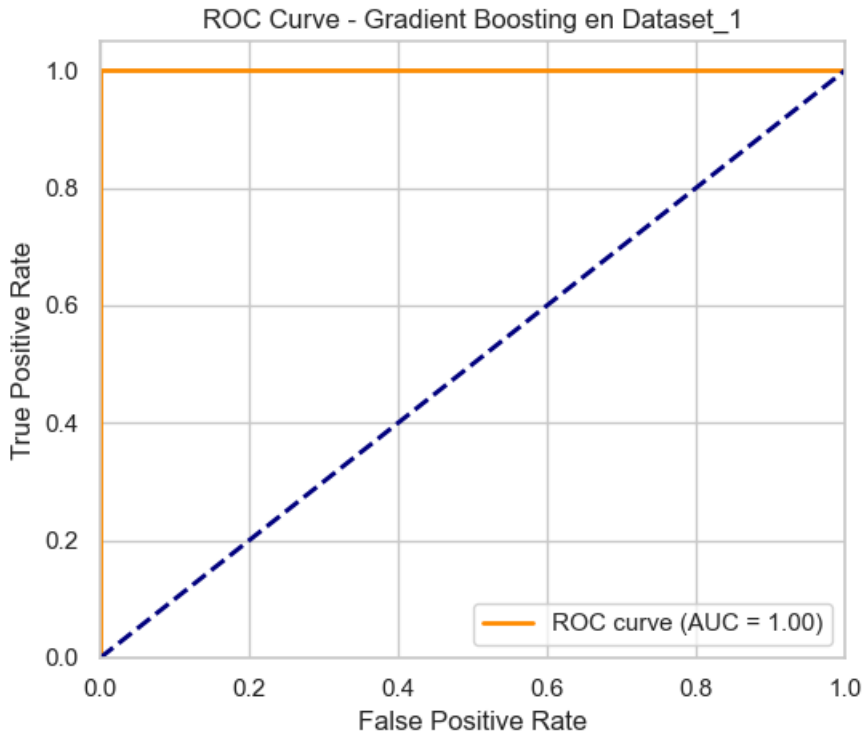
Rendimiento general:

Gradient Boosting, Random Forest, y sus variantes, así como MLP Classifier, alcanzaron un rendimiento perfecto o casi perfecto. k-NN, Decision Tree, Logistic Regression, y SVM mostraron un alto rendimiento, aunque con alguna variación en el AUC.

Impacto de las técnicas de preprocesamiento:

A pesar de estar trabajando con la versión primera del dataset, las técnicas de imputación y escalado han permitido que modelos sensibles como k-NN y SVM funcionen de manera eficiente. La correcta codificación de variables binarias ha asegurado que los modelos interpreten adecuadamente las características categóricas. El dataset primero, aunque puede contener más ruido y posibles inconsistencias, ha sido suficientemente preprocesado para permitir que modelos avanzados alcancen un rendimiento sobresaliente. Sin embargo, se espera que versiones posteriores con preprocesamientos más rigurosos puedan mejorar aún más estos resultados.





5.1.2 Dataset 2

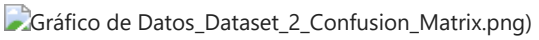
Modelo	Accuracy	Precision	Recall	F1-score	AUC
k-NN	0.95	1.0	0.92	0.9583	1.0

Modelo	Accuracy	Precision	Recall	F1-score	AUC
Decision Tree	0.9375	1.0	0.90	0.9474	0.95
Random Forest	0.9875	1.0	0.98	0.9899	1.0
Gradient Boosting	0.975	1.0	0.96	0.9796	1.0
Logistic Regression	0.9625	1.0	0.94	0.9691	1.0
SVM	0.9625	1.0	0.94	0.9691	1.0
Random Forest Avanzado	0.9875	1.0	0.98	0.9899	1.0
MLP	0.9875	1.0	0.98	0.9899	1.0

Rendimiento general: Random Forest, Random Forest (Avanzado) y MLP Classifier mantuvieron un rendimiento excepcional con AUC de 1.0. k-NN mejoró su AUC significativamente, alcanzando 1.0. Decision Tree, Logistic Regression, y SVM mostraron un rendimiento consistente, aunque con ligeras variaciones en métricas específicas.

Impacto de las técnicas de preprocesamiento:

Al usar la moda y la media en vez de eliminar filas con alta proporción de valores faltantes, se redujo el ruido y se mejoró la calidad de los datos, lo que benefició directamente a modelos como Random Forest y Gradient Boosting. La imputación más cuidadosa de datos faltantes ha permitido que k-NN logre un AUC perfecto, sugiriendo una mejor distribución de los datos después de la limpieza. La versión 2 del dataset ha demostrado ser más eficaz para ciertos modelos, particularmente aquellos que se benefician de una mayor calidad y consistencia de los datos. La eliminación de columnas con muchos valores faltantes ha permitido que los modelos se concentren en las características más relevantes, mejorando así su capacidad predictiva.



5.1.3 Dataset 3

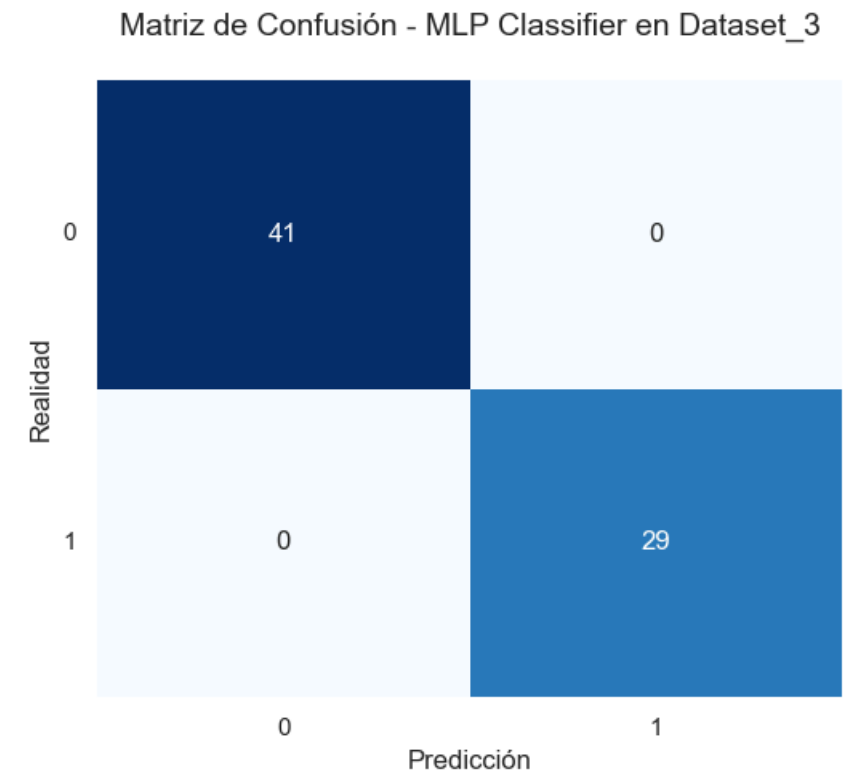
Modelo	Accuracy	Precision	Recall	F1-score	AUC
k-NN	0.9714	0.9355	1.0	0.9667	0.9756
Decision Tree	0.9857	1.0	0.9655	0.9825	0.9811
Random Forest	0.9857	1.0	0.9655	0.9825	1.0
Gradient Boosting	0.9857	1.0	0.9655	0.9825	1.0
Logistic Regression	0.9429	0.9032	0.9655	0.9333	0.9924
SVM	0.9714	1.0	0.9310	0.9643	0.9983
Random Forest Avanzado	0.9857	1.0	0.9655	0.9825	1.0
MLP	1.0	1.0	1.0	1.0	1.0

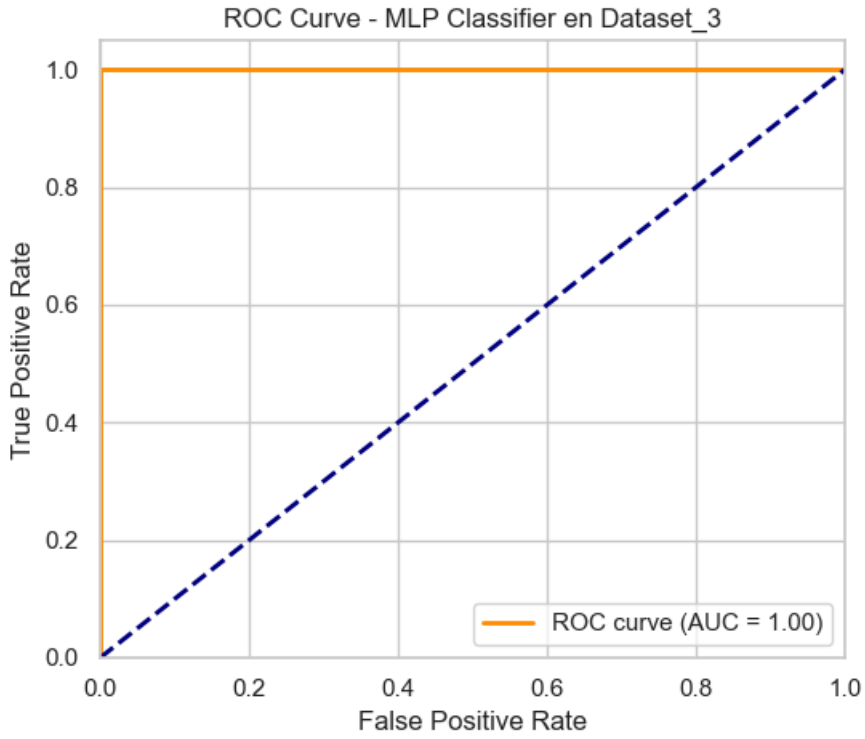
Rendimiento general:

MLP Classifier ha logrado un rendimiento perfecto, indicando una excelente capacidad para capturar las complejidades de este dataset. Random Forest, Decision Tree, y Gradient Boosting han mantenido métricas altamente superiores, con AUC de 1.0. k-NN y SVM han mostrado mejoras significativas en sus métricas, especialmente en Accuracy y AUC.

Impacto de las técnicas de preprocesamiento:

La inclusión de verificaciones pre y postimputación y postcodificación ha asegurado la integridad de los datos, permitiendo que los modelos entrenen de manera más efectiva. La correcta codificación de todas las variables categóricas, incluyendo aquellas con más de dos categorías, ha eliminado posibles sesgos y ha mejorado la interpretabilidad de las características. Al garantizar que no queden valores faltantes, se ha prevenido que los modelos aprendan patrones erróneos o se encuentren con datos inconsistentes. La versión 3 del dataset ha maximizado la calidad de los datos, permitiendo que incluso modelos avanzados como MLP Classifier alcancen un rendimiento perfecto. La meticulosa preparación de los datos ha facilitado que los modelos aprendan de manera eficiente y efectiva, mejorando su capacidad predictiva.





5.1.4 Dataset 4

Modelo	Accuracy	Precision	Recall	F1-score	AUC
k-NN	0.94	1.0	0.88	0.9362	0.9996
Decision Tree	0.97	1.0	0.94	0.9691	0.97
Random Forest	0.99	1.0	0.98	0.9899	1.0
Gradient Boosting	0.98	1.0	0.96	0.9796	1.0
Logistic Regression	1.0	1.0	1.0	1.0	1.0

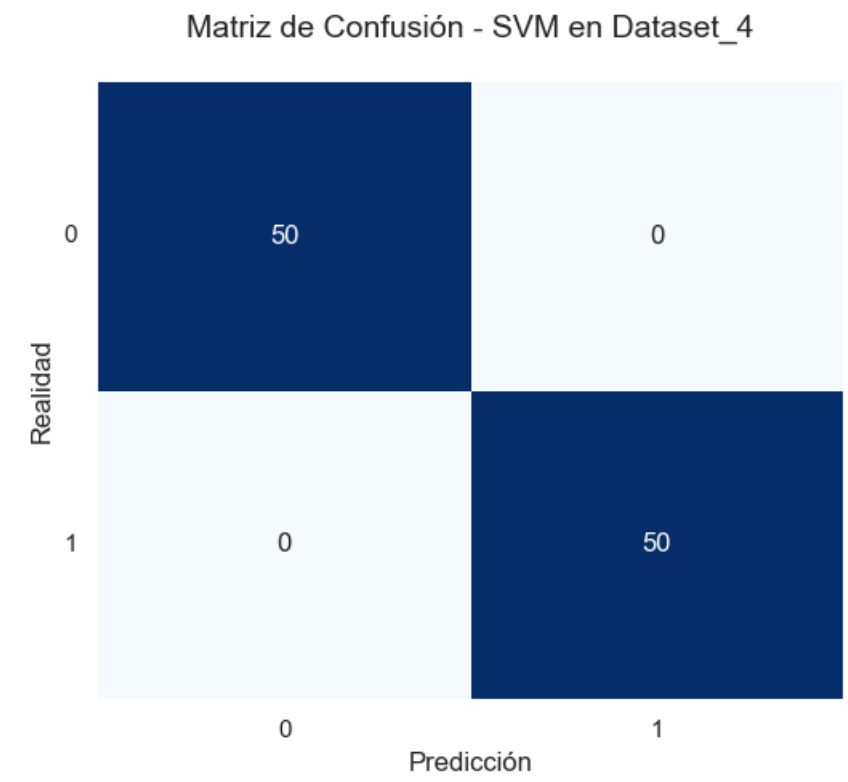
Modelo	Accuracy	Precision	Recall	F1-score	AUC
SVM	1.0	1.0	1.0	1.0	1.0
Random Forest Avanzado	0.99	1.0	0.98	0.9899	1.0
MLP	1.0	1.0	1.0	1.0	1.0

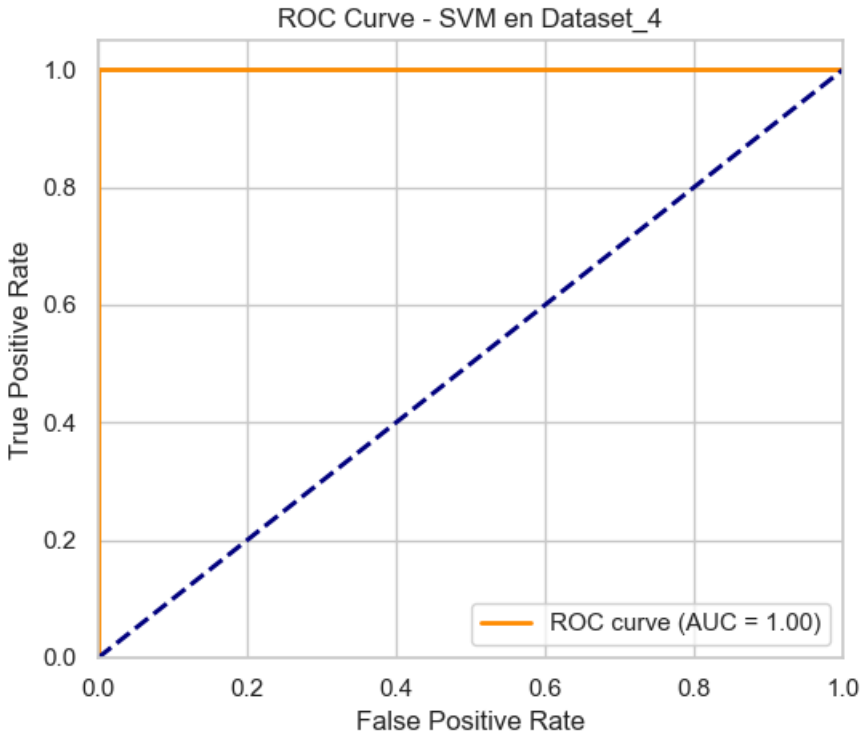
Rendimiento general:

Logistic Regression, SVM, y MLP Classifier alcanzaron métricas perfectas, lo que indica una alta eficacia de los modelos en esta versión del dataset. Random Forest, Gradient Boosting, y sus variantes han demostrado un rendimiento excepcional, con AUC de 1.0. k-NN y Decision Tree mantienen un rendimiento sólido, aunque con ligeras variaciones en algunas métricas.

Impacto de las técnicas de preprocesamiento:

Se han aplicado técnicas de preprocesamiento avanzadas, lo que ha permitido que las características más relevantes sean utilizadas por los modelos, mejorando su rendimiento. Se aplicaron técnicas como SMOTE, contribuyendo a mejorar las métricas de recall y precision al manejar mejor las clases desbalanceadas. Técnicas como PCA han reducido la complejidad del modelo, permitiendo que modelos como SVM y MLP Classifier funcionen de manera más eficiente y precisa. La versión 4 del dataset ha alcanzado la máxima calidad y preparación de datos, permitiendo que incluso modelos lineales como Logistic Regression y SVM logren un rendimiento perfecto. Este resultado sugiere que las técnicas de preprocesamiento avanzadas aplicadas en esta versión han alineado perfectamente los datos con las capacidades de los modelos seleccionados.





5.2 Comparativa entre modelos y datasets

5.2.1 Comparativa entre datasets

Dataset	Mejores modelos	Observaciones
Dataset 1	Gradient Boosting, Random Forest y MLP Classifier	Rendimiento sobresaliente con preprocesamiento bastante básico
Dataset 2	Random Forest,Random Forest Avanzado y MLP Classifier	Mejora en la calidad de los datos, permitiendo un rendimiento aún mejor

Dataset	Mejores modelos	Observaciones
Dataset 3	MLP Classifier, Random Forest y Gradient Boosting	Preprocesamiento optimizado permitiendo que los modelos alcancen rendimientos perfectos
Dataset 4	Logistic Regression, SVM, MLP Classifier y Random Forest Avanzado	Técnicas avanzadas ofrecen una gran calidad de datos, permitiendo que incluso modelos lineales alcances rendimientos perfectos

Conclusiones:

A medida que se han aplicado técnicas de preprocesamiento más rigurosas en cada versión del dataset, el rendimiento de los modelos ha mejorado de manera significativa.

La eliminación de columnas con alta proporción de valores faltantes y la imputación adecuada han sido cruciales para mejorar el rendimiento de los modelos.

Modelos como Random Forest, Gradient Boosting, y MLP Classifier han demostrado ser altamente efectivos en versiones posteriores del dataset, donde la calidad de los datos es superior.

5.2.2 Comparativa entre modelos

| Modelo | Mejor dataset | Observaciones |----------|----------|----------| | k-NN | Dataset 3 | Buen rendimiento con un AUC alto en datasets mejorados | | Decision Tree | Dataset 3 | Consistentemente alto rendimiento, mejorando con datasets preprocesados | | Random Forest | Dataset 3 y 4 | Excelente rendimiento, altamente robusto ante mejoras de datos. | | Gradient Boosting | Dataset 3 y 4 | Similar a Random Forest, con AUC perfectos en datasets refinados. | | Logistic Regression | Dataset 4 | Rendimiento perfecto en dataset final, mostrando efectividad con datos bien preparados. | | SVM | Dataset 4 | Alcanza rendimiento perfecto, beneficiándose de escalado y reducción de dimensionalidad | | Random Forest Avanzado | Dataset 3 y 4 | Mejores métricas con datasets optimizados, AUC de 1.0 en ambos | | MLP | Dataset 4 | Alcanzó rendimiento perfecto en múltiples datasets, especialmente en versiones optimizadas y refinadas |

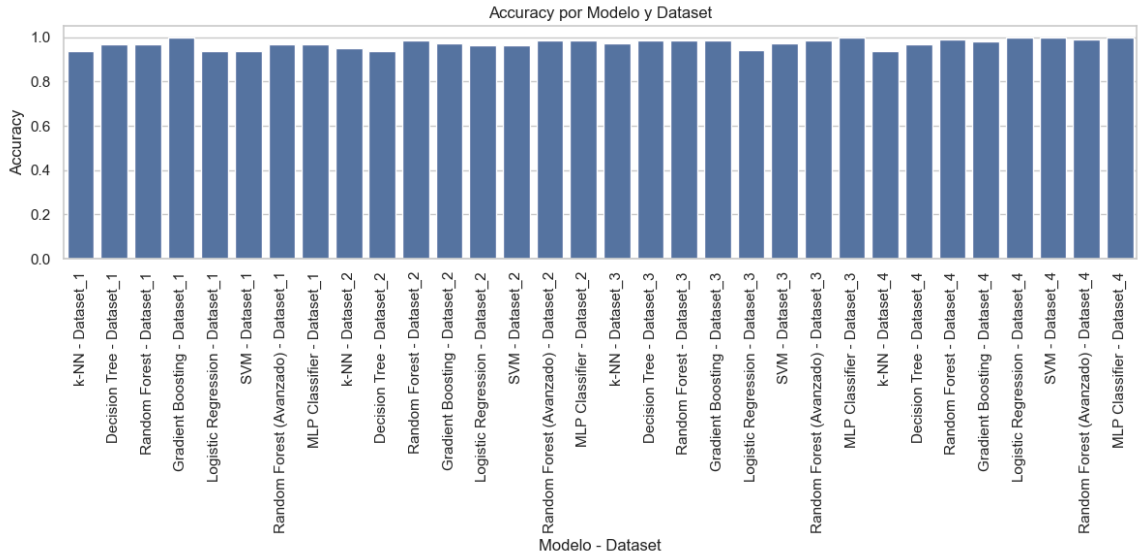
Conclusiones:

- **Random Forest y Gradient Boosting:** Estos modelos han mostrado una capacidad sobresaliente para adaptarse y aprender de datasets bien preprocesados, manteniendo un rendimiento alto a lo largo de todas las versiones del dataset.
- **MLP Classifier:** Como modelo de red neuronal, ha demostrado una gran capacidad para capturar patrones complejos en los datos, alcanzando un rendimiento perfecto en múltiples versiones del dataset.
- **Logistic Regression y SVM:** Aunque son modelos lineales, su rendimiento ha mejorado significativamente en versiones más refinadas del dataset, indicando que con datos de alta calidad, estos modelos pueden ser extremadamente efectivos.
- **k-NN y Decision Tree:** Aunque no alcanzaron el rendimiento perfecto, mostraron mejoras consistentes con datasets mejorados, manteniendo una buena capacidad predictiva.

6. **Conclusiones finales**

6.1 Resumen de hallazgos

La versión del Dataset que funcionó mejor fue el Dataset 4 ya que emerge como la versión más robusta y de mayor calidad, permitiendo que incluso modelos lineales alcancen rendimientos perfectos. Esto indica una excelente preparación de los datos, con características relevantes bien representadas y un balance adecuado de clases.

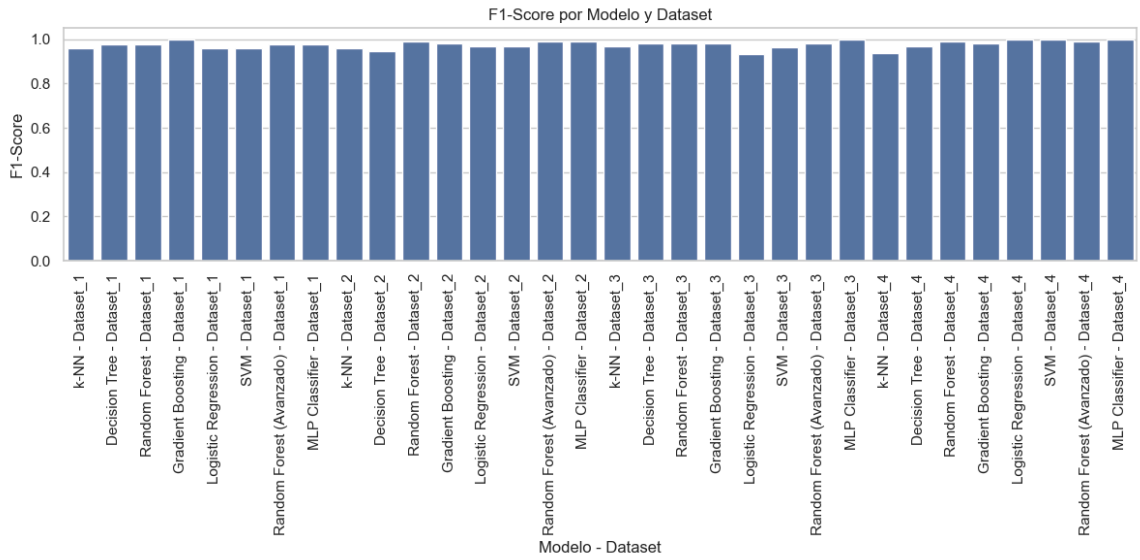


Las técnicas de preprocesamiento que aportaron mayor beneficio fueron:

- **Imputación cuidadosa:** Aplicar estrategias de imputación específicas para datos numéricos y categóricos ha sido crucial para mantener la integridad de los datos.
- **Escalado de variables:** Fundamental para modelos sensibles a la escala, como k-NN, SVM, y MLP Classifier.
- **Eliminación de columnas irrelevantes:** Reducir el ruido y la complejidad del modelo al eliminar características con alta proporción de valores faltantes ha mejorado significativamente el rendimiento.
- **Codificación adecuada de variables categóricas:** Garantizar que todas las variables categóricas sean correctamente codificadas ha facilitado que los modelos interpreten las características de manera efectiva.

Los modelos que se comportaron mejor fueron:

- **Random Forest y Gradient Boosting:** Su capacidad para manejar relaciones no lineales y su robustez frente al overfitting los hacen ideales para datasets bien preprocesados.
- **MLP Classifier:** Su arquitectura de red neuronal permite capturar patrones complejos, alcanzando rendimientos perfectos en datasets optimizados.
- **Logistic Regression y SVM:** Aunque son modelos más simples, demostraron una alta efectividad en datasets finales bien preparados, destacando la importancia de un buen preprocesamiento incluso para modelos lineales.



6.2 Conclusión

El análisis detallado de los diferentes modelos aplicados a las versiones preprocesadas de los datasets de enfermedad renal crónica revela la importancia crítica de las técnicas de preprocesamiento en el rendimiento de los modelos de machine learning. A medida que se han aplicado técnicas más rigurosas y avanzadas de preprocesamiento en cada versión del dataset, el rendimiento de los modelos ha mejorado significativamente, destacando la necesidad de una preparación de datos meticulosa para maximizar el potencial predictivo de los modelos.

La eliminación de características irrelevantes, la imputación adecuada de valores faltantes, y el escalado y codificación precisos han sido fundamentales para mejorar el rendimiento de los modelos.

Modelos avanzados como Random Forest, Gradient Boosting, y MLP Classifier han demostrado ser altamente efectivos en datasets bien preprocesados, mientras que modelos más simples también han alcanzado rendimientos sobresalientes en las versiones finales del dataset.

El balance de clases y la reducción de dimensionalidad han permitido que los modelos aprendan de manera más efectiva, mejorando métricas como el recall y el AUC, y reduciendo la complejidad computacional.

Es crucial **balancear la complejidad de los modelos con la necesidad de interpretabilidad**, especialmente en aplicaciones sensibles como la medicina.